Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 6, No. 1, 2023

Tower Defense Game Design based on Unity3D

Shuohan Chen

Guangdong University of Technology, Guangzhou, 510000, China

Abstract: In today's society, with the rapid development of computer technology, personal computers have been integrated into our lives, and computer games have become a way of entertainment for people. Among them, tower defense games, as a branch of strategy games, have also been loved by the majority of players. On the computer game download platform, such as Steam's download list, you can also see all kinds of excellent tower defense games listed, such as “Bloom TD” and “Plants vs. Zombies”. However, in recent years, most tower defense games lack innovation, and the level game play is monotonous. Players are beginning to get tired of the gameplay of classic tower defense games, and their popularity has begun to decline. In order to solve this problem, this article is based on the Unity3D game engine, combined with the characteristics of UGC and RTS games, and made some innovations to the classic tower defense games. According to the actual needs, a tower defense game called “The Rise of The Tribes” was developed. The main work of this paper is to analyze and design The game, realize The switching between The three scenes in The game and each interface, and solve how to build buildings, how to defend defensive buildings, and how to deploy and move soldiers and attack, how to generate, collect and consume resources, completed The debugging and testing of The game, summarized the development process and made an outlook on how to improve the game.

Keywords: Unity3D; Tower Defense Game; Game Scene; Resource.

1. Introduction

In 1946, the computer was invented at the University of Pennsylvania. In the early days, computers were not used by ordinary people, but by the military and scientists. In 1973, the first personal computer was born in a laboratory in France, followed by the personal computer mouse was invented. After the personal computer was invented, computer games were created. The computer game needs the player to use the computer to do the platform, through the operation mouse and the keyboard, realizes the human-computer interaction, finally achieves on the computer plays the game the entertainment way.

Computer games are not an accident of the moment, but an inevitable result of the development of science and technology [1]. Regardless of the game carrier, computer games are not that different from traditional games in nature. Both are played in specific environments, cooperating or competing with other players according to specific rules of the game, the ultimate winner-take-all entertainment [2].

Tower defense games don't have a long history among all types of games. In 1990, an arcade game, fortress, was born. To most tower defense players, this game is considered the originator of tower defense games. However, for the game fortress, it was very different from modern tower defense games. It does not require the player to expend resources to build the turret, nor does it compensate the player for resources after he has destroyed the enemy. However, from another perspective, the game had finally eliminated the action or shooting elements of similar games in the past, allowing players to set up their own turrets, guide the cannonballs to land, and build castles, the formation of the strategy-centered play, no longer like those games in the past, can only be the shooting core and defensive play combined.

Now, in the popular mind, tower defense games as we know them tend to follow this logic: the player uses the resources in his or her hands to produce a relatively fixed defensive unit → the defensive unit kills the enemy offensive unit to obtain resources → the enemy attacks as the level increases → the player kills the resources obtained to re-invest in the production of the defensive unit to enhance the strength of the enemy units to destroy. In other words, to play a tower defense game well, players must have a high degree of understanding of the level content and reasonably allocate relatively limited resources to the production of different defense units, in order to eliminate all enemy units in a level and protect one's “Base”.

If the above criteria are strictly applied, the TD gameplay that originated in the map editor of Warcraft III: Reign of Chaos is the true ancestor of tower defense. Warcraft III: Reign of Chaos has a lot to offer future developers.

For example, tower defense games in many people's minds, is how to build a defense tower, the proper use of resources, in a fixed line to stop the enemy attack of the game, as far as possible to minimize the ARPG side of the operation, this is also what makes TD different from many RPG defense maps. This kind of map with strong characteristics is represented by works such as element TD. Players only need to understand the intensity difference and difference between AI enemy attack waves in the process of strategy, and according to this information on the use of resources to make decisions, and at any time to strengthen the defense units or recycling and other adjustments can be.

In tower defense games, which have been popular with most people in recent years, developers often package the game's external performance without improving or innovating on the core set of rules.

Today, tower defense games fall into two categories. The first type of tower defense game is very flat on the map, basically no obstacles, monsters from the beginning, according to the shortest route to the end, therefore, players need to place the fort or other obstacles, change the path of the monster and extend the time of the turret to hit the enemy. Generally speaking, this type of tower defense game has a huge amount of empty space on the entire map. Even the entire map can be built with a very high degree of freedom, however, it was especially painful for players with poor route planning skills or those with OCD. A classic in this category

is GEM TD.

Another type of tower defense game gave up the more complicated design of allowing players to change the map layout by building turrets and other obstacles, thus prolonging the path of the enemy, instead, it fixed the monster's path and the location of the tower, limiting the number of towers so players had to figure out what tower to build at each location to make the game more difficult. One such game is balloon TD.

Based on these two categories, and then give tower defense game into the formation system, the birth of many new ways of tower defense games. For example, the anti-tower defense game alien series, where the players become the attacking side, combined with the card game Royal War, where players attack each other, as well as the integration of UGC + RTS online game “Tribal conflict” and so on.

Tower defense game with its unique style, rich levels, different gameplay, has been loved by many players, with a relatively broad market space [3]. It is of great significance and value to develop a strategy tower defense game for exploring the design and realization of tower defense game and its numerical balance.

