

LAUNCH PAD FOR EDUTAINMENT SOFTWARE SUITE

By

K.A.S.N.Sumathipala

(03/AS/010)

**This thesis is submitted in partial fulfillment of the requirement for the degree of Bachelor of Science
in Physical Sciences of the Faculty of Applied Sciences, Sabaragamuwa University of Sri Lanka**

Department of Physical Sciences & Technology

Faculty of Applied Sciences

Sabaragamuwa University of Sri Lanka

Belihuloya


MARCH 2009

DECLARATION BY CANDIDATE

I hereby declare that this thesis is my own work and effort and that it has not been submitted anywhere for any award. Where other sources of information have been used, they have been acknowledged.

..22|04|2009..

Date


.....

K.A.S.N.Sumathipala

CERTIFICATE OF APPROVAL



We hereby declare that this thesis is from the student's own work and effort, and all other sources of information used have been acknowledged. This thesis has been submitted with our approval.

Dr. R.G.N. Meegama

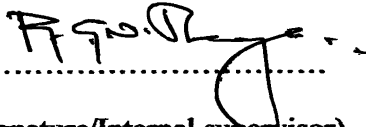
Senior Lecturer

Department of Statistics & Computer Science

Faculty of Applied Sciences

University of Sri Jayewardenepura

Nugegoda.


.....
(Signature/Internal supervisor)

.....22/04/2009.....
(Date)


Mr. R. Maddegoda

Consultant- Technology

Virtusa Corporation

Sir Chittampalam A. Gardiner Mw

Colombo 2.


.....
(Signature/External supervisor)

.....07-04-2009.....
(Date)

Dr. C.P.Udawatte

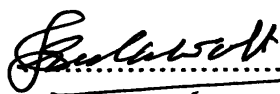
Head of the Department

Department of Physical Sciences

Faculty of Applied Sciences

Sabaragamuwa University

Belihuloya.


.....
(Signature/Head of the Department)

.....2009-04-22.....
(Date)

*Affectionately Dedicated To My Loving Family
Members and Teachers*

ACKNOWLEDGEMENT

“This project would have not been successful without the support of many individuals”.

First and foremost I wish to thank my internal supervisor Dr.R.G.N.Meegama, whose wisdom, expertise, constant guidance and encouragement have enabled me accomplish success from this seemingly impossible task. My deepest gratitude is extended to my external supervisor Mr. R. Maddegoda whose wisdom has helped the project immensely.

I express my sincere gratitude to all my team mates in the training and development team, fellow Virtusans and Virtusa Corporation for providing me the opportunity to carry out my industrial training at Virtusa Corporation.

And I express my sincere gratitude to Dr .C.P.Udawatte, Head of the Department, Department of Physical Sciences, Faculty of Applied Sciences, Sabaragamuwa University of Sri Lanka, for guiding me toward a successful completion.

My deepest gratitude goes to my mother, my father for empowering me with sufficient discipline and experience for which have helped me in make successes such as this a possibility.

I would also like to express my heart-felt gratitude towards the lectures of Sabaragamuwa University and all my friends and peers for supporting me throughout this project in so many different ways.

ABSTRACT

In order to increase the usage of computers among primary grade students, an interactive environment in which users can easily maneuver within the screen, is proposed. This project involves creating a game environment for such students.

Agile software development methodology which refers to a group of software development methodologies that are based on similar principles is used for this project. According to the main requirements, the proposed system is developed using simple yet attractive games for primary grade students of schools in Sri Lanka. An initial requirements analysis was conducted in order to identify user and system requirements. The requirement of this project was achieved by interviewing the subject matter experts, reviewing similar applications and reading sample documents. Unified Modeling Language (UML) was used to convert the requirements into an analysis model. For potential users, such as students, and teachers and their corresponding tasks, the terms of UML diagram were identified. The analysis model was then translated into a design model. To verify this system, class diagrams, sequence diagrams, logical system architecture diagram and entity relational diagrams were designed. The system was implemented using the Java language where Eclipse Europa is used as the Integrated Development Environment (IDE) to implement Java, which is a separate java editing tool. The user interfaces and images were implemented by using Blend and GNU Image Manipulation Program (GIMP) and Flash CS3. Evaluative feedbacks were requested in each project meeting with team members. The components were tested individually and finally the integrated system was tested. At the end of the development process, the main objective of the project was achieved and the Ministry Of Education was satisfied with the functionalities, usability, security and reliability of the system.

CONTENTS

ACKNOWLEDGEMENT.....	v
ABSTRACT.....	vi
CONTENTS.....	vii
LIST OF ABBREVIATIONS.....	xi
LIST OF FIGURES.....	xii
LIST OF TABLES.....	xiii
CHAPTER 1 INTRODUCTION.....	1
1.1 INTRODUCTION OF VIRTUSA.....	1
1.2 VIRTUSA GAME DEVELOPMENT SPECIAL INTEREST GROUP.....	1
1.3 PROJECT OVERVIEW.....	2
1.4 MAJOR CHALLENGES.....	2
1.5 OBJECTIVES.....	2
1.5.1 Overall Objective.....	3
CHAPTER 2 REVIEW OF LITERATURE.....	4
2.1 JAVA PROGRAMMING LANGUAGE.....	4
2.2 ECLIPSE EUROPA IDE.....	6
2.3 GIMP (GNU IMAGE MANIPULATION PROGRAM).....	6
2.4 BLENDER.....	7
2.5 MYSQL.....	7
2.6 JASPER REPORTS.....	8
2.7 IREPORT.....	8
2.7.1 Features of iReport.....	8
2.8 BRIEF HISTORY OF GAMES.....	9
2.9 GAME ENGINES.....	10
2.10 FREE AND OPEN SOURCE GAME ENGINES.....	11
2.11 UNIFIED MODELING LANGUAGE.....	13
2.12 USE CASE DIAGRAMS.....	13

2.12.1	Elements of Use Case Diagram	13
2.13	CLASS DIAGRAMS	14
2.14	SEQUENCE DIAGRAMS	14
2.15	OBJECT ORIENTED DESIGNING	14
2.16	OBJECT ORIENTED TERMS AND CONCEPTS	15
2.16.1	Class	15
2.16.2	Objects	15
2.16.3	Property & Methods	15
2.16.4	Association	16
2.16.5	Aggregation	16
2.16.6	Inheritance	17
2.16.7	Encapsulation.....	17
2.16.8	Polymorphism.....	18
2.17	QA TESTING.....	18
2.17.1	Testing Levels.....	19
2.17.2	Test Cases.....	20
2.17.3	Test Data.....	21
CHAPTER 3 TECHNOLOGICAL DEVELOPMENT.....		22
3.1	INTRODUCTION TO SOFTWARE DEVELOPMENT METHODOLOGY	22
3.1.1	Software Development Process	22
3.1.2	Agile Development.....	22
3.2	GAME DEVELOPMENT LIFE CYCLE	24
3.3	SCIRRA CONSTRUCT RAPID GAME AUTHORIZING SYSTEM.....	25
3.3.1	Features of Scirra Construct	25
3.4	JMONKEY ENGINE	26
3.4.1	LWJGL.....	27
3.4.2	JOGL	27
3.4.3	Features of JMonkey Engine	27
3.5	REALITY FACTORY	28

3.5.1	Features of Reality Factory.....	28
CHAPTER 4	UML DESIGNING	30
4.1	USE CASE DIAGRAM	30
4.1.1	Use Case Descriptions	31
4.2	CLASS DIAGRAM.....	36
4.3	SEQUENCE DIAGRAMS	37
4.3.1	Teacher can add games to the Launch Pad	37
4.3.2	Teacher can Remove Games from the LaunchPad	38
4.3.3	Teacher can filter the games	39
4.3.4	Teacher can play games.....	40
4.3.5	Teacher can view student reports	41
4.3.6	Student can play games	42
4.3.7	Student can view reports.....	43
CHAPTER 5	GAME CONCEPTS & ANALYSIS	44
5.1	ASCENDING TRAIN (OR DESCENDING TRAIN).....	44
5.2	ODD/EVEN NUMBER SEPARATOR	45
5.3	DISTANCE AND DIRECTIONS (TREASURE HUNT).....	46
5.4	VIRTUAL SHOP	47
CHAPTER 6	SYSTEM DEVELOPMENT.....	49
6.1	DEVELOPMENT ENVIRONMENT	49
6.1.2	Software Environment.....	49
6.2	API USED FOR IMPLEMENTATION.....	49
6.3	INTEGRATING DEVELOPMENT ENVIRONMENT	50
6.4	REPORT GENERATION	51
CHAPTER 7	SYSTEM TESTING & DEPLOYMENT	52
7.1	TEST STRATEGY.....	52
7.2	UNIT TESTING.....	52
7.2.1	Test Cases.....	53
7.3	SYSTEM TESTING.....	56

7.4	DEPLOYMENT ENVIORNMENT	56
7.4.1	Hardware Requirements	56
7.4.2	Software Requirements.....	56
CHAPTER 8 CONCLUSION		57
8.1	CONCLUSION	57
8.2	FUTURE CONSIDERATION	57
REFERNCES		58
INDEX.....		60

LIST OF ABBREVIATIONS

GDSIG	Game Development Special Interest Group
CSR	Corporate Social Responsibility
UML	Unified Modeling Language
JVM	Java Virtual Machine
IDE	Integrated Development Environment
SQL	Structured Query Language
JDBC	Java Database Connectivity
GUI	Graphical User Interface
FPS	First Person Shooters
RPG	Role-Playing Game
OOP	Object Oriented programming
FRS	Functional Requirement Specification
SRS	System Requirement Specification
QA	Quality Assurance
SIT	System Integration Testing
GPL	General Public License
LWJGL	Light Weight Java Game Library
JOGL	Java OpenGL
OpenGL	Open Graphics Library
OpenAL	Open Audio Library
jME	jMonkey Engine
API	Application Programming Interface

LIST OF FIGURES

Figure 2.9 Aggregation.....	16
Figure 2.7 An example of a Dog Class.....	16
Figure 2.8 Associations	16
Figure 2.10 Inheritance.....	17
Figure 2.11 Example of Encapsulation Concept	18
Figure 3.1 Agile Development Model.....	23
Figure 3.2 Game Development Life Cycle.....	24
Figure 3.4 A Demo Game Created with Scirra Construct	26
Figure 3.6 A Game Created with jMonkey Engine	27
Figure 3.8 A Game Created with Reality Factory	29
Figure 4.1 Gives the use case diagram for the game.	30
Figure 4.2 Class Diagram	36
Figure 4.3 Teacher Add Games.....	37
Figure 4.4 Teacher remove games.....	38
Figure 4.5 Teacher Filter Games	39
Figure 4.6 Teacher can play games	40
Figure 4.7 Teacher view reports	41
Figure 4.8 Student play games	42
Figure 4.9 Student view reports.....	43
Figure 5.2 Odd/Even Number Separator	45
Figure 5.1 Ascending Train	45
Figure 5.3 Treasure Hunt.....	46
Figure 5.4 Virtual Shop	48
Figure 6.1 Development Environment	50
Figure 6.2 iReport Development Environment	51
Figure 7.1 QA Testing.....	52
Figure 7.2 Pre Requisite	53
Figure 7.3 Expected Result.....	54
Figure 7.4 Logging Screen	54
Figure 7.5 Student Last Played Game	55

LIST OF TABLES

Table 2.1 Game Engine Overview	12
Table 4.1 Add Game Use Case Description	31
Table 4.2 Remove Game Use Case Description.....	32
Table 4.3 Filter Game Use Case Description	33
Table 4.4 View Reports Use Case Description	34
Table 4.5 Launch Game Use Case Description.....	35
Table 4.6 Teacher Add Games	37
Table 4.7 Teacher remove games	38
Table 4.8 Teacher Filter Games	39
Table 4.9 Teacher can play games.....	40
Table 4.10 Teacher view student reports.....	41
Table 4.11 Student play games.....	42
Table 4.12 Student view reports	43
Table 5.1 Ascending Train	44
Table 5.2 Odd/Even Number Separator	45
Table 5.3 Treasure Hunt	46
Table 5.4 Virtual Shop	47
Table 6.1 Software Development Environment	49
Table 7.1 Test Case 1	53
Table 7.2 Test Case 2	55
Table 7.3 Deployment of the software	56

CHAPTER 1 INTRODUCTION

1.1 INTRODUCTION OF VIRTUSA

Virtusa Corporation is a leading global technology innovation services provider that creates competitive advantage for its clients. Virtusa was founded in 1996 by the prominent technology entrepreneur, Kris Canekaratne, who has assembled a strong leadership team from well-known companies like Infosys, IBM, Aether, 3Com and John Keels. Previously known as eRUNWAY, Inc., Virtusa has grown beyond being an efficient provider of product and application development services to being the partner of choice in creating competitive advantage for its clients using technology solutions.

Headquarters in Westborough, MA, Virtusa employs the finest global technology talent, spread across its Advanced Technology Centers in the US, India and Sri Lanka. It also has sales and marketing offices in several locations around the world.

1.2 VIRTUSA GAME DEVELOPMENT SPECIAL INTEREST GROUP

Virtusa Game Development Special Interest Group (Game Dev SIG) is a knowledge sharing group that is open for anyone who is interested in game development within Virtusa.

Virtusa Game Development Special Interest Group (GDSIG) is currently working on an internal project to develop few simple computer games for kids. This project is focusing on implementing simple computer games in any technology, creating attractive 2D or 3D arts for kids, integrating sound effects. This is a voluntary project and anyone can work for this project in their spare time.

1.3 PROJECT OVERVIEW

As a part of the Corporate Social Responsibility (CSR) initiatives of Virtusa, GDSIG involves in a voluntary project to provide edutainment software for primary school children in Sri Lanka. The purpose of this proposed system is to develop simple games in an interactive and an attractive manner aiming primary grade students attending schools of Sri Lanka.

At the moment, the system consists of several educational games in different format without any integration into a unique system. Moreover, the existing system is developed solely for entertainment without concentrating on learning tools. It contains action and racing games that may improve only hand coordination movement of students. In order to further develop the scenario, we propose to change the core of the system with the manipulation of the whole environment.

1.4 MAJOR CHALLENGES

The major challenges of this project are:

- To gather requirements - by interviewing, organizing formal discussions and reviewing sample documents.
- To develop a concrete understanding of the game concepts.
- To acquire technical skills this related to graphics programming.

