Orbital 2025

Nanban Requiem

Team ID: 7366
Team Name: Spicy Tartar Chicken Nanban Don
Luke Aidan Tan (A0307717B)
Chen Ziming (A0308962W)

Team Name

Spicy Tartar Chicken Nanban Don

Proposed Level of Achievement

Apollo 11

Liftoff Poster and Video

Poster: 7366.png
Video: 7366.mp4

Milestone 1 Poster and Video

Poster: 7366.png
Video: 7366.mp4

Milestone 2 Poster and Video

Poster: 7366.png
Video: 7366.mp4

Milestone 3 Poster and Video

Poster: 7366.png
Video: 7366.mp4

Project Log

Nanban Requiem Project Log

Past Builds

v1.0.0
Windows: Nanban Requiem (Windows).zip
macOS: Nanban Requiem (macOS).zip

v2.0.0
Windows: Nanban Requiem (Windows).zip
macOS: Nanban Requiem (macOS).zip

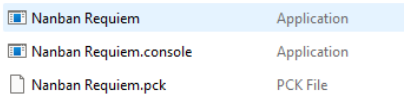
v3.0.0
Windows:Nanban Requiem (Windows).zip
macOS: Nanban Requiem (MacOS).zip

Instructions

Windows Users

1. Download the zip file from the **Current Build** section and extract it.

2. You will now see a folder named **Nanban Requiem**. Open this folder.



3. Launch the application Nanban Requiem, highlighted in the image above. If Windows Defender flags it as unsafe, click **more info** and **run anyway**.

MacOS Users

1. Download the zip file from the Current Build section and extract it.
2. Navigate to **Nanban Requiem.app/Contents/MacOS/Nanban Requiem**
3. Launch the application

Motivation

We are both huge fans of the popular 2D tower defense game, Arknights. It offers highly addictive gameplay, encouraging players to strategise and plan their defense given the situations in each map. What separates it from other common tower defense games is that it introduces unique elements into gameplay that forces you to think outside the box, juggling limited resources and defending against enemies in a time constraint, while keeping the gameplay fresh and non repetitive. We decided to try our hand at making a tower defense game that also offers this level of enjoyment, delivering an immersive experience for new and experienced tower defense players, while adding our own flair to it.

User Stories

- As a strategic player, I can deploy both melee and ranged towers with unique abilities so that I can experiment with different combinations and adapt to enemy types.
- As a player who values immersion, I can see detailed animations and visual effects for towers, enemies, and projectiles so that the gameplay feels more dynamic and engaging.
- As a challenge-seeking player, I can face waves of increasingly powerful enemies while managing limited resources, encouraging me to think critically and weigh the priorities of different choices.
- As a creative player, I can interact with the terrain/environment so that I can influence the battlefield and come up with unique strategies.

Storyline

What began as a peaceful lunch break has erupted into a full-blown lunch time war. You were just about to dig into your lunch, when all of a sudden, hordes of hungry enemies came rushing in. They have a single objective - to partake in your bowl of Spicy Tartar Chicken Nanban Don.

But you refuse. And you are not defenseless.

Every second counts. With the allies standing beside you, strategically deploy your towers across the map, be it close quarters or ranged warfare. Every tower matters. Every bite is at stake.

This is the Nanban Requiem.

Scope

Project Scope

Nanban Requiem is a 2D top-down tower defense game developed using Godot engine, where the core gameplay revolves around strategically deploying two types of towers: ranged and melee, to defend the base from waves of incoming enemies.

Description

Towers serve as the primary defense mechanism, where ranged towers attack enemies from a distance while melee towers block enemy progress and engage in close combat, with a limited block capacity. There will be a cap for the number of towers deployed at any given time to introduce critical thinking involved in understanding the enemy and using the towers' abilities to one's advantage.

Enemies consist of both melee and ranged types, navigate a predefined path using a path-following algorithm. Ranged enemies can attack towers or the base from a distance, while melee enemies must reach their target directly.

The win condition is achieved when all enemy waves on the map are cleared. The lose condition occurs if the Base HP is reduced to zero by enemies entering the base.

Tech Stack

Our game will be developed using the following tools and technologies:

Godot Engine 4.4.1

- A lightweight, open-source game engine used to build our 2D gameplay systems and visuals.

C#

- Used for implementing core game systems and logic that benefit from object-oriented structure. Integrated with Godot for performance and scalability

GDScript

- Godot's built-in scripting language, ideal for rapid prototyping and implementing core game logic.

Piskel

- Used for designing in-game animated sprites and pixel art

Git & GitHub

- Used for version control, collaboration, issue tracking, and managing the development process.

Feature Timeline

Liftoff (12 May - 19 May)

1. Designing liftoff poster and presentation video
2. Learn how to effectively version control using Git/Github through workshops

Milestone 1 - Ideation (20 May - 2 June)

1. Creation of Main Menu
2. Creation of Map 1
3. Creation of melee and ranged tower classes
4. Creation of melee and ranged enemy classes
5. Basic UI for gameplay (Build Mode, Pause/Play, Fast Forward)
6. Dedicated targeting system for towers and enemies
7. Projectile class, projectile spawning and tracking logic
8. Basic path following algorithm for enemies
9. Tower and Enemy despawn logic
10. Towers blocking enemy logic
11. Base HP and Game Over condition

Milestone 2 - Prototype (3 June - 30 June)

1. Integration of C# in addition to GDScript for OOP/SWE
2. Manage tower deployment and deployment resources
3. Creation of settings menu (volume modulation etc)
4. Addition of Map Selection Screen
5. Addition of Music soundtrack
6. Creation of more towers and enemy types/classes
7. Addition of tower skills
8. Polishing of Map 1 as well as the addition of Map 2 and 3
9. Polishing of UI and sprites

Milestone 3 - Extended system (1 July - 28 July)

1. Map 4 (Final boss stage)
2. Balancing of towers and enemies stats to make the game more challenging

3. Map 5 (Chicken Don stage)
4. Deployment Point (DP) system
5. Retreating towers
6. Polishing of game UI (viewport changes, background art and more)
7. Polishing and creation of tower/enemy sprites
8. User testing

Current Progress

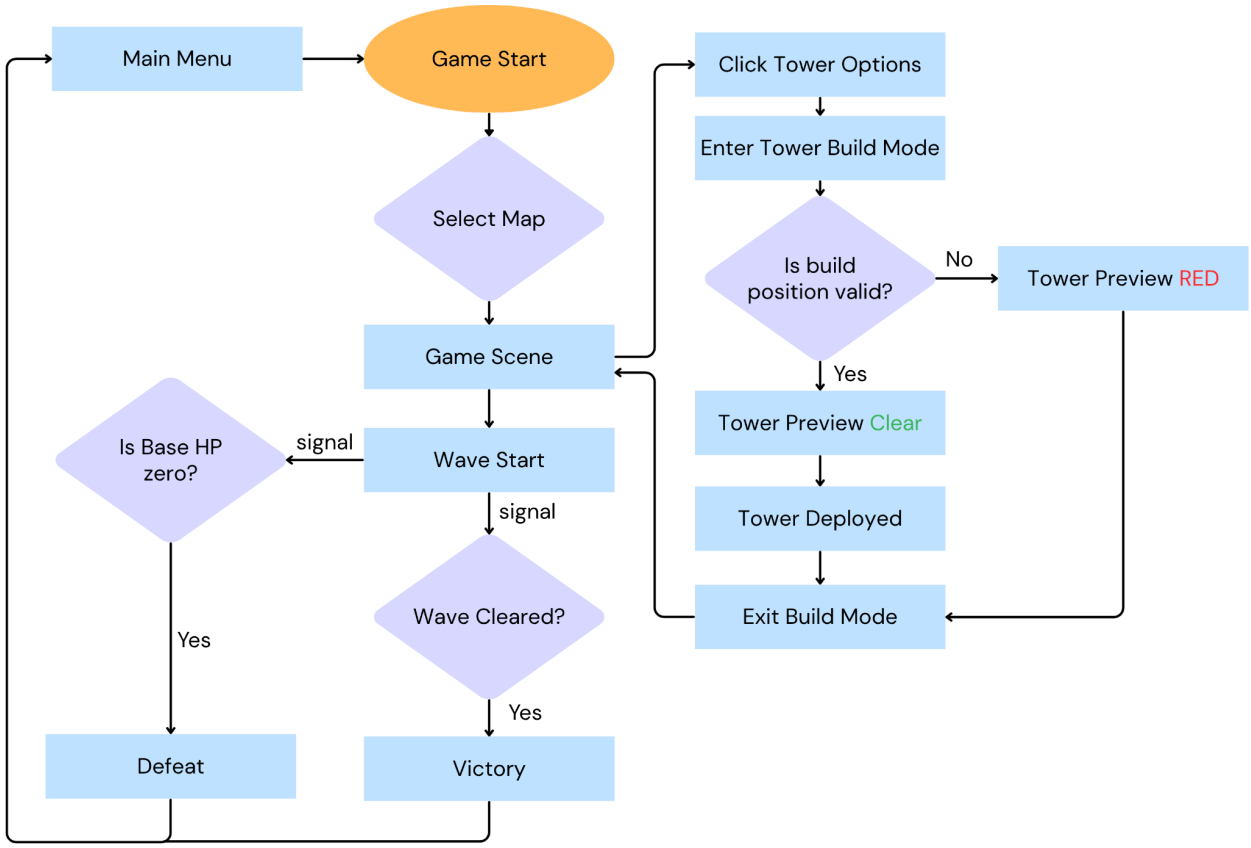
Full documentation of these features can be found in the next section.

Features	Details
Main Menu	Functional “New Game” and “Quit” buttons
Journal	Tower and enemy entries to check for lore and stats
Volume settings	Modulate master, music and SFX volume
Music	Music for the main menu and maps have been added. Music is subject to change
Map Select	Manually choose between available maps using the UI interface
Game Map	4 maps have been created 3 normal maps 2 final boss maps
Tower limit	A tower limit of 5 has been added to balance the game. Another balancing mechanic is in the works.
Towers	6 unique towers ● Pew Pew ● Rocketeer ● Boulder ● Inverse Boulder ● Ranged Tower Girl ● Nyx
Enemies	Common enemies ● Samurai ● Rocket Samurai ● Mighty Whitey Boss enemies ● Duskborne ● Voidmire

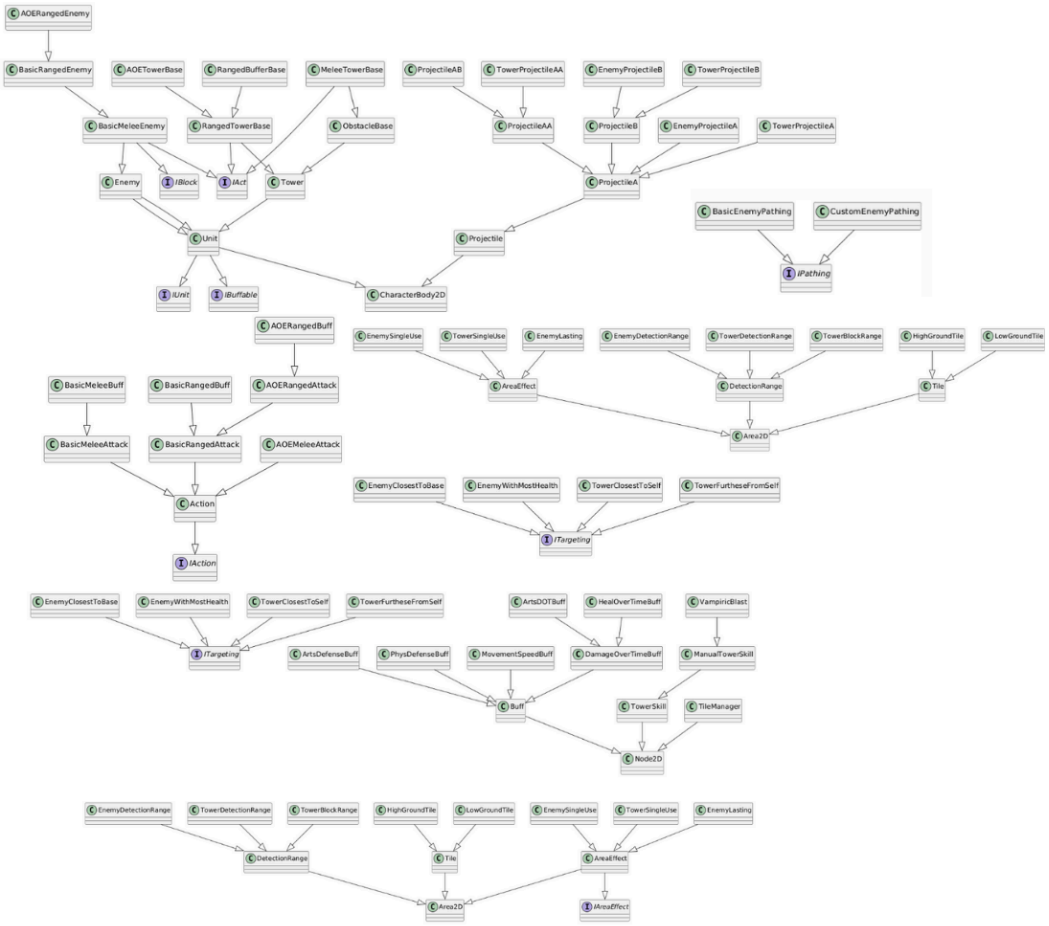
	• Priestess • Chicken Don
UI additions / improvements	Quit Window • Pop up window that allows player to confirm before quitting End Game Screen • Victory or Defeat screen upon game won or game lost
Tower UI	Towers now have an interface for skills and retreating
Boss Map (Priestess)	• 3 phases • 2 alternating bosses • 5 unique skills per boss
Boss Map (Chicken Don)	• 3 phases • 2 unique passives / skills • 100% Pure Chicken

