Visualisation of Multi-Dimensional Simulation Models

Asuka Ishikawa

Master of Science

The University of Edinburgh

December 2005

To my parents,
Noritada and Atsuko
and my sister Eri

Abstract

In recent years, the rapid growth in computational power and the Internet bandwidth have made the creation of complex graphical computer simulations over the web possible. Accordingly, a variety of web-based simulation libraries that exploit 2D graphics have emerged.

However, despite its visual impressiveness and high potentiality in the field of computer simulation and visualisation, a simulation language that uses 3D graphics has not yet been produced.

Conventional wisdom dictates that performing computer simulations with 3D graphics on the web is computationally too overwhelming and hence very slow. However, in light of the huge improvement in 3D graphics technology, which is becoming so accessible and familiar in this day and age, such an allegation is questionable.

This project addresses the possibility and potentiality of a web-based simulation library with 3D visualisation and attempts to provide a method of implementation of such a simulation library.

Acknowledgements

First, and foremost, I wish to express my gratitude to my supervisor Prof. Roland N. Ibbett for his invaluable guidance and support.

I also want to thank Dr. Murray Cole for giving me an opportunity to demonstrate my project in front of his students.

Finally, I would like to thank my family and friends for all their help and support.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Asuka Ishikawa*)

Table of Contents

Chapter 1

Introduction

1.1 Introduction

Computer Simulation is a powerful and flexible methodology that encapsulates the dynamic characteristics of a real world system and converts them into a form of computer model. Many industries rely on computer simulation in order to ensure product quality, safety, stability and credibility. Computer simulation is particularly effective when analysing problems with no closed form-solutions.

Following the introduction of Java, the platform-independent language developed by Sun Microsystems, there have been several web-based simulation libraries emerging that permit the creation of simulation programs as Java application and applets. Recent tools include SimKit, JavaSim, JSIM and SimJava. [27] [28] [30] These simulation libraries accompany visualisation with animated two-dimensional images that illustrate the state of simulation. A recent survey carried out by Naps and an analysis of twenty-one experiments performed by Hundhausen provide evidence to suggest that visualisation effectively helps learners to understand complex concepts behind simulation models. [2] However, these two-dimensional simulation libraries generate the superficial aspect of simulation models that can only be viewed from a single perspective. Thus, there is a high demand for simulation libraries that are capable of capturing models that involve complex topology and provide facilities to examine the models from whatever perspective that users desire.

At the same time, there have been spectacular changes in the world of computing both in software and hardware. One of the most noteworthy changes, which is due to the drastic improvements in CPU power and the increasing affordability of graphics hardware, is the fact that three-dimensional graphics contents have become ever more pervasive in our daily life. Entertainment is arguably the most common way in which people today are exposed to three-dimensional graphics. With respect to software, several established three-dimensional graphics APIs and file formats have emerged. Among them, most notably, Java3D, the Java-based three dimensional graphics API developed by Sun Microsystems, has particularly gained attention above all others. [64] Its characteristic of being platform-independent, and able to perform fast three-dimensional graphics rendering over the Internet without much performance loss has had a great impact on graphical application developers. The technology has been exploited in a wide range of areas including education and training, research and commerce. Java3D has undoubtedly become a de facto standard for web-based platform-independent three-dimensional graphics API.

Tradition dictates that simulations with three dimensional graphics over the web are somewhat overwhelming and result in serious performance degradation. However, after witnessing the present euphoria surrounding Java3D and the drastic improvement of graphics hardware and of Internet bandwidth, the marriage of a simulation language such as SimJava and Java3D to produce a web-based three dimensional simulation library seems naturally conceivable. Despite the two-dimensional simulation libraries being static and only able to be experienced from a single perspective, SimJava, together with Java3D, allows simulation models to be visualised from whatever angle the end user desires and in many ways that images and text simply do not allow.

Therefore, this project investigates the feasibility and potentiality of three-dimensional graphics applications on the web and embarks on the development of the world's first platform-independent three-dimensional simulation language.

1.2 Project Structure

Needless to say, whether in freeware or shareware, the software market has always been volatile. Every day, new software emerges and existing software becomes extinct. Software survival has become ever more important in software development. Key issues include good support, intuitive graphical user interface, performance optimisation, strategic software testing and project management.

In the case of Hase++, the language comes with HASE, a simulation environment for building simulation models written in Hase++. [33] [36] SimJava was based on Hase++. The simulation environment not only eases the process of building simulation models, but also significantly enhances users' understanding of the language with its intuitive GUI. As a result, the language and the simulation environment are widely used in many industries and research fields. From this experience, it was thought to be crucial to develop a simulation environment for SimJava3 in order to prevent the language from dying out.

Once the language and the simulation environment are complete, it is important to provide some simple example models that are accessible on the web. These allow users to examine what the language is capable of and enhance the understanding of the language.

Thus, the project primarily consists of three parts: the three-dimensional simulation language, the simulation environment, and some example models using the first two.

1.3 Main project aims

- Like many applications using three-dimensional visualisation, visualisation generation takes substantially longer and long response times result in a bad graphical user interface. Therefore, performance optimisation of the visualisation generation becomes one of the top priorities. In order to address this issue, careful and systematic software design and testing needs to be carried out.

- As well as becoming acquainted with visualisation and simulation issues, and further developing technical skills, one of the prime objectives of this project is to acquire some of the proven project-management and testing skills that are considered vital in the real world industries today. As the project was deemed to require a considerable amount of implementation work, such techniques seemed to be even more important. Therefore, it was important to take this opportunity to use some of the well-established software-engineering and testing techniques that have been applied to this project and further stress the importance of them in both academic and commercial IT discipline.

- From the end-user's perspective, the project should have a well-designed graphical user interface. This is an important issue as it crucially influences the software's usability.

- The project should have a high degree of openness. This will lead to a good prospect of future development of the project.

1.4 Thesis Structure

Before proceeding to the more detailed discussions of this project, this section briefly describes the structure of this report.

Chapter two provides background materials that are relevant to this project. The chapter begins with reviews of the latest three-dimensional graphics technologies. Followed by this, we evaluate some of the web-based simulation tools and a simulation environment that are closely related to the project. The rest of the chapter outlines other materials that are relevant to this project

Chapters three, four and five cover the design and implementation of the project. Chapter three describes the design and implementation of the three-dimensional web-based simulation library. Chapter four describes the development of the simulation environment using the results of Chapter three. Chapter five provides example simulation models using the results of Chapters three and four.

Chapter 6 describes issues concerning project management and the testing procedure.

Chapter 7 serves as an evaluation of this project. The problems encountered during the course of this project and their solutions are presented.

Chapter 8 summarises the project and discusses several of the achievements and deficiencies identified.

Chapter2
Background

2.1 Three-dimensional Graphics Technology

This section reviews some of the cutting-edge three-dimensional graphics technologies that were considered as candidate technologies for the next generation of SimJava. Justification and reasoning regarding the selection of the technology are discussed in the following chapter.

2.1.1 Java3D

Java3D is a low-level three-dimensional scene graph-based graphic programming API for the Java platform developed by Sun Microsystems. [60] It was first introduced by Henry Sowizral at the annual JavaOne developers conference. [64] Today, Java3D is used in a wide range of applications such as mechanical computer-aided design (MCAD), scientific visualisation, product visualisation, collaborating engineering, eCommerce solutions, and computer games.

Java3D is essentially the Java interface for interactive three-dimensional graphics. In other words, it is an extension to the core Java APIs. Hence, Java3D inherits the advantages of Java, most notably being platform-independent. Java3D shields the developers from platform-specific details. This allows Java3D applets and applications to run on computers with different operating systems featuring different low-level graphics API or different graphics hardware.

Java3D is a part of the Java Media family, a suite of media-centric Java APIs that allow developers to create complex multimedia programs. Therefore, programmers need to import the class libraries under javax,media.j3d, the top level package and the utility classes provided by javax.vecmath in order to make use of the technology.

The technology allows graphics programmers to create comprehensive and platform-independent applets and applications that feature interactive and animated three-dimensional graphics with three-dimensional specialised sound capabilities. In contrast to many multimedia APIs, Java3D is high-level platform-independent three-dimensional graphics API. This makes the process of creating animated three-dimensional contents quick and easy without sacrificing much of the performance. The API provides scene graph compilation and other optimisation techniques.

In Java3D, a scene is described as a scene graph. [10] A scene graph is a tree-like data

structure used to store, organise and render three-dimensional scene information. The scene graph mechanism was designed for scalability and enables high levels of optimisation and multiprocessor rendering. Moreover, Java3D supports binary compression of specific geometric objects. This allows developers to reduce the bandwidth required to transmit geometric data over a network with little loss in the object quality.

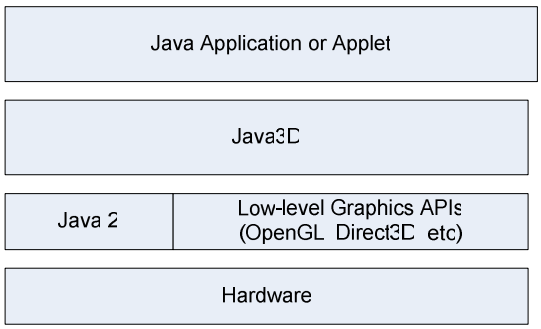

Fig. 2.1 Java3D API Layer

As shown in Figure 2.1, Java3D was designed to be a layer on top of low-level graphics APIs such as OpenGL and Direct3D, to take advantage of hardware acceleration. [66] This structure conceals low-level rendering details so that programmers can concentrate on scene composition.

At the moment, the latest Java3D version is 1.3 and it is now considered to be a full-featured graphics API. In order to view Java3D contents, Java3D has to be run on Java 1.2 or later. Furthermore, viewers need to have the Java3D plug-in installed into their web browsers. For UNIX users, Java3D is implemented on top of OpenGL. For Win32 users, Java3D is available for OpenGL and Direct3D. [64]

2.1.2 OpenGL

OpenGL is a cross-language, cross-platform and vendor-independent API for the development of two-dimensional and three-dimensional graphics applications. The API contains over 250 function calls that make it possible to create complex three-dimensional models from simple primitives. [67] Today, the technology is widely used in many industries and research fields including Computer Aided Design (CAD), virtual reality, scientific visualisation, information visualisation and video games.

Prior to the introduction of OpenGL, many hardware vendors had different graphics libraries. This situation made the process of software development very costly and time-consuming. Hence, OpenGL was introduced to serve as the standard for computer graphics technology

OpenGL is mainly based on earlier work on the SGI IRIS GL library. [5] Essentially, it is a specification and a document stating a set of functions and precise behaviours that they must perform. It is a role of hardware vendors to provide the implementations of the API and therefore software developers can merely focus on how to use the functions. In general, these implementations attempt to take advantage of the capabilities of the hardware in order to achieve a high degree of efficiency. One of the problematic restrictions of its predecessor, IRIS GL, was that it only provided access to features supported by the underlying hardware. Therefore, if the hardware did not provide a feature then the application could not use it. However, OpenGL overcomes this problem by emulating unsupported functionality through software. It is therefore possible for applications running on low-power computer to use advanced graphics.

OpenGL is a free-lance product managed by the OpenGL Architecture Review Board which was formed in 1992. [45] OpenGL allows vendors to provide additional functionality as extensions. The most notable extension which may concern this project is the OpenGL Performer Library which makes the creation of real-time visual simulation application possible.

On the 6th of August, 2005, OpenGL revealed the Microsoft's intentions for supporting OpenGL in Windows Vista, the latest version of its operating system. [52] According to the report, Microsoft is currently planning to layer OpenGL on top of Direct3D. This implies a significant reduction of OpenGL's performance on new Windows machines. It is suspected that the performance degrades as much as 50%. Moreover, the implementation further restricts OpenGL on Windows machines by only allowing version 1.4 of OpenGL, and no extension to it will be possible.

2.1.3 VRML

Virtual Reality Modelling Language (VRML) is a file format that represents three-dimensional interactive vector graphics. [19] It was primarily designed to display interactive three-dimensional graphics on the World Wide Web. The file content can be viewed in the main web browsers with the VRML plug-in installed. The main VRML plug-ins used today are Cosmo Player and Cortona.

VRML was first introduced in November 1994. [16] It was based on the API and the file format of Open Inventor Software Component developed by Silicon Graphics, Inc. In short, VRML is a text format with .wrl extension that contains a number of tagged codes which specify vertices and edges for a three-dimensional polygon along with the surface colour, image-mapped textures, shininess and transparency. In order to gain more control over three-dimensional models, it is possible to embed a short program written in Java or JavaScript (ECMAScript) into a VRML file. The integrated time definition makes the construction of animated objects easy. Moreover, it is possible to incorporate URLs into each of the graphics components so that when they are clicked, a web browser will fetch a web-page or a new VRML file. In general, nearly all the three-dimensional graphics editors can save three-dimensional models in VRML format. The file format is frequently used as an interchange of three-dimensional models from CAD systems.

Despite the huge improvement in real-time three-dimensional graphics, VRML capability has remained essentially the same since the release of its version 2 in 1997. Currently, the latest version of VRML is VRML97 but it has now been superseded by X3D. Nevertheless, the technology is still widely used, particularly in the areas of education and research.

2.1.4 DirectX

DirectX is a set of multimedia application programming interfaces developed by Microsoft for Windows machines. The API comprises various components that are in the form of COM-compliant objects including DirectGraphics, DirectInput, DirectPlay, DirectSound, DirectMusic, DirectSetup and DirectX Media Objects. [1] [43] It allows multimedia applications to have access to the advanced features of high-performance hardware such as graphics cards and sound cards. DirectX achieves this by translating generic hardware commands into hardware specific code. The technology was first introduced in 1995 and it is now considered to be one of the main 3D graphics APIs on Windows machines. [49]

The technology comes with some powerful toolkits that make it easy for software developers to build DirectX applications. Examples of such toolkits are D3DX and PIX which are both included in DirectX 9.0 SDK. [48] The D3DX library exploits the latest graphics technology that makes mesh optimisation, tangent-frame generation, and

pre-computed radiance transfer (PRT) possible. PIX helps software developers to optimise their Direct3D applications by allowing them to investigate each and every Direct3D call and visualise it as it renders.

Unlike OpenGL, DirectX cannot fully emulate functionality which is not provided by hardware. Although DirectX includes a fully featured software vertex-processing pipeline, no emulation is provided for pixel level operations.

The technology was mainly developed for computer games. Today, due to its high speed and high quality rendering ability that takes advantages of the latest 3D graphics hardware, it is also used in many software fields such as the engineering sector. Many of the latest multimedia hardware devices only have DirectX-compatible drivers.

Currently, the latest version of DirectX was 9.0c. DirectX 10, which was initially called Windows Graphics Foundation, is expected to be released as a part of the next generation of Windows, Windows Vista. With DirectX, users can experience better three-dimensional graphics with immersive and audio effects.

2.2 Web-based Simulation Libraries

This section intends to investigate some of the web-based simulation libraries that come within the scope of this paper.

2.2.1 SimJava

SimJava is a process-based discrete event simulation library, originally authored by Ross McNab and Fred Howell at the University of Edinburgh in 1996. [27] The library was written in Java and based on HASE++, a C++ simulation library. [33] [36] HASE++ was in turn based on Jade's SIM++. As the library is based on Java, it inherits the virtue of being portable and platform-independent. The primary purpose of developing SimJava was to allow live simulations to be incorporated into web documents describing the behaviour of computer architectures. [28]

The visualisation of a simulation model is performed on a Java applet or a frame with a collection of animated graphical representations of entities, ports, parameters and links. Prior to this project, these components were graphically represented in GIF file format.

Each entity is assigned a light-weight thread which is controlled by the simulation kernel. Entities can communicate with each other by scheduling event objects through ports. Ports are linked by unidirectional communication links.

The simulation kernel is responsible for maintaining thread control, advancing simulation time and keeping track of the event queues. The kernel advances simulation time in an event–driven manner, meaning that the simulation clock progresses on the basis of the event scheduling times.

SimJava comes with a variety of statistical tools that allow performance assessment. In 1997, Costas Simatos added considerable statistical and reporting capabilities to SimJava. This improvement allows modellers to add detailed statistical measurements to simulation entities and perform output analysis to test the quality of collected data. Furthermore, much effort has gone into the automation of all possible tasks allowing modellers to focus solely on the modelling aspect of simulations. Automated tasks range from seeding the random number generators used in simulations to producing detailed and interactive graphs. [62]

At its most basic level, SimJava is composed of three packages, eduni.simjava, eduni.simanim, and eduni.simdiag. eduni.simjava is a package for creating stand-alone and text-only Java simulations, which produce a trace file as the output by default. eduni.simanim provides visualisation of simulations by integrating with the simjava package though the trace file it produces. eduni.simdiag is a collection of JavaBeans-based classes for displaying simulation results. On top of this, there is a variety of packages which serve as extensions to the basic SimJava packages. Among them are Distributed SimJava, SimProd and GridSim.

Distributed SimJava is an extension to SimJava, developed by the MITRE corporation, that adds distribution capability to SimJava' s basic kernel. [53] [54] This extension enables modellers to construct distributed and web-based simulations, thus sharing the simulation's workload. The extension utilises the Remote Method Invocation (RMI) facilities of JDK 1.1. RMI is akin to Remote Procedure Call (RPC) apart from the former being object oriented. With this extension, entities are transformed into remote objects and their methods can be invoked from another Java Virtual Machine, potentially on different hosts. The primary purpose of the extension is to augment SimJava's interoperability rather than to increase speed.

SimProd is another SimJava extension package developed by the Asian Institute of Technology. [6] SimProd is a reusable object library that is specifically designed for developing flexible simulation models of production systems on the web.

The final SimJava extension to be mentioned is GridSim, developed by the University of Melbourne. [50] [51] It is a discrete-event grid simulation kit which is specially designed to perform simulation over a heterogeneous grid environment where resources and users are geographically distributed across multiple administrative domains with their own policies. The toolkit provides a suite of facilities for application construction, information services for resource discovery, and interfaces for mapping application tasks to resources. With these facilities, users can simulate resource brokers or grid schedulers to evaluate the performance of scheduling algorithms or heuristics in a completely repeatable and

controllable manner which is otherwise impossible.

Today, SimJava has been used in a number of research and student projects. Such examples include “Working model of the internals of the DLX processor” by Iain Patterson and “Simulation of distributed commit protocol” by Steven Ramsay. [55] [58] This proves that SimJava is widely acknowledged as a reliable and flexible simulation tool among many research institutes. The language is well-suited for modelling hardware and parallel software.

2.2.2 Automatic Data Distribution Tool (ADDT)

ADDT is a web-based tool developed by the Department of Computer Science in New Mexico Tech. [4] The tool allows the simulation and visualisation of automatic data distribution algorithms in three-dimensional graphics. ADDT maps data structures of composite grid applications onto processors in both homogeneous and heterogeneous parallel systems and attempts to achieve an efficient mapping to optimise load balance and communication. The primary use of ADDT includes composite grid applications such as nuclear reactor simulations and aerodynamic simulations which involve multiple processors.

