МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина “ПиК ЯП”

Отчет по ДЗ

“Разработка игры на движке Unity с использованием языка C#”

Выполнили:

Куртинец Роман ИУ5-31Б
Рахманов Виталий ИУ5-31Б
Сербенюк Полина ИУ5-31Б

Преподаватель:
Гапанюк Ю.Е.

Москва 2025

Описание проекта

Целью данного проекта было изучение особенностей движка Unity и создание прототипа survivor-игры с использованием языка программирования C#. Игра содержит анимированные спрайты персонажей и оружия, их коллизию с объектами на поле, появление врагов за пределами видимости, нанесение им урона по ограниченной области, а также систему накапливания очков.

Рис. 1 Игровое поле

Также для удобства было создано несколько префабов, которые находятся на сцене: оружее, скелет, элементы декорации и кружки опыта.

Рис. 2 Префабы проекта

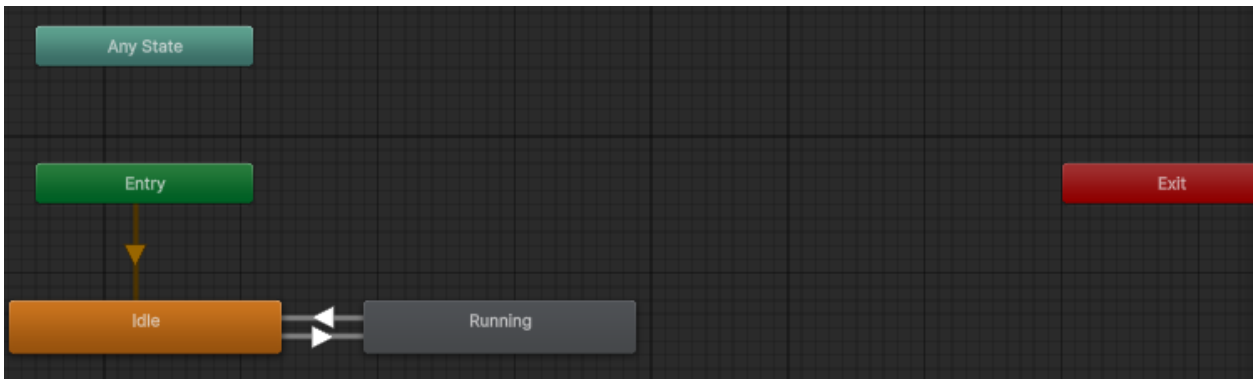

Рис.3 Контроллер анимаций персонажа (Аналогично для врагов)

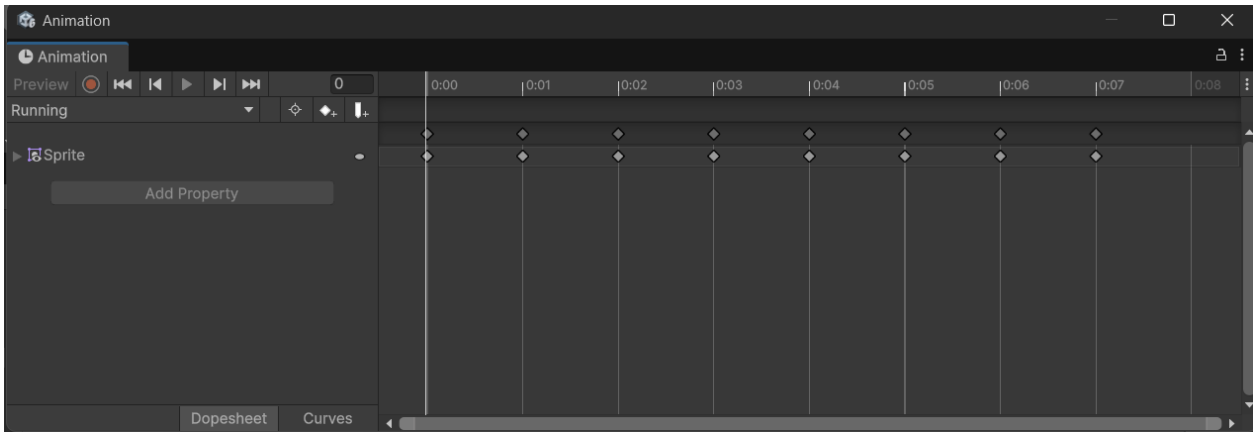

Рис.4 Аниматор персонажа (Аналогично для врагов)

При разработке также использовалась технология NavMesh.

