Masters Project

Technical Animation Using Maya & Mental Ray

Carrie Sullivan

MSc Computer Animation

NCCA. Bournemouth University

8th September 2006

Table of Contents

Introduction

This project is a continuation of the Major Animation project in which an animation involving a large number of spiders was produced. Creating an animation that requires a large number of moving creatures is difficult and time consuming to animate manually. Procedural techniques were therefore developed to animate the spiders systematically.

The simulation of the spiders was achieved using a combination of Maya's *[5]* Dynamics system and Maya's *[5]* Embedded Language (MEL). Once the behaviour of the spiders had been simulated, the spider motion generated was baked so that the results could be converted to keyframes, allowing for the spider behaviour to be edited if required. In order to provide optimum control, a tool developed using MEL was used to create motion paths from the animation by converting the motion into curves. These curves were then used as motion paths for the spiders. Not only did this allow the motion of the spiders to be easily manipulated by selecting and translating the control vertices of the curves, but also allowed for the spiders' acceleration to be measured along the path, so that the speed of the spiders' legs could be controlled accordingly.

At the end of the Major Animation project, whilst an animation had been successfully produced, a number of further improvements were required. The Major Animation project's main focus was on the technical aspects of the spider simulation and as a result, the animation had been shaded and rendered using only very basic techniques. For the Masters Project, a production pipeline for the rendering of the animation was required in order to allow high quality rendering using several different passes.

As well as improving the rendering, several other issues were outstanding at the end of the Major Animation project, including enhancements to the spider walk cycles and improvements to the curtain motion and collisions with the radiator. As well as addressing these issues, it was proposed that processes to allow secondary motion to be applied to the spider animation where appropriate would be incorporated into the Masters project, along with improvements to the shading, lighting, shadowing and texturing of the objects in the scene.

Previous Work

In the paper by Anjyo and Faloutsos [1], methods are discussed in which 3D motions of insects and arachnids are captured in order to drive computer generated models. Traditional motion capture of creatures this small is not possible, so Anjyo and Faloutsos [1] use specialist equipment and dynamic programming [6] techniques to extrapolate the information. Their method utilises video footage of the insect or arachnid from three cameras and a user driven labelling process that takes advantage of tracking techniques and a 3D point generating algorithm to capture specific points on the subject.

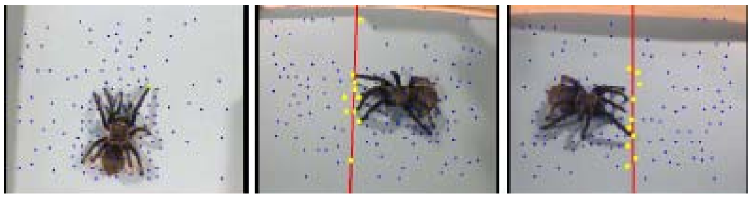


Figure 1: three views of a spider with tracking data [1]

Due to the difficulty in obtaining usable video footage of creatures as small and fast as ants and arachnids, a motion synthesis process that extracts and extends the motion information was also used. This process generates new motion programmatically and this is added to the video footage to give the characteristic motion sequences for each creature. The final motion can then be applied to multiple similar subjects.

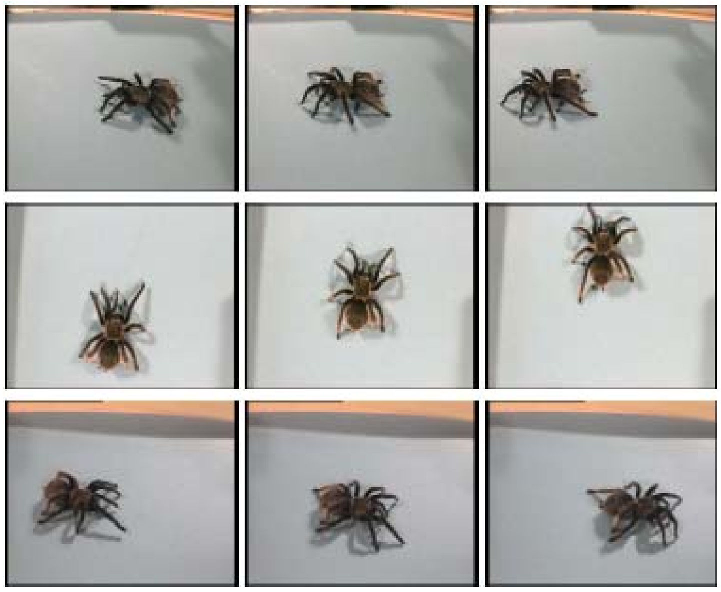


Figure 2: the start, middle and end frames from a 120 frame sequence for three views of a spider [1]

In the final stage, Anjyo and Faloutsos [1] utilise an algorithm that enables path following by finding the optimal path for multiple simulated creatures. Whole groups of arachnids or insects all moving in a coherent and natural manner can be generated using this approach.

ap Cenydd and Teahan's [2] work also addresses the animation of arachnids, in attempting to simulate the motion of such creatures across arbitrary surfaces. Their solution begins with a behavioural system similar to Reynolds' flocking algorithm [7] that generates the wandering behaviour of the spiders, with the goal of the spider being to randomly explore its environment.

Once the steering forces have been determined, they are passed into a locomotion system that uses forward Euler integration to update the spider's position. At each step in the integration, the steering force is applied to the spider's mass to generate acceleration. The acceleration is then added to the previous velocity of the spider to calculate a new velocity and the new velocity is added to the spider's previous position in order to calculate the spider's new position.

The spider's orientation is then calculated using a map of the environment that is generated during the initial stages of the simulation to store the edge connectivity of each polygon in the scene. For a determined number of steps, the spider then uses a polygon-ray intersection test to determine which polygon it is currently standing on. Using the environment map, the test will be carried out against the polygons which immediately surround the last known polygon. Once the intersected polygon is located, the spider's up vector is set to interpolate between its existing up vector and the inverse of the polygon's normal vector.

A similar orientation process to that of ap Cenydd and Teahan *[2]* was used during the Major Animation project using Maya's *[5]* rigid body dynamics system, where the spiders' orientation was calculated to match the direction it was facing. In the case where the desired orientation was more or less than a specified amount, the spider was slowly rotated towards the desired orientation in order to prevent erratic changes to the simulation.

The final part of ap Cenydd and Teahan's *[2]* process involves a procedural gait generator and an inverse kinetics solver. The gait generator is the section most of interest for this Masters project and is discussed in more detail in the Technical Background section of this document.

Technical Background

Procedural Animation

Procedural animation allows objects to be animated using a set of rules and procedures to create convincing results from relatively little input. Generally the rules are based on real world physics, expressed mathematically. Common uses for procedural animation include cloud, smoke, gas and fire effects using particle systems, fur and hair dynamics, fluid dynamics, cloth dynamics and rigid body dynamics. Procedural character animation usually centres on flocking [1] or crowd behaviours, but as in this project, can also be used to control motion and generate walk cycles. Some such methods are discussed in the paper by ap Cenydd and Teahan [2].

Ambient Occlusion

In order to produce a global illumination type effect without sacrificing large amounts of render time, it was decided to apply an ambient occlusion shader to the objects in the scene.

Ambient Occlusion is based on how much ambient environment light a surface receives. Surfaces occluded by other objects receive less light than those fully exposed to the environment and this is reflected in the ambient occlusion which darkens surfaces only partially visible to the environment. This results in the kind of soft shadowing normally associated with global illumination and other more complex indirect lighting techniques.

Ambient occlusion is achieved by casting rays in a hemisphere around the surface normal. The final occlusion amount is determined by the number of rays that hit other surfaces or objects in the scene. As the strongest contribution comes from the general direction of the surface normal, the results are weighted in favour of samples cast in that direction. In addition to calculating the direction of most occlusion, the direction of least occlusion, also known as the bent normal, is obtained and used in place of the surface normal to obtain more accurate environment lighting. The equation for approximating the amount of shadow for the surface is as follows:

$$1 - \frac{r \cos\theta_E \max\left(1,\ 4\cos\theta_R\right)}{\sqrt{\frac{A}{\pi} + r^2}}.$$

Equation 1: shadow approximation [8]

Which equates to one minus the amount that the other objects (known as the emitters) shadow the object that is being shadowed (also known as the receiver). In the above equation, A represents the area of the emitter, θ_E is the angle between the emitter's normal and the vector from the emitter to the receiver and θ_R is the corresponding receiver's angle. A graphical representation of this equation is shown in Figure 3.

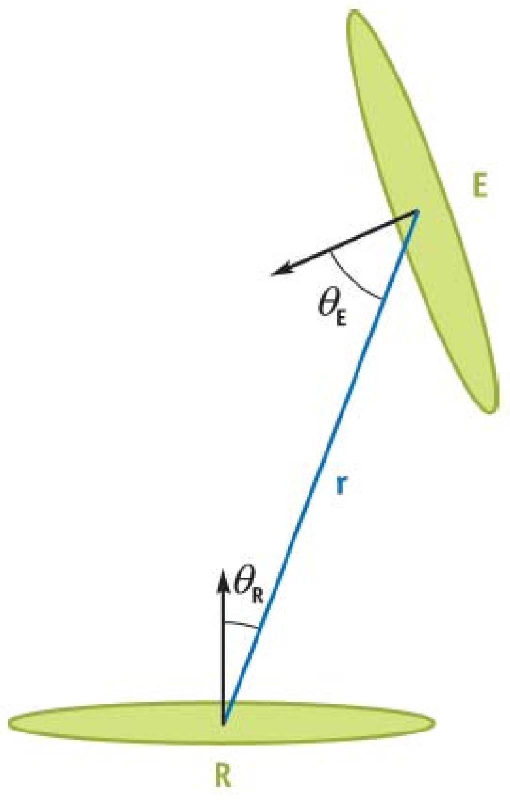


Figure 3: Diagram demonstrating how the receiver object (R) receives light or shadow from the emitter object (E), with the distance between the centres of the two objects being represented in the line r [8].

Implementation

Animation Tools & Techniques

Procedural Animation

The procedural animation of the spider walk cycles for the Major Animation project was largely produced using trigonometric functions. Whilst this method was suitable for distant spiders, the motion tended to lack weight when viewed up close and was far too regular to match the idiosyncrasies of natural spider movement. As a result, the procedural animation of the spiders was reviewed and improved for the Masters project.

In order to add weight to the original walk cycle, the legs needed to be planted on the ground for a longer period of time. Initially, a sine curve was used to control the translate y attribute of the spider's legs. To prevent the feet going through the floor, the translate y attribute is limited to positive values only.

As the legs on the spider rig are parented to the spider's body, the translate x attribute of the legs is also offset by the amount the spider's body is moving relative to the legs during the period that the translate y value is zero. This ensures that the legs are stationary for the period that the leg is on the ground, preventing an unnatural sliding motion that would have existed if the legs were allowed to move with the body. By combining the algorithms applied to the translate x and translate y attributes of the spider's legs, the pattern shown in Figure 4 is achieved.

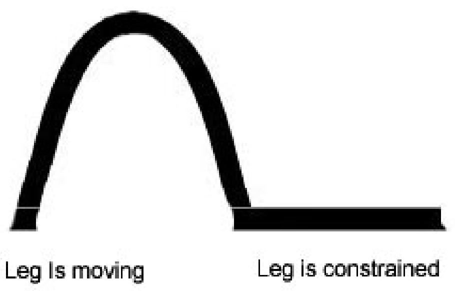


Figure 4: Diagram demonstrating how the sine function can be used to control the spider's walk cycle

Given the basic equation y = sin(x), y will equal zero when x = 0, 180 and 360 degrees. By increasing the step size (or translate x value) by 2 units and multiplying the y translation by 90 degrees, a fairly realistic timing can be achieved for a spider to raise and lower its legs. During the period that the foot is on the ground, the translate x attribute of the leg is moved in the opposite direction to the body, to secure the foot in its current position. The modulus function is used to determine at which point the foot should be stationary (as the translation of the body approaches multiples of 4, the leg movement slows and as it gets further away from 4, the movement increases). This produces a characteristic stepping motion where the leg is either accelerating with the upwards stroke, decelerating as the foot nears the ground or remaining at a constant rate whilst the foot is constrained to the floor.

