Tribhuvan University

Faculty of Humanities and Social Sciences

APOCALYPSE "A Tower Defense Game"

A PROJECT REPORT

Submitted to

Department of Computer Application

Kathmandu Bernhardt College

In partial fulfillment of the requirements for the Bachelors in Computer Application

Submitted by

Bishwas Shrestha

Symbol no : 4610524

Under the Supervision of

Anil Verma

ABSTRACT

Entertainment has always been a big part of life for everyone. Young or old everyone needs entertainment to enjoy their life and for their mental wellbeing. Apocalypse is a tower defense game with strategic environment for the players to build their own strategy developed using flow field path finding algorithm and collision handling algorithm which is mostly used for tower defense game. Levels are based on the player’s performance and strategy. As level progresses the enemies gets stronger and level of difficulty increases. This game is built by using JavaScript, game maker language, tiled. Strategic games are a kind of entertainment which is sought out by many people. Being competitive and applying the strategy is a part of thrill that people would like to experience in their life. Apocalypse is supposed to provide that strategic experience to the players.

Keywords: JavaScript game, Path finding algorithm, Game maker studio 2, Strategy, Entertainment, Tower defense

ACKNOWLEDGEMENT

I would like to express our sincere gratitude to Project Supervisor Mr. Anil Verma Sir, who continuously helped and guided us through different problems faced during this whole coding process “Apocalypse” on Game Maker Studio. I would like to thank Co-Ordinator Mr. Rambabu Mahato for assisting in the different scope of the database of the programs. I would like to thank Kathmandu Bernhardt College for providing all the resources and ideal environment for the project developing process and encourage us to develop such a project. I would also like to share our gratitude to Tribhuvan University for providing the opportunity to evaluate our programming skills through this project.

Bishwas Shrestha

Table of Contents

Title **Page No.**

ABSTRACT .... **iii**

ACKNOWLEDGEMENT .... **iv**

List of figures .... **viii**

List of abbreviations .... **ix**

CHAPTER 1: INTRODUCTION **1**

1.1 Introduction to Apocalypse 1

1.2 Problem Statements 2

1.3 Objectives 2

1.4 Scope and limitation 2

1.5 Development Methodology 2

1.5.1 The goal of the study 3

1.6 Report Organization 3

CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW **4**

2.1 Study of existing system 4

2.2 Literature review 4

CHAPTER 3: SYSTEM ANALYSIS AND DESIGN **7**

3.1 System Analysis 7

3.1.1Requirement analysis 7

3.1.2 Feasibility Analysis 7

3.1.3 System modeling (Object oriented approach) 8

3.2 System Design 11

3.3 Algorithm Details 13

CHAPTER 4: IMPLEMENTATION AND TESTING **16**

4.1 Implementation 16

4.1.2 Implementation Details of Modules (Description of procedures/functions) .. 17

4.2 Testing 19

CHAPTER 5: CONCLUSION AND FUTURE RECOMENDATION.................21

5.1 Lesson learnt / outcomes...21

5.2 Future recommendations...21

Conclusion 21

APPENDIX

REFERENCES

List of figures

Figure No. **Page No.**

Figure 1 A screenshot of Vector Tower Defense 2 (Candy stand, 2008) 6

Figure 2 class and objects diagram 8

Figure 3 Activity diagram 9

Figure 4 state and sequence diagram 10

Figure 5 Component diagram 12

Figure 6 Working mechanism... 13

Figure 7 Flow field path finding 14

Figure 8 Collision handling 15

Figure 9 Tiled.. 16

List of abbreviations

AI:	Artificial intelligence
DnD:	Drag and drop
GML:	Game maker language
GMS:	Game maker studio
HTML:	Hypertext Markup Language
JS:	JavaScript
TD:	Tower defense
TDG:	Tower defense game

CHAPTER 1: INTRODUCTION

1.1 Introduction to Apocalypse

Apocalypse, a tower defense game for windows-based operating system based on an android game. In Apocalypse player has to stop enemies from reaching the end using their heroes collecting coins stopping enemies. Defeating enemies are rewarded with coins. Powerful heroes cost more coins. Compared to platformers and idle games, Tower Defense seems to get a higher average revenue per paying user. And the average revenue per daily active user is significantly higher, too. This is likely because the average playtime is almost double those other genres.

So when you create a Tower Defense game, the key is making sure your players stick around. Adding new enemies, towers and level layouts can be quite straightforward, but can have a massive impact on the Meta. By trickling out these updates over months, you can keep players coming back and make sure your retention is high.

Tower defense (TD) is a type of games that required player(s) to build tower in order to prevent certain amounts of enemies from approaching their base. It is known as real-time strategy game or turn-based game and several well-known TD games are such as Plants vs Zombies and TD in Warcraft III [1]. This genre of game provide an important testbed for Artificial Intelligence (AI) research by allowing the testing and comparison of new and experimental approaches on a challenging but well-defined problem.

Tower defense is a subgenre of strategy games where the goal is to defend a player's territories or possessions by obstructing the enemy attackers or by stopping enemies from reaching the exits, usually achieved by placing defensive structures on or along their path of attack This typically means building a variety of different structures that serve to automatically block, impede, attack or destroy enemies. Tower defense is seen as a subgenre of real-time strategy video games, due to its real-time origins, even though many modern tower defense games include aspects of turn-based strategy. Strategic choice and positioning of defensive elements is an essential strategy of the genre.

1.2 Problem Statements

Just about all of the most popular video games have some requirement for problem-solving and/or critical thinking? This promotes adaptability and cognitive flexibility. These are really important skills to have in any kind of problem-solving task.

Studies have indicated that compared to non-gamers, experienced gamers are better at: tracking objects; keeping track of several objects simultaneously; filtering out irrelevant information; switching from task to task; detecting changes in visual layouts; and 3D mental rotation.

1.3 Objectives

The major objectives of the project are:

- To develop a game for entertainment purpose.
- To demonstrate unique skills in designing, emerging as well as project management so as to expand modified video games in support of diverse users.

1.4 Scope and limitation

1.4.1 Scope

When people talk about video games development, we often read that the rate of project abandonment is very high and it’s true; it’s fun, attractive, with the right functionality, and with enough marketing to be seen among the many that are published every day.

1.4.2 Limitation

I. System to require DirectX: Version 11 or higher.
II. Requires both keyboard and mouse.
III. Require Microsoft visual C++ 2015.

1.5 Development Methodology

An iterative approach to project management and software development that helps deliver value to their audience faster and with fewer headaches. Instead of betting everything on a launch, an agile process delivers work in small, but consumable, increments. Requirements, plans, and results are evaluated continuously so developer have a natural mechanism for responding to change quickly.

Waterfall model is an example of sequential model. In this model, the software development activity is divided into different phases and each phase consists of a series of tasks and has different objectives. It is divided into phases and output of one phase becomes the input of the next phase. It is mandatory for a phase to be completes before the next phase starts. There is no overlapping in the waterfall model.

1.5.1 The goal of the study

In this study we have used a video game of the Tower Defense genre to detect the game processes associated with mathematical concepts and procedures relevant to students aged from 10 to 12 years. The main goal of the study is use this characterization of the students' gameplay as a problem solving activity to identify mathematical learning opportunities that can be promoted while playing.

1.5.2 Description of the video game used

Tower Defense is a sub–genre of real–time strategy video games. The goal is to prevent enemy units, who arrive in waves, from crossing the map and attacking our base. To achieve this, defense towers have to be built which can assault enemy units as they pass. Funds are earned for eliminating each enemy unit, and these must be used to build or reinforce towers, so that strategic considerations are based on the choice and placement of the towers and resource management. There is a great variety of games in this genre but, in all of them, both the enemies and the defensive towers have different offensive and defensive skills, and each has a different cost. The video game used in the qualitative study is Vector Tower Defense 2, by Candy stand. The aesthetics of the game is neutral, and there are no elements of violence, making it suitable for students from the age of 10. One of the features of this video game design is that it includes bonus points at the end of each wave of enemy attack – a specific percentage of the banked money – so that choosing the right moment to invest in new towers is yet another element of the game's strategic aspects.

