ANALYSIS STAGE

Introduction

My idea is a 2D, top-down tank game with puzzle elements where the player controls a small tank that can move in any direction and fire projectiles at enemies / obstacles using a rotating turret. The goal of the game is to navigate through and fight off enemies in a variety of levels which will get increasingly more difficult due to the changing design of the environment and larger amounts of more advanced enemies. The challenge and puzzle elements of the game will arise from the dynamic situations that the physics of the projectiles can create, the player will be able to use these physics to their advantage but so may the enemies. The enemies in the game will also take the form of small tanks with small aesthetic changes to clearly signify and telegraph to the player what type of enemy they are so the player can adjust their strategy. The player will also be able to obtain upgrades or power-ups as they progress through the levels that will enhance their fighting abilities.

The game will feature a menu screen which will allow the player to select several different difficulties that will either strengthen or weaken the AI so the experience can be tailored to the skill level of the player. The number of levels is subject to change, so the menu may also contain a level selection screen. However, I am aiming for around 4-5 levels which will be played chronologically so this feature may not appear in the final product. The final level will be a boss fight that the player will use all their accumulated skill and abilities to defeat, ideally the different difficulties will each have different flavour for this fight or potentially completely different bosses all together.

The current working name of my game is ‘Tank Trouble’ which sums up the overall theme of the game. The concept is inspired by the flash games of the 2010s, notably ‘Tiny Tanks’ and the game ‘Tiny Tanks!’ (Which is different) on Roblox. I found these games enjoyable and challenging back when I used to play them, and hopefully the game will be reminiscent of that experience both gameplaywise and aesthetically. Research will have to be done on the target audience of this game because whilst I found similar games enjoyable up to the age of roughly 10 years old, I do believe this game has the potential to appeal to audiences close my current age or older if appropriate additions / changes are made.

Research

The game Tiny Tanks is the primary inspiration for my project and will have the most similarity.

A video game screen with tanks and text

AI-generated content may be incorrect.

1. The main menu screen for the game Tiny Tanks at [Tiny Tanks - Play on Armor Games](https://armorgames.com/tiny-tanks-game/18063)

A video game screen with a brick wall and a brick wall

AI-generated content may be incorrect.

2. Level 3 of the game

A video game with a video game

AI-generated content may be incorrect.

3. Level 10 of the game

A video game screen with a tank and a tank

AI-generated content may be incorrect.

4. A boss fight at the end of the 3rd stage

|  |  |
| --- | --- |
| Things I’d like to include | Things I wouldn’t like to include |

|  |  |
| --- | --- |
| The simplistic art style of the main menu and levels. They fit a kind of ‘toy’ aesthetic that is visually pleasing and fits the setting of the game. It would also help save time in the design phase. | Lack of settings or difficulties. This game lacks a settings tab from the main menu or difficulty selection screen. This is something that I think would complement the game. |
| The minimalist levels allow the player to focus more on using the environment and the gameplay meaning that information is communicated quickly and easily. This is important because using the environment is a key part of the game. | Shadows and tank tracks. Whilst these are amusing additions to the game, they can distract from the main gameplay and are unnecessary time spent on a feature that does not complement the gameplay of the game. |
| Multiple distinct enemy types, each with a different colour and small but noticeable change to their design which communicates which type of enemy they are and what to expect from them. | Some enemies have colours that are too similar to each other or have designs that look virtually the same, this could lead to confusion during the game over which approach to take when fighting them. |
| Boss fights. These are a satisfying way to end a stage, or different difficulty in my case, because they put all the skills the player learned to the test. | Too many boss fights, the game Tiny Tanks is played in a linear way through several different stages which all have a unique boss at the end. Whilst this is a good way to layout a game, I would like to provide more difficulties but less levels to save time and provide more replay ability. |

The game Commando is a classic flash game which provides a somewhat similar experience.

A screenshot of a video game

AI-generated content may be incorrect.

1. The main menu

A video game with a cartoon character and a military vehicle

AI-generated content may be incorrect.

2. First level of the game: [Commando - Play on Armor Games](https://armorgames.com/play/141/commando)

A video game screen with a cartoon military vehicle and a wooden fence

AI-generated content may be incorrect.

3. Collectibles and upgrades

|  |  |
| --- | --- |
| Things I’d like to include | Things I wouldn’t like to include |

|  |  |
| --- | --- |
| The pixel art style is heavily associated with early 2010s and flash games as a whole, I’d like to take inspiration from this theme for my game. | The art style is too detailed and complicated for a game like mine, but still is a good basis for a similar, simpler aesthetic. |
| A score system, this provides more replay ability for the game as it challenges the play to best themselves and get a better score each time they play. | Side scrolling levels, they don’t fit my game’s premise. It would be preferable to use arena style levels that are always fully in view of the player. |
| Special alternate weapons. When the player presses the space bar they expend their special ammo and do a powerful attack. I’d like to include something similar that allows the player to shoot a missile or lay a mine etc.. | Health bar system. I’d rather add armour or powerups that make the player stronger, health wouldn’t fit with the short level design of my game. |
| Collectibles and upgrades found during the game. When an enemy dies, there is a chance for them to drop a special ammo and there are upgrades littered throughout the game. |  |

The game Tiny Tanks! Which is a Roblox game inspired by my first example, was a game I liked to play when I was younger and takes several of the elements of the original and puts it in a multiplayer framework.

A screenshot of a video game

AI-generated content may be incorrect.

1. A match of the game found on Roblox.com

A screenshot of a video game

AI-generated content may be incorrect.

2. Example of combat and player death

|  |  |
| --- | --- |
| Things I’d like to include | Things I wouldn’t like to include |

|  |  |
| --- | --- |
| The toy art style is a key focus of this game and the whole game takes place inside of a toy box with a plastic environment. I think this would provide a good premise for many of the arenas in my game whilst keeping the design relatively simple for gameplay reasons. | The player’s view is too small to fully view the map; this does not fit my game because the player needs to be able judge the movements of the enemies. |
| A score system, this provides more replay ability for the game as it challenges the play to best themselves and get a better score each time they play. | Multiplayer functionality wouldn’t fit with my game due to time limitations, added complexity and unavailability through greenfoot. |
| Different types of tanks with different playstyles. Different tanks with different strengths and weaknesses would help to make the game more replay able and allow the player to find their personal playstyle. | 3D graphics, I believe, are too complex for my project and are not supported by greenfoot. 3D would also not fit my specific circumstances as they would be too detailed and distracting for the player. |
| Collectibles and upgrades gained from killing enemies. |  |

Stakeholders

The target demographic for my game would primarily be males who are relatively inexperienced gamers between the ages of 10 – 17. The game will fit this group because despite the increasing difficulty, the controls will be easy to pick up and much of the game’s combat will require more thought than pure muscle memory. The game intends to provide entertainment as this is the age group that is perhaps the most prone to boredom but can also help with hand to eye coordination and train logical thinking as the player will have to carefully consider their approach to each level.

