Game Programming

Sci-Fi Demo: Lecture II

Weapon Setup

Izzet Fatih Senturk

Weapon Setup

- Drag and drop Weapon prefab from Assets/Game_Models/ to Hierarchy
- We want to put the Weapon in front of us as an FPS game
  - Drag it in front of the Player
- Make the Weapon as the child of the Main Camera
  - Set x, y, z 0 and then move it to the right side of the Player
  - Set z rotation -90
  - Adjust final position and rotation as you like (0.53, -1.35, 3 and 0, 4.12, -90)
- Select Main Camera
  - Set Clipping Planes 0.1

Crosshair

- Add UI -> Image
  - Select sprite: Knob as the source image
  - Set x, y, z: 0
  - Set width, height: 10
  - Set color: light blue
  - Rename: Cross_hair

Hide Cursor

- Hide the mouse cursor and unhide it when the escape key is pressed
  - Hide cursor
  - Set cursor lock mode: Locked

```
void Start()
{
    _controller = GetComponent<CharacterContr
    Cursor.visible = false;
    Cursor.lockState = CursorLockMode.Locked;
}
```

- Unhide the mouse cursor when the escape key is pressed
  - Hide cursor
  - Set cursor lock mode: Locked

```
void Update()
{
    if (Input.GetKeyDown(KeyCode.Escape))
    {
        Cursor.visible = true;
        Cursor.lockState = CursorLockMode.None;
    }
    CalculateMovement();
}
```

Raycasting

- Raycasting casts a ray from point *origin* in direction *direction* of length *maxDistance* against all colliders in the scene
- If mouse left click
  - Cast a ray from center point of the main camera

```
void Update()
{
    if (Input.GetMouseButtonDown(0))
    {
        //Vector3 centerOfTheScreen = new Vector3(Screen.width/2f, Screen.height/2f, 0);
        //Ray rayOrigin = Camera.main.ScreenPointToRay(centerOfTheScreen);
        Vector3 centerOfTheView = new Vector3(0.5f, 0.5f, 0);
        Ray rayOrigin = Camera.main.ViewportPointToRay(centerOfTheView);
        RaycastHit hitInfo;

        if (Physics.Raycast(rayOrigin, out hitInfo))
        {
            Debug.Log("Raycast hit: " + hitInfo.transform.name);
        }
    }
```

Muzzle Flash

- Drag and drop the Muzzle_flash from Assets/Effects to the Hierarchy Window under weapon (as a child)
- Enable Muzzle Game object when the left button click (fire)
- Define a handler variable for the Muzzle and set it through Inspector window

```
[SerializeField]
0 references
private GameObject _muzzleFlash;
```

```
void Update()
{
    if (Input.GetMouseButton(0))
    {
        _muzzleFlash.SetActive(true);
        //Vector3 centerOfTheScreen = r
        //Ray rayOrigin = Camera.main.S
        Vector3 centerOfTheView = new V
        Ray rayOrigin = Camera.main.Vie
        RaycastHit hitInfo;

        if (Physics.Raycast(rayOrigin,
        {
            Debug.Log("Raycast hit: " +
        }
    }
    else
    {
        _muzzleFlash.SetActive(false);
    }
```

Hit Marker

- See what we are shooting
- Define a variable for the Hit Marker prefab and set it through the Inspector Window (Find under Assets/Effects)

```
[SerializeField]
0 references
private GameObject _hitMarkerPrefab;
```

- Instantiate the Hit Marker when Ray Cast collision is detected

```
if (Physics.Raycast(rayOrigin, out hitInfo))
{
    Debug.Log("Raycast hit: " + hitInfo.transform.name);
    Instantiate(_hitMarkerPrefab, hitInfo.point, Quaternion.LookRotation(hitInfo.normal));
}
```

Hit Marker

- Cleaning up Hit Marker instances

```
if (Physics.Raycast(rayOrigin, out hitInfo))
{
    Debug.Log("Raycast hit: " + hitInfo.transform
    GameObject hitMarker = Instantiate(_hitMarker
    Destroy(hitMarker, 1f);
}
```

Weapon Sound

- Select Weapon
  - Add a new component: Audio Source
  - Set AudioClip: Shoot_Sound
  - Turn Play on Awake off and Loop on
- Define AudioSource variable for the Weapon and serialize it
  - Set AudioSource from Inspector. Select Player and drag and drop Weapon on Weapon Audio

```
[SerializeField]
0 references
private AudioSource _weaponAudio;
```