ADDT has been developed with Apache Tomcat, HTML, XML and Java Servlet Technology. As shown in Fig. 2.2, the system forms a client-server model. [4] An input file is submitted to the server machine. The server performs the distribution algorithm to distribute the data in the input file and generates output files which contain distribution and performance data. Those files are then converted into XML files. Java Servelet acts as a web interface between these steps. Finally, the XML files are processed and VRML files which allow three-dimensional visualisation on the web are generated.

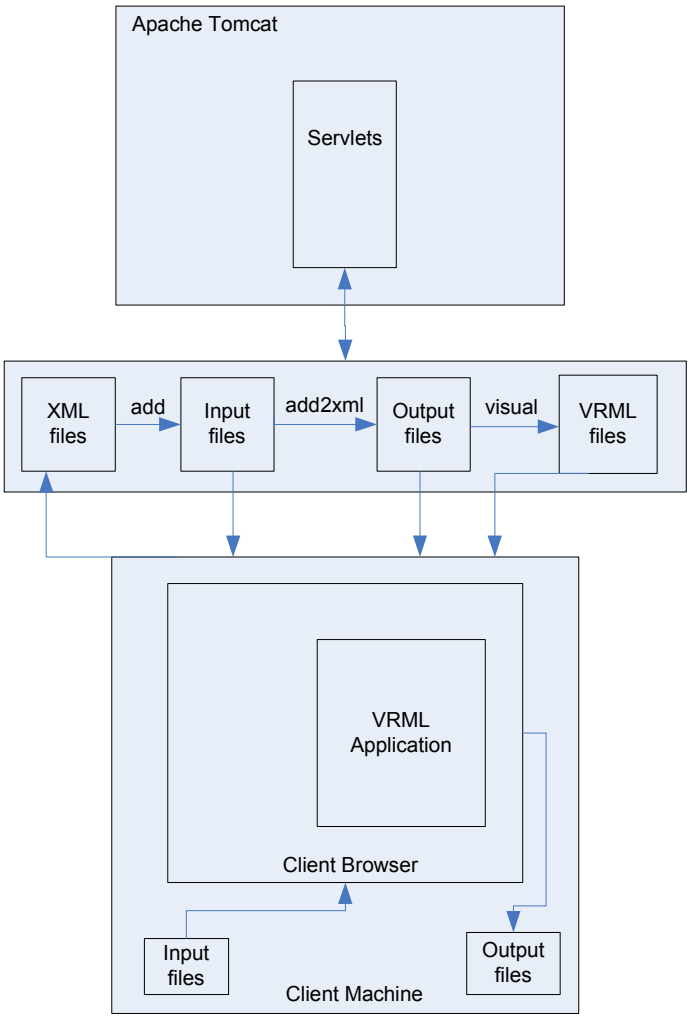

Figure 2.2 ADDT Structure Diagram

Distribution algorithms quite often involve some complex concepts. However, visualisation enhances users' understanding of the concepts by illustrating how the grids are distributed onto processors, the computation cost on each processor, the communication cost between processors and the list of grids on each processor.

ADDT contains many elements that SimJava3 had hoped to achieve. These elements include three-dimensional visualisation over the web, portability of the simulation model, interactivity with the simulation model and a system structure that efficiently translates simulation output into visualisation.

ADDT's future objective is to optimise the generation of visualisation, as currently it accounts for the most of the computational time.

2.2.3 Java And Web-based Algorithm Animation (JAWAA)

JAWAA is a scripting language developed by Duke University for creating web-based educational animations. [2] JAWAA features various built-in data structures which can easily be animated. Users begin with a program written in any programming language. Then the system takes in the program and outputs JAWAA commands into a .anim file. The file is placed in a web-accessible directory along with an HTML page which contains a reference to the .anim file. The HTML document is then loaded by a web browser and an animation is run.

The language comes with JAWAA Editor which allows users to create animations graphically by laying out objects and modifying them across time. With this, users can build animations without much knowledge of the JAWAA language and avoid the tedious guesswork involved in placing objects in the right place with textual commands.

The language is primarily designed for educational purposes. Most simulation models built with JAWAA are simple animations to illustrate elementary computer science concepts. There have not been any complex simulation models that are at the research or commercial level.

Therefore, at the present stage, we can safely say that JAWAA excels in creating very simple animations to enhance students' understanding of computer science concepts, but is inadequate for research or commercial purposes where the preciseness of an animation is vital and involves complex computation.

2.3 Simulation Environment

As the second part of the project attempted to build a simulation environment which is akin to HASE, it was worthwhile examining some of the important features of HASE.

2.3.1 HASE and Hase++

HASE is a Hierarchical computer Architecture design and Simulation Environment which allows for the rapid development and exploration of computer architectures at multiple levels of abstraction, encompassing both hardware and software. [29] HASE itself has been based on object-oriented simulation languages. The first implementation of HASE was developed using DEMOS, the second version in Sim++ and the current version in Hase++. Since then, the system has evolved in various forms to meet the requirements of several projects for which it has been used and it is now a sophisticated and relatively stable environment.

The system provides visual verification of a hierarchically structured simulation model via an animation in a design window. It does this by generating trace files which can then be used to perform animation of the design so as to show data movements, state changes and parameter values of entities.

The initial idea of the system came from a simulator built for MC88000 systems, written in OCCAM and run on Meiko Computing Surface at the Edinburgh Parallel Computing Centre. [36]

Hase++ is a discrete event simulation library with a programming interface similar to that of Sim++, but implemented using C++ and threads. It includes a set of library routines for process-oriented discrete event simulations and run time systems for multi-threading many objects in parallel and keeping track of simulation time. The language provides a set of useful tools that makes the creation of discrete event simulations with entities running in separate threads possible. Each entity is assigned an independent thread; threads are synchronised by a central class that maintains the future and deferred event queues.

HASE requires two types of file to specify an architecture: an Entity Description Language (EDL) file and an Entity Layout File (ELF). EDL is used to describe the parameters and components of architecture, whereas ELF is concerned with the physical properties of components such as the screen position information of displayed entities, ports and parameters. The important benefit of the animator is that it lets users check that the model produces correct results. Fig. 2.3 provides a graphical representation of the HASE software architecture. [32]

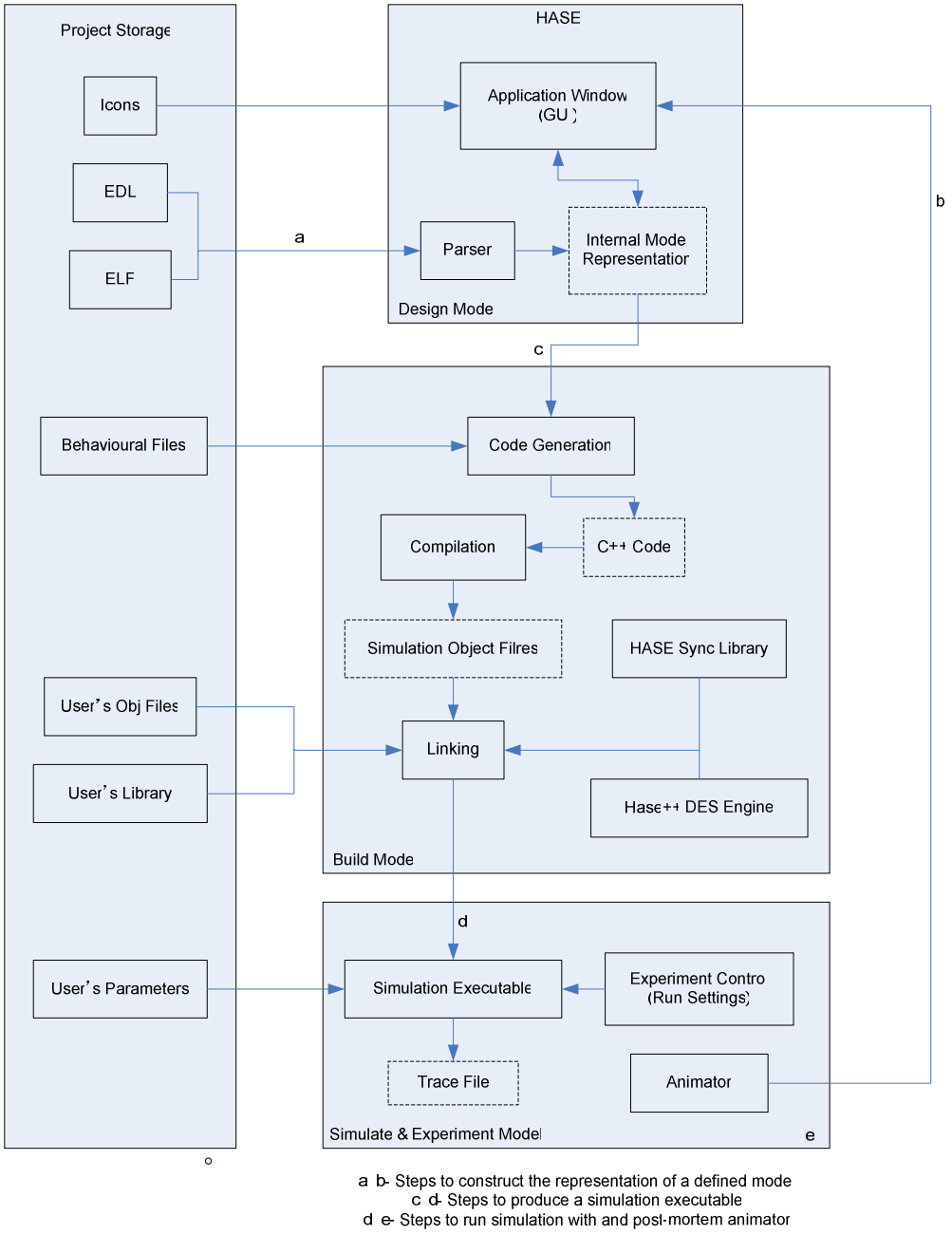

Fig 2.3 HASE System Overview

Later versions of HASE come with a facility called JavaHase that enables the conversion of HASE simulation models into web-accessible Java applets. [37] The resulting applets are essentially SimJava applets as they use the SimJava simulation engine.

HASE has been used in a number of research projects. Examples of them are the modelling

of scalable implementations of the hierarchical PRAM model of parallel computation on a 2D mesh, the evaluation of the performance of multiprocessor interconnection networks and a model of the QCDOC architecture.

There have been radical changes in HASE in the year 2005. Recently, David Dolman and Juan Carlos Diaz y Carballo at the University of Edinburgh have produced a Java version of HASE. Not only have the simulation engine and graphical user interface been translated into Java, but the latest version is accompanied by a suite of sophisticated tools with a more interactive and intuitive graphical user interface. It is expected that the future development of HASE will be more focused on the Java version of HASE than its predecessor in C++.

Chapter 3

SimJava3

The following three chapters describe the design and implementation of this project. They highlight problem areas that had to be addressed while developing the project.

The design phase requires the creative procedure of deriving a solution which satisfies the software requirement. Each chapter starts with the description of essential design concepts and careful decisions that had to be carried out. At the same time, essential implementation issues that arose throughout the development of this project are discussed. The key features and the functionality of the simulation library are outlined. However, as there were in excess of 50,000 lines of code involving this project, only a brief overview could be presented.

3.1 Introduction

The overall system design of the new version of SimJava is very much akin to its predecessor. Many of the key principles were inherited. This not only enables the reuse of reliable system components, but also encourages the new system to be compatible with existing simulation models that had been created with the previous version of SimJava.

SimJava3 is essentially composed of two key components, eduni.simjava and eduni.simanim. The correlation between the two components is illustrated in Fig. 3.1. Note that more detail and functionality will be added to this diagram as the discussion continues.

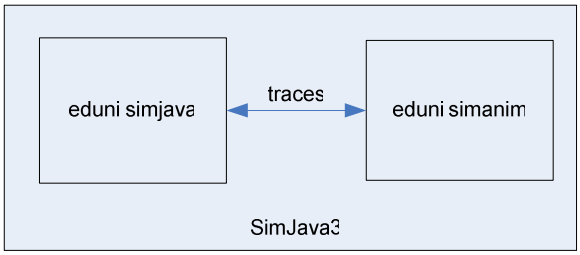

Fig 3.1 Correlation between simjava and simanim

The prime focus of this part of the project is the enhancement of the eduni.simanim package, so as to provide the simulation library with a three-dimensional visualisation capability. Some modifications are also made to the eduni.simjava package.

eduni.simjava is the core simulation package, which is capable of producing text-only simulations. The eduni.simanim package serves as a companion package to the eduni.simjava package and provides a graphical representation and animation of a simulation model. In essence, eduni.simjava contains a simulation kernel and eduni.simanim contains a visualisation engine. The simulation kernel and the visualisation engine collaborate by the means of tracing. A trace is an interface between the simulation kernel and the visualisation engine for the purpose of advancing both a simulation and an animation.

Before going any further, some of the essential components of the eduni.simjava package are highlighted

3.2 The key components of the eduni.simjava package

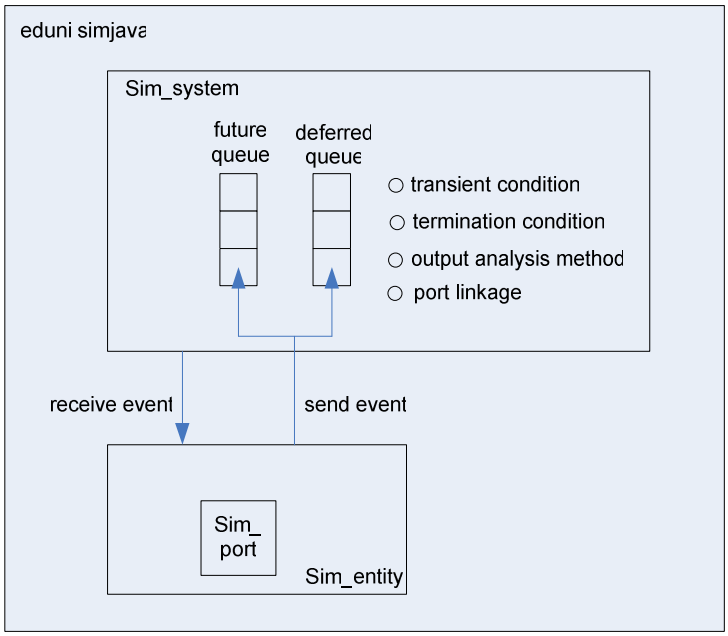

Fig. 3.2 eduni.simjava package

- **Sim_system**

The most important component of the eduni.simjava package is Sim_system which is a discrete event simulation kernel. The simulation kernel is responsible for every aspect of simulations. All entities involved in a simulation need to be registered with the kernel. The kernel manages these entities and synchronises their behaviour with the use of semaphore objects. The simulation time advances on the basis of the event scheduling times.

Furthermore, every port linkage needs to be defined through the simulation kernel.

The kernel keeps track of two types of event queue, namely the future queue and the deferred queue. All events, generated or transmitted by entities, are assigned to either of the queues. “The future queue holds all the events that have been generated by entities but whose delivery time has not yet been reached. The deferred queue holds all the events received by entities that were busy at the time they were delivered.” [61] The kernel manages and respectively processes each of the events in the queues.

Modellers are allowed to specify a transient condition, a termination condition and an output analysis method to be included in a simulation model. They are controlled internally by the simulation kernel.

By default, at each simulation completion, the simulation kernel produces a simulation report. The report contains general information about the simulation run, such as the duration of the simulation run and the output of statistical measurements. In addition, the kernel provides modellers with control over the details of a report.

- **Sim_entity**

An instance of the Sim_entity class represents an entity or a process in a simulation model. Sim_entity is an independent process unit that encapsulates a portion of the dynamic characteristics of a simulation model. Indeed, the whole simulation model is considered to be a collection of interacting processes or entities. Each entity is assigned a unique name. The communications and interactions between entities are accomplished by scheduling events via ports. Each entity runs in a light-weight thread whose behaviour is controlled by the simulation kernel.

Modellers can add detailed statistical measurements to simulation entities. The results of those measurements are collected at the end of each simulation run and an output analysis is performed to assess their quality.

In order to define an entity, modellers must create a subclass of the Sim_entity class. The constructor of a subclass is where all the initialisation of an entity is performed. This is where ports, statistical measurements and random number generators are defined. Subsequently, the modellers needs to override the body() method where the behavioural aspect of an entity is portrayed.

The generic implementation of an entity class is as follows:

```
import eduni.simjava.*; //Import the simjava package

class EntityName extends Sim_entity
{
        private Sim_port out; //declare port variables here.
        Private Sim_stat stat; /declare a statistical measurement variable here.

        EntityName()
        {
                super(); //call the constructor of Sim_entity.
                out=new Sim_port("portname");//create the port.
                add_port(out); //add the port to the entity.

                stat=new Sim_stat(); / /create a stat object.

                //specify statistical measurements here.
                stat.add_measure(Sim_stat.THROUGHPUT);
                set_stat(stat); //add the stat object to the entity.
        }

        body()
        {
                //while the simulation is running
                while(Sim_system.running())
                {
                        //describe the behavioural property of the entity here.
                }
        }
}
```

- **Sim_port**

Sim_port represents a port which is the means by which entities communicate with each other by transferring event objects. Ports are defined and held by an entity class.

In general, ports are organised in pairs. Modellers must explicitly define port linkages through the simulation kernel. Each source port is linked to a single destination port but a destination port can be linked with multiple source ports.

- **Sim_event**

Sim_event represents an event which is passed between entities during a simulation.

3.3 Enhancements to the eduni.simanim package

Having introduced the main components of the eduni.simjava package, we shall now focus on design and implementation issues that arose during the development of the new eduni.simanim package. The three-dimensional visualisation capability is included in the eduni.simanim package.

3.3.1 Graphics technology

Having examined all possible three-dimensional graphics technology in the previous chapter, this section serves as the justification of the choice of a three-dimensional graphics technology to be used for the next generation of SimJava.

The first and obvious decision was not to use DirectX. Although DirectX is, in general, compatible with most of the latest graphics hardware, its fast graphics rendering performance does not have precedence over the advantage of being platform-independent as seen in the other graphics technologies. Making simulation models available for only Windows users would seriously detract from the key purpose of SimJava, namely incorporating a “live diagram” in a document that can be easily distributed and viewed from any platform on the web.

In contrast, OpenGL seemed to be a better candidate than DirectX, as its application can be run on various platforms. However, the technology exposes application developers to many low-level rendering details. This may be useful when we build elegant three-dimensional

graphics models, but as our focus is on scene composition rather than fine rendering, this is considered to be an unnecessary burden. Moreover, as mentioned in the previous section, due to claims of a possible significant decrease in OpenGL's performance in the next version of Windows and the fact that Windows retains the highest share in the current Operating System market, the use of this technology may lead to future difficulties. Another important argument is that OpenGL is typically used for developing standalone applications rather than online applications.

This situation left the author with the choice between either VRML or Java3D. The question of which technology is more suited for online three-dimensional graphics application has always been a highly controversial issue. Both VRML and Java3D support high-level scene composition. VRML has existed for longer than Java3D and has been favoured by many computer science researchers and commercial software programmers. On the other hand, there are very few research projects and literature references that exploit Java3D. This situation has made it considerably difficult for learners to acquire the Java3D skills. Having said this, VRML has not been updated since its last update back in 1997. Therefore, this technology is now considered to be long-since obsolete.

