Game Design Project Document

Mancala Defence

YAN Zehao, XIU Lei, XIAO Yang

June 11, 2021

Contents

1 Game Mechanism **4**
1.1 Overview . . . 4
1.2 The Map and Cells . . . 5
1.2.1 The Map . . . 5
1.2.2 The Favourite Cells and Vulnerable Cells . . . 5
1.2.3 Overloading and Exhaustion . . . 5
1.2.4 Special Rules for Spawning Units on Cells . . . 6
1.2.5 Attack Damage Modification for Unit on Cell . . . 6
1.3 Mancala . . . 6
1.3.1 General Rule for Mancala . . . 6
1.3.2 Special Rules for Mancala . . . 7
1.3.3 Chain Effect of Mancala . . . 7
1.4 Units . . . 8
1.4.1 Unit Attributes and the Building Activity . . . 8
1.4.2 Unit AI . . . 8
1.5 Enemies . . . 8
1.5.1 Enemy Types . . . 8
1.5.2 Waves . . . 8
1.5.3 Enemy AI . . . 9
1.6 Cards and Coins . . . 9
1.6.1 Cards . . . 9

2 Game Strategy **9**
2.1 Overview . . . 9
2.2 Resource Management . . . 10
2.3 Cards and Coins . . . 10
2.4 Units . . . 10
2.4.1 Short-term Planning and Long-term Planning . . . 10
2.5 Cell Efficiency . . . 11
2.6 Mancala . . . 12

3 UI Design **12**
3.1 Overview . . . 12
3.2 Start Scene . . . 13
3.3 Select Stage . . . 14
3.4 Game Scene . . . 15

3.5 Pause UI . . . 16
3.6 End UI . . . 17
3.7 Upgrade UI . . . 18

4 UI Implements **18**
4.1 Map . . . 18
4.2 Cell . . . 19
4.2.1 Choose Cell . . . 19
4.2.2 Cell Material Change . . . 19
4.2.3 Highlight Cell . . . 19
4.3 Unit Factory . . . 20
4.3.1 Generate Unit . . . 20
4.4 Unit . . . 20
4.4.1 Unit Spawn . . . 20
4.4.2 Unit move . . . 21
4.4.3 Unit Choose . . . 21
4.4.4 Unit Upgrade . . . 22
4.4.5 Lifebar . . . 22
4.5 Card and Card Factory . . . 23
4.5.1 Draw Card . . . 23
4.5.2 Use Card . . . 23
4.5.3 Reset Card Position . . . 23
4.6 PlayerAssets . . . 23
4.6.1 Cards Image . . . 23
4.7 Player Interface . . . 23
4.7.1 Hint Message . . . 23
4.7.2 Choose Type . . . 24
4.8 GameManager . . . 24
4.8.1 Player Option . . . 24
4.8.2 Pause Button . . . 24
4.8.3 Life, Coin and Wave . . . 25
4.9 Camera . . . 25
4.9.1 Camera Move . . . 25
4.9.2 Camera Control . . . 25
4.10 Start Menu . . . 26

5 Stage Design **26**
5.1 Wave Information . . . 26

5.2 The Favourite Cells and Vulnerable Cells of Map 27
5.3 Opening setting 28

6 Game Mechanism Implement 28
6.1 Enemy System 28
6.1.1 Enemy Factory 28
6.1.2 Wave and Waypoint 29
6.2 Map 29
6.2.1 Map Design 29
6.3 Unit 31
6.3.1 Unit attack 31
6.3.2 Unit upgrade 31
6.4 Bullet 31
6.5 Card System 32
6.5.1 Draw Card Speed 32
6.5.2 Numerical system 32

7 List of Terms 33

8 Contributors 33

1 Game Mechanism

1.1 Overview

Mancala Defence is a **tower defence game** combining mechanisms of **mancala** and **card drawing**. The game goal, like many other tower defence games, is to prevent *enemies* from entering *base*. To achieve this, the player should dynamically placing *units* on the *map*, and the *units* will automatically attacking the *enemies* until they are killed. Once placed, the *units* cannot be easily moved to the other *cells* except through the *Mancala* operation. One of the economy system is built upon card drawing.

The Game Mechanism needs to be designed to encourage player to play *Mancala*, to use *cards* instead of *coins* (another resource). And it should also force players to balance the advantage and disadvantage placing *units* on *Vulnerable Cells*.

1.2 The Map and Cells

1.2.1 The Map

The *map* consists an *enemy spawning point* (*ESP*), a *base*, a *path* where enemies will follow towards the *base*, and may *cells* where friendly *units* can be placed on.

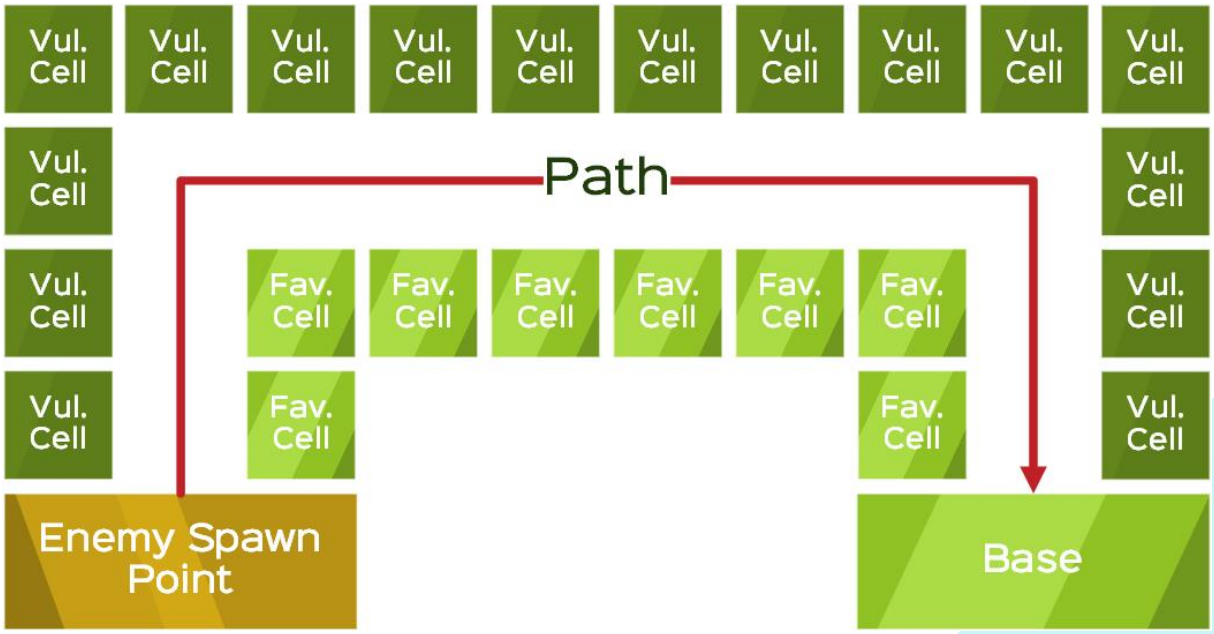


Figure 1: the *map*, only an example

1.2.2 The Favourite Cells and Vulnerable Cells

The *cells* are divided into two categories: *Favourite Cells* (shown as Fav. Cell above) and *Vulnerable Cells* (shown as Vul. Cell above). *Base* is a special *favourite cell*, while *enemy spawning point* is **not** a *cell*, as player cannot place *units* on it.

1.2.3 Overloading and Exhaustion

Cell will be *overloaded* when too many *units* are placed on it. After some time (determined by how many *units* are placed on it) it will become *exhausted*. All the *units* on it are all disabled (will no attack) unless they are moved to other *cells* (might by *Mancala* move). The *cell* will stay *exhausted* unless *unit* on it is cleared out. *Base* can **neither** be *overloaded* **nor** be *exhausted*.

