

JO

# Underst

ORANGE

# **Understanding 3D Animation Using Maya**

John Edgar Park

# Understanding 3D Animation Using Maya

John Edgar Park  
Los Angeles, CA  
USA  
john@understandingmaya.com

*Cover illustration:* Superstock

Library of Congress Cataloging-in-Publication Data  
Park, John Edgar, 1972–

Understanding 3D animation using Maya / John Edgar Park.  
p. cm.

Includes bibliographical references and index.

ISBN 0-387-00176-X (softcover : alk. paper)

1. Computer animation. 2. Maya (Computer file) 3. Three-dimensional display systems.

I. Title

TR897.7.P365 2004

006.6'96—dc22

2004049925

ISBN 0-387-00176-X

Printed on acid-free paper.

© 2005 Springer Science+Business Media, Inc.

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, Inc., 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Printed in the United States of America. (MVY)

9 8 7 6 5 4 3 2 1

SPIN 10901045

springeronline.com



*For Erin and Ronan*

---

# Acknowledgments

I must thank the following people. Without their help and support, this book would not have been possible.

The students of Friedman 3D for giving me the reason to write this book and for patiently enduring the ever-changing curriculum.

Eric Huelsman, of Studio Arts, thanks for bringing me to Friedman in the first place.

Wayne Wheeler, Ann Kostant, Wayne Yuhasz, Louise Farkas, and all the people at Springer who brought this book to life.

David Fugate of Waterside Agency for all of your great help and advice.

Danielle Lamothe, LeiLei Sun, and Erica Fyvie at Alias for much assistance.

Sande Scoredos, Don Levy, and Jerry Schmitz for help with the clearances at Sony Pictures Imageworks.

Bill Victor and Chris Endicott for all of your helpful insights during the technical review process.

Alfredo Fajardo for your keen eye for detail during the exhaustive technical edit.

Dave Colman and Marcelo Vignali for your wonderful illustrations.

Phil Pignotti for the terrific drawings, especially the chicken.

Mom and Dad for your love and encouragement.

Most of all, to my wife Erin, thank you for your unwavering support, love, patience, and fortitude of spirit through all of this.

---

# Preface

## **Overview**

If learning Maya seems like an overwhelming task, and you don't know where to begin, then this book is for you. You'll learn the fundamentals from the ground up, in an order that'll make sense. After reading *Understanding 3D Animation Using Maya*, you'll be able to create original 3D animation from scratch.

This is a self-training manual for beginning Maya users. It's both a general guide for understanding 3D computer graphics and a specific guide to learning Maya. By learning the fundamentals, you'll be able to tackle more complex tasks later on – your subsequent work in Maya will fit into a familiar framework.

With this book, you'll gain extensive knowledge in the key areas of 3D animation:

- Workspace – Getting the most out of the interface and workflow;
- Modeling – Creating objects and characters using real-world techniques;
- Animation – Adding motion to your objects and bringing your characters to life;
- Shading – Developing the look of your surfaces;
- Lighting – Illuminating your scenes with artfully placed lights;
- Rendering – Filming your scenes with virtual cameras.

## **Organization and Features**

Have you noticed the sheer number of menu items and buttons available in Maya? If so, you probably realize that this is one huge piece of software – Maya can do a wide range of things with a tremendous amount of depth. To make this power usable and to make Maya's features accessible to you, I cover all six fundamental topics in each chapter – this works better than isolating subjects such as modeling or rendering. Each chapter advances the concepts and skills a bit further than the last, while always emphasizing the big picture. From my experience teaching Maya, I've found it's beneficial to learn the basics and how they relate to each other, rather than to focus, at the outset, on a single topic in too much detail.

After reading about each topic, you'll be presented with one or more hands-on tutorials. By completing the tutorials, you'll get a good feel for a typical 3D animation workflow, and you'll be well on your way to creating your own 3D animation project.

Here's a breakdown of what you'll find in each chapter

## **Chapter 1 – Essential Skills**

You'll get your feet wet in this chapter: make your way around the interface; model some simple characters using primitive building blocks; animate a ball using keyframes; play around with the color and shininess of materials; place point lights with shadows; and render images of your finished scenes.

## **Chapter 2 – Refinements**

Take a closer look at important details such as project management and file organization, and explore the different geometry types (NURBS, polygons, and subdivision surfaces). Copy keyframes and place pivot points for control over your animation; learn how to choose different material types to create plastic, metal, and other shaders; place spotlights and manipulate their parameters; and add sophistication to your camera work.

## **Chapter 3 – Intermediate Skills**

Find out how to use the Hypergraph to explore Maya's node-based architecture; create models using curve-based techniques; fine-tune your animation with the Graph Editor; add detail to your materials with texture maps; develop sophisticated lighting with soft shadows and gobos; and render realistically using depth of field.

## **Chapter 4 – Adding Character**

Learn how marking menus, shelves, and other interface elements can be used to speed up your workflow; model your own characters using polygon and subdivision sculpting techniques; build complex animated assemblies with hierarchies and groups; add bump maps and transparency to your materials; increase the realism of your lighting with area lights and raytracing; and render effects such as glows and lens flares.

## Chapter 5 – Wiring Things Up

All elements in Maya can influence each other: use the Artisan painting interface for surface operations; connect multiple surfaces seamlessly with fillets; fly objects through your scenes with path animation; build skeletons to animate your characters; use displacement techniques to add surface details; apply custom UV mapping for accurate texture placement; illuminate your scenes with three-point lighting; and tell visual stories with camera cuts.

## Chapter 6 – Bringing It All Together

In the final chapter it's time to combine your scenes into the final animated short: make the interface work for you with layers, hiding and showing, and snapping; add deformation techniques to your modeling toolkit; rig props and characters using animation constraints, inverse kinematics, and skin weighting; paint textures directly onto your models in 3D; use mental ray's Final Gathering for realistic, photon-based lighting; add motion blur for an added touch of realism; and import multiple elements into your final scene.

## CD-ROM

The CD included with this book contains project directories with all the files you'll need to complete the tutorials and a finished version of each scene. In case you don't already own Maya, the CD has a full copy of the latest Maya Personal Learning Edition ready for you to install on Windows 2000, Windows XP, or Mac OSX. This free version of Maya enables you to start learning today.

## Audience

This book is for anyone who is new to 3D animation in general or migrating to Maya from another 3D software package. It's aimed at everyone interested in being creative in 3D, including designers, animators, game artists, sculptors, photographers, illustrators, Web designers, fine artists, engineers, and visual communications students. The possibilities with Maya are nearly limitless; it's used for a wide range of tasks:

- Feature film and video animation and visual effects;
- Game asset creation and level design;
- Interface design;
- Graphic design;

- Medical, scientific, and military visualization;
- Architectural presentation;
- Design visualization;
- Theatrical lighting and set design.

This book provides you with everything you'll need to learn the most essential Maya skills. By applying an artistic eye, a passion for problem solving, and a willingness to practice, you'll find that the concepts of 3D animation aren't very difficult. Once you've applied these concepts to your own work a few times, they'll be yours forever.

### ***Technical Notes***

The techniques presented in this book will work in nearly any version of Maya. For compatibility reasons, the scenes have been saved in the Maya 4.5 .ma format. There should be no problems opening the scenes with versions 4.5, 5.0, or 6.0 under Irix, Red Hat Linux, Macintosh OSX, or Windows 2000/XP. For online book updates, go to [www.understandingmaya.com](http://www.understandingmaya.com).

The Maya Image Gallery contains images from the book that can only be fully appreciated in full color, as well as select images from a few animated short films.

In order to get the most out of the tutorials, there are some Maya preferences that you should set before getting started. Once you have launched Maya, click on the main menu **Window > Settings/Preferences > Preferences...** In this window, click **Edit > Restore Default Settings**, then change the following settings:

<b>Interface</b> category:	<b>Open Attribute Editor = In Separate Window</b> <b>Open Tool Settings = In Separate Window</b>
<b>Kinematics</b> category:	<b>Joint Size = 0.5</b>
<b>Keys</b> category:	<b>Default In Tangent = Linear</b> <b>Default Out Tangent = Linear</b>
<b>Timeline</b> category:	<b>Playback Speed = Real-time [24 fps]</b>
<b>Undo</b> category:	<b>Queue Size = 100</b>

---

# Contents

Preface .....	ix
Maya Image Gallery .....	facing page 126
<b>1. Essential Skills.....</b>	<b>1</b>
1.1 Introduction to 3D Space – Cartesian Coordinate System .....	3
1.2 Workspace – Getting Around Maya.....	7
Input .....	9
Transformation Tools .....	10
Move Tool .....	11
Rotate Tool .....	11
Scale Tool .....	12
Show Manipulator Tool.....	13
Workspace Exercise – Transformation Tools .....	13
1.3 Modeling — Primitives, Wireframes, Surfaces, and Normals .....	17
Modeling Exercise – Chick .....	19
Modeling Exercise – Barn .....	26
1.4 Animation – Keyframe Basics .....	32
Animation Exercise – Ball Drop.....	33
1.5 Shading — Color, Specular, and Ambient .....	36
Shading Exercise – Shading Donuts .....	37
1.6 Lighting — Point Lights: Intensity, Color, and Shadows .....	40
Lighting Exercise – Lighting Donuts .....	41
1.7 Rendering — Outputting a Scene to Frames .....	45
Rendering Exercise – Rendering Donuts .....	46
1.8 Project 1 — Ball Rolling Down an Inclined Plane .....	51

<b>2. Refinements .....</b>	<b>53</b>
2.1 Workspace – Organization .....	54
The Outliner .....	55
Keyboard Shortcuts for Transformations .....	56
Project and File Management .....	57
2.2 Modeling – Geometry Types .....	59
NURBS .....	59
Polygons .....	59
Subdivision Surfaces .....	60
Component Mode .....	62
Modeling Exercise – NURBS Components .....	63
2.3 Animation – Copying Keyframes and Moving Pivots .....	68
Copying Keyframes .....	68
Animation Exercise – Copying Keyframes .....	68
Pivot Points .....	71
Animation Exercise – Pivot Points .....	73
2.4 Shading – Material Types .....	76
Shading Exercise – Material Types .....	77
2.5 Lighting – Spotlights .....	79
Lighting Exercise – Spotlight .....	80
2.6 Rendering – Camera Basics .....	83
Rendering Exercise – Two-node Camera .....	84
2.7 Project 2 – Rolling Egg .....	87
<b>3. Intermediate Skills .....</b>	<b>89</b>
3.1 Review – Attribute Editor .....	90
3.2 Workspace – Hypergraph .....	90
3.3 Modeling – Curves and Surfaces .....	92
Drawing Curves .....	92
Revolves .....	93
Construction History .....	94
Modeling Exercise – Revolves and Curve History .....	96



Modeling Exercise – Revolve and Extrude .....	97
Lofting .....	100
Modeling Exercise – Lofting Curves: Slide .....	101
Modeling Exercise – Lofting Curves: Fish .....	103
3.4 Animation – Fine Tuning Animation .....	106
Animation Exercise – Tangents .....	109
Animation Exercise – Tangent Plane .....	114
3.5 Shading – Texture Maps .....	120
Shading Exercise – Barn Textures .....	120
3.6 Lighting – Soft Shadows and Gobos .....	130
Soft Shadows .....	130
Lighting Exercise – Soft Shadow .....	130
Gobos .....	132
Lighting Exercise – Barn Gobo .....	133
3.7 Rendering – Depth of Field .....	136
Rendering Exercise – Depth of Field .....	136
3.8 Project 3 – Curved Half Pipe .....	139
<b>4. Adding Character .....</b>	<b>141</b>
4.1 Review – Graph Editor .....	142
4.2 Workspace – Efficient Workflow .....	142
Marking Menus .....	143
Hotbox .....	143
Interface Elements .....	143
Shelves .....	144
4.3 Modeling – Box Modeling with Polygons and Sub-d Surfaces .....	144
Modeling Exercise – Box Modeling Saddle .....	146
Modeling Exercise – Smooth Polygon Chicken .....	154
4.4 Animation – Hierarchies and Groups .....	162
Animation Exercise – Little Red Wagon .....	163
Transform Order .....	167
Animation Exercise – ‘Toon Tire .....	167

4.5 Shading – Bump Maps and Transparency .....	171
Bump Maps .....	171
Transparency .....	172
Shading Exercise – Butterfly .....	172
4.6 Lighting – Area Lights and Raytracing .....	176
Lighting Exercise – Cornell Box Area Lights .....	177
Raytracing .....	180
Lighting Exercise – Shiny Egg Raytrace .....	180
4.7 Rendering – Camera Effects .....	183
Light Glows and Lens Flares .....	183
Rendering Exercise – Optical FX Egg .....	184
4.8 Project 4 — Chicken Added to Scene .....	187
<b>5. Wiring Things Up .....</b>	<b>189</b>
5.1 Workspace – Artisan Tools .....	190
Workspace Exercise – Artisan .....	191
5.2 Modeling – Connecting Surfaces Using Fillets .....	193
Modeling Exercise – Submarine Fillets .....	194
5.3 Animation – Motion Paths, Driven Keys, Added Attributes, Skeletons .....	196
Motion Paths .....	196
Animation Exercise – Pursuit Plane Path .....	197
Driven Keys .....	202
Animation Exercise – Driven Keys .....	203
Added Attributes .....	205
Animation Exercise – Added Attributes .....	206
Skeletons .....	209
Animation Setup Exercise – Chicken Skeleton .....	211
Posing the Skeleton .....	220
Animation Exercise – Keyframing the Skeleton with MEL .....	221
5.4 Shading – Displacement and UV Texture Mapping .....	227
Displacement .....	227
Shading Exercise – Displacement Wing .....	227

UV Texture Mapping .....	232
Shading Exercise – UV Chicken .....	233
5.5 Lighting – Three-point Scheme .....	251
Lighting Exercise – Three-point Lighting Plane .....	252
5.6 Rendering – Camera Cuts .....	256
Rendering Exercise – Camera Cuts .....	256
5.7 Project 5 – Preparing for Animation .....	260
<b>6. Bringing It All Together .....</b>	<b>261</b>
6.1 Review – Editing Your Camera Cuts .....	262
6.2 Workspace – Layers, Show by Type, Hiding and Showing .....	262
Layers .....	262
Show by Type .....	262
Hiding and Showing .....	263
Snapping .....	263
6.3 Modeling – Cluster Deformers .....	264
Cluster Deformers .....	264
Modeling Exercise – Clusters .....	265
Modeling Exercise – Chicken Head Clusters .....	269
6.4 Animation – Constraints, Inverse Kinematics, and Skinning .....	271
Constraints .....	271
Animation Exercise – Point Constraint Egg .....	274
Animation Exercise – Multiple Constraint Pump .....	276
Inverse Kinematics .....	278
Animation Exercise – IK Rigging the Chicken .....	279
Character Skinning .....	283
Animation Exercise – Skin Binding .....	283
6.5 Shading – 3D Paint .....	290
Shading Exercise – 3D Painting .....	291
6.6 Lighting – Final Gathering in Mental Ray .....	294
Lighting Exercise – Final Gathering .....	294
6.7 Rendering – Motion Blur .....	299

## xviii Contents

---

Rendering Exercise – Motion Blurred Chicken .....	299
6.8 Layout – Importing Elements .....	301
Layout Exercise – Importing Elements .....	301
6.9 Project 6 – Bringing It All Together .....	305
<b>Index .....</b>	<b>307</b>

## Color Plate I

---

### Maya Image Gallery

The ChubbChubbs and Early Bloomer images in this gallery were created by talented artists at Sony Pictures Imageworks using Maya and other proprietary software. All other images were created by the author, except for On The Ball character modeling and animation and the chicken sketches, which were created by Philip Pignotti.



The ChubbChubbs (c) 2002 Sony Pictures Imageworks Inc. All rights reserved.

## Color Plate II

---



The ChubbChudds (c) 2002 Sony Pictures Imageworks Inc. All rights reserved.

### Color Plate III

---



The ChubbChubbs (c) 2002 Sony Pictures Imageworks Inc. All rights reserved.

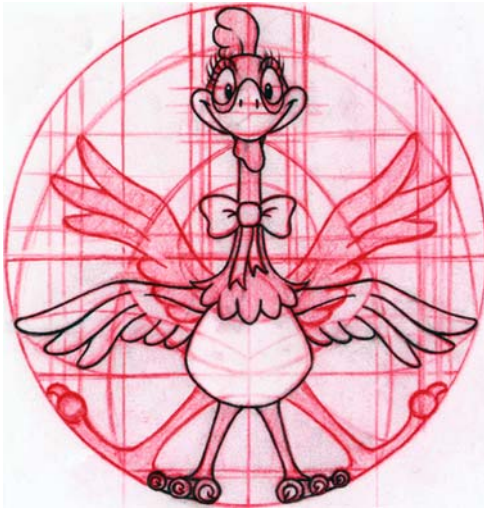


The ChubbChubbs (c) 2002 Sony Pictures Imageworks Inc. All rights reserved.

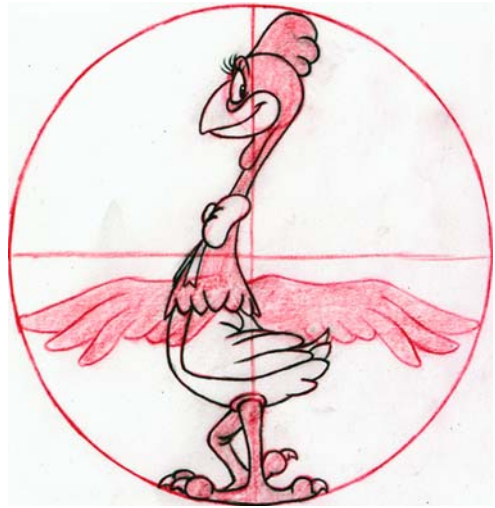


## Color Plate IV

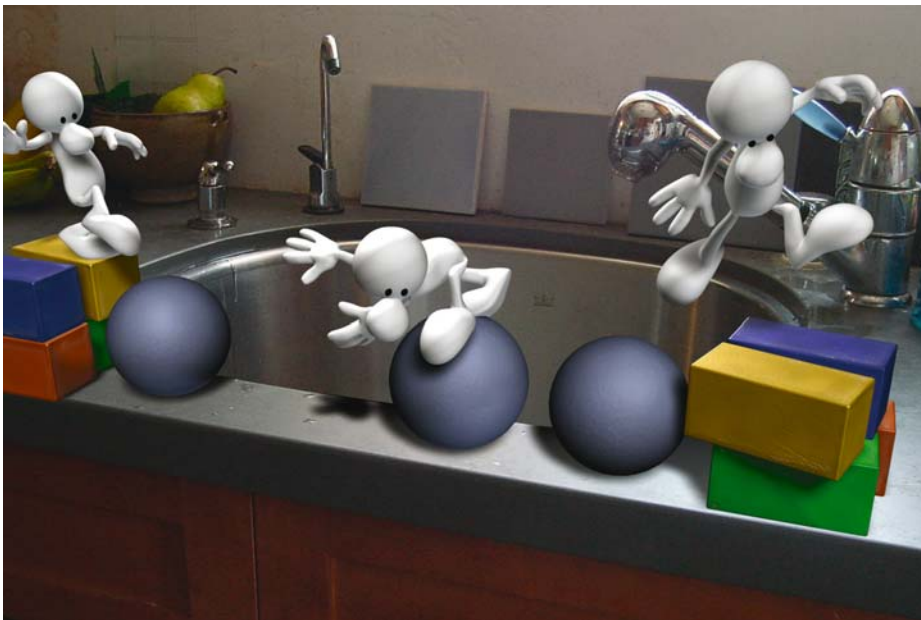
---



Front view of the chicken design



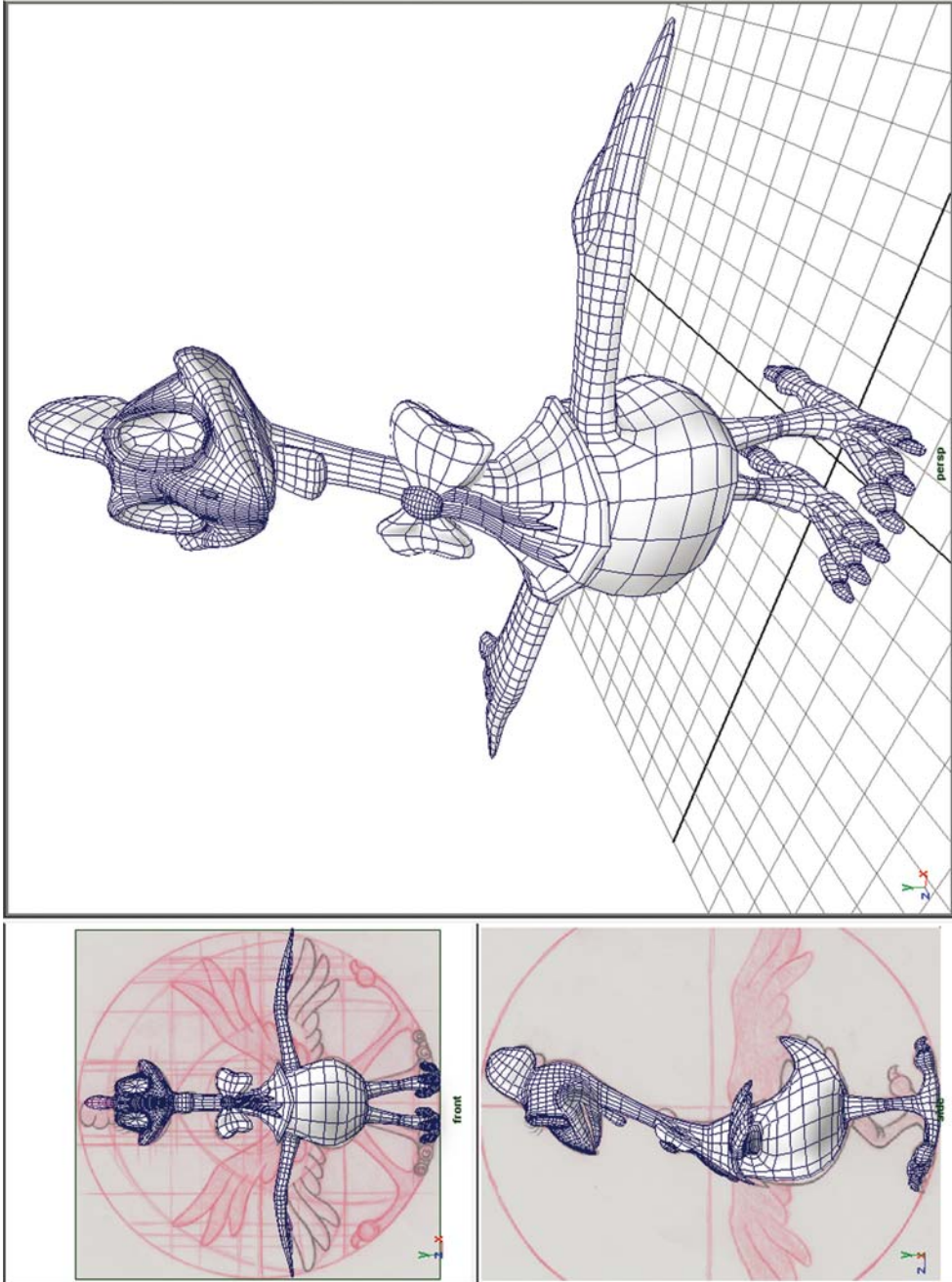
Side view of the chicken design



On the Ball by Philip Pignotti

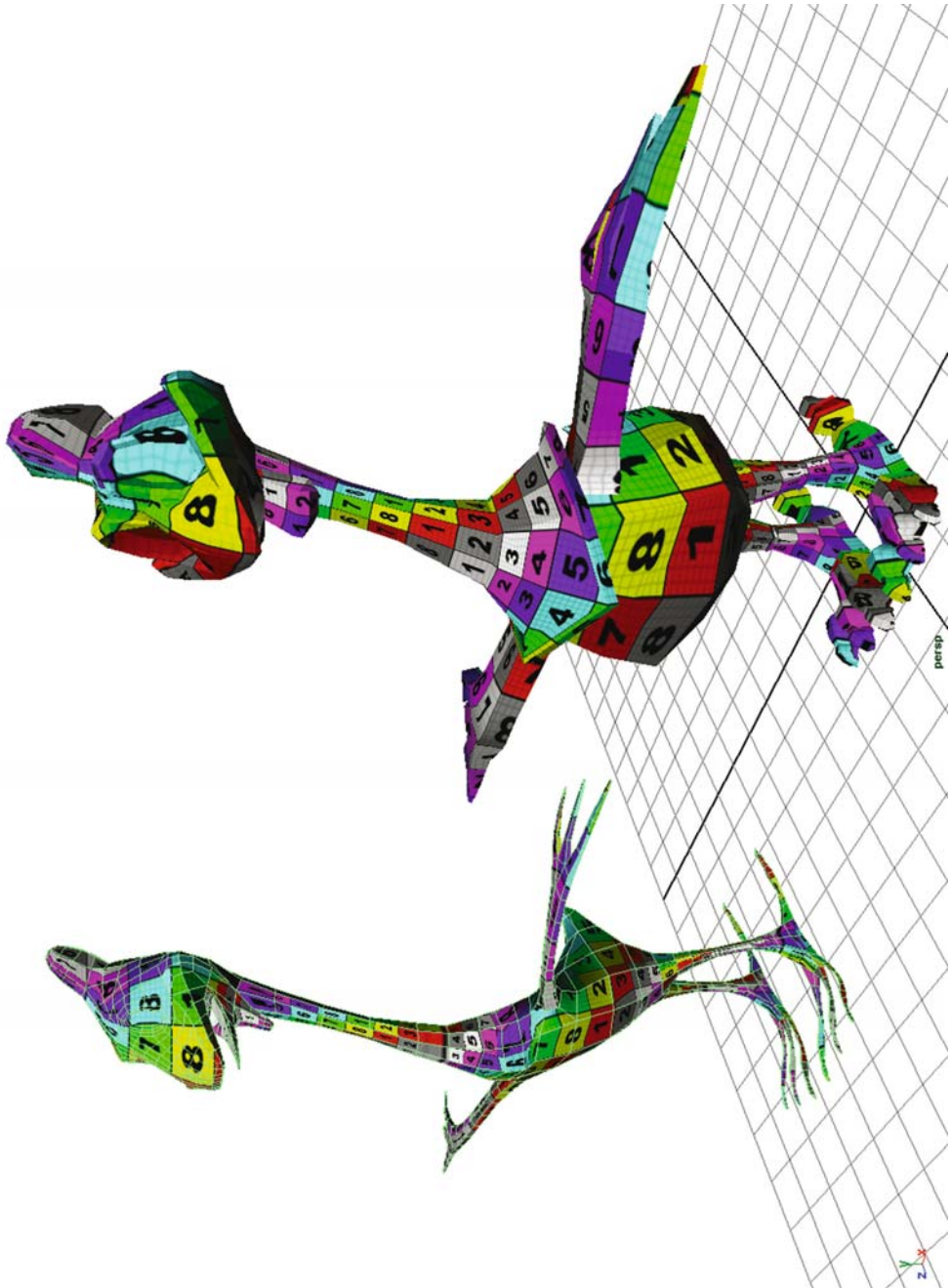


# Color Plate V



Polygonal model of the chicken

# Color Plate VI



UV texture map transfer on the chicken

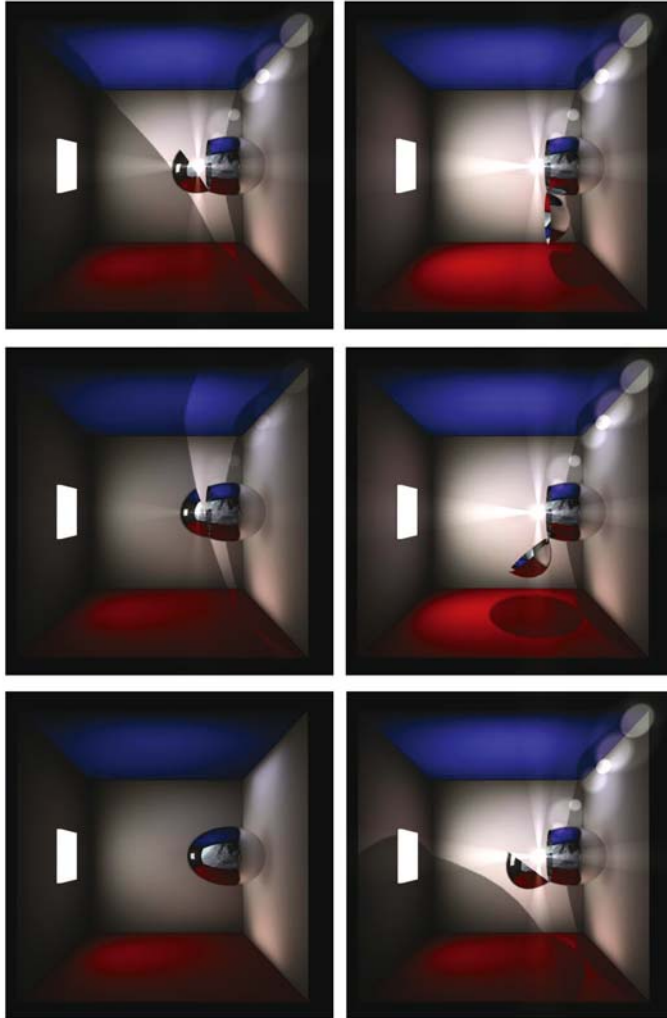
# Color Plate VII



Final merged scene elements

# Color Plate VIII

---



Optical effects egg from outer space

## Color Plate IX

---



Displaced wing rivets



Plane lit with Final Gather



Color Plate X

---



Depth of field inside the barn

## Color Plate XI

---



Three point lighting the plane



Early Bloomer (c) 2003 Sony Pictures Imageworks Inc. All rights reserved.

Color Plate XII

---



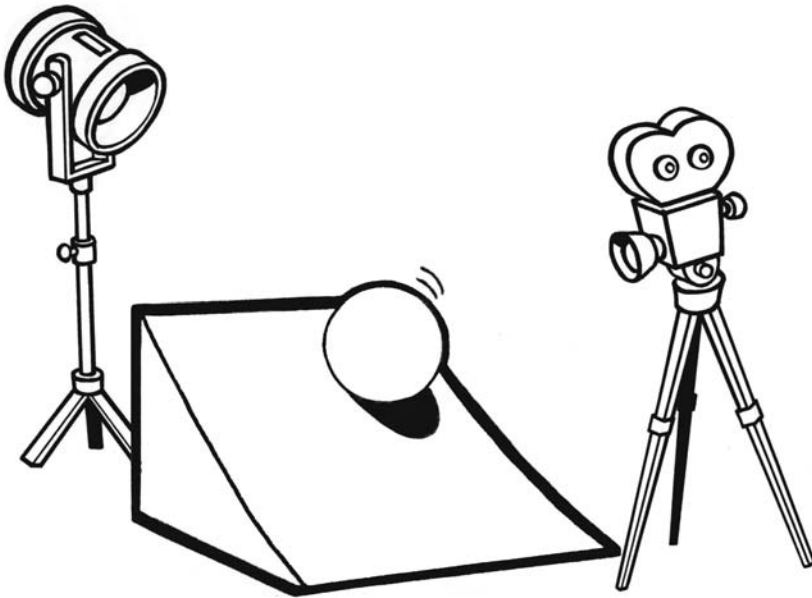
Early Bloomer (c) 2003 Sony Pictures Imageworks Inc. All rights reserved.



---

# Chapter 1

## Essential Skills



## 2 1. Essential Skills

---

3D animation in Maya is created by bringing together many parts to form a greater whole. Understanding how these parts relate to each other is one of the keys to learning Maya. Instead of completely focusing on one part at a time, we'll always keep the bigger picture in mind.

So what are these different parts? They are workspace, modeling, animation, shading, lighting, and rendering. This isn't an exhaustive list of Maya's capabilities—I've left out some things such as dynamics, particles, fur, and cloth. The particular areas we'll be focusing on, however, are the fundamentals.

Here is how the basic relationship between these elements works: using the workspace, models are created to represent the form of an object, these models are animated, the look of the model is determined through shading techniques such as materials and textures, lights are placed into the scene, and finally all of these elements are rendered through a camera, to produce the finished frames of your shot.

The skills you learn in these areas (denoted throughout the book by the icons in Fig. 1.1) will enable you to create a short animation of a ball rolling down a ramp. OK, it's probably not going to be the most exciting movie you'll ever create, but it's a good place to start.

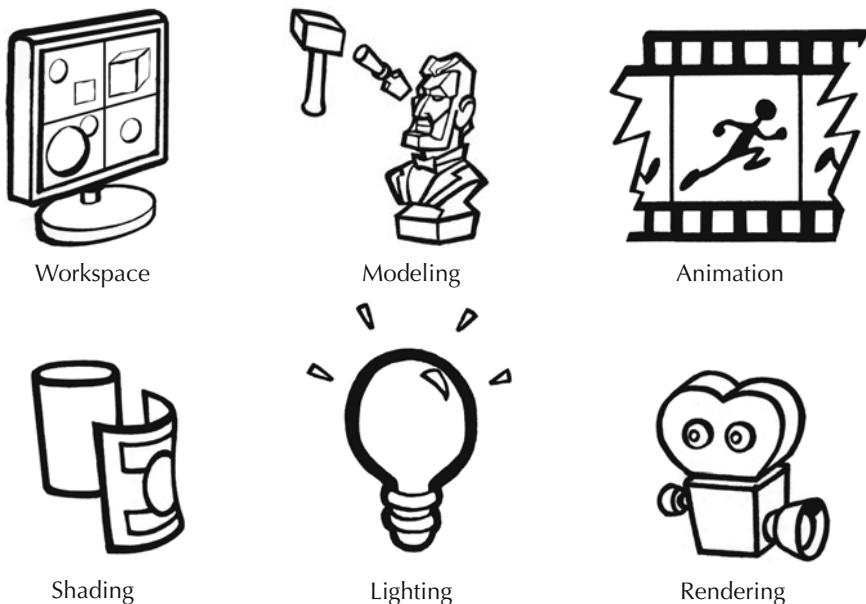


Figure 1.1: Fundamentals of 3D

## 1.1 Introduction to 3D Space – Cartesian Coordinate System

Two-dimensional space is represented with the X- and Y-axes. 3D animation adds depth, or the Z-axis. This is seen in Figures 1.2 and 1.3 of the plane in perspective.



Figure 1.2: 2D space

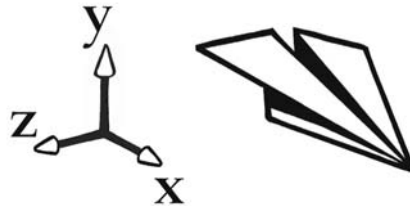


Figure 1.3: 3D space

This three-axis reference system is known as the Cartesian coordinate system. It is used to define many properties of a 3D object, including its position, rotation, and scale. For example, an object's position can be represented as 12, 0, 10 – meaning that it is 12 units to the right of center on the X-axis, 0 units from center along the Y-axis, and 10 units along the Z-axis.

Look at the small view axis locator that appears in the lower left corner of any view panel. It indicates the direction of the X-, Y-, and Z-axes in the world coordinate system. Notice the directions the axes point, as well as their colors. Each axis points in its positive direction; so up = positive Y, down = negative Y, right = positive X, left = negative X, out = positive Z, and in = negative Z. The X-, Y-, and Z-axes are represented by red, green, and blue, respectively. In fact,

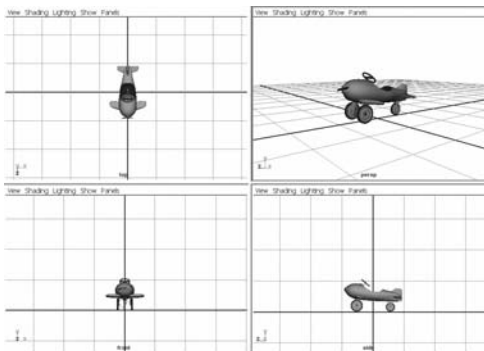


Figure 1.4: Translation: 0,0,0 Rotation: 0,0,0 Scale: 1,1,1

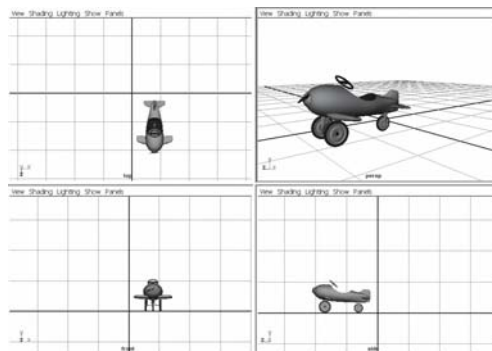


Figure 1.5: Translation: 2,0,3 Rotation: 0,0,0 Scale: 1,1,1

## 4 1. Essential Skills

---

throughout Maya there is a consistent link between XYZ and RGB.

In the examples in Figures 1.4 - 1.10, note the position, rotation, and scale of the plane. (Position is in units; rotation is in degrees; scale is in multipliers.)

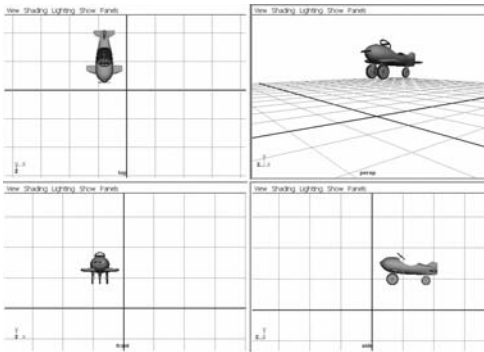


Figure 1.6: Translation: -2,2,-3; Rotation: 0,0,0; Scale: 1,1,1

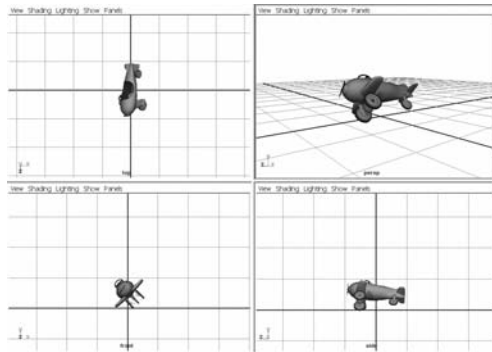


Figure 1.7: Translation: 0,0,0; Rotation: 0,0,45; Scale: 1,1,1

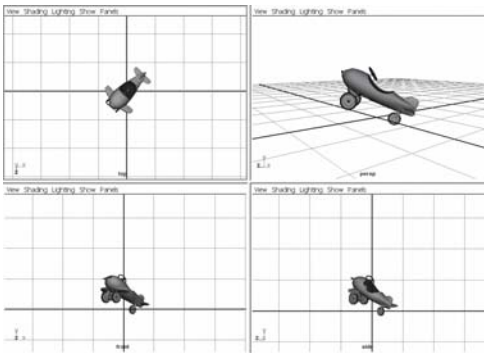


Figure 1.8: Translation: 0,0,0; Rotation: -20,-45,0; Scale: 1,1,1

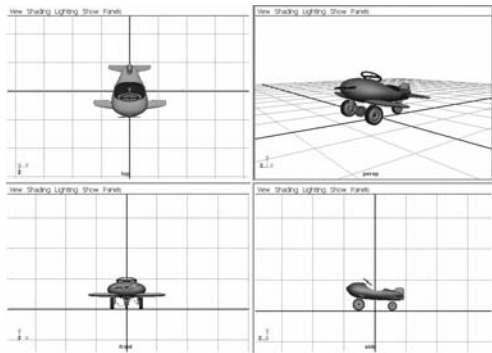


Figure 1.9: Translation: 0,0,0; Rotation: 0,0,0; Scale: 2,1,1

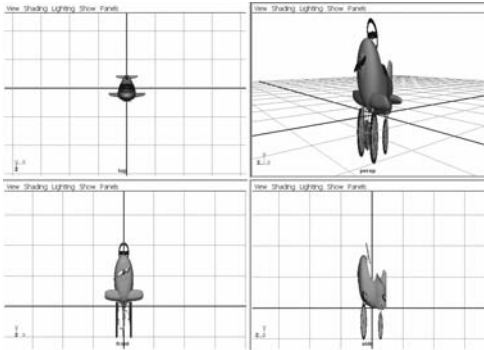


Figure 1.10: Translation: 0,0,0; Rotation: 0,0,0; Scale: 1,3,0.5

Let's go ahead and open up one of the scenes provided on the **Understanding Maya** CD-ROM, just to get a feel for 3D space. Click on **File > Open...** (see Figure 1.11). Browse to the Chapter01/scenes folder and open PursuitPlane01\_begin.ma (see Figure 1.12).

Click the play button to see the airplane move, rotate, and scale on the three axes. (see Figures 1.13 - 1.15.) When you're done watching, you can press stop.

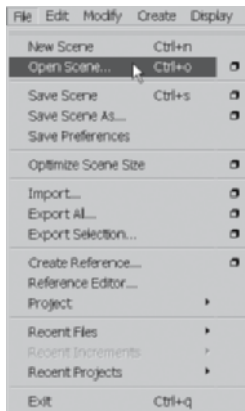


Figure 1.11: Open scene

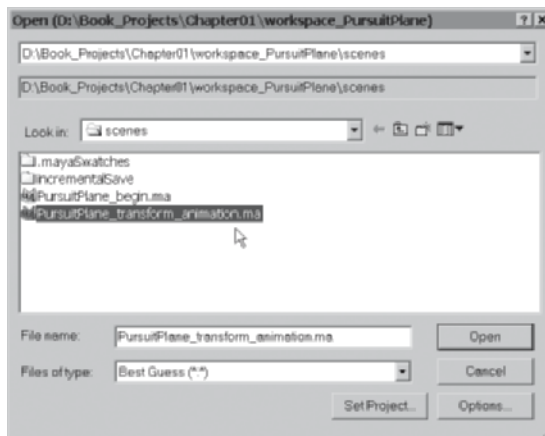


Figure 1.12: File browser

## 6 1. Essential Skills

---

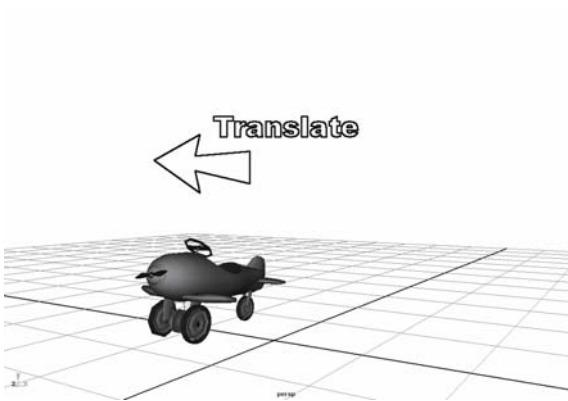


Figure 1.13: Translate

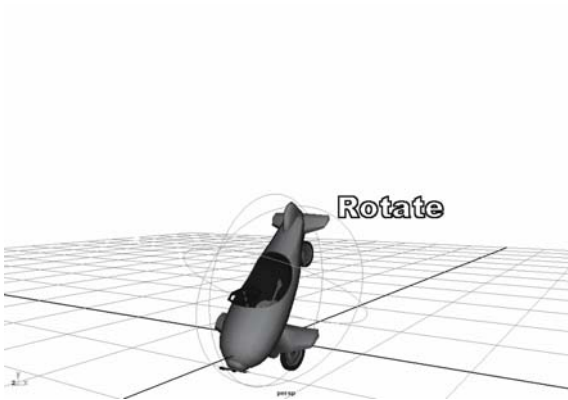


Figure 1.14: Rotate

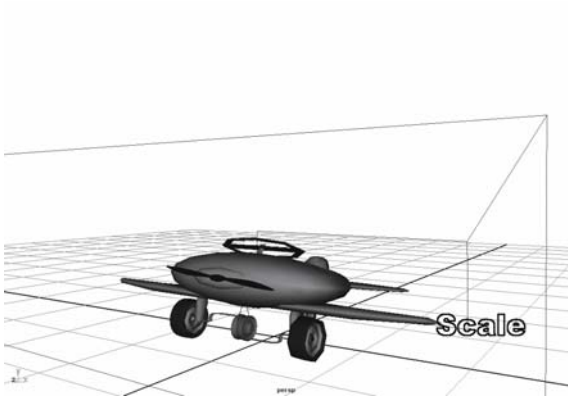
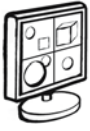


Figure 1.15: Scale

## 1.2 Workspace – Getting Around Maya



The Maya workspace presents the tools and user interface (UI) elements necessary to create all aspects of your animation (see Figure 1.16). In order to avoid being overwhelmed, some task-specific elements are only there when you need them, and others are ever-present. Let's look at some of the most commonly used elements.

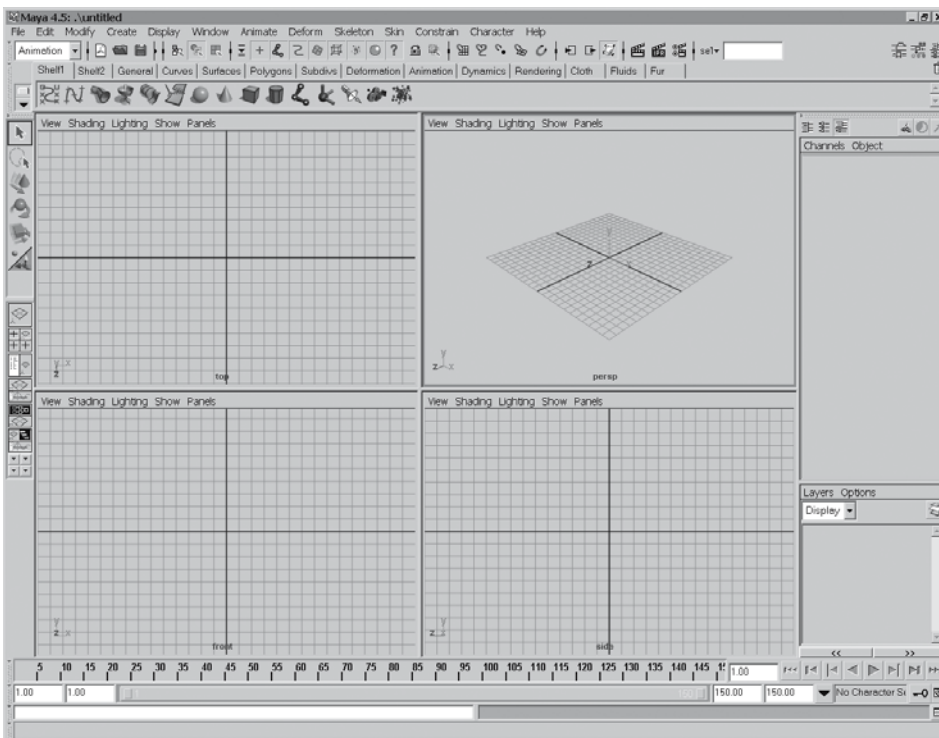


Figure 1.16: Maya's interface

Menu bar: This is the familiar menu bar of almost any application. The first six menus remain the same no matter with which subset of the Maya interface you are working. The remaining items will vary based on context (see Figure 1.17).



Figure 1.17: Menu bar

## 8 1. Essential Skills

---

Menu Sets: These allow you to select the specific contextual menu items for the mode of work you are in, i.e., Animation, Modeling, Rendering, and Dynamics (see Figure 1.18).



Figure 1.18: Menu sets

Panels: These are the perspective or orthographic viewports in which you do your work. These may be split into multiple views and can display editing windows and 3D scene data (see Figure 1.19).

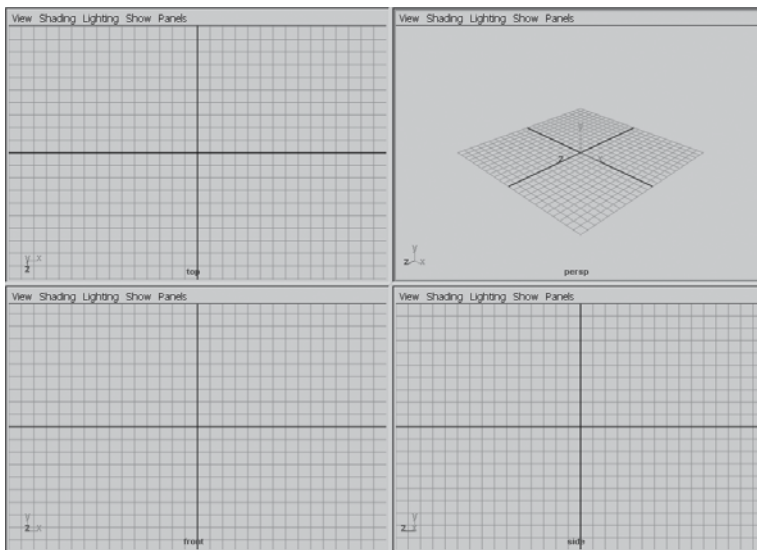


Figure 1.19: Panels



Figure 1.20: Help line

Help Line: Look to this area for instructions and feedback on using the various tools in Maya (see Figure 1.20).



Figure 1.21: Time slider



Time Slider: This shows you the time, in frames, of your animation. Keyframes will appear here (see Figure 1.21).



Figure 1.22: Range slider

Range Slider: This is where you can specify the length of your animation or the subsection you are currently working on (see Figure 1.22).



Channel Box: This is an area that displays an object's keyable values such as translation, rotation, scale, and visibility (see Figure 1.23).

## Input

You will generally use the mouse and keyboard as your primary input devices for Maya. In this book, we'll cover a number of different combinations, shortcuts, and interface elements that allow easy access to Maya's tools and functions.

The three-button mouse is the preferred weapon of choice for Maya. You select menus and buttons exactly the same way you normally do – move the cursor to the **File** menu, for example, and then click the left mouse button to expand the dropdown menu. Where things are a bit different in Maya is inside the 3D panels, time slider, and data fields. The left mouse button is used to select and alter things, the middle mouse button is used to change things on multiple axes, and the right mouse button is used to access contextual menus.

Figure 1.23: Channel box  
mouse button.

The standard abbreviations for the three buttons are: **LMB** = left mouse button, **MMB** = middle mouse button, and **RMB** = right

View navigation is accomplished with keyboard and mouse button combinations. In order to look at your models from all sides, pan around the screen, and get a closer look at an area you will hold down the ALT key on the keyboard while pressing a mouse button and moving the mouse. These are called Tumble, Track and Dolly moves, just as in traditional camera work. A visual representation can be seen in Figure 1.24.

**Maya Camera Navigation**

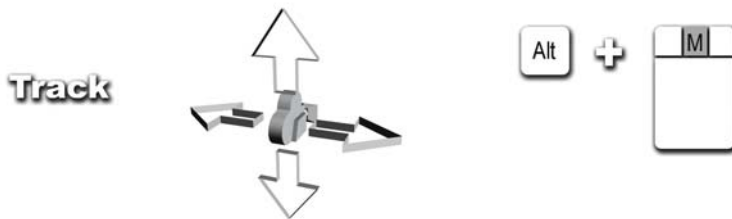


Figure 1.24: Camera tools

### Transformation Tools

Transformation in Maya refers to the translation, rotation, and scale of an object in 3D space. (Translation is another word for movement or position.) Perhaps the three most frequently used tools are the **Move**, **Rotate**, and **Scale Tools**. For this reason, they have a prominent place in the user interface, and their hotkeys are in a home position on the keyboard (at least for right-handed mousers).

Additionally, some objects have special functions that can be accessed through the manipulator handle. These can control things such as the aim of a spotlight or the construction history of a surface.

## Move Tool

In many cases, you'll want to position an object in space. The tool that is used for such a translation is the Move Tool (see Figures 1.25 and 1.26). The Move Tool looks similar to the view axis, but with arrows at the end of each axis. Notice that the colors of the Move Tool handles correspond to the three axes. By **LMB** clicking and dragging along a handle, the object moves along that one axis.



Figure 1.25: Move tool icon

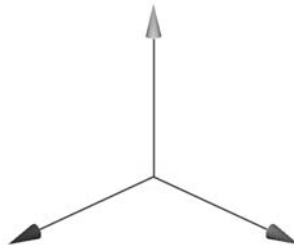


Figure 1.26: Move tool handle

## Rotate Tool

Another common manipulation is rotation, which is accomplished with the Rotate Tool (see Figures 1.27 and 1.28). To use the rotate tool, select and **LMB**-drag to increase or decrease the rotational angle. The manipulator handles allow you to rotate on any of the individual axes by using the inner circles, or around the screen axis by using the outer circle. The screen axis is an axis created by drawing a vector from the center of your view camera through the center of the object.

## 12 1. Essential Skills

---



Figure 1.27: Rotate tool

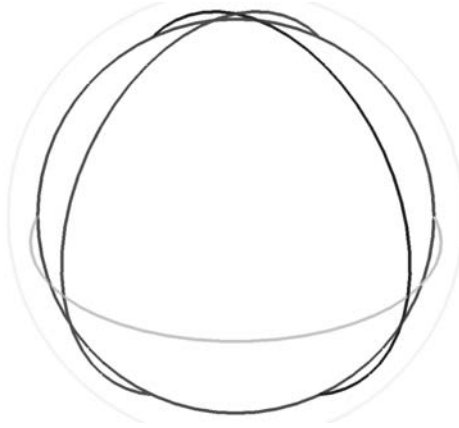


Figure 1.28: Rotate tool

### Scale Tool

With the Scale Tool you can change the scale of an object along the individual axes (see Figures 1.29 and 1.30). Again, select one of the axis handles and **LMB**-drag to increase or decrease the scale of an object. Or, drag from the tool's center to scale on all three axes simultaneously.



Figure 1.29: Scale tool

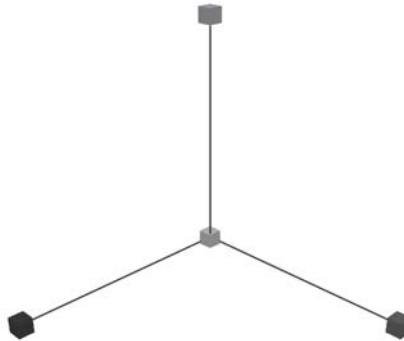


Figure 1.30: Scale tool

## Show Manipulator Tool

The Show Manipulator Tool's appearance and function varies depending upon the object on which you use it (see Figures 1.31 and 1.32). As we'll see in the next exercise, an object can also be customized to automatically display a specific manipulator handle (such as one of the other transformation tools) when it is selected.

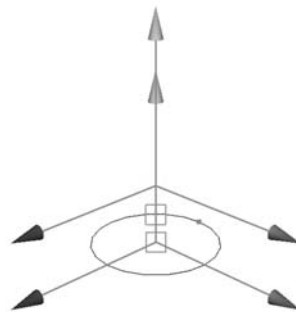


Figure 1.31: Show manipulator tool

Figure 1.32: Show manipulator tool

## Workspace Exercise – Transformation Tools

To try out some of these transformation tools we'll do an exercise. There are many of these hands-on exercises throughout the book, and all of them can be completed using nearly any version of Maya or Maya Personal Learning Edition. (An installation for the latest version of Maya PLE is included on the CD-ROM that comes with this book.) It is assumed that you are running Maya with the default factory settings and all tools and commands have been reset.

Also, I'd like to point out that the online help documents that come with Maya are an excellent source of additional information. Should you ever want to explore a tool, command, or technique in greater depth, press **F1** on your keyboard and search the manual!

In this exercise, you will translate, rotate, and scale an airplane model (see Figures 1.33).

1. Launch Maya.
2. Click **File > Project > Set...**

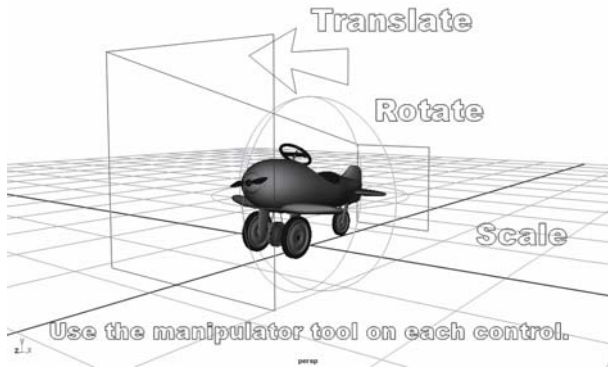


Figure 1.33: The pursuit plane transformation exercise

3. The **Browse for Folder** window appears. Browse to your CD-ROM and select `understanding_maya\Chapter01\workspace_PursuitPlane`. Click **OK**.
4. Click **File > Open Scene...**
5. The **Open** window appears and is already set to the `understanding_maya/Chapter01/PursuitPlane/scenes` directory. Select the file `PursuitPlane_begin.ma` and click **Open**.
6. When the scene has opened you will see an airplane model and three labeled control objects. These control objects have been created to familiarize you with the transformation tool functions.
7. Choose the **Move Tool** from the **Tool Box**.
8. Click on the *TranslateControl* arrow to select it. Notice the **Move Tool** handle. It represents X, Y, and Z translation with red, green, and blue arrows. These arrows are aligned with the View Axis in the lower-left corner of the **Perspective** panel.
9. Now, **LMB** click-drag the red translateX handle left and right. The airplane translates on the X-axis. Also, notice how the translateX handle has turned yellow. This indicates that it's the currently selected handle.
10. As you continue to **LMB** click-drag the translateX handle left and right, take a look at the **Channel Box** on the right side of the screen. The Translate X channel's value is increasing and decreasing as you move the airplane. Notice how dragging the handle in the

same direction its arrow points results in an increase in value. Dragging in the opposite direction that the arrow points results in a decrease in value.

11. In order to reset the airplane's position (see Figure 1.34), go to the **Channel Box** and **LMB**-click the number in the box to the right of Translate X. This highlights the entire value. Type 0 and then press the enter key. The airplane has now returned to the center of the scene.

12. Repeat steps 9, 10, and 11 on the green translateY handle and the blue translateZ handle, zeroing each channel after you're done. ("Zeroing" means resetting a channel to its de-

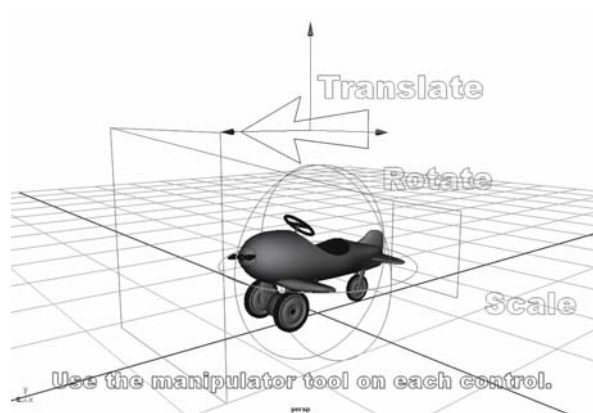
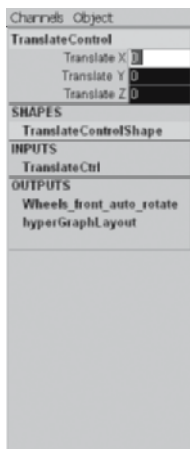


Figure 1.34: Resetting the Channel Box    Figure 1.35: Using the Translate Tool

fault value. In the case of Translation and Rotation this number is zero, whereas for Scale this number is one.)

13. If you look at the center of the **Move Tool** handle you will notice a small box. **LMB** click-drag this box to change the translation values on multiple axes simultaneously (see Figure 1.35). Once you're done, zero out the Translate X, Translate Y, and Translate Z channels.

14. Next, we'll look at rotation. Switch to the **Rotate Tool** (see Figures 1.36) and click on the *RotateControl* sphere to select it. You will now see the Rotate Tool handle, which has rotateX, rotateY, and rotateZ handles represented by red, green, and blue rings. The outer ring (which is highlighted yellow when you first switch to the Rotate Tool) allows you to rotate an object around the view axis.

## 16 1. Essential Skills

---

15. Now, **LMB** click-drag the red rotateX handle left and right. The airplane rotates on the X-axis, just like the pitch control on a real plane. To understand the effect of the different axes of rotation, think of this: If you were to skewer the model with a stick that ran along the X-axis, spinning the model around that stick would result in X-axis rotation.

16. As in the previous example of the **Move Tool**, when you LMB click-drag the rotateX handle, the **Channel Box** indicates an increase and decrease in value for the Rotate X channel. When you pitch the airplane's nose down, Rotate X increases. When you pitch the airplane's nose up, Rotate X decreases.

17. Reset the airplane's Rotate X value by zeroing it in the **Channel Box**.

18. Repeat steps 15, 16, and 17 on the green rotateY handle and the blue rotateZ handle. Be sure to zero out the rotation values when you're done.

19. Next, we'll try out the **Scale Tool** (see Figure 1.37). In the **Tool Box**, switch to the **Scale Tool** icon and then click on the *ScaleControl* box to select it. The Scale Tool handle looks very similar to the **Translate Tool**, but instead of arrows, there are boxes at the ends of the individual axes. Once again, scaleX, scaleY, and scaleZ are color-coded red, green, and blue. The initially highlighted center handle allows for proportional scaling in all three dimensions.

20. To widen the airplane, **LMB** click-drag the red scaleX handle to the right. Dragging to the left will narrow the airplane. Notice this time that as the Scale X value approaches and then surpasses zero, the airplane seems to pass through itself and emerge as a mirror image of the original model. (This behavior will be explored further when we look at object duplication.)

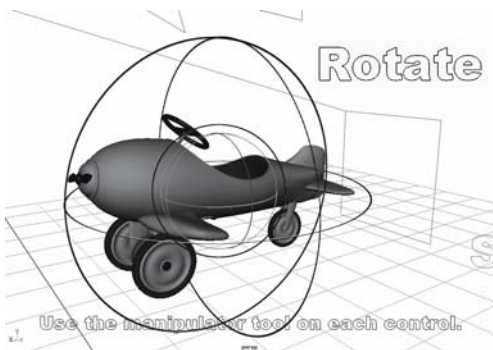


Figure 1.36: Using the Rotate Tool

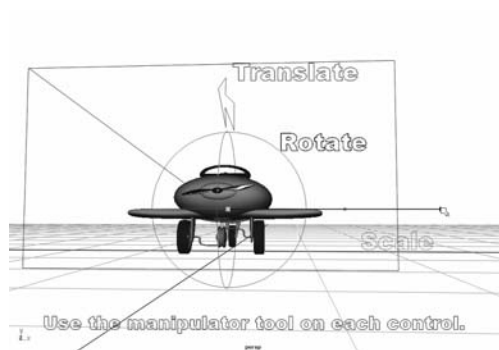


Figure 1.37: Using the Scale Tool



21. Reset the airplane's Scale X value to 1 in the **Channel Box**. Notice that resetting, or "zeroing," a scale to its default value means setting it to one, not zero, as is the case with translation and rotation.

22. Repeat steps 20 and 21 on the green scaleY handle and the blue scaleZ handle. Reset the values to 1 when you are finished.

23. You won't normally use the **Show Manipulator Tool** on an object like this airplane. However, the three transformation controls in this scene (the arrow, sphere, and box) have been specially set up for use with the **Show Manipulator Tool**. Let's try this now.

24. In the **Tool Box**, switch to the **Show Manipulator Tool**. Click the *TranslateControl* arrow. You will see the Translate handle appear.

25. Without switching tools (the **Show Manipulator Tool** is still active) click on the *RotateControl* sphere. Notice how it automatically displays the Rotate handle.

26. Lastly, click on the *ScaleControl* box. Again, the **Show Manipulator Tool** updates to display the Scale handle.

The key things you've learned in this exercise are how the transformation tools work, the relationship between an object's values and the **Channel Box**, and that there are often multiple ways to do something inside the Maya interface.

## 1.3 Modeling — Primitives, Wireframes, Surfaces, and Normals



3D models are mathematical descriptions of an object's geometry. These descriptions include points, lines, curves, and surfaces and are often represented onscreen by a wireframe model. Models can range from simple primitives to very complex meshes. A geometric primitive is a basic building block of 3D geometry, such as a sphere, cube, cylinder, cone, or torus. In the following examples we'll look at a primitive sphere.

In Figure 1.38 you see the wireframe representation of a sphere. The same sphere can be represented in a faceted shaded mode, as in Figure 1.39. And a smooth shaded mode is shown in Figure 1.40.

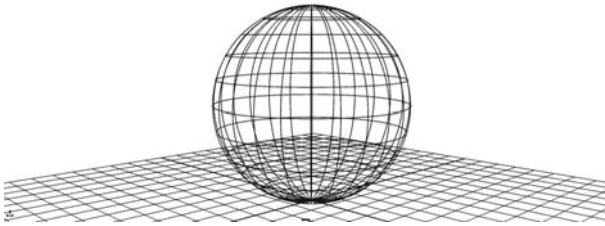


Figure 1.38: Wireframe sphere

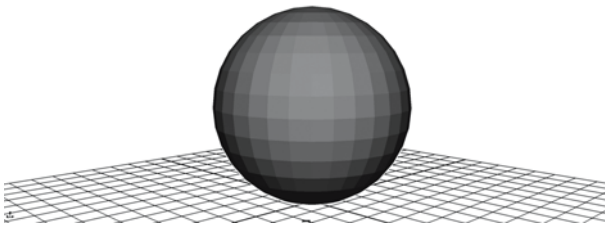


Figure 1.39: Faceted shaded sphere

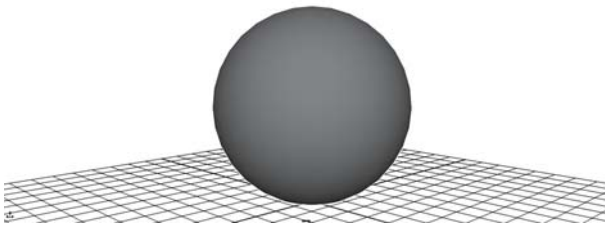


Figure 1.40: Smooth shaded sphere

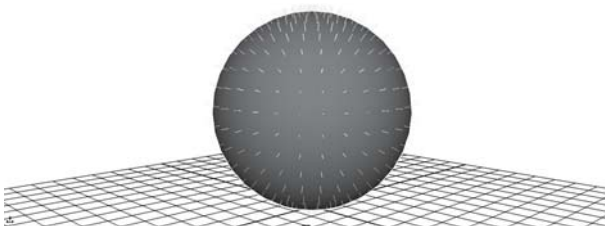


Figure 1.41: Smooth shaded sphere with visible normals

One of the sphere's surface properties represented in Figure 1.41, is the direction of its surface normals. A normal is a short line that grows perpendicular to the surface, much like a hair from your head. It describes the direction that the surface patch or face is pointing. In essence, every surface has a "front" and a "back." Normals point out of the front of a face.

### Modeling Exercise – Chick

To become familiar with some of the basic geometric primitives, you'll build a simple stylized character. A model of a baby chick will be formed by creating primitive objects and then moving, rotating, and scaling them into place. While it is possible to model characters using much more complex techniques, this building block approach can be very effective.

In Maya, newly created objects sit right in the middle of the world, a place known as the origin. After creating a primitive you will then translate, rotate, and scale it into position. Remember that the **Channel Box** gives you precise numerical feedback on your transformations, so watch the numbers as you position, rotate, and scale the various pieces of your model.

One thing to keep in mind while working is that the numbers in the **Channel Box** are less important than the look of your model onscreen. In some tutorials, replicating the exact numbers will be very important. This is not one of those tutorials. (I've listed the numeric values I used so you can match the model exactly if you like, but feel free to build a different bird.) So, relax, use your artistic judgment, and most important, have fun modeling.

1. Create a new project directory by clicking **File > Project > New...**
2. Enter the name *Modeling\_Chick* in the **Name** field.
3. Click the **Use Defaults** button, followed by the **Accept** button. This instructs Maya to generate a directory structure on your hard drive that will act as the home to the various related parts of your projects. Scenes, sounds, rendered frames, texture files, and many other types of data will be properly stored and easily accessible when you use the project.
4. Let's start out by creating a body for the chick. Click **Create > NURBS Primitives > Sphere**. This places an object named *nurbsSphere1* into the scene, with its center located at the world origin, or 0, 0, 0 (see Figure 1.42).

In Maya, the grid acts as a reference point for objects, sort of like a virtual floor. This is an arbitrary floor, however. You are welcome to place your models anywhere in the scene, even below the grid. Since the grid does not appear when you render the final frames of animation, your scene's relative position to the grid will not matter. That said, it is convenient in many scenes to treat the grid like a floor, which is what we'll do throughout this book.

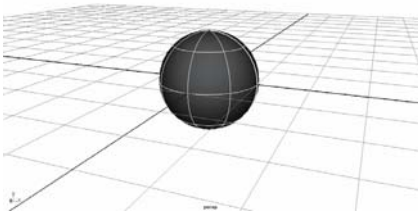


Figure 1.42: Initial body sphere

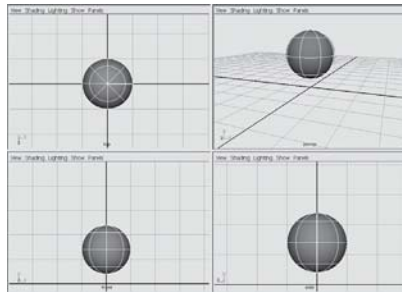


Figure 1.43: Sphere positioned above grid

5. Next, we'll move the sphere above the grid. With the *nurbsSphere1* object still selected, choose the **Move Tool** from the **Tool Box**. Click-drag the green Y-axis handle upwards, until the *nurbsSphere1* is 1.4 units above the grid (see Figure 1.43). Remember to check the **Channel Box** to verify, or edit, the value you have changed.

This is a good time to give the *nurbsSphere1* object a more descriptive name. It requires a good bit of discipline to name your objects as you create them, but I promise that this effort will pay off. Typically, your scenes will become filled with tens or even hundreds of objects; your ability to select and manipulate those objects is greatly improved by descriptive naming.

6. One place to rename an object is in the **Channel Box** as in Figure 1.44. Highlight the name *nurbsSphere1* by LMB-clicking it. Next, type the name *body* and press **Enter** on the keyboard.

7. Save the scene by clicking **File > Save Scene**. In the **Save Window** type `modeling_chick` in the **File Name:** field. Click the **Save** button.

8. We'll create the chick's head in much the same way. Click **Create > NURBS Primitives > Sphere**.

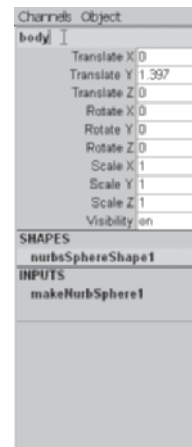


Figure 1.44: Renaming in the Channel Box

9. Move this new sphere up in the Y-axis and out in the Z-axis to place it above the body. The values in the **Channel Box** should read Translate X = 0, Translate Y = 2.8, and Translate Z = 0.33 (see Figure 1.45).

10. Rename this sphere *head*.

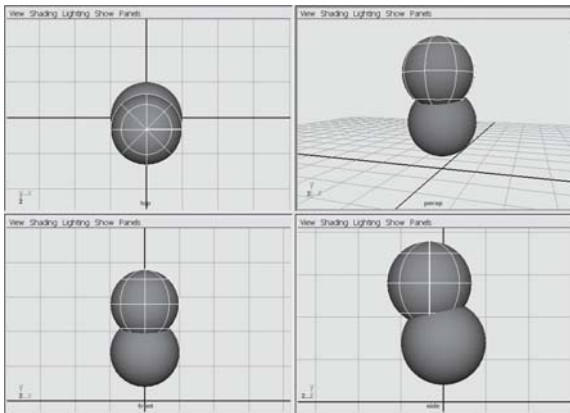


Figure 1.45: Repositioned head

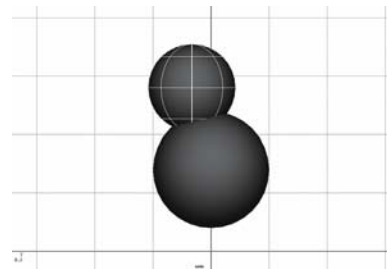


Figure 1.46: Rescaled head

11. The chick's *head* is too large, so we'll scale it down a bit (see Figure 1.46). Pick the **Scale Tool** from the **Tool Box**. By default, when you switch to one of the transformation tools, it can be manipulated on all three axes simultaneously. Notice the yellow box in the **Scale Tool's** center – click on this yellow box, then drag to the left to scale the *head* down in size. Stop when you reach 0.75 on all three axes.

12. Next, let's give this bird a beak. Click **Create > NURBS Primitives > Cone** (see Figure 1.47). Rotate the cone 100 degrees on the X-axis. This orients it properly. Translate it up and out on the Y- and Z-axes to a position of 0, 2.6, 0.8. (In describing an object's translation, the three coordinates are always listed in the order X,Y,Z. The same is true of rotation and scale.) Finally, scale the cone down in size to 0.42, 0.42, 0.42.

13. Be sure to rename the cone *beak*.

14. OK, the chick is coming together nicely now – time to give him some wings. For this, we'll use a couple of spheres. By scaling them non-proportionally we can get an appropriate flattened shape. Click **Create > NURBS Primitives > Sphere**. Rename the sphere *right\_wing*.

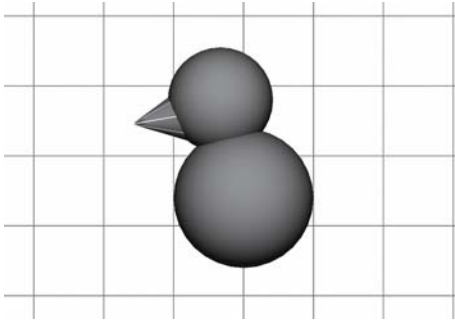


Figure 1.47: Beak cone

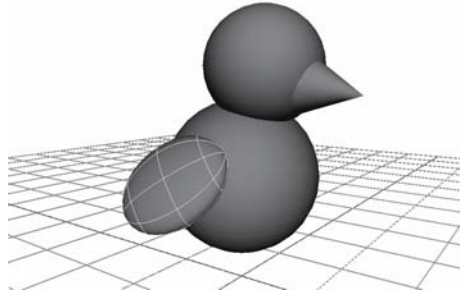


Figure 1.48: Right wing

15. Translate the *right\_wing* to -1.1, 1.45, -0.2. This places it off to the right side of the chick's body.

16. Using the **Scale Tool**, or through direct input in the **Channel Box**, scale the *right\_wing* to 0.2, 0.68, 0.5.

17. Last, rotate the *right\_wing* to 48, 8, -28. The wing is now in its proper place (see Figure 1.48).

18. For the other wing, create another NURBS sphere, name it *left\_wing*, and then repeat the above steps. The only differences will be in the translation and rotation: translation = 1.1, 1.45, -0.2 and rotation = 48, -8, 28.

19. Save your scene. Since it has already been named, you can simply click the floppy disk icon on the **Status Line**, or press **<ctrl> s** on your keyboard (see Figure 1.49).

20. For the chick's legs, some simple cylinders will do the trick. Click on **Create > NURBS Primitives > Cylinder**. Rename the cylinder *right\_leg*. Move the *right\_leg* into position: translation = -0.35, 0.45, 0.

21. Scale the leg down to size; Scale = 0.07, 0.35, 0.07.

22. Build the *left\_leg* the same way, but be sure to translate to the other side: translation = 0.35, 0.45, 0 (see Figure 1.50).

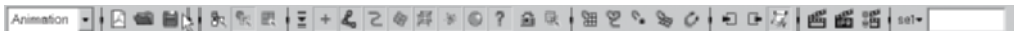


Figure 1.49: The save disc icon on the Status Line

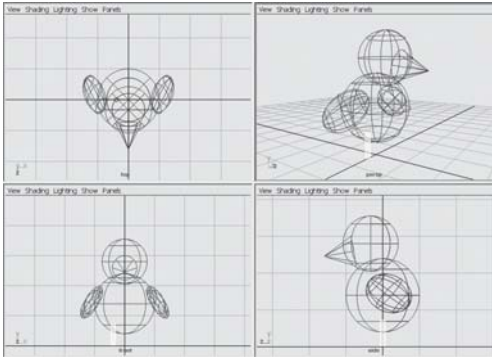


Figure 1.50: Leg placement

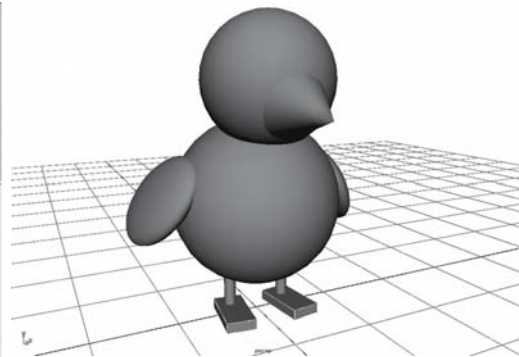


Figure 1.51: Foot position

23. Save your scene.

24. We'll use a couple of squashed cubes for the feet (see Figure 1.51). Click **Create > Polygon Primitives > Cube**. Rename the cube *right\_foot*. Scale the cube down to more foot-like proportions: Scale = 0.35, 0.12, 0.7.

25. Next, rotate the *right\_foot* object slightly on the Y-axis;  $-8^\circ$  will do. This tilts it out a bit.

26. Last, move the *right\_foot* into position under the *right\_leg* object. Translation = -0.37, 0.08, 0.18.

27. Repeat these steps for the *left\_foot*, making sure to rotate the other direction,  $8^\circ$ . Translation will be 0.37, 0.08, 0.18.

28. The tail and crest will be built in a similar fashion to the wings. First, you'll create a sphere, then you'll non-uniformly scale it, rotate it, and position it into place. Click **Create > NURBS Primitives > Sphere**. Name this sphere *tail*.

29. Move the *tail* to the back of the body. Translation = 0, 1.25, -0.75.

30. Scale the *tail* into an appropriate shape. Scale = 0.2, 0.45, 0.7.

31. Finally, rotate the *tail* to a  $35^\circ$  angle on the X-axis in order to tilt it up (see Figure 1.52).

32. For the crest (see Figure 1.53), click **Create > NURBS Primitives > Sphere**. Name this sphere *crest*.

33. Move the *crest* to the top of the head. Translation = 0, 3.5, 0.2.

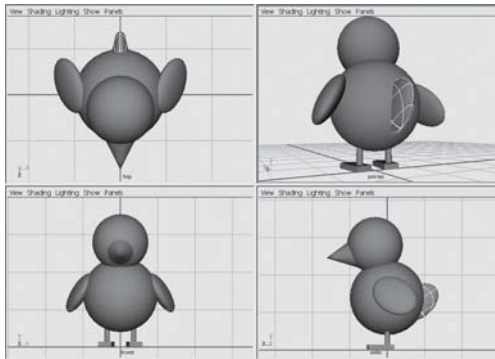


Figure 1.52: Tail rotation

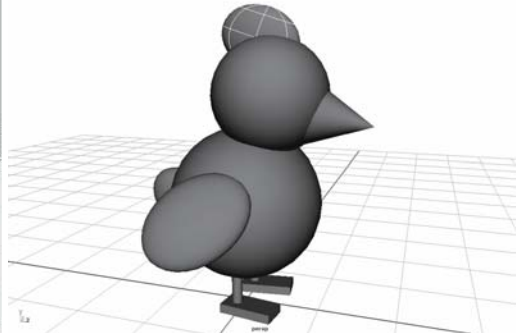


Figure 1.53: The crest

34. Scale the *crest* into shape. Scale = 0.17, 0.4, 0.52.

35. Rotate the *crest* 18° on the X-axis.

36. OK, you're in the home stretch now – time to make the eyes (see Figure 1.54). Click **Create > NURBS Primitives > Sphere**. Name this sphere *right\_eye*.

37. You know the drill by now: move, rotate, and scale the *right\_eye* into place. The settings are: Translation = -0.4, 2.92, 0.77; Rotation = 65, -40, 15; Scale = 0.2, 0.2, 0.2.

38. Repeat the above procedure to make the *left\_eye*. The settings stay the same for scale (unless you're building a crazy character; we all know that crazy characters have mis-

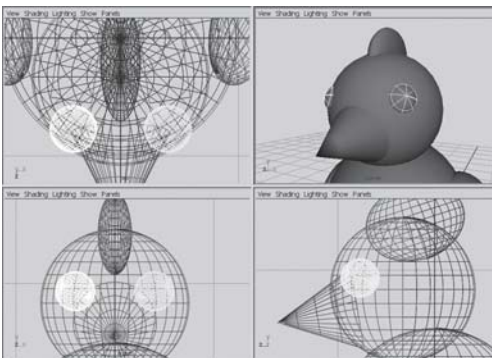


Figure 1.54: Eyeballs

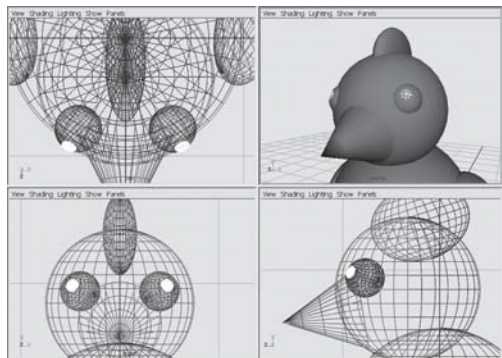


Figure 1.55: Pupils



matched eye sizes!) while Translation = 0.4, 2.92, 0.77, and Rotation = 65, 40, -15.

39. Make the two pupils (see Figure 1.55) from NURBS spheres as well, with these settings:

a. Name = *right\_pupil*; Translation = -0.48, 2.98, .92; Rotation = 65, -23, 8; Scale = 0.065, 0.03, 0.065.

b. Name = *left\_pupil*; Translation = 0.48, 2.98, .92; Rotation = 65, 23, -8; Scale = 0.065, 0.03, 0.065.

40. The eyebrows will be the finishing touch. For these we'll use the torus primitive (by the way, "torus" is fancy math-talk for "doughnut"). Click **Create > Polygon Primitives > Torus**. Rename the object *right\_brow*.

41. Translate, rotate, and scale the *right\_brow* into position (see Figure 1.56). We'll poke only a small portion of the torus out of the *head*. Translation = -0.4, 3.021, 0.75; Rotation = 97, -36, -4; Scale = 0.2, 0.12, 0.2.

42. For the *left\_brow*, create another polygonal torus, and use these values: Translation = 0.4, 3.021, 0.75; Rotation = 97, 36, 4; Scale = 0.2, 0.12, 0.2.

43. Save your scene, and step back a bit to admire your work (see Figure 1.57). Good job!

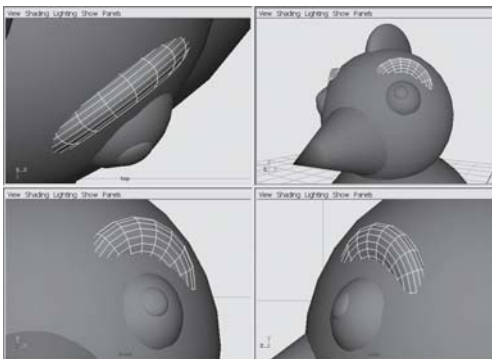


Figure 1.56: Eyebrow placement

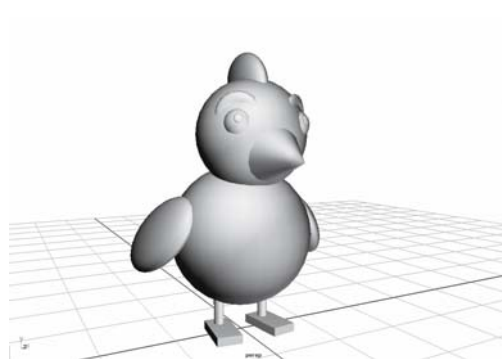


Figure 1.57: Finished chick model

### Modeling Exercise – Barn

In this modeling exercise we'll use the polygon and NURBS primitives as building blocks for a barn and silo. Numeric precision is a bit more important here than in the last exercise, because of the symmetrical construction of the building. We'll use the **Channel Box** to enter in exact values, and duplicate elements that appear more than once on the model.

1. Create a new project directory by clicking **File > Project > New...**
2. In the Name field enter the name Intro\_Project.
3. Click the **Use Defaults** button, followed by the **Accept** button.

4. Click **Create > Polygon Primitive > Plane options**. In the options box (see Figure 1.58), for both **Subdivisions Along Width** and **Subdivisions Along Depth** change the number to 1. All other settings should be at their defaults. Now, click the **Create** button. (Note: Throughout all the exercises in this book, you'll want to be sure to use default tool and action settings, unless otherwise specified. One way to ensure this is to click **Edit > Reset Settings** in each tool's options window before use.)

