國立臺北科技大學
2022 Spring 資工系物件導向程式實習
期末報告

Plants vs. Zombies

第 27 組

108590048 方文昊

108590056 鄭琳玲

一、簡介

1. 動機

PopCap Games, a subsidiary of Electronic Arts (EA), created the Plants vs. Zombies video game franchise. The series follows David "Crazy Dave" Blazing's associates as they use his plants to defend against a zombie invasion led by Dr. Edgar George Zomboss. Although Plants vs. Zombies is a simple game, it holds a special place in the hearts of many people. There is also a subset of Plants vs. Zombies fans who enjoy experimenting with the game's unique gameplay. They even counted the attributes of different plants and created a small almanac with information about the plants and zombies featured in the gameplay. We wanted to make our childhood game through this opportunity to learn and apply what we have learned from Object Oriented Programming class.

2. 分工

方文昊

- ○ Writing, debugging, and packaging the main game program.

鄭琳玲

- ○ Find, processing and making sure that the materials can be used, and write some logics of the game.

二、遊戲介紹

1. 遊戲說明

We offered ***Adventure Mode*** as the main mode in our gameplay. There are 10 playable levels spread across two maps (***Day*** and ***Night***), with 5 levels on Day and 5 levels on Night.

a) The lawn is grid-style, with the player's house to the left. The player arranges various types of plants on the grid's individual squares.

b) Each plant has a unique defense strategy, such as shooting, exploding, or blocking. The player can select 8 of plant types from seed packets at the start of each level, and must pay to place them with a currency called ***sun***. The player collects sun by clicking on sun icons that appear randomly on the lawn or by using plants that produce sun, such as Sunflowers.

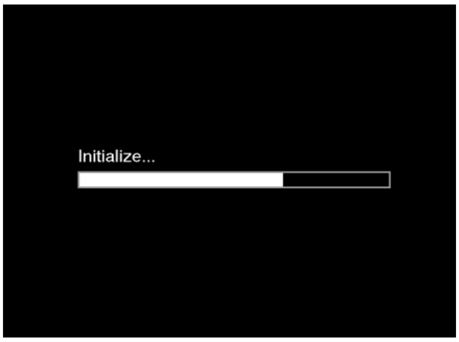
c) A single-use lawnmower is placed at the left end of each lane; if a zombie reaches this end, the mower activates and kills all zombies in that lane.

d) If a zombie reaches the end of a lane whose mower has already been used, the player has to restart the level.

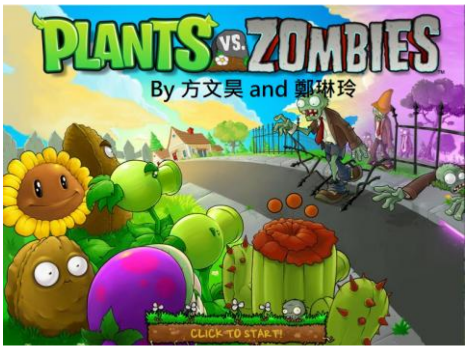
Buttons	Description
Q + Left Click	Go to the previous level (goes to Main Menu on Lvl 1-1)
W	Add 500 suns
E + Left Click	Go to the next level (goes to Main Menu on Lvl 2-5)
A	Zombies move twice as fast
S	Zombies move normally
D	Zombies move half as fast
Z	Reset all the cooldowns for planting plants
X	Kill all zombies on screen and drop the note
C	Kill all zombies on screen

2. 遊戲圖形

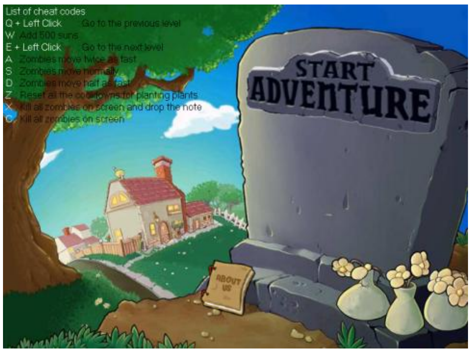
1. When the player launches the game, the game will load.



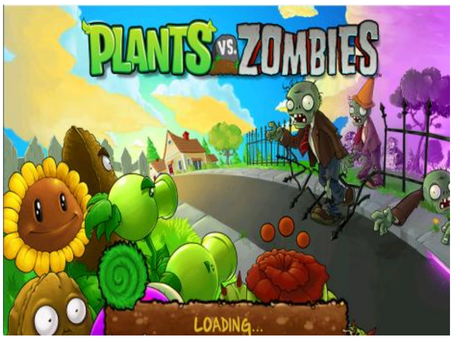
2. The screen that appears after the game has been loaded.



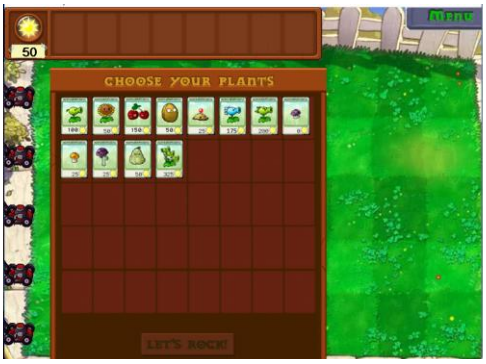
3. Game Main Menu; Featuring Start Adventure button and About Us


List of cheat codes
Q + Left Click Go to the previous level
W Add 500 suns
E + Left Click Go to the next level
A Zombies move twice as fast
S Zombies move normally
D Zombies move half as fast
Z Reset all the cooldowns for planting plants
X Kill all zombies on screen and drop the note
C Kill all zombies on screen

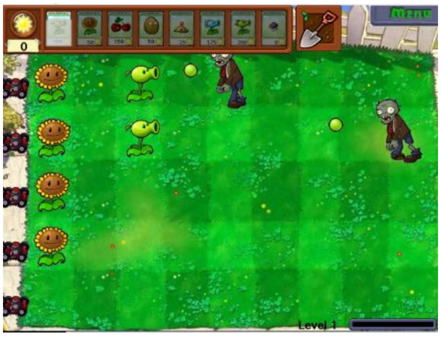
4. Once Start Adventure Button is clicked, then loading screen will appear and the user can click anywhere to continue



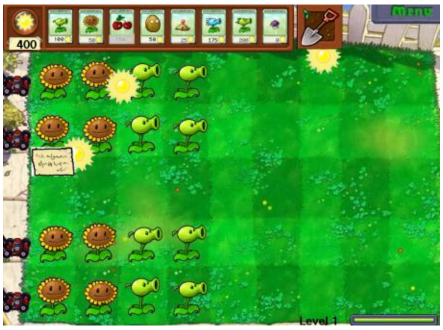
5. The player then pick the plants that they want to use and start by clicking Let’s Rock.



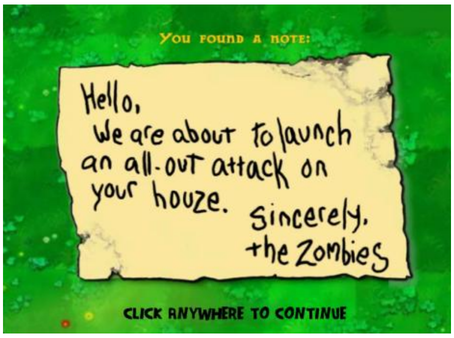
6. After the plants has been chosen, the gameplay will start.