The stride length is set to a maximum of 2 units and because the absolute value of the equation is obtained, the movement will always vary between 0 and 2 along the positive X axis, thus driving the leg forward. As each leg needs to be moving at different intervals, offsetting the legs by the speed that the body is moving is not enough. Therefore, a further amount is introduced to ensure each leg is not stepping at exactly the same rate. The timing of which was roughly based on ap Cenydd and Teahan's *[2]* findings where they determined that in a tetrapod walk cycle, legs on opposing sides move at similar times in a wave-like succession approximately 180 degrees out of phase with each other, as shown in Figure 5.

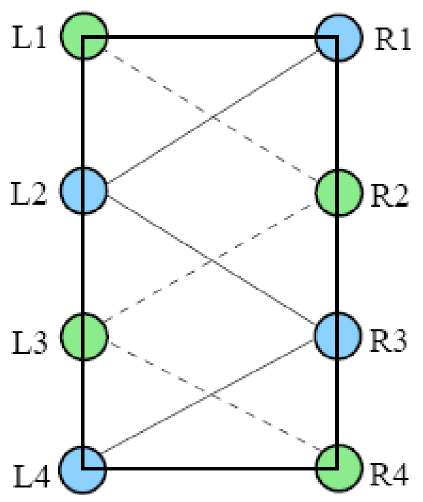


Figure 5: a typical arachnid alternating tetrapod gait [2]

Whilst the timing of the legs gave good results, the problem with using a sine function to generate a walk cycle is that it produces such a smooth curve that the acceleration and deceleration of the spider's legs as they lift off or return to the ground are equal. This results in a walk cycle that is too "robotic" in appearance and nothing like natural spider movement. In particular, the deceleration is too rapid and results in a walk that is much too weighted.

Several approaches were available to resolve the problem, including adding noise to the existing function. However getting the balance between too much or too little noise proved extremely difficult and it suffered the same inherent problem as the sine curve in achieving a natural looking walk, in that the resulting movement seemed obviously computer generated.

In the paper by ap Cenydd and Teahan [2] a set of equations were used to produce an animation curve for the spiders legs using a differential-based inverse kinetics system, an example of which is shown in Figure 6.

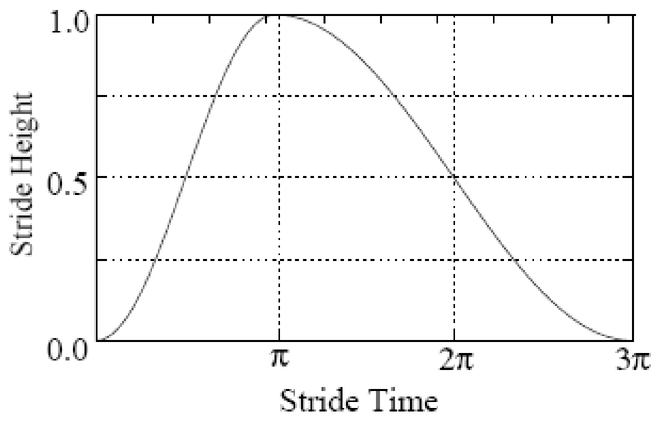


Figure 6: Example step path achieved by ap Cenydd and Teahan's [2] simulation

ap Cenydd and Teahan's *[2]* algorithms are based on the behavioural state of the arachnid and the interpolation as it changes speed, orientation or intent. The methods developed by ap Cenydd and Teahan *[2]* are computationally expensive and are used to simulate just a single spider in real time. Therefore, their method could not be replicated in this project using the existing MEL code, owing to the number of spiders being animated.

In order to emulate the functionality without the computational expense of ap Cenydd and Teahan's *[2]* method, an animation curve was created for the y translation of each of the spider's feet (an example is shown in Figure 7).

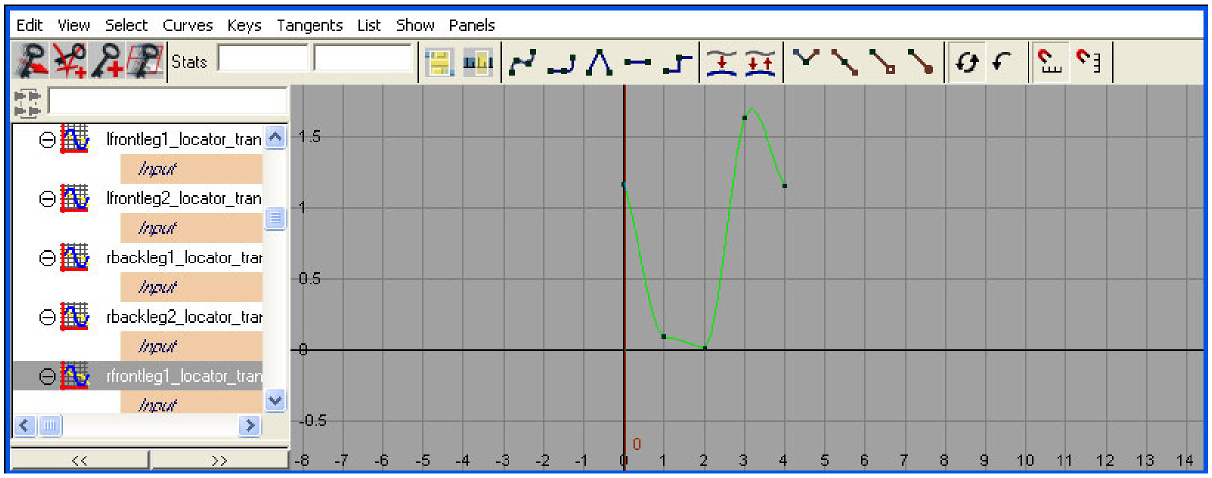


Figure 7: Graph Editor view of animation curve used as input to procedural walk cycle

This animation curve was then connected to the time attribute in Maya [5], so that it could be used to drive the walk cycle in the same way as was originally performed using the sine function. The benefit of this method over the sine method is that the curve can be adjusted as appropriate by utilising the Graph Editor in Maya [5] until the desired motion is obtained. It also means that irregularities can be introduced between different legs, being much more akin to how they would exist in nature.

Whilst it was initially more difficult to get the correct balance of acceleration and deceleration using this method, once this had been achieved, a much more natural motion for the spiders was obtained.

Animation Across Uneven Surfaces

For the parts of the animation where the spiders need to walk across uneven surfaces, a method needed to be devised to ensure that the legs did not penetrate the surface whilst they were doing so. This was achieved by using geometry constraints in Maya to restrict the entire spider to the object's surface and then offsetting the spider's body by the height of its legs. In order for the spider to be oriented to the surface of the object, a normal constraint was also used.

For the scene involving a spider walking across a teddy bear, a more complex method was required. For this scene, there is a gap between a desk and the bear, so a method to constrain the spider to the two surfaces, whilst still allowing it to cross the gap was required. Two locators were created and constrained to the two surfaces and the spider was then point and orient constrained to both of them for differing levels of influence. The first locator was attached to the desk and animated along a motion path heading towards the bear. The second locator was attached to the surface of the bear via a geometry constraint and animated across the bear's head.

At the beginning of the scene, the weighting was heavier for the first locator, so that the spider followed along its path. Once the spider neared the second locator, the influence of the second locator became stronger until the spider reached the top of the bear's head. From this point, the influence of the first locator dropped away, enabling the spider to follow the second locator over the bear's head. The influence of both locators had to be carefully controlled, to get a seamless switch between the two and also to allow the spider to cross the gap between the desk and the bear.

Once the motion obtained using this method was correct, a tool developed in MEL during the Major Animation project was used to convert the animation of the spider to a curve in order to generate a motion path for the spider. The spider could then be animated along the motion path with greater control and the walk cycle expression applied to achieve the correct leg motion.

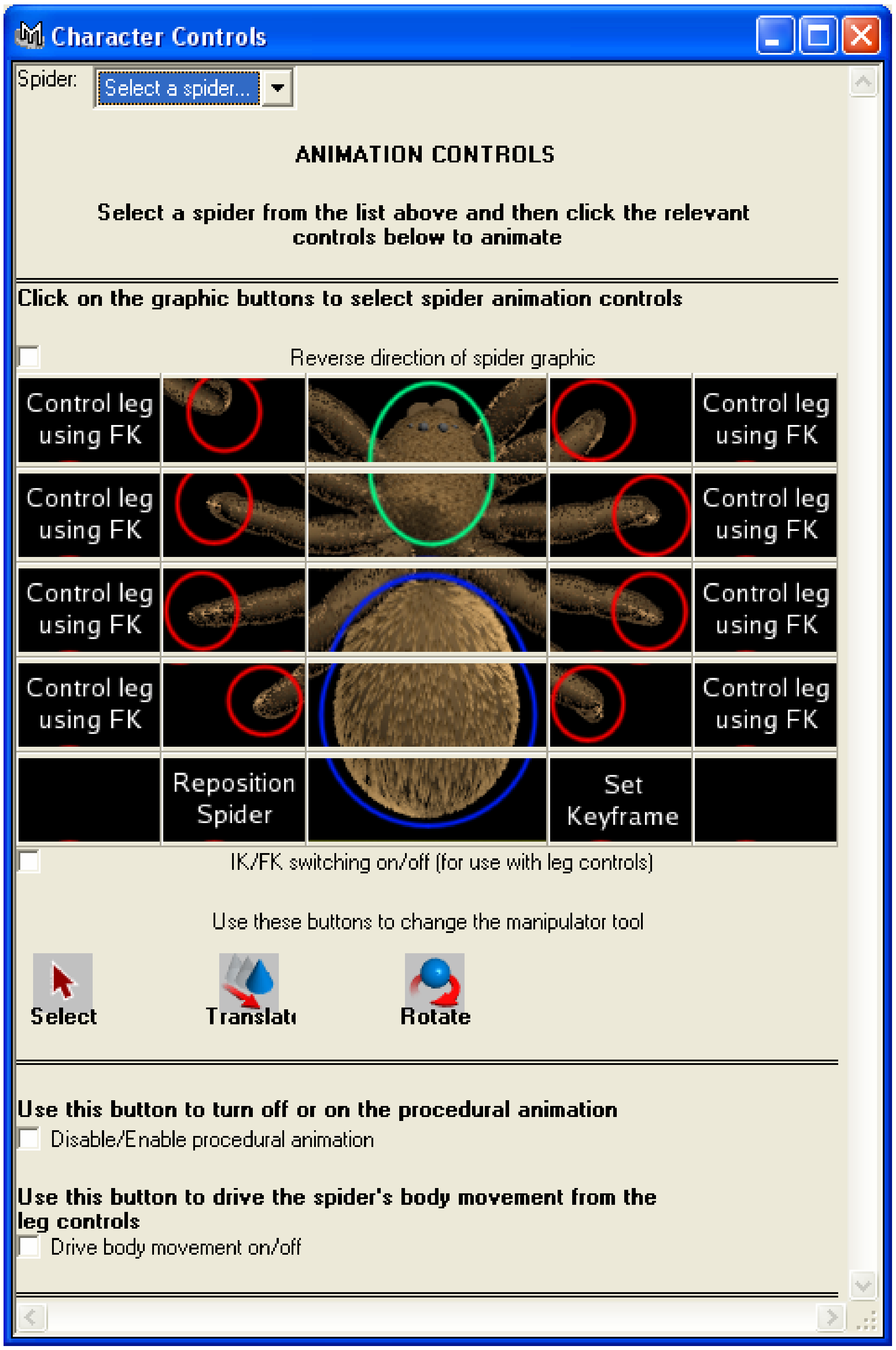
Varying Speed of Animation

The spider velocity in the animation for the Major Animation project was derived based on the average of its acceleration along the length of a motion path. This speed was then taken and used as input to drive the spider walk cycle. This worked fine for spiders moving at a constant rate, as the speed of the legs would always match. However, if the speed of the spider had varied along its path, the leg movement would no longer correspond.

