HOTELLA
Vacancy
POOL

1 Texture Effects

All too often textures are overlooked as a solution for creating effects. The tendency is to think of textures as simply a means for coloring objects. The tutorials in this first chapter demonstrate some ways in which engaging visual effects can be created quickly and easily simply by taking advantage of the power that textures have to offer.

Chapter Contents

Create Animated Effects with a Ramp Texture

This first challenge involves adding an effect to the opening shot of an animated short. Figure 1.1 shows part of the storyboard for the opening. The camera moves down and out to reveal a dilapidated hotel sign with some broken lights. The gag is that the lighted parts of the sign spell "Hell." The neon vacancy sign is just barely hanging on. The director would like to suggest that the hotel is possessed by demons, so as the camera stops, the Vacancy sign lights up, one of the bolts gives away, and the sign swivels so that the arrow points downward. Subtle, no?

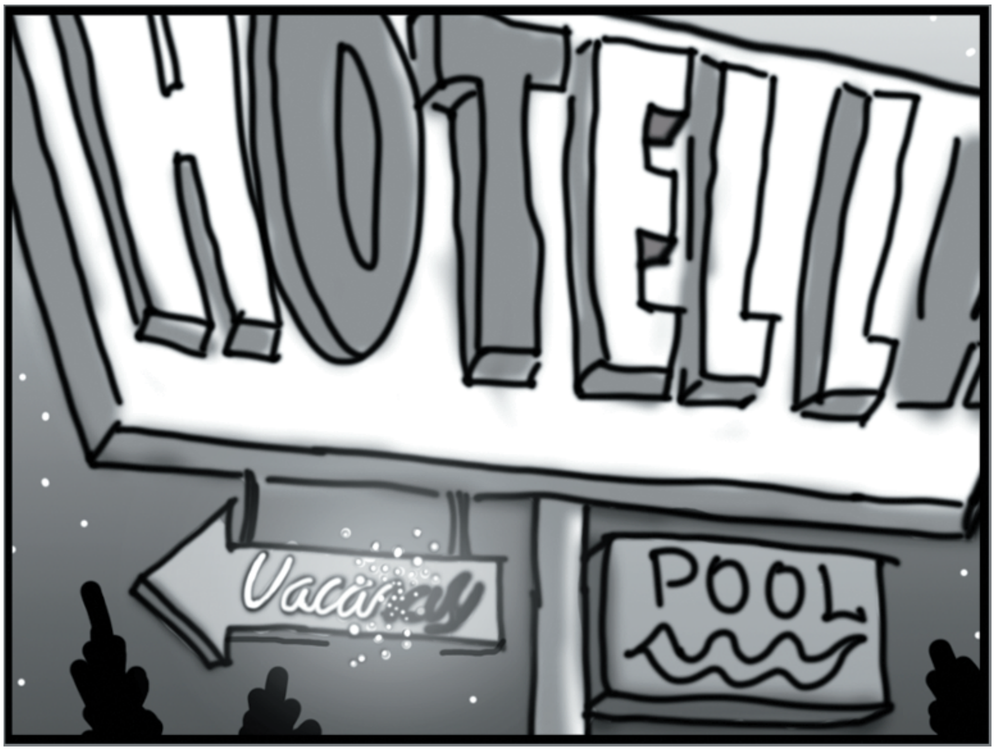


Figure 1.1 The storyboard shows a neon sign magically coming to life.

Rather than have the sign blink on like a real neon sign would, the director would like the light of the neon Vacancy sign to start at the letter *V* and travel to the end over the course of about a second. A particle effect should be added to the leading end of the light.

The sign has already been modeled, and the basic camera move, along with the animation of the broken sign, is established. For the Vacancy sign, there is a NURBS curve but no neon geometry as of yet.

This effect should be fairly easy to create by extruding geometry along the length of the curve. A shader can be applied to the geometry, and then an animated ramp attached to the incandescence of the shader will provide the lighted effect. Whenever you're confronted with an effect, it's a good idea to design a rig with a few simple controls that can be easily animated. Whenever possible, the rig should be set up to anticipate changes as easily as possible because art directors are fussy and tend to change their minds a lot. This tutorial will take you through the steps of setting up a rig so that one control can be used to animate the light traveling along the length of the word *Vacancy* as well as the position emitter for the particle effect.

This exercise demonstrates one way of creating the rig and introduces you to several methods for navigating the node hierarchy in Autodesk® Maya®. Every Maya artist has their own style of working, but once in a while it's a good idea to explore alternative methods of navigating the Maya interface. Doing so will open possible workflows that you may not be aware of, which can increase your efficiency as well as your understanding of how Maya works.

Create the Sign Geometry

The geometry for the Vacancy sign can be extruded along the existing curve in the scene. Before you start extruding geometry using NURBS geometry, I suggest you consider applying a Paint Effects stroke to the curve and then convert the stroke into polygons. The reason for this is that it's easier to control the twisting and pinching that may occur as the geometry follows the curve. In addition, the UV texture coordinates created by the Paint Effects geometry will be easily adapted so that the ramp that creates the neon light can be animated without too much work.

1. The project files for this chapter can be downloaded from from the book's support site. Use your web browser to navigate to `www.sybex.com/go/mayavisualeffects2e` and download the `Chapter01_project`. Once the files have been downloaded, unzip them to your local drive. Open Maya and use the File menu to set the Project to `Chapter01_project`. Then open the signSTart.ma file located in the Scenes directory of the `Chapter01_project` file directory. Once the scene is open, switch to the Persp camera using the View menu in the main viewport.
2. In the Outliner, expand the vacancy_sign group, and select the curve node named vacancyNeonCurve (see Figure 1.2).

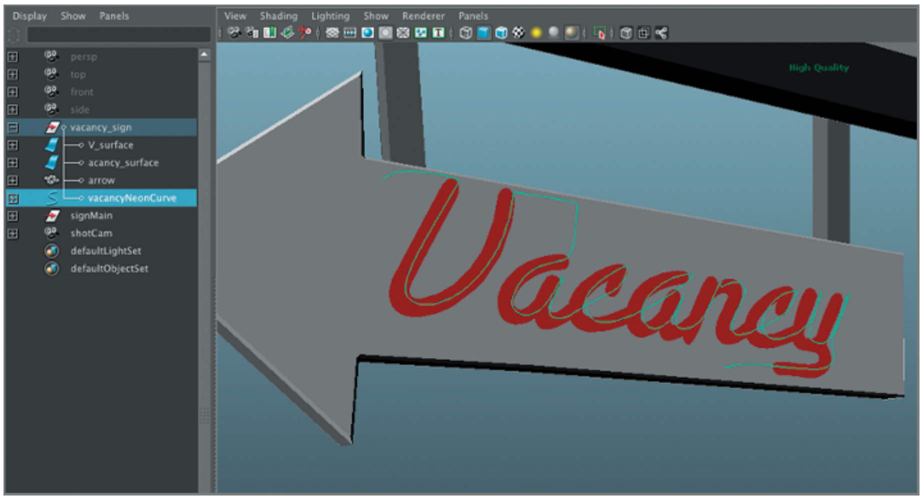


Figure 1.2 Select the vacancyNeonCurve node in the Outliner.

3. Switch to the Rendering menu set, and choose Paint Effects › Curve Utilities › Attach Brush To Curves (see Figure 1.3). The default stroke, which is a thick black line, is applied to the curve.

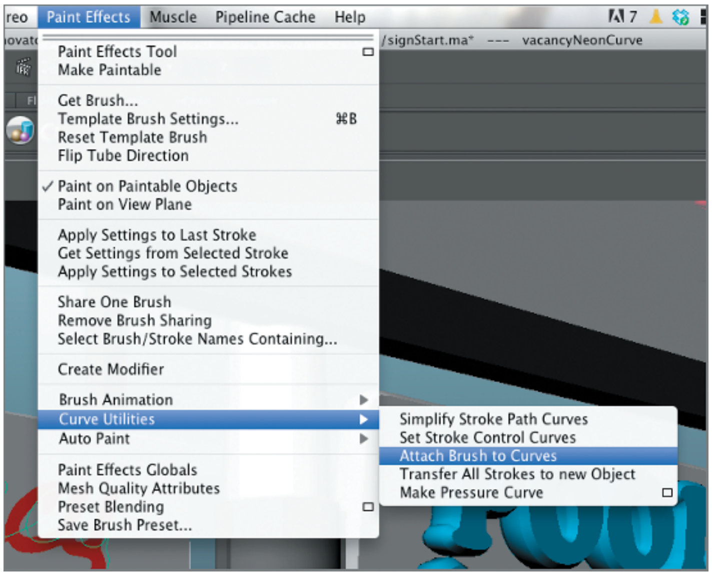


Figure 1.3 Use the Paint Effects menu to attach the default brush stroke to the curve.

4. Select the Stroke1 node in the Outliner, and open the Attribute Editor.
5. To convert the stroke into geometry, choose Modify › Convert › Paint Effects To Polygons › Options; in the options, turn on Quad Output and Hide Strokes.
6. Maya creates a new node for the converted geometry and places it in a group called brush2MeshGroup. Expand this group, and select the brush2Main node. Press Shift+P to unparent the mesh, which moves it out of the group brush2MeshGroup.
7. Double-click the brush2Main node in the Outliner so it becomes highlighted, and rename the brush2Main node to vacancyNeonGeo. You can select and delete the brush2MeshGroup node (see Figure 1.4).

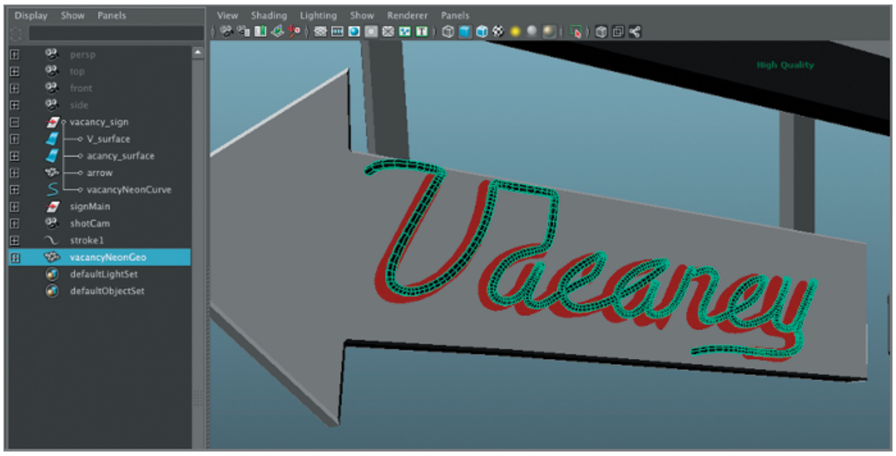


Figure 1.4 Convert the Paint Effects stroke into polygons. Ungroup the node, and name it vacancyNeonGeo.

Make sure you do not delete the Stroke1 node and don't delete the history on the vacancyNeonGeo node. Since the stroke is applied to the curve that is contained inside the vacancy_sign group, which is keyframed, the vacancyNeonGeo node will inherit the animation (see the sidebar "Construction History" for information on how this works). The advantage of keeping the history on the vacancyNeonGeo node is that you can change the width and other attributes of the vacancy geometry by tweaking the settings on the stroke1 node. This is helpful if a picky art director decides changes need to be made to the sign at some point in the future.

Construction History

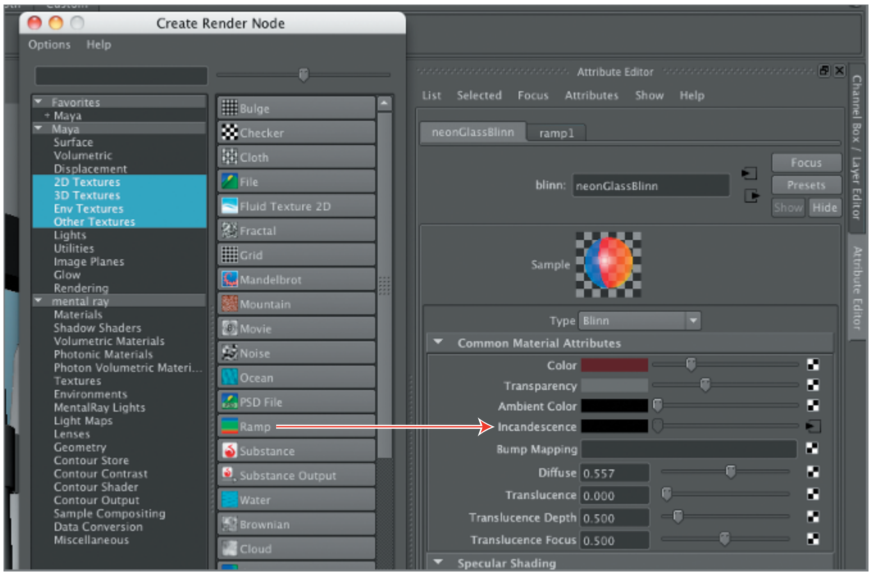
Maya keeps a record of all the changes that go into the nodes that you create when you model an object as well as how different nodes may be connected. This is known as *construction history*. When you create a model such as the neon sign, Maya remembers that the surface of the neon tube was created when you attached a Paint Effects stroke to the tube and that you then converted the tube into polygons. So, if you make changes to the original curve or the Paint Effects stroke, the changes propagate all the way to the polygon surface of the tube; thus, the polygon tube "inherits" any changes you make to the original curve or any of the other nodes that went into the construction process. You can take advantage of construction history when you animate the nodes that are connected through construction history. You can toggle construction history on or off by clicking the history script icon located on the status bar.

Set Up the Neon Shader

Here's where the fun begins. To create the effect of the light traveling along the length of the neon tube, you'll attach a ramp to the incandescence channel of a Blinn shader.

1. Open the Hypershade to find the shader named neonGlassBlinn. This is a simple shader that has already been set up to create the look of the neon glass. Select the shader, and MMB drag it on top of the vacancyNeonGeo in the Perspective view.
2. In the Hypershade, select the neonGlassBlinn shader, and open its settings in the Attribute Editor. Click the checkered box to the right of the Incandescence channel to open the Create Texture Node panel.
3. Select the 2D Textures category on the left side of the panel, and click Ramp to create a ramp node (see Figure 1.5). Hold the mouse pointer over the Perspective view, and press the 6 hotkey to switch to textured view. Make sure that the Renderer panel in the viewport window is set to High Quality Rendering; otherwise, you won't see the ramp texture update properly as it's applied to the geometry.

   In Perspective view, the rainbow pattern of the ramp appears on the neon tube, but it is a repeating pattern that looks more festive than demonic. To fix this, you can adjust the UVs of the vacancyNeonGeo node.



blinn: neonGlassBlinn
Type Blinn
Diffuse 0.557
Translucence 0.000
Translucence Depth 0.500
Translucence Focus 0.500

Figure 1.5 Use the Create Render node panel to attach a ramp texture to the Incandescence channel of the neon-GlassBlinn material.

4. Select the vacancyNeonGeo node, and choose Window › UV Texture Editor. If you zoom out (waaaaay out), you'll see that the UVs are a long vertical strip. In the UV Texture Editor's menu panel, expand the Textures menu, and select vacancyNeonGeo|blinn2SG|ramp1 so that the ramp appears in the UV Texture Editor.

As you can see, the rainbow colors of the ramp appear in the upper quadrant of the UV Texture Editor. What you don't see is that outside of this area, the ramp texture repeats over and over to infinity, which is what the colors on the neon tube continually repeat; the UVs for the tube continue well outside of that upper quadrant (see Figure 1.6).

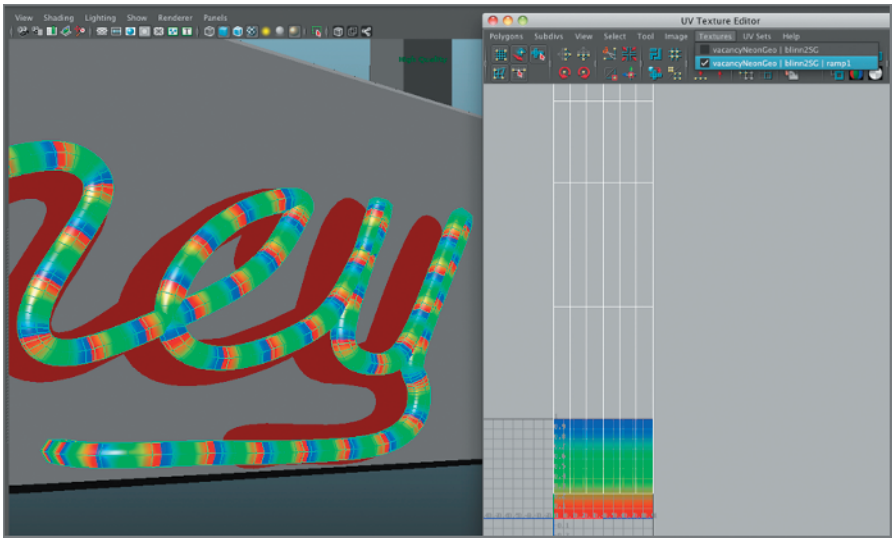
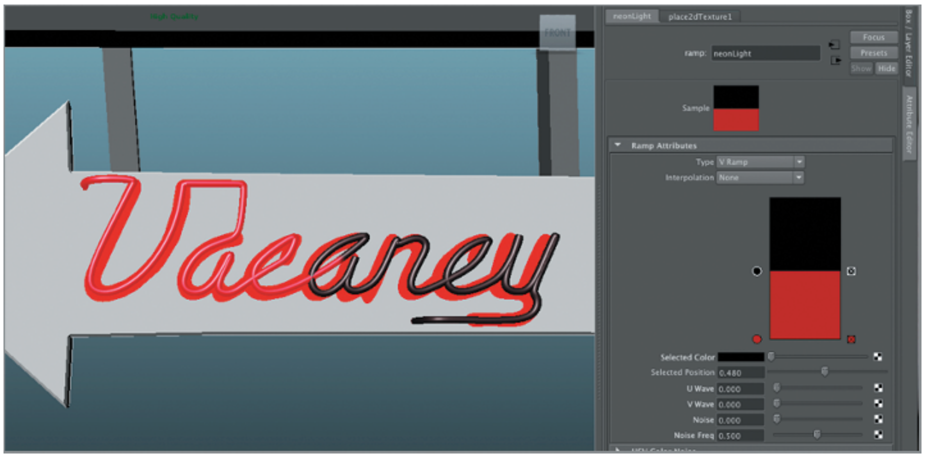

View Shading Lighting Show Renderer Panels
UV Texture Editor
Polygons Subdivs View Select Tool Image Textures UV Sets Help
vacancyNeonGeo | blinn2SG
vacancyNeonGeo | blinn2SG | ramp1

Figure 1.6 The UV Texture Editor shows that the UV texture coordinates of the neon sign geometry go well beyond the upper quadrant of the texture space. The result is that the ramp texture repeats along the length of the glass tube.

5. Right-click the UVs, and select UV from the marking menu. Select some of the UV coordinates in the editor (the selected coordinates will be highlighted in green). Hold the Ctrl key, right-click again above the selected UVs, and choose To Shell from the marking menu. This selects all of the connected UVs of the glass tube geometry.
6. From within the UV Texture Editor, choose Polygons › Normalize. This forces the UVs to fit within the upper-right quadrant of the UV texture space. Now the ramp is no longer repeating. The UV coordinates will look fairly dense in the UV Texture Editor, but in Perspective view, you'll see that the ramp goes from red to green to blue across the length of the sign.
7. Select the ramp1 node on the Textures tab of Hypershade, and open its settings in the Attribute Editor.
8. Click the *x* at the top of the ramp to delete the blue color at the top of the ramp. Set the middle color to black; then set Interpolation to None.
9. Set the bottom color to a devilish shade of red.
10. Try dragging the circle for the black color up and down; you'll see the light move along the neon tube (see Figure 1.7). Name the ramp node neonLight.



ramp: neonLight
Type: V Ramp
Interpolation: None
Selected Color
Selected Position 0.480
U Wave 0.000
V Wave 0.000
Noise 0.000
Noise Freq 0.500

Figure 1.7 Move the black color of the ramp up and down; the color updates on the tube geometry in Perspective view.

To make the sign easier to read, you can apply a separate dark shader to the parts of the tube that connect the letters. This replicates the way actual neon signs are constructed in the real world, adding an extra touch of realism.

11. Right-click the vacancyNeonGeo geometry in Perspective view, and choose Face. Carefully select the individual faces connecting the *V* and the *A*. Once they are selected, switch to the Rendering menu set, and choose Lighting/Shading › Assign Favorite Material › Lambert. This creates a new Lambert shader and applies it to the selected faces. Do the same for the other parts of the sign. Assign the same Lambert shader to these selected faces (you can use Lighting/Shading › Assign Existing Material to do this easily). Figure 1.8 shows the result.

12. Edit the Lambert shader that has been applied to these faces so that the Diffuse color is a very dark gray.

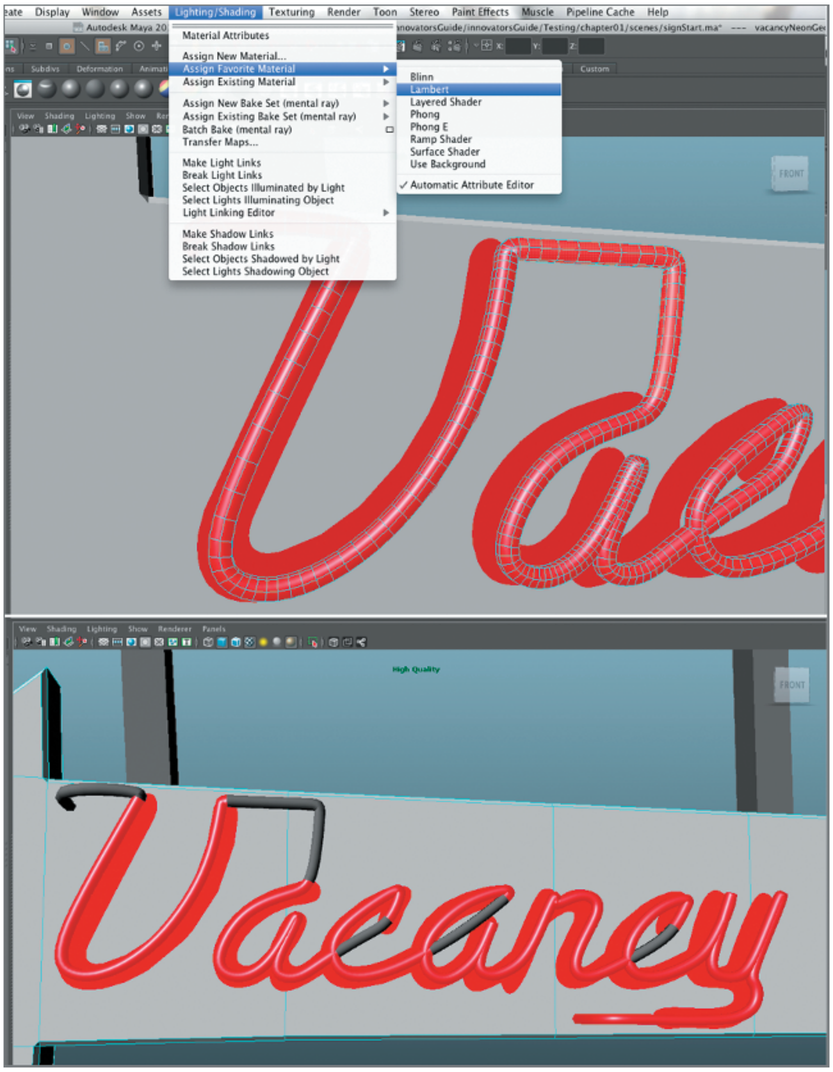


Figure 1.8 To make the sign easier to read, select the faces between some of the letters, and apply a dark gray Lambert shader.

