УДК 004

Разработка сетевой игры с генерацией боя в Unity 3D с использованием плагина photon

Меркулов Андрей Сергеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Целью исследования является разработка сетевой игры с генерацией боя, используя плагин photon для Unity 3D. Для выполнения задачи выбран язык программирования C#. Практическая значимость работы заключается в создании сетевой игры с генерацией сражения.
Ключевые слова: информационные технологии, сеть, сервер, unity, photon.

Development of a network game with the generation of combat in Unity3D using the photon plugin

Merkulov Andrey Sergeevich
Sholom-Aleichem Priamursky State University
Student

Abstract
The purpose of the study is to develop a network game with the generation of combat using the photon plugin for Unity3D. The C# programming language is selected to perform the task. The practical significance of the work is to create a network game with battle generation.

1 Введение

1.1 Актуальность

В гейм-журналистике, да и в российском игровом сообществе достаточно редко рассматриваются вопросы, посвященные осмыслению места и роли геймдевелопмента среди других сфер человеческой деятельности. Сравнительно недавно на была опубликована статья П.Прохоренко "Авторский геймдизайн", в которой были затронуты весьма существенные аспекты по упомянутой теме. Сам факт, отражающий хотя бы начало обсуждения места и роли геймдевелопмента в современном мире, безусловно, отраден. В целом же просматривается отсутствие каких-либо устоявшихся мнений о том, что такое "игроделание", в сравнении с близкими к нему сферами человеческой деятельности. А, следовательно, каковы те ориентиры, которыми в перспективе могли бы руководствоваться "режиссеры" новых игр.

1.2 Обзор исследований

Н.Л. Горбач, Т.М. Кальченко в статье рассмотрели некоторые ролевые игры онлайн, оказывающие положительное влияние на изучение иностранного языка. [1] А.Л. Марченко описал синтаксис языка программирования C# 2.0.NET и множества классов, применяемых для разработки приложений на платформе.NET [2]. Н.В. Кучин, К.О. Поляков, В.А. Щербаков рассмотрели подход к моделированию затухания звука в различных средах, параметры которых можно самостоятельно редактировать или проводить исследование новых материалов [3] Н.А. Иванова привела, что психологические особенности геймеров (игроков в компьютерные игры) - тема чрезвычайно актуальная сегодня, поскольку распространенность геймерства очень широка и продолжает расти, охватывая все поколения людей. [4] К.А. Кардашевская исследовала проблемы виртуальных объектов, возникающих из многопользовательских онлайн-игр [5].

1.3 Цель исследования

Целью исследования является разработка сетевой игры с генерацией боя.

2 Материалы и методы

Для разработки игры был использован язык программирования C# и среда Unity.

3 Результаты исследования

В данной работе показан процесс создания многопользовательской игры.

3.1 Создания лобби

Для создания лобби необходимо создать скрипт, который будет контролировать все функции lobby (создания комнаты, подключения к существующей, задания ника) Для работы с Photon необходимо подключить библиотеку Photon.pun и заменить стандартный класс uniy MonoBehaviour на MonoBehaviourPunCallbacks (рис.1).

```
using Photon.Pun;
using UnityEngine.UI;

Скрипт Unity (1 ссылка на ресурсы) | Ссылок: 0
public class LoboManager : MonoBehaviourPunCallbacks
{
```

Рисунок 1. Подключения библиотеки

Функция OnConnectedToMaster() отвечает за подключения к серверу (рис.2).

```
public override void OnConnectedToMaster()
{
    //Log("connected to master");
    panel.SetActive(false);
}
```

Рисунок 2. Подключения к серверу

Ниже приведены две функции: joinRoom (рис.3) подключения к комнате и CreateRoom создания комнаты.

```
public void CreateRoom()
{
    PhotonNetwork.NickName = NicknameInput.text;
    PlayerPrefs.SetString("NickName", NicknameInput.text);
    PhotonNetwork.CreateRoom(null, new Photon.Realtime.RoomOptions { MaxPlayers = 20, CleanupCacheOnLeave = false });
}

public void joinRoom()
{
    PhotonNetwork.NickName = NicknameInput.text;
    PlayerPrefs.SetString("NickName", NicknameInput.text);
    PhotonNetwork.JoinRandomRoom();
}
```

Рисунок 3. Подключения к комнате

Функция OnJoinedRoom подключает игрока, создавшего комнату (рис.4).

```
public override void OnJoinedRoom()
{
    //Log("joined the room");

    PhotonNetwork.LoadLevel("Game");
}
```

Рисунок 4. Перенос игрока в комнату

3.2 Создания игрового персонажа

Персонаж представляет собой спрайт, управляемый скриптами. В данном случае на Player находится два скрипта playerControl (рис.5).

