“Nimble: A Game Maker”

A Project Report

Submitted in partial fulfillment of the

Requirements for the award of degree of

Bachelor of Science (Information Technology)

By

“Jafri Tahrim Jafar Abbas”

Roll no: 13

Under the esteemed guidance of

Asst. Prof. Aarya Joshi

DEPARTMENT OF INFORMATION TECHNOLOGY

SHREE L. R. TIWARI DEGREE COLLEGE OF ARTS, COMMERCE & SCIENCE, Mira Road

(***Affiliated to University of Mumbai***)

THANE, 401107

MAHARASHTRA

2022-23

SHREE L. R. TIWARI DEGREE COLLEGE
(Arts | Commerce | Science)
Approved by Goverment of Maharashtra & Affiliated to University of Mumbai

CERTIFICATE

Department of Bachelor Science (Information Technology)

This is to certify this project report entitled " Nimble: A Game Maker" submitted to **University of Mumbai, Maharashtra** is bonafide record of work done by " Jafri Tahrim Jafar Abbas " for Academic Year 2022-23 under the guidance of " Prof. Aarya Joshi" from 3o July (project starting date) to _____(project completion date).

Date of Evaluation: _____

_____ Internal Examiner (Guide)

_____ College Stamp

_____ External Examiner

_____ Head of the Department

Acknowledgement

I would like to convey my deep appreciation to my Guide Asst. Prof. Aarya Joshi of Information Technology department for her valuable guidance and encouragement in completion of my project work.

I would also like to express my gratitude to our principal Dr. Sanjay Mishra for providing all the required facilities to accomplish my work.

Finally, I would like to thank my parents and friends, and especially my cousin Tanveer Haider Jafri for his continuous help, without them this task would not have been completed.

Jafri Tahrim Jafar Abbas

DECLARATION

I hereby declare that project entitled "**Nimble: A Game Maker**" done at Shree L.R. Tiwari Degree college, not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university. The project is done in partial fulfillment of requirement for the award of degree of **Bachelor of Science (Information Technology)** to be submitted as final year project as a part of our curriculum.

Jafri Tahrim Jafar Abbas

INDEX

Chapter 1
Introduction

1.1 Background

Nimble is a 2D platform game maker designed for younger age group. Unlike majority of game making software and engines in the market, Nimble is a code-free application that makes game development easier and quicker. While the existing popular game engines provide hundreds and thousands of functionalities to make their games more realistic, they can also make the process very complex, which might demotivate young audience from creating games.

As time passes, the world of gaming is growing faster than ever, and so is the curiosity of kids who wonder how games are being made. While the software development industry is constantly trying to make software and automation tools easier to utilize, they often miss out the young audience out of the picture, this is where Nimble comes in. Nimble provides simple and easy to understand user interface, which provides simple yet functional tools for making platform games. It provides young developers options to choose content from their own systems, which lets them unleash their creative side and also make the game making process fun.

1.2 Objectives

The objective of this project is to make a simple and easy to use game making platform which requires no prior coding knowledge. It is especially created and aimed for audience of younger age group and for those who are not familiar with coding.

Functionalities provided by Nimble Game Maker:

1. Providing a platform where people with no knowledge of coding, especially kids, can make platform games.
2. Interactive, simple, and easy to use UI (User Interface).
3. Import images from their own system (bmp, png, tga and jpg).

4. Use audio from their own system (WAV, OGG/Vorbis and FLAC).
5. Simulates physics interaction inside the game.
6. Manages input generated by various input devices (mouse, keyboard).
7. Provides structure of the game world.
8. Renders graphics of the game with graphics API..

1.3 Purpose, Scope and Applicability

1.3.1 Purpose

This project aims to provide a interactive, simple and easy to use environment for game development. It is designed to be a code free game development platform for those who do not know how to code. It will provide prewritten structure of platform games where everything is one click away. This application will present users an option to choose graphics from their own PCs, supported images formats being bmp, png, tga and jpg. It will also allow users to choose audio files from their PCs, supported audio formats will be WAV, OGG/Vorbis and FLAC.

The purpose of this application is to provide system to render game graphics with the help of graphics API, and also to define behaviour of certain objects (player, obstacle) through physics simulation. It also aims to present an easy to use User Interface to the users, where the user will customize the game graphics, audio, input keys and animation for the game.

1.3.2 Scope

The scope of the project includes:

- **Graphics Rendering:** In this application, graphics support and rendering will be provided by SFML library which is based on OpenGL API. It will give users an option to choose graphics from their own system. SFML only supports bmp, png, tga and jpg.
- **Physics interaction:** This functionality will simulate physics interaction and 2D transformation inside the game world, like collision detection, speed, gravity.

- **Sound effects and tracks:** The application will provide users means to import audio files into the game with the help of SFML library. SFML only supports WAV, OGG/Vorbis and FLAC.
- **Events:** This functionality binds a method to input event, which is generated by input devices like mouse and keyboard. Since, input system is prewritten, through UI users will be able to customize it.
- **Structure of game:** This functionality brings entities and their components together and provides the structure of the game world. It will be prewritten, therefore users won't have to script it in the application.
- **User Interface:** By creating an interactive UI, the application will provide users the benefit of doing minimal work for creation of a platform game. This functionality will eliminate a scripting system, and hence removing the requirement to code.

1.3.3 Applicability

This application will allow users to create 2D platform games quickly and easily. It will provide functionalities to import audio and graphics from the user's system.

Like majority of the game engines, Nimble will provide system to render graphics through graphics API, system to apply physics to game entities, means to apply 2D transformation to objects and it will also create a entity-component system to bring together entities with their required components (graphics, physics, audio). But unlike majority of the game engines, everything in the application required to make the structure of the game world will be prewritten.

The input events generated by the input devices, like mouse and keyboard, will bound with the methods, the user will be able to customize which key to press to trigger the event. With easy to use and understand UI (User Interface), the user will only have to enter required values to customize the game world to their liking, then the application will create and render the game for them, this will make Nimble a fast, easy and code-free game maker.

This makes our application best suitable for younger age group and for people who do not have any prior knowledge about coding and game development. This application works without the need of internet. This game developing platform will immensely reduce the hardwork and time required to make 2D platform games.

1.4 Organization of report

This document contains all features of the project named “Nimble: A Game Maker”, which is all about creating 2D platform games with no requirement to code.

In **Chapter 1** we discussed the background, objectives, purpose, scope and applicability of the project.

In **Chapter 2,** the subsequent chapter will include survey of technologies that will be used to make this project.

In **Chapter 3** we will study about the existing system in the market and their drawbacks, then we will study about our proposed system and what it offers. Furthermore, we will study about the requirement of the project, which will discuss functional as well as technical requirements. Functional requirement will include requirements from the user’s end, whereas technical requirements will include software specifications as well as specifications to run the project. This chapter will also include project planning and project scheduling.

In **Chapter 4** we will be discussing different designs of the system, which will include Module Design, Models, Entity-Relationship Diagram, Data Flow Diagram and UML Diagrams.

In **Chapter 5**, pseudo code, and external packages and libraries will be discussed along with testing approaches like unit testing and integrated testing.

In **Chapter 6,** the results of the implemented project and their functionalities will be discussed.

In **Chapter 7**, the documentation of our project will be concluded while discussing the limitation of our system and future scope of this project, which will include system maintenance and future enhancement to it.

Chapter 2

Survey of Technologies

➢ Tools used in this project:

- Visual Studio
- Inno Setup
- ReSharper

➢ Technologies used in this project:

- SFML library
- C#
- .NET framework
- SQLite

➢ Tools

○ Visual Studio

- Visual Studio is one of the most popular and powerful integrated development environment (IDE) developed by Microsoft. It is used to edit, debug, and build code, publish computer programs, as well as websites, web apps, web services and mobile apps.
- It supports thirty-six different programming languages including C, C++/CLI, C#, F#, Visual Basic .NET, XML, HTML, JavaScript, TypeScript and CSS.
- Other languages such as Node.js, Python and Ruby are supported via plug-ins.
- It uses Microsoft software development platforms which includes Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight.
- Visual Studio can produce managed code as well as native/machine code.

○ Inno Setup

- Inno Setup is free and among the best tools for creating application installers pack.

- It packs the application and all the files that are needed for its operation into one compact .exe file.
- This tool will allow us to create the necessary links and shortcuts, all the registry entries we might need, and also an uninstaller.
- This will help us create a perfectly customized installation process which will be displayed in an interface with a great design.
- Starting from Windows 2000, Inno Setup supports all the versions of Windows OS.
- Inno Setup supports 64-bit and 32-bit installation.

○ ReSharper

- ReSharper is one of the most popular Visual Studio plug-in, developed by JetBrains.
- It helps with code navigation, code analysis, code maintenance, provides editing features, auto fixes code issues and much more.
- One of its most popular feature is code refactoring, allow us to rename, move, and safely delete symbols; introduce and inline fields, variables, or parameters, and carry out many more transformations effortlessly.
- ReSharper is gaining popularity among developers for writing and maintaining code in a more manageable and enjoyable way, adopting the best coding practices, and delivering higher quality applications faster.