2. Introduction to the Technology

2.1. Unity3d Game Engine

Unity is a game development engine that has both a free version for a small number of developers and a paid version for large team companies, combined with its cross-platform capabilities, giving it a very high market share. Especially in the mobile game industry, there are plenty of games developed using Unity, such as the Knights of the spirit and Monument Valley. And Unity isn't just a game engine, it's also seen in other industries. It can be used to create simulations of smart homes, to visualize buildings designed by construction companies, and in many other scenarios. In order to accommodate the various platform features and complex requirements, the Unity version updates are also very timely, the latest and coolest features will be the first to hand to the developers.

Unity's underlying interface is written in C + +, and the engine also encapsulates Unity engine for interaction with the C + + underlying interface. DLL and Unity engine. DLL, which takes advantage of Mono's cross-platform features and uses c # as the game scripting language for rapid game development [6]. Thanks to the editing mode in Unity, developers can adapt the Editor page to their liking as long as they place the corresponding scripts in the Editor folder. Also, Unity has a resource store where developers can download their own plugins, skyboxes, particle effects or resource packs to create great designs. In addition, Unity has posted videos, works and tutorials on its website for developers to learn from, as well as building a community for developers to discuss issues and improve themselves.

Unity is also popular with many developers because of its cross-platform capabilities. Unity can ship games to Windows, iPhone, Android, and, in the latest version, developers can use virtual joysticks on mobile devices, which makes development much more efficient. [7]

2.2. Development Languages

Unity3D supports javascript, C # and Boo, among which javascript and C # have a wide development population and rich development resources, most Unity developers use C # as their development language.

C # was launched by Microsoft in July 2000. NET Framework, a programming language. Syntactically, C # is very similar to Java in terms of syntax such as single inheritance, interfaces, and the process of compiling C # into intermediate code and then running it is similar to Java [8] . However, C # is very different from Java because the two languages were developed by different companies and the platform for C # services is net. NET. Java. Framework, which it edits. NET Windows network framework is very convenient.

The language definition of C # inherits C and C + + , and improves it at the same time, drawing on the advantages of many programming languages, has its own unique characteristics.

Overall, the C # language features are clear, concise, stable, and secure, taking advantage of its parent C/C + + language, but at the same time, they have managed to get rid of their complexity [9]. At the same time, the application of a wider platform, and more easy to learn, therefore, the project choose to use C # as a development language.

2.3. UGUI Components

The UGUI was developed by the NGUI authors and is a set of UI components officially released by Unity and integrated into the Unity compiler since Unity4.6[10]. The original UGUI is relatively simple, the number of users is far less than Ngui, but with the development of technology in recent years, UGUI components tend to mature, more and more developers use it to develop game interface. The basic elements provided by the UGUI include text, images, buttons, sliders, and scrolling components, along with a powerful EventSystem event system for managing UI elements [11]. In the UNITY2017 release, the “Gallery” concept was added to make it more complete. For Unity, cross-platform functionality makes it attractive to a wide range of developers, but it also brings with it the problem that the resolution of the device used to develop the design release is variable, so, uGUI focuses on adaptive processing, providing anchors, layouts, alignments, and canvases to solve the problem of different resolutions on different devices.

3. Game Requirements Analysis and Design

3.1. Overview of Tower Defense Games

Before doing the requirements analysis, we need to understand in detail what tower defense game, tower defense game what kind of rules, and these rules naturally become our game requirements.

Tower defense game as its name implies that the player needs to build defensive towers, and through certain strategies to prevent the enemy's attack, and eliminate the enemy. For classic tower defense games, there are a few principles:

The enemy will be born from a fixed position on the map, then will follow one or more fixed path, from the starting point to the end, after the end of the damage to the player;

Players can build defense towers on both sides of the enemy's path. The towers will search for enemies in their attack range and attack them when they are in their attack range

Players need to use resources to build and upgrade towers, and to remove towers can also be a certain amount of resources to compensate. At the beginning of the game, the

player should have some initial resources to build defense towers, destroy the enemy can get the corresponding resources reward.

The strategy of the game focuses on the timing and location of the tower, while the playability of the game is reflected in the complexity of the map path, the enemy's health and defense, the function and attack power of the towers, and the input and output of money.

3.2. Game Content

Although the classic tower defense game has developed rapidly in recent years, the same gameplay has begun to cause players to gradually feel tired, so this project reference game “Tribal Conflict”, will be the classic tower defense game for appropriate innovation, combined with UGC and RTS game characteristics, no longer stuck in the old routine.

3.2.1. Gaming Background

The game in this project is called the rise of the tribes and is set in a land of hills and plains where there are many tribes. The player plays the role of a tribal chieftain and needs to establish his own tribe. He needs to build blacksmith shops, barracks, walls, and other buildings from the village to train soldiers and strengthen his tribe to resist the invasion of foreign enemies, at the same time can be appropriate to other tribes of aggression, pillaging resources, for the rise of the tribes and fight.

3.2.2. Rules of the Game

This game is a combination of UGC games and RTS games tower defense game, reference to the game “Tribal conflict” playstyle. In the game, players collect gold coins in mines and use gold coins to build walls, cannons and other defense facilities to resist foreign invaders. At the same time, they can build barracks and holy water pools by collecting holy water, produce soldiers and archers, and when armed forces have been built up, attack other tribes and plunder more resources.

3.3. Game Requirements Analysis

According to 3.1 and 3.2 content analysis of the two sections, we got the game requirements of the game, the following will elaborate the game requirements of the game.

