Міністерство освіти та науки України

Український державний університет науки і технологій

Факультет: Комп’ютерні технології і системи

Кафедра: Комп’ютерні інформаційні технології

Спеціальність: 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедрою «КІТ»

доцент. Горячкін В. М.

"\_\_\_\_"\_\_\_\_\_\_\_\_2022 р.

З А В Д А Н Н Я

до курсового проекту

студент Кулик Сергій Вадимович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема проекту: Розробка гри «Monster Attack»

2. Дата видачі завдання 1 лютого 2022 р.

3. Термін виконання курсового проекту  **28 серпня 2022 року**

Керівник курсового проектування **\_\_\_\_\_\_\_\_\_\_** //

Завдання прийняв до виконання \_\_\_\_\_\_\_\_\_\_ //

1. **Постановка задачі**

Розробка гри на об’єктно орієнтованій мові програмування. Формат гри – 2D, жанр – захист вежі. Ціль гри полягає в захисті своєї головної будівлі від атаки різних ворогів.

1. **Вимоги до програми**

*В ході розробки програмного продукту, були висунуті такі вимоги:*

1. Інтерфейс зрозумілий для користувача будь-якого віку.
2. Надійність. Збій в роботі одного елемента не повинен призводити до серйозних наслідків.
3. Гра не повинна сильно навантажувати систему та викликати зависання або збої системи.

*Вимоги до апаратної частини програми, що розробляється:*

* Наявність маніпулятору «миша»
* 100 МБ дискового простору для встановлення гри
* Операційна симтема Windows (7, 8, 10, 11)

1. **Середовище та мова програмування**

Для створення гри я обрав рушій Unity3D. Для програмування ігрової логіки використовується мова програмування C#. При написанні скриптів в Unity на C# використовується парадигма об’єктно-орієнтованого програмування, суть якої заключається в використанні в коді об’єктів, які взаємодіють між собою. Об’єктно-орієнтоване програмування покращує читабельність коду, робе його простішим, підвищує модульність програмного забезпечення, зменшується дублювання коду та підвищується його безпека.