7. Once the level has been completed, then a Note will show up.



8. And when the note is clicked, the note will show up.

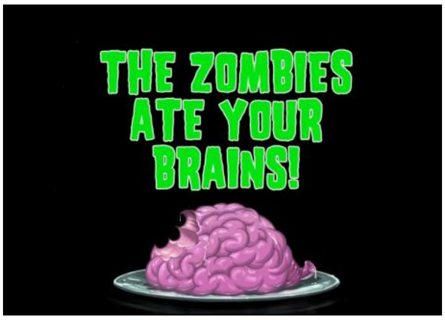


You found a note:

Hello,
We are about to launch an all-out attack on your houze.
Sincerely,
the Zombies

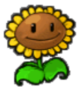
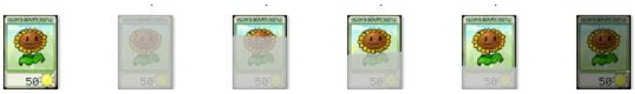
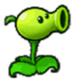
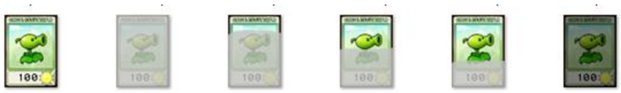
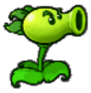
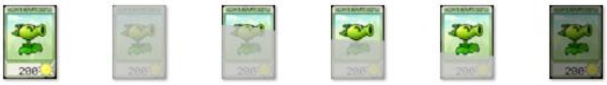
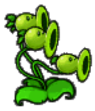
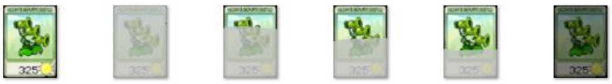
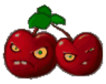
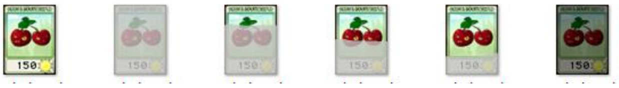
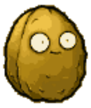
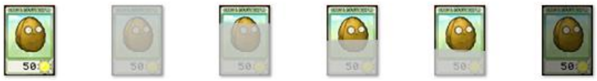
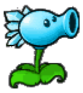
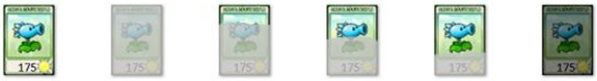
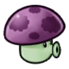
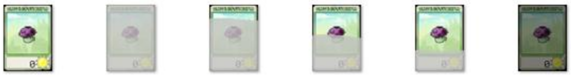
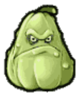
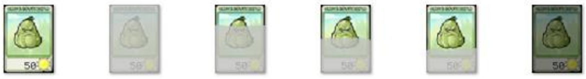
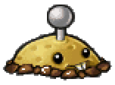
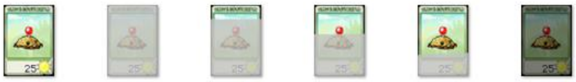
CLICK ANYWHERE TO CONTINUE

9. If the player fails to complete the level, then the game will be over.



THE ZOMBIES ATE YOUR BRAINS!

植物

Name	Appearance	Card	Output
Sunflower			
Peashooter			
Repeater			
Threepeater			
Cherry Bomb			
Wall Nut			N/A
Snow Pea			
Puffshroom			
Squash			N/A
Potato Mine			

Scaredy shroom			
Iceshroom			
Jalapeno			

殭屍

Name	Appearance	
	Normal (Normal, Attack & Dead State)	Frozen (Normal, Attack & Dead State)
Zombie		
Flag Zombie		
Conehead Zombie		
Buckethead Zombie		
Newspaper Zombie		

3. 遊戲音效

音遊戲音效	說明
01. Title Screen.wav	Loading Screen
02. Main Menu.wav	Main Menu
03. Choose Your Seeds.wav	Choosing Plant
04. Day Map.wav	Used during Day Map
06. Night Map.wav	Used during Night Map
07. On Note UI.wav	Level Up Note
08.-losemusic.wav	Game Over
SFX cherrybomb.wav	When Cherrybomb is exploded
SFX jalapeno.wav	When Jalapeno is exploded
SFX lawnmower.wav	When Lawnmower is used
SFX newspaper rip.wav	When the Newspaper rips
SFX plant.wav	When the plant is planted
SFX-potato-mine.wav	When potato mine exploded
SFX-splat.wav SFX-splat2.wav SFX-splat3.wav	When Peashooter, Repeater and Threerepeater bullet hits the zombie
SFX-Sun-Picked.wav	When the sun is clicked
Voices-newspaper-rarrgh.wav Voices newspaper rarrgh2.wav	When the Newspaper Zombie lost its newspaper

三、程式設計

1. 程式架構

a. UI Elements

UI Elements consist of elements that repeatedly appear through the game or have a logic complex enough that we deemed they should have their own classes. These elements are:

1. Background: Loads a bitmap at position 0,0
2. SeedCard: Cards used in-game for the player to select a plant to be planted.
3. Button: a button that changes bitmap when it is hovered on or clicked
4. Checkbox: Similarly to Button, this changes only when clicked. Furthermore, when clicked, it executes a function that was given when it was created.

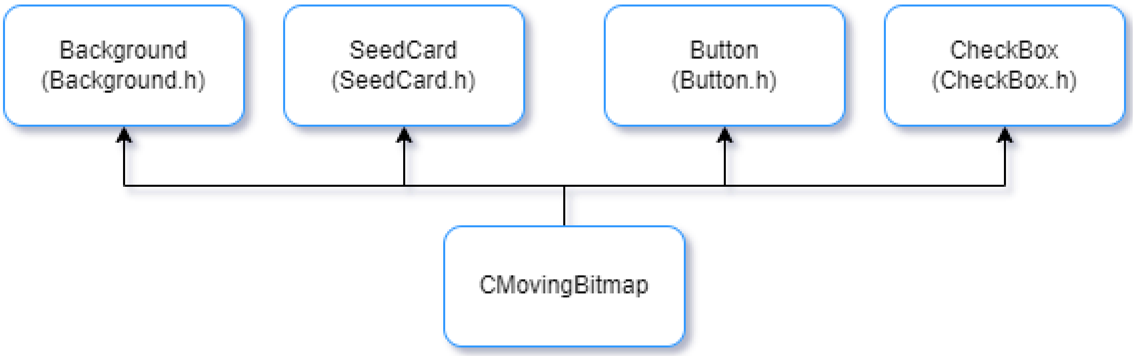


Diagram of the Inheritance of UI elements.

