Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет имени Г.В. Плеханова»

**МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ.**

Специальность: 09.02.07 Информационные системы и программирование

Квалификация: Программист

ОТЧЕТ

ПО ОБЩЕПРОФЕССИОНАЛЬНОЙ ДИСЦИПЛИНЕ: МДК 01.01 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

“Камера от первого лица, очки жизней и стамины”

Листов: 8

|  |  |
| --- | --- |
| Выполнил студент  Группы П50-4-21  Плахова Анастасия Вадимовна  «\_25\_» сентября\_\_ 2023 года | Проверил преподаватель  \_\_\_\_\_\_\_\_\_\_\_А.Ю.Бушин  «\_\_\_\_» \_\_\_\_\_\_\_2023 года |

Москва 2023

Цель: При помощи базовых знаний устройства движка Unity и языка C#, создать камеру от первого лица и очки жизней.

Для второй практической открываем файл, который мы создали в предыдущей и чуток будем модернизировать его, а именно добавим шкалу здоровье и выносливости.

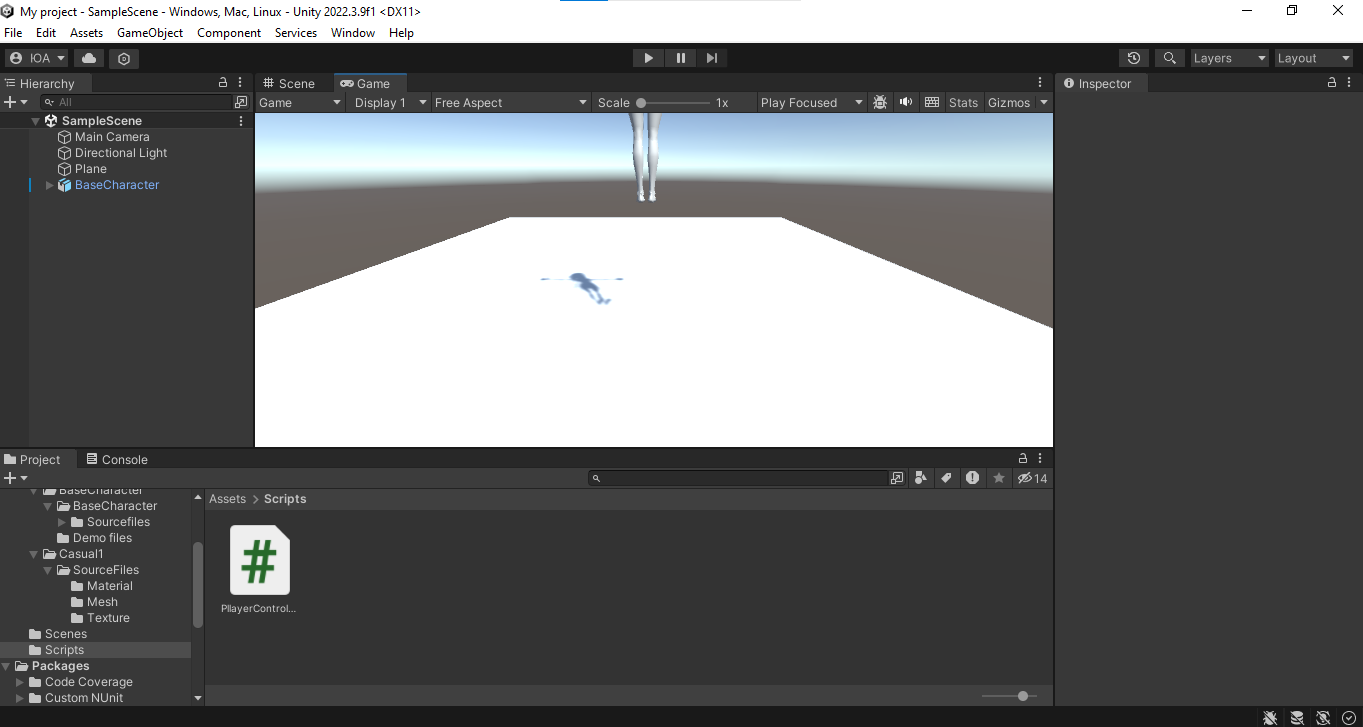


Рисунок начальный файл

Так что для начала нам надо перенести камеру в саму модель и переименовать название камеры, камеру переименуем на FPSCamera и перемещаем камеру внутри модельки.

