VR Shooter

106/11/24

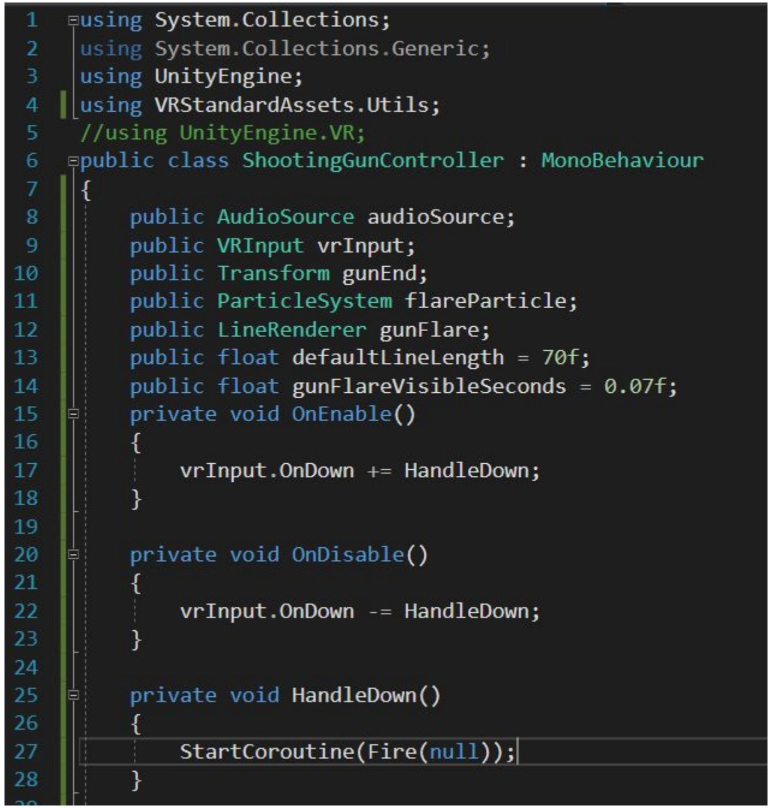
Setup Envirnoment

- 替換Camera to VR SampleScenes/Prefabs/Utils/MainCamera
- Switch platform to Android
- Setup VR Supported -> Cardboard SDK
- Minimun API Set to level 19
- Add Gvr Editor Emulator to MainCamera
- Use ALT + Mouse Test it!
- Save Scene

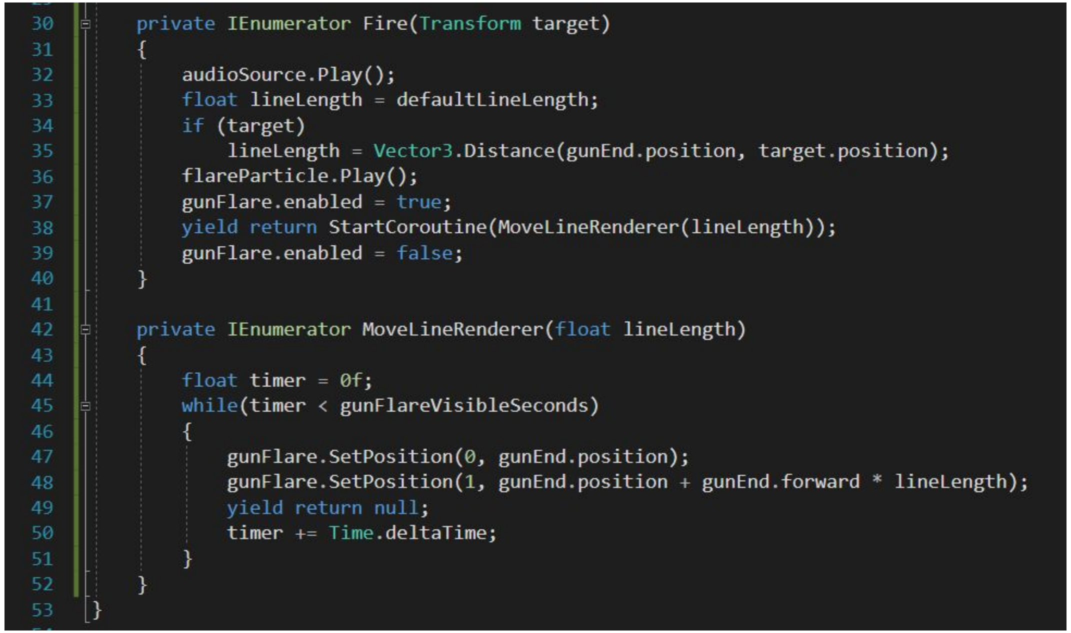
Setup ShooterWeapon

- Drag ShooterWeapon.prefab into Scene from Prefabs Folder
- Modify MainCamera Y to 1.5f
- Drag MainCamera onto UIMovement.CameraTranform Field
- Select GunRay Material onto LineRenderer Materials Field
- Open ShootingGunController than Edit it

Setup Weapon Fire Behavior



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using VRStandardAssets.Utils;
//using UnityEngine.VR;
public class ShootingGunController : MonoBehaviour
{
    public AudioSource audioSource;
    public VRInput vrInput;
    public Transform gunEnd;
    public ParticleSystem flareParticle;
    public LineRenderer gunFlare;
    public float defaultLineLength = 70f;
    public float gunFlareVisibleSeconds = 0.07f;
    private void OnEnable()
    {
        vrInput.OnDown += HandleDown;
    }

    private void OnDisable()
    {
        vrInput.OnDown -= HandleDown;
    }

    private void HandleDown()
    {
        StartCoroutine(Fire(null));
    }
```

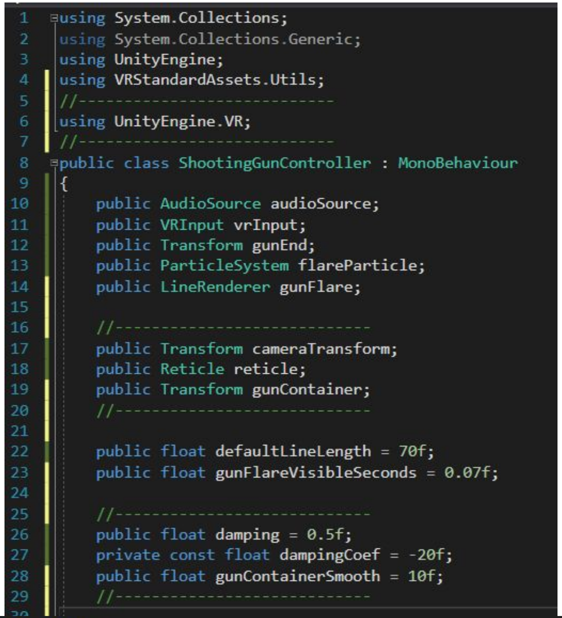


```
    private IEnumerator Fire(Transform target)
    {
        audioSource.Play();
        float lineLength = defaultLineLength;
        if (target)
            lineLength = Vector3.Distance(gunEnd.position, target.position);
        flareParticle.Play();
        gunFlare.enabled = true;
        yield return StartCoroutine(MoveLineRenderer(lineLength));
        gunFlare.enabled = false;
    }

