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

Создание проекта, передвижение персонажа.

Листов: 7

|  |  |
| --- | --- |
| Выполнил студент  Группы П50-4-21  Игошев Р. В.  «\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2023 года | Проверил преподаватель  \_\_\_\_\_\_\_\_\_\_\_А.Ю.Бушин  «\_\_\_\_» \_\_\_\_\_\_\_2023 года |

Москва 2023

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

Для начала необходимо переместить камеру внутрь модели персонажа и переименовать ее в FPSCamera, что будет означать камеру от первого лица.

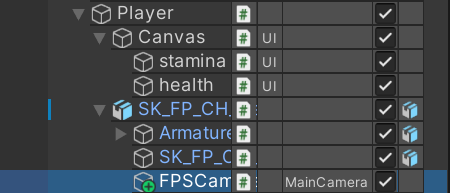


Рисунок 1 – Расположение камеры от первого лица

Структура готового проекта выглядит следующим образом:

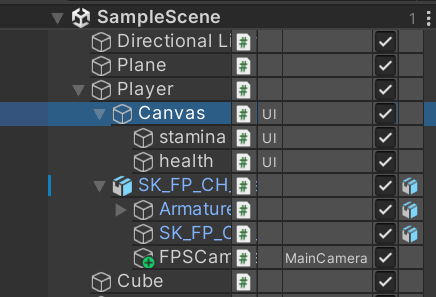


Рисунок 2 – Структура объектов проекта

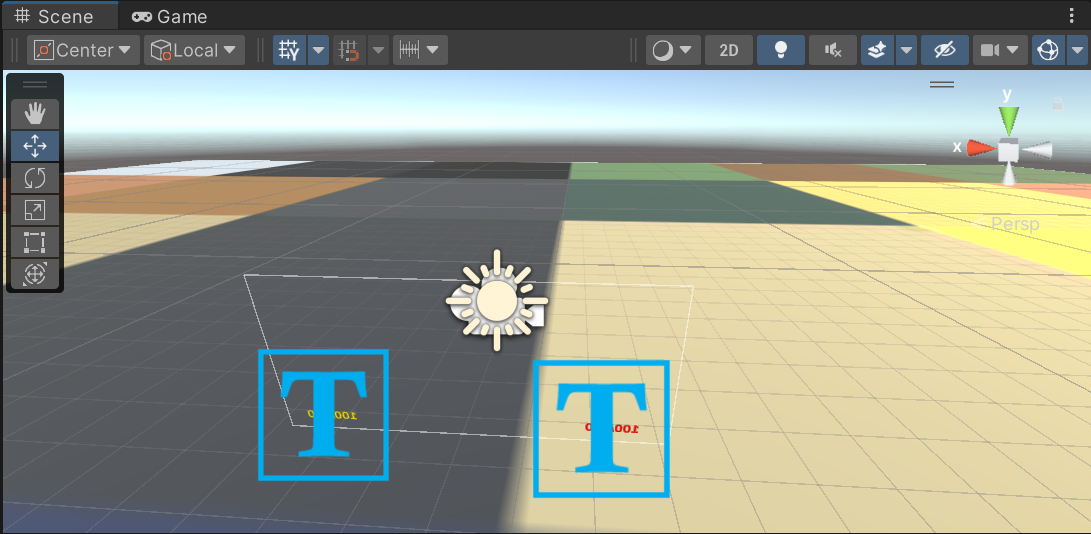


Рисунок 3 – Сцена проекта

Код работы:

using System.Collections;

using System.Collections.Generic;

using TMPro;

using UnityEngine;

using UnityEngine.InputSystem;

public class PlayerController : MonoBehaviour

{

public float MovementSpeed = 2.0f;

public float JumpForce = 5.0f;

public float RotationSmoothing = 20f;

public GameObject HandMeshes;

private TextMeshProUGUI Staminatxt;

private TextMeshProUGUI Healthtxt;

public GameObject Stamina;

public GameObject Health;

private Rigidbody \_Rigidbody;

private bool IsGrounded;

public float DistationToGround = 0.1f;

private float pitch;

private float yaw;

private float coeff;

private float stamina = 100.0f;

private int maxhp = 100;

public int currenthp = 100;

private bool IsRecoveryActive = false;

void Start()

