Министерство науки и высшего образования Российской Федерации
Муромский институт (филиал)
Федерального государственного бюджетного образовательного учреждения
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Факультет ИТР

Кафедра ПИн

ЛАБОРАТОРНАЯ РАБОТА №3

По Компьютерной графике

Руководитель

Привезенцев Д.Г.
(фамилия, инициалы)

(подпись) (дата)

Студент ПИн - 121
(группа)

Ермилов М.В.
(фамилия, инициалы)

(подпись) (дата)

Муром 2023

Лабораторная работа №3

Тема: Создание игры Tower Defense

Скрипты проекта

TowerPlacer.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Tilemaps;


public class Tower
{
    public Vector3Int Vector { get; set; }
    public Tile Tile { get; set; }
    public int Level { get; set; }
    public float CurrentAngle { get; set; }
    public float TimeTill { get; set; }
    public Vector3 WorldPosition { get; set; }
    public float TimeToReload { get; set; }
    public float ReloadTime { get; set; }
    public GameObject Bullet { get; set; }
    public int Damage { get; set; }

}

public class TowerPlacer : MonoBehaviour
{
    public Tilemap mapPlatforms;
    public Tilemap mapTowers;
    public Tile Tower;
    public Tile TileBase;
    private Camera mainCamera;
    public GameObject Bullet;
    List<Tower> towers;
    void Start()
    {
        mainCamera = Camera.main;
        towers = new List<Tower>();
    }
    float timeTill = 0.0f;
    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            Vector3 click = mainCamera.ScreenToWorldPoint(Input.mousePosition);
            Vector3Int cellClick = mapPlatforms.WorldToCell(click);
            if (mapPlatforms.GetTile(cellClick) == TileBase)
            {
                mapTowers.SetTile(cellClick, Tower);

                towers.Add(new Tower)
                {
                    Vector = cellClick,
                    Tile = Tower,
                    Level = 1,
                    WorldPosition = click,
                    TimeTill = 0.0f,
                    Damage = 25,
                    Bullet = Bullet,
                    ReloadTime = 1f
                });
            }
        }
        if (timeTill > 2)
        {
            foreach (var tower in towers)
            {
                tower.CurrentAngle += 45;
                Matrix4x4 matrix = Matrix4x4.TRS(Vector3.zero, Quaternion.Euler(0f, 0f,
tower.CurrentAngle), Vector3.one);
                mapTowers.SetTransformMatrix(tower.Vector, matrix);
            }
            timeTill = 0.0f;
        }
        timeTill += Time.deltaTime;
        foreach (var tower in towers)
        {
            if (tower.TimeToReload == 0)
            {
                Collider2D[] colliders = Physics2D.OverlapCircleAll(new
Vector2(tower.WorldPosition.x, tower.WorldPosition.y), 2);
                Debug.Log(colliders.Length);
                foreach (Collider2D collider in colliders)
                {
                    Shoot(collider, tower);
                    tower.TimeToReload = tower.ReloadTime;
                    break;
                }
            }
            else
                tower.TimeToReload = Mathf.Max(0, tower.TimeToReload - Time.deltaTime);
        }
    }
    void Shoot(Collider2D target, Tower tower)
    {
        Vector3 startPosition = tower.WorldPosition;
        Vector3 targetPosition = target.transform.position;
        startPosition.z = tower.Bullet.transform.position.z;
        targetPosition.z = tower.Bullet.transform.position.z;
        GameObject newBullet = (GameObject)Instantiate(tower.Bullet);
        newBullet.transform.position = startPosition;
        BulletBehavior bulletComp = newBullet.GetComponent<BulletBehavior>();
        bulletComp.target = target.gameObject;
        bulletComp.startPosition = startPosition;
        bulletComp.targetPosition = targetPosition;
        bulletComp.damage = tower.Damage;
    }
}
```

SpawnEnemy.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class SpawnEnemy : MonoBehaviour
{
    [System.Serializable]
    public class Wave
    {
        public GameObject enemyPrefab;
        public float spawnInterval = 2;
        public int maxEnemies = 20;
    }
    public GameObject[] waypoints;
    private int wave;
    public Wave[] waves;
    public int timeBetweenWaves = 5;
    private float lastSpawnTime;
    private int enemiesSpawned = 0;
    private void Start()
    {
        lastSpawnTime = Time.time;
        wave = 0;
    }
    void Update()
    {
        if (wave < waves.Length)
        {

            float timeInterval = Time.time - lastSpawnTime;
            float spawnInterval = waves[wave].spawnInterval;
            if (((enemiesSpawned == 0 && timeInterval > timeBetweenWaves) ||
            timeInterval > spawnInterval) &&
            enemiesSpawned < waves[wave].maxEnemies)
            {
                lastSpawnTime = Time.time;
                GameObject newEnemy = Instantiate(waves[wave].enemyPrefab) as
GameObject;
                newEnemy.GetComponent<MoveEnemy>().waypoints = waypoints;
                enemiesSpawned++;
            }
            if (enemiesSpawned == waves[wave].maxEnemies &&
            GameObject.FindGameObjectWithTag("Enemy") == null)
            {
                wave++;
                enemiesSpawned = 0;
                lastSpawnTime = Time.time;
            }
        }
    }
}
```

