Turris

Portfolio

Computer Science

COMP208: Group Software Project

Team 62

Oliver Legg – Kieran Baker – Thomas Coupe – Daniel Taylor – Alessandro Wang

Contents

Design

Summary

Our group intends to make a tower defence game that can entertain users for hours. The overall goal of the game is to:

- Entertain
- Challenge
- Intrigue
- Escape
- Compete

To complete these goals, we would like to give an experience that the user wants from a game. Not everyone would want to compete – therefore we'd like to give them choices and options within the game. To tackle these problems, we have some main objectives to achieve this:

- Create a satisfying experience for the user to interact with the game and impress them with gameplay quality, graphics and sound effects.
- Create an environment in the game that will challenge the user's ability. The user can challenge themselves by settings the difficulty from easy, medium and hard.
- Intrigue the user on what happens when you win the game or play a level with their own strategy.
- Create an environment of the game and overall experience that will captivate the user and help them to escape.
- Create endless mode which allows the user to have a certain time that they survive for. This can be compared with another player's score – this will bring a competitive aspect of the game.

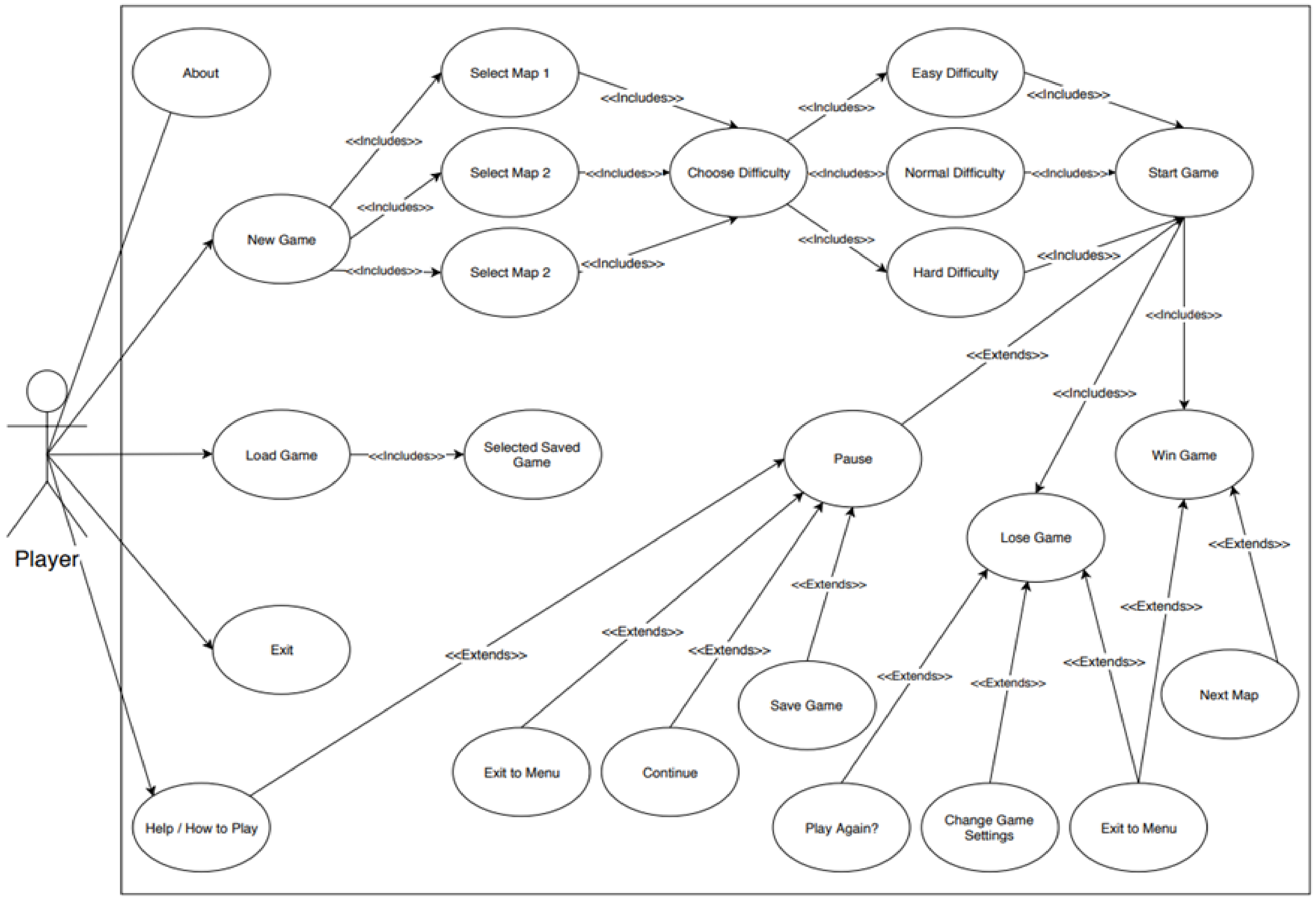
We want the user to feel good when playing the game and want them to feel rewarded for playing well. To do this, we have we do have to difficulty aspect and loss aspect, if you can only win at the game, it will feel undeserved and meaningless.

Business rules

- To reduce storage space, each user will be limited to one save file which will store data for a single game. If the user wishes to save a different game, the old file must be deleted.
- The save file must only contain the relevant data that is required for the save such as the remaining lives, wave number and turret locations, any extra information should be omitted to reduce file size on the disk.
- In the event of an error occurring while the user is attempting to save or load their game any data for the current save file should not be modified or deleted.
- The name that the player enters must be at least three characters long and no longer than 15 characters. The name entered must only contain letters and numbered and must not have any special characters in it.
- Only 5000 enemies can be spawned as a time as this would be too computationally intensive to process for our users. This would also consume a lot of memory in the computer, eventually slowing it down.

- As there will be a grid-based placement system, there will be a certain number of towers to be placed on the map. If the screen in 800x600 and each tile is 100x100, there will be 48 tiles (8x6 = 48) and a lot of these will be taken up by paths. Therefore, there is a constraint on how many towers you can place on a map at a time.

Use Cases


Player
About
New Game
Load Game
Exit
Help / How to Play
Select Map 1
Select Map 2
Select Map 2
Selected Saved Game
Choose Difficulty
Easy Difficulty
Normal Difficulty
Hard Difficulty
Start Game
Win Game
Lose Game
Pause
Save Game
Continue
Exit to Menu
Play Again?
Change Game Settings
Exit to Menu
Next Map
<<Includes>>
<<Extends>>

Use Case Descriptions

ID	**UC 1**
Name	New Game
Description	Allow the player to start a new game
Actor	Player
Pre-condition	Game is launched and player ready to play
Post-condition	Map options displayed
Main flow	1. System displays available maps
Includes	UC 5 (Select Map)
Extensions	UC 17 (Play Again?)

ID	**UC 2**
Name	Load Game
Description	Allow the player to start a saved game
Actor	Player
Pre-condition	The player Saved previously a started game
Post-condition	Map options displayed
Main flow	1. Player select desired saved game 2. Player loads in the desired saved game
Includes	Selected Saved Game
Extensions	None

ID	**UC 3**
Name	Help/How to play
Description	The system will display a rules and mechanics of the game It can be also selected when you pause during a game
Actor	Player
Pre-condition	Game is launched
Post-condition	Help and guides displayed
Main flow	1.Player get initial idea of how the game work and how to play it
Includes	None

Extensions	None

ID	**UC 4**
Name	Exit
Description	Allow the player to shut down the game and back to the desktop
Actor	Player
Pre-condition	Game is launched and plyer not willing to play
Post-condition	Game is exited
Main flow	1. Player stops desiring to play 2. Game exited
Includes	None
Extensions	None

ID	**UC 5**
Name	Select Map
Description	Allow the player to choose a desired map
Actor	Player
Pre-condition	Game is launched and player ready to play
Post-condition	Map options displayed
Main flow	1. Player given option to select map
Includes	UC 6 (Choose difficulty)
Extensions	None

ID	**UC 6**
Name	Choose Difficulty
Description	Allow the player to pick the desired difficulty in a new game
Actor	Player
Pre-condition	Player has already chosen the map
Post-condition	Map difficulties displayed
Main flow	1. Player given option to select map difficulty
Includes	UC 7, UC8, UC 9

Extensions	None

ID	**UC 7**
Name	Easy Difficulty
Description	Player is playing game on easy
Actor	Player
Pre-condition	Player has chosen the difficulty
Post-condition	Map ready to start
Main flow	1. Player selected difficulty 2. Map ready to start
Includes	UC 10 (Start Game)
Extensions	None

ID	**UC 8**
Name	Normal Difficulty
Description	Player is playing game on normal
Actor	Player
Pre-condition	Player has chosen the difficulty
Post-condition	Map ready to start
Main flow	1. Player selected difficulty 2. Map ready to start
Includes	UC 10 (Start Game)
Extensions	None

ID	**UC 9**
Name	Hard Difficulty
Description	Player is playing game on hard
Actor	Player
Pre-condition	Player has chosen the difficulty
Post-condition	Map ready to start

Main flow	1. Player selected difficulty 2. Map ready to start
Includes	UC 10 (Start Game)
Extensions	None

ID	**<u>UC 10</u>**
Name	Start Game
Description	After player settled desired options the game will finally start
Actor	Player
Pre-condition	Player decided map and difficulty
Post-condition	Map started
Main flow	1. Player selected options 2. Map started
Includes	UC 12, UC 13 (Lose Game, Win Game)
Extensions	UC 11 (Pause)

ID	**<u>UC 11</u>**
Name	Pause
Description	When the player is playing the game and want a break or want to exit or save the game
Actor	Player
Pre-condition	Player is currently playing the game
Post-condition	Game paused
Main flow	1. Player decides to take a break 2. Game is paused
Includes	None
Extensions	UC3, UC 14, UC 15, UC 16 (Help/How to play, Exit to menu, Continue, Save game)

ID	**<u>UC 12</u>**
Name	Lose Game
Description	Player lose the game after reaching a limit

Actor	Player
Pre-condition	Player is playing the game
Post-condition	Player is not able to defend a certain amount of enemy
Main flow	1. Player is playing 2. Player cannot defend a certain amount of enemy 3. Player loses the game
Includes	None
Extensions	UC 17, UC 14 (Play Again?, Exit to menu)

ID	**<u>UC 13</u>**
Name	Win Game
Description	Player win the game after defending a certain number of rounds
Actor	Player
Pre-condition	Player is playing the game
Post-condition	After several rounds if player health is more than zero
Main flow	1. Player is playing 2. Player kills a needed amount of enemies 3. Player win the game
Includes	None
Extensions	UC 14, UC 18 (Exit to menu, Next Map)

ID	**<u>UC 14</u>**
Name	Exit to menu
Description	A option for the player to exit from the current map and back to the menu, game datas will be lost if player decides or forgets to save, it will also display in situation where the player lose the game of wins it
Actor	Player
Pre-condition	Player is currently playing in a map
Post-condition	Player wants to go back the menu
Main flow	1. Player currently in a map 2. Player pause and want to save the game or the player won or lost the game 3. Exit to main menu option taken

Includes	None
Extensions	None

ID	**UC 15**
Name	Continue
Description	Continue is an option when the game is paused by the player
Actor	Player
Pre-condition	The game is paused
Post-condition	Player decides to continue
Main flow	1. Game paused 2. Game unpaused
Includes	None
Extensions	None

ID	**UC 16**
Name	Save Game
Description	Save game is an option displayed when the game is paused, and it is a very important feature to preserve the player's progress
Actor	Player
Pre-condition	Game paused
Post-condition	Player decides to save his progress
Main flow	1. Game paused 2. Progress saved
Includes	None
Extensions	UC 19 (selected saved game)

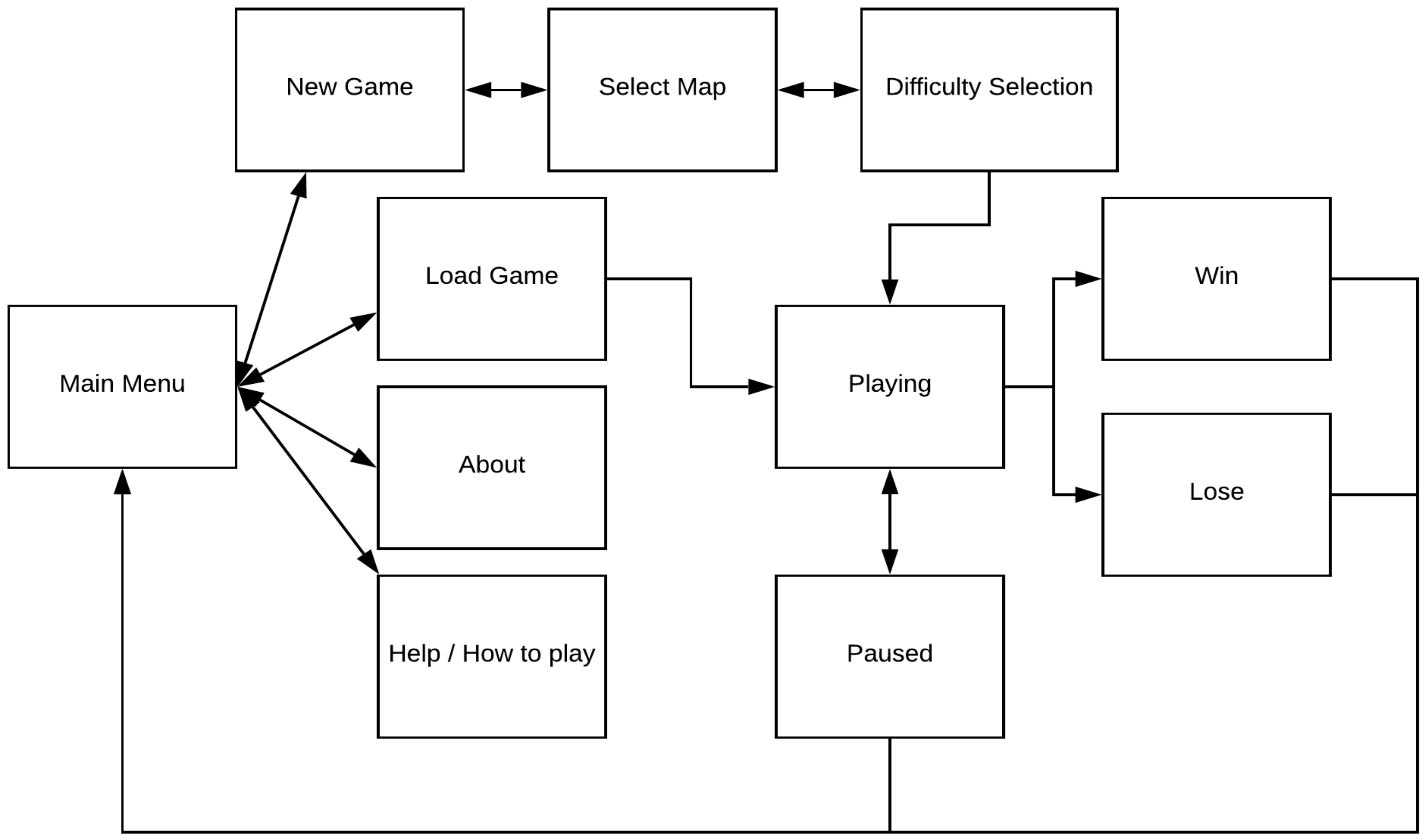
ID	**UC 17**
Name	Play Again?
Description	Option displayed when the player lose the game
Actor	Player
Pre-condition	Player lost the game

