

LAUNCH PAD FOR EDUTAINMENT SOFTWARE SUITE

BY

**A.I.SUWANDARATHNE
(03/AS/011)**

**This thesis is submitted in partial fulfillment of the requirements for the
degree of Bachelor of Science in Physical Sciences.**

**Department of Physical Sciences,
Faculty of Applied Sciences,
Sabaragamuwa University of Sri Lanka.
Belihuloya.**


April 2009

DECLARATION

The content described in this thesis was practically implemented by me at the Virtusa Corporation and the Faculty of Applied Sciences under the supervision of Mr. Gayan Subasinghe and Dr. R.G.N. Meegama and the report described on this thesis has not been submitted by any one for another degree.

22-04-2009

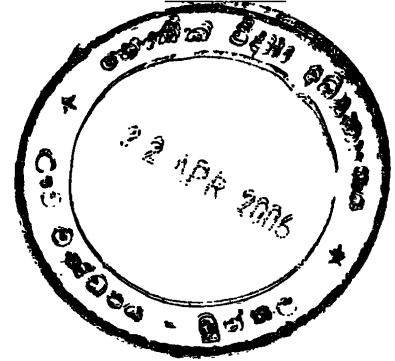
Date



.....

Asela I. Suwandarathne

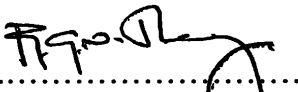
(03/AS/011)



CERTIFICATE OF APPROVAL

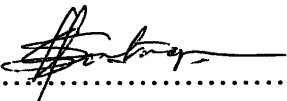
We hereby declare that this thesis is from the student's own work and effort, and all other Sources of information used have been acknowledged. This thesis has been submitted
With our approval.

Dr. R.G.N Meegama,
Senior Lecturer,
Department of Statistics and Computer Science,
University of Sri Jayewardenepura,
Gangodawila,
Nugegoda


.....
(Signature/Internal Supervisor)

.....
(Date)

Mr. Gayan Subasinghe,
ATC Project Manager,
Virtusa (Pvt.) Ltd,
117 Sir Chittampalam A.Gardiner Mw,
Colombo 2.


.....
(Signature/External Supervisor)

.....
02/04/2009
.....
(Date)

Dr. C.P.Udawatta
Head/Department of Physical Sciences,
Faculty of Applied Sciences,
Sabaragamuwa University of Sri Lanka,
Belihuloya.


.....
Signature

.....
2009.04.22
.....
(Date)

Affectionately Dedicated To My Parents and Sister

ACKNOWLEDGEMENT

Carrying out an industrial training in a virtual context is challenging and requires significant effort and support from all concerned. In this respect, I have been blessed with many people who extended their support and encouragement throughout the training period.

My heartfelt gratitude goes to my supervisor, Dr. Gayan Meegama Department of Statistics and Computer Science, University of Sri Jayewardenepura, who constantly guided me during the training period. Undoubtedly, his supervision helped me looking into new dimensions of motivation.

Secondly, I would like to 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., especially I wish to express my deepest gratitude to my external supervisor Mr.Gayan Subasinghe, ATC Project Manager, Virtusa Corporation, for his advice, encouragement and guidance through the study and for sparing his valuable time in bringing this study to a successful completion and also to my technical leader Mr. Ramesh Maddegoda, Virtusa Corporation for his encouragement and guidance through this project.

I wish to express my sincere gratitude to Prof Mahinda Rupasinghe, the Vice Chancellor, Sabaragamuwa University of Sri Lanka, Prof. K.B. Palipane, The Dean, Faculty of Applied Sciences, Sabaragamuwa University of Sri Lanka, Dr Chandana Udawatta, Head, Department of Physical Sciences, Faculty of Applied Sciences, and Senior lecturer Dr. Nirmali Wickramarathne Sabaragamuwa University of Sri Lanka, for guiding me toward a successful completion.

I express my heart-felt gratitude towards the lectures for their cooperation through out my study and my colleagues for their individual help and guidance at all times.

Last but not least, my heartiest thanks go to my parents, the main pillars of my life, who constantly encouraged and supported me within their capacity throughout the course of this industrial training.

ABSTRACT

One of the key problems identified in using computers for primary school children is to make the students get use to the machines. It has been identified that students will use the computers more often if an attractive interactive environment, such as adventure games, was provided.

The main objective of this project is to launch an Edutainment software suit, which can help players to select each and every game according to the player's preference in the given launch pad.

The software development process used for this project was Agile. The requirements were based on the main objectives as stated by the Ministry of Education (MoE). Major requirements of this project were fulfilled by interviewing the game experts, teachers, reviewing similar applications and also reading sample documents. Unified Modeling Language (UML) was used to convert the requirements into an analysis model. Inputs from the potential users such as students and teachers and their corresponding tasks were considered when the Use Case diagram was drawn in UML diagram. 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 diagram were created. Implementations part was done using the Java language while Virclipse 2.0 was used as the IDE to implement in Java, which is a separate editing platform developed by Virtusa Corporation. For the designing, open source software, such as Blend, GNU Image Manipulation Program (GIMP) and Flash CS3 were utilized. An iterative approach was applied to each phase mentioned above to give an opportunity to MoE to get interactively involved in the development.. Evaluative feedbacks were acquired in each meeting with team members. Refinements and modifications were carried out after each team meeting to fulfill the client's expectations.

Unit testing was performed to ensure that the functionality of individual components are accurate. Finally, the individual components were integrated together and system testing was done to ensure that the necessary functionalities expected by the customers were delivered. Thorough testing was done on performance, security features, stress testing and user acceptance testing in order to fulfill some of the non functional requirements. The main objective of the project was successfully met and the Ministry of Education was fully satisfied with the successful completion of the system's functional and non functional requirements.

CONTENT

DECLARATION	I
ACKNOWLEDGEMENT	IV
ABSTRACT	V
ACRONYMS AND ABBREVIATIONS	VI
LIST OF FIGURES	VIII
LIST OF TABLES	IX
CONTENT	X
CHAPTER 1	1
INTRODUCTION	1
1.1 Introduction of Virtusa Corporation	1
1.2 About Virtusa Game Dev SIG with MoE.....	1
1.3 Project Overview	2
1.4 Major Challenges.....	2
1.5 Objectives	3
1.5.1 Major Objective	3
1.5.2 Overall Objectives.....	3
CHAPTER 2	5
REVIEW OF LITERATURE	5
2.1 Games History	5
2.2 Java Programming Language	6
2.2.1 The History of Java Technology	6
2.2.2 Java as Modern Language	7
2.2.3 Comparability between Java and other languages.....	8

LIST OF TABLES

Table2.1 : Comparison between computer languages	9
Table2.2 : Elements of a sequence diagram	18
Table2.3 : Game Engine Overview	22
Table4.1 : Game adding to the Launch pad	37
Table4.2 : Game removing from the Launch pad.....	38
Table4.3: Game filter from the Launch pad	38
Table4.4: View Reports	39
Table4.5: Game Launch	40
Table4.6 : Add game to the launch pad description	43
Table4.7 : Remove Game From the launch pad description.....	44
Table4.8 : Filter games from the launch pad description	45
Table4.9 : Student/Teacher can view scores description.....	46
Table4.10 : Teacher and Student can launch games.....	48
Table5.1 : Game Concept for the Ascending Train game	50
Table5.2 : Game Concept for the ODD/EVEN number separator game.....	51
Table5.3 : Game Concept for the Distance and Directions game.....	52
Table5.4 : Game Concept for the Virtual shop game	53
Table6.1 : Development Environment.....	54
Table7.1 : Sample Test Cases.....	59
Table7.2: Sampls Test Case II	61
Table7.3: Software Requirement of the Deployment Environment	62

LIST OF FIGURES

Figure2.1 : Properties & Methods in a class.....	13
Figure2.2 : Inheritance.....	13
Figure2.3 : Encapsulation.....	14
Figure2.4 : Class Symbol.....	17
Figure2.5 : GIMP tool.....	20
Figure3.1 : Agile Development Process.....	29
Figure3.2 : Game Development Life Cycle.....	30
Figure3.3 : A Demo Game created with Scirra Construct.....	32
Figure3.4 : A Demo game created with Java Monkey Engine.....	33
Figure3.5 : Demo game created with Reality Factor.....	35
Figure4.1 : Use Case Diagram for our system.....	36
Figure4.2 : Class Diagram for the system.....	41
Figure4.3 : Add game to the Launch Pad.....	42
Figure4.4 : Remove game from the launch pad.....	43
Figure4.5 : Filter games from the launch pad.....	44
Figure4.6 : Student Can View Reports.....	45
Figure4.7 : Teacher can view reports.....	46
Figure4.8 : Teacher can launch games.....	47
Figure4.9 : Student can launch games.....	47
Figure5.1 : Overview of the Ascending Train Game.....	50
Figure5.2 : Overview of the ODD/EVEN number separator Game.....	51
Figure5.3 : Overview of the Distance and directions Game.....	52
Figure5.4 : Overview of the Virtual Shop Game.....	53
Figure6.1 : Class Hierarchy.....	55
Figure6.2 : iReport.....	56
Figure7.1 : Before implement add game test case.....	58
Figure7.2 : After implement add game test case.....	59
Figure7.3 : Before implement login test case.....	60
Figure7.4 : After implement login test case.....	60

MB	Mega Byte
MoE	Ministry of Education
OLPC	One Laptop Per Child
OOP	Object Oriented Programming
OpenAL	Open Audio Library
OpenGL	Open Graphics Library
OSGI	Open Services Gateway initiative
PC	Personal Computers
QA	Quality Assurance
RAM	Random Access Memory
RGB	Red Green Blue
SDK	Software Development Kit
SE	Software Engineer
SIT	System Integration Testing
SRS	System Requirement Specification
UAT	User Acceptance Testing
UI	User Interface
UML	Unified Modeling Language
VGA	Video Graphic Array
WWW	World Wide Web

ACRONYMS AND ABBREVIATIONS

ANSI	American National Standards Institute
API	Application Programming Interface
AVI	Audio Video Interleave
CARB	Code Architecture Review Board
CSR	Corporate Social Responsibilities
ECMA	Ecma International
E-R Diagram	Entity Relational Diagram
FPS	First Person Shooters
FRS	Functional Requirement Specification
Game Dev SIG	Game Development Special Interesting Group
GIF	Graphics Interchange Format
GIMP	GNU Image Manipulation Program
GIP	Global Innovation Process
GPL	General Public License
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IP	Intellectual Property
ISO	International Organization for Standardization
IT	Information Technology
J2EE	Java 2, Enterprise Edition
J2ME	Java 2 Micro Edition
J2SE	Java 2, Standard Edition
JDBC	Java Database Connectivity
JOGL	Java Open Graphic Library
JRE	Java Runtime Environment
LWJGL	Lightweight Java Game Library

TESTING & DEPLOYMENT.....	57
7.1 Testing Strategy	57
7.2 Unit Testing	57
7.2.1 Test Cases.....	57
7.2.2 Test Data	58
7.3 System Testing	61
7.4 Deploy Environment.....	61
7.4.1 Hardware Requirements	61
7.4.2 Software Requirements	62
CHAPTER 08.....	63
CONCLUSION	63
8.1 Future Consideration	63
REFERENCES	64
INDEX	65

3.3.2 JMonkey Engine.....	32
3.3.3 Reality Factor	33
CHAPTER 04.....	36
DESIGNING DEVELOPMENT	36
4.1 Use Diagram	36
4.1.1 Use Case Descriptions.....	37
4.2 Class Diagram.....	41
4.3 Sequence Diagrams	42
4.3.1 Add Game to the Launch Pad.....	42
4.3.2 Remove Game from the Launch Pad.....	43
4.3.3 Filter games from the Game Launcher.....	44
4.3.4 Student and Teacher can view Reports	45
4.3.5 Launch Game	47
CHAPTER 05.....	49
GAME CONCEPTS & ANALYSIS.....	49
5.1 Ascending Train (or Descending Train).....	50
5.2 Odd/Even number Separator.....	51
5.3 Distance and Directions (Treasure Hunt)	52
5.4 Virtual Shop.....	53
CHAPTER 06.....	54
DEVELOPMENT ENVIRONMENT	54
6.1 Development Environment.....	54
6.1.1 Hardware Environment	54
6.1.2 Software Environment.....	54
6.2 Application Programming Interface (API) used for implementation	54
6.3 Integrating Environment.....	55
6.4 Reporting	56
CHAPTER 07.....	57