It is possible that some in this group would be unfamiliar with a keyboard, which is why the controls are very simple. They will be similar to any flash game or simple pc game (WASD and mouse) which means that even inexperienced players will be able to pick up the game and play even without a tutorial, but I will have to survey the ages 11 - 17 to find the best combinations to suit them.

I will conduct my research mostly on students of college and secondary age to help with the design process, finding the most convenient but also most enjoyable feature to include in the game.

Results of my survey:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

User requirements

These are partially based upon the results of the stakeholder survey. My opinions of certain features do differ but are heavily influenced by the opinions of the stakeholders.

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Explanation | Justification | Importance |
| Main menu | A screen that will appear when the player first loads up the game. | Needed for the player to access difficulties, options or any other pregame features before the game begins. | High |
| Options menu | A menu accessible from the main menu that contains game options such as volume, controls etc.. | Important so that the player can tailor the game to their needs and adjust any settings if they desire. | Medium |
| Difficulties | Several options that change the level of the enemies and the abilities that they are capable of. | Creates valuable replay ability to the game to lengthen play time and provide a greater challenge to more skilled players. I consider this to be quite important but was not highly valued by the stakeholder survey. | Medium |
| Level selection | A menu where the player can select a level in which they can immediately begin the game at. | Means that the player does not have to fully replay each level to get back to a section of the game they’d like to play. | Low |
| Boss fights | A high-level enemy that the player will face at the end of each stage / section of the game. | Provides a satisfying end to each section and was highly valued by the stakeholders that completed my survey. | High |
| Different types of playable tanks | Different variations of tank that the player can choose to play as. Provides different strengths and trade-offs per tank. | Allows for different playstyles and provides more replay value as the player is motivated to try different ‘builds’. However, this feature was not highly valued by the stakeholders. | Low |
| Upgrades obtainable in an upgrade tree | Upgrades for the player’s tank that can be obtained at the end of each level through a skill tree the player can invest in. | Different branches could provide different benefits or weaknesses, also provide a sense of meaningful progression and enhance the player’s abilities. | Medium |
| Pixelated art style | A pixelated appearance to the game’s environment and entities. | The most favoured art style by the stakeholders, provides an aesthetically pleasing and simple style that fits well with the game. | High |
| Powerups obtainable mid round | Powerups that are obtainable by the player mid round that provide a temporary buff. | Encourages different ways to play levels and are satisfying for the player to use. This art style was also overwhelmingly favoured by the stakeholders. | Low |
| Toy-plastic theme | A general theme that each level will take. simplistic layouts and art style. Tank models will also be designed to appear as toys. | Plays into the game well due to the simplistic theme and should not be too difficult to adhere to. Adds a unique recognisable art style. | Medium |
| Playable levels | Levels that the player can interact with through their tank, will feature enemy tanks. | Vital for the game to be functional, the game revolves around moving from level to level. | High |
| Enemy tanks | Enemies that will attack the player, their projectiles will bounce off walls and they will move around the levels. | These will be the core of the gameplay loop, with killing all enemies in each level being the main goal of the game. | High |
| Different types of enemy tanks | A variety of different enemy tanks that each have different flavour, with strengths and weaknesses. (Eg: powerful shot but slow reload.) | To add more flavour to levels and encourage the player to try different approaches to combat situations. | High |
| Destructible objects | Destructible obstacles that will appear in levels, once destroyed the player can shoot and move through the empty space. | Encourages more critical thinking, adds different strategies and more dynamics to each level. | Low |
| Projectiles that bounce off walls | Projectiles, when fired, will be given a lifespan, will travel forward and bounce off walls until it runs out of time or hits an enemy tank. | A very important feature as it adds far more critical thinking into the gameplay as the player can bounce their shots around walls to hit enemies. Adds more multidimensionality to the gameplay loop. | High |
| A turret that can rotate 360 degrees | A turret that the player can control with their mouse that rotates 360 degrees and fires projectiles when left mouse button is clicked. Enemies also require turrets. | A key feature to the game as it is essential to the combat features in allowing the player to fire quickly and easily in response to enemies. Avoids frustration from a keyboard controlled turret as mouse is more intuitive to a human player. | High |

Easy Desirable Hard

|  |  |
| --- | --- |
| Important   * Different types of enemy tanks * Toy-plastic theme * Pixelated art style * Boss fights * Turret that rotates 360 degrees * Upgrades obtainable in an upgrade tree * Difficulties | Essential   * Enemy tanks * Playable levels * Main menu * Projectiles that bounce off walls |
| Low priority   * Powerups obtainable mid round * Different types of playable tanks * Level selection | Optional   * Destructible objects * Options menu |

Less Desirable

Success criteria

Success criteria to judge against at the end of the project.

* Does my project meet the essential criteria?

1. Enemy tanks that attack the player and move around the map?

2. Multiple playable levels? (at least 3-4)

3. A main menu that the player will see when the game first starts?

4. Projectiles that bounce off walls?

5. A turret that rotates 360 degrees?

* Does my project meet the important criteria?

1. Different types of enemy tanks that have different appearances and capabilities?

2. A toy-plastic theme?

3. A pixelated art style?

4. Boss fights / boss fight where the player fights a strong enemy with unique abilities?

5. Upgrades obtainable through an upgrade tree?

6. Multiple difficulties that noticeably change the game?

* Does my project meet the optional criteria?

1. Destructible objects that when destroyed can be shot and moved through?

2. An options menu where the player can tweak settings to their liking?

* Does my project meet the low priority criteria?

