

BETA DEFENCE



William Cheung

Abstract

An account of the analysis, design and creation of a single player tower defence

Submitted as part of the OCR Computer Science coursework requirements: H446/03
The Sixth Form College, Colchester.

Contents

Analysis	6
Introduction	6
Stakeholders	6
Research.....	6
Talking with the stakeholders.....	6
The Tower- Idle Tower Defence.....	7
Summary of The Tower - IDLE Tower Defence	12
Bloons Tower Defence 6	13
Summary of Bloons Tower Defence 6.....	28
Success Criteria	29
Software/Hardware Requirements.....	32
Prototype Development/Technical research.....	32
1.Entity Collision	32
1.1 Creating the player.....	32
1.2 Making the player radius	37
1.3 Making Enemies	39
1.4 Enemy Ai	41
1.5 Enemy Spawn Overlap	45
1.6 Enemy Overlapping.....	47
1.7 Tower Overlapping.....	49
1.8 Tower Projectile	51
Menu Research	53
2.0 Making a Basic Menu	54
2.1 Basic Health UI	57
Upgrade Shop Research.....	58
Project Decomposition	59
Interface Designs	60
Game Menu	60
Game.....	61
Shop	62
Pause Menu	63
Upgrade Info	64
Leader board	65
Internal and External Storage	66

Server side.....	66
Player Table.....	66
Client-side	66
Tower class.....	66
Enemy class.....	67
Shop class.....	68
Button class.....	68
Algorithms.....	69
Menu.....	69
Draw Menu	69
Enemy	70
Random Enemy	70
Move Enemy	70
Tower	71
Draw Tower.....	71
Detect In Range.....	71
Game.....	72
Current Health State	72
Development log.....	73
Version 0.1	73
Drawing the basics	73
Enemy creation	74
Version 0.2	77
Enemy creation issue	77
Version 0.3	80
Adding expression to enemies.....	80
Projectile Creation	81
Version 0.4	81
.....	82
Limiting the projectiles created	85
Version 0.5	85
Projectile Collision.....	85
Enemy targeting.....	88
Cooldown	89
Manual Fire	93
Version 0.6	97

Connecting the Menu	99
Making a pause Screen	102
Version 0.7	108
Health Bar System.....	108
Version 0.8	112
Creating a losing screen	112
Version 0.9	115
Balancing the game.....	115
Version 1.0	118
Shop UI.....	118
Development Testing.....	121
M1 Manual Tower.....	121
M2 Enemies	122
M3 Health	123
M4 Menu.....	124
S1 Tower radius.....	125
S2 Simple Enemy Ai.....	126
S3 Difficulty in game	126
S4 Enemy Creation.....	126
S5 Projectile	127
S6 Home.....	127
S7 Simple Upgrades	127
S8 Earning Money	127
COULD Criteria	128
C1 Waves.....	128
C2 Types Of Enemy	128
C3 Settings.....	129
C4 Sound/Music	129
C5 Particles.....	130
C6 Powerups	130
C7 Additional Upgrades	130
C8 Shop/Shop UI	130
C9 Advanced Enemy AI	131
C10 Statistics	131
C11 Leader board	131
C12 Losing Screen	131

WON'T Criteria.....	132
W1 Multiplayer	132
W2 Transactions.....	132
W3 Cosmetics.....	132
W4 Login	132
Evaluation	133
Stakeholder Testing	133
Starting the Game	133
The Tower	134
Enemies.....	134
Shop System.....	135
Impressions on Game UI.....	135
UI Screens	135
Results from stakeholder testing.....	136
Starting the Game	136
The Tower	137
Enemies.....	139
Shop System.....	140
Impressions on Game UI.....	141
UI Screens	141
Success of the project.....	143
M1 Manual Tower + S5	143
M2 Enemies + S2 + S4	143
M3 Health System.....	143
M4 Menu + S6	145
S1 Tower Radius.....	145
S3 Difficulty + C1	145
S8 Earning Money	146
C3 Settings.....	146
C8 Shop/Shop UI	146
C12 Losing Screen	146
Assessment of usability features	147
Limitations of the project.....	147
Future Development	148
References	150
Code	151

Index.html.....	151
Button.js.....	152
Enemy.js.....	153
Game.js	155
Style.css.....	162
Tower.js.....	163

Analysis

Introduction

After playing various tower defences throughout my childhood and enjoying them quite thoroughly, I thought it would be a great idea to design and create my own tower defence game. My project will be based on the single player tower defence game, The Tower – Idle Tower Defence with a few modified and additional features of my own.

A tower defence game is a strategy game where the player decides a way to prevent enemies reaching their “base” and will typically upgrade a tower or multiple towers using currency dropped from defeated enemies in order to achieve this. A variety of tower upgrades are used allowing the user to be quite free with their choice of strategy instead of a set one. Enemies will usually progressively get harder across set waves or rounds, thus will drop more currency in order to compensate for this. The game will either continue endlessly until the player either has completed all the available waves in which they are rewarded with perhaps cosmetics or other in game currency or has lost due to the game eventually becoming too difficult.

This project will exhibit a reasonable level of complexity suited for A level as User-drawn graphics will be used to constantly monitor the movement of enemies and the player’s mouse pointer around the canvas; object-oriented programming will also be used mainly to create projectiles, players and enemies in the game; the use of a physics library may be used to deal with entity collisions (when an enemy touches a player or a projectile touches an enemy) or the direction projectiles will move. Also, computational methods like complex algorithms will be used for an example, in the movement from the start screen to the game screen; the pausing of all entities in the game when the pause button is clicked and resetting all the players stats and difficulty when the player has lost the game.

Stakeholders

My primary stakeholder will be my cousin, Jayden, who will be primarily suggesting potential ideas that could be added into the game and will be involved throughout the project by iterative testing of the game as it is being developed, by providing constructive valid feedback and will also help find any potential bugs or glitches that may need to be fixed to allow me to make sure the game to be a success and enjoyable experience as a whole.

In the project, I have the largest stake as it is up to me for the development and the completion of the project. Currently I am quite weak in programming due to the lack of practise, therefore, from the completion of the project, I will have hopefully expanded on my current limited knowledge and confidence of programming in JavaScript and later on intend to continue Computer Science in University, which I hope will eventually lead into a job related to this area of expertise.

By the completion of the project, I hope to have a working basic tower defence game that has all the basic functions of a tower defence game with additional features that may have not been used from other games before that the user will enjoy to continue playing and won’t easily be bored from.

Research

Talking with the stakeholders

When thinking of potential ideas that I could add in my game I had a small discussion with my primary stakeholder, Jayden on what a basic tower defence game should include with a few additional ideas inspired from different games.

These Include:

- A set amount of money added to the players money total
- Game difficulty
- Shop Interface
- Settings – including sounds and music
- Different maps that can be chosen or randomised
- A set amount of enemies
- Different types of enemies
- Cap limits for certain upgrades
- A set or endless amount of waves
- Player spawning with a set amount of health – could potentially be changed

And potentially:

- Draggable Objects
- Certain powerups – letting tower shoot quicker for certain amount of time
- A certain build a tower can take – tank build, attack build
- A set path or route the enemies will take
- Perhaps targeting options – attack the strongest enemy first or quickest
- A variety of smaller other towers with other upgrades.
- Certain upgrades being unlocked once certain round achieved
- XP system – player levelling
- Player username
- Login system – allow players to save their data and have individual accounts

I also had a small discussion with my friends who suggested:

- A working tower if definitely needed in order for the enemies to work and vice versa
 - Enemies need to pathway find themselves to defeating the objective
 - Based on “The Tower” the tower should only be able to hit enemies in that radius (being a circle)
 - When using upgrades balancing should be taken into account when creating the towers.
 - If enemies are to go by waves, enemy’s stats should be ramped up according to the wave.
- The Tower – Idle Tower Defence

The Tower- Idle Tower Defence

The Tower - Idle Tower Defence is a game made by the company Tech Tree Games in 2021 and is currently available on the apple play store, android play store and various game websites on the internet according to The Tower – Idle Tower Defence Wiki Fandom (Fandom W. , 2022), which is a fan made Wikipedia on the game that is regularly being updated involving details on the tower, upgrades and other feature included in the game.

In the game, the player is a tower that will automatically seek out enemies within their line of sight which is a circle radius. The main aim of the game is to survive as long as possible from enemies by upgrading their stats from money dropped from defeating enemies – these

stats can be upgraded whilst the game is continuing in the background, maintaining an interactive gameplay. As rounds the difficulty will increase: the enemies will have more health but have less speed and will also drop more money, furthermore, the quantity of enemies spawned will also increase. Enemies will have a set path they will take per round to ensure every round will be different and challenging for the player. There is a tournament mode, but I won't include it in since my project as my tower defence will be single player.



The start screen of the game contains the battle button that will load you into the game and will start the game.

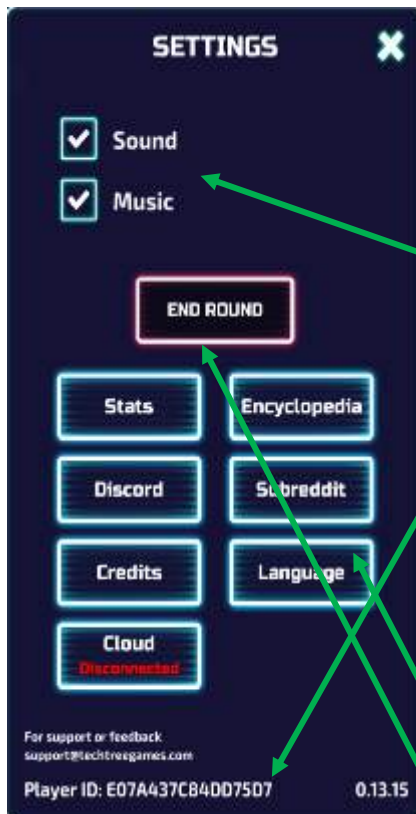
In the very top left, the c in the circle is a permanent coin that can be gained by defeating special yellow enemies in the game or can be obtained by defeating tough enemies. This currency can be used by clicking the hammer icon will bring you to the workshop which gives the tower permanent upgrades and also buy additional skills that will also help the tower.

In terms of the upgrade system, the blue sword icon are upgrades that will benefit the tower through its attack stat, allowing the player to do more damage to an enemy so the waves would be easier. The red shield icon will have upgrades that will help the tower with sustaining more damage such as reducing the amount of damage all enemies will do the player through the defence upgrade; the star icon will not benefit the tower immediately but will over time by allowing the player to gain more money this is done through upgrades like increasing the amount of money the player gains per round or the money gained from defeating enemies by a set multiplier, these can be manually upgraded by the player in game or like other upgrades can be permanently upgraded to a set amount using the coins.



When purchasing the Multishot upgrade, 2 new potential upgrades were added to the attack upgrades section, %chance for the tower to shoot multiple bullets, starting at a 1% chance at base and an additional upgrade, multishot targets which at base level would shoot 2 additional bullets to the first bullet if the multishot %chance had happened.

After unlocking multishot, Unlock Rapid Fire Upgrades will appear and replacing the previous multishot button and costs 1500 coins thus being more powerful but costing a lot more.



The setting button will lead to multiple amounts of options with the additional option to end the round if the player is currently in the game.

There is a toggleable option for sound and music allowing the user to hear the game music and sound produced by the towers if they wanted to.

Every user has a randomly generated 16-digit code – a mix of capital letters and numbers allowing more than trillions of users to be created due to the number of combos that can be generated- to allow the user to have their own individual account that can be saved and backed up with a cloud account, their game data will be synced automatically and saved so their data can be accessed from a different device without having to make another account.

The language option is a customisable button to change the whole in-game language for the user, the current available options for language for version 0.13.15 are English, Portuguese, Spanish, Italian, Russian, Korean, Chinese, French and German.

END ROUND button will only appear if the player is in a game and will quite simply end the round the player is currently on and also end the current game the player is playing.



The stats button shows the player a list of total statistics based off the players such as how much money earnt, enemies defeated, when player started and other in game statistics (as shown in the diagram on the left) that will automatically be updated once the player has met the requirements like completing another wave in a new game and then accessed the stats button in the settings.

Other game stats will be shown to the player such as when the player is defeated. This will show the player what round they got up to, their personal best round and what type of enemy they were killed by and how many coins they earnt in the game.

The retry button will restart the game, putting the players stats back to default or up to the permanent upgrades that have been bought in the workshop.

Home will put the player back to the home screen, allowing the stats button to update its contents.



When defeated, this game stats UI will appear, and the background game will be stopped and cannot be continued.

There is also a multiplier button that will lead you to watch an ad in order for the game to gain ad revenue and allow the player to earn more coins to buy permanent upgrades for the next game so they will hopefully get further in the round than before.



Gems are a premium currency that can only be obtained in small amount through in game boxes that appear very infrequently in games and is obtained by clicking on a diamond shaped box when they appear. These gems can also be bought using IRL money to buy large quantities of these gems in order to speed up the progression of the game.

The gems are used to buy card slots which go up to a maximum of 6 card slots a player can own. Gems are also used to buy cards, which like the coins are used to give the player permanent upgrades to make the game easier to progress for the player. Duplicate cards will allow the player to upgrade these cards to improve the buffs they give the player. These cards currently only have 3 rarities: common, rare and epic, epic being the most strongest and the rarest to obtain.



When you load in the game you will be met with a hexagon and a circle. The hexagon will be your tower and the circle is the amount of range your tower can see. If an enemy were to enter the range of the circle the tower would start shooting, prioritising the nearest enemy in range, this means if an enemy had entered the circle the tower would immediately start attacking it, however if there was another enemy that is quicker or closer to the tower, the tower would retarget on to that enemy instead. If many enemies are the same distance away from the tower such if they were directly on the tower, the tower would target the first enemy that had reached the tower until it is defeated, this will go on until all the enemies are gone.



Another currency, cash is dropped from enemies. As usual in most games, the tougher the enemies are, the more cash they will drop. This cash will be used to upgrade the tower with the upgrades the player has unlocked, though, these upgrades are not permanent and will reset once the player has been defeated.

I also noticed during the game if enemies were instantly defeated, they would show (for roughly 1 second) how much money they dropped and with larger tougher enemies doing this with the addition of the amount of damage the tower did showing up next to the enemy as each projectile hits the enemy.

Furthermore, once an enemy was defeated, a range of small circle particles would appear in the enemy's colour portraying that the enemy was killed by the tower for the player.

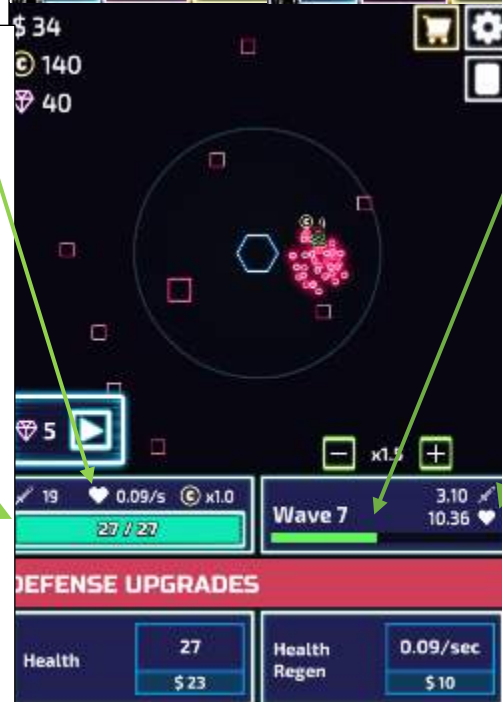
When analysing the enemies, I realised on round one and onwards set enemies would spawn in set amounts and intervals on each round but would spawn in random locations all through the entire game, making each game different from each other perhaps used to make the game more strategic from a perspective since you would not know where each enemy would spawn but you would know what type of enemy would spawn and the amount that would spawn. The tower would also sometimes lock onto one enemy one round whilst a different game the tower would be on a different enemy.



The game additionally has a speed button, being able to increase the game speed by +0.5 and -0.5 the maximum being x1.5 game speed and the lowest 0, which ends up pausing the game with clear white text on the screen, game paused. Even though I wasn't able to unlock this in my own playthrough, according to: (Fandom W. , 2022), you can further increase the game speed up to x5. The use of this game speed button will either increase or decrease all entities on the map rate of fire (tower) and the movement speed and attack (enemy)

In The left bar of the upgrade system, the amount of damage the player does, amount of regeneration the player will passively heal and the amount of coins x by a multiplier based on how far the player has upgraded it is shown to the player.

The left bar items update immediately as soon as the player decides to upgrade a stat linked to the icon. Upgrading health in this situation will immediately change the health bar from 27/27 to 31/31 since I am full health.



The wave bar will increment once per round once it is completed and the green progress bar will show the player how much of the wave they have completed until the next wave will appear.

Upgrading stats that will benefit overall damage from the tower/defence/health for an example: attack speed or upgrading health will result in the average damage and health bar to be updated once per wave.

Whilst play testing, in a situation where I am 3/27 health and I had just upgraded my health; my health would be increased by roughly 30% of my current max hp so I would now have 4/31.

Buying upgrades in game will require the player to press the upgrade button on the desired upgrade they want but they need the amount of money to do this. If the player can buy the upgrade, the upgrade box containing the amount of money it would cost will have a light blue border around it clicking on this will use the players money, apply the upgrade to the tower, and replace the old price of the upgrade with the new price also updating the value of the new stat. Whilst if the player cannot afford the upgrade, the box will be slightly darker blue indicating that the player cannot afford the upgrade and the button being unclickable. If the player has just recently got enough money, there is roughly a 1 second delay before the button changes from dark blue to light.

Summary of The Tower - IDLE Tower Defence

From my own personal test of the game: The Tower - IDLE Tower Defence, I can conclude it was an overall fun experience of a tower defence game that kept the user engaged in the game through its grindy status though that was also its downside. The Tower contained your typical upgrading system health/defence with a few upgrades such as additional money and further additional special upgrades that could be unlocked through an immense grind from the player, as some upgrades were locked behind certain high wave requirements. The tower defence had very interesting special effects when the enemies were defeated: enemies exploding into particles and had a ton of different strategies that could be applied by any different player picking up the game. Whilst the game was a very large grind and very sided towards paying to win to progress quickly, the game has a large amount of variety from the range of upgrades and randomness from the enemy spawning system making every game wave unique to a different strategy. Furthermore, the need to grind the game allows the player to be more addicted and put more time into the game. The addition of sound of music and sound from the towers was quite exciting, it allowed a more realistic engaging game rather than a just a player tapping their screen. However, my biggest critic of the game was how slow the game progression was since coins and gems were the most needed currency, they were the hardest to obtain and it didn't help the drop rates of certain items were quite low, this led to other in game events being accessible to only long-time players since most of them were locked due to players needing to complete high waves that were impossible without large investment into the game.

Pros:

- Overall is fun game, it had plenty of upgrades and abilities the player could choose from
- Particle effect from defeating enemies was really nice
- Game upgrade system could pop up and be closed allowing player to see the game more easily
- A small range of enemy types allowing them to be easily remembered
- Encyclopaedia – informed user on what certain upgrades did and info on enemies
- Music, sound togglable option
- Language option allow users of different language to play game

Cons:

- A bit too grindy – when farming mobs they barely or almost never dropped any coins
- Quite catered towards pay to win – buying gems in shop gave you massive progression compared to a f2p player, upgrade packs have massive value.
- Too many upgrades ended up confusing me and also other players online.
- Certain events like tournaments were locked behind high wave completion therefore making the game a very long-time consuming process since tower upgrades took a long time to buy for.
- Ads whilst is beneficial for the player to get extra boosts only gave a slight increase in currency and players can just buy this boost permanently if they were to buy it rather than watch the ads.
- Card rarity is just too low for the player to receive.
- No tutorial for the player – the player has to learn the game for themselves
- Extremely high investment in game is needed whether it be money, or a lot of time is required to progress.
- Music is limited

Bloons Tower Defence 6

Bloons Tower Defence 6 (BTD6 as many fans call it) is a popular game made by the company Ninja Kiwi and is the sixth instalment of the Bloons TD franchise. The game was released on Android and iOS on June 13, 2018 and was later brought to Windows and Steam (18th Dec 2018). On all platforms BTD6, cost money to purchase the game on Steam it will cost £7.19 and on mobile devices it will cost £5 but every so often will have a discount on holidays or special days. Bloons TD 6 is a huge update from the previous Bloons games which introduced new and improved 3D graphics from the old 2D graphics, line of sight mechanics, heroes, 5 tier upgrades, a third path upgrade and a lot more according to the Bloons TD 6 fandoms (Fandom B. T., 2022)



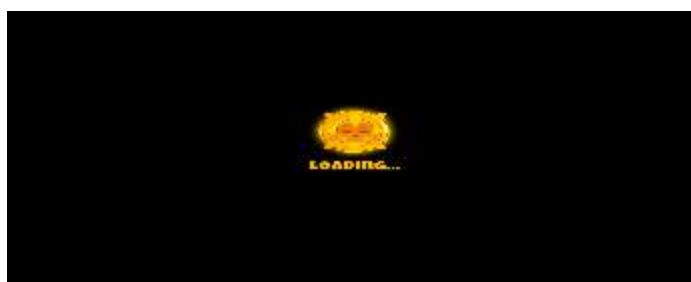
When researching for a tower defence game to potentially gain ideas from, I looked around on the app, Steam, which is a software used to buy and play new and popular games released and when searching tower defence, the first result was a familiar title to me: Bloons Tower Defence. Having played Ninja Kiwi's previous games: Bloons Monkey City and Bloons TD Battles that had 2D towers models and thoroughly enjoying the games, I knew this game would be very beneficial for my research to make a fun lasting game that the players will enjoy. The game currently (16/06/22) has an outstanding amount of positive reviews and is highly recommended to play when reading the most recent reviews. Furthermore, there was a massive special discount on the game since, reducing the price to £1.60 from the £7.19, therefore, I decided to buy the game.





After downloading the game files, I was met with a plane animation transitioning to the start screen page which has a custom Bloons Tower Defence Art.

Firstly, I tried clicking on the I've Played Before to see what it does. This ended up leading me to a sign up/ login page where the user can create or login into another account that they own, allowing the user to own multiple accounts and so, I signed up and created an account using my email in order to save my progress and transfer my data easily if I needed to use another device.



When signed up and clicking the start button the screen transitions from the start screen to a loading screen as seen on the left. This screen is used many times in the game when the player is loading a map into a game or an even so will be commonly seen.



There is quite a short but vague story that the game tries to make, simply, bloons are invading the monkey village (map) and the player, who we can assume to be a monkey has to place towers to defend the bloons from reaching the end of the path.

The button will immediately bring the play to a tutorial to teach the player the basics on how to play the game.

The player will start on 150 hearts and 650 coins to begin with. Hearts can be regenerated with some monkey upgrades but usually these monkeys will cost a lot of money to do this.



Coins can be generated from popping enemy bloons – the harder the more money they will drop; completing a round; from certain monkey abilities and special abilities.

The faint green arrows indicate the entrance path the bloons come from and the direction they will take which we can assume to be only on the grey concrete.

The faint red arrows indicate where the bloons will exit, telling the player that they probably should not let any bloons get past this point otherwise their health will go down, resulting in them losing the game and progress.



The tutorial begins by telling the player to drag the tower Dart Monkey to the designated circle area.

Currently all other functions are locked, being greyed out, apart from dragging the dart monkey and clicking the settings button are available – all the other options seem to be available once the tutorial is complete.

It seems all other towers apart from the dart monkey are permanently locked and there must be a certain progression system needed to unlock them.



The tutorial uses a hand animation dragging the dart monkey to the desired area highlighted in golden implying that the dart monkey must be placed there by the player in order for the tutorial to be continued.

When clicked around I could only select the dart monkey and I tried placing the monkey in a different area to see what would happen. The monkey's area of view turned red, meaning that the monkey couldn't be placed in this area and when trying to place that monkey in a red area, the monkey would just hover over the area, not being able to be placed.

Placing the monkey in a desired area, the circle ring in terms of the tutorial allowed the dart monkeys vision to be clear, showing that the monkey can be placed in this area.



The circle ring seems to show the dart monkeys current area of view in which it can detect enemies. Enemies outside this area will not be seen by the monkey so will not be attacked unless they go into the circle view that the monkey can see.



After placing the first monkey in the desired area, the monkey guide then tells the player to place another monkey.

Although, this time, this monkey can drag and be placed anywhere on the map where the ground is grass as we can assume that this monkey can only attack on ground.

Boat or ship-based monkeys also seem to have this feature where they can only be placed on areas that have water such as the submarine monkey.

Also, when dragging around the monkeys the monkey would seem to flail around as they are being dragged across the screen.



Depending on where a monkey is placed, they can only see a certain amount of area. As seen in the photo the dart monkey can only see some part of the track as an obstacle is in the way, obscuring its vision of the path.

You can see each individual monkey's range by clicking on and off them and when not clicking on a monkey it will return you to a clear view of the map and the monkeys.

Other 3d visual animation is used such as when a monkey is idle, it will start playing with its dart, the chimneys will start emitting puffs of smoke and the sea will make a looping wave animation.



Clicking on the monkeys will allow a variety of features to be explored. Depending on the placement of where the monkey is where the monkey's profile will appear : if the monkey is clicked and is on the left side, its profile will appear on the right-hand side and vice versa.

Stats such as the amount of bloons the monkey has popped is available to the player. These stats will update in real time immediately after a monkey has popped a balloon.

The blue info bar will show short description information on what the upgrade will do to the specified tower.

The sell button will remove the monkey from the map and give back to the player 80% of the original cost so moving towers will cost the player.



As explained by the monkey guide, monkeys will gain experience by popping bloons which will allow them to unlock new upgrades that can expand the monkeys overall dps (damage per second) by upgrading their current damage and range in which they can see.

According to (Fandom B. , 2022) the amount of experience a monkey will gain is not also by the amount of pops they do but the cost of the monkey itself. Experience can be gained by completing each round. The more expensive the monkey is, the more experience the monkey will receive when a round ends. Experience can also be gained more quickly through harder difficulties that the player can choose or certain in game events. Cheaper towers will tend to have less experience to upgrade requirements whilst more expensive towers have a very large amount of experience pool needed to upgrade them since they are more powerful, especially in the later rounds in the game.

When a monkey has unlocked enough experience, the bars will turn green, indicating that an upgrade is available, whilst it will be red if the player does not have enough experience to purchase the permanent upgrade.

Clicking the upgrade button will lead the player to the upgrade menu of the specified monkey. Different monkeys will have different upgrades, up to 15 upgrades in total.

Each upgrade and “crosspath” as some bloons td players will say have a unique feature to them allowing each tower upgrade to make the same tower have a different unique version and use from each other despite being the same tower.

The upgrades leading from the sharp shots will allow the dart monkey to specialise in hitting multiple enemies at once whilst the long-range dart path changes the dart monkey into a more single target focused monkey, doing large amounts of damage but only to one balloon

As you go up the upgrades, the experience requirements also drastically increase; these requirements are different for every tower in the game.





Every tower can be upgraded 5 times in one path but can additionally be upgraded twice in another path to make a “unique crosspath”. For an example this dart monkey can be upgraded to a Ultra-Juggernaut and can additionally be upgraded twice in another path : this monkey can be either upgraded up to very quick shots or up to enhanced eyesight depending on the players choice anything after this upgrade will be greyed out and locked.

A towers “5th tier upgrade” is considered the towers most powerful upgrade and costs a substantial amount of money from popping the balloons, each upgrade path will have a unique 5th tier upgrade.



Each upgrade will have a different tower picture depending on the upgrade.

The lightning icon is an upgrade that will allow the tower to activate a special ability in the game after a certain amount of time has passed, this ability's use is infinite but takes time to recharge depending on how strong it is .

After buying the upgrade with the required experience, the upgrade for that monkey will be permanently available to be bought by the player but will still cost money to buy from popping enemy balloons .

Every upgrade must be upgraded in order, so the 5th upgrades can only be unlocked after unlocking the other 4 upgrades.

Each tower will specialise in different circumstances. They will have their own individual experience requirements and will gain different amount of experience.

Trying to purchase a new upgrade when you don't have enough experience will end up giving you an error saying that you do not have enough xp and gives the player feedback on how to earn the xp.

The XP bar will change in real game time as soon as the player spends the xp, the value in the bar will change to the total amount of xp left.





After purchasing the quick shot upgrade from the upgrade's menu, when clicking on the dart money, the upgrade is now available to buy with the cost of 100 coins.

Clicking on the button will result in your total coins reducing by 100 coins and the profile picture of the default dart monkey to update to the respected upgraded one.

Out of the five squares one is now highlighted in green to show that you have upgraded this monkey once in this upgrade path.



The old quick shots upgrade is now replaced with the next upgrade "very quick shots" and there is now red text on the upgrades as I do not currently have enough xp to continue upgrading the dart monkey.

I noticed there was a targeting system on most monkeys which had 4 options : strong, last, close and the default option, first. When setting a tower on first targeting, the tower will target the first enemy detected in the tower's current radius. On strong targeting, the tower will prioritise the strongest enemy first and ignore all other enemies in its radius; if there were only a group of enemies that were the same strength the tower would target the nearest enemy first and if a stronger enemy was to enter the towers radius, the tower would then retarget onto the stronger enemy. Close targeting is similar to first targeting but after the first balloon is further away than another enemy, it is ignored and the tower will refocus onto a different enemy. Finally, last targeting will quite evidently target the last enemy that has entered the current towers radius all enemies apart from the last enemy are ignored.



The tutorial then introduces and gives the player a free special tower, a hero called Quincy. Quincy is completely different from the other types of towers, whilst he had permanent upgrades like the other towers, there can only be one of him at a time. Quincy does not require the player to grind xp to level him up, rather he levels up by himself by a set amount every round. Whenever Quincy levels up there is a clear indication through a text "LEVEL UP" popping up next to him when this happens.





Like other hero's and monkeys, Quincy can only be deployed on a grass tile and not a water tile since he is not a water-based tower.

Heroes, like monkey, will get stronger by accumulating xp, , heroes will level up themselves by either the player upgrading the hero manually or the round ending, the xp gained will not be based on the enemies defeated, rather a set amount of xp is gained per round.

Clicking on Quincy will also cause the tower to emit out voice lines .

Each will have a short description about themselves and also will have a short description on what the next level will do to benefit the hero .

Upgrading the hero using money can vary on how much xp the hero has gained. In this instance, Quincy can be upgraded using 80 coins and is roughly 60% of the way through level 1 but if the bar was 90% of the way, the upgrade will cost less, 10 coins and if he was 0% of the way, the full upgrade would cost 200 coins.



Heroes can reach a "power spike" when unlocking abilities which in Quincy's case is level 3. This allows a small button in the left-hand corner with the appropriate ability button. As soon as the required level is met, the ability timer will count down as indicated by a small timer animation. When an ability is ready the picture will no longer have a timer and will be clear.

Abilities in general can stack up to whatever number of towers upgraded to meet the requirements, though, for heroes you can only get one of their abilities since you can only place one hero down and other towers you can place multiple.

Clicking on the button, when ready will cause the hero to activate their personal ability which will either enhance their own current stats or has a global game effect.

After using an ability, a set cooldown based on the tower's ability will happen, this will happen infinitely.

During some testing, I found abilities will not recharge while the game is in a paused state, it will only recharge once the player is currently engaging with a wave.



Completing a few round into the game allowed me to level up my account. It seems the player itself has an xp system that can be viewed in the menu.

This popup will appear, pausing the current game and dimming the screen when a player levels up.

At this stage of the game, the player levelling up will then give the player a choice to choose a single tower to have permanently.

Touching each tower plays a small spotlight animation behind the tower and will also align the tower to the middle of the screen.

It seems every time the player will level up from now on, the player will be able to pick a new tower and will eventually be able to collect all the towers.

Each tower will have their own name and a short description in what they specialise in doing. Interestingly here it does say the positives and negatives of using the bomb tower – it can hit many enemies but shoots very slowly.

The green unlock button we can presume to unlock the desired tower the player will choose and since there is no prompt or guide it seem the player has a free choice in whatever tower they want to pick.



In this phase only 1 out of the 5 towers can be picked. After picking a tower, the next time the player will level up, there will be 4 choices left. This will keep going until the last tower is picked. These current selection of towers are called “primary monkeys” and when they are all collected, a new set of monkeys such as the “military monkeys” will be unlockable by the player levelling up. This will continue until every monkey is unlocked in the game; when this happens, for every level the player receives will cause them to gain monkey cash (premium currency) the use of this will be explained later on.

I bought and unlocked the tack shooter which is now available to buy in game and is not locked like the other monkeys.

The tack shooter seems to be slightly more expensive than the dart monkey since the dart monkey is the cheapest tower in the game and the tack shooter being more powerful than the dart monkey in the later game.





The tack shooter has a completely different range, upgrade, abilities and cost of upgrades compared to the dart monkey since they are different towers.

It has a unique firing mechanic : it sprays a bunch of needles in a circle radius around it, allowing it to hit multiple enemies at once.

The upgrades further enhance this mechanic by increasing the number of needles fired, the speed at which they are fired or increasing the tack shooters global range.



Allowing enemies to reach the end of the path will cause the player to start losing lives which cannot be regenerated without a few upgrades : some monkeys can passively regenerate the players health through certain upgrades.

Different enemies will cause the player to lose a different amount of lives. In this gameplay, allowing a blue balloon to reach the end of the track will reduce the players life count by 2 and red balloons will reduce the players health by 1, this immediately updates the health counter to the correct number.

It seems all the balloons will take a follow the track in a general direction until the reach the end, where they disappear.

A set number of balloons, type of balloons and set speed of the balloons will be set each round.

Every balloon type and seem to go by the rainbow each balloon having unique hp and speed.



Completing a game of balloons tower defence 6 will give you the victory screen where you are given a set amount of monkey money and also a medal depending on the difficulty of the challenge. If you had previously done the challenge before, the rewards will be greatly reduced but still will be given out to the player.

Pressing continue will lead you back to the home screen page.



A daily reward system is implemented by Ninja Kiwi to keep the player coming back to the game at least once a day by giving the player small but beneficial rewards over time.

This keeps the player base active and motivates them to continue playing the game.

Trying to change your own personal time to trick the game into giving you infinite daily rewards will not work as it seems there is a built-in clock system registered to give the player rewards at a specific time – being a new day.



The achievement system and an extremely great number of missions that the player can do giving rather generous rewards when doing so.

Completing an achievement will be indicated in game when the player reaches a certain requirement by a small popup that an achievement has been completed by the player.

Once completed, the achievement will be highlighted gold and will give the specified rewards when collected by the player.

There is a search bar immediately filtering out specific achievements for the player and also has a progress bar to show the player into how far into the achievement they have completed.

Each achievement has a special and unique aim to them, and a short but concise description into how the specific achievement can be completed.

Hidden achievement are also there as an easter egg by hardcore bloons tower defence fans, that also give even more generous rewards but cannot be seen or known by the casual player.



Another shop system full of cosmetics can be bought with trophies which are unlockable through certain event daily or weekly.

These trophies modify the skins of balloons and monkeys and even can create special animations once bought and activated when enemies are defeated.



Looking at the home screen, there is another daily chest that gives the player small rewards every day, convincing the player to continue playing the game.

It seems there are also daily maps and events, though, this is locked behind level 20, however, this might be because they require a certain skill level to do so the game wants to make sure the player base is ready when doing this.

The players profile and level is now available to see through the homepage when clicking on the player icon.

The shop button will lead the player into another shop where premium currency can be bought with real life money, that will greatly progress the player in the game, these prices will vary based on the season a special events.



There is an accessible Jukebox that will player a variety of in game songs created by Ninja Kiwi themselves. A playlist can be created by the player that will cycle through the designated songs or if no songs are selected, a random song will be chosen each game.

The players profile will display a range of data : the players name, game played, the towers they have used the most, the heroes they have used the most, medals and many more. The player can choose through the togglable option whether they want their stats to be shown publicly to the world or privately (their stats can only be seen by themselves).



On the heroes tab, it shows all of the unlockable heroes and the current available ones you currently own. Certain heroes can only be unlocked once the player reaches a certain player level and then can be fully unlocked using the premium currency, monkey money which is earned from events, achievements and complete maps.

Each hero can be levelled up to a maximum of level 20 in game and usually is where it becomes its strongest form.

Skins can also be bought with monkey money, though they allow the hero to gain no beneficial increase in stats apart from cosmetics.



As of the 20th of June 2022, the game has 72 available maps for the player to complete, 18 maps for each difficulty. The difficulties ramp from beginner, intermediate, advanced and expert. Beginner maps will usually have the balloons take one very long route to the end of the map whilst expert usually have multiple routes the bloons take which have an extremely short path with further restriction also implemented making the challenge extremely difficult to complete.

At the start of the game, most of the beginner maps are unlocked and all other difficulties are locked. These can be unlocked once a certain number of maps from a previous difficulty have been completed.

Every map has 3 difficulties: easy, medium and hard, one of which must be chosen. Hard will give 75 more monkey money than medium and medium giving 50 more money than easy. Completing the set difficulty will give you the appropriate medal: easy = bronze medal, medium = silver medal and hard = gold medal, these medals will then be displayed and printed onto the player's profile permanently.

The change hero option will allow the player to switch between heroes as they can only select one in a game to choose, once chosen, the hero cannot be switched out and if so the map and current progress in the map would have to be reset.





In the settings, there are quite a few features :

- Sounds settings are togglable to whatever volume the user prefers. There are 3 types of sounds that can be modified : the sound of the towers shooting a projectile, the music and heroes voice lines/dragging sound of towers.
- Backup function allow user to save their data to the cloud where their current game and progress will be saved and then can be accessed from a different device.
- Screen size can be used to set the current game resolution and size of the game screen can be adjusted.
- Languages option will change the whole game language from the default option : English to whatever the user wants the game language to be in.
- Account will lead you to your current statistics of your game progress.
- Hotkeys can be used to easily change and place towers with a single button, for an example the button 1 can be assigned to select the dart monkey to be placed and can be reassigned to the button y if the player wanted it.
- The monkey button will show all the current available towers the player owns and how much xp the player has for that tower. Xp highlighted in green indicating that an upgrade is available whilst xp in grey indicates that an upgrade is not available.
- These powers are slightly different from abilities from monkeys or heroes. These powers are used to temporarily help the player in the game. Insta Monkeys will allow the player to instant place a specified monkey with a set upgrade path – these can be earned only through special events so there is a limited amount a player can have. Powers can be bought using monkey money, they usually help the player for a specific circumstance or for the entire round . Powers like the banana farmer will act as a regular monkey but will not attack and will only pick up bananas and monkey boost is a temporary boost for 20 seconds that increase all towers attack speed by a set amount.