Post-condition	Player decides to play or no to play again
Main flow	1. Player lose the game 2. System displays play again message 3. Player play again
Includes	None
Extensions	None

ID	UC 18
Name	Next Map
Description	Option displayed when the player wins the game
Actor	Player
Pre-condition	Player wins in current map
Post-condition	Player decides to play the next map
Main flow	1. Player wins in current map 2. Next map option displayed 3. Player enters in the next map
Includes	None
Extensions	None

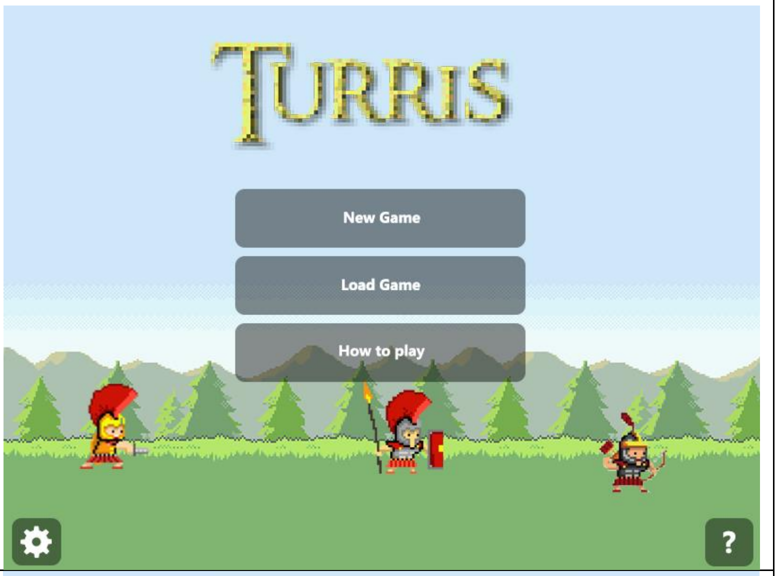
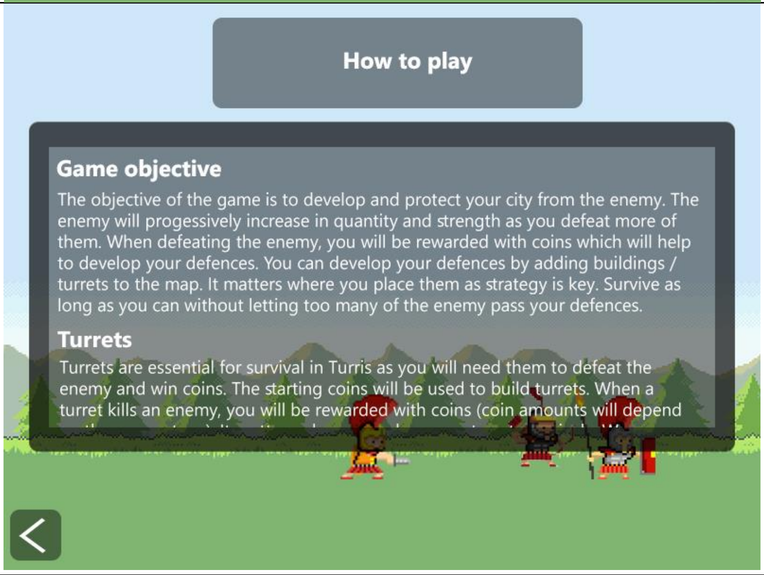
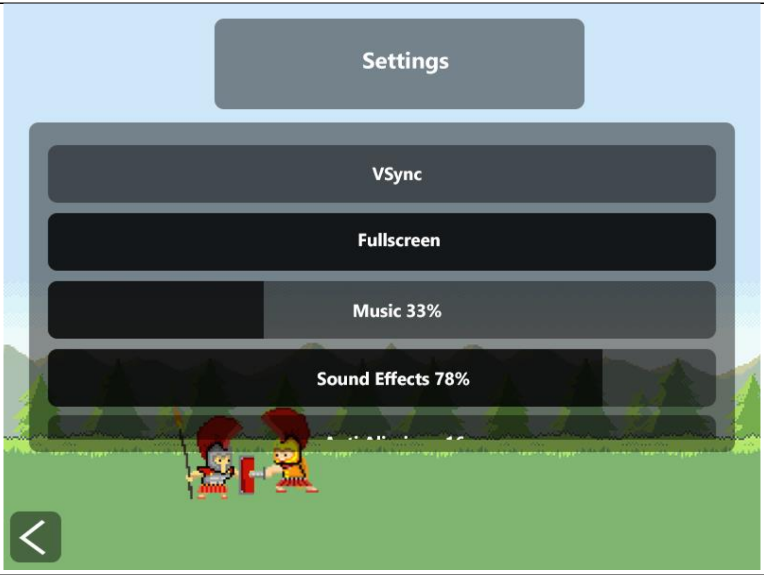
ID	**UC 19**
Name	Selected saved game
Description	Player in the menu can choose a saved progress through the load game option
Actor	Player
Pre-condition	Player has saved at least one progress previously
Post-condition	Map ready to start
Main flow	1. Player select saved game 2. Player loads in saved game 3. Map starts
Includes	None
Extensions	None

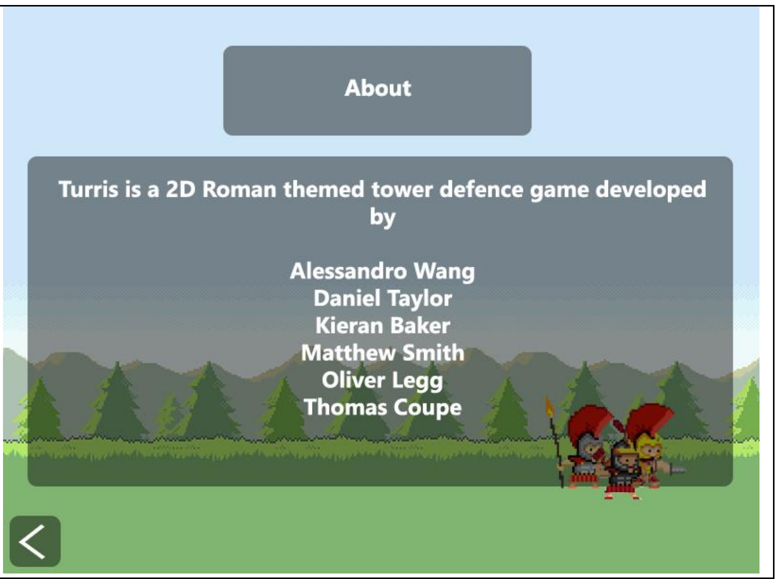
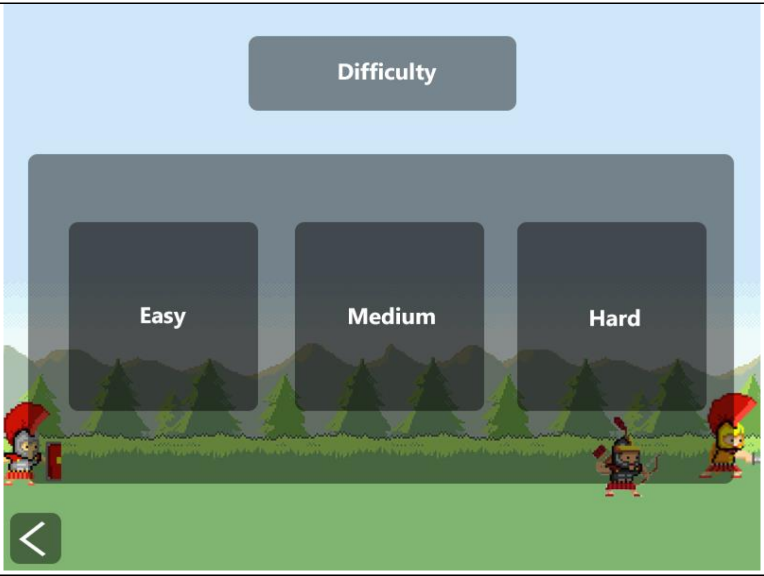
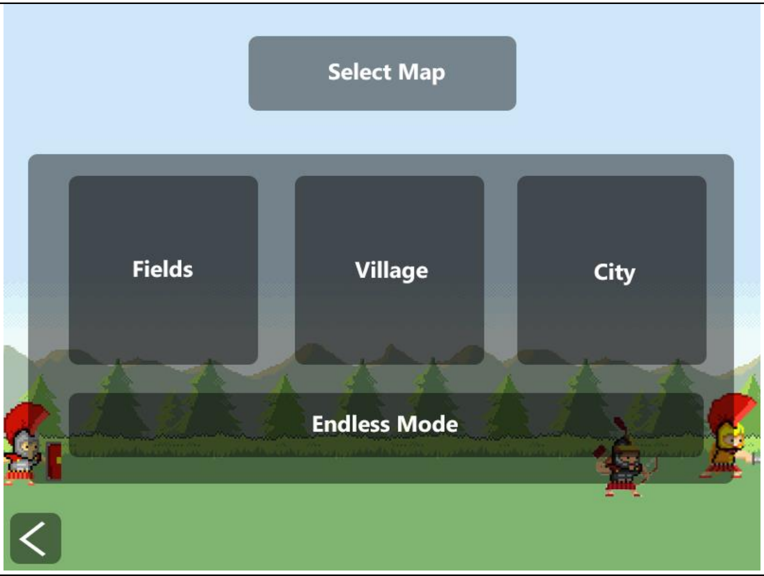
Process and states of the game

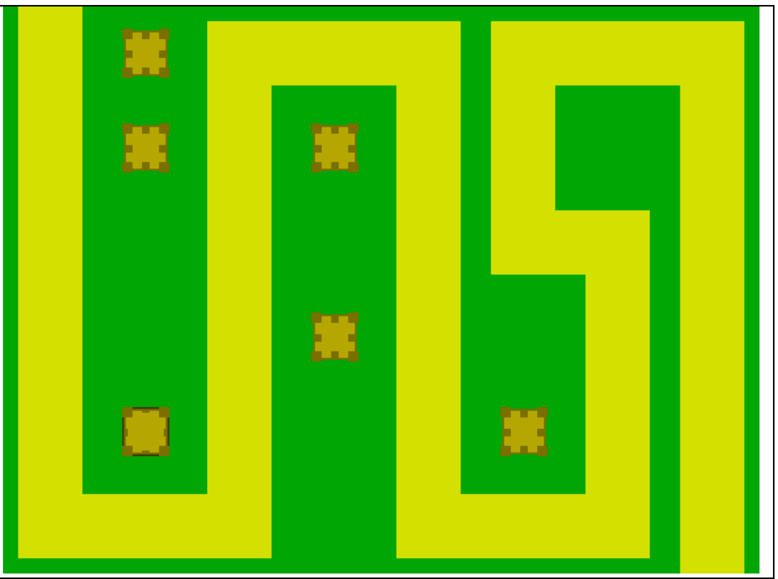
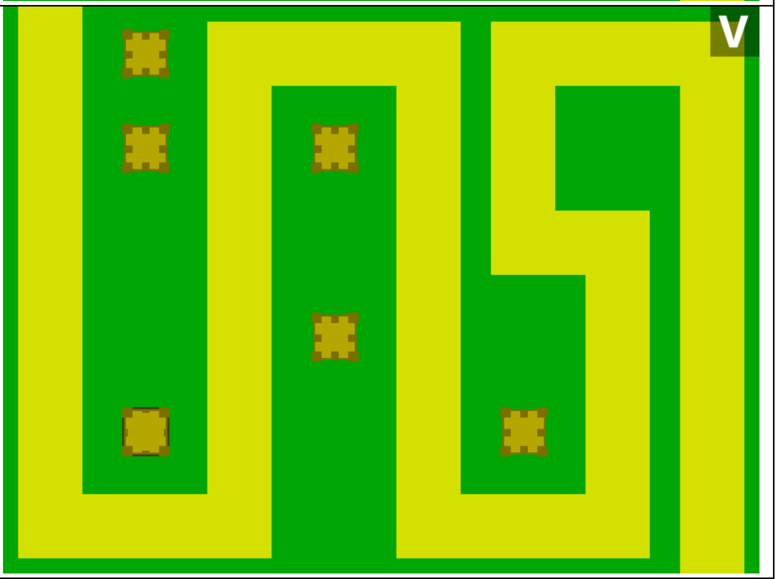
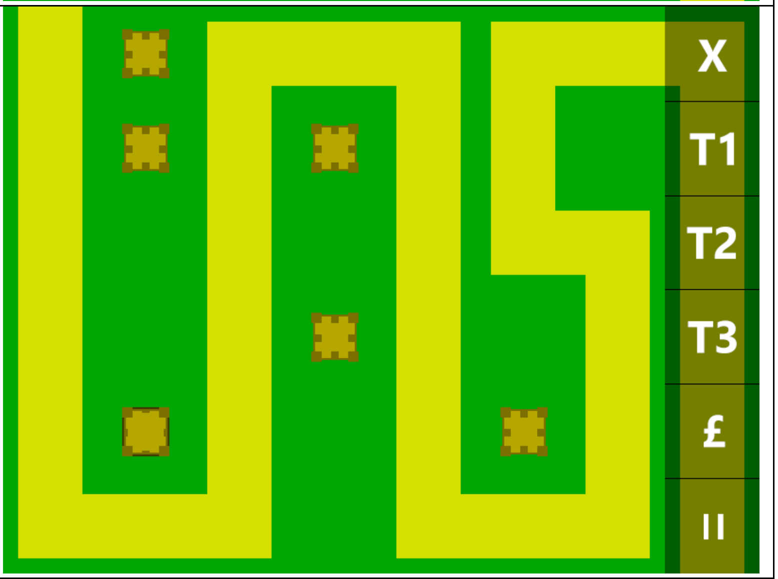


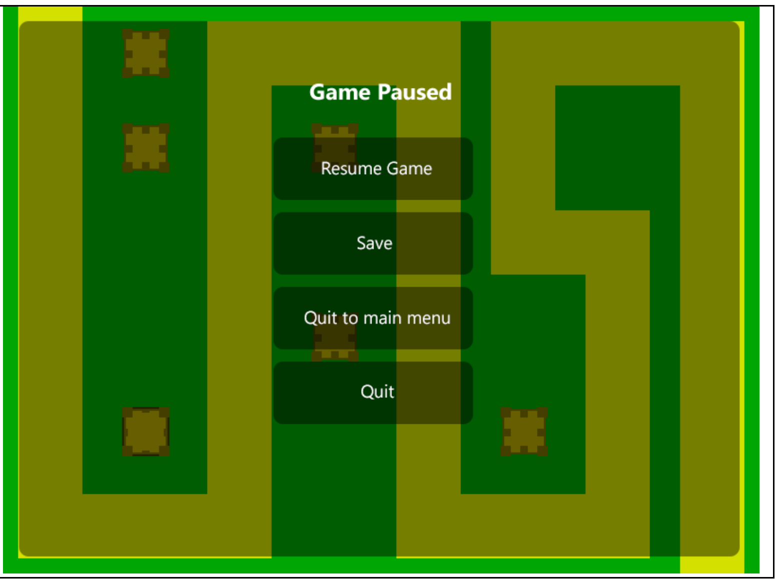
The above diagram shows the states in which the games will be in. You can see that when you access the main menu, you can create a new game, load the game, go to the about screen and final the help screen. Once you are on all these states, you can go back to the main menu. However, if you proceed onto another state, for example, the playing state, you are not able to navigate directly back to the main menu without pausing the game, winning the game or losing the game.

