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 № 4

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

Содержание

[Практическая работа №4 3](#_Toc55299984)

[Интерфейс и сбор монет 3](#_Toc55299985)

[Отображение здоровья игрока 3](#_Toc55299986)

[Переключение изображений панели здоровья 8](#_Toc55299987)

[Сбор монет 13](#_Toc55299988)

[Анимированные монеты 17](#_Toc55299989)

[Добавление аудио 18](#_Toc55299990)

[Переключение музыкальных треков 21](#_Toc55299991)

[Использование аудио микшера 22](#_Toc55299992)

[Воспроизведение звуковых эффектов 22](#_Toc55299993)

[Меню паузы 26](#_Toc55299994)

[Приостановка игры 27](#_Toc55299995)

[Меню опций 29](#_Toc55299996)

[Управление громкостью через пользовательский интерфейс 29](#_Toc55299997)

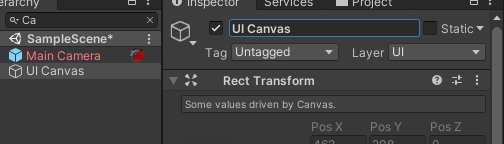
# Практическая работа №4

Цель: Интерфейс и сбор монет (отображение здоровья игрока, переключение изображений панели здоровья, эффект собирания монет, анимированные монеты), добавление аудио (воспроизведение музыки, переключение музыкальных треков, использование аудиомикшера, воспроизведение звуков эффектов, добавление звуковых эффектов к объектам), меню паузы (создание меню паузы, приостановка игры, меню опций, управление громкостью через пользовательский интерфейс, настройка Skybox.)

## Интерфейс и сбор монет

### Отображение здоровья игрока

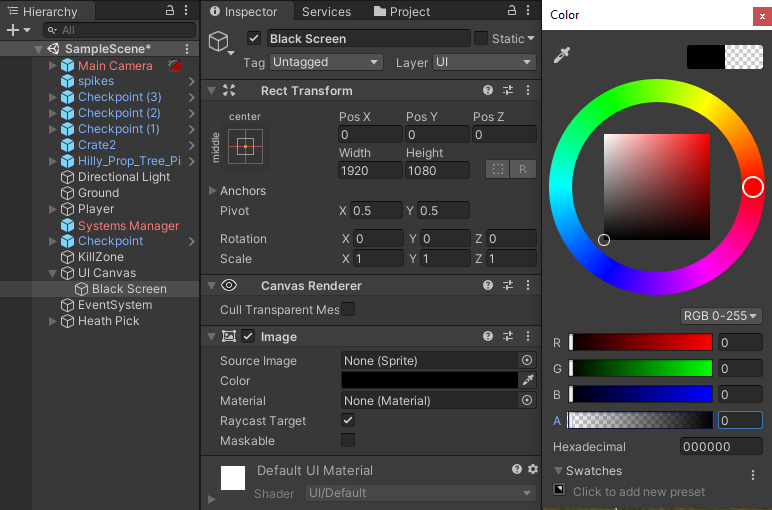
1. Объект Canvas переименовываем в отображение UI Canvas.



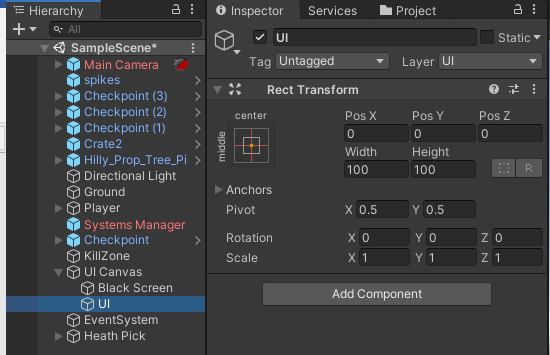
1. Временно включаем отображение слоя с UI.