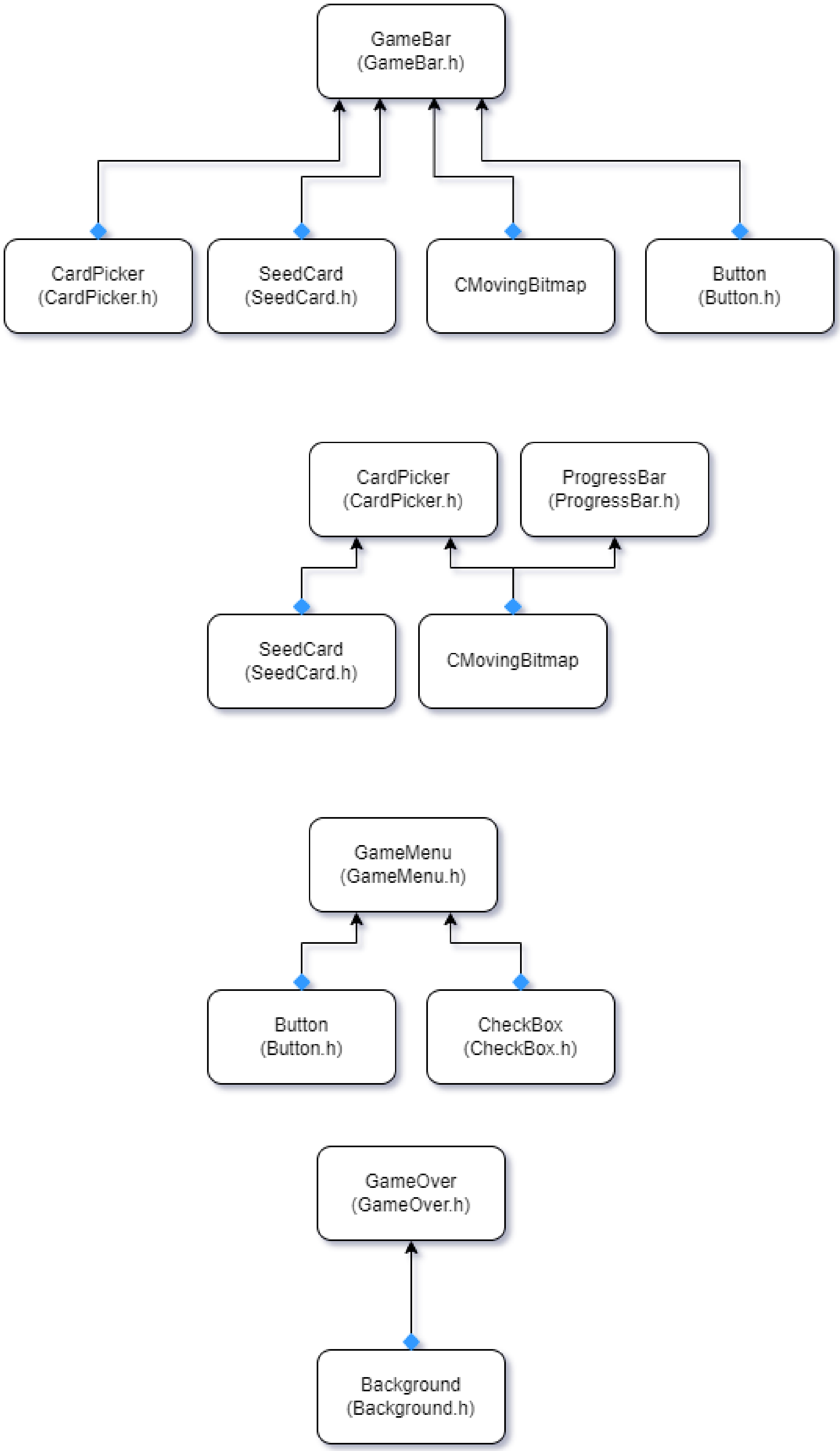


Diagram of the composition of UI Elements

b. Miscellaneous Elements

Miscellaneous elements are objects used during game logic. These includes:

1. Bullet: the bullets shot by some plants

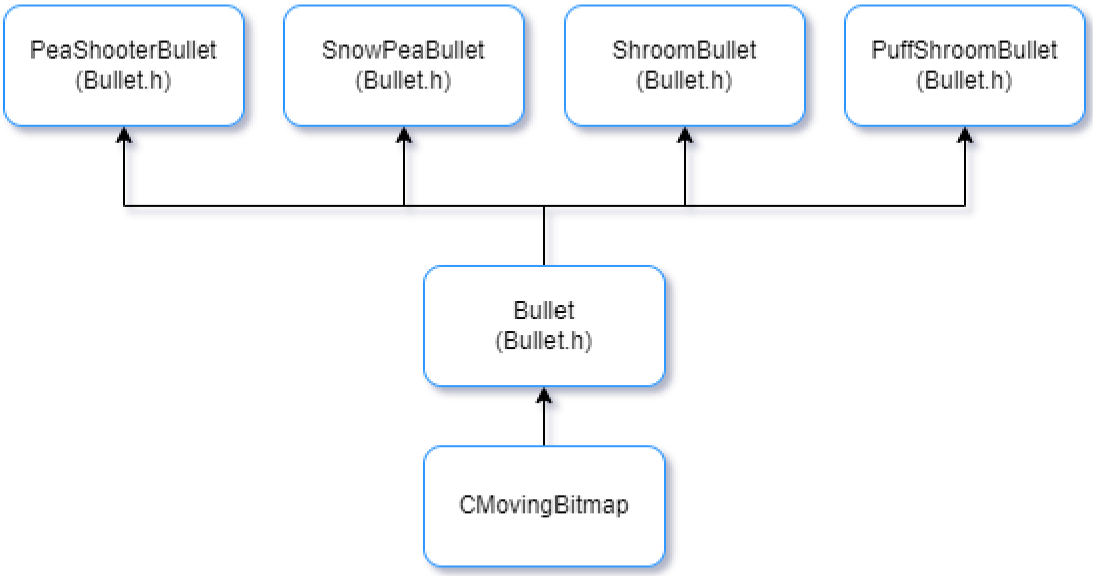


Diagram of the Inheritance of the Bullet class and its child classes.

2. Cooldown: a class that uses a timer to track the cooldown for Plants, Zombies and Seed Cards. It does also handle the logic for the in-game clock.

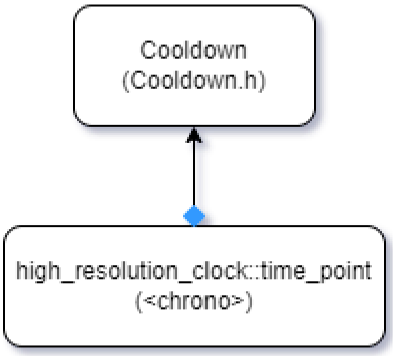


Diagram of the Composition of the Cooldown class.

3. Lawnmower: the lawnmower is an object that prevents the player from losing when a zombie walks off the board in a specified lane for the first time. It removes all zombies on that lane and then is removed from the match. While the removal parts are handled by the map, this class handles the animation of the Lawnmower.

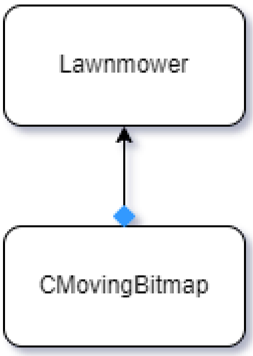


Diagram of the Composition of the Lawnmower class.

4. Sun: the suns are the player's currency. The sun classes handle the animations and hold the value of the suns that are present in the board, be they fell from the sky or they were generated by a plant.

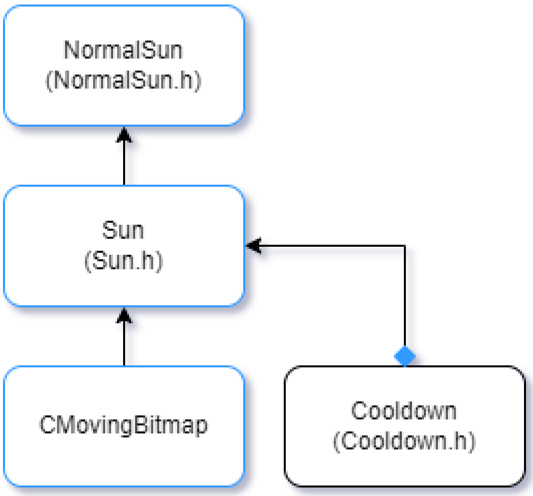


Diagram of the Inheritance and Composition of Sun and its child classes.

5. SoundInfo and Soundboard: SoundInfo holds the data relevant to a sound, be it music or sound effects. Soundboard was made in order to allow the player to disable sound effects or music in the main menu. It does also make it safer for the game to load sounds.

