BEHAVIRAL LEARNING GAME APPLICATION (BLG)

Wejdan Abdullah Al_shereef	435302254
Amal Awedah Al sleman	435301051
Fadiah Fahad Al_atmi	435300314

DEPARTMENT OF COMPUTER SCIENCE

COLLEGE OF COMPUTER SCIENCE AND INFORMATION

SYSTEMS

NAJRAN UNIVERSITY

April 2019

BEHAVIRAL LEARNING GAME APPLICATION (BLG)

Wejdan Abdullah Al_shereef	435302254
Amal Awedah Al sleman	435301051
Fadiah Fahad Al_atmi	435300314

Supervisor: Somaya Al_hazmi

Co-Supervisor: Maha Al wuthaynani

جامعة نجران
NAJRAN UNIVERSITY

A FINAL PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE (B.Sc.) IN COMPUTER SCIENCE

ABSTRACT

Game applications and games, in general ,are considered as one of the most beloved ways for children to spend their time. Unfortunately most of the existent video games are either inappropriate for children or has no deeper meaning beside winning the game ,and that is why we are presenting this project. The game application we are intending to develop will combine the concept of Role Playing Games with the concept of Behavioral Learning to create a Behavioral Learning Game. The game will have colorful interfaces and friendly looking characters. It will be a story based game with a meaningful message. The targeted users for this project are children ,and our goal is enabling them to learn while having fun. In conclusion all children love games, and there are many ways to make the games meaningful.

ACKNOWLEDGMENT

First and foremost we send our praises and thanks to Allah for His endless grace and mercy during this challenging time. We have put out best efforts and hopes in this project, However it would not have been possible without the support of our supervisor *Ms.Somaya Al_hazmi* and our co-supervisor *Ms.Maha Alwuthaynani* and we extend our sincere thanks and gratitude to all of faculty staff further more special thanks to *Ms.Raniah Zaheer* for her help and advises regarding the engineering aspects of the project. Thank you all and may Allah bless you.

TABLE OF CONTENT

Contents

LIST OF FIGURES

List of abbreviation

ABBREVIATION	**FULL FORM**
RPG	Role Playing Game
BL	Behavioral Learning
BLG	Behavioral Learning Game
SDK	Software Development Kit
JDK	Java Development Kit
UML	Unified Modeling Language
API	Application Programming Interface

CHAPTER ONE

INTRODUCTION

CHAPTER ONE

INTRODUCTION

1.1. Introduction:

Nowadays technologies and machines are everywhere. They affect our lives and how we behave and interact with our environment, knowing that it was necessary to create a modern ways to drive the new generation to learn.

The traditional methods of teaching children how to behave focus giving orders to the children and them obeying instead of developing an observant mind that can recognize what is wrong and what is right, how to act in a certain situations ,and how to treat other people.

In order to fix that and improve the children's abilities of solving problems along side with presenting them a new space to educate themselves we are presenting this project.

In this project we aim to deliver a new way of presenting games. We will be using the concept of Role Playing Games (RPGs) that gives the player the ability of controlling characters in a virtual world [1], together with the concept of Behavioral Learning(BL) which means the player could learn without having the intention to learn something, and without conscious [2]. Combining these concepts we will present a Behavioral Learning Game(BLG).

The game will support the learning process, learning through observation and experiencing different situations in a virtual world. Our game will bring the gracefulness of the old children stories back by creating a virtual world out of these stories. A world where the child can explore and be a part of.

This project will be developed for children within the age group(6-12) years old. The target platforms are PCs and Android devices, meaning this game application can be installed on mobile devices, other Android devices and personal computers of the children , furthermore can be installed on the computers of educational institutions.

In this chapter, we will define the project by a describing the problem, presenting the features of the application, presenting the objectives of the project, determining the project's scope, discussing the limitations of this project.

1.2. problem statement:

General problem:

The current game applications are mostly divided into two main categories, Games for adults which are entertaining and addictive but absolutely inappropriate for children. Games for children mostly educational that focus on direct deliverance of information, or entertaining games with no deeper meaning or real purposes beside having fun.

Specific problem:

We want to create a game that has a meaningful story line. That said our specific problem is how are we going to convert our story into RPG game.

1.3. Creating the idea :

At the beginning we considered making an education game based on the concept of RPGs, however after searching and reading more about the current educational games and their flaws. We decided to combine between two concepts that is RPGs and behavioral learning.

The game will be built by using the methods of RPGs, while the challenges and the story support behavioral learning.

1.4. System description:

This application will be built with one type of users in mind that is children.

They can access the application easily and use it, enjoy themselves and learn at the same time, without feeling the pressure of having to score a specific grade or worrying about being evaluated on their work, they will be entertained while stimulating their imagination and creative side.

1.5. Features of the application:

This behavioral learning game is an application software that combines the characteristic of the modern applications with the charm of the vintage stories. It will be a game with one player mood.

This game application is a story based game, where the player can control the main character (Moon).

It is developed to deliver a virtual experience that catches the interest of the users, stimulate their imagination, push them to make decisions, let them interact with the game's characters, and observe what is happening around them. Through the game the player should collect some items in order to win the game. By the end of the game the player will hopefully learn indirectly while having fun.

This application software can be installed on the schools' computers and users' personal devices, thus they will be able to use it at their free time.

1.6. The objectives and aims:

The aim of this project is to deliver a new way of learning, and to attract as many people as possible to play this game, not only that but also getting the permission to install it in the educational institutions’ computer devices.

We set the following objectives that will help us to achieve these aims:

Final product objectives :

Developing a software system (RPG) that supports Behavioral and Creative Learning that could catch the children attention and drive them to learn indirectly while enjoying their time.

Introducing some issues the children have to deal with in our society, and how can they overcome these issues.

Personal objective:

Studying System analysis and Design based on the requirements of behavioral learning and game development.

1.7. The scope of the project

The game application we are intending to create will be created based on the concept of Role Playing Games combined together with the concept of Behavioral Learning.

The game will contain only three levels. It will be a story based game with a main character which is the only character that could be controlled by the player goes with the name (Moon).

All the dialogs in the application will be written in English. Our game will be offline game with one player mode.

1.8. limitations of the project:

Time Limitation :

We have a fixed time. As a result of that the game will only contain three levels.

Recourse Limitation :

We do not have a prior reference to look at since our game will be the first game built in our university.

Language Limitation :

The support of Arabic language in the game engine we will be using to develop the game is still on its early stages which will prevent us from developing the game for Arab users.

Software and hardware Limitation :

We will be using our personal computers to develop the game since we do not have the budget to buy any advance devices.

1.9. Report outlines:

In the rest of this thesis 6 chapters will be discussed as following:

Chapter 1: introduction

In this chapter we will define the concept of the project, a description of the problem, features of the application, the objectives and aims of the project, and the project's scope.

Chapter 2: Theoretical background

This chapter will include the background study of the project, evaluating the similar systems, problems in similar systems, features of the purposed system, and conclusion.

Chapter 3: Method of Investigation and Analysis

This chapter illustrates project methodology, requirements analysis, game design analysis, UML diagrams including use cases diagram, class diagram, sequence diagram, activity diagram.

Chapter 4: System Design

This chapter the system's interfaces, the game's data, along with the technical and artistic stages for creating our game will be presented and discussed.

Chapter 5: Testing

In this chapter we will discussed the testing process, how tested the functionality of our system and how it supposed to flow from one window to another.

Chapter 6: Conclusion and future work

This chapter will include conclusion and future work for this documentation.

CHAPTER TWO

THEORETICAL BACKGROUND

CHAPTER TWO
THEORETICAL BACKGROUND

2.1 Introduction:

In this chapter, we will discuss some information about similar existing systems, we will examine the advantages and disadvantages of the existing systems, then we will introduce solutions to these disadvantages through developing a new system.

In addition, we will distribute a questionnaire for this application, that contains a number of questions related to the targeted users, and ask them to answer the questions ,after that we will analyze the results.

The purpose of this chapter is to discover the problems in the existing systems to avoid them when we develop our game application.

2.2 background study:

Upon searching for a game application used in the educational institutions in the Kingdom of Saudi Arabia we could not find any application that uses the concept of behaviorism.

First, we present a basic background study as shown below:

	Educational games for kids	Tree world	Fortnite
Dose it address children related issues ?	No	No	No
Is it appropriate for children ?	Yes	Yes	No
Dose it contain educational content ?	Yes	Yes	No
Dose it supports self-growth ?	No	No	No

<u>Educational games for kids application :</u>

Your kids will learn alphabet, numbers, colors, shapes, days of the week, months of the year, planets of our solar system, space and much more.

Our educational game shows children the alphabet letters and teaches them to recognize letters as they appear. As a result, preschoolers kids learn the letters sounds much faster [3].

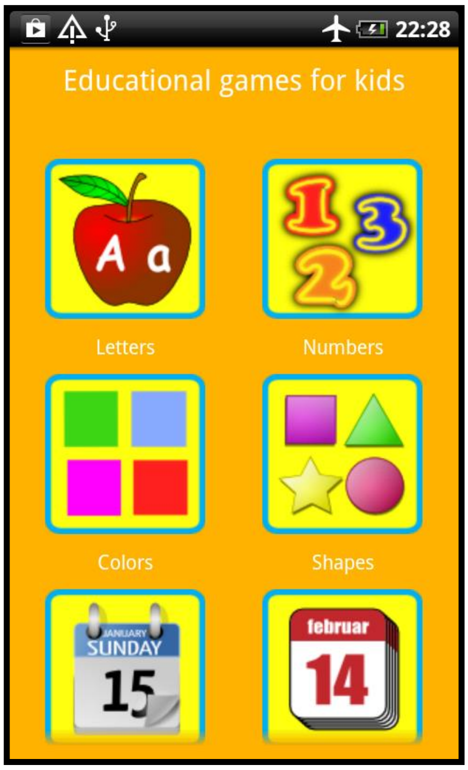


Fig 2.1: Educational Games for Kids Interface

Advantages:

- The amount and diversity of the information presented in the game.
- The colorful interface.

Disadvantages:

- There is no appropriate challenges that stimulate the player's creativeness.
- Lack of imaginations.
- No enthusiasm while playing the game.

<u>Tree World:</u>

Grow your tree to the sky and make it a home for adorable critters

learn all type of different animals. [4]

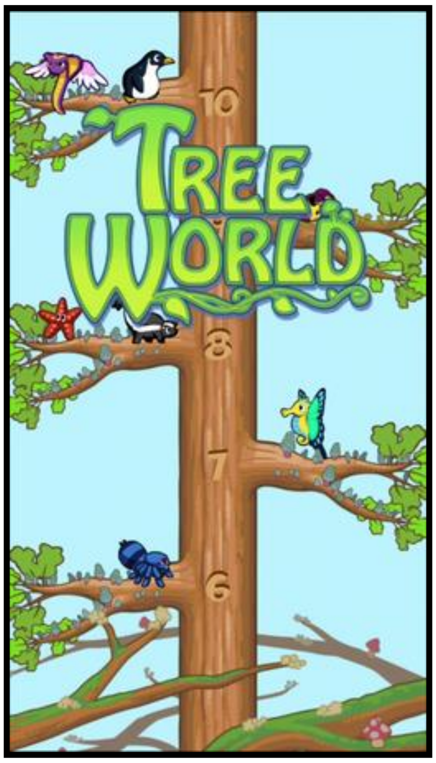


Fig 2.2: Tree World Interface

Advantages:

- The catchy interface.
- The variation of characters.
- New kind of challenges.

Disadvantages :

- There is no story-line to be followed.
- Only one kind of information will be learned.
- The way of playing is limited.

Fortnite :

Two large teams fight for the Victory Royale in an action-packed mode where the first team to get 100 eliminations wins!.[5]

FORTNITE

Fig 2.3: Fortnite Logo

<u>Not to be confused:</u>

Fortnite is a great game we do not doubt that, the aspect we are discussing is that the game is not appropriate for children as it is clearly stated in the game rating.

Fortnite rating:

In the UK the Video Standards council rate Fortnite as PEGI 12 (Pan European Game Information) for frequent scenes of mild violence. It is not suitable for persons under 12 years of age.[6]

Advantages:

- The high quality graphics and controls.
- The various challenges and missions.
- It is portable to many platforms .

Disadvantages:

- The amount of violence the game contains.
- The game is very addictive, children may become very addicted to it in some cases.

2.3 Questionnaire:

We have made and distributed a questionnaire to find more about children's opinions, knowledge of games and the connection that they have to smart devices and games in general.

This survey has been written in Arabic and analyzed in English, the following charts show the results of the survey that reflects 20 kids' opinion:

Question one: how do you prefer to spend your time ?

We found that 35% prefer to spend their time using smart phones equal to 35% that prefer watching TV , and the rest vary in their answers.

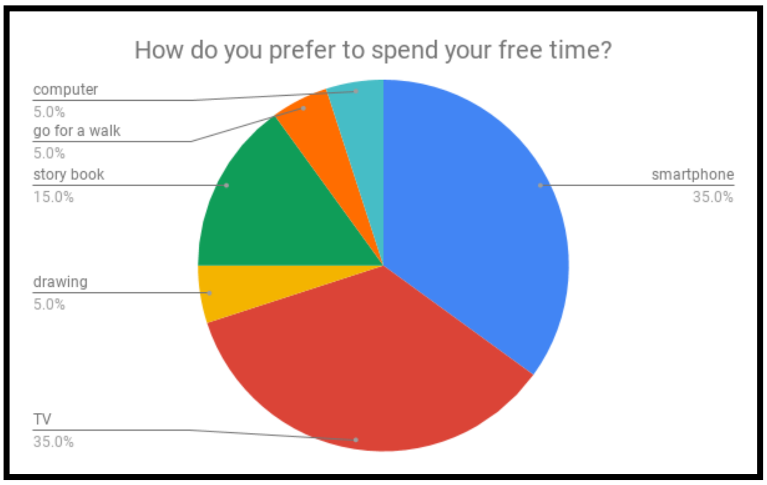


Fig 2.4: Analyzing Answer of Question One

Question two : How many hours you spend using smart phones daily?

55% spend more than three hours using smart devices.

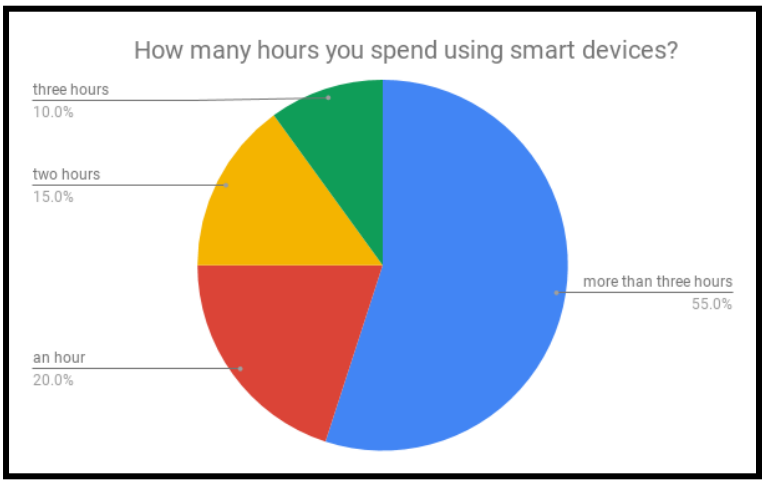


Fig 2.5: Analyzing Answer of Question Two

Question three :do you know what are video games are ?

Yes: 90%   No:10%

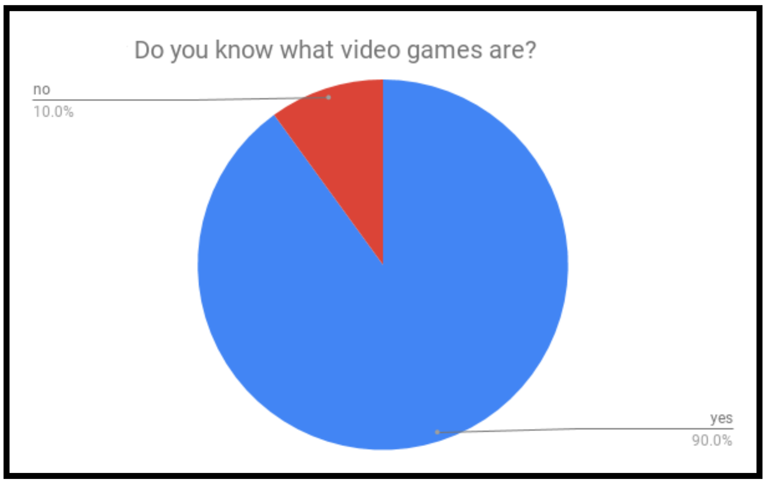


Fig 2.6: Analyzing Answer of Question Three

Question four: do you remember the last game you played ?

Yes : 70% No:30%

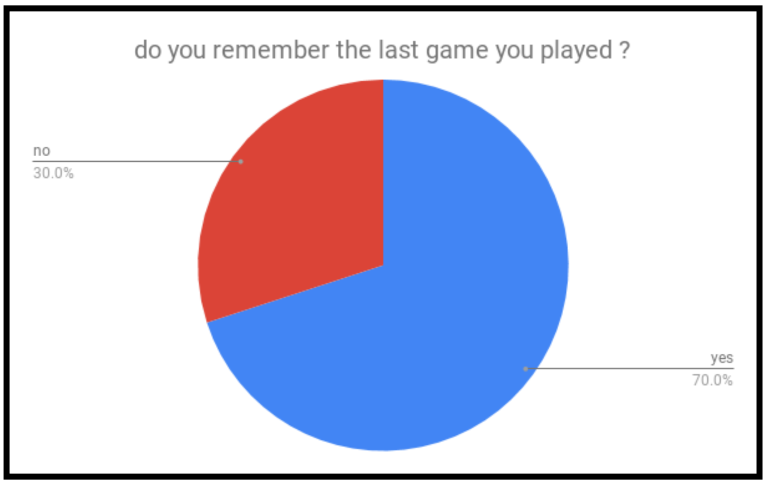


Fig 2.7: Analyzing Answer of Question Four

Question five :if you answered yes to question four , can you name at least three characters from that game ?

Yes : 65% No:35%

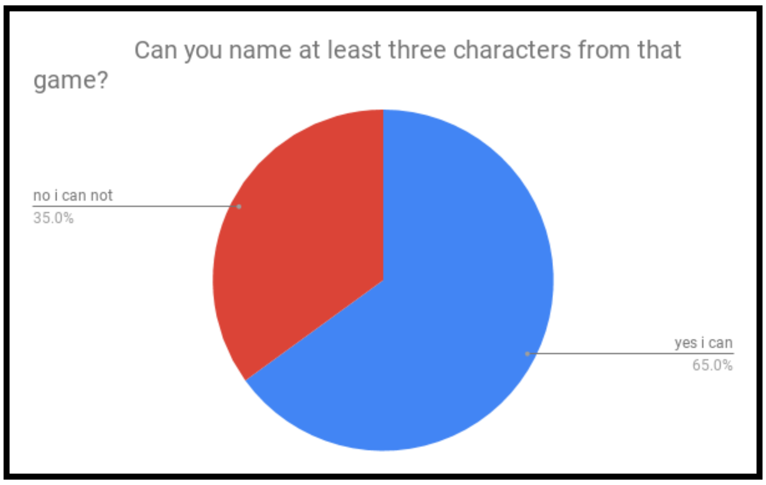


Fig 2.8: Analyzing Answer of Question Five

Question six :you have two options …..

1. read a school book :40%
2. play a game :60%

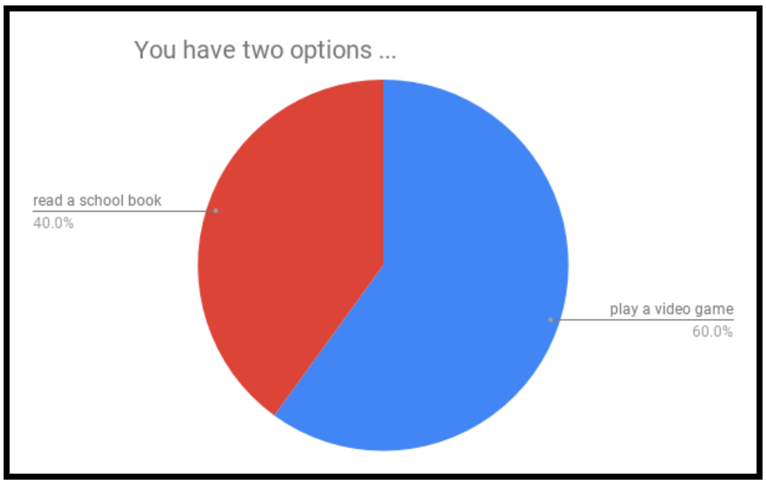


Fig 2.9: Analyzing Answer of Question Six

The data we collected prove the following points:

- The new generation is strongly connected to the smart devices, and the they spend a lot of their time using them.
- The smart devices hold great possibilities and can provide new ways of learning.
- The new generation has a lot of knowledge about video games as they play them a lot.
- In general, children will always want to explore, play and have fun regardless of the time, generation or place.

2.4 Drawbacks in the existing systems:

- Majority of games that are rated for children lack a main storyline.
- Lack of challenges and imagination.
- Educational games focus on the same information that was given in the class, and does not expand the player's creativeness.
- Majority of children games does not address children related issues that exist in the society.
- Some games can be inappropriate for children.

2.5 Features of the proposed system:

The proposed system is a game application; this application will introduce a new method of learning, that will support the educational field.

The learning process will accrue While the players entertain themselves.

The game will use the concept of behaviorism,by applying it over RPG game.

The game will have a meaningful story.

The game will have a simple and catchy interfaces.

<u>The features of BLG include:</u>

- Easy to understand and use.
- The game will work offline.
- One player mode.
- Collect stones to pass a level.
- Interacting with the other characters, that represent the story elements.
- Users will be able to control only the main character.

Advantages of the proposed system:

- Children will be able to use it at their free time.
- Children will be able to learn while having fun.
- The story complexity can be increased to support older audience.
- Using graphical image processing, to catch the attention of the children.
- The story and its characters are designed to match the children's taste.
- The game will contain different challenges and quests that will stimulate the child creative and logical thinking.

