I'm working on a fps game on Unity, actually just a prototype using primitives. Actually working on the player's shooting mechanics before moving to the rest. The fov should be in god's eye view because there will be spheres rolling inside a big box (the playground). The camera should be fixed and look just at this box from above. The re's a black sphere that the player shouyd defend at any cost by shooting at the enemies. Enemies are other spehres moving within the play gorund. All the spheres move freely, and the black sphere (here called "Cindy") will be destroyed on collision, and it's game over. The enemies are spawn on a fixed number based on the scene level. First scene there's one enemy and so on. Once all the spawned enemies are destroyed there's another spawn. Each scene lasts for 33 seconds. There will be a score system for this prototype.

Let's go to Unity.

I provide here an overview of my hierarchy structure: - Ground - Counter - Box (made of cudes, here as targets) - Player (child: Capsule, Main Camera, Gun, Dynamic Crosshair Canvas) - Spawn Manager.

The Shooting Point is an empty object located at the gun tip (-0.39, 0.99, 0.23).

The Gun Object position is set to -0.29, 0, 0.9 and it's parent to an empty object set as Pivot to assure proper rotation toward the gun handle.

The Gun Object, the Crosshair Canvas and the Main camera are all child object of the Player.

The player position is set to 0, 1, 0.

The canvas is set to the center (0, 0, 0) with pivot 0.5, 1.

By pressing Play, wIdth and Height resize from 100 to 75 though. Camera FOV is set to 80. The bullet instantiation point is at -0.39, 0.99, 0.23.

The bullet prefabs' transforms are set as follow; position (-0.29, 1, 0.9), rotation (18.695, -67.603, 0), scale (0.1, 0.1, 0.1). By draggig them in the hierarchy they show exactly at the shooting point.

The Spawn Manager is used to spawn the bullets through an ObjectPooler, they become its child as soon as they’re spawned and deactivate when they collide or when they go out of bounds.

The reticle for the crosshair is a child canvas on a canvas called "Dynamic Crosshair" which has Screen Space - Overlay set as render mode and UI Scale Mode on Scale with Screen Size. No other adjustments are been made from the default values. The Reticle Canvas has Pivot (0.5, 0.5) and two scripts attached: Reticle, Move Crosshair. Here's the Move Crosshair: The Reticle Canvas is made by five child Canvas (UI Images), that form the crosshair: Top Reticle Line (width 2, height 2, pivot 0.5, 1), Bottom Reticle Line (width 2, height 25, pivot 0.5, 0), Left Reticle Line (width 2, height 25, pivot 0, 0.5), Right Reticle Line (width 2, height 25, pivot 1, 0.5). Center (width 7, height 7, pivot 0.5, 0.5). Left and Right Reticle have Z Rotation set to -90.

The Gun Object has a Capsule Collider Component and the Shooting Script.

The Main Camera is positioned 0.7 on the Y and has the LookAtCrosshair Script.

The Reticle has the Reticle script and the MoveCrosshair script.

The bullets have the MoveForward script.

The player’s capsule has a Capsule Collider.

The Enemy Prefab has the MoveEnemy script, a Box Collider and an animator.  
The Object Cindy has the MoveCindy script, a sphere collider, a rigidbody.

The Spawn Manager empty object has the SpawnManager Script (to spawn enemies) and the ObjectPooler script.

The GUI object has some child nested canvases used for the GUI.  
The Game Manager object controls all the UI actions and has a relative script attached.

Here all the updated scripts:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using TMPro;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

public class GameManager : MonoBehaviour

{

public TextMeshProUGUI scoreText;

public TextMeshProUGUI gameOverText;

public GameObject titleScreen;

public Button restartButton;

private Button playButton;

private int score;

public bool isGameActive;

public float timeLeft;

public bool timerOn = false;

public TextMeshProUGUI timerText;

public GameObject pauseScreen;

private bool paused;

public void StartGame()

{

isGameActive = true;

score = 0;

timerOn = true;

UpdateScore(0);

UpdateTimer(37);

titleScreen.SetActive(false);

}

void ChangePaused()

{

if(!paused)

{

paused = true;

pauseScreen.SetActive(true);

Time.timeScale = 0;

}

else

{

paused = false;

pauseScreen.SetActive(false);

Time.timeScale = 1;

}

}

void Update()

{

if(timerOn)

{

if(timeLeft > 0)

{

timeLeft -= Time.deltaTime;

UpdateTimer(timeLeft);

timerText.text = "TIME" + timeLeft;

}

else

{

Debug.Log("Time is over");

timeLeft = 0;

timerOn = false;

restartButton.gameObject.SetActive(true);

isGameActive = false;

}

}

if (Input.GetKeyDown(KeyCode.P))

{

ChangePaused();

}

}

void UpdateTimer(float currentTime)