1. **Діаграма класів**

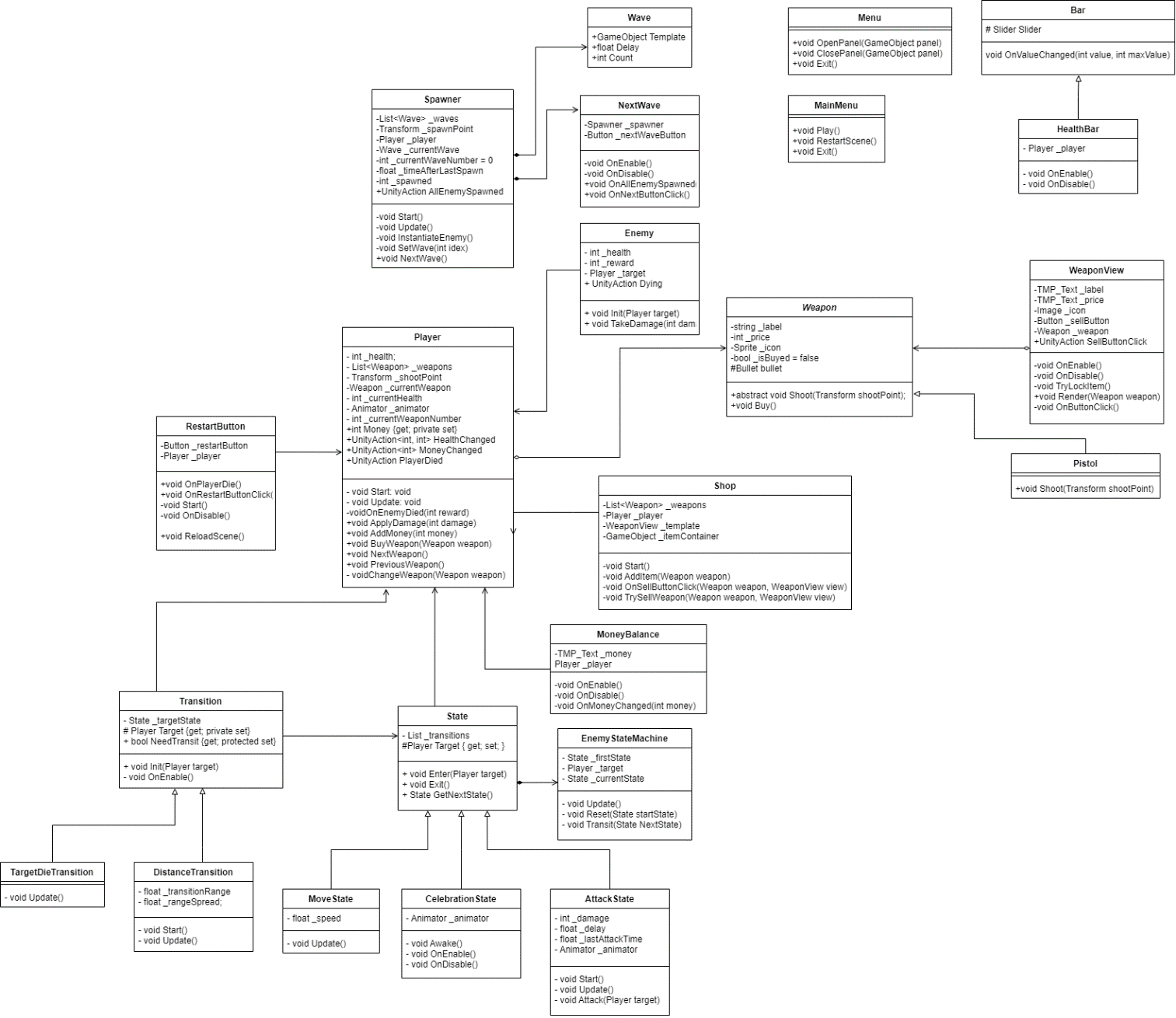
****

Рис. 1 – Діаграма класів

1. **Текст програми**

**Клас DistanceTransition**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class DistanceTransition : Transition

{

[SerializeField] private float \_transitionRange;

[SerializeField] private float \_rangeSpread;

private void Start()

{

\_transitionRange += Random.Range(\_rangeSpread, \_rangeSpread + 1);

}

private void Update()

{

if (!Target) return;

if (Vector2.Distance(transform.position, Target.transform.position) <= \_transitionRange)

{

NeedTransit = true;

}

}

}

**Клас TargetDieTransition**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class TargetDieTransition : Transition

{

private void Update()

{

if (Target == null)

{

NeedTransit = true;

}

}

}

**Клас EnemyStateMachine**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[RequireComponent(typeof(Enemy))]

public class EnemyStateMachine : MonoBehaviour

{

[SerializeField] private State \_firstState;

private Player \_target;

private State \_currentState;

public State Current => \_currentState;

private void Start()

{

\_target = GetComponent<Enemy>().Target;

Reset(\_firstState);

}

private void Update()

{

if (\_currentState == null)

return;

var nextState = \_currentState.GetNextState();

if (nextState != null)

Transit(nextState);

}

private void Reset(State startState)

{

\_currentState = startState;

if (\_currentState != null)

{

\_currentState.Enter(\_target);

}

}

private void Transit(State NextState)

{

if (\_currentState != null)

{

\_currentState.Exit();

}

\_currentState = NextState;

if (\_currentState != null)

{

\_currentState.Enter(\_target);

}

}

}

**Клас State**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public abstract class State : MonoBehaviour

{

[SerializeField] private List<Transition> \_transitions;

protected Player Target { get; set; }

public void Enter(Player target)

{

if (enabled == false)

{

Target = target;

enabled = true;

foreach (var transitions in \_transitions)

{

transitions.enabled = true;

transitions.Init(Target);

}

}

}

public void Exit()

{

if(enabled == true)

{

foreach (var transition in \_transitions)

{

transition.enabled = false;

}

enabled = false;

}

}

public State GetNextState()

{

foreach (var transition in \_transitions)

{

if (transition.NeedTransit)

{

return transition.TargetState;

}

}

return null;

}

}

**Клас Transition**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public abstract class Transition : MonoBehaviour

{

[SerializeField] private State \_targetState;

protected Player Target { get; private set; }

public State TargetState => \_targetState;

public bool NeedTransit { get; protected set; }

public void Init(Player target)

