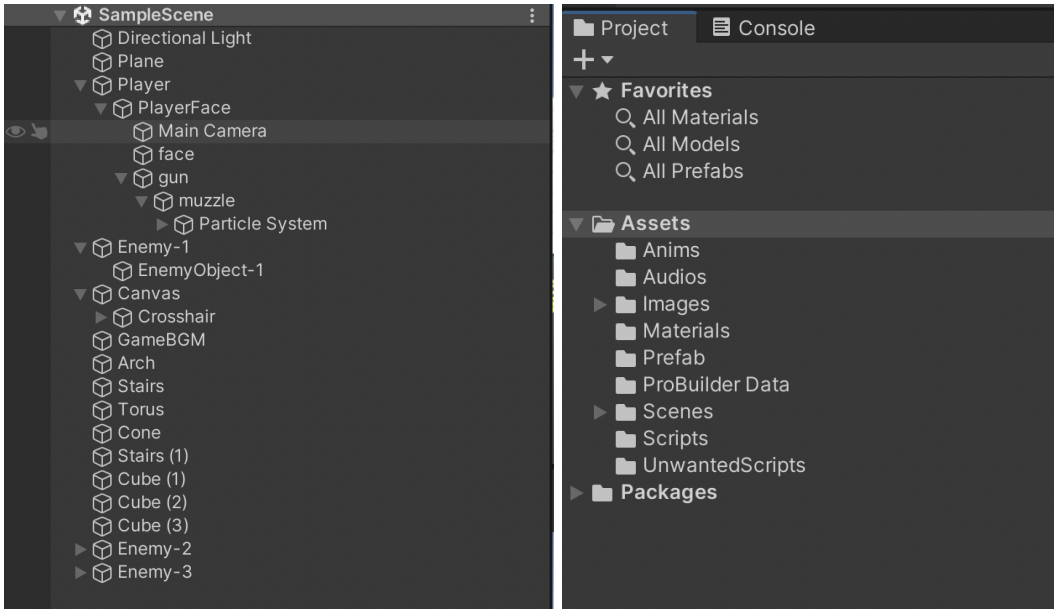
Report

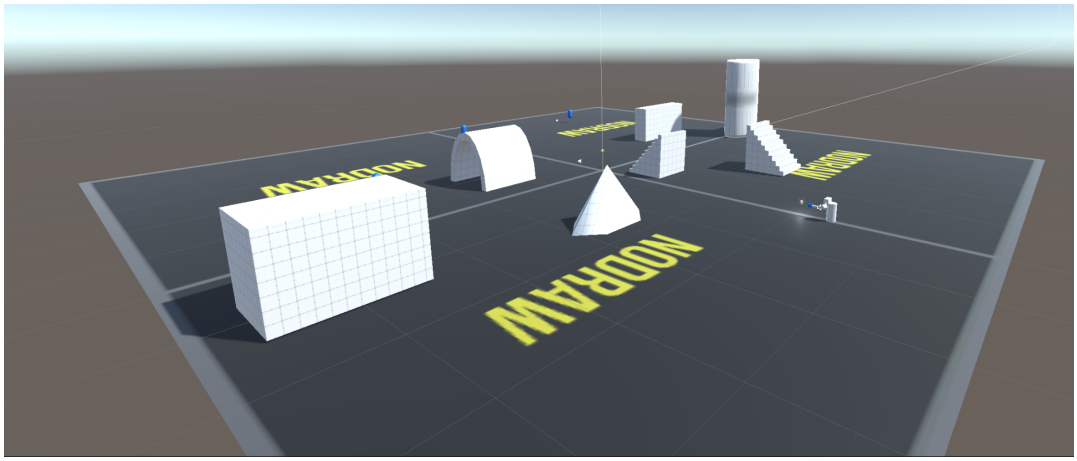
Unity Project:

The 3D objects required have been created using unity controllers(Drag and drop). Below is the screenshot of the hierarchy the objects are created.