1.2.4 Special Rules for Spawning Units on Cells

- *Units* can not be directly spawned on a *Vul. cell* unless it is *activated*, that is, any *unit* has been move onto it beforehand (through *Mancala*).
- *Units* can **never** be spawned on the *base*.
- *Units* can **never** be spawned on the *cell* that has number of unit reached the limit.

(In code the *base* is represented by `BaseCell`)

1.2.5 Attack Damage Modification for Unit on Cell

- *Units* on *vul. cells* will have attack damage decreaced.
- If all *units* (more than 1) on the cell is of the same type, their attack damages will be increased.

1.3 Mancala

1.3.1 General Rule for Mancala

Mancala, which name comes form the tabletop game *Mancala*, is a move allowing player to redistribute *units*. Player should specify a *cell* to perform *Mancala* first. All the *units* on it will then distributed into the succeeding *cells*, one *unit* per *cell*.

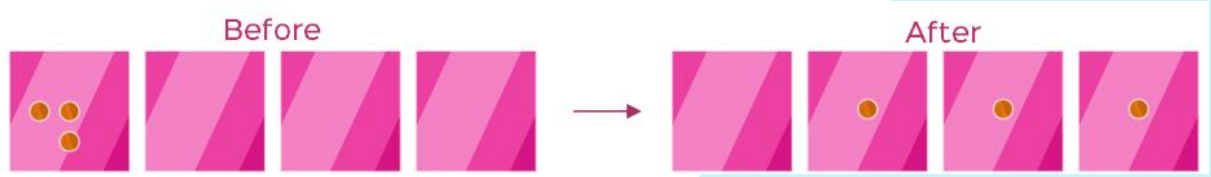


Figure 2: *Mancala* Example

The sequence of *cells* is: The *fav. cell* closest to *ESP* ->the adjacent *fav. cell* ->… -> the *fav. cell* closest to *Base* -> the *vul. cell* closest to *Base* -> the adjacent *vul. cell* -> … -> the *vul. cell* closest to *ESP* -> the *fav. cell* closest to *ESP*. The following figure is a clearer description. (Notice that the loop **does not** pass *ESP*)

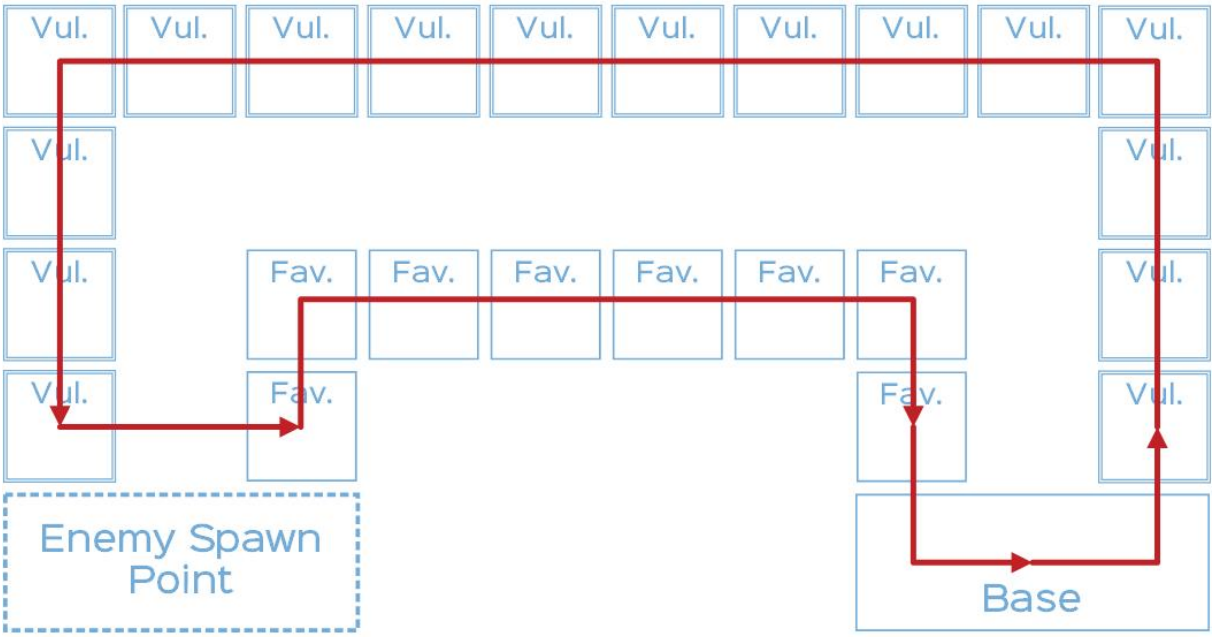


Figure 3: *Mancala* Sequence

1.3.2 Special Rules for Mancala

When the last *unit* is placed on the *base*, player can preform another *Mancala* **for free**.

1.3.3 Chain Effect of Mancala

Mancala is a way to resolve *exhaustion*. But due to its redistribution mechanism, it may trigger chain effect. The player should think twice before they move.

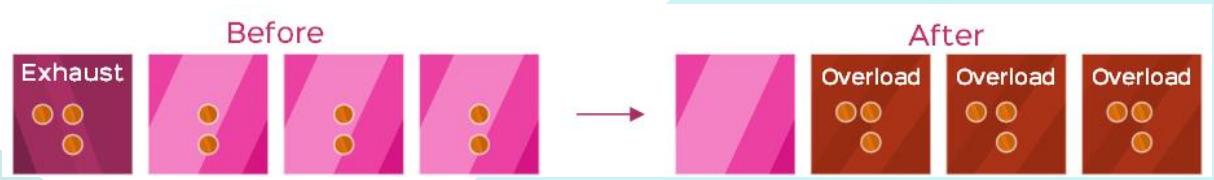


Figure 4: *Mancala* Chain Effect

1.4 Units

1.4.1 Unit Attributes and the Building Activity

Units has three attribute values: life, damage, and *skill*. *Skill* is the amount that a *unit* can contribute to *building activity* at a time.

When placed on the *base*, *units* will contribute to the *building activity* .Instead of attacking the *enemies*. The rate of *building activity* (the sum of *skill* for all *units* on the *base*) is related the frequency of card drawing.

Units has 3 types: White, Green and blue. It has a base attribute values shown as following:

Table 1: Unit Types

type	life	damage	skill
White	+	-	++
Green	++	+	-
Red	-	++	+

(-: disadvantage, +: advantage, ++: great advantage)

1.4.2 Unit AI

Unit will automatically attack *enemy* entered its range.

1.5 Enemies

1.5.1 Enemy Types

There are two kinds of enemies: one is Common Enemy(C. enemy), and the other is Elite Enemy(E. enemy). E. enemy can be further divided into three types: E. enemy, Q. enemy, Boss. The main difference between them is the attack mode.

1.5.2 Waves

Enemies are organised in *waves*.

1.5.3 Enemy AI

Enemy do not “attack” at all, *unit* gets damage when it attacks an *enemy*.

1.6 Cards and Coins

1.6.1 Cards

Card is an important resource. Player can draw cards periodically if there are any *units* on *base*. The frequency is depends on the sum of *skill* for all *units* on the *base*.

The *card* has types corresponding to *units*. Player can use *card* in following ways:

- By discarding 2 cards of the same type, player can spawn a corresponding *unit* on a *cell* (subject to spawning restrictions).
- By discarding 3 cards of the same type, player can update the corresponding *unit factory*, and then player can spawn more powerful *units*.
- By discarding 3 cards of the same type, player can perform *Mancala*.

There exists a *hand limit*. Once the limit is reached, the game will discard the earliest drawn card automatically.

1.6.1.1 Coins *Coins* are acquired by killing *enemies*. *Coin* can be consumed in following ways:

- Upgrade **one** attribute of a *unit*.
- Extend *hand limit*.

2 Game Strategy

2.1 Overview

Mancala Defence is a game requires strategy, especially dynamic planning. Players need to carefully plan their resource usage and unit placement. Also, players are expected to keep the balance to achieve success in this game.

