

# User Interaction Technical Final Project

# ——Unity3D Tower Defense Game

1953729 吴浩泽

任 课 教 师：邓 浩

2023年5月

**catalogue**

[**1. Game overview** 1](#_Toc134464075)

[**1.1 Background** 2](#_Toc134464076)

[**1.2 game introduction** 2](#_Toc134464077)

[**2. Game scene design** 2](#_Toc134464078)

[**2.1 start scene design** 2](#_Toc134464079)

[**2.2 map layout design** 3](#_Toc134464080)

[**2.3 enemy role design** 3](#_Toc134464081)

[**2.4 Fort design** 4](#_Toc134464082)

[**2.4.1 turret control script** 4](#_Toc134464083)

[**2.4.2 bullet control script** 4](#_Toc134464083)

[**2.5 design of emplacement** 5](#_Toc134464084)

[**2.6 end interface design** 5](#_Toc134464085)

[**3. Introduction to game operation** 5](#_Toc134464086)

[**3.1 start the game** 5](#_Toc134464087)

[**3.2 playing games** 6](#_Toc134464088)

[**3.2.1 money** 6](#_Toc134464089)

[**3.2.2 emplacement** 7](#_Toc134464090)

[**3.2.3 Fort upgrade and demolition** 7](#_Toc134464091)

[**3.2.4 Exit** 8](#_Toc134464092)

[**3.3 end of game** 8](#_Toc134464093)

[**4. User interaction design** 9](#_Toc134464094)

[**4.1 player perspective movement** 9](#_Toc134464095)

[**4.2 free emplacement of turrets** 9](#_Toc134464096)

[**4.3 exit control** 9](#_Toc134464097)

[**5. How the game works** 10](#_Toc134464098)

[**6. Project advantages and disadvantages** 10](#_Toc134464100)

[**7. Contribution of team members** 10](#_Toc134464111)

**1. Game overview**

**1.1 Background**

Tower defense game is a strategy game where players aim to prevent enemies from entering specific areas on the game map. Players need to build and upgrade defense towers to resist enemy attacks, and use various strategies and tactics to protect their bases or resources. These games are usually based on a fixed map layout, where enemies move along specific paths, and players need to utilize the special abilities and layout of towers to eliminate enemies to the greatest extent possible.

Players can usually choose different types of defense towers, each with its own unique characteristics and attack methods. For example, some towers can fire bullets to attack enemies, while others can unleash magic or special abilities. Players need to choose appropriate tower types based on the characteristics and weaknesses of their enemies, and continuously upgrade and improve them in the game to adapt to increasingly powerful enemies.

Tower defense games typically have multiple levels or waves, with each wave increasing the size and strength of the enemy. Players need to make decisions before each wave of enemies arrives, including building new towers, upgrading existing towers, or using special abilities to counter enemies. In the process of the game, players may also encounter various obstacles, special enemies, or additional challenges that require flexibility in response.

The success of tower defense games usually depends on the player's strategic decision-making and resource management abilities. Players need to protect their base while utilizing limited resources to build and upgrade towers, and find the best layout and strategies to deal with different types of enemies.

Tower defense games are very popular in the gaming industry, and there are already many classic Tower defense game series, as well as many different styles and themes of tower defense games to choose from.

**1.2 game introduction**

This project is about the design of a 3D tower defense game. This project is developed using the Unity3D game engine, focusing on user interaction. This project includes two game scenes: the startup scene and the main scene. After starting the game, you can click Start to enter the main scene and open the official game, or you can click Exit to exit the game directly. When running the game in the main scene, you can also click Exit to exit the game.

After entering the game interface, you can see two bases. The upper left corner is the stronghold that generates enemies, and the lower right corner is the home we need to protect. By constructing a turret to effectively shoot at the enemy, one can block their attack. When all enemies are eliminated, victory is declared. When an enemy enters our home, we declare defeat.

