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

Противники

Листов: 14

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

Москва 2023

Цель работы: При помощи встроенных инструментов Unity реализовать искусственный интеллект противника.

Начать следует с рассмотрения различных компонентов текущей сцены Unity.

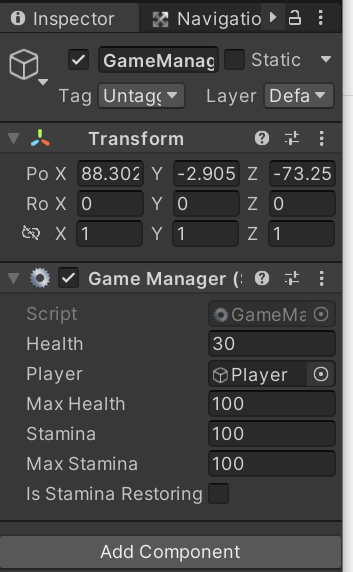


Рисунок 1 – Настройки компонента GameManager

В данной практической будет создан противник-слайм, который будет атаковать игрока. Ниже приведены его параметры.



Рисунок 2 – Параметры скрипта слайма

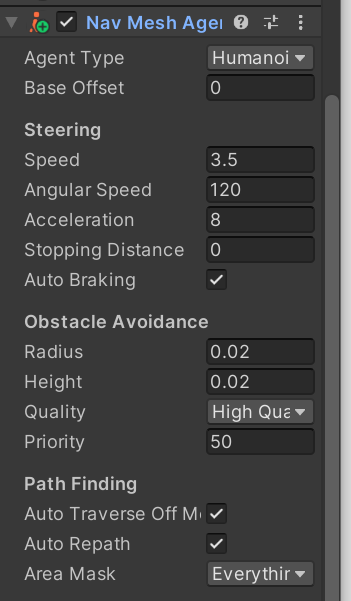


Рисунок 3 – Компонент навигации слайма

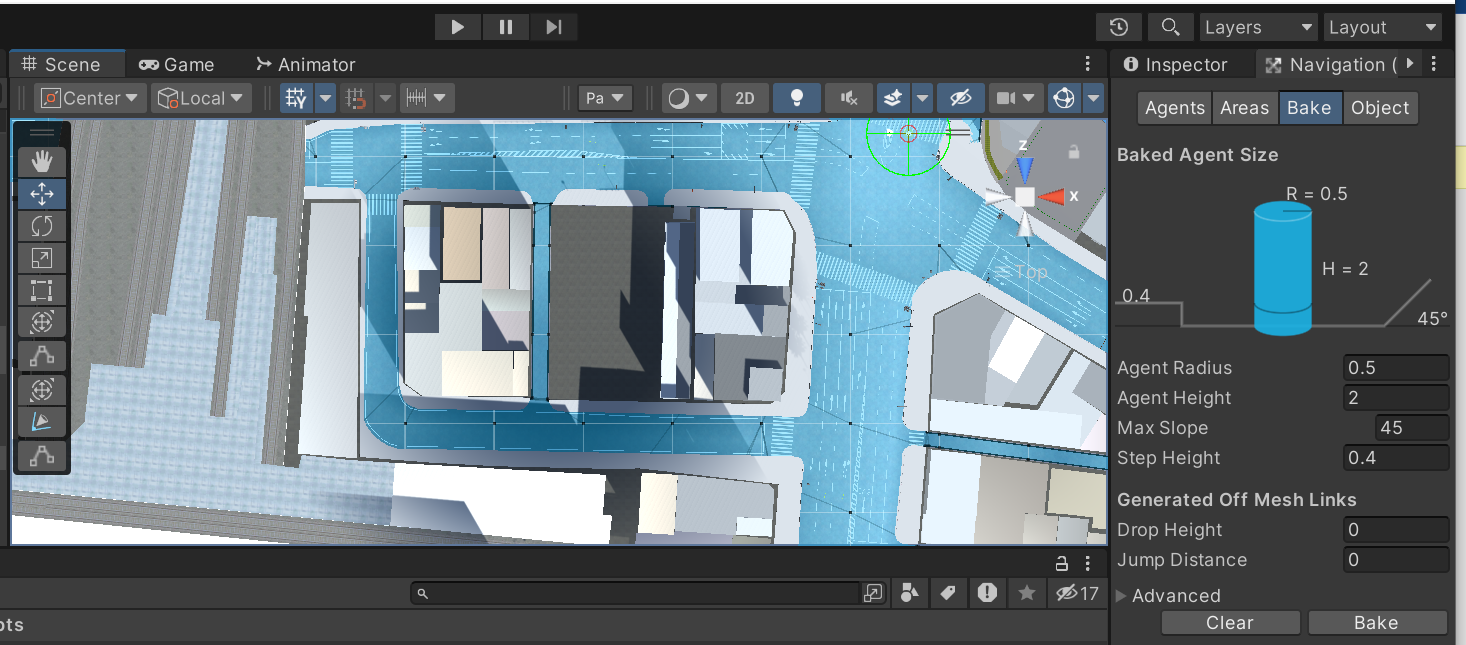


Рисунок 4 – Карта перемещения слайма (Синим выделено доступное пространство)

