УДК 004

Создание игры «tiles hop» на unity engine

Вихляев Дмитрий Романович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассматривается создание 3d игры «tiles hop» на игровом движке unity. Для реализации будут использоваться готовые объекты среды unity и скрипты написанные на языке программирования C#. В результате исследования будет приведён пример создания игры с подробным описанием её реализации.
Ключевые слова: unity, C#, игра.

Creating a «tiles hop» game on unity engine

Vikhlyaev Dmitry Romanovich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses the creation of a 3d game "tiles hop" on the unity game engine. Ready-made unity environment objects and scripts written in the C# programming language will be used for implementation. As a result of the research, an example of creating a game with a detailed description of its implementation will be given.
Keywords: unity, C#, game.

1 Введение

1.1 Актуальность

В наше время компьютерные игры являются частью общественных и культурных сфер, таких как искусство, образование, и даже спорт стал, связан с играми в объединенном названии киберспорт. Игровая индустрия тесно связана с индустрией компьютеров и даже имеет для неё большое значение. Игровой движок Unity позволяет делать кроссплатформенные игры с использованием новейших технологий. Огромное количество игровых компаний пользуются этим движком при создании своих проектов. На нем, возможно, создать игру, способную работать на более чем двадцати различных операционных систем, в которые входят персональные компьютеры, игровые консоли (PlayStation, Xbox и т.д.), мобильные устройства, интернет-приложения и другие.

1.2Обзор исследований

А.М.Васильев иследовал создание компьютерной игры на unity 3d [1]. В.И.Вахрушев реализовал игру-головоломку "расстановка мебели" с использованием unity и языка C# [2]. А.А.Новичков описал среду разработки "unity 3d" для создания компьютерных игр [3]. Д.С.Романов разработал мультиплеерноую игру на платформе unity 3d. [4]. И.В.Бахтин создал игровое приложение с использованием среды разработки игр unity [5].

1.3Цель исследования

Цель исследования – используя среду 3D моделирования игр unity engine, создать игру «tiles hop» со всеми необходимыми для полноценной игры, механиками.

2 Материалы и методы

Для реализации используются игровой движок unity, язык программирования C#, IDE MonoDevelop.

3 Результаты и обсуждения

Цель игры «tiles hop» заключается в том, чтобы игровой персонаж, которым является «шар» достиг финиша, прыгая по плитам, если шар недолетает до плиты, он падает и игра заканчивается.

В начальное состояние игры шар находится на неподвижной платформе. В момент старта он начинает двигаться вперёд без остановки. После того как платформа заканчивается, появляются небольшие плиты, которые двигаются по горизонтали не зависимо друг от друга.

Управление персонажем происходит за счёт нажатия клавиш на клавиатуре. Над шаром можно воспроизвести три события, из которых два сменяют положение вправо и влево, а третий реализует прыжок.

В качестве пространства под плитами послужит стандартный объект unity «terrain». Установив ему текстуру схожую с водой, действия игрового мира станут происходить, словно в океане. Для этого нужно зайти в меню inpector->terrain (script)->edit textures->add texture и выбрать оттуда текстуру «water fallback» (рис. 1).

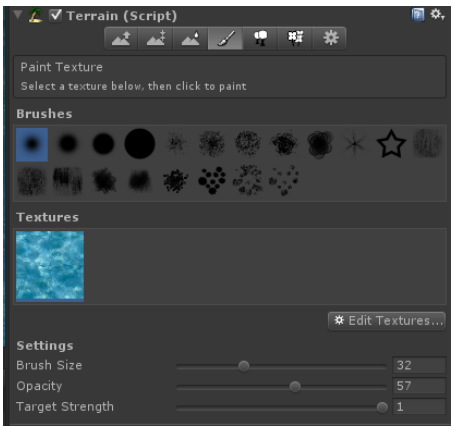

Рис. 1. Добавление текстуры объекту «terrain»

Установив текстуру земли на плиты, игра приобретет смысл, что игровой персонаж может прыгать по маленьким островкам, а если упадёт в воду, игра закончится (рис. 2).

