A PROJECT REPORT

ON

"MONSTER ASSAULT"

Submitted in partial fulfilment of the requirements for the award of the Degree of

Bachelor of Technology

In

Computer Science & Engineering

Submitted by

MAYANK K RASTOGI	ARSHYA PANY	SHILPI KUMARI
(1005083)	(1005032)	(1005274)

Under the guidance of

MR. LALIT KUMAR VASHISHTHA

Assistant Professor

School of Computer Engineering

SCHOOL OF COMPUTER ENGINEERING

KIIT UNIVERSITY

BHUBANESWAR-24

SCHOOL OF COMPUTER ENGINEERING

KIIT UNIVERSITY

BHUBANESWAR-24

C E R T I F I C A T E

This is to Certify that the project entitled "**Monster Assault**" is being carried out by Mayank K Rastogi (1005083), Arshya Pany (1005032), Shilpi Kumari (1005274) in partial fulfilment for the award of degree of Bachelor of Technology in Computer Science & Engineering at School of Computer Engineering, KIIT University, Bhubaneswar during the academic year 2013–14 under my supervision. The matter embodied in this project is original and has not been submitted for the award of any other degree.

.

Signature of Guide

(Mr. Lalit Kumar Vashishtha)

Asst. Professor

Acknowledgement

On the very outset of this report, we would like to extend our sincere & heartfelt obligation towards Mr. Lalit Kumar Vashishtha who helped us in this endeavour. Without his active guidance, help, cooperation & encouragement, we would not have made headway in the project. We are ineffably indebted to him for conscientious guidance and encouragement to accomplish this assignment. We extend our gratitude to KIIT University for giving us this opportunity. At last but not least gratitude goes to all our friends who directly or indirectly helped us to complete this project report.

Abstract

The game 'Monster Assault' is a 2D-platformer android and desktop game written in JAVA. It is a simple game where the user is required to move the character 'Bob' using arrow keys or the on-screen touchpad. Bob can jump and fire to dodge hurdles and kill monsters which gain him points. Completing one level takes him to the next. This game uses libgdx game development framework and OpenGL for graphics.

Table of Contents

1. Introduction

1.1 Purpose

This game is a computer/mobile game with an interactive interface, in which the user explores a series of worlds (levels), collecting artifacts, and fighting monsters. The set of worlds, artifacts and monsters can be extended or replaced to give different game variations.

The goal of this project is to produce an entertaining application for the Android marketplace. They will be playable on any phone supporting the android operating system as well as on a computer.

1.2 Scope

This specification establishes the functional, performance, and development requirements for our Monster Assault game. The only input to this project is the user's attention and strategic moves; and the only output is the entertainment that is a consequence of the input.

1.3 Technologies to be used

1.3.1 Java:

Java is a computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to bytecode (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers.

1.3.2 Android:

Android is a Linux-based operating system designed primarily for touchscreen mobile devices such as smartphones and tablet computers. Android is open source and Google releases the code under the Apache License. Android is a software bunch comprising not only operating system but also middleware and key applications. Android Inc was founded in Palo Alto of California, U.S. by Andy Rubin, Rich Miner, Nick Sears and Chris White in 2003. Later Android Inc. was acquired by Google in 2005. After original release there have been number of updates in the original version of Android.

Android applications are written in java programming language. Android is available as open source for developers to develop applications which can be further used for selling in android market. There are around 200000 applications developed for android with over 3 billion+ downloads. Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model.

1.3.3 Eclipse:

Eclipse is a multi-language Integrated development environment (IDE) comprising a base workspace and an extensible plug-in system for customizing the environment. It is written mostly in Java. It can be used to develop applications in Java and, by means of various plug-ins, other programming languages including Ada, C, C++, COBOL, Fortran, Haskell, JavaScript, Lasso, Perl, PHP, Python, R, Ruby (including Ruby on Rails framework), Scala, Clojure, Groovy, Scheme, and Erlang.

The Eclipse software development kit (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.

1.3.4 Libgdx:

Libgdx is a cross-platform game and visualization development framework. It currently supports Windows, Linux, Mac OS X, Android, iOS and HTML5 as target platforms.

Libgdx allows you to write your code once and deploy it to multiple platforms without modification. Instead of waiting for your latest modifications to be deployed to your device or to be compiled to HTML5, you can benefit from an extremely fast iteration cycle by coding your application mainly in a desktop environment. You can use all the tools of the Java ecosystem to be as productive as you can be.

Libgdx lets you go as low-level as you want, giving you direct access to file systems, input devices, audio devices and OpenGL via a unified OpenGL ES 1.x and 2.0 interface.

On top of these low-level facilities, libgdx offers a powerful set of APIs that help you with common game development tasks like rendering sprites & text, building user interfaces, playing back sound effects and music streams, linear algebra and trigonometry calculations, parsing JSON and XML, and so on.

1.3.5 OpenGL ES:

OpenGL (Open Graphics Library) is a cross-language, multi-platform application programming interface (API) for rendering 2D and 3D computer graphics. The API is typically used to interact with a Graphics processing unit (GPU), to achieve hardware-accelerated rendering. OpenGL was developed by Silicon Graphics Inc. (SGI) from 1991 and released in January 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, flight simulation, and video games.

OpenGL for Embedded Systems (OpenGL ES) is a subset of the OpenGL computer graphics rendering application programming interface (API) for rendering 2D and 3D computer graphics such as those used by video games, typically hardware-accelerated using a graphics processing unit (GPU). It is designed for embedded systems like smartphones, computer tablets, video game consoles and PDAs. The API is cross-language and multi-platform.

2. Overall Description

2.1 Product Perspective

Input - Provides a unified input model and handler for all platforms. Supports keyboard, touchscreen, accelerometer and mouse where available.

Graphics - Enables the drawing of images to the screen using the hardware provided OpenGL ES implementation.

