

COMSATS UNIVERSITY ISLAMABAD, ABBOTTABAD

Game development Lab Assignment 2

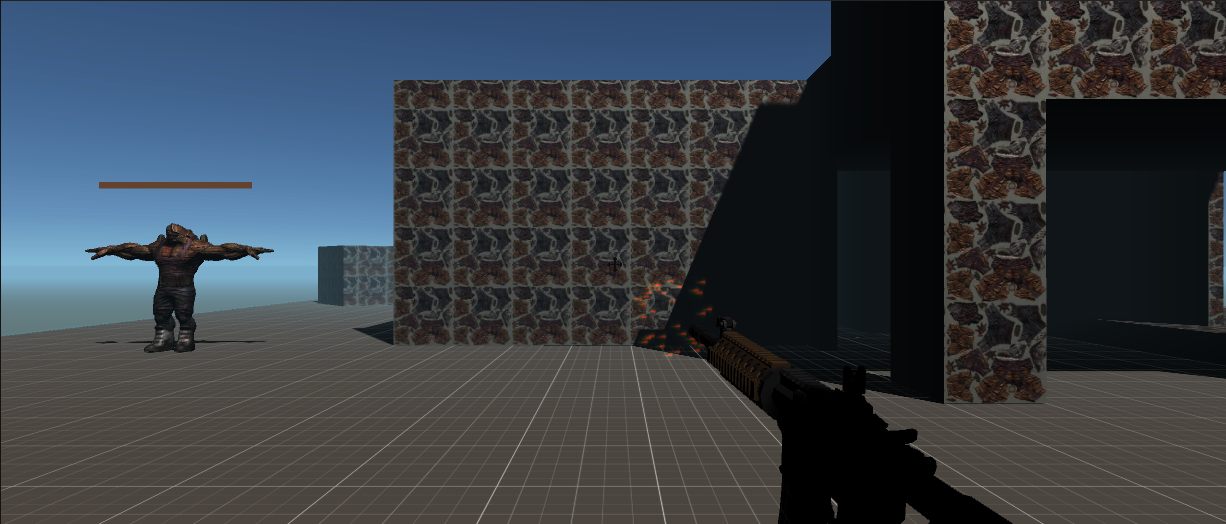
FPS game

***Submitted by:***

Laiba binte tahir FA21-BSE-019

***Submitted to:***

Sir Muhammad Ibtisam Gul



A video game screen with a person holding an object

Description automatically generated

A video game screen with a person lying on the ground

Description automatically generated

**1. Controls Script:**

* Manages player inputs using Unity's InputAction system for walking and an action (kill).
* Toggles an enemy's walking animation using the enemyAnimator.
* Enables or disables animations based on user input for walking or other actions.

using UnityEngine;

using UnityEngine.InputSystem;

public class Contols : MonoBehaviour

{

    [SerializeField]

    InputAction walk, kill;

    [SerializeField]

    Animator enemyAnimator;

    private void OnEnable()

    {

        walk.Enable();

        kill.Enable();

    }

    private void Update()

    {

        OnWalk();

        if(kill.IsPressed())

        {

            enemyAnimator.enabled = false;

        }

    }

    private void OnWalk()

    {

        if (walk.IsPressed() && enemyAnimator.enabled)

        {

            if (enemyAnimator.GetBool("isWalking"))

            {

                enemyAnimator.SetBool("isWalking", false);

            }

            else

            {

                enemyAnimator.SetBool("isWalking", true);

            }

        }

    }

}

**2. Navigate Script:**

* Uses a NavMeshAgent to move a character towards a destination while detecting a player using raycasts.
* Animates states (walking, running, attacking) based on distance to the target and plays corresponding sounds.
* Resets to the original position when the player is not detected, stopping animations and sounds.

using System;

using UnityEngine;

using UnityEngine.AI;

public class Navigate : MonoBehaviour

{

    [SerializeField]

    NavMeshAgent agent;

    [SerializeField]

    GameObject destinationObject;

    [SerializeField]

    Animator animator;

    [SerializeField]

    float detectDist, runDist, attackDist;

    [SerializeField]

    float distance;

    Vector3 originalPos;

    [SerializeField]

    Transform rayPoint;

    bool isDetected;

    [SerializeField]

    LayerMask playerLayer;

    // Audio variables

    [SerializeField]

    AudioSource audioSource;

    [SerializeField]

    AudioClip moveSound;

    [SerializeField]