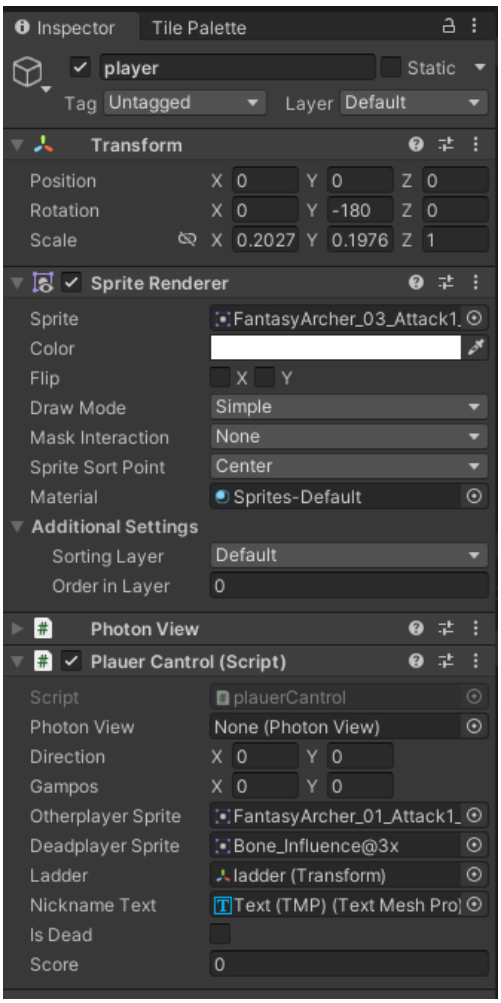

Рисунок 5. Настройка плеера

Скрипт game Manager отвечает за подключения и отключения игроков от комнаты. При старте игрок создается в случайной точке (рис.6).

```
Vector3 Pos = new Vector3(UnityEngine.Random.Range(1, 15), UnityEngine.Random.Range(1, 5));
PhotonNetwork.Instantiate(playrPrefab.name, Pos, Quaternion.identity);
```

Рисунок 6. Создания игрока

В скрипте playerControl задается направления движение игрока (рис.7), а в скрипте mapController реализуется само движение (рис.8).

```
private void HandleInput()
{
    //Direction = Vector2Int.zero;
    if (Input.GetKey(KeyCode.LeftArrow)) Direction = Vector2Int.left;
    if (Input.GetKey(KeyCode.RightArrow)) Direction = Vector2Int.right;
    if (Input.GetKey(KeyCode.UpArrow)) Direction = Vector2Int.up;
    if (Input.GetKey(KeyCode.DownArrow)) Direction = Vector2Int.down;

    if(Input.GetMouseButtonDown(0))
    {
        touchstarted = Camera.main.ScreenToWorldPoint(Input.mousePosition);
    }
    else if(Input.GetMouseButtonUp(0))
    {
        Vector2 touchEnded = Camera.main.ScreenToWorldPoint(Input.mousePosition);
        Vector2 swipe = touchEnded - touchstarted;

        if(swipe.magnitude > 2)
        {
            if (Mathf.Abs(swipe.x) > Mathf.Abs(swipe.y))
            {
                if (swipe.x > 0) Direction = Vector2Int.right;
                else Direction = Vector2Int.left;
            }
            else
            {
                if (swipe.y > 0) Direction = Vector2Int.up;
                else Direction = Vector2Int.down;
            }
        }
    }
}
```

