

Getting Started

Overview

The RIB files in this section are intended to guide you through the basics of working with RenderMan. Each example has been carefully chosen to introduce a broad selection of concepts relating to 3D computer graphics. The explanations accompanying each example are quite brief and are only intended to touch upon the ideas being presented. Don't worry if the material looks terribly confusing. As the course unfolds, the principles underpinning the concepts will be reiterated and illustrated many times over.

When a technical term is used for the first time it is printed in italics. You should make every effort to understand its meaning before continuing with the next section, "Shaping Up – Library Objects and Polygons".

At the conclusion of this section you will be able to

- write, save and send a simple scene description to PRMAN,
- set the basic characteristics of a virtual camera,
- use the basic transformations ie. translation, rotation and scaling,
- distinguish parameters from RIB statements,
- differentiate world space from camera space,
- understand the role of default settings.

Example 1

RIB

```
#ortho disk1.RIB
#using a default camera
WorldBegin
  Disk 1 0.5 360
WorldEnd
```

The purpose of this RIB file is to present a minimal scene to PRMAN and to introduce the basics of interacting with the scripting and rendering environment.

The first two lines show the use of the hash symbol # to indicate these lines are comments and must be ignored by the renderer. Comments can be included anywhere in a RIB file - they are the equivalent of post-it notices.

WorldBegin is a RIB statement and as such must be spelled exactly as shown ie. a single word with two capitalisations. Essentially it notifies RenderMan that objects comprising a scene description—a *virtual world*—are about to be defined.

Disk is a RIB statement that defines, by the three *parameters* (numbers) that follow it, a flat circular disk situated 1 unit along the z axis, 0.5 units in radius and a full 360 degrees in extent. Approximately, half the RIB statements (or commands as they will be referred to) you will use in this course require parameters. In all cases each parameter must be separated by at least one space. They may, however, be spread over several lines of text and have comments at the end of each line, for example,

```
Disk
  1    #unit along the z axis
  0.5  #units in radius
  360  #degrees
```

Finally, WorldEnd indicates the description of the scene, or world, has been completed. This small RIB file is interesting not just for what it describes but also for what it omits. Although it does not specify the characteristics of a virtual camera to view the scene ie. its location and orientation, or the surface colour and material characteristics of the disk, or how it is lit, the renderer is able to produce an image because, in the absence of specific information, it makes several assumptions and uses a number of *default* settings. In particular, RenderMan has provided

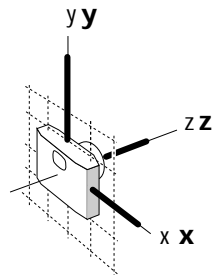
- an *orthographic* view looking along the z axis with the camera and the world sharing a common *origin*,
- an image called Untitled measuring 320x240 pixels,
- a matte white surface for the disk that does not require external lighting.

Visualizing example 1

RIB

```
#ortho disk1.RIB  
#using a default camera
```

two comments about the scene, the first gives the name of the file and the second is a brief note about the scene



the default camera creates an orthographic view, 320x240 pixels, with a name supplied by RenderApp

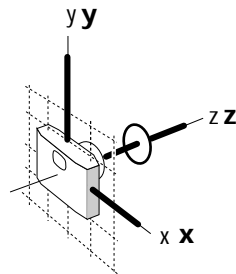
initially the origins of the camera and world coincide

WorldBegin

begin describing the world

```
Disk 1 0.5 360
```

create a disk situated 1 unit along the z axis, 0.5 units in radius and 360 degrees in circumference



WorldEnd

description of the world complete

Example 2

RIB

```
#perspective disk.RIB
#setting a perspective view
Projection "perspective" "fov" 40
WorldBegin
  Translate 0 0 3
  Disk 0 0.5 360
WorldEnd
```

The purpose of this file is to show the way in which a virtual camera using *perspective* projection can be set up before the world is defined and also to introduce the use of *translation* to move objects in a scene.

