*Jack Maguire*

*Candidate Number: 0109*

*Candidate Centre: 51518*

*apex turrets*

# Table Of Contents

[Table Of Contents 1](#_Toc29914857)

[Analysis 2](#_Toc29914858)

[Research 2](#_Toc29914859)

[Background and Problem Definition 2](#_Toc29914860)

[Analysis of Similar Systems 2](#_Toc29914861)

[Kingdom Rush 2](#_Toc29914862)

[Bloon TD5 3](#_Toc29914863)

[Brackeys Tower Defence Game 4](#_Toc29914864)

[Problem With Current System 4](#_Toc29914865)

[Identification of Users, Their needs and acceptable limitations 4](#_Toc29914866)

[Identification of Users and their needs **Error! Bookmark not defined.**](#_Toc29914867)

[Acceptable Limitations **Error! Bookmark not defined.**](#_Toc29914868)

[Objectives 6](#_Toc29914869)

[Proposed Solution with justification of Language 7](#_Toc29914870)

[Design 8](#_Toc29914871)

[Technical Solution 9](#_Toc29914872)

[Testing 10](#_Toc29914873)

[Evaluation Error! Bookmark not defined.](#_Toc29914874)

# Analysis

## Research

### Background and Problem Definition

Currently, at KCS, we have about 1000 pupils and numerous members of staff, and other than free periods, we have two breaks, one in the morning for 20 minutes, and one in the afternoon starting at either 12:35 or 1:15 (sometimes there is an extra lesson), and ending at 1:40 (for sports), or at 2:10 (for lessons afterwards). In the other half of the school (the junior school), there was a well-equipped playground, with monkey bars, play-frames and table-tennis tables. There was also the library (homework was forbidden in the day). In the Senior School, however with vastly more pupils, and there are only spaces in which to gossip, or the library in which there are computers (homework only), or the ICT Suite (in which Games are banned).

This creates a problem, that the people who don’t gossip, can get easily bored, with only homework, reading or sitting around to do. My proposed solution, a Tower Defence Game aims to solve that, in that the IT Staff allow self-coded Games, and that no previous pupils have made large scale games, similar to any standard of Game Development (pupils are taught only python at a basic level, and most students take a dislike to the Subject due to the teacher for the school being shouty, and that you are only taught in the first year (Year 9)).

A Tower Defence Game, is a game which is 3rd person based, and one has enemies to kill, but instead of shooting them like most games, you build turrets or towers to kill the enemies. These cost money, but as enemies die, they give you money, and you can build more towers. If the enemies get to the end of their path however, then you lose lives, and if you have 0 lives, then Game Over. You win the level, if you kill all of the enemies. As you progress through the game, there are more types of enemies, and towers, and you can upgrade the towers. It resembles some other games like Factorio, in that you try to automate it so that you don’t have to do anything. In more advanced versions again, you have power-ups or heroes.

## Analysis of Similar Systems

### Kingdom Rush

One good example of a tower defence game is Kingdom Rush, a popular game that features a lots of features. The UI is clear and well-defined, and there are a lot of levels, to the point where I have not yet come close to finishing the game. It features no ads, and the only way of them making money, is buying more in-game currency (used for things like ‘Freeze all Enemies’), or buying heroes (they sit where you tell them, and kill enemies as an AI would in a typical shooting game).

However, this game may be available on Steam and Mobile, but steam and downloading .exe files has been blocked by the school, and mobile phones are banned for the whole day.

Below are 2 photos, showing the first level of the mobile version,

A picture containing indoor, grass, table, cake

Description automatically generatedA picture containing indoor, table, grass

Description automatically generated

These images show the first level of the game. The UI is clear, giving an idea of lives remaining, money and current wave. To build a tower, you tap on the spot, and it gives options. This also comes in later, for selling or upgrading towers. Link to app store:

<https://apps.apple.com/gb/app/kingdom-rush/id516378985>

|  |  |
| --- | --- |
| Pros | Cons |
| * easy to use * free * no ads * lots of levels, turrets, enemies and extras * clear ui | * **Only downloadable via steam, or phone.** |

### Bloon TD5

A screen shot of a computer

Description automatically generated

Next, is a different yet still fun style, or bloon TD5. In this version, enemies are balloons (bloons), of different speed and popping strength. Then, instead of set turret spaces, there are areas to put down balloon poppers, and some are valid, and others invalid. In this case, you could only put down turrets on land on the planets. The strategy of the positioning with limited money, makes it quite interesting. Many types of turrets exist, and keyboard shortcuts can be used to quickly grab types of turrets to place, a system I will integrate into my own game. However, the game is hosted on Kongregate, with limitations such as a need for flash, which is being discontinued by popular browser Chrome soon. Link: <https://www.kongregate.com/games/Ninjakiwi/bloons-td-5>

|  |  |
| --- | --- |
| pros | cons |
| * easy to use (slightly less so than kingdom rush, but not much) * free * lots of turrets, levels, extra | * **Ads** * **broswer-based (needs internet)** * **slightly cluttered ui** |

### Brackeys Tower Defence Game

There are several game development youtubers (for beginners), the most famous that I have found being Brackeys, a Unity Game Developer with over 900,000 subscribers. He has several series’ for making a game, one of which is a tower defence game, and while this could not be counted as a release itself, it was one of the only such games, that could be downloaded locally for free on a Mac (there is no release, follow the tutorial). This is inherently free, and has different levels, tower upgrades, different enemies, and most importantly, the easy ability to generate more enemies. The enemies are spheres, but work fine. The two turrets can be placed on any node on the map, and cost money. The UI looks clean due to a general lack of anything unnecessary.