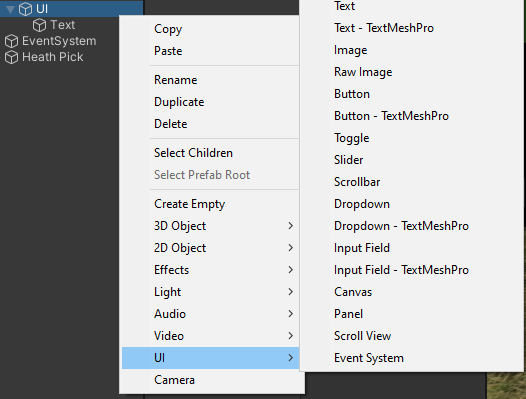
1. Изменю прозрачность BlackScreen на ноль



1. Внутри объекта UI Canvas, создадим элемент в виде пустого объекта и дадим эму имя UI.

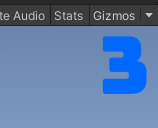


1. Внутри объекта UI создаем объект текста

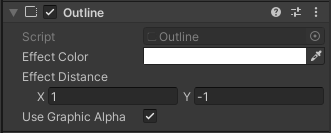


1. Настройка объекта Text

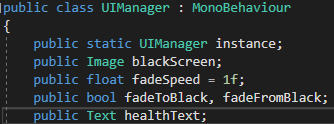




1. Добавление к объекту Text компонента outline и его настройка

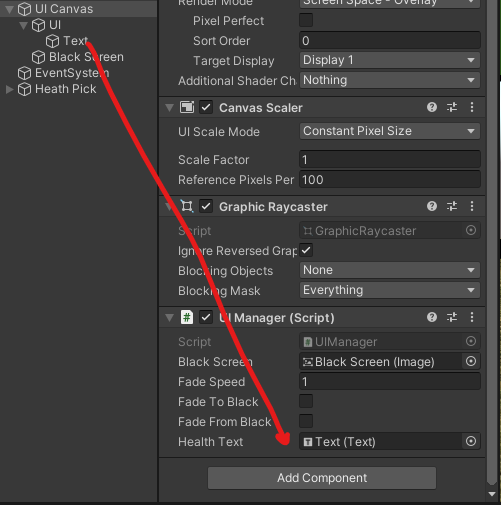


1. Изменяем скрипт UIManager, добавили public переменную для обозначения здоровья



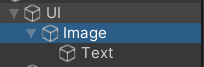
1. Создание метода для обновления текста здоровья и листинг кода UIManager

|  |
| --- |
| 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;  }  public void UpdateUI()  {  UIManager.instance.healthText.text = currentHealth.ToString();  }  // 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;  }  UpdateUI();  }  }  public void ResetHealth()  {  currentHealth = maxHealth;  UpdateUI();  }  public void AddHealth(int amountToHeal)  {  currentHealth += amountToHeal;  if (currentHealth > maxHealth)  {  currentHealth = maxHealth;  }  UpdateUI();  }  } |

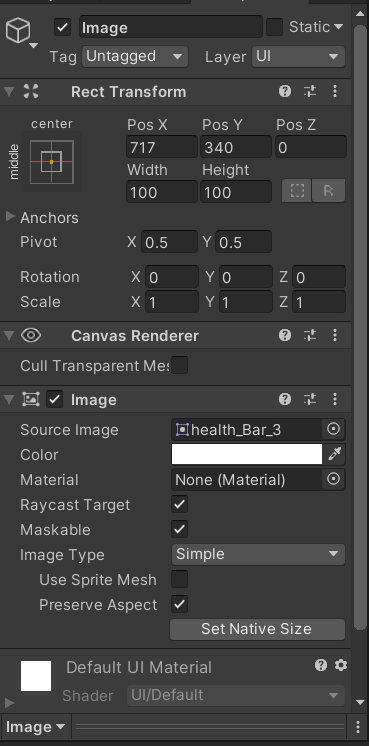


### Переключение изображений панели здоровья

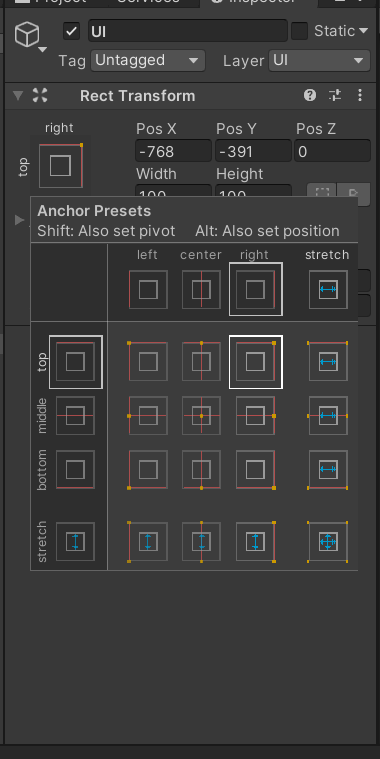
1. Добавим к объекту UI объект Image и внутрь него поместим объект Text