NavMesh (навигационная сетка) — это структура данных и технология в разработке игр и робототехнике, которая позволяет персонажам (агентам) ИИ находить путь в 2D/3D-сценах, автоматически обходя препятствия, такие как стены, здания или другие объекты, и двигаясь по доступным поверхностям, например, полу, дорожкам, создавая "карту" проходимых областей для быстрой навигации.

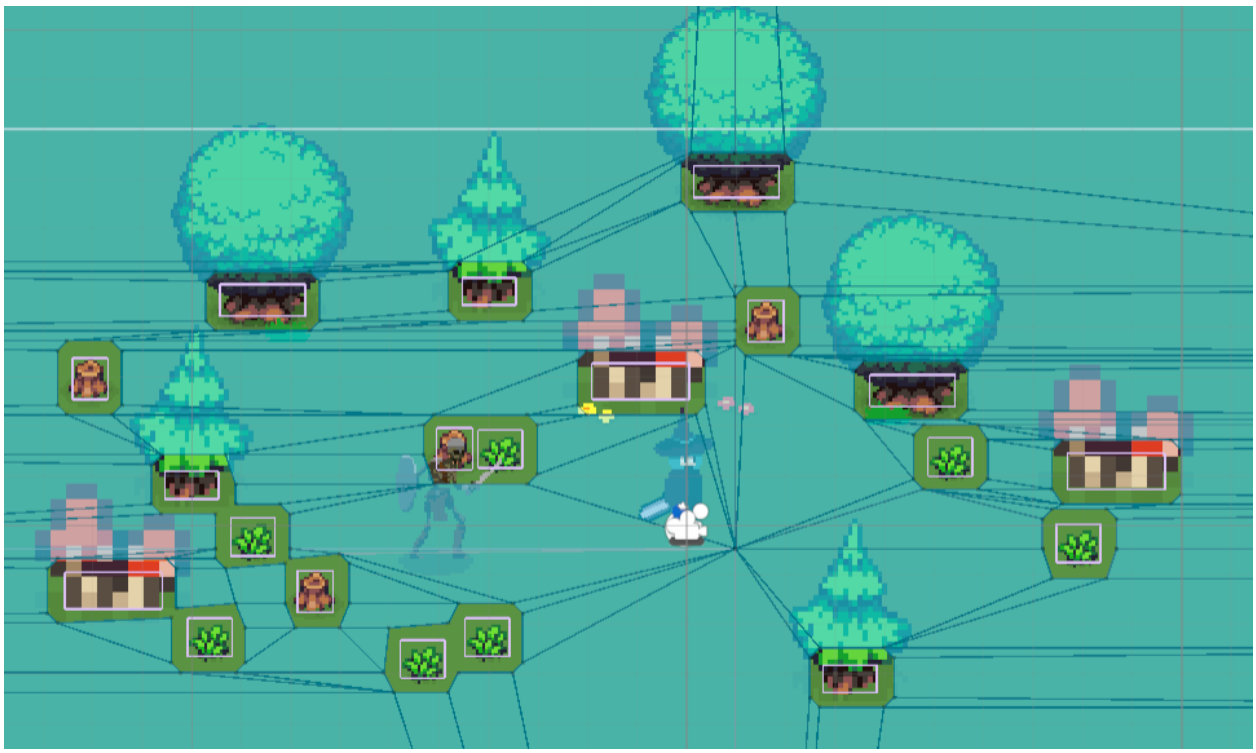

Рис. 4 Связь объектов в NavMesh

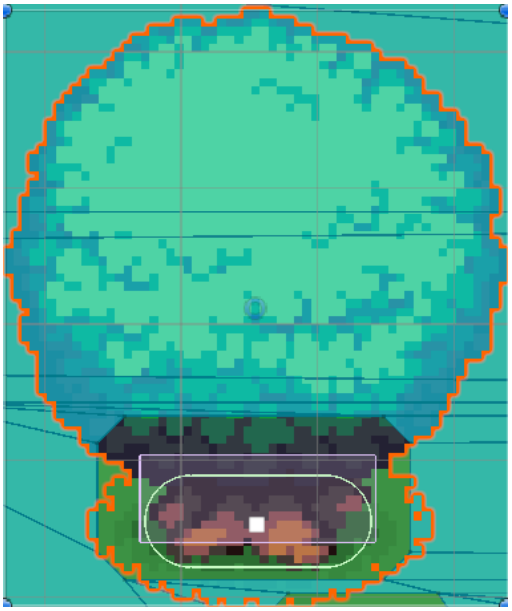

Рис. 5 Дерево имеет коллизию только в нижней части, чтобы персонаж не сталкивался с листвой и мог обойти препятствие

Рис. 6 Область, в которой регистрируется попадание по врагу

Рис. 7 Спрайты использованных декораций

Листинг Кода

1) Фрагменты кода, отвечающие за управление перемещением, анимацией и атакой 2D-персонажа игрока, включая обработку ввода, физику движения и визуальное отображение.

Player.cs:

```
using System.Collections;
using System.Collections.Generic;
using Unity.Mathematics;
using UnityEngine;

[SelectionBase] // для того, чтобы при перетаскивании в движке перса,
перетаскивался не только спрайт

public class Player : MonoBehaviour
{
```

```
    [SerializeField] private float playerSpeed = 5f;
    // благодаря [SerializeField] можно менять переменную в самом движке, а не
коде
    private Vector2 supGrid;
    Vector2 Grid;


    private Rigidbody2D rb; // эта штука для перемещения объектов (аналог
БЕГАЮЩЕГО крестоносца)

    private float minSpeed = 0.0f;
    private bool isRunning = false;

    public static Player Instance { get; private set; }
    private void Awake() // запускается до Start() 1 раз, ею мы инициализируем
компоненты
    {
        Instance = this;
        rb = GetComponent<Rigidbody2D>(); // Rigidbody2D в движке отвечает за
физику
        // здесь мы её пробуждаем
    }

    void Start()
    {
        GameInput.Instance.OnPlayerAttack += Player_OnPlayerAttack;
    }

    private void Player_OnPlayerAttack(object sender, System.EventArgs e)
    {
        ActiveWeapon.Instance.GetActiveWeapon().Attack(); // вызываем активное