User interface

This is the main menu state. When you start the game. This is the first thing you'll see. As you can see you will have access to: • New Games • Loading old games • Document on how to play the game • About screen • Settings menu	
The how to play menu will go into detail about not only how to play the game, but it will also show you how to be much better. You can revert to the main menu by pressing the bottom left button.	How to play Game objective The objective of the game is to develop and protect your city from the enemy. The enemy will progessively increase in quantity and strength as you defeat more of them. When defeating the enemy, you will be rewarded with coins which will help to develop your defences. You can develop your defences by adding buildings / turrets to the map. It matters where you place them as strategy is key. Survive as long as you can without letting too many of the enemy pass your defences. Turrets Turrets are essential for survival in Turris as you will need them to defeat the enemy and win coins. The starting coins will be used to build turrets. When a turret kills an enemy, you will be rewarded with coins (coin amounts will depend
The settings menu will have technical settings and options that you might want to select. A darker coloured button will mean that setting is active, whereas a light button will mean that setting is off or inactive. For example, in the picture you will see that fullscreen is on and vsync is off. The music is set to 33% as you can see from the slider and the sfx is set to 78%. This is clarified by the label on the button too.	Settings VSync Fullscreen Music 33% Sound Effects 78%

This is the about menu which shows what the game is about and who developed the game.	 Turris is a 2D Roman themed tower defence game developed by Alessandro Wang Daniel Taylor Kieran Baker Matthew Smith Oliver Legg Thomas Coupe
The player selects the difficulty of the game here. They're also able to revert to the main menu.	
The selection of the map is here, they have 3 choices (and might be given more in the future), there is also an endless mode which selects a special map.	

This is the playing state of the game. This shows the path that the enemies will go down in on the field. The enemies will start from the top left and finish at the bottom left. Anywhere where the grass is on the grid, you will be able to place turrets. As you can see, turrets are on the field.	
This is the playing state of the game again, but this is taken with the GUI minimised. When you press on the button, it will create the menu shown on the next page.	
This is the GUI maximised. The button functionality works like so: • X (closes the GUI) • T1 (places turret tier 1) • T2 (places turret tier 2) • T3 (places turret tier 3) • £ (sells) turrets • Pause (pauses the game)	

The game is paused here, and you can choose to resume the game, save the game and quit to the main menu. Movements of the enemies and bullets will stop as well as the shooting of turrets. Time will also not increase in endless mode when paused. The pause state can be initiated by pressing the pause button on the previous state, or by pressing the ‘Esc’ key.	 Game Paused Resume Game Save Quit to main menu Quit

Functional Description / Specification

Introduction

A Functional Description allows us to explain all the details of what we are creating to someone who has no idea or prior information regarding the project. This means that we must be very accurate when writing the specific details for our game. I have considered the feedback that we received from Sebastian Coope which was to go into a lot more detail when we are talking about our game, this makes sense as we want to be able to explain to the outside audience as clearly as possible.

Gameplay

Turris is the name of the game we have come up with, we have decided that it is going to be a tower defense game with turrets / towers, different ammo types, multiple enemy types, multiple maps and 3 difficulties.

The game will include waves of enemies that approach from one side of the map and attempt to make it to the other, if a certain number of enemies reach the other side then they will defeat the player. To stop these waves of enemies, the player can place turrets that are going to be firing / attacking the enemies to stop them from making it to the end. Depending on the difficulty and or map, this can be made harder for the player.

Turrets / Towers

We do not want to go over the top with the details for the game, in fact we want to be able to make it relatively simple for us to create and then make changes as we go along to improve the game. Therefore, we are setting the default number of different towers to 3, this will be useful to us when we are first trying to understand how our game is going to work.

The towers will each have a different cost, the more powerful the tower the higher the cost. It is simple to understand, however saving money to survive the higher rounds is going to be a player-based decision that could allow them to pass certain waves, it is a game of skill after all.

The towers will have a certain range, we will use grid references within a list, to calculate the distance from the towers and the enemies, again the higher cost of the tower the larger range it will have, this will give the player a large advantage as they proceed through the waves.

There are going to be multiple enemies, so placing multiple towers around the map will be a common strategy for the player due to the fact they will have to defeat many enemies at different areas of the map, obviously a single tower will not be able to fire across the whole map, so it is down to the player to decide where these towers are placed (this will again be done with grid references so that we can snap the towers into certain places, we might have harder levels where placements in some parts of the map are out of bounds).

Different Ammo Types

Within the game there are also going to be different ammo types, this means that the towers can have different types of ammo that the player can either purchase or select if they have already purchased some. Obviously, this is going to be a strategic move by the player, as they are going to be the ones to decide whether they will need to purchase the ammo from the shop within the next wave of enemies. They will be able to select the different ammo types for each individual tower; they will be limited to how many times they can fire that specific ammo by how much of it they purchased. There are going to be 3 different ammo types, these will have different properties such as increased damage or can hit multiple targets.

Enemies

We have to have different enemies within our game to make it a challenge and more interesting for the player, I have also stated that we do not want to go over the top and make our game too complicated as we are first building it, so again we are going to have 3 main enemies that are going to spawn in the game. Usually, the game will start off as easy on any difficulty, with the easiest enemies being the in the first few waves, however as the game progresses, we are going to introduce the other enemies that are going to be a concern to the player. The enemies will have different amount of health points that require you to hit over time to defeat them, the further the player gets in the game, the more enemies are spawned, as well as more of the higher tier / harder enemies will spawn and give the player a challenge. These enemies might have different speeds as well, meaning they will travel through the map faster, putting pressure on the player for them to be quick and defeat the enemies before they reach the end goal.

Maps

We have discussed the maps in detail, as we believe they are going to be the foundation of the game and how we are going to control all the tower placements, enemy pathing and other key factors that are going to create the game. We are designing the maps very basic at first to get an understanding of how we are going to move the enemies in terms of flags and grid references, however we are going to have 3 maps that will all get harder when the player completes one and moves to the next. I mentioned previously that we might have areas where a player can't place a turret, such as a river or a lake in which there is no way a player can place a turret as it is meant to be positioned on land. (We might include towers that can only be placed on water in the future to give the game a little more detail)

The maps are going to be based around a certain path, the design is up to us on how the path changes throughout the map, as there are going to be twists and turns that the enemies follow. The enemies will always follow that specific path so that the user knows where he or she can or can't place the turrets to fight the enemies. The path will change on each map and might be longer or shorter which will add to the difficulty of the game.

Difficulties

The difficulties are self-explanatory, we are going to have Easy, Normal and Hard. In terms of how that changes the game, the difficulties will increase the health of the enemies and possibly the speed. The main thing that will change is the health of the player, as the player loses once a certain number of enemies pass through the end goal, so the amount it takes for the player to lose will lower the harder the difficulty is. Therefore, to explain a little better, Easy mode will have 150 health, Normal will have 100 and Hard will have 50, this makes the game more intense for the player and allows us to increase the difficulty of the game automatically without making and drastic changes. We can also increase the number of enemies that spawn within the levels when we change the difficulty so that it is not always the same. Hard mode might even have a bonus final round in which you have to defeat an overall boss to win the game.

We believe that adding multiple difficulties gives the player a sense of accomplishment once they have completed a map, they will then feel like they are progressing through the game as they go through the levels and increase the difficulty if they choose.

Shop

The shop is a very important part of the game, it is the main area of player interaction outside of the map, we have to make sure that the shop is integrated perfectly into the game, so that there are no issues when the player is attempting to spend his points / currency to get a different tower or ammunition.

The shop must be very easy to access as the player will need to purchase towers as the waves of enemies are progressing, the shop will have towers and ammunition to purchase, we might update this later so that we have upgrades although that will be once the main aspects of the game are implemented.

Currency

The currency is closely related to the shop as it is going to be how we purchase towers and ammunition, the currency will be gathered by defeating enemies, a certain amount of currency will be given to the player once an enemy has been defeated. The harder the enemy is to defeat the more currency the player will receive once it has been defeated. The player should be able to spend their currency in the shop as the game / waves are progressing, meaning we must make sure that the shop is always accessible for the player so that they can spend their earned currency.

Pseudocode

Turret

```
ABSTRACT CLASS Turret(x, y)

    FILE texture
    FLOAT rateOfFire
    INTEGER range
    INTEGER x
    INTEGER y
    INTEGER level
    INTEGER MAX_UPGRADE = 3
    INTEGER damage

    FUNCTION ABSTRACT upgrade()

    FUNCTION getX()
        RETURN x
    END

    FUNCTION getY()
        RETURN y
    END

    FUNCTION fire_at_enemy(enemy)
        AIM_AT(enemy)
        FIRE()
        WAIT(rateOfFire)
    END

END
CLASS Turret_I EXTENDS Turret

    INTEGER range = 500
    INTEGER level = 1
    INTEGER damage = 10
    FLOAT rateOfFire = 0.5

    FUNCTION create(x, y)
        PUSH_TO_ABSTRACT_CLASS(x, y, "Turret_I.png")
    END

    FUNCTION getX()
        RETURN x
    END

    FUNCTION getY()
        RETURN y
    END

    FUNCTION fire_at_enemy(enemy)
        AIM_AT(enemy)
        FIRE()
        WAIT(rateOfFire)
    END

    FUNCTION upgrade()
        IF level < MAX_LEVEL THEN
```

```
            rateOfFire = rateOfFire - 0.2
            range = range + 100
            damage = damage + 2
            level = level + 1
        END
    END
END
CLASS Turret_II EXTENDS Turret

    INTEGER range = 600
    INTEGER level = 1
    INTEGER damage = 20
    FLOAT rateOfFire = 0.5

    FUNCTION create(x, y)
        PUSH_TO_ABSTRACT_CLASS(x, y, "Turret_II.png")
    END

    FUNCTION getX()
        RETURN x
    END

    FUNCTION getY()
        RETURN y
    END

    FUNCTION fire_at_enemy(enemy)
        AIM_AT(enemy)
        FIRE()
        WAIT(rateOfFire)
    END

    FUNCTION upgrade()
        IF level < MAX_LEVEL THEN
            rateOfFire = rateOfFire - 0.2
            range = range + 120
            damage = damage + 3
            level = level + 1
        END
    END
END
CLASS Turret_III EXTENDS Turret

    INTEGER range = 300
    INTEGER level = 1
    INTEGER damage = 70
    FLOAT rateOfFire = 2.5

    FUNCTION create(x, y)
        PUSH_TO_ABSTRACT_CLASS(x, y, "Turret_III.png")
    END

    FUNCTION getX()
        RETURN x
    END

    FUNCTION getY()
        RETURN y
    END

    FUNCTION fire_at_enemy(enemy)
        AIM_AT(enemy)
        FIRE()
        WAIT(rateOfFire)
    END

    FUNCTION upgrade()
        IF level < MAX_LEVEL THEN
            rateOfFire = rateOfFire - 0.2
            range = range + 50
            damage = damage + 7
            level = level + 1
        END
    END
END
```

Enemy

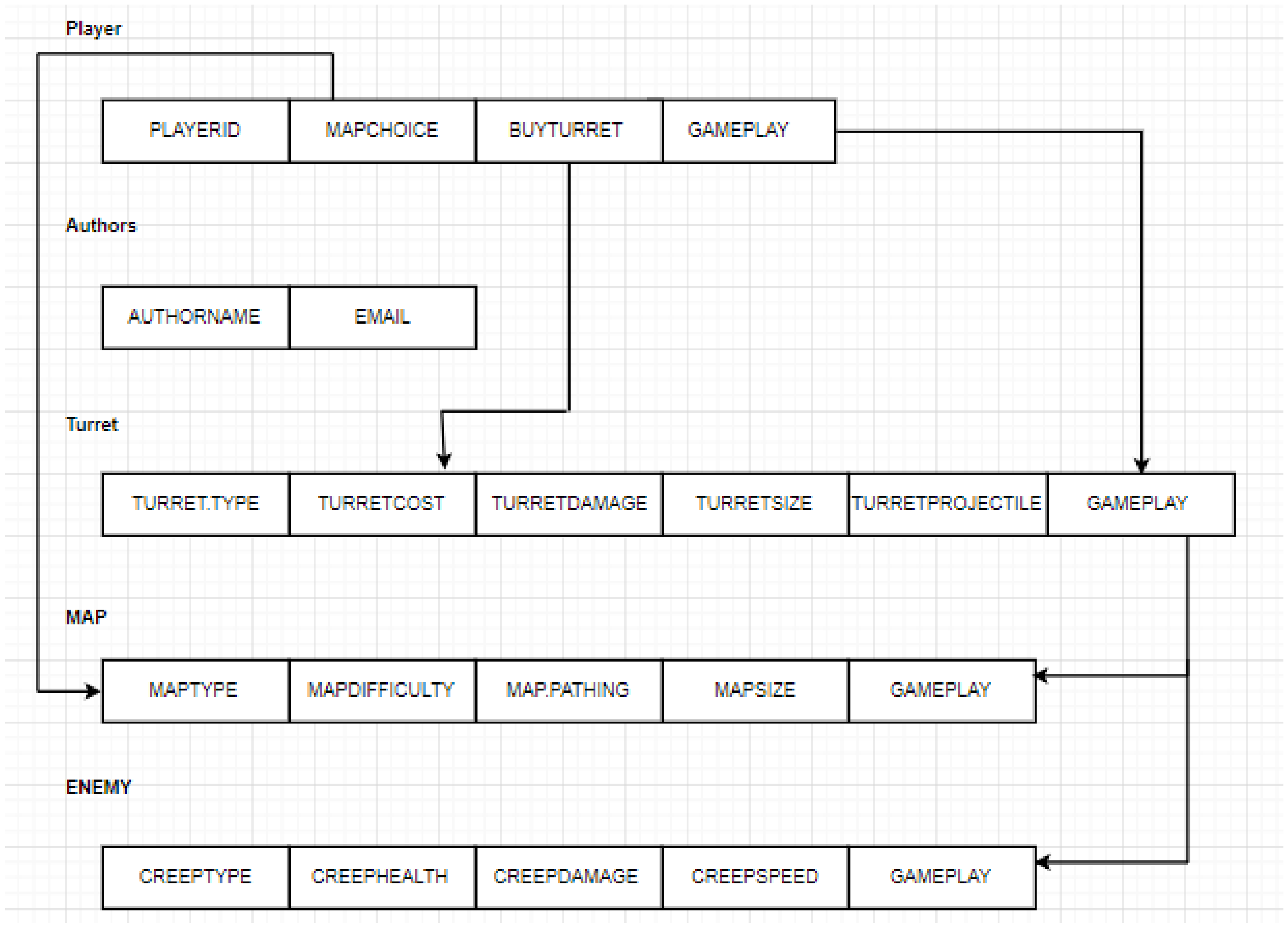
```
ABSTRACT CLASS enemy_1(INTEGER x, INTEGER y)

    FILE texture
    FLOAT rateOfFire
    INTEGER range
    INTEGER x
    INTEGER y
    INTEGER speed
    INTEGER damage

    FUNCTION ABSTRACT upgrade()

    FUNCTION getX()
        RETURN x
    END

    FUNCTION getY()
        RETURN y
    END

    FUNCTION move_to(INTEGER pos_x, INTEGER pos_y)
        IF pos_x != x AND x THEN
             MOVE FORWARD
       ELSE
            RETURN FALSE
        END
    END

END

CLASS enemy_2(INTEGER x, INTEGER y) EXTENDS enemy_1

    INTEGER range = 25
    INTEGER x = 0.1
    INTEGER y = 0.4
    INTEGER speed = 0.35
    INTEGER damage = 25

    FUNCTION ABSTRACT upgrade()

    FUNCTION getX()
        RETURN x
    END

    FUNCTION getY()
        RETURN y
    END

    FUNCTION move_to(INTEGER pos_x, INTEGER pos_y)
        IF pos_x != x AND x THEN
             +pos_x
       ELSE
            +pos_y
        END
    FUNCTION takeDamage()
       IF arrowhit
```

