Министерство науки и высшего образования Российской Федерации

Федерального государственного бюджетное образовательное учреждение высшего образования

«Российской экономический университет имени Г. В. Плеханова»

**МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ.**

ПРАКТИЧЕСКИЕ РАБОТЫ № 3

ДИСЦИПЛИНА: «Разработка программных модулей»

Специальность: 09.02.07 Информационные системы и программирование

Квалификация: Программист

Листов:

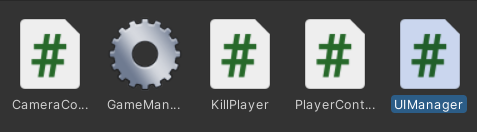
|  |  |  |
| --- | --- | --- |
| Выполнил студент |  | Проверил преподаватель |
| Группа П50-3-18 |  | \_\_\_\_\_\_\_\_\_\_\_О.Б.Солдатова |
| Пахомов Даниил Александрович |  | «\_­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­\_\_» \_\_\_\_\_\_\_\_2020 года |

Москва, 2020

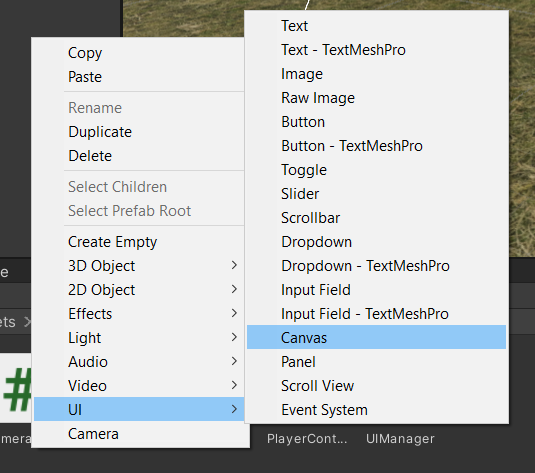
# Практическая работа №3

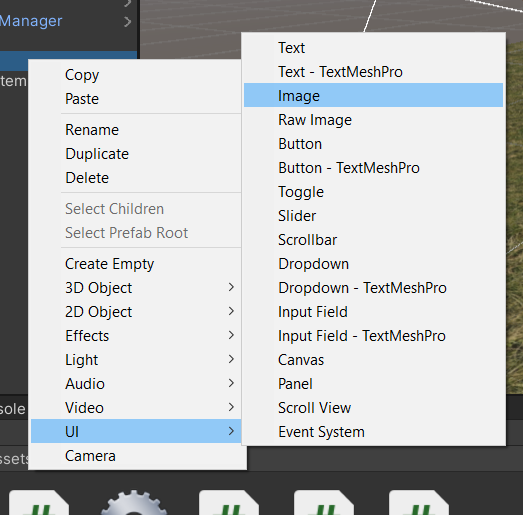
Цель: Базовые системы: Настройка перехода в черный цвет при возрождении. Checkpoint: Добавление контрольных точек, Отображение контрольной точки как активированной, Добавление частиц в контрольные точки, Отключение неактивных контрольных точек. Система здоровья: Добавление системы здравоохранения, Отбрасывание игрока назад, Непобедимость и мигание, Пикап здоровья, Создание эффектов, ЗАДАНИЕ - Эффект частиц здоровья. (Самостоятельная работа)

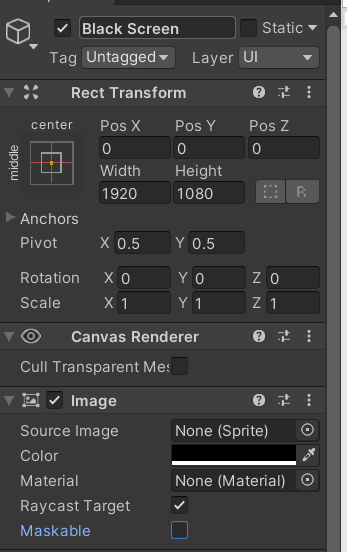
1. БАЗОВЫЕ СИСТЕМЫ
   1. Создаем скрипт, отвечающий за дизайн окна