➢ Technologies

○ SFML library

- SFML short for Simple and Fast Multimedia Library is popularly used by developers to ease the development of games and multimedia applications.
- It is designed to provide a simple application programming interface to various multimedia components in computers.
- It works on multiple platforms such a Windows, Linux and macOS.
- Although it is a C++ multimedia library, it has official bindings for C and .NET languages. And through community development, it has bindings for Java, Python, Ruby, Go and other languages.
- SFML provides access to windowing, graphics, audio and network.

- It is based on OpenGL, therefore without any extra effort, it's windows are ready for any OpenGL calls.
- SFML is used heavily in game development, and to an extent game engine development.
- Even though 3D rendering is not supported by SFML, yet it is used for context creation by developers.

○ C#

- C# is general purpose, high-level programming language developed by Microsoft.
- It is object oriented language and runs on .NET framework.
- C# is designed for CLI or Common Language Infrastructure. CLI is a set of specifications by Microsoft that allows use of multiple high-level languages on different platforms without rewriting the code for the specific platform.
- Several features provided by C# will help us create secure, robust and durable applications.

○ .NET framework

- .NET is a software development framework developed by Microsoft for building and running application on Windows.
- It is part of the .NET platform, which is a collection of technologies for building apps for Windows, Linux, macOS, Android and other Operating Systems.
- When an application runs, the CLR takes the assembly and uses JIT (just in time compiler) to turn it into machine code, which then can execute on the specific architecture of the computer it is running on.
- .NET framework supports running websites, services, desktop apps, and more on Windows.
- Currently .NET framework supports more than 60 languages including C#, F#, VB.NET, J#, Python, Cobra, COBOL, Perl, Oberon, ML, VC++, JScript.NET, APL, Pascal, Eiffel, Smalltalk, ADA and more.
- In .NET framework, CLR (Common Language Runtime) provides services like thread management, garbage collection, type-safety, exception handling, and more.
- The Class Library of the .NET framework includes APIs for reading and writing files, connecting to databases, drawing, and more.

- **SQLite**

- SQLite is a software library which provides a relational database management system.
- It is open-source, stand-alone, self-contained, zero-configuration, transaction relational database engine designed to be embedded into an application.
- SQLite is light weight in terms of setup, database administration, and required resources.
- It is great as on-disk file format for desktop applications.
- We can use SQLite to store and manage lots of files, files can be binary or blobs.
- Files can also be saved in the device while their paths are saved in the database.

Chapter 3
Requirement and Analysis

3.1 Existing System

The existing game development applications are too vast and complex for new users to understand, and also the reason why they become quite overwhelming for the young audience to operate. Since they provide hundreds and thousands of functionalities, these applications are also getting bigger and are taking more space on the disk.

Knowledge of coding is a prerequisite in majority of the game engines and game development tools, which is not ideal for young users and those who do not know how to code. With high complexity and also a need to code for specifics, the process becomes time consuming. The majority of the game engines and development are generally not free.

3.1.1 Drawbacks of current system

1. Complex
2. Requires knowledge of coding
3. Time Consuming
4. Overwhelming for younger age group
5. Generally not free
6. Bigger in size

3.2 Proposed System

Nimble game maker will be having 6 components:

1. User Interface
2. Engine
3. Renderer
4. Database Connection

5. Local Database
6. Coordinates Input

User Interface

- The user interface will be interactive, easy to understand and easy to use.
- It will contain fields to customize a platform game, like which key to bind to which event, which audio files to play for certain event and customize actors, background, footer, header, etc.
- Users will be able to see the imported graphics and changes made on the display field.
- The menu strip will have options for creating, opening, saving and running game files.

Engine

- The engine will contain the prewritten structure of a platform game so the users will not be required to code.
- It will contain functions that will simulate physics interaction within the game world, defining how the game components interact with each other.
- It will also contain functions for managing audio system for the game.

Renderer

- The renderer will be implemented by SFML library.
- It will provide a window module that will make it possible to create application windows, and will collect user input, such as mouse movement or key presses.
- It will provide a graphics module which will provide all functionalities that are related to 2D rendering which includes images, texts, shapes, and colors.
- It also provides a module to work with sound. It will let us load a music theme and play it on the computer's speakers.

Database Connection

- This component will be a class containing code for SQLite Database Connectivity.
- It will contain code for CRUD (Create, Read, Update and Delete) operations for the game scores.

Local Database

- The database management system will be implemented by SQLite Database.
- It will manage game scores and it's entries will be managed via the coded database connection to the game splash screen.

Coordinates Input

- This component will take input of coordinates from the user from UI and will send it to the engine.

3.3 Requirement Analysis

3.3.1 Functional Requirements

For the software we're developing, the following requirements were raised during the analysis of the needs of users:

- The user should be able to custom pick audio for the events and appearance of the objects in the game through the user interface.
- The user can pick graphics and audio from their own computers.
- The users should be able to customize the input setting, position of the objects in the window.
- The user should be able to obstacle speed in the game.
- The user should be able to see the changes made to the game on the display field in the User Interface.
- The user should be able to run the game.

- The user should be able to create more games.

After analyzing the requirements of the task to be performed, the next step will be to analyse the problem and understand its context. The first activity in the phase will be studying the existing system and second activity is to understand the requirements and domain of the new system. Both these activities are equally important, but the first activity serves as a basis of giving the functional specifications and then providing successful design of the proposed system. Understanding the properties and requirements of a new system is more difficult and it also requires creative thinking. Understanding of existing running system is also difficult, as improper understanding of present system can lead diversion from solution.

3.3.2 Technical Requirements

The software requirements specification has been produced at the culmination of the analysis task. The function and performance that are allocated to the software as part of system engineering are refined by establishing a complete information description, a detailed functional and behavioral description, an indication of performance requirements and design constraints, appropriate validation criteria, and other data pertinent to requirements.

Hardware Specification

Disk Space: 30 GB or higher

Memory: 8 GB RAM

Processor: more than 2 cores

Input/Output: keyboard, mouse, speakers/headphones,monitor

Connection: Internet connection not required

Software Specification

- Windows OS
- Visual Studio
- Inno Setup
- ReSharper
- SFML library
- SQLite

3.4 Project Planning

Project planning involves the creation of set of plans which will help the developers to guide and manage time, cost, quality, risk.

Following are the process included in project planning:

- **Scope Planning:** The initial process is defining the project scope and determining appropriate methods to accomplish the project. This also includes specifying the project's scope to facilitate work-breakdown structure creation.
- **Preparing the work breakdown structure:** In this step, the project is split into tasks and sub tasks, and forms a work breakdown structure by grouping and listing various **Project Scheduling:** In this step, schedules of activities involved in the project and their implementation sequences are listed.
- **Resource Planning:** This step concerns with the things which are responsible to accomplish the project.
- **Budget Planning:** In this step we will be estimating how much cost or budget is required for the completion of the project. It also includes required resources and the way in which they are used to balance the cost and time are being estimated.
- **Risk Management:** This step involves risk planning and migration strategy used to handle risks.
- **Quality Planning:** This step is taken to ensure the quality of the product.It is checked with some already defined quality criteria.

3.5 Project Scheduling

Gantt chart, commonly used in project management, is a great way to schedule project activities. The chart contains list of activities along y-axis and the time-scale along x-axis. Each activity is represented by a bar and the position and length of the bar reflects the start date, duration and end date of each activity. This allows us to see:

- What the various activities in the project are
- When each activity begins and when it ends

- How long those activities are scheduled to last
- Where does the activity overlap with other activities, and by how much
- The start and the end date of whole project

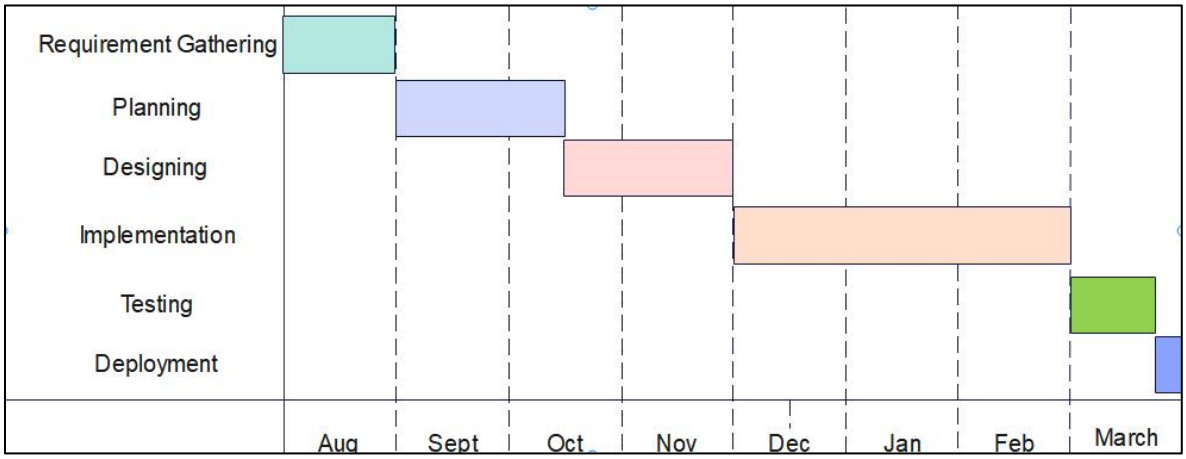


Fig 3.1 Gantt Chart of Nimble: Game Maker

Task	Start Date	End Date	Duration
Requirement Gathering	August 1	August 31	4
Planning	September 1	October 14	6
Designing	October 15	November 30	6
Implementation	December 1	Feburary 28	12
Testing	March 1	March 20	3
Deployment	March 21	April 1	1