Before the world is defined the statement `Projection` establishes a perspective view with a *field of vision* of 40 degrees - this is one of several statements that control a virtual camera. Note that two of its three parameters are words. So that RenderMan does not attempt to interpret them as RIB statements or commands, textual parameters are always given in quotes ie. "".

As in the previous example, the scene consists of a single disk but this time the origin of the *coordinate system* has been moved 3 units along the z axis before the disk is defined. The `Translate` command has three parameters,

`Translate x y z`

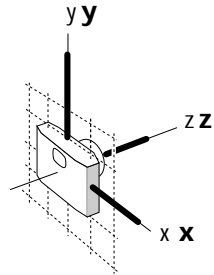
to move, what may be thought of as a three dimensional cursor around the *world space*.

Visualizing example 2

RIB

Projection "perspective" "fov" 40

set the camera to give a perspective view with a field of vision of 40 degrees, the size and name of the image are supplied by RenderApp



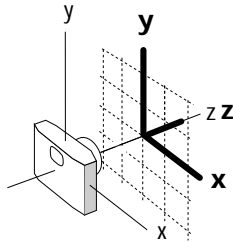
initially the origins of the camera and world coincide

WorldBegin

begin describing the world

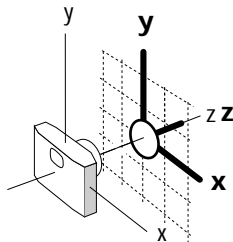
Translate 0 0 3

move the origin 3 units along the z axis



Disk 0 0.5 360

create a disk situated at the origin, 0.5 units in radius and 360 degrees in circumference



WorldEnd

description of the world complete

Example 3

RIB

```
#tube.RIB
#rotating an object
Projection "perspective" "fov" 40
WorldBegin
  Translate 0 0 5
  Rotate -120 1 0 0
  Color 1 0 0
  Cylinder 1 -1 1 360
WorldEnd
```

In this scene a cylinder is introduced into a world space that has, for the purpose of better viewing, been rotated and moved away from the camera. The cylinder has also been coloured.

The RIB statement `Cylinder`, with four parameters,

```
Cylinder radius depth height arc
```

shows how, like the disk in the previous example, an object from RenderMan's library of primitive shapes can be used in a scene. The cylinder and disk, as well as the other surfaces in the RenderMan library, will be dealt with in detail in the next section.

This file uses another type of *transformation*,

```
Rotate angle x y z
```

which in this instance turns the coordinate system 120 degrees anti-clockwise around the x axis BEFORE the origin is translated 5 units along the z axis of the world. Although the renderer reads the transformations in the order in which they appear, it postpones applying them until an object is *declared*, at which time it back-tracks and uses the transformations from last to first—like bullets in the magazine of a gun, the last one loaded is the first to be shot!

A cylinder is created within the redefined *world coordinate system*. Since the camera is fixed to the old world origin, the renderer produces an image looking slightly into the top of the cylinder. Using a fixed camera and trying to obtain a particular viewing angle by orientating an object in a scene is only adequate for simple compositions. In the next section the virtual camera is positioned relative to the world - much like a hand held camera in real photography.

`Color` (note the north American spelling) specifies a hue in terms of three *components*,

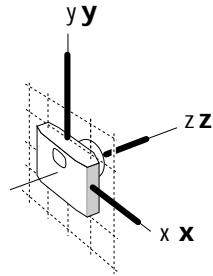
```
Color red green blue
```

A colour is applied to each object until another is declared in the RIB file.