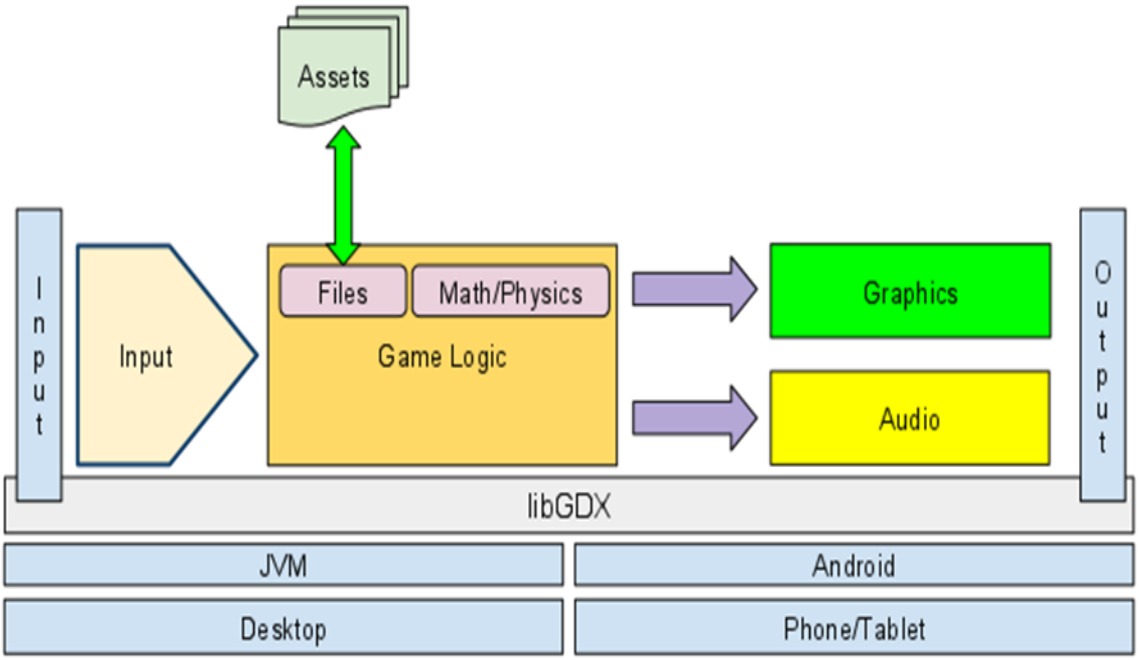
Files - Abstracts file access on all platforms by providing convenient methods for read/write operations regardless of the media.

Audio - Facilitates sound recording and playback on all platforms.

Math - Utility module providing fast math calculations geared towards game development.

Physics - Complete wrapper for Box2D physics simulation engine

.



2.2 Software Interface

Front End:

- Android application or desktop application

Back End:

- Java
- Libgdx
- OpenGL ES
- Android OS/ Any OS that runs Java

2.3 Hardware Interface

Minimum Requirements:

Computer			
	Processor	**RAM**	**Disk Space**
Windows XP / Linux	Intel Pentium III or AMD-800 MHz	256 MB	10 MB

Mobile Device			
	Processor	**RAM**	**Disk Space**
Android 2.1 (Eclair)	600 Mhz	256 MB	10 MB

Recommended Requirements:

Computer			
	Processor	**RAM**	**Disk Space**
Windows 7 / Linux	All Intel or AMD - 1 GHz	512 MB	10 MB

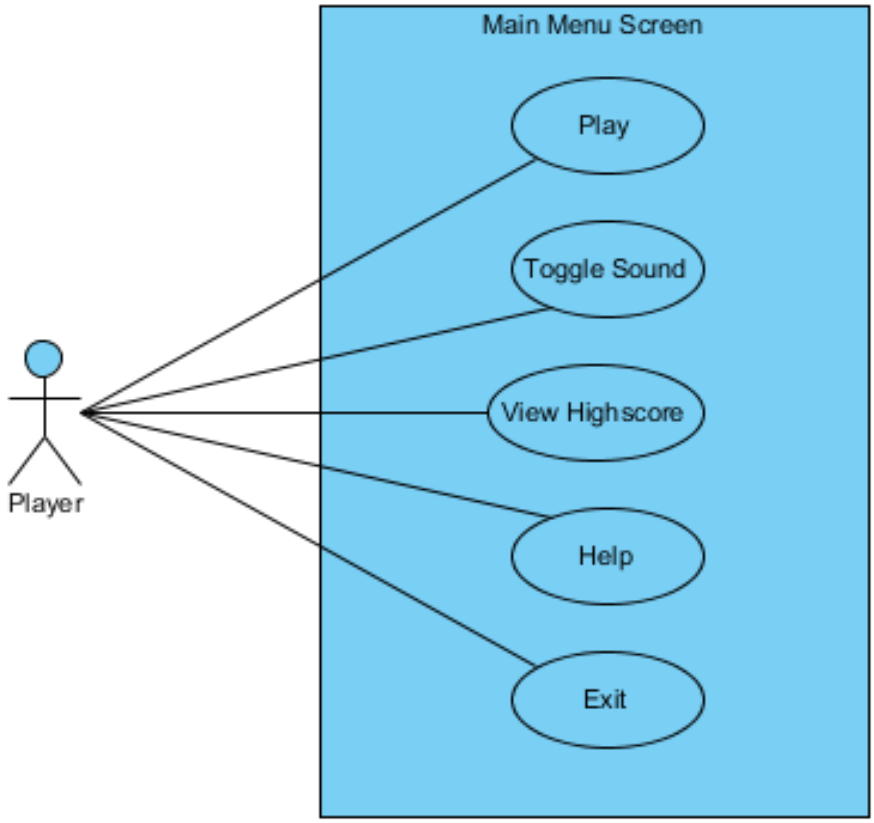
Mobile Device			
	Processor	**RAM**	**Disk Space**
Android 4.1.2 (Jelly Bean)	1 Ghz	512 MB	10 MB

2.4 Constraints

- If the game is being run on computer, the computer should be able to run Java applications.