оружее и выполняем атаку
    }

    private void Update() {
        Grid = GameInput.Instance.GetMoveVector(); // сетка движения персонажа
    }

    private void FixedUpdate() // эта функция запускается каждый кадр в игре
    {
        HandleMovement();
    }

    private void HandleMovement()
    {
        rb.MovePosition(rb.position + Grid * (playerSpeed *
Time.fixedDeltaTime));
        // Time.fixedDeltaTime - можно воспринимать как константу (она равна
0.02)
        // умножаем это всё для корректировки скорости
        supGrid = Grid;
```

```
        if (Mathf.Abs(Grid.x) > minSpeed || Mathf.Abs(Grid.y) > minSpeed)
        {
            isRunning = true;
        }
        else
        {
            isRunning = false;
        }
    }

    public Vector2 GetGrid() => supGrid;

    public bool IsRunning() => isRunning;
}
```

PlayerVisual.cs:

```
using NUnit.Framework.Internal;
using UnityEngine;

public class PlayerVisual : MonoBehaviour
{
    private Animator animator; // Получили доступ к аниматору
    private SpriteRenderer spriteRenderer;
    private const string IS_RUNNING = "IsRunning"; //с помощью неё будем
обновлять аниматор

    private void Awake()
    {
        animator = GetComponent<Animator>();
        spriteRenderer = GetComponent<SpriteRenderer>();
    }

    private void Update()
    {
        animator.SetBool(IS_RUNNING, Player.Instance.IsRunning()); //SetBool
обновляет значение аниматора, bool потому что у на isRunning тоже bool
        PlayerViewDirection();
    }

    private void PlayerViewDirection()
    {
        if (Player.Instance.GetGrid().x < 0) // чтобы перс смотрел в сторону
движения
        {
            spriteRenderer.flipX = true;
        }
        else if (Player.Instance.GetGrid().x > 0)
        {
            spriteRenderer.flipX = false;
```

```
        }
    }
}
```

2) Фрагменты кода, отвечающие за поведение, здоровье и систематическое появление врагов (скелетов) в игре, включая их преследование игрока, получение урона, выпадение опыта при смерти и появление за пределами видимости камеры.