2.3	Object Oriented Programming principles.....	11
2.3.1	Class.....	12
2.3.2	Objects.....	12
2.3.3	Properties and Methods.....	12
2.3.4	Inheritance.....	13
2.3.5	Encapsulation.....	14
2.3.6	Polymorphism.....	14
2.4	Unified Modeling Language (UML).....	14
2.5	Use Case Diagrams.....	15
2.5.1	Elements of Use Case Diagram.....	15
2.6	Class Diagrams.....	16
2.7	Sequence Diagrams.....	17
2.7.1	Entity Classes.....	18
2.7.2	Boundary Classes.....	19
2.7.3	Control Classes.....	19
2.8	Virclipse IDE.....	20
2.9	Gimp tool.....	20
2.10	Game Engines.....	21
2.10.1	Freeware Engines.....	22
2.11	Testing.....	23
2.11.1	Testing Levels.....	24
2.12	MySQL.....	25
2.13	JasperReports.....	26
2.14	IReport.....	27
CHAPTER 03.....		28
TECHNOLOGICAL DEVELOPMENT.....		28
3.1	Introduction to Software Development Methodology.....	28
3.1.1	Software Development Process.....	28
3.1.2	Agile Development.....	28
3.2	Game Development Life Cycle.....	30
3.3	Game Engines.....	31
3.3.1	Scirra Construct (Rapid Game Authoring System).....	31

CHAPTER 1

INTRODUCTION

1.1 Introduction of Virtusa Corporation

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 Canekeratne, 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 been 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 About Virtusa Game Dev SIG with MoE

Virtusa Game Development Special Interest Group (Game Dev SIG) is a knowledge sharing group. This special interest group is open for anyone who is interested in game development within Virtusa. The Game Dev SIG understands Game Development as a serious industry which requires very high technical expertise on multiple technologies. The Game Dev SIG believes that Game Development has a high business value in global level.

As a part of the corporate social responsibility (CSR) initiatives of Virtusa, Game Dev SIG involves in a voluntary project to develop edutainment software for primary schools of Sri Lanka.

1.3 Project Overview

Purpose of this proposed system is to develop simple games in an interactive and attractive manner for primary grade students attending schools in Sri Lanka. .

At the moment, the system posses several educational games in different formats. That are not integrated into a unique system. That helps primary students without concentrating on learning tools.

By this edutainment game software for the primary schools covered most of the mathematics and the language subjects for the grade 3 and grade 4 students. For the mathematics scheme we used basic arithmetic, sorting numbers and puzzles as major concepts.

Following learning techniques were used to get the optimum usage for this age group on account of the interactive learning is preferred.

- Group Work
- Observation
- Activities
- Guided Play (Guidance provided by the teachers)
- Desk work

Duration of edutainment session was requested to limit to 30 minutes of maximum by the particular schools according to the requirements collected by interviewing, subject matter experts. Virtusans have designed the initial game concepts and graphics and those will be reviewed by the subject matter experts and content experts.

1.4 Major Challenges

Following major challenges were identified during the project period. They can be listed as follows,

- To implement the whole edutainment system, using open source software.
- 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.
- For the development data stored in inbuilt notepad files, that is because to get the high efficiency.
- To acquire technical skills which are to be applied to the Virtusa standards as well as touch their main functionalities.

- All the equipments and images not copied from any place any person and everything should be under the intellectual property (IP) rights.
- All the architecture built according to the design documents. /such as Use Case, Sequence Diagrams and the class diagrams
- Project releases may upload once per week.

1.5 Objectives

It is intended as a teaching as well as the self learning tool. Basically, the students will have to work by themselves while the initial guidance/assistance will be given by the teacher. The major objectives and goals behind the project work are presented below. These are evaluated at the end of the project to ascertain whether they have been met successfully.

1.5.1 Major Objective

- 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. The games can be selected by teacher for their students and students can play games through it. It is also possible students to see the results as well as see the progress of their game skills

1.5.2 Overall Objectives

- Deploy the mathematics skills of students

Most of the primary schools are keeping their eye on increasing student skills since the small age. As an experiment, MoE decide to increase mathematical level of the students because mathematics is essential for the day to day life activities.

- Report Demonstration

Report demonstration is planned to be act in two ways. Considering students account, their skills can be developed through this. Displaying the results score on screen, the students can be motivated to achieve high scores. On teacher's point, it is visible the student's progress levels to teachers and they can advice to students easily. That's how the project implemented with the report demonstration.

- Increase market opportunities

This implies the way of interacting with the real time conditions. That helps to increase market opportunities with knowledgeable students with ability of using different algorithms.

- **Take the latest technology to students**

Most of the students lack of knowledge on latest technology. MoE is planning to familiarize the latest technology to student's mind through this. It is pity that even teachers of most of rural villages have no idea on handling computers. But development of this system, make available the latest technology to the students as well as teachers who are away from the capital cities.

- **Measure the progress of the students**

By providing some time to play these kinds of edutainment games, it is possible to improve student's language and mathematical skills. Also teachers can measure the progress of using new way of technology through evaluating their performance during grade 3 and grade 4.

CHAPTER 2

REVIEW OF LITERATURE

2.1 Games History

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. 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.

2.2 Java Programming Language

2.2.1 The History of Java Technology

In the early 90s, extending the power of network computing to the activities of everyday life was a radical vision. 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. Led by James Gosling, the team worked around the clock and created the programming language that would revolutionize our world – Java. Java was started as a project called "Oak". [www2]

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. [www1]

Unlike conventional languages which are generally designed either to be compiled to native (machine) code, or to be interpreted from source code at runtime, Java is intended to be compiled to a bytecode, which is then run (generally using JIT compilation) by a Java Virtual Machine. [www2]

In 1997, Sun approached the ISO/IEC JTC1 standards body and later the Ecma International to formalize Java, but it soon withdrew from the process. Java remains a proprietary de facto standard that is controlled through the Java Community Process. Sun makes most of its Java implementations available without charge, with revenue being generated by specialized products such as the Java Enterprise System. Sun distinguishes between its Software Development Kit (SDK) and Java Runtime Environment (JRE) which is a subset of the SDK, the primary distinction being that in the JRE the compiler is not present. [www2]

There were five primary goals in the creation of the Java language:

1. It should use the object-oriented programming methodology.
2. It should allow the same program to be executed on multiple operating systems.
3. It should contain built-in support for using computer networks.
4. It should be designed to execute code from remote sources securely.
5. It should be easy to use by selecting what was considered the good parts of other object-oriented languages.

To achieve the goals of networking support and remote code execution, Java programmers sometimes find it necessary to use extensions such as CORBA, Internet Communications Engine, or OSGI. [www2]

2.2.2 Java as Modern Language

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. From mobile phones to handheld devices, games and navigation systems to e-business solutions, Java is everywhere.

The design requirements of the JavaTM programming language are driven by the nature of the computing environments in which software must be deployed. The massive growth of the Internet and the World-Wide Web leads us to a completely new way of looking at development and distribution of software. To live in the world of electronic commerce and distribution, Java technology must enable the development of secure, high performance, and highly robust applications on multiple platforms in heterogeneous, distributed networks.

Operating on multiple platforms in heterogeneous networks invalidates the traditional schemes of binary distribution, release, upgrade, patch, and so on. To survive in this jungle, the Java programming language must be architecture neutral, portable, *and* dynamically adaptable. The system that emerged to meet these needs is simple, so it can be easily programmed by most developers; familiar, so that current developers can easily learn the Java programming language; object oriented, to take advantage of modern software development methodologies and to fit into distributed client-server applications; multithreaded, for high performance in applications that need to perform multiple concurrent activities, such as multimedia; and interpreted, for maximum portability and dynamic capabilities.

Even 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.

With the advent of Java 2 (released initially as J2SE 1.2 in December 1998), new versions had multiple configurations built for different types of platforms. For example, J2EE targeted enterprise applications and the greatly stripped-down version J2ME for mobile applications. J2SE designated the Standard Edition. In 2006, for marketing purposes, Sun renamed new J2 versions as Java EE, Java ME, and Java SE, respectively. [www3]

2.2.3 Comparability between Java and other languages

Programming languages are used for controlling the behavior of a machine (often a computer). Like natural languages, programming languages conform to rules for syntax and semantics. There are thousands of programming languages and new ones are created every year. Few languages ever become sufficiently popular that they are used by more than a few people. General Comparison of programming languages with Java indicates in following table. Table 2.1 shown a comparison of different programming languages.

Language	Intended use	Paradigm(s)	Standardized?
Assembly Language	General	- - -	No
C	System	imperative	Yes, ANSI C89, ISO C90/C99
C++	Application, System	Imperative, Object-oriented, generic	Yes, ISO
C#	Application	imperative , Object-oriented, Functional, Generic, Reflective	Yes, ECMA, ISO
Java	Application, Web	imperative , Object-oriented, Functional, Generic, Reflective	No
Python	Application, Scripting, Web	imperative , aspect-oriented, Functional, Generic, Reflective	No
Visual Basic .NET	Application, Education	Object-oriented, Event Driven	No

Table2.1 : Comparison between computer languages

Major Paradigm of using java language

- Platform Independent

Java was designed to not only be cross-platform in source form like C, but also it is compiled in binary form. Since this is impossible across processor architectures, Java is compiled to an intermediate form called byte-code. A Java program never really executes natively on the host machine. Rather a special native program called the Java interpreter reads the byte code and executes the corresponding native machine instructions. Thus, to port Java programs to a new platform all that is needed is to port the interpreter and some of the library routines. Even the

compiler is written in Java. The byte codes are precisely defined, and remain the same on all platforms. The second important part of making Java cross-platform is the elimination of undefined or architecture dependent constructs. Integers are always four bytes long, and floating point variables follow the IEEE 754 standard for computer arithmetic exactly. You don't have to worry that the meaning of an integer is going to change if you move from a Pentium to a PowerPC. In Java everything is guaranteed.

However the virtual machine itself and some parts of the class library must be written in native code. These are not always as easy or as quick to port as pure Java programs. [www4]

- **Object Oriented and Familiar**

Primary characteristics of the Java programming language include a simple language that can be programmed without extensive programmer training while being attuned to current software practices. The fundamental concepts of Java technology are grasped quickly; programmers can be productive from the very beginning.

The Java programming language is designed to be *object oriented* from the ground up. Object technology has finally found its way into the programming mainstream after a gestation period of thirty years. The needs of distributed, client-server based systems coincide with the encapsulated, message-passing paradigms of object-based software. To function within increasingly complex, network-based environments, programming systems must adopt object-oriented concepts. Java technology provides a clean and efficient object-based development platform. [B3]

- **High Security Performances**

Java technology is designed to operate in distributed environments, which means that security is of paramount importance. With security features designed into the language and run-time system, Java technology lets you construct applications that can't be invaded from outside. In the network environment, applications written in the Java programming language are secure from intrusion by unauthorized code attempting to get behind the scenes and create viruses or invade file systems.

- **Interpreted, Threaded, and Dynamic**

The Java interpreter can execute Java byte codes directly on any machine to which the interpreter and run-time system have been ported. In an interpreted platform such as Java technology-based system, the link phase of a program is simple, incremental, and lightweight. You benefit from much faster development cycles--prototyping, experimentation, and rapid development are the normal case, versus the traditional heavyweight compile, link, and test cycles.

While the Java Compiler is strict in its compile-time static checking, the language and run-time system are dynamic in their linking stages. Classes are linked only as needed. New code modules can be linked in on demand from a variety of sources, even from sources across a network. In the case of the Hot Java Browser and similar applications, interactive executable code can be loaded from anywhere, which enables transparent updating of applications. The result is on-line services that constantly evolve; they can remain innovative and fresh, draw more customers, and spur the growth of electronic commerce on the Internet. [www3]

2.3 Object Oriented Programming principles

Object-Oriented Programming (OOP) represents an attempt to make programs more closely model the way people think about and deal with the world. In the older styles of programming, a programmer who is faced with some problem must identify a computing task that needs to be performed in order to solve the problem. Programming then consists of finding a sequence of instructions that will accomplish that task. But at the heart of object-oriented programming, instead of tasks we find objects—entities that have behaviors, that hold information, and that can interact with one another. Programming consists of designing a set of objects that model the problem at hand. Software objects in the program can represent real or abstract entities in the problem domain. This is supposed to make the design of the program more natural and hence easier to get right and easier to understand.