1.5 OBJECTIVES

Implement console Software for the Edutainment software suite. Our major objective is to develop the game portal which is the first interface user can see. Teachers can select the games for their students and students can play games through it. Also students can see the results and can their game skills.

1.5.1 Overall Objective

- **Developing Educational Computer Games for Children of Sri Lankan Schools**
- **Develop the Mathematical skills of students**
- **Provide instructions in both Sinhala and Tamil languages**
- **Report Demonstration**
- **Increase market opportunities**
- **Take Latest technology for students**
- **Measure the progress of students**

CHAPTER 2 REVIEW OF LITERATURE

To achieve the objectives of this project, in-depth knowledge of the following topics were used.

- UML Diagrams
- Object Oriented Programming principles
- Java 5.0
- Eclipse Eurapa IDE 2.0
- Scirra Construct
- JMonkey Engine
- Reality Factory
- Blender
- Gimp
- MySql
- iReport
- Jasper Reports

2.1 JAVA PROGRAMMING LANGUAGE

In 1991, a small group of Sun engineers called the "Green Team" believed that the next wave in computing was the union of digital consumer devices and computers. The Green Team demonstrated their new language with an interactive, handheld home-entertainment controller that was originally targeted at the digital cable television industry. Unfortunately, the concept was much too advanced for them at the time. But it was just right for the Internet, which was just starting to take off. In 1995, the team announced that the Netscape Navigator Internet browser would incorporate Java technology [www4].

The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code that can run on any Java virtual machine (JVM) regardless of computer architecture.

Today, Java not only permeates the Internet, but also is the invisible force behind many of the applications and devices that power our day-to-day lives [www5]. From mobile phones to handheld devices, games and navigation systems to e-business solutions, Java is everywhere!

In our project we have used the latest version of Java 5.0 as the programming Language. Basically we have used java 2D graphics package for our graphical requirements. Also Java has more advantages over the other programming languages like,

- Java is easy to learn.
- Java was designed to be easy to use and is therefore easy to write, compile, debug, and learn than other programming languages.
- Java is object-oriented. This allows you to create modular programs and reusable code.
- Java is platform-independent.

One of the most significant advantages of Java is its ability to move easily from one computer system to another. The ability to run the same program on many different systems is crucial to World Wide Web software, and Java succeeds at this by being platform-independent at both the source and binary levels.

In other word java has several disadvantages which we have to concern like,

- Performance: Java can be perceived as significantly slower and more memory-consuming than natively compiled languages such as C or C++ [www6].
- Look and feel: The default look and feel of GUI applications written in Java using the Swing toolkit is very different from native applications. It is possible to specify a different look and feel through the pluggable look and feel system of Swing.
- Single-paradigm language: Java is predominantly a single-paradigm language. However, with the addition of static imports in Java 5.0 the procedural paradigm is better accommodated than in earlier versions of Java.

2.2 ECLIPSE EUROPA IDE

In this project we have used Eclipse Europa as the Integrated Development Environment [www15]. Eclipse is a multi-language software development platform comprising an IDE and a plug-in system to extend it. It is written primarily in Java and is used to develop applications in this language and, by means of the various plug-ins, in other languages as well— C/C++, Perl, PHP and more. We have used Eclipse to code java classes, compile them and run those classes. Eclipse Europa IDE is one of the updated and a modified version of Eclipse IDE.

2.3 GIMP (GNU IMAGE MANIPULATION PROGRAM)

The GIMP (GNU Image Manipulation Program), is a raster graphics editor used to process digital graphics and photographs [www8]. GIMP is a freely distributed piece of software for such tasks as photo retouching, image composition and image authoring. It works on many operating systems, in many languages. In this project Gimp is used to design the user interfaces, graphics etc.

2.4 BLENDER

Blender is a free 3D graphics application. It can be used for modeling, UV unwrapping, texturing, rigging, water simulations, skinning, animating, rendering, particle and other simulations, non-linear editing, compositing, and creating interactive 3D applications [www12]. Blender is available for several operating systems, including Microsoft Windows, Mac OS X, Linux, IRIX, Solaris, NetBSD, FreeBSD, and OpenBSD with unofficial ports for BeOS, SkyOS, AmigaOS, MorphOS and Pocket PC. Blender has a robust feature set similar in scope and depth to other high-end 3D software such as Softimage, Cinema 4D, 3ds Max, Lightwave and Maya. These features include advanced simulation tools such as rigid body, fluid, cloth and soft body dynamics, modifier based modeling tools, powerful character animation tools, a node based material and compositing system and Python for embedded scripting.

2.5 MYSQL

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by MySQL AB. MySQL AB is a commercial company, founded by the MySQL developers [www11].

- MySQL is a database management system.

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server.

- MySQL is a relational database management system.

A relational database stores data in separate tables rather than putting all the data in one big storeroom. This adds speed and flexibility. The SQL part of "MySQL" stands for "Structured Query Language." SQL is the most common standardized language used to access databases and is defined by the ANSI/ISO SQL Standard.

- MySQL software is Open Source.

Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything.

- The MySQL Database Server is very fast, reliable, and easy to use.

2.6 JASPER REPORTS

Jasper Reports is the best open source reporting engine available for Java community [www16]. It is developed by a small big genius called Teodor Danciu. Jasper Reports has always had one lack: it doesn't provide an adapted tool to visually design reports. Jasper Reports provides the necessary features to generate dynamic reports, including data retrieval using JDBC (Java Database Connectivity), as well as support for parameters, expressions, variables, and groups. Jasper Reports also includes advanced features, such as custom data sources, scriptlets, and sub reports.

2.7 IREPORT

iReport is a program that helps users and developers that use the Jasper Reports library to visually design reports [www14]. Through a rich and very simple to use GUI, iReport provides all the most important functions to create nice reports in little time.

2.7.1 Features of iReport

- 98% of Jasper Reports tags support
- Visual designer with tools for draw rectangles, lines, ellipses, text fields, charts, sub reports...
- Built-in editor with syntax highlighting for write expression
- Support of all JDBC compliant databases
- Support for sub reports
- Facilities for fonts

2.8 BRIEF HISTORY OF GAMES

Computer games were introduced as a commercial entertainment medium in 1971, becoming the basis for an important entertainment industry in the late 1970s/early 1980s in the United States, Japan, and Europe [www18]. The first generation of PC games was often text adventures or interactive fiction, in which the player communicated with the computer by entering commands through a keyboard. The first text-adventure, *Adventure*, was developed for the PDP-11 by Will Crowther in 1976, and expanded by Don Woods in 1977. By the 1980s, personal computers had become powerful enough to run games like *Adventure*, but by this time, graphics were beginning to become an important factor in games.

Prior to game engines, games were typically written as singular entities. Thus, most game designs through the 1980s were designed through a hard-coded rule set with a small amount of level and graphics data. The term "game engine" arose in the mid-1990s, especially in connection with 3D games such as first-person shooters (FPS). Modern game engines are some of the most complex applications written, frequently featuring dozens of finely tuned systems interacting to ensure a finely controlled user experience. The continued refinement of game engines has created a strong separation between rendering, scripting, artwork, and level design. First-person shooter games remain the predominant users of third-party game engines, but they are now also being used in other genres. As game engine technology matures and becomes more user-friendly, the applications of game engines has broadened in scope, and are now being used for serious games: visualization, training, medical, and military simulation applications.

2.9 GAME ENGINES

The game engine is generally the library of core functions used in the game, usually related to graphics, input, networking and other systems. Another way to understand what a game engine is would be considering them as the non game-specific part of the game, so we can have several games ranging from RPGs to FPSs using the same engine. There are many game engines that are designed to work on game consoles and desktop operating systems such as Linux, Mac OS X, and Microsoft Windows. The core functionality typically provided by a game engine includes a rendering engine ("renderer") for 2D or 3D graphics, a physics engine or collision detection (and collision response), sound, scripting, animation, artificial intelligence, networking, streaming, memory management, threading, and a scene graph.

Game engines provide a suite of visual development tools in addition to reusable software components. These tools are generally provided in an integrated development environment to enable simplified, rapid development of games in a data-driven manner. These game engines are sometimes called "game middleware" because, as with the business sense of the term, they provide a flexible and reusable software platform which provides all the core functionality needed, right out of the box, to develop a game application while reducing costs, complexities, and time-to-market—all critical factors in the highly competitive game industry.

Some game engines only provide real-time 3D rendering capabilities instead of the wide range of functionality required by games. These engines rely upon the game developer to implement the rest of this functionality or assemble it from other game middleware components. These types of engines are generally referred to as a "graphics engine," "rendering engine," or "3D engine" instead of the more encompassing term "game engine." However, this terminology is inconsistently used as many full-featured 3D game engines are referred to simply as "3D engines." A few examples of graphics engines are: Realm Forge, Truevision3D, OGRE, Crystal Space, Genesis3D, Irrlicht and JMonkey Engine.

2.10 FREE AND OPEN SOURCE GAME ENGINES

These engines are available for free use, but without the source code being available under an open source license. Many of these engines are commercial products which have a free edition available for them.

- **Adventure Game Studio** – Mainly used to develop third-person pre-rendered adventure games, this engine is one of the most popular for developing amateur adventure games.
- **Build engine** – A first-person shooter engine used to power Duke Nukem 3D
- **dim3** – Freeware 3D JavaScript engine for the Mac (although finished games are cross platform).
- **DX Studio** – Real-time professional 3D engine and editing suite produced by World weaver Ltd
- **Game Maker Lite** – Object-oriented game development software with a scripting language as well as a drag-and-drop interface
- **JMonkeyEngine** – An open-source, BSD licensed Java scene graph engine.

A comprehensive list of game engines is shown in Table 2.1

Table 2.1 Game Engine Overview

Name	Language	Platform	License	Graphics	Sound	Scripting
AgateLib	.NET	Windows / Mono	Free	2D via Direct3D or OpenGL	Yes	No
AGL Engine	C++	Windows	Commercial	2D via DirectDraw, Direct3D or OpenGL	Yes	No
C4 Engine	C++	Windows , Mac, PS3	Commercial	3D	Yes	Visual Scripting
DXGame Engine	VB6	Windows	Free	2D+ via Direct3D	Yes	No
Game Maker	Delphi	Windows	Free and Commercial	2D/3D	Yes	Its own scripting language(GML)
Ghost Engine	C++	Windows	Engine code is Zlib/libPNG -licensed	3D via OpenGL/DirectX,	No	-
Jet3D	C/C++	Windows	Free	3D via DirectX		
JGame	Java	Windows , Unix, MacOSX	Free (BSD)	2D	Yes	No
jMonkey Engine	Java	Windows , Linux, MacOS X	Free (BSD)	3D via LWJGL	Yes - OpenAL Sound	Yes - jMonkey Scripting Framework
The RealFeel Engine	VB6	Windows XP/Vista	Free (Closed Source)	2D	Yes	No
Reality Factory	None needed	Windows	Genesis 3D license	3D via Genesis3D (DirectX)	Yes	Yes
Visual3D .NET	.NET 2.0 (C#)	Windows , Xbox 360	Commercial, Free Student Commercial & Non-commercial	3D via DirectX or XNA	Yes	C#, VB.NET, C++.NET, J# (Java), JScript.NET (JavaScript), IronPython, Visual Programming/Modeling

2.11 UNIFIED MODELING LANGUAGE

The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system [www9]. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components. UML is a 'language' for specifying and not a method or procedure. The UML [B1] is used to define a software system; to detail the artifacts in the system, to document and construct - it is the language that the blueprint is written in. The UML may be used in a variety of ways to support a software development methodology (such as the Rational Unified Process) - but in itself it does not specify that methodology or process. UML defines several types of diagrams: class, use case, sequence, collaboration, activity, diagrams etc.

2.12 USE CASE DIAGRAMS

The Use case diagram is used to identify the primary elements and processes that form the system. The primary elements are termed as "actors" and the processes are called "use cases." The Use case diagram shows which actors interact with each use case.

2.12.1 Elements of Use Case Diagram

- **Actor** - An actor portrays any entity that performs certain roles in a given system. The different roles the actor represents are the actual business roles of users in a given system. An actor in a use case diagram interacts with a use case. For example, for modeling a banking application, a customer entity represents an actor in the application. Similarly, the person who provides service at the counter is also an actor.
- **Use Case** - A use case in a use case diagram is a visual representation of distinct business functionality in a system. The key term here is "distinct business functionality." As the first step in identifying use cases, you should list the discrete business functions in your problem statement. Each of these business functions can be classified as a potential use case.

2.13 CLASS DIAGRAMS

Class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, packages and objects. Classes are composed of three things: a name, attributes, and operations. Class diagrams also display relationships such as containment, inheritance, associations and others. The association relationship is the most common relationship in a class diagram. The association shows the relationship between instances of classes. Another common relationship in class diagrams is a generalization. A generalization is used when two classes are similar, but have some differences. Class diagrams are used in nearly all Object Oriented software designs. Use them to describe the Classes of the system and their relationships to each other.

2.14 SEQUENCE DIAGRAMS

A Sequence diagram depicts the sequence of actions that occur in a system. The invocation of methods in each object, and the order in which the invocation occurs is captured in a Sequence diagram. This makes the Sequence diagram a very useful tool to easily represent the dynamic behavior of a system. A sequence diagram is made up of objects and messages.

2.15 OBJECT ORIENTED DESIGNING

Object-oriented programming (OOP) is a programming paradigm that uses "objects" and their interactions to design applications and computer programs. Object-orientation is so called because this method sees things that are part of the real world as objects. A phone is an object in the same way as a bicycle, a human being, or insurance policies are objects. In everyday life, we simplify objects in our thinking – we work with models. Programming techniques may include features such as encapsulation, modularity, polymorphism, and inheritance. It was not commonly used in mainstream software application development until the early 1990s. Many modern programming languages now support OOP. For the software engineer, object-oriented technology encompasses object-oriented programming languages, object-oriented development methodologies, management of object-oriented projects, object-oriented computer hardware, and object-oriented computer aided software engineering, among others.

Many of the terms commonly used in object-oriented technology were originally used to describe object-oriented programming (coding) concepts. Specifically, although the terms were borrowed from a non-computer-software perspective, they were first used extensively to describe concepts embodied in object-oriented programming languages, such as Smalltalk, C++, Java, and Eiffel.