Fig 3.2 Table used in Gantt Chart for Nimble: Game Maker

Chapter 4

System Design

4.1 Module Division

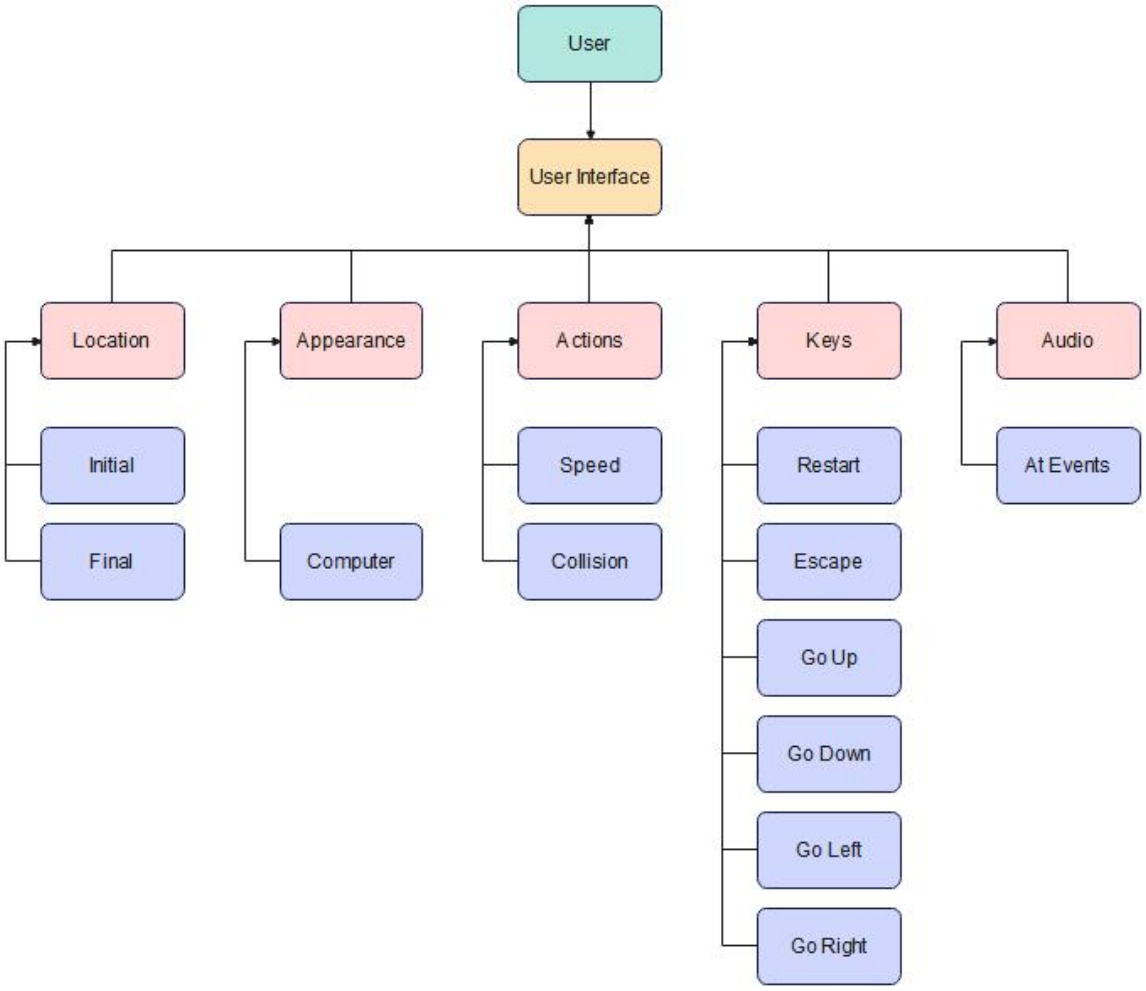


Fig 4.1 Module Division of Nimble Game Maker

4.1.1 Event Table

In the event table we have described all the modules and the events that occur with the help of those modules.

Definition of contents of event table:

- **Event table:** Table that list events in rows and the key piece of information about each event in columns.
- **Event:** An event occurs at specific time and place, and should be described and remembered by the system.
- **Trigger:** It is an occurrence that tell the system that an event has occurred, either arrival of data needing processing or of a point in time.
- **Source:** It is an external agent or actor that supplies data to the system.
- **Activity:** It is behavior that the system performs when an event occurs.
- **Response:** It is an output produced by the system that goes to the destination.
- **Destination:** It is an external entity that receive data from the system.

Description: Events are transactions, events are the cause of the transactions and events will generate transactions. While the analyst is developing the lists of events, they should note the additional information about the events for later use. Event Table is a convenient way to record information about the requirements for information of the software project.

Sr no.	Event	Source	Trigger	Activity	Destination
1	Initial Location	User	Initial Location	Changes initial location of the object	Screen
2	Final Location	User	Final Location	Changes final location of the object	Screen
3	Appearance from Computer	User	Game Entities	Changes appearance of the game entities	Screen
4	Gravity	User	Player Object	Enables gravity for the player object	Screen
5	Collision	User	Player Object	Enables collision detection for the player object	Screen
6	Restart	User	Game State	Restarts the game	Screen
7	Escape	User	Game State	Closes the game	Screen
8	Up	User	Player Object	Makes player go up	Screen
9	Down	User	Player Object	Makes player go down	Screen
10	Right	User	Player Object	Makes player go right	Screen
11	Left	User	Player Object	Makes player go left	Screen
12	Audio At Events	User	Event State	Plays audio at certain events	Speakers

Table 4.1 Event Table for Nimble Game Maker

4.2 Model Used

Here we use Waterfall Model, it is a linear, sequential approach to the software development life cycle and is the oldest, simplest and widely used process model for software development.

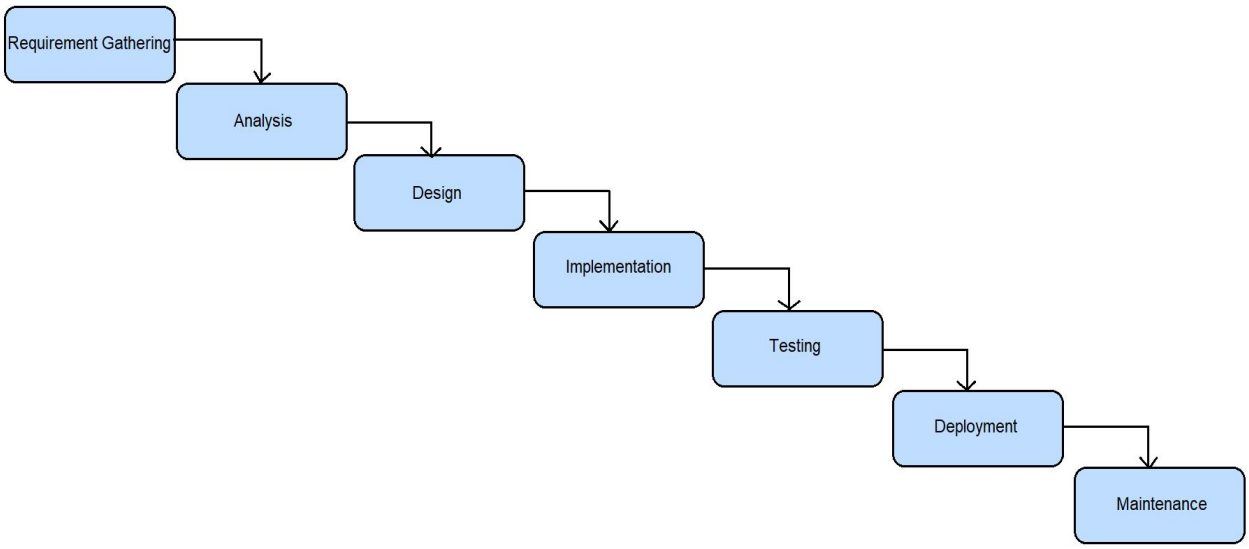


Fig 4.2 Waterfall Model

4.3 ER Diagram

An Entity Relationship Diagram (ERD) or Entity Relationship Model, is a visual representation of different entities within a system and how they are related to each other. Entity relationship diagrams are widely used in software engineering during the planning stages of a software project. They are helpful in identifying different system elements and their relationships with each other.