1.6 Report Organization

The remainder of this report proceeds as follows. 1.1 summarizes the introduction for the tower defense. 1.2 provides a summary of problem statements .1.3 provides the objectives. 1.4 includes scope and limitations. Finally, 1.5 summarizes the methodology for developing the tower defense, which are key inputs for the analysis.

CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Study of existing system

According to Aura Hernández-Sabaté1, Meritxell Joanpere1, Núria Gorgorió1, Lluís Albarracín1 1* Universität Autònoma de Barcelona, in this study we have used a video game of the Tower Defense genre to detect the game processes associated with mathematical concepts and procedures relevant to students aged from 10 to 12years. The main goal of the study is using this characterization of the students' gameplay as a problem-solving activity to identify mathematical learning opportunities that can be promoted while playing. The video game used in the qualitative study is Vector Tower Defense 2, by Candystand.I thas been accessible on the web since 2008. The aesthetics of the game is neutral, and there are no elements of violence, making it suitable for students from the age of 10. In this game, the enemy has to move along a specific path, and players have a large variety of towers, upgrades and bonus points, prompting them to exploit their strategic skills to the limit. One of the features of this video game design is that it includes bonus points at the end of each wave of enemy attack – a specific percentage of the banked money – so that choosing the right moment to invest in new towers is yet another element of the game's strategic aspects. The game provides players with information on different levels. The panel containing the most information is the game map, which takes up the whole central part. This area shows the path the enemy must take the position of the towers, the shots taken from the towers and the bonus objects on them. General information is provided at the top of the screen, such as the number of the next enemy wave, available funds, the interest percentage they generate and the lives remaining. The different towers that can be used at each moment are shown on the right, including a description of them and details of the enemy. Figure 2 displays a description of one of the towers, showing the cost, the damage it inflicts in each attack and the bonus points for the various types of towers expressed as percentages [1]

2.2 Literature review

Tower Defense is a sub–genre of real–time strategy video games. The goal is to prevent enemy units, who arrive in waves, from crossing the map and attacking our base. To achieve this, defense towers have to be built which can assault enemy units as they pass.

Funds are earned for eliminating each enemy unit, and these must be used to build or reinforce towers, so that strategic considerations are based on the choice and placement of the towers and resource management. There is a great variety of games in this genre but, in all of them, both the enemies and the defensive towers have different offensive and defensive skills, and each has a different cost. The video game used in the qualitative study is Vector Tower Defense 2, by Candystand. It has been accessible on the web since 2008. The aesthetics of the game is neutral, and there are no elements of violence, making it suitable for students from the age of 10. In this game, the enemy has to move along a specific path, and players have a large variety of towers, upgrades and bonus points, prompting them to exploit their strategic skills to the limit. One of the features of this video game design is that it includes bonus points at the end of each wave of enemy attack – a specific percentage of the banked money – so that choosing the right moment to invest in new towers is yet another element of the game's strategic aspects. [2]

According to Adrian Rusu, Robert Russell, Edward Burns, and Andrew Fabian from Department of Computer Science, Rowan University Younger audiences find our real-time strategy, tower-defense game very appealing, as it is engaging, immersive, and fun. We use metaphors to simulate similar strategies that would be used in a real software project to perform software maintenance. Other educational software engineering games simulate general aspects of software engineering from a project management perspective.[3]

Kebritchi, Hirumi and Bai used a commercial video game designed specifically to foster the learning of mathematics. The latter is based on the formulation of mathematical questions and problems akin to those commonly used in classrooms.[4]

Van den Heuvel–Panhuizen, Kolovou and Robitzsch used a dynamic online game designed to exercise early algebra problem solving. This video game proposes a sequence of problems to the students and monitors their progress. Their results show that online work improves their marks in problem–solving, but does so by using strategies that are approached differently in classroom environments, such as in trial–and–error strategy. [5]

Chow, Woodford and Maes, used an online version of the game "Deal or not deal" which is similar to tower defense with similar concept and confirm that students improve their knowledge on expected values in an introductory statistics course at the same time as they improve their overall ability to process information and make logical decisions. [6]

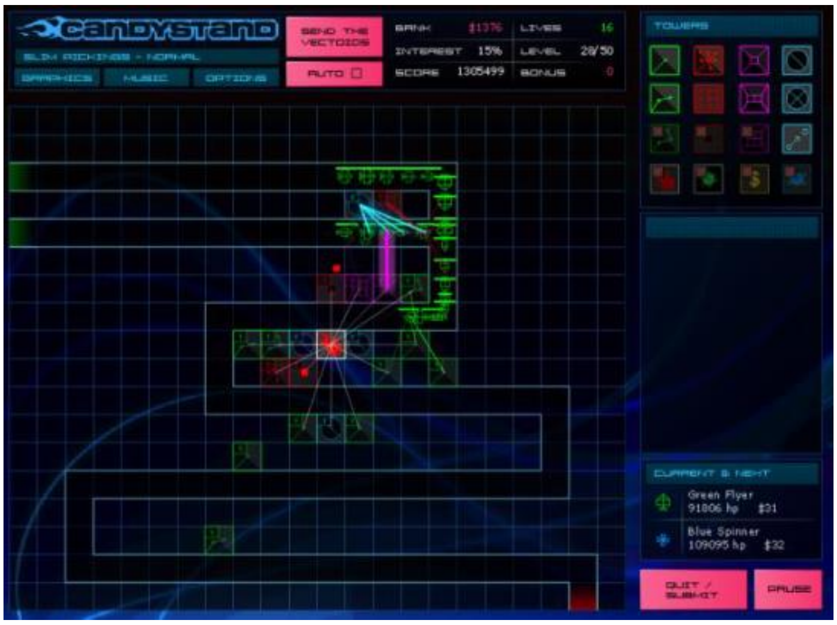


Figure 1 A screenshot of Vector Tower Defense 2 (Candy stand, 2008)

CHAPTER 3: SYSTEM ANALYSIS AND DESIGN

3.1 System Analysis

3.1.1Requirement analysis

Requirement analysis results in the specification of operational characteristics of software: indicates interface of software with other system elements and establishes constrains the software must meet. The requirement analysis is mainly categorized into two types functional and non-functional:

A. Functional requirements

- GUI: User friendly interface are built for easier understanding including menu screen and help screen.
- Logs List: Create a list of error log if game crashes or unable to execute.
- Portable: Since the game does not require a lot of storage it is portable and can be shared though storage devices.

B. Non-functional requirements

- Scalability: Modification and upgrade can be done in future.
- Security: No special permission are needed to execute the file.
- Usability: Efficiency to use because help and tips rooms are developed for user to get idea of the game.

3.1.2 Feasibility Analysis

a. Technical feasibility

Technical feasibility assesses the current resources (hardware and software) and technologies, which are required to accomplish user requirements. It requires a computer with a windows-based operating system installed. Today every organization has computer, so it is not an extra cost. Performance may differ from the specifications of the machines.

b. Economic Feasibility

No any payment features are included since it is created using free tools and applications. Completely free in nature.

c. Operational feasibility

Since it is created using all the free tools and applications it will be available to all the users that are interested in games.

3.1.3 System modeling (Object oriented approach)

3.1.3.1 Object Modelling: Object & Class Diagram

Class diagram shows your classes and their relationships. An Object Model Diagram shows the interaction between objects at some point, during run time. A Class Diagram will show what the Objects in your system consist of (members) and what they are capable of doing (methods) mostly static.