Clicking about on the screen and experimenting on some maps, some objects are removeable by the player by using money, this will play a short animation of a monkey dismantling the object and after a few seconds the object will be removed, allowing the player to place towers in that area if the can fit. I also saw this mechanic being used on another water map where water can be frozen using an freezer, allowing the player to place ground-based monkeys on the ground where the water should have been.



There are 3 main difficulties in Bloons Tower Defence 6 : Easy, Medium and Hard. In easy mode, bloons are modified to be slower and the ending wave is set to round 40, this mode giving the least rewards while using the least effort and unlike hard and medium mode, all towers will cost less making the mode, well, extremely easy for the user. This mode has a limited number of special modes so is only recommended to new players. Medium mode is a mode designed for casual player : the mode makes the balloons normal speed and the monkey's normal price, there are slightly more new game modes in medium mode like apocalypse where balloons appear in large hoards. Hard mode is the most challenging being the hardest content in the game, CHIMPS being the hardest restricting the player of external use of powers and beneficial passive abilities, this game mode is a true test of skill for any hardcore bloons tower defence player as it extremely difficult to complete as a strategy is generally required.

Summary of Bloons Tower Defence 6

In conclusion, my playthrough with Bloons Tower Defence 6 was very worthwhile, as not only did the game give some useful insight to how generic tower defences worked but also introduced new and advanced features such as placeable objects or special powerups – unique features that other tower defence games have not previously done. The game for the current price of £1.60 was absolutely worth the cost. The tower defence was packed with content for the avid tower defence player base from ongoing weekly and daily events and monthly updates to keep the players coming back for more. Cosmetics gained from completing events made spending time on the events very worthwhile, as it gave a permanent additional feature such as skins for towers and enemies and even special animations. The ongoing development and constant updates also made the game always fresh to play, with the addition of new maps usually being added every so often, it was no surprise this game had a large fanbase and active players.

Pros:

- Has a lot of content to play around with – daily events – weekly events
- Just a great game – clean 3d animations – high quality graphics – massive amount of content
- Daily rewards keep players coming back
- Wide range of tower types allowing a vast amount of strategy – this also includes a wide range of upgrades for each individual tower.
- Wide range of maps – each maps allowing a different tower to shine in a different strategy
- Difficulty levels – new players and old players can choose their level of difficulty allowing them to both enjoy the game at their own pace
- A very casual game – you can play a map and leave – your data will be saved and you can continue your game later.
- Cosmetics give a very nice addition to the game – from changing what your enemy look like to your towers and even special animations once an enemy is defeated.
- Generous rewards – depending on the difficulty level the player completes, a nice amount of rewards are given, on top of the events and challenges.
- Music and sounds of the enemies being defeated are a nice addition – the music is professionally custom created by Ninja Kiwi and has a large range of soundtracks depending on the map. The music and sound can also toggled to the soundtrack the player wants.
- Upgrading towers cannot be purchased meaning the player must play the game in order to progress their towers – fully denying any players in paying to progress

Cons:

- Not a free game – Without the offers will cost £7.19 and even so, offers for the game only come around only during special occasions which is not very often.
- The extreme amount of content was quite overwhelming for a new player there was a multitude amount of upgrades, abilities for the player to try to understand and explore, it became confusing at times.
- Despite being a paid game, there is still a shop to buy large amounts of cash (used to purchase powers, skins)
- Xp system whilst isn't too bad on the less expensive towers gets really time consuming once you get onto upgrading the more powerful towers
- Heroes are pretty expensive to purchase

Success Criteria

In order for my game to be as successful as possible in this project, I must prioritise certain features over others in order to meet the deadline requirements. I have done this by following the acronym, MOSCOW.

Must – Features that are absolutely fundamental for the game to be the game itself.

Should - Features required to structure the game

Could - Features that are not necessarily needed but can be added if plausible and I have extra time.

Won't – Features that cannot be added due to the limited amount of time and the complexity level.

MUST Criteria

#	Feature	Description
M1	Tower	The tower will be able to be drawn on the canvas in the middle of the screen and will automatically shoot enemies within a certain range. It will prioritise the nearest enemy first and cannot be controlled by the player and can only be modified through upgrades.
M2	Enemies	A set amount of enemies will be randomly generated throughout the canvas and will make their way towards the tower with a set health, damage and speed. Once colliding with the tower, the enemies will keep colliding with the tower until the tower has lost all of its health.
M3	Health system	At the start of the game health will be set to a default integer and can be increased or decreased. The amount of health the player has will determine the state of the game as when the player's health reaches < 0 then the game will end whilst if the player still has health, the game will continue. The amount of health will be visually indicated by a health bar, that will decrease based on the amount of health the player has left. Health will decrease by a set amount once an enemy has collided with the tower, which will deal a set amount of damage. Some upgrades may increase the health of the player or even regenerate some of the player's health by a set amount every x seconds.
M4	Menu	An interface that the player will use to start the game when they want to start playing the game, this may also include some interactive features such as a settings icon or some objects that follow the cursor.

SHOULD Criteria

S1	Tower radius	The tower will shoot and only shoot an enemy if it is within the radius of the tower, otherwise the tower will be idle for that period of time.
S2	Simple Enemy Ai	Enemies will make their way towards the tower (the middle of the screen) by locating where the tower is and what direction it has to take in order to get to the tower. It will eventually make its way to the tower and it by colliding with it

S3	Difficulty	Enemies will start off very weak, being able to be instantly defeated by 1 shot from the tower. They will gradually get stronger every x amount of enemies that have been defeated. Enemies health and speed will increase by a set amount in order to balance out the player upgrading their tower.
S4	Enemy Spawn	Every time an enemy is drawn, there will be a check that will make sure they will be spaced apart from each other and they will all make their way towards the tower. If I were to add waves, enemies could have the ability to spawn in herds in order to be challenging for the player since they will need specific upgrades to counter this.
S5	Projectile	This will be produced by the tower once an enemy is in range of the tower. When an enemy is in range, a projectile will appear from the tower's cannon and move in the direction of the nearest enemy within the towers range at a certain speed. Once a projectile hits an enemy, it will do x amount of damage to the enemy. The bullet will disappear as soon as soon as it hits an enemy.
S6	Home	The home screen button will allow the player to redirect back to the menu in case the player wanted a way to reset their way quickly.
S7	Simple Upgrades	By earning coins dropped by enemies, the player can upgrade their towers stats : health, damage, and attack speed. I will modify it so their will be a cap to how many times an upgrade can happen through the use of coloured bars. Upgrades will exponentially increase in price the more they are upgraded.
S8	Money	Money will be dropped by enemies based on how difficult they were to defeat. This currency is the main way the player will upgrade the tower in order to progress.

COULD Criteria

C1	Waves	The number of enemies appear and type of enemies will be based on the wave. The higher the wave, the more enemies that will spawn and the stronger certain enemies will be . Every 10 waves a special boss will spawn which will have a large amount of health and the wave will only end once this boss is defeated.
C2	Types Of Enemy	There will be different types of enemies that will have different amounts of health and speed and traits – the enemies health and speed will increase the more rounds the player completes
C3	Settings	The settings icon will allow the player to exit the game or toggle the music on or off.
C4	Sound/Music	Music will be able to be turned on or off depending on the user's preference . Sounds will be emitted from the tower shooting a bullet, the player pressing the shop icon and when the player purchases an upgrade.
C5	Particles	If the tower defeats an enemy the enemy will explode and emit small particles around where it was destroyed – this will vary based on the type of enemies.
C6	Powerups	A temporary buff to the tower that will be able to be triggered by the player every x amount of seconds – it will usually have a long cool down to make sure the buff are used strategically.
C7	Additional Upgrades	Some interesting upgrades could be added, such as the addition of the tower to shoot multiple bullets at once, allowing bullets to pass through multiple enemies, and increasing the towers attack speed. I could also add an option for defence, to

		reduce the enemies damage by a set % although this may need a lot of testing to balance this upgrade out.
C8	Shop/Shop UI	The shop will be indicated by a shopping trolley icon that the player can access whilst in game. It will have permanent and temporary upgrades the player can collect in their current game. When a player doesn't have enough money, the option to buy that item will be greyed out and the player will be unable to purchase this until they fit the required money count. Whilst the player is in the shop, the game will be paused until the player exits out the shop.
C9	Advanced Enemy AI	Elite enemies will have special movement unlike normal enemies, they will move in a curved way in which it will be difficult for the tower to shoot unless the tower has enough attack speed.
C10	Statistics	A statistics button will hold all the players stats from every game they have played, this includes information on how many of enemies, bosses, waves and money the player has achieved.
C11	Leader board	A scoring system that will save the players data – their score and name on the server. The interface will be accessible from the menu and will display the top 3 scores with the users who have achieved this score.
C12	Losing Screen	When the player has no more health, then the game will end halting the movement of the enemies and the tower. During this time, a screen presumably have the text "You have Died" will appear and further options to allow the player to play the game again or quit the game.

WON'T Criteria

W1	Multiplayer	If I were to make the tower defence a multiplayer, I would have to research on how to make a client-based server in order to make sure both players can have separate upgrades from each other and shops. Towers will be separated from a set distance from each other and enemies will have double their stats from usual, since there will be another player involved and so a lot of balancing may be required, this may be difficult to balance so I may consider this in a future development of my game.
W2	Transactions	Dealing with legal issues would be a massive problem in terms of a small game like this therefore in this version on my game, this will not be required.
W3	Cosmetics	Cosmetics such as skins for the enemies and tower and custom particle effects could be implemented as a nice quality of life change to promote players to continue to play the game, however, I feel this will take too much time to create custom cosmetics, so I would have to implement this in a future release if I were to continue developing the game.
W4	Login	A login system would save the statistics and progress of the player allowing them to access their progress from any device and making so they won't have to reset their progress. Whilst this would be a nice feature to add, I do not think it will be

		absolutely necessary for this version but could be used in a future development of the game if it were to go global.
--	--	--

Software/Hardware Requirements

My game will be created in the programming language of JavaScript since I have the most experience in this language and it is suitable language for making a single player web-based game.

I will develop my game using the online IDE, Replit, as I can access the files for my game in both college and home as long as I have a stable internet connection and it discards the use of any specialist hardware needed. I will use the p5 library to draw my home screen and other objects and also use certain p5 functions to deal with the physics side to the game.

Players of the game will not be required any specialist hardware or software to play the game other than updating their browser most recent JavaScript version running.

Prototype Development/Technical research

1.Entity Collision

In this part of the program I want to create a prototype on the collision detection between the tower and an enemy where the enemy will spawn outside the map and gradually make its way towards the tower. Once the enemies touch the tower, the tower will take a set amount of damage reducing the health bar based on the health left. Whether an enemy enter the towers range, I intend to make it so the tower will automatically run a fire command which will prioritise and target the nearest enemy, this will have a set cooldown when firing in order to balance this out.

1.1 Creating the player

When dealing with entity collision, I will first need to create a player. In this case, since the game is a tower defence, I will first need to create a tower. I started by researching on how to create an object and figured I would need to use the p5 library in order to handle drawing objects onto canvas and also draw all the entities I would need, so I loaded the script to allow p5 to work into my html file.

```
<html>

<head>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.1.13/p5.js" type="text/javascript">
  </script>
  <script src="game.js" type="text/javascript">
  </script>
  </script>
</head>
```

</html>

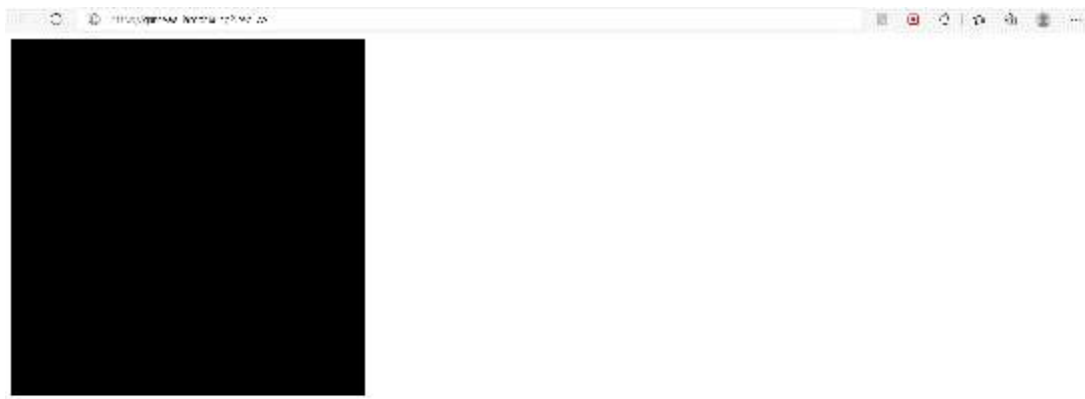
I also created a game.js to hold where the map of the game would be and tower.js file to setup a draw function so the tower could be drawn onto the canvas.

When creating a basic player, I first needed to understand how to draw a simple circle. To do this I looked on the p5 library, which showed to create a circle I would first have to create a canvas in the game.js file, so I tried creating one with the help of w3schools

```
function setup() {  
  createCanvas(500, 500);  
}
```

Now that I had created a canvas, I wanted to test it out by simply drawing/colouring in the background to do this I would have to use the draw() function and then set the background to a designated colour, in this case, I chose black.

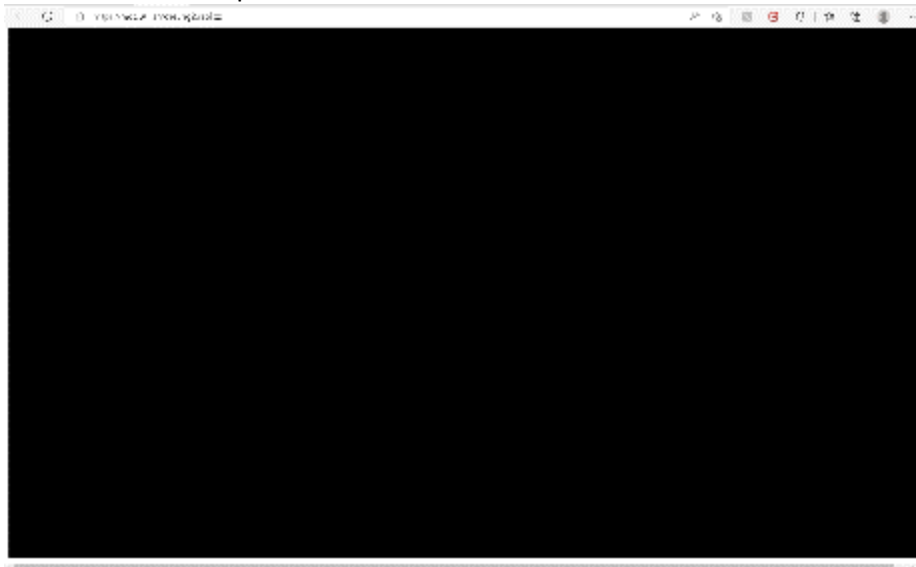
```
function draw() {  
  background("black");  
}
```



Whilst it did colour in the background, it only coloured the canvas by a width and height of 500, so I need a way for the canvas to cover the whole screen at once. To do this, I modified the previous createCanvas() code and set the parameters to the windowWidth and the windowHeight.

```
function setup() {  
  createCanvas(windowWidth, windowHeight);  
}
```

This seemed to fix the previous issue and now produced the result of filling in the entire background rather than a small portion in black:



After showing the background for the main game, Jayden suggested that this background is rather plain and could be improved through an animated background but also considered that due to this being a single stakeholder developing the game rather than a group, he suggested this background would be ideal to keep the simplicity of the game and further animations or adjustments could be made in a later version in the development of the program.

I then wanted to create and draw a tower onto this canvas, so I would have to use the class function to help me create it. I created a class function called tower where the x and y coordinate and the colour of the tower will be parsed in through the setup() function in the game.js file.

```
class Tower {  
  constructor(x, y, c) {  
    this.x = x;  
    this.y = y;  
    this.colour = c;  
  }  
}
```

After that I would have to actually draw the tower, and so in the Tower class, I implemented a draw function. I wanted to make the tower in the middle of the screen so I would have enough space for enemies to have time to spawn. To find the width and height of the centre of the screen, I would just have to divide the canvas's width and height by 2.

```
draw(){  
  circle(windowWidth/2, windowHeight/2 );  
}
```

This didn't end up drawing the circle and there was no console error so I went back to w3schools to investigate. It turns out I needed to use the push() function and declare a tower variable to allow a circle to be drawn on the canvas.

```

var towers = [];

function setup() {
  createCanvas(windowWidth, windowHeight);
  towers.push(new Tower(windowWidth, windowHeight, "lightblue"));
}

function draw() {
  background("black");
}

```

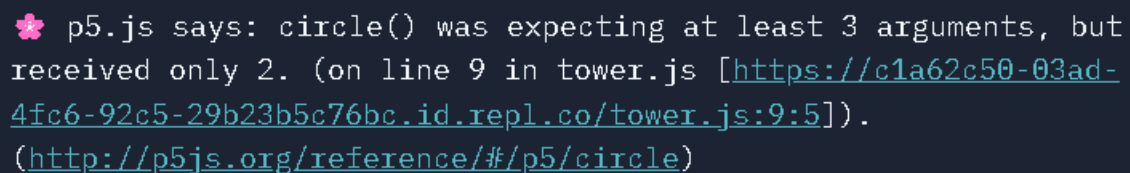
However this still wouldn't draw the circle onto the map. I continued to look at the draw() function on p5 and saw that a for loop must be used to be able to run the draw command in the tower.js file.

```

function draw() {
  background("black");
  for (t of towers) {
    t.draw();
  }
}

```

Even though this did seem to work, I still received an error on the console :



```

❁ p5.js says: circle() was expecting at least 3 arguments, but
received only 2. (on line 9 in tower.js [https://c1a62c50-03ad-
4fc6-92c5-29b23b5c76bc.id.repl.co/tower.js:9:5]).
(http://p5js.org/reference/#/p5/circle)

```

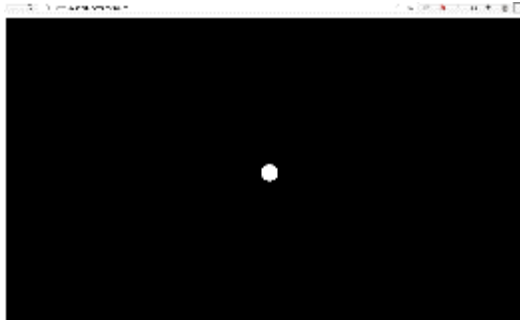
I realised when creating the circle, an x coordinate, y coordinate and diameter must be required when drawing the circle, so I updated the draw() function in the tower.js file.

```

draw(){
  circle(windowWidth/2, windowHeight/2, 50);
}
}

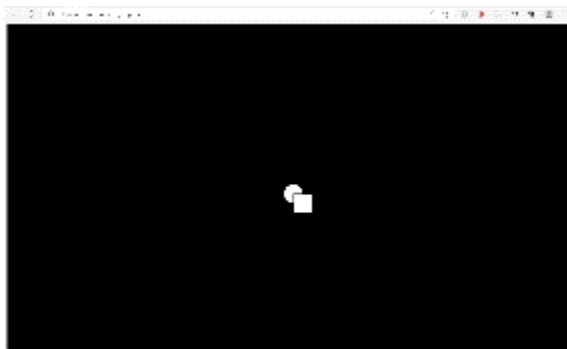
```

Which produced a small white circle in the centre of the screen :



I also wanted to add a small barrel on the end of the circle to make it easier for the player to know where the bullets will appear from the tower. To do this, I needed to add the `rect()` function to the `draw` function in order to draw a rectangle, I then would have to iterate through the rectangles coordinates until I found the perfect angle in which the barrel would look suitable.

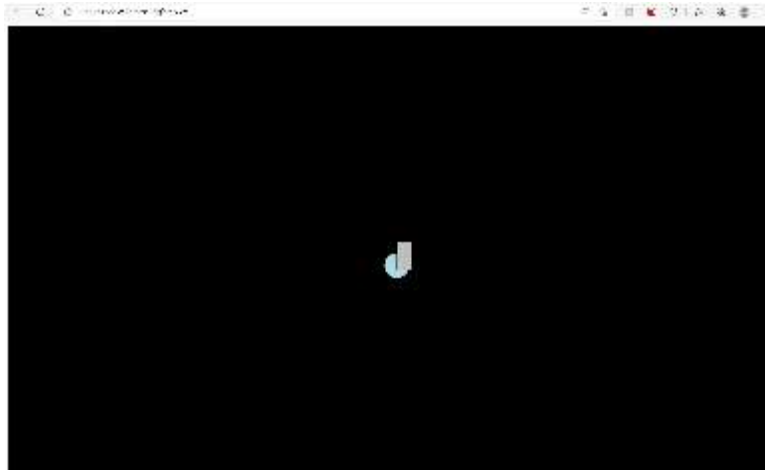
```
draw(){  
  circle(windowWidth/2, windowHeight/2, 50 );  
  rect(windowWidth/2, windowHeight/2, 50)  
  
}
```



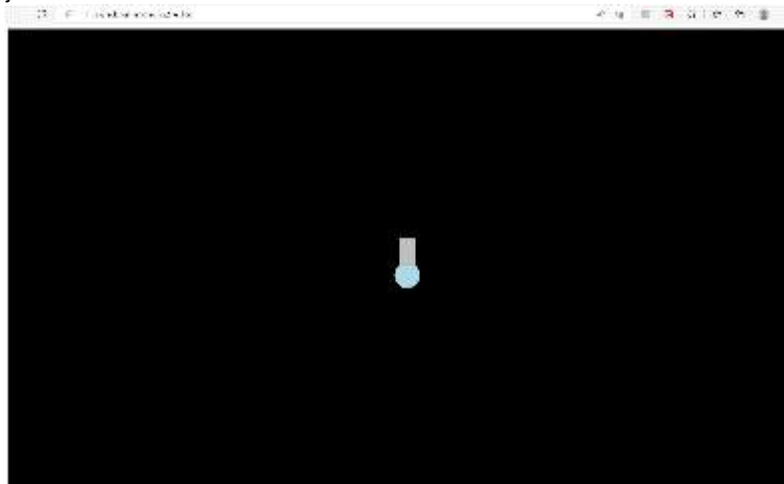
This ended up creating a small square that was slightly overlapping on the circle.

```
draw(){  
  
  fill(this.colour)  
  circle(windowWidth/2, windowHeight/2 + 20, 50 );  
  rect(windowWidth/2, windowHeight/2 , 30, 55)  
  
}
```

I then used the `fill (this.colour)` function to parse in the specified colour in the `towers.push` function and modified the circle and rectangle to an appropriate width for the tower though, the rectangle still wasn't in the centre of the circle like how I wanted it.



```
draw(){
    fill(this.colour)
    rectMode(CENTER)
    circle(windowWidth/2, windowHeight/2 + 20, 50 );
    fill("silver")
    rect(windowWidth/2, windowHeight/2 , 30, 55)
}
}
```



Adding rectMode(CENTER) allowed the rectangle to be positioned in the middle of the circle allowing it to look like a tower now.

I also added the fill("silver") before rect() in order to colour in the rectangle.

1.2 Making the player radius

The next thing I wanted to add was the radius of where the towers could see the enemy and so, I did the same thing as my player class, but this time I would have to draw only the outline of the circle and set the inside of the circle to be the same colour as the background

```
class TowerRange{
    constructor(x, y) {
        this.x = x;
        this.y = y;
    }
}
```

I created another class called TowerRange and did mostly the exact same as I did with the tower class.

After some testing, using stroke() would allow me to highlight the outline of the circle, and using fill() will fill in the circle in which I set to black since it was the

```

}

draw(){

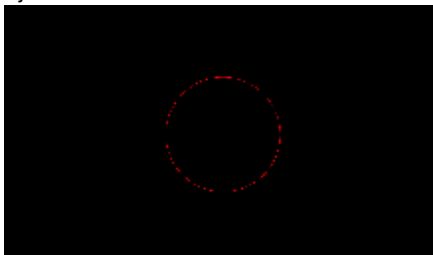
  stroke("red")
  strokeWeight(2)
  fill("black")
  circle(windowWidth/2, windowHeight/2, 400 );
  noStroke()
}
}

```

```

function draw() {
  background("black");
  for (t of towers) {
    t.draw();
  }
  for (r of towersRange)
    r.draw()
}
}

```



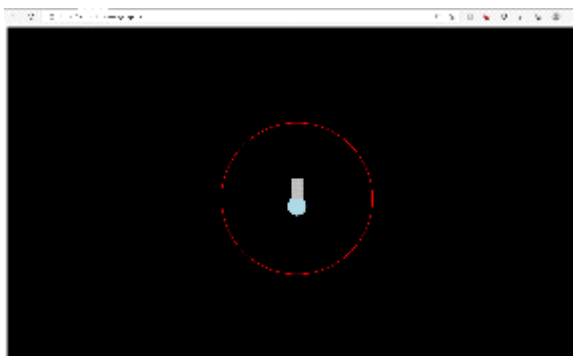
However, this would end up creating and drawing the circle radius over the tower, blocking it from view, leaving only the red circle radius in view on the screen.

```

function draw() {
  background("black");
  for (r of towersRange)
    r.draw()
  for (t of towers) {
    t.draw();
  }
}
}

```

To fix this, I had to draw the TowerRadius before the Tower in order to make the tower be drawn over the tower radius so the tower could be seen by the player.



1.3 Making Enemies

Now, I needed to create some enemies that can be defeated by the tower, to do this I will have to use similar code to the tower and tower radius class in order to draw a set amount of enemies on the canvas. When I have created some enemies, once an enemy is in the towers range, the tower will run a fire() function which will prioritise attacking the nearest enemy in the towers range.

```
class Enemy {
  constructor(x, y) {
    this.x = x;
    this.y = y;
  }

  draw() {
    fill("red")
    rect(Math.random() * windowWidth, Math.random() * windowHeight, 50)
  }
}

function setup() {
  createCanvas(windowWidth, windowHeight);
  rectMode(CENTER);
  towers.push(new Tower(windowWidth, windowHeight, "lightblue"));
  towersRange.push(new TowerRange(windowWidth, windowHeight));
  enemys.push(new Enemy(windowWidth, windowHeight));
}
```

I created a separate file name enemy.js to make accessing the code more easier. I then, once again used the class function with the same variables as I used in the player function to simplify the process, but I then realised a few conditions had to be set for the game : a set number of enemies were to be drawn on the canvas and that they had to be randomly drawn on the board.

To attempt to do this, I tried using the Math.random() function. By multiplying the Math.random function by the canvas width and height, I would be able to draw the enemies within a random position on the canvas. I also used the fill function to colour in the enemies in a default colour enemies are, red.

```
function draw() {
  background("black");
  for (r of towersRange) {
    r.draw()
  }
  for (t of towers) {
    t.draw();
  }
  for (e of enemys) {
    e.draw();
  }
}

function spawn() {
  for (let i = 0; i < 20; i++) {
    enemys.push(new Enemy())
  }
}
```

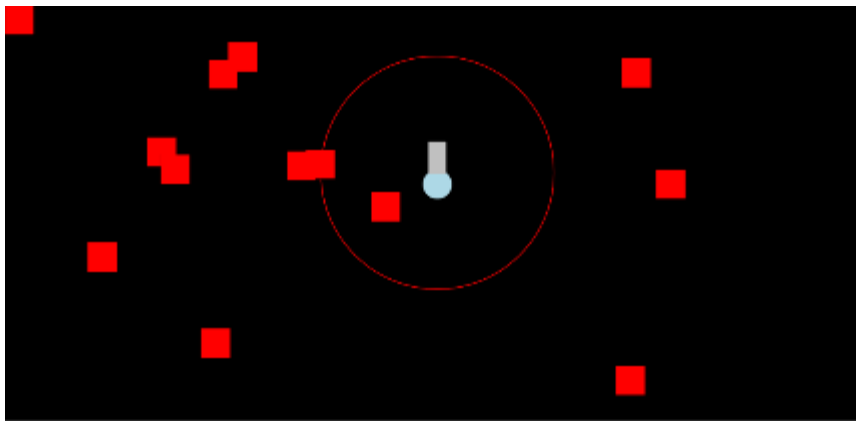
After that, I used the same process as I used in creating the towers by creating a for loop to draw the enemy and enemys.push to add a new enemy to the enemy array producing:



```
enemys is not defined
    at setup (https://c1a62c50-03ad-4fc6-92c5-29b23b5c76bc.id.re
    at p5._setup (https://cdnjs.cloudflare.com/ajax/libs/p5.js/1
    at p5._start (https://cdnjs.cloudflare.com/ajax/libs/p5.js/1
    at new p5 (https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.1.
    at _globalInit (https://cdnjs.cloudflare.com/ajax/libs/p5.js
```

I did not declare the variable of enemys so the enemys.push function had nothing to add to a non-existent array. So, I had to declare a new variable enemys to fix this :

```
var enemys = [];
```

The output listed out a group of problems : a single enemy was created every few milliseconds and very rapidly, however, this was not what I wanted, as the enemies were drawn on the canvas, there was only really one enemy being drawn but multiple times in quick succession : once an enemy was drawn, they were immediately replaced by a newly drawn enemy and were not actually being drawn but replaced instead. Furthermore, enemies could spawn within the tower's radius, on the tower which is not what I wanted and they could also overlap on each other being an issue. Also, one enemy was created whilst I wanted a set amount of enemies to be created.

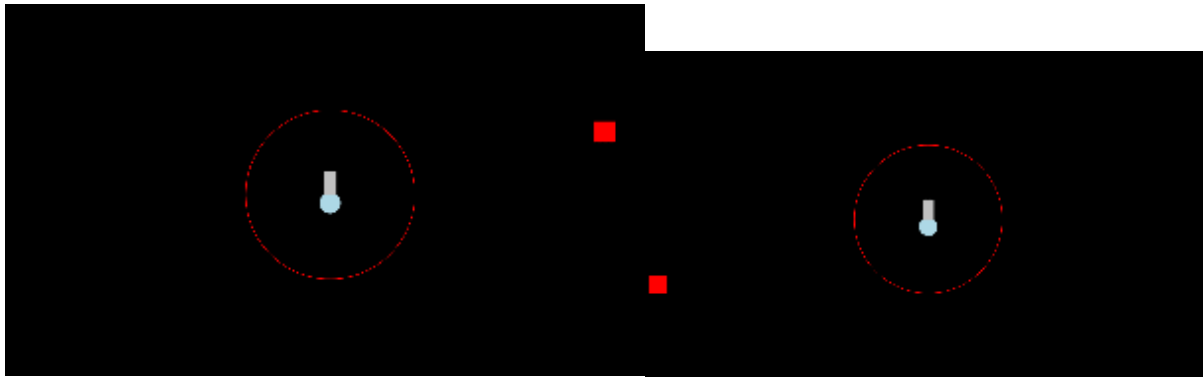
First, I had to try to draw an enemy on a random position on the canvas before trying to make it move.

```
class Enemy {
  constructor(x, y) {
    this.x = Math.random() * windowWidth;
    this.y = Math.random() * windowHeight;
  }

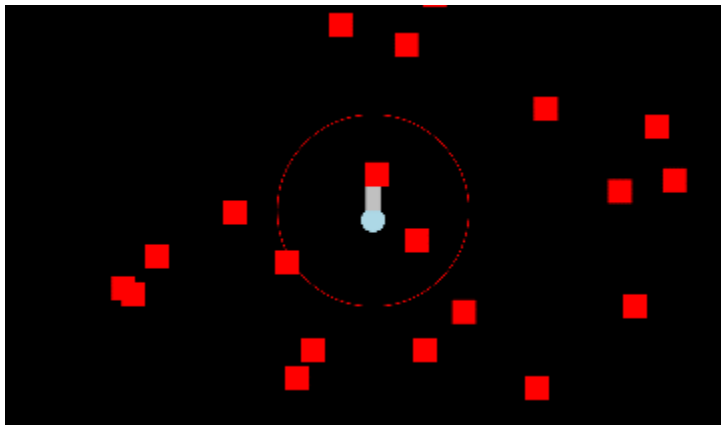
  draw() {
    fill("red")
    rect(this.x, this.y, 50)
  }
}
```

I replaced this.x and this.y to the random coordinate function so that the enemy would actually be drawn on the canvas rather than be replaced every few seconds.

This produced a single enemy being drawn randomly on the canvas that wouldn't be replaced, however, I still wanted multiple enemies to be drawn rather than just a single enemy.



```
function spawn() {
  for (let i = 0; i < 20; i++) {
    enemys.push(new Enemy(windowWidth, windowHeight,))
  }
}
```



I created a spawn function using a for loop to iterate through the enemy array until 20 enemies were drawn throughout the canvas. Whilst this did create the enemies as intended I realised a few problems : Firstly, enemies could be able to be drawn within the tower's radius, on the tower itself and on other enemies. Secondly I would need to calculate a way to allow the enemies to make their way towards the player and then make a function to allow the tower to detect these enemies but only when the enemies are in the towers range.

After some testing, Jayden suggested the enemies could have some other features such as a set sprite apart from being just filled red. This would be done to make the game seem more appealing and unique to play since "a red square seems quite boring and lack of character" , I may consider adding these features in the final development of the game when I have finished prototyping the enemies.

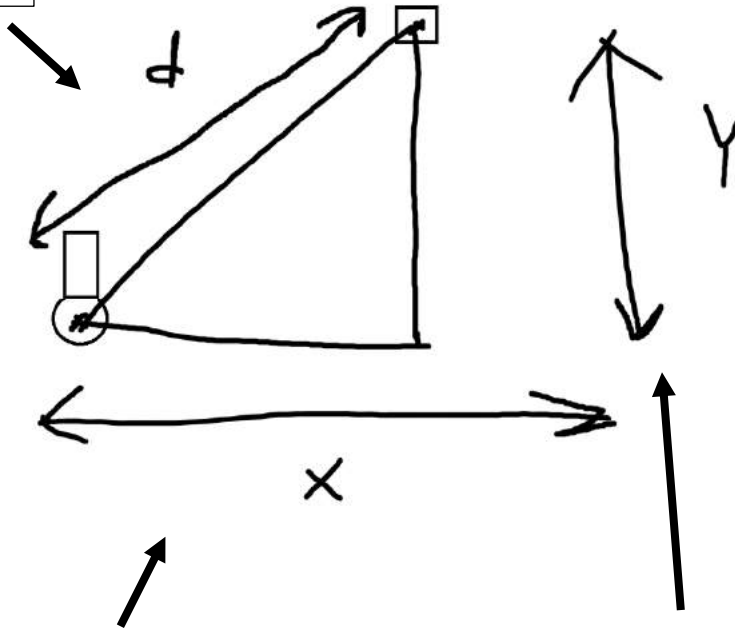
1.4 Enemy Ai

Currently, enemies would be drawn randomly on the board but would not move since their coordinates weren't updated. I wanted to make the enemy make their way from the outside of the map towards the tower based on their position, and, once colliding with the tower, will continue colliding with tower until the tower has no more health thus, resulting in a game over screen. However, I had trouble understanding on how to find the distance between the tower and the enemy and then applying this distance to update the enemy's position until it meets the tower.

Research – Pythagoras Theorem

Firstly, in the game, every enemy will have a different coordinate and position from each other, so will each have a unique distance and angle they will have to move in order to get to the tower. Currently, to get the x and the y distance from the enemy to the tower , I would have to subtract the towers coordinates from the enemy's coordinates; this will give me the x and y distance but will not give me the angle and actual distance from the enemy to the tower. After reading (Pythagorean Theorem, 2022) from Wikipedia I figured out using this equation would help me since it would find the distance and the

Unknown distance can be calculated using Pythagorus.



To visualise this more clearly, I drew a quick sketch of the enemy and the tower and how this equation can be used in this instance. I need to find the distance from the enemy and the tower which is currently unknown. This is modelled by the letter d and I currently have the distance of x and y , so, to find the distance, I would have to square the x distance of the enemy and tower, and then would need to square root this answer since it would be the distance squared, to find the actual distance needed for each individual enemy to move. I can then use this value to find the angle the enemy would need to use to reach the tower.

Stored in the enemy.js -
Dist.x variable that contains the x distance between the enemy and tower

Stored in the enemy.js - Dist.y variable that contains the y distance between the enemy and tower

```
class Enemy {  
  constructor(x, y) {  
    this.x = Math.random() * windowWidth ;  
    this.y = Math.random() * windowHeight ;  
    this.speed = 1  
  }  
}
```

First, I added a new variable to the enemy class, speed using this.speed and set it to 1. This will determine how quickly the enemy will move when going towards the tower – I will modify the speed of the enemy later if I were to add waves so the enemy speed would be based on the wave the player is on.

Then I needed to find the X and Y distance between the tower and the enemy so I declared the variable Dist.X which will hold the X distance between the objects and DistY which holds the Y distance between the objects. I subtracted $windowWidth/2$ (the current position

```

class Enemy {
  constructor(x, y) {
    this.x = Math.random() * windowWidth ;
    this.y = Math.random() * windowHeight ;
    this.speed = 1
    this.DistX = windowWidth / 2 - this.x;
    this.DistY = windowHeight / 2 - this.y;
    //finds the distance between tower and enemy
    this.distance = Math.sqrt((this.DistX*this.DistX) + (this.DistY*this.DistY));
    //find angle between enemy and player
    this.angle = Math.atan2(this.DistX, this.DistY)
  }

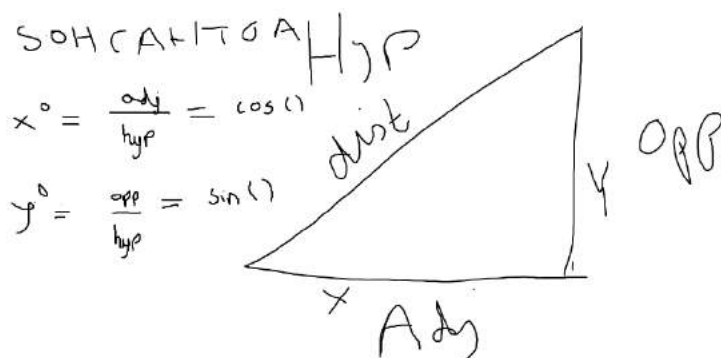
  draw() {

    //finds what angle the enemy needs to move by x degrees
    this.speedX = Math.cos(this.angle);
    //
    this.speedY = Math.sin(this.angle);

    //moves enemy's x coordinate based on the speed
    this.x += this.speedX * this.speed
    //moves enemy's y coordinate based on the speed
    this.y += this.speedY * this.speed

    fill("red")
    rect(this.x, this.y, 50)
  }
}

```

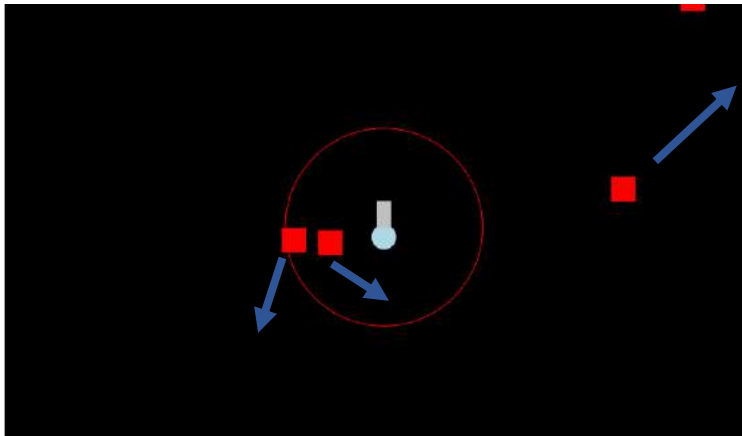


To find the distance I had to apply the Pythagorean Theorem equation, so I squared the x distance by the x distance and the y distance by the y distance and then square rooted(using the function math function : Math.sqrt) to find the distance between the tower and enemies.

