Preface

With the emergence of the Java 3D API, the creation of high quality 3D animated graphics for Java applications and applets has become a possibility. Being a high-level API based on OpenGL and DirectX, Java 3D allows developers to produce object-oriented graphics applications that are platform-independent. Numerous applications in fields ranging from business, science, medical to education have been implemented based on this technology. One well known example is the Maestro project, which allows users to navigate the 3D world of Mars from a desktop computer based on inputs from eight 360-degree cameras onboard the rover.

In one of our research projects in this area, we have used Java 3D to develop a Web-based real time 3D oscilloscope experimentation system, which has been launched at National University of Singapore. This application enables users to carry out a physical electronic experiment that involves the use of an actual oscilloscope, a signal generator, and a circuit board remotely through the Internet. Specifically, the control of the various instruments are carried out in real time through the use of a Java 3D based interface on the client side, with the results of the experiment being also reflected or displayed appropriately on 3D instruments in the same interface.

In this application, Java 3D is used to create a virtual 3D world or room in which the 3D instruments reside. The mouse is used for both navigation in this world as well as for operating the instruments through, say, dragging a sliding control or a rotary control or clicking or switching appropriate buttons on the instruments. Associated commands that cause the real instruments in a remote physical laboratory to operate accordingly are then sent through the Internet in real-time. Experimental results corresponding to, say, a change in the real oscilloscope display, are then sent from the instrument control server back to the Java 3D client to result in a real-time change in the display of the virtual 3D oscilloscope in the virtual 3D world.

Apart from the room and instrument geometry, three important and difficult issues that have been tackled are navigating behavior, collision detection and picking behavior. Specifically, navigating behavior controls how the user is able to walk around in the virtual laboratory as well as the positions and angles of the view platform, as when the user attempts to get a better view. The use of appropriate collision detection ensures that the user is not able to traverse any solid objects such as walls, tables and instruments, while a customized picking behavior is necessary for the user to adjust the controls on the instruments precisely.

To satisfy these requirements and noting that the users will not be familiar with the use of special keys for 3D navigation, a more sophisticated and customized navigating system has been designed and developed. In this system, navigation can be done by using either the mouse or the keyboard. Specifically, the position and direction of the view platform or viewpoint can be changed by simply using the mouse to press two specially designed groups of control objects, a navigating speed slider, a translation, and a rotation icon.

To change the user's "walking" speed through the 3D virtual laboratory, the navigating speed slider can be adjusted. This will change the delays used in the main processing steps of the navigating function. An icon with six straight arrows allows the user to move in a straight translational manner. Pressing a ball in the center of the icon will reset the viewpoint to its initial position. The other icon with four curved arrows allows the user to rotate around the current position. The ball in the center will reset the viewpoint to a horizontal one.

With 3D scene-based navigation and manipulation implemented, the system is able to provide a more realistic 3D feel to users who are conducting real-time Web-based experimentations. In the course of designing and developing this application, a large number of Java 3D example and program codes has been written, and an API library for the creation of similar Web-based 3D experiments has been developed. Specifically, the library includes a series of code segments and classes for defining the geometry and appearance of control buttons, knobs, sliders, clips and scope displays as well as their behavior in a 3D world.

This has culminated in the writing of this book, which aims to provide programmers with a simple but yet complete, comprehensive, and detailed coverage of all the important topics in Java 3D.

In particular, this book includes a large number of programming examples for the reader to master this graphics API to develop sophisticated Java 3D graphic programs. Specifically, the use and significance of keywords, syntax, classes, methods, and features that make up the API are illustrated with 300 figures, 200 code fragments, and 100 examples throughout the 450 pages of the book to provide an easy-to-read and easy-to-use learning experience.

All of the important Java 3D topics, including geometry, appearance, navigation, picking, animation, interaction, texture, light, background, fog, shade, input device, sound, and advanced view will be covered. Both novice and advanced graphics programmers, including those who know Java but do not have any background in computer graphics, will find the book useful from the large number of working examples provided. In addition, each chapter is written in a relatively independent manner so that readers with specific interests can make use of the examples in certain chapters without the need to read through other chapters.

In total, the book consists of 13 chapters covering the various topics, and is organized in a step-by-step style. Discussions on basic 3D graphics, Java 3D overview, 3D geometry, appearance, texturing, animation, and interaction are discussed in the first six chapters. Subsequently, more advanced topics on navigating, picking, input device and are explored. The use of more complicated multiple views and audio are then discussed, culminating in the last chapter, which presents the Web-based 3D experiment application in detail. The following gives a brief synopsis on each of the chapters.

Chapter I provides an overview of interactive 3D graphics, OpenGL, virtual reality, VRML, Java 3D and mixed reality. The main purpose is to give an outline on the relationship between these related technologies and applications. This also serves to place Java 3D in the appropriate context from the general perspective of 3D graphics creation and presentation.

Although many programming languages are available for creating 3D graphical applications, only Java 3D, VRML and the subsequently developed X3D are suitable for Web-based virtual reality development. As a result, while other tools are also briefly introduced, this chapter will discuss, analyze and compare VRML and Java 3D in detail. Subsequent chapters in this book will focus on various aspects of Java 3D with an aim to provide a comprehensive experience in terms of understanding and programming using Java 3D technology.