{

Target = target;

}

private void OnEnable()

{

NeedTransit = false;

}

}

**Клас AttackState**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[RequireComponent(typeof(Animator))]

public class AttackState : State

{

[SerializeField] private int \_damage;

[SerializeField] private float \_delay;

private float \_lastAttackTime;

private Animator \_animator;

private void Start()

{

\_animator = GetComponent<Animator>();

}

private void Update()

{

if(\_lastAttackTime <= 0)

{

Attack(Target);

\_animator.Play("Attack");

\_lastAttackTime = \_delay;

}

\_lastAttackTime -= Time.deltaTime;

}

private void Attack(Player target)

{

target.ApplyDamage(\_damage)

}

}

**Клас CelebrationState**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[RequireComponent(typeof(Animator))]

public class CelebrationState : State

{

private Animator \_animator;

private void Awake()

{

\_animator = GetComponent<Animator>();

}

private void OnEnable()

{

\_animator.Play("Celebrate");

}

private void OnDisable()

{

\_animator.StopPlayback();

}

}

**Клас MoveState**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class MoveState : State

{

[SerializeField] private float \_speed;

private void Update()

{

if (Target != null)

transform.position = Vector2.MoveTowards(transform.position, Target.transform.position, \_speed \* Time.deltaTime);

}

}

**Клас Enemy**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Events;

public class Enemy : MonoBehaviour

{

[SerializeField] private int \_health;

[SerializeField] private int \_reward;

private Player \_target;

public Player Target => \_target;

public int Reward => \_reward;

public event UnityAction<Enemy> Dying;

public void Init(Player target)

{

\_target = target;

}

public void TakeDamage(int damage)

{

\_health -= damage;

if (\_health <= 0)

{

Dying?.Invoke(this);

Destroy(gameObject);

}

}

}

**Клас Bar**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public abstract class Bar : MonoBehaviour

{

[SerializeField] protected Slider Slider;

public void OnValueChanged(int value, int maxValue)

{

Slider.value = (float)value / maxValue;

}

}

**Клас HealthBar**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class HealthBar : Bar

{

[SerializeField] private Player \_player;

private void OnEnable()

{

\_player.HealthChanged += OnValueChanged;

Slider.value = 1;

}

private void OnDisable()

{

\_player.HealthChanged -= OnValueChanged;

}

}

**Клас MainMenu**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour

{

public void Play()

{

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

}

public void RestartScene()

{

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

}

public void Exit()

{

Application.Quit();

}

}

**Клас Menu**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Menu : MonoBehaviour

{

public void OpenPanel(GameObject panel)

{

panel.SetActive(true);

Time.timeScale = 0;

}

public void ClosePanel(GameObject panel)

{

panel.SetActive(false);

Time.timeScale = 1;

}

public void Exit()

{

Application.Quit();

}

}

**Клас MoneyBalance**

using System.Collections;

using System.Collections.Generic;

using TMPro;

using UnityEngine;

public class MoneyBalance : MonoBehaviour

{

[SerializeField] private TMP\_Text \_money;

[SerializeField] private Player \_player;

private void OnEnable()

{

\_money.text = \_player.Money.ToString();

\_player.MoneyChanged += OnMoneyChanged;

}

private void OnDisable()

{

\_player.MoneyChanged -= OnMoneyChanged;

}

private void OnMoneyChanged(int money)

{

\_money.text = money.ToString();

}

}

**Клас NextWave**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class NextWave : MonoBehaviour

{

[SerializeField] private Spawner \_spawner;

[SerializeField] private Button \_nextWaveButton;

private void OnEnable()

{

\_spawner.AllEnemySpawned += OnAllEnemySpawned;

\_nextWaveButton.onClick.AddListener(OnNextButtonClick);

}

private void OnDisable()

{

\_spawner.AllEnemySpawned -= OnAllEnemySpawned;

\_nextWaveButton.onClick.RemoveListener(OnNextButtonClick);

}

public void OnAllEnemySpawned()

{

\_nextWaveButton.gameObject.SetActive(true);

}

public void OnNextButtonClick()

{

\_spawner.NextWave();

\_nextWaveButton.gameObject.SetActive(false);

}

}

**Клас RestartButton**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class RestartButoon : MonoBehaviour

{

[SerializeField] private Button \_restartButton;