Create an Animation Control for the Sign

At this point, you could consider yourself done, but before you declare "mission accomplished," you should take some time to set up an animation control. Taking the time to do this now can save you trouble later when the director starts asking for tweaks on the animation. In addition, any animators you share the project with will appreciate not having to hunt around to find the keyframed attributes within the many panels Maya has to offer.

You can create custom controls in lots of ways. In this example, you'll create an arrow-shaped control using a NURBS curve. When the control is connected to the ramp, dragging it back and forth will control the progress of the light along the sign. This way, the animation can be controlled directly in Perspective view, eliminating the need to hunt around for the animated settings in the Attribute Editor.

1. Switch to a front view, and zoom in on the sign. Turn on the grid using the View menu in the upper left of the viewport.
2. Choose Create › EP Curve Tool › Options. In the Options, set Cure Degree to Linear. This option means that the curves will be straight lines.
3. Activate Snap To Grids by clicking the icon that looks like a magnet over a grid on the top menu bar of Maya. Create a straight line by clicking two different points below the sign, as shown in Figure 1.9.

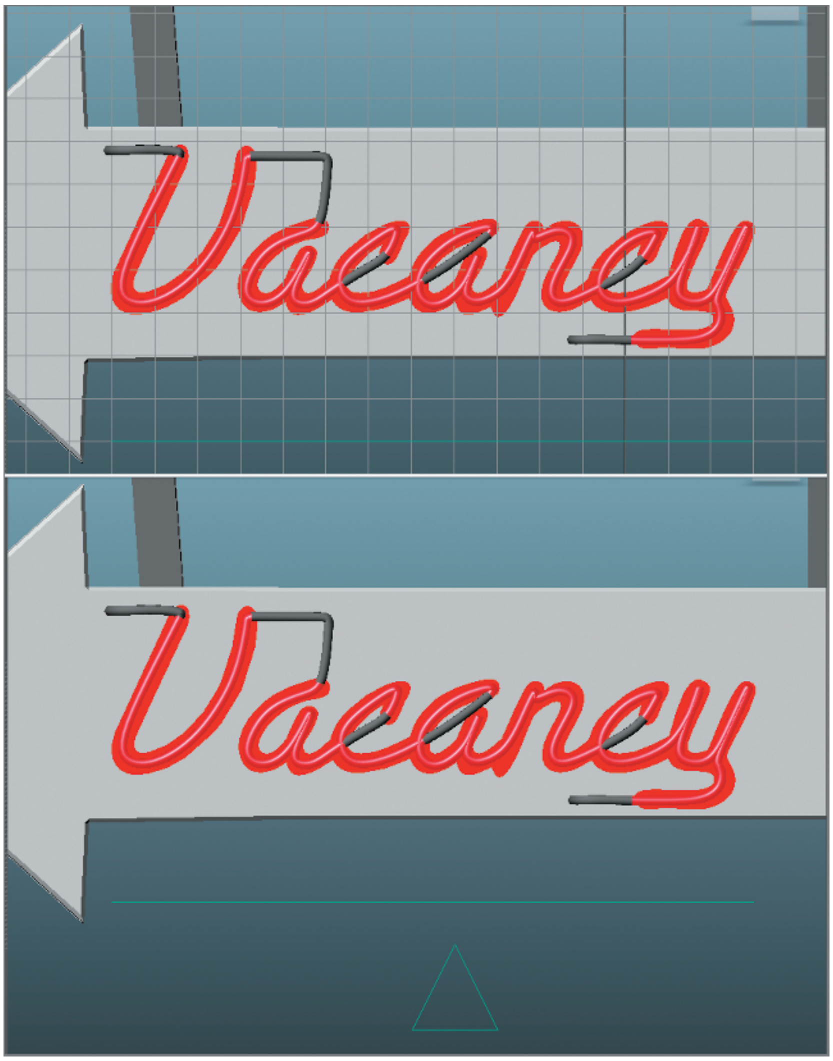
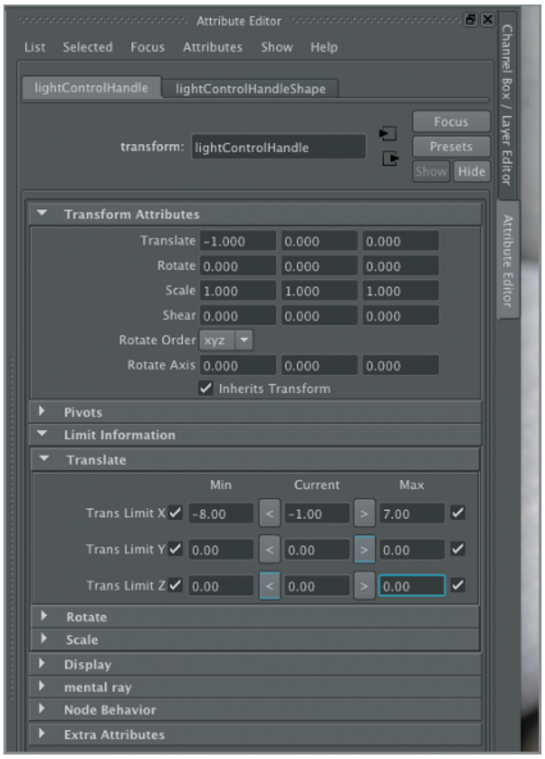


Figure 1.9 Create a line below the Vacancy sign using the EP Curve tool (top image). Create a triangle below the line (bottom image).

4. Double-click the curve in the Outliner, and name it lightControl. This line will provide a visual indication for the length of the control.

5. Using the EP Curve tool again, click three times to create a triangle just below the line. Name the triangle lightControlHandle. This triangle will be the actual control that is used to animate the light of the sign (see the bottom image in Figure 1.9).
6. Select lightControlHandle; then Shift+select the light control. Press the P hotkey to parent the lightControlHandle to the lightControl. Turn off the grid so you can easily see the control curves.
7. Select lightControlHandle, and choose Modify › Center Pivot. Open the Attribute Editor for the lightControlHandle. Make sure the tab in the Attribute Editor is set to lightControlHandle and not lightControlHandleShape. You're going to set translation limits on the handle, so you want to make sure you're working on the transform node of the curve and not the shape node.
8. Expand the Limit Information rollout; then expand the Translate rollout within this section. Move the lightControlHandle to the left end of the line, and click the arrow to the left of the Current setting, as shown in Figure 1.10. Click the check box next to Trans Limit X to set the minimum value.



transform: lightControlHandle

Translate	-1.000	0.000	0.000
Rotate	0.000	0.000	0.000
Scale	1.000	1.000	1.000
Shear	0.000	0.000	0.000
Rotate Order	xyz		
Rotate Axis	0.000	0.000	0.000

✓ Inherits Transform

	Min	Current	Max	
Trans Limit X ✓	-8.00	-1.00	7.00	✓
Trans Limit Y ✓	0.00	0.00	0.00	✓
Trans Limit Z ✓	0.00	0.00	0.00	✓

Figure 1.10 Set the minimum and maximum Translate values for the handle so that it does not go beyond the length of the lightControl curve.

9. Move the lightControlHandle all the way to the right, and use the same techniques to set the Max setting. Once this is set, the lightControlHandle is restricted on the X axis so that it can't move beyond the length of the line.
10. Set the Min and Max values for Trans Limit Y and Trans Limit Z to 0 so that the lightControlHandle can't be moved along these axes by mistake (see Figure 1.10).

11. To connect the arrow to the ramp, you can use a set-driven key. This allows you to use the attribute of one node to control the attribute of another. Switch to the Animation menu set, and choose Animate › Set Driven Key › Set. This will open the Set Driven Key control panel.

12. Select lightControlHandle, and click the Load Driver button in the Set Driven Key window. Select Translate X on the upper-right panel. This means that the Translate X attribute of lightControlHandle will drive whatever attribute is selected in the lower-right panel. Of course, loading the appropriate attribute for the ramp is a little tricky.

    The Translate X attribute of the lightControlHandle needs to be connected via a driven key to the position attribute of the neonLight ramp texture's black color marker. This attribute can't be selected directly, so you have to dig into the Outliner a little to load the correct setting into the Set Driven Key panel.

13. In the Outliner, expand the Display menu at the top, and turn off DAG Objects Only. The Outliner now shows all the nodes in the scene. Type **neonLight** in the search field at the top so that the Outliner shows only this node, which is the ramp texture.

14. In the Display menu of the Outliner, make sure Attributes (Channels) is selected so that attributes are shown in the Outliner. Select the Color Entry List attribute, as shown in Figure 1.11. Click the Load Driven button in the Set Driven Key window.

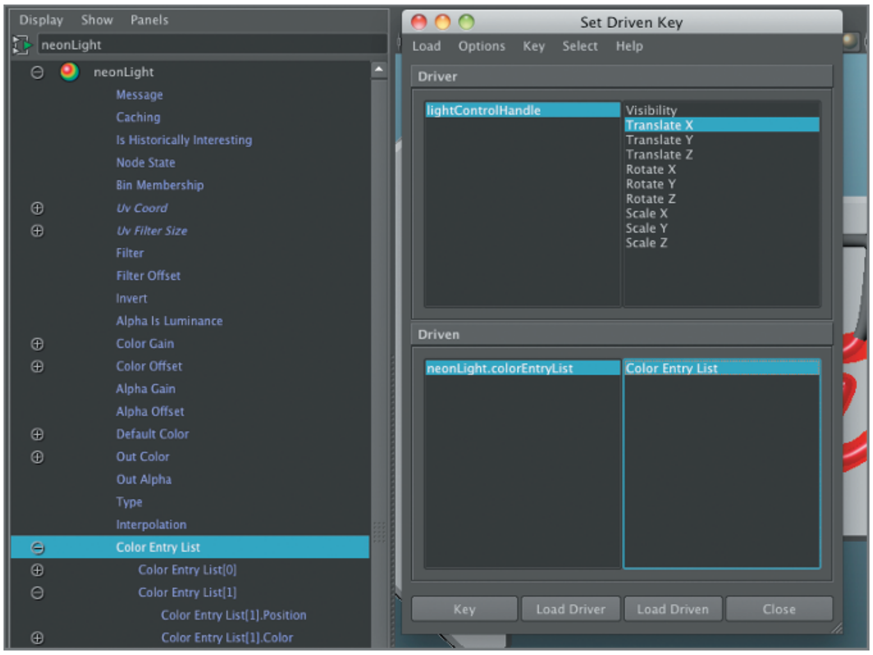


Figure 1.11 Use the Outliner to select the Color Entry List attribute of the neonLightChannel.

15. Move the lightControlHandle all the way to the left. Select the neonLight.colorEntryList heading in the lower left of the Set Driven Key window; this loads the neonLight ramp in the Attribute Editor. Select the circle for the black color of the

ramp, and move it almost all the way to the bottom (if you move it too far down, the ramp will turn red since this color will overlap the red marker of the ramp).

16. Click the Key button in the Set Driven Key window; this sets a key so that when the lightControlHandle is all the way at the left, the black color will be almost all the way at the bottom.

17. Move the lightControlHandle all the way to the right. Select the neonLight.colorEntryList heading in the lower left of the Set Driven Key window; this loads the neonLight ramp in the Attribute Editor again. Select the circle for the black color of the ramp and move it all the way to the top. Press the Key button again. This sets a second keyframe (see Figure 1.12).

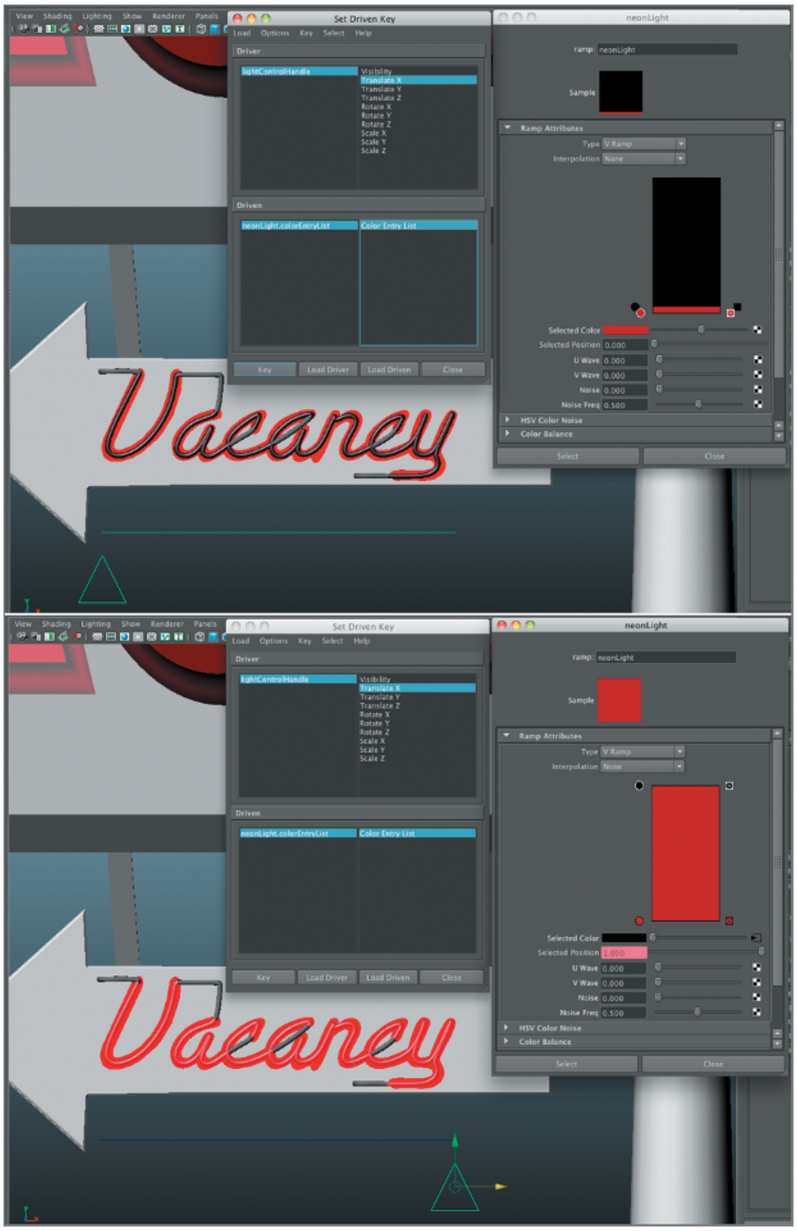


Figure 1.12 The Translate X attribute of the lightControlHandle is set up to drive the animation of the ramp texture.

18. Move the lightControlHandle back and forth, and you'll see the red color moves along the length of the neon light tube. It takes a fair amount of setup, but now that the control is working, you no longer have to hunt around the various nodes to set keyframes. You can simply keyframe the Translate X attribute of lightControlHandle, and the light effect will update.

Add a Particle Emitter

The final piece of the puzzle for this effect is a particle effect that follows the leading edge of the animated light. This can be achieved by attaching an emitter to the curve and using the Translate X lightControlHandle to the drive position of the emitter along the curve.

1. Continue with the scene from the previous section. Switch to the nDynamics menu set. You can reset the Outliner using the View menu to turn on DAG Objects Only and turn off the attributes. Remember to clear neonLight from the field at the top of the Outliner.
2. Choose nParticles › Create nParticles › Points. This sets the style of the nParticles to a preset, which can be adjusted to create the sparkling effect (the left image in Figure 1.13).

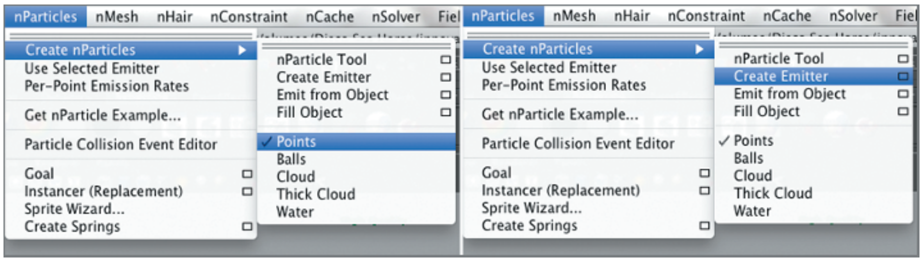
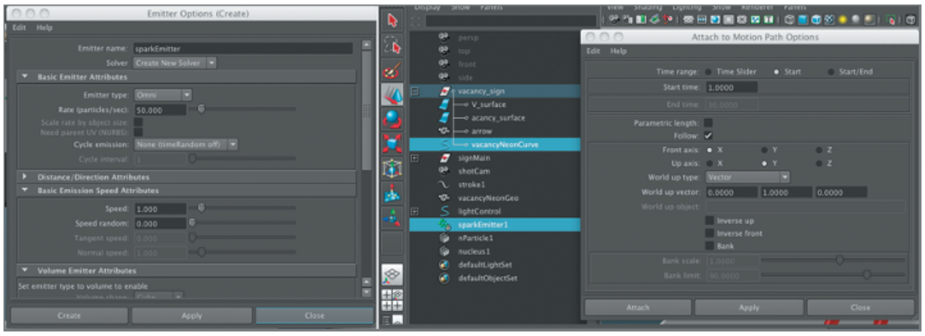


Figure 1.13 Set the style of the nParticles to Points and then create an emitter.

3. Choose nParticles › Create nParticles › Emitter › Options to open the Emitter Options panel (the right image in Figure 1.13). Set the Emitter type to Omni, Rate (particles/Sec) to 50, and Speed to 1. Set the name of the emitter to sparkEmitter. Click Create to make the emitter (the left image in Figure 1.14).
4. In the Outliner, select the vacancyNeonCurve node and the emitter1 node. Switch to the Animation menu set, and choose Animate › Motion Paths › Attach To Motion Path › Options. In the options, set Time Range to Start, and leave Start Time at 1. Leave the other options at their default values. Click Attach to connect the emitter to the Vacancy sign curve (see right image in Figure 1.14).

   You can use the lightControlHandle to drive the position of the emitter along the curve as well as the rate of nParticle emission. You'll need to connect the Translate X attribute of lightControlHandle to the U Value attribute of the motionPath node that connects the emitter to the curve of the Vacancy sign. This attribute already has a keyframe applied to it by default, so first you'll need to select the node and remove the keyframe.


Emitter Options (Create)

Emitter name: sparkEmitter
Solver: Create New Solver

Basic Emitter Attributes
Emitter type: Omni
Rate (particles/sec): 50.000
Scale rate by object size:
Need parent UV (NURBS):
Cycle emission: None (timeRandom off)
Cycle interval: 1

Distance/Direction Attributes

Basic Emission Speed Attributes
Speed: 1.000
Speed random: 0.000
Tangent speed: 0.000
Normal speed: 1.000

Volume Emitter Attributes
Set emitter type to volume to enable

Attach to Motion Path Options

Time range: Time Slider [✓ Start] Start/End
Start time: 1.0000
End time: 30.0000
Parametric length:
Follow: ✓
Front axis: [✓ X] Y Z
Up axis: X [✓ Y] Z
World up type: Vector
World up vector: 0.0000 1.0000 0.0000
World up object:
Inverse up
Inverse front
Bank
Bank scale: 1.0000
Bank limit: 90.0000

Figure 1.14 Create the emitter (left image) and use Attach To Motion Path to attach the emitter to the vacancyNeon-Curve (right image).

5. If you know the name of the node you need to select, you can reduce the amount of hunting and clicking in the interface by taking advantage of the Select By Name field at the top of the interface. Click the down arrow to the right of the coordinate input fields on the top menu of the interface, and from the pop-up choose Select By Name. In the field, type **motionPath1** (see Figure 1.15). This is the node that is responsible for connecting the sparkEmitter to the vacancyNeonCurve.

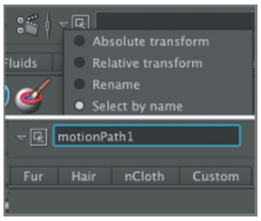


Figure 1.15 Use the Search By Name field at the top of the menu bar to select nodes easily.

6. In the Channel Box, you'll see the attributes for motionPath1. Right-click the highlighted box next to U Value, and choose Break Connections (Figure 1.16). This removes the keyframe for this attribute; make sure that U Value is set to 0. This places the emitter at the start of the curve.

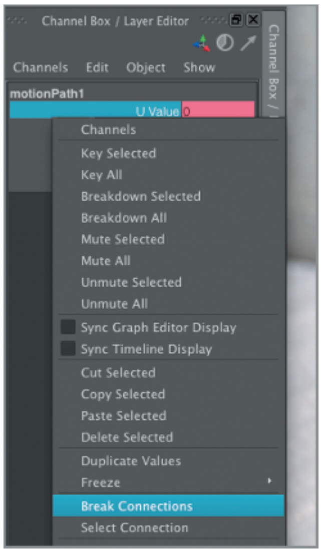


Figure 1.16 Break the connection for the U Value channel of motionPath1 to remove the keyframes.

7. Switch to the Animation menu set, and select the lightControlHandle. Choose Animate › Set Driven Key › Set. This opens the Set Driven Key panel again. With the lightControlHandle selected, click Load Driver.
8. Select motionPath1, and click Load Driven in the Set Driven Key box (see step 5 in this exercise).
9. Move lightControlHandle all the way to the left. In the top right of the Set Driven Key panel, select Translate X; in the bottom right, select U Value. Click the Key button to create the keyframe (see Figure 1.17).

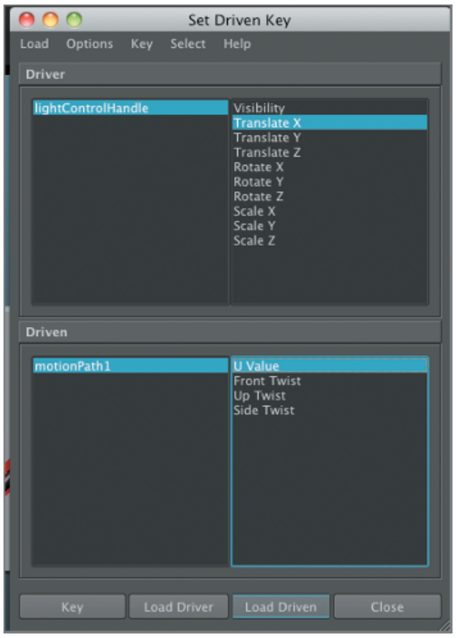


Figure 1.17 Load the lightControlHandle node as the driver and motionPath1 as the driven node in the Set Driven Key panel.