Game Flow

Game Flow Diagram

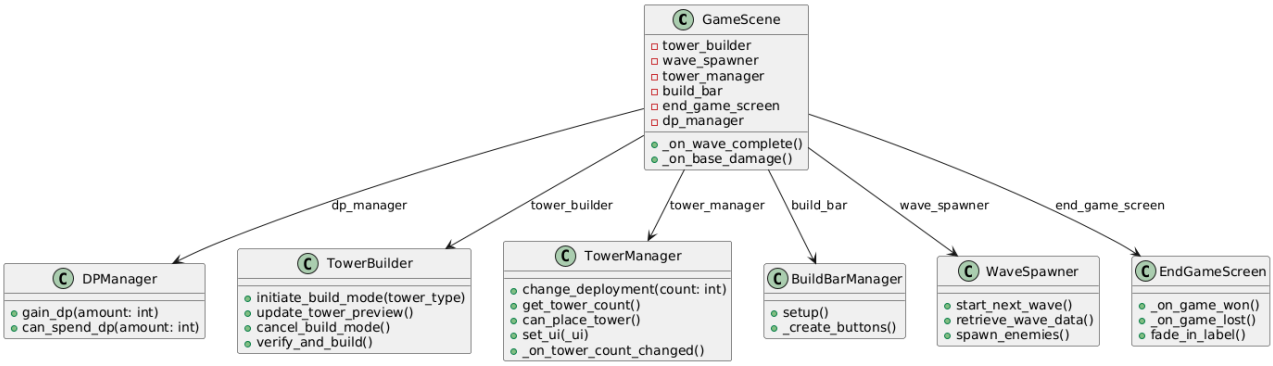


Class Diagram Overview



GameScene

GameScene is the orchestrator responsible for the gameplay experience. It acts as the root scene node that manages and coordinates all tower defense gameplay, including tower building, enemy waves, deployment mechanics and UI updates. This design architecture follows the Separation of Concerns, where each subsystem is modular and self contained.



GameScene Class Diagram

Features

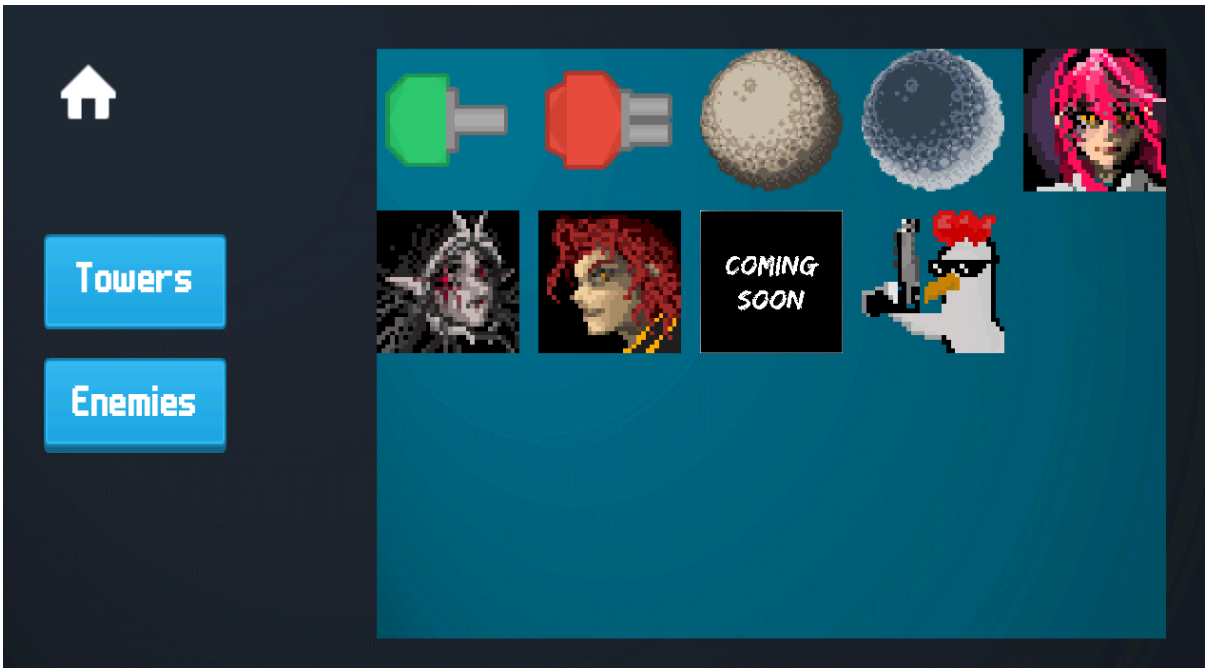
Main Menu

The Main Menu consists of the New Game, Journal, Settings, and Quit buttons.

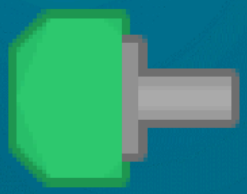
New Game loads the first map scene, where the gameplay begins.
Journal opens up the Journal.
Settings opens up the volume settings.
Quit closes the game.

Journal

The journal can be accessed from the main menu. It contains the entries for all towers and enemies in the game. When an entry is opened, a window of the corresponding tower or enemy's information will be displayed. The information includes the name, description, stats and the sprite.



Pew Pew



Pew Pew is the first prototype of the engineers' ranged towers.

It shoots fast and accurate rockets that deal moderate damage to enemies.

Health: 1000
Ranged DMG: 15
ASPD: 3

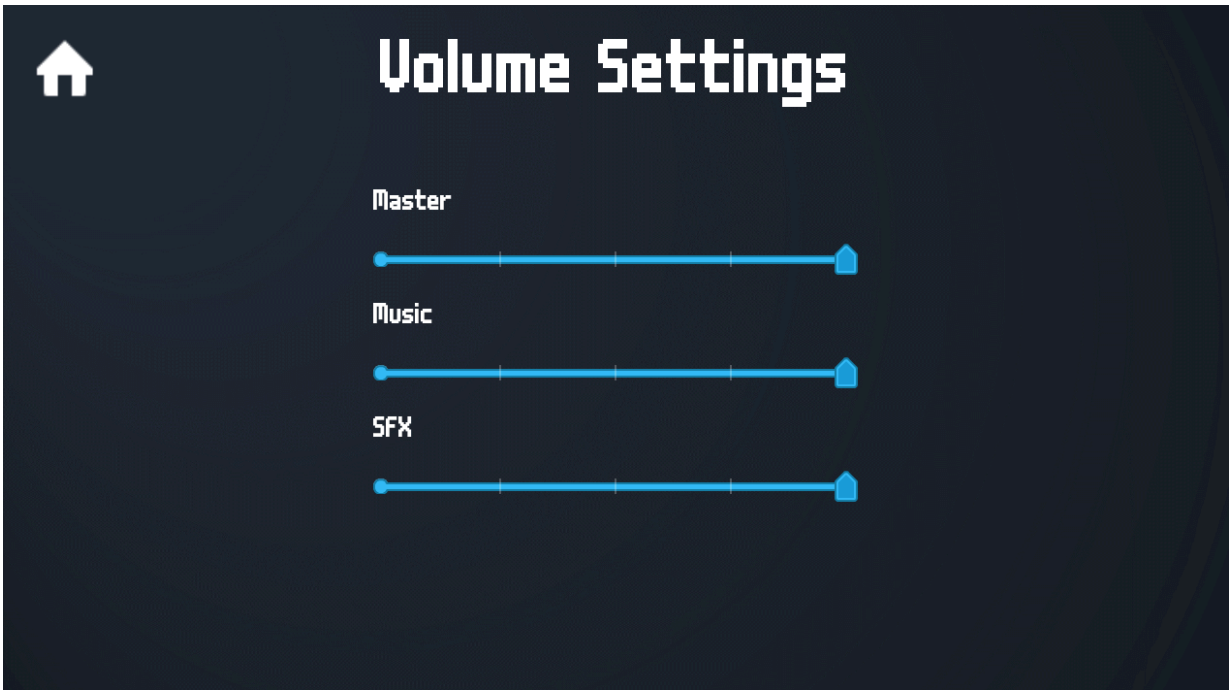
Samurai

He is the first enemy you encounter in the game. Using his sword, he deals melee damage to foes in his way.

Health: 500
Speed: 100
Melee DMG : 100

Volume Settings

The volume settings consist of the Master, music and SFX volume.
Master volume controls the overall game volume.
Music controls the music soundtracks in the main menu and in gameplay.
SFX controls the button sounds and tower / enemy sounds. Tower / enemy SFX sounds have not been implemented.

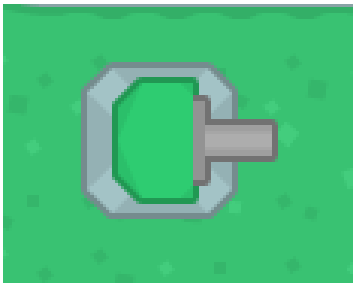
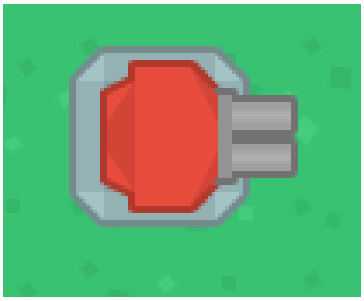
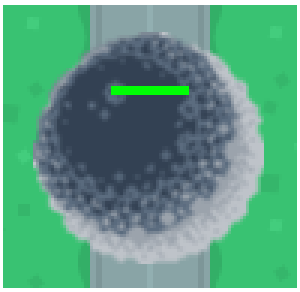


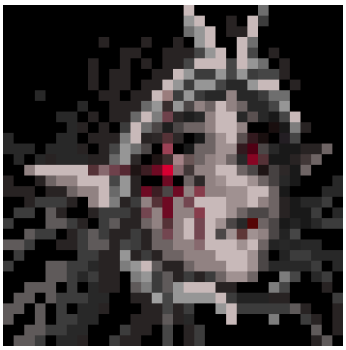
Map Select

We added Map Select in conjunction with the new maps added. This allows for a more dynamic gameplay experience by letting players choose their preferred map. The Map Selection system is integrated into the UI, enabling seamless transitions between maps and encouraging replayability as players adapt their strategies to different environments.

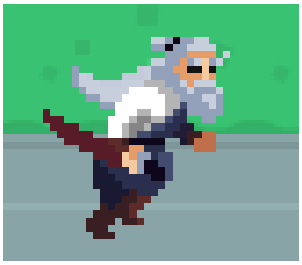
Map Select uses **A** and **D** to switch maps. We also added **left click** for fast select, which instantly brings the Chicken Icon to the clicked map. Clicking again will confirm the chosen map and enter the GameScene.

Towers

 Pew Pew	Pew Pew is the first ranged tower in the game. It shoots projectiles at the enemies, constantly targeting the enemy closest to the base. HP: 800 Ranged attack damage: 35 Attack speed: 3
 Rocketeer	Rocketeer has the same stats as Pew Pew, but deals Area of Effect (AoE) damage to enemies HP: 800 Ranged attack damage: 80 Attack speed: 0.4
 The Boulder	The Boulder is the first melee tower that blocks enemies. It serves as a deployable obstacle or a meatshield to hinder enemy movement. It is also unable to attack enemies. Respect the Boulder. HP: 2400 Block count: 3
 Inverse Boulder	The Inverse Boulder is a copy of The Boulder that offers melee attacks. HP: 1800 Block count: 2 Melee attack damage: 60

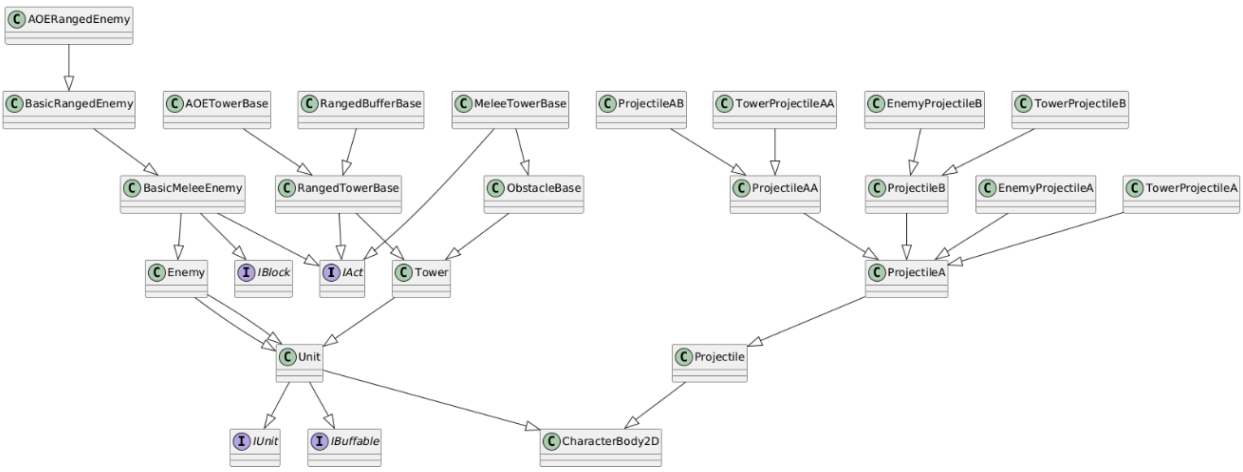
 Astraea	Astraea is a ranged tower that deals single target attacks with slow debuffs. HP: 1600 Ranged attack damage: 60 Attack speed: 0.8
 Nyx	Nyx is a ranged tower that deals single target attacks with bleed Damage Over Time effects (DoT). She also has a skill that deals AoE damage as well as heals herself. (Skill is activated by pressing the skill icon when available) HP: 600 Ranged attack damage: 120 Attack speed: 0.2

