УДК 004.93.1

СОЗДАНИЕ КАМЕРЫ ИГРОВОГО СИМУЛЯТОРА

Дубок Д.А.,
Научный руководитель – Разоренов Н.А., к.т.н., доцент

Скрипты дадут возможность управления компонентами и помогут выстроить игровую логику и механику. В Unity скрипты можно создавать на языках C# и JavaScript, что при использовании C# позволяет создавать их по всем правилам ООП, однако будет затрачено некоторое время на изучение классов и методов, используемых при создании скриптов и обращении к различным объектам на сцене[1]. JavaScript на фоне C# используется в Unity заметно реже, хотя это обусловлено предпочтениями разработчиков, нежели другими особенностями.

Хоть это и может показаться весьма затратным по времени, не стоит забывать о том, что Unity имеет весьма подробную документацию, что упростит изучение. В дополнение к этому Unity можно расширить с помошью дополнительного модуля JustLogic, который позволяет создавать логику приложения через инспектор объекта, а не с помощью написания скриптов на выбранном ЯП.

Для получения игрового приложения необходимо создать систему управления персонажем, пользовательский интерфейс, систему учета времени, а также камеру.

Для создания камеры от третьего лица нужно создать несколько объектов: объект, который служит точкой фокуса взгляда (размещается на персонаже) и опора камеры, которая будет являться родительским объектом камере и, соответственно, будет служить точкой с координатами 0,0,0 для дочерней камеры. Иерархия приведена на рисунке 1.

Рисунок 1 – Иерархия объектов, связанных с камерой

Camera Holder содержит в себе объект Pivot (точка фокуса, расположенная на персонаже), который содержит камеру, размещенную на некотором расстоянии от неё и объект TargetLook (направление взгляда персонажа при вращении камеры вокруг него). Скрипт управления камерой будет прикреплен к Camera Holder.

В качестве основных объектов, с которыми придется работать в скрипте, будут камера, Pivot, персонаж на сцене, Camera Holder и файл конфигурации камеры Camera Config, который содержит такие параметры,

как скорость вращения, граничные значения угла камеры по оси Y (вверх-вниз), значения скорости поворота камеры по осям и другие.

Для создания подобного файла конфигурации необходимо создать скрипт CameraConfig и указать перед объявлением класса CameraConfig строку [CreateAssetMenu(menuName = “Camera/Config”)], а после обявления класса – все необходимые параметры. После этого можно вызвать контекстное меню нажатием правой кнопки мыши и выбрать Create – Camera – Config, а затем установить нужные параметры, которые будут использоваться в скрипте камеры.Подобный подход задания параметров камеры позволяет производить отладку перемещения камеры без остановки отладки всего приложения и сразу применять новые значения [2].

Скрипт можно разделить на несколько частей.

Первая: установка позиции камеры. За это отвечает функция HandlePosition, в которой происходит получение параметров из файла кофигурации камеры и происходит интерполирование между исходной и измененной позицией Pivot и камерой для плавного перемещения, так как эта функций вызывается в функции FixedTick, которая в свою очередь вызывается в функции Update постоянно через определенный интервал. Код функции HandlePosition приведен на рисунке 2.

```
void HandlePosition()
{
    float targetX = cameraConfig.normalX;
    float targetY = cameraConfig.normalY;
    float targetZ = cameraConfig.normalZ;

    Vector3 newPivotPosition = pivot.localPosition;
    newPivotPosition.x = targetX;
    newPivotPosition.y = targetY;

    Vector3 newCameraPosition = camTrans.localPosition;
    newCameraPosition.z = targetZ;

    float t = delta * cameraConfig.pivotSpeed;
    pivot.localPosition = Vector3.Lerp(pivot.localPosition, newPivotPosition, t);
    camTrans.localPosition = Vector3.Lerp(camTrans.localPosition, newCameraPosition, t);
}
```