10. Select the lightControlHandle, and move it all the way to the right. Double-click the motionPath1 in the bottom left of the Set Driven Key panel; its attributes appear in the Channel Box. Set U Value to 1. This places the emitter at the end of the curve. In the Set Driven Key panel, click the Key button to set the second key.
11. Hold the mouse pointer over Perspective view, and press 4 to switch to wireframe view so you can see the emitter (represented as a small circle). Move the lightControlHandle back and forth; you should see the emitter move along the curve (see Figure 1.18).

    At the moment, the position of the emitter will not be perfectly in sync with the colors of the ramp, but you'll fix that in a moment using the Graph Editor. Before you get to that, you can also connect the rate of the emitter to the lightControlHandle so the final rig uses a single control to animate the color of the ramp; the position of the emitter, the rate of the particle emission, and all of these attributes can be fine-tuned using the Graph Editor.

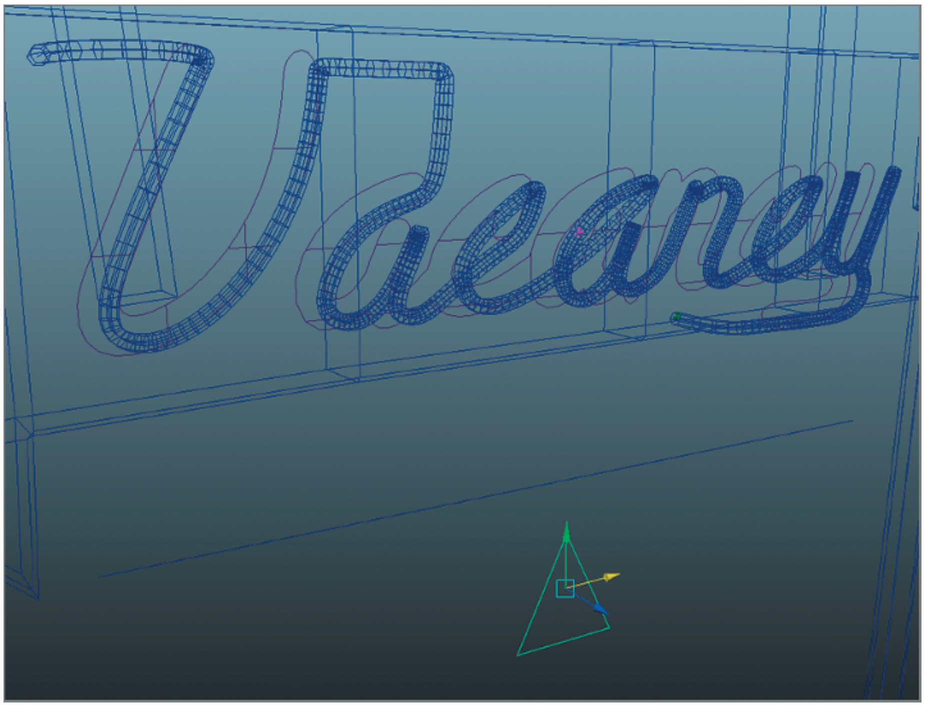


Figure 1.18 As you move the lightControlHandle along the X axis, the emitter (shown as a pink circle) moves along the curve.

12. Select sparkEmitter in the Outliner. In the Set Driven Key panel, click Load Driven. lightControlHandle should still be loaded as the driver.
13. Use the Set Driven Key panel to keyframe the emitter so that when lightControl-Handle is all the way to the left, the Rate attribute of the emitter is 0, and when lightControlHandle is all the way to the right, the emitter's Rate attribute is 100. You can use the same technique that you used in steps 9 and 10 to do this.
14. This is probably a good point at which to save your scene file.

Animate the Effect Using the Custom Rig

The rig is essentially complete, but for the animation to behave the way the director wants, you'll need to fine-tune the animation curves created by the set-driven key. This way, the position of the spark is in line with the color of the ramp, and the rate of the spark emission is a little more elegant.

Keep in mind that the director will want to tweak the look of the effect quite a bit; the rig is designed to reduce the amount of work it takes to change the animation in the future. The rig takes a few more steps to set up, but the payoff comes later when you have to constantly change the animation to match the director's vision.

1. The shot requires that the light on the sign start on frame 320 and end around frame 360. Set the Time Slider to frame 320. Move the lightControlHandle all the way to the left. Hold the mouse over the Translate X channel in the Channel Box; right-click and choose Key Selected.

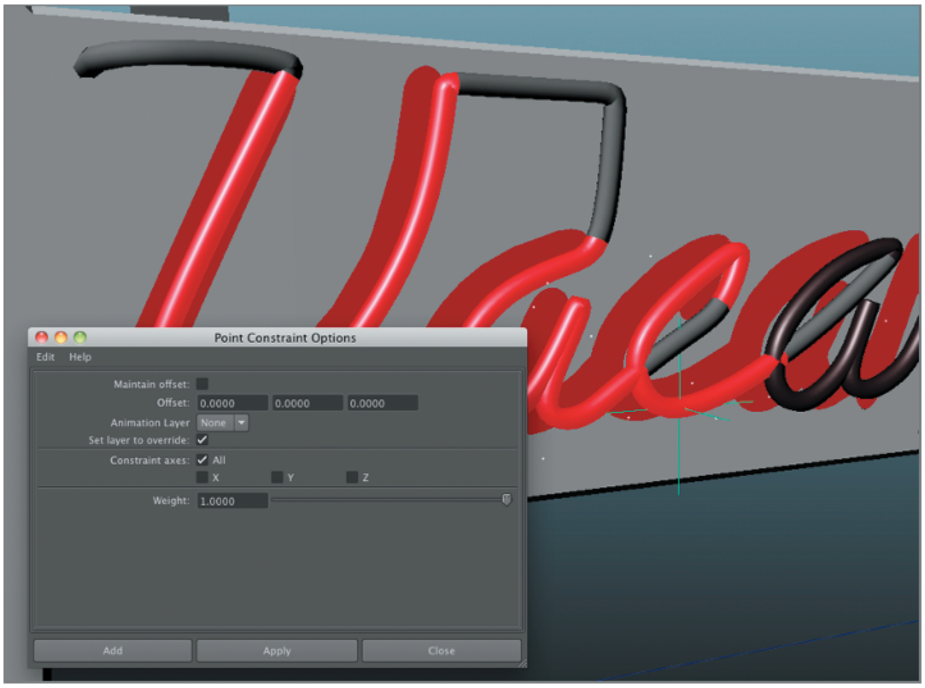
2. Move the Time Slider to 360, move the lightControlHandle all the way to the right, and set a second keyframe.
3. Switch to the ShotCam camera in the viewport. Rewind and play the scene.

Playback Preferences

The playback speed in the Time slider's preferences should be set to Play Every Frame, and Max Playback Speed should be set to Real-Time (24fps); otherwise, the nParticle dynamics will not play correctly. You can set these options in the Preferences window.

It's a bit lackluster at the moment, so it's time to do some tweaking.

4. Switch to Perspective view in the viewport, and zoom in on the sign. Set the Time Slider range so that it starts at 300 and ends at 380; this way, you don't have to sit through the whole animation every time you review it.
5. It may be a bit hard to see the emitter's position and the progression of the red color as it moves along the length of the glass tube. You can make things a bit easier by simply constraining a locator to the emitter. Choose Create › Locator to add a locator to the scene. In the Outliner, select sparkEmitter, and Ctrl+select locator1. Switch to the Animation menu set, and choose Constrain › Point › Options. In the options, make sure Maintain Offset is disabled. Click Add to make the constraint (see Figure 1.19).
6. Scrub along the Timeline. You should see the locator move along the curve. To fix the timing, you can work with the Graph Editor.



Point Constraint Options

Maintain offset: ☐
Offset: 0.0000 0.0000 0.0000
Animation Layer: None
Set layer to override: ✓
Constraint axes: ✓ All
☐ X ☐ Y ☐ Z
Weight: 1.0000

Figure 1.19 Constrain a locator to the sparkEmitter so that it's easier to see where the emitter is on the vacancySignCurve.

Edit Connections Using the Node Editor

The process of tweaking the various animation curves that are part of the rig will require selecting a lot of nodes that are connected already via a set-driven key but also spread out among the various panels of Maya. Maya offers a number of ways to view connected nodes. One of the newest additions to the tool set is the Node Editor, which, in some ways, is like the older Hypergraph but slightly more elegant. As the next few steps demonstrate, you can use this editor to easily select the nodes you have already connected and even get rid of some nodes that have been created along the way that aren't really needed.

1. In the viewport window, choose Panels › Layout › Three Panes Split Top. Use the Panel menu in each pane to set the top-left viewport to Node Editor, the top right view to Persp, and the bottom view to Graph Editor.
2. In Persp view, select the lightControlHandle so that it turns green. In the Node Editor, click the icon for input and output connections. The connected nodes appear as boxes connected by curved lines (see Figure 1.20).

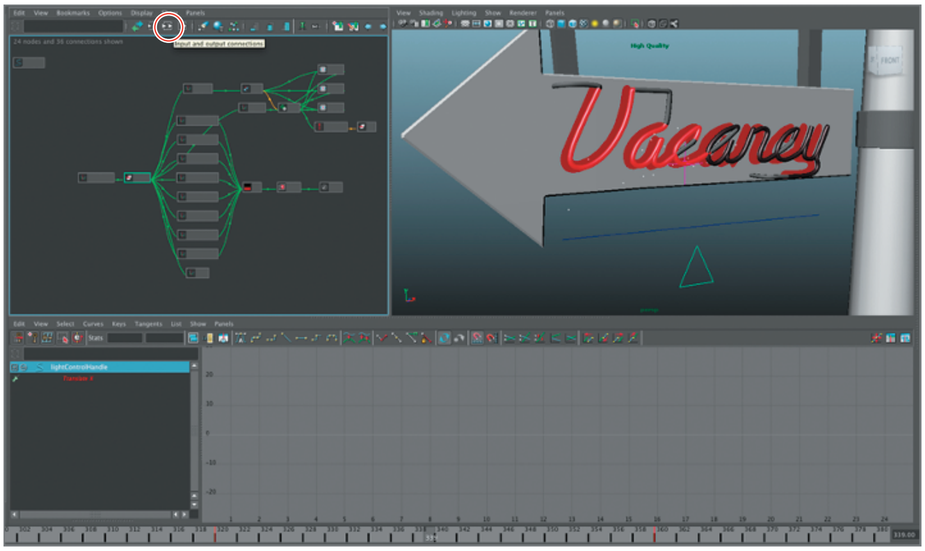

Input and output connections
24 nodes and 36 connections shown
High Quality
FRONT
lightControlHandle
Translate X

Figure 1.20 Arrange the panels so that you can see the Node Editor, the viewport, and the Graph Editor. Graph the input and output connections for lightControlHandle in the Node Editor.

3. Hold the Alt key while RMB dragging in the Node Editor to zoom in on the node boxes; take a look at the animation curve nodes. By selecting the animation curve nodes, you will see the animation curves appear in the Graph Editor (see Figure 1.21).

   There are quite a few animation curve nodes. Several of them were created when you used a set-driven key to drive the ramp's color position; as a result, a few animation curves were added that don't really improve the scene.
4. In the Node Editor, select the node named neonLight_colorEntryList_1_colorB, and press the Delete key to remove it from the scene.

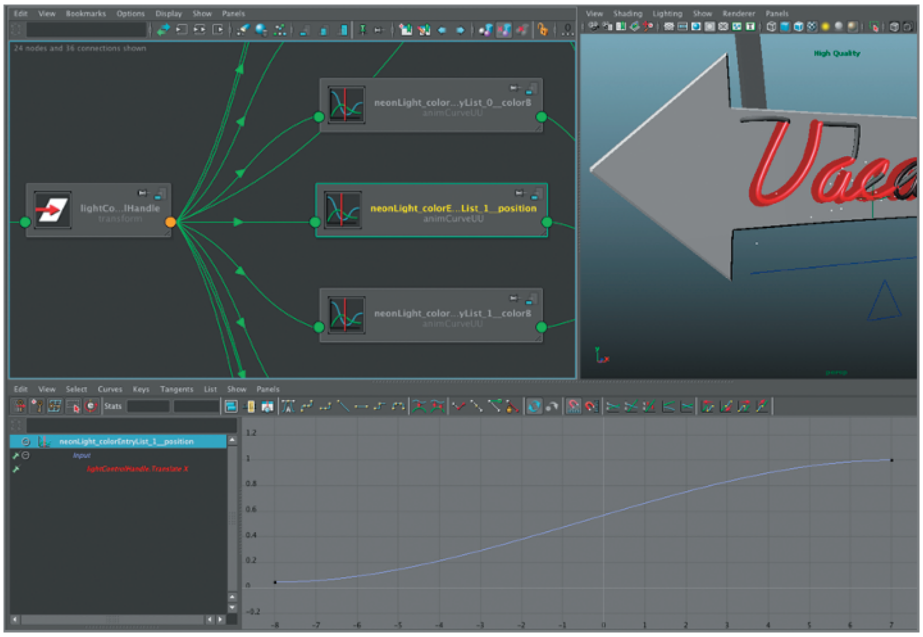

neonLight_color...yList_0_colorB
lightCo...lHandle
neonLight_colorE...List_1_position
neonLight_color...yList_1_colorB
neonLight_colorEntryList_1_position
High Quality

Figure 1.21 Select the animation curve nodes in the Node Editor; the Graph Editor updates to display the animation curve.

5. Use the same process to delete the following nodes. These are keyframes that have been set on the colors of the ramp, but you need only animation curves for the position of the ramp colors, not the color values themselves. Delete the following nodes:

   neonLight_colorEntryList_1_colorR

   neonLight_colorEntryList_1_colorG

   neonLight_colorEntryList_0_colorR

   neonLight_colorEntryList_0_colorG

   neonLight_colorEntryList_0_colorB

   neonLight_colorEntryList_0_Position

6. In the Node Editor, select the neonLight_colorEntryList_1_Position node, and Shift+select the motionPath1_uValue node. You'll see the animation curves appear on the Graph Editor below. Drag a selection around both curves, and press F to focus the view on the curves.

7. Press the Linear Tangents button to make both curves straight, which removes the easing in and out of the animation (see the top image in Figure 1.22).

   The animation curves on the graph indicate how the animation of the attributes is tied to the Translate X values of the lightControlHandle node. The X axis values correspond to the Translate X value of lightControlHandle, the Y axis values correspond to the values of neonLight_colorEntryList_1_Position and motionPath1_uValue, both of which range between 0 and 1. To make the animations a bit more in sync, you should edit the curves so that they match, as in the following steps.

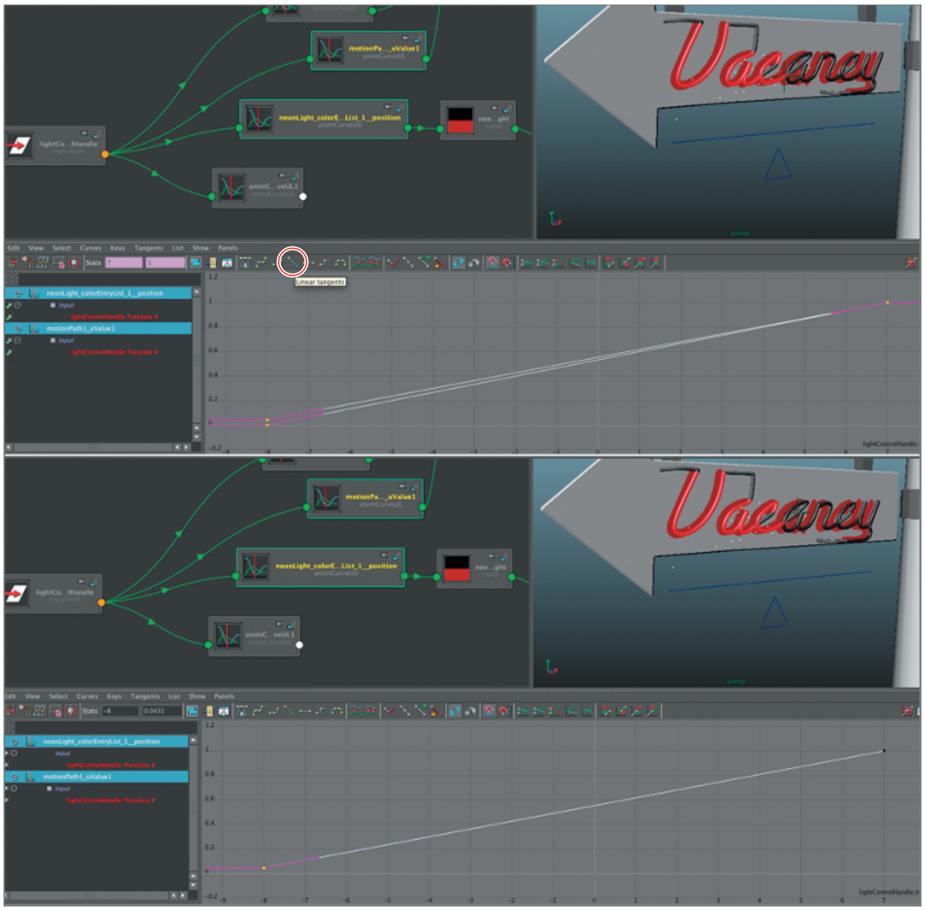


Figure 1.22 Set the animation curve tangents for the selected nodes to linear (top image). Move the first keyframe on motionPath1_uValue up to match the first keyframe value of neonLight_colorEntryList_1_Position (bottom image).

8. Select the lower keyframe on the far right, which is the starting value of motion-Path1_uValue. Press W to switch to move mode, and drag the key upward on the Graph Editor so that it matches the starting point of neonLight_colorEntryL-ist_1_Position (see the bottom image in Figure 1.22).
9. Play the animation; you should see that the position of the locator is more closely aligned with the leading edge of the color of the ramp as it moves along the tube. You can continue to tweak the position of the keyframes on the Graph Editor to refine the animation if you like.
10. Next you can edit the animation of the emitter rate. In the Node Editor, select the emitter1_rate node. The animation curve appears on the Graph Editor at the bottom of the window. Press F to focus on the curve.

    The graph shows that the rate starts at 0 when the lightControlHandle is all the way on the left. The rate gradually increases as the lightControlHandle moves to the right. It would most likely look better if the emitter started at a higher rate, increased, and then dropped off sharply as the lightControlHandle reaches the right side.

11. Use the Insert Key function and the Move Key function to add keyframes near the front and the end of the curve. Use the Move Keyframe tool to edit the keys on the Graph Editor so that they resemble Figure 1.23. This doesn't have to be an exact match; in fact, there are opportunities to create variations on the look of the particle effect through editing this curve. Just make sure you keep the start and end points at the same place so that the emitter is off when the light-ControlHandle is at either end of the lightControl.

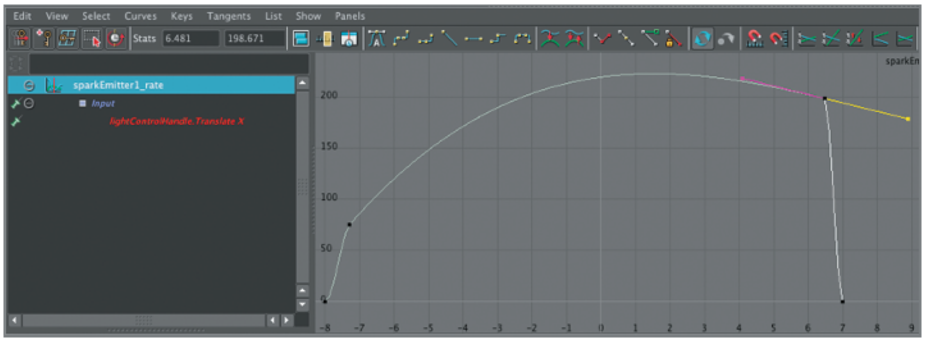
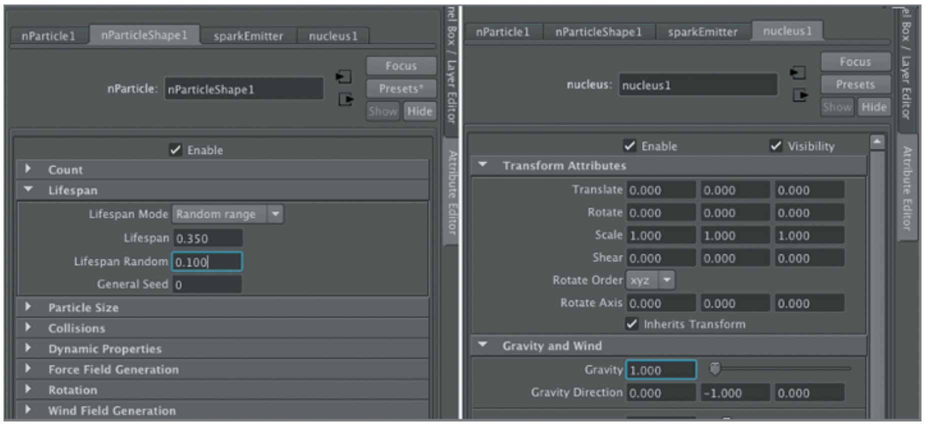


Figure 1.23 Edit the animation curve for the rate of the emitter.

12. Rewind and play the animation.

13. To make the nParticles behave more like sparks, open the Outliner window, and select the nParticle1 node. Open its Attribute Editor to the nPartcle1Shape tab, and set LifeSpanMode to Random Range. Set LifeSpan to 0.35 and Lifespan Random to 0.1. This means the nParticles will live for 0.35*1 seconds plus or minus 0.1 seconds (see left image in Figure 1.24).

14. Switch to the nucleus1 tab, and set the Gravity slider to 1 so that the nParticles don't fall quite as quickly (see right image in Figure 1.24).


nParticle1 | nParticleShape1 | sparkEmitter | nucleus1

nParticle: nParticleShape1

✓ Enable

Count

Lifespan

Lifespan Mode Random range
Lifespan 0.350
Lifespan Random 0.100
General Seed 0

Particle Size
Collisions
Dynamic Properties
Force Field Generation
Rotation
Wind Field Generation

nParticle1 | nParticleShape1 | sparkEmitter | nucleus1

nucleus: nucleus1

✓ Enable ✓ Visibility

Transform Attributes

Translate 0.000 0.000 0.000
Rotate 0.000 0.000 0.000
Scale 1.000 1.000 1.000
Shear 0.000 0.000 0.000
Rotate Order xyz
Rotate Axis 0.000 0.000 0.000
✓ Inherits Transform

Gravity and Wind

Gravity 1.000
Gravity Direction 0.000 -1.000 0.000

Figure 1.24 Edit the animation lifespan attributes for nParticle1 and the gravity strength for the Nucleus node to change the behavior of the particles.

15. Play the animation, and experiment with the timing. All you need to do is edit the keyframes set on the Translate X attribute of LightControlHandle. To edit the rate, select the sparkEmitter, and edit its keyframes on the Graph Editor.