2.6 conclusion

In this chapter, we discussed a few similar existing systems to our project. Also, we clarified the results of the survey we distributed, and analyzed it based on the results we got. We mentioned some of the existing similar systems' drawbacks, which we will be trying to avoid in our game application. Lastly, we declared the main features of the proposed system.

CHAPTER THREE

METHOD OF INVESTIGATION AND ANALYSIS

CHAPTER THREE
METHOD OF INVESTIGATION AND ANALYSIS

3.1 Introduction:

This chapter deals with the detailed analysis of the application by, firstly selecting a methodology to be followed, and planning for the phases and steps.

Secondly requirements analysis (functional and nonfunctional) will be presented. Thirdly presenting the test plan that will be followed to verify the system while discussing the risks that may arise and the solutions to be followed.

After that we will be presenting the use of UML diagrams that will enable us to specify, visualize and construct the application. At the end of this chapter a separate section will be included about the API together with its functions and diagrams. As for the technical documentation it will be discussed in Project 2.

3.2 Project Methodology:

Project Management Methodology is a strictly defined combination of logically related practices, methods, and processes that determine how to plan, develop, control and deliver a project in the best way possible throughout the continuous implementation process until a successful completion and termination is achieved . It is a scientifically-proven, systematic and disciplined approach to project design, execution, and completion. [7]

3.2.1 Software Development Methodology:

In this project we will be applying the waterfall model as the project's methodology for designing and implementing. The waterfall model is a sequential design process, used in software development processes, each phase must be completed before the next phase can begin and there is no overlapping in the phases, the whole process of software development is divided into separate phases, in which progress is seen as flowing steadily downwards through these phases: [8]

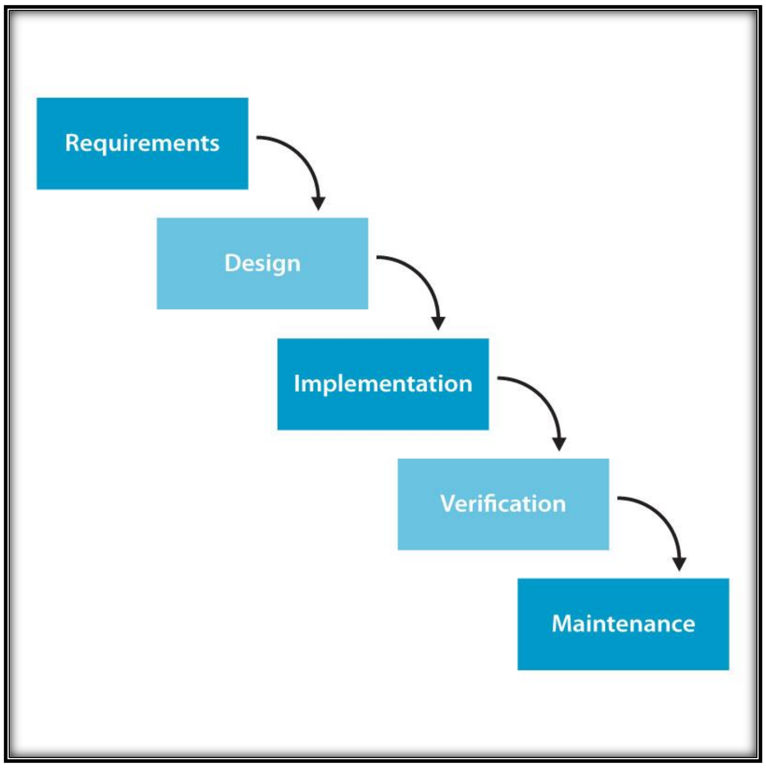


Fig 3.1: Methodology (Waterfall)

❖ **Requirements**

This is the first phase of development where all the requirements are gathered and documented. In this phase we collect the data that will be used for constructing the application.

❖ **Analysis**

This phase contains analyzing all the gathered requirements whether the requirements are valid or not. We analyze the data that were collected in the previous step.

❖ **Design**

In this phase, all the system designs are analyzed and specified like hardware system configuration, and architecture. Moreover we design the menus, the interactions between the elements of the game, the graphical units, and build the different levels of the project according to the analyzed data and requirements from the previous steps.

❖ **Implementation**

This phase involves, performing all the development work, and development components after that the levels are handed over to testing.

❖ **Verification and maintenance**

Testing phase starts by testing each level or component, and maintaining the problems that may appear during the testing operation.

Advantages of waterfall model:

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- The start and end point for each phase is set, which make it easy to measure the progress.
- The tasks remain as stable as possible throughout the development process.

3.3 Tools Used:

To write this report, we used the following tools :

❖ **Microsoft Word 2016**

Microsoft word is a suitable tool to write and chick the grammar of the report and asserting the layout and fonts of the report.

❖ **Star UML**

The software we used to draw the UML diagrams.

3.4 Feasibility Study:

A feasibility study is an analysis used in measuring the ability and acceptance to complete a project successfully including all relevant factors. [9]

Feasibility study enables us to evaluate the project and find the strength and weakness points by studying the following feasibility aspects:

3.4.1 Operational Feasibility

The BLG application will have friendly interfaces and simple menus. Ones the new users can use and understand easily, meaning it will be suitable for all types of users.

3.4.2 Technical Feasibility

To implement the application, we need a specific software and hardware. The devices and applications we need in order to build the game are available for free and we can find them easily.

3.4.3 Social Feasibility

The BLG will support many aspects like educational process, entertainment and self-growth, all of which will be achieved through observation and experiencing different situations in a virtual game world.

3.4.4 Economic Feasibility

To build this application and attract users to use it, we do not need to purchase anything. Therefore, from an economic point of view, this application is not a loss for the user as it will be available for free and it will not need a special hardware to run it.

3.5 Hardware / software requirements :

3.5.1 Hardware Requirement:

- ❖ personal computer.
- ❖ CPU: core i7 -7500.
- ❖ Microsoft® Windows® 10 (64-bit).
- ❖ 8 GB RAM minimum, 12 GB RAM recommended.
- ❖ HDD 500 GB.
- ❖ Android devices.
- ❖ Wacom tablet.

3.5.2 Software Requirement:

❖ Unity engine

Unity is a game engine and integrated development environment (IDE) for creating interactive media, typically video games. [10] Unity gives users the ability to create both 2D and 3D games, and the engine offers a primary scripting API in C#.

❖ software development kit (SDK)

a set of tools used for developing applications provided by hardware and software providers.

❖ java development kit(JDK)

This library of JAVA that has functions and procedures of language.

❖ android studio

This program used as editor to convert the application to the Android devices.

❖ media pang

This program used to create the designs and assets for the game.

❖ fire alpaca

This program used as Painting and drawing software and image editor as well as creating the animation sheets.

❖ visual studio

This program used as editor to write the codes.

3.6 Requirement Specifications:

3.6.1 Functional Requirements

Functional requirements describe the behavior of the application and specify what the application should do.

The user functionalities:

- A player can view information on how to play the game. Since the game will be made for children the information will be written in a simple and fun way.
- The player can view the credits if they want to know more about the creators.
- A player can start a new game or load a saved game from a checkpoint.
- A player can pause the game then continue it.
- A player can quite the game from the pause menu.
- A player can view the collected items.
- A player can win the level or win the game.
- A player may fail the level then start from a checkpoint or restart the level.
- A player will be able to control the main character.
- A player will interact with different story elements.

The assets functionalities:

- The camera will follow the player.
- The main character will be controlled by the player.
- Different elements will be collectible for the player.

3.6.2 Non-Functional Requirements

Non-functional requirements describe how the application should behave and specify how the application performs a certain function.

❖ **Performance:**

The system should be easy to use, and capable of performing its operations in a short time.

❖ **Reliability:**

The application should perform all of its tasks in a correct and satisfying way.

- **Safety and Security :**

  The ability to secure the application and provide protection to the users.
  - The system works offline, that makes it more secure.
  - Only the owner of the device can access the game.
  - Updates and modifications will be happening through reliable sources.
- **Usability:**

  With the design of simple and creative interfaces and menus the new users will be able to use the application easily and smoothly.
- **Availability:**

  This application can be easily downloaded and installed on Android device and personal computers.

3.7 Test plan:

The test plan is a document that describes the software testing scope and activities. It is the basis for formally testing any software/product in a project. [11]

In our test plan we will focus on the next four points:

- **Unit test plan**

  In this step, we will test each level and the components of that level.
- **Integration test plan**

  In this step we will test the integration of two levels or more, how the control flows from one level to the next, also we are going to check if there are any components in need of redesigning or modifying.
- **System test plan**

  In this step, we test the application as a whole and how it functions on the target platforms.
- **Acceptance test plan**

  In this last step, we test the acceptance of the application, where we run a user test which demonstrates the application ability to meet the original objectives and system requirements.

3.8 Risks

In this step we discuss the risks that we may face and present the solutions that we will implement in each case:

- ❖ **Running out of time**

We have already started planning and designing the assets and the interfaces in order to save us some time that can be used in coding.

- ❖ **Problems with coding and graphics.**

That problem could be avoided by reading and examining similar projects alongside with studying the process of building them, while at the same time train ourselves in writing codes using C#.

- ❖ **We do not have experience in uploading applications on the market, that may result in our game to be left on the shelf.**

To solve this problem we started reading and discussing the ways and methods of marketing games and publishing them.

3.9 Use Case diagram:

Use case diagram is usually referred to as behavior diagrams used to describe a set of actions (use case) that the system should perform in collaboration with one or more external users (actors). [12]

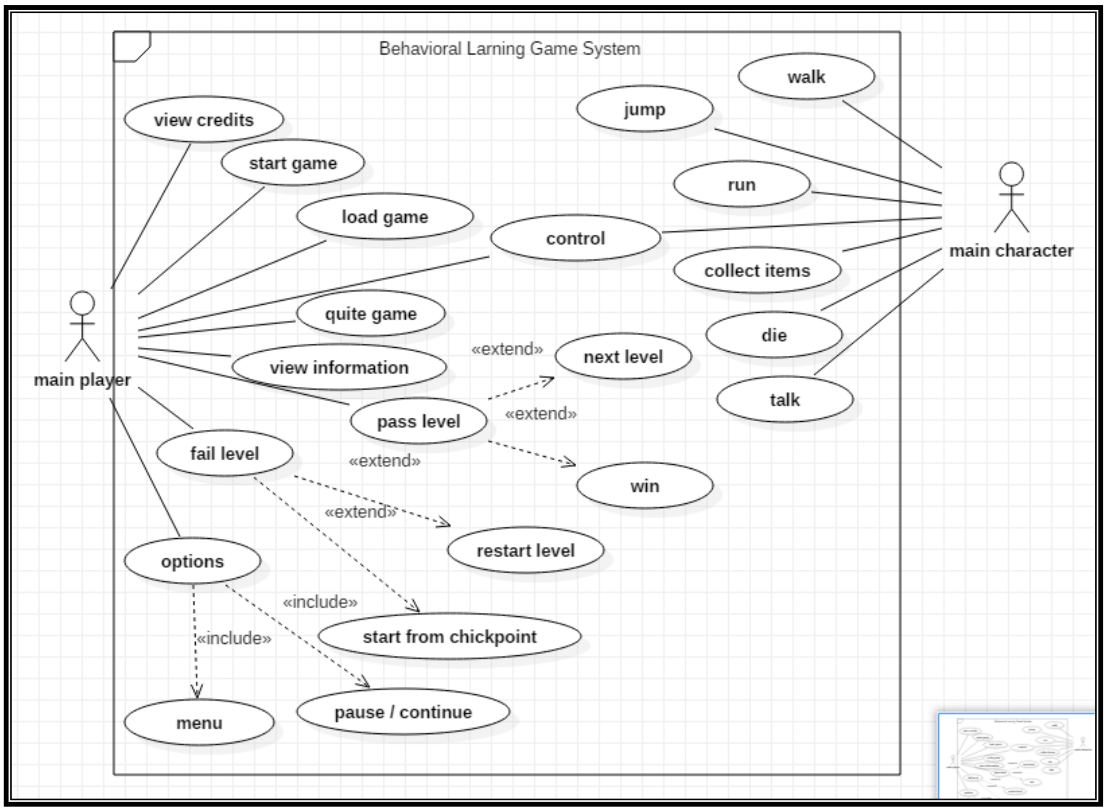


Fig 3.2: Use Case Diagram

3.10 Use Case Description:

In this diagram, we have one user that is the player as shown above, and the user can interact with the system by the following actions:

❖ Starting a new game case:

After starting the application the user can start a new game, where in the case of choosing a start game icon the first level will be automatically loaded.

❖ Loading a saved game case:

The player can start from a saved point if they have played the game before.

❖ View information case:

If the player was curious about how to play the game, how to win or what may cause losing the game they can press view the information icon.

❖ **View credits case:**

If the payer wanted to know more about the creators and the building of the game they can view the credits by pressing on its button.

❖ **Quitting the game case:**

The player can quit the game at any level.

❖ **Controlling the main character case:**

The player can only control the main character through different levels of the game, and the different functions for the main character is a part of this function where the main player will control the main character to do the following:

- Walk.
- Jump.
- Run.
- Die, in specific cases.
- Collect items.
- Talk .

These functionalities will be discussed more in depth in the section 3.14 Technical documentation:

❖ **Passing a level case:**

The player can pass levels by completing the required challenges and passing levels will have two extended cases:

- In case the level was not the final level, the player will pass to the next level.
- In case the level was the final level, the player will win the game.

❖ **Failing the level case:**

The player may fail the levels due to not completing the required challenges to pass and in this case, the player will have one of two options:

- The player can start from a previous checkpoint.
- The player can restart the level.

❖ **Options case:**

The player will have two options inside the level itself:

- Pause option: Where the player can pause the game, then continue when they want.
- Menu option: Where the player can view the collected items from the items' menu.

3.11 Activity diagram:

The activity diagram is used to describe the dynamic aspect of the application, it represents the flow from one activity to another.

3.11.1 Opening the game application:

Initially the user will open the game application then will have the options shown in the main interface of the game, the user can start a new game or load a saved one from a checkpoint and if they wanted to know the basic information of how to win a level or fail one they can view the information document or if they wanted to know more about the creators they can view the credits option as shown below:

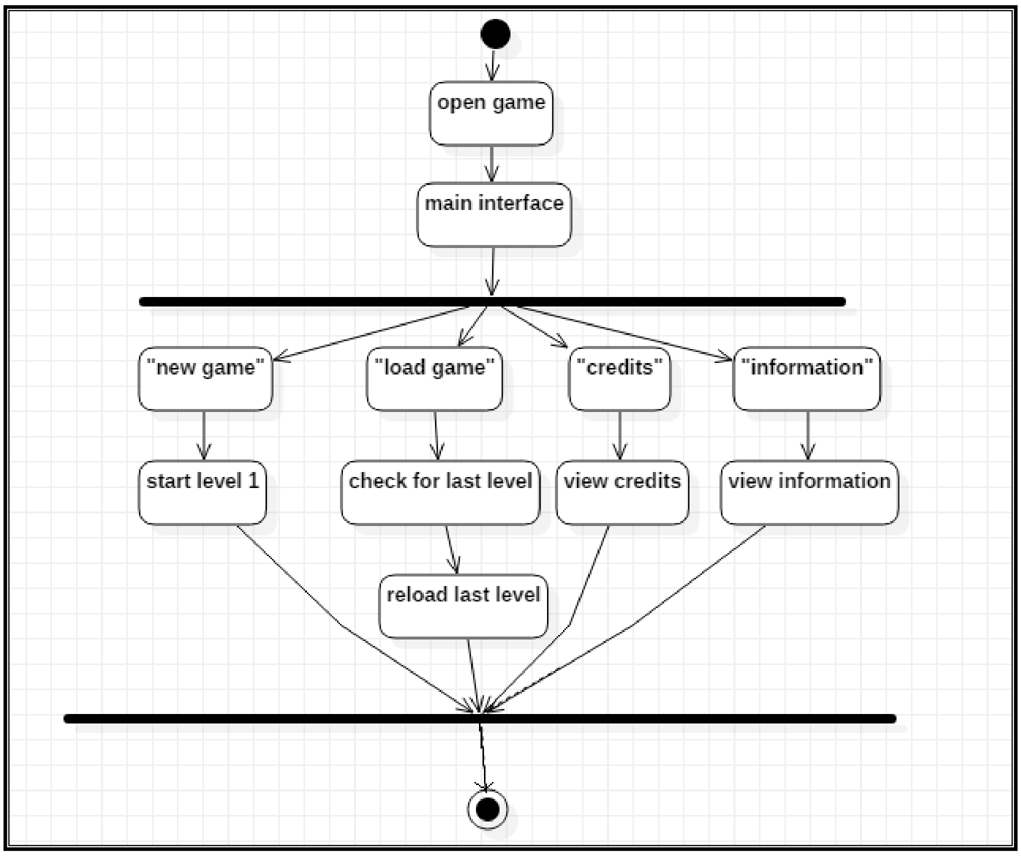


Fig 3.3: Activity Diagram for Open Game

3.11.2 New game:

The user can start a new game from [start game]option where the first level will be loaded to be played as shown below:

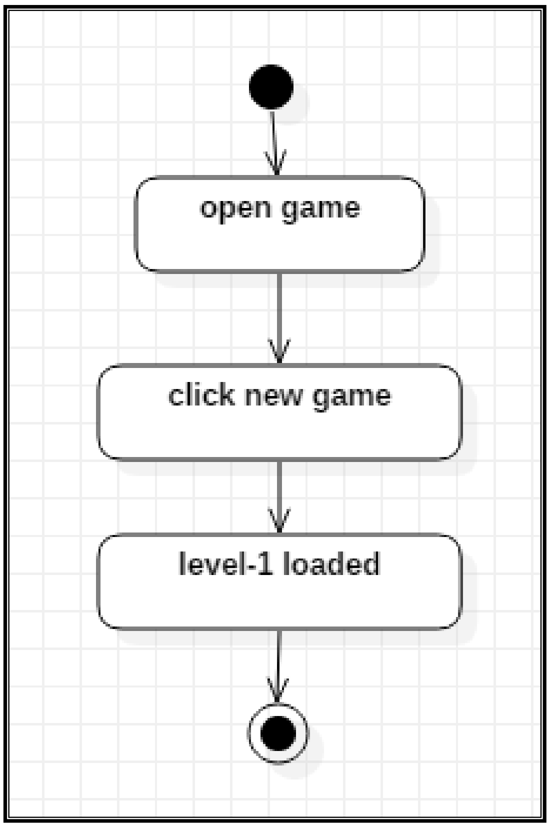


Fig 3.4: Activity Diagram for New Game

3.11.3 Pass/ fail a level:

The Player will start the level and play it ,when reaching the end a check process will be running to determine if the player has passed the level or not. If the player passes the level they will be moved to the next level but if the level was the last level they will win the game, in the other hand if they did not pass the level they will start again as shown in the diagram:

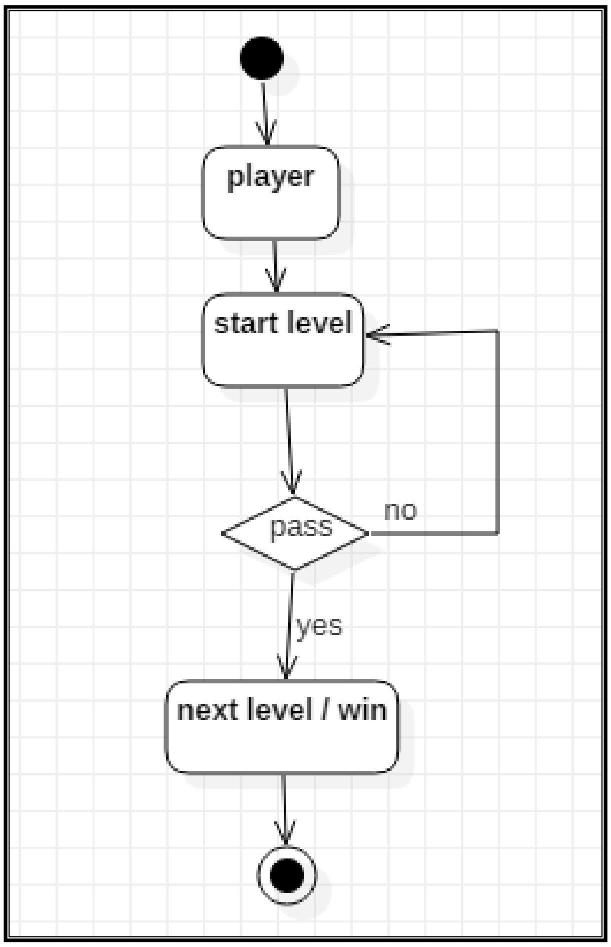


Fig 3.5: Activity Diagram for Pass/Fail Level

3.11.4 Load a game:

The player can load a saved game from a checkpoint, which means they will not have to start the game from the beginning as shown below:

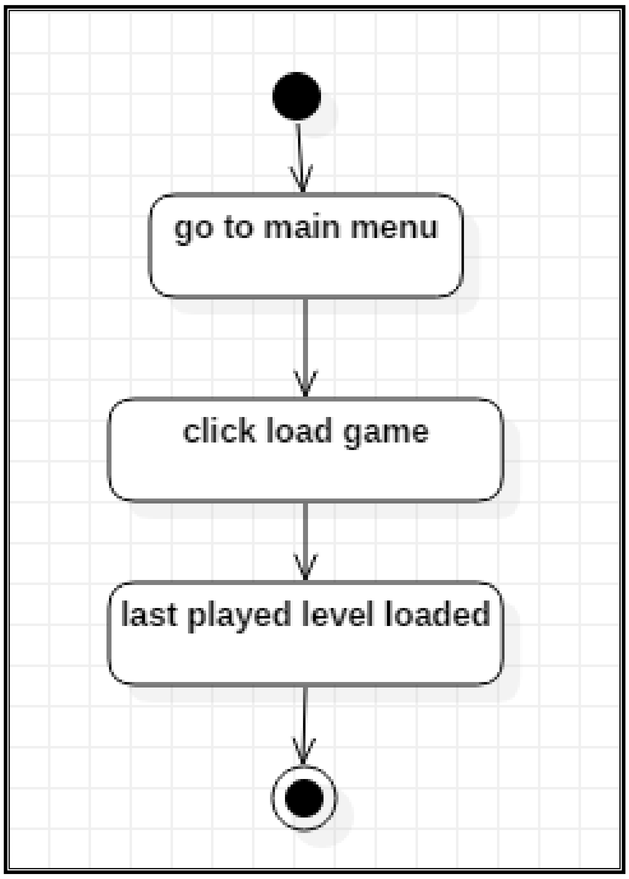


Fig 3.6:Activity Diagram for Loading Game

3.11.5 view information:

The player can view the levels' information and how to win them from the information option as shown below:

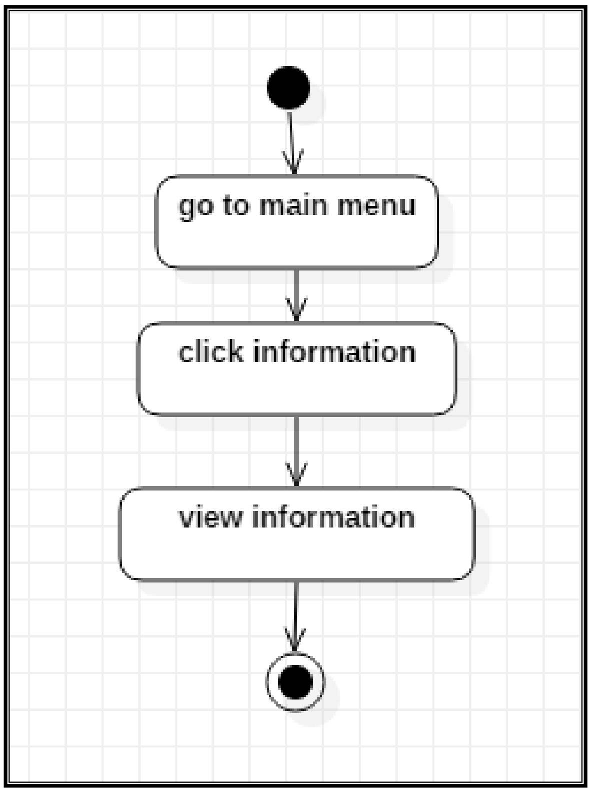


Fig 3.7: Activity Diagram for See Information

3.11.6 Exit the game:

The player can exit the game from inside the level itself. This option can be accessed from the pause menu as shown below:

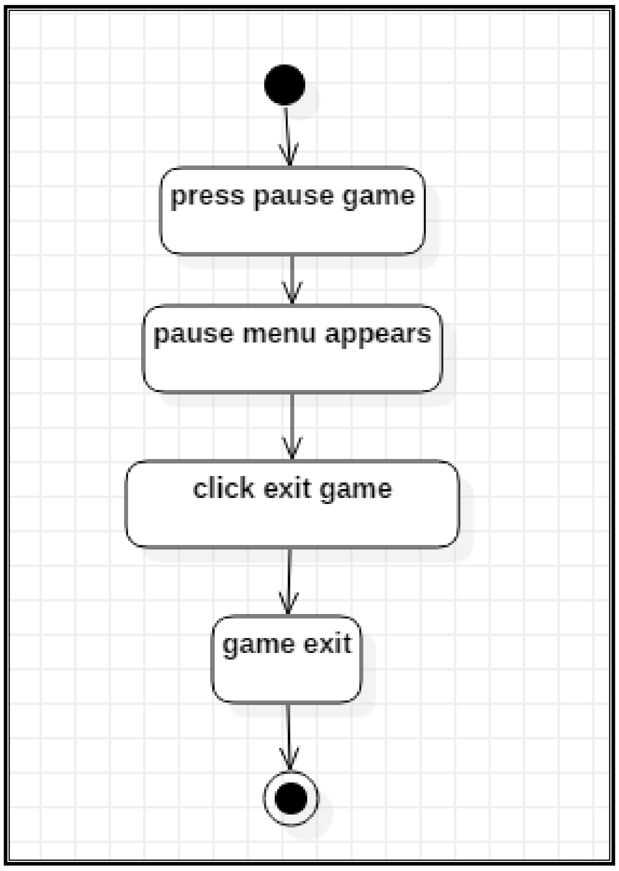


Fig 3.8: Activity Diagram for Exit Game

3.12 Sequence Diagram:

The Sequence Diagram models are the collaboration of objects based on a time sequence. It shows how the objects interact with others in a particular scenario of a use case.

3.12.1 Start a game:

The user of the game can start a new game, after opening the application they can choose the option to start a new game then the first level will be loaded as shown in Fig 3.9:

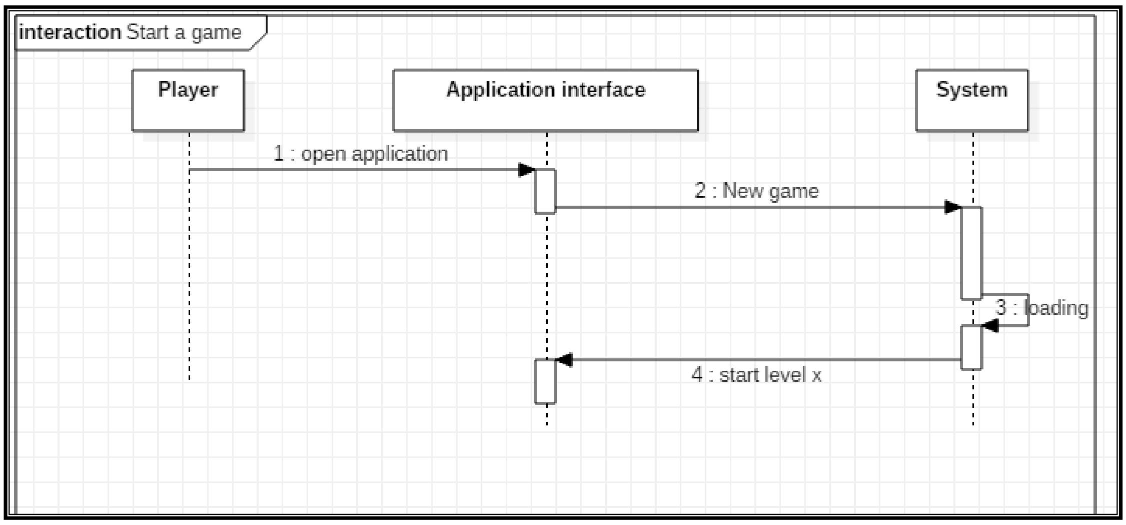


Fig 3.9: Sequence Diagram for Starting a New Game

3.12.2 Winning/ Passing a level:

The player can pass a level and win the game, after finishing a specific level the system will check the condition if they pass or not, and if the level was the last level the player will win the game as shown below:

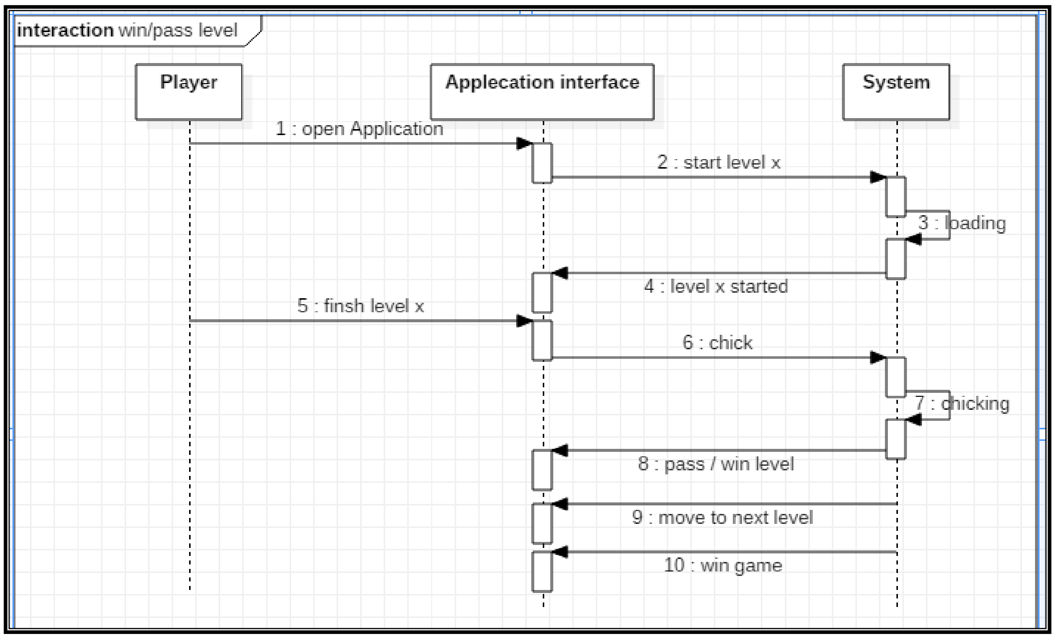


Fig 3.10:Sequance Diagram for Win / Pass Level

3.12.3 Failing a level:

Similar case to win/pass the level. The system will check after finishing a specific level and if the requirements for passing the level are not fulfilled the player will fail the level as shown in Fig 3.11:

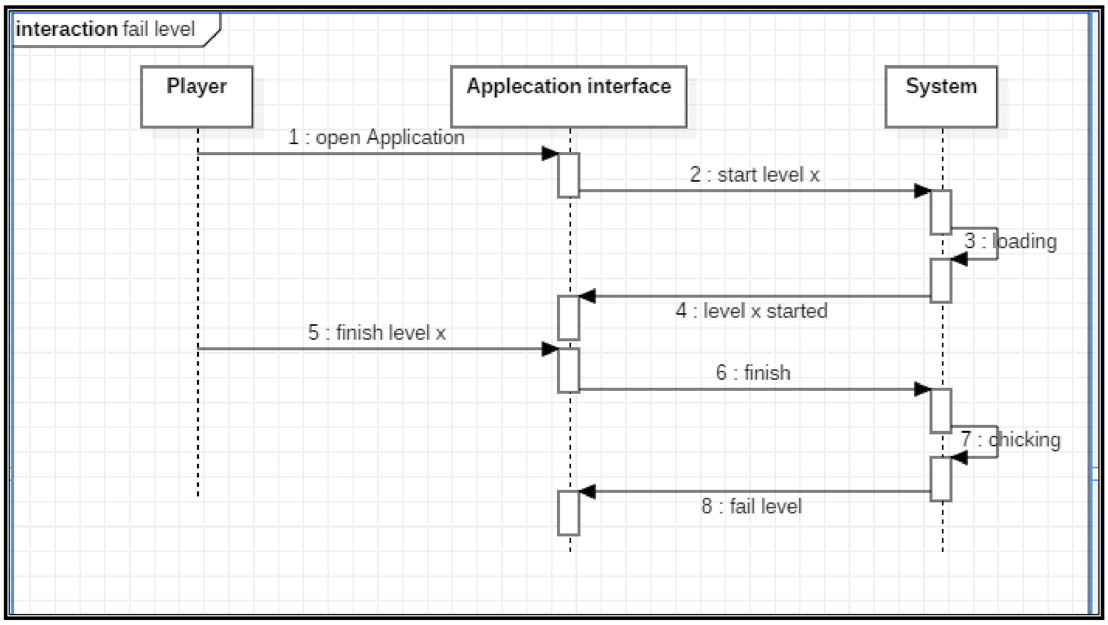


Fig 3.11:Sequance Diagram for Failing a Level

3.12.4 Credits:

If the users of the application wanted to know more about the creators or building of the game they can view the credits as shown in Fig 3.12:

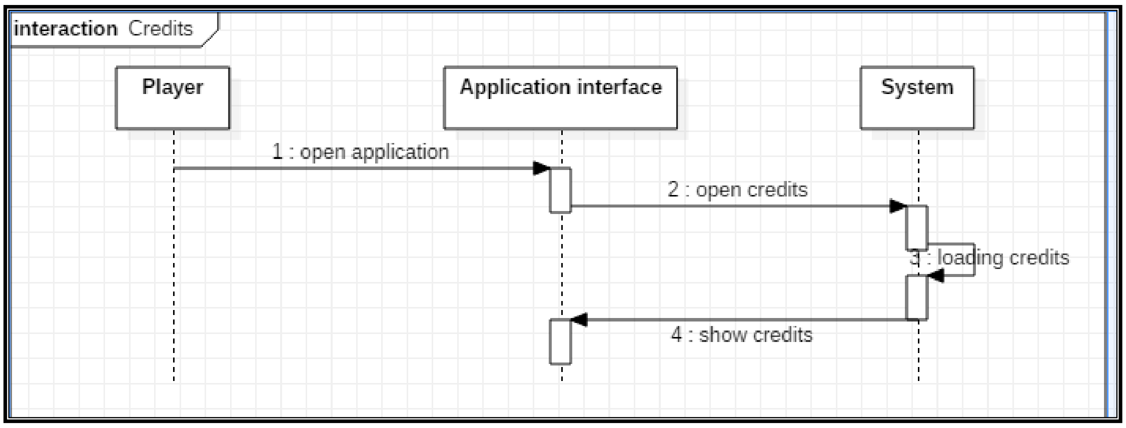


Fig 3.12:Sequance Diagram for Viewing Credits

3.12.5 Information:

In case the new players wanted to know more about the game and how to play it, win or fail levels they can view the related information as shown below:

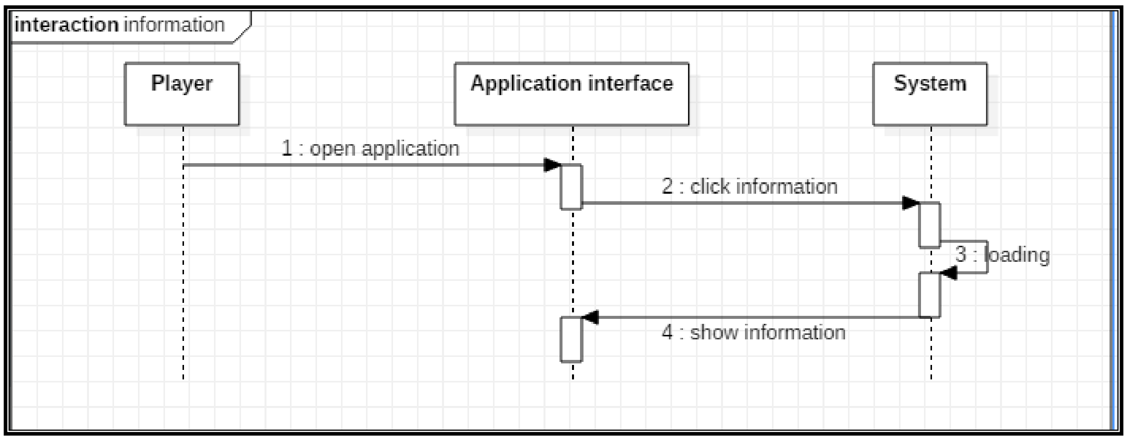


Fig 3.13:Sequance Diagram for Viewing Information

3.13Class Diagram:

Class Diagram is the tool, that specifies all objects, classes their attributes and operations and how they interact with each other, which is the core of our system.

Each class is represented by rectangle subdivided into three compartments: the top compartment contains the name of the class, the middle compartment contains the attributes of class, and the bottom compartment contains the operations the class can execute, while the multiplicity on each edge represents how many objects of each class take part in the relationships, we represent our class diagram in figure 3-14:

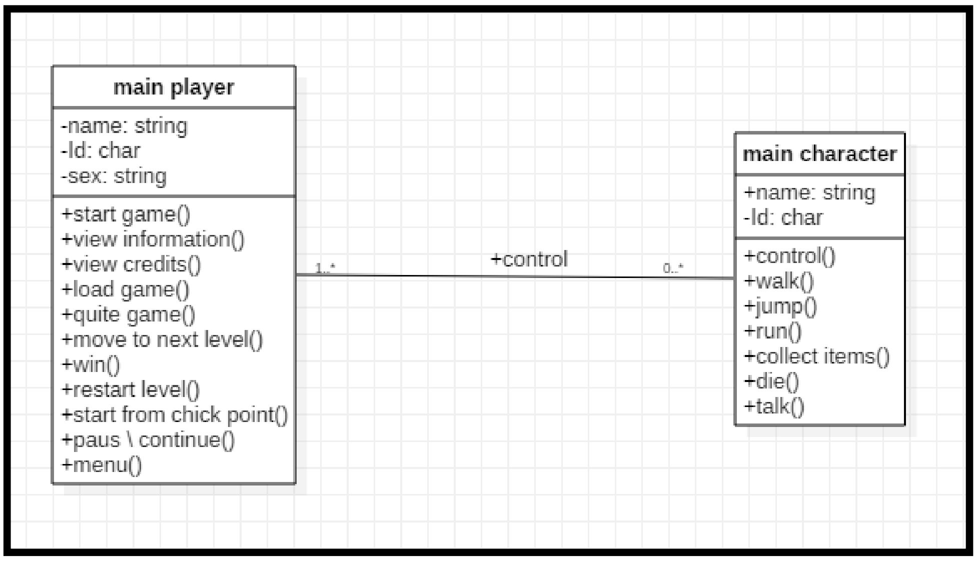


Figure 3.14 : Class Diagram

As shown in figure 3-14, the main class (main player) have the function control over the main character, this point and the related functionalities will be discussed more in depth in the next section .

3.14 Technical documentation:

3.14.1 Introduction:

In this section we will clarify the API for this project and discussing the functional specifications aspect along with the usage of white-box testing to demonstrate checking of a specific conditions in the system .

3.14.2 API:

API stands for Application programming interface. The “A” in API can be a piece of an application, the whole application or a whole server, or just about any piece of software. APIs provide an interface to stored data fitting the needs of an application allowing software to speak to each other. [13]

We used API to be able to clarify the following points:

- The internal interactions between the different objections.
- How the controls of the main character will be.
- How the process for checking for passing level will accrue.
- The main reason we used API, to distinguish between the functionalities of the main user and the functionalities of the main character.

For clarifying the objects, attributes and the relationships among the component in the system we used Class Diagram for this point as shown below:

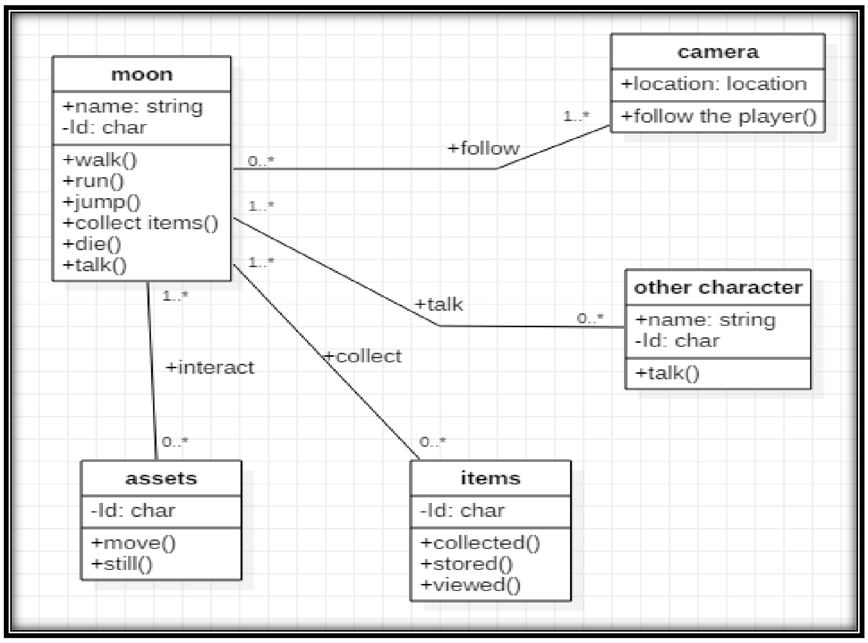


Fig 3.15: Class Diagram for Control Function

In the class diagram shown in fig 3.15, the main player function control is shown as it branched to include the following functions:

- The main character functions (moon).
- The camera functions.
- The assets functions.
- The items function.
- The other characters functions.

3.14.3 Classes:

❖ Moon class:

The main and only character that the player will be able to control as we mentioned previously.

❖ The camera class:

The main and only camera that will cover and show the level.

❖ The other characters class :

The other characters will present the story elements that is important in our game.

❖ The assets class:

The assets can be defined as everything we will use in building the environment of the levels.

❖ The items class:

The compounds that can be collected and sorted.

3.14.4 functional specifications:

In this section we will discuss the functions and interactions of each class in a deeper level.

<u>**Moon functionalities:**</u>

- Moon will be able to walk.
- Moon will be able to run.
- Moon will be able to jump.
- Moon will able to talk and interact with the other characters.
- Moon can collect different items.
- Moon could die if the player is not careful.

The camera functionalities:

The camera will follow the main character throughout the level.

The other characters functionalities:

The other characters that moon can talk to.

The assets functionalities:

The assets can be movable or still in its place.

The items functionalities:

The items that can be collect through the game.

3.14.5 White box-check:

White-box testing (also known as structural testing, clear box testing), is a method for checking the internal mechanism of a system or component. [14]

In this point we used white check-box method for testing two conditions in our game, win and fail as we present below:

The pseudo code:

We clarify here that obtaining the Stone from the set of the items is the condition to pass the level.

I short for item and S short for stone

```
IF I=S
THEN LEVEL =PASS
ELSE
LEVEL <>PASS
RETURN LEVEL
```

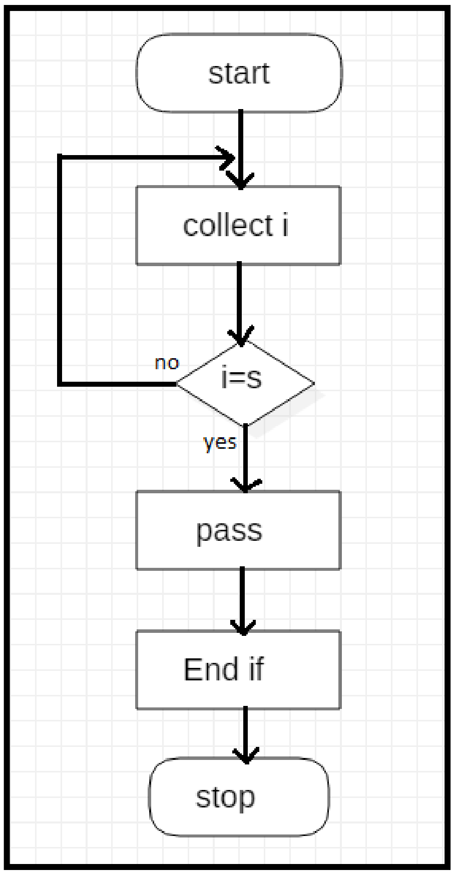


Fig 3.16: White Box-Check

3.14.6 Passing or falling the level condition:

As shown in fig 3.16 the condition for passing the level, if the specific item was found by the player and obtained then they will be able to pass on to the next level, if not they will not be able to and will have to start again (fail) until they find the key (stone) to pass the level.