Enemies

 Samurai	The Samurai is the first enemy encounter in the game. It is a melee enemy that only attacks towers in the path. HP: 400 Movement Speed: 100 Melee Attack DMG: 100
 Rocket Samurai	The Rocket Samurai is a ranged variant of the Samurai with both melee and ranged modes. When blocked by a melee tower, it will initiate melee mode, attacking the tower until it is dead. When not blocked, it will launch projectiles at the nearest tower. Rocket Samurai deals twice more damage in ranged mode than in melee mode. HP: 300 Movement Speed: 100 Melee Attack DMG: 30 Ranged Attack DMG: 60
 Mighty Whitey	Mighty Whitey is a buffed variant of the Samurai, with double its melee attack damage and movement speed, but half its HP. HP: 300 Movement Speed: 200 Melee Attack DMG: 120
	DuskBorne is the first map boss in the game. It has high HP and high attack damage. HP: 2000 Movement Speed: 75

DuskBorne	Melee Attack DMG: 500
 Voidmire	Voidmire is the second boss in the game. When in ranged mode, it launches projectiles that deal high damage. HP: 4000 Movement Speed: 75 Melee DMG: 200 Ranged DMG: 400

Unit and Projectile Classes

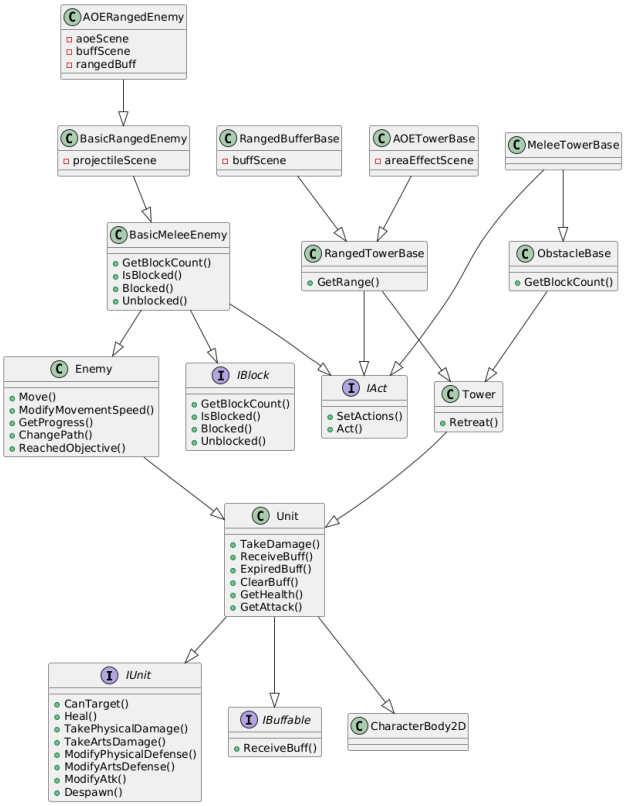


Unit

Initially, we used GDScript to design the towers and enemies as isolated scenes, without using any inheritance or interfaces. Each tower and enemy was designed from scratch with its own attacks, movement, damage handling and animation. While this worked for our early prototype with 4 towers and 3 enemies, we realised that this approach was not scalable.

GDScript was lightweight and tightly integrated with the Godot 4 engine, designed specifically for Godot, making certain operations easier and more intuitive. However, it offered limited support for advanced OOP features such as interfaces and abstract classes. C# on the other hand had full support for OOP, better performance and runtime for more complex logic, and we also wanted to take this opportunity to learn C#. Therefore, we made the switch to C#.

We had adopted this approach due to the time constraints in Milestone 1 to push out a working solution, but we were also making plans to refactor the codebase with C# in Milestone 2 in order to utilise inheritance and interfaces to increase reusability, extensibility and clearer separation of concerns.

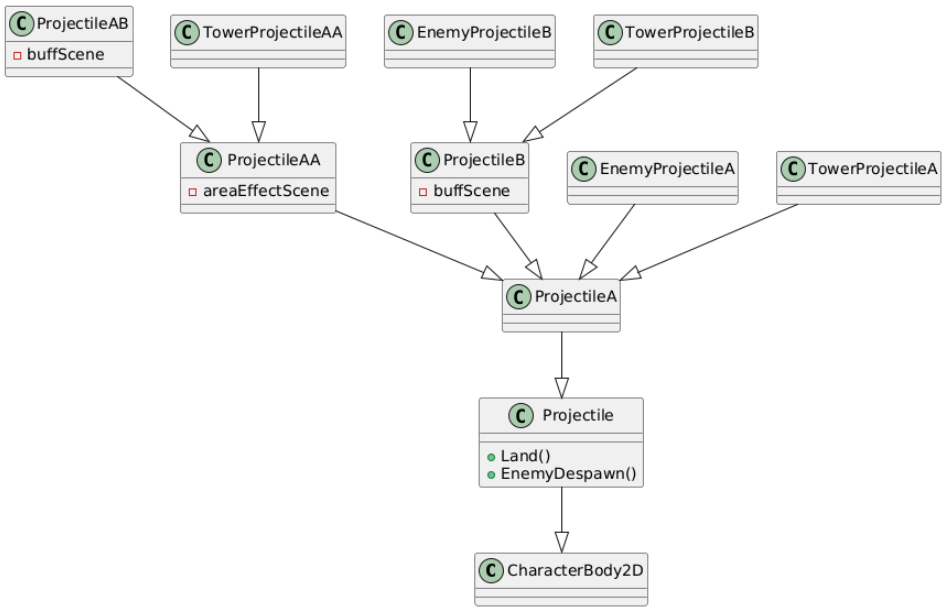


By the midpoint of Milestone 2, we introduced two main hierarchies, towers and enemies, which both stemmed from the class Unit. This allows shared logic such as attacking or getting hit to remain centralised, while allowing for specific behaviours in unique subclasses. We broke down Towers into many different TowerBase types such as MeleeTowerBase, RangedTowerBase, RangedBufferBase to allow for specialised behaviours for each tower variant. Similarly, enemies also inherit from BasicMeleeEnemy, BasicRangedEnemy and AOERangedEnemy.

By implementing this object-oriented design, it helped us better organise the logic for our towers and enemies, as well as laying a strong foundation for expanding our arsenal in future updates or milestones.

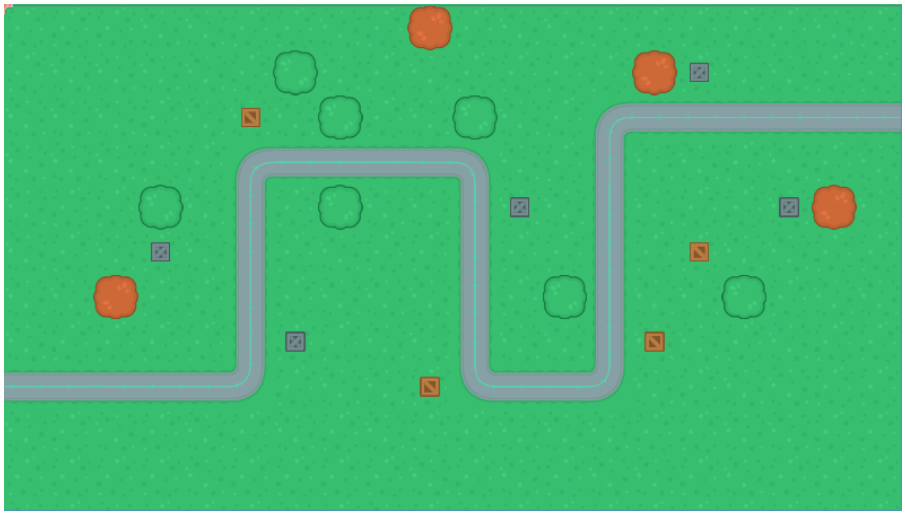
Projectiles

Same rationale as for Unit. We utilised an inheritance driven structure for the projectiles.


ProjectileAB
- buffScene
TowerProjectileAA
EnemyProjectileB
TowerProjectileB
ProjectileAA
- areaEffectScene
ProjectileB
- buffScene
EnemyProjectileA
TowerProjectileA
ProjectileA
Projectile
+ Land()
+ EnemyDespawn()
CharacterBody2D

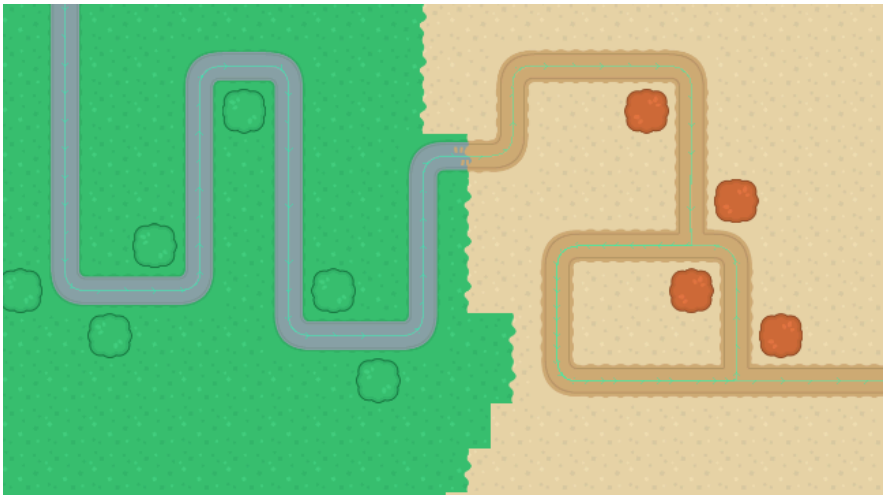
Common Maps

Map 1 (Highway to Spice)



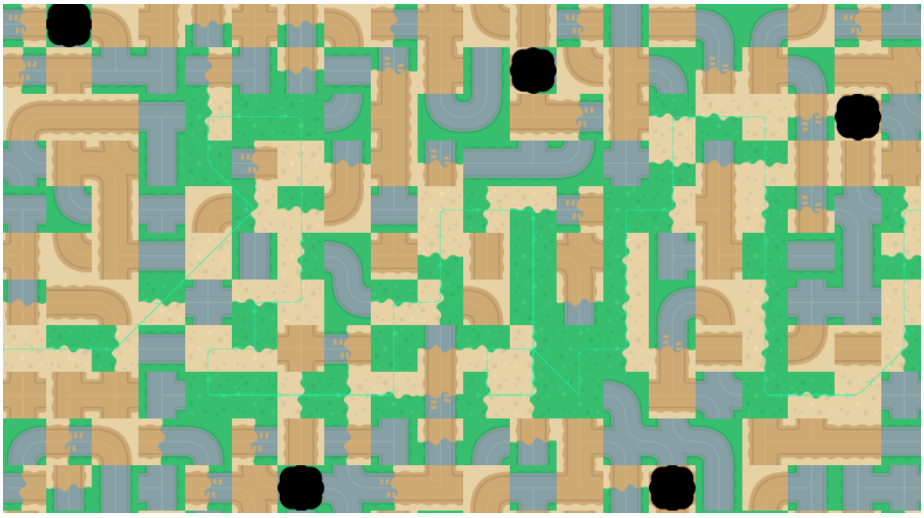
The first map offers a basic linear path that gives players time to experiment with the unique features of each tower and defend against the waves of enemies. It also introduces Duskborne, a fast moving melee enemy that deals very big damage to low ground towers.

Map 2 (Another One Bites the Don)



Map 2 introduces a new boss enemy Voidmire, who specialises in ranged projectiles. His projectiles deal very high damage to towers. However, in melee mode, his damage output is low. The player may gain an advantage from using low ground towers such as Boulder and Inverse Boulder to switch it to melee mode, taking away its main source of damage.

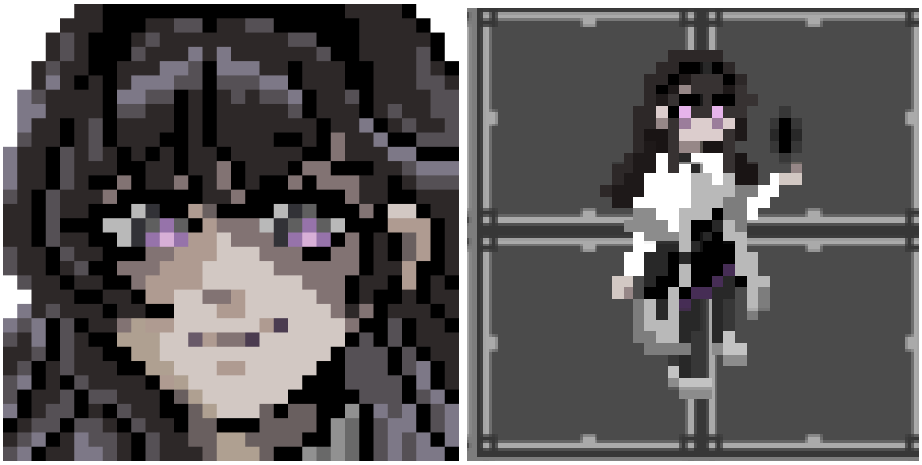
Map 3 (404 Terrain Not Found)



Map 3 is more of a joke map with high and low ground textures swapped to disorient the player. Boss enemies from previous maps make an appearance here as well, adding extra difficulty.