For the Masters project, in order to enable more variation in the spiders' motion, the value of the U variable of the spiders' motion paths was obtained. This gave the exact location of the spider along the curve at any given time. This was used to calculate the exact acceleration of the spider, rather than using an approximation as before. The spiders could then be moved at varying rates along the motion path and their legs would speed up or slow down accordingly.

Secondary Animation

Due to the procedural nature of the animation, the portrayal of any spider movement outside of its normal walk cycle is not included. Therefore, a tool was created to assist with areas of the spider motion where secondary animation was required. A screenshot of the tool can be seen in Figure 8.


Character Controls

Spider: Select a spider...

ANIMATION CONTROLS

Select a spider from the list above and then click the relevant controls below to animate

Click on the graphic buttons to select spider animation controls

☐ Reverse direction of spider graphic

Control leg using FK				Control leg using FK
Control leg using FK				Control leg using FK
Control leg using FK				Control leg using FK
Control leg using FK				Control leg using FK
	Reposition Spider		Set Keyframe	

☐ IK/FK switching on/off (for use with leg controls)

Use these buttons to change the manipulator tool

Use this button to turn off or on the procedural animation

☐ Disable/Enable procedural animation

Use this button to drive the spider's body movement from the leg controls

☐ Drive body movement on/off

Figure 8: screenshot of Maya secondary animation tool developed using MEL

The tool is based on the Synoptic View that is available in XSI *[11]* and has been coded using MEL using a similar approach to that detailed by Chris Maraffi *[9]* and Susanne Werth *[18]*. The custom icons in the graphical view were produced by taking a low quality rendering from Maya *[5]* which was then edited using Photoshop *[10]*. By utilising the user interface commands in MEL, the images were reconstructed within the GUI, allowing a graphical representation of the spider and intuitive selection of its joints for animation.

The tool can be used to assist with animating any of the spiders in the Maya *[5]* animation scene files, so long as they share the same rig structure and naming standards. Its effectiveness is largely dependent on the quality of the rig and it assumes that any limits that need to be imposed on the joints have already been set before using the tool.

On running the script, the Maya *[5]* scene file is searched for all transform object nodes that are prefixed with the word "spider". The returned list is then displayed as a drop-down selection, allowing the user to select the correct spider for animation. As soon as the user needs to animate another spider in the scene all that is required is for a new spider to be selected from the list.

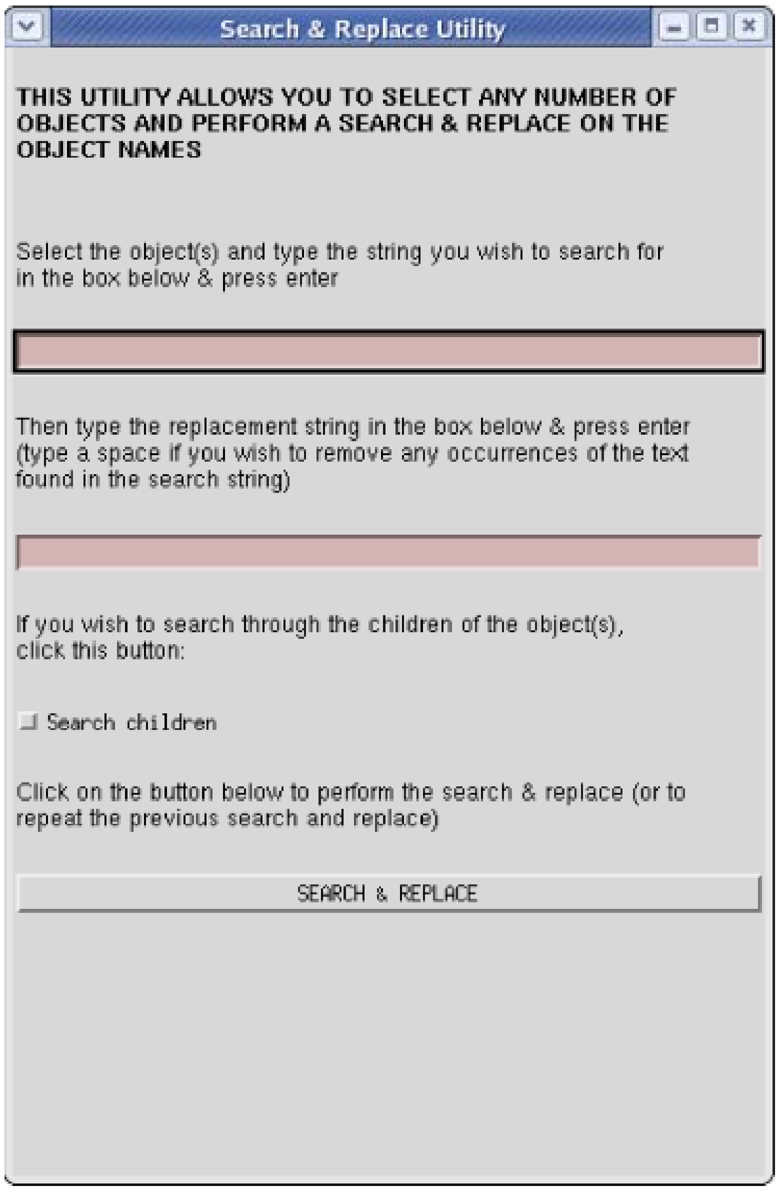
As well as the selection list, the user is presented with some other intuitive controls for animating the spider rig. A default manipulator has been intelligently set for each of the joints shown in the graphical view, based on the type of joint or locator selected. For example, when animating the selected spider using the IK handles on the legs, the move manipulator tool will be selected. Further buttons have been included in the user interface to allow one of the other manipulator tools to be selected instead if the default manipulator if required. Options also exist to switch from Inverse Kinematics (IK) to Forward Kinematics (FK) for the spider legs, should they need to be animated in this way.

The initial version of the interface contained a single graphical representation of the spider shot from overhead. However, during testing it was discovered that the selection process can become unintuitive if the spider being animated faces the opposite direction to the animation tool, in which case selecting the right or left components has to be done in reverse order to the spider being animated. As a result, a checkbox was added to enable the user to reverse the direction of the graphical representation of the spider to enable easier animation.

The tool also provides options to remove or include the procedural animation during the secondary animation process. For example, if slight adjustments to the existing procedural animation are required, the procedural walk cycle can be left in place. If the procedural animation is removed it will be held in memory for the duration that the user is working on the animation, in the event that it needs to be reinstated.

Other Tools & Utilities

When using the tool that assists with secondary animation, the spiders in the scene must have the same naming standards. When spiders are copied and pasted, or imported between scenes, Maya will tend to add a namespace to the names of the pasted or imported objects, preventing the secondary animation tool from locating the correct joints for animation. Whilst there are options to disable the namespacing for imported objects, there is no such option for pasted objects. Therefore, a further tool was developed in MEL to provide a search and replace utility for renaming any number of objects in a Maya scene.



THIS UTILITY ALLOWS YOU TO SELECT ANY NUMBER OF OBJECTS AND PERFORM A SEARCH & REPLACE ON THE OBJECT NAMES

Select the object(s) and type the string you wish to search for in the box below & press enter

Then type the replacement string in the box below & press enter (type a space if you wish to remove any occurrences of the text found in the search string)

If you wish to search through the children of the object(s), click this button:

☐ Search children

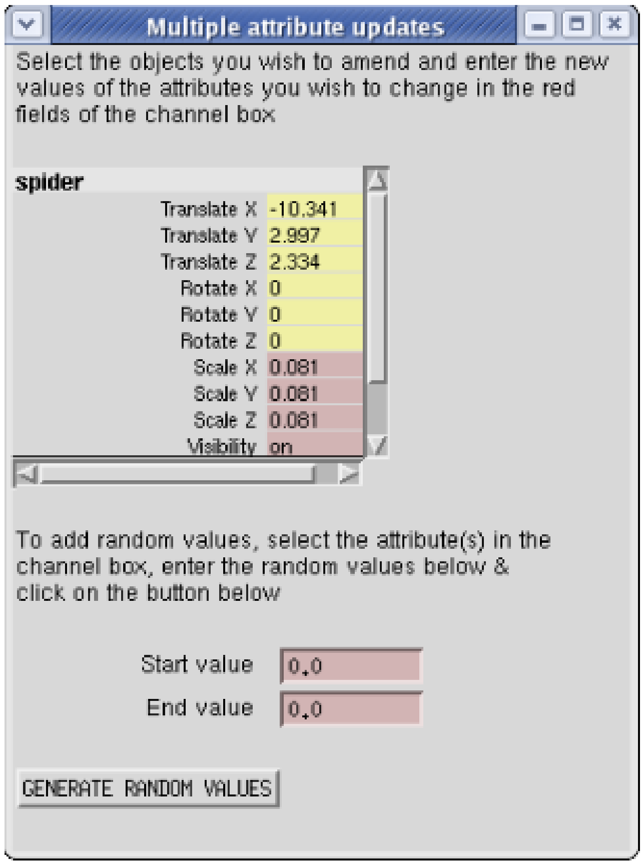
Click on the button below to perform the search & replace (or to repeat the previous search and replace)

SEARCH & REPLACE

Figure 9: Search and replace tool written using Maya's [5] Embedded Language (MEL)

This tool takes advantage of the MEL substitute and tokenize functions to search through a list of selected objects and replace any occurrences of the search string with the replacement string. Dependency nodes of parent objects can also be located and renamed as required and a checkbox is provided to allow for this.

In addition to the search and replace tool and in order to modify the keyable attributes of any number of objects at once, a further tool was developed. As well as being able to modify the properties of the selected objects using the channel box, the ability to introduce random numbers for any of the objects' keyable attributes was also included, thus avoiding the need to key expressions individually into Maya to perform the same functionality. A screenshot of the final tool can be seen in Figure 10.



Multiple attribute updates

Select the objects you wish to amend and enter the new values of the attributes you wish to change in the red fields of the channel box

spider

Translate X	-10.341
Translate Y	2.997
Translate Z	2.334
Rotate X	0
Rotate Y	0
Rotate Z	0
Scale X	0.081
Scale Y	0.081
Scale Z	0.081
Visibility	on

To add random values, select the attribute(s) in the channel box, enter the random values below & click on the button below

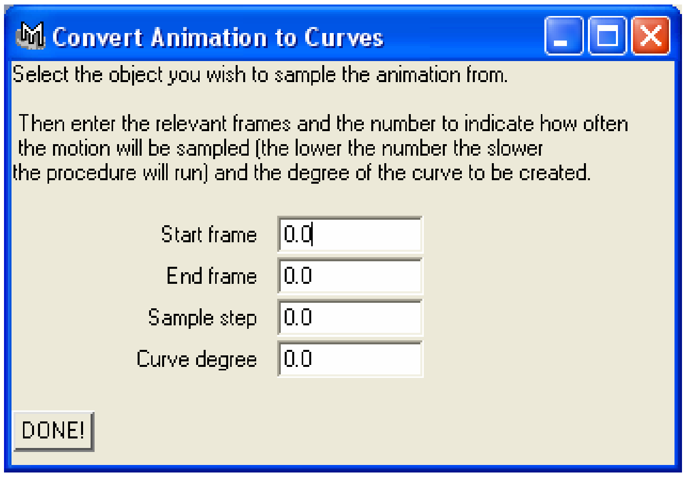
Start value 0.0

End value 0.0

Figure 10: Multiple object attribute updates tool written using Maya's [5] Embedded Language (MEL)

This tool was used to modify the sizes of all the spiders in the animation, in order to provide more variation than was seen in the animation at the end of the Major Animation project and match the varying sizes that occur naturally between spiders.

As well as the above tools, a User Interface was added to the tool used during the Major Animation project to create curves from the spider simulation in order to provide editable motion paths (full details of this tool can be found in the Major Animation project report). A screenshot of the interface can be seen in Figure 11.


Convert Animation to Curves

Select the object you wish to sample the animation from.

Then enter the relevant frames and the number to indicate how often the motion will be sampled (the lower the number the slower the procedure will run) and the degree of the curve to be created.