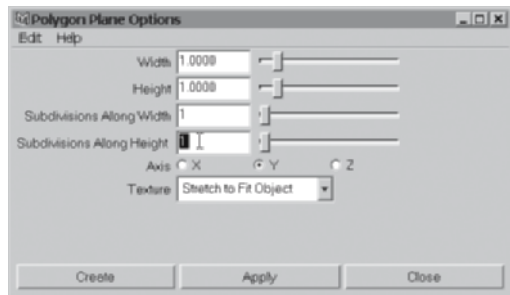


Figure 1.58: Polygon Plane Options box

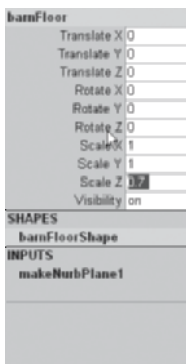


Figure 1.59: Scaling the plane

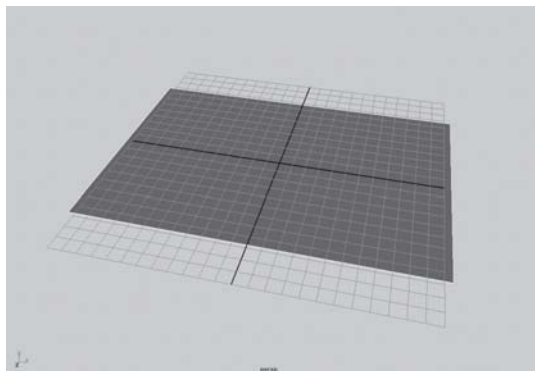


Figure 1.60: The scaled floor plane

5. In the **Channel Box**, set the *polyPlane1* object's Scale X = 25 and Scale Z = 17.5 (see Figures 1.59 and 1.60).

6. Once your scenes grow to a moderate level of complexity (which happens pretty quickly) it can be difficult to sift through all the different elements. In order to maintain organization, give your objects sensible descriptive names. These names will be associated with your objects in all of Maya's editors and listings. In the top of the **Channel Box** click on the name of your object to highlight it, then rename it *floor*. You are limited to naming your objects with normal alphanumeric characters and the underscore ("\_") symbol.

7. Next, create a polygon cube for the corner post. This can be done either by clicking **Create > Polygon Primitives > Cube**, or by clicking on the **Poly Cube** button on the **Shelf**. The shelf gives you quick access to commonly used tools and actions, and can be customized to fit your workflow.

8. Using the **Move Tool** and the **Scale Tool**, as well as direct numeric input in the **Channel Box**, move the new *pCube1* object to TranslateX -12.5, TranslateY = 5, TranslateZ = 8.75, and ScaleY = 10 (see Figure 1.61). These sets of coordinates can be listed as Translation = -12.5, 5, 8.75 and Scale = 1, 10, 1. Maya always presents XYZ in this same order.

9. Rename the object *barnPost01*.

10. Create another polygon cube.

11. Move and scale it to Translation = 12.5, 5, 8.75 and Scale Y = 10.

12. Rename it *barnPost02*.

13. Create another polygon cube, move and scale it to Translation = 12.5, 5, -8.75; Scale to 1, 10, 1 and rename it *barnPost03*.

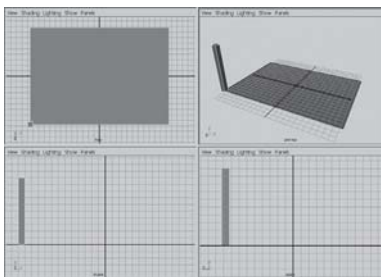


Figure 1.61: Creating a post

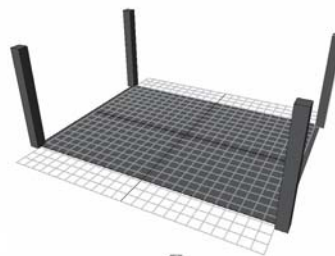


Figure 1.62: Adding the fourth post

## 28 1. Essential Skills

---

14. Create another polygon cube, move, and scale it to Translation = -12.5, 5, -8.75; Scale to 1, 10, 1 and rename it *barnPost04* (see Figure 1.62).

15. Next, you'll build some walls (see Figure 1.63). Create a polygon cube; Translation = 0, 5, 8.75; Scale = 24, 10, 0.5 and rename it *barnWall01*.

16. For the opposite wall, create a poly cube; Translation = 0, 5, -8.75; Scale = 24, 10, 0.5 and rename it *barnWall02*.

17. Here are the remaining walls (see Figure 1.64), all of which are made from polygon cubes:

- a. *barnWall03*: Translate -12.5, 5, 0; Scale 0.5, 10, 16.5.
- b. *barnWall04*: Translate 12.5, 5, 6.5; Scale 0.6, 10, 3.5.
- c. *barnWall05*: Translate 12.5, 5, -6.5; Scale 0.5, 10, 3.5.
- d. *barnWall06*: Translate 12.5, 9, 0; Scale 0.5, 2, 9.5.

18. Next, you'll make a couple of sliding doors (see Figure 1.65). Create a polygon cube: Translation = 13, 4, 5; Scale = 0.5, 8, 5. Name it *barnDoor\_Left*. The underscore is used to separate the elements of the name, since Maya doesn't allow spaces in object names.

19. For the other door, create a polygon cube: Translation = 13, 4, -5; Scale = 0.5, 8, 5.

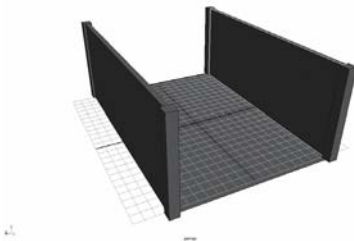


Figure 1.63: The first two walls

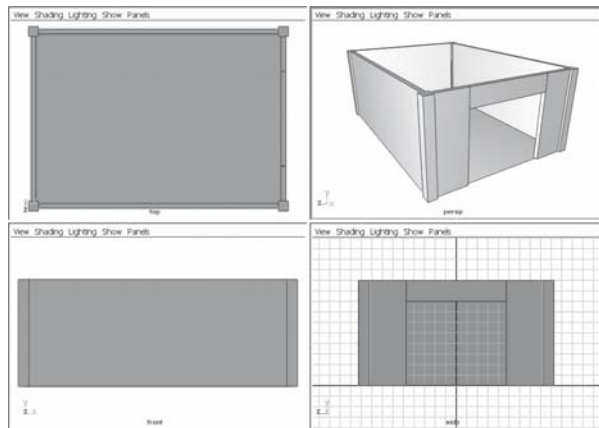


Figure 1.64: All four walls



Figure 1.65: Sliding doors

Name it *barnDoor\_Right*.

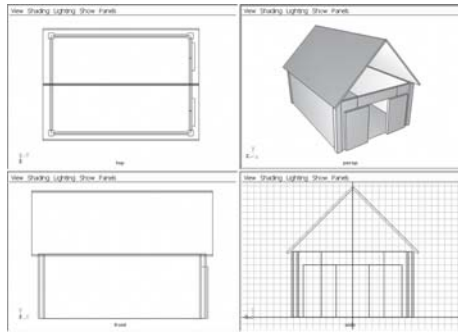


Figure 1.66: The roof

20. The roof will also be made from poly cubes, however, these will need to be rotated as well as translated and scaled (see Figure 1.66). Create a poly cube: Translation = 0, 14.75, 5; Rotation = 45, 0, 0; Scale = 28, 0.3, 14. Name it *barnRoof01*.

21. For the other side of the roof: *barnRoof02*: Translate 0, 14.75, -5; Rotate -45, 0, 0; Scale 28, 0.3, 14.

22. We'll use a cylinder to make the upper walls of the barn (see Figure 1.67). By reducing the number of subdivisions around the cylinder's axis, you can make a variety of useful primitive shapes. Click **Create > Polygon Primitives > Cylinder** options. In the options window, change the number of **Subdivision Around Axis** to 3. Click the **Create** button.

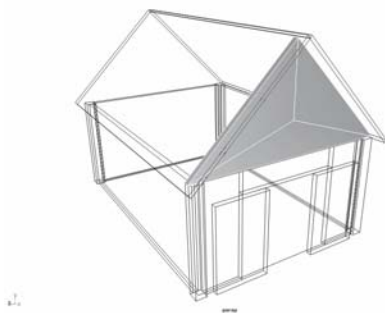


Figure 1.67: Three-sided cylinder wall

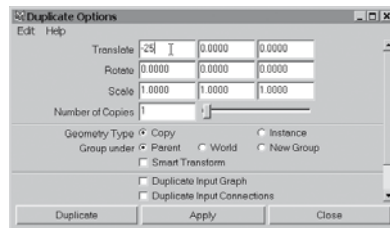


Figure 1.68: Duplicating the wall

## 30 1. Essential Skills

---

23. Transform the cylinder: Translate = 12.5, 13.2,0; Rotate = 0,0,90; Scale = 6.3, 0.25, 10.9.

24. Name the cylinder *barnWall07*.

25. In order to save time, you can take an object you have already created and duplicate it. Select *barnWall07* and click **Edit > Duplicate options** (see Figure 1.68). In the options window, be sure to reset the settings. The default settings will create a copy in the same place as the original. We want to make a new wall on the opposite side of the barn, so change the Translate X setting to a -25. (In most cases, when you see three data fields in a row, they refer to XYZ, so the first box here should change from 0 to -25.) This will make the copy and move it to the back of the barn in one step. Click the **Duplicate** button.

26. For added details, we'll build an air vent out of NURBS primitives. Click **Create > NURBS Primitives > Cylinder**.

27. Place the cylinder on top of the roof; Translate = 5, 19.5, 0, and rename it *barnVent01* (see Figure 1.69).

28. Now, make a hood for the vent. Click **Create > NURBS Primitives > Cone**. Instead of going to the options window to specify the cone's creation attributes, we'll use Maya's node-based architecture to alter some of the attributes after the cone has been created. In the **Channel Box**, look at the area marked **INPUTS**. As long as Construction History is turned on in the **Status Line**, you will see a node called **makeNurbCone1** in the bottom of the **Channel Box**. Click on it to reveal its attributes (see Figure 1.70). Change the **Radius** to 1.5, and the **Height Ratio** value to 1.

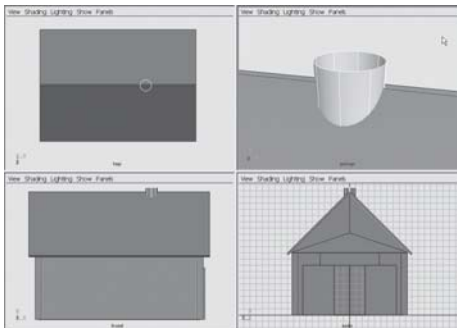


Figure 1.69: The vent

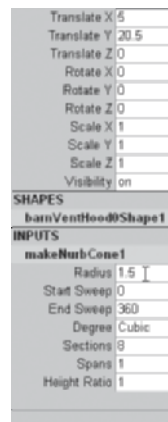


Figure 1.70: Hood radius

29. Translate the cone to 5, 20.5, 0. Rename it *barnVentHood01* (see Figure 1.71).

30. Duplicate the vent and hood: select *barnVentHood01*, then **shift-select** *barnVent01*. Click **Edit > Duplicate options**. Change Translate X to -10 and then click the **Duplicate** button.

31. We'll use a large cylinder to create a silo adjoining the barn (see Figure 1.72). Create a NURBS cylinder, Translate = 8, 12, -12; Scale = 4, 12, 4. Rename it *siloWalls*.

32. Top off the silo by creating a NURBS sphere, Translate = 8, 24, -12; Scale = 4, 4, 4. (Note that in this case there would be little practical difference between scaling the sphere uniformly to 4, and changing the input node **radius** to 4.) Name it *siloDome*.

33. For a bit of detail, we'll make a band that encircles the silo. Create a NURBS torus. In the input node, change the **radius** to 4 and the **height ratio** to 0.1.

34. Translate the torus to 8, 22, -12, and rename it *siloRing*.

35. You are done modeling the barn (see Figure 1.73), so click **File > Save Scene As...** and type in the name Barn01.ma. Click the **Save** button.

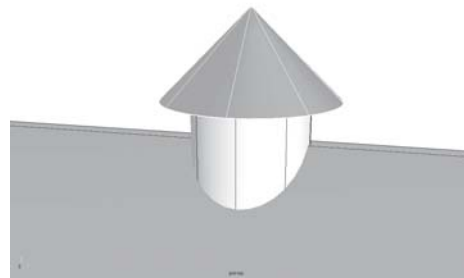


Figure 1.71: The vent hood

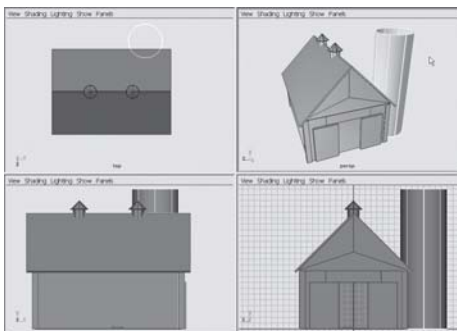


Figure 1.72: The silo

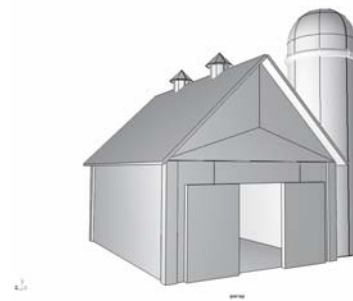


Figure 1.73: The finished barn model

## 1.4 Animation – Keyframe Basics



Without going into a history of animation or a discussion on persistence of vision, the base-level definition of animation we will work with is this: a change in value over time. This can be as simple as the sky's color turning from day to night, or as complex as a three-headed dog chasing a busload of screaming babies. The common thread is that over a period of time, things change.

In Maya, animation is generally created by specifying the key moments (or most important poses) when values change (see Figure 1.74). Maya automatically interpolates the values from one key to the next. So, if you want to move a box from the left side of the room to the right side over the course of three seconds, you would only need to place two keyframes. The first keyframe would mark the box's position at the left side of the room on frame one, and the second keyframe would mark the box's position on the right side of the room at frame 72.

Here is the order of operations for setting keyframe animation:

1. Go to a specific point in time.
2. Change an object's values.
3. Set a keyframe.
4. Go to a new point on the timeline.
5. Change the values.
6. Set another keyframe.
7. Hit the **play** button.

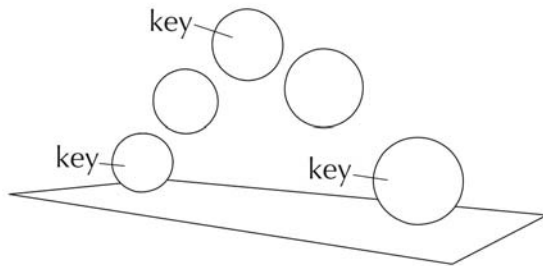


Figure 1.74: Simple animation

So, for each key pose you want to set for an object, you will place a keyframe. Learn the following phrase: Time-Value-Key. These are the three things you will do whenever you want to set a keyframe. Time – go to a frame on the timeline. Value – change the value of an attribute, such as translation, rotation, or scale. Key – set a keyframe.



## Animation Exercise – Ball Drop

For your first animation exercise you will animate the position of a ball. By setting keyframes at the beginning, middle, and end of the time range you will cause Maya to interpolate the values between the keys. This automated process is called in-betweening.

Before you begin, it's always a good idea to double-check the animation settings. Click **Window > Settings/Preferences > Preferences...** Click on the **Timeline** category. In the **Playback** preferences section, change the Playback Speed to Real-time [24 fps]. Click **Save** when you are done. This will allow you to more accurately evaluate your performance during animation playback.

1. Create a NURBS sphere.
2. Name it *ball*.
3. *Time* – set the Time Slider to frame 1 (see Figure 1.75).

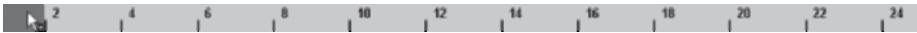


Figure 1.75: Time = Frame 1 on the timeline

4. *Value* – using the Move Tool, translate the ball to a value of 1 on the translateY axis (see Figure 1.76). You can type an exact value in the Channel Box; click in the data field to the right of the Translate Y channel name, type a **1**, and then press **Enter**.

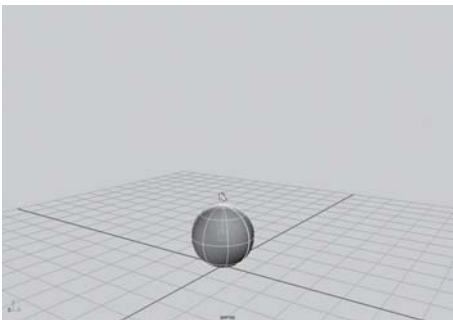
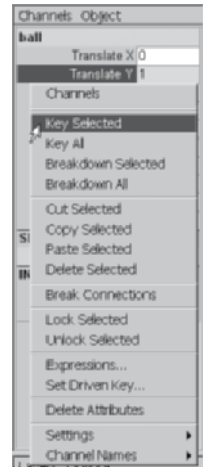


Figure 1.76: Value = 1 on the Y-axis

## 34 1. Essential Skills

5. *Key* – With the ball still selected, click on the Translate Y channel name. This highlights the channel. Now, **RMB**-click the Translate Y channel name. (While the **RMB** is held you will see a drop down contextual menu that allows you to select many animation-related actions.) From the **RMB** menu, drag the cursor over **Key Selected** and release the **RMB** as in Figure 1.77.



6. You will see the Translate Y channel's data field turn orange. This indicates that you have successfully set a keyframe.

7. Follow the pattern again. *Time* – set the Time Slider to frame 12.

8. *Value* – the ball goes up. Use the Move Tool to lift the ball up to 6 on the Y-axis (see Figure 1.78).

9. *Key* – In the Channel Box, highlight the Translate Y channel name. **RMB** click-drag the channel and choose **Key Selected** from the dropdown menu.

Figure 1.77: Key the frame

10. You've just placed two keyframes, which is the minimum requirement for some sort of animation to take place. Click the Rewind button and then the Play button to see your animation (see Figures 1.79 and 1.80). Since the Range Slider is currently set to play frames one through 24 before looping back to the beginning, you will see the ball start on the ground, rise for 12 frames, and then hold for 12 frames. When done, press the Stop button.

11. Time to set one more keyframe, so: *Time* – drag the Time Slider to frame 24.

12. *Value* – Put the ball back on the ground with the Move Tool; Translate Y = 1.

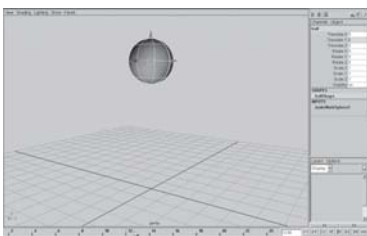


Figure 1.78: Value = 6 on the Y-axis



Figure 1.79: The rewind button



Figure 1.80: The play button

13. *Key* – Highlight the Translate Y channel in the Channel Box, **RMB**-click, and choose **Key Selected**.

14. Press the Play button to see your animation. The ball now bounces up and down as the animation playback range loops.

Frame 1: Translate Y = 1.

Frame 12: Translate Y = 6.

Frame 24: Translate Y = 1.

In Figure 1.81 we see the final animation with some of the automatically in-betweened frames.

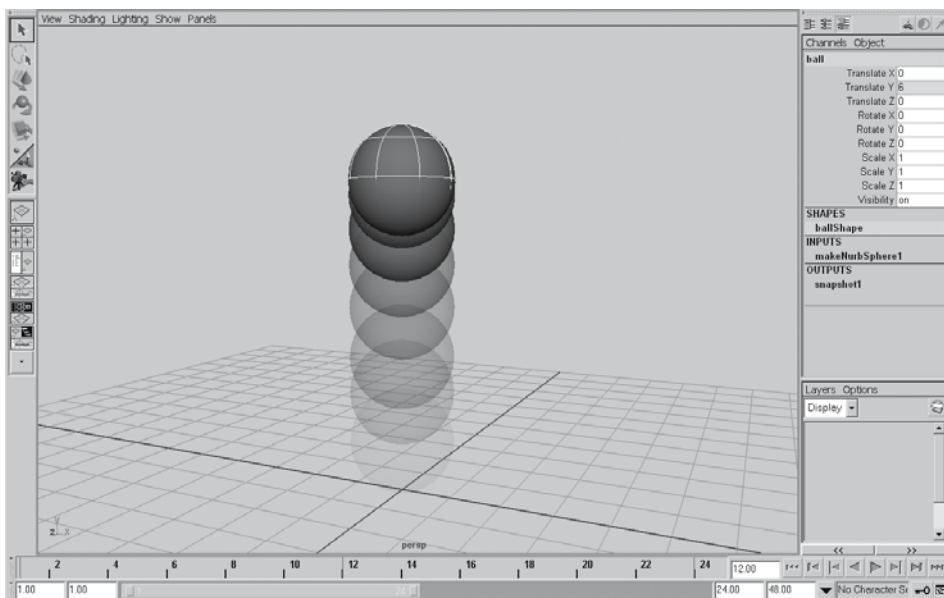
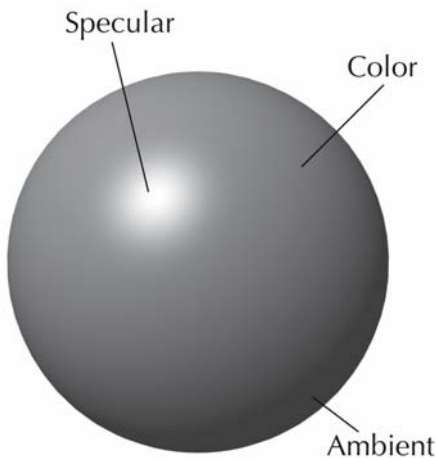


Figure 1.81: Finished animation time-lapse

## 1.5 Shading — Color, Specular, and Ambient



Shading defines the look of your model when it is rendered. We'll begin with a common shading style, called Phong shading, and look at three of its attributes (see Figure 1.82): color, specular color, and ambient color. Later we will add textures, texture maps, transparency, glows, and other effects.

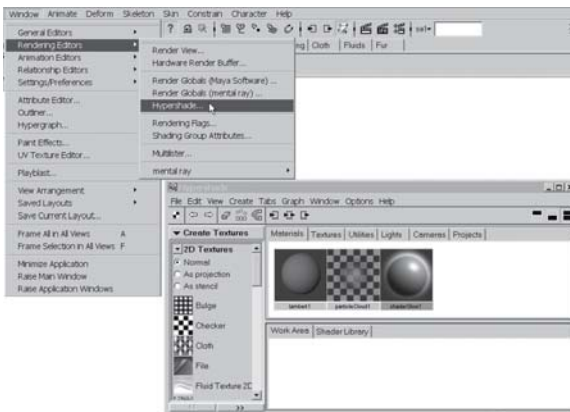


**Color** is the main color of the material where light strikes it.

**Specular** is the bright highlight region of the surface material.

**Ambient** is the color of a section that receives no light. Normally black, but can be brightened for effects like self-illumination.

Figure 1.82: Basic shader components



Most of your shader creation will take place in the Hypershade. Go to **Window > Rendering Editors > Hypershade...** to open up a floating Hypershade window (see Figure 1.83).

Figure 1.83: Hypershade window

## Shading Exercise – Shading Donuts

1. Set your project to `understanding_maya\Chapter01\shading_Donuts`.
2. Open the `phong_donuts_begin.ma` scene.
3. Highlight the **Perspective** view panel and choose the Smooth Shade All option (keyboard Hotkey **5**) so you see the scene in color (see Figure 1.84).
4. When you are working, it's a good idea to select a view panel arrangement that matches the task. In this case, two stacked panes with the Hypershade and Perspective panels works well (see Figure 1.85). You can pick this arrangement by clicking on a panel menu's **Panels > Saved Layouts > Hypershade/Persp**. Or, better yet, click on the Hypershade/Persp button in the Quick Views section of the Tool Box.



Figure 1.84: Setting smooth shading

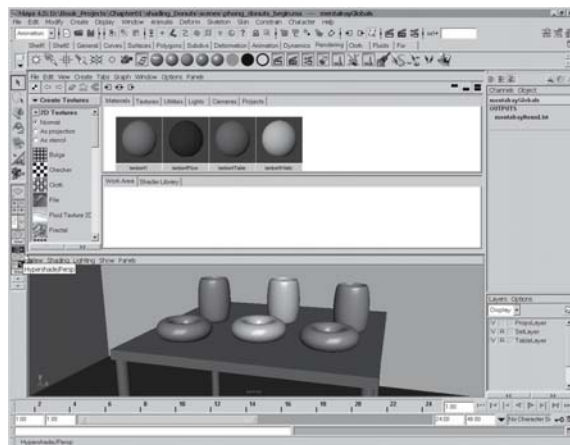


Figure 1.85: Hypershade/Perspective layout

5. In the Hypershade, look at the Materials tab area. Here you'll see all of the Materials that are in the scene. Pay close attention to the *phongRed*, *phongGreen*, and *phongBlue* materials. These materials have been applied to the three doughnuts and cups on the table.
6. The relationship between materials and surfaces is a good example of Maya's node-based architecture. For example, the *phongGreen* material is connected to the *MiddleCup* object and the *MiddleDonut* object. If the connected attributes (in this case, the color green) change on the material node, we will see a change on the objects. Double-click on the

## 38 1. Essential Skills

---

*phongGreen* material to open the Attribute Editor. In the Common Material Attributes section, click on the green color swatch once to open the Color Chooser window (see Figures 1.86 and 1.87). Using the color wheel, select a new color. Notice how this color changes the material in the Hypershade as well as both objects in the Perspective window. It is important to remember that any changes you make to a material will affect *all* objects that have been assigned that material.

7. Close the Color Chooser window by clicking the **Accept** button. Close the Attribute Editor.

8. Next, assign the *phongBlue* material to the *LeftDonut* object. One easy way to do this is to **MMB** click-drag the *phongBlue* material sample onto the *LeftDonut* in the perspective panel. When you release the **MMB**, the *LeftDonut* will change from red to blue.

9. In the Hypershade menu, select **Create > Materials > Phong** to make a new material. Double-click the new material to open the Attribute Editor. In the “*phong*” field at the top, rename the material *phongYellow*.

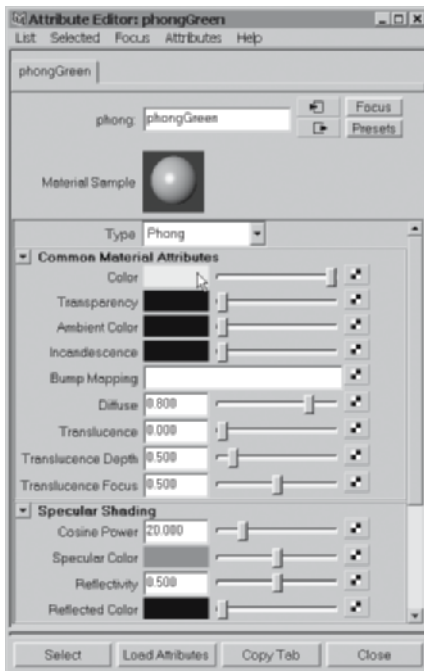


Figure 1.86: Material settings



Figure 1.87: The Color Chooser

10. In the Common Material Attributes section, click on the grey color swatch to open the Color Chooser window. Pick a yellow color and click **Accept**.
11. Close the Attribute Editor and then **MMB** click-drag the *phongYellow* material onto the *RightCup* object.
12. Create another Phong material, this time naming it *phongMagenta*.
13. In the Color Chooser, change the color to magenta (300, 1, 1 in the HSV fields as in figure 1.88).
14. Change the Ambient Color to white by dragging the slider to the right, and then close the Attribute Editor.
15. Drop this material onto the *MiddleCup* object (see Figure 1.89).
16. Create one more Phong material, and name it *phongAqua*.
17. In the Color Chooser, change the color to an aqua-blue color (HSV 180, 0.5, 0.9).
18. Change the Specular Color to bright green (HSV 120, 1, 1).
19. Drop this material onto the *LeftCup* object.
20. Save the scene by clicking **File > Save Scene As...** and name it *phong\_donuts\_end.ma*.



Figure 1.88: HSV color fields

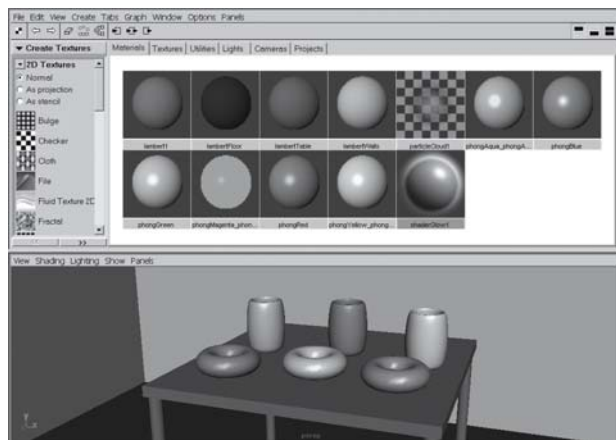


Figure 1.89: Middle cup material assignment

## 1.6 Lighting — Point Lights: Intensity, Color, and Shadows



Lighting in Maya is much like lighting for theater or live action film. To illuminate a scene, you must place lights in space, brighten or dim them, color them, and set countless other of their attributes. You can think of lighting in realistic terms: *How high should this light be to cast a flattering glow on my subject? Where will light come from to fill in harsh shadows?* You can also go well beyond reality. *Would my scene look better if I had a light that tinted my background without affecting the subject? What if some lights cast no shadows?*

Maya allows you to create six different types of lights for use in a wide variety of lighting scenarios. We'll look first at the simple point light. It's a lot like a normal household light bulb, in that it casts lights in all directions from a single source. By varying its position, intensity, and color you can achieve many moods and effects in your scene. Figure 1.90 shows a single point light in the center of some objects. Notice the omni-directional pattern of light thrown.

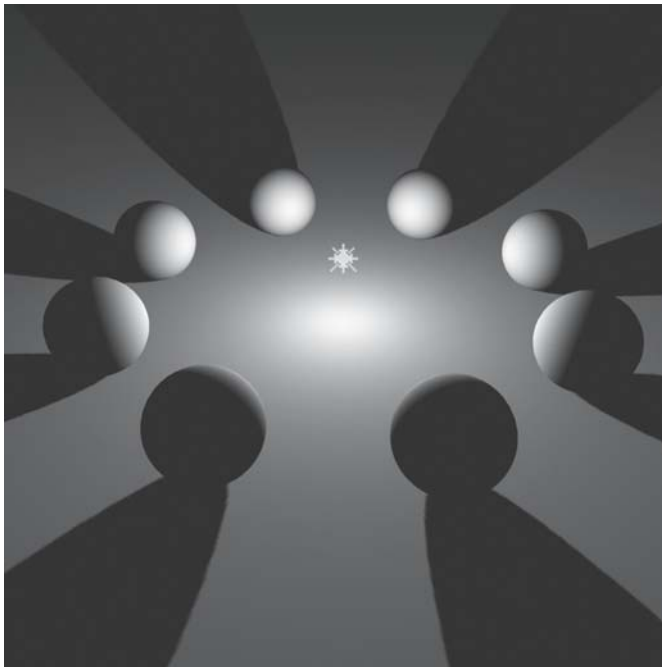


Figure 1.90: Point light throw pattern



## Lighting Exercise – Lighting Donuts

1. Set your project to `understanding_maya\Chapter01\lighting_Donuts`.
2. Open the scene `pointlight_donuts_begin.ma`. This scene looks just like the scene you worked with in the last shading exercise, with the addition of a white globe object. (With its neutral shape and color, this object will help you evaluate the scene lighting.)
3. Click **Create > Lights > Point Light** to place a new point light in the scene.
4. In the perspective view, press **5** to turn on smooth shading (see Figure 1.91), then press **7** to turn on hardware lighting (see Figure 1.92). Notice how the scene is being up-lit by the point light, sort of like an incandescent bulb.
5. With the *pointLight* selected, use your move tool to raise it above the table 30 units on the Y-axis.
6. Try moving the *pointLight* back and forth on the X-axis, paying attention to how the scene lighting changes.
7. An important note on point lights: They cannot be selected by direct clicking in a view panel. You must click-drag a marquee around them in order to make the selection.
8. The default point light in Maya casts white light at an intensity of 1. To change these values, select the *pointLight* and open the Attribute Editor by pressing the hotkey combination **Ctrl-a**.

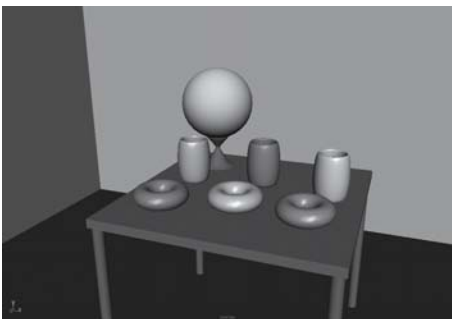


Figure 1.91: Default lighting



Figure 1.92: Point lighting

## 42 1. Essential Skills

9. In the **Point Light Attributes** section of the **Attribute Editor**, click on the **Color** swatch (see Figure 1.93). Drag the **Attribute Editor** and the **Color Chooser** so that they are out of the way and you can see the scene in your **perspective** view. Click-drag in the color wheel and notice how the entire scene is affected by tinting the light.

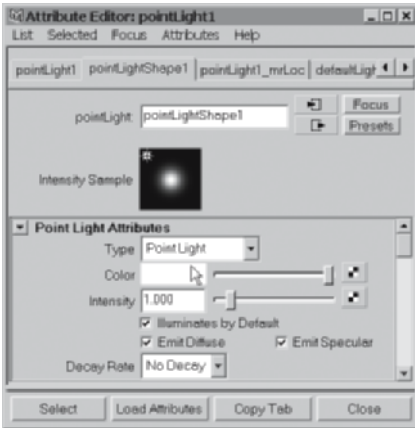


Figure 1.93: Adjusting color

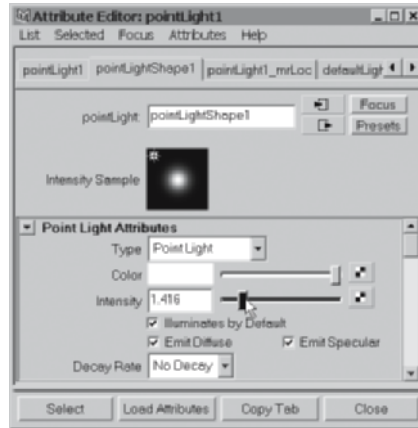


Figure 1.94: Adjusting intensity

10. Unless you're going for an extreme effect or mood, a small color tint on your lights goes a very long way. With overly strong light color, all of your careful selection of material colors will be thrown out the window.

11. Close the **Color Chooser**. Now, try varying the intensity of the *pointLight*. In the **Attribute Editor**, slide the intensity slider (see Figure 1.94) lower and higher, paying attention to the effect on your scene. You'll probably notice that even small adjustments can have a great impact on your scene lighting. As in Figure 1.95, it's easy to either "blow out" your scene with too high an intensity value or shroud your scene in darkness with too low an intensity value.

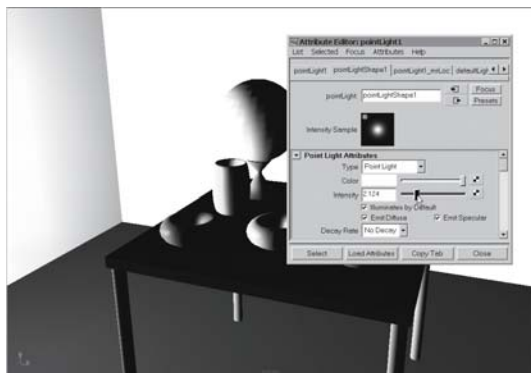


Figure 1.95: Too much intensity

12. In Maya, nearly every attribute of a node can be animated. The transform node of the `pointLight`, for example, can be keyframed for position on its X, Y, and Z translate channels. The shape node of the `pointLight` can be keyframed to brighten the light over time on its Intensity channel.

13. Set the **Range Slider** to 72 frames (see Figure 1.96).



Figure 1.96: The range slider set to 72 frames

14. Go to frame 1 of the timeline.

15. Move the `pointLight` to -16, 10, 0.

16. Set a keyframe for `translateX`, `translateY`, and `translateZ`.

17. Advance to frame 36.

18. Move the `pointLight` to 0, 20, 0.

19. Set a keyframe for each translate channel.

20. Advance to frame 72.

21. Move the `pointLight` to 16, 10, 0.

22. Key the translate channels.

23. Play the animation to see the light arc over the scene.

24. Rewind to frame 1.

25. In the **Channel Box**, look at the `pointLightShape` node channels. Find the **Intensity** value. Change the value to 0.3. This will darken the lighting at the beginning of the shot.

26. Set a keyframe for the **Intensity** channel.

27. Advance to frame 22.

## 44 1. Essential Skills

---

28. Increase the **Intensity** to 1.

29. Key the **Intensity** channel.

30. Advance to frame 46.

31. Leave the **Intensity** set to 1, but set a keyframe to hold the value during the middle of the animation.

32. Advance to frame 72.

33. Decrease the **Intensity** back down to 0.3.

34. Key the **Intensity** channel.



Figure 1.97: Frame 1



Figure 1.98: Frame 18



Figure 1.99: Frame 36



Figure 1.100: Frame 54

35. Play your animation. Now, the scene brightens, holds, and then dims while it arcs over the scene, as in Figures 1.95 - 1.101. If you don't see light levels changing during playback, make sure that you place the panel into shaded and lit mode (5 and 7 on the keyboard).

36. Stop the playback.

37. When you are finished, save your scene as `pointlight_donuts_end.ma`.

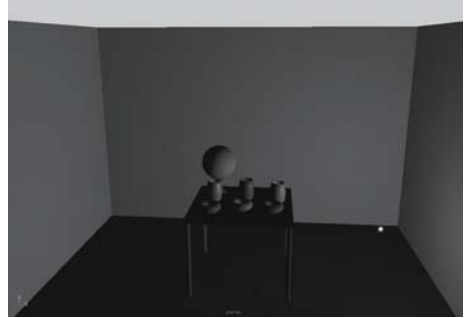


Figure 1.101: Frame 72

## 1.7 Rendering — Outputting a Scene to Frames



Your final step in creating a 3D animation will be to produce a sequence of production-quality frames that can be output to video, film, or a compressed video format such as an MPEG-2 QuickTime movie. When you instruct Maya to render your animation, it takes all of the data in the camera's view and calculates a per-pixel bitmap image for each frame of the animation (see Figure 1.102).

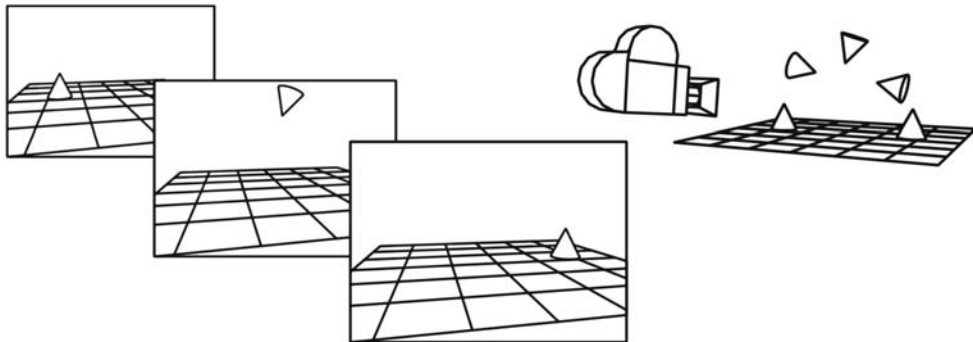


Figure 1.102: Rendering a sequence of frames

### Rendering Exercise – Rendering Donuts

1. Set your project to `understanding_maya\Chapter01\rendering_donuts`.
2. Open the scene `render_donuts_begin.ma`. This scene is like the `lighting_donuts_begin.ma` scene, except the globe has been removed and the perspective view camera has been locked down so it can't be moved accidentally.
3. Make your **perspective** view full-sized, and press **5** and **7** to shade and light the view panel.
4. Advance to frame 26. We will do some test renders from this frame.
5. Switch to the Rendering Menu Set (see Figure 1.103). Click **Render > Render Current Frame...** (see Figure 1.104).
6. The **Render View** window opens up, and the current frame of your animation is rendered into it as in Figure 1.105. These default settings are small and of low quality. Let's make things look better.
7. Close the **Render View** window.
8. The resolution, quality, format, naming, and other settings for your renders are edited in the Render Globals window. Click on **Window > Rendering Editors > Render Globals...** (see Figure 1.106).
9. The resolution of a frame is described by its number of horizontal pixels and vertical pixels. Open the **Resolution** turn-down menu. Change the **320x240** preset to **640x480**.

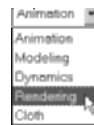


Figure 1.103: Rendering menu set

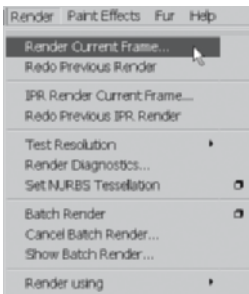


Figure 1.104: Render current frame

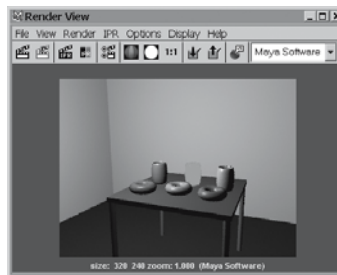


Figure 1.105: Render view

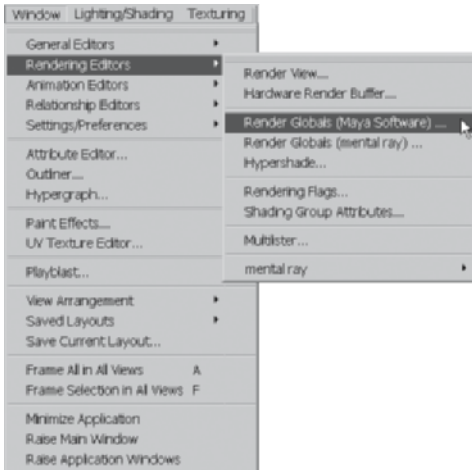


Figure 1.106: Opening Render Globals

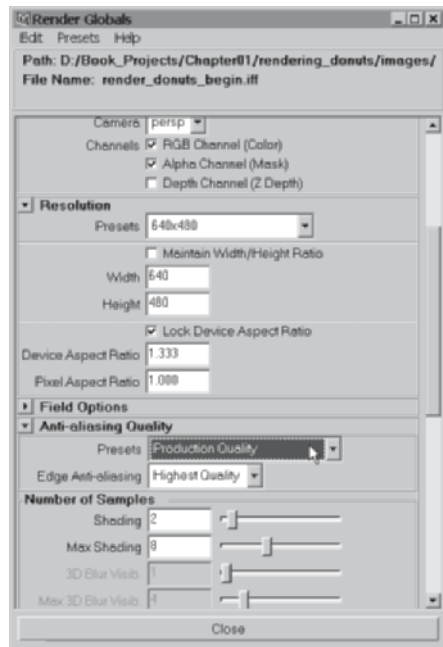


Figure 1.107: The Render Globals

10. Open the **Anti-aliasing Quality** turn-down menu (see Figure 1.107). Select the **Production Quality** preset. (In Maya 5.0 and later, this is found in the “Maya Software” tab.)

11. Close the **Render Globals** window.

12. Re-render the current frame, this time by clicking the first small clapboard icon in the **Status Line**. This frame is of greater size and quality (see Figure 1.108).

13. Close the **Render View** window.

14. Now let's add some shadows to the render. Shadows are not turned on by default in Maya, but they can be activated in each individual light. (There are many shadow settings beyond on/off; we'll be looking



Figure 1.108: Anti-aliased render

more closely at shadow properties in Chapter 2.)

15. Select the *pointLight1* and then open the **Attribute Editor**. Open the **Shadows** turn-down menu. Within the **Depth Map Shadow Attributes** turn-down menu, click the **Use Depth Map Shadows** box (see Figure 1.109). We'll accept all of the default settings of the depth map shadow. Close the **Attribute Editor**.

16. Re-render the current frame by clicking the first clapboard icon (see Figure 1.110). Notice how the shot looks more natural, and the objects are more grounded in the scene (see Figure 1.111). Without shadows, things often seem to float unnaturally.

17. Close the **Render View** window.

18. Rewind the **Time Slider**. Test render the first frame. It's always a good idea to test render a few frames at the beginning, middle, and end of an animation before you tell Maya to render the entire sequence. This way you can spot any problems of lighting or camera work before committing to a long batch render process.

19. After you are satisfied that there are no problems in the lighting at the beginning, middle, or end of the shot, it is time to set up a batch render. There is little practical difference between preparing individual frames the way we have been doing, and setting up a batch render.

You could, in theory, render an entire shot manually: move to frame 1; render a frame; save the image to disk; repeat. Now, that would get tedious pretty quickly, so we let

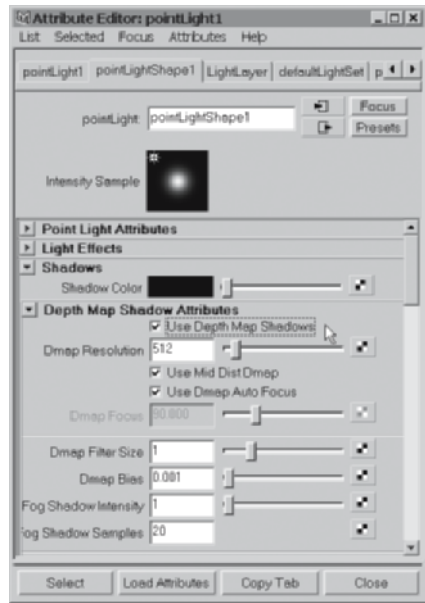


Figure 1.109: Turn on shadowing



Figure 1.110: Render frame button



Figure 1.111: Shadow render



the software do it for us. The batch render process renders each frame of animation sequentially and stores each frame onto your hard drive. Once finished, these frames can be viewed flipbook-style or further edited using a compositing program or video editor.

20. Open the **Render Globals** window. You can use the third clipboard icon in the status bar (see Figure 1.112).



Figure 1.112: Render globals button

21. At the top of the **Render Globals** window (in Maya 5.0 and later, this is found in the “Common” tab) you can see the path where your finished frames will be stored. This should point toward the images directory of your current project. The file name allows you to preview what your naming convention will be for the frames.

22. In the **Image File Output** turn-down menu, change the file name to **donuts**.

23. Next, in the **Frame/Animation Ext** dropdown menu, choose **name.#.ext**. This indicates to Maya that you plan to render a sequence of frames.

24. The **Start Frame** 1 value is correct, but you need to change the End Frame value to 72, the number of your last frame.

25. A value of 1 in the **By Frame** option means that you will not be skipping frames, such as a stop-motion or cell-animation done on “two’s” or “three’s.”

26. Frame Padding should be set to 4. Watch the **File Name:** area when you make this change. Some operating systems and applications will incorrectly order the numbers 1 to 20 like this: 1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2, 20, 3, 4, 5, 6, 7, 8, 9. Notice how terribly out of order this “alphabetizing” leaves us. By padding the frame four places we get this: 0001, 0002, 0003, 0004, 0005, 0006, 0007, 0008, 0009, 0010, 0011, 0012, 0013, 0014, 0015, 0016, 0017, 0018, 0019, 0020.

27. The **Image Format** can be left at Maya IFF (iff). Now, the **File Name:** area indicates that you will be rendering donuts.0001.iff through donuts.0072.iff.

28. To decrease the render time, change the **Resolution** preset to 320x240.

29. Once these settings are entered (see Figure 1.113), close the **Render Globals** window.

## 50 1. Essential Skills

---

30. To begin the batch render, click **Render > Batch Render** (see Figure 1.114). You won't see individual frames render into the Render View during a batch render (Maya's busy rendering!) but the **Command Feedback** line will indicate the percentage of progress for each frame. As it finishes a frame, it gets stored in your project's images directory.

31. Render times are dependent on the overall speed of your workstation; this is a simple scene, so the batch render should only take a few minutes to render. (Later, you'll find it's necessary to weight the benefits of "expensive" render tricks – such as raytraced reflections or large numbers of lights – against the increase in render time.) When it is complete, the **Command Feedback** line will read Result: Rendering Completed.

32. In order to view the sequence, launch FCheck, the Alias frame viewer. This can be launched from a system shell by typing "fcheck", from the fcheck icon in your bin directory or from the Windows Start Menu > Run... command as in Figure 1.115.

33. In FCheck, click on **File > Open Animation ...** Browse to the understanding\_maya\Chapter01\rendering\_donuts\images directory. Click on the donuts.0001.iff frame to highlight it, and click the **Open** button.

34. FCheck will load the sequence of frames into your system RAM, and then play it back. You should see the light and shadows arcing across the scene. In FCheck you can **LMB** click-drag the playback window in order to scrub back and forth through your animation. Release the **LMB** to resume play.

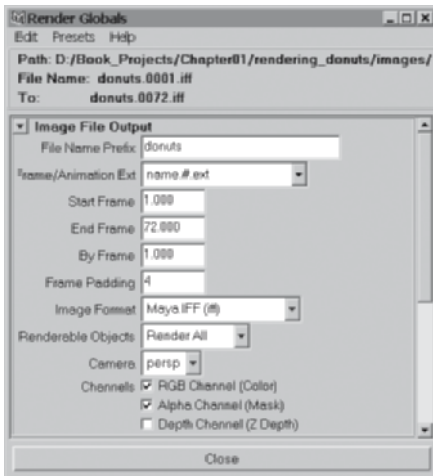


Figure 1.113: Render globals

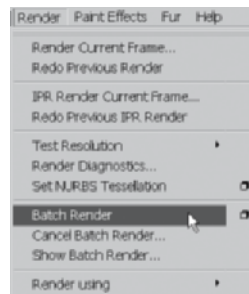


Figure 1.114: Batch rendering

35. When you are done watching, close FCheck.

36. In Maya, save your scene as **render\_donuts\_end.ma**.

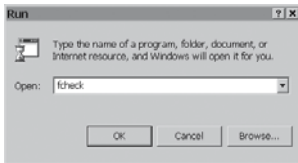


Figure 1.115: Launching fcheck

## 1.8 Project 1 — Ball Rolling Down an Inclined Plane

For your first project, create and render a three-second animation of a ball rolling down a tilted board. Be sure to use all the skills you've learned so far in the areas of modeling, animation, shading, lighting, and rendering.

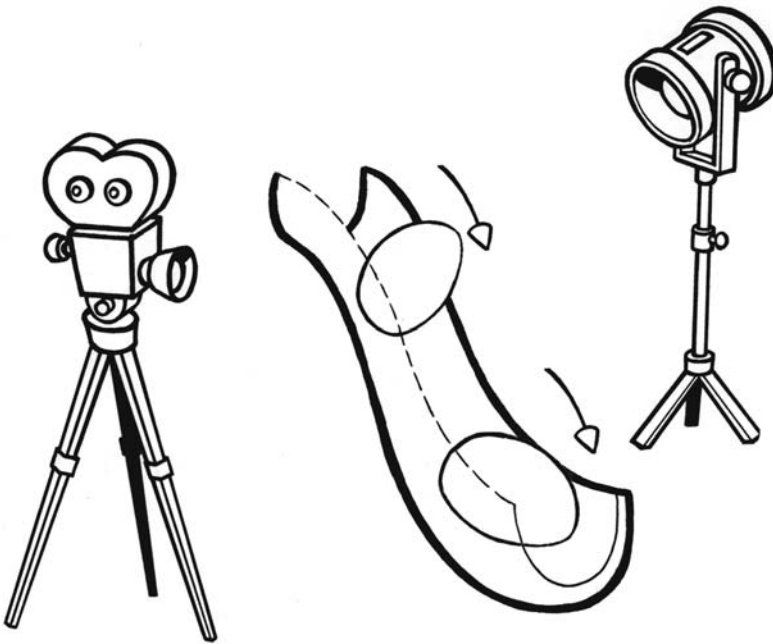
Include:

- Geometric primitives,
- Keyframed animation,
- Colored Phong shaders, and
- Point lights with shadows.

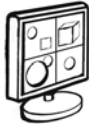
---

# Chapter 2

## Refinements



## 2.1 Workspace – Organization



If you’ve ever read through the Alias manuals, you may remember this line: “Maya is nodes with attributes that are connected.” It keeps coming up again and again due to its importance in defining the way the software works. It also bears mentioning that there are often a few different terms used to convey the same concepts.

One example of this is the word **Channel**. A channel is really no different than a keyable attribute. Maya’s nodes are filled with both keyable and non-keyable attributes. For any given node, the **Channel Box** (see Figure 2.1) is the list of the most commonly keyframed attributes. The **Channel Box** gives you access to these values, and it allows you to change and set keyframes for them.

Also present in the **Channel Box** are the nodes to which the model is connected. Here is a simple example. The Transform node is the top node of an object. The Transform node defines the movement, rotation, and scale of an object. Below that is the Shape node, which stores information about the physical appearance of an object (such as vertex tweaks). At the bottom of the **Channel Box** are the Input nodes. These represent the initial creation of the object, as well as other operations that have been applied. If you were to change one of these Input node attributes, such as Radius or End Sweep of a NURBS sphere, you would see the object change just as if you had originally created the sphere with those attributes.

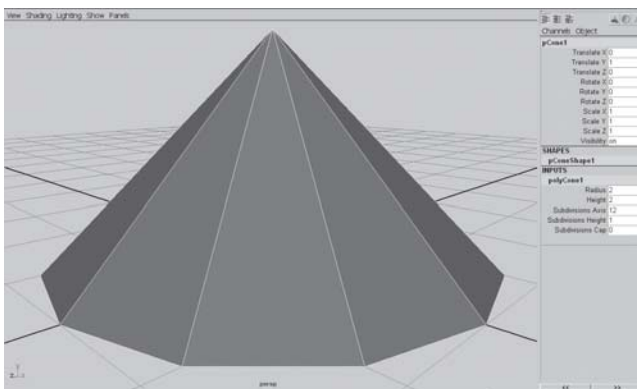


Figure 2.1: The Channel Box and Perspective Panel

## The Outliner

As you have already seen, the Maya workspace has many different ways of presenting information about the objects in your scene. One such way is the **Outliner**. At its most basic, the Outliner is a list of the objects in your scene. Select **Window > Outliner . . .** to open up a new **Outliner** window.

In the scene shown in Figure 2.2, we can see that the **Outliner** lists the four viewports, the primitive objects, the point light, and the default light and shader sets. Note that the small icons help you to identify the type of object. For example, the NURBS primitives have a small patch icon, while the polygon plane primitive has a more grid-like icon.

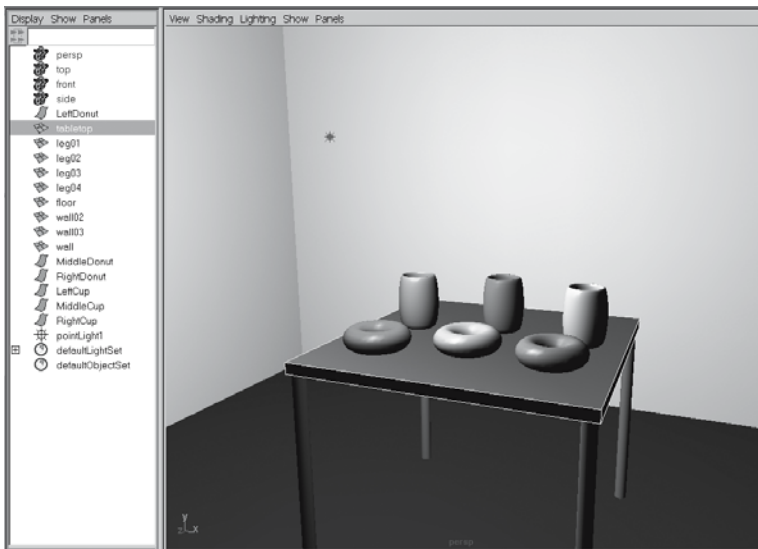
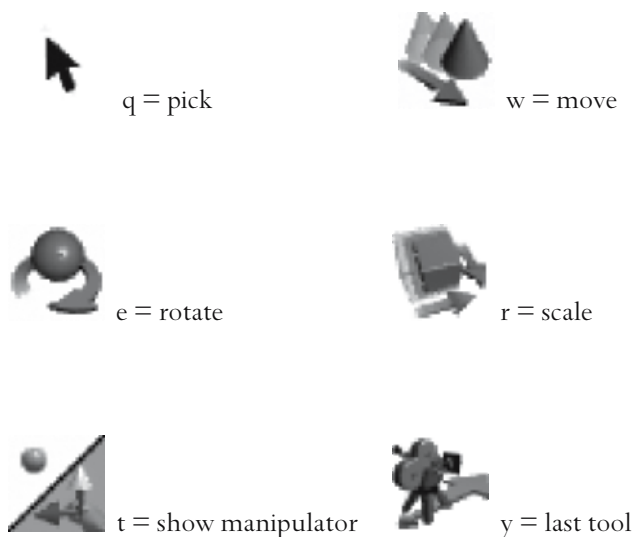


Figure 2.2: The Outliner and Perspective Panel

Try picking objects in the **Outliner** and watch as they are selected in the **perspective** panel. This can be very helpful in complex scenes where direct object selection is difficult. There are other uses for the **Outliner** which we will explore in later lessons, but for now you'll use it for picking objects.

### Keyboard Shortcuts for Transformations

The transform tools and their corresponding locators are a necessity when it comes to doing much of anything within the Maya workspace. They're so important that they have their own set of convenient keyboard shortcuts. This way you don't have to select them in the toolbox every time you want to change among selecting, moving, rotating, and scaling modes. The keyboard shortcuts are QWERTY.



These manipulators all allow you to constrain your operation along a specific axis:

- Red = X axis;
- Green = Y axis;
- Blue = Z axis;

Yellow represents the currently selected axis. For two-axis translation, you can **ctrl + LMB** click an axis to remove it from the constraint set, thereby limiting the operation to the other two axes. This can be especially useful when animating an object while working in a **perspective** or **camera** panel.

## Project and File Management

Project and file management can be one of the most important aspects of making a production run smoothly. Keeping track of the scenes, source images, textures, sound files, custom UI elements, and rendered images can be a real chore. Maya has a default file structure to aid your organization of these assets.

Whenever you start a new project (see Figure 2.3), you should go to the **File > Project > New...** menu to create a new directory structure.

Name your project (see Figure 2.4) and then accept the default settings, and Maya will create a new project directory with a series of subdirectories. This is where Maya will go to save scenes, browse for textures, and output rendered images.

Think of the project in these terms: your Maya scene (.ma, .mb) contains geometry, animation, lights, material, and other information that you create inside Maya. Scenes are stored in the **scenes** directory of your project.

All elements created outside Maya – such as textures you have painted in Photoshop or audio files for synching with your animation – need a place to reside that your scene can easily find. The project directory structure has a **sourceimages** directory for your textures (.jpg, .tga, .iff, .bmp, etc.), a **sound** directory for your audio (.wav, .aiff), and so on. You establish a relationship between your scene and these elements, but it is never anything more than a link between two elements. Your texture and sound files are never embedded into your scene file; they are merely pointed at. This is why the path established using the project directory structure is so impor-

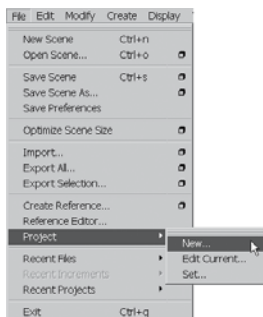


Figure 2.3: New project

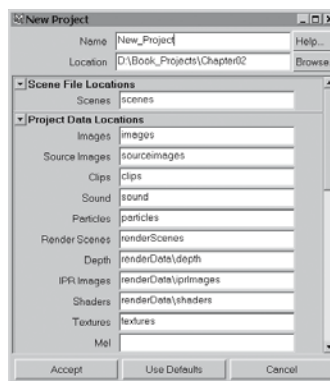


Figure 2.4: Project naming





Figure 2.5: Project directories

tant (see Figure 2.5). If an element is moved or deleted from the directory your Maya scene expects to find it in, you'll get an error, and the scene may not render correctly.

Even things that are output from your Maya scene, such as rendered frames, need a home. For this, the project contains an **images** directory. Here, you'll find your finished frames after an animation has been batch rendered.

Here is an example of file management in action. Let's say you are beginning work on a short film entitled "The Chicken Farm." First you would launch Maya, and then click **File > Project > New...** to bring up the New Project window. You could name the project **ChickenFarm**, and then choose the path where you would like the project stored. (The default is in your user account's **maya/projects** directory.) Next, click the **Use Defaults** button, to automatically name the directories where different project elements will be stored. Click the **Accept** button. This tells your operating system to build the set of directories needed.

Now, if you sketch some concepts for your character, paint some textures, or create model sheets, they can all be saved into the **ChickenFarm/sourceimages** directory. When you record a dialog track, it can be saved into the **ChickenFarm/sounds** directory. In Maya, when you begin working on the first scene, you will click **File > Project > Set**. Browse to the **ChickenFarm** project and click **OK**. When you then save your scene, the **ChickenFarm/scenes** directory will automatically be chosen as your save directory. If you need to import your character sketches for reference, the **ChickenFarm/sourceimages** directory will be selected. When you need to place your audio file into the timeline, the correct directory will open first. Finally, when you render the animation, Maya will send your finished frames to the proper **ChickenFarm/images** directory.

## 2.2 Modeling – Geometry Types



Geometry comes in a few different flavors. Maya can be used to create NURBS, polygonal, and subdivision surface geometry. Each has its own particular strengths and weaknesses; learning to use all three types is important. Here's a broad oversimplification that'll get a bunch of people all worked up.

- NURBS are good for creating organic squishy objects.
- Polygons are good for hard-edged, man-made objects.
- Subdivision surfaces are good for everything.

OK, that said, let's take a closer look.

### NURBS

NURBS stands for Non-Uniform Rational B-Spline. Unless you are on CG Mathematician Jeopardy, you'll never need to know what that means. The important part is that NURBS curves and surfaces can be a good way to create organic models (see Figure 2.6). NURBS models are resolution independent (sort of like the vector graphics in Illustrator and Flash), which means your surfaces are defined mathematically to look good when rendered, no matter how close your camera gets.

All renderers in wide use today, including Maya's native render engine, must convert your geometry to polygons right as they begin rendering. So why don't you see telltale polygonal facets? The trick involved here is that the sphere, for example, is described by the NURBS surface parameters as being perfectly round. When the scene is rendered, Maya can use as many facets as necessary to make the surface appear smooth.

### Polygons

Polygonal geometry is made up of facets - called faces. These three-sided polygonal faces are the result of connecting the dots of three points - called vertices. The lines used to connect each

vertex are called edges (see Figure 2.7). Polygonal geometry can be great for creating hard surfaces like buildings and machines, or if you use enough polygons, organic characters and props.

Unlike NURBS surfaces, polygonal geometry doesn't easily adapt its resolution to remain smooth looking in close-ups. Polygonal objects are made of a finite number of facets regardless of proximity to the camera. But polygons can be quicker to render as a result. And polygonal objects can support branching topography (such as fingers or spokes), with which NURBS surfaces have a hard time.

### Subdivision Surfaces

Subdivision surface objects act as a sort of hybrid between the other two surface types (see Figure 2.8). They are resolution independent, like NURBS, and yet they allow arbitrary topology, like polygons. They are the bee's knees. With sub-d objects it is possible to add localized detail where you need it, punch holes through the surface, and define hard creases and smooth curves in the same model.

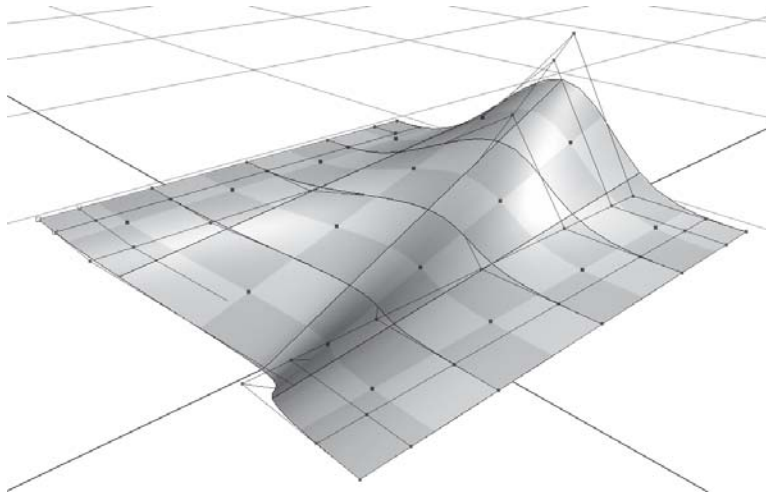


Figure 2.6: NURBS surface

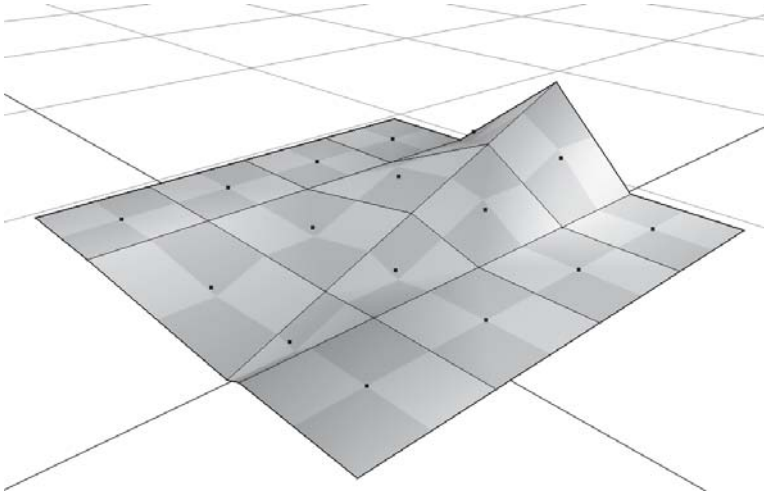


Figure 2.7: Polygon surface

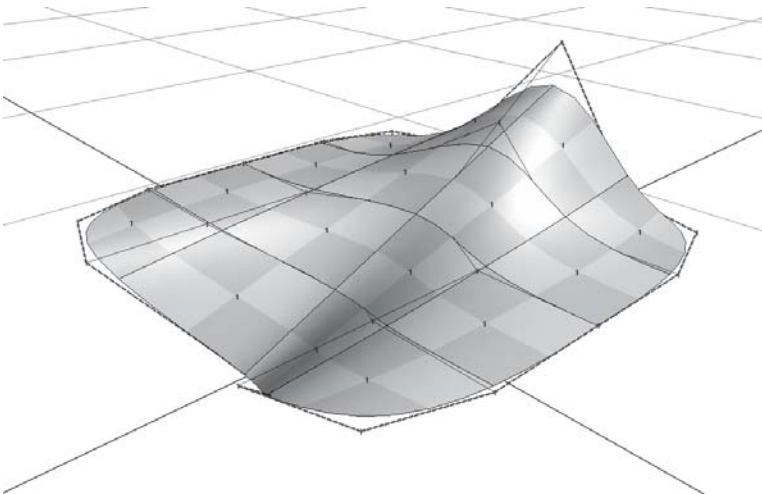


Figure 2.8: Subdivision surface

## Component Mode

Components are the parts that lie just below the object level of a model. At the component level you'll find the points, lines, and surfaces that make up the model (see Figure 2.9).

Points can be used to create a new shape, sort of like pushing and pulling clay. A NURBS point is called a **control vertex**, or **CV**. A polygonal and subdivision surface point is a **vertex**.

Lines connect the points. A NURBS line that lies directly on the surface is called an **isoparm**. (NURBS points are connected with **hulls**.) A polygonal and subdivision surface line is called an **edge**.

When the points and lines are doing their job, you'll end up with a surface. A NURBS surface is called a **patch**. The polygonal and subdivision surface equivalent is called a **face**.

Use the **Status Bar** component icons to pick these different components. Alternately, press **F8** on your keyboard to switch into component mode.

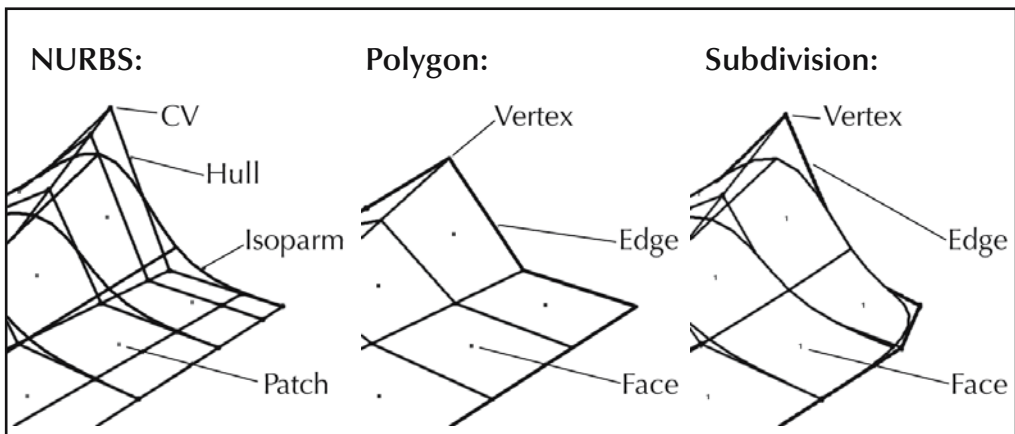


Figure 2.9: Geometry components

## Modeling Exercise – NURBS Components

1. Set your project to `understanding_maya\Chapter02\modeling_icecream`.
2. Open the scene `modeling_icecream_begin.ma`.
3. You will see the familiar room and table, but this time there is a small stand on the table (see Figure 2.10). You will be modeling an ice cream cone, some ice cream, and a cherry.
4. First, click on **Create > Polygon Primitives > Cone** to create the cone. Rename this object *waffleCone*.
5. Dolly out so you can see the cone. Use the **Move Tool** to place the cone in the middle of the stand, about 0, 12.5, -5 on translate X, Y, and Z.
6. Next, use the **Rotate Tool** to flip the cone around 180 degrees on the Z-axis.
7. Use the **Scale Tool** to elongate the cone, about a factor of 1.75 units on the Y-axis (see Figure 2.11).
8. For the remainder of the exercise, be sure that the **Construction History** button is in the **On** position. You can check this in the Status Line. The **Off** state has a red X going through it.
9. Create a NURBS sphere. Rename it *Scoop*.

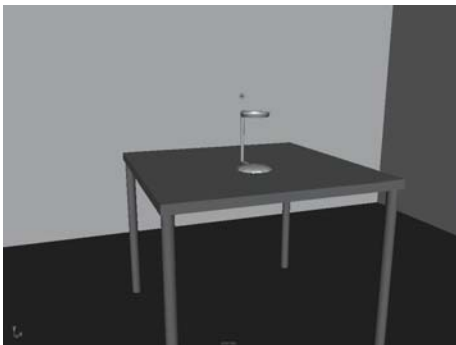


Figure 2.10: Initial scene

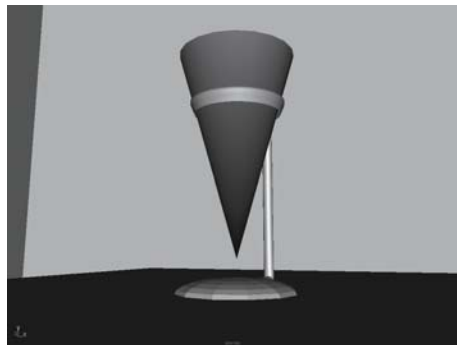


Figure 2.11: Added cone

## 64 2. Refinements

---

10. Translate the *Scoop* to 0, 14.5, -5.

11. In the Channel Box, look for the INPUTS node. Click on the word **makeNurbsSphere1**. This opens up the list of input channels for the *Scoop*.

12. Increase the number of Sections to 10, and the number of Spans to 6. You will see the number of Isoparms increase on the *Scoop*. This extra geometry will be needed in order to reshape the ice cream – the more detail you want in a surface, the more geometry it'll require.

13. The standard mode of selection in Maya is called **Object Mode**. All of your picking occurs on the object level, such as lights, surfaces, and cameras. When you want to select components, such as the Control Vertices (CVs), edges, and faces of an object, you must choose **Component Mode**. To switch to component mode, click on the Select By Component Type button in the status line.

14. Make sure that the Points component type is selected. This is the square point icon just after the Select By Component Type button.

15. The CVs for the *Scoop* are now displayed and can be selected and translated. Try picking and then using the **Move Tool** on CVs or whole rows of CVs. Reposition them to form a more natural-looking ice cream scoop (see Figure 2.12).

16. One way to get even-looking shape transformations is to marquee select entire rows of CVs and then use the **Scale Tool** on them. This has the effect of pushing all of the selected CVs away from each other, or pulling them in toward the selection's center.

17. The **Rotate Tool** can also be used to make a broad change to a selection of CVs.

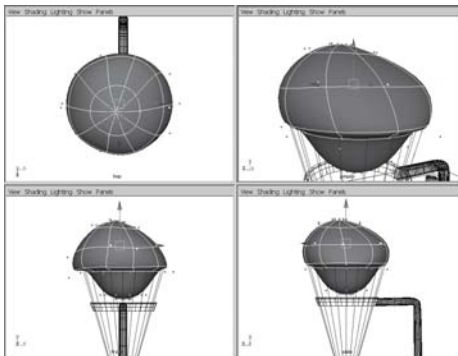


Figure 2.12: The reshaped sphere scoop

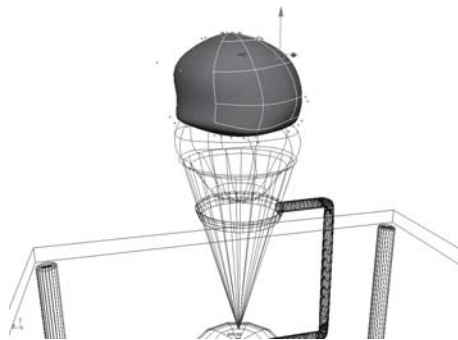


Figure 2.13: The second scoop

18. Be sure to check the top, front, side, and perspective camera views when sculpting a surface.
19. When you are satisfied with the shape of the *Scoop*, return to object mode. You can click the Select By Object Type button on the status line, or press **F8** to toggle between object and component pick masks.
20. Repeat steps 8 through 18 in order to make a second scoop named *Scoop02*. This scoop should be translated higher on the Y-axis so that it sits on top of the first one (see Figure 2.13).
21. Using the same techniques as the ice cream scoops, model a cherry for the top of the scoops. We will do the stem separately, so just create the cherry itself for now (see Figure 2.14). Name it *Cherry*.
22. For the stem, create a NURBS cylinder, and name it *Stem*.
23. Translate and scale the *Stem* into position above the *Cherry*. (Try scale 0.035, 0.5, 0.035 if you want a general guideline.)
24. In the INPUTS node of the Channel Box for the Stem, click on the word **makeNurbCylinder1**. Increase the number of Spans to eight.
25. With the *Stem* selected, toggle to Component Mode (**F8**) and sculpt the stem using the **Move**, **Rotate**, and **Scale Tools** on the CVs (see Figure 2.15).

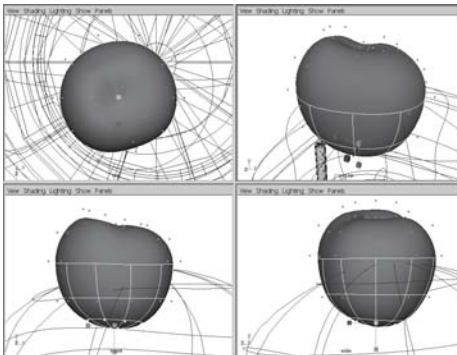


Figure 2.14: Modeling the cherry

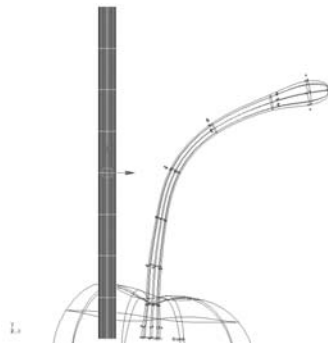


Figure 2.15: From cylinder to stem



## 66 2. Refinements

---

26. Now, let's create some materials to apply to the cone, scoops, cherry, and stem. Click on the Quick View button for Hypershade/Persp. In the Hypershade menu, click **Create > Materials > Phong**.

27. Since you'll be editing the attributes of a number of materials, it's helpful to "dock" the **Attribute Editor** into the interface. A floating window can get in the way if you are using it a lot. To dock the Attribute Editor, click the **Show or hide the Attribute Editor** button in the status line. This takes the place of the Channel Box/Layer Editor.

28. Change the name of the newly created Phong material to *phongCone*.

29. Change the **Color** attribute to a yellow/orange color (HSV 47, 0.9, 0.9).

30. Decrease the **Cosine Power** to 7 – this has the effect of broadening the specular highlights of the material, making it look less like plastic.

31. Drag the **Specular Color** slider to the left, to about the one-quarter position. This decreases the brightness of specular highlights, dulling the material.

32. **MMB**-drag the *phongCone* material onto the *waffleCone* object.

33. Create another Phong material. Name it *phongVanilla*.

34. Change the **Color** to slightly off-white.

35. Increase the **Ambient Color** slider slightly.

36. Set the **Cosine Power** to 16.

37. Lower the **Specular Color** to about the one-quarter position.

38. **MMB**-drag the *phongVanilla* material onto the *Scoop* object.

39. In the Hypershade, select the *phongVanilla* material, click **Edit > Duplicate > Shading Network**. This makes a copy called *phongVanilla1*. Rename it *phongChocolate*.

40. Change the *phongChocolate* color to brown (HSV 27, 0.65, 0.4).

41. **MMB**-drag *phongChocolate* onto the *Scoop02* object.

42. For the cherry, create a new Phong material, and name it *phongCherry*.

43. Change the color to deep red (HSV 3, 0.9, 0.8)
44. For a tight highlight, increase the **Cosine Power** to 80.
45. Change the **Specular Color** to a light grey, about three-quarters to the right on the slider.
46. **MMB**-drag the *phongCherry* material onto the *Cherry* and *Stem* objects (see Figure 2.16).
47. Save the scene as **modeling\_icecream\_end.ma**.

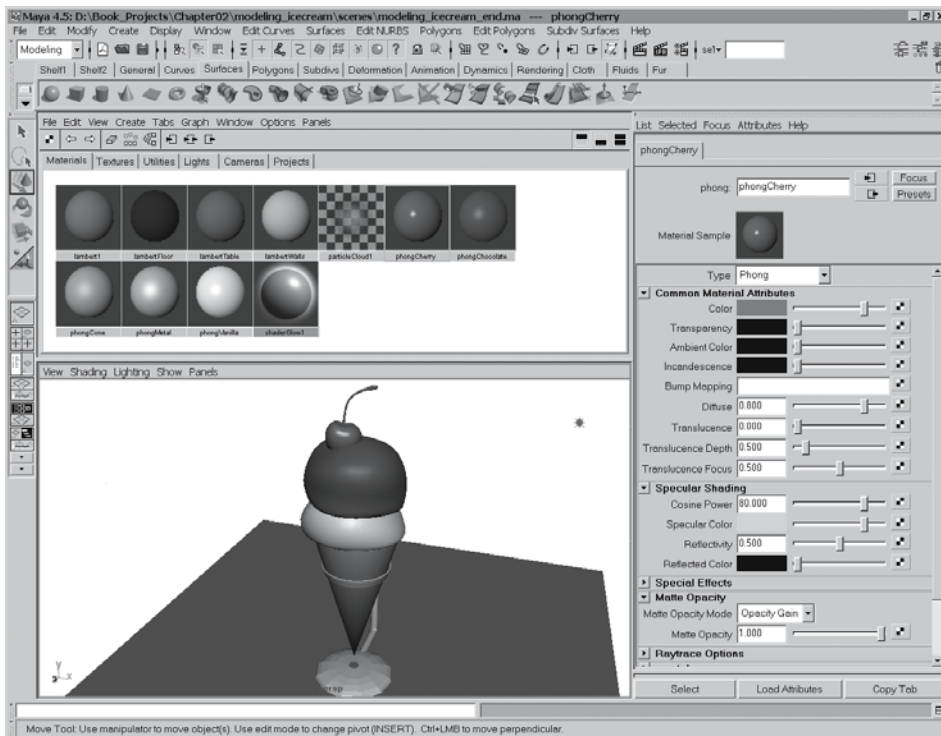


Figure 2.16: The finished model

## 2.3 Animation – Copying Keyframes and Moving Pivots



There are many tools in Maya that can be used to automate or speed up otherwise tedious tasks. One such task is the creation of regular repeating movements in an animation. For example, to make a yo-yo go up and down on a string you would:

- Place a keyframe for the yo-yo's Translate Y and Rotate Z channels at frame 1;
- Then go to frame 20, move the yo-yo down, rotate it; and place keyframes for these new values; and finally,
- Go to frame 40, move it up, reverse the rotation, and set keyframes.

### Copying Keyframes

To throw the yo-yo again, it isn't always necessary to repeat this procedure. Instead, you could copy the Rotate Z and Translate Y keyframes at frames 1, 20, and 40, and then paste duplicate sets of keyframes at any points along the timeline when the yo-yo is thrown. The **Time Slider** provides a convenient way to do this. With an object selected, the **Time Slider** displays a red tick mark for each keyframe. By <shift> + **LMB**-dragging across a range of keyframes, you can select multiple keyframes. By **RMB**-clicking on the highlighted frames, you can **Copy** the entire range of keyframes. Then you scrub to a new point on the timeline, **RMB**-click, and choose **Paste > Paste**, to make a second iteration of the keyframes. We'll try it out in the next exercise.

### Animation Exercise – Copying Keyframes

1. Set your project to `understanding_maya\Chapter02\animation_copykeys_yoyo`.
2. Open the scene `animation_copykeys_yoyo_mid.ma`.
3. Press the **Play** button. The yo-yo goes down and then up one time (see Figure 2.17).
4. Stop the playback and rewind.

Now, we'll copy the existing keyframes on the yo-yo. The **Time Slider** is much like the **Chan-**

**nel Box** and other elements of the Maya interface, in that it displays information specific to the currently selected object. Notice how when nothing's selected, the **Channel Box** is blank. Similarly, the **Time Slider** displays no red tick marks to indicate the presence of keyframes. This will change once you select an object that has been keyed.

- a. Select the *yo-yo* object.
- b. In the **Time Slider**, hover the cursor over frame 1.
- c. Press and hold the **Shift** key on your keyboard, while simultaneously click-dragging from frame 1 to frame 41. You will see the highlighted section of the timeline turn red (see Figure 2.18).

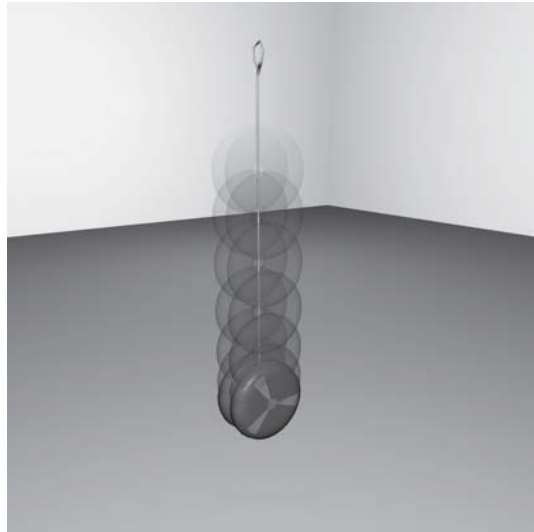


Figure 2.17: Yo-yo in action

- d. RMB-click anywhere inside the selected range of keyframes, and click on **Copy** in



Figure 2.18: Selecting a range of keys

the context menu (see Figure 2.19). This copies all of the selected keyframes.

5. The Translate Y and Rotate Z keyframe values, interpolations, and relative spacing are now stored in a copy buffer. Just as in a word processing or email program, you can paste that information into a new place. Scrub to frame 50 on the **Time Slider**. This will be the insertion point for your new set of keyframes. **RMB-click** on frame 50, and click on **Paste > Paste** in the context menu (see Figure 2.20). Notice that there are three new keyframe ticks on the timeline. (Since the Translate Y and Rotate Z channels were originally keyed together, Maya displays one tick mark at each frame. You will use the **Graph Editor** when you need to edit these channels separately.)
6. Rewind and then play your animation. The yo-yo now cycles twice in quick succes-

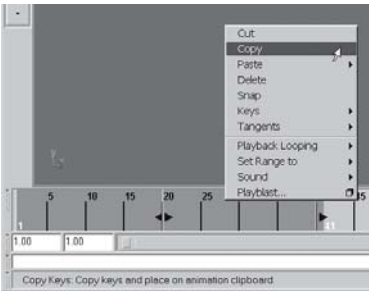


Figure 2.19: Copying keys

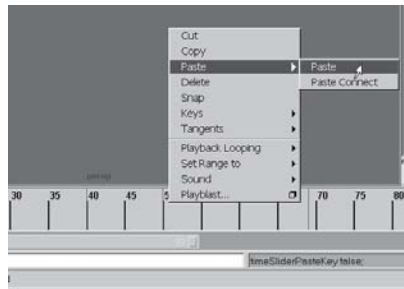


Figure 2.20: Pasting keys

sion.

7. Since the copied keyframes are still in the copy buffer, you can paste another iteration of the motion without re-copying. Go to frame 96 and paste the keyframes again.

8. Play back the animation. Let's say that you've changed your mind and would like the last throw to occur much later. Instead of deleting, or undoing and then re-pasting, we'll use another method. In much the same way as you highlighted a range of keyframes for copying, you can re-time an animation by sliding that range of keyframes on the **Time Slider**.

9. **Shift-LMB**-click the keyframes from frame 96 to frame 136. Notice that the red highlighted section has four arrows in it. Click-drag the middle set of arrows (they are positioned around frame 113) to the right. As you drag, pay attention to the white frame numbers at the head and tail of your selected range. When the head number reaches frame 130, stop sliding (see Figure 2.21).