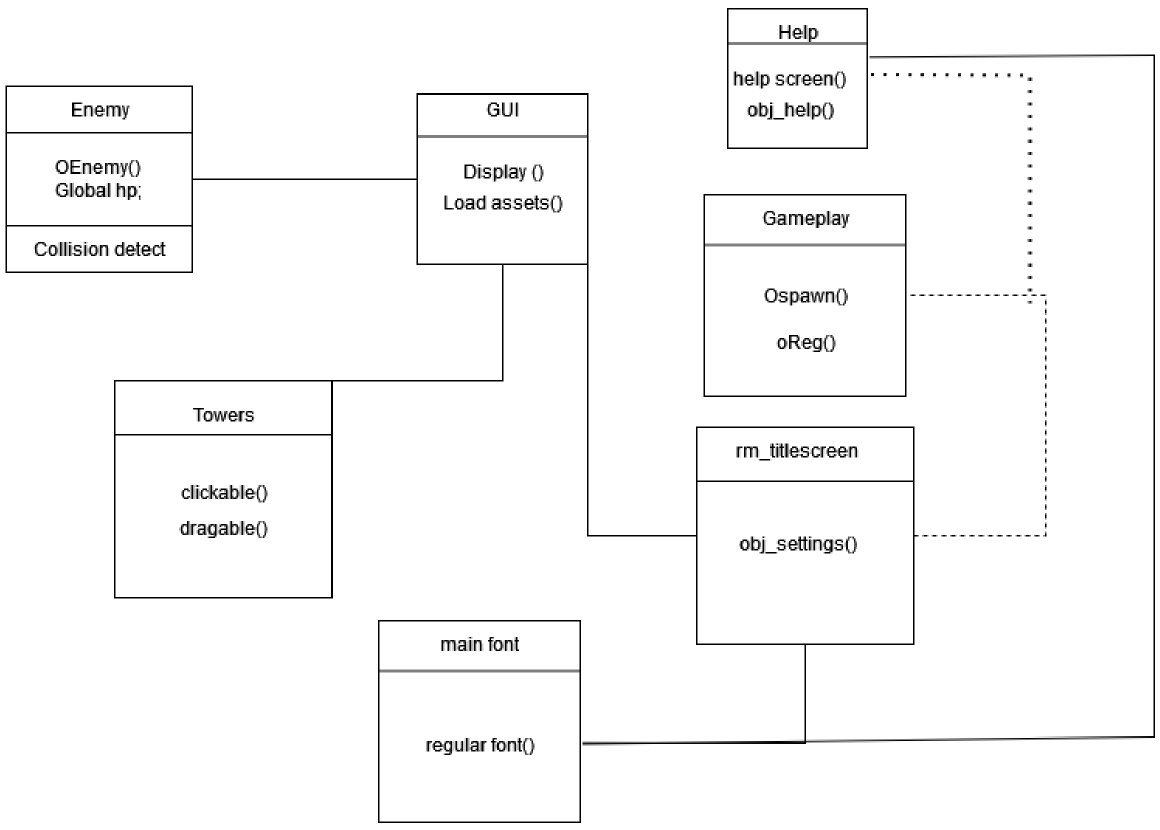


Figure 2 class and objects diagram

3.1.3.2 Activity diagram

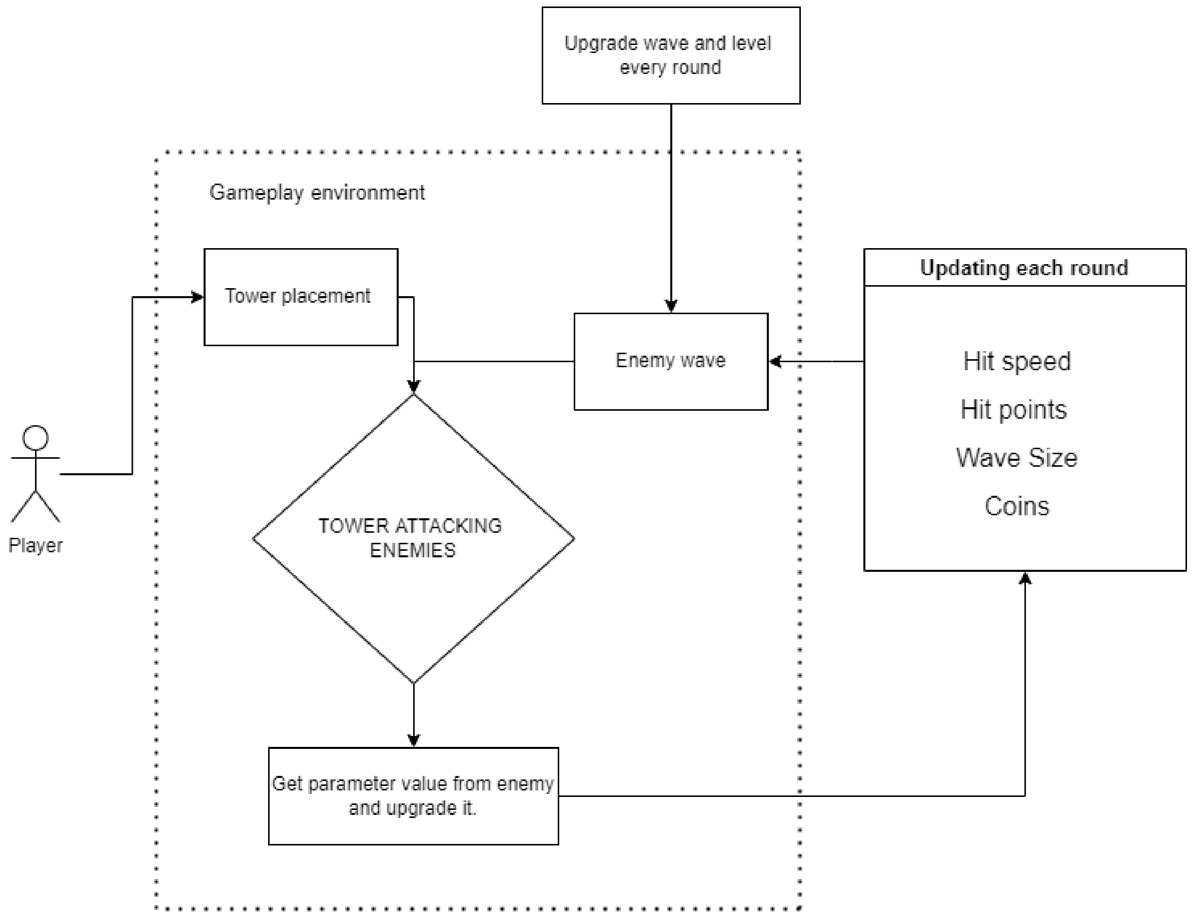


Figure 3 Activity diagram

The game play environment is interacted by the player and the functions are done according to the player input. Player are able to deploy the tower and decide the tower placements on the map where as the enemy progresses and the enemy waves are upgraded. The objects that updates each round are enemy hit speed, hit points, wave size and coins.