I assigned this.angle to store the angle the enemy will need to follow to get to the tower. Using Math.atan2 will return the angle from the given (x,y) coordinates so I tried to use this.

Now I had the angle the enemy needed to follow, I would have to find out the x and y angle the enemy would need to follow, to do this, It would be necessary to use trigonometry.

Using the acronym : SOHCAHTOA I figured out to find the X angle, I would need to use the cos() function and sin to find the y angle



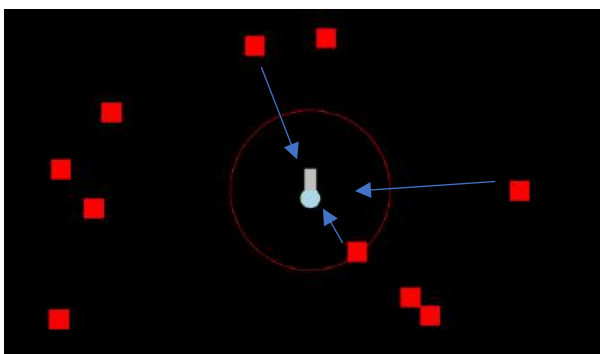
The enemies would now move in a certain direction, but rather than move towards the tower would move in a random direction.

```
class Enemy {
  constructor(x, y) {
    this.x = Math.random() * windowWidth ;
    this.y = Math.random() * windowHeight ;
    this.speed = 1
    this.DistX = windowWidth / 2 - this.x;
    this.DistY = windowHeight / 2 - this.y;
    //finds the distance between tower and enemy
    this.distance = Math.sqrt((this.DistX*this.DistX) + (this.DistY*this.DistY));
    //find angle between enemy and player
    this.angle = Math.atan2(this.DistX, this.DistY)
  }
}
```

When looking back on my code I looked over the math.atan2 function to see if this was the reason to why the enemies were moving the wrong direction. Reading (contributors, n.d.) showed when using Math.atan2, the coordinates used is in the order (y,x) rather than the order I used, (x,y)

```
this.angle = Math.atan2(this.DistY, this.DistX)
```

Therefore, I changed the Math.atan 2 to the y,x coordinates.



Now, enemies would correctly make their way towards the tower.

However, the enemies would then overlap over each other and also pass through the tower rather than attacking the tower which is what I wanted.

1.5 Enemy Spawn Overlap

Currently, enemies would spawn randomly in the canvas when drawn in, this would sometimes end up randomly allowing the enemies to be drawn on top of each other, making this a problem that needed to be solved. I wasn't completely confident that I could do this so I watched a few videos explaining how and what must be done in order for objects to not be drawn overlapping each other. One particular video was extremely helpful into my approach to this problem: (Train, 2016) demonstrated that using p5's built in dist function could find the distance between objects alongside using a flag being set false and a while loop to be used to check if the objects were intersecting with each other. If the circle is intersecting with another circle, the flag isColliding would be set true and he would use break to stop the object being drawn and breaking out of the loop whilst if the circle wasn't intersecting, if isColliding is false, the circle would be drawn in a free space.

```
class Enemy {  
  constructor(x, y) {  
    this.x = Math.random() * windowWidth;  
    this.y = Math.random() * windowHeight;  
  
    this.x = x  
    this.y = y
```

Firstly, I needed to be able to check if a new enemy that is going to be drawn would be drawn over another existing enemy – if the enemy is to be drawn over an existing enemy, it cannot be drawn, but if it can be drawn the enemy would be drawn at that location, to do this I would need access the enemy location in both the game.js file and the enemy file by parsing in the values x and y but currently, they weren't being read since they weren't used so I changed them to the values x and y.

```
function spawn() {  
  //loops through array until 20 enemies are created  
  for (let i = 0; i < 20; i++) {  
    // assumes enemy is not colliding  
    colliding = false
```

Then, I created a flag in the spawn function in the game.js file: "colliding" to false to assume that enemies created aren't currently colliding and would reset the flag to false for each enemy being created.

```
// generate a random x coordinate  
xco = Math.random() * windowWidth  
// generate a random y coordinate  
yco = Math.random() * windowHeight
```

After that, using the previous Math.random() code, I stored the values in a new variable xco, storing the enemies x coordinate and yco which would store the enemies y coordinate, this would be in a for loop which in this case would continue looping until 20 enemies were drawn in the canvas.

To keep checking if an enemy would be the same distance from another enemy, another loop would need to be used.

```
// loop through existing enemy list to see if current enemy being drawn is too close to existing enemy
for (let j = 0; j < enemys.length; j++) {
  // finds the distance of enemy that is going to be drawn with existing ones
  var d = dist(xco, yco, enemys[j].x, enemys[j].y)
```

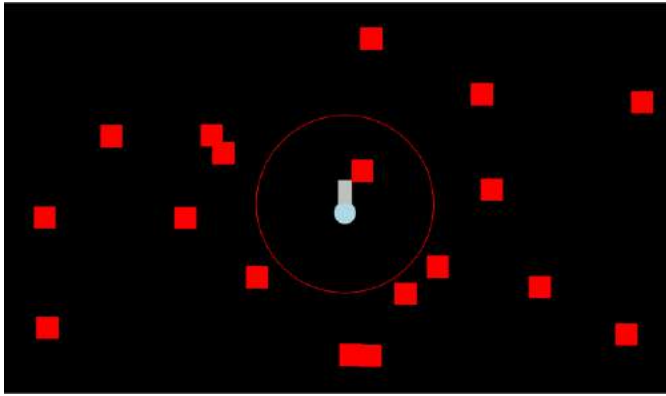
Using p5's dist function used 4 values to find the distance between the enemies and existing enemies. Xco,yco contained the x and y coordinates of the enemy that are going to be created and enemys[j].x , enemys[j].y hold existing enemies x and y coordinates. I created the variable d to hold the results of the calculation of the distance between the new enemy and existing enemy as this will be needed later.

Enemies would be drawn too close to each other, so I wanted them to be spaced apart from a set distance. Utilising the if condition - if the distance between the enemy that was going to be drawn to an existing enemy was within a distance of 50 or less, then the enemy would not be drawn, this would set the colliding flag to true and so would not draw an enemy in that location, so the loop would then generate a new x and y coordinate and compare it until a new enemy has a distance that is greater than 50 of each other.

```
// if enemy is too close - within a distance of 50 of each other don't create
if (d < 50)
  // sets colliding as true so enemy cannot be drawn there
  colliding = true
}
```

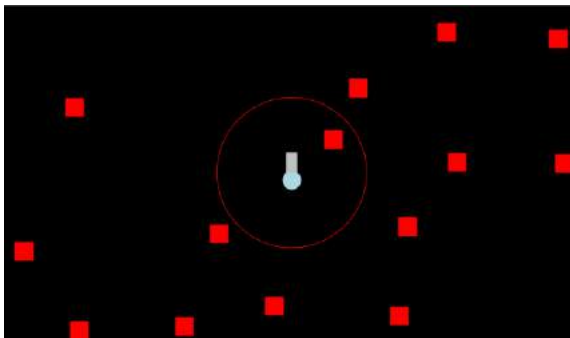
Another if statement is required if the new enemy's distance between an existing enemy is greater than 50 then the enemy can be drawn on the canvas through the coordinates that fit this criteria.

```
// if enemy is not colliding, then create enemy in that position
if(!colliding){
  enemys.push(new Enemy(xco, yco))
}
}
```



As a result of adding this code, the enemies would spawn apart from each other, though, I found on certain occasion through repeated testing sometimes the enemies would slightly overlap over each other when spawned, this is because my enemy had a size of 50 when drawn on the canvas and the minimum distance an enemy had to be from each other was 50 so would result in the enemies slightly overlapping over each other.

```
if (d < 150)
```



Therefore to fix this, I increased the minimum distance enemies had to be when drawn on the canvas.

Making enemies now be drawn far apart from each other and prevent them from being drawn over each other in the spawn function. They still, however, would eventually overlap over each other when making their way towards the tower and would overlap over the tower and so, I would need to detect the collision of enemies on the tower.

1.6 Enemy Overlapping

To allow enemies to not overlap each other when moving towards another enemy, I would have to deal with entity collision – checking if an enemy is inside or touching another enemy. To achieve this, using similar code in 1.5 when I tried making the enemy move towards the tower but instead checking the enemies x and y with other existing enemies coordinate to see if they are overlapping, if so then collision has occurred, and so will stop moving. I will first try making a prototype of this so when an enemy touches another enemy, they will stop moving, indicating that collision has occurred. I will not have to worry about the enemies colliding with each other they are travelling towards the tower since they all spaced apart from 1.5 and are moving at the same speed though I may have to adjust this if I were to making enemies with different speed.

```
// assumes enemy is not colliding
let colliding = false

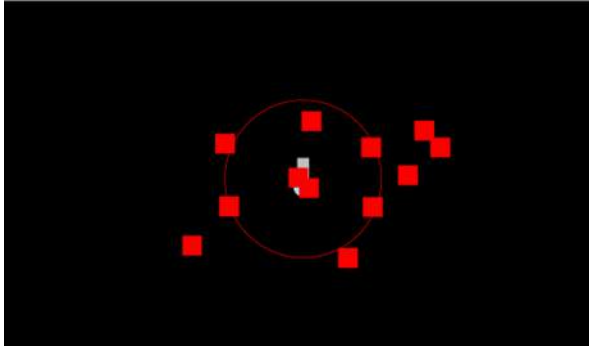
// loop through existing enemy array to see if current enemy is
too close to existing enemy
for (let j = 0; j < enemys.length; j++) {
  if (this !== enemys[j]) {
```

Firstly, we have to assume the enemy is not currently colliding with another enemy until it is checked, if it is colliding with another enemy, I can set the colliding flag to true and then make the enemy stop moving once this condition has been met.

Using a for loop will go through all the enemies in the enemy array and allow me to compare each enemy with each other to see if they are colliding.


```
var d = dist(this.x, this.y,
enemys[j].x, enemys[j].y)
```

To compare the distance between each enemy, I used p5's dist function which will find the distance between 2 coordinates, this will be stored in the variable , d.

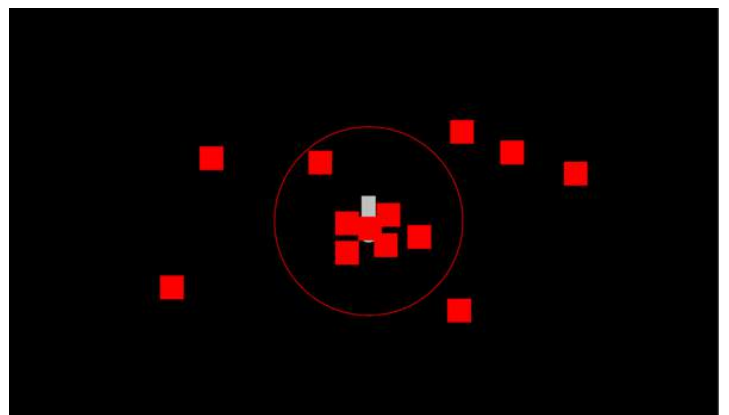


```
if (d < 0)
  // sets colliding as true so enemy
  cannot be drawn there
  colliding = true
}
// if enemy is colliding
if (colliding) {
  //stop
  this.speed = 0
}
```

When trying this , the results were unsuccessful, the enemies would just pass through each other when colliding, therefore the enemy technically hadn't collided.

```
if (d < 50)
```

After looking at my dist function I used the console.log on the value from dist and I noticed I didn't take into account that each enemy I drew had a size of 50 and so if each enemy were within a distance of 50 from each other, they would then be colliding . After modifying the distance required, enemies would now stop moving when colliding with other enemies



1.7 Tower Overlapping

Now, enemies would move towards the tower and would stop once they collided with each other, although this was not the same once they reached the tower, I wanted it so once the enemies had reached the tower, they would slowly damage it – I will add damage and health for the enemies and tower after I have finished creating a projectile for the tower. For now I needed to make it so once the enemy has reached the tower, they will stop, making sure that a collision flag is set to true. This will not be too difficult and rather simple as I have done a similar concept in 1.6 with the enemies, so I would just need to follow the same structure with the enemies but instead replace the enemies coordinate with the towers one .

```
towercolliding = false

// loop through existing enemy list
for (let k = 0; k < enemys.length; k++) {
  // finds the distance between existing enemy and tower
  var towerdist = dist(enemys[k].x, enemys[k].y,
    towers[0].x, towers[0].y)
  // if current existing enemies position is not equal to tower
  if (towers[0] != enemys[k]) {

    // if enemy is too close to tower
    if (towerdist < 50) {

      towercolliding = true

    }
  }
}
```

Firstly, I tried following the same structure previously used in 1.6 creating a flag that assumes that the enemy is not currently touching the tower so is able to move. Using the for loop will go through the enemy array so I can compare every enemy with the towers coordinate.

Creating the variable towerdist will hold the distance at the which the enemy is from the tower.

If the enemy is within a distance of 50, then they are touching the tower, so the flag tower colliding will be set true thus allowing me to halt its movement.

After running this to test if the program would function correctly I received an error that towercolliding was not defined, I realised I had not declared towercolliding so in order to fix this I used the let function to declare the variable.

```
let towercolliding = false
```

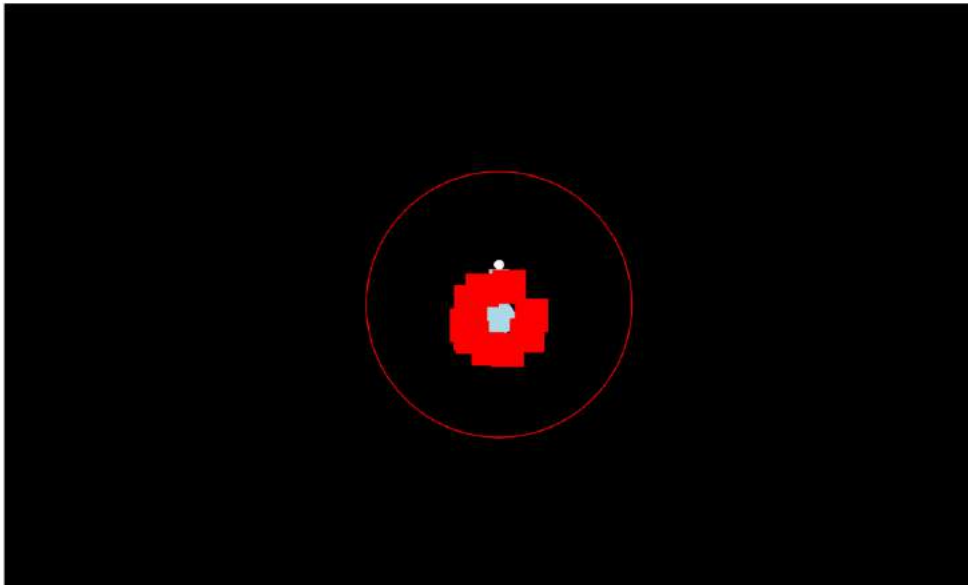
This seemed to have fixed the issue, so now I needed to make some conditions once an enemy is touching the tower – I currently haven't added a health and damage system yet so I will not add this yet but will add this in a later version so for now I will only update the speed.

```
var towerdist = dist(enemys[k].y, towers[0].y)
```

After looking over my code, I realised that my towerdist code that calculated the distance of the enemies to the tower would only calculate the distance of just 1 enemy rather than all the enemies, leading to all the enemies ending up stopping once a single enemy had touched the tower.

```
var towerdist = dist(this.x, this.y, towers[0].x, towers[0].y)
```

So to fix this, I changed the tower dist code to this.x and this.y which contained all the enemys x and y coordinates and make sure only enemies that are touching the tower will stop moving rather than all of them,



Now all the enemies will make their way towards the tower and stop as soon as they touch the tower's body. This counteracted the code from 1.6 where enemies would stop when colliding with each other but me and Jayden both agreed that enemies pilling on top of each other looked better so I decided to remove the code from 1.6 and implement this code.

1.8 Tower Projectile

Now that I have working enemies for the tower to shoot at, I would need to make the tower shoot a projectile at the enemy and then once the bullet has touched an enemy, the bullet and enemy will be removed from the array. I wanted the bullets to shoot out from the cannon, and so I would need that coordinates for the cannon, to make things easier, I created a new separate class for the tower called Tower Cannon .

```
class TowerCannon {  
  constructor(x, y) {  
    this.x = windowWidth / 2  
    this.y = windowHeight / 2 - 26,  
    30, 55  
  }  
  draw() {  
    fill("silver")  
    rect(windowWidth / 2,  
    windowHeight / 2 - 26, 30, 55)  
  }  
}
```

I used the previous code from the cannon to make the cannon a individual class from the tower and adjusted its position.

```
var towerCannon = [];
```

Var towerCannon creates a new array for the towerCannon allowing a new cannon to be pushed in .

```
towerCannon.push(new  
TowerCannon(this.x,    this.y,  
"silver"));
```

This would push in the new cannon object into the towerCannon array in order to draw the tower cannon onto the canvas

```
for (c of towerCannon) {  
  c.draw()  
}
```

The for loop in the setup function calls the draw function in the towerCannon class, allowing it to be drawn on the canvas

```
class Projectile {
  constructor(x, y) {
    this.x = x
    this.y = y
    this.speed = 1

    draw() {
      fill("white")
      ellipse(towerCannon[
0].x, towerCannon[0].y ,
15)

```

Then I created a new class for projectiles. The projectiles when the enemies are within the range of the tower, a projectile will be created at the position of the towers cannon is since that is where the projectile will be fired from.

Since the enemies aren't too small the projectiles will be drawn quite small as there will be no issue with them colliding and for balancing so a projectile will not accidentally hit 2 enemies at the same time.

```
function inRange() {
  //loops through enemy array
  for (let i = 0; i < enemys.length; i++) {
    //assumes that the enemy is not in range
    var hit = false
    //if enemy is in range, return hit as true
    hit = collideRectCircle(enemys[i].x,
enemys[i].y, 50, 50, windowWidth / 2,
windowHeight / 2, 400);

    console.log('colliding?', hit);

    //checks if enemy is in range
    if (hit) {
      projectiles.push(new
Projectile(towerCannon[0].x, towerCannon[0].y)

```

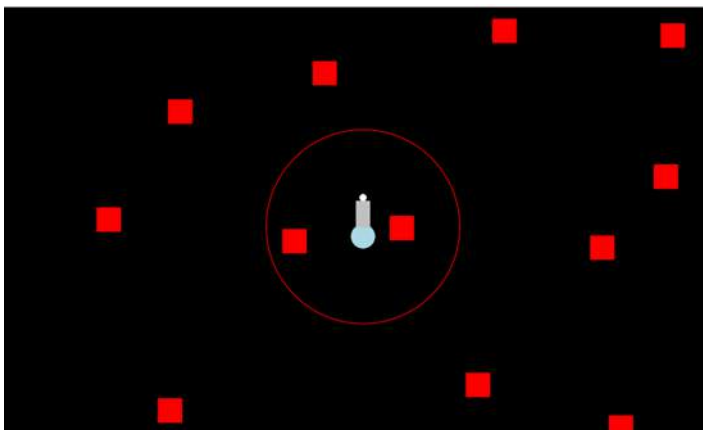
To detect if a certain enemy is within the tower's radius, I looped through the enemies' array to check if any of the enemies are currently within the range of the tower.

First it is assumed that the enemies are not within range of the tower.

Then, using the p5 collide library (bmoren, 2022) inputting the enemy's coordinates and the tower radius coordinates will be set true if an enemy is within range of the tower.

Using console.log allowed me to see if it would be detected if enemies were within range of the tower.

If an enemy is detected as within range of the tower, then a projectile can be created for that enemy.



Results of the testing showed once an enemy was in range, projectiles were drawn at the tower cannons position, though, the game seemed to significantly slow down due to an excessive amount of projectiles being created. I assume this is due to the hit flag being set true indefinitely for each enemy as so a number of projectiles will be created as a result.

Overall this prototype was successful but In the final development of the project I will need to try to solve this issue alongside moving each projectile to an enemy it is targeting and also limit the amount of projectiles created.

Menu Research



Diep.io has a game menu where you can pick up to 8 different game modes depending on what you would want to play and will switch you to that server depending on the game mode chosen. New updates are written on the top left showing the player if new upgrades have been added to the game or some general bug fixes. To enter the game, the player must put their name in the textbox, once done, they can press enter to start the game.



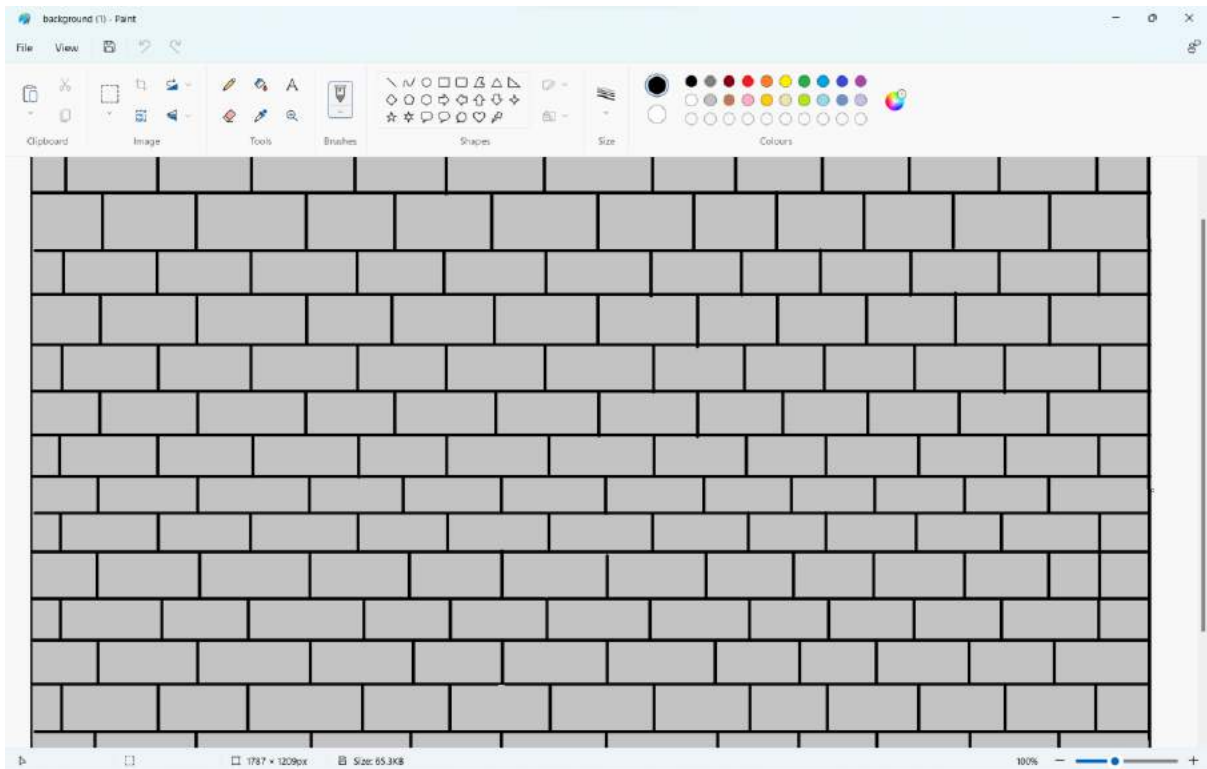
Stranger Things is a popular TV series that has been created into a game. The title has an ominous background to fit the title of Stranger Things. Other than this, there is a simple white start button which will start the game.



Cut the rope is a puzzle game where the player can access and complete multiple puzzle levels. The menu is well animated, the scissors move in a way the look like they are cutting an invisible rope, which in a way is the main premise of the game. There is a bright red button that will lead the player to the levels menu for the player to choose their game. The f icon will lead the player to cut the rope's Facebook page where updates will be poste regularly. The camcorder will play a short trailer/animation on the game to entice the reader in the lore of the game. Settings can adjust sound, music and you can also see all your achievements and trophies if you have collected any in your playthrough.

2.0 Making a Basic Menu

From my must criteria, I wanted to have a menu this is so the player can conveniently start the game for the player once they want by pressing the start button. To achieve this, I needed to first create the menu itself. I use the built-in Microsoft paint app to create a basic background.



After creating the background, I saved it as a png (background.png) and imported it over to replit by dragging it from my files over to replit. Now that the background picture is accessible, I now had to create a canvas and then use p5's preload function to load my external image to be my background.

```
let start;

function preload() {
  start = loadImage("background.png");
}

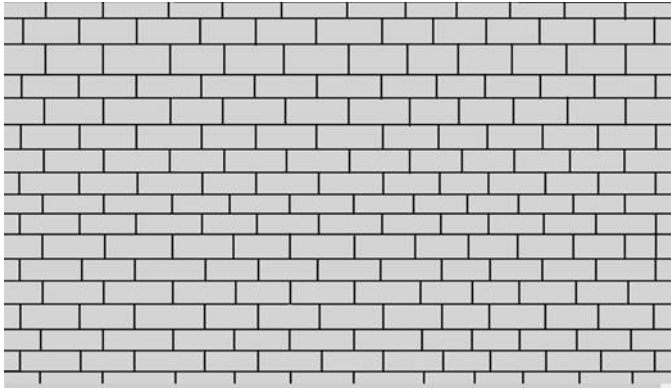
function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(start)
```

To begin with, I would need to make use of p5's preload function – this would have to be used before the setup function otherwise it would not work. By declaring a variable – which I called start, then assigning the variable to the loadImage function that will quite simply load my image. Since I called my image background.png I inputted this into loadImage.

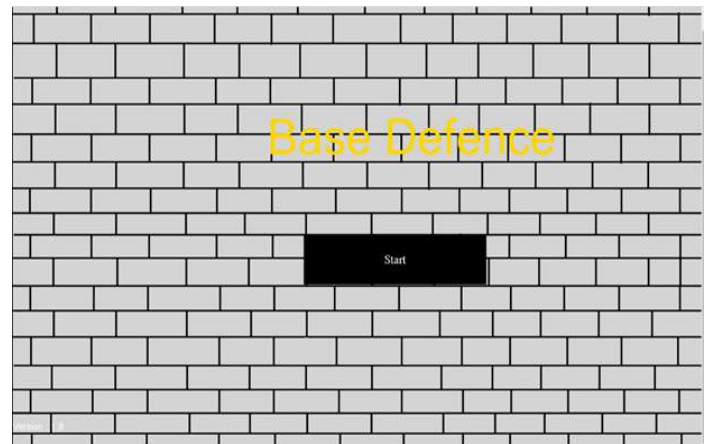
The setup function will create a canvas the size of the whole window by using windowWidth and windowHeight.

Draw() will use my stored variable start and quite simply draw my image – this will load in the image I saved as the background for the menu.



I didn't have any issues when loading in the background it seemed to fill the whole screen without error. Now that I had a background, I wanted to make a start button that will load the game as soon as the player touches the button and the game title Beta Defence.

```
function draw() {
  background(start)
  textSize(20)
  noStroke()
  fill("gold")
  textSize(100)
  stroke("black")
  strokeWeight(0)
  text("Base Defence", 560,
320);
```



By using p5's textSize to adjust the title and text() to draw the title I was able to make my title for my game on the menu. The problem now was that the title of the game can't really be seen at the moment since there is no outline on the text.

```
strokeWeight(5)
```

I realised this was due to the strokeWeight being set to 0 that caused this issue, as strokeWeight defined how thick outline of my title would be made it so as it was set to 0, there would be no outline for the title.



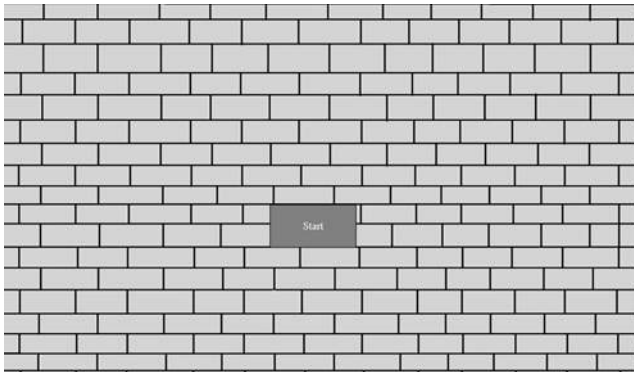
After adjusting the strokeWeight to my preference, the title of the game seems more visible for the player now.


```
touching = false

d = dist(button.width ,
button.Height, mouseX,
mouseY)
if (d < 50)
    touching = true
```

In most games which have menus, the start button will usually change colour to indicate that the player's mouse is hovering over the button, whilst this won't be required in the final version of the game I wanted to try and implement this if I were to add further visual graphics to my menu.

Similar to my [1.5 Enemy Spawn Overlap](#) code , I used the dist() function to calculate the distance between the players mouse position and the button, if the mouse position is within a certain distance of the players mouse, then the button will change colour, in this case, grey.



When trying this out, the mouse would change colour when the mouse position was touching the button, however this wasn't very accurate since only when the mouse was in a certain area of the button, the button would then change colour, it didn't take into account the whole of the buttons position.

I suspect this may be due to the button being drawn in centre mode, which would explain why the button would only turn grey whether the mouse cursor was within the centre of the button. Since the button is a graphics development I decided not to continue developing this, though if I have any further time in the final development, I may consider adding this.

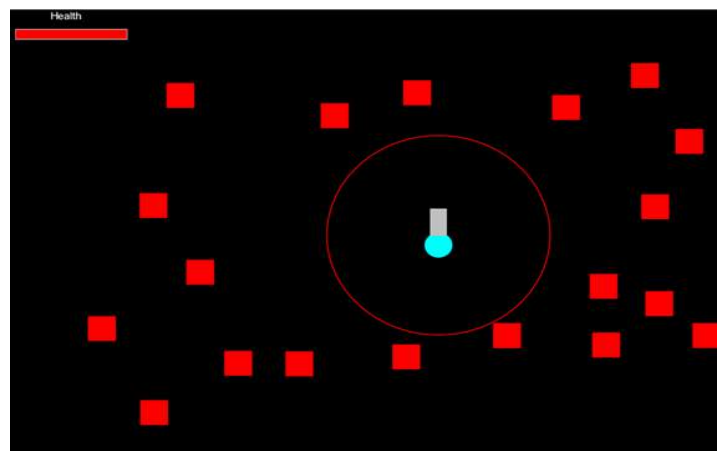
2.1 Basic Health UI

In the game, the enemies will make their way towards the tower, and once colliding with the tower, the tower will take a set amount of damage. The amount of health left from the player will be shown from a health bar UI in the top left-hand side since it takes the least amount of space and is not blocking any of the game as it is running. How the health bar will presumably work in the final version of the game is that it will be drawn a set size on the canvas and will decrease (indicated by the health bar decreasing) by a set amount based on health currently remaining. As I do not currently know how to make the health bar decrease by the amount of health remaining and will begin on the actual development, I will focus on the development of this in the final development of the project. For now I created a prototype of what the basic health system should look like when the player is at full health.

```
fill("white")
stroke("black")
strokeWeight(1)
textSize(20)
text("Health", 100, 20)
fill("red")
stroke("white")
strokeWeight(1)
rect(110, 50, 200, 20)
```

Since the background of the game is black, it would be more appropriate to colour in the health text as white as it is clearer and easier for the player to see. As for the health bar, in games health is usually defined as a red bar that indicates how much health a player has left, so I decided to colour this bar in red. I also added a stroke to this health bar since the bar couldn't be seen as clearly. I adjusted the health bar size to the left-hand corner of the screen since I plan to add some settings buttons on the right hand side of the screen later.

For now, this health bar other than aesthetics doesn't really do anything, once I begin to test out how to work with health in the final version of the program, I will use this code as a reference on what the final development of what a health bar UI could look like.



Upgrade Shop Research

Depending if my game reaches the minimum of the MUST section from my [Success Criteria](#), I will consider the choice of upgrades the player could chose to spend their money on since the addition of difficulty will require the player to have better stats to counter this.



Looking at some popular game shops, this gun game has a range of weapons that can be purchased using the in-game currency, gold when scrolling through the shop. When a gun is purchased, it can be upgraded. Whenever this happens, an upgrade can be indicated by one of the bars being filled blue. The money requirement of upgrading that weapon will increase and when the gun is "fully upgraded" the upgrade button will be unavailable but only for that weapon.



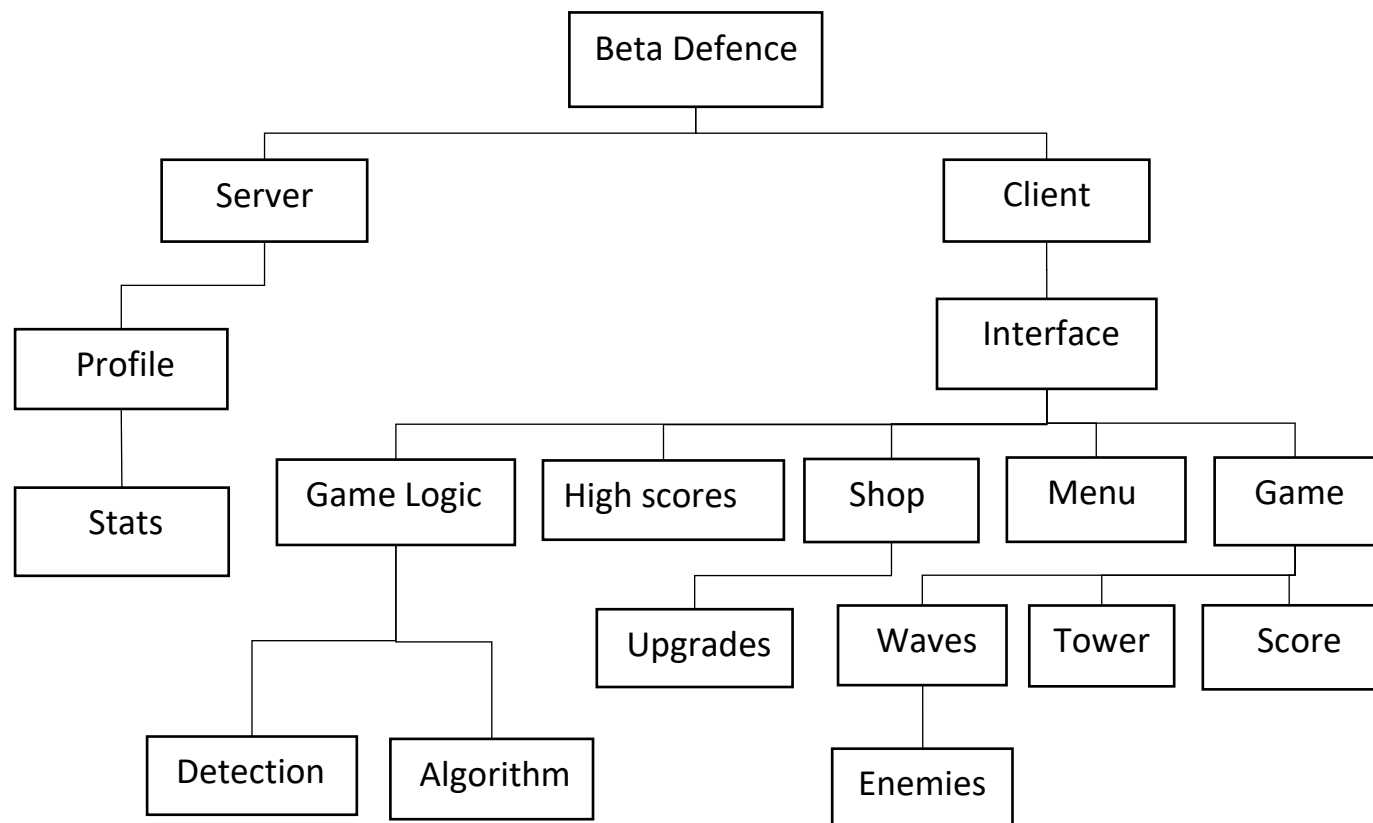
This different game has a similar shop UI, though uses 2 different currency : gems and a money system even though it isn't clear. When an upgrade is purchased a red circle will be filled in red and when fully upgraded will no longer be able to be upgraded as the upgrade button will be hidden. The shop also has items that can be purchased with gems. As long as the player can purchase the item, the player can purchase as many of these items as they want as there is no limit.



Diep.io is a game that takes a slightly different approach from the other 2 games, that being dependent on an xp system. Each time the player levels up, they receive a skill point, to which they can spend on an upgrade from 1-8. Pressing 1-8 will give the player the desired upgrade and the skill point will be consumed. An upgrade will also be indicated by a coloured bar system. Once the player has purchased all upgrades possible, they will no longer receive anymore skill points since there is no more possible upgrades, therefore there would be no point to received anymore skill points.

Design

Project Decomposition



Assuming that my game has been fully developed when finished, my project will follow the client server model where the server will hold and maintain every player's profiles and their scores whilst the client does the majority of the work as it will handle the main structure in order for the game to work.