10. Play your animation. The third throw has been time-shifted.



Figure 2.21: Sliding keys

11. What if you wanted to not only change *when* the third throw happened, but *how quickly* it happened? The relative spacing between the keyframes that comprise one throw is fine, but the absolute length of a throw needs to be changed. This is the role of the arrows at the ends of the red highlighted range. Click-drag the head arrow to the right, stopping at frame 146 (see Figure 2.22). Notice how the entire range has been scaled, meaning that the

relative spacing is intact – the middle tick mark is still equidistant from the head and tail.

12. Now, when you play your animation you'll see that the third throw is much quicker than the first two.



Figure 2.22: Scaling keys

13. Save your scene as **animation\_copykeys\_yoyo\_end.ma**.

## Pivot Points

Everything that rotates does so around a pivot point. You can't get around this simple fact. Look at any object that rotates: the hands on a clock, a knob on a radio, the ball in a mouse, the wheels on a chair, your elbow, the earth, and so on. These things all have a single point around which they rotate. Roll a ball and the pivot point is at its center. Swing a stopwatch dangling from a chain and the pivot point is right at your fingertips.

In Maya, you must learn how to control pivot points in order to control your animation. The pivot point of an object is where you will see the **Move**, **Rotate**, and **Scale Tool** manipulators appear. Pivot points usually are located in the same place as hinges and axles (see Figure 2.23).

Try this simple exercise.

1. Create a polygon cube.

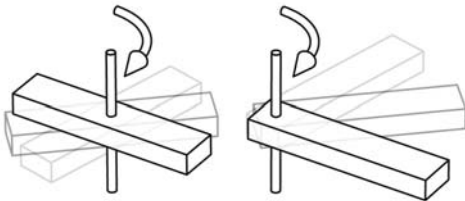


Figure 2.23: Pivot points

## 72 2. Refinements

---

2. Scale the cube to be shaped like a door from the front view panel. ScaleX = 3.5; ScaleY = 6; ScaleZ = 0.3. Move the cube up 3 units on the TranslateY axis. Name the cube *leftDoor*.
3. With *leftDoor* still selected, click **Edit > Duplicate options**. Reset the tool settings, and then change the **Translate** settings to 4, 0, 0. This will make a duplicate of the *leftDoor* that is 4 units to the right. Click the **Duplicate** button.
4. Rename the new object *rightDoor*.
5. With *rightDoor* selected, choose the **Move Tool**. Notice that the manipulator is located at the center of the object. Now, press the **Insert** key on your keyboard. This puts the manipulator into edit mode, allowing you to move the pivot point itself.
6. Click-drag the pivot point to the left edge of the *rightDoor* object, like a single hinge (see Figure 2.24). Now, press the **Insert** key again to return to translate mode.
7. Switch to the **Rotate Tool**. Pick and rotate the *leftDoor* object on its RotateY axis. Notice how it swings around its center.
8. Pick and rotate the *rightDoor* object on its RotateY axis. This one swings more like a normal door, thanks to the relocation of its pivot point (see Figure 2.25).
9. One last thing to point out: the pivot point affects scaling as well as rotation. Try scaling each of the doors on the ScaleX axis. The *leftDoor* scales from the center outward, while the *rightDoor* scales away from its hinge (see Figure 2.26).

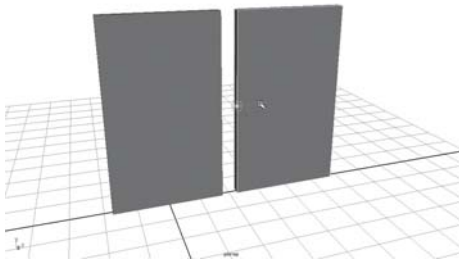


Figure 2.24: Moving the pivot point

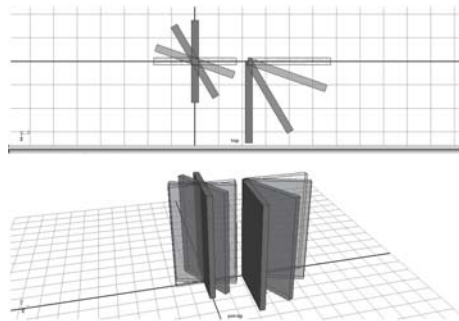


Figure 2.25: Rotating around the pivot point

You'll need to pay special attention to pivot points when creating multi-part assemblies. A multi-part assembly could be a mechanical device such as a car, or a skeleton for animating a creature. Each part must have a carefully considered pivot point.

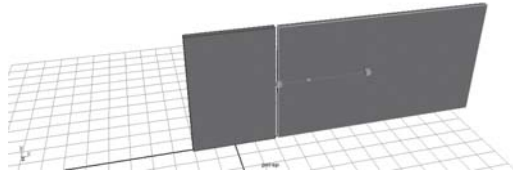


Figure 2.26: Scaling from the pivot point

### Animation Exercise – Pivot Points

1. Set your project to `understanding_maya\Chapter02\animation_pivot_wagon`.
2. Open the scene `animation_wagon_pivots_begin.ma`.
3. You'll see a little red wagon, made up of a number of parts, such as the body, wheels, handle, yoke, and axle. Try selecting and rotating each part. You will find that the pivot points are not placed correctly, therefore the rotations are incorrect (see Figure 2.27). Be sure to undo any rotations when you are done.
4. Use the **Insert** key to edit the pivot points of each object. Start with the *handle*. It needs to be able to rotate up and down on its Z-axis. By using the front view panel, align the pivot point of the *handle* object to the very end where it intersects the *yoke*. Hit **Insert** when you are done. Switch to the **Rotate Tool** and try out the handle. It now works much better (see Figure 2.28).

5. The *yoke* object currently rotates around its pivot point at the world origin. This breaks the entire steering assembly. Instead, the pivot should be moved to the high point of the *yoke*, where it is nearly touching the underside of the wagon body. Hit **Insert** and reposition it there. When done, press **Insert** again to return to the **Rotate Tool** and try rotating on the Y-axis. The *yoke* now pivots properly (see Figure 2.29). You may have noticed that the different parts of the wagon aren't yet interacting. For example, the *yoke*



Figure 2.27: Bad pivot points



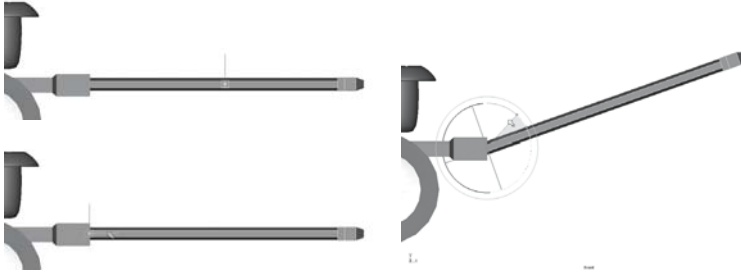


Figure 2.28: Correcting the handle pivot point

rotation has no effect on the position of the front axle, wheels, or handle. You will be fixing this problem in a later exercise dealing with hierarchies in Chapter 4.

The wheel rotations need to be fixed with a bit more precision than did the handle and yoke. It would be very apparent if the wheels didn't rotate around their exact center – they'd wobble. To place the pivot point correctly, we'll use the **Snap to points** option. This will allow us to swap the pivot point to the vertex at the wheel's hub.

6. Select the *wheel\_FrontRight* object. Press **Insert** to switch to pivot point editing mode. Next, click the **Snap to points** button on the **Status Line**, or hold the **V** key on the keyboard, to invoke point snapping. Hover the cursor over the vertex at the red hub center. Now, **MMB**-click the vertex. This will make the pivot point jump to that position (see Figure 2.30).
7. Press **Insert** to return to the **Rotate Tool**. The wheel now spins properly.
8. Repeat this for each wheel. You may also use the **LMB**-click-drag along with the **Snap**

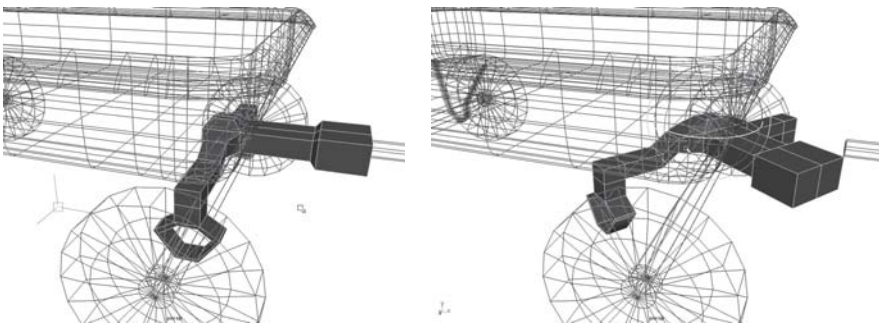


Figure 2.29: Correcting the yoke pivot point

to **points** option to drag the pivot point to the hub. Just be careful to drag from the **Move Tool's** center, not on a single axis.

9. All that's left now are the axles. Unfortunately, they lack a good central vertex for point snapping. Instead, we will use Maya's automatic pivot centering command. This works because the geometric center of the object is in the same location as our desired rotational pivot.

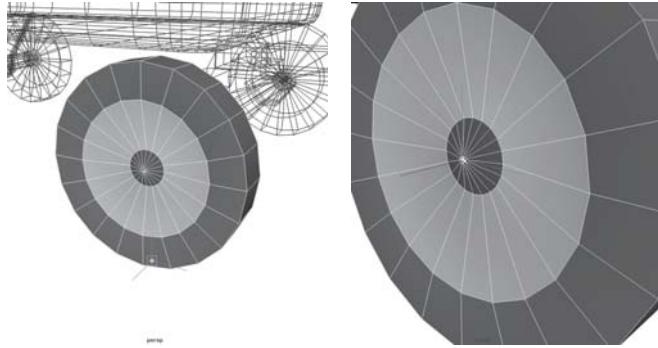


Figure 2.30: Vertex snapping the wheel's pivot point

10. Select the *axle\_front* object. Click **Modify > Center Pivot** (see Figure 2.31). Now, use the **Rotate Tool** and spin the *axle\_front* object around its Z-axis to test it.

11. Repeat this for the *axle\_rear* object. Now, when you rotate any of the movable parts of the wagon, there are no unnatural pivot points.

12. Save your scene as **animation\_wagon\_pivots.ma**.

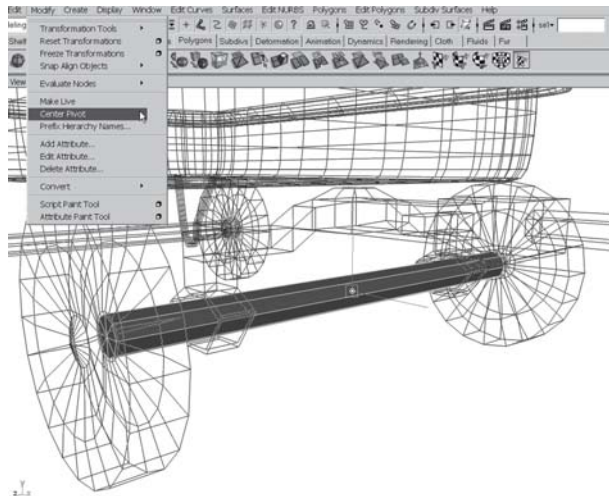


Figure 2.31: Centering the axle pivot point

## 2.4 Shading – Material Types



On the most basic level, a material defines how a surface looks when rendered. While all material types share common attributes, such as **Color**, **Ambient Color**, and **Diffuse**, the way in which specularity is defined varies among the different material types. In fact, the most intrinsic value of a material is its **Type**. The major material types differ fundamentally in the way they handle specularity. Specularity helps define shininess, glossiness, or highlighting. Look at the different materials in Color Plates XI and XII.

The mathematical algorithms used to process this specularity give the material types their names. (They are generally named after the computer graphics researcher who developed the algorithm.) Here is a guide to Maya's major material types (see Figure 2.32), and how you can control their specular settings.

**Lambert:** This material type has a dull or matte finish. It has no specular attributes.

**Phong:** A shiny, plastic-like quality defines this material type. **Cosine Power** values broaden or tighten the size of highlights.

**Blinn:** This is often used for metallic surfaces. The **Eccentricity** value varies the size of highlights.

**PhongE:** Similar to **Phong**, but with an increased ability to create soft highlights. **Highlight Size** controls, you guessed it, highlight size! **Roughness** allows you to vary the highlight's fade from center outward.

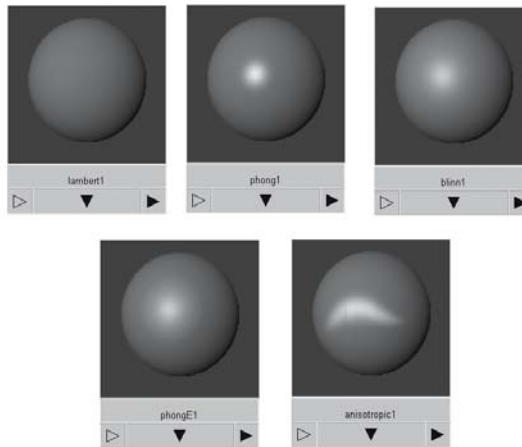


Figure 2.32: Material types

**Anisotropic:** Special-use material type for directional highlights. This material type allows you to specify a directional highlight, such as brushed metal, hair, tightly woven fabric, vinyl records, etc. **Angle** controls the direction of the highlight (caused by micro-grooves in the surface you are simulating). **Spread X** and **Spread Y** change the size of the highlight. **Roughness** allows you to fade the highlight, much as in the **PhongE** material.

## Shading Exercise – Material Types

1. Set your project to `understanding_maya\Chapter02\shading_material_types_donuts`.
2. Open the scene `shading_material_types_donuts_begin.ma`.
3. You are going to create a new material for each doughnut. When working on the shading of a scene it is helpful to configure the interface with a **perspective** view panel, the **Hypershade**, and an **Attribute Editor**. One way to set this up is by selecting the **Quick View** button under the **Tool Box** called **Hypershade/Persp**. Then, click the **Show** or **hide the Attribute Editor** button in the **Status Line**. This is also called “docking” the **Attribute Editor**, since it is no longer a floating window (see Figures 2.33 and 2.34).



Figure 2.33: Docking button

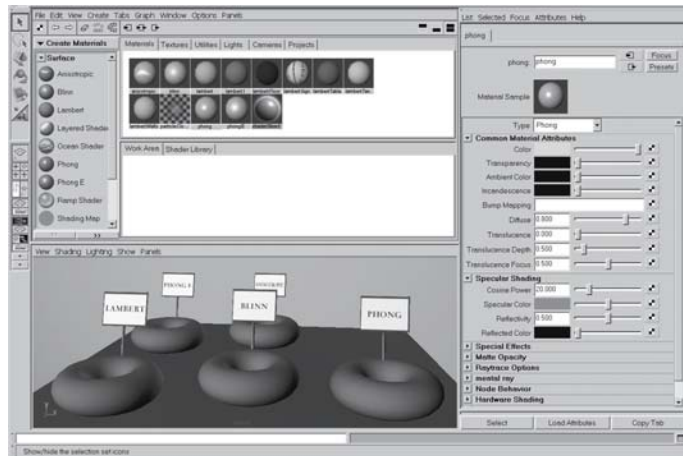


Figure 2.34: Shading layout

4. Now create five new materials by clicking in the **Hypershade's** menu on **Create > Materials > Lambert**, followed by **Blinn**, **Phong**, **PhongE**, and **Anisotropic**.
5. Assign each material to the appropriately labeled doughnut. (Remember that you can **MMB**-drag materials onto objects.)
6. Most material attributes cannot be accurately judged through interactive shaded mode. To see the results of your shading efforts, you'll need to render the current frame. First, click on the **Render Globals** button on the **Status Line**. Choose the **Resolution** preset of **640x480**, and set the **Anti-aliasing Quality** preset to **Intermediate** (see Figure 2.35). Close the **Render Globals** window, and then click the **Render the current frame** but-

## 78 2. Refinements

ton. Take a look at the different specularly types the materials provide (see Figure 2.36).

7. You can create wildly different results by varying the specularly settings of the different materials. Try these settings and then re-render the frame to compare.

a. Blinn: Eccentricity: 0.6, Specular Roll Off: 0.5.

b. Phong: Cosine Power: 60, Specular Color: White.

c. PhongE: Roughness: 0.8, Highlight Size: 0.4.

d. Anisotropic: SpreadX: 10, SpreadY: 1.75, Roughness: 0.85, Specular Color: White.

8. Save the scene as `shading_material_types_donuts_end.ma`.

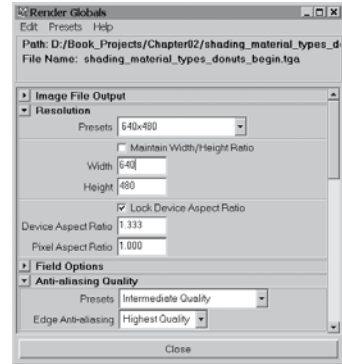


Figure 2.35: Render globals

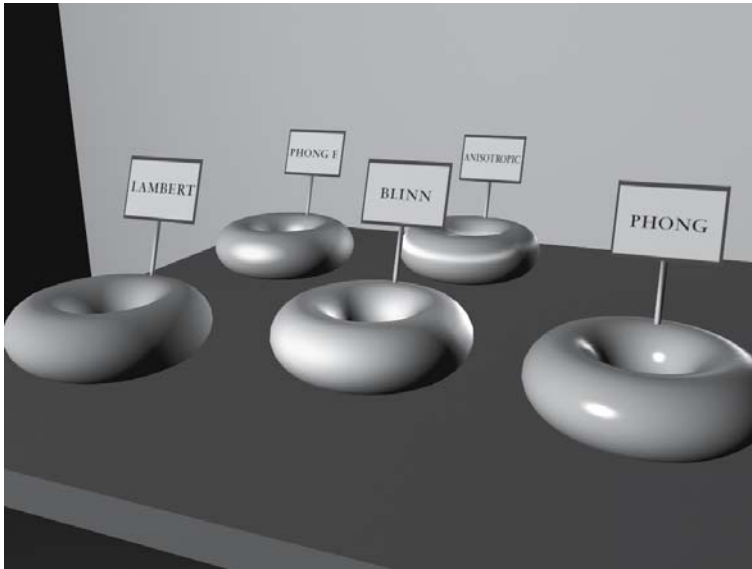


Figure 2.36: Rendered material types

## 2.5 Lighting – Spotlights



Spotlights are more complex than the simple point light we looked at in the first chapter. You can see some nice examples of spotlights in Color Plates I and III. The spotlight radiates light from a single point toward a target in a cone-like fashion, much as does a flashlight (see Figure 2.37). The controls are also more complex than those found on a point light, because Maya allows you to specify things like the angle of the cone of light and the way in which light dissipates. Let's look at a few of the parameters of the spotlight.

The radius of the cone is defined by the **Cone Angle** setting.

The soft dissipation at the edge of the light is controlled with the **Penumbra angle**.

The falloff of the inner "hotspot" is controlled with the **Dropoff** attribute.

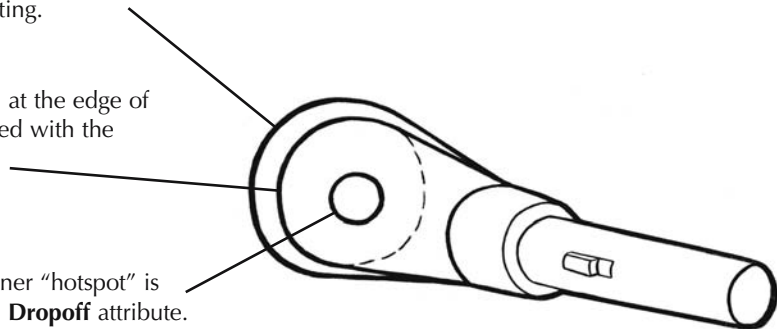


Figure 2.37: Spotlight attributes

The spotlight node is the where the body of the light is. The target node is the point at which the spotlight is aimed. By using the Show Manipulator Tool, or the **t** hotkey, you can access this extra node. Wherever the target is, the spotlight points (see Figure 2.38).

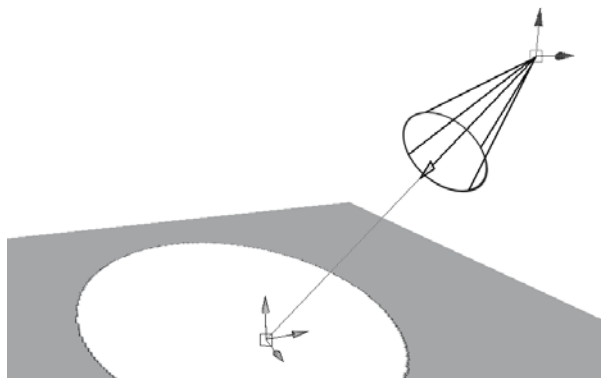


Figure 2.38: Spotlight and aim

## Lighting Exercise – Spotlight

1. Set your project to `understanding_maya\Chapter02\lighting_spotlight`.
2. Open the scene `lighting_spotlight_icecream_begin.ma`.
3. Yep, it's the familiar room with an ice cream cone on the table. Currently there is a general illumination in the scene. One way to check the lighting is to select shaded mode and hardware lighting mode for the **perspective** panel. With the **perspective** panel active, press **5** and then **7** on your keyboard.
4. Add a spotlight to the room by clicking **Create > Lights > Spotlight**. You should see the back wall light up where the light is aimed.
5. Switch to the **Move Tool**, and place the light 12 units up on the Y-axis and 20 units on the Z-axis. You'll notice the area of illumination on the wall changes as you move the light.
6. The back wall has a moderate number of subdivisions in it. This is necessary if you want to see the effects of your lighting in the shaded view panels. However, there are limitations to what the hardware lighting mode will display. For a better idea of what your lights look like, go ahead and render a frame. Click on the **Render the current frame** button in the **Status Line**.
7. You'll notice that your spotlight casts a bright circle of light and currently casts no shadows (see Figure 2.39). Since shadow casting in Maya is a function of your lights, we'll use the **Attribute Editor** to turn on the *spotLight1* shadows.
8. With *spotLight1* selected, click **Ctrl-a** to open the **Attribute Editor**. Open the **Shadows** turn-down menu, and then check the box next to **Use Depth Map Shadows** (see Figure 2.40). Re-render the frame (see Figure 2.41).
9. When the light moved away from the wall, its circle became bigger. Another way to control how much area is covered by the light is the **Cone Angle**. This attribute alters the shape of the spotlight cone. A wide angle covers a greater area, while a narrow angle



Figure 2.39: Default spotlight

focuses the beam more tightly. In the **Attribute Editor** change the *spotLightShape1*'s **Cone Angle** to  $20^\circ$ , and re-render the frame (see Figure 2.42).

10. Try widening the **Cone Angle** to  $80^\circ$  and render the frame. This now encompasses the side walls, too. This is wider than we need, so reduce the **Cone Angle** down to  $30^\circ$ .

11. You may have noticed that the circle of light is uniformly bright from its center outward. Most light sources diminish as they radiate out from their centers. This is described as the **Dropoff** of a light. The default dropoff value for the Maya spotlight is 0; in the **Attribute Editor** increase the **Dropoff** to  $40^\circ$ , then render the frame (see Figure 2.43).

12. This looks better, but the edge of the light is still too hard. The **Penumbra Angle** controls how hard or soft the edge of the light is. Increase the **Penumbra Angle** to  $4^\circ$ . Now, when you re-render the frame you'll have a much more pleasing, realistic light (see Figure 2.44).

13. If you need to aim the spotlight precisely, it can be difficult to simply rotate it with the **Rotate Tool**. Try this: move the time slider to frame 30 – the ice cream cone has leapt into the air – and now aim the spotlight at the cone using the **Rotate Tool**. Translate the spotlight to the left, about -15 units on the X-axis. The aim is off, and the cone has been

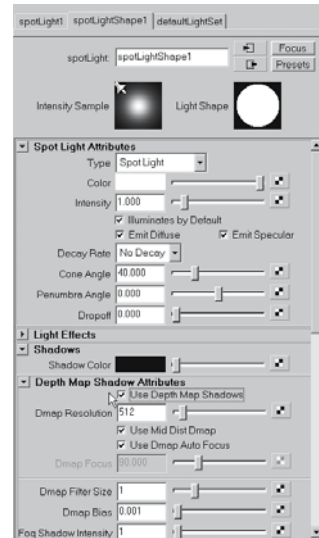


Figure 2.40: Shadow attributes



Figure 2.41: Shadow render

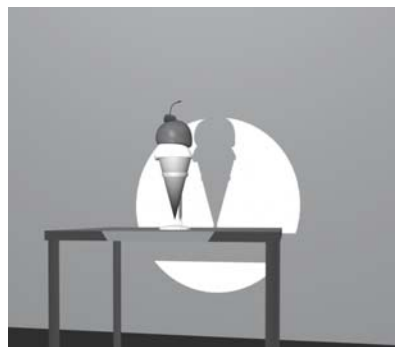


Figure 2.42: Cone angle of  $20^\circ$





Figure 2.43: Dropoff angle of 40°



Figure 2.44: Penumbra angle of 4°

plunged into darkness! You must now rotate the light again to re-aim it. There is a simpler way: the **Manipulator Tool** (see Figure 2.45).

14. With the *spotLight1* still selected, choose the **Show Manipulator Tool** icon from the **Tool box**. Two manipulators appear: one that controls the body of the light, and one that controls the aim of the light. If you move the body, its rotation will automatically change to remain aimed at its target manipulator. By moving the aim manipulator, you are controlling the spotlight's rotation in an intuitive manner.



Figure 2.45: Target manipulator

15. Now, try to animate the light so that it follows the ice cream cone's acrobatic antics. Notice that only the *spotLight1*'s transform node **Translation** and **Rotation** channels are changing when you move the manipulators. So, to keyframe the follow spot, go to frame 15, place the aim manipulator right on the ice cream cone, and set a keyframe for all three rotation channels.

16. Advance to frame 30, move the spotlight's aim manipulator up to the ice cream cone, and set a keyframe on the rotation axes.

17. Repeat this process for frame 40, to illuminate the landing. You can also try to aim the light at the cone during the remaining two leaps it makes. Save the scene when you're done.

## 2.6 Rendering – Camera Basics



The last time we rendered a series of frames we did so from the **perspective** panel view. The view you get through the **perspective** panel usually looks more interesting than the **Front** or **Side** views because of its non-orthographic view and its wide-angled lens. When you want to create even more interesting shots, in which there are pans, dollies, cranes, and even simulated Steadicam-style movements and rack-zooms, you'll need to animate the camera. Since the **perspective** view isn't really designed to be animated, you'll create a new camera. In Maya, cameras have settings for nearly every attribute of a real-world film camera, and even some beyond that. Additionally, they can be animated to move and rotate through your scene.

First, let's create a new camera. Click on **Create > Cameras > Camera and Aim**. This makes a two-node camera (see Figure 2.46). In the **Outliner** you will see a new item called the *camera1\_group*, which contains two nodes, the *camera1* and the *camera1\_aim*. This is a frequently used camera type in Maya. (The single-node camera can be difficult to aim, and the three-node camera can lead to motion sickness!)

To properly line up your shot, you can look through the lens of a new **Camera**. The view panel lists the name of the current camera at the bottom of the screen. In order to change the view of a panel, click **Panels > Perspective > Camera1** (see Figures 2.47 and 2.48).

Try to get used to animating the position of the Camera node and the Camera Aim node (this is similar to the target of your spotlight). The Camera always points at its Aim node, so you can leave the Camera alone and animate the Aim node to simulate a tripod-mounted camera panning a scene. In order to simulate a moving camera that holds its aim on a specific subject, place the Aim node near the subject, and then animate only the translation of the camera body node.

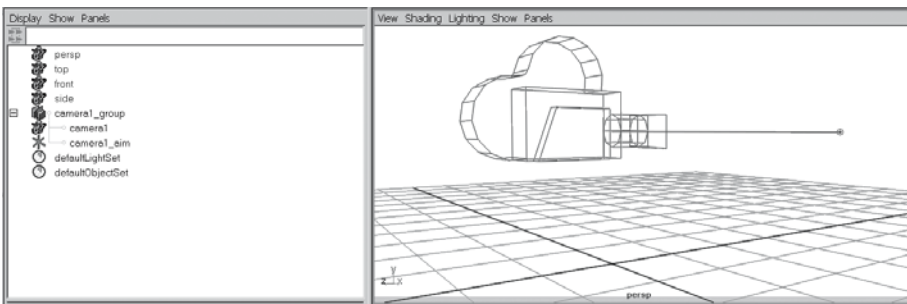


Figure 2.46: The two-node camera

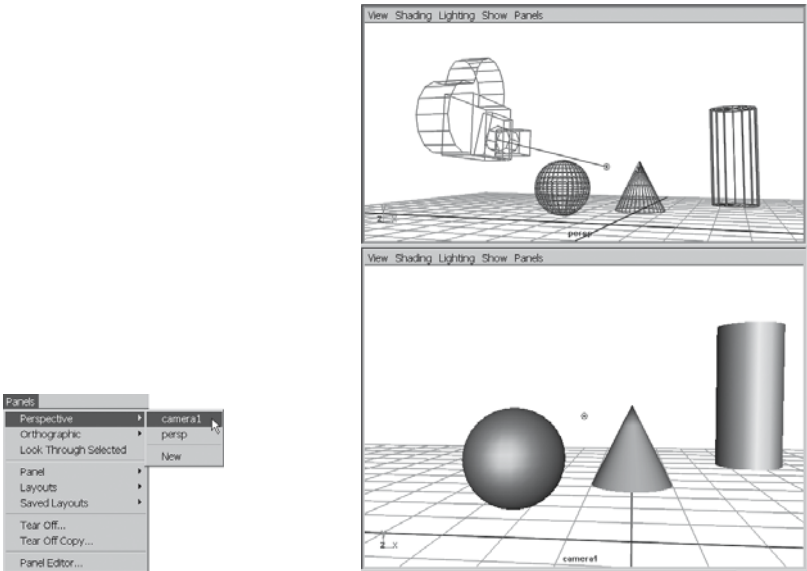


Figure 2.47: Changing views

Figure 2.48: Viewing through the camera

One important setting to get familiar with in the Camera **Attribute Editor** is the **Focal Length** attribute. This is equivalent to the focal length (measured in millimeters) of a 35mm film camera. By lowering the focal length setting, you'll get a wide-angle shot; raise it for a long telephoto lens.

**Rendering Exercise – Two-node Camera**

1. Set your project to `understanding_maya\Chapter02\rendering_camera`.
2. Open the scene `rendering_camera_barn_begin.ma`.
3. Play the animation to see the plane start up and drive out of the barn. The view is static and uninteresting (see Figure 2.49); we'll create a camera and animate it for a more interesting shot.



Figure 2.49: Initial fixed view

4. Click **Create > Cameras > Camera and aim**. The new camera is built at the world origin, which is right inside the barn.
5. In the standard four-view panel layout (click the **Four View** icon in the **Quick View** area under the **Tool Box**), you'll turn the **side** view into the **camera1** view. Do so by clicking in the **side** view panel menu: **Panels > Perspective > camera1**. Now you are looking through the camera's lens in the bottom right panel (see Figure 2.50). You can judge your camera and view positioning through this panel while you move the camera's nodes in the other panels.

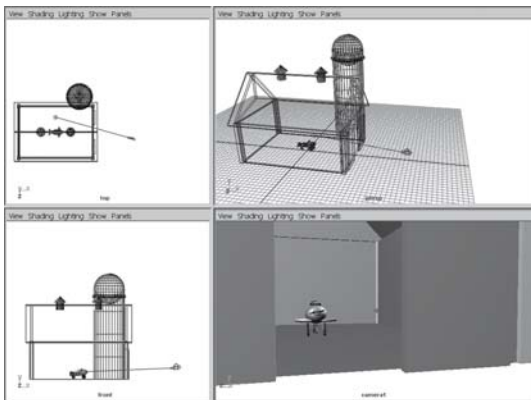


Figure 2.50: Positioning the two-node camera

6. Currently, the *camera1\_group* node is selected. Select the *camera1* node in the **Outliner** by clicking the plus sign next to the *camera1\_group* node, and then highlighting the *camera1* node. Use the **Move Tool** to bring the *camera1* node outside the barn, approximately 25, 3, 2 for translate X, Y, and Z.
7. At frame 1, set a keyframe for the *camera1* node's Translate X, Y, and Z channels. You can do this by highlighting those three channels in the **Channel Box** and then **RMB**-clicking to choose **Key Selected**.
8. Scrub the time slider to frame 48, move the *camera1* node farther away from the barn (to about 45, 3, 12) and keyframe those three channels (see Figure 2.51).
9. Play the animation. The camera is trucking away from the barn nicely, but the view is fixed in place. In order to follow the plane, we'll animate the *camera1\_aim* node.

## 86 2. Refinements

---

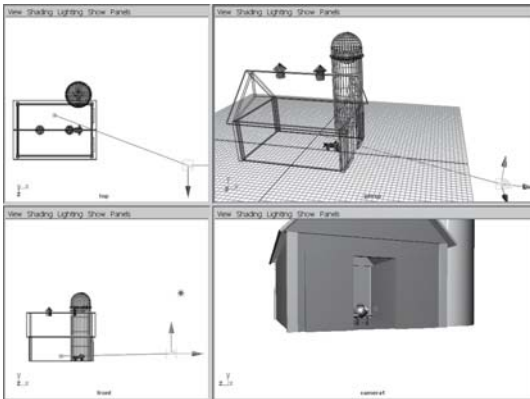


Figure 2.51: Pulling the camera back

10. Stop playback and then go to frame 60. This is when the camera will begin panning.
11. In the **front** panel, select the *camera1\_aim* and move it up about 2 units on the Y-axis. Set a keyframe for the Translate X channel, which is currently at 0.
12. At frame 98, move the *camera1\_aim* node to 50 on the X-axis. Set a key for this channel.
13. Rewind and then play your animation. The camera now follows the plane (see Figure 2.52).
14. Save your scene.

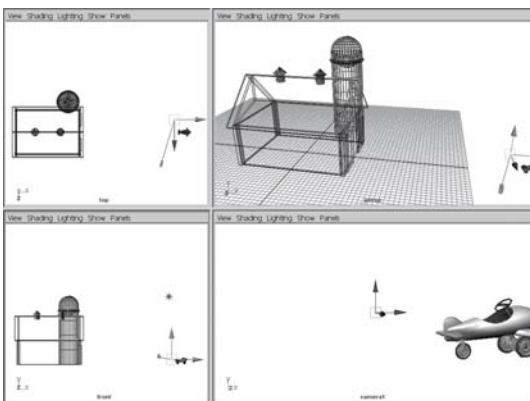


Figure 2.52: Animating the target

## 2.7 Project 2 – Rolling Egg

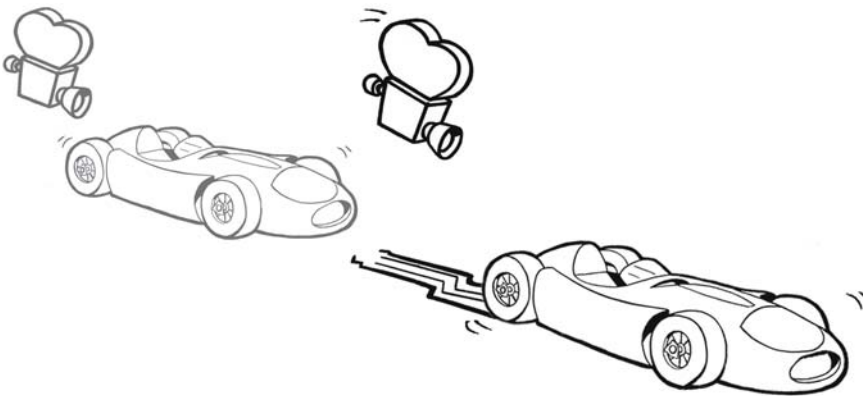
In Chapter 2's assignment you will make the ball-rolling scene a bit more visually interesting by shaping the ball into an egg in component CV mode and by creating a half-pipe from a set of detached cylinder faces.

- Use the copy keyframe command to repeat the egg's rotation.
- Set a spotlight.
- Create an animated camera.
- Select different shader types and specular settings.

---

# Chapter 3

## Intermediate Skills



### 3.1 Review – Attribute Editor

Remember the Alias party line: Maya is nodes with attributes that are connected? Well the Attribute Editor is the main panel you will use to change these attributes on your objects, lights, cameras, and shaders. The **Ctrl-a** keyboard shortcut brings it up quickly, so that you may use it without interrupting your workflow. Its contents are determined by the attributes present in the selected node. Here is a sphere's shape node in the **Attribute Editor** (see Figure 3.1).

Note how some of the attributes are only informational while others are editable. You'll find that switching to the input node that was used to generate the sphere allows you to alter settings such as the sweep angles and radius used to create the object. (OK, it's a cheap trick, but you really should try animating the NURBS sphere's sweep angles for some cool Pac-Man chomping action.)

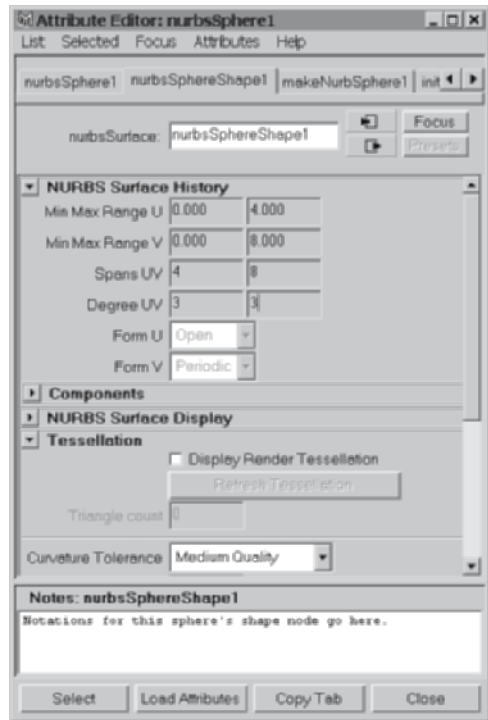
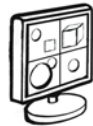


Figure 3.1: Attribute Editor



There is no better way to visualize the connection of nodes to each other than the **Hypergraph**. The **Hypergraph** is really a flowchart of your scene and your objects. It can be arranged to represent the physical layout of nodes and their relationships to each other. (We will cover this more in the Chapter 4 Animation section on hierarchies.)

Take a look at the **Hypergraph** from a simple scene in Figure 3.2. We can see a few of the objects, lights, and cameras here - notice that the selected “egg” object is highlighted both in the **perspective** panel and in the **Hypergraph**. This is called the Scene Hierarchy Mode.



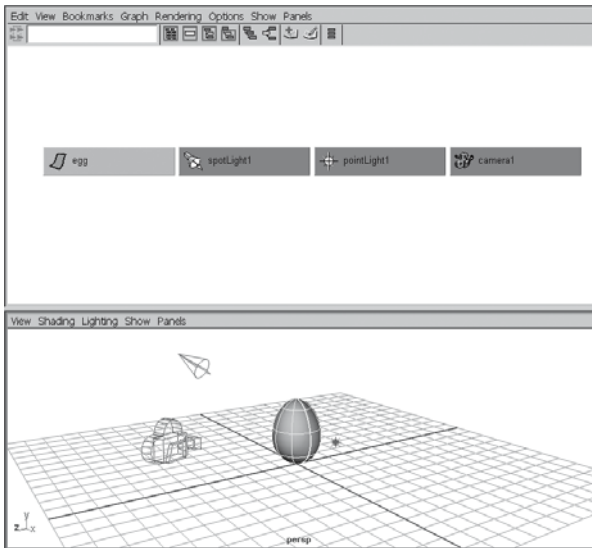


Figure 3.2: Hypergraph scene hierarchy

If we expand the input and output connections of the egg (see Figure 3.3), we can see its Dependency Graph, as in Figure 3.4. The Dependency Graph displays the interconnected nodes that make up the object. In this case, the nodes are the **Input** node: *makeNurbSphere1*, **Shape** node: *eggShape*, **Transform** node: *Egg*, and the **Render** node: *initialShadingGroup*.

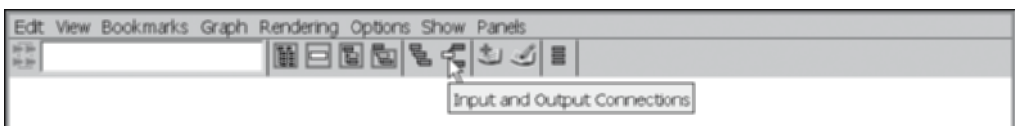


Figure 3.3: Revealing the dependency graph



Figure 3.4: Dependency graph

### 3.3 Modeling – Curves and Surfaces



The NURBS curve is an invaluable tool to use in the creation of surfaces, paths, and control objects. There are three primary ways to create them in Maya: Edit Point (EP) Curve Tool, Control Vertex (CV) Curve Tool, and Pencil Curve Tool.

It is also possible to create new curves by extracting existing curves from other surfaces. For direct curve creation, we will focus on using the CV Curve tool.

#### Drawing Curves

A CV Curve is a curve generated by placing control vertices in your scene (see Figure 3.5). Once you have placed four or more CVs, Maya will display the curve you are creating. The CVs' relationship to their underlying curve is similar to that of a NURBS sphere's CVs and their surface. They act as reference points for the weight and direction of a curve being drawn through the hulls that connect the CVs.

Notice how the pair of CVs that were drawn closest together create the shortest hull, and therefore exhibit the most weight upon the curvature of the underlying span. To make a pinch or tight crease draw your CVs close together. For gentle curves draw the CVs far apart.

If you need to make a closed curve, draw your final CV near, but not on top of the first CV. Hit the enter key to end curve construction, then choose **Edit Curves > Open/Close Curves** from the main menu (see Figures 3.6 - 3.8).

You'll need to be in the **Modeling Menu Set**.

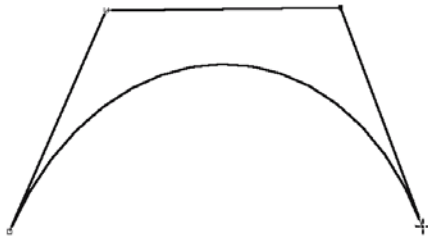


Figure 3.5: CV curve tool

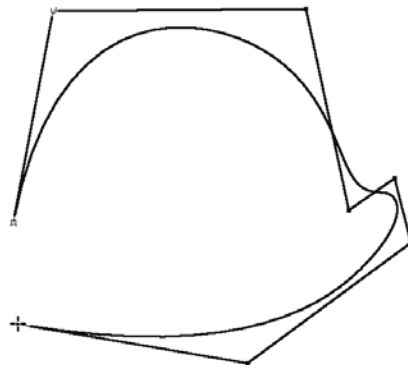


Figure 3.6: Preparing to close the curve



Figure 3.7: Close curve menu

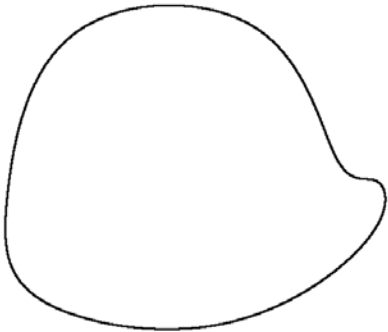


Figure 3.8: Closed curve

### Revolves

Now that you're familiar with drawing curves, let's use one to create a surface. The **Surfaces > Revolve** command acts like a lathe for your curve. By sweeping your curve about one axis of its pivot point, the **Revolve** creates a surface.

The curve in Figure 3.9 was made with the CV Curve Tool.



Figure 3.9: CV curve



Figure 3.10: Revolve menu

Revolving the curve around its Y-axis creates a bowl (see Figures 3.10 - 3.12).

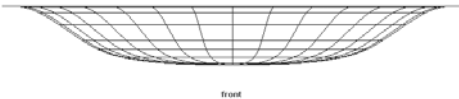


Figure 3.11: Surface revolve on Y-axis

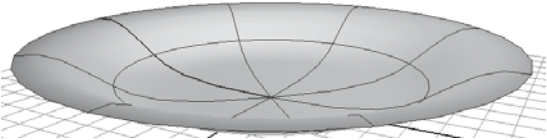


Figure 3.12: Shaded revolved surface in perspective

The choice of which axis to revolve around is very important. It is fairly common to build your curve in the front view with the intention of revolving around the Y-axis; like clay on a potter's wheel, the curve spins around to form the surface. It's perfectly acceptable, however, to spin the curve horizontally on the X-axis. This is similar to spinning a piece of wood on a lathe to create a chair leg or a baseball bat. In Figure 3.13 it has been revolved around the X-axis.

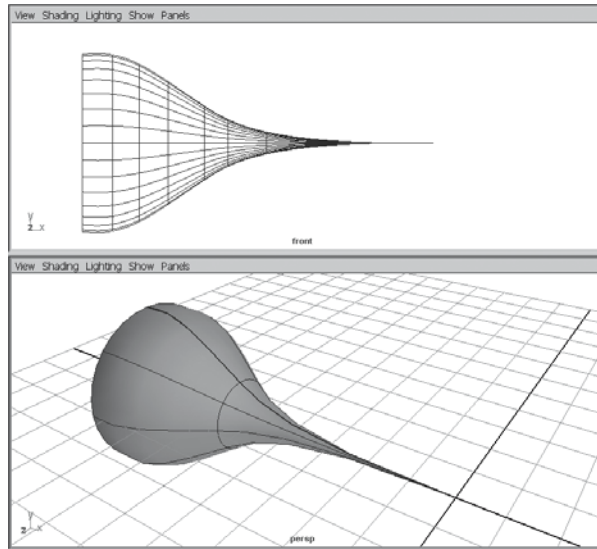


Figure 3.13: Surface revolve on X-axis

### Construction History

An important concept to understand in Maya is that of **construction history**. When you create a model, the inputs that you use are stored as the model's construction history. In the case of a NURBS sphere, that history includes the Radius and Sweep of the sphere. As long as you haven't deleted the sphere's **construction history** you can always alter those inputs that were used to create it (see Figure 3.14 - 3.16).

In the case of a surface that has been derived from a curve (such as the **Revolve** operation), the

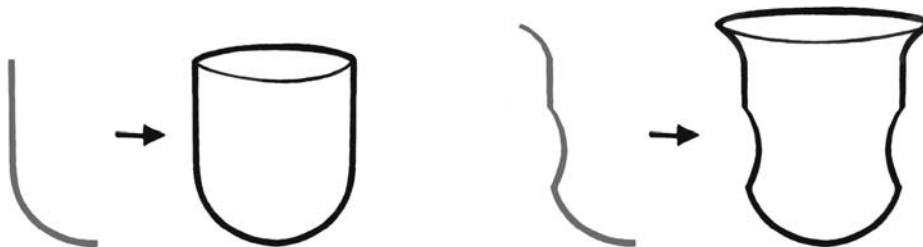


Figure 3.14: Construction history

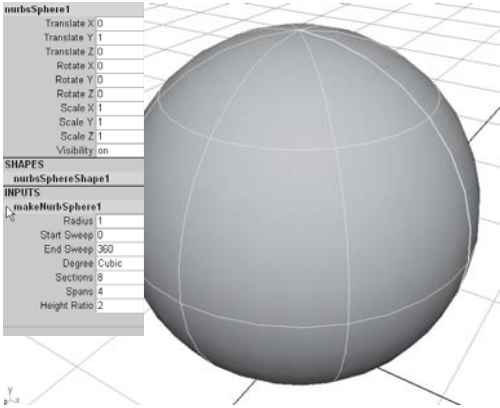


Figure 3.15: Start sweep of 0°

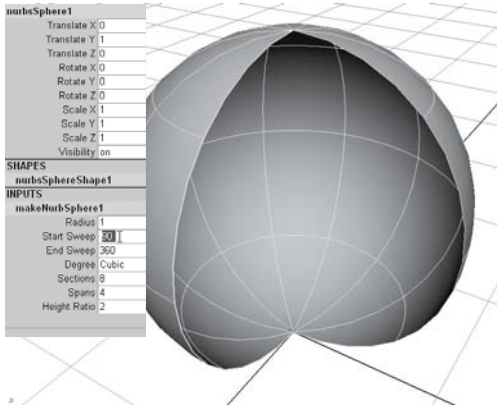


Figure 3.16: Start sweep of 90°

CV Curve you drew is one of the surface's input nodes. If you change the location of one of the curve's CVs, the surface will update to reflect the change. Note the example in Figure 3.17.

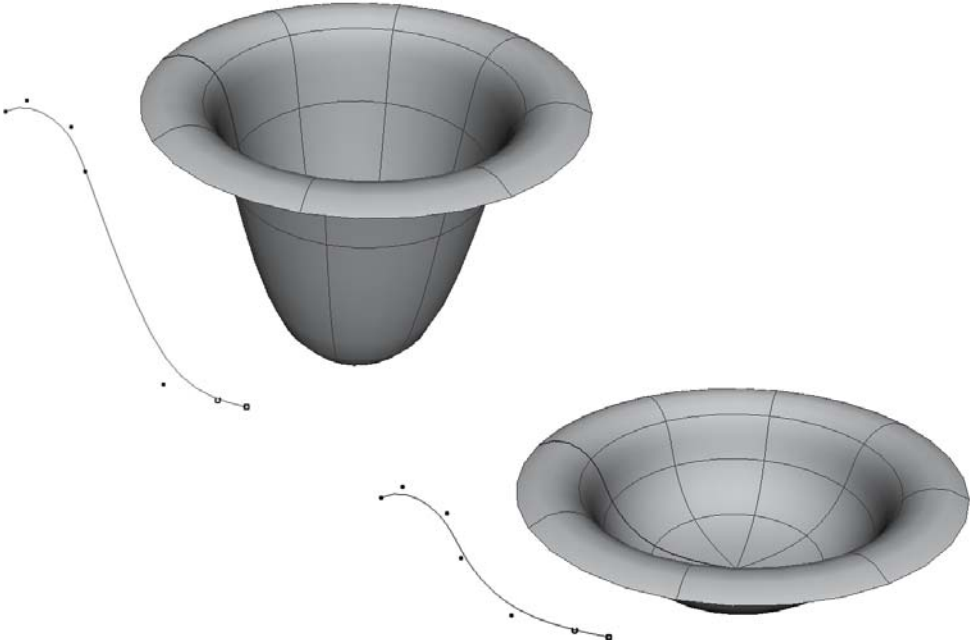


Figure 3.17: Using construction history to reshape a revolve

### Modeling Exercise – Revolves and Curve History

1. Set your project to `understanding_maya\Chapter03\modeling_curve_history_bottles`.
2. Open the scene `modeling_curve_history_bottles_begin.ma`.

3. You'll see a small set of shelves with some milk bottles on them. There is also a set of NURBS curves (see Figure 3.18).

4. The three bottles are each dependent on the curve to their left, since **construction history** was turned on when their curves were revolved. Any changes made to a curve will affect its related bottle surface.

5. Try selecting `bottleCurve01`. **RMB**-click on the curve, and choose **Control Vertex** from the component mask marking menu. Use the **Move Tool** to move some of the CVs. Notice how you are reshaping the `bottle01` object by doing this.



Figure 3.18: Revolved surfaces

6. A good way to create many objects with slight variations is to duplicate a single curve and then alter it before using the **Revolve** command. These variations help prevent your scenes from being overly perfect and uniform. We'll now use the duplicated curves to make some more unique milk bottles.
7. Select `bottleCurve05` and then click **Surface > Revolve options**. Reset the revolve settings and then click the **Revolve** button. This will make a new NURBS surface that has a construction history dependency on `bottleCurve05`.
8. Name the revolved surface `bottle05` and then move it a few units to the right.
9. Now select `bottleCurve05` again, and reshape its CVs in component mode. The `bottle05` object will update as you move the curve's CVs (see Figure 3.19). Often it's easier to model interactively like this than to try to perfect the curve shape without a surface to reference.

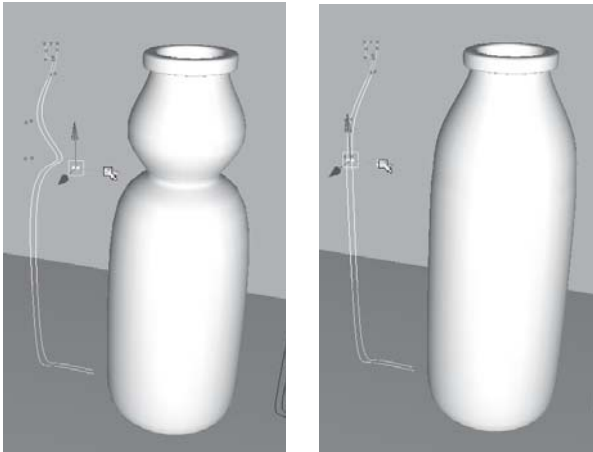


Figure 3.19: Reshaping the bottles through construction history

10. When you are satisfied with the bottle's new shape, repeat the revolve and modification process on the *bottleCurve06*.

11. Here is another interesting thing you can do with **construction history**: Animate the bottle shapes by altering the curves. Try selecting one curve, and then switch to component mode for CVs. Select some CVs, and then press the **s** key on your keyboard to keyframe their positions. Scrub to a new point on the timeline and move the CVs to reshape the curve. Keyframe the CV positions again. When you play the animation you'll see that the bottle that is connected to that curve will also change shape.

### Modeling Exercise – Revolve and Extrude

1. Set the project to `understanding_maya\Chapter03\modeling_revolve_bucket`.
2. Open the scene `modeling_bucket_begin.ma`.
3. You will see a single NURBS curve that has been created with its endpoints aligned to the Y-axis origin.
4. Select the *bucketCurve*. In the **Modeling** menu set, click **Surfaces > Revolve options**. In the options box, choose the defaults by clicking **Edit > Reset Settings**. Since the curve can be swept around its Y-axis to create a bucket surface, the **Axis Preset** should remain on the **Y** setting. Click the **Revolve** button.
5. This creates a NURBS surface in the shape of a bucket (see Figure 3.20). Press **3** on your keyboard to display the full resolution of the object. Rename the object *bucket*.

6. Add two rings to the sides. The NURBS torus will work well for this. Click **Create > NURBS Primitive > Torus options**. Switch the **Axis** to **Z**, **Radius** to **0.5**, and **Minor Radius** to **0.1**. The small **Minor Radius** will make for a thinner, more ring-like shape compared to the default, doughnut-like proportions. Click the **Create** button.

7. Name the object *leftRing*, and move it to the rim on the bucket's left side, or Translate X = 9.7, Translate Y = 15.9.



Figure 3.20: Bucket with rings

8. To make the ring for the other side, select *leftRing* and click **Edit > Duplicate options**. Reset the settings, and then change the **Translate** option to -19.4, 0, 0. (Since the original had been moved 9.7 units on the X-axis, and is a perfectly symmetrical object, a mirror of it can be created by setting the duplicated version to translate back to the origin -9.7 units plus an additional -9.7 to the right side of the *bucket* object, for a total of -19.4 units.) Click the **Duplicate** button.

9. Rename the new ring *rightRing*.

Last, we'll make a handle. For this we'll create another NURBS curve. This time, instead of revolving it, we'll extrude a circle along it. The extrusion will essentially skin a surface along the path of the handle-shaped curve.

10. In Maya, you can begin creating a curve in one view panel and then switch to another panel to continue setting down points. You should take advantage of this feature whenever you are creating a curve that has a three-dimensional shape. Click **Create > CV Curve Tool** and start clicking one point in the **front** view panel right above the *leftRing* object.

11. Switch to the **side** view panel, and click six more points to loop the handle curve through the *leftRing*.

12. Switch back to the **front** view panel and continue creating the handle. When you near the top of the *rightRing*, switch to the **side** view panel and finish the curve as you began it. Press **Enter** on your keyboard to finish the curve (see Figure 3.21).

13. Name the curve *handleCurve*.



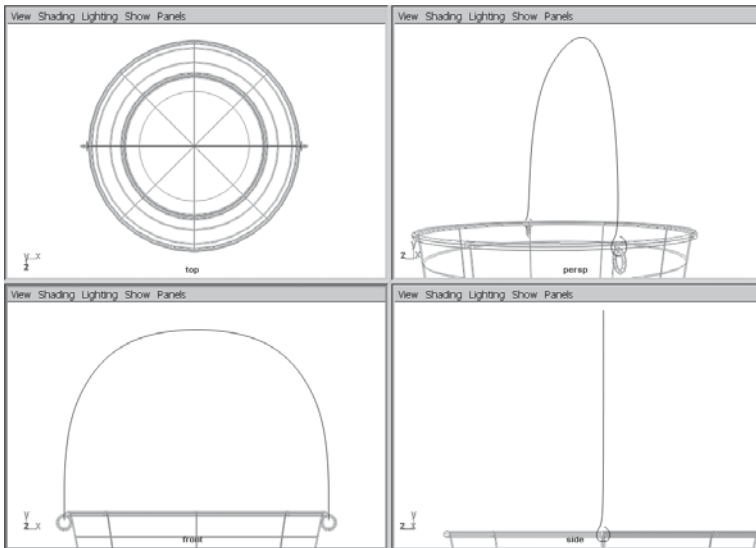


Figure 3.21: Handle curve

14. Next, create a circle to extrude along the *handleCurve*. Click **Create > NURBS Primitives > Circle**. Rename it *profileCircle*.

15. In order to extrude properly, the *profileCircle* must be snapped to the end of the *handleCurve*. Switch to the **Move Tool**. Now, press the **Snap to curves** icon in the **Status Line**. **MMB-click-drag** on the *handleCurve*. You'll notice that the *profileCircle* slides along the *handleCurve*. Continue **MMB-click-dragging** until you reach one end of the *handleCurve*. (see Figure 3.22)

16. Scale the *profileCurve* to 0.175 on all three axes. Rotate it about  $60^\circ$  on the X-axis to bring it in line with the *handleCurve*'s direction.

17. With *profileCircle* selected, shift-select the *handleCurve*. Click **Surfaces > Extrude options**.

18. In the **Extrude Options** window, reset the settings, and then change the **Result**

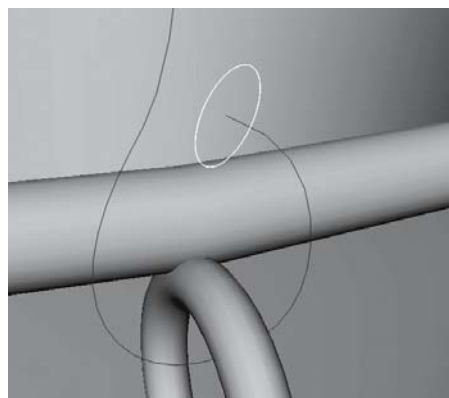


Figure 3.22: Aligned to the profile curve

## 100 3. Intermediate Skills

---

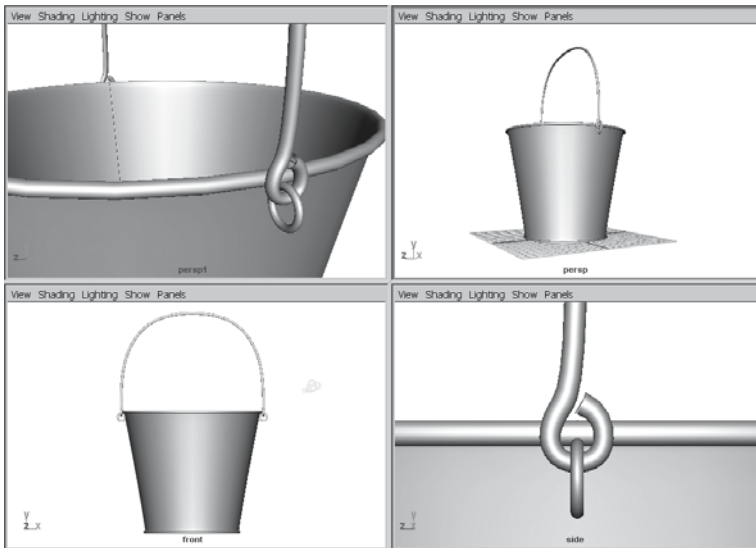


Figure 3.23: Extruded handle surface

**Position** setting to **At Path**. This makes the profile follow the path's position. Click the **Extrude** button. This will create the surface as in Figure 3.23.

19. Name the new surface *bucketHandle*.

20. Save your scene as **modeling\_bucket\_end.ma**.

## Lofting

Another way to generate a NURBS surface from curves is called lofting. Lofting is conceptually the same as building the ribs that are found inside a boat and then stretching a skin across those ribs to create a surface with a precise form. Here are some duplicated closed curves and the loft they generate.

Try building some objects with the loft command, such as a computer mouse or a boat. Notice how the order in which you pick the curves determines the flow of the object.

## Modeling Exercise – Lofting Curves: Slide

1. In the **front** view, use the **CV Curve Tool** to create a curve in the shape of a water slide's cross-section. Make the curve about six units wide and four units tall (see Figure 3.24).

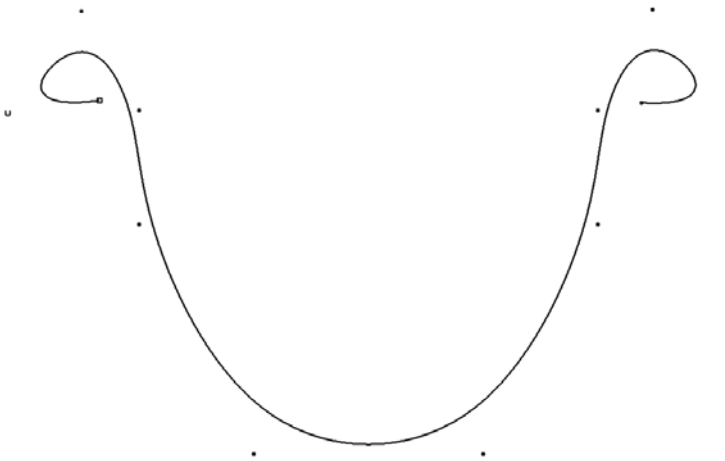


Figure 3.24: Loft profile curve

2. Notice how the curve's pivot point is at the world origin (0,0,0) regardless of where you drew it on screen. Just to make things easier to transform later, we'll center the curve's pivot point by selecting the curve, and then clicking **Modify > Center Pivot**.
3. Lofts are created from a series of curves. You'll generate a better loft when those curves have the same number of points and are drawn in the same order (left to right, for example) as each other. This is a good time to use the **Duplicate** command. With the curve selected, click **Edit > Duplicate options**. In the **Duplicate Options** window, click **Edit > Reset Settings**, to get back to the defaults. Now, we'll tell each successive curve to be moved out on the Z-axis and down on the Y-axis, all while rotating a bit. Try these settings: **Translate: 0, -2, 5**; **Rotate: 0, 30, 0**; and **Number of Copies: 12**. Click the **Duplicate** button (see Figures 3.25 and 3.26).
4. Since the number of copies (12) multiplied by the angle of rotation (30°) is equal to 360°, the duplicated curves form a complete spiral. Select all of the curves in the **Outliner**,

## 102 3. Intermediate Skills

---

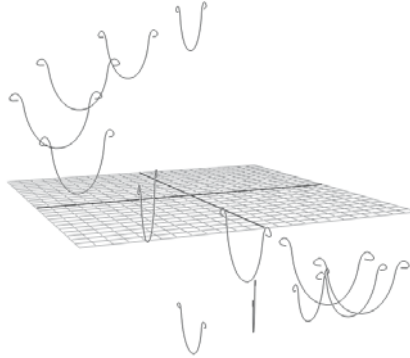
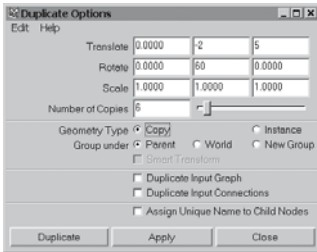


Figure 3.25: Duplication options      Figure 3.26: Loft profiles

starting with *curve1*, and then shift-clicking on *curve13*. (In the next exercise we'll explore how curve selection order will affect the type of lofted surfaces you create.)

5. In the **Modeling Menu Set**, click **Surfaces > Loft options**. Reset the settings, and then click the **Loft** button. This will make a single surface that is stretched across all of your profile curves (see Figure 3.27).

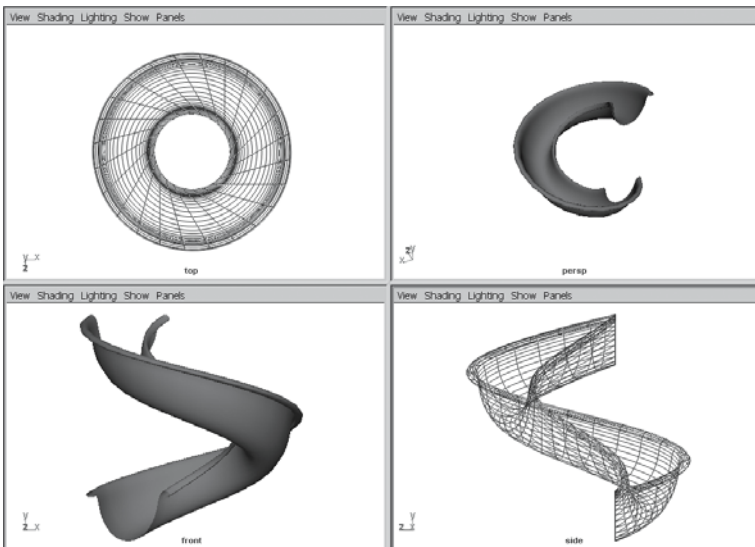


Figure 3.27: Lofted surface

## Modeling Exercise – Lofting Curves: Fish

1. In this exercise we'll take a look at how the selection order of your profile curves affects lofted surfaces. Set your project to `understanding_maya\Chapter03\modeling_lofting`.

2. Open the scene `lofting_order_fish_begin.ma`.

3. You'll see a series of NURBS curves in the scene (see Figure 3.28). They have been shaped to form the outline of a fish. In order to build the surface of the fish model, you'll want to select the curves and then perform a loft. As you'll see, the order in which you select the curves affects the outcome of the loft command.

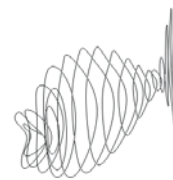


Figure 3.28: Loft curves

4. In the **perspective** panel, select all the curves by **LMB-click-dragging** a marquee around the entire set of curves.

5. With your curves selected, click on **Surfaces > Loft options...**

6. Just to be sure you're using the default settings, click **Edit > Reset Settings** in the **Loft Options** menu, then click the **Loft** button.

7. You should see a strange looking, inside-out fish! Click **3** on the keyboard while the new *loftedSurface1* object is selected. This will place it into fine smoothness mode (see Figure 3.29).

So what happened here? Why is this fish so darned ugly? As it turns out, the marquee selection we used to pick the curves was not specific enough to create a predictable lofted surface. Maya

always lofts curves in the order in which they were selected. By starting out with a vague selection order (the marquee selection picks objects in the same order as their original creation) the loft is not what we expected.

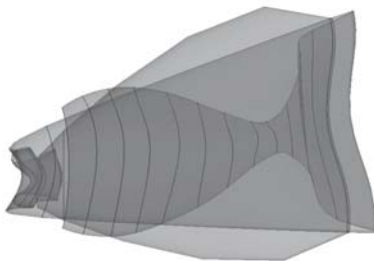


Figure 3.29: Incorrect loft order

8. Delete the poor mangled fish so we can start over again by selecting the *loftedSurface1* object and pressing **Delete** on your keyboard. Be careful not to delete any curves.

9. This time, we'll try to be a bit more specific about

## 104 3. Intermediate Skills

---

selection order. Switch to your **side** panel by **RMB**-clicking on it. Then maximize the **side** panel by tapping the **Spacebar**.

10. While holding the **Shift** key to make a multiple selection, click on each curve starting from the far left and heading to the far right of the screen. You'll notice each curve turn white, except for the final selection, which turns green.

11. Again, generate the surface by clicking **Surfaces > Loft**.

12. Well, this fish is looking a little healthier, but still has some issues (see Figure 3.30). Press **3** to see the smooth representation of the surface. You'll notice that a strict sequential selection of curves made for some problems in the head and mouth area.

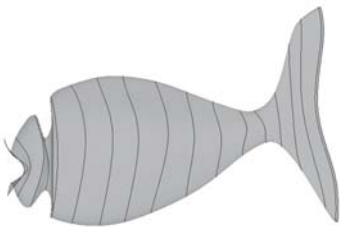


Figure 3.30: Ugly fish

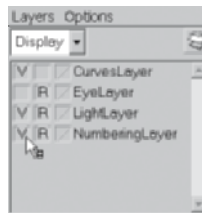


Figure 3.31: Layers

13. Delete this unfortunate freak of nature (*loftedSurface1*) so we can try one final time.

14. There is a very specific order of selection that will work for these curves. To see the order numbered on-screen, turn on the visibility of the *NumberingLayer* in the Layer Editor, by clicking the empty box at the far left of the layer (see Figure 3.31). It will turn to a “V” when clicked and you'll see a set of number annotations appear in the modeling views.

15. Maximize your **side** panel so you can clearly see the order of curve selections. Using the **Shift** selection technique, click each curve from the first through the twentieth.

16. Create the surface by clicking **Surfaces > Loft**.

17. A much happier fish! Go ahead and increase the display smoothness and take a look at the model in the **perspective** panel. Hit **5** on your keyboard to turn on hardware shading mode. To really see what's going on, it can be helpful to make the surface partially transparent (see Figure 3.32). In your **perspective** panel menu, click **Shading > Shade**

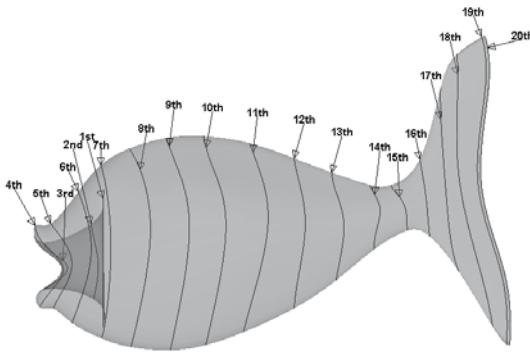


Figure 3.32: X-ray mode

### Options > X-Ray.

Now you can see a bit more clearly how the curve order has been used to create the loft. Look at curves one through eight. Notice how we start with a medium-sized curve set inside the fish's interior, and then move toward the front of the mouth with curves two and three. Curves four, five, and six help define the protrusion of the mouth and then flow toward the head.

18. Let's add a couple of finishing details. First, turn off the **perspective** panel's X-Ray mode. (**Shading > Shade Options > X-Ray.**) Next, hide the numbers and curves by turning the visibility off on the *NumberingLayer* and *CurvesLayer*.

19. Rename the *loftedSurface1* object, calling it *fish*. **RMB**-click on the *fish* object and choose from the marking menu **Materials > Assign Existing Material > Fishphong**. To see the texture, turn on hardware texturing by pressing **6** on your keyboard. You can also turn on the visibility for the *EyeLayer* (see Figure 3.33).

20. Save your scene as **lofting\_order\_fish\_end.ma**.



Figure 3.33: Final fish

### 3.4 Animation – Fine Tuning Animation



What is animation in Maya? One answer is: values that change over time. Think of the pattern we have established in our animation exercises: time-value-key to set the first keyframe, change time-value-key to set the second, and so on. This process is used to lay down the foundation of your animation, the key poses. However, the keys only tell part of the story when it comes to animation. What happens *in between* the keyframes is just as important.

Imagine this. There's a drag race between two cars, one driven by Susie and one by Fred. When the light turns green, both drivers hit the accelerator. Susie's tires grip the road; she takes off smoothly, reaches top speed, sees a brick wall at the end of the track, slows down, and comes to a complete stop, her bumper just touching the wall. Her time: 30 seconds flat. Simultaneously, Fred's bald tires squeal, smoke, and slip for the first 20 seconds of the race. Then, suddenly, he gains traction, shoots down the road at an insane speed, fails to see the brick wall and slams right into it, coming to a complete and sudden stop. Thankfully, Fred's airbag deploys and he escapes injury. Fred's time: also 30 seconds flat.

What is the moral to this gripping story? Susie and Fred started and stopped their cars at exactly the same time and place, but they had wildly different experiences during the race. In animation terms, they had identical keyframes; it was the in-betweens that made all the difference (see Figures 3.34 and 3.35).

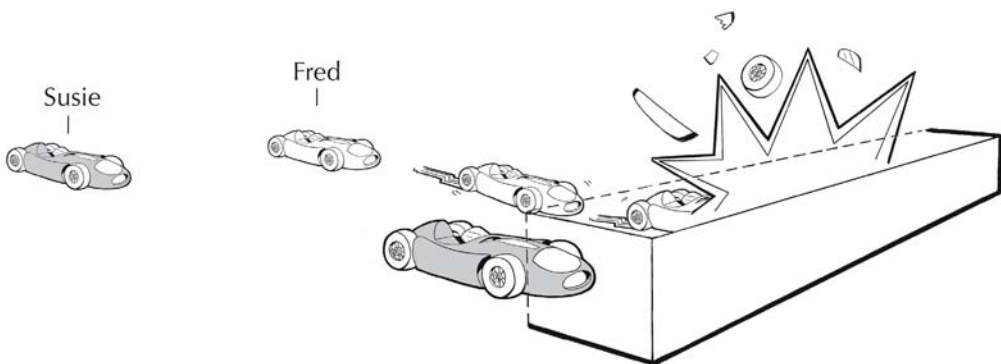


Figure 3.34: Drag racin'



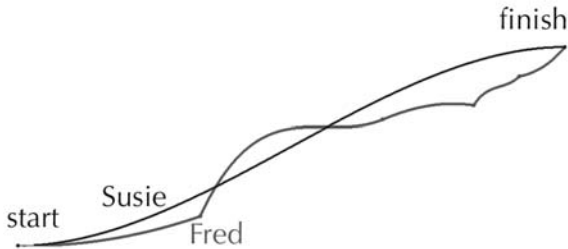


Figure 3.35: Function curves of the race

To have precise control over the timing of your animation you must use the **Graph Editor**. The **Graph Editor** is your home base for tweaking the value and timing of your keyframes. Look at the simple animation and the corresponding **Graph Editor** view in Figure 3.36.

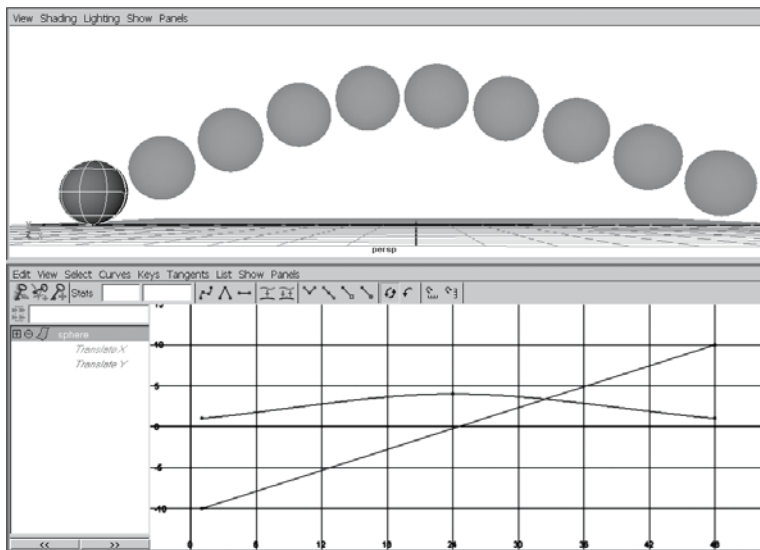


Figure 3.36: Function curves

The vertical axis in the **Graph Editor** represents value while the horizontal axis represents time. Notice that the ball is animating on the Y-axis from 1 to 4 and back to 0 over the course of 48 frames. It travels along the X-axis from -10 to 10.

## 108 3. Intermediate Skills

---

If you were to move the keyframe at frame 24 to the left (say to frame 10) the ball would bounce up to its full height very quickly.

Another important thing to notice is how the curvature of a set of keyframes in the **Graph Editor** mimics the motion's timing and acceleration in the actual animation. If you were to rotate the Bezier handles of a keyframe, the timing of the ball will change to represent a sudden acceleration and gradual deceleration. Try changing the type of line with the Graph Editor's Tangents menu and play back your animation. What kind of animation will these different splines produce?

In Figures 3.37 - 3.39, the keyframes have remained the same, but the in-between frames have changed dramatically.

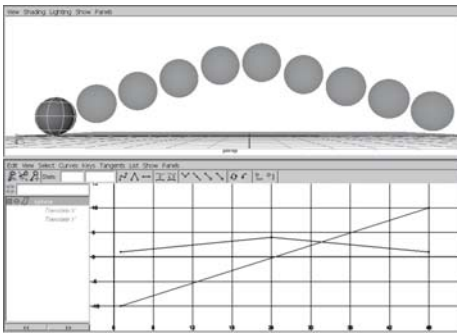


Figure 3.37: Linear tangents

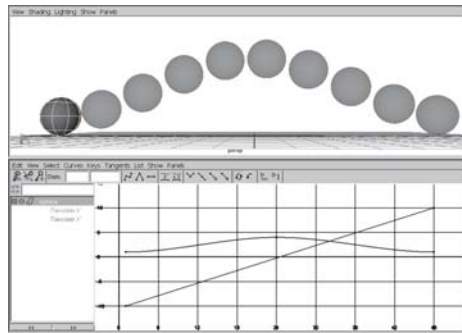


Figure 3.38: Flat tangents

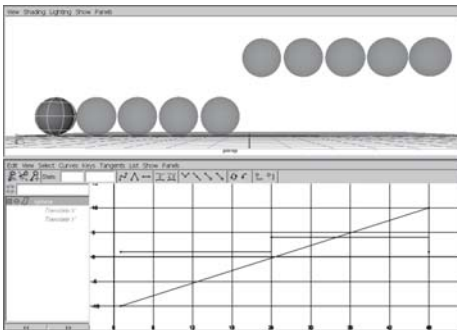


Figure 3.39: Stepped tangents

## Animation Exercise – Tangents

1. Set your project to `understanding_maya\Chapter03\Animation_tangents`.
2. Open the scene `animation_tangent_spheres.ma`.
3. You'll see a sphere sitting on the ground. Make sure your animation playback preferences have been set to real-time (24 fps). Press play. The sphere goes up and down over the course of two seconds. Notice how the sphere's motion is abrupt and mechanical. When the animation is played, the sphere goes from stillness to full speed in an instant, and when it changes directions it does so suddenly. Select the `sphereLinear` object and then move the cursor over the **Graph Editor** panel and press the **a** hotkey. You will see a function curve that describes the sphere's motion. The three points represent keys; the curve represents the in-between frames (also called interpolation).
4. Go ahead and play the animation again with the `sphereLinear` object selected. Notice the function curve in the **Graph Editor** as the sphere goes up and down (see Figure 3.40). See how the abrupt change in direction on frame 24 matches the shape of the curve? The **Graph Editor** is simply a plot of a key's value (the vertical ruler on the left) over time (the horizontal ruler on the bottom). If you were to draw a line straight up from frame 24 until you hit your curve and then draw a line to the left until you hit the value ruler you would find a value of 12 units up on the translateY channel. This works for not only keyframes, but any point in time. Repeat this reverse plotting for frame 12 and you should arrive at a value of 6.

In this case, the type of interpolation between the keyframes has a linear tangency. Linear tangency draws a straight line between keyframes. In effect, there is an equal amount of value change between any given frame and the next. The sphere

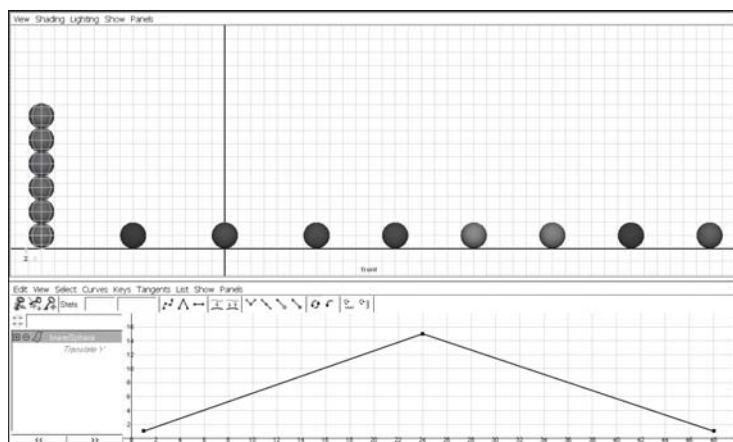


Figure 3.40: Linear tangents

## 110 3. Intermediate Skills

moves exactly 0.5 units every frame of the animation. This is the equivalent of a car that is instantly moving at 45 MPH the moment you touch the gas pedal. In linear tangency there is no acceleration; just instant top speed. In traditional animation this would be a series of evenly spaced drawings.

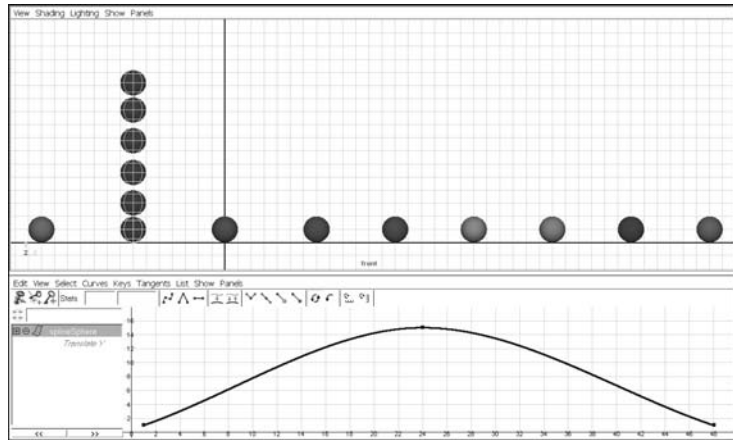


Figure 3.41: Spline tangents

5. In your **Layer Editor** turn on the visibility for the *sphereSplineLayer*. A new sphere appears. Play the animation and notice the difference between the *sphereLinear* and the *sphereSpline*. Keep in mind that both of these spheres have identical keyframes set for their *translateY* channels – they start and stop at the exact same time.

6. Select the *sphereSpline* object and take a look at its function curve in the **Graph Editor** (see Figure 3.41).

This is an example of spline tangency. If you do a reverse plot of the value-over-time you'll see that the in-between values near frame 24 change very little. This corresponds to the sphere's slowness at the top of its height. A 2D animator would achieve

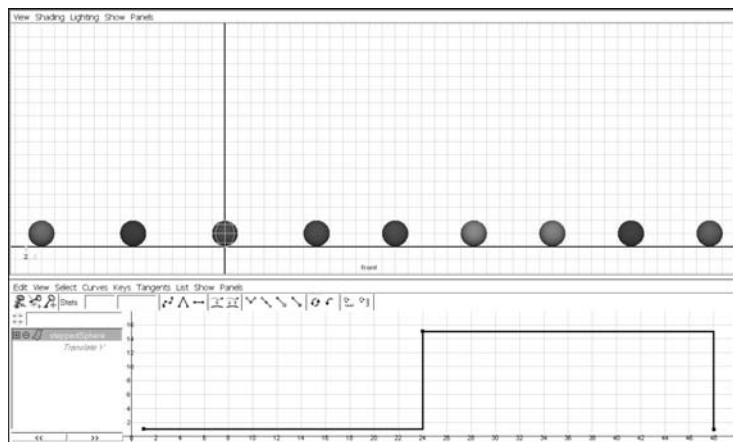


Figure 3.42: Stepped tangents

this with tightly spaced drawings near the keys.

If you take a good look at the *sphereSpline* object's motion it seems to ease its way into and out of the middle keyframe. This ease-in and -out (also called slow-in and -out) is the default motion a spline interpolation will give you.

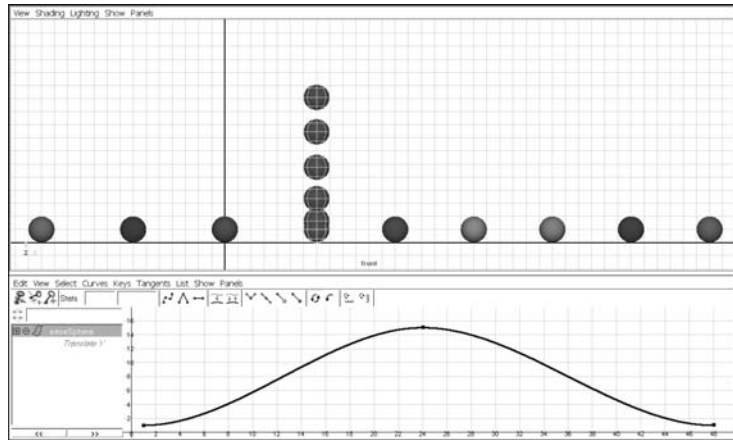


Figure 3.43: Slow-in and slow-out tangents

7. Using the **Layer Editor** again, make the *sphereSteppedLayer* visible. Play the animation and watch the third ball. This one seems to “pop” from position to position.

Take a look at the *sphereStepped* object's function curve (see Figure 3.42). Rather than making any gradual value changes from frame to frame, stepped tangency holds the value at a constant right up until the moment it changes. This is useful for things such as blocking out an animation, blinking lights, and hiding things through their visibility channel.