MoveEnemy.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class MoveEnemy : MonoBehaviour
{
    [HideInInspector]
    public GameObject[] waypoints;
    private int currentWaypoint = 0;
    private float lastWaypointSwitchTime;
    public float speed = 1.0f;
    public Transform SpriteTransform;
    void Start()
    {
        lastWaypointSwitchTime = Time.time;
    }
    void Update()
    {
        Vector3 startPosition = waypoints[currentWaypoint].transform.position;
        Vector3 endPosition = waypoints[currentWaypoint + 1].transform.position;
        float pathLength = Vector3.Distance(startPosition, endPosition);
        float totalTimeForPath = pathLength / speed;
        float currentTimeOnPath = Time.time - lastWaypointSwitchTime;
        gameObject.transform.position = Vector2.Lerp(startPosition, endPosition,
        currentTimeOnPath / totalTimeForPath);
        if (gameObject.transform.position.Equals(endPosition))
        {
            if (currentWaypoint < waypoints.Length - 2)
            {
                currentWaypoint++;
                lastWaypointSwitchTime = Time.time;
                RotateIntoMoveDirection();
            }
            else
            {
                Destroy(gameObject);
            }
        }
    }
    void RotateIntoMoveDirection()
    {
        Vector3 newStartPosition = waypoints[currentWaypoint].transform.position;
        Vector3 newEndPosition = waypoints[currentWaypoint + 1].transform.position;
        Vector3 newDirection = (newEndPosition - newStartPosition);
        float x = newDirection.x;
        float y = newDirection.y;
        float rotationAngle = Mathf.Atan2(y, x) * 180 / Mathf.PI;
        SpriteTransform.transform.rotation = Quaternion.AngleAxis(rotationAngle,
        Vector3.forward);
    }
}
```

HealthBar.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class HealthBar : MonoBehaviour
{
    public float maxHealth = 100;
    public float currentHealth = 100;
    private float originalScale;
    public void Start()
    {
        originalScale = gameObject.transform.localScale.x;
        currentHealth = 100;
    }
    private void Update()
    {
        Vector3 tmpScale = gameObject.transform.localScale;
        tmpScale.x = currentHealth / maxHealth * originalScale;
        gameObject.transform.localScale = tmpScale;
    }
}
```

GameManagerBehavoir.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class GameManagerBehavior : MonoBehaviour
{
    public int Wave { get; set; }
    void Start()
    {
        Wave = 0;
    }
    void Update()
    {

    }
}
```

EnemyCount.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class EnemyCount : MonoBehaviour
{
    Text text;
    public static int enemys;
    void Start()
    {
        text = GetComponent<Text>();
    }
    void Update()
    {
        text.text = enemys.ToString();
    }
}
```

