Описание

Проект является игрой в жанре «2Dплатформер». Главный герой(ГГ), управляемый игроком, имеет возможность перемещаться (вправо, влево), прыгать, а также стрелять, причем при зажатии кнопки стрельбы, мощность выстрела будет увеличиваться, а при ее отпускании, будет происходить сам выстрел. На уровне находятся враги, имеющие три режима поведения, режим патрулирования, в котором противники перемещаются по области, боевой режим, в котором они ведут огонь по ГГ, а также режим боевой готовности, при котором враги стоят на месте, и ожидают появления ГГ. В проекте присутствуют ловушки: наносящие урон, такие как шипы, вращающийся топор; падающие плиты. Простые объекты: ящик, аптечка.

Основные механики

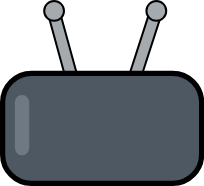
* 1. Персонаж
     + Перемещение [1.1]
     + Прыжок [1.2]
     + Стрельба (с накоплением энергии выстрела) [1.2]
  2. Противник
     + Если «видит» персонажа, открывает по нему огонь [2.1]
     + Патрулирует область, от препятствия до препятствия [2.2]
     + При потере персонажа из вида и нахождении персонажа все поля «видимости», переходит в режим патрулирования [2.3]
     + При смерти, есть шанс выпадения случайного предмета [2.4]
  3. Турель
     + Если ГГ в зоне атаки, начинает его отлеживать
     + Если между ГГ и турелью нет препятствий, и ГГ находится в зоне действия, турель атакует ГГ
     + При выходе из зоны атаки, турель превращает отслеживание ГГ
  4. Объекты
     + Ящик
       - Обладает физикой [3.1]
       - При уничтожении выпадает заложенный предмет [3.2]
       - При уничтожении разлетается на осколки [3.3]
     + Шипы
       - При касании наносят урон ГГ [3.4]
       - При касании Отталкивают ГГ от себя [3.5]
     + Топор
       - Движется по дуге [3.6]
       - При касании наносит урон ГГ [3.7]
       - При касании Отталкивает ГГ от себя [3.8]
       - Уничтожает ящики [3.9]
     + Падающая платформа
       - Через время n, после касания с ГГ, падает вниз [3.10]
     + Аптечка
       - Восполняет жизни ГГ [3.11]
  5. Интерфейс
     + Полоска жизней ГГ [4.1]
     + Меню паузы [4.2]
     + Игровое меню [4.3]
     + Настройки [4.4]
  6. Фон
     + Двигается в след за ГГ, давая ощущение его не статичности и отдалённости.

Графика

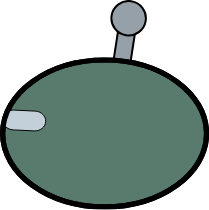
# Земля:



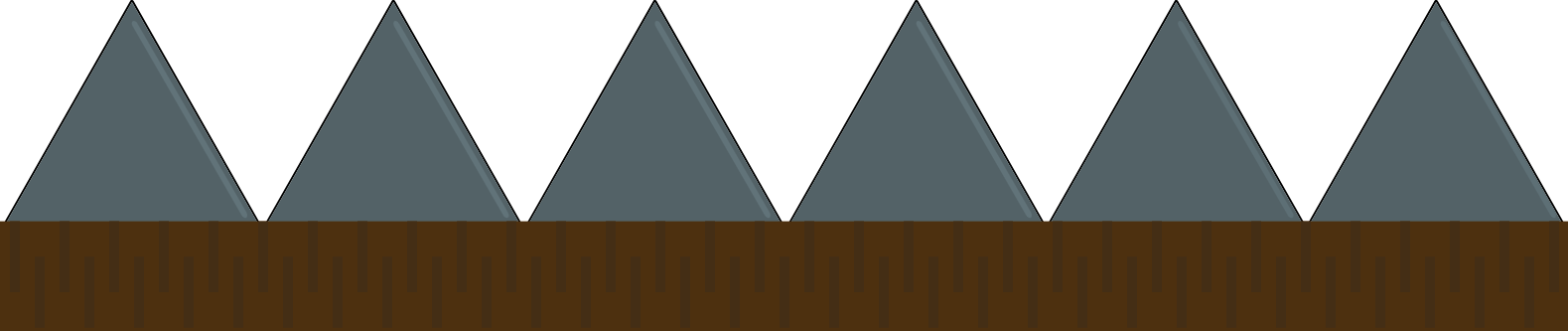
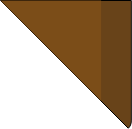
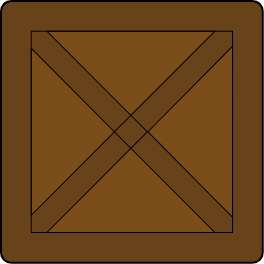
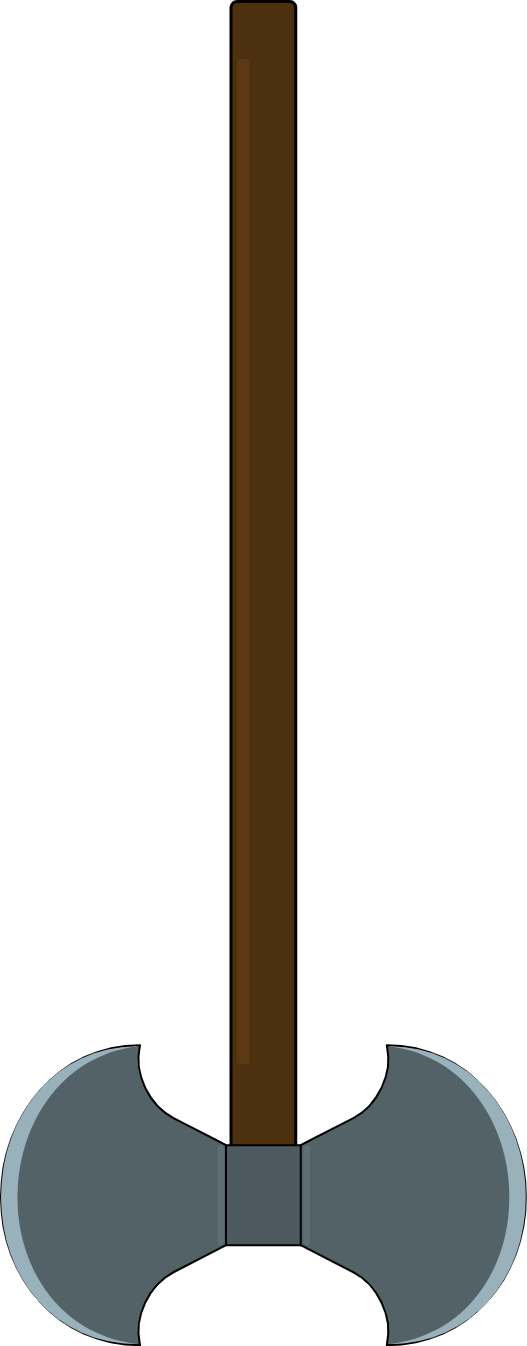
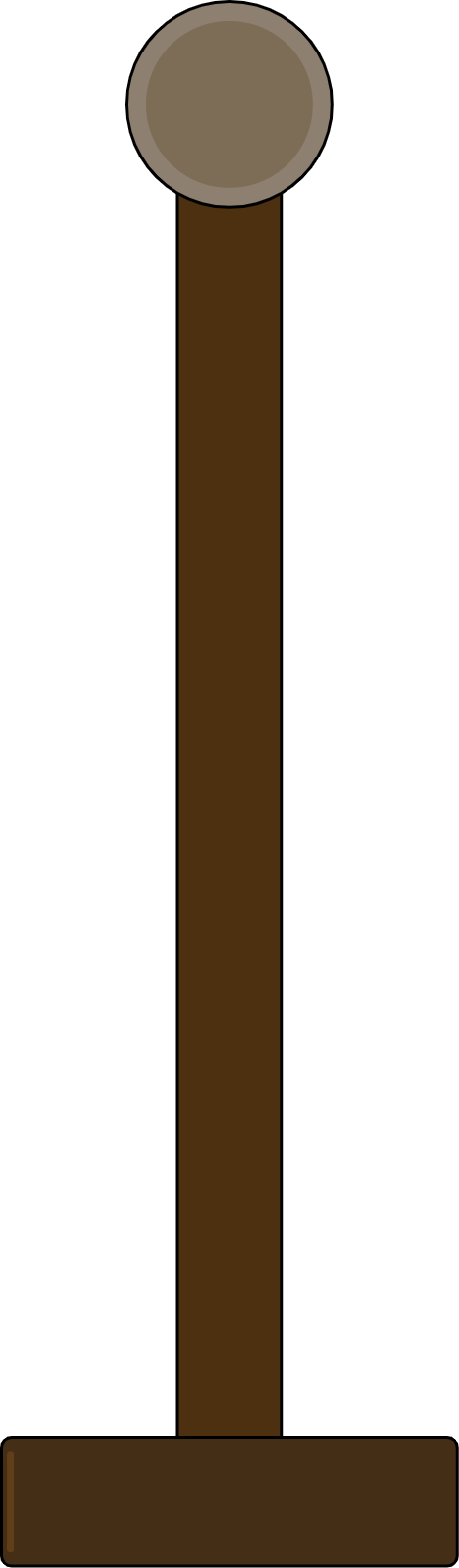
# Робот(ГГ)



