An Improved Camera

Overview

In this section the characteristics of a virtual camera will be further refined to include,

- depth of field, and
- motion blur.

In the previous chapters, RIB scripts have typically defined a camera with the following statements,

```
Display "mypicture" "framebuffer" "rgba"
Projection "perspective" "fov" 40
Format 800 800 1
```

Such a camera, however, only approximates the behaviour of a real camera. In particular it acts like a pin-hole camera in that it sharply focuses all parts of an image. Real cameras are strictly limited in their ability to simultaneously focus the images of objects placed at different distances in front of the lens – this is a physical limitation that depends solely on the ratio of the diameter of the aperture of the camera's lens to its focal length. In photography this ratio is called the "f-stop" or "f-number" of a lens. This optical limitation is aesthetically exploited by photographers when the foreground and/or the background of a composition is deliberately defocused.

Real camera's are likewise unable to sharply focus on objects in a scene if either the camera is moving, particularly if it is hand-held, or if the scene, or parts of it, are in motion. To accomodate what, in computer graphics, is called "motion blur" the specification of a synthetic camera must include the idea of an image being formed over a finite period of time ie. as if using a shutter. The 'basic' virtual camera used so far has assumed all motion is frozen by capturing an instantaneous representation of a synthetic scene.

If photo-realism is to be achieved in the digital domain then similiar facilities should be available to 3D computer graphics. By providing mechanisms to introduce, what are in effect, digital versions of "depth of field" and "motion blur", RenderMan greatly improves a designers ability to deliver photo-realistic imagery. Interestingly, some real world photographic nuances are not available in RenderMan, these include effects caused by,

- lens flare,
- optical defects such as astigmatism and chromatic aberration, and
- the mundane effects of dirty or scratched optics!

Because there is a slight difference in the way "field of vision" (fov) is defined in photography compared to RenderMan, this section also provides some examples of converting from one system to the other.

Depth of Field

The RIB statement shown below in bold italics have been inserted to demonstrate the way in which a virtual camera can form an image of a scene as if it had, like a “real”camera, a limited depth of field. The statement that sets the depth of field requires three parameters,

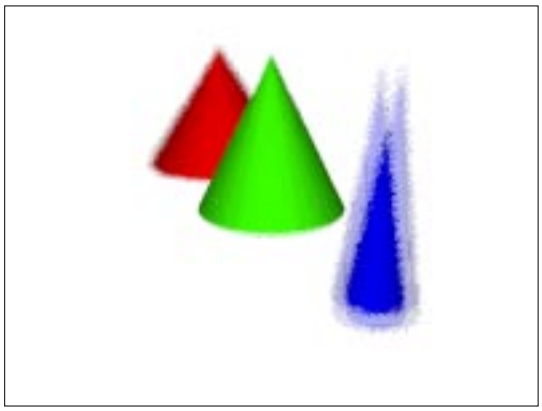
DepthOfField f-stop focal length focal distance

Since the field of view ("fov") has been previously specified in the RIB script (Projection "perspective" "fov" 45) it is best to use the default value of 1.0 for the focal length; RenderMan will then preserve the relationship between the height of the viewing frame, or the width if that is narrower than the height, and the field of view ie. the image scale remains constant.

Deliberately altering the focal length without making a corresponding change to the fov is like a photographer trying to control the sharpness of an image by changing both the lens AND the format of the camera on which it is mounted – a very bizarre thing to do!

RIB script

```
Display "fuzzy" "framebuffer" "rgba"
Projection "perspective" "fov" 45
Format 400 300 1
DepthOfField 2.0  1.0   5.0

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

WorldBegin
  Color 0 1 0    #green cone
  Cone 1.5 0.75 360

  Color 1 0 0    #red cone
  Translate -2.0 0.0 0.0
  Cone 1.5 1.0 360

  Color 0 0 1    #blue cone
  Translate 4.0 0.0 0.0
  Cone 1.5 0.25 360
WorldEnd
```

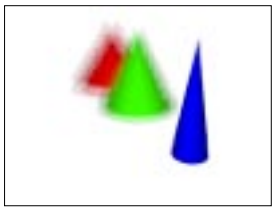


Depth of field set to

f-stop	f2.0
focal length	1.0 units
focal distance	5.0 units

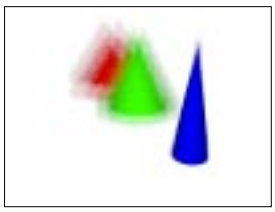
The visual effect of setting other combinations of f-stop and focal distance are shown on the next page.