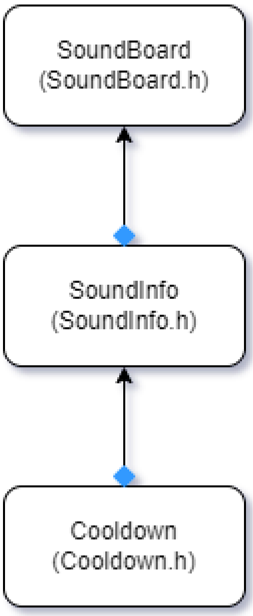


Diagram of the Composition of SoundInfo and SoundBoard.

c. Plants

Plants are a fundamental part of the game. They allow the player to have an input on the game and their type and use are the fundamental differences between winning or losing a game.

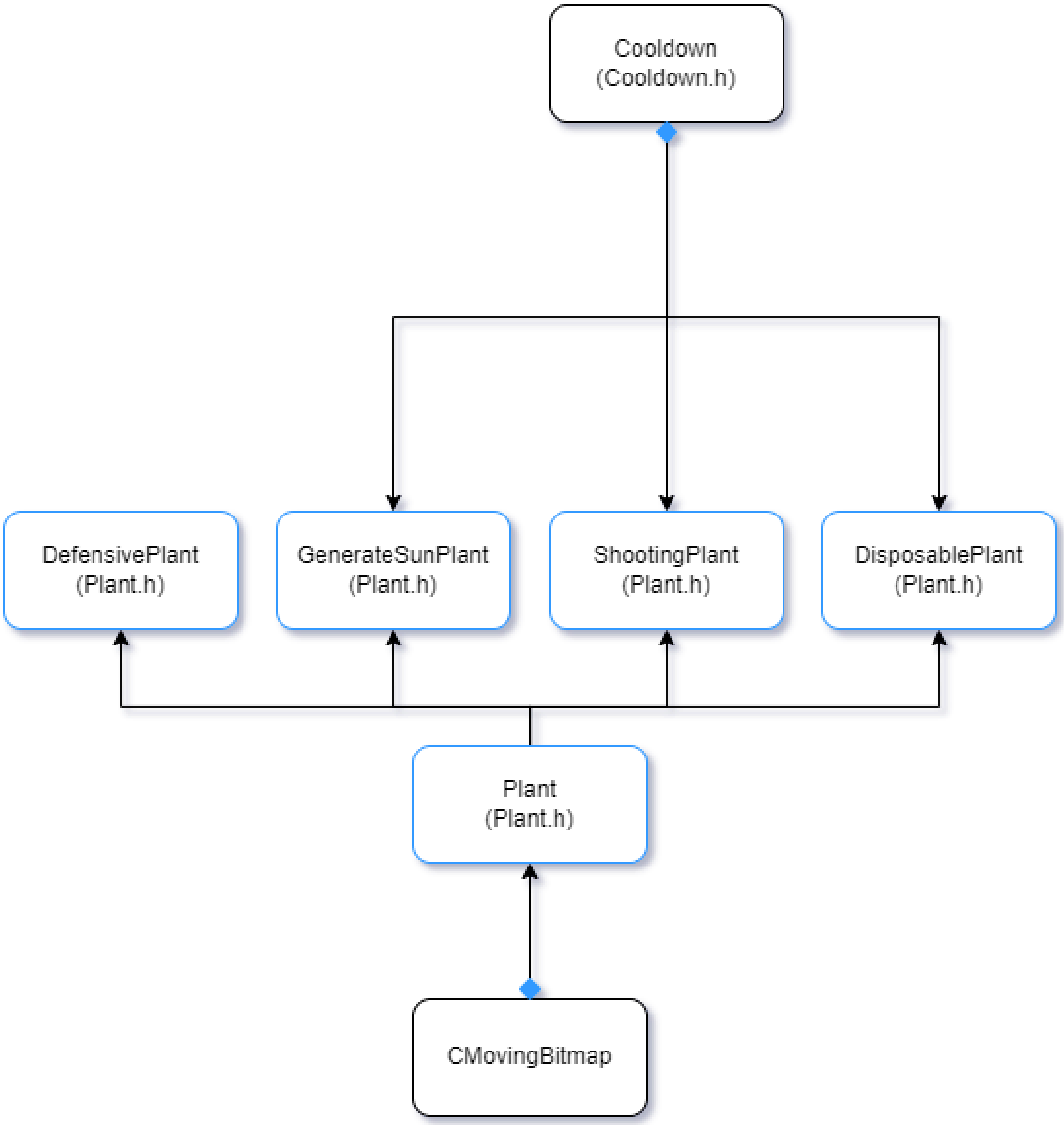


Diagram of the Inheritance and Composition of Plant classes

Our game has 4 classes of plants, with each class having one or multiple type of plants :

1. Defensive Plants: simple plants with high endurance but don’t do anything.
    a. Wall Nut

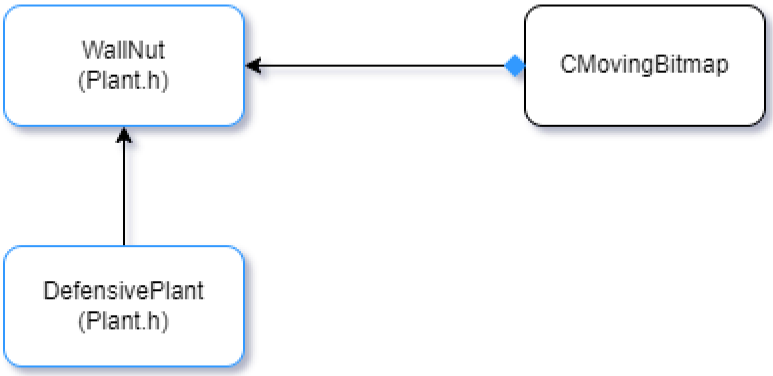


Diagram of the Inheritance and Composition of Defensive Plants

2. Sun Generating Plants: plants that generate sun over time
    a. Sun Flower

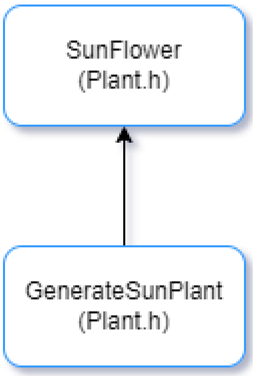


Diagram of the Inheritance and Composition of Sun Generating Plants

3. Shooting Plants: plants that shoot projectiles at zombies. The zombie doesn't necessarily need to be in the same lane for the plant to shoot.
    a. Pea Shooter
    b. Repeater Pea
    c. Threepeater
    d. Snow Pea
    e. Puff Shroom
    f. Scaredy Shroom