Start frame	0.0
End frame	0.0
Sample step	0.0
Curve degree	0.0

Figure 11: User Interface for tool that converts animation to curves

Lighting

The animation was set at night and so the lighting in the scenes had to reflect this. Whilst the lighting had to draw attention to the spiders, so as to highlight the procedural animation, the images also had to appear as if they were lit by a low light source. To provide more contrast in the images, a slightly brighter light was applied only to the spiders and "shadows only" lights were used to further darken some of the shadows in the scene. The method for creating shadows only lights is described in the publication by Birn *[3]* where the illumination of the light that needs to cast shadow can be removed by duplicating the light and negating the intensity value of the copy.

As well as using low key lighting techniques, another common practice to suggest darkness is to shift the colour balance towards blue tones and is an approach that has been adopted in this animation. In the paper by Barsky, Kosloff, Pasztor and Upstill *[13]*, the loss of saturation and perception of "blue shift" in night images is discussed. Barsky, Kosloff, Pasztor and Upstill *[13]* described this effect using the following formulae:

$$C_{WB}' = V * k_{Night} + C_{WB}(1 - k_{Night})$$
$$C_{RG}' = C_{RG}(1 - k_{Night})$$
$$C_{YB}' = V * k_{Blue} + C_{YB}(1 - k_{Night})$$

Equation 2: "blue shift" functions [13]

where V represents the scotopic luminosity (or rod response in the human eye) as a function of wavelength multiplied by the incoming spectrum of light, integrated across all wavelengths. As most computer generated images do not provide both photopic (or cone response) and scotopic luminance values for each pixel, a single luminance value (C_{WB}) is substituted for V into the night adaptation model. The dark adaptation model becomes a simple linear transformation of the original opponent channels:

$$[C_{WB} \quad C_{RG} \quad C_{YB}] \cdot \begin{bmatrix} 1 & 0 & k_{blue} \\ 0 & (1-k_{night}) & 0 \\ 0 & 0 & (1-k_{night}) \end{bmatrix} = [C_{WB}' \quad C_{RG}' \quad C_{YB}']$$

Equation 3: night adaptation algorithm *[13]*

To understand these formulae, an appreciation of human eye anatomy is required. For image processing, the human retina is comprised of two types of photoreceptor cells, rods and cones, so named because of their appearance under a microscope. The rods are more sensitive to low light levels and unlike the cones that have differing spectral sensitivity, rods have the same levels of response. As a result the rods are able to process luminance but not colour information, so in lighting levels that are too low to excite the cones properly, scenes appear to be almost monochromatic.

For this reason, it makes no sense to present an image lit by moonlight in full colour. Indeed, the technique of "day for night" photography used by filmmakers is to apply a dense blue filter to the camera lens in order to eliminate the most colour information and contrast from the image.

A similar technique has been adopted for this animation where the colours have been desaturated by introducing blue tinted lighting to the scene, thus producing the characteristic "blue shift" effect. An example can be seen in Figure 12.

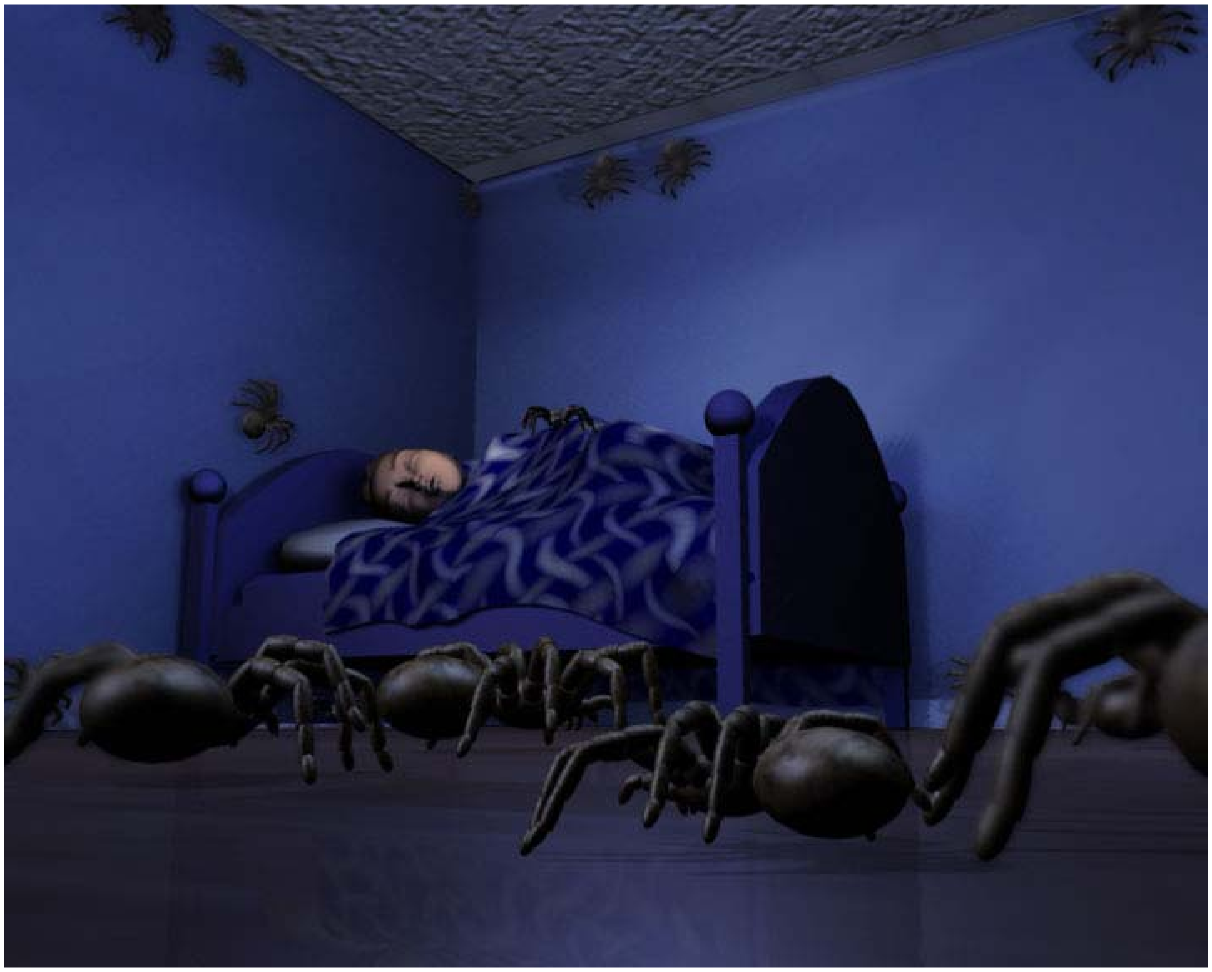


Figure 12: Rendering from animation showing the "blue-shift" effect

To simulate the moonlight filtering through tree branches outside the window, a shadow map was attached to the light used as the moonlight, as can be seen in Figure 13.

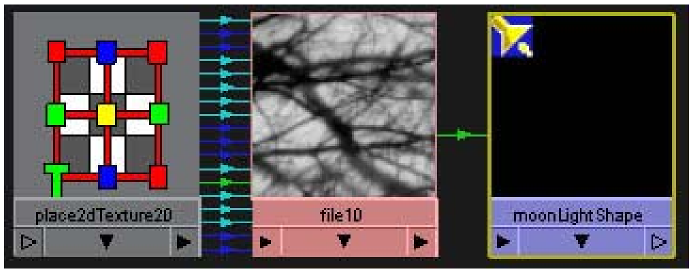


Figure 13: Image taken from Maya's [5] hypergraph showing network of light acting as moonlight

This then worked as a multiplier, so that areas coloured black would be represented as zero and areas shown as white represented as one (with grey areas having some value in between). When the light's intensity is multiplied by the shadow map, only the white areas of the map retain the light's original intensity. For all other areas of the map, the light's intensity will be decreased by the value represented in the map. The final results can be seen in Figure 14.

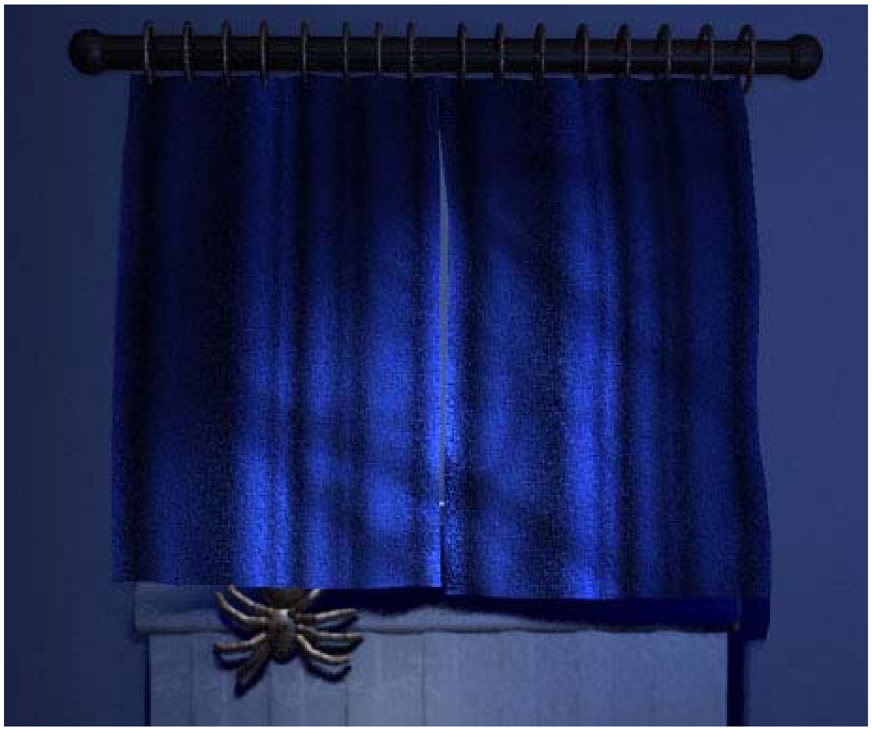


Figure 14: Image showing results of tree shadow map applied to the light acting as moonlight

Texturing & Shading

For the Major Animation project, the animation was shaded using the standard Lambert [12] and Blinn shaders in Maya [5]. The texturing was also extremely simple, using just a few image files, such as those for the curtains, ceiling and floor.

For the Masters project, the texturing and shading has been vastly improved. Dirt maps and shading techniques have been used in order to introduce imperfections to the surfaces in the scenes. A layered shader was used on the floor to add dirt to the existing laminate texture and the painted surfaces such as the radiator and the door were given a fractal based bump map to simulate the appearance of gloss painted surfaces. An example of the new gloss paint shader is shown in Figure 15.

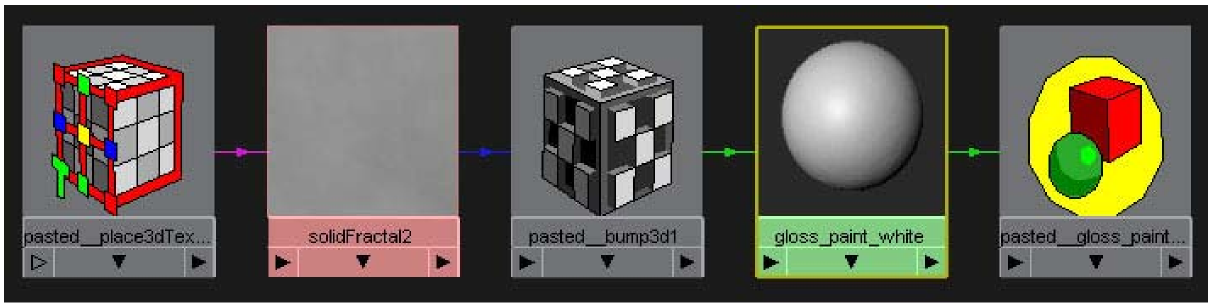


Figure 15 Gloss paint shading network, displayed using Hypershade in Maya [5]

In his literature, Birn [3] states that unlike Maya's Lambert shader, a real surface always has a very small amount of specular highlight, no matter how matte the surface may appear. Taking his work as reference, the Lambert [12] shaders were removed from the scenes and replaced with new shaders that included some specular content. The shaders were also very slightly displaced using bump maps to give a more realistic appearance.