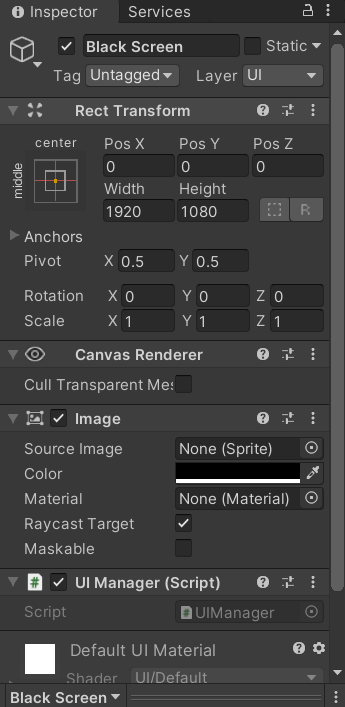
* 1. Добавим в иерархию объект дизайна внешнего вида окна под названием Canvas.







* 1. Привязываем скрипт UIManager к объекту Canvas



* 1. Листинг кода UIManager

|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  using UnityEngine.UI;  public class UIManager : MonoBehaviour  {  public static UIManager instance;  public Image blackScreen;  public float fadeSpeed = 1f;  public bool fadeToBlack, fadeFromBlack;  void Start()  {  instance = this;  }  // Update is called once per frame  void Update()  {  if(fadeToBlack)  {  blackScreen.color = new Color(blackScreen.color.r, blackScreen.color.g, blackScreen.color.b, Mathf.MoveTowards(blackScreen.color.a, 1f, fadeSpeed \* Time.deltaTime));  if(blackScreen.color.a == 1f)  {  fadeToBlack = false;  }  if(fadeToBlack)  {  blackScreen.color = new Color(blackScreen.color.r, blackScreen.color.g, blackScreen.color.b, Mathf.MoveTowards(blackScreen.color.a, 0f, fadeSpeed \* Time.deltaTime));  if(blackScreen.color.a == 0f)  {  fadeFromBlack = false;  }  }  }  }  } |

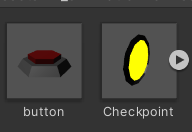
* 1. Листинг кода GamaManager

|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class GameManager : MonoBehaviour  {  public static GameManager instance;  private Vector3 respawnPosition;  private void Awake()  {  instance = this;  }  // Start is called before the first frame update  void Start()  {  Cursor.visible = false;  Cursor.lockState = CursorLockMode.Locked;  respawnPosition = PlayerController.instance.transform.position;  }  // Update is called once per frame  void Update()  {    }  public void Respawn()  {  StartCoroutine(RespawnCo());  }  public IEnumerator RespawnCo()  {  PlayerController.instance.gameObject.SetActive(false);  CameraController.instance.theCMBrain.enabled = false;  UIManager.instance.fadeToBlack = true;  yield return new WaitForSeconds(2f);  UIManager.instance.fadeToBlack = true;  PlayerController.instance.transform.position = respawnPosition;  PlayerController.instance.gameObject.SetActive(true);  }  } |

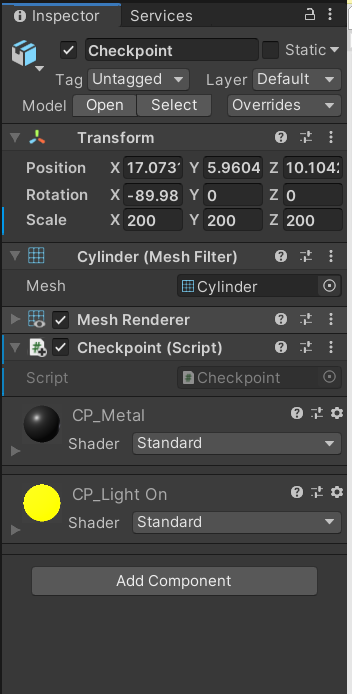
1. Добавление CHECKPOINT
   1. Отключаем отображение UI слоя



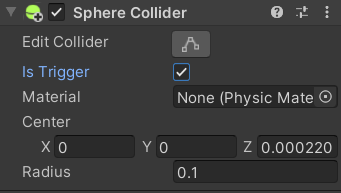
* 1. Размещаем объект checkpoint на нашей площадке в произвольном месте



* 1. Создаем новый скрипт и кидаем на объект Checkpoint



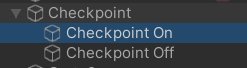
* 1. На объект Checkpoint добавляем SphereComponent и делаем ему настройку



