

# 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 accommodate 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 similar 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,

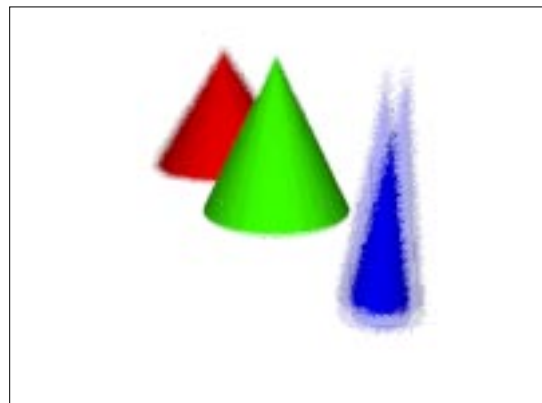
```
DepthOfField f-stop focal length focal distance
```

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

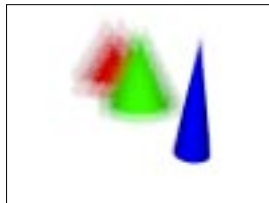
The visual effect of setting other combinations of f-stop and focal distance are shown on the next page.

## Depth of Field – continued



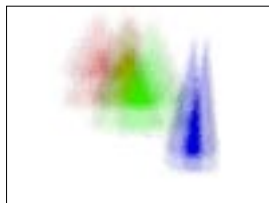
*DepthOfField 2.0 1.0 3.0*

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



*DepthOfField 1.4 1.0 3.0*

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



*DepthOfField 1.4 1.0 2.0*

*ie. focus 1 unit in front of the near cone using a very wide aperture lens.*



*DepthOfField 2.0 1.0 7.0*

*ie. focus on the far cone using a moderately wide aperture 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 occurring 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

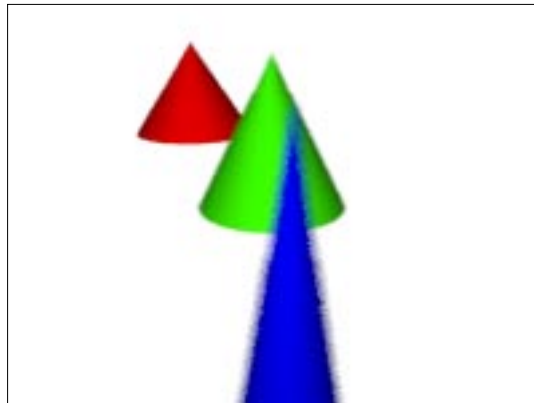
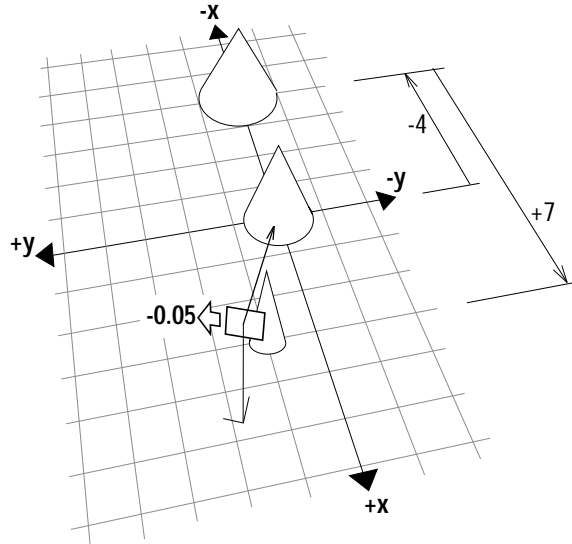
```
Display "camera motion" "framebuffer" "rgba"
Projection "perspective" "fov" 45
Format 400 300 1
Shutter 0 1
```

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

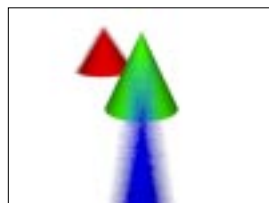
```
WorldBegin
  Color 0 1 0 #green cone
  Cone 1.5 0.75 360

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

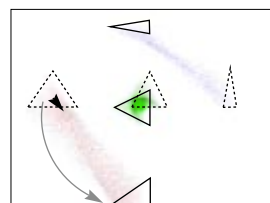
  Color 0 0 1 #blue cone
  Translate 7.0 1.0 0.0
  Cone 1.5 0.25 360
WorldEnd
```



In the second example, motion blur has been achieved by rotating the camera  $3^\circ$  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  $90^\circ$  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?

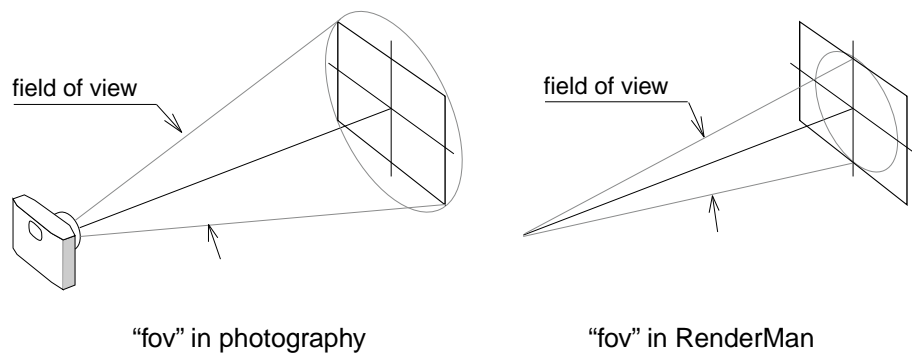
## Field of View

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 possibilities 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.