2.5 Use Case Models

2.5.1 Main Menu Screen:



Play: The user presses ENTER or touches the play button on the screen to dismiss the introduction and start playing the game.

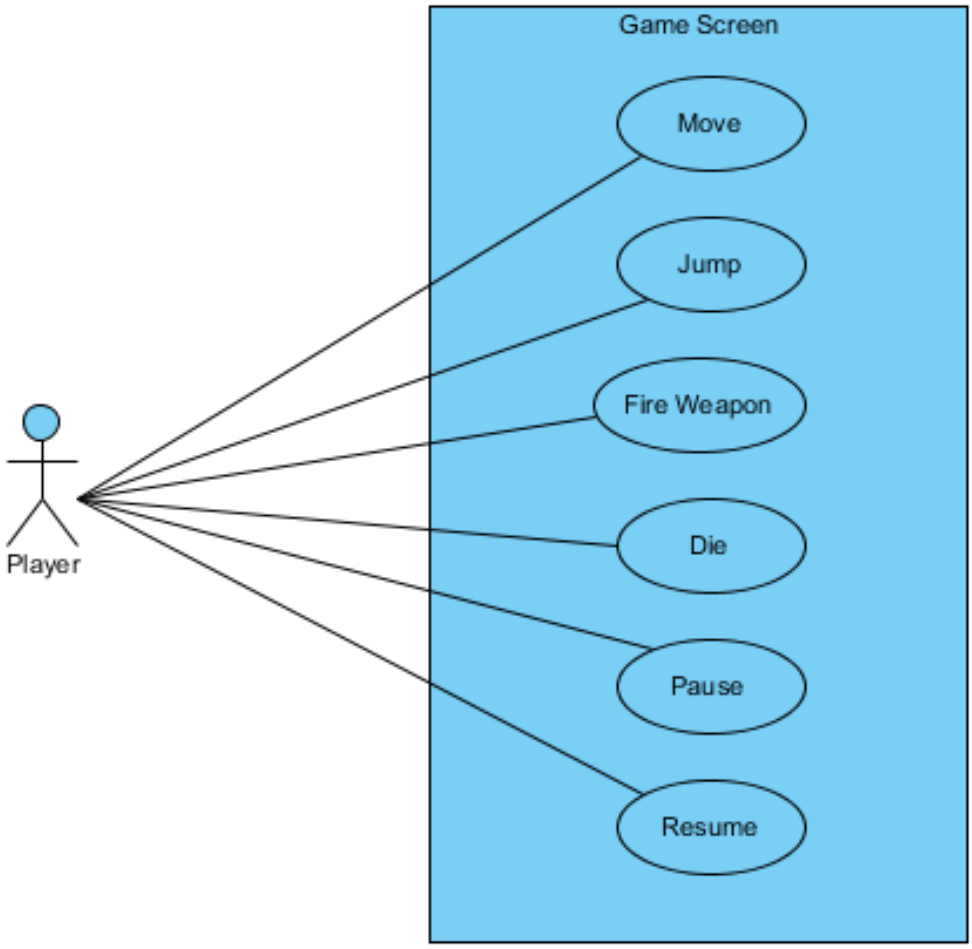
Toggle Sound: The user presses this button to switch the sounds on or off.

View High-scores: The user selects this option to view his last 10 highest scores.

Help: Displays the instructions and strategies to play the game.

Exit: To quit the game and return back to the OS.

2.5.2 Game Screen:



Move: The player moves the character via the arrow keys or the on-screen touchpad, allowing him to move in the left or right direction.

Jump: The user presses the Z key or taps the jump icon on the screen to jump.

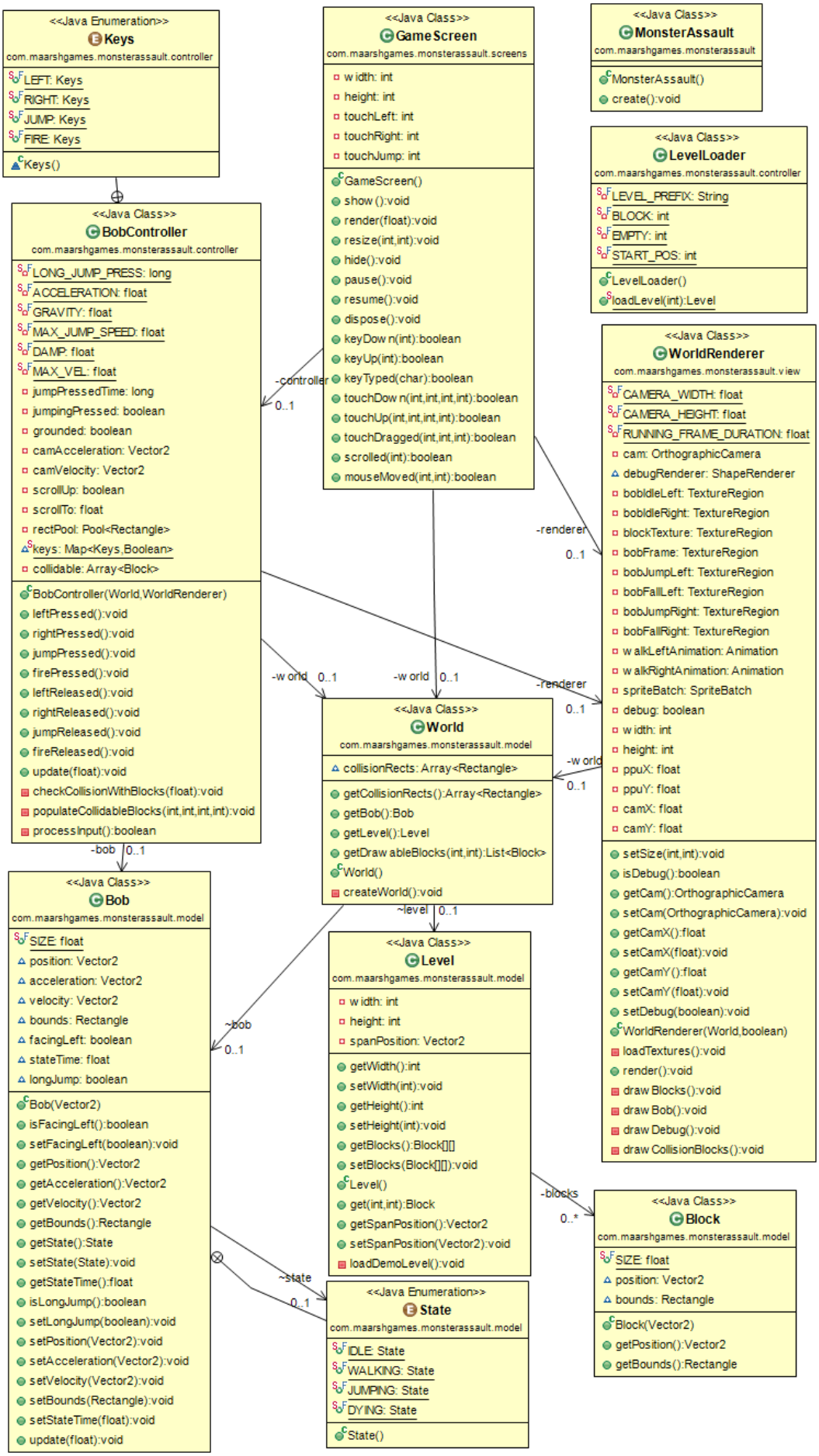
Fire Weapon: The user presses the X key or taps the fire icon on the screen to fire laser beams.