Boss Map (Priestess)

This stage features a duo boss with 3 phases, Priestess / PRTS. In this stage, the goal is to beat Priestess while she attempts to use the PRTS and her minions to drain your objective health. This stage features a unique UI that displays the information of each boss at the top, as well as a difficulty select (we strongly recommend beginner).



Priestess

PRTS

Difficulty Select

Since the original stage was designed with difficulty in mind, we implemented a beginner mode for this stage (that is consistently beatable) which tones down the boss stats, moves and enemy waves, effectively shortening the fight duration.

Stage Layout

To easily demarcate melee and ranged tiles, we represent them with black and white tiles respectively. Do note that the top and bottom two rows (covered by the UI), as well as any tiles in the boss path, are restricted for tower deployment.

Stage Mechanics

The fight starts in Phase 1, where Priestess will try to attack the player with her skills and minions as she teleports around fixed positions on the map. Upon reaching her destination, the center stage, she will trigger Phase 2 by clearing all non-boss entities and switching the map layout.

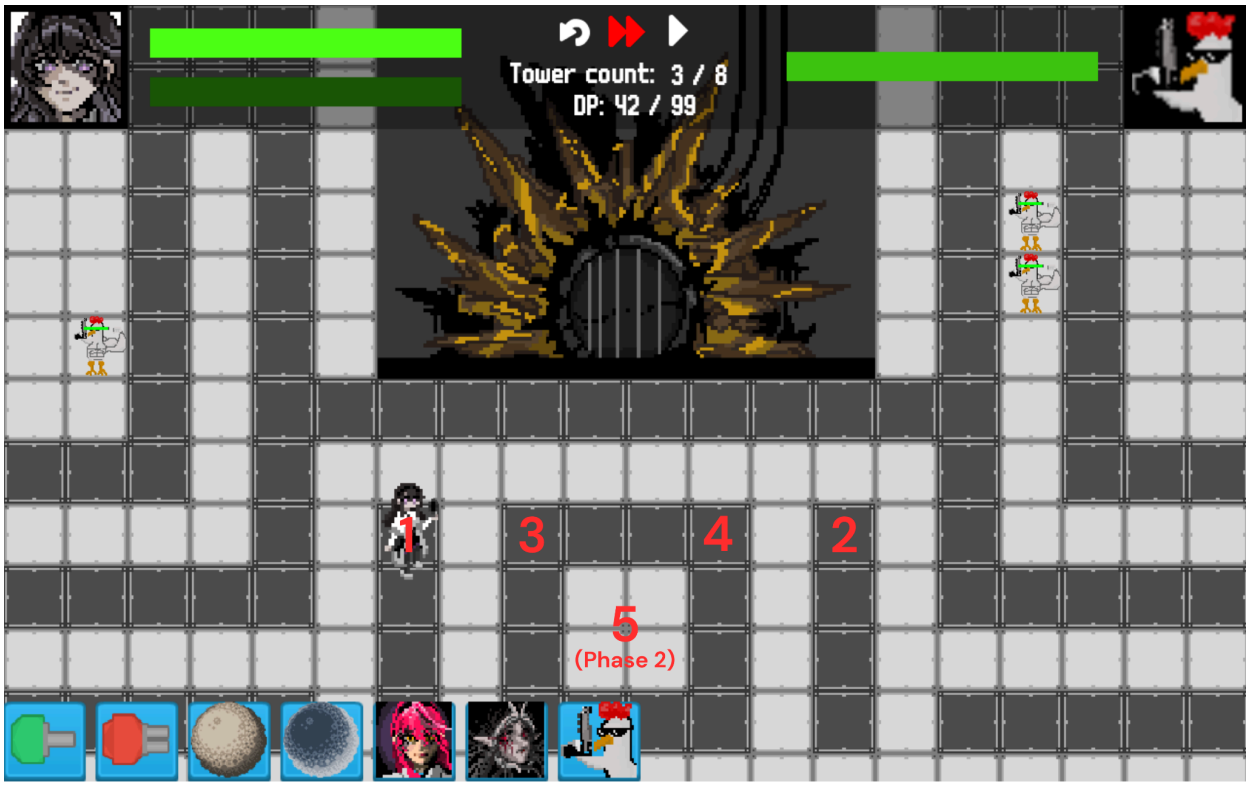
In Phase 2, the acting boss will switch between Priestess and PRTS as the former attempts to connect to the latter. While the PRTS is active, the player will suffer from corrosion damage, which slowly drains their objective health. The goal is to defeat PRTS as quickly as possible, which would force the fight back to Priestess and put her on a cooldown before she may attempt to reconnect. Since there is no limit to how many times the PRTS can be activated, the only way to progress to Phase 3 is by bringing Priestess' Hp down to 0.

In Phase 3, Priestess becomes untargetable and invulnerable as she constantly barrages the map whilst sending even stronger minions towards the ally base. If the player is able to survive this desperation phase, they will emerge victorious.

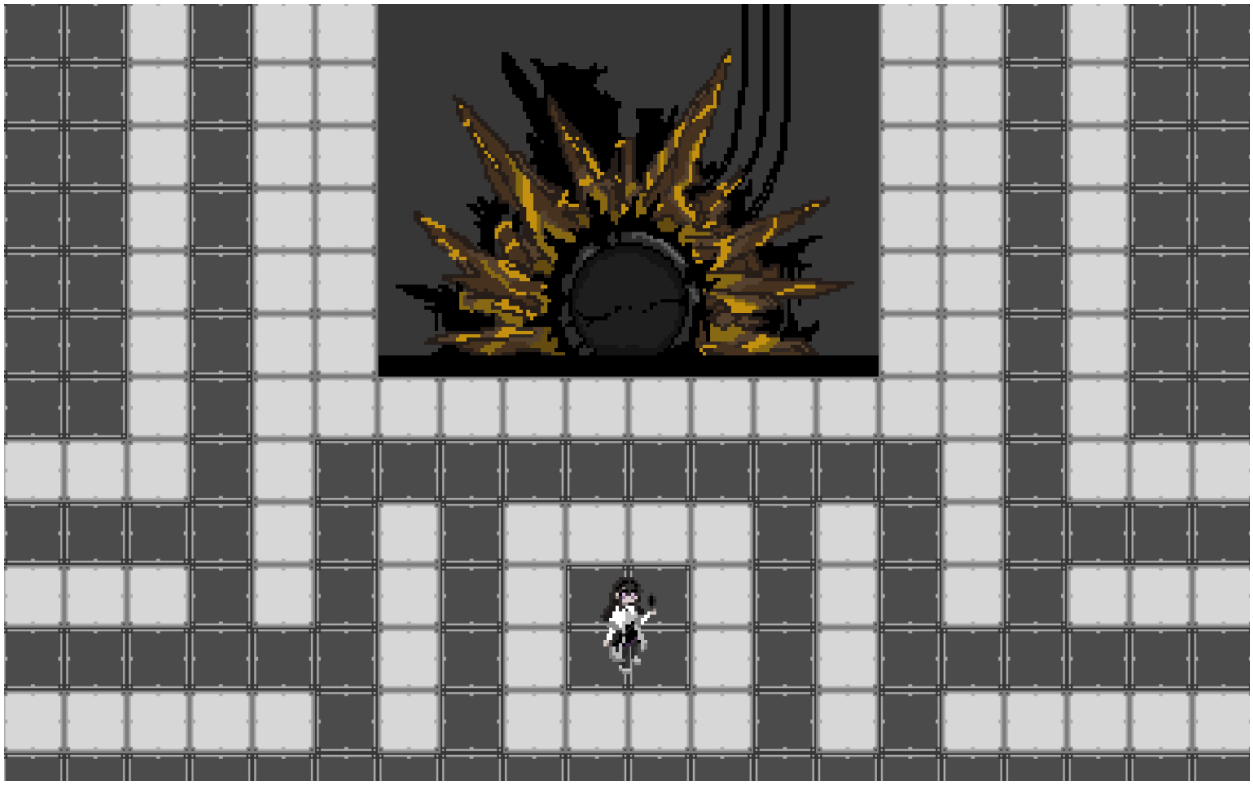
Detailed Fight Information (Spoilers)

To differentiate difficulty: Stats on Beginner (Stats on Hard)
For stat changes during gameplay: Stat1/Stat2 during Condition1/Condition2

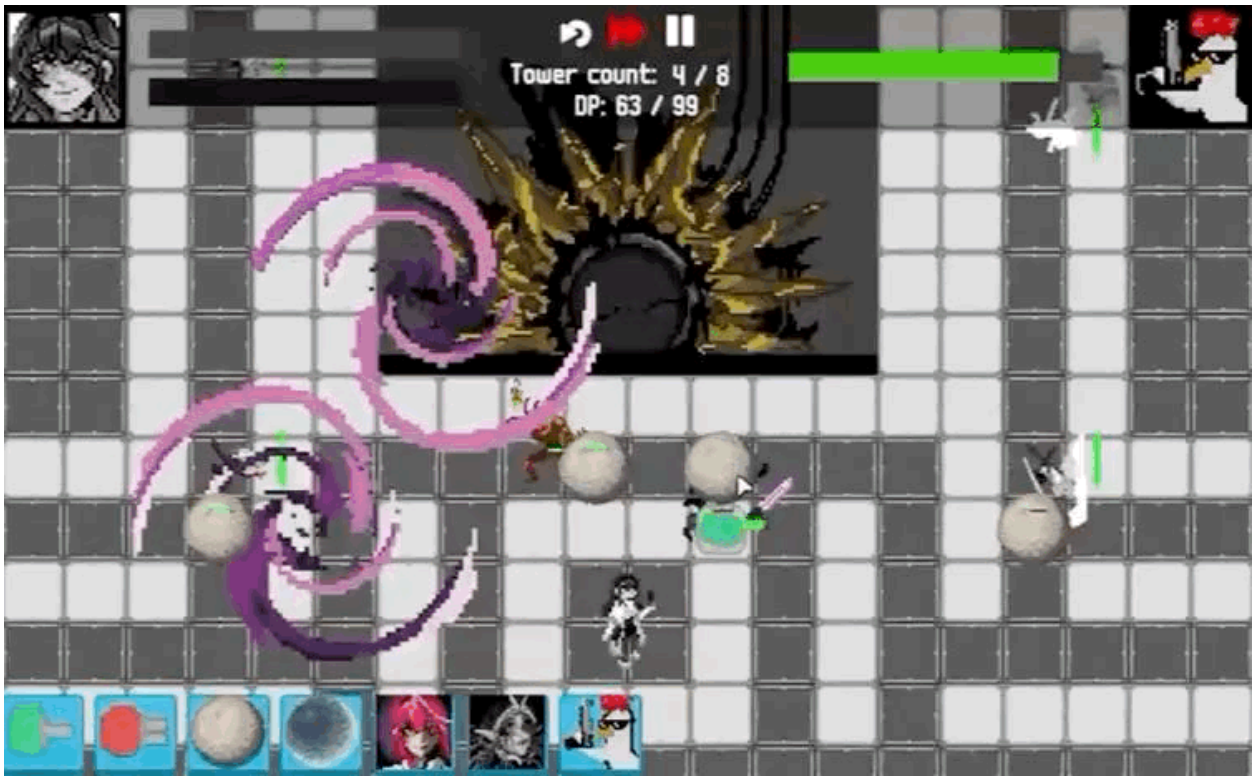
The base has a maximum health of 4200 (3600), and each regular enemy / miniboss leaked deals 200 / 1000 damage to the base.



Phase 1 lasts 80 (120) seconds. Priestess will teleport between the 5 positions in the above order. If the player somehow manages to bring Priestess down to 50% HP in Phase 1, she will immediately teleport to the center stage and trigger the next phase.



Above is the layout for Phase 2 onwards of the fight. In this phase, when one boss is active, the other will be inactive, invulnerable and untargetable. PRTS will drain 1 point of objective health for every 0.1 seconds it is active on field, effectively giving the player 7 (6) minutes in PRTS mode if they do not leak any enemies. Once kicked out of PRTS mode, it takes Priestess 60 (90) seconds before she may reactivate it.



Phase 3 lasts 45 (60) seconds, during which Priestess will only use the skill “Fates Finale”, and send an extra 2 (4) minibosses towards the ally base. All the player needs to do is survive this phase to win.

Boss Abilities

Both Priestess and PRTS each have 5 unique skills, ranging from enemy buffs to damage dealing skills, and even ones that mess with your screen. Each skill has a node attached to the boss scene, which handles trigger conditions, along with a paired method in the boss that details skill execution. C# events are extensively used to enable communication between bosses, their skills, and the stage manager.

Each boss has its own ability cooldown duration. When at 0, the boss will check for valid targets, then filter out all skills that have their trigger conditions met. The boss will then pick the available skill with the highest skill priority to cast.