2.2 Resource Management

2.3 Cards and Coins

Unlike common defence games, players have **two kinds** of resources in *Mancala Defence*: *cards* and *coins*.

Cards are the **major resource** in this game. They are randomly drawn, and **requires time to produce**. Player can consume cards to do main operations: Unit spawn, Mancala and upgrade units.

Coins are the minor resource in this game. They can do small or emergency "fixes" to the current situation, such as upgrade one unit or extend the hand limit. They are gained by **killing enemies**.

The two kinds of resources have some common purposes. For example, player can upgrade units either by cards or coins(one by one).

2.4 Units

Units are also resources, with they spawned onto cells. Usually, units serve the role of attacking enemies. However, they can also moved to *base*, where they can increase the speed of drawing cards but cannot attack enemies.

2.4.1 Short-term Planning and Long-term Planning

The choice between tackling current situation and use the resources to invest in long-term benefits exists everywhere in the game. Player are expected to keep the balance. If they unable to do so, they will face the following situations:

- Too many resource are used for long-term benefits. Player **cannot tackle the current situation**.
- Too many resource are used for short-term benefits. Player **cannot solve the future waves**.

The effects of five options can be divided into short-term effects and long-term effects:

Table 2: The effects of five options

Short-term	Long-term
Spawn unit	Upgrade unit
Fix unit	Add hand limit

Short-term	Long-term
	Mancala

Notice that the long-term effect options can also **tackle current situations** sometimes.

2.5 Cell Efficiency

Units on cell can only cover a limited, circle range. And obviously, the enemy paths inside the ranges of cells are not of equal length. Thus, cell efficiency is also need to be considered.

Cells located **inside the corner** of enemy path and **adjacent to two** (or more) paths is considered efficient.

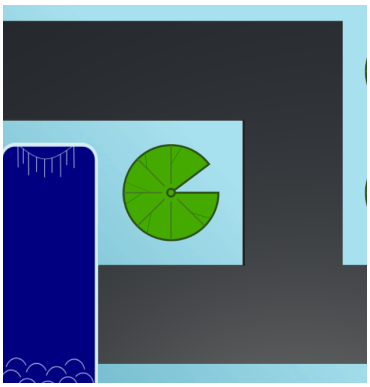


Figure 5: Cell of good efficiency

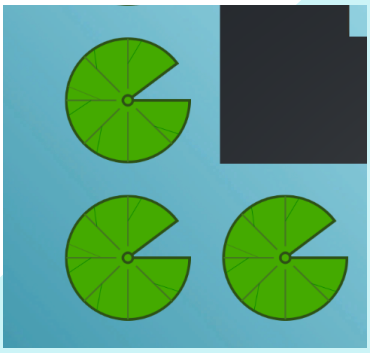


Figure 6: Cell of bad efficiency

2.6 Mancala

Once deployed, the unit is waiting for be destroyed, or change its place by Mancala. Mancala plays the following roles in the game:

- The **only way** to place units on base.
- *Activate* the cell for direct spawning later on.
- Distribute units to avoid E. enemies form attacking the units on the whole cell.

3 UI Design

3.1 Overview

The theme of UI is pond scene. The elements containing lotus leaves, frogs, pearls and carps. All created by ourselves.