Рисунок 2 – Функция HandlePosition для управления перемещения камеры

Вторая часть содержимого файла скрипта – функция HandleRotation, которая отвечает за вращение камеры по осям. Код функции приведен на рисунке 3.

```
void HandleRotation()
{
    mouseX = Input.GetAxis("Mouse X");
    mouseY = Input.GetAxis("Mouse Y");

    smoothX = mouseX;
     smoothY = mouseY;

    lookAngle += smoothX * cameraConfig.Y_rot_speed;
    Quaternion targetRot = Quaternion.Euler(0,lookAngle,0);
    mTransform.rotation = targetRot;

    titAngle -= smoothY * cameraConfig.Y_rot_speed;
    titAngle = Mathf.Clamp(titAngle, cameraConfig.minAngle, cameraConfig.maxAngle);
    pivot.localRotation = Quaternion.Euler(titAngle, 0, 0);
}
```

Рисунок 3 – Функция HandlePosition для управления вращения камеры

Движение в сторону взгляда осуществляется с помощью ранее рассмотренной функции перемещения и функции FixedTick, которая также упоминалась ранее. Поворот происходит благодаря интерполяции вектора направления движения и вектора взгляда камеры. Реализация приведена ниже.

```
void FixedTick()
{
    delta = Time.deltaTime;

    HandlePosition();
    HandleRotation();

    Vector3 targetPosition = Vector3.Lerp(mTransform.position, Character.position,1);
    mTransform.position = targetPosition;
}
```

Для того, чтобы эти функции вызывались постоянно, необходимо функцию FixedTick вызвать в функции Update, которая создается автоматически в момент создания скрипта.

В конечном результате после прикрепления скрипта к объекту CameraHolder и установления значения для публичных переменных так, как это показано на рисунке 4, будет получена работоспособная камера от третьего лица.

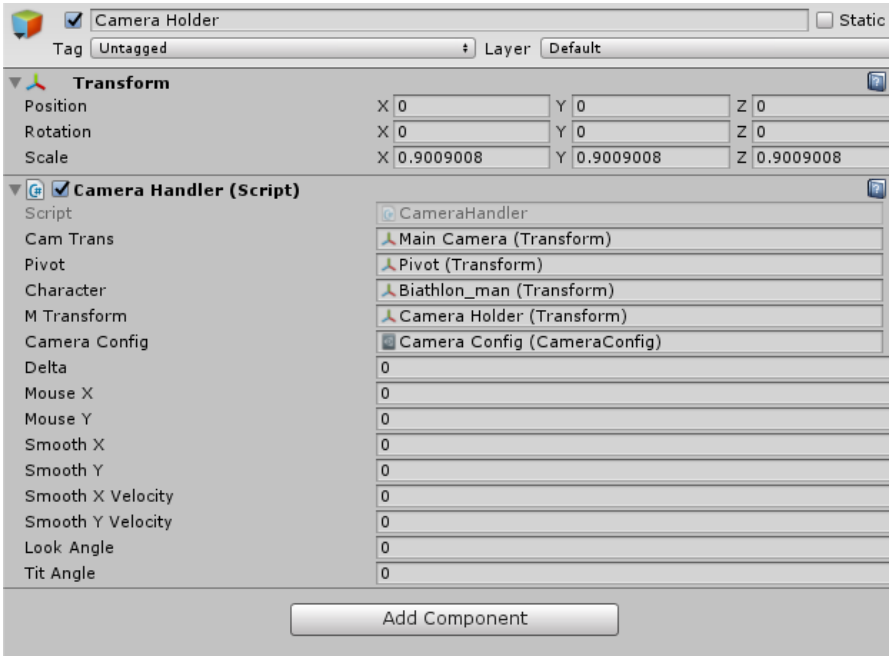

Рисунок 4. – Значения публичных переменных объектов, передаваемых в скрипт камеры CameraHandler

Литература

1. Docs.unity3d [Электронный ресурс]. Режим доступа: https://docs.unity3d.com/ru/current/Manual/CreatingAndUsingScripts.html, свободный, - Загл. с экрана. – Язык русский, английский.

2. Youtube.com [Электронный ресурс]. Режим доступа: https://www.youtube.com/watch?v=E797xIJQJP8&t=666s, свободный, - Загл. с экрана. – Язык русский.