- Play audio at fire
  - And stop audio when fire is off

```
void Update()
{
    if (Input.GetMouseButton(0))
    {
        _muzzleFlash.SetActive(true);
        if (_weaponAudio.isPlaying == false)
        {
            _weaponAudio.Play();
        }
        //Vector3 centerOfTheScreen = new Vec
        //Ray rayOrigin = Camera.main.ScreenP
        Vector3 centerOfTheView = new Vector3
        Ray rayOrigin = Camera.main.ViewportP
        RaycastHit hitInfo;

        if (Physics.Raycast(rayOrigin, out hi
        {
            Debug.Log("Raycast hit: " + hitIn
            GameObject hitMarker = Instantiat
            Destroy(hitMarker, 1f);
        }
    }
    else
    {
        _muzzleFlash.SetActive(false);
        _weaponAudio.Stop();
    }
```

Ammunition

- You can only fire if you have ammo
- Spend ammo at fire and reload ammo when press R key
- Define variables for current ammo and max ammo

```
private int _currentAmmo;
1 reference
private int _maxAmmo = 50;

// Start is called before the
0 references
void Start()
{
    _controller = GetComponent
    Cursor.visible = false;
    Cursor.lockState = CursorL

    _currentAmmo = _maxAmmo;
}
```

Ammunition

- Define a void method for shooting
  - Check the ammo count first

```
void Update()
{
    if (Input.GetMouseButton(0) && _currentAmmo > 0)
    {
        Shoot();
    }
    else
```

```
private void Shoot()
{
    _muzzleFlash.SetActive(true);
    _currentAmmo--;
    if (_weaponAudio.isPlaying == false)
    {
        _weaponAudio.Play();
    }
    //Vector3 centerOfTheScreen = new Vecto
    //Ray rayOrigin = Camera.main.ScreenPoi
    Vector3 centerOfTheView = new Vector3(0
    Ray rayOrigin = Camera.main.ViewportPoi
    RaycastHit hitInfo;

    if (Physics.Raycast(rayOrigin, out hitI
    {
        Debug.Log("Raycast hit: " + hitInfo
        GameObject hitMarker = Instantiate(
        Destroy(hitMarker, 1f);
    }
}
```

Ammunition Reload

- Make sure that you don't reload while reloading
  - Define a Bool variable to check if we are currently reloading

```
private bool _isReloading = false;
```

- Check if R key is pressed at Update

```
if (Input.GetKeyDown(KeyCode.R) && _isReloading == false)
{
    _isReloading = true;
    StartCoroutine(Reload());
}
```

- Define reload coroutine

```
IEnumerator Reload()
{
    yield return new WaitForSeconds(1.5f);
    _currentAmmo = _maxAmmo;
    _isReloading = false;
}
```

Display Ammo

- Create a new C# script: UIManager_sc
- Attach the new script to the Canvas (as a new component)
- Under Canvas, create a new UI –> Text
  - Rename it as Ammo_Text
  - Place it to bottom left corner and anchor it
  - Increase font size
  - Set color: white
  - Set horizontal and vertical overflows
- In the script
  - Add UI library
  - Add a reference to the text component, serialize it and set the value from the Inspector window
  - Define a public method to update ammo count in the new script

Display Ammo

- Update UIManager_sc

```
using UnityEngine;
using UnityEngine.UI;

0 references
public class UIManager_sc : MonoBehaviour
{
    [SerializeField]
    1 reference
    private Text _ammoText;

    0 references
    public void UpdateAmmo(int count)
    {
        _ammoText.text = "Ammo: " + count;
    }
}
```

Display Ammo

- Where do we call UpdateAmmo function?
  - In the Player script, get a reference to UIManager_sc

```
private UIManager_sc _uiManager_sc;
```

- Get UIManager at Start

```
_uiManager_sc = GameObject.Find("Canvas").GetComponent<UIManager_sc>();
if (_uiManager_sc == null)
{
    Debug.LogError("UI Manager is NULL");
}
```

- Update ammo count display. Also call from Reload.

```
private void Shoot()
{
    _muzzleFlash.SetActive(true);
    _currentAmmo--;
    _uiManager_sc.UpdateAmmo(_currentAmmo);
```

```
IEnumerator Reload()
{
    yield return new WaitForSeconds(1.5f);
    _currentAmmo = _maxAmmo;
    _uiManager_sc.UpdateAmmo(_currentAmmo);
    _isReloading = false;
}
```