3.15 Conclusion:

In this chapter, we discussed the project methodologies and described the feasibility study. Then, we specified the hardware and software requirement and made functional and nonfunctional requirement analysis. Other than that we discussed the test plan that we will be following and how to avoid the risks that may arise, Finally there was a use case, activity, sequence and class diagrams that represent our application.

Overall we discussed in this project the analytic aspect of designing and planning our game, in the first chapter we went over the creating of the idea, the problems in the existing systems and the scope of our project.

In the second chapter, we discussed the similar existing systems and analyzed the collected data from the target users.

In the third chapter, we presented the UML diagrams and the technical documentations that specify the internal interactions and functionalities and we mentioned the future work of our project.

CHAPTER FOUR

SYSTEM DESIGN

CHAPTER FOUR
SYSTEM DESIGN

4.1 Introduction:

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system, and it needs to satisfy the requirements of the users [15]. In this chapter the system's interfaces, the game's data, along with the technical and artistic stages for creating our game will be presented and discussed.

4.2 Users Interfaces (UI) Design and implementation:

The User interface (UI) design is the process of making interfaces in software or computerized devices with a focus on looks or style. Designers aim to create designs users will find easy to use and pleasurable. UI design typically refers to graphical user interfaces [16].

For our system, there are a total of seven user interfaces and windows beside the game scene, those interfaces are divided into two sections user interface out of the game scene, and user interface within the game scene(pop-up windows).

4.2.1 Users Interface Out of The Game Scene:

The user Interfaces that are out of the game scene are the ones that will be presented to the user before starting the actual game.

4.2.1.1 The Main Interface:

This interface is going to show up when clicking on the game icon and will contain four option buttons the Start New Game button, the Credit button, the Loading button, and lastly the Information button.

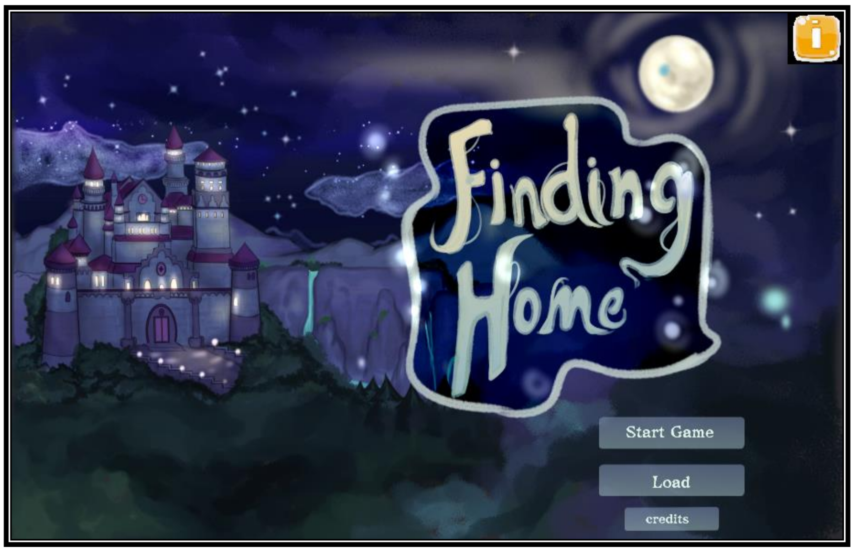


Fig 4.1:The Main Interface

4.2.1.2 The Credit Interface:

This interface is about the game's designers' information and how to contact them.

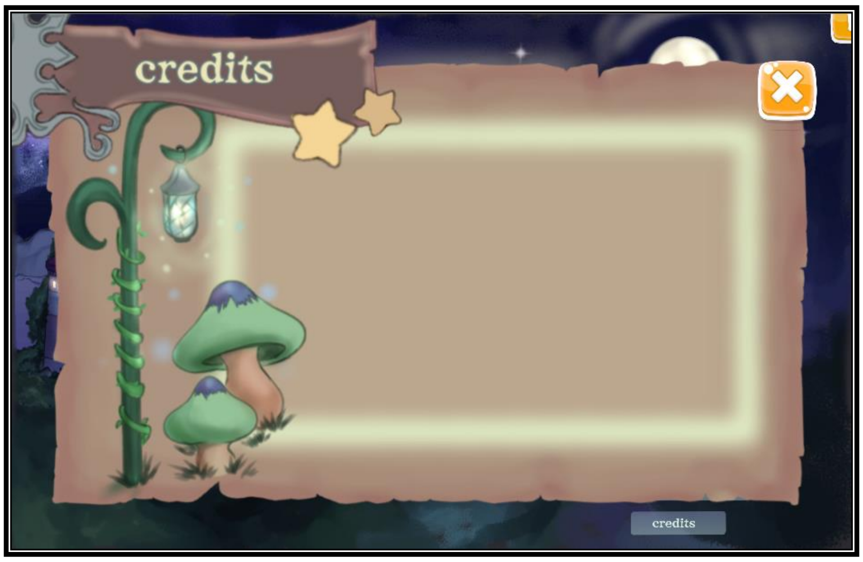


Fig 4.2: The Credit Interface

4.2.1.3 The Loading Interface:

The loading interface will contain options of the game levels and will drive the user to the selected level if its available for them .

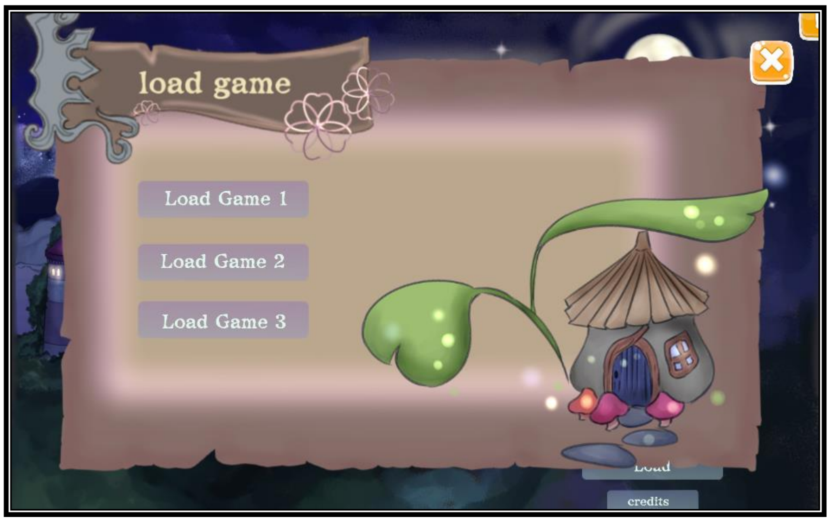


Fig 4.3: The Loading Interface

4.2.1.4 The Information Interface:

The information interface will contain the instructions on how to play the game and what is the meaning of certain items that are used in the game.

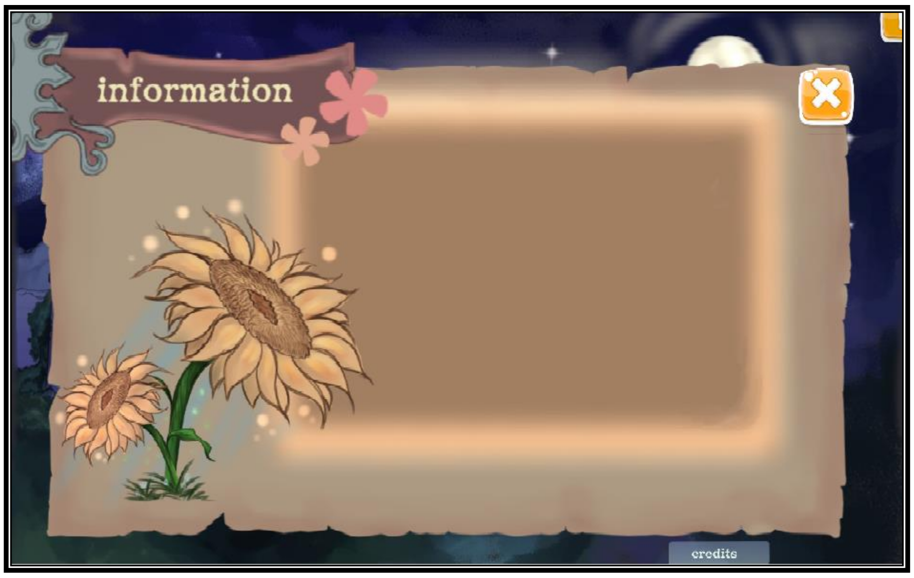


Fig 4.4 : The Information Interface

4.2.2 The User Interfaces (pop-up windows) within The Game Scene:

These windows are going to pop-up to the user when demanding an available operation while playing the game and will have the game scene in the background.

4.2.2.1 The Pause Option:

This window will pop-up when the player wants to stop the game for a little while before they continue playing again.

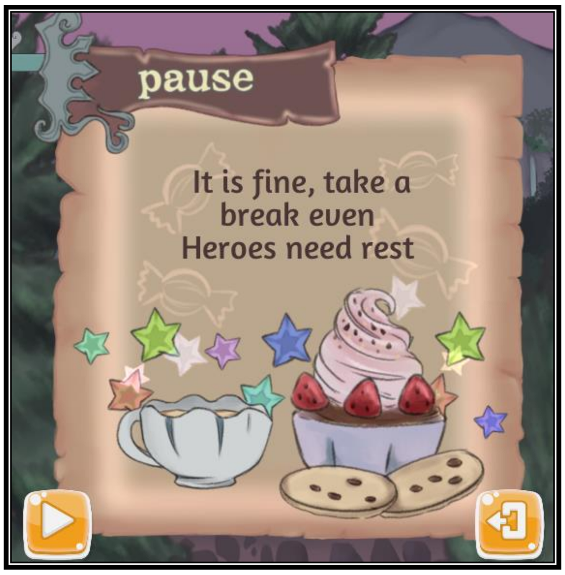


Fig 4.5: The Pause Option

4.2.2.2 The Menu:

This menu has an icon to click on which is going to show the items the user has collected so far.

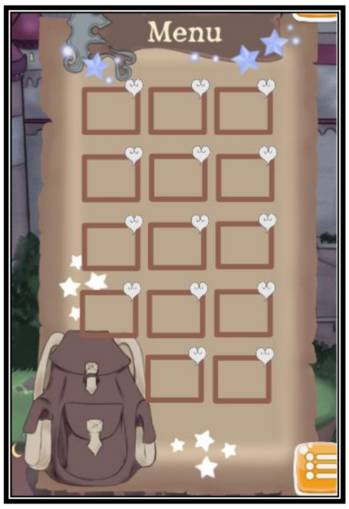


Fig 4.6: The Menu

4.2.2.3 The Information Option:

Information option has an icon to click on which is going to show the items and some information the user has collected in level.

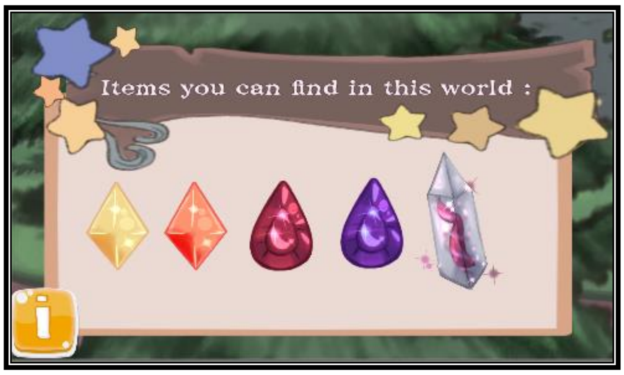


Fig 4.7: The Information Option

4.2.2.4 The Win Gate Window:

The win gate window is going to pop-up after winning the level, and it is going to congratulate the players for their achievement.

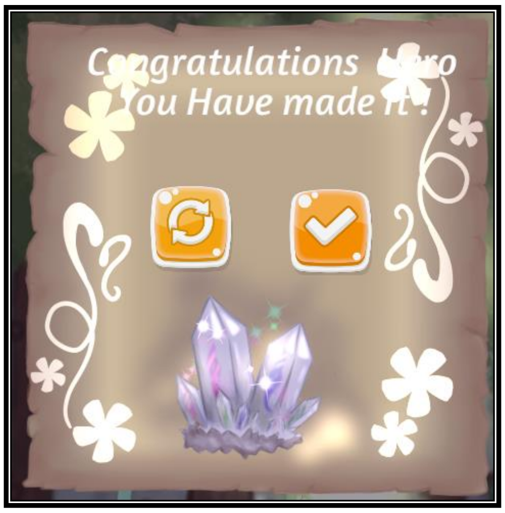


Fig 4.8: The Win Gate Window

4.2.2.5 The Fail Gate Window:

The fail gate window is going to pop-up when the player fails the game, and it is going to motivate the player to try again.

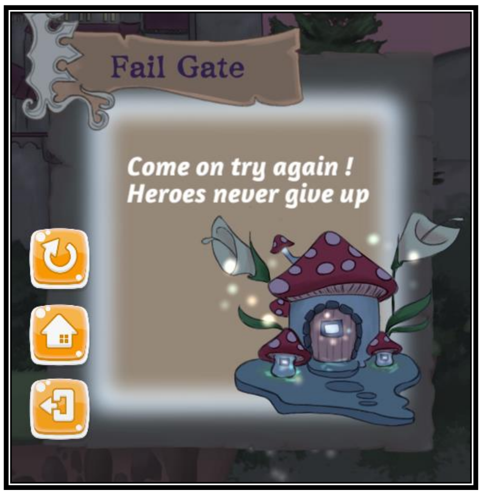


Fig 4.9: The Fail Gate Window

4.2.3 The Game Scene:

The option start game in the main user interface will take the player to the game scene where they can start adventuring the game's world and collecting the desired items in order to move to the next level and eventually win the game.

4.3 The Game's Data:

The game we will be presenting is an off-line game that could be installed in the users' devices. being an off-line simple game that does not contain complex data there will be no need to create a database.

Based on the information above our game can be considered as a Saved Game which is a piece of digitally stored information about the progress of a player in a video game.

The game will be created and marketed as a whole with all the data it contains. When a user download the game they will be able to play it level by level as it was structured. The game will have some constraint of not allowing the opening of a level before winning the previous one.

4.4 The Stages for Creating a Game:

To develop any game someone need to come up with an idea of what the game is going to talk about, and what is the purpose behind creating that game, after that someone will move to the next step which is deciding what designing style will be used to create the characters and assets for the game.

The designing style refers to the dimensions and the drawing style for characters and game objects. For our game we will be creating a Two Dimensions Game (2D Game) meaning the game events will be happening on a 2D plane and typically are either side-scrolling or vertically-scrolling [17].

Furthermore, we will be using a technique called Vector Art which is an increasingly popular form of digital illustration created based on geometric shapes [18], then we will start converting our idea to an actual game by going through four stages:

4.4.1 Drawing Stage :

Drawing style is the way to draw the characters, assets, and the whole game world along with decorating them together.

❖ **Characters' Drawing Style: to use**

The game characters are one of the most powerful elements for any game, it could drive children to download, or remove the game too, and from there devices knowing that we designed the characters to look loveable and colorful.

The drawing style we decided to use for the game characters is Anime Drawing Style rather than Cartoon Style since it is more realistic and it complements our game's story.

❖ **Moving The Character:**

To make a 2D character move we need to draw multiple picture frames for every simple movement in a circular way, this means that the picture the movement start with will be the same that movement will end with. Those picture frames will be controlled using a code to make the character appear as if it was doing the movement.

The main character available movements:

- **Moon's Idle State:**

Moon will be standing in her place but she will blink her eyes and appear like if she was breathing.

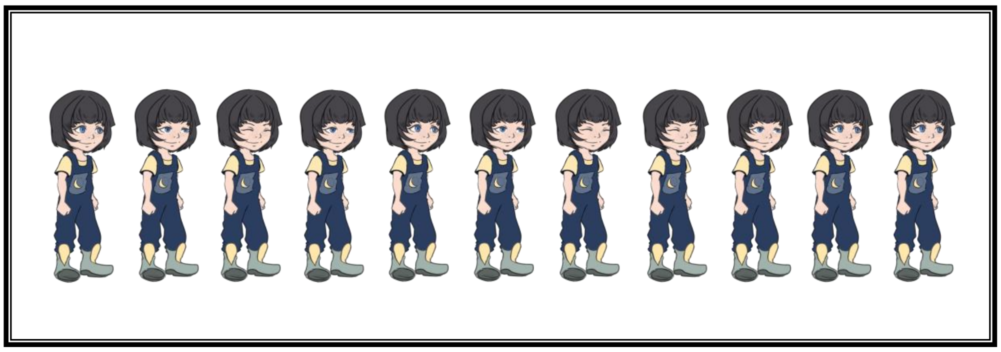


Fig 4.10: Moon's Idle State

- **Moon Walking:**

The multiple images combined together will make moon appear like if she was walking.

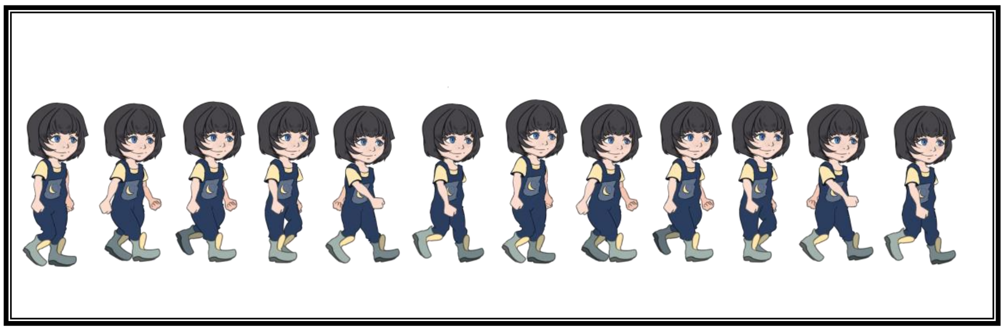


Fig 4.11: Moon Walking

- **Moon Running:**

Combining the picture frames together will make Moon appear like if she was running.

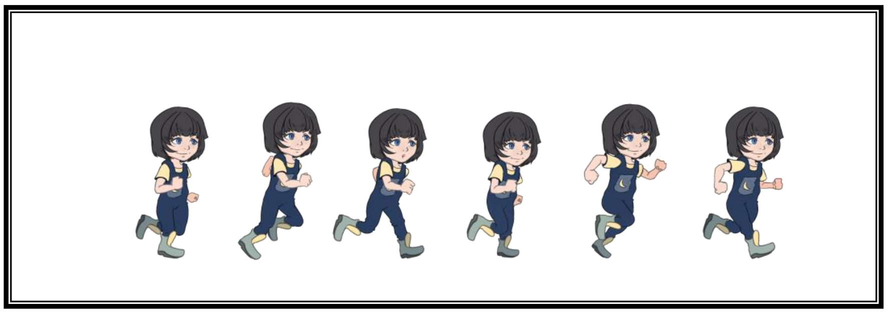


Fig 4.12: Moon Running

- **Moon Jumping:**

There are two jumping types in our game normal jump and high jump ,both jumping types will use the same picture frames, and the code well decide how the jump is going to be.

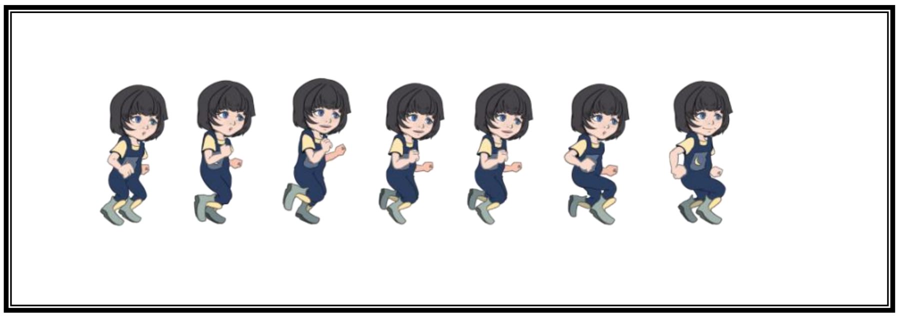


Fig 4.13: Moon Jumping

4.4.2 Assets Design:

The assets are the objects that are going to create the world when decorated into the game scene meaning we need to pay close attention to them in order to make the game scene look appealing to the user's eye.

Our goal for the assets is to make helpful items that connects the users to the game either by catching there attention or by making them participate in creating the game flow.

❖ **Types of Assets We Used:**

- **Intractable Elements**

  Intractable Elements are the elements that the main character controlled by the player can interact with.

  There are many indictable elements in the game. some can be triggered when the character walk by them, and some others can help the player go further in the game level.

- **Collectable Elements**

  Game elements that can be collected and used by the player.

- **Learning Elements**

Since our game is supposed to support the learning process we concluded many element to educate the children starting from presenting information, to telling a story, and demanding the player to make decisions.

4.4.3 World Building Stage:

After drawing all elements needed to create the game We can build the virtual world by simply dragging the elements and characters that we have created into the game scene in Unity Game Engine and giving them some proprieties**.**

❖ **GameObject:**

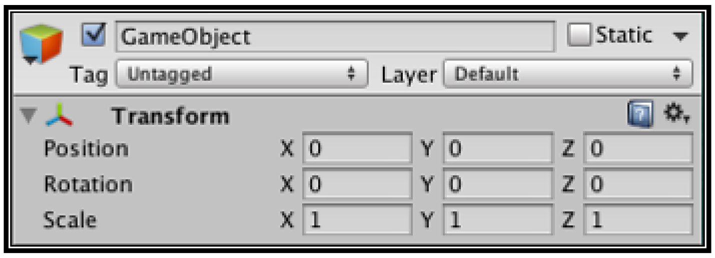
The objects that has been dragged to the game scene will be called GameObjects which is the most important concept in the Unity Editor.