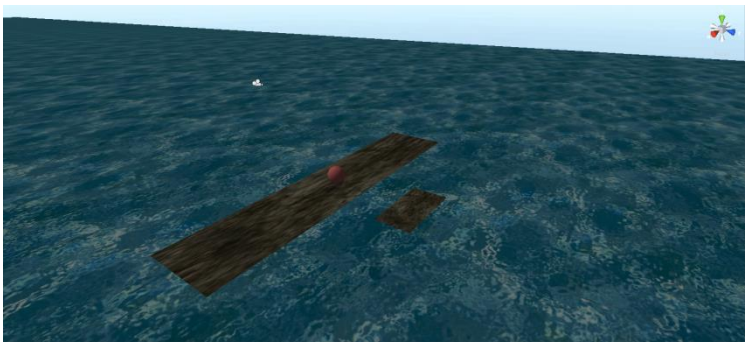

Рис. 2. Начальное состояние игры

Плиты двигаются по горизонтали в определённых границах, если плита достигает одной из границ, то начинает двигаться в противоположную сторону. В зависимости от того места где плита образовалась, она начинает движение либо в право либо влево. Таким способом достигается независимость движения одной плиты от другой. За изменения положения в пространстве отвечает метод «transform.position», к которому прибавляются значения типа «Vector3» по заданной оси координат (рис. 3).

```
using UnityEngine;
using System.Collections;

public class moveplet : MonoBehaviour {

    public int max_sdvig=6;
    public bool left=false;
    public bool right=true;
    public float speed=2;
    void Start () {
        if(transform.position.x<=0)
        {
            left=true;
            right=false;
        }
    }
    void Update () {

        if(transform.position.x > max_sdvig){
            left=true;
            right=false;
        }
        else if(transform.position.x < -max_sdvig){
            left=false;
            right=true;
        }
        if(right){
            transform.position += Vector3.right * Time.deltaTime*speed;

        }
        else if(left)
        {
            transform.position += Vector3.left * Time.deltaTime*speed;

        }
    }
}
```

Рис. 3. Движение плиты по горизонтали

Для того чтобы процессор всё время не занимался движением всех плит в игре, создаётся генератор плит, который будет по мере необходимости создавать новые плиты и удалять старые. Для этого понадобится создать переменные объекта плиты, которая будет создавать её копии, позиции игрока, с помощью которой можно будет определить дошёл ли игрок до той точки, когда нужно создавать новые плиты и удалять старые.

Впереди игрока всегда будут пять плит, которые будут создаваться в разных координатах по оси их движения. Создать копии объекта можно с помощью метода «(GameObject)Instantiate», параметрами которого являются копируемый объект, координаты положения и угол вращения. Когда у игрока координаты по оси его движения будут больше чем координаты одной из плит, она уничтожится при помощи метода «Destroy», а впереди появится новая (рис. 4).

```
public class TileGenerator : MonoBehaviour {

    public GameObject tilePrefabs;
    private List<GameObject> activeTiles=new List<GameObject>();
    private float spawnPos=2;
    private float tileLength=0.4f;
    private int rastoyanie = 4;
    [SerializeField] private Transform player;
    private int startTiles=6;

    void Start () {
        for(int i=0; i<startTiles; i++)
        {
            var itog=Random.Range(0,5);
            switch(itog)
            {
                case 0:
                    SpawnTile(0);
                    break;
                case 1:
                    SpawnTile(4);
                    break;
                case 2:
                    SpawnTile(-8);
                    break;
                case 3:
                    SpawnTile(-4);
                    break;
                case 4:
                    SpawnTile(8);
                    break;
            }
        }
    }
    void Update () {

        if(player.position.z>spawnPos-(startTiles*(tileLength+rastoyanie-1)))
        {
            SpawnTile(Random.Range(-6,6));
            DeleteTiles();
        }
    }
    private void SpawnTile(int tileIndex)
    {
        GameObject nextTile = (GameObject)Instantiate(tilePrefabs, transform.forward*spawnPos+transform.right*tileIndex, transform.rotation);
        activeTiles.Add(nextTile);
        spawnPos+=rastoyanie;
    }
    private void DeleteTiles()
    {
        Destroy(activeTiles[0]);
        activeTiles.RemoveAt(0);
    }
}
```