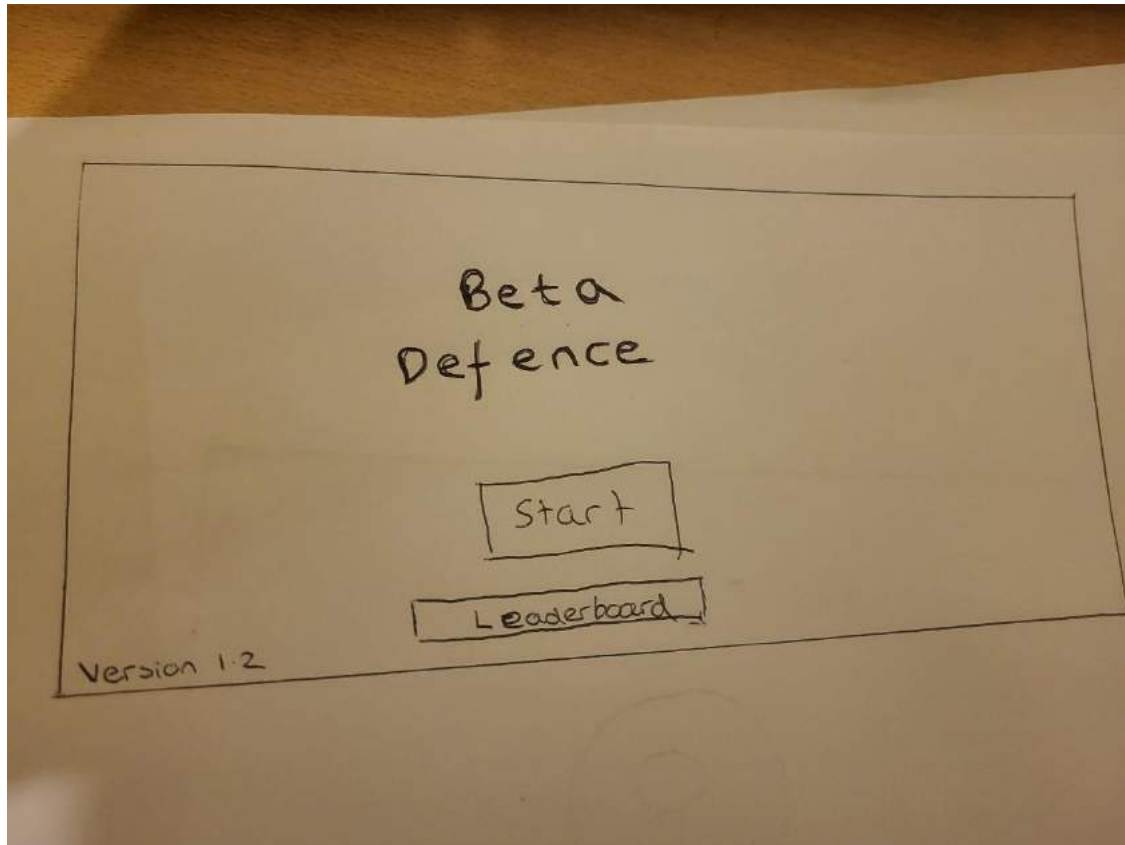
Waves, Tower, and Score will all change throughout the game, each having certain conditions to be met in order for this to happen, for an example the score could be changed based on the number of enemies killed or the highest wave the player has achieved this could also be impacted if I were to add a difficulty level that could drastically affect how the score works.

When the player will begin playing the game, they will need to interact with certain interfaces like the shop in order to progress through the game. The player should be able to interact with these interfaces which will update certain game objects in the game logic. The player can interact in the game using their mouse and clicking on the buttons and if on a mobile device then tapping the buttons will also have the same affect.

Game logic will be based on 2 key factors, the detection of objects, which in this case will be the detection of an enemy if it is within the range of the tower in which it will shoot and when a projectile collides with an enemy, it will deal a set amount of damage. Algorithms will be used to generate a set number of enemies in a wave and also determine a random location outside the canvas where an enemy can be drawn and also make sure it cannot overlap over another.

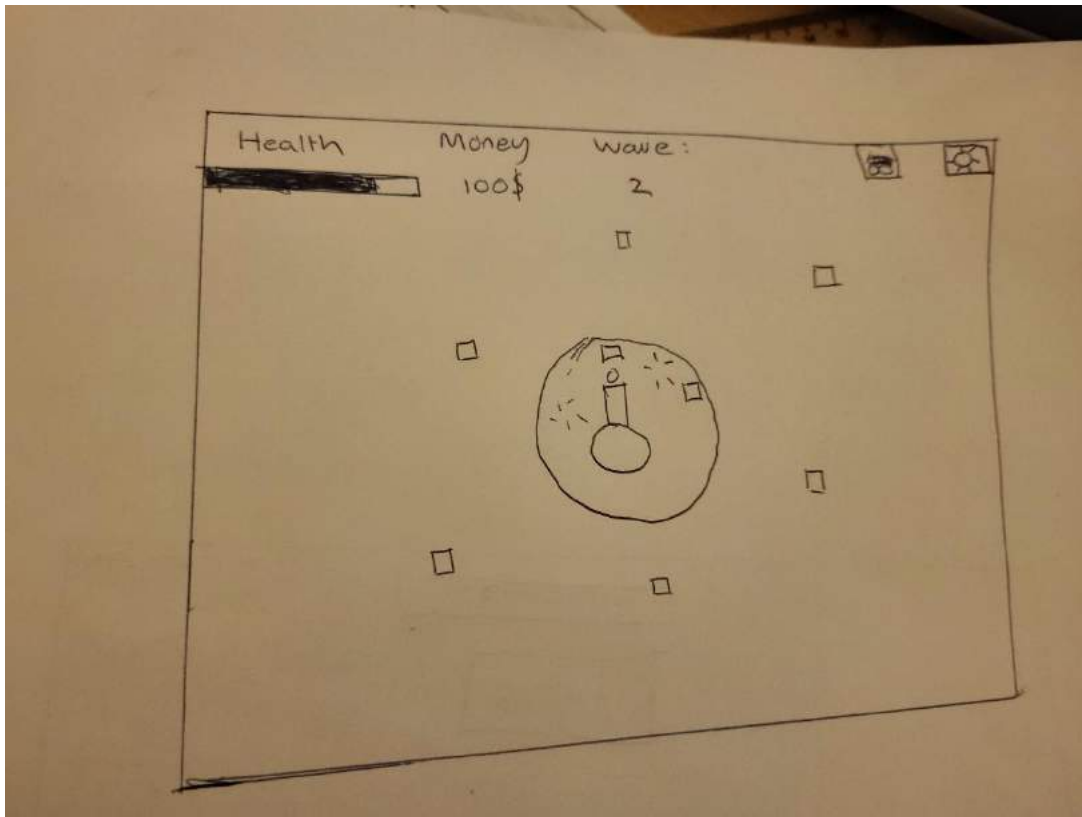
Interface Designs

Game Menu



Here is a possible game design menu I may implement. It will have a relatively large main game title since the menu has to entice the player to play the game, a relatively simple rectangular start button in order to start the game and a leader board if I have time for that. As soon as the start button is pressed the game will immediately start, the tower having no upgrades a set amount of health, and a set number of enemies being randomly spawned on the map. This will make sure once the player has quit the game, they won't be able to keep their upgrades and the waves won't start on a difficult wave. I may add some other objects in the menu since it currently is quite basic, this may include difficulty options and some interactive towers on the home screen page that shoot towards the mouse cursor. One other design I considered is where the player can type in their name in a textbox and once the enter key is pressed, the game will start. This will be a useful menu design if I were to add a leader board system as now I would be able to get the players username and save it to the server.

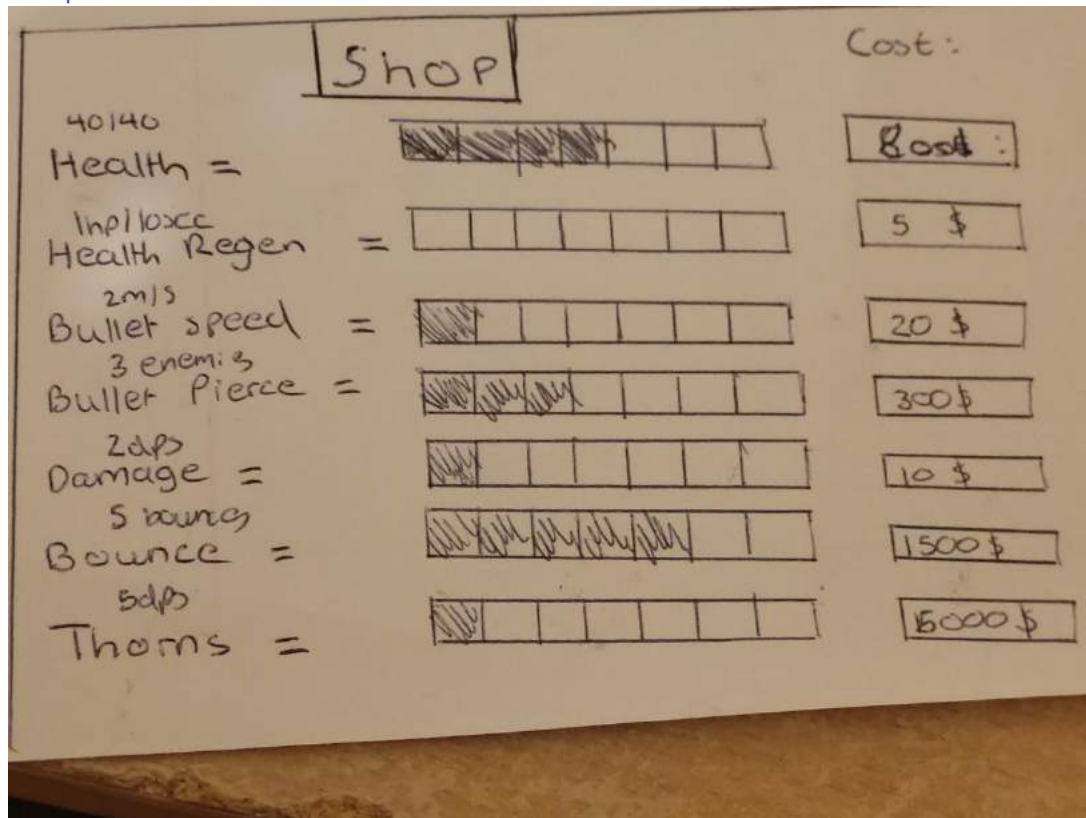
Game



In most default games the most important attributes to the player such as health or money on the left side of the screen and any other additional button will be displayed on the right-hand side of the screen since it does not cover or distract the player when playing the game, so, I also decided to follow this layout .

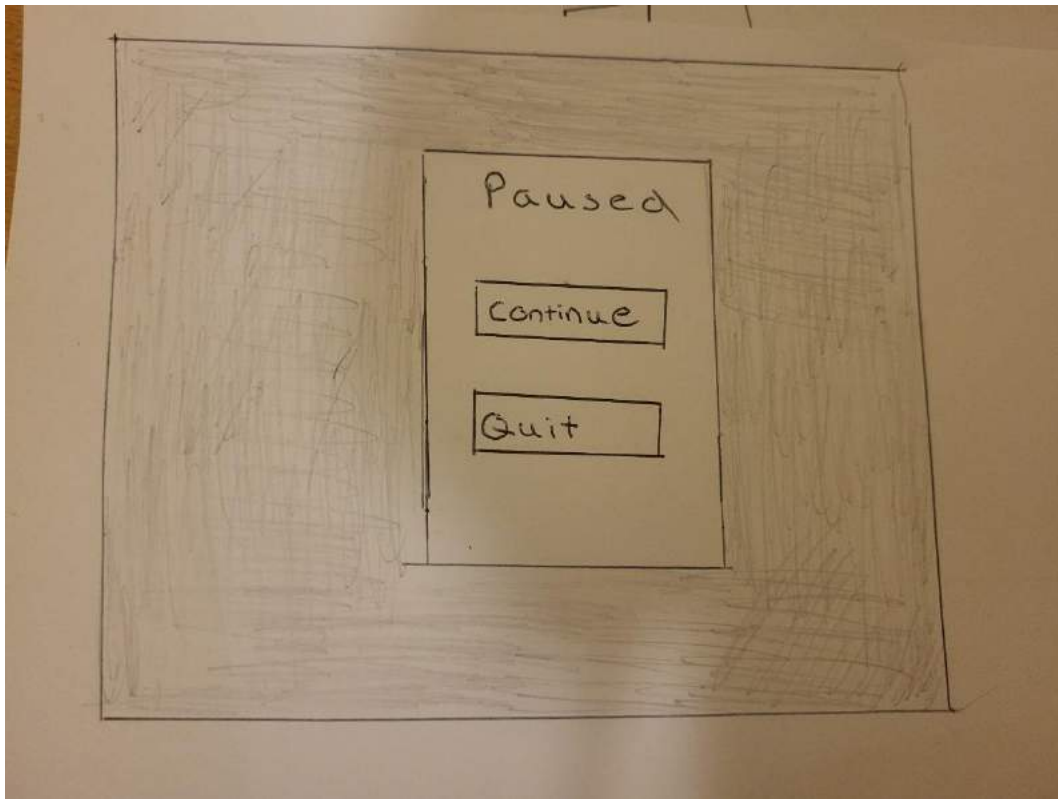
During the game, if an enemy is within the towers range, the tower will shoot that enemy and will prioritise the closest enemy first, when an enemy is defeated by a bullet I may add some particles or special effects to indicate this happening. The amount of money gained based on the enemy defeated and the wave the player has cleared. When an enemy does reach the tower, the health bar will update according to how much damage the tower has taken. A wave will be complete once all the enemies are defeated in that wave. Money, wave, and tower health values will automatically update when their conditions are met. The tower is positioned in the middle of the screen since it is the main thing the player will be focusing on .The settings button will pause the game and allow the player to adjust some settings, take a pause or quit the game. The shop icon will allow the player to purchase upgrades in order handle more harder enemies on later waves. I may also add some additional sound if a player's mouse is hovering over certain icons.

Shop



In the shop there will be a variety of upgrades that can affect the towers strength. Once the player has upgraded an upgrade once, the upgrade will increase in price by a set amount and also depending on the upgrades power – this may be difficult due to the balancing of certain upgrades and the amount of money the player receives therefore the prototype may not be accurate to the final version. When the player has enough money, the button will be lit up, indicating that the upgrade can be purchased and if not then the upgrade will be dimmed out and the button cannot be pressed. Once an upgrade has been purchased, a bar of that upgrade will be coloured, indicating that an upgrade has been purchased. There will also be a cap limit to certain upgrades once the upgrades bar has been fully purchased. The upgrades will also be ordered in terms of price to make it clear to the player which upgrades to prioritise over others.

Pause Menu

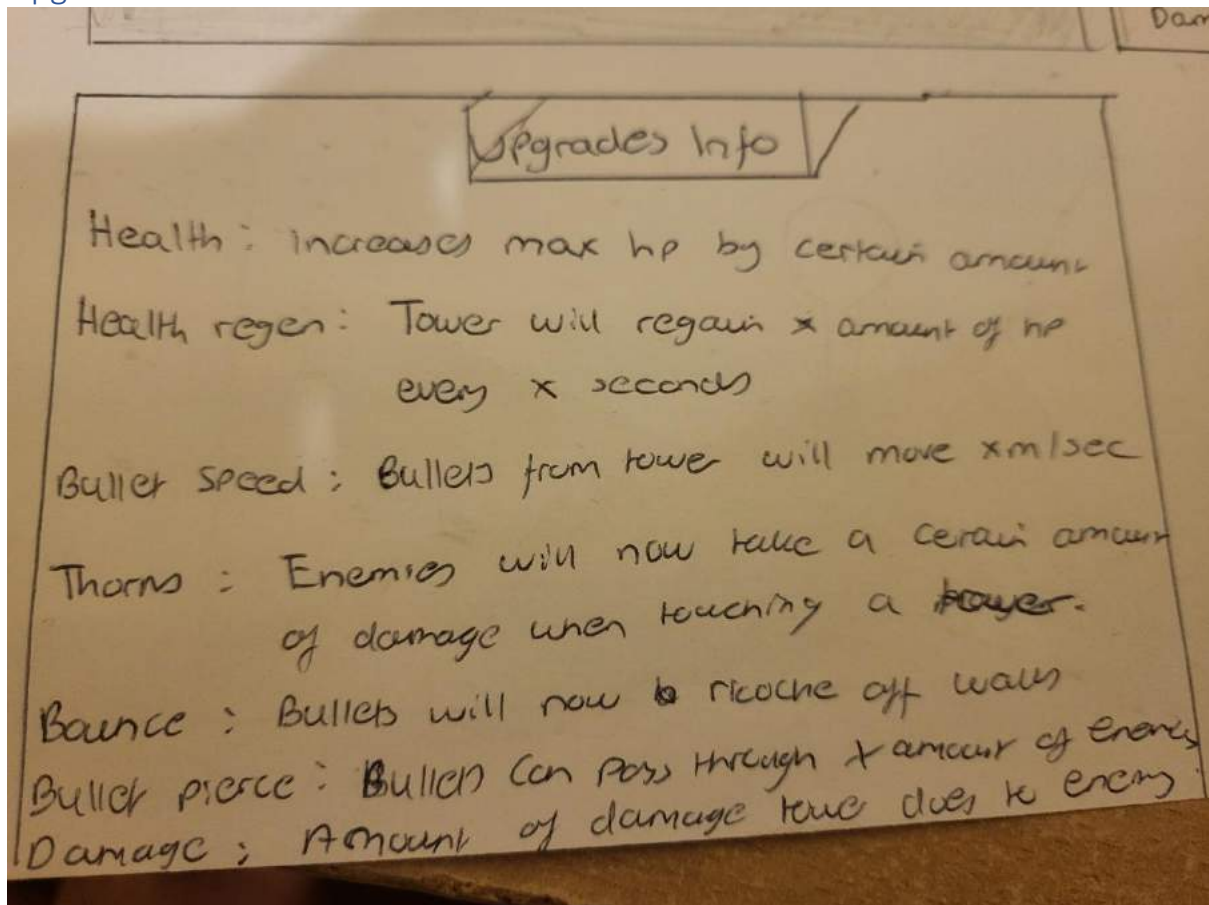


When the pause button is pressed, a pause menu will appear in the middle of the screen so the player can clearly see the game has been paused . During this time, all enemies, player projectiles and the shop will be inaccessible as the game is paused . The player will have 2 options to either press continue, which will resume where the player had left off before they had clicked the settings button.

Quit will lead the player back to the game menu. This will reset all game progress as any of the global variables - upgrades, waves, money, or health will be reset back the default values before the game had been started.







During the time the game is paused, the background will become dim but still slightly transparent so the player can still slightly see their game, this could be used to potentially allow the player to organise their strategy if they encounter an upcoming difficult wave or just allow the player to pause the game if they would like to play the game later.

Upgrade Info



I could possibly make the upgrades info separate area from the shop by making an info button that will contain all the upgrades information on what all the upgrades in the shop will do when purchased. This is done so new players can understand what certain upgrades will do without being blind when purchasing the upgrades. Though I could also make more clearer alternative design where I could display this info when the players mouse is hovering over the specific upgrade, a small UI will appear showing the respected information on the upgrade since it would be more efficient and one less button for the player to press.

Leader board

Leader Board			
			Score :
		AdamLikeUnicorns	909102
		JimBrowning86	850000
		Blublackvoid	846200

If I have time in my project, I will have every players stored score and profile saved on a server which I will then add to the leader board according to the top 3 highest scores. This will label the users username and score. This may be difficult though as I have no knowledge on how to store data on a server and so will be considered as an additional feature I can add if have time in the development of my game.

Internal and External Storage

Server side

Assuming my project includes the use of the storage of data on a server, the server will only hold a limited amount permanent data that being the players name and score which will then be transferred onto the leader board depending on, if the players score is larger than the highest score out of all the players that have played the game .

Player Table

Field name	Datatype	Comment
Player Name	String	A player name must be unique, therefore, a name inputted by the player must not have been used before to prevent the leader board being filled with multiple people with the same name.
Player Score	Integer	A series of integers obtained from the player after a game : this will be a unique value for every individual player based on the amount of enemies defeated/defeated/number of waves completed. These scores will be saved on the server and displayed on the leader board if the score is the largest out of all the players who have played the game.
Game State	JSON string	A JSON structured string that stores the state of the game, the amount of health the player has, the amount of money the player has earned during the game and if upgrades have been locked/unlocked.

Client-side

My first few versions of my game will primarily store the game state of the game in the local storage as well as key data structures, that being classes that will clarify the storage and interaction of the tower, enemies also including the usage of the shop.

Tower class

The tower class will store data on the tower object alongside methods which will use the tower data.

Attributes	
Coordinates	Contain the coordinates of a set location where the tower will be drawn on the canvas – this will contain the parameters (x,y,r).
Player Health	A set integer that determines how many times the tower can be hit before the game ends.
Player Name	The name of the player which will be unique.

Cannon	Will be drawn with the parameters (x,y,w,h)
Tower Radius	The area in which the tower can see the enemies. Once enemies can be seen by the tower, a new projectile will be pushed into the projectiles array this will iterate until no more enemies are within the range of the tower.
Projectile	An object that will be pushed into the projectile array once an enemy had collided with the tower's radius . The projectile automatically will move towards the nearest enemy within the towers range and once collided with the enemy, will be removed from the array and deal x amount of damage to the enemy. If damage done to the enemy causes the enemies health to be below 0, that enemy will also be removed from the enemies array.
Methods	
Draw(x,y)	Takes the parameters of the tower and draws it onto the canvas with the given coordinates once the game state is set to play.
CanFire(x,y)	Detects and determines when a new projectile can be drawn at the given coordinates, this happens when an enemy is within the towers range.
Move(x,y)	Moves the projectile towards the nearest enemy at a set speed.
Update(x,y)	The towers cannon will be updated and redrawn to wherever the tower is currently shooting at the enemy from. If the player takes damage, the player health bar will decreased based on the % of health has left.
Attributes	
Position	The set parameters (x,y) location of where the tower, tower range, cannon and projectile will be drawn from.
Velocity	The rate at which the tower projectile will move towards the enemies
Direction	Determines what angle the tower cannon and projectile must move in order to be facing and moving towards each enemy.

Enemy class

The enemy class will store data on the enemy objects alongside methods which will use the enemy data.

Attributes	
Coordinates	Contain the coordinates of a set location where the tower will be drawn on the canvas – this will contain the parameters (x,y,r).
Enemy Health	A set integer that determines how many times the enemy can be hit before it is defined as “dead” so is removed from the enemy's array.
Enemy Type	The unique identification of a certain enemy.
Methods	
DrawRandom(x,y)	Generates a random coordinate within the canvas and checks if an enemy can be drawn there.
CanDraw(x,y)	If an existing enemy is already drawn at the x,y coordinate, another location will be checked with existing enemies in the enemy array until a new enemy can be drawn or until the set amount of enemies pushed into the array has been reached.
DrawAmount(x,y)	Iterates through enemy array until a set amount of desired enemies are drawn on the canvas as long as they do not overlap each other.
Move(x,y)	Makes the enemy move towards the tower at a set speed.
Update(x,y)	Enemies speed, health and amount of enemies will be increased based on the wave the player is currently on. Enemies will also be removed from the array once that enemies health has reached below 0.
Defeated	Checks every enemy in enemies array and if an enemy has a health < 0, remove that specific enemy from the array and emit particles or explosions in that area if enabled by the player.

Attributes	
Position	The set parameters (x,y) location of where the enemy will be drawn from.
Velocity	The rate at which the enemy will move towards the tower
Direction	Determines what angle the enemy must move in order to make its way towards the tower

Shop class

The shop class will store set values on upgrades as well as methods used cooperatively with the tower class by updating the tower data.

Attributes	
Name	A unique set of string that determines what a specific upgrade is named.
Image	An image of that specific upgrade.
Unlocked	A Boolean value which at default will be set to false. Determines whether an upgrade has been unlocked by the player or not.
UnlockCost	The amount of currency the player will need to unlock an upgrade, can range from 0 and onwards. If cost is free then upgrade can automatically be purchased.
IncreaseUpgrade	The cost in which to increase the power of an upgrade. This will increase by a set amount depending on the upgrade. If the player can afford the upgrade, the button will be set to true so the upgrade button will be enabled and if not false, where the button is disabled until the player meets the upgrade requirements.
MaxUpgrade	The maximum amount of times the player can increase a specific upgrade. Once this has occurred will be set True, and upgrade button will be disabled.
Methods	
DrawShop(x,y)	Pauses the game and draws the shop onto the canvas once the shop icon has been clicked by the player this method is inherited from the games button class.
CanUnlock	Checks if the players current money with the locked upgrade price. If player money < unlock price, CanUnlock will be set to false, so the player cannot unlock the upgrade. If player money > unlock price CanUnlock will be set to true, so player can unlock that specific uograde.
PlaySound	Once item in shop has been purchased, short game upgrade sound will emit.
IncreasePrice	Increases the cost of an upgrade by a set value once the player has purchased an upgrade.
CanUpgrade	Checks if the player has purchased the maximum amount of a certain upgrade, if so Can upgrade will be set to false, else the player can continue purchasing that upgrade as long as they have enough currency.
Attributes	
Position	The set parameters (x,y) location of where the shop will be drawn from.
Button	An interactive object that will run set code once pressed by the player. Can be enabled or disabled based on the conditions.
Update	Can change data on the enemies based on the wave, tower attributes such as health or damage when an upgrade has been purchased and whether certain button are disabled or enabled.

Button class

The shop class will store set costs on upgrades as well as methods used cooperatively with the tower class by updating the tower data this is key for the main basis of the game to work.

Attributes

Name	Determines the name of a button used.
Image	An image of that specific button.
State	A Boolean value which at default will be set to true. Determines if button is usable or not. Also checks if game buttons has been pressed, if so change the game state depending on the button.
Methods	
DrawButton(x,y)	Draws certain buttons onto the canvas in a set location based on the game state
IsEnabled	Checks if button can be pressed or not – is usually set to true. Is automatically set to false if certain requirements are not met, e.g – upgrade is more expensive than the player's money.
PlaySound	Once a specific button has been pressed, play sound. Sound can vary based on the button and state.
Disable	Dims out a button indicating the button cannot be pressed or used, this will be set to true if requirements are met.
Attributes	
Position	The set parameters (x,y) where the buttons will be drawn.
Type	Determines what type of button is used based on the game state, the status of that button – if it is enabled or not and the code that the button will run once it has been pressed by the player. Can vary from game buttons which will affect the game state or the shop buttons which will change the data from the tower class if clicked.
Update	Changes data on the game state or tower class based on the button pressed.

Algorithms

The main basis of the game will consist of features from the Must criteria from the MOSCOW analysis, the algorithms in this section will cover how these algorithms are used and will show the minimum requirements of what my project will include when completed.

Menu

Draw Menu

This algorithm will be called when the game is started.

Parameters : window width, window height, game state, button type

- If game state is currently in the menu
 - Create a new canvas with the parameters of the window width and height
 - Locate the preinstalled image in the server files
 - Load this image as the background image for the game
- Draw the title text in initialised location
 - Detail text in set colour and font
- Draw start button
 - If button is pressed change game state to play and draw game

To test this I will :

- Set the game state to menu to check if everything in the menu is drawn properly in the intended location.
- Using console to see the current game state in order to see if the correct variables are drawn correctly.

- Try generating multiple buttons to see when clicking a button if the game state will change and if so checking if a new screen will be drawn.

Enemy

Random Enemy

This algorithm will be called when the game button start is pressed.

Parameters : random (x coordinate in canvas width) , random (y coordinate in canvas height)

- When game state is currently in game, draw onto canvas a set number of enemies required.
 - Select a random location that is within the canvas width and height.
 - Find a random (x,y) coordinate that is free within set parameters, but not near the tower, that being a set distance away.
 - If there is a space free to draw the enemy then the enemy can and will be drawn in that location.
 - If a new enemy about to be drawn on the canvas intersects with an already existing enemy drawn, then do not draw that enemy and look for a different location to draw this new enemy – ensuring that the enemy will be drawn on the canvas as soon as there is space.
 - Continue looping until enemy's array is filled with the required enemies for that wave .
 - Until the enemy's array has the set number of enemies, the loop will continue to draw the enemies.
 - Return the set number of enemies.

To test this algorithm I will :

- Performing a stress test on the algorithm by trying to generate a large quantity of enemies on the canvas to see if all the enemies are able to appear on the canvas at once showing me if they don't overlap over each other during this.

Move Enemy

This algorithm will be called every animation frame once the game is currently playing and as long there is an enemy present in the enemy's array.

Parameters: enemy position, tower position

- Check if enemy is present in enemy's array
 - Iterate through enemy array so the distance between all enemies can be found
 - Calculate distance between every enemy's x and y coordinate with towers x and y coordinate using Pythagoras.
 - Move the enemy in the direction multiplied by the current enemy speed.
 - Update enemy's location every frame.
- If enemy is touching the tower
 - Deal set amount of damage
 - That specific enemy touching the tower will stop moving towards the tower but set amount damage is continuously still dealt every few seconds.

To test this algorithm I will:

- Refresh the game multiple times to see if the enemy actually changes position

- When refreshing the game, check if the enemy when moving will only move towards the tower and not out the map, also checking if the enemy will stop when touching the tower.
- Using console to see if flag Touching is set to true to determine if entity collision has happened
- Change game state to play so I can see if the game actually draws the enemy's movement.

Tower

Draw Tower

This algorithm will be called when the game button start is pressed. This is slightly different to the drawing of enemies since the tower is drawn in a set location rather than a random one.

Parameters : (initialized x coordinate in canvas width), (initialized y coordinate in canvas height)

- When game state is currently in game, draw onto canvas the tower into the canvas alongside the tower cannon in set coordinates.
 - If a new enemy about to be drawn on the canvas intersects with an already existing enemy drawn, then do not draw that enemy and look for a different location to draw this new enemy – ensuring that the enemy will be drawn on the canvas as soon as there is space.
- Continue looping until enemy's array is filled with the required enemies for that wave .
 - Until the enemy's array has the set number of enemies, the loop will continue to draw the enemies.
 - Return the set number of enemies.

To test this algorithm I will :

- Performing a stress test on the algorithm by trying to generate a large quantity of enemies on the canvas to see if all the enemies are able to appear on the canvas at once showing me if they don't overlap over each other during this.

Detect In Range

This algorithm will be called every animation frame once the game is currently playing and as long there is an enemy present in the towers range.

Parameters: enemy position, tower range position, tower cannon position

- Compare the enemy's position with the tower's range
 - If the enemy is currently touching or within a certain distance of the towers range
 - Set flag InRange = true
 - If flag InRange is true
 - Push in a projectile array into projectile array at the tower cannons position
 - Check if another projectile can be drawn if there is a cooldown.
 - Iterate through enemy's array until nearest enemy is found
 - Compare distance between nearest enemy and projectile
 - Update the tower cannons position to wherever the nearest enemy position is by comparing both x and y coordinates
 - Move projectile in that direction towards the nearest enemy
 - Once collided with enemy, remove projectile from array and deal a set amount of damage to enemy, if

the enemy has a health < 0 then also remove that specific enemy from enemy's array.

- Update projectile's location every frame.

To test this algorithm I will:

- Set the generation of enemies to be near or within the towers range to see if the tower will start shooting the enemies.
- Check when the collision between enemies and projectiles happens, only the enemy and projectile colliding with each other will be removed from their respected array.
- Use the console to check if the flag InRange is set to true once an enemy has entered the range of the tower
- Iterate testing's of the game -- during the running of the game I will check to see if the tower cannon is redrawn to where it is firing at the enemy.

Game

Current Health State

This algorithm will be called during the game state : play and will update the data in the tower health class and also update the display of the health bar when required.

Parameters: tower health

- At the start of the game, draw health bar in set location where the player can see the game and the health bar clearly.
 - Health bar will be filled drawn and indicated by a red bar.
 - Check every frame if tower takes damage
 - If the tower has taken damage
 - Update the health bar by filling in a white bar over the health bar based on the percentage of health the tower has left.
- If the current health is equal or less than 0
 - Change game state to lose screen
 - Lose screen is drawn
 - Lose screen buttons are drawn
 - Enemy arrays is cleared

To test this I will :

- Make use of the console to check current health of the tower at the start and during the game, the current game state of game, the current value the health is and the state of enemies in the enemy array after the game state is lost.

Development log

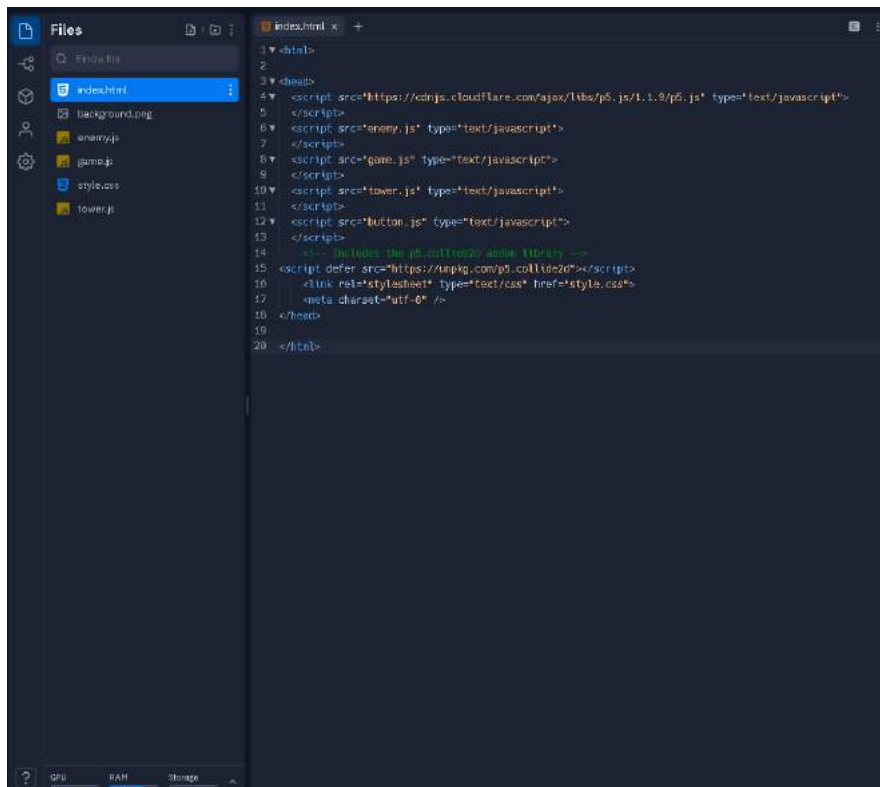
The use of iterative testing through the means of prototyping allowed me to create the main frameworks of how the main game will work . I will now use new and old feedback from Jayden and my stakeholders combining with the use of the previous prototype developments to attempt to improve and develop on my game in order to hopefully create a slightly more polished and finished tower defence game for my project.

Version 0.1

Drawing the basics

To start with, I will begin with the main basis of the game : the actual game itself which includes the game background, tower, and the creation of enemies .

Firstly, I created the necessary files required for each variable that will be required for the game and made sure these files will be called in the html files so they could be ran by linking the file to the script, also making sure to add in the latest version of the p5 library in order to be able to draw the main objects of my game and the p5 collision library which allows an easier alternate way to detect entity collision detection to which is very important in my game.



Then I created the canvas, making sure to set it to be the window width and window height so the background would be able to fill the whole screen for the player and then create background of the game to which I found no problems with. Looking at previous feedback from Jayden, I thought about creating a moving background and so would be more appealing to play and enjoy the game, however, I decided this may be too resource intensive on the game and may also be too distracting for the player when they are playing the game. Since this does seem like a nice idea, I may come back to this idea at a future stage in the development of the game but for now I should create the main features of the game first and so, I kept the background as black for simplicity.

Previously in my prototype there was a slight inconvenience when loading the game, there would be a default scrollbar on the side of the screen that would block some of the screen and was relatively distracting for the player.



To remove this, I simply had to set the scrollbar as hidden in the style.css file in order to remove the scrollbar from view for the player.

```
body{  
  overflow: hidden;  
}
```

After that, I created each class for the tower and enemies using previous code from my prototype filled them in with their respected attributes, was relatively straightforward due to the numerous testing in the prototypes - [1.1 Creating the player](#) I then declared the necessary arrays for each object required for the game so when pushed in their respected array would be drawn onto the canvas.



Also, unlike the prototype, I decided not to add the collision between enemies and other enemies since it would lead to the enemies being stacked on top of each other and so would have to wait in line with each other before reaching the tower this reducing the speed of the game, Jayden also commented on this, suggesting that the enemy-on-enemy collision was not really required and looked better without it.

Enemy creation

The only problem arose when I came to the creation of the enemies. From my prototype [1.3 Making Enemies](#), I used a for loop that would allow a set number of enemies to be attempted to be drawn

onto the canvas, if they were drawn over each other, they would not be drawn. Since this would happen, only the enemies that could be drawn onto the canvas would be drawn e.g I wanted 10 enemies to be drawn onto the canvas but since the function didn't allow the enemies to be drawn on top of each other, only the enemies that were able to be drawn were drawn and the enemies that couldn't be drawn ended up not being drawn, overall leading to less than the required enemies being drawn onto the canvas. It seems this current code would only attempt to draw the enemies and rather not continue to draw the enemies until the set amount of enemies had been drawn.

How this algorithm should work is that enemies shouldn't be able to be drawn near each other and 20 enemies should be drawn :

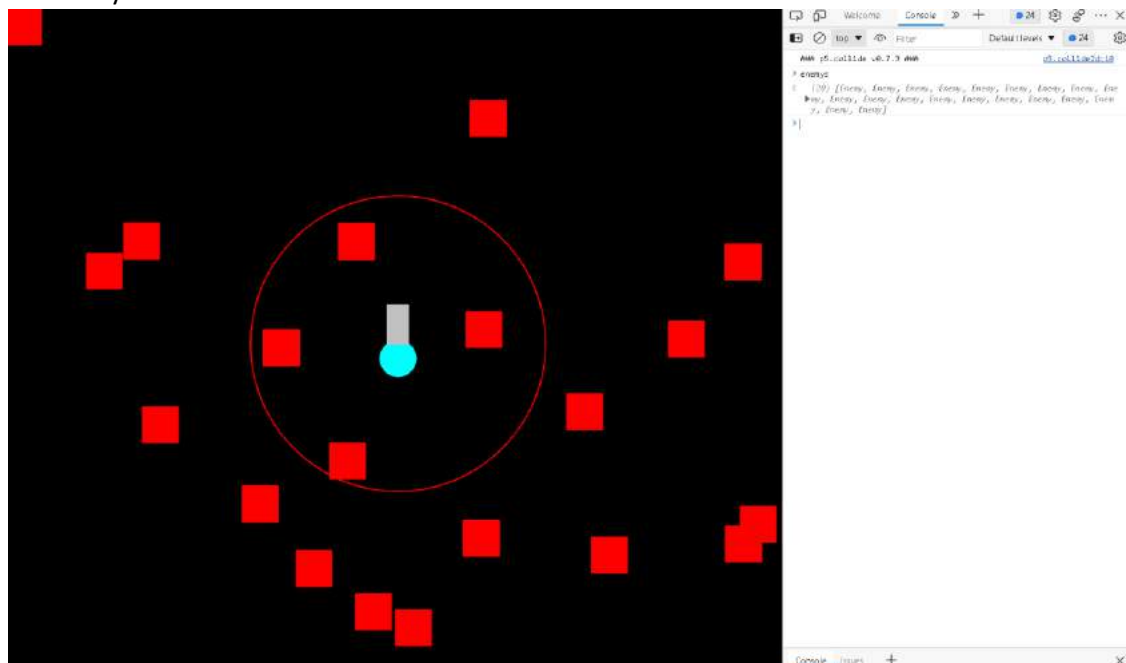
- If enemies are within a distance of 100 each other
- Do not create the enemy and so, create a new enemy
- Keep creating enemies until there are 20 enemies in the enemies array present.