From the discussions in this chapter, the differences between VRML and Java 3D will be better appreciated. It will be pointed out that, as one of the two important development tools for Web-based virtual reality, Java 3D has established itself as an important modeling and rendering languages for more specialized applications that involve, for example, database accesses, customized behaviors and home use mobile devices such as PDA, mobile phone, and pocket PC.

Chapter II is a relatively short chapter laying the ground work for the creation of a virtual world in Java 3D. This chapter introduces the programming paradigm or the scene graph approach. Specifically, after providing some basic knowledge on VirtualUniverse, SimpleUniverse, Locale, BranchGroup, and TransformGroup objects, which form the virtual world framework, this chapter outlines how one can build a virtual world through specifying a scene graph.

The scene graph in Java 3D is for the purpose of describing the objects in a virtual 3D world, and is a tree like structure consisting of a hierarchy of nodes containing information on objects or groups of objects on geometries, shapes, lights, sounds, interactions, and so on. Specifically, the root of the scene graph is a virtual universe that may have several local branches. Also, each locale may hold related objects that are next to one another at a certain location in the 3D world, and may be made up of many branch and transform groups.

Each branch group is a subgraph of the scene graph, and can be compiled for rendering efficiency. Also, by setting certain capabilities, branch groups can be attached or removed for interaction with the user during run time. In addition to the content branch, which describes the visual objects in the virtual world, the scene graph also needs at least a viewing branch for describing the how the user views the 3D world. The setting up of this branch can be

carried out easily by invoking a simple universe. Alternatively, multiple views of the same virtual world can be obtained for applications involving multiple displays.

Chapter III focuses on creating shapes and 3D objects that can be rendered by Java 3D using both core and utility classes. Different approaches to object creation will be explored, helping programmers to construct complex shapes using simple building blocks.

In this chapter, several basic geometry classes that can be used to specify the geometry of visual objects in Java 3D will be introduced and discussed. Specifically, PointArray, LineArray, TriangleArray, and QuadArray are useful for building objects using a series of points, lines, triangles and quadrilaterals, while for structures where the series of lines or triangles are adjacent to each other in a certain manner, the use of LineStripArray, TriangleStripArray, and TriangleFanArray may be more convenient and lead to faster rendering.

The problem of requiring certain vertices to be repeated when these basic classes are used can be overcome through using their indexed versions, where the sequence of vertices can be supplied via some integer indexing arrays. Complex objects can also be created through appropriately combining objects built from different classes. Also, simple geometrical shapes such as boxes, spheres, cones or cylinders can be easily generated using some predefined utility classes in Java 3D.

In Chapter IV, the appearance of the created 3D objects is discussed, including some parameters that control how they will be presented to the user. Important appearance attributes are illustrated by using examples so that the effected changes can be better appreciated.

For most virtual reality or game applications, point, line and polygon are the basic primitives for constructing objects in the 3D world. The chapter therefore gives an in depth account of the various basic attribute settings, including rendering modes, visibilities, colors and material properties, that can be applied to these primitives.

Although extensive use of basic attributes such as color and material will be able to make an object realistic to the human user, the amount of programming codes needed will in general be very lengthy and time consuming to develop if the object has complicated geometry or appearance. As an example, to create an object with many color patterns on, say, a curve surface, many zones or strips may need to be individually defined using the appropriate color or material properties. Since this is time consuming, Java 3D allows the use of what is known as texturing and image mapping, which will be discussed in the next chapter.

Building on Chapter IV, Chapter V describes the technique of texture mapping to add realism to virtual scenes. The use of texture modes and attributes in Java 3D, which is relatively straightforward and effective for adding color and realistic details to the surface of a visual object, will be presented to give programmers a reasonable palette of texturing techniques with which to work on.

Specifically, texture objects are referenced by appearance objects, and have a variety of parameters that can be adapted to suit different needs through the Texture and TextureAttributes classes. The mapping of a texture image to a surface can be performed manually by using setTextureCoordinate to set texture coordinates. It can also be automatically carried

out through the TexCoordGeneration class. The application of multiple textures to a surface can give a very realistic visual effect on the visual objects created in the virtual universe.

Chapter VI explores other issues that lead to better environmental realism. These including lighting, fog, and background that can be used to further enhance the appearance of the virtual world. In general, these environmental factors affect the appearance of the object through their interaction with its material attribute.

Specifically, the use of ambient, directional, point and spot lights will be presented. Topics involving material and normal settings, which determine how light will be reflected, will also be discussed. Some examples on the use of linear and exponential fog to smooth a scene and to prevent the sudden appearance of distant objects so as to enhance its emotional appearance will be given. Then, the use of simple color, image, and geometry based backgrounds will be illustrated.

Chapter VII discusses the use of interpolators and alpha classes for object animation in the virtual world. Simple animated movements such as rotation, translation and their combinations will be covered. More advanced animation techniques such as scaling, transparency, and morphing will also be discussed. In addition, The billboard and the level of detail (LOD) classes, which are useful for creating animation at a reduced rendering level, will be presented.