{

currentTime += 1;

//float minutes = Mathf.FloorToInt(currentTime / 60);

float seconds = Mathf.FloorToInt(currentTime % 60);

timerText.text = string.Format("00", seconds);

}

// Update score with value from target clicked

public void UpdateScore(int scoreToAdd)

{

score += scoreToAdd;

scoreText.text = "KILLS" + score;

/\*if(score < 0)

{

GameOver();

timerOn = false;

}\*/

}

// Stop game, bring up game over text and restart button

public void GameOver()

{

gameOverText.gameObject.SetActive(true);

restartButton.gameObject.SetActive(true);

isGameActive = false;

}

// Restart game by reloading the scene

public void RestartGame()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().name);

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class LookAtCrosshair : MonoBehaviour

{

public Camera fpsCam;

public GameObject bullet;

private Vector3 targetPoint;

[SerializeField] private GameObject shootingPoint;

[SerializeField] private GameObject gun;

[SerializeField] private Transform reticle;

private float distance = 50;

void FixedUpdate()

{

UpdateTargetPoint(); // Ensure targetPoint is updated before FixedUpdate ends

Ray rayOrigin = fpsCam.ScreenPointToRay(new Vector2(reticle.position.x, reticle.position.y));

RaycastHit hit;

if (Physics.Raycast(rayOrigin, out hit, distance))

{

targetPoint = hit.point;

// Rotate the gun towards the target point

Vector3 direction = (targetPoint - shootingPoint.transform.position).normalized;

gun.transform.rotation = Quaternion.LookRotation(direction);

//Debug.DrawRay(rayOrigin.origin, rayOrigin.direction \* hit.distance, Color.red);

//Debug.DrawRay(rayOrigin.origin, rayOrigin.direction \* distance, Color.blue);

//Debug.Log("Target Point: " + targetPoint);

}

}

void UpdateTargetPoint()

{

float step = bulletSpeed \* Time.deltaTime;

targetPoint = Vector3.MoveTowards(transform.position, reticle.position, step);

}

private float bulletSpeed; // Speed of the bullet

public void SetBulletSpeed(float speed)

{

bulletSpeed = speed;

}

public Vector3 GetTargetPoint()

{

return targetPoint;

}

}

using UnityEngine;

public class MoveCindy : MonoBehaviour

{

public Transform target;

private Vector3 direction;

private Rigidbody cindyRb;

private GameManager gameManager;

private float speed = 5;

private float bounce = 100;

private void Start()

{

cindyRb = gameObject.GetComponent<Rigidbody>();

gameManager = GameObject.Find("Game Manager").GetComponent<GameManager>();

}

private void Update()

{

// Calculate the direction to the target

direction = (target.position - transform.position).normalized;

MoveAwayFromTarget(direction);

}

private void MoveAwayFromTarget(Vector3 direction)

{

// Move away from the target (opposite direction)

transform.Translate(-direction \* speed \* Time.deltaTime, Space.World);

RotateTowardsDirection(-direction);

}

private void RotateTowardsDirection(Vector3 targetDirection)

{

// Rotate smoothly towards the target direction

Quaternion targetRotation = Quaternion.LookRotation(targetDirection);

transform.rotation = Quaternion.Slerp(transform.rotation, targetRotation, Time.deltaTime \* 5f);

}

private void OnCollisionEnter(Collision other)

{

if (other.gameObject.CompareTag("Wall"))

{

// boing

cindyRb.AddForce(Vector3.back \* bounce, ForceMode.Impulse);

// Update the direction after the bounce

direction = (target.position - transform.position).normalized;

}

}

private void OnTriggerEnter(Collider other)

{

// gameManager.AddLives(-1);

other.gameObject.SetActive(false);

Destroy(gameObject);

Debug.Log("Game Over");

gameManager.GameOver();

}

}

using UnityEngine;

public class MoveCrosshair : MonoBehaviour

{

public Camera cam;

private float horizontalInput;

private float verticalInput;

bool mouseMoves;

private float speed = 2000;

private float fieldOfView = 60;

private float xRange = 960;

private float yRange = 540;

void Start()

{

Cursor.visible = false;

}

void Update()

{

// player controller

if (Input.GetAxis("Mouse X") != 0)

{

mouseMoves = true;

transform.position = Input.mousePosition;

} else if (Input.GetAxis("Horizontal") != 0 && !mouseMoves || Input.GetAxis("Vertical") != 0 && !mouseMoves)

{

horizontalInput = Input.GetAxis("Horizontal");

transform.Translate(Vector3.right \* Time.deltaTime \* speed \* horizontalInput);

verticalInput = Input.GetAxis("Vertical");

transform.Translate(Vector3.up \* Time.deltaTime \* speed \* verticalInput);

if (Input.GetKeyDown(KeyCode.LeftShift))

