CREATE GAMES !

Here we are at issue 4 already.

I'll hope you found all previous issues helpful. Before I continue, I will explain to you why I create these magazines. Because it's fun and it is always good to help out people. I am no experienced programmer and my way of doing things might not be the best way but, all my examples and scripts work and that is what new people like me want to see and use. I collect and learn from all the tutorials out there, and every tutorial has his/her way of scripting and saying that their way is the best. For example, there are more ways to script movement, however I don't care which is the best way as long as it works and I can use/adapt it to my own projects and….. it should be easy to understand so I can change it to my likings. I see a lot of tutorials that explain scripts like you understand all the logic in it. In most cases new people learning do not see that logic yet and so it needs to be explained step by step. I do a lot of scripting now very fast without the need of thinking to long about it, it's all about repeating and trying. Now let's read on in this new issue of Create Games! Have fun.

In this issue we will create a 2D rail shooter. A rail shooter is a game where the camera is moving over a premade path. It has a crosshair you can use so you can shoot targets that appear along the way. You can make the camera's journey as long as you need and you can make it move in any direction you like. You can see the whole project being made here : https://www.youtube.com/watch?v=LR1M5J0mT9I&t=229s

Ready let's inspect those scripts. I will explain the parts you are new to as for familiar scripts you should know them if you read the previous issues.

<u>RailCamera Script</u>

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

public class RailCamera : MonoBehaviour
{
///////Rail shooter 2D  Script by René Pol RP-Interactive.nl© 01-07-2021/////////
```

Creating an int variable so you can set the direction the camera should move with a round number.

```
public int MovingDirection;
```

Creating 4 float variables for where the camera should stop. So EndLeft is the position on the x-axis to left and so on.

```
public float EndLeft, EndRight, EndUp, EndDown;
```

Creating the last float variable for the camera movement speed.

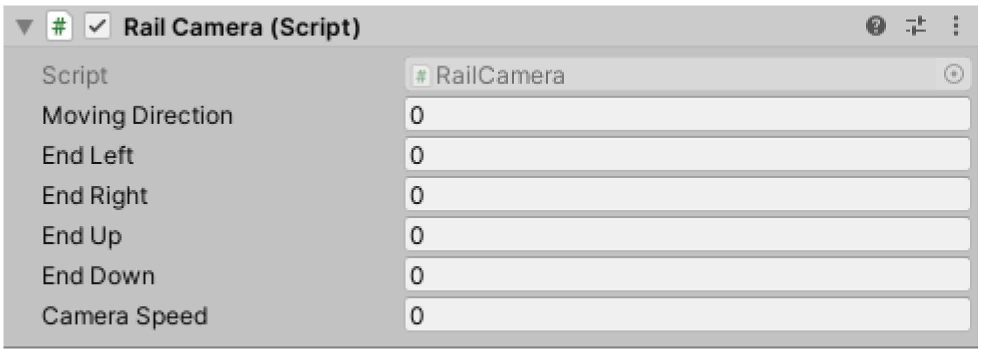
```
public float CameraSpeed;
```

Using 1 till 4 for the movingDirection variable makes the camera move accordingly so when the number is set to 1 the camera will move to the left, 2 is right,3 is up and 4 is down. Using the speed that is given in the Camera Speed variable.

```
void Update()
{
if(MovingDirection == 1)
{
transform.Translate(Vector2.left * CameraSpeed * Time.deltaTime);
}
if (MovingDirection == 2)
{
transform.Translate(Vector2.right * CameraSpeed * Time.deltaTime);
}
if (MovingDirection == 3)
{
transform.Translate(Vector2.up * CameraSpeed * Time.deltaTime);
}
if (MovingDirection == 4)
{
transform.Translate(Vector2.down * CameraSpeed * Time.deltaTime);
}
```

In this example the camera moves to the left when it is smaller than the given EndLeft coordinates the MovingDirection is set to 0 so no movement will happen. The camera is at the end of the rail.