3.3.1. Functional Requirements

The functional requirements of the game can be divided into game maps, buildings, bullets, soldiers, resources, etc. The following will be a detailed introduction of the functional requirements of these objects.

1.The game map..

The game map mainly includes the map base map, the map attribute, the building construction position, the soldier moving path that the player can see in the game and so on. The details of the map are as follows:

(1) the bottom of the map. The base map is designed by artists and loaded into the game by programmers in the form of pictures, frame animations, etc. . The coordinates of all the elements on the map are taken as the reference frame of the actual pixel size of the base map [12] .

(2) map attributes. Map attributes include map size, map type, map scale, etc. .

(3) location of the building. Building location refers to the coordinates of the location where the building can be built, as there may be some obstacles or gradients in the map that prevent the building from being built at that location.

(4) soldier move path. Soldier's moving path is a set of coordinates of the target position that the soldier can move autonomously after placing the soldier. Because on the map there will be some soldiers can not cross the place, such as buildings, trees, rivers and so on.

2.Buildings

Because this game combines the RTS game, the game buildings not only include defense towers, as well as barracks, mines, blacksmiths and other buildings. Buildings are the core of the game, which can be divided into appearance, type, numerical attributes, non-numerical attributes, behavior logic, etc. . Details of the buildings are as follows:

(1)Appearance. The appearance that the building shows to the player's appearance, is the building picture display, this game uses the static picture as well as the sequence frame's way display.

(2)Type. The buildings in the game are classified into attack and function types. Attack buildings are defense towers, while function buildings include military buildings, decorative buildings, resource buildings and other function buildings.

(3)Numerical properties. Numerical attributes define the building's level, health, build/upgrade time, and resources required to build/upgrade. There are also unique values for different buildings, such as the speed at which the barracks can produce soldiers, the inventory of holy water bottles and money banks, and the attack power and range of the towers.

(4)Non-numeric properties. Non-numerical attributes include building upgrade displays, upgrade effects, destroy effects, and defense tower attack effects.

(5)Behavioral logic. Behavioral logic includes the defense tower to search for the enemy and attack the enemy and so on.

3.Soldiers

A soldier is an entity that a player uses to attack other players in the game. It can attack and destroy buildings, and it can also attack other players' soldiers. Soldiers can be divided into basic attributes, marching attributes, and attacking attributes, status attributes and behavioral logic. Here's a look at soldiers:

(1)The underlying property. Base attributes include soldier type, soldier appearance, soldier name, soldier volume, and soldier collision volume.

(2)Marching attributes. The marching attribute refers to the behavior attribute of the soldier from the initial point to the attacking target location, which includes: the soldier's marching speed, the animation resources that the soldier uses when walking, the soldier's current position, the soldier's movement speed, the soldier's movement speed, the soldier's movement speed, the soldier's movement speed, the soldier's movement speed, the soldier's movement speed, the soldier's movement speed, the soldier's movement speed, the soldier's movement speed, the soldier's movement speed, the soldier's movement speed, the soldier's movement speed, the soldier's movement The soldier's path, etc. .

(3)Attack attributes. Attack attribute refers to the behavior attribute of soldiers when attacking the target, which includes the animation, attack force, life, attack speed, attack range and so on.

(4)Status attributes. Status attribute refers to the real-time status of soldiers, it includes: Rest State, defense state, attack state, death state, etc. .

(5)Behavioral logic. The behavior logic of the soldier includes: when the soldier is generated in his tribe, the initial state is resting state; when other players attack his tribe, the soldier will become defensive state. In a defensive state, soldiers will move around in a certain area, and search into

the attack range of the enemy, if the search of the enemy, the soldiers will attack the enemy until the enemy died; When the player leads his soldiers to attack other players, the soldiers will turn to attack mode, find their way and attack enemy players' soldiers and buildings until all targets are destroyed. If the soldier dies, the soldier becomes dead, and the number of soldiers is reduced by one.

4.Resources

Resources are also an important part of the game, and players need to consume resources to build and upgrade buildings and train soldiers. Resources can be divided into: appearance, type, numerical attributes, non-numerical attributes, etc. . Here is a detailed description of the resources:

(1)Appearance. The appearance is the resources to show the player's appearance, the game uses the static picture way to show the resources.

(2)Type. In this game, the types of resources are gold coins and holy water, respectively for building/upgrading buildings = and training soldiers.

(3)Numerical attributes. The numerical properties of a resource include: the interval between builds, the number of builds, the number of accommodations, and so on.

(4)Non-numeric properties. Non-numeric attributes include: generated effects, collected effects.

3.3.2. Non-functional Requirement

The game's non-functional requirement can be divided into the following points: availability, reliability, performance and support, system requirements and so on. Below is a detailed description of the game's non-functional requirement:

1.Usability

This game is a major game for young and middle-aged, which requires more user-operated content, need more than the classic tower defense game thinking. But in the user's operation design more intuitive, so that users can get on to the game faster.

2.Reliability

In the game due to various reasons Flash or forced to close, the game can record the player's current game progress, to ensure that the file is not lost.

3.Performance

The performance of the game is mainly reflected in the following aspects:

(1) resource preloading: resource preloading refers to the need to preload the game picture resources before entering the game scene. The preload time of this game is ≤8s.