A picture containing crossword puzzle, text

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Above is the main level (left), the level select (middle), and the start of the game (right). Firstly, and most important is the game itself. The nodes are made of cubes, and compose the entire map, other than the enemy path. Turrets can be placed, and after they are bought, a simple click is needed to place it, and they all have full rotation. Lives are shown at the top, and money, time remaining, and turrets to buy are all at the bottom. The waves are predetermined, and the level select, gives options to load different levels (all contained in similar scenes, not too different from JavaFX scenes, In that they can exist, while not being shown). Then the play screen is the model of one of the turrets, that rotates with text to either quit the game, or go to the level select screen. Link to github repository below:

<https://github.com/Brackeys/Tower-Defense-Tutorial>

|  |  |
| --- | --- |
| pros | cons |
| * free * desktop * no wifi needed * clear ui | * **have to follow tutorial / download unity to get it.** * **simplistic** * **limited assets** * **simplistic enemies / ground** |

## Problem With Current System

## Identification of Users, Their needs and acceptable limitations

Questions for Client:

**Q1: What do you currently do during our breaks?**

A: Currently, I either walk around and talk to friends or sometimes use the time to catch up on work or communicate with teachers. During the lunch break, I sometimes attend a club (which lasts around 30 minutes).

**Q2: What problems are there with this?**

A: I do agree that there are times when there is nothing to do sometimes (especially during lunch break), which can lead to boredom. Sometimes my friends may be ill/occupied with something or clubs may be cancelled, which leaves nothing to do for about an hour.

**Q3: How do you think these problems could be tackled with my proposed solution?**

A: This solution provides a system which, while tackling the obvious issue of boredom during breaktimes, may also motivate other students to create their own games for others to play.

**Q4: Do you believe that there are any essential features that are currently missing?**

A:

* As the system runs on a user login basis, there should be a way for a player to play the game (as much as possible) without signing in or connecting to the database.
* The game should have a tutorial level which explains the basic mechanics of the game to a new player.
* There should be a level list screen, which allows the user to go back an replay a level whenever they want to. Future levels should be visible but inaccessible (locked).

**Q5: Are there any features with my proposed system that you find unnecessary?**

A: While a message feature in the leaderboard is perhaps a nice addition, it would lead to possible bugs and errors in formatting. As well as this, a profanity filter would not be able to catch any implicit profanity (innuendo, racist connotations, etc.)

**Q6: Do you have any ideas about UI, or anything else to do with the Front-End?**

A: The UI should be as easy to use as possible for both old and new players:

* The player should be asked to login (or continue as guest per my request in Q4) **before** seeing the main menu.
* There should be a button the top left / top right of the main menu screen which allows the player to sign in/access their account settings.
* The main menu should then have the essential options for any game:
  + New game
  + Resume game
  + Select level
  + Leaderboard
  + Quit
  + (Anything else)

Ben has asked for reasonable things, and I will attempt to put in all of these.