**2. Game scene design**

**2.1 start scene design**

In the starting scene, we use the turret in the game scene as the main object of the interface. To reflect the 3D effect, we will set the camera rotation animation effect in the starting scene, making users feel like they are in a 3D game from the beginning. Not only that, we also provide users with two buttons to start the game and exit, both of which are based on user interaction and automatically sense the position of the mouse. When the mouse is placed on it, it senses and scales the font to prompt the user.

**2.2 map layout design**

In the map layout, we use a plane as the basic scene and create map cubes and road cubes on the plane. Different turrets can be placed on map cubes. Both the start and end cubes are introduced into the model and rendered through pipelines.

**2.3 enemy role design**

The enemy character mainly mounts two scripts, namely the enemy control script and the enemy data script. The enemy control script is mainly responsible for managing the action status of enemy characters, including death. Enemy data scripts are mainly responsible for managing enemy character data.

**2.3.1 enemy control script**

|  |  |
| --- | --- |
| Variable (class) | describe |
| EnemyStates{Guard，Chase，Dead} | Enemy state enumeration class, including the enemy character in the game process may appear in three states |
| animation | Enemy character selector, control animation effects |
| SwitchAnimation（） | The enemy animation selection function adjusts the animation according to the current state of the enemy character |
| SwitchStates（） | The enemy state selection function, according to the trigger conditions, switches the status of the enemy role. The state value is enemystates enumeration class |
| Attack（） | Attack function, attack the target and cause damage according to the numerical value |
| Hit（） | Hit function, triggered by player character, deals damage to enemy character |
| TargetInSkillRange（） | Judge whether the attack object is in the attack range and switch the state |
| FoundPlayer（） | Look for player characters.In the process of going to the destination, if the player character is found in the attack range, he will stop going to the destination and fight with the player character |
| Shot（） | When the enemy character receives the fort design, it will cause damage to the enemy according to the battery data and enemy data |

**2.3.2 CharacterData**

|  |  |
| --- | --- |
| variable | describe |
| UpdateHealthBarOnAttack | Update enemy character's health value (event trigger function) |
| characterData | Basic values of enemy players (including health, defense, block, etc.) |
| attackDta | Attack value class (including attack power, attack frequency, critical hit rate, etc.) |
| TakeDamage（） | Damage function, through the numerical value, calculate the damage to the enemy and update the health value |

**2.4 Fort design**

There are three types of fortresses: laser fortresses, machine gun fortresses, and missile fortresses. The turret scripts for these three types of turrets are the same. The attack methods of machine gun fortresses and missile fortresses are similar, so only the bullets they fire need to be changed. Laser fortresses are different from the other two. In the script, use the bool variable to distinguish specific attack functions.

**2.4.1 turret control script**

|  |  |
| --- | --- |
| variable | describe |
| bulletPrefab | Control the type of bullets fired by the fort |
| firePosition | The firing position from which a bullet or laser is fired |
| useLaser | Is laser used |
| attackRate | Attack rate |
| damageRate | Laser damage rate |
| function |  |
| OnTriggerEnter()/OnTriggerExit() | Monitor whether the monster is in attack range |
| Attack() | Attack function |
| UpdateEnermy() | Update the list of enemies in range |

**2.4.2 bullet control script**

|  |  |
| --- | --- |
| variable | describe |
| damage | Amount of damage done by bullets |
| speed | The speed at which a bullet flies |
| distanceArriveTarget | Determine the distance of collision with target |
| target | Monster target |
| Attack() | Track the monster and call the monster's blood deduction function after impact |
| Die() | After hitting the monster, the bullet explodes to destroy it |

**2.5 design of emplacement**

In the game interface, players select the fortress to place by clicking the mouse. After clicking, the background color of the selected fortress changes, indicating that the fortress has been selected. Then, move the mouse over the white square on the map. When the square turns red, it indicates that the square is selected. If the money is enough, click on the box to place the fortress.