1. Привязываем к объекту Image картинку с тремя лепестками и сделаем не большую настройку



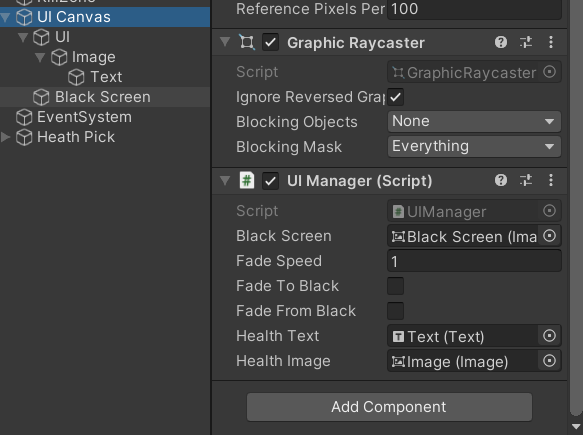
1. Сделаем привязку объекту с помощью якорей чтобы при изменении размера окна здоровье оставалось всегда в правом верхнем углу



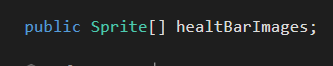
1. В файле UIManager создаём переменную для привязки картинки



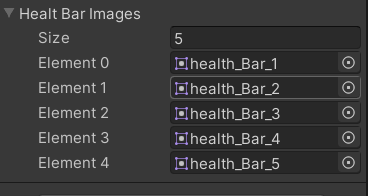
1. Привязываем через инспектор, объект к созданной переменной.



1. В файле Health Manager создаем массив для Sprite объектов.



1. Заполняем массив нужными картинками

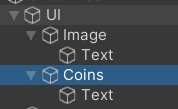


1. Меняем код, листиг кода

|  |  |
| --- | --- |
| HealthManager | 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  public Sprite[] healtBarImages;  private void Awake()  {  instance = this;  }  void Start()  {  currentHealth = maxHealth;  UpdateUI();  }  public void UpdateUI()  {  UIManager.instance.healthText.text = currentHealth.ToString();  switch (currentHealth)  {  case 5:  {  UIManager.instance.healthImage.enabled = true;  UIManager.instance.healthImage.sprite = healtBarImages[4];  break;  }  case 4:  {  UIManager.instance.healthImage.sprite = healtBarImages[3];  break;  }  case 3:  {  UIManager.instance.healthImage.enabled = true;  UIManager.instance.healthImage.sprite = healtBarImages[2];  break;  }  case 2:  {  UIManager.instance.healthImage.sprite = healtBarImages[1];  break;  }  case 1:  {  UIManager.instance.healthImage.sprite = healtBarImages[0];  break;  }  case 0:  {  UIManager.instance.healthImage.enabled = false;  break;  }  }  }  // 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;  }  UpdateUI();  }  }  public void ResetHealth()  {  currentHealth = maxHealth;  UIManager.instance.healthImage.enabled = true;  UpdateUI();  }  public void PlayerKilled()  {  currentHealth = 0;  UpdateUI();  }  public void AddHealth(int amountToHeal)  {  currentHealth += amountToHeal;  if (currentHealth > maxHealth)  {  currentHealth = maxHealth;  }  UpdateUI();  }  } |
| GameManager | using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class GameManager : MonoBehaviour  {  private Vector3 respawnPosition; // переменная для получения позиции  public static GameManager instance; //переменная для переопределения класса  // Start is called before the first frame update  public GameObject deathEffect;  private void Awake()  {  instance = this;  }  void Start()  {  Cursor.visible = false;//----------  Cursor.lockState = CursorLockMode.Locked; // 2 строчки для скрытья и показа курсора  respawnPosition = PlayerController.instance.transform.position;// получение позиции игрока  }  // Update is called once per frame  void Update()  {  }  public void SetSpawnPoint(Vector3 newSpawnPoint)  {  newSpawnPoint.y += 2f;  respawnPosition = newSpawnPoint;  }  public void Respawn()  {  HealthManager.instance.PlayerKilled();  StartCoroutine(RespawnCo());  Debug.Log("я респаун");    }// вызов самого респавна  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);  } //Респавн с задержкой  } |