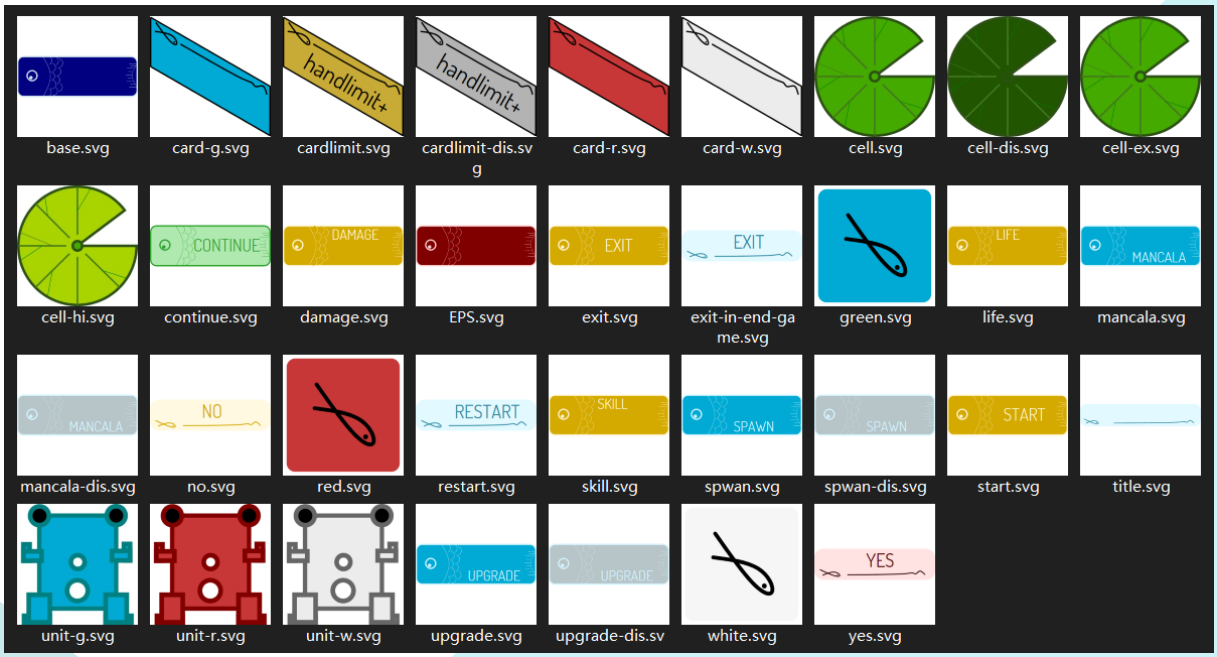


Figure 7: UI Elements

3.2 Start Scene

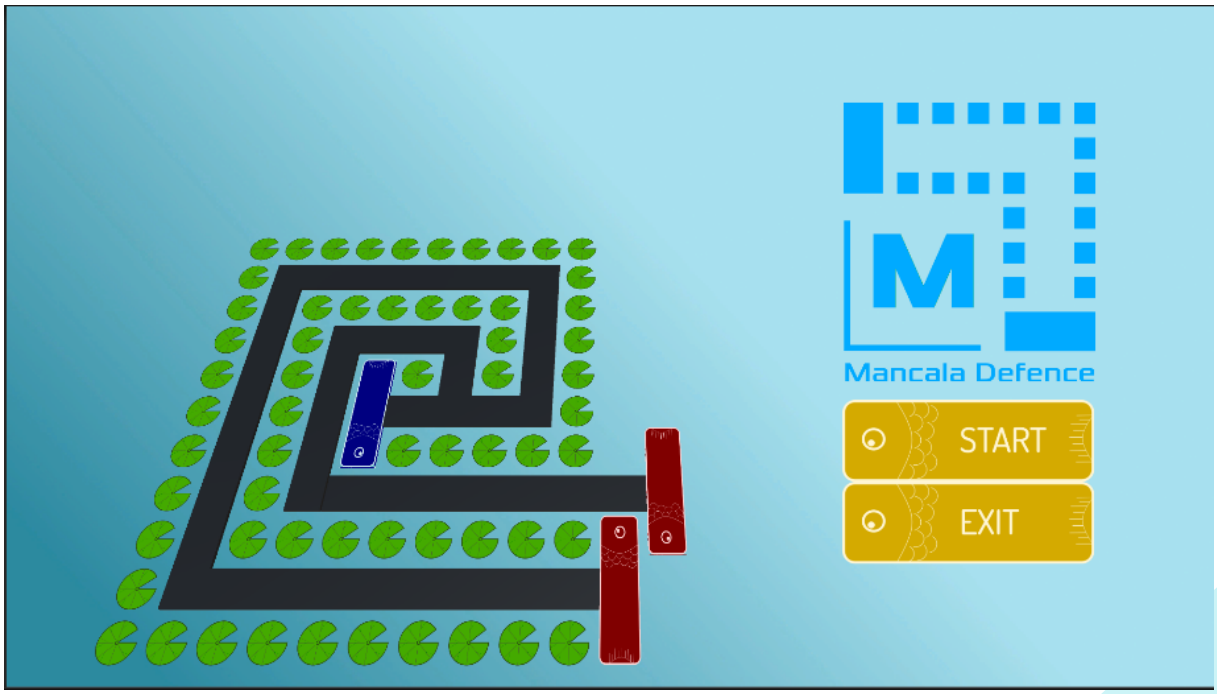


Figure 8: Start Scene

1. Have two buttons to start game or exit.
2. Show the second map on the left.
3. Have the logo of game.

3.3 Select Stage

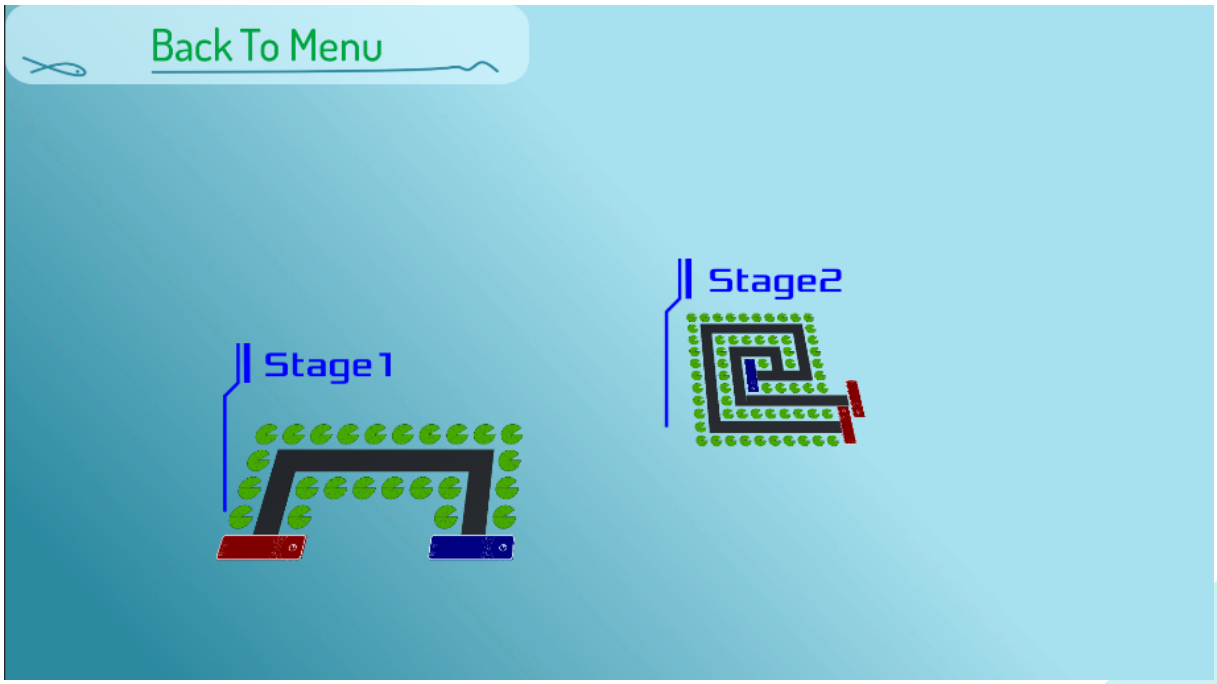


Figure 9: Select Stage

1. Have the back button in the upper left corner.
2. Show all the stage maps. When click any of them, will change scene to corresponding stage.

3.4 Game Scene

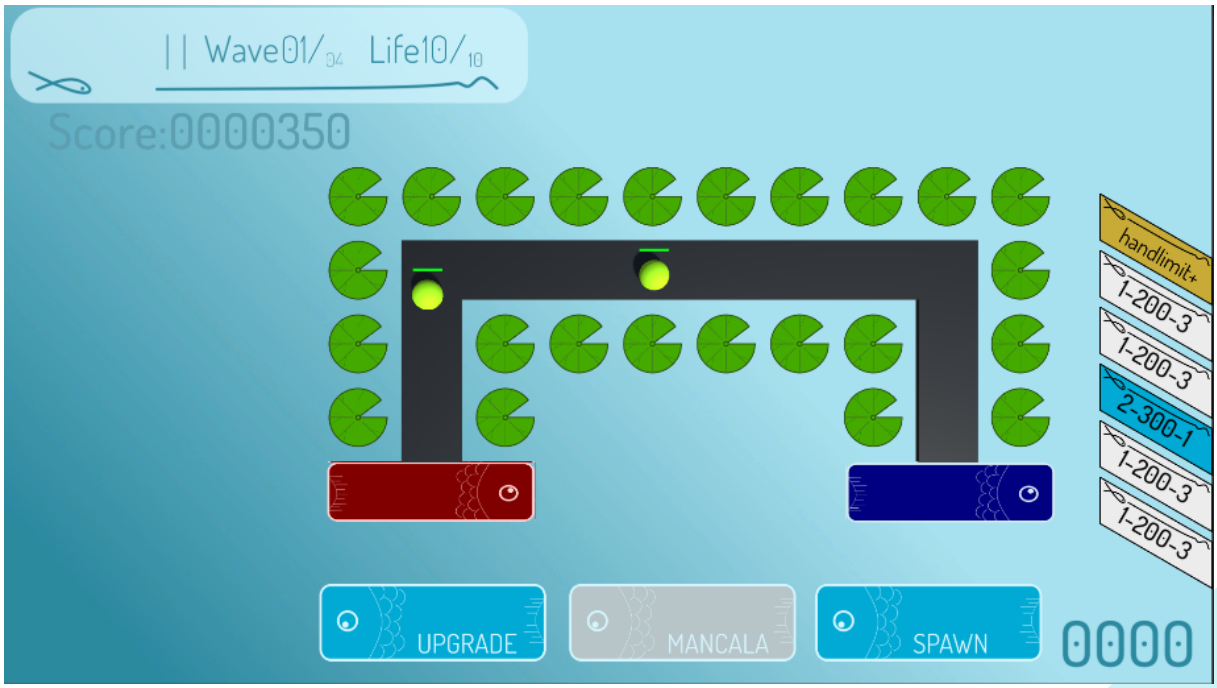


Figure 10: Game Scene

1. Pause button, wave text and life text are in the upper left corner.
2. Coin text is in the lower right corner.
3. Cards and extend handlimit button are on the right.
4. Player operations buttons are below.
5. Hint message is in the upper right corner. Only when player choose wrong cell or don't have enough resource, hint message will show.

