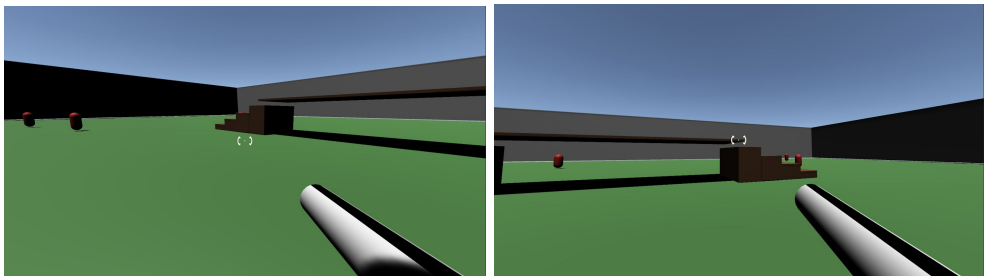
Tyler Bentley
CS412
Russel Butler

Overview

The goal of this assignment was to create a simple fps level with movement, shooting mechanics and enemies that run around. This assignment had 3 major components of: Camera and Movement , Crosshair , Level and Enemies.

Camera and Movement

1. Mouse Look

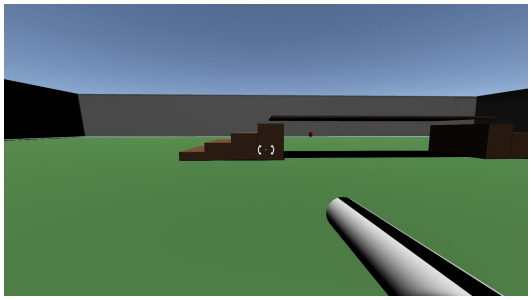
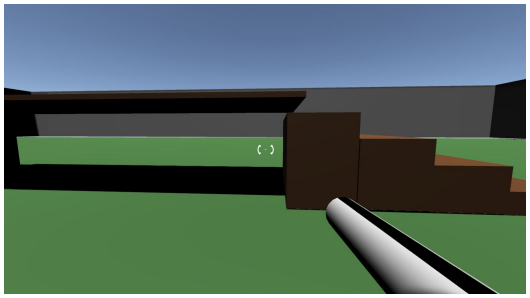


Using the mouse , a vertical input will pan the camera controller up and down and a horizontal input will move from side to side

```
// Update is called once per frame
Unity Message | 0 references
void Update()
{
    //gets a value to change the rotation of the camera
    float mouseX = Input.GetAxis("Mouse X") * mouseSensitivityX * Time.deltaTime;
    float mouseY = Input.GetAxis("Mouse Y") * mouseSensitivityY * Time.deltaTime;

    //Rotation
    playerBody.Rotate(Vector3.up * mouseX);
    xRotation -= mouseY;
    xRotation = Mathf.Clamp(xRotation,-90,90);
    transform.localRotation = Quaternion.Euler(xRotation,0f,0f);

}
```

Float mouseX and MouseY are modified by a sensitivity constant then those values rotate the camera about the x and y axis.

2. Movement

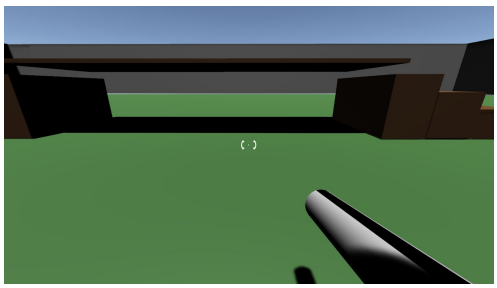
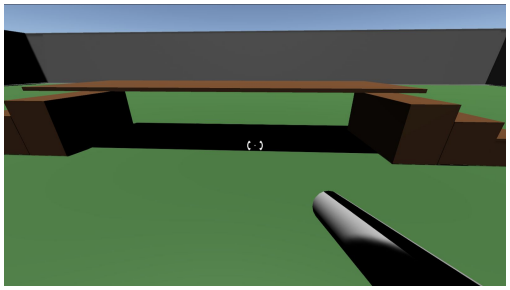
Movement is performed using the WASD keys, there input modifies the position of the player on the x and z axis.

```
//WASD
float x = Input.GetAxis("Horizontal");
float z = Input.GetAxis("Vertical");
Vector3 move = transform.right * x + transform.forward * z ;
controller.Move(move/15);

velocity.y += gravity * 2f * Time.deltaTime;
controller.Move(velocity * Time.deltaTime);
```

The FPS controller setup in unity already natively binds W,A,S and D to the Horizontal and Vertical Axes. The code inputs those to a vector of which the controller moves by.