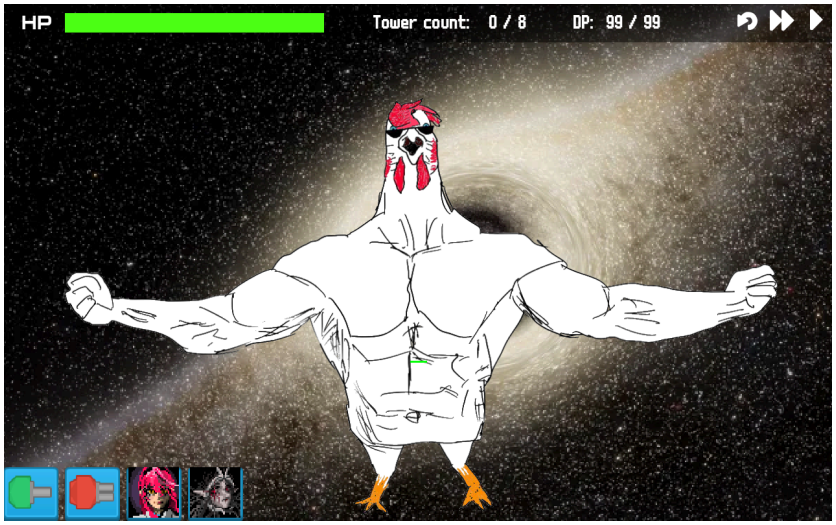
Below are detailed information on each Boss' skills:
Priestess (Ability Cooldown: 2.5 ~ 3 seconds)

Skill	Description	Trigger Conditions
Inferno (Priority: 0)	Multi target ranged arts attack inflicting ArtsDef -40% debuff lasting 10 seconds. Prioritises towers with highest Atk.	Cooldown: None 2/3 targets when below 100/50 % Hp 1.0/1.1/1.2/1.2 (1.0/1.1/1.2/1.3) % Atk Dmg when below 100/75/50/25 % Hp
Cocytus (Priority: 1)	Multi target Heal on enemies that provides a 10 second 40% Atk buff. Prioritises enemies with lowest Hp.	Cooldown: 6s 2/3 targets when below 100/75 % Hp 1.0 % Heal
Starfire	Inverts the in-game UI for 15 seconds. The trick is to always believe in the mouse location when choosing a tile.	Twice at 75/25 % Hp
Erebus (Priority: 2)	Multi target arts area attack that prioritises towers furthest from boss.	Cooldown: 15s 2/2/3 (2/3/4) targets when below 100/50/25 % Hp 0.8/0.9/1.0 % Atk Dmg when below 100/75/25 % Hp
Hyades	Removes tower deployment bar for 15 seconds, preventing deployment.	Once at 50 % Hp
FatesFinale	For a period of time, gains invincibility and continuously rains down arts area attacks on random locations.	When at 0 % Hp 0.5 (0.7) % Atk Dmg every 0.25s for 45 (60) s

PRTS (Ability Cooldown: 4 ~ 5 seconds)

Skill	Description	Trigger Conditions
Achlys (Priority: 0)	Prioritising ranged towers, inflicts a tower with a -100 % Atk debuff for 8 seconds. When inflicted, the host will fire a physical projectile at the nearest ally tower whenever it attacks.	Cooldown: None Projectile does 1.5 % of host's Atk Dmg
Astrape	Deploys a shield. When shield is active, continuously fires physical projectiles at random towers that inflict PhysDef -25% debuff lasting 5 seconds. When shield is broken, it will be stunned.	Once at 50 % Hp 4000 (5000) Shield Hp 0.4 % Atk Dmg every 0.6 (0.4) s Stunned for 10 (5) s
Charybdis (Priority: 2)	Inflicts a tower with a bomb debuff that lasts 1.5 seconds. If the tower does not manage to land an attack on the boss before the bomb expires, it will explode, dealing arts area damage before spreading to the next tower that has not yet been afflicted. Prioritises towers furthest from the boss.	Cooldown: 20s 1.2 % Atk Dmg
Helios (Priority: 3)	Removes all enemies and towers in a line. Targets a random tower with a random direction (vertical / horizontal).	Cooldown: 28s
Pharos (Priority: 1)	Randomly chooses multiple towers and creates arts damage over time tiles in a random pattern (4 types) around them.	Cooldown: 16s 2/2 (2/3) targets when below 100/50 % Hp Tiles deal 0.6 % Atk Dmg every 2s for 6s

Boss Map (Chicken Don)



Phase 1

Chicken Don has very slow movement speed in Phase 1. Chicken Don asserts its dominance and does not attack during this phase. Players must bring it down to 50% HP to trigger Phase 2.

Phase 2: Eyes Over Heaven

On phase change, Chicken Don's movement speed starts increasing exponentially via the formula: $movementSpeed = originalSpeed \times 1.05^{lifetime}$

If the player does not deal damage fast enough, Chicken Don will move too fast and enter the base, leading to an instant defeat. The player may benefit from using towers with movement slow debuffs such as Astraea. Once Chicken Don's HP reaches 25%, its movement speed will stop increasing and Phase 3 will begin.

Phase 3: Total Global Oblivion

Chicken Don will start using black hole attacks that instantly kill towers. His attack priority is the tower closest to him, so the player must ensure that any ranged towers responsible for debuffing his speed must be placed furthest from him at any point in time. Triggering Phase 1 and 2 as fast as possible is vital to beating the Chicken Don, as it may be difficult to defeat him in time by Phase 3.

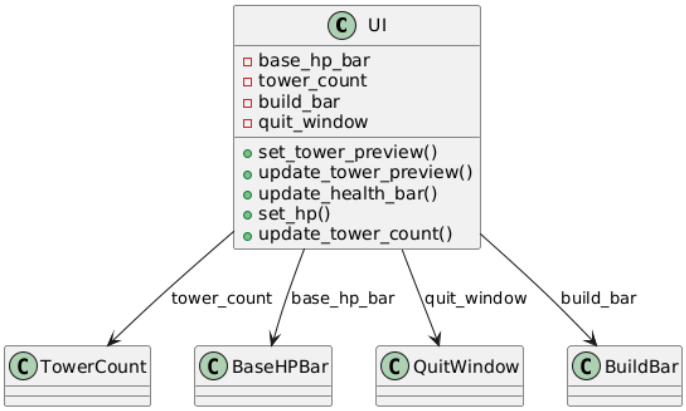
Rewards

Upon defeating Chicken Don, the player will unlock Chicken Don as a ranged tower in all maps. The Chicken Don tower's stats are magnified and served as a debugging tower during our game development. Although highly discouraged, we let players have access to Chicken Don just in case they want to breeze through certain parts of the game.

In-Game UI

UI

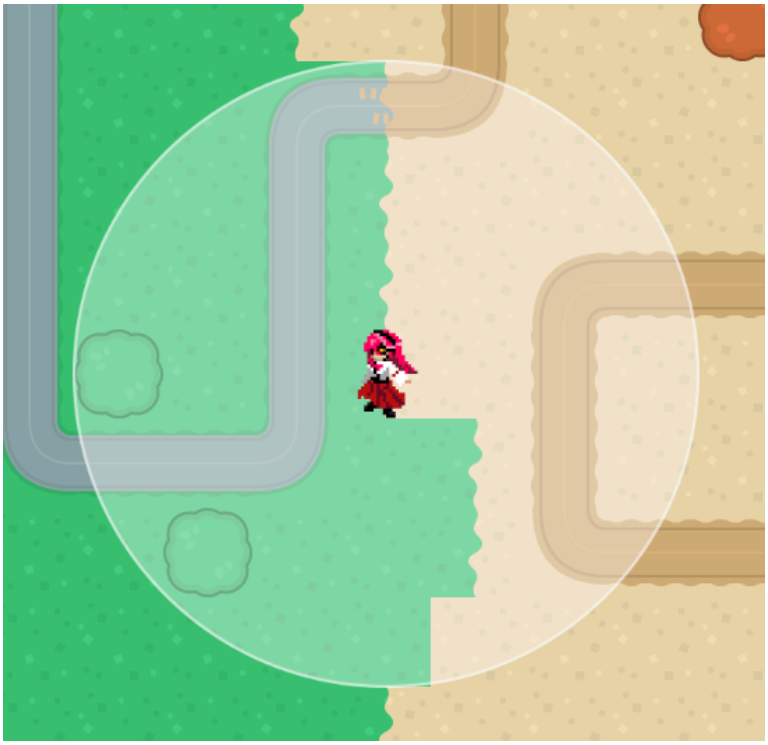
UI works hand-in-hand with GameScene, controlling the majority of the UI aspects in the game. It acts as the bridge between game logic and visuals, updating visual elements to reflect in-game events such as placing towers, changing base health, and valid tower placements. This class follows the Single Responsibility Principle, ensuring that all UI concerns are handled in one dedicated place. This allows for a more centralised UI logic, as well as more extensibility for further additions such as BossUI that extends the UI class.



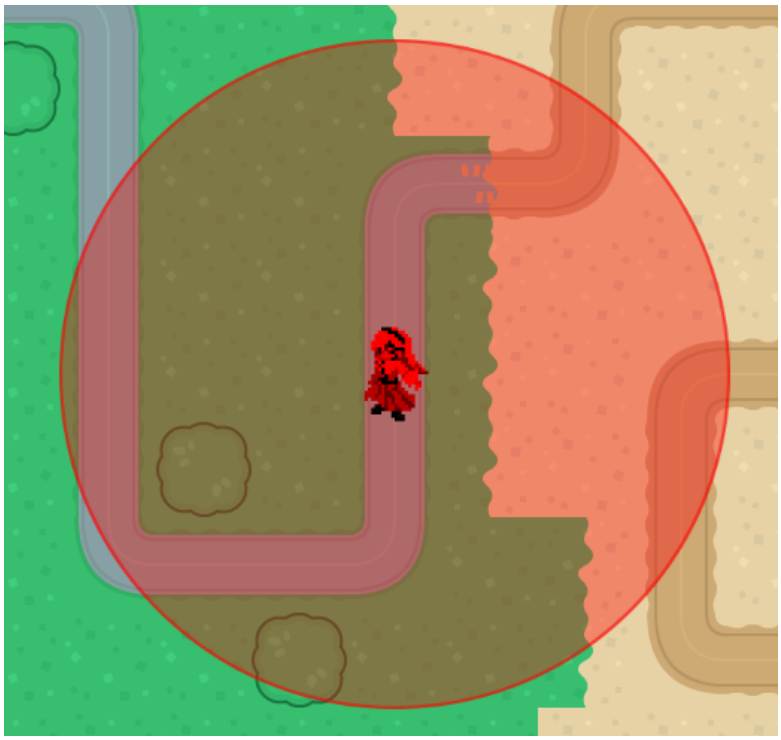
Settings

The In-Game UI consists of Pause / Play, Fast Forward and Restart.
The Pause / Play button gives players full control over the game's flow by allowing them to pause and resume gameplay at any time.
The Fast Forward button toggles between 1x and 2x speed to fit different players' needs. We added this feature as we felt that this was a much needed QoL feature in tower defense games, that addresses a common pain point in tower defense games: extended downtime between waves or during slower combat segments.
The Restart button prompts the player if he wishes to quit the game.

Build Mode



In Build Mode, the tower selected will be shown in a Tower Preview, where said tower will follow your mouse to where you want to deploy it on the map. If the build location is valid, the tower will be deployed successfully.



If the build location is invalid, the tower and its range indicator will flash red. If the player attempts to build on an invalid location, the Build Mode will be cancelled. Invalid locations include specific tiles and existing towers on the map.

Base HP

The total remaining base HP is indicated on the top left hand corner of the screen. At full HP, the HP bar is green. With more HP lost, the colour of the bar changes to yellow, followed by

red. The base HP reduces by 1 with each enemy entering the base. Upon the base HP reaching zero, the game will end and the player will be redirected to the Main Menu.

Skills

Certain towers such as Ranged Tower Girl 2 have manual skills that can be activated in battle. Upon clicking the skill button, the skill will activate and the button will disappear.