Рисунок 7. Выбор направления движения игрока

```
private void Update()
{
    if (PhotonNetwork.Time > lastTicKTime + 1 &&
        PhotonNetwork.IsMasterClient &&
        PhotonNetwork.CurrentRoom.PlayerCount >= 1)
    {
        Vector2Int[] directions = plauers
            .Where(p => !p.IsDead)
            .OrderBy(p => p.PhotonView.Owner.ActorNumber)
            .Select(p => p.Direction)
            .ToArray();

        RaiseEventOptions options = new RaiseEventOptions { Receivers = ReceiverGroup.Others };
        SendOptions sendOptions = new SendOptions { Reliability = true };
        PhotonNetwork.RaiseEvent(42, directions, options, sendOptions);

        PerformTick(directions);
    }

    for (int i = 0; i < plauers.Count; i++)
    {
        if (plauers[i].IsDead == true)
        {
            GameManager.Leave();
        }
    }
}
```

Рисунок 8. Движение игрока

При старте создается сетка, на которой в дальнейшем будет генерироваться локация (рис.9).

```
private void Start()
{
    cells = new bool[constante.FieldwIDTH, constante.FieldwIHEIGNT];

    for (int x = 0; x < cells.GetLength(0); x++)
    {
        for (int y = 0; y < cells.GetLength(1); y++)
        {
            Tilemap.SetTile(new Vector3Int(x, y, 0), cel);
            SetCell(new Vector2Int(x, y), true);
        }
    }

    for (int x = 0; x < cells.GetLength(0); x++)
    {
        Tilemap.SetTile(new Vector3Int(x, -1, 0), BedtocTile);
        Tilemap.SetTile(new Vector3Int(x, cells.GetLength(1), 0), BedtocTile);

    }

    for (int y = 0; y < cells.GetLength(1); y++)
    {
        Tilemap.SetTile(new Vector3Int(-1, y, 0), BedtocTile);
        Tilemap.SetTile(new Vector3Int(cells.GetLength(0), y, 0), BedtocTile);

    }
}
```

Рисунок 9. Создание сетки

После создания сетка заполняется тайлами разного типа (рис.10).

```
public void SetCell(Vector2Int pos, bool set)
{
    cells[pos.x, pos.y] = set;
    TileBase cell;

    if (!set)
    {
        cell = null;
    }
    else if (pos.y < 9)
    {
        cell = StoneCelTile;
    }
    else if (pos.y < 11)
    {
        cell = Random.value > 0.5f ? StoneCelTile : DirtCeilTile;
    }
    else if (pos.y < 19)
    {
        cell = DirtCeilTile;
    }
    else if (pos.y < 21)
    {
        cell = Random.value > 0.5f ? GrassCellTile : DirtCeilTile;
    }
    else
    {
        cell = GrassCellTile;
    }

    Tilemap.SetTile((Vector3Int)pos, cell);
}
```

Рисунок 10. Заполнение сетки

При копании вверх под игроком создаётся лестница (рис.11).

```
Ссылок: 1
public void SetLadderLength(int length)
{
    for (int i = 0; i < Ladder.childCount; i++)
    {
        Ladder.GetChild(i).gameObject.SetActive(i < length);
    }

    while (Ladder.childCount < length)
    {
        Transform lastTile = Ladder.GetChild(Ladder.childCount - 1);
        Instantiate(lastTile, lastTile.position + Vector3.down, Quaternion.identity, Ladder);
    }
}
```

Рисунок 11. Создание лестницы

Если игрок А проходит под лестницей игрока В он проигрывает (рис.12).

```
Ссылок: 1
private void MinePlayerBlock(plauerCantrol player)
{
    if (player.Direction == Vector2Int.zero) return;

    Vector2Int targetposition = player.gampos + player.Direction;

    if (targetposition.x < 0) return;
    if (targetposition.y < 0) return;
    if (targetposition.x >= cells.GetLength(0)) return;
    if (targetposition.y >= cells.GetLength(1)) return;

    if (cells[targetposition.x, targetposition.y])
    {
        SetCell(targetposition, false);
        player.Score++;
    }


    Vector2Int pos = targetposition;
    plauerCantrol mineplauer = plauers.First(p => p.PhotonView.IsMine);

    if (mineplauer != player)
    {
        while (pos.y < cells.GetLength(1) && !cells[pos.x, pos.y])
        {
            if (pos == mineplauer.gampos)
            {
                PhotonNetwork.LeaveRoom();
                break;
            }
            pos.y++;
        }
    }
}
```

Рисунок 12. Удаления игрока

3.3 Работа программы

При запуске игры появляется меню с полем для ввода ника и кнопками, при нажатии на которые или создаться комната, или осуществляется перенос (рис.13).

