Wydział Matematyki Stosowanej

Development of Computer Games

3D MONSTER SHOOTER

D, Adam Wojtala:

Date: 28.01.2021

2020

Work Distribution Table:

Name/Surname	***Description of game development part***
Adam Wojtala	*Full project work, mechanics, whole game set with assets elements.*

Description of Monster Shooter Game

Description of Your Game.

1. 3D game
2. What type is your game? 3D shooter game
3. What genre is your game? *Singleplayer, survival game*
4. Platforms (mobile, PC or both?) *PC platform*
5. Scenario Description. Whole scene is in the scary dungeon. Player, which is at the start in front of first huge doors must find correct path to escape from this terrible place. In addidtion, dungeon is full of monsters, wchich player must beat to survive and win the game.

GAME FEATURES

List of features

1. Main player and character in game
2. Weapon for player
3. Health, damage for player
4. Zombie different enemy with mechanics
5. Zombie spawner
6. Headshoot monsters
7. Pick up system for player
8. Doors in game, keys for it
9. UI (main menu, screens etc.)

Feature #1. Main player and character in game

Main player in the game is made by standard assets package (reference of package in section: “Assets and resources used”). Character holds a gun (also next section), which is equipped to “FirstPersonCharacter”. “Also shootpoint” is equipped to this, which contains a mark to shoot by pistol.

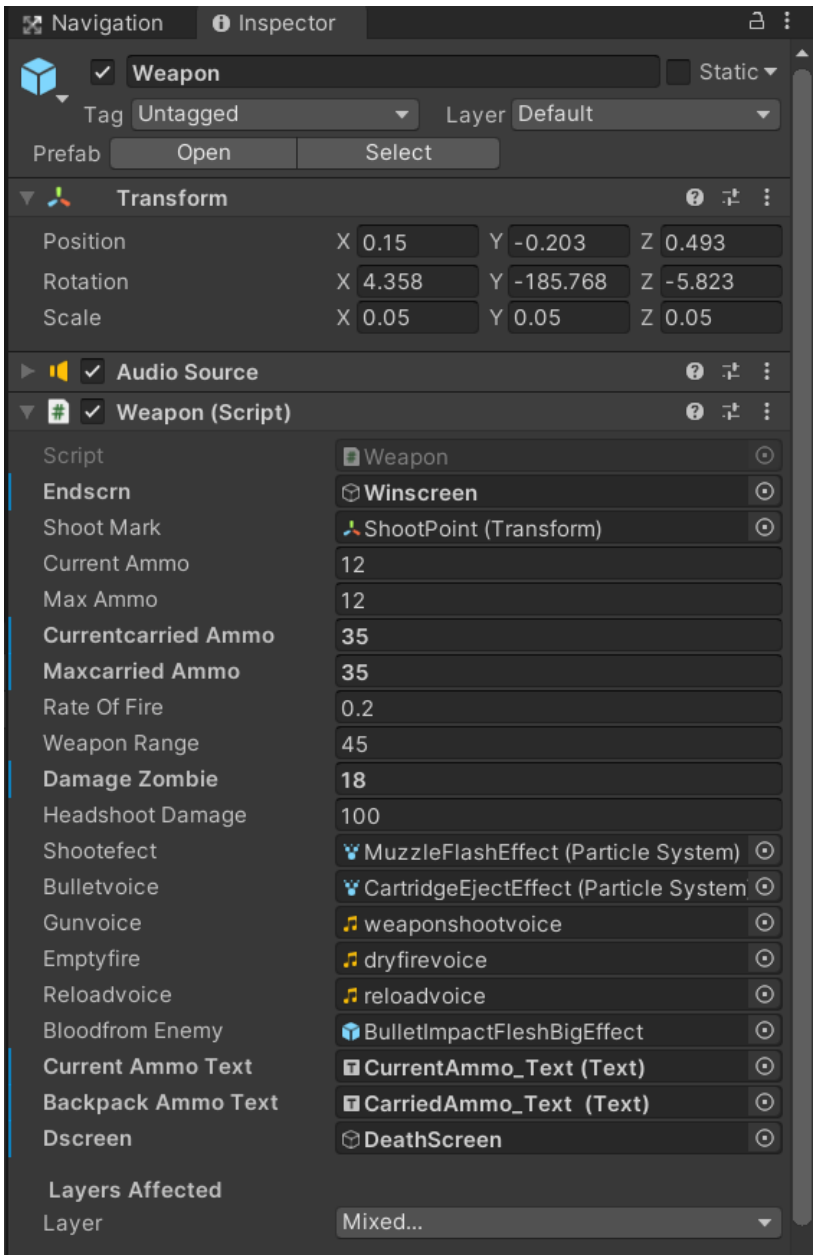
Below is the fragment of First Person Controller script, which is generated automatically by standard assets package.

```
public class FirstPersonController : MonoBehaviour
{
    [SerializeField] public bool m_IsWalking;
    [SerializeField] public float m_WalkSpeed;
    [SerializeField] public float m_RunSpeed;
    [SerializeField] [Range(0f, 1f)] public float m_RunstepLenghten;
    [SerializeField] public float m_JumpSpeed;
    [SerializeField] public float m_StickToGroundForce;
    [SerializeField] public float m_GravityMultiplier;
    [SerializeField] public MouseLook m_MouseLook;
    [SerializeField] public bool m_UseFovKick;
    [SerializeField] public FOVKick m_FovKick = new FOVKick();
    [SerializeField] public bool m_UseHeadBob;
    [SerializeField] public CurveControlledBob m_HeadBob = new CurveControlledBob();
    [SerializeField] public LerpControlledBob m_JumpBob = new LerpControlledBob();
    [SerializeField] public float m_StepInterval;
    [SerializeField] public AudioClip[] m_FootstepSounds;    // an array of footstep sounds that wil
    [SerializeField] public AudioClip m_JumpSound;           // the sound played when character leav
    [SerializeField] public AudioClip m_LandSound;           // the sound played when character touc
```

Table 1. FPC script code fragment

Feature #2. Weapon for player

Player has a weapon, which is the pistol as an element downloaded from asset store. Gun is equipped to First person controller and has functionality of shooting, reloading ammo and also dry fire voice (link in next section). Pistol has damage value, which can player shoot against zombie enemy, and its value is 18 points. Also backpack for ammo is equal to 35 and current carried ammo is 12 bullets. Other weapon are mentioned in the screen below, such as rate of fire. Player can shoot by: left mouse click and by: ctrl key, reload by Q key.