2.16 OBJECT ORIENTED TERMS AND CONCEPTS

2.16.1 Class

A class is used to describe something in the world, such as occurrences, things, external entities, roles, organization units, places or structures. A class describes the structure and behavior of a set of similar objects. It is often described as a template, generalized description, pattern or blueprint for an object, as opposed to the actual object, itself. Once a class of items is defined, a specific instance of the class can be defined. An instance is also called “object”.

2.16.2 Objects

Objects are the physical and conceptual things we find in the universe around us. Hardware, software, documents, human beings, and even concepts are all examples of objects. The class of Dog defines all possible dogs by listing the characteristics and behaviors they can have; the object Lassie is one particular dog, with particular versions of the characteristics. A Dog has fur; Lassie has brown-and-white fur. Objects are thought of as having state. The state of an object is the condition of the object, or a set of circumstances describing the object. We also think of the state of an object as something that is internal to an object. For example, if we place a message in a mailbox, the (internal) state of the mailbox object is changed, whereas the (internal) state of the message object remains unchanged.

2.16.3 Property & Methods

Properties in a class are used to present the structure of the objects: their components and the information or data contained therein. An instance of a class has the properties defined in its class and all of the classes from which its class inherits.

Methods in a class describe the behavior of the objects. It represents a function that an instance of the class can be asked to perform.

Figure 2.7 depicts an example of a Dog class.

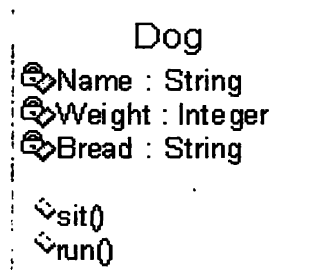


Figure 2.1 An example of a Dog Class

2.16.4 Association

An association is a relationship between different objects of one more classes. A simple example of an association is the relationship among an enterprise, departments and employees is shown in figure 2.8

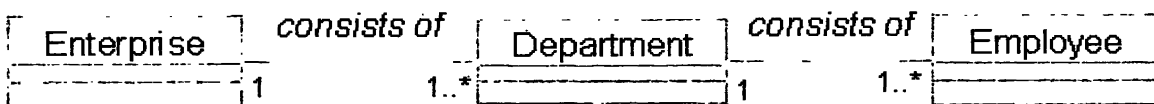


Figure 2.2 Associations

2.16.5 Aggregation

Aggregation is a special form of association. Aggregation is the composition of an object out of a set of parts. A car, for example, is an aggregation of tires, engine, steering wheel, brakes and so on. Aggregation represents a “has” relationship: a car has an engine. Instead of aggregation, some people talk about “whole-part” hierarchy. For example, figure 2.9 shows, where an Enterprise represents a “whole” end and Department represents a “part” end.

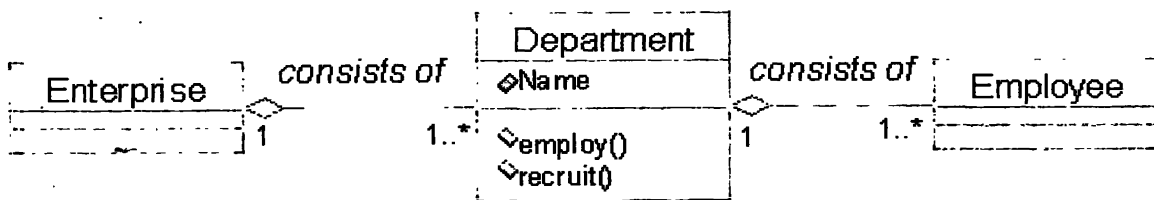


Figure 2.3 Aggregation

2.16.6 Inheritance

Inheritance is the property whereby one class extends another class by including additional methods and/or variables. The original class is called the super class of the extending class, and the extending class is called the subclass of the class that is extended. Since a subclass contains all of the data and methods of the super class plus additional resources, it is more specific. Figure 2.10 shows an example, where Circle and Rectangle inherit from GeomFigure and own all attributes and methods from GeoFigure.

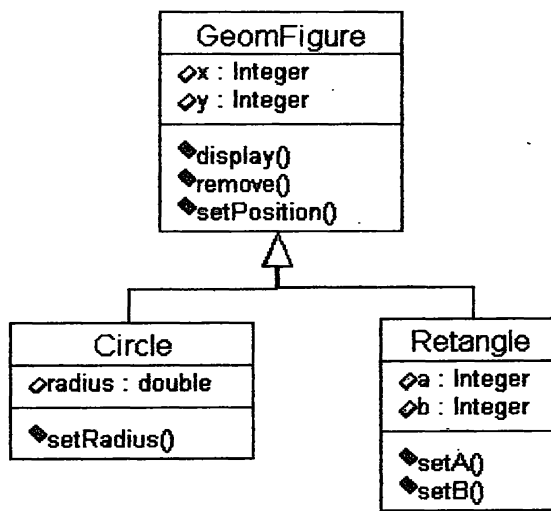


Figure 2.4 Inheritance

2.16.7 Encapsulation

Encapsulation means as much as shielding. Each object-oriented object has a shield around it. Objects can't 'see' each other. They can exchange things though, as if they are interconnected through a hatch. Figure 2.11 shows the concept of the encapsulation. It separates the external aspects of an object from the internal implementation details of the object, which are hidden from other objects. The object encapsulates both data and the logical procedures required to manipulate the data.

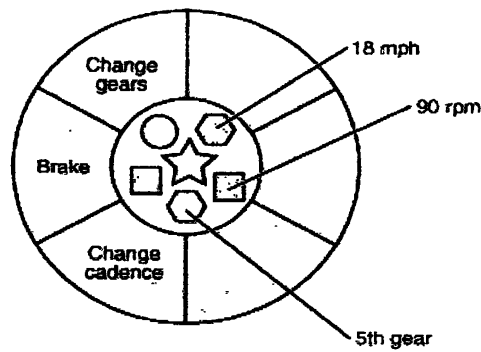


Figure 2.5 Example of Encapsulation Concept

2.16.8 Polymorphism

Polymorphism indicates the meaning of “many form.” In object-oriented design, polymorphism present a method can has many definitions. Polymorphism is related to Overloading and Overriding. Overloading indicates a method can have different definitions by defining different type of parameter. Overriding indicates that subclass and parent class have the same methods, parameters and return types.

2.17 QA TESTING

Software Testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate [www26]. This includes, but is not limited to, the process of executing a program or application with the intent of finding software bugs.

A primary purpose for testing is to detect software failures so that defects may be uncovered and corrected. This is a non-trivial pursuit. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions. The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions as well as examining the aspects of code: does it do what it is supposed to do and do what it needs to do. In the current culture of software development, a testing organization may be separate from the development team. There are various roles for testing team members. Information derived from software testing may be used to correct the process by which software is developed.

A common source of requirements gaps is non-functional requirements such as testability, scalability, maintainability, usability, performance, and security. Software faults occur through the following process. A programmer makes an error, which results in a defect in the software source

code. If this defect is executed, in certain situations the system will produce wrong results, causing a failure. Not all defects will necessarily result in failures.

2.17.1 Testing Levels

- **Unit Testing**

The primary goal of unit testing is to take the smallest piece of testable software in the application, isolate it from the remainder of the code, and determine whether it behaves exactly as you expect. Each unit is tested separately before integrating them into modules to test the interfaces between modules. Unit testing has proven its value in that a large percentage of defects are identified during its use.

The most common approach to unit testing requires drivers and stubs to be written. The driver simulates a calling unit and the stub simulates a called unit. The investment of developer time in this activity sometimes results in demoting unit testing to a lower level of priority and that is almost always a mistake. Even though the drivers and stubs cost time and money, unit testing provides some undeniable advantages. It allows for automation of the testing process, reduces difficulties of discovering errors contained in more complex pieces of the application, and test coverage is often enhanced because attention is given to each unit.

- **Integration Testing**

'Integration testing' called abbreviated I&T is the phase of software testing in which individual software modules are combined and tested as a group. It follows unit testing and precedes system testing.

Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

The purpose of integration testing is to verify functional, performance and reliability requirements placed on major design items. These design items are exercised through their interfaces using Black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test that all components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e. unit testing.

The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages.

Some different types of integration testing are big bang, top-down, and bottom-up.

- **System Testing**

System testing of software is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic [www27]. System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing is an investigatory testing phase, where the focus is to have almost a destructive attitude and tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification(s). System testing includes the Load testing and Stress testing. Once the Load testing and Stress testing is completed successfully, the next level of Alpha Testing or Beta Testing will go ahead.

- **System Integration Testing**

System Integration Testing (SIT), in the context of software systems and software engineering, is a testing process that exercises a software system's coexistence with others. System integration testing takes multiple integrated systems that have passed system testing as input and tests their required interactions. Following this process, the deliverable systems are passed on to acceptance testing.

Systems integration testing (SIT) is a testing phase that may occur after unit testing and prior to user acceptance testing (UAT). Many organizations do not have a SIT phase and the first test of UAT may include the first integrated test of all software components.

2.17.2 Test Cases

In software engineering, the most common definition of a test case is a set of conditions or variables under which a tester will determine if a requirement or use case upon an application is partially or fully satisfied. It may take many test cases to determine that a requirement is fully satisfied. In order to fully test that all the requirements of an application are met, there must be at least one test case for each requirement.

If the application is created without formal requirements, then test cases can be written based on the accepted normal operation of programs of a similar class. Test cases are not written at all but the activities and results are reported after the tests have been run. What characterizes a formal, written test case is that there is a known input and an expected

output, which is worked out before the test is executed. The known input should test a precondition and the expected output should test a post condition.

A test case includes:

- The purpose of the test.
- Special hardware requirements, such as a modem.
- Special software requirements, such as a tool.
- Specific setup or configuration requirements.
- A description of how to perform the test.
- The expected results or success criteria for the test.

7.17.3 Test Data

A set of data created for testing new or revised applications. Test data should be developed by the user as well as the programmer and must contain a sample of every category of valid data as well as many invalid conditions as possible.

CHAPTER 3 TECHNOLOGICAL DEVELOPMENT

3.1 INTRODUCTION TO SOFTWARE DEVELOPMENT METHODOLOGY

3.1.1 Software Development Process

Software Engineering is concerned with concepts, processes and tools that support the timely and cost effective development of quality software. A software development process is a structure imposed on the development of a software product. Synonyms include software lifecycle and software process. There are several models for such processes, like Waterfall process, Iterative process, Prototyping, Spiral etc.

3.1.2 Agile Development

Agile software development is a group of software development methodologies that are based on similar principles [www1][www2]. In the late 1990's several methodologies began to get increasing public attention. Each had a different combination of old ideas, new ideas, and transmuted old ideas. The peak time of agile software development evolved in the mid 1990s as part of a reaction against "heavyweight" methods, as a typical example by a heavily regulated, regimented, micro-managed use of the waterfall model for software development, the processes that software engineers actually perform effective work.

There are many specific agile development methods. Most promote development iterations, teamwork, collaboration, and process adaptability throughout the life-cycle of the project [www3]. Agile chooses to do things in small increments with minimal planning, rather than long-term planning. Iterations are short time frames which typically last from one to four weeks. Each iteration is worked on by a team through a full software development cycle, including planning, requirements analysis, design, coding, unit testing, and acceptance testing when a working product is demonstrated to customers. These are known as iterations with a full project consisting of several hundred iterations. However, the chief aim is to produce a functional, usable piece of software in each iteration. So, iteration is actually as a mini-project that includes all the same planning, design, programming and testing as in a large project. Because of the short nature of these iterations, agile is seen as a lower risk software development model. If the iteration doesn't work, it can be modified without causing significant delays or cost overruns. It also cuts out much of the bureaucracy and restrictions of heavier-weight models. Ideally, at the end of each reiteration the software should be ready, or at least almost ready, for release. At this point, the team re-evaluates the entire project and decides on the next step.

Because of its focus on fast turnaround times, the agile model encourages person-to-person communication. Very often, software development teams using this model will work together in an open-plan office with all meetings being held face to face.

The following are other features that describe software development projects that use agile methodologies:

- The fast turnaround time and the regular delivery of working software should ensure customer satisfaction
- Late changes can be handled easily, or even welcomed
- Progress is measured by the delivery of working software
- Clients and developers communicate regularly face-to-face
- All meetings within the development team are held face-to-face
- All developers are highly competent and trustworthy

Figure 3.1 illustrates the agile life cycle

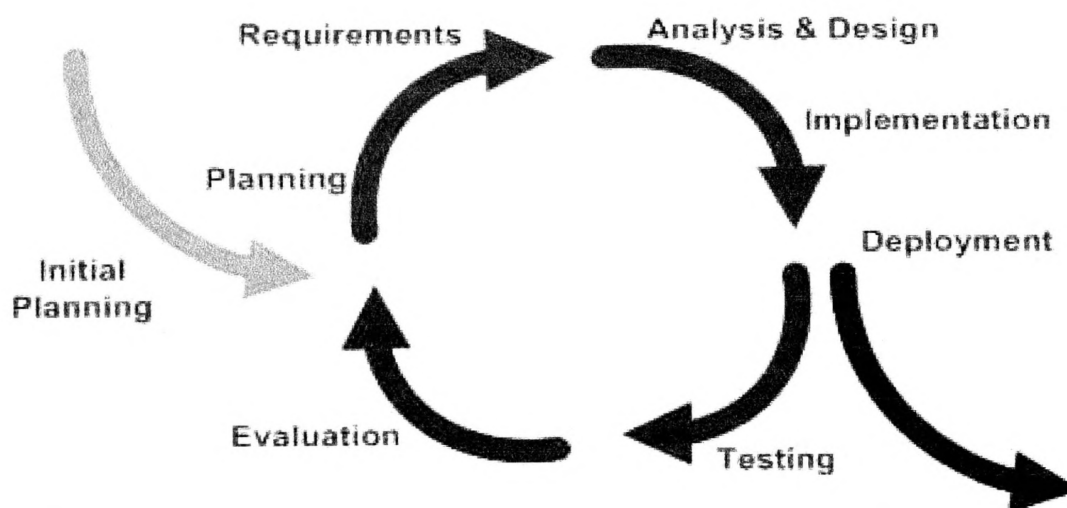


Figure 3.1 Agile Development Model

3.2 GAME DEVELOPMENT LIFE CYCLE

The diagram in Figure 3.2 shows the development life cycle of games. According to the cycle, the first phase defines the Game Concept, where depends upon collected requirements as highlighted in the original project proposal. Basically, primary schools need mathematical and language concepts so as to facilitate the learning environment of students. Under the pre-production, resources that need to implement the system are collected which includes modeled games as well as the developing aspects. Because of prototyping method, we can clarify all the artifacts in the system. The testing and releasing phases are completed next.

