ДОДАТОК А

Програмний код ігрового додатку

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

public class Enemy : MonoBehaviour
{
    private GameObject Player;
    private Rigidbody rigidbody;

    void Awake()
    {
        Player = GameObject.FindGameObjectWithTag("Player");
        rigidbody = GetComponent<Rigidbody>();
    }

    void Update()
    {
        transform.LookAt(Player.transform);
        rigidbody.AddForce(Time.deltaTime * 30f * transform.forward);
    }
}

public class EnemySpawner : MonoBehaviour
{
```

```
    public GameObject Player;
    public GameObject Enemy;

    public float Distance = 10f;

    private System.Random random;

    void Start()
    {
        random = new System.Random();
        StartCoroutine(Spawn());
    }

    private IEnumerator Spawn()
    {
        while (true)
        {
            yield return new WaitForSeconds(2f);
            var position = Player.transform.position;
            position.z = Distance * (random.Next(0, 2) * 2 - 1);
            position.x = Random.Range(position.x - Distance, position.x + Distance);
            //position.y = Random.Range(position.y - Distance, position.y + Distance);
            Instantiate(Enemy, position, Quaternion.identity).name = Enemy.name;
        }
    }
}
```

```
public class Gun : MonoBehaviour
{
    public ParticleSystem muzzleFlash;

    public float damage = 10f;
    public float range = 100f;
    public float fireRate = 50f;
    public float impactForce = 30f;

    public Camera fpsCam;

    public GameObject ImpactEffect;

    private float nextTimeToFire = 0f;

    // Update is called once per frame
    void Update()
    {
        if (Input.GetButtonDown("Fire1") && Time.time >= nextTimeToFire)
        {
            nextTimeToFire = Time.time + 1f / fireRate;
            Shoot();
        }
    }

    private void Shoot()
    {
```

```
            muzzleFlash.Play();
            RaycastHit hit;
            if (Physics.Raycast(fpsCam.transform.position, fpsCam.transform.forward,
out hit, range))
            {
                var target = hit.transform.GetComponent<Target>();
                if (target != null)
                {
                    target .TakeDamage(damage);
                }

                if(hit.rigidbody != null)
                    hit.rigidbody.AddForce(-hit.normal* impactForce);

                var impact = Instantiate(ImpactEffect, hit.point,
Quaternion.LookRotation(hit.normal));
                Destroy(impact, 6f);
            }
        }
    }

    public class MouseLook : MonoBehaviour
    {
        public float mouseSensitivity = 100f;

        public Transform playerBody;
```

```
    private float xRotation = 0f;

    void Start()
    {
        Cursor.lockState = CursorLockMode.Locked;
        Cursor.visible = false;
    }

    void Update()
    {
        float mouseX = Input.GetAxis("Mouse X") * mouseSensitivity *
Time.deltaTime;
        float mouseY = Input.GetAxis("Mouse Y") * mouseSensitivity *
Time.deltaTime;

        xRotation -= mouseY;
        xRotation = Mathf.Clamp(xRotation, -90f, 90f);

        transform.localRotation = Quaternion.Euler(xRotation, 0f, 0f);
        playerBody.Rotate(Vector3.up * mouseX);
    }
}

public class PlayerMovement : MonoBehaviour
{
    public CharacterController CharacterController;
```

```
    public Transform groundCheck;
    public float groundDistance = 0.4f;
    public LayerMask groundMask;

    public float speed = 12;
    public float gravity = -9.81f;
    public float jumpHeight = 2;

    private Vector3 velocity;
    private bool isGrounded;

    private void Update()
    {
        isGrounded = Physics.CheckSphere(groundCheck.position, groundDistance,
groundMask);

        if (isGrounded && velocity.y < 0)
            velocity.y = -2f;

        float x = Input.GetAxis("Horizontal");
        float z = Input.GetAxis("Vertical");

        Vector3 motion = transform.right * x + transform.forward * z;

        CharacterController.Move(Time.deltaTime * speed * motion);

        if (Input.GetButtonDown("Jump") && isGrounded)
```

```
        {
            velocity.y = Mathf.Sqrt(jumpHeight * -2f * gravity);
        }

        velocity.y += gravity * Time.deltaTime;

        CharacterController.Move(Time.deltaTime * velocity);
    }
}

using TMPro;

public class Score : MonoBehaviour
{
    public static Score Instance;

    public TMP_Text Text;

    private int score = 0;

    private void Awake()
    {
        Instance = this;
    }

    public void AddScore()
    {
```

```
        score += 10;
        Text.text = score.ToString();
    }
}

public class Target : MonoBehaviour
{
    public float health = 50f;

    public void TakeDamage(float amount)
    {
        health -= amount;
        if (health <= 0f)
        {
            Die();
        }
    }

    private void Die()
    {
        Score.Instance.AddScore();
        Destroy(gameObject);
    }
}
```

№ док.	Позначення	Найменування	Дод. відом.
1		Текстові документи	
	ГЮІК.ХХХХХ1649Ст.07ПЗ	Пояснювальна записка	__ с.
2		Графічні документи	
		Презентаційний матеріал	__ А4.
3		Інші документи	
		Електронна версія ПЗ	1 шт.
		Відзив керівника	1 с.
		Рецензія	1 с.

Изм.	Лист	№ докум.	Підпис	Дата	ГЮІК. ХХХХХ1649Ст.07ВД		
Розроб.		Лаухін Ю.Р.			Механізми захисту у багатокористувацьких онлайн ігрових додатків	Лит.	Лист 1 / Листів 1
Перевір.		Балагура Д.С.				Т	
Н.контр.		Конєва Н.Ф.				ХНУРЕ Кафедра БІТ	
Затверд.		Халімов Г.З.					