# Робот(Враг)



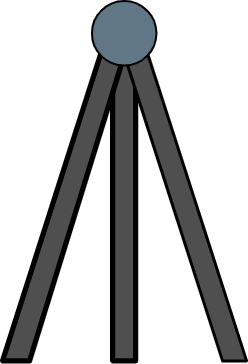
# Объекты:



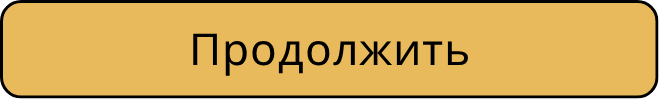
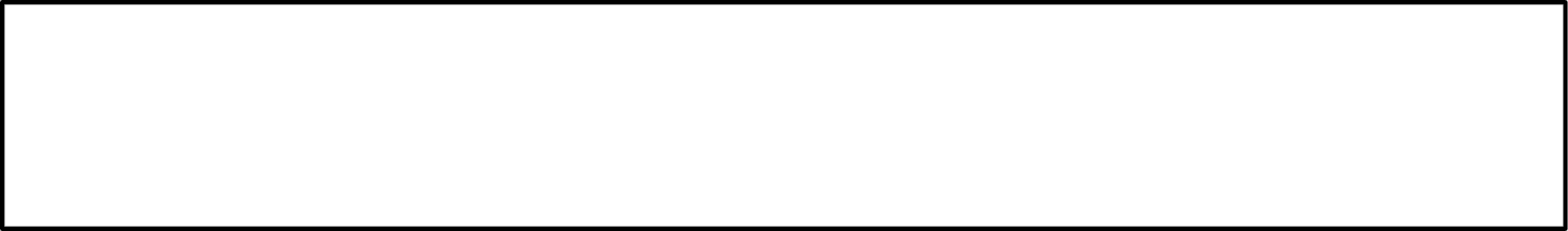
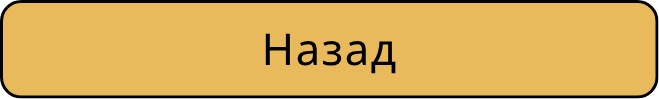
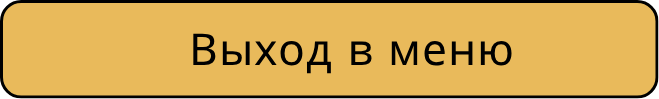
# Фон:



# Турель:



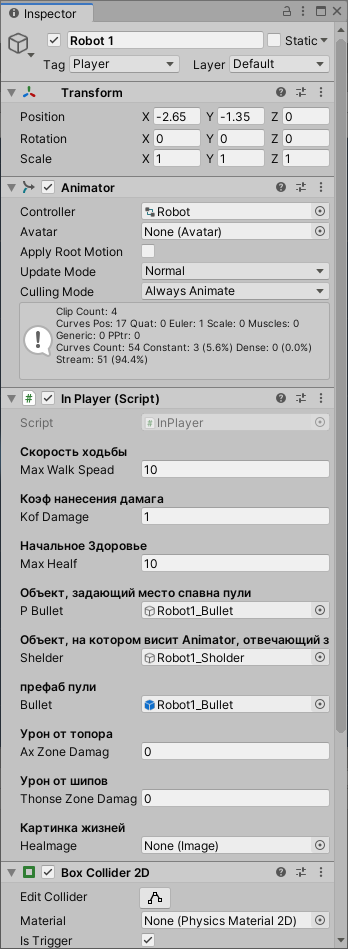
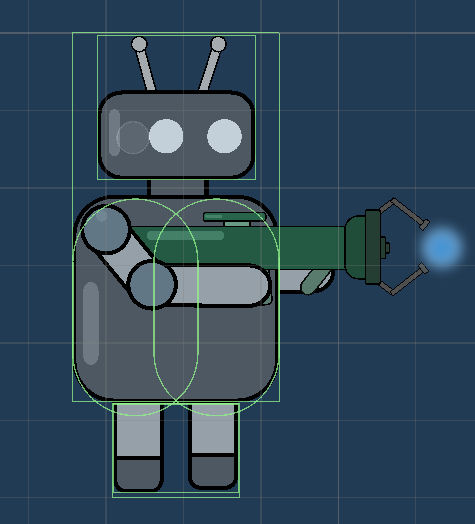
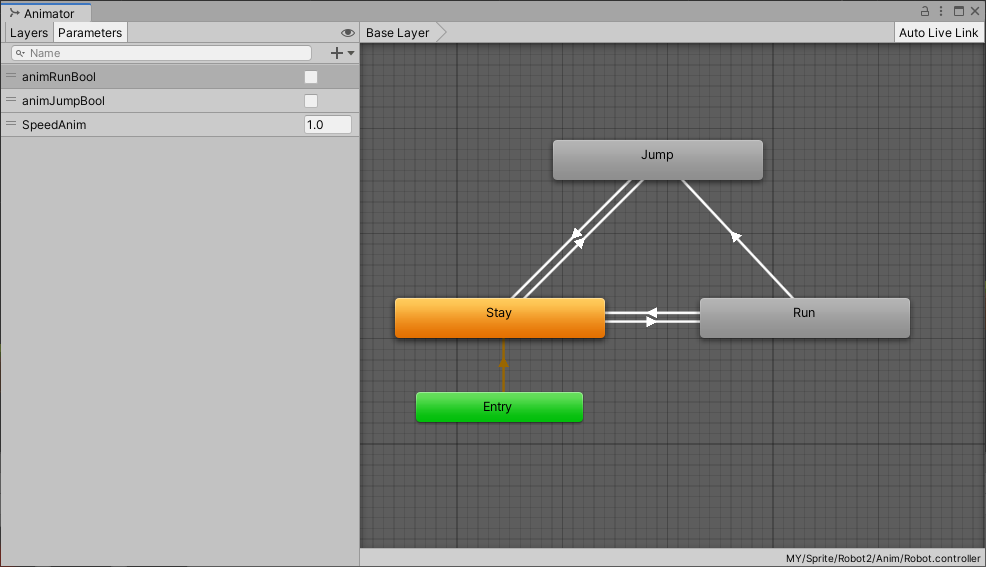
# Интерфейс:



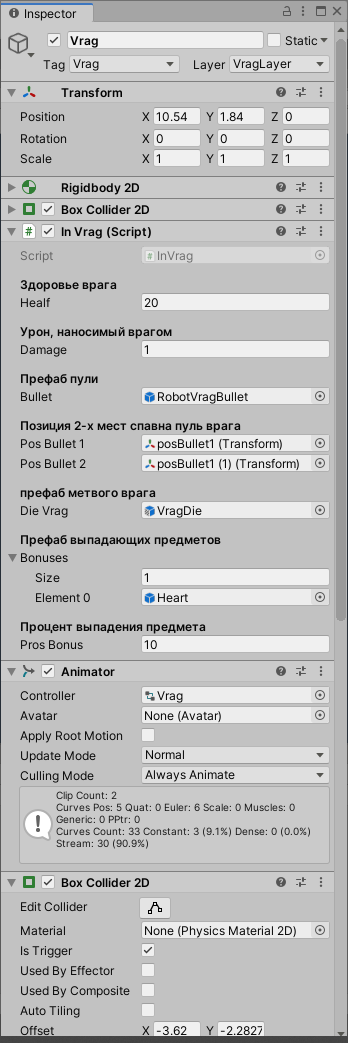
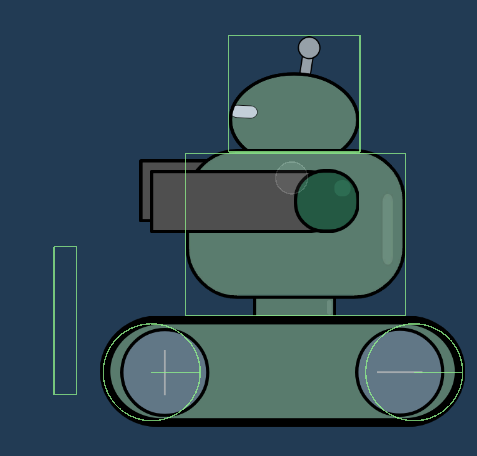
Ход работы