8. Linear, spline, and stepped are the main curve tangency types. You can vary the

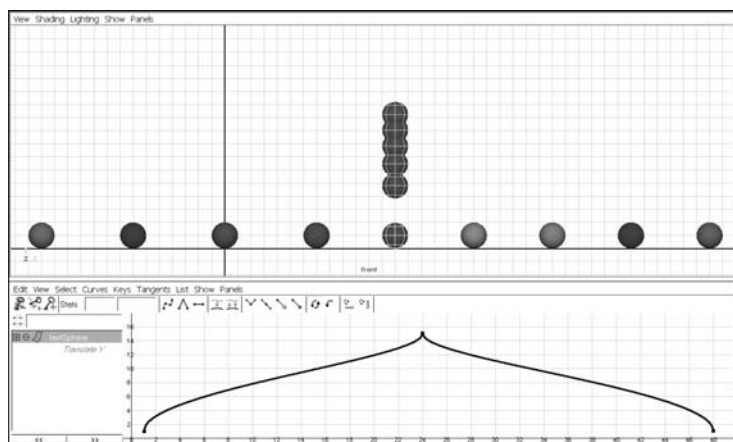
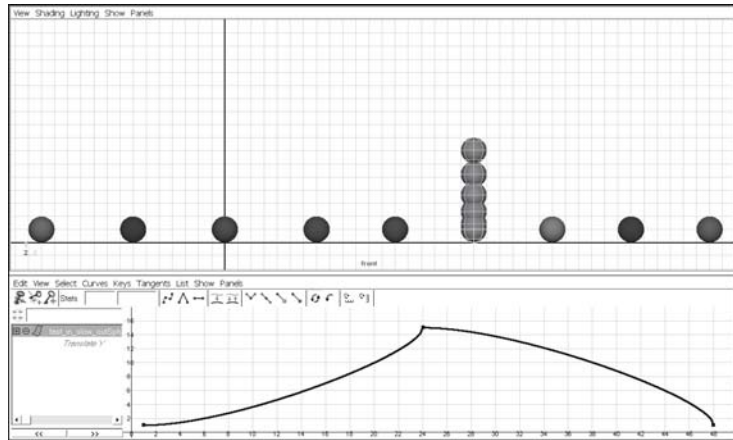


Figure 3.44: Fast-in and fast-out tangents

## 112 3. Intermediate Skills

in- and out-tangency for any keyframe, not just the entire curve. By altering the direction and weight of a spline tangency, you can create other types of motion. In the **Layer Editor**, turn on the visibility of the *slow\_in\_outLayer*.



9. Select the *sphereSlow* object and look at its function curve (see Figure 3.43). Similar to spline tangency, this flat tangency makes the curve slow into and out of each keyframe. There will be gradual acceleration and gradual deceleration at all value changes.

10. Turn on the visibility of the *fast\_in\_outLayer*. Select *sphereFast* and look at its curve (see Figure 3.44). Based on spline tangency, these in-betweens have large value changes from frame to frame. This causes rapid acceleration and rapid deceleration at the keyframes. In traditional animation this is the equivalent of drawings that are spaced far apart. Fast-in and -out tangency is achieved by breaking the tangency of the curve at a keyframe. We'll look at how to do this in the next exercise.

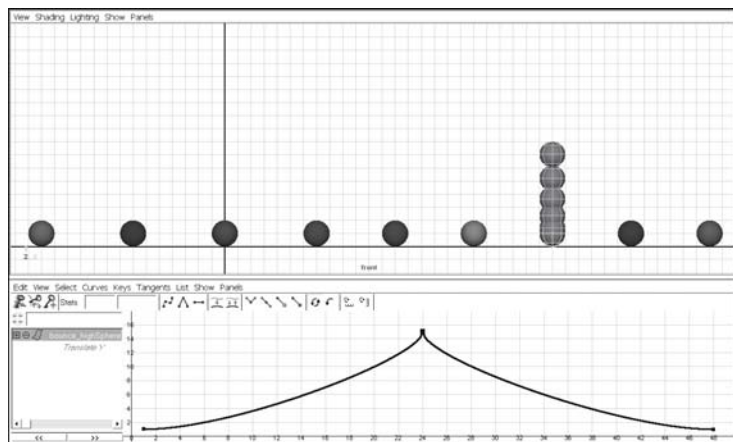


Figure 3.46: Inverted bounce tangents

11. Turn on the visibility of the four remaining layers. These spheres all have variations on spline tangency. By breaking and varying the direction and weight of the function curves, it is possible to make the sphere rapidly accelerate, stick to the ceiling, and then gently drop (fast-in, slow-out), bounce off the ceiling (bounce-high), bounce off of the floor (bounce-low), and anticipate and overshoot its positions (see Figures 3.45 - 3.48).

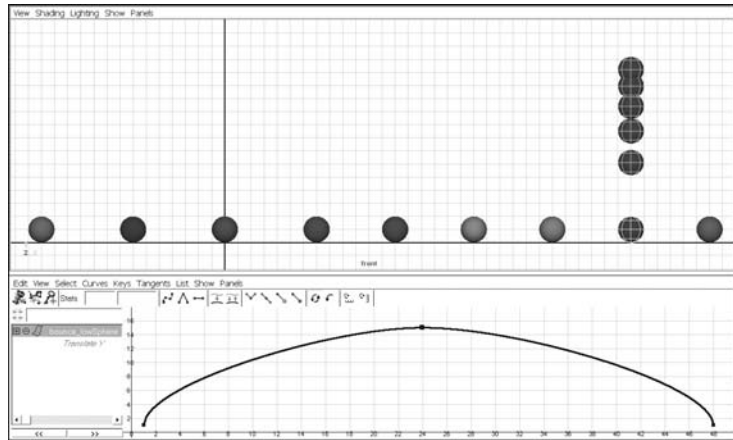


Figure 3.47: Hang-time tangents

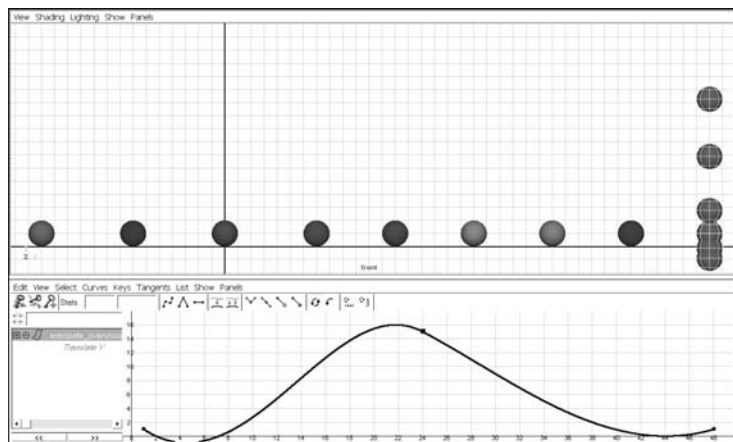


Figure 3.48: Overshooting tangents

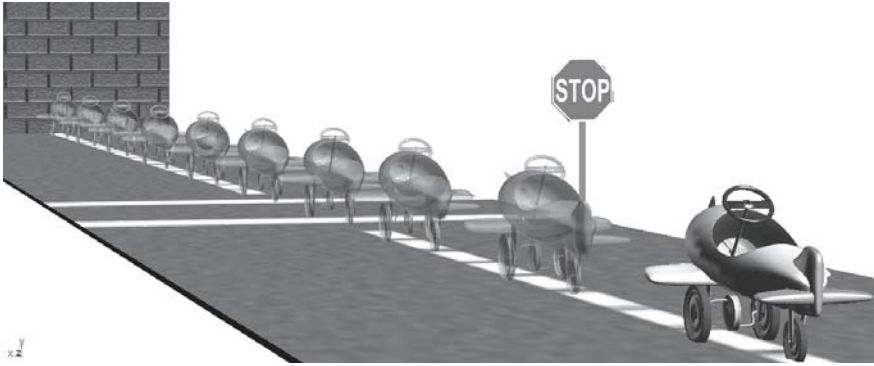


Figure 3.49: Linear plane animation

### Animation Exercise – Tangent Plane

To edit function curve tangency in a more practical example, we'll control the acceleration and braking of the airplane model.

1. Set your project to `understanding_maya\Chapter03\animation_tangent_plane`.
2. Open the scene `animation_tangent_plane_begin.ma`.
3. Press **play** – you'll see the airplane moving at a constant rate (see Figure 3.49).
4. In Hierarchy mode, select the `pursuitPlaneGroup`. (You can switch to Hierarchy mode by clicking the **Select by hierarchy and combinations** button on the **Status Line**.)
5. Open the **Graph Editor** and look at the TranslateZ channel. You will see a set of keyframes that mark the times when the plane should start, stop at the stop sign, accelerate, and then stop again. Currently, these keyframes are all on a straight line, so there is only a constant motion from beginning to end.
6. You will select and move the keyframes in the **Graph Editor** in order to alter the timing of the animation. First, advance to frame 65 on the timeline. Choose the **Move Tool** from the **Tool Box** (you can also press the **w** hotkey on your keyboard) so that you can move the keyframes. Now select the third keyframe, which is at frame 65.
7. In the **Graph Editor**, some things work a bit differently than in the model view panels. The **Move Tool**, for example, is used by pressing the middle mouse button instead of



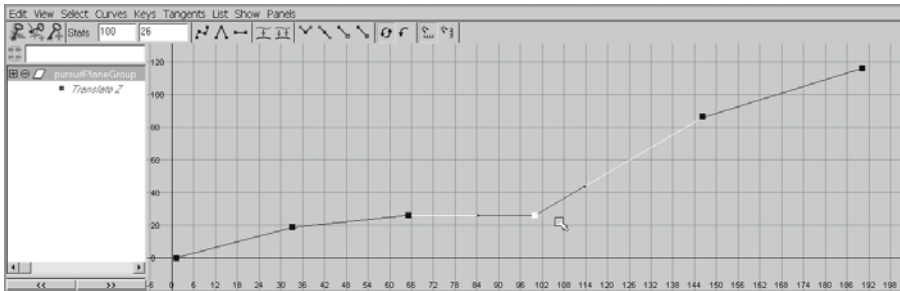


Figure 3.50: Retiming keyframes three and four

the usual left mouse button. **MMB-click-drag** the keyframe in order to move its time and value. Horizontal movement changes the time of the keyframe, while vertical movement changes the value. You should try to get the plane to stop in front of the stop sign by moving the keyframe to a value of 26.

- With the third keyframe selected, hold the **Shift** key on your keyboard while **MMB**-dragging down. Watch the **Key Value** data field in the upper left corner of the **Graph Editor**. Bring this keyframe down to a value of 26. (You can type the number directly into the **Key Value** field if you like.) You'll see the plane move back in front of the stop sign.
- We want to have the plane stop for a few seconds. Select the fourth keyframe (the one at frame 100) and move it to a value of 26. This means that the value is constant for 74 frames – about three seconds. Notice the shape of your function curve in the **Graph Editor**: A flat line indicates a hold during animation (see Figure 3.50).
- Rewind and then play back your animation. You'll see the plane stop and hold in front of the sign for a few seconds before continuing on to the end.

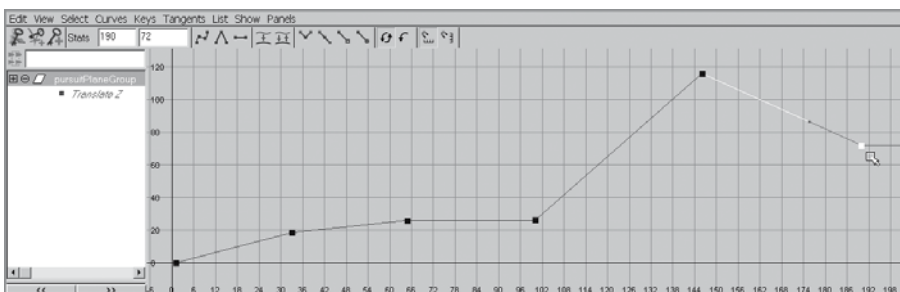


Figure 3.51: New keyframe values

## 116 3. Intermediate Skills

---

8. Next we'll get the plane to hit the wall much sooner, so that we have some time at the end for it to bounce back. Set the time slider to frame 146. Now, select the keyframe at frame 146. Increase its value in the **Graph Editor** until it is touching the wall, or 116 units (see Figure 3.51). (Who the heck puts a brick wall at the end of a runway?!) Play the animation to see the new result.

9. Instead of allowing the plane to hold for the last two seconds, we'll make it roll backward a bit. Advance the time slider to 190. In the **Graph Editor**, select the last keyframe and move its value down to about 72 units (see Figure 3.51). This time when you play the animation you'll see the plane bounce back off the wall.

At this point you've worked out all the key poses and timings of the animation, but the in-between frames are very stiff and mechanical. It's time for you to add some finesse to the animation. When the keyframes were initially created for this animation, they were created with **linear** interpolation. This is the equivalent to evenly spacing the in-betweens; the points between the extremes are evenly subdivided.

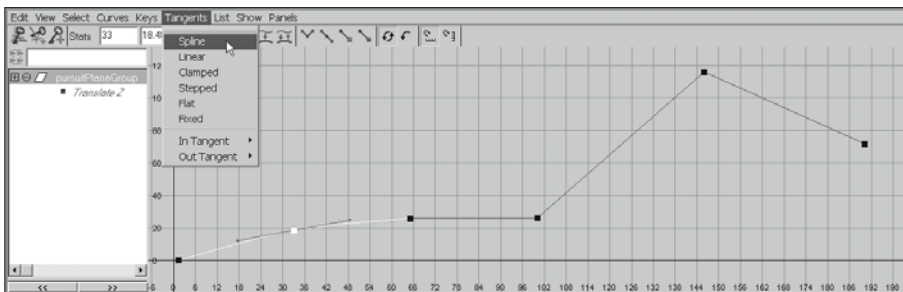


Figure 3.52: Eased acceleration

To create better timing between the keys we need to alter the shape and tangency of the function curve. Each keyframe has an incoming and an outgoing tangency type and direction. These can be controlled with Bezier handles.

10. Select the keyframe at frame 33. In the **Graph Editor** notice how the line segment on either side of this point is perfectly straight. This means there is no gradual deceleration as the plane reaches the stop sign. The speed change is very abrupt. In the **Graph Editor** menu, click on **Tangents > Spline**. You've made a much gentler speed change (see Figure 3.52). Play back your animation to see the plane ease into the stop.

Easing-into or easing-out of a position is also known as a slow-in and a slow-out. During a slow-in the motion cushions gently into a position, like the plane now does at the stop sign. A slow-out is a gentle acceleration.

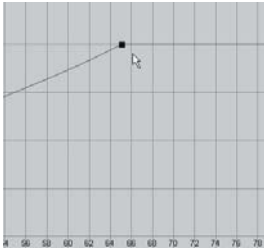


Figure 3.53: The hitch

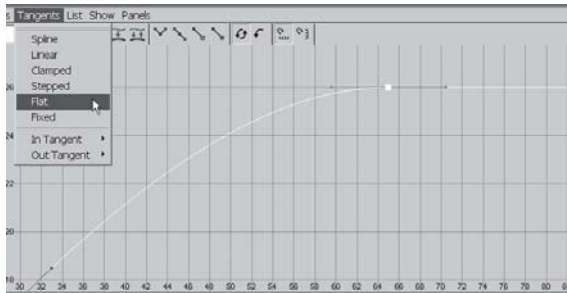


Figure 3.54: Flattened tangents to the rescue

11. Look closely at the way the curve reaches the third key at frame 65 (see Figure 3.53). There is a bit of a hitch in the curve, which translates into a jerk in the motion as the plane stops. To smooth this out, we need to flatten the tangency as it makes its way into the key. To do this, begin by selecting the third keyframe in the **Graph Editor**. Next, drag a selection box around the blue handle on the left of the keyframe. This is the **In Tangent** handle for the keyframe. Click (in the **Graph Editor** menu) on **Tangents > Flat**. This has smoothed the kink out of the curve (see Figure 3.54). Try playing back the animation to see the results. You now have a perfect deceleration (slow-in) at the stop sign.

12. You probably want to create a more realistic acceleration after the hold. Currently, the plane jumps from 0 to 60 in an instant. A slow-out will do just the trick. Select the fourth key at frame 100 in the **Graph Editor**. Next select the **Out Tangent** handle to the right of the keyframe. Again, make the tangency flat, this time by clicking the **Flat tangents** icon. Play the animation to see your new improved acceleration.

13. Now, we'll make the plane bounce sharply off the wall. This can be accomplished through fast-in and fast-out tangency for the key at frame 146. To do this, select the keyframe and then highlight the blue **In Tangent** handle. **MMB-click-drag** it down to about an  $80^\circ$  angle as in Figure 3.57.

14. Select the brown **Out Tangent** handle on the right side of the keyframe. Drag its value down to about a  $-70^\circ$  angle. This will create a sharp peak as the curve enters and exits the key. This shape is used for fast bounces (see Figures 3.55 - 3.57).

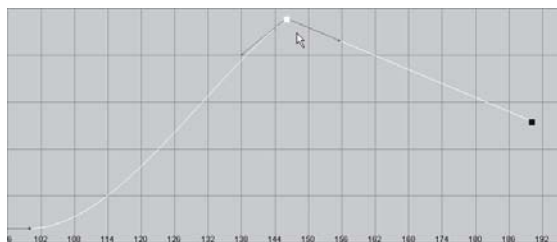


Figure 3.55: Frame 146

## 118 3. Intermediate Skills

---

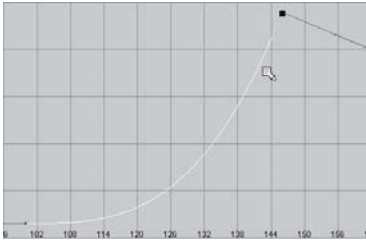


Figure 3.56: In tangent at  $80^\circ$

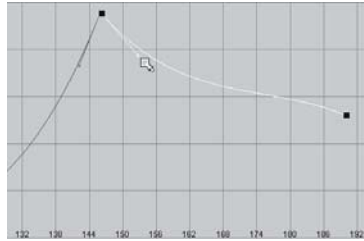


Figure 3.57: Out tangent at  $-70^\circ$

15. Give the final keyframe a nice slow-in to finish things off (see Figure 3.58). Play the animation to see the final results (see Figure 3.59). Remember, the keyframes changed very little during the course of this exercise - it's the in-betweens that you really got to do all the work.

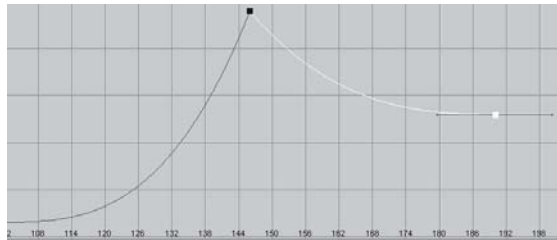


Figure 3.58: Soft braking at the end

16. Save your scene as `animation_graphEditor_plane.ma`.

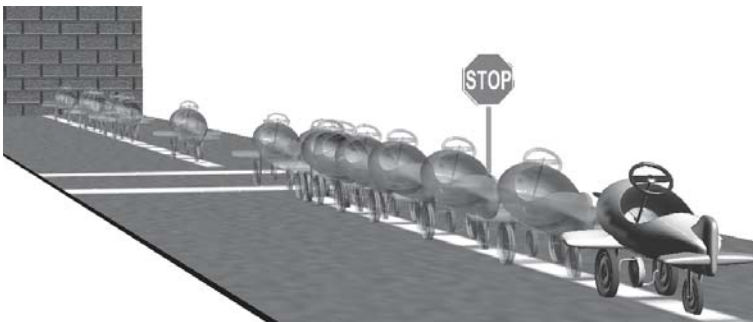


Figure 3.59: The finished animation

## 3.5 Shading – Texture Maps



So far, all of our shaders have used simple RGB values in the color attribute. What if you want something more interesting? Well, any image file you can create – through painting, scanning, photography, or image creation programs such as Adobe Photoshop – can be used as a texture map on your shader (see Figures 3.60).

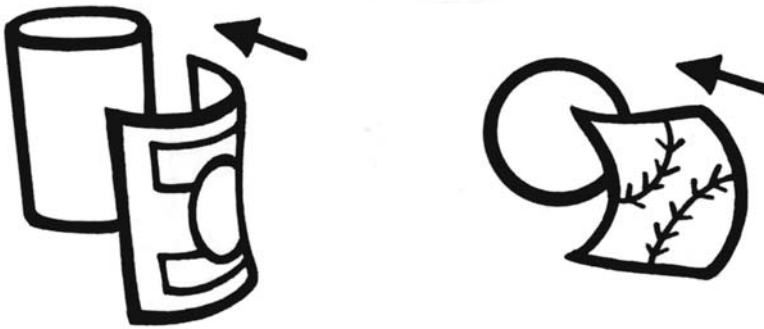


Figure 3.60: Texturing an object

By clicking the checker icon next to the **Color** attribute of a material you will bring up the **Create Render Node** box. Choose **File** from the **2D Textures** selector. Now, browse to an image file such as a .jpg, .tga, .tif, or .bmp by clicking the folder icon. (The default place to store your images is in the current project's sourceimages folder.) This image now covers the surface color of the material (see Figure 3.61).

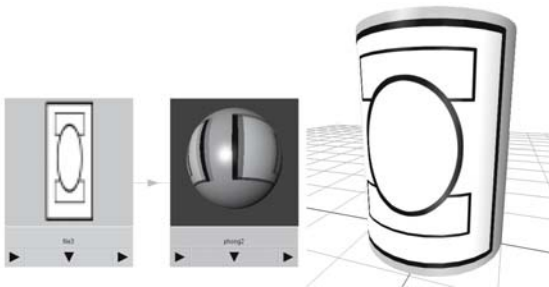


Figure 3.61: File texture in color attribute

## 120 3. Intermediate Skills

---

### Shading Exercise – Barn Textures

1. Set your project to `understanding_maya\Chapter03\shading_barn`.
2. Open the scene `color_barn_begin.ma`.
3. You'll see the barn you created waiting for some texture to be applied (see Figure 3.62).
4. To begin, open the **Hypershade** and create a new Lambert material. **RMB**-click on the material, choose **Rename** from the contextual menu, and rename it *lambertRoof*. (Hit the **Enter** key when you are done.)

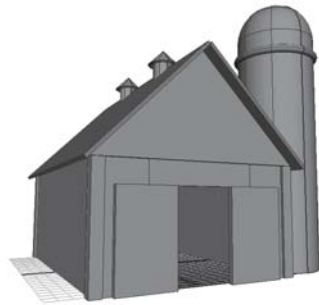


Figure 3.62: Untextured barn

5. Double-click the *lambertRoof* material sample in the **Hypershade** in order to open it into the **Attribute Editor**. Notice the area called **Common Material Attributes**; this is where we will change the **Color** attribute from a simple value to a texture mapped value. To the right of the **Color** swatch and slider is the **Create Render Node** button (the one that looks like a 2 x 2 checker). We are going to connect a file texture to this input. Click on the **Create Render Node** button. This opens the **Create Render Node** window. With the **Textures** tab on top, make sure that the **With New Texture Placement** box is checkmarked. Now, in the **2D Textures** section, be sure that **Normal** is the selected mode. Finally, click the **File** button (see Figures 3.63 and 3.64).

So far you've specified that the material's **Color** attribute is going to be mapped to a **File** texture. This file texture will be created with a normal, UV-based texture placement utility.

6. The **Attribute Editor** is now focused on the **File** node. This is where we can specify which file to use. In the **File Attributes** section, look for the **Image Name** attribute. To

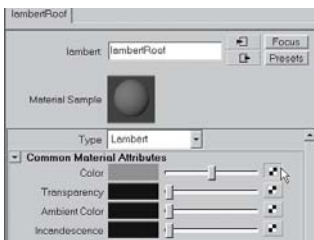


Figure 3.63: Mapping a file

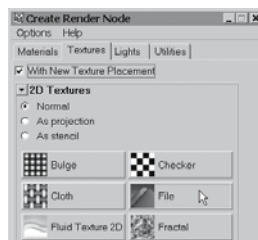


Figure 3.64: Create file

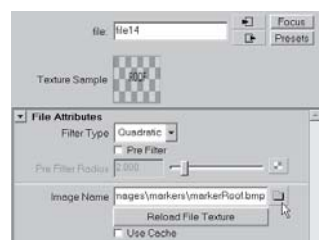


Figure 3.65: File selection

the right of this is a folder icon. Click the icon to launch a file browser. In the file browser, find the **sourceimages/markers** directory of the current project. Here you'll choose the **markerRoof.bmp** image and click **Open** (see Figure 3.65).

7. Your *lambertRoof* material sample in the **Hypershade** is now colored like the file texture mapped to its **Color** attribute. **MMB**-click-drag the material from the **Hypershade** onto the *barnRoof01* and *barnRoof02* objects.

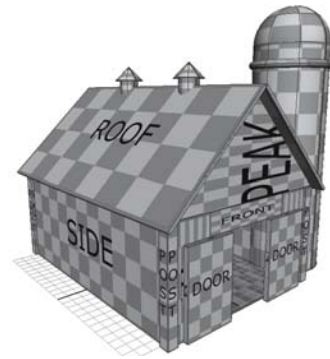


Figure 3.66: Textured barn

8. In the **perspective** panel, click the **6** key on your keyboard to turn on smooth shading and hardware texturing. You should see the checkerboard pattern with the word “Roof” on it (see Figure 3.66). The other objects already have textures applied. These markers are used to help you align the mapping of textures to your objects. Once the mapping is correct, you can swap the markers for your final textures.

There are two things working together to make your texture map appear correctly on an object. One is the **place2dTexture** placement utility connected to the file. The other is the UV mapping of the object itself. Let's take a look at these things and how they work together.

9. In the **Hypershade**, select the *lambertRoof* material (a yellow outline surrounds the selected material). In the **Hypershade** menu, click **Graph > Input Connections**. You'll now see in the **Work Area** of the **Hypershade** three connected nodes (see Figure 3.67). The *place2dTexture14* utility node, the *file14* node (this contains a link to the bitmapped graphic file named *markerRoof.bmp*), and finally the *lambertRoof* material node. Double-click the *place2dTexture14* node to open it in the **Attribute Editor**.

10. The *place2dTexture14* node determines how the file texture is displayed within the UV texture space of the object. The roof objects were originally created from polygon cube primitives that had automatically assigned texture coordinates. Each face of those cubes was set to fill the entire UV texture space. So the **Coverage** setting of 1.000 (U) and 1.000 (V) that you see in the **Attribute Editor** right now places one iteration of the file texture inside each face, from corner to corner.

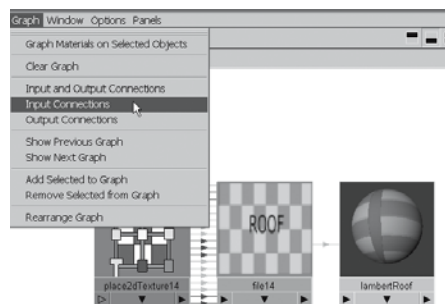


Figure 3.67: Graph input connections

## 122 3. Intermediate Skills

---

a. Try changing the **Coverage** to 0.5(U) and 0.5 (V) and see what happens. Only one quarter of the available UV space is covered by the file texture (see Figure 3.68).

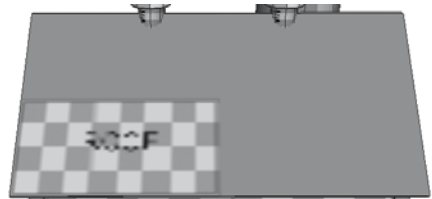


Figure 3.68: Coverage of 0.5, 0.5

b. Now, try a **Coverage** of 2 (U) and 2 (V). The UV space of each face is only able to display one quarter of the total file texture (see Figure 3.69). When done, reset the **Coverage** to 1 (U) and 1 (V).

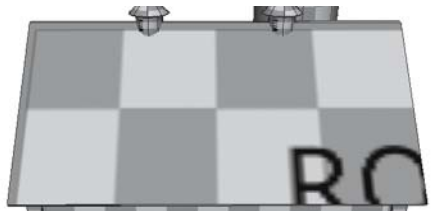


Figure 3.69: Coverage of 2, 2

What this means, is that based on the UV mapping coordinates of a given surface, the *place2dTexture* utility can be used to display more or less of a texture within that space. For another example, try changing the **Repeat** UV settings to 2, 2. Again, the UV texture space of each surface remains the same, but the number of texture iterations within that space has changed (see Figure 3.70).



Figure 3.70: Repeat of 2, 2

11. Reset the **Repeat** UV settings to 1, 1.

12. One look at the *barnRoof02* object and you can tell that there's a texture-mapping problem. The word "Roof" is upside down. You may want to try fixing this problem by altering some settings in the *place2dTexture1* node. Take a look at the **Rotate** UV attribute. If you set the slider to a value of 180° you'll see the entire texture rotate to the correct orientation. There is a problem, however; take a look at the *barnRoof01* object. Its texture is now upside down! Reset the **Rotate** UV attribute to 0.

13. So what's going on? How can you fix one side of the roof without ruining the other? The answer lies in the UV mapping of the individual objects. You see, the *lambertRoof* material (along with its connected *file1* texture and *place2dTexture1* utility) is assigned to both objects. Any changes to the material will be seen on both objects. What *can* be changed is the explicit UV mapping of the individual objects.

14. Select the *barnRoof02* object in the **Perspective** panel. Then click **Windows > UV Texture Editor...** The **UV Texture Editor** shows you the UV coordinates of your model in a flattened view. Click **Image > Display Image** in order to turn off the *markerRoof.bmp*



image (which will drive you mad if you stare at it for too long). What you'll see is the unwrapped cube sitting on the UV texture space (see Figure 3.71). (If you don't see anything at first, dolly out or press the **a** key on your keyboard to frame the unwrapped cube.)

15. We want to rotate the *barnRoof02* object's UVs  $180^\circ$  in order to flip the texture. **RMB**-click anywhere inside the **UV Texture Editor** and choose **UV** from the **marking menu**. Now, marquee-select across the entire wireframe to pick all of the UVs. They'll highlight to a green color when selected.

16. With the UVs selected, click in the **UV Texture Editor's** menu on **Polygons > Rotate UVs options**. Change the **Rotation Angle** setting to  $180^\circ$ . Click the **Rotate** button. The texture now looks correct on the *barnRoof02* object (see Figure 3.72).

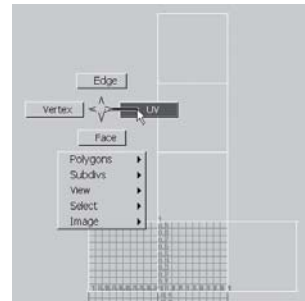


Figure 3.71: Picking UVs

17. You can now close the **UV Texture Editor**.

18. Use this same technique to correct the *barnWall02* object's UV coordinates.

19. On many workstations, once you alter the settings of the *place2dTexture* node, your hardware texturing will get very blurry. To correct this, select the material node (*lambertPeak* in this case) and look for the **Hardware Texturing** settings in the **Attribute Editor**. Change the **Texture quality**

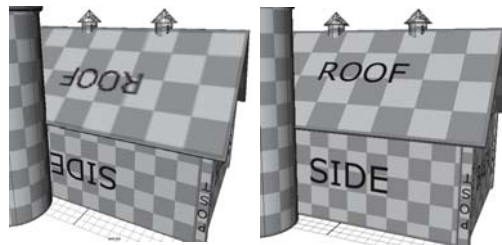


Figure 3.72: Before and after rotating UVs

attribute from **Default** to **High** or **Highest**. Only change this setting when making critical adjustments; you may experience slow frame rates depending on your graphics card's memory, drivers, and chipset if too many materials are displaying high-quality textures. Note: These settings do not alter the texture quality of software rendering. Only real-time, OpenGL-based rendering is affected.

20. The peaks of the barn are in need of some texture rotation. Unlike the roof objects, this can be fixed in the texture placement utility. Remember, changes in the texture placement utility will be seen on all objects that share a material assignment. In this case, such

## 124 3. Intermediate Skills

---

a change is a good thing, since it speeds up your work. In the **Hypershade**, select the *lamBERTPeak* material. Now, select the *place2dTexture5* tab in the **Attribute Editor**. Set the value for **RotateUV** to 90°.

21. This UV rotation has taken care of the texture rotation on the *barnWall07* and *barnWall08* objects, but now it's evident that the texture is misaligned and stretched. We can fix these problems by changing the translation and repetition values of the texture placement node. Try changing the **Translate Frame** settings and the **RepeatUV** settings until you see the red border edges and the checkers are square. (I settled on **Translate Frame** = 0.260, -0.080 and **RepeatUV** = 1.350, 1.190 as seen in Figure 3.73).

22. Another object that can be corrected through the *place 2dTexture* node is the *siloWalls* object. Pick the *lamBERTSilo* material in the **Hypershade**. In the **Attribute Editor**, click on the input connections button next to the **Color** slot. Choose the *place2dTexture8* tab at the top of the **Attribute Editor**. Change the **RotateUV** value to 90°.



Figure 3.73: Original; RotateUV = 90°; Translate Frame = 0.26, -0.08 and RepeatUV = 1.35, 1.19

23. This orients the texture correctly, but it would be nice to have the red border seam at the back of the object, with the word “Silo” on the front. To change this, you can alter the **Offset** value of the *place2dTexture8* node. The second field (V) corresponds to this direction in the texture space of the cylinder. Set the **Offset** values to 0 (U) and 0.5 (V). Now the texture faces the front of the object.

24. Once again, the texture may have become blurry. If you need to sharpen it for placement purposes, increase the **Hardware Texturing > Texture quality** setting. This is found in the *lamBERTSilo* material node.

25. The dome on top of the silo can be fixed in a similar way. Select the *lamBERTDome* node, open the *place2dTexture9* node in the **Attribute Editor**, and set the **RotateUV** attribute to 90°. Set the **Offset** (V) value to 0.5.

26. The *siloring* object is also going to need a texture placement node adjustment. Again, the **RotateUV** must be set to  $90^\circ$  to correct the orientation. The **Offset (V)** should be set to 0.5 to place the word “Ring” on the front of the silo ring. Since the texture is currently wrapped all the way around the object, half of it is hidden. On the U direction of the object, we want to repeat the texture twice – once for the visible outer portion of the object, and once for the hidden inner section. Set the **RepeatUV** setting to 2 (U) and 1 (V). Notice how this makes the red border edge of the placement marker visible on all edges of the object.

27. For the *barnVent* objects and the *barnHood* objects protruding from the barn’s roof, the same technique you used on the *silowalls* will work. Choose the *lambertVent* material and change its *place2dTexture11* node’s settings to **RotateUV** =  $90^\circ$  and **Offset** = 0 (U) and 0.5 (V). Use these same settings on the *lambertHood* material’s *place2dTexture12* node (see Figure 3.74).

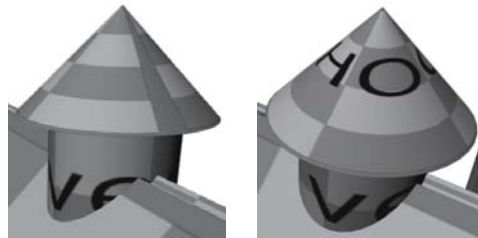


Figure 3.74: Fixing the vent and hood

28. The *barnFloor* object will be easy to fix. Change the *lambertFloor* material’s texture placement node to a **RotateUV** value of  $270^\circ$ .

The *barnPost03* and *barnPost04* objects have some problems with UV orientation on their outside faces. In this case, we’ll need to apply a completely new set of mapping coordinates to those individual faces. The posts were created originally from polygon cube primitives, with their default texture settings. In this mode (named **Normalize the Whole Object**) the faces of the cube have shared UV texture points. This is similar to having neighboring polygons share vertices. If you move a vertex on a cube you affect all connected faces. The same is true of moving UVs when they are shared. What this means is that you cannot alter the UVs of the one problem face without causing new problems on the connected face. What we want is to normalize the UV textures onto individual faces.

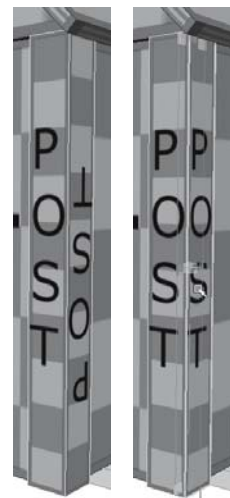


Figure 3.75: Fixing posts

29. Select the *barnPost03* object. In component mode (F8), select the face with the upside-down “Post” writing. In the **Modeling** menu set, click on **Edit Polygons > Texture > Planar Mapping options...** Here you are able to choose the size, po-

## 126 3. Intermediate Skills

---

sitioning, and direction of the new set of UV coordinates. Click **Edit > Reset Settings** to choose the defaults. Change the **Image Scale** value to -1 (U) and 1 (V). (This will keep the coordinates from going on backward.)

30. Click the **Project** button. You'll see a correctly oriented texture marker on the side of the post, as in Figure 3.75. (You'll want to return to object mode after applying projections; Maya automatically selects the object's faces after you create a projection.)

31. Repeat this process for the *barnPost04* object.

32. We'll use a similar technique to unify the three objects that comprise the front wall of the barn. When multiple faces or multiple objects are selected, the **Planar Mapping** command will fit the projection around the entire selection area. First, select the *barnWall04*, *barnWall05*, and *barnWall06* objects. Then click on **Edit Polygons > Texture > Planar Mapping options...** Click **Edit > Reset Settings** to reset to the defaults. Looking at the scene in the **perspective** panel, you can see that the front walls of the barn are facing down the X-axis. So in order to map the projection properly, click on the **Mapping Direction** setting's **X axis** radio button. Then click the **Project** button (see Figures 3.76 and 3.77).

Now the three front wall objects have consistent mapping coordinates. This means that you won't see any seams between the objects, and they will display textures with the same density and aspect ratio. This density of texture on an object's surface is known as texel density. When the objects in a scene have uniform texel density they appear to belong together. When the texel densities are mismatched, your scene will look more like a bunch of individually constructed parts, rather than a collective whole.

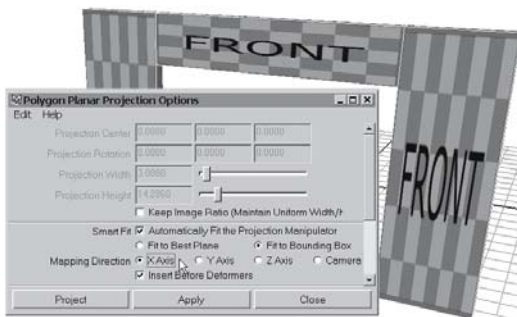


Figure 3.76: Planar projection options

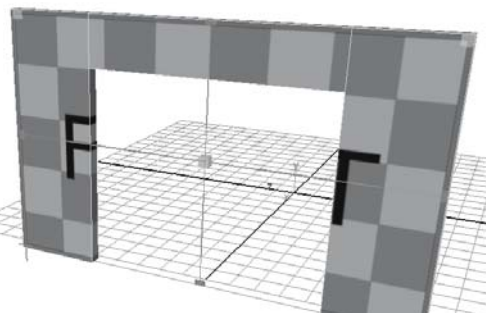


Figure 3.77: Unified texture

The same corrections should be made for the small faces of the roof and door objects. (The walls have this problem too, but those faces are hidden, so you don't necessarily need to fix them.) In these cases, we'll want to interactively translate, rotate, and scale the planar projection manipulators to get a tight fit (see Figures 3.78 - 3.81).

33. Select the *barnRoof01* object's top and bottom faces. (Note: These faces are the ones with the stretched textures on them, not the large faces we mapped at the beginning of the exercise.) Click on **Edit Polygons > Texture > Planar Mapping options...** Click **Edit > Reset Settings** and change the **Mapping Direction** to **Y axis**. As with the posts, we need a different aspect ratio than the default. Try an **Image Scale** setting of 1 (U) and 0.6 (V). Then click the **Project** button.



Figure 3.78: Projecting the bottom and top faces

34. Repeat the above step for the *barnRoof02* object.

35. In component mode, select the two diagonal end faces on the *barnRoof01* object. Then click **Edit Polygons > Texture > Planar Mapping options...** Reset the settings to default, then choose **Mapping Direction = X Axis**, and **Image Scale = 1 (U) and 0.6 (V)**. Then press the **Project** button.

36. In the **perspective** panel, look at the *polyPlanarProj* manipulator. It is a multifunction manipulator used for translating and scaling UV texture coordinates. The red and green

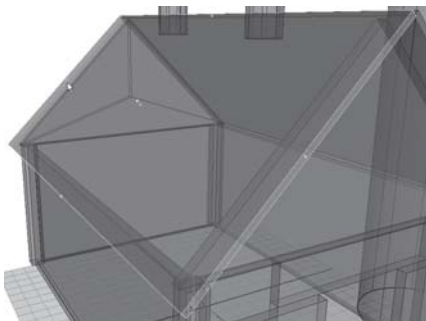


Figure 3.79: Selecting end faces



Figure 3.80: Projecting the end faces

## 128 3. Intermediate Skills

---

lines control translation. The red and green boxes control scale. Try these handles out now, and when you're done, undo the changes.

By clicking the red, cross-shaped handle in the corner, you switch the manipulator's mode to that of a multiple transform tool. This offers the same control as the default mode, plus the ability to rotate the projection. Click the light blue circle in order to invoke the familiar rotate manipulator. You will use some of these tools to rotate the texture projection on the diagonal end faces of the roof objects.

37. The texture mapping coordinates have been applied at the wrong angle for the selected faces. To correct this, click on the red cross handle in the corner of the projection manipulator. This switches the mode of the manipulator. Click once on the light blue circle to bring up the rotation rings. Drag the blue rotation ring to  $45^\circ$ . (In the **Channel Box** this reads as the **Rotate Y** channel.)

38. Now, drag the red scale box to a **Projection Width** of 13.65. This type of interactive projection placement is very helpful when you're texturing your surfaces.

39. Repeat the previous three steps for the *barnRoof02* object.

40. The door objects' top faces should be fixed next. Select the *barnDoor\_left* object. In component mode, pick the top and bottom faces. Assign a planar projection from the **Y Axis**, with an **Image Rotation** of  $90^\circ$ , and an **Image Scale** of 0.1 (U) and 1 (V) (see Figure 3.82).

41. Repeat the previous step for the *barnDoor\_right* object.

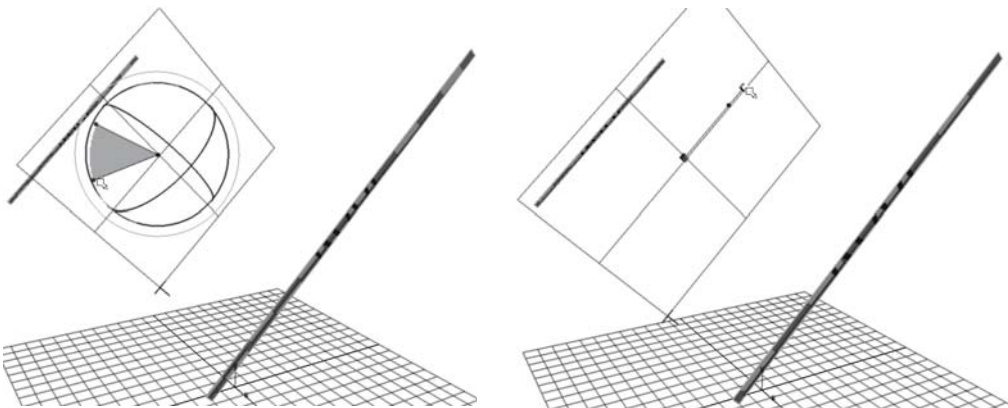


Figure 3.81: Rotating and scaling the projection manipulator



Figure 3.82: Top projection for the door

42. The final correction will be the side faces of the door objects. Pick the *barnDoor\_left* object's two side faces. Assign a planar projection from the **Z Axis**, with an **Image Scale** of 0.1 (U) and 1 (V) (see Figure 3.83).

43. Repeat the previous step for the *barnDoor\_left* object. Be sure to save your scene.

Whew, we made it! Now you can assign any file texture in place of the texture markers and know that your placement, rotation, and texel densities will be correct (see Figure 3.84). One easy way to do this is to use the marker image files as a background template when painting in Photoshop or any other image-processing application.

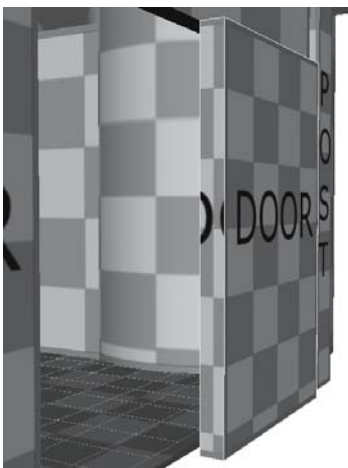


Figure 3.83: Door side fixed



Figure 3.84: Final barn with replaced textures

### 3.6 Lighting – Soft Shadows and Gobos



Realistic shadows can really improve the quality of your scenes. It can be jarring to look at an otherwise realistically lit scene that lacks the proper shadows. Depending on the light source that casts them, realistic shadows can be either hard or soft at their edges, uniform or gradual in their darkness, and even vary in color. There are many ways to adjust the properties of your shadows to achieve the look you want.

#### Soft Shadows

The shadows thrown by a light can be adjusted in the **Attribute Editor** of the light's shape node. In the **Shadows** attribute section you can make many significant changes to the way a shadow will look. For a depth map shadow, the resolution can be lowered to remove details (this helps soften harsh shadows) while simultaneously softening the edges through depth map filtering. Raytraced shadows can be softened by increasing the radius and shadow rays cast by the light.

#### Lighting Exercise – Soft Shadow

1. Set your project to `understanding_maya\Chapter03\lighting_softShadows`.
2. Open the scene `lighting_softShadow_begin.ma`.
3. The scene is of the interior of the barn and a large cylinder that has been placed inside it. In the **Render Globals**, set the resolution to 800 x 600 and the anti-aliasing quality to production level. Then render a frame from the **camera 1** view. You'll see a hard shadow cast from the cylinder onto the floor and wall (see Figure 3.85).
4. Now, select the spotlight and open the **Attribute Editor**. In the **Shadows** section, turn the **Depth Map Shadow** resolution down to **128**. Then, turn the **Filtering** up to **3**. Re-render the scene and notice how the edges of the shadow have softened (see Figure 3.86). A smaller depth map is less able to render highly detailed areas of a shadow casting shape. Filtering applies a blur to this low resolution shadow.



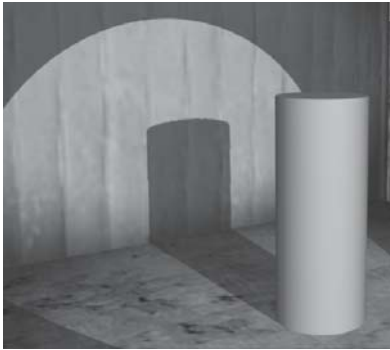


Figure 3.85: Default Dmap shadow

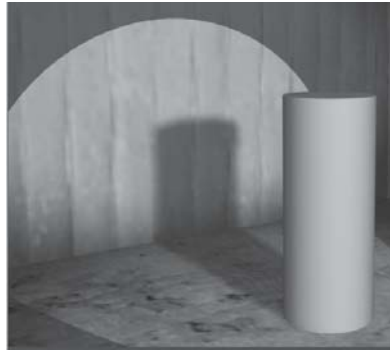


Figure 3.86: Soft Dmap shadow

5. In some cases, it may be necessary to use a raytraced shadow, for example, if you have transparent materials on your shadow casting objects. With raytraced shadows we'll use different settings to soften the shadow's edges.

6. First, in the **Render Globals**, turn on the **Raytracing** checkbox. Then, in the spotlight's attribute editor, turn on **Raytraced Shadows**. Render now and you'll see a harsh shadow (see Figure 3.87). This works well for scenes lit with bright sunlight.

7. Now let's look at softening those edges. In the **Attribute Editor**, increase the **Light Radius** to 1. Then, set the number of **Shadow Rays** to 7. Render the scene and you'll see the edges have softened (see Figure 3.88). Unfortunately, soft raytraced shadows are very time consuming to render, so you should generally try to use depth map shadows if at all possible.

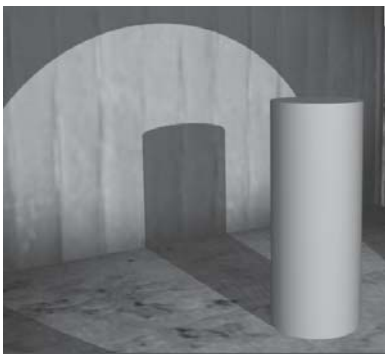


Figure 3.87: Default raytraced shadow

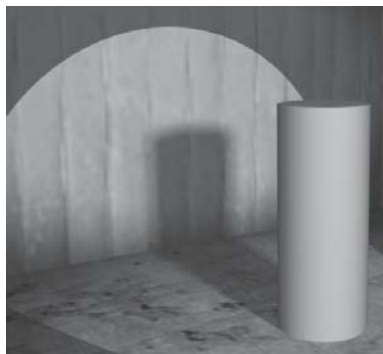


Figure 3.88: Soft raytraced shadow

## 132 3. Intermediate Skills

---

What are some cases where you would have to use a raytraced shadow? If you have a surface with a transparency matte in its material or a colored semi-transparent material such as stained glass, depth map shadows will spoil the illusion. In these cases, one solution would be to raytrace the shadows (see Figures 3.89 - 3.91).

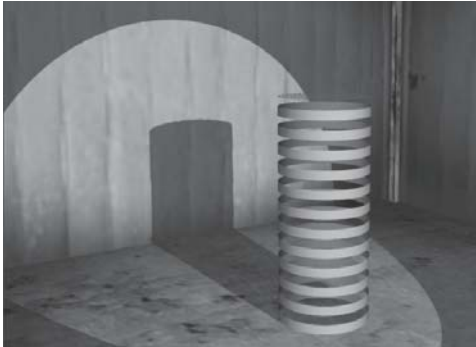


Figure 3.89: Dmap shadow

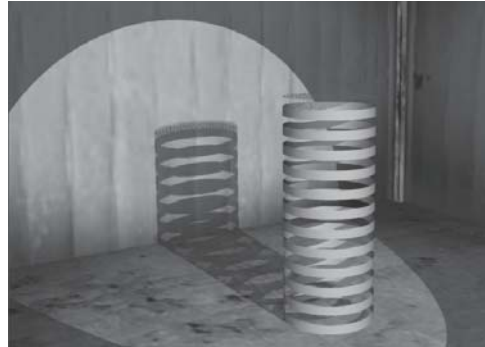


Figure 3.90: Raytraced shadow

### Gobos

When some portion of a light is occluded from hitting its target, a shadow is formed (see Figure 3.92). Often when you want a specific shadow throw pattern to appear, instead of literally placing a 3D model in the path of light, you'll instead use a gobo. A gobo is a pattern that goes between the light and the target. The use of gobos (also called “cookies”) to cast shadows in the shape of leaves, Venetian blinds, jail cell bars, overhead telephone wires, etc. is common in theater, television, and film lighting. Gobos are also employed in 3D animation to suggest the presence of objects that are not really in the scene. In any case where the camera will see a shadow (but not the object casting that shadow), it is more efficient to use a gobo than a real object.

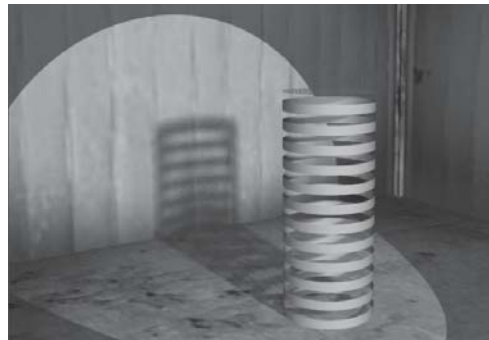


Figure 3.91: Soft raytraced shadow

You can add a gobo by mapping a texture to either the intensity or color slot of your light. This texture can be procedural, like a grid or checker pattern, or it can be an image file.

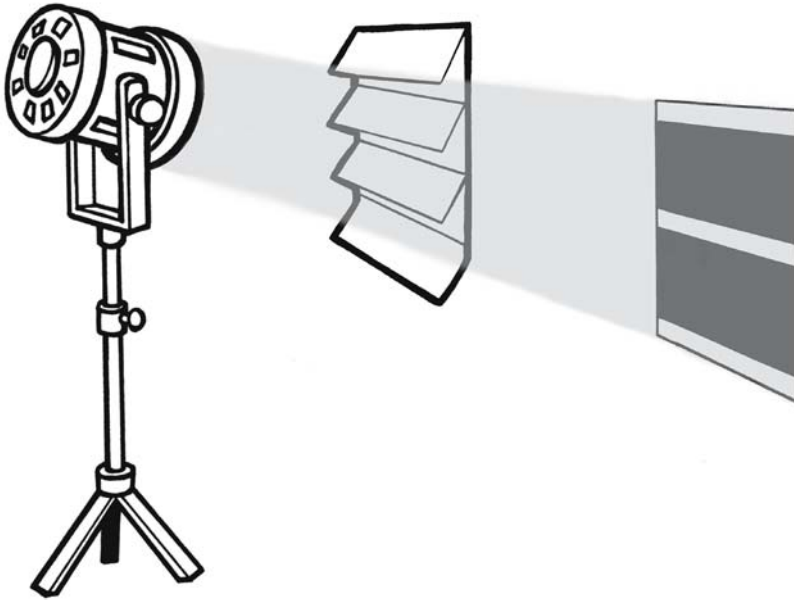


Figure 3.92: Venetian blinds in front of a light

### Lighting Exercise – Barn Gobo

1. Set your project to `understanding_maya\Chapter03\lighting_barn_gobos`.
2. Open the scene `lighting_barn_gobo_begin.ma`.
3. Activate the **Camera1** view panel and then open the **Render Globals**.
4. Set the **Resolution** preset to 640 x 480, and then set the **Anti-aliasing Quality** preset to **Intermediate**. (You may want to choose **Production** quality settings at the later stages of lighting setup, but **Intermediate** settings can be a real time saver in the beginning, allowing you to try more variations in a given time period.)
5. Close the **Render Globals** and then render a single frame through the **Camera 1** panel by clicking the **Render the current frame** button on the status line. You'll see the barn interior lit with a very obvious spotlight (see Figure 3.93). Unless there's a group of clever kids putting on a summer play in the hayloft, this looks a bit unnatural. We'll be

adding a realistic windowpane shadow to break up the spotlight. (In much the same way those kids would have if they got their hands on a **gobo** or **cookie** – a sheet of metal with a pattern cut into it used for this very purpose in theatrical lighting.)

6. The spotlight has been positioned to cast light and shadow as if it were coming through a small window in the upper part of the barn. Perhaps this is a second-level storage area, with a pitchfork and other tools leaning against the window. To suggest all of these things without building models and casting true shadows, a texture map has been painted. Using **fcheck**, open up the image file in your project's sourceimages folder named *gobo\_dirtyWindow.bmp*. This is an 8-bit, grayscale map that will block light in the black areas (see Figure 3.94).

7. Close **fcheck** and return to **Maya**. Select the *spotLight1* and open the **Attribute Editor**. Currently, the **Intensity** of the light is set to a uniform value of 1. Click on the **Create Render Node** button to the right of the **Intensity** slider. In the **Textures** tab of the **Create Render Node** window, make sure that the **Normal** radio button is checked right under the **2D Textures** dropdown arrow. Click the **File** texture button. This maps an empty **File** texture node to the **Intensity** of the spotlight.

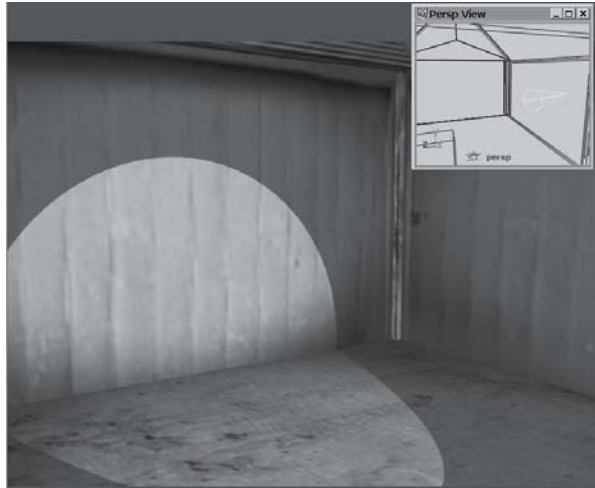


Figure 3.93: Normal spotlight throw pattern



Figure 3.94: Gobo file

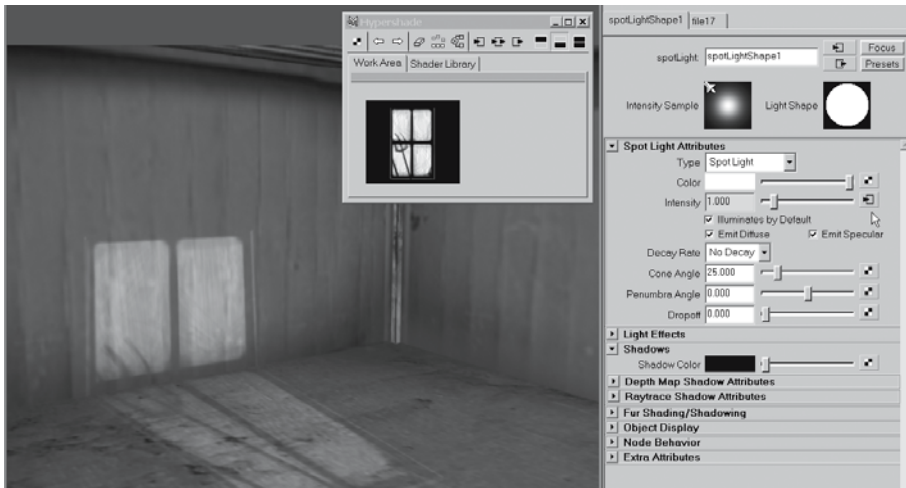


Figure 3.95: Spotlight with gobo

8. In the **Attribute Editor**, click the folder icon to the right of the **Image Name** slot. Select the *gobo\_dirtyWindow.bmp* file and then click the **Open** button. In effect, this places the gobo in front of the spotlight's beam. The gobo will now block light in the black areas, and allow it through where it is white. The grayscale image affords us 256 values from black to white.

9. Activate the **Camera 1** view panel and render a frame. You'll see that the circle of light is gone; it has been replaced by the more realistic window cutout (see Figure 3.95). (To add more tools to your shadow area, you could try painting different objects into the bitmap, and then reload the file texture.)

10. If you look closely, there are some unattractive edge artifacts along the windowpane shadows. These are caused by the **Intermediate** level **Anti-aliasing Quality** setting we chose in the **Render Globals**. You can set this preset to **Production** quality in order to soften the jagged edges. Then re-render the frame.

11. Save your scene as **lighting\_barn\_gobo\_end.ma**.

### 3.7 Rendering – Depth of Field



Typically, one of your goals in rendering will be to mimic the look of film shot through a still or motion picture camera. By default, Maya’s renderer operates a bit like a perfect camera in a vacuum – there are no flaws. This sounds great, but it’s nothing like what audiences are used to seeing. Introducing flaws and optical phenomena to the Maya camera can go a long way toward mimicking realism. Depth of field is one such technique. Notice the depth of field in Color Plate II.

In a real-world film camera, in order to get a good exposure, you may need to increase the amount of available light hitting the negative. This is done by opening up the aperture. (Note: The camera’s aperture is measured by a setting known as an f-stop; low f-stop numbers are equal to wide apertures.) As this aperture is increased, the camera’s ability to keep a deep focus is decreased. So, while you may have focused on a subject in the middle distance, the background and foreground will go out of focus. This effect is called “depth of field.”

Depth of field can be used to great effect. It makes your 3D scenes look more like they were photographed with a film camera. Also, you can use a change in depth of field to focus the audience’s attention on a different subject without even moving your camera.

#### Rendering Exercise – Depth of Field

1. Set your project to `understanding_maya\Chapter03\rendering_dof`.
2. Open the scene `rendering_dof_begin.ma`.
3. In the scene you will see the airplane sitting in the barn. In the foreground is a sphere and in the background, a cylinder. Activate the **Camera1** view panel (the active panel has a thicker border around it). Now click the **Render current frame** button on the shelf.
4. In the **Render View** window you’ll see an in-focus render of the plane as in Figure 3.96. Such an infinite focus is definitely possible in the real world if there’s lots of available light, but it sure isn’t very dramatic, is it?
5. Let’s look at some of the measurements in the scene (see Figure 3.97). If you look at the **Perspective 1** view, you’ll see that some distance measurement tools have been created between the camera and the three subjects. (This is found under **Create > Measure Tools**)



Figure 3.96: Render without depth of field

> **Distance Tool.**)

6. Select the *cameraShape1* object. In the **Attribute Editor**, turn on the **Depth of Field** checkbox.

7. Next, set the camera's aperture to a small number, such as an **F Stop** of 4. Remember, a small aperture equals a narrow depth of field. (Unlike a real film camera, Maya's cameras won't actually adjust the amount of incoming light when the f-stop is changed, so don't worry that you'll change the apparent brightness of your scene.)

8. The **Focus Distance** can be set to equal the camera's distance from the *pursuitPlane*, which the distance tool reports as about 12.7 units. This will produce a depth of field effect where only the plane is in focus. Render your scene (see Figure 3.98 and Color Plate X).

9. Try setting the **Focus Distance** to 10.6 for an in-focus

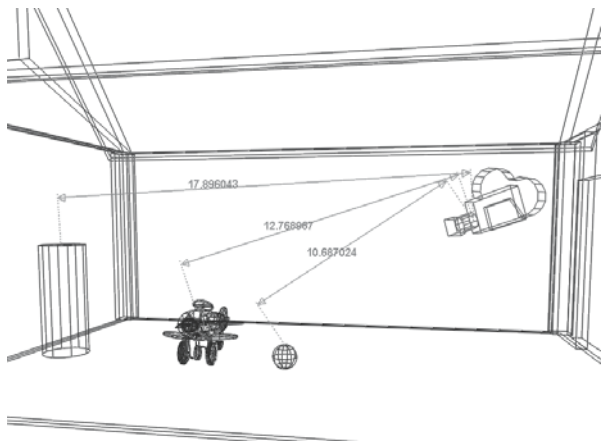


Figure 3.97: Measuring distance from camera

## 138 3. Intermediate Skills

---



Figure 3.98: Focus distance of 12.7 units

sphere as in Figure 3.99, and then try 17.9 for the in-focus cylinder as in Figure 3.100.

10. You may also want to experiment with different **F Stop** values from 1 to 64, in order to see the change in focal depth. This value can also be animated for a “rack focus” effect. This technique is often used to guide the viewer’s eye from one subject to another without moving the camera.



Figure 3.99: Focus distance of 10.6 units



Figure 3.100: Focus distance of 17.9 units



### 3.8 Project 3 – Curved Half Pipe

In Chapter 3's assignment you will make the egg roll down a curved half pipe. Add character to the egg by making it teeter on the brink of rolling at first, using the Graph Editor.

- Use the Graph Editor to shape the ease-in and ease-out.
- Create realistic shadows.
- Use a file texture in your shader.
- Shoot the scene through a depth of field camera.

---

# Chapter 4

## Adding Character



### 4.1 Review – Graph Editor

The **Graph Editor** is the visual representation of the animation values and timing of your scene. The in-betweens are represented by the curves flowing between the points (keyframes). In other words, the way an object gets from point A to point B to point C is determined by the tangents in the **Graph Editor**. To use it effectively you must be familiar with the different tangent types shown in Figure 4.1.

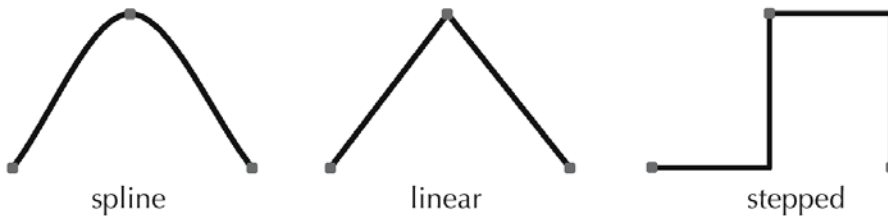
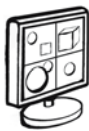


Figure 4.1: Tangency types

- Spline — this will give you soft, organic types of motion.
- Linear — creates very stiff, mechanical motion.
- Stepped — there are no in-betweens; objects will pop from one key to the next.

Note that these types can be mixed by breaking tangency and then using different in-tangents and out-tangents on a single keyframe.

### 4.2 Workspace – Efficient Workflow



Maya's user interface is much more than meets the eye at first. It's clear that you can run commands and select tools through the traditional menus, but you can also access these things from a series of keyboard shortcuts and special radial menus. The buttons at the top of the interface can be rearranged, deleted, or created from scratch. And, parts of the interface can be toggled on and off to suit your needs.

## Marking Menus

Most software applications have built-in keyboard shortcuts (**Hotkeys**) to make your work easier and quicker. For example, in any word-processing program, why move your cursor up to **Edit > Copy** when you can just hit **ctrl-c** on the keyboard? Maya takes this usability concept and adds to it. Anywhere that your cursor is on the screen you have access to a type of radial menu called a **Marking Menu**.

Try holding **h + LMB**. This brings up the marking menu for switching menu sets. Drag the cursor over the menu set to which you want to switch (see Figure 4.2).

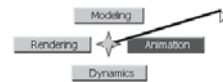


Figure 4.2: Marking menu

## Hotbox

The mother of all marking menus is the **Hotbox**, which can be invoked by pressing and holding the spacebar (see Figure 4.3).

The **Hotbox** brings in-depth menus to your work area. By default, it contains the Common Menu, Pane Specific Menu, Menu Set Menu, Recent Commands, and Hotbox Controls. It is also possible to draw strokes in the cardinal directions once you have held the spacebar. This pops up four more marking menus for quick interface customization and pick mask selection.

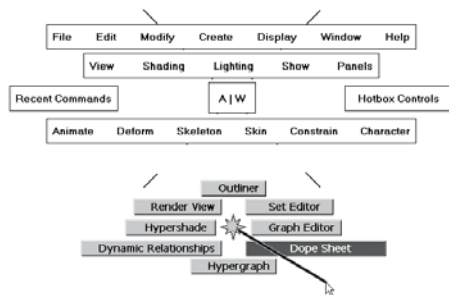


Figure 4.3: Hotbox

## Interface Elements

The entire Maya interface is customizable. You can create a workspace that suits your needs by turning on and off elements of the display in the **Display > UI Elements** menu. For example, you may not need the time slider and range slider when you are modeling - turn them off to reduce clutter and increase your modeling area.

### Shelves

An even greater amount of customization is possible if you turn commonly used commands and tools into clickable buttons on a custom **Shelf** tab (see Figure 4.4). To do this, hold down the **ctrl + alt + shift** keys while you select a normal menu item. (In Linux, click **alt + shift**.) A new MEL button is created that performs the identical action to the menu item you had picked.



Figure 4.4: A custom shelf

### 4.3 Modeling – Box Modeling with Polygons and Sub-d Surfaces



Creating good-looking, highly detailed models can be a very difficult task. If the model you're building is mechanical and hard edged, you will usually create it with a collection of primitive shapes or extruded curves. This is similar to the approach taken in CAD software. Organic shapes – ones with curves, wrinkles, swoops, undercuts, branching surfaces, etc. – often require a different approach. One such approach is the CGI equivalent to stone sculpture, known as “volume modeling,” “box modeling,” or “derived surface modeling.”

In box modeling, a rough, low-geometry shape is formed first. Later it is smoothed into a much rounder, high-geometry shape (see Figure 4.5). The low-geometry model acts as a control mesh, or cage, for the high-geometry, derived surface. One benefit of this method (over starting out with high-geometric density) is the relative ease with which you can edit the sparse geometry of your control mesh.

In fact, if you have ever reshaped a NURBS sphere by moving its CVs, you have already used a form of control mesh/derived surface modeling. The Control Vertices of a NURBS surface define the placement of the high resolution surfaces underneath. Try reshaping a NURBS sphere vs. a high-

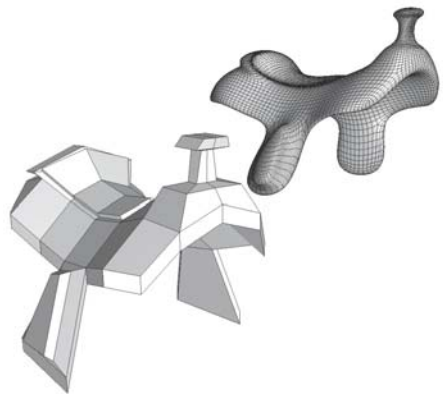


Figure 4.5: Derived surface model

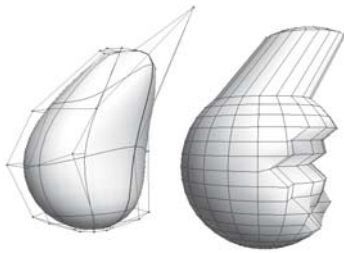


Figure 4.6: NURBS vs. polys

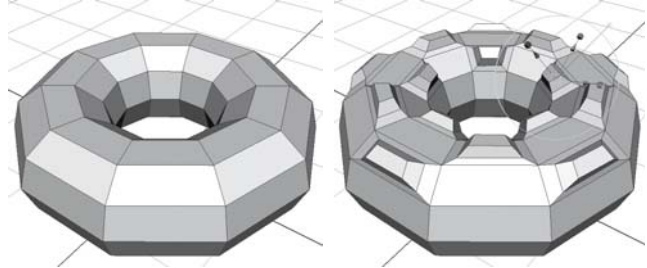


Figure 4.7: Adding detail to a polygonal model

density polygon sphere (see Figure 4.6). There's no contest: the NURBS CVs are easy to edit, while the polygonal vertices are too numerous to be edited efficiently. If, however, you could reshape a very few points that controlled the very many vertices on the polygonal sphere, things would be much easier. This is a key benefit of box modeling.

OK, so now you're saying to yourself, "If this box modeling is here just to make my polygons act like NURBS, why not just use NURBS in the first place?" Good point! However, there are some frequently encountered situations where you will want to use polygons. On a polygonal object, it's easy to add detail to the surface, like adding clay to a sculpture (see Figures 4.7 and 4.8). If you want to branch your surface out (think fingers on a hand, or nostrils in a nose) polygonal objects are usually simpler to control than NURBS ones.

Starting with a polygonal primitive, such as a cube or sphere, you can extrude and shape faces much like roughing out a hunk of marble with a stone and chisel. Using the **Edit Polygon > Extrude Face** command, the **Edit Polygon > Split Polygon Tool**, along with the **Move**,

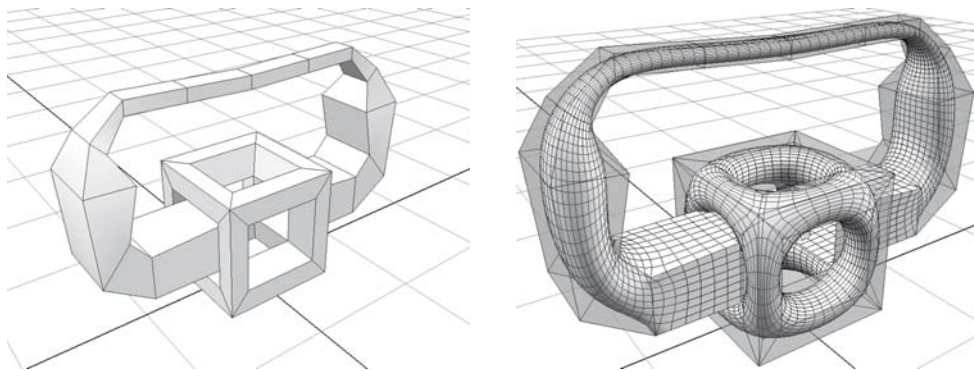


Figure 4.8: Low polygon control mesh and its smoothed, derived polygonal surface

## 146 4. Adding Character

**Rotate**, and **Scale Tools**, it is possible to quickly rough out a basic shape. These low-polygon models can be refined later with the **Polygon>Smooth** command or act as the basis for a Subdivision surface.

When working on this type of model, it's a good idea to make frequent use of the **RMB** component marking menu (see Figure 4.9). This contextual menu allows you to quickly switch between the major component pick masks of a polygonal object: Vertex, Face, and Edge, as well as return to Object mode.

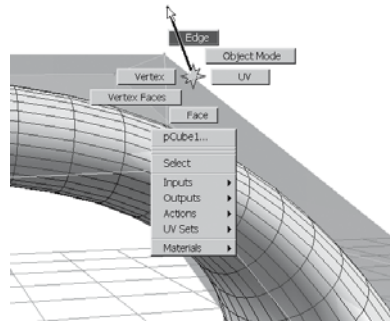


Figure 4.9: Component marking menu

### Modeling Exercise – Box Modeling Saddle

1. Make sure that **Construction History** is turned on.
2. Create a polygonal cube with the following options: **Width = 2, Height = 0.1, Depth = 1, Subdivisions Along Width = 6, Subdivisions Along Height = 1, and Subdivisions Along Depth = 4**. Rename the cube *saddle*.
3. **RMB**-click on the cube to invoke the pick mask marking menu. Select **Vertex**.
4. From the **top**, **front**, and **side** view panels, use the **Move** and **Scale Tools** to reshape the cube into the general form of a horse saddle (see Figure 4.10).

5. Next, we'll begin to extrude faces of the *saddle* in order to form the horn, leg flaps, and seat of the saddle. There is one setting we need to check before extruding. The **Polygons > Tool Options > Keep Faces Together** option must have a checkmark next to it.

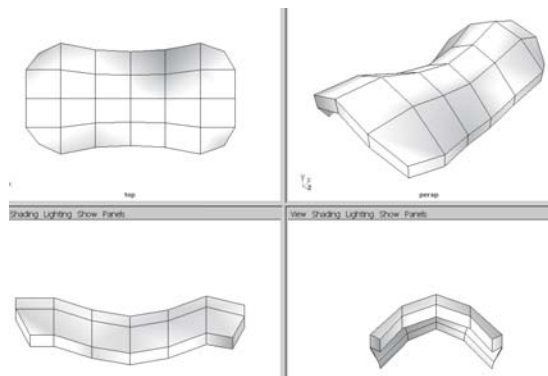


Figure 4.10: Roughed out form

This ensures that neighboring faces will still share an edge after the extrusion. In other words, if you were to select a row of six faces running down the length of the saddle and extrude them with **Keep Faces Together** turned on, you would create a ridge. With **Keep Faces Together** turned off, you'd create six separate humps like the example in Figure 4.11.

6. **RMB**-click on the *saddle* and select the **Face** pick mask. Select the two faces that are at the top-front area (the two farthest left faces in the **top** view).

7. In the **Modeling Menu Set**, click **Edit Polygons > Extrude Face**. The moment you select this command, there are six new faces created, one on each outer edge of the selected faces. The **Manipulator Tool** you see can now be used to translate, rotate, and scale the selected faces. Use the blue Translate Z arrow to lift the faces up a bit. Then use the red and green Scale X and Scale Y boxes to shrink the selected faces a small amount, as in Figure 4.12.

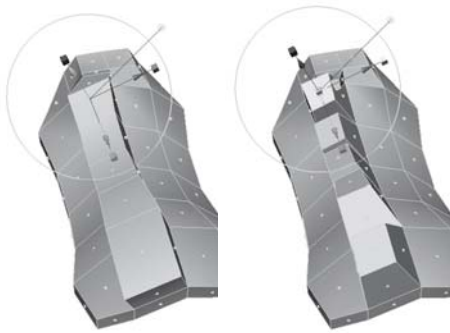


Figure 4.11: Keep Faces Together on vs. off

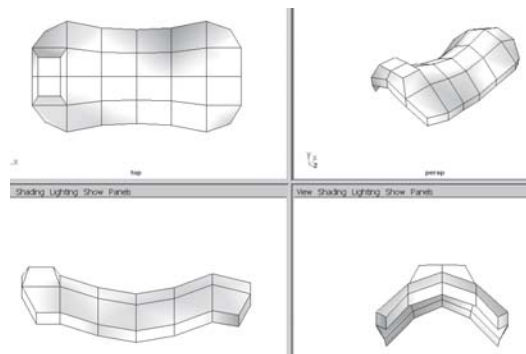


Figure 4.12: First face extrusion

Notice how the **Manipulator Tool** axes are oriented to the local values of the selected faces, not the usual world alignment. This is helpful when extruding faces away from, or in toward, the object's center.

8. Next, with the two faces still selected from the last step, repeat the **Extrude Face** command by pressing the **g** key on your keyboard. Again, this extrudes the faces together. Use the **Manipulator Tool**'s translate and scale handles to shape the horn base as in Figure 4.13.

9. Repeat the **Extrude Face** command, this time lifting the horn shape up a good amount. This will be the shaft of the horn as in Figure 4.14.



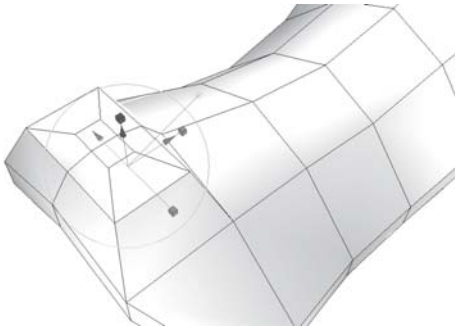


Figure 4.13: Horn base

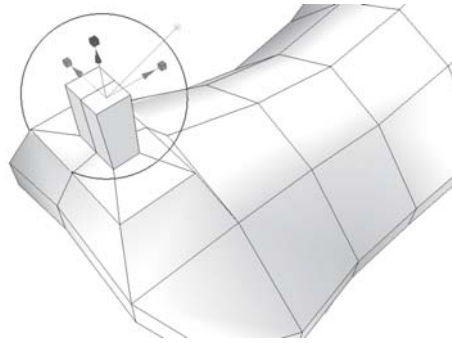


Figure 4.14: Horn shaft

10. The next extrusion will be translated only a small amount on the local Z-axis. A large amount of X and Y scaling will be used to define the knob on top of the horn (see Figure 4.15).

11. Extrude one more time to define the top of the knob itself as in Figure 4.16.

When extruding faces, it is often possible to build up the model symmetrically. By selecting faces on opposite sides of the mode and then extruding them simultaneously, you can take advantage of the equal-but-opposite nature of the local Z-axis. However, some manipulations will not provide the results you desire. If, for example, moving two such faces on the Y-axis causes one to go up while the other goes down, you may need to switch the **Manipulator Tool's** mode from **Local** to **Global**. This is done by clicking the **Toggle Manips** handle that looks like a bull's eye coming off the tool at a 45° angle.

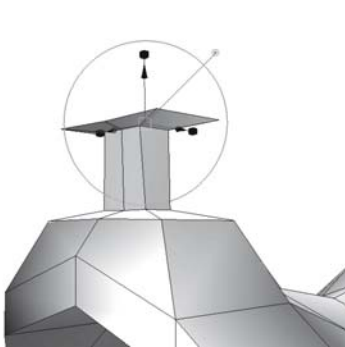


Figure 4.15: Knob base

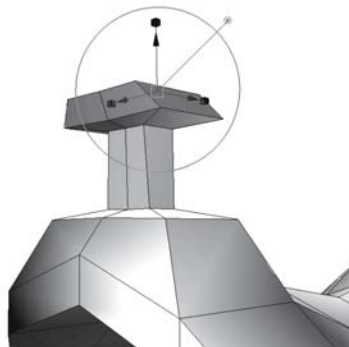


Figure 4.16: Knob top

12. Select the face that lies at the middle of the *saddle's* length, along the edge. Then navigate to the opposite side and shift-select the equivalent face on the other side (see Figure 4.17). These faces will be extruded to become leg flaps.

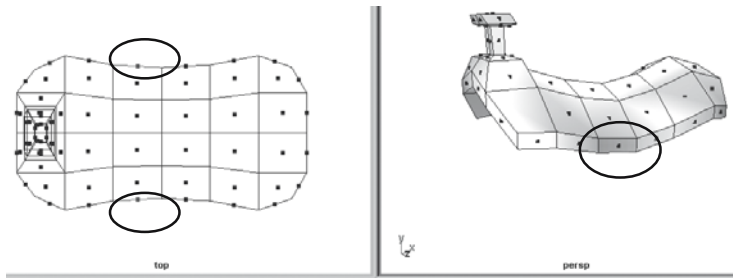


Figure 4.17: Leg flap face selection

13. Use the **Extrude Face** command, and then pull the faces outward on the Z-axis. Next, toggle the manipulator to **Global** mode. Translate the faces downward on the Y-axis to form leg straps. Scale out the faces on the X-axis (see Figures 4.18 and 4.19).

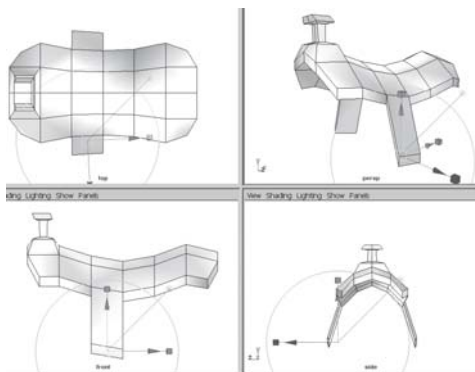


Figure 4.18: Extruded flaps

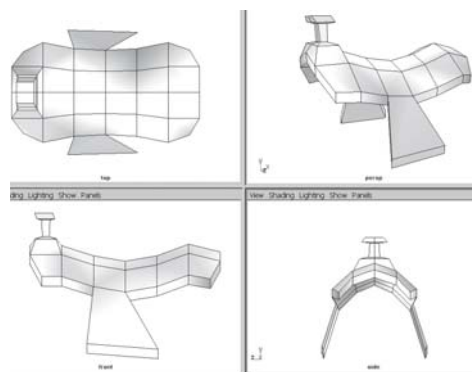


Figure 4.19: Scaled flaps

The **Extrude Face** command is very useful for defining surface details, such as padding, buttons, muscles, bumps, and lumps. By extruding a face, scaling it, extruding again with only a small amount of translation, and then scaling a final time, you can create complex surface shapes with undercuts, bevel, intakes, and radius edges.

## 150 4. Adding Character

All of these shapes have been created on the easily edited box model; smoothing the surface later will provide the attractive curvature we want. As you practice building box model extrusions, it is a good idea to frequently smooth the model to see the effects of your editing using the **Polygons > Smooth** command (see Figure 4.20). Undo the **Smooth** operation when you're done looking at it.

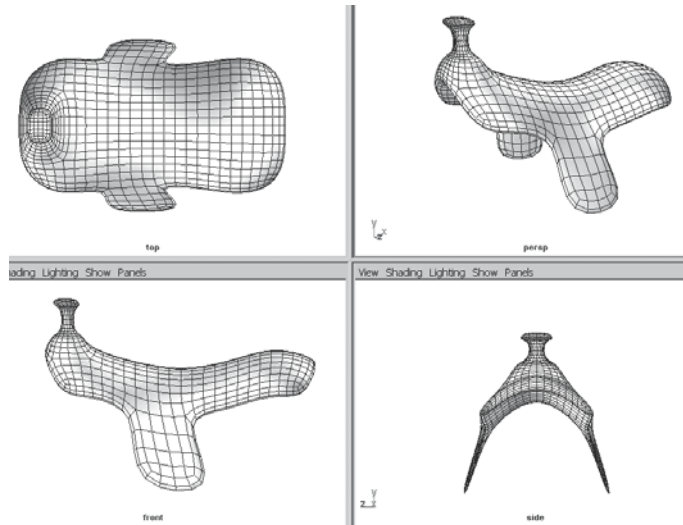


Figure 4.20: Smoothed saddle

14. Select the large face at the center of the leg strap, as well as its opposite equivalent on the right side of the *saddle*. Extrude the faces and then scale them a bit smaller. Only translate on the Z-axis a small amount (see Figure 4.21).

15. Repeat the above procedure to raise the level of the padding.

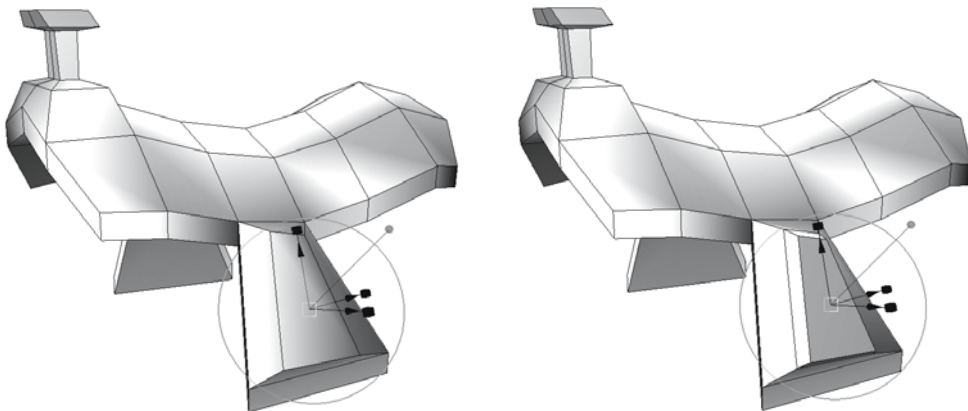


Figure 4.21: Extrude and scale

16. Select the two adjacent faces on top of the *saddle* that are second from the end. This is where you will create a raised section for the seat. Extrude the faces together and then lift them up from the surface, move them back on the X-axis and scale them smaller on the X-axis (see Figure 4.22).

17. The final two extrusions will be used to make extra seat padding in the *saddle*. Select the six faces from the center to the seat cushion. Extrude these faces together. Now scale them smaller on the X- and Y-axes, and translate them down into the *saddle* on the Z-axis.