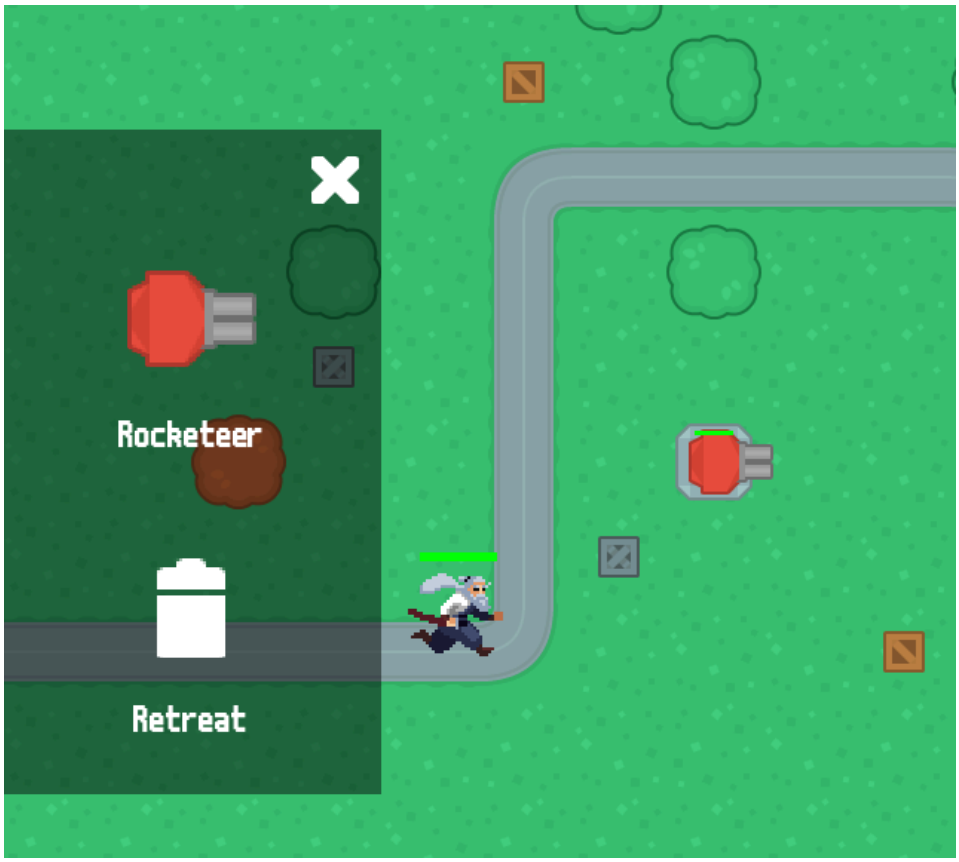
Pre-skill available

Skill Available

Hovering over skill button

Tower UI

In order to balance the fixed tower count, we added a tower UI which offers retreat, relocate and skill buttons. The retreat button allows players to remove a tower from the map, freeing up the tower count, allowing players to deploy another tower of their choice. When tower UI is visible, the game will be in slow motion (more details in the **Balancing the Game** section).



End Game Screen

In Milestones 1 and 2, our game would end abruptly upon winning or losing. In Milestone 3, we added Victory and Defeat screens after each game to create a smoother and more satisfying conclusion to the gameplay experience. The player can then click anywhere on the screen to return back to the main menu.

Victory screen

Defeat screen

Balancing the game

In the early stages of our game development, towers were able to be deployed without any limit, making the map stages trivial to beat. After conducting user testing and getting feedback regarding this, we added a tower limit and a DP system to counter this issue, ultimately making our game more challenging.

Tower Limit

Using a tower manager class, we were able to introduce a tower limit to cap the number of deployable towers on the map at once. This would prevent players from needlessly spamming units for an easy stage clear.

The tower limit varies for different maps, depending on the difficulty. Typically, higher tower count is set for more difficult maps such as Boss Map (Priestess) and Boss Map (Chicken Don).

Deployment Points (DP) System

While the tower limit did reduce how much firepower a player could use at once, we found that it was sometimes not sufficient as a balancing measure due to the strength of certain towers. As such, we added an additional DP system.

The DP system consists of a DP generator and DP costs for each tower. Players must wait for the generator to generate enough DP before they are able to deploy a tower. There is a hard cap of 99 DP where DP generation will stop until DP is used. Each tower is set to cost 5 DP.

Upon retreating, the DP system will refund you 3 DP. This will reward players in two cases:

1. The tower is not in range of the enemy, thus retreating and relocating with a lower DP loss.

2. The tower is almost dead, and retreating it would recoup some of the DP cost instead of letting it die and getting 0 DP back. This mechanic rewards players for being vigilant of their tower HP.

Slow Motion

After receiving feedback regarding the lack of time for strategy and decision making during gameplay, we decided to introduce slowmo in certain aspects of the in-game UI. Slowmo sets the engine timescale to 0.3x speed.

Introducing slowmo would give players more time to strategise and think without having to worry about enemies speeding into the base in the meantime.

When a player has entered Build Mode, the game will be set to slow mo. Since tower placement can significantly impact the outcome of the game, slowing down time during this phase allows players to think and act more strategically. This improves the overall experience by reducing pressure on the player and making critical decisions feel more deliberate and rewarding. Slow motion ends when build mode is exited.

Additionally, when a player clicks on a tower to open the Tower UI, the game also enters slow motion. This gives players the time and clarity needed to activate skills or retreat towers without feeling rushed. Slow motion ends when tower UI is exited.

Testing

Integration Testing

We adopted functionality testing to ensure our game functions as intended. It involves testing the game's core features, mechanics, and gameplay to identify and fix any bugs or issues that could hinder the player's experience. This also verifies that the game adheres to all technical and functional requirements outlined in design documents and specifications.

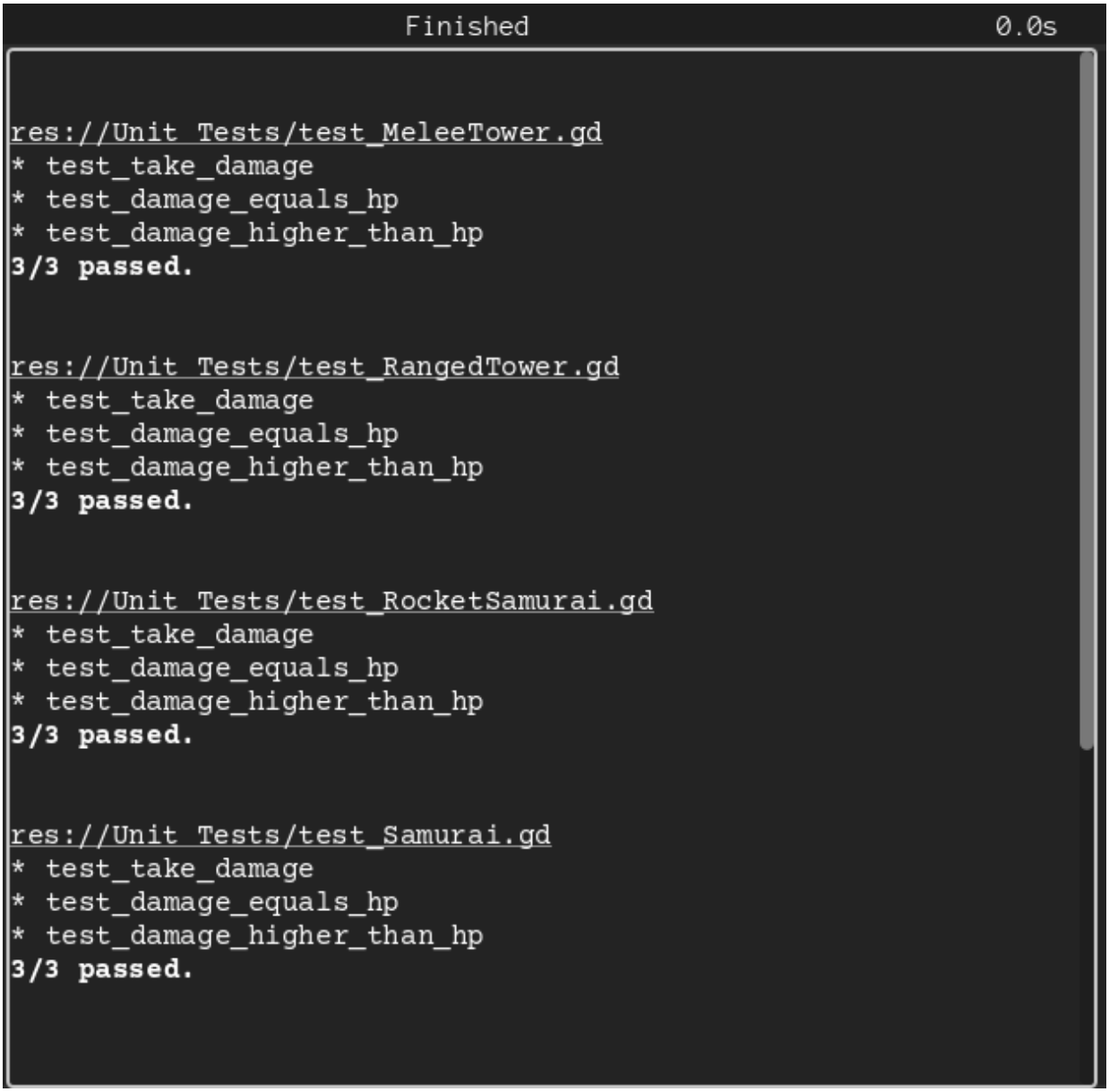
No.	Test Case	Results
1	Click Quit on Main Menu	Game window closes
2	Click New Game on Main Menu	Map Select loads up
3	Click Pause button during gameplay	Gameplay is paused
4	Click Play button during gameplay	Gameplay is resumed
5	Click Fast Forward button	Gameplay speeds up to 2x speed
6	Click Restart button during gameplay	Game is brought back to main menu
7	Click Tower build button	Tower Preview mode entered, selected tower and its range indicator follows the mouse to any point on the map
8	Hover mouse over a valid map tile while in Tower Preview mode	Selected tower and range indicator in tower preview remain unchanged
9	Hover mouse over an invalid map tile while in Tower Preview mode	Selected tower and range indicator in tower preview turn red
10	Hover mouse over an existing Tower while in Tower Preview mode	Selected tower and range indicator in tower preview turn red

11	Click a valid map tile while in Tower Preview mode	Tower is successfully deployed onto the selected map tile
12	Click an invalid map tile while in Tower Preview mode	Build Mode is cancelled
13	Click a map tile with an existing Tower while in Tower Preview mode	Build Mode is cancelled
14	Samurai enters attack range of Ranged Tower	Ranged Tower attacks Samurai with projectiles Visual Health Bar of Samurai decreases
15	Samurai leaves the attack range of Ranged Tower	Ranged Tower stops attacking Samurai with projectiles
16	Samurai gets killed by Ranged Tower	Samurai despawns from the map
17	Ranged Tower enters the attack range of Rocket Samurai	Rocket Samurai attacks Ranged Tower with projectiles Visual Health Bar of Ranged Tower decreases
18	Ranged Tower is no longer in attack range of Rocket Samurai	Rocket Samurai stops attacking Ranged Tower with projectiles
19	Ranged Tower gets killed by Rocket Samurai	Ranged Tower despawns from the map
20	Enter Build Mode and click on a tile with a previously despawned Tower	Tower is successfully deployed onto the selected map tile
21	Melee Tower placed in front of enemy path	Enemies are blocked by the Melee Tower. Rocket Samurai changes from ranged to melee mode Samurai performs melee attacks
22	Melee Tower is destroyed	Enemies continue moving along the path. Rocket Samurai returns to ranged mode.
23	Enemy enters base	Base HP reduces by 1

24	Base HP reduces to zero	Game is redirected back to Main Menu

Automated Unit Testing

We used Godot Unit Testing (GUT) from the Godot assets library and ran a couple of test cases for Ranged and Melee Tower and Enemy functions. This helped to ensure that each unit of code in the game worked as intended and met the requirements, improving the overall quality of the game. It also allowed us to detect and fix issues early before they snowballed into bigger problems.



```
================================================
= Run Summary
================================================

---- Totals ----
Scripts            4
Tests              12
  Passing          12
Asserts            12
Time               0.016s


---- All tests passed! ----
```

User Testing

In order to gain valuable feedback to improve our game, we conducted user testing with 3 users, asking them to be as brutal with their feedback as possible. These 3 users tested our game and provided feedback on the various aspects such as gameplay mechanics, user interface, difficulty balance, and overall enjoyment. This feedback has been vital in pinpointing areas for improvement and prioritising changes for development in MS3. The main points of criticism with their respective game improvements are shown in the table below.

No.	Criticism / Issues	Evaluation	Improvements
1	The enemies should come in waves usually in these types of games and rather than a strict tower limit maybe some kind of currency because the tower limit becomes an issue when there are different types of enemies which need to be countered in different ways.	A rigid tower cap limits strategy and replayability. Adding a currency system would allow better decision making.	We added a tower limit and DP system to balance the towers. More information can be found in the **“Balancing the game”** section above.

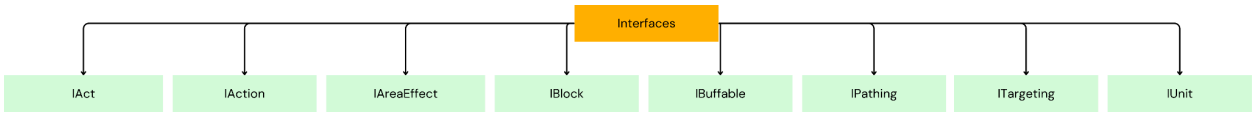
2	Restart disrupts immersion by jumping to the main menu, and the fast-forward button doesn't clearly indicate the current speed setting.	The in-game UI experience is disjointed and needs more visual cues.	We added a pop up menu to allow the player to confirm their quitting of the game. We also added an extra fast-forward texture to indicate if the game is in 1x or 2x speed. More information can be found in the **"In-Game UI"** section
3	Map selection isn't very intuitive. It's unclear which button confirms the choice, and input is limited to A/D keys, with no mouse support.	Map selection creates a clunky experience, especially for players that expect mouse interaction.	We added mouse inputs as well as quick select, allowing for a smoother selection of maps.
4	The Priestess boss map's mechanics may be too complex or difficult for beginners to beat.	Sudden difficulty spikes from Maps 1-3 to Boss Map could scare away beginners.	We added an easy difficulty mode specifically for the boss map (priestess).
5	The main menu has a plate of chicken for no good reason.	You do not appreciate the Chicken Don.	We added a Chicken Don boss fight in response to this.