3.1.3.3 Dynamic Modelling: State & Sequence Diagram

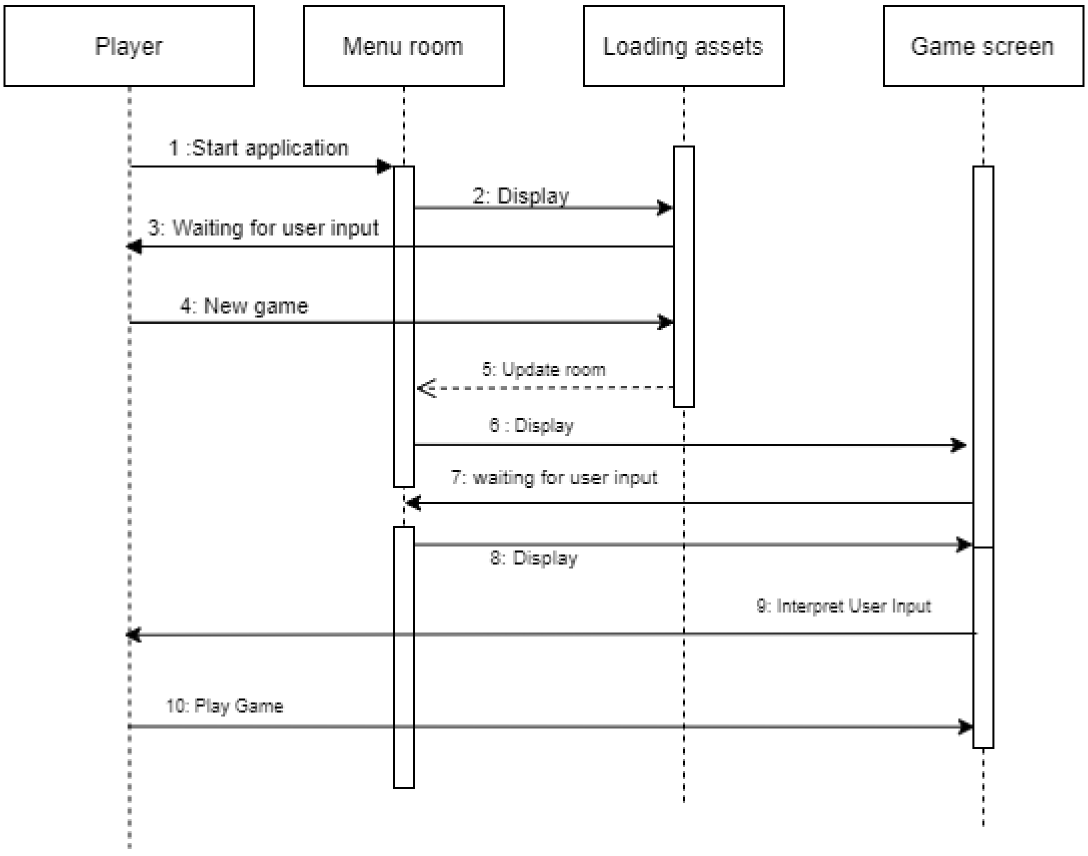


Figure 4 state and sequence diagram

According to above sequence diagram the components in rectangle boxes are objects that are used in the game and whereas the dashed vertical line shows the sequential events that occur to an object during the charted process. The starting process is the execution of the program i.e. start application and the ending process is playing game which is playable due to the process in game screen object.

3.2 System Design

3.2.1 Refinement of Classes and Object

Constructing new classes from existing ones by inheritance or sub classing is a characteristic feature of object-oriented development. Imposing semantic constraints on sub classing allows us to ensure that the behavior of super classes is preserved or refined in their subclasses. This paper defines a class refinement relation which captures these semantic constraints. The class refinement relation is based on algorithmic and data refinement supported by Refinement Calculus. Class refinement is generalized to interface refinement, which takes place when a change in user requirements causes interface changes of classes designed as refinements of other classes. We formalize the interface refinement relation and present rules for refinement of clients of the classes involved in this relation.

Generalisation is the refinement of a class into more refined classes. Generalisation allows the developer to model objects into hierarchical structures based on their similarities. The class being refined is called super-class and the refined versions of it are called sub-classes. Each sub-class inherits the attributes and operations from their super-class. Methods and attributes can then be refined and the sub-class also adds specific attributes and operations. A discriminator is a variable of enumeration type, which indicate which property of an object is being abstracted. The most important use of inheritance is the conceptual simplification it makes trough the reduction of independent features in the system. Generalisation refers to the relationship among classes and inheritance refers to the method of reusing attributes and operations in the hierarchic. By defining a feature with the same name a subclass can override a superclass feature. Overriding is a process of refining and replacing the overridden feature. Generalisation is used for extension and restriction. [8]

3.2.1 Component Diagram

A component diagram, also known as a UML component describes the organization and wiring of the physical components in a system.