Object-Orientation is a set of tools and methods that enable software engineers to build reliable, user friendly, maintainable, well documented, reusable software systems that fulfill the requirements of its users. It is claimed that object-orientation provides software developers with new mind tools to use in solving a wide variety of problems. Object-orientation provides a new view of computation. A software system is seen as a community of objects that cooperate with each other by passing messages in solving a problem.

Object-oriented programming is one of several programming paradigms. Other programming paradigms include the imperative programming paradigm (as exemplified by languages such as Pascal or C), the logic programming paradigm (Prolog), and the functional programming paradigm (exemplified by languages such as ML, Haskell or Lisp). Logic and functional languages are said to be declarative languages.

An object-oriented programming language provides support for the following Concepts and those described as follows. [B3]

2.3.1 Class

A class is a blueprint or prototype from which objects are created. This section defines a class that models the state and behavior of a real-world object. It intentionally focuses on the basics, showing how even a simple class can cleanly model state and behavior.

Once a class of items is defined, a specific instance of the class can be defined. An instance is also called "object".

2.3.2 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.

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. It is not uncommon to hear people talk about the "state information" associated with a particular object. For example, the state of a bank account object would include the current balance, the state of a clock object would be the current time, the state of an electric light bulb would be "on" or "off." For complex objects like a human being or an automobile, a complete description of the state might be very complex. Fortunately, when we use objects to model real world or imagined situations, we typically restrict the possible states of the objects to only those that are relevant to our models. [www3]

2.3.3 Properties and Methods

Properties in a class are used to present the structure of the objects: their components and the information or data contained therein shown in figure 2.1. 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. 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. [www6]

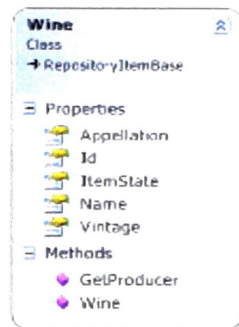


Figure 2.1 : Properties & Methods in a class

2.3.4 Inheritance

Different kinds of objects often have a certain amount in common with each other. Mountain bikes, road bikes, and tandem bikes, for example, all share the characteristics of bicycles (current speed, current pedal cadence, current gear). Yet each also defines additional features that make them different: tandem bicycles have two seats and two sets of handlebars; road bikes have drop handlebars; some mountain bikes have an additional chain ring, giving them a lower gear ratio.

Object-oriented programming allows classes to inherit commonly used state and behavior from other classes. In this example, Bicycle now becomes the superclass of MountainBike, RoadBike, and TandemBike. In the Java programming language, each class is allowed to have one direct superclass, and each superclass has the potential for an unlimited number of subclasses: [www3].

Figure 2.2 give a sample inheritance case.

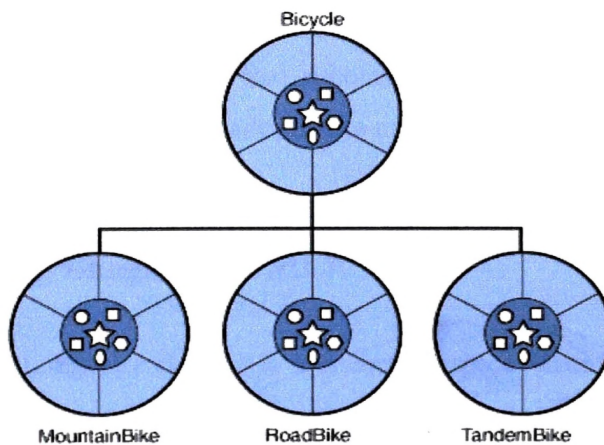


Figure 2.2 : Inheritance

2.3.5 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 2-4 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.

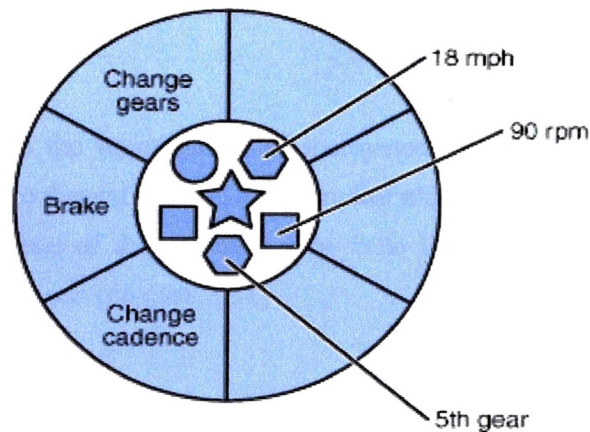


Figure 2.3 : Encapsulation

2.3.6 Polymorphism

Polymorphism indicates the meaning of “many form.” In object-oriented design, polymorphism present a method can has many definitions (forms). 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 (namely to redefine the methods in parent class). [www3]

2.4 Unified Modeling Language (UML)

The Unified Modeling Language (UML) is a general purpose visual modeling language that is used to specify , visualize , construct and document the artifacts of a software system. It captures decisions and understanding about system that must be constructed. It is used to understand, design , browse, configure, maintain and control information about such systems. It is intended

for use with all development methods, life cycle stages, application domain and media. The modeling language is intended to unify past experience about modeling techniques and to incorporate current software best practices into a standard approach. UML includes semantic concepts, notations and guidelines. It has static, dynamic environmental and organizational parts. It is intended to be supported by interactive visual modeling tools that have code generators and report writers. The UML specification does not define a standard process but it is intended to be useful with an iterative development process. [B1]

2.5 Use Case Diagrams

The use case view models the functionality of the system as perceived by outside users called 'Actor's. A use case is a coherent unit of functionality expressed as a transaction among actors and the system. The purpose of the use case view is to list the actors and use cases and show which actors participate in each use case. [B1]

2.5.1 Elements of Use Case Diagram

The use case view captures the behavior of the system, subsystem or class as it appears to an outside user .It partitions the system functionality into transactions meaningful to actor's idealized users of a system. The pieces of interactive functionality are called use cases. A use case described an interaction with actors as a sequence of messages between the system and one more actors.

- **Actor**

An actor is an idealization of an external person, process or thing interacting with a system, subsystem or class. An actor characterizes the interactions that outside users may have with the system. At run time, one physical user may be bound to multiple actors within the system. Different users may be bound to the same actor and therefore represent multiple instances of the same actor definition.

Each actor participate in one or more use cases. It interacts with the use case by exchanging messages. The internal implementation of an actor is not relevant in the use case. An actor may be characterized sufficiently by a set of attributes that define its state.

Actors may be defined in generalization hierarchies, in which is an abstract actor description is shared and augmented by one or more specific actor descriptions. An actor may be a human,

another computer system or some executable process. An actor is drawn as a small stick person with the name below it. [B1]

- **Use Case**

A use case is a coherent unit of externally visible functionality provided by system unit and expressed by sequence of messages exchanged by the system unit and one or more actors of the system unit. The purpose of a use case is to define a piece of coherent behavior without revealing the internal structure of the system.

The definition of use case includes all the behavior it entails the main sequences, different variations on normal behavior, and all the exceptional conditions that can occur with such behavior, together with the desired response. From the user's point of view, these may be abnormal situations. From the system's point of view, they are additional variations that must be described and handled.

In the model execution of each use case is independent from the others, although an implementation of the use case may create implicit dependencies among them due to shared objects. Each use case represents an orthogonal piece of functionality whose execution can be mixed with the execution of other use cases. A use case is drawn as an ellipse with its name inside or below it. It is connected by solid lines to actors that communicate with it. [B1]

2.6 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. Class diagrams describe three different perspectives when designing a system, conceptual, specification, and implementation. These perspectives become evident as the diagram is created and help solidify the design. This example is only meant as an introduction to the UML and class diagrams. If you would like to learn more see the Resources page for more detailed resources on UML.

Classes are composed of three things: a name, attributes, and operations. Below is an example of a class. Figure 2.4 given an example of class symbol.

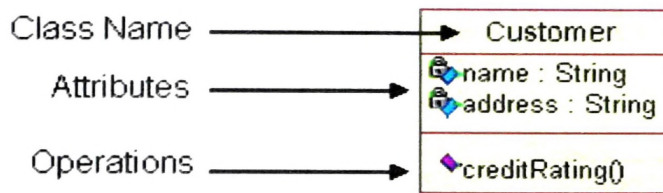


Figure 2.4 : Class Symbol

Class diagrams also display relationships such as containment, inheritance, associations and others. Class diagrams are some of the most difficult UML diagrams to draw. To draw detailed and useful diagrams a person would have to study UML and Object Oriented principles for a long time. Therefore, this page will give a very high level overview of the process.

Before drawing a class diagram consider the three different perspectives of the system the diagram will present; conceptual, specification, and implementation. Try not to focus on one perspective and try seeing how they all work together.

When designing classes consider what attributes and operations it will have. Then try to determine how instances of the classes will interact with each other. These are the very first steps of many in developing a class diagram. However, using just these basic techniques one can develop a complete view of the software system. [www7]

2.7 Sequence Diagrams

A sequence diagram is made up of objects and messages. Objects are represented exactly how they have been represented in all UML diagrams—as rectangles with the underlined class name within the rectangle. 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 two-dimensional in nature. On the horizontal axis, it shows the life of the object that it represents, while on the vertical axis, it shows the sequence of the creation or invocation of these objects. Because it uses class name and object name references, the Sequence diagram is very useful in elaborating and detailing the dynamic design and the sequence and

origin of invocation of objects. Hence, the Sequence diagram is one of the most widely used dynamic diagrams in UML. Table 2.2 given the elements of a sequence diagram.


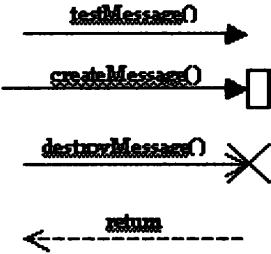
<i>Element and its description</i>	<i>Symbol</i>
<p>Object: The primary element involved in a sequence diagram is an Object—an instance of a class. A Sequence diagram consists of sequences of interaction among different objects over a period of time. An object is represented by a named rectangle. The name to the left of the ":" is the object name and to its right is the class name.</p>	
<p>Message: The interaction between different objects in a sequence diagram is represented as messages. A message is denoted by a directed arrow. Depending on the type of message, the notation differs. In a Sequence diagram, you can represent simple messages, special messages to create or destroy objects, and message responses.</p>	

Table2.2 : Elements of a sequence diagram

2.7.1 Entity Classes

An entity class models information and associated behavior that is generally long lived. This type of class may reflect a real-world entity or it may be needed to perform tasks internal to the system. They are typically independent of their surroundings; that is, they are not sensitive to how the surroundings communicate with the system. Many times, they are application independent, meaning that they may be used in more than one application.

The first step is to examine the responsibilities documented in the flow of events for the identified use cases (i.e., what the system must do). Entity classes typically are classes that are needed by the system to accomplish some responsibility. The nouns and noun phrases used to describe the responsibility may be a good starting point. The initial list of nouns must be filtered because it could contain nouns that are outside the problem domain, nouns that are just language expressions, nouns that are redundant, and nouns that are descriptions of class structures.

Entity classes typically are found early in the Elaboration Phase. They are often called "domain" classes since they usually deal with abstractions of real-world entities.

2.7.2 Boundary Classes

Boundary classes handle the communication between the system surroundings and the inside of the system. They can provide the interface to a user or another system (i.e., the interface to an actor). They constitute the surroundings-dependent part of the system. Boundary classes are used to model the system interfaces.

Each physical actor/scenario pair is examined to discover boundary classes. The boundary classes chosen in the Elaboration Phase of development are typically at a high level. For example, you may model a window but not model each of its dialogue boxes and buttons. At this point, you are documenting the user interface requirements, not implementing the interface.

User interface requirements tend to be very vague—the terms user-friendly and flexible seem to be used a lot. But user-friendly means different things to different people. This is where prototyping and storyboarding techniques can be very useful. The customer can get the "look and feel" of the system and truly capture what user-friendly means. The what is then captured as the structure and behavior of the boundary class. During design these classes are refined to take into consideration the chosen user interface mechanisms—how they are to be implemented.

Boundary classes are also added to facilitate communication with other systems. During design, these classes are refined to take into consideration the chosen communication protocols.

2.7.3 Control Classes

Control classes model sequencing behavior specific to one or more use cases. Control classes coordinate the events needed to realize the behavior specified in the use case. You can think of a control class as "running" or "executing" the use case—they represent the dynamics of the use case. Control classes typically are application-dependent classes.