BulletBehavior.cs

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

public class BulletBehavior : MonoBehaviour
{
    public float speed;
    public int damage;
    public GameObject target;
    public Vector3 startPosition;
    public Vector3 targetPosition;

    private float distance;
    private float startTime;

    private Transform SpriteTransform;


    void Start()
    {
        startTime = Time.time;
        distance = Vector2.Distance(startPosition, targetPosition);
        SpriteTransform = gameObject.GetComponentInChildren<Transform>();
    }
    void Update()
    {
        if (target == null)
        {
            Destroy(gameObject);
            return;
        }
        targetPosition = target.transform.position;
        RotateIntoMoveDirection();
        float timeInterval = Time.time - startTime;
        gameObject.transform.position = Vector3.Lerp(startPosition, targetPosition,
       timeInterval * speed / distance);
        if (gameObject.transform.position.Equals(targetPosition))
        {
            if (target != null)
            {
                HealthBar healthBar = target.GetComponentInChildren<HealthBar>();
                healthBar.currentHealth -= Mathf.Max(damage, 0);
                if (healthBar.currentHealth <= 0)
                {
                    Destroy(target);
                }
            }
            Destroy(gameObject);
        }
    }
    void RotateIntoMoveDirection()
    {
        Vector3 newStartPosition = transform.position;
        Vector3 newEndPosition = targetPosition;
        Vector3 newDirection = (newEndPosition - newStartPosition);
        float x = newDirection.x;
        float y = newDirection.y;
        float rotationAngle = Mathf.Atan2(y, x) * 180 / Mathf.PI;
        SpriteTransform.transform.rotation = Quaternion.AngleAxis(rotationAngle,
       Vector3.forward);
    }
}
```

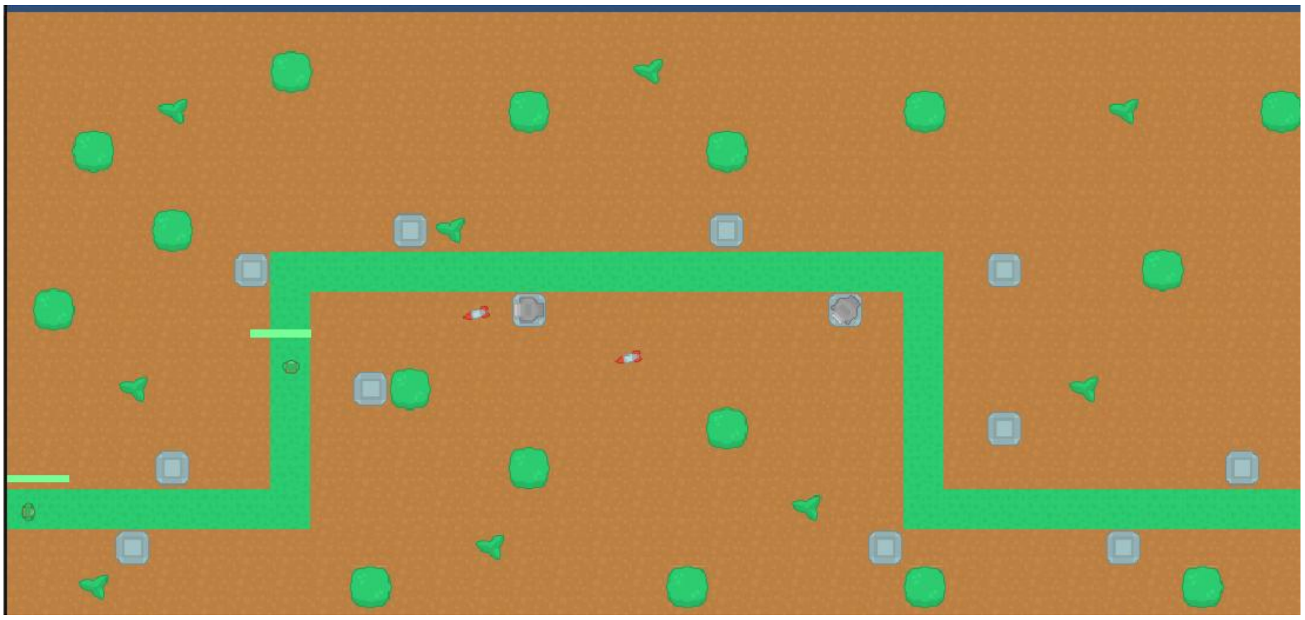
Рисунок 1 - Пример игры