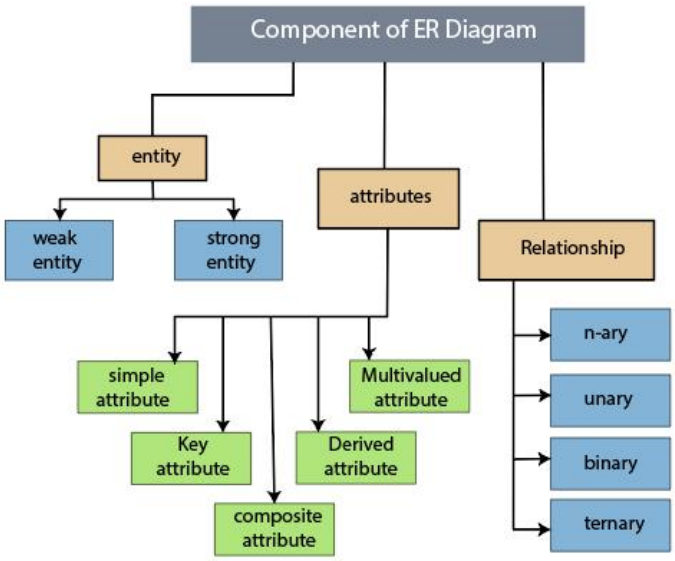


Fig 4.3 Components of Entity Relationship Diagram

- Entity Relationship Diagram symbols and notations mainly contains three basic symbols which includes rectangle, oval and diamond shapes to represent relationships between elements, entities and attributes.
- There are also some sub-elements which are based on main elements in Entity Relationship Diagram.
- ER Diagram is a visual representation of data that describes how data is related to each other by using different ERD Symbols and Notations.