```
if(transform.position.x < EndLeft)
{
MovingDirection = 0;
}
}
}
```



Rail Camera (Script)

Script	RailCamera
Moving Direction	0
End Left	0
End Right	0
End Up	0
End Down	0
Camera Speed	0

Once the script is added to the camera you can give it all the options you need. You can get the endpoints by using the coordinates from the inspector view (Transform positions)

Bullet Script

The bullet is a saved prefab that has a Circle Collider 2D and the nametag Bullet. Is has a script that simply makes the bullet disappear after the seconds you decide it should. This prefab will be used by our crosshair Script. This prefab has also a rigidbody2D with the gravity set to 0 and the constraint Z box selected. You should know by now that for collision at least one of the 2 game objects need a rigidbody2D.

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

public class Bullet : MonoBehaviour
{

///////Rail shooter 2D  Script by René Pol RP-Interactive.nl© 01-07-2021//////////
```

As soon as the bullet sprite appears it will destroy itself after 1 second.

```
void Start()
{
Destroy(this.gameObject, 1f);
}
}
```

Crosshair Script

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

public class Crosshair : MonoBehaviour
{
///////Rail shooter 2D  Script by René Pol RP-Interactive.nl© 01-07-2021//////////
```

Creating a float variable for the bullet number to use.

```
public float BulletAmmo = 25;
```

Defining the game object to use as bullet (Stored in the prefab folder)

```
public GameObject BulletImpact;
```

Defining 2 audio files to use for shooting or having no bullets.

```
public AudioClip Shoot, Empty;
```

Creating a point at the camera so we can make the audio always play near the camera.

```
public Transform Cameraspot;
```

At start we make sure the normal cursor is set to invisible as we use only the crosshair sprite.

```
void Start()
{
Cursor.visible = false;
}
```

A shoot function is created. When the BulletAmmo variable is bigger than zero the audio shoot will play near the camera (Cameraspot). A bullet prefab is created at the position of the crosshair with no rotation used and 1 is subtracted from the BulletAmmo variable. When the BulletAmmo variable is equal or less than 0 it plays the empty sound only.

```
public void ShootBullet()
{
if(BulletAmmo > 0)
{
AudioSource.PlayClipAtPoint(Shoot, new Vector3(Cameraspot.position.x,
Cameraspot.position.y, -10));
Instantiate(BulletImpact, transform.position, Quaternion.identity);
BulletAmmo -= 1;
}
else
{
AudioSource.PlayClipAtPoint(Empty, new Vector3(Cameraspot.position.x,
Cameraspot.position.y, -10));
}
}
```

When the left mouse button is clicked the ShootBullet function will be executed. The transform.position is always screentoworldpoint defined as mouseCursorPos. Bullets will appear wherever is clicked.

```
void Update()
{
if (Input.GetMouseButtonDown(0))
{
ShootBullet();
}
Vector2 mouseCursorPos = Camera.main.ScreenToWorldPoint(Input.mousePosition);
transform.position = mouseCursorPos;
}
}
```

Enemy01 Script

I created a simple enemy sprite with the tag name Enemy and a boxCollider2D When the bullet will hit it it will remove itself and plays the hurt sound.

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

public class Enemy01 : MonoBehaviour
{
///////Rail shooter 2D  Script by René Pol RP-Interactive.nl© 01-07-2021//////////
```

The sound file to use is defined with the name Hurt. We use a transform like you did before so the sound will always play near the camera.

```
public AudioClip Hurt;
public Transform Cameraspot;
```

When collision with an object that has the tag Bullet happens the sound will play near the camera and both objects get destroyed.

```
void OnCollisionEnter2D(Collision2D other)
{
if (other.gameObject.tag == ("Bullet"))
{
AudioSource.PlayClipAtPoint(Hurt, new Vector3(Cameraspot.position.x, Cameraspot.position.y, -10));
Destroy(other.gameObject);
Destroy(this.gameObject);
}
}
}
```

I created 2 UI images. One is called HUD and it is all black except for the part that shows what the player is able to see. The other is a level clear panel that has to show when the end of the level is reached.

In order to make the clear panel visible and invisible I'll add a few lines of script to our already made camera script.

RailCamera Script (Part2)

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

As we use UI elements, we need to insert this library line. If you do not do this the script will not work and give you errors.

```
using UnityEngine.UI;

public class RailCamera : MonoBehaviour
{
///////Rail shooter 2D  Script by René Pol RP-Interactive.nl© 01-07-2021//////////

public int MovingDirection;
public float EndLeft, EndRight, EndUp, EndDown;
public float CameraSpeed;
```

We make a reference to the UI image that needs to be invisible at start.

```
public Image ClearedImage;
```

At start we use SetActive and set it too false. Now this image will not be showing until it is set to true.