As far as the technical aspect is concerned, the most distinctive boundary between these technologies is that Java3D is a programming API whereas VRML is a file format. Accordingly, in the case of VRML, three-dimensional content is loaded into a program for display purposes, whereas Java3D programs are executed directly by the Java runtime system. Although VRML allows modeller to define some runtime semantics, the technology is not as robust as the combination of Java and Java3D. In this respect, Java3D excels in encapsulating the dynamic characteristics of simulation models.

As a result, Java3D was recognised as the ideal three-dimensional graphics technology for the next generation of SimJava.

3.3.2 The main components of the eduni.anim package

Let us now move on to the description of the main constituents of the eduni.anim component that play an important role in the system.

- **Anim_applet3d**

Anim_applet3d is essentially a Java applet. The applet presents a canvas that displays the three-dimensional graphical representation and animation of a simulation model with intuitive graphical user interface components that gives modellers control over the simulation.

When modellers wish to add animated visualisation into their simulation model, they must subclass Anim_applet3d and override some or all of the following methods. The only method that is required to be overridden is anim_layout().

anim_layout():

This is where the topology of a simulation model is formed. First of all, if required, modifications of the seed sequences of sample generators need to be performed. Followed by this, entities are instantiated and ports are linked via the simulation kernel.

anim_init():

Should modellers wish to add additional graphical user interface components to the Anim_aplet3d applet, those components must be specified within this method. The components can be added above or below, or to the right or left.

anim_start():

The code written within this method is executed at the start of each simulation run.

anim_completed():

The behaviour of a simulation model described within this method is executed upon completion of each simulation run.

anim_output():

This is where modellers specify whether or not to include a report or simulation messages in the applet. Simulation messages are the textual output that would be displayed during the running of a standalone text-only simulation. Moreover, the method controls the details of a report and simulation messages.

sim_setup():

The method specifies a transient condition, a termination condition and an output analysis method, if they are being used in a simulation model. They are defined through and maintained by the simulation kernel.

- **Sim_anim3d**

In essence, Sim_anim3d is a visualisation engine that is responsible for all aspects of visualisation and animation. The visualisation engine contains a canvas which depicts

three-dimensional graphics content. At each simulation start-up, the visualisation engine initialises the canvas, and constructs and compiles graphical contents in a scene-graph manner. Moreover, it constantly checks for interactions by users with the simulation model. The visualisation engine allows users to navigate through the three-dimensional graphical environment. Further details of these navigation mechanisms are covered later in this chapter.

When entities are pointed to with a mouse cursor, their parameter values will be displayed on the screen. If a state parameter is set for an entity, the simulation engine changes the colour of the entity according to its internal state.

The visualisation engine collaborates with the simulation kernel by means of transfers of events and traces, and updates the display accordingly.

- **Anim_entity3d**

This is the visual metaphor of a Sim_entity instance. The class contains only the display information of an entity, such as the entity's shape and coordinate values, and does not contain any behavioural characteristics. There are four entity types, BOX, SPHERE, CONE and CYLINDER. If the BOX type is set, an entity needs to be given with length, width, and height values. In the case of the SPHERE type, a radius value is required. Lastly, for the CONE and CYLINDER types, radius and height values need to be given. These values are measured from the physical centre of the entity.

- **Anim_port3d**

The instance of this class gives a graphical representation of a port. It retains an absolute coordinate value and a relative coordinate value. The absolute coordinate value is the coordinate value that is centred at the origin of the entire scene. In contrast, the relative coordinate value is the distance from the centre of its parent entity. Whenever the parent entity is relocated, the absolute coordinate value is recalculated. If a port is a source port, then Anim_port3d draws a link to its destination port. The width, colour and pattern of links can be changed manually by users. Four types of link pattern are available, PATTERN_SOLID, PATTERN_DASH, PATTERN_DOT and PATTERN_DASH_DOT. By default, links are one pixel thick, have a solid pattern and are white in colour.

Anim_port3d is in charge of animating message transfers. Messages are passed along links and are visible from any perspective. In other words, they are always oriented towards the viewers. By default, messages are 30pt in font size, with "Serif" font, in plain font style and in red font colour. These settings can be changed manually by modellers.

Message transfer is done in discrete steps. First, the distances between a source port and a destination port in the x, y, and z directions are calculated. These distances are then divided by the animation steps. The default number of animation steps is set to ten. At each animation step, the x, y and z of the coordinate values of messages are incremented by the divided values respectively. In this way, messages that are submitted at the same time reach their destinations at the same time, even though the distances they travel are different.

In general, modellers do not accurately model the distance between entities in graphical simulations. Hence, this approach was preferred in order to conceal the disparity between the distance in the applet and the actual distance. Should modellers wish to precisely model the distance of a simulation model, then a delay can be incorporated into a message transfer.

The following pseudo code illustrates this behaviour.

```
int anim_steps=10; //animation steps

float dx=dest_port.x-source_port.x; //distance in the x direction
float dy=dest_port.y-source_port.y; //distance in the y direction
float dz=dest_port.z-source_port.z; //distance in the z direction

for (i=0; i<=anim_steps; i++)
{
        if (i == anim_steps)
        {
                //at the end of the animation steps move the message to a very far place
                  where viewers cannot see it.
                port.move_msg(1000.0);
        }
        else
        {
                double msg_pos=(1.0*i)/anim_steps;
                double mx = (double)( msg_pos*dx);//new x value
                double my =(double)(    msg_pos*dy); //new y value
                double mz=    (double)(msg_pos*dz); //new z value

                //update coordinate values
                msg.update_coordinate_values(mx, my, mz);
        }
}
```

The original plan for the animation of packet transfers was to use the Interpolator class. The Interpolator class is a utility class provided by the Java3D library. An interpolator is a node

that performs smooth animations by changing a transform or attributes over time. The use of interpolator nodes would achieve much smoother animations compared to the method described earlier. However, this idea was rejected, as interpolator nodes account for large CPU utilisation. An animation can seriously deteriorate, if a large number of packet transfers are performed simultaneously, .

- **Anim_param3d**

Anim_param3d is an animation parameter which displays data or the state of its parent entity during an animation. Anim_param3d comes in four different types: NAME, VALUE, NAME_VALUE and STATE parameters. The NAME parameter shows the name of a parameter along with its parent entity. The VALUE parameter displays the value held by a parameter. The value is updated by the parent entity during a simulation. The NAME_VALUE parameter shows both the name of a parameter and its associated value along with the parent entity. The STATE parameter is used to reflect the internal state of its parent entity by changing the colour of the entity. The NAME, VALUE and NAME_VALUE parameters are displayed when a mouse cursor points at entities.

- **Anim_event3d**

Anim_event3d holds an animation event. Note that there is a distinct boundary between Anim_event3d and Sim_event events. Sim_event events are processed by the simulation kernel whereas Anim_event3d events are processed by the visualisation engine.

Whenever an entity sends data or updates its parameters, Sim_entity sends Sim_event objects to the simulation kernel. The simulation kernel receives the events and translates them into traces. The traces are then submitted to the visualisation engine. Subsequently, the visualisation engine translates the traces into Anim_event3d events and writes them into the event queue of Anim_applet3d.

3.4 Overall System Flow

Following the introduction of the important components of the environment, we are now ready to carry out a closer inspection of the overall flow of the system. Fig. 3.3 represents the high level account of the system. Note Fig. 3.3 is the refinement of Fig. 3.2

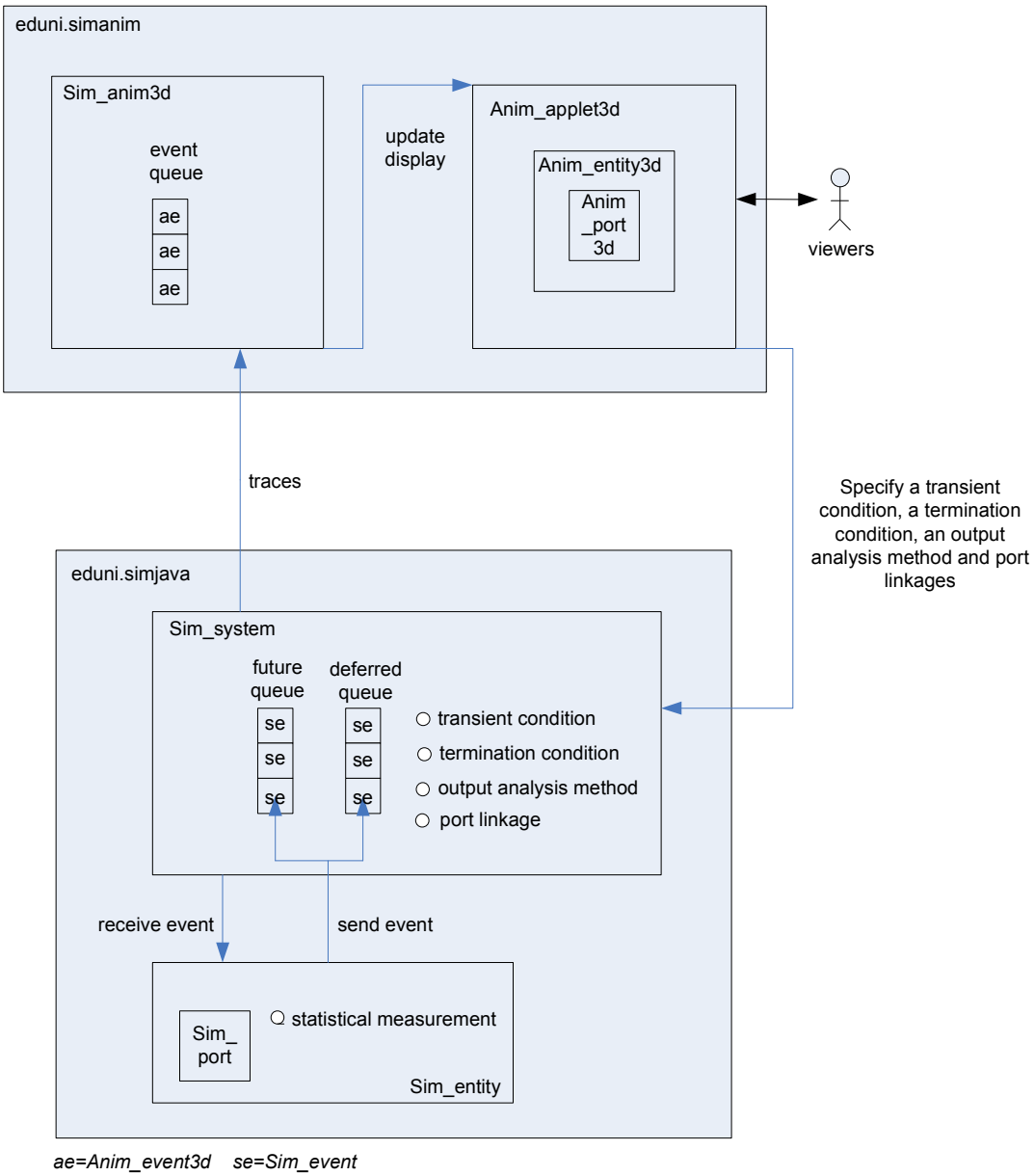

Fig. 3.3 Overall System Flow

3.4.1 Initialisation

When a web browser loads a simulation file, an applet initialises itself and all the key components of a simulation model. Then, the applet sets up its graphical user interface components such as buttons and menus. At the same time, the applet also creates additional graphical user interface components supplied by modellers.

Followed by this, the Sim_anim3d class is instantiated for the creation of a visualisation engine. Subsequently, a simulation kernel is initialised. Once this process is complete, a transient condition, a termination condition and an output analysis method are defined and registered through the simulation engine. The completion of this process establishes the foundation of the system.

Next, the visualisation engine commences the construction of the topology of a simulation model in a scene-graph manner. A scene graph is a tree-like data structure which is composed of nodes holding three-dimensional scene information. Fig 3.4 presents how a scene graph is organised in SimJava3. Instances of Sim_entity and Sim_port are then produced. Accordingly, instances of Anim_entity3d and Anim_port3d are generated. If specified, statistical measurements are included into the instances of Sim_entity.

Each instance of Sim_entity is assigned its corresponding Anim_entity3d object. Similarly, each instance of Sim_port is assigned its corresponding Anim_port3d object. The important point to note is that Anim_entity3d and Anim_port3d contain data that are for display purposes whereas Sim_entity3d and Sim_port3d hold behavioural properties. Lastly, port linkages are defined through the simulation kernel.

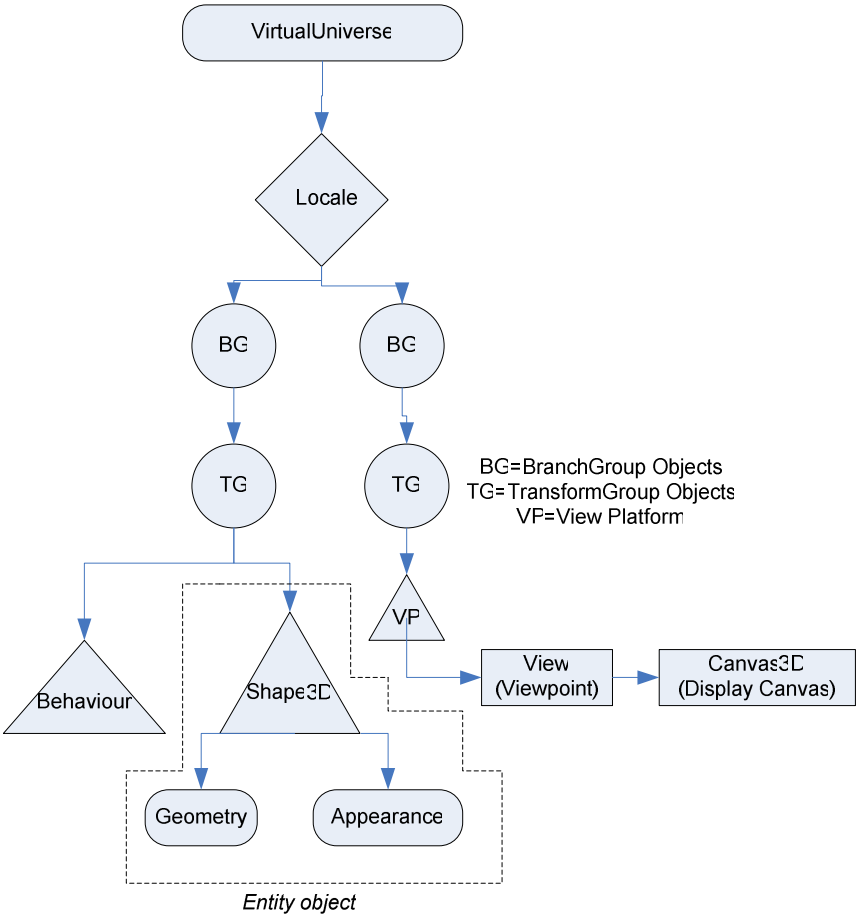

Fig 3.4 Scene graph structure

3.4.2 Simulation Run

Let us now examine the underlying activities of the system during a simulation run. In other words, this section reveals what happens under the surface when viewers click the run button of an applet.

1. If the previous simulation run is terminated by users clicking the stop button, then the applet initialises the simulation kernel and the visualisation engine. In addition, the code within the anim_start() method is executed.

2. A transient condition, a termination condition and an output analysis method are defined and initialised through the simulation kernel.

3 The simulation kernel starts all the threads of entities. By the means of semaphore objects, the system holds the entities until they are up and ready. Once all the entities are ready, they

begin to execute their body() method.

4. The visualisation kernel checks whether the transient or termination conditions have been satisfied, or whether users have pressed the stop button. If so, proceed to step 10. Otherwise continue to the next step.

5. The simulation kernel takes the first Sim_event object from the future queue. The simulation kernel processes the event and advances the simulation time to the timestamp of the event. This is the reason why SimJava3 is called an event-driven simulation library. If the event is of type SEND then it is delivered to its destination entity. If the destination entity is busy, the event is put into the deferred queue. The events in the deferred queue may be processed later by the destination entity. The statistical measurements assigned to the destination entity are updated.

6. The simulation kernel checks the next event in the future queue. If that event has the same timestamp as the previously processed event, then go back to step 5 and process it accordingly.

7. If the simulation has been paused, halt the simulation engine and the visualisation kernel until the simulation is restarted.

8. The visualisation engine processes Anim_event3d events from its event queue. If an event is of type PARAM, the parameter value of its source entity gets updated. In the case of the PARAM_STATE type, the colour of its source entity will be updated accordingly. If the event is of type SEND, the animation of a package transfer is performed.

9. If the timestamp of the next Anim_event3d event has the same or smaller value than the previously processed Anim_event3d event, then return to step 8.

10. The simulation kernel finalises the simulation run and terminates all the running entities. In addition, the simulation kernel updates the graphical user interface components accordingly. Lastly, the code specified within the anim_complete() method is executed.

3.4.3 Navigation

The system allows users to interact with or navigate through a simulation model. There are three navigation modes. The navigation mode can be switched by pressing different mouse buttons.

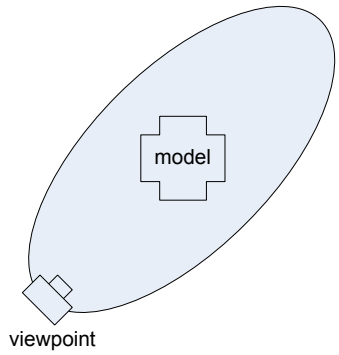

Fig. 3.5 Moving the viewpoint to get the rotating effect

Dragging the left mouse button rotates the geometry of a simulation model around its centre. However, rather than actually rotating a simulation model, the system achieves this by moving a viewpoint. A simulation model may contain many entities and from the system's perspective, rotating such a model is overwhelming. Instead, moving a viewpoint involves much less overload and leads a simulation model to appear to spin at a fixed speed no matter how complex the model becomes.

Dragging the middle mouse button (or ALT+left button) allows users to zoom in and out. This has the effect of viewers appearing to fly through the scene. The right mouse button translates a viewpoint in the direction of mouse movement. This makes viewers appear to walk through the scene.

3.5 Graphical User Interface Design

The graphical user interface is the only means by which users can interact with a simulation model. The screenshot of an example simulation model is shown in Fig. 3.6.