1. Powerups obtainable mid round by the player?

2. Different types of playable tanks with different strengths and weaknesses?

3. A level selection menu where the player can choose a level to start playing at?

Whilst ideally most of these features could and would be included in the game, it is not realistic to pursue every single feature with equal priority due to time limitations and the limits of my programming skills. These are the expected limitations of my project and a justification on why these targets may not be met.

|  |  |
| --- | --- |
| **Limitation** | **Justification** |
| Upgrades obtainable in an upgrade tree | The actual functionality and complexity of implementing this feature is yet to be determined, but I’m not entirely sure how well this feature will fit with the game and how I’m going to be able to implement it. It would most certainly not work with a level select because a feature like this railroads the game down a mostly linear level system due to the fact that each upgrade is exclusively obtained at the end of each level and a compromise to make both features compatible would be too complex to be realistically completed within the timeframe of the project and limitations of my programming skills. |
| Level selection screen | Not compatible with the feature above due to forestated reasons. I’m unsure of how this would work with the difficulties as I intend for a smaller selection of levels and a variety of difficulties that would modify each level uniquely. |
| Powerups obtainable mid round | This feature might be made mostly obsolete and pointless by an upgrade tree. Powerups may ultimately make the game more complex and might be hard to balance in conjunction with other features. |
| Destructible objects | Whilst this feature may be relatively easy to implement, it may not fit very well with the levels depending on how they are designed in the future and designing levels around the usage of these destructible objects may be lengthy and just serve to be added complexity. |

Hardware and Software requirements

|  |  |
| --- | --- |
| **Hardware requirements** | **Software requirements** |
| -Any general purpose computer should suffice due to the low CPU requirements of my game.  -Does not have a large storage requirement.  -A mouse will be required as a key part of interacting with the game (controlling the turret).  -A keyboard is also required to control the player’s tank.  -Speakers or headphones are needed to experience the full game experience as there will be sound. But ultimately are optional.  -A monitor is required. | -Use of greenfoot means that the game will be available on any relatively recent Windows, MacOS or Linux device.  -No specific software requirements.  -User will have to download greenfoot to play. |

Computational methods

1. Decomposition

The game will be broken down into multiple smaller parts / problems that will make the game more manageable for a computer. For example, these parts will essentially be the various enemies, the levels, the player and any other objects or entities that appear in the game world. The decomposition of these sections will help with making this project a computable task and help with concurrency where the different aspects of the game can be run simultaneously to improve the speed and efficiency of the game.

1. Abstraction

Abstraction will be important in this project due to the large number of additional levels, enemy types and other features. To keep the game as simple as possible for the player, they will only be shown what is immediately relevant to them. Eg: Each level will show the player the environment of that specific level, destroyed enemies will be removed from the level and any destroyed objects will be effectively removed as well.

1. Concurrency

In the game there will always be several different tasks be carried out at the same time, for example: Enemies will move towards the player, the player will be moving both the tank and aiming its turret at the same time and there will be multiple projectiles moving around the map with different variables at play. With use of concurrency, it allows the game to move at a faster pace and reinforce the importance of being computable as a human player could not carry out so many concurrent tasks.

DESIGN STAGE

Development Plan

Stage 1: Initial design

* In this stage I will create a groundwork I can build off for the rest of the project. I aim to create the GUI; settings tab and lay the groundwork for the difficulty selection screen and level select. These will be completed first as they are all priorities because the rest of the game requires them to function.

Stage 2: Design the player and the enemies

* In this stage I will create the player's tank and make it as functional and intuitive to control as possible. The enemies will also be created, them will need to be programmed to be able to move around and fire at the player. The player and enemies' projectiles must bounce off the wall of the environment as well.

Stage 3: Design the levels

* In this stage I aim to design the layout of all the levels (3-4 levels), apply my game's intended art style to them and create solid geometry that the tank's projectiles and bounce off. The reason this stage will be done after the player and enemies is that it will be easier to first create the enemies and then adapt the environment around their quirks to make the game faster to develop and more fun for the player.

Stage 4: Test the current product

* The goal is to test the current project, which must be relatively functional. And make sure that any bugs with the more important features are fixed before any more intricate complex features are added.

Stage 5: Difficulties and level select

* Building from the placeholder difficulty select, selecting a different difficulty when starting a game will change the types of enemies that the player faces (Add in various enemy types that have enhanced stats). The level select will also be created that will give the player the ability to skip ahead to any level that they would like to play.

Stage 6: Boss fights and powerups

* These will be left until last to assure that more time will be spent on the core elements of the game. Boss fights, or possibly just one boss fight depending on time restraints, will be created where the player must fight a powerful enemy. If there is time remaining, upgrades could also be added that can be obtainable during the transition between levels and enhance the player's tank.