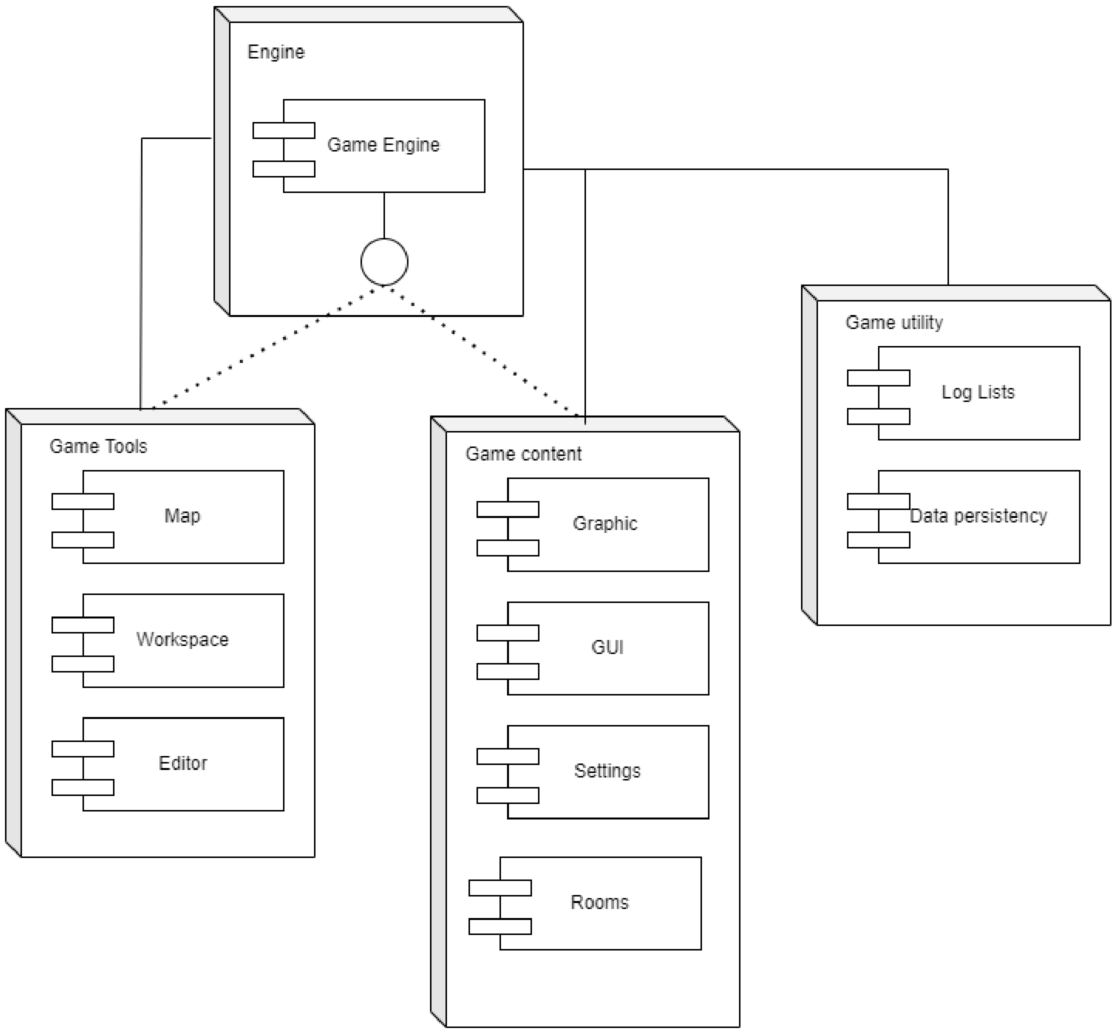


Figure 5 Component diagram

3.3 Algorithm Details

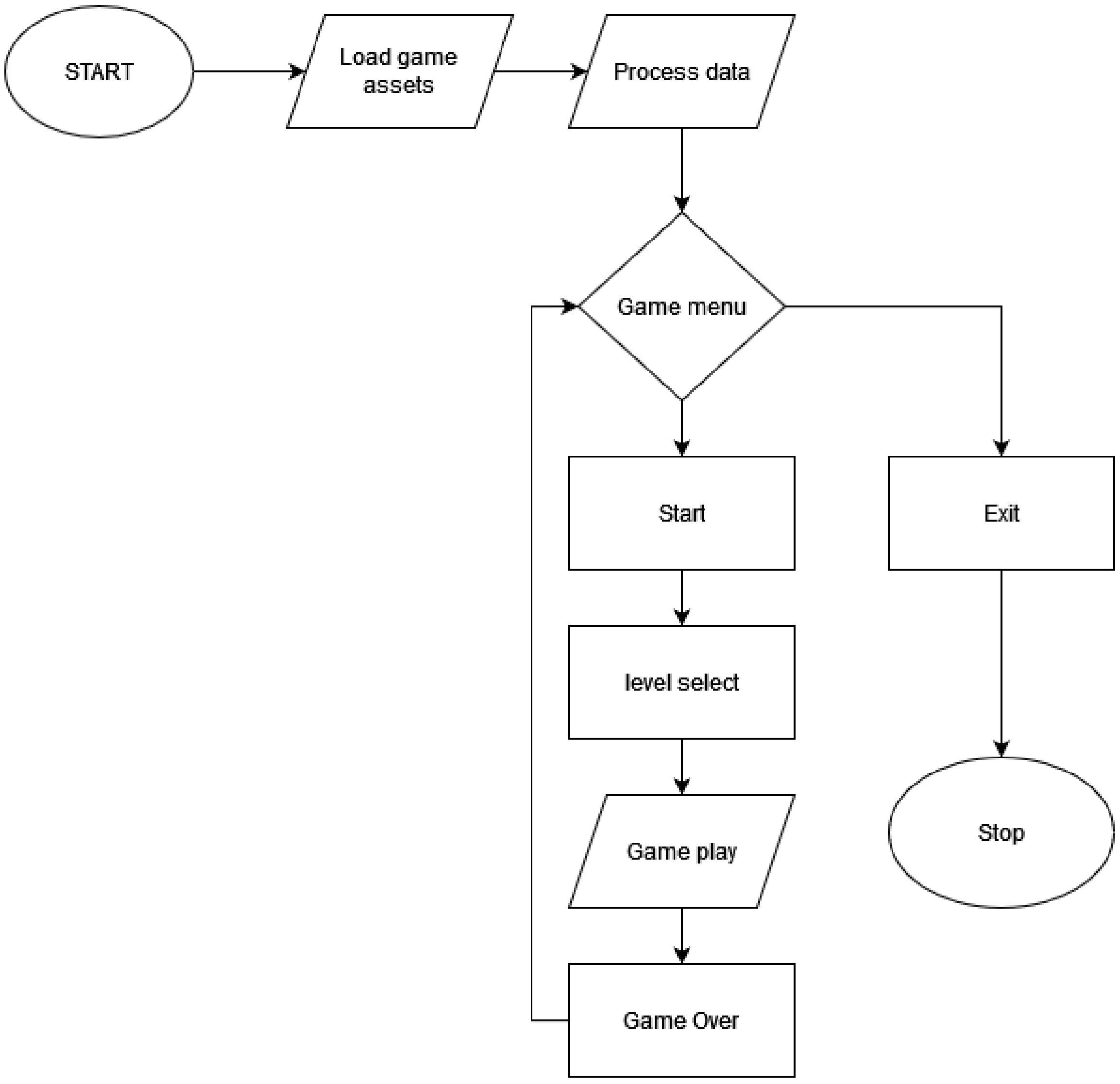


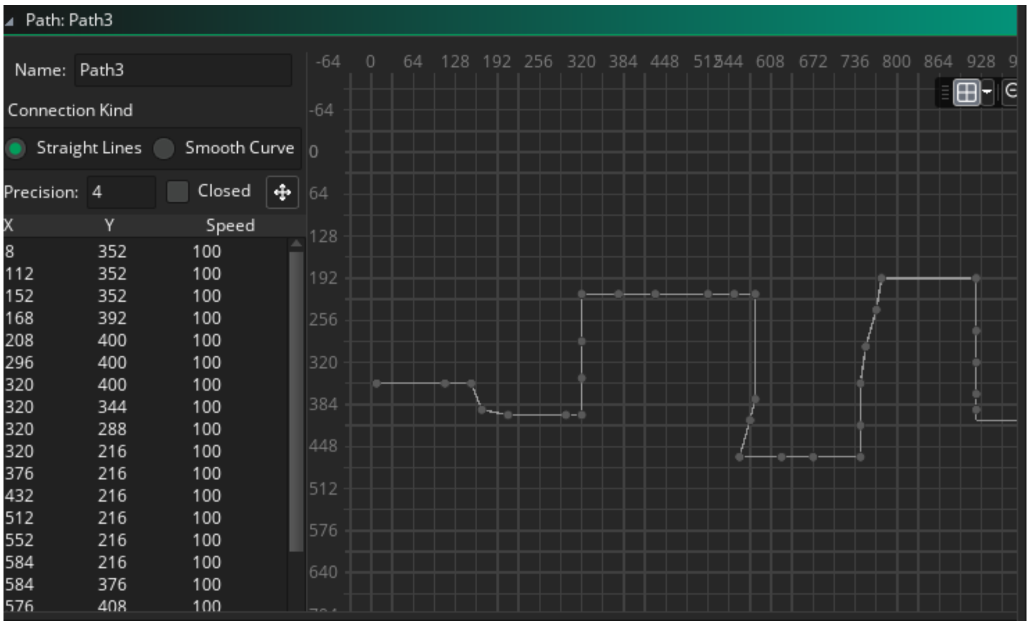
Figure 6 Working mechanism

Flow Field Path finding algorithm for Tower Defense

In a Tower Defense game, there are many enemies that are all headed to the same place. In many Tower Defense games, there is a predetermined path, or a small number of paths. In some, such as the classic desktop tower defense, you can place towers anywhere, and they act as obstacles that affect the paths taken by enemies.

Graph search algorithms like A* are often used to find the shortest path from one point to another point. You can use this for each enemy to find a path to the goal. There are lots of different graph search algorithms we could use in this type of game. To solve this problem we need either a *vector field* (also called a flow field).