18. As in Figure 4.23, extrude the selection and translate up to form the padding.

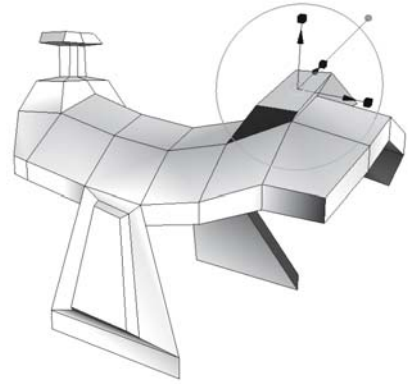


Figure 4.22: Building the seat

You have just finished modeling the rough control mesh for your *saddle*. You can make any adjustments at this point by moving, rotating, and scaling the vertices, edges, and faces of the model. The final step is to smooth the surface into a more organic, rounded shape. Depending on your needs, this derived surface can be generated through the **Polygons > Smooth** command, or by converting the model to a **Subdivision Surface** in the **Modify > Convert** menu.

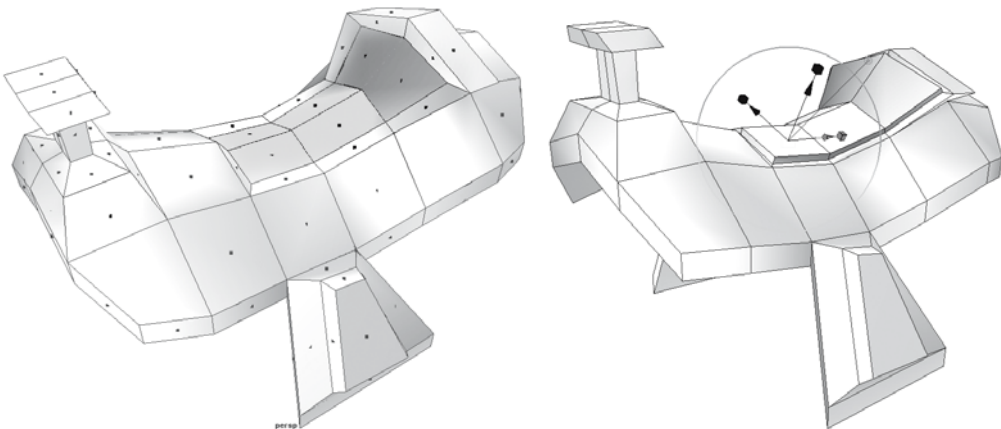


Figure 4.23: Selection and extrusion

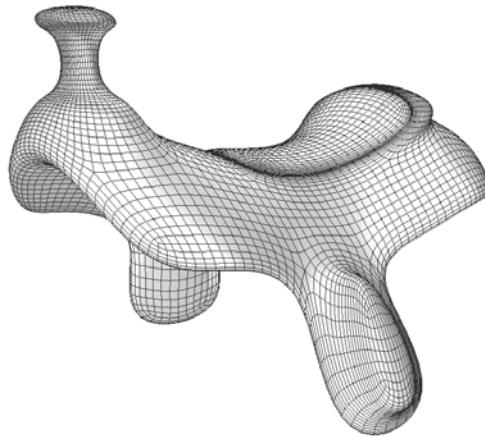
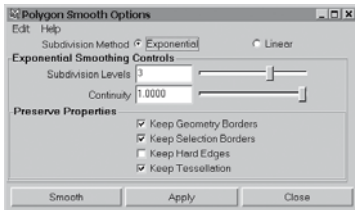


Figure 4.24: Polygon smooth options    Figure 4.25: The smoothed saddle

19. Select the *saddle*, and then click **Polygons > Smooth options**. **Subdivision Levels** determines how smooth the surface will be. A range of 1-3 is usually enough; try setting this to 3, and then click the **Smooth** button (see Figure 4.24).

20. Your *saddle* is now a nice, organic shape, defined by the original box model you built (see Figure 4.25). Save your scene – you can use it as a prop in the barn of your chicken animation.

The model you just created can be used to further explore Maya’s node-based architecture – the **dependency graph**. Since **construction history** was on when you began, a series of connected nodes was created as you built the model. For simplicity, the object shows up in the **Outliner** as simply: *saddle*. This is only the tip of the iceberg, however. If you were to select *saddle* in the **Hypergraph** and show its input connections, you would see that the *saddle* is the transform node, which is the parent of the *saddleShape* node, which in turn is dependent on each edit you made to the model going all the way back to the original *polyCube1* node.

Maya considers the object somewhat chronologically when deciding what to display on-screen. In Maya’s mind (OK, you’re right, I’ve got to stop thinking of Maya as a cognitive being) the *saddleShape* is really formed by creating a polygonal cube with 6 x 1 x 4 subdivisions, tweaking it into a rough saddle shape, extruding some faces eleven different times, and then smoothing it with three subdivision levels. This series of dependent sequential nodes can be modified to edit the final shape. With the end result of the smoothed *saddle* on-screen, it’s possible to alter one of the upstream *polyExtrudeFace* nodes.



Figure 4.26: Dependency graph of the saddle

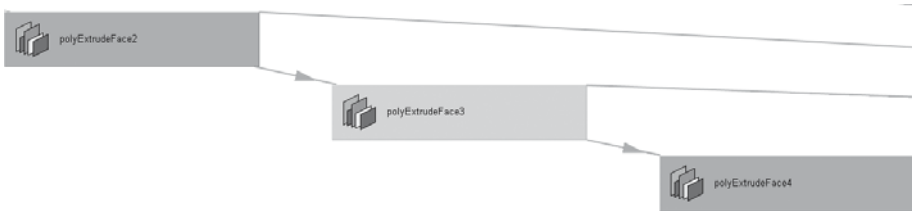


Figure 4.27: polyExtrudeFace3 node selected

21. Select the *saddle* object. Open the **Hypergraph**, and then click **Graph > Input Connections**.

The scary-looking flowcharts you see in Figures 4.26 and 4.27 represent the steps you took while modeling the *saddle*. On the far left is the *polyCube1* node. Along the way are the many *polyExtrudeFace* nodes that were created each time you used the **Extrude Face** command. (Maya always evaluates this **dependency graph** from left to right). These nodes also contain the exact values of offset and scaling you chose for each extrusion. You can change those values now in order to effect the final model.

22. Select the node called *polyExtrudeFace3*. (Remember that you can use the standard camera navigation tools to track and dolly in the **Hypergraph**.) In the **perspective** view panel you will see a **Manipulator Tool** (press **t** on the keyboard if it doesn't show up automatically). Try scaling and translating the manipula-

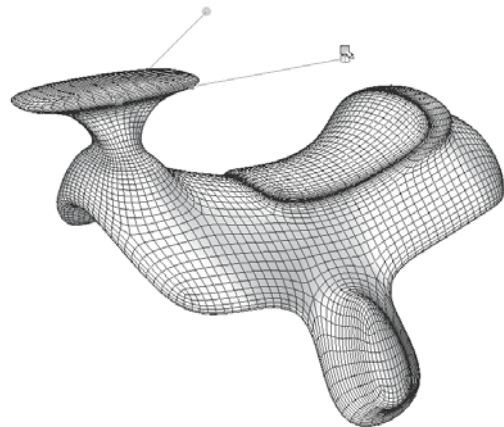


Figure 4.28: Altering polyExtrudeFace3

## 154 4. Adding Character

---

tor handles. Notice how the model responds as if you had originally used these extrusion values (see Figure 4.28).

23. It is also possible to see what the model would have looked like if you had never performed a particular action. Select the *polyTweak1* node. This node contains the values of the vertex editing you did at the beginning of your modeling. Open the **Attribute Editor**. In the **Node Behavior** section, change the **Node State** from **Normal** to **HasNoEffect** (see Figure 4.29). Notice how the *saddle* has flattened out, since you have in effect removed those tweaks from the dependency graph (see Figure 4.30). When you are done, return the **Node State** to **Normal**.

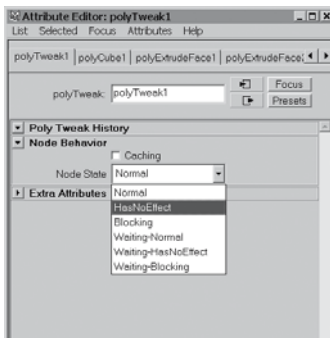


Figure 4.29: Node has no effect

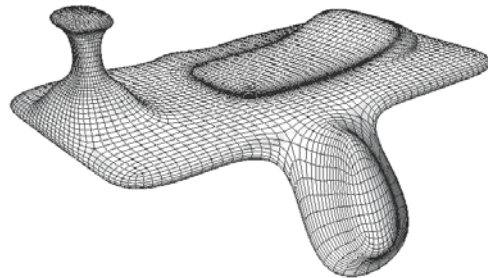


Figure 4.30: Saddle without tweaks

### Modeling Exercise – Smooth Polygon Chicken

Smooth polygon and subdivision surface models are often created by first building a low-polygon model, and then later doing a conversion. In this exercise, you'll use some of the most common polygon modeling methods to sculpt the rough form. The conversion to a smooth polygon surface will allow for easier deformations and skinning later.

1. For this exercise, you'll open up a modeling template. Set your project to `understanding_maya\Chapter04\modeling_derived_chicken`.
2. Open the scene `modeling_chicken01.ma`.
3. In the **front** and **side** views you'll see image planes that have reference drawings on them as in Figure 4.31 and Color Plate IV. Use these to align your model as you work.

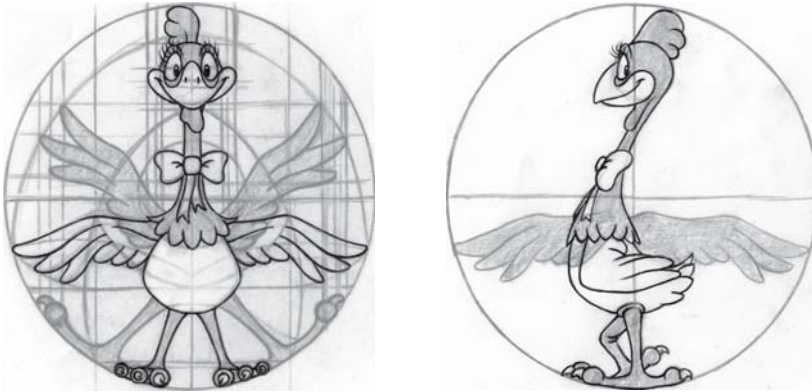


Figure 4.31: Reference drawings

4. You'll begin with a simple polygon cube. Then, using only two tools, you can rough out the form of the entire model. These are:

a. **Edit Polygons > Extrude Face** and

b. **Edit Polygons > Split Polygon Tool**.

5. In general, it's a good idea to extrude one level of detail, and then adjust the vertices before doing another extrusion. As you're working, you can turn on the visibility for the skinLayer. This layer contains the finished model in referenced wireframe format. If necessary, you may snap the vertices of your model to the equivalent points on the reference model.

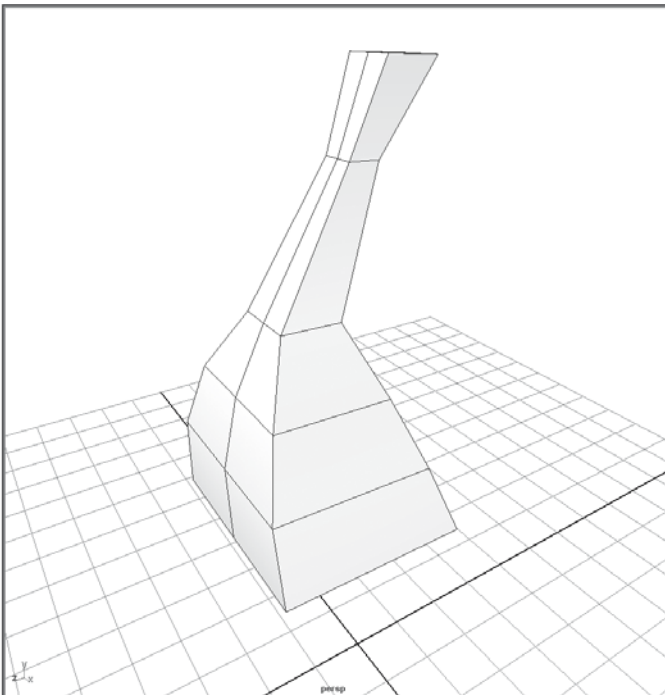
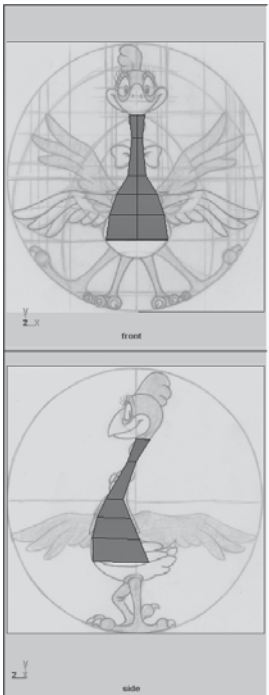
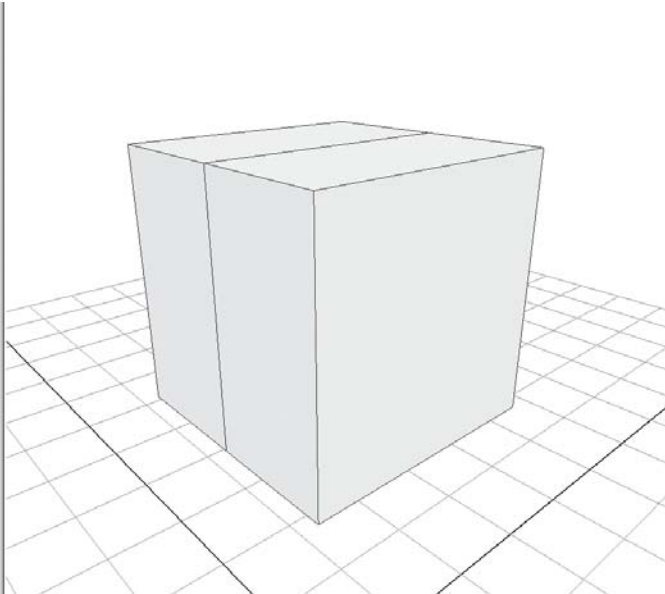
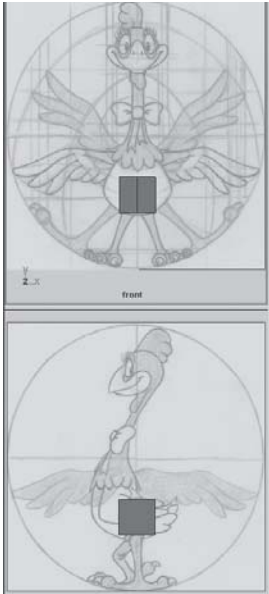
6. The **Split Polygon Tool** is useful when you need to subdivide an edge to add detail. It is good form to use only three- and four-sided polygons in your models, especially when converting to subdivision surfaces later.

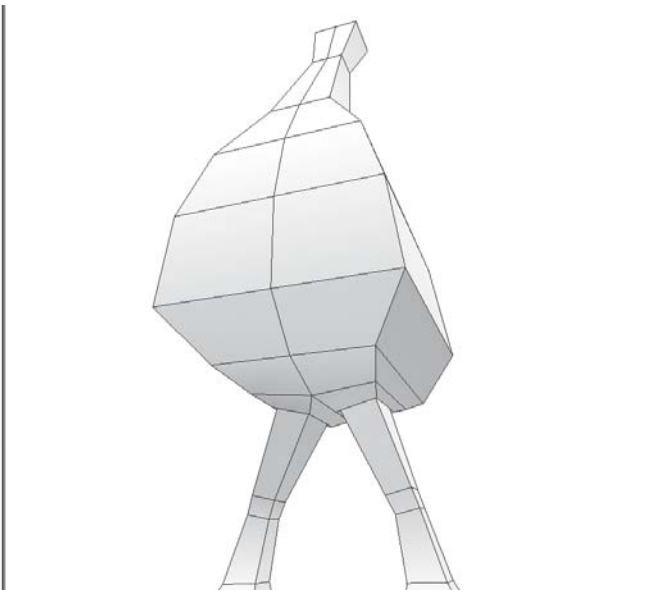
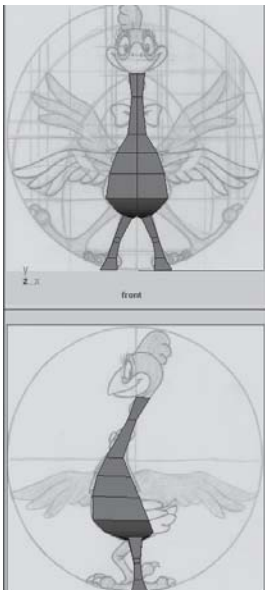
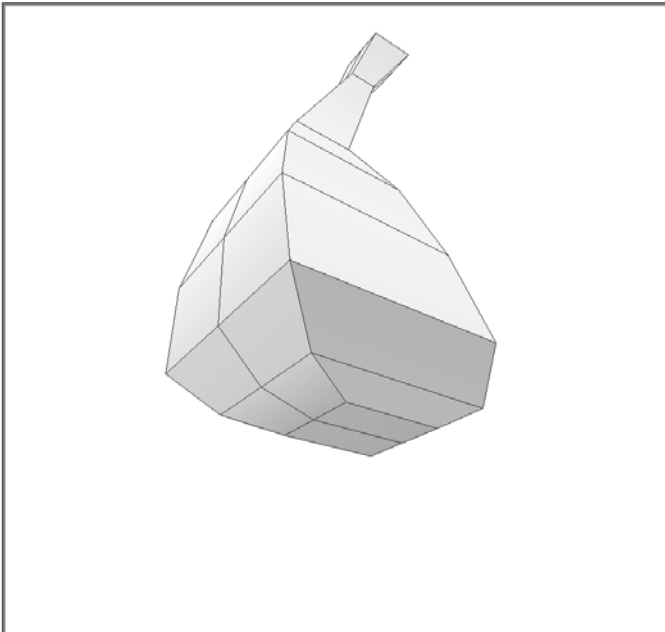
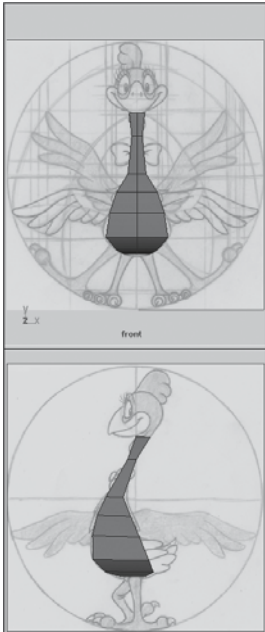
7. Another useful modeling technique is to create only half of your model, and then mirror your geometry at the end of the process with the **Polygons > Mirror Geometry** command. It's fine to begin working symmetrically, but by the time you reach an area of moderate detail you should focus on only a single side of the character.

Take a look at the figures on pages 156 - 161, and use them as a guide for your own model. You can also check Color Plate V.

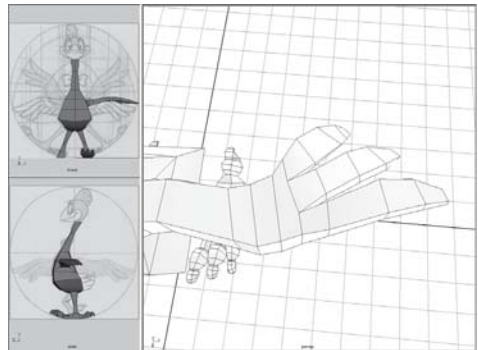
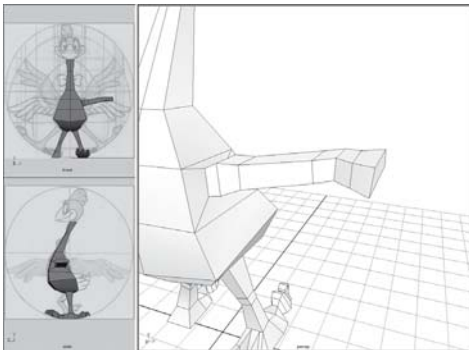
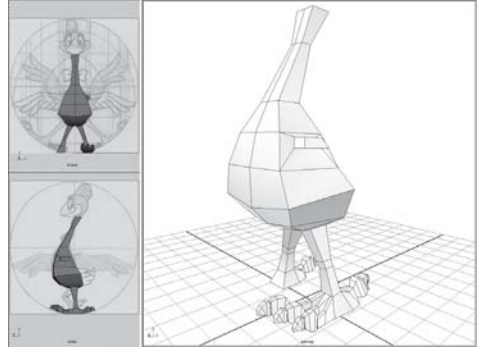
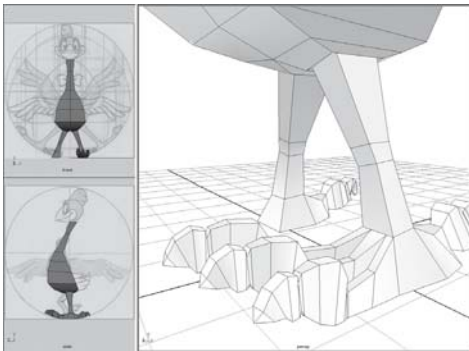
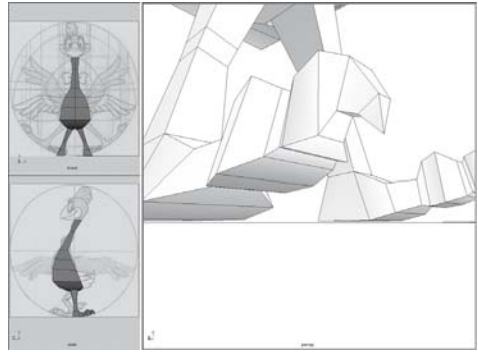
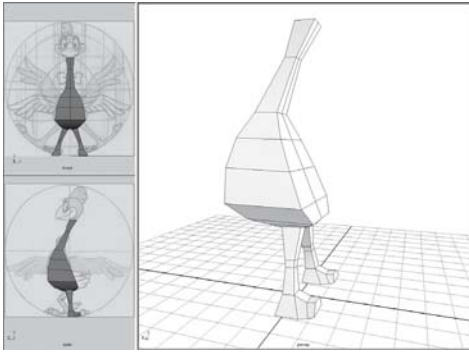


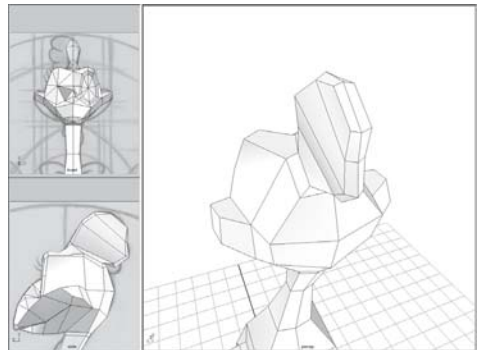
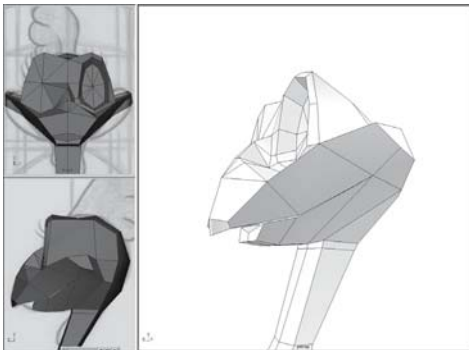
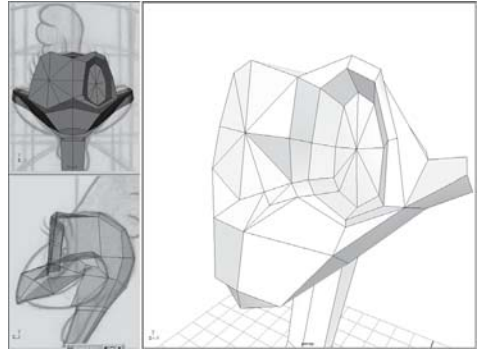
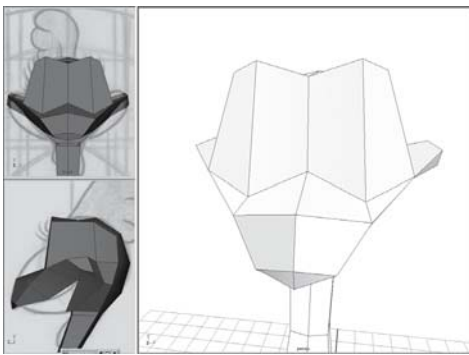
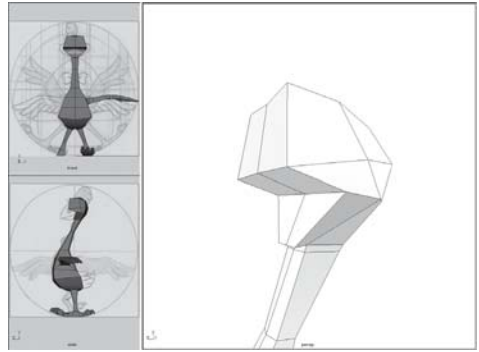
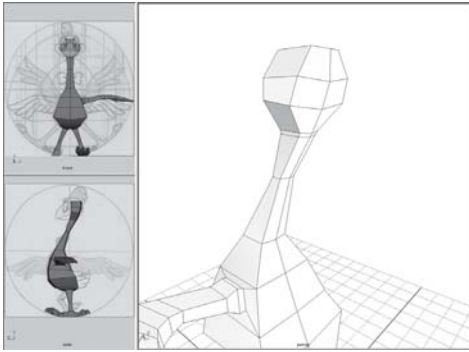
# 156 4. Adding Character





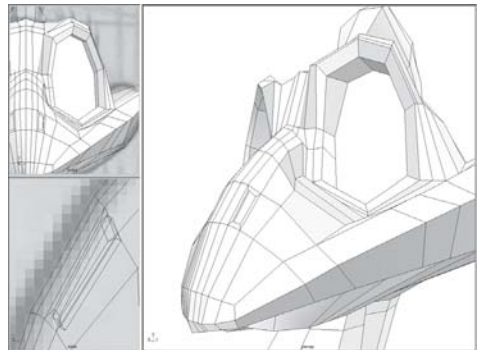
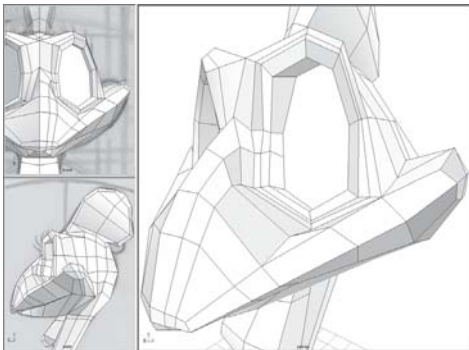
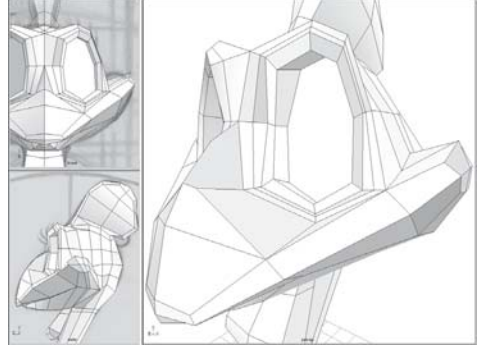
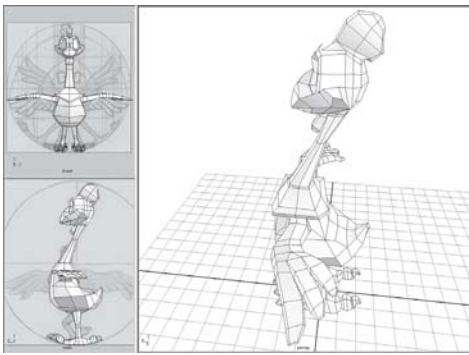
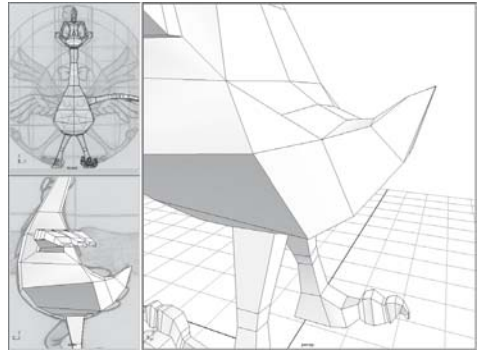
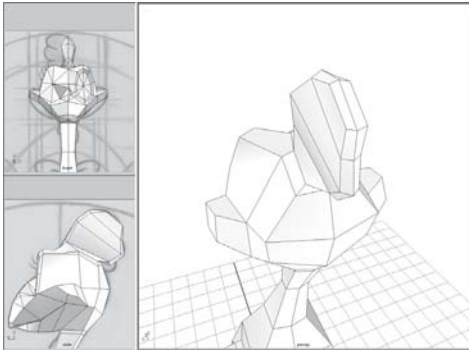
# 158 4. Adding Character

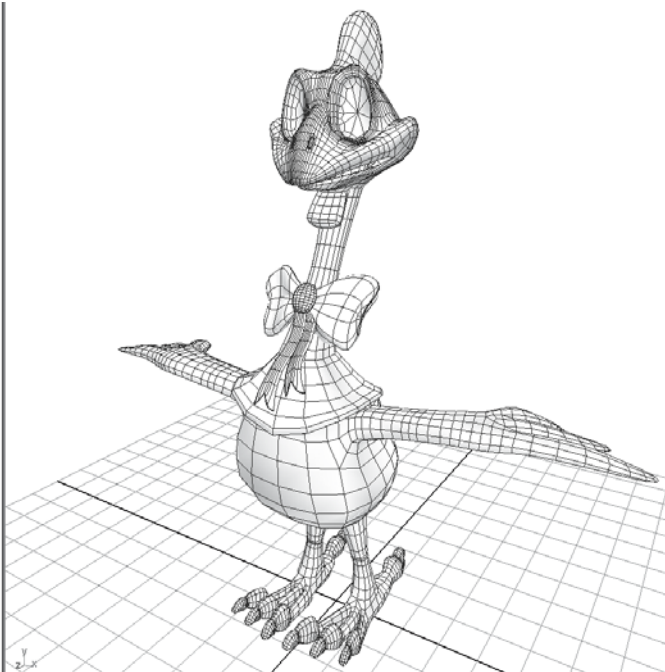
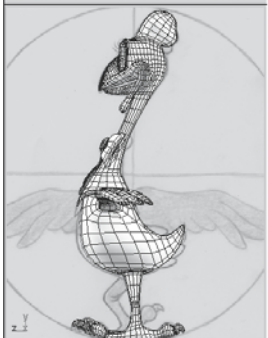
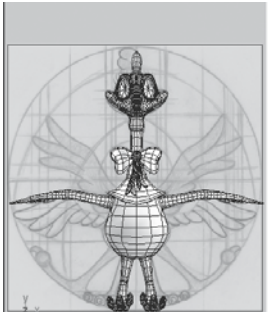
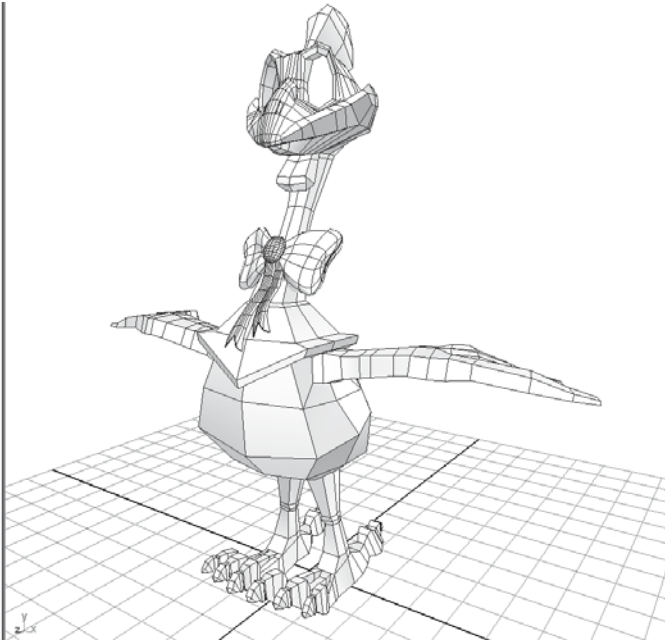
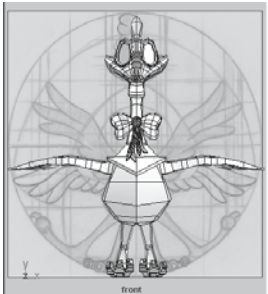




# 160 4. Adding Character

---





### 4.4 Animation – Hierarchies and Groups



To make the job of animating complex models and motions easier, you can link objects together and bundle them into groups. Links are created hierarchically, which means that one object will be higher up on the list of influence than another. This top-level object is called a parent; under it are its children. Objects that are bundled together are called Groups and work similarly to Parent/Child hierarchies.

In these examples the Hypergraph is used to show the relationship between linked objects. In Figure 4.32, example **a.** there are two normal, unrelated objects. Example **b.** has the cone as parent. Example **c.** has the torus as parent. And, example **d.** has the two objects grouped together.

When a parent is moved, its children come with it. When a child is moved, the parent is unchanged. An example of when you would use a hierarchy is an abacus and its beads. If the whole abacus is being moved, the beads must move with it. When the beads slide around, the abacus sits still. So the abacus is the parent and the beads are the children.

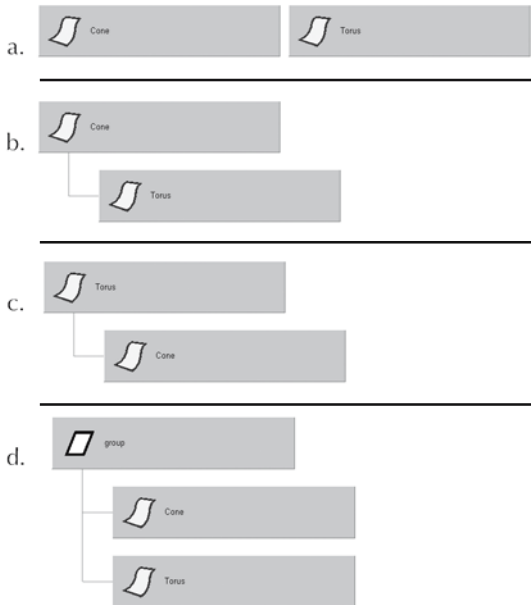


Figure 4.32: Hierarchical relationships



Hierarchies can be used in many situations, but they often fit into one of two categories: multiple-part animations or multiple-motion animation. Multi-part is a collection of objects, such as a car body, wheels, steering wheel, doors, and trunk (see Figure 4.33). Multi-motion can be a single object, such as a flying saucer, that's animated on one axis on multiple levels, such as oscillating up and down, while flying up and down. This secondary animation (the oscillation) can be difficult to achieve on a single transform node, so a hierarchy is established with multiple transform nodes.

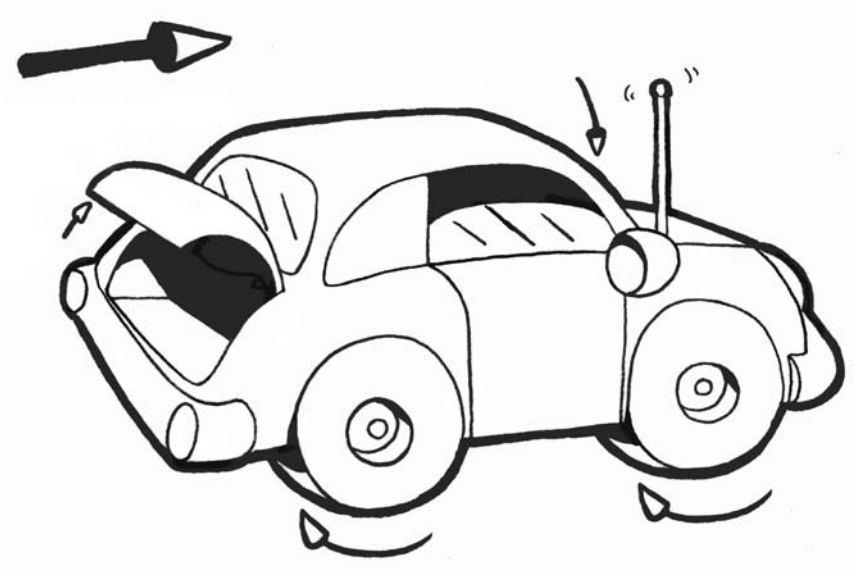


Figure 4.33: Multi-part hierarchical animation

### Animation Exercise – Little Red Wagon

1. Set your project to `understanding_maya\Chapter04\animation_grouping`.
2. Open the scene `animation_grouping_wagon_begin.ma`.

You will see a red wagon that is composed of a body, four wheels, two axles, a steering hitch, and a handle. These different parts must remain together during animation; it would be nearly impossible to keep track of everything if you had to move and keyframe all the individual parts (see Figure 4.34). What we need is a series of subassemblies that can be animated on their own, while retaining their relationship to the greater assembly.



## 164 4. Adding Character

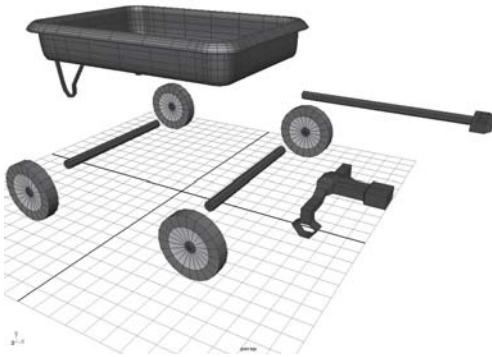


Figure 4.34: Wagon parts

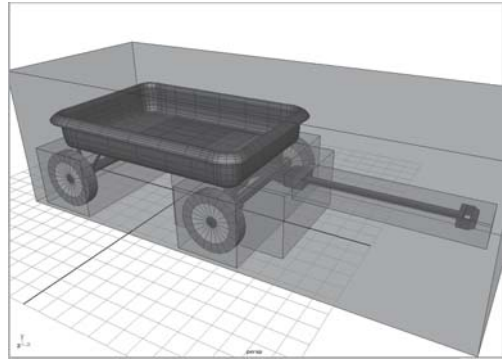


Figure 4.35: Group visualization

3. Turn on the visibility of the GroupBox layer. This layer contains a set of boxes that mimic the arrangement of subassemblies you will use to rig the wagon (see Figure 4.35).
4. Use the **Outliner** to examine the different subassemblies that make up the *wagonGroupBox* hierarchy (see Figure 4.36). At first there is only a *wagonGroupBox* node visible in the **Outliner**. By clicking on the plus sign next to the *wagonGroupBox* you can reveal its children nodes: *yokeGroupBox* and *rearGroupBox*. By clicking the plus sign next to *yokeGroupBox*, you can reveal the rest of the hierarchy: *handleGroupBox* and *frontGroupBox*.
5. Now, click on each *\_GroupBox* node in the **Outliner** to see which corresponding object is selected in the **perspective** view panel. Also, pay attention to the **Channel Box** to see what you'll animate on a given group node.

These boxes are only here to assist you in visualizing how grouping works. If each piece of real geometry in the wagon (i.e., *handle*, *wagon\_body*, *wheel\_FrontRight*, etc.) were contained by an imaginary box, and you were to lift the imaginary box, all of its contents (the geometry) would come along for the ride.

In this multi-part mechanical assembly, the wagon, it's important to group collections of your real geometry in sensible ways. What defines sensible? Well, in this case sensible groupings would include any objects that rotate together around a common pivot point. The rear wheels and axle are one simple example (see the *rearGroupBox*).

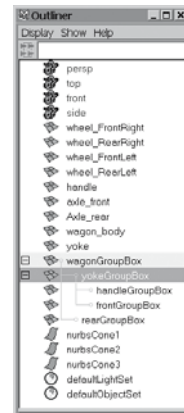


Figure 4.36: Outliner

Slightly more complex is the front assemblage. In it, the front wheels and axle rotate together as a group in order to roll (*frontGroupBox*), but this group also rotates with the yoke in order to turn (*yokeGroupBox*).

The handle functions similarly to the *frontGroupBox*, in that it rotates on its own to lift up and down, and turns with the yoke as a member of the *yokeGroupBox*.

Last, the entire wagon assembly needs to be able to turn and move around your scene, so the *yokeGroupBox*, *rearGroupBox*, and the wagon itself are all grouped under a node called the *wagonGroupBox*.

6. Now, you'll create the wagon's hierarchy. First, hide the boxes by clicking the "V" button next to the *GroupBoxLayer* in the **Layer Editor**.
7. In the **Outliner**, select the three objects that make up the rear assemblage: *axle\_rear*, *wheel\_RearRight*, and *wheel\_RearLeft*. (Use the **ctrl** key plus **LMB** to control the selection.)
8. Click **Edit > Group options** and reset the settings. Click the **Group** button to group these three objects together. When you group the objects, a new transform node called *group1* is created. This *group1* node is the parent of the three objects.
9. Rename the *group1* node *rearGroup*.

Go ahead and try rotating the *rearGroup* on its Z-axis. Notice how the pivot point is at the world origin (see Figure 4.37). Undo the rotation you just did, resetting the value to zero.

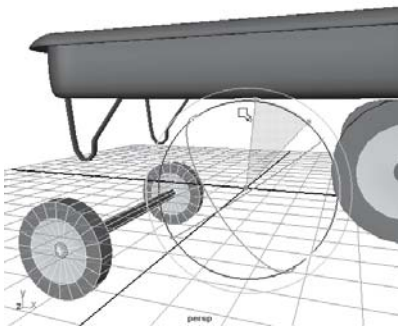


Figure 4.37: Group with bad pivot

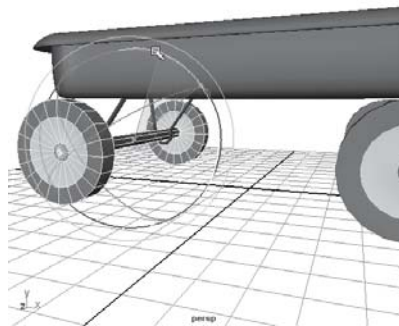


Figure 4.38: Centered axle pivot

## 166 4. Adding Character

---

10. Since newly created group nodes in Maya have their pivot points centered at the world origin, you will need to use the **Modify > Center Pivot** command. This places the *rearGroup*'s pivot at the center of its child geometry. If you rotate the *rearGroup* now you should get a much better result (see Figure 4.38).

11. Next, select and then group together these three objects: *axle\_front*, *wheel\_FrontRight*, and *wheel\_FrontLeft*. (You can use the **ctrl-g** hotkey.) Rename the group *frontGroup* and center its pivot.

12. Select the newly created *frontGroup* as well as the *handle* and *yoke* objects. Group these together, and rename the new group *yokeGroup*. Notice that groups can contain other groups.

Since this group won't be rotating at the geometric center of its child objects, the **Center Pivot** command won't set the proper pivot. You'll need to manually place the *yokeGroup*'s pivot point.

13. With *yokeGroup* still selected, pick the **Move Tool**, press the **Insert** key on your keyboard, and position the pivot point to the point where the *yoke* meets the *wagon\_body*. This is the point around which the entire *yokeGroup* assembly will rotate when turning the wagon.

14. The final hierarchical link to make is between the two top-level groups and the *wagon\_body* object. Top-level groups are those that appear at the top of the scene view in the **Hypergraph** or with a plus/minus sign next to them in the **Outliner**. In other words, they're at the top of their respective hierarchies. Select *rearGroup*, *yokeGroup*, and lastly *wagon\_body*, and then click **Edit > Parent options**. Reset the settings and then click the **Parent** button.

15. Test the system by moving and rotating the *wagon\_body*. All of the other parts should come along for the ride. Pick the *yokeGroup* and rotate it around on the Y-axis (see Figure 4.39). Try spinning the *rearGroup* on the Z-axis. You've just built a multi-part assemblage that can be independently animated without falling apart!

16. Save your work.

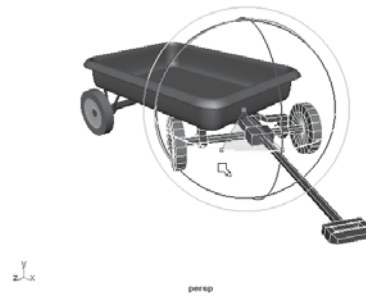


Figure 4.39: Rotated yoke group

## Transform Order

Hierarchies can be used to solve problems involving transformation order, too. If you stretch a sphere and then rotate it, you'll see a propeller-like motion. This occurs because Maya evaluates scale, then rotation, and lastly translation of an object. What if you want to rotate the object, but maintain its elongated silhouette? It can be done by setting up a series of parented nodes above the object. Each node will handle one specific transform. This way, we can circumvent Maya's natural transform order.

### Animation Exercise – 'Toon Tire

In this exercise we'll create a spinning tire that takes on cartoon proportions as it spins out, gains traction, and then takes off.

1. Create a capped NURBS cylinder with the following settings.

**Axis** = Z.

**Height** = 1.

**Caps** = Both.

Rename the cylinder *wheel* and then move it up to a value of 1 on the Y-axis.

In this type of hierarchy, it is important that the *wheel's* position, rotation, and scale not change, since its transforms will all be relative to the hierarchy above it. If the *wheel* moves, then all of the pivot points of the groups above will be incorrect.

2. Click **Window > General Editors > Channel Control...** Click on the Locked tab, and then **ctrl-LMB** select the three translate, rotate, and scale channels in the Non Locked side. Press the **<<Move** button to send them over to the Locked side. Now your Move, Rotate, and Scale tool will be grayed out when the *wheel* is selected.
3. In the keyable tab, select all the Keyable channels and then press the **Move>>** button to send them over to the Non-Keyable side as in Figure 4.40. (This is just a cleanup step to remove unneeded channels from the interface.)
4. Group the wheel to itself (**ctrl-g**), and rename this group *SpinGroup*. Click **Modify > Center Pivot** so that the group can rotate properly. In the **Channel Control** editor, remove and lock all of the *spinGroup's* keyable transform channels other than rotateZ.

## 168 4. Adding Character

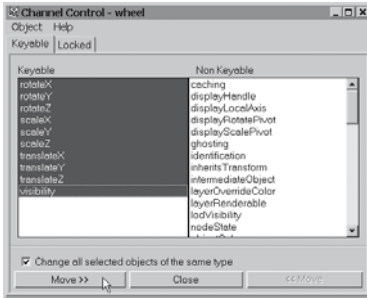


Figure 4.40: Channel control

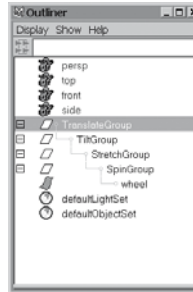


Figure 4.41: Final hierarchy

5. Select *SpinGroup* and group it to itself. Do not change the pivot point; we want it to stretch from the bottom of the object. Rename this group *StretchGroup*. Remove and lock all keyable transform channels other than *scaleX*, *scaleY*, and *scaleZ*.
6. Group *StretchGroup* to itself, and rename the new group *TiltGroup*. Remove and lock all keyable transform channels other than *rotateZ*.
7. Group *TiltGroup* to itself and name the new group *TranslateGroup*. Remove and lock all keyable transform channels other than *translateX*, *translateY*, and *translateZ*.
8. With this hierarchy in place, it is now possible to spin the wheel, squash and stretch it, tilt it, and move it (see Figure 4.41). The order of transformation evaluation will keep the wheel's spinning motion from interfering with the scaling.

Frequently, spinning objects can be difficult to “read.” Think of a moving car’s spinning hubcap, which seems to move fast, then slow, stand still, then move backwards. This optical illusion can often be overcome by introducing irregularity to the pattern of the spinning object.

9. In the **Hypershade**, create a new **Lambert** material. Map a **Ramp** texture to the **Color** attribute of the material, using the default values. Assign this material to your *wheel*. Now the *wheel*'s rotation will be much easier to see.

10. In the front panel, select the *wheel*. Now “pick walk” up (by pressing the up arrow on your keyboard) once to select the *SpinGroup*. At frame one, set the *rotateZ* value to  $0^\circ$  and keyframe it.

11. Advance the time slider to frame 72. Rotate the *SpinGroup*  $-4320^\circ$  on the Z-axis. Set a keyframe.

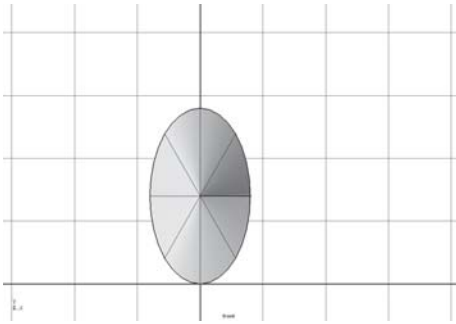


Figure 4.42: Frame 1 stretch

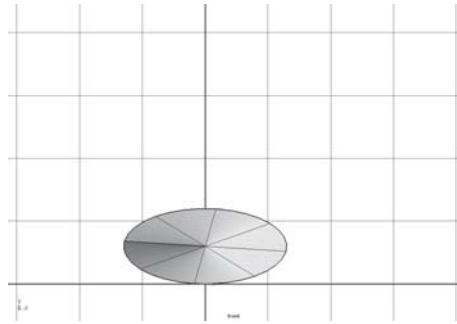


Figure 4.43: Frame 10 squash

12. Play the animation. The *wheel* (as a child of the *SpinGroup*) will rotate at a very high rate.

13. Stop and rewind the **Time Slider**.

14. Pick walk up to the next node, the *StretchGroup*. At frame 1, give the *wheel* some stretch by scaling 1.4 on the Y-axis, and 0.8 on the X- and Z-axes. Set a keyframe for the scale channels (see Figure 4.42).

15. Advance to frame 10, squash the *StretchGroup* down to 0.6 on the Y-axis and 1.3 on the X- and Z-axes. Key these channels (see Figure 4.43).

16. Go to frame 18, stretch the Y-axis scale up to 1.4, and the X- and Z-axes down to 0.8. Then set a keyframe (see Figure 4.44).

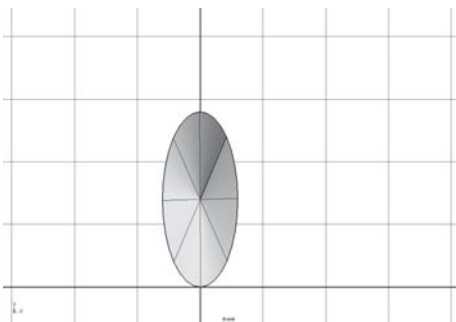


Figure 4.44: Frame 18 stretch

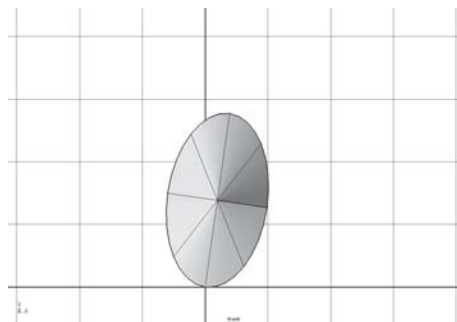


Figure 4.45: Frame 1 tilt

## 170 4. Adding Character

---

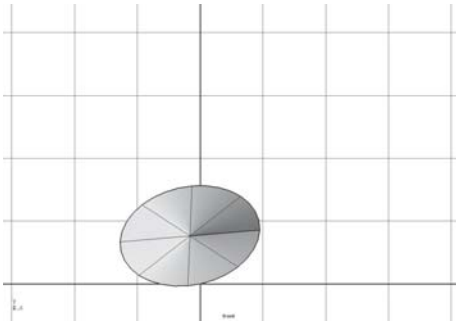


Figure 4.46: Frame 13 tilt

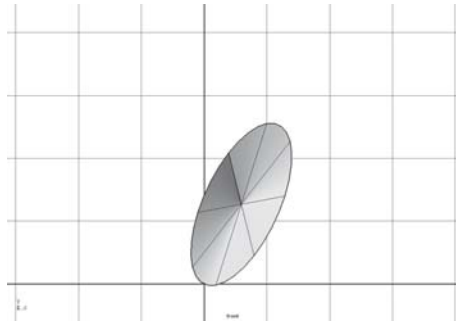


Figure 4.47: Frame 25 tilt

17. Play the animation. Notice how the *wheel* now spins *before* it squashes and stretches. We have successfully circumvented Maya's natural transformation order!

18. Stop and rewind. Now, we'll tilt the *wheel* forward and backward in anticipation of its big move. On frame 1, pick walk up to the *TiltGroup* and rotate it  $-8^\circ$  on the Z-axis. Set a keyframe (see Figure 4.45).

19. Advance to frame 13, rotate the *TiltGroup* to  $12^\circ$  on the Z-axis, and set a keyframe (see Figure 4.46).

20. On frame 25, set another keyframe for rotateZ in order to hold the *TiltGroup* (see Figure 4.47).

21. Next, go to frame 29, rotateZ to  $-25^\circ$ , and key it.

22. Play the animation to test it.

23. The last keyframes to set are for translation. Go to frame 13, pick walk up to the *TranslateGroup* and set a keyframe for translateX.

24. On frame 20, translateX to -0.5 and set a keyframe (see Figure 4.48).

25. Using the same value of -0.5, set another translateX keyframe at frame 30.

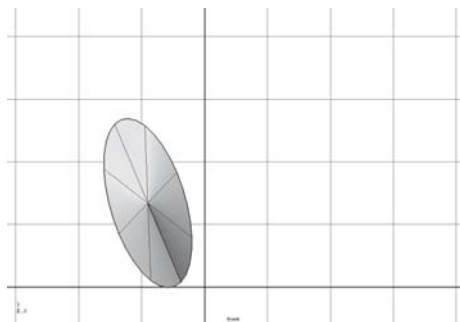


Figure 4.48: Frame 20 translation

26. Advance the time slider to frame 72, translate the *TranslateGroup* 80 units on the X-axis, and set a keyframe.

27. Play your animation back to see the final effect (see Figure 4.49).

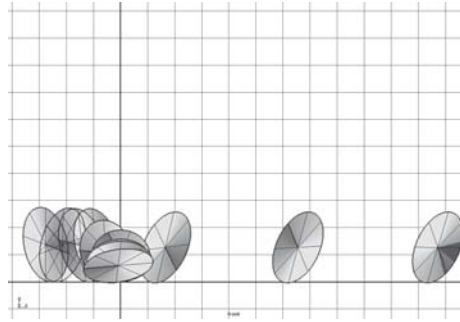


Figure 4.49: Final animation of the ‘toon tire

## 4.5 Shading – Bump Maps and Transparency



Look at most objects in the real world and you’ll find that they have not just color, texture, and shininess but a whole host of other attributes that make them unique. The surface of a piece of sandpaper, for example, has a “bumpiness” that is caused by many tiny bits of sand that are adhered to the paper. In Maya, you can map a file texture or simple noise texture to the bump map slot of a material to mimic this effect. A bump map will cause the light that hits the object’s surface to play off it as if there were actual positive and negative depths.

### Bump Maps

Without adding geometry to the surface of a sphere, bump maps increase the perceived detail (see Figure 4.50). With the exception of very smooth surfaces such as glass and polished stone, almost all materials should incorporate some amount of bump mapping.



Figure 4.50: Identical geometry before and after bump mapping with this pattern



### Transparency

The transparency of a material determines - you guessed it - how see-through it is. You may simply change the Transparency attribute with the slider. For more specific, localized, or complex transparency it's possible to map a texture to the transparency attribute.

A black and white polkadot pattern placed into the transparency slot, for example, will cause an object to have alternating opaque (where the map is black) and transparent (where the map is white) areas when rendered (see Figure 4.51).



Figure 4.51: Identical geometry before and after transparency mapping with this pattern

### Shading Exercise – Butterfly

1. Set your project to `understanding_maya\Chapter04\shading_butterfly`.
2. Create a new scene.
3. In the side view, create a curve in the half-profile shape of a butterfly's body, being careful to snap the first and last points to the thick grid line at the Z-axis origin by turning on **Grid Snapping** (see Figure 4.52).
4. With the curve selected in Object Mode, use the **Revolve** command with the axis set to **Z** to create the surface.
5. Name the object *butterflyBody*.
6. In CV selection mode, reshape the points of the surface to create a better shape (see Figure 4.53).
7. Now we'll create some simple wings. Click on **Create > Polygon Primitives > Plane**

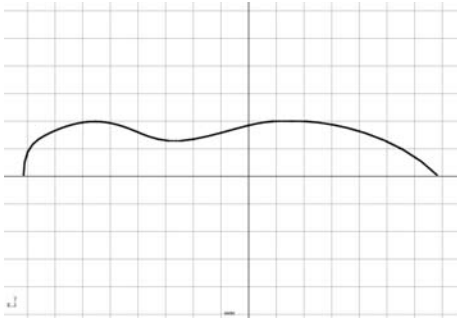


Figure 4.52: Revolve curve

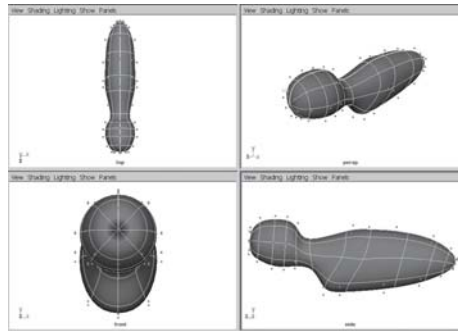


Figure 4.53: Butterfly body

**options.** Set the number of **Subdivisions Along Width** and **Subdivisions Along Height** to 1, and click **Create**.

8. Name the plane *leftWing*.

9. In the top view panel, move and scale *leftWing* into position, about 10 on TranslateX, 20 on ScaleX, and 30 on ScaleZ.

10. When you're developing the look of your scene, it can be helpful to arrange the panels with a **Perspective** and **Hypershade** split, as well as the **Attribute Editor** docked on the right side of the workspace. Once you have this set up, create a new **phong** material, and name it *phongWing*.

11. Assign *phongWing* to the *leftWing* object by **MMB**-drag-dropping it. In the **perspective** panel, smooth shade the scene and turn on hardware texturing by pressing **5** and then **6** on your keyboard (see Figure 4.54).

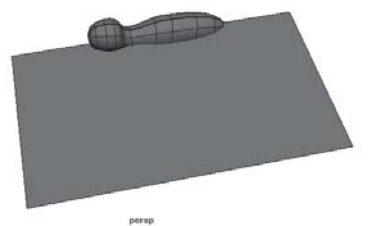


Figure 4.54: Butterfly wing

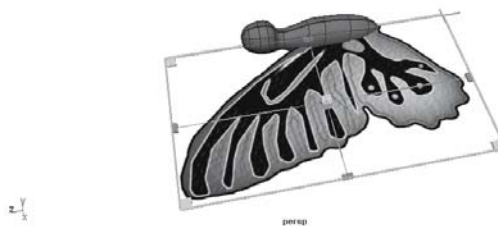


Figure 4.55: Color and transparency applied

## 174 4. Adding Character

---

12. Next we'll map some of the *phongWing* material's attributes with file textures. In the **Attribute Editor**, click on the **Create Render Node** button (the little checker icon) to the right of the **Color** attribute. When the **Create Render Node** window pops up, choose the **Textures** tab, check **With New Texture Placement** on, choose **Normal** from the **2D Textures** turn-down menu, and then click on the **File** icon. In the **file** field, rename this node *fileWingColor*.

13. In the **Attribute Editor**, click the folder icon next to the **Image Name** field. From the **Open** window, choose the *bfly\_color.tga* image, then click the **Open** button. (The 32-bit Targa file contains 24 bits of color information and an 8-bit alpha channel, which is used to specify areas of transparency. Maya will automatically connect this file node to both the color and transparency attributes of the material.)

14. To map the texture correctly on the *leftWing* object, choose the **Modeling** menu set, and select the *leftWing* object. Then click **Edit Polygons > Texture > Planar Mapping options**. Reset the settings, then choose **Y Axis** in the **Mapping Direction** setting. Then click the **Project** button.

15. The *leftWing* is now mapped with the color and transparency texture. It's currently upside down, so we'll fix that in the **Attribute Editor**. The *polyPlanarProj1* tab of the *leftWing* object should be selected. Under the **Projection Attributes** turn-down menu, set the **Rotate** attribute to 90, 0, 0.

16. Select the *phongWing* material in the **Materials tab** of the **Hypershade**. Then, click **Graph > Input Connections**. This expands the shading network graph in the **Work Area tab**, revealing the connections between the *fileWingColor* node and the *phongWing* node's color and transparency attributes.

17. Hover the cursor over the top connection between the file and material nodes. You'll see a bit of pop-up text appear to indicate which attributes are connected. In this case it says that the *fileWingColor.outColor* attribute is connected to the *phongWing.color* attribute.

18. This time, hover the cursor over the bottom connection between the file and material nodes. This indicates that the *fileWingColor.outTransparency* connects to the *phongWing.transparency* attribute (see Figure 4.55).

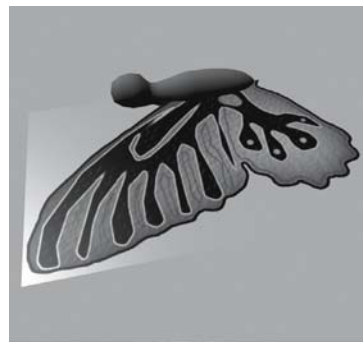


Figure 4.56: Improper specular map

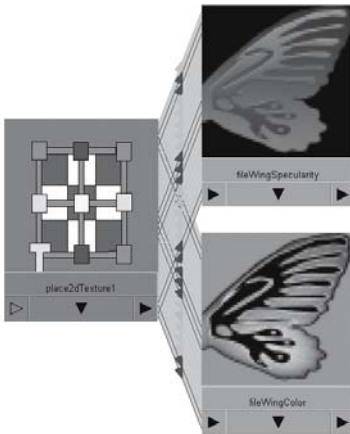


Figure 4.57: Shading network



Figure 4.58: Corrected specularity

19. Render a single frame of your scene. You may notice that from certain angles, the effect of the transparency is ruined due to the specularity of the Phong material, as in Figure 4.56. We'll use a specular map texture to specify shiny parts of the wing.

20. Select the *phongWing* material in the **Hypershade**. In the **Attribute Editor**, click on the **Create Render Node** button next to the **Specular Color** attribute. Uncheck **With New Texture Placement**, and choose **File** texture. For the **Image Name** attribute, browse to the *bfly\_specularity.bmp* file. Rename this node *fileWingSpecularity*.

21. The *fileWingSpecularity* node needs to use the same 2d placement utility as the color file. This can be done in the **Hypergraph** by **MMB**-drag-dropping the *place2dTexture1* node onto the *fileWingSpecularity* node and choosing the **Default** option (see Figure 4.57).

22. Now re-render the frame. The shiny area has been corrected, as in Figure 4.58.

23. The last file you'll map is for the bumpiness of the material. Select the *phongWing* material. In the **Attribute Editor** you will click the **Create Render Node** button to the right of the **Bump Mapping** attribute. Again, choose a **File** texture. This will create two things: a **Bump 2d** utility, and a **File** texture. Rename the **bump2d1** node *bump2dWing*. Rename the file texture *fileWingBump*.

24. Select the *fileWingBump* node. In the **Attribute Editor**, browse to the *bfly\_bump.bmp* file. As before, connect the shared *place2dTexture1* node to the *fileWingBump* node (see Figure 4.59).

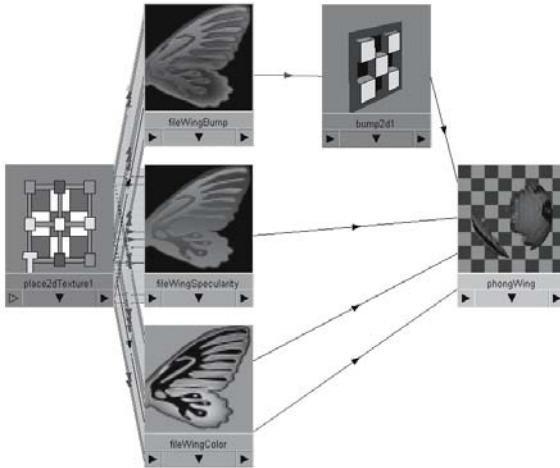


Figure 4.59: Finished shading network



Figure 4.60: Duplicated wing

25. Re-render the scene to see the change. If you want less extreme bumpiness, lower the **Bump Depth** value of the *bump2dWing* node to something more moderate, like 0.3.

26. Now you can duplicate the *leftWing* object with a *ScaleX* value of -1, to mirror it. Move it into position on the other side of the body (see Figure 4.60).

27. Save your scene as *shading\_butterfly\_end.ma*.

## 4.6 Lighting – Area Lights and Raytracing



The type of light we see in the real world is a very complex entity to re-create on the computer. It's the actual energy of light bouncing off surfaces, casting colored bounce light and soft shadows, that we try to emulate when placing light sources in Maya. The spot, point, and direct lights we've used so far are fairly simple, largely unrealistic types of light when photorealism is the goal.

Some rendering engines, such as mental ray, use a method of indirect light calculation called global illumination. Global illumination approximates the actual energy of light and its ability to bounce from one surface to the next (see Figure 4.61). In a traditional live-action lighting scheme, many bounce cards are set up to increase the amount of reflected light. Radiosity renderers can use such bounced light as well. However, radiosity can be a very slow, processor-intensive way to achieve realistic lighting.

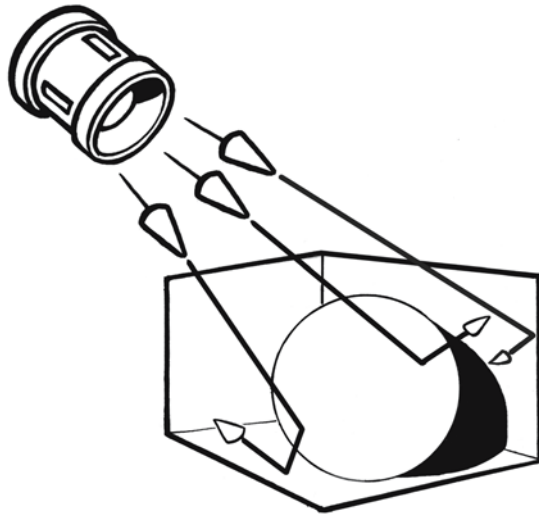


Figure 4.61: Bounced light

A way to fake such bounced light in Maya's software renderer is to supplement your lighting setup with Area lights. Area lights are two-dimensional cards that emit light from multiple points on their surface. You can place them in your scene much as you would a bounce card in a live-action film shoot. They will cast light from their surface and can be resized to tune the lighting. Changing their color will help achieve the radiosity look.

### Lighting Exercise – Cornell Box Area Lights

1. Create a NURBS cube by clicking **Create > NURBS Primitives > Cube**. NURBS cubes are actually a group containing six planes to define the sides. Since these planes are separate objects, we can easily color them independently of each other. However, we need to be careful to select the highest node (*nurbsCube1*) before moving, rotating, or scaling the cube, or else it will fall apart.
2. Scale and translate the *nurbsCube1* node to create a room for our lighting test. A scale value of 8 on all three axes will work. Translate 4 units on the Y-axis to set the bottom of the cube on the grid.
3. Delete the face of the cube named *frontnurbsCube1*. Removing this wall allows us to peer inside the room.

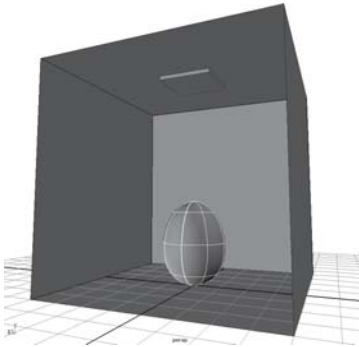


Figure 4.62: Room and egg

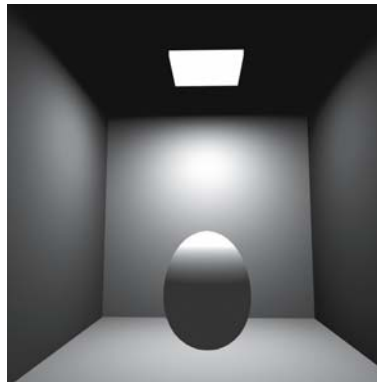


Figure 4.63: Rendered top area light

4. To suggest a light source for the room, create a polygon cube primitive, rename it *lightbox*, and place it just below the ceiling. Translate it to 0, 7.9, 0 and scale it to 2, 0.15, 2.
5. Create an egg to place inside the room. You can start with a NURBS sphere and then adjust the CVs to form the egg shape. Rename the object *egg* (see Figure 4.62).
6. Open the **Hypershade** by clicking **Window > Rendering Editors > Hypershade...** Create four Lambert materials. Color one red, one blue, one off-white, and one white. Assign the red material to the left wall, the blue one to the right, the off-white one to the floor, ceiling, and back wall, and the white one to the egg.
7. Create a Lambert material for the light fixture. Color this one white, and set its Incandescence value to white. This will add to the illusion that it is emitting light.
8. Next, to begin lighting, click **Create > Lights > Area Light**. Move the area light to the top of the room (7.9 on the Y-axis) and rotate it  $-90^\circ$  on the X-axis so it points down. Hit the 7 key on your keyboard to view interactive hardware lighting.
9. Render the current frame and notice the nice throw pattern of light on the walls (see Figure 4.63). There are dark corners and bright centers to the walls. Area lights emit more or less light, depending on their size. Scale the light up to 1.25 on all three axes and then re-render the frame. Notice how the lighting has increased. See Figure 4.63.
10. In the **Attribute Editor**, lower the area light's intensity to 0.35, then render the frame. The lighting on the egg is a bit more subtle this way, but it hits the walls more evenly.

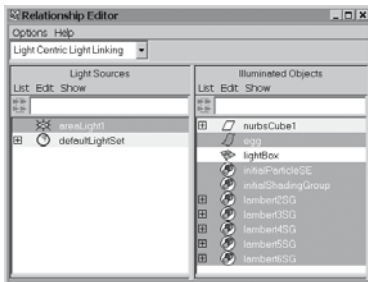


Figure 4.64: Light linking

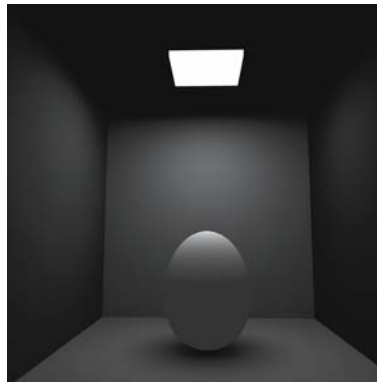


Figure 4.65: Softened shadows

11. Turn on the **Use Depth Map Shadow** checkbox for the *areaLight1*. Render and you'll notice a problem – the *lightBox* object is casting a shadow and obscuring most of the light. In order to remedy this problem, open the **Window > Relationship Editors > Light Linking > Object-Centric...** In this window, the object that you select on the left can be linked or unlinked from the scene lights as in Figure 4.64. Click the *lightBox* entry in the **Illuminated Objects** list. Notice that *areaLight1* is highlighted in the **Light Sources** list, indicating a connection. To break the connection, click on *areaLight1* in the list. Render the frame and you'll see that the problem has been fixed.

12. Now the egg's shadow looks a bit sharp. In the **Attribute Editor**, change the **Dmap Resolution** to 64, and set the Dmap Filter Size to 5. Render a frame and you'll see that this softens the shadows as in Figure 4.65.

13. Create another area light for the left wall. This one will simulate a red light being reflected off the wall. Leave the rotation values at 0,0,0, and move it to 0, 2.5, 4. Again, try a low intensity between 0.05 and 0.1. Set the light's color attribute to an HSV value of 360, 0.3, 1. Render the scene to see the result.

14. You can duplicate the left wall light to create the blue light for the right wall. Move the new light to 0, 2.5, -4 and rotate it 180° on the X-axis. Change the light's color attribute to HSV 240, 0.3, 1 in order to add a bluish cast.

15. We need to add one more area light to the scene to cast light from the front (see Figure 4.66). Translate the new light to 3.3, 2, 0 and rotate it 90° on the Y-axis. Set the intensity to 0.2.

16. Render the scene. You should see an image similar to Figure 4.67.



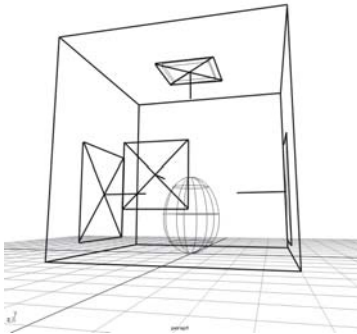


Figure 4.66: Light positions

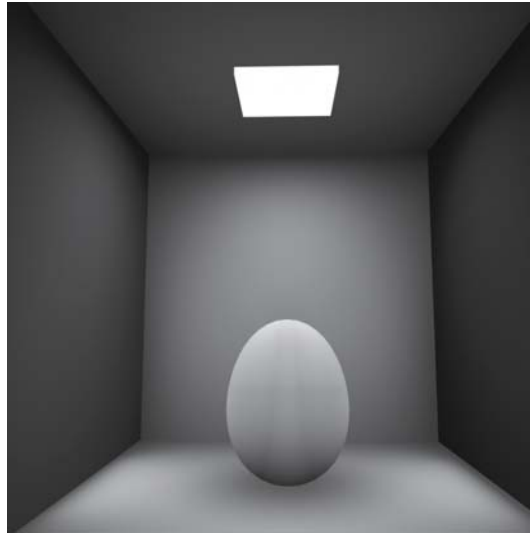


Figure 4.67: Fully lit egg

### Raytracing

Raytracing is an accurate method of calculating the reflections light rays make as they bounce around your scene. The path of a given light ray is “traced” by Maya’s renderer, resulting in precise interpretations of your light placements, shadow and material reflections, transparency, and refraction.

The first step in using raytracing is to turn on the **Raytracing** option in the **Render Globals**. Next you must specify where you want to spend the extra rendering time (raytracing is *not* fast) in your scene. By using raytraced shadows in your lights and turning up the reflectivity in a shader you’ll begin to see realistic interactions of light and surfaces.

### Lighting Exercise – Shiny Egg Raytrace

1. Set your project to `understanding_maya\Chapter04\lighting_raytrace`.
2. Open the scene `lighting_raytrace_box_begin.ma`.

3. Select the *egg* object and open the **Attribute Editor**. Pick the *lambertegg* tab, and then change the **Type** attribute from **Lambert** to **Phong**. Notice one of the new attributes available in the **Specular Shading** area is **Reflectivity** (see Figure 4.68). Once raytracing is enabled, this value will determine how reflective the surface is.
4. Open the **Render Globals** by clicking the third clipboard icon on the **Status Line**. Set the **Resolution** to the **1k Square** preset. Also, increase the **Anti-aliasing Quality** to **Intermediate**.
5. In the **Raytracing Quality** section of the **Render Globals**, click the checkbox to turn on **Raytracing**. Then, set the **Reflections** to a value of 3, and the **Refractions** and **Shadows** to 0.
6. Render the frame. You'll see the room reflected off the *egg*'s surface.
7. Now, try increasing the **Reflectivity** value of the *egg*'s Phong material to a value of 0.75. When you render again, the egg will be more reflective (see Figure 4.69).



Figure 4.68: Reflectivity attributes

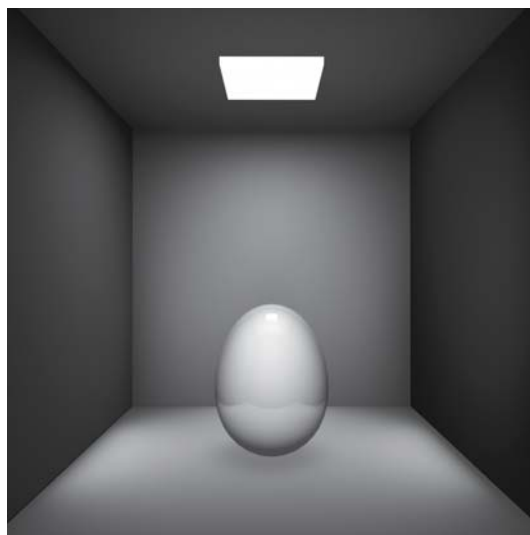


Figure 4.69: Raytraced render

## 182 4. Adding Character

You'll notice that the *egg* is reflecting an empty black area where the front wall of the room is missing. Rather than build more geometry to correct this problem, you can use a file texture. The **Reflected Color** attribute of the Phong material will be used on any part of the *egg* that isn't reflecting true scene geometry. This is a great way to suggest a complex environment.

8. Select the *phong1* material in the **Hypershade**. In the **Attribute Editor**, click the **Create Render node** button next to the **Reflected Color** attribute. Choose the **Env Ball** texture from the **Environment Textures** dropdown list, making sure that **With New Texture Placement** is turned on (see Figure 4.70).

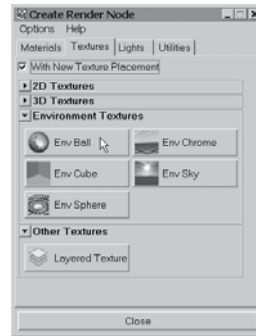


Figure 4.70: Env ball texture

The environment ball lets you map a file texture to the inside of an invisible, infinitely large sphere that surrounds your scene. This image is usually created by photographing a chrome ball (such as a large Christmas tree ornament) in an environment you want to capture. This distorted image is then un-distorted and reflected off your 3D objects.

9. You can graph the shader network you are working on, by selecting the *phong1* material in the **Hypershade**, and clicking **Graph > Input Connections**. Select the *envBall3* node.

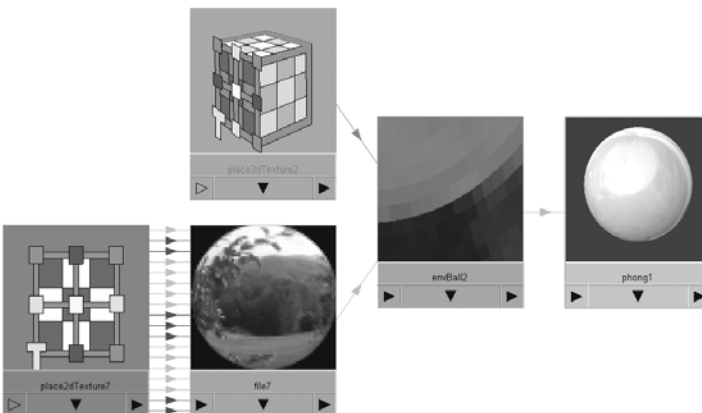


Figure 4.71: Env ball mapped to the reflected color attribute

10. In the **Attribute Editor**, click the **Create Render node** button next to the **Image** attribute. Choose **File** from the **2D Textures** dropdown list.

11. Click the **Image Name** folder, and browse to the *sp\_bfrm.bmp* image (see Figure 4.71).

12. Render the frame. This time you see an outdoor reflection in the *egg*, along with the true scene reflections (see Figure 4.72).

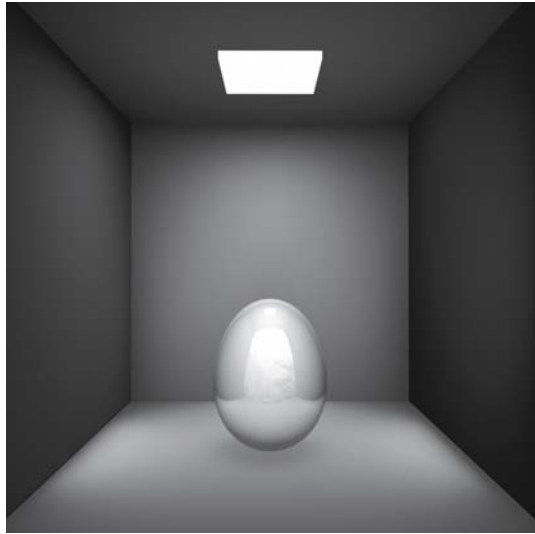


Figure 4.72: Outdoor reflection

## 4.7 Rendering – Camera Effects



In the real world, if you point a camera at a bright light source and snap a picture, you'll get a bright glowing spot in your picture, along with a lens flare. The glowing is caused by overexposure. The lens flare is produced by the series of glass lenses that the light passes through on the way to your negative.

In Maya, however, neither of these elements will appear in your renders by default. Let's say that you create a point light. If you point a camera at it and render the frame, your render will be black. To see the light glow and lens flare, you must enable **Optical FX**.

### Light Glows and Lens Flares

**Optical FX** allows you to add glows, halos, and lens flares to your lights. The **Optical FX** node is created by selecting a light and clicking the **Light Glow** attribute **Create Render node** button. Then you can access the many settings of the **Optical FX** node. These settings control the type of glow, halo, and lens flare that will be seen in your renders.

### Rendering Exercise – Optical FX Egg

1. Set your project to `understanding_maya\Chapter04\rendering_optical_egg`.
2. Open the scene `rendering_optical_egg_begin.ma`.
3. Play the animation. You'll see a hinged egg opening its top.
4. Create a point light by clicking **Create > Lights > Point Light**. Place the light inside the top of the egg, where it'll be seen when the egg opens as in Figure 4.73.
5. On frame 16 render a frame. The light is affecting the scene, but it is not itself visible.
6. Open the **Attribute Editor**, with the `pointLight.Shape1` selected. In the **Light Effects** section, click the **Create Render node** button next to the **Light Glow** attribute. This will automatically create and connect an **Optical FX** node, as well as a render sphere attached to the light (see Figure 4.74).
7. Render the frame. Now you can see the light and a four-point glow effect. If you like, experiment with the different **Glow Types** and related settings.

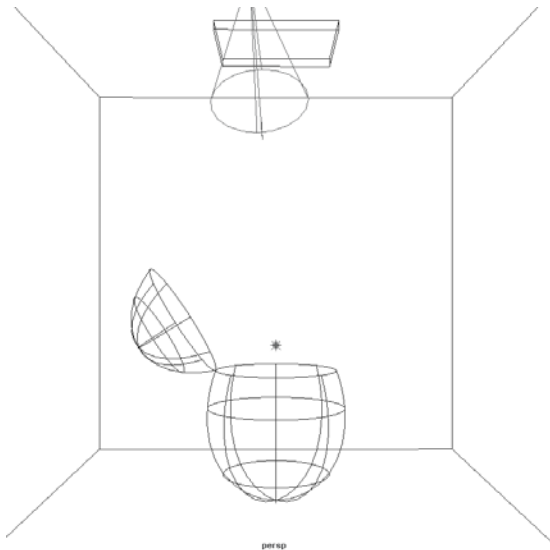


Figure 4.73: Point light placement

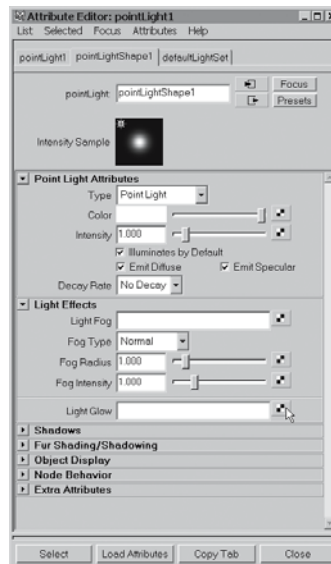


Figure 4.74: Enabling light glow

8. The **Lens Flare** is not enabled by default. Turn it on by clicking the **Lens Flare** checkbox, and then render the frame to see the result (see Figure 4.75).

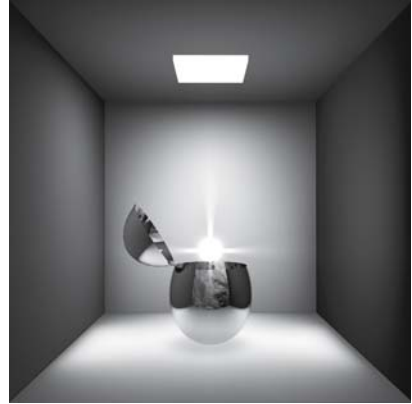


Figure 4.75: Lens flare and light glow

When you are fine-tuning light settings, effects, materials, and other render settings, it can be pretty inefficient to continually re-render the scene after every adjustment. For these cases, Maya provides an excellent solution: the **Interactive Photorealistic Renderer**, or **IPR**.

The **IPR** updates your rendered image in close to real-time. As you adjust the settings on your effects, you'll see instant results appear on your final image.

9. First, turn off **Raytracing** in the **Render Globals**. (**IPR** doesn't work with raytracing.) With your **perspective** view active, click the **IPR** render button on the **Status Line**. Your frame will be rendered, and then you'll be instructed to "Select a region to begin tuning." **LMB**-click-drag a marquee selection around the portion of your image you'd like to refine, as in Figure 4.76. (The smaller the area, the quicker the updates will be.)

10. Now continue to tune the lens flare attributes, such as **Flare Vertical**, **Flare Horizontal** (used to rotate the row of flare shapes), **Flare Intensity**, and **Flare Min/Max Size** (see Figure 4.77).



Figure 4.76: IPR render



Figure 4.77: Tuned lens flare

## 186 4. Adding Character

---

11. When you are finished, click the **IPR** stop sign icon. This clears the **IPR** layers from memory. You should now render a frame with the usual render button. (But be sure to turn **Raytracing** back on in the **Render Globals** first.)

12. Return to frame 1 on the timeline and render a frame. There's a problem: the light should not be visible when it's hidden from view by the egg's lid. When a light is fully occluded from a camera, no lens effects can occur. In order to make this work in Maya, you must set the render sphere to an appropriate radius. When the render sphere is fully hidden from the camera, so is the optical effect.