Switching Over to C#

One of the main goals set in milestone 2 was the restructuring of enemy / tower logic using C# and object oriented principles, with the aim of achieving greater modularity by building layers of abstraction. Currently, all enemy-tower interaction logic has been fully implemented in C#, with some newer features built on top.

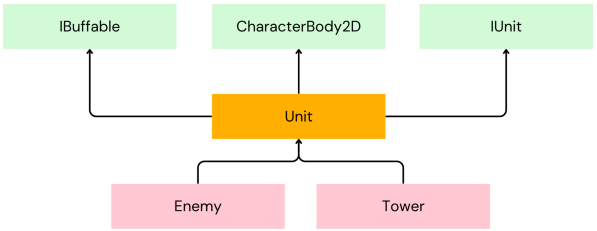
Using Interfaces

Interfaces are a powerful tool offered by C# which are used to guarantee implementation of certain methods. We always start by working with interfaces, especially when we are not fully certain how things should behave, or how such behaviour should be implemented.



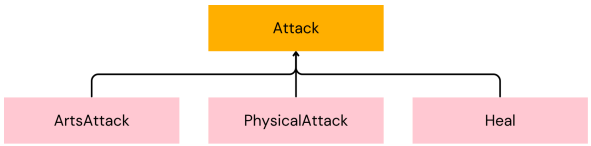
A Base Unit

Some of the first few interfaces are IDespawn and IGetHit, which marks an entity as “attackable”. We later recognised these as core behaviours for all towers and enemies, and merged them into a single IUnit interface. After introducing buffs, we created a single Unit class to hold all the defensive logic alongside the physics logic from Godot.

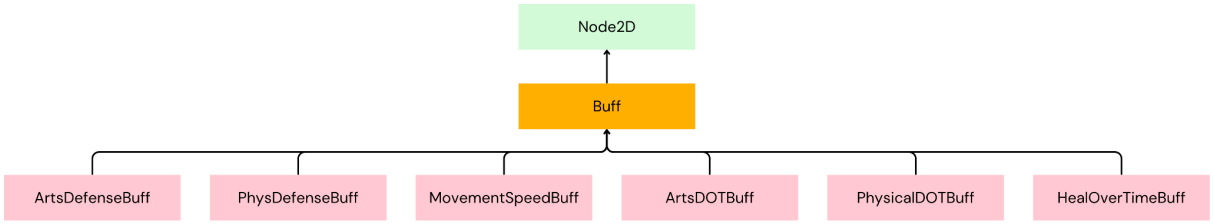


Abstracting Entity Interactions

The lowest level of abstraction for a tower defense gameplay would be direct interactions between entities. These include attacks and heals, which are defined in their own C# classes, within a hit method. Here, polymorphism is used to assign different types of attacks to their corresponding hit response method in the target.

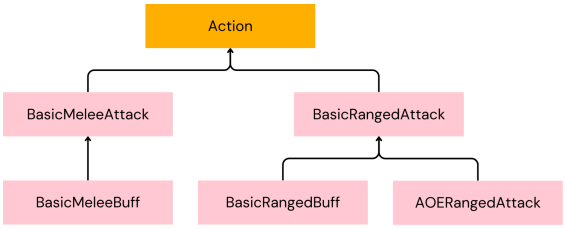


Buffs and debuffs are more complicated since they interact with their target over a duration. We settled on making them Godot scene classes, so we could also attach animations to buff scenes. Any buff may be accepted by an IBuffable, but not all types of buffs may have effects on a specific unit type. Buffs of the same type (by id) do not stack.

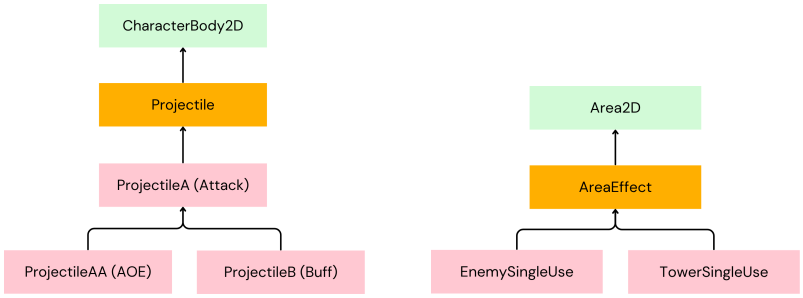


Building on Abstractions

Having just the attack and buff subclasses alone is not enough, as certain actions, like firing a projectile, may require more setup and stored data. The Action subclasses serve as wrappers around basic interactions, encapsulating the complete offensive behaviour performed by a unit. Inheritance is used to build complexity in the action performed.

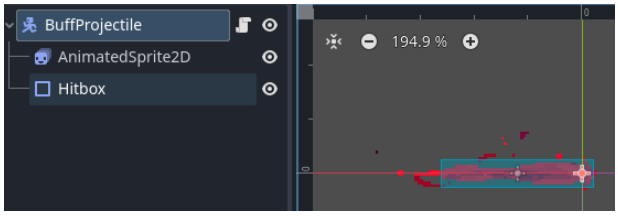


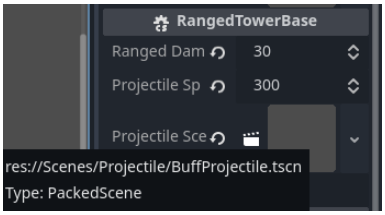
Other behavioural components require physics logic, and are therefore contained within Godot scene classes. Projectile takes in an attack / buff and provides it a trajectory, whereas AreaEffect takes in an action and performs it on all valid targets in range.



Export Variables

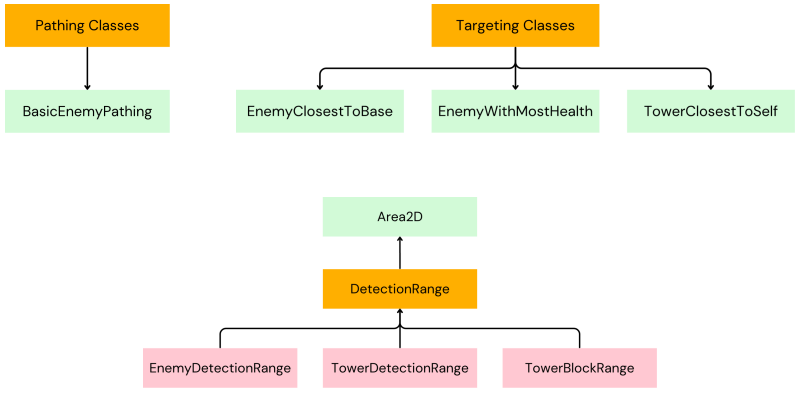
One benefit of such scene classes is the ability to declare them as an Export variable, which allows setting a default reference directly in the Godot editor. This allows for scenes like projectiles to be prebuilt before assignment to a unit, or be swapped out on demand.





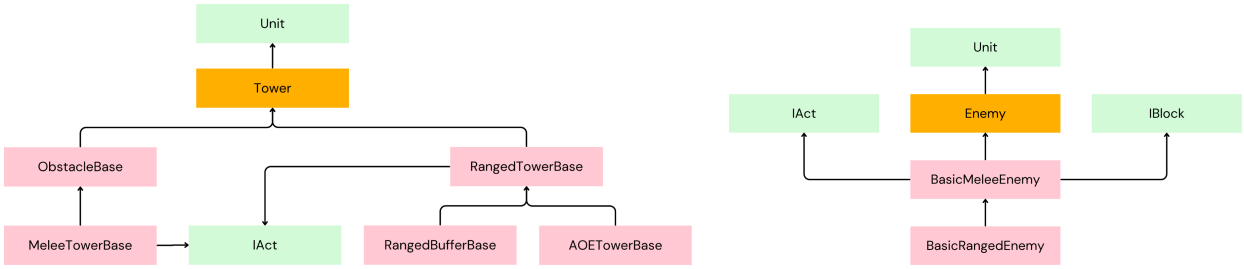
Separation of Concerns

With the offensive behaviour provided by actions, and defensive behaviour provided by the unit class, the rest of the responsibilities can be broken down into their own classes. This includes the pathing, targeting, and DetectionRange classes.



Flow of Logic

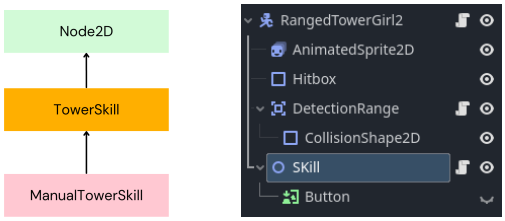
All the classes are now assembled to create Enemy and Tower classes through composition. These define a unit's base behaviour, from their stats to their basic skills. Primitive variables can be exported, which removes redundancy of creating new classes for units that share the same behaviour but have different stats.



Unit interaction is executed through layers of relay. For example, when BasicRangedTower attacks, it first obtains a list of targets from its detection range, which is then fed through a targeting class to obtain a target. The target is given to a BasicRangedAttack action, which creates the projectile using the tower's fields and feeds it an attack class set by the tower. The projectile then carries the attack and executes it on the enemy upon landing.

Scene Composition

In Godot, child nodes are fairly decoupled from their parents, and can function independently whilst the parent exists. We used this design to create TowerSkill, which acts as a separate offensive entity attached to the parent tower. While the skill stores its tower as a reference, the tower does not need to know what skill it has, enabling more flexibility in design.



SWE Principles

Single Responsibility Principle

Achieved by separating concerns and ensuring each class handles only one type of logic, such as tracking for detection ranges and target picking for targeting classes.

Open-Closed Principle

The former is achieved through composition which creates modularity. The latter is achieved using well planned abstraction that reduces backtracking. The enemy class is not created by modifying the unit class' code, but by adding a pathing class to it.

Liskov Substitution Principle

Achieved through well designed inheritance, such as by ensuring all subtypes of unit do not cause exceptions when responding to hits and buffs accepted by unit.

Interface Segregation Principle

We attempt to follow this principle as much as possible, avoiding adding unnecessary logic, such as actions, to classes like obstacle. However, passing around packed scenes pose a limitation - initialization methods must sometimes be overloaded rather than overridden, creating redundant code.

Dependency Inversion Principle

Achieved alongside the aforementioned modularity, with emphasis on decoupling. A ranged tower's detection range and targeting class each function independently.

SWE Patterns

Component Pattern

Godot's scene system follows a component pattern, where each node in a scene may be treated as a component providing different data or logic. Detection ranges, skills, hitboxes and sprites are some components that can be attached to an entity to extend functionality.

Observer Pattern

Both Godot signals and C# events enable the observer pattern. We use them extensively to decouple entity interactions like attacking and despawning, by having attackers subscribe to their target's despawn event such that they may readjust when their target is lost.

Singleton Pattern

Singletons are typically used when there is only one instance of it and multiple objects can use it. We used singletons to manage our audio system and to store static game data, such as enemy wave information, or basic entity details like default stats. This gives us quick access to such data at any time and from anywhere.

For example, our game has a main singleton named GameData which stores wave information. GameScene then communicates with WaveSpawner, which loads in all the enemy wave information for that specific map. The WaveSpawner then spawns in each enemy in the list.

Factory Pattern

Godot's packed scenes act as factory patterns that represent a scene blueprint. We often use them to enable more flexibility and modularity in building and creating entities, passing around projectile scenes to ranged units in place of projectile instances.

Problems Encountered

Besides the issues briefly mentioned in SWE Principles, we also encountered some other problems.

Balancing

Balancing the towers and enemies units in a tower defense game proved to be challenging, especially when accounting for players with different levels of experience. Finding the sweet spot for the difficulty of the game was tricky due to the multitude of factors involved such as the strength of towers and enemies, map layouts, tower deployment limits and base health. Moreover, since we are veteran tower defense players, our standard for difficulty might be different from the average beginner. We intend to perform user testing to gain this insight to make our game more challenging yet enjoyable for our players.

In Milestone 3, we added the Boss Map, which proved to be way more difficult than Map 1, 2 and 3 combined. The addition of the boss map was in response to the feedback surrounding the easy difficulty for the earlier maps. In Boss Map, our boss had a lot of attacks and phases, which required the player to adopt different strategies to counter each of them. For a beginner, this stage would be impossible unless he were to have sat through multiple iterations of the boss and had seen all the moves before, or perhaps read this README.

