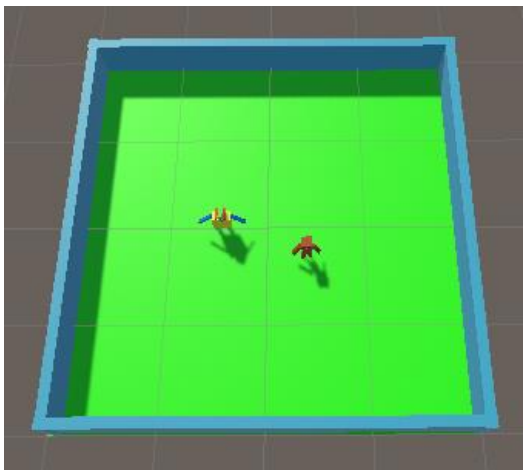
Lab URL: https://hub.labs.coursera.org:443/connect/sharediufevvlr?forceRefresh=false

Task 1: Create a 3D scene using basic graphics techniques, laying object out using transforms



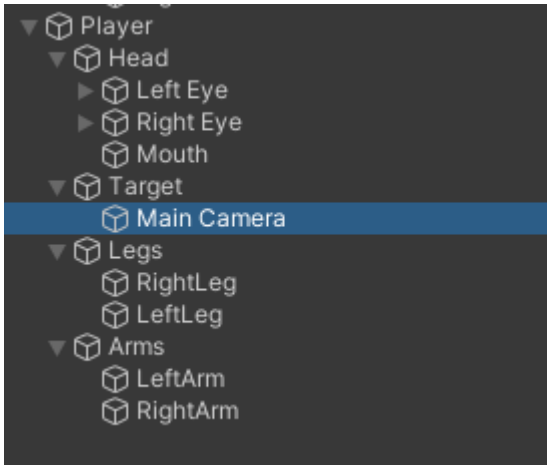
For this task, I used plane and cubes to create the floor and walls respectively. By changing the scale value of the cube, the cube transformed into a wall and by duplicating the wall into 4, changing of the position of the wall which became the wall that surrounded the plane creating a bounded field. By creating materials and applying to the floor and walls have made the place more better in aesthetic effect than the default white colour.

Task 2: Create at least 1 moving object that uses user input (e.g. keyboard input) to move an object (or the camera) using transforms

For this task, I have created a wooden stickman object to move around with keyboard. A PlayerMovement script(screenshot is at the last page of this report) is attached to it. The code that moves the stickman is the same as what is taught in coursera, however, I added animation for jump and walk because it will make the stickman move more realistic. If the user pressed vertical or horizontal input, it will change the boolWalk parameter to true to trigger the walk animation else it will set to false which will default to the idle animation. For the jump, I have used OnCollisionEnter function to track if the stickman have landed on the floor or not and if it has landed on the floor, the boolJump and boolInAir will be set to false which will then set the stickman back to it's idle animation. I created material for brown and black which is used for the stickman colour and black for the eye ball colour.

Extended Task: Implement a third person camera that follows a moving object

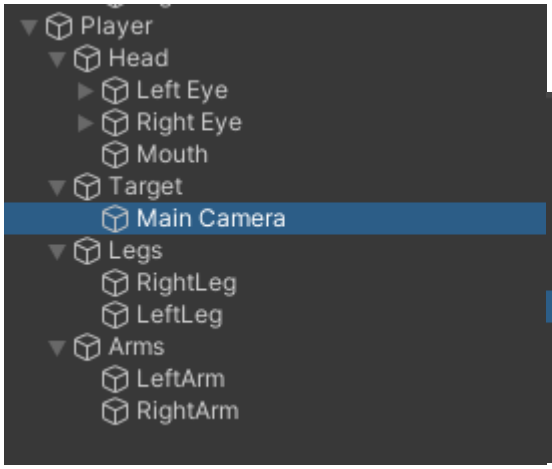
For this task, I have referred to youtube tutorial which will be referenced in the last page of the report. The reason why I used youtube tutorial code than the one we are taught in coursera is because what I wanted to achieve for the movement control to be like genshin impact control. This means that the character will face towards where the mouse pointed and move forward when the mouse move left or right. So to do that, the player object must have an empty object which is the target. This target object is placed behind of the player object and the main camera is placed inside the target object. So when the character move, the target will follow the character and the main camera will follow the target which created the third party view. The CameraController script is attached to the Main Camera.

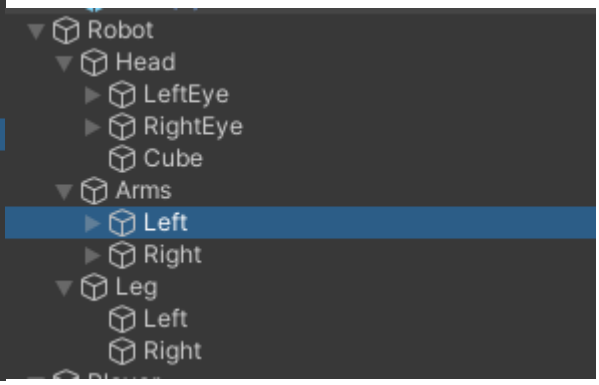


In the CameraController script, the code in the Start function is just making the cursor invisible. The code in the LateUpdate function is tracking the mouse input, for mouseX(left and right) we add in the “Mouse X” value * rotationSpeed which is 3 in this case. For mouseY(up and down) we minus the “Mouse Y” value * rotationSpeed. The reason is because we want the camera to follow the mouse direction when moving up and down if it is a plus, it will be an inverse action meaning moving up the mouse will move the camera down vice versa. Next clamping the value of mouseY is to prevent the screen from flipping around(limit how much the camera can move). If left shift is pressed, the mouse can be moved without the player facing the same direction else player will always face the direction the camera is facing.