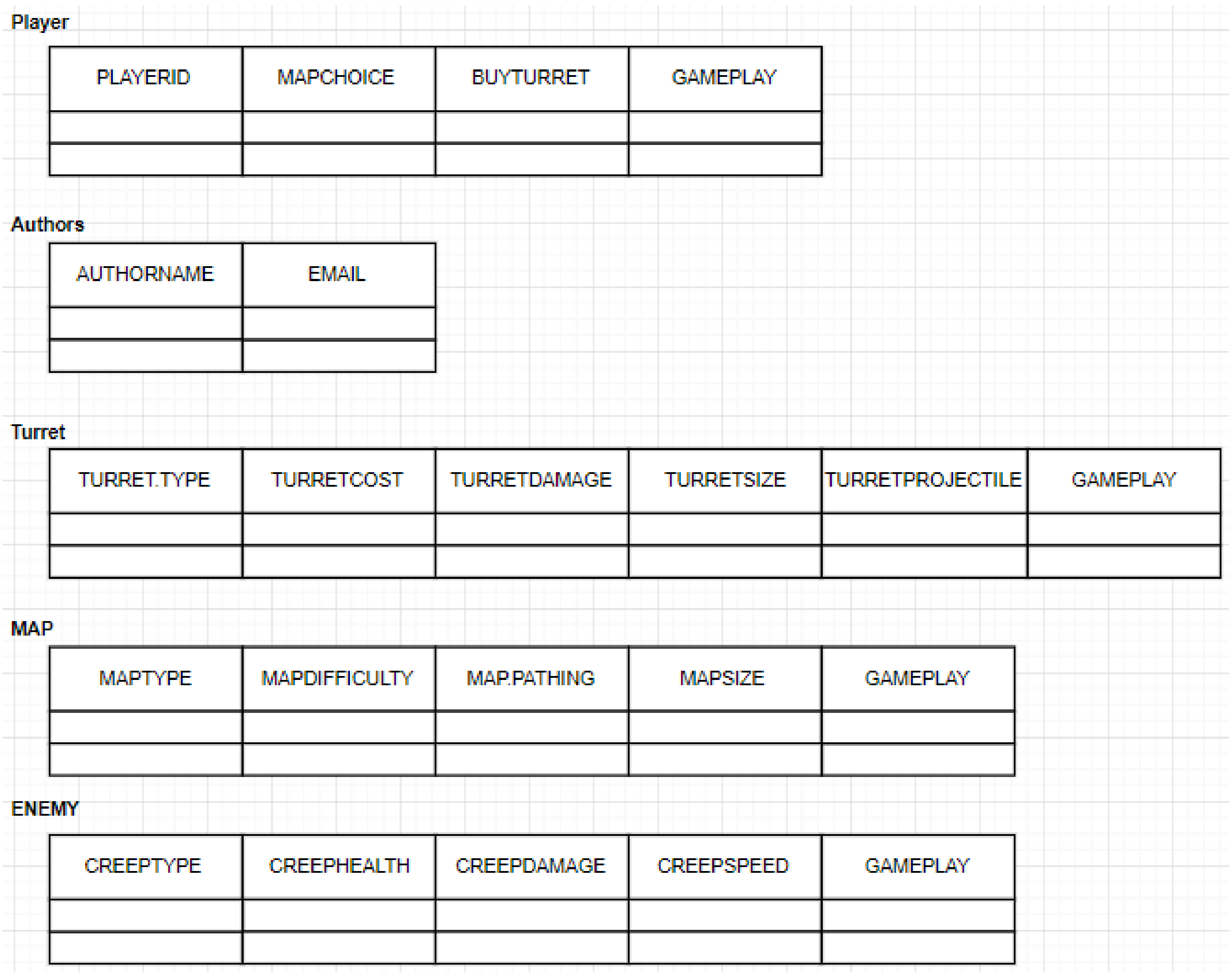
```
          health - 100
        ELSE
          health = health
      END
    END

END

CLASS enemy_3(INTEGER x, INTEGER y) EXTENDS enemy_1

    INTEGER range = 35
    INTEGER x = 0.1
    INTEGER y = 0.4
    INTEGER speed = 0.50
    INTEGER damage = 35

    FUNCTION ABSTRACT upgrade()

    FUNCTION getX()
        RETURN x
    END

    FUNCTION getY()
        RETURN y
    END

    FUNCTION move_to(INTEGER pos_x, INTEGER pos_y)
        IF pos_x != x AND x THEN
             +pos_x
      ELSE
            +pos_y
        END
    FUNCTION takeDamage()
      IF arrowhit
          health - 100
        ELSE
          health = health
      END
    END

END
```

Player

```
CLASS player_shop

      INTEGER itemPrice
      STRING itemName
      INTEGER stock
      INTEGER playerCurrency

      FUNCTION purchaseItem()
             IF itemPurchased
                    playerCurrency - itemPrice
                    stock - 1
                    player +itemName
             ELSE
                    playerCurrency = playerCurrency
      END

      FUNCTION sellItem()
            IF itemSold
                  playerCurrency + itemPrice / 3
            END
      END

Class player

      INTEGER playerCurrency
      INTEGER playerHealth
      STRING playerName
      INTEGER playerLevel
      INTEGER ammoCount
      STRING ammoType
      INTEGER towerAvailable

      FUNCTION enemyKilled()
            IF enemy_health = 0
                  enemyKilled +1
                  playerCurrency +50
            END

      FUNCTION enemyCrossed()
            IF enemyAtEnd
                  playerHealth - 1
            END
      END
```

Table Structures





Player

PLAYERID	MAPCHOICE	BUYTURRET	GAMEPLAY

Authors

AUTHORNAME	EMAIL

Turret

TURRET.TYPE	TURRETCOST	TURRETDAMAGE	TURRETSIZE	TURRETPROJECTILE	GAMEPLAY

MAP

MAPTYPE	MAPDIFFICULTY	MAP.PATHING	MAPSIZE	GAMEPLAY

ENEMY

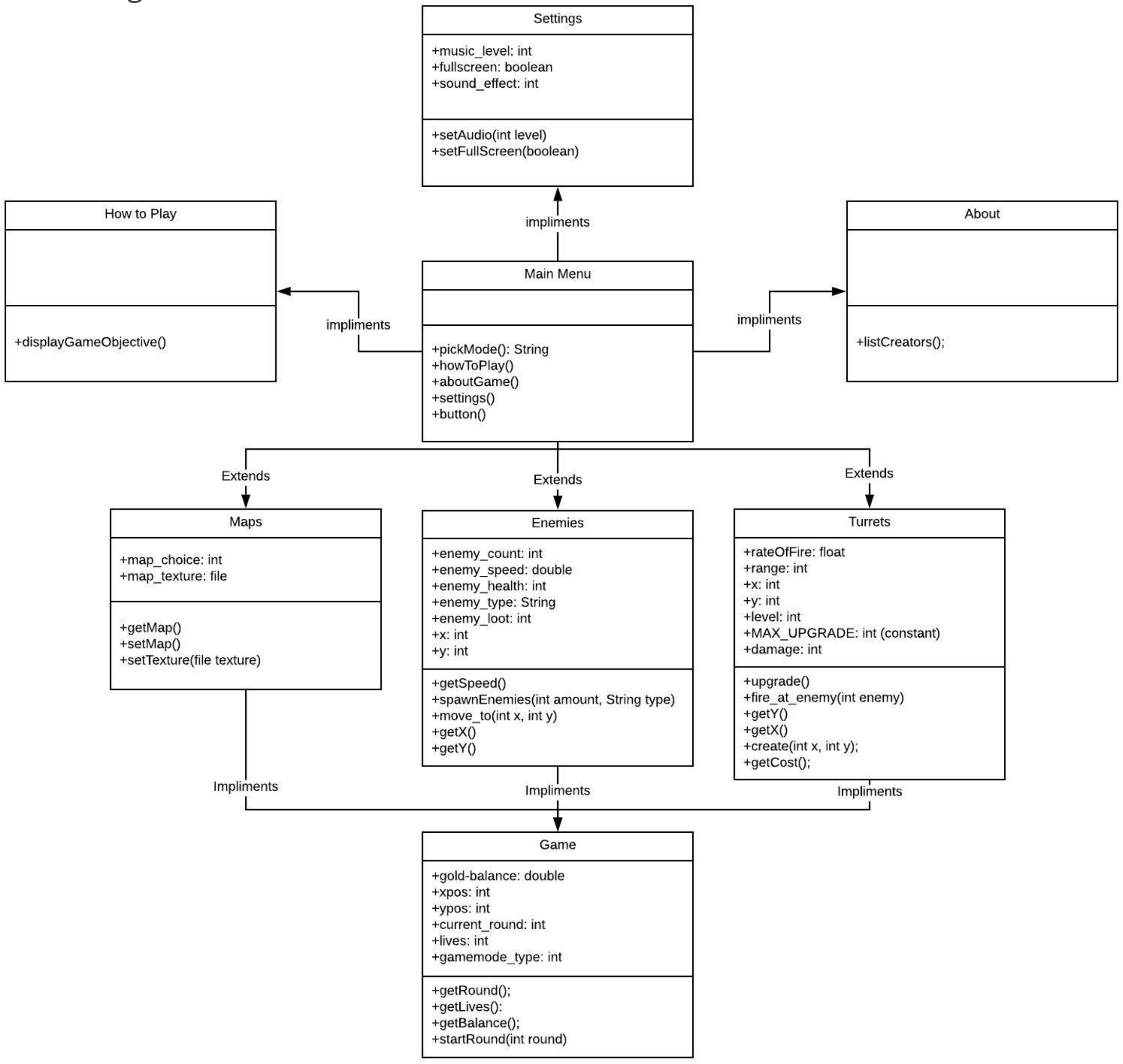
CREEPTYPE	CREEPHEALTH	CREEPDAMAGE	CREEPSPEED	GAMEPLAY

Data Dictionary

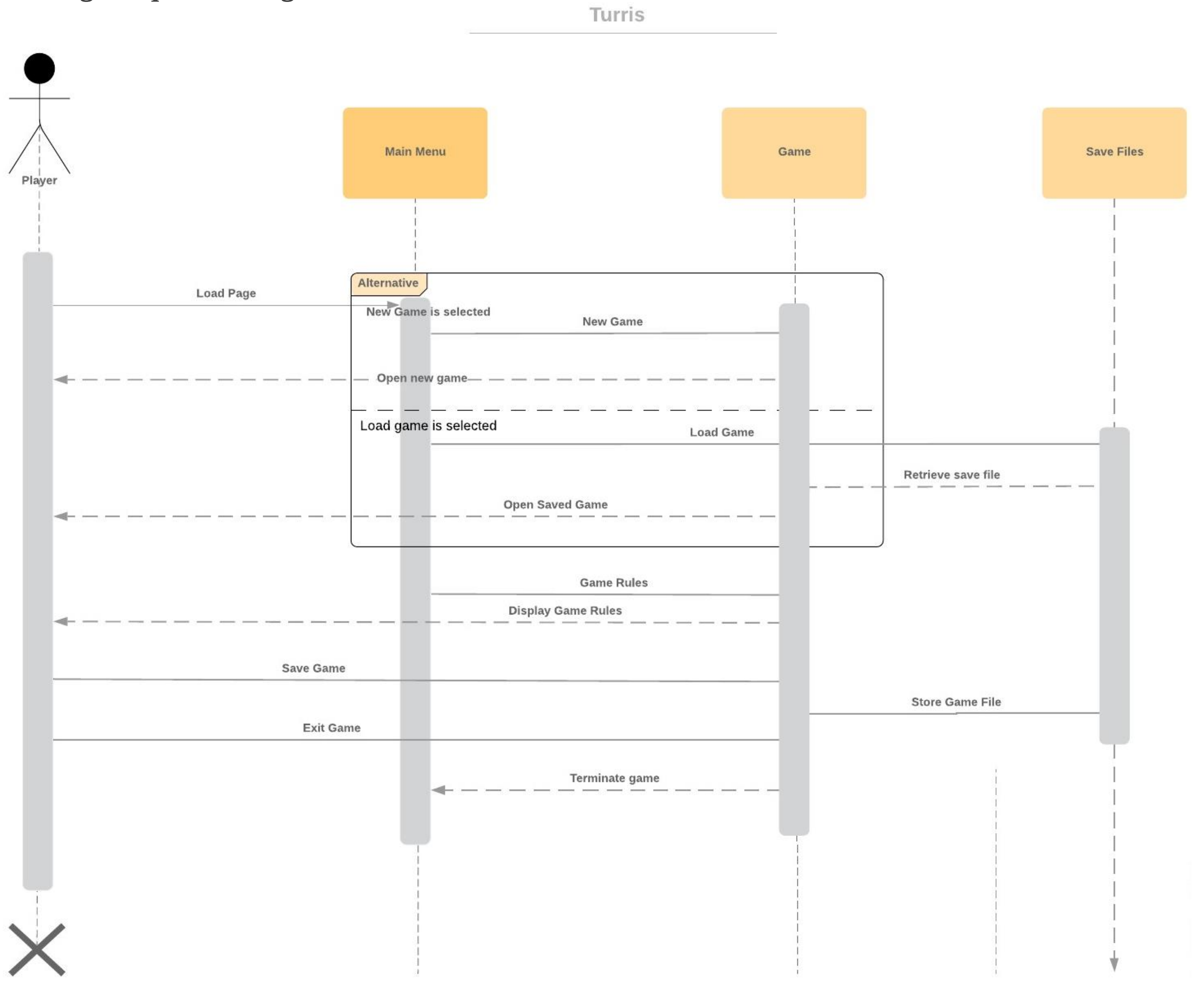
Data Dictionary - Turris				
Name	Data Type	Length	Scope	Purpose
player_lives	Integer	1 - 20	public	This value holds the remaining number of lives the player has. Once this value is at 0 the game will be over.
gold_balance	double	0.00 - 20000.00	public	When the player kills an enemy, they receive a gold reward to buy and upgrade towers. This value represents the total amount of gold the player has.
xPosition	Integer	Width of frame	public	xPosition represents the location of where the turret is along the width of the frame.
yPosition	Integer	Height of frame	public	yPosition represents the location of where the turret is along height of the frame.
Turret_x	Integer	0-800	public	This value represents where the player is location along the x axis giving the x position of the player on the frame.
Turret_y	Integer	0-600	public	This value represents where the player is location along the y axis giving the position of the player on the frame.
Current_round	Integer	1 – 30 or if on endless mode its unlimited	public	This value represents the current round that the player is on, as the round value increases so does the enemy count.
Enemy_count	Integer	unlimited	public	The enemy count represents how many enemies are to be spawned in on that round. As there is an endless mode there is no fixed maximum number of enemies that can be spawned.
enemy_speed	Double	0.01 – 5.00	public	This value represents the speed that the enemies move along the path.
Turret1_price	Double	200.00	Public	This value represents the total cost for Turret 1
Turret2_price	Double	500.00	Public	This value represents the total cost for Turret 2
Turret3_price	double	1000.00	public	This value represents the total cost for Turret 3
Turret1_level	Integer	1 – 5	Public	This value represents the current tower level for tower 1
Turret2_level	Integer	1 – 5	Public	This value represents the current tower level for tower 2
Turret3_level	Integer	1 – 5	public	This value represents the current tower level for tower 3
turret_damage	Integer	0 - 100	public	The integer value represents the amount of damage a tower does to an enemy per each shot from the tower.
map_choice	Integer	1 - 3	public	The map selected is represented as an integer value. As there is three maps the user can choose between 1, 2 and 3 to select which map they want to play.
enemy_health	Integer	0 - 100	public	This value represents how much help each of the enemies have.
enemy_type	String	"Gladiator", "Warrior", "Fighter"	public	The enemy type can be chosen between three types of enemies. These enemies are defined by Strings.

turret_range	Integer	Amount of blocks the turret can reach	public	This value represents how many blocks the tower projectiles can reach from their position.
Enemy_loot	Integer	The amount of gold an enemy drops when killed	public	This value represents how much gold the enemy will when it is
gamemode_type	Integer	1 - 2	public	This value represents which game mode has been selected.