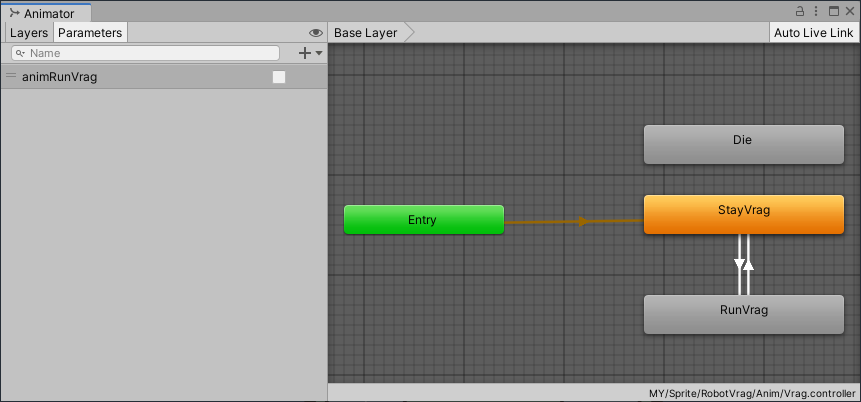
В первую очередь, на камеру, созданную на сцене, вешаем скрипт «CameraScript», который будет вести камеру за ГГ, сглаживая резкие перемещения.

Собираем из спрайтов персонажа(робота), создаем ему анимации бега, прыжка, стрельбы, смерти. Вешаем на объект, коллайдеры, задающие его геометрию, и триггеры, контролирующие нахождение на земле, и взаимодействие с другими объектами. Вешаем на него скрипт «InPlayer». Назначаем тег «Player»

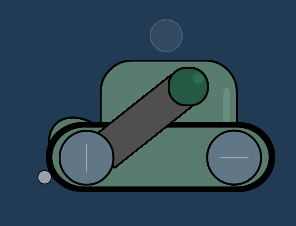
  

Собираем из спрайтов Врага, создаем ему анимации езды и смерти. Вешаем на него коллайдеры, описывающие его геометрию и триггеры, реализующие его взаимодействие с окружающими объектами. Вещаем на него скрипт «InVrag». Назначаем тег «Vrag»

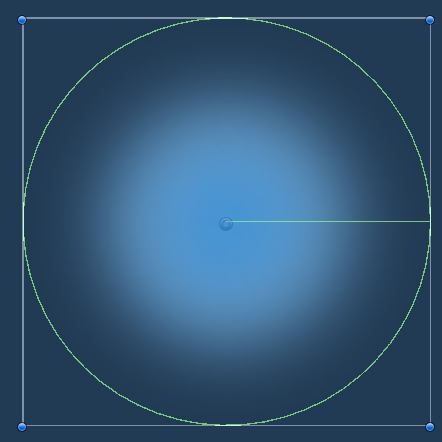
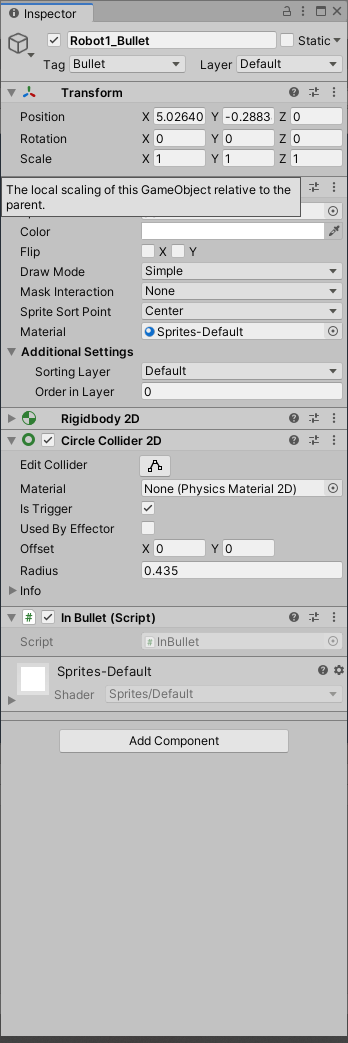
 



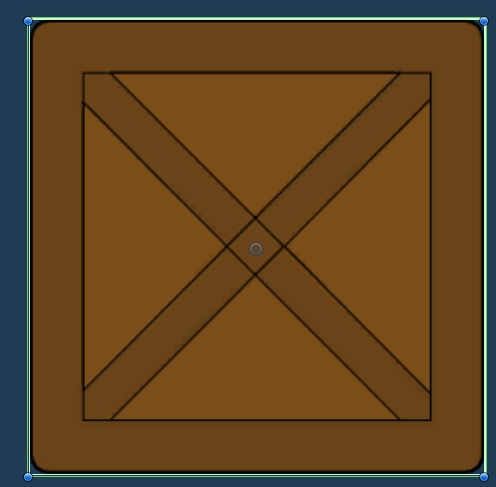
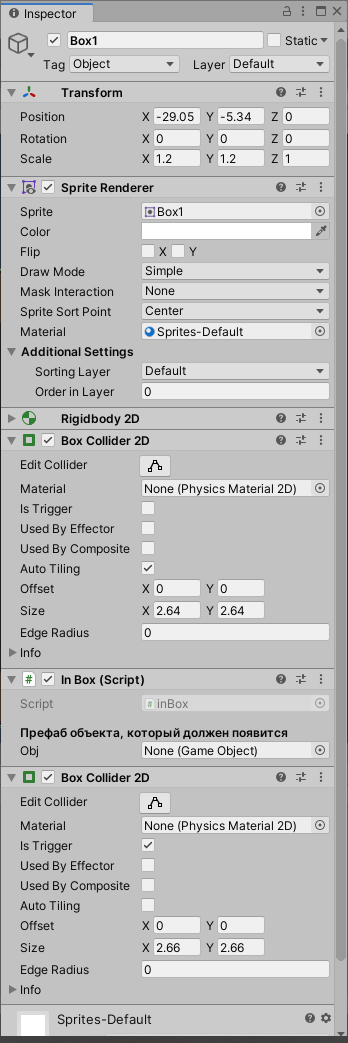
Из спрайтов врага собираем объект мертвого врага, он появляется после смерти противника, и представляет из себя просто изображение.



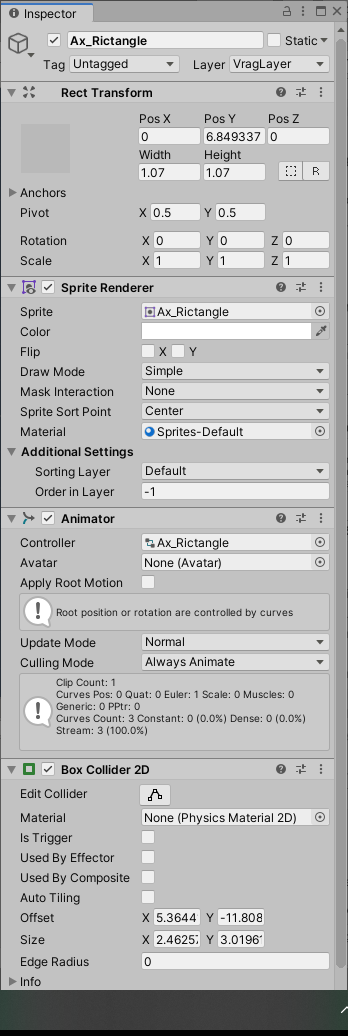
Создаем два вида «пуль» (для ГГ и врага). Вешаем на них триггер, и скрипт «InBullet». Назначаем тег «Bullet»



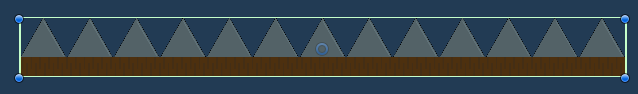
Создаем ящик, вешаем на него коллайдер и скрипт «inBox». Назначаем тег «Box»



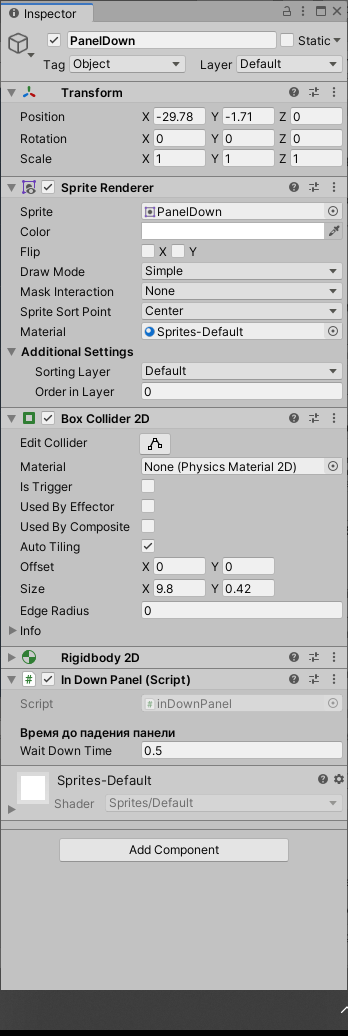
Создаем топор, вешаем на него коллайдеры и задаем анимацию движения. Назначаем тег «Ax»