Рис. 4. Генерация и уничтожение плит

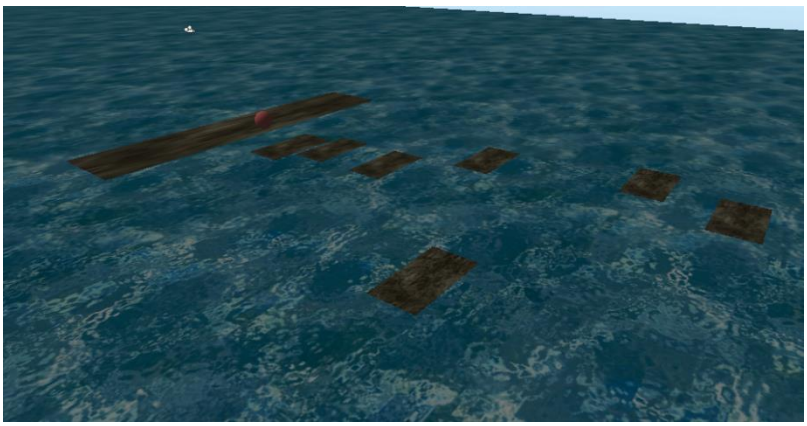

Рис. 5. Реализация создания плит

Чтобы шар обладал физическими свойствами ему нужно добавить компонент «Character Controller» после этого он сможет контактировать с плитами. Для установления события при нажатии клавиш используется метод «Input.GetKey», параметром которого является символьный код клавиши. Необходимо установив значение гравитации, при котором объект не сможет бесконечно подниматься вверх при прыжке. Так же нужно определить свойство заземленности с помощью метода «controller.isGrounded», при котором шар сможет прыгать. Тогда как если шар находится в воздухе, то прыжок работать не будет. (рис. 6,7).

```
public class PlayerController : MonoBehaviour
{
    private CharacterController controller;
    private Vector3 dir;
    [SerializeField] private int speed;
    [SerializeField] private float jumpForce;
    [SerializeField] private float gravity;

    void Start()
    {
        controller = GetComponent<CharacterController>();
    }

    private void Update()
    {
        if (Input.GetKey("right"))
        {
            right();
        }

        if (Input.GetKey("left"))
        {
            left();
        }

        if (Input.GetKey("up"))
        {
            if (controller.isGrounded)
                Jump();
        }
    }
    private void right()
    {
        transform.position+=Vector3.right * 0.2f;
    }
    private void left()
    {
        transform.position+=Vector3.left * 0.2f;
    }
    private void Jump()
    {
        dir.y = jumpForce;
    }
    void FixedUpdate()
    {
        dir.z = speed;
        dir.y += gravity * Time.fixedDeltaTime;
        controller.Move(dir * Time.fixedDeltaTime);
    }
}
```

Рис. 6. Программа для управления игровым персонажем

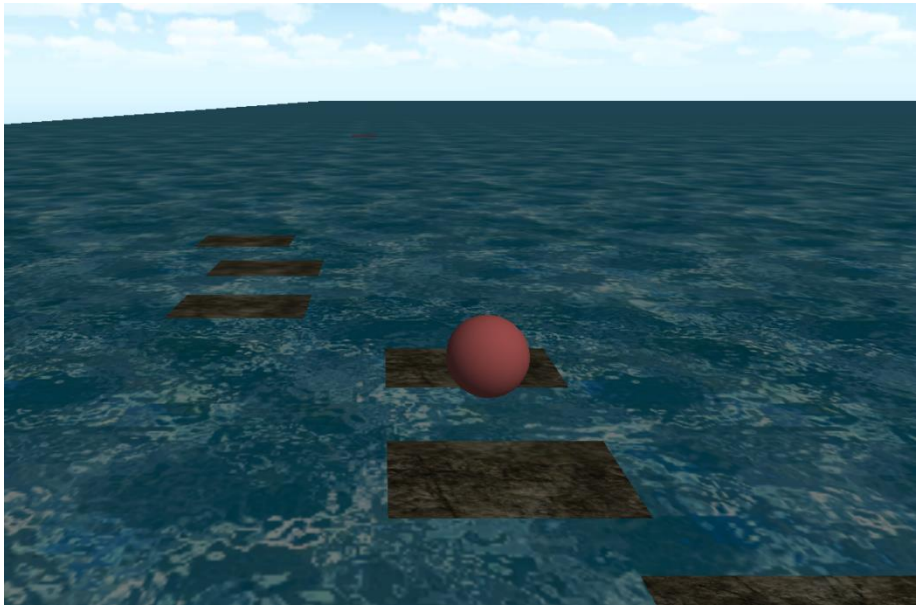

Рис. 7. Реализация управления персонажем

Далее необходимо чтобы камера следовала за персонажем. Понадобится переменная, указывающая на позицию игрового персонажа. Затем получив разницу положения игрока и камеры на старте игры, на каждом кадре будет устанавливаться новое положение камеры по оси движения персонажа (рис. 8).

```
public class CameraController : MonoBehaviour
{
    [SerializeField] private Transform player;
    private Vector3 offset;

    void Start()
    {
        offset = transform.position - player.position;
    }

    void FixedUpdate()
    {
        Vector3 newPosition = new Vector3(transform.position.x, transform.position.y, offset.z + player.position.z);
        transform.position = newPosition;
    }
}
```