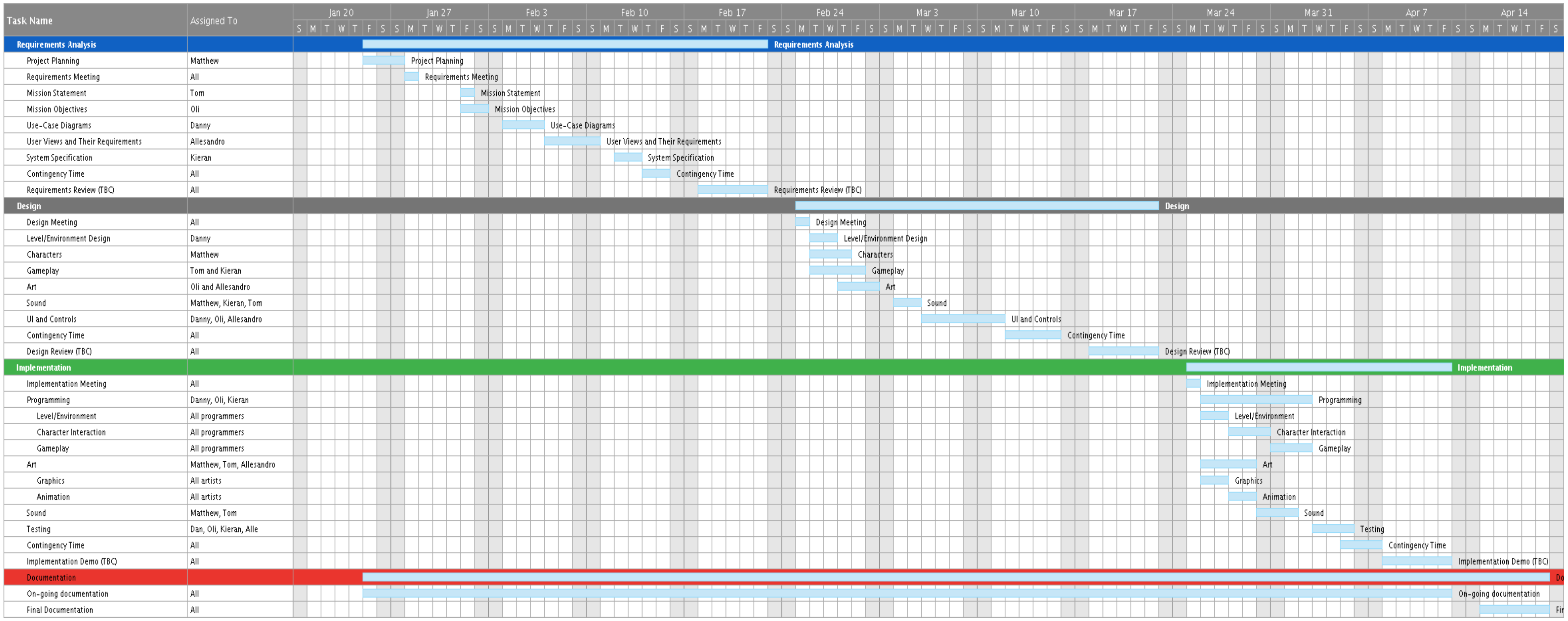
Class Diagram



Message Sequence Diagram



Original Gantt Chart

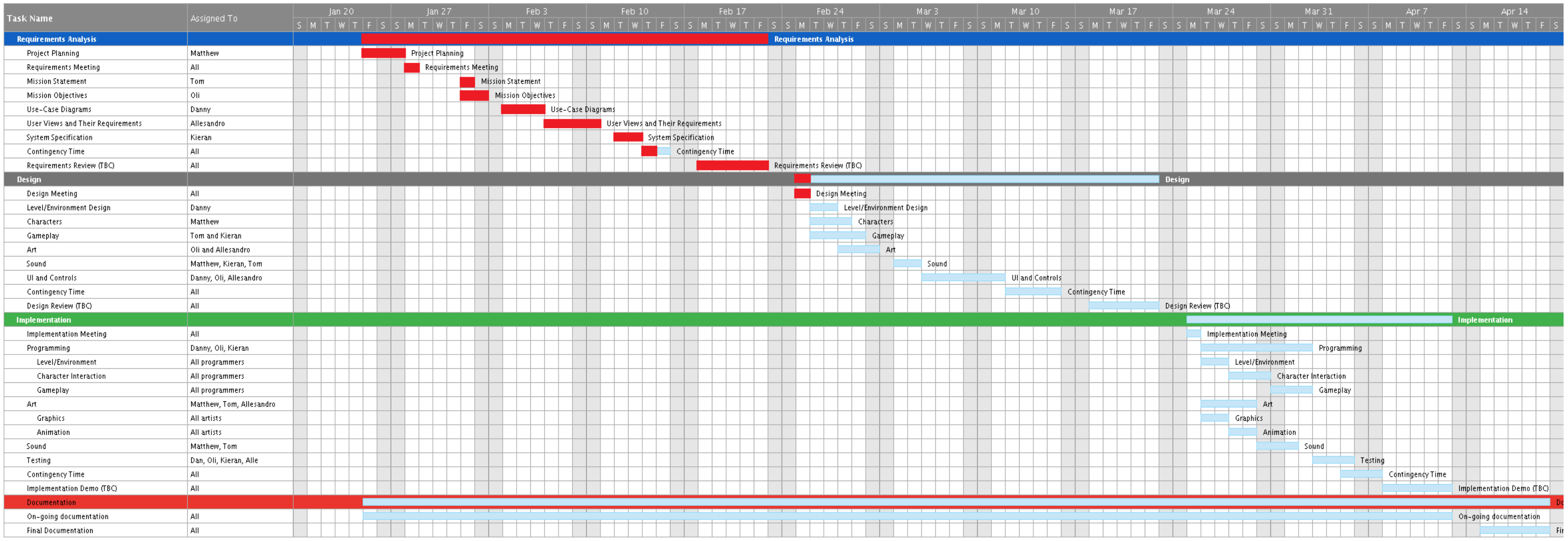


	Task Name	Assigned To
1	**Requirements Analysis**	
2	Project Planning	Matthew
3	Requirements Meeting	All
4	Mission Statement	Tom
5	Mission Objectives	Oli
6	Use-Case Diagrams	Danny
7	User Views and Their Requirements	Allesandro
8	System Specification	Kieran
9	Contingency Time	All
10	Requirements Review (TBC)	All
11	**Design**	
12	Design Meeting	All
13	Level/Environment Design	Danny
14	Characters	Matthew
15	Gameplay	Tom and Kieran
16	Art	Oli and Allesandro
17	Sound	Matthew, Kieran, Tom
18	UI and Controls	Danny, Oli, Allesandro
19	Contingency Time	All
20	Design Review (TBC)	All
21	**Implementation**	
22	Implementation Meeting	All
23	Programming	Danny, Oli, Kieran
24	Level/Environment	All programmers
25	Character Interaction	All programmers
26	Gameplay	All programmers
27	Art	Matthew, Tom, Allesandro
28	Graphics	All artists
29	Animation	All artists
30	Sound	Matthew, Tom
31	Testing	Dan, Oli, Kieran, Alle
32	Contingency Time	All
33	Implementation Demo (TBC)	All
34	Documentation	
35	On-going documentation	All
36	Final Documentation	All

Updated Gantt Chart

Up till now, we have completed every task within the time constraints set out by our original plan. This includes all tasks associated with the requirements analysis. We had our review of our requirements document and have been given valuable feedback. Currently we are on track to completing our design document. The review has already been decided to take place on 22nd March.

We have not made any major changes to the Gantt chart as when we had our design meeting, every task we discussed fulfilled the plan. We have indicated in red what tasks have already been completed. You may notice we did not use all our contingency time for requirements analysis. This is because we worked efficiently and didn't come across too many hurdle (this has been indicated on the Gantt Chart below).



Task Name	Assigned To
Requirements Analysis	
Project Planning	Matthew
Requirements Meeting	All
Mission Statement	Tom
Mission Objectives	Oli
Use-Case Diagrams	Danny
User Views and Their Requirements	Allesandro
System Specification	Kieran
Contingency Time	All
Requirements Review (TBC)	All
Design	
Design Meeting	All
Level/Environment Design	Danny
Characters	Matthew
Gameplay	Tom and Kieran
Art	Oli and Allesandro
Sound	Matthew, Kieran, Tom
UI and Controls	Danny, Oli, Allesandro
Contingency Time	All
Design Review (TBC)	All
Implementation	
Implementation Meeting	All
Programming	Danny, Oli, Kieran
Level/Environment	All programmers
Character Interaction	All programmers
Gameplay	All programmers
Art	Matthew, Tom, Allesandro
Graphics	All artists
Animation	All artists
Sound	Matthew, Tom
Testing	Dan, Oli, Kieran, Alle
Contingency Time	All
Implementation Demo (TBC)	All
Documentation	
On-going documentation	All
Final Documentation	All

Testing

Introduction

We have tested the game to see if the functionality of the game works. Something we checked a lot is to see if elements loaded correctly and how the user's interactions with the input got the correct output. For example, I tested the fullscreen button. When the user presses the button, the user should be put in fullscreen. If the user is in fullscreen mode, they should be put into windowed mode. The button should change graphically depending if the user's window is fullscreen too. Depending on a test's performance, it will be colour coded like so:

Test passed	🟩
Test failed – fixed	🟨
Test failed – unable to fix	🟥

Main Menu

Test No	Test Name	Description	Data Input	Expected Outcome	Actual Outcome	Remedial Action
1	Main menu appears	Main menu displaying all graphics	-	Main menu appears with the background displayed and clouds moving in the background correctly	As expected	-
2	All Buttons highlight	Testing if the buttons can be interacted with graphically	mouse_x mouse_y	Buttons can be highlighted when hovering over it with the mouse	As expected	-
3	New game button	Testing to see if the new game button clicks	mouse_x mouse_y mouse_leftb	Button is pressed and user is taken to the map selection state	As expected	-
4	Load game button	Testing to see if the load game button clicks	mouse_x mouse_y mouse_leftb	Button is pressed and user is taken to the loaded save	As expected	-
5	Settings button	Testing to see if the settings button clicks	mouse_x mouse_y mouse_leftb	Button is pressed and the user is taken to the settings menu	As expected	-
6	How to play button	Testing to see if the how to play button clicks	mouse_x mouse_y mouse_leftb	How to play button is pressed and the documentation is then found online on the Turris website	As expected	-
7	About button	Testing to see if the about button clicks	mouse_x mouse_y mouse_leftb	The about button is pressed and then the user is taken to the about state	As expected	-
8	Back button	Testing to see if the back button clicks. The same back button is used on the map selection, settings and	mouse_x mouse_y mouse_leftb	When you hover over the button, it should also change colour	As expected	-

		about state. If you press the button, it should take you back to the main menu state.				
9	Exit button	Testing to see if the exit button clicks	mouse_x mouse_y mouse_leftb	When you press the exit button, it should close the program.	As expected	-
10	Map selection screen	Testing to see if the map selection screen works	-	The map selection screen should display the main menu backgrounds, the 3 maps to choose from and the 'Select map' header at the top of the page	As expected	-
11	Map Selection Highlighting maps	Test to see if interacting with the maps tells you anything	mouse_x mouse_y	Hovering you mouse over the maps should change the 'Select map' header to the map name to either 'Grassy Greens', 'Sand Dunes' or 'Tricky Track'. There should also be a border around the map with appearing buttons over the map image that you can then click on.	As expected	-
12	Standard mode	Clicking on maps on standard mode	mouse_x mouse_y mouse_leftb	Clicking on maps on standard mode should take you to the map you specified on that standard mode.	As expected	-
13	Continuous mode	Clicking on maps on continuous mode	mouse_x mouse_y mouse_leftb	Clicking on maps on continuous should take you to the map you specified on that standard mode.	As expected	-
14	Settings screen	Settings screen should display all buttons and sliders	-	Clicking on the settings button should display all the buttons and sliders	As expected	-
15	Settings buttons	The buttons (apart from back) checked if interactable	mouse_x mouse_y	If the user hovers the mouse over the Fullscreen and Toggle mute button, the background of the mute button should dim to seem interactable	The buttons didn't dim when you hover the mouse over it.	Fixed the issue and uploaded to github. The colours now change to the following: hovering colour when button is on is darker green; hovering colour when

						button is off is lighter grey.
16	Settings sliders	Moved the sound effects and music volume sliders on the settings window	mouse_x mouse_y mouse_leftb	Moving the sliders should scale from numbers 0-100	As expected	-
17	Settings Volume	Moving the music and sound effects slider will scale the volume of the game	mouse_x mouse_y mouse_leftb	Moving the music and sound effects will change the volume of the music ranging from 0-100. I know if this works by hearing the volume of the music and clicking on buttons plays	As expected	-
18	Settings fullscreen button	Fullscreen button toggles fullscreen	mouse_x mouse_y mouse_leftb	If you press the fullscreen button and the game is in windowed mode, it should make the game fullscreen. If you press the fullscreen button and the game is in fullscreen mode, it should put the game in windowed mode	As expected	-
19	Settings mute button	Mute button mutes sound	mouse_x mouse_y mouse_leftb	Settings button disables all sound when pressed and re-enables sound when pressed again	As expected	-
20	About screen	About screen should show and draw the correct details and buttons	-	When switching to the about state, the game displays the Turris creators and two buttons: GitHub and Website	As expected	-
21	About screen button interactions	Test to see if the buttons are interacted with the mouse	mouse_x mouse_y	When hovering over github and website button with the mouse, the buttons display a paler colour to display that they can be interacted with or clicked	As expected	-
22	About screen button press	Test to see if the functions of the buttons perform the correct task.	mouse_x mouse_y mouse_leftb	When clicking the buttons with the mouse,	As expected	-