* 1. Листинг кода

|  |  |
| --- | --- |
| Checkpoint | using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class Checkpoint : MonoBehaviour  {  // Start is called before the first frame update  void Start()  {    }  // Update is called once per frame  void Update()  {    }  private void OnTriggerEnter(Collider other)  {  if (other.tag == "Player")  {  GameManager.instance.SetSpawnPoint(transform.position);  }  }  } |
| GameManager | public void SetSpawnPoint(Vector3 newSpawnPoint)  {  respawnPosition = newSpawnPoint;  } |

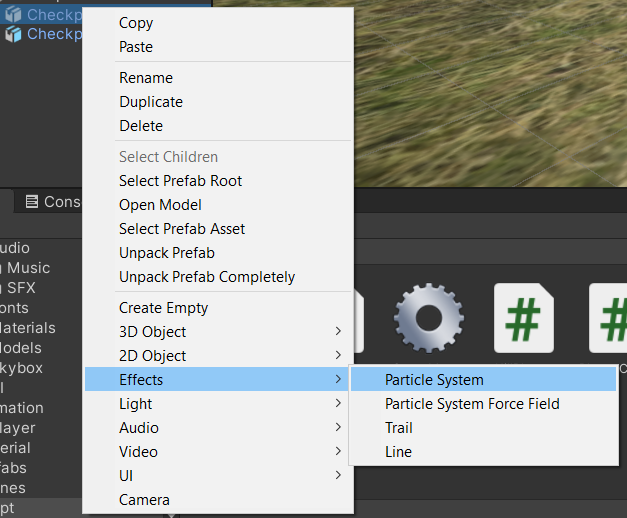
* 1. Отображение контрольной точке как активированной создаем два объекта внутри Checkpoint



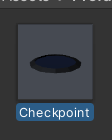
* 1. Листинг кода

|  |  |
| --- | --- |
| Checkpoint | using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class Checkpoint : MonoBehaviour  {  // Start is called before the first frame update  void Start()  {    }  // Update is called once per frame  void Update()  {    }  private void OnTriggerEnter(Collider other)  {  if (other.tag == "Player")  {  GameManager.instance.SetSpawnPoint(transform.position);  cpOff.SetActive(false);  cpOn.SetActive(true);}  }  } |
| GameManager | public void SetSpawnPoint(Vector3 newSpawnPoint)  {  respawnPosition = newSpawnPoint;  } |