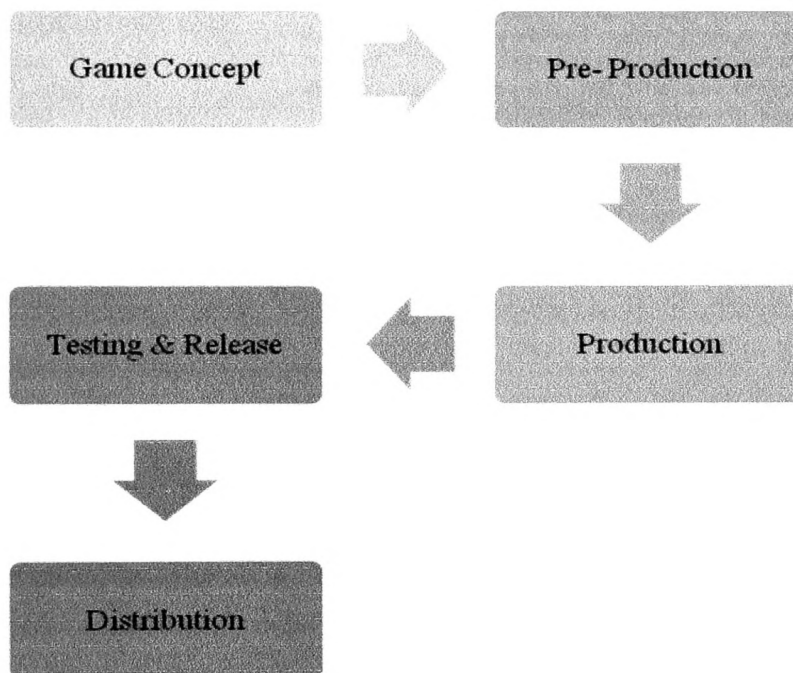


Figure 3.2 Game Development Life Cycle

3.3 SCIRRA CONSTRUCT RAPID GAME AUTHORIZING SYSTEM

Construct is free powerful and easy to use development software for both DirectX 9-based games and applications [www23]. It includes an event based system for defining how the game or application will behave, in a visual, human-readable way - easy enough for complete beginners to get results quickly. Optionally, advanced users can also use Python scripting to code our creations. Construct is not a commercial software project, and is developed by volunteers. It is 100% free to download the full version - no nag screens, adverts or restricted features at all.

3.3.1 Features of Scirra Construct

Create games and applications with:

- Super fast hardware-accelerated DirectX 9 graphics engine
- Add multiple pixel shades for special effects, including lighting, HDR, distortion, lenses and more
- Advanced rendering effects like motion blur, skew and bump mapping (3D lighting)
- Innovative Behaviors system for defining how objects work in a flexible way
- Physics engine for realistic object behavior
- Place object on different layers for organizing display, paralleling, or whole-layer effects - also freely zoom individual layers in and out with high detail
- Python scripting for advanced users - however, Construct's Events system is still powerful enough to complete entire games without any scripting.
- Smaller, faster specialized runtime for applications

Construct is developed open source under the General Public License (GPL). This means we can download and use Construct for free, but it also means that the underlying source code - the code that defines how the program works - is also freely available. This means other programmers are free to fix errors in the code and make their own contributions to construct.

Figure 3.4 depicts a demonstration of Scirra construct.



Figure 3.3 A Demo Game Created with Scirra Construct

3.4 JMONKEY ENGINE

jME (jMonkey Engine) is a high performance scene graph based graphics API. jME was built to fulfill the lack of full-featured graphics engines written in Java [www21]. Using an abstraction layer, it allows any rendering system to be plugged in. Currently, both **LWJGL** and **JOGL** OpenGL bindings are supported. jME is completely open source under the BSD license.

jME was created by Mark Powell in 2003 while he was investigating OpenGL rendering. After discovering LWJGL he decided that Java (his language of choice) would be perfect for his own graphics tools. These tools soon grew into a primitive engine. After reading David Ebery's 3D Game Engine Design, scene graph architecture was implemented. It was then that jME became part of Sun's Java.net software repository.

3.4.1 LWJGL

The Lightweight Java Game Library (LWJGL) is a solution aimed directly at professional and amateur Java programmers alike to enable commercial quality games to be written in Java. LWJGL provides developers access to high performance cross platform libraries such as OpenGL (Open Graphics Library) and OpenAL (Open Audio Library) allowing for state of the art 3D games and 3D sound. Additionally LWJGL provides access to controllers such as Gamepads, Steering wheel and Joysticks. All in a simple and straight forward API.

3.4.2 JOGL

JOGL (Java OpenGL) are a set of bindings to OpenGL that are officially supported by Sun.

3.4.3 Features of JMonkey Engine

- jME is scene graph based architecture. The scene graph allows for organization of the game data in a tree structure, where a parent node can contain any number of children nodes, but a child node contains a single parent. Typically, these nodes are organized spatially to allow the quick discarding of whole branches for processing.
- jME's camera system uses frustum culling to through out scene branches that are not visible. This allows for complex scenes to be rendered quickly, as typically, most of the scene is not visible at any one time.
- jME also supports many high level effects, such as: Imposters (Render to Texture), Environmental Mapping, Lens Flare, Tinting, Particle Systems, etc.
- jME supplies the user with easy to use, but powerful application classes for building the application. Jumping into jME should be a quick and painless process. With a small learning curve.

Figure 3.6 gives a scene developed using JMonkey Engine.

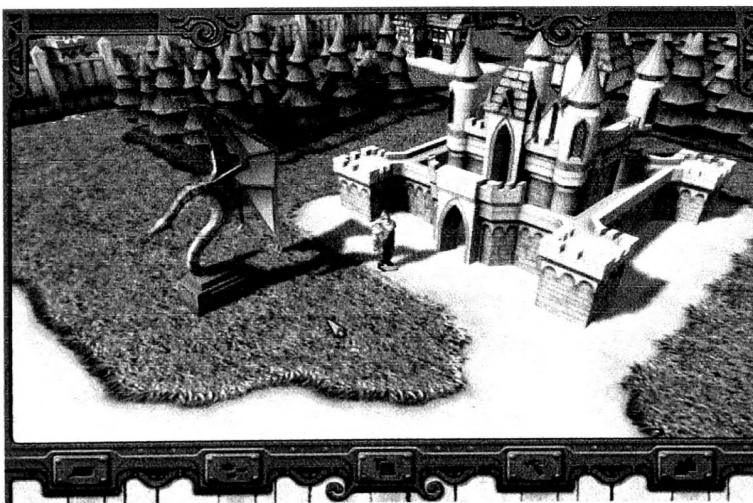


Figure 3.4 A Game Created with jMonkey Engine

3.5 REALITY FACTORY

Reality Factory is a program that - in conjunction with other tools - allows us to create 1st and 3rd person perspective games without programming! Reality Factory is built on top of the powerful Genesis3D Open Source engine and supports all major 3D graphics cards. Reality Factory provides most of the tools we need to make a game [www19]. We will still need a program to create actors (characters and props in our game) and software to make textures with, but what we won't need is a C/C++ compiler and a couple of coders to build our engine for us. By using objects called "entities" which you place in our world, we can set up a game - with audio effects, multiple soundtracks, and special effects. Reality Factory is intended to be a "rapid game prototyping tool" - it is able to make playable, interesting games across a wide range of genres but it's not optimized for any ONE kind of game.

3.5.1 Features of Reality Factory

- Complete game & machine creation system without requiring any programming knowledge.
- Predefined character and camera controls provide 1st and 3rd person viewpoints, changeable on-the-fly in-game as desired
- Complete interactive conversation engine, complete with a GUI conversation tree builder for writing your conversation scripts
- Customizable script editor for creating scripts
- Basic physics, collision detection
- Per vertex, light mapping, radiosity
- Dynamic colored (RGB) lighting
- Projected Shadows
- Basic multi-texturing, bump-, sphere-, mip-mapping, procedural textures
- Video AVI & animated GIF support for cut scenes and animated level textures
- Dynamic texturing effects such as procedurals, animations and morphing
- Key frame animation, skeletal animation, animation blending
- Customizable effects & explosions system
- 3D audio engine with mp3, wav and support

Figure 3.8 has a scene of a game created using Reality Factory.

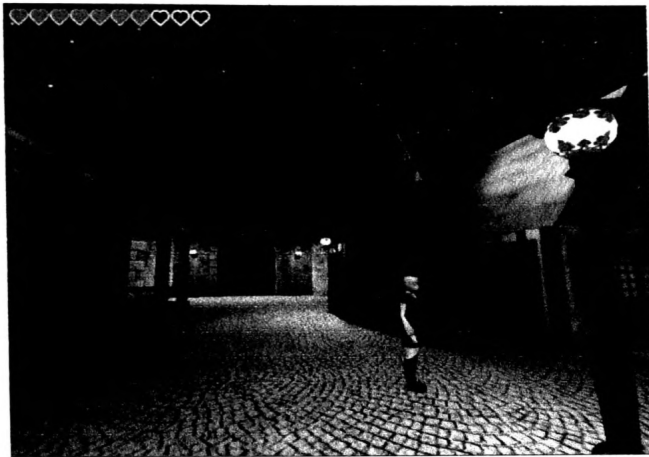


Figure 3.5 A Game Created with Reality Factory

CHAPTER 4 UML DESIGNING

4.1 USE CASE DIAGRAM

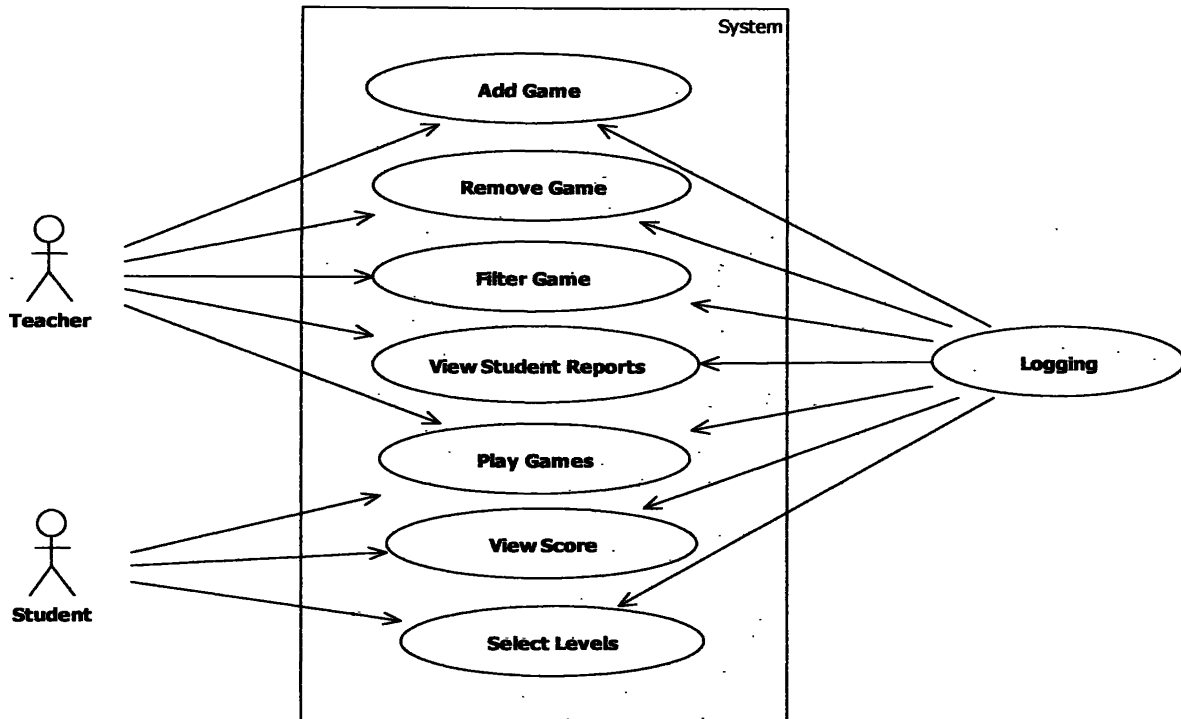


Figure 4.1 Gives the use case diagram for the game.

- **Teacher** - Teacher represents the main actor of the system. Teacher can add games to the portal remove games from the system, filter games; he/she can view the student progress. Also teacher acts like the administrator of the entire system.
- **Student** – Student represents the second main role of the system. Student is the final end use of the system. Student can play the games which only teacher permits him to play. Also he can select the level of the game he wants to play from finished levels. Student can see the progress of their subject knowledge.
- **Logging** – Logging use case is entirely based on the security of the system. To advance the system first of all actors have to log on to the System.

- **Add Game** – This use case is responsible with adding a game to the system. Only the Teacher can add games to the portal.
- **Remove Game** – This use case deals with removing existing games of the system. Only Teacher can remove the games from the system.
- **Filter Game** – This use case is responsible about the filtering of the games. According to the students subjective knowledge Teacher can filter games.
- **View Student Reports** – Teacher can see the progress of the students by examine the progress reports of the students.

4.1.1 Use Case Descriptions

Table 4.1 to 4.5 gives the use cases.