In the early stages of the Elaboration Phase, a control class is added for each actor/use case pair. The control class is responsible for the flow of events in the use case. The use of control classes is very subjective. [B2]

2.8 Virclipse IDE

In this project I have used Eclipse Integrated Development Environment as the coding tool. We have used Eclipse to code java classes, compile them and run those classes. Virclipse IDE is of the updated and a modified version of a Eclipse IDE. Therefore for this project a customized version of eclipse called Virclipse is used. Virclipse is a customized in Virtusa as a productivity improvement tool for software development with some additional Eclipse plug-ins.

2.9 Gimp tool

The **GIMP (GNU Image Manipulation Program)**, is a raster graphics editor used to process digital graphics and photographs. 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. Figure 2.5 given a sample scene of GIMP.

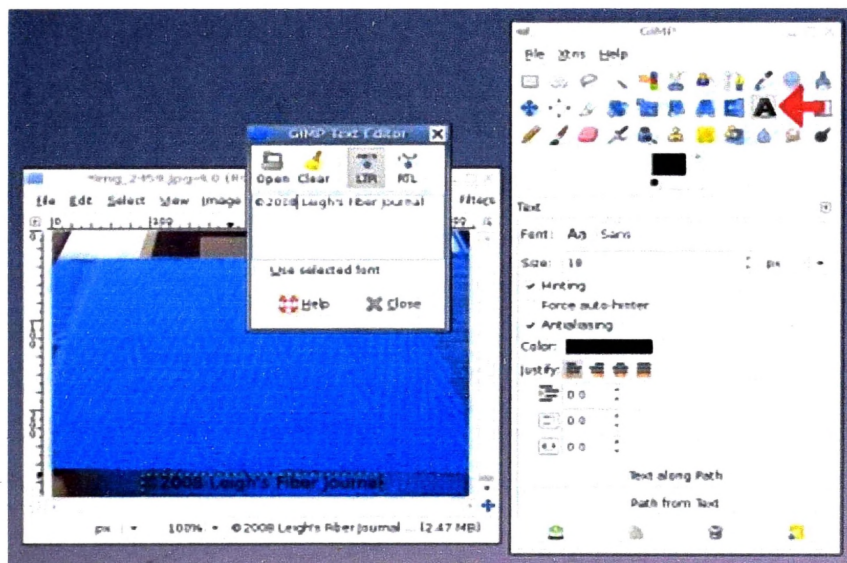


Figure 2.5 : GIMP tool

2.10 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 games 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: RealmForge, Truevision3D, OGRE, Crystal Space and Genesis3D. Table 2.3 given the overview of game engines.

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	-
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	3D via Genesis3D (DirectX)	Yes	Yes

Table2.3 : Game Engine Overview

2.10.1 Freeware 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.

Out of all freeware engines for our development we were used JMonkeyEngine , Scirra Construct and reality factor.

2.11 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. 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 (mistake), which results in a defect (fault, bug) 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.

2.11.1 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. 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.[www13]

2.12 MySQL

MySQL is the world's most popular open source database software, with over 100 million copies of its software downloaded or distributed throughout its history. With its superior speed, reliability, and ease of use, MySQL has become the preferred choice for Web, Web 2.0, SaaS, ISV, Telecom companies and forward-thinking corporate IT Managers because it eliminates the major problems associated with downtime, maintenance and administration for modern, online applications.

Many of the world's largest and fastest-growing organizations use MySQL to save time and money powering their high-volume Web sites, critical business systems, and packaged software including industry leaders such as Yahoo!, Alcatel-Lucent, Google, Nokia, YouTube, Wikipedia, and Booking.com. The flagship MySQL offering is MySQL Enterprise, a comprehensive set of production-tested software, proactive monitoring tools, and premium support services available in an affordable annual subscription.

MySQL is a key part of LAMP (Linux, Apache, MySQL, PHP / Perl / Python), the fast-growing open source enterprise software stack. More and more companies are using LAMP as an alternative to expensive proprietary software stacks because of its lower cost and freedom from platform lock-in. The MySQL database is owned, developed and supported by Sun Microsystems, one of the world's largest contributors to open source software. MySQL was originally founded and developed in Sweden by two Swedes and a Finn: David Axmark, Allan Larsson and Michael "Monty" Widenius, who had worked together since the 1980's.

Advantages than other database:

- The best and the most-used database in the world for online applications
- Available and affordable for all
- Easy to use
- Continuously improved while remaining fast, secure and reliable
- Fun to use and improve
- Free from bugs

2.13 JasperReports

JasperReports is an open source Java reporting tool that can write to screen, to a printer or into PDF, HTML, Microsoft Excel, RTF, ODT, Comma-separated values and XML files.

It can be used in Java-enabled applications, including Java EE or Web applications, to generate dynamic content. It reads its instructions from an XML or .jasper file.

JasperReports is an open source reporting library that can be embedded into any Java application.

[www5]

Features include:

- The engine allows report definitions to include charts, with the rendering provided by the JFreeChart library which supports many chart layouts, such as Pie, Bar, Stacked Bar, Line, Area, Scatter Plot, Bubble, and Time series.
- Scriptlets may accompany the report definition, which the report definition can invoke at any point to perform additional processing. The scriptlet is built using Java.
- can be invoked before or after stages of the report generation, such as Report, Page, Column or Group.
- Sub-reports

There are many tools providing JasperReport capabilities IReport is one of them.

2.14 IReport

IReport is a program that helps users and developers that use the JasperReports library to visually design reports. Through a rich and very simple to use GUI, iReport provides all the most important functions to create nice reports in little time. [www5]

Features of iReport:

- 98% of JasperReports 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

CHAPTER 03

TECHNOLOGICAL DEVELOPMENT

3.1 Introduction to Software Development Methodology

3.1.1 Software Development Process

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 and each describes approaches to a variety of tasks or activities that take place during the process.

A decades-long goal has been to find repeatable, predictable processes or methodologies that improve productivity and quality. Some expertise try to systematize or formalize the seemingly unruly task of writing software. Others are focusing on applying project management techniques to writing software. Problems arising during software projects such as delivering late and over budget can easily be eliminated through application of project management techniques. Project management has been a challenge and it urge for effective management since it has been problematic to meet the expectations of software projects in terms of functionality, cost, or delivery schedule.. There are several models available for development of effective project management. Those processes can be identified as Waterfall process, Iterative process, Prototyping, Agile development, Spiral etc. [B1]

3.1.2 Agile Development

Agile software development is a group of software development methodologies that are based on similar principles. Agile methodologies generally promote a project management process that encourages frequent inspection and adaptation, a leadership philosophy that encourages teamwork, self-organization and accountability, a set of engineering best practices that allow for rapid delivery of high-quality software, and a business approach that aligns development with customer needs and company goals.

There are many specific agile development methods. Most promote development iterations, teamwork, collaboration, and process adaptability throughout the life-cycle of the project. Agile chooses to do things in small increments with minimal planning, rather than long-term planning. Iterations are short time frames (known as 'timeboxes') which typically last from one to four weeks. Each iteration is worked 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 stakeholders. This helps to minimize the overall risk, and allows the project to adapt to changes quickly. Documentation is produced as required by stakeholders. Iteration may not add enough functionality to warrant releasing the product to market, but the goal is to have an available release (with minimal bugs) at the end of each iteration. Multiple iterations may be required to release a product or new features.

The following are other features that describe software development projects that use agile methodologies: Figure 3.1 shown the development process of agile development.

- 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

[www15]

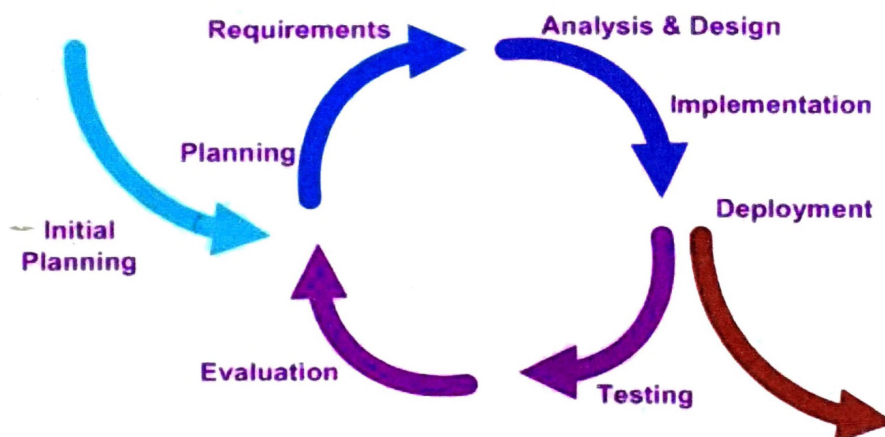


Figure 3.1: Agile Development Process

3.2 Game Development Life Cycle

Basically, there are five major phases in our game development life cycle. Out of all five phases, the Game Concept is considered as the most important phase. Therefore, when figuring out the most suitable Game concept for our project, initially focusing on requirements and ideas regarding Game Concepts found that they believe all Game Concepts should be based on mathematics as well as on language. After several team discussions, suitable Game Concept for our project was developed.

In Pre-production phase, we carried out all the preliminary design work which includes designing use case diagrams, class diagrams and sequence diagrams and selecting the required software. During next phase, which was the production phase, all the design concepts were implemented by carrying out all code level functionalities.

In testing and release phase we performed all kinds of testing such as unit testing as well as system testing to ensure the accuracy product's functionality. Figure 3.2 shown the game development life cycle.

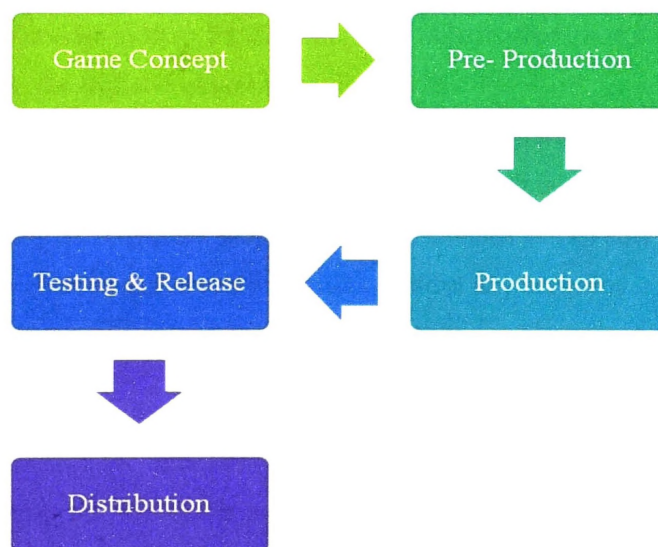


Figure 3.2: Game Development Life Cycle

3.3 Game Engines

3.3.1 Scirra Construct (Rapid Game Authoring System)

Construct is free powerful and easy to use development software for both DirectX 9-based games and applications. 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.

Features of Scirra Construct

- 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 3.3 show a demo game made up with Scirra construct.

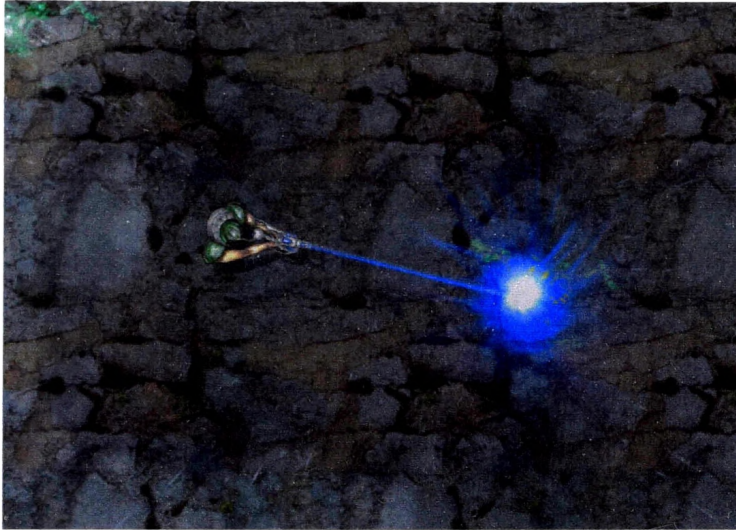


Figure 3.3: A Demo Game created with Scirra Construct

3.3.2 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. 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.