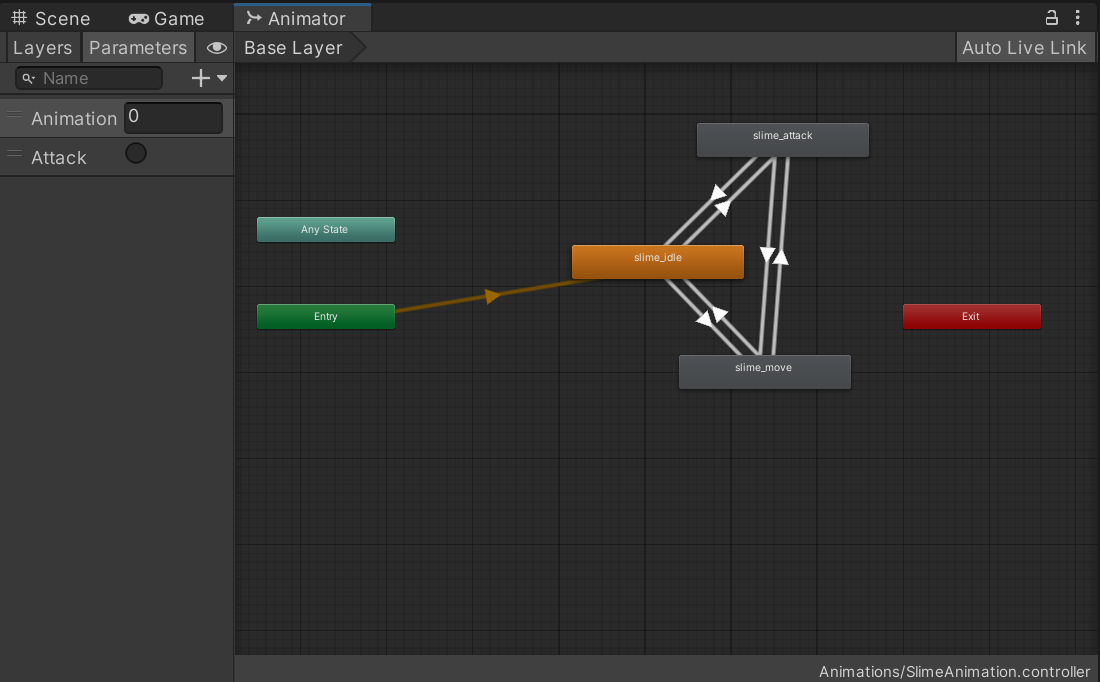


Рисунок 5 – Файл аниматора слайма

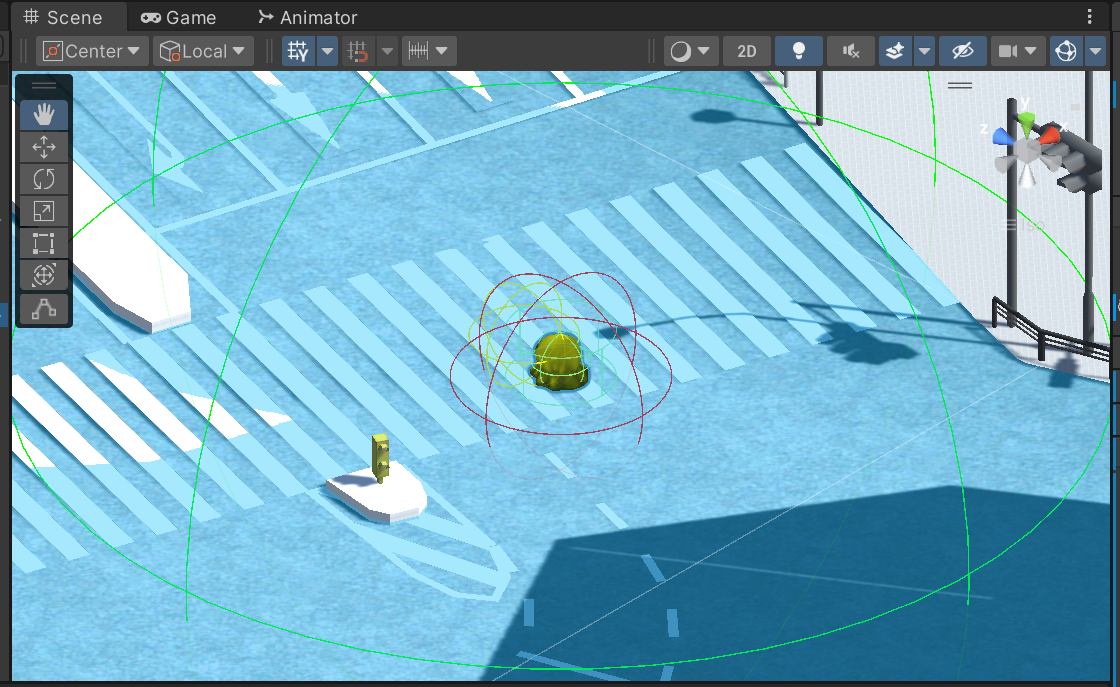


Рисунок 6 – Слайм на сцене

Результаты работы:

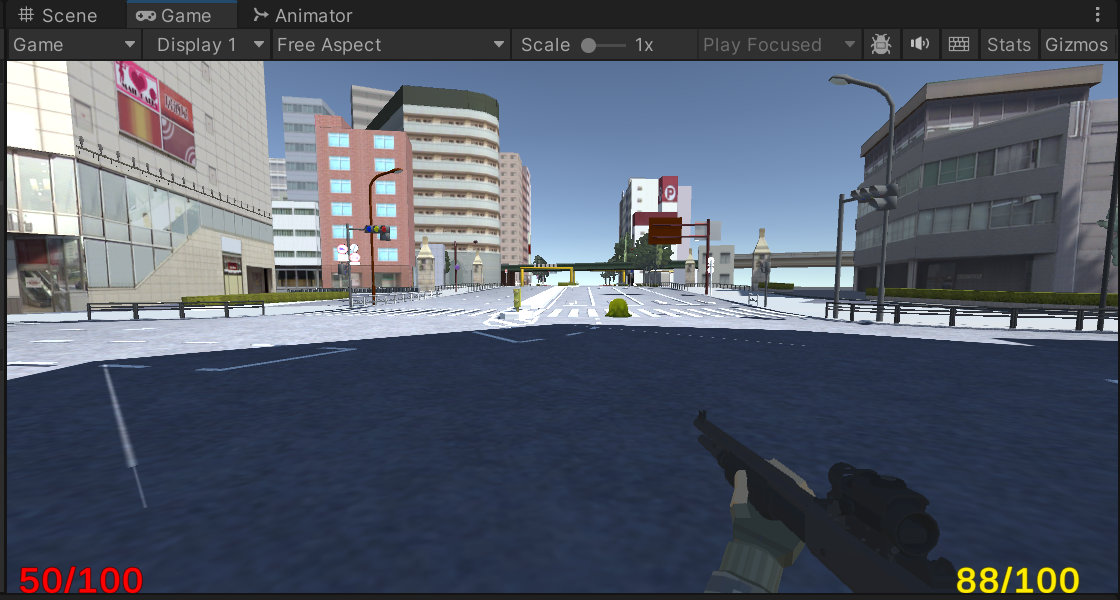


Рисунок 7 – Слайм до того как заметил игрока



Рисунок 8 – Слайм после того как заметил игрока

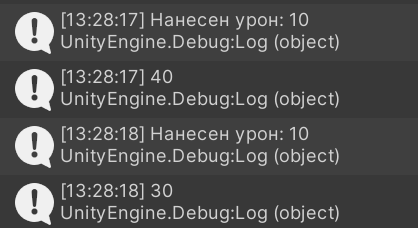


Рисунок 9 – Слайм наносит урон игроку



Рисунок 10 – Демонстрация нанесения урона

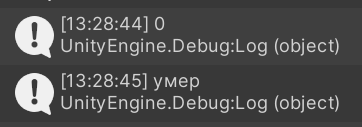


Рисунок 11 – Игрок умер

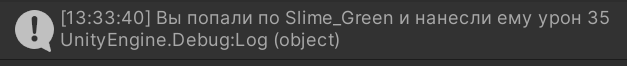


Рисунок 12 – Стрельба по слайму



Рисунок 13 – Исчезновение слайма после его смерти

Код GameManager:

using System.Collections;

using System.Collections.Generic;

using TMPro;

using UnityEngine;

using UnityEngine.UI;

public class GameManager : MonoBehaviour

{

public int Health;

public GameObject Healthobj;

public GameObject Player;

public static GameManager ManagerInstance;

private void Awake()