Curtain Simulation

For the Major Animation project, the simulation of the curtains was performed using Maya's *[5]* soft body dynamics system, weighting the curtains so that the simulation was strongest at the lower middle portion of each curtain and had little to no effect at the top, where the curtain is suspended from the curtain pole.

Once the general simulation had been set up, springs were introduced and the stiffness and dampening of the springs was adjusted until they behaved as cloth   Objects surrounding the curtains were set as collision objects for the curtains to react to. Whilst collisions with the walls were working correctly, in some cases interpenetration was occurring during periods of the simulation as the curtains collided with the radiator. An example of this problem can be seen in Figure 16.

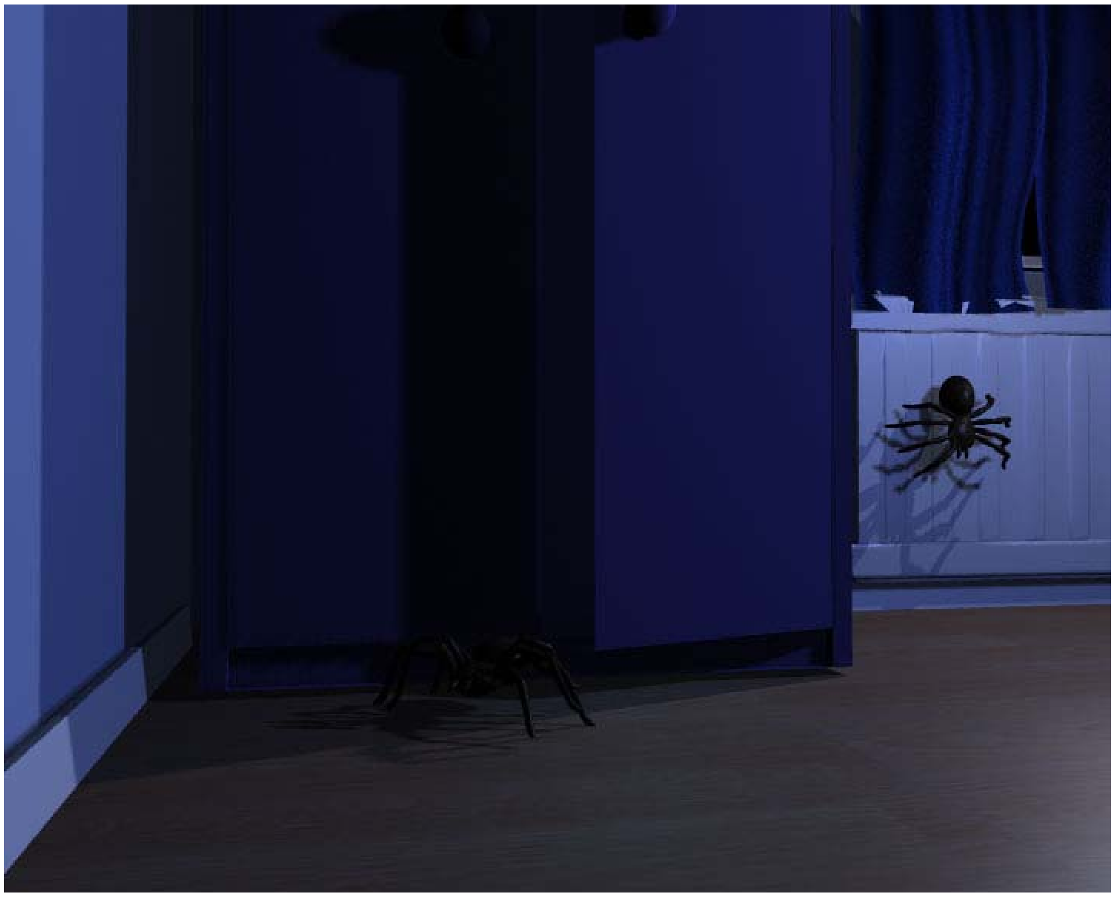


Figure 16: Screenshot from animation at the time of the Major Animation project, where interpenetration of the curtain with the radiator can be clearly seen.

Stand-in geometry had been created to use for the radiator collision detection during the Major Animation project. For the Masters Project, as well as fine tuning the collision tolerance, the radiator stand-in was scaled up by a small amount and moved slightly further back from the curtains (as the starting position of the curtains prior to this was too close for all collisions to be detected). As a result, the curtain simulation now works as expected and the curtains correctly react to the radiator. An example screenshot from the new animation can be seen in figure 17.

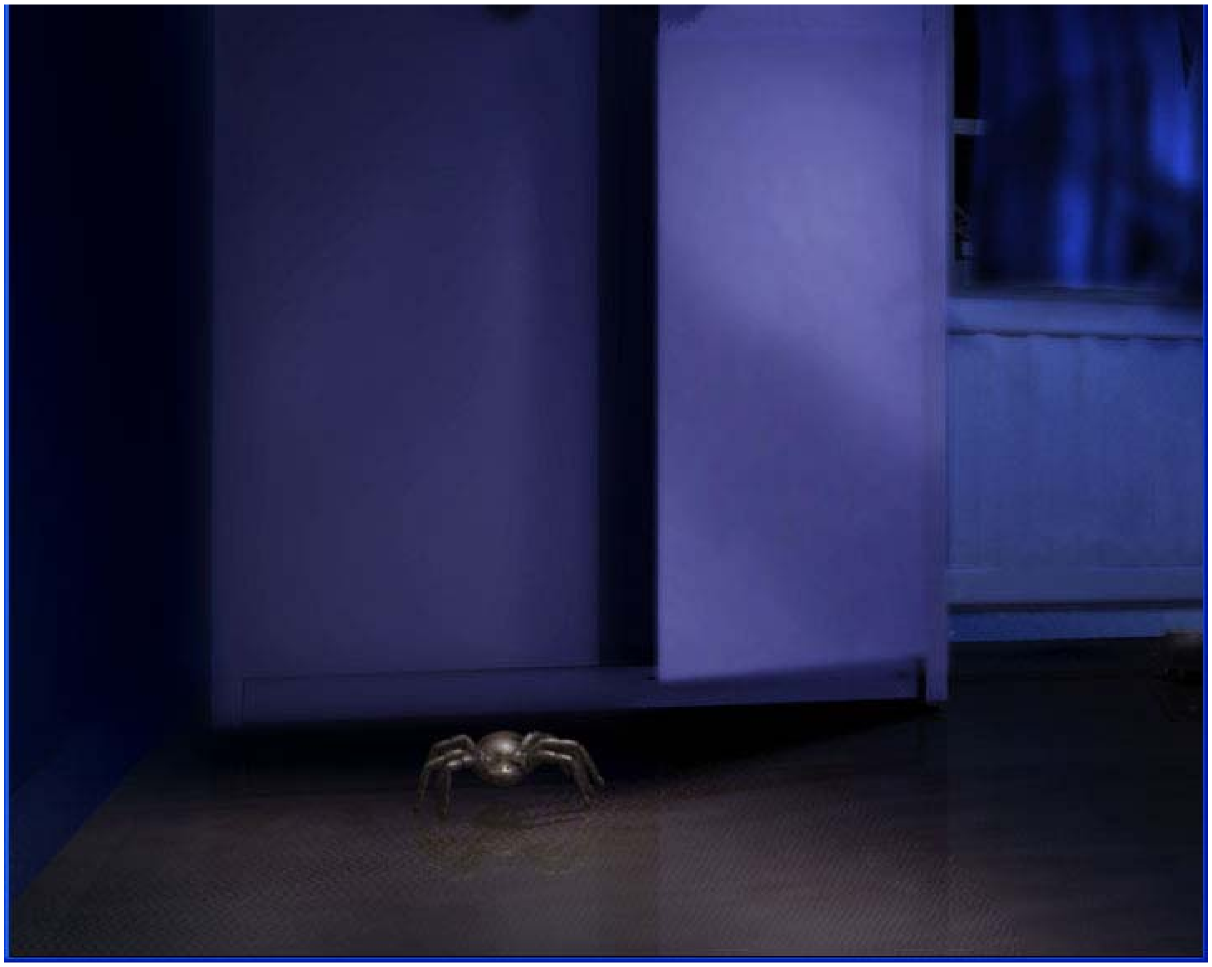


Figure 17: Screenshot from latest version of the animation, where the curtain simulation is now working correctly.

Rendering Techniques

For the Major Animation project all the rendering was done in a single pass. Due to the simplicity of the scenes and shaders at that stage, this was an adequate solution, although it still required a full render of the complete frame every time an amendment was made to some aspect of the scene. An example scene from the Major Animation project can be seen in Figure 18.

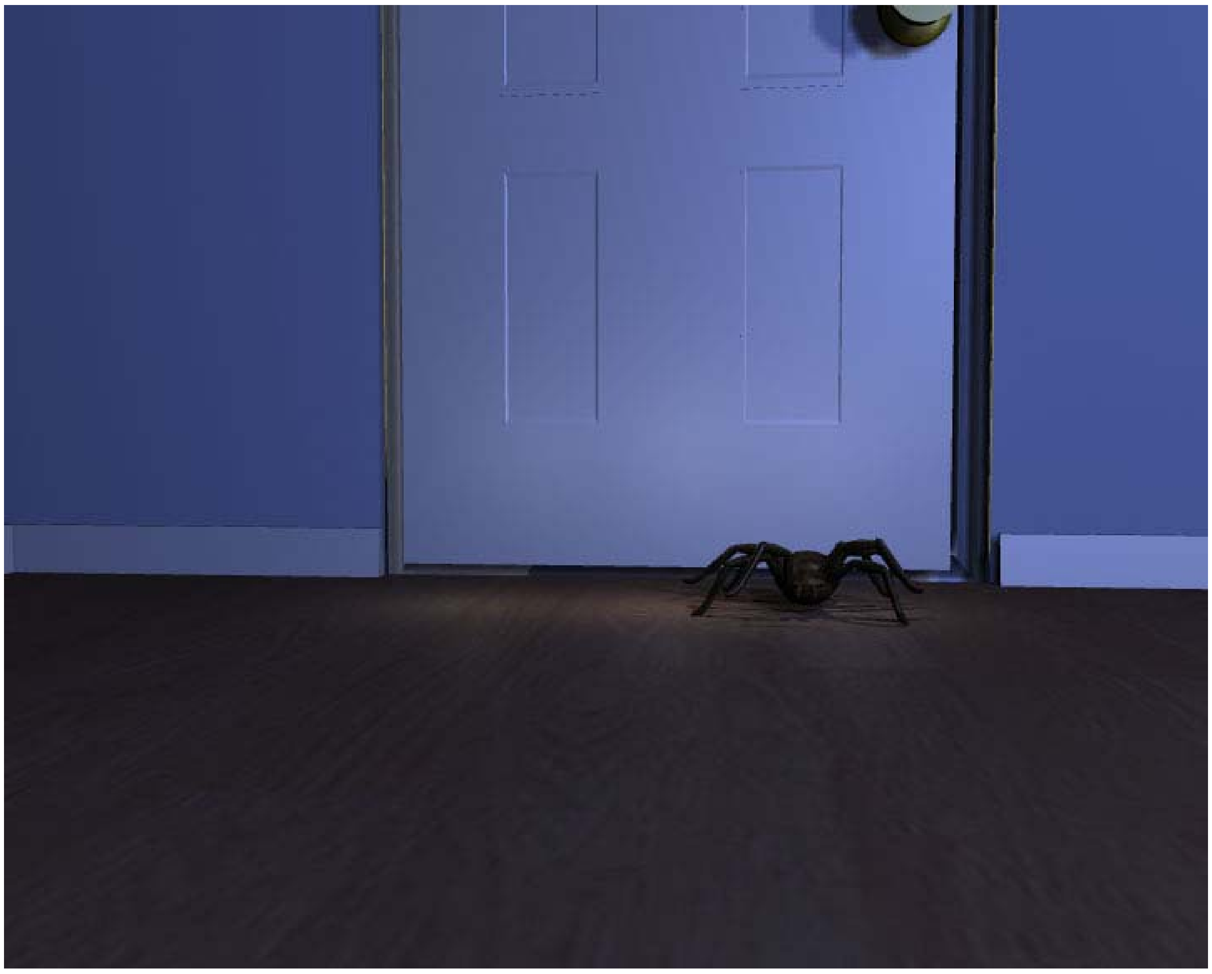


Figure 18: Screenshot from the animation at the time of the Major Animation project

For the Masters project a layered composition, similar to that discussed in the work by Birn [3], was chosen for the rendering to allow for the multiple scenes to be rendered in the most efficient time, with the composition being done later in Shake [17]. This technique also allows for renders to be done using different software if required.