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*.

JOGL

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

Features of JMonkey Engine

- jME is a scenegraph based architecture. The scenegraph 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. [www9]

Figure 3.4 illustrates a demo game using JMonkey engine.

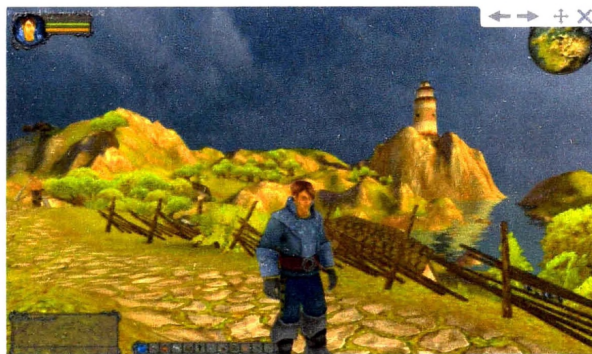


Figure 3.4 : A Demo game created with Java Monkey Engine

3.3.3 Reality Factor

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.

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.

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 3.5 shown demo game created with reality factor.

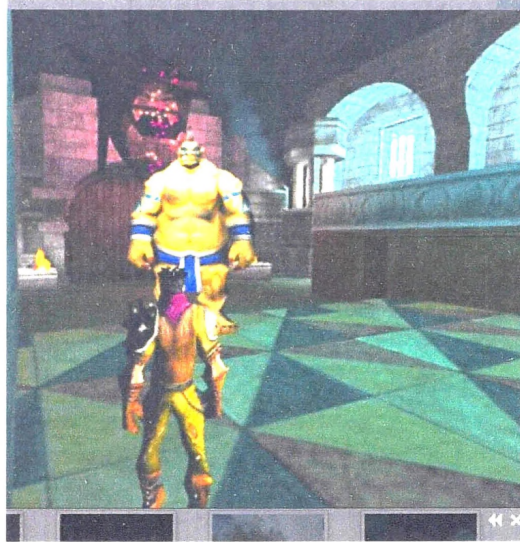


Figure 3.5 : Demo game created with Reality Factor

CHAPTER 04

DESIGNING DEVELOPMENT

4.1 Use Diagram

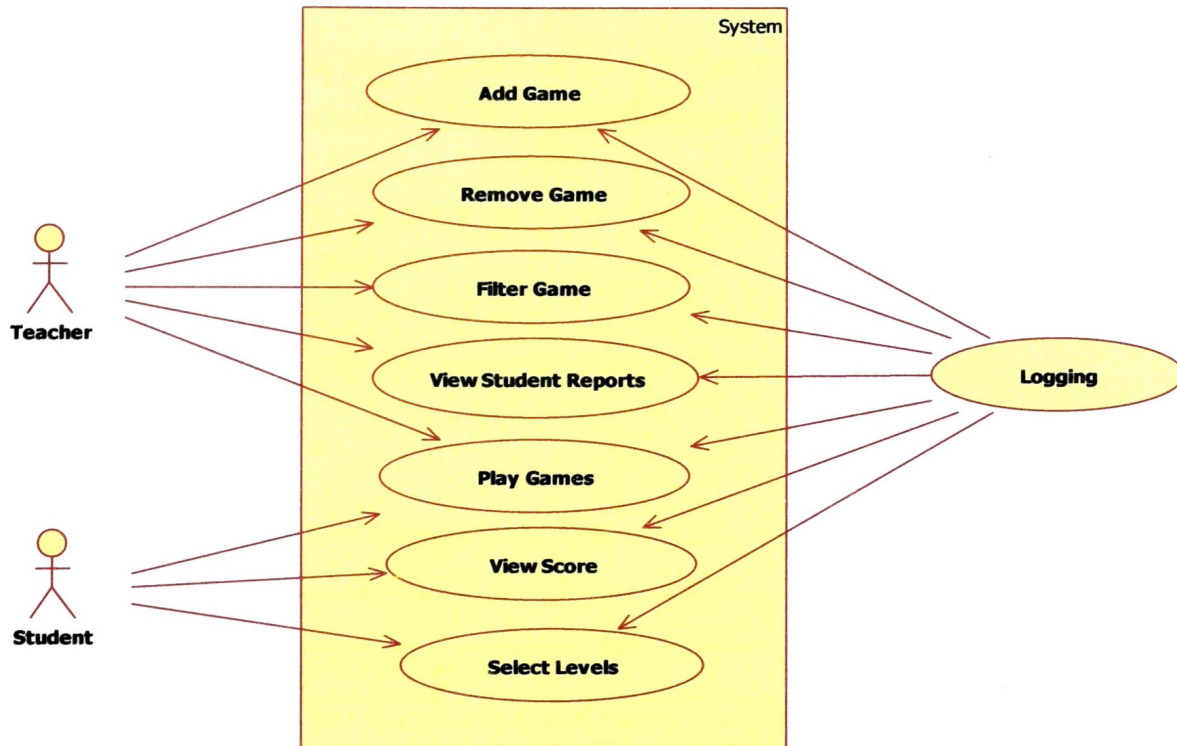


Figure 4.1: Use Case Diagram for our system

- **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

Above Figure 4.1 shown a use case diagram for teacher and student actors.

4.1.1 Use Case Descriptions

Table 4.1 – 4.5 shows the use case description for design use case diagram

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

Table4.1 : Game adding to the Launch pad

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 4.) Selected games removed from the system.
Alternate Flows	Should have games in between 1 and 5
Post Condition	Games remove from the launcher pad

Table4.2 : Game removing from the Launch pad

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.

Table2.3: Game filter from the Launch pad

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.

Table4.4: View Reports

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.

Table4.5: Game Launch

4.2 Class Diagram

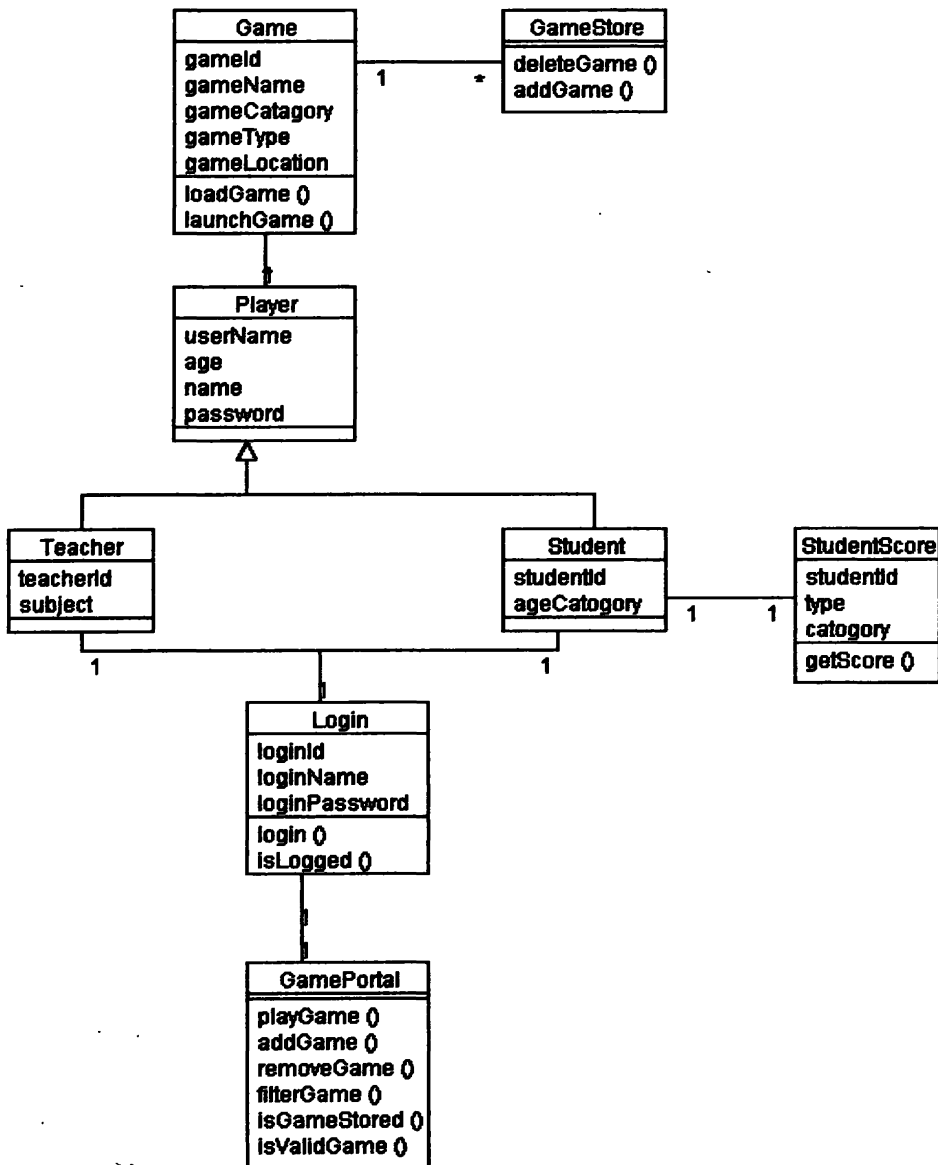


Figure 4.2: Class Diagram for the system

Figure 4.2 indicates the class diagram for the edutainment launch pad and its functionalities. Player class specialized into the teacher and student, which are the main entity classes on our system. The diagram shows corresponding methods and attributes for each and every class.

Out of the given class structure, StudentScore and the GameStore are the boundary classes and GamePortal , Login Game are the control classes. According to above structure, most of the functionalities depend upon Game and GamePortal classes.

4.3 Sequence Diagrams

According to the use case diagram ,

Figure 4.3 - 4.9 and tables of 4.6 – 4.10 show the sequence diagram of the use cases.

4.3.1 Add Game to the Launch Pad

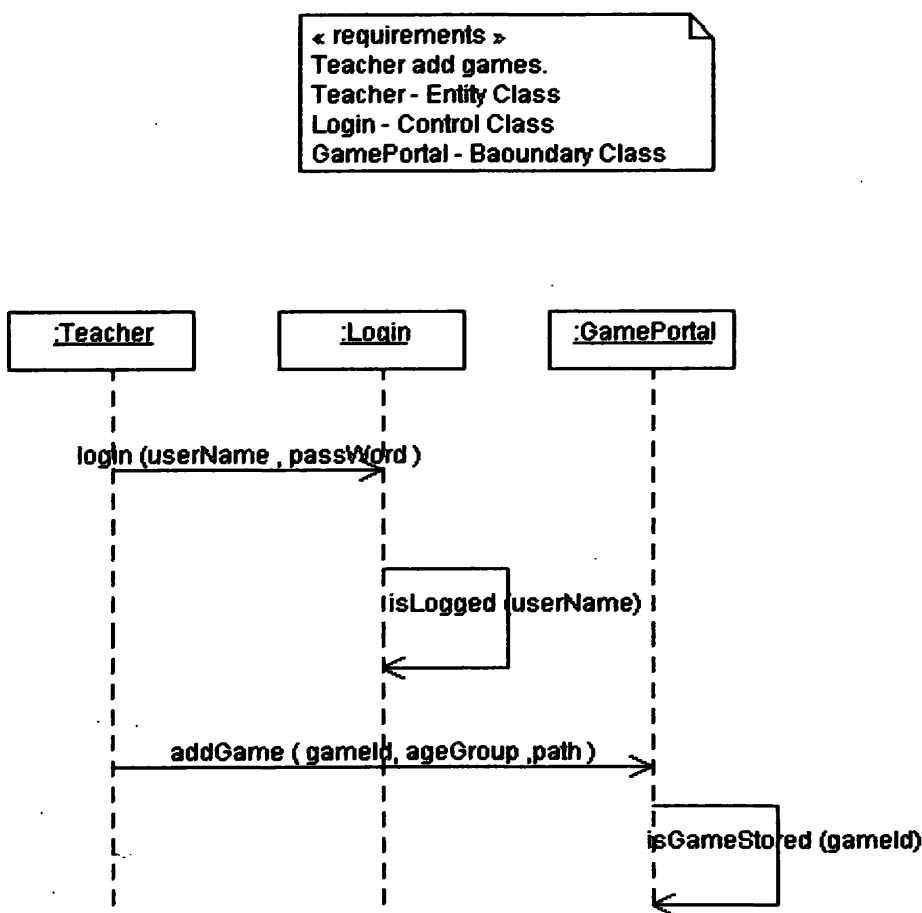


Figure 4.3 : Add game to the Launch Pad

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. addGame and isGameStored methods are used to implement main functionalities.