    private IEnumerator MoveLineRenderer(float lineLength)
    {
        float timer = 0f;
        while(timer < gunFlareVisibleSeconds)
        {
            gunFlare.SetPosition(0, gunEnd.position);
            gunFlare.SetPosition(1, gunEnd.position + gunEnd.forward * lineLength);
            yield return null;
            timer += Time.deltaTime;
        }
    }
}
```

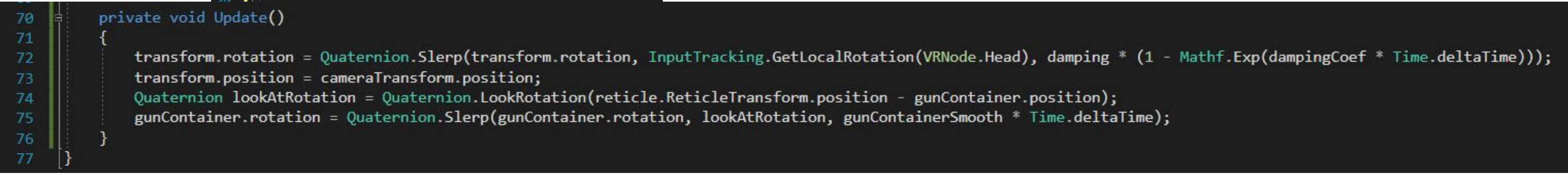
Setup ShooterWeapon

- Assign AudioSource from Self
- Assign VR Input from MainCamera
- Assign GunEnd from Child Transform
- Assign FlareParticle from Child Transform
- Assign GunFlare from Self
- Test it with Fire1 hotkey from Input Manager
- Apply Prefab
- Save Scene

Setup Weapon Follow Gaze Position



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using VRStandardAssets.Utils;
//---------------------------
using UnityEngine.VR;
//---------------------------
public class ShootingGunController : MonoBehaviour
{
    public AudioSource audioSource;
    public VRInput vrInput;
    public Transform gunEnd;
    public ParticleSystem flareParticle;
    public LineRenderer gunFlare;

    //---------------------------
    public Transform cameraTransform;
    public Reticle reticle;
    public Transform gunContainer;
    //---------------------------

    public float defaultLineLength = 70f;
    public float gunFlareVisibleSeconds = 0.07f;

    //---------------------------
    public float damping = 0.5f;
    private const float dampingCoef = -20f;
    public float gunContainerSmooth = 10f;
    //---------------------------
```



```
    private void Update()
    {
        transform.rotation = Quaternion.Slerp(transform.rotation, InputTracking.GetLocalRotation(VRNode.Head), damping * (1 - Mathf.Exp(dampingCoef * Time.deltaTime)));
        transform.position = cameraTransform.position;
        Quaternion lookAtRotation = Quaternion.LookRotation(reticle.ReticleTransform.position - gunContainer.position);
        gunContainer.rotation = Quaternion.Slerp(gunContainer.rotation, lookAtRotation, gunContainerSmooth * Time.deltaTime);
    }
}
```

Setup ShooterWeapon

- Assign CameraTransform from MainCamera
- Assign Reticle from MainCamera
- Assign GunContainer from Child Transform
- Assign FlareParticle from Child Transform name ShooterFPSWeapon
- Test it with Fire1 hotkey and ATL + Mouse
- Apply Prefab
- Save Scene

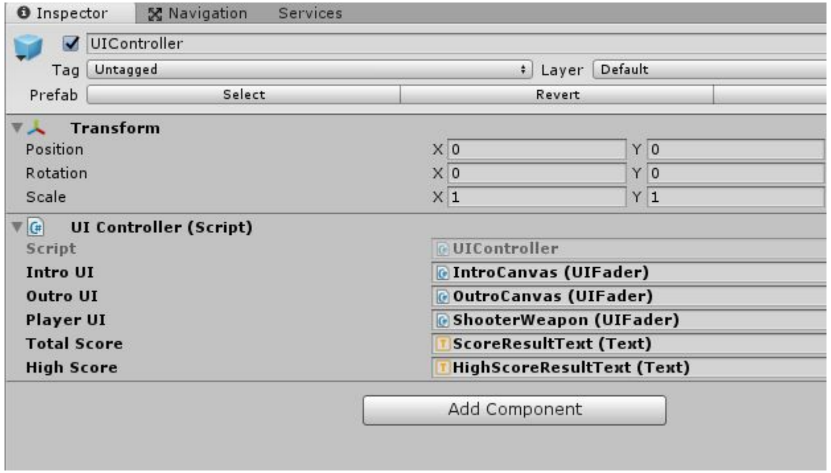
Setup GUI

- Drag GUI into Scene from Prefabs Folder
- SelectionSlider -> Assign MenuSelect into OnFilledClip
- SelectionSlider -> Assign MainCamera into VR Input
- SelectionSlider -> Assign MainCamera into SelectionRadial
- Save Scene

Setup System

- Drag System into Scene from Prefabs Folder
- Open UIController Edit it

Setup UIController



Inspector | Navigation | Services

UIController

Tag Untagged | Layer Default

Prefab | Select | Revert

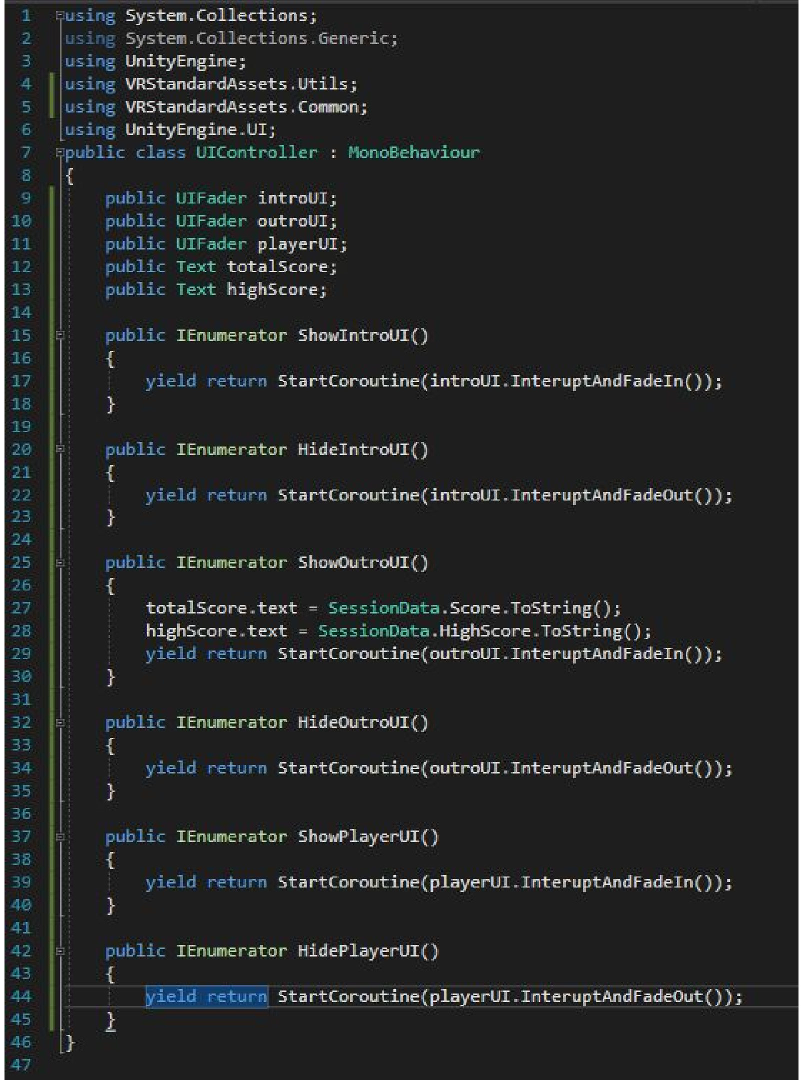
Transform

Position	X 0	Y 0
Rotation	X 0	Y 0
Scale	X 1	Y 1

UI Controller (Script)

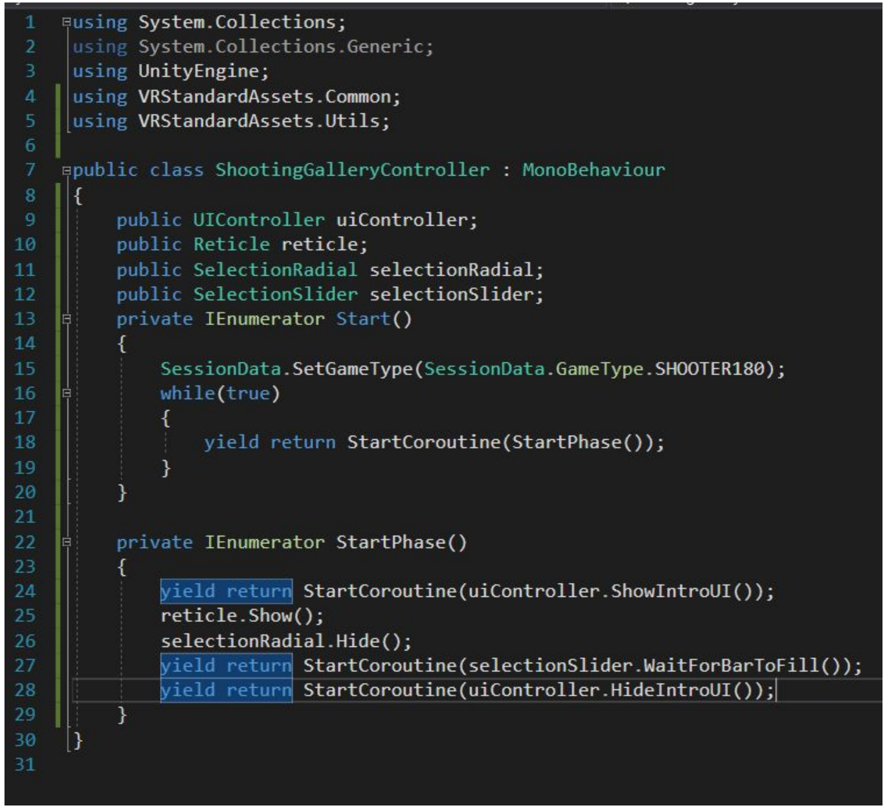
Script	UIController
Intro UI	IntroCanvas (UIFader)
Outro UI	OutroCanvas (UIFader)
Player UI	ShooterWeapon (UIFader)
Total Score	ScoreResultText (Text)
High Score	HighScoreResultText (Text)