EnemyAI.cs:

```
using System;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;
using GameUtils;

public class EnemyAI : MonoBehaviour
{
    [SerializeField] private float enemySpeed = 2.5f;

    private NavMeshAgent navMeshAgent; //navMesh - это система навигации для NPC

    private Transform player; // для будущей позиции игрока

    private void Awake()
    {
        navMeshAgent = GetComponent<NavMeshAgent>();
        navMeshAgent.updateRotation = false; // поворот спрайта
        navMeshAgent.updateUpAxis = false; // отключаем поворот для Z
    }

    private void Start()
    {
        navMeshAgent.speed = enemySpeed; // скорость скелетов
        if (Player.Instance != null)
        {
            player = Player.Instance.transform; // подтягиваем позицию игрока
        }
    }

    private void Update() // тут мы проверяем, в каком состоянии объект находится
    {
        navMeshAgent.SetDestination(player.position);
        ChangeFacingDirection();
    }

    private void ChangeFacingDirection()
    {
        if (navMeshAgent.velocity.x < 0) // если враг идёт влево
```

```
        {
            transform.rotation = Quaternion.Euler(0, -180, 0); // запись,
поворачивающая объект
        }
        else if (navMeshAgent.velocity.x > 0) // если враг идёт вправо
        {
            transform.rotation = Quaternion.Euler(0, 0, 0);
        }
    }

}
```

EnemyEntity.cs:

```
using UnityEngine;
using System;

public class EnemyEntity : MonoBehaviour
{
    [SerializeField] private int _maxHealth;
    [SerializeField] private GameObject experiencePickupPrefab; // префаб кружка
xp

    private int _currHealth;
    private bool isDead = false; // чтобы не наносить урон мёртвому

    private void Start()
    {
        _currHealth = _maxHealth;
    }

    public void TakeDamage(int damage) // враг получает урон
    {
        if (isDead) return; // если враг мёртв, то ничего не делаем
        _currHealth -= damage;

        if (_currHealth <= 0)
        {
            isDead = true;
            DetectDeath();
        }
    }

    private void DetectDeath()
    {
        Vector3 deathPosition = transform.position;

        if (experiencePickupPrefab != null)
        {
            Instantiate(experiencePickupPrefab, deathPosition,
Quaternion.identity); // спавн кружка опыта
```

```
        }

        Destroy(gameObject);
    }
}
```

EnemySpawner.cs:

```
using UnityEngine;
using System.Collections;

public class EnemySpawner : MonoBehaviour
{
    [SerializeField] private GameObject enemyPrefab; // префаб скелета
    [SerializeField] private float cameraSafeRadius = 14f; // радиус, чтобы враги
спавнились за камерой
    [SerializeField] private float spawnInterval = 2.5f;
    [SerializeField] private Vector2 mapBounds = new Vector2(50, 50); // размер
карты

    private Camera mainCamera;

    private void Start()
    {
        mainCamera = Camera.main;
        StartCoroutine(SpawnEnemiesRoutine());
    }

    private IEnumerator SpawnEnemiesRoutine()// данная штука умеет останавливать
цикл на
    // время (надо для спавна каждые 2.5 секунды)
    {
        while (true)
        {
            SpawnEnemy();
            yield return new WaitForSeconds(spawnInterval); // эта штука как раз
заставляет цикл ждать до следующего обновления
        }
    }

    private void SpawnEnemy()
    {
        Vector2 spawnPos;

        do
        {
            spawnPos = new Vector2(Random.Range(-mapBounds.x / 2, mapBounds.x /
2),
            Random.Range(-mapBounds.y / 2, mapBounds.y / 2));
        } while (IsPointInCameraView(spawnPos)); // повторяем рандом, пока враг
не будет спавниться за пределами камеры
```

```
        Instantiate(enemyPrefab, spawnPos, Quaternion.identity /*смотрит в
стандартном направлении*/);
        // создаём врага
    }

    private bool IsPointInCameraView(Vector2 point) // для проверки на то, что
точка находится в камере
    {
        Vector2 cameraPos = mainCamera.transform.position;

        float distance = Vector2.Distance(point, cameraPos); // расстояние от
точки до камеры

        return distance < cameraSafeRadius; // если расстояние меньше безопасного
радиуса, точка в зоне камеры
    }
}
```