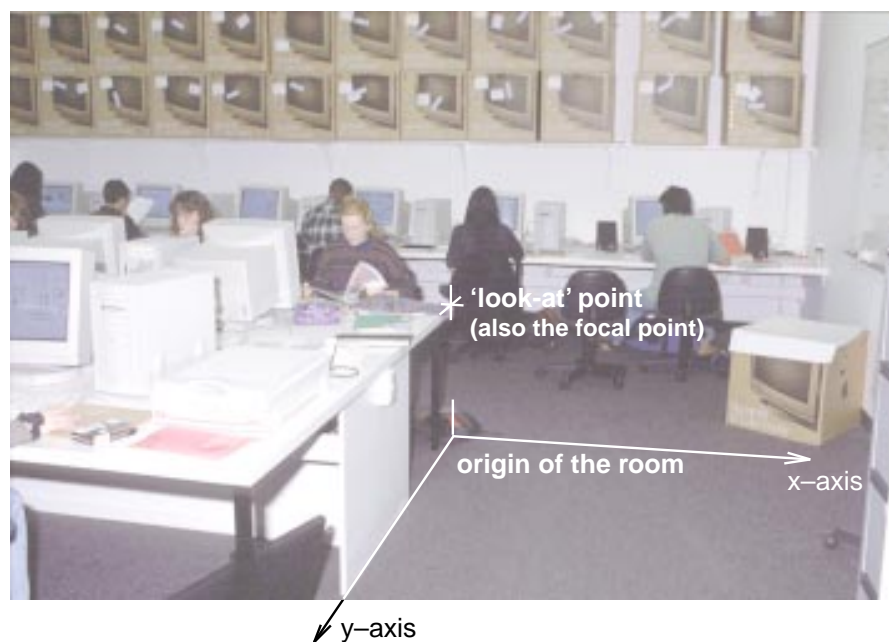
## Matching a VR Camera to a Real Camera

This section provides a general over-view of the technique of compositing images generated by 3D computer graphics and photography.

When photographing the scene that will be used as a “background” the following information must be recorded:

- location of the camera – its x, y and z coordinates from a known datum,
- location of the ‘look-at’ point – its coordinates relative to the same datum,
- focal length of the lens,
- f-stop of the lens,
- focal distance – not always the same as the ‘look-at’ point,
- general position, orientation and intensity of the light sources, and the position and dimensions of any surfaces on, or against which any virtual 3D objects will be placed.

In the example shown below, the origin of the room ie. the datum, was defined as a point on the floor directly beneath the far corner of the computer bench. Since the camera was positioned to ‘look-at’ the same point, the corner of the bench is in the centre of the frame of the photograph.



The following measurements were made relative to the origin:

- camera position 940mm, 4510mm and 1520mm above the floor,
- ‘look-at’ point 730mm directly above the origin,
- lens focal length, 35mm wide angle,
- lens f-stop, f4,
- focal distance; the lens was accurately focused on the corner of the bench,
- the scene was principally lit by a flash gun 150mm to the left, and 150mm above the lens,
- the computer bench is 2440mm wide.

## Matching Camera's – continued

On the basis of those measurements the following RIB script was written in order to **test** the accuracy with which objects, such as cylinders for example, could be placed in a virtual space equivalent to the original room. The locations chosen were,

the origin,  
the 'look-at' point and  
the leading corner of the computer bench.

It was assumed the x-axis of the world space extended from left to right across the room.

### RIB

```
#room.rib
Projection "perspective" "fov" 35
Display "untitled" "framebuffer" "rgba"
Format 1183 787 1

ObjectBegin 1
  Cylinder 4 0 100 360
ObjectEnd

LightSource "distantlight" 2 "intensity" 3
"from" [-150 150 0] "to" [-150 150 100]

#Camera from [940.0 4510.0 1520.0]
#Camera to [0.0 0.0 730.0]
Rotate -99.7 1 0 0
Rotate 191.8 0 0 1
Translate -940.0 -4510.0 -1520.0

WorldBegin
  LightSource "ambientlight" 1 "intensity" 0.1
  Surface "plastic"
  Color 1 0 0
  ObjectInstance 1 #cylinder at the origin

  Color 0 1 0
  Translate 0 0 730
  ObjectInstance 1 #cylinder at the lookat point

  Color 0 0 1
  Translate 0 2440 0
  ObjectInstance 1 #cylinder at the front edge of the bench
WorldEnd
```

In particular note the field of view has been set to 35° rather than the 'proper' value of 37.8° as shown on the chart on the next page. Because the actual focal length of a camera lens often deviates from its notional focal length it is always necessary to experiment with this factor until the 'test objects' apparently lie in the correct places when the rendered image is overlaid onto the photograph.

## Matching Camera's – continued

### Comparisons of “fov” used in 35mm photography and RenderMan *(Format must set the viewing frame to a ratio of 3:2)*

focal length	photographic fov	RenderMan fov
17	104	70.4
24	84	53.1
28	75	46.4
<b>35</b>	<b>64</b>	<b>37.8</b>
50	47	26.9
70	34	19.5
135	18	10.1
200	12	6.9