A game like Desktop Tower Defense has lots of enemy positions (sources) and one destination for all of them. This puts it into the **all sources, one destination** category. Instead of running A* once per enemy, we can run an algorithm once, and it will calculate the path for all enemies. Even better, it will calculate the shortest path from every location, so as enemies get jostled around or new enemies created, their paths have already been computed. This is sometimes called a **flow field** algorithm. [8]



Path: Path3

Name: Path3

Connection Kind

[✓ Straight Lines] Smooth Curve

Precision: 4 Closed

X	Y	Speed
8	352	100
112	352	100
152	352	100
168	392	100
208	400	100
296	400	100
320	400	100
320	344	100
320	288	100
320	216	100
376	216	100
432	216	100
512	216	100
552	216	100
584	216	100
584	376	100
576	408	100

Figure 7 Flow field path finding

```
frontier = Queue()
frontier.put(start )
came_from = dict()
came_from[start] = None

while not frontier.empty():
  current = frontier.get()
  for next in graph.neighbors(current):
    if next not in came_from:
      frontier.put(next)
      came_from[next] = current
```

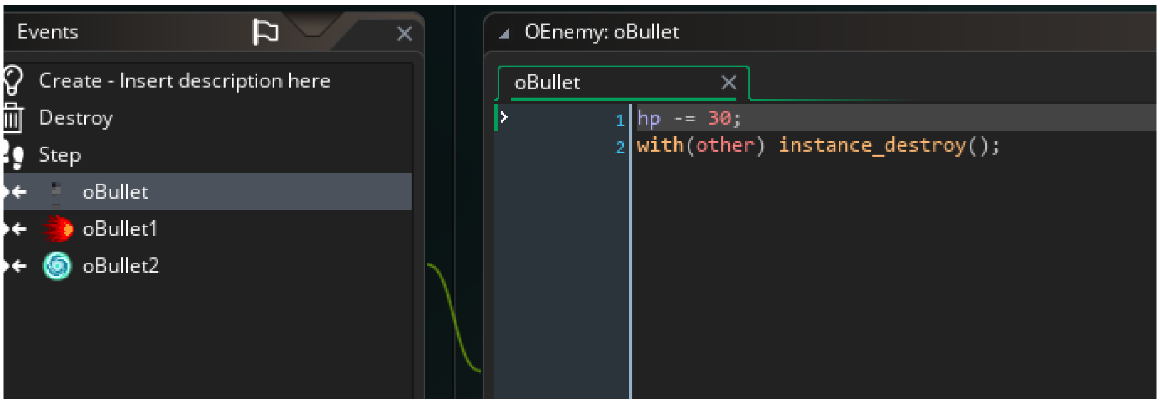
Mathematically,

$$d'(x, y) = d(x, y) + h(y) - h(x).$$

Collision handling algorithm

Collision detection and response are important for interactive graphics applications such as vehicle simulators and virtual reality. Unfortunately, previous collision detection algorithms are too slow for interactive use. The paper presents a new algorithm for rigid or articulated objects that meets performance goals through a form of time critical computing. The algorithm supports progressive refinement, detecting collisions between successively tighter approximations to object surfaces as the application allows it more processing time. The algorithm uses simple four dimensional geometry to approximate motion, and hierarchies of spheres to approximate three dimensional surfaces at multiple resolutions. In a sample application, the algorithm allows interactive performance that is not possible with a good previous algorithm. In particular, the new algorithm provides acceptable accuracy while maintaining a steady and high frame rate, which in some cases improves on the previous algorithm's rate by more than two orders of magnitude. [7]

In Apocalypse flow field path finding algorithm and collision handling are used on the enemy sprite. When the game is loaded the declared path is traveled by the enemy sprite and similarly the collision handling algorithm is used on the enemy sprite as: when the enemy is collided to the bullet object the enemy ‘hp’ variable value is reduced by the given number to the bullet. For example: if the ’hp’ of enemy sprite is 100 and constant stored in a bullet object is 20 when bullet object is collided to the enemy sprite its ‘hp’ is reduced by 20.



```
hp -= 30;
with(other) instance_destroy();
```

Figure 8 Collision handling

CHAPTER 4: IMPLEMENTATION AND TESTING

4.1 Implementation

4.1.1 Tools Used (CASE Tools, Programming Languages, Database platforms)

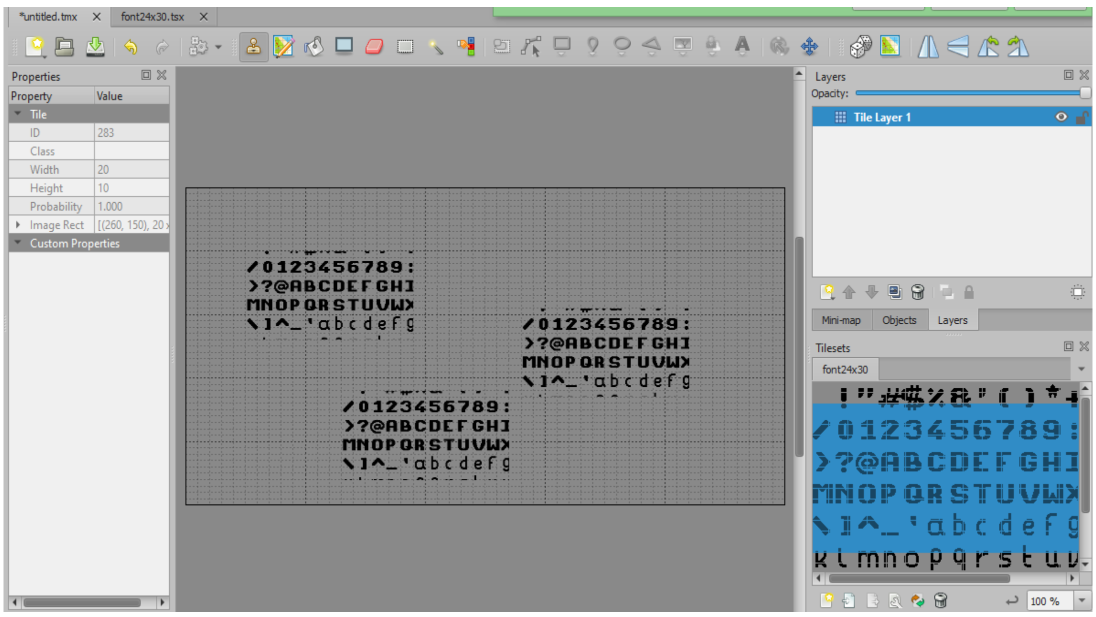
The different tools & programing language used in project:

- **Diagram Tool:**

  The components of the system, and the flow of the data and control between these components are demonstrated by diagram tools by using graphs. "Draw.io" and "wireframe" are the diagram tool used in the project.

- **Tiled:**

  Tiled is a 2D level editor that helps you develop the content of your game. Its primary feature is to edit tile maps of various forms, but it also supports free image placement as well as powerful ways to annotate your level with extra information used by the game. Tiled focuses on general flexibility while trying to stay intuitive.



Property	Value
Tile	
ID	283
Class	
Width	20
Height	10
Probability	1.000
Image Rect	[(260, 150), 20
Custom Properties	

Figure 9 1 Tiled

- **Game maker language:**

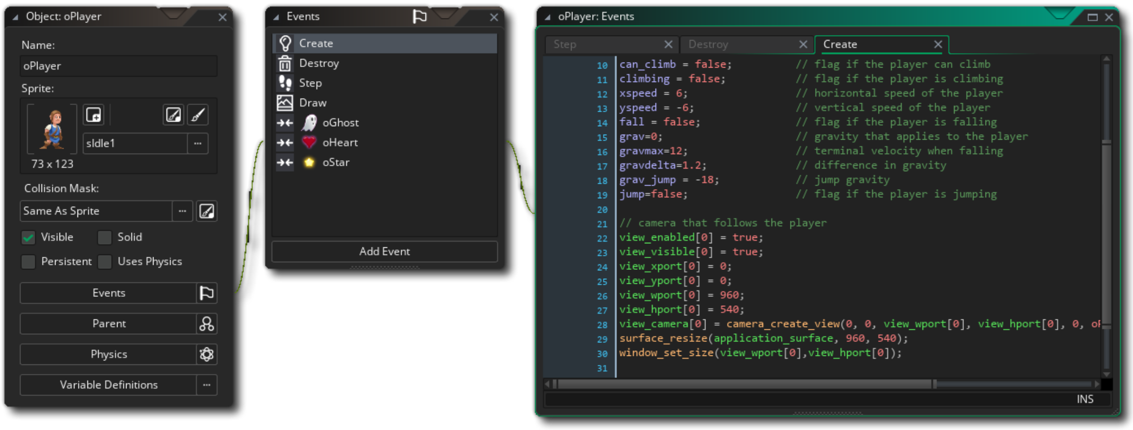
  The Game Maker Language (also called simply GML) is the proprietary Game Maker **scripting language**. This language is structured to permit users to create their games in an intuitive and flexible way while offering all the power of any other major programming language.