Pause: To temporarily halt the game, the player presses the Esc key or taps the pause icon on the screen. The game will display “Game Paused” in the center of the screen. Switching away from the game will place the game in a paused state.

Resume: The user presses the resume button to resume the gameplay.

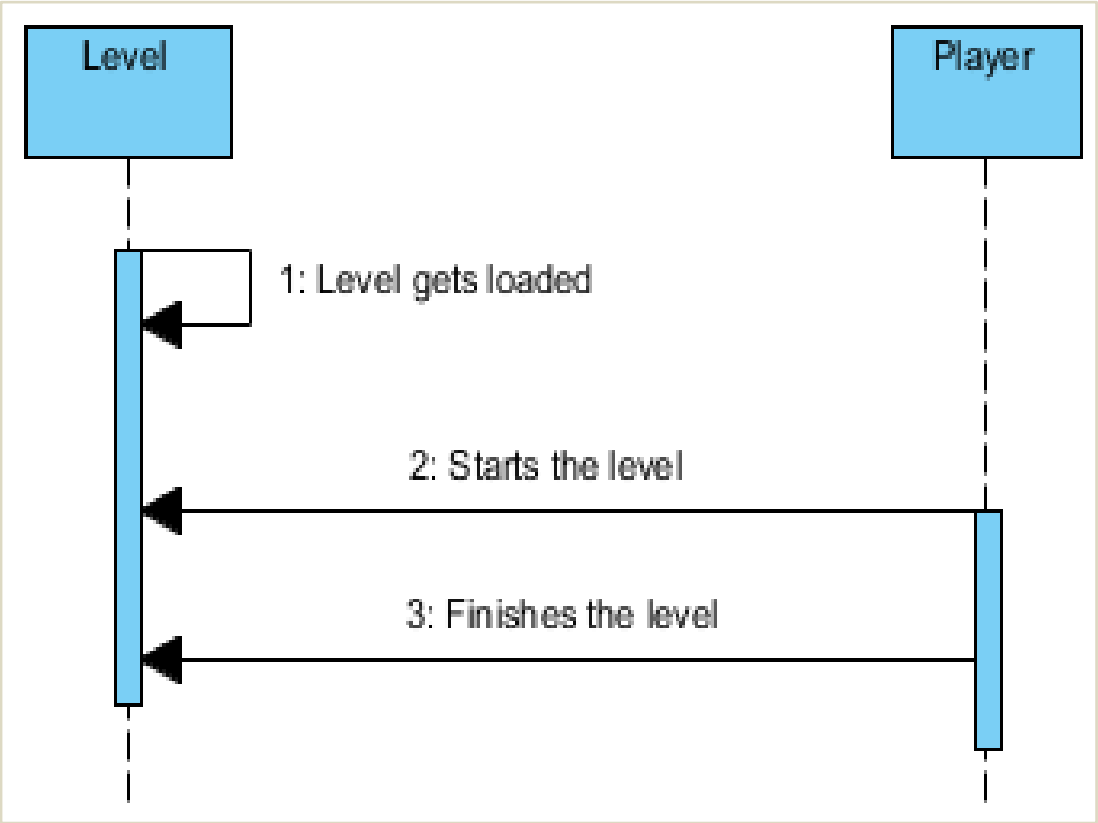
Die: The user is shown this screen when the character dies.