[SerializeField] private Player \_player;

public void OnPlayerDie()

{

\_restartButton.gameObject.SetActive(true);

}

public void OnRestartButtonClick()

{

\_restartButton.gameObject.SetActive(false);

\_restartButton.onClick.RemoveListener(OnRestartButtonClick);

}

private void Start()

{

\_restartButton.gameObject.SetActive(false);

\_player.PlayerDied += OnPlayerDie;

\_restartButton.onClick.AddListener(OnRestartButtonClick);

}

private void OnDisable()

{

\_player.PlayerDied -= OnPlayerDie;

}

public void ReloadScene()

{

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

}

}

**Клас Shop**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Shop : MonoBehaviour

{

[SerializeField] private List<Weapon> \_weapons;

[SerializeField] private Player \_player;

[SerializeField] private WeaponView \_template;

[SerializeField] private GameObject \_itemContainer;

private void Start()

{

for (int i = 0; i < \_weapons.Count; i++)

{

AddItem(\_weapons[i]);

}

}

private void AddItem(Weapon weapon)

{

var view = Instantiate(\_template, \_itemContainer.transform);

view.SellButtonClick += OnSellButtonClick;

view.Render(weapon);

}

private void OnSellButtonClick(Weapon weapon, WeaponView view)

{

TrySellWeapon(weapon, view);

}

private void TrySellWeapon(Weapon weapon, WeaponView view)

{

if (weapon.Price <= \_player.Money)

{

\_player.BuyWeapon(weapon);

weapon.Buy();

view.SellButtonClick -= OnSellButtonClick;

}

}

}

**Клас WeaponView**

using System.Collections;

using System.Collections.Generic;

using TMPro;

using UnityEngine;

using UnityEngine.Events;

using UnityEngine.UI;

public class WeaponView : MonoBehaviour

{

[SerializeField] private TMP\_Text \_label;

[SerializeField] private TMP\_Text \_price;

[SerializeField] private Image \_icon;

[SerializeField] private Button \_sellButton;

private Weapon \_weapon;

public event UnityAction<Weapon, WeaponView> SellButtonClick;

private void OnEnable()

{

\_sellButton.onClick.AddListener(OnButtonClick);

\_sellButton.onClick.AddListener(TryLockItem);

}

private void OnDisable()

{

\_sellButton.onClick.RemoveListener(OnButtonClick);

\_sellButton.onClick.RemoveListener(TryLockItem);

}

private void TryLockItem()

{

if (\_weapon.IsBuyed)

{

\_sellButton.interactable = false;

}

}

public void Render(Weapon weapon)

{

\_weapon = weapon;

\_label.text = weapon.Label;

\_price.text = weapon.Price.ToString();

\_icon.sprite = weapon.Icon;

}

private void OnButtonClick()

{

SellButtonClick?.Invoke(\_weapon, this);

}

}

**Клас Weapon**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public abstract class Weapon : MonoBehaviour

{

[SerializeField] private string \_label;

[SerializeField] private int \_price;

[SerializeField] private Sprite \_icon;

[SerializeField] private bool \_isBuyed = false;

[SerializeField] protected Bullet bullet;

public string Label => \_label;

public int Price => \_price;

public Sprite Icon => \_icon;

public bool IsBuyed => \_isBuyed;

public Bullet Bullet => bullet;

public abstract void Shoot(Transform shootPoint);

public void Buy()

{

\_isBuyed = true;

}

}

**Клас Pistol**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Pistol : Weapon

{

public override void Shoot(Transform shootPoint)

{

Bullet.MousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);

Bullet.TargetPosition = (Bullet.MousePosition - (Vector3)shootPoint.position).normalized;

Instantiate(Bullet, shootPoint.position, Quaternion.identity);

}

}

**Клас Bullet**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Bullet : MonoBehaviour

{

[SerializeField] private int \_damage;

[SerializeField] private float \_speed;

public Vector2 \_mousePosition;

public Vector3 \_targetPosition;

public Vector3 MousePosition

{

get { return \_mousePosition; }

set { \_mousePosition = value; }

}

public Vector3 TargetPosition

{

get { return \_targetPosition; }

set { \_targetPosition = value; }

}

private void Start()