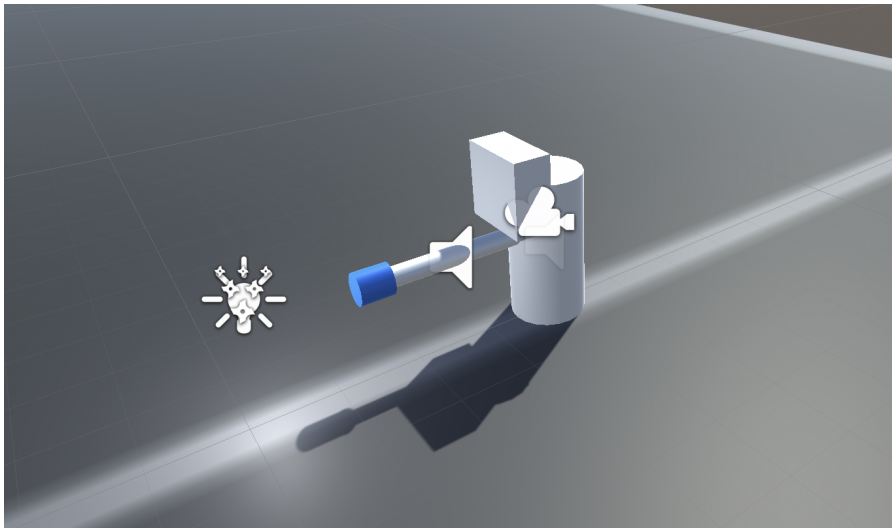


Environment :

Have used ProBuilder to create objects onto the plane. Also added the background audio that plays when the game is running.

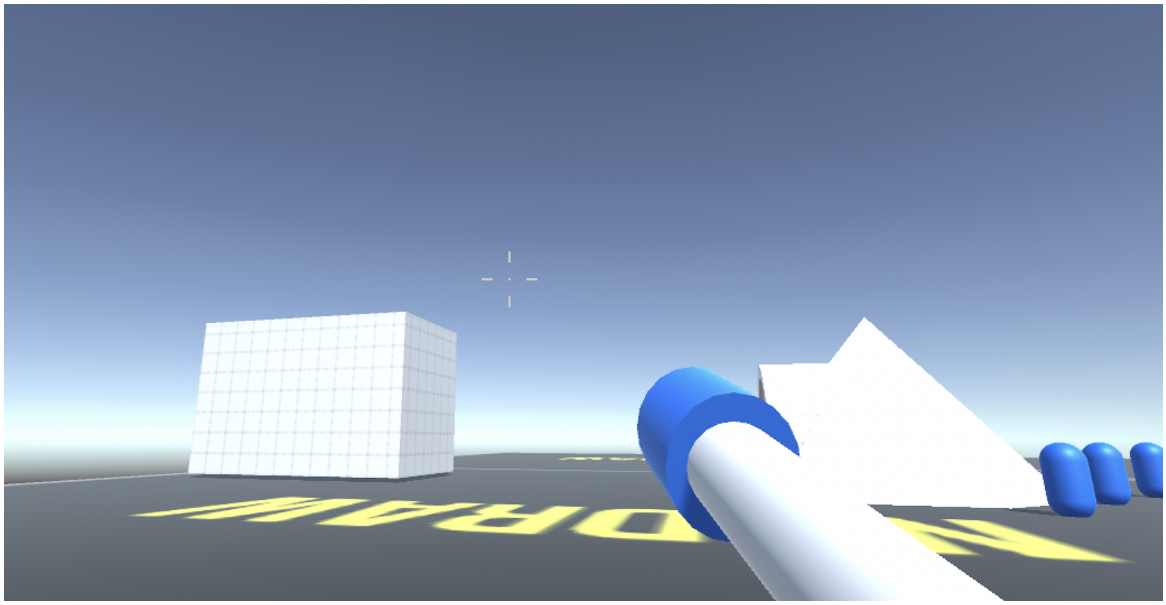


- The Player Object is replicated with a player body and gun along with a muzzle. The main camera is placed slightly beside the gun so that we can get a better feel of the game.



Basic camera control for movement and look :

- To achieve player movements along with **mouse direction and WASD keyboard move** we mapped the player controller component with the script.
- The key factors like Player speed, rotation, look at speed, velocity are taken into consideration while replicating realistic play movements based on the selection of keys.
- Dynamically fetching the X,Y,Z coordinates of the player's at each frame and applied to the updated play rotation.
- Using Unity's Predefined shortcuts, we fetched the keys corresponding to the vertical and horizontal motion.