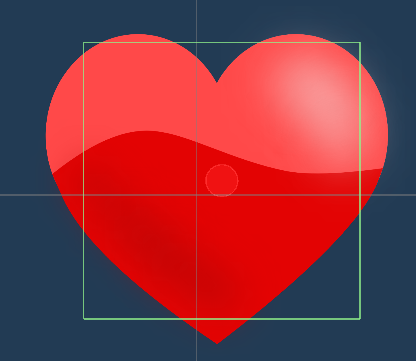
Создаем шипы, вешаем на них коллайдер. Назначаем тег «Thonse»



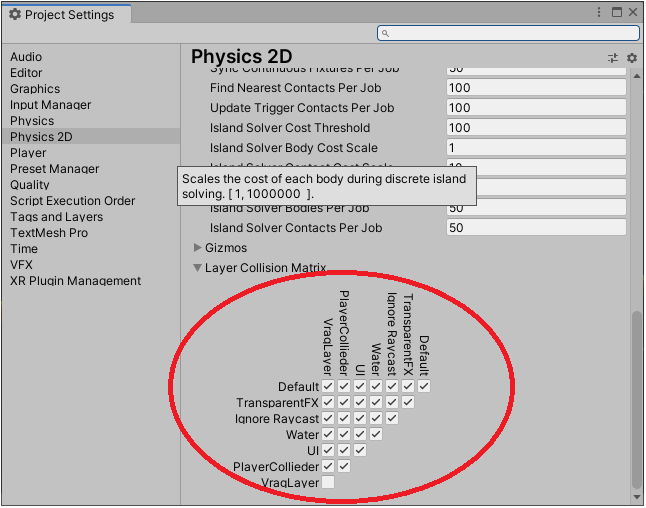
Создаем падающую панель, вешаем на нее коллайдер и скрипт «inDownPanel». Назначаем тег «DownPanel»



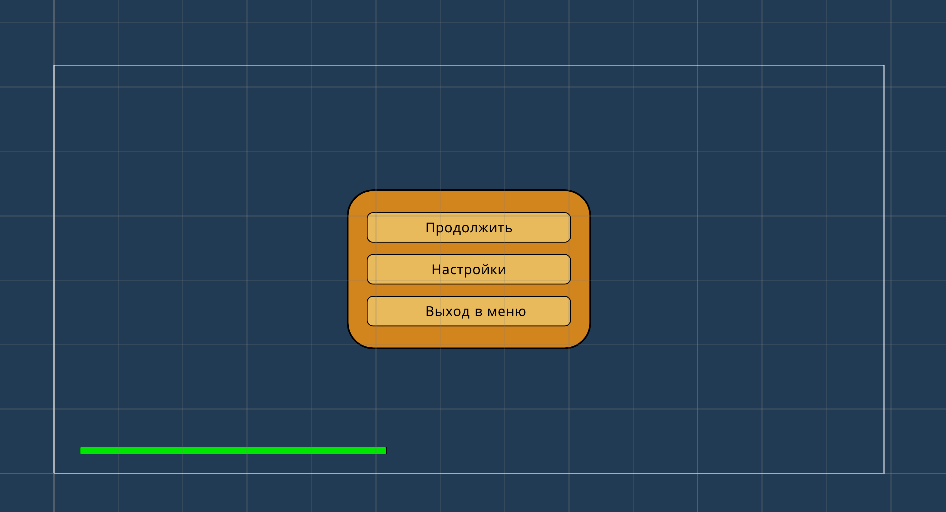
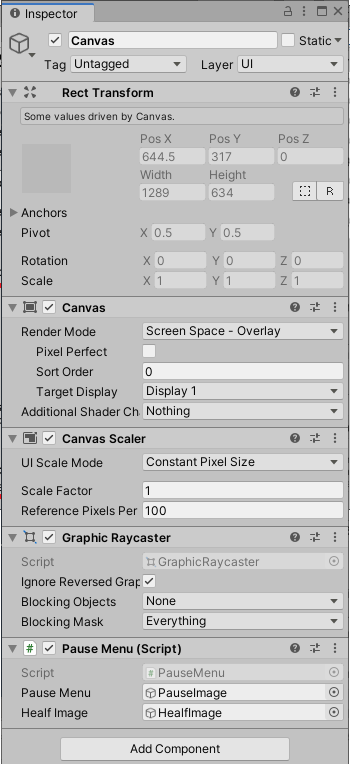
Создаем аптечку, задаем анимация движения, вешаем на нее триггер. Назначаем тег «Heart»



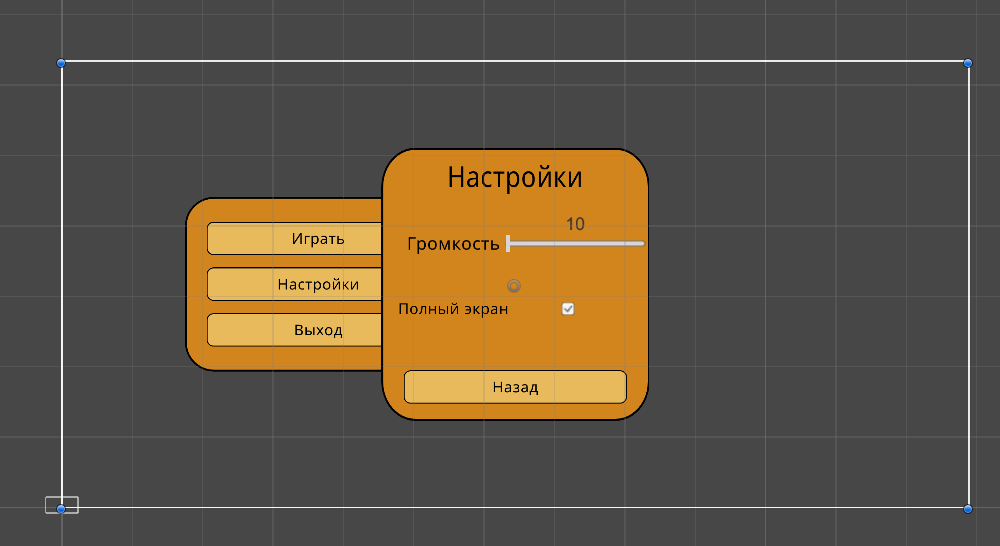
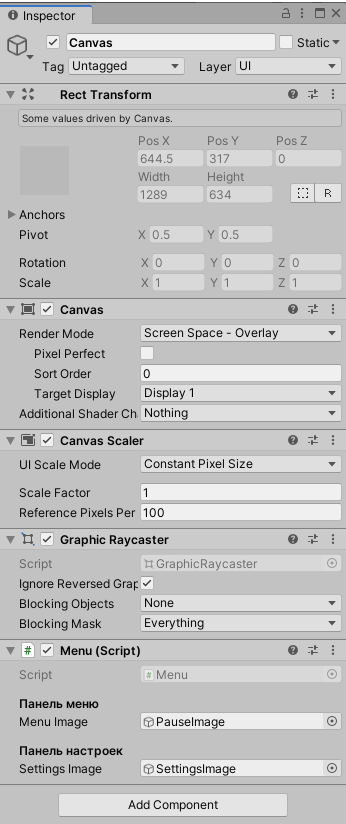
Задаем взаимодействие layer-ов. layer-ы «Vrag» не должны взаимодействовать друг с другом.

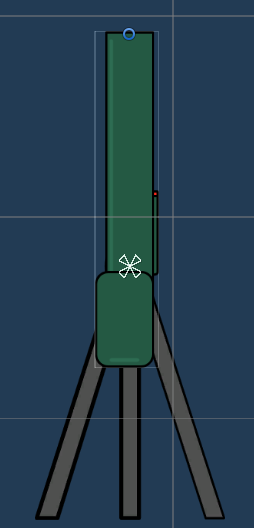
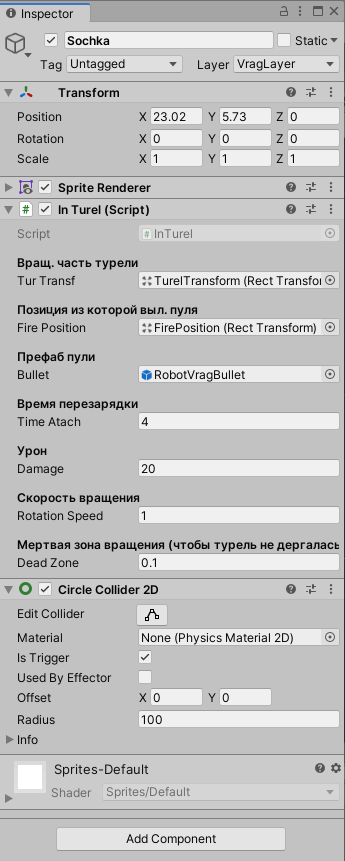


Создаем Convas, на который крепим полоску жизней, меню паузы и скрипт «PauseMenu». Назначаем события нажатий кнопок на нужные методы скрипта «PauseMenu».



