

Animation

Overview

The RenderMan interface provides a mechanism by which a sequence of image files can be produced by rendering several scene descriptions contained in a **single** RIB file. Each scene description corresponds to an individual rendered image, and in turn, each of these forms one frame of an animation. A small part of a very simple animation RIB file is shown below.

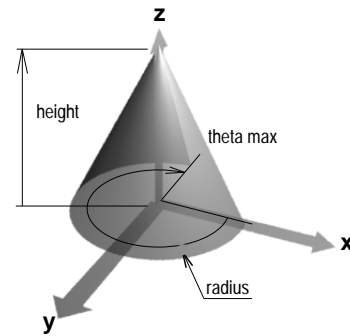
RIB script

```
Projection "perspective" "fov" 45
Format 400 300 1

Translate 0 0 5
Rotate -110 1 0 0
Rotate 70 0 0 1

FrameBegin 1
  Display "grow.001" "file" "rgba"
  WorldBegin
    Color 0 1 0 #green cone
    Cone 0.25 0.75 360
  WorldEnd
FrameEnd

FrameBegin 2
  Display "grow.002" "file" "rgba"
  WorldBegin
    Color 0 1 0 #green cone
    Cone 0.30 0.75 360
  WorldEnd
FrameEnd
:
:
: (additional FrameBegin / FrameEnd blocks)
:
:
FrameBegin 16
  Display "grow.016" "file" "rgba"
  WorldBegin
    Color 0 1 0 #green cone
    Cone 1.25 0.75 360
  WorldEnd
FrameEnd
```



Cone height radius thetamax

Notice how each scene description is “bracketed” by the paired statements, FrameBegin/FrameEnd. In this example a cone is increasing in height from 0.25 to 1.25 units. Quite clearly in the context of writing RIB files with a word processor an enormous amount of tedious work ie. copying, pasting and editing, would be required to produce anything but very brief and simple animations. To enable you to experiment relatively conveniently with animation techniques you will be using a utility program called “FrameUP”.

Using FrameUP

A common technique used in animation is to define a series of *key-frames* that specify the characteristics of a scene that undergoes change over a period of time. In this way an animator does not have to attend to the specific details of an animation on a frame by frame level. Unfortunately, RenderMan only deals with animations at this low level – it does not have the concept of key-framing built into its scripting language. Normally an animator would use an interactive system that would, for example, allow key-frames to be defined and then subsequently converted into long RIB files that would be processed by a renderer in the normal way. Generally animators do not read these RIB files, much less know what to do with them even if they did!

The utility software, FrameUP, has been designed to allow you to produce animations with RenderMan but at the same time to avoid the chore of producing each and every frame description by hand. By processing your animation file BEFORE it is passed to the renderer, FrameUP allows some “extra functionality” to be added to a RIB file. For example, the previous animation could have been written as follows,

ANIMATION script

```
Projection "perspective" "fov" 45
Format 400 300 1
Display "grow" "file" "rgba"
Translate 0 0 5
Rotate -110 1 0 0
Rotate 70 0 0 1
```

```
Tween "from" 1 "to" 2 "frames" 16
```

KeyFrameBegin 1

```
WorldBegin
  Color 0 1 0 #green cone
  Cone 0.25 0.75 360
WorldEnd
```

KeyFrameEnd

KeyFrameBegin 2

```
WorldBegin
  Color 0 1 0 #green cone
  Cone 1.25 0.75 360
WorldEnd
```

KeyFrameEnd

The lines in bold printing highlight three of the most important additional statements supported by FrameUP.

Upon reading the Tween statement, FrameUP produces a sequence of inbetween frames, hence its name, that represent the changes that occur in going "from" keyframe 1 "to" keyframe 2 over the duration of 16 "frames".

FrameUP – continued

Any number of key frames can be used so long as each has a unique number or tag ie.

```
KeyFrameBegin 3
WorldBegin
  Color 0 0 1          #changed to blue
  Cone 1.25 0.75 360  #height the same as keyframe 2
WorldEnd
KeyFrameEnd
```

Pairs of key frames that will be tweened must have an identical structure – only numeric parameters are allowed to change. For example, it would be illegal to substitute the cone in key frame 2 for a sphere. If FrameUP finds any mistakes of this kind it will warn you and refuse to process the animation file. However, like RenderMan it ignores comments.