## Objectives

1. The game should not be repetitive.
   1. There should be different turrets.
      1. Firstly, there should be at least two different turret templates.
      2. Then there should be upgrades available for each turrets (at least one, but more should be able to come.)
         1. An upgrade constitutes of an increase to a statistic positively, such as more damage, or shooting more often.
   2. There should be different enemies
      1. Currently, there is one enemy that is fast, but can be shot down fast.
      2. And another that is the reverse.
   3. There should be different levels
      1. First, will be set levels, in which enemies come and go, in a set order (determined by a config file), and score will be determined by money remaining, money used, and hearts remaining.
      2. Then there will be an infinite ‘gauntlet’ mode, in which enemies are thrown semi-randomly (there will be checks not to be too slow, or too fast), and score is determined based on factors like time between enemies, enemies killed and the above factors for the set levels.
2. The game should be able to differentiate users.
   1. A user can select a difficulty, and have that kept for them.
   2. Each user should be able to keep a unique account.
      1. A password will be required, of a certain length.
      2. As well as a username of a certain length.
      3. The password will be hashed using SHA256, and stored in that way.
      4. The attributes of each player scored locally:
         1. Overall Score,
         2. Levels complete and score for each,
         3. Gauntlet high score;
      5. As a new user is created, a new file is created with the attributes of that player.
3. The game should be as efficient as possible.
   1. Storage Space Efficiency
      1. All turret, enemy, and level data will be stored on a Github repository, and fetched at runtime.
4. The players should be able to share statistics.
   1. The email will be stored, so a python script could be used to email others about the score.
   2. There will be a leaderboard for set levels, and for the Gauntlet.
      1. On the leaderboard, it will be in rows, sorted by score, and only display the top 10, and it will show the username, a message, and their score.
         1. In case of any unsavoury language, each word will be checked against a database (<https://raw.githubusercontent.com/RobertJGabriel/Google-profanity-words/master/list.txt>), and censored appropriately (eg. F\*c\*).
5. Running Example:
   1. The user will be presented with a main menu, with a Welcome Screen and two buttons to either login or to see the leaderboard.
      1. If they click on the leaderboard, then it will show it, as in 4.2.1
   2. If they click login, a new window will open to login, with an simple GUI to create a user, or to login.
      1. The details are stored locally on a cfg file.
   3. Once the user has logged in, they will be able to choose to replay a level, or to progress to the next.
      1. If they choose to progress to the next level, it will be the same as a previous only a different level shown.
         1. Then the level will be shown, with a grid system for the tiles, an enemy base, a home base, and GUI for purchasing towers, as well as to inform about lives and money.
            1. The grid system is that each icon has an ID, and dependant on a cfg file, it dynamically shows different tiles to change up the level.
            2. At the enemy base, the enemies spawn, and they slowly make their way across ‘path’ squares, and attempt to get to the home base. If they reach the base, then the player loses an amount of lives, and points for the end of the level. If the player runs out of lives, then Game Over.
            3. To combat the enemies, the player can buy turrets, and these can shoot the enemies, to kill them before they reach the base. The turrets get better, but so do the enemies, and turrets can be upgraded.
            4. In the shop, the player buys the turrets.
      2. The player could also play the Gauntlet, which is the same, only the level never ends until game over.
   4. If they choose to replay a level, they play through it as a normal level, and when it finishes, the old data from the player’s file is deleted, and the new data is inserted.

## Proposed Solution with Justification of Language

After analysing the similar systems, I have decided to use Java as the language, a language famous for desktop game development, for its versatility and ease of use for all platforms.

## Use of Java

Java is a relatively easy programming language to use, similar to C# and C, and is widely used, with famous examples including Minecraft. I have also decided to use a popular IDE, IntelliJ IDE. I have submitted an application for the free full student version, and I have succeeded, and so am using IntelliJ IDEA Ultimate. It has auto-code-completion, based on previous code, and it has a dark mode. The other main option was Eclipse, famous for its Class Wizard, and external library manager, but before moving to Java, I coded in C# for a few months, and got used to the auto-fill of Visual Studio (also the Community Edition), and so typing with no code completion felt strange.

IntelliJ also has syntax highlighting, and an automatic constructor for the constructor, getter, setter, override, and toString methods. I have now been coding using Java for more than a year, but with C# for about 2 years, and experience can be transferred over, with little difference.

Due to its popularity, Java also has many questions and answers on online forums such as stack overflow, which helps if the documentation is unclear or vague (which can happen sometimes with Oracle).

|  |  |
| --- | --- |
| pros of using java | cons of using java |
| * lots of community questions and answers. * widely used. * versatile. * very good ide. | * **JDK must be installed.** * **for games – no built-in entity/rendering system (like unity)** |
|  |  |

# Design

## Textual Overview

In the game – it starts with the game double checking whether the user wishes to play. If they say no, the program ends. If they say yes however, it launches into the program. Then, the UI is shown for the main game – a full screen window, with a collection of squares and a path to the left, and buttons and a status text area to the right.

First, the left hand side – a collection of squares, forming a path, and after an initial grace period (30 seconds), enemies start spawning going from the start of the path to the end of that path. The start is the square that looks like a satanic ritual, and the end is the bluer square. These enemies come in waves – a few enemies with a set gap between them, and then another set gap, before another wave and so on. Each enemy has a value depending on their strength, and you have a certain amount of lives. If an enemy gets to the end – then it takes away however much it is worth from the player’s lives.

Now, for the right hand side – a few buttons and a status area. There is one button for each type of tower, and when that button is clicked – information appears about the tower. An option is then given to select a location for that tower. If there is no space left, an error message is given to the user. If there is not enough money left, then an error message is given to the user. If they select a location, money is subtracted from the player’s account and the turret appears where the user specified. Under is also a sell button – and the player has the choice to sell a turret – NB: Each turret has a sale value, that is **less** than their buying value to ensure that each buy is worthy and there are no unlimited money schemes.

Finally, the status window – this shows the player how much money and how many lives they have left, as well as all of their turrets (name, position, initial investment, resale value).

## System flow diagram - listen to the code.Flowchart

## Class Diagrams and Algorithm Explanations

### Config File Reader

### Coordinate Class

### Custom Change Listeners

### Resource Manager

### Square Class

### Base Entity Class

### Turret Class

### Enemy Class

### Canvas Class

### Player Manager Class

### TurretFrame and TurretManager

### Waves

### Main Class

Hi

# Technical Solution

# Testing

# Evaluation