International Journal of Computational Intelligence Systems and Applications. Volume 11, Issue 1, (2024) 18-30
http://www.cayley-nielson.com/ijcisa/

3D Game - Design and Development of the Magic Park

Lin Shi[1], **Zhigang Li**[2]

[1] School of Artificial Intelligence, North China University of Science and Technology ,Tangshan, Hebei, China

[2] Computing Center, TangShan College, Tangshan, Hebei, China

E-mail: *12389248@qq.com*

Abstract

This paper designs and develops a third person 3D game - Magic Park. The core of the game is UI framework management module, combat module, and enemy behavior logic module and so on. It provides players with a good gaming experience through cool graphics and real combat feel. The game uses C# as the scripting language, MySQL database to obtain and store game data, Unity3D engine as the core technology, shader language to beautify the scene, Animation state machine to make the character movement smoother.

Keywords: RPG; Space Sci-Fi; Unity; C#.

1. Introduction

At present, the development of the game industry is in full swing, and 3D games can give people a three-dimensional feeling, so that players in the game can experience the length, width and height of the three metrics, and the ability to rotate the perspective of 360 degrees, which greatly increases the game's degree of freedom and fun as well as the authenticity of the game, 3D games are making a splash.

While 3D game development involves a wide range of knowledge, more and more fields need to use 3D technology to enhance user experience and efficiency. Using 3D game development technology, in the direction of game production, combining VR and games, to bring players a more realistic and interesting game experience; in the direction of architectural animation, the use of 3D development technology to simulate the architectural design and construction process, to improve the quality and efficiency of the design; in the direction of education and training, the use of 3D development technology to create virtual classrooms and laboratories, to enhance the interest in learning and the effect of the direction of health care, the use of 3D development technology to simulate the human body, to improve the user's experience and efficiency. In the direction of medical and health care, 3D development technology is used to simulate the human body structure and disease process to improve the diagnosis accuracy and treatment effect. Through the 3D game development exercise related technology, for the development of all walks of life in various fields have good help.

This paper mainly designs and implements a 3D game - Magic Park. Firstly, a brief introduction will be given to the functions accomplished in this game, through the introduction of the functions, there is a general understanding of the whole game functions and game presentation.

Then there will be a brief introduction to the planning and material work of the game, which are the preparations that need to be done before developing this game. By introducing this part, we can have a more detailed understanding of how the preparatory work for developing this game is carried out, including the core gameplay, art style and so on.

After the introduction of the preparatory work, in order to have an overall understanding of the game, the game framework and various parts of the game script will be introduced to introduce the functions achieved by each script as well as the construction and integration of the overall framework, after which some important functional scripts in the game will be introduced.

2. Planning and preparation

The pre-development planning mainly includes the core gameplay design of the game, the art style and the development environment used for the project.

2.1. Design of the overall content of the game

This system is supported by the Unity3D development platform, which can support virtual devices such as mobile phones, PCs, game consoles, and VR, and has the characteristics of flexibility and versatility [1].Players can enter this game by logging in and registering after opening the game.

After logging in, he creates his favourite character and enters the game lobby. In the lobby, the character attributes, backpack and character list can be viewed and other functions. By finding NPCs to accept corresponding mainline quests, enter different levels to defeat various types of enemies, complete the mainline quests and obtain quest rewards.

2.2. Target platform and development environment of the game

This game is suitable for PC and other mainstream platforms, and is suitable for Windows 10 development system. When running the game, you should make sure that the computer system used is one that supports Unity, and most of the PC systems on the market at present support Unity, but it is still recommended to use the system on Windows 10 to run this game, and the development environment used on the PC is Visual Studio 2017.

2.3. Preparation of game materials

After the scheme of the game is basically determined, it is necessary to prepare the materials needed for the game, including model materials, particle effects materials and picture materials. Model materials include the capsule map scene, characters, enemies and NPC models, etc. Picture materials include model textures, login backgrounds, UI buttons and so on. The texture images make the models more beautiful and realistic, and the UI images ensure that the overall interface is beautiful and unified with the overall game style. All the image resources are placed in the project file Resources\Icons\ folder, and all the model resources in the project are stored in the EnemyPrefab and Role folder in the Resources folder, the file format is prefab.