3. Jump/Gravity

The gravity of this level emulates the values of our real world gravity.

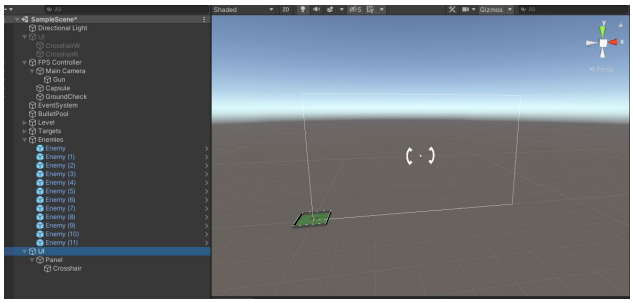
The space bar is the bound ‘jump’ button, it gives the player a certain amount of upward velocity. This velocity is then flipped due to the negative acceleration (downwards) known as gravity.

```
public CharacterController controller;
public Transform groundCheck;
public float groundDistance = 0.4f;
public LayerMask groundMask;
Boolean isGrounded;
Vector3 velocity;
public float gravity = -9.81f;
public float jumpheight = 3f;

// Update is called once per frame
Unity Message | 0 references
void Update()
{
    isGrounded = Physics.CheckSphere(groundCheck.position, groundDistance, groundMask);
    if(isGrounded && velocity.y < 0)
    {
        velocity.y = -2f;
    }
    //WASD
    float x = Input.GetAxis("Horizontal");
    float z = Input.GetAxis("Vertical");
    Vector3 move = transform.right * x + transform.forward * z ;
    controller.Move(move/15);

    velocity.y += gravity * 2f * Time.deltaTime;
    controller.Move(velocity * Time.deltaTime);

    if(Input.GetButtonDown("Jump") && isGrounded)
    {
        velocity.y = Mathf.Sqrt(jumpheight * -2 * gravity);
    }
```

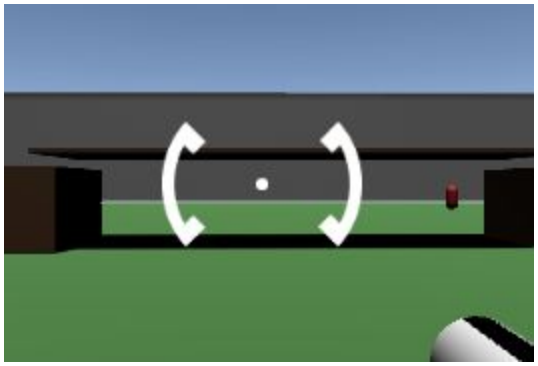
A boolean isGrounded checks to see if the player is on the ground.
Unity aswell has pre-assigned “jump” to spacebar.
Jumping assigns a certain amount of y velocity based on jump height and gravity.

Crosshair

1. 10% visible at all time (does not get occluded by objects)
    a. The crosshair is a UI canvas and therefore is applied to the visible screen and not the rendered world. It' never occluded as it is always drawn above everything.

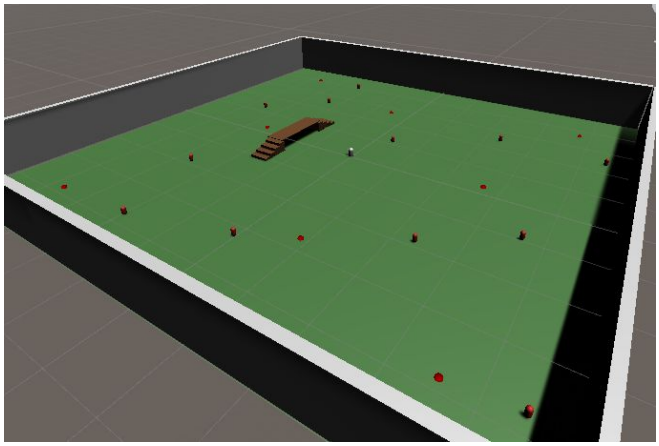


2. 10% cross-shaped (see image above)