Рис. 8. Слежение камеры за игровым персонажем

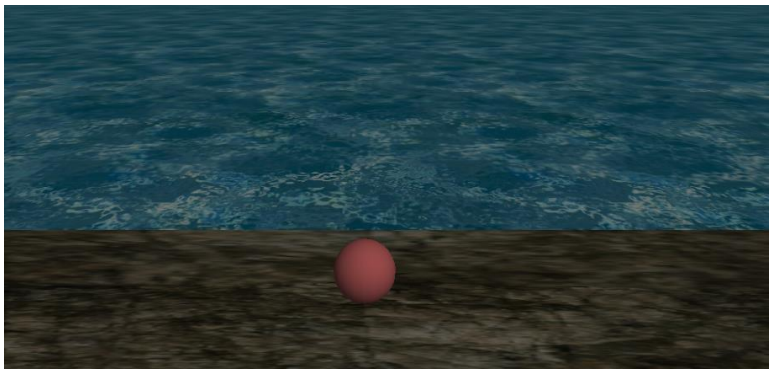

Рис. 9. Расположение камеры на старте игры

Чтобы игру можно было закончить нужно создать триггер, который переключится на другую сцену. Триггер устанавливается в окне «inspector» -> «“объект”collider»->is Trigger (рис. 10).

Рис. 10. Установка триггера

Метод «OnTriggerEnter» определяет столкновение триггера с объектом, если игрок с указанным тегом сталкивается с триггером, то происходит смена сцен с помощью метода «Application.loadedLevel». Указав значением текущей сцены следующей по порядку (рис. 11).

```
public class Finish : MonoBehaviour
{
    void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Player")
            if (Application.loadedLevel + 1 != Application.levelCount)
                Application.LoadLevel(Application.loadedLevel + 1);
            else
                Application.LoadLevel(0);
    }
}
```

Рис. 11. Смена сцены

Указать сцены для хранения можно в поле «scenes in built» нажав клавиши «ctrl+shift+B» (рис. 12).

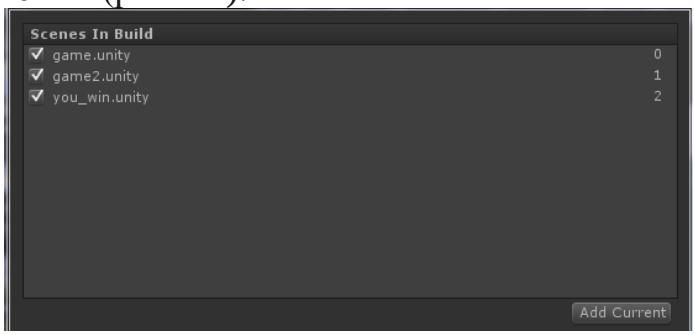

Рис. 12. Установка порядка и хранения сцен

В конце игры игрока уведомляют о прохождении (рис. 13).

Рис. 13. Переход на сцену с информацией о прохождении игры

Таким образом, была описана реализация программы для создания игры «tiles hop» на игровом движке unity с использованием скриптов языка программирования C#.

Библиографический список

1. Васильев А.М. Создание компьютерной игры на unity 3d// Заметки ученого. 2021. № 12-1. С. 52-54.
2. Вахрушев В.И. Реализация игры-головоломки "расстановка мебели" с использованием unity и языка C#// В сборнике: Развитие современной науки: теоретические и прикладные аспекты. сборник статей студентов, магистрантов, аспирантов, молодых ученых и преподавателей. Под общей редакцией Т.М. Сигитова. Пермь, 2016. С. 23-25.
3. Новичков А.А. Описание среды разработки "unity 3d" для создания компьютерных игр// В сборнике: XIX Ломоносовские чтения. Материалы Всероссийской научной конференции. Северный (Арктический) федеральный университет им. М.В. Ломоносова, Филиал в Коряжме; Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского, Арзамасский филиал. 2017. С. 53-55.
4. Романов Д.С. Разработка мультиплеерной игры на платформе unity 3d// Научный журнал. 2018. № 6 (29). С. 30-35.
5. Бахтин И.В. Создание игрового приложения с использованием среды разработки игр unity// Форум молодых ученых. 2019. № 2 (30). С. 250-267.
6. Попов А.А., Горшков Е.В., Асроров А.З. Разработка проекта компьютерной игры в среде unity// Интерэкспо Гео-Сибирь. 2018. Т. 2. № 8. С. 31-32.
7. Васильев В.И. Создание игрового приложения в среде разработки игр unity// Форум молодых ученых. 2019. № 2 (30). С. 378-391.
8. Семенцова Т.Ю. Разработка детских игр на движке unity// В сборнике: XLVII итоговая студенческая научная конференция УдГУ. Материалы всероссийской конференции. Ответственный редактор А.М. Макаров. 2019. С. 25-26.