3. Requirements Analysis and Design

3.1. Game Requirements Analysis

The game should try its best to achieve the completeness and playability of the game, and complete the control of the characters by manipulating the mouse and keyboard. To achieve the first registration and login to enter the game, create and manipulate the character and NPC dialogue to accept the task, enter different levels, release skills to defeat the enemy, complete the corresponding tasks to obtain the copy of the task rewards.

And through which you can add skill effects. The use of shaders to complete the scene and character

beautification, add background music, game effects will greatly enhance the game's cool picture sense, and bring players a good game experience [2].

3.2. Game Functions

Understanding the functional overview of this game gives an initial understanding of the game. Through this understanding, you can generally know the gameplay of this game and have a simple understanding of the gameplay of this game. The "Magic Park" game mainly includes the registration interface, login interface, main interface, character creation interface, space station and various copies, the following will be a brief introduction to some of the "Magic Park" game scenes and the effect of the operation of the specific steps are as follows.

(1) Run the game, enter the game registration and login interface, after the registration is completed, enter your user name and password to log in the game. As shown in Fig.1 as shown in Fig.2.

Fig.1 registration screen

Fig.2 login screen

(2) After registering and logging in, you will enter the main interface of the game with options to start the game, view characters, and option settings, as shown in Fig.3.Click on the option settings to adjust the volume level and mouse sensitivity, as shown in Fig.4.

Fig.3 main interface

Fig.4 Option Settings screen

(3) After clicking Start Game to enter the character creation interface, players can choose a specific character according to their personal preferences.As shown in Fig.5 and Fig.6.

Fig.5 Create a male character

Fig.6 Create a female character

(4) Set the name and confirm whether it is available and click start, and confirm the creation, as shown

in Fig.7.You can enter the game screen, the character will be initialized in the space station, where you can release skills at will, accept the corresponding tasks after dialogue with different NPCs, and unlock the corresponding copy scene after successfully accepting the task.As shown in Fig.8

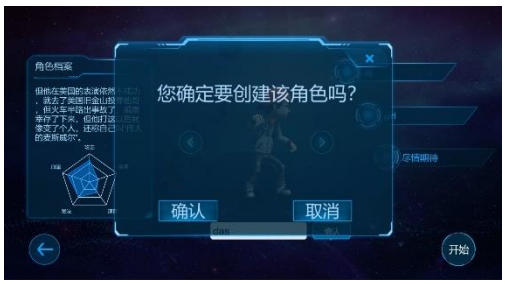

Fig.7 Create a role

Fig.8 Initialising the role

(5) The space station is the place where the character receives quests and selects maps, as shown in Fig.9.There are a total of 4 NPCs at this location, and each NPC will provide different quests.Press E near the NPCs to talk to them and accept quests, as shown in Fig.10.

Fig.9 release a skill

Fig.10 Dialogue with NPCs

(6) Accepting Quests. There are 4 NPCs in this scene, each carrying a different quest, and each quest carries a different generous reward. Dialogue with each NPC and receive the corresponding task, as shown in Fig.11.The received tasks can be viewed by pressing the "T" key to bring up the task list, unfinished for the accepted state, complete the task will become completed state. As shown in Fig.12.

Fig.11 Task Acceptance List

Fig.12 List of Accepted Tasks

(7) Enter the copy. After accepting the mission, first enter copy one, as shown in Fig.13 this level has three waves of monsters, the first two waves have two monsters in each wave, after defeating the first two waves of monsters, the third wave of the BOSS Gario appears, the BOSS has a high attack power, and will sprint to attack after two attacks, you need to be careful to avoid it. After defeating the BOSS, a victory screen will pop up, as shown in Fig.14, and the map console will be unlocked. After the completion of Copy 1, enter Copy 2. There are 5 waves of monsters in this level, with new exploding monsters and transient robots. After recapturing the space station, prepare for the Trial of Destiny. Enter copy three and defeat 4 enemies to complete mission three. Enter copy four. After completing the Trial of Destiny, prepare for the

mission "The Final Battle", this level has different monsters in different paths and rooms, defeat the monsters in each room in order to get to the final room.