13. Change the *sphereShape1* radius to 0.4. Render the frame – no more glow!

14. Try batch rendering this animation in order to see the glow appear as the egg opens (see Figure 4.78 and Color Plate VIII).

15. Save your scene.

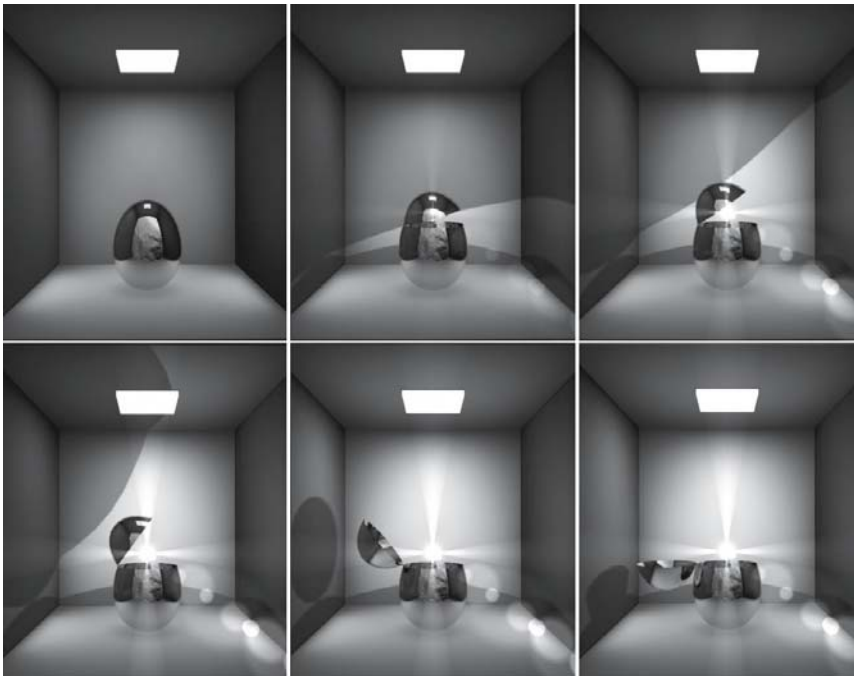


Figure 4.78: Behold the animated space egg

### 4.8 Project 4 — Chicken Added to Scene

Model the chicken character using polygonal and subdivision surfaces. Refine the egg's animation using the different tangency types.

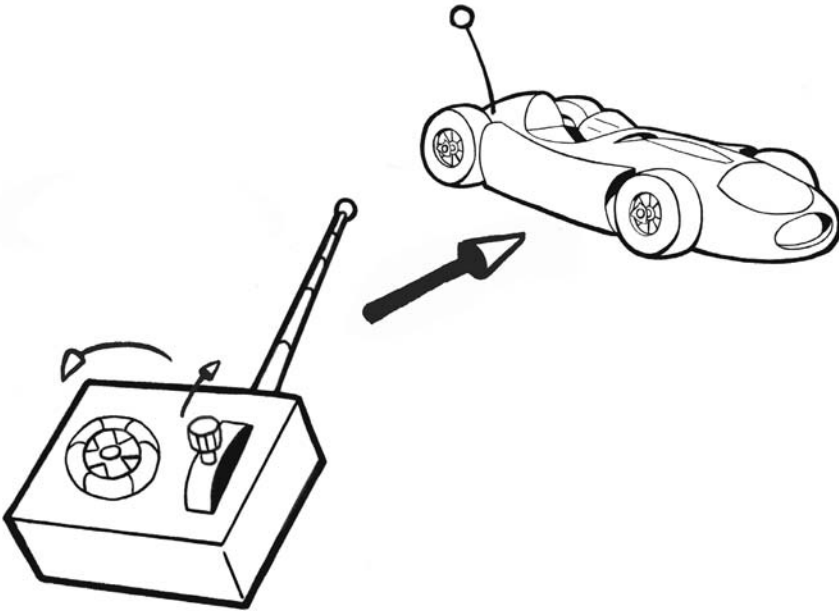
- Use the Graph Editor to influence the egg's in-between.
- Create more complex shaders using bump mapping, reflections, transparency, and refractions.
- Fake colored bounce light with Area lights.
- Try adding optical effects to the render.



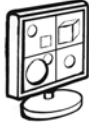
---

# Chapter 5

## Wiring Things Up



### 5.1 Workspace – Artisan Tools



Admittedly, the three-button mouse isn't the best human-machine interface ever designed. For certain operations, a pen and tablet work much better. The Maya tools that benefit most from this interface are known as the **Artisan** tools. **Artisan** is a paintbrush-like paradigm for directly affecting data on an object's surface. Pushing and pulling the surface of a model, painting weight values for bone and cluster assignments, selecting points on a surface, painting vertex colors, and even painting flowers onto a hillside are all accomplished with this interface.

Using the Stamp Profile, Operation, and Sculpt Variables settings you can change the brush's functionality. Here are the ten tools that use the **Artisan** interface and their functions:

- Sculpt Surfaces Tool – model NURBS surfaces,
- Sculpt Polygons Tool – model polygonal surfaces,
- Paint Selection Tool – select objects or components such as vertices, edges and UVs,
- Paint Weights Tool – control the percentage of influence a deformer has over points,
- Script Paint Tool – open brush interface for writing your own MEL paint tool,
- Attribute Paint Tool – another catchall tool brush,
- Paint Set Membership Tool – control point memberships in rigid skin binds,
- Paint Skin Weights Tool – control joint influence in smooth skin binds,
- Paint Vertex Color Tool – vertex color used in real-time video games, and
- Paint Textures Tool – 3D paint tool for NURBS and polygon surfaces.

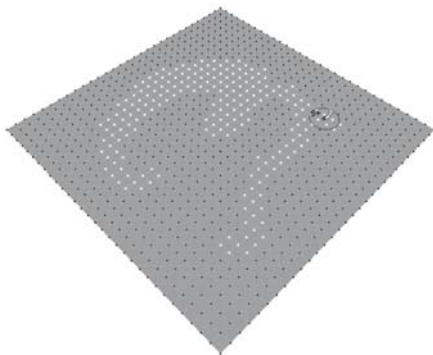


Figure 5.1: Paint select

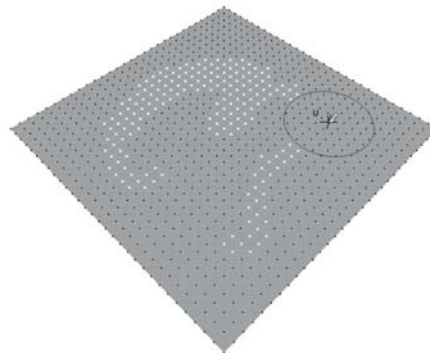


Figure 5.2: Paint unselect

## Workspace Exercise – Artisan

1. First you'll create a large polygon plane object with which to work. Click **Create > Polygon Primitives > Plane**. In the **Inputs** section of the **Channel Box**, set the **Width** and **Height** to 20 each and the number of **Subdivisions** to 30 in both directions.
2. Dock the **Tool Settings** to the right side of the interface by clicking **Display > UI Elements > Tool Settings**.
3. Selecting irregular patterns of vertices can be difficult with the normal **Select Tool**. The Artisan interface has a nice alternative. With the *pPlane1* object selected, click **Edit > Paint Selection Tool**. Notice that your cursor has changed into a paintbrush.
4. Now try selecting some vertices by **LMB-click-dragging** across the surface. You've selected some vertices based on the radius of your brush and the current artisan operation setting (see Figure 5.1).
5. In the **Tool Settings** palette, increase the **Radius (U)** value to 3, and then switch the **Paint Operation** to **Unselect**.
6. Drag across the surface to remove some of the vertices from the current selection (see Figure 5.2).
7. While you still have some vertices selected, switch to the **Move Tool**. Translate the vertices up a few units on the Y-axis. You can see how this technique can be very useful for creating landscapes and similar environments (see Figure 5.3).

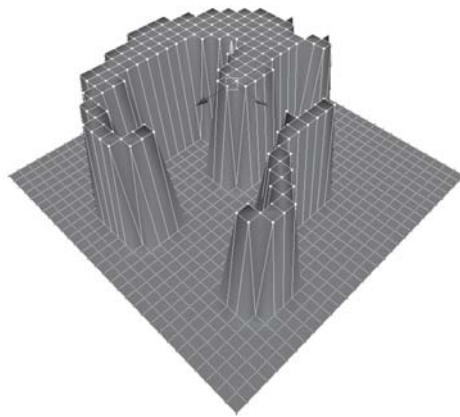


Figure 5.3: Move vertices

8. Undo the translation you performed on the vertices by clicking **Ctrl-z** on the keyboard.

## 192 5. Wiring Things Up

---

The artisan brush that combines selection and translation called the **Sculpt Polygons Tool**. This tool can push or pull vertices along the surface normal direction of your model.

9. Select the *pPlane1* in object mode.
10. In the **Modeling Menu Set**, click **Edit Polygons > Sculpt Polygons Tool**. In the **Tool Settings** pallet, click **Reset Tool**.
11. Try **LMB**-click-dragging the cursor across the surface. Notice how it indents wherever it passes (see Figure 5.4).
12. You can set your brush radius and the operation, as before, in the **Tool Settings** pal-

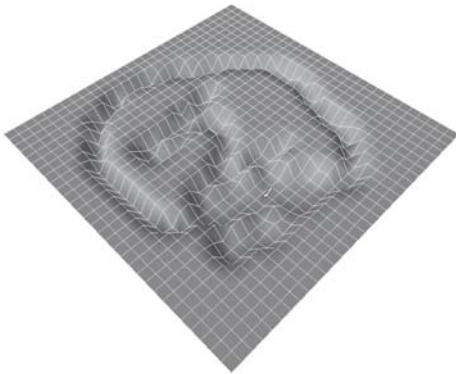


Figure 5.4: Sculpt push

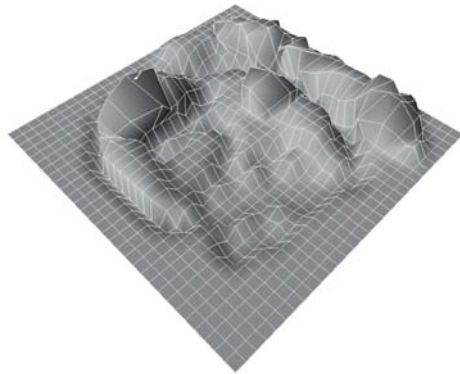


Figure 5.5: Sculpt pull

let. Try a **Radius** of 1, and a **Paint Operation** of **Pull**. Now when you drag the cursor, you're creating mounds on the surface (see Figure 5.5).

13. The results are a bit on the rough side. Switch the **Paint Operation** to **Smooth**. Next press the **Flood** button a few times. Each time this button is pressed, the vertex positions are averaged, smoothing out the overall appearance of the object without adding or subtracting any geometry (see Figures 5.6 and 5.7).

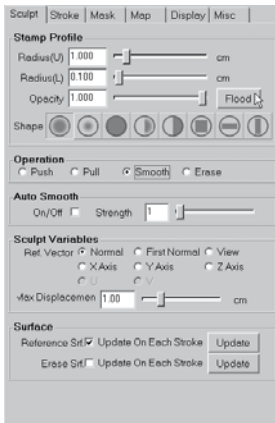


Figure 5.6: Smooth operator

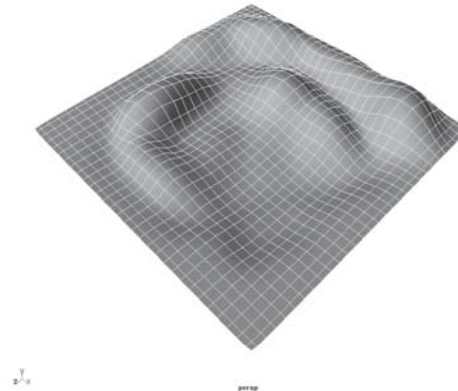


Figure 5.7: Paint smooth

## 5.2 Modeling – Connecting Surfaces Using Fillets



Frequently, you'll build a NURBS model from a series of intersecting parts, such as NURBS primitives, lofts, and revolves. In these cases, it is possible to create blended transitions to mask unsightly object intersections. One way to create such blends is with the **Circular Fillet** command. (A fillet is a concave surface that blends between the tangencies of two other surfaces as in Figure 5.8.)

These blended surfaces are particularly useful when creating props, vehicles, and other “hard surface” models. Due to the intensive calculations performed to maintain tangency between all three surfaces, blends can be very slow to deform in animation. It's best to avoid using blends to connect a character's arm to his torso, for example.

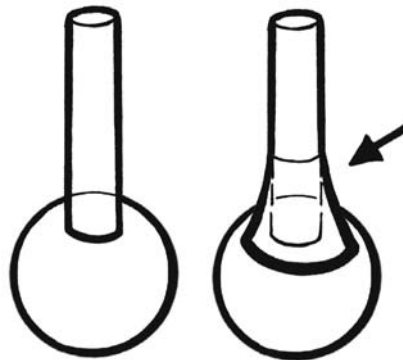


Figure 5.8: Surface fillet

### Modeling Exercise – Submarine Fillets

1. Set your project to `understanding_maya\Chapter05\modeling_submarine`.
2. Open the scene `modeling_fillets_submarine_begin.ma`.

You'll see a toy submarine that has been modeled primarily by reshaping NURBS cylinders (see Figure 5.9). There are some abrupt intersections between parts that we would like to fix by creating circular fillets to hide those transitions.



Figure 5.9: Initial model

3. Switch to the **Modeling Menu Set**.
4. Select the *subPeriscope* object; then shift select the *subStack* object.
5. Click on **Edit NURBS > Surface Fillet > Circular Fillet** options...
6. Reset the default settings, and then change the **Radius** to 0.3.

7. Click the **Apply** button. (We'll be creating multiple fillets, so leave this window open.)

8. The new object that appears connects the periscope to the stack of the submarine. Press **3** on your keyboard to increase the display resolution. The *circularFilletSurface1* object is tangent to the periscope at its top and tangent to the stack at its bottom (see Figure 5.10).

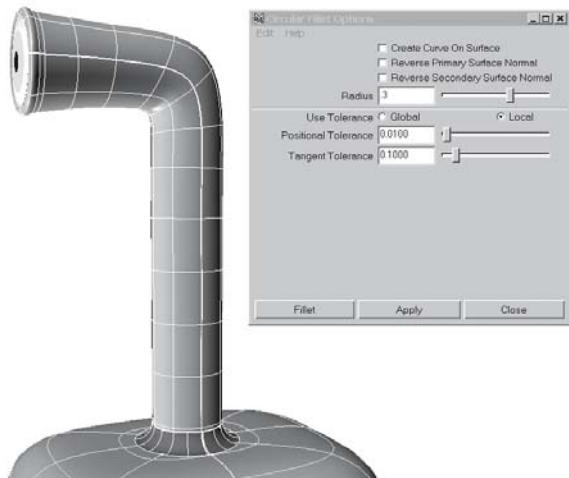


Figure 5.10: Periscope fillet

9. Next you'll build a fillet between the stack and the body of

the submarine. For this one, a larger radius will work well. Select the *subStack* object and then **shift** select the *subBody* object.

10. In the **Circular Fillet Options** window, change the **Radius** to 1.0. Then click the **Apply** button.

11. The fillet object has a greater size, meaning that its reach extends farther from the area where the two original surfaces intersect (see Figure 5.11).

12. You can now create fillets for the tail objects as well. First, select the *subTail01* object, then **shift** select the *subBody* object. Change the **Radius** setting in the **Circular Fillet Options** window to 0.25, and then click the **Apply** button.

13. Repeat this procedure for the *subTail02* and *subTail03* objects.

14. The last step is to apply the same material to the fillet objects as is on the rest of the submarine parts (see Figure 5.12). In the **Outliner**, select all of the *circularFilletSurface* objects. Switch to the **Rendering** menu set. Click on **Lighting/Shading > Assign Existing Material > subPhong**.

15. Save your scene as **modeling\_fillet\_submarine.ma**.

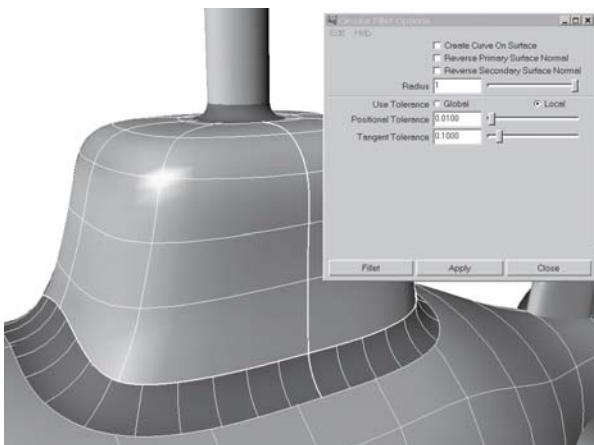


Figure 5.11: Stack fillet



Figure 5.12: Finished submarine

### 5.3 Animation – Motion Paths, Driven Keys, Added Attributes, Skeletons



You may sometimes find that there is a very specific route you'd like an animated object to take. For example, you may want to get a plane to fly a certain flight pattern (see Figure 5.13), a train to run along some tracks, or a grape to arc up in a loop and drop into your character's waiting mouth. These types of animations are often accomplished with the aid of a **Motion Path**.

#### Motion Paths

A **Motion Path** is a NURBS curve that has been called into special duty: to supply an object with translation and rotation values over time. Unlike the familiar "Time-Value-Key" method of animation, wherein you manually set the key positions for an object in flight, when you use a motion path, the object will automatically traverse the path over a specified amount of time.

The steps for using a motion path are straightforward: 1) create a NURBS curve, 2) select an object and the curve, and 3) click **Animate > Motion Paths > Attach to Motion Path**. If you were to play the animation back you'd see your object zipping along the curve, rotating to follow the curve's bends and turns. There are a number of options that give you fine control over the path animation. These serve to change the orientation of your object, the manner in which the object "banks" around turns, and the timing of the animation.



Figure 5.13: Motion path animation



Once your object is running down the path in the correct direction, the next most important thing is the timing. It's pretty simple to change the timing of a path animation. In the **Graph Editor**, the motion path animation is represented by a function curve. This curve graphs value over time, just as with “normal” keyframe animation. The value, in this case, refers to the beginning (0) and end (1) of the path. As usual, time represents frames along the timeline. So a motion path that has a 0 value keyframe at frame 1 and a 1 value keyframe at frame 144 is taking 6 seconds to go from start to finish.

You've probably already guessed that adding keyframes and altering tangency on the motion path's function curve will yield some neat results. For example, a fast-out spline tangency at the first keyframe will cause an object to shoot down the path. A series of keyframes that alternate between ascending and descending values will cause the object to move forward and then backward along the path. By placing a slow-in spline tangency at the last keyframe, the object will cushion into the end of the path.

### Animation Exercise – Pursuit Plane Path

In this exercise, we'll get the airplane to fly around the barn using a motion path.

1. Set your project to `understanding_maya\Chapter05\animation_plane_motion_path`.
2. Open scene `path_plane_begin.ma`.

In the scene you'll see the barn and airplane, plus a simple sky and ground environment. We want to create the flight path for the plane first.

3. Looking through the **top** view panel, use the CV curve tool to draw a path for the plane. Click **Create > CV Curve Tool**. Then click the points to define your path. The first point you click will be the beginning of the **Motion Path**. Press **Enter** on your keyboard when you're done (see Figure 5.14).

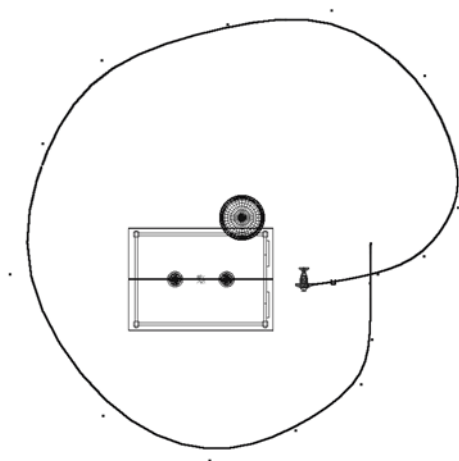


Figure 5.14: NURBS curve from top view

## 198 5. Wiring Things Up

---

The curve is flat on the ground at this point. This would be fine if the plane was supposed to taxi around, but we want to get it airborne! Let's edit the curve to give it some height.

4. Switch the curve you just created into component mode to edit the CVs. **RMB**-click on the *curve1* object to bring up the marking menu and choose **Control Vertex**. Reposition some of the CVs using the **Move Tool** until you like the new flight path (see Figure 5.15). Don't worry; you can still edit the curve after the plane has been attached to it.



Figure 5.15: Reshaped path curve

5. Press **F8** twice to return the curve to object mode.

The pursuit plane is a hierarchy made up of many individual parts. In order to successfully attach this group to the motion path, we'll need to pick the top node of the plane's hierarchy.

6. In the **Outliner**, select the top node of the plane hierarchy, named *pursuitPlaneGroup*.
7. In the **Outliner**, **ctrl**-select the *curve1* object.
8. Using the **Range Slider** controls, set the time line range to 1-350.
9. In the **Animation Menu Set**, click on **Animate > Motion Paths > Attach to Motion Path options** (see Figure 5.16).

The options allow us to control the duration of the flight, the direction the plane will face as it goes down the path, and the degree of banking.

10. Reset the options by clicking **Edit > Reset Settings**.
11. For the **Time Range** section, we'll choose **Time Slider**. This means that the plane will traverse the path over the entire range of your animation, 1-350.

12. Leave the **Follow** option checked on. This tells the plane how to orient itself as it moves along the path. In order for the plane's rotation to do what we expect, we need to choose which end of the plane points forward and which point points up.

In describing the rotation of an airplane on three axes, the terms **yaw**, **pitch**, and **roll** are used (see Figure 5.17).

If you're sitting in the cockpit, piloting the plane:

**Yaw** is the "left" and "right" rotation of the plane as you stay level with the ground.

**Pitch** is the "up" and "down" tilting of the plane to head toward the sky or the ground.

**Roll** is the "banking" of the plane as you "tip" the right wing toward the ground and the left at the sky (or vice versa.)

13. Currently, the plane's nose points along the Z-axis when all rotation axes are at zero. This is the setting we'll use for determining which axis to select. Click on **Z** for the **Front Axis**. (If we chose the default X-axis, the plane would fly sideways around the path.)

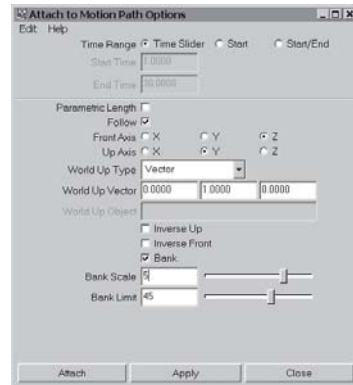


Figure 5.16: Motion path options

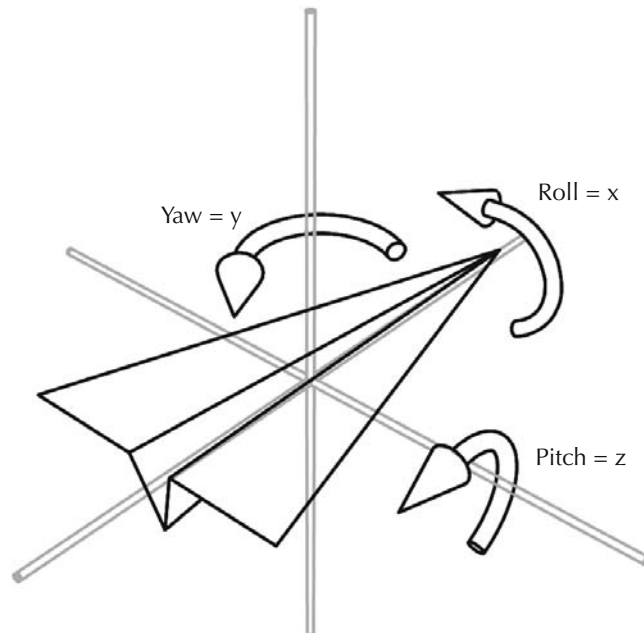


Figure 5.17: Yaw, pitch, and roll rotations

## 200 5. Wiring Things Up

---

14. The **Up Axis** should remain on **Y-axis** so that the top of the plane points up.
15. The **World Up** settings can remain at the default **Vector** and **0, 1, 0**. These settings tell the plane which way is up. In this case, Y is up since 0, 1, 0 stands for X, Y, Z.
16. **Inverse Up** and **Inverse Front** are simple ways to flip the plane's axes, in case you want to have it fly upside-down and backwards. (Or, more commonly, to correctly attach an object to a path when that object was built facing a negative axis.)
17. In airplane terminology, we've taken care of **yaw** and **pitch**, but what about **roll**? Turn on the **Bank** checkbox. Now you have access to the **Bank Scale** and **Bank Limit** sliders. Set the **Bank Scale** to 5 and the **Bank Limit** to 45. Later you can adjust these values to increase or decrease the amount of **roll** the plane has. Click the **Apply** button.
18. Play your animation. The plane has taken flight using your path.

The plane is now stuck firmly to the path, but that doesn't mean you can't alter the animation. The plane travels from the beginning to the end of the path over the course of your timeline by default. In order to alter that timing you need to change something called the **U Value** of the motion path. **U Value** is the normalized length of the curve; 0 is the start and 1 is the finish.

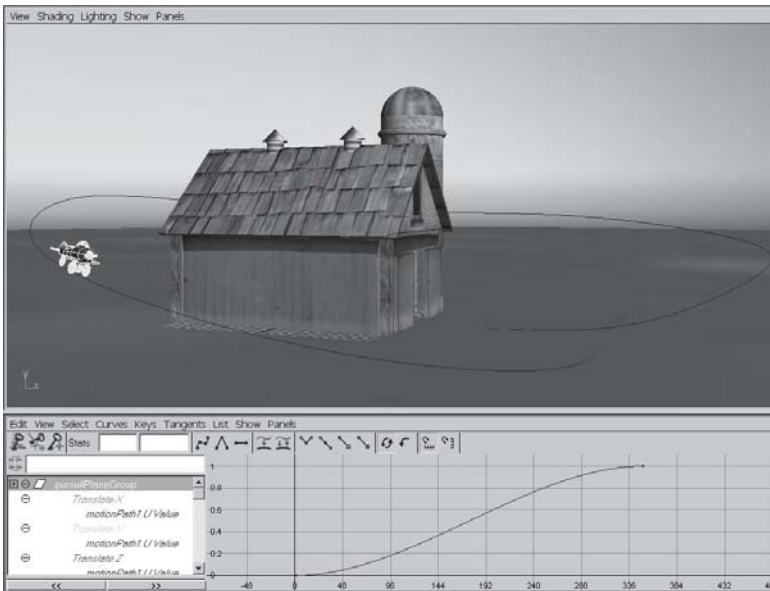


Figure 5.18: Motion path with ease-in and -out

Add keyframes to the **U Value** attribute to tune the path animation.

19. Select the *pursuitPlaneGroup* and then open the **Graph Editor**.

20. Notice that all of the translation and rotation values of the group are mapped to the same **U Value** input as in Figure 5.18. Select the second keyframe and lower its value on the vertical axis to 0.5. When you play the animation, the plane will only reach the path curve's halfway point.

21. Undo the keyframe value change. Now, using the **Add Keys Tool** button, select the **U Value** curve in the **Graph Editor**. Then, **MMB**-click to create six more keyframes.

22. Notice that each keyframe you add creates a corresponding locator on the motion path itself. Switch to the **move tool**. By selecting and then **MMB**-click-dragging the locators on the motion path itself, you can vary the timing. Alter the locator values so that you can get the plane to take off slowly, speed up, slow down, stall, speed up again, and then land gently (see Figure 5.19).

23. Save your scene.

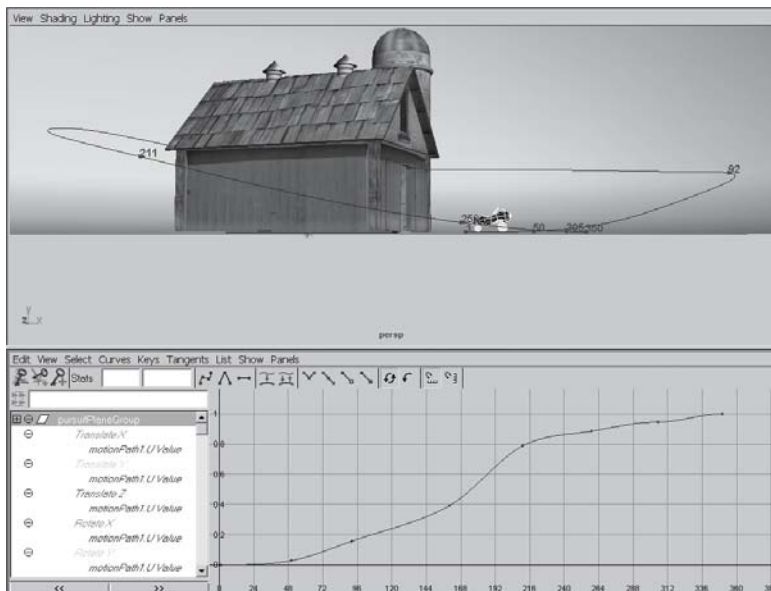


Figure 5.19: Finished motion path animation

## 202 5. Wiring Things Up

---

### Driven Keys

Maya's architecture – nodes with attributes that are connected – allows us to wire our own connections to simplify the creation of animation. For example, if you want a cause and effect behavior – such as flipping a switch to turn on a light – you can wire the rotation of your switch to the intensity of a point light (see Figure 5.20).

One way to wire such a connection is with a driven key. In the above example, the switch's rotation drives the light's intensity. The switch is the driver; the light is the driven.

The process for setting this relationship goes like this.

- a. Specify the switch's rotation as the driver node and channel.
- b. Specify the light's intensity as the driven node and channel.
- c. Rotate the switch to the “off” position.
- d. Lower the light's intensity to 0.
- e. Set a driven key for this relationship.
- f. Rotate the switch to the “on” position.
- g. Raise the light's intensity to 1.
- h. Set a driven key for this new relationship.

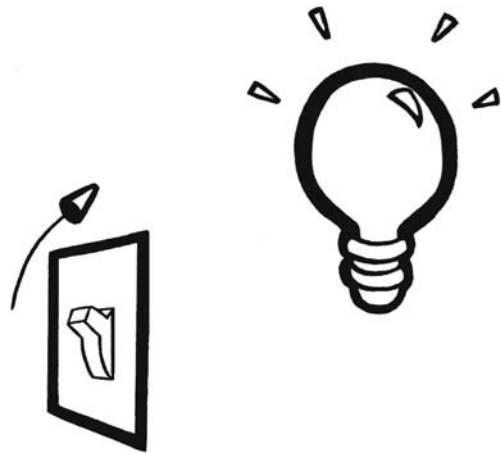


Figure 5.20: Driver (switch) and driven (bulb)

Now whenever a character comes into the room and flips the switch, the light will be driven to turn on. In other words, animating the switch rotation forces the light intensity to be animated as well. You don't need to worry about the timing of the light matching the timing of the switch. This is one huge benefit of using driven keys.

## Animation Exercise – Driven Keys

1. Set your project to `understanding_maya\Chapter06\animation_drivenkey`.
2. Open the scene `drivenkey_barndoors_begin.ma`.
3. Switch to the Animation menu set. To simplify selections, choose the **Select by hierarchies and combinations** pick mask button on the Status Line (see Figure 5.21).



Figure 5.21: Hierarchy selection mode

You'll be moving the plane into the barn. When the plane approaches the left door, it will slide open. After the plane is safely inside, the door will slide closed. Instead of setting keys on both the plane and the door, you will set up a driven key relationship. This will allow you to change the timing of the plane's entrance without worrying about the door. Just like a conditional if-then statement in a computer program ("if" this happens, "then" that happens), you can create any number of conditions within the relationship.

4. To establish the relationship between driver and driven, click on **Animate > Set Driven Key > Set options** (see Figure 5.22).
5. In the **perspective** panel, select the *pursuitPlaneGroup*.
6. In the **Set Driven Key** window, click the **Load Driver** button. The *pursuitPlaneGroup* node is loaded.
7. The left side of the **Set Driven Key** window indicates the driver or driven node, while the right side allows you to select the particular channels involved. Click on the `translateX` channel to select it. The driver is now established as `pursuitPlaneGroup.translateX`.
8. In the **perspective** panel, select the *barnDoor\_Left*.
9. In the **Set Driven Key** window, click the **Load Driven** button. The *barnDoor\_Left* node is loaded.

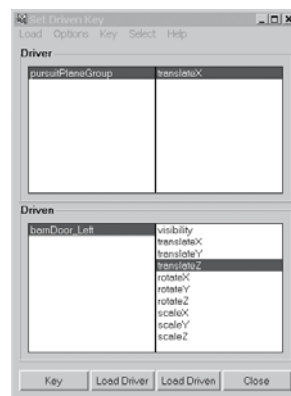


Figure 5.22: Driven key dialog

## 204 5. Wiring Things Up

---

10. You will want the door to slide on its Z-axis, so click on the translateZ channel in the Driven section. The driven is now established as `barnDoor_Left.translateZ`.

11. The specific values of how close the plane is when the door starts opening must now be chosen and keyframed. In the **perspective** panel, select the *pursuitPlaneGroup* and use the **Move Tool** to position it a small distance from the door. Translate X = 20 will do. Since the door is already in the closed position, your values are ready to be keyed (see Figure 5.23). In the **Set Driven Key** window, click the **Key** button.

12. Next, you will indicate where the plane will be when the door is fully open. Move the *pursuitPlaneGroup* closer, to about 16 on the X-axis. Now, slide the door open by moving it to 8 on the Z-axis (see Figure 5.24). Having provided the second set of “if” and “then” statements, click the **Key** button in the **Set Driven Key** window.

13. You’ll want the door to stay open while the plane enters the barn. Move the *pursuitPlaneGroup* in to about 8 on the X-axis. Leave the door’s translate X value alone, and press **Key**.

14. Lastly, move the *pursuitPlaneGroup* in to about 1 on the X-axis, slide the *barnDoor\_Left* closed, which is Translate Z = 2.5 (see Figure 5.25), and press **Key**.

15. Close the **Set Driven Key** window.

16. Now test things out. Select the *pursuitPlaneGroup* and move it in and out of the barn. As soon as the plane reaches the door, it will slide open. Once it clears the entrance, the door slides closed.

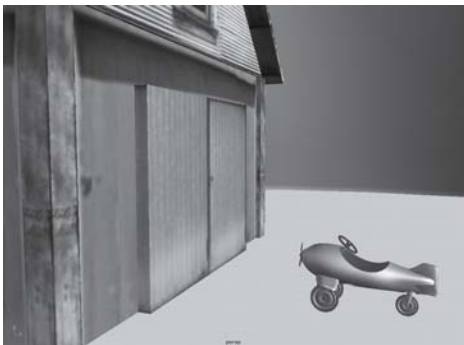


Figure 5.23: First driven key



Figure 5.24: Second driven key



17. Try animating the plane going into the barn. Even as you finesse the plane's timing in the **Graph Editor**, the door will always open and close properly.

18. Notice in the **Graph Editor** the relationship between the driver and the driven (see Figure 5.26). This is displayed whenever the driven object is selected. It is possible to edit driven key tangency just as you do in key-framed animation. You can experiment with this to create a relationship in which the door opens gradually, but stops opening abruptly.

19. Save your scene.

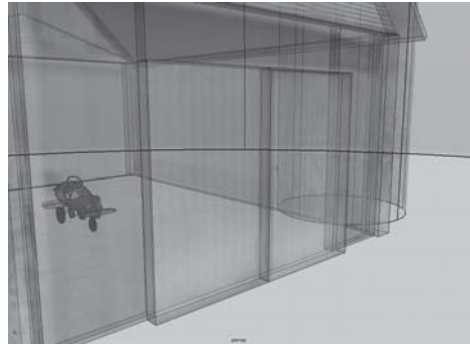


Figure 5.25: Final driven key

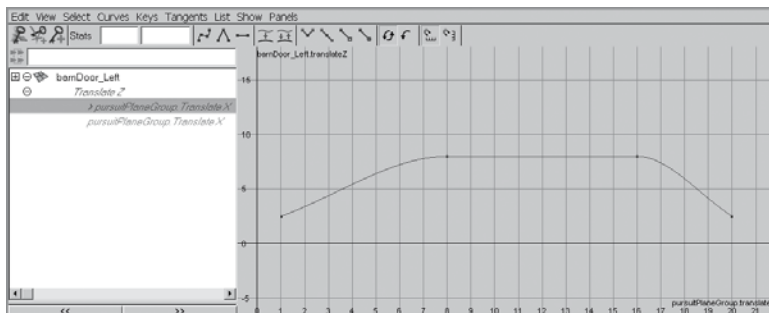


Figure 5.26: Driven key relationships in Graph Editor

## Added Attributes

Driven keys can be used to wire a relationship between added attributes on one object and the keyable attributes on another. This is sort of like creating a remote control. It's convenient to collect all your controls in one place; it's easier to select a character's left hand control and have access to all your finger curling attributes, than it is to pick each individual knuckle to rotate by hand. Let's take a look at setting up a remote control with added attributes.

### Animation Exercise – Added Attributes

1. Set your project to `understanding_maya\Chapter05\added_attributes`.
2. Open the scene `added_attributes_remote_begin.ma`.

In the scene you will find a remote control curve (see Figure 5.27), which has been parented to the perspective camera. (This is a simple way to add unobtrusive interface elements to Maya.) You will centralize all of the airplane controls on this remote object by adding attributes and setting driven keys.

3. Select the remote. (Make sure your selection mode is set to **object**, not **hierarchy**.) You won't need to transform it, so go to **Window > General Editors > Channel Controls...** and remove all keyable channels.
4. Next, click on **Modify > Add Attribute...**
5. In the **Attribute Name** field, type `Up Down`. Leave the **Data Type** at the default of **Float**. In the **Numeric Attribute Properties** section, set the **Minimum** to **0** and the **Maximum** to **25**. Click **Add** (see Figure 5.28).
6. Notice how there is now a channel in the **Channel Box** called `Up Down`.

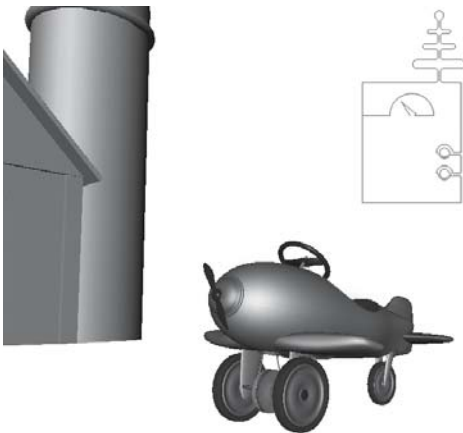


Figure 5.27: Remote plane interface

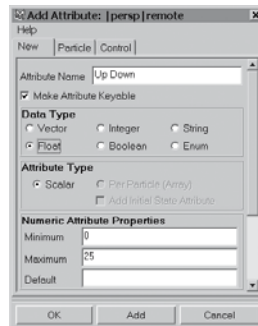


Figure 5.28: Adding attributes

7. Add the following additional attributes:

<b>Right Left:</b>	min = -10, max = 10,
<b>Fore Back:</b>	min = -10, max = 10,
<b>Pitch:</b>	min = -5, max = 5,
<b>Yaw:</b>	min = -20, max = 20,
<b>Roll:</b>	min = -10, max = 10.

You'll see these appear at the bottom of the **Attribute Editor** for the *remote*'s transform node as a series of sliders, as in Figure 5.29. (If they don't appear, be sure to click on the **remote** tab instead of the **remoteShape** tab at the top of the **Attribute Editor**.)

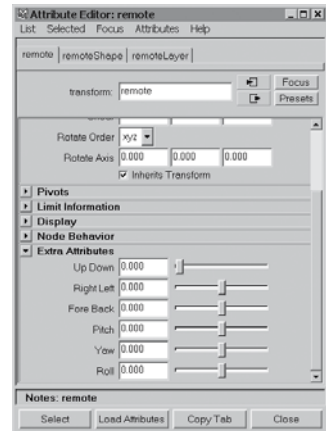


Figure 5.29: Added attributes

8. These added attributes are useless on their own.

They're keyable channels that aren't connected to anything at all. Sort of like taking a knob and sticking it to the front of a stereo. You can turn it left and right, but the volume isn't going to change until you wire things up. Both the **Connection Editor** and driven keys are valid ways to make such a connection.

9. The Translate Y, Translate Z, Rotate X, Rotate Y, and Rotate Z channels of the *pursuitPlaneGroup* can all be directly wired to the Up Down, Fore Back, Pitch, Yaw, and Roll channels of the *remote*. This is because there is no filtering required; an Up Down value of 10 equals a Translate Y value of 10, for example. Use the **Connection Editor** when you need no filtering of values.

10. Select the *remote*. Click on **Window > General Editors > Connection Editor...**

11. In the **Connection Editor** click **Reload Left** (see Figure 5.30).

12. Using the hierarchy pick mask, select the *pursuitPlaneGroup* in the perspective view. In the **Connection Editor** click the **Reload Right** button.

13. Highlight the **Up Down** channel on the left. On the right, open the **Translate** section and then highlight the **Translate Y** channel, as in Figure 5.30. This connects the two attributes.

14. Continue on in this way, making the following connections: **Fore Back** = Translate Z, **Pitch** = Rotate X, **Yaw** = Rotate Y, and **Roll** = Rotate Z.

## 208 5. Wiring Things Up

15. To wire the plane's Translate X channel, we can't make a direct connection, so close the **Connection Editor**. We need to filter the relationship between the remote's Right Left value and the plane's Translate X.

Filtering is used so that unlike values can work together harmoniously, for example, if you want a left turn of a cube to equate to a right turn of a cone; there must be a filter of some kind. A mathematical expression is one kind of filter that can be used. In this case, the expression would be: `cone.rotateY = -1 * cube.rotateY;`

In plain language: The cone's rotate Y channel is equal to the opposite of the cube's rotate Y channel.

16. In the case of the remote and the plane, the mismatch that needs filtering is between the plane's home position (Translate X = 40) and the remote's **Right Left** "home" position of zero. The expression that will correct this mismatch is: `pursuitPlaneGroup.tx = remote.RightLeft + 40;`

17. To enter this expression, click **Window > Animation Editors > Expression Editor...** In the field marked **Expression:** type the above expression, and then click the **Create** button (see Figure 5.31).

18. Now, close the **Expression Editor** window and try using the virtual **MMB** slider on the remote's added attributes. Go ahead and keyframe these values to make the plane hover, dip, and rotate a bit in midair before landing it.

19. Save your scene.

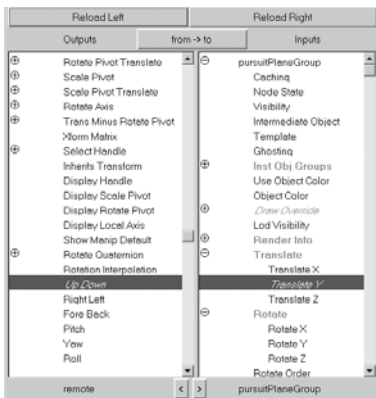


Figure 5.30: Connection editor

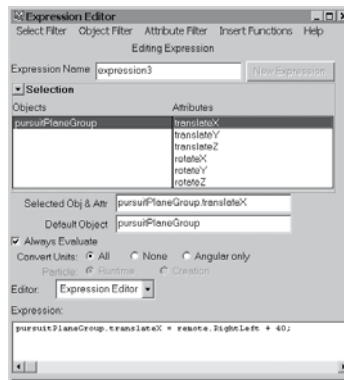


Figure 5.31: Expression editor

## Skeletons

Most characters are just a bag of bones that rotate. It's true. Think about your own body. The majority of your motion is achieved through the rotation of a bone around its pivot point: the joint. With the possible exception of trying to look impressive at the beach, your muscles are just there to rotate your joints. Character mechanics is largely a study in rotations. Due to the importance of joint rotation in movement, the foundation of character animation in Maya is the skeleton.

Using the **Joint Tool**, you create bones that are automatically connected to each other through joints, and then animate the skeleton's joint rotations much as you would the armature of a stop-motion puppet. Bones and joints are really just non-rendering helper nodes that generate a parent-child hierarchy when they are built. So when a parent joint rotates, all of the children bones and joints rotate as well.

Skeletons also serve a second purpose. In fact, take a moment to think about why we even use a skeleton inside a character model. What is the ultimate goal of character mechanics? It is to build a system that allows the manipulation of a character's many body parts. This system can run the gamut from pure simplicity to mind-numbing complexity.

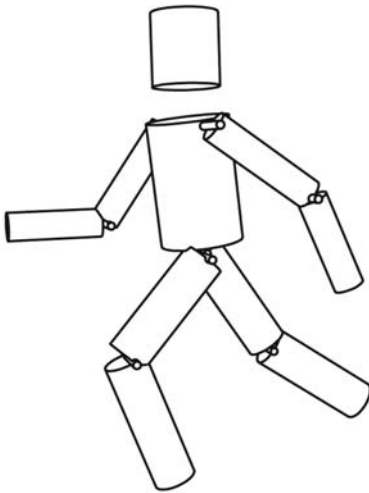


Figure 5.32: Simple character hierarchy

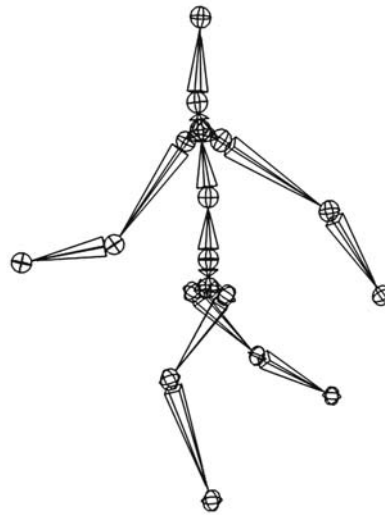


Figure 5.33: Joint hierarchy

## 210 5. Wiring Things Up

---

If the character in question is rigid and segmented, like an artist's mannequin or a robot, the system can be simple. Posing the character would be possible through parenting and then rotating the different limbs, appendages, and segments (see Figure 5.32). Try building a character entirely out of scaled NURBS cylinder primitives. (To avoid shearing, be sure to **Freeze transformations** before proceeding. To understand why this is necessary, think about our **'toon tire** exercise in Chapter 4.) By placing each cylinder's pivot point in a sensible place, and parenting each one to its logical parent cylinder, it is fairly easy to create a quick poseable character.

The skeleton provides a natural hierarchy with sensibly placed pivot points as well. Instead of moving pivot points, and carefully building the sphere-to-sphere hierarchy, you can just as easily create a skeleton for your cylinder man (see Figure 5.33), and parent each segment to a nearby joint. So, if the pivot points and hierarchy of the segmented character are this easy to create, why do we need skeletons? Well, the important other function they serve is that of deformation.

Not every character is composed of separate rigid segments. When you model a character with a limb, appendage, or section that spans across a bendable joint, you need a skeleton. Skeletons, when properly connected to geometry, are capable of deforming that geometry. To better understand skeletal deformation, think of it in terms of hierarchies. Instead of parenting an entire object to a single joint, we'd like to be able to parent some of an object's components (such as vertices or CVs) to one joint, and the remaining components to a different joint. In Maya, this is accomplished through **skin binding**.

For example, in order to bend at the elbow, a single arm model should have about half of its CVs associated with the shoulder joint, and the other half (those below the elbow) associated with the elbow joint. When the elbow joint rotates, the geometry will be appropriately deformed (see Figure 5.34).

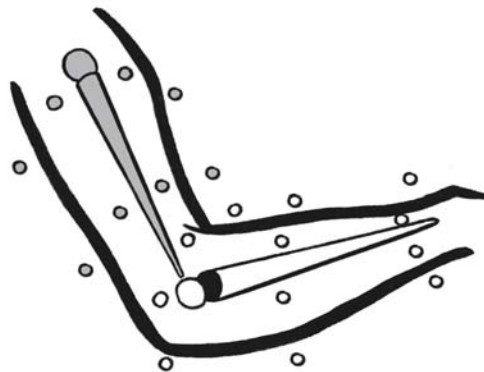


Figure 5.34: Mesh-to-skeleton skin binding

## Animation Setup Exercise – Chicken Skeleton

1. Set your project to `understanding_maya\Chapter05\animation_rigging_chicken`.
2. Open the scene `skeleton_chicken_begin.ma`.

In the scene you'll see a templated copy of the chicken model, which you will use as a guide for creating the skeleton (see Figure 5.35). The skeleton is a set of subhierarchies that all belong to one overall hierarchy. You can create each subhierarchy separately, and then parent them into the overall skeletal hierarchy later.

The placement of the joints is the first thing to consider when building the skeleton. Joint placement determines the points around which the character's limbs will swing, the spine will rotate, and the toes will curl, and so on. Placing joints can be tricky, however. Once you've started to build a joint chain, it can be problematic to reposition joints. The orientation of each joint is very important; moving joints around tends to wreak havoc on their orientation.



Figure 5.35: Mesh template

One way to approach joint placement, while avoiding this orientation problem, is to start out with locators, instead of joints. Begin by placing locators at all of your joint locations. You may want to scale them up a bit to make them easier to see. When you're satisfied with their positions, you'll snap joints to those locators.

3. Viewing the scene from the **side** view panel (working in the **side** view will ensure that these locators are placed at the exact center line of the model), click **Create > Locator** and move the locator to the character's center of gravity. This is the origin point for the entire skeleton you'll be making, also referred to as the "root."
4. Place two more locators, one above and one below the root locator (see Figure 5.36).
5. Continue placing five more locators to define the spine (see Figure 5.37).
6. Carefully place a locator at the neck, the base of the skull, the top of the skull, and the end of the chicken's comb (see Figure 5.38).

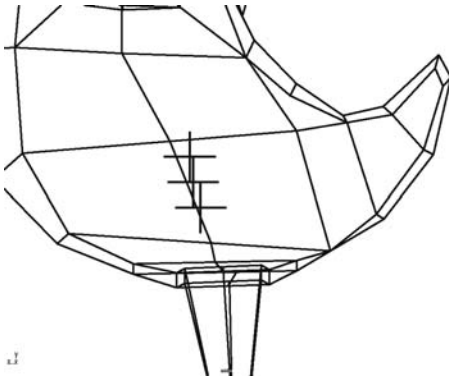


Figure 5.36: Root locator placement



Figure 5.37: Spine locators

7. Later, when binding the chicken to your skeleton, it may be helpful to have an upper and lower beak bone. To accommodate this, create a pair of locators at the base and tip of the upper beak. Repeat this for the lower beak (see Figure 5.38).

8. For tail articulation, you'll want to add three locators at the base, middle, and tip of the tail.

It isn't necessary to manually create both the left and right limbs at this stage. After the joints and bones are in place on the character's left side, we'll use Maya's **Mirror Joint** command to automatically generate the right side. This is a great benefit of working with symmetrically designed characters.

9. Still working from the side, place locators for the top of the thigh, knee, ankle, and ball of the foot (see Figure 5.39). These will initially be placed at the center line of the model, and then moved to align with the chicken's left leg from the **front** view panel. For now, just make your adjustments on the Y- and Z-axes.

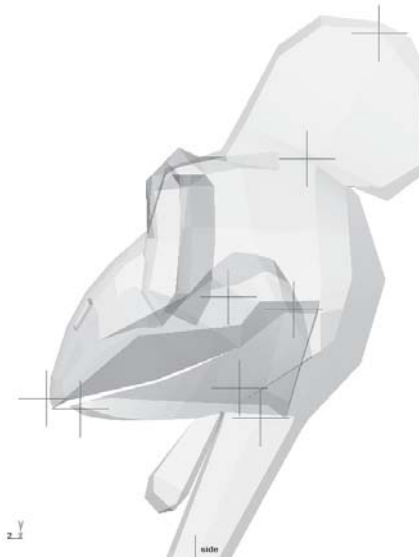


Figure 5.38: Comb and beak locators



10. Place three locators at the base of the toe, first knuckle, base of the claw, and end of the claw (see Figure 5.39). This is a general placement guide worked out from the **side** view; once we move to the other views, we'll create duplicate sets for each toe.

11. For the talon at the back of the foot, place two locators, one at the base and one at the tip. This is a more simplified setup than the front toes (see Figure 5.39).

12. Switch to the **front** view panel to line up the leg locators with the character's left leg geometry. Center the foot and toe locators within their respective geometry.

13. Working in all four standard views, duplicate the middle toe locators and re-align them with the big toe geometry. Then, repeat this for the little toe (see Figure 5.40).

14. The final joint placements to create are for the left wing. We'll approach the chicken's wing much like a human arm. In the **front** view panel, create a locator at the collar bone level. (This will be the base joint pivot for things like shoulder shrugging, so it shouldn't be too far from the spine on the X-axis.)

15. Next, create locators for the shoulder, elbow, and wrist.

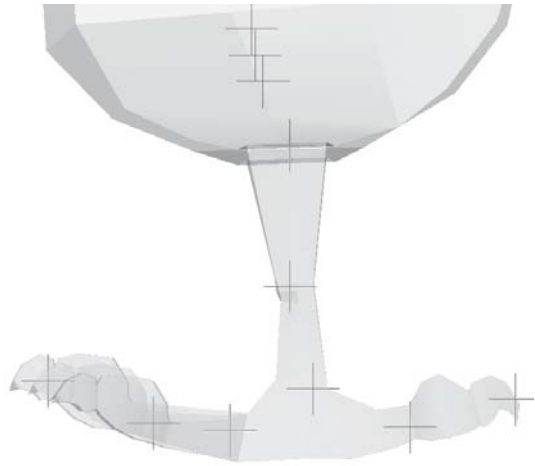


Figure 5.39: Leg and toe locators

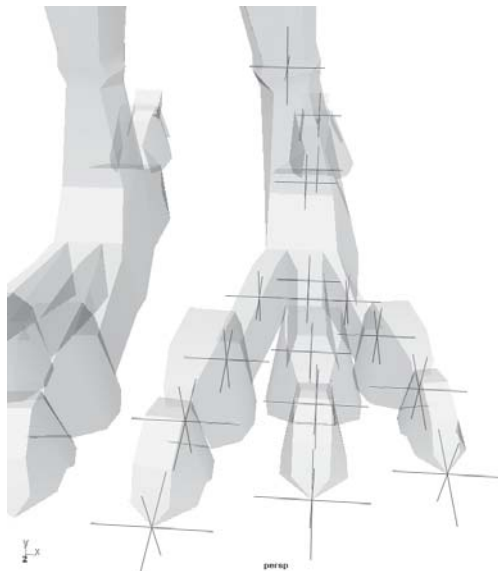


Figure 5.40: Toe locators

## 214 5. Wiring Things Up

---

16. Again, using all four standard views, center the new locators inside their geometry (see Figure 5.41).

17. Similar to the toe setup, you will now create finger joint locators. Start with a palm locator and then add four locators to each finger – base, middle, head, and tip.

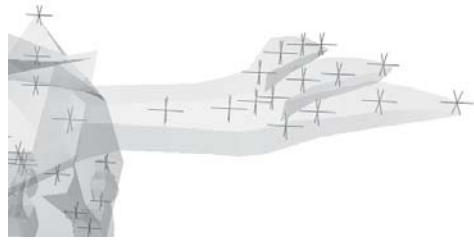


Figure 5.41: Wing locators

18. Be sure to check out your joint placements in all views, including the **perspective** camera to line things up.

19. Name the locators according to the names of the locators on the *jointpositionLayer*. This will also serve as the naming guide for the joints.

At this point, you should save your scene. We'll begin building the skeleton now, snapping joints to the locators.

20. In the **Animation** menu set, Click on **Skeleton > Joint Tool** options (see Figure 5.42). Click the **Reset Tool** button.

a. **Degrees of Freedom** are set to allow each joint to rotate freely on all three axes. This is a good default setting; while it's possible to lock some axes of rotation on individual joints, this can lead to problems during animation, such as gimbal lock.

b. **Auto Joint Orient** is set to **xyz** by default. This specifies the local rotate orientation of each joint. With this setting turned on, the X-axis of each joint will point at its child, making it much more natural to control. The other settings can also be left at their defaults. Close the **Tool Settings** window.

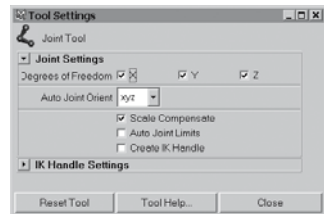


Figure 5.42: Joint settings

21. To fit the scale of the scene, we'll want to view the joints at a smaller size than the default. Click **Display > Joint Size > 25%**.

22. In the **side** view panel, begin by creating the spine joints. First, make sure the **Snap to points** button is turned on, which will make it easy to place each joint exactly on top of the locators we've positioned. You will click the **LMB** on each of the locators, starting

with the *lumbar\_locator*. When you reach the *combTip\_locator*, press **Enter** to complete the tool (see Figure 5.43).

Ten joints is a mid-range number for the spine and neck – you can get away with many fewer or many more, depending on the flexibility and range of motion your character has. In forward kinematic animation, if your skeleton has an inordinately large number of joints, posing the character can become cumbersome. It's a good idea to strive for efficiency in skeletal design.

23. Naming joints is an important part of skeletal construction. An **Outliner** full of nodes called *joint23*, *joint24*, *joint25*, and so on, can be confusing – instead, your nodes should have descriptive names. Name the joints based on their corresponding locators, with the “\_jt” suffix instead of “\_locator.”



Figure 5.43: Joint snapping

24. Even with this moderate number of spinal joints, during animation you will likely want to rotate several joints simultaneously to speed up the posing process. For this to work, it will be necessary for the joints to be oriented in the same direction. Currently, each spinal joint is aligned with its local X-axis pointing down the length of its child bone. This is due to the **Auto-orientation** option that was set in the **Joint Tool**. However, the local Z-axes are not in agreement from joint to joint. To see the problem, try this:

- a. Go to the **Outliner** and select *lumbar\_jt* and then shift-select *neck\_jt*.
  - b. In the **side** view panel, use the **Rotate Tool** and rotate the selected joints on the Z-axis. Notice how the lumbar region curls up in one direction, while some of the thoracic and cervical joints rotate in the opposite direction. In other words, positive Z-axis rotation means different things to different joints in the spinal column.
  - c. Undo any rotations you have applied to the spinal joints.
25. By unifying the local rotation axes in the spine, you will be able to rotate all the joints simultaneously.

## 216 5. Wiring Things Up

---

a. First, select *lumbar\_jt* in the **Outliner**, and then shift-LMB select *combBase\_jt*, which will select the entire spine hierarchy.

b. To display the local rotation axes on the selected joints, click **Display > Component Display > Local Rotation Axes**.

c. You should see the X-axis aligned with the bones, while there is no agreement about whether the Z-axis points left or right. We'll solve this by rotating some of the axes so that the Y-axes all point out of the back of the spine (see Figure 5.44).

d. Select *thoracic1\_jt*, and then switch to **component mode**. Turn off all pick masks, then turn on the **miscellaneous** pick mask, which contains the **Local Rotation Axes** component. Select *thoracic1\_jt*'s local rotation axis.

e. We want to rotate each axis precisely, so we'll use a MEL (Maya Embedded Language) command that adds 90° of relative X-axis rotation each time it is invoked. In the **Command Line**, type the following MEL command: `rotate -relative -objectSpace 90 0 0;` and then press **Enter**. You will see the local rotation axis turn 90°.

f. LMB-click in the **Command Line**, and then press the up-arrow on your keyboard. This calls up the most recently typed MEL command. Press **Enter** and the command will be repeated.

g. Your *thoracic1\_jt* local rotation axis is now correct. Now you'll repeat this process for each joint in the spinal hierarchy that doesn't have its Y-axis pointing out (see Figure 5.44). To take some of the tedium out of the workflow, we'll create a **Shelf Button** for the 90° rotation command. You can then simply select a joint's local rotation axis and click the button as many times as you need in order to align it.

h. Once again, click in the **Command Line**, and press the up-arrow on your keyboard. Highlight the entire command by triple-LMB-clicking. Now, MMB-click-drag the command up to an empty spot on the **Shelf**. You just created a clickable

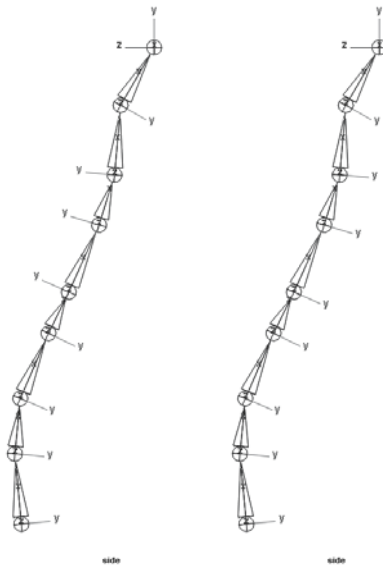


Figure 5.44: Bad vs. good orientation

MEL button for fixing local rotation axes.

i. Test it by selecting the *thoracic1\_jt* local rotation axis (be sure that you are still in component mode), and then click on the new **Shelf** button. Each time you press it, the axis rotates 90°. Undo this rotation to get back to the proper alignment.

j. Repeat this process for any other spinal hierarchy joints necessary, so that they all have the Y-axis facing out of the back.

26. Now that the joints are all aligned properly, switch back to object mode. In the **Outliner**, select all of the joints in your spine hierarchy except for *combTip\_jt*. Use the **Rotate Tool** to curl the spine up on the Z-axis. This time you should see the joint rotations behave uniformly (see Figure 5.45). Undo when you are finished testing rotation to get back to the original rotation. (If needed, you can type 0 into the **channel box** for the rotations and press **Enter**.)

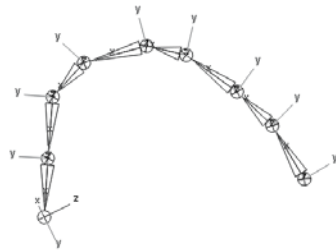


Figure 5.45: Proper rotations

27. Select all the spinal hierarchy joints in the **Outliner** (from *lumbar\_jt* to *combBase\_jt*) and then click **Display > Component Display > Local Rotation Axes** to toggle their visibility off.

28. Save the scene as **skeleton\_chicken\_mid01.ma**.

29. Using the **Joint Tool**, snap to the beak locators to create the two beak joints.

30. Rename the beak joints according to the naming in the *jointpositionLayer*, and then parent *beakBase\_jt* to the *skull\_jt*. (To do this, select *beakBase\_jt* first, shift-select *skull\_jt* second, and then choose **Edit > Parent**.)

31. Using the same method as above, create the jaw joints, *jawBase\_jt* and *jawTip\_jt*.

32. The leg hierarchy is next. In the **side** view, with the templated joints as a guide, use the **Joint Tool** to create eight joints. Start with the hip and end with the toe.

33. Name the joints *hip\_jt*, *knee\_jt*, *ankle\_jt*, *ball\_jt*, *midt0e4\_jt*, *midt0e3\_jt*, *midt0e2\_jt*, and *midt0e1\_jt*.

34. Create the separate big, pinky, and talon hierarchies by snapping to their respective locators (see Figure 5.46).

## 218 5. Wiring Things Up

---

35. Rename the toe joints according to the *jointpositionLayer*.

36. Parent each of the toes to the main foot by shift-selecting *bigtoe4\_jt*, *pnktoe4\_jt*, and lastly *ball\_jt*; then click **Edit > Parent**. Parent *tlntoe2\_jt* to *ankle\_jt*.

37. Select the *ball\_jt* and flip its local rotation axis in component mode, using the MEL button you created. Unify all of the other toe joints so that they all have their Y-axis pointing up (see Figure 5.46).

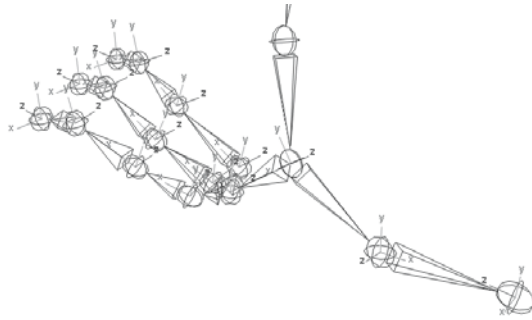


Figure 5.46: Toe joint orientation

Notice that we've created the leg and arm joints with generic names (no mention of left and right). Later, you'll duplicate the left leg and move the copy to the other side of the body. It's a good workflow practice to build symmetrical parts only once and then mirror them. Once the hierarchies exist on both sides, we'll rename them to designate left and right.

The entire skeletal structure will ultimately be a single hierarchy with the *root\_jt* at the top of it. Notice how all bones are tapered, pointing from their parent joints toward the children joints. Since the *root\_jt* will have no parents, only children, all bones will radiate away from it. There can be only one root in a hierarchy. In order to delineate the lower and upper skeletal subhierarchies, we'll create a subroot joint for each of them, very near the true root.

38. Use the **Joint Tool** to create a single joint, snapped to the *root\_locator*. Press **Enter** to complete the tool. Rename this joint *root\_jt*. By creating single, free-floating joints, we'll ensure that they're aligned with the world's rotation axes. This will help if you need to twist the hips, turn the whole character, or turn only the shoulders.

39. Two more joints: just below the *root\_locator*, create a single joint snapped to the *pelvis\_locator*. Rename this joint *pelvis\_jt*. Just above the root, create the *sacrum\_jt*.

40. To form the hierarchy properly, select the *pelvis\_jt* and *sacrum\_jt* and parent them to the *root\_jt*. You should see two new bones appear, both pointing from the *root\_jt* outward. Remember, bones always indicate the direction of hierarchical rank. Parents point toward their children.

41. Now let's parent the subhierarchies into the main skeleton. Select the *hip\_jt* joint and parent it to the *pelvis\_jt*. Select the *lumbar\_jt* and parent it to the *sacrum\_jt*.

42. Switch to the **perspective** view panel. The skeleton only has one leg at the moment; let's fix that. Select the *hip\_jt*, then click **Skeleton > Mirror Joint options**. Reset the tool settings, and then click on the **YZ** option for the **Mirror Across** plane. Click the **Mirror** button. A new leg subhierarchy is created on the right side of the body, and it is automatically parented to the pelvis.

Maya doesn't like to give a duplicate name to the top node in a hierarchy. You will notice that the *hip\_jt* duplicate is named *hip\_jt1*. You will now give both legs descriptive names to help differentiate left from right.

43. Select *hip\_jt* in the **Outliner**, and then shift-click the plus sign next to it. This expands the entire hierarchy, like a list of subdirectories or folders. Instead of manually adding a "left" or "right" designator to each joint, we'll automate the process. With *hip\_jt* still selected, click on **Modify > Prefix Hierarchy Names...**

44. In the **Enter Prefix** field, delete the default text and type *L\_*, then click **OK**.

45. Look at the **Outliner** to see that all the joints in the left leg have had the prefix added.

46. Repeat this process for the right leg, using the *R\_* prefix.

47. Save your scene, just to be on the safe side.

48. Now, you'll create the left arm hierarchy. Using the **Joint Tool**, snap the joints from the *clavicle\_locator* to the *midfinger1\_locator*.

49. Rename your joints according to the *jointpositionLayer*; (see Figure 5.47) then parent the *clavicle\_jt* to the *thoracic4\_jt*.

50. Adjust the joints' local rotation axes in component mode so that the Z-axis of each arm joint points up. Remember to use your MEL shelf button.

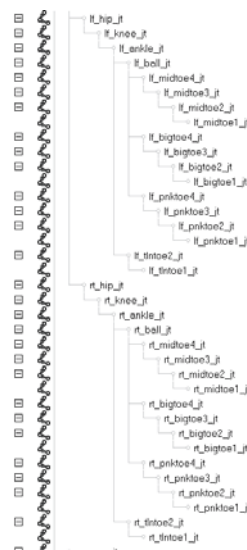


Figure 5.47: Name prefixes

## 220 5. Wiring Things Up

---

51. Create separate hierarchies for the remaining fingers, snapping to the locators as you go. Rename them according to the *jointpositionLayer*.

52. The finger local rotation axes should be unified, so they have upward-pointing Y-axes.

53. Parent the new *pnkfinger4\_jt* and the *bigfinger4\_jt* to the *palm\_jt*.

54. Select the *clavicle\_jt* and use the **Mirror Joint** command to create the right arm. Then rename both arms using the same **Prefix Hierarchy Names** command as before.

Your chicken skeleton is now complete (see Figure 5.48). Be sure to save the scene! In the next section we'll learn how to pose the skeleton for animation.



Figure 5.48: Final skeleton

### Posing the Skeleton

The chicken skeleton can be posed and animated just like a stop motion puppet. In the next chapter we'll attach the chicken geometry to its skeleton and refine some of the puppeteering controls. Right now, try setting the skeleton into poses using only the **Rotation Tool**.

As you test out the skeleton's range of motion you'll notice that it's easier to begin a pose by rotating joints at the center of the body first. Then, when you have these joints posed, you can move outward, adjusting joints farther down in the hierarchy. This is due to the fact that in a forward kinematics skeleton, parents affect children, but not vice versa. For example, rotating the wrist joint will not throw off the pose you have set on the elbow and shoulder.

When you animate this skeleton, you'll want a simple way to set a keyframe for every joint in the hierarchy each time you work out a pose. This way, you can set the pose one joint at a time, key the entire pose with one button, and move to a new point on the timeline to set the next pose. The way we'll create this keyframe button is with Maya's scripting language, MEL.

Everything you can do with Maya using the graphical interface has an equivalent MEL command. By typing the command `cone` in the command line, for example, you can create a



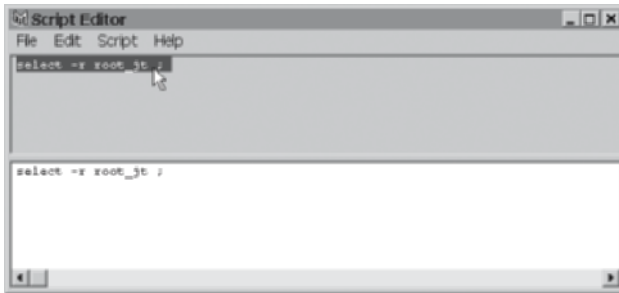


Figure 5.49: Script editor window

NURBS cone just as if you had clicked **Create > NURBS Primitives > Cone**. MEL scripts can range from simple workflow enhancers to sophisticated programs that add greatly to Maya's functionality (this is one reason Maya is popular at large effects and animation facilities).

A great benefit of using MEL instead of the graphical interface is that you can combine many commands into a single **script** that can be executed with the click of a button. For posing and animating the chicken skeleton, you may want a number of MEL buttons to make your work easier. You can create a button that will select all of the finger joints on the left hand, so you can curl them into a fist without making a multiple selection. Also, you can create a button that resets the skeleton to a neutral stance. If you make a pose you like and plan on reusing it during the animation, you can create a button that strikes that particular pose. Since we may need to key almost sixty different joints each time the character has been posed, it makes sense to have a single MEL button that takes care of it. Animator Philip Pignotti used this type of shelf button interface while creating the animation seen in Color Plate IV.

### Animation Exercise – Keyframing the Skeleton with MEL

1. Set your project to `understanding_maya\Chapter05\animation_posing`.
2. Open the scene `posing_skeleton_begin.ma`.

The first thing we want is an easy way to select every joint in the hierarchy. It will be the foundation of some of our other posing scripts.

3. Open the **Script Editor** by clicking **Window > General Editors > Script Editor**. While watching the upper area of the **Script Editor**, click on the `root_jt` in the **perspec-**

## 222 5. Wiring Things Up

---

tive view panel. Notice how the command `select -r root_jt;` is echoed back at you. This is one of the best ways to learn about MEL commands - by watching the **Script Editor** as you perform actions in Maya. Each line of MEL code will end with a semicolon (;).

4. Highlight the line `select -r root_jt;` in the upper area and **MMB**-drag it to the lower input area of the **Script Editor** (see Figure 5.49). The input area is where you will construct your MEL scripts.

5. Click **Edit > Select Hierarchy** in the main menu. This selects all the joints that are children of the `root_jt`. But wait! Why didn't anything change in the **Script Editor's** history area? Not every command is echoed by default. To see all commands, click **Script > Echo All Commands** in the **Script Editor**.

6. Try clicking **Edit > Select Hierarchy** in the main menu again. This time, a number of lines show up in the **Script Editor**. Not everything that appears in the history area is necessary to run a command. It sometimes takes trial and error to learn what parts you need in your scripts. In this case, `select -hierarchy;` will work. **MMB**-drag this line into the bottom area, making sure to put each line of code on a separate line.

It's a good idea to test your scripts a little bit at a time, instead of waiting for the whole thing to be written. This makes it easier to solve problems as you go.

7. First, clear your selection by clicking on empty space in the **perspective** view.

8. To execute this script, highlight the entire text and then click the numeric keypad **Enter** key on your keyboard. (There are a number of other ways, including **Script > Execute**, and **ctrl-Enter**, but those methods will clear your input area. On keyboards that lack a numeric keypad, you may be forced to reselect and **MMB**-drag your script back into the input area.) If you check in the **Outliner**, you can see that your script works; it has picked the root and every joint below it in the hierarchy.

9. To make this skeleton selection script more accessible, we'll turn it into a shelf button. First, click the down arrow to the left of the shelves, choose **New Shelf**, and name the shelf `chicken_controls`.

10. Now, highlight the entire script in the input area, and **MMB**-drag it up to the `chicken_controls` shelf. Now, any time you click the skeleton select button, you will grab the entire hierarchy.

11. In order to make it easier to distinguish your shelf button, let's add a label to it. Click **Window > Settings/Preferences > Shelves...** to open the **Shelf Editor**. In the tab

labeled **Shelf Contents** make sure that your new button is selected in the upper area. In the **Icon Name** field, type Hi. This will appear at the bottom of the MEL button on your shelf. Click the button labeled **Save All Shelves**.

12. When you're posing a character and don't like the direction things are heading, it's often easiest to reset to a zeroed-out pose and start again. For this, we'll create another MEL button. This button will work like this:

```
//Select the root joint
select -r root_jt;
//Select the hierarchy
select -hi;
//Rotate everything to zero
rotate -os 0 0 0;
//Deselect everything
select -clear;
```

Notice in the above script how things are formatted and spelled. If you enter the text exactly as written into the **Script Editor** it will work. What about the plain language descriptions? These are called comments and Maya knows to ignore them because they are preceded by double forward slashes (//).

13. Once you've typed in the above script, highlight it and **MMB**-drag it to the **chicken\_control** shelf. In the **Shelf Editor**, name this button "rst" (for reset).

14. To set up a quick animation workflow, you'll want to script a button that keys the pose with a single click. Select the root joint and then use the **Set Key options** to key the hierarchy below it. Reset the **Set Key Options**, and then choose:

- a. **Set Keys on All Keyable Attributes** – this keys all unlocked channels;
- b. Change **Hierarchy** to **Below** – this keys the selected object and its children;
- c. Uncheck **Shapes** – you only want to key the transform node of a joint.

15. Press the **Set Key** button. By watching the **Script Editor**, you can capture the single line necessary to repeat this in a MEL button. (See Figure 5.50.) **MMB**-drag the line to the shelf. Then, rename the button "Key." Now, each time you pose the skeleton, just press the "Key" button.

## 224 5. Wiring Things Up

---

16. To quickly delete all the character's keyframes at a particular point in time, you should create the following button.

```
// Select the root
select -r root_jt;
// Select the hierarchy
select -hi;
//Delete the keys for the current time
timeSliderClearKey;
//Deselect everything
select -clear;
```

17. Create a button for this and name it del (see Figure 5.51).

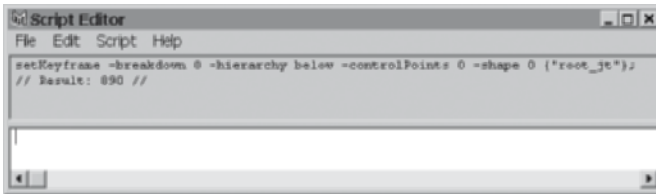


Figure 5.50: Keyframe script



Figure 5.51: Shelf

Now that you've set up the interface, it's time to animate. (Normally, you wouldn't animate the skeleton by itself – you would first attach the character's geometry. We'll be covering that aspect of character setup in Chapter 6.)

Posing a forward kinematic skeleton is all about joint rotations. Like a stop-motion animator, you'll rotate the spine and limb joints one by one until the pose looks right. It can be helpful to start with the joints nearest to the character's root, and then work your way outward.

18. Animate a simple scene of the chicken pecking at the ground. You'll follow the familiar animation steps of time – value – key (see Figures 5.52 - 5.58).

You can use the images and timing chart as a guide.

- frame 1 neutral;
- frame 10 arms tucked, bent;
- frame 14 rear back in anticipation;
- frame 18 down, with the head tipped back;
- frame 20 up slightly, to gulp some seeds;
- frame 22 down again;
- frame 24 up;
- frame 26 down;
- frame 29 up higher;
- frame 33 anticipate down (gulp);
- frame 38 reset to start pose, arms tucked.

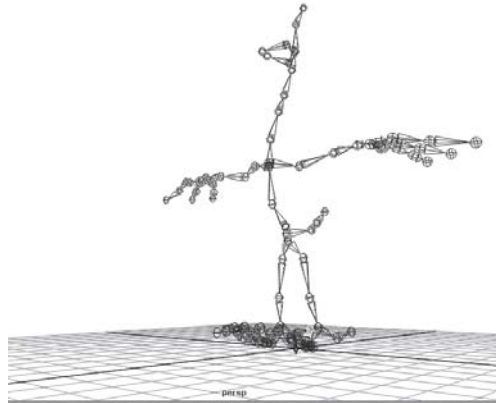


Figure 5.52: Frame 1 neutral

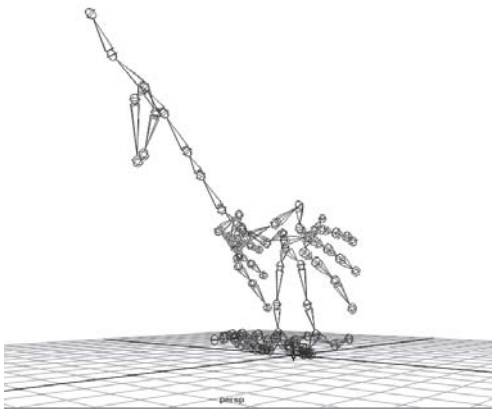


Figure 5.53: Frame 10 arms tucked

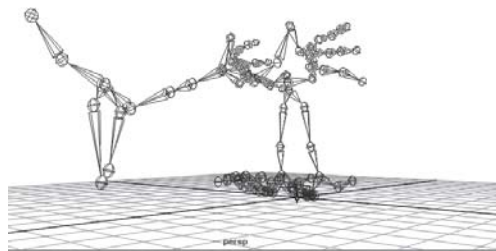


Figure 5.54: Frame 18 down

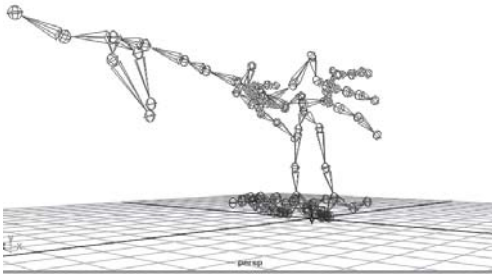


Figure 5.55: Frame 22 gulp

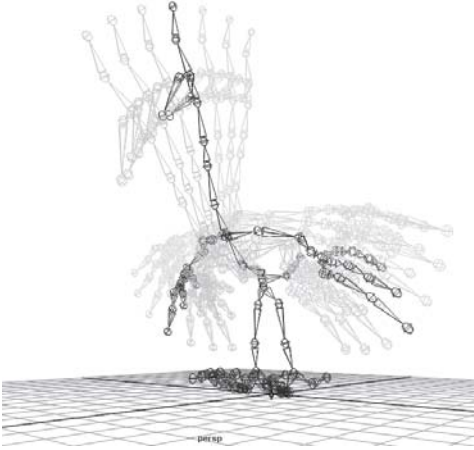


Figure 5.56: Time lapse

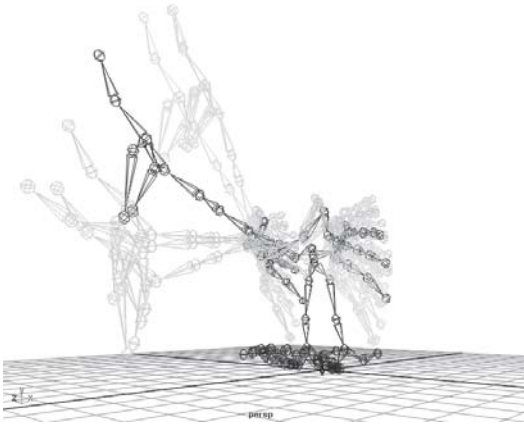


Figure 5.57: Time lapse

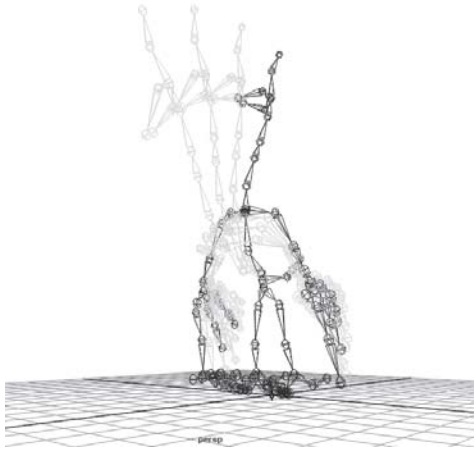


Figure 5.58: Time lapse

## 5.4 Shading – Displacement and UV Texture Mapping



In the shading section of Chapter 4 we looked at bump maps, a way to simulate details on the surface of an object. While they generally do a great job, under enough scrutiny the illusion is broken and a bump map alone may not suffice. In these cases, a displacement map may be the answer.

For example, a golf ball, the edge of a quarter, or a tile floor will look acceptable up to a certain point using only a bump map. However, once the camera zooms in close enough, the lack of geometric detail in the model will become evident.

### Displacement

Displacement maps really occupy a place somewhere between shading and modeling techniques. By using a displacement map on a surface's shader the Maya renderer tessellates the geometry enough to physically displace it. The actual mesh is altered during the render pass, unlike a bump map's surface normal trickery, so you end up with more accurate results. Be warned that this extra calculation and resultant geometry can greatly lengthen the time it takes to render a frame.

In practice, a grayscale file is generated with an image-processing application such as Photoshop or Painter, and then applied to the Displacement Map attribute of a shader. The pixels on the file that are white represent the highest "peaks" of displacement and the black pixels represent zero displacement.

### Shading Exercise – Displacement Wing

In this exercise we'll create displaced geometry on the plane's wing to give it a more detailed appearance.

1. Set your project to `understanding_maya\Chapter05\shading_displacement`.
2. Open the scene `displacement_wing_begin.ma`.

## 228 5. Wiring Things Up

---

You should see our airplane from previous exercises. We'll focus on creating a new shader for the wing, but these techniques can be applied to many of the plane's surfaces. A Blinn material type is useful when creating a metal look, so we'll start with that.

We'll be too close to the wing in these renders to rely on painted color maps alone for the rivets and metal plate details. Bump maps would work up to a certain distance, but under extreme close-ups, we'll need the true geometry created by the displacement maps.

3. In the **Hypershade**, create a **Blinn** material and name it *wingBlinn*.
4. Apply the *wingBlinn* material to the *wingLeft* object and then render the current frame. In the **Render View** window (see Figure 5.59).

We'll try out a bump map first. Since they're much quicker to render than displacement maps, it's generally a good idea to try the bump before resorting to a displacement.



Figure 5.59: Non-displaced wing



5. Click the **Show Top and Bottom Tabs** button in the **Hypershade**. Then create a **File** texture node with the default setting. Rename this node *fileRivets*.
6. With the new *fileRivets* node selected, in the **Attribute Editor**, click the file browse icon and choose the **rivets.tga** file. In this file, the brightest pixels will represent the highest portions of the bump and displacement.
7. **MMB**-click-drag both the *fileRivets* node and the *wingBlinn* node into the **Work Area** of the **Hypershade**. Then, **MMB**-click-drag the *fileRivets* node over the *wingBlinn* node, choosing the **Bump Map** connection from the dropdown list (see Figure 5.60).

8. Render the frame and inspect the bump mapping effect. Notice how it disappears at the edges of the wing (see Figure 5.61). One problem with bump maps is that they are ineffective at the horizons of your objects.

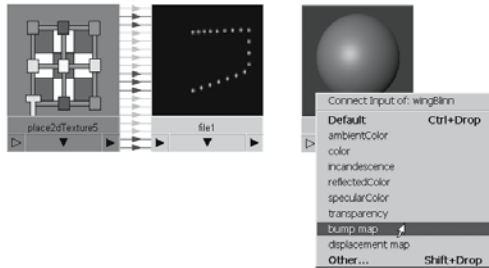


Figure 5.60: Bump mapping

9. Go back to the **Work Area** of the **Hypershade**. Here, you can break the connection between *fileRivets* and *bump2d*, by clicking on the line that connects them and then pressing the **Delete** key on the keyboard.