To see a finished version of the scene, open the `signEnd.ma` file in the `scenes` folder of the `Chapter01_project`.

Further Study

Generally speaking, anything you can do to make animation easier is usually a good idea. While it's not too difficult to use a ramp to create the effect of the sign coming on, by taking the extra time to create a customized rig, you'll find that there's less work later. This becomes particularly important when deadlines start to get tight. You also learn more about how Maya works in the process.

As an additional challenge, see whether you can edit the rig so that movement of the arrow along the Y axis controls the rate of the emitter. This way, you don't have to open the Graph Editor every time you need to change the rate, which is an attribute your director will most likely want to tweak fairly often.

Do It with MEL

In this exercise, you probably noticed how much you needed to jump from one panel to another in Maya. You can often increase your productivity and reduce the number of button clicks by incorporating some basic MEL scripts into your workflow.

MEL is Maya's own scripting language. It can be used to do everything from automating simple repetitive actions to creating sophisticated custom interfaces that are integrated into Maya. We'll keep things simple so you can see how. Even if you're not the world's most experienced scripter, you can still get a lot of mileage out of a few basic tricks.

For many of the examples in this book, I have asked my friend and colleague Max Dayan to create some original scripts based on my exercises. Max is an extremely talented scriptwriter and Maya artist, and he is a highly respected instructor at the Gnomon School of Visual Effects in Hollywood. He is the perfect person to take you through the process of creating an original, production-ready script.

These scripts are formatted to make them easy to understand; some of the scripting techniques Max is using are designed to help you learn MEL. In production, Max's scripts are much more streamlined and efficient. You can find examples of the scripts in the `scripts` directory of the same `Chapter01_project` project that contains the files you've been using.

The following example shows how to write a script to connect the translation of the light control curve to the animation of the position of the color input marker on the ramp texture using a set-driven key. I'll let Max take it from here.

In this example, you will see a simplified version of connecting a controller (locator1) to the position attribute of a ramp (ramp1).

Continues

Do It with MEL *(Continued)*

To help this example along, the script will also create a polygon plane and a Blinn shader so that once the script is done, you need to click only a single button on the shelf to set up the entire rig. I want to stress that this is a simplified version of the script. It is meant as a beginning guide to help you understand how scripting can assist with redundant tasks. Let's start with some basic scripting principles.

- Computers are dumb. To get a computer to do what you want it to do, your instructions need to be clear and precise; otherwise, you will see an error message.
- Make sure you are spelling commands correctly and using the proper character case.
- A semicolon (`;`) tells Maya that the line of code is finished and to move on to the next line.
- A hyphen before text indicates a flag (e.g., `-someTextHere`). A flag is a parameter or setting that a MEL command has access to. Most of the time, a flag will require you to input a value.
- To create a new line in the script, press the Enter/Return key on the main part of your computer's keyboard (where all the letters are). To run a command, select the text, and press the Enter key on your computer's numeric keypad (laptops without a numeric keypad may require that you use a modifier key along with the Enter key). It is crucial to understand that the two Enter keys do two different things.
- When you need some help, type **`help;`** into the scripting editor, and press the Enter key on the numeric keypad of your computer.
- Maya's Script Editor is where you'll write the scripts. The Script Editor can work with either MEL or Python; you'll be using MEL in this book. Make sure that the tab in the Script Editor says MEL, not Python. To switch from one or the other, open the Script Editor, and choose Command › New Tab. A pop-up window will appear; click the MEL button to create a new MEL tab.
- If you're unfamiliar with using the Script Editor, review Maya's documentation on how to enter MEL scripts in Maya.
- Text that is preceded by a double slash (`//`) is a comment. Maya ignores any text after the double slashes. Comments are used to make the script easier for users to read.

Now you'll create a shader and the ramp. Both the shader and ramp use the command `shadingNode`. When creating the shader, you use the `-asShader` flag and then provide the type of shader you want to create. In this example, a Blinn is what you are using.

1. Open the Script Editor, and type the following lines:

```
shadingNode -asShader blinn;

// Using the same command with the -asTexture flag will create a
texture node. The type of texture node we want is a ramp.

shadingNode -asTexture ramp;
```

Continues

Do It with MEL *(Continued)*

2. Now you need to connect the ramp texture to the Blinn shader. The `connectAttr` command is how you will connect the two attributes. The first value (`ramp1.outColor`) will be connected to the second value (`blinn1.color`). This is no different from MMB dragging the ramp into the Color attribute of the Blinn in the Attribute Editor.

   ```
   connectAttr ramp1.outColor blinn1.color;
   ```

3. Create your geometry using the `polyPlane` command. In the following code line, you will see the long names of the flags to help you understand what each flag is actually doing. Often in scripting you will use the short names to speed up the scripting process.

   ```
   polyPlane -width 2 -height 2 -subdivisionsX 1 -subdivisionsY 1
   -axis 0 1 0 -createUVs 2 -constructionHistory 1;
   ```

   Because the plane is still selected, you can simply use the `hyperShade` command with the `-assign` flag.

   ```
   hyperShade -assign blinn1;
   ```

4. The last thing the script needs to create for the basic control rig is the locator. The `spaceLocator` command will create a locator at 0, 0, 0 in world space. You will use the locator as the controller for the ramp.

   ```
   spaceLocator -p 0 0 0;
   ```

At this point, you have created and connected a shader, texture, and polygon plane, and you have your controller (locator1). Now that you have created all the pieces you will need, it's time to set up the effect. For this example, you want two colors on your ramp. So, delete the middle (green) color entry. The `removeMultiInstance` command will delete the middle color entry. The color entries are stored in an array and count from 0 up. In a default ramp texture, the bottom color is `[0]` (red), the middle color is `[1]` (green), and the top color is `[2]`(blue).

```
removeMultiInstance -break true ramp1.colorEntryList[1];
```

The `setAttr` command allows you to set a value for any attribute you want. Some attributes will require a `-type` to be specified, and some will not. The easiest way to see whether an attribute has a type flag is to adjust manually in Maya and then look in the Script Editor for the resulting MEL command.

It's always a good idea to write out in English what you want to do and then translate it into MEL; even experienced scriptwriters do this. So, here's what you want to do next:

1. Set the ramp to have an Interpolation value of `none`. This will create an abrupt transition between color entries.
2. Set the top color to white (`1, 1, 1`).
3. Set the bottom color to black (`0, 0, 0`).

   ```
   setAttr "ramp1.interpolation" 0;
   setAttr "ramp1.colorEntryList[2].color" -type double3 1 1 1 ;
   setAttr "ramp1.colorEntryList[0].color" -type double3 0 0 0 ;
   ```

Continues

Do It with MEL *(Continued)*

Note that the `color` attribute of the ramp takes a `-type double3` value. This is because `color` is a three-part attribute (red, green, and blue) also known as a *vector*.

4. Now you need to set up the set-driven keys. What you want to do here is set the desired values of the two attributes you are connecting (`locator1.translateX` and `ramp1.colorEntryList[2].position`).

   ```
   setAttr "locator1.translateX" 1;
   setAttr "ramp1.colorEntryList[2].position" 1;
   ```

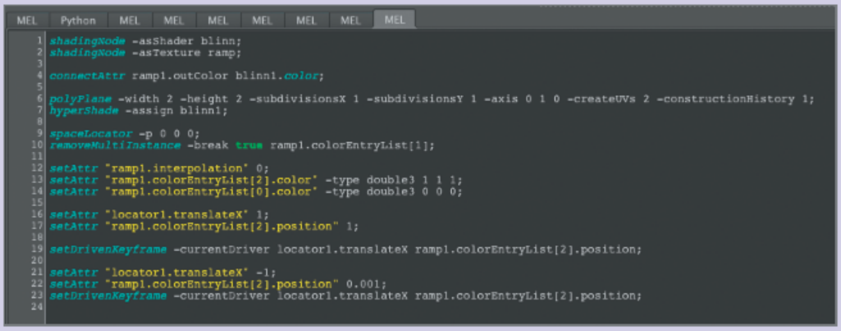
5. Once you have the values set, you simply run the `setDrivenKeyframe` command. The first value is the `driver` (`locator1.translateX`), and the second is the `driven` (`ramp1.colorEntryList[2].position`):

   ```
   setDrivenKeyframe -currentDriver locator1.translateX ramp1.
   colorEntryList[2].position;
   ```

6. Set the second attribute values and repeat the same `setDrivenKeyframe` command.

   ```
   setAttr "locator1.translateX" -1;

   setAttr "ramp1.colorEntryList[2].position" 0.001;

   setDrivenKeyframe -currentDriver locator1.translateX ramp1.
   colorEntryList[2].position;
   ```

That's the whole script; this shows how it looks in the Script Editor:



```
shadingNode -asShader blinn;
shadingNode -asTexture ramp;

connectAttr ramp1.outColor blinn1.color;

polyPlane -width 2 -height 2 -subdivisionsX 1 -subdivisionsY 1 -axis 0 1 0 -createUVs 2 -constructionHistory 1;
hyperShade -assign blinn1;

spaceLocator -p 0 0 0;
removeMultiInstance -break true ramp1.colorEntryList[1];

setAttr "ramp1.interpolation" 0;
setAttr "ramp1.colorEntryList[2].color" -type double3 1 1 1;
setAttr "ramp1.colorEntryList[0].color" -type double3 0 0 0;

setAttr "locator1.translateX" 1;
setAttr "ramp1.colorEntryList[2].position" 1;

setDrivenKeyframe -currentDriver locator1.translateX ramp1.colorEntryList[2].position;

setAttr "locator1.translateX" -1;
setAttr "ramp1.colorEntryList[2].position" 0.001;
setDrivenKeyframe -currentDriver locator1.translateX ramp1.colorEntryList[2].position;

```

To run the script, start a new scene, select the script in the Script Editor so that it's all highlighted, and then press the Enter key on your computer's numeric keypad. If all goes well, you'll see a polygon plane and a locator. Make sure shaded view is activated in Perspective view and then try moving the locator back and forth; you should see the ramp colors on the plane move as well. For best results, make sure High Quality Rendering is enabled in the viewport's Renderer menu. If it's not working, you may have left something out, or you may have a typo; go through the script again to see whether there is an error. Note that you could choose to set the ramp type to a U Ramp, and then the ramp would follow the same direction as the locator.

Continues

Do It with MEL *(Continued)*

Although the script you just created saves you a good deal of work, for this specific task, it isn't very dynamic. This script is what is known as *hard-coded*. Every object you created has a very specific name. Because of this, you can run this script only once and only if you haven't previously created a ramp, Blinn, or polyplane.

A better way to script is to use variables in the place of hard-coded object names. Max has written a more dynamic version of the script with a description of each command. Open the `rampCntrl.mel` script located in the `scripts` directory of the `Chapter01_project` and take a look at Max's code. Max has included descriptions of each step along the way. Many of the exercises in this book have his own MEL scripting examples included, and each script gets progressively more dynamic and a bit more challenging. The only way to learn MEL is to practice! We encourage you to go through each script and also check out the scripts we've included with the project files. See whether you can figure out how to adapt these techniques for your own projects. As you accumulate more scripts, you'll find that you are able to do more in less time in Maya, and that's the kind of thing that makes employers very happy.

Use Ambient Occlusion for Holographic Effects

For this next challenge, I want you to imagine you are working on a pitch for an effects shot in a sci-fi adventure television series. This means the studio you are working for is bidding on a project, and to win the job, you need to show them what you and your fellow artists are capable of. Pitches are fun because you get to flex your creative muscle and show off a bit. The downside is that even if you get the job, much of the work you do will never make it to the final cut. What's more is that studios can devote only limited resources (depending on the job), and the turnaround is often very fast. But it's a great way to learn to think on your feet.

The shot in question involves creating a novel look for a futuristic 3D computer interface. The final version will show a hologram of a mysterious island as it is being scanned by a satellite. The director wants to see lines of light that reveal the contours of the island, as shown in the style board in Figure 1.25.

Style Boards

Style boards are a type of storyboard that focuses on the look and feel of the final images rather than the action in the shot. Usually they are created by the art director using Photoshop, images from stock footage or the Internet, and 3D renders.

The director has also stated that he does not want a typical wireframe; he'd like something a bit more futuristic and different.

The basic 3D elements of the shot have already been created; the scene has been set up, lit, and textured; and an animated camera has been included. All you'll need to do is add the scanlines that appear across the contours of the island.

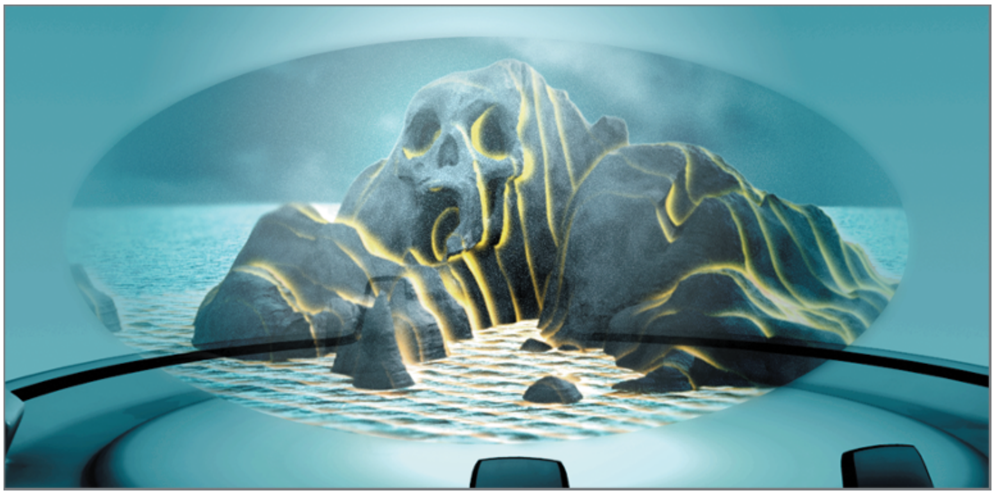


Figure 1.25 The style board showing a possible look for the lighted lines of the hologram

The most obvious technique would be to project an animated texture onto the 3D model, but in the interest of achieving something more interesting, you can take a different approach. You can use 3D geometry to intersect the island and the ocean surface and repurpose an ambient occlusion texture node to make bright lines appear along the intersection points. The neat thing about this is that you can use the shape of the intersecting geometry to create a wide variety of different looks fairly quickly. This gives the director some choices to help him decide which look he would like to present to the potential client.

Here's how you get started:

1. Open the `holoGramStart.ma` file from the Scenes directory of the `Chapter01_project` This can be downloaded from the book's support site at `www.sybex.com/go/mayavisualeffects2e`. The scene file contains the basic island geometry as well as geometry for the ocean and a simple sky dome. The camera pivots around the front of the island.
2. You can test the technique using a single plane. Choose Create › Polygon Primitives › Plane to add a plane to the scene. If Interactive Create is checked at the bottom of the Create menu, Maya will ask you to draw the plane on the grid; if this option is not on, the plane appears at the center of the grid (see Figure 1.26). Either way, use the Channel Box to set the following dimensions and position of the plane:

   Translate X, Y, and Z: 0

   Rotate X: 90

   Rotate Y: 0

   Rotate Z: 0

   Scale X: 120

   Scale Y: 1

   Scale Z: 120

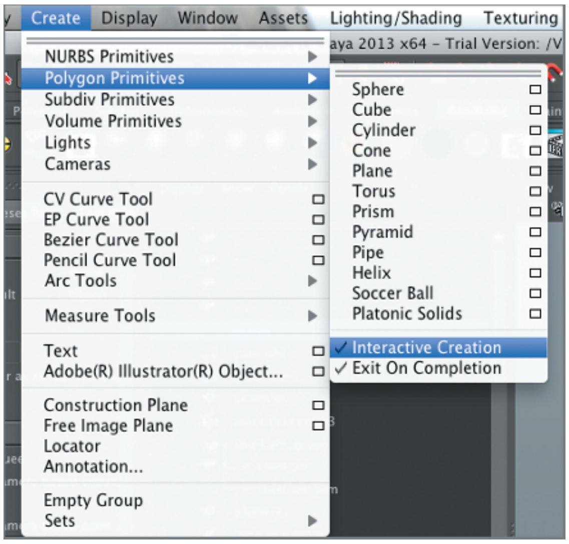


Figure 1.26 Create a polygon plane. If Interactive Create is on, Maya will ask you to draw the plane on the grid.

3. The plane appears intersecting the island geometry. Name the plane lightBeam01 (see Figure 1.27). Create a Lambert shader, and apply the texture to the lightBeam01. Name the Lambert material lightLineMat.

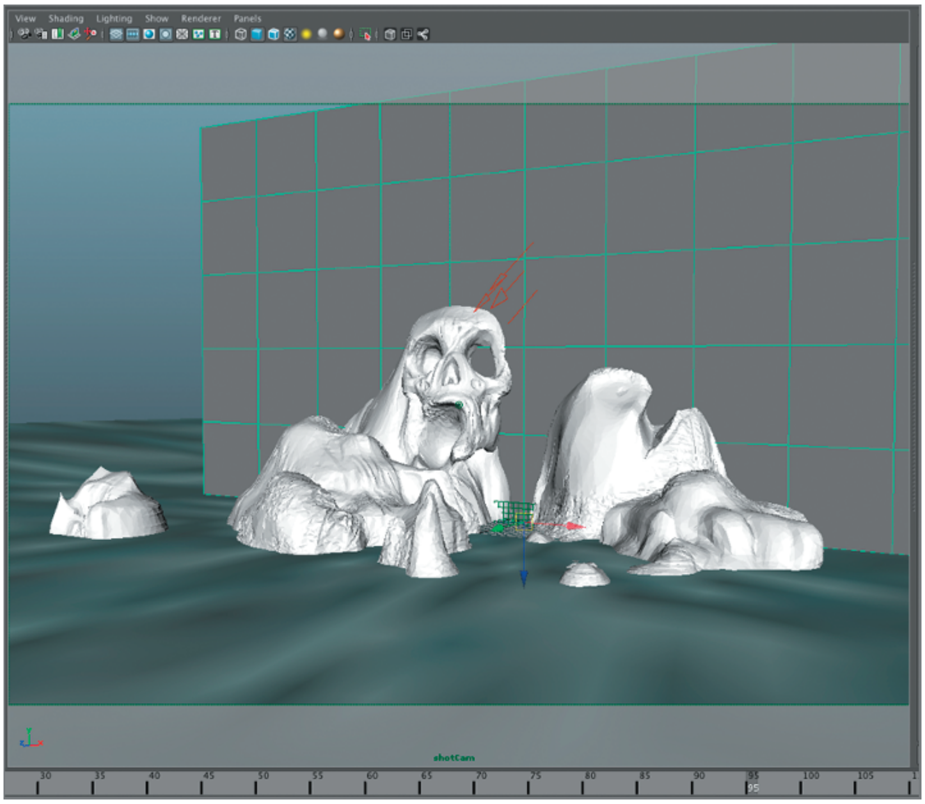


Figure 1.27 The plane intersects the geometry of the island.

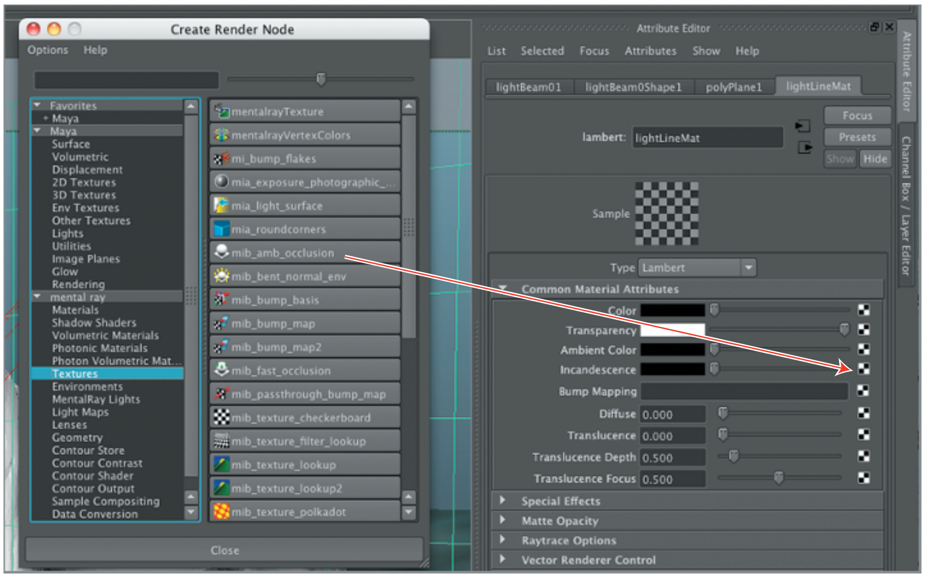
4. Open the Attribute Editor for lightLineMat. Set the following attributes so that the plane itself does not appear in the render:

   Color: Black

   Transparency: White

   Diffuse: 0

5. Click the checkered box to the right of the Incandescence channel for lightLine-Mat. This opens the Create Render Node window. In the mental ray section, click the mib_amb_occlusion icon to connect a new ambient occlusion texture to the incandescence of the lightLineMat shader (see Figure 1.28).



lambert: lightLineMat

Type: Lambert

Diffuse 0.000

Translucence 0.000

Translucence Depth 0.500

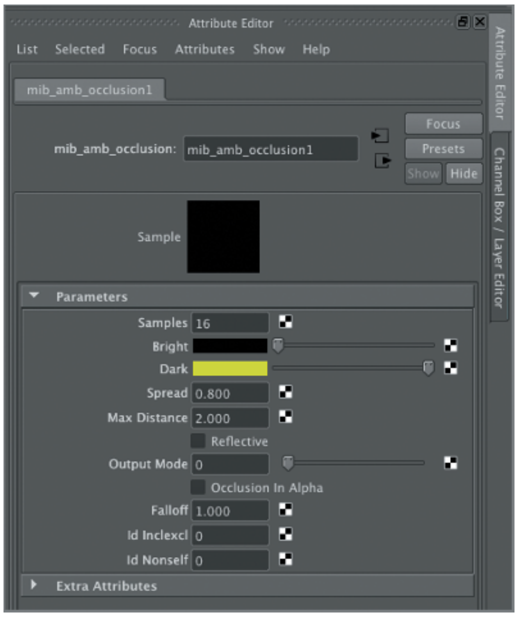
Translucence Focus 0.500

Figure 1.28 Create a new mib_amb_occlusion1 node, and connect it to the incandescence of lightLineMat using the CreateRenderNode window.

6. The Attribute Editor automatically switches to the settings for this node. In the settings for mib_amb_occlusion1, set the Bright value to black and the Dark value to a light greenish yellow. Set Max Distance to 2 (see Figure 1.29).

7. Open the Attribute Editor for lightBeam01. On the lightBeamShape01 tab, under Render Stats, turn off Casts Shadows and Receive Shadows (see Figure 1.30). By deactivating these options, there is no chance the polygon plane will affect the sunlight cast on the island geometry.

8. Open the Render Settings window. The Render Using menu should already be set to Mental Ray. All of the options have been set so that the scene renders with final gathering and the resolution is set to 800 × 454. This setting creates an image that is small enough to test the look without taking too long to render.