Depth of Field – continued



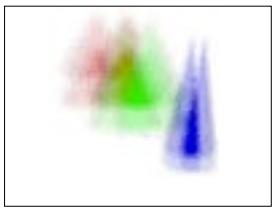
DepthOfField 2.0 1.0 3.0

ie. focus on the near cone using a moderately wide aperature lens.



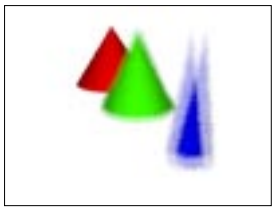
DepthOfField 1.4 1.0 3.0

ie. focus on the near cone using a very wide aperature lens.



DepthOfField 1.4 1.0 2.0

ie. focus 1 unit infront of the near cone using a very wide aperature lens.



DepthOfField 2.0 1.0 7.0

ie. focus on the far cone using a moderately wide aperature lens.

Motion blur

The role of a virtual camera in photo-realistic rendering is to replicate the most important attributes of a real camera. For example an image recorded onto normal film stock or an electronic imaging device such as a CCD, is formed over a brief period of time as the result of a shutter opening and closing. Any motion occuring during this period of exposure results in the image, or parts of it, being blurred.

Since the location of every object in a virtual world, including the camera, is specified by one or more transformations, RenderMan simulates the effect of motion blur by allowing any transformation, or indeed any "physical" dimension to have several values. The first value is applied when the virtual "shutter" opens and the last one is applied when it closes. The opening and closing times of the shutter are set using the following statement,

```
Shutter   openTime     closeTime
```

Normally, openTime will be set to 0 and closeTime will be 1.

Because our RIB files are hand-written, motion blur will be controlled using just two values for any particular dimension or transformation that needs to exhibit motion. Note that attributes such as colour and shading cannot be changed during the opening and closing of the shutter. In the case of a transformation such as a rotation, motion blur would be specified as follows,

```
MotionBegin [0   1]
   Rotation  70  0 0 1    #Transformation at openTime
   Rotation  85  0 0 1    #Transformation at closeTime
MotionEnd
```

In the case of the dimension of an object being varied, such as the height of a cone, the MotionBegin/MotionEnd block might look like this,

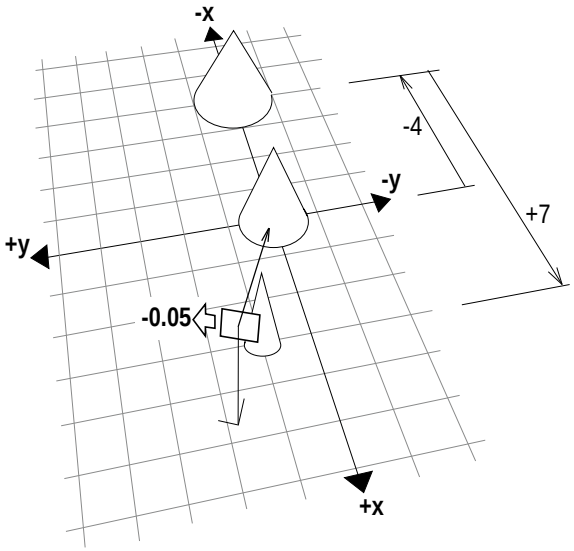
```
MotionBegin [0   1]
   Cone 1.5  0.25  360    #Cone pointing up at openTime
   Cone -1.5  0.25 360    #Cone pointing down at closeTime
MotionEnd
```

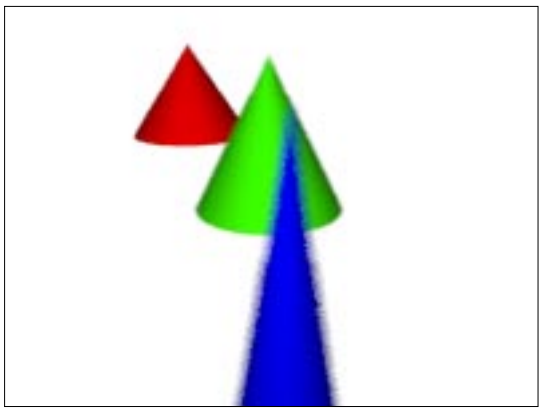
In the example shown on the next page, motion statements have been used to introduce blur by moving the world 0.05 units from right to left parallel to the x-axis of the camera as the shutter opens and closes from time 0.0 to time 1.0. The units of time, like those of the x, y and z axes, do not refer to any particular system of measurement – they could represent seconds, hours or days.