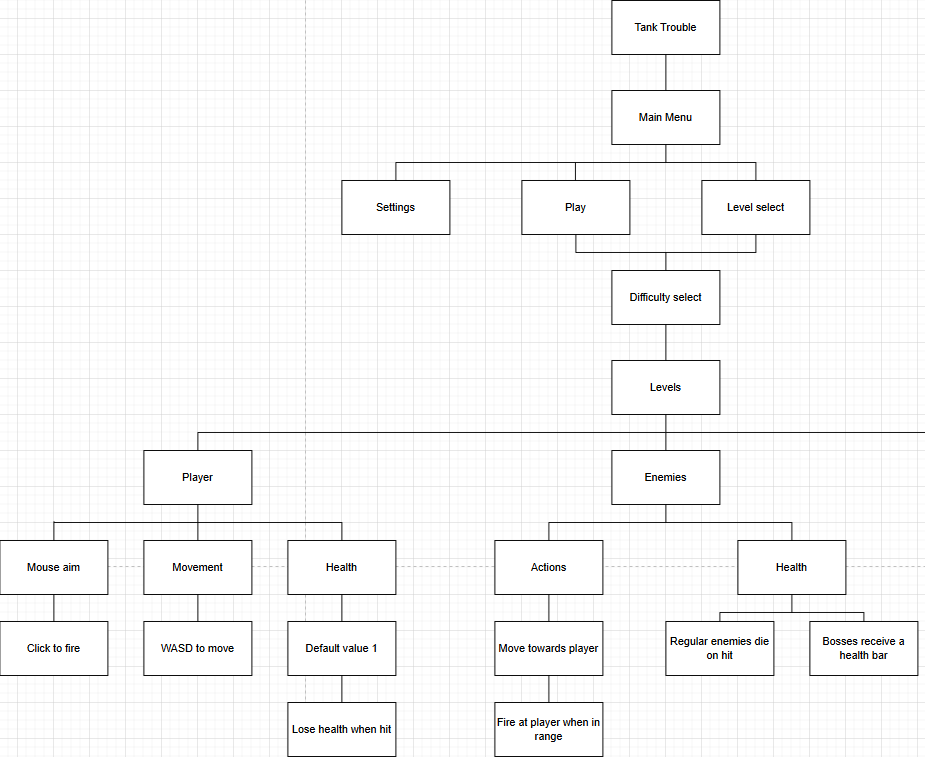
Stage 7: Additional features

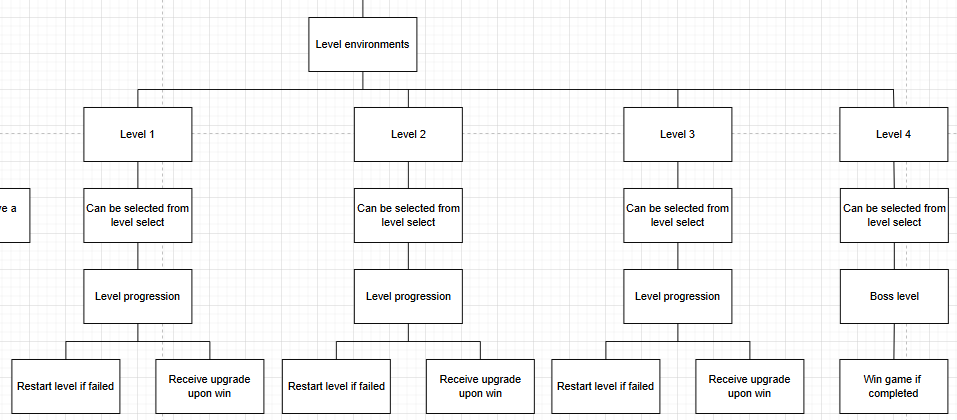
* If time is left, any additional low priority features may be added such as destructible objects in-level, a revised upgrade tree, and a settings menu.

Stage 8: Evaluation

* Evaluate the final product and make any necessary changes.