10. Next, highlight and delete the *bump2d* node that is connected to the *wingBlinn*.

11. This time, we'll **MMB**-click-drag the *fileRivets* node onto the *wingBlinn* node and choose **Displacement Map** from the list (see Figure 5.62). This connection is actually made farther down the shading network, so the lines won't be created between the two nodes we



Figure 5.61: Rendered bump map

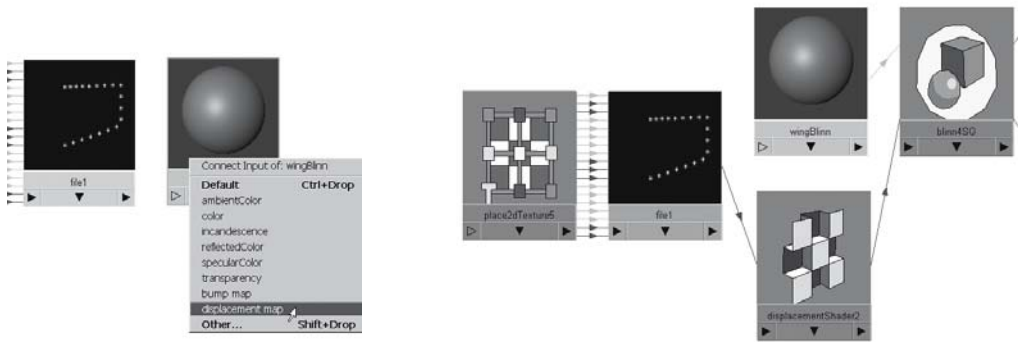


Figure 5.62: Connecting displacement      Figure 5.63: Shader network with displacement

used. Try graphing the input and output of the *wingBlinn* to see the whole shading network (see Figure 5.63).

Before we can render this image, we need to prepare the *wingLeft* surface for the displacement. At render time, Maya will tessellate the surface geometry into many tiny triangulated polygons. Those polygons will then be offset from the surface to a varying degree, based upon the grayscale values of your displacement map. The finer this tessellation, the better your displacement will look.

12. Select the *wingLeft* object and look at it in the **Attribute Editor**. Open the **Displacement Map** section. Increase the **Initial Sample Rate** to 25. This increases the tessellation of the geometry sufficiently to displace smoothly.

13. Now render the frame.

Whoa, this is a bit extreme! (And fairly evil looking, to boot.) What's happened? The displacement is exceeding the limits of our wing's bounding box (see Figure 5.64). This is because the white pixels on the **rivets.tga** are so much brighter than the non-displaced (black) portion of the image. There are a few ways to correct this problem.

One solution is to increase the **Bounding Box Scale** of the wing tessellation, but these huge protruding cylinders are not the look we want for the plane's wing. Another way to fix this would be to decrease the contrast of the file texture being used. This would require you to open the **rivets.tga** file in Photoshop or some other image editor. The third method lets you stay inside Maya to fix the problem.

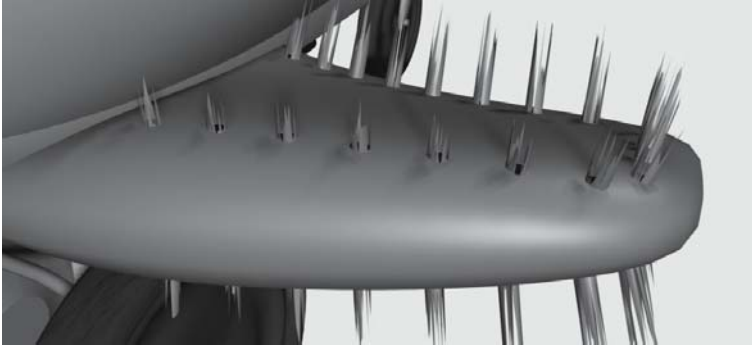


Figure 5.64: Problematic displacement

14. In the **Hypershade**, select the *fileRivets* node. Open the **Attribute Editor** and go to the **Color Balance** section. Here you will adjust the brightness of your image by decreasing the **Alpha Gain** attribute. The lower this number is, the less you'll displace the *wingLeft* object's surface. Set the **Alpha Gain** to a value of 0.1.

15. Render the frame again. This time the displacement is much better looking (see Figure 5.65).

16. Our goal is to create the look of rivets, so this is still a bit large. Drop the **Alpha Gain** attribute of the *fileRivets* node down to 0.02, and render the frame.

17. The rivet displacements are more subtle now (see Figure 5.66 and Color Plate IX). Also, they work during camera close-ups much better than the original bump map technique did. Save your scene.



Figure 5.65: Alpha gain of 0.1 improves the displacement



Figure 5.66: Final displacement

## UV Texture Mapping

Polygonal and subdivision surface models require that you work out the UV mapping before you can display textures on their surfaces. UV mapping is also important in binding your character to a skeleton. There are a few useful tools that you can use during the character skinning process that require good, non-overlapping UV coordinates. So before we can attach the chicken to the skeleton, we'll want to unwrap the model's UVs.

The UV mapping process usually follows these steps. First, select a set of connected faces. Next, project a planar or cylindrical UV mapping set onto the faces. Finally, you'll manually correct distortions and overlapping UVs.

UV mapping can be a bit confusing at first; don't panic! You may find it helpful to take a look at the scene files provided on the CD. In each scene, select the chicken model and then view the

texture layout in the **UV Texture Editor**. Also, check the input nodes in the **Channel Box** to see the construction history of the model. By selecting projection nodes in the **Channel Box** and then pressing **t** on the keyboard, you can view the projection placement.

### Shading Exercise – UV Chicken

1. Set your project to `understanding_maya/Chapter05/shading_uv_chicken`.
2. Open the scene `uvmap_chicken_begin.ma`.
3. Set the **perspective** panel to display **Hardware Texturing**. You'll see the chicken model with a texture map applied to it. This utility map is an 8 x 8 grid pattern used to help you unwrap your UV mapping coordinates. Right now, the pattern appears only on the original cube faces (see Figure 5.67). The faces created by extrusions and splits need to be given UV coordinates in order to properly display the pattern.

Be sure that **Construction History** is turned on. This will help you later on in the tutorial.

4. We'll start with the chicken's neck area, applying a cylindrical UV projection to it.

- a. Select the faces of the neck (see Figure 5.68). Since you may need to re-select this area later, a good habit to get into is the creation of **Quick Select Sets**. Do so by clicking **Create > Sets > Quick Select Set...** Name the set `QS_uv_neck`. Later, you can re-select these faces by clicking **Edit > Quick Select Sets > QS\_uv\_neck**.

- b. In the **Modeling Menu Set**, click **Edit Polygons > Texture > Cylindrical Mapping**.

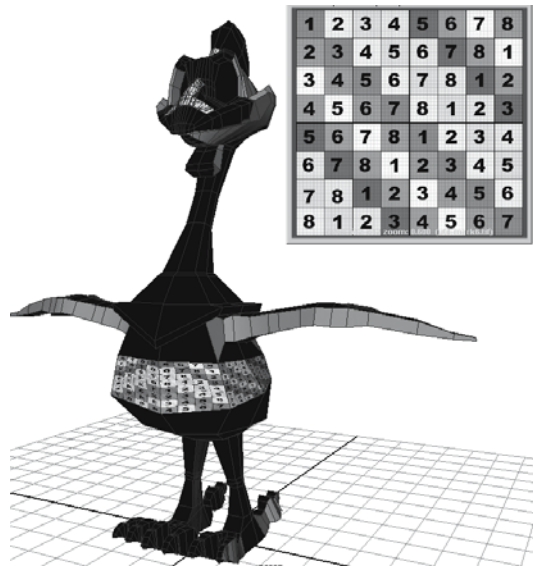


Figure 5.67: Inadequate UVs

## 234 5. Wiring Things Up

---

c. To wrap the projection all the way around the neck, increase the **Projection Horizontal Sweep** value to  $360^\circ$ . This attribute can be found in the **polyCylProj1** node in the **Channel Box**.

This is starting to look better – we can now see the utility marker on the neck. There is, however, a lot of texture stretching and overlapping going on in the upper part of the neck (see Figure 5.69). To fix this, we'll go to the **UV Texture Editor**. (You may want to dock this editor into a split view panel next to your **perspective** panel.)

5. Some of the UV points are outside the 0 to 1 texture range. In other words, they are falling off the edges of the image. There is generally a single UV associated with each vertex of the faces you have mapped. If these UVs are stretched out in a pattern that looks radically different from the polygonal mesh of your model, the texture map won't apply

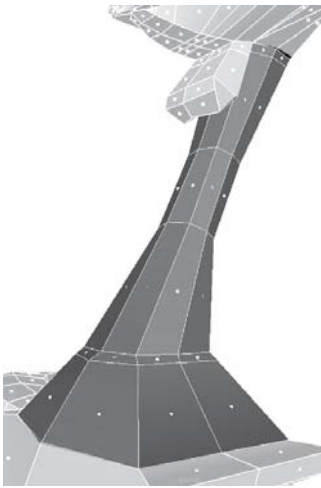


Figure 5.68: Select faces

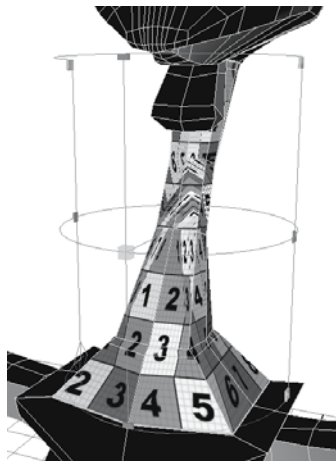


Figure 5.69: UV projection

evenly to your model.

6. We'll be working at the component level, moving UVs and sometimes selecting edges. The editor looks neat right now, but will soon become cluttered with the rest of the model's components. This is because the whole model currently occupies the same 0 to 1 quadrant of UV space. Let's move the neck's UVs to another quadrant temporarily. Using the projection manipulator, move the UVs to the adjoining quadrant below (see Figure 5.70).

7. If the interface is too cluttered you can uncheck the **Display Image** option in the **UV Texture Editor's Image** menu. Also, changing the background color is helpful. This can be set by clicking on the main menu bar **Window > Settings/Preferences > Colors** and altering the **3D View** color.

8. There is a large disconnected area running down the center of the UVs. To seam this back together, use the **Sew** command.

a. **RMB-click** to switch to edge mode. Pick the long, left-leaning diagonal edge. Notice how this automatically highlights its neighboring edge, because these UV edges are actually a single, shared polygonal edge on the model.

b. In the panel menu, click **Polygons > Sew UVs**. This collapses the adjacent UV endpoints into a single UV.

9. Work your way up the model's center, sewing the UVs as you go (see Figure 5.71).

10. Next, pick and move the stretched UVs to the center. Be sure to switch to UV component mode and use the **Move Tool**.

11. The texture mapping on the neck looks much better down the center line. There is a badly stretched area, however. This is caused by the far-off UV. You may not always be able to tell easily where this UV belongs. Which UV is its partner? To determine this, you can select a UV on the model in the **perspective** panel, and then look for which UVs become highlighted in the **UV Texture Editor**.

12. Now that you know where to move it, pick the single far-off UV. Use point snapping to move it to its corresponding UV (see Figure 5.72).

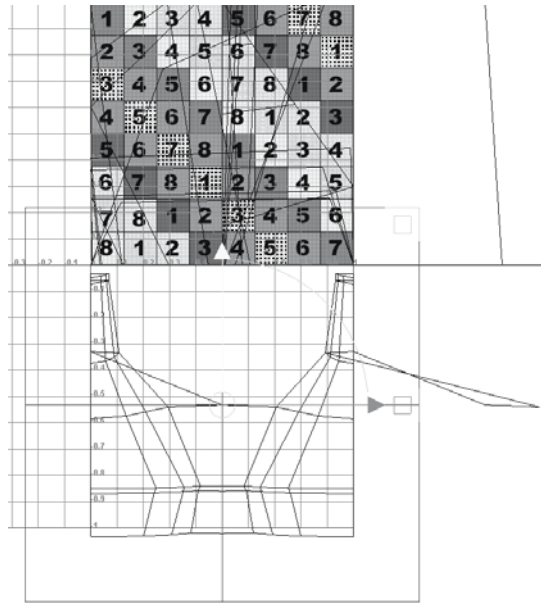


Figure 5.70: Translate the projection

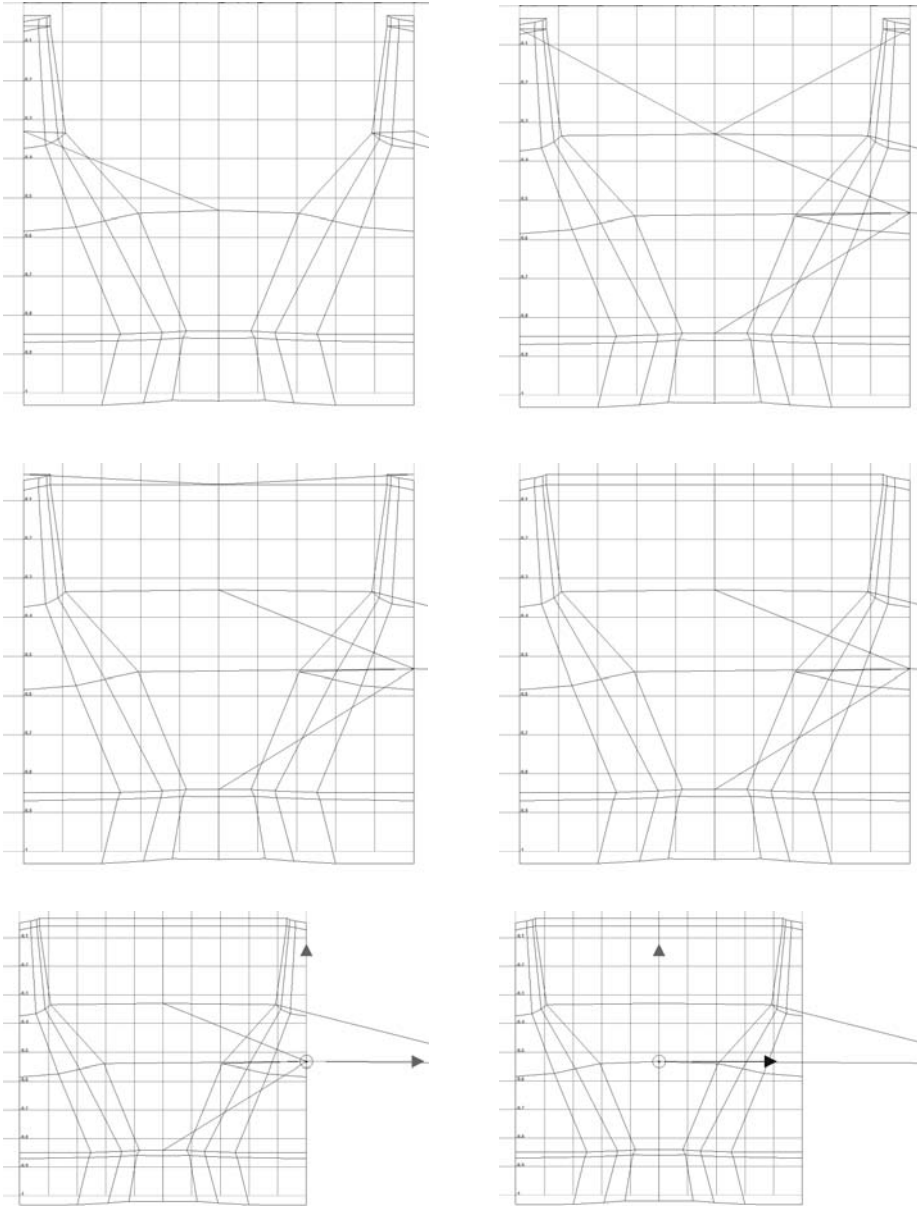


Figure 5.71: Correcting the neck UVs



13. You have two UVs on top of each other which need to be merged into a single UV. Click-drag a selection around the two overlapping UVs. Then click **Polygons > Merge UVs** in the panel menu (see Figure 5.73).

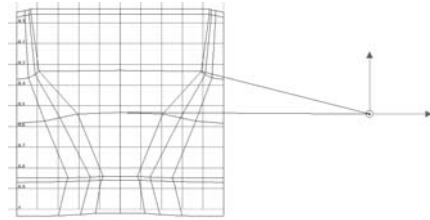


Figure 5.72: Select the stray UV

14. To correct more of the edge warping as in Figure 5.74, drag a selection around all the UVs on one edge and then click **Polygons > Straighten UV Border** in the panel menu.

15. You can continue to lay the UVs into a more even grid using the **Scale** and **Rotate Tools** on select UVs (see Figures 5.75 - 5.77).

16. Notice, too, that the **RMB menu set** allows you to perform the most common commands on the UVs in the UV editor.

In this next section we'll look at the body. The overhanging neck collar presents some potential problems. How can you project UVs onto this area without streaking and overlapping the texture?

One way to go about this is to project UVs onto an easier-to-map proxy model and then transfer the mapping to the original model. The proxy model must be exactly the same as the original model in its number of vertices and overall topology, but those vertices don't have to be in the same place. This means you can:

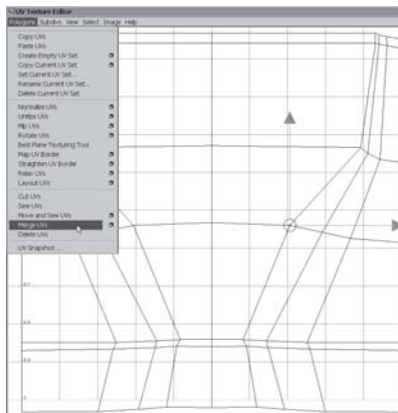


Figure 5.73: Merge UVs

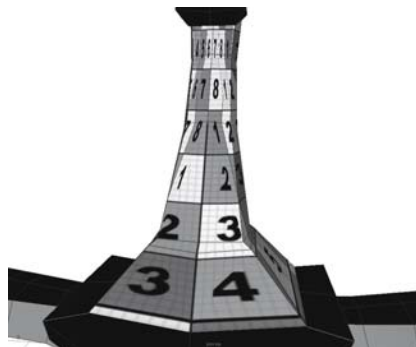


Figure 5.74: Edge warping

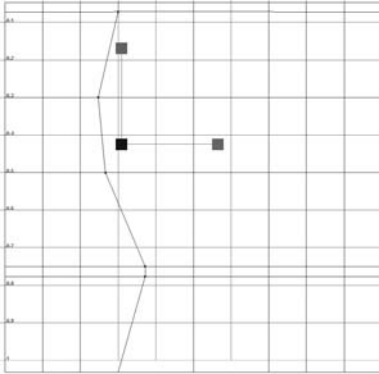


Figure 5.75: Warped UVs

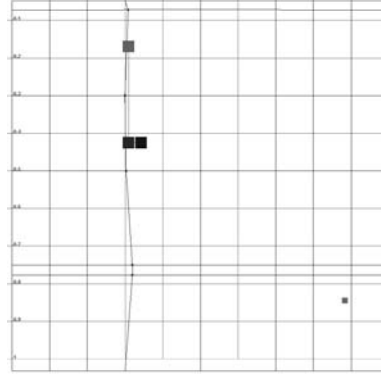


Figure 5.76: Straightened UVs

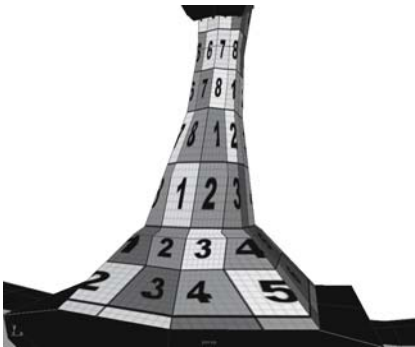


Figure 5.77: Unwarped UVs

- a. Duplicate the original model;
  - b. “Melt” this proxy model into an easy-to-map shape;
  - c. Project UVs onto the proxy model; and
  - d. Transfer the proxy’s UVs back onto the original model.
17. Select the *polyChickenSkin* model. Duplicate with transform (**Shift-d**) and move the new model a few units away. Name the duplicate *uv\_proxy\_chicken*.
18. With the proxy model selected in object mode, click **Polygons > Average Vertices**

**options** in the main menu. Using the defaults, click the **Apply** button three times. The poor proxy bird is looking pretty sickly! The positions of the vertices have been averaged, which sort of “melts” the model. The benefit is that the overlapping details have all but disappeared (see Figure 5.78).

19. Select the proxy model’s body faces. Create a **Quick Select set** named *QS\_uv\_body*.

20. Map a cylindrical projection to the body faces by clicking **Edit Polygons > Texture > Cylindrical Mapping**. In the **Channel Box**, set the **Projection Horizontal Sweep** value to 360° (see Figure 5.79).

21. Move this new projection to the lower left quadrant of the **UV Texture Editor**.

22. To fix the back, pick the four vertical edges on the proxy chicken’s back shown in Figure 5.80.

23. In the **UV Texture Editor**, click **Polygons > Cut UVs**. This essentially unmerges the two UVs shared at each selected vertex position. Using the move tool, move the UVs out on the X-axis (see Figures 5.81 and 5.82). (These selections can be tricky, so use the undo command if you have to move a UV to the opposite side.)

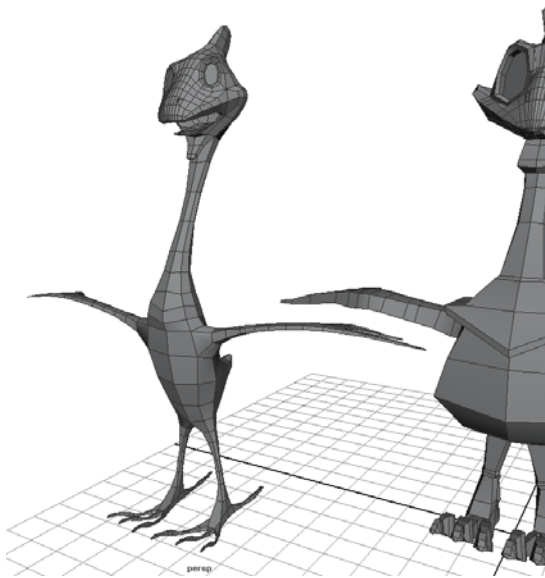


Figure 5.78: Proxy chicken

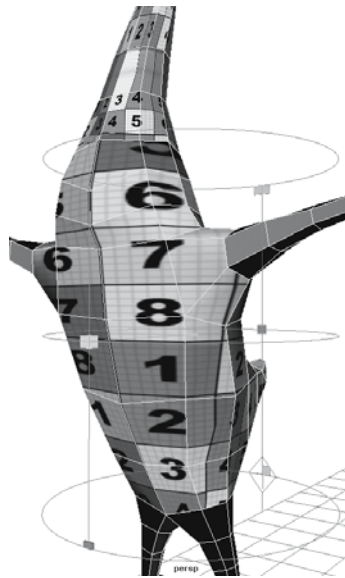


Figure 5.79: Body projection

## 240 5. Wiring Things Up

---

24. Now it should be clear where the two stray UVs should go. Go ahead and move them, using point snap. Merge the overlapping UV pairs.

Notice in Figure 5.83 how the arm and shoulder faces are badly stretched. This is caused by overlapping UV coordinates, as seen in Figure 5.84.

25. To fix this stretching, we'll average the UV positions. This will "melt" the UVs, much as we did with the vertices to create the proxy model. In the **UV Texture Editor**, pick all the UVs in the body projection. Click **Polygons > Relax UVs options**. Choose the default options and then click the **Apply** button three times. This averages the inner UV positions while pinning the border UVs in place. (Pinning prevents the seams from opening up.)

The arm sockets are much improved. You can further refine them by moving the individual UVs (see Figures 5.85 and 5.86). It's important to avoid overlapping the UVs or stretching the faces too much.

26. You can see a similar overlap problem at the top of the back. You should correct this by

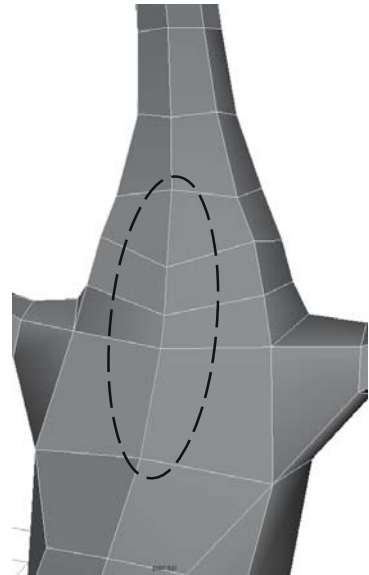


Figure 5.80: Selected edges

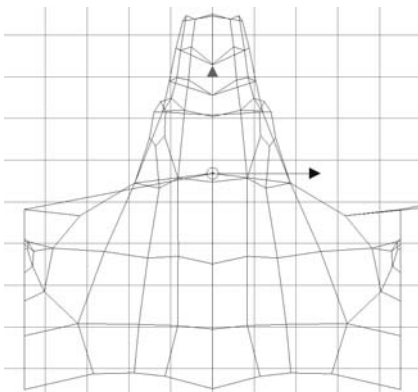


Figure 5.81: Merged UVs to be cut

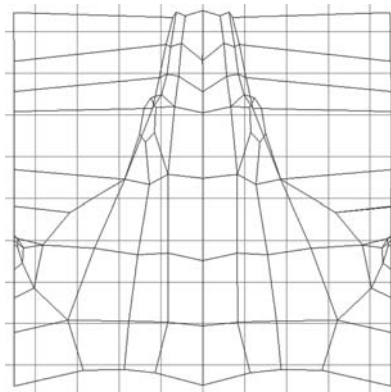


Figure 5.82: Cut and flattened UVs

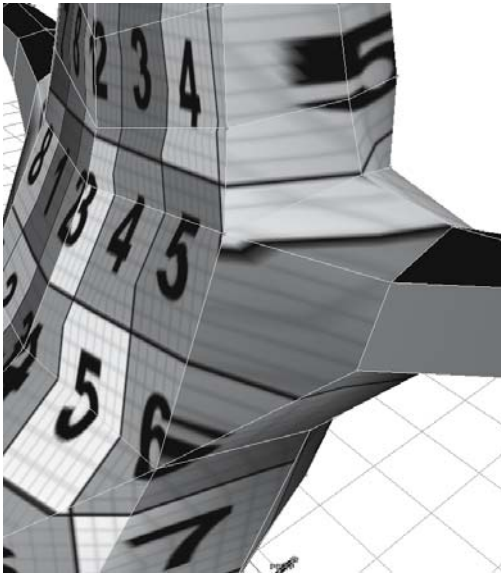


Figure 5.83: Smeared texture

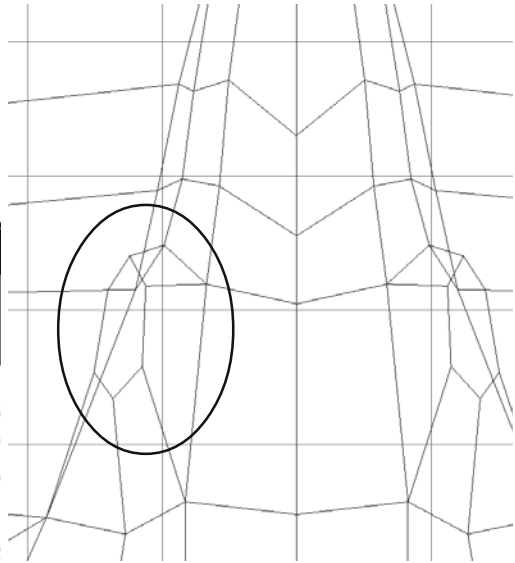


Figure 5.84: Overlapping UVs

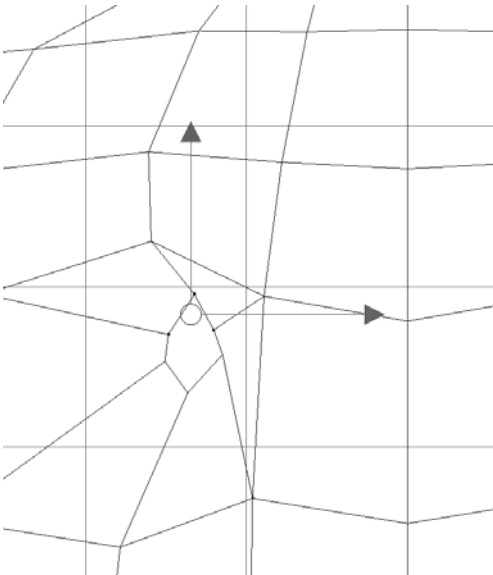


Figure 5.85: Relaxed and tweaked UVs

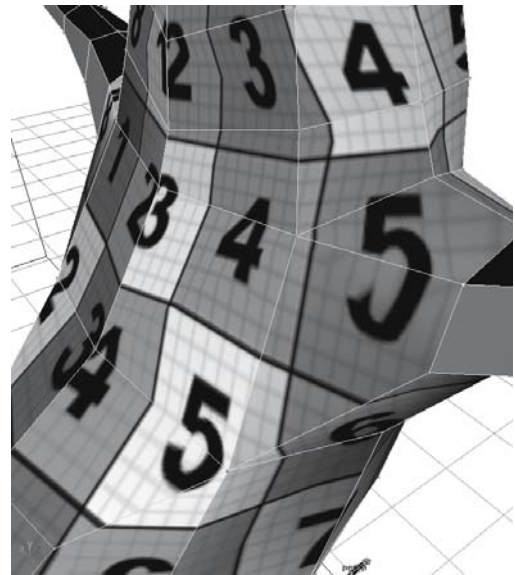


Figure 5.86: Corrected texture

## 242 5. Wiring Things Up

---

manually repositioning the offending UVs. The tail has some overlaps that can be fixed manually, too.

Looking at the proxy chicken, the utility texture is now displayed on the entire body and neck without any overlapping. While it isn't perfectly uniform in terms of density, these UVs are definitely good enough to allow 3D texture painting and skin weighting later on.

Now, let's transfer these weights from the proxy model back onto the original model.

27. In object mode, select the *uv\_proxy\_chicken* first, and then **shift-select** the *polyChickenSkin* model second. On the main menu, click **Polygons > Transfer options**. Reset the settings and then press the **Transfer** button.

The UVs have now been transferred, and your *polyChickenSkin* model displays the utility texture correctly. Notice in particular how the overhanging faces at the base of the neck are not streaked – this would have been a very tricky area to map without the proxy transfer (see Figure 5.87).

Maya's **Construction History** has established a link between the UV mappings of the two chicken models. As you continue to refine the *uv\_proxy\_chicken* you'll be able to evaluate the results immediately on the *polyChickenSkin* model.

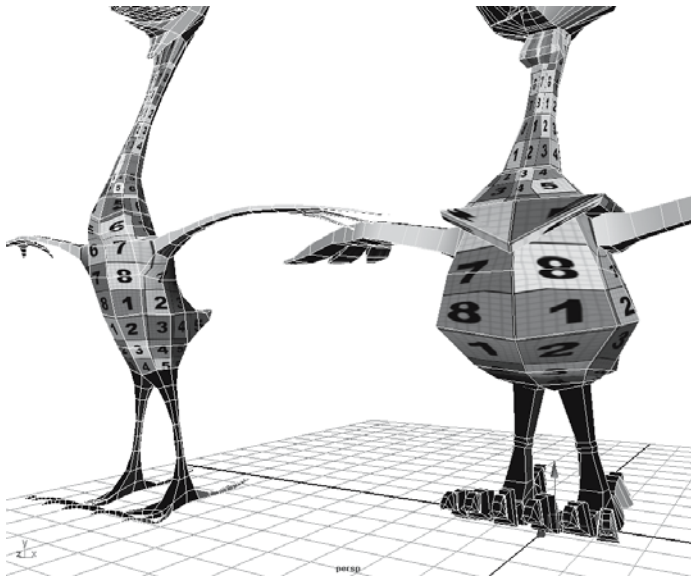


Figure 5.87: UVs transferred from proxy to original model

Next, we'll align and connect the neck UVs with the body UVs. While this step isn't strictly necessary, it can make it easier to visualize the mapping during the 3D paint process.

28. Re-select the *uv\_proxy\_chicken* in object mode. Then, in the **UV Texture Editor**:

- a. RMB-click and choose **Edge** mode.
- b. Select the bottom row of edges of the neck UV set.
- c. In the panel menu, click **Polygons > Move & Sew options**. Reset the settings and then click the **Move & Sew** button.

This has moved one set to align with the other, and merged the overlapping UV points with each other. You now have a single UV shell for the neck and body faces, as in Figure 5.88.

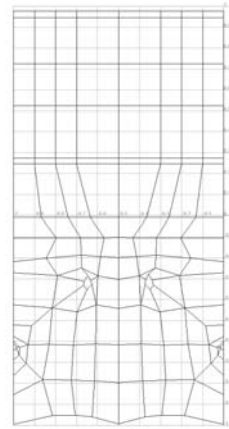


Figure 5.88: Sewn UVs

The next section we'll map is the left leg. The process is the same as the one we've used on the neck and the body.

- Select the faces of the left leg.
- Create a **Quick Select Set** called *QS\_uv\_lf\_leg*.
- Apply a cylindrical mapping projection to the selection.
- Unfold the UVs in the **UV Texture Editor**.

You'll notice that there are too many overlaps to unfold by hand. We'll let Maya neaten things up a bit before we proceed. Maya can automatically lay your UVs out so that they don't overlap (this is the basis for the **Automatic Mapping** projection type). Unfortunately, this tends to explode your original UV shell into a large number of separate ones, so we'll need to sew them back together.

29. Click in the main menu on **Edit > Quick Select Sets > QS\_uv\_lf\_leg** to select all of the leg faces (see Figure 5.89).

30. In the **UV Texture Editor**, click **Polygons > Layout UVs options**. Reset the settings and then uncheck **Rotate For Best Fit**. (This will preserve the up-and-down orientation of the UV shells.) Click the **Layout UVs** button.

The pieces of the leg are now scattered around the 0 to 1 UV space. It's time to gather them up.

31. In edge selection mode, **LMB**-click on the edge of a loose UV shell. Notice the “partner” edge, which also becomes highlighted. Use the **Move and Sew UVs** command to re-unite the two shells. You will do this for each piece until you have a single shell for the entire leg. You will need to manually move UVs in some cases – this can be a time-consuming puzzle to solve, but the results are well worth it.

32. Sometimes, you’ll want to relocate the seam on your UV shell. For example, if the seam is at the front of the leg, you may want to place it on the inside. Here, it will draw less attention to itself. To do so,

a. Select an entire vertical row of edges where you want the seam to be. You can make this selection in the **perspective** panel if you like.

b. In the **UV Texture Editor**, click **Polygons > Cut UVs**. This separates the shared UV points at each vertex.

c. You now have two UV shells. Select the outside edge row of one shell. You’ll see the other outside row become selected, since they represent shared vertices on the polygon mesh.

d. Click **Polygons > Move and Sew UVs**. One shell has merged with the other to create a single shell with your new seam location.

33. You may want to do some manual repositioning of UVs to match the layout of the polygonal mesh. In other words, the more a UV face looks like the 3D polygonal face, the better (see Figure 5.90). This will help unify the density and eliminate stretching.

The mapping is starting to look good on the original model (see Figure 5.91). The next section we’ll map is the head. Here is an overview of the process.

- Select the head faces.
- Create a **Quick Select Set** called *QS\_uv\_head*.

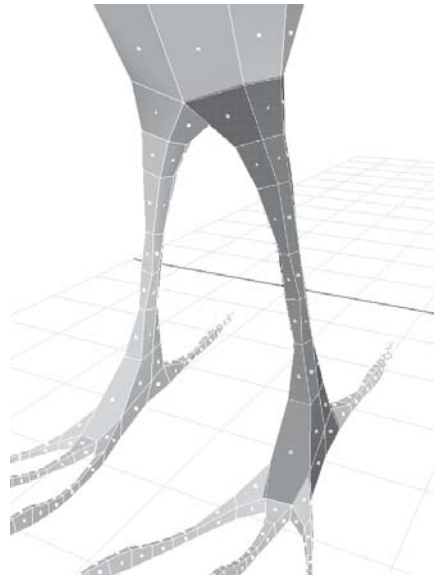


Figure 5.89: Leg face selection



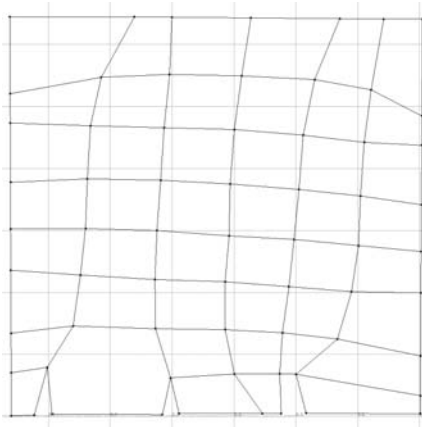


Figure 5.90: Leg UV mapping

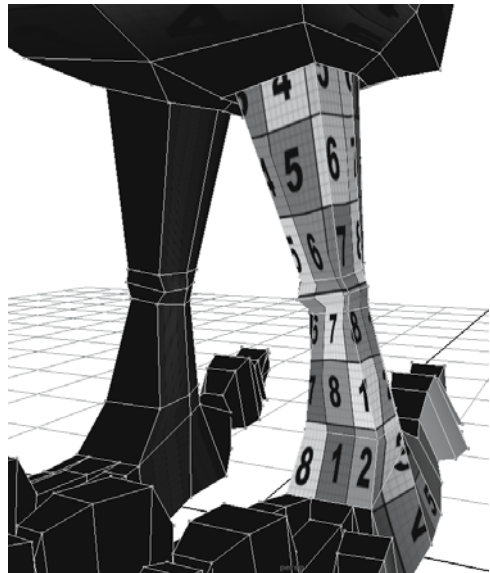


Figure 5.91: UVs transferred to original model

- Apply a **Cylindrical Mapping** projection.
- In the **UV Texture Editor**, move this projection to its own quadrant of UV space.
- Using the same tools as on previous parts, unfold the UVs.

One way to deal with the overlapping UVs in the head is to select the *QS\_uv\_head* set and then use **Layout UVs (Polygons > Layout UVs options** in the **UV Texture Editor**). Reset to the default options, and then switch **Separate** mode from **All Intersecting** to **Folds**. The **Folds** option tells Maya to fix overlapping faces with surface normals that point at opposite directions: for example, the undercutting section of the eye socket or the inner walls of the nostrils. This technique will give you a good starting point for further tweaking.

The illustrations accompanying this tutorial should serve as a guide for mapping the rest of the chicken (see Figures 5.92 - 5.105). You'll use these same basic techniques on each section until the entire mesh displays the utility texture properly.

You'll sometimes notice that a mapping projection fits over the entire 0 to 1 UV space, giving much more texel density to the faces than is really needed. This mismatch in texture scale will look strange later, when you start painting the model.

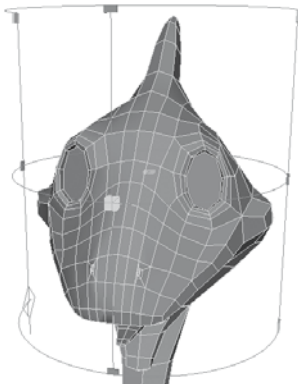


Figure 5.92: 180° Head projection

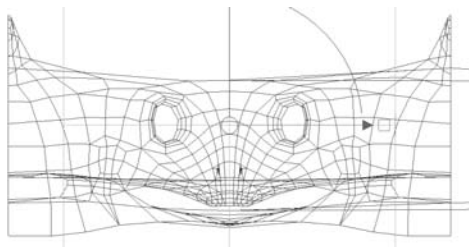


Figure 5.93: Unwrapped head projection

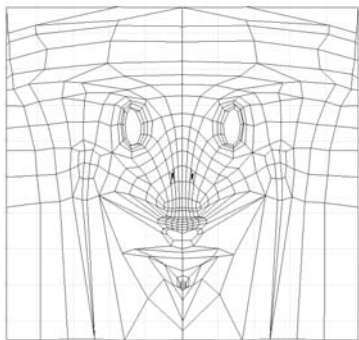


Figure 5.94: Corrected head UVs



Figure 5.95: Head texture

If you map the waddle faces, for example, they will initially cover the same area as the entire body does (see Figures 5.96 and 5.97). In order to fix this, you can scale the projection down to cover a smaller area of UV space (see Figures 5.98 and 5.99).

34. Now that the entire model has good UV coordinates, it's time to place them all into the 0-to-1 texture space (see Figures 5.106 - 5.108 and Color Plate VI). Packing the textures like this makes it possible to paint on the model with a single texture map. When you're finished, delete history on your *polyChickenSkin* model and save the scene.

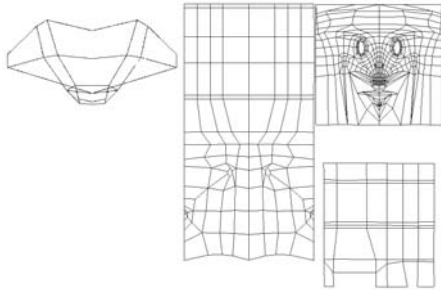


Figure 5.96: Overly large mapping area

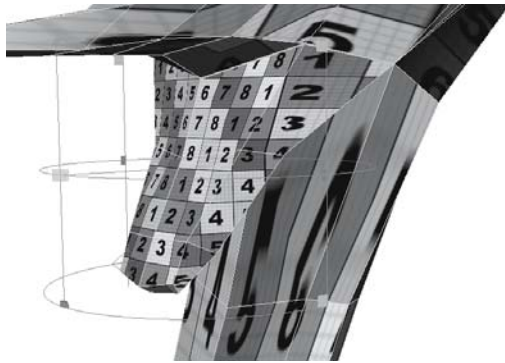


Figure 5.97: Extreme texel density

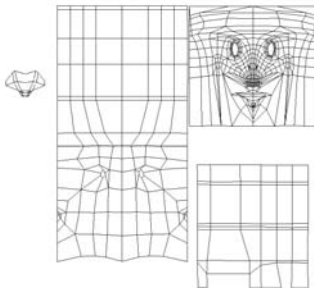


Figure 5.98: Corrected mapping area

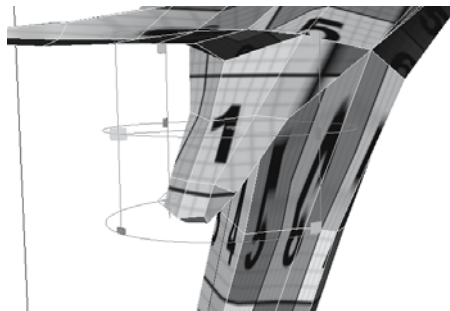


Figure 5.99: Unified texel density

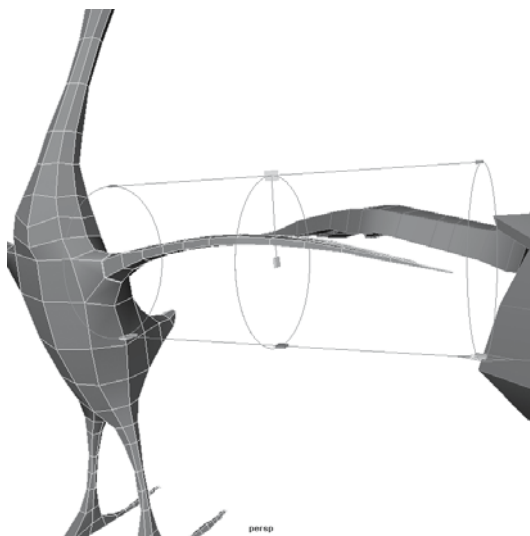


Figure 5.100: Cylindrical wing projection

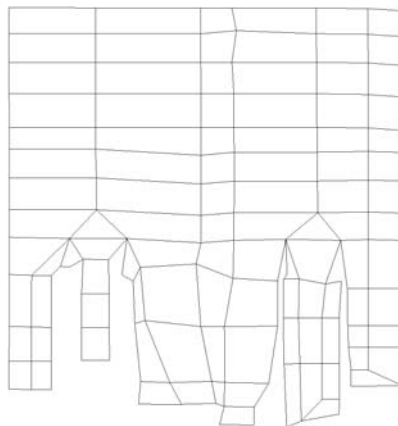


Figure 5.101: Straightened wing UVs

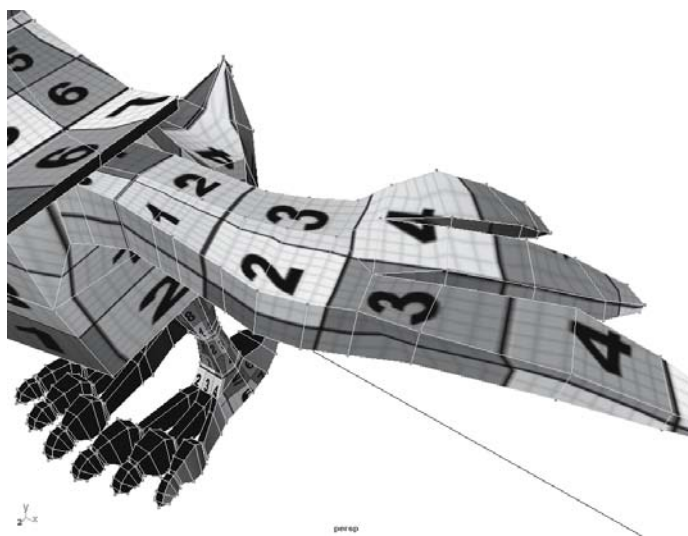


Figure 5.102: Wing mapping transferred from proxy model

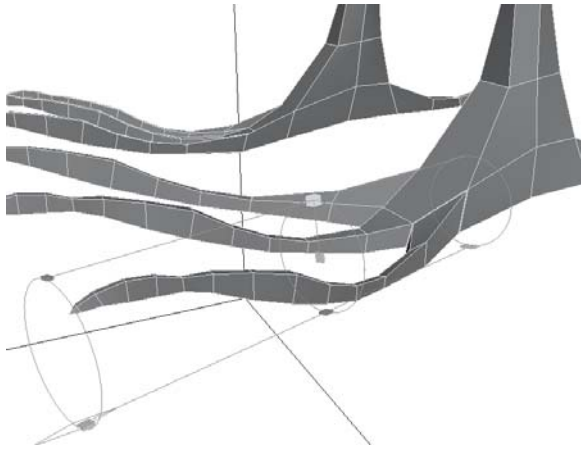


Figure 5.103: Cylindrical toe projection

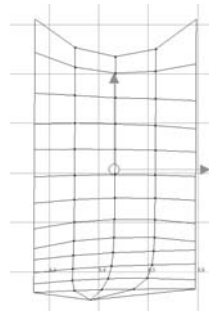


Figure 5.104: Straightened toe UVs

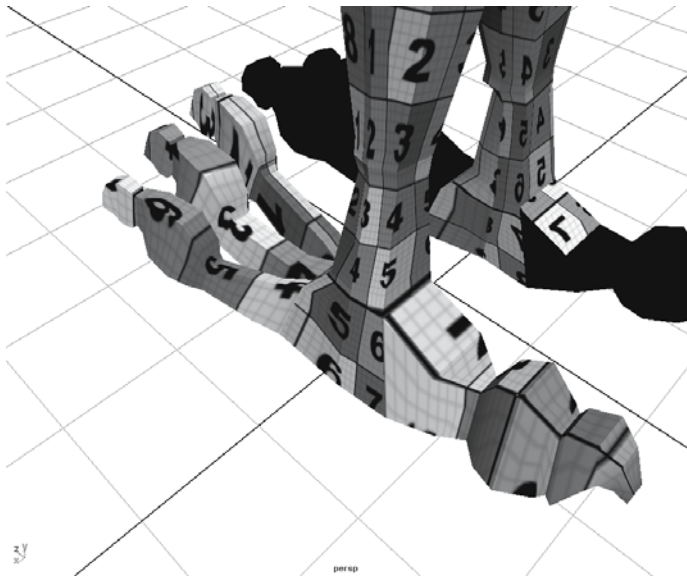


Figure 5.105: Toe mapping transferred from proxy model

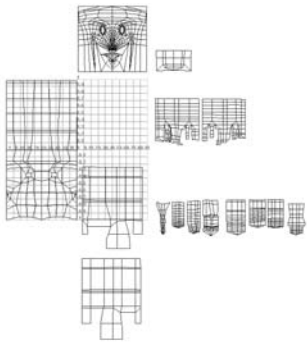


Figure 5.106: Messy UV layout

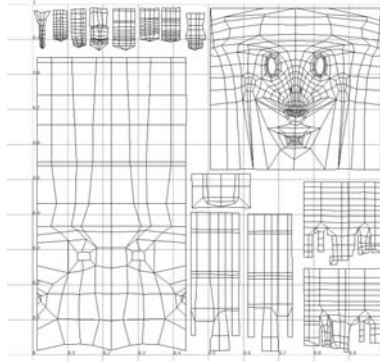


Figure 5.107: UVs in 0-to-1 space

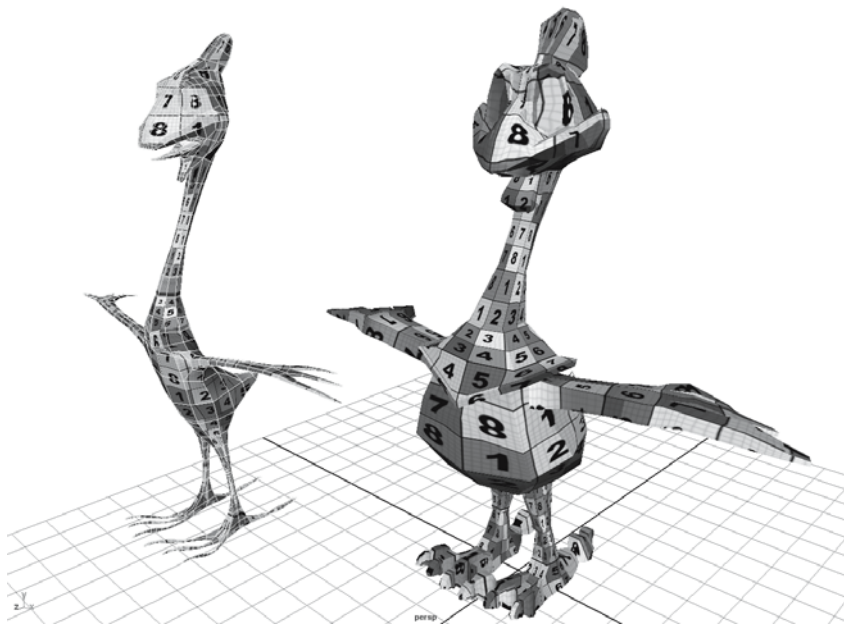


Figure 5.108: Final UV mapping

## 5.5 Lighting – Three-point Scheme



Maya allows you to create and place as many light sources in your scene as you like. This can be a bit overwhelming, so it's often a good idea to use traditional, real-world lighting as a starting point. To illuminate your subject, try a three-point lighting scheme: key light, rim light, and fill light.

Each of these light positions is used for a particular purpose. The **key** light is the main spotlight that shines over the shoulder of the camera from a 3/4 angle above the subject. The **rim** light produces a slight edge or rim of light around the edge of the subject, popping it out against the background. These lights are usually positioned behind the subject pointing nearly directly at the camera. The **fill** light is usually a dim light positioned lower than the subject in order to fill in some of the dark shadows and add bounced color from below, as in Figure 5.109.

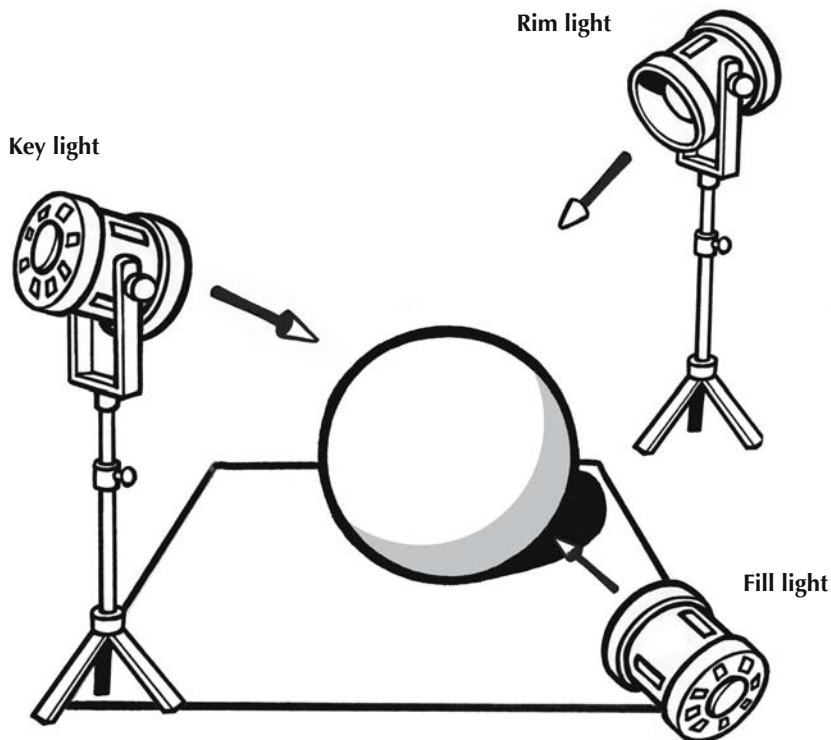


Figure 5.109: Three-point lighting

### Lighting Exercise – Three-point Lighting Plane

1. Set your project to `understanding_maya\Chapter05\lighting_three_point`.
2. Open the scene `lighting_three_point_plane_begin.ma`.

Take a look at the scene. You'll see the airplane parked inside the barn. Also, you may notice that the **perspective** view is locked off, meaning you can't change the view. This has been done by locking the transform channels in the **Channel Box**. Often when you are lighting a shot, it's with a particular camera view in mind. It's not generally possible to light a scene to look good from all angles, so focus on perfecting the one camera view that the audience will see.

3. To work most effectively when lighting your scene, you should set up a custom workspace. A four-pane setup with the following panels will do the trick:

- **Perspective** view to evaluate real-time lighting,
- **Render** view for rendering IPR images,
- **Key light** view (initially this will be the default **top** view),
- **Rim/Fill light view** (initially this will be the default **front** view), and
- **Attribute Editor** docked on the right.

4. To begin, create the key light by clicking **Create > Lights > Spotlight**. Name this light `key_light`.

5. While the `key_light` is still selected, click in the **top** view panel's menu on **Panels > Look Through Selected**. This will change the view to look through the `key_light`. Try using your normal camera manipulation controls (**alt-LMB**, **alt-MMB**, etc.) to point the light at the airplane (see Figure 5.110).

6. You can set the **perspective** view to give you real-time lighting feedback by pressing **7** on the keyboard. Evaluate the lighting on the plane as you adjust the `key_light`'s position.

7. In the **Attribute Editor**, turn on **Depth Map Shadows** for the `key_light`. To soften the shadows, set the **Dmap Resolution** to **128**, and the **Dmap Filtering** to **3**.

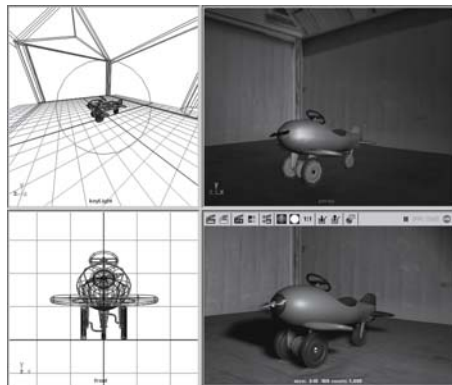


Figure 5.110: Setting the key light



8. Next, open the **Render Globals** window by clicking the icon at the top of your **Render View**. In the **Common** tab, set the **Resolution** to **640x480**. In the **Maya Software** tab, set the **Anti-aliasing Quality** to **Production Quality**.

As you add lights and adjust them, you can use the **IPR** renderer for near-real-time feedback.

9. Click the **Render IPR Frame** button in the **Render View**. When it finishes, **LMB**-drag a selection around the image.

10. Now, increase the *key\_light*'s **Intensity** to 1.15 in the **Attribute Editor**, while watching the **IPR** render. You can also add a little warmth to the **Color** attribute. (Try a very slight reddish hue, since there is a lot of red paint in the barn.)

11. Create another spotlight, this time naming it *fill\_light*. Set the bottom left panel to **Look Through Selected** so that you can aim the *fill\_light*.

12. Aim the *fill\_light* at the underside of the *pursuit\_plane* (see Figure 5.111). You may want to increase the **Cone Angle** to around 80°. This helps fill in some of the dark areas. In real life, light that bounces off the ground and walls would keep the plane's underside from looking so dark.

13. Lower the *fill\_light*'s **Intensity** to a value of 0.4 and warm its **Color** up to a brownish tint.

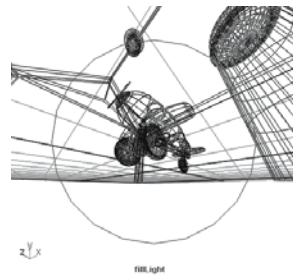


Figure 5.111: Fill light aim

One problem the *fill\_light* has introduced is a small, secondary highlight on the plane's nose, as seen in Figure 5.112. A very diffuse light source such as a fill light should not produce a highlight. We'll fix the problem by setting the *fill\_light* to emit only diffuse light.

14. In the **Attribute Editor**, with the *fill\_light* selected, uncheck the box in front of **Emit Specular**. Notice how the second highlight has disappeared from the **IPR** view (see Figure 5.113).

15. Next, add a spotlight to the scene, and name it *rim01\_light*. In one of your camera view panels, choose **Panels > Look Through Selected**. Aim the *rim01\_light* at the back side of the plane, using a high enough angle to throw a bright rim of light across the top of the plane's nose (see Figures 5.114 and 5.115).

16. Increase the *rim01\_light*'s **Intensity** to 2.5.

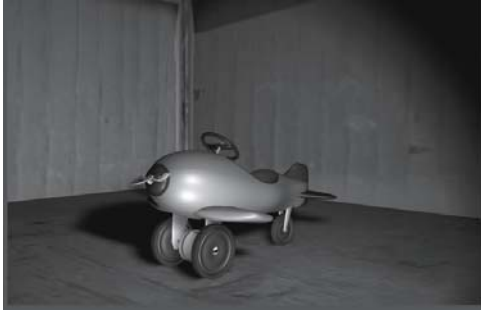


Figure 5.112: Extra fill light specular

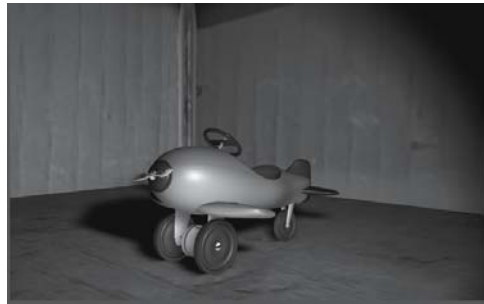


Figure 5.113: Specular removed

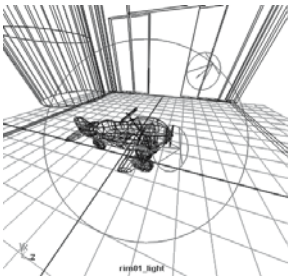


Figure 5.114: Rim01 position



Figure 5.115: Rim01 contribution

17. Repeat these steps to create *rim02\_light*, which adds a defining rim to the plane's tail (see Figures 5.116 and 5.117).

You've probably noticed that the floor of the barn has become overly bright, even though the plane looks just right. It's often difficult to balance the subject and environment lighting. Fortunately, in Maya it's easy to exclude surfaces from the illumination of specific lights.

18. Click **Window > Relationship Editors > Light Linking > Object Centric**. This window allows you to disconnect surfaces from the effects of your scene lights. On the left side of the window, pick the *barnGroup*. Now, click on the *fill\_light*, *rim01\_light*, and *rim02\_light*. This leaves only the *key\_light* to affect the barn.

19. Stop your **IPR** render by clicking the small stop sign icon in the **Render View**. Click on the normal **Render Current Frame** icon to see the final, anti-aliased render, as in Figure 5.118 and Color Plate XI.

20. Save your scene.

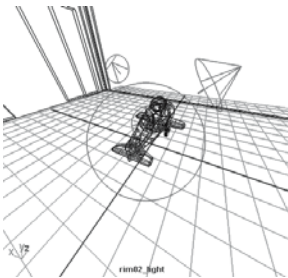


Figure 5.116: Rim02 position



Figure 5.117: Rim02 contribution



Figure 5.118: Finished render

### 5.6 Rendering – Camera Cuts



When telling a visual story, you may choose to cut between different cameras for the shots in a sequence. In traditional cinematography you would run separate cameras and then edit the footage together later. This is similar to the way you do it with virtual cameras in Maya.

When you place multiple cameras into the scene you must carefully plan your edits in advance. You'll render a partial frame range through each camera. You'd waste a lot of time rendering the full frame range through each camera and then cutting those frames in editing.

In this scene there are three cameras that will be used for three different shots:

- Establishing shot through Camera A, frame 1-72,
- Medium shot through Camera B, frame 73-199, and
- Point-of-view shot through Camera C, frame 200-439.

In the Render Globals window, you'll select Camera A for frames 1-72, then batch render. Next, switch the Render Globals to Camera B for frames 73-199 and batch render. Finally, you'll render Camera C from frame 200-439. You can use a non-linear video editing program such as Adobe Premiere or Apple Final Cut to cut together your shots. In such cases, it's often a good idea to render extra frames at the head and tail of each shot. This gives you the flexibility to use cross-fades and other transition effects that require overlapping footage.

### Rendering Exercise – Camera Cuts

1. Set your project to `understanding_maya\Chapter05\rendering_camera_cuts`.
2. Open the scene `camera_cuts_barn_begin.ma`.
3. This scene contains the plane's flight from the barn, using a motion path for animation. You'll add three cameras to the scene in order to capture the action (see Figure 5.119).
4. Create a two-node camera by clicking **Create > Cameras > Camera and Aim**. This camera will be used for the establishing shot and final shot. In the **Outliner**, rename `camera1_group` as `cam_aGroup`, `camera1` as `cam_a` and `camera1_aim` as `cam_a_aim`.

5. Set one of your upper-left view panels to look through *cam\_a*. Use the camera control keys to aim the camera at a 3/4 angle to the barn's front right side. (This could represent a camera on a crane in real life, so get it up off the ground a little bit.)
6. Next, create a second two-node camera called *cam\_b*. (Rename the related group and aim nodes as well.) View through this camera in your lower-left view panel.
7. Aim *cam\_b* to look at the front door of the barn. This should also be from a 3/4 angle, but much closer; as if mounted on a tripod.
8. The third camera you'll create is a single-node camera called *cam\_c*. View through *cam\_c* in your bottom-right view panel.
9. Working in the **perspective** view, move and rotate *cam\_c* to a position inside the cockpit of the *pursuitPlane*, looking just over the steering wheel. This point-of-view camera will give the sense of sitting inside the plane.
10. To keep *cam\_c* inside the plane, you'll want to parent it to the highest node in the plane's hierarchy. In the **Outliner**, select *cam\_c* first, and then **ctrl**-select the *pursuitPlaneGroup* node. Click **Edit > Parent**.

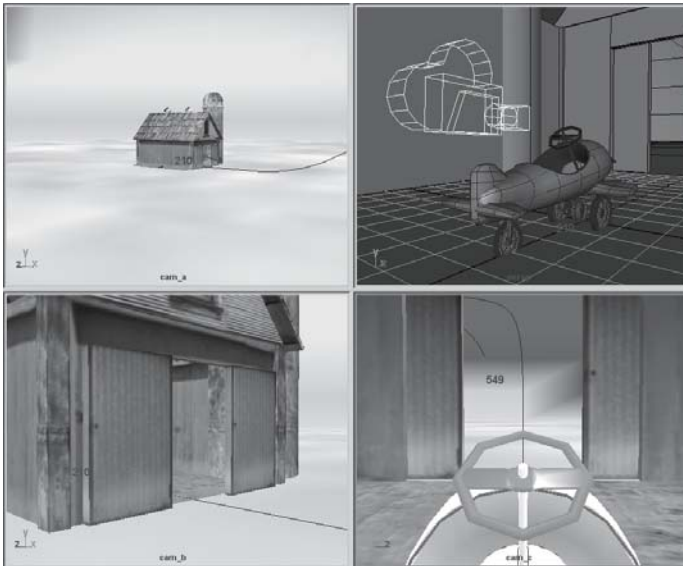


Figure 5.119: Cameras placed

## 258 5. Wiring Things Up

---

11. Play the animation and you'll see the scene from the plane's point-of-view.

You should assess the camera cuts before you begin a lengthy render. One simple way to do this is to set Maya's preferences to allow animation in multiple views simultaneously.

12. Click **Window > Settings/Preferences > Preferences**. Then choose the **Timeline** category. In the **Playback** area, set the **Update View** attribute to **All** (see Figure 5.120). Close the preferences window by clicking the **Save** button.

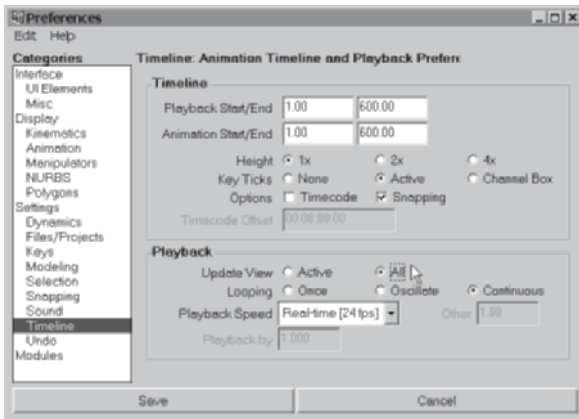


Figure 5.120: Playback preferences

13. Play your animation – you can watch the three cameras in real-time in order to make decisions about when to cut between cameras. For example:

- Long shot                      cam\_a                      frame 0001 – 0072,
- Medium shot                  cam\_b                      frame 0073 – 0199,
- Point of View                cam\_c                      frame 0200 – 0439.

Maya's **Playblast** feature is a good way to test your edit decisions before committing to a lengthy batch render. It is easy to set up the **Playblast** settings to save your frames to disk. You can **Playblast** each cut through the appropriate camera view and create a single sequence that will play your edits as a single movie.

14. To **Playblast** the sequence, first open the **Render Globals** window. Set the **Resolution** to the **320x240** preset. We'll use this as the **Playblast** resolution, so that all camera cuts render at the same size. Close the **Render Globals**.

15. Next, **RMB**-click in the **cam\_a** view panel to make it active. **Playblasts** are rendered through the active camera.

16. Set the **Range Slider** from 1 – 72, then **RMB**-click on the **Time Slider**, and choose **Playblast... options** (see Figure 5.121).

17. Reset the settings; then set the options to:

- Viewer = **fcheck**,
- Display Size = From Render Globals,
- Scale = 1,
- Save to File = on (checked), and
- Image file prefix = **plane**.

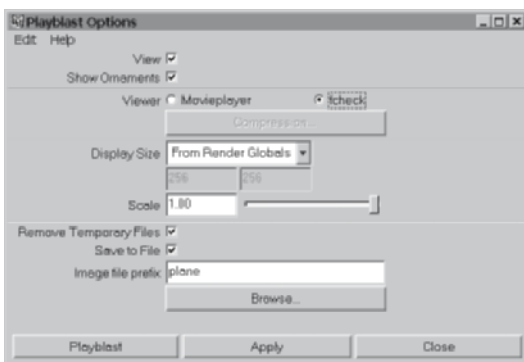


Figure 5.121: Playblast options

18. Click the **Playblast** button. Maya will capture the first three seconds through **cam\_a** and save the frames to disk. **Fcheck** will launch when the sequence is complete and play the frames. Close **fcheck** when you're done.

19. Focus the view by **RMB**-clicking on **cam\_b**.

20. Set the **Range Slider** to 73 – 199.

## 260 5. Wiring Things Up

---

21. **RMB**-click on the **Time Slider** and choose **Playblast...**

22. Maya will warn you that frames may be overridden. This is due to that fact that we've saved the previous set of frames. Since we've changed the **Range Slider**, nothing will be overridden. Click **Ok** to continue.

23. For the final cut, focus the view on **cam\_c**.

24. Set the **Range Slider** to 200 – 439.

25. **Playblast** the sequence.

The sequences you have playblasted were named identically, and have an uninterrupted numerical order. If you look in the **images** directory of your project, you'll see that this is, in essence, a single sequence; plane.0001-0439.iff. By viewing this sequence with **fcheck**, you'll see your finished movie, with the camera cuts built in.

26. Save your scene.

### 5.7 Project 5 – Preparing for Animation

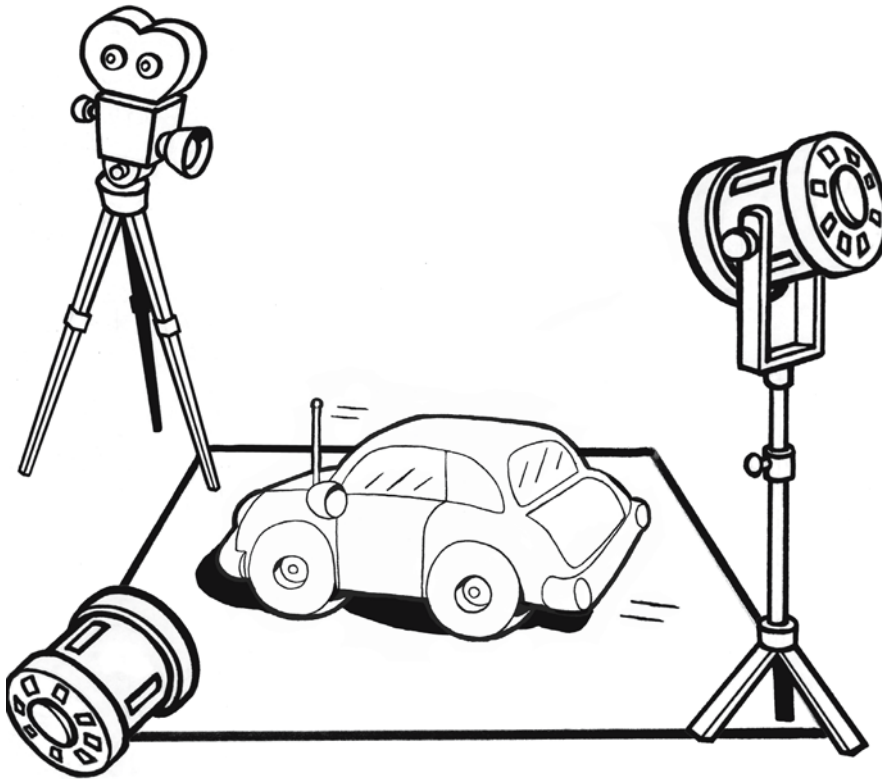
In this project, you should get your scene ready for animation. This includes:

- Character setup and rigging,
- Camera cuts,
- MEL shelf buttons for animation, and
- Three-point lighting.



# Chapter 6

## Bringing It All Together



### 6.1 Review – Editing Your Camera Cuts

You should carefully storyboard your project, and then animate multiple cameras to tell the story with wide shots, medium shots, close-ups, pans, tilts, dollies, zooms, etc. Once you are satisfied with the quality of your shots (tested by rendering individual frames at the beginning, middle, and end of a camera shot) you can set the Render Globals to render each sequence of frames.

These sequences can then be edited together in Premiere, Final Cut, or any other video editing software. If you want to cross-fade between shots, be sure to render some extra head and tail frames on each sequence.

### 6.2 Workspace – Layers, Show by Type, Hiding and Showing



When the complexity of your scene grows beyond just a few simple objects, it can be difficult to select and manipulate individual objects. Here are a number of ways to reduce the clutter and make it easier to manage your scenes.

#### Layers

Layers can be used to organize a number of objects into discrete sets, which can then be shown, hidden, or templated (frozen) depending on what you are working. For example, a **SetLayer**, **PropsLayer**, and **CharacterLayer** could help you to organize your scene. By hiding the Set and Props Layers, you can focus on the character.

#### Show by Type

You can turn off the display of entire object categories. The **Show** menu in each panel (for example, top, side, and perspective) allows you to specify what is displayed. By unchecking

**NURBS Surfaces, Polygons, and Subdiv Surfaces** you could focus on the Joints, IK Handles, Lights, and Cameras in your scene.

### Hiding and Showing

The third way to alter your view of the objects in your scene is Hiding and Showing. The **Display > Hide** and **Display > Show** menus allow you to toggle the visibility of things in the scene. Keep in mind that hidden lights will not illuminate any scene objects.

### Snapping

The snapping tools are used to constrain the translation and rotation of objects to either a predefined grid or to specific points, curves, or angles. While movement snapping is toggled on, the center of the move tool will turn from a yellow square to a yellow circle. Here are the four major types of snapping and their associated hotkeys.

#### **Grid Snapping: x**

When used in conjunction with the move tool, grid snapping allows you to move an object in precise, one-grid increments. With grid snapping on, the move tool snaps the object or component to any grid intersection in the workspace.

#### **Curve Snapping: c**

Curve snapping allows you to translate an object to any point along a curve. Use the **MMB** to click on the curve to which you want to snap.

#### **Point Snapping: v**

Point snapping allows you to move an object to an individual point, such as a vertex or CV.

#### **Angle Snapping: j**

When used in conjunction with the rotate tool, this allows you to rotate an object in predefined angle increments. This angle can be changed in the Rotate Tool options box.

### 6.3 Modeling – Cluster Deformers



We know that Maya is not a series of isolated modules, but rather a cohesive environment composed of many interconnected building blocks. The lines that we draw between sections of the workflow can be quite blurry. The topic of deformers is a good example of this. Deformers live in a space somewhere between modeling and animation. They can be used for both tasks equally well.

One explanation of this duality is tied to Maya's node-based architecture. Most objects contain a **transform node** and a **shape node**. We use the transform node to position, rotate, and scale an object. When we want to animate the object flying around the room, the transform node can be keyframed.

Modeling tasks (such as sculpting CVs or extruding faces) predominantly take place at the shape node level. If you need to animate the shape itself, you can keyframe individual component positions. However, this can be tedious to create and difficult to edit, due to the large number of components involved.

Here's where deformers fit in. Deformers also work at the shape-node level, so they can be used for modeling. Let's say you need to build a molehill out of a polygonal plane. Rather than gradually position hundreds of individual vertices with the **Move Tool**, you can position those same vertices very quickly using a cluster deformer.

Deformers can also be used at the shape-node level for animation. As in the above example, the individual deformer's animation values are much easier to create and edit than their hundreds of component counterparts.

Keep these things in mind as you work on the deformer exercises in this chapter. Any method you devise for reshaping an object using a deformer could become a valuable animation tool as well!

#### Cluster Deformers

Clusters are transform nodes used to control individual or sets of components. Usually these will be vertices or CVs. By clustering a set of vertices you can make it easier to select and transform those vertices during animation. Imagine having to repeatedly select the same set of points

and keep track of all their individual keyframes. Not fun at all. Clusters greatly simplify this process.

### Modeling Exercise – Clusters

1. Create a polygonal cylinder with the following options: **Radius = 5; Height = 1; Subdivisions Around Axis = 20; Subdivisions Along Height = 1; Subdivisions on Caps = 10; Axis = Y.**
2. In component mode, with pick mask set to **Vertex**, select all of the vertices on the top of the cylinder (see Figure 6.1).
3. Next, with the vertices selected, click **Deform > Create Cluster**. You'll see a new type of selection handle, in the form of a letter "c," appear on screen. This signifies that a set of components has been formed into a cluster.
4. Select the *cluster1Handle* and then translate it up a couple of units on the Y-axis. Notice how the top of the cylinder moves with it, while the bottom stays put (see Figure 6.2). Animating the Translate Y value of the *cluster1Handle* would be a much easier way to animate the cylinder's shape than setting keyframes for each individual vertex. This is especially true when it comes time to edit your keyframes in the **Graph Editor**. One curve will affect all vertices in the cluster.

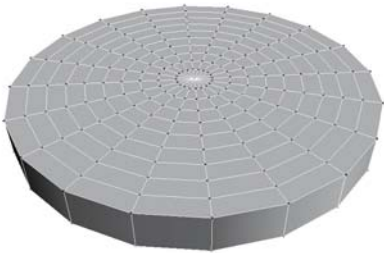


Figure 6.1: Selected vertices

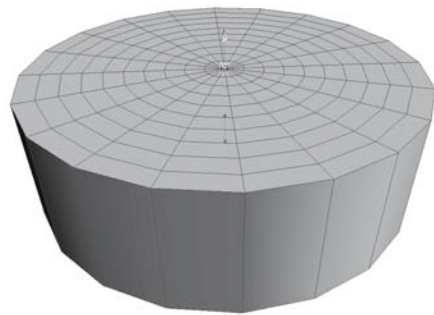


Figure 6.2: Translated cluster

## 266 6. Bringing It All Together

---

Initially, we selected a set of vertices to include in the cluster deformer's membership. So all the members move the same amount as their deformer. A very interesting and useful function of the cluster deformer is its ability to affect each vertex to a varying degree. This process of "weighting" the deformer allows us to give the deformation effect a kind of falloff.

5. We want to pick all of the vertices that form the outer ring of the cylinder. One way to do this is through the **Convert Selection** command. It can be easier to select one type of component and then transfer that selection to another type. Switch to the **Edges** pick mask, and then select one edge on the outermost ring of the cylinder top (see Figure 6.3).

6. In the **Modeling Menu Set**, click **Edit Polygons > Selection > Select Contiguous Edges**. This gives us a neat selection of the outer ring of edges.

7. To switch this selection of edges into their equivalent vertices, click **Edit Polygons > Selection > Convert Selection to Vertices**. Voila, instant selection!

8. We want these vertices to be influenced by their cluster deformer a bit less than the others in the set. Click **Window > General Editors > Component Editor**. This window allows you to edit data that are stored within each individual vertex. The values we want to change are found under the **Weighted Deformers** tab. You'll see the vertices listed by name on the left (Maya gives each vertex a unique name, such as *vx[213]*), and their deformer weight values to the right. The values can range from zero to one. A value of zero means that the component is entirely unaffected by the deformer, while a weight value of one means that the component is affected fully by the deformer.

9. Highlight the entire column of ones by click-dragging from top to bottom. Now move the value slider at the bottom of the **Component Editor** to see how the lower values

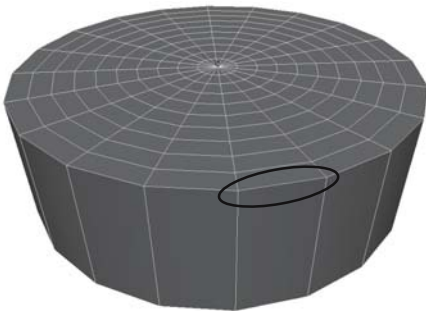


Figure 6.3: Selected edge

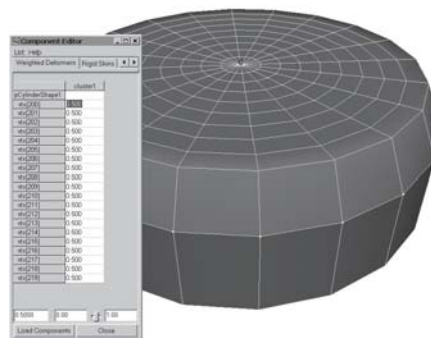


Figure 6.4: Weighted vertices

affect the vertices. Set a value of 0.5 (see Figure 6.4).

10. Close the **Component Editor**. Return to **Object Mode** and select the *cluster1Handle*. When you translate it, the outer ring of vertices moves half the distance (see Figure 6.5).

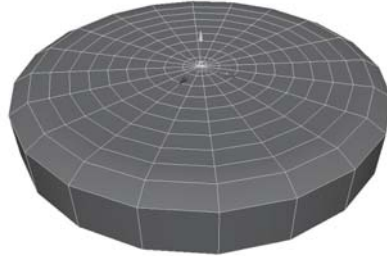


Figure 6.5: Translated weighted cluster

11. When you are using clusters as a modeling tool, the **Component Editor** can be a bit too time consuming and hands-off. A good way to sculpt cluster weights is with the **Paint Cluster Weights Tool**. (This tool is really just a better interface for doing the same things you can do in the **Component Editor**.) To use it, first select the cylinder object. Then dock the **Tool Settings** to the right of the interface by clicking the **Show Tool Settings** button in the **Status Line**. (Whenever you use one of Maya's Paint Tools, you should dock the **Tool Settings**. This acts as your home base for changing brushes, operations, and display options.) Then in the **Animation Menu Set**, click **Deform > Paint Cluster Weights Tool** (see Figure 6.6).

12. Make sure that your **perspective** view panel is in **Smooth Shaded** mode. You should see the cylinder turn from its default gray color to a white-gray-black gradient. The significance of the colors is this: white equals a value of one, or fully weighted; black equals a value of zero, or unaffected; and gray values allow for a full range between these two extremes. Notice that the top vertices are white, and they have been transformed the exact same amount as the cluster handle. The outer ring of the top row (the ones we

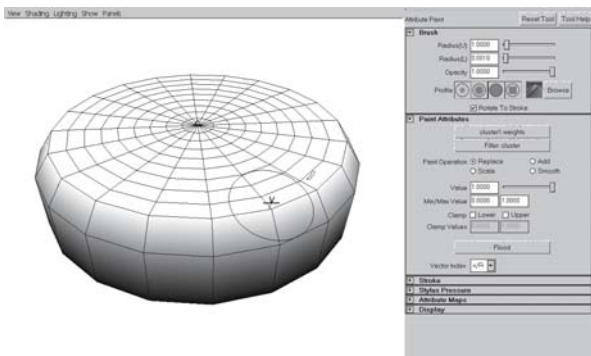


Figure 6.6: Preparing to paint cluster weights

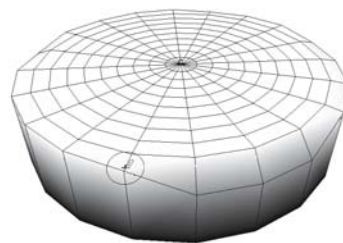


Figure 6.7: Painting weights to 1

## 268 6. Bringing It All Together

---

manually changed to a 0.5 weight value in the **Component Editor**) is painted gray. The bottom vertices of the cylinder were not included in the cluster deformer set at all, so they appear black. (It is also possible to paint a vertex black without removing it from the deformer set.)

13. The cursor has turned into a paintbrush, and it's now possible to alter the vertices' weight values by "painting" them. Let's give the outer row of vertices a full weighting of one (white) to return them to their original state. First, set the **Brush Radius(U)** to about **0.4**. Next, make sure the **Paint Operation** is set to **Replace**. This tells the brush to swap the painted vertex's old value with the new one. Set the **Value** to **1.0**, and then click-drag to paint the edge vertices white. As you paint them, they'll jump up to the same height as the cluster handle (see Figure 6.7).

14. Next, change the **Value** to **0.25** and paint a small circle of dark gray in the center of the cylinder. Again, notice how the vertices drop as their value has been decreased (see Figure 6.8).

15. Since the paint brush is being used in **Replace** mode, the painted vertices' values are being changed absolutely to the weight you specified. The **Paint Cluster Weights Tool** can also be used to determine the average between two values in order to smooth things out. Switch the **Paint Operation** to **Smooth** with a **Value** of **1**. Now paint across the entire surface. Notice the nice rounded look you can achieve with this operation (see Figure 6.9). (Note: You may want to use a larger **Brush Radius(U)** for this. The **Hotkey** to interactively change the brush size is "**b**" on your keyboard. Press **b** while **LMB**-dragging left/right to decrease or increase the brush size.)

16. Save your work.

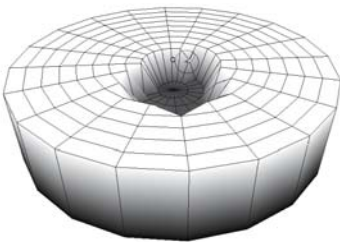


Figure 6.8: Painting weights to 0.25

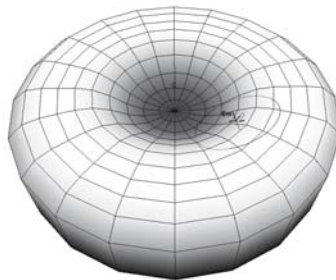


Figure 6.9: Painting smooth weights



## Modeling Exercise – Chicken Head Clusters

1. Set your project to `understanding_maya\Chapter06\modeling_deformers`.
2. Open the scene `modeling_cluster_chickenHead_begin.ma`.
3. You'll see the head of the chicken we modeled in Chapter 4 (see Figure 6.10). Use the component mode vertex pick mask to select the *chickenHead's* comb vertices (see Figure 6.11).

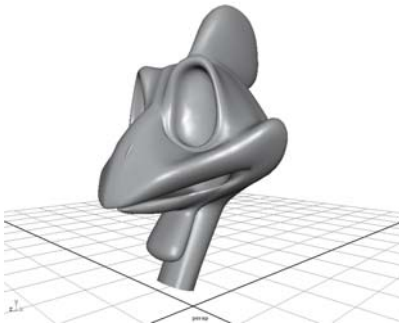


Figure 6.10: Original chicken head



Figure 6.11: Comb points selected

4. In the **Animation Menu Set**, click **Deform > Create Cluster**.
5. Now, return to object mode and pick the *cluster1Handle* object. Rename it *clusterCombHandle*. It's a good idea to rename clusters as you create them, so that it's easy to distinguish them later. Since the cluster deformer is a transform node that contains a subset of an object's components, you can perform all three transformation types to the set. Try using the **Move**, **Rotate**, and **Scale Tools** on the cluster. Notice how this affects only the deformed vertices, while leaving the rest of the model alone (see Figure 6.12).
6. Undo the transformations you just did to the *clusterCombHandle* to return it to its original shape. Now scale the handle up in all three axes to a value of **1.4**. This looks pretty good, but the transition between the

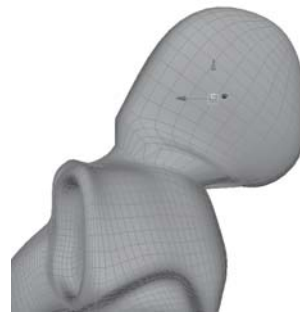


Figure 6.12: Scaled comb cluster

## 270 6. Bringing It All Together

---

affected and the unaffected vertices is too rough. We'll fix this with the **Paint Cluster Weights Tool**.