Add Component

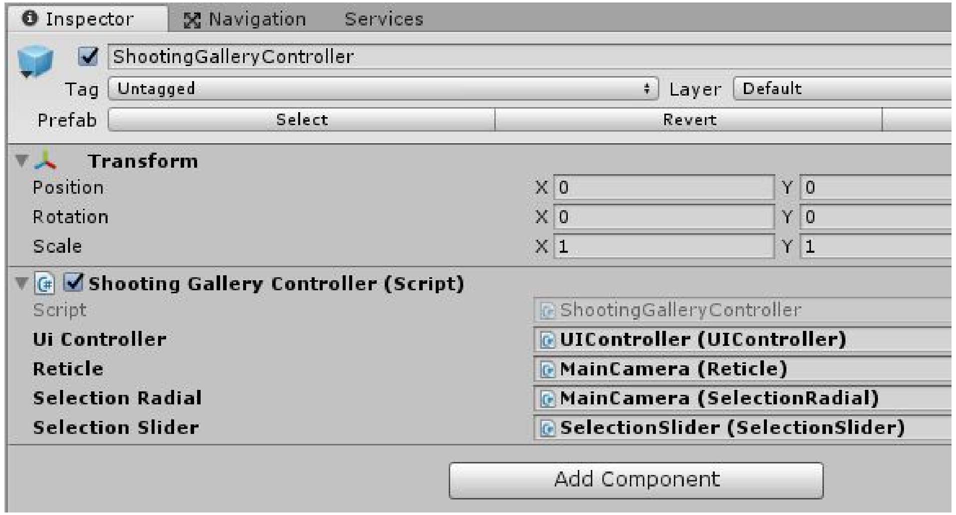


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using VRStandardAssets.Utils;
using VRStandardAssets.Common;
using UnityEngine.UI;
public class UIController : MonoBehaviour
{
    public UIFader introUI;
    public UIFader outroUI;
    public UIFader playerUI;
    public Text totalScore;
    public Text highScore;

    public IEnumerator ShowIntroUI()
    {
        yield return StartCoroutine(introUI.InteruptAndFadeIn());
    }

    public IEnumerator HideIntroUI()
    {
        yield return StartCoroutine(introUI.InteruptAndFadeOut());
    }

    public IEnumerator ShowOutroUI()
    {
        totalScore.text = SessionData.Score.ToString();
        highScore.text = SessionData.HighScore.ToString();
        yield return StartCoroutine(outroUI.InteruptAndFadeIn());
    }

    public IEnumerator HideOutroUI()
    {
        yield return StartCoroutine(outroUI.InteruptAndFadeOut());
    }

    public IEnumerator ShowPlayerUI()
    {
        yield return StartCoroutine(playerUI.InteruptAndFadeIn());
    }

    public IEnumerator HidePlayerUI()
    {
        yield return StartCoroutine(playerUI.InteruptAndFadeOut());
    }
}
```

Setup ShootingGalleryController - StartPhase



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using VRStandardAssets.Common;
using VRStandardAssets.Utils;

public class ShootingGalleryController : MonoBehaviour
{
    public UIController uiController;
    public Reticle reticle;
    public SelectionRadial selectionRadial;
    public SelectionSlider selectionSlider;
    private IEnumerator Start()
    {
        SessionData.SetGameType(SessionData.GameType.SHOOTER180);
        while(true)
        {
            yield return StartCoroutine(StartPhase());
        }
    }

    private IEnumerator StartPhase()
    {
        yield return StartCoroutine(uiController.ShowIntroUI());
        reticle.Show();
        selectionRadial.Hide();
        yield return StartCoroutine(selectionSlider.WaitForBarToFill());
        yield return StartCoroutine(uiController.HideIntroUI());
    }
}
```



Inspector | Navigation | Services

ShootingGalleryController

Tag: Untagged | Layer: Default

Prefab: Select | Revert

Transform

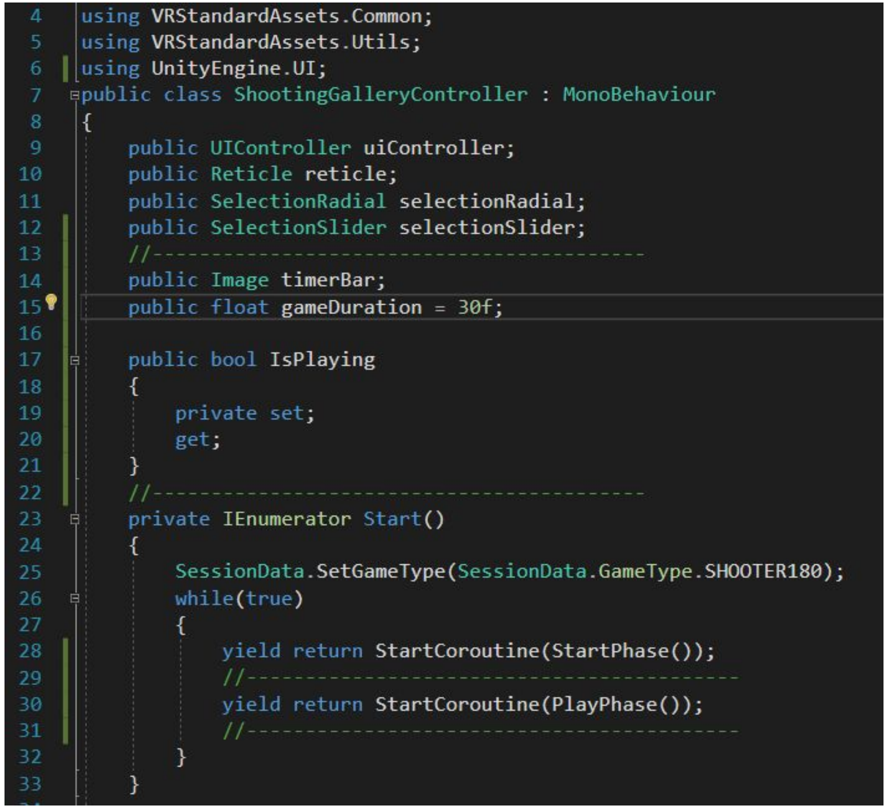
Position	X 0	Y 0
Rotation	X 0	Y 0
Scale	X 1	Y 1

Shooting Gallery Controller (Script)