    AudioClip attackSound;

    [SerializeField]

    AudioClip killSound;

    void Start()

    {

        agent = GetComponent<NavMeshAgent>();

        originalPos = transform.position;

    }

    void Update()

    {

        if (Detect())

        {

            isDetected = true;

        }

        if (isDetected && CalculateDistance(transform.position, destinationObject.transform.position) < detectDist)

        {

            agent.SetDestination(destinationObject.transform.position);

            animator.SetBool("isWalking", true);

            if (!audioSource.isPlaying && !animator.GetBool("isRunning") && !animator.GetBool("isAttacking"))

            {

                audioSource.clip = moveSound;

                audioSource.Play();

            }

            if (CalculateDistance(transform.position, agent.destination) < runDist)

            {

                animator.SetBool("isRunning", true);

                audioSource.clip = moveSound;

                if (!audioSource.isPlaying)

                {

                    audioSource.Play();

                }

                if (CalculateDistance(transform.position, agent.destination) < attackDist)

                {

                    agent.ResetPath();

                    animator.SetBool("isRunning", false);

                    animator.SetBool("isWalking", false);

                    animator.SetBool("isAttacking", true);

                    if (!audioSource.isPlaying)

                    {

                        audioSource.clip = attackSound;

                        audioSource.Play();

                    }

                }

                else

                {

                    animator.SetBool("isAttacking", false);

                    animator.SetBool("isWalking", true);

                }

            }

            else

            {

                animator.SetBool("isRunning", false);

            }

        }

        else

        {

            isDetected = false;

            agent.ResetPath();

            agent.SetDestination(originalPos);

            if (CalculateDistance(transform.position, originalPos) < 0.25)

            {

                animator.SetBool("isWalking", false);

                agent.ResetPath();

                audioSource.Stop();

            }

        }

    }

    private float CalculateDistance(Vector3 from, Vector3 to)

    {

        distance = Vector3.Distance(from, to);

        return distance;

    }

    private bool Detect()

    {

        Ray ray = new Ray(rayPoint.position, rayPoint.forward);

        Debug.DrawRay(rayPoint.position, rayPoint.forward \* detectDist, Color.red);

        bool isHit = Physics.Raycast(ray, detectDist, playerLayer);

        return isHit;

    }

    public void OnHit()

    {

        audioSource.clip = killSound;

        audioSource.Play();

    }

}

**3. GunFire Script:**

* Handles shooting mechanics by firing rays from the camera's center point.
* Triggers muzzle flash and sound effects when firing, and spawns blood splatter effects if the ray hits an enemy.
* Stops the enemy's animator and applies a force to the rigidbody of the hit object to simulate impact.

using UnityEngine;

using UnityEngine.InputSystem;

public class GunFire : MonoBehaviour

{

    Camera cam;

    [SerializeField]

    InputAction fire;

    [SerializeField]

    ParticleSystem muzzleFlash;

    [SerializeField]

    ParticleSystem bloodSplatterPrefab;

    [SerializeField]

    AudioSource gunAudioSource;

    [SerializeField]

    float muzzleFlashDuration = 0.1f;

    bool isFired;

    private void OnEnable()

    {

        fire.Enable();

    }

    private void OnDisable()

    {

        fire.Disable();

    }

    private void Start()

    {

        cam = Camera.main;

    }

    private void Update()

    {

        if (fire.WasPressedThisFrame())

        {

            isFired = true;

        }

    }

    private void FixedUpdate()

    {

        Vector2 screenPos = new Vector2(Screen.width / 2, Screen.height / 2);

        Ray ray = cam.ScreenPointToRay(screenPos);

        RaycastHit hit;

        bool isHit = Physics.Raycast(ray, out hit, Mathf.Infinity);

        if (isHit)

        {

            if (isFired)

            {

                // Trigger muzzle flash and gun sound

                PlayMuzzleFlash();

                PlayGunSound();

                // Show blood splatter at the hit location if enemy is hit

                if (hit.collider.gameObject.CompareTag("Enemy"))

                {

                    SpawnBloodSplatter(hit.point, hit.normal);

                    Rigidbody rb = hit.rigidbody;

                    if (rb != null)

                    {

                        rb.gameObject.GetComponentInParent<Animator>().enabled = false;

                        rb.AddForceAtPosition(-rb.transform.forward \* 400f, hit.point, ForceMode.Impulse);

                    }

                }

                isFired = false;

            }

        }

    }

    private void PlayMuzzleFlash()