Table4.6 : Add game to the launch pad description

4.3.2 Remove Game from the Launch Pad

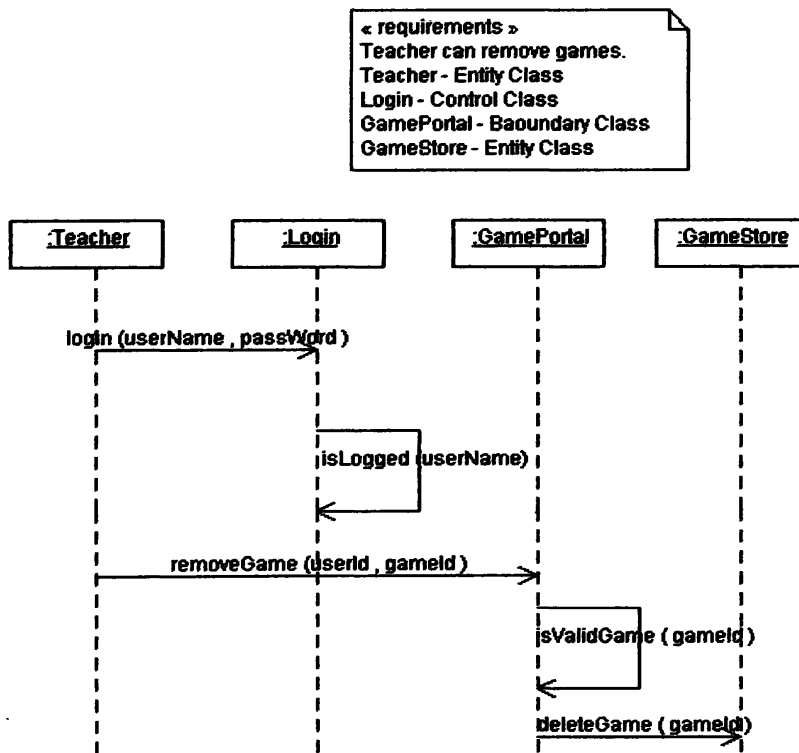


Figure 4.4 : Remove game from the launch pad

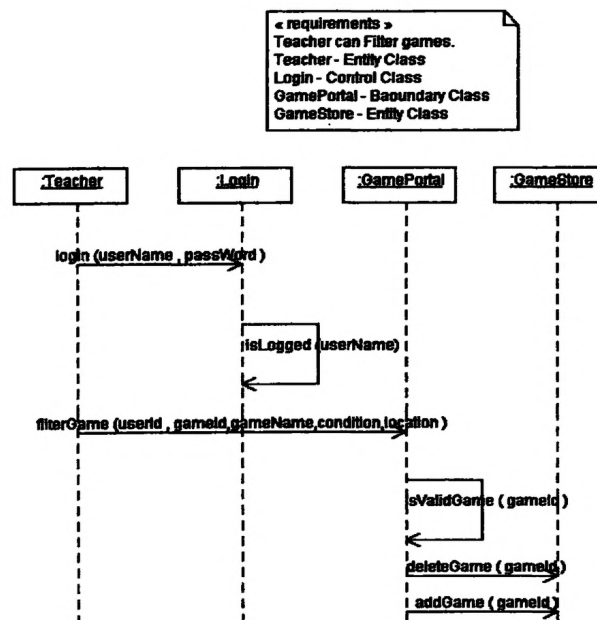
Involved Classes	<ul style="list-style-type: none"> ▪ Teacher ▪ Login ▪ GamePortal ▪ GameStore
Pre Condition	To remove a game Teacher must first log into the system using username and password.
Description	This scenario explains about the class behaviors when Teacher remove unwanted games from the Launcher. The added games were store in the GameStore class. In that case removeGames and deletedGame methods take all the main functionalities.

Table4.7 : Remove Game From the launch pad description

As above diagram same scenario acts on logging to the system.

Then call the removeGame method with the corresponding parameters to remove game. Like wise adding in GamePortal class validate the game before removing. Then delete the game from GameStore class.

4.3.3 Filter games from the Game Launcher

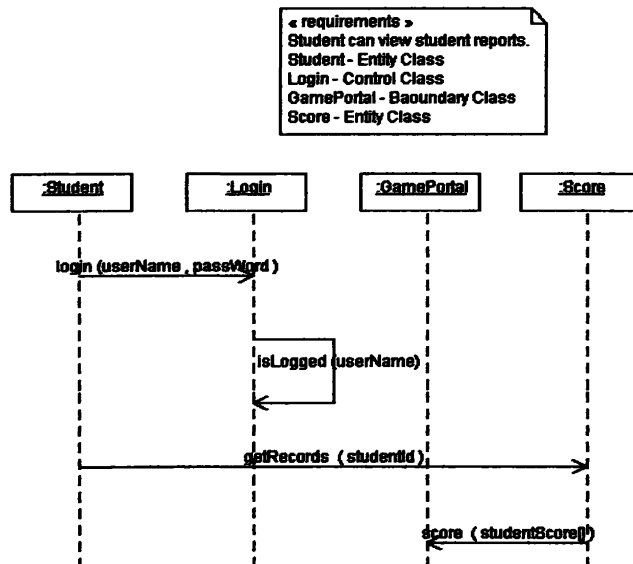


1 : Filter games from the launch pad

Involved Classes	<ul style="list-style-type: none"> ▪ Teacher ▪ Login ▪ GamePortal ▪ GameStore
Pre Condition	To filter a game Teacher must first log into the system using username and password.
Description	This scenario explains about the class behaviors when Teacher filter the games according to the types of student categories. To make that functionality strong filterGame deletedGame and addGame methods are important.

Table4.8 : Filter games from the launch pad description

4.3.4 Student and Teacher can view Reports



2 : Student Can View Reports

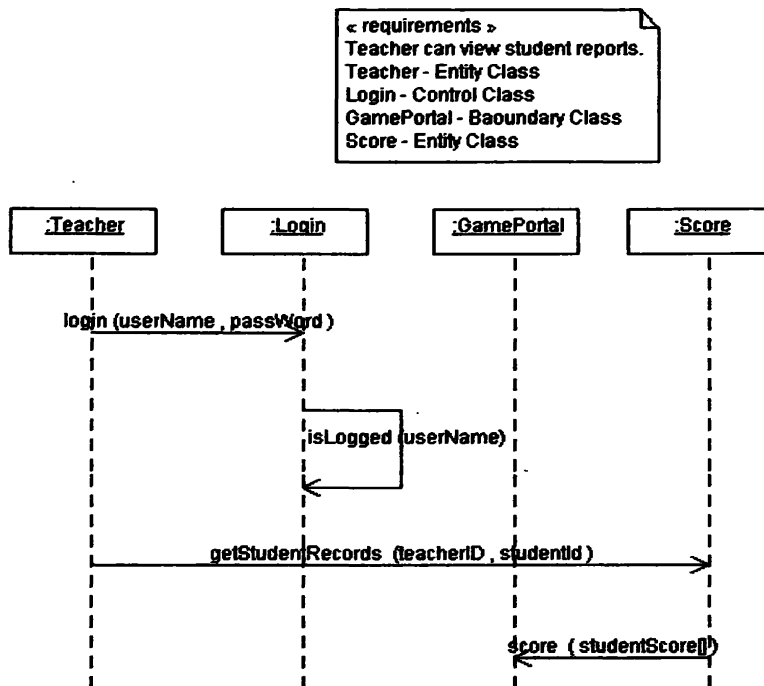


Figure 4.7: Teacher can view reports

Involved Classes	<ul style="list-style-type: none"> ▪ Student/Teacher ▪ Login ▪ GamePortal ▪ Score
Pre Condition	To view reports both Student and Teacher must first log into the system using username and password.
Description	This scenario explains about the class behaviors when Student/Teacher view the reports. getStudentRecords and score are the respective methods important in that scenario.

Table4.9 : Student/Teacher can view scores description

4.3.5 Launch Game

- Teacher can launch a game

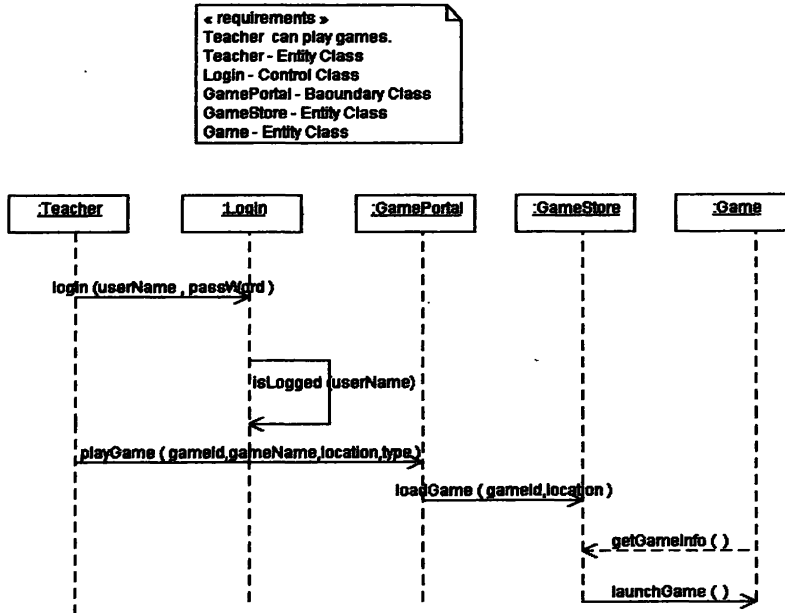


Figure 4.8 : Teacher can launch games

- Student can launch game

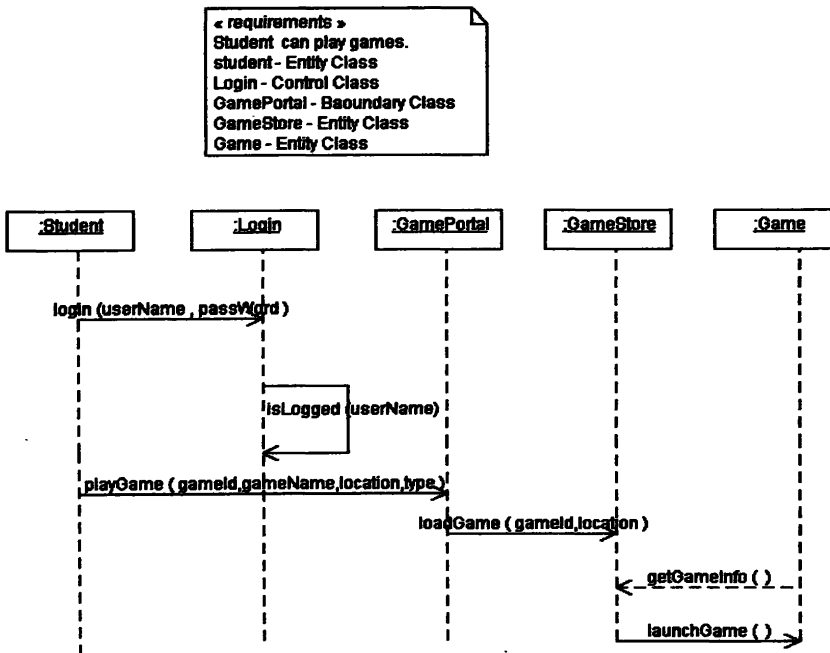


Figure 4.9 : Student can launch games

Involved Classes	<ul style="list-style-type: none"> ▪ Teacher/ Student ▪ Login ▪ GamePortal ▪ GameStore ▪ Game
Pre Condition	To launch a game both Teacher and Student must first log into the system using username and password.
Description	This scenario explains about the class behaviors when Teacher /Student launch games. playGame method takes all the important functionality during the game launching. Rest of the methods like loadGame and launchGame are supportive methods on to this method.

Table4.10 : Teacher and Student can launch games

CHAPTER 05

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**
Introduction implies the objective of the selected game. Out of concepts what are the sub areas going to touch from this game. As an example under the mathematics concept we can try out the identification of sorting ,arithmetic operations etc.
- **Background (optional)**
Background implies how the game background looks like and how those figures and pictures are arranged.
- **Description**
Description gives the clear idea about the game instructions.
- **Platform(s)**
Platform means game design platform. It can be either Windows platform or Linux platform.
- **Concept art (optional)**

This chapter discusses some of the game concepts that we have used in this project. Table 5.1 figure shows game concept for ascending train game.