Table 4.1 Add Game Use Case Description

Use Case Number	1
Use Case Name	Add game to the Game Launcher
Use Case Description	To the game launcher pad actor named Teacher can add games according to the student's level.
Primary Actor	Teacher
Precondition	Teacher should log into the system before adding games.
Trigger	Pressing the add button.
Basic Flow	<ol style="list-style-type: none"> 1.) There are several games displayed 2.) Teacher should select games to be display in the launch pad 3.) Then teacher should press the add button to add the games 4.) Selected games added to the system.
Alternate Flows	Should select less than or equal 5 games to add to the launcher pad
Post Condition	Games add to the launcher pad

Table 4.2 Remove Game Use Case Description

Use Case Number	2
Use Case Name	Remove game from the Game Launcher
Use Case Description	To the game launcher pad actor named Teacher can remove games according to the student's level.
Primary Actor	Teacher
Precondition	Teacher should log into the system before adding games.
Trigger	Pressing the Remove button.
Basic Flow	<ol style="list-style-type: none"> 1.) There are several games displayed 2.) Teacher can remove the selected games 3.) Then teacher should press the remove button to remove the games 5.) Selected games removed from the system.
Alternate Flows	Should have games in between 1 and 5
Post Condition	Games remove from the launcher pad

Table 4.3 Filter Game Use Case Description

Use Case Number	3
Use Case Name	Filter games from the Game Launcher
Use Case Description	To the game launcher pad actor named Teacher can filter what kind of games should be in the game launcher.
Primary Actor	Teacher
Precondition	Teacher should log into the system before adding games.
Trigger	Pressing the Filter button.
Basic Flow	<ol style="list-style-type: none"> 1.) There are several games displayed 2.) All the games are with different game types. 3.) Teacher can select either game type is language or mathematics. 4.) After selecting the game type teacher should press on filter button.
Alternate Flows	Should select either type from the game.
Post Condition	Display selected types of game sin the launcher pad.

Table 4.4 View Reports Use Case Description

Use Case Number	4
Use Case Name	View Reports
Use Case Description	Student actor as well as the teacher actor can check the reports. From student's part they can see their previous marks as well as teachers can see student's level in each type of games.
Primary Actor	Both Teacher and Student
Precondition	Teacher as well as the student should log into the system before adding games.
Trigger	Pressing the View Report button.
Basic Flow	<p>Student:</p> <ol style="list-style-type: none"> 1.) In each logging student can see view report button. 2.) After pressing the View report student can see their history Report. <p>Teacher:</p> <ol style="list-style-type: none"> 3.) In each logging teach can see view report button. 4.) Teacher can view the student's report.
Alternate Flows	<p>Student:</p> <p>They can see only their marks</p> <p>Teacher:</p> <p>They can see marks on each and every student.</p>
Post Condition	Display previous records and marks.

Table 4.5 Launch Game Use Case Description

Use Case Number	5
Use Case Name	Launch Game
Use Case Description	Student actor as well as the teacher actor can play the games.
Primary Actor	Both Teacher and Student
Precondition	Teacher as well as the student should log into the system before playing the games.
Trigger	After reach to the game point in game launcher.
Basic Flow	<p>Student:</p> <p>Teacher:</p> <ol style="list-style-type: none"> 1.) In game launcher it has several games. 2.) By selecting the game either student or teacher can play the game.
Alternate Flows	<p>Student:</p> <p>Teacher:</p> <p>Can play only one game at a once.</p>
Post Condition	Teacher or Student can play the game.

4.2 CLASS DIAGRAM

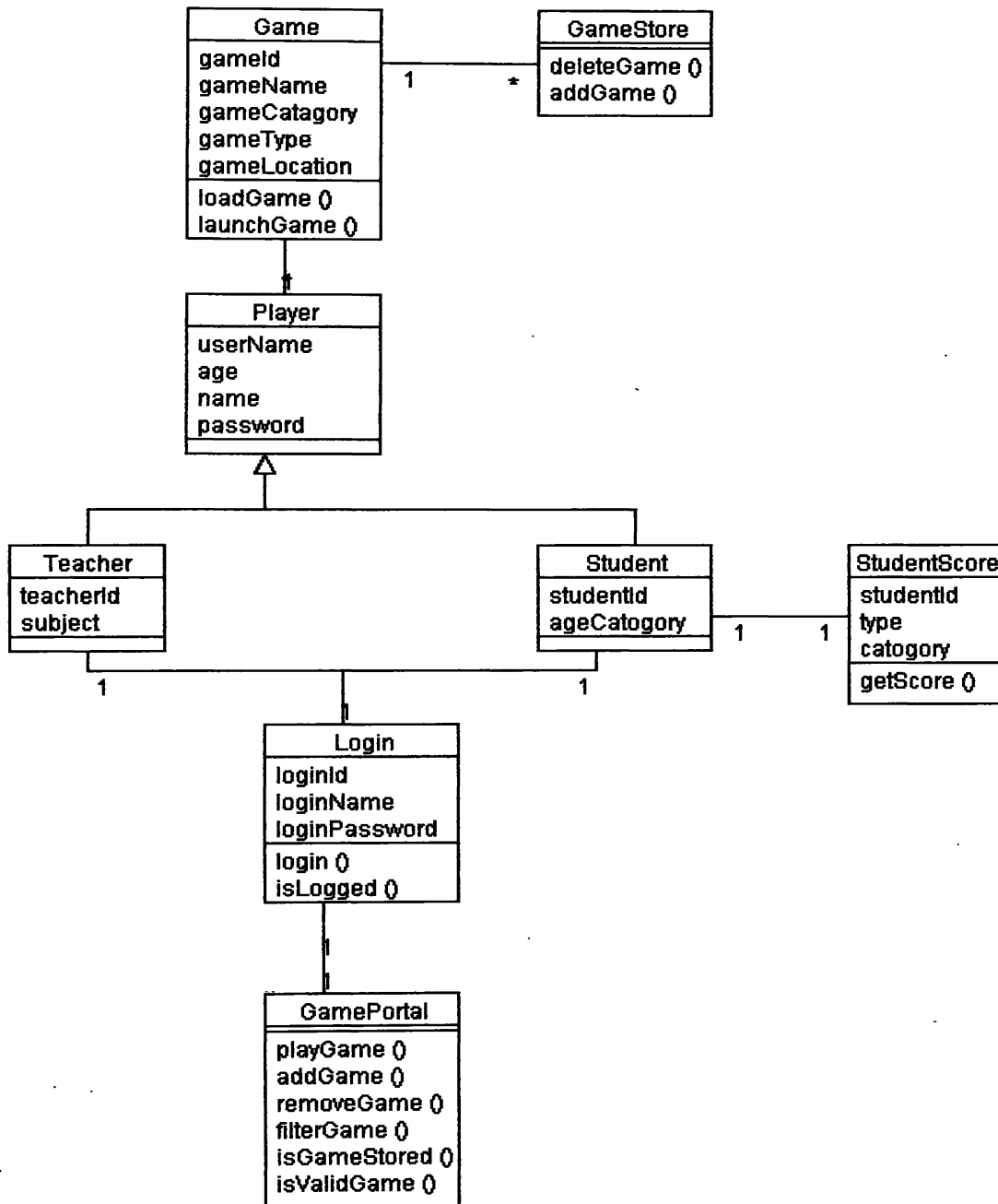


Figure 4.2 Class Diagram

Figure 4.2 indicates the class diagram for the edutainment launch pad and its functionalities where the Player class is specialized into the teacher and student, the main entity classes in the system. Given figure indicates corresponding methods and attributes for each and every class.

4.3 SEQUENCE DIAGRAMS

According to the UML design the sequence diagrams is drawn as in the figure 4.3 to 4.9 and tables 4.6 to 4.12.

4.3.1 Teacher can add games to the Launch Pad

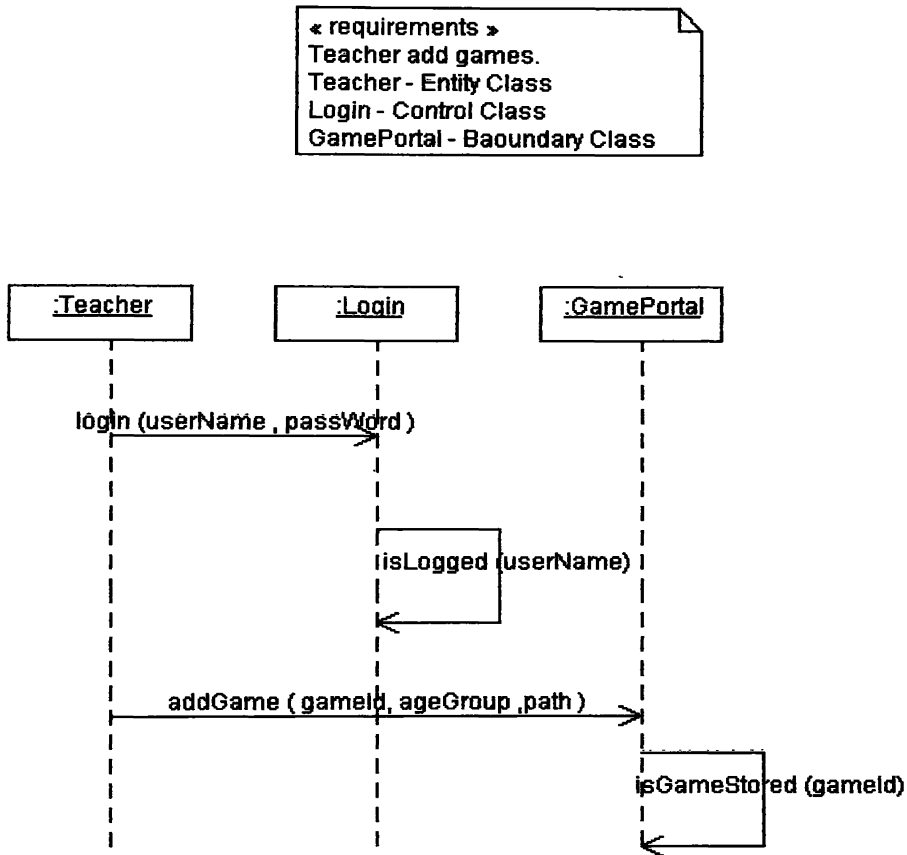


Figure 4.3 Teacher Add Games

Table 4.6 Teacher Add Games

Involved Classes	<ul style="list-style-type: none"> • Teacher • Login • GamePortal
Pre Condition	To add a game Teacher must first log into the system using username and password.
Description	This scenario explains about the class behaviors when Teacher adds a game to the Launcher.

4.3.2 Teacher can Remove Games from the LaunchPad

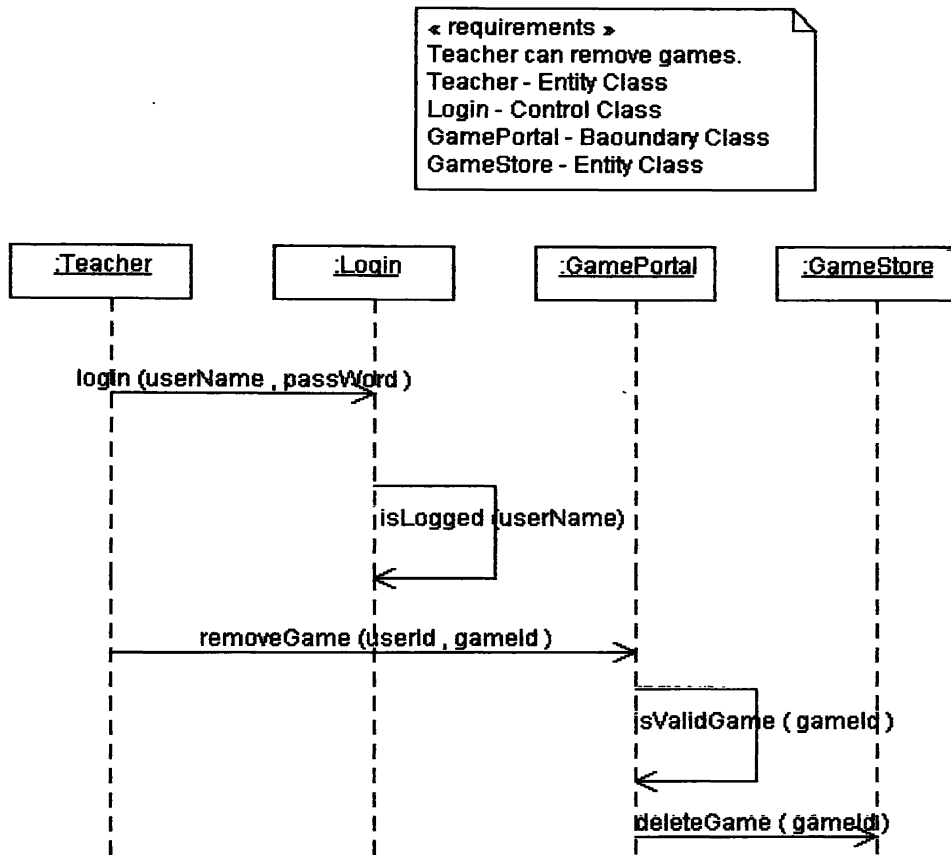


Figure 4.4 Teacher remove games

Table 4.7 Teacher remove games

<p>Classes</p>	<ul style="list-style-type: none"> • Teacher • Login • GamePortal • GameStore
<p>Pre Condition</p>	<p>To remove a game Teacher must first log into the system using username and password.</p>
<p>Description</p>	<p>This scenario explains about the class behaviors when Teacher remove unwanted games from the Launcher. The added games were store in the GameStore class.</p>