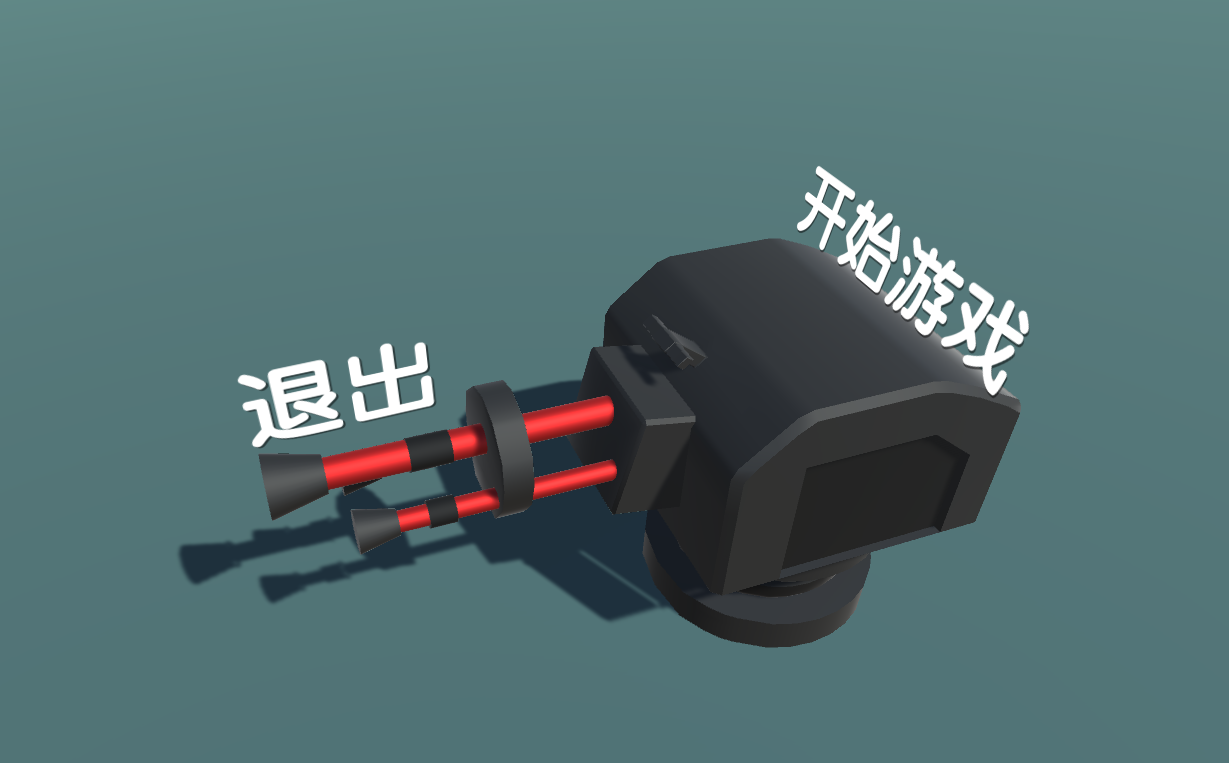
**2.6 end interface design**

In the ending interface, we consider that users can still experience the scene in the game (regardless of whether it is successful or not) at the end of the game, so we set the ending interface to semi transparent and display the success or failure of the previous game. In addition, we have set up replay and menu buttons, which are designed for user interaction.