Whole weapon look can be see at first screen in features section. Script for weapon with functions, such as: Dryfire(), Shoot(), Shootray(), Reload() and other:

```
void Start()
{
    shootefect.Stop();
    Bulletvoice.Stop();
    gunshootv1 = GetComponent<AudioSource>();
    AmmoUI();
}

Unity Message | Odwołania: 0
void Update()
{
    if (Input.GetButton("Fire1") && currentAmmo > 0)
    {

        if(!Endscrn.activeSelf )
        {
            Shoot();
        }

    }
    else if (Input.GetButton("Fire1") && currentAmmo <= 0)
    {
        if (!Endscrn.activeSelf )
        {
            DryFire();
        }

    }
    else if (Input.GetKeyDown(KeyCode.Q) && currentAmmo <= maxAmmo)
    {
        if (!Endscrn.activeSelf )
        {
            Reload();
        }
    }
}
```

```
void DryFire()
{

    if (Time.time > nextFire)
    {
        nextFire = 0f;
        nextFire = Time.time + rateOfFire;

        if(!dscreen.activeSelf)
        {
            gunshootv1.PlayOneShot(emptyfire);
        }
    }
}

1 odwołanie
void Shoot()
{
        if (Time.time > nextFire)
        {
            nextFire = 0f;
            nextFire = Time.time + rateOfFire;
            currentAmmo--;

                if(!dscreen.activeSelf)
                {
                    gunshootv1.PlayOneShot(gunvoice);
                }

            StartCoroutine(ShootEffects());
            ShootRay();
            AmmoUI();
        }
}
1 odwołanie
void ShootRay()
{
    if (Physics.Raycast(shootMark.position, shootMark.forward, out hit, weaponRange, layer))
    {

        if (hit.transform.tag == "Enemy")
        {

            ZombieHealth zombiehealth = hit.transform.GetComponent<ZombieHealth>();
            zombiehealth.HealthAmount(damageZombie);
```

```
1 odwołanie
void Reload()
{
    if (currentcarriedAmmo <= 0) return;
    StartCoroutine(ReloadTime(1f));
     gunshootv1.PlayOneShot(reloadvoice);
}

1 odwołanie
IEnumerator ReloadTime(float time)
{
    while(time>0f)
    {
        isReloading = true;
        time -= Time.deltaTime;
        yield return null;
    }

    if(time<=0f)
    {
        isReloading = false;
        int ammoneeded = maxAmmo - currentAmmo;
        int AmmoCalculate = (currentcarriedAmmo >= ammoneeded) ? ammoneeded : currentcarriedAmmo;

        currentcarriedAmmo -= AmmoCalculate;
        currentAmmo += AmmoCalculate;

        AmmoUI();
    }
}
1 odwołanie
IEnumerator ShootEffects()
{
    Bulletvoice.Play();
    shootefect.Play();

    yield return new WaitForEndOfFrame();

    shootefect.Stop();
    Bulletvoice.Stop();
}

Odwołania: 4
public void AmmoUI()
{
    CurrentAmmoText.text = currentAmmo.ToString();
    BackpackAmmoText.text = currentcarriedAmmo.ToString();
}
```

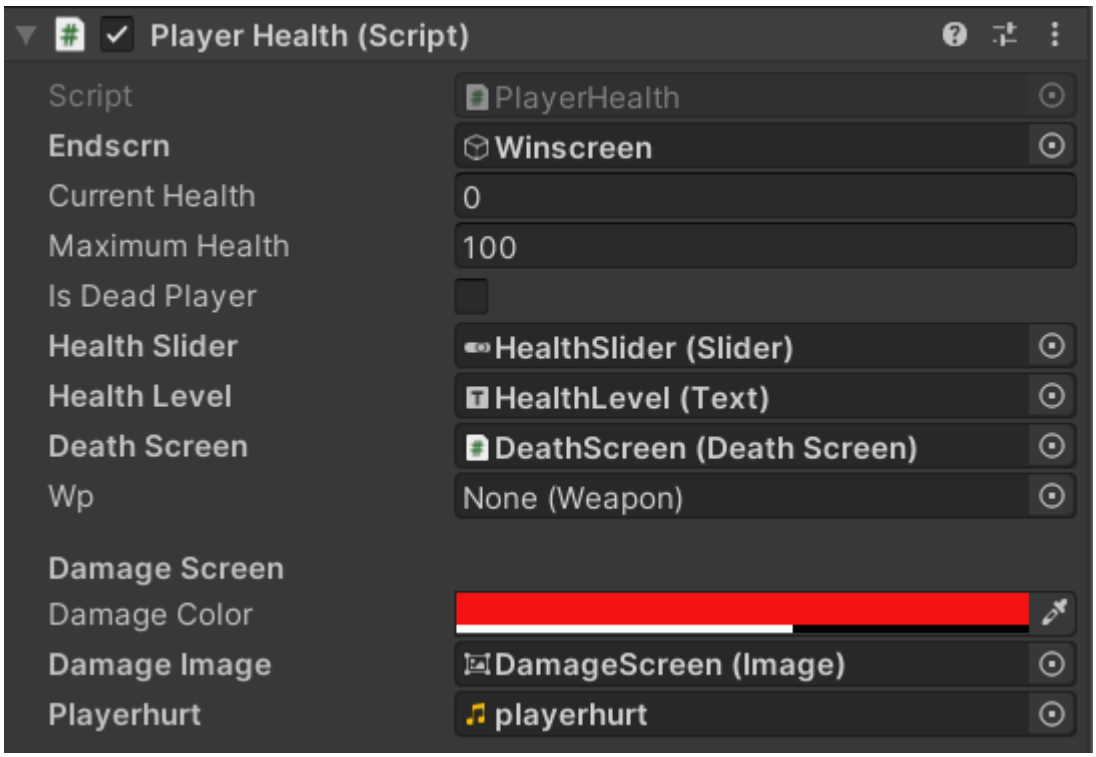
Another important function is zombie kill counter, every time player is killing one monster counter is one up. Script which tells about that:

```
public static MainUI instance;

[SerializeField]
TextMeshProUGUI numofkills;

[HideInInspector]
public int killsnumber;

Unity Message | Odwołania: 0
private void Awake()
{
    if(instance==null)
    {
        instance = this;
    }
    else
    {
        Destroy(gameObject);
    }
}

1 odwołanie
public void Updatenumofkills()
{
    numofkills.text = killsnumber.ToString();
}
```

Feature #3. Health, damage for player:

Player has a health bar with 100 points of health, every time monster attack player he is losing his hp, when health bar is equal to 0, player is dead. When zombie attack player, blood damage screen is activated like in the screen below. When also player is losing his hp, the hurt sound is activated.