{

var angle = Mathf.Atan2(TargetPosition.y, TargetPosition.x) \* Mathf.Rad2Deg;

transform.rotation = Quaternion.Euler(0, 0, angle + 180);

}

private void Update()

{

transform.Translate(TargetPosition \* \_speed \* Time.deltaTime, Space.World);

}

private void OnTriggerEnter2D(Collider2D collision)

{

if (collision.gameObject.TryGetComponent(out Enemy enemy))

{

enemy.TakeDamage(\_damage);

Destroy(gameObject);

}

if (collision.gameObject.name == "DestroyBullet")

{

Destroy(gameObject);

}

}

}

**Клас Player**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Events;

[RequireComponent(typeof(Animator))]

public class Player : MonoBehaviour

{

[SerializeField] private int \_health;

[SerializeField] private List<Weapon> \_weapons;

[SerializeField] private Transform \_shootPoint;

private Weapon \_currentWeapon;

private int \_currentHealth;

private Animator \_animator;

private int \_currentWeaponNumber;

public int Health => \_health;

public int Money { get; private set; }

public event UnityAction<int, int> HealthChanged;

public event UnityAction<int> MoneyChanged;

public event UnityAction PlayerDied;

private void Start()

{

ChangeWeapon(\_weapons[\_currentWeaponNumber]);

\_currentWeapon = \_weapons[0];

\_currentHealth = \_health;

\_animator = GetComponent<Animator>();

}

private void Update()

{

if (Input.GetKeyDown(KeyCode.Mouse0))

{

\_animator.Play("Shoot");

\_currentWeapon.Shoot(\_shootPoint);

}

}

private void OnEnemyDied(int reward)

{

Money += reward;

}

public void ApplyDamage(int damage)

{

\_currentHealth -= damage;

HealthChanged?.Invoke(\_currentHealth, \_health);

if (\_currentHealth <= 0)

{

PlayerDied?.Invoke();

Destroy(gameObject);

}

}

public void AddMoney(int money)

{

Money += money;

MoneyChanged?.Invoke(Money);

}

public void BuyWeapon(Weapon weapon)

{

Money -= weapon.Price;

MoneyChanged?.Invoke(Money);

\_weapons.Add(weapon);

}

public void NextWeapon()

{

if (\_currentWeaponNumber == \_weapons.Count - 1)

{

\_currentWeaponNumber = 0;

}

else

{

\_currentWeaponNumber++;

}

ChangeWeapon(\_weapons[\_currentWeaponNumber]);

}

public void PreviousWeapon()

{

if (\_currentWeaponNumber == 0)

{

\_currentWeaponNumber = \_weapons.Count - 1;

}

else

{

\_currentWeaponNumber--;

}

ChangeWeapon(\_weapons[\_currentWeaponNumber]);

}

private void ChangeWeapon(Weapon weapon)

{

\_currentWeapon = weapon;

}

}

**Клас Spawner**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Events;

public class Spawner : MonoBehaviour

{

[SerializeField] private List<Wave> \_waves;

[SerializeField] private Transform \_spawnPoint;

[SerializeField] private Player \_player;

private Wave \_currentWave;

private int \_currentWaveNumber = 0;

private float \_timeAfterLastSpawn;

private int \_spawned;

public event UnityAction AllEnemySpawned;

private void Start()

{

SetWave(\_currentWaveNumber);

}

private void OnEnemyDying(Enemy enemy)

{

enemy.Dying -= OnEnemyDying;

\_player.AddMoney(enemy.Reward);

}

private void Update()

{

if(\_currentWave == null)

{

return;

}

\_timeAfterLastSpawn += Time.deltaTime;

if (\_timeAfterLastSpawn >= \_currentWave.Delay)

{

InstantiateEnemy();

\_spawned++;

\_timeAfterLastSpawn = 0;

}

if(\_currentWave.Count<=\_spawned)

{

if(\_waves.Count>\_currentWaveNumber + 1)

{

AllEnemySpawned?.Invoke();

}

\_currentWave = null;

}

}

private void InstantiateEnemy()

{

Enemy enemy = Instantiate(\_currentWave.Template, \_spawnPoint.position, \_spawnPoint.rotation, \_spawnPoint).GetComponent<Enemy>();

enemy.Init(\_player);

enemy.Dying += OnEnemyDying;