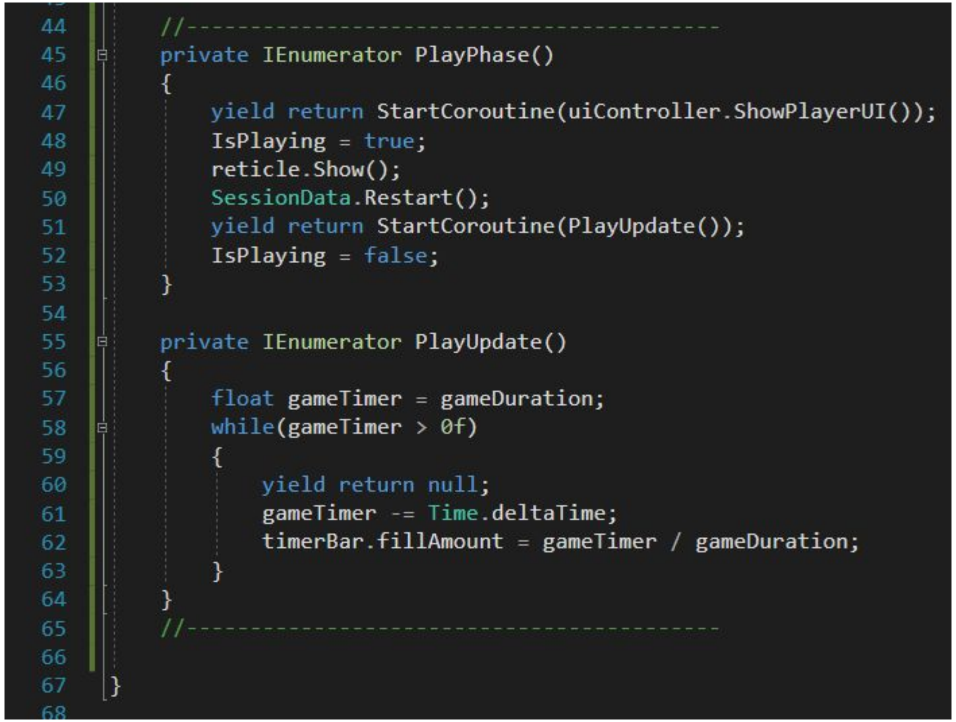
Script	ShootingGalleryController
Ui Controller	**UIController (UIController)**
Reticle	**MainCamera (Reticle)**
Selection Radial	**MainCamera (SelectionRadial)**
Selection Slider	**SelectionSlider (SelectionSlider)**

Add Component

Setup ShootingGalleryController - PlayPhase



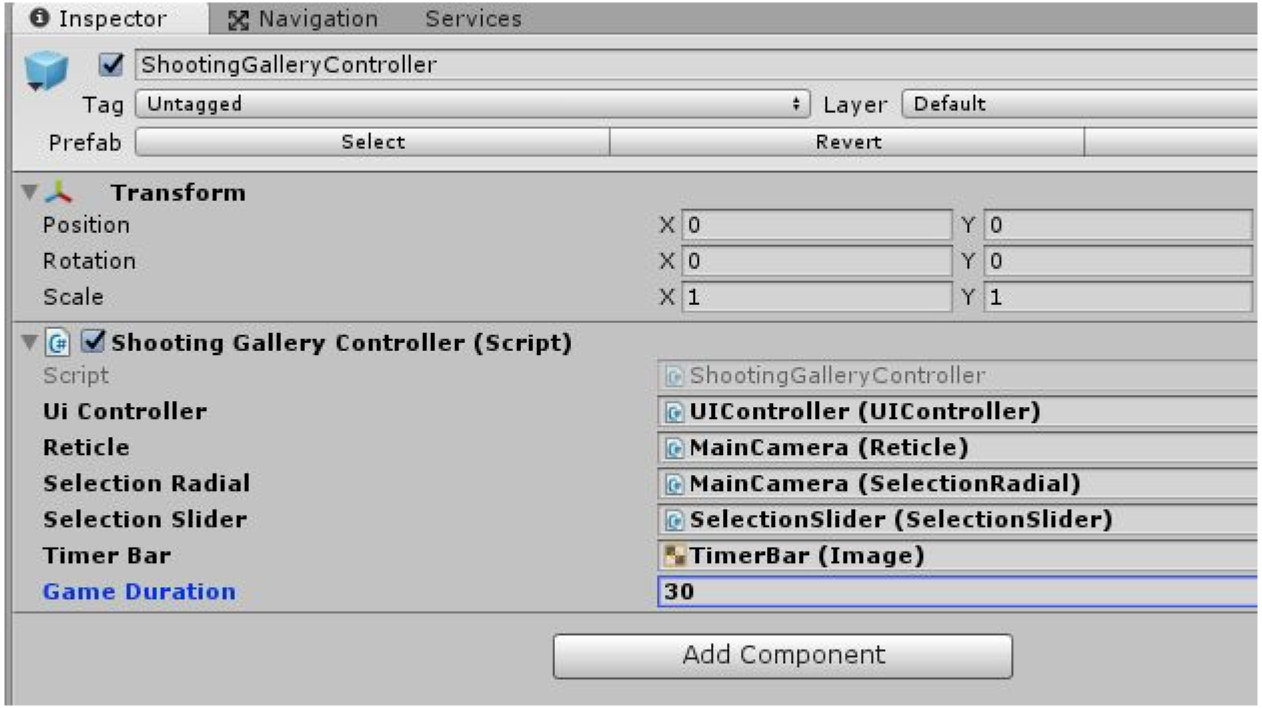
```
using VRStandardAssets.Common;
using VRStandardAssets.Utils;
using UnityEngine.UI;
public class ShootingGalleryController : MonoBehaviour
{
    public UIController uiController;
    public Reticle reticle;
    public SelectionRadial selectionRadial;
    public SelectionSlider selectionSlider;
    //---------------------------------------
    public Image timerBar;
    public float gameDuration = 30f;

    public bool IsPlaying
    {
        private set;
        get;
    }
    //---------------------------------------
    private IEnumerator Start()
    {
        SessionData.SetGameType(SessionData.GameType.SHOOTER180);
        while(true)
        {
            yield return StartCoroutine(StartPhase());
            //---------------------------------------
            yield return StartCoroutine(PlayPhase());
            //---------------------------------------
        }
    }
```



```
    //---------------------------------------
    private IEnumerator PlayPhase()
    {
        yield return StartCoroutine(uiController.ShowPlayerUI());
        IsPlaying = true;
        reticle.Show();
        SessionData.Restart();
        yield return StartCoroutine(PlayUpdate());
        IsPlaying = false;
    }

    private IEnumerator PlayUpdate()
    {
        float gameTimer = gameDuration;
        while(gameTimer > 0f)
        {
            yield return null;
            gameTimer -= Time.deltaTime;
            timerBar.fillAmount = gameTimer / gameDuration;
        }
    }
    //---------------------------------------

}
```

Setup TimeBar Image

- Assign TimerBar from ShooterWeapon -> PlayerGUI -> TimerBar
- Test it
- Save Scene



ShootingGalleryController
Tag Untagged Layer Default
Prefab Select Revert

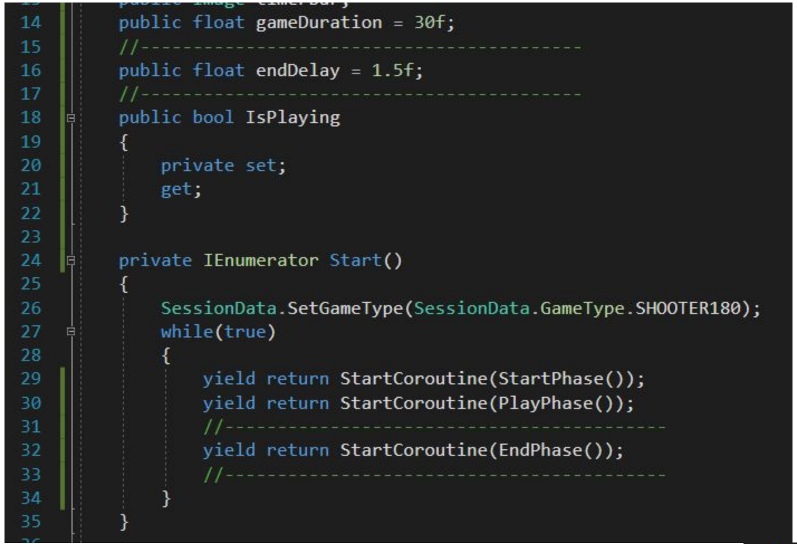
Transform

Position	X 0	Y 0
Rotation	X 0	Y 0
Scale	X 1	Y 1

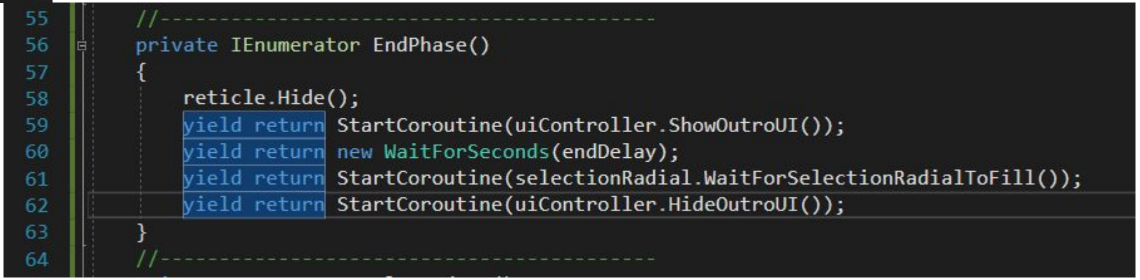
Shooting Gallery Controller (Script)