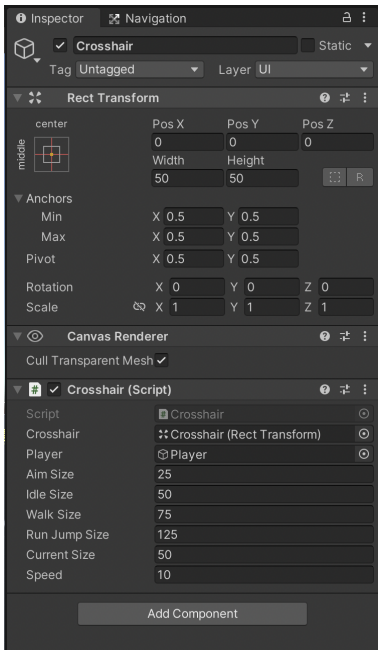
- Created a function to check if the player is in air or on ground to limit the jumping.For jump, we fetched the corresponding key and wherever the key is pressed we added some

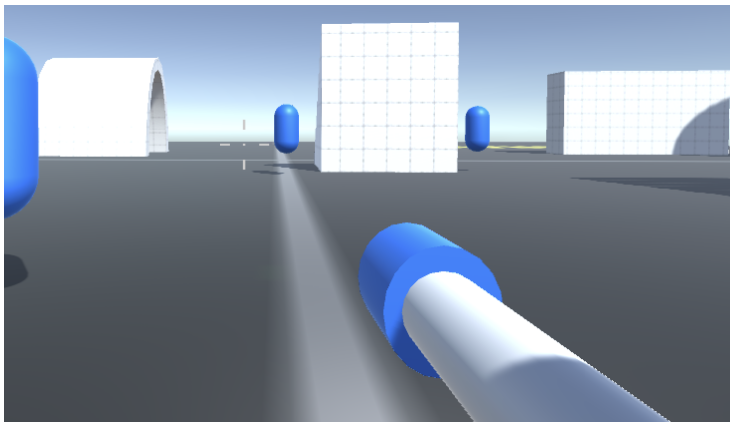
velocity along with gravity which represents the **player's jump motion** along with time check to eliminate the continuous jump in air.

```
groundCheck();
velocity += gravity * Time.deltaTime;

float jumpVal = Input.GetAxis("Jump");

if(isGrounded && jumpVal == 1){
    velocity = jumpForce;
}

controller.Move(new Vector3(0, velocity, 0) * Time.deltaTime);

Vector3 forward = transform.TransformDirection(Vector3.forward);
Vector3 right = transform.TransformDirection(Vector3.right);
```

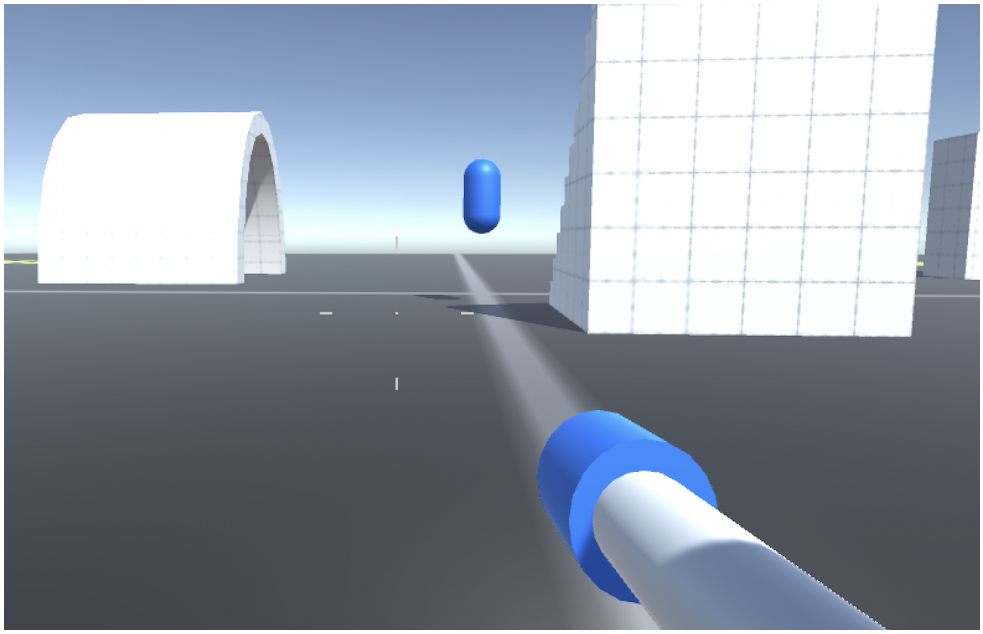
Dynamic Crosshair :

- With the help of canvas>panel we created a crosshair with appropriate coordinates and color to get a crosshair look.