Some initial renders were performed using RenderMan [14], taking advantage of the RenderMan Artist Tools [7]: Whilst RenderMan [14] would have been the preferred renderer if custom RenderMan [14] shaders had been required, following some of the early test renders it was discovered that the lighting varied significantly from test renders performed using both the Mental Ray [16] renderer and Maya's [5] own renderer. In addition, problems were experienced with the RenderMan [14] fur plug-in.

As no RenderMan [14] shaders were explicitly required and Mental Ray [16] provided good results and render times, it was decided that the Mental Ray [16] renderer would be used for rendering the animation instead. Some issues were experienced when rendering fur, but these were overcome during the compositing process.

Using the techniques detailed in this section, resulted in much improved images than were seen for the Major Animation project. The new version of the same scene shown in Figure 18, using these new rendering techniques, can be seen in Figure 19.

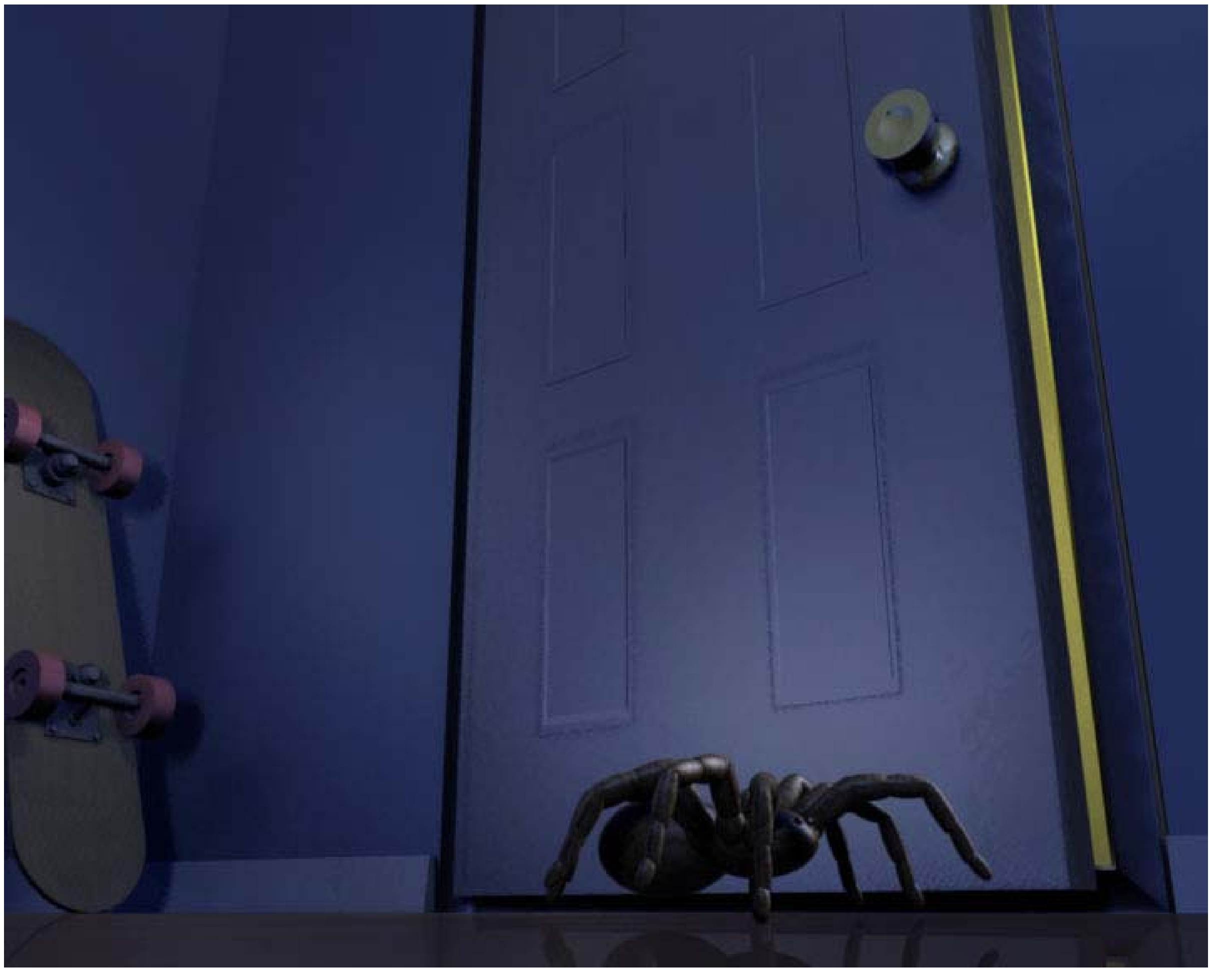


Figure 19: Screenshot from the animation using the new rendering techniques

<u>*Rendering Pipeline*</u>

A pipeline had to be set up to deal with the number of different renders being performed. A well-planned directory structure was required to hold the different passes and Maya *[5]* scene files. In some cases, additional scene files were created specifically for some of the rendering passes, especially where the shaders on the objects had to be replaced, such as to produce the ambient occlusion pass (Maya 7.0 provides an integrated solution, but this version was not available for this project).

In order to integrate the different passes, a bash script was developed to read through a sequence of image files and rename and/or renumber them. This script was not only used to renumber files that needed to be input into Shake [17] for compositing, but was used to integrate the final renders into one sequence ready for exporting to a movie file. A visual representation of the rendering pipeline can be seen in Figure 20.

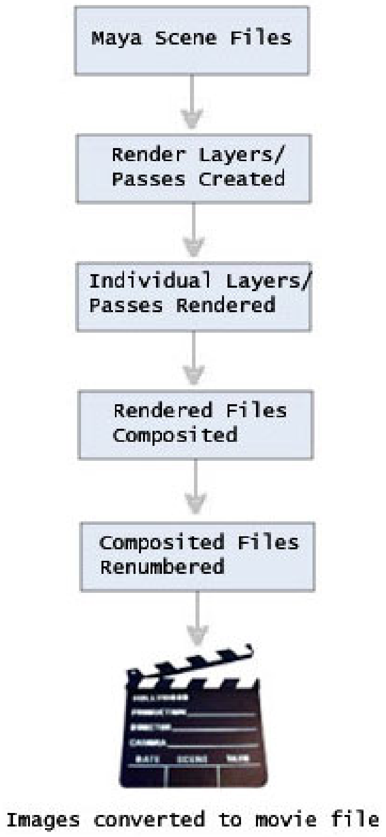

Maya Scene Files
Render Layers/ Passes Created
Individual Layers/ Passes Rendered
Rendered Files Composited
Composited Files Renumbered
Images converted to movie file

Figure 20: Diagram showing rendering pipeline

<u>*Render Layers/Passes*</u>

Each scene was comprised of a series of different layers and passes, the most simple of which separated out the static objects from the moving objects and the ambient occlusion pass from the beauty pass. Some files had a large number of different components, depending on the level of control required. The most complex structure existed in the bear scene, which comprised of the following layers:

- Curtains
- Background
- Bear (for use with the fur pass)
- Hero spider
- Wall spiders

and the following passes:

- Colour (comprising of the diffuse and specular components of the scene)
- Shadows
- Fur
- Ambient Occlusion

Problems were experienced using the render layers and passes settings in Maya *[5]* 6.5 running on Linux. Changing these properties in the render globals settings of Maya*[5]* caused the software to terminate with a fatal error. The only way to overcome this problem was to set the options using MEL script rather than the render globals editor and then reset everything back once the passes had been rendered. The resetting was crucial as otherwise the software would terminate in the same manner as using the render globals option.

Initially, the shadow pass option in the render globals settings in Maya *[5* was used to produce the shadow pass. However, the shadows produced in this way were significantly different from the shadows produced when incorporated with the rest of the scene. Therefore, instead of using this option, the objects in the scene that receive shadows were assigned a "Use Background" shader. An object with a Use Background shader does not show up in the scene; but shadows or reflections from surrounding objects will be shown when the image is rendered. By turning off the primary visibility flag on the surrounding objects, and setting the reflectivity in the Use Background shader to zero, a shadow pass can be created, as shown in Figure 21.

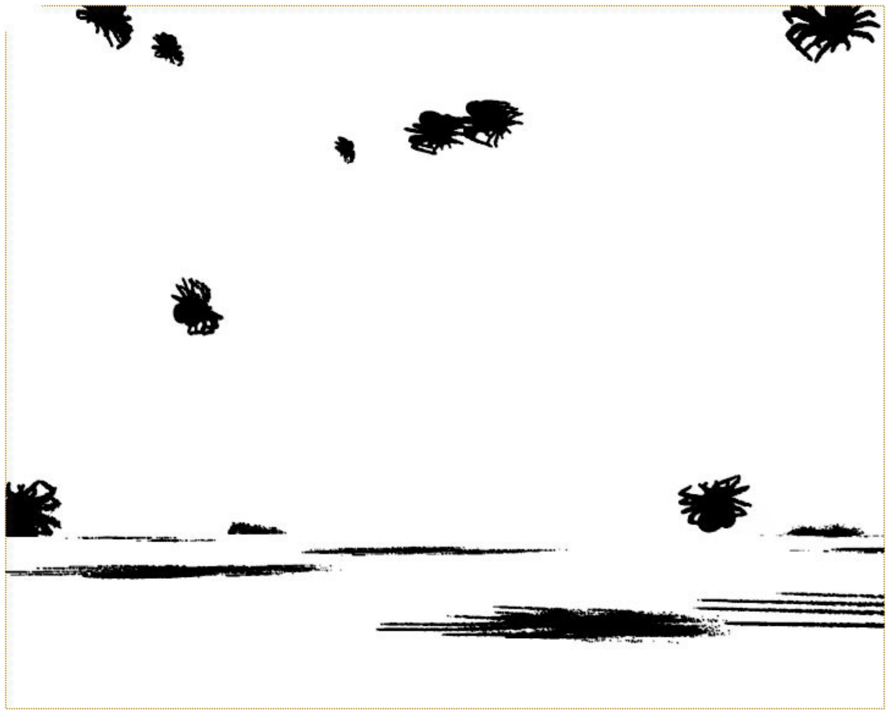


Figure 21 Shadow pass produced from applying the Use Background shader to the floor and walls.

The ambient occlusion pass for the animation was achieved by linking the incandescence channel of Maya's *[5]* Lambert *[12]* shader up to the Mental Ray *[16]* ambient occlusion shader (see Figure 22 below).

