Design and Development of RPG Based on Unity3D

Hao LIU* and Dong-Lin CHEN

School of Information Engineering, Beijing Institute of Fashion Technology, Beijing, 100029, China
E-mail: gxyliuh@bift.edu.cn

*Corresponding author

Keywords: Unity3D, RPG, Game development.

Abstract. RPG (Role-playing Game) is one of the most common types of games in the game market, and Unity3D is a very popular engine. Usually, a RPG project development contains of the overall design, the description of mission system, the interface system design and the development of the game scene. The usage of NGUI plug-in to build UI (User Interface) has introduced in this paper. Based on that, the function of game is divided into menu, dialog box and knapsack. The design of mission system and the common methods of game role control have been analyzed in more detail. Through a series of test cases, it proved the game can be operated as expected.

Introduction

With the rapid development of the economy and technology, we want to be able to get a perfect experience. Unity3D is one of the most popular game engines in recent years, there are many popular games which are developed by Unity3D, such as Temple Run and Hearth Stone. RPG is one of the popular type of game. The player chooses a role and finish the mission for upgrading, and then complete a more advanced mission. During the development process of RPG, the design of user interface is very important. Besides, realizing the character control, making the game has a delicate menu, the transition animation, intelligent characters and interesting mission system was needed. This paper describes the complete development process of a game project which developed by Unity3D and C# programming language [1].

Game System Design and Implement

Game System Design

The main process of game development includes market research, preliminary design, detailed design, development (scene modeling, scripting, etc.), debugging optimization and release. The system design mainly includes the mission system, the UI system and the character control. The menu system to achieve the main menu, dialog box and knapsack, the knapsack make the character be able to collect the items, Mission system provides the core of the game play. The entire game system design diagram is shown in Figure 1.
Mission System Design

The mission system can guide the player and make the player clear the system function. The mission system design should follow the basic rules: mission trigger, mission completion and mission reward.

Generally, the mission of the game will be triggered when certain conditions are met, then the player can receive tasks from the NPC. The task execution process is implemented by the combat program, so the core gameplay is to verify the number of monsters killed. Players need to destroy the enemy is divided into two categories: ordinary enemies and BOSS. The collected items are divided into the following three categories: the enemy is bound to drop the item, such as a certain amount of money; the designated enemy randomly dropped items; items which are visible and available on the map.

In RPG, task triggered by the collision detection, when the NPC detects the player, the mission system will determine the progress which the player have completed, then the system assigned the task according to the player level.

User interface Design

The menu system includes the design and implement of the menu, the knapsack and the animation of interface transition, the sub module of the menu system has a lot of similarities. Figure 2 shows the game interface; it uses NGUI plug-ins to achieve.
Menu and Knapsack System

Knapsack system is particularly important in RPG games, it provides container of equipment for the character, when the player buys medicine, weapons, and other items, and the items will be displayed in the knapsack. In addition, when the character needs to use the items, they can be choosing in the knapsack.

The Animation of Interface Transition

The most concise way to load scenes in Unity3D is to use the function Application. LoadLevel(SceneName), this method is used to load the scene synchronously, which is suitable for the game with small scene resources. The transition interface is mainly used in the following occasions: the start menu to the game scene, the game ends to the outcome scene.

Character Control

In this RPG, the role of the game mainly including the player, the enemy and NPC. The movement control and steering control of the player is different from the enemy. In addition, the collision detection of the character is also required.

Character Move Control

At first, set the keyboard input receiver: h=Input.GetAxis(Horizontal), gets the X axial movement in three dimensional coordinates, v=Input.GetAxis(Vertical), gets the Y axial movement in three dimensional coordinates. In the Unity3D engine, the default control keys for Horizontal and Vertical are A, D, W, S, then we can move the character by press the buttons on the keyboard [2]. The player through Character Controller. SimpleMove(new Vector3 (h,0,v)) function to achieve move control, we need add the Character Controller component to Player, and the SimpleMove function be able to achieve the desired results.

Character Rotation Control

Script of player look at the mouse.

  //Create a ray from camera to the point of the mouse [3].
  Ray ray =Camera.main.ScreenPointToRay(Input.mousePosition);
RaycastHit hitInfo; //Define variables for receiving information.
if(Physics.Raycast(ray, out hitInfo, 200)) { //Ray detection
    Vector3 target = hitInfo.point;
    target.y = transform.position.y; //receive information.
    transform.LookAt(target); //rotation(player look at the mouse).
}

Collision Detection

Collision detection module is the core of the physics engine, based on the physical world. It consists of three basic modules: initial collision detection, accurate collision detection and accurate intersection [4].

In the initialization phase, the collision detection module gives each object a bounding box. Accurate collision detection stage does the space division of the whole virtual world. Finally, accurate collision information is obtained by using the convex body algorithm and the ray detection algorithm in the precise intersection module, as shown in Figure 3.

![Collision detection module flow chart](image)

Figure. 3. Collision detection module flow chart.

When the collision box is set to trigger, if a collision is detected, the relevant information is returned, and then determine whether the need to carry out the next operation. In this game, the triggered collision is mainly used in the scene where the player interacts with NPC.

Test Case

Bug was found earlier, the lower cost of repair [5], Test case involves task testing and item update, the task testing includes three parts, the task trigger, the task flow and the cross correlation. Task trigger mainly includes two aspects: trigger condition and task record. Here enumerates the test case of task trigger.
Test function: task trigger. Test purpose: check the task whether can be triggered normally. Test input sequence: Player level is not enough task requirements whether can trigger the task. Player has just reached the requirements of the task on the level, would whether trigger the task correctly. If cannot trigger the task, prompt information is correct or not.

Test results: failed to trigger the task when the collision is too large, the task triggered successfully after the adjustment, as shown in Fig. 4.

Figure. 4. Adjust the collider size.

Conclusion

Through the RPG project which developed by Unity3D, the paper gave an overview of the overall design of RPG, UI and the game scene development, shown a brief introduction to the use of the NGUI plug-in to build UI. This paper analyzes the common methods of the mission system design; it has discussed the character control at the character design part. In addition, the project has written the test case, in order to ensure that the game can be run as expected.

Acknowledgement

This work is supported by Beijing Institute of Fashion Technology under Grant NO. NHFZ2016011 & NO. NHFZ2016068/003/007, and the corresponding author of this paper signs liuhao.

References