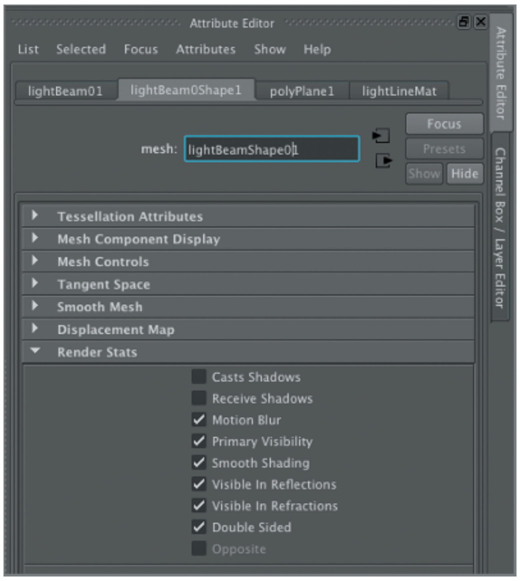


mib_amb_occlusion: mib_amb_occlusion1

Sample

Samples	16
Bright	⬛
Dark	🟨
Spread	0.800
Max Distance	2.000
	☐ Reflective
Output Mode	0
	☐ Occlusion In Alpha
Falloff	1.000
Id Inclexcl	0
Id Nonself	0

Figure 1.29 Choose colors and settings for the mib_amb_occlusion1 texture.

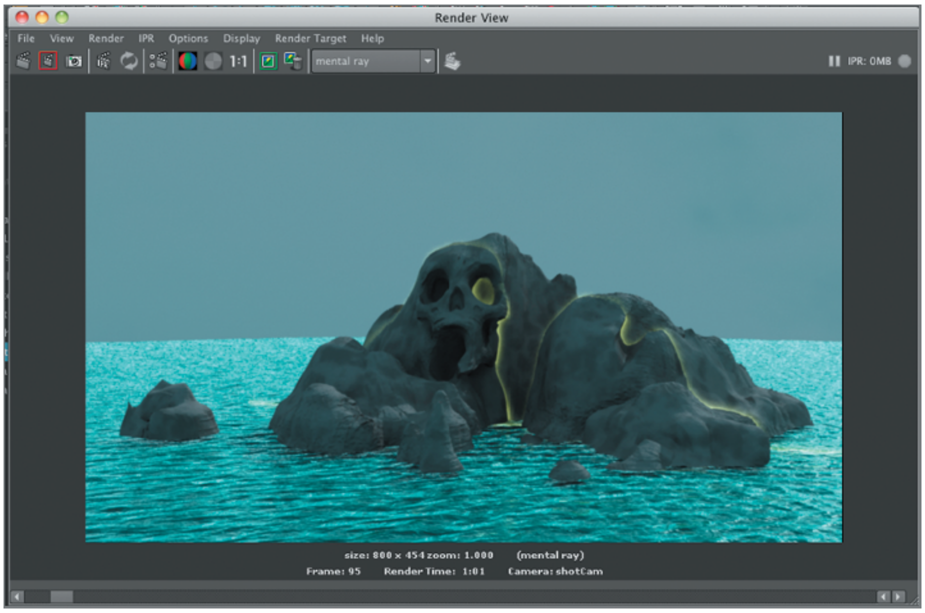


mesh: lightBeamShape01

☐ Casts Shadows
☐ Receive Shadows
✓ Motion Blur
✓ Primary Visibility
✓ Smooth Shading
✓ Visible In Reflections
✓ Visible In Refractions
✓ Double Sided
☐ Opposite

Figure 1.30 Turn off Cast and Receive Shadows in the Render Stats section of the Attribute Editor for the lightBeamShape01 node.

9. Open the Render View window, and from the Render menu, choose Render › Render › shotCam. After a minute or so, the rendered image appears in the Render View window. Click the Keep Image button to keep this image stored in the memory of the Render View window. This way, you can compare it to future renders as you adjust the settings (see Figure 1.31).

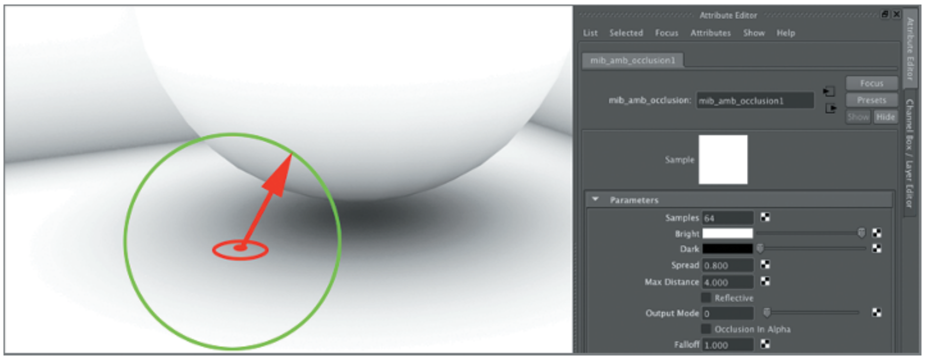


```
size: 800 x 454 zoom: 1.000     (mental ray)
Frame: 95     Render Time: 1:01     Camera: shotCam
```

Figure 1.31 Render the image using the Render View window.

The render appears after a minute or so; what you should see is the same island model but with a yellow line stretching across the surface where the lightBeam01 plane and the island and ocean geometry intersect.

Normally ambient occlusion is used to add ambient shadowing to a surface. When rays from the rendering camera hit each point of the surface, mental ray sends a second series of rays bouncing off the initial point into various directions in space. If a second surface is encountered by one of these bounced rays within the distance specified by Max Distance in the texture node, then the surface is shaded using a mixture of the colors specified in the swatches of the ambient occlusion node. These swatches are labeled "Bright" and "Dark." Objects that are closer together get a higher percentage of the Dark color; objects that are farther away get a higher percentage of the Bright color. Anything beyond the Max Distance setting is colored using the RGB values in the Bright Color channel of the ambient occlusion settings (see Figure 1.32).


Attribute Editor
List Selected Focus Attributes Show Help
Focus
Presets
Show Hide
Attribute Editor
Channel Box / Layer Editor

mib_amb_occlusion1

mib_amb_occlusion: mib_amb_occlusion1

Sample

Parameters

Samples 64
Bright
Dark
Spread 0.800
Max Distance 4.000
Reflective
Output Mode 0
Occlusion In Alpha
Falloff 1.000

Figure 1.32 Each point of the surface using the texture is sampled so that objects closer together receive more of the Dark color specified in the texture and objects that are farther away receive more of the Bright color specified in the texture.

In the case of the lightBeam01 plane that has the lightLineMat applied to it, the Dark color swatch of the mib_amb_occlusion texture has been set to a light green, and the Bright color swatch has been set to black; the result is inverted, so instead of ambient shadowing, you see a greenish glow on the plane wherever the points are within two units of the intersection between LightBeam01 and the island and ocean geometry. By plugging this into the incandescence of a transparent Lambert, the rest of the plane is invisible, and the result looks like a bright green line of light across the island surface.

10. Save the scene to your local drive.

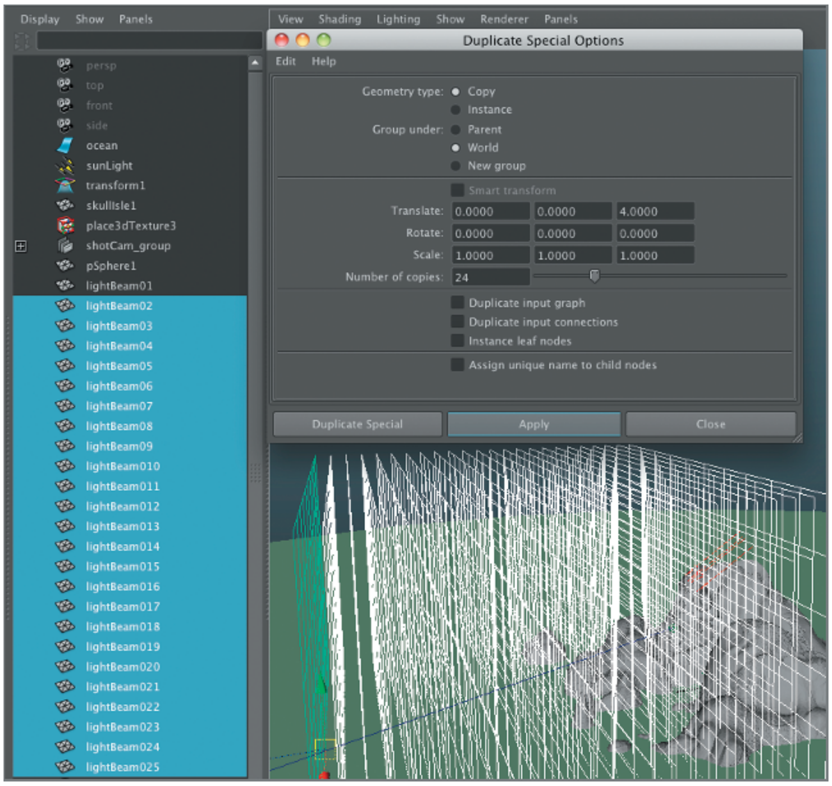
Mib_amb_occlusion Max Distance

By default the Max Distance setting in the attributes of the mib_amb_occlusion texture node is set to 0. mental ray interprets 0 to mean the same as infinity in this situation, meaning that mental ray will try to calculate how close every single point in the scene is to that initial point of contact where the camera ray hits the surface. This can slow down your render a lot and also produce a very dark surface in some situations. Always set Max Distance to a value above 0 to cut down on unnecessary calculations and save render time. When things go wrong with a shader that uses this texture as part of the network, always check to see whether this setting has been left at zero as a possible reason for the problem.

Animate the Glowing Line Effect

Now that the plane has been positioned and the shader has been created, things get a little more fun. By finding creative ways to intersect the plane geometry with the island and the ocean geometry, you can create some variations to present to the director. The nice thing about this setup is that it offers more flexibility than projecting a texture through a light, and it can all be done within Maya without the need for external programs. The same shader can be applied to any geometry that intersects the island and ocean geometry.

1. Select the lightBeam01 plane, and choose Edit › Modify › Freeze Transformations. This means the current translate and rotate values in X, Y, and Z are set to 0, and the current scale in X, Y, and Z is set to 1. This makes it easier to base the position and scale attributes of duplicate planes on the current position of this one.
2. To create multiple lines on the surface, you can simply duplicate the existing lightBeam01 plane. Select the plane, and choose Edit › Duplicate Special › Options. In the options, set Geometry Type to Copy, Group Under to World, Translate Z to 4, and Number Of Copies to 24.
3. Click the Apply button. Maya adds 24 copies of the plane to the scene, and each one is moved 4 units in Z from the previous copy, resulting in an array of planes that form a cube (see Figure 1.33).
4. In the Outliner, Shift+select all 25 lightBeam surfaces, and group them (Ctrl+B). Name the group lightScanPlanes.

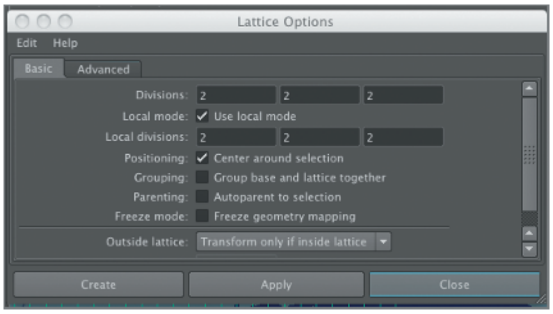


Geometry type: [● Copy] Instance
Group under: Parent [● World] New group
Smart transform
Translate: 0.0000 0.0000 4.0000
Rotate: 0.0000 0.0000 0.0000
Scale: 1.0000 1.0000 1.0000
Number of copies: 24
Duplicate input graph
Duplicate input connections
Instance leaf nodes
Assign unique name to child nodes

Figure 1.33 Duplicate the lightBeam01 plane to create an array.

To make the lines appear like they are scanning the island, you can have each plane intersect the island one after the other starting from the back and moving toward the front. Setting keyframes for the translation of each plane is a bit on the tedious side, so to save time, you can apply a Lattice deformer to the group of planes, deform the Lattice deformer itself as a way to offset the planes in space, and then simply animate the translation of the Lattice deformer. It may sound complex at first, but it's actually pretty easy.

5. Select the lightScanPlanes group in the Outliner. Switch to the Animation menu set, and choose Create Deformers › Lattice › Options. In the options, set all three fields for Divisions to 2. All three fields for Local Divisions should also be 2. Click Create to make the Lattice deformer (see Figure 1.34).



Divisions: 2 2 2
Local mode: ✓ Use local mode
Local divisions: 2 2 2
Positioning: ✓ Center around selection
Grouping: Group base and lattice together
Parenting: Autoparent to selection
Freeze mode: Freeze geometry mapping
Outside lattice: Transform only if inside lattice

Figure 1.34 The options for the Lattice deformer

6. The Lattice deformer is essentially a cube that fits over the entire group of planes. Select the ffd1Lattice node in the Outliner, and rename it lightPlaneLattice.
7. Switch to a side view, and press the 4 key to switch to wireframe mode so that you can see the planes. Select lightPlaneLattice, and move it above the island geometry, as shown in Figure 1.35.

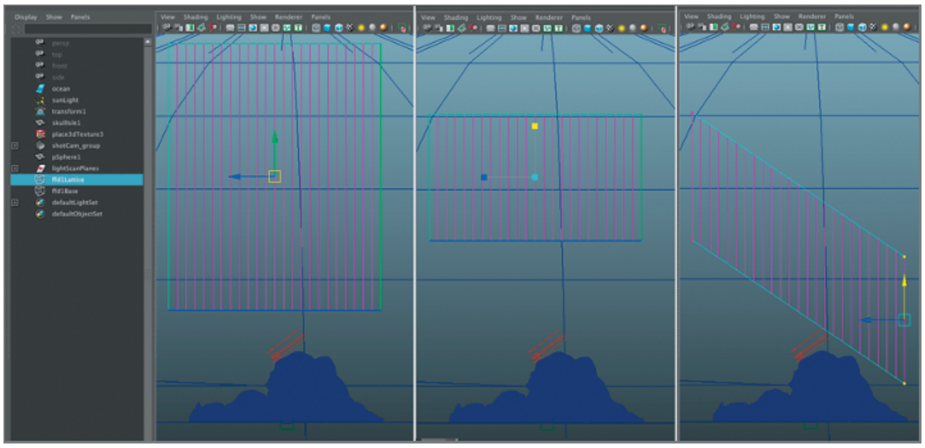


Figure 1.35 Move the Lattice deformer above the island geometry (left image), reduce the scale of the Lattice deformer along the Y axis (center image), and move the points on the right side of the Lattice deformer to offset the planes along the Y axis (right image).

8. The planes are a bit tall, so you can use the Lattice deformer to scale the group down a little. With the Lattice deformer selected, press the R hotkey to switch to the Scale tool. Reduce the scale of the Lattice deformer along the Y axis so it's just little taller than the island.
9. With lightPlaneLattice, switch to component selection mode so that you can select the points of lightPlaneLattice. Drag a selection marquee over the points on the right side of the Lattice deformer, and move these points downward. This deforms the group so that each plane is offset along the Y axis (the right image in Figure 1.35).
10. Now it's just a simple matter of setting keyframes on the Y translation of the lightPlaneLattice to animate the lines progressively scanning the island. Set the Timeline to frame 0. In the Outliner, select lightPlaneLattice, and press Shift+W to keyframe the translation channels.
11. Set the Timeline to frame 24. From the side view, move the lightPlaneLattice so that it's below the island. Press Shift+W to set another keyframe.
12. With lightPlaneLattice selected, open the Graph Editor. From the menu within the Graph Editor, choose View › Infinity so that the Graph Editor shows the animation curves beyond the last keyframe. Drag a selection marquee around the animation curves, and press the F hotkey to focus the view on the curves.
13. With the keyframes selected, choose Tangents › Linear to make the keyframes straight lines (if they aren't already). Choose Curves › Post Infinity › Cycle so

that the motion repeats an infinite number of times after the last keyframe. This means each time the Lattice deformer reaches the bottom, it will pop back up to the top and repeat the motion, making it look like the scan is repeating over and over.

14. You can select the animation curves for Translate X and Translate Z and delete them if you want to keep the Graph Editor nice and tidy. Figure 1.36 shows the resulting animation curves.
15. Save the scene to your local hard drive.

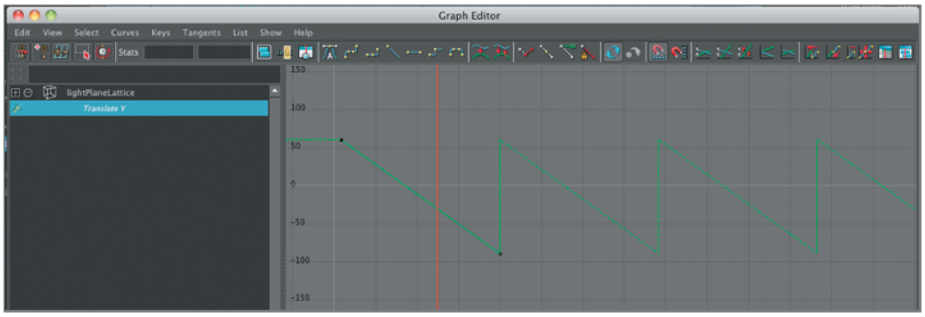


Figure 1.36 The keyframes for the Y translation of the Lattice deformer on the Graph Editor are set up to create an infinitely repeating loop.

The resulting animation should have the lightPlaneLattice repeatedly moving down from above and passing through the island and ocean geometry. Of course, to see the effect in action, you can render the sequence.

Render the Effect Using a Render Layer

If you render out the scene using the current setup, the effect will work but it's hard to see. You can use a render layer to isolate the scanning beams of light effect from the geometry of the island. This approach has several benefits. For example, since the island and ocean have already been rendered, you don't need to rerender them. Creating a render layer without the lighting and shading will also save time, and you have more options for changing things such as the color and the brightness when you composite the effect over the original animation. Using a program such as Adobe After Effects, you can make changes in the overall look without having to re-create the original Maya render. This is very helpful, especially if the art director is a little indecisive.

1. Continue with your saved scene from the previous section. In the Layer Editor below the Channel Box, click the Render button to switch to the Render Layer Editor.
2. Right-click the layer, and choose Copy Layer. This creates a duplicate of the masterLayer (the left image in Figure 1.37). The duplicate is named defaultRenderlayer1. Double-click the name of the layer, and change it to scanLinesLayer. Click the clapboard icon to the left of masterLayer to turn the green check mark into a red *x*. This means that the masterLayer will not render (the right image in Figure 1.37).

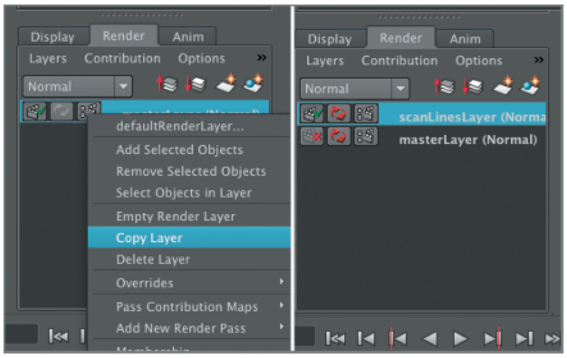
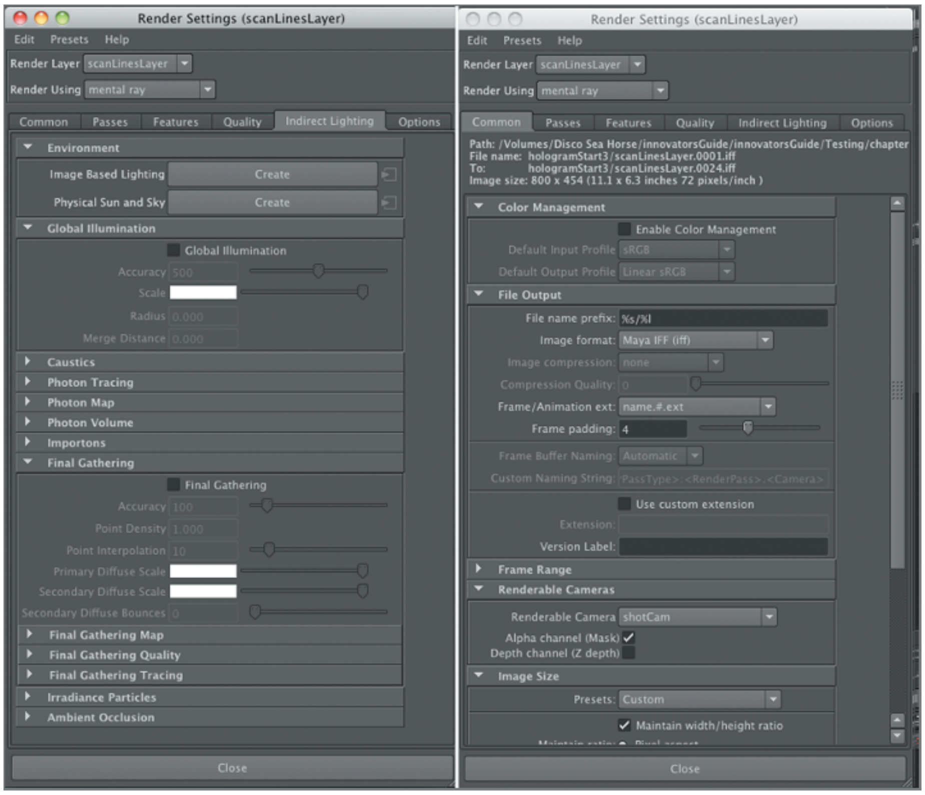


Figure 1.37 Copy the masterLayer in the Render Layer Editor and rename it scanLinesLayer.

3. Select the scanLinesLayer layer, and click the Render Settings icon to open the Render Settings window. Click the Indirect Lighting tab, and expand the Final Gathering settings. Disable the check box next to Final Gathering. This effect is not needed to create the scanlines and only adds to render time, so it's a good idea to turn it off.

4. On the Common tab, make sure that shotCam is selected as the renderable camera. In the field next to Filename Prefix, type **%s/%l**. These tokens tell Maya to place the sequenced images it renders in a folder named after the scene, and each file is named after the layer (see Figure 1.38).