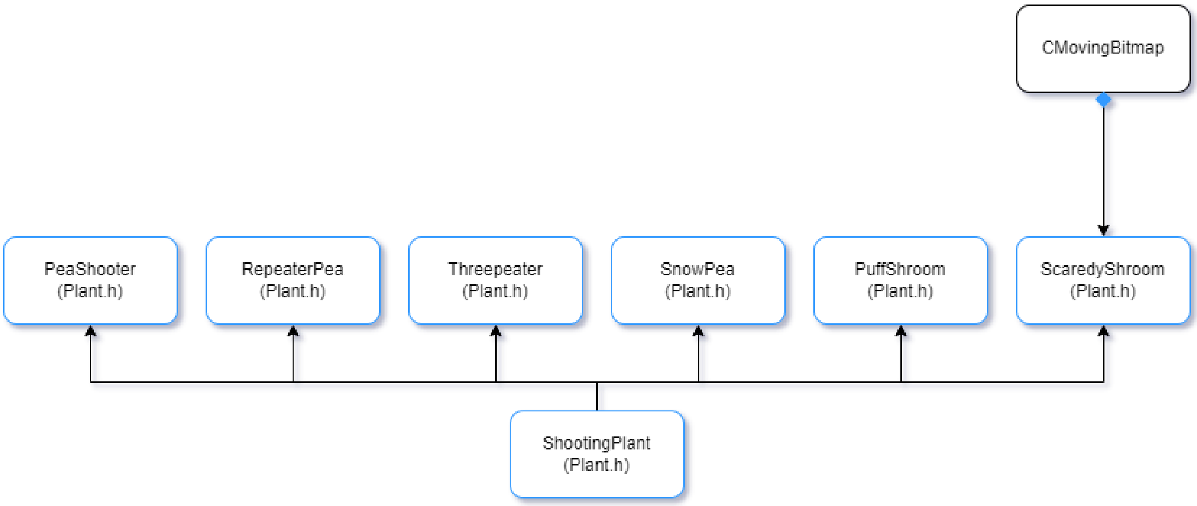


Diagram of the Inheritance and Composition of Shooting Plants

4. Disposable Plants: one-use only plants that upon being used they are deleted.
    a. Ice Shroom
    b. Potato Mine
    c. Squash
    d. Jalapeno
    e. Cherry Bomb

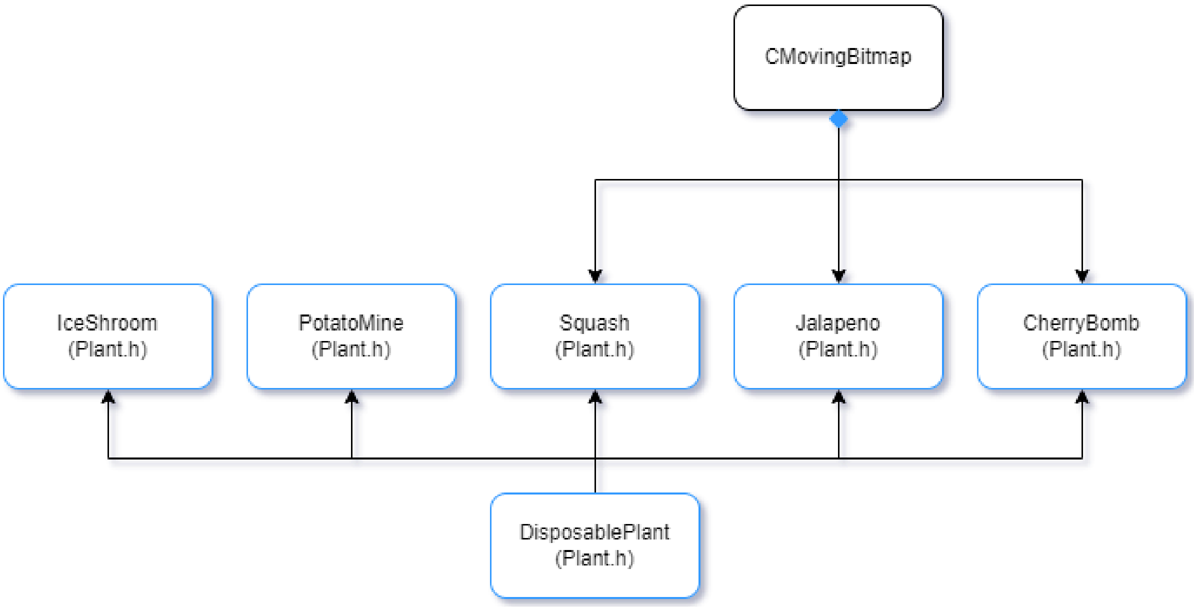


Diagram of the Inheritance and Composition of Disposable Plants

d. Zombies

Zombies are the enemies the player will have to face. While the ones we have added have an extremely simple artificial intelligence, where they will walk leftward until they find a plant. When they find a plant they will attack it. When defeated, zombies will have their head roll out while they will fall to their knees. In our game, we have 6 type of zombies:

1. Normal Zombies: regular zombies
2. Buckethead Zombies: Most resilient type of zombies.
3. Conehead Zombies: The second most resilient type of zombies. Less resilient than Buckethead Zombies.
4. Flag Zombies: regular zombies but they hold a flag indicating to the player the beginning of a wave of zombies.
5. Newspaper Zombies: Third most resilient type of zombies. Less resilient than Conehead Zombies.

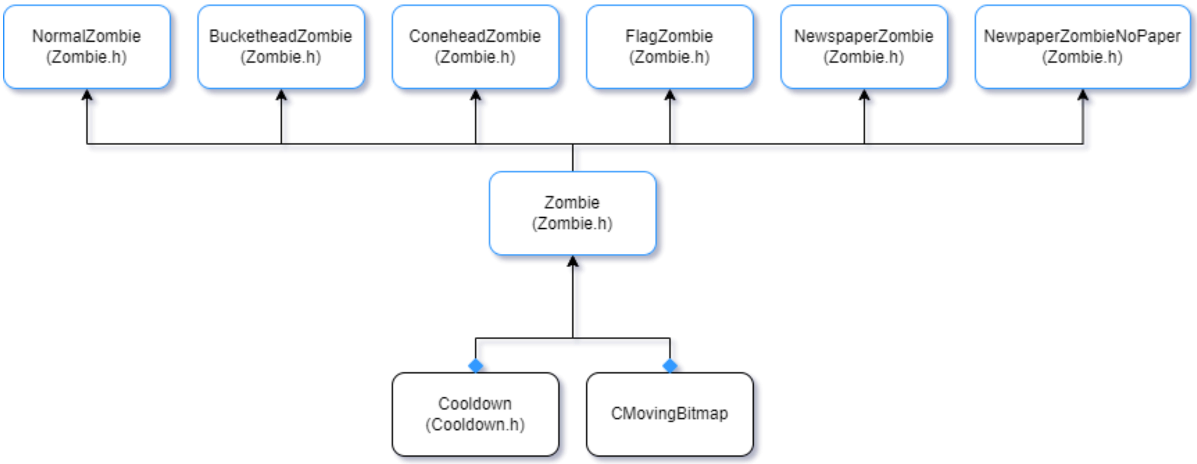


Diagram of the Inheritance and Composition of Zombies

e. GameMode

GameMode is a type of object that dictates and handles the game's logic. It tracks if the player is in the Main Menu, or the About Us page, or is currently playing the game. It effectively is a layer on top of `CGameStateRun`.