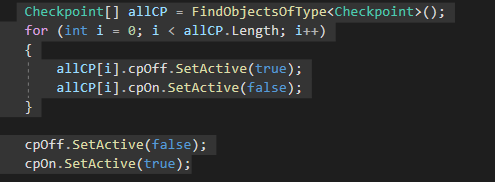
* 1. Добавление частиц



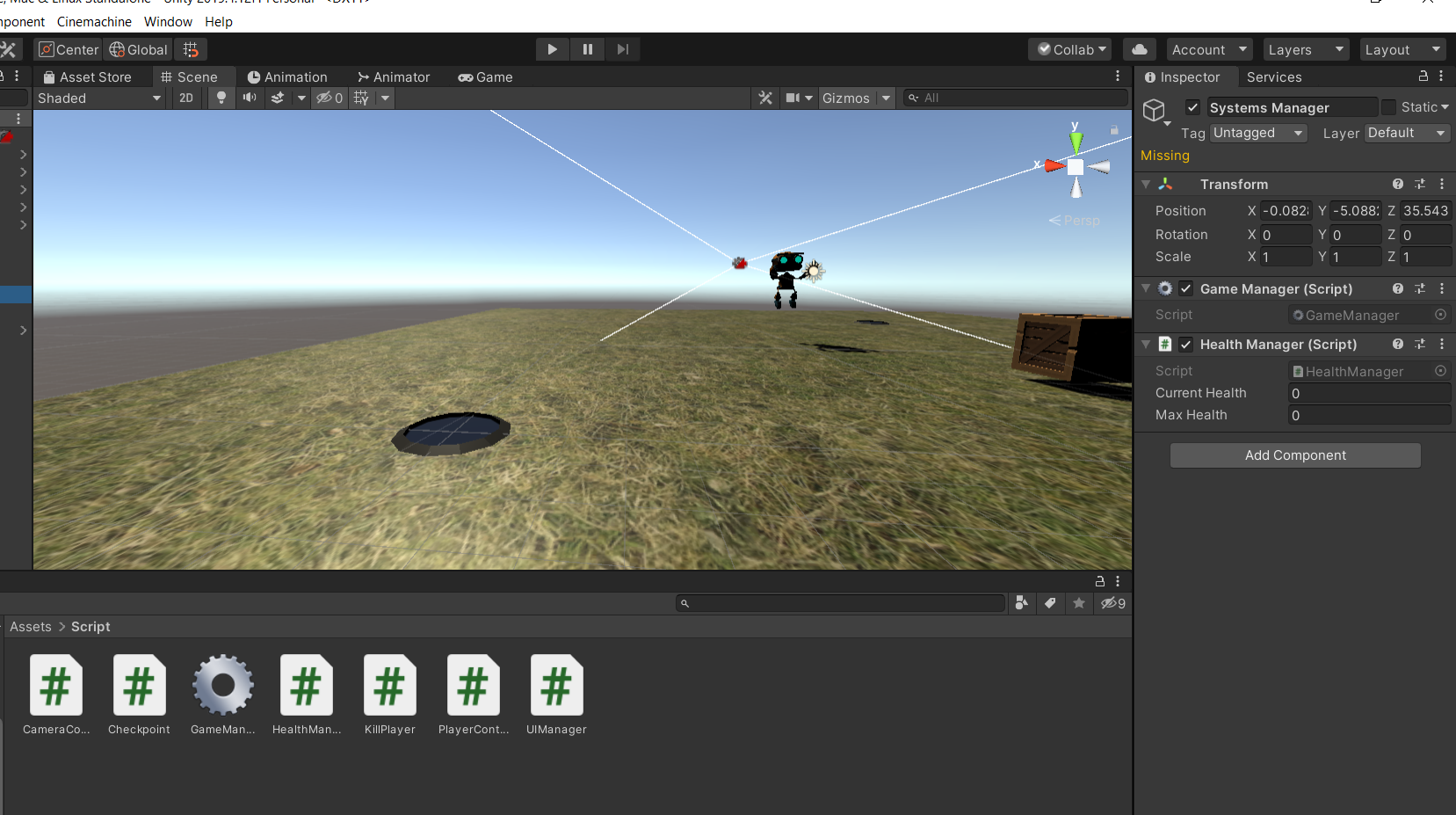
* 1. Настройка Particle System
  2. Создаем Prafabs



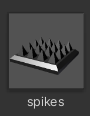
* 1. Добавляем 3 Checkpoint на площадку
  2. Листинг кода



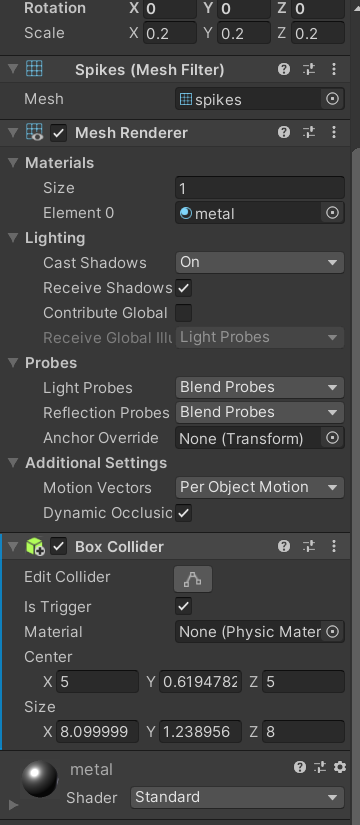
1. Система здоровья
   1. Создаем скрипт HealthManager и кидаем его на Systems Manager