In addition, any number of Tween statements can be inserted. For example, the animation script on the previous page could have included the key frame shown above so that the cone increases in height, then changes from green to blue before returning slowly to its original height and colour, ie.

```
Tween "from" 1 "to" 2 "frames" 16
Tween "from" 2 "to" 3 "frames" 16
Tween "from" 3 "to" 1 "frames" 30
```

In each segment of the animation the changes, be they height or colouration, would change linearly ie. at a constant rate. FrameUP also allows changes to occur more gracefully ie.

```
Tween "from" 1 "to" 2 "frames" 16 "smooth"
```

In this case the rate of 'growth' of the cone would be slow at first, becoming more rapid around the 8th frame and then finally it would gradually slow down to the final frame (16). Infact, by tween'ing from key frame 1 to 2 and then back to key frame 1 using "smooth" on both sequences the cone would have a decidedly "bouncy" feel to its motion.

The way in which the tweening occurs can be further controlled by the use of a technique called "easing-in" and "easing-out" – these are sometimes referred to as "fairing-in" and "fairing-out". For example, the cone could be made to abruptly spring to its full height in the first sequence using the following statement,

```
Tween "from" 1 "to" 2 "frames" 16 "easeout" 1.0
```

On the other hand the second sequence could convey the feeling that the apex of the cone is falling as if it were under the influence of gravity ie. slow at first but becoming faster and faster,

```
Tween "from" 2 "to" 1 "frames" 16 "easein" 1.0
```

FrameUP – animated texture and displacement maps

In each case the number following "easein" and "easeout" may be set to any value between 0 and 1. A value of "easeout" 0.3, for example, would indicate that only the final 30% of the inbetweened frames would be eased-out ie. the rate of change would decrease to zero.

Finally, it should be noted that easing-in and easing-out may, to a certain extent, be combined ie.

```
Tween "from" 2 "to" 1 "frames" 100 "easein" 0.2 "easeout" 0.4
```

In this animation, 100 frames in length, only the first 20% and the last 40% of the sequence are effected by easing-in and easing-out. FrameUP (as of version 0.90) will not allow easing-in and easing-out to overlap. In other words if you attempt to set "easein" to 0.6 AND "easeout" to 0.7, FrameUP will reduce the "easeout" period to 0.4, since 60% and 40% add up to 100%. However, this restriction may change in later versions of the software.

FrameUP is able to animate texture and displacement maps in the sense that key frames can specify two different image files for corresponding pairs of MakeTexture statements, for example,

```
Tween "from" 1 "to" 2 "frames" 25
```

```
KeyFrameBegin 1
  MakeTexture "myImage.001" "myTex.tx" "periodic" "periodic" "gaussian" 2 2
  WorldBegin
    Surface "texmap" "texname" ["myTex.tx"] "maptype" 2
    Sphere 1 -1 1 360
  WorldEnd
KeyFrameEnd
```

```
KeyFrameBegin 2
  MakeTexture "myImage.025" "myTex.tx" "periodic" "periodic" "gaussian" 2 2
  WorldBegin
    Surface "texmap" "texname" ["myTex.tx"] "maptype" 2
    Sphere 1 -1 1 360
  WorldEnd
KeyFrameEnd
```

The important point to notice here is that each of the multiple images files MUST have a 3 digit extension and that the number of frames specified in the Tween statement, 25 in the example shown above, matches the number of images files to be converted to texture files with the MakeTexture statement. Of course ALL of the image files (eg. myImage.001 to myImage.025) must be in the same folder as the RIB file – FrameUP cannot create missing files by morphing images.

Because texture files are generally very large each frame of the animation uses the same name for the texture produced by MakeTexture ie. "myTex.tx", naturally any name can be used for the texture file. In this way, each frame merely over-writes the previously used texture file and thus saves disk space.