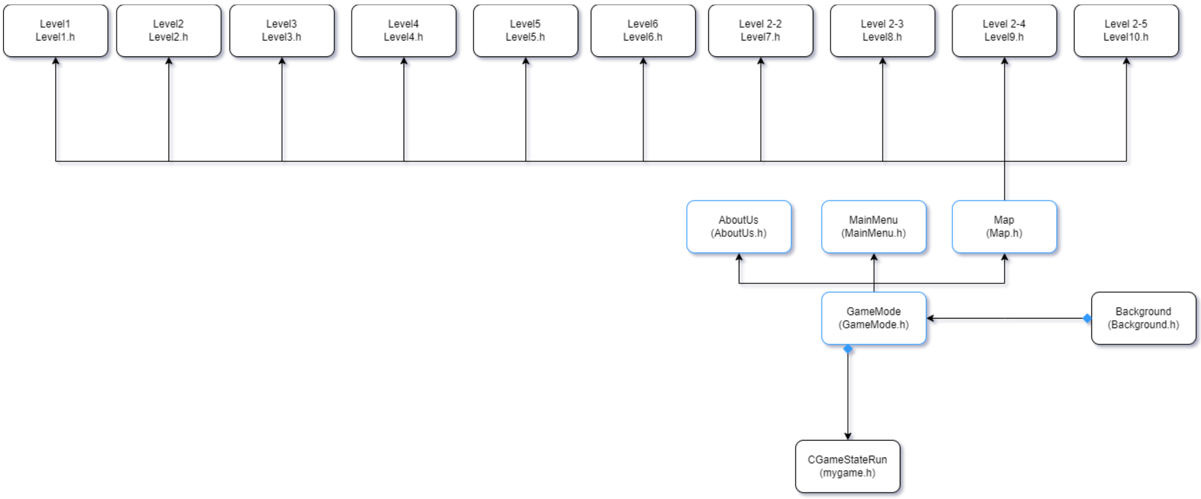


Diagram of the Inheritance and Composition of GameMode

In our game, we have 3 type of Game Modes:

1. Main Menu: this is the Main Menu the player goes to after initializing the game.
2. About Us: this is a screen containing the credits of the game.
3. Map: this is where the game's main logic is handled. While it is the child class of GameMode, it is also the parent class for the different levels in the game.

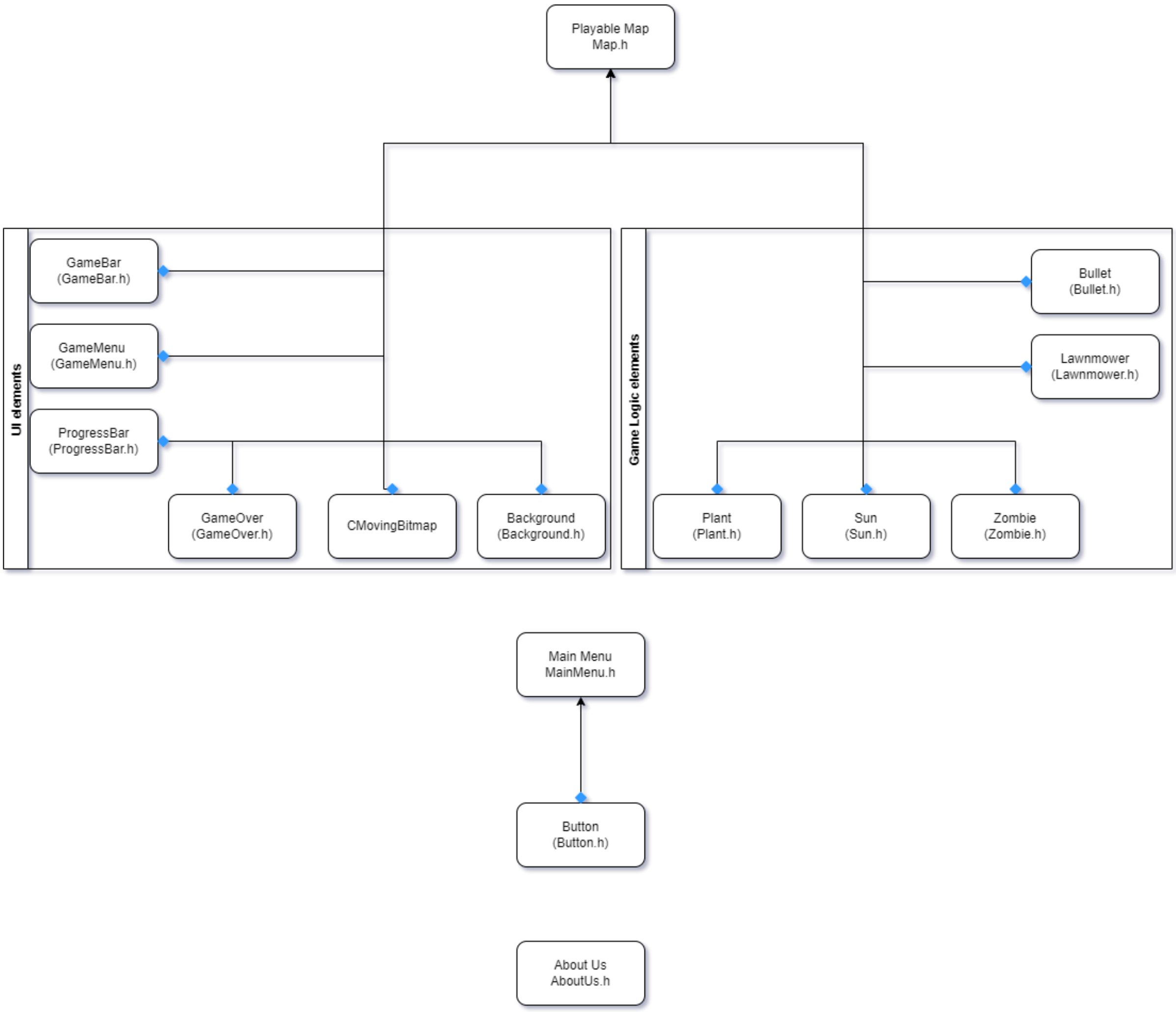


Diagram of the Inheritance and Composition of GameMode child classes.

2. 程式類別

Will do on Friday or after

類別名稱	標頭檔 （.h）	原始碼 （.cpp）	說明
Background	18	28	A child class of CMoving Bitmap used to set up a bitmap or a set of bitmaps at position 0,0 on the screen.
Button	18	38	A child class of CMoving Bitmap used to represent the clickable buttons in the game
CardPicker	134	106	The class responsible for the logic behind the player’s picking card actions at the beginning of each level.
CheckBox	24	53	A child class of CMoving Bitmap used to represent checkboxes in the game. It has only been used in the in-game menu for the SoundFX and Music checkboxes.
GameBar	58	308	The class responsible for tracking the amount of Suns the player has, if a plant card can be used by the player, as well as the plants the player chose at the beginning of the level.
GameMenu	65	176	The class responsible for the pause menu inside a level. It is also responsible for the options in the in-game menu.
GameOver	28	35	The class responsible for displaying a Game over screen when the player fails.
ProgressBar	56	88	The class responsible for displaying the

			progress bar inside the level as well as track the player’s progress.
SeedCard	66	135	Class responsible for the seed cards the player uses to select the plants they want to use in a level as well as the cards on the Game bar that allow the player to plant plants.
Level_1	62	N/A	A child class of Map, it holds the constant data for Level 1, such as Zombie spawning logic, what the previous and next level are, as well as the file path for the map cleared screen.
Level_2	94	N/A	A child class of Map, it holds the constant data for Level 2, such as Zombie spawning logic, what the previous and next level are, as well as the file path for the map cleared screen.
Level_3	100	N/A	A child class of Map, it holds the constant data for Level 3, such as Zombie spawning logic, what the previous and next level are, as well as the file path for the map cleared screen.
Level_4	102	N/A	A child class of Map, it holds the constant data for Level 4, such as Zombie spawning logic, what the previous and next level are, as well as the file path for the map cleared screen.
Level_5	115	N/A	A child class of Map, it holds the constant data for Level 5, such as Zombie spawning