- **GML Visuals:**

This is Game Maker’s **Visual Scripting** method that uses blocks of *actions* which are chained together to create your game's logic.

- **GML code**

  Write GML code using functions, variables and other coding constructs, and create your game's logic.



Name: oPlayer
Sprite: sIdle1
73 x 123
Collision Mask: Same As Sprite
[✓ Visible] Solid
Persistent Uses Physics
Events
Parent
Physics
Variable Definitions

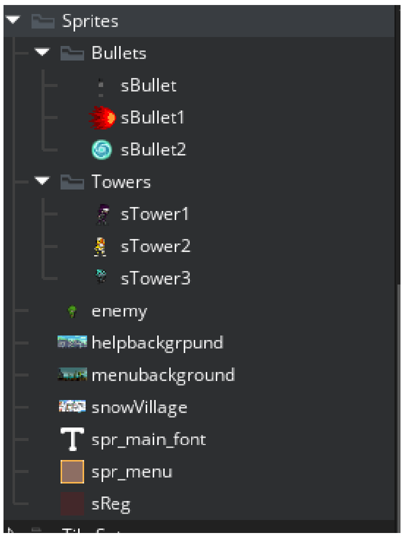
```
can_climb = false;          // flag if the player can climb
climbing = false;           // flag if the player is climbing
xspeed = 6;                 // horizontal speed of the player
yspeed = -6;                // vertical speed of the player
fall = false;               // flag if the player is falling
grav=0;                     // gravity that applies to the player
gravmax=12;                 // terminal velocity when falling
gravdelta=1.2;              // difference in gravity
grav_jump = -18;            // jump gravity
jump=false;                 // flag if the player is jumping

// camera that follows the player
view_enabled[0] = true;
view_visible[0] = true;
view_xport[0] = 0;
view_yport[0] = 0;
view_wport[0] = 960;
view_hport[0] = 540;
view_camera[0] = camera_create_view(0, 0, view_wport[0], view_hport[0], 0, oP
surface_resize(application_surface, 960, 540);
window_set_size(view_wport[0],view_hport[0]);
```

Figure 10 GML

4.1.2 Implementation Details of Modules (Description of procedures/functions)

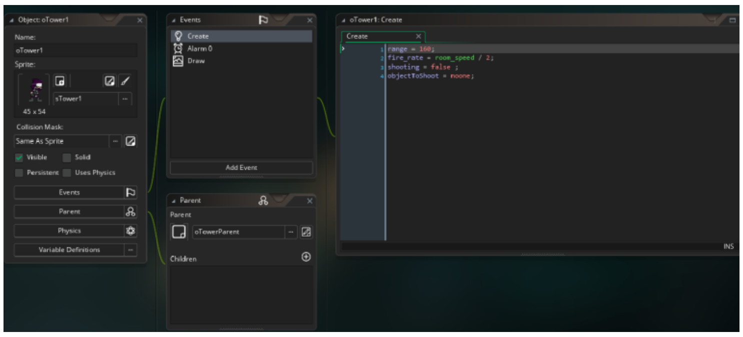
Creating game assets and animated sprites. Here we use the entities that are used in games like images, characters, audio, effects etc.



Sprites
Bullets
sBullet
sBullet1
sBullet2
Towers
sTower1
sTower2
sTower3
enemy
helpbackgrpund
menubackground
snowVillage
spr_main_font
spr_menu
sReg

Figure 11 Game assets sprites

Creating sprite objects


Object: oTower1
Name: oTower1
Sprite: sTower1
45 x 54
Collision Mask: Same As Sprite
[✓ Visible] Solid
Persistent Uses Physics
Parent: oTowerParent

```
range = 160;
fire_rate = room_speed / 2;
shooting = false ;
objectToShoot = noone;
```

Figure 12 creating objects

Creating rooms for user to move through

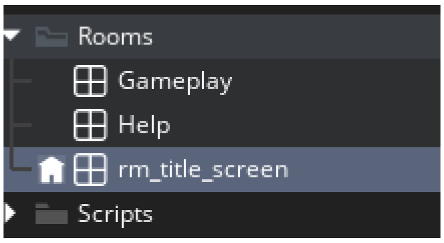
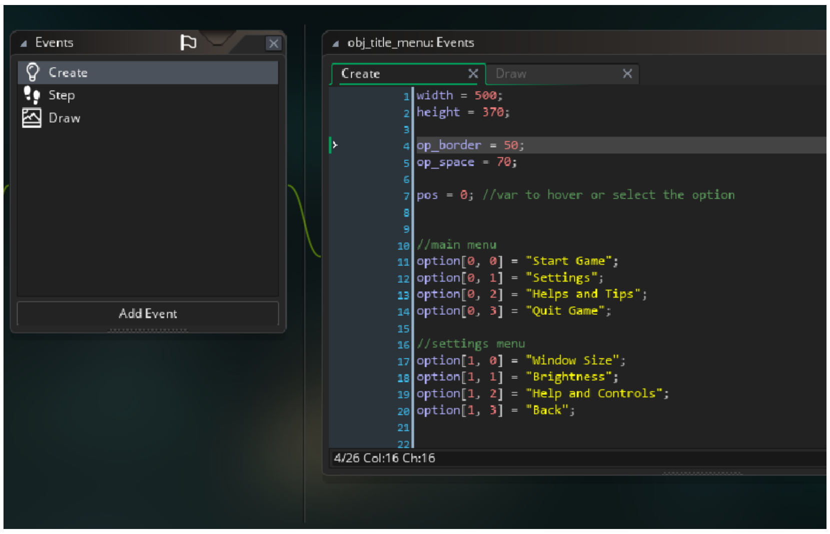


Figure 13 game rooms

Creating menu using multi-dimensional array



```
width = 500;
height = 370;

op_border = 50;
op_space = 70;

pos = 0; //var to hover or select the option


//main menu
option[0, 0] = "Start Game";
option[0, 1] = "Settings";
option[0, 2] = "Helps and Tips";
option[0, 3] = "Quit Game";

//settings menu
option[1, 0] = "Window Size";
option[1, 1] = "Brightness";
option[1, 2] = "Help and Controls";
option[1, 3] = "Back";


```

4/26 Col:16 Ch:16

Figure 14 menu list

4.2 Testing

It’s a process of executing a program or application with the intent of finding the software bugs. It can be also stated as the process of validating and verifying that a software program or application or product meets the business and technical requirements that guided its design and development. Testing can be done using varieties of level.

Table 1: Test Cases for Unit Testing

S.N.	Test Cases	Test Data Input	Expected Outcome	Obtained Test Result
1.	Load assets	Import assets	Write chunks	Pass
2.	Check with invalid assets	No any object	Idle system	Error output
3.	Check with larger assets	Higher resolution assets	Maximum size extend	Code editor crashes
4.	Check with empty room	Empty room with no objects	Blank screen	Pass
5.	Check audio	Input audio files	Should be able to play audio files	fail
6	Move over menu	W to move up S to move down	Move through menu	Pass
7	Select an option	Enter button	Change between rooms	Pass
8	Increase enemy by 2 every round	Add a loop to the enemy object	Increase enemy number	Pass

4.2.2 Test Cases for System Testing

System testing during development involves integrating components to create a version of the system and then testing the integrated system.