Script	ShootingGalleryController
Ui Controller	UIController (UIController)
Reticle	MainCamera (Reticle)
Selection Radial	MainCamera (SelectionRadial)
Selection Slider	SelectionSlider (SelectionSlider)
Timer Bar	TimerBar (Image)
Game Duration	30

Setup ShootingGalleryController - EndPhase

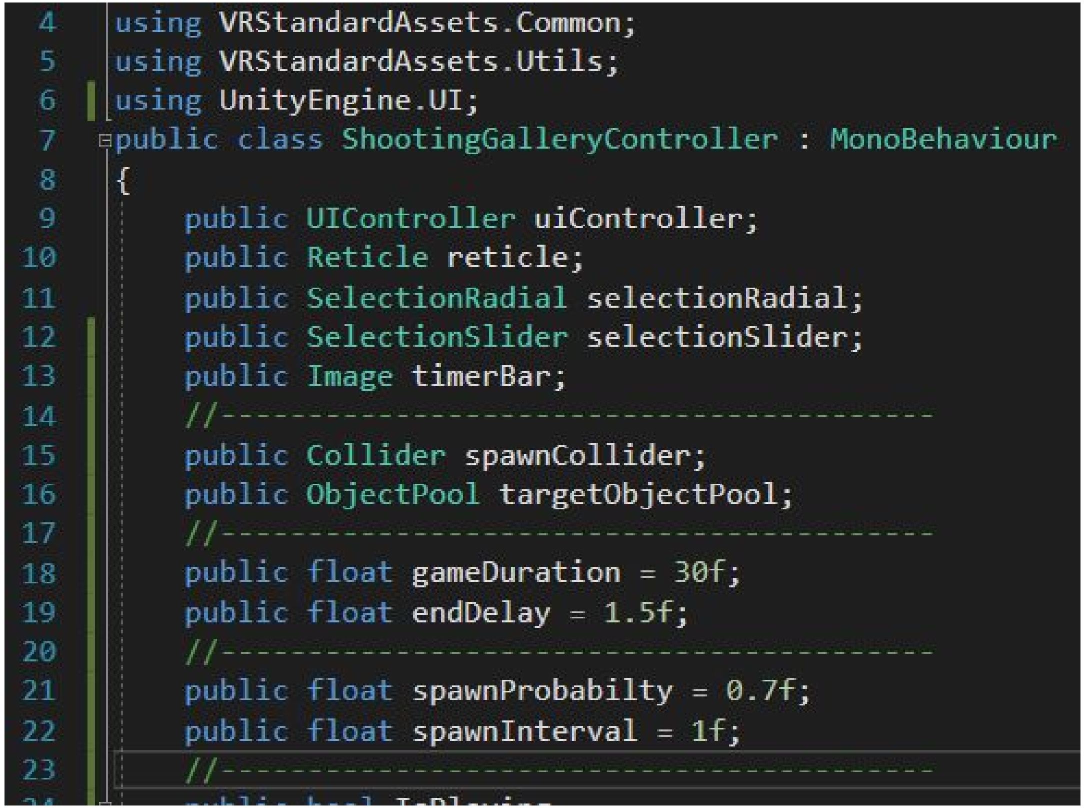


```
    public float gameDuration = 30f;
    //----------------------------------------
    public float endDelay = 1.5f;
    //----------------------------------------
    public bool IsPlaying
    {
        private set;
        get;
    }

    private IEnumerator Start()
    {
        SessionData.SetGameType(SessionData.GameType.SHOOTER180);
        while(true)
        {
            yield return StartCoroutine(StartPhase());
            yield return StartCoroutine(PlayPhase());
            //----------------------------------------
            yield return StartCoroutine(EndPhase());
            //----------------------------------------
        }
    }
```

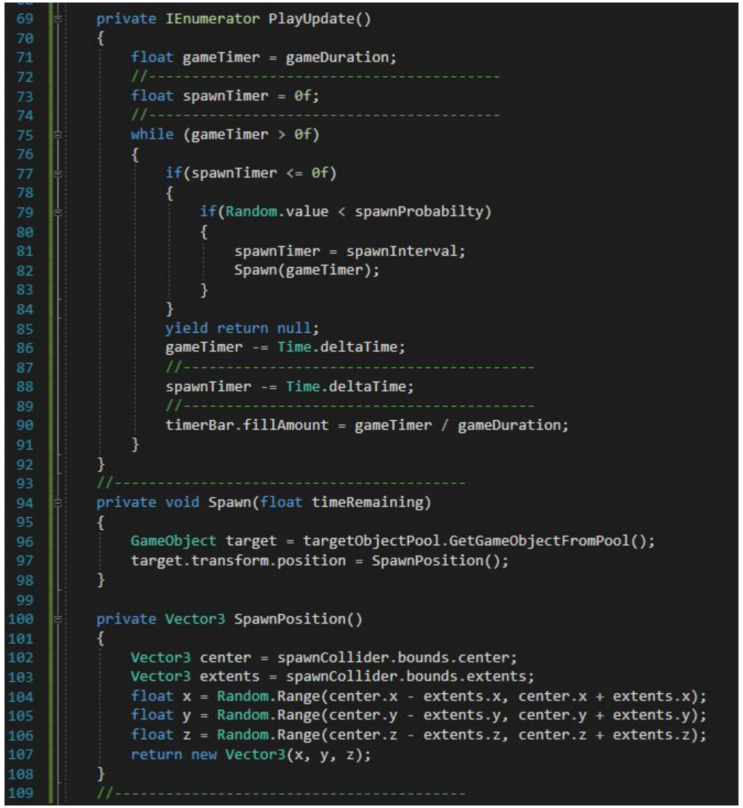


```
    //----------------------------------------
    private IEnumerator EndPhase()
    {
        reticle.Hide();
        yield return StartCoroutine(uiController.ShowOutroUI());
        yield return new WaitForSeconds(endDelay);
        yield return StartCoroutine(selectionRadial.WaitForSelectionRadialToFill());
        yield return StartCoroutine(uiController.HideOutroUI());
    }
    //----------------------------------------
```