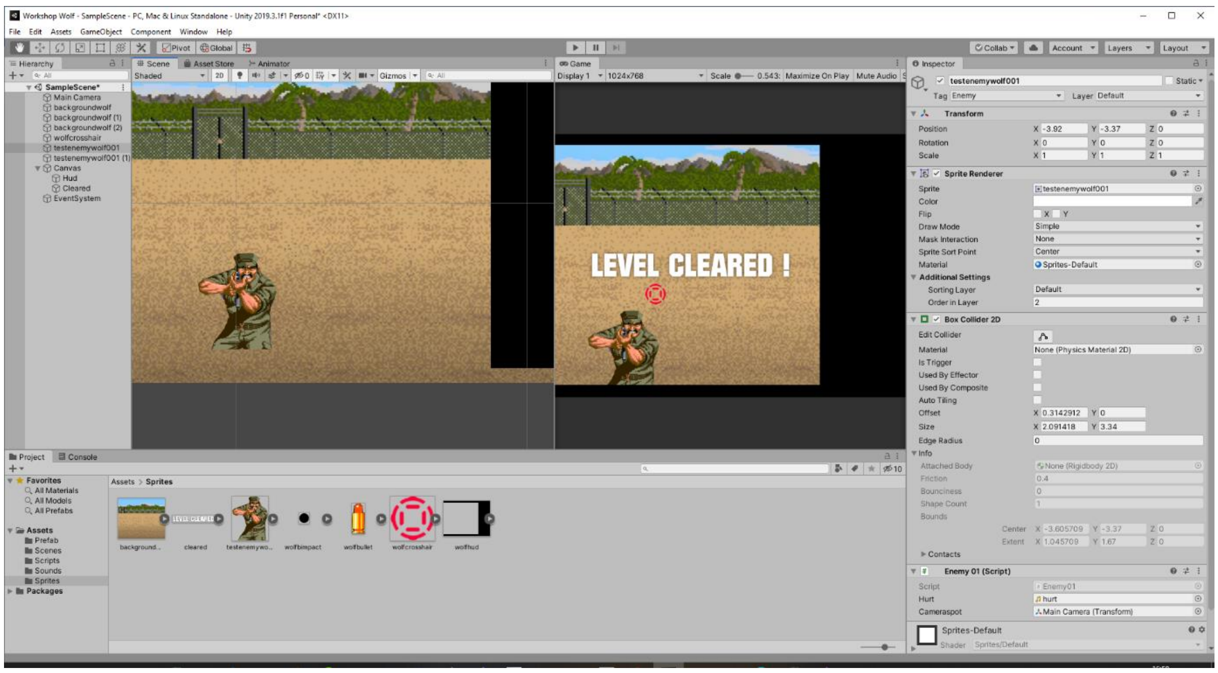
```
void Start()
{
ClearedImage.gameObject.SetActive(false);
}

void Update()
{
if(MovingDirection == 1)
{
transform.Translate(Vector2.left * CameraSpeed * Time.deltaTime);
}
if (MovingDirection == 2)
{
transform.Translate(Vector2.right * CameraSpeed * Time.deltaTime);
}
if (MovingDirection == 3)
{
transform.Translate(Vector2.up * CameraSpeed * Time.deltaTime);
}
if (MovingDirection == 4)
{
transform.Translate(Vector2.down * CameraSpeed * Time.deltaTime);
}
```

When the end of the level is reached, we make the UI image visible.

```
if(transform.position.x < EndLeft)
{
MovingDirection = 0;
ClearedImage.gameObject.SetActive(true);
}
}
}
```

You have now the basics of a rail shooter. All objects that you encounter on the way you can give a script how it should react when hit by the gameObject with the name Tag Bullet. You can add more UI elements and add score for points. It's all up to your imagination.



testenemywolf001
Tag Enemy Layer Default

Transform
Position X -3.92 Y -3.37 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Sprite Renderer
Sprite testenemywolf001
Flip X Y
Draw Mode Simple
Mask Interaction None
Sprite Sort Point Center
Material Sprites-Default
Additional Settings
Sorting Layer Default
Order in Layer 2

Box Collider 2D
Material None (Physics Material 2D)
Offset X 0.3142912 Y 0
Size X 2.091418 Y 3.34
Edge Radius 0
Info
Attached Body None (Rigidbody 2D)
Friction 0.4
Bounciness 0
Shape Count 1
Bounds
Center X -3.605709 Y -3.37 Z 0
Extent X 1.045709 Y 1.67 Z 0

Enemy 01 (Script)
Script Enemy01
Hurt hurt
Cameraspot Main Camera (Transform)

Sprites-Default
Shader Sprites/Default

How To ?