Создаем отдельную сцену, на которой создаем Convas, на который крепим меню и скрипт «Menu». Назначаем события нажатий кнопок на необходимые методы скрипта «Menu».





С помощью созданных объектов, проектируем Уровеь1.

Скрипты

# CameraScript

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CameraScript : MonoBehaviour

{

public float damping = 1.5f;

public Vector2 offset = new Vector2(2f, 1f);

public bool faceLeft;

private Transform player;

private int lastX;

void Start()

{

offset = new Vector2(Mathf.Abs(offset.x), offset.y);

FindPlayer(faceLeft);

}

public void FindPlayer(bool playerFaceLeft)

{

player = GameObject.FindGameObjectWithTag("Player").transform;

lastX = Mathf.RoundToInt(player.position.x);

if (playerFaceLeft)

{

transform.position = new Vector3(player.position.x - offset.x, player.position.y + offset.y, transform.position.z);

}

else

{

transform.position = new Vector3(player.position.x + offset.x, player.position.y + offset.y, transform.position.z);

}

}

void Update()

{

if (player)

{

int currentX = Mathf.RoundToInt(player.position.x);

if (currentX > lastX) faceLeft = false; else if (currentX < lastX) faceLeft = true;

lastX = Mathf.RoundToInt(player.position.x);

Vector3 target;

if (faceLeft)

{

target = new Vector3(player.position.x - offset.x, player.position.y + offset.y, transform.position.z);

}

else

{

target = new Vector3(player.position.x + offset.x, player.position.y + offset.y, transform.position.z);

}

Vector3 currentPosition = Vector3.Lerp(transform.position, target, damping \* Time.deltaTime);

transform.position = currentPosition;

}

}

}

# InPlayer

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using System.Linq;

//using UnityEngine.SceneManagement;

public class InPlayer : MonoBehaviour

{

#region Настраиваемые характеристики

[Header("Cкорость ходьбы")]

public float maxWalkSpead = 10f;

[Header("Коэф нанесения дамага")]

public int kofDamage;

[Header("Начальное Здоровье")]

public int maxHealf;

[Header("Объект, задающий место спавна пули")]

public GameObject pBullet;//Объект, задающий место спавна "пули"

[Header("Объект, на котором висит Animator, отвечающий за оружие")]

public GameObject shelder;//Объект, на котором "висит" Animator, отвечающий за оружие

[Header("префаб пули")]

public GameObject bullet;//префаб пули

[Header("Урон от топора")]

public int axZoneDamag;

[Header("Урон от шипов")]

public int thonseZoneDamag;

[Header("Картинка жизней")]

[SerializeField]

Image healmage;

#endregion

#region Переменные

private GameObject convas;

private int healf;

readonly string[] trueJumpTag= {"Ground","Vrag", "Object" };//теги с котором разрешено взаимодейтивие прыжка

private float move=1;//показывает направление(если игрок мертв то должен быть равен 0)

#endregion

#region Флаги

public bool stopMovingFlag=true;//отключить передвижение

bool replaceFlag = true;//флаг, контролирующий завершение перезарядки

private sbyte jumpFlag =0;//флаг, разрешающий прыжок

bool facingRight = true;//поворот в право

bool DamageFlag = true;//поворот в право

#endregion

/// <summary>

/// При запуске скрипта

/// </summary>

void Start()

{

healf = maxHealf;// заполняем шкалу здоровья

transform.localScale = new Vector2(0.1f, 0.1f);

StartCoroutine(StartTransport());

convas = GameObject.FindGameObjectWithTag("Convas");

}

/// <summary>

/// Обновление происходит каждый кадр

/// </summary>

void Update()

{

#region Атака

if (Input.GetKeyDown(KeyCode.Space))//если нажат пробел

{

shelder.GetComponent<Animator>().SetBool("shootBool", true);//запускаем анимацию увеличения пули

}

if (Input.GetKeyUp(KeyCode.Space)&& replaceFlag)// Если пробел отпущен, и перезарядка завершена

{

GameObject Bullet = Instantiate(bullet, pBullet.transform.position, pBullet.transform.rotation);//создаем копию префаба "пули" на месте обозначенном pBullet

Bullet.transform.localScale = new Vector2(pBullet.transform.localScale.x, pBullet.transform.localScale.y);//задаем ему размер, соответствующий размеру пули в анимации "зарядки"

Bullet.GetComponent<Rigidbody2D>().velocity = new Vector2(facingRight ? 100f : -100f, Bullet.GetComponent<Rigidbody2D>().velocity.y);//задаем "пуле скорость, с которой она будет двигаться"

Bullet.GetComponent<InBullet>().damage = kofDamage \* (int)(pBullet.transform.localScale.x\* pBullet.transform.localScale.x \* pBullet.transform.localScale.x);// увеличивае урон, передываемый пуле

shelder.GetComponent<Animator>().SetBool("shootBool", false);//запускаем анимацию перезарядки

replaceFlag = false;//поднимаем флаг начала перезарядки

StartCoroutine(Replace());//запускаем карутину отмеряющую время перезарядки

}

#endregion

#region Прекращение движения

if (Input.GetKeyUp(KeyCode.A) || Input.GetKeyUp(KeyCode.D) || stopMovingFlag)//в момент отпускания кнопок движения

{

GetComponent<Rigidbody2D>().velocity = new Vector2(0, GetComponent<Rigidbody2D>().velocity.y);// останавливаем ГГ

GetComponent<Animator>().SetBool("animRunBool", false);//прекращаем анимацию бега

}

#endregion

#region Начало прыжка

if (Input.GetKeyDown(KeyCode.W)&&jumpFlag>0&& !stopMovingFlag)//если нажата кнопка и разрешен прыжок

{

//jumpFlag=0;// запрещаем прыжок

GetComponent<Rigidbody2D>().velocity = new Vector2(GetComponent<Rigidbody2D>().velocity.x, 20f);// задаем силу прыжка

GetComponent<Animator>().SetBool("animJumpBool", true);// запускаем анимацию начала прыжка

}

#endregion

#region Отслеживание падения

if (GetComponent<Rigidbody2D>().velocity.y < 0)//если ГГ падает

{

GetComponent<Animator>().SetFloat("SpeedAnim", -1);//воспроизводим анимация прыжка назад

}

else

{

GetComponent<Animator>().SetFloat("SpeedAnim", 1);//если нет падения, то ставим скорость нормальной

}

#endregion

}

private void FixedUpdate()

{

if ((Input.GetKey(KeyCode.A) || Input.GetKey(KeyCode.D))&&!stopMovingFlag)//если зажата клавиша движения

{

move = Input.GetAxis("Horizontal");

GetComponent<Rigidbody2D>().velocity = new Vector2(move \* maxWalkSpead, GetComponent<Rigidbody2D>().velocity.y);//задаем ГГ скорость и направление для движения

GetComponent<Animator>().SetBool("animRunBool", true);//запускаем анимацию бега

#region Отслеживание поворота

if (move > 0 /\*&& !Die\*/&& !facingRight)//если ГГ двигается направо и повернут налево

{

Flip();

}

else if (move < 0 /\*&& !Die\*/&& facingRight)//елси ГГ двигается налево и повернут направо

{

Flip();

}

#endregion

}

}

private void OnTriggerEnter2D(Collider2D other)//если в тригер входит другой collider

{

if (trueJumpTag.Contains(other.gameObject.tag) && !other.isTrigger)//если это земля

{

jumpFlag++;//разрешаем выполнить прыжок

GetComponent<Animator>().SetBool("animJumpBool", false);//воспроизводим выход из анимации прыжка

}

if (other.gameObject.tag == "Ax"&& DamageFlag)//если это шипы

{

GetComponent<Rigidbody2D>().velocity = new Vector2(facingRight ? (-20f\*Mathf.Cos(other.transform.rotation.z)) : ( 20f \* Mathf.Cos(other.transform.rotation.z)), Mathf.Abs(50f \* Mathf.Sin(other.transform.rotation.z)));// откидываем от шипов