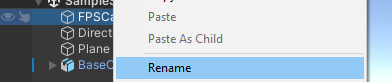


Рисунок переименовав

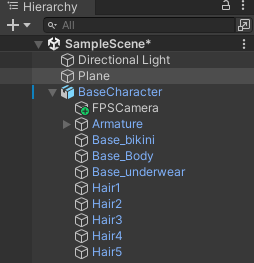


Рисунок После перемещения

Теперь же надо переместить камеру в модель, и должно получиться примерно так.

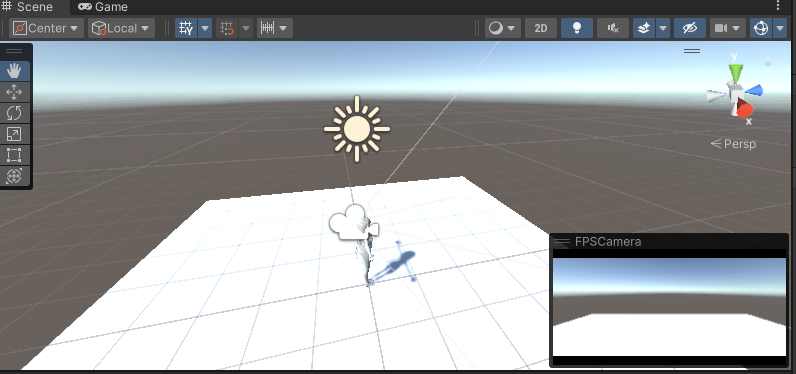


Рисунок камера в модели

Теперь насчёт кода. Этот код реализует вращение камеры вокруг двух осей: горизонтальной (yaw) и вертикальной (pitch) с использованием ввода с мыши. Вращение ограничено углами -60 до 90 градусов по вертикали. Затем создаются кватернионы для плавного вращения. Первый применяется к объекту HandMeshes, а второй к текущему объекту, содержащему этот скрипт. Код обеспечивает плавное управление камерой с помощью мыши.

public void SetRotation()

{

//Считываем движения мыши по X

yaw += Input.GetAxis("Mouse X") \* Sensivity;

//Считываем движения мыши по Y

pitch -= Input.GetAxis("Mouse Y") \* Sensivity;

//Делаем ограничение вращения по оси Y

pitch = Mathf.Clamp(pitch, -60, 90);

//Куда должен повернуться игровк

Quaternion SmoothRotation = Quaternion.Euler(pitch, yaw, 0);

//Линейная интерполяция (вращение со сглаживанием)

HandMeshes.transform.rotation = Quaternion.Slerp(HandMeshes.transform.rotation, SmoothRotation, RotationSmoothing \* Time.fixedDeltaTime);

//То же самое для всего игрового объекта, но только по оси Y

SmoothRotation = Quaternion.Euler(0, yaw, 0);

transform.rotation = Quaternion.Slerp(transform.rotation, SmoothRotation, RotationSmoothing \* Time.fixedDeltaTime);

}

Теперь же нам надо перенести ссылку на модель и настроить скорость, высоту прыжка и вращение. В поле Hand Meshes мы выбираем нашу модель, вписав туда название модели.

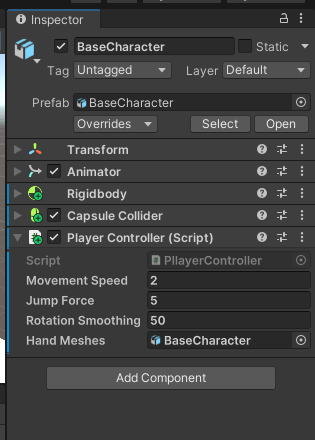


Рисунок ссылка на модель

Если же камера находится неровно, то нужно отцентрировать камеру во вкладках модели, камеры и Armature, всё должно быть равно 0 на Rotation



Рисунок отцентровка

Теперь же добавим стамину и здоровье. Для этого создаем новый скрипт и называем его “GameManager”.



Рисунок Создание скрипта

Далее мы должны написать код для здоровья и стамины. Для этого открываем скрипт GameManager и записываем следующие:

public class GameManager : MonoBehaviour

{

//текущее кол-во жизней

private int Health;

//максимальное кол-во жизней

public int MaxHealth = 100;

//очки выносливости

private int Stamina = 100;

//Проверка восполнения выносливости в данный момент

public bool IsStaminaRestoring = false;

private void Start()

{

Health = MaxHealth;

}

private void FixedUpdate()

{

StaminaCheck();

}

private void StaminaCheck()

{

Debug.Log("Стамина " + Stamina);

//Если стамины нет, востаннавливаем

if (Stamina <= 0) StartCoroutine(StaminaRestore());

}

//Асинхронный метод для востановления выносливости

private IEnumerator StaminaRestore()

{

IsStaminaRestoring = true;

//Задержка в 3 секунды

yield return new WaitForSeconds(3);

Stamina = 100;

IsStaminaRestoring = false;

}

public void SpendStamina()

{

//Тратим выносливость

Stamina -= 1;

}

}

Теперь же открываем код для игрока и надо там чуток подправить.