(2) response to player commands: the game is controlled by the player using the keyboard and mouse, for the player through the keyboard and mouse issued commands, the game needs a quick response. The response time of the game is ≤0.1 s.

(3) frame rate: in-game frame rate is a measure of the number of frames displayed, that is, how many frames are played per second. The higher the frame rate, the smoother the game. This game has a frame rate of ≥50 fps.

4. supportability

This game is a combination of RTS tower defense game, need to design a good game map, buildings, soldiers, resources and other objects, so that players get a better game experience.

System requirements

Minimum configuration:

Operating system: Windows Vista

Processor: 1 GHZ

Memory: 512MB RAM

3D Accelerated 128MB Graphics Card

DirectX version: 9.0c

Storage: 100MB of free space is required

3.3.3. Other Requirements

1.UI

The game's UI system through the UNITY3D engine UGUI plug-in design. Because the game's UI is relatively simple, there is no need to pre-read the UI.

The game UI should include the following features:

(1)In the same scenario, any two UI interfaces should jump between each other;

(2)When a UI is not needed, it should be cleared

(3)For different types of computer, support for the adaptation of the screen;

2.Develop the system

The game needs to build on the relevant values, and then will build the values into the building.

3.Data System

This system can count the building's build value in the game, as well as the soldiers' and buildings' combat value when the player fights. After uploading to the server, it is convenient for the developer to evaluate the balance of the game based on the data, and adjust some of the values accordingly

4.File storage

The game may save some game archives and so on data to store in the local file, the convenience game reads.

3.4. Game Architecture Design

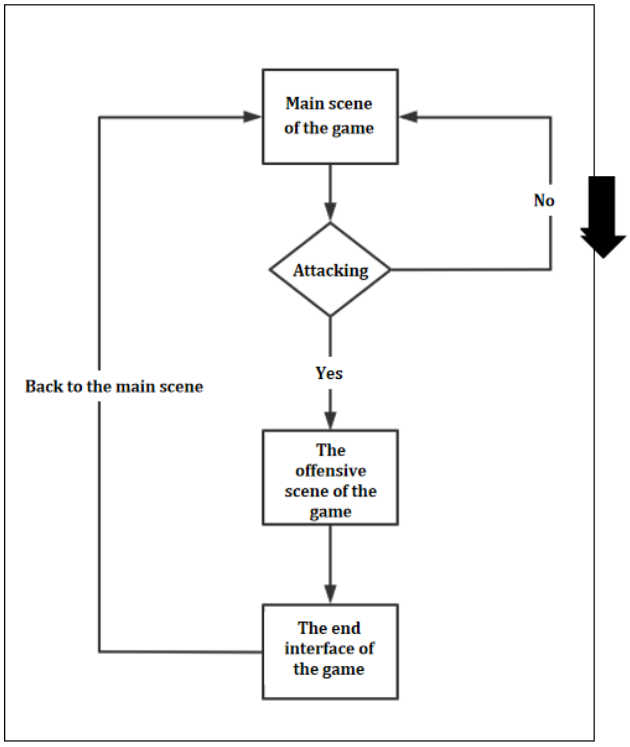

Figure 1. game flow chart

Based on the analysis of tower defense games on the market and thinking about the characteristics of the game, the structure of the game will include the following parts. First, the main scene of the game, which is the most important scene in the game, building buildings, collecting resources, training soldiers, etc. are all done in this scene; secondly, the game's offensive scene, when the player attacks other players will enter the scene, in this scenario, the player can deploy soldiers

to attack the other tribe. The above two scenarios are the focus of the game and are the ones that give the player the game experience. Finally, the game's attack end interface, when the player ends the attack, will enter the scene, show the player's attack results. This is the structure of the game.

Based on the above analysis and design of the game architecture, you can design the flow diagram of the game, as shown in Figure 1.

3.5. HUD Game Design

Heads up display (Hud) . First used in military pilots, the pilot does not need to bow to access the dashboard information, without hindering the normal flight field of view observation, reducing the problem of distracted driving [14] . In the game, the game HUD layer refers to the game-related information, such as levels, blood, money, etc. , displayed in the game screen. With this information, players would be able to learn about the most critical state of the game at any time without having to open other game interfaces to view it. This would eliminate some redundant steps and improve the game experience.

Based on the analysis of sections 3-4, the HUD layer in the main scene would be divided into two parts. In the upper part, it would present real-time information about the player's resources, including the amount of gold coins, holy water, and precious stones, players' user names and levels. This information needs to be updated in real time by an internal program.

The bottom half shows the player's Attack button and Shop button. Click on the attack button, the player will enter the game attack scene, attack other tribes, click on the store button, the player will enter the store menu, in the store menu, the player can choose to expend resources to build buildings. This area allows the player to operate and needs to be identified by the actions of the player's mouse and keyboard.

So in the main scene of the game, HUD layer to achieve the functions are: player information updates, resource updates, building construction and so on.

The HUD layer design for the game attack scene is similar to the HUD layer design for the main game scene. Only part of the scene is the HUD, which displays the Home scene button (Home) and the owned soldier. Click on the main scene button, will return to the main scene; click soldiers, then click the game map can place soldiers area, soldiers will be deployed down, automatically attack the enemy. This area also allows the player to operate, the need to identify the player's mouse and keyboard operations. The HUD design of the scene is shown in figure 3.7.