Motion blur – continued

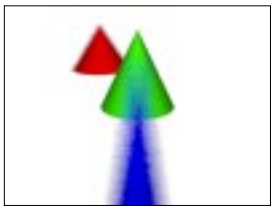
RIB script

```
Display "camera motion" "framebuffer" "rgba"
Projection "perspective" "fov" 45
Format 400 300 1
Shutter  0   1

MotionBegin [0 1]
  Translate 0 0 5
  Translate -0.05 0 5
MotionEnd
Rotate -110 1 0 0
Rotate 70 0 0 1

WorldBegin
  Color 0 1 0    #green cone
  Cone 1.5 0.75 360

  Color 1 0 0    #red cone
  Translate -4.0 0.0 0.0
  Cone 1.5 1.0 360

  Color 0 0 1   #blue cone
  Translate 7.0 1.0 0.0
  Cone 1.5 0.25 360
WorldEnd
```

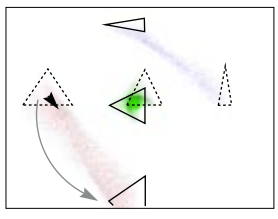




In the second example, motion blur has been achieved by rotating the camera 3° around the z-axis. Since the green cone is situated at the origin, and is itself symmetrical, it remains sharply focused. Question: why in the third example has a **linear** blur been the result of rotating the camera 90° around the z–axis?



```
Translate 0 0 5
Rotate -110 1 0 0
MotionBegin [0 1]
  Rotate 70 0 0 1
  Rotate 73 0 0 1
MotionEnd
```



```
Translate 0 0 10
MotionBegin [0 1]
  Rotate 0   0 0 1
  Rotate 90 0 0 1
MotionEnd
Rotate -90 1 0 0
```

notice the linear blurring although a rotation was specified?

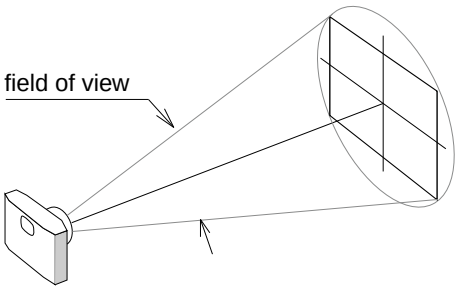
Field of View

In photography, the **field of view** (fov) of a lens, somewhat like the cone of vision of an eye, indicates the extent to which objects that are situated away from the direct line of sight (ie. the optical axis) can be focused on the film plane. However, photographers generally relate to the creative possibilites of a particular lens not in terms of its angular field of view – measured in degree's – but rather in terms of more general labels such as "wide-angle", "telephoto" etc. that are themselves based on the **focal length** of the lens.

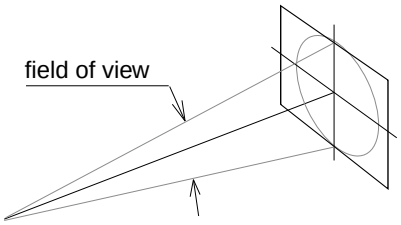
In real world photography it is convenient to change the field of view either by selecting a lens with a different focal length, or as in the case of a zoom lens, by directly altering its focal length. When setting up a virtual camera with RenderMan it is more convenient to accept the default focal length (1.0 unit) and directly adjust the fov parameter using the Projection statement, for example,

```
Projection "perspective" "fov" 40
```

Unfortunately fov is measured slightly differently by RenderMan compared to normal photography. In photography fov is measured across the diagonal of the film plane, while in RenderMan it is measured across the narrower of the two sides of the image. It is particularly important to take this difference into account when trying to match the view of a virtual camera to that of a real camera, for example, when compositing images from photography and computer graphics.



"fov" in photography



"fov" in RenderMan