3.5 Pause UI

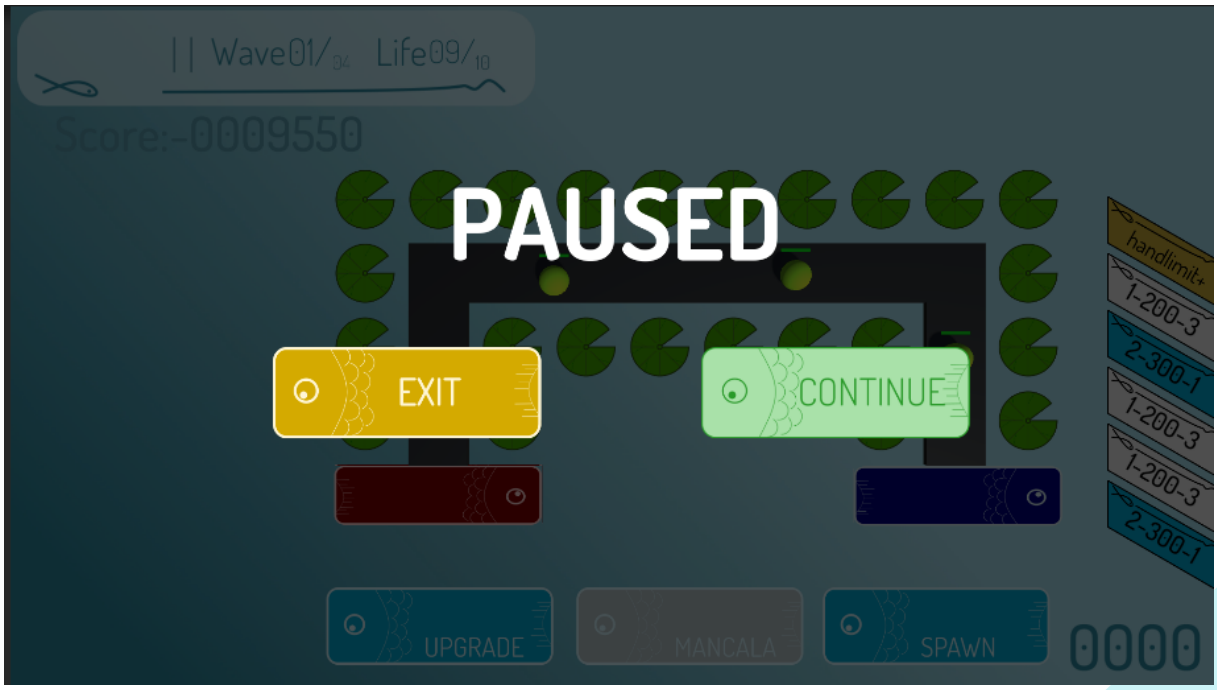
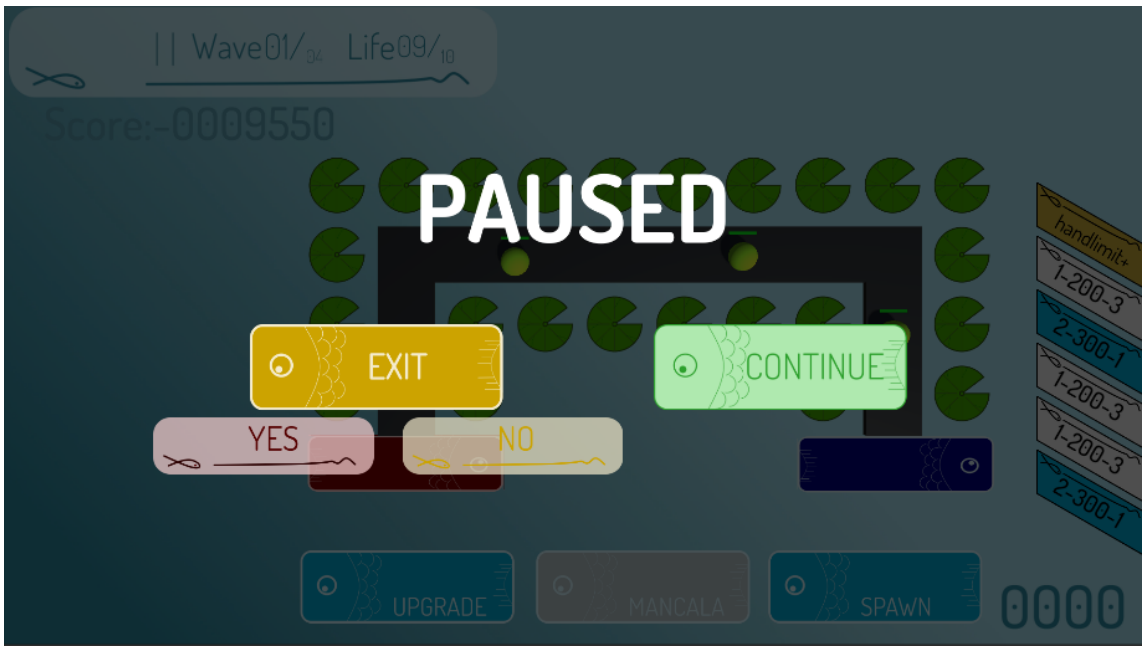


Figure 11: Pause UI

1. Have a text to show game is paused.
2. Use a translucent background to cover the game scene.
3. After click continue button, game will continue and pause UI will hide.
4. After click exit button, there will be another two buttons to check click.



5. After click yes button, player will get back to the start scene.
6. After click no button, the two buttons will hide.

3.6 End UI

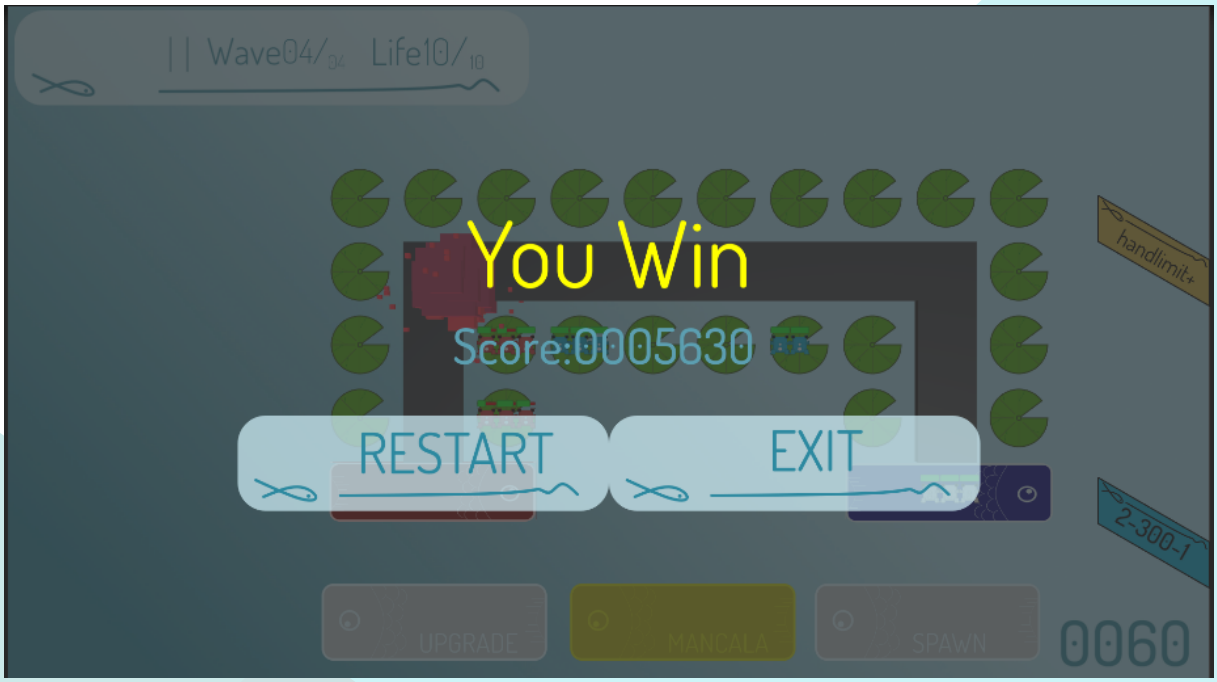


Figure 12: End UI

1. Have a text to show game win or lose.
2. In the center of screen have a text to show player's final score.
3. After click restart button, player will restart present stage.
4. After click exit button, player will get back to the start scene.

3.7 Upgrade UI

Figure 13: Upgrade UI

1. Have three buttons to show different attributes.
2. Click any of it will upgrade corresponding unit attribute when have enough coins, or end upgrading and show hint message.
3. If unit dead, upgrading will be forced to stop.

4 UI Implements

4.1 Map

Use prefab cell, basecell and enemy point form the fundamental map.

4.2 Cell

4.2.1 Choose Cell

1. Use flag `IsHighLighting` to check.
2. When it's time to call cell choosing function, first high light all cells that are valid for unit spawn or mancala.
3. In update function, use raycast to get the cell chosen until cancelled.
4. Call `PlayerChooseCellCallback()` to continue player operation and end highlight.