To give a GameObject the properties it needs to become a light, or a tree, or a camera, you need to add components to it. Depending on what kind of object you want to create, you add different combinations of components to a GameObject.

Every object in your game is a GameObject, from characters and collectible items to lights, cameras and special effects. However, a GameObject can't do anything on its own; you need to give it properties before it can become a character, an environment, or a special effect [19].

❖ **Components:**

A GameObject contains components and the most common component is the Transform Component each game object must have a transform component [20].



☑ GameObject ☐ Static
Tag Untagged Layer Default

Transform

	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

Fig 4.14:Components of Game Object

Some component used in 2D games:

- **The Transform Component:**

It is impossible to create a GameObject in the Editor without a Transform Component. This component defines the GameObject's position, rotation, and scale in the game world and Scene view. The Transform Component also enables a concept called 'parenting' which is a critical part of working with GameObjects [21].

- **The camera component:**

Physical Camera properties simulate real-world camera formats on a Unity camera. This is useful for importing camera information from 3D modeling applications that also mimic real-world cameras [22].

- **Canvas Components:**

The Canvas is a GameObject with a Canvas component that contains all UI Components [23].

❖ **Physics:**

To have convincing physical behavior, an object in a game must accelerate correctly and be affected by collisions, gravity and other forces. Unity's built-in physics engines provide components that handle the physical simulation for you. With just a few parameter settings, you can create objects that behave passively in a realistic way (ie, they will be moved by collisions and falls but will not start moving by themselves).

By controlling the physics from scripts, you can give an object the dynamics of a vehicle, a machine, or even a piece of fabric. This section gives an overview of the main physics components in Unity, with links for further reading [24].

4.4.4 Story Stage:

The game we will be presenting is a story based game, beside that there is a story going on while the player plays the game. Two types of stories are used in the game world:

❖ **Obligatory Story:**

The obligatory story is a learning element, and it refers to the information about the game world, or the proprieties of a specific item that is presented to the user when passing a triggering object. The player can only read the information and skep them.

❖ **Intractable Story:**

Intractable stories are very useful way of delivering information and training the reader to solve problems and make decisions. The intractable Story is used in our game as a learning element.

4.4.5 Scripting Stage:

Scripting is an essential ingredient in all games. Even the simplest game needs scripts, to respond to input from the player and arrange for events in the gameplay to happen when they should. Beyond that, scripts can be used to create graphical effects, control the physical behavior of objects or even implement a custom AI system for characters in the game [25].

4.5 Conclusion:

In this chapter we presented the types of interfaces we used, and discussed them. Then, we talked about the game data and how is it going to be stored. Other than that we discussed the four stages we went through to create our game in details.

CHAPTER FIVE

TESTING

Chapter Five
Testing

5.1 Introduction:

The designed software needs to go through constant software testing to find out the problems and errors that might face the user. It is done so that the client gets a system that works properly, and is safe for their devices.

5.2 Testing plan:

Testing is a process of executing a program with the aim of finding error [26]. We will test the system to ensure that it is working correctly without any problem before handing it to the user. The testing process goes according to the following steps:

- Firstly after finishing the implementation of level one in our game we are going to test how it works on a PC then we will convert it to fit an Android devices after that we are going to test that version too.
- Secondly we will start implementing the second level, then we will test it in a similar testing process of the first level, same goes for the last level.
- Thirdly we are going to attach all the levels and the main menu together and test them, first for PC, then for Android.

5.2.1 Unit Testing:

It is a level of software testing where individual units/components of software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output [27].

5.2.2 Integration Testing:

Integration testing has two types: Top down and Bottom up.

- Black Box testing: - It is used for validation.

  In this type of testing we ignore internal working mechanism and
Focus on the output.

- White Box testing: - It is used for verification.

  In this type of testing we focus on internal mechanism. How to get the desired output? [28]

5.3 Project Scenarios Evaluation and Discussions:

In this section we will discuss the implementation results, and evaluate each one in details. The purpose of this section is to show how the system processes its functionality, and tasks.

5.3.1 Test Case (1): Main User Interface:

The main user interface is designed to lead the users to other windows they desire.

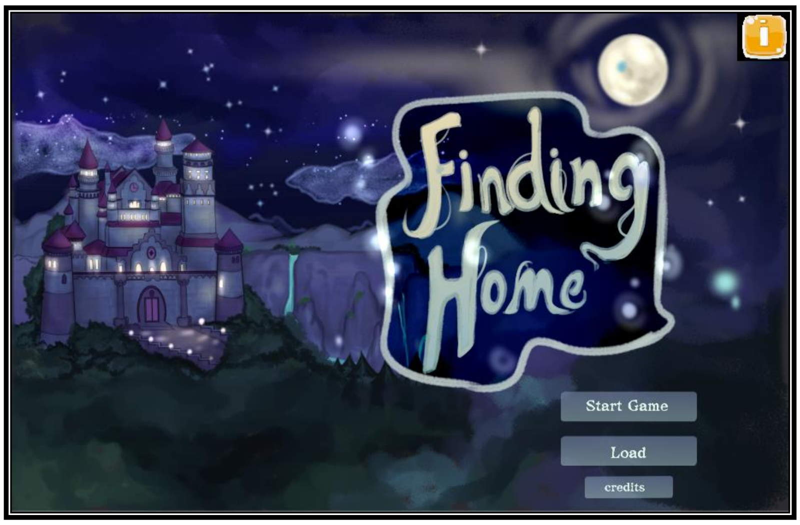


Fig 5.1: The Main User Interface

5.3.1.1 Main Interface: Start Game

Start game button should be able to drive us to the game's virtual world when clicked on.

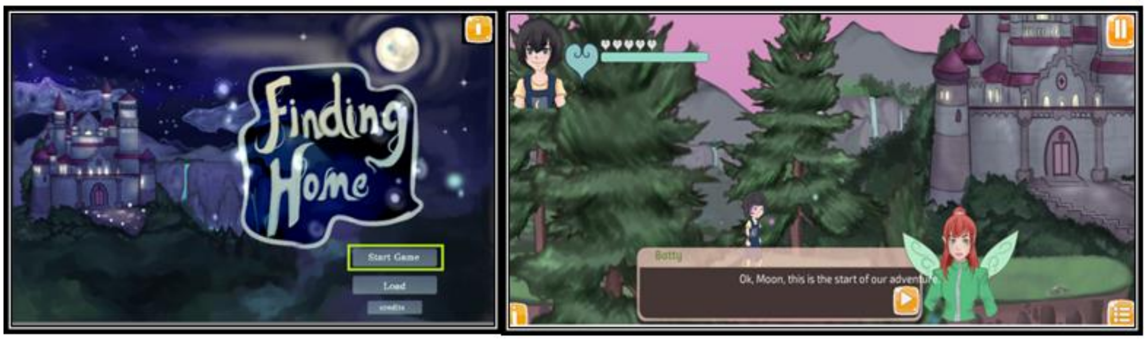


Fig 5.2: Start Game Button

5.3.1.2 Main Interface: Load

The load button should lead us to another window, which has three options of the game levels to choose from. The load window well drive us to the desired game level when selected.

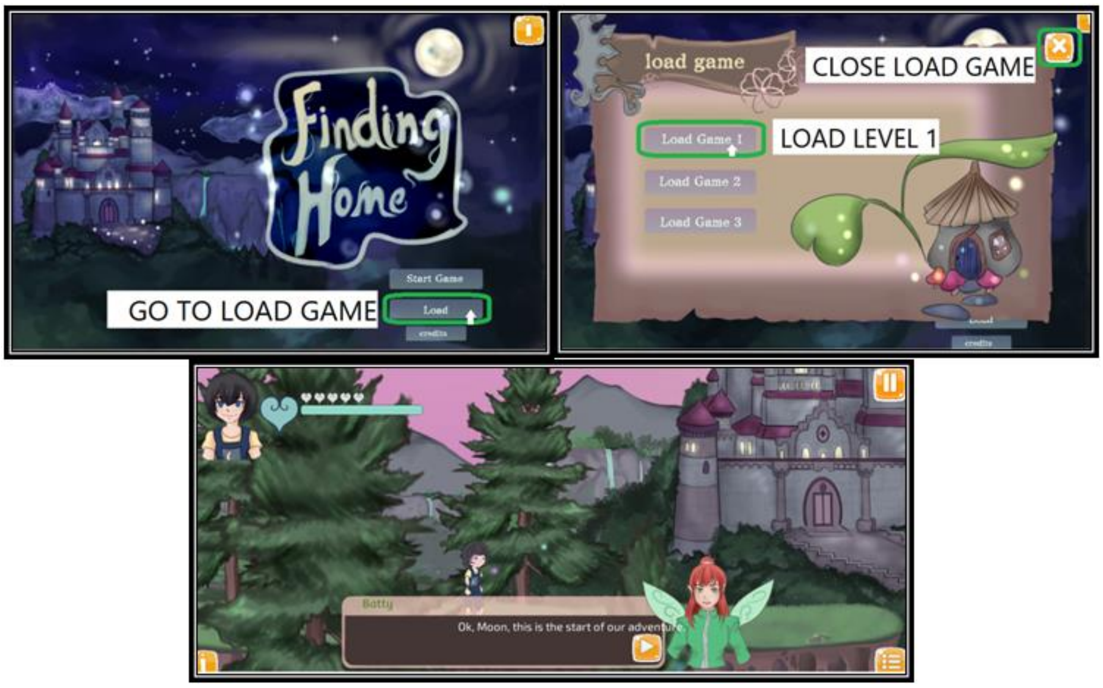


Fig 5.3: Load Game Button

The user has the ability to close the load window by clicking on the X icon on the top-right of the loading window. If the player wants to move to the second or third level the loading window will demand another window to show up which is the password window.

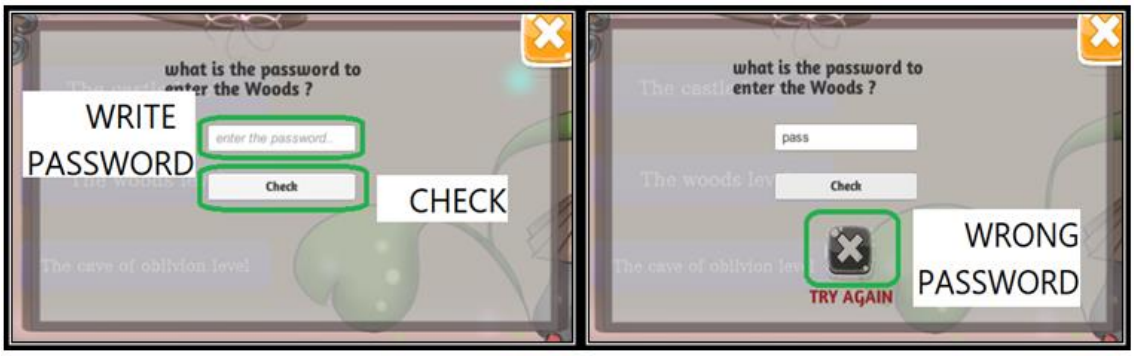

what is the password to enter the Woods ?
WRITE PASSWORD
enter the password..
Check
CHECK
what is the password to enter the Woods ?
pass
Check
TRY AGAIN
WRONG PASSWORD
The cave of oblivion level

Fig 5.4: Check from Password

After finishing a level a password will be given to the user. that password can be used to open the next level from the load window.

5.3.1.3 Main Interface: Credit

When clicking on the credit button the user will be shown some information about the creators of the game and how to contact them.

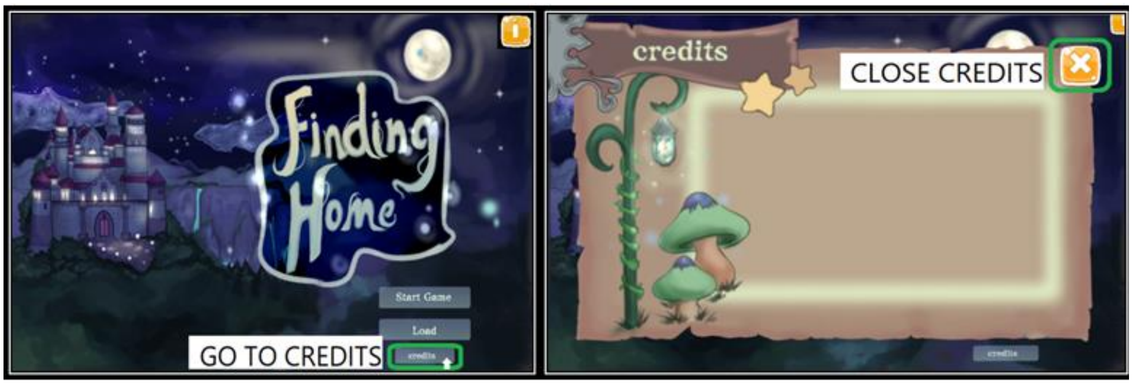


Fig 5.5: Credits Button

5.3.1.4 Main Interface: Information

There is an icon on the top-right of the main interface that icon is for information window. The information window will show some instructions about how to play the game.

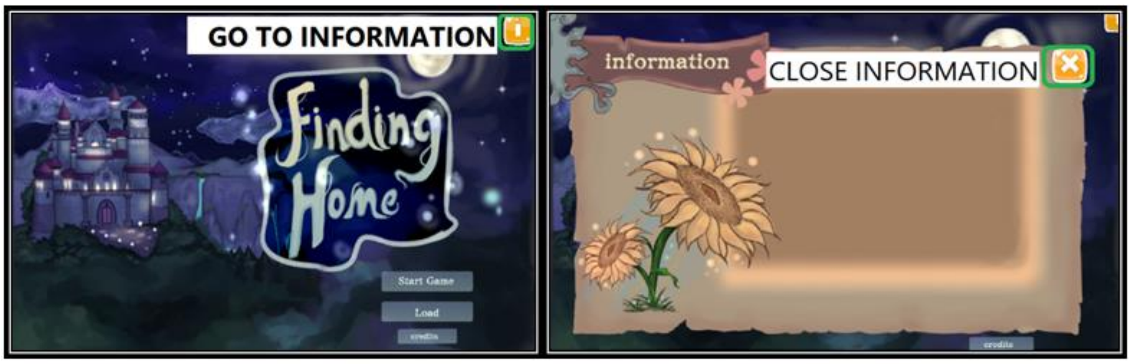


Fig 5.6: Information Button

5.3.2 Test Case (2): Menus and Pop-up Windows

There are a lot of icons, buttons, and pop-windows within the game scene that get activated when the player click on them.

5.3.2.2 The Pause Button:

It is a button on the top-right of the game scene. The pause button can be activated by a simple click. When clicking on the pause button a window will pop-up, and it is going to contain tow options exit the game or, continue.

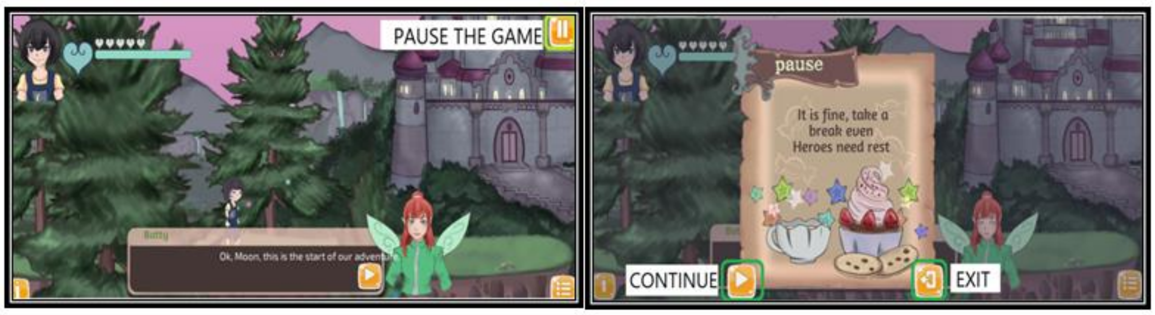

PAUSE THE GAME
Batty
Ok, Moon, this is the start of our adventure
pause
It is fine, take a break even Heroes need rest
CONTINUE
EXIT

Fig 5.7: The Pause Button

5.3.2.2 The Menu Button

When clicking the menu button a menu window will be presented on the bottom-right of the screen. The menu will contain the items the player has collected so far. The player can close the menu window by clicking on the X icon.

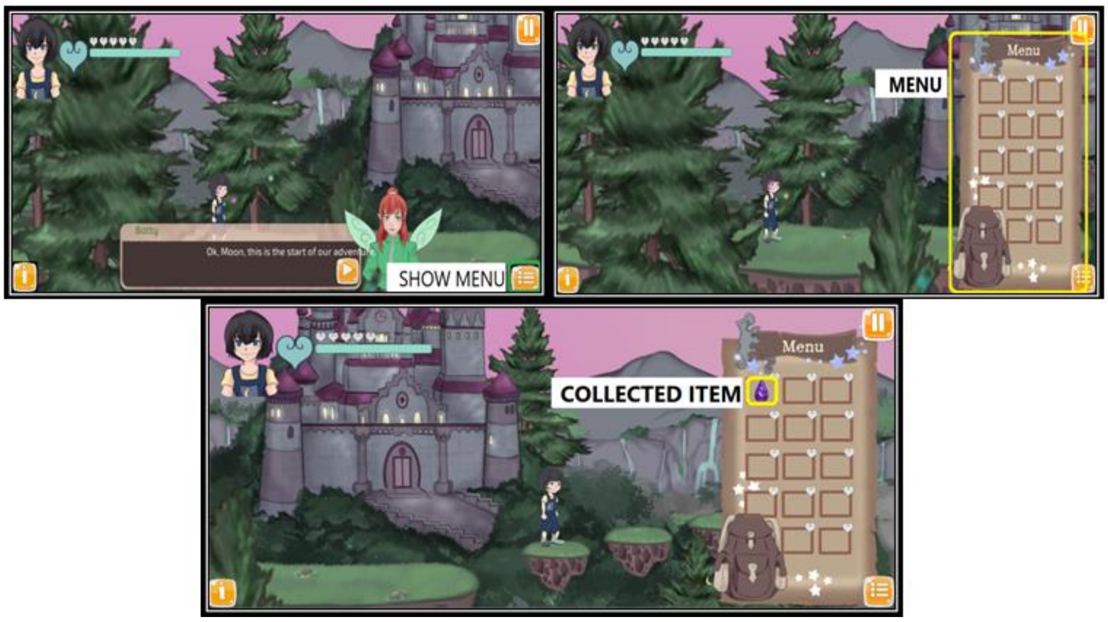
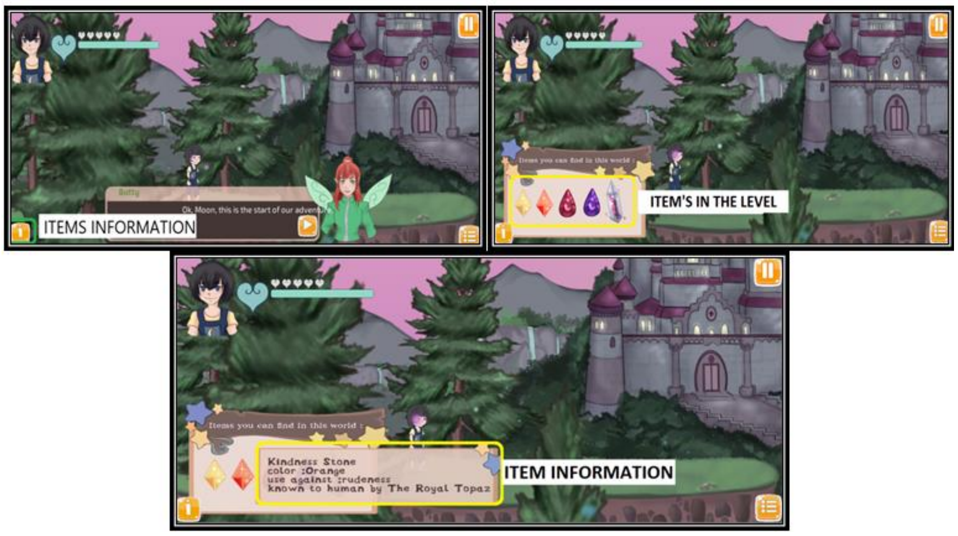


Fig 5.8: The Menu Button

5.3.2.4 The Information Button:

The information button is a button located in the bottom-left side of the game scene. When clicking on this button a small window will show up, the window will contain all items in that game level, and when clicking on one of the items some information about it will be presented.



Ok, Moon, this is the start of our adventure
ITEMS INFORMATION
Items you can find in this world :
ITEM'S IN THE LEVEL
Items you can find in this world :
Kindness Stone
color :Orange
use against :rudeness
known to human by The Royal Topaz
ITEM INFORMATION

Fig 5.9:The Information Button

5.3.2.4 Win Gate Window:

Win gate window shows up when the player finishes the level with the dusk stone collected, the player then can pass through the win gate. There are two options in the win gate window one is to move to the next level, and the other is to play the level again.