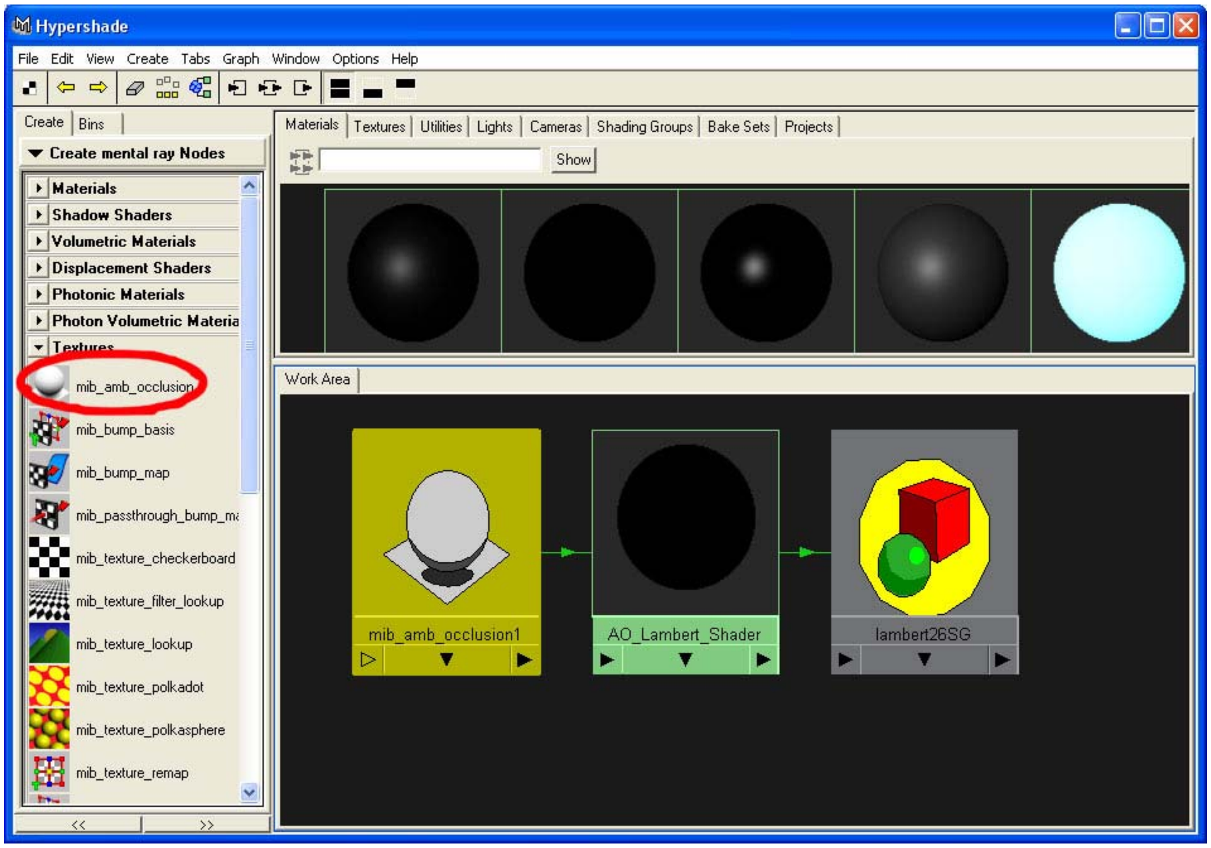


Figure 22 Ambient Occlusion shader, displayed using Hypershade in Maya [5]

The Lambert [12] shader with the ambient occlusion material was then applied to all objects in the scene and rendered. An example of a rendered ambient occlusion pass can be seen in Figure 23.

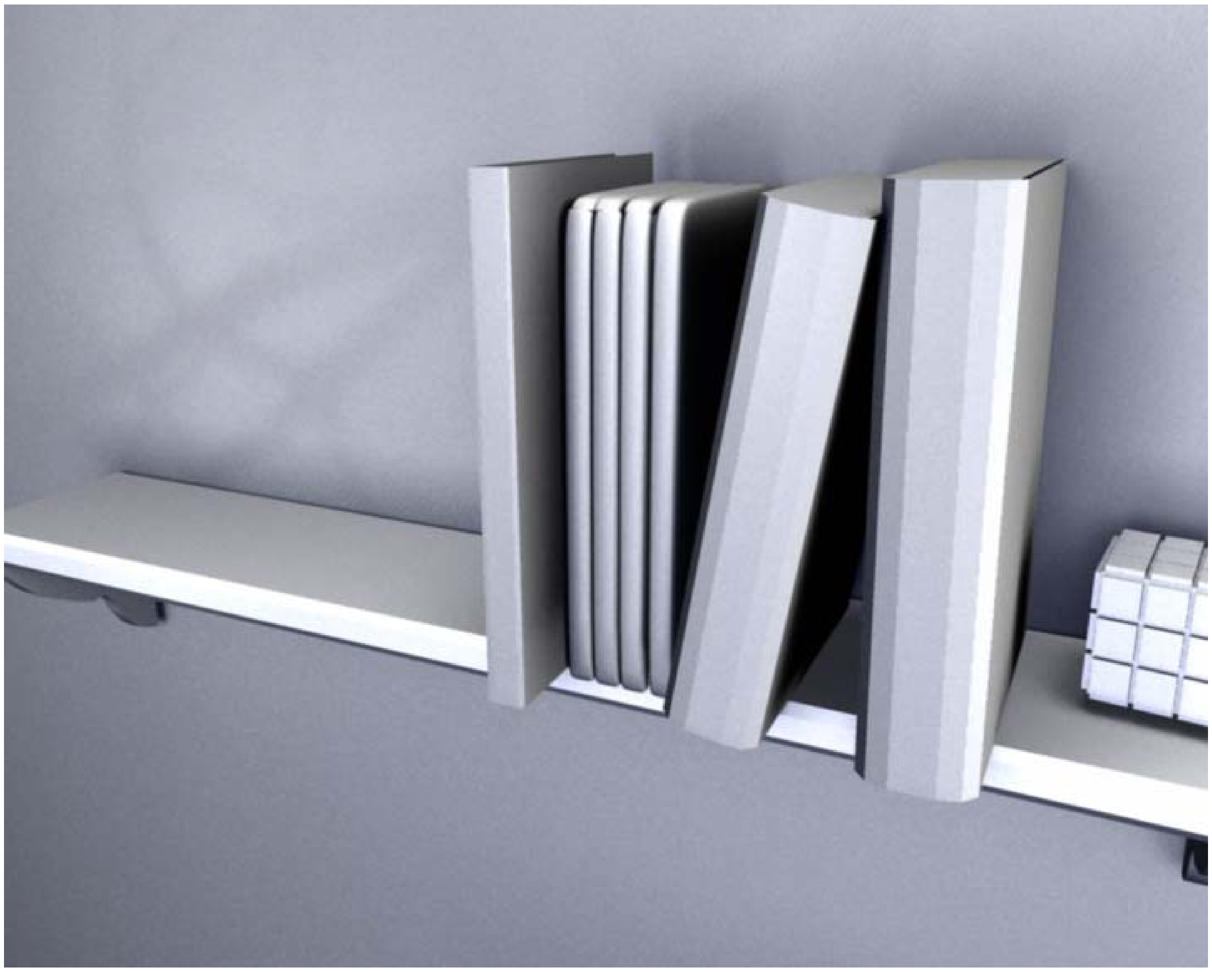


Figure 23 Shelf scene rendered using Ambient Occlusion shader

Rendering Realistic Fur

Rendering the fur for the bear scenes proved extremely challenging, as no single renderer provided a complete solution. Rendering with Mental Ray *[16]* produced fur that was aesthetically unpleasing (as shown in Figure 24 below), but it did produce excellent shadows.

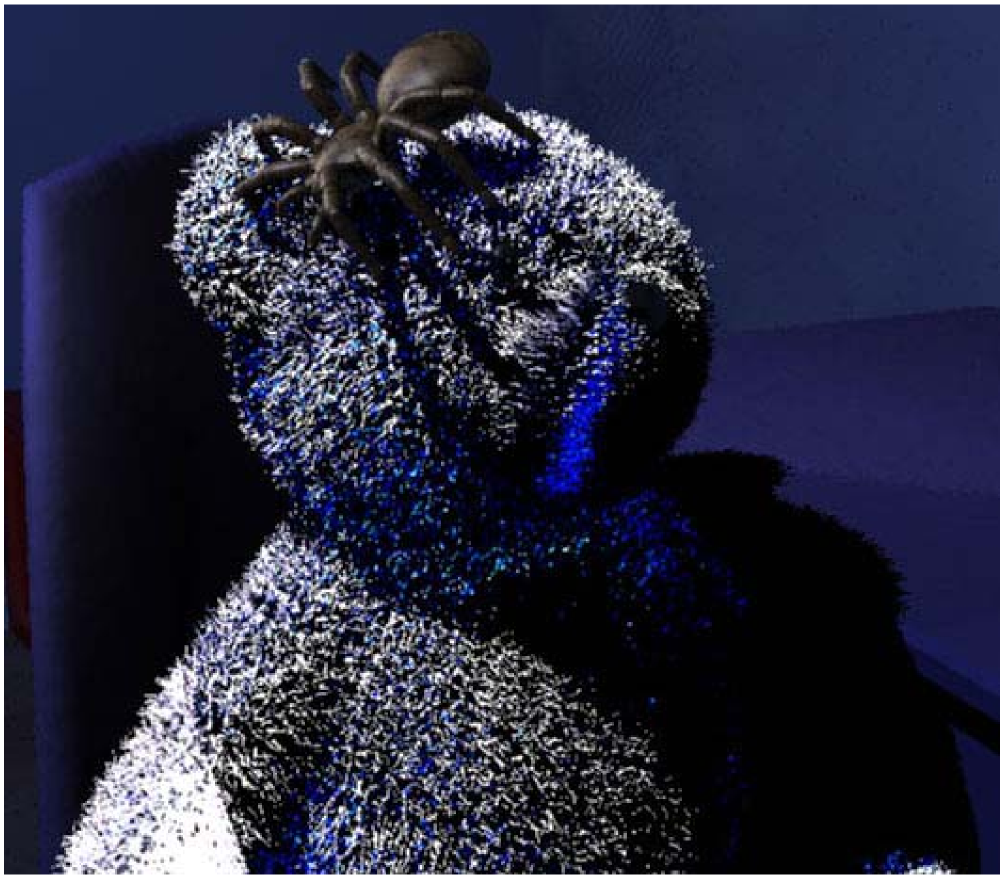


Figure 24: Shadowed fur rendered using Mental Ray [16].

Rendering fur using Maya's inbuilt renderer could only be done without shadows, as when shadows were activated, it resulted in the overexposure and shadow displacement shown in Figure 25.

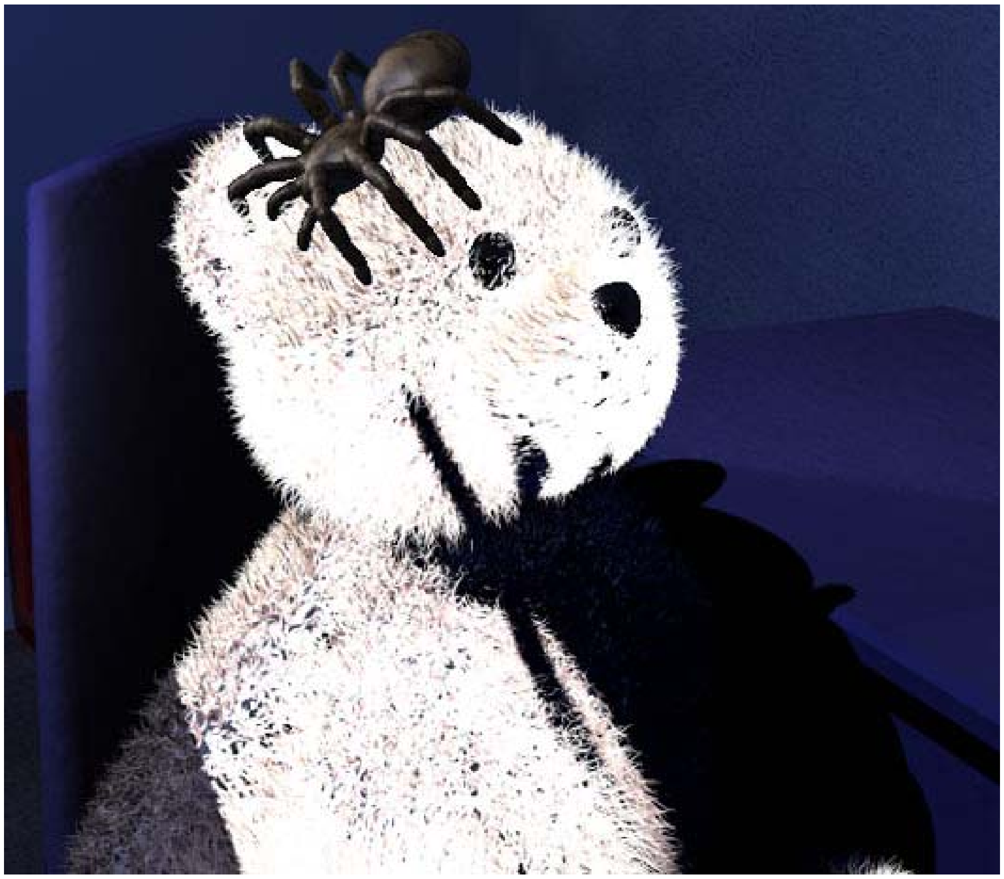


Figure 25: Shadowed fur rendered using the Maya [5] software renderer.