7. With the *chickenHead* object selected in object mode, dock the **Tool Settings** and then click **Deform > Paint Cluster Weights Tool**. (When painting cluster weights, pick the object, not the deformer, before invoking the paint tool.)

8. With the **Paint Operation** set to **Smooth**, set the **Value** to **1**. We want to uniformly apply the smoothing value to all of the vertices, so instead of using the paint brush, you'll use the **Flood** button. You may need to click it a few times until you're satisfied with the results, as in Figure 6.13.

This looks much better. If you want to, you can continue adding clusters to other collections of vertices. Sometimes you'll use this technique to re-sculpt a model into a different shape (see Figure 6.14). In other cases, you'll use clusters to create muscle-like animation controls.

9. Try setting keyframes on the cluster handles themselves. Animate them to create a gallery of funny faces for the chicken.

10. Save your scene when you are finished.

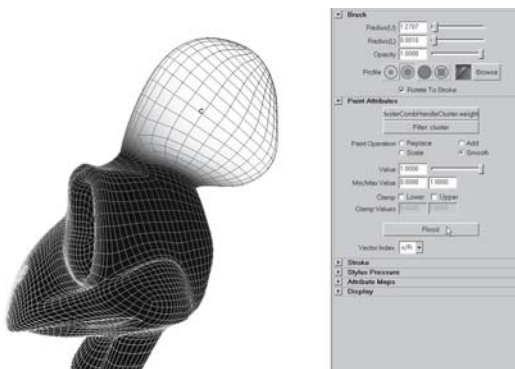


Figure 6.13: Smooth weights flooded

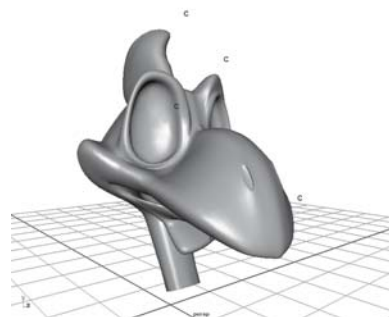


Figure 6.14: Multiple clusters for distortion

## 6.4 Animation – Constraints, Inverse Kinematics, and Skinning



When you're setting up a scene for animation you may sometimes run into this problem: "I'd like to parent one object to another, but only for the first half of the animation." Or, you might run into this one: "I need to parent only the translation of one object to another, not the rotation or scale." Well, the answer in both cases is this: Use a constraint.

### Constraints

Constraints are tools in Maya that allow you to selectively control the influence of one object over another. This control can be animated to increase or decrease over time. And the control can involve one or more of the following transforms:

- Translation – use the **Point Constraint**,
- Rotation – use the **Orient Constraint** or **Aim Constraint**, and
- Scale – use the **Scale Constraint**.

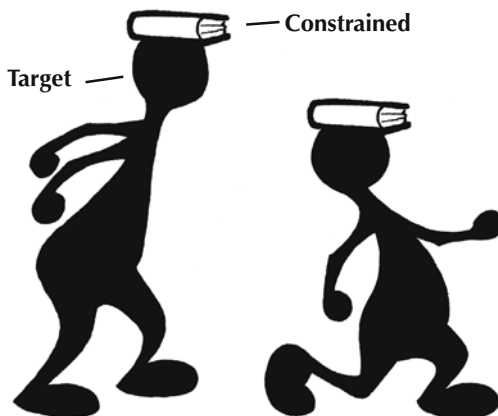


Figure 6.15: Book point constrained to character

## 272 6. Bringing It All Together

---

Set up a constraint by first selecting a target object and then an object to be constrained (see Figure 6.15). Next, choose one of the constraints. For example, select a joint on top of your character's head and then a book, and finally choose the **Point Constraint**. Now the book will be perfectly "balanced" on top of the character – unlike parenting, there is no rotation inherited by the book as the character moves around the scene; only translation is passed along.

**Orient Constraints** pass rotation information from one object to another.

**Aim Constraints** cause one object's rotation to be influenced by another object's position in space (see Figure 6.16). This is similar to the way you aim a two-node camera.

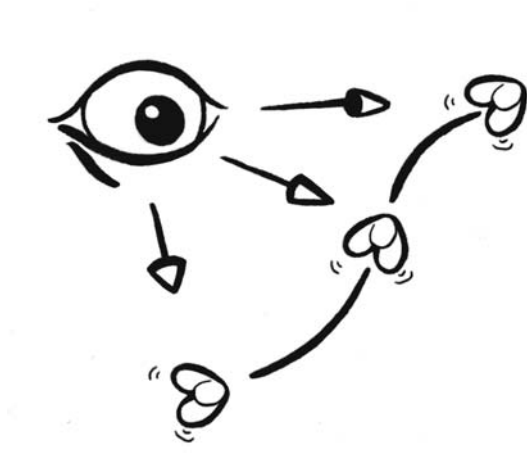
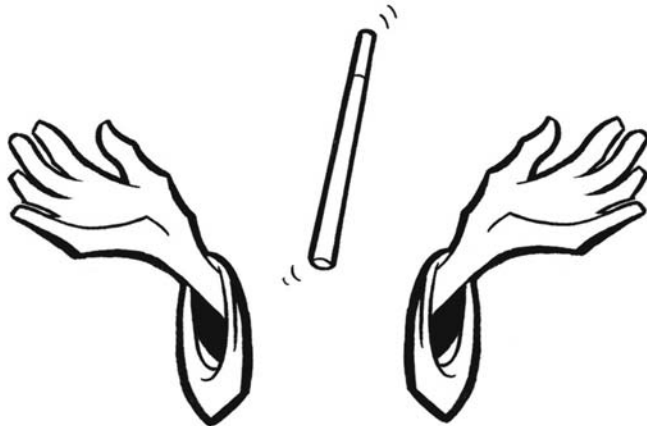


Figure 6.16: Eye aim constrained to the fly

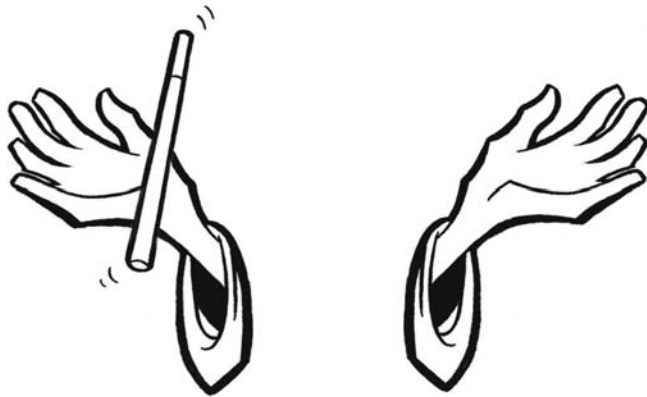
**Scale Constraints** do just what they say, constrain one object's scale to another's.

Not only is it possible to mix and match constraints, but you can also get multiple targets involved. Let's say you were to constrain the translation (**Point Constraint**) and rotation (**Orient Constraint**) of a magic wand (the constrained object) between a magician's hands (the two target objects). The wand would appear to float between his hands no matter where they went. By animating the **weights** of the constraints, you could pass the wand from hand to hand with perfect accuracy (see Figure 6.17).

a. Right hand = 0.5  
Left hand = 0.5



b. Right hand = 1  
Left hand = 0



c. Right hand = 0  
Left hand = 1

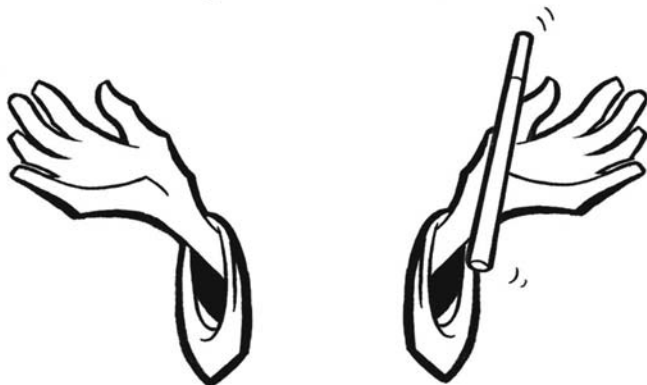


Figure 6.17: Animated point constraint weighting

## 274 6. Bringing It All Together

---

### Animation Exercise – Point Constraint Egg

1. Begin a new scene by clicking **File > New Scene**.
2. Create a NURBS sphere primitive, and name it *egg*.
3. In component mode, move and scale the CVs to shape the model into an egg.

When you're using constraints, the location of your object's pivot point is very important. Point constraints snap one object to another based upon their pivot point locations.

4. In object mode, reposition the pivot point to the base of the *egg*. Remember, this is done with the **Insert** key after having picked the **Move Tool**.
5. Next, we'll create a NURBS circle primitive, and name it *targetCircle*. This will be one of the targets of the *egg*'s point constraint setup.
6. Create a locator, and name it *targetJack*.
7. When setting up constraints, you'll first select the target – in this case, pick *targetCircle*. Then you'll select the object to be constrained – **Shift-select** the *egg*.
8. In the **Animation** menu set, click **Constrain > Point**.

Now any translation you apply to the *targetCircle* will be imparted to the *egg*. Try moving the circle around to see this in action (see Figure 6.18). You'll also notice that unlike a parent-child hierarchy, the constrained object cannot be translated by itself. The *egg* itself seems frozen to move on its own.



Figure 6.18: Egg constrained to circle

Another interesting difference between a point-constrained object and a child object is that *only* translation is transferred from the target to the constrained. If you try rotating or scaling the *targetCircle* object you'll see that the *egg* remains unaffected.

- Next let's add a second point constraint. Select the *targetJack* locator, then **Shift-select** the *egg*. Choose **Constrain > Point**. Move the *targetJack* and you'll see the *egg* move to a point halfway between the circle and the jack.

With two objects telling the egg what to do, it finds the halfway point between them. This is due to the initial weight value of 1 being set on both targets. If you change those weight values you can transfer the egg from one target to another. This technique is often used when a character picks up and sets down a prop.

- Pick the *egg*. In the **Channel Box's** Shapes section, click on the **targetJack W1** weight value and set it to zero. The *egg* moves to the circle (see Figure 6.19). Move the *targetCircle* around and you'll see that it is back to having full control over the egg.

- Set the **targetJack W1** weight to 1 and the **targetCircle W0** weight to 0. Now the jack has full influence (see Figure 6.20). Move the jack around to test this out.

Try animating the weight values to "move" the egg from one target to the other.

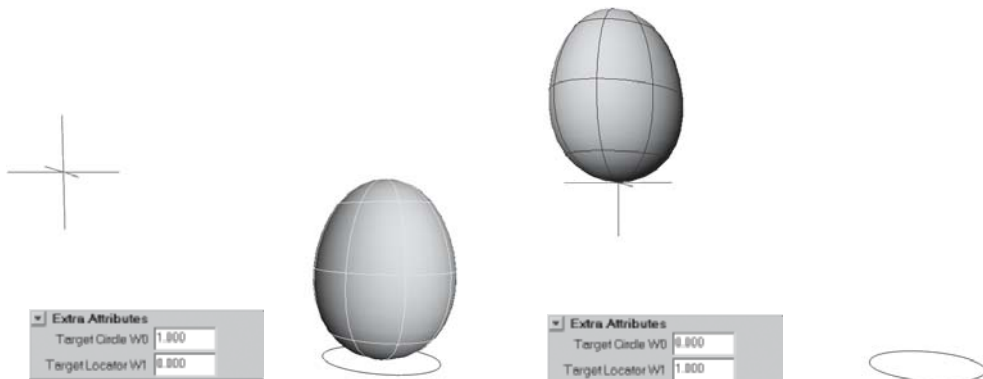


Figure 6.19: Circle weighted to 1

Figure 6.20: Locator weighted to 1

## 276 6. Bringing It All Together

---

When you create props, you'll often need to rig them for animation. A carefully engineered prop is much easier to animate than a loose collection of individual parts. You'll find that constraints are tremendously useful when constructing such assemblies.

The point constraint is used in these constructions as a way to pin things together. If you need a limited rotation on a part, aim constraints are very useful. By constraining one object's rotation to another object's position, you can make simple machines such as levers, slider cranks, and pistons.

### Animation Exercise – Multiple Constraint Pump

1. Set the project to `understanding_maya\Chapter06\animation_constrain_pump`.
2. Open the scene `constraint_pump_begin.ma`.

You'll see the pieces have been assembled for an old-fashioned water pump. The parts are there, but they need to be set up for animation. You'll make the pump functional by adding constraints.

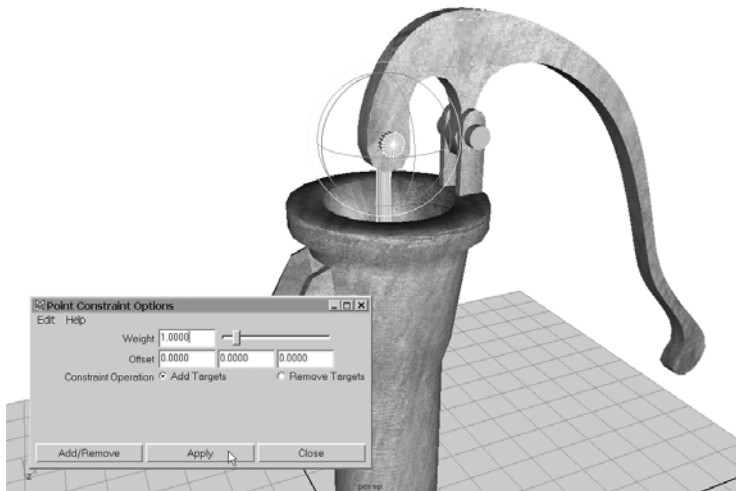


Figure 6.21: Cylinder point constrained to pin



3. Take a look at the long cylinder named *piston*. Notice that its pivot point is at the top. This will make it possible to constrain its position and rotation, as if a hinge pin were inserted through it.
4. First, we'll point constrain the *piston* to the front pin on the handle. This pin is a child of the handle. This means that the handle's rotation will be transferred onto the pin, thereby moving it up and down. Select the pin; then shift-select the *piston*.
5. Create a point constraint by clicking **Constrain > Point** from the animation menu set. This will snap the *piston* to the pin (see Figure 6.21).

Go ahead and test it by rotating the handle. You'll see that the cylinder moves up and down, but its rotation is unaffected. It looks as if it is dangling from the pin. Since we want it to appear as if there's a plunger connected to the bottom of the cylinder, we must add an additional constraint.

An aim constraint will allow you to point the cylinder at any other object's pivot point. You could create a locator for this if needed. But in our case, there is already an object in the scene that will work – the pump *base* itself.

6. Select the *base*, then shift-select the *piston*. Click **Constrain > Aim options**. Reset the settings, set the **Aim Vector** to  $Y = -1$ , as in Figure 6.22, and press **Apply**.

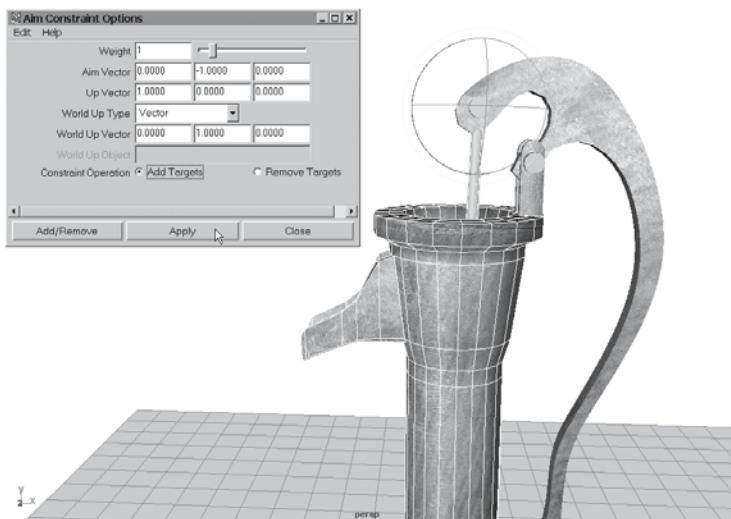


Figure 6.22: Cylinder aim constrained to pump base

## 278 6. Bringing It All Together

---

7. Now try rotating the handle up and down. The *piston* will move up and down and also exhibit a slight back and forth rotation that hinges around the pin. Save the scene.

For an added effect, try making water flow from the pump whenever the handle is pushed all the way down. One simple way to do this is by linking the transparency of a shader to the rotation of the handle using driven keys. If this shader is applied to a modeled water surface, when the pump handle is pushed the water will seem to flow forth (see Figure 6.23). For an example of this, check out the *constraint\_pump\_end.ma* scene on the CD.



Figure 6.23: Flowing water surface driven by pump rotation angle

### Inverse Kinematics

Inverse Kinematics can be used to aid you in the process of posing your character. By creating Inverse Kinematics (IK) chains, you can control a number of related joint rotations at once. You can think of IK as a type of marionetting. On a Forward Kinematic skeleton, you would have to rotate the shoulder, elbow, and wrist in order to place the hand (think of an action figure or artist's mannequin). With the addition of IK, you can pose the entire arm by moving only the hand.

IK handles are the translation component of an IK solution. What is being solved is the problem of joint rotation based on a given target. For example, if you place your hand on your head, as in

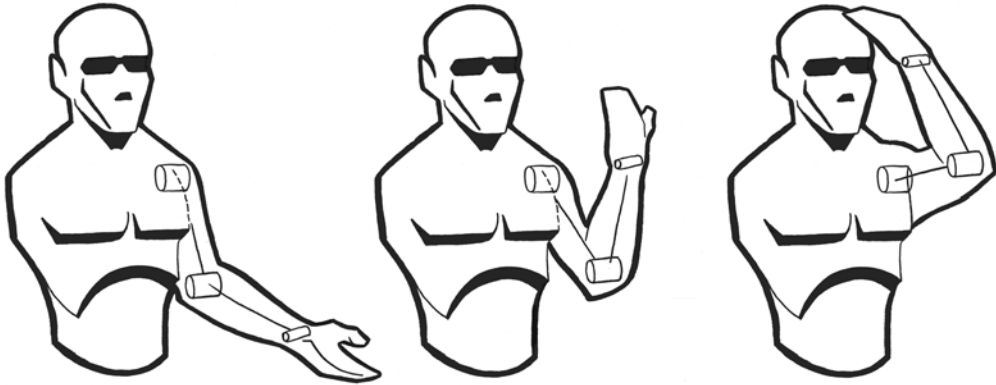


Figure 6.24: IK at wrist solves rotation of shoulder and elbow

Figure 6.24, what rotations would your shoulder and elbow joints have to go through? This is the problem the IK solver solves.

A biped skeleton will often have the pelvis as the top-level node. The hips and spine are below it in the hierarchy. Most of your skeleton will be left in FK mode. Use IK to connect the shoulders to the wrists, and the hips to ankles. The immediate benefit will be feet that stay where you tell them to, instead of floating and sliding around when the root is moved.

In practice, this skeleton is animated by keyframing the position of the IK handles and the rotation of those joints that are not part of an IK chain, such as the neck, waist, and spine. A more advanced rig may contain locator handles, constraints, joint angle limits, and set driven keys to assist in the animation process. But the general principles remain the same from the most basic to the most complex skeleton rig.

### Animation Exercise – IK Rigging the Chicken

1. Set your project to `understanding_maya\Chapter06\animation_ik_chicken`.
2. Open the scene `ik_chicken_begin.ma`.
3. You'll see the skeleton we created before in the FK rigging section. Some color-coded control curves have been added to simplify selection during animation (see Figure 6.25). (These controls are currently hidden with layers, which you'll turn on later.) Using the

**Connection Editor**, the *ct\_root\_ctrl* (“center root control”) curve controls the *root\_jt* joint’s rotation and translation.

You’ll add inverse kinematic controls to this skeleton to aid in keeping the feet planted. Make sure you are in the **Animation menu set**.

4. To add an IK solver to the leg, click **Skeleton > IK Handle Tool options**.

5. Reset the settings and then turn on the **Sticky** option. Sticky IK handles will stay put even when the skeleton hierarchy moves. Click the **Close** button.

6. Place the crosshair cursor over the *lf\_hip\_jt* joint and click. This sets the start point for the IK solver.

7. To set the end effector, place the crosshair cursor over the *lf\_ankle\_jt* joint and click.

8. You’ll see a locator called *ikHandle1* appear at the ankle. Rename it *lf\_ankle\_ik*. Pick and move this to see the IK solver at work (see Figure 6.26). The hip and knee joints rotate to give the ankle the freedom to move. Undo the movement when you’re done.

9. Repeating steps four through eight above, create another IK solution on the right side of the skeleton from the hip to the ankle.

10. Try testing the foot planting ability of the rig. Select the *root\_jt* joint and move it up and down. Notice how the IK handles stay put. If you move the skeleton far enough away, the joints will reach their limit and the feet will come off the ground. IK handles do not cause bones to stretch.

11. There’s one problem – the feet are rotating through the ground when the root is translated

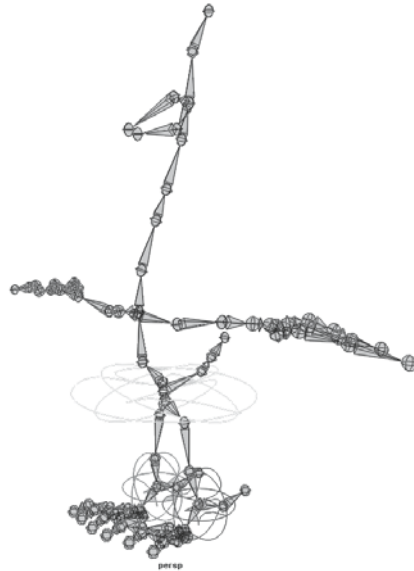


Figure 6.25: Skeleton with control curves

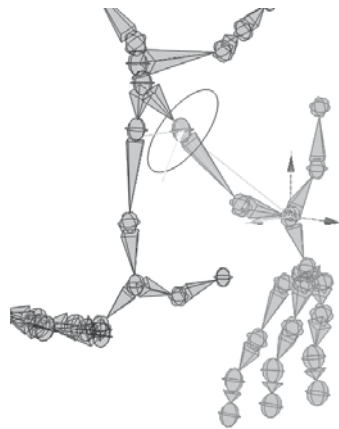


Figure 6.26: Leg IK handle

downward. To fix this, we'll add IK handles from the ankle to the ball of the foot.

12. Using the **IK Handle Tool**, click first on the *lf\_ankle\_jt* joint and then on the *lf\_ball\_jt* joint (see Figure 6.27). Name this *lf\_ball\_ik*.

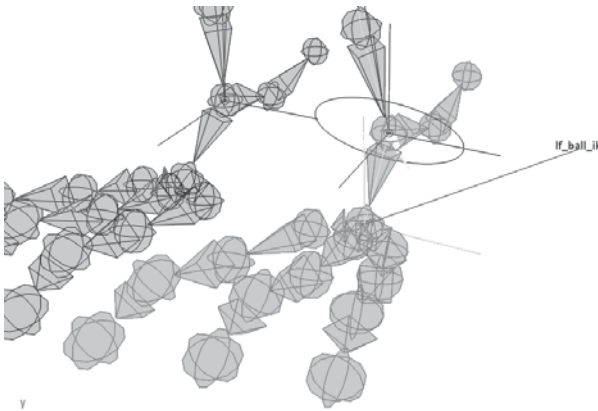


Figure 6.27: Ankle to ball IK

13. Repeat the previous step for the right foot, naming the handle *rt\_ball\_ik*.

14. Now, when you move the skeleton up and down from the *ct\_root\_ctrl*, you'll see the feet are planted firmly on the ground.

15. Try rotating the *lf\_ankle\_ik* handle. Nothing happens! This is because the rotate plane IK solver has its own **Twist** attribute to control rotation of the IK plane. In the **Channel Box**, try changing the **Twist** channel with the virtual slider technique (highlight the name field, as in Figure 6.28, then **MMB**-drag inside a view panel) This will knock the knees in and out, as in Figure 6.29.

With the two IK handles at the ankle and ball, you'll want a single control to animate them. Grouping them together would work, but might be difficult to select. For this reason, it is better to parent them to an easily selected object such as a locator or curve.

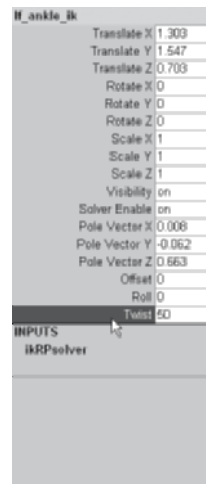


Figure 6.28: Twist channel

## 282 6. Bringing It All Together

---

16. Some additional control curves have already been created for you. They are in the two layers named **rt\_layer** and **lf\_layer**. Make both layers visible by clicking in the empty boxes at the beginning of the layer (they'll toggle to a "V" when the layer is visible).

17. Parent both of the left side IK handles to the *lf\_foot\_ctrl* curve. Then, rotate and move the control curve to test it out (see Figure 6.30). Undo the movement when you are done, returning the control curve to zero translation and rotation.

18. Repeat this for the right side as well, using the *rt\_foot\_ctrl*.

19. For a simple animation test, try setting keyframes on the *ct\_root\_ctrl* curve to get the chicken jumping up and down while flapping its wings.

20. When you are done with the animation test, save your scene.

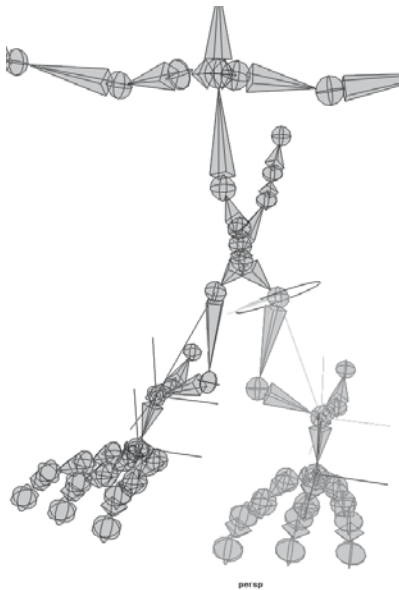


Figure 6.29: Twisting the leg IK

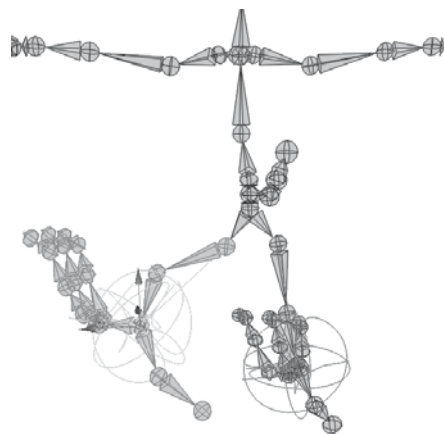


Figure 6.30: Foot control curves

## Character Skinning

Skinning is the process of connecting a character's meat to his bones. Getting the skeleton to control the character's mesh can be tricky. Using Maya's **Smooth Bind** command and **Paint Skin Weights** tool is one of the best ways to approach this process.

Any time a single mesh is controlled by a series of joints, you are performing a deformation on the mesh. To control the deformation, you can change the amount of influence each joint has on a given point on the mesh. This is a "weighting" process, much like the way we alter **cluster** deformer influences. In fact, when a character is bound to a skeleton, Maya creates "skin clusters" that tell each joint how much influence they have over the skin points.

Here is a process overview for skinning a character.

- Delete the history on the character mesh. All the modeling and UV mapping operations that lead to this point can be safely collapsed to a single shape node in order to speed up the character interaction.
- Keyframe the entire rig in a neutral bind pose at frame -10. This allows you to easily reach the bind pose later if you need to make changes.
- Smooth bind the mesh to the skeleton.
- Paint the skin cluster weights to create the best possible deformation.

## Animation Exercise – Skin Binding

1. Set your project to `understanding_maya\Chapter06\skinbind_chicken`.
2. Open the scene `bind_chicken_begin.ma`.
3. Set your view to a single **perspective** panel and dock the **Tool Settings** to the right side of the interface. (You can do this with the icons in the upper right of the interface, or by clicking **Display > UI Elements > Tool Settings**.)
4. Enter -10 as the beginning frame of the **range slider**. Go to frame -10 on the time slider. You'll bind the character on this frame so that you can always return to the bind pose later without affecting your animation.

## 284 6. Bringing It All Together

---

5. You'll want to set a keyframe on the entire pose at this point. All of the keyable objects end in the suffix "ctrl", which makes them easy to keyframe with a short MEL script. Enter the following into the **command line** and then press **Enter**.

```
setKeyframe {"*ctrl"};
```

This has set a keyframe on all of the control curves; you'll want to **MMB**-drag this script to your shelf for use in animation later.

6. Now you can bind the skin to the skeleton. In wireframe mode, select the *root\_jt* and then **shift-select** the *PolyChickenSkin* object. (Based upon the settings we'll use in the next step, you could have selected any joint. But a little superstition never hurt anyone, right?)

7. From the **Animation menu set**, click **Skin > Bind Skin > Smooth Bind options**. Reset the settings and then change the **Max Influences** to 2. Click the **Bind Skin** button (see Figure 6.31). Notice that **Bind to Complete Skeleton** is being used for the bind.

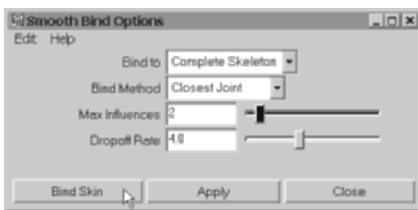


Figure 6.31: Smooth bind options

That's it, you're done! OK, not quite – the skin is now bound to the skeleton but we'll need to make some adjustments before it's perfect.

Try moving some of the controls around to pose the chicken. (Turn on their visibility in the **Layer Editor** first.) You'll see that some areas look quite good – like the neck and wrists. During the bind calculations, there was very little ambiguity about to which joints the vertices in these areas should be weighted.

Other areas show major problems – such as the head and legs areas. Here we'll be doing the most weight painting to fix things. These areas are tricky because there are many possible joints to which each vertex can be bound.



Here are a few steps to take when painting the skin weights on a character.

- Start out with some simple range-of-motion keyframes. These will allow you to test the skinning without leaving the **Paint Weights Tool**.
  - Only paint one-half of the character – left or right. The other half can be corrected later with the **Mirror Skin Weights** command.
  - Weight the vertices at the extremities first, and then move inward.
  - When you like how a particular joint’s influences are behaving, lock it. This prevents Maya from changing things later when your weights are normalized.
8. Move the time slider to frame 5 and pose the left arm up (see Figure 6.32). Key the pose with the MEL button we made in step 5.
  9. On frame 10, lower the arm and key it (see Figure 6.33).
  10. On frame 15, move the *lf\_hand\_ctrl* in toward the chest to get a bend in the elbow (see Figure 6.34). Key this pose.

These poses will allow you to easily scrub the time slider while you’re painting weights in order to test the skinning.

11. To begin painting the weights (on frame 1), select the *PolyChickenSkin* object and then click **Skin > Edit Smooth Skin > Paint Skin Weights Tool** (see Figure 6.35).

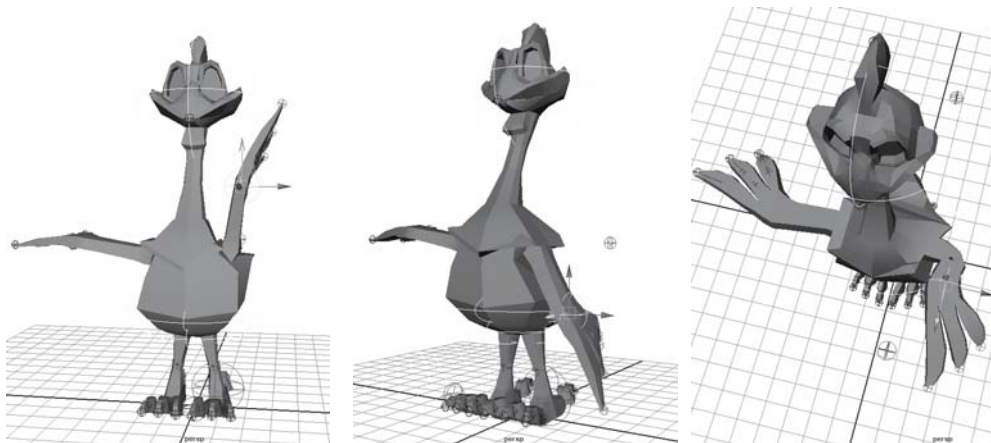


Figure 6.32: Frame 5 arm raised    Figure 6.33: Frame 10 lowered    Figure 6.34: Frame 15 arm bent

## 286 6. Bringing It All Together

Make sure the **perspective** view panel is in shaded mode. You'll see white shading where the current joint has an influence of one and black shading where that joint has an influence of zero. In the **Tool Settings** box you'll see a list of **Influences**, which are all of the joints in the skeleton. The joint you choose from the influence list is the one you'll be able to paint (see Figure 6.35).

One problem you can fix easily is the underarm area.

12. Select the *lumbar\_jt* from the list. (Looking at this list, you can tell how important it is to name your joints when you begin character construction!) Most of the chest and belly vertices are white, meaning they are influenced by this joint. The underarm vertices, however, are weighted to shoulder and elbow joints.

13. With the **Paint Operation** set to **Replace**, and the **Value** set to **1**, go to frame 5 and paint over the underarm vertices. As you paint, they'll move to the position the *lumbar\_jt* places them (see Figures 6.36 and 6.37).

14. Move on to the *thoracic4\_jt* to fix the collar area above the shoulder. With the time slider at frame 10, paint the weight back onto the collar vertices.

15. At frame 15, pick the *lf\_shoulder\_jt* influence. Set the operation to **Smooth** and begin painting the inner elbow area. The smoothing will blend the weights so they're much more even (see Figures 6.38 and 6.39).

The elbow is looking good. As you continue to paint other weights in this area, you won't want to accidentally change the elbow's weighting. You can avoid this problem by "holding" the weight values of the *lf\_elbow\_jt*.

16. In the **Tool Settings** box, click the **Toggle Hold Weights on Selected** button. You'll see the word **(Hold)** appear after the joint name.

You can quickly apply all of the weight values you've painted to the right half of the character. This will save you a lot of time in the long run. You'll put the character back into the bind pose

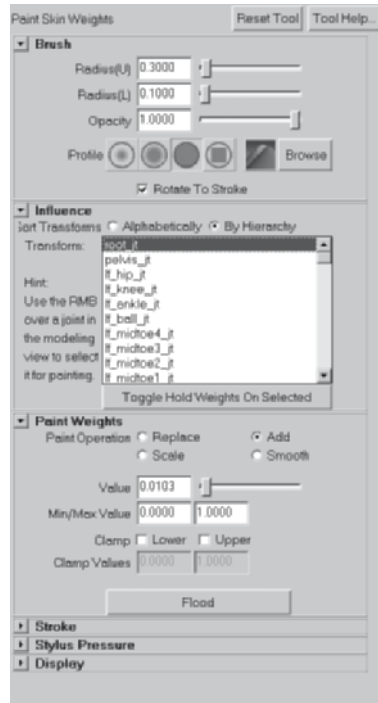


Figure 6.35: Paint tool options

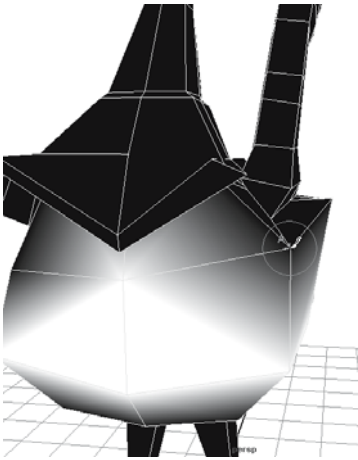


Figure 6.36: Before painting influence

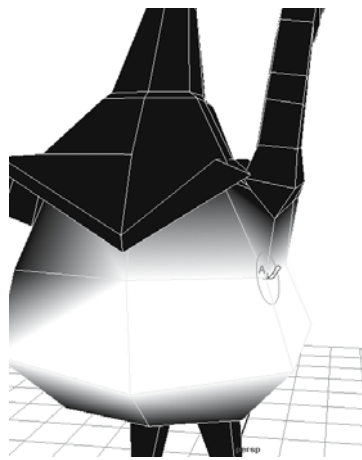


Figure 6.37: After painting influence

and then apply the **Mirror Skin Weights** command to the skin.

17. Set the time slider to -10. This returns the chicken to the bind pose.

18. Use the **Select Tool** (q on the keyboard) to pick the *PolyChickenSkin* object.

19. Click **Skin > Edit Smooth Skin > Mirror Skin Weights** options. The **Mirror Across** axes should be set to **YZ** and the **Direction** box checked for **Positive to Negative**. This will mirror values from left to right. Click the **Mirror** button.

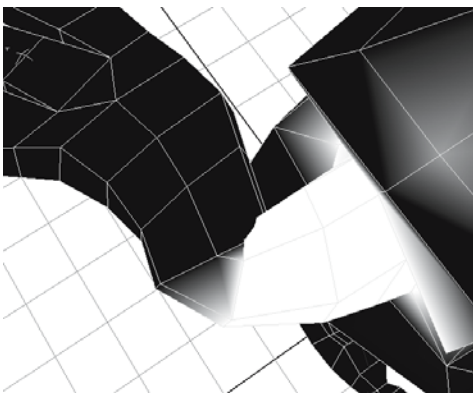


Figure 6.38: Before paint smoothing

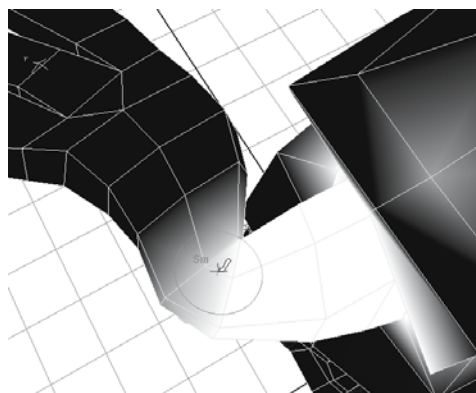


Figure 6.39: After paint smoothing

## 288 6. Bringing It All Together

---

20. To test out the mirroring, go to frame 15 and then move the *rt\_hand\_ctrl* to mimic the position of the *lf\_hand\_ctrl*. The skinning on both elbows is the same!

Using the techniques above, you'll now paint the weights for the legs. Start by adding weight to the *pelvis\_jt* for the areas that are caving in. Smooth the weights in the knee area, first on the *lf\_hip\_jt*, then on the *lf\_knee\_jt*. Mirror weights over to the right side when you are done.

The head, comb, and beak areas need work as well. The easiest way to start is by replacing the *skull\_jt*'s weight with a value of 1 for all the vertices in the rigid parts of the chicken's head.

21. Using the *ct\_head\_ctrl*, set keyframes on frames 1 and 5 for the **Beak\_open** channel (pose it opened and closed) and the **Comb\_bend** channel (pose it front and back).

22. Select the skin and switch to the **Paint Skin Weights** tool.

23. Select the *skull\_jt* influence from the list.

24. Using the **Replace** operation with a **Value** of 1, paint over all the rigid parts of the skull and eye sockets, as in Figures 6.40 and 6.41.

25. It's very helpful to scrub the time slider back and forth between your two keyframes in order to evaluate the skinning. There may be problems that are unnoticeable when you're

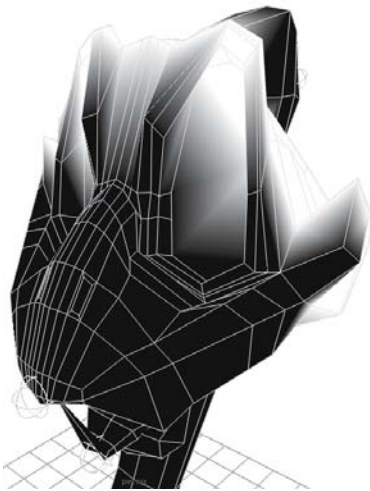


Figure 6.40: Before replacing weights

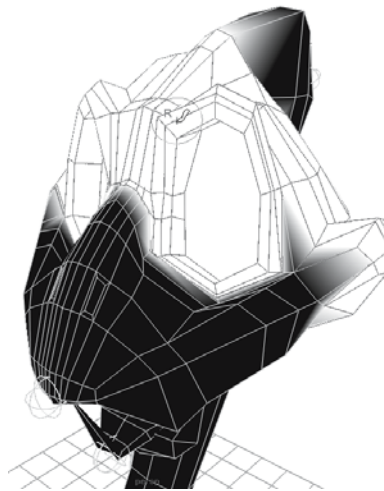


Figure 6.41: After replacing weights

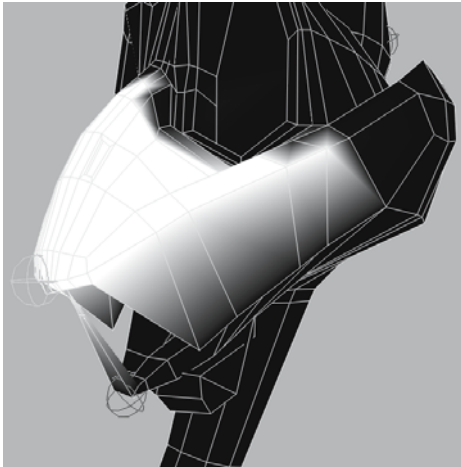


Figure 6.42: Problem beak

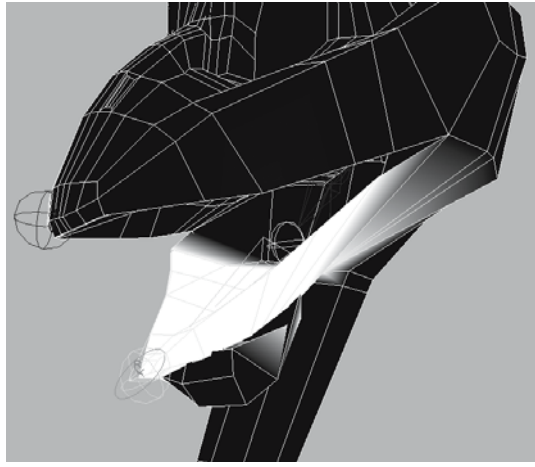


Figure 6.43: Corrected beak weights

working on a static frame. These will usually reveal themselves as you play back the animation.

26. The area where the comb connects to the skull is a bit distorted right now. Select the *combBase\_jt* from the influence list and use the **Smooth** operation to ease the weighting in this area.

27. To fix the mouth area, select the *beakBase\_jt* influence and then **Add** value back into the vertices that are currently moving with the jaw. The **Add** operation can be set to a small value, such as 0.2, so that you can gradually shift the weighting to the proper joint. This works well in areas that should share influence like the corners of the mouth.

28. The chicken has developed a bit of a terrifying fang on her lower beak! Fix this by selecting the *jawBase\_jt* and **Replacing** the weight of the stray vertex with a **Value** of 1 (see Figures 6.42 and 6.43).

You should continue testing the character's range of motion in order to refine the skin weights at this point.

One benefit of skinning the low polygon character is that the weighting will transfer to the smoothed polygon mesh. So, you've weighted the "light" model, which has relatively few vertices to deal with, and is quick to pose. Later, you can simply smooth the light model into the "heavy" model before you render the scene.

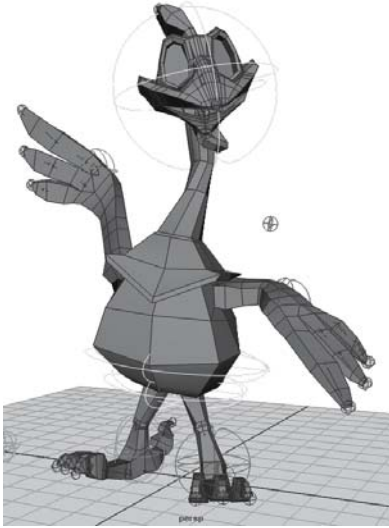


Figure 6.44: Posed low poly model



Figure 6.45: After smoothing the mesh

29. You can try out this indirect binding method now by selecting the *PolyChickenSkin* object and then clicking in the **Modeling Menu set** on **Polygons > Smooth options**. Set the divisions to 2 and then click the **Smooth** button (see Figures 6.44 and 6.45).

30. The bound mesh is looking good! Note: Your rig is much heavier and slower to update now, so you should Playblast your scene to see true timing when testing animation. Save your scene.

## 6.5 Shading – 3D Paint



Applying paint to your models is a great way to get just the look you want. Hard-edged models, like many props and sets, are fairly easy to paint in a 2D paint package such as Photoshop. Complex organic models, such as characters, are very difficult to paint in 2D. Think of the odd distorted appearance of the UV layout we did in Chapter 5. It's almost impossible to predict where you should paint on such a layout in order to get the stroke where you want it. There is a solution to this problem: the **3D Paint Tool**.

The **3D Paint Tool** enables you to drag brush strokes across your model in an intuitive way. These strokes can add color or grayscale values to virtually any material attribute that can accept a file texture. Color, specularity, bump, transparency, diffuse, and many others can accept paint from this tool.

One crucial step in preparing a polygonal model for painting is the UV layout process. Since we've already prepared the chicken model, we'll paint it in this exercise.

### Shading Exercise – 3D Painting

1. Set your project to `understanding_maya\Chapter06\shading_3dpaint_chicken`.
2. Open the scene `chicken_paint_begin.ma`.

A Lambert material has been applied to the chicken model with a blank file texture in its color attribute. This is the texture map onto which we'll be painting.

3. Dock the **Tool Settings** box into the right side of the interface by clicking **Display > UI Elements > Tool Settings**. This is where you'll control the brush size and paint colors as you work.
4. Select the `PolyChickenSkin` object and then click **Texturing > 3D Paint Tool**. (Be sure that you are in the **Rendering menu set**.) Click the **Reset Tool** button (see Figure 6.46).
5. Set the **perspective** panel to **Hardware Texturing** mode by pressing **6** on the keyboard.
6. In the **3D Paint Tool Settings** box, choose a color for your brush stroke. (This is done with the first **Color** swatch; the **Flood Color** can be used like a paint bucket to cover the entire model or section.)

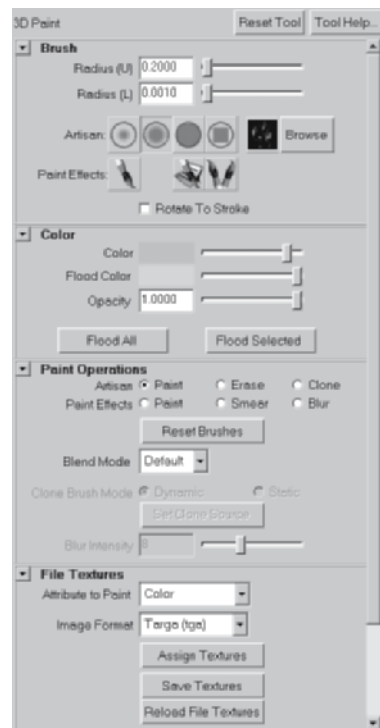


Figure 6.46: 3D paint settings

## 292 6. Bringing It All Together

---

7. Click and drag a paint stroke across the model (see Figure 6.47). The red outline will tell you how large your brush is. If needed, change this with the **Radius (U)** setting.

On this model, it may be helpful to paint symmetrically. This way, if you painted detail on the left side of the beak, it would automatically appear on the right side as well.

8. To turn on symmetrical painting, open the **Stroke** section of the **Tool Settings** and choose the X-axis of **Reflection**.

9. Now, paint another stroke. You'll see it mirrored across the X-axis, as in Figure 6.48.

Continue painting the chicken in this manner. You will probably want to start with a large brush in order to block out the major color sections. Then switch to a smaller brush to add details (see Figures 6.49 and 6.50).

Occasionally, you should click the **Save Textures** button in the **File Textures** section of the **Tool Settings** box. This will save the file texture to the **3dPaintTextures** directory of your project. This texture can be opened into Photoshop for fine-tuning if you like.



Figure 6.47: Painting the model

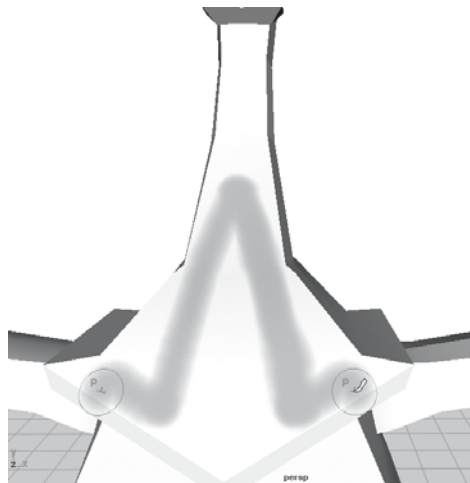


Figure 6.48: Reflection paint



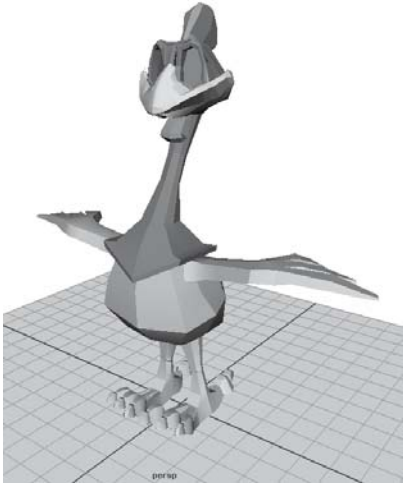


Figure 6.49: Painted model

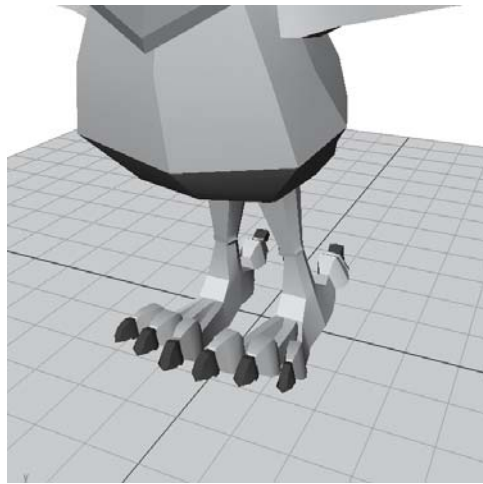


Figure 6.50: Adding details

In keeping with the low- to high-polygon workflow, this character is easier to paint initially in low-polygon mode. The painted textures will flow very nicely onto the high-polygon mesh when you increase the smoothing level.

10. Dock the **Channel Box** into the right side of the interface. Click on the *PolyChickenSkin* object's Input node for the **polySmoothFacel**. Increase the **Divisions** from 0 to 2 (see Figure 6.51).

11. Save your scene.

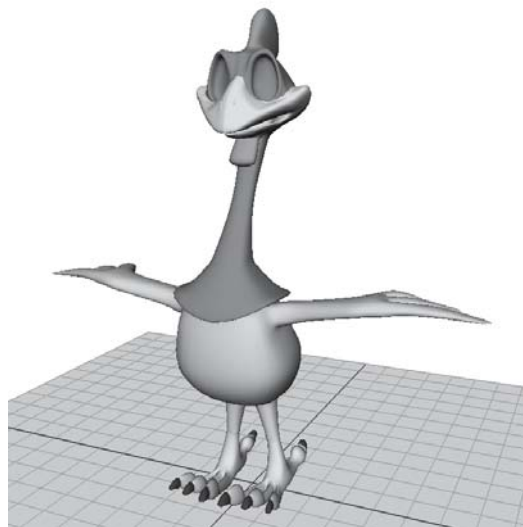


Figure 6.51: Smoothed model

### 6.6 Lighting – Final Gathering in Mental Ray



You can create very realistic lighting effects using Maya and mental ray. mental ray is a powerful renderer made by mental images that was added to Maya beginning with version 4.5. With the release of Maya 5.0, mental ray was fully incorporated into the Maya lighting/rendering pipeline, and has been further enhanced in Maya 6.0. The entire mental ray renderer is too large a subject to cover here (see the excellent book *Rendering with mental ray*, by T. Driemeyer for more information), but we'll take a look at one mental ray feature called Final Gathering.

Final Gathering creates indirect lighting effects by simulating bounced light coming off the object in your scene. Final Gathering doesn't even require that you place any "real" Maya lights in your scene. You'll create a plane or dome surface from which most of your light will emanate. This works like the extremely diffuse lighting of an overcast day. Note: This project requires Maya 5.0 or higher, or the mental ray plug-in for Maya 4.5.

#### Lighting Exercise – Final Gathering

1. Set your project to `understanding_maya\Chapter06\lighting_final_gather`.
2. Open the scene `final_gather_plane_begin.ma`.

The pursuit plane model is parked, waiting to be lit for a product shot. The first step in using Final Gathering is to create a dome or card that will give off light.

3. Create a polygon sphere and scale it by a factor of 8 on all axes. Name it *dome*.
4. Move the *dome* up above the scene about 9 units on the Y-axis.
5. Switch to the faces component mode and delete the bottom half of the sphere.

The normals of a sphere point outward by default. In order to send the Final Gather light rays in toward the scene we'll need to reverse the normals.

6. Back in object mode, select the sphere and click **Edit Polygons > Normals > Reverse**. (This menu is found in the **Modeling Menu set**.) You can confirm that the nor-

mal direction has been properly reversed by clicking **Display > Polygon Components > Normals**. When you're done, toggle the normals off.

In order to use the *dome* as a light, you should assign a new material that will control the color and intensity of the "light."

7. RMB-click on the *dome* and choose **Materials > Assign New Material > Lambert**. In the **Attribute Editor**, set the **Incandescence** to white and the **Diffuse** to 1 (see Figure 6.52).

To set up the render, you will switch to the mental ray engine. The **Render Globals** window has settings for turning on Final Gathering and adjusting its quality settings.

8. Open the **Render Globals** window. At the top, switch the **Render Using** option to **mental ray**.

9. The **Common** tab holds values that are shared by all rendering engines. The right tab contains settings specific to the chosen renderer. In Maya 4.5 this is limited to the Maya Software and mental ray engines. In Maya 5.0 and 6.0, the choices are Maya Software, mental ray, Maya Vector, and Maya Hardware.

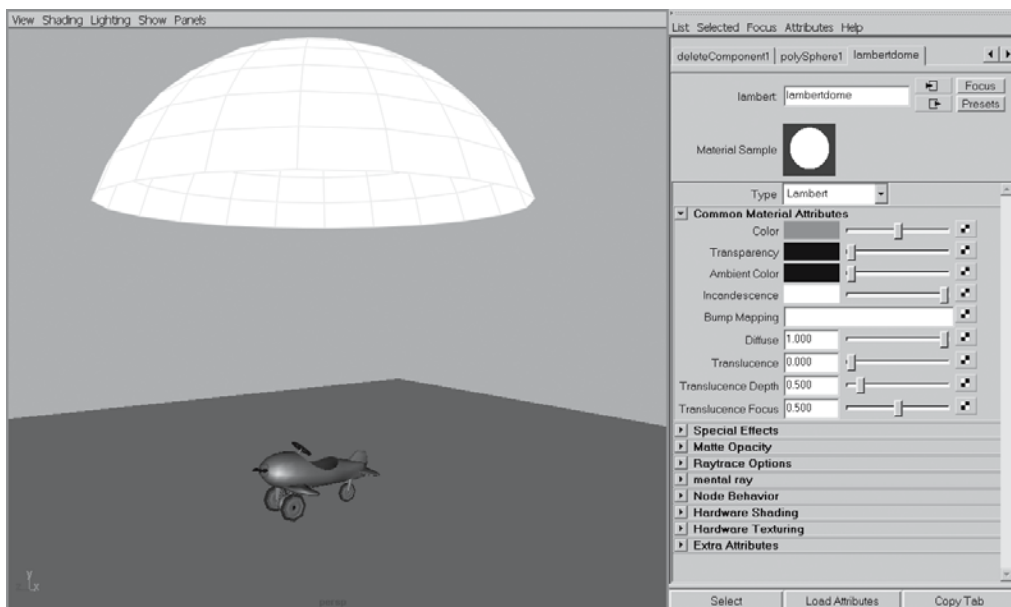


Figure 6.52: Light dome and its material attributes

## 296 6. Bringing It All Together

10. Set the **Resolution Preset** to **Full 1024**.

11. Switch to the **mental ray** tab. Open the **Final Gather** section and turn it on by checking the box in front of **Final Gather**. The number of **Final Gather Rays** will affect the quality vs. speed of your render. Try a lower setting at first, such as 750. (The typical value here is between 500-2000 rays.) See Figure 6.53.

12. The **Min Radius** and **Max Radius** values are used to optimize the light calculations based on the physical size of your scene. A good rule of thumb is to set the minimum to 1% of the area you want to illuminate and the maximum to 10% of the area. In this scene, the area around the pursuit plane is about 8 units. Set the **Min Radius** to **0.08** and the **Max Radius** to **0.8**.

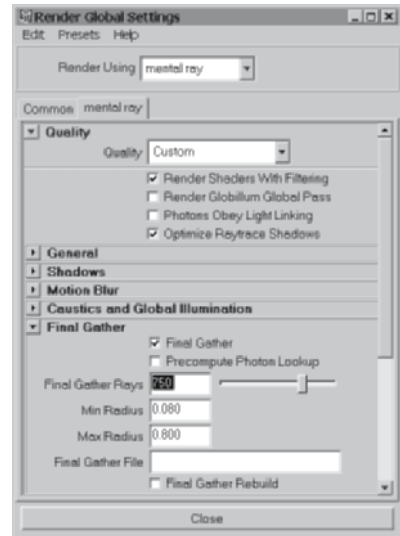


Figure 6.53: mental ray settings

Render the frame through *camera1* to test the Final Gathering. This looks good, but is too dark underneath the plane, as in Figure 6.54. The ground should be reflecting more light. One way to control this is with the **Irradiance** setting of the ground's material.

13. Open the **lambertground** material into the **Attribute Editor**. In the **mental ray** section, increase the **Irradiance** value to a medium-dark gray.



Figure 6.54: Dark rendering



Figure 6.55: Increased ground irradiance

This time when you render the scene, the ground will bounce more light onto the pursuit plane's underbelly, as in Figure 6.55. The scene is still a bit dark, and is not currently giving us any realistic color-bleed from the red surfaces. By adding a true Maya light source to the scene, you'll fix both of these problems.

14. Create a point light and position it as a typical key light. To avoid extra highlights on the shiny surfaces of the plane, turn off the **Emit Specular** component of the point light.

That's much nicer looking! Notice the red color-bleed on the ground near the wheels. The



Figure 6.56: Added point light

## 298 6. Bringing It All Together

---

overall lighting level has been increased, too, as in Figure 6.56. Now, you'll want to fine-tune some of the quality settings for your finished render.

15. In the **Render Globals** window, choose the **mental ray** tab (see Figure 6.57). In the **General** section, set the following settings:

- **Raytracing** = on;
- **Max Reflection Rays** = 2;
- **Max Refraction Rays** = 0;
- In the **Sampling Quality** section, increase the **Max Samples** to 2.

Render the final image (see Figure 6.58 and Color Plate IX). You can continue to play with some of the sample settings and increase the **Final Gather Rays** to refine the quality.

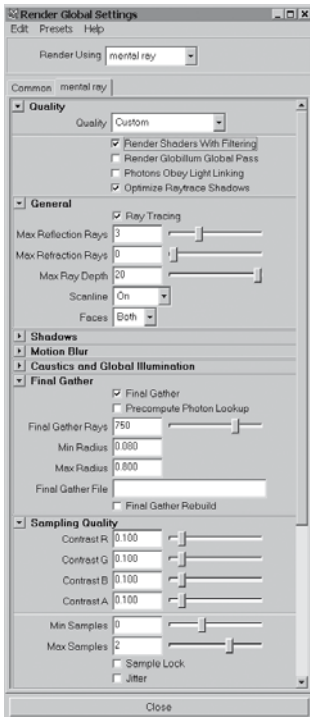


Figure 6.57: mental ray settings



Figure 6.58: Final rendering

## 6.7 Rendering –Motion Blur



Film cameras aren't perfect. In medium- to low-light situations, fast-moving objects can exceed the ability of the shutter to capture their movement on a single frame. This will cause an effect known as "motion blur." We are so accustomed to this effect that its absence can be very noticeable, so motion blur is often artificially added to 3D animation.

One of the last things you'll set up before rendering your final animation is motion blur. Maya's motion blur works by looking at the pixels on the current frame, and then comparing them with the pixels a few frames backward and forward on the timeline. If there is a great change in position between frames – high speed motion – Maya will create a blurry edge behind and in front of the moving object.



Figure 6.59: Motion blur

### Rendering Exercise – Motion Blurred Chicken

1. Set your project to `understanding_maya\Chapter06\rendering_motionblur`.
2. Open the scene `blur_chicken_begin.ma`.

The chicken has been animated to perform a quick head turn (see Figure 6.60). We'll do a quick batch render to take a look at the animation without any blur applied.

3. Open the **Render Globals**, and change the following settings:

- a. **File Name Prefix** = `noblur_chick`;
- b. **Frame/Animation Ext** = `name#.ext`;
- c. **End Frame** = 24;



Figure 6.60: Chicken head turn

## 300 6. Bringing It All Together

---

- d. **Frame Padding** = 4;
  - e. **Camera** = Camera1;
  - f. **Resolution** = 640x480;
  - g. **Anti-aliasing Quality** = 3D Motion Blur Production.
4. Close the **Render Globals** and then click **Render > Batch Render** in the **Rendering Menu set**. The batch render will prompt in the **Command Feedback Line** when it's finished.
  5. Take a look at your render using **fcheck**. If you want to launch **fcheck** from within Maya for Windows, you can create a MEL shelf button with the following code.

```
system ("start fcheck.exe");
```

This type of movement will look more natural with a bit of motion blur. You don't need to change the animation to create motion blur; the software renderer will simulate motion blur by averaging the movement of pixels over a range of frames and subframes.

6. Open the **Render Globals** and go to the **Motion Blur** section. Turn on the **Motion Blur** checkbox (see Figure 6.61). In Maya 5.0 and higher, this is in the **Maya Software** tab.
7. Choose the **3D Motion Blur Type**, which is slower but more accurate than the simple **2D** type. With 3D motion blur checked on, increase the **Blur by Frame** value to 2.
8. Set the **File Name Prefix** in the **Render Globals** to be **blur\_chick**.

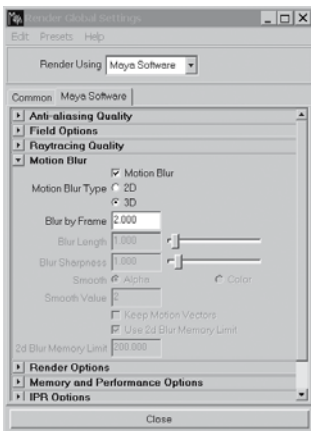


Figure 6.61: Motion blur settings

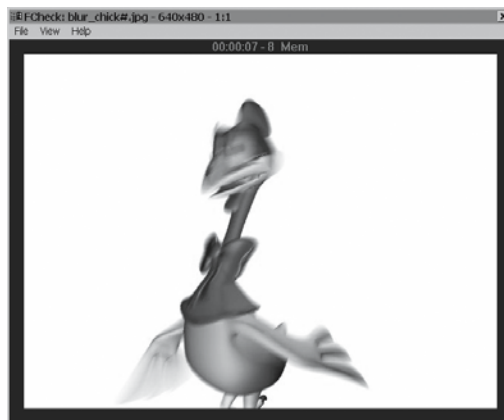


Figure 6.62: Motion blurred frame



9. Batch render the scene by clicking **Render > Batch Render**.
10. When the render is done, open it in **fcheck**. This is much better (see Figure 6.62)! Save your scene.

## 6.8 Layout – Importing Elements

Having created all of these different parts, it's time to combine them into one scene. It is often easier to create your models, work on animation setup, and create materials in individual scenes, rather than do all of your work in a single gigantic scene file. But now it's time to lay out all of the elements into the combined scene. There are two ways of doing this: importing and referencing.

Importing a scene allows you to bring a full copy of one scene into another. This is sort of like merging them. In the end you will have a scene that can stand by itself with no dependencies on outside scene files.

Another type of scene merge is called “referencing.” Referencing works as does importing, but with one important distinction – a referenced scene is only a link to a pre-existing scene elsewhere on your hard drive or network. A small node is added to your scene that points to the full external scene. This can be helpful if you need to work on the layout while someone else is still working on the elements, since updates to the referenced scenes will automatically feed into the larger scene. In this exercise, however, all of the elements are finished, so importing will be the simpler way to go.

### Layout Exercise – Importing Elements

1. Set your project to `understanding_maya\Chapter06\importing_barn`.
2. Open the scene `importing_begin.ma`.
3. The *barn*, *ground*, and *sky* are together in this scene, waiting for the other elements to be imported.

## 302 6. Bringing It All Together

4. In order to bring in the other elements, click **File > Import ...** (see Figure 6.63).

5. From the **Import** window, choose the **chicken.ma** file, then click the **Import** button. The scene will be imported into the current scene (see Figure 6.64), including any lights, cameras, materials, animation, and layers. The layers are very helpful in managing the complexity of a large scene, once many elements have been imported.

6. Import the following scenes:

- **bottles.ma** (see Figure 6.64),
- **bucket.ma** (see Figure 6.65),
- **butterfly.ma** (see Figure 6.66),
- **egg.ma** (see Figure 6.67),
- **pitchfork.ma** (see Figure 6.68),
- **plane.ma** (see Figure 6.73),
- **pump.ma** (see Figure 6.69),
- **saddle.ma** (see Figure 6.70),
- **slide.ma** (see Figure 6.71), and
- **wagon.ma** (see Figure 6.72).

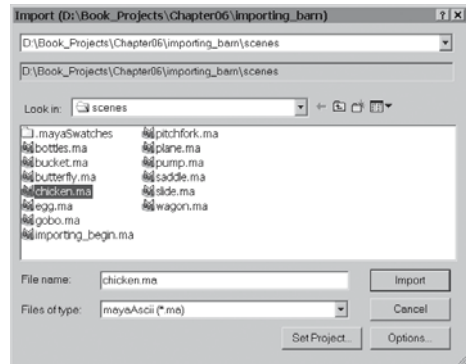


Figure 6.63: Import window



Figure 6.64: Chicken and bottles imported



Figure 6.65: Bucket



Figure 6.66: Butterfly



Figure 6.67: Egg

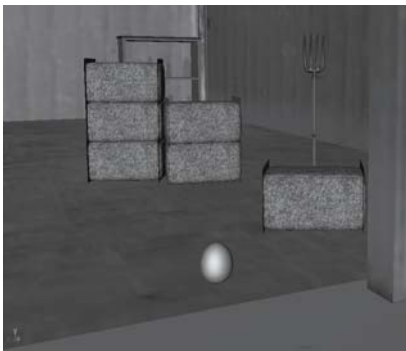


Figure 6.68: Bales and pitchfork



Figure 6.69: Pump



Figure 6.70: Saddle

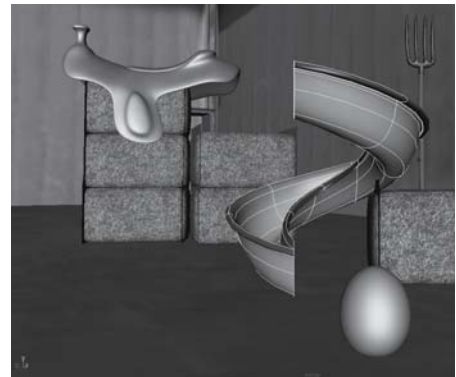


Figure 6.71: Slide

## 304 6. Bringing It All Together

---

7. You can now arrange the different elements however you like (see Figure 6.73 and Color Plate VII). Be sure to choose the top-level node of any parts that belong to groups, otherwise you may pull them apart. One way is to switch to **Hierarchy** mode while you lay out the scene.

8. The rest is up to your imagination! Animate the character, props, and camera any way you like.

9. When you are done with animation, render out a frame to check the lighting. You may find that it is necessary to limit the effect of certain lights to only some objects, so that the scene doesn't get blown out with light. One way to adjust this is through the **Relationship Editors > Light Linking > Light Centric**.

10. Batch render your scene when you are finished in order to create a final movie. You can create a series of frames to bring into an editing or compositing package, or render out a QuickTime or .avi movie.

11. Save your scene when you are done.



Figure 6.72: Wagon

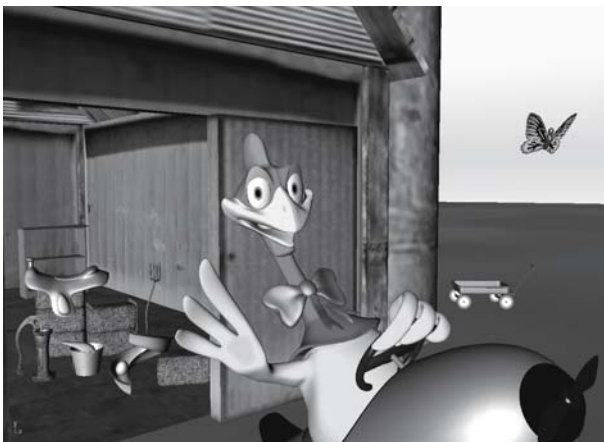


Figure 6.73: Final layout and animation

### 6.9 Project 6 – Bringing It All Together

For your final project, you'll bring together all of the elements you have created for animation, lighting, and rendering. Be sure to pay attention to:

- IK animation,
- 3D painting,
- Indirect lighting,
- Optical effects, and
- Constraints for props.

---

# Index

## A

added attributes, 196, 205  
Adobe Premiere, 256  
aim constraint, 271, 272, 277  
airplane, 5, 13-17, 197-199, 206-207, 252-253  
Alias manuals, 54  
alpha channel, 174  
alpha gain, 231  
ambient, 36-39, 75  
angle, 76  
angle snapping, 263  
animation, fine-tuning, 106-118  
animation timeline, 32-35, 43-45, 50, 58, 68, 109, 186, 258, 299  
anisotropic, 76, 78  
anti-aliasing, 47, 77, 133, 135, 253, 300  
aperture, 136-137  
Apple Final Cut, 256  
area lights, 176-183  
Artisan, 190-192  
attribute editor, 38-39, 41, 48, 81-83, 90, 120, 124, 130, 173-174, 207, 253

audio, 57  
auto-orientation, 215  
automatic mapping, 243  
average vertices, 239  
axis, 3-6, 12, 94, 126, 149, 191, 199, 290

## B

ball drop exercise, 33-35  
banking, 200, 218, 222, 223, 224  
barn, 26, 29-31, 84-85, 120-136, 204, 254-257  
batch render, 50  
Bezier handles, 108  
binding, 210, 212, 283, 290  
bitmap, 45, 134  
blends, 193  
blinn, 76-78, 228  
blur. *See* motion blur  
bone, 190, 209, 212-218, 280, 283  
bottles, 96-97  
bounce, 112, 113, 177, 294  
box modeling, 144-146  
branching, 60  
brush, 190-192, 268

bucket, 97  
bump maps, 171-176, 228  
butterfly, 172

## C

cage, 144  
camera, 10, 82-86, 183, 256.  
*See also specific topics*  
cartesian coordinates, 3-6, 94, 149, 191, 290  
center pivot, 75, 101, 166  
channel box, 14-22, 26-27, 30, 43, 54, 85, 164, 168, 191, 206, 217, 252, 275, 281  
chicken, 154, 187, 211-226, 269-270, 279  
children, 162, 275  
close-ups, 60  
cluster, 264-269, 283  
color, 36-41, 75, 119, 168, 231  
command feedback, 50  
command line, 216, 284  
common material attributes, 38, 39, 120  
component editor, 266  
component mode, 62, 64  
cone, 79, 80, 208, 220, 253

connecting surfaces, 193-195  
connection editor, 207, 208, 280  
connections, input, 121, 124, 152-153, 174-182  
connections, output, 91  
constraints, 271, 276  
construction history, 63, 94-97, 146, 152, 233, 242  
control curves, 282, 284  
control mesh, 144-145, 151  
control vertex, 62-64, 92, 96, 98, 101, 197-198  
convert, 151, 266  
cookies. *See* gobos  
coordinates, 3-6, 94, 149, 191, 290  
copying, 68-69, 101  
cornell box, 177-183  
cosine power values, 76  
coverage, 121, 122  
cranes, 83  
create render node, 119, 120, 174  
cube, 23  
curves, 92-105, 263. *See also specific topics*  
CV. *See* control vertex  
cylindrical mapping  
projection, 245

## D

default file structure, 57  
default light, 55  
deformation, 154, 190, 210, 264-269, 283  
degrees of freedom, 214  
delete history, 247  
dependency graph, 91, 152, 153  
depth map shadow, 48, 80, 179, 252  
depth of field, 136-138  
derived surface modeling, 144  
diffuse, 75  
displacement, 227-231  
distance measurement tool, 136  
dmap resolution, 179, 252  
dock, 66, 77, 173, 191, 234, 252, 267, 270, 283, 291-293  
dollies, 9, 83, 262  
donut, 25, 37-38, 41, 45-51, 77, 98  
driven keys, 196, 202-209  
dropoff, 79, 81  
duplicate, 30, 31, 101

## E

ease-in, 111, 139, 200  
ease-out, 139  
eccentricity, 76  
edges, 59-64, 124, 151, 229, 234  
egg, 184-186, 274-276, 303  
emit specular, 297  
end effector, 280  
environment texture, 182  
expression, 208  
extrude options, 97-99, 145, 147-149, 153

## F

f-stop, 136-138  
face, 62  
facets, 17-18  
fast-in, 111  
fast-out, 111  
fcheck, 134, 301  
file management, 57-58  
fill light, 251  
fillets, 193, 194-195  
filtering, 130  
Final Gathering, 294-296  
fine tuning, 106-118  
fish, 103-105  
flare, 185  
flash, 59

flood, 192, 270, 291  
flowchart, 90, 153  
focal length, 83  
focus, 136-138  
follow, 199  
forward kinematics, 220  
four view, 84  
frame padding, 49, 320  
front, 83  
function curves, 107, 197

## G

geometry, 17, 59-67, 144, 172. *See also specific topics*  
global illumination, 177  
glossiness, 76  
glow, 183, 184  
gobos, 130-135  
graph editor, 107-109, 114, 116, 142, 197, 205, 265  
grayscale, 227, 230  
grid-like icon, 55  
grid snapping, 263  
groups, 162-171

## H

halo, 183  
hardware lighting, 41, 80, 178  
hardware texturing, 233

height ratio, 30, 31  
help line, 8  
hiding and showing, 263  
hierarchies, 90, 114, 162-171, 223, 304  
highlighting, 76  
hotbox, 143  
hotkeys, 143  
hulls, 62  
hypergraph, 152, 153, 162, 166  
hypershade, 36, 120-121, 124, 173-175, 178, 181, 228-229

## I

icons, 2, 223  
IK. *See inverse kinematics*  
Illustrator, 59  
image format, 49  
image scale, 126  
images directory, 58  
importing, 301-302  
in-betweening, 33-35, 106-116, 142, 187  
indirect lighting effects, 294  
input nodes, 54, 91  
insert key, 72, 73, 274  
intensity, 40, 43, 44, 134, 253  
interactive photorealistic renderer (IPR), 185, 255

interface elements, 143  
interpolation, 32-33, 69, 109, 111, 116. *See also in-betweening*  
inverse kinematics (IK), 278-279, 281-282  
IPR. *See interactive photorealistic renderer*  
irradiance, 296  
isoparm, 62

## J

joints, 209-215, 218, 224, 272, 278

## K

keep faces together, 147  
key light, 251  
key select, 85  
keyable attributes, 54, 223  
keyboard shortcut, 56, 90, 142-143  
keyframe, 9, 22-35, 54, 68-69, 106-108, 197, 220, 264, 282-284

## L

Lambert material, 76, 120, 168, 296  
lathe, 93-97  
layers, 110, 111, 165, 262, 284



layout UVs, 243, 245  
left mouse button (LMB),  
11-12, 16, 50, 68, 165, 191-192  
lens flares, 183-185  
light linking, 179, 254, 304  
lighting, 2, 40-41, 78-80, 130,  
176-183, 250-251, 254, 304  
line, 17, 59, 62, 63, 99, 109  
linear tangents, 109, 142  
LMB. *See* left mouse button  
local, 147-148, 172, 214, 216-  
217  
local rotation axis, 216-218  
locators, 211-212  
lofting, 100-105  
low polygon, 145-146, 154,  
289, 293

## M

manipulator tool, 81, 147,  
148  
mapping, 126-127, 174  
marking menu, 96, 105, 123,  
143, 146, 198  
marquee selection, 103, 185  
material attributes, 295  
material types, 38, 75-76,  
295. *See also specific types*  
Maya Embedded Language  
(MEL), 216, 221-223, 285  
Maya Personal Learning  
Edition (PLE), 13-14  
mental ray, 294  
menu bar, 7, 235  
menu sets, 8, 192, 198, 267,  
280, 294  
merge, 237, 239-240, 243-244,  
301  
metallic surfaces, 76  
mirror geometry, 155  
mirror joint, 212  
mirror skin weights, 285,  
287  
mirroring, 288  
middle mouse button  
(MMB), 38-39, 99, 115, 117,  
121, 173, 201, 208, 281  
modeling, 2, 17, 59-67, 92-  
105, 227  
motion blur, 299  
motion paths, 196-201  
mouse buttons, 9, 16. *See*  
*specific buttons*  
move tool, 11, 27, 63-64, 72,  
75, 85, 96, 99, 114, 191, 269,  
274  
multi-part assemblies, 73  
multiple motions, 163

## N

name, 206, 299  
nbump map, 228  
new scene, 274  
node-based architecture, 30,  
91, 152, 154, 264  
non-uniform rational B-  
spline, 59  
normals, 17, 294  
NURBS, 19-21, 54, 59, 63, 90-  
100, 144-145, 177, 196, 221,  
274

## O

object mode, 64, 217, 239,  
267  
omni-directionality, 40  
optical FX, 183, 184-186  
organic models, 59  
orientation, 21, 124, 125, 147,  
196, 199, 211, 214, 271-272  
origin, 19, 73, 84, 97-98, 101,  
165-166, 172  
orthographics, 8, 83  
oscillation, 163  
outliner, 55, 85, 152, 164,  
198, 215, 217, 222, 256  
out tangent, 117  
overlaps, 242-243

**P**

Pac-Man, 90  
paint cluster, 267-268, 270  
paint skin weights, 283-285  
paintbrush, 190, 191  
panels, 8, 37, 253  
pans, 83, 262  
parenting, 162, 166, 209, 271, 282  
partner edge, 244  
patch, 62  
path animation, 196-197, 201  
pencil curve tool, 92  
penumbra angle, 79, 81  
perspective, 37, 41, 46, 55, 80-83, 90, 104-105, 122, 127, 153, 173, 203, 214, 222, 233-235, 252, 283, 286  
phong material, 38, 76, 78, 173-175, 181  
pick, 21, 56, 72, 168-170  
pinning, 240  
pitch, 199, 200  
pivot points, 68-69, 71-75, 93, 101, 199, 200, 277  
pixels, 45-46, 227-230, 299-300  
planar mapping, 126, 127  
plane, 5, 13-17, 197-199, 206-207, 252-253

playback, 33  
playblast, 260  
PLE. *See* Maya Personal Learning Edition  
point-constrained object, 271, 272, 275  
point lights, 40, 42-45, 55, 184  
point snapping, 263  
polygons, 23, 27, 55-63, 71, 125, 145, 152-154, 191-192, 237, 289, 293  
preferences, 33, 109, 129, 222, 235, 258  
prefix hierarchy names, 219-220  
primitives, 17, 55  
procedural texture, 132  
production, 47, 135  
profile curve, 99-103  
project management, 57-58  
projection, 126-128, 234  
proxy, 237, 238  
pull, 192  
pump, 276, 303  
pursuit plane, 197-201  
push, 62, 64, 190-192, 278

**Q**

quick select set, 239, 243-244  
quick view, 84

**R**

radial menu, 142-143  
radiosity, 177  
radius, 4, 31, 192  
ramp, 168  
range slider, 9, 43, 260, 283  
raytracing, 131, 176-183  
referencing, 301  
reflection, 50, 180-183, 187, 292, 298  
refraction, 180-181, 187, 298  
relationship editors, 179, 254, 304  
relax UVs, 240  
renaming, 120, 223  
render, 59  
rendering, 2, 36, 45-51, 59, 77, 80-87, 133-138, 181, 228, 252-256, 262, 298-300  
repeat UV, 122  
resolution, 46, 49, 59, 77  
revolve, 93-97  
RGB values, 119  
rigging, 164, 211, 260, 276, 279-280, 283, 290  
rim light, 251  
right mouse button (RMB), 34-35, 68, 104, 120, 146-147, 235-237  
roll, 200

## 312 Index

---

rolling egg exercise, 86-87, 199  
root, 211, 279-280  
rotation, 3, 4, 10-11, 63-64, 71-74, 81-82, 101, 122-128, 220, 224, 269-272, 280  
roughness, 76

## S

saddle, 151-154, 303  
scale, 3, 4, 6, 10-16, 21, 27, 63-64, 71, 128, 237, 269-271  
scene hierarchy, 90  
scene merge, 301  
script, 221-223  
sculpt, 65, 154, 190-192, 267, 270  
selection, 102-104, 266  
set driven key, 203, 204  
settings, 33, 222-223, 235, 258  
sew UVs, 235  
shading, 2, 36-40, 55, 75, 80, 104, 119-130, 171-176, 227-250  
shape node, 90, 91, 264  
shapes, 223, 275  
shearing, 210  
shelf, 144, 216-219, 222-223, 260, 300  
shift key, 104  
shininess, 76  
show, 13, 17, 262  
side view, 83, 84  
single-node camera, 83, 257  
skeletons, 196, 209-210, 220-221, 279  
skin binding, 190, 210, 283, 289  
slide, 42, 101, 162, 203-204, 302  
slow-in, 111, 116, 117-118, 197  
slow-out, 111, 116, 117  
smoothing, 103, 150, 152, 192-193, 267, 283, 286  
snapping, 74, 99, 214, 263  
sound directory, 57  
sourceimages directory, 57, 58  
spacebar, 104  
span, 92  
specularity, 36-39, 75, 77, 175  
sphere, 17-24, 54, 59, 63, 90, 94, 103, 113, 136, 167, 171, 182, 210, 274, 294  
spline, 110, 112, 116, 142, 197  
split polygon tool, 155  
spotlight, 78-80, 251  
spread X, 76  
spread Y, 76  
status bar, 62  
status line, 22, 30, 74, 77, 80

stepped tangents, 110, 142  
sticky option, 280  
sticky tangents, 112  
stretching, 240  
subdivisions, 60, 151-154, 173, 265  
surfaces, 17, 60, 92-105. *See also specific topics, effects*  
sweep, 54, 90, 93-95, 234, 239

## T

tangents, 108-118, 193  
target, 275  
tessellation, 230  
texture, 36, 119-129, 134, 227, 232-235, 247, 291  
texture placement, 120-125, 174-175, 182  
three-axis system, 3-6, 94, 149, 191, 290. *See specific axes*  
three-button mouse. *See specific buttons, functions*  
three-dimensional paint, 290  
three-node camera, 83  
three-point scheme, 250-255  
tilts, 262  
timeline, 32-35, 43-45, 50, 58, 68, 109, 186, 258, 299  
time range, 198

time slider, 48, 68-69, 169, 198  
time-value-key method, 196  
toggle, 148  
tools, 16, 267, 283, 291. *See also specific icons, features*  
torus, 25, 37-38, 41, 45-46, 49-51, 77, 98  
track, 9  
transform nodes, 54-56, 91, 264  
transform order, 167  
transformation tools, 10  
translation, 6, 10, 16, 82, 101, 124, 271, 280  
transparency, 171-176  
tumble, 9  
twist, 281  
two-dimensional texture, 3, 119-120, 174-177, 290  
two-mode camera, 84-87, 272

## U

underscore, 27, 28  
up axis, 200  
use defaults, 26, 58  
user interface (UI), 7, 57, 119-120, 174, 291

UV mapping, 122-123, 232-240, 244-245, 290

## V

vector graphics, 59  
vertex colors, 190  
vertices, 62, 239. *See specific topics*  
viewports, 55  
visibility, 9, 18, 104, 105, 110-113, 123, 155, 164, 184-186, 217, 263, 282-284  
volume modeling, 144

## W

wagon, 163-166  
water pump, 276  
weighting, 266, 272, 275, 283, 286, 289  
wireframes, 17  
workflow, 142-144  
workspace, 2, 7, 54-58, 90-91, 142-144  
world origin, 3, 19, 73, 84, 101, 165-166, 172  
World Up, 200

## X

x-axis, 3-6, 94, 126  
x-ray mode, 105

## Y

y-axis, 3-6, 191  
yaw, 199, 200

## Z

z-axis, 148, 149, 150  
zeroing, 15, 223, 227  
zooms, 262