4.3.3 Teacher can filter the games

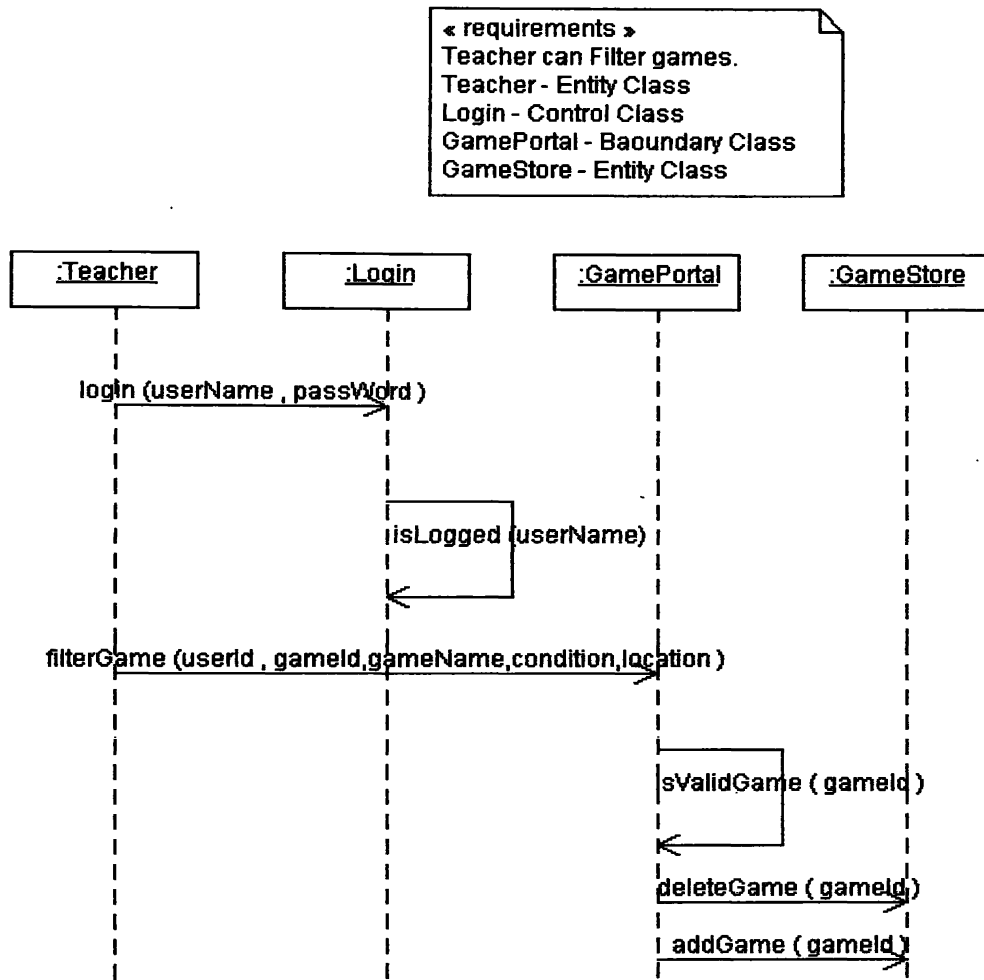


Figure 4.5 Teacher Filter Games

Table 4.8 Teacher Filter Games

<p>Classes</p>	<ul style="list-style-type: none"> • Teacher • Login • GamePortal • GameStore
<p>Pre Condition</p>	<p>To remove a game Teacher must first log into the system using username and password.</p>
<p>Description</p>	<p>This sequence diagram explains the class behavior of the system when Teacher filters the game in the Game Launcher. Teacher can add or remove games from the Launcher as needed.</p>

4.3.4 Teacher can play games

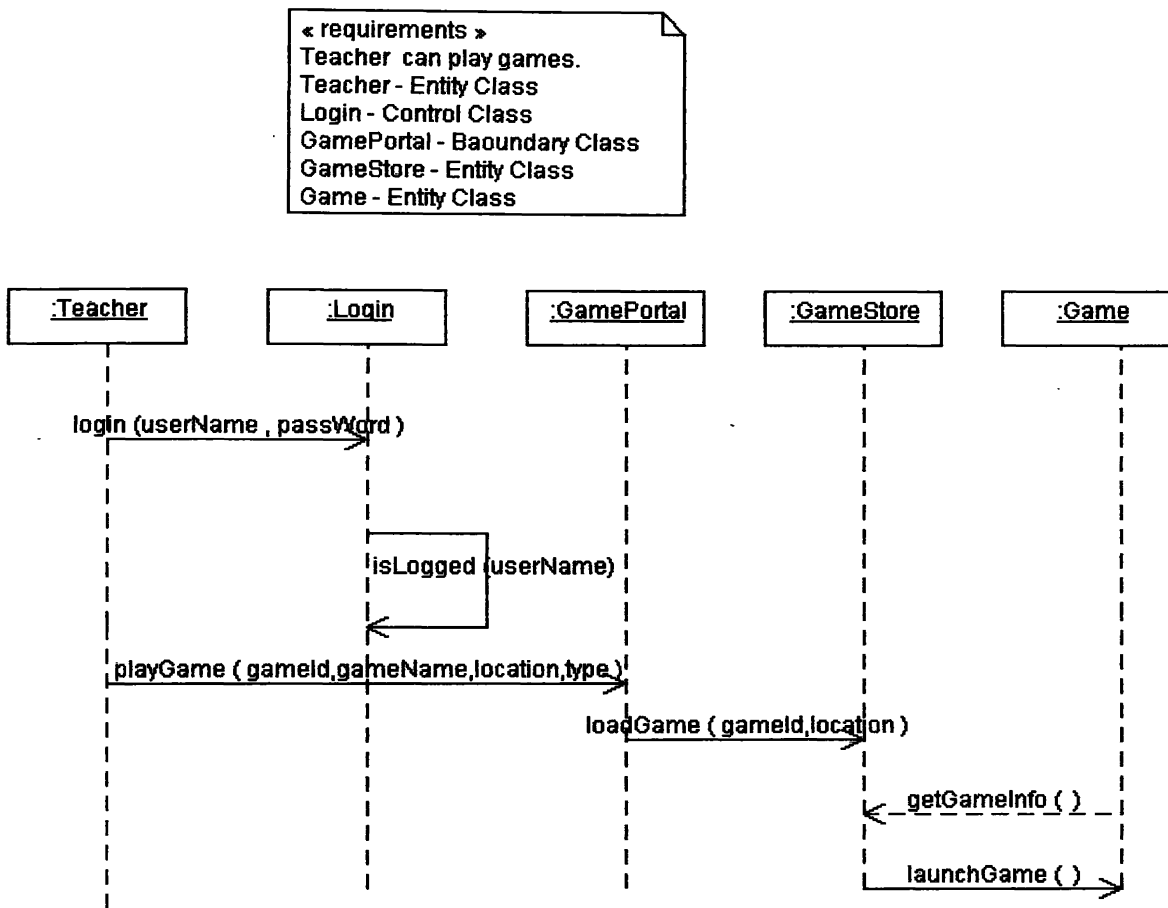


Figure 4.6 Teacher can play games

Table 4.9 Teacher can play games

<p>Classes</p>	<ul style="list-style-type: none"> • Teacher • Login • GamePortal • GameStore • Game
<p>Pre Condition</p>	<p>To remove a game Teacher must first log into the system using username and password.</p>
<p>Description</p>	<p>This describes the class behaviors and method calling of the system when Teacher play a game.</p>

4.3.5 Teacher can view student reports

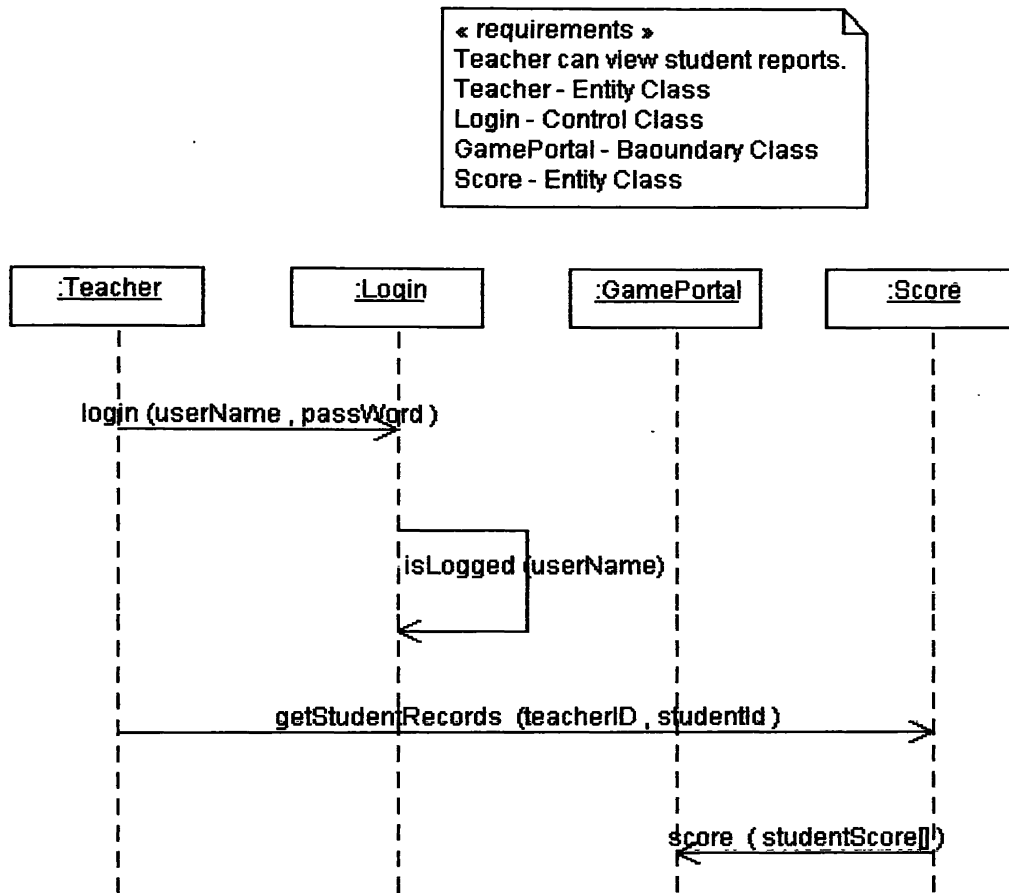


Figure 4.7 Teacher view reports

Table 4.10 Teacher view student reports

<p>Classes</p>	<ul style="list-style-type: none"> • Teacher • Login • GamePortal • Score
<p>Pre Condition</p>	<p>To remove a game Teacher must first log into the system using username and password.</p>
<p>Description</p>	<p>This sequence diagram explains the system behavior when Teacher view student Reports. Here Teacher can view the progress of the students.</p>

4.3.6 Student can play games

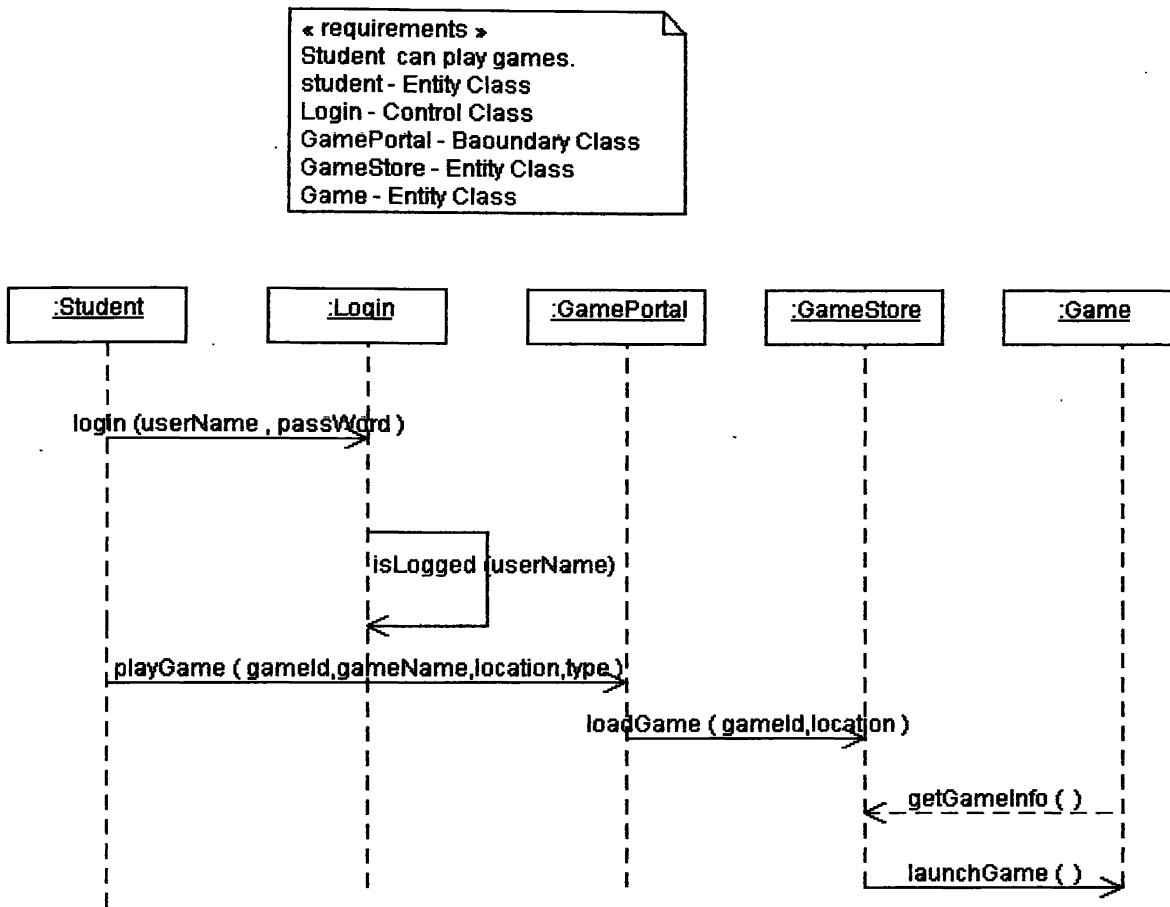


Figure 4.8 Student play games

Table 4.11 Student play games

<p>Classes</p>	<ul style="list-style-type: none"> • Teacher • Login • GamePortal • GameScore • Game
<p>Pre Condition</p>	<p>To remove a game Teacher must first log into the system using username and password.</p>
<p>Description</p>	<p>This senarion shows the method invocation when Student play games.</p>

4.3.7 Student can view reports

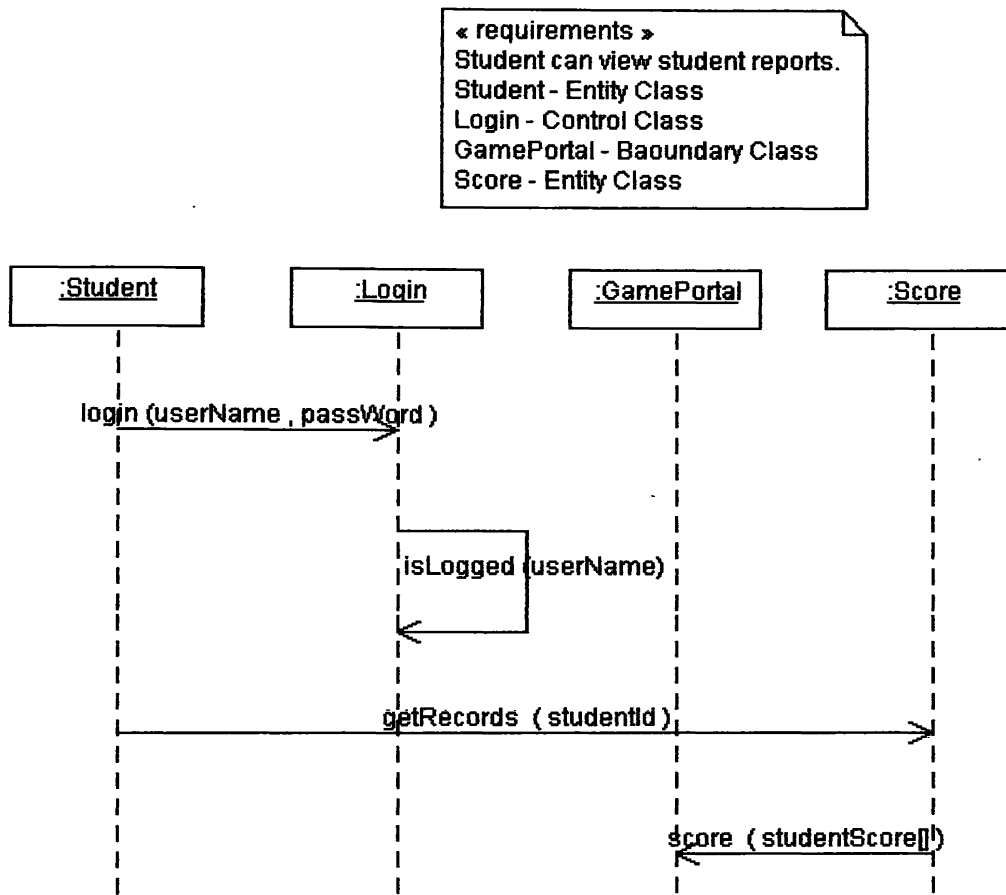


Figure 4.9 Student view reports

Table 4.12 Student view reports

Classes	<ul style="list-style-type: none"> • Student • Login • GamePortal • Score
Pre Condition	To remove a game Student must first log into the system using username and password.
Description	This sanarion is responsible for the method invocation when student view his/her report.