Visualizing example 3

RIB

Projection "perspective" "fov" 40

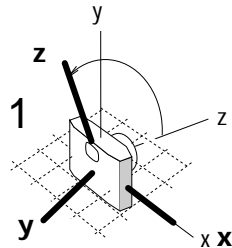


set the camera to give a perspective view with a field of vision of 40 degrees, the size and name of the image are supplied by RenderApp

initially the origins of the camera and world coincide

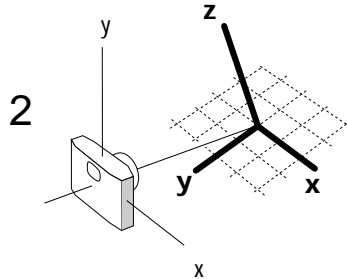
WorldBegin

```
Translate 0 0 5  
Rotate -120 1 0 0
```



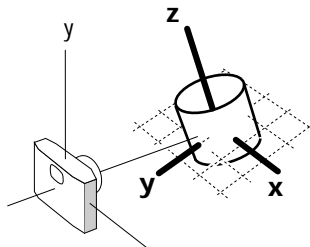
begin describing the world

the transformations are applied in reverse order; first an anti-clockwise rotation of 120 degrees around the x axis, followed by a translation of 5 units along the z axis



```
Cylinder 1 -1 1 360
```

create a cylinder, 1 unit in radius, with a base 1 unit below and a top 1 unit above the origin, 360 degrees in circumference



WorldEnd

description of the world complete

Example 4

RIB

```
#scaled tube.RIB
#scaling
Display "scaling" "framebuffer" "rgb"
Projection "perspective" "fov" 40
Format 200 150 1
WorldBegin
  Scale 0.3 0.3 0.3
  Translate 0 0 5
  Rotate -120 1 0 0
  Cylinder 1 -1 1 360
WorldEnd
```

This example introduces the idea of *scaling* the world space, and therefore, any objects placed in it. It also illustrates the way in which the characteristics of a virtual camera can be further refined and controlled.

Like the previous example, a cylinder is introduced into a world space that has been rotated and translated for better viewing. However, in this example the world space has also been uniformly reduced to 30% of its original scale.

In this and all future scenes, the RIB statements `Display` and `Format` are used to provide additional control over the imagery produced by the virtual camera. `Display` uses three parameters to specify

- the name of the image,
- where to put the image, and
- what information the image should contain.

`Format` uses three numeric parameters

```
Format image width image height pixel ratio
```

Although it appears first, `Scale` only takes effect after the rotation and translation have been applied - remember, transformations are applied in reverse order. The `Scale` statement uses three parameters,

```
Scale x y z
```

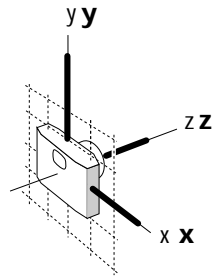
to enlarge or reduce a coordinate system along its x, y and z axes.

Visualizing example 4

RIB

```
Display "scaling" "framebuffer" "rgb"
Projection "perspective" "fov" 40
Format 200 150 1
```

set the camera to give a perspective view with a field of vision of 40 degrees, set the size of the image to 200x150 pixels storing rgb information

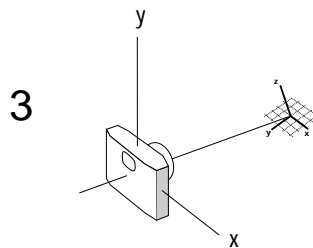
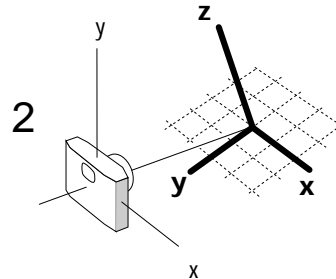
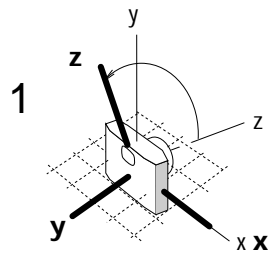


initially the origins of the camera and world coincide

WorldBegin

begin describing the world

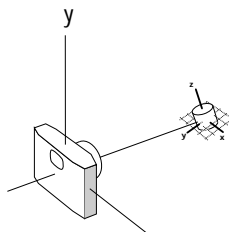
```
3 Scale 0.3 0.3 0.3
2 Translate 0 0 5
1 Rotate -120 1 0 0
```



the transformations are applied in reverse order; first an anti-clockwise rotation of 120 degrees around the x axis, then a translation of 5 units along the z axis, followed by the uniform scaling

```
Cylinder 1 -1 1 360
```

create a cylinder, 1 unit in radius, with a base 1 unit below and a top 1 unit above the origin, 360 degrees in circumference