{

\_Rigidbody = GetComponent<Rigidbody>();

Healthtxt = Health.GetComponent<TextMeshProUGUI>();

Staminatxt = Stamina.GetComponent<TextMeshProUGUI>();

Healthtxt.text = $"{currenthp}/{maxhp}";

}

void FixedUpdate()

{

GroundCheck();

if (Input.GetKey(KeyCode.Space) && IsGrounded) Jump();

\_Rigidbody.MovePosition(CalculateMovement());

SetRotation();

Recovery();

}

public void SetRotation()

{

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

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

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

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

HandMeshes.transform.rotation =

Quaternion.Slerp(HandMeshes.transform.rotation, SmoothRotation,

RotationSmoothing \* Time.fixedDeltaTime);

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

transform.rotation = Quaternion.Slerp(transform.rotation, SmoothRotation,

RotationSmoothing \* Time.fixedDeltaTime);

}

private Vector3 CalculateMovement()

{

if ((Input.GetKey(KeyCode.LeftShift) || Input.GetKey(KeyCode.RightShift)) && CanYouSprint()) {

coeff = 2.0f;

stamina -= 25 \* Time.fixedDeltaTime;

Staminatxt.text = $"{(int)stamina}/100";

}

else { coeff = 1.0f; }

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

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

return \_Rigidbody.transform.position + new Vector3(HorizontalDirection, 0, VerticalDirection) \* Time.fixedDeltaTime \* MovementSpeed \* coeff;

}

private void Recovery()

{

if (stamina < 100.0)

{

stamina += 10 \* Time.fixedDeltaTime;

Staminatxt.text = $"{(int)stamina}/100";

}

else

{

stamina = 100.0f;

Staminatxt.text = $"{(int)stamina}/100";

IsRecoveryActive = false;

}

}

private bool CanYouSprint()

{

if (IsRecoveryActive) return false;

else if (stamina <= 0)

{

stamina = 0.0f;

Staminatxt.text = $"{(int)stamina}/100";

IsRecoveryActive = true;

return false;

}

else return true;

}

private void Jump()

{

\_Rigidbody.AddForce(Vector3.up \* JumpForce, ForceMode.Impulse);

}

private void GroundCheck()

{

IsGrounded = Physics.Raycast(transform.position, Vector3.down,

DistationToGround);

}

private void OnDrawGizmosSelected()

{

Gizmos.color = Color.red;

Gizmos.DrawLine(transform.position, transform.position + (Vector3.down \*

DistationToGround));

}

}

Описание некоторых методов, которые не были описаны в прошлой практической работе:

Recovery – метод отвечает за восстановление выносливости для спринта.

CanYouSprint – возвращает bool-значение, отвечающее на вопрос «Может ли в данный момент игрок использовать спринт? », ответ на который зависит от наличия выносливости и проведения восстановления после спринта.

SetRotation – метод, позволяющий игроку вращать FPS камерой, созданной ранее.

Для вывода информации о спринте можно использовать элемент Canvas. Для данной цели, будут актуальны следующие настройки объекта:

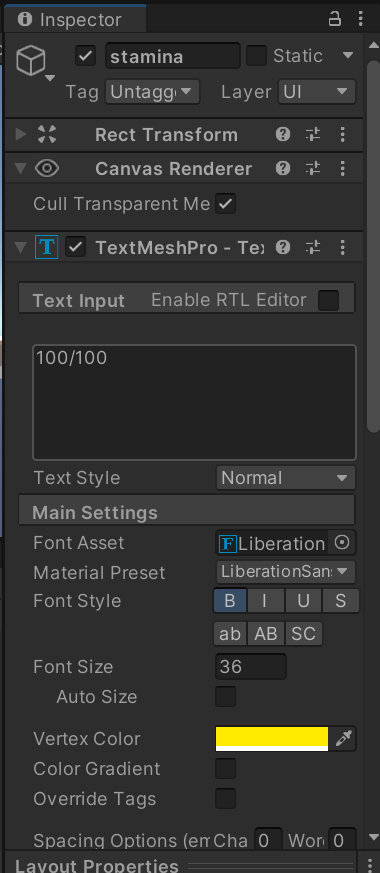


Рисунок 4 – Настройки текста выносливости

Результат работы:

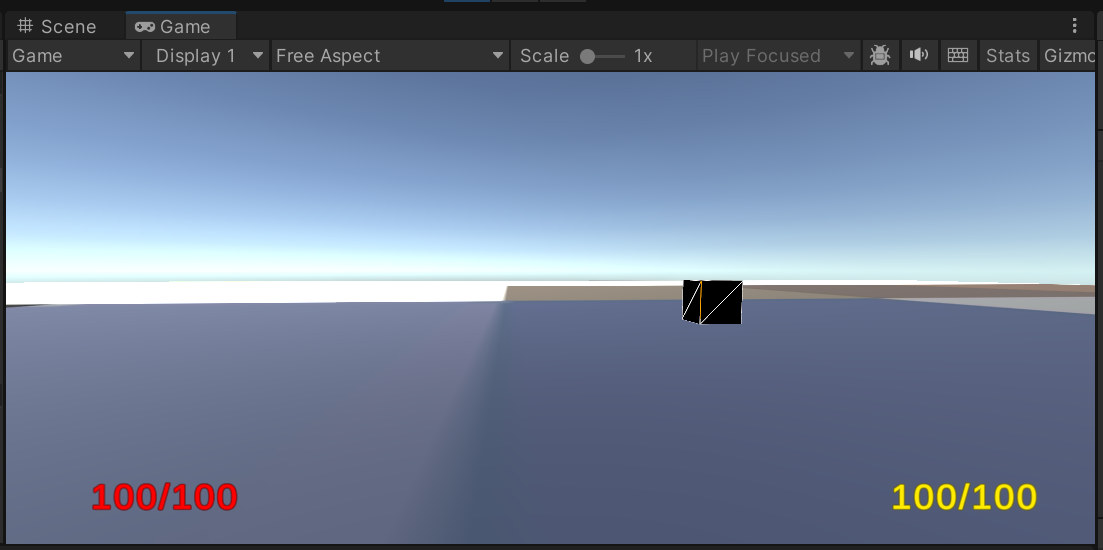


Рисунок 5 – Игра до использования спринта

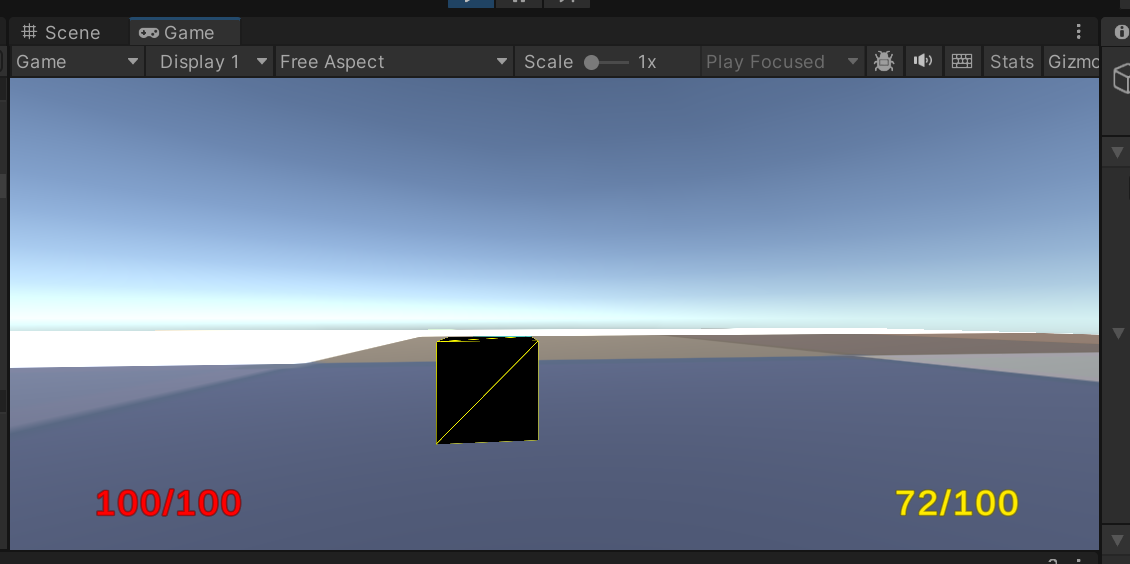


Рисунок 6 – Игра после использования спринта

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