Render Settings (scanLinesLayer)

Render Layer: scanLinesLayer
Render Using: mental ray

Global Illumination: Accuracy 500, Scale, Radius 0.000, Merge Distance 0.000

Final Gathering: Accuracy 100, Point Density 1.000, Point Interpolation 10, Primary Diffuse Scale, Secondary Diffuse Scale, Secondary Diffuse Bounces 0

Path: /Volumes/Disco Sea Horse/innovatorsGuide/innovatorsGuide/Testing/chapter
File name: hologramStart3/scanLinesLayer.0001.iff
To: hologramStart3/scanLinesLayer.0024.iff
Image size: 800 x 454 (11.1 x 6.3 inches 72 pixels/inch)

Default Input Profile: sRGB
Default Output Profile: Linear sRGB
File name prefix: %s/%l
Image format: Maya IFF (iff)
Image compression: none
Compression Quality: 0
Frame/Animation ext: name.#.ext
Frame padding: 4
Frame Buffer Naming: Automatic
Custom Naming String: <PassType>:<RenderPass>.<Camera>
Renderable Camera: shotCam
Alpha channel (Mask) ✓
Presets: Custom
✓ Maintain width/height ratio

Figure 1.38 Turn off Final Gathering for the scanLinesLayer layer in the Render Settings window (left image). Add tokens to the Filename Prefix field on the Common tab so that the image sequence is named after the layer (right image).

5. Set Frame/Animation Ext. to name.#.ext and Frame Padding to 4. Make sure that Start Frame is set to 1 and End Frame is set to 120. The sequence will be named `scanLinesLayer.0001.iff`. Naming the images after the render layer makes compositing a little easier.

   To isolate the render lines, you can't simply hide the island and ocean geometry in the scanLines layer because then the lightBeam planes will have nothing to intersect, and the ambient occlusion calculation won't take place. If this happens, then the lines won't render. So, to get around this, you can apply a black surface shader to the geometry.

6. Open the Hypershade, and click the Surface Shader button in the list of Maya shaders. Name the new node hideGeoMat.
7. Select the island geometry in the viewport. In the Hypershade, right-click hideGeoMat, and choose Assign Material To Selected. Assign this material to the skyDome geometry as well (see Figure 1.39).

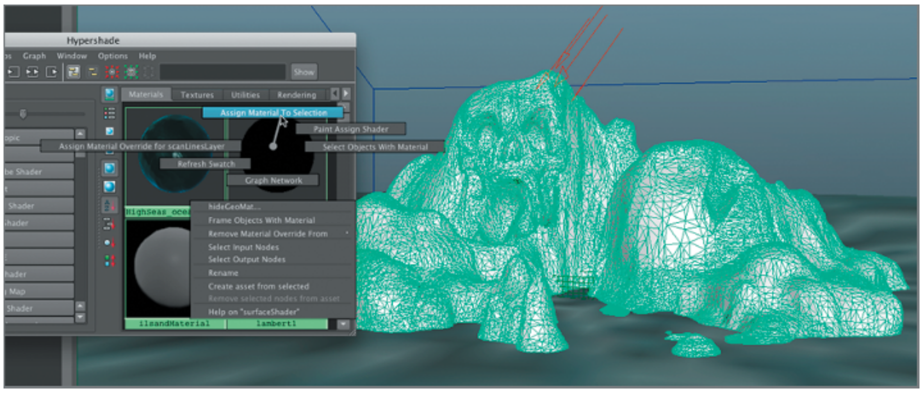
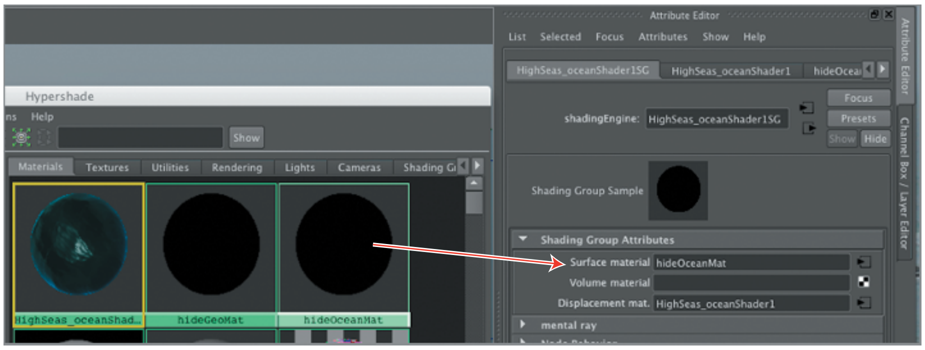


Figure 1.39 Assign the hideGeoMat material to the island geometry.

You can use the same material for the ocean as well except that the ocean geometry has a texture displacement applied to it, which creates the look of the waves on an otherwise flat surface. If you apply the surface shader directly to the ocean geometry, you'll lose the connection between the displacement shader and the ocean, and the scanlines won't match the shape of the water correctly in the final composite. The following steps demonstrate how you can apply the black surface shader to the ocean without losing the shape of the waves created by the displacement.

8. Create a second surface shader, and name it hideOceanMat.
9. In the Hypershade, click the Highseas_oceanShader1 material. This is the shader that was already applied to the ocean geometry in the original scene. Open its Attribute Editor. Click the Graph Input and Output connections so that you can see the Highseas_oceanShader1SG node. This is the shading group node that connects the material to the ocean geometry.
10. MMB drag the hideOceanMat from the Hypershade to the SurfaceMaterial slot on the HighSeas_oceanShader1SG tab of the Attribute Editor (see Figure 1.40).



shadingEngine: HighSeas_oceanShader1SG

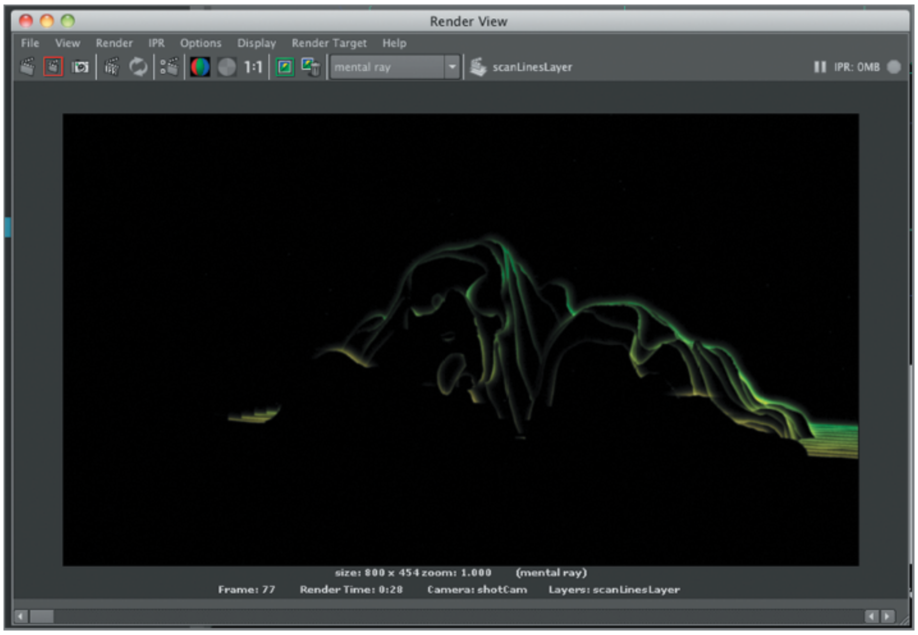
Surface material hideOceanMat

Volume material

Displacement mat. HighSeas_oceanShader1

Figure 1.40 Assign the hideGeoMat material to the island geometry.

11. Create a test render in the Render View window using the shotCam camera. Figure 1.41 shows the result.



```
size: 800 x 454 zoom: 1.000     (mental ray)
Frame: 77    Render Time: 0:28    Camera: shotCam    Layers: scanLinesLayer
```

Figure 1.41 The scanLines layer is rendered in the Render View window.

At this point, you can render the sequence and view the result using FCheck. The image sequence created by the scanLines layer can be composited on top of the sequence that has already been rendered. Using a blending mode such as Screen in the compositing software, the lines appear over the top of the island geometry creating a cool effect. Open the `hologramDone.mov` movie from the `movies` folder in the `Chapter01_project` directory to see how the finished composite looks. To see a version of the completed scene, open `hologramEnd.ma` from the `scenes` folder in the `Chapter01_project` directory.

Further Study

You can use this approach to create a wide variety of effects. Try creating an array of different shapes such as cylinders, cones, or spheres, and animate them intersecting the island geometry in different orientations to see what other effects you can create. Try using a similar arrangement to highlight the contours of a mechanical object.

Do It with MEL

Max Dayan has created a little script that automates creating a Lattice deformer with clusters for handles. This script automates the process of setting up the lightPlaneLattice deformer described in this tutorial, but you can use this same script for many other similar tasks. You can type the following text into the Script Editor or open the `latticeClusterCntrl.mel` file found in the `scripts` directory of the `Chapter01` project.

This script creates a polygon plane and then duplicates the plane using a loop. The planes are placed into a group, and then the Lattice deformer is applied to the group. Clusters are then attached to the Lattice deformer points, giving you an easy way to animate the Lattice deformer. It's important to note that some of the commands are enclosed in the ` mark. This is found above the Tab key on your computer; don't confuse this with the apostrophe key (') or the script won't work! The script also uses variables, which are preceded with the $ sign.

To test the script one line at a time, select the text of the command in the Script Editor, and press the Enter key on the numeric keypad. This is a good way to troubleshoot whether the script has errors or typos. It's also a good way to understand what each line of the script does.

1. Create the polygon plane.

```
string $planeGeo[]=`polyPlane -axis 1 0 0 -height 10 -width 10
-subdivisionsX 1 -subdivisionsY 1`;

string $groupName=`group`;
```

2. Duplicate the plane.

```
select $planeGeo[0];

duplicate -rr;

move -r 1 0 0;
```

3. Create an array of groups using a loop; this loop repeats the duplicate command from the previous line, adding a translational offset on the X axis.

```
for ($i=1; $i<23; ++$i){

     duplicate -rr -st;

}

select $groupName;
```

Continues

Do It with MEL *(Continued)*

4. Create a Lattice deformer around the group.

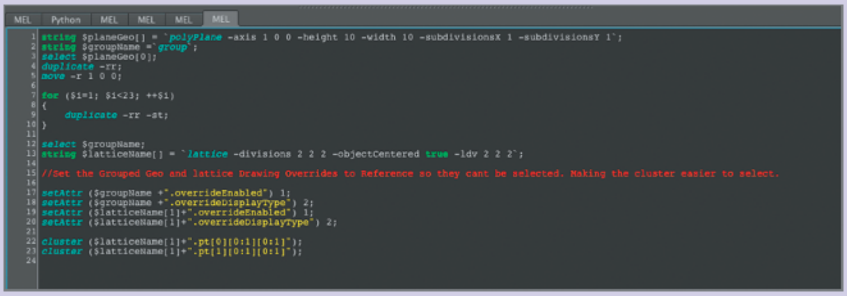
```
string $latticeName[]=`lattice  -divisions 2 2 2 -objectCentered
true  -ldv 2 2 2`;
```

5. Set the grouped geometry and Lattice deformer's Drawing Overrides settings to Reference so they can't be selected, making the cluster easier to select.

```
setAttr ($groupName +".overrideEnabled") 1;

setAttr ($groupName +".overrideDisplayType") 2;

setAttr ($latticeName[1]+".overrideEnabled") 1;

setAttr ($latticeName[1]+".overrideDisplayType") 2;
```

6. Create clusters for each end of the Lattice deformer.

```
cluster ($latticeName[1]+".pt[0][0:1][0:1]");

cluster ($latticeName[1]+".pt[1][0:1][0:1]");
```



```
string $planeGeo[] = `polyPlane -axis 1 0 0 -height 10 -width 10 -subdivisionsX 1 -subdivisionsY 1`;
string $groupName =`group`;
select $planeGeo[0];
duplicate -rr;
move -r 1 0 0;

for ($i=1; $i<23; ++$i)
{
    duplicate -rr -st;
}

select $groupName;
string $latticeName[] = `lattice -divisions 2 2 2 -objectCentered true -ldv 2 2 2`;

//Set the Grouped Geo and lattice Drawing Overrides to Reference so they cant be selected. Making the cluster easier to select.

setAttr ($groupName +".overrideEnabled") 1;
setAttr ($groupName +".overrideDisplayType") 2;
setAttr ($latticeName[1]+".overrideEnabled") 1;
setAttr ($latticeName[1]+".overrideDisplayType") 2;

cluster ($latticeName[1]+".pt[0][0:1][0:1]");
cluster ($latticeName[1]+".pt[1][0:1][0:1]");
```

Generate Creative Text Effects

You may think of commercials as those annoying interruptions you have to suffer through while watching TV, but in fact they are a major source of work for CG artists. They may not be as glamorous as working on a blockbuster film, but you may find that they often present interesting creative challenges.

For this scenario, I want you to imagine that your supervisor has asked you to create text-based effects for a sports event package. For this shot, a blimp is moving across the screen. On the side of the blimp, a lighted sign announces "2014 Curling Invitational Championship Monday Night!!!" This seems simple enough, but of course there is a catch: the sign has to look like individual lights on the side of the blimp. The text is moving like a typical lighted sign, but at the same time, the individual lights fly in from midair dynamically and then attach themselves to the blimp to form the sign. Figure 1.42 shows the storyboard for the effect.

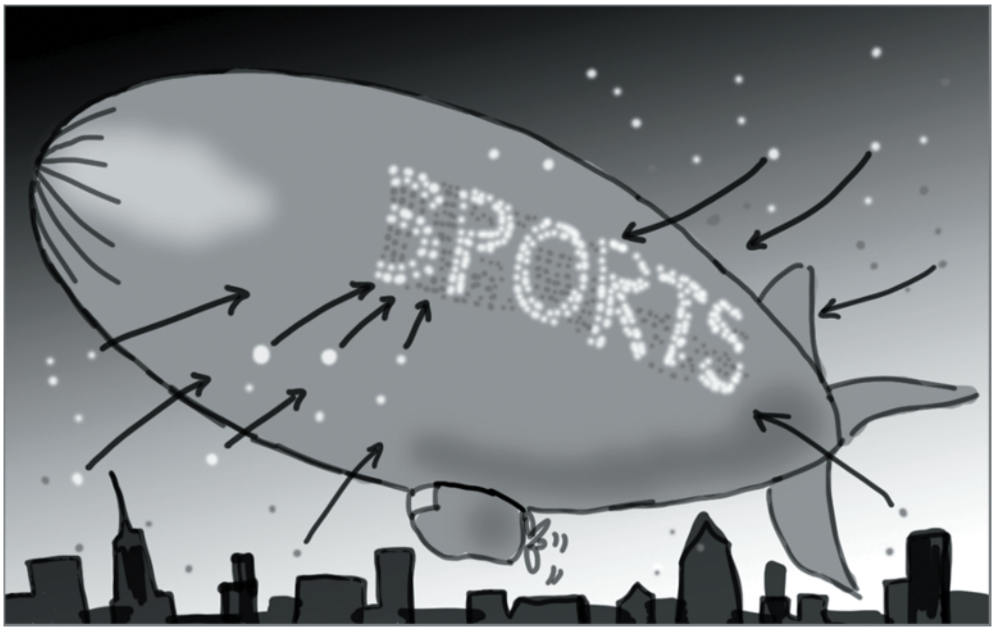


Figure 1.42 The storyboard image for the blimp effect

This presents a number of challenges. First you'll need to create the animated sign that looks like an array of lights; then, you'll need to create the dynamic simulation of the sign forming on the side of the blimp as it flies through the air. The blimp has already been modeled, but that's about it. You'll need to create the rest, and this tutorial shows you how to create the entire effect with Maya alone.

Create the Text

Creating the text at first seems deceptively easy: just render some animated text and project it onto the geometry on the side of the blimp. The problem is that the sign on the blimp needs to look like an array of individual lights. Each light in the array is either on or off. If you simply project animated text, it won't look like a lighted sign that is made up of individual lights; it will look like an animated CG texture. The key to solving the first part of the challenge is about creative use of texture resolution.

Think of the lighted sign on the side of a blimp as a grid. Each square in the grid can be thought of as an individual pixel in the animated texture. So, if you render a very low-resolution sequence of the text moving across the grid and then scale up the texture, each pixel should look like a square light turning on or off to form the text once it is projected onto the geometry. To really sell the effect, the animated texture should be aliased, meaning that the edges of the letters should look "blocky" or "jagged." This is exactly the opposite of what you normally want in your CG renders.

To start, you'll need to determine the size of the geometry on the side of the blimp, and then figure out how to render some low-resolution text that will then be projected onto that geometry.

1. Open the `blimpStart.ma` file from the `scenes` directory of the `Chapter01_project`. This can be downloaded from the book's support site at `www.sybex.com/go/mayavisualeffects2e`. This scene has the blimp sitting at the center of the grid.

The black area on the side of the blimp represents where the sign will need to be placed (see Figure 1.43).

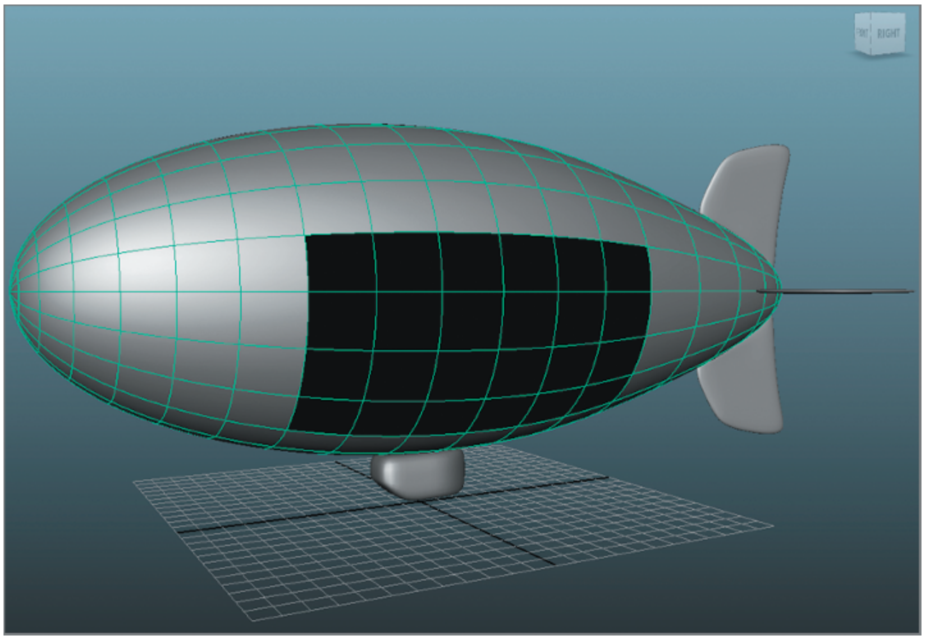


Figure 1.43 The blimp model has a large black area that indicates where the sign will need to go.

2. Create a polygon plane, and place it beside the blimp near the dark area. Use the following settings for the plane:

   Translate X: 11.414

   Translate Y: 8.741

   Translate Z: -0.0323

   Rotate X: 0

   Rotate Y: 0

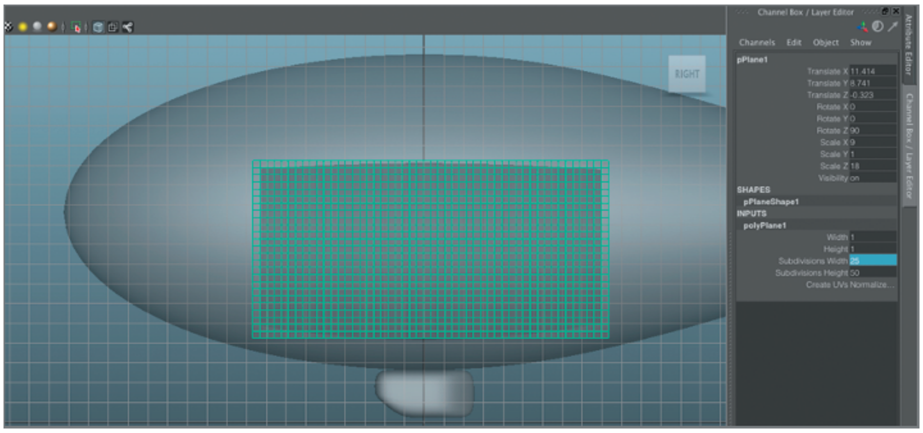
   Rotate Z: 90

   Scale X: 9

   Scale Y: 1

   Scale Z: 18

3. The result is a plane that is half as tall as it is wide. Turn on Wireframe On Solid so you can see the divisions of the plane. In the Channel Box under INPUTS, set Subdivisions Width to 25. Set Subdivisions Height to 50. The result is that the plane is now made up of squares. Think of each square as one of the lights on the sign (see Figure 1.44). Name the plane signPlane.

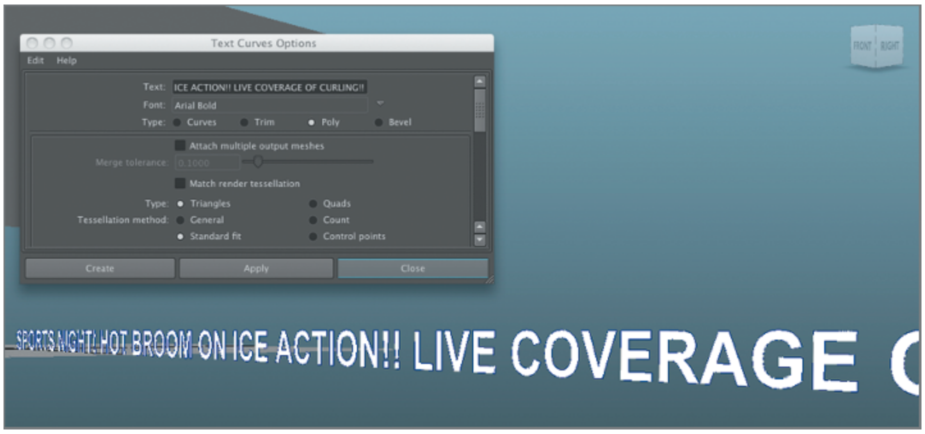


pPlane1
Translate X 11.414
Translate Y 8.741
Translate Z -0.323
Rotate X 0
Rotate Y 0
Rotate Z 90
Scale X 9
Scale Y 1
Scale Z 18
Visibility on
SHAPES
pPlaneShape1
INPUTS
polyPlane1
Width 1
Height 1
Subdivisions Width 25
Subdivisions Height 50
Create UVs Normalize...

Figure 1.44 A plane is placed beside the blimp and subdivided into squares.

4. Select the blimp group in the Outliner, and hide it so you can focus on the sign-Plane. To create the text, choose Create › Text › Options. In the Text box, enter the text in the field. You can choose an appropriately bombastic message; use something such as SPORTS NIGHT! HOT BROOM ON ICE ACTION!! LIVE COVERAGE OF CURLING! Make the text all capital letters, and choose a font that is clear and easy to read such as Arial Bold.

5. In the options, set the output to Poly. This will create flat polygon letters from NURBS curves. Create and apply a surface shader to the letter geometry. Set the color of the surface shader to white (see Figure 1.45).



Text: ICE ACTION!! LIVE COVERAGE OF CURLING!!
Font: Arial Bold
Type: ● Curves ● Trim [✓ Poly] ● Bevel
Attach multiple output meshes
Merge tolerance: 0.1000
Match render tessellation
Type: [✓ Triangles] ● Quads
Tessellation method: ● General ● Count
[✓ Standard fit] ● Control points

Figure 1.45 Create the text that the sign will display.