### Сбор монет

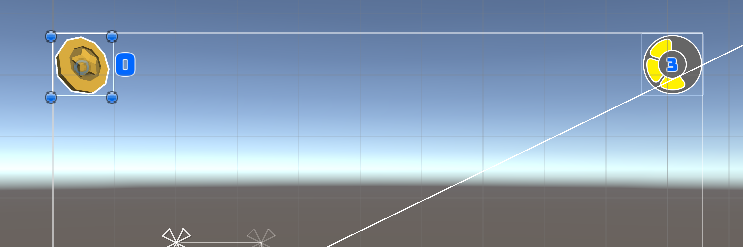
1. Создаем такой же объект как со здоровьем



1. Настройка картинки и текста



1. Расположение объекта на экране и конечный результат



1. Для появившейся монетки добавляем пустой объект и вкладываем модель в него, сам пустой объект переименуем в Coin Puckup. Самое главное, чтобы при размещении у объектов совпадали центры, иначе при добавлении коллайдера, будет неправильно отрабатывать триггер при столкновении с персонажем.



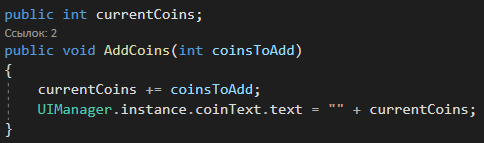
1. Настройки Coin Pick



1. Создаем скрипт и добавляем его на Coin Pick
2. Листниг файла CoinPick

|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class CoinPick : MonoBehaviour  {  // Start is called before the first frame update  public bool isFullCoin ;  public GameObject deathEffect;  void Start()  {    }  public int value;  // Update is called once per frame  void Update()  {    }  private void OnTriggerEnter(Collider other)  {  if (other.tag == "Player")  {  Destroy(gameObject);  GameManager.instance.AddCoins(value);  Instantiate(deathEffect, PlayerController.instance.transform.position + new Vector3(0f, 1f, 0f), PlayerController.instance.transform.rotation);  AudioManager.instance.PlaySFX(4);  }  }  } |

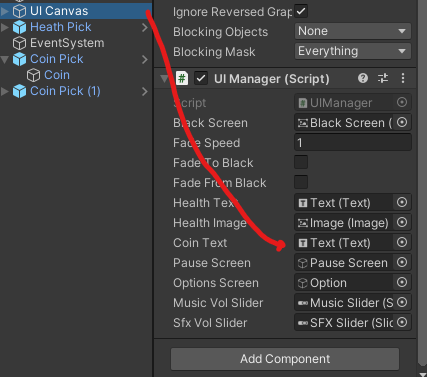
1. GameManger



1. Создаем переменную для связи текста с четом монет. UIManager.

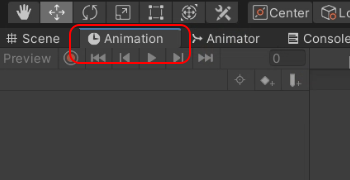
****

1. Присваиваем объект к переменной

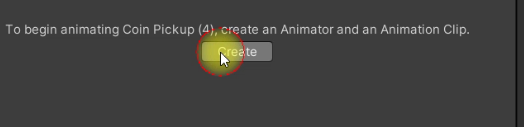


### Анимированные монеты

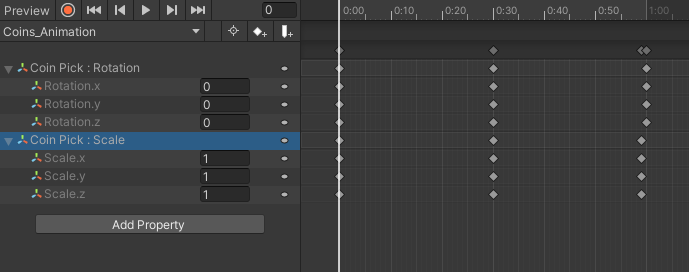
1. Выбираем монетку и заходим во вкладку Animation