- The crosshair coordinates were updated with help for script based on player motion like **Idle, Running, Jumping, Aiming**.



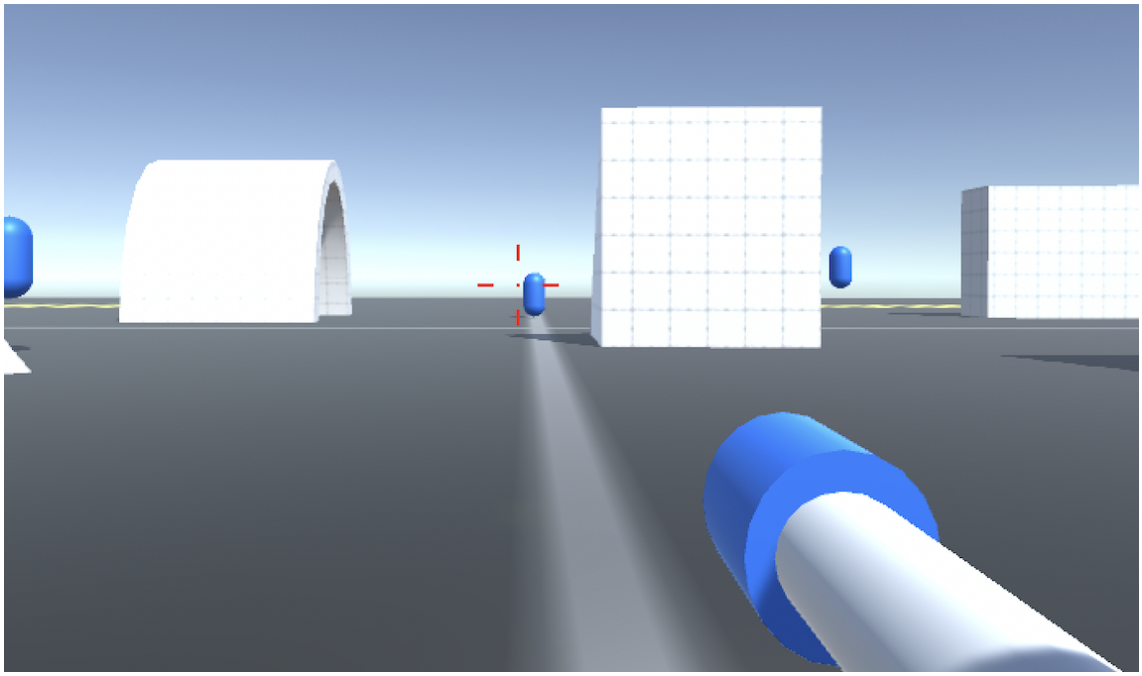
- Updated the coordinates vector3 component to get the zoom in effect while aiming

```
void Update(){

    if (Aiming)
        currentSize = Mathf.Lerp(currentSize, aimSize, Time.deltaTime * speed);
    else if (Walking)
        currentSize = Mathf.Lerp(currentSize, walkSize, Time.deltaTime * speed);
    else if (Running || Jumping)
        currentSize = Mathf.Lerp(currentSize, runJumpSize, Time.deltaTime * speed);
    else
        currentSize = Mathf.Lerp(currentSize, idleSize, Time.deltaTime * speed);

    crosshair.sizeDelta = new Vector2(currentSize, currentSize);

}
```

(field) float Crosshair.currentSize

- With the help of Rayhit, able to change the color of the crosshair whenever the crosshair is onto the enemy.