5.1 Ascending Train (or Descending Train)

Table 5.1 figure shows game concept for odd even number separator game.

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.
Platform	Windows / Linux

Table5.1: Game Concept for the Ascending Train game

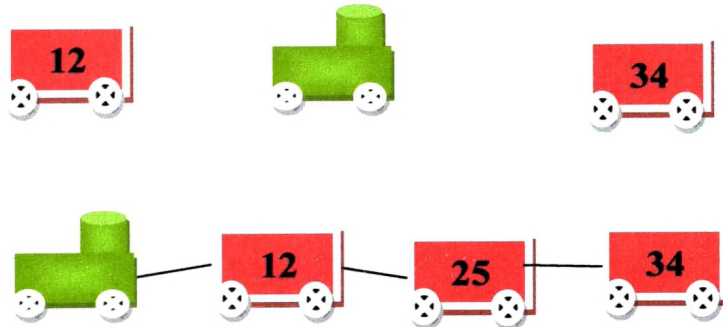


Figure5-1 : Overview of the Ascending Train Game

5.2 Odd/Even number Separator

Table 5.2 shows the game concept and the Figure 5.2 corresponds the overview of the Odd/Even separator game.

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.
Platform	Windows / Linux

Table5-2 : Game Concept for the ODD/EVEN number separator game

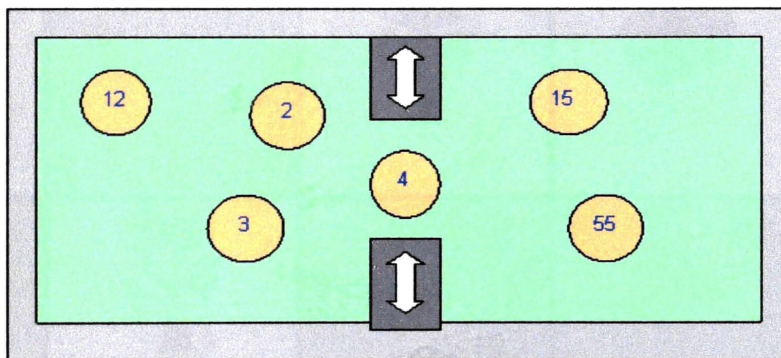


Figure5-2 : Overview of the ODD/EVEN number separator Game

5.3 Distance and Directions (Treasure Hunt)

Table 5.3 lists the game concept of the distance and direction game and the figure 5.3 shows the overview of that game.

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.
Platform	Windows / Linux

Table5.33 : Game Concept for the Distance and Directions game

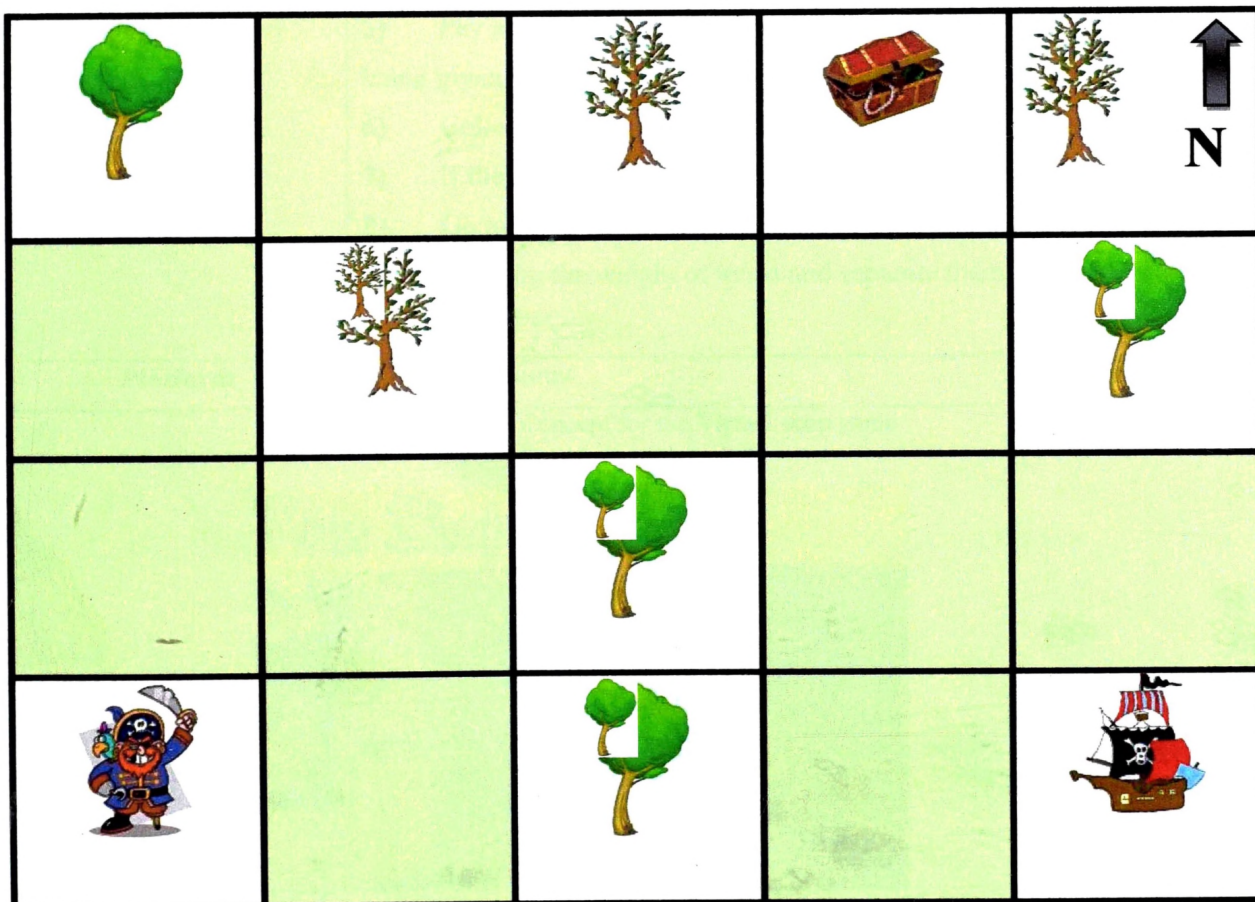


Figure5.3 : Overview of the Distance and directions Game

5.4 Virtual Shop

Table 5.4 shows the game concept and the figure 5.4 corresponds the overview of the game.

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. 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

Table5.4 : Game Concept for the Virtual shop game

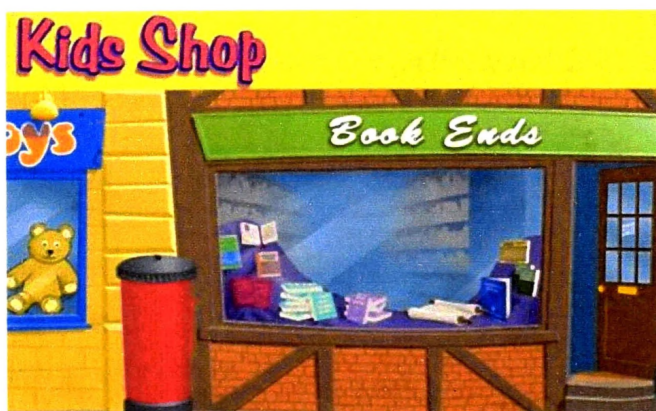


Figure 5.4 : Overview of the Virtual Shop Game

CHAPTER 06

DEVELOPMENT ENVIRONMENT

6.1 Development Environment

A brief introduction of the development environment is given below.

6.1.1 Hardware Environment

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

6.1.2 Software Environment

Table 6.1 given the system environment of the system

IDE	Virclipse (Customized of eclipsed)
Languages	<ul style="list-style-type: none"> ▪ Java ▪ Java 2D Graphic Package
Operating system	Windows XP/ Windows Vista / Linux
Third Party Components and Tools	<ul style="list-style-type: none"> ▪ Scirra Construct ▪ Java Monkey Engine ▪ Napkinlaf freely available jar file
Enhancing Tools	<ul style="list-style-type: none"> ▪ Blender ▪ Gimp
Database	Text Pads

Table6.1 : Development Environment

6.2 Application Programming Interface (API) used for implementation

- Java 1.5 API
- Java 2D API
- Scirra Construct Tutorial
- JMonkey Engine API
- JMonkey Engine User Guide

6.3 Integrating Environment

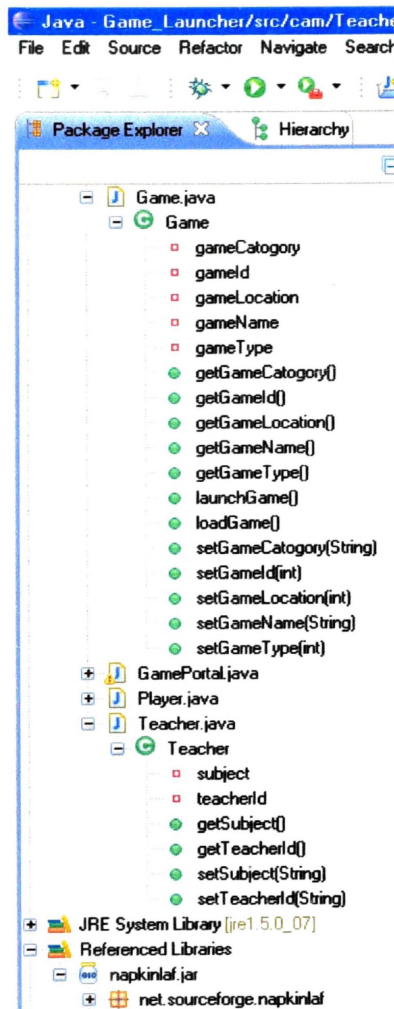


Figure6.1 : Class Hierarchy

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 can identify the used classes according to the class diagrams and sequence diagrams. For the reference library, we have used napkinlaf jar file as the selected jar file is open source.

6.4 Reporting

There are different team members for different functionalities. Therefore regarding reporting there is a separate team. They all conducted this by using of Jasper Reports. The selected JasperReports are designed by the iReport. Once again according to the Virtusa policies we cannot figure out any of the reports. All reports are up to the Virtusa standards. Figure 6.2 indicates the iReport environment.

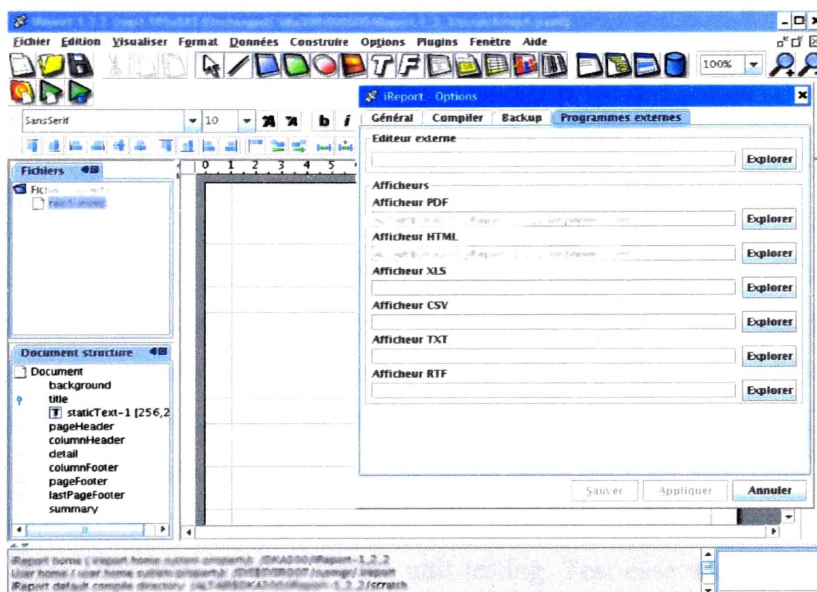


Figure6.2 : iReport

CHAPTER 07

TESTING & DEPLOYMENT

7.1 Testing Strategy

According to typical software development process there are mainly two teams. One for development and other for testing. Testing was found to be a challenging task and those are the people looking at major aspect of quality product. Testing is done by several phases. Considering the whole testing process, the development team conducted the unit testing in work it is supposed to write test case related with the system. After integrating the system, the integration testing was conducted.