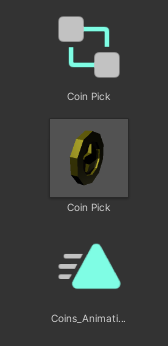


1. Нажимаем на кнопку Create



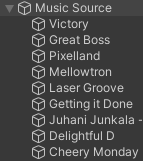
1. Созданная анимация





## Добавление аудио

1. Создаем объект в который поместим всю нашу музыку



1. Настройка объектов



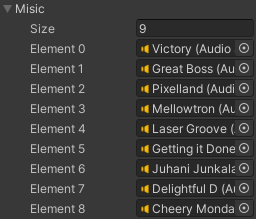
1. Теперь создадим новый скрипт, для проигрывания музыки - > AudioManager и привяжем его к объекту systems manager.



1. Листинг кода

|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  using UnityEngine.Audio;  public class AudioManager : MonoBehaviour  {  // Start is called before the first frame update  public static AudioManager instance;  public AudioSource[] misic;  public AudioSource[] sfx;  public int levelMusicToPlay;  private int currentTrack;  public AudioMixerGroup musicMixer, sfxMixer;  public void Awake()  {  instance = this;  }  void Start()  {  PlayMusic(levelMusicToPlay);  }  // Update is called once per frame  void Update()  {  if (Input.GetKeyDown(KeyCode.M))  {  currentTrack++;  if (currentTrack == 10) currentTrack = 0;  PlayMusic(currentTrack);  }  if (Input.GetKeyDown(KeyCode.J))  {  for (int i = 0; i < misic.Length; i++)  {  misic[i].Stop();  }  }  }  public void PlayMusic(int musicToPlay)  {  for (int i = 0; i < misic.Length; i++)  {  misic[i].Stop();  }  misic[musicToPlay].Play();  }  public void SetMusicLevel()  {  musicMixer.audioMixer.SetFloat("MusicVol", UIManager.instance.musicVolSlider.value);  }  public void PlaySFX(int sfxToPlay)  {  //инача  sfx[sfxToPlay].Play();  }  public void SetSFXLevel()  {  sfxMixer.audioMixer.SetFloat("SfxVol", UIManager.instance.sfxVolSlider.value);  }  } |

1. Привязка музыки к массиву



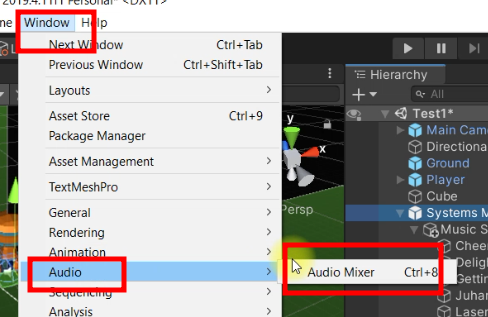
### Переключение музыкальных треков

1. Функция которая будет переключать музыку и выключать ее.

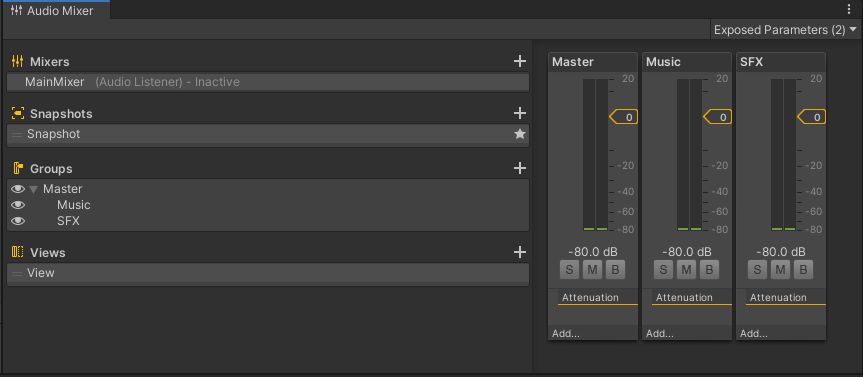
|  |
| --- |
| void Update()  {  if (Input.GetKeyDown(KeyCode.M))  {  currentTrack++;  if (currentTrack == 10) currentTrack = 0;  PlayMusic(currentTrack);  }  if (Input.GetKeyDown(KeyCode.J))  {  for (int i = 0; i < misic.Length; i++)  {  misic[i].Stop();  }  }  } |