The various animation classes provided by Java3D are usually quite complete in terms of their functionality. Very often, just a few parameters will be sufficient to implement a variety of simple and basic animation in Web-base virtual reality applications. For more complex scenarios, these classes can be further defined with more specific codes to give rise to more complicated movements.

The movements of objects in a 3D world are very often the result of the user manipulating these objects or just navigation through them. As an example, the animation that allows a 3D clock hand to turn may need to be re-initiated if the user presses a certain reset button in the 3D world. The issue of interactions is therefore closely related to animation and is the main concern of the next chapter.

To detect and deal with interactions from the user, Chapter VIII delves into some basic issues on event detection and processing. These include capturing the key pressed, mouse movement, finding changes in the state of the virtual object and time lapsed. In Java 3D, the detection of these events or detection conditions are based on examination of the appropriate components of the behavior class of an object.

Specifically, to specify and implement an interaction, it is necessary to make use of some special behaviors and events that Java 3D provides or to refine or customize these interaction functions. In general, through the construction of custom wakeup conditions and criteria, the system will be able to provide changes to the virtual 3D scene and objects through some appropriate processStimulus methods when the relevant stimulus or trigger condition is received. Complicated behavior can be handled by creating specialized wakeup triggers that respond to combinations of wakeup conditions, by having behaviors that post events, by detecting object collisions as well as the entry and exit of objects and viewing platforms into certain spatial bounds.

After giving a basic foundation of event detection and processing, the next two chapters provide a more advanced coverage of the topic in two important interaction scenarios. These correspond to the picking of objects and use navigation in the 3D world.

Chapter IX discusses the use of the picking behavior class for the purpose of picking objects of interest. Using simple utility classes such as PickRotationBehavior, PickTranslateBehavior, and PickZoomBehavior is straightforward, although the picking behavior may not be flexible enough for most applications.

In general, the simple operation of picking an object in the real world is actually very complicated and involves many senses. To allow the user to pick objects in the virtual 3D world as realistically as possible, Java 3D has a variety of picking shapes, such as PickRay, PickConeRay PickCylinder and PickBounds, that can be used to customize the picking behavior. After discussing these in some detail in this chapter, an application example involving the use of the controls in a 3D instrument panel will be provided.

Chapter X is on another important interaction behavior, that for the user to navigate or move in the virtual world. At the beginning of this chapter, the basic navigation classes provided by Java 3D are introduced. Due to the fact that they are not very flexible, these classes cannot be used for navigating in most virtual reality applications.

As a result, there is a need to make use of Java 3D utility classes as well as more specialized user-defined behavior classes for designing customized navigation behavior in many virtual reality applications. This chapter will discuss how rotation and translation matrices can be used for calculating the position and orientation of the objects as the viewpoint changes. The use of navigation tools for moving and turning with the help of keyboard, mouse, joystick, and other external devices will also be presented. In addition, another important issue, that involves the collisions of objects and how these can be handled, will be discussed in this chapter.

In Chapter XI, some advanced topics needed for generating multiple views of the virtual universe in Java 3D will be discussed. Illustrated with examples on configuring the viewing window to the virtual world, one will be able to see the virtual world from different perspectives, resulting in customizing viewpoints. Real life applications such as portal view in immersive virtual reality environment and video wall configuration will be introduced.

In Chapter XII, how 3D sound sources and aural characteristics can be integrated into the virtual world built using Java 3D will be outlined. Java 3D supports three types of sound sources, BackgroundSound, PointSound, and ConeSound, which will become audible if the activation radius intersects with the scheduling bounds of the sound. Controls can also be made available to turn a sound source on or off, set its gain, release style, continuous playback style, looping, priority, and scheduling bounds. In addition, by creating a SoundScape object with appropriate AuralAttributes, a special acoustical environment can be simulated.

In the last chapter, we provide some detailed design and discussions on an application where Java 3D is used in a Web-based real time 3D oscilloscope experimentation system. Outlined earlier, this application enables users to carry out a physical electronic experiment that involves the use of an actual oscilloscope, a signal generator, and a circuit board remotely through the Internet.

We are particularly thankful to Prof. Ben M. Chen, Dr. Xiang Xu, and Dr Lu Shijian for their kind help and assistance. We are also thankful to Ye Jiunn Yee, Yupinto Ngadiman, Nguyen Trung Chinh, Henky Jatmiko Gunawan, Ang Wee Ngee, Au Heng Cheong, Teo Sing Miang, Lee Chi Shan, Tam Sai Cheong, Thian Boon Sim, Subramanian S/O Annamalai, Cheong Yew Nam, Ho Chang Sheng Herman, Wu Sin Wah, Ng Chea Siang, Lim Tiong Ming, and Thulasy Suppiah, for their help and contribution in the testing and debugging of the various source codes. Last, but certainly not least, we would like to acknowledge the National University of Singapore and the Singapore Advanced Research and Education Network for providing us with research funds that lead to this book.

January 2008
Ko Chi Chung
Cheng Chang Dong
Singapore