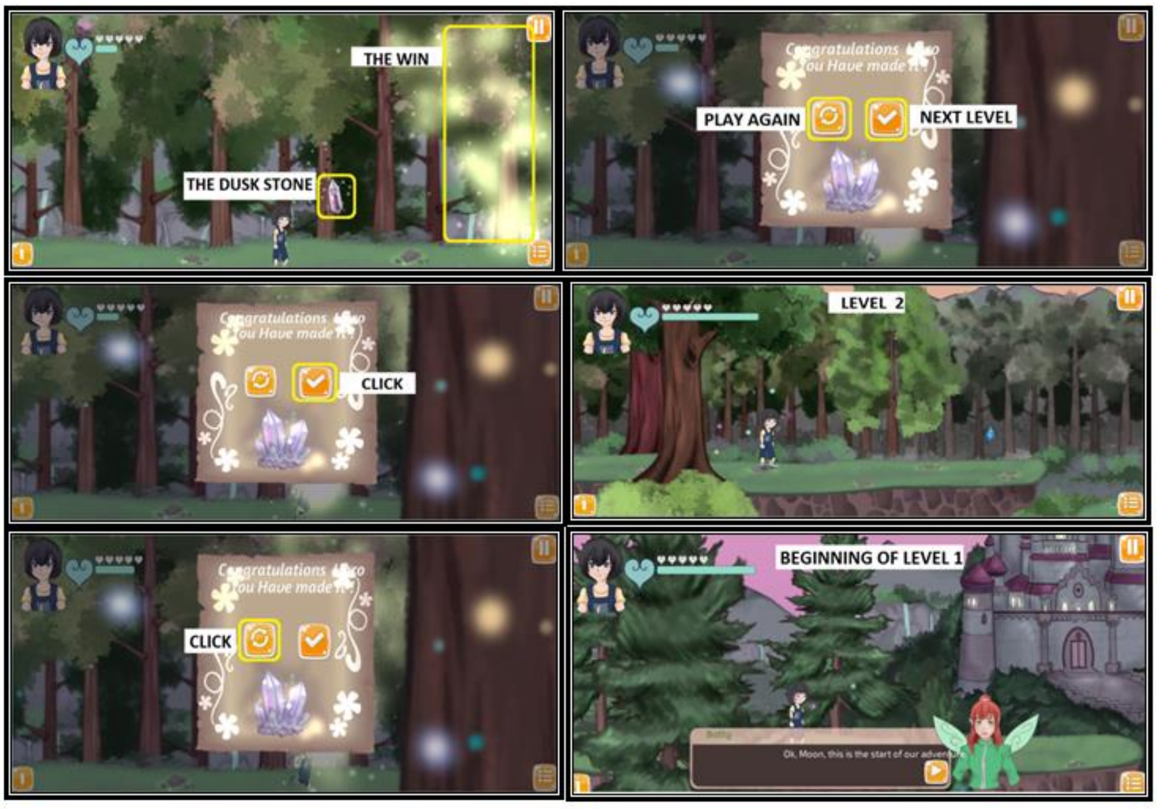


Fig 5.10: The Win Gate Window

5.3.2.5 Fail Gate Window:

It is the window that pops-up when the player loses the game. The player has a life slider, when the life slider comes to an end the player fails the game. in this window there are two options one is to play the level again, the other is to go to the main interface.

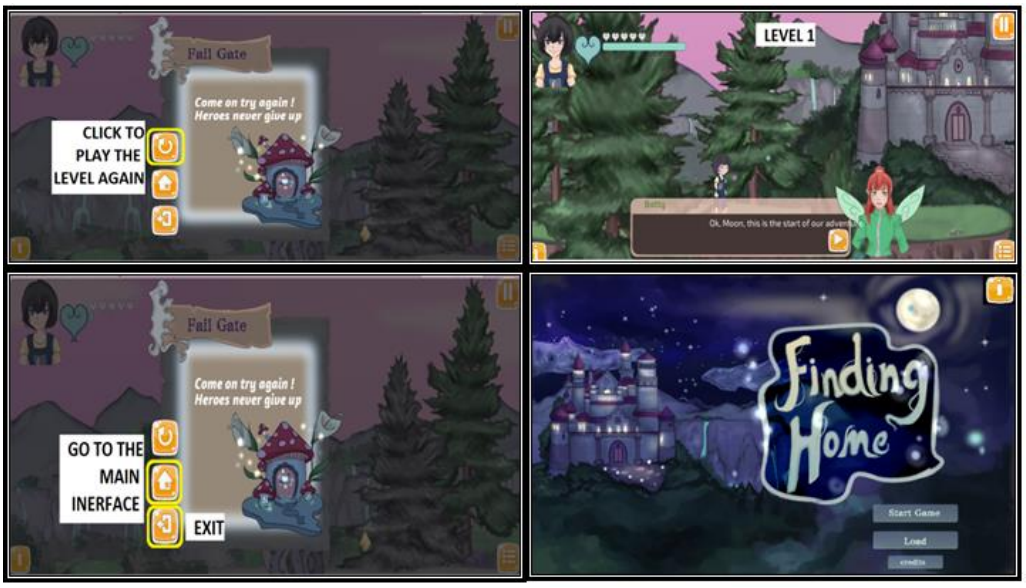

Fail Gate
Come on try again ! Heroes never give up
CLICK TO PLAY THE LEVEL AGAIN
LEVEL 1
Fail Gate
Come on try again ! Heroes never give up
GO TO THE MAIN INERFACE
EXIT
Finding Home
Start Game
Load
credits

Fig 5.11:The Fail Gate Window

5.3.3 Test Case (3): Character Functions

We need to test the functions of the main character and check if they are working properly.

- Walk Function

  Moon can walk from one place to another.

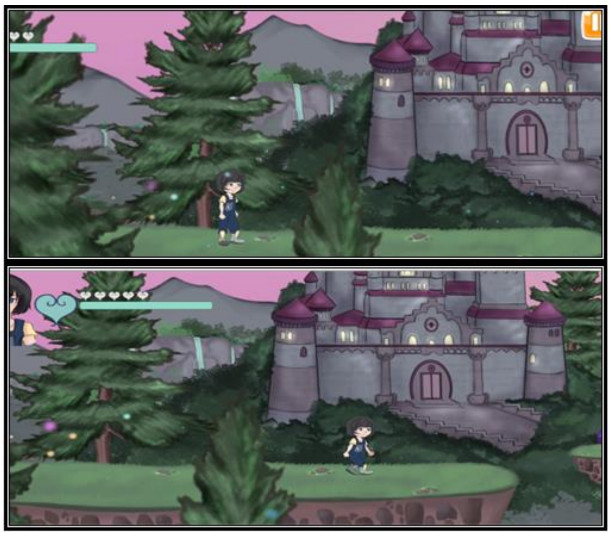


Fig 5.12 : The Walk Function

- Run Function

  Moon can increase her speed and start running.

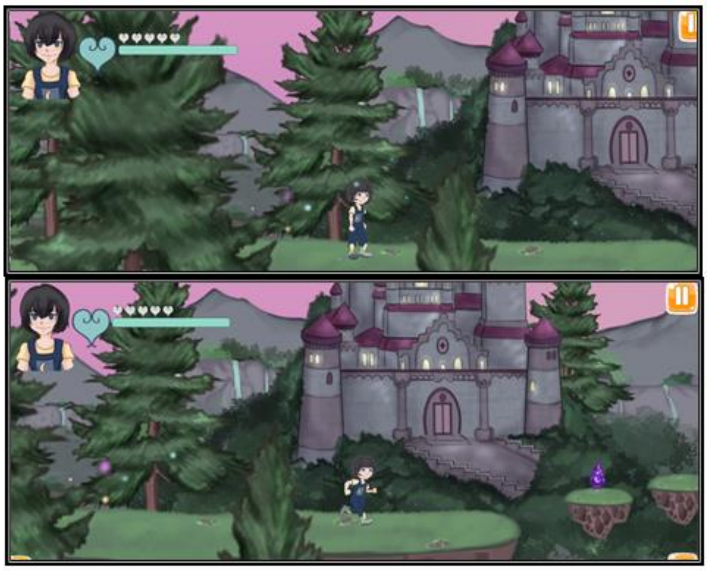


Fig 5.13 : The Run Function

- Jump Function

  Moon can jump and double jump . the script will decide which type of jumping is going to be used.

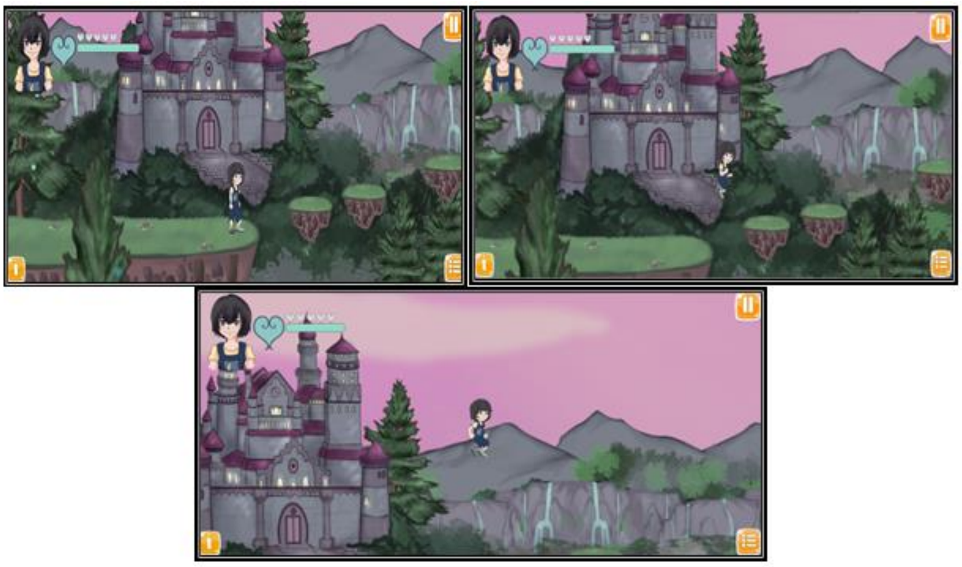


Fig 5. 14: The Jump Function

- Flip Function

  Moon is capable of flipping sides and walking in the wanted direction.

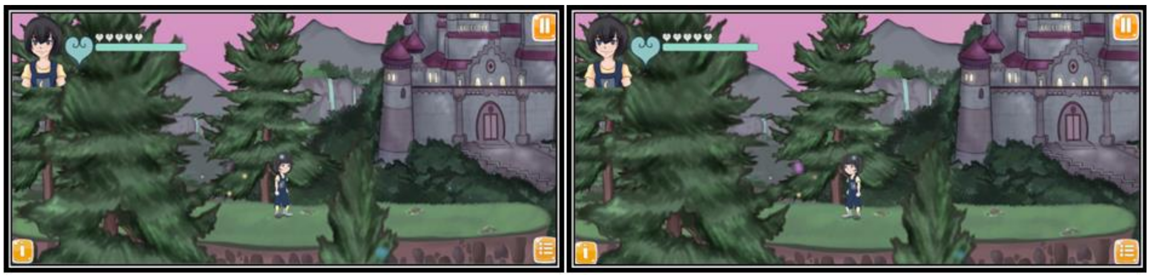


Fig 5.15: The Flip Function

- Collect Function

  Moon is able to collect some items in the game. The figure below shows that when Moon touches an item it will be collected and will appear in the menu.

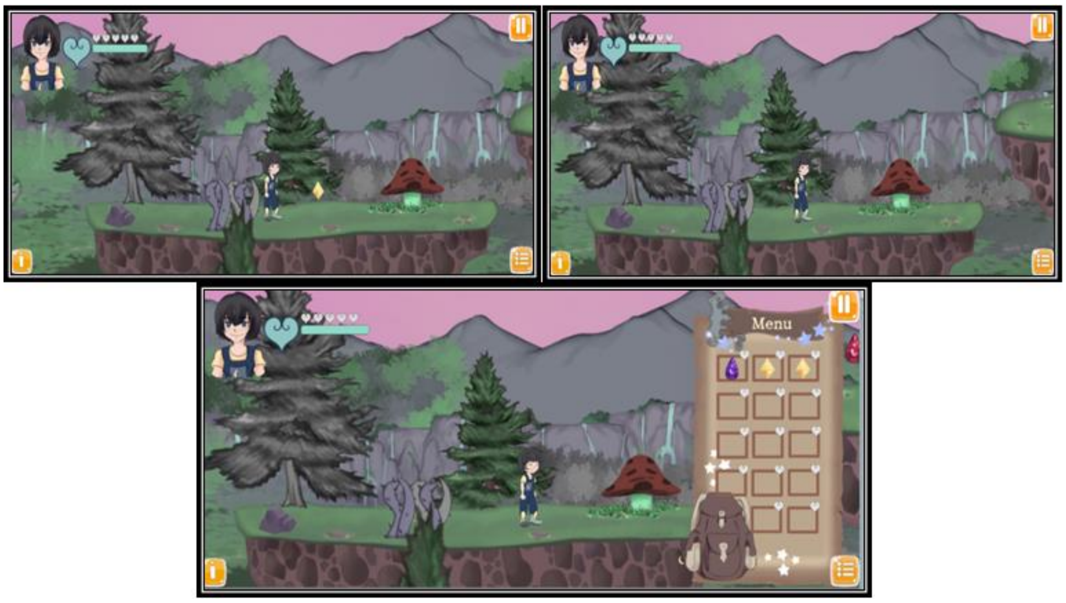


Fig 5.16:The Collect Function

- Die Function

Moon will die if her life slider gets emptied. and that happens either by touching the thorns, or falling from the ground.

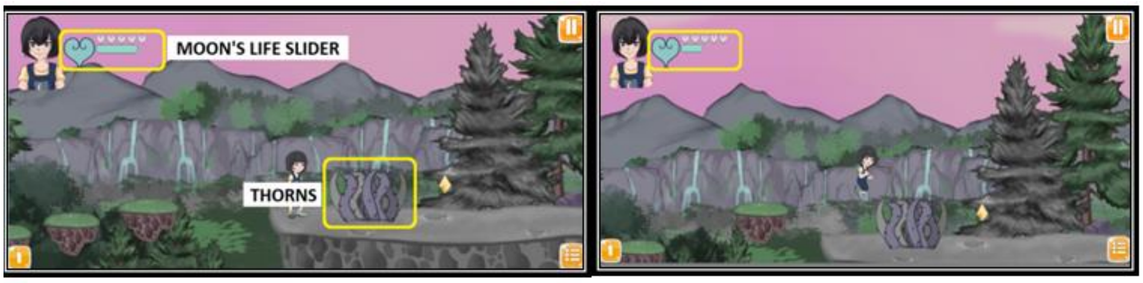


Fig 5.17: The Die Function (Thorns)

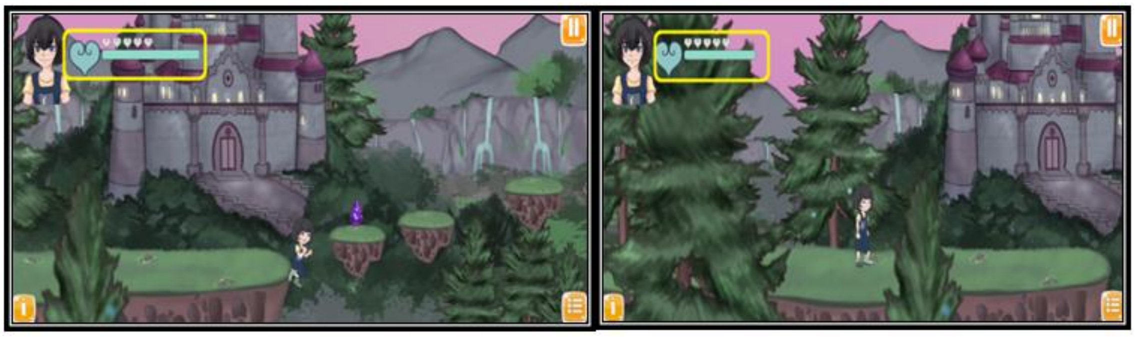


Fig 5.18: The Die Function (Fail)

5.3.4 Test Case (4): Elements' Functions

The main character in our game is able to interact with some world elements. One of the examples for that is collecting elements as we showed previously. In this section we will focus on the changes that happen to the elements when Moon walks by them or touches them.

- Chick Point Mushroom

When Moon passes by this mushroom, if she fell from the ground later she will be able to start again from the mushroom rather than the beginning of the level, that will happen if her life slider has not been emptied yet.

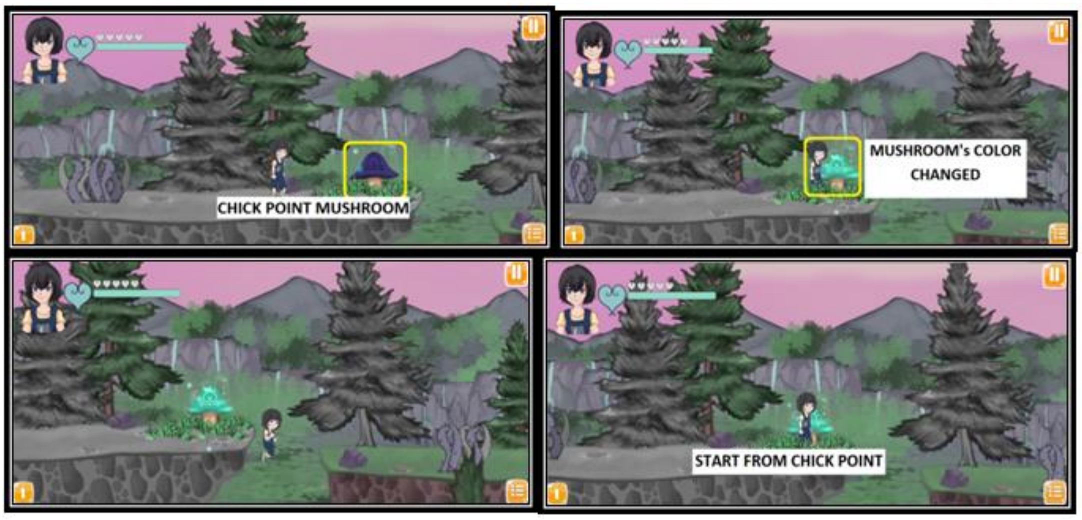


Fig 5.19: The Check Point Mushroom

- Jump Mushroom

When Moon touches this mushroom the mushroom will change its appearance, and if Moon jumps on it, the mushroom will give her a little push resulting in her having higher jump.

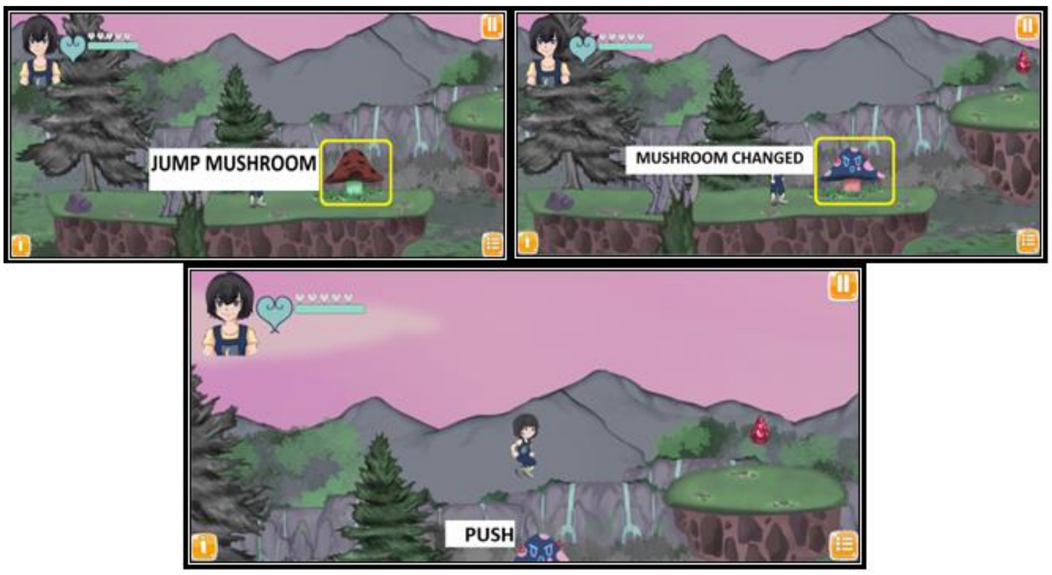


Fig 5.20:The Jump Mushroom

- Triggers

  Some elements in the game trigger conversation like stones, animals…..etc.

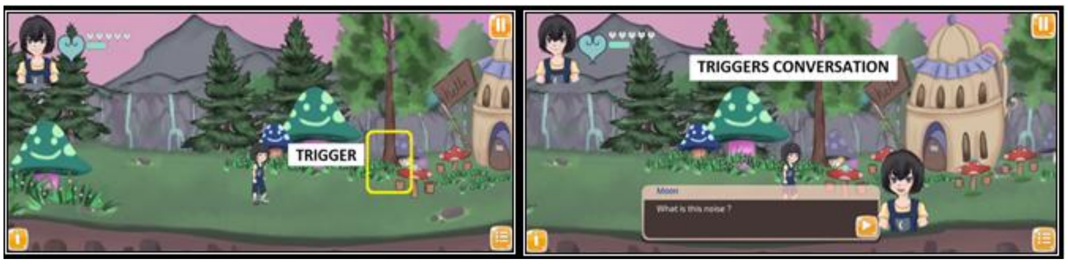


Fig 5.21: The Trigger Conversation

5.4 conclusion:

In this chapter we discussed the testing process as it is one of the important steps to create a software. We showed how we tested the functionality of our system and how it supposed to flow from one window to another.

CHAPTER SIX

CONCLUSION AND FUTURE WORK

CHAPTER SIX

CONCLUSION AND FUTURE WORK

6.1 Conclusion:

In this documentation we have discussed the idea of behavioral learning game(BLG), as it is a game that will be presented to the children to enable them of enjoying their time, and learn both at once, which is the main goal of our project. The game will motivate the children to read more as it is a story based game.

This documentation shows the methodologies and technics we have used to create the game. After designing the game, and testing it we started working on converting the game to fit Android devices, and now we have a game of three levels that work on windows as well as Android devices.

6.2 Future Work:

BLG can be expand to target more audience by increasing the difficulty of the challenges, adding more levels and puzzles using more advance software and hardware and using three dimensional graphics along with two dimensional graphics, while asking the preemption to install it inside the educational institutions' computers.

APPENDIX CODING

In the following pages we will be presenting the codes that were used to create the game.