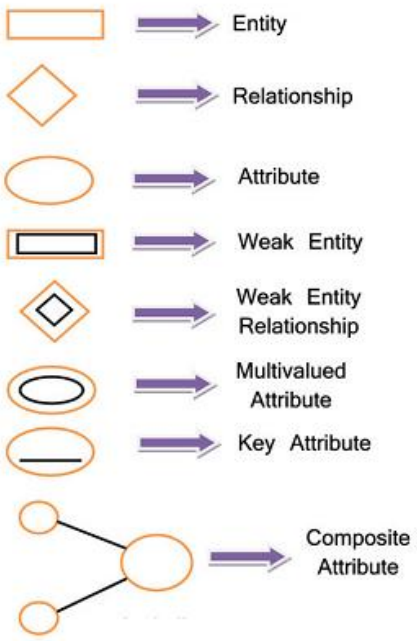


Fig 4.4 Notations of ER Diagram

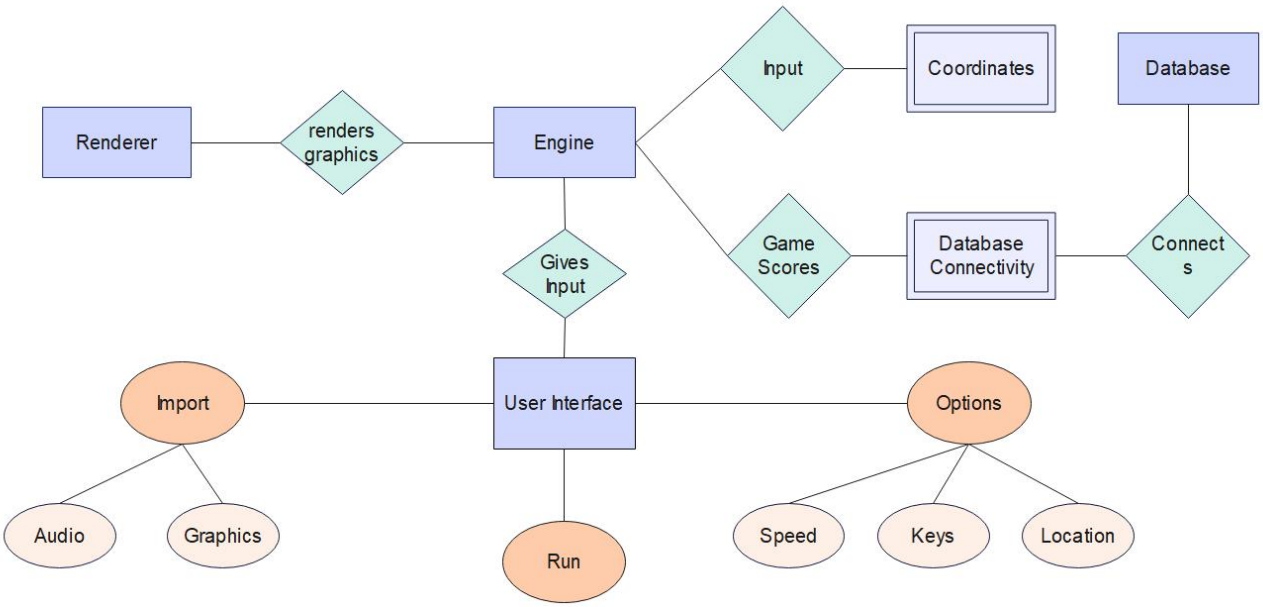


Fig 4.5 ER Diagram of Nimble Game Maker

4.4 Data Flow Diagram

- **0th DFD level:**

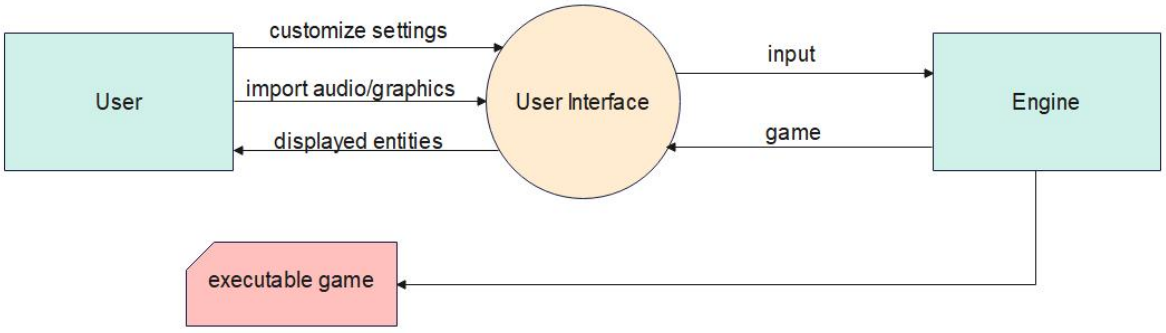


Fig 4.6 Data Flow Diagram level 0

- **1st DFD level:**

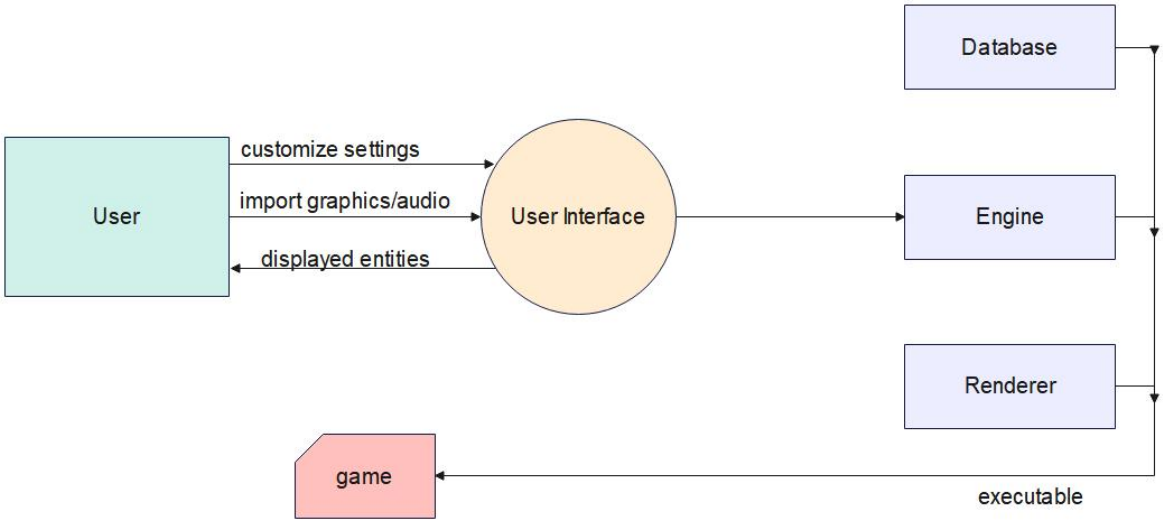


Fig 4.7 Data Flow Diagram level 1

- **2nd DFD level:**

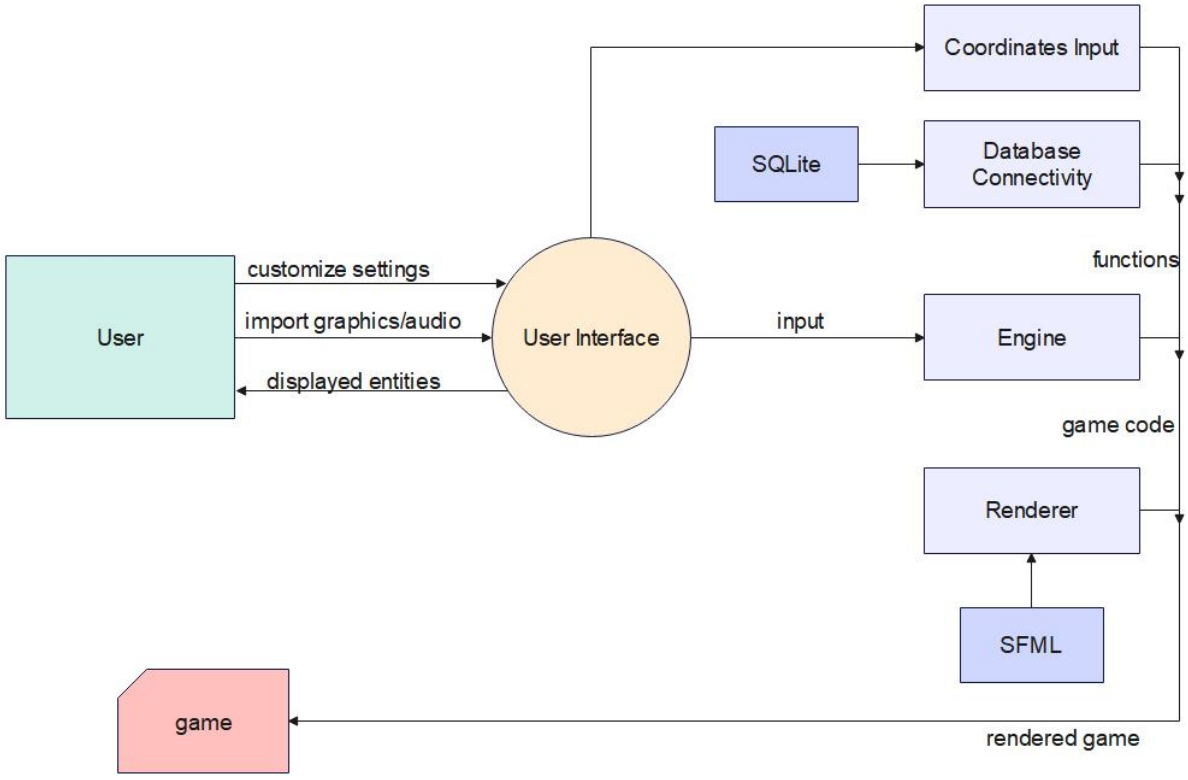


Fig 4.8 Data Flow Diagram level 2

4.5 UML Diagrams

- UML is short for Unified Modeling Language.
- It is a standard visual modeling language used for modeling business and similar processes, analysis, design, and implementation of software-based systems.
- UML diagrams can be beneficial when:
- We're bring new team members or when developers switching teams up to speed quickly.
- Navigating source code.
- We're planning out new features before any programming takes place.
- We communicate with technical and non-technical audiences with more ease.
- UML diagrams have two main categories; structural diagram and behavioral diagrams.
- Class Diagram falls under Structural Diagrams.
- Use Case Diagram and Activity Diagram falls under Behavioral Diagrams.

4.5.1 Class Diagram

- A class diagram depicts the static structure of a system.
- It shows us relationships between different classes, objects, attributes, and operations.
- The class diagrams are widely used in the modeling of object-oriented systems as they are the only UML diagrams, which can be mapped directly with different object-oriented languages.
- It can be used for analyzing and designing the static view of an application.
- It also describes responsibilities of a system.

4.5.2 Use Case Diagram

- Use Case Diagram is used for representing the dynamic behavior of a system.
- By incorporating use cases, actors, and their relationships, it encapsulates the system's functionalities.
- It accumulates the system's requirement, that is including internal and external influences.

- It shows how an entity from an external environment can interact with a part of our software system.

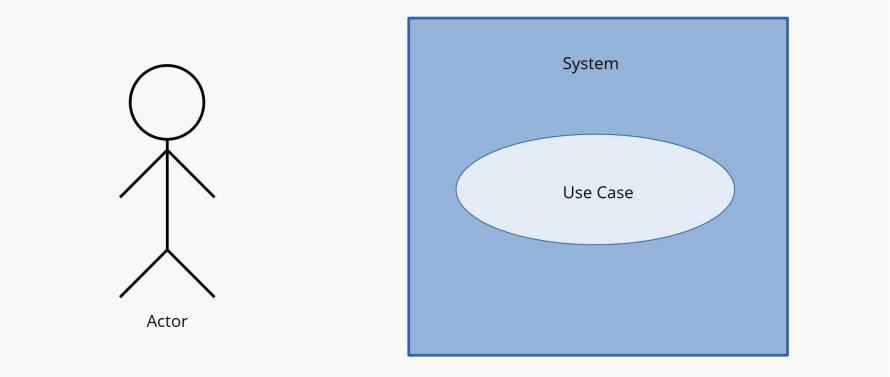


Fig 4.9 Use Case Diagram Model

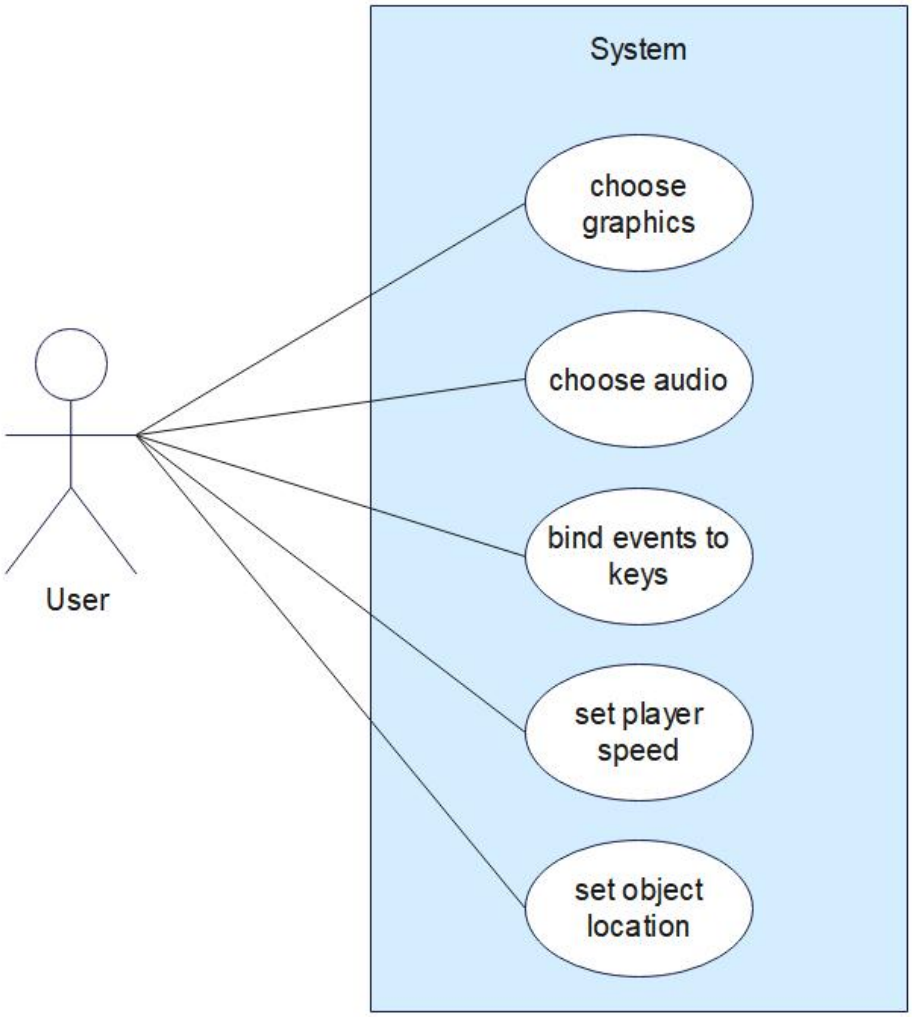


Fig 4.10 Use Case Diagram for Nimble Game Maker

4.5.3 Activity Diagram

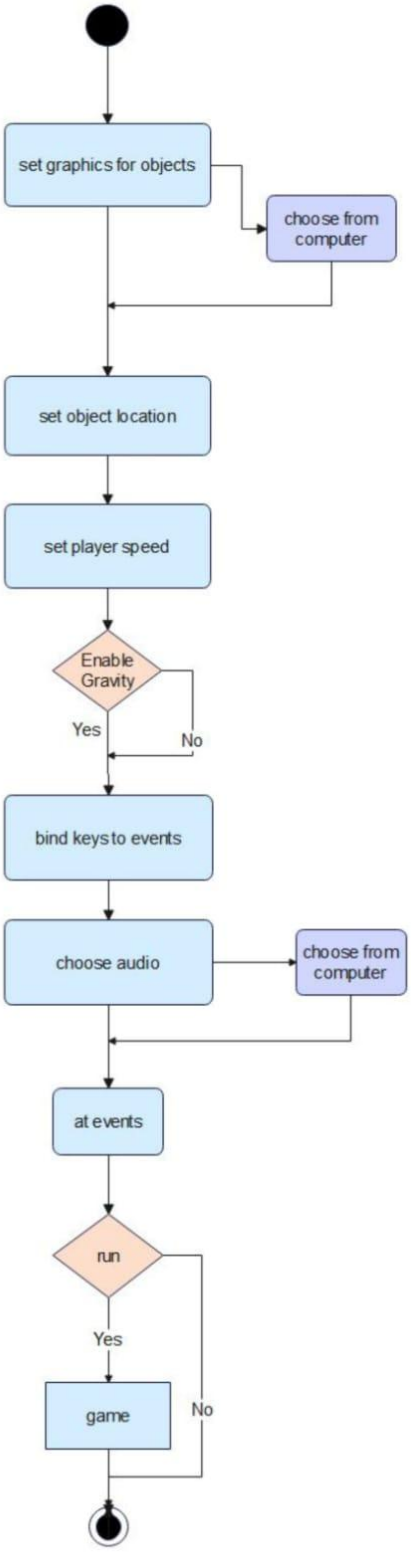


Fig 4.11 Activity Diagram for Nimble Game Maker

4.6 Test Case Design

Testing is crucial for the success of any software as no system design is ever perfect. Testings are two phases. First phase is during the software engineering which is during the module creation. Second phase is after the software is made. This is system testing which verifies that the whole set of program is compatible and working together.