4.2.2 Cell Material Change

1. Use renderer and materials set in prefab.
2. When highlighting the cells or the cells are exhausted, use render to change their material.
3. When highlight end, check `IsExhausted` to determine material is normal or exhausted.

4.2.3 Highlight Cell

1. When choose cell, use a list to store all the highlighted cells, and change their material.
2. When highlight end, use `HighLightCellsEnd()` function to change all the changed material to previous material.

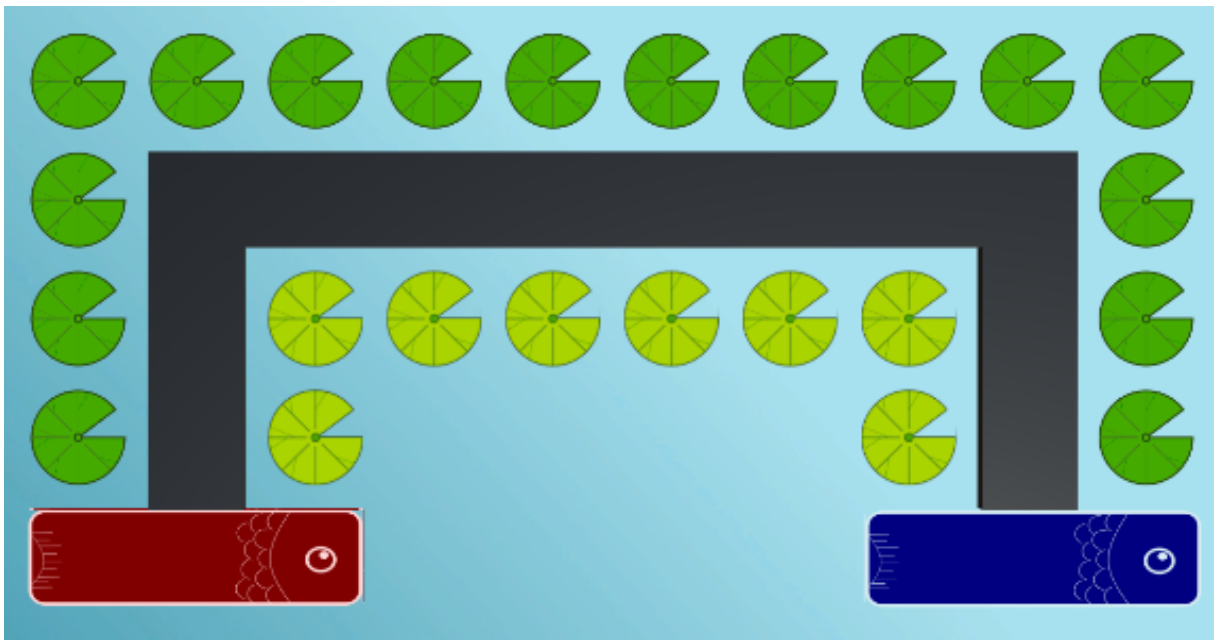


Figure 14: Cell Highlighted

4.3 Unit Factory

4.3.1 Generate Unit

1. Use three units in gameobject `UnitFactory` that are white, green and red.
2. The three units are not active so won't be seen in camera.
3. When call the function, return the unit same as parameter type.

4.4 Unit

4.4.1 Unit Spawn

1. Call `Instantiate()` function to get a new unit same as the type in unit factory.
2. Copy attributes and set the new unit active. Each unit's position is set different, so the units on the same cell will not overlap.
3. Use `presetPosition` and cell unit count to make sure they have different position.

4.4.2 Unit move

1. When player perform mancala, units on the chosen cell will be moved to a new cell.

2. Change their parent and set new position.

Figure 15: Unit move - Before

Figure 16: Unit move - After

4.4.3 Unit Choose

1. When it's not choosing cell or choosing type, check if player click left mouse button.

2. Use raycast to get if a unit is chosen. If true,then show the unit upgrade canvas and set canvas position.

3. The attributes of the unit will be shown on the canvas.

Every choose cell operation will stop unit choose.

4.4.4 Unit Upgrade

Figure 17: Upgrade UI

1. When a unit is chosen, use three buttons to control attribute upgrade.
2. Add a listener for each button onclick. When player click one button, call `TryUpgrade` function.
3. In the function, check if coins are enough for upgrading, then upgrade the chosen attribute.

4.4.5 Lifebar

1. Use a slider.
2. When unit life point reduce, reset slider value so it will reduce like lifebar.

4.5 Card and Card Factory

4.5.1 Draw Card

1. Same as unit factory, `Instantiate()` a new card from card factory.

2. Set position, rotation, active and parent. Return the new card.

4.5.2 Use Card

Destroy the game object and reset other card position.

4.5.3 Reset Card Position

Get all the card remained, then set their position in order.

4.6 PlayerAssets

4.6.1 Cards Image

`CardsImage` is just a visualization for `Cards` in PlayerAssets. Each time `Cards` change, change `CardsImage`(Draw Card, Discard or Use Card).

4.7 Player Interface

4.7.1 Hint Message

Figure 18: One of the Hint Messages

1. Simply use a text.

2. When need to show the message, set it's text and show it, then call `Invoke()` function to hide it in 2 seconds.

3. `Cancelinvoke()` at first to prevent new message being hidden too fast.

4.7.2 Choose Type

Figure 19: Choose One from the Two colors

1. Show the valid types and unit message.

2. When player click certain button, call `PlayerCardTypeCallback()` to continue choose cell or finish upgrade unit.

4.8 GameManager

4.8.1 Player Option

1. Offer four buttons: Unit Spawn, Mancala, Unit Upgrade and Extend Hand Limit. Also add a listener for button `onclick`.

2. When player click the button, call `PlayerCardOption()` or check coins for extend hand limit.

3. In update function, check if the operations are available. If so, change color to white, or change color to gray.

Specially, when free mancala is available, mancala button color will change.

4.8.2 Pause Button

1. Call `SetGamePause()` function to change if pause.

2. When pause, show pause UI which has two buttons. Player can use them to back to menu or continue game.

4.8.3 Life, Coin and Wave

Figure 20: Wave and Life

Figure 21: Coin

1. Use three texts to show life and wave. When player life point reduce, coin gain, coin cost or wave change, change the text.

2. Use `ToString()` to get text and format.

4.9 Camera

4.9.1 Camera Move

1. When choosing a unit, camera will move smoothly to get near to the unit chosen.
2. Use optimized `Leap()` function to make move speed unchanged.
3. When choosing operation end, move camera back to position before. Also smoothly.
4. Use a bool variable to make this move only valid when need.

4.9.2 Camera Control

1. Player can use keyboard to control camera move.

2. Use `Input.GetAxis()` to get input, this allows player use 'W', 'A', 'S', 'D' or "Up", "Down", "Left", "Right" to control move. Also, player can use mouse scroll wheel to zoom in or zoom out.
3. Use `CheckPosition()` to make sure player can't move out of game scene. When camera is on boundary, it will not be move further.
4. Player can simply press space key to get back to original position. This position is fixed for each scene.

4.10 Start Menu

1. Have two buttons to start game or exit.
2. Use `Application.Quit()` to exit game.
3. When start button is pressed, camera will also move smoothly to show all the stages.
4. Set the two map different position, so at first there will only show the second map. Only when start button is pressed, player will see the the first map.