• Player Control:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class playerControler : MonoBehaviour
{
    //movment variables
    public float MaxSpeed; //how much speed the character can move accessable from
unity
    Rigidbody2D MyRB; //refrvenced to the rigidbody to the charecter
    public Animator MyAnim; //refrenced to the animator
    bool FacingRight = true; //flip the character
    public float jumpSpeed = 5f;

    //fall detect
    public Vector3 Respone; // from wher the player will respone
    //public bool canMove;


    //jumping variables
    //what is ground,that the player can jump on
    public LayerMask WhatisGround;
    //location of the circle ,where to check the ground
    public Transform groundChack;
    //to make desicoun if player was on the ground a check machnesim
    public float groundRaduis = 0.2f;
    bool grounded = false; // the character is on the ground
    public float jumpForce = 700f;
    //double jump
    bool doubleJump = false;


    //palyer health script
    PlayerHealth health;
    //damege taken
    public float damege;
    //if a conversation started the player stop moving
    public textManager check;


    // Use this for initialization
    void Start()
    {

        MyRB = GetComponent<Rigidbody2D>();
        MyAnim = GetComponent<Animator>();


        Respone = transform.position;    // initial respone point When the game
statrts
```

```
        health = GetComponent<PlayerHealth>(); // to get a refrence to the player
health script
        check = FindObjectOfType<textManager>();



    }



    // Update is called once per frame
    void FixedUpdate()
    {



        // check to see if player is on the ground or not will return T/F
        grounded = Physics2D.OverlapCircle(groundChack.position, groundRaduis,
WhatisGround);
        //referenced to the animator
        MyAnim.SetBool("Is grounded", grounded);

        if (grounded)
            doubleJump = false;

        //refrenced to verical speed
        MyAnim.SetFloat("vspeed", MyRB.velocity.y);




        //jump with out animation



        if (Input.GetButtonDown("Jump") && grounded == true)
        {
            MyRB.velocity = new Vector2(MyRB.velocity.x, jumpSpeed);


        }
```

- **Camera Control:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class camerafollowPlayer : MonoBehaviour {

      public Transform target; // the object "transform" the camera will follwo in
this case the player
      public float smothSpeed; // the smoothiness of the camera trastions
      public Vector3 offSet;// offset between the camera and the player
      public bool boundry; // boundry that the camera cant go bound it
      public Vector3 minPos; // min boundry
      public Vector3 maxPos; // max boundry


    void FixedUpdate()
     {

        Vector3 WantedPosion = target.position + offSet; // the wanted posiont of
the cemer
        Vector3  smothMovment  =  Vector3.Lerp(transform.position,  WantedPosion,
smothSpeed*Time.deltaTime); //the smoth of the camera

        transform.position = smothMovment;  //addin two vectors  , the postion of
the transform equal the smoth of the movment


        if (boundry) // to determine the camera boundry
            transform.position   =   new   Vector3(Mathf.Clamp(transform.position.x,
minPos.x, maxPos.x),
               Mathf.Clamp(transform.position.y, minPos.y, maxPos.y),
               Mathf.Clamp(transform.position.z, minPos.z, maxPos.z));}}
```

- **Player Health:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.UI; //for using the ui elemnts
using UnityEngine;

public class PlayerHealth : MonoBehaviour
{

    public float MaxHealth; //character full health
    float currentHealth; //current health
    public GameObject DeathAffect; //the death affect

    public Slider HealthSlider;// the ui element that will show the health state
    public GameObject GameoverUI;


    private void Start()
    {
        currentHealth = MaxHealth; // at first the current health equal the max
health
        GameoverUI.SetActive(false);
```

```
         HealthSlider.maxValue = MaxHealth; //the full health of the slider will 
equal the max health of the player
         HealthSlider.value = MaxHealth;//current health equal the max health
    }



    public void AddDamege(float damege)
    {


        if (damege <= 0) return;


        currentHealth -= damege; // decrease the health by the amount of the damege
        HealthSlider.value = currentHealth;// change the value of the slider


        if (currentHealth <= 0) // if ran out of health die
        {

            MakeDead();
        }
    }

    public void MakeDead()
    {
        Destroy(gameObject); // destroy the object
        GameoverUI.SetActive(true);
        Instantiate(DeathAffect,   transform.position,   transform.rotation);   // 
instantiat the affect at this  position
    }
}
```

- **Damage:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EnemyDamege : MonoBehaviour {

    public float Damege; // the amount of damege can be determine from the main 
interface of the engine
    public float DamegeRange;// the range the damege can accure in
    public float bushForce;// the bush force that will bush the player away from the 
object
    float NextDamege; // when the next damege can accure


    private void Start()
    {
        NextDamege = 0f; // no space between the damege rounds

    }
    public void OnTriggerStay2D(Collider2D Other)
    {

        if  (Other.tag  ==  "Player"  &&  NextDamege  <  Time.time)  //  if  the  player
collides with the object
        {
            PlayerHealth                       thePlayerHeath                         =
Other.gameObject.GetComponent<PlayerHealth>();  //  get  a  refrence  to  the  player
health
            thePlayerHeath.AddDamege(Damege);   //  use  this  function  that  is  in  the
player health script
            NextDamege = Time.time + DamegeRange; // the next damege accures
            BushBack(Other.transform); //function


        }

    }
```

- **Start Plying:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.UI;
using UnityEngine;

public class StartPlaying : MonoBehaviour {

    public GameObject BLOCK;
    public GameObject button;
       // Update is called once per frame
       void Update () {

        Time.timeScale = 0f;
    }

    public void PlayGame()
        BLOCK.SetActive(false);//strat the game
        Time.timeScale = 1f;

    }
}
```

- **Pause Game:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.SceneManagement;
using UnityEngine;

public class PAUSE : MonoBehaviour {
    public GameObject pauseMenu; //the ui pause menu
    public GameObject pauseButton;//the poase button

    public void PauseGame()
    {
        pauseMenu.SetActive(true); //if presed the button then set the menu active
and diable the button
        pauseButton.SetActive(false);
        Time.timeScale = 0f;

    }

    public void ResumeGame()
    {
        pauseMenu.SetActive(false);//opposit to the one before se the menu off and
set the button on
        pauseButton.SetActive(true);
        Time.timeScale = 1f;

    }

    public void QuitGame()
    {
        Debug.Log("guitting game....");
        Application.Quit(); //will quit the application
    }
```

- **Loading Level:**

```
using System.Collections;
using UnityEngine.SceneManagement;
using UnityEngine;
using System.Collections.Generic;
public class NewBehaviourScript : MonoBehaviour {
 public void LoadLevel(int SceneIndex)
    {

        StartCoroutine(LoadAsync(SceneIndex));
    }
    IEnumerator LoadAsync(int SceneIndex)
    {
        AsyncOperation operation = SceneManager.LoadSceneAsync(SceneIndex);
        while(!operation.isDone)
        {
            Debug.Log(operation.progress); //the current prograss

            yield return null;
        }
    }
}
```

- **Level Loader:**

```
using System.Collections;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using UnityEngine;
public class LevelLoader : MonoBehaviour {

    public GameObject LoadingScreen;  //the loading ui object
    public Slider slider; //ui slider
    public void LoadLevel(int SceneIndex)
    {

        StartCoroutine(LoadAsync(SceneIndex)); //load the scene in asyncrenation
    }

    IEnumerator LoadAsync(int SceneIndex)
    {
        AsyncOperation  operation  =  SceneManager.LoadSceneAsync(SceneIndex);  //will
show information about the prograss
        LoadingScreen.SetActive(true); //will show the loading screen
        while (!operation.isDone) //while the operation is not done
        {

            float  prograss  =  Mathf.Clamp01(operation.progress  /  .9f);  //the  loding
between 0-0.9
            Debug.Log(prograss); //the current prograss
            slider.value = prograss; //set the slider value to the prograss value

            yield return null;

        }
    }
}
```

- **Winning Check:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.UI;
using UnityEngine;

public class WinnerCheck : MonoBehaviour {
    public GameObject WinnerUIGate;//the ui winner menu
    public GameObject deskStone;//the goal of the level

    private void OnTriggerEnter2D(Collider2D other)
    {
        //if the player has the stone and reach to the gate then open it
        if (other.tag == "Player" && (deskStone == null ))
        {
            Debug.Log("THE GATE IS OPEN  ");
            WinnerUIGate.SetActive(true);
            Time.timeScale = 0f;
        }
    }
}
```

- **Win Gate:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class WnnerGate : MonoBehaviour
{


    public GameObject RestartButton;
    public GameObject NextLevelButton;

    public void REstartLevel()
    {
        Debug.Log("restarting level one ");
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);   //reload   the
current scene
        Time.timeScale = 1f;
    }

    public void NextLevel()
    {
        Debug.Log("TO LEVEL TWO WE GO ..!! ");
        Time.timeScale = 1f;
        SceneManager.LoadScene("Scenes/LevelTwo");//to the second level
    }

    public void NextThreeLevel()
    {
        Debug.Log("TO LEVEL three WE GO ..!! ");
        Time.timeScale = 1f;
        SceneManager.LoadScene("Scenes/LevelThree");//to the third level
    }

    public void GoodBye()
    {
        Debug.Log("endGame!! ");
        Time.timeScale = 1f;
        SceneManager.LoadScene("Scenes/FinalGoodbye");//to the third level
    }
}
```

- **Password:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using UnityEngine;

public class PasswordFORL3 : MonoBehaviour {

    public InputField passwordField; //the inputfield
    public Button check;//button that will trigger the check password
    public GameObject tryAgain;//tryagain object that will appear on the screen
    public Button close; //the close button

    private void Start()
    {
        check.onClick.AddListener(passwords); //when the button clicked the password
will be checked
        close.onClick.AddListener(Clearmenu);//when the closed button is presed the
cleanmenu will be triggered
    }



    public void passwords()
    {
        // Get Password from Input
        string password = passwordField.text;

        string foundPassword = "yellow"; // the password to pass to  the level
        if ((foundPassword == password)) //the the interedpassword = the password
        {
            Debug.Log("open");
            SceneManager.LoadScene("Scenes/LevelThree"); //the second level will be
loaded
        }
        else
        {
            tryAgain.SetActive(true);  //if  not  the  try  again  object  will  be
triggered
        }
    }

    public void Clearmenu()
    {
        passwordField.text = " "; //clear the inputfield

        tryAgain.SetActive(false); //hide the try again onject
        return; //return everything to it origenal state

    }
}
```

- **Password Check:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using UnityEngine;

public class PasswordcCHECK : MonoBehaviour
{


    public InputField passwordField; //the inputfield
    public Button check;//button that will trigger the check password
    public GameObject tryAgain;//tryagain object that will appear on the screen
    public Button close; //the close button

    private void Start()
    {
        check.onClick.AddListener(passwords); //when the button clicked the password
will be checked
         close.onClick.AddListener(Clearmenu);//when the closed button is presed the
cleanmenu will be triggered
    }



    public void passwords()
    {
       // Get Password from Input
        string password = passwordField.text;

        string foundPassword="blue"; // the password to pass to  the level
        if  ((foundPassword == password)) //the the interedpassword = the password
        {
            Debug.Log("open");
            SceneManager.LoadScene("Scenes/LevelTwo");  //the  second  level  will  be
loaded
        }
        else
        {
            tryAgain.SetActive(true);  //if  not  the  try  again  object  will  be
triggered
        }
    }

    public void Clearmenu()
    {
          passwordField.text = " " ; //clear the inputfield

        tryAgain.SetActive(false); //hide the try again onject
        return; //return everything to it origenal state

    }
}
```

- **Game Over:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement; //using the scane mangger to control the scane 
loded
public class GAMEOVER : MonoBehaviour {


    public GameObject gameOverUI; //the game over Iu menu
    public GameObject RestartGameButton, QuitGame, menu; // the buttons


    public void Quitgame()
    {
        Debug.Log("guitting game....");
        Application.Quit();  // will qite the application
    }

    public void RestartGame()
    {

        Debug.Log("game over...");
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);   //reload   the 
current scene
    }


    public void menuButton()
    {

        Debug.Log("loding menu ");
        SceneManager.LoadScene("Scenes/menu");//will load the main menu scane
    }
}
```

- **Jump Mushroom:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class JumpMuchrom : MonoBehaviour
{

    public Sprite Sleep;//the defult sprit
    public Sprite Awake;// the sprite that will replace the defult sprit
    SpriteRenderer CheckJumpCondition;//sprite render to control the sprites
    public bool JumpCheck;// a boolean value just to check
    public float bushForce;//how much the jump force will be
    // Use this for initialization
    void Start()
    {
        //get the component from sprite
        CheckJumpCondition = GetComponent<SpriteRenderer>();
    }
    //chack
    private void OnTriggerEnter2D(Collider2D Other)
    {
        if (Other.tag == "Player") //if the player pass by then ..
        {
            CheckJumpCondition.sprite = Awake;//replace the sprite and
            JumpCheck = true;

            BushBack(Other.transform); //trigger the bush
        }
    }
    void BushBack(Transform BushedObject) // the direction where the player will be
bushed
    {
        Vector2   BushDirection   =   new   Vector2(1,   BushedObject.position.y   -
transform.position.y).normalized;//the bush direction
        BushDirection *= bushForce;//multiplay by the froce
        Rigidbody2D                             BushedRB                           =
BushedObject.gameObject.GetComponent<Rigidbody2D>();//the player riged body
        BushedRB.velocity = Vector2.zero;//velocity in the y direction
        BushedRB.AddForce(BushDirection,  ForceMode2D.Impulse);//what  kind  of  force
we will use

    }
}
```

- **Check Point:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class CheckPoint : MonoBehaviour {
    public Sprite darkCheckPoint;  // to determine the sprite "defoalt " one
    public  Sprite  lightCheckPoint;  //  the  sprite  thet  will  take  the  place  of  the
other defoult sprite
    SpriteRenderer CheckPointSpriteRender;//to be able to changr the sprite
    public bool CheckPointReach; // a bool value
    // Use this for initialization
    void Start () {
        CheckPointSpriteRender  =  GetComponent<SpriteRenderer>();    //  to  get  the
sprite compnent from the engine
       }
    void  OnTriggerEnter2D(Collider2D  Other)  //  on  trigger  function  that  will  check
when the tag object pass thi point
    {
        if (Other.tag == "Player")  // if the tag player enters the collidor
        {
            CheckPointSpriteRender.sprite  =  lightCheckPoint;  //  the  sprite  will
change
            CheckPointReach = true; // the check poit reached
        }
    }
}
```

- **Falling Platform:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class fallingPlatform : MonoBehaviour {
    Rigidbody2D platform; //rigid body that will be affected by the gravity
    public float fallDelay; //how much time before the platform falls
    private void Start()
    {
        platform = GetComponent<Rigidbody2D>(); //define the reigidbody components
    }
    //colliding condition checking
    private void OnCollisionEnter2D(Collision2D Other)
    {
        if (Other.collider.CompareTag("Player"))//if what called player collide with
this object then ..
        {
            StartCoroutine(fall()); //start the routine
        }
    }
    IEnumerator fall()//will control the function
    {
        yield return new WaitForSeconds(fallDelay); //wait for ..
        platform.isKinematic = false;//the gravity will affect the object
        GetComponent<Collider2D>().isTrigger = true; //set the trigger to passable
        yield return 0;
    }
}
```

- **Lamp:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class lamps : MonoBehaviour {
    //public varibles
    public Sprite off;//the defult sprit
    public Sprite on;// the sprite that will replace the defult sprit
    SpriteRenderer PassCondition;//sprite render to control the sprites
    public GameObject Affects;// the kind of affect that will be triggered when the
player pass by it

    // Use this for initialization
    void Start () {
        PassCondition = GetComponent<SpriteRenderer>();
    }

    private void OnTriggerEnter2D(Collider2D Other) // to check if the player passes
this point
    {
        if (Other.tag == "Player")
        {
            PassCondition.sprite = on;
            Instantiate(Affects,    transform.position,    transform.rotation);    //
instantiat the affect at the position
        }
    }
}
```

- **Platform Movement:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class platformMovement : MonoBehaviour {

    Vector3 positionA;//the defului pos
    Vector3 positionB;//the new position

    Vector3 Nextposition;
    [SerializeField]
    float speed; // speed of the movment
    [SerializeField]
    Transform child; //the paltform
    [SerializeField]
    Transform transformB;//the platrom in the new postion



       // Use this for initialization
       void Start () {
        positionB = transformB.localPosition; //the position b will be the pos of
the b platform in its local place
        positionA = child.localPosition;//the pos of a will be th epos of the child
platform
        Nextposition = positionB;
       }

       // Update is called once per frame
       void Update () {
        move(); //move function for the paltform

       }

    private void move()
    {
        child.localPosition = Vector3.MoveTowards(child.localPosition, Nextposition,
speed * Time.deltaTime);//the cild platfrom move toward ..


        if (Vector3.Distance(child.localPosition, Nextposition) <= 0.1) //change the
distincation
        {
            changeDestination();
        }


    }

    void changeDestination()//logical condtion
    {
         Nextposition= Nextposition != positionA ? positionA : positionB;

    }
}
```

- **Timer:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.UI;
using UnityEngine;

public class Timer : MonoBehaviour
{
    [SerializeField]
    Text countDown; //set a timer from the ui element
    [SerializeField]
    GameObject gameoverMenu; //the game over menu
    float currentTime = 0f; //current time that will change second by second
    float startingTime = 360f;//the starting time in seconds

    private void Start()
    {
        currentTime = startingTime; // at the start the current time will be same as
the starting time
    }


    private void Update()
    {
        currentTime -= 1 * Time.deltaTime; //the time will decreas second by second
        countDown.text = currentTime.ToString("0"); // only the integer number will
appears

        if(currentTime <= 60)
        {
            countDown.color = Color.red; // change color if the time is less than 60
        }
        if(currentTime <=0)
        {
            gameoverMenu.SetActive(true); // when the time is out game is over
            currentTime = 0;
        }

    }
}
```

- **Effects:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Affects1 : MonoBehaviour {


    //public varibles
    public Sprite Still; //the defult sprit
    public Sprite Pass; // the sprite that will replace the defult sprit
    SpriteRenderer PassCondition; //sprite render to control the sprites
    public GameObject Affects; // the kind of affect that will be triggered when the
player pass by it


    // Use this for initialization
    void Start() {

        //get the component from sprite
        PassCondition = GetComponent<SpriteRenderer>();


    }


        private void OnTriggerEnter2D(Collider2D Other) // to check if the player
passes this point
    {

        if (Other.tag == "Player")
        {
            PassCondition.sprite = Pass;
            Instantiate(Affects,   transform.position,   transform.rotation);   //
instantiat the affect at the position
        }


    }
}
```

- **Active Object:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class activiatObject : MonoBehaviour {

    public GameObject dialougeBox;
    public bool activeObject;


       // Update is called once per frame
       void Update () {
        if (activeObject == true)
        {
            dialougeBox.SetActive(true);
        }
        else
            dialougeBox.SetActive(false);

       }
    private void OnTriggerEnter2D(Collider2D Other)
    {
        if (Other.tag == "Player")
        {
            activeObject = true;
        }
        else
            activeObject = false;
    }
}
```

- **Trigger Stone:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class triggerStore : MonoBehaviour {

  //when the player collect the stone it will disappear from the level

           private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.tag == "Player")
        {
            Destroy(gameObject);
        }
    }

}
```

- **Trigger conversation:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.UI;
using UnityEngine;

public class triggerConversation : MonoBehaviour {


    public Dialouge dialouge;



    //triggr the conversation
    private void OnTriggerEnter2D(Collider2D Other)
    {
        if (Other.tag == "Player")
        {

            FindObjectOfType<textManager>().startDialouge(dialouge);//will    satart
the dialoge
            Destroy(gameObject);//the dialouge will work for one time only

        }

    }



}
```

- **Dialog:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class dialoge : MonoBehaviour {


    public TextAsset TextFile;
    public string[] textLines;


       // Use this for initialization
       void Start () {
        if(TextFile != null )
        {
            textLines = (TextFile.text.Split('\n'));
        }

       }}

       Textbox manager:
using System.Collections;
using System.Collections.Generic;
using UnityEngine.UI;
using UnityEngine;


    [System.Serializable]//to let this class show on the editor
    public class Dialouge
    {
          //the elements that can be change each conversation
         public Sprite Actor;//the actor icon
          public string name;//name
         [TextArea(4,10)] //wide of the text
         public string[] sentences;

    }
```

- **Text manager:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.UI;
using UnityEngine;
using System;

public class textManager : MonoBehaviour {


    /*the group of elemnts that can be definrd from the inspecter */
    public Text nameText;
    public Text DialougeText;
    public Animator anim;
    public Image imagespot;
    //we dont want the palyer to move the cahracter when a story begin
    public bool canMove;
    public playerControler conturl; //refrence to the player control
    private Queue<string> sentences;//queue of strings


       void Start () {
        sentences = new Queue<string>();
        conturl  =  FindObjectOfType<playerControler>();//to  get  the  components  fro
the other script

    }

     public void  startDialouge(Dialouge dialouge)
    {

         canMove = false;
        if (!canMove)
        {
            conturl.enabled = false; //disable the control
            conturl.MyAnim.SetBool("stop", true);
            conturl.MyAnim.SetBool("Is grounded", true);
        }

        anim.SetBool("Open", true); //start the dialouge box animation



        imagespot.sprite  =  dialouge.Actor;  //to  be  able  to  use  multiple  texts  and
images
        nameText.text = dialouge.name;
        sentences.Clear(); //if there is left sentence then clear it befor the next

        foreach  (string  sentence  in  dialouge.sentences)  //put  the  sentences  in  a
queue
        {
            sentences.Enqueue(sentence);

        }

        DisplayNextSentences();
    }

    public void DisplayNextSentences()
    {
        if(sentences.Count == 0) // if there is no more sentences then disable the
dialouge box
        {
            EndDialouge();
            return;

        }

       string Sentence= sentences.Dequeue(); //get the sentence of the queue

        StopAllCoroutines(); //stop the type affect
        StartCoroutine(TypeSentence(Sentence));




    }
    IEnumerator TypeSentence (string sentence) //the type affect
    {
        DialougeText.text = "";

        foreach(char Letter in sentence.ToCharArray()) //type each charater at a 
time
        {
            DialougeText.text += Letter;
            yield return null;

        }

    }
     void EndDialouge()
    {

         canMove = true;
        if ( canMove) // able the movement
        {
            conturl.enabled = true;
            conturl.MyAnim.SetBool("stop", false);
        }
            anim.SetBool("Open", false); //close the dialouge box