3) Фрагменты кода, отвечающие за сбор и систему накопления игрового опыта, где подбираемые объекты увеличивают текущий опыт, а при достижении порога – повышают уровень персонажа.

ExperiencePickUp.cs:

```
using UnityEngine;

public class ExperiencePickUp : MonoBehaviour
{
    [SerializeField] private int xpAmount = 10;

    private bool isPickedUp = false;

    private void OnTriggerEnter2D(Collider2D collision) // срабатывания триггера
при подходе к шарику xp
    {
        if (isPickedUp) return; // проверяем, что мы ещё не подобрали кружочек

        if (collision.CompareTag("Player")|| collision.GetComponent<Player>() !=
null)
        {
            if (ExperienceSystem.Instance != null)
            {
                ExperienceSystem.Instance.AddXp(xpAmount);
            }
            isPickedUp = true; // кружочек подобран
            Destroy(gameObject);
        }
    }
```

```
}
```

ExperienceSystem.cs:

```
using UnityEngine;
using System;

public class ExperienceSystem : MonoBehaviour
{
    public static ExperienceSystem Instance { get; private set; }

    [SerializeField] private int xpPerPickup = 10; // xp с одного моба
    [SerializeField] private int xpForLevelUp = 100; // xp на левел-ап

    private int currXp = 0;
    private int currLevel = 1;

    public event Action<float> OnXpChanged; // изменение XP
    public event Action<int> OnLevelUp;

    private void Awake()
    {
        Instance = this;
    }

    public void AddXp(int xp)
    {
        currXp += xp;
        if (currXp >= xpForLevelUp)
        {
            currXp -= xpForLevelUp; // сбрасываем xp
            ++currLevel;
            if (OnLevelUp != null) OnLevelUp.Invoke(currLevel);
        }

        float progress = (float)currXp / xpForLevelUp; // считаем, сколько на
данный момент опыта
        if (OnXpChanged != null) OnXpChanged.Invoke(progress);
    }

}
```

4) Фрагменты кода, отвечающие за работу меча как активного оружия игрока, включая его анимацию взмаха, отслеживание направления атаки, нанесение урона врагам через коллайдер и синхронизацию визуальной части с игровой логикой.

ActiveWeapon.cs:

```
using UnityEngine;
```

```
using NUnit.Framework.Internal;

public class ActiveWeapon : MonoBehaviour
{
    public static ActiveWeapon Instance {get; private set;} // для просмотра, какое оружие сейчас выбрано

    [SerializeField] private Sword sword;

    private void Awake()
    {
        Instance = this;
    }

    private void Update()
    {
        FollowSidePosition();
    }

    public Sword GetActiveWeapon() => sword;

    private void FollowSidePosition() // поворот оружия в нужную сторону
    {
        if (Player.Instance.GetGrid().x < 0)
        {
            transform.rotation = Quaternion.Euler(0, 180, 0); // Quaternion - штука для поворота обьектов
        }
        else if(Player.Instance.GetGrid().x > 0)
        {
            transform.rotation = Quaternion.Euler(0, 0, 0);
        }
    }
}
```