CHAPTER 5 GAME CONCEPTS & ANALYSIS

A game-concept document expresses the core idea of the game. It is a one- to two-page document that is necessarily brief and simple in order to encourage a flow of ideas. The target audience for the game concept is all those to whom we want to describe our game.

A game concept should include the following features:

- Introduction
- Background (optional)
- Description
- Key features
- Genre
- Platform(s)
- Concept art (optional)

This chapter discusses some of the game concepts that were used in this project.

5.1 ASCENDING TRAIN (OR DESCENDING TRAIN)

Table 5.1 gives the summary of the Ascending train game.

Table 5.1 Ascending Train

Introduction	This game was designed to improve the mathematical skills of the students. The main objective of this game is to teach students about the ascending and descending order of the numerical numbers.
Background	There are few coaches in ground with a number on it. Also there is a train engine.
Description	You have to collect the coaches using train engine in ascending order to make a train. If you collect a coach with a wrong number the game will be reset and you have to start from beginning scene in figure 5.1.
Platform	Windows / Linux

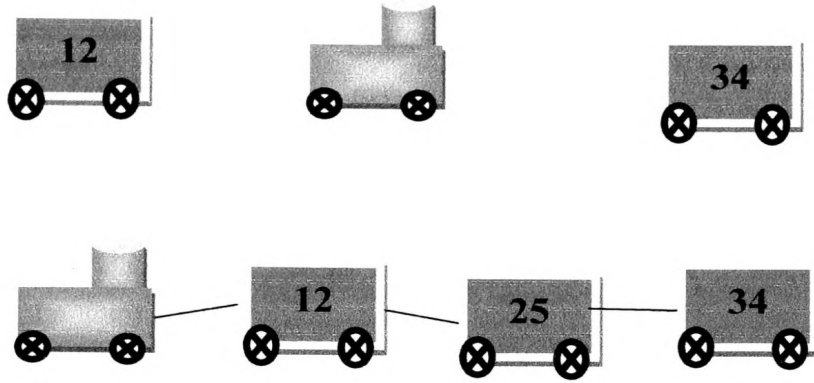


Figure 5.1 Ascending Train

5.2 ODD/EVEN NUMBER SEPARATOR

Table 5.2 gives the scene of odd/even separator game.

Table 5.2 Odd/Even Number Separator

Introduction	This game was designed to improve the mathematical skills of the students. The main objective of this game is to teach students odd and even numbers.
Background	There are few bouncing balls inside a box with two parts separating with a moving gate.
Description	You have to separate some bouncing balls using a moving gate. You have to put odd numbers in right side and even numbers in left side as in figure 5.2.
Platform	Windows / Linux

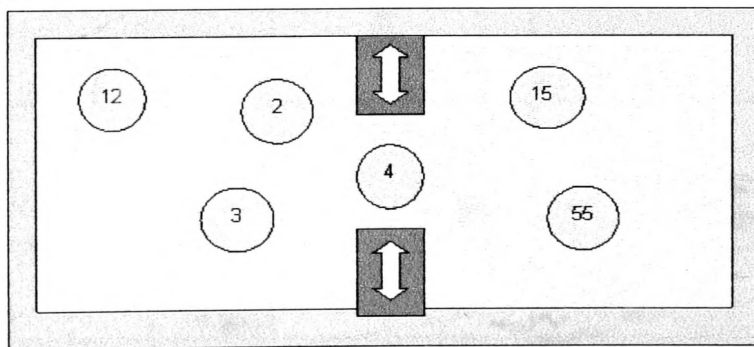


Figure 5.2 Odd/Even Number Separator

5.3 DISTANCE AND DIRECTIONS (TREASURE HUNT)

Table 5.3 gives the scene of distance and directions game.

Table 5.3 Treasure Hunt

Introduction	This game was designed to improve the mathematical skills and to teach about the main directions NORTH, EAST, SOUTH, and WEST. Also this game tries to teach how to count.
Background	There is a map with a pirate and a treasure.
Description	You have to move the pirate step wise to the treasure by avoiding obstacles. You can have a treasure hunt based on knowledge of directions and distance as figure 5.3.
Platform	Windows / Linux

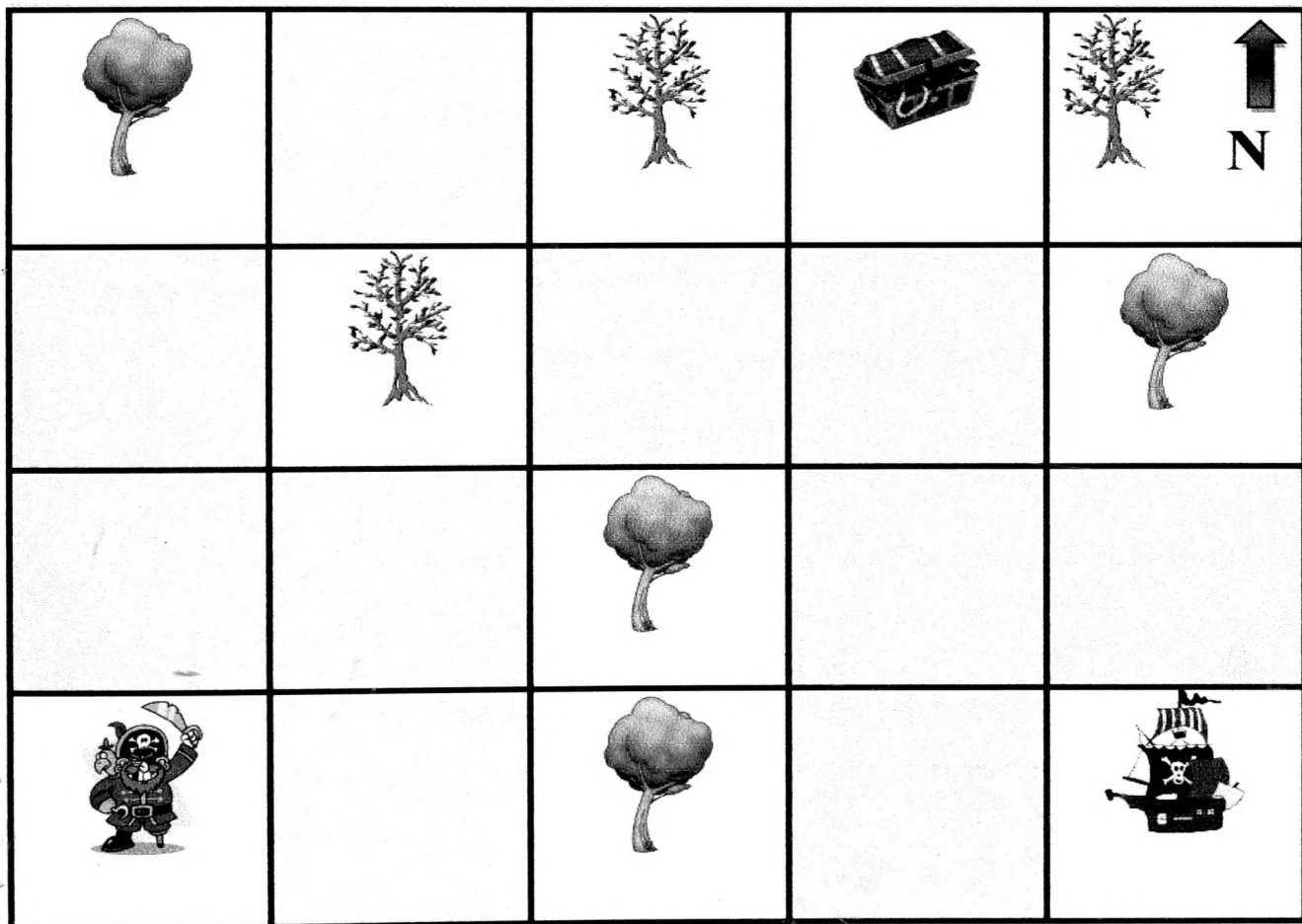


Figure 5.3 Treasure Hunt

5.4 VIRTUAL SHOP

Table 5.4 gives the scene of virtual shop game.

Table 5.4 Virtual Shop

Introduction	This game was designed to improve the skills of using money and to improve the billing & balance, selecting necessary items for the money they have, measure the weight of items, & separation.
Background	There is a Kids shop and student given the money and the item list to buy. Student has to click and order the items and finally have to pay the bill.
Description	<ol style="list-style-type: none"> 1) Students have to buy a list of items from their school Shop (figure 5.4). 2) Mother has given ----- Rupees for that. (Example 2 – 50 Rupees Notes, 1- 20 Rupees Note & 3- 5 Rupees Coins 1- 1 Rupees Coin) 3) Student visits virtual shop 4) Order items as per the money they have. 5) Pay amount of money using virtual coins and notes which have being given by Mother). 6) Collect the balance 7) If they want to buy any more (for the balance) go back to 4) 8) Go to home 9) Measure the weight of items and separate them based on that.
Platform	Windows / Linux

Kids Shop



Figure 5.4 Virtual Shop

CHAPTER 6 SYSTEM DEVELOPMENT

6.1 DEVELOPMENT ENVIRONMENT

A personal computers with processor 3.0 GHz Intel Pentium 4, RAM 2GB and Windows XP as operating system was used to implement the system.

6.1.2 Software Environment

Open source free license softwares were used through out this project.

Table 6.1 Software Development Environment

IDE	Eclipse Europe
Languages	Java 1.5 Java 2D graphics package Jasper Reports
Operating system	Windows XP
Third Party Components and Tools	<ul style="list-style-type: none"> ▪ Scirra Construct ▪ Java Monkey Engine ▪ Reality Factory ▪ Text pad ▪ iReport
Enhancement tools	<ul style="list-style-type: none"> ▪ Blender ▪ Gimp
Database	MySql

Table 6.1 depicts the various types of softwares used in the system.

6.2 API USED FOR IMPLEMENTATION

- Java 1.5 API
- Java 2D graphics API
- Jasper Reports 3.1.4 API
- iReport user guide
- Scirra construct API
- jMonkey Engine API

6.3 INTEGRATING DEVELOPMENT ENVIRONMENT

Figure 6.1 illustrates the class hierarchy of the eclipse development environment. Coding standards, all the algorithms and naming Conventions were according to the Virtusa policies. We have used some open source freely available jars as required for the purpose.

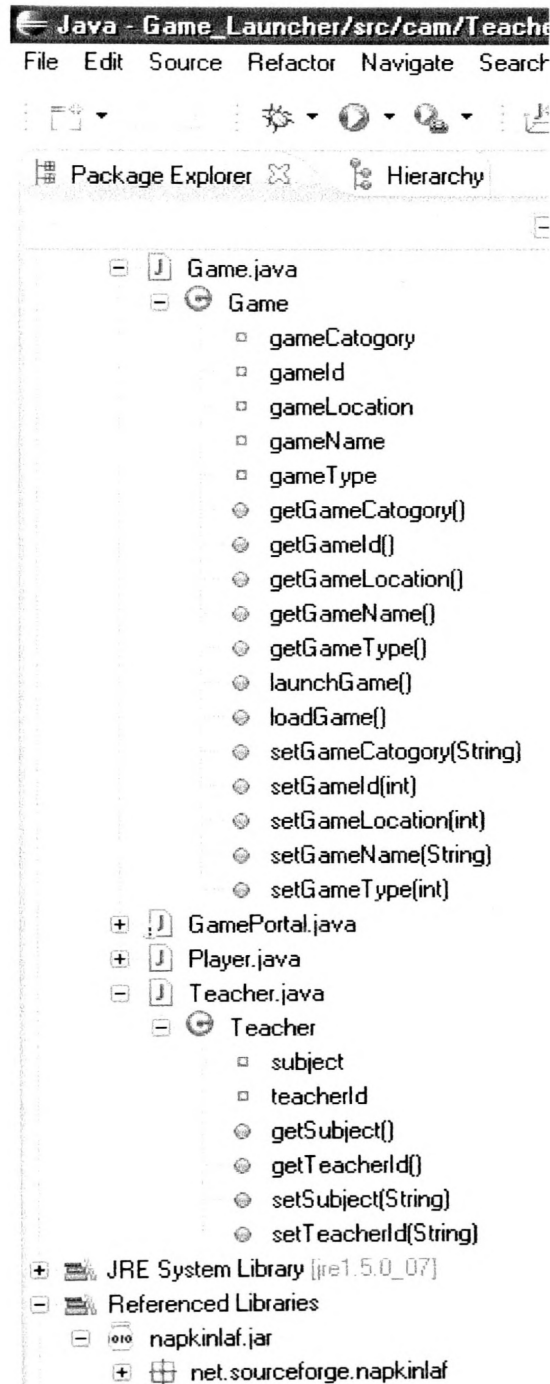


Figure 6.1 Development Environment

6.4 REPORT GENERATION

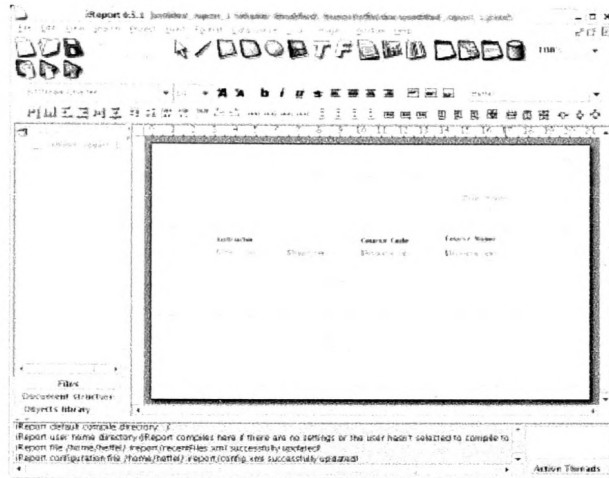


Figure 6.2 iReport Development Environment

Report generation was conducted by a separate team within the project team. They were conducted this by using Jasper Reports and reports were designed using iReport tool. Which creates *.jrxml file. By using Jasper API data is subsequently sent to the .jrxml file. Figure 6.2 illustrates an iReport development environment.