Damage(axZoneDamag);// наносим урон

DamageFlag = false;// на время запрещаем наносить урон

StartCoroutine(DamageTrue());//отчитывыем время, во время которого нельзя наносить урон

}

if (other.gameObject.tag == "Thonse" && DamageFlag)//если это шипы

{

jumpFlag=0;

GetComponent<Rigidbody2D>().velocity = new Vector2(GetComponent<Rigidbody2D>().velocity.x, 15f);// откидываем от шипов

Damage(thonseZoneDamag);// наносим урон

DamageFlag = false;// на время запрещаем наносить урон

StartCoroutine(DamageTrue());//отчитывыем время, во время которого нельзя наносить урон

}

if (other.gameObject.tag == "Portal")//если это финиш

{

StartCoroutine(FinishTransport());

stopMovingFlag = true;

}

if (other.gameObject.tag == "Lift" && other.isTrigger)//если это финиш

{

stopMovingFlag = true;

GetComponent<Animator>().SetBool("animJumpBool", false);//воспроизводим выход из анимации прыжка

}

if (other.gameObject.tag == "DieZone")//если это зона смерти

{

Die();

}

if (other.gameObject.tag == "Heart"&& healf<maxHealf)//если это жизнь

{

healf = maxHealf;//востанавливаем здоровье

UpdateCrossBar();//обнавляем полоску жизней

Destroy(other.gameObject);//уничтожаем аптечку

}

}

private void OnTriggerExit2D(Collider2D other)

{

if (trueJumpTag.Contains(other.gameObject.tag)&& !other.isTrigger)//если это земля(если оторвались от земля, то прыгать нельзя)

{

jumpFlag--;//запрещаем выполнять прыжок

}

if (other.gameObject.tag == "Lift"&&other.isTrigger)//если это финиш

{

stopMovingFlag = false;

}

}

/// <summary>

/// Переворот ГГ

/// </summary>

void Flip()//переворот ГГ

{

facingRight = !facingRight;//флаг поворота

Vector2 theScale = transform.localScale;//сохраняем размеры

theScale.x \*= -1;//отражаем объект

transform.localScale = theScale;//задаем новый размер(отраженный по х)

}

/// <summary>

/// Получение урона

/// </summary>

/// <param name="\_damag"></param>

public void Damage(int \_damag)

{

healf -= \_damag;//наносим урон

if (healf <= 0)//если жизни закончились

Die();//функция начала смерти

UpdateCrossBar();//обновляем полоску жизней

#region Смена цвета

foreach (var child in GetComponentsInChildren<SpriteRenderer>())//перебираем все дочерние объекты

child.gameObject.GetComponent<SpriteRenderer>().material.color = new Color(0.9f, 0.5f, 0.5f);//и задаем им красноватый оттенок

StartCoroutine(ColorTransformBack());//ззапускаем картутину возвращение цвета

#endregion

}

/// <summary>

/// Смерть

/// </summary>

void Die()

{

stopMovingFlag = true;

GetComponent<Animator>().Play("Die");

}

public void EndDie()

{

convas.GetComponent<PauseMenu>().OnRestartPressed();

}

/// <summary>

/// Обновление полоски жизней

/// </summary>

private void UpdateCrossBar()

{

healmage.fillAmount = (float)((float)healf / (float)maxHealf);//уменьшаем размер спрайта

if (healf < 3 \* maxHealf / 4)//если она < 3/4 красим ее в желтый цвет

healmage.color = Color.yellow;

else if (healf < maxHealf / 3)//если она < 1/3 красим ее в красный цвет

healmage.color = Color.red;

else

healmage.color = Color.green;//иначе она зеленная (> 3/4)

}

/// <summary>

/// Карутина задающ. время на перезарядку

/// </summary>

/// <returns></returns>

IEnumerator Replace() // карутина

{

yield return new WaitForSeconds(0.7f);//ждем 0.7 сек

replaceFlag = true;//разрешаем повторный выстрел

}

/// <summary>

/// Возвращает цвет через 100мс

/// </summary>

/// <returns></returns>

IEnumerator ColorTransformBack() // карутина

{

yield return new WaitForSeconds(0.1f);//ждем 0.1 сек

foreach (var child in GetComponentsInChildren<SpriteRenderer>())//перебираем все дочерние объекты

child.gameObject.GetComponent<SpriteRenderer>().material.color = new Color(1f, 1f, 1f);//и возвращаем им стандартный цвет

}

/// <summary>

/// карутина, разрешающая дамаг(от объектов)

/// </summary>

/// <returns></returns>

IEnumerator DamageTrue() // карутина

{

yield return new WaitForSeconds(0.5f);//ждем 0.5 сек

DamageFlag = true;//разрешаем нанесение урона

}

IEnumerator FinishTransport() // карутина

{

yield return new WaitForFixedUpdate();

transform.localScale = new Vector2(transform.localScale.x- 0.005f, transform.localScale.y- 0.005f);

if (transform.localScale.x < 0.1)

{

//SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);

convas.GetComponent<PauseMenu>().GoNextLVL();

}

else

StartCoroutine(FinishTransport());

}

IEnumerator StartTransport() // карутина

{

yield return new WaitForFixedUpdate();

transform.localScale = new Vector2(transform.localScale.x + 0.02f, transform.localScale.y + 0.02f);

if (transform.localScale.x < 1f)

StartCoroutine(StartTransport());

else

{

transform.localScale = new Vector2(1f, 1f);

stopMovingFlag = false;

}

}

}

# InVrag

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using UnityEngine;

public class InVrag : MonoBehaviour

{

#region Настраиваемые переменные

[Header("Здоровье врага")]

[SerializeField]

private int healf;

[Header("Урон, наносимый врагом")]

public int damage;

[Header("Префаб пули")]

[SerializeField]

private GameObject bullet;

[Header("Позиция 2-х мест спавна пуль врага")]

[SerializeField]

private Transform posBullet1;

[SerializeField]

private Transform posBullet2;

[Header("префаб метвого врага")]

[SerializeField]

private GameObject dieVrag;//префаб картинки метвого врага(выставляется после смерти)

[Header("Префаб выпадающих предметов")]

public GameObject[] bonuses= new GameObject[1];

[Header("Процент выпадения предмета")]

public float prosBonus;

#endregion

#region Флаги

private bool atachFlag1 = true;//флаг, разрешающий атаку 1 пушки(ближней)

private bool atachFlag2 = false;//флаг, разрешающий атаку 2 пушки(дальней)

bool facingRight = false;//поворот в право

bool stay = true;//Дивжется ли вграг

#endregion

#region Переменные

private float move = -1f;//показывает направление(если игрок мертв то должен быть равен 0)

GameObject player;

#endregion

void Start()

{

player = GameObject.FindGameObjectWithTag("Player");// задаем для объекта player объект с тегом Player(игрока)

}

void Update()

{

}

/// <summary>

/// получение урона

/// </summary>

/// <param name="\_damag"></param>

public void Damage(int \_damag)

{

healf -= \_damag;//уменьшаем жизни

if (healf <= 0)

Die(); //если жизни закончились, враг мертв

#region Изменение цвета

foreach (var child in GetComponentsInChildren<SpriteRenderer>())//перебираем все дочерние объекты

child.gameObject.GetComponent<SpriteRenderer>().material.color = new Color(0.9f, 0.5f, 0.5f);//и задаем им красноватый оттенок

StartCoroutine(ColorTransformBack());//ззапускаем картутину возвращение цвета

#endregion

#region Поворот в сторону игрока

if (player.transform.position.x <= transform.position.x)//поворачиваем в сторону игрока

{

if (facingRight)

Flip();

}

else

{

if (!facingRight)

Flip();

}

#endregion

}

/// <summary>

/// Смерть врага

/// </summary>

private void Die()

{

move = 0f;//запрещаем движение

gameObject.GetComponent<Animator>().Play("Die");//проигрываем анимацию смерти(в конце которой установлен event на метод EndDie)

}

/// <summary>

/// Уничтожение объекта

/// </summary>

void EndDie()

{

for(int i=0; i < bonuses.Length; i++)//перебираем все элементы в массиве бонусов

{

if(Random.Range(0f, 99f)< prosBonus)//если вероятность совпала с i бонусом

{

Instantiate(bonuses[i], transform.position, transform.rotation);//спавним этот бонус

break;//прекращаем перебор

}

}

Instantiate(dieVrag, transform.position, transform.rotation);//выставляем статичную картинку мертвого врага

Destroy(gameObject);//уничтожаем объект

}

private void FixedUpdate()

{

try

{

if (Vector2.Distance(transform.position, player.transform.position) < 60f && move != 0f)// если ГГ находится в зоне поражения и не мертв