### Использование аудио микшера

1. Добавление окна регулировки звуков



1. Настройка микшера



1. Выбираем мелодии и назначаем им микшер

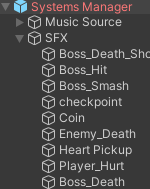


### Воспроизведение звуковых эффектов

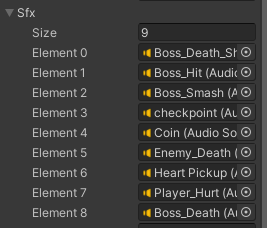
1. Листинг кода

|  |  |
| --- | --- |
| AudioManager | using System.Collections;  using System.Collections.Generic;  using UnityEngine;  using UnityEngine.Audio;  public class AudioManager : MonoBehaviour  {  // Start is called before the first frame update  public static AudioManager instance;  public AudioSource[] misic;  public AudioSource[] sfx;  public int levelMusicToPlay;  private int currentTrack;  public AudioMixerGroup musicMixer, sfxMixer;  public void Awake()  {  instance = this;  }  void Start()  {  PlayMusic(levelMusicToPlay);  }  // Update is called once per frame  void Update()  {  if (Input.GetKeyDown(KeyCode.M))  {  currentTrack++;  if (currentTrack == 10) currentTrack = 0;  PlayMusic(currentTrack);  }  if (Input.GetKeyDown(KeyCode.J))  {  for (int i = 0; i < misic.Length; i++)  {  misic[i].Stop();  }  }  }  public void PlayMusic(int musicToPlay)  {  for (int i = 0; i < misic.Length; i++)  {  misic[i].Stop();  }  misic[musicToPlay].Play();  }  public void SetMusicLevel()  {  musicMixer.audioMixer.SetFloat("MusicVol", UIManager.instance.musicVolSlider.value);  }  public void PlaySFX(int sfxToPlay)  {  //инача  sfx[sfxToPlay].Play();  }  public void SetSFXLevel()  {  sfxMixer.audioMixer.SetFloat("SfxVol", UIManager.instance.sfxVolSlider.value);  }  } |
| CoinPickup | using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class CoinPick : MonoBehaviour  {  // Start is called before the first frame update  public bool isFullCoin ;  public GameObject deathEffect;  void Start()  {    }  public int value;  // Update is called once per frame  void Update()  {    }  private void OnTriggerEnter(Collider other)  {  if (other.tag == "Player")  {  Destroy(gameObject);  GameManager.instance.AddCoins(value);  Instantiate(deathEffect, PlayerController.instance.transform.position + new Vector3(0f, 1f, 0f), PlayerController.instance.transform.rotation);  AudioManager.instance.PlaySFX(4);  }  }  } |

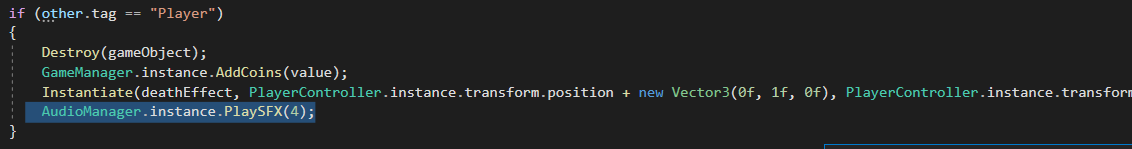
1. Создаем объект и помещаем туда все звуковые объекты