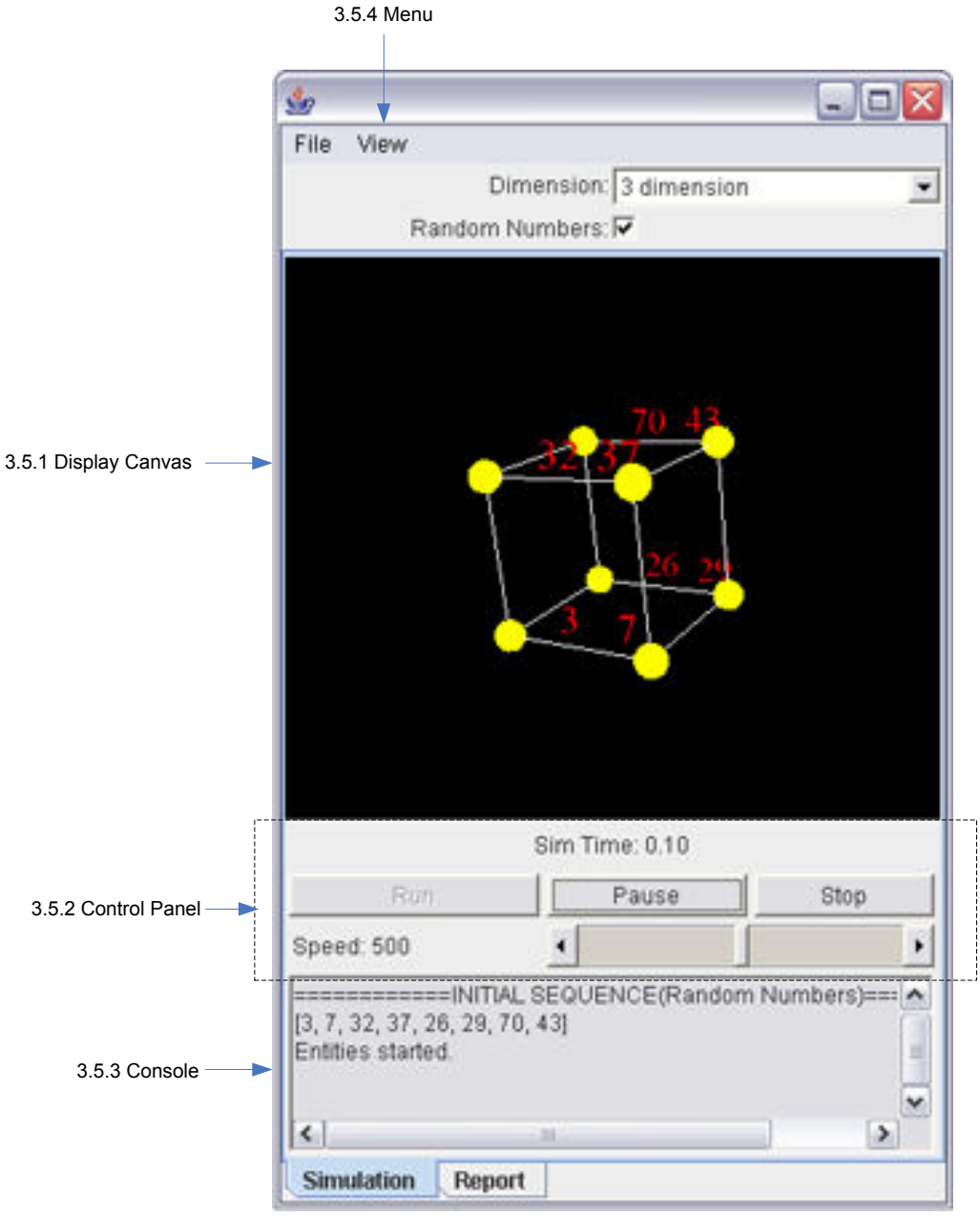

Fig3.6 SimJava3 Screenshot

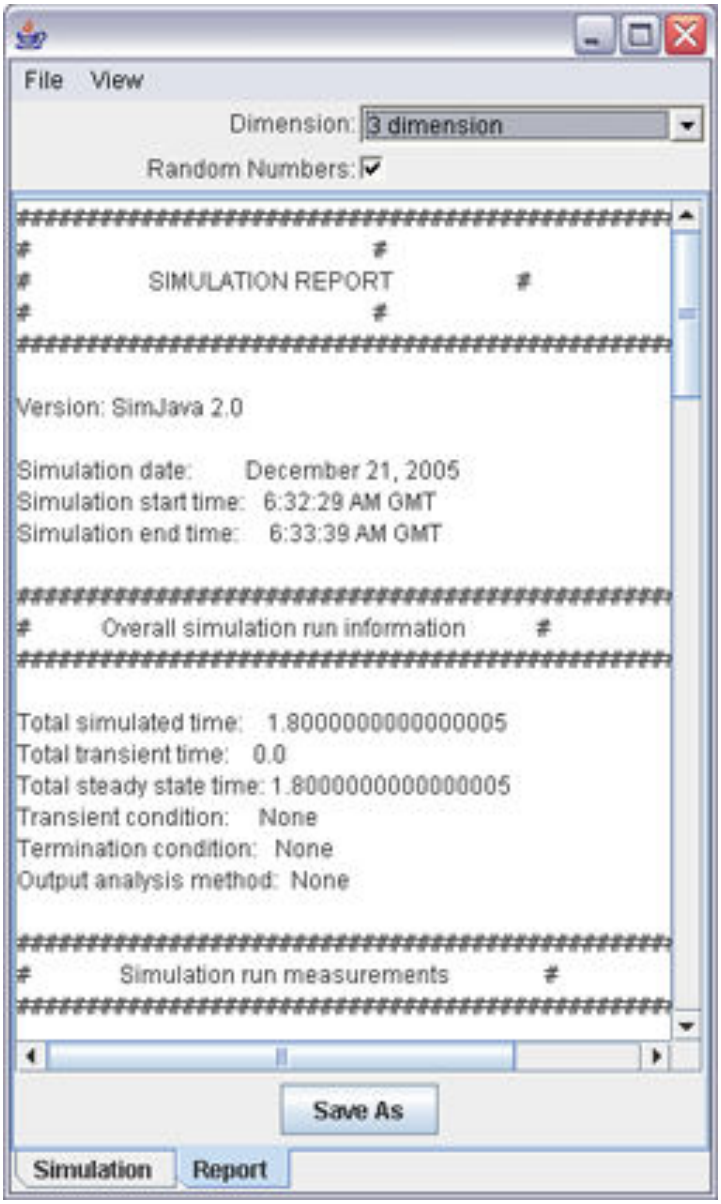

Fig. 3.7 Report Panel

- **Display Canvas**

The display canvas shows the simulation model in three-dimensional graphics. Users can interact with the simulation model by using the mouse.

- **Control Panel**

The control panel gives users control over a simulation run. The buttons let users start, pause, restart and stop a simulation. The simulation speed can be changed by moving the slider.

- **Console**

The console displays messages which modellers would see on the standard output if the simulation were run as a standalone text-only form.

- **Menu**

The menu allows users to quit the program, reset the viewpoint and display all the parameters.

- **Report Panel**

If users have chosen to generate a report, the report is displayed on the report panel (Fig. 3.7) at the end of each simulation run. The report can be saved as a text file for later analysis.

Chapter 4
Simeon

4.1 Introduction

Simeon is a simulation environment which allows the rapid development of simulation models in SimJava3.

Although simJava3 provides the building blocks for creating multi-dimensional simulation models, much of the development time still has to be devoted to geometry work. This involves daunting and frustrating tasks with numerous coding and checking processes which frequently entail trivial errors. The simulation environment fills this void with an intuitive graphical user interface that will allow users to concentrate solely on the behavioural aspect of the models and build their own simulation applications very quickly. Moreover, its effective and efficient I/O management allows modellers to export and save the simulation models in a most convenient way.

This chapter discusses the design and implementation issues concerning the simulation environment.

4.2 Overall System Design

As the system was going to be developed in Java, it was natural to think that the entire system would be organised in an object-oriented manner. The system was decomposed into modules, where a module is an independent subsystem. For the system to be reliable and efficient, it was necessary to impose some kind of meaningful architectural design.

Among many standard architectural designs, a centralised control model was applied to the system. Fig. 4.1 illustrates the overall system structure. The rectangles represent specific classes and the ellipses represent an aggregation of the classes in the simulation environment.

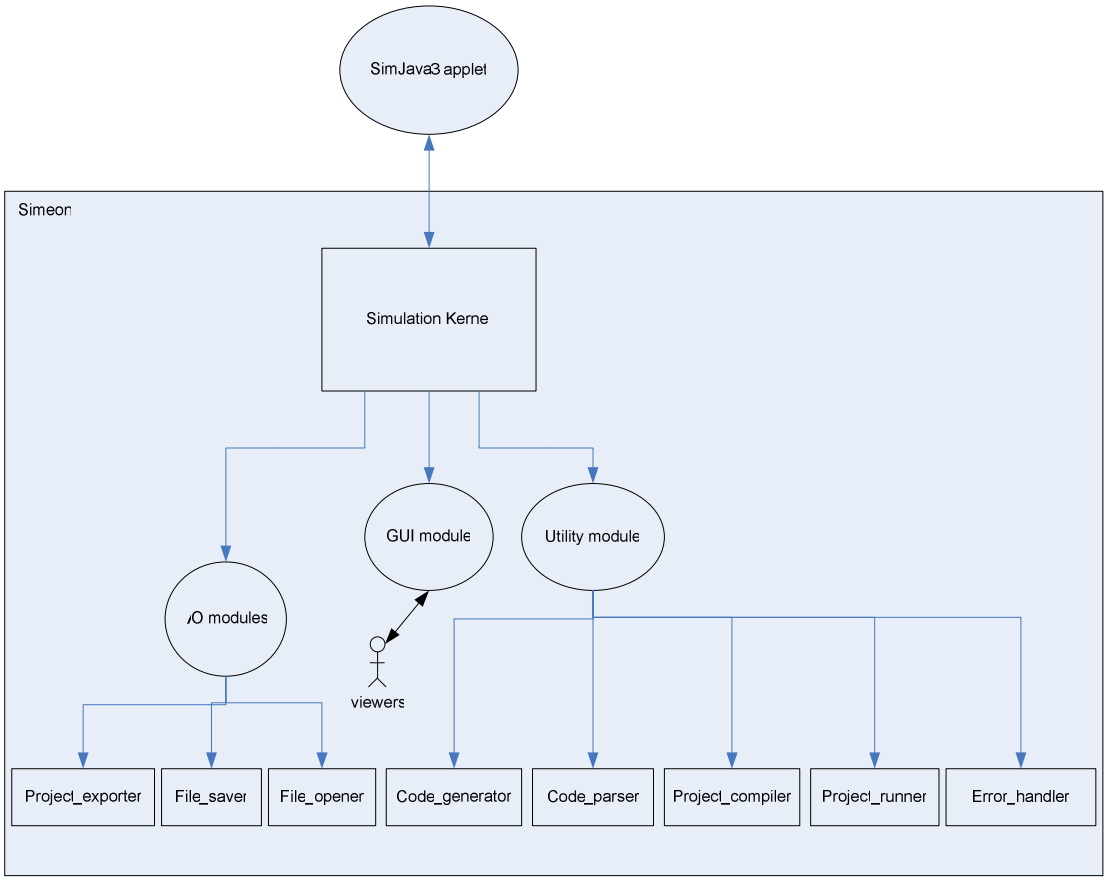

Fig 4.1 Overall system design

Centralised control models fall into two categories, the call-return model and the manager model. In the call-return model, subsystems are involved using traditional subroutine or function calls. The control of the system is passed from a high-level routine down to a low-level routine. Once the low-level routine finishes its process, it can either call other routines or return the control to its parent. This is only applicable for sequential systems. In contrast, in the case of the manager model, subsystems are run as separate processes. The kernel or the manager module controls the starting, stopping and coordination of other subsystems. The manager model is applicable to both sequential and concurrent systems. Simeon mostly follows the call-return model except in the relationship between the kernel and the simulation module, as each simulation model is assigned a separate thread. Hence, the system has the properties of both the call return model and the manager model.

Here, the simulation environment kernel is appointed as the system controller and it has a responsibility for managing the execution of other components of the system. The other subsystems serve as subordinates to the kernel.

This approach allows the subsystems to communicate only through defined interfaces.

Hence、the overall system maintains highly cohesive modules that are loosely coupled. The strength of this approach is the simplicity of analysing control flows.

Having discussed the overall system structure, we now move on to the description of the main components of the system.

4.3 Data modules

The system consists of data modules that are arranged in a hierarchical order. Fig. 4.2 illustrates this.

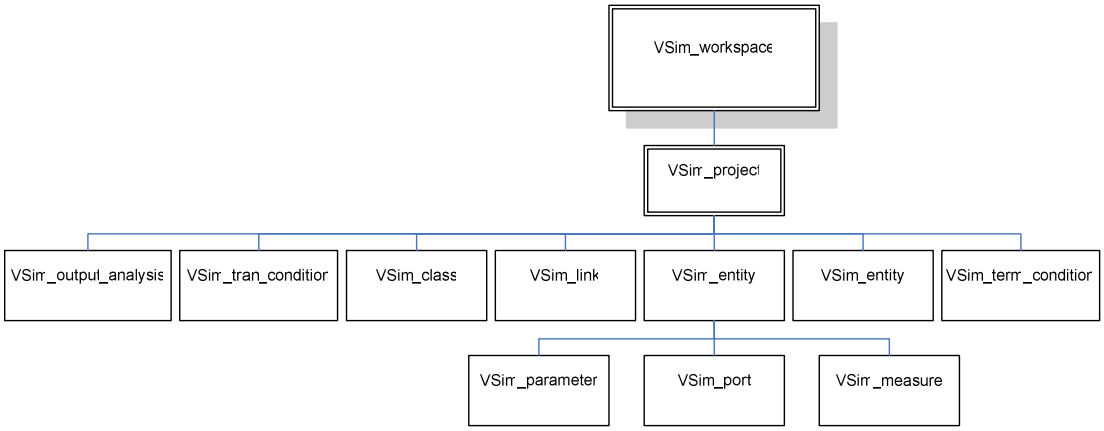

Fig. 4.2 Data modules hierarchy

- **VSim_object**

VSim_object resembles the Object class in the standard Java library. It is the root of the class hierarchy. All other data modules extend this class and may implement some of its methods. The purpose of this class is to generalise the other data modules.

- **VSim_workspace**

VSim_workspace is a representation of a file of the simulation environment. The class manages multiple projects. The following sections give the definitions of the project and the file format. Moreover, the class holds general information about a file such as the date created, the date modified, the file name and the URL address.

- **VSim_project**

A VSim_project object is a unit of a simulation model. Its role is analogous to that of the Anim_applet3d class. The class holds all the data that is relevant to a simulation model such as entities, source code of the simulation model, port linkages, the definitions of a transient condition, a termination condition and an output analysis method, and simulation message and report settings. When the source code is compiled successfully, a VSim_project object holds an instance of Anim_applet3d class that corresponds to a simulation model.

- **VSim_entity**

A VSim_entity object is an abstract representation of an instance of Sim_entity class. It contains all the properties that Sim_entity holds except for behavioural properties.

- **VSim_class**

VSim_class contains source code of an entity which is involved in a simulation model.

- **VSim_port**

VSim_port is an abstract representation of the Anim_port3d class. VSim_port contains all the information held by Anim_port3d except its graphical implementation.

- **VSim_link**

A VSim_link object holds a port linkage.

- **VSim_parameter**

VSim_parameter is akin to Anim_param3d. Like VSim_port, VSim_parameter contains all the attributes intrinsic to Anim_param3d except its graphical implementation.

- **VSim_tran_condition**

VSim_tran_condition contains the definition of the transient condition specified in a simulation model.

- **VSim_term_condition**

VSim_term_condition contains the definition of the termination condition specified in a simulation model.

- **VSim_output_analysis**

VSim_output_analysis contains the definition of the output analysis method specified in a simulation model

- **VSim_measure**

VSim_measure holds the statistical measurement setting of an entity.

4.4 System Behaviour

This section highlights the key features of the system along with the description of the system components that are involved in the process.

4.4.1 File Format

This section explains how simulation models that were developed with this system are preserved.

Fig. 4.3 illustrates the file structure.

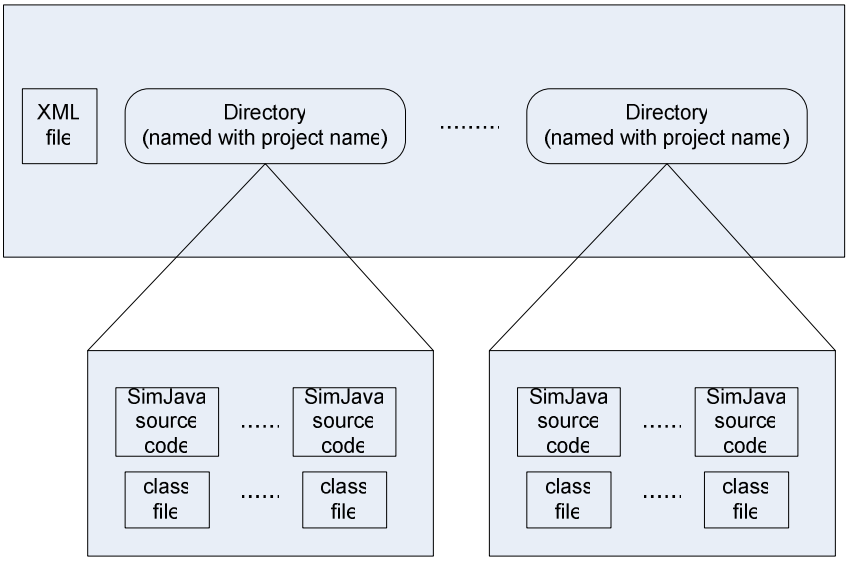

Fig. 4.3 File structure

In short, the file format is a zip file. A zip file has a decisive advantage in that it binds an XML file, SimJava codes and classes into a single file and then compresses them. This is ideal as we can keep the size of a file to a minimum and achieve a high degree of portability. The XML file is the translation of an instance of the VSim_workspace class. The SimJava codes and classes are saved under directories that are named with their parent project name. This file format is defined as a Sim file with a .sim extension throughout this project.

4.4.2 Export project

The system provides users with an option to export a project. In exporting a project, the system automatically generates an HTML document that loads the corresponding SimJava applet. The SimJava code, class files and an HTML document are preserved in a zip file.

4.4.3 Compile Project

When users choose to compile a project, the system generates as many threads as the number of entity classes in a project. Each thread is assigned an entity code and they are compiled in parallel. Once all the entity codes are compiled, the applet code is compiled. At the end of each compilation process, the system displays the result of the compilation process in the console. If the compilation was unsuccessful, the system notifies users about the problem by displaying error messages in the console.

4.4.4 Exception Handling

It is impractical to assume that the system, or any program for that matter, is completely error-free. Particularly as this is the first release of the software, it is plausible that users may encounter some errors in the system. Rather than allowing the system to malfunction, users should be notified about the error in an informative way.

In order to address this issue, if users ever encounter a flaw in the system, an error dialog shows up with the description of the error and asks the users whether they want to report this error to the development team by means of an automatically-sent e-mail.

4.4.5 User Help

The help document is accessible from the program environment. When users click the help menu button, the system launches the default web browser and displays the latest help document. If the users' machines are not connected to the Internet, the system shows the help document which comes with the program.

4.5 Graphical User Interface

Since user interface plays a significant role in the environment's usability, a significant amount of work has done on this aspect of the project. We shall examine the features of the graphical user interface of the environment in an order that users may encounter them in the course of developing a simulation model. Where necessary, the path and the shortcut key to a graphical user interface component are presented.

4.5.1 Overall Graphical User Interface Design

As shown in Fig. 4.4, the overall graphical user interface can be divided into four main parts. The menu and toolbar provide users with access to all features and operations of the system.