Player health options in inspector of unity editor:



Script for player health with a couple important functions, such as: DamagePlayer(), UpdateHealthUI(), DeadPlayer():

```
void Start()
{

    CurrentHealth = MaximumHealth;
    healthSlider.value = MaximumHealth;
    UpdateHealthUI();

    PlayerHurt = GetComponentInParent<AudioSource>();
}

// Update is called once per frame
Unity Message | Odwołania: 0
void Update()
{
    if (CurrentHealth < 0)
    {
        CurrentHealth = 0;
    }


    if (isDamaged)
    {
        damageImage.color = damageColor;
    }
    else
    {
        damageImage.color = Color.Lerp(damageImage.color, Color.clear, colorSmoothing * Time.deltaTime);
    }

    if (!IsDeadPlayer)
    {
        isDamaged = false;
    }
```

```
public void DamagePlayer(float damage)
{

    if (CurrentHealth > 0)
    {
        if (damage >= CurrentHealth)
        {

            isDamaged = true;
            DeadPlayer();
        }
        else
        {

            isDamaged = true;
            CurrentHealth -= damage;

            if(!Endscrn.activeSelf)
            {
                PlayerHurt.PlayOneShot(playerhurt);
            }
        }
        UpdateHealthUI();
    }
}

Odwołania: 5
public void UpdateHealthUI()
{
    healthLevel.text = CurrentHealth.ToString();
    healthSlider.value = CurrentHealth;
}

1 odwołanie
void DeadPlayer()
{
    CurrentHealth = 0;
    IsDeadPlayer = true;
    healthSlider.value = 0;
    if (!Endscrn.activeSelf)
    {
        PlayerHurt.PlayOneShot(playerhurt);
    }
    Debug.Log("Player is dead");
    UpdateHealthUI();
    deathScreen.GameOverscreen();
}
```

Feature #4. Zombie enemy with mechanics:

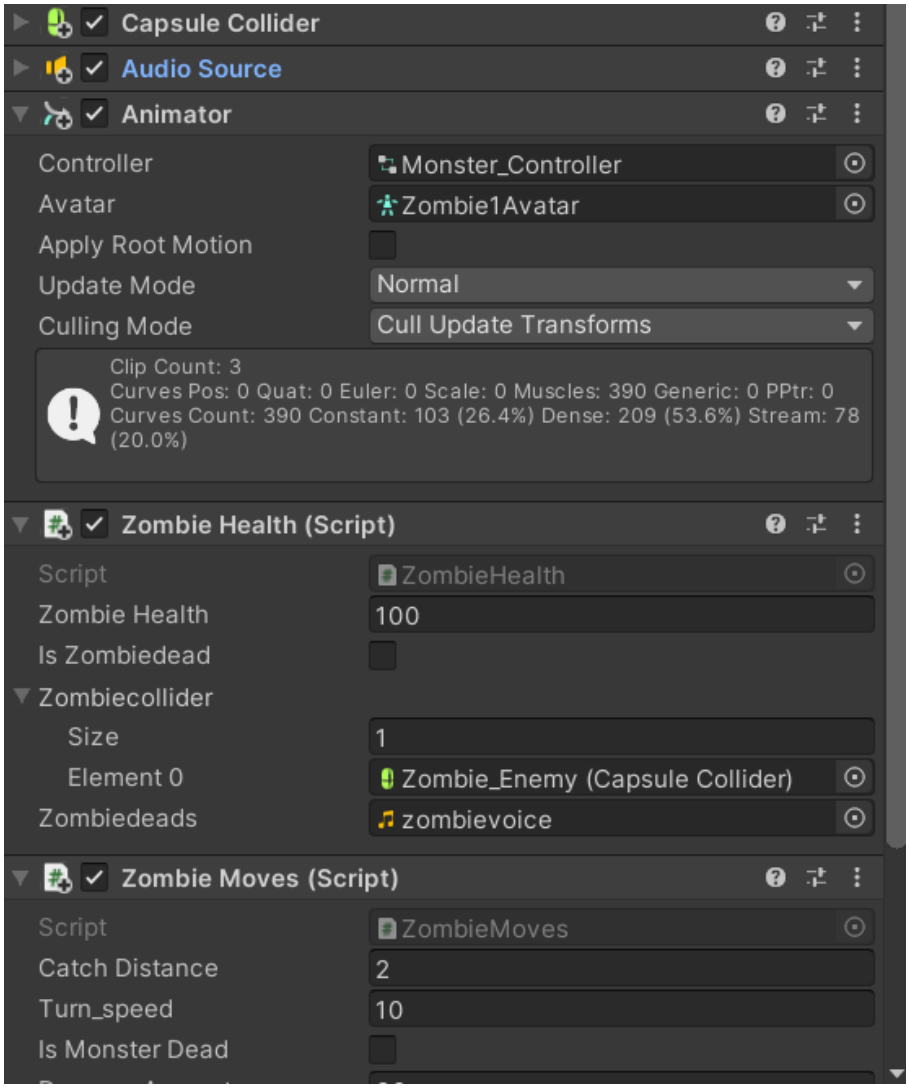
Zombies are main enemies for player. Monster objects have nav mesh agent, so they can find player character in every place in game world. Zombies have also health points, they are equal to 100 points. Every time zombie gets shoot from player, there is a blood animation according with it. Zombie has walking, attack and death animation.

In game there are four type of zombies:

- Zombie normal: middle speed, middle health points and middle damage from attack.

- Zombie small: fast speed, low heal, and low attack.
- Zombie big: low speed, high health and high attack points.
- Zombie final boss: middle speed, very high health points and very high attack.

Example options of zombie (normal) in unity editor:



Screens about zombie in game:

Functions in script for zombie moves:

```
1 odwołanie
public void ZombieDeathAnimation()
{
    isMonsterDead = true;
    animation.SetTrigger("isDead");
}

1 odwołanie
void CatchPlayer()
{
    nav.updateRotation = true;
    nav.updatePosition = true;
    nav.SetDestination(target.position);
    animation.SetBool("isWalking", true);
    animation.SetBool("isAttacking", false);
}

1 odwołanie
void AttackPlayer()
{
    nav.updateRotation = false;
    Vector3 direction = target.position - transform.position;
    direction.y = 0;
    transform.rotation = Quaternion.Slerp(transform.rotation, Quaternion.LookRotation(direction), turn_speed * Time.deltaTime);
    nav.updatePosition = false;
    animation.SetBool("isWalking", false);
    animation.SetBool("isAttacking", true);
    StartCoroutine(AttackbyMonster());

}

1 odwołanie
IEnumerator AttackbyMonster()
{
    CanAttack = false;
    yield return new WaitForSeconds(0.5f);

    if (!isMonsterDead)
    {
        PlayerHealth.singleton.DamagePlayer(DamageAmount);
    }

    yield return new WaitForSeconds(AttackTimes);
    CanAttack = true;
}

1 odwołanie
void StopMonster()
{
    CanAttack = false;
    animation.SetBool("isWalking", false);
    animation.SetBool("isAttacking", false);
}
```

Scripts for zombie health:

```
public float zombieHealth = 100f;
ZombieMoves zombieAI;

public bool isZombiedead;
public Collider[] zombiecollider;

AudioSource ZombieDeads;

[SerializeField]
AudioClip zombiedeads;

Unity Message | Odwołania: 0
private void Start()
{
    zombieAI = GetComponent<ZombieMoves>();
    ZombieDeads = GetComponentInParent<AudioSource>();
}

1 odwołanie
public void HealthAmount(float lesshealth)
{
    if(!isZombiedead)
    {
        zombieHealth -= lesshealth;

        if (zombieHealth <= 0)
        {
            ZombieDead();
        }
    }

}

1 odwołanie
public void ZombieDead()
{

    isZombiedead = true;
    zombieAI.ZombieDeathAnimation();

    ZombieDeads.PlayOneShot(zombiedeads);

    foreach(var collider in zombiecollider)
    {
        collider.enabled = false;
    }
    zombieHealth = 0f;

    MainUI.instance.killsnumber++;
    MainUI.instance.Updatenumofkills();
    Destroy(gameObject, 1);
```