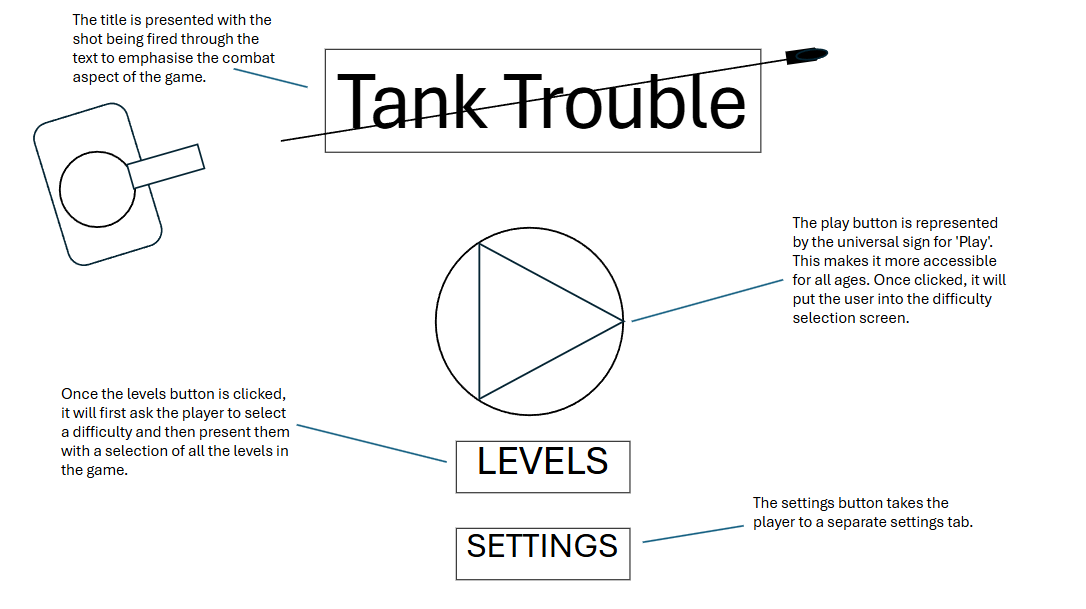
Structure Diagram

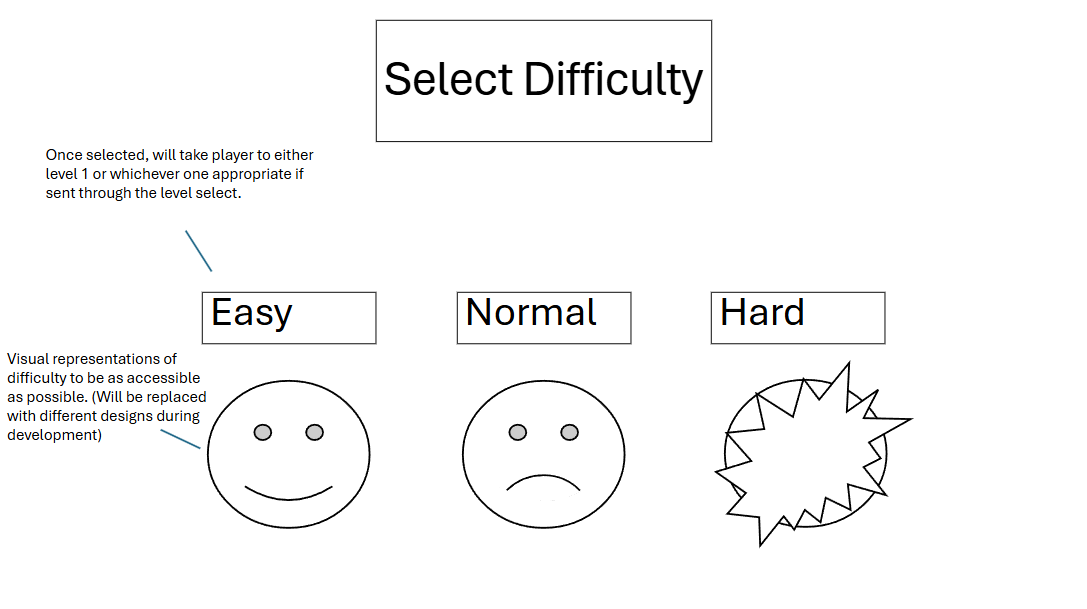


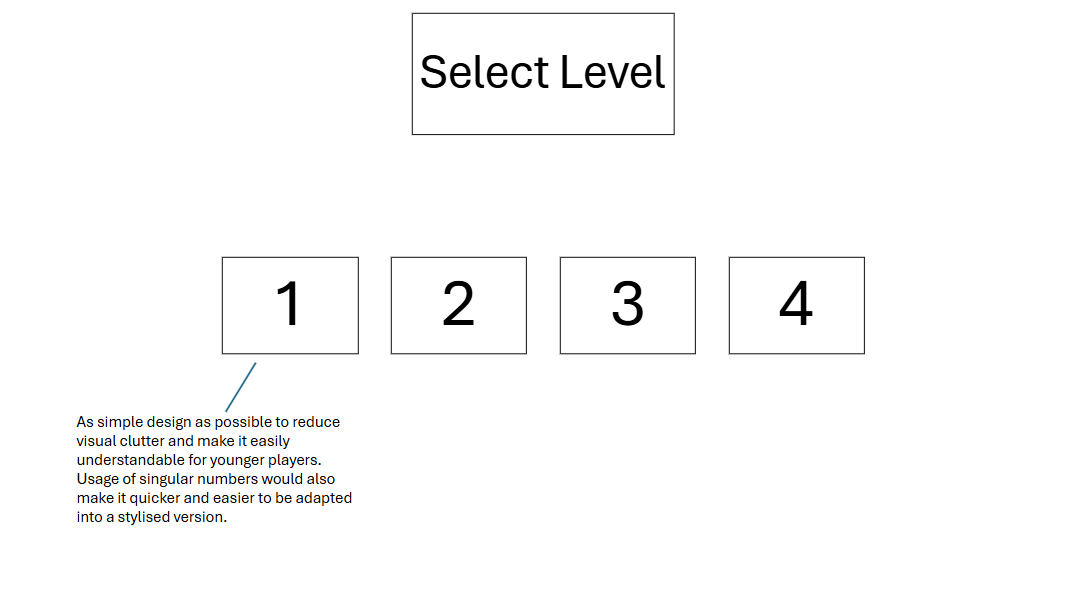


Dsffdfafadafsaasfadfasf

GUI Design







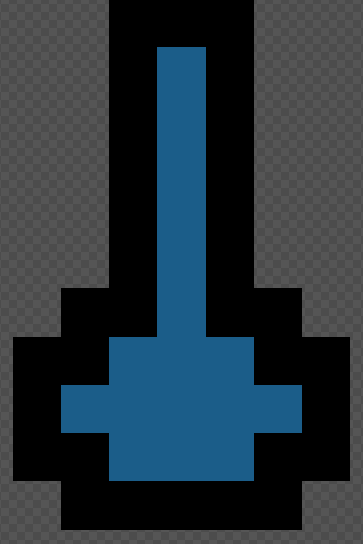
DEVELOPMENT STAGE

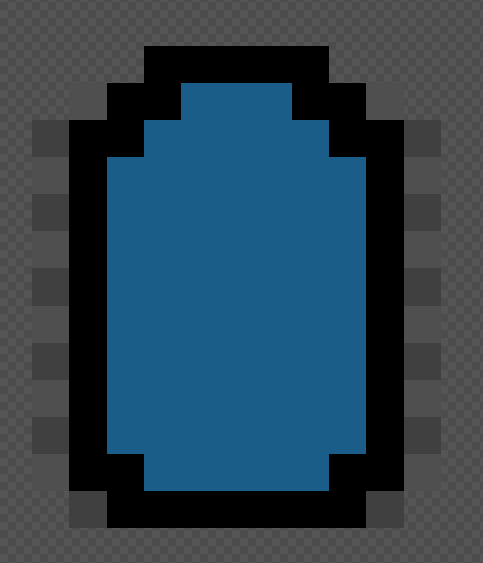
I started off my project on section 2 by designing the tank model sprites that would be used by the player and enemies throughout the game. I chose to do this first instead of section 1 first as I also require a tank sprite to put on the main menu. I used the site ‘Piskel’ to design the models before importing their pngs into Greenfoot. - Add more detail

SECTION 2

Section 2 test plan

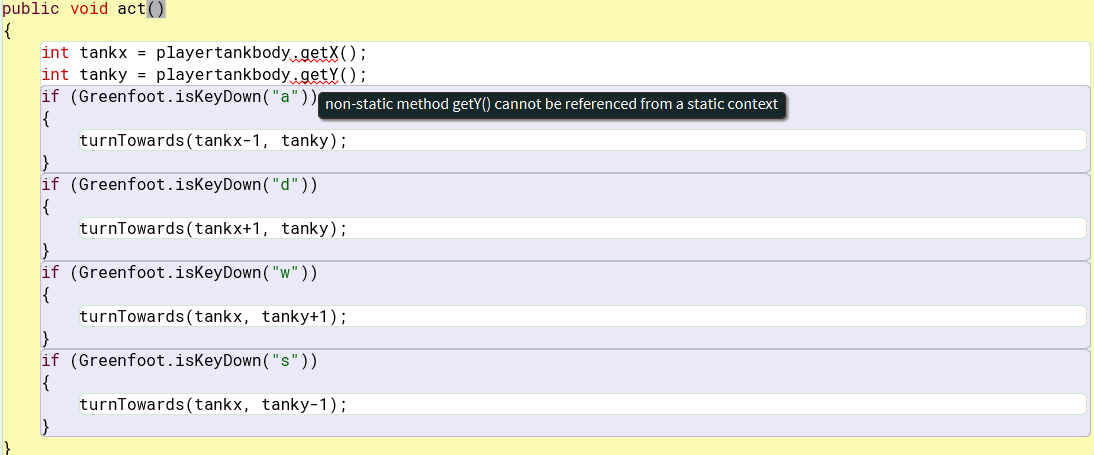
|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Expected Result** | **Real Result** | **Fail/Success** |
| Can the player move their tank using the WASD keys? |  |  |  |
| Can the player move their turret 360 Degrees? |  |  |  |
| Can the player tank shoot its gun? |  |  |  |
| Do the player’s projectiles bounce off the wall? |  |  |  |
| Do the player’s projectiles disappear after 4 seconds? |  |  |  |
| Do the enemies’ shoot at the player? |  |  |  |
| Do the enemies move around? |  |  |  |
| Do the enemies’ projectiles bounce off the wall? |  |  |  |
| Do the enemies’ projectiles disappear after 4 seconds? |  |  |  |
| Can the player destroy enemies by shooting them? |  |  |  |
| Can the player die to enemy shots? |  |  |  |
| Can the player only fire a maximum of 5 shots at a time? |  |  |  |
| Can the enemies only fire a maximum of 3 shots at a time? |  |  |  |

 The turret of the tank model

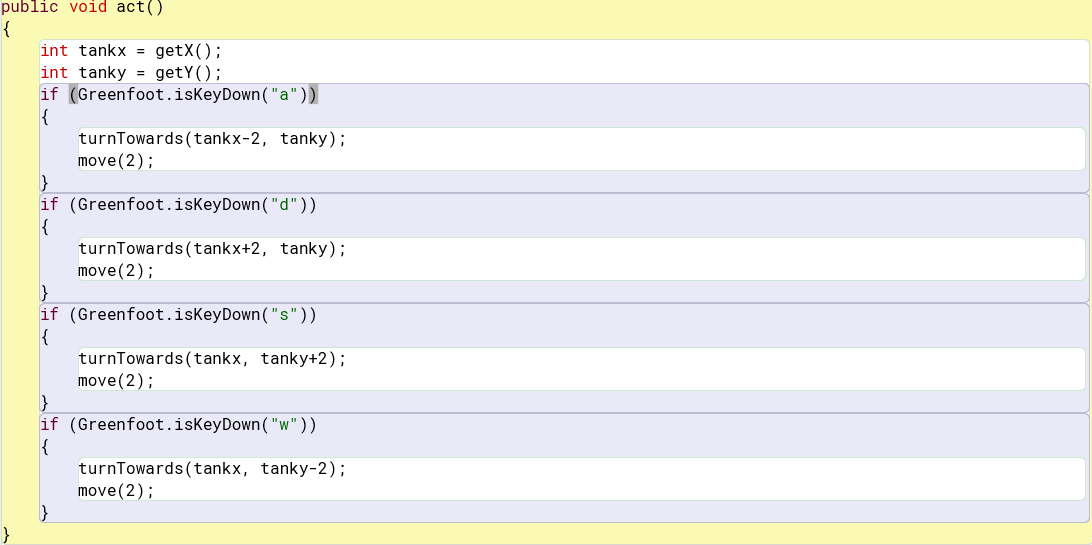


The body of the tank.