6. Place the text group so that in the side view it appears in front of the plane, and make sure Wireframe On Shaded is still activated. You'll use the plane as a guide for the placement of the text, keeping in mind that each square will be a pixel in the final render. Try to match the top and bottom of the letters with the polygon grid on the plane (see Figure 1.46).

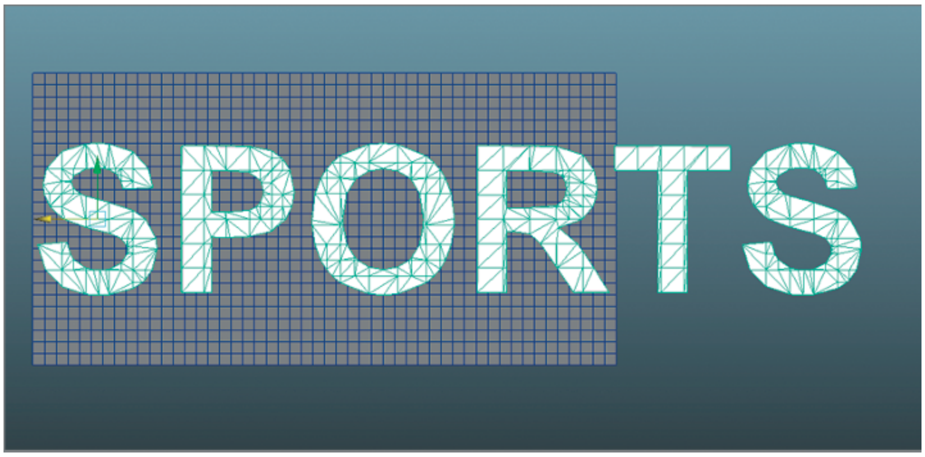


Figure 1.46 Arrange the text group in front of the plane in the side view. Scale the group so that the letters fit within the polygon grid on the plane.

7. Open the Render Settings window, and set Resolution to Width 50 and Height 25. This matches the number of polygons in the plane. In the side view, turn on the resolution gate using the icons at the top of the viewport panel. Zoom into the plane, and try to match the resolution gate so that it fits the edges of the plane as exactly as possible.
8. Select the text group, and use the Move and Scale tools to scale the letters so that they fit within the rows of squares on the plane, leaving some space at the top and bottom.
9. Save a version of the scene to your local disk.

Create an Animated Texture from the Text

The text needs to be animated in such a way so that in each frame it moves over exactly one square. This will make it look as though each light in the array is turning on or off. You don't want the squares to look as though they are half lit, which would ruin the effect of the text being created by lights. Creative use of keyframes and the Graph Editor makes this an easy thing to accomplish.

1. In the side view, select the text group, and move it so that the edge of the letters are aligned as much as possible with the vertical and horizontal lines on the plane, as shown in Figure 1.47.
2. Select the text group, and create a keyframe on the Translate Z channel.
3. Move the Timeline two frames ahead. Move the text one square to the left, and set a second keyframe on the Translate Z channel.
4. Open the Graph Editor, and select the Translate Z channel on the left side of the panel. Press F to focus on the keyframes. Select the keys, and set the tangents to stepped tangents.

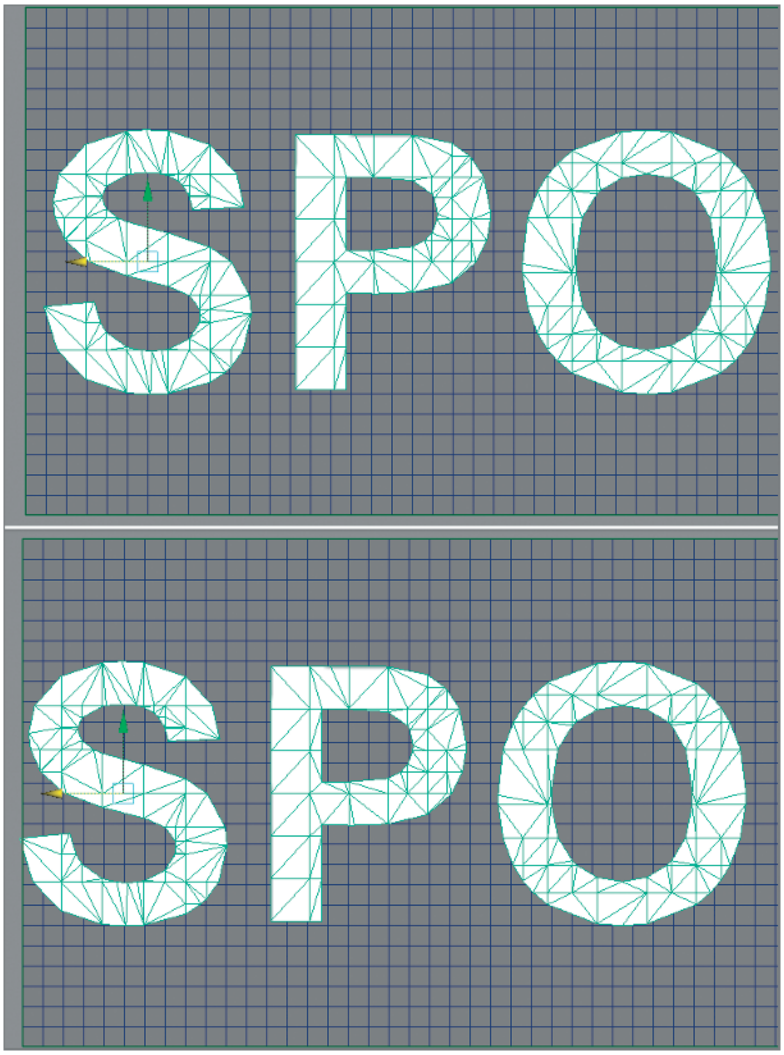


Figure 1.47 Animate the text group by moving the text group one square to the left and creating keyframes.

5. In the View menu of the Graph Editor, turn on Infinity. Drag a selection around the curve, and choose Curves › Post Infinity › Cycle With Offset (see Figure 1.48).

   The curve will become stair-stepped in shape, with the dashed line stepping upward to the right for as far as the Graph Editor displays. This means the group will move to the left one square every two frames for infinity. This creates a very staggered motion as the letters of the sign move from right to left, which suits the style of animation you would expect from an array of lights.

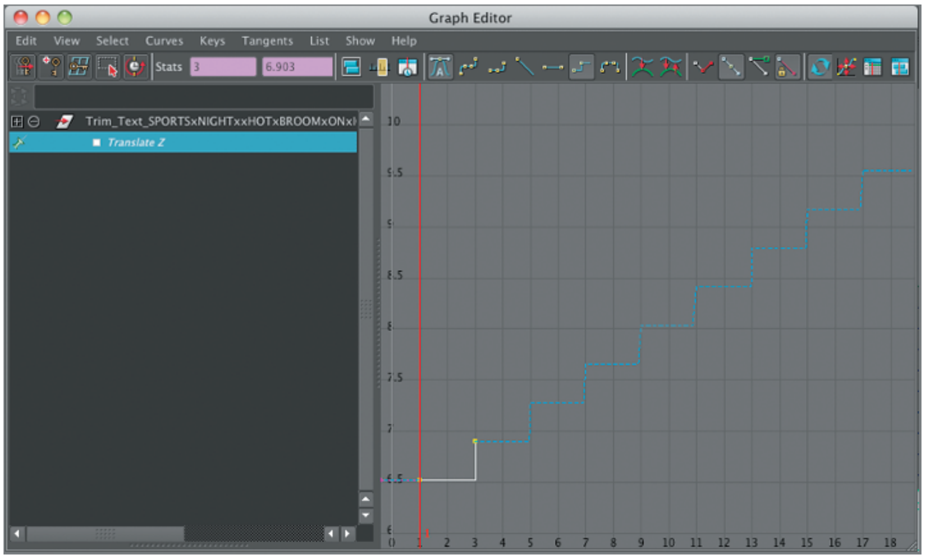

Graph Editor
Edit View Select Curves Keys Tangents List Show Help
Stats 3 6.903
Trim_Text_SPORTSxNIGHTxxHOTxBROOMxONx
Translate Z

Figure 1.48 Animate the text group by moving the text group one square to the left and creating keyframes.

To create the animated text, you'll render a sequence of 300 frames that will then be mapped as a texture to the sign geometry. The resolution of the sequence is 50 pixels wide by 25 pixels tall, which matches the number of polygons in the sign. You'll render the sequence using the Maya Hardware rendering option, which eliminates any anti-aliasing that might cause the text in the sign to become blurry. This blurriness can ruin the effect of the text being created by individual lights on the blimp. Three hundred frames may not be enough to display the entire message, but that's OK; for this demonstration, 300 frames should be plenty.

6. Open the Render Settings window, and set the Render Using menu to Maya Hardware. On the Common tab, set the following (see Figure 1.49):

   File name Prefix: blimpText

   Image Format: Maya IFF(iff)

   Frame/Animation ext.: name.#.ext

   Frame Padding: 3

   Start Frame: 1

   End Frame: 300

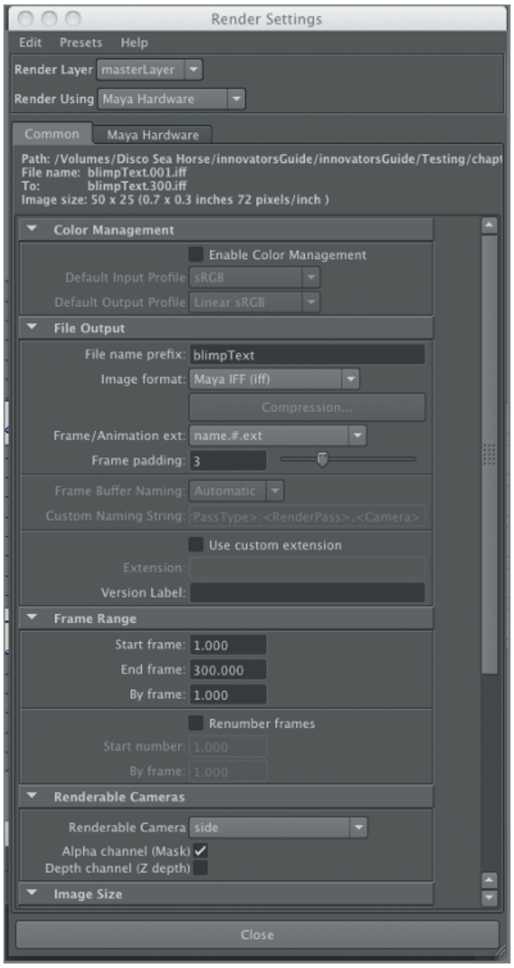
   By Frame: 1

   Renderable Camera: Side

   Width: 50

   Height: 25

7. On the Maya Hardware tab, set the Presets menu to Preview.
8. In the viewport window, select the signPlane geometry, and hide it (Ctrl+H); you want only the text to be visible in the render.


Render Settings

Edit Presets Help

Render Layer: masterLayer

Render Using: Maya Hardware

Common | Maya Hardware

Path: /Volumes/Disco Sea Horse/innovatorsGuide/innovatorsGuide/Testing/chapt
File name: blimpText.001.iff
To: blimpText.300.iff
Image size: 50 x 25 (0.7 x 0.3 inches 72 pixels/inch)

Color Management
- Enable Color Management
- Default Input Profile: sRGB
- Default Output Profile: Linear sRGB

File Output
- File name prefix: blimpText
- Image format: Maya IFF (iff)
- Compression...
- Frame/Animation ext: name.#.ext
- Frame padding: 3
- Frame Buffer Naming: Automatic
- Custom Naming String: <RenderPassType>:<RenderPass>.<Camera>
- Use custom extension
- Extension:
- Version Label:

Frame Range
- Start frame: 1.000
- End frame: 300.000
- By frame: 1.000
- Renumber frames
- Start number: 1.000
- By frame: 1.000

Renderable Cameras
- Renderable Camera: side
- Alpha channel (Mask) ✓
- Depth channel (Z depth)

Image Size

Close

Figure 1.49 Set the options in the Render Settings window.

9. Save the file to your local disk. Switch to the Rendering menu set, and choose Render › Batch Render.

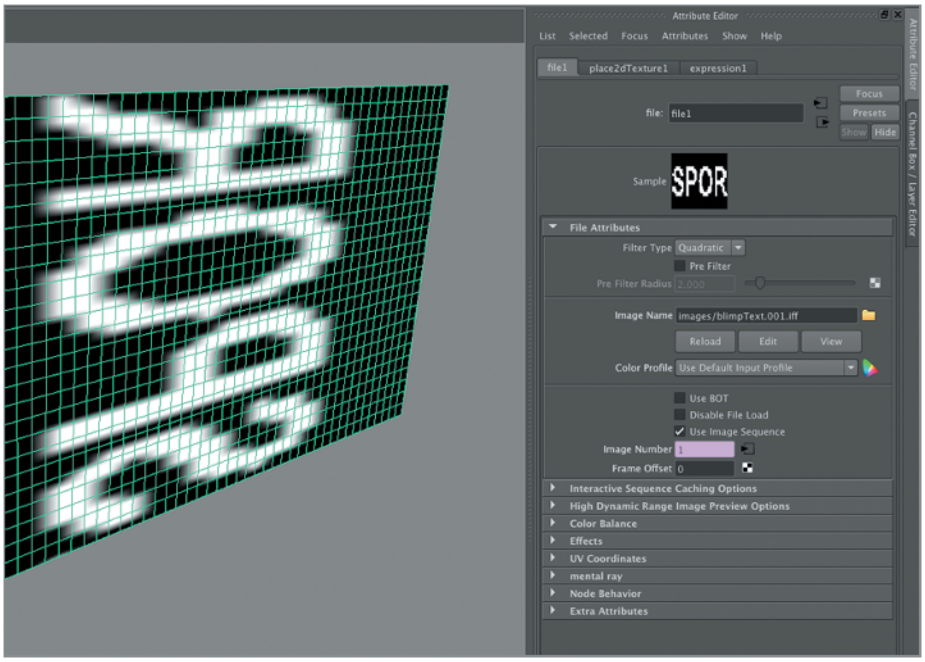
Maya goes through the process of rendering each frame in the sequence and then places the image files of the frame in the `Images` directory of the current project.

Apply the Animated Text Sequence to the Sign Geometry

Now you're ready to create the effect of the sign by applying the animated texture sequence to the sign geometry using a file texture node.

1. Select the text group, and hide it. Unhide the signPlane geometry.
2. Select the signPlane geometry. Right-click the plane in the viewport window, and choose Assign New Material from the marking menu. From the pop-up panel, choose Lambert. This creates a new Lambert shader for the plane.
3. The Attribute Editor for the Lambert shader should appear. Name the Lambert material signTextMaterial. Set the Color slider to a black color. Click the texture swatch next to Incandescence to open the Create Render Node window. From the Texture section, choose File.

4. When you create the file texture node, Maya should automatically open the Attribute Editor for the file texture node. Click the Folder icon next to Image Name, and use Maya's file browser to navigate to the `image` directory of the current project. Select the file `blimpText.001.iff`. This is the first frame of the sequence you rendered in the previous section. Turn on the Use Image Sequence options. This sets up the texture so that the animation of the text is visible on the plane.
5. In the viewport window, set the renderer to Viewport 2.0. You should see the first few letters of the text appear on the plane. There's a good chance that it may be aligned with the vertical axis of the plane and not the horizontal. This is because the UV texture coordinates of the plane are not set up to match the side view (see Figure 1.50).



file: file1

File Attributes

Filter Type: Quadratic

Pre Filter: ☐

Pre Filter Radius: 2.000

Image Name: images/blimpText.001.iff

Color Profile: Use Default Input Profile

Use BOT: ☐

Disable File Load: ☐

Use Image Sequence: ✓

Image Number: 1

Frame Offset: 0

Interactive Sequence Caching Options

High Dynamic Range Image Preview Options

Color Balance

Effects

UV Coordinates

mental ray

Node Behavior

Extra Attributes

Figure 1.50 The letters appear on the plane, but they are oriented along the wrong axis.

6. Switch to the Polygon menu set. Switch to the side view in the viewport window. Choose Create UVs › Planar Mapping › Options. In the options, set Fit Projection to Bounding Box, and set the Project From option to Camera. This creates UV texture coordinates for the plane based on the current camera view. Click Apply. The first few letters of the text should now appear in the proper orientation.
7. Play the animation. You should see the text move across the screen (remember that you need the viewport renderer to be set to Viewport 2.0 and hardware texturing should be enabled).
8. Save the file to your local disk.

Animate the Reverse Disintegration of the Sign

The final part of the effect involves the "reverse disintegration" of the sign. This means making it appear as if the lights of the sign are swirling around in the air and then come together to form the sign on the side of the blimp. Believe it or not, this last part is pretty easy compared to all the steps it takes to animate the text of the sign! All you need to do is deform the sign plane so it fits the shape of the blimp, separate the sign into individual polygons, and then use nCloth to animate the dynamics.

1. Continue with the scene from the previous section. You can select and delete the text groups because you don't need them anymore. Select and unhide the blimp.
2. There are a number of ways to make the sign fit the side of the blimp. To keep things simple, you can use two Bend deformers to shape the sign so that it matches the curvature of the blimp. Select the signPlane geometry, and move it next to the blimp. Switch to the Animation menu set. With the sign selected, choose Create Deformers › Nonlinear › Bend. Do this twice to create two Bend deformers.
3. Use the following settings for the Bend deformers. You may need to tweak these values to get a more precise fit in your scene. It doesn't have to be absolutely perfect; it just has to be close enough so that the effect is convincing from a moderate distance (see Figure 1.51).

   Bend1Handle:

   Translate X: 8.827

   Translate Y: 9.478

   Translate Z: -0.323

   Rotate Z: 168.11

   Scale X, Y, and Z: 9

   Bend1:

   Curvature: 1

   Bend2Handle:

   Translate X: 7.647

   Translate Y: 8.875

   Translate Z: -0.323

   Rotate X: 90

   Rotate Z: 180

   Scale X, Y, and Z: 9

   Bend2:

   Curvature: 0.2
4. Adjust the position of the sign geometry until you're satisfied that it looks good. Select signPlane, and choose Edit › Delete By Type › History. This freezes the geometry to its deformed shape as well as removes the bend shape nodes.

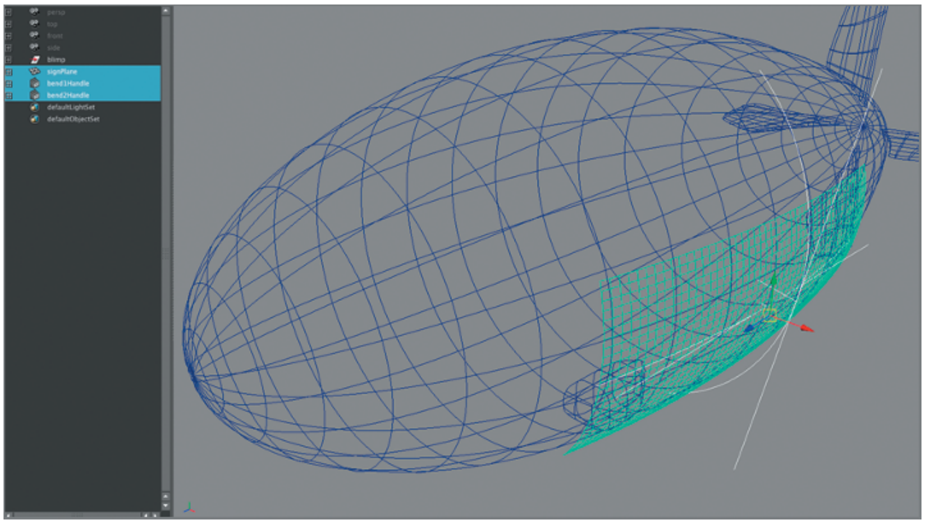


Figure 1.51 Use a pair of Bend deformers to shape the sign so that it fits the side of the blimp.

5. Parent signPlane to the Blimp group. Select signPlane from within the Blimp group, and choose Modify › FreezeTransformations. This establishes the current position, orientation, and scale of signPlane as the default values.
6. Animate the blimp moving slowly forward along the Z axis over the course of 300 frames. When you play through the animation, you should see the text on the side (see Figure 1.52).

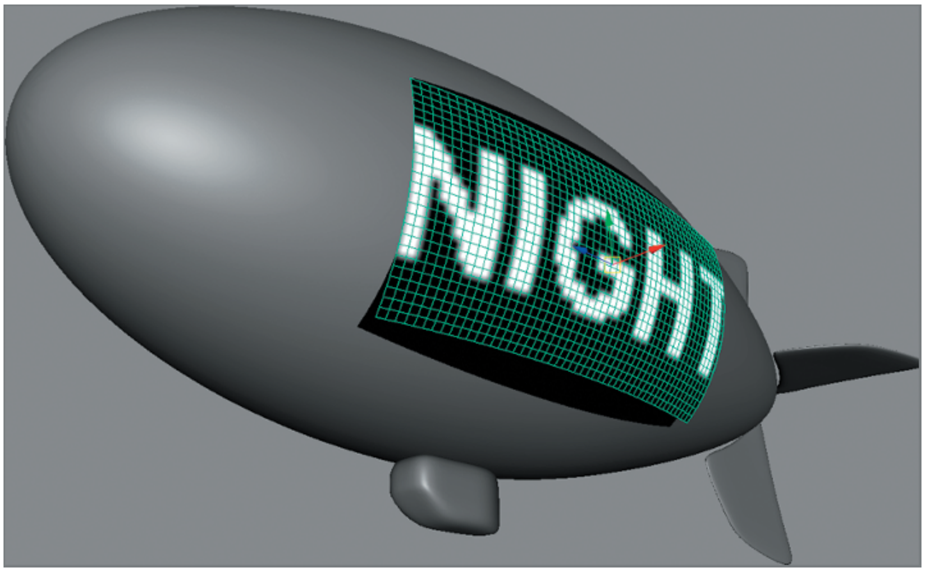


Figure 1.52 The text on the sign animates as the blimp moves forward in space.

Now to make the sign look more like lights, you'll break the sign into its individual polygons, scale the polygons down a little, and then convert the polygons into an nCloth object.

7. In the Outliner, expand the Blimp group, and hide the Fins, blimpBody, and Gondola nodes so that just the sign is visible. Rewind the animation to frame 1.
8. Switch to the Polygon menu set. Right-click the signPlane, and choose Edges to switch to Edge Selection mode. Drag a selection over the entire plane so that all of the edges are selected.
9. Choose Edit Mesh › DetachComponent. It may not look like much has changed, but now each poly has been separated from its neighbor (see Figure 1.53).