Game Play

Playing

Test No	Test Name	Description	Data Input	Expected Outcome	Actual Outcome	Remedial Action
1	Grassy Greens	Tiles load in correctly from the level_1.csv file	Mouse_X Mouse_y Mouse_leftb	Tiles display correctly and the enemy takes the correct path from the beginning of the map to the end. I will also check to see if turrets should place where they're supposed to place and don't place where they're not supposed to place	As expected	-
2	Sand Dunes	Tiles load in correctly from the level_2.csv file	Mouse_X Mouse_y Mouse_leftb	Tiles display correctly and the enemy takes the correct path from the beginning of the map to the end. I will also check to see if turrets should place where they're supposed to place and don't place where they're not supposed to place	As expected	-
3	Tricky Track	Tiles load in correctly from the level_3.csv file	Mouse_X Mouse_y Mouse_leftb	Tiles display correctly and the enemy takes the correct path from the beginning of the map to the end. I will also check to see if turrets should place where they're supposed to place and don't place where they're not supposed to place	As expected	-
4	Standard Mode	Check if game ends on winning	Mouse_X Mouse_y Mouse_leftb	The game ends when you reach the level's max level. And says congratulations, you win.	As expected	-
5	Continuous Mode	Check if game doesn't end	Mouse_X Mouse_y Mouse_leftb	The game doesn't end when surpassing max level. Waves come through as specified by the wave multiplier algorithm in the code. The wave templates should follow the	As expected	-
6	Enemies	Enemy behaviour	-	When starting the game, the enemies follow the paths that were laid out. The spawns are delayed as specified in the round file.	As expected	-
7	Rounds end	Rounds should end	-	When all enemies of the round are dead, the round ends	As expected	-

8	Enemy death rewards	Enemy death provides you with gold	-	Enemy death provides you with the correct amount of gold	As expected	-
9	Losing lives	When enemies pass to the end off the track and go off the screen, you should lose a life	-	Enemy surpassing the track should remove a life.	As expected	-
10	Enemies health	Enemies get shot with arrows	-	When enemies are shot with arrows, they lose health according to the damage of the arrow	As expected	-
11	Enemy speed	Enemies move at the correct speed	-	Enemies move at the correct speed that they're given, relative to the speed modifier of the game	As expected	-
12	Button Interaction	Buttons should dim when hovering over it with the mouse	Mouse_X Mouse_Y	Hovering the mouse over buttons dims the button to show that you can interact with it.	As expected	-
13	Start Button	Starts the round	Mouse_X Mouse_Y Mouse_lef tb	Pressing the start button starts the game	As expected	-
14	Pause Button	Pauses the game	Mouse_X Mouse_y Mouse_lef tb	Pressing the pause button pauses the game and sets the button text to 'Resume'	As expected	-
15	Resume Button	Resumes the game	Mouse_X Mouse_y Mouse_lef tb	Pressing the resume button unpauses the game and sets the button text to 'Pause'	As expected	-
16	Speed Modifier Button	Changes the pace of the game.	Mouse_X Mouse_y Mouse_lef tb	1x is regular speed 2x is double regular speed 4x is four times regular spd.	As expected	-
17	GUI box for player's progress	Shows a box for coins, lives, the round they're on and the cost of the turrets	-	The gui box shows	As expected	-
18	Players progress stats	Shows the stats: coins, lives, the round they're on and the cost of the turrets	-	Shows the player stats in the gui box	As expected	-

19	Towers selection	Towers can only be selected if you have the correct amount of money	Mouse_X Mouse_y Mouse_leftb	A tower can be selected if you have the correct amount of money. The game should now display a grid of where to place the turret if you can afford. Otherwise, the turret buttons should appear red and they should be unable to be interacted with.	As expected	-
20	Tower Cost	Testing to see if the turrets display the cost and displays them correctly	-	Hovering over the tower button should display the cost (in green if you can afford them and in red if you can't afford then).	As expected	-
21	Turret range placing	Turret should display turret range when placing	Mouse_X Mouse_y Mouse_leftb	The turret shows the correct turret radius when placing	As expected	-
22	Turret range	Turret should display turret range when interacting with the specified turret	Mouse_X Mouse_Y	The turret shows the correct turret radius when interacting with the mouse	As expected	-
23	Placing turret	Testing to see if placing the turret works and place onto the grid	Mouse_X Mouse_y Mouse_leftb	Placing the turret onto the grid and displays correctly and able to interact with it. Placing the turret should scale with upgrades too	As expected	-
24	Turret Functionality	Turret stats	-	Depending on what turret you place, it will have the correct textures, damages and damages that increase with upgrade. The turrets display the correct texture and only fire arrows when they have an arrow ready and they can only fire them an enemy if they're in their radius.	As expected	-
25	Turret upgrades	Correct turret functionality on upgrade	Mouse_X Mouse_Y Mouse_rightb	Upgrading the turret, the turret displays the correct upgrade cost. This also increases the turret damage and other stats depending on the turret on upgrade. Finally, the upgrade only happens if you have the right amount of coins. This would then take that amount of coins away from the player.	As expected	-

26	Arrows	Arrow functionality	-	Arrows fire in the correct direction at the correct speed and when collided, they do the correct amount of damage to the enemy. The arrows also display the correct texture	As expected	-
27	Selling	Towers can be sold	Mouse_X Mouse_Y Mouse_rightb	Towers are the only cells that can be sold. When the tower is sold, you are given a small portion of money returned. The tile then returns to its previous tile. The cursor displays red on tiles that can't be sold and green on tiles that can be. The portion of money returned is clearly shown above the tile to be sold in yellow text. The radius of the turret should also be shown	As expected	-
28	Save Game button	Save game button should save the game to the start of the round	Mouse_X Mouse_Y Mouse_leftb	Save game button should save the game to the start of the round. Saving the game should also pull up a menu and pause the game.	As expected	-
29	Settings button	Settings button should open the settings	Mouse_X Mouse_Y Mouse_leftb	The settings button opens the settings menu and pauses the game.	As expected	-
30	Quit game button	Quit game button should open a menu	Mouse_X Mouse_Y Mouse_leftb	Pressing the quit button opens a quitting menu and pauses the game	As expected	-
31	Finishing the game	Winning or losing the game in standard mode	Mouse_X Mouse_Y Mouse_leftb	Winning or losing the game in standard mode shows you these correct statistics: difficulty, round reached, lives, total revenue, total kills, arrows fired, buildings built, buildings upgraded. Winning the game will display 'Congratulations! You win!', losing the game will show, 'Unlucky you lose'. There's a new button that can be interacted with and when clicked, it exits the game to the main menu. If you lose the game, the save of that game is removed as you have lost	As expected	-

32	GUI button	Testing if the GUI navigation panel works	Mouse_X Mouse_Y Mouse_leftb	The GUI navigation panel button opens the navigation panel and changes direction to open and close the menu. The navigation panel should display all buttons.	As expected	-

GUI that pauses the game

Test No	Test Name	Description	Data Input	Expected Outcome	Actual Outcome	Remedial Action
1	Settings screen	Settings screen should display all buttons and sliders	-	Clicking on the settings button should display all the buttons and sliders	As expected	-
2	Settings sliders	Moved the sound effects and music volume sliders on the settings window	mouse_x mouse_y mouse_leftb	Moving the sliders should scale from numbers 0-100	As expected	-
3	Settings Volume	Moving the music and sound effects slider will scale the volume of the game	mouse_x mouse_y mouse_leftb	Moving the music and sound effects will change the volume of the music ranging from 0-100. I know if this works by hearing the volume of the music and clicking on buttons plays	As expected	-
4	Settings fullscreen button	Fullscreen button toggles fullscreen	mouse_x mouse_y mouse_leftb	If you press the fullscreen button and the game is in windowed mode, it should make the game fullscreen. If you press the fullscreen button and the game is in fullscreen mode, it should put the game in windowed mode	As expected	-
5	Settings mute button	Mute button mutes sound	mouse_x mouse_y mouse_leftb	Settings button disables all sound when pressed and re-enables sound when pressed again	As expected	-
6	Settings buttons	The buttons (apart from back) checked if interactable	mouse_x mouse_y	If the user hovers the mouse over the Fullscreen and Toggle mute button, the background of the mute button should dim to seem interactable	The buttons didn't dim when you hover the mouse over it.	Fixed the issue and uploaded to github. The colours now change to the following: hovering colour when button is on is

						darker green; hovering colour when button is off is lighter grey.
7	Quit save	The quit button should display a menu when clicked	mouse_x mouse_y mouse_leftb	The quit button should display a menu when clicked and create text asking if you want to save your progress as quitting deletes your progress. The menu also creates three buttons, quit, continue and 'save and quit'. The functionality of the buttons works and are completely interactable.	As expected	-
8	Save game	The save button should display a menu when clicked	mouse_x mouse_y mouse_leftb	The save button should display a menu when clicked. This should create some text which says, 'Game saved' and 2 buttons to quit and Continue. You can interact with these buttons and when you press quit, you quit to the main menu. When you press continue, you continue the game that you're currently on.	As expected	-

Sound

Test No	Test Name	Description	Data Input	Expected Outcome	Actual Outcome	Remedial Action
1	Volume	The volume of the sliders set in the settings changes the volume	Mouse_x Mouse_y	The volume of the sliders set in the settings changes the volume	As expected	-
2	Music Main menu	Music on the main menu plays and stops in the correct state	-	The main menu music plays on the main menu and in the settings, on about and map selection correctly and at the correct volume	As expected	-
3	Music playing in the Playing state	Music plays on the playing state	-	The playing state music plays on the playing state and in the settings in the playing state and any other menu that appears within the playing states	As expected	-
4	Lose life	SFX	-	plays when you lose a life	As expected	-
5	Loss	SFX	-	plays when lose the game	As expected	-
6	Enemy dies	SFX	-	1 of 2 plays when enemies die	As expected	-
7	Enemy hurt	SFX	-	1 of 2 plays when enemies get hurt	As expected	-
8	Menu Click	SFX	-	Plays when you click something in the main menu	As expected	-
9	Round complete	SFX	-	Plays when a round is completed in the game	As expected	-
10	Turret place	SFX	-	plays when you place a turret	As expected	-
11	Turret shoot	SFX	-	Plays when a turret shoots an arrow	As expected	-
12	Turret upgrade	SFX	-	Plays when you upgrade a turret	As expected	-
13	Victory	SFX	-	Plays when you win the game	As expected	-

Project Report

People and roles

Who was on the project?

- Oliver Legg - 201244658
- Thomas Coupe - 201241037
- Kieran Baker - 201234727
- Daniel Taylor - 201234245

What did people do?

In the development of Turris, there was a lot to do as the project was quite optimistic. We needed to make sure that the implementation of the game was similar to how we designed within the design document. To do this, whilst implementing the game we used the Gantt chart designed to organise roles during the implementation. However, we came across some issues due to people not completing their tasks on time and not contributing, meaning that other people needed to complete tasks that were not assigned to them. Below is a table showing some of the implementations made by the contributors:

Contributor	Tasks completed
Oliver Legg	• Implemented the foundations of Turris • Implemented the GUI, main menu and how the user will respond to the usage of this user interface • Implemented the fundamental states of the game win states, lose states and overall structure of the game • Implemented the Enemies and their pathfinding • Handled the IO of rounds, saves and loading of games. • Tested the game and fixed bugs
Kieran Baker	• Implemented the audio system within Turris • Implemented the map selection and continuous mode. • Implemented the Turret upgrades • Helped with the texture design and decorations • Developed the online documentation • Fixed and polished many aspects of the game
Tom Coupe	• Created Settings page, Help page and About page. • Created the slider used to control volume • Designed some of the map, enemy and turret textures. • Implemented some of the GUI features.

How was the group organised?

As previously mentioned, we organised and assigned tasks to people by using the Gantt chart designed during the design documentation. However, this did not work effectively. Instead, when implementing Turris we would communicate via an online chat where we would decide who would be doing what. We primarily used the design document to decide what we needed to implement next. People would then discuss what they were going to implement in the group chat allowing other people to work on different features that needed implementing. Oliver Legg focussed on the actual implementation of the gameplay and the saving/loading system. Kieran focussed on features that were described during the design process such as map selection, turret upgrades and even the audio system used within Turris. Tom Coupe focussed on some of the GUI features such as the settings, help and about pages along with designing some of the in-game textures.

When developing Turris, we were used Git and GitHub to work on the project. Git helped a lot within the organisation of the project, it allowed us to see what other members of the team were working on, as well as the ability to see the code other people had implemented. Git allowed us to revert to previous versions so that if we didn't like how a new feature was implemented, we would revert to the previous version. Putting the files on GitHub was very easy to access and allowed us to pull and merge what other people had been doing on the project. Git and GitHub has been an incredibly important tool in the project, we wouldn't have reached our goals without it.

Another way we organised the tasks in the project was to create lists of features that were not yet implemented in a text file. We did this every few days and shared it with each other and by doing this, we were able to focus on certain tasks sequentially. This depended on us communicating with each other and letting each other know what features we're all implementing.

Application Overview

Application domain

Turris was made to create a fun and easy to play tower defence game which is aimed at all types of users. We wanted to give the user a choice between three maps each ranging in difficulty and also a choice between the classic tower defence mode and continuous mode. Everything we set out to complete in the mission statement has been implemented in Turris - along with many other features. We have expanded upon our original mission statement by adding features - such as buying/selling turrets, customisable settings and even obstructive decorations on maps. This prevents you from placing turrets in strategically advantageous positions, making the game harder. Overall, we have implemented everything we wanted to from our original mission statement, along with many other additional features.