{

RaycastHit2D hit = Physics2D.Raycast(posBullet1.position, (facingRight ? Vector2.right : Vector2.left), 60f, 7);//кидаем рейкаст по напровлению взгляда

if (hit.collider.gameObject == player)//если он попал в игрока

{

stay = false;//запрещаем двигаться дальше

GetComponent<Animator>().SetBool("animRunVrag", false);//прекращаем анимацию бега

if (atachFlag1)//если разрешена атака 1 пушки

{

GameObject Bullet1 = Instantiate(bullet, posBullet1.position, posBullet1.rotation);//создаем копию префаба "пули" на месте обозначенном posBullet1

Bullet1.GetComponent<Rigidbody2D>().velocity = new Vector2(move \* 20f, Bullet1.GetComponent<Rigidbody2D>().velocity.y);//задаем "пуле скорость, с которой она будет двигаться"

Bullet1.GetComponent<InBullet>().Damage = damage;//Передаем пуле урон

atachFlag1 = false;//запрещаем дальше атаковать

StartCoroutine(WaitAtach1());//запускаем перезарядку 1 пушки

}

if (atachFlag2)//если разрешена атака 2 пушки

{

GameObject Bullet2 = Instantiate(bullet, posBullet2.position, posBullet2.rotation);//создаем копию префаба "пули" на месте обозначенном posBullet2

Bullet2.GetComponent<Rigidbody2D>().velocity = new Vector2(move \* 20f, Bullet2.GetComponent<Rigidbody2D>().velocity.y);//задаем "пуле скорость, с которой она будет двигаться"

Bullet2.GetComponent<InBullet>().Damage = damage;//Передаем пуле урон

atachFlag2 = false;//запрещаем дальше атаковать

}

GetComponent<Rigidbody2D>().velocity = new Vector2(0f, GetComponent<Rigidbody2D>().velocity.y);//останавливаем врага

}

}

else

stay = true;//если игрока нет в зоне, продолжаем двигаться

}

catch

{

}

if (stay)//если разрешено движение

{

GetComponent<Rigidbody2D>().velocity = new Vector2(move \* 5f, GetComponent<Rigidbody2D>().velocity.y);//задаем ГГ скорость и направление для движения

GetComponent<Animator>().SetBool("animRunVrag", true);//запускаем анимацию бега

#region Отслеживание поворота

if (move > 0 /\*&& !Die\*/&& !facingRight)//если враг двигается направо и повернут налево

{

Flip();

}

else if (move < 0 /\*&& !Die\*/&& facingRight)//елси враг двигается налево и повернут направо

{

Flip();

}

#endregion

}

}

readonly string[] tagBadName = { "Bullet", "Player", "Untagged" };// Теги предметов, не требующие разворота

private void OnTriggerEnter2D(Collider2D other)//если в тригер входит другой collider

{

if (!tagBadName.Contains(other.gameObject.tag))//если тег этого объекта вынуждает повернуться

{

Flip();//выполняем разворот

}

}

void Flip()//разворот врага

{

facingRight = !facingRight;//флаг поворота

Vector2 theScale = transform.localScale;

theScale.x \*= -1;

transform.localScale = theScale;

move \*= -1;

}

/// <summary>

/// Перезарядка для 1 пушки

/// </summary>

/// <returns></returns>

IEnumerator WaitAtach1() // карутина

{

StartCoroutine(WaitAtach2());//запускаем ожидание атаки 2 пушки

yield return new WaitForSeconds(4f);//ждем 4 сек

atachFlag1 = true;//разрешаем атаку 1 пушки

}

/// <summary>

/// Перезарядка для 2 пушки

/// </summary>

/// <returns></returns>

IEnumerator WaitAtach2() // карутина

{

yield return new WaitForSeconds(2f);//ждем 2 сек

atachFlag2 = true;//разрешаем атаку 2 пушки

}

/// <summary>

/// Возвращает цвет через 100мс

/// </summary>

/// <returns></returns>

IEnumerator ColorTransformBack() // карутина

{

yield return new WaitForSeconds(0.1f);//ждем 0.1 сек

foreach (var child in GetComponentsInChildren<SpriteRenderer>())//перебираем все дочерние объекты

child.gameObject.GetComponent<SpriteRenderer>().material.color = new Color(1f, 1f, 1f);//и возвращаем им стандартный цвет

}

}

# InBullet

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class InBullet : MonoBehaviour

{

public int damage { get; set; }

void Start()

{

StartCoroutine(DestroyBullet());//запускаем отсчет времени жизни пули

}

void Update()

{

}

/// <summary>

/// Если пуля входит в тригер

/// </summary>

/// <param name="other"></param>

public void OnTriggerEnter2D(Collider2D other)

{

if (other.tag == "Vrag")//если это враг

{

other.gameObject.GetComponent<InVrag>().Damage(damage);//наносим дамаг врагу через фун-ию

Destroy(gameObject);//уничотжение пули

}

if (other.tag == "Player"&&other.isTrigger)//если это collider игрока

{

other.gameObject.GetComponent<InPlayer>().Damage(damage);//наносим дамаг игроку через фун-ию

Destroy(gameObject);//уничотжение пули

}

if (other.tag == "Turel" && !other.isTrigger)//если это collider турели

{

other.gameObject.GetComponentInParent<InTurel>().Damage(damage);//наносим дамаг турели через фун-ию

Destroy(gameObject);//уничотжение пули

}

if (!other.isTrigger)//если это другой collider

{

Destroy(gameObject);//уничотжение пули

}

}

/// <summary>

/// Корутина уничтожения пули, если она не во что не попала

/// </summary>

/// <returns></returns>

IEnumerator DestroyBullet() // дедлайн уничтожения "пули"

{

yield return new WaitForSeconds(5f);//засекаем 5 сек

Destroy(gameObject);//уничтожаем объект(пулю)

}

}

# inBox

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class inBox : MonoBehaviour

{

[Header("Префаб объекта, который должен появится")]

public GameObject obj;

void Start()

{

}

void Update()

{

}

bool d = true;

public void OnTriggerEnter2D(Collider2D other)

{

if ((other.tag == "Bullet"|| other.tag == "Ax")&& d)//если это пуля или топор

{

d = false;

if (obj != null)// если добавлен префаб

Instantiate(obj, transform.position, transform.rotation);//добавляем элемент

Die();

}

}

void Die()

{

gameObject.GetComponent<BoxCollider2D>().enabled = false;//отключаем видимость объекта

gameObject.GetComponent<SpriteRenderer>().enabled = false;//и колайдер

for (int i = 0; i < transform.childCount; i++)//обращаемся ко всем дочерним объектам

{

transform.GetChild(i).gameObject.SetActive(true);//активируем

transform.GetChild(i).GetComponent<Rigidbody2D>().velocity = new Vector2(Random.Range(-3f, 3f), 5f);//задаем равндомное движение

transform.GetChild(i).GetComponent<Rigidbody2D>().AddTorque(Random.Range(-50f, 50f));//и вращение

}

StartCoroutine(DieTime());//и удаляем эти объекты речерз некоторое время

}

int n=0;

IEnumerator DieTime() // карутина

{

yield return new WaitForSeconds(0.5f);//ждем

transform.GetChild(n++).gameObject.SetActive(false);//дезактивируем дочерние объекты

if(n==8)//если это последний элемент, удаляем весь объект(box)

Destroy(gameObject);

StartCoroutine(DieTime());

}

}

# PauseMenu

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class PauseMenu : MonoBehaviour

{

public GameObject pauseMenu;// объект самого меню pause

public GameObject RestartMenu;// объект самого меню restart

public GameObject healfImage;// полоска жизней

private bool pauseBool = false;

void Start()

{

Cursor.visible = false;

// задаем размер и расположение полоски жизней и меню паузы, пропорционально размеру экрана

pauseMenu.transform.localScale = new Vector2(gameObject.GetComponent<RectTransform>().rect.width / 1326f, gameObject.GetComponent<RectTransform>().rect.height / 694f);

pauseMenu.transform.position = new Vector2(gameObject.GetComponent<RectTransform>().rect.width / 2, gameObject.GetComponent<RectTransform>().rect.height / 2);

healfImage.transform.localScale = new Vector2(gameObject.GetComponent<RectTransform>().rect.width / 1326f, gameObject.GetComponent<RectTransform>().rect.height / 694f);

healfImage.transform.position= new Vector2((healfImage.GetComponent<RectTransform>().rect.width / 10f \* healfImage.transform.localScale.x) + (healfImage.GetComponent<RectTransform>().rect.width / 2f \* healfImage.transform.localScale.x), healfImage.GetComponent<RectTransform>().rect.height \*3 \* healfImage.transform.localScale.y);

}

void Update()