5 Stage Design

5.1 Wave Information

Table 3: Wave Information for Stage 1

Wave	Enemy Type	#Enemy	Total Damage
1	C. Enemy	5	30
2	E. Enemy	2	70
3	Q. Enemy	1	110
4	Boss	1	125

Table 4: Wave Information for Stage 2

Wave	Enemy Type (path1 and path2)	#Enemy	Total Damage
1	C. Enemy+C. Enemy	2+2	24
2	C. Enemy+C. Enemy	5+3	48

Wave	Enemy Type (path1 and path2)	#Enemy	Total Damage
3	E. Enemy+C. Enemy	2+5	100
4	C. Enemy+E. Enemy	20+5	295
5	Q. Enemy+E. Enemy	2+5	395
6	Boss + Boss	1+2	375
7	Q. Enemy+Boss	5+4	1050
8	Boss +Q. Enemy	30+80	12550

5.2 The Favourite Cells and Vulnerable Cells of Map

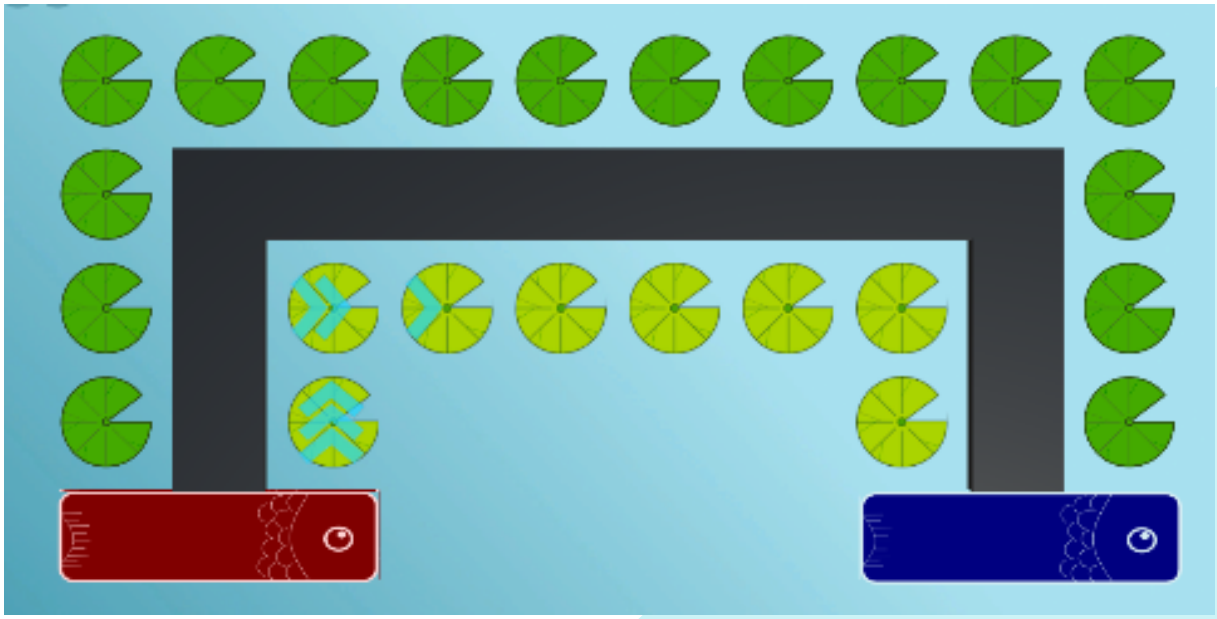


Figure 22: Favourite cells and vulnerable cells in stage 1

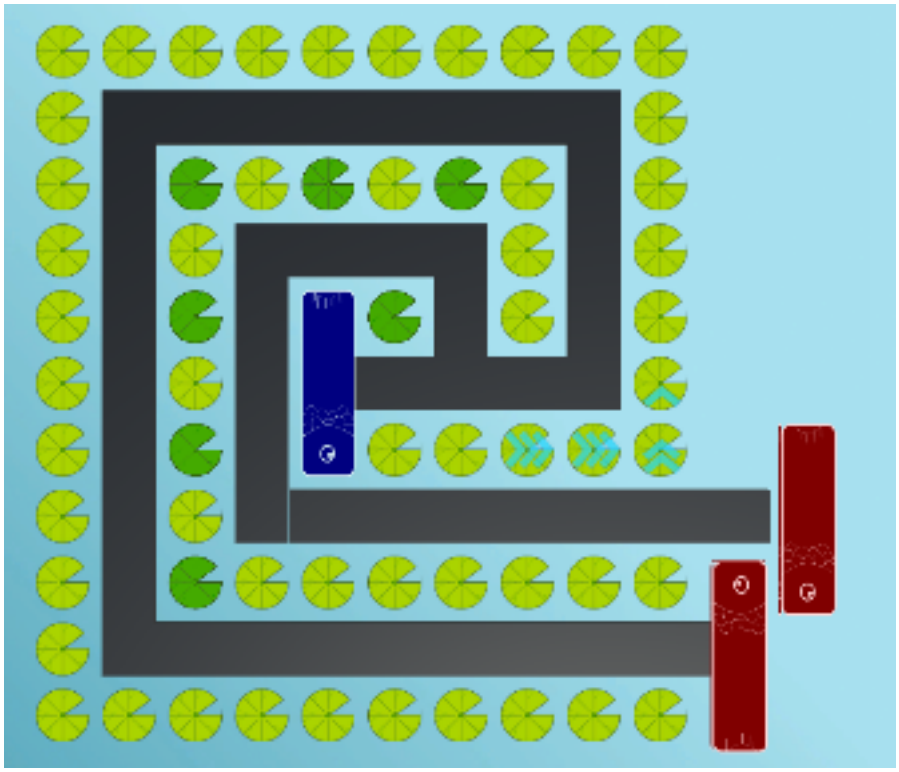


Figure 23: Favourite cells and vulnerable cells in stage 2

The yellow part is the Favourite Cells where the units can be placed in the beginning, The green part is the Vulnerable Cells where the units cannot be placed in the beginning.

5.3 Opening setting

At the beginning of the game, players will **randomly draw 3 cards**.

6 Game Mechanism Implement

6.1 Enemy System

6.1.1 Enemy Factory

Use Coroutine to spawn enemies, the enemy is generated by waves, and the wave information is stored in a separate data structure. Wave information includes the type, number and generation interval of enemies in each wave, as well as the interval between waves. The design of specific waves is related to

the numerical system. When all enemies are generated and the number of surviving enemies is 0, the player will win.

6.1.1.1 Enemy attribute

1. Life: Enemy's life value when life<=0, enemy will be destroyed.
2. Speed: Describe the enemy's move speed.

6.1.1.2 Enemy move

1. Way points are stored in a list, Enemy will move to the current waypoint, if they arrive the current waypoint, then they will move to next waypoint, until they arrive the player base.
2. when enemy arrive the player base, the player life will cut back, when player life<=0, the game over.

6.1.1.3 Enemy attack

1. When C. enemy get damage, it will Rebound damage to the unit which attacks it.
2. When E. enemy get damage, it will Rebound damage to the units which are on the same cell.

6.1.2 Wave and Waypoint

Waves is a list to store wave information. Waypoint is a list to store waypoint information.

6.2 Map

Use unity cube to build Rood, and create some waypoint that lead enemy to move.

6.2.1 Map Design

1. Stage 1 is a simple teaching level, showing the type of enemy, and players can also be familiar with the game operation here.