1. White Box Testing

White box testing also called as clear, glass box or structural testing. It is a testing technique that evaluates the code and the internal structure of a program. White box testing involves developers/testers looking at the structure of the code. When the developer knows the internal structure of a product, tests are conducted to ensure that the internal operations performed according to the specification. And all the internal components are adequately exercised.

2. Black Box Testing

Black box testing is done when a system is tested with no prior knowledge of its internal workings. A tester provides an input, and observes the output which is generated by the system under test. This gives us a possibility of identifying how the system responds to expected and unexpected user actions, its response time, usability issues and reliability issues. Black box testing is a powerful testing technique as it exercises a system end-to-end. Along the way, what a black box test does is, it evaluates all relevant subsystems, including UI/UX, web server or application server, database, dependencies, and integrated systems.

3. Alpha Testing

Alpha Testing is a type of software testing is used to identify bugs before releasing the software product to the real users or public. This test is a type of acceptance test. The main purpose of alpha testing is to refine the software product by finding and fixing the bugs that

were not discovered through previous tests. It is referred to as an alpha testing only because it is done early on, near the end of the development of the software, and before Beta Testing.

4. Unit Testing

Unit Testing is a type of software testing. In this individual units or components of a software are tested. The purpose of this test is to validate that each unit of the software code performs as expected.

5. Integration Testing

Integration testing is defined as the process of testing the interface between two software units or modules. It's focus is on determining the correctness of the interface. The aim of integration testing is to expose faults in the interaction between integrated units.

6. Validation Testing

The purpose of validation testing is to determine if the existing system complies with the system requirements and performs the dedicated functions for which it is designed along with meeting the goals and needs of the organization.

7. System Testing

System Testing is a level of software testing in which a complete and integrated software is tested. This test aim is to evaluate the system's compliance with the specified requirements.

Chapter 5
Implementation and Testing

5.1 Code

5.1.1 Implementation Approach

Components to be implemented:

- User Interface
- Engine
- Coordinates Input
- Database Connectivity

I. First step will be creating the User Interface, for which we will use Windows Form App. of .NET Framework. We will select required UI controls and drag it onto the form, from the properties panel we will change its name and/or its behaviour to achieve our desired User Interface. The UI will then be referenced to the Engine.

II. Next step will be creating another visual studio project for our engine, this will be our core component where all the game structure code will be written. We will reference SFML in this project and start building a render window and code for our game structure. This is where the game will run.

III. Then we will create a class for Coordinates input, which will take coordinates of game objects as input from the UI and send it to the Engine, so that those objects will appear on the desired coordinate points in the game window. This class will be referenced to the engine.

IV. Finally after creating a local database with SQLite for storing game scores, we will establish a database connection by creating a class. This class will contain code for

CRUD operations (Create, Read, Update, Delete). This class will be referenced to the engine too.

5.1.2 Libraries and Packages

Let's take a look at the libraries used in building of this application:

- **SFML:** It provides a simple application programming interface (API) to various multimedia components in the computer. With SFML we can draw, render and display game objects, create splash screen and run our game.
  SFML includes 5 modules, that is, system, window, graphics, audio and network.
  But for our application, we are only going to use 4 of these:

  - **sfml-system:** It is a Base module of SFML, defining various utilities including providing vector classes, Unicode strings and conversion functions, threads and mutexes, timing classes.
  - **sfml-window:** It provides OpenGL-based windows, and abstractions for events and input handling, basically creating a render window where all the events of our game will be occurring.
  - **sfml-graphics:** This module is responsible for creating and drawing 2D shapes, texts, sprites and other entities of our game onto the render window.
  - **sfml-audio:** With this module, playing sounds from our computer system into the game will be made possible.

- **SQLite:** It is a light-weight library embedded inside software application. Here we will use it for local storage of the game scores.

5.1.3 Code

To understand the working of the developed application better, let's take a look at the code of its different components.

From Engine:

- **Importing inputs from UI and Creating Render Window:**

```
namespace Nimble_Core
{
    public class Game
    {
        public Game(// input variables as parameters)
        {
            var mode = new VideoMode(VideoMode.DesktopMode.Width,
VideoMode.DesktopMode.Height);

            RenderWindow window = new RenderWindow(mode, projectName,
Styles.Default);
```

- **Importing Graphics:**

```
// Load the game assets
            Texture backgroundTexture = new Texture(backgroundPath); //
backgroundPath
            Texture playerTexture = new Texture(playerPath); // playerPath
            Texture footerTexture = new Texture(footerPath); // footerPath
            Texture obsTexture1 = new Texture(obstaclePath); // obstaclePath
            Texture obsTexture2 = new Texture(obstaclePath); // obstaclePath
            Texture obsTexture3 = new Texture(obstaclePath); // obstaclePath
            Texture obsTexture4 = new Texture(obstaclePath); // obstaclePath
            Texture obsTexture5 = new Texture(obstaclePath); // obstaclePath
            Texture headerTexture = new Texture(headerPath); // headerPath
            Texture splashBlack = new Texture(splashPath); // splashPath
```

- **Collision:**

```
// Collision detection
                FloatRect obs_rect1 = obs1.GetGlobalBounds();
                FloatRect obs_rect2 = obs2.GetGlobalBounds();
                FloatRect obs_rect3 = obs3.GetGlobalBounds();
                FloatRect obs_rect4 = obs4.GetGlobalBounds();
                FloatRect obs_rect5 = obs5.GetGlobalBounds();
                FloatRect player_rect = player.GetGlobalBounds();
                FloatRect header_rect = header.GetGlobalBounds();
```

```
                FloatRect footer_rect = footer.GetGlobalBounds();
if ((obs_rect1.Intersects(player_rect) && obs1Checked) ||
                    (obs_rect2.Intersects(player_rect) && obs2Checked) ||
                    (obs_rect3.Intersects(player_rect) && obs3Checked) ||
                    (obs_rect4.Intersects(player_rect) && obs4Checked) ||
                    (obs_rect5.Intersects(player_rect) && obs5Checked) ||
                    (header_rect.Intersects(player_rect) && headercheckbox)
||
                    (footer_rect.Intersects(player_rect) && footercheckbox))
                {

                    RenderWindow splash_window = new RenderWindow(new
VideoMode(800, 600), "Splash Screen");

                    collisionsound.Play();
```

- **Sound:**

```
SoundBuffer    bufferstart    =    new    SoundBuffer(BufferStartPath);    //
BufferStartPath
            Sound startsound = new Sound(bufferstart);

            SoundBuffer buffercollision = new
SoundBuffer(BufferCollisionPath); // BufferCollisionPath
            Sound collisionsound = new Sound(buffercollision);



            SoundBuffer bufferend = new SoundBuffer(BufferEndPath);
            Sound endsound = new Sound(bufferend); // BufferEndPath
```

From UI:

- **Mouse Position to show coordinates:**

```
 private void field_panel_MouseMove(object sender, MouseEventArgs e)
        {
            coordinates.Text = "X: " + e.X + " Y: " + e.Y;
            X = e.X;
            Y = e.Y;
        }
```

From Coordinates:

- **Taking coordinates:**

```
public class CoordinatesPoint
    {
        public int x = 0, y = 0;
        public CoordinatesPoint(int x, int y)
        {
            this.x = x;
            this.y = y;
        }
```

From Database Connectivity:

- **Insert data:**

```
SQLiteCommand sqlite_cmd = this.sqlite_conn.CreateCommand();

                sqlite_cmd.CommandText = "INSERT INTO " + Table_Name + " (Id,
Score) VALUES(NULL, " + score + "); ";
```

- **Read data:**

```
string readData = "SELECT * FROM " + Table_Name;

                using (var cmd = new SQLiteCommand(readData,
this.sqlite_conn))
                {
                    SQLiteDataReader dataReader = cmd.ExecuteReader();

                    while (dataReader.Read())
                    {
                        getReaderData.Add(dataReader.GetInt64(0));
                        getReaderData.Add(dataReader.GetInt64(1));
                    }
```

- **Update data:**

```
using (SQLiteCommand command = this.sqlite_conn.CreateCommand())
```

```
                {
                    command.CommandText = "update " + Table_Name + "  set
Score = :score where ID=:id";
                    command.Parameters.Add("id", DbType.Int64).Value = id;
                    command.Parameters.Add("score", DbType.Int64).Value =
score;
                    command.ExecuteNonQuery();
                    command.Dispose();
                }
```

- **Delete data:**

```
using (SQLiteCommand command = this.sqlite_conn.CreateCommand())
                {
                    command.CommandText = "DELETE FROM " + Table_Name + "
WHERE id=:id";
                    command.Parameters.Add("id", DbType.Int64).Value = id;
                    command.ExecuteNonQuery();
                    command.Dispose();
```

5.2 Testing Approach

5.2.1 Unit Testing

- During the development of our application, we must perform Unit Testing.
- In Unit Testing we test the smallest testable unit of an application independently.
- This test is specifically performed by the developer to test the correctness and logic of the code.
- It was performed on small functions like render window, collision detection, object movement, game loop, etc.