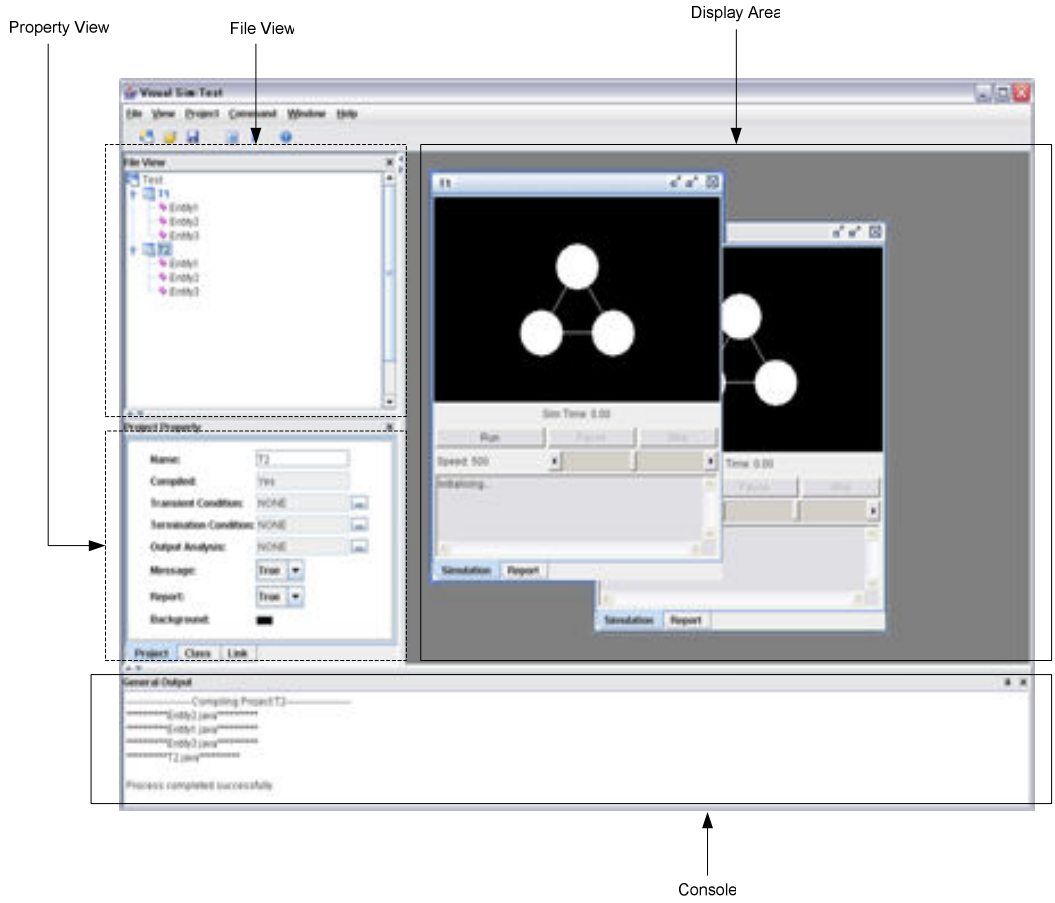

Fig. 4.4 Overall Graphical User Interface Design

The display area is where simulation applets are shown. Each applet is contained in an internal window. Multiple applets can be displayed in this area. The virtue of this interface is that it can reduce window clutter. Providing some window managing features such as cascading and tiling windows, this interface gives good control to users. The operations on windows are provided under the Windows menu.

The console informs users of the state of the simulation environment. For example, when users compile a project, the environment notifies users of which SimJava file is currently being compiled and the result of the compilation process. If the compilation process was unsuccessful, error descriptions are presented in the console. When a long output is produced, users can detach the console for easier reading.

File View allows users to manage and navigate through a Sim file. Components, such as projects and entities, can be added or deleted.

Property View displays the properties of data modules. When users modify the properties,

the environment automatically updates the source code.

File View and Property View are placed in the main window in order to provide users with quick access to the most commonly used operations. More details of File View and Property View are presented later in this chapter.

Console, File View and Property View can be set to 'visible' or 'invisible' by clicking the menu buttons, under the View menu.

4.5.2 Workspace Creation

Path: Menu->File->New->Workspace
Shortcut key: Ctrl+N

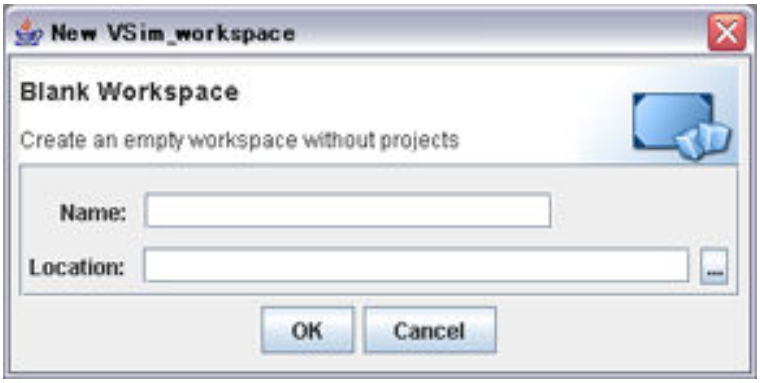

Fig. 4.5 New Workspace Window

The construction of a simulation model with this environment starts with the creation of a workspace, which in turn creates a Sim file. Users are required to provide a workspace name and the directory in which the file will be saved. If an invalid workspace name or URL is specified, a dialogue window informs the users about the problem and asks them to re-enter the values.

4.5.3 Project Creation

Path: Menu->File->New->Project
Shortcut key: Ctrl+P

The construction of a project is carried out in a wizard fashion as seen on many commercial applications. In other words, users are confronted with a series of program screens which enable the creation of a project in a step-by-step manner.

Screen 1:

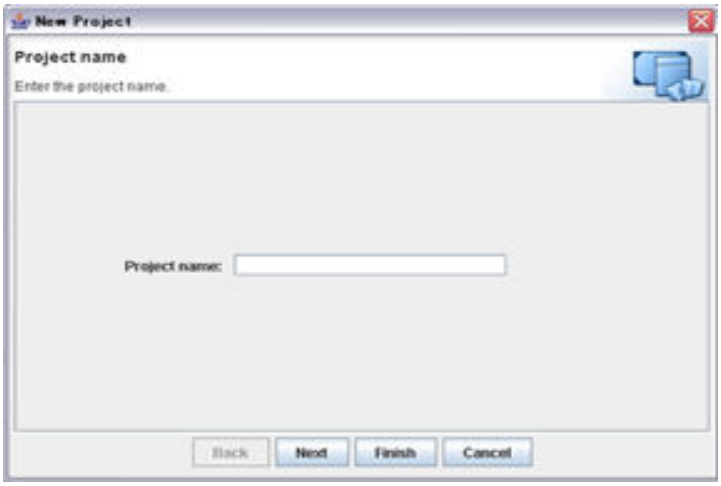

Fig. 4.6 New Project Window (Screen 1)

Users are asked to enter a project name. The project name must start with a capital letter and should not contain any space.

Screen 2:

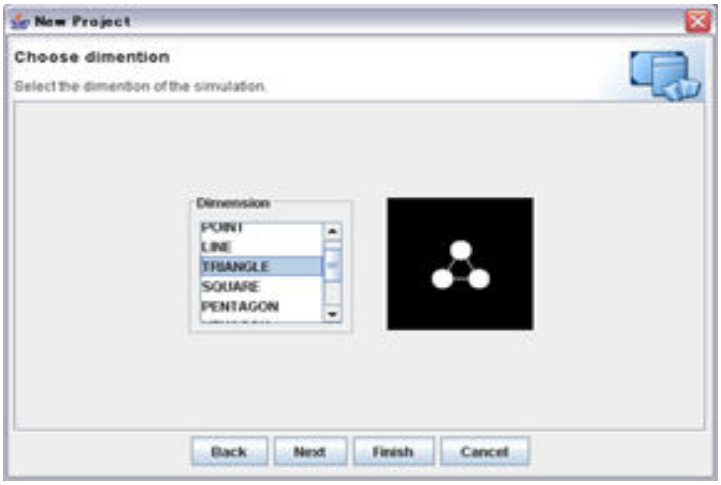

Fig. 4.7 New Project Window (Screen 2)

Here, the topology of a simulation model has to be chosen. For instance, this could be a three-dimensional hypercube or a two-dimensional mesh.

Screen 3:

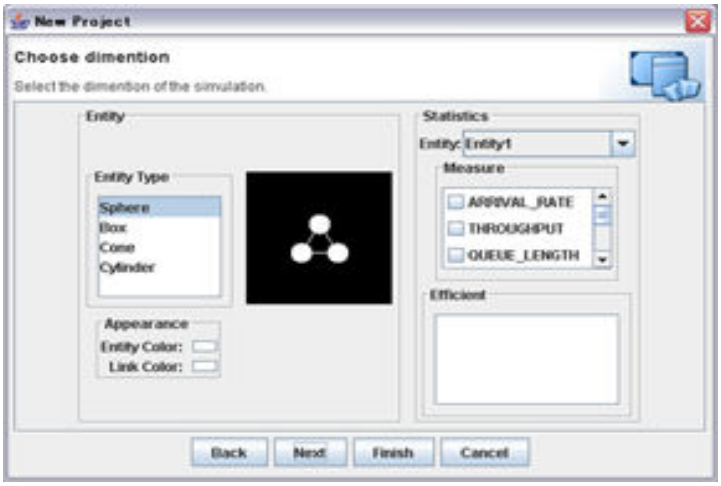

Fig. 4.8 New Project Window (Screen 3)

The shape and colour of entities and the colour of links need to be chosen. In addition, statistical measurements can be specified for entities.

Screen 4:

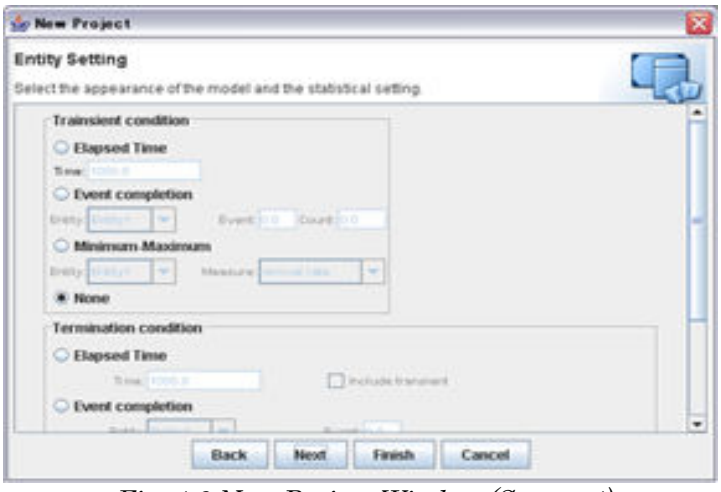

Fig. 4.9 New Project Window (Screen 4)

A transient condition and a termination condition can be defined here.

Screen 5:

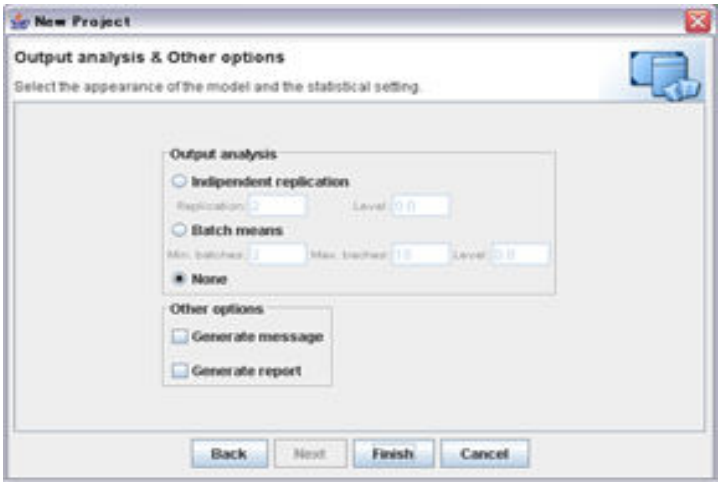

Fig. 4.10 New Project Window (Screen 5)

Lastly, an output analysis method, simulation messages and report settings may be defined.

4.5.4 Navigating though a Sim file and changing the properties of data modules

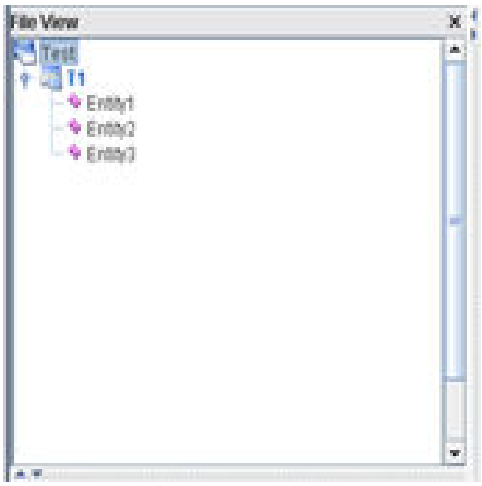

Fig. 4.11 File View

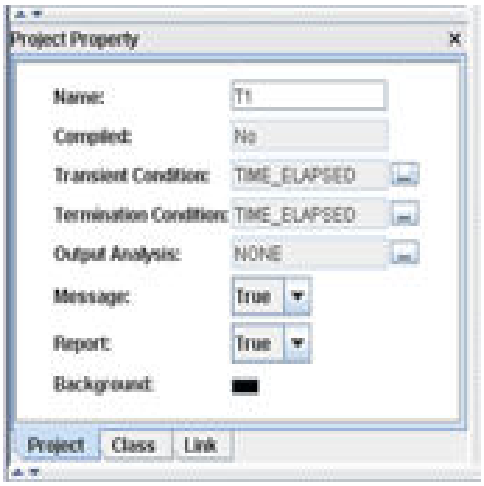

Fig. 4.12 Property View

As mentioned earlier, File View allows users to navigate through a Sim file. File View presents the content of a Sim file in a tree manner. When users select a project node, the system sets the corresponding project as active and highlights the project node. Only one

project can be activated at a time. Any operations, such as the compilation and modification of properties, are applied to the active project.

For each click on tree nodes, Property View displays the property of the corresponding data module. Property View allows users to change the properties through its interface.

4.5.5 Modifying the source code of a simulation model

Path: Menu->Project->Source Code
Shortcut key: Ctrl+K

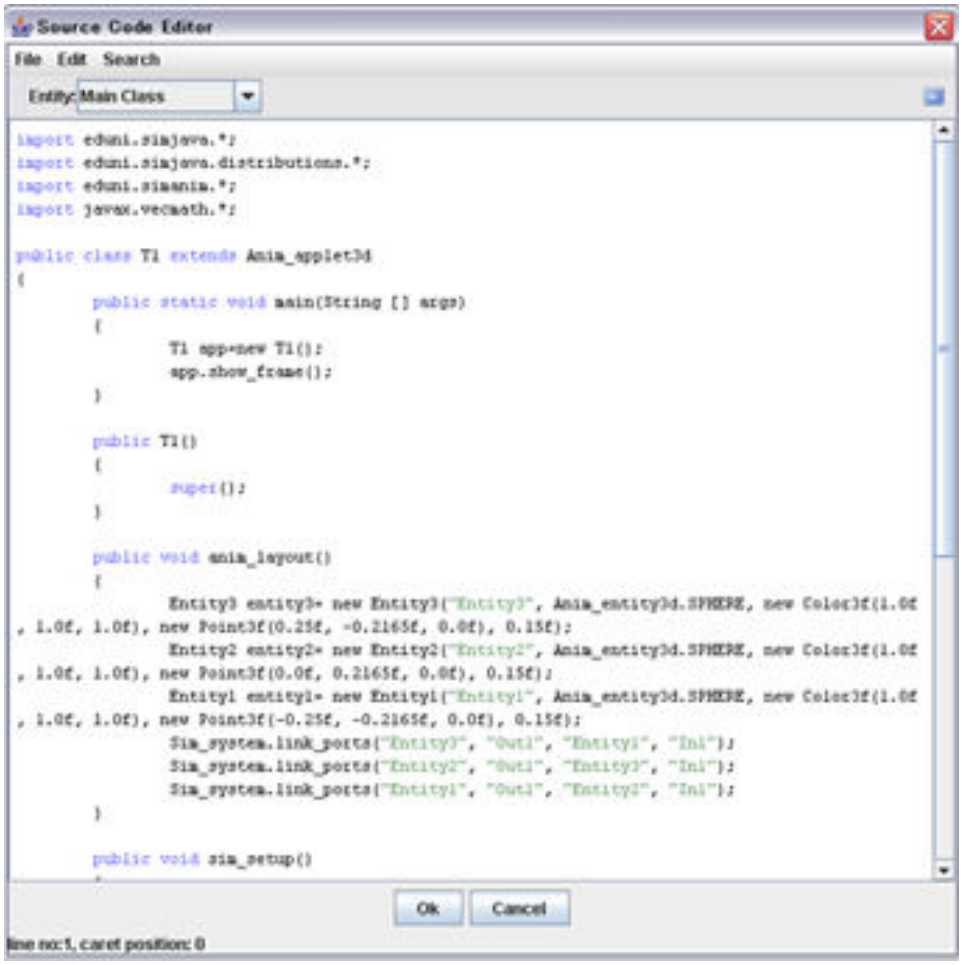

Fig. 4.13 Code Editor

The source code of a simulation model can be manually modified with a code editor which is provided by the system. The code editor contains most of the common operations as seen on many source code editors. When changes are made, they are reflected on the

corresponding data modules.

4.5.6 Adding or deleting entity instances

Path: Menu->Project->Entity Instance

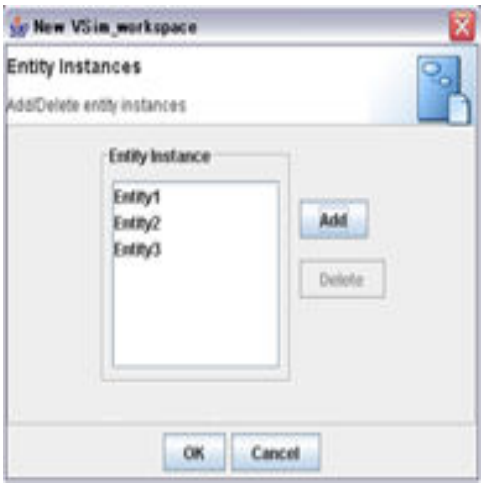

Fig. 4.14 Entity Instance Window

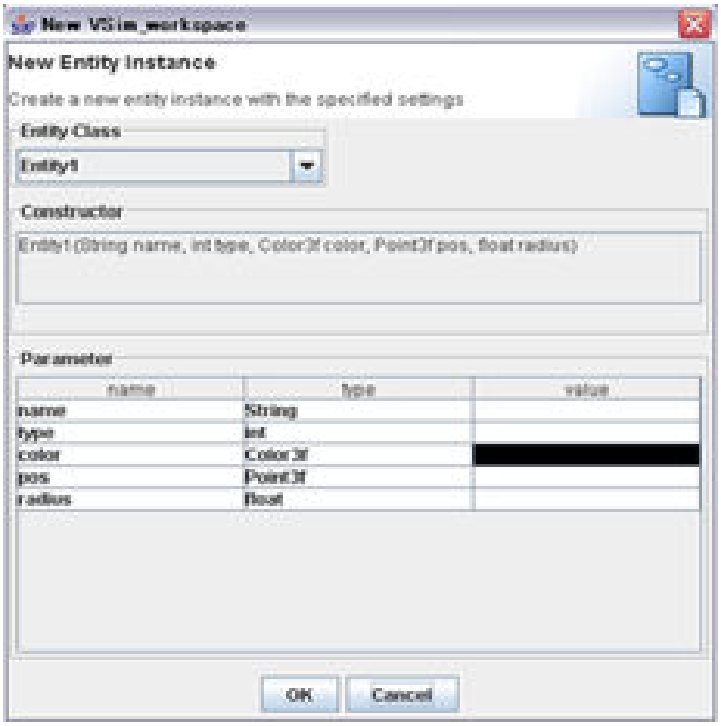

Fig. 4.15 New Entity Instance Window

Entity instances can be added or deleted through the graphical user interface without having to modify the source code manually. When adding an entity instance, users are asked to select an entity class from which the entity is instantiated. Then, the program parses the entity code and determines the parameter values which need to be supplied by users. Subsequently, users are required to input the parameter values. If a parameter value is of type ‘colour’, users can specify a colour value with a colour chooser dialog.