I attempted to add a secondary condition to the spawn function (this would draw the set amount of enemies)

```
if (d < 100 && enemys.length > 20)
```

Whilst the console did show that the set amount of enemies (20 enemies) were drawn onto the canvas, it didn't fit the drawing distance requirement from each enemy : each enemy weren't drawn within a distance of 100 of each other therefore only one condition was met which was incorrect. This may be because my code attempts to draw 20 enemies and if the enemies cannot be drawn, they will not be drawn.

I basically needed to make sure 20 enemies were drawn onto the canvas but also would be drawn a



set distance from each other. After looking online at the type of loops used in JavaScript, I noticed instead of using a for loop which will attempt to draw my enemies a while loop will continue to

attempt to draw my enemies until the array reaches a fixed amount which is what I attempting to do here.

```
while(enemys.length < 20){
```

Replacing the for loop with the while loop made it mandatory for the enemies to be drawn onto the canvas with the set conditions of being random, within the canvas and being drawn a set distance away from each other.

Unfortunately, I still had an issue – enemies could still be drawn within the towers range and on the tower itself when starting the game, this would and could lead to the player dying to the enemies immediately at the start of the game which would not be a good game design, especially for a tower defence, so it would be necessary for me to set another condition that will limit the enemies from drawing within the towers area.

Version 0.2

Enemy creation issue

Continuing on from the previous issue, currently every enemy is being drawn too close to the tower when being created, so I would have to implement a secondary condition to prevent the enemies from being drawn this way before they are actually drawn.

Now this meant the enemies required an additional requirement, the algorithm for each enemy should have these requirements:

- If an enemy is drawn near the tower
- Do not draw the enemy and create a new enemy
- Additionally, enemies cannot be drawn near each other
- Each enemy must have a random position on the canvas
- A total of 20 enemies will be drawn with these 3 conditions.

First I tried using p5's collide library (bmoren, 2022) – which would detect if the enemy is being drawn within the towers range by returning true as soon as an enemy meets this condition and so, if this condition is not true, then the enemy being drawn is not within the towers range, then I can draw the enemy in that location this could also help preventing the enemies from being drawn within a set distance from the tower.


```

for (let j = 0; j < enemys.length;
j++) {
    // finds the distance of enemy
    that is going to be drawn with
    existing ones
    var d = dist(xco, yco,
enemys[j].x, enemys[j].y)
    hit = collideRectCircle(xco,
yco, 50, 50, windowWidth / 2,
windowHeight / 2, 400);
    console.log(hit)

    if (d < 100)
        colliding = true

    if(hit)
        colliding = true
}
// if enemy is not colliding, then
create enemy in that position
if (!colliding) {
    enemys.push(new Enemy(xco, yco))
}

```

Firstly, I looped through the enemy array to make sure all enemies will fit the condition to be drawn within a set distance of each other and also not be drawn within the towers area.

P5's dist function is used here to find the distance between the enemy that is going to be drawn with an existing enemy. If the enemy that is going to be drawn is within a set distance from an existing enemy, it will not be drawn, the while loop previously will make sure this enemy will eventually be drawn whilst fitting these conditions though.

Adding console.log will output if the enemy is drawn within towers range if so, I can make the necessary changes if this happens.

Using the p5 collide library, I stored the variable hit to detect once an enemy has successfully collided with an enemy, this will allow me to confirm if an enemy is being drawn within the tower and so I can prevent the enemy from being drawn there.

If the enemy is drawn within a distance of 100 and if the enemy is drawn within the tower's radius, the flag collide will be set true, thus the enemy won't be drawn.

If the enemy is not colliding within the towers range and is drawn within a set distance from each other, a new enemy can be drawn in that specific position

```

    for (let j = 0; j <
enemys.length; j++) {

        // finds the distance of
enemy that is going to be
drawn with existing ones
        var d = dist(xco, yco,
enemys[j].x, enemys[j].y)

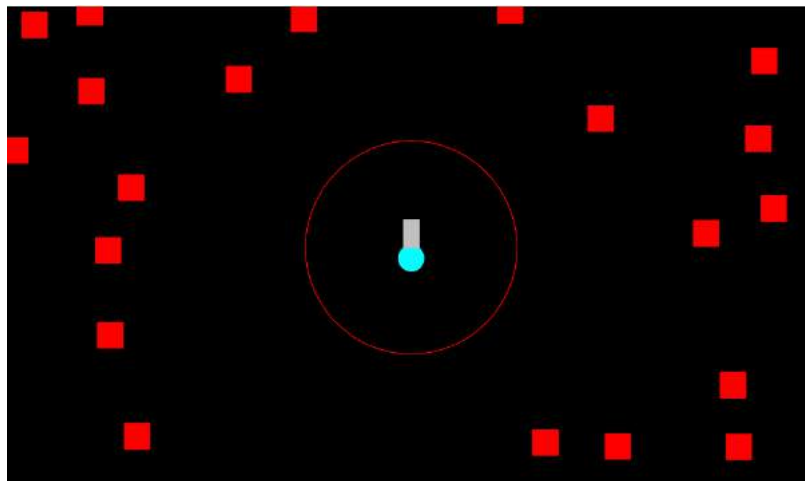
// if enemy is too close -
within a distance of 100 of
each other don't create

        if (d < 100)
            colliding = true
    }

    if (!colliding) {
        enemys.push(new Enemy(xco,
yco))
    }

```

As a result, by adding this to the previous code in the beginning of the development that made sure enemies would be drawn a set distance from each other, this seemed to fit all the necessary requirements for the enemy when being drawn – enemies will now be drawn a set distance from each other and not be drawn within the tower – if I wanted to make the enemies to be drawn further from the tower, I would have to just change the if statements that checks if the enemy is drawn near the tower to be further. I did make sure to refresh the program multiple times again to see if there would be another odd occasion where some enemies would be drawn within the towers area, but this time every condition seemed to met when drawing the enemies.



Now that the enemies are appropriately drawn, the next step to do is to make the enemies move towards the tower. This was previously covered in the prototype section [1.4 Enemy Ai](#) so programming the enemies to move towards the tower shouldn't be too difficult.

Version 0.3

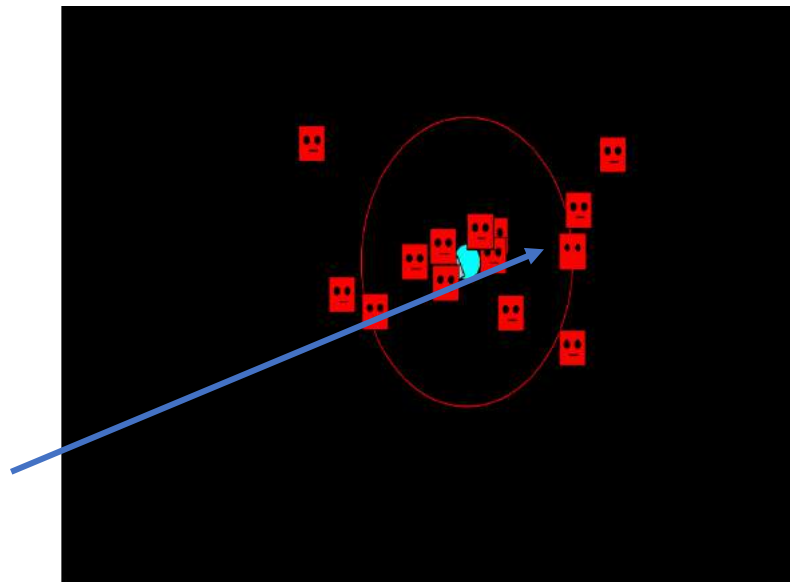
Adding expression to enemies

Looking at some previous feedback from Jayden, one criticism was that the enemies looked rather plain and boring as they were just squares ' rather than enemies which should have some sort of character or facial expressions to have some sort of unique characteristic. So, I will begin by focusing on adding some basic expressions for the enemies to make them seem more enemy like and lively, though, I don't expect to implement any sort of high graphics for the enemies as this will consume too much time. Knowing this, I may consider this in a later development if I have time for it, since there are higher priorities in the must section from the MUST section from the MOSCOW analysis that must be dealt with first.

```
fill("black")  
  
ellipse(this.x + 10, this.y - 5,  
10)  
  
ellipse(this.x - 10, this.y - 5,  
10)  
  
line(this.x + 10, this.y + 10  
, this.x - 5, this.y + 10)
```

To draw some eyes for the enemy I used the ellipse function to draw 2 circles on the enemy filled in black and drawing a small mouth using the line function. After messing around with the coordinates of the line and circles the enemies now have a simple face and expression.

There is one slight bug here where just one of the enemies will be drawn without a mouth and just the eyes, I'm not too sure why this has happened but I suspect this may be due to something being drawn over the mouth of that enemy. Since it is a small bug and not impactful for the development of the game I will not spend time any further time to fix this bug since I will not have time to implement the essential features of the game, though, at a later development I will attempt to fix this later.



Projectile Creation

In my prototyping, I didn't manage to get far in the creation of projectiles – I only managed to detect once an enemy was in range of the tower and then create a projectile at my tower cannon, my prototype was not able to fire towards each enemy. My current idea is to create a cannon that will automatically fire towards each enemies once they are in range of the tower, that being the circle, this would be balanced by making the cannon fire at a fixed interval which will be tested by my stakeholders for balancing. This section of the code is very important as the enemies being shot at is the main basis of the game.

How the automatic tower should work is through this process:

Tower

- If there is an enemy is within range of the tower and a cooldown for the tower to fire has passed.
- Create a projectile at the towers position
- If an enemy is being targeted by the tower
- Projectile will progress towards the nearest enemy
- The tower will fire this projectile every x number of seconds through a cooldown
- The tower will only be able to fire once the cooldown is finished.

Projectiles

- If the projectile collides with an enemy
- Remove the specific projectile that has hit an enemy from projectiles array
- Remove specific enemy that has been hit by a projectile from enemies' array

Money

- If an enemy has been "killed" as it has been hit
- Give the player money depending on the enemy killed
- Update the players current money count

Firstly, I had to detect an enemy being "in range" of the tower so then the tower can create a projectile, then shoot in that enemy's direction. In order to do this, to detect the enemy being in range of the tower, I will make use of the p5 collide library (bmoren, 2022) to accomplish this.

```
function inRange() {
    //loops through enemy array
    for (let i = 0; i < enemys.length;
i++) {
        //if enemy is in range, return
hit as true
        let inRange =
collideRectCircle(enemys[i].x,
enemys[i].y, 50, 50, windowWidth / 2,
windowHeight / 2, 400);

        //checks if enemy is in range
        if (inRange) {[]} &&
projectiles.length < 1) {
            //fire the projectile towards
enemy

            projectiles.push(new
Projectile(towerCannon[0], enemys[i]))
        }
    }
}
```

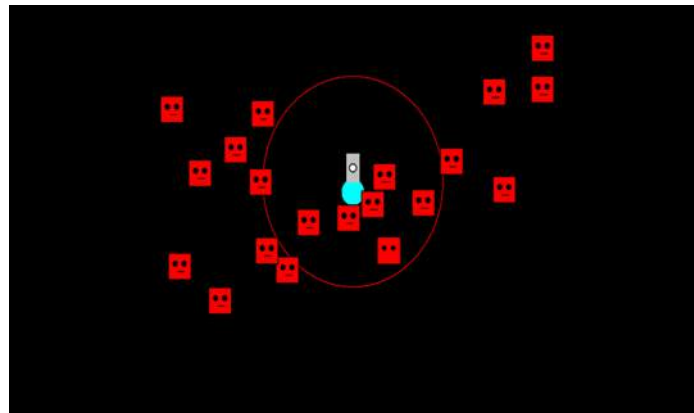
I created the function in range, which purpose is to detect if an enemy is in range of the tower and if an enemy is in range, then a projectile will be created.

Firstly looping through the enemies' array will allow a projectile to be created every time an enemy is within range of the tower.

Then, comparing each enemies coordinates with the tower's radius using p5's collideRectCircle function which will output true if an enemy is within the range of the tower.

Once an enemy is within range of the tower, then a projectile can be drawn on the towers cannon.

When attempting this, the game slowed down significantly. When checking the console I looked at the projectiles array and it was clear that the slowdown of the game was due to a very large - thousands of projectiles being created every frame since the enemies once in range was always setting the flag inRange to true which as a result created thousands of projectiles per second, thus lagging the game.




```
if (inRange && projectiles.length < enemys.length)
```

To try and limit the number of projectiles drawn that was causing the slowing of the game, I modified the condition to only allow a number of projectiles to be created based on the current enemies created. Whilst this did limit the amount of projectiles created, the projectiles still didn't move towards the enemy.

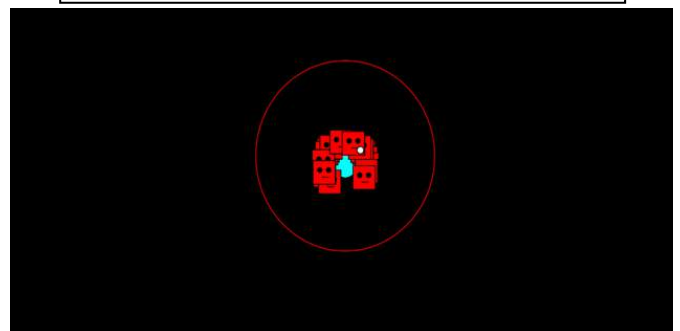
```
update() {
  // loops through enemy array
  for (let i = 0; i < enemys.length; i++) {
    this.DistX = enemys[i].x - this.x;
    // calculates y distance from projectile to enemy
    this.DistY = enemys[i].y - this.y;
    //find angle between projectile and enemy
    this.angle = Math.atan2(this.DistY, this.DistX)
    //finds x angle projectile needs to move
    this.speedX = Math.cos(this.angle);
    //finds y angle projectile needs to move
    this.speedY = Math.sin(this.angle);
    //moves projectile's x coordinate based on the speed
    this.x += this.angle * this.speed
    //moves projectile's y coordinate based on the speed
    this.y += this.angle * this.speed
  }
}
```

I began by trying to use Pythagoras Theorem – modifying previous code from my enemies to suit the projectiles since the enemies' code had a similar property where they would make their way to the tower so I thought this would be the same for the projectiles.

Once looping through the enemies' array, the x and y distance is found by subtracting the coordinates away from the projectile's coordinates.

Then finding the angle by using the math.atan2 function so the projectile can move in that direction.

Lastly updating the x and y coordinates of the projectile to move in that direction



This caused all the projectiles to move continuously and flicker back and forth to each enemy, I suspect this may probably be due the projectile not knowing which enemy to go towards and so tries to make its way towards each enemy, which it fails to do so since each enemy are in different coordinates and direction.

At this point I was pretty unsure on how to progress in the development in the projectiles so I asked one of my friends who has experience in creating projectiles on what I could possibly do to solve this issue. He suggested when making projectiles, making use of p5's vector function would be useful in this circumstance but I wasn't too sure on how the vectors worked, so I did some research before applying this. According to my research, a vector is an object that has 2 properties : a magnitude (speed) and a direction, this being evidently useful for my projectiles. After reading p5's create vector function worked (Qianqian Ye, 2022) I tried to attempt using this.

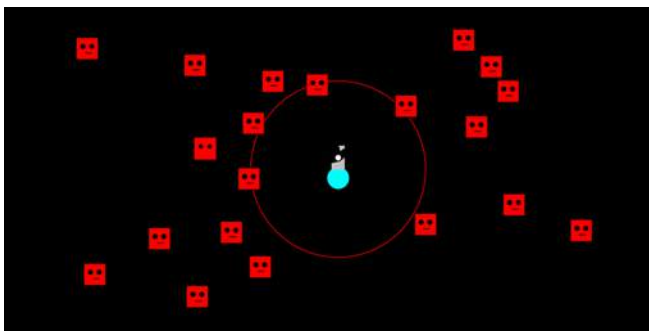
```
class Projectile {  
  constructor(src,tgt)  
    this.speed = 1
```

```
projectiles.push(new  
Projectile(towerCannon[0],  
enemys[i]))
```

```
this.pos=createVector(src.x,s  
rc.y)  
  
this.bv=createVector(tgt.x-  
src.x,tgt.y-src.y)  
  
this.bv.setMag(this.speed)
```

```
ellipse(this.pos.x, this.pos.y, 15)
```

```
update() {  
  this.pos.add(this.bv)  
}
```



To work out the position each projectile needs to move, I would need the coordinates of the tower and the target, that being the enemy. To get the coordinates in the constructor I would just have to parse in the coordinates of the enemy and the tower when the projectile is created in the game.

This.speed will also be useful in determining how quickly the projectile will move.

Then using the p5's createVector, I stored the current x and y position of the projectile with this.pos which once in the update function will be increased by the vector it needs to increase by.

This.bv (bullet vector) – will store the direction each projectile will have to move in order to make their way towards each enemy.

Also I set the projectiles speed by using p5's setMag so the projectile would move towards the enemy at an appropriate speed.

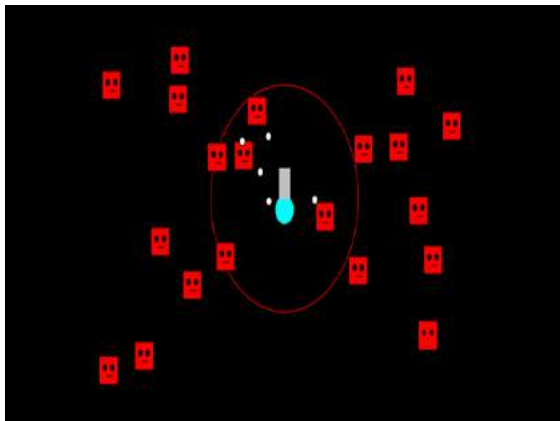
I also had to update the draw function of the projectile so the projectile position will update continuously once an enemy has been detected to be in range.

In the update function for the projectile using .add will add onto the projectile current coordinate and move it in the direction in accordance to the enemy it is targeting.

When removing the restriction on the number of projectiles created, it seemed the projectile were successfully being created once the enemies were in range of the tower and were also moving towards in the right direction. The only problem now being too many projectiles were being created.

Limiting the projectiles created

```
if (inRange &&  
    !enemys[i].targeted) {  
    //fire the projectile  
    towards enemy  
    enemys[i].targeted = true  
    projectiles.push(new  
    Projectile(towerCannon[0],  
    enemys[i]))  
}
```



In order to limit the number of projectiles created to a projectile per enemy I would have to add another condition where if an enemy has not been targeted by the tower yet and the enemy is in range, then a projectile can be drawn for that enemy.

Using !enemys[i].targeted will allow the specific enemy to be targeted and so a projectile can be created for that specific enemy.

Once the enemy has been targeted, they will no longer be targeted by the tower, though if I were to add health for enemies I would have to remove the flag so the tower will keep firing at the enemy until it has been killed.

When trying this out it was successful - only enemies in range would have a projectile move towards them. However, this is rather unbalanced since there is no cooldown when enemies are being shot at. So I will need to add a set cooldown for the projectiles being shot and then, projectiles need to be able to collide with the enemies since currently they just pass through the enemies which doesn't do anything for the game.

Version 0.5

Projectile Collision

When enemies are in range of the tower, a projectile is now fired from the tower and move towards the nearest enemy. Once the projectile collides with the enemy, then the enemy will be removed from the canvas. Since I haven't implemented any damage system for the projectile fired and adding the attribute of health for each enemy, for now, each enemy will be "killed" from colliding with a single projectile and when this projectile hits an enemy it will also be removed as only that projectile colliding and a projectile colliding with multiple enemies at once is rather unbalanced from a game balancing perspective.

Firstly, there needs to be a function :

- if any projectiles are currently colliding with any existing enemies on the canvas
- then remove that enemy that is colliding
- Also remove the projectile that is colliding itself.

```
hasHitEnemy() {
  for (let i = 0; i <
enemys.length; i++) {
    let enemyHit =
collideRectCircle(enemys[i].x
, enemys[i].y, 50, 50,
this.pos.x, this.pos.y, 15)
```

To begin with, I declared a new function within the tower.js to detect if an enemy and a projectile is colliding with each other. First of all, to check if any existing enemies are colliding with a projectile, I would have to loop through the enemies' array. Then, to check if these enemies are colliding with a projectile, using p5's collide library (bmoren, 2022) function collideRectCircle, that will output true if a circle, the projectile and a rectangle the enemies are touching.

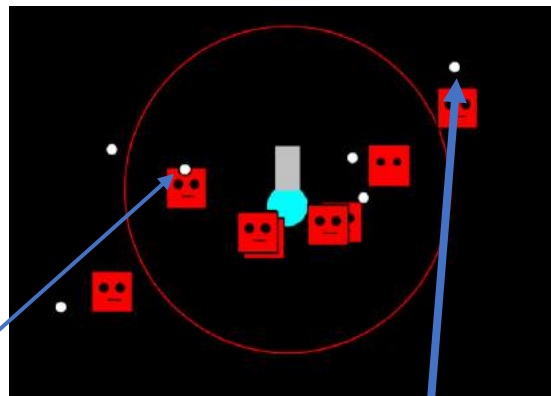
```
if (enemyHit) {
  console.log("colliding?
", enemyHit);
  enemys.splice(i , 1)
  projectiles.splice(proj
ectiles.indexOf(this), 1)
  return true
}
}
return false
```

Then, if an enemy has been detected to have collided with a projectile, the enemys.splice() code will remove the specific enemy from the enemies' array that has collided with a projectile. To find the individual projectile that has collided with the enemy and using indexOf will return the first index at which a given element can be found in the projectiles array allowing the individual projectile to be removed rather than all the projectiles at once

```
for (p of projectiles) {
  p.hasHitEnemy()
```

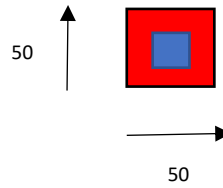
Then to enable this function to check if a projectile is colliding with another enemy only when there is an existing projectile within the projectiles array as it is only needed when a projectile and enemy exists within their respected array.

Unfortunately, when attempting to run this there were a 2 issues I encountered. Projectiles were killing the enemies as they would be removed as well as the projectiles in their arrays, however, for some reason, collision would only be detected if the projectile had collided with the centre of the enemies and not when colliding with the sides of the enemies as shown in the image. I suspect this is due to the enemies (which are drawn as a rectangle) are drawn in centre mode by default, therefore, collision would only be detected if a projectile had hit the exact centre of each enemy rather than the corner of an enemy.

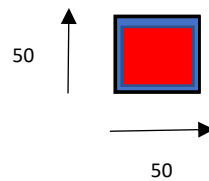


There was one other issue I encountered : quite simply, some enemies wouldn't be targeted by the tower for some reason and some would, leading to other enemies being shot at and some not which is completely unintentional.

To solve the issue of the projectiles not hitting the enemies, I could just redraw the enemies in corner mode `rectMode(CORNER)` so the collision between the projectiles and enemies will be more accurate since this would enable every enemy to have a larger hitbox to hit, but this would require repositioning of the enemies faces again, and so instead of going through this lengthy process again I decided figured out I could just add an offset to the current collision detection .



Current area of enemies where collision can be detected since enemies are drawn in `rectMode` centre, this hitbox of the enemy is rather limited causing the error of the tower missing enemies.

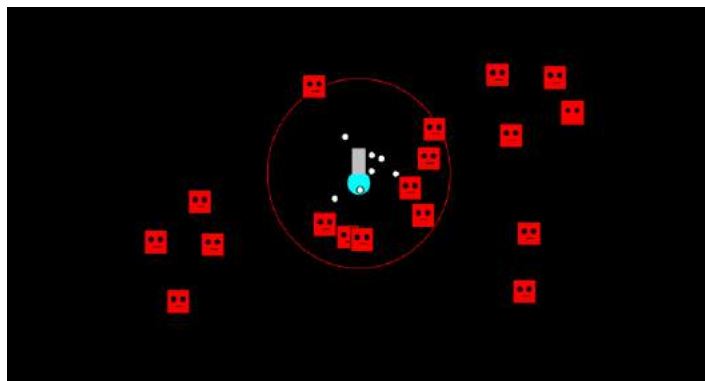


Collision when adding an offset will now output true when a projectile hits the side of the enemy rather than just the centre of the enemy now this will make the tower a lot more accurate now since the enemy has a much wider hitbox to hit .

Since each enemy had a width and height of 50, then if I subtracted 25, half of the enemy, then collision could be detected at the corner of each enemy rather than the center of the enemy.

```
let enemyHit =
  collideRectCircle(enemys[i].x -
    25, enemys[i].y -25, 50, 50,
    this.pos.x, this.pos.y, 15)
```

This worked reasonably well, whenever a projectile had collided with the edge of each enemy, they would now collide with the enemy and remove both itself and them from their arrays . Now there were 2 more issues: one issue was that some enemies were just not being targeted by the tower, another issue being the tower was clearly shooting a bit too quickly and so some sort of cooldown must be implemented .



Enemy targeting

Testing the tower firing a few more times, every single enemy would be targeted and would be hit except from a single enemy, so I thought that perhaps the second enemy was just being skipped for some reason. It came to me as no surprise as when I went through the targeting of the enemies, I understood why an enemy was being missed, it was due to the current for loop used when targeting each enemy.

```
for (let i = 0; i <
enemys.length; i++) {
}
```

As of currently, I thought when targeting each enemy, I could just normally loop through the enemies' array to ensure every enemy will be targeted by the tower. Unfortunately this was quite wrong since enemies could be killed.

As for the reason why this loop didn't work as caused the error that led to an enemy being not targeted by the tower is due to the loop skipping out the enemy since the loop has already passed the enemy in the array without checking it. As shown in a demonstration below :

To begin with if there were 3 enemies in the enemy array, then the first enemy will have the position 0 in the array and the second enemy has the position 1 in the array.

Example : Before enemy killed

Enemies array = enemy1 , enemy2, enemy3

Enemy 1 will have the position enemies[0] in the array

Enemy2 will have the position enemies[1] in the array

After enemy killed

Enemies array = enemy2, enemy3

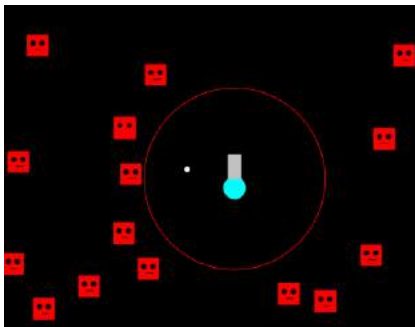
Enemy 2 will now have the position enemies[0] in the array

Enemy3 will have the position enemies[1] in the array

But if an enemy1 is killed by the tower, enemy2's position in the array will shift to position 0 in the array and as the for loop will increment by 1 when an enemy is fired at, the enemy in the position 1 in the array will be checked and not the first enemy in the array and so the enemy at position 0 in the array is skipped, leading to an enemy not being targeted and fired at, causing the issue of the tower missing out an enemy.

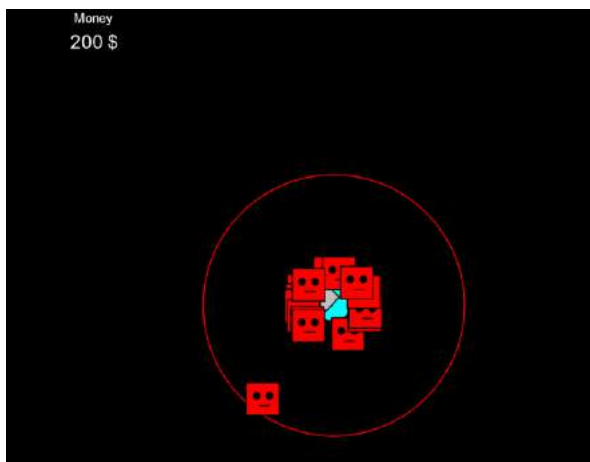
The most reasonable and simple way to fix this is to use a reverse for loop, which will ensure all the enemies will be checked as the last enemy that is drawn is checked first rather than the first enemy, this will make sure no enemies will be skipped when being targeted and shot at by the tower.

```
for (let i = enemys.length - 1; i  
>= 0; i--)
```



```
var money = 0
```

```
function drawGame() {  
  text("Money", 400, 20)  
  textSize(30)  
  text(money, 390, 60)  
  text("$", 430, 60)  
}
```



After making some modifications to for loop to a reverse for loop, the tower would now work correctly as it would now shoot a single projectile for each enemy after every 1 second had passed.

I first declared a new variable money, which will store the players current number of money they have gained. By default, this value will begin at 0 at the start of each new game.

So now when an enemy is killed by the tower the amount of money the player has will increase by 100, this number will update and will be displayed on the game screen. After adding some additional text, money and then displaying the variable money as some text, this money text will automatically update when the player gains some money.

Cooldown

The tower currently rapidly shoots every single enemy as soon as an enemy enters the towers range, so in order to have some balancing in the game and make the game somewhat of a challenge, there must be a cooldown whenever the tower shoots an enemy within range.

In order for this to happen :

- A set cooldown for the tower will hold an interval at which the tower can shoot.
- When the tower shoots at an enemy, the time that has passed since the tower has last shot will be stored
- If the current time is longer than the cooldown and time since the tower has last shot
- The tower can shoot a projectile at an enemy
- If not, the tower cannot shoot until the time since the tower has last fired has exceeded the cooldown

To start off with, I had no clue on how to store the timings, and so I looked on the p5 reference for help on this (McCarthy Q. Y., 2022), after looking around on the website for a while, I came across the section time and date which had functions that could store timings, which is exactly what I needed. There were 2 functions that seemed to be useful my current game cooldown : second () and millis() which will return the number of seconds and milliseconds since an event has occurred. I decided using millis() would be more appropriate here as I can have a more precise timing when shooting the tower as increasing in seconds could make the cooldown balancing a bit too difficult as it is not as precise as milliseconds .

```
var lastFired = 0
```

To begin with, we can assume the tower hasn't shot an enemy in the beginning of the game, and so I declared a variable lastFired that will be by default set to 0 since the tower hasn't shot at an enemy yet.

```
var cooldown = 1000
```

Continuing this, I created a variable that will store the amount of time that must pass until the tower can fire at another enemy again, since the p5 function used milliseconds, I thought the tower shooting every 1 second would be a reasonable speed for the tower to shoot at - this would be 1000 in milliseconds.

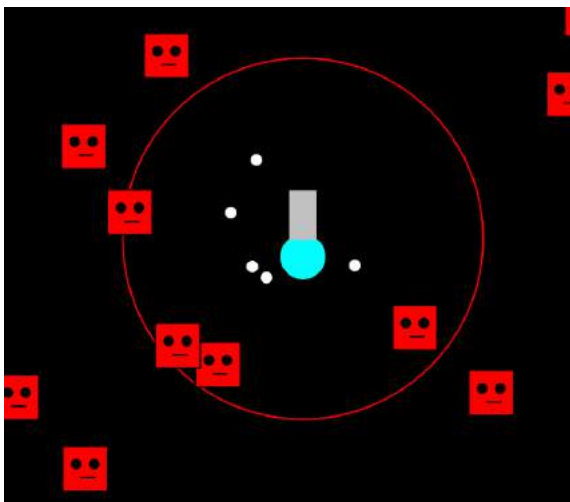
```
projectiles.push(new  
Projectile(towers[0],  
enemys[i]))
```

```
// note the time this  
happened, and exit the  
function
```

Then, when the tower detects and fires a projectile for an enemy, the time at which the projectile was created will be stored in the variable, `lastFired`, which will hold the amount of time since the projectile was created.

```
if (millis() > lastFired + cooldown)
```

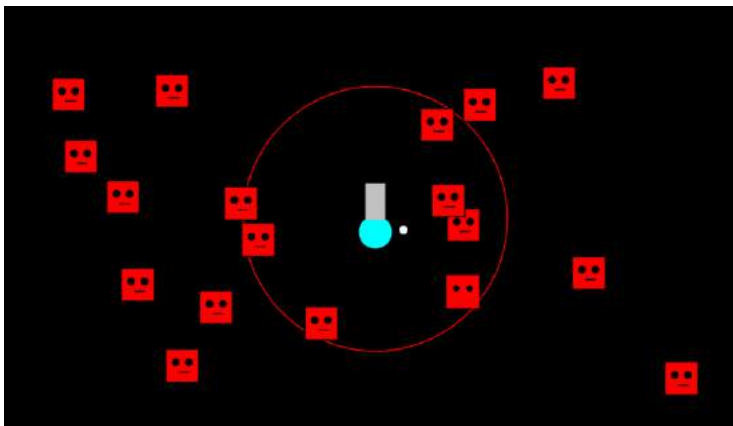
After that, once a projectile has been shot for an enemy, the process needs to be repeated for every single time an enemy is within range of the tower. For this to happen, using `return` will exit the function and then continuously check if the tower can fire at an enemy again.



The results of this clearly didn't work, the projectiles being fired did seem a little different from before, as projectiles didn't fire all at once but rather in groups, meaning that the cooldown did work, but not for each individual projectile. So to solve this, I would have to find out a way to make it so this cooldown will happen but only for each individual projectile. I wasn't too sure how to enable this so I searched on the internet on how this could be achieved. After some time through searching, I came across the JavaScript use of the code `return` on (W3Schools, 2022) which main purpose was to return a value and exit out of a function and since I check each individual projectile I thought this would be useful, and so I tried to implement this into the game.

```
return
```

Once a projectile has been fired by the tower and has stored the time from when the tower has actually fired, this value will be returned and then the function will be exited. When the function is carried out again the value of lastFired can be used again for each individual projectile .



After adding return, the testing of the tower was successful as the tower successfully shot at each enemy one at a time after 1 second has passed and will eventually shoot all the enemies over time rather than all the enemies at once immediately.

After a discussion on the tower shooting automatically, me and Jayden concluded that since the tower would be shooting the enemies automatically, the game would be rather boring, since the player would not really be engaging in the game as the tower would shoot all the enemies for them, and so the only thing the player could do was buy upgrades to which hasn't even been implemented at this point. Therefore, to make the game more productive and entertaining, I decided that the tower should be shot manually by the player using their mouse and the option for shooting automatically can be just used as an upgrade. This allows the player to actually play the game rather than just click some buttons and watch their tower do all the work for them.

The requirement for this needs :

- The tower cannot rotate towards the cursor's current position
- When the player touches the screen, a projectile will be created
- The projectile will move towards the position of where the screen was pressed
- Projectiles can only be created by the player every x seconds
- If the projectile fired collides with an enemy then remove both projectile and enemy from their respective array
- Reward the player with x amount of money

```
draw() {  
    fill("cyan");  
    stroke("black")  
    circle(windowWidth / 2,  
windowHeight / 2, 50);  
    fill("silver")  
    stroke("black")
```

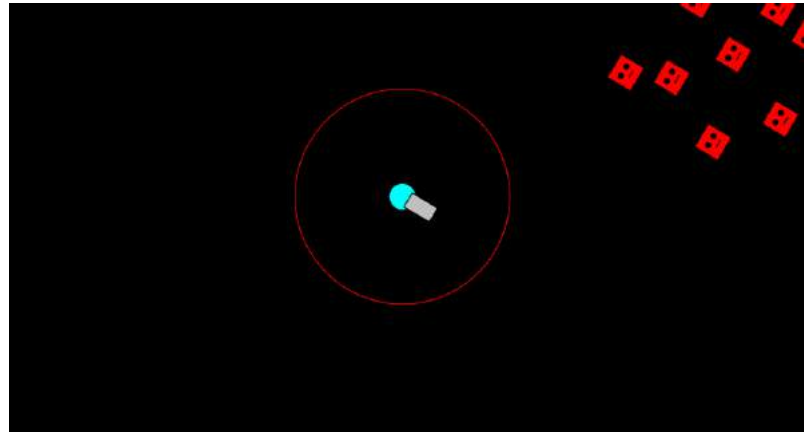
Whilst it was useful to have the tower cannon position by drawing the cannon individually from the tower as I was able to use the exact cannon's coordinates, I now wanted the cannon to move with the tower and so for this to happen I would have to draw the cannon with that tower rather than how the cannon is drawn now which is separately drawn from the tower. In order to do this, I had to remove the previous tower cannon class and code. To implement the cannon into the tower class, was quite simple as I just needed to copy whatever was in the previous draw function of the tower cannon code.

```
rectMode(CENTER)  
translate(this.x, this.y)
```

As I wanted the tower cannon to rotate around the middle of the tower, using rectMode center would achieve this as the cannon would be positioned in the middle of the tower as it rotates. I also added the p5 function translate since the tower cannon would only be rotating around the tower and nothing else, and so this was required to prevent the cannon from rotating anywhere else apart from the tower.

```
rotate(mouseX)
```

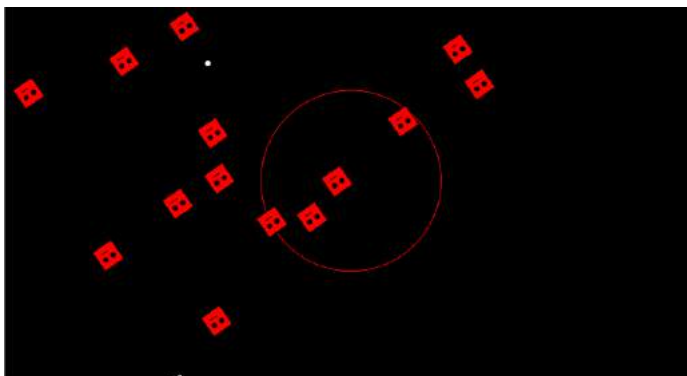
I then tried to rotate the tower based on the cursors current position, this clearly didn't work as intended though as the whole canvas rotated whenever I adjusted the mouse position rather than just the tower.



I hadn't encountered this error before and was really confused on why the whole canvas rotated with the mouse rather than just the tower, so I researched online on why this may have been the case. After some time, I came across an example (learn.digitalharbor, 2022) which completely explained why this occurred. I found out from the website that by using p5's pop() and push() methods, in simple terms will enable only the rotation to the tower as the tower will be in a new separate draw state from the canvas which was the main reason why the canvas was moving with the tower. In the example from the website, before pop and push were added, both squares would rotate rather than just the one square; after pop and push were added to an individual square, only the one square would rotate rather than both of them had their own individual drawing state.

```
pop()
  rotate(mouseX)
  fill("cyan");
  stroke("black")
  circle(windowWidth /
2, windowHeight / 2,
50);
  translate(this.x,
this.y)
  fill("silver")
  stroke("black")
  rect(0, 40, 30, 55)
push()
```

Applying pop and push to my game, when moving my mouse, the whole canvas would now move very rapidly when I moved or updated the position of my mouse, which was definitely not supposed to happen in this situation.