7.2 Unit Testing

Unit testing is a development procedure where programmers create tests as they develop software. The tests are simple short tests that test functionality of a particular unit or module of their code, such as a class or function or a procedure used to validate that individual units of source code are working properly. The goal of unit testing is to isolate each part of the program and show that the individual parts are correct.

All the public methods were tested by negative and positive manner and finally the code coverage of more than 80% may covered by the unit testing. Test case are involving with this process.

7.2.1 Test Cases

A test case is a detailed procedure that fully tests a feature or an aspect of a feature. Whereas the test plan describes what to test, a test case describes how to perform a particular test. Developers need to develop a test case for each test listed in the test plan.

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.

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.

7.2.2 Test Data

According to the test cases we have to fill all the procedures in following manner.

Test data used to check whether the system is running on corrective manner. With the following real time screen shots implement how the system is working on and for each of the time how can we write the test cases.

Implementation of Test Case ID of 'SPF_P1': Teacher trying to add a game to the launch pad

- Figure 7.1 shows a screen before adding the game into the launch pad

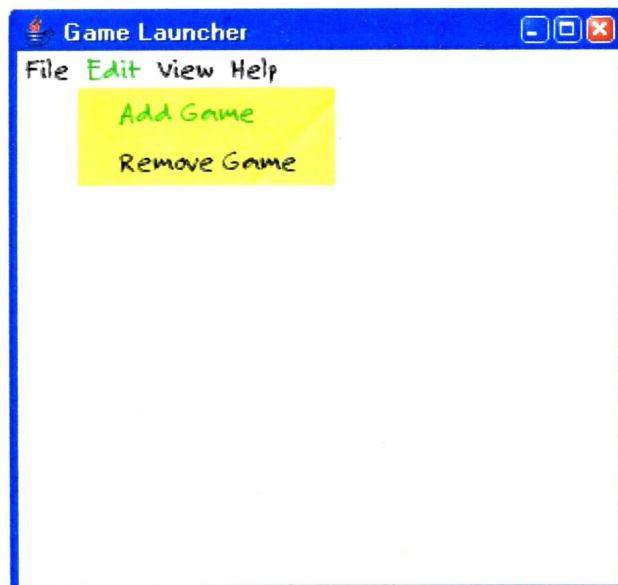


Figure7.1 : Before implement add game test case

- Figure 7.2 shows the screen after adding the game into the launch pad



Figure7.2 : After implement add game test case

- Table 7.1 illustrates corresponding Test Case for the add a game to the launch pad

Test Case ID	Test Case Description	Prerequisite	Test Procedure	Input Data	Expected Result	Actual Result	Test Result
SPF_P1	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

Table7.1 : Sample Test Cases

Implementation of Test Case ID of 'SPL_P1': Student logging to the system

- Figure 7.3 shows screen before login to the system



Figure7.3 : Before implement login test case

- Figure 7.4 shows test case after logged to the system

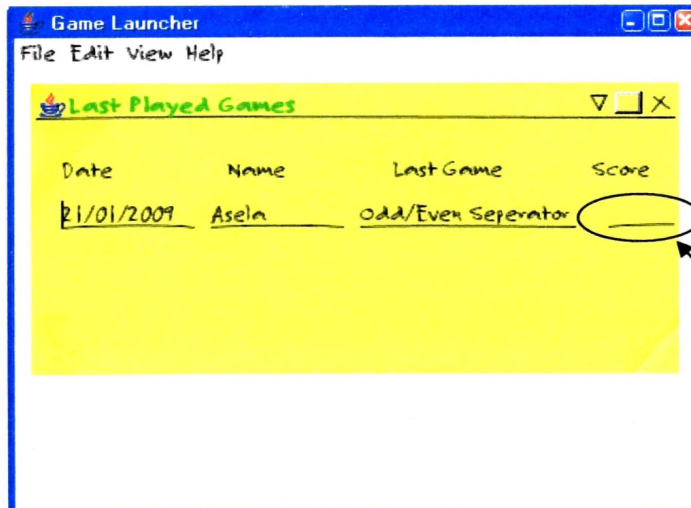


Figure7.4 : After implement login test case

- Table 7.2 illustrates corresponding Test Case for the student logging to the system

SPL_P1	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 previous game details	Logged to the launch pad student's details may not appear	Selected test case is success. Not completely executed.
--------	-------------------------------	-------------------------	---	--	--	---	---

Table7.2: Sample Test Case II

7.3 System Testing

System testing 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.

Thorough testing was done on following categories.

- Performance
- Security features
- Stress testing
- User acceptance

That is because we need to verify as well as the functional requirements of all the non functional requirements are satisfied in order to fulfill all the project requirements.

7.4 Deploy Environment

After integrating the system it is finally zipped into a jar file. Installing the system in client's machine involves copying and pasting the jar into a specific location.

To run the system only users have to double click the jar.

7.4.1 Hardware Requirements

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

7.4.2 Software Requirements

Table 7.3 given the software requirement of the system

Operating System	Windows XP
Packaging Tool	Jar
Third Party Components and Tools	Not used
Setup Machine	Machine should install MySQL before use the game launcher.

Table7.3: Software Requirement of the Deployment Environment

CHAPTER 08

CONCLUSION

The aim of this project was to implement a launch pad for edutainment software suit for primary schools students of Sri Lankan schools using free and open source software. Reusability and extensibility issues could be achieved to evolve the system.

The clients' comments on the prototype evaluation must be first fulfilled. The usability is the main issue on how comfortable potential users are of using this system. All functions they requested for main stage are working properly and user friendly. The original objectives and goals set forth in developing the system are achieved successfully.

User interfaces could adapt to the changes by modifying the program code. Although time spent on designing and implementing components was longer than traditional design and implementation, the time spent on future changes would be saved.

8.1 Future Consideration

Future consideration for this project is to enhance some of the present features and add new features for the system. Because of using open source software we can implement this edutainment launch pad for Linux machines as well. In near future, subject to the availability can implement the same system in OLPC as well.

Existing system is a menu driven system that could be enhanced to provide more user friendliness by adding some colorful themes to user interfaces.

Reporting system can be enhanced in a graphical manner instead of typical indication of score.

Extra features should be added to the system so that it can meet the client's requirements, such as using this game console for disable students.

REFERENCES

- [B1] - Rumbaugh, J. (2004) The Unified Modeling Language Reference manual, Addison Wesley Longman, Inc., 3-66
- [B2] - Quatrani, T. (2000) Visual Modeling with Rational Rose 2000 and UML, Publisher Pearson Education India, 77-85
- [B3] - Eliens, A. (2000) Principle of Object-Oriented Software Development, 2nd Edition, Pearson Education Limited 2000. 18-35
- [B4] - Sommerville. (1995). The fifth edition of Software Engineering. Addison Wesley Publishers in autumn, pp.210-400
- [B5] - Lieberman, H., Liu, H., Singh, P., Barry, B.: Beating common sense into interactive applications. Game Magazine 25(4) (2004) 63-76
- [B6] - Rollings, A. & Morris, D. (2004). Game Architecture and Design. New Riders Publish pp 120-133
- [WWW1] -The History of Java Technology or Home Page, URL:<http://www.java.com/en>
21st February 2009
- [WWW2] -Java History with Tutorial or Home Page, URL:<http://www.freejavaguide.com>
,21st February 2009
- [WWW3]-Developer Resources for Java Technology or Home Page, URL:
<http://java.sun.com> , 21st February 2009
- [WWW4] -Java News and Resources or Home Page, URL: <http://www.cafeaulait.org>,
22nd February 2009
- [WWW5] -JasperReports and iReports or Home Page, URL: <http://jasperforge.org/>, 1st
March 1 2009
- [WWW6] -Java Developer's Journal or Home Page, URL: <http://java.sys-con.com>, 22nd
February 2009
- [WWW7] -Unified Modeling Language Tutorial and Home Page, URL:
<http://atlas.kennesaw.edu>, 22nd February 2009
- [WWW8] -Java Software Developer or Home Page, URL: <http://www.developer.com/>,
22nd February 2009
- [WWW9] -Java Monkey Engine or Home Page, URL: <http://www.jmonkeyengine.com>,
2nd February 2009
- [WWW10] -Open Source Software or Home Page, URL: <http://sourceforge.net>, 18th
November 2008

INDEX

3

3D · 11, 29, 30, 31, 41, 42, 43, 44, 45
3D games · 11

A

Actor · 22
adaptation · 37
Agile software development · 37
aim · 75
API · 3, 42, 43, 68
Application Programming Interface · 3, 68

B

Boundary Classes · 2, 27

C

C# · 15
CARB · 40
class · 19
Class · 18, 23, 24
client-server · 13, 16
code architecture review board · 40
composition · 29
Computer · 11
Construct · 31, 41, 42
Control Classes · 2, 27

D

database · 34, 72
defect · 32
Deploy Environment · 3, 73
Designing · 46
Development · 7, 10, 12, 28, 37, 39, 40, 46, 68
DirectX · 30, 41

E

Edutainment · 9
Encapsulation · 20, 21
engine · 29, 41
Entity Classes · 2, 26
Event Driven · 15

F

frequent · 37

functions · 29
Future · 75

G

Game · 40
Game Concept · 40
Game Dev SIG · 7
game-concept · 62
Generic · 15
Gimp · 2, 29, 68
GIMP · 29, 68
GIP · 5, 6, 7
GNU Image Manipulation Program · 29

H

Hardware Requirements · 3, 73

I

IDE · 28, 68
IEEE · 16
Inheritance · 19
inspection · 37
instance · 18, 19, 26
Integrating · 3, 69
Integration Testing · 32, 33
IReport · 2, 35, 36
iteration · 38
Iterative process · 37

J

J2EE · 14
J2ME · 14
J2SE · 14
JasperReport · 35
JasperReports · 2, 35, 36, 70, 78
Java · 12, 13, 14, 15, 16, 17
java 2D · 14
Java Monkey Engine · 44, 68
Java Virtual Machine · 12
JDBC · 36
JMonkeyEngine · 31
JOGL · 42, 43

L

Languages · 68
life cycle · 7, 21, 40
LWJGL · 30, 42, 43

M

message · 16, 26
messages · 18, 22, 23, 25, 26
methodologies · 38
Methods · 19
models · 18, 19, 22, 26, 37
MoE · 7, 8, 10, 40
MySQL · 2, 34

N

non functional requirements · 73

O

Oak · 12
Object · 15, 16, 17, 18, 20, 24, 26
Object Orient Programming · 17
Object-oriented · 15, 18, 20
object-oriented programming · 13, 17, 18
objects · 41
OLPC · 7, 75
OOP · 17
open source · 42
OpenGL · 30, 42, 43
Operating system · 68
Overloading · 21
Overriding · 21

P

Pascal · 18
PC · 11
Performance · 73
Platform Independent · 15
Polymorphism · 21
Properties · 19
Prototyping · 37

R

radiosity · 44
Reality Factory · 31, 44
Reporting · 3, 69, 75
Requirement Specification · 33
requirements · 7, 8, 13, 14, 18, 38, 40, 73

S

Scirra Construct · 31, 41, 42, 68
security · 16, 47
Security · 16, 73
sequence diagram · 25, 26
Sequence diagram · 25, 26
Sequence Diagrams · 9
software · 9, 29, 38
Software Development Kit · 12
Software Requirements · 3, 74

System Requirement Specification · 33
System Testing · 33, 73
Systems integration testing · 34

T

Testing Levels · 2, 32
Testing Strategy · 3, 71
Third Party Components · 68
Tools · 68

U

UML · 21, 24, 25
Unified Modeling Language · 21
Unit Testing · 3, 32, 71
use case · 22, 23
Use Case · 9, 22, 23
user acceptance testing · 34

V

Virclipse · 28, 68
Virtusa · 5, 7, 8

W

Waterfall process · 37
Windows · 29, 30

X

XML · 35

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 : 

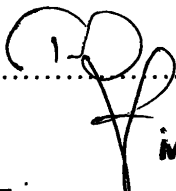
Certification of Scanning

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. NEIGHSOOREI.....

Signature : 

Date : .. 2017-09-22

Mrs. T.N. NEIGHSOOREI
(MSc, PGD, ASLA, BA)
Librarian
Sabaragamuwa University of Sri Lanka
P.O. Box 02, Belihuloya, Sri Lanka
Tel: 0094 35 2280045
Fax: 0094 45 2280045

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