1. Помещаем каждую музыку в созданный массив



1. Добавляем звук на монеты. Выбираем какой звук нам нужен

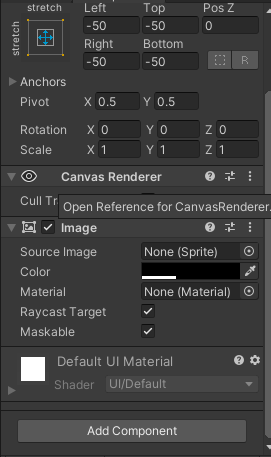


1. Проделываем все те же действия на остальные объекты

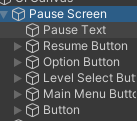
## Меню паузы

1. Добавляем панель и настраиваем

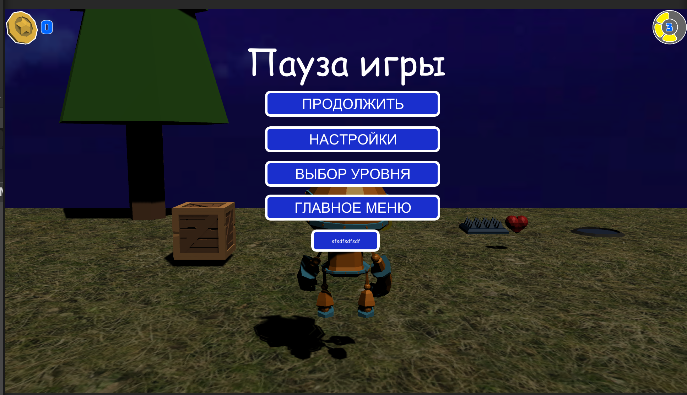




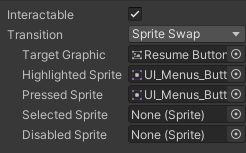
1. Добавляем в нее объекты и размещаем так как мы хотим



1. Конечный результат

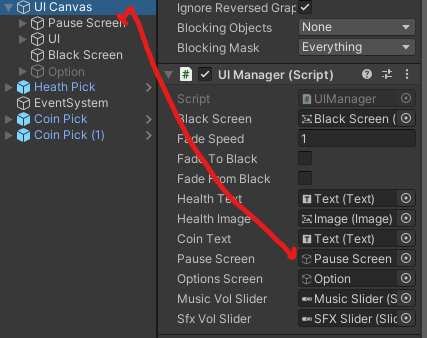


1. Добавляем действия при наведение и нажатии на кнопки будет менять цвет

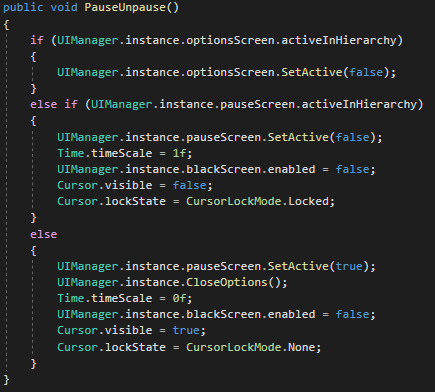


### Приостановка игры

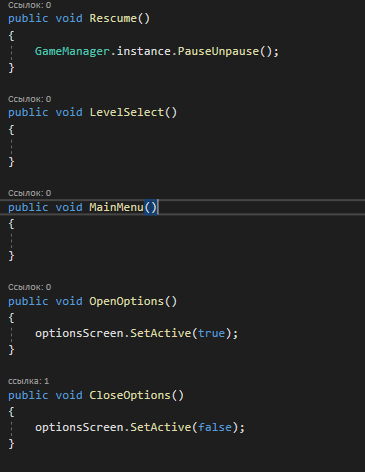
1. Созадем в ней переменную для окна паузы и привязываем к ней через инспектор нашу панель с Pause Screen.



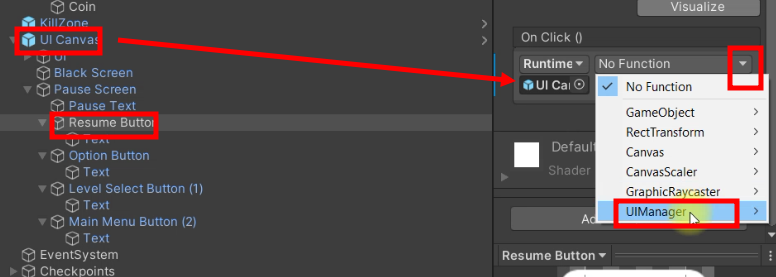
1. В Game Manager делаем функцию для вызова окна паузы