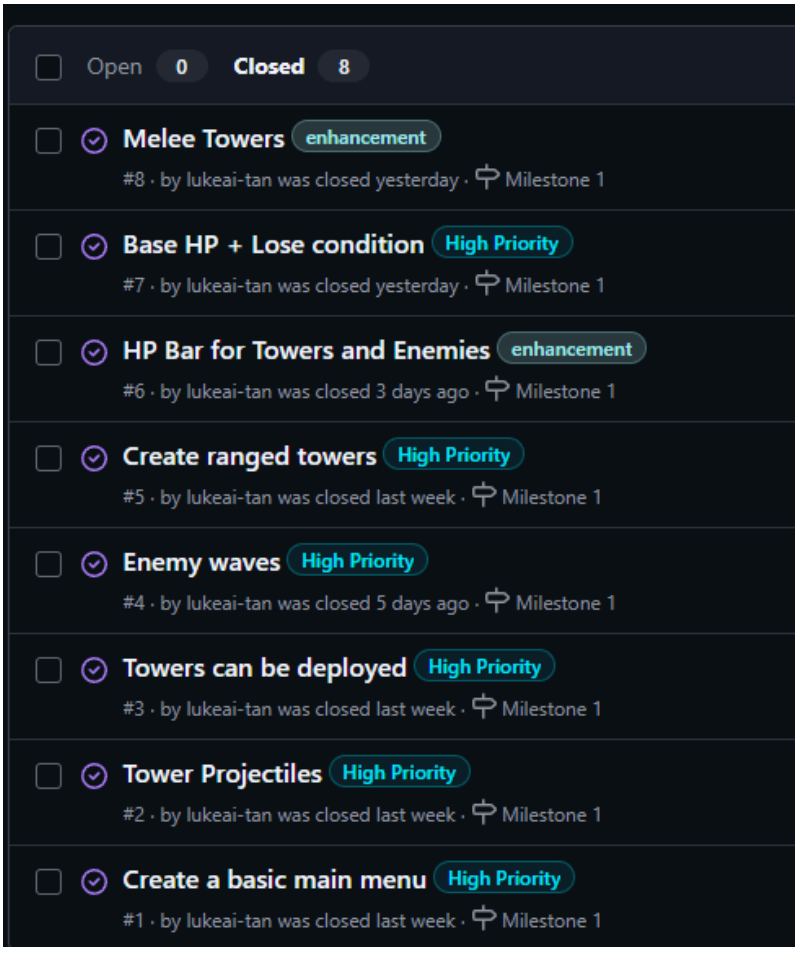
Project Management

Agile Methodology

Each milestone is completed in the form of 4 week sprints, where we detail the core features and extensions to be implemented in each milestone using a feature timeline. Although the milestone deadlines were set by the Orbital program, we still maintained internal agility by breaking down milestones into weekly goals, continuously integrating and testing new features, as well as using GitHub issues, labels and milestones to track task completion

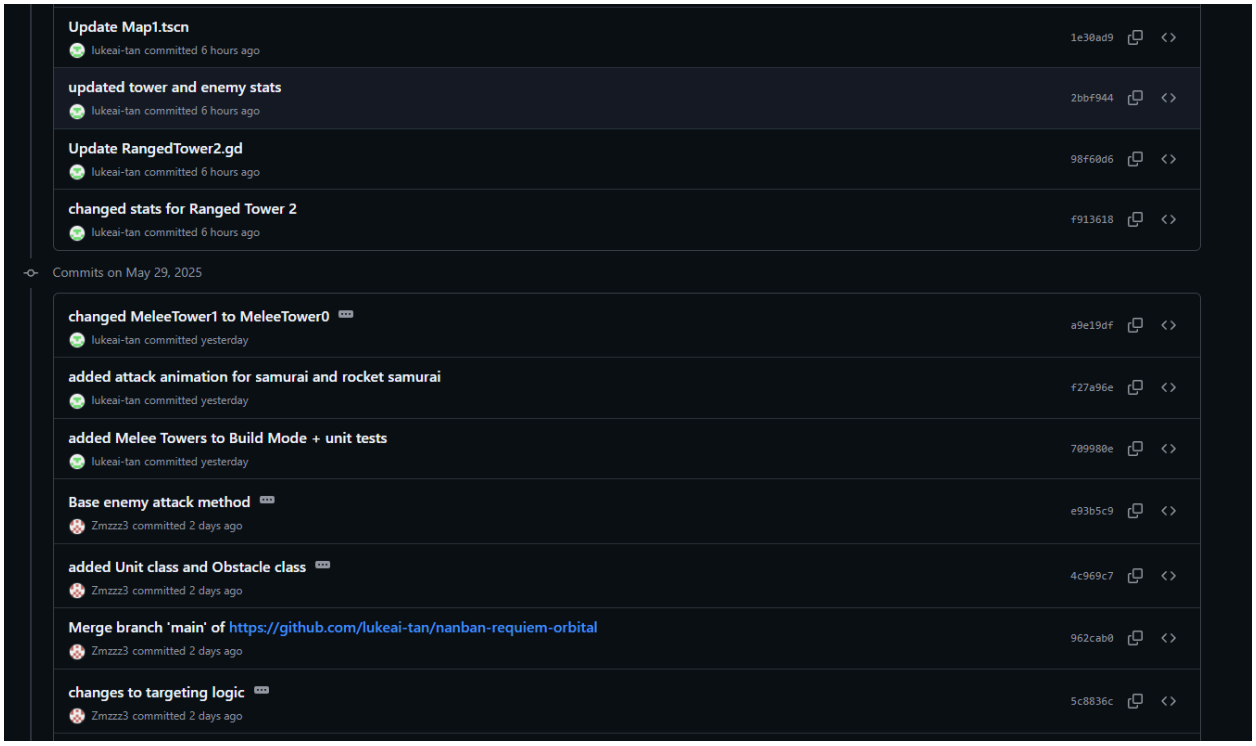
Issues / Labels / Milestones

To effectively manage development and ensure progress aligned with our goals, we utilised GitHub's project management tools, specifically milestones, issues, and labels, in conjunction with our outlined feature timeline.



Version Control

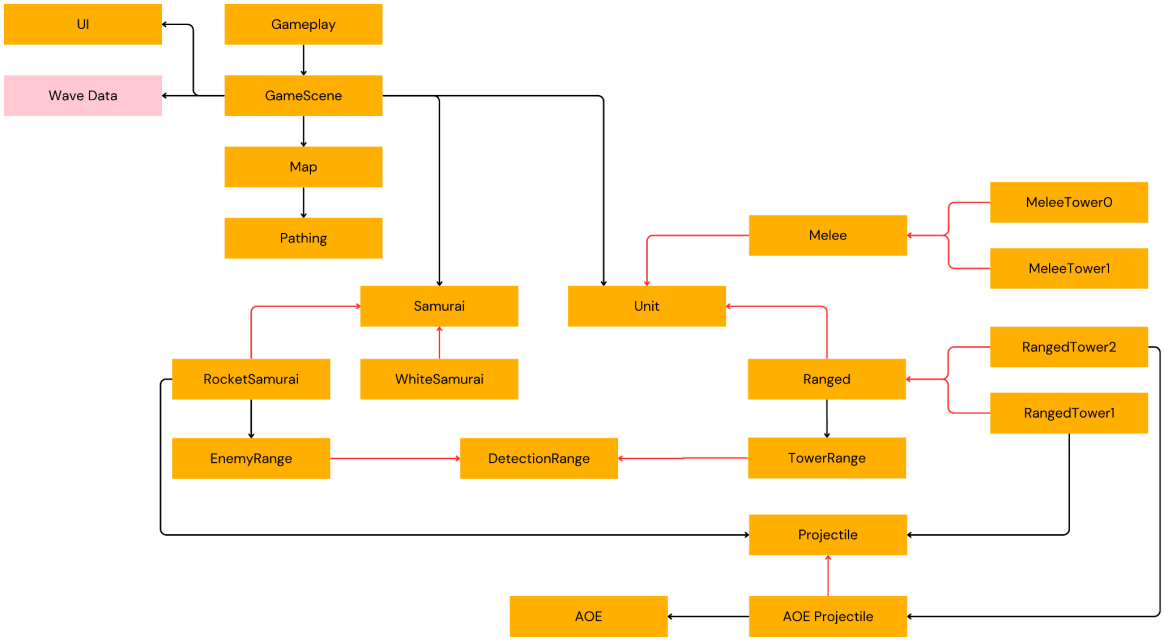
Using Git, we commit changes frequently and merge them into the main branch before pushing to GitHub.



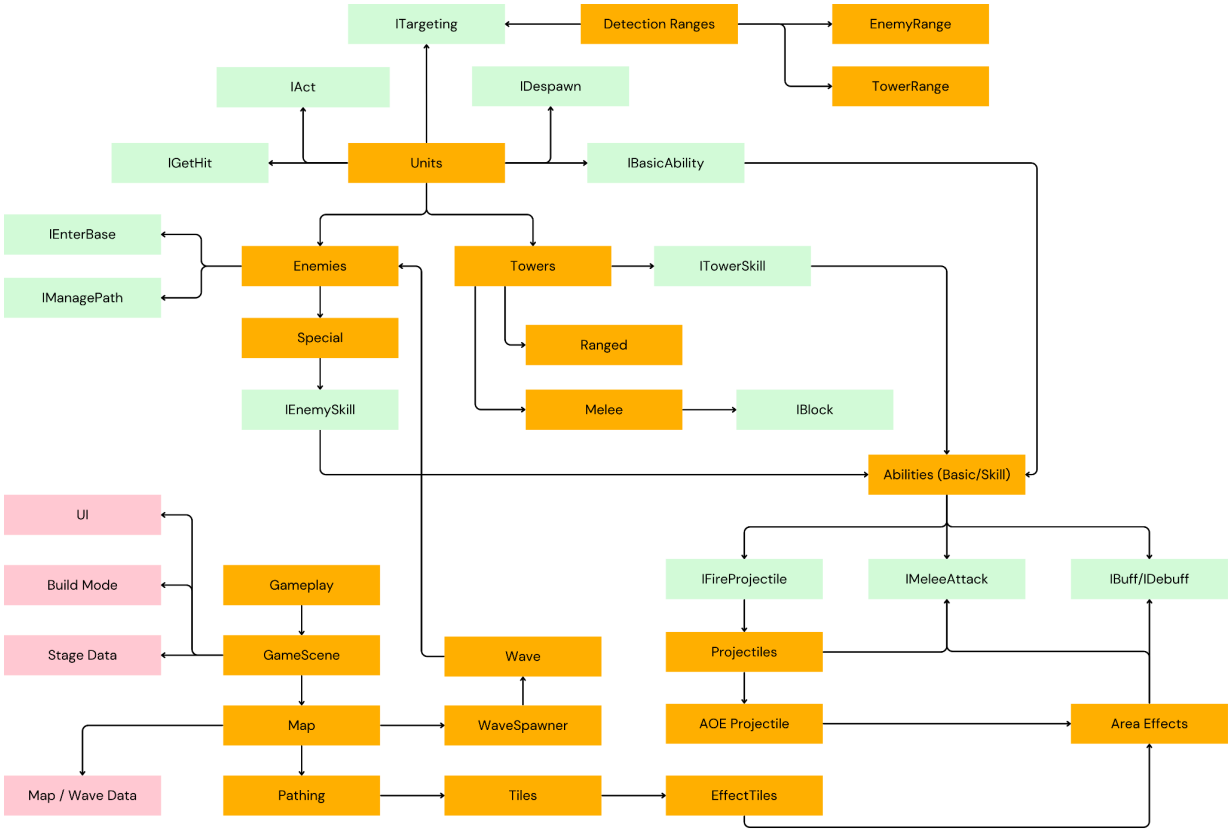
In Milestone 2, we adopted a branching strategy using two main branches: main and dev. We pushed new commits into the dev branch before creating a pull request to merge into main. This workflow ensures continuous integration, minimises merge conflicts and supports efficient team collaboration.

Appendix A: Unused Class Diagrams

Milestone 1:



Milestone 1 (Planned):



Appendix B: Old SWE Principles (MS1)

Single Responsibility Principle

An example is the AOE class - its only responsibility is to apply damage to enemies within its range. An example for improvement would be the current DetectionRange class - which handles both detection and targeting. In our proposed diagram, we would shift the targeting to a separate class implementing the ITargeting interface.

Open-Closed Principle

In our proposed diagram, we plan to use interfaces in C# to define modular classes with specific roles - such as IFireProjectile for firing projectiles, or IBuff/Debuff for applying a self-buff or enemy debuff. These classes will allow us to define different units using the same base class by extending its behaviour in different ways.

Liskov Substitution Principle

Our class hierarchy is created based on the LSP. For example, Unit class is only able to receive hits, while its subclass Melee is able to both receive hits and block enemies. This allows for situations such as in Samurai, where we are able to refer to any subtype of Unit because we are sure it responds to hits within Unit's expectations.

Interface Segregation Principle

In our proposed diagram, we define many different interfaces with their own specific responsibility. When creating tower classes, we only implement interfaces that are relevant to the tower's role - such as the IBlock interface that allows melee towers to block enemies. Ranged towers do not implement this as they do not block enemies.

Dependency Inversion Principle

The proposed ITargeting interface is an example of an abstraction - it only defines the action of picking a target, not how the target is picked. A tower only needs to ensure that its targeting class returns a target, allowing for modifications to the targeting logic without requiring changes to the tower class, leading to less coupling.

Art and Animation Sprites

1. Map Tilesets: Top-down Tanks Redux · Kenney. Done by Kenney
2. Ranged Tower assets: Tower Defense (Top-Down) · Kenney . Done by Kenney
3. UI Icons: Game Icons · Kenney. Done by Kenney
4. Samurai assets: Samurai 2D Pixel Art by Mattz Art. Done by Mattz Art
5. Main Menu theme: Synth Cities Environment by ansimuz. Done by ansimuz
6. Battle theme: Carnival of Jesters.
7. Projectiles: Particle FX by RagnaPixel . Done by RagnaPixel
8. DuskBorne: NightBorne Warrior. Done by CreativeKind
9. Voidmire: Boss: Demon Slime by chierit - Game assets. Done by chierit
10. Map 5 OST: Mozart | Lacrimosa but the Epic Version . Done by Adil Kamal - YouTube
11. Map 5 Background: Goodfon Wallpaper
12. More projectiles: 50 Assorted Sprite Effects by jellyfish0. Done by jellyfish0
13. Boss Map OST: The Army of Minotaur / Epic Orchestral Battle Music (CC-BY) . Done by Makai Symphony - YouTube