CHAPTER 7 SYSTEM TESTING & DEPLOYMENT

7.1 TEST STRATEGY

Considering the nature of the system, testing was found to be a challenging task. Testing is conducted in several steps. In development time system is conducted unit testing for already identified test cases and after integrating the entire system the system testing was conducted by the QA team. Figure 7.1 shows the QA testing procedure.

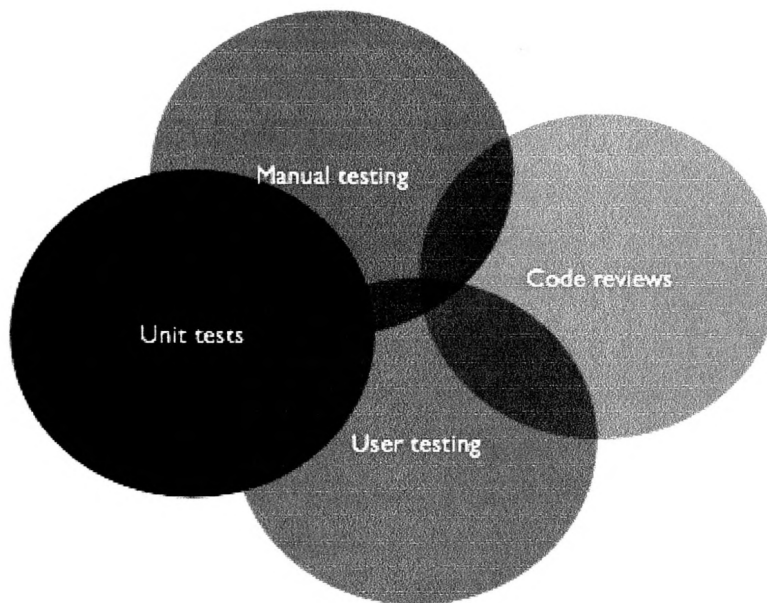


Figure 7.1 QA Testing

7.2 UNIT TESTING

Unit testing was conducted by the QA team. Each unit is tested separately before integrating them into modules to test the interfaces between modules. Unit testing has proven its value in that a large percentage of defects are identified during its use. All the public methods were tested by negative and positive manner and finally the code coverage of more than 80% was covered by the unit testing.

7.2.1 Test Cases

Test cases were written by a team member who understands the function or technology being tested, and each test case was submitted for peer review. Figure 7.2 and Table 7.1 shows the scene of a testcases.

- **Test Case 1 :**

Prerequisite



Figure 7.2 Pre Requisite

Table 7.1 Test Case 1

Test case ID	Test Case Description	Prerequisite	Test Procedure	Input Data	Expected Result	Actual Result	Test Result
	Teacher trying to add a game to the launch pad	User login to the system with appropriate login credentials.	After the logging. Select add game from the menu bar. Games select and click on add selected games button.		Selected game(s) should added to the system	Selected game(s) added to the launch pad	Test Case succeeded

Expected Result

Figure 7.3 and Table 7.2, 7.4 shows the expected result and pre- requisites.



Figure 7.3 Expected Result

- Test Case 2 :

Prerequisite

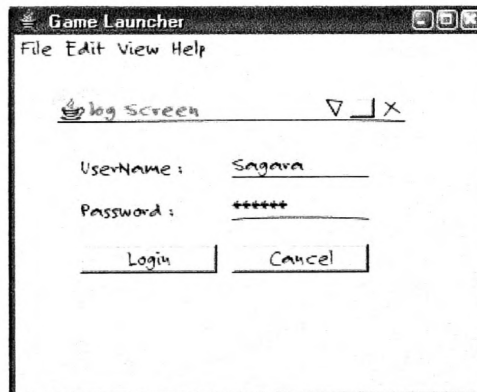


Figure 7.4 Logging Screen

Table 7.2 Test Case 2

Test case ID	Test Case Description	Prerequisite	Test Procedure	Input Data	Expected Result	Actual Result	Test Result
	Student logging to the system	Launch pad should open.	After the loading of launch pad. Student should enter the user name and password. Then click on logging	Student details with correspond usernames & passwords may need to add to the database.	Logged to the launch pad viewer with student's details	Logged to the launch pad student's details may not appear	Selected test case is success. Not completely executed.

Expected Result

Figure 7.5 shows the expected result.

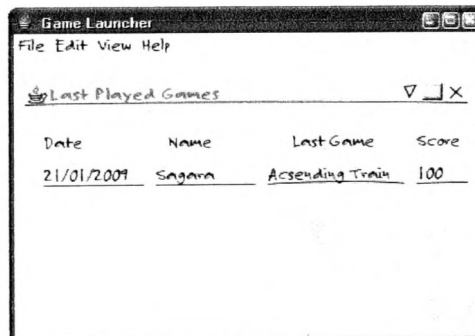


Figure 7.5 Student Last Played Game

7.3 SYSTEM TESTING

In this project system testing was conducted by the QA team. The entire system is tested as per the requirements.

7.4 DEPLOYMENT ENVIRONMENT

The main Deployment environment is based on hardware environment & software environment. Hardware environment describes about the physical requirements of the system to set up the software. On the other hand, software environment describes about the software environment of the user's machine and required software to run the game portal. After integrating the system, it is zipped into a jar file. Installing the system in user's machine involves copying and pasting the jar into a specific location.

7.4.1 Hardware Requirements

A personal computer with processor 3.0 GHz Intel Pentium 4, RAM 512MB

7.4.2 Software Requirements

Pre requisites: Table 7.3 shows the required pre requisites of the system.

Table 7.3 Deployment of the software

Operating System	Windows XP / Windows Vista / Linux
Packaging Tool	Jar file
Third Party Components and Tools	Not used
Setup Machine	MySql

CHAPTER 8 CONCLUSION

8.1 CONCLUSION

The main objective of this project was to implement a launch pad for edutainment software suit for primary school children by using open source tools and freely available softwares. Reusability and extensibility issues could be achieved to evolve the system.

Finally, the main objective of the project was achieved successfully and the software was very effective. However continues monitoring must be done in order to check whether the system meets its goal in the long run.

8.2 FUTURE CONSIDERATION

For further consideration of this project it is better to add new features like voice identification for software to be used by the blind or disable children. Also this project can be implementing for Linux platform to be use in OLPC laptops.

As this is a menu driven application, it can be modified to be a more user friendly application. In reporting it only displays the reports by using the scores that a student earns in games. This can be modified to show in a graphical representation

REFERENCES

- [www1] Agile Software Development or Home page, URL: [Http://en.wikipedia.org](http://en.wikipedia.org), 26th January 2009
- [www2] Manifesto for Agile Development or Home page, URL: <http://agilemanifesto.org/>, 26th January 2009
- [www3] What is Agile Development or Homepage, URL: <http://www.javalobby.org>, 27th January 2009
- [www4] The History of java technology or Homepage, URL: <http://java.com>, 27th January 2009–
- [www5] Java History with Tutorial or Home Page, URL: <http://www.freejavaguide.com>, 28th January 2009
- [www6] Developer Resources for Java Technology or Home Page, URL: <http://java.sun.com/>, 28th January 2009
- [www7] Java News and Resources or Home Page, URL: <Http://www.cafeaulait.org>, 29th January 2009
- [www8] General Image Manipulation Program or Homepage, URL: <http://en.wikipedia.org>, 29th January 2009
- [www9] Unified Modeling Language or Homepage, URL: <http://en.wikipedia.org>, 29th January 2009
- [www10] UML or Homepage, URL: <http://www.sparxsystems.com>, 29th January 2009
- [www11] What is MySql or Homepage, URL: <http://dev.mysql.com/doc>, 30th January 2009
- [www12] Introduction to Blender or Homepage, URL: <www.blender.org>, 2nd February 2009
- [www13] Blender or Homepage, URL: <http://en.wikipedia.org>, 2nd February 2009
- [www14] iReport Graphical Designer or Homepage, URL: <http://www.jasperforge.org>, 3rd February 2009
- [www15] Jasper Reports and iReports or Home Page, URL: [http:// www.ireport.com/](http://www.ireport.com/), 11th February 2009
- [www16] Eclipse or Homepage, URL: [http:// www.eclipse.org](http://www.eclipse.org), 11th February 2009

- [www17] Introduction to Eclipse or Homepage, URL: <http://en.wikipedia.org>, 12th February 2009
- [www18] Introduction to Game or Homepage, URL: <http://en.wikipedia.org>, 12th February 2009
- [www19] Introduction to Game or Homepage, URL: <http://gpwiki.org>, 13th February 2009
- [www20] Reality Factory or Homepage, URL: <http://www.realityfactory.info>, 14th February 2009
- [www21] Reality Factory or Homepage, URL: <http://gpwiki.org>, 14th February 2009
- [www22] jMonkey Engine 3D game engine or Homepage, URL: <http://www.jmonkeyengine.com/>, 14th February 2009
- [www23] Introduction to jMonkeyEngine or Homepage, URL: <http://gpwiki.org>, 14th February 2009
- [www24] Scirra Construct or Homepage, URL: <http://www.scirra.com/>, 17th February 2009
- [www25] Illustrate Informative Use Cases or Homepage, URL: <http://www.altova.com/>, 17th February 2009
- [www26] Object Oriented Programming or Homepage, URL: <http://en.wikipedia.org>, 18th February 2009
- [www27] Software Testing or Homepage, URL: <http://www.ece.cmu.edu>, 19th February 2009
- [www28] Software Testing Information or Homepage, URL: <http://www.onestoptesting.com/>, 22nd February 2009
- [B29] Rumbaugh, J. (2004) The Unified Modeling Language Reference manual, Addison Wesley Longman, Inc., 3-66
- [B30] Quatrani, T. Visual Modeling with Rational Rose 2000 and UML, Publisher Pearson Education India, 77-85
- [B31] Eliens, A. (2000) Principle of Object-Oriented Software Development, 2nd Edition, Pearson Education Limited 2000. 18-35
- [B32] Sommerville. (1995). The fifth edition of Software Engineering. Addison Wesley Publishers in autumn, pp.210-400
- [B33] Lieberman, H., Liu, H., Singh, P., Barry, and B.: Beating common sense into interactive applications. Game Magazine 25(4) (2004) 63-76

INDEX

A		J	
Actor	13	Java	4, 5
Aggregation	16	L	
Agile	21	lifecycle	21
association	14, 16	M	
B		Mathematical	3
Blend	4	methodologies	21, 22
C		Methods	15
class	13, 14, 15, 17	O	
Class diagrams	14	Object	4, 14, 15
composition	6, 16	objects	14, 24
Computer	3, 9	Objects	15
Construct	4, 24	OOP	14
D		open source	24
DirectX	12, 24	P	
E		Properties	15
Edutainment	2	Prototyping	21
engine	10, 24	R	
F		radiosity	28
functions	10, 13	requirements	2, 4, 21, 23
G		S	
Game	1, 2, 23	Sequence diagram	14
Gimp	4, 6	software	21, 22
I		Software Engineering	21
Inheritance	17	U	
iteration	21	UML	4, 13

Use case	13	Virtusa	1
Use Case	13	W	
V		waterfall	21
Virclipse	4, 6	Windows	10, 12

National Digitization Project

National Science Foundation

Institute : Sabaragamuwa University of Sri Lanka


1. Place of Scanning : Sabaragamuwa University of Sri Lanka, Belihuloya

2. Date Scanned : ...2017-09-22.....

3. Name of Digitizing Company : Sanje (Private) Ltd, No 435/16, Kottawa Rd,
Hokandara North, Arangala, Hokandara

4. Scanning Officer

Name :S.A.C. Gandaruman.....

Signature :.....

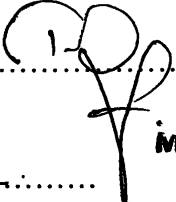
Certification of Scanning

I hereby certify that the scanning of this document was carried out under my supervision, according to the norms and standards of digital scanning accurately, also keeping with the originality of the original document to be accepted in a court of law.

Certifying Officer

Designation : LIBRARIAN.....

Name : T.N. NEIGHSOOREL.....

Signature :.....

Date : ...2017-09-22.....

Mrs. T.N. NEIGHSOOREL
(MSSc, PGD, ASLA, BA)
Librarian
Sabaragamuwa University of Sri Lanka
P.O. Box 02, Belihuloya, Sri Lanka
Tel: 0094 45 2280045
Fax: 0094 45 2280045

"This document/publication was digitized under National Digitization Project of the National Science Foundation, Sri Lanka"