For the tank’s movement, I first tried to program a system where the tank would face whatever direction (W, A, S or D) immediately after the key was pressed instead of slowly turning using the turn() method. I did this because it would lead to more responsive feeling controls which would be important in a game that requires a quick reaction time such as this one.



I initially tried to use the playertankbody.getX() and playertankbody.getY() method to find the x and y coordinate of the tank and then get it to face the direction that would correspond with pressing the WASD controls. However, I did run into an issue where the non-static methods getX() and getY() could not be referenced from a static context.



I fixed this by changing my code to this version which also fixed a bug where the up and down movement keys were backwards (Caused by me accidentally mixing them up).

The physical tank model was also far too small on the screen (It’s just the tank body without the turret for now):

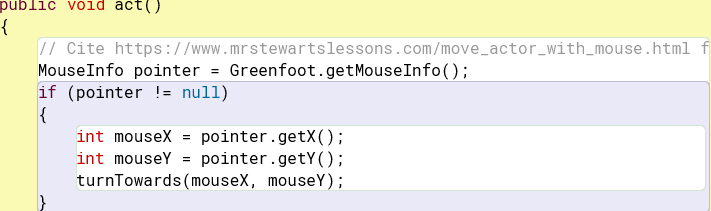


So, I fixed it using the following method to increase the scale of the model. This brings the tank in line with the vision of the scale of the levels, makes the hitbox slightly larger (for enemies as well) and makes the game easier to see, overall improving quality of life.

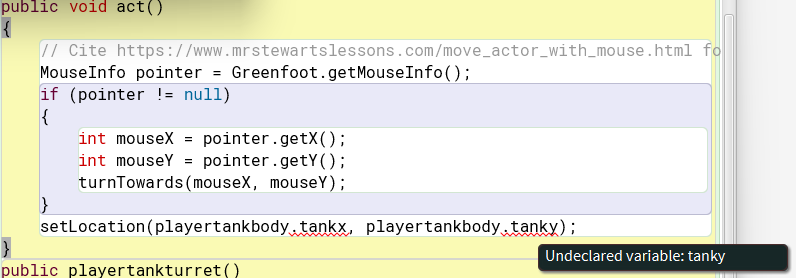


Next, I attempted to program the functionality of the tank’s turret.

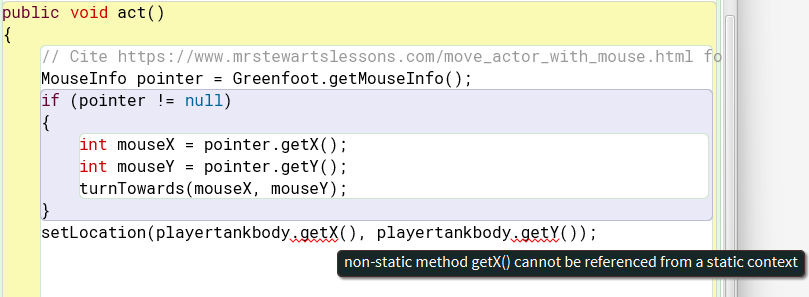
First, I went to <https://www.mrstewartslessons.com/move_actor_with_mouse.html> to learn how to turn an actor towards the mouse and adapted this code. I decided that the tank turret would be mouse controlled because it was the most intuitive and convenient solution to turret movement, allowing the player a quick reaction time and more natural feeling control over the shooting so that precise shots can be made. This will be important as the bouncing bullets mechanic is introduced, which the player will have to line their shots up for.



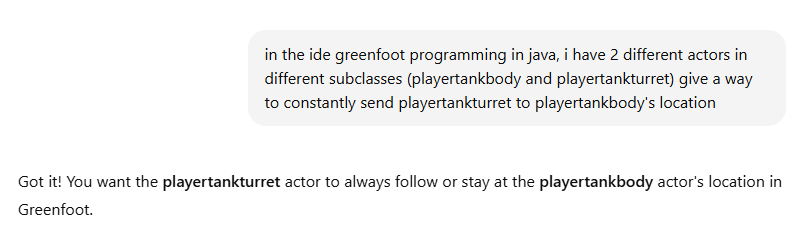
Next, I attempted to get the turret to constantly move to the location of the body so they would function as a single entity despite being 2 different actors. This is the easiest way to do this, with a possible alternative of having them being a single tank subclass, spawning in an image of a turret, programming the functionality of it in the tank body subclass and having the image face the mouse. However, I found this way to be the easiest and it was the originally planned way to program in the tanks into the game.



However, I did still run into significant issues when trying to program this into the game as the problem of referencing a non-static method from a static context prevented me from doing so for a while. I was unable to find a way to fix this method entirely using my own limited knowledge.



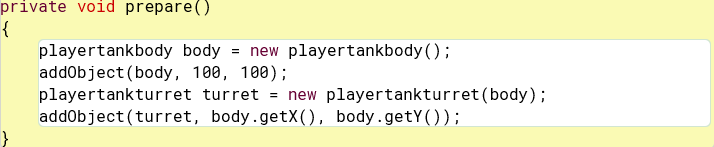
The way I fixed the problem was by going to the website [ChatGPT](https://chatgpt.com/) and inputting the exact prompt **“in the ide greenfoot programming in java, i have 2 different actors in different subclasses (playertankbody and playertankturret) give a way to constantly send playertankturret to playertankbody's location”**

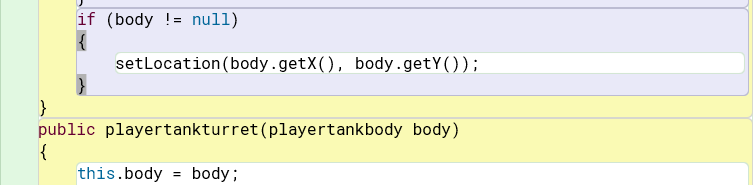


The code produced by ChatGPT is as follows:



And this was the code that I ended up adding into the game, which fixed the problem.