Fig.13 Access to copy one

Fig.14 Victory screen

(8) Online team. This game is a multiplayer online competitive game, can be more than one person to team up together to challenge the copy. After connecting to the same IP address, multiple players can appear in the same scene at the same time, as shown in Fig.15, and can see each other releasing skills and attacks, as shown in Fig.16, and at the same time, they can also complete the interaction between each other and chat with each other as a team.

Fig.15 Simultaneous scenes

Fig.16 Simultaneous release of skills

(9) Create a team. To experience the joy of teaming up to challenge a copy, you need to press the "O" key to open the team panel, as shown in Fig.17.Into the team panel, you can name the team through the team name for a name, and can be introduced to the team in the team profile to introduce the purpose of the team, and can be set to set the maximum number of team limitations, set up after clicking on the creation of the team, as shown in Fig.18..

Fig.17 Create a team

Fig.18 Set team limitations

3.3. Game Script File Design

This section will introduce the functions of the UI framework, character management module, enemy class module, input and output module, network management module, database, shader and other files, and the scripts and files that make up these modules will be described in detail below.

3.3.1. UI Framework Module

(1) Core management script UIManager.cs

UIManager, as the core management class of UI framework, has three main roles: parsing and saving all the panel information, creating and saving all the instances of panels, and managing and saving all the display panels.

(2) Page Control Script MapPanel.cs

The key of the game lies in the flexible switching between the game and the UI interface. When some functional interfaces are opened, the game will be paused in order to focus more on the operation of the functional interfaces, and at the same time, the game content is also preserved in order to continue the game after the operation.

3.3.2. Input/Output Management Module

The key of the game is interactivity, and the input/output module is the core of human-computer interaction. We know that when the player presses the preset keys on the keyboard, the game state will be changed, for example, the player can use the "W", "S" on the keyboard, "A", "D" keys on the keyboard to control the movement of the character, the "T" key can view the current mission status, etc. Such interaction ability is dependent on the input/output management system to listen. The corresponding input/output script is NormalInput.cs.

3.3.3. Character Management Module

Character management module is the most core part of this project, which will be introduced in detail from the aspects of character animation (Fsm character management animation), skill effects, skill damage judgement, sound management and so on. When we enter the space station or a copy of the game, the character will be instantiated and the character control script will be enabled. This includes the FSM character animation management script FsmBase.cs, the character management core script ThirdPlayerController.cs, the skill base class script PropBase.cs and the skill management script AjPropManager.cs.

3.3.4. Enemy Logic Module

(1) Enemy base class EnemyBase.cs

Each monster in the game needs to attack, and attack in various ways, so in order to facilitate the writing of clear code and the subsequent writing of various monster attack code, first create a monster base class for all other monster attack scripts to inherit and call.Including blood, attack power, attack range, focus on the target, state detection, death sink and other functions.

(2) Enemy Trigger Generation Script EnemyTrigger.cs

After entering the copy, different enemies will appear in different scenes, and the enemies will appear in waves, and the number of each appearance is not certain.The enemies will appear in different waves, and the number of enemies will not be certain each time they appear.So we create this enemy trigger script to judge which kind of enemy should be generated in different copies and different locations.

(3) Enemy Blood Bar Script EnemyHpBar.cs

Enemies in the copy will attack and be attacked, and the most intuitive basis for players to judge the various states of the enemy is the blood bar, and the blood bar should always follow the enemy movement.This script is used to control the creation of the enemy's blood bar, the blood level update, the

blood bar following the enemy and other functions.

3.3.5. Network Management Module

(1) Client communication implementation script PhotonEngine.cs

The connection between the client and the server needs an intermediary to be realised, the script realises the connection and interaction of the server and sends requests and data to it, and sets it as a singleton mode, so that only one exists in any scene and interacts with the server through it.