5.2.2 Integration Testing

- Now that we have tested individual components of our application, we will perform Integration Testing on them.

- In Integration Testing we integrate the components together and test them as whole.
- Following steps were taken:

1. Integrating the UI with Engine
2. Integrating Coordinates Input class with Engine
3. Integrate SQLite Database with Database Connectivity
4. The connectivity class with the Engine

And finally we tested all of them as a whole.

5.2.3 Test Results

Sr no.	Test Case	Input	Expected Output	Actual Output	Pass / Fail
1	Object Location	Provide object location to be displayed on the game window	Object appears/moves to the specifies location on the Game Window	As Expected	Pass
2	Image Display	Provide object's image path that needs to be displayed on the screen (UI and Game Window)	Selected image appears on the Game Window	As Expected	Pass
3	Audio Play	Provide audio file path that needs to be	Selected audio is played in the Game Window	As Expected	Pass

		played in the Game Window at specific events	certain events		
4	Keys	Specify keys for player movement	Player movement is binded to specified keys	As Expected	Pass
5	Gravity	Enable or disable gravity for player object	Player moves downward y-axis with time	As Expected	Pass
6	Collision	Collide player with obstacles	Detects player collision with objects	As Expected	Pass
7	Speed	Specify incoming obstacle speed	Moves obstacles at specified speed	As Expected	Pass
8	Scores	Take scores based on elapsed time (in seconds)	Display scores on Splash Window	As Expected	Pass
9	Reset input fields	Event click on ‘Reset’ on File menu	Clears input fields on the UI	As Expected	Pass
10	Delete Scores	Event click on ‘Delete Scores’ on File menu	Delete scores from the database	As Expected	Pass

5.2.4 Other Tests Performed

1. White Box Testing

It is a type of testing where inner workings of the application is tested and verified. This test is usually done by the development team as it requires testers who has knowledge about the working of the application.

2. Black Box Testing

It is a type of testing where exterior workings of the application is tested. This test is usually done by testers who weren't involved in the development of the application and hence have no knowledge about the working of the application.