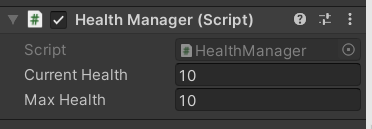
* 1. Добавляем объект на площадку



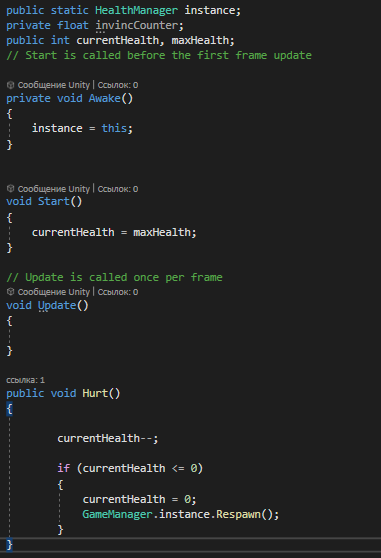
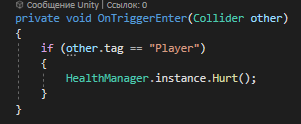
* 1. Настройка Spikes



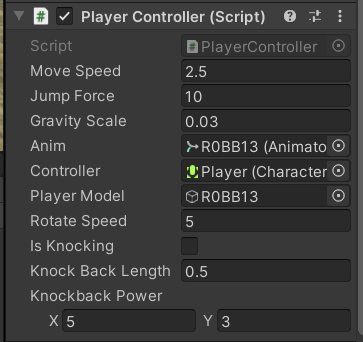
* 1. Настройка жизни



* 1. Создаем скрипт HurtPlayer и делаем листинг кода



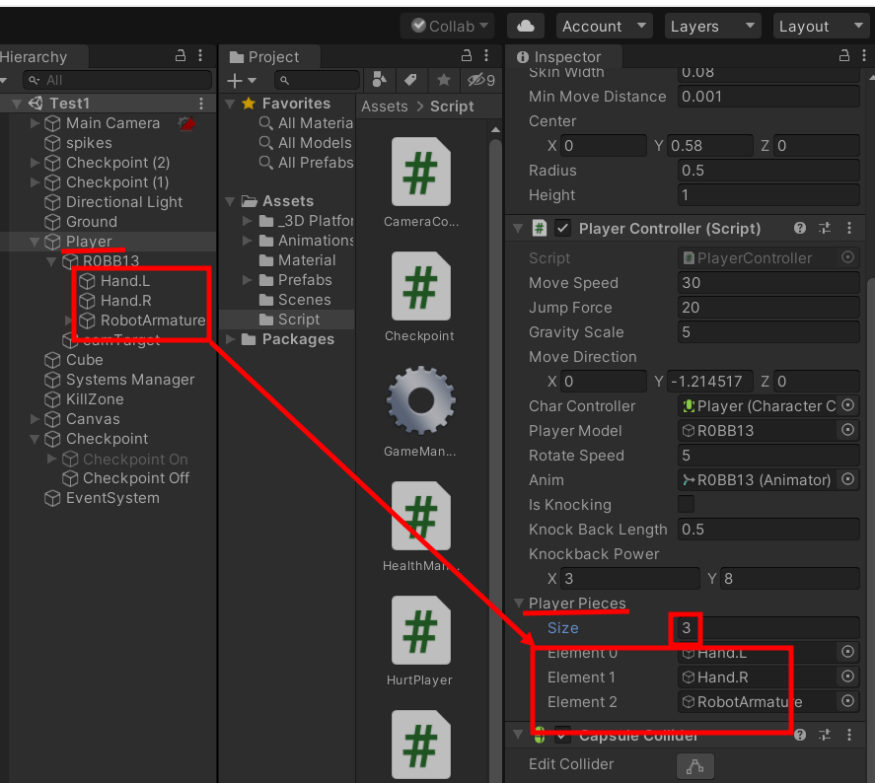
* 1. Отбрасывание игрока



|  |  |
| --- | --- |
| PlayerController | using System.Collections;  using System.Collections.Generic;  using System.Threading;  using UnityEngine;  public class PlayerController : MonoBehaviour  {  public float moveSpeed; //Скорость передвижения  public float jumpForce; //Высота прыжков  public float gravityScale; //Сила гравитации  public Animator anim; //  public CharacterController controller; //Объект контроллера персонажа  public GameObject playerModel;//  public float rotateSpeed;// Скорость поворота камеры  private Vector3 \_moveDirection;// Переменная отвечающая за движение  private Camera theCam; // Камера  public bool isKnocking;  public float knockBackLength = .5f;  private float knockbackCounter;  public Vector2 knockbackPower;  public static PlayerController instance; //переменная для переопределения класса  private void Awake()  {  instance = this;  }  void Start()  {  theCam = Camera.main;  }  void Update()  {  float yStore = \_moveDirection.y;  if (!isKnocking)  {  //\_moveDirection = new Vector3(Input.GetAxisRaw("Horizontal") \* moveSpeed, \_moveDirection.y, Input.GetAxisRaw("Vertical") \* moveSpeed);  \_moveDirection = (transform.forward \* Input.GetAxisRaw("Vertical")) + (transform.right \* Input.GetAxisRaw("Horizontal"));  \_moveDirection.Normalize();// Нормализация движения  \_moveDirection = \_moveDirection \* moveSpeed;  \_moveDirection.y = yStore;  if (controller.isGrounded)  {  \_moveDirection.y = 0f;  if (Input.GetButtonDown("Jump"))  \_moveDirection.y = jumpForce;  }//Защита от Бесконечных прыжков  \_moveDirection.y = \_moveDirection.y + (Physics.gravity.y \* gravityScale);//Гравитация  controller.Move(\_moveDirection \* Time.deltaTime);//скорость обновления  if (Input.GetAxisRaw("Horizontal") != 0 || Input.GetAxisRaw("Vertical") != 0)  {  transform.rotation = Quaternion.Euler(0f, theCam.transform.rotation.eulerAngles.y, 0f);  Quaternion newRotation = Quaternion.LookRotation(new Vector3(\_moveDirection.x, 0f, \_moveDirection.z));  playerModel.transform.rotation = Quaternion.Slerp(playerModel.transform.rotation, newRotation, rotateSpeed \* Time.deltaTime);  }//Корректное перемещение персонажа  }  if(isKnocking)  {  knockbackCounter -= Time.deltaTime;  \_moveDirection = playerModel.transform.forward \* -knockbackPower.x;  \_moveDirection.y = yStore;  if (controller.isGrounded)  {  \_moveDirection.y = 0f;  }  \_moveDirection.y += Physics.gravity.y \* Time.deltaTime \* gravityScale;  controller.Move(\_moveDirection \* Time.deltaTime);  if (knockbackCounter <= 0)  {  isKnocking = false;  }  }  anim.SetFloat("Speed", Mathf.Abs(\_moveDirection.x) + Mathf.Abs(\_moveDirection.z));// для работы переменной которая отвечает за анимацию бега  anim.SetBool("Grounded", controller.isGrounded);// для работы переменной которая отвечает за анимацию прыжка  }  public void Knockback()  {  isKnocking = true;  knockbackCounter = knockBackLength;  \_moveDirection.y = knockbackPower.y;  controller.Move(\_moveDirection \* Time.deltaTime);  }  } |
| HealthManager | public void Hurt()  {  currentHealth--;  if (currentHealth <= 0)  {  currentHealth = 0;  GameManager.instance.Respawn();  }  else  {  PlayerController.instance.Knockback();  }  } |

* 1. Непобедимость и отбрасывание листинг

|  |  |
| --- | --- |
| HealhtManager | using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class HealthManager : MonoBehaviour  {  public static HealthManager instance;  public float invincibleLength = 2f;  private float invincCounter;  public int currentHealth, maxHealth;  // Start is called before the first frame update  private void Awake()  {  instance = this;  }  void Start()  {  currentHealth = maxHealth;  }  // Update is called once per frame  void Update()  {  if (invincCounter > 0)  {  invincCounter -= Time.deltaTime;  for (int i = 0; i < PlayerController.instance.playerPrieces.Length; i++)  {  if (Mathf.Floor(invincCounter \* 5f) % 2 == 0)  {  PlayerController.instance.playerPrieces[i].SetActive(true);  }  else  {  PlayerController.instance.playerPrieces[i].SetActive(false);  }  if (invincCounter <= 0)  {  PlayerController.instance.playerPrieces[i].SetActive(true);  }  }  }  }  public void Hurt()  {  if (invincCounter <= 0)  {  currentHealth--;  if (currentHealth <= 0)  {  currentHealth = 0;  GameManager.instance.Respawn();  }  else  {  PlayerController.instance.Knockback();  invincCounter = invincibleLength;  }  }  }  } |
| PlayerController | public GameObject[] playerPrieces; |