So in the game attack scenario, the HUD layer implements functions such as soldier deployment.

3.6. Building Design

Building design is one of the main focuses of the game and is highly relevant to the player's gaming experience. In this game, buildings are divided into functional buildings and defensive buildings. Here are the two types of buildings:

3.6.1. Functional Buildings

Functional buildings can not attack soldiers and have certain functionality. Functional buildings can be subdivided into the following categories:

(1)Military buildings: including training camps, barracks, shipyards

(2)Decorative architecture: three different types of trees

(3)Resource Building: Holy Water Collector, Holy Water Bottle, gold mine, gold storage pot, windmill, Gem mine

(4)Other buildings: town center, architect's hut, city wall

3.6.2. Defensive Buildings

A defensive structure that can attack soldiers and defend a tribe. Defensive buildings include: Cannon, Arrow Tower.

3.7. Soldier Design

The design of soldiers is also one of the main points of this game, it can help players attack other tribes. This game is designed for different types of soldiers, different types of soldiers have different attributes, flexible deployment of them, and the appropriate strategies to help players defeat other tribes. There are three types of soldiers in this game:

(1)Swordsman: High Health, high attack power, but short attack range.

(2)Archer: capable of striking from a distance, suitable for standing behind a swordsman and dealing fatal blows to other tribes.

(3)Architect. Used only to build buildings, with architect's huts appearing.

3.8. Resource Design

Tower defense game and resources are closely related, in this game, resources are divided into three categories, are: gold coins, holy water, precious stones. Different resources have different functions.

4. Game Implementation

4.1. Introduction to Key Technologies

The game has two key technologies, respectively A * pathfinding algorithm and LOOKAT method, the following will be introduced respectively:

4.1.1. A * Routing Algorithm

A * algorithm is a typical heuristic search algorithm, often used in the game artificial intelligence to find the way to achieve, searching for the least costly path from the starting state to the target state is done primarily by examining the contiguous or contiguous states of a particular state [16] . In this game, a * pathfinding algorithm is used for soldier movement.

In the A * algorithm, the search area is first simplified to a set of quantifiable nodes, and then the cost of each node is calculated by an evaluation function [17] . The evaluation function is: F = g + H. G represents the cost of moving from the current node to the current node; H represents the cost estimate from the current node to the end point. Start at the starting point, select the point with the lowest value of F until the end, then go back to get the shortest path.

A * The search process is as follows:

(1)Add starting point A to the Open List and set starting point A to the current node;

(2)Determine the adjacent nodes around the current node. If the node is an obstacle, or is already in the Open List or Close List, no processing is done; if the node does not fit in the three cases, add it to the Open List and add the current node to the Close List.

(3)Calculate the F value of adjacent nodes, and if any are already in the Open List, compare whether the F value under the current path is smaller. If it is smaller, the F value of that node is updated.

(4)Query for node M with the smallest F value in adjacent nodes, place it in the Close List, and set the current node as the parent.

(5)If node m is the end point, the pathfinding is successful; anyway, with node m as the current node, step (2) is returned, and the loop repeats until node m is the end point.

The A * pathfinding plug-in used in this game is downloaded from Github. Due to space limitations, the implementation code is given in Appendix A.

4.1.2. Lookat Method

Lookat method in this game for defensive building attack targets and soldiers to the target, will look at the target, make the game more realistic, improve the game experience.

Implements the LOOKAT method, which rotates the object's axis toward the target. We need to get the position of the object, the position of the target, using the VECTOR2 function, get the corresponding angle, thus rotating the object, point to the target. The implementation code is as follows:

```
Public Void LookAt (Vector3 Point)
{
Vector2 A = utilities. Getscreenposition
(this.transform.position-new Vector3(0,0,1)) ;
Vector2 b = utilities. Getscreenposition
(this.transform.position) ;
Vector2 c = Utilities. Getscreenposition (point) ;

Float angle = utilities. ClockwiseAngleOf3Points (A, B,
C) ;
This. This
   }
```

4.2. Implementation of the Game Interface

After thinking about the game, we decided to separate the game map scene and the game interface, using the Main Camera class to display the game map, use the camera CameraBound category to display the various interfaces of the game.

4.2.1. Implementation of the Main Game Interface

In this game, the game's main interface class is called GameOverlayWindow and is controlled by the GameOverlayWindow script.

First, use the Awake function to initialize the game interface; then, use the Start function to set the initial value of the game's three resources: gold coins, holy water, and precious stones, and at the same time, call the COLLECTRESOURCE function to ensure that the resource information is updated in real time; finally, set the Attack button and the Shop button, and activate them with the OnClick () function.

4.2.2. Implementation of the Game Attack Interface

In this game, the game attack interface class named attack overlaywindow, by the attack overlaywindow script control.

First, the game interface is initialized using the Awake function; then the Home button is set; finally, the swordsman and Archer buttons are set, and the corresponding counter display is configured. The Next button was originally set up to go to the Next tribe, but since only one tribe has been developed so far, the Next button is not shown.

4.2.3. Implementation of the Game Attack End Interface

In this game, the game attack closing interface class is named Resultwindow and is controlled by the ResultWindow script.

First, use the Awake function to initialize the game interface; then, set the ReturnHome button; and finally, use the SETDATA function to compute the results of the interface, such as the number of swords and archers used to win or lose.