Sword.cs:

```
using System;
using UnityEngine;

public class Sword : MonoBehaviour
{
    [SerializeField] private int _damageAmount = 5;

    public event EventHandler OnSwordSwing; // событие для того, чтобы система поняла, что пора взмахнуть мечом

    private PolygonCollider2D _polygonCollider2D;

    private void Awake()
    {
```

```
        _polygonCollider2D = GetComponent<PolygonCollider2D>();
    }

    private void Start()
    {
        AttackColliderTurnOff();
    }

    public void Attack()
    {
        AttackColliderTurnOffOn();

        if (OnSwordSwing != null) OnSwordSwing.Invoke(this, EventArgs.Empty);
    }

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision == null || collision.transform == null) return; // проверка
на уничтоженный объект
        if (collision.transform.TryGetComponent(out EnemyEntity enemyEntity))
enemyEntity.TakeDamage(_damageAmount);
        // проверка на столкновение с каким то другим коллайдером
    }

    public void AttackColliderTurnOff() // выключаем коллайдер, когда взмаха
мечом нет
    {
        _polygonCollider2D.enabled = false;
    }

    private void AttackColliderTurnOn() // включаем коллайдер, когда взмаха мечом
есть
    {
        _polygonCollider2D.enabled = true;
    }

    private void AttackColliderTurnOffOn() // это для того, чтобы человек при
нажатии 10^10 кликов
    //  в секунду всё равно наносил урон
    {
        AttackColliderTurnOff();
        AttackColliderTurnOn();
    }
}
```

SwordVisual.cs:

```
using UnityEngine;

public class SwordVisual : MonoBehaviour
{
```

```
    [SerializeField] private Sword sword;

    private Animator animator;
    private const string ATTACK = "Attack";

    private void Awake()
    {
        animator = GetComponent<Animator>();
    }

    private void Start()
    {
        sword.OnSwordSwing += Sword_OnSwordSwing;
    }

    private void Sword_OnSwordSwing(object sender, System.EventArgs e)
    {
        animator.SetTrigger(ATTACK);
    }

    public void TriggerEndAttackAnimation()
    {
        sword.AttackColliderTurnOff();
    }
}
```

5) Фрагменты кода, отвечающие за плавное следование камеры за персонажем.

CameraFollow.cs:

```
using UnityEngine;

public class CameraFollow : MonoBehaviour
{
    [SerializeField] private Transform player; // создаём переменную для будущего хранения ссылки на местоположение игрока
    [SerializeField] private Vector3 offset = new Vector3(0, 0, -10); // координаты камеры (она ровно над игроком)
    [SerializeField] private float smoothSpeed = 0.125f; // плавность полёта камеры

    private void LateUpdate() // LateUpdate выполняется после всех Update, т.е. после того, как персонаж двинулся
    {
        Vector3 newPos = player.position + offset;
        Vector3 smoothPos = Vector3.Lerp(transform.position, newPos, smoothSpeed); // новая позиция камеры (откуда,
        // куда, насколько плавно)
```

```
        transform.position = smoothPos; // перемещение камеры
    }
}
```

6) Фрагменты кода, отвечающие за обработку всего пользовательского ввода в игре через систему Input System, включая управление перемещением персонажа и активацию атаки по нажатию кнопки.

GameInput.cs:

```
using System;
using UnityEngine;
using UnityEngine.InputSystem;

public class GameInput : MonoBehaviour
{

    public static GameInput Instance { get; private set; }

    private PlayerInputActions playerInputActions;

    public event EventHandler OnPlayerAttack; // создаём событие

    private void Awake()
    {
        Instance = this;
        playerInputActions = new PlayerInputActions();
        playerInputActions.Enable();

        playerInputActions.Combat.Attack.started += PlayerAttack_started; //
атака персонажа
    }

    private void PlayerAttack_started(InputAction.CallbackContext obj)
    {
        if (OnPlayerAttack != null) OnPlayerAttack.Invoke(this, EventArgs.Empty);
    }

    public Vector2 GetMoveVector()
    {
        Vector2 Grid = playerInputActions.Player.Move.ReadValue<Vector2>();
        return Grid;
    }
}
```

7) Фрагменты кода, отвечающие за генерацию случайного нормализованного направления в двухмерном пространстве для использования в игровой логике.

Utils.cs:

```
using UnityEngine;

namespace GameUtils{
public static class Utils
{
    public static Vector3 GetRandomDir()
        {
            return new Vector3(Random.Range(-1f, 1f), Random.Range(-1f,
1f)).normalized;
        }
}
}
```