****

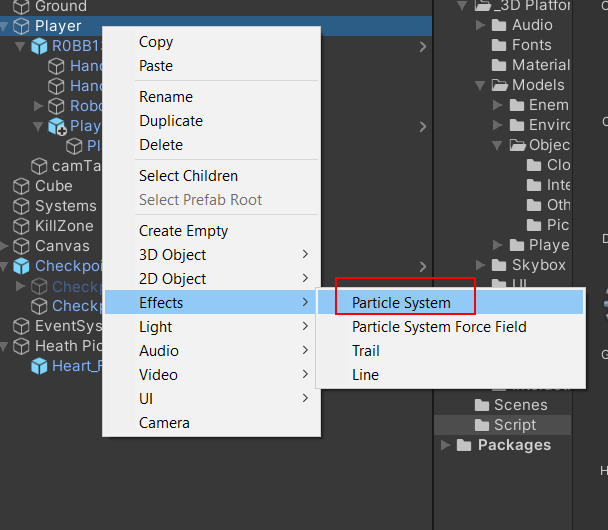
* 1. Элемент здоровья

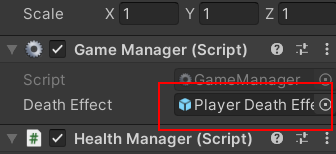


****

|  |  |
| --- | --- |
| HealthPickup | using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class HealthPickup : MonoBehaviour  {  // Start is called before the first frame update  public int healAmount;  public bool isFullHeal;  void Start()  {  }  // Update is called once per frame  void Update()  {  }  private void OnTriggerEnter(Collider other)  {  if (other.tag == "Player")  {  Destroy(gameObject);  if (isFullHeal)  {  HealthManager.instance.ResetHealth();  }  else  {  HealthManager.instance.AddHealth(healAmount);  }  }  }  } |
| HealthManager | public void ResetHealth()  {  currentHealth = maxHealth;  }  public void AddHealth(int amountToHeal)  {  currentHealth += amountToHeal;  if (currentHealth > maxHealth)  {  currentHealth = maxHealth;  }  } |

1. Создание эффекта





* 1. Листинг кода

|  |  |
| --- | --- |
| DestroyOtherTime | using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class DestroyOtherTime : MonoBehaviour  {      // Start is called before the first frame update      public float lifetime;      void Start()      {        }      // Update is called once per frame      void Update()      {          Destroy(gameObject,lifetime);      }  } |
| GameManager | using System.Collections;  using System.Collections.Generic;  using UnityEngine;  using UnityEngine.SceneManagement;  public class GameManager : MonoBehaviour  {      public static GameManager instance;      private Vector3 respawnPosition,camSpawnPosition;        public GameObject deathEffect;      private void Awake()      {          instance = this;      }      // Start is called before the first frame update      void Start()      {          Cursor.visible = false;          Cursor.lockState = CursorLockMode.Locked;          respawnPosition = PlayerController.instance.transform.position;        }      // Update is called once per frame      void Update()      {        }      public void Respawn()      {          StartCoroutine(RespawnCo());      }      public IEnumerator RespawnCo()      {          PlayerController.instance.gameObject.SetActive(false);          CameraController.instance.theCMBrain.enabled = false;          UIManager.instance.fadeToBlack = true;          Instantiate(deathEffect, PlayerController.instance.transform.position + new Vector3(0f,1f,0f), PlayerController.instance.transform.rotation);          yield return new WaitForSeconds(2f);          UIManager.instance.fadeFromBlack = true;          PlayerController.instance.transform.position = respawnPosition;          CameraController.instance.theCMBrain.enabled = true;          PlayerController.instance.gameObject.SetActive(true);      }      public void SetSpawnPoint(Vector3 newSpawnPoint){          respawnPosition = newSpawnPoint;      }  } |

Вывод: все указанные задания были выполнены и также самостоятельная работа. Базовые системы: Настройка перехода в черный цвет при возрождении. Checkpoint: Добавление контрольных точек, Отображение контрольной точки как активированной, Добавление частиц в контрольные точки, Отключение неактивных контрольных точек. Система здоровья: Добавление системы здравоохранения, Отбрасывание игрока назад, Непобедимость и мигание, Пикап здоровья, Создание эффектов, ЗАДАНИЕ - Эффект частиц здоровья. (Самостоятельная работа)