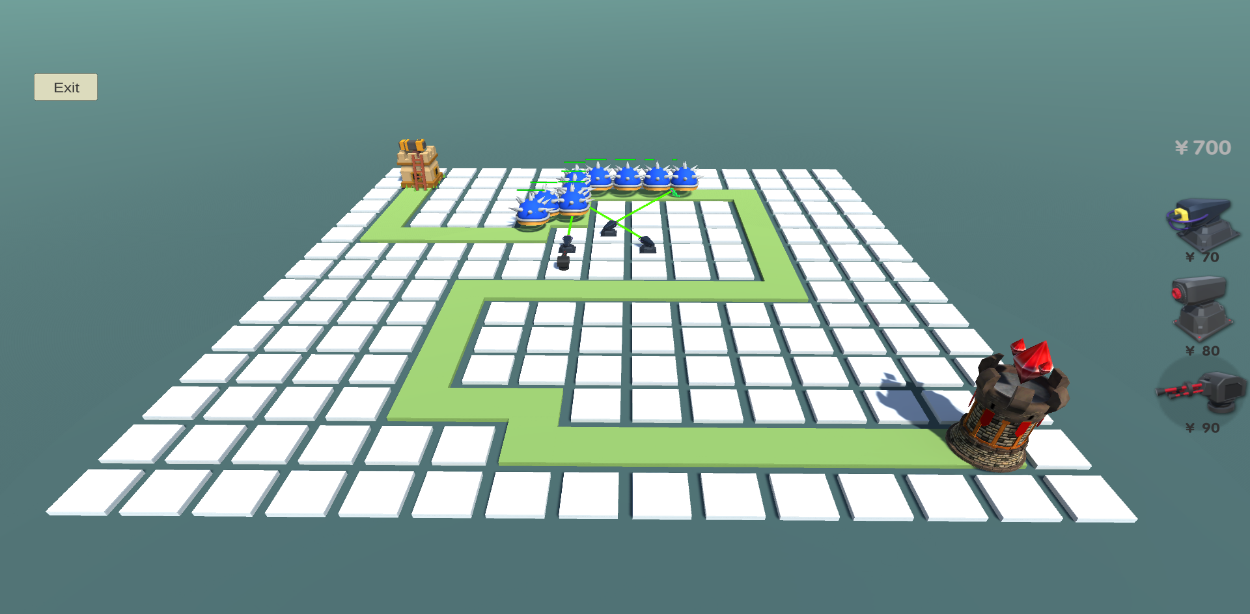
**3. Introduction to game operation**

**3.1 start the game**

After entering the game, the user will see the game initialization interface, which includes two options: "Start Game" and "Exit". Click 'Start Game' to enter the game scene; click 'Exit' to exit the game directly.



**3.2 playing games**

After entering the game interface, you can see two bases. The upper left corner is the stronghold that generates enemies, and the lower right corner is the home we need to protect. By constructing a turret to effectively shoot at the enemy, one can block their attack. When all enemies are eliminated, victory is declared. When an enemy enters our home, we declare defeat.

**3.2.1 money**

Players can see the amount of funds in the right sidebar of the game interface. The funds are used to build and upgrade fortresses.

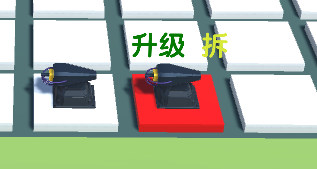


**3.2.2 emplacement**

The construction of fortresses can only be carried out on high platforms (white squares). After selecting the fortress, players can choose the high platform (white block) to place, and the selected position will turn red. If the player's current amount is greater than or equal to the amount required for the fortress, they can click the left mouse button to place the fortress.