When looking back on the example, I realised I had made a mistake and had used pop() first instead of push, which was the main cause on why the whole canvas rotating rather than just the tower cannon as the rotate was being applied to everything on the canvas rather than just the tower cannon.

```

translate(this.x, this.y)

fill("silver")

stroke("black")

rotate(mouseX)

rect(0, 40, 30, 55)

```

After switching around pop and push, only the tower would rotate but I wanted rather only the cannon to rotate rather than the whole tower itself, to change this, I moved the rotate code to where the cannon is drawn so the rotation will only affect the cannon.

Applying this fortunately rotated only the tower cannon according to the mouse's x position, but since rotate could only accept 1 parameter, I wasn't sure how to move the cannon towards the mouse.

After some time thinking I decided that in order for the cannon to rotate towards the mouse, I would need to find the angle at which the cannon would need to rotate so it can face towards the tower, this was quite similar to the creation of the projectiles previously, and so I thought using vectors here may be useful.

```

this.pos =
createVector(this.x,
this.y)
this.mouse =
createVector(mouseX,
mouseY)

```

To find the angle at which the tower cannon would need to rotate, first I will need the position of the tower, to do this, I stored the x and y position of the tower as a vector in the variable this.pos and the x and y position of the mouse in the vector variable, this.mouse.

```

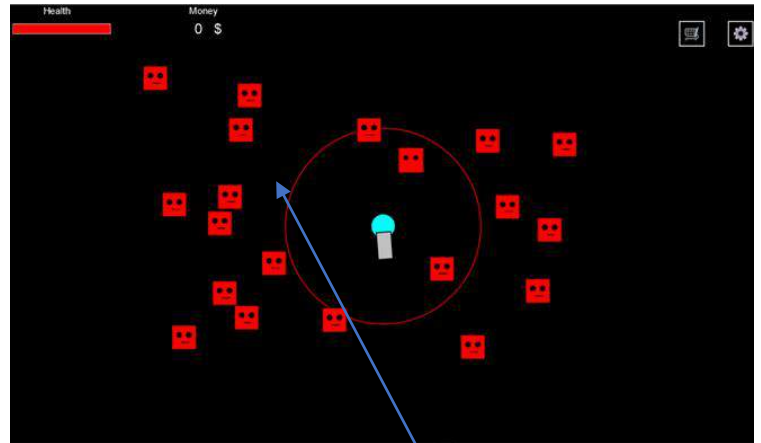
this.angle =
p5.Vector.sub(this.pos,
this.mouse).heading()

```

Then, after reading the p5 website on vectors (Qianqian Ye, 2022) I found that by using 2 vectors it was possible to calculate the angle of these 2 vectors by first subtracting both vectors using .sub() then using .heading() which function is to calculate an angle from the 2 vectors parsed in. I stored this angle in a new variable, this.angle. Now that I have the angle at which the cannon needs to rotate to face the mouse cursor I will just have to rotate the cannon to this angle.

```
rotate(this.angle )
rect(0, 40, 30, 55)
pop()
```

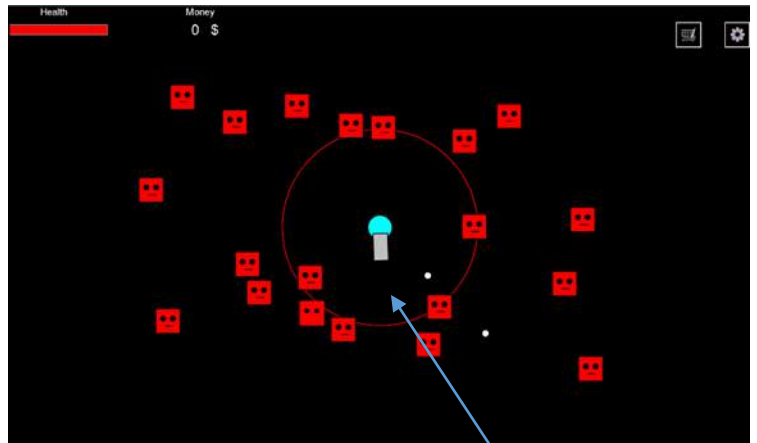
The cannon of the tower now rotates when depending on the mouse position of the players mouse, however, the cannon currently does not face the mouse and rotates away from the mouse I'm pretty sure this is due to the tower cannon not facing the mouse when being drawn in, so I would have to adjust its position to be facing the mouse when it is drawn



If the mouse position is here then the cannon rotates but not towards the mouse.

```
rotate(this.angle - 80)
```

After some trial and error of modifying the angle at which the cannon is drawn when the game starts, the tower cannon at the mouse position and so will now rotate in accordance with the players mouse position.



Current player mouse position

```
class manualProjectile {
  constructor(x, y) {
    this.x = x
    this.y = y

    draw() {
      fill("white")
      stroke("black")
      ellipse(this.x, this.y , 15)
    }
  }
}
```

As projectiles is already used for the automatic firing upgrade, I decided to make a new class for manual fired projectiles since whilst both being similar, both functions have slightly different functionality .

```
function mouseClicked() {
    manualProjectiles.push(new
manualProjectile(towers[0].x,
towers[0].y)) }

```

P5's mouseClicked function will run the function once when the mouse is clicked, this will make it so when the player presses their mouse onto the canvas, a projectile will be drawn at the tower's coordinates once.

```
class manualProjectile {
  constructor(x, y) {
    this.x = x
    this.y = y
    this.speed = 2
    this.DistX = mouseX - towers[0].x;
    // calculates y distance from tower to
    enemy
    this.DistY = mouseY - towers[0].y;
  }
}

```

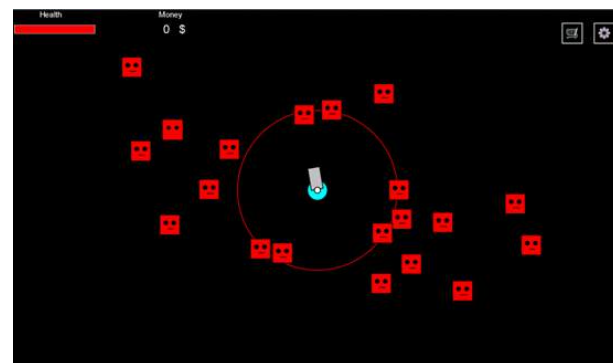
```
this.DistX = mouseX - towers[0].x;
// calculates y distance from tower and mouse
this.DistY = mouseY - towers[0].y;
//find angle between tower and mouse
this.angle = Math.atan2(this.DistY, this.DistX)

```

```
update() {
  this.speedX = Math.cos(this.angle);
  //finds y angle projectile needs to move
  this.speedY = Math.sin(this.angle);

  //moves projectile's x coordinate based on
  the speed
  this.x += this.speedX * this.speed
  //moves projectile's y coordinate based on
  the speed
  this.y += this.speedY * this.speed
}

```

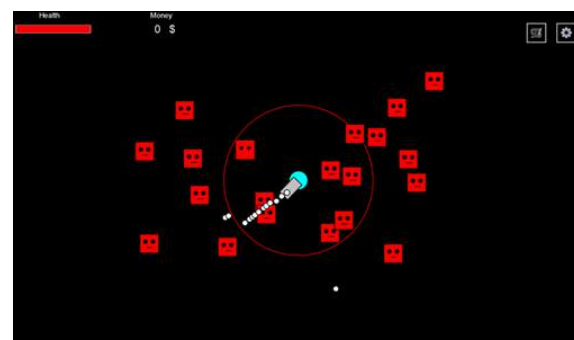


Then, in order to make the projectiles move towards the mouse, I would need to find the angle at which the projectile would have to move from the tower.

To do this, previous code from my prototype [1.4 Enemy Ai](#) can be applied here where the projectiles will move towards the mouse calculating the angle the projectile needs to move.

Firstly to find the angle the projectile needs to move the x and y distance from the projectile to the mouse's current position. Then, using Math.atan2 will calculate the angle required for the projectile to move. After that, using the cos and sin function I can calculate the x and y angle the projectile needs to move.

Then to move the projectile, the respected angles will be added onto the x and y coordinates at the speed of the projectile




```

hashitenemy() {
    for (let i = enemys.length - 1;
i >= 0; i--) {
        let enemyHit =
collideRectCircle(enemys[i].x - 25,
enemys[i].y - 25, 50, 50, this.x,
this.y, 15)

        if (enemyHit) {
            enemys.splice(i, 1)
            manualProjectiles.splice(man
ualProjectiles.indexOf(this), 1)
            money += 100        }
        }
    }
    return false
}

```

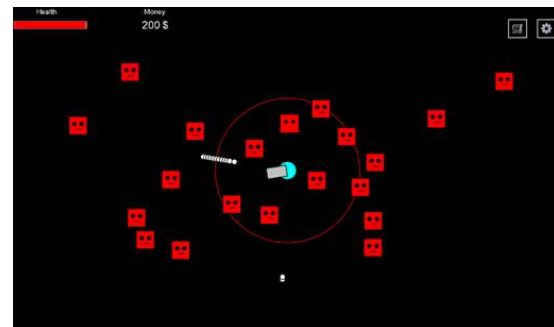
Since the manual projectile will also kill the enemies when touching them rather than writing new code, borrowing some previous code from my automatic shooting cannon seemed appropriate here.

After calling the hasHitEnemy function, the manual projectiles will now be able to kill enemies when they make contact with each other.

```

for (m of manualProjectiles) {
    m.draw()
    m.update()
    m.hasHitEnemy()
}

```



```

var manualFire = 0
var manualCooldown = 500

function mouseClicked() {
    if (millis() > manualFire +
manualCooldown) {
        manualProjectiles.push(new
manualProjectile(towers[0].x,
towers[0].y))
    }
}

```

Since you could click on the screen multiple times per second, a cooldown is required to balance out the number of projectiles the player can shoot at a given interval.



Just like the automatic fire I created previously, I made a cooldown by creating a variable that stored the cooldown and the time since the last projectile had been created, if the current time has exceeded the cooldown then a new projectile can be created in which the player could shoot a projectile once every half of a second, though this seems quite quick so I may have to balance this later.

Version 0.6

Connecting the Menu

From my previous development, I was able to create a basic layout of what my menu will look like but I wasn't able to link the menu to the actual game. Luckily, my college website had an exemplar on how to achieve this by linking and switch screens using buttons (Colchsf, 2022), and so I after looking at the example thoroughly and making sure I understood how certain functions of the scene switching function, I tried to follow a similar structure to the one in the example.

```
class Button {  
    constructor(text, x, y, w, h,  
click) {  
        this.x = x  
        this.y = y  
        this.text = text  
        this.width = w  
        this.height = h  
        this.click = click  
        this.enabled = true  
    }  
}
```

For each button in the game the constructor will store all of the buttons parameters - each button will have some text written inside of the button which will be centred in the middle of the button, an x and y coordinate where the button will be positioned, a width and height which will determine how large each of the buttons will be and this.click will be used so when a button is created, a function can be pass onto it and the code with the button will run when the button is clicked by the player.

```
clicked() {  
    if (this.enabled) {  
        if (mouseX > this.x - this.width  
/ 2 && mouseX < this.x + this.width  
/ 2 && mouseY > this.y - this.height  
/ 2 && mouseY < this.y  
+ this.height / 2) {  
            // button is touched  
            this.click()  
        }  
        render()  
        rectMode(CENTER)  
        textAlign(CENTER)  
    }  
}
```

Clicked() will check to see if the players current mouse position is within the range of each button, and if this cursor is within range of the button and has been clicked by the player, the respected function can be run .

The render() method will allow the button to be drawn onto the respected game mode.

textAlign() will make sure all text passed in each button will be drawn within the center of the buttons.

Then to draw the main game itself, the game state must begin in the menu during this, the game title will be drawn, the background of the game will include the imported image as the background as previously developed in [2.0 Making a Basic Menu](#), and this will also have a start button which when this start button is pressed, will begin the game.

```
var buttons = []  
var menuButtons = []
```

First to create some buttons for the menu, I have to declare the variable menuButtons and buttons to allow new buttons to be created and added to their array.

```
var gameMode = 'menu'
```

By setting the current game mode as menu, the game will draw the menu and menu buttons as the starting screen when the player loads in the game.

```
function preload() {  
  start =  
  loadImage("background.png");  
}
```

As previously used before, the p5 preload() function will allow my imported image to be loaded into my game as the game menu background, to do this, I will have to store the background image as the variable start.

```
buttons = menuButtons  
}
```

Since the starting buttons are set to the menu buttons then when starting the game only the menu buttons will be drawn on the canvas on the menu screen.

```
function mousePressed() {  
  for (b of buttons) { b.clicked() }  
}
```

P5's mousePressed() function will run the code once every single time the mouse or cursor from the player is pressed once. The for loop will loop through the buttons array and constantly check if the buttons on the screen have been pressed, if they have been pressed, then the function related to the button will be carried out.

```
for (b of buttons) {  
  b.render()  
}
```

As for each button, b.render() will make sure the buttons are drawn correctly on the screen and the functions are carried out

```
function draw() {
  switch (gameMode) {
    case 'menu':
      drawMenu()
      drawTitle()
      break
  }
}
```

The switch function will allow the screens to be drawn over the other screen and basically allows the transitioning of the menu to the game.

When the game state is current in the menu (this being the default game mode at the start of the game) the menu screen will be drawn as well as the title.

```
function drawMenu() {
  background(start)
}
```

I decided to draw the menu background and the title separately as the title and background have 2 different purposes when being carried out .

```
function drawTitle() {
  fill("gold")
  textSize(100)
  stroke("black")
}
```

```
menuButtons = [
  new Button("Start", windowWidth
/ 2, windowHeight / 2 + 100, 250,
100, () => { gameMode = 'play';
buttons = gameButtons;
background(50) })),
-
]
```

Since the buttons being drawn at the start of the game will be the menu button I will now just need to create a button to press.

I appropriately named the button start as most games have a start button to begin the game and after some fiddling with the positioning of the menu button, start I was happy enough with the appearance of the button.



After that once the start button is pressed the game mode will be set to play, I implemented the code of the functions gameMode = "play" and buttons = gameButtons this is used to change the game mode from the menu to the game and will draw the game and game buttons when the start button is pressed. When the start button is pressed, the game will be drawn and the game will start through the drawGame () function . During this, the tower and enemies will be drawn through the setup function, though this may need to be changed later since the setup function is only called once and if I wanted to make a reset button the creation of enemies will have to have an individual function that can be called multiple times.

```
new Button("Start", windowWidth
/ 2, windowHeight / 2 + 100,
250, 100, () => { gameMode =
'play'; buttons = gameButtons;
background(50) })),
```

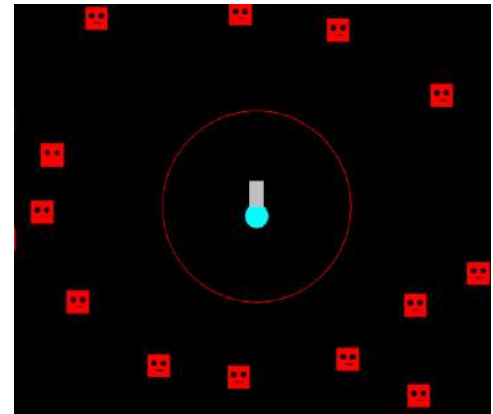
```
case 'play':  
    drawGame()  
    break
```

Once the game mode is set to play, the game will be drawn once, this will happen when the start button is pressed, and when the game mode switches from menu to play, the menu screen will be replaced by the game screen.

```
function drawGame() {  
    background("black");  
}
```

Just as before the game background will be drawn in black since it is simple and not too flashy for the player to stare at for long periods of time.

Testing this out had no issues, when clicking the start button, the game will immediately be drawn and will load and start correctly, enemies will correctly make their way towards the tower and the tower will be able to shoot each enemy successfully.



Making a pause Screen

For the designing of the pause screen, I wanted to have 2 buttons, a quit button, that will redirect the player back to the menu and a continue button in case the player just in case the player wanted to continue the game this is the most common 2 buttons within a settings screen in most games I have played and seen so I intend to add these first. I also intend the pause screen to have a transparent background for aesthetic and functionality purposes as transparent backgrounds are quite frequently used in tower defence games as it allows the player to acknowledge the game is paused through the sudden background change but also be able to see the current state of the game whilst the game is paused which can allow the player to prepare for upcoming enemies.

The pause screen will work as followed:

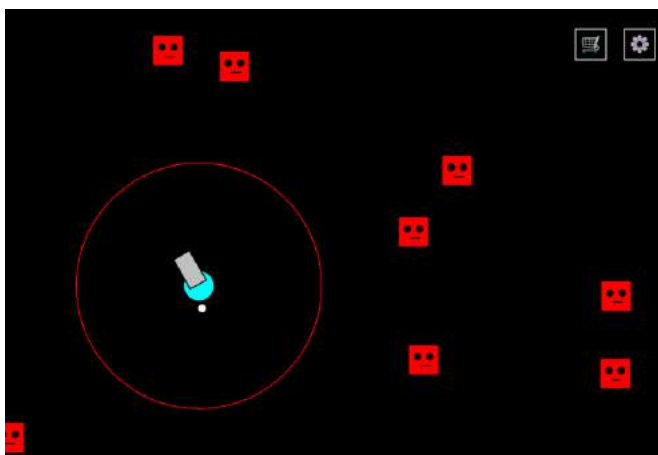
- If the pause button is pressed
- The pause screen will be drawn over the game screen and pause the game
- The pause screen will be a small menu positioned in the middle of the screen
- The background will be transparent for the player when in the menu.
- When the game is paused, there will be 2 options : to continue the game or quit the game
- Quitting the game will reset the game and redirect the player to the menu
- Continue will switch the game back to play mode and the game will continue for the player
- The game will be paused until one of these buttons are pressed

```
var gameButtons = []
```

Firstly I declared a new variable that will hold all the game buttons that will be drawn as soon as the game starts.

```
gameButtons =  
[new Button('⚙️', 1500, 60, 50, 50
```

Then I created a new game button, with a cog icon since most games have the settings button as this. After some positioning and size adjusting



When the settings button is pressed, the game will be paused, and the pause menu and buttons will be drawn.

```
() => { gameMode = 'pause'; buttons =  
pauseButtons },
```



```
var pauseButtons = []
```

For the pause menu I had to create a new variable, pause buttons which will hold all the buttons that are relevant to pause screen in the pause buttons array.

```
pauseButtons = [  
  new Button("Continue", windowWidth / 2,  
    windowHeight / 2, 250, 50, () => { gameMode  
    = 'play'; buttons = gameButtons }),  
  new Button('Quit', windowWidth / 2,  
    windowHeight / 2 + 100, 250, 50, () => {  
    gameMode = 'menu'; buttons = menuButtons })
```

During the time the pause menu state is triggered, the pause buttons will be drawn onto the screen alongside the pause menu itself.

When the continue button is pressed by player, the game state will return to play and the game buttons will be reapplied to the canvas.

If the quit button is pressed, the player will be redirected to the menu and so the game mode will be set to menu and the relevant menu buttons will be drawn.

Resetting the game

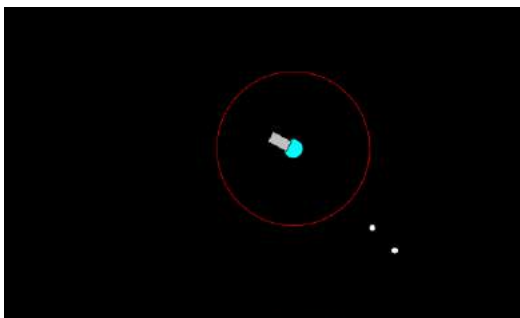
```
function reset() {  
  enemys.length = 0  
  projectiles.length = 0  
  manualProjectiles.length =  
  0  
}
```

The continue button already works as intended when clicked as the game continues as clicked, but when the quit button is pressed and the player is redirected to the menu, when the player starts a new game, they do not start with a new game and rather the previous one as enemies will be in the same position as the last game.

In order to fix this problem, I would need to create a new reset function that will reset all the global variables to their default values.

```
function drawMenu() {  
  //when player is in menu reset  
  reset()  
}
```

Then when the player has quit the game and is in the menu, the game can reset .



When applying this, all the projectile created were completely removed from the previous game, however for some reason when starting a new game now, enemies would not be created.

I suspected the enemies are not drawn was due to a problem with the spawn function which enables the enemies to be drawn onto the canvas when the game is started.

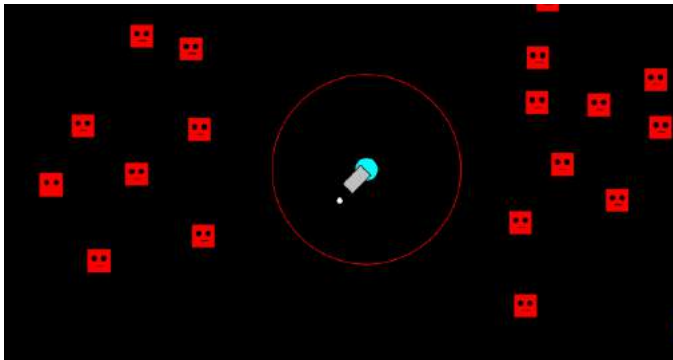
```
function setup() {  
  spawn()  
}
```

Not too long after I realised the problem and cause on why my enemies weren't being reset when the player left the game.

This was due to me calling the spawn() function that drew the enemies in the setup() function which will only be used once when the game starts and then will not be called again.

```
function drawGame() {  
  spawn()  
}
```

To fix this issue I just had to create these enemies once the game has actually started.



Now enemies will successfully reset every single time the player leaves the game and when creating a new game, a new set of enemies will be created and all the pause buttons will work as intended when pressed by the player.

```
case 'pause':
    drawPause()
    break
```

When the game state is in pause, then the pause screen can be drawn.



```
function drawPause() {
    fill("white")
    text("Paused", windowWidth /
2, windowHeight / 2 - 125)
```

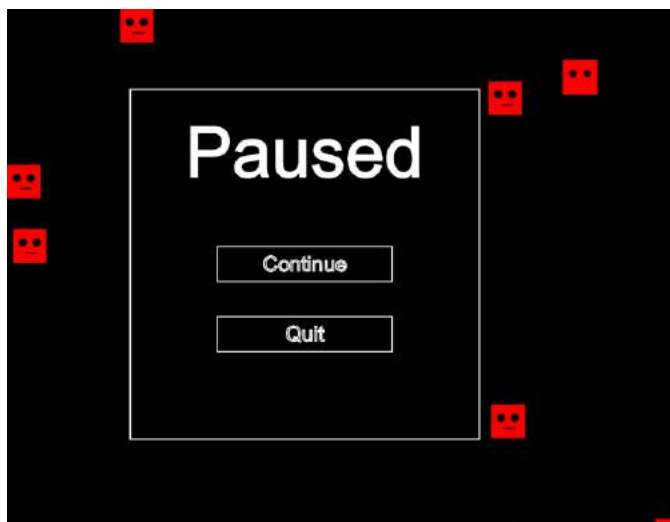
To begin the menu, I created the text paused over the pause buttons and filled it in white since it was a clear neutral colour that could easily be seen in the game.



```
textSize(100)
```

The pause text was a little too small to read though, so I increased the size of the text using `textSize()` and then adjusting the size until I thought the text size looked reasonable enough to see

After some trial and error when drawing the box around the pause buttons, I achieved a satisfactory pause menu that will pause the game and have the options to continue or quit the



```
fill("white")
fill("black")
rect(windowWidth / 2,
windowHeight / 2, 500, 500)
textSize(100)
fill("white")
text("Paused", windowWidth / 2,
windowHeight / 2 - 125)
```

For the transparent background for the pause menu, I wasn't sure how I could achieve this, so I looked for information on the p5 website. When looking at the background section (McCarthy Q. Y., 2022) there were a wide variety of colours I could apply to the background. After reading how colour is applied to each background, I learnt that I could determine the opacity of the background by adjusting the fourth parameter when drawing the background .

```
function drawPause() {  
  background(255, 255, 255, 0.6)
```

When the pause screen is paused, the background will be transparent. To make this work, I just have to set the background to white, and in the fourth parameter of background, I can adjust the opacity to the amount of transparency I want the background to look like

The game can now be seen from the pause menu through the transparent background. When applying the background, the transparent background didn't apply immediately but gradually faded in which I completely did not expect but added a really nice addition of a fading background. Jayden really liked the pause screen having a background that gradually faded as it looked "aesthetically pleasing". This unintentional addition seemed quite nice so I decided to keep in this background feature as it didn't affect the game and looked nice in general.



As far as I am aware, I understood that the cause of background gradually fading was due the background being drawn multiple times over itself, causing the background to appear as if it was gradually fading in causing a nice unintentional animation to occur.

Version 0.7

Health Bar System

In most tower defence games the player will lose when their health reaches below 0, so in my game, when the enemies are touching the tower, then the player should lose a set amount of health based on the enemies' damage, this will be indicated by a health bar for the player.

The health bar should keep track on the maximum amount of health the player currently can have and the current health the player is on. The health bar should be drawn a set size and decrease according to the amount of health the player has left. Once a health system has been implemented – if the players current health is below 0, then a losing screen can be implemented, as the player needs to actually lose the game in order for it to be a tower defence game.

There will be 2 aspects of the health bar: the appearance and the functionality:

Appearance:

- Health bar will be filled in red since by default, health is distinguished by red
- The health bar will be drawn in the top left corner since it won't block out most of the game
- Health bar will be drawn as a small rectangle bar with a reasonable size that won't take up too much of the actual game screen.

Functionality:

- When an enemy deals damage to the tower
- Update current health
- Health bar will reduce depending on the amount of health the player has left
- As the player loses health, replace the part of the health bar with black.
- If the player has no more health left
- End the game and switch to the losing screen

When researching how a health bar could be implemented I came across a very useful example on p5 (manno, 2022) which had a health bar that would have a starting health and would increase gradually at a set rate until it reached the end of the maximum health, this would be indicated by a bar filled in with red. The health bar had a set size and would update according to the current health. So technically I would have to take a similar approach to this example but instead of increasing the current health, I would have to decrease the current health if the enemies are touching the tower. Since I have collision with the tower already, this shouldn't be too difficult to decrease the health once the enemies are touching the tower.

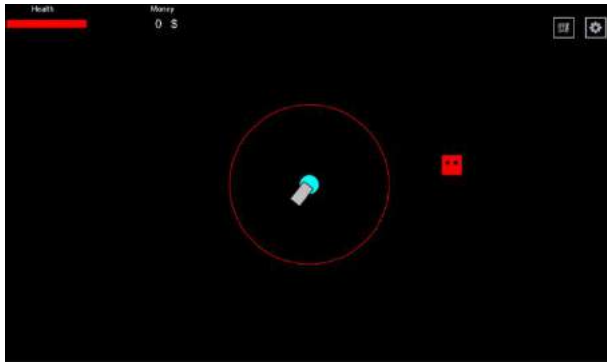
```
var health = 400;  
var maxHealth = 400;
```

First, I have to declare how much health my player will have when starting the game , in this case I thought 400 would be an appropriate amount since enemies the ramping of the enemies spawn rate will increase it will be essential for the player to have a reasonable starting health .

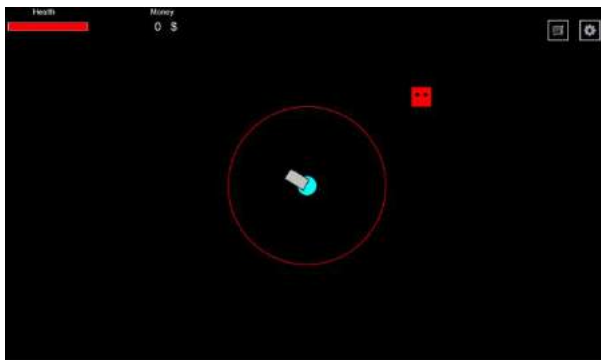
```
text("Health", 100, 20)  
rect(10, 40, 200, 20);  
noStroke();  
fill("red");  
rect(10, 40, map(health,  
0, maxHealth, 0, 200),  
20);
```

Then added to my game code, I applied some additional text "health" to where my health bar would be since it would indicate that the UI is a health bar if some players didn't know.

To draw the health bar I drew a rectangle that will be filled in red based on the player health. Using the map function as used in the p5 example, I learnt that this limits the values at which the health bar can be filled in red as. By using this function, the health bar will be drawn a set size and also decrease based on the amount of health the player has left as the bar can be filled to the maximum when the players health is equal to 400 and will fill in the health bar in red based on the players current health



The health bar would now be drawn onto the game screen though there wasn't an outline for it since the background is black.



```
stroke("white");
strokeWeight(2);
noFill();
```

Adding a stroke and a stroke weight allowed the health bar to be seen more easily, and also using no fill made sure only the outline of the health bar would be filled in white and not the health bar itself

```
this.damage = 1
```

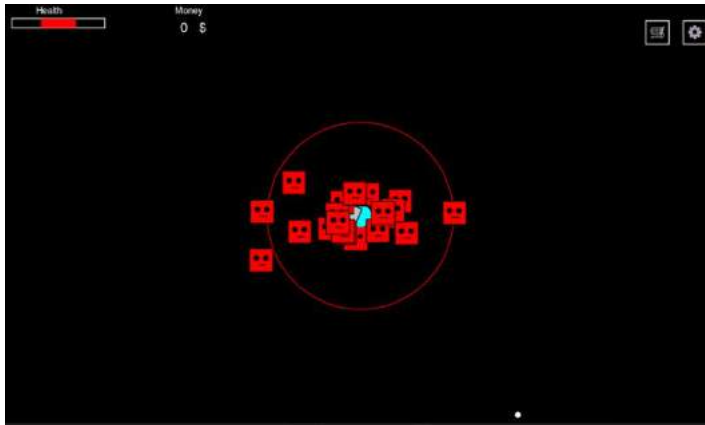
To make the health bar decrease, I would have to decrease the player's health when an enemy is detected to be touching the tower. Since I had already coded the enemies to stop when they touch the tower I could just implement a damage system here. For now each enemy will do 1 damage to the player – this could adjust later when a certain amount of enemies is on the canvas but will require testing.

```
if (towercolliding) {
  //stop
  this.speed = 0.00001
  health -= this.damage
}
```

Since I had already developed collision between enemies and the tower in [1.7 Tower Overlapping](#), this shouldn't be too difficult here.

Then once an enemy is colliding with the tower, the player's health will reduce by the enemy's damage.

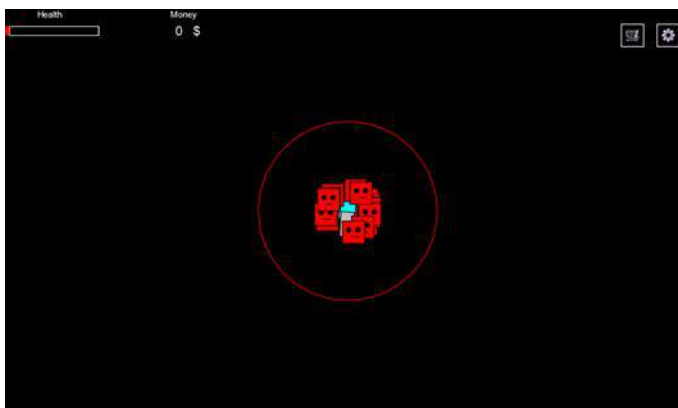
And instead of the enemy not moving when it reaches the tower, the enemy will move but very slowly so the tower will technically be colliding with the tower so this damage can occur.



The health bar would decrease but instead of moving right to left it would decrease to the centre of the square.

```
rectMode(CORNER);
```

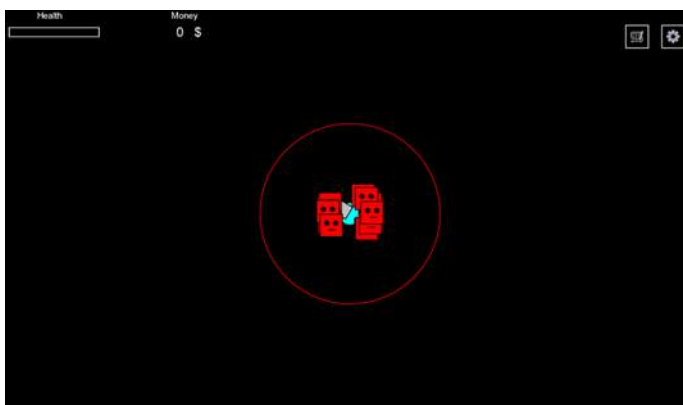
After some time looking at the p5 map and rect function I found out that rectangles in p5 were by default were being drawn in centre mode (McCarthy, 2022) which as a result caused this issue. So by setting the health bar to be drawn in corner mode, it should solve the issue.



Now when enemies were touching the tower the health bar would reduce from right to left. One other problem is that the bar kept reducing – I looked at the console and found this was due to the current health value was reducing to negative

```
if(health < 0){
  health = 0
```

To prevent the player from having negative health, if the player has negative health then their health will be set to 0 and so the minimum health the player can have is 0 rather than a negative value.



The health bar now works as intended – the player will take damage and the health bar will reduce until it reaches 0, now that this has been achieved, I will need a losing screen once the player has no more health and it is the end of the game.

Version 0.8

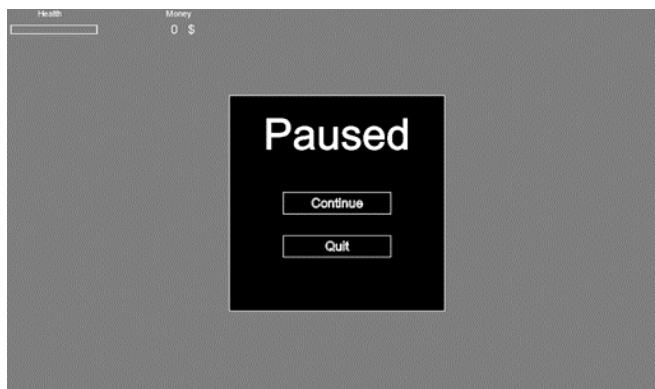
Creating a losing screen

Now that most of the basics of the game function correctly, the player now needs to actually be able to lose the game once they have no more health so the game won't infinitely go on.

- Once the players health reaches 0, then a losing screen will appear, stopping the game for the player,
- In the losing screen there will be a quit button so the player can leave the game if they want to, this will redirect the player back to the menu.
- A play again button will also be required in case the player wants to play the game again, when clicked a new game will be made.

It will be required to use the previous health code where when the health of the player decreases below 0, as when this happens, the requirement for losing screen can happen since the player has lost the game.

```
if(health < 0){  
    health = 0  
    gameMode = "pause", buttons =  
    gameButtons
```



Since I hadn't created a death losing yet, I used the pause screen first as an example before making it to see if I could make the screen be drawn once a certain condition was met, in this case the players health being 0. Using the previous health code once the player had reached 0 health, then the pause screen would appear.

When applying this, the test was immediately successful so I can now work on replacing this with a losing screen instead.

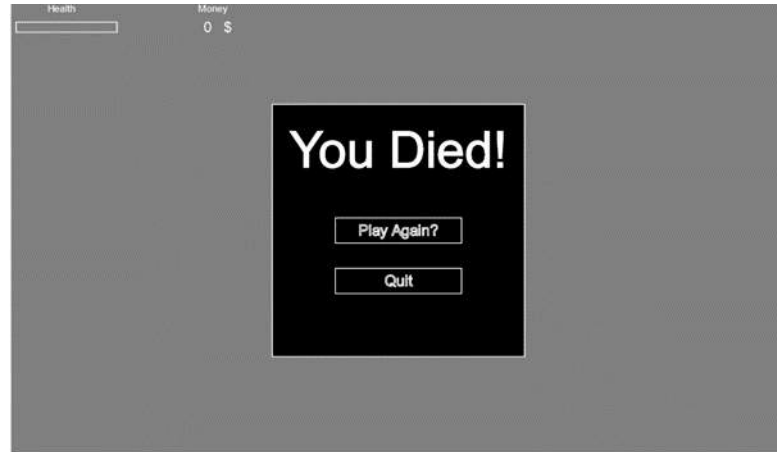
Since the player dying is completely different from all the other game modes, I created the new game mode dead so once the player has no more health the game mode . This will stop the game like the pause but will reset the game in both options.

```
if(health < 0){  
    gameMode = "dead", buttons =  
    deadButtons  
  
    case 'dead':  
        drawDead()  
        break
```

Once the players health is 0 or below, then the game mode is set to dead, then when this happens, a death screen function will be called and so will draw losing screen.

I thought the pause screen layout was completely fine and would look great as the death screen as well. Using this will also save time in coding as I would not have to create a new layout and so I reused the pause screen and replaced the text to replicate what a typical death screen would look like.

```
function drawDead(){
  background(255, 255, 255,
0.6)
  fill("white")
  fill("black")
  rect(windowWidth / 2,
windowHeight / 2, 500, 500)
  textSize(100)
  fill("white")
  text("You Died!",
windowWidth / 2, windowHeight
/ 2 - 125)
}
```



```
deadButtons =
[new Button("Play Again?",
windowWidth / 2, windowHeight /
2, 250, 50, () => { gameMode =
'play'; buttons = gameButtons
}),
new Button('Quit', windowHeight / 2 + 100, 250,
50, () => { gameMode = 'menu';
buttons = menuButtons })),
]
```

Now to make the death screen buttons, once the play again button is pressed, a new game will be created and all the stats are reset, this will be similar to the start button in the menu.

The quit button does the exact same function as the quit button in the pause menu so I used the same code for that.

The results showed the quit button worked as intended, but when pressing the play again button, the game would not reset, instead would update the game previously for a frame, then redraw the death screen



I figured since the menu start button and play again button are similar in functions, they would work the same way, however this clearly did not work, so I thought I may need to create a new way to start the game again from the losing screen menu.

```
deadButtons =  
  [new Button("Play  
Again?", windowWidth /  
2, windowHeight / 2,  
250, 50, () => {  
  gameMode = 'replay';  
  buttons = gameButtons  
})],
```

First I attempted to make a new game mode since using the previous play button didn't work - when pressing the play again button, I tried to redirect the user to a different game mode, replay – this would reset the game by redrawing the game for the player.

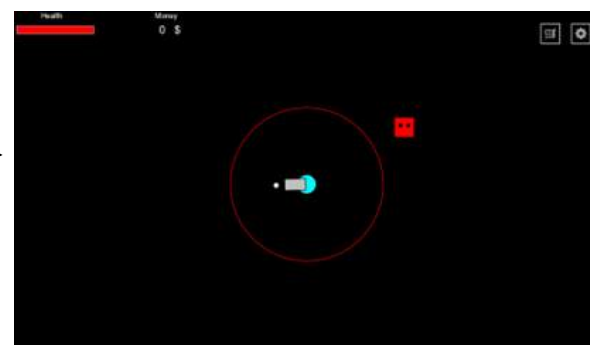
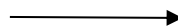
```
case 'replay':  
  gameMode = "play"  
  drawGame()  
  break
```

Applying this didn't change anything and the issue of the losing screen kept appearing.