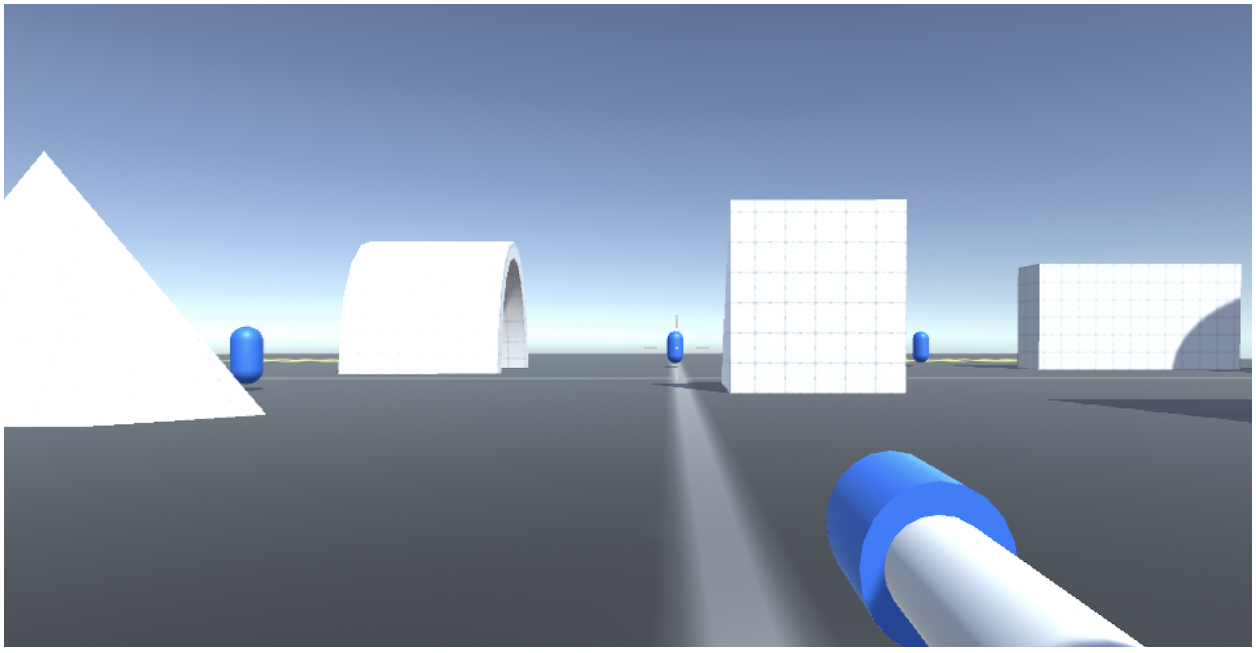
```
RaycastHit hit;
if (Physics.Raycast(transform.position, transform.forward, out hit, 200f))
    if ((hit.transform.gameObject.name == "EnemyObject-1") || (hit.transform.gameObject.name == "EnemyObject-2") || (hit.transform.gameObject.name == "EnemyObject-3"))
        foreach (Image crosshairImage in crosshairImages)
            crosshairImage.color = new Color(1f, 0f, 0f, 1f);
    else
        foreach (Image crosshairImage in crosshairImages)
            crosshairImage.color = new Color(0.8f, 0.8f, 0.8f, 1f);
else
    foreach (Image crosshairImage in crosshairImages)
        crosshairImage.color = new Color(0.8f, 0.8f, 0.8f, 1f);
```



Single level with gameplay :