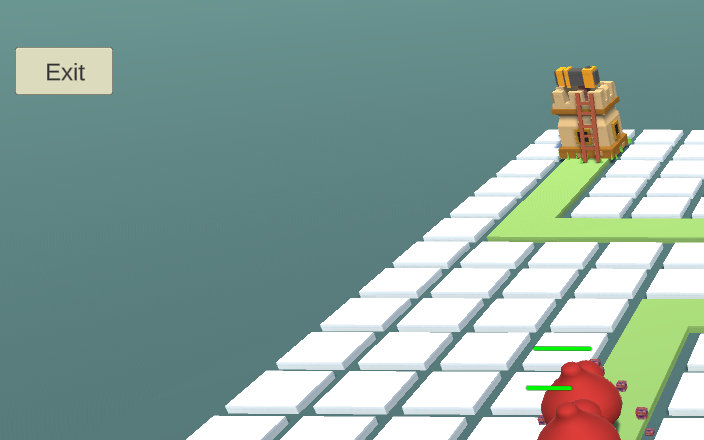


**3.2.3 Fort upgrade and demolition**

Players can upgrade or dismantle turrets that have already been created. Click to select the fortress, and you can see the upgrade and remove buttons. Click on the upgrade button. If the current amount is greater than the amount required for the upgrade, the fortress upgrade can be completed. Click 'Remove' to remove the built fortress and return some funds.

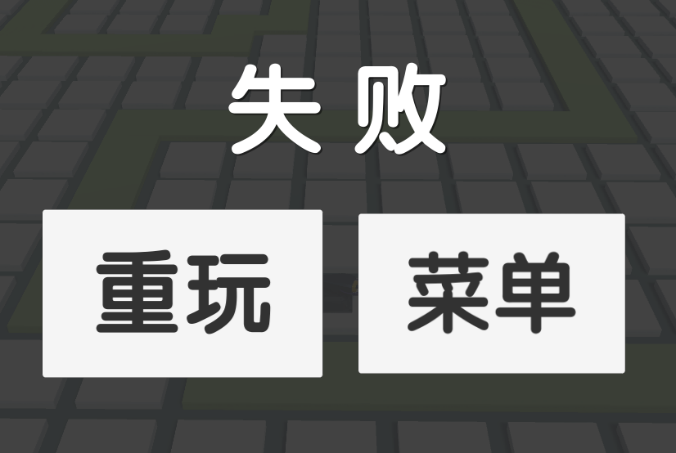
**3.2.4 Exit**

Users can see the "Exit" button in the upper left corner of the game interface. Click 'Exit' and users can exit the game.



**3.3 end of game**

Five waves of enemies have been designed in the current level. If all enemies are defeated, the game will win; If any enemy enters the home, the game fails.



**4. User interaction design**

**4.1 player perspective movement**

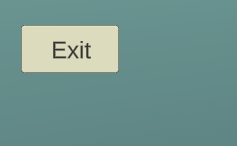
Users can use WSAD and the up, down, left, and right arrow keys to control the movement of the perspective, achieving a 3D effect. Users can also zoom in and out through the mouse's scroll wheel.

**4.2 free emplacement of turrets**

In the game interface, players select the fortress to place by clicking the mouse. After clicking, the background color of the selected fortress changes, indicating that the fortress has been selected. Then, move the mouse over the square on the map. When the square turns red, it indicates that the square has been selected. If there is enough money, click on the square to place the fortress.

**4.3 exit control**

You can end the entire game by pressing the Exit button in the middle of the game. Players can freely control the game's switches without having to end the game because they have other things to do.



1. **How the game works**

To run the game, decompress the source code file, find the out folder in the source file, and double-click Tower Defense Project Unity5.5.exe, you can run the game.



1. **Project advantages and disadvantages**

Advantages:

1. The project has more interaction with users, improving players' sense of participation and the fun of the game.

2. The project has a certain level of difficulty, which can easily stimulate players' competitiveness and allow them to continuously train the most effective methods to achieve victory.

3. The project model is more refined, which can improve user satisfaction with the game.

4. The design of the project interface is simple and generous, mainly highlighting the characteristics of the game and not excessively interfering with players.

5. There are many scripts in the project, and many game effects are integrated in the project, which is very helpful for improving the user experience.

Disadvantages:

1. The project does not implement the player save function and cannot save different game progress for different players.

2. The project has not yet completed the selection of multi-level maps, and the complexity needs to be improved.

3. The project did not achieve more turrets, more monster construction, and was not yet flashy enough.

**7. Contribution of team members**

The contribution ratio of group members is the same, and the score is equal.