4.2.4. Implementation of the Game Store Interface

In this game, the game store interface class is named ShopWindow and is controlled by the ShopWindow script, the CategoryItem script, and the SubCategoryItem script.

The store's interface is divided into two levels. The first level is a CategoryItem of the building, including Army, Defence, Other, Resources, Treasure and Decorations, buttons are set up, and when you click the button, you go to the corresponding sub-category. On the second floor are subcategoryitems, such as the Cannon and Tower in the Defence building, which have buttons that can be clicked to make a purchase. The interface of the subcategory is shown in Figure 4.6. Finally, the background of the shop is added to form the shop interface. The store interface also has a BackButton and a CloseButton.

4.2.5. The Realization of the Game Map

The game map consists of four elements: Sand, Ground, Grid, and Cliff.

In this section, map movement is an important and difficult point, which is described below:

In this game, the main Camera will be moved to achieve the game screen map movement, controlled by the Camera Manager. The OnScenePan () function is used to move the map, and the UpdatePanInertia () function is used to move the map. At the same time, we should also set the boundaries of the map, that is, set the boundaries of the main camera to move, the clampamera () function to achieve.

Through the above steps, you can move the map.

4.3. Building the Building

Building buildings is the core of the game, but also the need to achieve the difficult point of technology. We will implement it in two parts.

4.3.1. Builder

In this game, architects are required to build buildings that can not be generated immediately. Therefore, when building a building, look for idle architects. If not, it can not be built, and the window “All builders are busy” will pop up, as shown in figure 4.10. It is controlled by the BaseItem script and the Builders Busy Window script, which first searches for idle architects via the GetFreeBuilder () function and then activates the architect to build via the BuilderAction () function.

4.3.2. Building

After the player has purchased the building from the store interface, the building will be generated randomly on the map where it can be built, controlled by the SceneManger script and the GroundManager and Soundmanager scripts, as shown in figure 4.11, this is done in five steps:

(1) Find the free space

First, we need to find a random free position for the building. The GetRandomFreePositionForItem () function is used to obtain a valid random position, and the GetFreePositionForItem () function gives the building a free position.

(2) free-position build detection

Before building, you need to detect the previously selected location to determine whether it can be built. You can not build beyond the map or if there are other buildings in the location. If the selected free position is not valid, the free position is re-selected and implemented by the IsPositionPLacable () function.

(3) building-generated location

After these two steps, the final building position is obtained, which is controlled by the GetItemInPosition () function.

(4) building ids were generated

Each building has its own unique ID, for a new building, we need to give it a new ID, so that the convenience of background access to its data.

(5) building generation

After the above four steps, all the links are ready, and the final building is generated by the AddItem () function.

4.3.3. Moving the Building

After building is randomly generated, if the player is not satisfied with the location of the building, the building can be moved. There are two special situations when moving a building. First, you can't move the map when you're in an attack scenario, and second, the Wall is a special type of building that takes on different shapes when multiple walls are built together. Moving the building will deal with these two situations. Moving a building is divided into four steps: selecting a building, starting to move, moving, and stopping to move. This is controlled by the BaseItem script and the SceneManager script together with the Soundmanager script. The implementation is shown in figure 2.

(1)Select the building

Implemented by the set selected () function, the selected UI appears when selected and the sound effects are played.

(2)Start moving

Controlled by the OnItemDragStart () function, the movement is checked to see if there is a special case, if it is in a game attack scenario, it can not be moved; if it is a wall that is moved, it needs to be updated.

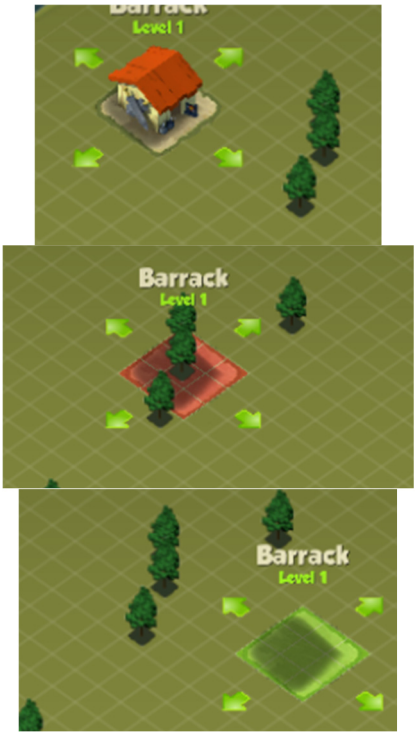

Figure 2. moving buildings

(3)Moving

Is implemented by the ONITEMDRAG () function, and then starts moving, updating the location of the building and displaying the UI based on whether the location can be placed, the grid color of the selected building's UI appears green; otherwise, it appears red.

(4)Stop moving

Controlled by the ONITEMDRAGSTOP () function, the location is checked for placement, and if it is, the grid color of the UI for the selected building is green; otherwise, it is red, and the location information is updated.

4.4. The Realization of Defensive Buildings

Defensive buildings defense named Defender class, it's steps are as follows: in the attack area to search for attackers, activate the defense, attack the target. The effect is shown in figure 3. It is controlled by the Defender script.

(1) search for attackers

Implemented by the SEARCHFOR attacker () function, which uses cooperative exploration to search for attackers and enter a defensive state if the searched attacker is not destroyed.