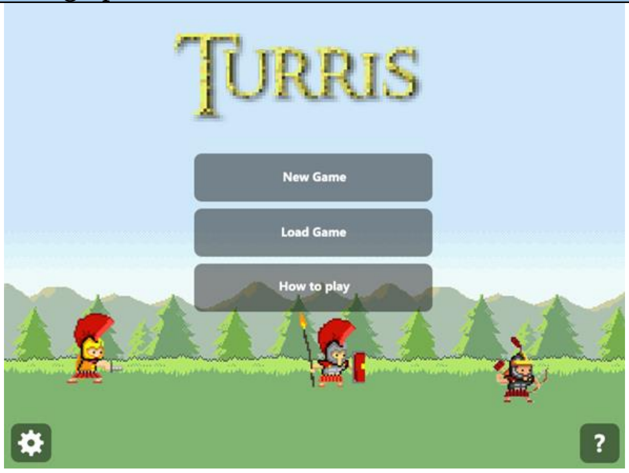
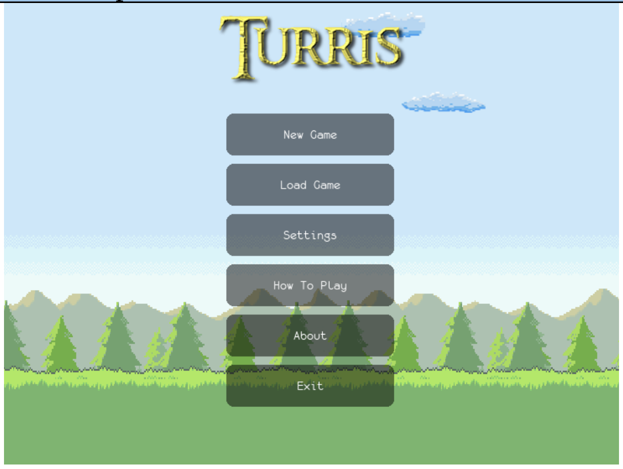
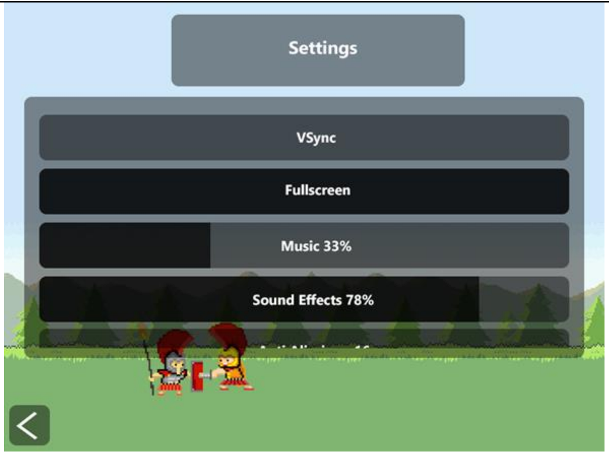
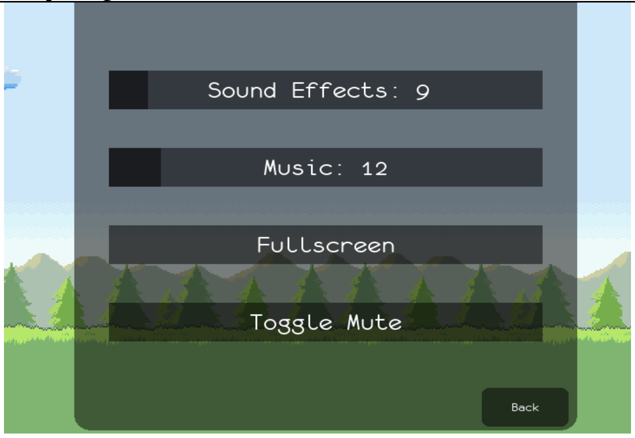
Types of user

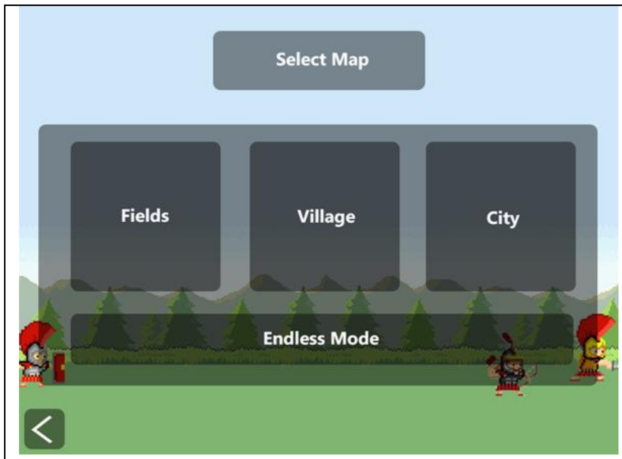
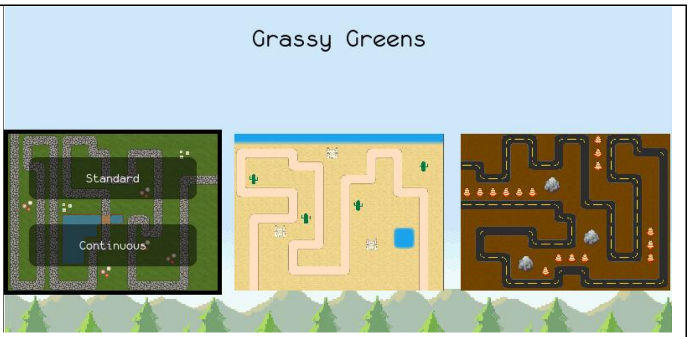
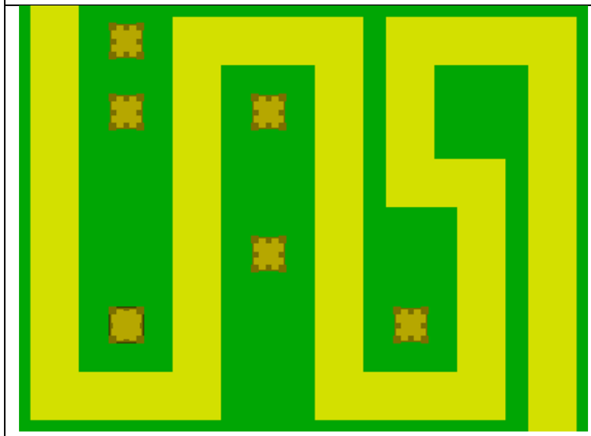
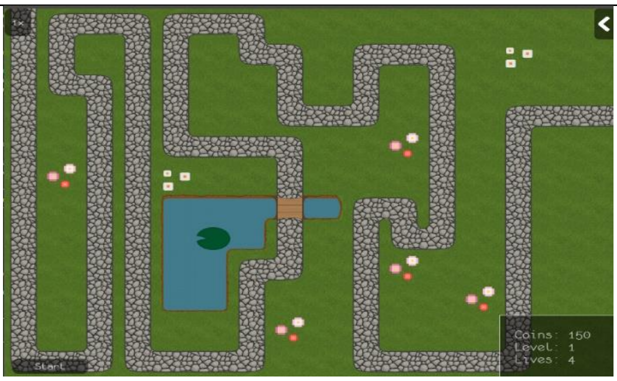
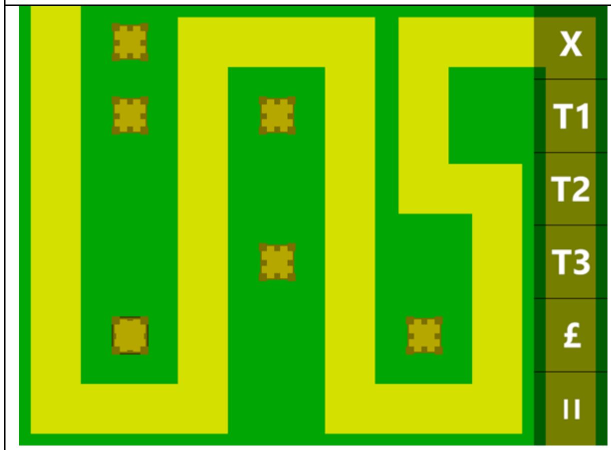
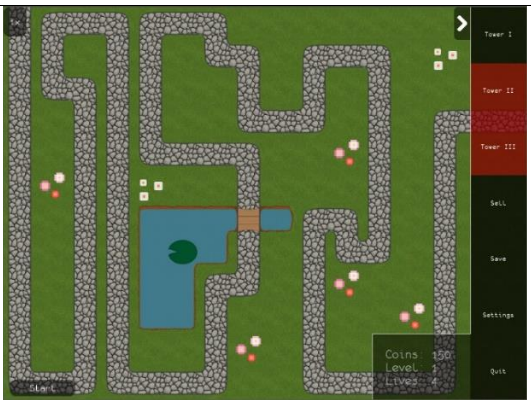
To play Turris, you don't need any other skills than a basic understanding of what a tower defence game is – which you will be educated on, in the how to play documentation. The main types of users of our system would most likely be kids and teenagers. However, the system is easy to use for all types of users, thus, the system is not restricted to a specific type of audience. The user interface of Turris is very simple and easy to navigate. You don't need any in depth computer skills, other than knowing how to user the mouse. Our online user guide gives the user of a detailed description of how to use everything in Turris, this user guide is very simple and easy to follow. Overall, there is no specific type of user for our system as Turris is a simple and easy to play game. However, we would expect that the system was used by a younger audience in general, hence the easy and user-friendly GUI.

Achievements

What did we produce?

When implementing Turris, we produced exactly what we wanted. We used the requirements and design documentation to decide what features we needed to add and how we would go about adding them. As you can see, from some of the screenshots below we clearly followed the design documentation when implementing Turris. Below is some examples of what was designed and how those features were actually implemented.

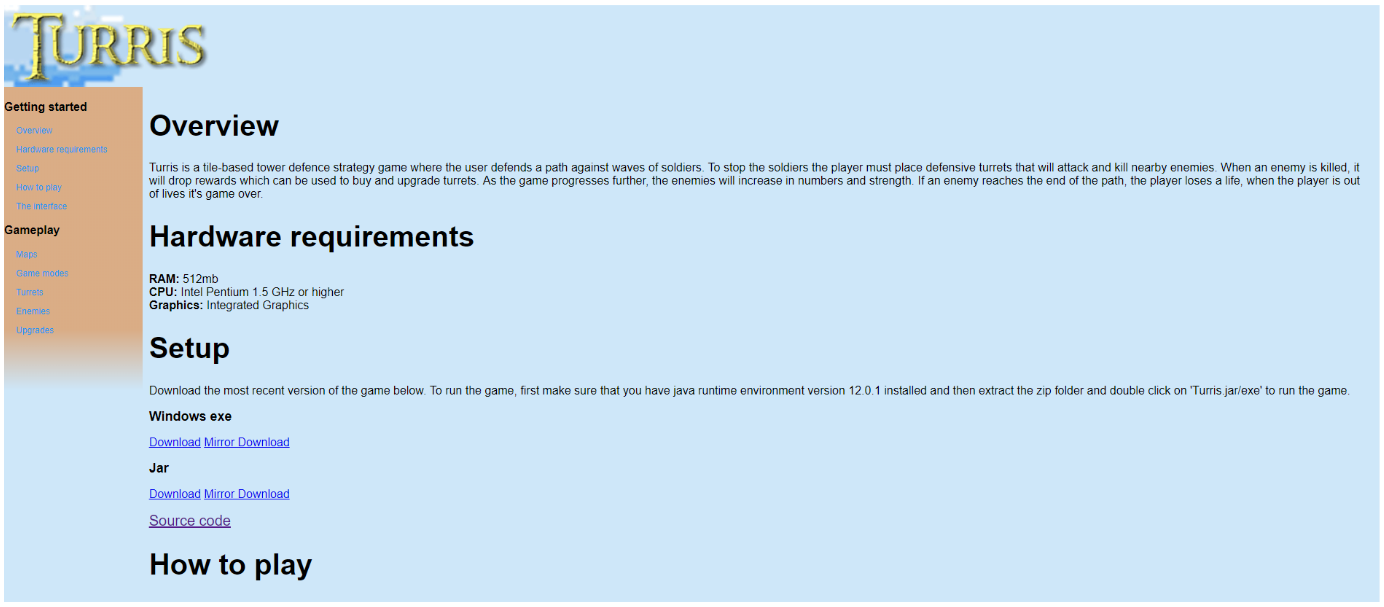
Design process	After Implementation
 This was the design of the main menu. The screenshot shows the original design, which has the settings page and the about page in the bottom corners. The rest of the buttons are in the centre.	 TURRIS New Game Load Game Settings How To Play About Exit In the implementation, we decided to keep all the main menu components together. This makes the navigation of the system much easier to use and we think this is an improvement to the system. The design did not include an exit option which was a very important feature to have.
 Settings VSync Fullscreen Music 33% Sound Effects 78% This is the design of the settings page. The settings page design includes sliders to control sound effects and volume as well as toggle buttons to enable Vsync and full screen.	 Sound Effects: 9 Music: 12 Fullscreen Toggle Mute Back This is the final implementation of the settings page. We kept the sliders and toggle buttons as designed. However, we removed the Vsync button as this limited the number of computations to your monitor's refresh rate per second. This prevented us from being able to use the speed modifier in game – therefore it was removed

 This is the design of the map selection feature. This feature was a very important part of Turris; therefore, we wanted the implementation to be like the design.	 As you can see, the implementation was very similar to the design however it is more efficient and compact this way. We were going to have two separate selections for the game type, however this allows you to select the map and the game mode in the same place.
 This was the design of how the gameplay was going to look. This design was lacking in detail as we did not know during the design process what textures we would be using.	 The screenshot above is the design of the "Grassy Greens" map. As you can see, the implementation of the gameplay was quite different to how we designed it during the design process. However, we are much happier with the implemented design.
 This screenshot from the design process shows the in-game navigation system that the user interacts with in the game.	 Tower I Tower II Tower III Sell Save Settings Quit Start Coins: 150 Level: 1 Lives: 4 The implementation of the GUI is much like the design with some additional features such as in-game access to the settings and the ability to save the game. The abbreviations and symbols in the design were changed it was unclear as to what some of the buttons did. Therefore, we added more detail

	in the button descriptions. We also added more GUI, including the number of coins the user has, the level they're on and the lives they have remaining.

Extra Features Implemented

Turris was heavily based upon the design process when it came to implementation. However, there were some features added to the system that were not discussed during the design process. One example is the in-depth user guide on the Turris website. We originally did not design having a website that users could go to for help with using the system.



Overview

Turris is a tile-based tower defence strategy game where the user defends a path against waves of soldiers. To stop the soldiers the player must place defensive turrets that will attack and kill nearby enemies. When an enemy is killed, it will drop rewards which can be used to buy and upgrade turrets. As the game progresses further, the enemies will increase in numbers and strength. If an enemy reaches the end of the path, the player loses a life, when the player is out of lives it's game over.

Hardware requirements

RAM: 512mb
CPU: Intel Pentium 1.5 GHz or higher
Graphics: Integrated Graphics

Setup

Download the most recent version of the game below. To run the game, first make sure that you have java runtime environment version 12.0.1 installed and then extract the zip folder and double click on 'Turris.jar/exe' to run the game.

Windows exe

Download Mirror Download

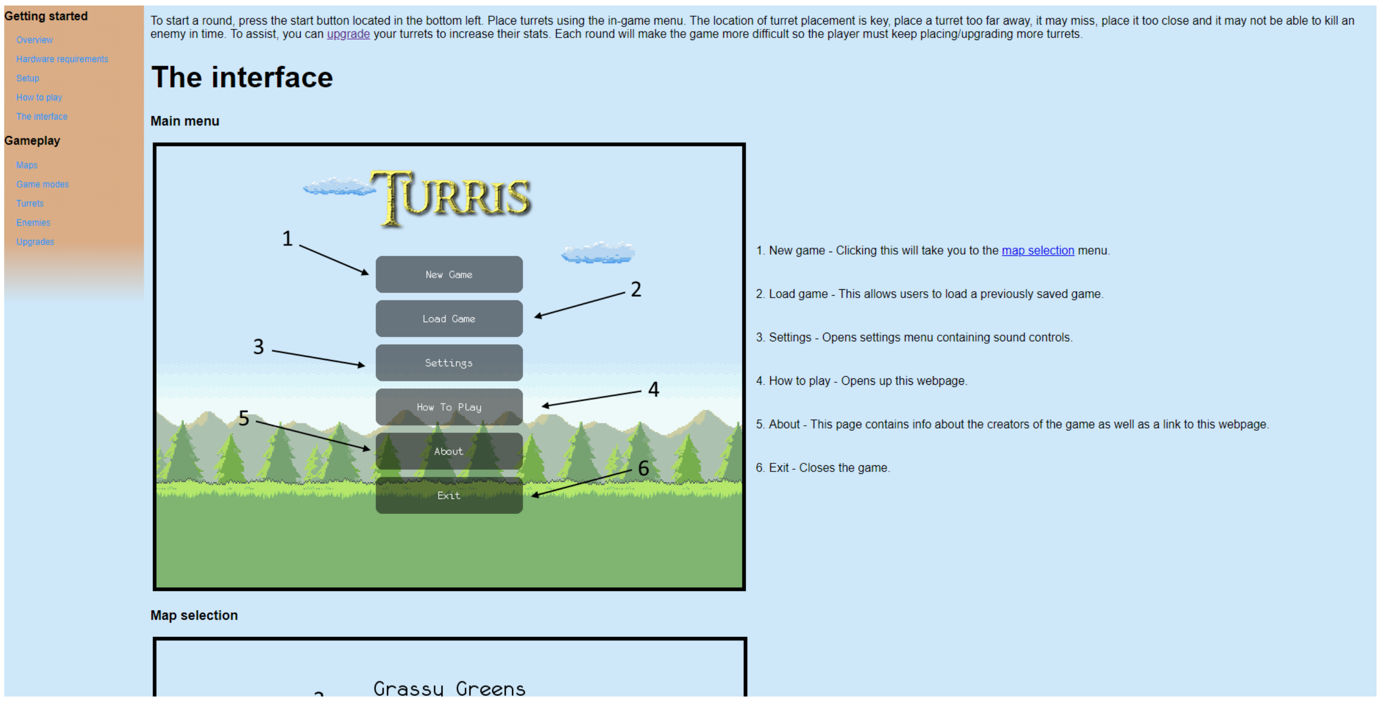
Jar

Download Mirror Download

Source code

How to play

However, during the implementation process, the "how to play" section was not in depth enough for someone who had never used the system before. This led us to implement the user manual on the Turris webpage. Below are some screenshots showing the information on the Turris webpage. Having the how to play on a website makes it easy to change too. If we add new content to the game, it would be easier to add this to a website instead of changing the code for the how to play section in the game.