{

ManagerInstance = this;

}

public int MaxHealth = 100;

public float Stamina = 100;

public float maxStamina;

private TextMeshProUGUI Healthtxt;

public bool IsStaminaRestoring = false;

void Start()

{

Health = 50;

Healthtxt = Healthobj.GetComponent<TextMeshProUGUI>();

}

void FixedUpdate()

{

//healthBar.fillAmount = Health / MaxHealth;

//StaminaCheck();

}

private void StaminaCheck()

{

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

}

public IEnumerator StaminaRestore()

{

IsStaminaRestoring = true;

yield return new WaitForSeconds(3.0f);

Stamina = 100;

IsStaminaRestoring = false;

}

public void SpendStamina()

{

Stamina -= 1;

}

public void Healing(int HealthPointCount)

{

if (Health + HealthPointCount >= MaxHealth) Health = MaxHealth;

else Health += HealthPointCount;

Debug.Log("HP: " + Health);

}

public void damagePlayer(int Count)

{

if (Health > 0)

{

Health -= Count;

Debug.Log($"Вам нанесен урон в размере {Count}");

}

else

{

Health = 0;

Debug.Log("Вы умерли. Ваши останки обглодали падальщики и догрызли дикие собаки. Поделом.");

}

Healthtxt.text =

$"{Health}/{MaxHealth}";

}

}

Код EnemyController:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.AI;

public class EnemyController : MonoBehaviour

{

public float ViewDistance = 20f;

public float AttackDistance = 4f;

public GameObject AttackPoint;

public float AttackRange = 1f;

public LayerMask Player;

public int AttackDownSeconds = 1;

public int healt = 30;

public ParticleSystem DamageParticle;

public bool EnableAttack = true;

private Transform \_Target;

private NavMeshAgent \_Agent;

private Animator \_Animator;

private GameManager \_GameManager;

private float DistanceToPlayer;

private void Start()

{

\_Target = GameManager.ManagerInstance.Player.transform;

\_Agent = GetComponent<NavMeshAgent>();

\_Animator = GetComponent<Animator>();

\_GameManager = GameManager.FindAnyObjectByType<GameManager>();

}

private void FixedUpdate()

{

SetAnimation();

DistanceToPlayer = Vector3.Distance(

\_Target.position,

transform.position);

if (DistanceToPlayer <= ViewDistance)

{

\_Agent.SetDestination(\_Target.position);

transform.LookAt(\_Target.position);

if (DistanceToPlayer <= AttackDistance && EnableAttack)

{

StartCoroutine(AttackCountDown());

}

}

if (healt <= 0) Death();

}

private void Attack()

{

Collider[] HitedColliders = Physics.OverlapSphere(AttackPoint.transform.position,

AttackRange, Player);

EnableAttack = true;

foreach (Collider HitedCollider in HitedColliders)

{

\_GameManager.damagePlayer(10);

Debug.Log(\_GameManager.Health);

}

}

private IEnumerator AttackCountDown()

{

EnableAttack = false;

yield return new WaitForSeconds(1f);

Attack();

}

public void DealDamage(int Count)

{

healt -= Count;

DamageParticle.Play();

}

private void Death()

{

DamageParticle.transform.parent = null;

DamageParticle.Play();

Destroy(gameObject);

}

private void OnDrawGizmosSelected()

{

Gizmos.color = Color.yellow;

Gizmos.DrawWireSphere(AttackPoint.transform.position,

AttackRange);

Gizmos.color = Color.green;

Gizmos.DrawWireSphere(transform.position, ViewDistance);

Gizmos.color = Color.red;

Gizmos.DrawWireSphere(transform.position, AttackDistance);

}

private void SetAnimation()

{

if (DistanceToPlayer <= AttackDistance)

\_Animator.SetTrigger("Attack");

else

{

if (DistanceToPlayer <= ViewDistance) \_Animator.SetInteger("Animation", 1);

else \_Animator.SetInteger("Animation", 0);

}

}

}

Вывод: При помощи встроенных инструментов Unity реализовали искусственный интеллект противника.