3.3.6. Shader Development

A shader is a program that runs on a graphics processing unit, and most of the rendering in this game is done through shaders. In order to improve the effect of game graphics, the development of this game uses a variety of shader effects. Such as occlusion perspective effects, Gaussian blur, motion blur, X-ray effects and so on. This game is an online role-playing game, the camera will follow the character to move. Inevitably, there will be some buildings in the scene between the camera and the character, so the camera can not determine the position of the character. An occlusion perspective effect shader is used to colour the occluded part of the character to make it appear on the screen [3].

3.3.7. Database tool files

The database tool file is MySQLConnectUtil.cs, which is used to obtain a connection to the database, complete the insertion, deletion, and update operations on the data and the function of getting data from the database. This game uses the database to store usernames and passwords as well as to store role information and to review login account information, etc

3.4. Game Framework Design

The following will introduce each module and the overall running framework of the game according to the actual running order of the game, so that we can better grasp the development process of the game, and the following is the detailed running process.

(1) When you start the game, you will first enter the registration and login interface. Before that, you need to make sure that the server side and database side have been successfully deployed. When you start the game for the first time, you need to enter your personal information to register, after registration, close the registration interface pop-up window and enter your user name and password to enter the game successfully.

(2) Enter the main menu page of the game, in this page, the player can set the game properties, such as volume, mouse sensitivity and other options. After setting, click the "Start Game" button in the main menu, and enter the next scene after the game progress bar is loaded.

(3) game character role selection scene, in this scene, the player can click on the "+" icon for the creation of characters, this game has two characters, the player can view the character profile, carefully read the character's background story and check the character's strengths and weaknesses, according to the player's personal preferences to choose the right character for themselves, the choice is complete! After the selection is completed, enter the character's name and click the "Confirm" button.

(4) Once the above preparations are complete, you can successfully log in to the space station. Inside the space station, players can release skills according to the skill casting prompts at the bottom of the screen, and understand the way of attacking skills. In addition, the game contains preset shortcut keys, such as "O"

for the group request, "I" for the game shop, "T" for the mission status, "F" is to switch the alignment and so on.

(5) Open the team interface, you can create your own team, you can also view the created team, choose to join the team created by others, wait for the captain to agree to the request to join the team, the captain for the selection of copies, and follow the captain's actions to jump to different scenes, to achieve the function of the team to challenge the copies.

(6) There are 4 different NPCs in the space station, players can unlock the corresponding copy quests after dialogue with different NPCs according to the game plot, after accepting the quests, you can press the shortcut key "T" to view the quest list, and press "M" key to open the map in the trigger area of the map console, and select "M" key to open the map, and select "M" key to open the map. Press "M" in the trigger area of the map console to open the map and select the accepted copy quest to break through.

(7) There are 4 copies of the game, namely, the beginning of the journey, the recapture of the space station, the trial of destiny, and the final battle. Players must return to the space station after capturing the map console, dialogue with the corresponding NPC to accept the next mission, and then select the corresponding copy of the map console of the space station to continue to break through the game.

(8) There may be enemies with different attack methods in different locations of different copies. If necessary, players can use the equipment in their backpacks to resupply. Players need to use their own flexible positioning, good sense, tacit co-operation to defeat the enemy and achieve success in the game.

3.5. Database Design

The game includes a total of eight tables, including the character attribute information table, the backpack information table and the user information table. These nine tables include the data needed for this game. They provide functions such as user registration and logging in and initialising and caching character and backpack information.

4. Concrete implementation of functionality

This part describes the specific implementation methods of each functional module of the game mentioned in the previous chapter. Due to the space limitation, only the key function implementation methods and codes inside some key modules are given.

4.1. Implementation of UI framework module

The first part introduced is the implementation of UI framework module, which consists of core management, page control, UI framework startup script, etc. It is used to manage all the panels (e.g. login panel, main menu panel, etc.) and jump between control panels and panels.