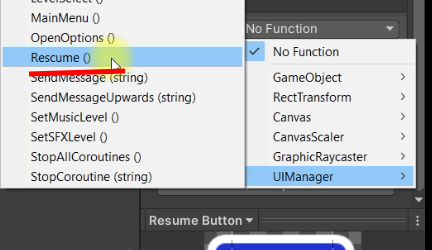


1. В UI Manager делаем функции обработки нажатий кнопок



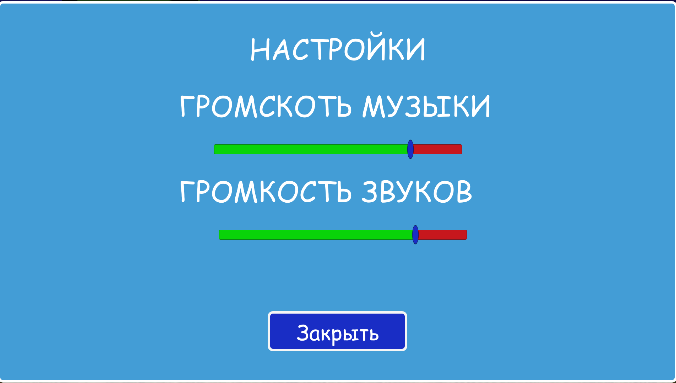
1. Накладываем созданные функции на кнопки



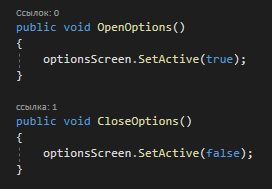


### Меню опций

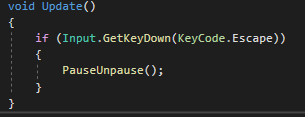
1. Создаём такое же окошко как и меню паузы и помещаем на данную панель slider



1. Создаем функция для открытия и закрытия окна настройки



GameManager

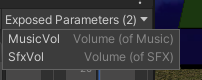


### Управление громкостью через пользовательский интерфейс

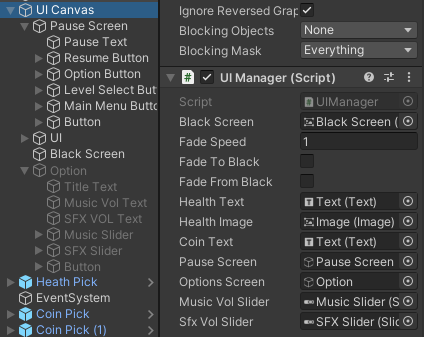
1. добавляем переменные public AudioMixerGroup musicMixer, sfxMixer; После связываем их с элементами микшера в самом Unity



1. Создаем параметры для работы с микшером

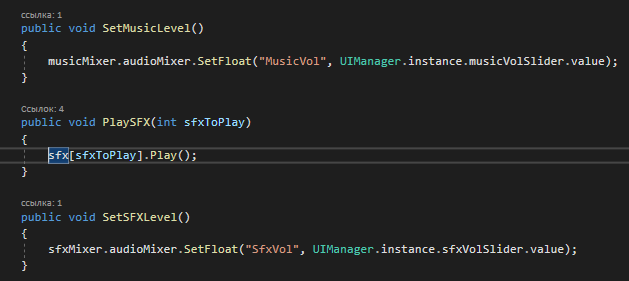


1. Привязка слайдера

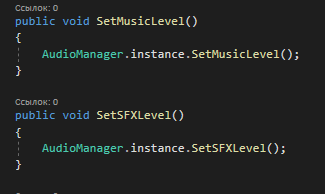


1. Листинг кода

AudioManager

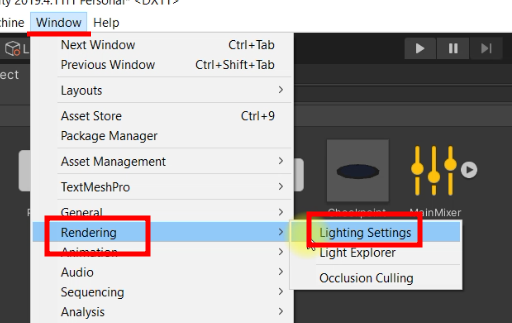


UIManager



### Настройка skybox.

1. Ищем настройку, чтобы изменить Skybox



1. Ищем в магазине моделей нужное нам небо, либо берем его в папке с моделями и добавляем в это окно настройки.