However, rendering fur without shadows was not adequate, as it did not provide very realistic results, especially for an animation set at night. In order to render the fur and shadows, a complicated network was required in Shake *[17]* to take the best parts of the images produced by the two different renderers and composite them together. Taking the images produced using Mental Ray *[16]* and using Shake *[17]* to turn them into shadow maps allowed a non-shadow rendered fur produced using Maya's *[5]* inbuilt renderer to be shadowed correctly. The final result can be seen in Figure 26.

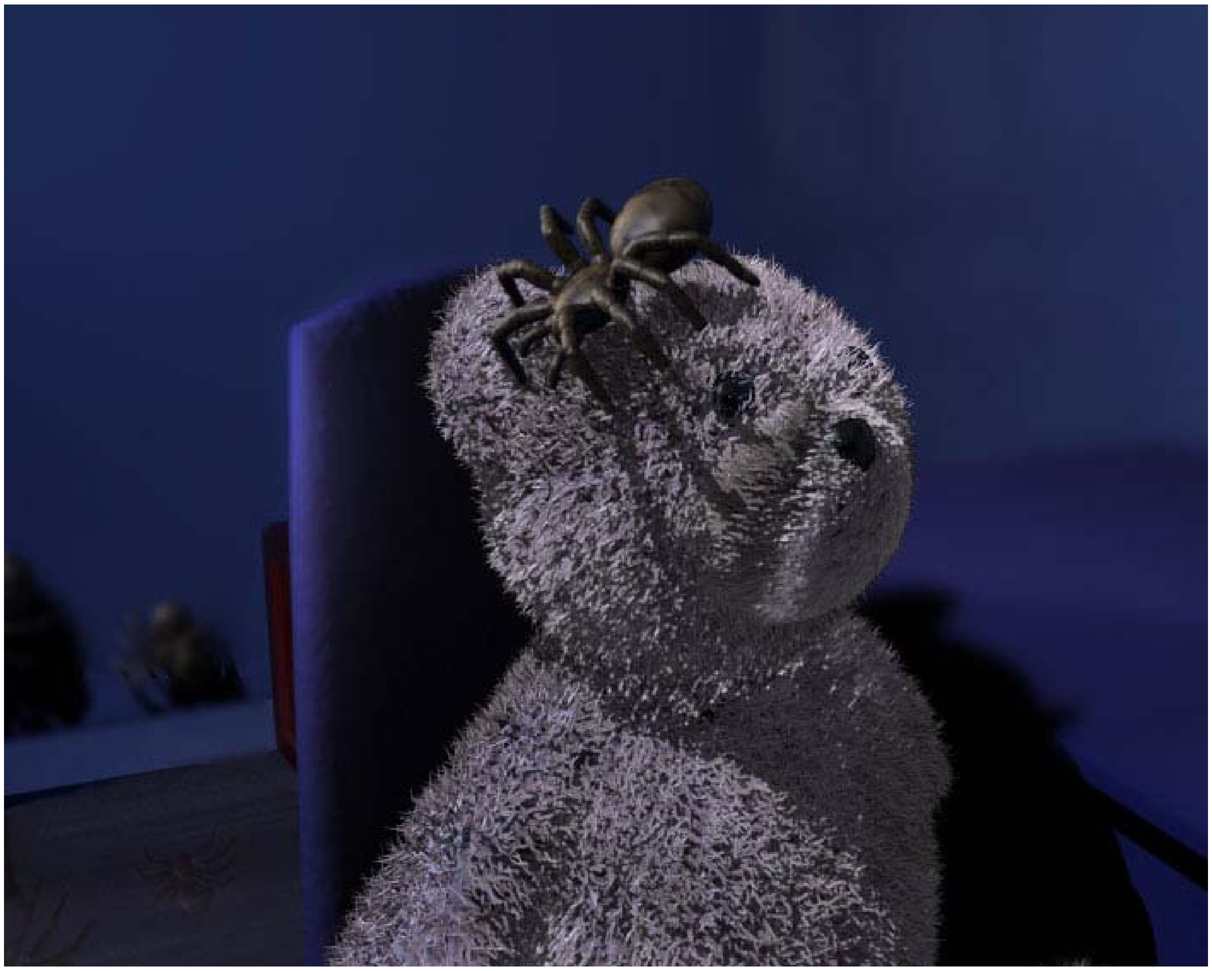


Figure 26: Final rendered image produced with Mental Ray [16], Maya [5] and Shake [17].

Compositing

Due to the number of different layers and passes being rendered, a new Shake *[17]* script file had to be produced for every scene in the animation. An example network in Shake *[17]* for compositing the bear scene can be seen in Figure 27.

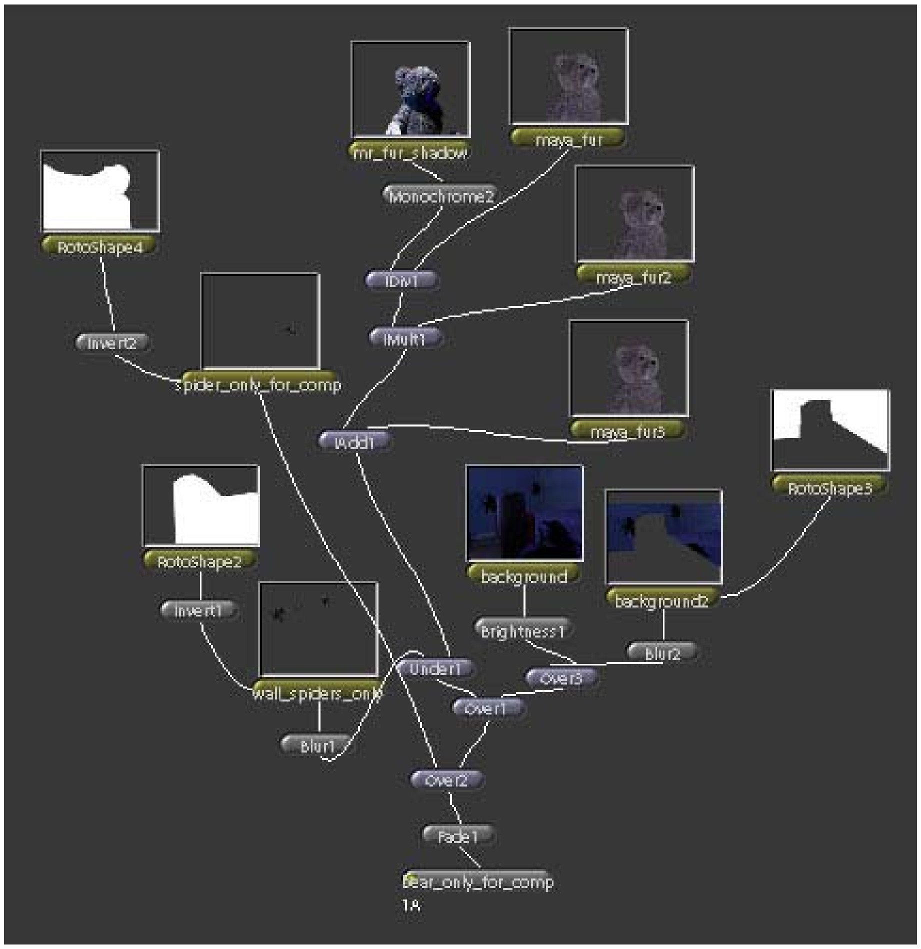


Figure 27: Example of node network in Shake [17] used to produce final renders for bear scene.

All of the layers, passes and composited images had to be carefully managed to ensure they could be easily located and to avoid files being overwritten by one another.

The depth of field effects in the scenes were creating during the compositing process, as it was easier to create and control this effect during the final compositing process than in all the separate passes. Compositing the images would have also been much more difficult had the scenes been rendered with the depth of field already in place.

Enhancements

The procedural walk cycles could be further improved with some very small adjustments to the acceleration and deceleration of the leg motion in some places to make the ease in and ease out a touch smoother. In addition, the timing of the leg movement when the spiders are turning could be enhanced. Otherwise, the animation seems to work well.

Due to time constraints, only a small amount of secondary animation was produced. Given more time, the secondary animation tool could have been used more widely in the animation. As the spiders were created before the secondary animation tool was developed, they were created as normal Maya [5] objects and then animated. If they had been created as character objects instead, this would have made searching for them much easier. As it stands, the animation tool performs its search by looking at all transform objects prefixed by the word “spider”. This means that if non-spider objects in the scene have this prefix, they will be picked up in the search and will appear in the drop-down list when the tool is loaded. Whilst this is more of an annoyance than a problem, for the tool to work correctly, this would need to be addressed.

The problem with the spiders’ legs going through surfaces was only temporarily resolved for the purposes of this animation and a better solution could have been put in place if time had allowed, especially where the spiders are walking on uneven surfaces.

The tool to assist with the secondary animation produces an error the first time it is loaded, and needs to be loaded for a second time before it can be used. Some of the error checking for all of the tools could also be tightened.

Conclusion

Overall, the Masters Project has met its objectives and the render quality of the animation far exceeds that presented for the Major Animation project. The time taken to set up the lighting and produce the render passes was far greater than originally anticipated and this did affect some of the later processes, especially the introduction of secondary animation in the scenes, which was only extensively carried out in the shelf scene.

To further improve the animation, a proper solution for constraining the spider's legs to the surfaces needs to be put in place, similar to the method set out in the paper by ap Cenydd and Teahan [2]. However, it should be noted that ap Cenydd and Teahan's [2] paper details a very complete solution and as such, is a project in its own right, so could not have easily been implemented as a part of this project due to time constraints.

The procedural spider animation works well, particularly with the new shadow passes. The fur on the bear and the shadow that appears as the spider crawls over it is particularly effective.

References

1. Anjyo, K. & Faloutsos, P., Capture and Synthesis of Insect Motion. *SIGGRAPH*, 2005
2. ap Cenydd, L. & Teahan, W., Arachnid Simulation: Scaling Arbitrary Surfaces, *EU UK Theory and Practice of Computer Graphics*, 2005.
3. Birn, J., [Digital] Lighting & Rendering, *New Riders Press*; 1st edition, January 2000.
4. Shake Software, Apple Software, 2002.
5. Maya Software, Autodesk Media & Entertainment (formerly Alias Systems Corporation), February 1998
6. Bellman, R., Applied Dynamic Programming, *Princeton University Press*, 1962.
7. Reynolds, C. Flocks, Herds and Schools: A Distributed Behavioural Model, Published in *Computer Graphics*, **21**(4), July 1987, pp. 25-34. (ACM SIGGRAPH '87 Conference Proceedings, Anaheim, California, July 1987.)
8. Bunnel, M., Dynamic Ambient Occlusion and Indirect Lighting, GPU Gems 2, *Addison-Wesley*, 2005.
9. Maraffi, C., Maya® Character Creation: Modeling and Animation Controls, *New Riders*, 2003.
10. Adobe Photoshop, Adobe Systems, 1990.
11. Softimage XSI, Softimage Co., 1994
12. Lambert, J. H., Photometria, published 1760.
13. Barsky, B.A., Kosloff, T.J., Pasztor, E.C. & Upstill, S.D., An Opponent Process Approach to Modeling the Blue Shift of the Human Color Vision System, *ACM SIGGRAPH International Conference Proceeding Series; Vol. 73*, 2004.
14. RenderMan Interface, Pixar Animation Studios, 1988.
15. RenderMan Artist Tools, Pixar Animation Studios.

16. Mental Ray, Mental Images, 1989.

17. Shake, Apple Computer, 1997.

18. Werth, S., Creating a Graphic User Interface Animation Setup for Animators from Maya Secrets of the Pros, *SYBEX Inc*., 2002.