4.1.1. Core Management Script UIManager.cs

This script is used to parse and save all panel information, create and save instances of all panels, manage and save all display panels. Parsing the Json file that stores the panels enables smooth switching of the UI interface and reduces lagging [4].

```
1 private static UIManager _instance;
2 public static UIManager Instance {
3       get {
4             if (_instance == null) {
```

```
5                    _instance = new UIManager();}
6                return _instance; }}
7    private Dictionary<UIPanelType,
8                        string> panelPathDict;
9 [Serializable]
10     class UIPanelTypeJson {
11           public List<UIPanelInfo> infoList;}
```

4.1.2. UI framework startup script GameScene2Root.cs

As a launcher for the UI framework, this script loads out the stored scripts for each interface, providing methods for moving interfaces in and out of the stack. Below is the code that provides the logic for panel display and popup.

```
1      if (UIManager.Instance.PanelCount > 0){
2        GamingPanel panel = UIManager.Instance.PeekPanel() as GamingPanel;
3        KnapsackPanel knapPanel = UIManager.Instance.PeekPanel() as KnapsackPanel;
4        InfoPanel infoPanel = UIManager.Instance.PeekPanel() as InfoPanel;
5        TeamPanel teamPanel = UIManager.Instance.PeekPanel() as TeamPanel;
6        AimPanel aimPanel = UIManager.Instance.PeekPanel() as AimPanel;
7        ChatPanel chatPanel = UIManager.Instance.PeekPanel() as ChatPanel;
8        MapPanel mapPanel = UIManager.Instance.PeekPanel() as MapPanel;
9        AimPanel aimPanel = UIManager.Instance.PeekPanel() as AimPanel;
10           if (panel == null && paramter.inputEsc){
11               UIManager.Instance.PopPanel();}
12           UIManager.Instance.PushPanel(UIPanelType.MapPrefabPanel) }
```

4.2. Implementation of Character Management Module

The following will introduce the key content of the project - character management module, this module basically implements the manipulation of the character, including the change of character animation, the release of skills and damage and other core gameplay. It mainly introduces the core script of character management and the script of skill base class.

4.2.1. Character management core script ThirdPlayerController.cs

This script is the core part of the mission control, used to achieve the character's movement and displacement control, state change, different states of the camera's perspective and walking sound effects added and so on.

First of all, it is the judgement of the character in different states, when the character receives the influence of different factors and changes the state, according to the changed state, judge the current conditions and make the action to be achieved in the state.

```
1 if (_dying)
2       return lastAnimState = PlayerState.Dying;
3 if (enterSkill||enterDamage) {
4       return lastSkillState;}
5 if(_mainCamera==null){
```

```
6       _mainCamera = GameObject.FindGameObject (Tags.MainCamera).transform;}
7 Vector3 wantedFwd = Vector3.Normalize(new Vector3
8 (_mainCamera.forward.x, 0, _mainCamera.forward.z));
9 _enterSkillCameraQuaternion = Quaternion.LookRotation(wantedFwd);
10      if(_takedamage){
11              enterDamage = true;
12              return PlayerState.Damage; }
```

This part of the code completes the method of modifying the character's state. It calculates and sets the angle of the camera when it releases a skill, and then returns the corresponding state according to the flag bits of each state and the player's input.

4.2.2. Skill base class script PropBase.cs

Skills have some public attributes, you can extract these public parts into a base class, whose main functions include the basic attributes of the skill, judging the damage range of the skill, judging whether the skill can be released and the method of releasing the skill.

```
1 public bool CanReleaseSkill(float currMagic){
2 return enable && Mathf.Approximately(ColdTime, 0) &&
3 (Mathf.Approximately(currMagic, magicLess) || currMagic > magicLess); }
4 public virtual void ReleaseSkill(GameObject prop, Transform player,
5 bool attackscene, ref float currMagic){
6    currMagic -= magicLess;
7    ColdTime = maxColdTime;
8    float propHurt = maxHurt;
9    if (attackscene){
10              if (isRectProp && rectArea != null){
11              foreach (GameObject enemyobject in TranscriptManager._instance.enemyList){
12              enemyobject.SendMessage("TakeDamage", propHurt); }}}
```

This script is for determining whether or not a skill can be released based on the cooldown time and Magic value. After releasing the skill, it will reduce the player's magic value, put the skill on cooldown and calculate the damage taken by the enemy. Algorithms for Rectangle, Umbrella and Explosion attack modes are implemented for damage judgement and detection [5].