Extras

For steering behaviours for controlling characters, I have implemented a simple seek and flee action for the robot. It can be activated by pressing 1(Alpha1) to activate seek and 2(Alpha2) for fleeing. The code for this action can be found here. For creating a hierarchical moving objects is the robot and the stickman.





References:

Making a third person camera: https://youtu.be/7nxpDwnU0uU?t=190

PlayerMovement.cs

```
1 reference
[SerializeField] float movementSpeed;
2 references
private Rigidbody playerRigidBody;
4 references
private bool isJumping;

0 references
private void Start()
{
    playerRigidBody = GetComponent<Rigidbody>();
}
```

```
private void Update()
{
    if (Input.GetButton("Vertical") || Input.GetButton("Horizontal"))
    {
        this.GetComponent<Animator>().SetBool("boolWalk", true);
    }
    else
    {
        this.GetComponent<Animator>().SetBool("boolWalk", false);
    }

    if (Input.GetButtonDown("Jump"))
    {
        if (isJumping == false)
        {
            isJumping = true;
            playerRigidBody.AddForce(transform.up * 10f, ForceMode.Impulse);
            this.GetComponent<Animator>().SetBool("boolJump", true);
        }
    }

    if (isJumping)
    {
        this.GetComponent<Animator>().SetBool("boolInAir", true);
    }
    else
    {
        this.GetComponent<Animator>().SetBool("boolJump", false);
        this.GetComponent<Animator>().SetBool("boolInAir", false);

    }

    float forward = Input.GetAxis("Vertical");
    float side = Input.GetAxis("Horizontal");

    Vector3 movementDirection = new Vector3(side, 0, forward).normalized;
    transform.Translate(movementDirection * movementSpeed * Time.deltaTime);
}
```

```
private void OnCollisionEnter(Collision other)
{
    if (other.collider.tag == "Floor")
    {
        isJumping = false;
    }
}
```

CameraController.cs

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

0 references | You, 37 seconds ago | 1 author (You)
public class CameraController : MonoBehaviour
{
    2 references
    public float rotationSpeed = 3;
    3 references
    [SerializeField] GameObject camTarget;
    1 reference
    [SerializeField] GameObject player;
    4 references | 6 references
    private float mouseX, mouseY;

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

    0 references
    private void LateUpdate()
    {
        mouseX += Input.GetAxis("Mouse X") * rotationSpeed;
        mouseY -= Input.GetAxis("Mouse Y") * rotationSpeed;
        // prevent camera from flipping around
        mouseY = Mathf.Clamp(mouseY, -35, 50);       You, now • Uncommitted changes
        Debug.Log(mouseY);

        transform.LookAt(camTarget.transform);

        if (Input.GetButton("Fire3"))
        {
            camTarget.transform.rotation = Quaternion.Euler(mouseY, mouseX, 0);
        }
        else
        {
            camTarget.transform.rotation = Quaternion.Euler(mouseY, mouseX, 0);
            player.transform.rotation = Quaternion.Euler(0, mouseX, 0);
        }
    }
}
```

RandomObjectMovement.cs

```
public class RandomObjectMovement : MonoBehaviour
{
    2 references
    [SerializeField] GameObject player;
    2 references
    private float movementSpeed = 5f;
    1 reference
    private float distanceFromTarget = 3f;
    2 references
    private float rotationSpeed = 10f;
    3 references
    private bool isFleeing = false;


    // Start is called before the first frame update
    0 references
    void Start()
    {
        SeekPlayer();
    }

    // Update is called once per frame
    0 references
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Alpha1)) isFleeing = false;
        else if (Input.GetKeyDown(KeyCode.Alpha2)) isFleeing = true;

        if (isFleeing) FleeFromPlayer();      You, 1 second ago • Uncommitted changes
        else SeekPlayer();
    }

    2 references
    private void SeekPlayer()
    {
        Vector3 direction = player.transform.position - this.transform.position;
        // rotate the robot to face the player
        transform.rotation = Quaternion.Slerp(transform.rotation, Quaternion.LookRotation(direction), rotationSpeed * Time.deltaTime);
        // check if current object is near target, if is further away from target, move to target
        if (direction.magnitude > distanceFromTarget)
        {
            Vector3 moveVector = direction.normalized * movementSpeed * Time.deltaTime;
            transform.position += moveVector;
        }
    }

    1 reference
    private void FleeFromPlayer()
    {
        Vector3 direction = this.transform.position - player.transform.position;
        // check if current object is near target, if is near from target, move away from the target
        if (direction.magnitude < 100f)
        {
            // rotate the robot away form the player
            transform.rotation = Quaternion.Slerp(transform.rotation, Quaternion.LookRotation(direction), rotationSpeed * Time.deltaTime);
            Vector3 moveVector = direction.normalized * movementSpeed * Time.deltaTime;

            transform.position += moveVector;
        }
```