К объявление переменных нам нужно добавить:

public float SprintSpeed = 10.0f;

private GameManager \_GameManager;

Далее мы добавляем в метод Start ссылку на твёрдое тело:

\_GameManager = FindObjectOfType<GameManager>();

Так же обновим метод FixedUpdate:

Перед вызовом метода setRotation:

if (Input.GetKey(KeyCode.LeftShift) && !\_GameManager.IsStaminaRestoring)

{

\_GameManager.SpendStamina();

\_Rigidbody.MovePosition(CalculateSprint());

}

else \_Rigidbody.MovePosition(CalculateMovement());

Далее мы добавим ещё один метод CalculateSprint:

private Vector3 CalculateSprint()

{

float HorizontalDirection = Input.GetAxis("Horizontal");

float VerticalDirection = Input.GetAxis("Vertical");

Vector3 Move = transform.right \* HorizontalDirection + transform.forward \* VerticalDirection;

return \_Rigidbody.transform.position + Move \* Time.fixedDeltaTime \* /\* Меняется только это -> \*/ SprintSpeed;

}

Теперь мы закончили писать код и переходим к самой сцене. Для этого мы создаём пустой игровой объект с именем GameManager и прикрепляем в него наш скрипт

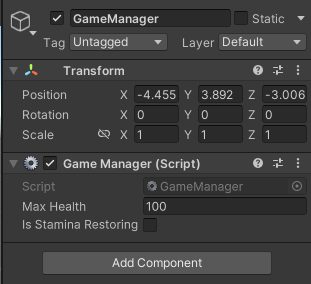


Рисунок 9 GameManager скрипт

Так же для будущего мы можем создать текст, который будет отображать наше здоровье и стамину.

Для этого нам надо создать текст. Для этого нажимаем на правую кнопку мыши выбираем вкладку UI и объект Text – TextMeshPro, и создаётся такой большой прямоугольный экран, этот экран является экраном пользователя и там можно настроить, где именно будет наш текст

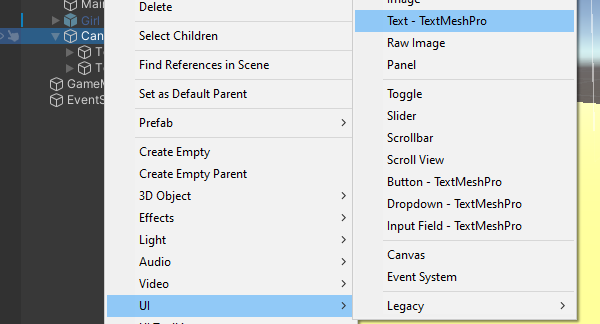


Рисунок 10 создание текста

Так же мы можем перекрасить сам текст и вообще его отредактировать.

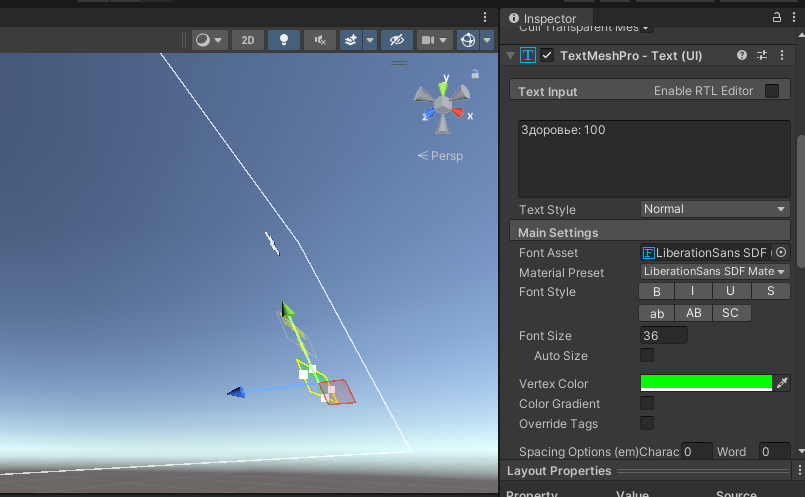


Рисунок 11 расположение текста и его редактирование

Ссылка на видео: <https://youtu.be/RKemDlf8oqo> .

Вывод: При помощи базовых знаний устройства движка Unity и языка C#, создала камеру от первого лица и очки жизней.