The losing screen would only be drawn if the players health was below or equal to 0 the cause of this error may have been due to me not resetting the players health, and as a result the losing screen was drawn multiple times as the players health was always 0 and wasn't reset.

```
case 'replay':  
  reset()  
  gameMode = "play"  
  drawGame()  
  break
```

After adding the reset function to the game mode, replay, the play again button would now work for the player and reset the game successfully.



Version 0.9

Balancing the game

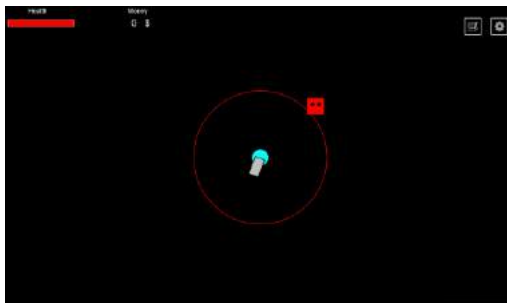
Enemies spawn rate

As of now a set number of enemies are created all at once, and so I will need to balance this out. In most games the game will start on a single enemy and once that enemy is defeated, more will be created. I decided to take this approach as it seemed quite balanced for the progression of the game.

How I want this to work is :

- Create one enemy at the start of the game
- If an enemy is killed, increase the number of enemies that can be created by 1
- Continue drawing these enemies until the maximum number of enemies have been created.

```
enemysSpawned = 1  
  
while (enemys.length <  
enemysSpawned)
```



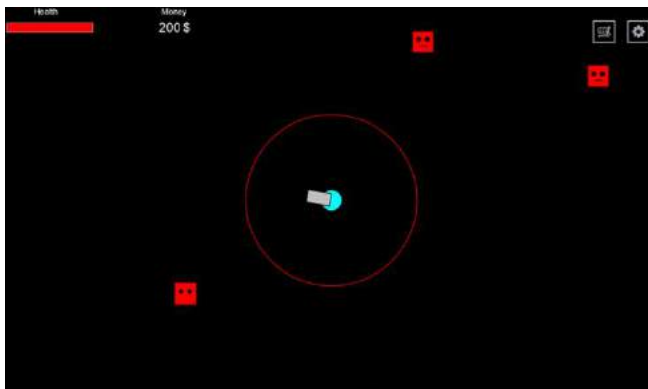
My first approach on this was creating a set number of enemies. To begin with I wanted 1 enemy to be created and the number of enemies to be drawn on the canvas would be determined by the variable, enemysSpawned.

As mentioned previously, the while loop made sure that the enemies would be drawn onto the canvas but as a result always made sure that a set amount of enemies are on the canvas making an infinite wave of enemies.

Now when enemies are killed by the tower, another enemy will be created at a random position on the canvas that isn't within the towers range.

```
if (enemyHit) {  
    console.log("colliding?", enemyHit);  
    enemys.splice(i, 1)  
    manualProjectiles.splice(manualProjectiles.indexOf(this), 1)  
    money += 100  
    enemysSpawned ++  
}
```

Adding an increasing number of enemies wasn't too difficult either, basically, by referring back to when an enemy is killed by a projectile from the player, once an enemy had been killed by the tower, then the amount of enemies that can be "spawned" can be increased by one, and so an infinite increasing amount of enemies can be created



From a demonstration, by killing 2 enemies, the number of enemies now being “spawned” for the play will be 3 enemies on the canvas. The problem now is that as soon as an enemy is killed, then a new additional enemy will spawn immediately, this would eventually become too overwhelming for the player, even if I added the addition of upgrades and so I will need a cooldown or rate at which enemies can be drawn.

```
enemys.push(new Enemy(xco,
yco))
    lastSpawned = millis()
}
```

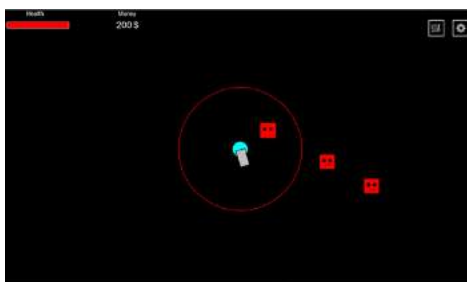
Making a enemy cooldown is exactly the same as my previous attempt at a projectile cooldown, once an enemy had been drawn on the canvas I would need to store how long it has been since that enemy has been drawn.

```
var spawnCooldown = 1000
```

Then making a cooldown on how long it should be between each enemy creation before another enemy can be drawn

```
while(enemys.length <
enemysSpawned && millis() >
lastSpawned + spawnCooldown) {
```

Comparing the current time to the last time an enemy was “lastSpawned” with the addition of the cooldown at which they can be “spawned” and so this will limit how many enemies can be drawn at a set interval.



I had no issues with the testing, one more enemy would be created after an enemy had been killed and this would happen after 1000 milliseconds had passed.

```
var spawnCooldown = 700
```

After some testing from my stakeholders, they found the spawnCooldown was too long as the player could essentially kill all the enemies before they had even spawned and so I adjusted the spawnCooldown to make the enemies be drawn slightly quicker to balance this out.

```
function reset() {
  enemys.length = 0
  projectiles.length = 0
  manualProjectiles.length = 0
  health = maxHealth
  money = 0
  enemysSpawned = 1
}
```

When restarting the game the amount of enemies created was not reset so when the player started a new game a large amount of enemies would be already moving towards the player rather than just the one – I had forgotten to reset the number of enemies once the player had pressed the play again or play button and so I just had to reset the number of enemies “spawned” once this reset was triggered.

A slight tower fire rate modification

```
var manualCooldown = 1300
```

Some adjustments were also required for the players manual fire mode as the player could fire way too quickly, outgunning the spawn rate. After multiple tests I made the tower shoot once every 1.3 seconds, this shouldn't be too much of an issue in the starting number of enemies but may cause the enemies to now eventually overwhelm the player since they can't shoot quick enough, this can be fixed with the addition of upgrades though.

Changing damage taken

```
health -= this.damage * 0.1
```

I check the console which logged the health when the player took damage and when looking at this, I noticed the amount of health reduced far too quickly for the player as from 1 enemy, the player was taking 1 damage per every millisecond, making even a single enemy quite unbalanced. By multiplying the damage by 0.1, the tower will now take 1 damage per second by each enemy, this seemed more bearable now for the player.

Version 1.0

Shop UI

As money can be generated from being enemies and the difficulty of enemies gradually increase, then the use of money will be prevalent for the player and so for the player to spend their money they will have to go to the shop. In order to do this, I will have to create a new shop button in which the player will press to access the shop .

This should work accordingly :

- If shop button is pressed by the player
- Pause the current game
- Draw the shop screen and relevant shop buttons

```
new Button('🛒', 1370, 60, 50, 50, () => { gameMode = 'pause';
```

After some modification on the of the positioning of the shop button I received my desired result. When pressing the shop button, the game will now pause.

After this, the shop button has to draw the shop UI onto the canvas, during this time, the game will be paused for the player since they would need to be focusing on what upgrades they will buy when in the shop.

```
new Button('🛒', windowWidth - 130, 60, 50, 50, ()  
=> { gameMode = 'shop'
```

Applying my previous knowledge from the settings button, I positioned the shop button next to the settings button since those 2 buttons will be used the most by the player and is also positioned away from the main game so won't be distracting, also, I made the button an shopping cart since it would be an appropriate icon as the players will go to the shop to buy upgrades.



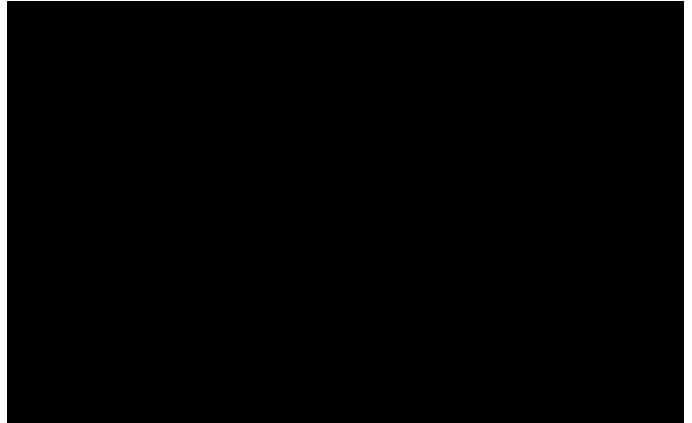
In order for this to happen, this would involve a new game mode which I will call shop in relevance to the button and once the player is in this shop mode, the shop screen alongside some upgrade buttons can be applied.

```
case 'shop':
  drawShop()
  break
}
```

Once the game mode has been switched to shop, then the function drawShop() will be called, this will draw the shop once on the canvas.

```
function drawShop() {
  background("black");
}
```

When applying this, when shop button was pressed, the screen was successfully filled in black. Now for the shop, I will need to be able to leave the shop, and so some shop buttons will need to be made.



```
new Button('🛒',
  windowWidth - 130, 60, 50,
  50, () => { gameMode =
    'shop'; buttons =
    shopButtons
```

The moment the shop button is clicked by the player, the shop buttons and the shop UI will be drawn on the screen

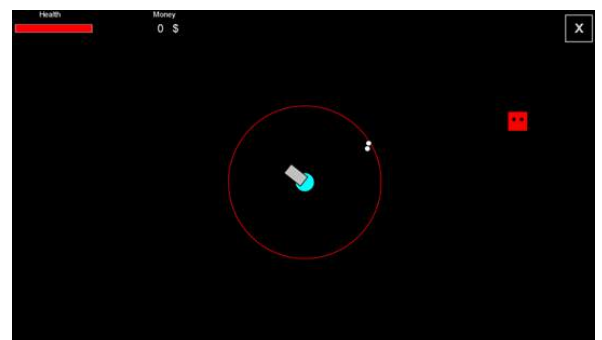
```
shopButtons =
[new Button("X", windowWidth - 50
, 50 , 70, 70, () => { gameMode =
'play'; buttons = shopButtons}),]
```

For now, I tried to make an exit button for the player so they can leave the shop and continue with the game.

Running the program, I received an error - I forgot to define the shopButtons as a variable so I ended up with an error that shopButtons was not defined and so to fix this I just had to declare the shopButtons as a variable.

```
var shopButtons = []
```

After fixing the mistake, the exit button would appear in the shop, but when exiting out of the shop, the settings and shop button would not appear and instead, the exit button would remain during the game.



```
shopButtons =
[new Button("X", windowWidth - 50
, 50 , 70, 70, () => { gameMode =
'play'; buttons = gameButtons}),]
```

I quickly realised I had made an error in my exit button, to which once the button was pressed the buttons being drawn would be the shopButtons again which as a result drew the exit button instead of the settings and shop button. To change this, when the exit button in the shop is pressed then the buttons that will replace it will be the game buttons rather than the shop buttons.



This ended up fixing the previous buttons issue

```
function drawShop() {
  background("black");
  fill("white")
  textSize(100)
  text("Shop",
windowWidth/2, 100)
}
```



I added the aesthetics to the shop by simply adding the text shop to the shop, this is quite common in games to make sure the player knows they are in the shop for a game and not some random menu.

Unfortunately, I did not have enough time in the development of my project to add upgrades to the shop, however, as I have the shop UI sorted and have a money system, in a future development I will add some upgrades that will benefit the player which could include increasing the players health, bullet speed and add upgrades I have already developed but not implement yet, in this case the – auto fire upgrade that I developed in a development previously – during this, I will also have to account for the balancing of these upgrades, that being the price of the upgrades, the amount of times the player can upgrade these upgrades, and the power of the upgrades .

Development Testing

In the development and final testing of the project, each objective must be tested in iterative amounts depending on how mandatory it is in the criteria and the complexity of the criteria. This testing is done in order to test for any bugs and will also show any limitations my program may have.

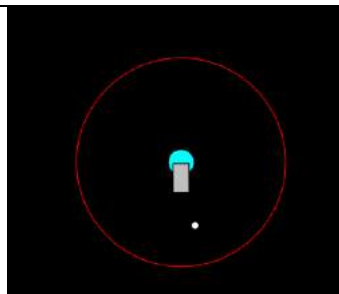
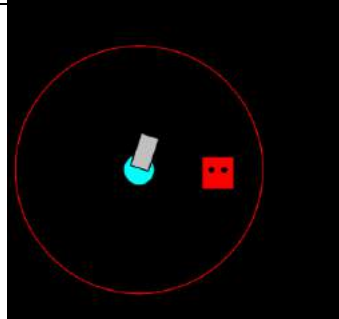
MUST Criteria

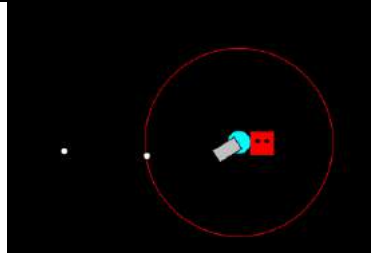
M1 Manual Tower

The basics of the tower will allow the player to control the tower

#	Description of test	Expected outcome	Result
M1.1	Tower setup	Every time the game is started, the tower should be drawn within the middle of the player's screen and filled in with blue for the main body of the tower and grey for the tower's cannon.	Success
M1.2	Rotating Cannon	The tower's cannon will update its position based of the direction according to the player's mouse position on the canvas.	Success
M1.3	Shooting Tower	When the player clicks on the screen, a projectile will be drawn at the towers position .	Success

M1 Testing

#	Description of test	Outcome of test	Result
M1.1	Tower setup		Success
M1.2	Rotating Cannon		Success

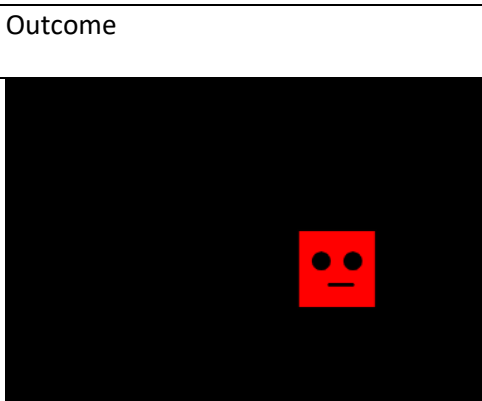
M1.3	Shooting Tower		Success
------	----------------	---	---------

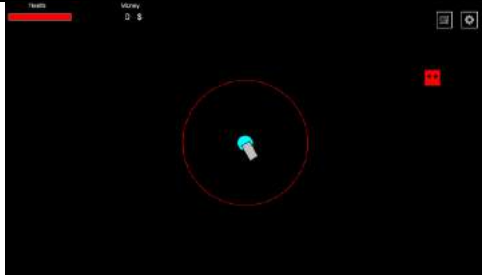

M2 Enemies

Enemies are the main objective for the player to beat, they will gradually increase in difficulty the more enemies defeated.

#	Description of test	Expected outcome	Result
M2.1	Enemy Appearance	Every enemy will be drawn with a face and will be filled in red.	Success
M2.2	Set Enemy Spawn	At the start of the game, one enemy will be drawn within the canvas.	Success
M2.3	Random Spawn	Enemies will be drawn within the canvas a set distance away from other enemies and must be drawn away from the tower's area.	Success
M2.4	Spawn Cooldown	Each enemy will be drawn every 1 second until the current limit of enemies the player has killed has been reached.	Success
M2.5	Enemy Movement	All enemies will make their way towards the tower at a speed of 1.	Success
M2.6	Damage System	When enemies are colliding with the tower, the enemies will deal 1 damage to the tower per second.	Success

M2 Testing

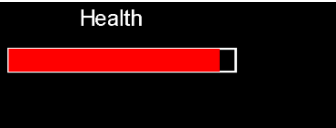
#	Description of test	Outcome	Result
M2.1	Enemy Appearance		Success

M2.2	Set Enemy Spawn		Success
M2.3	Random Spawn		Success
M2.4	Spawn Cooldown	As shown in video demonstration	Success
M2.5	Enemy Movement	As shown in video demonstration	Success
M2.6	Damage System	As shown in video demonstration	Success

M3 Health

#	Description of test	Expected outcome	Result
M3.1	Health Bar Display	A health bar UI will be drawn in the top left-hand side of the screen and is filled in red.	Success
M3.2	Health Bar Values	The health bar will have a maximum health value and minimum health value - it can contain at default the maximum of 500 health and the minimum amount of health being 0.	Success
M3.3	Health Bar Update	When the player takes damage from enemies, the health bar will decrease according to how much health the player currently has.	Success


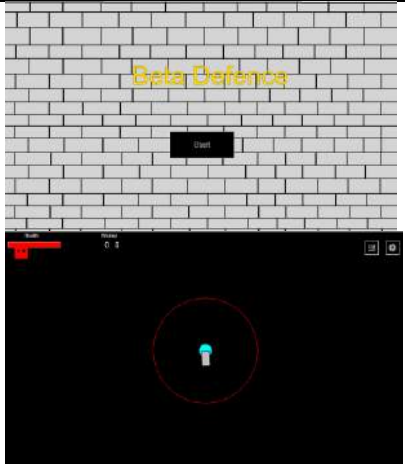
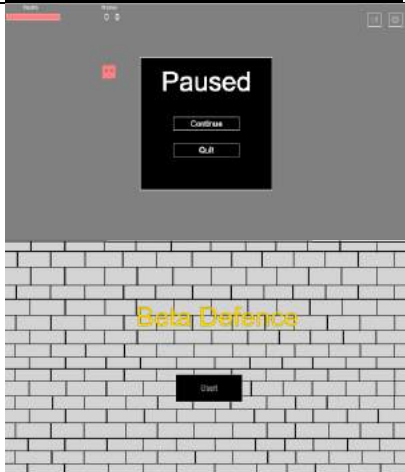
M3 Testing

#	Description of test	Expected outcome	Result
M3.1	Health Bar Display		Success
M3.2	Health Bar Values	As shown in video demonstration	Success
M3.3	Health Bar Update	As shown in video demonstration	Success

M4 Menu

#	Description of test	Expected outcome	Result
M4.1	Menu Screen	Background is drawn with imported image and start button .	Success
M4.2	Start Button	Start button when pressed, will immediately start the game for the player.	Success
M4.3	Pause Buttons	When the quit button is pressed, the player will be redirected to the menu.	Success

M4 Testing

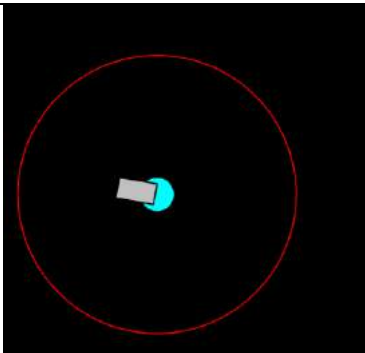
#	Description of test	Outcome	Result
M4.1	Menu Screen		Success
M4.2	Start Button		Success
M4.3	Pause Buttons		Success

SHOULD Criteria

S1 Tower radius

#	Description of test	Expected outcome	Result
S1.1	Tower Range Setup	The tower will have a range indicated by a red outline of a circle.	Success
S1.2	Detecting Enemy	Once an enemy is within the tower's range and the player has purchased the auto shoot upgrade, a projectile will automatically be drawn at the towers position.	Partial Success- upgrade works but hasn't been implemented.
S1.3	Tower Range Cooldown	Projectiles will be drawn at the tower's position every 2.5 seconds.	Success
S1.4	Projectile Movement	Projectiles will prioritise the nearest enemy within the towers range.	Success

S1 Testing

#	Description of test	Outcome	Result
S1.1	Tower Range Setup		Success
S1.2	Detecting Enemy	Not implemented	Partial Success- upgrade works but hasn't been implemented.
S1.3	Tower Range Cooldown	As shown in video demonstration	Success
S1.4	Projectile Movement	As shown in video demonstration	Success

S2 Simple Enemy Ai

#	Description of test	Expected outcome	Result
S2.1	Enemy Movement	Section is covered by M2.5	Success

S2 Testing

#	Description of test	Outcome	Result
S2.1	Enemy Movement	As shown in video demonstration	Success

S3 Difficulty in game

	Description of test	Expected outcome	Result
S3.1	Starting enemies	Section is covered by M2.2	Success
S3.2	Increasing enemies	When one enemy is killed by the player, an additional enemy will be drawn after 2.5 seconds.	Success
S3.3	Endless Mode	An infinite number of enemies will continue to be drawn on the canvas based on the number of enemies the player has killed.	Success

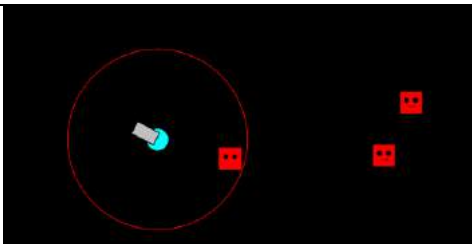
S3 Testing

	Description of test	Outcome	Result
S3.1	Starting enemies	Section is covered by M2.2	Success
S3.2	Increasing enemies	As shown in video demonstration	Success
S3.3	Endless Mode	As shown in video demonstration	Success

S4 Enemy Creation

#	Description of test	Expected outcome	Result
S4.1	Enemy Location	Section is covered by M2.2 and M2.4	Success

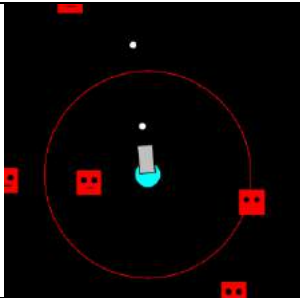
S4 Testing

#	Description of test	Outcome	Result
S4.1	Enemy Location		Success

S5 Projectile

#	Description of test	Expected outcome	Result
S5.1	Projectile Creation	Projectiles can either be created from the player clicking their mouse or using the auto shoot upgrade.	Success
S5.2	Speed of Projectile	Every projectile will have a speed of 2 when being fired from the tower.	Success
S5.3	Movement of Projectiles	Section is covered by M1.3 and S1.4.	Success

S5 Testing

#	Description of test	Outcome	Result
S5.1	Projectile Creation		Success
S5.2	Speed of Projectile	As shown in video demonstration	Success
S5.3	Movement of Projectiles	As shown in video demonstration	Success

S6 Home

#	Description of test	Expected outcome	Result
S6.1	Redirect Button	Section is covered by M4.3	Success

Testing

#	Description of test	Outcome	Result
S6.1	Redirect Button	Section is covered by M4.3	Success

S7 Simple Upgrades

Objective not attempted

S8 Earning Money

#	Description of test	Expected outcome	Result
S8.1	Money generation	The player by default will begin at 0\$ and when an enemy is killed by the player, the player's current amount of money will increase by 100\$.	Success
S8.2	Money Display	The amount of money the player currently has will be displayed in real time and	Partial success – player can't

		update whenever the player gains or loses money.	spend money yet.
S8.3	Purchasing	When the player purchases an upgrade, the appropriate amount of money is deducted from the players current money.	Fail – not implemented

S8 Testing

#	Description of test	Outcome	Result
S8.1	Money generation	As shown in video demonstration	Success
S8.2	Money Display	As shown in video demonstration	Partial success – player can't spend money yet.
S8.3	Purchasing	Not added	Fail – not implemented

COULD Criteria

C1 Waves

#	Description of test	Expected outcome	Result
C1.1	Increasing enemies	As shown in video demonstration	Success
C1.2	Enemy stats increase	Not added	Fail – not implemented

C1 Testing

#	Description of test	Outcome	Result
C1	Increasing enemies	Section covered in S3.	Success
C2	Enemy stats increase	Not added	Fail – not implemented


C2 Types Of Enemy

Objective not attempted

C3 Settings

#	Description of test	Expected outcome	Result
C3.1	Quit button	Section covered in M4.3	Success
C3.2	Continue button	When the continue button is pressed, the game will carry on as before it was paused.	Success
C3.3	Transparent Background	When settings button is used, the background will be transparent, allowing the game to still be seen during the pause menu and the game state will be paused.	Success
C3.4	Music Toggle	Music volume and sound can be adjusted by the player.	Fail – not implemented

C3 Testing

#	Description of test	Outcome	Result
C3.1	Quit button	Section covered in M4.3	Success
C3.2	Continue button	As shown in video demonstration	Success
C3.3	Transparent Background		Success
C3.4	Music Toggle	Not added	Fail – not implemented

C4 Sound/Music

#	Description of test	Expected outcome	Result
C4.1	Game music	Music will play throughout the duration of the game.	Fail – not implemented
C4.2	Additional Sounds	Using buttons or the firing of the cannon will make an appropriate sound when pressed.	Fail – not implemented

C5 Particles

Objective not attempted

C6 Powerups

Objective not attempted

C7 Additional Upgrades


Objective not attempted

C8 Shop/Shop UI

#	Description of test	Expected outcome	Result
C8.1	Shop Icon	A small shop icon will appear when the game has started.	Success
C8.2	Shop setup	When the shop icon is pressed the shop menu will be drawn changing the game state and current screen to the shop, during this, the game will be paused and the background will be fully black.	Success
C8.3	Upgrades	Upgrade buttons will be listed in ascending order of prices and can only be pressed when the player meets the money requirements for the upgrade.	Partially successful – not implemented

C8 Testing

#	Description of test	Outcome	Result
C8.1	Shop Icon		Success

C8.2	Shop setup		Success
C8.3	Upgrades	Not implemented	Partially successful

C9 Advanced Enemy AI

Objective not attempted

C10 Statistics

#	Description of test	Expected outcome	Result
C10.1	Money	Section covered by S8	Fail – not implemented

C10 Testing

#	Description of test	Expected outcome	Result
C10.1	Money	Money values displayed on statistics menu.	Fail – not implemented

#	Description of test	Outcome	Result
C10.1	Money	Not added	Fail – not implemented

C11 Leader board


Objective not attempted

C12 Losing Screen

#	Description of test	Expected outcome	Result
C12.1	Checking player's current health	If the player current health is 0 or below, then the losing screen will be drawn, thus ending the game	Success
C12.2	Quitting the game	Section covered in M4.3	Success
C12.3	Restarting the game	When the restart button is used, a new game is generated for the player, all previous stats such as the player's current health, enemies that can be	Success

		spawned and money are reset to default values.	
--	--	--	--

C12 Testing

#	Description of test	Expected outcome	Result
C12.1	Checking player's current health		Success
C12.2	Quitting the game	Section covered in M4.3	Success
C12.3	Restarting the game	As shown in video demonstration	Success

WON'T Criteria

W1 Multiplayer

Objective not attempted

W2 Transactions

Objective not attempted

W3 Cosmetics

Objective not attempted

W4 Login

Objective not attempted

Evaluation

As I have finished with the programming and made some necessary tests to make sure certain objectives have been met, I now need my stakeholders to give a non-biased evaluative response on what they think of my game. From the aim of this response, I hope to achieve some constructive feedback that I can hopefully apply and implement into a future development and also have some insight to how well my game is received from my stakeholders.

To do this, I will create a simple test plan with an appropriate amount of space to provide feedback, alongside with a link to my game.

Stakeholder Testing

Hello everyone, thank you for your contribution to the development of my game, Beta Defence. The first development of the game is now finished and so I will need your help for some feedback on the game. When testing, please don't refrain from any criticism and honest feedback, it will help make future development of the game much more beneficial and worthwhile.

When playing the game, keep in mind of the questions presented in the columns below and when answering them, take your time and put down your own opinion. If there are any specific details of my game that you like or dislike, do try to note them down.

Here's the link to the game : <https://beta-defence-final-development.williamcheung2.repl.co>

Thank you once again for your help and time in the development of my game, it really wouldn't be possible without your help honest feedback.

Looking forward to working with you in future developments,

William.

Starting the Game

When starting the game, you will begin from the game menu. Here, you will be presented with the game title and a start button that will start the game when pressing the start button.

Action	Comment
Is the game menu what you would expect from a typical tower defence game?	
Does the game title and screen attract you, the player to play the game?	
Is the start button too simple – any modifications I could make?	
Do you think any aspects of the menu could be improved and why?	

Comments :

The Tower

The main core of the game's progression will be completed from the player controlling the tower, the tower should be able to shoot at a fixed cooldown when clicking on the screen and will face towards where the players current mouse position is.

Action	Comment
Click on the screen when game starts.	
Does the default tower feel balanced when clearing enemies ?	
Would you be interested in any further development such as special effects to the projectiles?	
Do you think the tower model is suitable for the game or is there any other adjustments that could be taken in account?	
Is it clear what the purpose of the tower does?	
Are there any issues you have encountered when using the tower?	
Are there any further ideas you would like added to the tower?	
Does the player have enough starting health?	

Comments :

Enemies

The main objective of the game is to survive and kill as many enemies as possible. Enemies will randomly "spawn" within the game screen and make their way towards the tower, they will damage the tower once they touch the tower. To begin with there will only be one enemy but this will gradually increase the more enemy's you kill.

Action	Comment
What are your thoughts on how quickly the enemies move?	
Do new enemies "spawn" too quickly once a previous enemy has been killed?	
Do you think the enemies do enough damage to the tower or is this too much?	
What are your thoughts on the graphics of the enemies?	
Are there any issues with the enemies currently?	
Any other ideas that you would like to add to the enemies?	

Comments :

Shop System

Whilst, the shop system hasn't completely been implemented yet, in a future development, it should contain upgrades that will help the player progress through the game.

Action	Comment
Apart from your default health, damage upgrades, are there any upgrades you would be interested in the game.	
What are your current thoughts on the shop design?	
Any further items or ideas you could see being in the shop?	
Have you found any issues when accessing the shop?	

Comments :

UI Screens

During the game, you can access a few screen by pressing the appropriate button. There should be a pause screen, losing screen, shop, and the game menu.

Action	Comment
Does the pause menu look and work as expected?	
Aesthetic wise, do you think the buttons for the shop and settings are appropriate for their use?	
Does the health bar decrease look natural for the player?	
Do you think the losing screen's purpose is clear enough for the player?	
Are there any issues with any of these screens?	
Any other ideas you would like to improve on the current design or would like to add?	

Comments :

Results from stakeholder testing

Jayden and some other stakeholders tested my game and then emailed me back the result of their testing . From the testing relevant feedback is given back test plan previously given. Jayden is the 1st commentor.

Starting the Game

Action	Comment
Is the game menu what you would expect from a typical tower defence game?	<ol style="list-style-type: none">1. Whilst it is not as flashy or over the top as other tower defence games, from the simplicity of the game screen and title, it is clear that I am playing a tower defence game .2. Yes, the title is similar to your average tower defence game and the same with the background, but more popular games tend to have other options apart from start, like a stats option.3. Yep it looks pretty much the same as a regular tower defence game.4. It's pretty simple, but its function and works well.
Does the game title and screen attract you, the player to play the game?	<ol style="list-style-type: none">1. Other than the title being slightly off the centre the title whilst simple does do the job for a game.2. Yes, but the title is slightly off centre.3. The background is good, albeit a bit bland. The title card stood out, although it was off-centre.4. It's quite eye catching, but it is off-centre.
Is the start button too simple – any modifications I could make?	<ol style="list-style-type: none">1. Perhaps adding some highlighting when hovering over each button/ sounds would be really cool. Also having different fonts/colours for each button as they are currently all the same colour.2. Yes – could use a different look, maybe could do with an image background for the button.3. They are quite simple, it'll be nice if it were a custom image instead4.
Do you think any aspects of the menu could be improved and why?	<ol style="list-style-type: none">1. Other than moving the title to the centre, the title might need some shading to look more professional.2. Maybe a few different game modes to choose from at the beginning of the game.3. Centre and re-align menu elements and make the menu more colourful.4. More graphics or images could be added to be more appealing to the player.

Comments :

1. Despite being simple, the game certainly had an approachable feel to it, and other than some detail improvements/ moving the title, the menu seems pretty solid.
2. Overall the start of the game is quite good but could do with a couple of updates to the graphics and maybe another button.
3. The main menu is very basic but does what it needs to do. A few custom image assets and a more colour full background would definitely improve this screen.
4. The design of the menu is quite simplistic but straightforward and functional.

The Tower

Action	Comment
Click on the screen when game starts. Does a projectile get created?	<ol style="list-style-type: none">1. The tower fires wherever the player shoots which works well.2. Yep, the tower fires a projectile and takes its time trying to reload the next shot.3. Projectile fires and travels at a balanced speed.4. It shoots a projectile in the direction of the mouse position correctly.
Does the default tower feel balanced when clearing enemies ?	<ol style="list-style-type: none">1. The shooting speed at a default value is quite slow and definitely feels punishing for the player if they miss, this is quite balanced.2. It feels perfect when given the ability to 1 shot its targets but then the fire rate proves to give the game difficulty.3. The slow fire rate forced me to be accurate with my shots which I like.4. It one-shots any incoming enemy, but the increasing number of enemies that approach the tower eventually makes it so the tower is too underpowered to defend itself.
Would you be interested in any further development such as special effects to the projectiles?	<ol style="list-style-type: none">1. I would love if the projectiles had some sort of trail following behind them or some explosions whenever the player kills an enemy.2. It would be nice to have randomly firing projectiles or explosives that have a chance to occur when the player shoots.3. Effects when the tower shoots and enemies die would look very cool.4. I would love if varying types of projectiles and effects were available to use
Do you think the tower model is suitable for the game or is there any other adjustments that could be taken in account?	<ol style="list-style-type: none">1. The tower is quite simple, and only filled in with basic colours and no proper tower like features or detail, this would look nicer if they had some of these properties.2. The tower graphics are fine as they are but if the game's graphics were to be updated then I would suggest trying to change the tower as well.3. The tower model could look a bit more interesting, and the projectiles should start at the barrel and not the centre of the tower.

	4. It is pretty simple but conveys what it is clearly. A more detailed sprite could be used instead to make it more good looking.
Is it clear what the purpose of the tower does?	1. To shoot enemies using your mouse. 2. Yes – to turn towards an enemy and shoot projectiles in their general direction. 3. To shoot? Yes. 4. It is quite clear what it is meant to do.
Are there any issues you have encountered when using the tower?	1. Seems to function like a normal tower, so nothing. 2. The fire rate is a little slow but other than that it seems to work as intended. 3. Nothing 4. I feel as though the fire rate is a bit too slow and can't fend off a large number of enemies.
Are there any further ideas you would like added to the tower?	1. Some skins for the tower/additional detail that other tower defence games have for their tower and adding upgrades/ abilities would help the game progression as well. 2. Maybe a different type of projectile of multiple cannons in different directions. 3. An indicator showing when you can shoot again would be nice. 4. Different tower variants could be implemented, with different starting abilities.
Does the player have enough starting health?	1. Starting health seemed average and well thought out. 2. The health is ok at the beginning, but I encountered a problem when more enemies started to spawn and that was if many enemies were on the screen but only 1 was touching the tower then it would do significantly more damage than before which can really ramp up the difficulty when there aren't any upgrades 3. Yes this seemed balanced. 4. The health is fine, but the capability of the tower defending itself is too low.

Comments :

1. Difficulty definitely spikes as enemies will eventually overwhelm the player due to the low fire rate speed. Upgrades are definitely needed for the progression of the game to continue and be more engaging.

2. The health is ok at the beginning, but I encountered a problem when more enemies started to spawn and that was if many enemies were on the screen but only 1 was touching the tower then it would do significantly more damage than before which can really ramp up the difficulty when there aren't any upgrades

3. Other than some adjustments to the tower and perhaps adding some effects this was done pretty well.

4. I think that the tower works very well and is functioning correctly, but it cannot defend against the ramping difficulty.

Enemies

Action	Comment
What are your thoughts on how quickly the enemies move?	<ol style="list-style-type: none"> 1. I think the enemies moved at an average enough pace for the player to handle them. 2. Their movement too hasty for the player to try and keep up with. 3. The enemies move at fast enough speed to force the player to prioritise targets properly 4. The enemies move very well and are somewhat fast.
Do new enemies “spawn” too quickly once a previous enemy has been killed?	<ol style="list-style-type: none"> 1. At the start of the game, this seems relatively balanced, but as the game goes on, it start to get really quick and impossible to beat without upgrades of any sort. 2. The spawn rate of enemies is balanced right I feel. 3. The spawn rate is perfect at the beginning but increases too quickly for player to deal with them – think this will be solved by upgrades. 4. They spawn instantly after an enemy is killed, some spawning much closer to the tower than others – which is somewhat unbalanced.
Do you think the enemies do enough damage to the tower or is this too much?	<ol style="list-style-type: none"> 1. Enemies do a decent amount of damage, though I may of found a bug when multiple enemies are on screen, a single enemy will do a lot more damage to the player than expected. 2. It ramps the damage a lot when there are more enemies on the screen at one time. 3. They do enough damage to encourage to kill them ASAP. 4. A singular enemy does a very miniscule amount of damage – but on the instance that many enemies have spawned but only a few are damaging the tower, the damage done to the tower increases greatly which I think may be a bug.
What are your thoughts on the graphics of the enemies?	<ol style="list-style-type: none"> 1. I really enjoyed the quirkiness of the enemies, though, one issue is that one enemy does not spawn with a mouth occasionally . 2. The enemy’s graphics are fine except from 1 of them doesn’t have a mouth and when you kill it the mouth of another is removed - so that could be fixed. 3. They do enough damage to encourage to kill them ASAP. 4. They are quite plain, sometimes changing randomly when approaching the tower from a simple face to just eyes.

Are there any issues with the enemies currently?	<ol style="list-style-type: none"> 1. One enemy always being present without a mouth for some reason. 2. No, just the graphics with the mouth not being on one of them. 3. Spawn rate increases too quickly. 4. They are quite plain, sometimes changing randomly when approaching the tower from a simple face to just eyes.
Any other ideas that you would like to add to the enemies?	<ol style="list-style-type: none"> 1. Variants of enemies such a really bulky enemy that the player cannot 1 shot and must shoot multiple times before killing. 2. Possibly different types of enemies. 3. Different enemy types having different health, damage, and speed. 4. There could be different enemy variants that spawn, maybe increasing in difficulty to kill as waves progress.

Comments :

1. Enemies looks great apart from the one bugged enemy. Enemies do spawn a little too quickly, but I think adding upgrades will solve this issue.
2. The enemies graphics might need a change but other than that, the only thing I feel it needs are different types of enemies.
3. The enemies seem very balanced, apart from their spawn rate which is too fast. I assume this would be less of an issue when the shop system is fully implemented.
4. The enemies are mostly good, but the damage is bugged when a lot are spawned. More

Shop System

Action	Comment
Apart from your default health, damage upgrades, are there any upgrades you would be interested in the game.	<ol style="list-style-type: none"> 1. Exploding bullets that can hit multiple enemies, ricochet bullets that bounce between each enemy. 2. Fire rate to give a better response time, explosive bullets to deal area damage, piercing bullets to give them the ability to hit multiple enemies, other towers to auto shoot for the player. 3. Faster fire rate, shields, auto towers that fight for you, skills/items that provide special effects. 4. Different firing types like spread shot or multi-shot; varying projectiles like AOE and piercing; status effects like freezing, poison or burning.
What are your current thoughts on the shop design?	<ol style="list-style-type: none"> 1. Does look empty as of now, the addition of upgrades will solve this issue though.