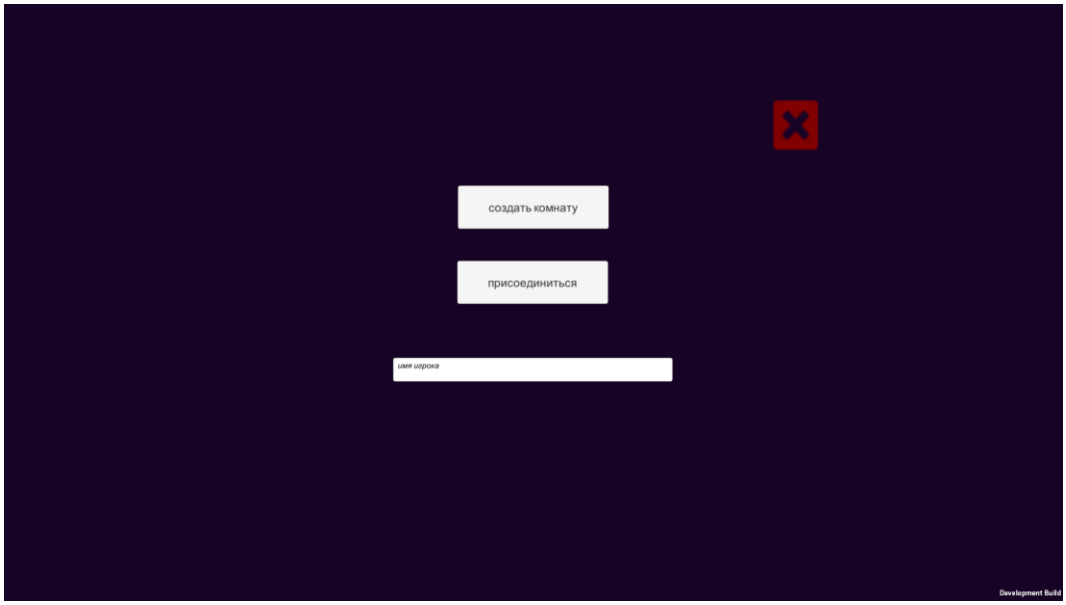

Рисунок 13. Главное меню

После нажатие Кнопки создания комнаты, игрока переносит в сгенерированную комнату, управление осуществляется нажатием на стрелки (рис.14).

Рисунок 14. Игровой процесс

4 Выводы

В результате данной работы была разработана многопользовательская онлайн игра в среде разработки Unity с использованием плагина photon. Также были рассмотрены основы работы с данным плагином.

Библиографический список

1. Горбач Н.Л., Кальченко Т.М. Использование многопользовательских ролевых онлайн игр в обучении английскому языку //Актуальные научные исследования в современном мире. 2020. № 6-7 (62). С. 70-73.
2. Марченко А.Л. Основы программирования на C# 2.0 // Интернет-Университет Информационных Технологий (ИНТУИТ). 2017. С. 552.
3. Кучин Н.В., Поляков К.О, Щербаков В.А. Имитационное моделирование затухания звука в различных средах на языке C#//В сборнике: Радиоэлектронные устройства и системы для инфокоммуникационных технологий. Доклады Всероссийской конференции (с международным участием). Сер. "Научные конференции, посвященные дню Радио" 2019. С. 317-321.
4. Иванова Н.А. Эксплицитные мотивы в повседневной жизни у мужчин, играющих в компьютерные онлайн-игры // Эмпирическое исследование Национальный психологический журнал. 2018. № 4 (32). С. 16-26.
5. Кардашевская К.А. Правовое регулирование виртуальных объектов из многопользовательских онлайн-игр //Политика, экономика и социальная сфера: проблемы взаимодействия. 2015. № 1. С. 17-21.