    }

}
```

- **Stone information:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class StonesInfo : MonoBehaviour {

    public GameObject Info; //a game object will be shown on the move of the mouse
    // Use this for initialization
    public void Start()
    {
        Info.SetActive(false); //at first it will be off

    }


    public void OnMouseOver()
    {

        Info.SetActive(true); // if the mouse move over it it will appear
    }
    public void OnMouseExit()
    {
        Info.SetActive(false); // if the mouse move from it it will disappear

    }
}
```

- **Story element:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.UI;
using UnityEngine;

public class PanelScript : MonoBehaviour {

    public GameObject panel;


    int counter;

    public void openPanel()
    {

        if (panel != null)
        {

            bool Isactive = panel.activeSelf;

            panel.SetActive(!Isactive);
        }

    }
}
```

- **Active text:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.UI;
using UnityEngine;

public class ActiveTextTaline : MonoBehaviour {
    public TextAsset Text;

    public int Startline;
    public int Endline;



    public TextBoxmanager textBox;
    public bool destoryWhenActivated;


       // Use this for initialization
       void Start () {
        textBox = FindObjectOfType<TextBoxmanager>();


    }



    private void OnTriggerEnter2D(Collider2D Other)
    {


        if (Other.tag == "Player")
        {
            textBox.ReloadScript(Text);
            textBox.CurrentLine = Startline;
            textBox.EndAtline = Endline;
            textBox.OnEnableTextbox();

            if (destoryWhenActivated)
            {
                Destroy(gameObject);
            }
        }
    }
}
```

❖ The Android Codes:

- **Player control:**

```
using System.Collections;
using System.Collections.Generic;
using UnityStandardAssets.CrossPlatformInput;
using UnityEngine;

public class playerControler : MonoBehaviour
{

    //movment variables
    public float MaxSpeed; //how much speed the character can move accessable from
unity
    Rigidbody2D MyRB; //refrvenced to the rigidbody to the charecter
    public Animator MyAnim; //refrenced to the animator
    bool FacingRight = true; //flip the character
    public float jumpSpeed = 5f;

    //fall detect
    public Vector3 Respone; // from wher the player will respone
    //public bool canMove;


    //jumping variables
    //what is ground,that the player can jump on
    public LayerMask WhatisGround;
    //location of the circle ,where to check the ground
    public Transform groundChack;
    //to make desicoun if player was on the ground a check machnesim
    public float groundRaduis = 0.2f;
    bool grounded = false; // the character is on the ground
    public float jumpForce = 800f;
    //double jump
    bool doubleJump = false;


    //palyer health script
    PlayerHealth health;
    //damege taken
    public float damege;
    //if a conversation started the player stop moving
    public textManager check;
    // Use this for initialization
    void Start()
    {

        MyRB = GetComponent<Rigidbody2D>();
        MyAnim = GetComponent<Animator>();


        Respone  =  transform.position;    //  initial  respone  point  When  the  game
statrts
        health  =  GetComponent<PlayerHealth>();  //  to  get  a  refrence  to  the  player
health script
        check = FindObjectOfType<textManager>();



    }



    // Update is called once per frame
    void FixedUpdate()
    {



        // check to see if player is on the ground or not will return T/F
        grounded   =   Physics2D.OverlapCircle(groundChack.position,   groundRaduis,
WhatisGround);
        //referenced to the animator
        MyAnim.SetBool("Is grounded", grounded);

        if (grounded)
            doubleJump = false;

        //refrenced to verical speed
        MyAnim.SetFloat("vspeed", MyRB.velocity.y);




        //jump with out animation



        if (CrossPlatformInputManager.GetButtonDown("Jump") && grounded == true)
        {
            MyRB.velocity = new Vector2(MyRB.velocity.x, jumpSpeed);


        }


        //if (!grounded) // cant turn while jump
            //return;

        //get the axis input
        float move = CrossPlatformInputManager.GetAxis("Horizontal");//how much we
moving




        //set the animator
        MyAnim.SetFloat("speed", Mathf.Abs(move)); //we dont care about the nigative
direction we care about the movement

        MyRB.velocity = new Vector2(move * MaxSpeed, MyRB.velocity.y);//to move the
character ,vector cus 2D (x,y)


        if (move > 0 && !FacingRight)
        {
            flip();
        }

        else if (move < 0 && FacingRight)
        {
            flip();
        }


    }


    void flip()
    {
        FacingRight = !FacingRight;
        Vector3 Scale = transform.localScale; // the local scale in unity x,y,z
        Scale.x *= -1; // to multipy the x scale with nigative to flip the image so
it face the other side
        transform.localScale = Scale; //tacke the upove and return it to the scale

    }

    void Update()
    {
        // jump / fall animation
        if ((grounded || !doubleJump) &&
CrossPlatformInputManager.GetButtonDown("Jump"))
        {
            MyAnim.SetBool("Is grounded", false);
            MyRB.AddForce(new Vector2(0, jumpForce)); //we the y axis force not x


            if (!doubleJump && !grounded) // double jumping
            {

                doubleJump = true;
                MyRB.AddForce(new Vector2(0, jumpForce));
            }

        }


    }
    void OnTriggerEnter2D(Collider2D other)
    {
        if (other.tag == "fallDetector") // if other objects enters the tag fall
selector
        {

            //damege
            health.AddDamege(damege);
            //what will happer if the player fall into the trigger by the tag name
            transform.position = Respone;

        }

        if (other.tag == "CheckPoint")
        {
            Respone = other.transform.position; // the respone point will be set to
the reached check point

        }

    }

}
```

❖ **The following scripts provided by the Unity assets**

- **Fungus assets:**

The flow chart

```
using UnityEngine;
using UnityEngine.EventSystems;
using System;
using System.Text;
using System.Linq;
using System.Collections.Generic;
using System.Text.RegularExpressions;

namespace Fungus
{


    /// <summary>
    /// Visual scripting controller for the Flowchart programming language.
    /// Flowchart objects may be edited visually using the Flowchart editor window.
    /// </summary>
    [ExecuteInEditMode]
    public class Flowchart : MonoBehaviour, ISubstitutionHandler
    {


        [HideInInspector]
        [SerializeField] protected int version = 0; // Default to 0 to always
trigger an update for older versions of Fungus.

        [HideInInspector]
        [SerializeField] protected Vector2 scrollPos;
        [HideInInspector]
        [SerializeField] protected Vector2 variablesScrollPos;

        [HideInInspector]
        [SerializeField] protected bool variablesExpanded = true;

        [HideInInspector]
        [SerializeField] protected float blockViewHeight = 400;

        [HideInInspector]
        [SerializeField] protected float zoom = 1f;

        [HideInInspector]
        [SerializeField] protected Rect scrollViewRect;

        [HideInInspector]
        [SerializeField] protected List<Block> selectedBlocks = new List<Block>();

        [HideInInspector]
        [SerializeField]   protected   List<Command>   selectedCommands   =   new
List<Command>();

        [HideInInspector]
        [SerializeField] protected List<Variable> variables = new List<Variable>();

        [TextArea(3, 5)]
        [Tooltip("Description text displayed in the Flowchart editor window")]
        [SerializeField] protected string description = "";

        [Range(0f, 5f)]
        [Tooltip("Adds a pause after each execution step to make it easier to
visualise program flow. Editor only, has no effect in platform builds.")]
        [SerializeField] protected float stepPause = 0f;

        [Tooltip("Use command color when displaying the command list in the Fungus
Editor window")]
        [SerializeField] protected bool colorCommands = true;

        [Tooltip("Hides the Flowchart block and command components in the inspector.
Deselect to inspect the block and command components that make up the Flowchart.")]
        [SerializeField] protected bool hideComponents = true;

        [Tooltip("Saves the selected block and commands when saving the scene. Helps
avoid version control conflicts if you've only changed the active selection.")]
        [SerializeField] protected bool saveSelection = true;

        [Tooltip("Unique identifier for this flowchart in localized string keys. If
no id is specified then the name of the Flowchart object will be used.")]
        [SerializeField] protected string localizationId = "";

        [Tooltip("Display line numbers in the command list in the Block
inspector.")]
        [SerializeField] protected bool showLineNumbers = false;

        [Tooltip("List of commands to hide in the Add Command menu. Use this to
restrict the set of commands available when editing a Flowchart.")]
        [SerializeField] protected List<string> hideCommands = new List<string>();

        [Tooltip("Lua Environment to be used by default for all Execute Lua commands
in this Flowchart")]
```

```
        [SerializeField] protected LuaEnvironment luaEnvironment;

        [Tooltip("The ExecuteLua command adds a global Lua variable with this name 
bound to the flowchart prior to executing.")]
        [SerializeField] protected string luaBindingName = "flowchart";

        protected static List<Flowchart> cachedFlowcharts = new List<Flowchart>();

        protected static bool eventSystemPresent;

        protected StringSubstituter stringSubstituer;

        #if UNITY_5_4_OR_NEWER
        #else
        protected virtual void OnLevelWasLoaded(int level)
        {
            LevelWasLoaded();
        }
        #endif

        protected virtual void LevelWasLoaded()
        {
            // Reset the flag for checking for an event system as there may not be 
one in the newly loaded scene.
            eventSystemPresent = false;
        }


        protected virtual void Start()
        {


            CheckEventSystem();
        }

        // There must be an Event System in the scene for Say and Menu input to 
work.
        // This method will automatically instantiate one if none exists.
        protected virtual void CheckEventSystem()
        {

            if (eventSystemPresent)
            {
                return;
            }

            EventSystem eventSystem = GameObject.FindObjectOfType<EventSystem>();
            if (eventSystem == null)
            {
                // Auto spawn an Event System from the prefab
                GameObject                          prefab                          = 
Resources.Load<GameObject>("Prefabs/EventSystem");
                if (prefab != null)
                {
                    GameObject go = Instantiate(prefab) as GameObject;
                    go.name = "EventSystem";
                }
            }

            eventSystemPresent = true;
        }

        private                                                              void
SceneManager_activeSceneChanged(UnityEngine.SceneManagement.Scene              arg0,
UnityEngine.SceneManagement.Scene arg1)
        {
            LevelWasLoaded();
        }

        protected virtual void OnEnable()
        {
            if (!cachedFlowcharts.Contains(this))
            {
                cachedFlowcharts.Add(this);
                //TODO  these  pairs  could  be  replaced  by  something  static  that
manages all active flowcharts
                #if UNITY_5_4_OR_NEWER
                UnityEngine.SceneManagement.SceneManager.activeSceneChanged       +=
SceneManager_activeSceneChanged;
                #endif
            }

            CheckItemIds();
            CleanupComponents();
            UpdateVersion();

            StringSubstituter.RegisterHandler(this);
        }

        protected virtual void OnDisable()
        {
            cachedFlowcharts.Remove(this);

            #if UNITY_5_4_OR_NEWER
            UnityEngine.SceneManagement.SceneManager.activeSceneChanged           -=
SceneManager_activeSceneChanged;
            #endif

            StringSubstituter.UnregisterHandler(this);
        }

        protected virtual void UpdateVersion()
        {
            if (version == FungusConstants.CurrentVersion)
            {
                // No need to update
                return;
            }

            // Tell all components that implement IUpdateable to update to the new
version
            var components = GetComponents<Component>();
            for (int i = 0; i < components.Length; i++)
            {
                var component = components[i];
                IUpdateable u = component as IUpdateable;
                if (u != null)
                {
                    u.UpdateToVersion(version, FungusConstants.CurrentVersion);
                }
            }

            version = FungusConstants.CurrentVersion;
        }

        protected virtual void CheckItemIds()
        {
            // Make sure item ids are unique and monotonically increasing.
            // This should always be the case, but some legacy Flowcharts may have
issues.
            List<int> usedIds = new List<int>();
            var blocks = GetComponents<Block>();
            for (int i = 0; i < blocks.Length; i++)
            {
                var block = blocks[i];
                if (block.ItemId == -1 || usedIds.Contains(block.ItemId))
                {
                    block.ItemId = NextItemId();
                }
                usedIds.Add(block.ItemId);
            }

            var commands = GetComponents<Command>();
            for (int i = 0; i < commands.Length; i++)
            {
                var command = commands[i];
                if (command.ItemId == -1 || usedIds.Contains(command.ItemId))
                {
                    command.ItemId = NextItemId();
                }
                usedIds.Add(command.ItemId);
            }
        }

        protected virtual void CleanupComponents()
        {
            // Delete any unreferenced components which shouldn't exist any more
            // Unreferenced components don't have any effect on the flowchart
behavior, but
            // they waste memory so should be cleared out periodically.

            // Remove any null entries in the variables list
            // It shouldn't happen but it seemed to occur for a user on the forum



            variables.RemoveAll(item => item == null);

            var allVariables = GetComponents<Variable>();
            for (int i = 0; i < allVariables.Length; i++)
            {
                var variable = allVariables[i];
                if (!variables.Contains(variable))
                {
                    DestroyImmediate(variable);
                }
            }

            var blocks = GetComponents<Block>();
            var commands = GetComponents<Command>();
            for (int i = 0; i < commands.Length; i++)
            {
                var command = commands[i];
                bool found = false;
                for (int j = 0; j < blocks.Length; j++)
                {
                    var block = blocks[j];
                    if (block.CommandList.Contains(command))
                    {
                        found = true;
                        break;
                    }
                }
                if (!found)
                {
                    DestroyImmediate(command);
                    Destroy(gameObject);
                }
            }

            var eventHandlers = GetComponents<EventHandler>();
            for (int i = 0; i < eventHandlers.Length; i++)
            {
                var eventHandler = eventHandlers[i];
                bool found = false;
                for (int j = 0; j < blocks.Length; j++)
                {
                    var block = blocks[j];
                    if (block._EventHandler == eventHandler)
                    {
                        found = true;
                        break;
                    }
                }
                if (!found)
                {
                    DestroyImmediate(eventHandler);

                }
            }


        }

        protected virtual Block CreateBlockComponent(GameObject parent)
        {
            Block block = parent.AddComponent<Block>();
            return block;


        }

}

}
```

❖ Unity standers assets

The cross plat form scripts

- **Touch inputs:**

```
using System;
using UnityEngine;
using UnityEngine.EventSystems;

namespace UnityStandardAssets.CrossPlatformInput
{
	public class AxisTouchButton : MonoBehaviour, IPointerDownHandler, 
IPointerUpHandler
	{
		// designed to work in a pair with another axis touch button
		// (typically with one having -1 and one having 1 axisValues)
		public string axisName = "Horizontal"; // The name of the axis
		public float axisValue = 1; // The axis that the value has
		public float responseSpeed = 3; // The speed at which the axis touch 
button responds
		public float returnToCentreSpeed = 3; // The speed at which the button 
will return to its centre

		AxisTouchButton m_PairedWith; // Which button this one is paired with
		CrossPlatformInputManager.VirtualAxis m_Axis; // A reference to the 
virtual axis as it is in the cross platform input

		void OnEnable()
		{
			if (!CrossPlatformInputManager.AxisExists(axisName))
			{
				// if the axis doesnt exist create a new one in cross 
platform input
				m_Axis = new 
CrossPlatformInputManager.VirtualAxis(axisName);
				CrossPlatformInputManager.RegisterVirtualAxis(m_Axis);
			}
			else
			{
				m_Axis = 
CrossPlatformInputManager.VirtualAxisReference(axisName);
			}
			FindPairedButton();
		}

		void FindPairedButton()
		{
			// find the other button witch which this button should be 
paired
			// (it should have the same axisName)
			var otherAxisButtons = 
FindObjectsOfType(typeof(AxisTouchButton)) as AxisTouchButton[];

			if (otherAxisButtons != null)
			{
				for (int i = 0; i < otherAxisButtons.Length; i++)
				{
					if  (otherAxisButtons[i].axisName  ==  axisName  &&
otherAxisButtons[i] != this)
					{
						m_PairedWith = otherAxisButtons[i];
					}
				}
			}
		}

		void OnDisable()
		{
			// The object is disabled so remove it from the cross platform
input system
			m_Axis.Remove();
		}


		public void OnPointerDown(PointerEventData data)
		{
			if (m_PairedWith == null)
			{
				FindPairedButton();
			}
			// update the axis and record that the button has been pressed
this frame
			m_Axis.Update(Mathf.MoveTowards(m_Axis.GetValue,     axisValue,
responseSpeed * Time.deltaTime));
		}


		public void OnPointerUp(PointerEventData data)
		{
			m_Axis.Update(Mathf.MoveTowards(m_Axis.GetValue,             0,
responseSpeed * Time.deltaTime));
		}
	}
}
```

- **The Jump:**

```
using System;
using UnityEngine;

namespace UnityStandardAssets.CrossPlatformInput
{
    public class ButtonHandler : MonoBehaviour
    {

        public string Name;

        void OnEnable()
        {

        }

        public void SetDownState()
        {
            CrossPlatformInputManager.SetButtonDown(Name);
        }


        public void SetUpState()
        {
            CrossPlatformInputManager.SetButtonUp(Name);
        }


        public void SetAxisPositiveState()
        {
            CrossPlatformInputManager.SetAxisPositive(Name);
        }


        public void SetAxisNeutralState()
        {
            CrossPlatformInputManager.SetAxisZero(Name);
        }


        public void SetAxisNegativeState()
        {
            CrossPlatformInputManager.SetAxisNegative(Name);
        }

        public void Update()
        {

        }
    }
}
```

Reference:

[1] Dale Janssen, Cory Janssen , Co-founder Janalta Interactive Inc.,Home page , Techopedia,Role-plying-game

https://www.techopedia.com/definition/27052/role-playing-game-rpg

[2] David L, January 31, 2007,"Behaviorism," in Learning Theories

https://www.learning-theories.com/behaviorism.html

[3] Zodinplex , November 9 ,2018, Google play , Educative, Education , Educational Games for Kids

https://play.google.com/store/apps/details?id=com.zodinplex.abc.kids.letters.educational.sounds.baby&hl=es

[4] Zombie Farm , Zombie Battle., November 2,2018 , Google play , Behavior Interactive, Tree World
https://play.google.com/store/apps/details?id=com.playforge.treeoflife
[5] 2018, Epic Games, Inc. Epic, Epic Games, Homepage , Fortnite

https://www.epicgames.com/fortnite/en-US/patch-notes/v6-31

[6] Andrew Robertson ,June 30, 2018 ,Homepage, Blog, Console, Advice, PEGI, PS4, Xbox One, Video Game Guides, Hot Topic

http://www.askaboutgames.com/parents-guide-to-fortnite-pegi-12/

[7] McConnell, E. (2010). Project Management Methodology: Definition, Types, Examples. Retrieved from my management guide

[8] Rahman, N., Wittman, A., & Thabet, S. (2016). Managing an Engineering Project. International Journal of Information Technology Project Management (IJITPM), 7(1), 1-17

http://www.base36.com/2012/12/agile-waterfall-methodologies-a-side-by-side-comparison/

[9] Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., & Peterson, A. S. (1990). Feature-Oriented Domain Analysis (FODA) Feasibility Study.

[10] John K. Haas**,** Worcester Polytechnic Institute. (2014). A history of the Unity game engine.

https://digitalcommons.wpi.edu/iqp-all/3207/

[11] W3 Total Cache, Dec 8,2018 , Software testing fundamentals, Home page, Test plan

http://softwaretestingfundamentals.com/test-plan/

[12] Khurana, N., Chhillar, R. S., & Chhillar, U. (2016). A Novel Technique for Generation and Optimization of Test Cases Using Use Case, Sequence, Activity Diagram and Genetic Algorithm. JSW, 11(3), 242-250.

[13] Ritsilä, A. (2018). GraphQL: The API Design Revolution.
https://scholar.googleusercontent.com/scholar?q=cache:fGmFSaLpW6UJ:scholar.google.com/&hl=ar&as_sdt=0,5&scioq=www.mulesoft.com/resources/api/what-is-an-api

[14] KAMPUS, B. P. T. S., & MISBAH, M. N. SISTEM INFORMASI KEUANGAN SESUAI STANDART.

[15] Dale Janssen, Cory Janssen, Co-founder Janalta Interactive Inc., 2008, Techopedia, System Design

https://www.techopedia.com/definition/29998/system-design

[16] **J. Maureen Henderson**, 2019,Forbes Media LLC, INTERACTION DESIGN FOUNDATION, User Interface (UI) Design

https://www.interaction-design.org/literature/topics/ui-design

[17] Brian Ashcraft, December 4, 2010, Kotaku, Definitions For 2D And 3D Are Broken

https://kotaku.com/our-definitions-for-2d-and-3d-are-broken-please-fix-5514956

[18] Jonathan Ball , December 22, 2014, Home Page, Creative Bloq, Graphic design

https://www.creativebloq.com/illustration/vector-art-1131698

[19] Unity Technologies. Publication: 2018.3-002T. Built: 2019-03-13. GameObjects

https://docs.unity3d.com/Manual/GameObjects.html

[20] Unity Technologies. Publication: 2018.3-002T. Built: 2019-03-13. Components

https://docs.unity3d.com/Manual/Components.html

[21] Unity Technologies. Publication: 2018.3-002T. Built: 2019-03-13. PhysicalCameras

https://docs.unity3d.com/560/Documentation/Manual/Transforms.html

[22] Unity Technologies. Publication: 2018.3-002T. Built: 2019-03-13.

https://docs.unity3d.com/Manual/PhysicalCameras.html

[23] Unity Technologies. Publication: 2018.3-002T. Built: 2019-03-13. UICanvas

https://docs.unity3d.com/Manual/UICanvas.html

[24] Unity Technologies. Publication: 2018.3-002T. Built: 2019-03-13. PhysicsSection

https://docs.unity3d.com/Manual/PhysicsSection.html

[25] Unity Technologies. Publication: 2018.3-002T. Built: 2019-03-13. ScriptingSection

https://docs.unity3d.com/Manual/ScriptingSection.html

[26] December 1,2015,teme STF, software test fundamental , compliance testing
http://softwaretestingfundamentals.com/unit-testing/

[27] Sandeep Jain, Vaibhav Bajpai, Shikhar Goel, Dharmesh Singh, Shubham Baranwal, Team Geeks for Geeks, unit Testing
https://www.geeksforgeeks.org/types-software-testing/

[28] Sandeep Jain, Vaibhav Bajpai, Shikhar Goel, Dharmesh Singh, Shubham Baranwal, Team Geeks for Geeks, Integration Testing

https://www.geeksforgeeks.org/types-software-testing/