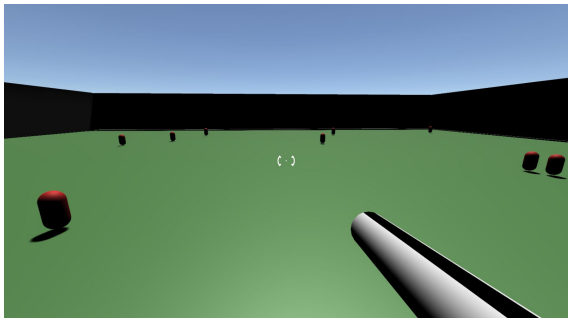
Level and Enemies

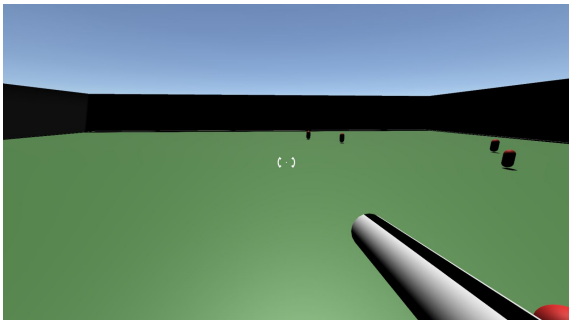
1. The Level



A simple level with a bridge to test step height , the floor and bridge are coloured to give a nicer look to the environment.

2. Enemy Behaviour





Enemies have 2 States: RUN and STAY.
Run has them pathfinding to one of 8 targets around the environment. Once at these points they stay for around 5 seconds before choosing a new target and moving to it.

```
void Update()
{
    enemy.updatePosition = false;
    enemy.updateRotation = false;
    enemy.updateUpAxis = false;
    NextState -= Time.deltaTime;


    switch (State)
    {
        case StateEnum.RUN:
            if (enemy.desiredVelocity.magnitude < 0.1f)
            {
                State = StateEnum.STAY;
                NextState = Random.Range(3f, 7f);
            }

            break;
        case StateEnum.STAY:
            if(NextState < 0)
            {
                State = StateEnum.RUN;
                target = PotentialTargets[Random.Range(0, PotentialTargets.Length)];
                enemy.SetDestination(target.transform.position);
            }
            break;

    }
    transform.position += enemy.desiredVelocity * Time.deltaTime;
}
```

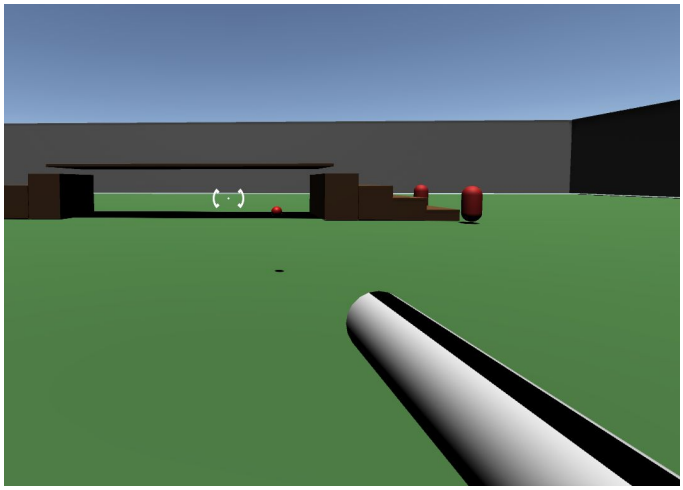
Using NavMeshAgent , the engine pathfinds through the static level to targets around the map. They switch states once the desired velocity is low meaning they are at the target.
Once some amount of time passes , the state changes and has a new target to go to.

The movement is done by code and the Engine pathfinding only gives the desired direction.

```
Unity Message | 0 references
private void OnTriggerEnter(Collider other)
{
    Destroy(badguy);
}
```

The enemies will also be deleted given a collision trigger, such as a bullet (covered later) or contact with the player (melee attack).

3. The Gun



The left mouse spawns small balls at the end of the barrel of the cylinder, these balls are given a velocity and launch forwards.

```
public GameObject bullet;
public AudioSource audioSourcePlayer;


Unity Message | 0 references
private void Start()
{
    audioSourcePlayer = GetComponent<AudioSource>();
}
// Update is called once per frame
Unity Message | 0 references
void Update()
{
   if(Input.GetMouseButtonDown(0))
    {
        audioSourcePlayer.Play();
        GameObject newBullet = objectpool.SharedInstance.getPooledObject();
        newBullet.transform.position = new Vector3(transform.position.x , transform.position.y, transform.position.z);
        newBullet.GetComponent<Rigidbody>().velocity = transform.up* 50;
        newBullet.SetActive(true);



    }
```

The gun spawns a bullet from the bullet pool which will last for 3 seconds and are then recycled.
When fired a simple “silenced gun sound” sound effect is played which was created by pauliep83 on https://freesound.org/people/pauliep83/sounds/34251/.