Setup ShootingGalleryController - Spawn Behavior



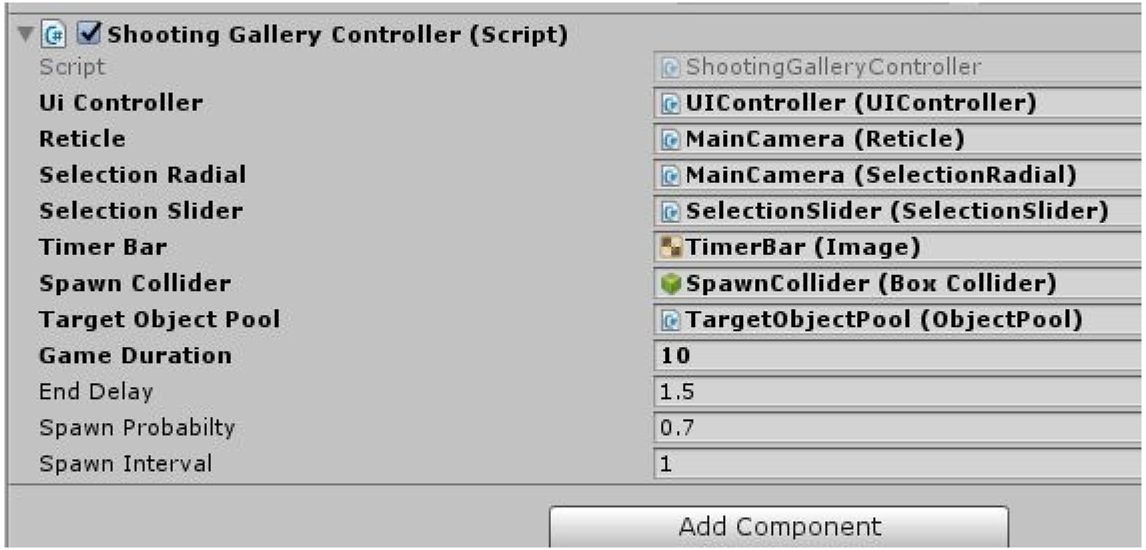
```csharp
using VRStandardAssets.Common;
using VRStandardAssets.Utils;
using UnityEngine.UI;
public class ShootingGalleryController : MonoBehaviour
{
    public UIController uiController;
    public Reticle reticle;
    public SelectionRadial selectionRadial;
    public SelectionSlider selectionSlider;
    public Image timerBar;
    //-----------------------------------------
    public Collider spawnCollider;
    public ObjectPool targetObjectPool;
    //-----------------------------------------
    public float gameDuration = 30f;
    public float endDelay = 1.5f;
    //-----------------------------------------
    public float spawnProbabilty = 0.7f;
    public float spawnInterval = 1f;
    //-----------------------------------------
```



```csharp
    private IEnumerator PlayUpdate()
    {
        float gameTimer = gameDuration;
        //-----------------------------------------
        float spawnTimer = 0f;
        //-----------------------------------------
        while (gameTimer > 0f)
        {
            if(spawnTimer <= 0f)
            {
                if(Random.value < spawnProbabilty)
                {
                    spawnTimer = spawnInterval;
                    Spawn(gameTimer);
                }
            }
            yield return null;
            gameTimer -= Time.deltaTime;
            //-----------------------------------------
            spawnTimer -= Time.deltaTime;
            //-----------------------------------------
            timerBar.fillAmount = gameTimer / gameDuration;
        }
    }
    //-----------------------------------------
    private void Spawn(float timeRemaining)
    {
        GameObject target = targetObjectPool.GetGameObjectFromPool();
        target.transform.position = SpawnPosition();
    }

    private Vector3 SpawnPosition()
    {
        Vector3 center = spawnCollider.bounds.center;
        Vector3 extents = spawnCollider.bounds.extents;
        float x = Random.Range(center.x - extents.x, center.x + extents.x);
        float y = Random.Range(center.y - extents.y, center.y + extents.y);
        float z = Random.Range(center.z - extents.z, center.z + extents.z);
        return new Vector3(x, y, z);
    }
    //-----------------------------------------
```

Setup Spawn Field

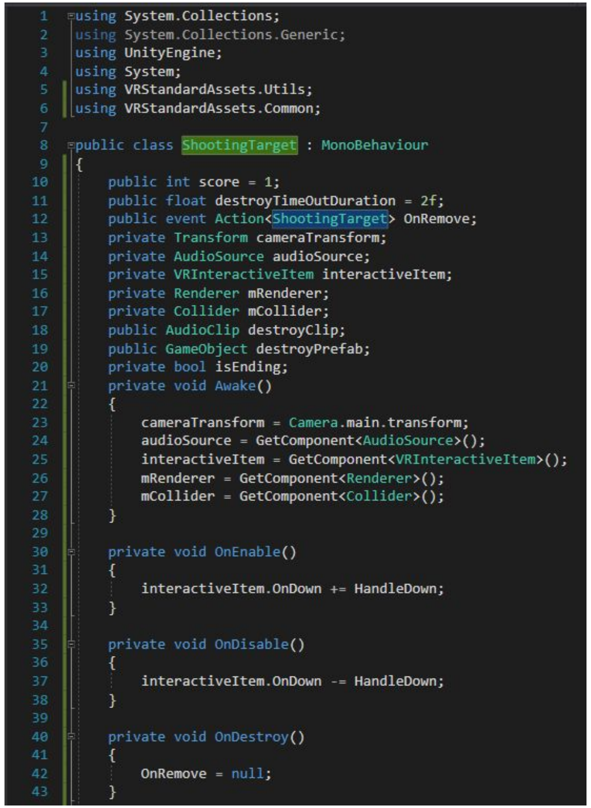
- Assign SpawnCollider from System -> SpawnCollider
- Assign TargetObjectPool from System -> TargetObjectPool
- Test it
- Save Scene



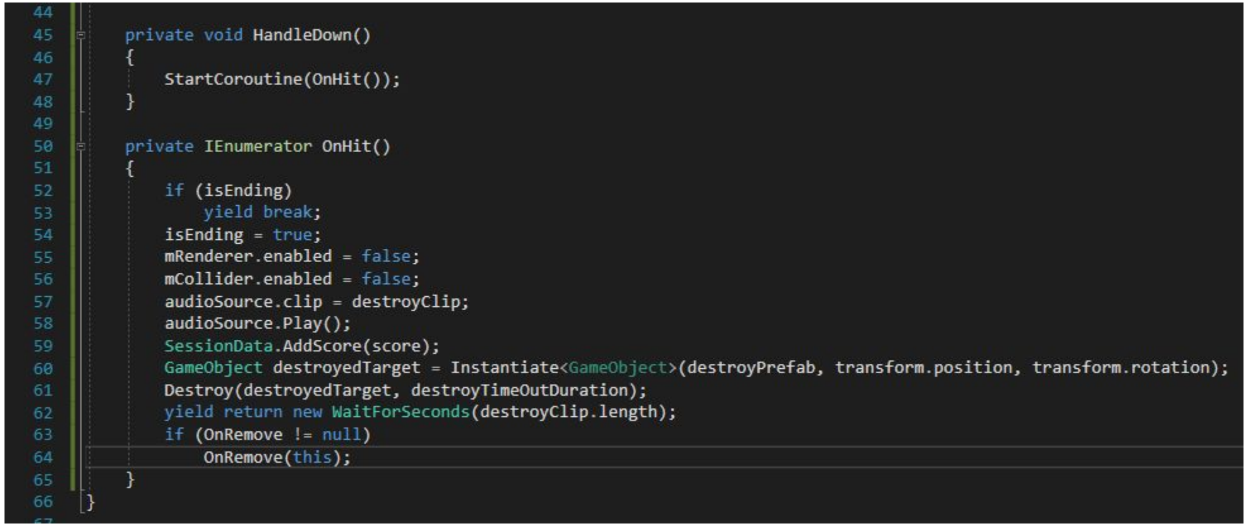
Shooting Gallery Controller (Script)

Script	ShootingGalleryController
Ui Controller	UIController (UIController)
Reticle	MainCamera (Reticle)
Selection Radial	MainCamera (SelectionRadial)
Selection Slider	SelectionSlider (SelectionSlider)
Timer Bar	TimerBar (Image)
Spawn Collider	SpawnCollider (Box Collider)
Target Object Pool	TargetObjectPool (ObjectPool)
Game Duration	10
End Delay	1.5
Spawn Probabilty	0.7
Spawn Interval	1

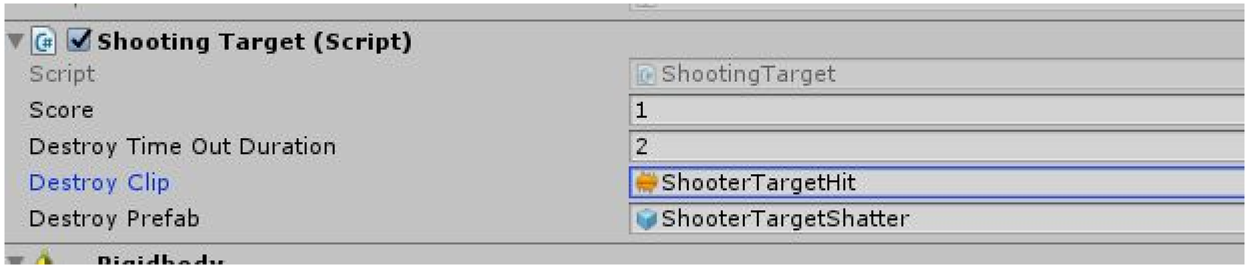
Setup ShootingTarget



```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;
using VRStandardAssets.Utils;
using VRStandardAssets.Common;

public class ShootingTarget : MonoBehaviour
{
    public int score = 1;
    public float destroyTimeOutDuration = 2f;
    public event Action<ShootingTarget> OnRemove;
    private Transform cameraTransform;
    private AudioSource audioSource;
    private VRInteractiveItem interactiveItem;
    private Renderer mRenderer;
    private Collider mCollider;
    public AudioClip destroyClip;
    public GameObject destroyPrefab;
    private bool isEnding;
    private void Awake()
    {
        cameraTransform = Camera.main.transform;
        audioSource = GetComponent<AudioSource>();
        interactiveItem = GetComponent<VRInteractiveItem>();
        mRenderer = GetComponent<Renderer>();
        mCollider = GetComponent<Collider>();
    }

    private void OnEnable()
    {
        interactiveItem.OnDown += HandleDown;
    }

    private void OnDisable()
    {
        interactiveItem.OnDown -= HandleDown;
    }

    private void OnDestroy()
    {
        OnRemove = null;
    }
```