Table 2: Test Cases for System Testing

S.N.	Test Cases	Test Data Input	Expected Outcome	Obtained Test Result
1.	Drag and drop character	Use left click for drag and drop	Characters can be placed	Characters are placed in the map
2.	User direction arrows for moving through menu	Key input direction arrows	Can move through menu	User is able to move through menus
3.	Use space bar to select	Click space bar to select	Execute the menu	Move to desired menu
4	Spawn enemy	Ospawn Object	Spawn at located object	Pass
5	End Game	Collision to the end	Game Over	Fail

CHAPTER 5: CONCLUSION AND FUTURE RECOMENDATION

5.1 Lesson learnt / outcomes

There have been several improvements in our programming language and writing skills as well as our time management skills before getting ready for this project. I conclude that this project has helped me gain more knowledge about the topic that we are indulged ourselves into “Game maker studio 2”. A lot was learned about proper time management as the project had to be submitted before the deadline along with the documentation due to time constraints. Although it is expectedly good, some new features to this system could be added in the upcoming days to make it more user friendly and efficient.

5.2 Future recommendations

In the near future the game can be upgraded and will get some new feature and fixes. Minimizing bugs and errors, getting new facilities to the player to store individual scores and make it even more competitive. The GUI of the project will also be improved and include more complicated algorithm or upgrading the gaming performance.

Conclusion

This project is done as Efficient as possible. The description of the background and the context of the project was thoroughly researched by the author. The purpose, scope, applicability, and requirement specifications of the system have been accurately explained. The author has included features and operations in detail including screen layouts and the limitations on which the project is being developed. Finally, the system is implemented and tested according to test cases. After the development of the system finally, it was tested and the views about results were exchanged. After testing, the limitations of the existing system were discussed. In conclusion, tools like Game Maker Studio, Tiled and backend tools like JavaScript were used in the development of my system.

APPENDIX

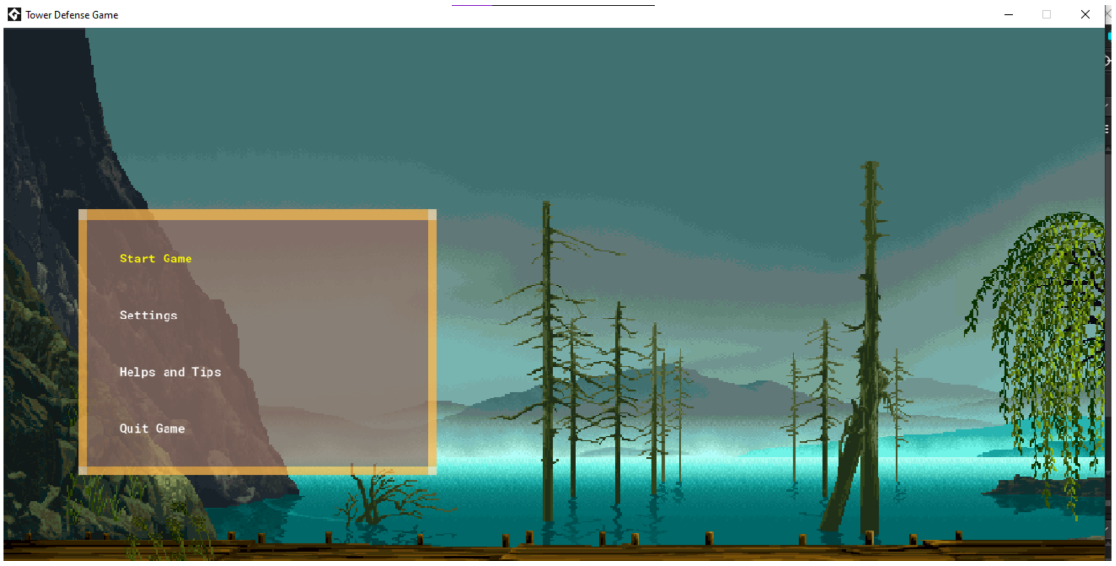


Figure 16 Gameplay 1

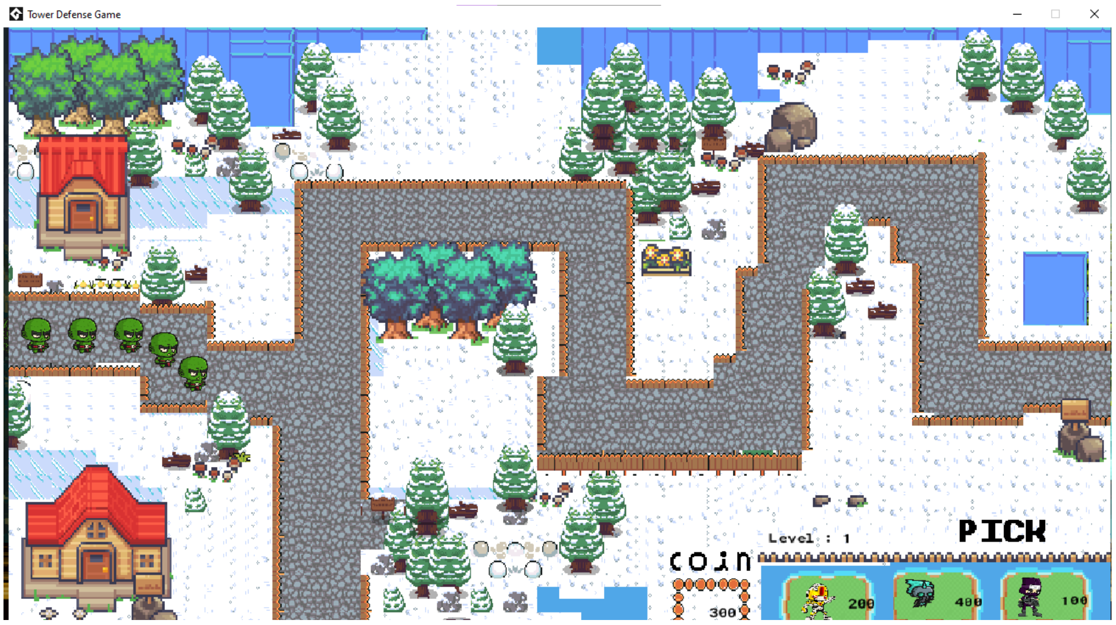


Figure 16 Gameplay 2

REFERENCES

[1]https://www.researchgate.net/profile/Lluis-Albarracin/publication/286913995_Mathematics_learning_opportunities_when_playing_a_Tower_Defense_Game/links/567331f108ae1557cf494de2/Mathematics-learning-opportunities-when-playing-a-Tower-Defense-Game.pdf

[2]https://www.researchgate.net/profile/Lluis-Albarracin/publication/286913995_Mathematics_learning_opportunities_when_playing_a_Tower_Defense_Game/links/567331f108ae1557cf494de2/Mathematics-learning-opportunities-when-playing-a-Tower-Defense-Game.pdf

[3] https://link.springer.com/chapter/10.1007/978-3-642-23456-9_32

[4] http://dx.doi.org/10.1016/j.compedu.2010.02.007 Kebritchi, M., Hirumi, A., & Bai, H., The effects of modern mathematics computer games on mathematics achievement and class motivation, Computers & Education, Vol. 55, Nr. 2, 2010.

[5] http://dx.doi.org/10.1007/s10649-013-9483-5 Van den Heuvel-Panhuizen, M., Kolovou, A., & Robitzsch, A., Primary school students' strategies in early algebra problem solving supported by an online game, Educational Studies in Mathematics, Vol. 84, Nr. 3, 2013.

[6] http://dx.doi.org/10.1080/0020739X.2010.519796 Chow, A. F., Woodford, K. C., & Maes, J., Deal or no deal: using games to improve students learning, retention and decision making, International Journal of Mathematical Education in Science and Technology, Vol. 42, Nr. 2, 2010.

[7] https://ieeexplore.ieee.org/abstract/document/466717

[8] https://link.springer.com/chapter/10.1007/3-540-63533-5_5