	<p>2. Its ok as of now, there is not much right now in terms of aesthetic but when it gets updated it could do with some aesthetic changes like smooth corners to provide a cleaner look.</p> <p>3. It'd be better if there was something to buy.</p> <p>4. It can be accessed easily, but there are no options available to buy.</p>
Any further items or ideas you could see being in the shop?	<p>1. Cosmetics such as hats or skins for the enemies/tower will be great!</p> <p>2. An upgrade for the tower to auto shoot enemies.</p> <p>3. Same as previous.</p> <p>4. Possibly cosmetic upgrades.</p>
Have you found any issues when accessing the shop?	<p>1. The shop seemed to work every time I pressed the button.</p> <p>2. From the many times I played the game I can say that there were no problems with the game's shop.</p> <p>3. Nope, loads up fine every time.</p> <p>4. No issues encountered.</p>

Comments :

1. The biggest thing the shop requires is things to actually purchase - upgrades, without it, it pretty much is useless being there.
2. The shop definitely needs update by adding upgrades to give the game some length and versatility.
3. The shop loads up, but without any upgrades I cannot say more than add the upgrades. They'd make gameplay a lot more engaging.
4. The shop can be accessed, but there is no use for it so far as nothing can be bought so I guess adding upgrades will be the next step here.

UI Screens

Action	Comment
Does the pause menu look and work as expected?	<p>1. The fading out of the screen was awesome I quite liked that.</p> <p>2. Yes – the aesthetic is pleasing and makes it obvious what it is.</p> <p>3. Yes, I like how it fades to white.</p> <p>4. It works very well, freezing the state of the game correctly and adding a little fade in effect which I found cool.</p>
Aesthetic wise, do you think the buttons for the shop and settings are appropriate for their use?	<p>1. Shopping trolley and settings were appropriate for their usage.</p>

	<p>2. Yes, they fit the feel of the game nicely and make sense to their attributes.</p> <p>3. Yes, they convey what they are perfectly.</p> <p>4. The images clearly convey what each button represents.</p>
Does the health bar decrease look natural for the player?	<p>1. Yep, it was clear when the player was taking damage, though actually seeing the number of health the player has would be beneficial.</p> <p>2. Yes, except from when there are many enemies on the screen and then one enemy does a lot more damage – then it seems unnatural.</p> <p>3. Yes, it decreases properly each time I got hit. Maybe add the actual value of their health in the health bar.</p> <p>4. They correctly show how much the player has left in health, but a numeric value would be nice.</p>
Do you think the losing screen's purpose is clear enough for the player?	<p>1. The purpose was clear.</p> <p>2. Yes, it indicates well that the player has been killed from the losing screen.</p> <p>3. Yes, it's very noticeable.</p> <p>4. It is very clear and straightforward but is quite similar to the pause menu might be nice to make a slight difference between the two.</p>
Are there any issues with any of these screens?	<p>1. All the screens I tested seemed to carry out their function correctly.</p> <p>2. No, the screens seem fine, except from the title at the beginning being off centre.</p> <p>3. Nope</p> <p>4. No, not at all.</p>
Any other ideas you would like to improve on the current design or would like to add?	<p>1. Some screens such as the pause menu and losing screen seem a little too similar, maybe adding some difference in detail could be used here.</p> <p>2. Maybe some extra colours or effect to give the game a slight aesthetic update to make things look a little more joyful.</p> <p>3. More colour could be added to each menu, like red in the game over screen.</p> <p>4. No, not at all.</p>

Comments :

1. Adding some upgrades to the shop and adding more detail and colour to make the game seem more lively would be nice. Some additional details to the menus and different screens might also need some improvement.
2. The UI doesn't seem as if it needs much change except from things to brighten up the game a little.
3. More colours could be added to each menu, like red in the game over screen.
4. Overall, the UI is well done and working well with no issues at all. The shop could use more work, however.

Success of the project

M1 Manual Tower + S5

The objective has been completed in tests M1.1 to M1.3 and S5.1 to S5.3. The stakeholders responded with some negative comments on the graphics and design of the tower but did also follow up with some positives on the functionality and purpose of the tower. For the graphics of the tower the comments from “The tower is quite simple”, “The tower model could look a bit more interesting”, “This is quite balanced”, “Forced me to be accurate – which I like”]. While I was developing the tower, it was rather difficult on producing visually pleasing graphics as I had to prioritise the production of the tower actually functioning first priority rather than the graphics, so after looking back on the feedback on the graphics on the tower, I completely understood from the developer’s standpoint of the game on why the stakeholders felt negatively on the graphics of the tower, for a future development, I will implement more advanced graphics for the tower by either using some sort of drawing software to draw the tower or ask one of my stakeholders to see if they would like to produce a model of a game model of a tower then implementing this using p5’s preload function. Despite the negatives of the feedback, as the developer of the game, I was rather pleased to hear some positive feedback on the balancing of the tower as the stakeholders did find the tower at a default level as balanced, this made me quite happy as I spent a great deal of time working on the development of the tower and tested multiple times to make sure the tower would shoot at a balanced rate.

M2 Enemies + S2 + S4

The objectives have been completed in tests M2.1 to M2.6, S2, and S4. The stakeholders were quite varied in their responses on the balancing of the enemies and had mostly positive comments on the graphics of the enemies. When asked on the balancing of the enemies, most of the stakeholders agreed on the of the “spawning” of each enemy was “balanced” during the start of the game, but gradually became too difficult as the game progressed. [“The spawn rate is perfect at the beginning... then increases too quickly”, “somewhat unbalanced”, “starts to get really quick and then becomes impossible], though the stakeholders did have some positive comments on the damage of the enemies which they thought were rather balanced at the start and moved pretty well for an average speed [“enemies move very well”, “the enemies moved at enough for the player to handle them”]. The balancing of enemies was particularly difficult during the development stage, as since I had not produced any upgrades for the tower, it was rather arduous to make the enemies be “spawning” at a pace where the player could keep up until a set point where upgrades would be needed for the player to progress as I hadn’t implemented the upgrades in the shop, this was clearly quite difficult to balance. As a result of this, I could definitely fix the issue of the enemies “spawn too quickly” by adding upgrades for the player, in order to allow the player handle the waves of enemies and so can progress further through the game, rather than only being able to progress further through the game.

M3 Health System

The objectives have been completed in tests M3.1 to M3.3. Most if not all the stakeholders were very positive on the design saying [“it was clear when the player was taking damage”, yes it does decrease properly each time I’m hit”]. Further comments on the functionality of the health system and in testing, saw no issues with the health system except from a very minor bug of a single enemy outputting too much damage to the player causing the health bar to reduce a bit too quickly, hopefully this can be resolved in a future development as it does cause a significant impact on the game’s playable state. A few stakeholders did comment that some numeral values to show the

actual value of the players health as a visual indication would be quite helpful, I will consider this in the future development section.

M4 Menu + S6

The objectives have been completed in tests M4.1 to 4.3 and S6. Stakeholders whilst had some comments of the title being somewhat [“plain” and “not as flashy or over the top as other tower defence games”], they were generally quite happy with the design albeit the title being slightly off centre of the screen, though this was only really due to me mistakenly resizing the screen to suit my device and not taking into account the other devices the game could be played on which could have different screen sizes. This definitely can easily be adjusted to suit other devices in a future development. I was really happy to see from a developer’s standpoint that the stakeholders relatively liked the design of the menu

S1 Tower Radius

The objectives have been partially complete in S1.1 to S1.4. I demonstrated that I was able to creating a working “upgrade” for the player to automatically shoot enemies in range at a cooldown per x seconds but never ended up implementing it into the shop through the upgrades system, though, this would definitely be implemented in a future release. I am a little sad that this upgrade was never implemented as the process of producing the upgrade for the tower to automatically shoot was rather challenging learning and was quite a lengthy process . At first, I didn’t understand how to detect the collision of circles and so I had to make use of the p5 collision library to which I had no experience with previously and then using vectors to move projectiles in the direction of enemies was also a relatively new concept that I had not encountered before and definitely was challenging in the long run in the development. Though this was definitely worthwhile development as later I was able to model the structure of the automatic tower to then produce a manual tower which has led to the basic functionality of the game to work nicely.

S3 Difficulty + C1

The objectives have been complete in S3.1 to S3.3 and C1.1 to C1.2. Evidently, nearly all the comments on the difficulty of the game came down to the game being balanced at the start since the player had a set fire rate and there being a limited number of enemies and then progressively becoming a bit too difficult and quite clearly “impossible” due to the subtraction of upgrades in the game. [“The spawn rate is perfect at the beginning”, “Increases too quickly for the player to keep up with – think this will be solved by adding upgrades”, “at the start of the game seems relatively balanced but then gets really quick and then impossible to beat”]. Such as most sections, the game is rather unbalanced since there is only so far the tower can progress without the need for upgrades. To add some upgrades, I would have to make an individual button which would be named for each upgrade and then a way for the button to be enabled or disabled depending on if the player reaches the money requirements for that upgrade. Once the specific button is enabled and has been pressed, the players current money will be reduced by an x amount depending on the upgrade price, then the corresponding upgrade will be implemented into the current game, e.g., upgrading health button will increase the health of the player by an x amount. I may also need to consider a set number of times the player can apply an upgrade and perhaps a visual indication for the player to see how many times they have upgraded.

S8 Earning Money

The objectives have been partially complete in S8.1 to S8.3. There weren't any too significant issues from the stakeholders other than there being no use for the money currently in the game and the shop being empty which is where upgrades can be implemented so the player can use their money to upgrade the tower. ["nothing can be bought", "I cannot say more than add upgrades", "biggest thing the shop and game requires is things to actually purchase – upgrades, without them it is pretty much useless."]. Though, as for upgrades, balancing will be very crucial as I would need to make sure the player is gaining enough amounts of money for each upgrades and not too much, to do this a lot of future testing on upgrades will be required. Other than the issue of money having no use, there is one minor issue I found myself personally where the \$ sign next to the money value does not move across the screen with the money value, making the value become overlayed which looks quite messy in the game. To fix this small issue, I would just have to output the value of the player's current money the update the position of text \$ whenever they are the same position.

C3 Settings

The objectives have been mostly complete in C3.1 to C3.4. Most comments on the pause screen were extremely positive they found the settings menu was quite clear in its purpose for each button and also the additional aesthetics of the fading of the screen was very well received. ["The fading out of the screen was awesome, I quite liked that", "the aesthetic is pleasing", "a little fade in effect which I found cool"] . This aspect of the program was really satisfying to produce and pretty successful for the game as In the development of the pause menu screen I took some careful consideration before deciding to add the effect to the settings menu and it ended up giving a really nice pause effect state that most retro games produced.

C8 Shop/Shop UI

The objectives have been partially complete in S8.1 to S8.3. There wasn't too much here said from the stakeholders since I wasn't able to completely implement an upgrades system. The icons for the shop were met with approving comments as all the stakeholders agreed that the shop icons suited the need and was clear what the outcome would be when clicked by the player , other than that, the majority suggested the shop was rather plain since there was "not much" and so required the addition of upgrades which would allow the shop to actually be more like a shop rather than an empty section of the game. As the developer, I am quite disappointed that I wasn't able to prioritise implement the upgrades system to the game as implementing the mechanics of the tower took longer than expected. Without the addition of upgrades to the game which would be the core aspects that will provide a way for the player to progress further into the game, without this feature, the game can only played to a certain point and so in a future development, adding upgrades must be priority. On the positive side, there were completely no issues when each stakeholder tested out opening the shop menu as no crashes or drastic gameplay affecting bugs occurred during this.

C12 Losing Screen

The objectives have been complete in C12.1 to C12.3 . Nothing too notable was said by the stakeholders here other than the function of the losing screen was clear for the player, though one stakeholder did say that the losing screen did look a bit too similar to the settings menu so may need some adjusting to have some variation in the games design. ["All the screens I tested seemed to carry out their function correctly", "the purpose of the losing screen is indicated well that the player has been killed from the losing screen", "It is very clear and straightforward but us quite similar to the pause menu and might be nice to make a slight difference between the two"]. For the losing screen perhaps some details such as adding a skull or some losing music for a future development.

Assessment of usability features

The graphics and user interface for the game was intentionally kept simple in order to provide a simple game that most people should be able to play without thinking too hard. The 2 main aspects which are core to the game include the game menu and the game screen itself.

Menu Screen

Some adjustment could be made to the start button in the menu screen perhaps by adding some shading or decoration to the buttons as but in terms of functionality it is robust does exactly what I want as the as begins the game whenever pressed. The start button has been tested a multitude number of times, each from different platforms (mobile, computer and laptop) and every time the start button always performs as intended, updating the game state, though, some adjustments may be needed to the menu screen as the title is slightly off centred on some screens on mobile devices.

The stakeholders did comment that “other than the title being slightly off centre, the title does do the job for the game” and “perhaps adding so highlighting over each button” when the player is hovering over each button and “an image background for the button could be nice”.

Gameplay

Most of the comments from stakeholders showed that they liked the design and understood how the key aspects of the game functioned especially the tower and the menu design.[“It feel perfect when given the ability to 1 shot the targets”. “the shooting speed feels punishing if the player misses which is quite balanced]

The game’s design was very evidently mostly well received by the stakeholders, though is not completely perfect just yet through some issues such as:

- The lack of usage for the shop button due to the upgrades system not being implemented into the game.
- The money for the player, \$ sign does not update with the players currency and ends up becoming overlayed.
- Sometimes, the health bar reduces a little too quickly for the player.

Overall the stakeholders were generally quite happy with the current state with the look of the game and provided some further suggestions that could be implemented in a future development of the game to improve the gameplay feel and further gameplay issues such as balancing can be fixed with the addition of upgrades.

Limitations of the project

One of the important limitation of the project is that there is a lack of compatibility of the game as the game was only really developed for devices that run on a windows computer and since the game is on a hosted website online in JavaScript, it will be unable to run offline and on other popular gaming devices such as phones or iPads since the screen for these devices are completely different from windows devices and so screen adjustments may need to be accounted for. To accomplish this, I will just have to simply, adjust every menu, button and screen to be drawn within the user’s current screen size though an issue here is that if a user’s screen is smaller than intended or the game, then the screen may become compressed into a small screen and so may be difficult to play. I

could perhaps fix this issue by using CSS by adjusting the maximum width and screen height the game screen can be when the game is run by the player.

Future Development

Overall, I think my project was somewhat successful – I managed to produce a working tower defence that whilst is not currently perfect has allowed me to significantly improve my programming skills in JavaScript and my programming knowledge and now I am much more confident from the production of the game. From the success criteria, I managed to fulfil all of the important aspects of the game as the player is now able to shoot the enemies with their mouse, gain money, lose health, and start and pause the game at will, though it is evident some bugs and balancing is required. There are still some ideas I would love to add and other ideas that are mostly ideas from the could criteria and add other ideas from my stakeholders who have recommended any improvements or additional ideas to which I could add in a future development as they would seem beneficial for the current state of my game or would be an interesting addition to add.

One the biggest features I would like to add to the game is the addition of upgrades (S7 and C7) as this was the overall majority negative feedback from my stakeholders due to the player currently having absolutely having no use for money and there also being the problem that the default tower on its own not being able to keep up the amount of enemies being created, which lead to the player being eventually overwhelmed as there are too enemies, since the player has a cooldown in which they can fire . This can all be solved by adding some upgrade buttons in the already implemented shop (C8). As for the upgrades most common upgrades that seem suitable for the tower would be able to increase the rate the player can shoot, the tower health as well as adding some further cosmetics my stakeholders have mentioned previously in the stakeholder testing section. These upgrades will require quite a bit of repetitive testing as each upgrades should cost a set balanced value and should increase when purchased by the player as they will benefit the player but at the same time I have to make sure these upgrades do not immediately give the player too much of an advantage to the point where the game becomes too easy and so I will have to individually balance each of the characteristics of the upgrades based on their cost, when they are purchased, when they are upgraded and how much of an upgrade the upgrade will provide.

During the testing of my game, it became apparent that in the future release of Beta Defence, there were some definitely some notable bugs/issues that would need to be fixed before a final release on the game can be produced. First of all it seems through all of the testing, all occasions showed during the game, a single enemy (M2), notably the first enemy in the enemies' array, would be drawn onto the canvas with a face but would have a face but not have a mouth, as soon as this enemy is killed by the tower, then the newest first enemy in the array would imitate these properties. I think this may be due to some sort of overlay issue on the first enemy where the enemy and face is being drawn over the mouth of the enemy causing this error and so I would have to check the spawn function where the enemies are being drawn to see if there is an issue when the first enemy is drawn and make some modifications so the face and mouth is applied to all the enemies instead of missing out the one enemy. Another issue found by the stakeholders is that at some point during the game, when there are multiple enemies, a single enemy will output more damage then intended, this could be due to multiple enemies being detected to be colliding with the tower mistakenly rather than just one, resulting in an increase in damage for one enemy, and so causing this error or this may be due to issues with the damage system for every enemy. I could possibly try to fix this by first testing a situation when this event happens and then check if multiple collisions are happening, if this is the case then I will need to check and perhaps rewrite the code to

the collision between enemies and the tower . Furthermore simple adjustments such a positioning the game title to fit all computer screens and modifications to the graphics of the game, perhaps I could hire an artist to produce some game models and UIs to make the game seem and feel more professional rather than the game as it is now which has a basic plain colour scheme.

Lastly, a feature I hadn't intended on implementing but may want to consider in a future development is the login system. I did mention this in the MOSCOW (W4) and would mainly purpose to save a player's progression in their game in case they wanted to pause the game and continue their game at a later time, which would be quite convenient. This would be implemented by introducing a client server system, first by creating button to create a new username and password which will store the player profile database onto the server, if the player tries to create an existing username, then it will be denied, also, an attempt limit to trying to login in each username to prevent hacking may be required. Then, once the player has logged in, and has entered the game, if they were to exit the game then the game will pause and the current game state and progression database for the player will be saved onto the server. Though one issue here is the problem if the player logs in at the same time on multiple devices may cause issues when updating the database.

That is all for the project, thank you very much for taking the time to read the development of Beta Defence.

References

- bmoren. (2022, 10 7). *P5 Collision Library*. Retrieved from Github:
<https://github.com/bmoren/p5.collide2D>
- Colchsfc. (2022, 8 20). *Screne Switching*. Retrieved from
https://moodle2.colchsfc.ac.uk/pluginfile.php/211449/mod_resource/content/0/Basic%20Scene%20Switching.pdf
- contributors, M. (n.d.). *math.atan2*. Retrieved from mozilla web developer:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/atan2
- Fandom, B. (2022, 6 20). *Experience Point Farming*. Retrieved from Bloons Fandom Wiki:
https://bloons.fandom.com/wiki/Experience_Point_Farming
- Fandom, B. T. (2022, 6 16). *Bloons Tower Defence 6*. Retrieved from Wiki Fandom:
https://bloons.fandom.com/wiki/Bloons_TD_6
- Fandom, W. (2022, 6 16). *The Tower Idle Tower Defence Wiki*. Retrieved from Wiki Fandom.
- learn.digitalharbor. (2022, 11 3). *p5.js Transformations: push() and pop()*. Retrieved from learn digital harbour: <http://learn.digitalharbor.org/courses/creative-programming/lessons/p5-js-transformations-push-and-pop/>
- manno. (2022, 10 29). *p5 health bar example*. Retrieved from p5:
<https://editor.p5js.org/manno/sketches/iX61TjgRv>
- Pythagorean Theorem*. (2022, 7 4). Retrieved from Wikipedia:
https://en.wikipedia.org/wiki/Pythagorean_theorem
- Qianqian Ye, L. L. (2022, 10 28). *p5 createVector*. Retrieved from p5.js:
<https://p5js.org/reference/#/p5/createVector>
- Train, T. C. (2016, March 16). *9.8: Random Circles with No Overlap - p5.js Tutorial*. Retrieved from Youtube: https://www.youtube.com/watch?v=XATr_jdh-44

Code

Index.html

Contains all the files stored on the server as well as some additional libraries.

```
<html>

<head>
  <!-- Enables the use of the p5 library and functions -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.1.9/p5.js"
type="text/javascript">
  </script>
  <!-- Allows enemys.js file to be accessible -->
  <script src="enemy.js" type="text/javascript">
  </script>
  <!-- Allows game.js file to be accessible -->
  <script src="game.js" type="text/javascript">
  </script>
  <!-- Allows tower.js file to be accessible -->
  <script src="tower.js" type="text/javascript">
  </script>
  <!-- Allows button.js file to be accessible -->
  <script src="button.js" type="text/javascript">
  </script>
  <!-- Includes the p5.collide2D addon library -->
  <script defer src="https://unpkg.com/p5.collide2d"></script>
  <link rel="stylesheet" type="text/css" href="style.css">
  <meta charset="utf-8" />
</head>

</html>
```

Button.js

A JavaScript file that deals with the appearance and functions of all the buttons in the game .

```
class Button {
  constructor(text, x, y, w, h, click) {
    this.x = x
    this.y = y
    this.text = text
    this.width = w
    this.height = h
    this.click = click
    //can button be clicked by the player?
    this.enabled = true
  }
  clicked() {
    //if the button can be clicked
    if (this.enabled) {
      //if current mouse position is within the buttons area
      if (mouseX > this.x - this.width / 2 && mouseX < this.x + this.width /
2 && mouseY > this.y - this.height / 2 && mouseY < this.y
+ this.height / 2) {
        // button is clicked by the player
        this.click()
      }
    }
  }
  render() {
    rectMode(CENTER)
    //makes sure text is in middle of button
    textAlign(CENTER)
    //fill button in as black
    fill(0)
    //creates an outline for the buttons
    stroke(255)
    rect(this.x, this.y, this.width, this.height)
    noFill()
    //adjusts text size in button
    textSize(30)
    //text for each button will be parsed in depending on the button
    text(this.text, this.x, this.y + 10)
  }
}
```


Enemy.js

The JavaScript file deals with the appearance and functionality of each enemy created within the game.

```
class Enemy {
  constructor(x, y, w, h) {
    this.x = x
    this.y = y
    this.w = w
    this.h = h
    //how much damage every enemy can take before it can die
    this.health = 1
    //the damage each enemy will deal when colliding with the tower
    this.damage = 1
    // sets the default speed enemies will move
    this.speed = 1
    // calculates x distance from tower to enemy
    this.DistX = towers[0].x - this.x;
    // calculates y distance from tower to enemy
    this.DistY = towers[0].y - this.y;
    //finds the distance between tower and enemy
    this.distance = Math.sqrt((this.DistX * this.DistX) + (this.DistY *
this.DistY));
    //find angle between enemy and player
    this.angle = Math.atan2(this.DistY, this.DistX)
  }

  draw() {
    //every enemy of this type is coloured red
    fill("red")
    rect(this.x, this.y, 50, 50)
    //creates an outline on each enemy
    fill("black")
    //draw eyes onto every enemy
    ellipse(this.x + 10, this.y - 5, 10)
    ellipse(this.x - 10, this.y - 5, 10)
    //draws mouth on each enemy
    line(this.x + 10, this.y + 10, this.x - 10, this.y + 10)
    stroke(1)
    // }
  }

  update() {
    //finds x angle enemy needs to move
    this.speedX = Math.cos(this.angle);
    //finds y angle enemy needs to move
    this.speedY = Math.sin(this.angle);
  }
}
```

```

//moves enemy's x coordinate based on the speed
this.x += this.speedX * this.speed
//moves enemy's y coordinate based on the speed
this.y += this.speedY * this.speed

//assumes enemies aren't colliding with the tower
let towercolliding = false

// loop through existing enemy list
for (let k = 0; k < enemys.length; k++) {
  // finds the distance between existing enemy and tower
  var towerdist = dist(this.x, this.y, towers[0].x, towers[0].y)
  // if current existing enemies position is not equal to tower
  if (towers[0] != enemys[k]) {
    // if enemy is too close to tower
    if (towerdist < 50) {
      towercolliding = true
    }
  }
  // if enemy is colliding with tower
  if (towercolliding) {
    // makes sure the enemy is colliding with the tower but doesn't
    overlap on the tower
    this.speed = 0.00001
    //deal damage to the tower based on the enemys damage per second
    health -= this.damage * 0.1
    //if the player has no more health
    if (health < 0) {
      //makes sure that the players health cannot go below 0
      health = 0
      //once player has no more health, then the player is "dead" so
      enable losing screen
      gameMode = "dead", buttons = deadButtons
    }
  }
}
}
}
}
}

```

Game.js

The JavaScript file deals with the main game, storing and running all the important functions of the game .

```
var buttons = []
var pauseButtons = []
var menuButtons = []
var gameButtons = []
var deadButtons = []
var shopButtons = []
//stores the current game mode the player starts on, that being the menu
var gameMode = 'menu'
var towers = [];
var towersRange = [];
var enemys = [];
var projectiles = [];
var manualProjectiles = []
var shopButtons = []
//the time since the tower has last shot
var lastFired = 0
//how long can it be until the tower can shoot again
var cooldown = 1000
//the time since the player has last shot
var manualFire = 0
//how long can it be until the player can shoot again
var manualCooldown = 1300
//the time it has been since an enemy has been created
var lastSpawned = 0
//how long it can be until another enemy can be created
var spawnCooldown = 750
// stores the default amount of money player has when beginning the game
var money = 0
//stores the default value of health the player begins with
var health = 400;
//the maximum amount of health the player can have
var maxHealth = 400;
//the beginning number of enemies can be created
enemysSpawned = 1

// loads in image for menu background
function preload() {
  //stores the menu image as a variable so can be accessed
  start = loadImage("background.png");
}

function setup() {
  //creates a canvas that will cover whole screen
  createCanvas(windowWidth, windowHeight)
  //stores every button relating to the menu
```

```

    menuButtons = [
        new Button("Start", windowWidth / 2, windowHeight / 2 + 100, 250, 100, ()
=> { gameMode = 'play'; buttons = gameButtons; background(50) })),
    ]
    //stores every button relating to the game
    gameButtons =
        [new Button('⚙️', windowWidth - 50, 60, 50, 50, () => { gameMode =
'pause'; buttons = pauseButtons })),
        new Button('🛒', windowWidth - 130, 60, 50, 50, () => { gameMode =
'shop'; buttons = shopButtons }))
    ]
    //stores every button relating to the pause menu
    pauseButtons = [
        new Button("Continue", windowWidth / 2, windowHeight / 2, 250, 50, () => {
gameMode = 'play'; buttons = gameButtons })),
        new Button("Quit", windowWidth / 2, windowHeight / 2 + 100, 250, 50, () =>
{ gameMode = 'menu'; buttons = menuButtons })),
    ]
    //stores every button relating to the losing screen
    deadButtons =
        [new Button("Play Again?", windowWidth / 2, windowHeight / 2, 250, 50, ()
=> { gameMode = 'replay'; buttons = gameButtons })),
        new Button("Quit", windowWidth / 2, windowHeight / 2 + 100, 250, 50, () =>
{ gameMode = 'menu'; buttons = menuButtons })),
    ]
    //stores every button relating to the shop
    shopButtons =
        [new Button("X", windowWidth - 50, 50, 70, 70, () => { gameMode = 'play';
buttons = gameButtons })),]

    //loads in the menu buttons first since this is the first screen player will
be on
    buttons = menuButtons
    //
    towers.push(new Tower(this.x, this.y,));
    towersRange.push(new TowerRange(this.x, this.y,))
}
function mousePressed() {
    for (b of buttons) { b.clicked() }
}
//reset global variables to default values
function reset() {
    enemys.length = 0
    projectiles.length = 0
    manualProjectiles.length = 0
    health = maxHealth
    money = 0
    enemysSpawned = 1

```

```

}

function draw() {
  //when player clicks button, switch game modes
  switch (gameMode) {
    case 'menu':
      drawMenu()
      drawTitle()
      break
    case 'play':
      drawGame()
      break
    case 'pause':
      drawPause()
      break
    case 'dead':
      drawDead()
      break
    case 'replay':
      reset()
      gameMode = "play"
      drawGame()
      break
    case 'shop':
      drawShop()
      break
  }

  for (b of buttons) {
    b.render()
  }
}

//when the player has no more health, draw the losing screen
function drawDead() {
  background(255, 255, 255, 0.6)
  fill("white")
  fill("black")
  rect(windowWidth / 2, windowHeight / 2, 500, 500)
  textSize(100)
  fill("white")
  text("You Died!", windowWidth / 2, windowHeight / 2 - 125)
}

//draws the menu screen
function drawMenu() {
  //when player is in menu, reset the game
  reset()

```

```

    //fills in the background by the image stored in variable start
    background(start)
}

//when game starts, draw the game
function drawGame() {
    background("black");
    fill("white")
    stroke("black")
    strokeWeight(1)
    textSize(20)
    text("Health", 100, 20)
    text("Money", 400, 20)
    textSize(30)
    text(money, 390, 60)
    text("$", 430, 60)
    rectMode(CORNER)
    stroke("white");
    strokeWeight(2);
    noFill();
    rect(10, 40, 200, 20);
    noStroke();
    fill("red");
    rect(10, 40, map(health, 0, maxHealth, 0, 200), 20);
    rectMode(CENTER)
    //enables enemies to be drawn
    spawn()

    //draws the towers range
    for (r of towersRange) {
        r.draw()
    }
    //draws the tower
    for (t of towers) {
        t.draw();
    }
    //for every enemy in the enemies array
    for (e of enemys) {
        //draw the enemy
        e.draw();
        //update the enemies position
        e.update();
    }
    // for every projectile in the projectiles array
    for (p of projectiles) {
        p.draw();
        //update the projectiles position
        p.update();
    }
}

```

```

        //checks if projectile has collided with some enemies
        p.hashitenemy()
    }
    // for every projectile shot by the player
    for (m of manualProjectiles) {
        //draw the projectile
        m.draw()
        //update the projectiles position
        m.update()
        //check if the projectile has collided with some enemies
        m.hashitenemy()
    }
}
// draws the main title of the game
function drawTitle() {
    fill("gold")
    textSize(100)
    stroke("black")
    text("Beta Defence", 800, 320);
}

//draws the pause screen when the pause button is clicked
function drawPause() {
    background(255, 255, 255, 0.6)
    fill("white")
    fill("black")
    rect(windowWidth / 2, windowHeight / 2, 500, 500)
    textSize(100)
    fill("white")
    text("Paused", windowWidth / 2, windowHeight / 2 - 125)
}

//draws the shop screen when shop button is pressed
function drawShop() {
    background("black");
    fill("white")
    textSize(100)
    text("Shop", windowWidth / 2, 100)
}

function spawn() {
    //loops through enemies array until a set number enemies are created whilst
    also drawing these enemies every 2 seconds
    while (enemys.length < enemysSpawned && millis() > lastSpawned +
spawnCooldown) {
        // assumes enemy is not colliding
        colliding = false
        // generate a random x coordinate for enemy

```



```

var xco = (windowWidth * Math.random())
// generate a random y coordinate for enemy
var yco = (windowHeight * Math.random())

//loops through enemies currently to see if they are drawn within the
towers area
for (let i = 0; i < enemysSpawned; i++) {
  //checks if enemys are created within the towers radius
  var check = dist(xco, yco, windowWidth / 2, windowHeight / 2)
  //if enemies are created within the towers radius then
  if (check < 400) {
    //generate a new random x coordinate
    xco = (windowWidth * Math.random())
    // generate a new random y coordinate
    yco = (windowHeight * Math.random())
  }
}

// loop through existing enemy list to see if current enemy being drawn is
too close to existing enemy

for (let j = 0; j < enemys.length; j++) {
  // finds the distance of enemy that is going to be drawn with existing
ones
  var d = dist(xco, yco, enemys[j].x, enemys[j].y)
  // if enemy is too close - within a distance of 100 of each other don't
create the enemy
  if (d < 100)
    // colliding flag set to true so enemy cannot be drawn
    colliding = true
}
// if enemy is not colliding with another enemy, then create enemy in that
position
if (!colliding) {
  enemys.push(new Enemy(xco, yco))
  //stores the last time a previous enemy was created
  lastSpawned = millis()
}
}
}

function checkHealth() {
  //if the player has no more health
  if (health < 0) {
    //then pause the game
    gameMode = "pause"
  }
}

```

```

function mouseClicked() {
  //if the player has fired a projectile and the cooldown has passed
  if (millis() > manualFire + manualCooldown) {
    //create a new projectile at the towers position
    manualProjectiles.push(new manualProjectile(towers[0].x, towers[0].y))
    //store the amount of time since the player has last fired
    manualFire = millis()
  }
}

function inRange() {
  // only check if the time now is more than COOLDOWN milliseconds since last
  event

  if (millis() > lastFired + cooldown) {

    //loops through enemy array
    for (let i = 0; i < enemys.length; i++) {
      //if enemy is in range, then inRange will be set to true and so will
      fire a projectile
      let inRange = collideRectCircle(enemys[i].x, enemys[i].y, 50, 50,
        windowWidth / 2, windowHeight / 2, 400);

      //checks if enemy is in range and if enemy is being targeted by the
      tower
      if (inRange && !enemys[i].targeted) {
        //fire new projectile towards enemy
        projectiles.push(new Projectile(towers[0], enemys[i]))
        // note the time this happened, and exit the function
        lastFired = millis()
        return
      }
    }
  }
}

```

Style.css

The CSS file removes the default scroll bar so it doesn't interfere with the main game.

```
body{  
    /* hides the scroll bar */  
    overflow: hidden;}
```

Tower.js

The JavaScript file stores the functions and appearance of the tower object in the game.

```
class Tower {
  constructor() {
    //position the tower in the middle of the player's screen
    this.x = windowWidth / 2;
    this.y = windowHeight / 2;
    //stores position of the tower
    this.pos = createVector(this.x, this.y)
  }
  //draws the body of the tower and the cannon of the tower
  draw() {
    //only move the tower and nothing else
    push()
    //positions the cannon with the tower
    rectMode(CENTER)
    //stores position of players cursor position
    this.mouse = createVector(mouseX, mouseY)
    //finds distance between players mouse and towers position
    this.angle = p5.Vector.sub(this.pos, this.mouse).heading()
    //fills in main body of tower in cyan
    fill("cyan");
    //create an outline for the tower
    stroke("black")
    //the body of the tower
    circle(windowWidth / 2, windowHeight / 2, 50);
    //makes sure the tower cannon is drawn with the main body of tower
    translate(this.x, this.y)
    fill("silver")
    //create an outline for the tower
    stroke("black")
    //rotates the tower cannon with an offset according to the players cursor
    position
    rotate(this.angle - 80.1)
    //draw cannon for tower
    rect(0, 30, 30, 55)
    pop()
  }
}

//how far the tower can technically "see" before it can automatically shoot an
enemy
class TowerRange {
  constructor(x, y, r) {
    this.x = x;
    this.y = y;
    this.r = r;
  }
}
```

```

// draws the towers range as a small red outline of a circle
draw() {
  //towers range is filled in red
  stroke("red")
  //adjusts how thin the towers range is
  strokeWeight(2)
  //gives outline for tower range
  fill("black")
  //the range of the tower, drawn using the x,y and radius
  circle(windowWidth / 2, windowHeight / 2, 400);
}
}

class Projectile {
  constructor(src, tgt) {
    //the speed the projectile will be moving at
    this.speed = 1
    //stores position of projectile through coordinates parsed in
    this.pos = createVector(src.x, src.y)
    //finds the distance between enemies and projectiles
    this.bv = createVector(tgt.x - src.x, tgt.y - src.y)
    //sets the speed at which the projectile when moving towards enemies
    this.bv.setMag(this.speed)
  }

  //drawn the projectile at towers position and fill in white
  draw() {
    //colour projectile white
    fill("white")
    //projectile will be a small ellipse at the position of the tower
    ellipse(this.pos.x, this.pos.y, 15)
  }
  //update the projectiles current position
  update() {
    this.pos.add(this.bv)
  }
  //checks if projectiles have collided with enemies
  hashitenemy() {
    //loops through enemy array to check if any enemy has collided
    for (let i = enemys.length - 1; i >= 0; i--) {
      //checks if an enemy is currently colliding with a projectile
      let enemyHit = collideRectCircle(enemys[i].x - 25, enemys[i].y - 25, 50,
50, this.pos.x, this.pos.y, 15)

      //if an enemy is hit by a projectile
      if (enemyHit) {
        console.log("colliding?", enemyHit);
        //remove that specific enemy from the enemies array

```

```

        enemys.splice(i, 1)
        //remove that specific enemy from the projectiles array
        projectiles.splice(projectiles.indexOf(this), 1)
        //increase the players current money by 100 when an enemy has been
killed
        money += 100
        //increase the number of enemies that can be "spawned" by 1
        enemysSpawned++
        return true
    }
}
return false
}
}

class manualProjectile {
    constructor(x, y) {
        this.x = x
        this.y = y
        //the speed the projectile will move when fired
        this.speed = 2
        //calculates the x distance between the mouse and the tower
        this.DistX = mouseX - towers[0].x;
        // calculates y distance from mouse to the tower
        this.DistY = mouseY - towers[0].y;
        //find angle between mouse and tower
        this.angle = Math.atan2(this.DistY, this.DistX)
    }

    draw() {
        //colours the projectile white
        fill("white")
        //gives the projectile an outline
        stroke("black")
        //draws the projectile as a small circle
        ellipse(this.x, this.y, 15)
    }

    update() {
        //finds the x angle projectile needs to move
        this.speedX = Math.cos(this.angle);
        //finds y angle projectile needs to move
        this.speedY = Math.sin(this.angle);

        //moves enemy's x coordinate based on the speed

```

```

    this.x += this.speedX * this.speed
    //moves enemy's y coordinate based on the speed
    this.y += this.speedY * this.speed
  }
  hashitenemy() {
    //loops through enemy array
    for (let i = enemys.length - 1; i >= 0; i--) {
      //checks if projectile is currently colliding with an enemy
      let enemyHit = collideRectCircle(enemys[i].x - 25, enemys[i].y - 25, 50,
50, this.x, this.y, 15)

      //if an enemy is currently colliding with an enemy
      if (enemyHit) {
        console.log("colliding?", enemyHit);
        //remove that specific enemy
        enemys.splice(i, 1)
        //remove that specific projectile
        manualProjectiles.splice(manualProjectiles.indexOf(this), 1)
        //increases players current money by 100
        money += 100
        //increases number of enemies that can be created by 1
        enemysSpawned++
        return true
      }
    }
    return false
  }
}

```