}

private void SetWave(int idex)

{

\_currentWave = \_waves[idex];

}

public void NextWave()

{

SetWave(\_currentWaveNumber++);

\_spawned = 0;

}

}

[System.Serializable]

public class Wave

{

public GameObject Template;

public float Delay;

public int Count;

}

1. **Приклад роботи програми**

****

Рис. 2 – Головне меню



Рис. 3 – Ігровий процес

****

Рис. 4 – Вигляд гри після смерті персонажа

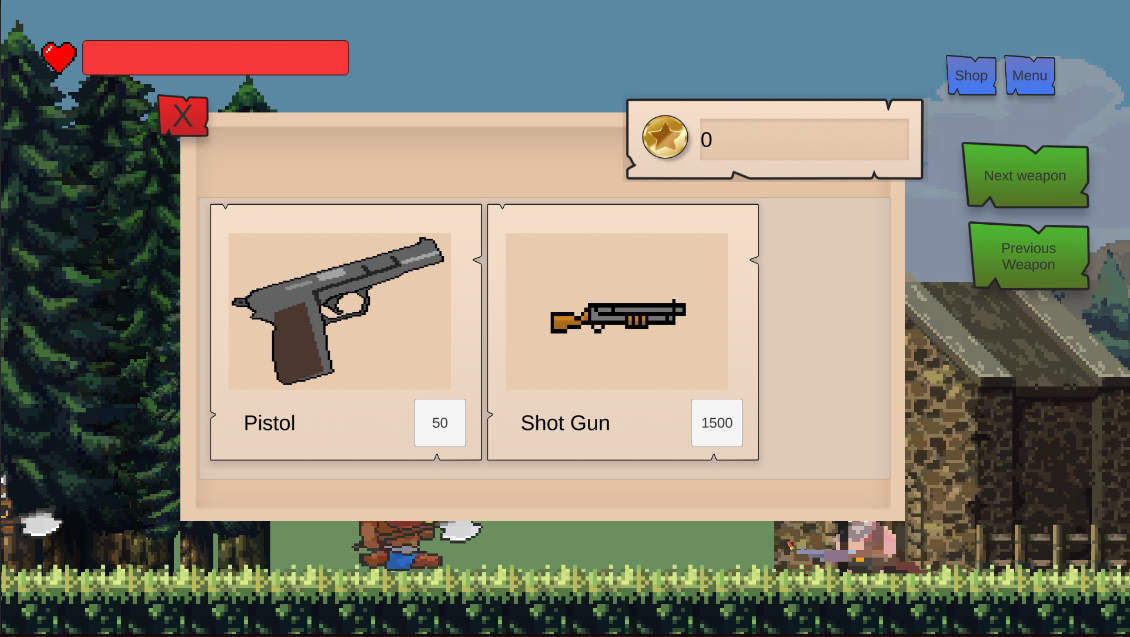


Рис. 5 – Вікно закупівлі іншої зброї

1. **Аналіз результатів**

В результаті виконання курсової роботи була розроблена та реалізована гра «Monster Attack». Головна ціль гри – відбивати хвилі монстрів, які нападають на вас та будинок. Під час розробки була реалізовані функції для налаштування хвиль противників, такі як кількість монстрів, які нападатимуть та час затримки між ініціалізацією противників. Розроблений магазин в який можна зручно додавати нові види зброї. Для поведінки монстів, була написана машина станів, яка перевіряє деякі умови переходу між станами і якщо ці умови задовільні, то перемикає стан монстра на наступний, наприклад на початок атаки або святкування перемоги. Все це було розроблено з використання об'єктно-орієнтованого програмування з дотримуванням основних принципів ООП, таких як інкапсуляція, спадкування та поліморфізм.

1. **Список використаних джерел**
2. Офіційна документація Unity [Електронний ресурс] -<https://docs.unity3d.com/Manual/index.html>
3. Об’єктно орієнтоване програмування [Електронний ресурс] - <https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>
4. Мова програмування C# [Електронний ресурс] - <https://uk.wikipedia.org/wiki/C_Sharp>
5. Машина станів [Електронний ресурс] - <https://gamedev.ru/code/articles/finite_state_machine>