{

speed \*= 2;

} if (Input.GetKeyUp(KeyCode.LeftShift))

{

speed /= 2;

}

} else

{

mouseMoves = false;

}

// force on screen

if (transform.localPosition.x < -xRange)

{

transform.localPosition = new Vector3(-xRange, transform.localPosition.y, transform.localPosition.z);

} if (transform.localPosition.x > xRange)

{

transform.localPosition = new Vector3(xRange, transform.localPosition.y, transform.localPosition.z);

}

if (transform.localPosition.y < -yRange)

{

transform.localPosition = new Vector3(transform.localPosition.x, -yRange, transform.localPosition.z);

} if (transform.localPosition.y > yRange)

{

transform.localPosition = new Vector3(transform.localPosition.x, yRange, transform.localPosition.z);

}

// restict cam

if (Input.GetKeyDown(KeyCode.X))

{

cam.fieldOfView = fieldOfView / 2;

} if (Input.GetKeyUp(KeyCode.X))

{

cam.fieldOfView = fieldOfView;

}

// hide mouse pointer

if (Input.GetKeyDown(KeyCode.Escape))

{

Cursor.visible = true;

} if (Input.GetKeyDown(KeyCode.Q))

{

Cursor.visible = false;

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class MoveEnemy : MonoBehaviour

{

public Transform target;

private Animator enemyAnim;

private GameManager gameManager;

private float speed = 3;

private float rotationSpeed = 5;

private bool shouldMove = true;

void Start()

{

enemyAnim = GetComponent<Animator>();

gameManager = GameObject.Find("Game Manager").GetComponent<GameManager>();

}

private void Update()

{

if (shouldMove)

{

// Calculate the direction to the target

Vector3 direction = (target.position - transform.position).normalized;

// Move towards the target

transform.Translate(direction \* speed \* Time.deltaTime, Space.World);

RotateTowardsDirection(direction);

enemyAnim.SetBool("IsWalking", true);

}

else

{

// Stop moving when close to the target

enemyAnim.SetBool("IsWalking", false);

}

}

private void RotateTowardsDirection(Vector3 targetDirection)

{

Quaternion targetRotation = Quaternion.LookRotation(targetDirection);

transform.rotation = Quaternion.Slerp(transform.rotation, targetRotation, rotationSpeed \* Time.deltaTime);

}

private void OnCollisionEnter(Collision other)

{

if (other.gameObject.CompareTag("Cindy"))

{

Destroy(other.gameObject);

Debug.Log("Game Over");

gameManager.GameOver();

shouldMove = false;

}

//Debug.Log("Collision with: " + other.gameObject.name);

}

private void OnTriggerEnter(Collider other)

{

other.gameObject.SetActive(false);

Destroy(gameObject);

gameManager.UpdateScore(1);

}

}

using UnityEngine;

public class MoveProjectile : MonoBehaviour

{

private Rigidbody bulletRb;

private Vector3 targetPoint;

public float speed = 50;

private float lowerBound = 3;

void Start()

{

bulletRb = GetComponent<Rigidbody>();

targetPoint = GameObject.Find("Main Camera").GetComponent<LookAtCrosshair>().GetTargetPoint();

// Set the initial velocity towards the target point

bulletRb.velocity = (targetPoint - transform.position).normalized \* speed;

}

void Update()

{

if (transform.position.z < -lowerBound)

{

gameObject.SetActive(false);

}

}

/\*void FixedUpdate()

{

// Rotate the bullet towards the direction of motion

transform.rotation = Quaternion.LookRotation(bulletRb.velocity.normalized);

}\*/

private void OnCollisionEnter(Collision other)

{

gameObject.SetActive(false);

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ObjectPooler : MonoBehaviour

{

public static ObjectPooler SharedInstance;

public List<GameObject> pooledObjects;

public GameObject objectToPool;

public int amountToPool;

void Awake()

{

SharedInstance = this;

}

// Start is called before the first frame update

void Start()

{

// Loop through list of pooled objects,deactivating them and adding them to the list

pooledObjects = new List<GameObject>();

for (int i = 0; i < amountToPool; i++)

{

GameObject obj = (GameObject)Instantiate(objectToPool);

obj.SetActive(false);

pooledObjects.Add(obj);

obj.transform.SetParent(this.transform); // set as children of Spawn Manager

}

}

public GameObject GetPooledObject()

{

// For as many objects as are in the pooledObjects list

for (int i = 0; i < pooledObjects.Count; i++)

{

// if the pooled objects is NOT active, return that object

if (!pooledObjects[i].activeInHierarchy)

{

return pooledObjects[i];

}

}

// otherwise, return null

return null;

}

}

using UnityEngine;

public class MousePointerFollow : MonoBehaviour

{

// Adjust this value to control the speed of the pointer's movement

public float speed = 10;

void Update()