4.5.7 Adding or deleting entity classes

Path: Menu->Project->Entity Class

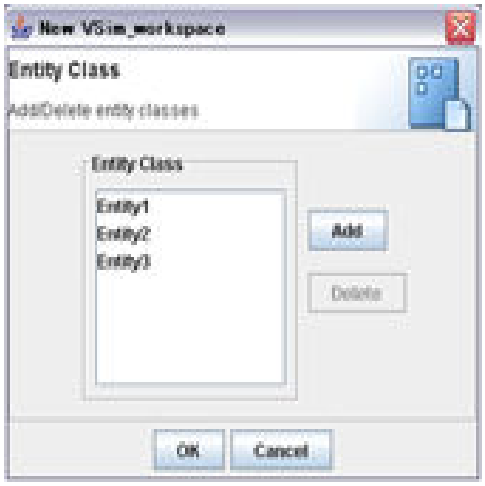

Fig. 4.16 Entity Class Window

Fig. 4.17 New Entity Class Window

Entity classes can be added or deleted graphically. Note, however, that all the entity instances that are derived from the deleted entity class will also be removed.

4.5.8 Adding or deleting port linkages

Path: Menu->Project->Entity Instance

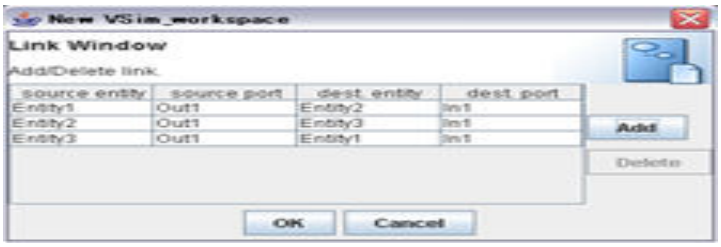

Fig. 4.18 Port Linkage Window

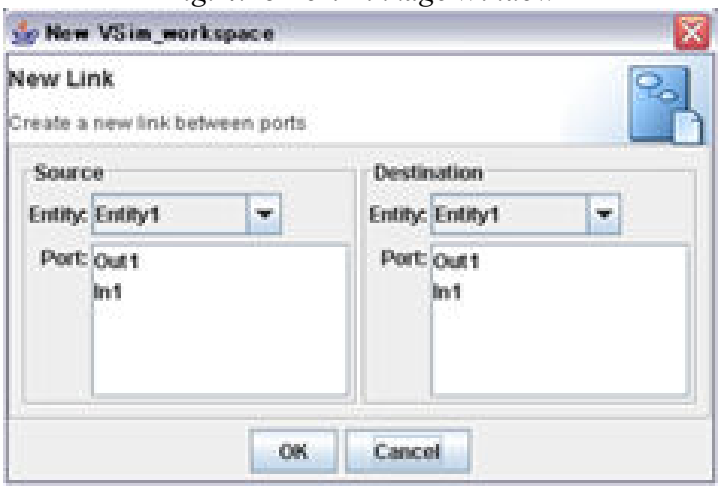

Fig. 4.19 New Port Linkage Window

The current port linkages are presented in a table. Each row in the table represents a port linkage. Users can delete a port linkage by selecting one of the rows.

A port linkage can be added by using a graphical window shown in Fig. 4.19. When a source or destination entity is selected, all the ports contained by those entities are displayed on lists. Users then select source and destination ports from the lists and confirm the port linkage by pressing the ok button.

4.5.9 Defining a transient condition and a termination condition

Path: Menu->Project->Transient/Termination Condition

Fig. 4.20 Transient/Termination Condition Window

A transient condition and a termination condition can be specified through a graphical window.

4.5.10 Defining an output analysis method

Path: Menu->Project->Output Analysis

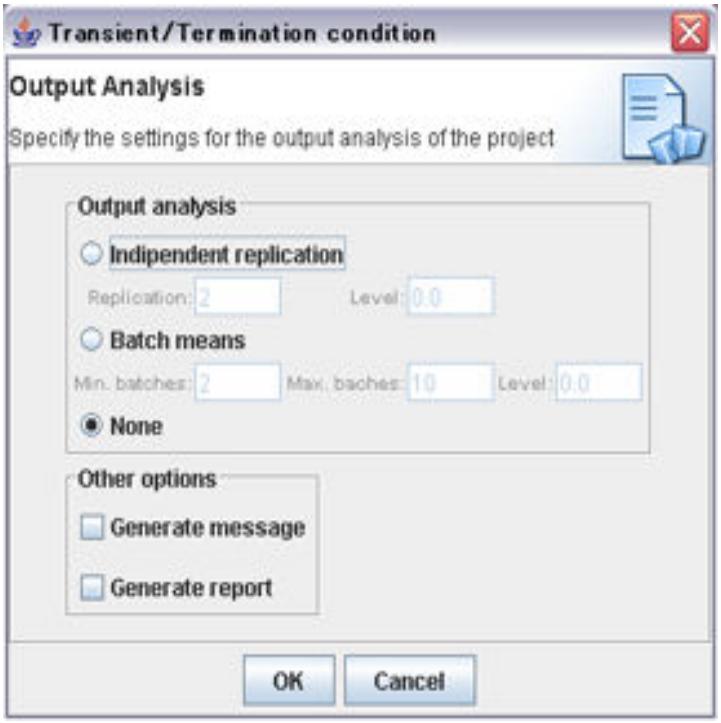

Fig. 4.21 Output Analysis Window

An output analysis method can be defined graphically.

4.5.11Compiling a Project

Path: Menu->Command->Compile
Shortcut key: Ctrl+1

Fig. 4.22 Console

This operation compiles the currently active project. As mentioned before, first, entity codes are compiled in parallel. Following this, an applet code is compiled. The result of the compilation process is displayed in the console.

4.5.12Run a Project

Path: Menu->Command->Run
Shortcut key: Ctrl+2

This operation runs the currently active project. If the project is not yet compiled, a dialog window shows up and users will be informed.

4.5.13 Help Document

Path: Menu->Help->Help Contents
Shortcut key: Ctrl+H

As mentioned in section 4.4.5, at any point, users may wish to refer to the help document. When users access the help menu, the system launches the default web browser and displays the latest help document which is available on the Internet. If the users' machines are not connected to the Internet, the system presents the help document which comes with the program by default.

Chapter 5
Example Simulation Models

5.1 Introduction

This chapter examines example simulation models that were created using SimJava3 and Simeon. The motivation for creating such models is to ensure and demonstrate the reliability of the simulation library and the simulation environment. In addition, they were intended to provide the modellers with an opportunity to become familiar with the syntax of the simulation library and the operations of the simulation environment.

Five example models have been produced of which four of them are hypercube models. We shall start with a description of a SIMD hypercube model. This model serves as a basic hypercube model and the rest of hypercube models have been produced by implanting different parallel algorithms to it. All the example models operated successfully.

This chapter continues with the description of each of the example models along with their screenshots.

5.2 SIMD hypercube model

One of the significant motivations for developing SimJava3 was to model hypercube algorithms with three-dimensional graphics.

In order to address this issue, this section begins with the creation of a SIMD (Single Instruction, Multiple Data) hypercube model. SIMD refers to a set of parallel computer machines that each executes a single instruction at a time. [8] Each instruction may be executed multiple times in parallel on different data, therefore producing multiple results. Each entity in the model represents a processor in a SIMD machine.

In order for a model to be classified as a SIMD machine, some kind of synchronisation mechanism needs to be imposed on it. Thus, a barrier synchronisation policy was applied to the model. This leads to the introduction of another entity, namely a clock controller. The clock controller is in charge of the timing of the execution of instructions. The clock controller submits a single barrier to each of the entities. When entities reach their barriers they are required to pause and only allowed to proceed when all of them have reached the barriers. This causes the model to execute instructions in lock-steps.

There are three types of instructions, SEND, RECEIVE and PROCESS. When the SEND instruction is executed, every entity sends its data to a destination entity. RECEIVE makes

all the entities wait for the transmitted data. The PROCESS instruction, orders all the entities to process the received data.

Each entity reflects changes in its internal state by changing its colour. When an entity is processing the SEND instruction, it changes its colour to blue. If an entity is processing the RECEIVE instruction, it changes its colour to yellow. Lastly, when an entity is executing the PROCESS instruction, it changes its colour to red.

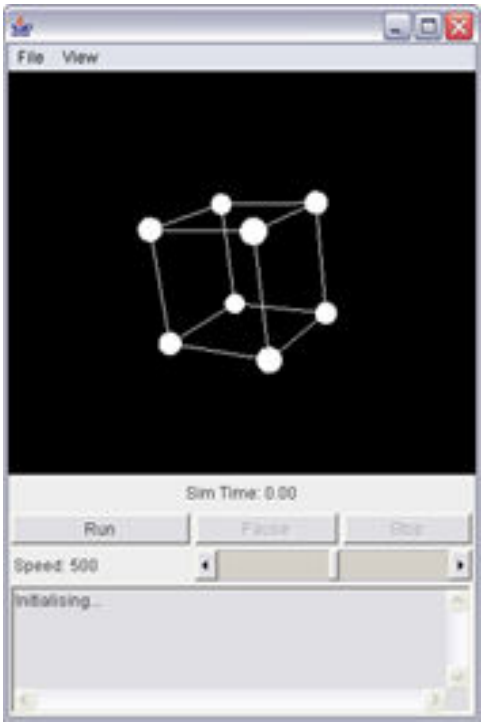

Fig. 5.1 Basic Hypercube Model

5.3 Bitonic Sort Model

The bitonic sort network is one of the fastest sorting networks whose lower bound complexity is Θ ($\log^2 n$). [21] [40] The important characteristic of this sorting network is that the sequence of comparisons is not data-dependent. In other words, unlike many other algorithms such as quick sort and merge sort, bitonic sort rearranges elements in fix time. Therefore, the algorithm is frequently employed in hardware or in parallel processor arrays.

The bitonic sort network is a comparison-based network model and like many others, the network contains a series of columns. Each column contains a number of comparators that are connected in parallel. In the case of the bitonic sort algorithm, the sorting of n elements

would involve log n columns or stages and each column or stage contains n/2 comparators

In this model, each entity or processor is mapped on to one element of an input sequence. The compare-exchange operations between two processors are performed only if the binary representation of their label differs in exactly one bit. Hence, mapping of the bitonic sort network onto a hypercube model is straightforward as two nodes in a hypercube are connected by an edge if and only if their labelling differs in exactly one bit position.

The algorithm embedded into each of the entities is shown below.

```
int A=0; //The data held by the entity
int R=0; //The received data

//dimention=dimention of the heypercube
for(int i=0; i<dimension; i++)
for(int j=i; j>=0; j--)
{
        int ii=i+1;
        int i1=dimension-ii;
        char c1=id.charAt(i1);

        int i2=dimension-j;
        char c2=id.charAt(i2);

        //If (i+1)st bit of the entity ID not equal to jth bit of the ID then
        if(c1!=c2)
        A=Math.max(A, R);
        else
        A=Math.min(A, R);


        //The entity with the id "000" will print the result of each stage
        if(Integer.parseInt(id, 2)==0)
        hyper.print_stage(i+1, i-j+1);
}
```

Dr. Cole at the University of Edinburgh kindly allowed this simulation model to be demonstrated for his students on the 17th of November, 2005. The outcome of this demonstration is discussed in the following chapter.

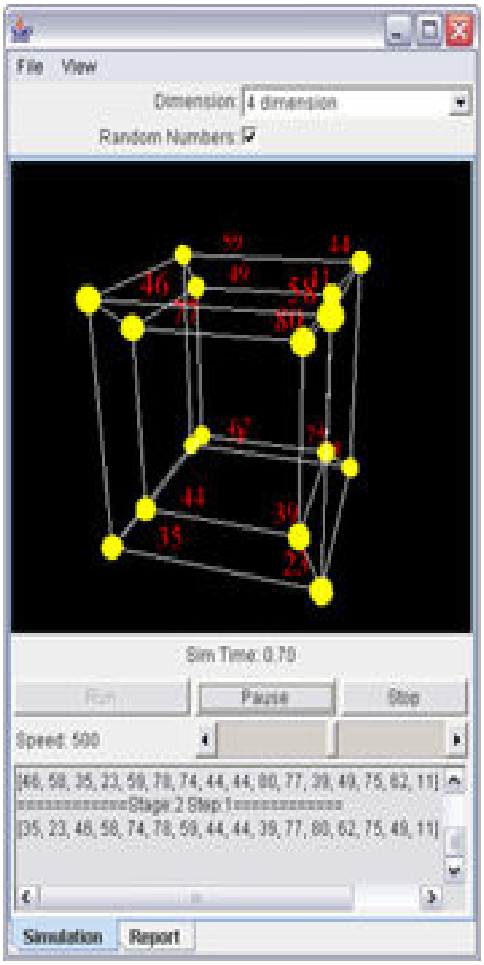

Fig. 5.2 Four-dimensional hypercube model running Bitonic Sort

5.4 Prefix Sum Model

In this example, the parallel prefix sum algorithm is implanted in each of the entities. The algorithm is presented below. Parallel prefix sum is another algorithm that is taught in many parallel computer science courses.

```
int S=0; //temporary variable
int T=0; //stores the sum of data held by all the entities
int A=0; //the initial data held by the entity
int B=0; //the received data

S=A;
T=A;

for(int i=0; i<dimension; i++)
{
        //send T to the entity whose ID differs from the ID of    this entity in exactly bit i.
        String dest=get_compliment(id, i);
        send( T, dest);

        B=recieve_data();
        int stage_index=dimension-1-i;

        if(id.charAt(stage_index)=='1')
        S=S+B;

        T=T+B;
}
```

5.5 Route Function Model

In the last hypercube example model, three routing functions are simulated. They are perfect shuffle, butterfly permutations and exchange permutations. A routing function is a useful mathematical description of a routing that maps source address to destination address. In general, when describing routing functions, source and destination addresses are represented of the form a_p….. a_i where the a_i are the individual bits of the binary integer. For example, if the integer is six, then a=110 and therefore a_2=1, a_1=1 and a_0=0. The behaviour of each of the routing functions is described below together with an example in the case of 8 sources and destinations.

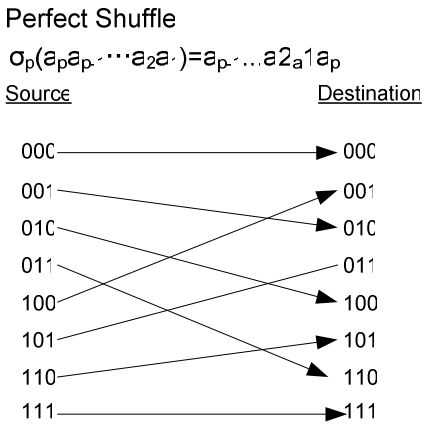

Fig. 5.3 Perfect Shuffle

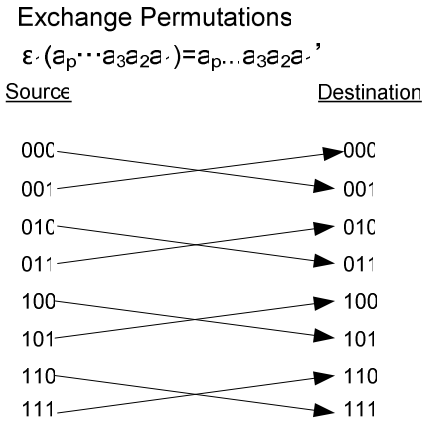

Fig. 5.4 Exchange Permutations

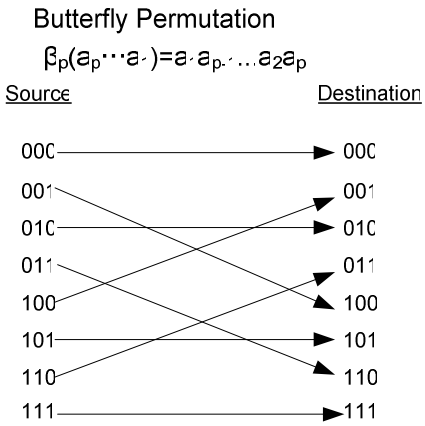

Fig. 5.5 Butterfly Permutation

5.6 Omega Network

The purpose of building this model is to demonstrate how easy it is to convert existing SimJava2 models into SimJava3 models. Here, we employed the omega network model in SmJava2 which was developed by Fred Howell. As a result, the only significant changes that were necessary were to use Anim_applet3d instead of Anim_applet and modify the coordinate values and size of entities and ports. This proved that the conversion of two-dimensional models to their three-dimensional equivalents is very straightforward.

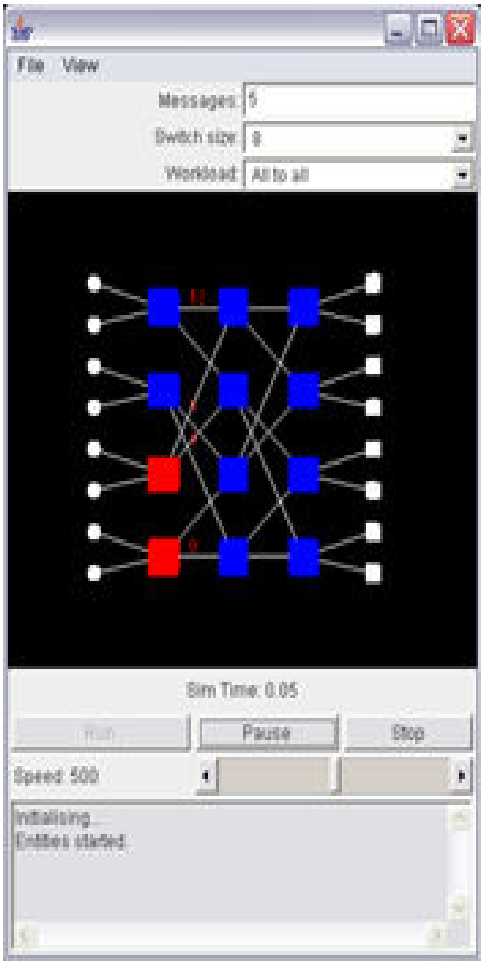

Fig.5.6 Omega Network

Chapter 6
Project Management and Testing

6.1 Introduction

Prior to this project, the author had some experience of being a member of large software development teams both in academia and in real world industry. This past experience has suggested that a methodical project management and testing approach is crucial to the successful creation of a quality product within the constraints. A haphazard attitude to a project will usually entail some serious flaws in a system which can quite often be expensive and chaotic.