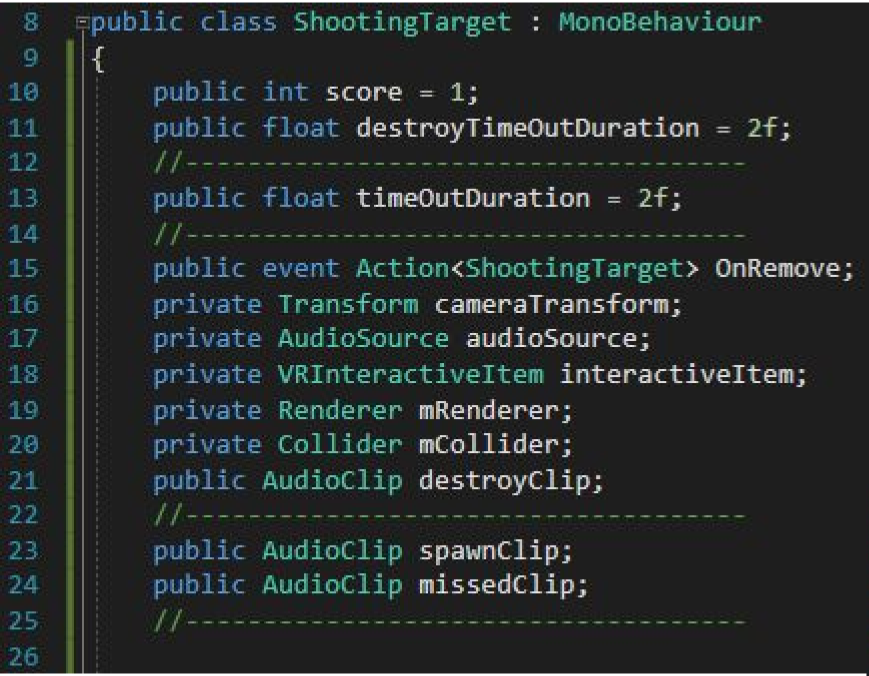
```csharp

    private void HandleDown()
    {
        StartCoroutine(OnHit());
    }

    private IEnumerator OnHit()
    {
        if (isEnding)
            yield break;
        isEnding = true;
        mRenderer.enabled = false;
        mCollider.enabled = false;
        audioSource.clip = destroyClip;
        audioSource.Play();
        SessionData.AddScore(score);
        GameObject destroyedTarget = Instantiate<GameObject>(destroyPrefab, transform.position, transform.rotation);
        Destroy(destroyedTarget, destroyTimeOutDuration);
        yield return new WaitForSeconds(destroyClip.length);
        if (OnRemove != null)
            OnRemove(this);
    }
}
```



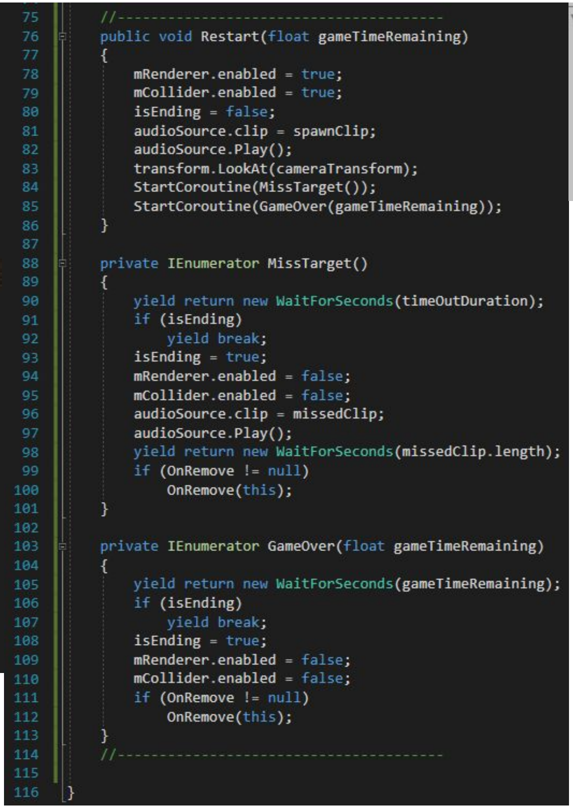
Shooting Target (Script)

Script	ShootingTarget
Score	1
Destroy Time Out Duration	2
Destroy Clip	ShooterTargetHit
Destroy Prefab	ShooterTargetShatter

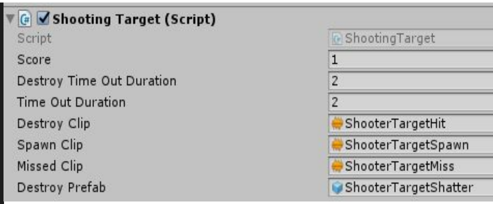
Setup ShootingTarget - LifeCycle



```
public class ShootingTarget : MonoBehaviour
{
    public int score = 1;
    public float destroyTimeOutDuration = 2f;
    //--------------------------------------
    public float timeOutDuration = 2f;
    //--------------------------------------
    public event Action<ShootingTarget> OnRemove;
    private Transform cameraTransform;
    private AudioSource audioSource;
    private VRInteractiveItem interactiveItem;
    private Renderer mRenderer;
    private Collider mCollider;
    public AudioClip destroyClip;
    //--------------------------------------
    public AudioClip spawnClip;
    public AudioClip missedClip;
    //--------------------------------------

```



```
    //--------------------------------------
    public void Restart(float gameTimeRemaining)
    {
        mRenderer.enabled = true;
        mCollider.enabled = true;
        isEnding = false;
        audioSource.clip = spawnClip;
        audioSource.Play();
        transform.LookAt(cameraTransform);
        StartCoroutine(MissTarget());
        StartCoroutine(GameOver(gameTimeRemaining));
    }

    private IEnumerator MissTarget()
    {
        yield return new WaitForSeconds(timeOutDuration);
        if (isEnding)
            yield break;
        isEnding = true;
        mRenderer.enabled = false;
        mCollider.enabled = false;
        audioSource.clip = missedClip;
        audioSource.Play();
        yield return new WaitForSeconds(missedClip.length);
        if (OnRemove != null)
            OnRemove(this);
    }

    private IEnumerator GameOver(float gameTimeRemaining)
    {
        yield return new WaitForSeconds(gameTimeRemaining);
        if (isEnding)
            yield break;
        isEnding = true;
        mRenderer.enabled = false;
        mCollider.enabled = false;
        if (OnRemove != null)
            OnRemove(this);
    }
    //--------------------------------------

}
```