{

// Get the mouse position in world space

Vector3 mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);

mousePosition.z = 0f; // Ensure the pointer stays in the 2D plane

// Clamp the position within the game screen range

mousePosition.x = Mathf.Clamp(mousePosition.x, ScreenBounds.Left, ScreenBounds.Right);

mousePosition.y = Mathf.Clamp(mousePosition.y, ScreenBounds.Bottom, ScreenBounds.Top);

// Move the pointer towards the mouse position

transform.position = Vector3.Lerp(transform.position, mousePosition, speed \* Time.deltaTime);

}

}

public static class ScreenBounds

{

public static float Left => Camera.main.ScreenToWorldPoint(Vector3.zero).x;

public static float Right => Camera.main.ScreenToWorldPoint(new Vector3(Screen.width, 0f, 0f)).x;

public static float Bottom => Camera.main.ScreenToWorldPoint(Vector3.zero).y;

public static float Top => Camera.main.ScreenToWorldPoint(new Vector3(0f, Screen.height, 0f)).y;

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class Reticle : MonoBehaviour

{

private RectTransform reticle;

private float restingSize = 75;

private float maxSide = 500;

private float speed = 10;

private float currentSize;

private void Start()

{

reticle = GetComponent<RectTransform>();

}

private void Update()

{

if (isMoving)

{

currentSize = Mathf.Lerp(currentSize, maxSide, Time.deltaTime \* speed);

} else

{

currentSize = Mathf.Lerp(currentSize, restingSize, Time.deltaTime \* speed);

}

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

}

bool isMoving {

get {

if (Input.GetKeyDown(KeyCode.Space) || Input.GetMouseButtonDown(0))

return true;

else

return false;

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Shooting : MonoBehaviour

{

public GameObject projectilePrefab;

private GameObject shootingPoint;

private GameObject mainCam;

private LookAtCrosshair followCrosshair;

void Start()

{

shootingPoint = GameObject.Find("ShootingPoint");

mainCam = GameObject.Find("Main Camera");

followCrosshair = mainCam.GetComponent<LookAtCrosshair>();

}

void Update()

{

// shooting

if (Input.GetKeyDown(KeyCode.Space) || Input.GetMouseButtonDown(0))

{

FireBullet();

}

}

void FireBullet()

{

GameObject pooledProjectile = ObjectPooler.SharedInstance.GetPooledObject();

if (pooledProjectile != null)

{

// Get the target point at the moment of shooting

Vector3 targetPoint = followCrosshair.GetTargetPoint();

// Set the position of the projectile to the shooting point

pooledProjectile.transform.position = shootingPoint.transform.position;

// Calculate the direction from the shooting point to the target point

Vector3 direction = (targetPoint - shootingPoint.transform.position).normalized;

// Set the rotation of the projectile towards the target point

//pooledProjectile.transform.rotation = Quaternion.LookRotation(direction);

// Activate the projectile

pooledProjectile.SetActive(true);

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class SpawnManager : MonoBehaviour

{

public GameObject enemyPrefab;

public int enemyCount;

private GameManager gameManager;

private float spawnRangeX = 25;

private float spawnRangeZ = 19;

public int waveNumber = 1;

void Start()

{

SpawnEnemyWave(waveNumber);

gameManager = GameObject.Find("Game Manager").GetComponent<GameManager>();

}

void Update()

{

enemyCount = FindObjectsOfType<MoveEnemy>().Length;

if (enemyCount == 0)

{

waveNumber ++;

SpawnEnemyWave(waveNumber);

}

}

private void SpawnEnemyWave(int enemiesToSpawn)

{

if (gameManager.isGameActive)

{

for(int i = 0; i < enemiesToSpawn; i++)

{

Instantiate(enemyPrefab, GenerateSpawnPosition(), enemyPrefab.transform.rotation);

}

}

}

private Vector3 GenerateSpawnPosition()

{

float spawnPosX = Random.Range(-spawnRangeX, spawnRangeX);

float spawmPosZ = Random.Range(-spawnRangeZ , spawnRangeZ);

Vector3 randomPos = new Vector3(spawnPosX, 0, spawmPosZ);

return randomPos;

}

}

Current Bugs:

1. The enemy starts following the sphere but then when it bounces on the wall it’s like he lost the target, he stops moving. Furthermore, when he stops the walking animation still continues
2. The shooting system is a bit weird. The crosshair is smooth and the bullets are properly following its center, but they’re not hitting the enemy. He doesn’t get damage. Also, there’s some lagging when pointing at the lower part of the screen.
3. Bugs on the UI. The timer countdown isn’t working and the game is still active while in Game Over, so I can’t pay the game back. May delete the Start screen.

Please help me fix it