The purpose of this section is to describe software engineering and testing techniques that have had a profound effect on the success of this project. The author strongly recommends the reader to spare some time to consider these issues, particularly if they are going to further develop this project.

6.2 Historical Issues and the Importance of Software Engineering and Software Testing

Software Engineering and, particularly, Software Testing are still very new disciplines in the field of computer science. The term "Software Engineering", meaning the systematic approach to a software development, was first recognised in 1970s. [63] Many techniques and papers that comprise many potential key determinants in conducting successful and efficient software development have been highly publicised.

Despite this, there has been a significant gap between the theory and the practice of software engineering. In spite of its huge potential usefulness in many software development organisations, its potential still remains largely unrealised. It is reported that about 75% of worldwide software development still has fairly immature software process development environments. [44] Furthermore, it is not far from the truth to say that Software Engineering still does not hold a strong position in universities' computer science curricula. This is quite an unsatisfactory situation and it is worthy of extensive discussion.

Software Testing is a fundamental procedure for delivering quality software within the constraints (e.g time, budget, working environment etc). It is the means by which people, methods, measurements, tools, and equipment are all integrated to test a software product. There are strong indications that anything between 40% and 70% of the development time for software is devoted to testing procedures. [38] It is no exaggeration to say that the

quality of a test process determines the success of software products. In accordance with the increasing complexity of software we build today, Software Testing has evolved considerably. However, many organisations still regard Software Testing as an after-development activity and greatly underestimate its importance. Testing activities still remain subordinate to development.

Therefore, in order to emphasise the importance of Software Engineering and Software Testing issues, the author attempts to demonstrate their effectiveness by applying some of the standards and well-proven Software Engineering and Software Testing techniques to this project. Whist human resource management is an important issue in Software Engineering, as the entire project was developed by the author working alone, it is not our present concern.

The tests were mainly carried out on a machine with the following hardware configurations.

CPU: 2 GHz
RAM: 512MB DDR
OS: Windows XP
Graphics card: nVidia 62MB
Network: School of Informatics at University of Edinburgh

6.3 Application and Description of the Applied Technique

Fig. 6.1 indicates the main stages involved in the course of developing and testing this project. The project combines the notion of the standard iterative lifecycle model with the Dotted-U Model introduced by The Software Development Technologies. [38]

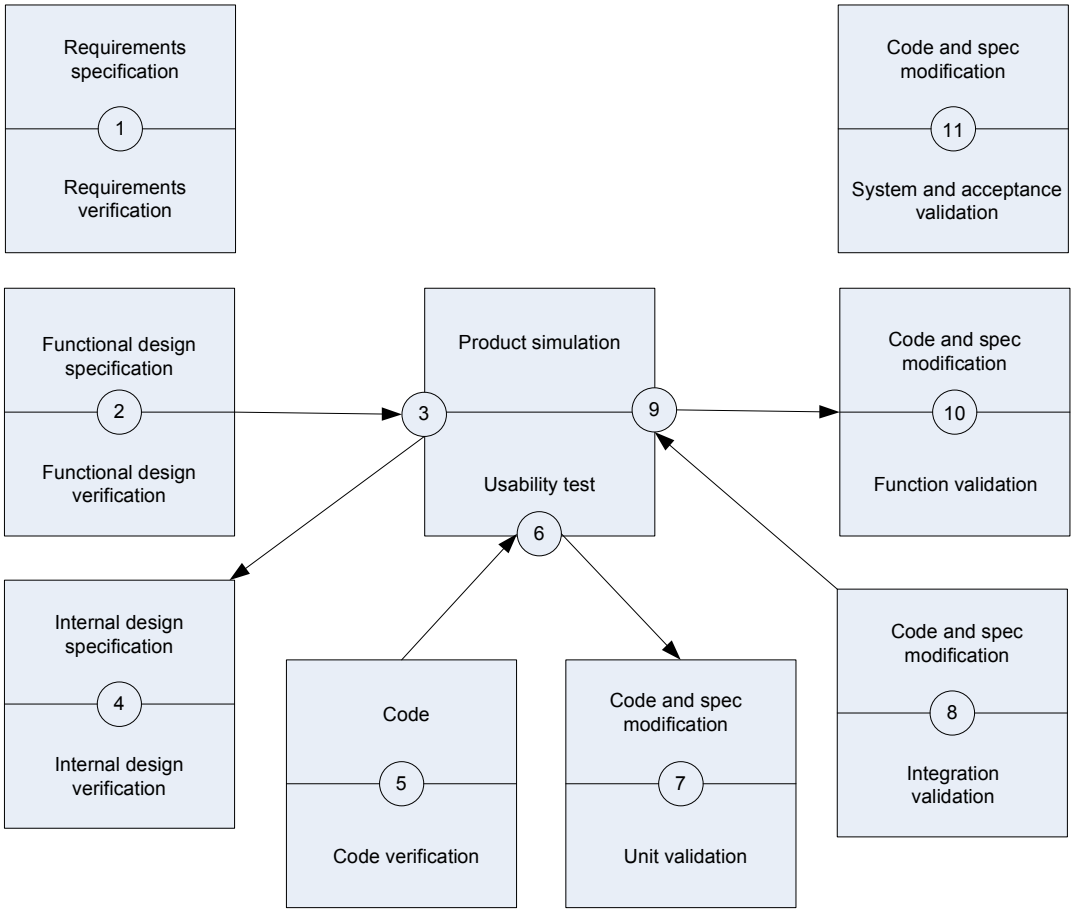

Fig 6.1 SDT Dotted-U Model

What is unique about this lifecycle is that unlike in the case of many standard software development lifecycles, we do not place the testing phase is not placed at the end of a development cycle. Rather, we regard the testing procedure as having its own lifecycle that runs in parallel with the development cycle. Therefore, the testing process is considered to be complementary to the development process and not a part of the development phase. By doing this, we can perform efficient and constructive tests throughout the entire lifecycle. Furthermore, errors can be detected in the early stages of development.

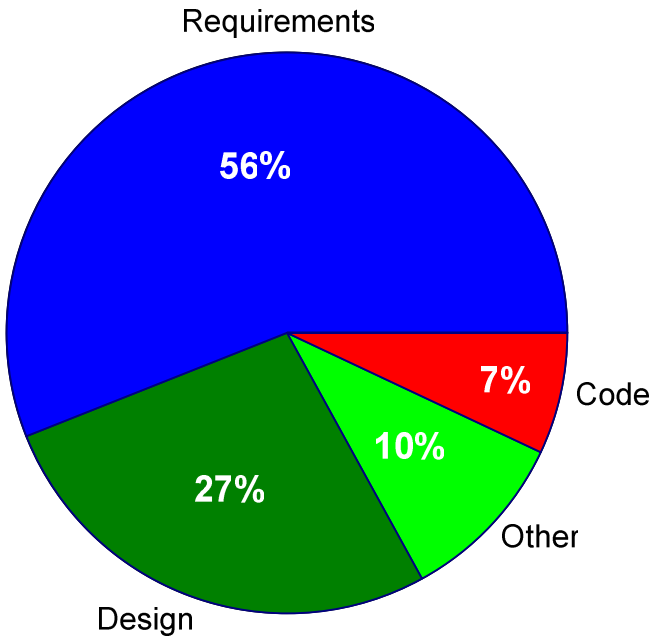

Fig. 6.2 Defect distribution

Strong evidence provided by Dick Bender suggests that the majority of errors are concentrated in the earlier stages of the development cycle. (Fig. 6.2) [38] There are strong indications that more than 50% of all defects are typically introduced in the requirement stage alone. [11] Errors, particularly those found in the early stages, can be quite troublesome and have the unpleasant characteristic of migration. If errors are undetected and allowed to migrate downstream, the cost of correcting them in the later stages of development will be considerably higher.

As in the standard iterative lifecycle, at the end the lifecycle, a deliverable is produced. The lifecycle is repeated and deliverables are accumulated until the whole system is built.

Software Testing is mainly composed of two processes, verification and validation. In short, verification involves desk checks of documents such as requirement specification, design specification and source code. It is a human examination or review of the work product. On the other hand, validation involves analysis of the execution of software. The lifecycle contains four verification processes and six validation processes. The following sections provide brief descriptions of each of the stages.

6.3.1 Risk Analysis

One can safely state that the creation of an absolutely error-free product is impractical because of the exhaustive testing it would require. Even if it were possible, the cost of the resulting product would be extremely high. Thus, compromises are inevitable. Instead, what we should aim for is the good use of testing resources or, in other words, cost-effective testing. In order to address this issue, careful judgement was exercised in order to create a hierarchy of testing procedures in order of priority.

The following is a list of the criteria by which priority was assigned to the various testing procedures.

1. Criticality: Perform rigorous tests on those parts of the system where their errors could result in a serious consequence. Those are defects that can affect the reliability of the system or disobey the user requirements.
2. Frequency: Focus tests on the parts of the system which are frequently used by users. For example, the project compilation method may be the most frequently used method in the environment. Flaws in the method would seriously degrade the environment's usability and therefore need particular attention.
3. Defect probability: Test the components of the system which are suspected to contain high proportion of errors.

Based on these criteria, intensive and careful testing procedures were carried out on the critical parts of the system, and a brief testing process for those parts which were less critical.

6.3.2 Verification

In general, most lifecycles are validation oriented. However, verification is proven to be more cost-effective and thorough than validation testing. Verification can find many errors that are impossible to detect through the validation process. Furthermore, it finds defects at the earliest possible time, thereby preventing unpleasant error migration and removing the necessity for huge development efforts.

In order to achieve a high degree of verification, three types of formal documents were produced. They are a requirements specification, a functional design specification and an internal design specification. During the course of this project, they were faithfully maintained and any changes to the project were immediately reflected in all documents. These documents were inspected at each phase of the development and testing lifecycles.

6.3.2.1 Requirements Verification

The purpose of the requirements verification is to make sure that the users' needs are properly understood. Requirement verification determines the capability of the software that is being developed and establishes the intended purpose of that software. Typically, changes to a requirement specification are inevitable and these changes need to be maintained.

6.3.2.2 Functional Design Verification

Functional design specification contains a set of external human interfaces of a product. It describes all behaviour of the product seen by users. Functional design specification does not concern any material that is not visible to the users such as the internal data structure or data diagrams. Functional design verification is a process of checking whether the requirements verification is successfully translated into a functional design verification. All items described in the requirement specification need to be translated into a functional design specification. Once the functional design verification is complete, an internal design specification is derived from the functional design verification. The internal design specification shows how a product is going to be built, with detailed descriptions of data structures, data flows and algorithms. Moreover, it clarifies the framework of the software.

6.3.2.3 Code verification

Coding involves the process of translating the internal design specification into source code. Many organisations regard this phase as their only testing activity. Examples of activities involved in this phase are comparing the source code with the internal design specification, examining the code against a language-specific checklist, and finding the cause of performance bottlenecks.

6.3.3 Validation

The black box technique is a testing strategy that tests the functional and behavioural aspects of the system without regard for the internal program structure. The white box technique, on the other hand, is a testing technique that tests the structural aspects of the system and requires explicit internal knowledge of the system.

For each validation phase, commonly used black-box methods were applied. They included equivalence partitioning, boundary-value analysis and error guessing. White-box methods were used in the most critical parts of the system. For instance, the I/O modules which open

and save a Sim file are called every time modellers use the simulation environment. Errors in this module would seriously degrade the system's usability and discourage users from using the environment. Hence, extra attention is necessary in order to preclude these kinds of errors.

6.3.3.1 Unit Testing

Unit testing involves the process of testing the individual components of the system. This detects inconsistencies between the components' internal specification and their actual behaviour.

6.3.3.2 Integration Testing

Integration testing combines the components tested in the previous phase and attempts to find errors in the interfaces between the components. If any errors are found they are most likely to be discovered in the most recently added component.

6.3.3.3Usability Testing

Usability testing measures how easy it is for users to use the program. The test attempts to measure the accessibility, responsiveness, efficiency and comprehensibility of the system.

6.3.3.4 Function Validation

The purpose of function validation is to find any discrepancies between the functional specifications of the system and its actual behaviour. When errors are discovered, the functional specifications and the program are revised.

6.3.3.5 System and Acceptance Validation

System and acceptance validation measures the consistency between the final version of the system and its original requirements and objectives. Acceptance validation measures the degree of user satisfaction with the project. In order to address this, demonstrations of the project were carried out with diverse audiences. From each of the demonstrations, invaluable feedback was collected.

Chapter 7
Evaluation

7.1 Introduction

This chapter is devoted to a discussion of the results of this project. Possible explanations and implications for the findings that were discovered during the course of the development phase are discussed.

Overall, in comparison with the requirement specifications, the project appears to have satisfied a majority of the objectives. The project adequately meets the main targets set at the beginning. However, due to time constraints, some of the desired features were unrealised. Further discussions about them are presented in the following chapter.

The testing procedure confirmed that all the software products are available for the Windows and Linux platforms. Insufficient tests were carried out to confirm that they are compatible with other platforms, however.

7.2 SimJava3

The first part of the project has successfully produced a three-dimensional platform-independent simulation library. The library provides users with a suite of useful tools for creating dynamic and interactive simulation models in three-dimensional space. In comparison with its predecessor, this new version of SimJava incorporates the capability of building and examining simulation models that possess a complex topology. Furthermore, the simulation models can be viewed from whatever perspective that the users desire. For example, if we were to model a torus network layer with the previous version of SimJava or HASE, the model would necessarily incur many link crossings and leads to confusion. Simjava3 fills this void by effectively capturing the dynamic characteristics of a real world system, enabling the users to thoroughly analyse the model.

The new version of SimJava inherits much of its precursor's capabilities except for SimDiag. Due to the tight schedule, it was not possible to perform enough tests to ensure that the new version was compatible with the SimDiag package. As far as the syntax is concerned, there are not many obvious changes in the new version. Therefore, it should be easy for previous SimJava users to become familiar with the syntax of the new version. In addition, the conversion of existing two-dimensional models into three dimensional models is very straightforward. It is worth pointing out that there are no differences in the size of class files between three-dimensional models and two dimensional models. Therefore, the downloading time for a three-dimensional model is equal to that of a two-dimensional

version.

Despite all these virtues, there is a price to be paid for its inspiring visualisation. It is apparent that, under whatever hardware or software configuration, the new version takes a relatively longer start-up time compared to the previous version. In the testing process, the new version took 5 seconds to load an applet whereas the previous version took only 2 seconds. Whether 5 seconds is an acceptable download time is a highly controversial issue in the Human Computer Interaction discipline. Interestingly and ironically, every HCI expert believes in a different tolerable waiting time. Nielsen claims a 10-second limit, whereas Shneiderman proposes a 2-second rule. Zona Research claims an 8-second rule, while Hoxmeier and DiCesare claim a 12-second rule. [20] [42] Therefore we are unable to determine whether a 5-second start-up time indicates good usability.

The performance of a simulation run is affected by the underlying hardware configuration. For example, when the navigation type is set to the walk mode, users can walk through a three-dimensional environment. If the model is run on a modern machine with fast graphics hardware, then the walk speed becomes reasonably fast. On the other hand, if the model is run on a low-power machine, then the walk speed becomes much slower. This is certainly not a trivial matter as it greatly reflects on the usability of the program and yet this problem remains unsolved.

Some variations in SimJava3's performance were observed depending on which low graphics APIs its Java3D was run upon. In general, simulation models appear to run faster with Direct3D-based Java3D. However, in some circumstances, OpenGL-based Java3D prevailed over Direct3D-based Java3D. An experiment was carried out to investigate this anomalous behaviour. The four-dimensional hypercube running the bitonic sort algorithm, which was created in the third part of the project, was used as a testing model. In addition, two variations of the model were presented. For the first version, all the entities were of type BOX and for the second version, all the entities were of type SPHERE. These models were run on Direct3D-based Java3D and OpenGL-based Java3D respectively under the same hardware and software configuration and with an equal simulation speed.

As a result, the difference in execution time is greater when running the testing model with BOX type. This is considered to be the case as OpenGL has one function for drawing a whole cube. On the other hand, Direct3D needs to break the cube into triangles, which results in many function calls and a significant decline in performance.

By default, if we try to run a simulation model with a large number of entities, Java3D may complain about insufficient memory. This is due to the fact that the Java virtual machine allocates a certain amount of memory for use on any given program, which is much smaller than the amount of memory on the average home system. However, when dealing with three-dimensional graphics in any circumstance, it is necessary to allocate an adequate amount of memory to the program.

In order to increase the memory limit, we use the –Xmx command for the running of the program. For instance, the following command reserves a maximum of 64 MB of space to be used by a Java Virtual Machine.

Java –Xmx64m program

Users may quite often experience a mouse flicker when the mouse pointer is placed over the canvas. If the problem becomes severe, users may experience some frustration as the task of selecting an entity, rotating a simulation model and checking the parameter values of the entity becomes more difficult. The problem was observed only for Win32 implementations. There are several possible explanations for its cause. One possible explanation derives from the fact that Java3D continuously attempts to gain 100% CPU utilization in order to achieve maximum performance. If this is the case, the condition may be improved by lowering the priority of all the Java3D threads. This can be done by using the static method, setJ3DThreadPriority(). Another possible cause is that the video driver could be outdated, and a driver update may improve the situation. The last possible resolution to the problem is to disable the pointer shadow.

In the course of the testing procedure, very occasionally, some errors that are intrinsic to Java3D were observed. As those errors occurred so rarely, the majority of them were not repeatable and hence the complexity of fixing them became an issue.

Some inadequacies in Java3D were discovered when it was executed on some versions of Java. Due to some fundamental internal changes in JDK 1.4, Java3D was not completely compatible with JDK 1.4. This incompatibility resulted in Java3D crashing on start-up. Sun's Java3D's website recommends the use of either Java3D 1.4 or 1.2.1_04 or later versions when JDK 1.4 is being used with Java3D.

In the case of JDK1.5, there was a flaw in JDK 1.5 Beta 1 which was due to the fact that the DLLs could not be found without explicitly setting the LD_LIBRARY_PATH environment variable. However, this error was later reported as fixed in the beta 2 version.

As for the previous version, the new version gives the users an option to add graphical user interface components into the applet by overriding the anim_init() method. In theory, users can add the AWT components or the SWING components. The library will not complain to users if they use the either of them. However, we recommend to users employ only the AWT components, as Java3D is still not completely compatible with SWING. Java3D uses heavyweight components but SWING uses lightweight components. Therefore, using the SWING components may provoke strange errors or anomalous behaviour in the program.

The application of methodical testing, as described in the previous chapter, has allowed this part of the project to achieve a high degree of reliability.

7.3 Simeon

In the second part of the project, a simulation environment, Simeon, was developed for SimJava3. Although SimJava3 provides useful tools for the easy creation of multidimensional simulation models, users would have to devote much of the development time to geometrical calculation. Statistical measurements, a transient condition, a termination condition, an output analysis and report details have to be set manually. Therefore, a great deal of knowledge about the simulation library would be required in order for the users to comfortably develop a simulation model. This involves numerous coding and checking processes and prevents the users from concentrating on the behavioural aspect of the simulation model.