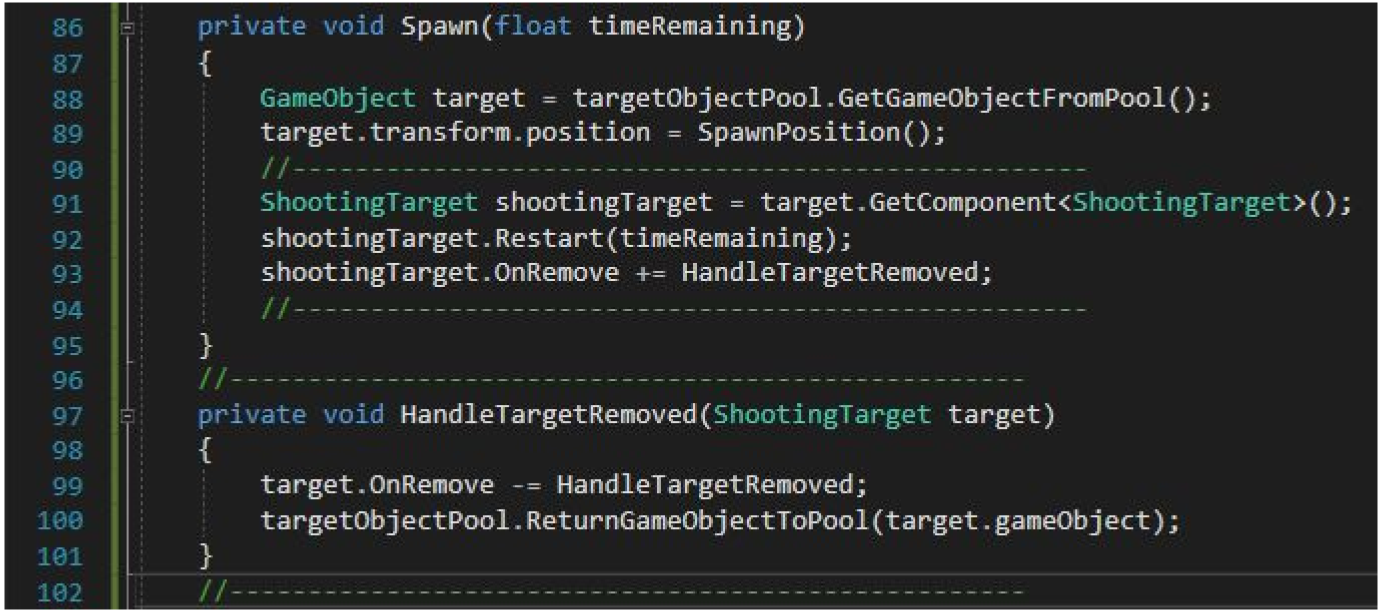


Shooting Target (Script)

Script	ShootingTarget
Score	1
Destroy Time Out Duration	2
Time Out Duration	2
Destroy Clip	ShooterTargetHit
Spawn Clip	ShooterTargetSpawn
Missed Clip	ShooterTargetMiss
Destroy Prefab	ShooterTargetShatter

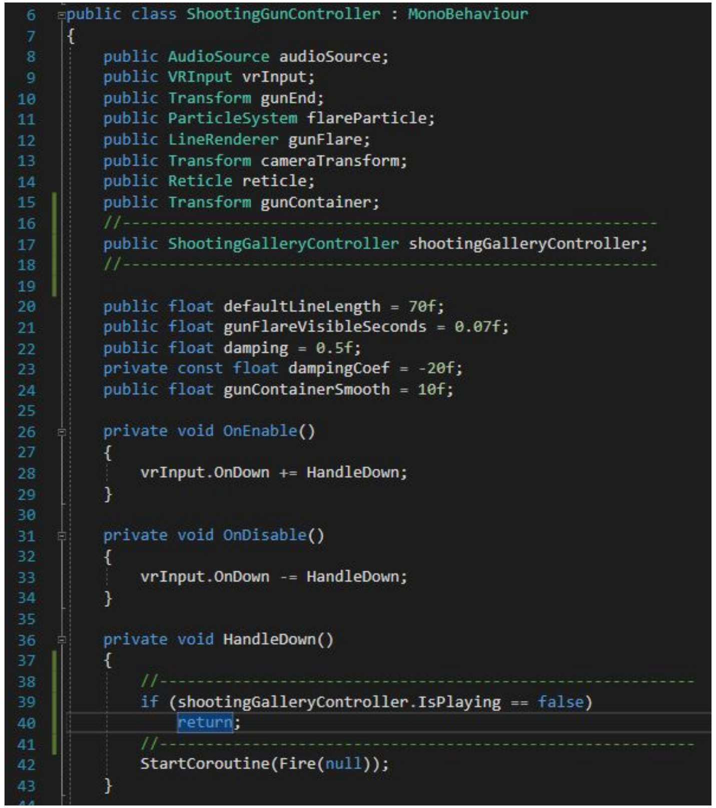
Setup ShootingGalleryController - ShootingTarget

- Test it
- Save Scene

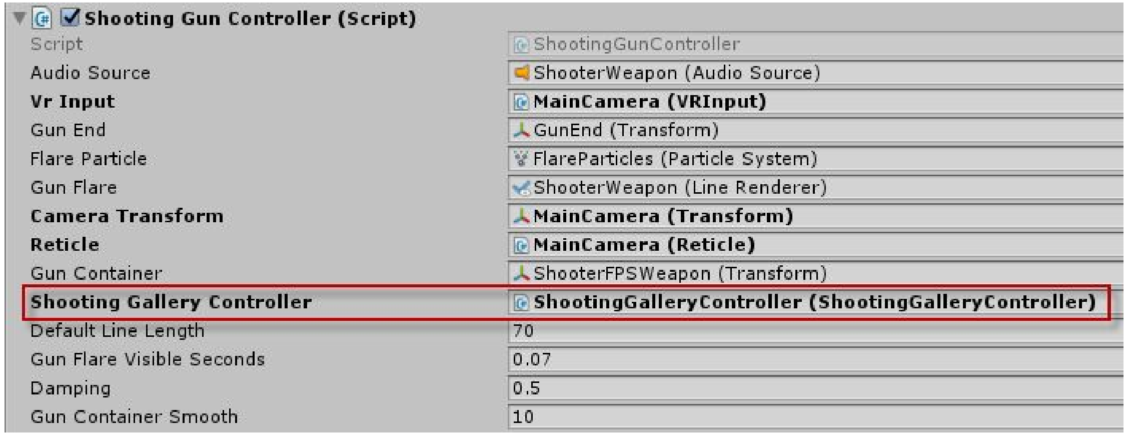


```
    private void Spawn(float timeRemaining)
    {
        GameObject target = targetObjectPool.GetGameObjectFromPool();
        target.transform.position = SpawnPosition();
        //---------------------------------------------------
        ShootingTarget shootingTarget = target.GetComponent<ShootingTarget>();
        shootingTarget.Restart(timeRemaining);
        shootingTarget.OnRemove += HandleTargetRemoved;
        //---------------------------------------------------
    }
    //---------------------------------------------------
    private void HandleTargetRemoved(ShootingTarget target)
    {
        target.OnRemove -= HandleTargetRemoved;
        targetObjectPool.ReturnGameObjectToPool(target.gameObject);
    }
    //---------------------------------------------------
```

Complete ShootingGunController



```
public class ShootingGunController : MonoBehaviour
{
    public AudioSource audioSource;
    public VRInput vrInput;
    public Transform gunEnd;
    public ParticleSystem flareParticle;
    public LineRenderer gunFlare;
    public Transform cameraTransform;
    public Reticle reticle;
    public Transform gunContainer;
    //-----------------------------------------------------------------
    public ShootingGalleryController shootingGalleryController;
    //-----------------------------------------------------------------

    public float defaultLineLength = 70f;
    public float gunFlareVisibleSeconds = 0.07f;
    public float damping = 0.5f;
    private const float dampingCoef = -20f;
    public float gunContainerSmooth = 10f;

    private void OnEnable()
    {
        vrInput.OnDown += HandleDown;
    }

    private void OnDisable()
    {
        vrInput.OnDown -= HandleDown;
    }

    private void HandleDown()
    {
        //-------------------------------------------------------------
        if (shootingGalleryController.IsPlaying == false)
            return;
        //-------------------------------------------------------------
        StartCoroutine(Fire(null));
    }
```



Shooting Gun Controller (Script)

Script	ShootingGunController
Audio Source	ShooterWeapon (Audio Source)
Vr Input	**MainCamera (VRInput)**
Gun End	GunEnd (Transform)
Flare Particle	FlareParticles (Particle System)
Gun Flare	ShooterWeapon (Line Renderer)
Camera Transform	**MainCamera (Transform)**
Reticle	**MainCamera (Reticle)**
Gun Container	ShooterFPSWeapon (Transform)
Shooting Gallery Controller	**ShootingGalleryController (ShootingGalleryController)**
Default Line Length	70
Gun Flare Visible Seconds	0.07
Damping	0.5
Gun Container Smooth	10

Setup BGM

- Drag Audio Prefab into Scene from Prefabs Folder
- Save Scene
- Have Fun!