Feature #5. Zombie spawner:

When player is walking in game world he is turning on zombie spawners. Then zombies appear in game and they are trying to catch and attack him. In game there are 12 different spawners, which are spawning different number of zombie objects. It is caused because OnTriggerEnter function and function called: SpawnZombies, which is presented in screen below:

```
public GameObject zombies;

public Transform[] SpawnPoints;

BoxCollider trigger;

Unity Message | Odwołania: 0
private void Start()
{
    trigger = GetComponent<BoxCollider>();

}

Unity Message | Odwołania: 0
private void OnTriggerEnter(Collider other)
{
    if(other.tag=="Player")
    {
        SpawnZombies();
        trigger.enabled = false;
    }
}

1 odwołanie
void SpawnZombies()
{

    foreach(var x in SpawnPoints)
    {
        Instantiate(zombies, x.position, x.rotation);
    }


}
```

Feature #6. Headshoot monsters:

When player hits zombie in his head, zombie die automatically and special blood effect for headshott is activated, example of it:

Script for hitting head by shoot from weapon:

```
else if (hit.transform.tag == "HeadofZombie")
{
    Debug.Log("headshoot");
    ZombieHealth zombiehealth = hit.transform.GetComponentInParent<ZombieHealth>();
    zombiehealth.HealthAmount(headshootDamage);
    gunshootv1.PlayOneShot(headshootvoice);
    Instantiate(bloodfromEnemy, hit.point, transform.rotation);
   hit.transform.gameObject.SetActive(false);

}
```

Script for headshoot effect:

```
public class HeadShootZ : MonoBehaviour
{


    public GameObject blood;
    Unity Message | Odwołania: 0
    private void OnDisable()
    {

        blood.SetActive(true);
    }
}
```

Feature #7. Pick up system for player:

Player can pick up objects, such as ammo pack and he can win the game by pick up interact with final doors in the game. Every time player is close to item object, it is hint about pick up this or interact with it. Like in the screens below:

There is a special script for pickups with functions like: Pickhealth(), Pickkey(), Finalescape():

```
    ray = maincamera.ViewportPointToRay(new Vector3(0.5f, 0.5f, 0));

    if (Physics.Raycast(ray, out hit, pickdistance, layer))
    {
        textpick.enabled = true;
        textpick.text = hit.transform.name.ToString();

        if (hit.transform.tag == "AmmoPack")
        {
            infopick = "AMMO PACK (PICK 'E')";
            textpick.text = infopick;
            Pickkey();
        }
        else if (hit.transform.tag == "HealthPack")
        {

            if (PlayerHealth.singleton.CurrentHealth < PlayerHealth.singleton.MaximumHealth)
            {
                Pickhealth();
                infopick = "HEALTH PACK (PICK 'E')";
                textpick.text = infopick;
            }
            else
            {
                infopick = "HEALTH FULL!";
                textpick.text = infopick;

            }

        }
        else if (hit.transform.tag == "Wingame")
        {
            infopick = "CLICK 'M' TO FINAL \n ESCAPE FROM DUNGEON!";
            textpick.text = infopick;
            Finalescape();

        }
    }
```

```
1 odwołanie
void Pickkey()
{
    if (Input.GetKeyDown(KeyCode.E))
    {
        Destroy(hit.transform.gameObject);
        weaponScript.currentcarriedAmmo += 15;
        picksound.PlayOneShot(pickaudioammosound);
        textpick.enabled = false;
        weaponScript.AmmoUI();
    }
}

1 odwołanie
void Finalescape()
{
    if(Input.GetKeyDown(KeyCode.M))
    {
        picksound.PlayOneShot(winsound);
        Endscrn.gameObject.SetActive(true);
    }
}

Odwołania: 0
IEnumerator Wait2()
{
    yield return new WaitForSeconds(2);
}

1 odwołanie
void Pickhealth()
{
    if (Input.GetKeyDown(KeyCode.E))
    {
        Destroy(hit.transform.gameObject);
        picksound.PlayOneShot(pickaudiohealthsound);
        textpick.enabled = false;


        HealthPick healthpickscript = hit.transform.GetComponent<HealthPick>();

        float healthbox = healthpickscript.healthbox;

        if (PlayerHealth.singleton.CurrentHealth + healthbox > PlayerHealth.singleton.MaximumHealth)
        {
            PlayerHealth.singleton.CurrentHealth = PlayerHealth.singleton.MaximumHealth;
            PlayerHealth.singleton.UpdateHealthUI();
        }
        else
        {
            PlayerHealth.singleton.Addhealth(healthbox);
        }
    }
}
```

Feature #8. Doors in game, keys for it

In game there are some types for doors, locked and free ones. For free doors we can click ‘E’ key to open them, but for locked doors we need special key. Keys are hidden in game rooms, and player must find them to open doors and go next. In game there are four keys and four types of locked doors:

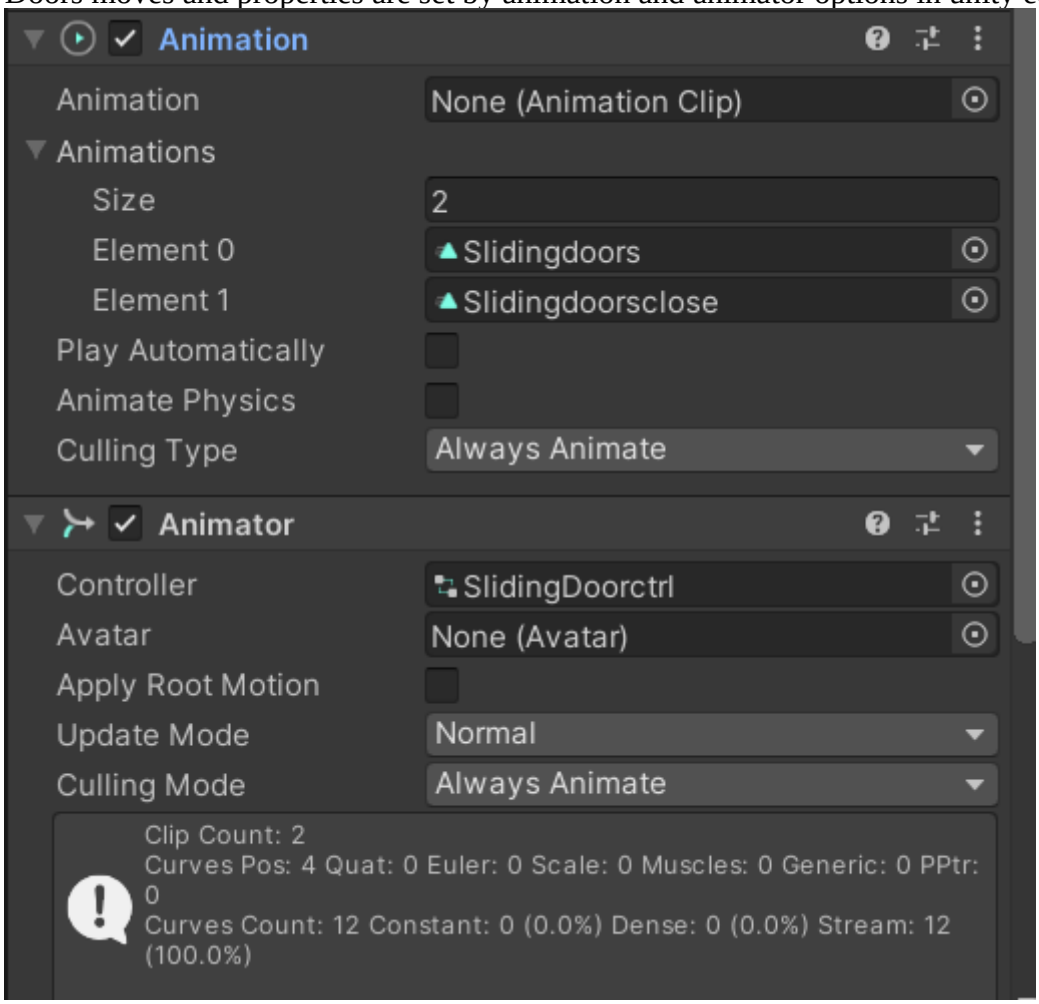
- White,
- Grey,
- Dark,
- Gold

When player also put a mouse direct to locked doors, he will get instruction which key is required for them.

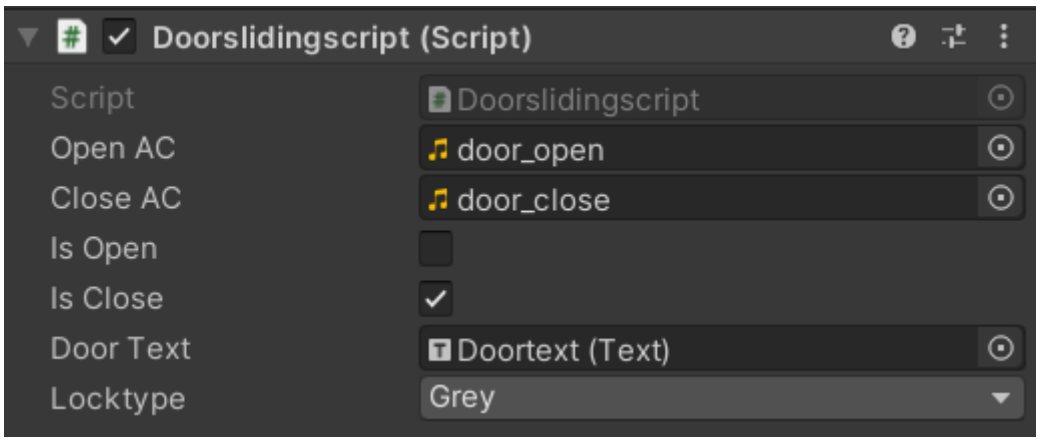
Example:

Free doors instruction:

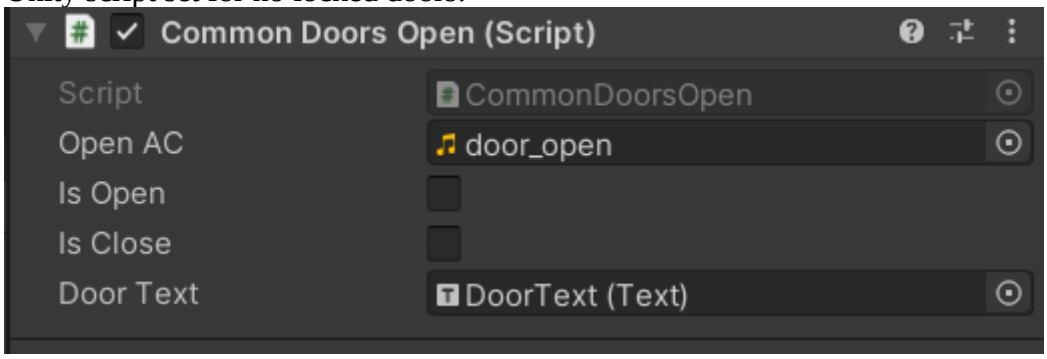
Doors moves and properties are set by animation and animator options in unity editor:



Unity set door script locked properties:



Unity script set for no locked doors:



Script for information at locked doors:

```
        }
        break;
    case "WhiteKeyDoors":
        if (hit.transform.tag == "WhiteKeyDoors")
        {
            infopick = "WHITE KEY REQUIRED TO OPEN THIS DOOR! (PICK 'E')";
            textpick.text = infopick;
        }
        break;
    case "DarkKeyDoors":
        if (hit.transform.tag == "DarkKeyDoors")
        {
            infopick = "DARK KEY REQUIRED TO OPEN THIS DOOR! (PICK 'E')";
            textpick.text = infopick;
        }
        break;
    case "GoldKeyDoors":
        if (hit.transform.tag == "GoldKeyDoors")
        {
            infopick = "GOLD KEY REQUIRED TO OPEN THIS DOOR! (PICK 'E')";
            textpick.text = infopick;
        }
        break;
    case "GreyKeyDoors":
        if (hit.transform.tag == "GreyKeyDoors")
        {
            infopick = "GREY KEY REQUIRED TO OPEN THIS DOOR! (PICK 'E')";
            textpick.text = infopick;
        }
        break;
    case "Keys":
        PickUpKey();
        break;
    default:
        break;
```

Script for door lock type:

```
void CheckTypeoflock()
{
    switch (locktype)
    {

        case LockType.White:
            if(MainScript.hasWhiteKey)
            {
                isClose = false;
                OpenDoors();
            }
            break;
        case LockType.Dark:
            if (MainScript.hasDarkKey)
            {
                isClose = false;
                OpenDoors();
            }
            break;
        case LockType.Gold:
            if (MainScript.hasGoldKey)
            {
                isClose = false;
                OpenDoors();
            }
            break;
        case LockType.Grey:
            if (MainScript.hasGreyKey)
            {
                isClose = false;
                OpenDoors();
            }
            break;
        default:
            break;
```

Script for opening doors:

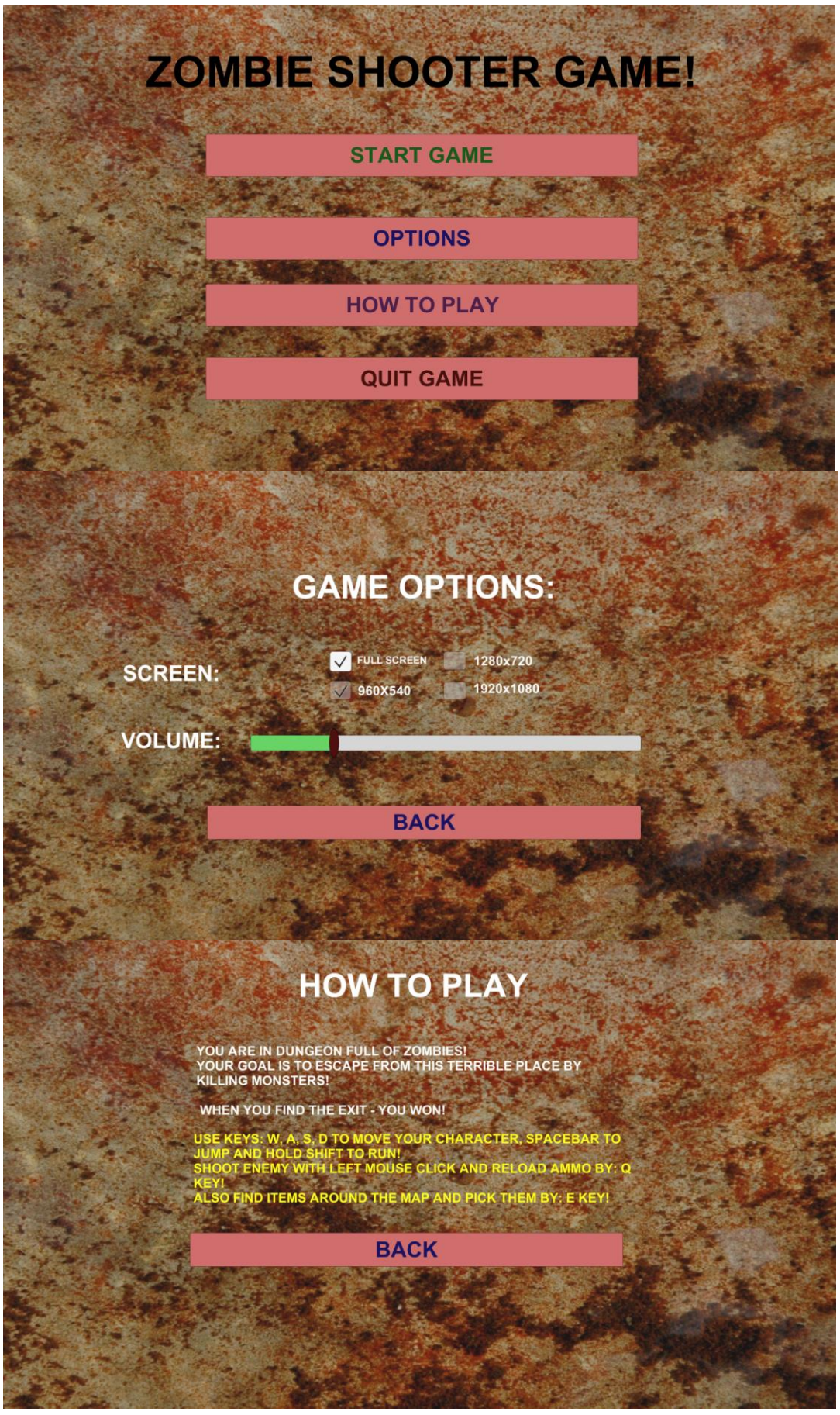
```
public void OpenDoors()
{
    isOpen = true;

    anim.SetTrigger("Open");
    doorAS.PlayOneShot(openAC);
    doorText.enabled = false;
}

1 odwołanie
public void CloseDoors()
{
    isOpen = false;
    anim.SetTrigger("Close");
    doorAS.PlayOneShot(closeAC);

}

Odwołania: 0
void UpdateDoorUI()
{
    doorText.enabled = true;
    doorText.text = doorstring.ToString();
}


Unity Message | Odwołania: 0
private void OnTriggerStay(Collider other)
{
    if(other.tag=="Player")
    {
        if(!isClose && !isOpen)
        {
            doorstring = "OPEN DOORS ('E' CLICK)!";


            if(Input.GetKeyDown(KeyCode.E))
            {
                OpenDoors();

                doorText.enabled = false;

            }
```

Script for keys set:

```
public Text keytextpickwhite;
public Text keytextpickdark;
public Text keytextpickgold;
public Text keytextpickgrey;

Odwołania: 5
public enum KeyType
{

    White, Dark, Gold, Grey
}

public KeyType keytype;

Unity Message | Odwołania: 0
private void OnDestroy()
{
    switch (keytype)
    {
        case KeyType.White:
            MainScript.hasWhiteKey = true;
            EnKeytextwhite();
            break;
        case KeyType.Dark:
           EnKeytextdark();
            MainScript.hasDarkKey = true;
            break;
        case KeyType.Gold:
            MainScript.hasGoldKey = true;
            EnKeytextgold();
            break;
        case KeyType.Grey:
            MainScript.hasGreyKey = true;
            EnKeytextgrey();
            break;
        default:
            break;
    }
```

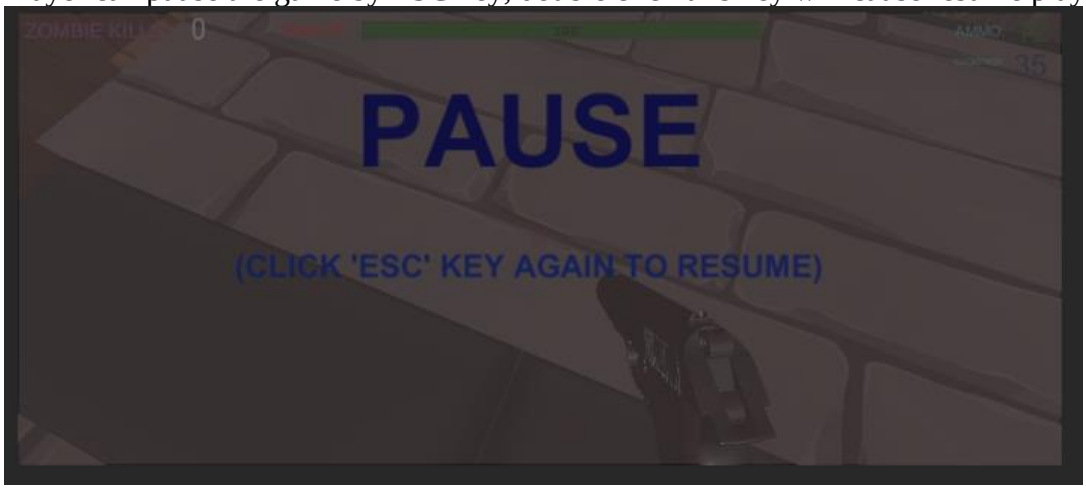
Feature #9. UI (Main menu, different screens, etc)

Game starts with main menu screen. It has opportunely functions like: game start, options with sound voice level, screen resolution or instruction how to play and also quit game function:

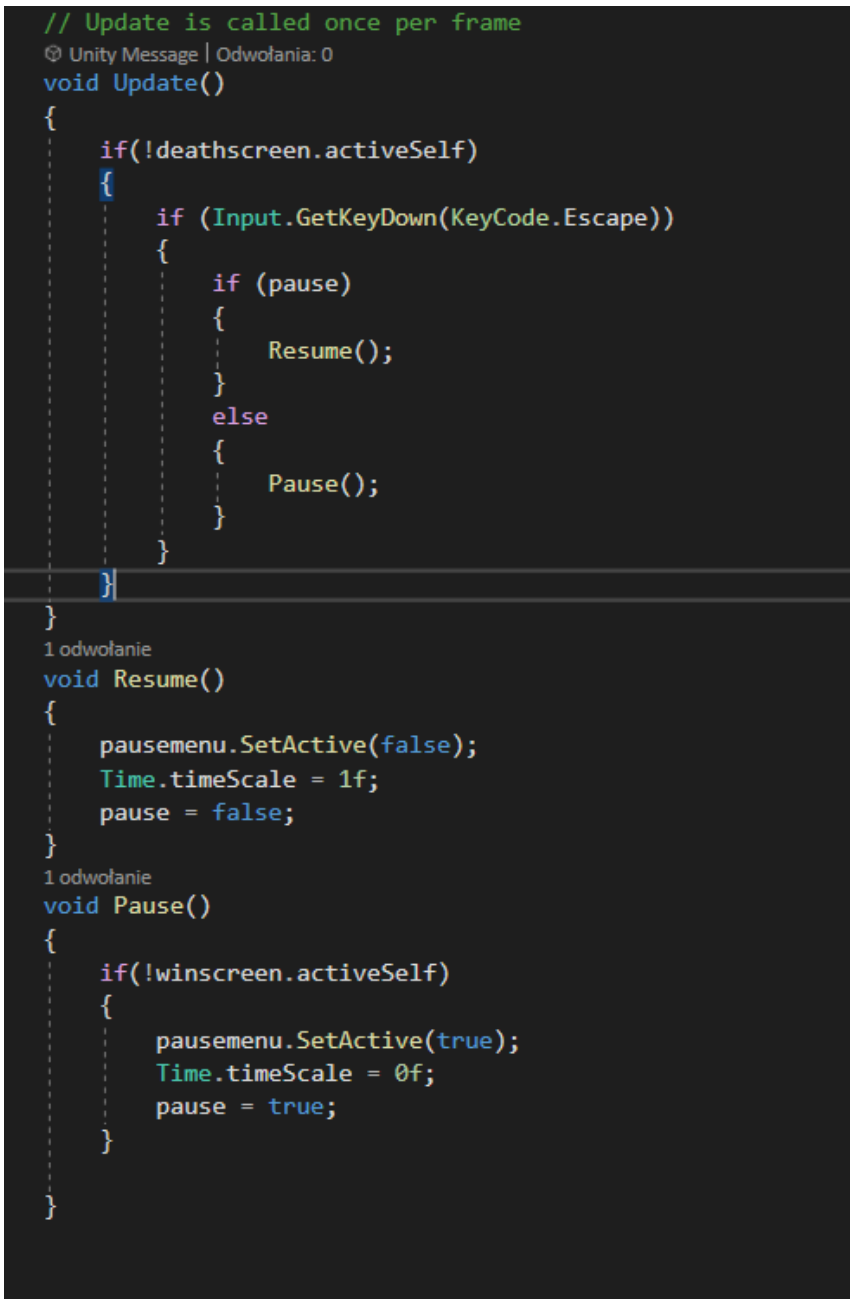


There are two special screens about game over and winning game:

Player can pause the game by ESC key, double click this key will cause resume playing.

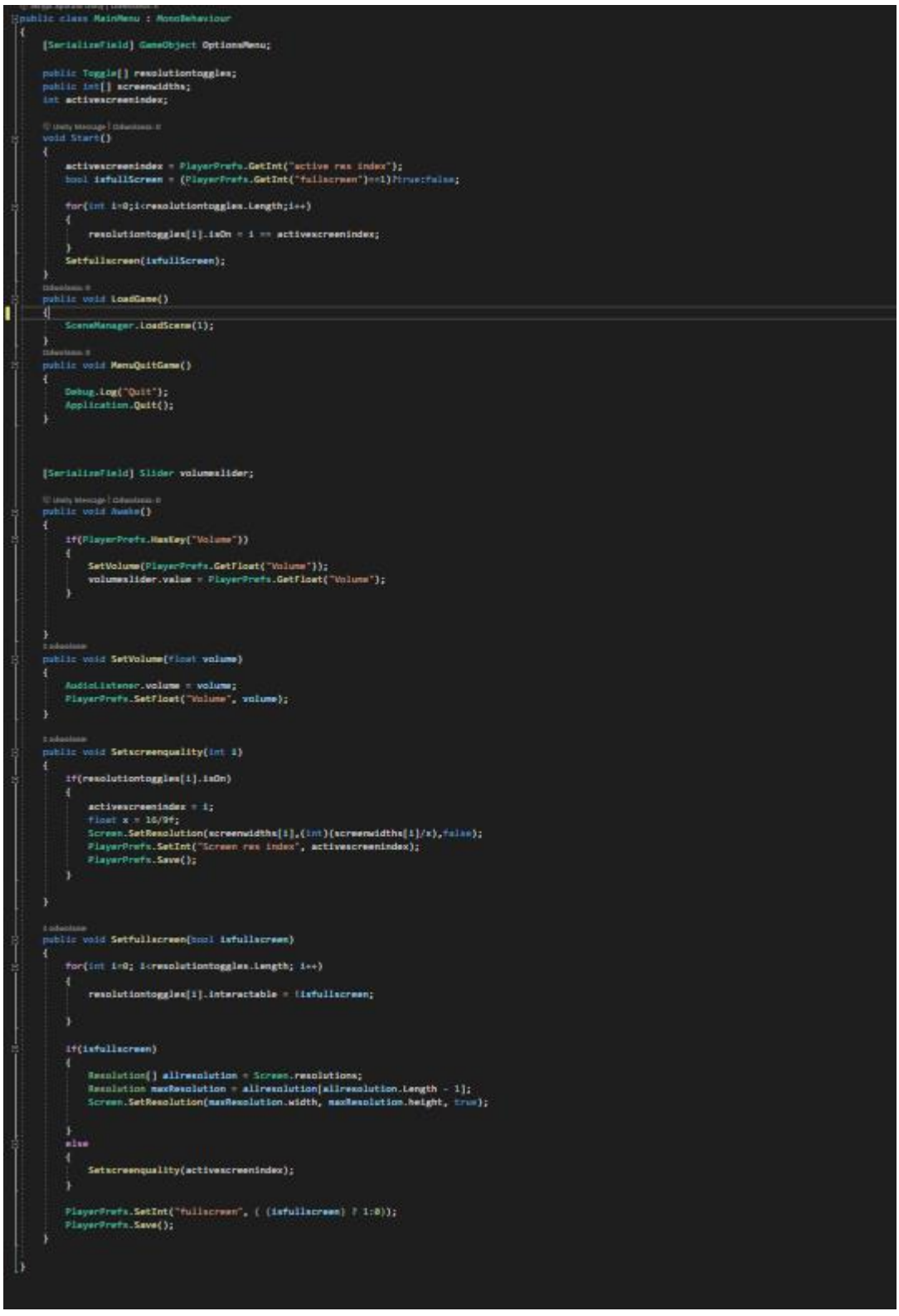


Script for pause system:



```
// Update is called once per frame
Unity Message | Odwołania: 0
void Update()
{
    if(!deathscreen.activeSelf)
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (pause)
            {
                Resume();
            }
            else
            {
                Pause();
            }
        }
    }
}
1 odwołanie
void Resume()
{
    pausemenu.SetActive(false);
    Time.timeScale = 1f;
    pause = false;
}
1 odwołanie
void Pause()
{
    if(!winscreen.activeSelf)
    {
        pausemenu.SetActive(true);
        Time.timeScale = 0f;
        pause = true;
    }

}
```