In order to address this issue, the simulation environment now features an intuitive and interactive graphical user interface. With the aid of this program, users can create simulation models with the least amount of coding, which in turn results in a significant reduction of development time. Furthermore, the program anticipates and prevents common user errors such as setting a minimum and maximum transient condition and any termination condition at the same time. The program verifies every important action that the users attempt to make (deleting project etc)

In order to measure the usability of the simulation environment, the program was inspected by people from a diverse background. Among them, particularly invaluable suggestions were received from Human Computer Interaction (HCI) expert Stephen Lu. [42] This feedback had considerable influence on the user interface of the program, achieving a higher degree of usability. Many revisions to the user interface were made and the final version allows the users to easily navigate between program screens. Multiple paths are provided to the most commonly used features.

Simeon shares certain features with HASE. On the other hand, in contrast, Simeon is capable of developing and running multiple simulation models in parallel. However, these simulation models do not advance their simulation time in a synchronous manner. If we are to coordinate the simulation time between the simulation models, significant changes to the kernel of the simulation library are inevitable.

As far as the reliability of the simulation environment is concerned, despite the application of a methodical testing procedure, further work remains to be done to overcome some remaining issues.

Intensive testing processes were carried out on the critical parts of the system, removing many of the fatal errors. Some errors are still concentrated around the source code editor. The environment sometimes fails to reflect changes made to the source code of a project. Each user has a different programming style and it is difficult for the environment to handle

every extreme of their behaviour. Therefore, despite the fact that this interfered with the usability of the system, it was necessary to enforce that the users follow the specific programming style.

The majority of errors remaining in the system are purely of a semantic aspect and do not affect the overall software integrity. However, necessary protections were installed into the software. If users ever encounter an error in the system, a dialogue window appears and they are asked to report the error to the developer team by means of an e-mail.

On the whole, the program can be considered adequately reliable and robust.

7.4 Example Models

In the final part of the project, some example simulation models were produced. The purpose of these was to ensure and demonstrate the reliability of the simulation library and the simulation environment. Moreover, they were intended to give the users an opportunity to become acquainted with the syntax of the simulation library.

The demonstration of the bitonic-hypercube model confirmed that the model successfully enhanced the students' understanding of this complex computer science concept. Furthermore, the virtue of the model being platform-independent and portable allowed students to examine the model more closely in their own time. Thus, it appears that the model helps both the students and the course deliverer.

The omega network simulation model proved that the conversion of existing two-dimensional simulation models into three-dimensional equivalent is very straightforward.

Chapter 8

Conclusion

8.1 Conclusion

This chapter serves as the summary of the project together with some last thoughts on the completed work. Several achievements and deficiencies identified at the conclusion of this project are discussed.

Prior to this project, application developers were united in their unquestioning belief that performing intense computation, particularly involving three-dimensional graphics over the internet, causes far too much overload with respect to the bandwidth and to the machines on which the program is run. Furthermore, the project immediately faced difficulty in finding sufficient relevant literature and resources.

Despite the huge improvement in bandwidth and computational power, web-based simulation is still a very early discipline in the history of computer science. As far as Java3D is concerned, there are still very few research projects that exploit the technology and there were very limited resources to acquire its skill. On the contrary, no substantive attempt at building a simulation library with three-dimensional visualisation had been made. This created considerable skepticism as to whether the project was possible. For that reason, the project was initiated with a careful and extensive feasibility study. Following this, a number of prototypes with increasing complexity were produced. They produced satisfactory results meeting most of the functional requirements. The success of those prototypes led to the conviction to continue the project.

The first part of the project successfully produced what is considered to be the very first web-based platform-independent three-dimensional simulation library in the world. The simulation library provides a suite of tools that enables the creation of simulation models involving complex topology. The careful selection of graphics technology, based on extensive background research, was a significant aid to the development of the simulation library. The Java3D class library offers a toolbox for defining and building three-dimensional scenes with ease and separates out many of the low-level processing tasks.

The integration of Java3D and SimJava not only presents models in appealing three dimensional graphics, it is also capable of encapsulating complex concepts involving multidimensional characteristics, and performs simulations in a dynamic and in an

interactive platform without sacrificing performance. When statistical measurements are incorporated into a simulation model, at the completion of a simulation run, the library presents statistical results in a different tab. Those results can also be saved in a text format for further analysis.

The simulations can be performed in stand-alone or applet fashion and the simulation models can be examined from any perspective that the users desire. The applet enables users to embed live diagrams into a web document which can be viewed by machines with different hardware and software configurations. The performance of the simulation library as a whole was investigated in several machines with heterogeneous platforms. In each case, the simulation library seems to have produced satisfactory results.

In the second part of the project, a simulation environment for SimJava3 was developed. As the scale of the software was considerable, a significant amount of effort was dedicated to this part of the project.

Although simJava3 provides the building blocks for creating multi-dimensional simulation models, much of the development time had still to be devoted to geometry work. This involves daunting and frustrating tasks with numerous coding and checking processes which frequently entail trivial errors. The simulation environment fills this void with an intuitive graphical user interface that will allow users to concentrate solely on the behavioural aspect of the models and enable them to build their own simulation applications very quickly. Moreover, its effective and efficient I/O management allows modellers to export and save the simulation models in a most convenient way.

It has been a great regret that some of the desired features could not be realised due to lack of development time. Prominent in the list of uncompleted desired features is the provision of authoring tools for two-dimensional simulations. Since the previous version of SimJava could already perform two-dimensional simulations, this task is expected to be straightforward. A detailed description of possible further improvements to the project is presented in the following section.

The third part of the project uses the results of the first two parts and produces some example simulation models. The opportunity to demonstrate the bitonic-sort model in front of MSc students was a huge encouragement and became an important driving force in delivering the project. This demonstration and the feedback from students confirmed that the model can effectively enhance the students' understanding of a difficult computer science concept that involves complex topology.

As well as the technical aspects of the project, some personal achievements were also made. From the beginning of the project, the scale of the project was thought to be substantial. Chief among concerns of this nature was the sheer size of the software. Ultimately, at the time of writing this report, the total number of lines of code involving this project has become approximately 50,000 lines. Moreover, additional constraints such as time and

limited resources were hard to overcome. Therefore, a systematic and efficient approach to the testing and project management was necessary. The project took advantage of some of the established and well-proven testing and project-management techniques. The application of such techniques had a profound effect on the success of this project. In addition, the project provided invaluable experience and skills in tackling a large software project. Despite the application of this procedure, however, the software still retains a number of flaws and hence there is still scope for future research on the matter. The experience of conducting this project has broadened the author's scope concerning the practical issues in the real world industry today.

To conclude, the project has not only contributed greatly to the advancement of computer simulation and visualisation but also has a particular importance in proving the potential and possibility of what online applications can do over the Internet today. The latest version of SimJava is truly a pragmatic tool for simulating models where dynamic visualisation and simulation speed are paramount. On the whole, the project has been a great success, most of the objectives attained and the project outputs realised. Finally, the author wishes for the further development of the project in the future.

8.2 Future Development

There are various ways in which this project can be developed further.

Currently, entities are represented as cubes, spheres, cylinders and cones. Another possible enhancement for SimJava3 is to allow modellers to import custom three-dimensional graphics contents. This is already known to be feasible as the latest version of Java3D supports the loading of a wide variety of three-dimensional modelling file formats. Such formats include OpenFlight, 3D Studio (3DS), VRML, X3D, Wavefront OBJ and many other formats.

There is still scope for performance optimisation of SimJava3. Several Java3D techniques which accomplish this are available on the Internet. As simulation models are complex programs that are typically harsh on system resources, it is worthwhile considering this issue.

This project provides some example simulation models. However, these models are insufficient to fully assess the potentiality of the simulation library. Initially, it was hoped that the project would be able to create more complex models, but due to time limitations, this was not possible.

As mentioned in chapter three, the idea of using interpolator nodes was rejected due to the large consumption of system resources. However, we can develop a custom interpolator node class, in order to realise smoother animations of packet transfers.

Frequently, in many computer science textbooks, a four-dimensional hypercube is represented with two cubes that are connected with arcs. Allowing modellers to link entities with arcs could add a different flavour to their simulation models.

With respect to Simeon, the simulation environment could allow modellers to develop two-dimensional simulation models. Since the previous version of SimJava already had a reliable support for creating two-dimensional simulation models, this extension is promising. Furthermore, there is a possibility that the conversion of two dimensional simulation models to three-dimensional equivalents and vice versa can be automated by the simulation environment.

Despite the fact that the simulation environment allows modellers to develop and run simulation models concurrently, those models are not executed in a synchronous manner. Thus, another possible enhancement to the environment would be to synchronise the simulation time of models that are executed in parallel.

Appendix

The source code, the class files and the instructions of running the programs can be found on http://www.asukaishikawa.com/project/index.htm

Bibliography

[1] Adams J. (2003), “Advanced Animation with DirectX”, Course Technology

[2] Akingbade A., Finley T., Jackson D., Patel P., and Rodger S. H. (2003), “JAWAA: Easy Web-Based Animation from CS0 to Advanced CS Courses”, Proceedings of the Thirty-Fourth SIGCSE Technical Symposium on Computer Science Education Reno, Nevada, USA, pp. 162-166

[3] Alistair E.R. Campbell, Geoffrey L. Catto, and Eric E. Hansen (2003), “Language-Independent Interactive Data Visualisation”, Proceedings of the Thirty-Fourth SIGCSE Technical Symposium on Computer Science Education Reno, Nevada, USA, pp. 215-219

[4] Ande R. N., Liebrock L. M. (2004), “Integrating VRML, Java, XML, and HTML in a web-based tool”, Proceedings of the 2004 Summer Computer Simulation Conference, San Jose California, pp. 513-519

[5] Angel E. (2004), “OpenGL: A Primer”, Addison Wesley

[6] Asian Institute of Technology, “SimProd:Tool for developing flexible simulation models of production systems on the web”, (http://www.geocities.com/ResearchTriangle/Lab/5232/SimProd.html)

[7] Axford T.H. (2002) , “Lecture Notes on Real-time Systems Programming”, School of Computer Science, University of Birmingham, UK

[8] Axford T. H. (2003), “Parallel Computer Architecture Lecture Notes”, School of Computer Science, University of Birmingham, UK

[9] Banks J., Carson J. S., Nelson B. L., Nicol D. M., and Nelson B. (2000), “Discrete Event System Simulation”, Prentice Hall

[10] Barrilleaux J.(2000), “3D User Interfaces With Java 3D”, Manning Publications

[11] Budgen D. (1994), “Software Design”, Addison Wesley,

[12] Carey R., and G. Bell G. (1997), “The Annotated VRML 2.0 Reference”, Addison-Wesley

[13] Chan P. (2002), ”The Java Developers Almanac: Version 1.4 Pt.A” , Addison Wesley

[14] Chan P. (2002), “The Java Developers Almanac: Version 1.4 Pt.B“, Addison Wesley

[15] Chen C. (2002), “Introduction to Visualisation and Virtual Environments”, Springer

[16] Crispen B. (2000), ”VRML Works”, (http://vrmlworks.crispen.org/history.html)

[17] Coe P. S., Howell F. W., Ibbett R. N., and L. M. Williams (1998), “Technical note: a hierarchical computer architecture design and simulation environment”, Volume 8 , Issue 4 , Special issue on Web-based modelling and simulation, ACM Press, pp.431 - 446

[18] Couch J. (2005), “Java 3D FAQ”, (http://java3d.j3d.org/faq/)

[19] Diehl S.(2001), “Distributed Virtual Worlds : Foundations and Implementation Techniques Using Vrml, Java, and Corba”, Springer Verlag

[20] Dix A., Finlay J., Abowd G., and Bale R. (2003), “Human Computer Interaction”(3rd edition), Prentice Hall

[21] Grama A., Gupta A., Karypis G., and Kumar V. (2003), “Introduction to Parallel Computing (2nd edition)“, Benjamin Cummings

[22] Harold E. R., and Means W. S. (2004), “XML in a Nutshell, Third Edition” (3rd edition)”, O'Reilly Media

[23] Henry Sowizral, Dave Nadeau(1999), “Introduction to Programming with Java 3D”, University of California(San Diego), July 1999

[24] Hilston J. (2003), “Modelling and Simulation lecture notes”, School of Informatics, University of Edinburgh

[25] Horstmann C., and Cornell G. (2004), “Core Java(TM) 2, Volume I—Fundamentals” (7th Edition), Prentice Hall

[26] Horstmann C., and Cornell G. (2004), “Core Java(TM) 2, Volume II--Advanced Features” (7th Edition), Prentice Hall

[27] Howell F., and McNab R. (1996), "Using Java for Discrete Event Simulation", Proceedings of the Twelfth UK Computer and Telecommunications Performance Engineering Workshop (UKPEW), University of Edinburgh, pp. 219-228.

[28] Howell F., and McNab R. (1998), "simjava: a discrete event simulation package for Java with applications in computer systems modelling", Proceedings of the First International Conference on Web-based Modelling and Simulation, San Diego CA, Society for Computer Simulation

[29] Howell F. W., Williams R., and Ibbett R. N. (1994), “Hierarchical Architecture Design and Simulation Environment Source”, Proceedings of the Second International Workshop on Modelling, Analysis, and Simulation on Computer and Telecommunication, IEEE Computer Society, pp. 363 – 366

[30] Howell F., “The SimJava homepage”, The Institute for Computing Systems Architecture, School of Informatics, University of Edinburgh, (http://www.icsa.inf.ed.ac.uk/research/groups/hase/simjava/)

[31] Hunter D., Watt A., Rafter J., Cagle K., Duckett J., and Patterson B. (2004), “Beginning XML”, John Wiley & Sons

[32] ICSA (The Institute for Computing Systems Architecture), “HASE User Guide”, School of Informatics, University of Edinburgh, (http://www.icsa.inf.ed.ac.uk/research/groups/hase/manuals/userguide/)

[33]ICSA (The Institute for Computing Systems Architecture), ”Hase++ Design documentation”, School of Informatics, University of Edinburgh, (http://www.icsa.inf.ed.ac.uk/research/groups/hase/manuals/design/hasepp.html)

[34] ICSA (The Institute for Computing Systems Architecture),”Hase++ Reference Manual”, School of Informatics, University of Edinburgh, (http://www.icsa.informatics.ed.ac.uk/research/groups/hase/manuals/hasepp/hasepp.html)

[35]ICSA (The Institute for Computing Systems Architecture), ”JavaHase Design documentation”, School of Informatics, University of Edinburgh, (http://www.icsa.inf.ed.ac.uk/research/groups/hase/manuals/design/javahase.html)

[36] ICSA (The Institute for Computing Systems Architecture), ”The HASE homepage”, School of Informatics, University of Edinburgh, (http://www.icsa.inf.ed.ac.uk/research/groups/hase/userguide/index.html)

[37] ICSA (The Institute for Computing Systems Architecture), “The JavaHASE homepage”, School of Informatics, University of Edinburgh, (http://www.icsa.inf.ed.ac.uk/research/groups/hase/javahase/)

[38] Kit E.(1995), “Software Testing In The Real World” (4th edition), Addison Wesley

[39] Kuchana P. (2004), “Software Architecture Design Patterns in Java”, AUERBACH

[40]Lang H. W., and Flensburg F. H.(2004), “Bitonic Sort”, (http://www.iti.fh-flensburg.de/lang/algorithmen/sortieren/bitonic/bitonicen.htm)

[41] Little C. M.(1997), “C++SIM Home Page”, Department of Computing Science, The University, Newcastle upon Tyne, UK (http://cxxsim.ncl.ac.uk/)

[42] Lu S., Datanomic Limited, Private Communication

[43] Luna F. D. (2003), “Introduction to 3D Game Programming with DirectX 9.0”, Wordware Publishing

[44] Maciaszek L. A., and Liong B. L. (2005), “Practical Software Engineering”, Addison Wesley

[45] Mark K. (1996), “OpenGL for the X Window System”, Addison Wesley

[46] McCarthy M., and Descartes A. (1998), “Reality Architecture: Building 3D Worlds in Java and VRML”, Prentice Hall

[47] McNab R., Simatos C., Howell F. (2002), “SimJava v2.0 API Specification”, University of Edinburgh, (http://www.icsa.inf.ed.ac.uk/research/groups/hase/simjava/doc/)

[48] Microsoft Corporation, “DirectX Developer Center”, (http://msdn.microsoft.com/directx/)

[49] Microsoft Corporation, “DirectX Home Page”, (http://www.microsoft.com/windows/directx/default.aspx)

[50] Murshed M., and Buyya R. (2002), “Using the GridSim Toolkit for Enabling Grid Computing Education “, International Conference on Communication Networks and Distributed Systems Modelling and Simulation, San Antonio, Texas, USA

[51] Murshed M., and Buyya R. (2002), “GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing”, The Journal of Concurrency and Computation: Practice and Experience, Volume 14, Issue 13-15, Wiley Press

[52] OpenGL.org (2004), (http://www.opengl.org/)

[53] Page E.H., Moose R.L., and Griffin S.P. (1997), “Web-Based Simulation in Simjava using Remote Method Invocation”, Winter Simulation Conference, Atlanta, GA

[54] Page E. H., “Distributed Simjava”, MITRE Corporation , (http://www.thesimguy.com/Projects/websim/simjava/HAZE)

[55] Patterson I. (1997), “Working model of the internals of the DLX processor”, MSc. Project report, University of Edinburgh

[56] Pooch U. W., and Wall J. A. (1992), “Discrete Event Simulation: A Practical Approach (Computer Engineering S.)”, CRC Press

[57] Post F. H., Nielson G. M., and Bonneau G. P. (2002), “Data Visualization: The state of the art “, Kluwer Academic

[58] Ramsay S. (1997), “Simulation of distributed commit protocol”, MSc. Project report, University of Edinburgh

[59] Ranka S., and Sahni S. (1990), “Hypercube Algorithms with Applications to Image Processing and Pattern recognition”, Bilkent University Lecture Series, Springer-Verlag New York

[60] Selman D. (2002), “Java 3D Programming”, Manning Publications

[61] Simatos C. (2002), “The SimJava Tutorial”, University of Edinburgh, (http://www.icsa.inf.ed.ac.uk/research/groups/hase/simjava/guide/tutorial.html email: C.Simatos@sms.ed.ac.uk).

[62] Simatos C. (2002), "Making SimJava Count", MSc. Project report, University of Edinburgh

[63] Sommerville I. (2001), “Software Engineering” (6th edition), Addison Wesley

[64] Sun Microsystems, “Sun's Java3D Homepage”, (http://java.sun.com/products/java-media/3D/)

[65] Sun Microsystems,”Official Java3D FAQ”, (http://java.sun.com/products/java-media/3D/forDevelopers/java3dfaq.html)

[66] Walsh A. E., Derringer D., and Gehringer D. (2002), “Java 3D API Jump-Start”, Prentice Hall

[67] Woo M., Neider J., Davis T., Shneider D. (1999), “OpenGL Programming Guide” (3rd edition), Addison-Wesley