{

if (Input.GetKeyDown(KeyCode.Escape))//если нажат Esc

{

if (!pauseBool)//если игра не на паузе

{

OnPausePressed();// ставим на паузу

}

else//если игра на паузе

{

OffPausePressed();// снимаем с паузы

}

}

if (Input.GetKeyDown(KeyCode.R))//если нажат R идет рестарт уровня

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);//пере загружаем данную сцену

Time.timeScale = 1;

}

}

public void OnRestartPressed()//при смерти включение панели рестарта

{

Cursor.visible = true;

RestartMenu.SetActive(true);

Time.timeScale = 0.5f;// останавливаем время

}

public void Restart()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);//перезагружаем данную сцену

Time.timeScale = 1;// останавливаем время

}

private void OnPausePressed()//включение паузы

{

Cursor.visible = true;

pauseBool = true;// объявляем, что игра на паузе

pauseMenu.SetActive(true);//активируем меню паузы

Time.timeScale = 0;// останавливаем время

}

public void OffPausePressed()// выключение паузы

{

Cursor.visible = false;

pauseBool = false;// объявляем, что игра больше не на паузе

pauseMenu.SetActive(false);//скрываем меню паузы

Time.timeScale = 1;// востанавливаем ход времени

}

public void ExitMenuPressed()

{

SceneManager.LoadScene(0);//запускаем сцену под индексом 0 ( меню )

Time.timeScale = 1;// востанавливаем ход времени

}

public void GoNextLVL()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);

}

}

# Menu

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Menu : MonoBehaviour

{

[Header("Панель меню")]

public GameObject menuImage;

[Header("Панель настроек")]

public GameObject settingsImage;

private bool activeFullScreen = true;

void Start()

{

// задаем размер и распложение меню, пропорционально размеру экрана

menuImage.transform.localScale = new Vector2(gameObject.GetComponent<RectTransform>().rect.width / 1326f, gameObject.GetComponent<RectTransform>().rect.width / 1326f);

menuImage.transform.position = new Vector2(gameObject.GetComponent<RectTransform>().rect.width / 4, gameObject.GetComponent<RectTransform>().rect.height / 2);

}

void Update()

{

}

public void PlayPressed()

{

//SceneManager.LoadScene("Scene\_1");//запускаем сцену под именем 1

SceneManager.LoadScene(1);//запускаем сцену под индексом 1

}

public void ExitPressed()

{

Application.Quit();//выход из приложения

}

public void OpenSettingsPressed()// открытие настроек

{

menuImage.SetActive(false);//убираем основное меню

settingsImage.SetActive(true);// активируем панель настроек

}

public void CloseSettingsPressed()// закрытие настроек

{

menuImage.SetActive(true);//активируем основное меню

settingsImage.SetActive(false);// убираем панель настроек

}

public void ActivateFullScreen()// полноэкранный режим

{

activeFullScreen = !activeFullScreen;//меняем флаг полноэкранного режима

Screen.fullScreen = activeFullScreen;//в зависимости от флага, выбираем опцию экрана

}

}

# inDownPanel

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class inDownPanel : MonoBehaviour

{

[Header("Время до падения панели")]

public float waitDownTime;

void Start()

{

}

void Update()

{

}

public void OnTriggerEnter2D(Collider2D other)//если коллайдер соприкасется с объектом

{

if (other.tag == "Player")//если это игрок

{

StartCoroutine(DownTime());//запускаем таймер падения панели

}

if (other.tag == "DieZone")//если это дестрой зона

{

Destroy(gameObject);//уничтожаем панель

}

}

IEnumerator DownTime() // время до падения панели

{

yield return new WaitForSeconds(waitDownTime);//засекаем 5 сек

GetComponent<Rigidbody2D>().constraints = RigidbodyConstraints2D.None;//убираем все заморозки Rigidbody2D

GetComponent<Rigidbody2D>().gravityScale = 3f;

StartCoroutine(DestroyTime());//запускаем таймер самоуничтожения

}

IEnumerator DestroyTime() // время до уничтожения панели

{

yield return new WaitForSeconds(7f);//засекаем 7 сек

Destroy(gameObject);//самоуничтожаем объект

}

}

# inTurel

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class InTurel : MonoBehaviour

{

[Header("Вращ. часть турели")]

public Transform turTransf;

[Header("Позиция из которой выл. пуля")]

public Transform firePosition;

[Header("Префаб пули")]

public GameObject bullet;

[Header("Время перезарядки")]

public float timeAtach=4f;

[Header("Урон")]

public int damage;

[Header("Cкорость вращения")]

public float rotationSpeed = 1F;

[Header("Мертвая зона вращения (чтобы турель не дергалась при x=0)")]

public float deadZone = 0.1F;//Мертвая зона вращения (чтобы турель не дергалась при x=0)

[Header("Здоровье")]

public int healf;

[Header("префаб мертвой турели")]

public GameObject dieTurel;

private bool attack = false;

private float rotateDirection = 0;//направление вращения ( "0" - не вращать, "1" - вправо и "-1" - влево)

private Transform player;

void Start()

{

StartCoroutine(SearchPlayer());

player = GameObject.FindGameObjectWithTag("Player").transform;

}

private void OnTriggerEnter2D(Collider2D other)

{

if (other.tag == "Player")//если это collider игрока

{

attack = true;//разрешаем отслеживание

}

}

private void OnTriggerExit2D(Collider2D other)

{

if (other.tag == "Player")//если это collider игрока

{

attack = false;//запрещаем атаку

}

}

void LateUpdate()

{

if (attack)

{

if (turTransf.InverseTransformPoint(player.position).x > deadZone / 2)//если ГГ левее турели, с учетом поворота, вращаем против часовой стрелки

rotateDirection = -1F;

else if (turTransf.InverseTransformPoint(player.position).x < -deadZone / 2)//если ГГ правее турели, с учетом поворота, вращаем по часовой стрелки

rotateDirection = 1F;

else

{

if (turTransf.InverseTransformPoint(player.position).y < 0)

rotateDirection = 1F;

else

rotateDirection = 0;

}

turTransf.rotation \*= Quaternion.Euler(0, 0, rotationSpeed \* rotateDirection);//вращаем турель

}

}

IEnumerator SearchPlayer() // карутина

{

yield return new WaitForSeconds(timeAtach);//ждем 2 сек

if(attack)

{

RaycastHit2D hit = Physics2D.Raycast(firePosition.position, firePosition.up, 100f, 9);//кидаем рейкаст по напровлению взгляда

if (hit.collider != null && hit.collider.gameObject.tag == "Player")//если он попал в игрока

{

GameObject bulletCopy = Instantiate(bullet, firePosition.position, firePosition.rotation);//создаем копию префаба "пули" на месте обозначенном posBullet1

bulletCopy.GetComponent<Rigidbody2D>().velocity = firePosition.up \* 20f; //задаем "пуле скорость, с которой она будет двигаться"

bulletCopy.GetComponent<InBullet>().damage = damage;//Передаем пуле урон

}

}

StartCoroutine(SearchPlayer());

}

public void Damage(int \_damage)

{

healf -= \_damage;

if (healf <= 0)

{

attack = false;

StartCoroutine(Die());

}

}

IEnumerator Die() // карутина

{

yield return new WaitForEndOfFrame();//ждем конца кадра

turTransf.rotation \*= Quaternion.Euler(0, 0, 0.4f);//вращаем турель

if(turTransf.rotation.eulerAngles.z >= 179)

{

Instantiate(dieTurel, transform.position, transform.rotation);

Destroy(gameObject);

}

StartCoroutine(Die());

}

}

# inFon

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class FonScript : MonoBehaviour

{

public float damping = 1.5f;

public Vector2 offset = new Vector2(2f, 1f);

private Transform player;

private float playerStartPosY;

private float playerStartPosX;

//public bool faceLeft;

//private int lastX;

void Start()

{

offset = new Vector2(Mathf.Abs(offset.x), offset.y);

FindPlayer();

}

public void FindPlayer()

{

player = GameObject.FindGameObjectWithTag("Player").transform;

//lastX = Mathf.RoundToInt(player.position.x);

transform.position = new Vector2(player.position.x, player.position.y + offset.y);

playerStartPosY = player.position.y;

playerStartPosX = player.position.x;

}

void Update()

{

if (player)

{

//int currentX = Mathf.RoundToInt(player.position.x);

//if (currentX > lastX) faceLeft = false; else if (currentX < lastX) faceLeft = true;

//lastX = Mathf.RoundToInt(player.position.x);

Vector2 target;

target = new Vector2(player.position.x + offset.x + 0.08f \* (playerStartPosX - player.position.x), player.position.y + offset.y + 0.1f \* (playerStartPosY- player.position.y));

Vector2 currentPosition = Vector2.Lerp(transform.position, target, damping \* Time.deltaTime);

transform.position = currentPosition;

}

}

}