4.3. Implementation of Enemy Logic Module

As a classic ARPG game, it is action and adventure fighting that go hand in hand. Therefore, the logic of enemy's behaviour is also a top priority. The following will introduce the implementation of the enemy logic module in detail, which mainly includes the implementation of the enemy base class, enemy trigger generation, enemy event management, and enemy bloodstripe script. Only the first two scripts are described here.

4.3.1. Enemy Base Class EnemyBase.cs

The following script introduces the enemy base class - FsmBase, which is created for all other monster attack scripts to inherit and call due to the many types of monsters in the game, which mainly includes the functions of blood level, attack power, attack range, focusing on the target, status detection, and death

sinking and so on. This part will be described in detail below.

```
1 public void CalcDistance(){
2   if (TranscriptManager._instance.player != null){
3   Transform player = TranscriptManager._instance..transform;
4   float tempDistance = Vector3.Distance(player.position, transform.position);
5   distance = tempDistance;
6   targetPlayer = TranscriptManager._instance.player; }
7   if (SyncPlayer._instance.playerTeamData.isLeader){
8       foreach (GameObject otherPlayer in SyncPlayer._instance.playerList){
9           Transform player = otherPlayer.transform;
10              float tempDistance = Vector3.Distance(player.position, transform.position);
11              if (tempDistance < distance){
12                  targetPlayer = otherPlayer; }}}}
```

This part of the code omits some of the definitions of enemy blood, speed, attack speed, attack range, initial distance from the target, animations, character flag bits, individual status flag bits, and synchronising enemy positions. It is mainly shown for getting the player position and calculating the distance between it and the player. If it is less than a certain distance, it will lock the player as the target, and if it is in group mode, it will lock the closest one to itself as the target.

4.3.2. Enemy Trigger Generation Script EnemyTrigger.cs

After entering the copy, different enemies will appear in different scenes, and the enemies will appear in different waves, and the number of enemies will be different each time. The enemies will appear in different waves, and the number of enemies will not be certain. So we created EnemyTrigger.cs to determine which kind of enemy should be generated in different copies and different locations.

```
1  CreateEnemyModel model = new CreateEnemyModel() { list = propertyList };
2  EventCenter.Broadcast(EventDefine.CreateEnemy, model);
3  while (TranscriptManager._instance.enemyList.Count > 0){
4      yield return null; }
5  BuildingController._instance.OpenScene2Door1(true);
6  while (!isTrigger){
7      yield return null;}
8  propertyList = new List<CreateEnemyProperty>();
9  AddEnemy(enemyPrefabs[0], enemyPos[2].position);
10     AddEnemy(enemyPrefabs[3], enemyPos[3].position);
11     model = new CreateEnemyModel() { list = propertyList };
12     EventCenter.Broadcast(EventDefine.CreateEnemy, model);
```

This part of the code is to add the corresponding location of the enemy prefabricated body and enemy generation position if it enters the trigger range of the enemy generation, send the delegate event to generate the enemy, and set whether it enters the trigger range or not to false after controlling the switch of the door.

4.4. Implementation of Network Management Module

Client communication script - PhotonEngine.cs, used to implement the communication service function of the game. The singleton pattern, which enables only one to exist in any scene and all interact to the server

side through it [18].

```
1 private void Awake() {
2    if (Instance == null){Instance = this;
3          DontDestroyOnLoad(gameObject);}
4    else if (Instance != this){ Destroy(gameObject); }
5    requestDict = new Dictionary<OperationCode, Request>();
6    eventDict = new Dictionary<EventCode, EventBase>();}
7 private void Start() {
8    peer = new PhotonPeer(this, ConnectionProtocol.Udp);
9    peer.Connect("192.168.43.153:5055", "PosSynServer");}
10      private void Update() {peer?.Service();}
```

This part of the code is to initialise the PotonEngine and ensure that only one exists. If the scene is changed and generated again, the newly generated one is destroyed, if the initialisation is successful then it is guaranteed not to be destroyed, if it already exists then it destroys the one that is generated again. Get the event information and request information in the dictionary. And receive the response from the server, and judge the status of the server and keep the server connected.