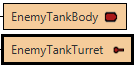




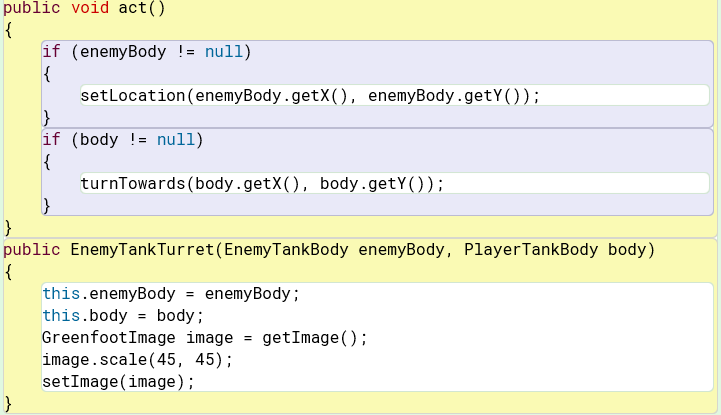
I personally would have preferred to not use AI in my project; however, I was unable to solve this problem on my own and time limitations prevented me from watching any materials that might have helped.

Next, I started working on the enemies that the player would fight in the levels I plan to add later.

First added the 2 new subclasses for the body and turret of the enemies, coloured red to make it clear for the player that these are enemies.



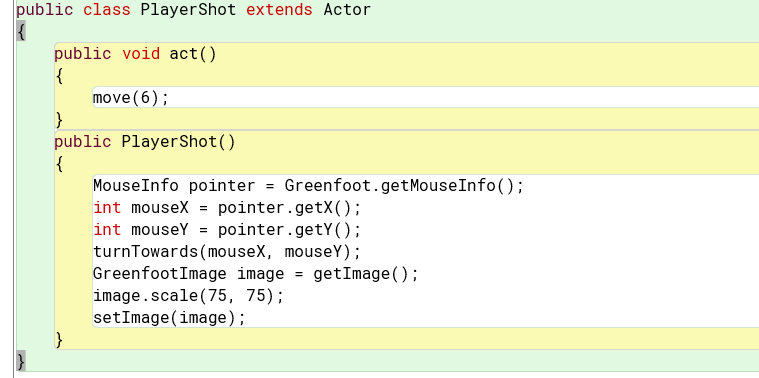
To get the enemy turret to face the player, I added a reference to the player’s tank inside of the enemy turret constructor and told it to constantly turn to the coordinates of the player’s tank.



Next, I attempted to program shooting into the game. The player would be able to fire 5 shots at a time before waiting for any of the 5 existing bullets to be destroyed to fire again. I did this by creating the new “PlayerShot” subclass and having the turret subclass instantiate a new PlayerShot object every time the space key is pressed. (Planning to change from pressing space to fire to left clicking later, but space was easier for the time being.)

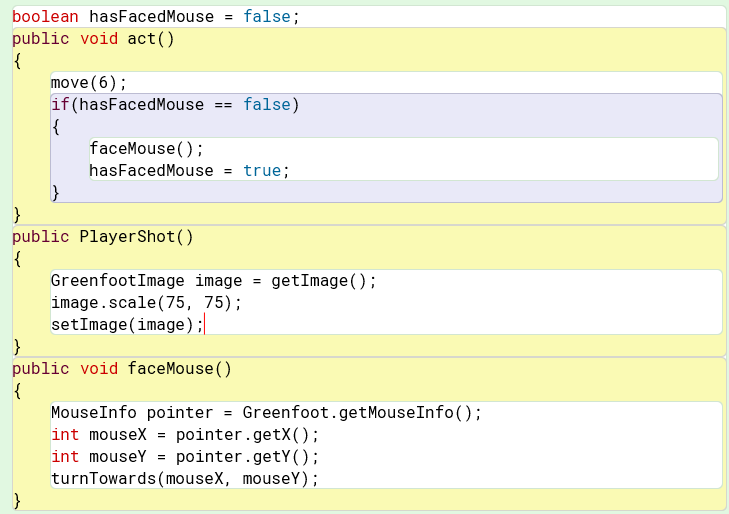


This would instantiate a new PlayerShot which I programmed like this:

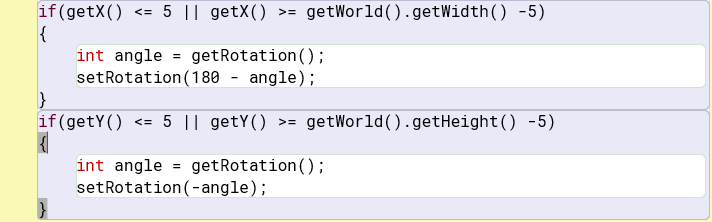


I put the “mouseInfo pointer” and “turnTowards” code into the constructor of the PlayerShot subclass so that when fired, it would fire in the direction that the player’s turret is facing, like with real tanks (And for obvious quality of life gameplay reasons). The reason I decided to put the code to fire inside of the turret subclass was that I thought it would be easier to group everything to do with the turret together and so I didn’t have to add a reference to the position of the turret in the PlayerShot subclass. However, I ran into an issue where the shots would only fire at specific angles as shown in the video **bugvideo1.**

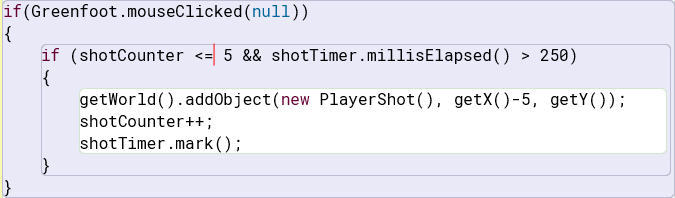
I fixed the problem by removing the code to face the mouse from the PlayerShot constructor, putting it into a separate method and having the act class call the method once (which was possible with the hasFacedMouse boolean). This is shown below:



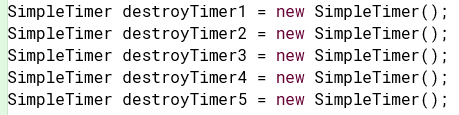
I then moved on to designing the player’s and the enemies’ shots behaviour and limiting the amount that can be fired at a time. I used the Tutorial 5: adding a randomly moving enemy tutorial in the Greenfoot documentation and also looked up the formula for deflection to help me and I got this code to make the player’s and enemies shots bounce off walls. The demonstration of this is in **demovideo1**.