			logic, what the previous and next level are, as well as the file path for the map cleared screen.
Level_6	62	N/A	A child class of Map, it holds the constant data for Level 6, such as Zombie spawning logic, what the previous and next level are, as well as the file path for the map cleared screen.
Level_7	95	N/A	A child class of Map, it holds the constant data for Level 7, such as Zombie spawning logic, what the previous and next level are, as well as the file path for the map cleared screen.
Level_8	100	N/A	A child class of Map, it holds the constant data for Level 8, such as Zombie spawning logic, what the previous and next level are, as well as the file path for the map cleared screen.
Level_9	101	N/A	A child class of Map, it holds the constant data for Level 9, such as Zombie spawning logic, what the previous and next level are, as well as the file path for the map cleared screen.
Level_10	163	N/A	A child class of Map, it holds the constant data for Level 10, such as Zombie spawning logic, what the previous and next level are, as well as the file path for the map cleared screen.
MapsDependencies	11	N/A	Small file that includes all the .h files for

			the Levels
Tile_Positions	75	N/A	File that lists the columns right, middle and left positions as well as the rows' up, middle and down positions on the screen.
AboutUs	23	47	A child class of GameMode, it displays the About Us image as well as the logic where if the user clicks anywhere while it is being displayed, the user will be sent back to the Main Menu.
GameMode	22	N/A	Parent class that determines if the current screen in-game is the Main Menu, the About Us screen, the Loading Screen or the playable Levels.
LoadingScreen	29	55	A child class of GameMode, it displays a Loading screen that waits for a specified amount of time (currently 1.0s) and then changes the displayed image to indicate to the player that loading is complete.
Main_Menu	56	136	A child class of GameMode, it displays the Main Menu and contains the logic for when the user clicks the buttons.
Map	123	805	A child class of GameMode, this is the main logic behind all the levels. It accepts the list of zombies that will appear and other parameters and will handle the player inputs and the game logic.
Bullet	30	N/A	Main logic behind the bullets shot by some plants. The bullets specific to each plant inherits from it.

PeaShooterBullet	6	14	Child Class of Bullet, it represents the bullets shot by the PeaShooter plant as well as the Repeater and the Threepeater.
ShroomBullet	17	33	Child class of Bullet, it is the projectile shot by mushrooms.
SnowPeaBullet	8	28	Child class of Bullet, it represents the bullets shot by the SnowPea. It applies a slowdown effect on zombies on contact.
Cooldown	39	77	Timer class that uses high_resolution_clock::time_point to measure the time between 2 events. It is usually used to measure the time needed for a plant to recharge before performing its action again,
Plant	205	N/A	Base class for all placeable plants in the game. It also contains the definition for its child classes.
Lawnmower	30	46	Class representing the Lawnmower animation. The game logic of the lawnmower is handled by Map
NormalSun	33	21	Child class of Sun, it represents a sun produced by a Sunflower or that fell from the sky and gives the player 25 Sun currency.
Sun	47	60	Base class of NormalSun, it represents the sun in the map.
CherryBomb	15	78	Child class of DisposablePlants, it is a placeable plant that explodes when a

			zombie is nearby, killing all zombies in its blast radius.
IceShroom	15	72	Child class of Disposable plants, it is a placeable plant that explodes when a zombie is nearby, freezing all zombies currently on screen.
Jalapeno	15	83	Child class of Disposable plants, it is a placeable plant that explodes when a zombie is nearby, killing all zombies on the current row.
PeaShooter	10	29	Child class of ShootingPlants, it is a placeable plant that will start shooting bullets when a zombie is in the same row the plant is in. When the bullet makes contact with the zombie, it will deal damage to the zombie.
PotatoMine	12	57	Child class of Disposable plant, it is a placeable plant that explodes when a zombie is nearby, killing that zombie. Unlike the CherryBomb, it requires some time from planting for it to become active.
PuffShroom	10	57	Child class of ShootingPlants, it is a placeable plant that will start shooting its own bullets when a zombie is in the same row and 3 columns ahead of the plant. When the bullet makes contact with the zombie, it will deal damage to the zombie.
RepeaterPea	10	40	Child class of ShootingPlants, it is a Pea Shooter that costs twice as much but shoots

			2 bullets instead of 1. Has the same fire rate.
ScaredyShroom	15	127	Child class of ShootingPlants, it is a Mushroom that sleeps during day maps and during night maps it is active. When active, it shoots at zombies that are in the same row as it.
SnowPea	9	29	Child class of ShootingPlants, it is a Pea Shooter that has the same fire rate as the normal Pea Shooter but when its bullets come into contact with a zombie, it applies a slowdown effect on the zombie.
Squash	15	79	Child class of DisposablePlants, it is a placeable plant that when a zombie is in front of it, it will do a hopping animation and then kill all the zombies in the column in front of it. It is deleted after being activated.
SunFlower	10	35	Child class of SunGeneratingPlants, it is a plant that generates sun over time.
Threepeater	11	68	Child class of ShootingPlants, it is a Pea Shooter that has the same fire rate as the normal Pea Shooter but it starts firing when there is a zombie in the row below, the same row or the row above. When shooting, it shoots one projectile on each of the 3 aforementioned rows.
WallNut	16	69	Child class of DefensivePlants, it is a plant that has high endurance and resilience but

			does absolutely nothing. A wall.
AudioConsts	31	N/A	File containing the IDs used for audios. The IDs are indeed immutable.
EntitiesUtil	33	N/A	File containing the template functions to find an object in a vector and safely delete an object in a vector.
GameModeConsts	45	N/A	File containing the const enumerators to decide which state the game should transition to as well as the const enumerator to indicate what type of zombie should be spawned with the instructions given from the levels to the Map.
GameModeUtils	145	N/A	File containing the function to change the state of the game. It also contains a Pseudo Random Number Generator that uses the game clock to set the random seed and then returns a number between the minimum and maximum boundaries given.
Soundboard	80	200	Static class responsible for controlling the sounds in the game. It is used to make safe use of CAudio as well as to control if the player input on the presence of music or SFX in the game.
SoundInfo	38	26	Class representing a file’s information, such as its id, its filePath and its track length as well as if it is being played or not and if it is being played on repeat.
VK_Consts	46	N/A	File defining all virtual keys. From 0-9 to A-Z.

Zombie	111	67	Base class for all appearing zombies in the game.
BucketheadZombie	5	102	Child class of Zombie, it is the most resilient class of zombie we implemented.
ConeheadZombie	5	102	Child class of Zombie, it is the second most resilient class of zombie implemented, right after BucketheadZombie.
FlagZombie	5	100	Child class of Zombie, it behaves like a NormalZombie but has different animations.
NewPaperZombie	5	94	Child class of Zombie, it is the third most resilient class of zombie implemented.
NewPaperZombieNoPaper	5	92	Child class of Zombie, it is the NewPaperZombie when it loses its paper.
NormalZombie	5	94	Child class of Zombie, its the most basic and common type of zombie the player can encounter.
config	4	N/A	The game's configuration, we only changed the in-game resolution and disabled the player from expanding the window.
mygame_initialize	1	8	The game state when initialized. We only added the initial screen and our names.
mygame_run	3	25	The game state when it is running, our changes were minimal here, only giving it a GameMode class pointer and passing its User input functions and their arguments to the GameMode pointer. With our current architecture, GameMode is indeed more of

			a layer running on top of CGameStateRun that handles the game’s logic while CGameStateRun communicates between the framework and our layer.
Total	2948	4047	Grand Total (.h + .cpp) = 6995 lines

3. 程式技術

Most of the Data structures used in the program are `vector` from the `<vector>` library in order to group multiple objects or pointers into a single and easily resizable collection. We also used the `find` algorithm from the standard library `std` in order to find objects in our vector. This was mostly used to either find an object in our vector or delete an object in our vector as we frequently ran into the issue of a null pointer being left in the vector regardless of the algorithms we tried to mitigate such issues.