2.6 Class Diagram

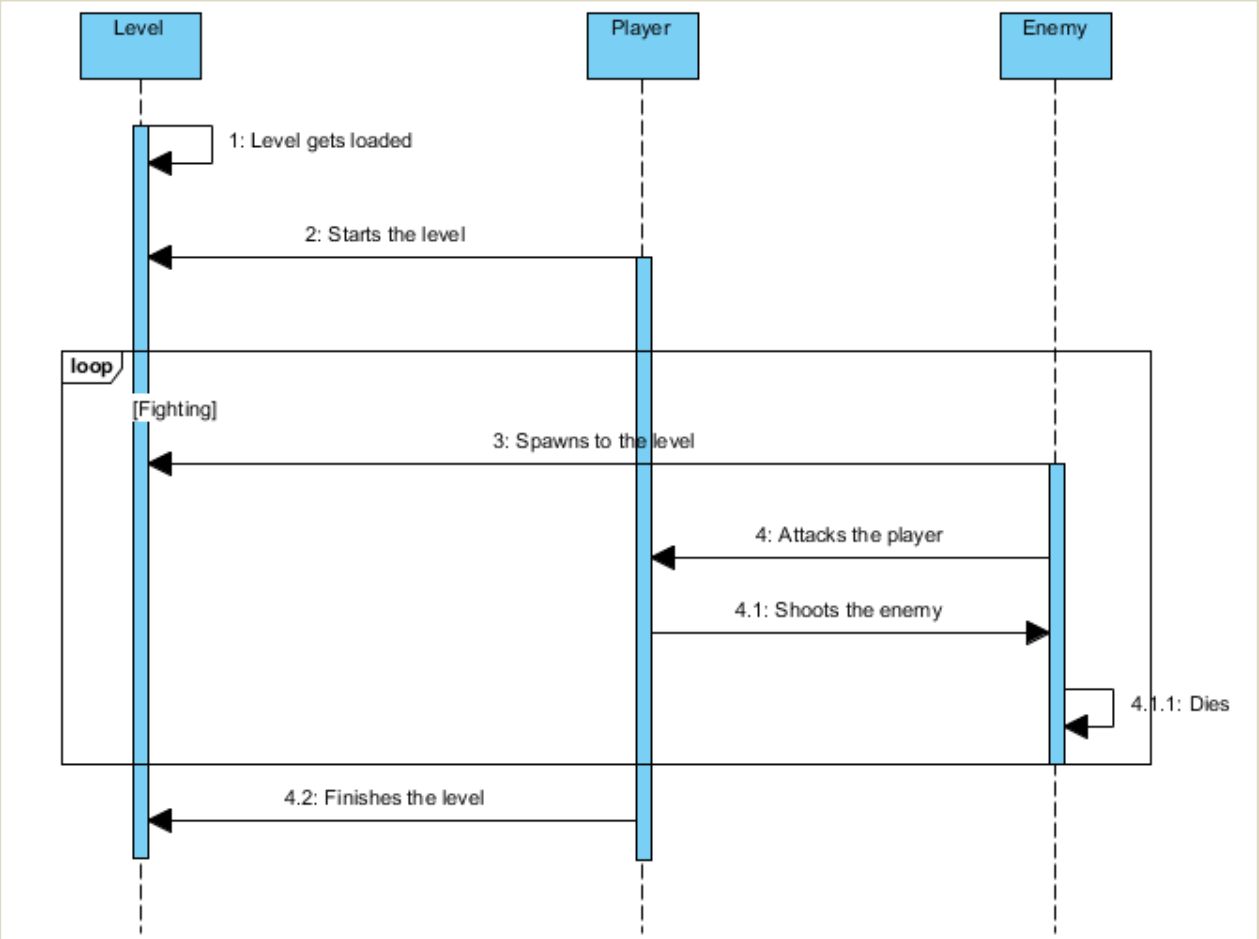


2.7 Sequence Diagram

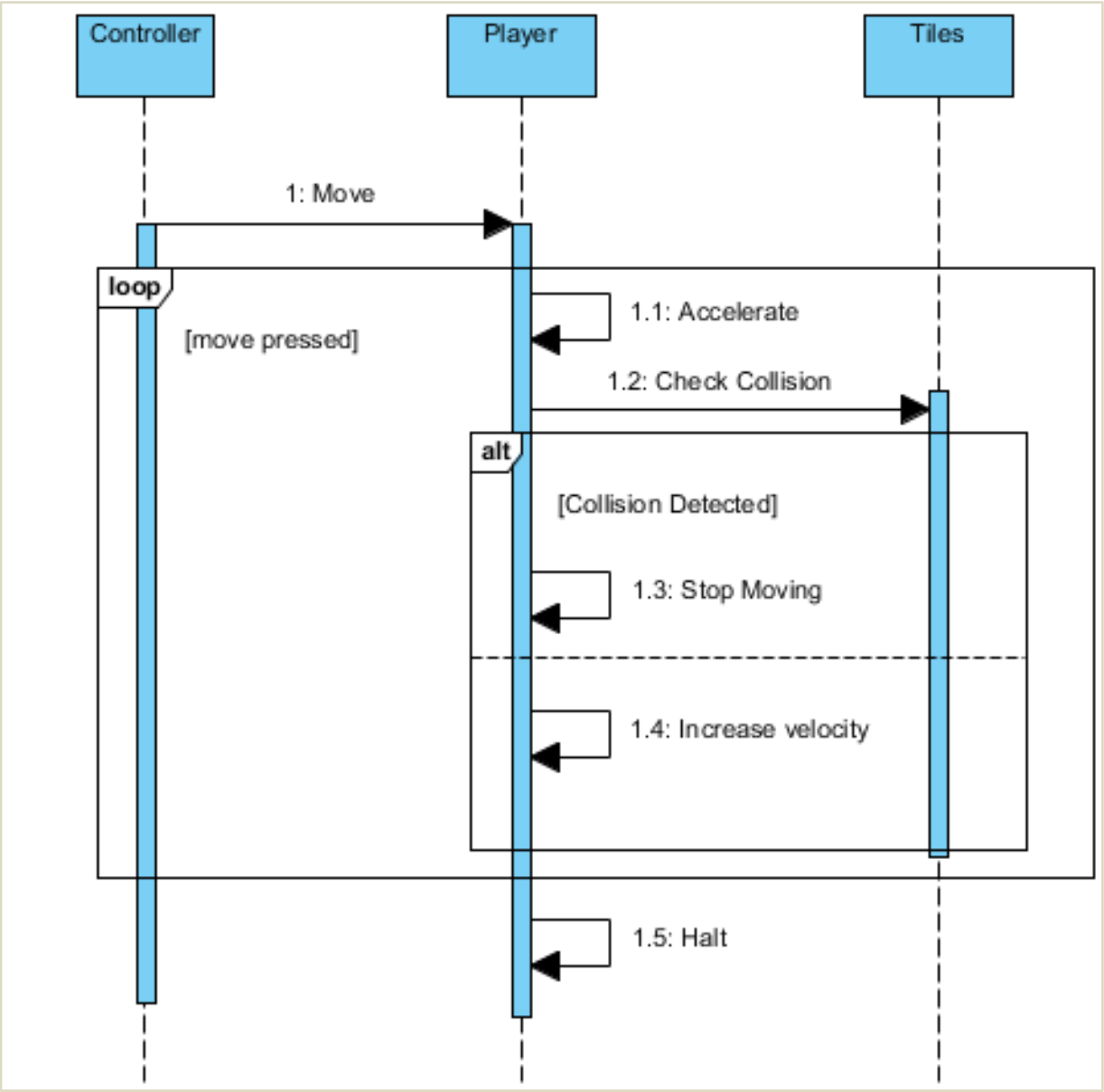
2.7.1 Basic game behaviour



2.7.2 Detailed game behaviour

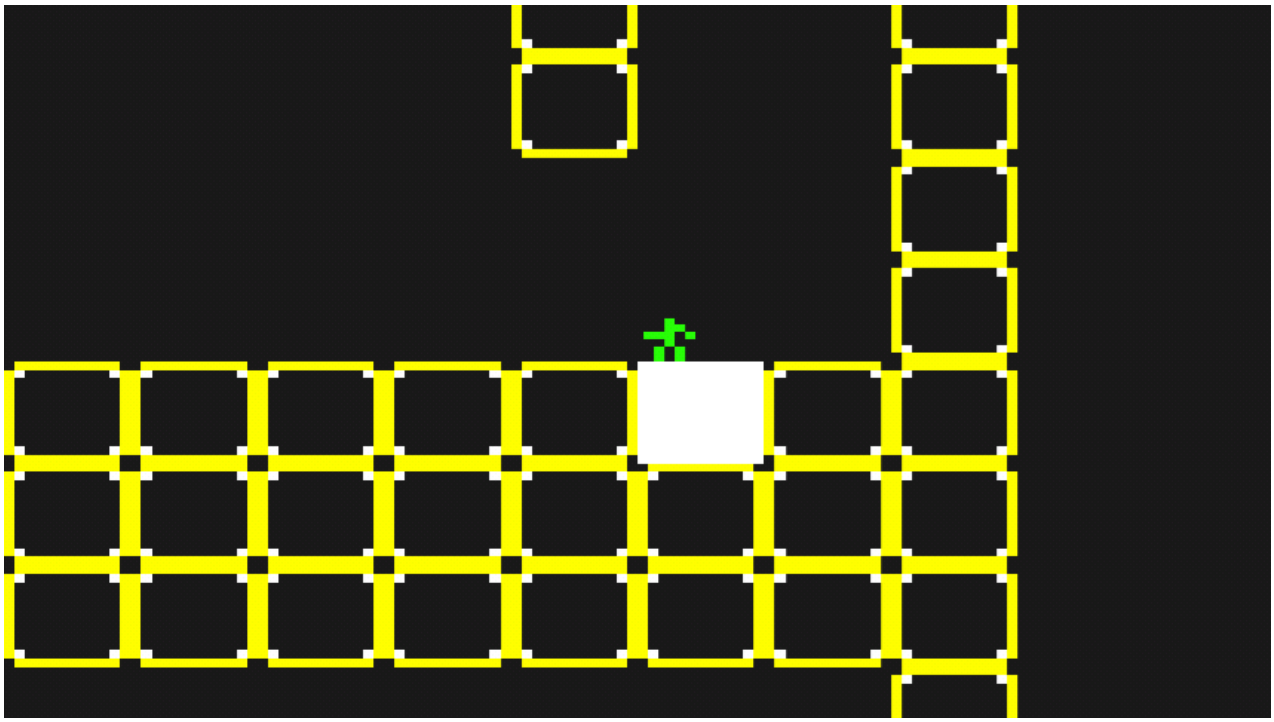


2.7.3 Move Player

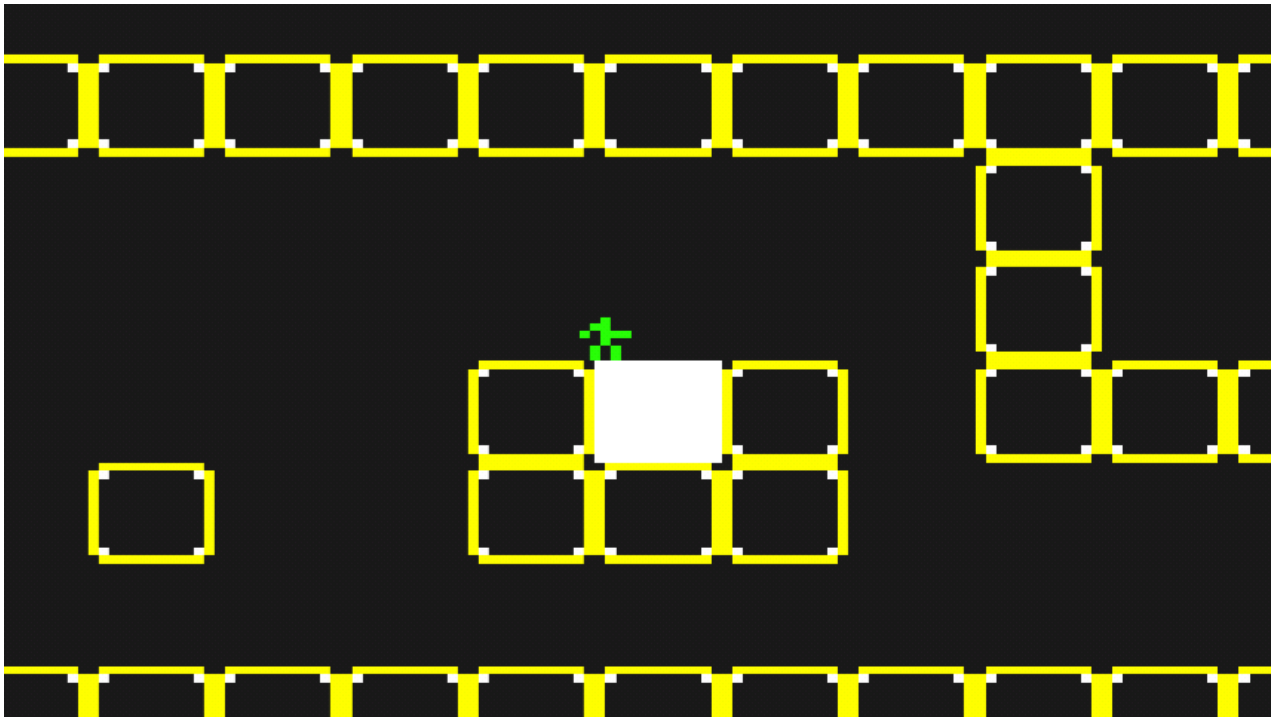


3.Sample Screenshots

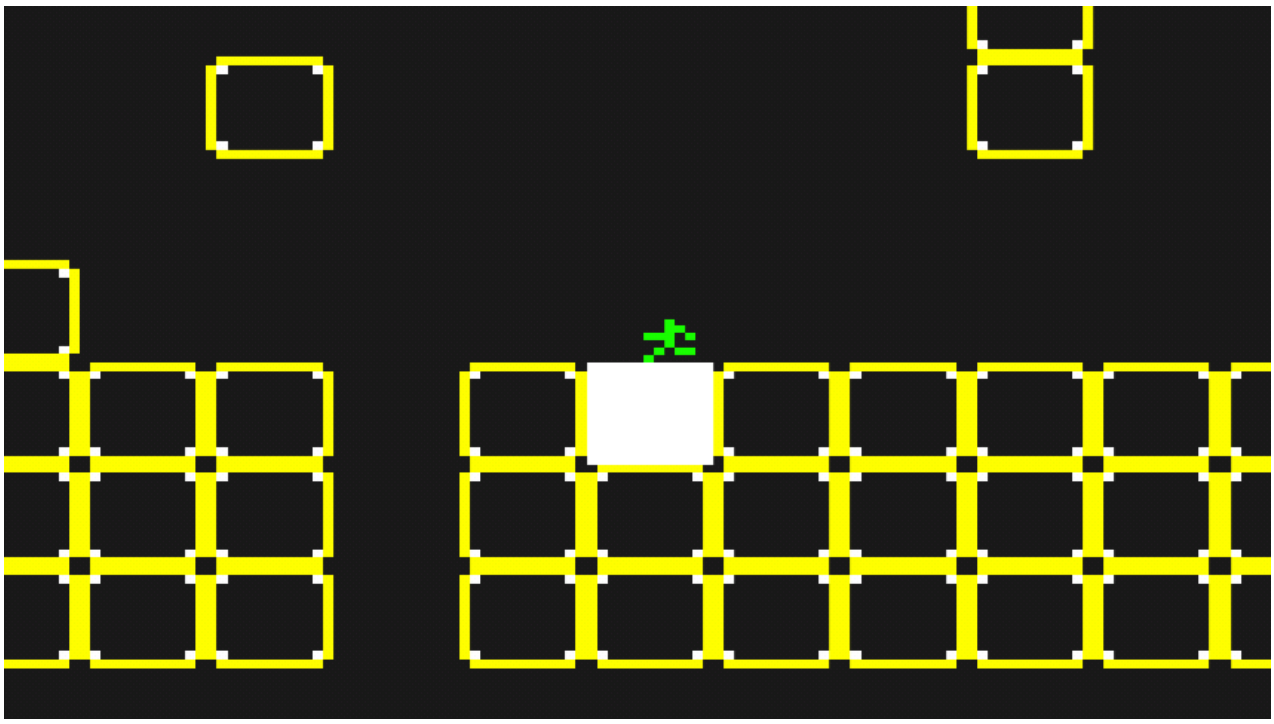
3.1 Player idle – facing left



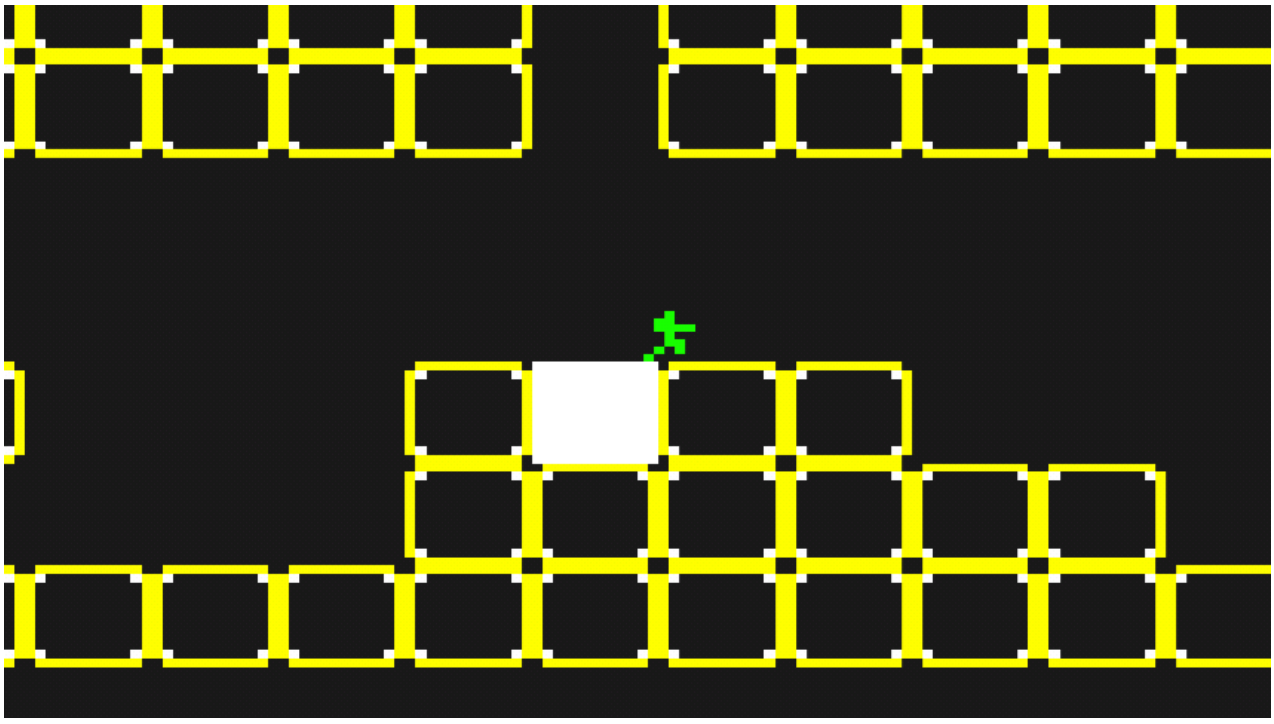
3.2 Player idle - facing right