(2)Initiate defense

Firstly, the nearest target is preferentially attacked, which is controlled by the GetNearestTargetItem () function, searching all the objects in the attack range, removing the non-attacker targets, comparing the distance of the remaining attacker targets, and obtaining the nearest target; Then start the defense, by Defend () function to achieve, search to the target, will enter the defense cycle, by DefendLoop () function to achieve, if the target health value is greater than 0, the defensive building to initiate an attack state, attack the target.

(3) attack the target

Implemented by the HitTarget () function, defensive buildings will attack the target until it is destroyed.

Figure 3. implementation of a defensive building

4.5. Realization of Soldiers

The realization of soldiers is divided into soldiers deployment, soldiers movement and soldiers attack. Here's how to make it happen. The effect is shown in figure 4.

4.5.1. Deployment of Soldiers

In this game, the soldier's deployment needs the player to click the soldier button first, then clicks on the game map the player wants to deploy the position. Because the actions and code for clicking buttons have already been explained above, this section will show you how to implement click-to-play maps. Clicking on the game map is called the ONTAPGROUND class and is controlled by the SceneManager script. Through the function OnTapGround () implementation, first to determine whether to choose a

swordsman or Archer, and then the interface to settle the number of soldiers consumption statistics, sensing the location of the Click, update the corresponding information.

4.5.2. Soldier Movement

The soldier's movement is based on the shortest path found by the A * algorithm. Through the Walker script, we will implement the soldier's movement in the following parts:

(1) updating the walking state

Controlled by the WalkUpdate () function, the location is updated in real time by calculating the distance traveled.

(2) walkupdate to the target location

By the WalkToPosition () function, at this point, because there is a special kind of building-the wall, we need to distinguish between the main game scene to walk or walk in the game attack scene. In the attack scenario, soldiers can destroy the wall without having to walk around it.

(3) end of walk

Implemented by the finish walk () function, which sets the walk state to false and updates the position to the target position.

4.5.3. Soldier Attack

A soldier's attack is like a defensive building's defense, but the soldier attacks the target based on the path he walks. The soldier's attack is controlled by an Attacker script, which consists of the following sections. Among them, the same code as the defense part of the defense building, such as (), () , etc. , will not be repeated.

(1) search for the nearest target to attack

It is realized by the AttackNearestTarget () function, and based on the method of getting the nearest target, the attack target is obtained.

Figure 4. soldier deployment, movement, and attack implementation

(2) attack

It is realized by the Attack () function, which first obtains the position of the target and controls the soldier to go to the target. When the target enters the Attack range, the soldier stops walking and enters the Attack cycle. The attack cycle is realized by the AttackLoop () function. If the target is destroyed, the attack will stop. If the target's life value is greater than 0, the attack will continue until the target's life value is less than 0.

(3) detect whether the target exceeds the attack range

Implemented by the CheckTargetBeyondRange () function, if the distance between the target and the soldier is less than or equal to the attack range, the soldier stops walking and enters the attack cycle.

(4) soldier attack sound

Implemented by the PlayHitSound () function, it is divided into swordsman attack sound and Archer attack sound.

4.6. Resource Implementation

Resource realization includes resource generation, resource collection, resource consumption and resource information update, which are controlled by production script and SceneManager script. The implementation is described in detail below.

4.6.1. Resource Generation

The generation of resources is relatively simple, as long as in the specified building in accordance with the production rate of the generation of resources can be. Resource generation is accomplished by the UpdateProduction () function. When the amount of production is greater than or equal to the specified number to be collected, the resource can be collected and the corresponding icon pops up. The effect is shown in figure 5.

Figure 5. resource generation

4.6.2. Collection of Resources

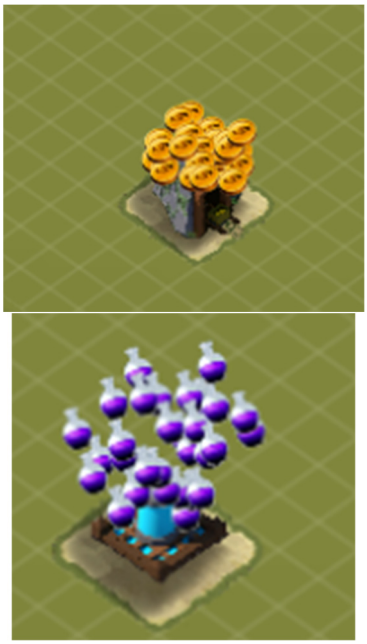

Figure 6. collection of resources

After the resource is generated, the player can collect the resource, and after collecting, the resource will be produced again. When the collection, will play the corresponding sound effects and launch the corresponding particle system, the information collected will be displayed in the main interface. The collection of resources is implemented by the CollectResource () function, which emits the particle system, plays the collection audio, and the collection UI disappears, updates the last collection time, and updates the amount of resources in the main interface, the effect is shown in figure 6.

4.6.3. Consumption of Resources

Resources are expended when players train soldiers or build buildings. Resource consumption is achieved by the ConsumeResource (-RRB- function, which determines whether there is enough storage to consume, cancels the operation if there is not enough, and updates the resouUIe UI when there is enough.

4.6.4. Update of Resource Information

Resource information needs to be updated because both resource collection and resource consumption can change the amount of resources. Resource updates are implemented by the RefreshResourceUIS () function and are updated on the resource UI of the game interface.