Furthermore, we also made use of `map` from the `<map>` library in order to have an easily accessible collection of constant objects. Finally, in order to measure and handle time-related operations, we used the `high_resolution_clock::time_point` class and the `duration_cast` algorithm from the `<chrono>` library.

四、結語

1. 問題及解決方法

No.	Issue	Solution
1	When a certain card is chosen, the chosen card in row two will not be counted and `Let's Rock' Button will not light up.	Some unit testing are done and several lines of code are removed.
2	Exception thrown: read access violation. ${ptr}-> was 0xDDDDDDDD	● Created a static template function that checks if an object in a pointer is in the vector ● Created a static template function that safely removes an object from a vector of pointer and deletes it ● Used the first function to safely delete the Sun* in displaySun over Map ● Used the first function to safely delete the Zombie* in zombies over Map ● Used the first function to safely delete the Bullet* in bullets over Map ● Added if clauses that begin by using the second function to check that the pointer being used is valid when using either of the 3 pointers above.
3	When we developed CAudio, and used .mp3 for the game, an error would pop up, and a suggestion saying that using .wav will solve the issue. But somehow when we added .wav, another issue shows up.	The issue is solved because we realized that the audio name shouldn't have spaces. So we added underscores (底線) to replace space.
4	There are memory leak on the	We had forgotten to include virtual in the

	framework.	destructors of our base classes, which meant that the destructors for the child classes would not be called.

2. 時間表

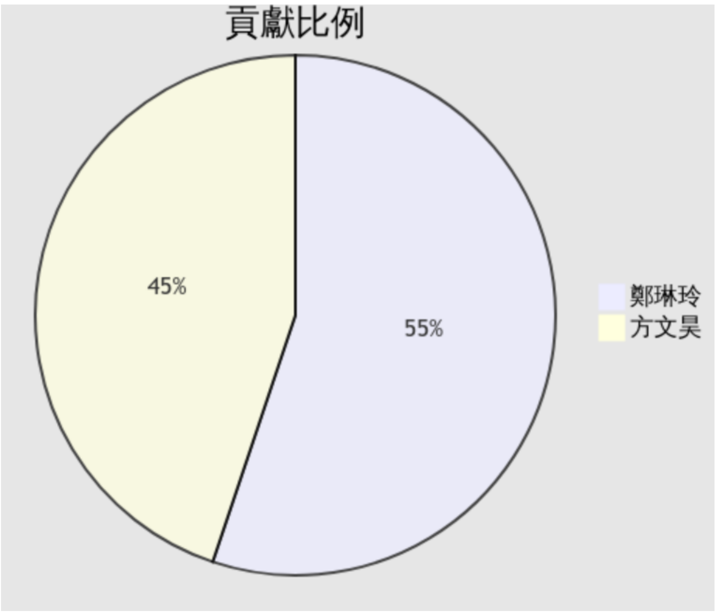
Week Count	方文昊 （小時）	鄭琳玲 （小時）	Description
1	10	8	介紹
2	8	8	- Completing Git Framework Practice - Game Features Planning - Dividing Labor - Finding materials
3	6	8	- Main Menu - Day Map Background - Card Picking Board
4	6	15	- Sun Random Placement Logic - Map Terrain Logic
5	10	30	- Sun Clicking and Counter Logic
6	20	5	- Plant Logic: Sunflower, Peashooter - Plant Placement Logic
7	20	5	- Zombie Walking, Attack, Dead Logic - Developed Zombie: Normal Zombie - First Demo
8	20	25	- Plant Logic: Cherrybomb, Wallnut, Potatomine - Plant Placement Logic - Picked Card Logic: Cooldown, Not Enough Sun and Unable to Use Logic
9	5	15	- Plant Bullet Attack Logic - Shovel Logic

			- Plant Logic: Snowpea, Repeaterpea, Puffshroom, Scaredyshroom, - Detection Logic
10	5	15	- Landmower Logic - In Game Settings Menu UI Logic - Developed Zombie: Conehead Zombie, Buckethead Zombie, Newspaper Zombie, Flag Zombie - Progress Bar Logic
11	15	30	- In Game Settings Menu Logic: Previous Level, Next Level, Restart Level, return to Main Menu - Level Ending Logic - Level 1 - 5 Difficulties Logic
12	7	10	- Fixing Exception thrown: read access violation. ${ptr}-> was 0xDDDDDDDD error - Plant Logic: Squash and Threeshooterpea - Second Demo
13	8	10	- Plant Logic: Iceshroom - Night Map Logic - Music Logic
14	5	7	- Music Logic - Researching on how to draw class diagram - Cheat Key added
15	5	8	- Fixing Music Logic issue - Made level 5 - 10 (Night Map)
16	10	5	- Fixed Memory Leak - Fixed Music Logic issue - Completing the report
17	10	5	Final Demo
Total	170	209	

3. 貢獻比例

鄭琳玲：170 小時，33.47%

方文昊：209 小時，66.53%



4. 自我檢核表

	項目	是否完成	說明為何不能完成
1	解決 Memory leak	完成	
2	自定遊戲 Icon	完成	
3	About 畫面	完成	
4	遊戲中說明如何操作	完成	
5	發佈(Release)版本	完成	
6	承上項，該版本可以正常運作	完成	
7	報告字型、點數、對齊、行距、頁碼等格式正確。	完成	

5. 收穫

鄭琳玲

This semester taught me how to use C++ to create games using a framework. When developing large-scale projects, I learned to plan properly at the start of development and pay attention to the cleanliness of my code, or is there any better way to improve development efficiency and game performance, and I also learned to communicate properly with team members. I also encountered many problems when researching games, but I tried to solve them one by one and ended up with a game that can be placed in my portfolio.

6. 心得或感想

鄭琳玲

This semester, I learned how to convert a game from a single frame to a complete game output. In addition to program planning and writing, the process entails gathering materials, recording sound effects, and so on. I still want to improve the UI, make the operation more user-friendly, and add different sound effects to enrich the game as I write the entire game. Finishing this game and continuing to improve it feels very rewarding.

I am grateful to the teacher for allowing us to use the concepts learned in the object-oriented programming class to complete our own work. When we encounter difficulties, he will advise us on how to solve the problems. I also want to thank the teacher for his assistance when we ran into a strange animation error in our program. That static error also helped me grasp a number of underlying concepts.

方文昊

This semester, I learned the hardships of making a game first hand. While one may think it is easy to make a game, the reality is that with deadlines developers may attempt to cut corners in order to have a finished product as soon as possible. While in our case, we did try to structure the game's classes such that it would be easier to reuse the code in the future and expand the current game's capabilities, this came at the cost of what we could show during the different demos we had. In particular during the first demo, where we were still writing out the game and UI logic, we only had a single level that was barely playable as opposed to other classmates who had much more. This however did pay off in the end as when it came to adding more levels, most notably before the second demo, we had a much easier time doing so since the levels are all child classes of the game's main logic.

In the end, I am grateful to the teacher for giving us this opportunity to not only apply the concepts we learnt during OOP, but also for giving us this experience of a collaborative project with my partner and the experience of developing a project with strict deadlines.

7. 對於本課程的建議、想說的話

鄭琳玲

I hope that this class has a clear roadmap of developing games by using medium difficulty games. So each team can follow that roadmap to reach a certain percentage of completion by each week. And hope this class can be remote since this class doesn’t need the students to use school’s equipment.

方文昊

The only feedback I have for this class is that the tutorial and documentation on LGF could be slightly expanded as there are some shortcomings. While we were successful in understanding the cycle of functions being called on LGF, we did have some major issues when it came to implementing audio into our game: this issue was due to us trying to load and play an audio file that had spaces in its file name. Furthermore the error description from LGF didn’t help us understand either.