    {

        if (muzzleFlash != null)

        {

            muzzleFlash.Play();

            StartCoroutine(StopMuzzleFlashAfterDelay());

        }

        else

        {

            Debug.LogWarning("Muzzle Flash particle sys is not assigned!");

        }

    }

    private void PlayGunSound()

    {

        if (gunAudioSource != null)

        {

            gunAudioSource.Play();

        }

        else

        {

            Debug.LogWarning("AudioSource is not assigned!");

        }

    }

    private void SpawnBloodSplatter(Vector3 position, Vector3 normal)

    {

        if (bloodSplatterPrefab != null)

        {

            // Instantiate the blood splatter effect at the hit location

            ParticleSystem bloodSplatter = Instantiate(bloodSplatterPrefab, position, Quaternion.LookRotation(normal));

            bloodSplatter.Play();

            // Destroy the blood splatter effect after it finishes

            Destroy(bloodSplatter.gameObject, bloodSplatter.main.duration + bloodSplatter.main.startLifetime.constantMax);

        }

        else

        {

            Debug.LogWarning("Blood Splatter prefab is not assigned!");

        }

    }

    private System.Collections.IEnumerator StopMuzzleFlashAfterDelay()

    {

        yield return new WaitForSeconds(muzzleFlashDuration);

        if (muzzleFlash != null)

        {

            muzzleFlash.Stop();

        }

    }

}

**4. PlayerSoundController Script:**

* Plays footstep sounds when the player is grounded and moving, selecting random sounds from an array.
* Plays a jump sound effect when the player jumps, providing an auditory cue for actions.

using UnityEngine;

public class PlayerSoundController : MonoBehaviour

{

    public AudioSource footstepAudioSource;

    public AudioClip jumpSound;

    public AudioClip[] footstepSounds; // Array to hold footstep sounds if varying by surface or speed

    private CharacterController characterController;

    private void Start()

    {

        characterController = GetComponent<CharacterController>();

    }

    private void Update()

    {

        HandleFootsteps();

    }

    void HandleFootsteps()

    {

        if (characterController.isGrounded && characterController.velocity.magnitude > 0.1f && !footstepAudioSource.isPlaying)

        {

            footstepAudioSource.clip = footstepSounds[Random.Range(0, footstepSounds.Length)];

            footstepAudioSource.Play();

        }

    }

    public void Jump()

    {

        if (characterController.isGrounded)

        {

            footstepAudioSource.PlayOneShot(jumpSound);

        }

    }

}

**5. Shoot Script:**

* Uses raycasting to detect and interact with objects when the player presses the shoot button.
* Disables the enemy animator upon hitting an enemy and applies an impact force to simulate damage.
* Logs details of hit objects and adjusts the impact based on proximity to the raycast hit point.

using System.Linq;

using UnityEngine;

using UnityEngine.InputSystem;

public class Shoot : MonoBehaviour

{

    [SerializeField]

    InputAction shootAction;

    [SerializeField]

    Transform shootPoint;

    bool isFired;

    private void OnEnable()

    {

        shootAction.Enable();

    }

    private void Update()

    {

        if (shootAction.IsPressed())

        {

            isFired = true;

        }

    }

    private void FixedUpdate()

    {

        ShootAction();

    }

    private void ShootAction()

    {

        RaycastHit hit;

        Vector2 screenPoint = new Vector2(Screen.width / 2, Screen.height / 2);

        Ray ray = Camera.main.ScreenPointToRay(screenPoint);

        Debug.DrawRay(ray.origin, ray.direction \* 1000f, Color.yellow);

        bool isHit = Physics.Raycast(ray, out hit, Mathf.Infinity);

        if(isHit)

        {

            if (isFired)

            {

                if(hit.rigidbody)

                {

                    GameObject hitGO = hit.rigidbody.gameObject;

                    Debug.Log("Info: " + hit.collider.gameObject.name);

                    Animator enemyAnimator = hitGO.GetComponentInParent<Animator>();

                    enemyAnimator.enabled = false;

                    Rigidbody[] rbs = hitGO.GetComponents<Rigidbody>();

                    Rigidbody hitRB = rbs.OrderBy(rigidbody => Vector3.Distance(rigidbody.position, hit.point)).First();

                    hitRB.AddForceAtPosition(-hitRB.gameObject.transform.forward \* 100, hit.point, ForceMode.Impulse);

                }

                isFired = false;

            }

        }

    }

}