Figure 7. the overall effect of the game

5. Debugging and Testing of the Game

5.1. Game Test

After debugging, the game can already be run on the computer, the game screen will not be stuck, nor Flash or forced to close the phenomenon, consistent with the stability of the game. At the same time, after playing the game, the difficulty was moderate and the game was playable and balanced.

Players into the game, according to their own preferences, the game experience, play the whole game. The normal game test is shown in Table 1. The results are shown in figure 8.

Table 1. Normal game test

Use Case name	Normal game
Basic description	The player plays the game normally
Test scenarios	Players play the game as they see fit
Expected results	The game is running smoothly with no apparent bugs or hiccups
Actual results	Found a bug in the game
Judgment	Pass

Figure 8. normal game test chart

6. Summary and Outlook

6.1. Full Text Summary

Because tower defense game in recent years into the old routine, lack of creativity, the game through the reference game “Tribal conflict”, combined with RTS, UGC and the characteristics of tower defense game, tower defense game for a certain degree of innovation. The main work of this article includes:

Game requirements analysis. According to the characteristics of tower defense game and the rules of the game, and the current popular tower defense game “Tribal conflict” as a reference, combined with UGC and RTS game characteristics, the demand for this game was analyzed.

Game Design. Based on the analysis of the requirements of the game, it designed three scenes in the game, fifteen buildings, two soldiers, three resources, and introduced their corresponding attributes and functions.

The realization of the game. According to the design of the game, the functions of the main scene, the attack scene and the attack scene, as well as the switching between them, the creation and movement of buildings, the attack of defensive buildings, the deployment of soldiers, mobile and attack, resource generation, collection and information updating are the core contents of the game.

Debugging and testing. Test the game, check whether the functions of the various parts of the implementation, whether the smooth operation of the game, etc.

6.2. Outlook

The game is still in the beta stage, with many details and features yet to be developed. The next steps will be to further refine the game and develop more features. For example:

Build the game's build upgrade system, buildings and soldiers will have a stronger function after the upgrade.

Add levels to the game as appropriate.

Adjust the price of a building.

Enrich the buildings and soldiers in the game.

Develop online functionality.

Fix game bugs.

I believe that the game will be more playable and more widely accepted and supported by the players after the game is improved and added more functions according to the problems at this stage.

References

[1] Roman M, Sandu I, Buraga s c. Owl-based modeling of RPG games [J] . Studia Universitatis Babes-bolyai, 2011,56(3): 83-85.

[2] Barreteau o, Le Page C, d'Aquino p. Role-playing games, models and negotiation processes [J] . Journal of Artificial Societies and Social Simulation, 2003,6(2): 41-44.

[3] Jia Xiuhai. A study on leisure game behavior of young and middle-aged people in Dalian [D] . Dalian: Dongbei University of Finance and Economics, 2011.

[4] Zhang Jing, Huo Yumei, Wang song, Liang Xiongjian. Business ecosystem analysis of mobile games [J] . Beijing: Journal of Beijing University of Posts and telecommunications (social sciences), 2008,04:15 -19.

[5] Lu Huang. Design and implementation of tower defense game and its numerical balance model [D] . Guangzhou: Sun Yat-sen University, 2015.

[6] Zhang Jiadong. Study on multi-base station positioning technology of simulated training vehicle [D] . Shenyang: Shenyang Ligong University of Technology, 2021. DOI: 10.27323/d. CNKI. GSGYC. 2021.00002.

[7] Tonge. Unity3d-particle System [DB/OL]. Https://www. cnblogs.com/tonge/p/3922545.html, 2014-08-19/2022-04-13.

[8] Hao Zhong, Xu Chen, Wen Qian Shu, Xiao Dong Chen. Research and Development on 3D Simulation Training Evaluation System of Substation Based on Unity3D [J] . Applied Mechanics and Materials, 2015,3759(727): 147-148.

[9] Liu Junyao. Strategy development game design and implementation based on Unity3D [D] . Jilin: Jilin University, 2017.

[10] Zhang Fufeng, Wang Min, Jin Hui. Implementation of dynamic material modification based on Unity3dugui [J] . Computer Literacy, 2020,16(25): 218-219. DOI: 10.14004/J. CNKI. CKT. 2020.3040.

[11] Xuanyu pine. Unity 3D game development [m] . Version 2. Beijing: People's Posts & Telecom Press, 2018:112.

[12] Wang Yifan. Design and implementation of tower defense mobile game framework based on Cocos2d-x game engine [D] . Wuhan: Central China Normal University, 2015.

[13] Translation by Bernd Bruegge, Allwn Dutoit, Ye Junming, Wang Wangzhu. Object-oriented software engineering, using UML, patterns, and Java [m]. 3rd ed. Beijing: Tsinghua University Press, 2011.

[14] Yang Fei, Lin Shaohua, Wu Chunjian. Head-up display in cars [J] . Light Vehicle Technology, 2013(11): 4.

[15] Dong Quanli. Design and implementation of tower defense game system based on IOS platform [D] . Huazhong University of Science and Technology, 2013.

[16] Yang Tianqi. Artificial intelligence and its applications [m] . Guangzhou: Jinan University Press, 2014:47-50.

[17] Dunway. Interactive design of ASTAR routing algorithm based on C # Winform [J]. Information technology and informatization, 2021(08): 80-83.