4.5. Implementation of shader development

The following section will introduce the shader module, the screen embellishment part of the project. The shader can complete the accurate calculation of real-time lighting, process a large amount of 3D data [6], etc. The most used in this game are the effects of occlusion culling, Gaussian blurring, and other effects to optimise the screen.

The following highlights the shader occlusion culling script - SeeThroughWall, there are a variety of obstacles in the game, when the manipulated character walks to the obstacles will be occluded, affecting the player's game experience, so the use of this shader script, can achieve the character is occluded after the silhouette of the display.

```
1        Properties{
2        _Color("Color", Color) = (1,1,1,1)
3        _MainTex("Albedo (RGB)", 2D) = "white" {}
4        _Glossiness("Smoothness", Range(0,1)) = 0.5
5        _Metallic("Metallic", Range(0,1)) = 0.0
6        _OcclusionColor("OcclusionColor",COLOR) = (1,1,1,0.5) }
7       Pass{
8             ZWrite off
9             Lighting off
10                Ztest Greater
11                Blend SrcAlpha OneMinusSrcAlpha
12                CGPROGRAM
```

The first part of the script is the attribute list, which is initialised to define the attribute list, the model's main colour, the main texture, glossiness, metallic sheen, etc., respectively, to facilitate the modification of the shader's attributes as needed. Sub-shaders, setting labels, turning off depth writing, turning off light testing, and turning on depth blending, are necessary prerequisites for implementing occlusion culling [10].

5. Conclusion.

In this project, we have designed and developed a 3D game, Magic Park, based on Unity engine using C# as scripting language and MySQL as database. The game presents a full 3D cool sci-fi space scene, where users can manipulate their favourite characters to defeat alien invaders, complete battles, and guard the capsule's safety in a space capsule full of technology and sci-fi. Users can control their favourite characters by manipulating the mouse and keyboard. Register and login to enter the game, create and manipulate the character to dialogue with NPCs to accept the mission, enter different levels, release skills to defeat the enemy, and complete the corresponding tasks in order to obtain the rewards of the copy mission. The use of shaders to beautify the scene and characters, add background music, game effects, etc. will greatly enhance the game's cool graphics, and bring players a good gaming experience. The game has a complete process, unique art style, so that the user in the intense and exciting 3D game at the same time, more deeply appreciate the charm of 3D games.

References

[1] Wu Yafeng, Xu Kaixin, Su Yaguang. Unity Game Development Technology and Typical Cases. Beijing: People's Posts and Telecommunications Publishing House, Beijing, China. 2019: 156-164.

[2] Jin Xizeng. Unity 3D\2D mobile game development. Beijing: Tsinghua University Press. 2019:88-112.

[3] Xuan Yusong. Unity 3D Game Development. Beijing: People's Posts and Telecommunications Publishing House, 2018:53-65.

[4] Lu Feng, He Yunfeng. Fundamentals of Computer Graphics (3rd Edition). Beijing: Electronic Industry Press. 2019:223-226.

[5] Chen, Jia-Dong. Unity3D script programming. Beijing: Electronic Industry Publishing House, Beijing, China. 2017:40-45.

[6] Zhang Yao. Unity 3D from Beginning to Hands-on. Beijing: China Water Conservancy and Hydropower Publishing House, Beijing, China. 2022:103-120.

[7] Frahaan Hussain. Learn OpenGL. Birmingham: Packt Publishing, 2018:88-95.

[8] Chintan Mehta, Ankit K Bhavsar. MySQL 8 Administrator’s Guide. Birmingham: Packt Publishing, 2018:132-150.

[9] Chen, Jia-Dong. Unity 3D Scripting - Developing Cross-Platform Games with C#. Beijing: Electronic Industry Publishing House, Beijing, China 2016:80-96.

[10] Kyle Halladay. Shader Development in Action. Beijing: Tsinghua University Press. 2021:67-74.