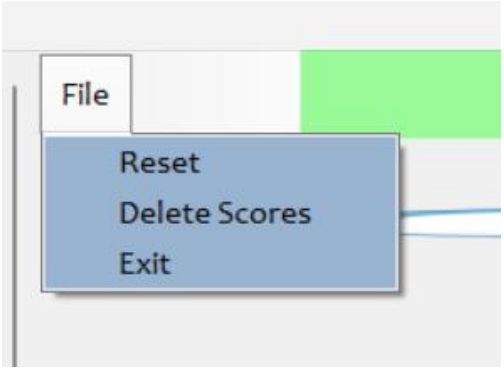
Chapter 6

Results and Discussion

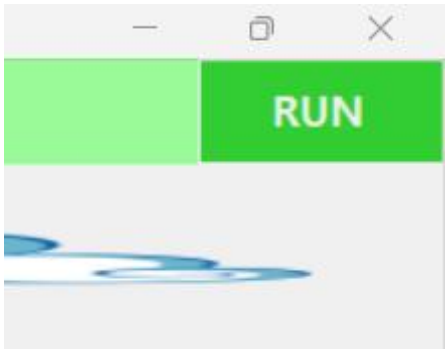
6.1 Results

Display Field

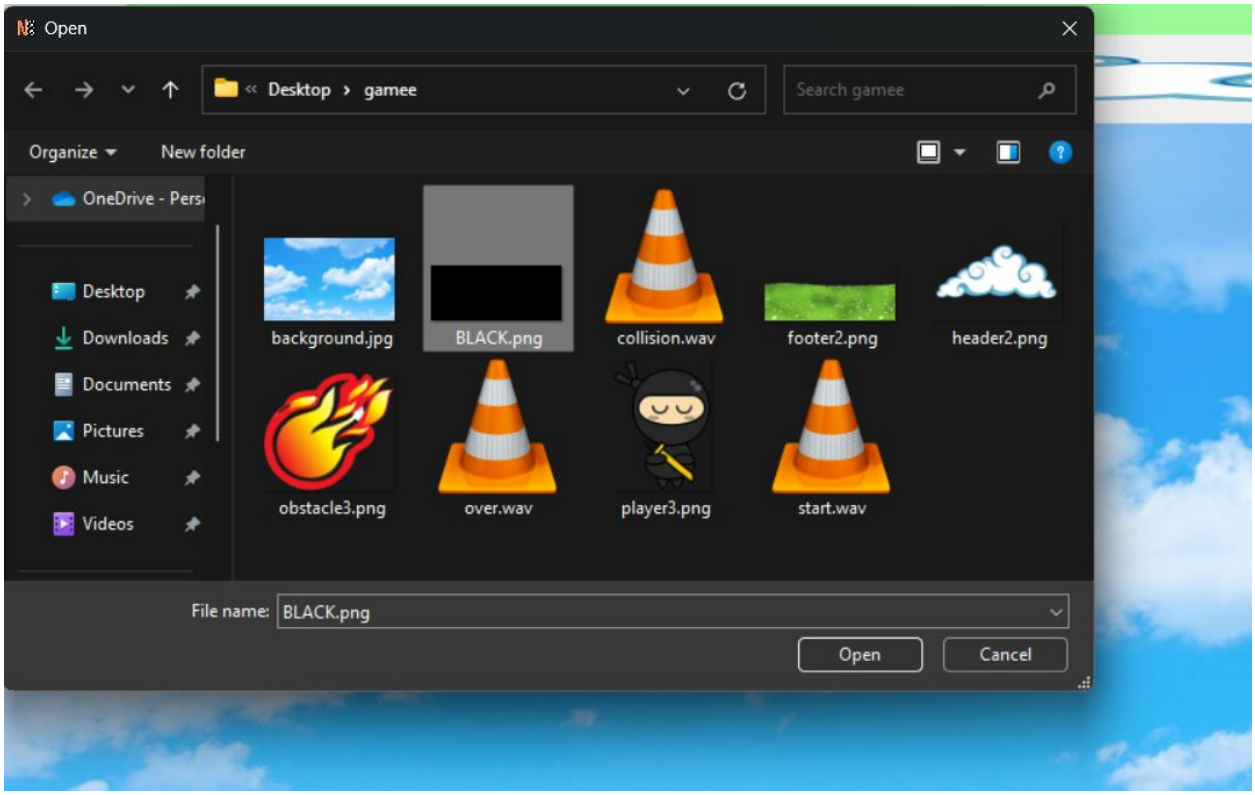
File Menu



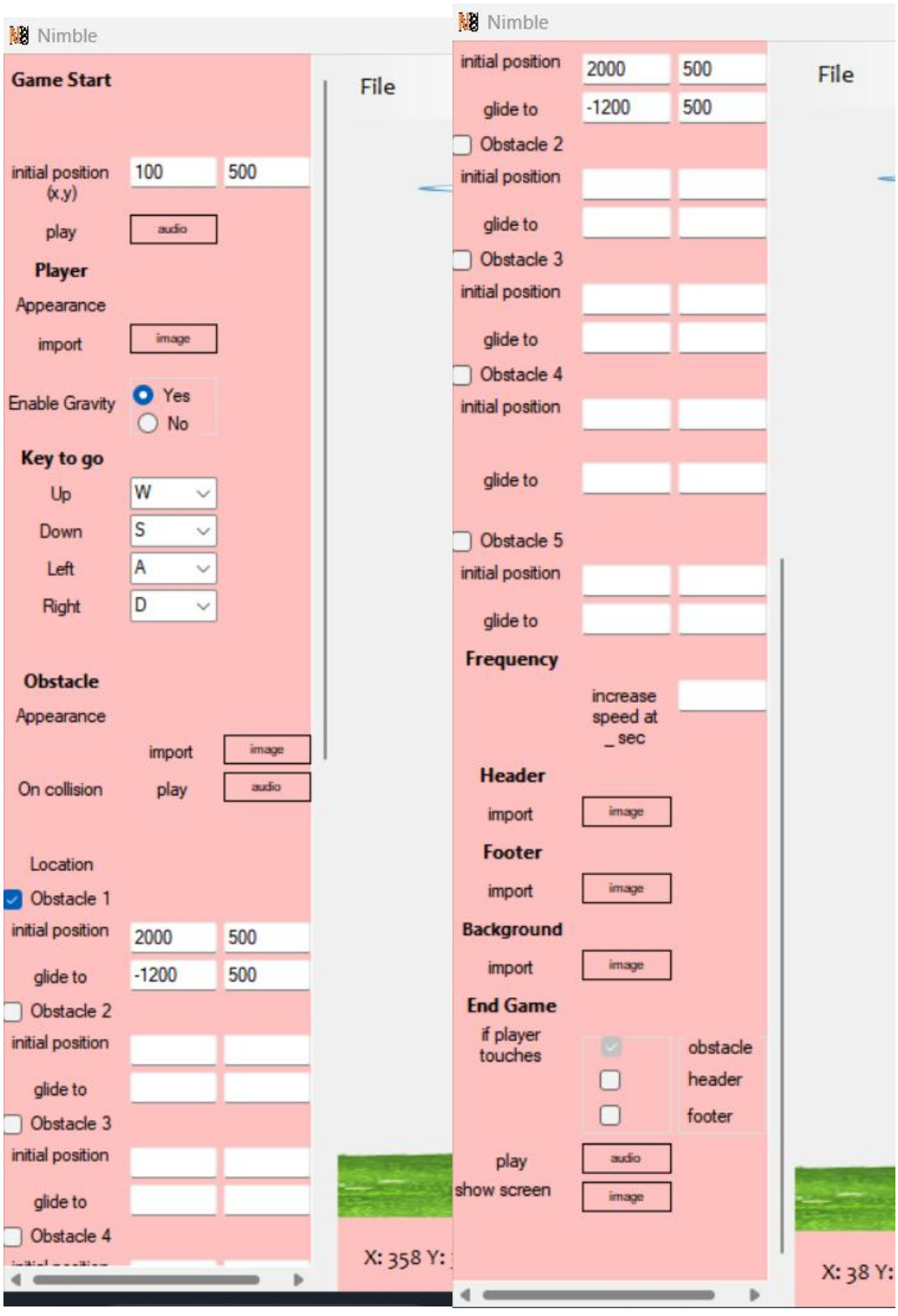
Run Button



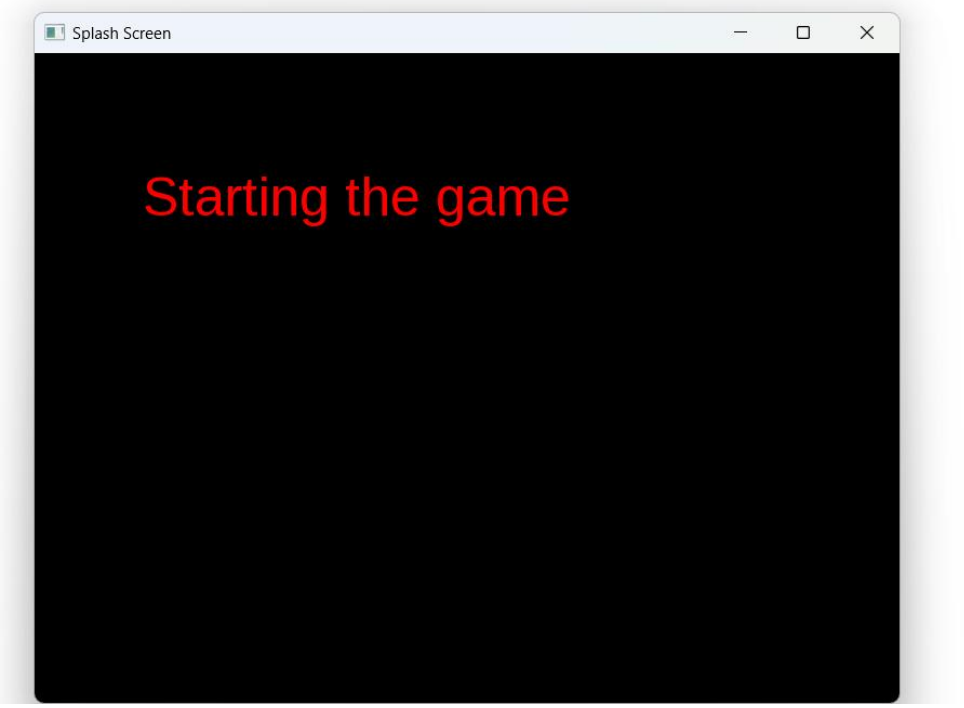
Open File Dialog Box



Properties Panel

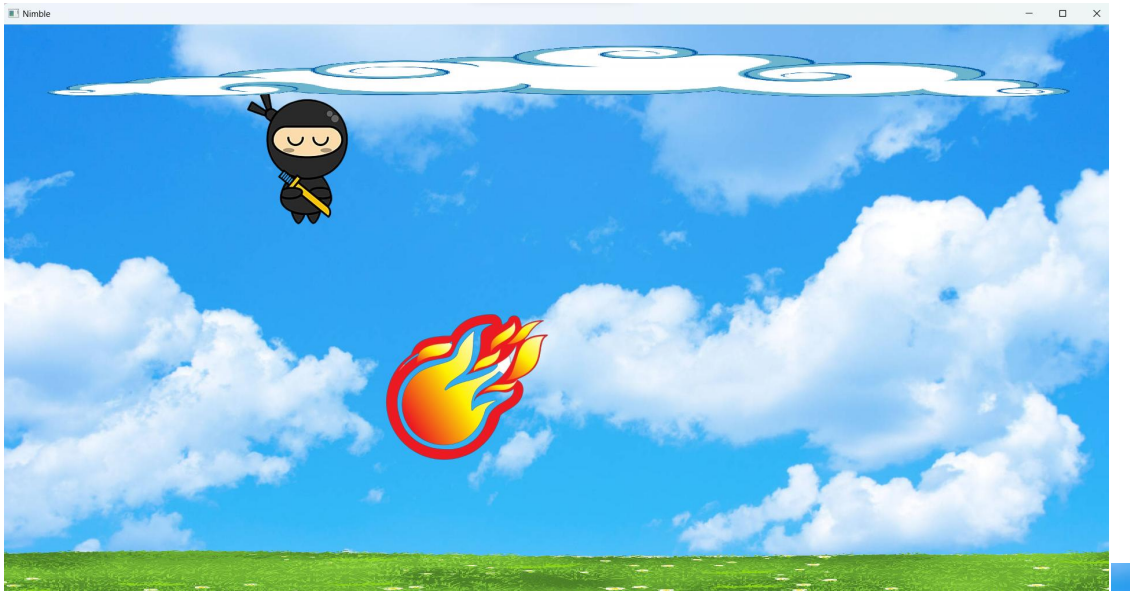


Starting game splash screen



Game Over splash screen

Game Window



6.2 Discussion

Display Field

Display field is a part of User Interface. In the display field, users will be able to preview their selected images, and also map them by coordinates with the help of coordinates provided at the footer of the screen, the points are provided by simply hovering mouse over them.

File Menu

The File Menu is placed at top-left part of the User Interface, where we are presented options to ‘Reset’ inputs in the input fields in properties panel, ‘Delete Scores’ from database so the high score of the game will be zero, and lastly ‘Exit’ with which we will be able to exit the User Interface.

Properties Panel

The Properties Panel takes the entire left panel of the User Interface. Here we will be able to select images and audio files to be used in the game, and also customize game settings as well.

Open File Dialog Box

The Open File Dialog Box appears when we click on ‘image’ or ‘audio’ button from the properties panel. We this we will be able to browse and select the specific images or audio files from our computers, and those exact files will be used in our game.

Run Button

This button is situated at the top right part of the screen. After we are done customizing game setting, a mere click on this button will execute the game i.e. it will create a game rendering window.

Start Game Splash Screen

Before the game is started, the Start Game Splash Screen will appear on our screens, along with displaying selected splash screen image file, and playing start game audio, after the completion of which Game Window will appear.

Game Window

Now a separate window for game is created. This is where the game runs. The players, obstacle, header, footer, background images will appear here, along with playing sound at collision event. The entire game will run at our specified input keys, location and speed, the ones we selected from Properties panel in UI.

Game Over Splash Screen

As soon as the player object collides with obstacle, or with header or footer if selected, the Game Over Splash Screen will appear indicating that the Game is over, along with current score and high score. Now the users will have option to either restart game by clicking on ‘Space’ button, or exit game by clicking on ‘Esc’ button.

Chapter 7
Conclusion and Future Work

7.1 Conclusion

All in all it has been a rewarding and exciting learning experience to work on this challenging project. Whilst the development of this project, I not only learnt .NET framework application with C#, SFML, local database handling, and game development, but also the entire software development process all together.It forced me to understand the need of my intended audience and expand my knowledge on different approaches to desktop application development.

7.2 Maintenance

With the deployment of this software application, the maintenance phase starts. The application will get monthly checks to see if it is working properly.
The maintenance includes:

- Fixing bugs and errors
- Check if the application is working properly
- Note any user's needs, if any

7.3 Limitations

- This game maker is only limited to endless-runner genre of platform games
- This game maker can only make 2 dimensional games
- It cannot make a separate executable file for every game that ran on the Engine so that users can play the exact created game again.

7.4 Future Works

- Expand to more 2D games genre with SFML.
- Expand to 3D games with OpenGL.
- Avail separate executable file creating option.
- User can change title bar name and logo for their game.
- Save state of the game-making process option.
- More visually appealing UI and splash screen.

Chapter 8

References

Full URLs of online references:

- https://www.sfml-dev.org/tutorials/2.5/
- https://chat.openai.com/
- https://stackoverflow.com/
- https://www.youtube.com/@thecodingfox9383/