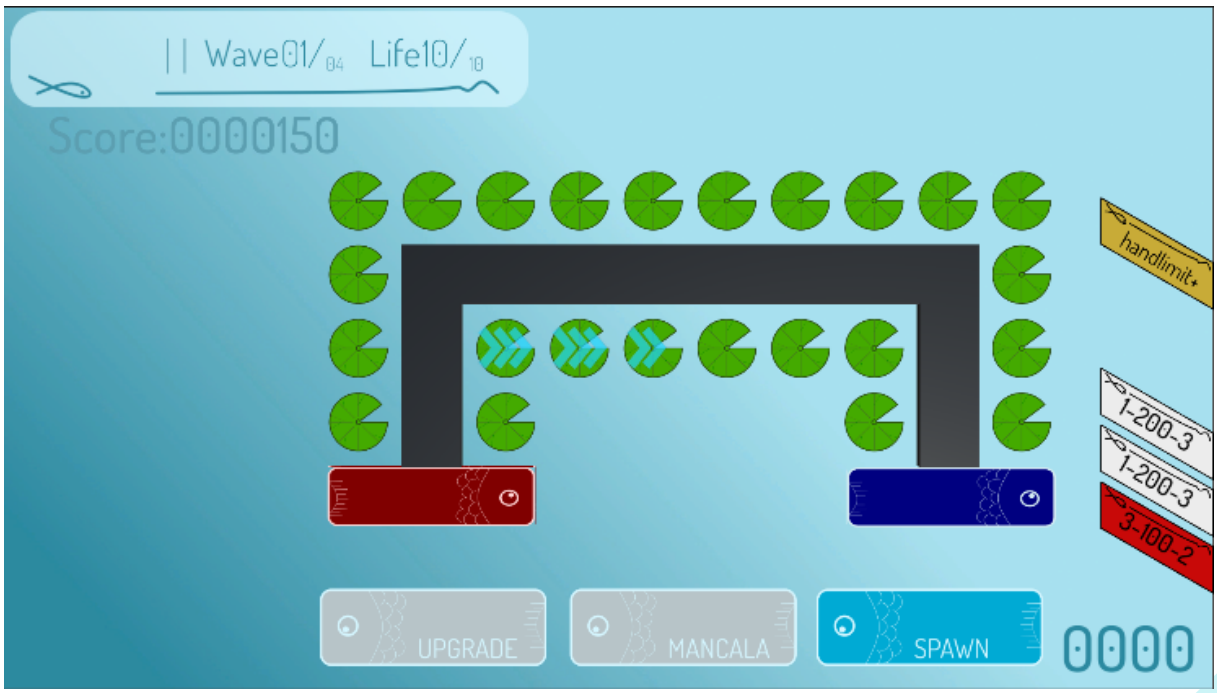


Figure 24: Map of stage 1

2. Stage 2 is a difficult level. As our map needs to form a loop, we are inspired by other games. We decided to use the second level map design into a spiral shape, with the design of two enemy factories, to enhance the fun and challenge of the game.

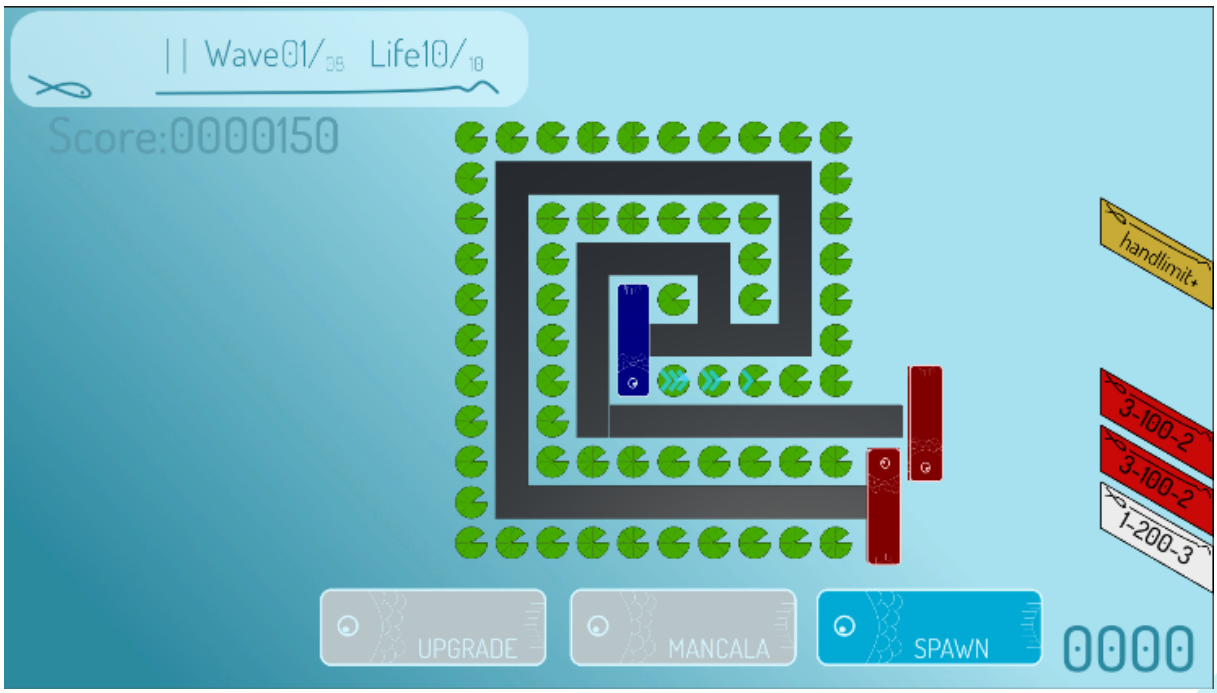


Figure 25: Map of stage 2

6.3 Unit

6.3.1 Unit attack

1. Create a empty object with a 3d collider to get collision detection with enemy.
2. Create a empty object as the firing point of the bullet.
3. When enemy get in the collider the enemy will be stored in a list,when they get out or die, they will be removed from the list.
4. Unit will attack the enemy in the list, it will Instantiate a bullet to attack.

6.3.2 Unit upgrade

When unit upgrade, their HP will increase by 10 points.

6.4 Bullet

1.When bullet created, it will set its target and damage value. 2.Bullet move to the target with a constant speed. 3.After bullet collided with the enemy, enemy will receive the damage and bullet will

be deleted.

6.5 Card System

When the game starts, the player will draw three cards as the starting card.

6.5.1 Draw Card Speed

The speed of drawing cards is related to the number of units on the player base. The formula is:

$$t = \frac{5}{1 + \sqrt{n}}$$

where n is the number of units on the player base, t is the time it takes a player to draw a card.

6.5.2 Numerical system

Because in this game, the range of the player's units is the same, so the rough definition of the four enemy damage can be as follows:

$$\text{damage} = (0.2 + \text{speed}) \cdot \text{life} \cdot k$$

where k is 0.5 for C. Enemy, and 1 for E. Enemy, Q. enemy and Boss.

Table 5: Enemy attributes according to type

Enemy	Life	Speed	Damage
C. enemy	10	1	6
E. enemy	50	0.5	35
Q. enemy	50	2	110
Boss	250	0.3	125

Then we calculate the wave intensity value according to the intensity value, so as to design the number of enemies in each attack wave, and make the wave intensity value roughly conform to a function curve.

7 List of Terms

Term	Meaning
activation	(see Special Rules for Spawning Units on Cells)
base/basecell	A special *favourite cell* that player should prevent *enemies* from.
building activity	(see Unit Attributes and the Building Activity)
card	A major resource that player can use to spawn *units*, etc..
cell	Where player can place units.
coin	A minor resource that player can use to upgrade *units*, etc..
enemy spawning point (*ESP*)	Place where *enemies* are spawned.
enemy	Entity that keeps approaching the *base*. Player will lose when one gets into the *base*.
exhaustion	A state of *cell*. All the *units* on it are all disabled on an *exhausted cell*.
favourite cell (*fav. cell*)	*Cell* where *units* stay unaffected when placing on it.
hand limit	How many *cards* player can hold.
Mancala	A special move redistributing *units*.
map	Collection of *enemy spawning point*, *base*, *path* and *cells*.
overloading	A state of *cell*, entered when too many *units* are placed on it. It may become *exhausted* afterwards.
skill	The amount that a *unit* can contribute to *building activity* at a time.
unit factory	(see Cards)
unit	Object that will automatically damaging the *enemies* nearby once placed.
vulnerable cell (*vul. cell*)	*Cell* where *units* are debuffed when placing on it.

8 Contributors

1. YAN Zehao (Project Manager)
    1. Game mechanism design

    2. Basic game logic implement
    3. UI element creation
    4. Document parts related to above
2. XIU Lei
    1. Stage design
    2. Game mechanism implement
    3. Document parts related to above
3. XIAO Yang
    1. UI design
    2. UI implement
    3. Document parts related to above