Script for main menu configuration:



```
public class MainMenu : MonoBehaviour
{
    [SerializeField] GameObject OptionsMenu;

    public Toggle[] resolutiontoggles;
    public int[] screenwidths;
    int activescreenindex;

    void Start()
    {
        activescreenindex = PlayerPrefs.GetInt("active res index");
        bool isfullScreen = (PlayerPrefs.GetInt("fullscreen")==1)?true:false;

        for(int i=0;i<resolutiontoggles.Length;i++)
        {
            resolutiontoggles[i].isOn = i == activescreenindex;
        }
        Setfullscreen(isfullScreen);
    }
    public void LoadGame()
    {
        SceneManager.LoadScene(1);
    }
    public void MenuQuitGame()
    {
        Debug.Log("Quit");
        Application.Quit();
    }



    [SerializeField] Slider volumeslider;

    public void Awake()
    {
        if(PlayerPrefs.HasKey("Volume"))
        {
            SetVolume(PlayerPrefs.GetFloat("Volume"));
            volumeslider.value = PlayerPrefs.GetFloat("Volume");
        }


    }
    public void SetVolume(float volume)
    {
        AudioListener.volume = volume;
        PlayerPrefs.SetFloat("Volume", volume);
    }

    public void Setscreenquality(int i)
    {
        if(resolutiontoggles[i].isOn)
        {
            activescreenindex = i;
            float x = 16/9f;
            Screen.SetResolution(screenwidths[i],(int)(screenwidths[i]/x),false);
            PlayerPrefs.SetInt("Screen res index", activescreenindex);
            PlayerPrefs.Save();
        }

    }

    public void Setfullscreen(bool isfullscreen)
    {
        for(int i=0; i<resolutiontoggles.Length; i++)
        {
            resolutiontoggles[i].interactable = !isfullscreen;

        }

        if(isfullscreen)
        {
            Resolution[] allresolution = Screen.resolutions;
            Resolution maxResolution = allresolution[allresolution.Length - 1];
            Screen.SetResolution(maxResolution.width, maxResolution.height, true);

        }
        else
        {
            Setscreenquality(activescreenindex);
        }

        PlayerPrefs.SetInt("fullscreen", ( (isfullscreen) ? 1:0));
        PlayerPrefs.Save();
    }

}
```

Scripts for correct result game sreens (game over or win):

```
    public GameObject Endscrn;

    Unity Message | Odwołania: 0
    void Start()
    {
        gameObject.SetActive(false);
    }


    1 odwołanie
    public void GameOverscreen()
    {
        if(!Endscrn.activeSelf)
        {
            gameObject.SetActive(true);
        }


    }

    Odwołania: 0
    public void ReturntoMenu()
    {
        SceneManager.LoadScene(0);

    }

    Odwołania: 0
    public void QuitGame()
    {

        Debug.Log("Exit");
        Application.Quit();

    }

    Odwołania: 0
    public void Playagain()
    {
        SceneManager.LoadScene(1);
    }
}
```

```
// Start is called before the first frame update
Unity Message | Odwołania: 0
void Start()
{
    gameObject.SetActive(false);
}

Odwołania: 0
public void Winscreen()
{
    gameObject.SetActive(true);
}

Odwołania: 0
public void QuitGame()
{
    Debug.Log("Exit");
    Application.Quit();
}
Odwołania: 0
public void Playagain()
{
    SceneManager.LoadScene(1);
}
Odwołania: 0
public void ReturntoMenu()
{
    SceneManager.LoadScene(0);

}
```

Assets and resources used

1. Texture #1. https://www.textureking.com/dsc_4439/
2. Texture #2. https://www.textureking.com/dsc_4406/
3. Texture #3. https://www.textures.com/download/SplatterRound0083/28774
4. Texture #4. https://www.textures.com/download/DoorsWoodDouble0463/117358
5. Texture #5. https://www.textures.com/download/Cargo0092/94777
6. Texture #6. https://www.textures.com/download/SignsVarious0145/107775
7. Texture #7. https://www.textures.com/download/SplatterShot0016/28767
8. Sound #1. https://freesound.org/people/fastson/sounds/399116/
9. Sound #2. https://freesound.org/search/?q=ammo+pick&f=&s=score+desc&advanced=0&g=1
10. Sound #3. https://freesound.org/people/Bird_man/sounds/275151/
11. Sound #4. https://freesound.org/people/mrickey13/sounds/515624/
12. Sound #5. https://freesound.org/people/nioczkus/sounds/396331/
13. Sound #6. https://freesound.org/people/EricsSoundschmiede/sounds/463721/
14. Sound #7. https://freesound.org/people/Loyalty_Freak_Music/sounds/407479/
15. Sound #8. https://freesound.org/people/Tuudurt/sounds/258142/
16. Sound #9. https://freesound.org/people/LittleRobotSoundFactory/sounds/270467/
17. Sound #10. https://freesound.org/people/SwagMuffinPlus/sounds/176146/
18. Asset #1. https://assetstore.unity.com/packages/3d/props/guns/pbr-rov-free-edition-53060
19. Asset #2. https://assetstore.unity.com/packages/3d/environments/stylized-hand-painted-dungeon-free-173934
20. Asset #3. https://assetstore.unity.com/packages/3d/characters/humanoids/zombie-30232
21. Asset #4. https://assetstore.unity.com/packages/3d/environments/3d-free-modular-kit-85732
22. Asset #5. https://assetstore.unity.com/packages/essentials/asset-packs/standard-assets-for-unity-2018-4-32351
23. Asset #6. https://assetstore.unity.com/packages/vfx/particles/blood-gush-73426
24. Asset #7. https://assetstore.unity.com/packages/3d/fps-akm-model-textures-63654
25. Asset #8. https://assetstore.unity.com/packages/essentials/asset-packs/unity-particle-pack-5-x-73777
26. Asset #9. https://assetstore.unity.com/packages/vfx/particles/blood-gush-73426
27. Asset #10. https://assetstore.unity.com/packages/3d/handpainted-keys-42044