Getting started

Overview

Hardware requirements

Setup

How to play

The interface

Gameplay

Maps

Game modes

Turrets

Enemies

Upgrades

To start a round, press the start button located in the bottom left. Place turrets using the in-game menu. The location of turret placement is key, place a turret too far away, it may miss, place it too close and it may not be able to kill an enemy in time. To assist, you can upgrade your turrets to increase their stats. Each round will make the game more difficult so the player must keep placing/upgrading more turrets.

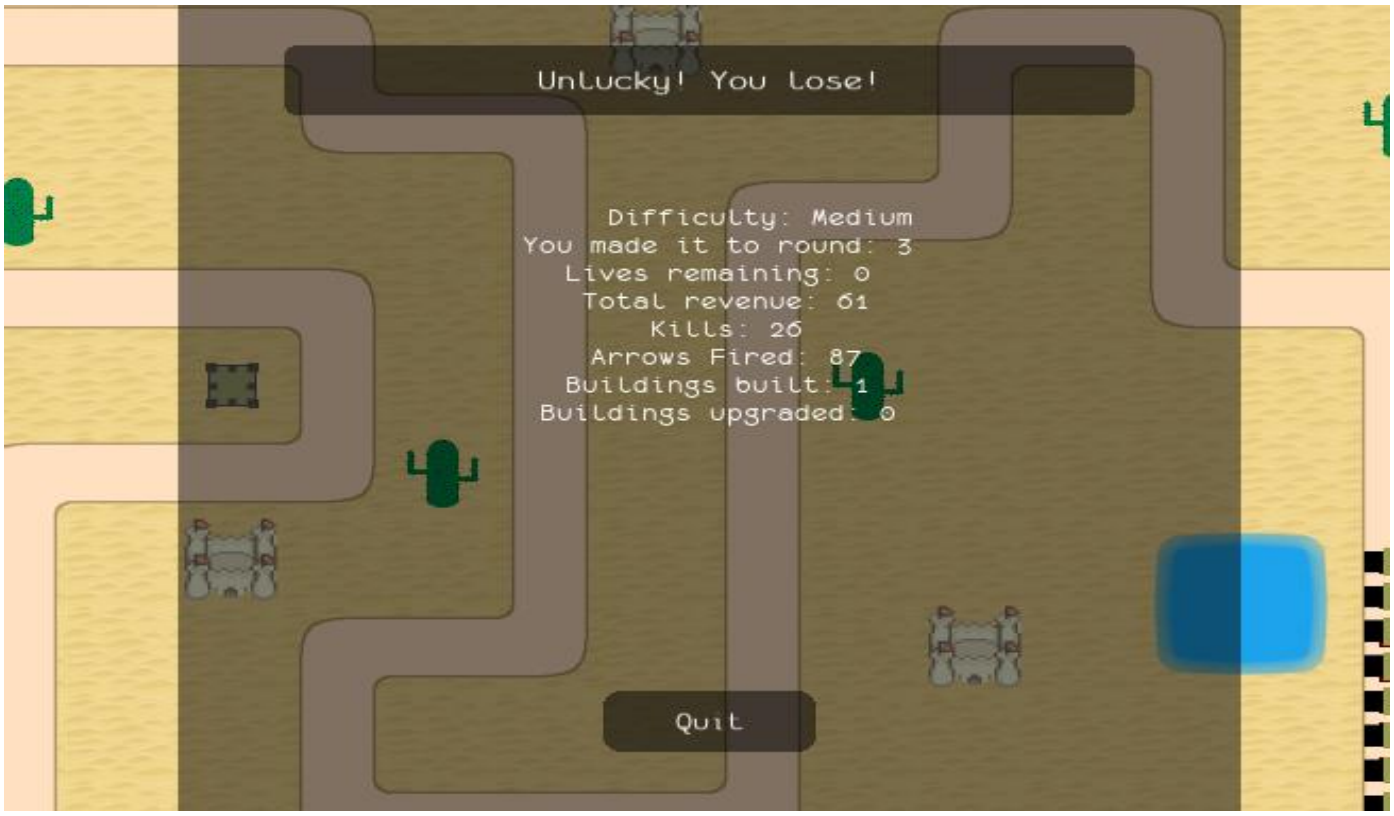
The interface

Main menu

1. New game - Clicking this will take you to the map selection menu.
2. Load game - This allows users to load a previously saved game.
3. Settings - Opens settings menu containing sound controls.
4. How to play - Opens up this webpage.
5. About - This page contains info about the creators of the game as well as a link to this webpage.
6. Exit - Closes the game.

Map selection

Another extra feature we decided to add which wasn't in the design documentation, was to give the user of our system a detailed overview of statistics which details how they did when playing the game. This means that the user can keep track of how well they have been doing in each game and can compare with previous games and potentially, other players. These statistics are displayed to the user at the end of each game no matter if they won or lose. Overall, we thought this was a good feature to add to the system, even though it was not in the design process, it makes the game much more rewarding to play and promotes potential competition to the game.



Difficulty: Medium
You made it to round: 3
Lives remaining: 0
Total revenue: 61
Kills: 26
Arrows Fired: 87
Buildings built: 1
Buildings upgraded: 0

Evaluation of the system

Overall, we feel that the implementation of Turris went well, we implemented almost everything specified in the design process. The actual gameplay of Turris is much better than we anticipated, this includes how the user interacts with the game along with the look and feel of the system. The system has an in-depth user manual which gives an in-depth overview of how Turris works along with how to use the system. The whole goal when implementing Turris was to create a fun and easy to play tower defence game and we feel we have met this goal. We have also provided the user with customisation of the system allowing them to adjust volume, adjust sound effects and even to enable full screen mode. The gameplay of Turris is the most important part of the system and we feel that it has been implemented exactly how we wanted it to be. The gameplay is really fun and the users of the system can select between different map difficulties which makes the game much more challenging. We also implemented a save/load game feature which allows users of the system to save their progress for the next time they want to play.

During the implementation we did come across some problems which were not expected. For example, originally, we gave the user of the system the ability to choose the max fps (frames per second) that their game would run at. We thought this was a good feature as it gives the user of the system much more customisation making the system feel more interactive. However, when implementing this feature, it caused many issues with lag and even caused problems with the turrets when they fired arrows. Due to these problems we decided to remove that feature from the system. If we were to add this feature again, we would perhaps have to implement it early on within the development process to avoid causing problems later on. Another problem we had was that as the rounds increased the game got very computationally expensive and slowed the game performance down. However, this problem was fixed fairly easily, it was due to the fact we were not deleting the arrow textures once they left the screen causing a lot of lag.

During the implementation we tried to follow the design documentation to ensure that our system was developed how we had designed it. However, some of the requirements specified in the design document did not make it into the implementation. For example, we were originally going to have multiple arrow types that the turrets would fire, however when it came to the implementation, we stuck with one arrow type for all turrets. We think this was a good thing not to implement as it may cause issues when trying to balance the game as the turrets already had different firing rates and ranges. Another design feature that did not make it into the implementation was the ability to allow users to select the difficulty on each map. Instead we decided to make the maps themselves have different difficulties, for example the first map “Grassy Greens” is an easier difficulty than the third map “Tricky Track”. Again, this is not what was designed however we felt that this was a better feature.

Among the contributors of Turris the functionality of the team was good. We would communicate ideas and implementations on a group chat which allowed everyone to give input on changes to the system. The good communication also meant that we knew what everyone was working on, allowing us to start working on other things that were not started yet. Overall the way we communicated as a team made the implementation process of Turris much more efficient. People each had different areas of the game that they would work on, for example some people would focus on the implementation of the game play whereas other people would focus on the GUI and textures. By each working on different areas it made the whole development process much more efficient.

Future development

Although almost all the features we originally intended to be in the game have been implemented, there are still a few more ways which we would like to improve and develop the game in the future to create a more interesting and fun experience for the users.

Scoreboard/Leader board

Currently Turris has a set of statistics that are shown to the player after winning/losing a game. Once the user has closed the screen, the statistics are lost no matter how good they are. In the future, a scoreboard system could be added to the main menu where the player will be able to track their best statistics such as highest round reached, or most enemies killed. To improve this system further, the user could enter their name and an online leader board could be created where all the players can compare their scores. This will give users an incentive to improve at the game and perform the best they can to reach the top of the leader board.

Enemies

The enemies in the game are currently quite basic, they all follow the same rules just with different properties. In the future, we could add more enemy types that don’t follow the standard rules such as; Enemies that split into more enemies upon death, enemies that regain health as they move or stealth enemies that can only be killed by certain types of turrets.

Turrets

Another improvement that could be made in the future would be to add more turrets each with their own properties. To accompany this, the maximum level of turret could be increased. To take this even further, the new turrets could have special abilities such as slowing enemies down or firing projectiles that penetrate through multiple enemies. Adding these changes to the game could create balancing issues, however, the new enemies can be used to balance this out.

Maps

The next development in the future would be to improve the maps. Firstly, new maps could be added with potentially a new difficulty level. Each map in Turret is currently of a fixed size, the size of new maps could be varied to create a more complex environment. Increasing the size of maps would also enable the addition of more features such as multiple paths for the enemies to move along or crossing paths where the enemies may not always follow the same route.

Accessibility

Currently Turris is only setup for windows systems. The game is written in java, so making it available on more platforms will be a very simple but effective improvement. The UI of the game is currently a fixed size however this could be an issue for a person using the system with poor eyesight and they may be unable to use the system effectively. To solve this issue, an option to scale the UI could be added into the settings menu.

Professional Issues

Introduction:

When creating a program or project using any programming language for profit or commercial purposes, it is important that you know the code of conduct as well as the things you can do and cannot do so that you do not get into legal trouble.

During the Software Engineering module, I learned that you can’t copy code for commercial benefit, however you can refer to algorithms that have already been created and change them in such a way to fit in your program. It is a great way of being able to find other ways of completing challenging problems.

There are many articles online that give information regarding the things you should remember so that you do not get into any legal trouble when developing your source code, however the best place to start is the BCS (British Computer Society) Code of Conduct in the U.K. In this section I will be referring to this information to

BCS Code of Conduct:

The society itself, is an organization that has separate rules to the standard general law. However, they are stricter and the disciplinary action for not following these rules results in expulsion from membership. This means that for any professional coder / company that are trying to make a profit or gaining copyrights for certain algorithms will have a lot of trouble, due to the BCS having some sort of involvement with this process.

It is important to note that the members are expected to use their own judgement to meet the requirements of code and seek advice if they are in doubt. This means that if there is any confusion relating to the code or the result, you are to seek help through the team you are working with, or the customer that you are going to be providing for.

The codes are broken down into four distinct sections, each of those sections have multiple items that are mandatory to comply with if you are going to remain a member of BCS.

Public Interest:

In terms of public interest, there is not much that we can relate to with the game we have created (Turris). However, one of the main codes is to "Conduct your professional activities without discrimination on the grounds of sex, sexual orientation, marital status, nationality, color, race, ethnic origin, religion, age, or disability, or of another other condition or requirement. We had to make sure we took this into consideration when creating our game, as we are making a game with turrets and zombies. This could have easily been changed to humans instead, however that could have been violent towards certain people. It is the small things like this that must be considered and not looked over, as it can cause you to run into legal problems.

We also had to make sure that the code itself does not contain any offensive material, there could always be a variable name that can be seen as offensive, this is why we made sure we looked over the code that we created in order to satisfy these rules.

Professional Competence and Integrity:

There are a couple of items in this part that go together, one of which is to only undertake work or provide a service that is within our professional competence. This means that we should only do something if we feel comfortable that we can complete it. In terms of our project this was a common practice, coming up with certain features that we would eventually undertake had to be feasible and realistic, so that we could complete the tasks. There is also the added item that states we should NOT claim any level of competence that we do not possess, I believe that this is something that relates to our reviews that we regularly had on the project, due to the fact that we were constantly asking the reviewer for recommendations. If they asked us if we could do something on our project, we would either say yes or no, but we would never claim that we could do something if we did not feel comfortable completing that task.

Another item in this section talks about progression. According to this item we are to "develop your professional knowledge, skills and competence on a regular basis, maintaining awareness of technological developments, procedures, and standards that are relevant to the field". This is extremely important when it comes to developing games, because the idea of technological developments can benefit the work that we are doing. In the future, there could be a better way of handling certain source code, either a different algorithm or library that can run the game we are developing and the assets within it a lot better. It also relates to another item in this section, which is respecting alternative viewpoints and always seek criticisms of the work. We stuck to this idea, we always made sure we knew that there were other ways of handling the situation, after looking at the benefits and negatives of each, we then decided which way of solving the problem. We also took the feedback from the advisors on the group software project module as well as the reviewers that looked at our project through the steps of the assignment.

Duty to Relevant Authority:

In terms of our project, there is not much to say for this section. A lot of the items do not relate to the work we are doing, usually the items within this section are for programs that are going to be considering sensitive information, which shouldn't be disclosed to the public on a regular basis.

However, I can make a comment on one of the items, which is "accepting professional responsibility for your work and for the work of the colleagues who are defined in a given context as working under your supervision". We always took responsibility for the work that we did, if we ever ran into any issues, it would be dealt with correctly and the responsibility would be split between the group if there were multiple areas that conflicted with each other. This should be a common practice as it keeps the team working together on problems that should be fixed with multiple people.

Duty to the Profession:

In terms of development, we always were looking to improve and that is one of the main items and practices of the BCS. In order to satisfy the item, we are to "seek to improve professional standards through participation in their development, use and enforcement". This means that we should always be improving our entire project through testing and practice. This is something that should be done on every project to make sure that all the requirements are satisfied for the customer or manager.

Bibliography

- Turris user manual - https://student.csc.liv.ac.uk/~sgkbaker/Turris/Documentation.html
- BCS (British computing society) issue a code of conduct - https://www.bcs.org/ - 2019