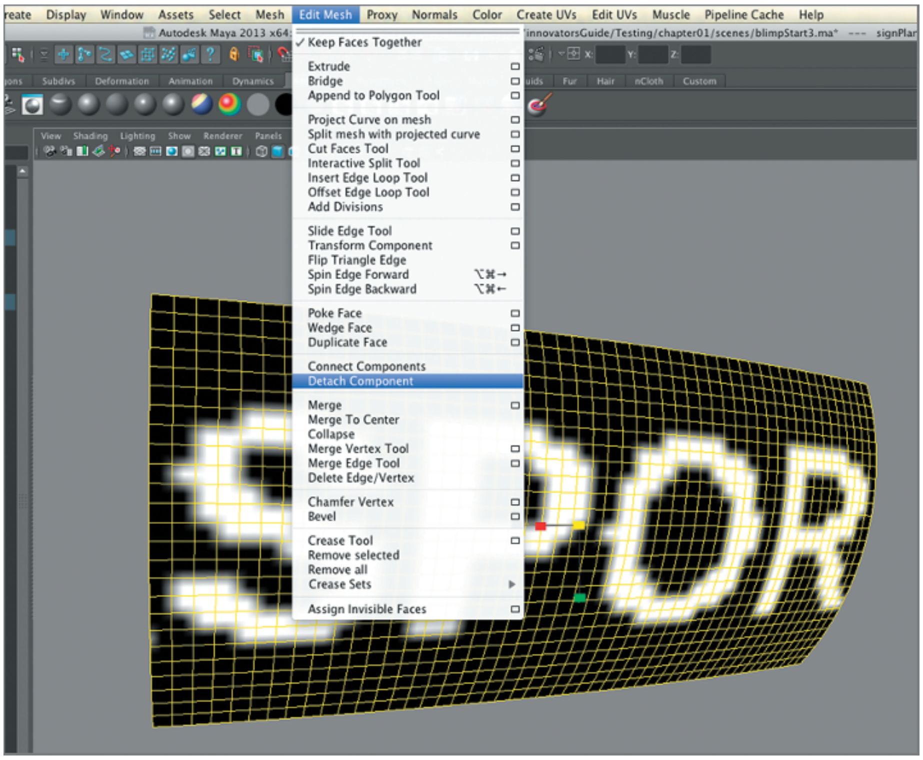


Figure 1.53 Use Detach Component to split the polygon surface into individual planes.

10. In the Outliner, select the signPlane. Choose Mesh › Separate. This makes each separated piece into its own surface with its own transform node. Each surface is placed in a group called signPlane.
11. In the viewport, drag a selection over all the polygons in the sign. Choose Modify › Center Pivot. This places the pivot point of each polygon plane at its center.
12. With all of the planes still selected, open the Channel Box, and set Scale X, Scale Y, and Scale Z to 0.7. This shrinks each of the polygon planes, creating space between them. Since the UV texture coordinates have been carried over from the original plane, the text should still appear to spell letters.
13. With all of the planes still selected, choose Mesh › Combine. This merges the separate pieces back into a single surface. Choose Edit › Delete By Type › History to remove construction history. The construction history is no

longer needed at this point and can significantly reduce playback performance when the model is converted into an nCloth.

14. The merged surface has been renamed and moved out of the blimp group. Name it signLights, and in the Outliner move signLights back into the Blimp group (see Figure 1.54).

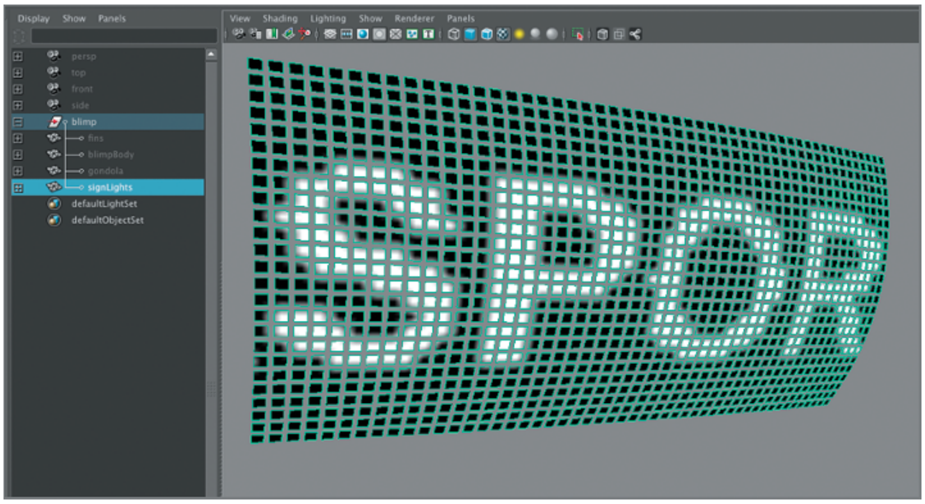


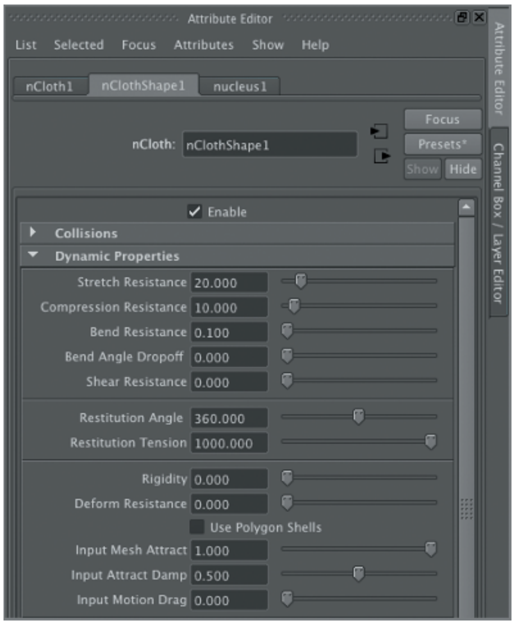
Figure 1.54 The result of these operations creates space between each of the polygons in the plane.

15. Save the file to your local disk.

Create the Dynamic Effects

To make the individual lights of the blimp fly around, you can convert the sign into an nCloth and then apply a turbulence field. By animating the Input Mesh Attract value on the nCloth node, you can animate the lights of the sign falling into place on the blimp.

1. Continue with the scene from the previous section. In the Outliner, expand the Blimp group and unhide the Fins, blimpBody, and Gondola nodes.
2. Switch to the nDynamics menu set. Select the signLights geometry, and choose nMesh › Create nCloth. This converts the polygon plane into a cloth object, which can respond to dynamic forces such as fields.
3. The new nCloth object may lose the connection to the signTextMaterial. If this happens, open the Hypershade, and MMB drag the signTextMaterial shader from the Materials tab onto the one of the polygons of the signLights object in the viewport. This reapplies the shader to the surface so that the text reappears.
4. Select the nCloth1 node in the Outliner, and open its Attribute Editor. Find the Input Mesh Attract slider under Dynamic Properties on the nClothShape1 tab of the Attribute Editor. Set the value of the slider to 1 (see Figure 1.55).



Stretch Resistance	20.000
Compression Resistance	10.000
Bend Resistance	0.100
Bend Angle Dropoff	0.000
Shear Resistance	0.000
Restitution Angle	360.000
Restitution Tension	1000.000
Rigidity	0.000
Deform Resistance	0.000
Use Polygon Shells	☐
Input Mesh Attract	1.000
Input Attract Damp	0.500
Input Motion Drag	0.000

Figure 1.55 Set the Input Mesh Attract slider to 1 in the nCloth Attribute Editor.

5. Rewind and play the scene. Playback will be slower since Maya has to calculate the dynamics on each of the polygons of the sign for every frame, but the sign should still move along with the blimp. This is because the original shape of the surface (aka the input mesh) is attracting each point of the nCloth at 100 percent strength. You can't see the input mesh, but it is there moving along with the blimp, and it's keeping the nCloth surface from falling away.
6. Select the signLights object, and choose Fields › Volume Axis. This creates a three-dimensional field that can be used to create turbulence and other types of motion. The field may not be visible in the viewport; switch the renderer to High Quality Rendering so you can see the field.
7. Scale the Volume Axis field up so that it encompasses the blimp as well as the space the blimp covers as it floats through the air (see Figure 1.56).
8. Open the Attribute Editor for the Volume Axis field. On the VolumeAxisField1 tab, set the following:

   Volume Axis Field › Magnitude: 5

   Volume Speed › Away From Axis: 0

   Volume Speed › Around Axis: 1

   Volume Speed › Turbulence: 25

   Volume Control › Trap Inside: 1

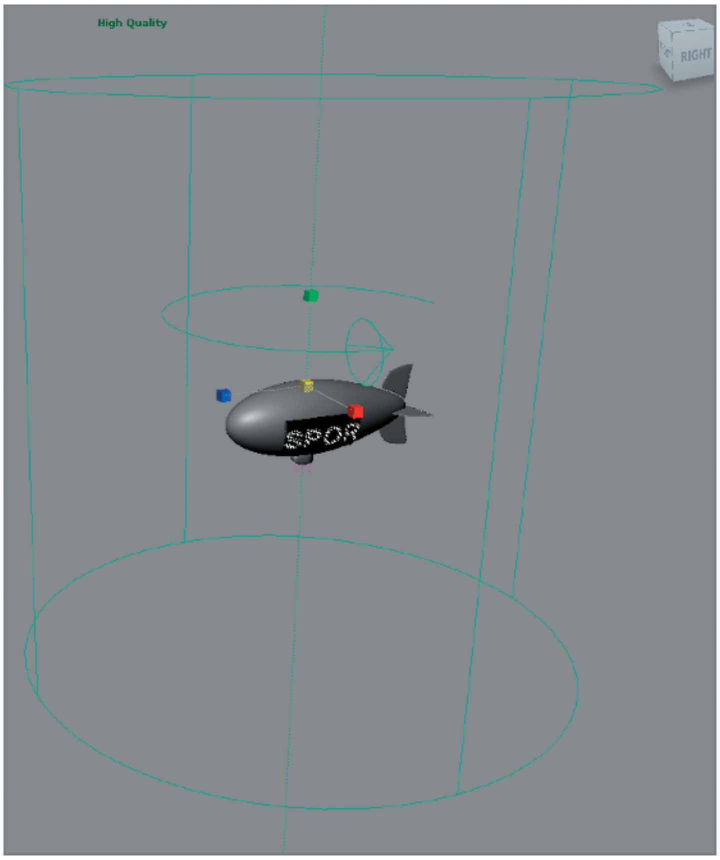


Figure 1.56 Add a Volume Axis Field to the scene.

9. Select nCloth1, and open its Attribute Editor. Set the Input Mesh Attract value to 0. Rewind and play the animation.

   As the animation plays, the polygons of the sign fly away from the blimp and break apart in the air. This is because the input mesh attract is now at 0, so the input mesh has no effect on the polygons of the sign, whereas the Volume Axis field is now able to move each polygon of the nCloth around. Since the polygons are not connected to each other, they behave just like particles, flying around within the field as if they were independent pieces.

10. Play the animation to around frame 20. The polygons should be blowing around the blimp. You want to start the simulation when the polygons are farther away from the blimp. Select the signLights node, and choose nSolver › Initial State › Set From Current (see Figure 1.57).

11. Rewind the animation; the polygons of the sign should still be away from the blimp in a jumbled mess. Open the Attribute Editor for the nCloth1 node. Find the slider for Input Mesh Attract, and make sure it is at 0. Right-click this field, and choose Set Keyframe (see Figure 1.58).

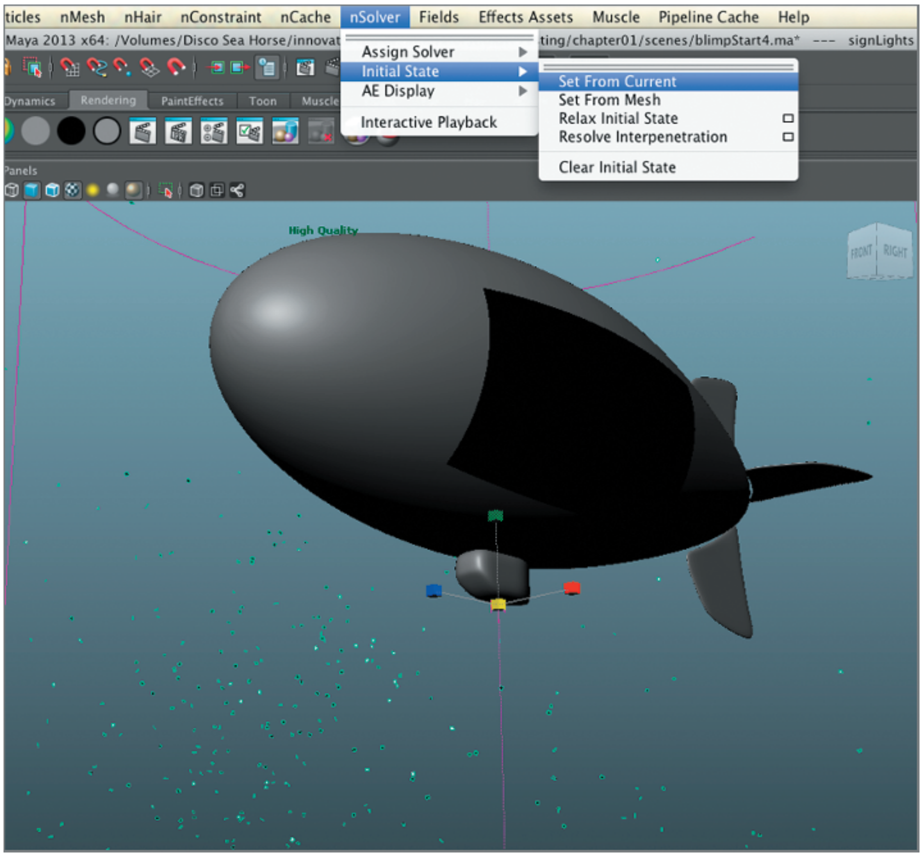

ticles nMesh nHair nConstraint nCache nSolver Fields Effects Assets Muscle Pipeline Cache Help
Maya 2013 x64: /Volumes/Disco Sea Horse/innovat...ting/chapter01/scenes/blimpStart4.ma* --- signLights
Assign Solver
Initial State
AE Display
Interactive Playback
Set From Current
Set From Mesh
Relax Initial State
Resolve Interpenetration
Clear Initial State
Dynamics Rendering PaintEffects Toon Muscle
Panels
High Quality
FRONT RIGHT

Figure 1.57 Set the initial state of the nCloth surface while the polygons have been blown away from the mesh.

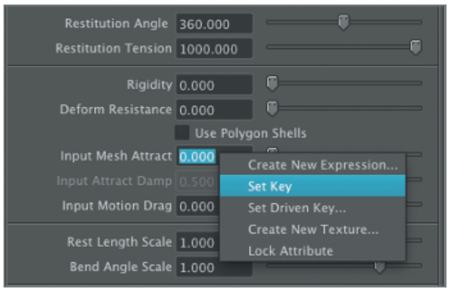

Restitution Angle 360.000
Restitution Tension 1000.000
Rigidity 0.000
Deform Resistance 0.000
Use Polygon Shells
Input Mesh Attract 0.000
Input Attract Damp 0.500
Input Motion Drag 0.000
Rest Length Scale 1.000
Bend Angle Scale 1.000

Figure 1.58 Keyframe the Input Mesh Attract value.

12. Move the animation to frame 150, and then set Input Mesh Attract to 1. Set another keyframe.

13. Rewind and play the animation. The polygons of the sign fly around chaotically and then come together to form the sign on the side of the blimp. This is because as the Input Mesh Attract strength rises, each polygon is attracted back to its initial spot on the sign (see Figure 1.59).

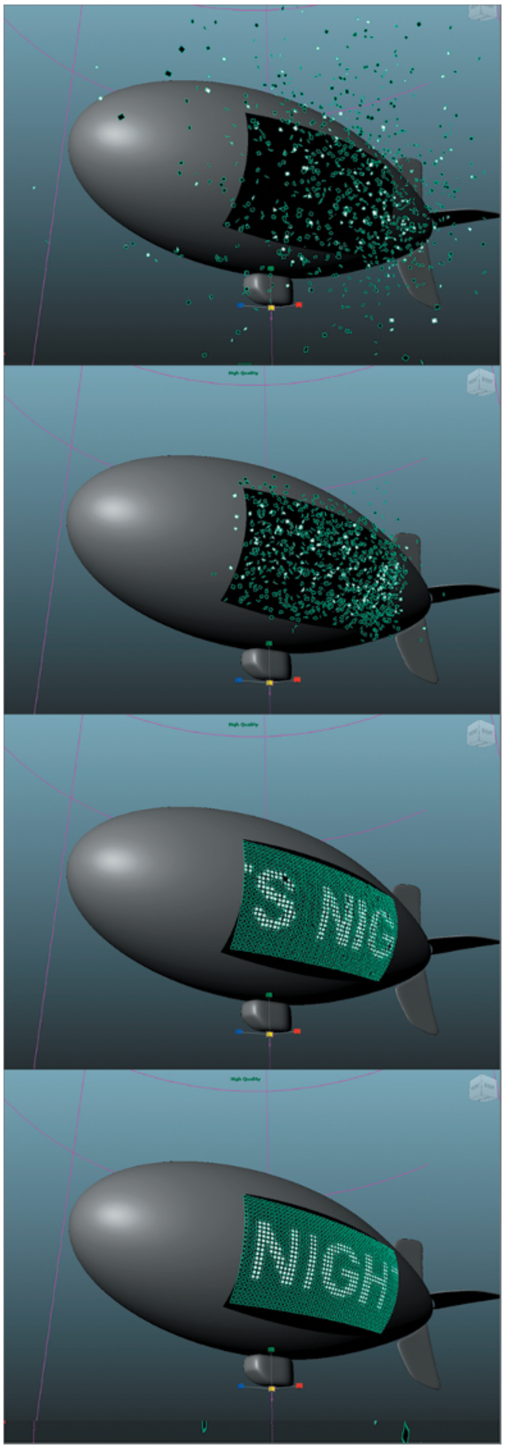


Figure 1.59 The final effect shows the sign being formed on the side of the blimp.

14. Create a playblast of the animation to see the effect in real time. You can find a completed version of the scene in the `scenes` folder of the `Chapter01_project` directory. The file is named `blimpEnd.ma`.

The basic effect is now complete. From this point on, you can experiment with the way in which the sign forms on the side of the blimp by editing the dynamic properties of the nCloth node, the properties of the Volume Axis field, and the animation of the Input Mesh Attract values. Try using different types of fields for the dynamic effect.

Further Study

See whether you can create a similar effect as the one shown in this exercise, but this time try animating text on a light sign that wraps around the front of a building. See whether you can use nCloth to make the text fly away from the building sign and down a street. Motion graphics effects like these are often in high demand in the world of commercials.

Do It with MEL

A great idea for a simple script would be to automate the process of splitting the polygon plane into individual pieces, scaling the pieces down, and then recombining the pieces into a single surface. Max shows you how this can be done.

This script will take a polygon object, break it into individual faces, and convert it to an nCloth object. I'm using a technique that separates the polygon faces, scales them down, and then recombines them, just like in the exercise. However, this isn't the only possible approach. Try this script and then take a look at all three versions of the script, which you can find in the `scripts` directory of the `Chapter01_project` directory. These scripts are named `confettiMachineV1.mel`, `confettiMachineV2.mel`, and `confettiMachineV3.mel`.

1. To start the script, you need to store your selected object in a string variable so that you can access it again later in the script. You know you want to use a string variable because you are storing the name of your polygon object. This approach means that the names of the objects are not hard-coded into the script, so you can use the same script again and again on different objects. This first line looks at the currently selected object and puts its name into a variable.

```
string $sel[]=`ls -sl`;
```

2. Separate the polygons of the selected object along each edge. First use the `ConvertSelectionToEdges` command to convert your object to edges.

```
ConvertSelectionToEdges;
```

3. Use the `DetachComponent` command to separate each face into its own shell.

```
DetachComponent;
```

4. Use the `select` command to select the original mesh.

```
select -r $sel ;
```

Continues

Do It with MEL *(Continued)*

5. Use the `polySeparate` command to break each face into its own `polyShape`.

   ```
   polySeparate;
   ```

6. Center the pivot of each mesh so it creates a visible distance between each face.

   ```
   CenterPivot;
   ```

7. Using the `scale` command with the `-scaleXYZ` flag, scale each mesh down 10 percent, or .9 .9 .9 in all three axes.

   ```
   scale -scaleXYZ .9 .9 .9;
   ```

8. The `polyUnite` command will recombine the meshes into a single object (shape node).

   ```
   polyUnite;
   ```

9. Delete the history to remove any unwanted nodes or dead transforms. The `delete` command with the `-ch` (construction history) flag will remove the history from your object.

   ```
   delete -ch;
   ```

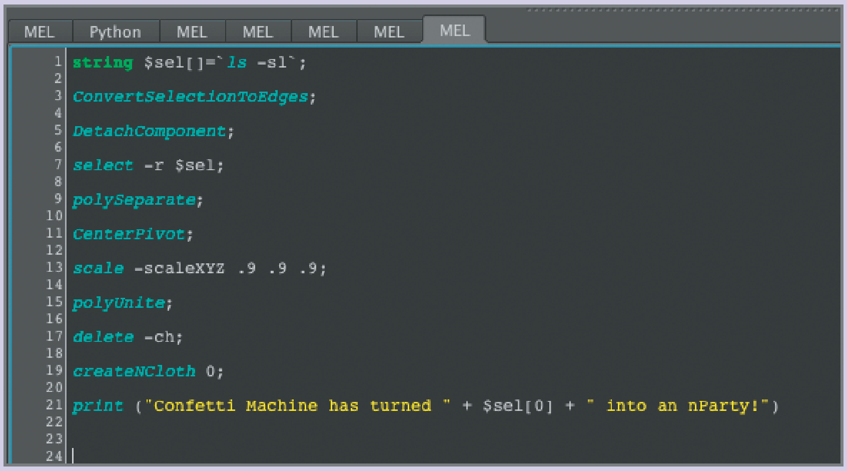
10. The `createNCloth` command turns the polygon mesh into an nCloth object. The argument `0` means local space, and `1` is world space.

    ```
    createNCloth 0;
    ```

11. Finally, it's a good idea to add a command that prints a message in the Script Editor that tells the user that the script ran properly.

    ```
    print ("Confetti Machine has turned " + $sel[0] + " into an
    nParty!");
    ```

The following image shows the script as it appears in the Script Editor:



```
string $sel[]=`ls -sl`;

ConvertSelectionToEdges;

DetachComponent;

select -r $sel;

polySeparate;

CenterPivot;

scale -scaleXYZ .9 .9 .9;

polyUnite;

delete -ch;

createNCloth 0;

print ("Confetti Machine has turned " + $sel[0] + " into an nParty!")


```

Do It with MEL *(Continued)*

You can use this script in many different scenes as a way to blow apart objects for cool disintegration effects. You can turn this script into a shelf button so that all you need to do is select a polygon object and then click the button on the shelf. To turn a script into a shelf button, switch to the Custom shelf, select the text of the script in the Script Editor, and choose File › Save Script To Shelf. Give the shelf button a short name, and remember to click the downward-facing arrow button to the left of the shelf and choose Save All Shelves. The button will appear on the Custom shelf, ready to use, each time you launch Maya.