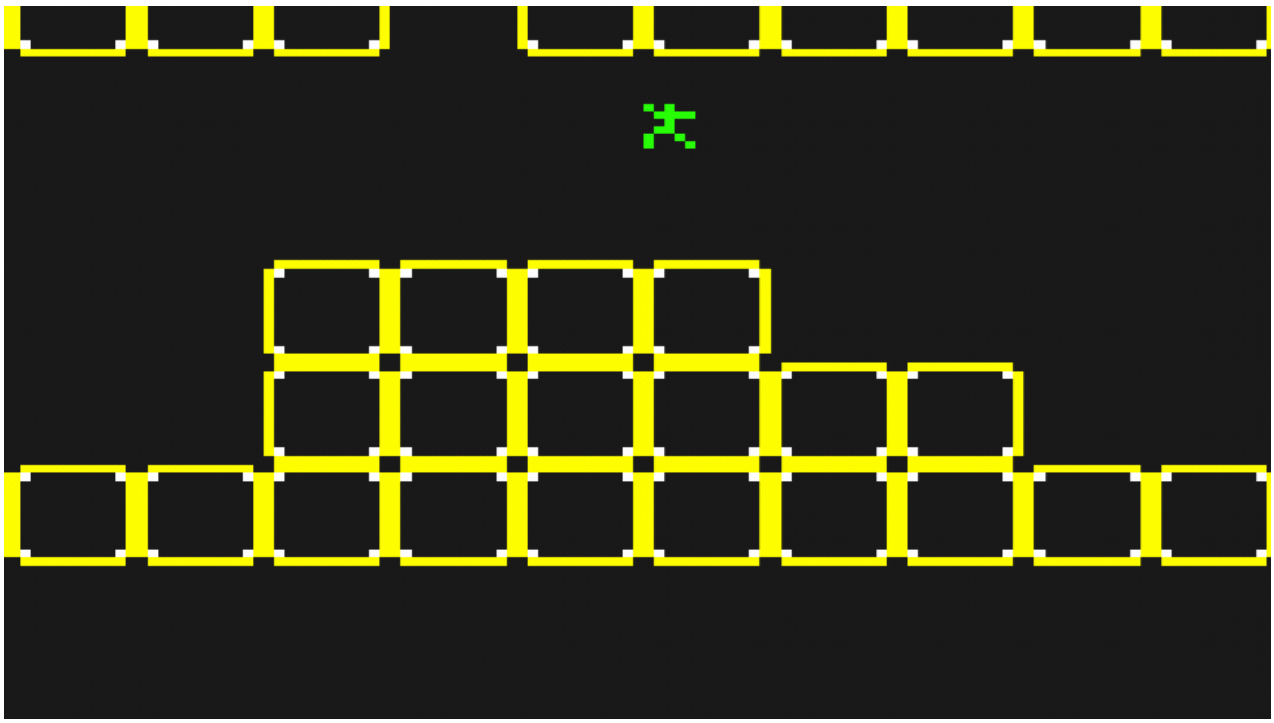
3.3 Player running left



3.4 Player running right



3.5 Player jumping



4. Conclusion & Future Scope

We have been implementing this game for about two months, and it has turned out to be a very learning and beneficial experience. This game is still a work in progress. Many other useful features can be added such as multiple levels, obstacles, monsters attacking, background music, sound effects, etc.

The game incorporates features like moving left, right, jumping and shooting. The movement of the player 'Bob' is decided by the collision detection with tiles and further calculations to accelerate or decelerate.

The experience of working in a team and integration of modules is a very important achievement for our team. We hope to make this game user friendly with good user interface and enjoyable for all age groups.

5. References

- Fundamentals of Software Engineering by Rajib Mall
- Beginning Android Games 2nd Edition by Mario Zechner and Robert Green
- Game Development Tutorial – Kilobolt Studios (http://www.kilbolt.com)
- Libgdx Documentation (http://libgdx.badlogicgames.com/documentation.html)
- Libgdx Wiki (https://github.com/libgdx/libgdx/wiki)
- Wikipedia (http://www.wikipedia.org)
- Open Game Art (http://www.opengameart.org)
- Sprite Land (http://www.spriteland.com)
- Stack Overflow (http://www.stackoverflow.com)