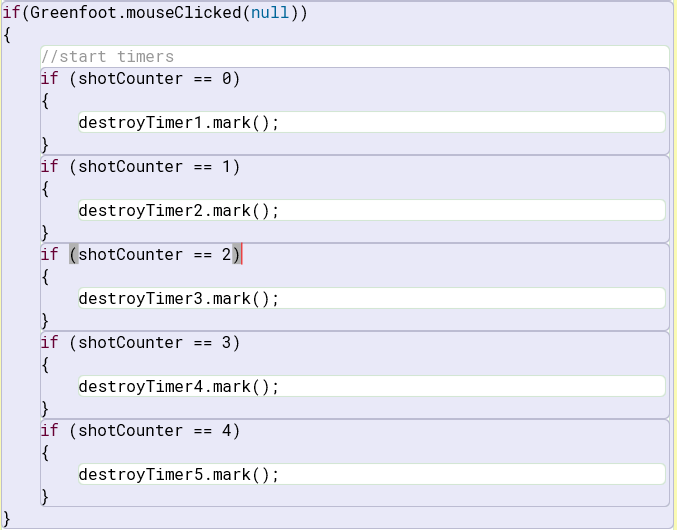


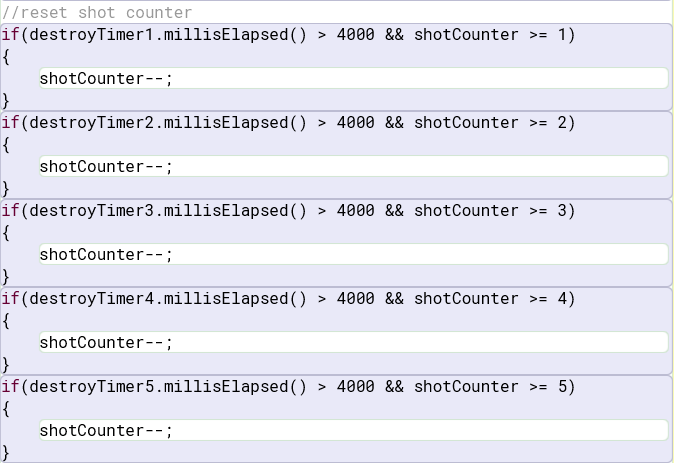
Then, I added a shot counter and a timer to prevent the player from firing too fast and too many shots. If the player was able to fire every single frame, without a maximum number of shots, the player would be able to fill up the map with shots and easily destroy the enemies. To do this, I used this ([Greenfoot Space Invaders - Part 6: Using the Timer to Limit Shooting)](https://www.youtube.com/watch?v=_opd4yPMbJI&list=PLtbMyZTNsHpojiaA93YyJrWzHCZNrTqH1&index=6) tutorial to figure out how to add a timer which would delay the shots.



This is what the firing code looks like after adding the shot delay. I then attempted to add a function that would only allow a maximum of 5 shots to be fired at any time, to do this, I added 5 separate timers to keep track of individual shots and whether they had passed the 4 second mark to guarantee that they had been destroyed.

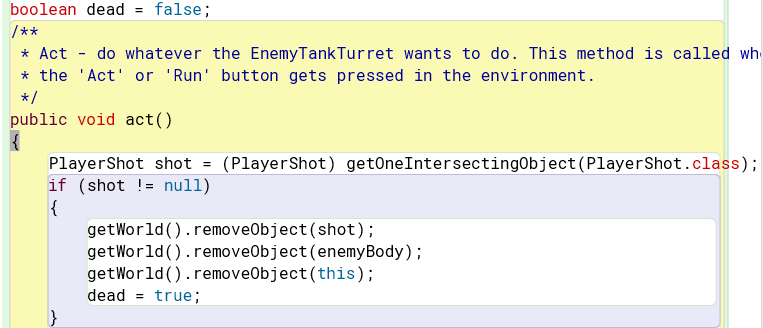






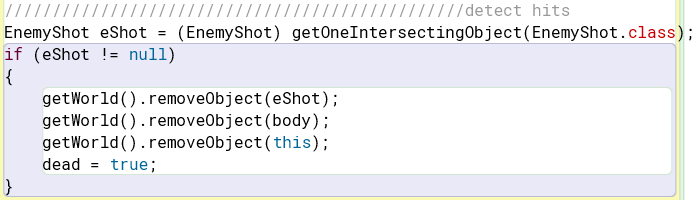
Upon implementation, I ran into a bug where more than the intended limit could be fired under certain conditions (such as spamming the mouse). However, I decided to move on as fixing this issue was not a priority and I had little time.

Next, I worked on making the enemies attack the player and detect when shots hit both the player and the enemies. I did this by using the “getOneIntersectingObject” function to find if a shot had touched the turret of the tank. I did this with the help of this tutorial: [Greenfoot Space Invaders - Part 4: Shooting and Collision Detection](https://www.youtube.com/watch?v=hRT-vi53cls&list=PLtbMyZTNsHpojiaA93YyJrWzHCZNrTqH1&index=4)

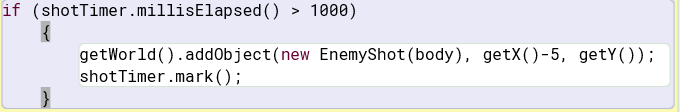


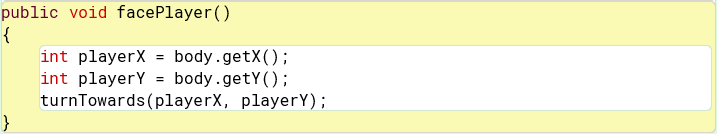
I decided to do this in the turret class because it already had a reference to the body of the tank within its constructor, allowing all of the tank’s components to be removed at once and in the same place which improved the simplicity of the code and was more time efficient.

I also added the same, modified code to make it functional for the player’s turret as well.



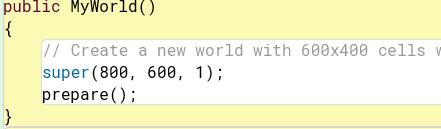
Then, I added this code to the act class of the enemy tank’s turret so that it would fire a shot at the player every second. This time was chosen to give the player some time react to the attack and also, so they don’t get overwhelmed by the existing shots bouncing around the map.





Then, I did something similar to what I did with the player’s shots by making them face the player to ensure they fire the same direction the enemy tank’s gun is facing. The enemy always faces their gun towards the player, so this works fine to give the impression that the shots are coming out of the barrel.

Additionally, I decided to expand the world from 600 x 400 pixels to 800 x 600 pixels to give the player more room to move around and dodge enemy shots and add 2 more enemies to this test area.



(Former size is in the // comment section.)

Gh

Gh

Gh

Gh

Gh