Introduction

Interactivity v scripting

Most designers, especially those who are new to computer systems, assume the only way to work with a computer is to use interactive software. Indeed, graphical user interfaces (GUI's, pronounced goo-eez) are taken so much for granted that it may appear strange, if not bizarre, to reject the ease-of-use that such systems offer in favour of an environment based on **text** and **scripting**. What possible advantage could there be in using a keyboard rather than a mouse for graphical input? Why exchange pull-down menu's, floating windows, dialog boxes and icons for an unfamiliar way of making images that requires a large investment of time to master and that emphasises thought, care and perfect attention to detail? The answer to these questions lies principally in the nature of a GUI.

Illusions and interfaces

The problem with interactive software is that their interfaces are designed to hide the intricacies of the algorithms and techniques upon which they are based. Infact, just as a conjurer deceptively presents fiction as fact, GUI's organise their illusions around metaphors that routinely entice us to accept the impossible. For example, in illustration software such as Aldus FreeHand or Adobe Illustrator, users interact with elements of their artwork as if they are on **separate layers**. Even operating systems encourage users to perceive windows as being stacked and ordered into layers. Thus, windows can be moved to the front or sent to the 'back'. But the notion that an image on a computer screen can have depth, let alone be comprised of layers, is pure fiction. This course is intended to take you behind the illusions in order to more fully understand the principles of 3D modelling and rendering.

Working in the area of 3D computer graphics without a GUI involves communicating directly with a software package called a renderer. A renderer is somewhat like a laser printer but instead of turning a **2D page description**, normally in a computer language called PostScript, into a printed image, it accepts a **3D scene description** and converts, or renders, it as an image that is either viewed on the computer monitor, or saved as an image file. Because most renderers are embedded within an interactive modeller or animation system the ways in which they can be used are strictly limited by the 'host' software. Infact, the only people who can really 'get at the renderer' are the programmers who wrote the modelling or animation software!

RenderMan

Renderers also form part of software libraries used on high-end graphics workstations. But these require a knowledge of a programing language such as "C", and traditionally, artists and designers have not been given access to such skills. Fortunately, there is a renderer that supports the type of commun-ication that we require–**PRMAN** is part of the innovative **RenderMan** system developed by **PIXAR**. RenderMan is intended to support the production of photo-realistic images based on a 'mini language' called **RIB**–**R**enderMan **I**nterface **B**ytestream. The intention of RenderMan is to separate modelling from rendering. In formulating their scene description standard, PIXAR established a number of rules by which the characteristics of a virtual world, and a virtual camera to view that world, can be communicated to a renderer. Because RenderMan organises the way modellers can pass information to renderers, PIXAR refers to their system as an **interface**. Information about a 3D scene is written as text and is stored in a RIB file. Normally these files are produced by an interactive modelling or animation application and are rarely seen by a naive user of a computer system. However, because the details of the RenderMan Interface have been published by PIXAR, anyone with access to a word processor can write or edit a RIB file "by hand" and can gain greater control over the entire image making process. In this course you will use RenderMan to explore the fundamentals of photo-realistic 3D computer graphics.

What is a Script?

Scripts are used to convey information about a production or performance. The samples given below are examples of textural and symbolic scripts. What ever form it takes, a script typically enables an author to pass sufficient information about the structure of a performance so that it can be, in some sense, true or faithfull to the original design. To work effectively, a script must adhere to certain rules that are understood by the author and the performer. For example, it would be a disaster for an actor playing the role of King Henry to speak the lines given in italics, "Aumerle locks the door."

Richard II Act 5.3 – scripting a theatrical performance

Enter Bolingbroke, crowned King Henry, with
Harry Percy, and other nobles

AUMERLE *(rising)*
Then give me leave that I may turn the key,
That no man enter till my tale be done.

KING HENRY
Have thy desire.

Aumerle locks the door.
The Duke of York knocks at the door and crieth

YORK *(within)* My liege, beware! Look to thyself!
Thou hast a traitor in thy presence there.

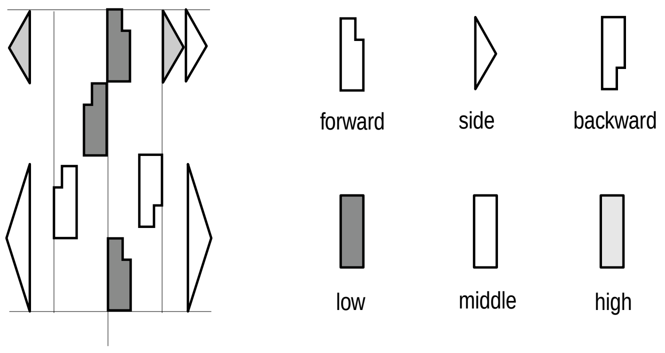
King Henry draws his sword

KING HENRY *(to Aumerle)* Villain, I'll make thee safe.

(NeXT Digital Press 1988)

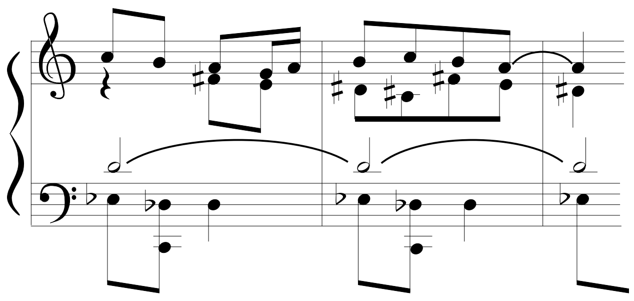
The scripting you will use in this course is no different to any other type of traditional scripting–you will be the author, PRMAN will be the performer and you will both conform to the rules defined by RenderMan.

The Labanotation System – scripting human movement



(The New Encyclopaedia Britannica vol 7 page 78)

From Three Pieces for String Quartet (No. 1) by Igor Stravinsky – notation for scripting music



(The New Encyclopaedia Britannica vol 24 page 530)

Why use scripting?

If scripting is so powerful it is appropriate to ask why interactive software is so popular? The answer lies in the breadth and flexibility of modern software design. In a production environment the majority of tasks a designer needs to address can be quickly and adequately tackled with interactive software. But for those who undertake innovative and experimental work, scripting of one kind or another, can offer significant advantages. At one end of the scale, scripting can mean writing an entire software package and at the other end it can mean writing so-called macro's for a spreadsheet. In an educational context, and more especially for a third level degree course, an investigative approach based on scripting means you will learn the general principles of 3D work rather than a single implementation. However, it should be recognized that RIB scripts (files) are NOT normally written **by hand**, but are usually produced by modelling and animation software and these can handle levels of modelling detail that would be impossible for any human to reproduce manually.