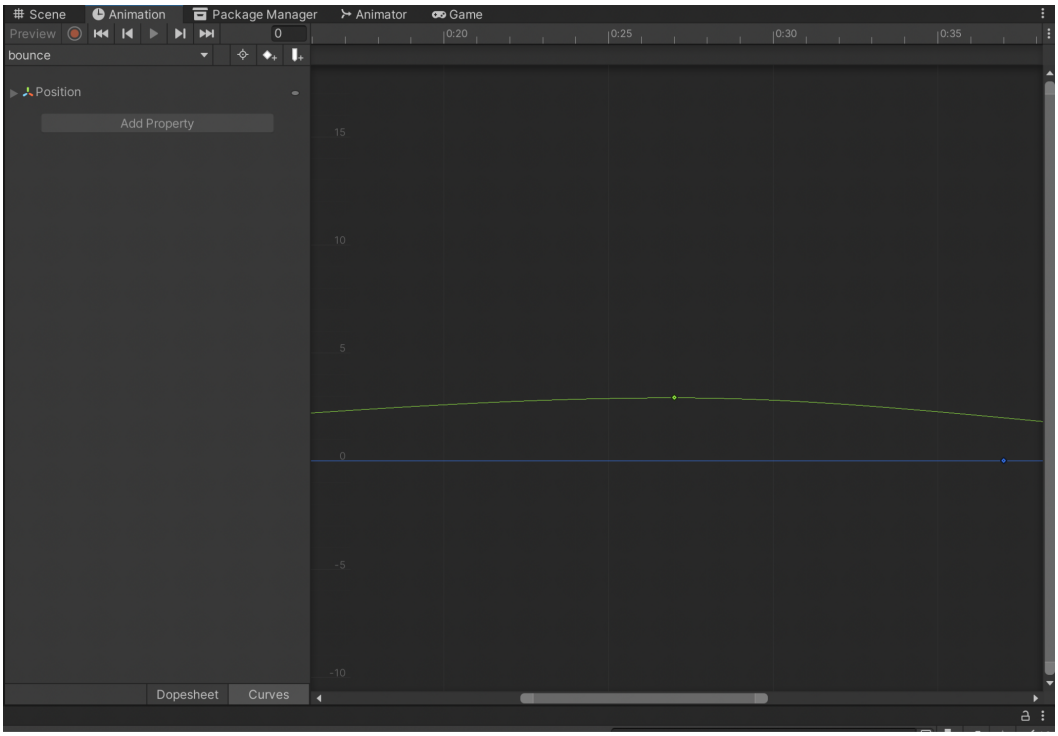
Enemy :

Create an enemy with the navmesh agent whose destination is set as the player location, so that enemy will create its own path to travel towards the player.



```
public class EnemyMovement : MonoBehaviour
{
    NavMeshAgent enemy;
    public Transform player;
    int hitRate = 3;
    // Start is called before the first frame update
    void Start()
    {
        enemy = GetComponent<NavMeshAgent>();
    }

    // Update is called once per frame
    void Update()
    {
        enemy.destination = player.position;
    }

    public void TakeDamage(int hitValue)
    {
        hitRate -= hitValue;
        Debug.Log("Bullet Hit By Enemy - " + hitRate);
        if (hitRate <= 0)
        {
            gameObject.SetActive(false);
        }
    }
}
```

Created an EnemyObject as a child of the enemy to add all kinds of post processing like animation so that it doesn't affect the enemy path towards the player.



Weapon :

- Created a 3D Object to replicate the gun.
- Used Rayhit component to find the target object, and wherever the target object is enemy then invoking the TakeDamage function to implement the enemy health.
- Whenever the 3 bullets collided with the enemy then treating the enemy got killed and deactivating it.

```
if (Input.GetAxis("Fire1") == 1 && Time.time > nextShot)
{
    nextShot = Time.time + timeBetweenShots;
    bullets[bulletCounter].SetActive(true);
    audioSource = GetComponent<AudioSource>();
    audioSource.Play();
    (GameObject.Find("gun")).GetComponent<gunBullets>().shoot();
    bulletCounter++;
}
```

```
public void shoot()
{
    muzzleFlash.Play();
    RaycastHit hit;
    if (Physics.Raycast(fpsCam.transform.position, fpsCam.transform.forward, out hit, 200f))
    {
        Debug.Log("target - " + hit.transform.gameObject.name);
        //if (hit.transform.gameObject.name == "EnemyObject-1") ;
        string hitObjName = hit.transform.gameObject.name;
        Debug.Log("Enemy Object Name - " + hitObjName);
        if (hitObjName.Contains("EnemyObject"))
        {
            string[] enemyObjNameArr = hitObjName.Split('-');
            string enemyName = "Enemy-" + enemyObjNameArr[1];
            Debug.Log("Enemy Name - " + enemyName);
            AudioSource source = GetComponent<AudioSource>();
            source.Play();
            (GameObject.Find(enemyName)).GetComponent<EnemyMovement>().TakeDamage(1);
        }
    }
}
```

Whenever the bullet is fired, a particle system and audio makes a realistic effect.