WorldEnd

description of the world complete

Example 5

RIB

```
#goblet.RIB
#assembling an object
Display "goblet" "framebuffer" "rgb"
Projection "perspective" "fov" 40
Format 150 200 1

WorldBegin
  Scale 1 1 1          #change these to squash and stretch the goblet
  Translate 0 0 5
  Rotate -120 1 0 0

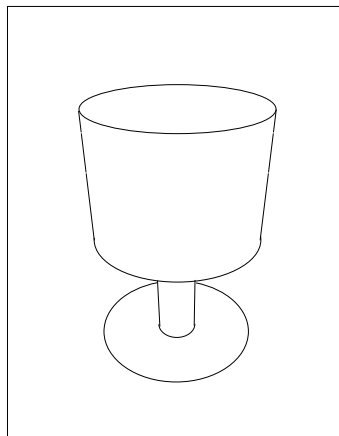
  Color 1.0 0.978 0.34 #gold
  Cylinder 1 0 1.5 360 #container
  Disk 0 1 360         #base of the container
  Cylinder 0.25 -1.5 0 360 #stem
  Disk -1.5 1 360      #base of the goblet
WorldEnd
```

In this example a number of basic library shapes are assembled into a simple goblet.

Color applies a uniform yellow hue to the entire goblet. Experiment with alternative colour schemes by introducing additional color statements between each part of the goblet. In particular, add another disk positioned a little below the rim of the goblet so that it appears to contain a coloured liquid.

The Scale statement, in effect, does nothing because it applies a uniform scaling factor of one. Change the scaling factor of each parameter to see how the goblet can be individually squashed and stretched in height, width and depth, for example,

```
Scale 1 2 1
```



Conclusion

By the time you finish the examples in this section and, no doubt, completed a few modifications of your own, you will have been introduced to many concepts, not only in 3D computer graphics in general, but also in the abstract world of RenderMan. This section concludes with a brief review of the *syntax* of RenderMan and an over-view of the structure of a RIB file.

Syntax

Twelve RIB statements or commands were used in “Getting Started” - by the conclusion of the course you will have dealt with approximately 35 of the entire range of 96 RIB commands. In addition to the hash symbol, the following statements:

- Projection/Display/Format – define a virtual camera,
- WorldBegin/WorldEnd – relate to the concept of a virtual world,
- Translate/Rotate/Scale – are examples of transformations, and finally
- Disk/Cylinder – insert library objects/surfaces into the world.

Incidentally, the words “statement” and “command” are used interchangeably. RIB statements form part a *language* recognised by the renderer. By human standards it is an impoverished language, but nonetheless, it is in its own right a complete system of communication. Some statements go together in pairs,

```
WorldBegin
WorldEnd
```

and bracket, what are called *blocks* of RIB. Other statements have words and/or numbers, called parameters, associated with them, for example,

```
Translate 0 0 5
Projection "perspective" "fov" 40
```

that, in the majority of cases, provide essential information without which the statement makes no sense.

Structure of a RIB file

At the beginning of a RIB file only a virtual camera exists, and therefore, all statements relate to it and to nothing else, for example,

```
Display...
Projection...
Format...
(anything else that is appropriate...)
```

As soon as the renderer ‘reads’ WorldBegin, the camera is ‘frozen’ and all subsequent statements effect the virtual world, for example,

```
WorldBegin
  Objects etc...
WorldEnd
```

and finally, WorldEnd marks the completion of the scene description.