Perhaps you know the game Pokémon Snap. It's a game where the player (Camera) moves over a premade path in a 3D world. Along the way the player can view this world and take pictures of it. Is it hard to make the camera follow a path not only horizontal but also vertical when needed? Not at all. Here is how to do it. You can see how it all works here:
https://www.youtube.com/watch?v=SH5dd9O5WPg

For this example, our world is just a 3d plane. So we need to create a path first so the camera will know how it should move. Place a 3d small sphere (Or any 3d model) it will be our first path point. Then copy it as many times as needed to create a path. Place them how you like Up down small or big distance just create a nice route. Create an empty game object and call it pathParent. Drag all pathpoints to it so they all become children of the pathParent. Now that you created you're path let's do some scripting for it:

FollowPath Script.

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

public class FollowPath : MonoBehaviour
{

////// Camera follows path 3D  René Pol 01-07-2021///////////////////
```

A float variable for speed is created and a Transform is defined to use called pathParent (yes that what we made) another Transform is defined called targetPoint and an int (Round numbers) variable is created called index.

```
public float speed = 0.5f;
public Transform pathParent;
Transform targetPoint;
int index;
```

The OndrawGizmos function will draw a red line (1,0,0) from the first pathpoint to the next and so on. It uses the Vector3 from and to. It will get the pathpoint by using the Transform and then using its children game Objects (pathpoints).

```
void OnDrawGizmos()
{
Vector3 from;
Vector3 to;
for (int a = 0; a < pathParent.childCount; a++)
{
from = pathParent.GetChild(a).position;
to = pathParent.GetChild((a + 1) % pathParent.childCount).position;
Gizmos.color = new Color(1, 0, 0);
Gizmos.DrawLine(from, to);
}
}
```

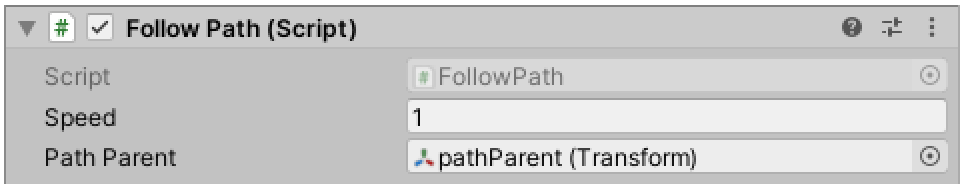
At start the index (All pathpoints) is set to 0 and the first targetPoint(Where to go) is set.

```
void Start()
{
index = 0;
targetPoint = pathParent.GetChild(index);
}
```

The gameobject will move towards the targetPoint and moves to it with the given speed. When it is closer than 0.1 to the targetPoint it will continuously move to the next and so on. It uses all of the Transform children. When the last child object is reached it starts all over.

```
void Update()
{
transform.position = Vector3.MoveTowards(transform.position, targetPoint.position,
speed * Time.deltaTime);
if (Vector3.Distance(transform.position, targetPoint.position) < 0.1f)
{
index++;
index %= pathParent.childCount;
targetPoint = pathParent.GetChild(index);
}
}
}
```

When this script is added to the main camera it will make the camera follow the path points. In the inpectors view you can set the speed and the Transform (Parent with children pathpoints) to use.



Follow Path (Script)

Script	FollowPath
Speed	1
Path Parent	pathParent (Transform)

The camera is now always looking at the targetPoints. As a bonus we will add some look around control with the mouse so it all looks like were in a fun park attraction ride.

LookAroundMouse Script.

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

public class LookAroundMouse : MonoBehaviour
{

////// Camera follows path 3D  René Pol 01-07-2021///////////////////
```

2 Vectors are defined for the mouse control.

```
Vector2 _mouseAbsolute;
Vector2 _smoothMouse;
```

We make sure that the degrees are set so it can't move pass the set degrees.

```
public Vector2 clampInDegrees = new Vector2(360, 180);
```

A bool variable will determine if the Cursor is locked or not. (True or false)

```
public bool lockCursor;
```

the sensitivity and smoothing are set.

```
public Vector2 sensitivity = new Vector2(2, 2);
public Vector2 smoothing = new Vector2(3, 3);
```

2 more Vectors to create target direction and target Character direction (I needed)

```
public Vector2 targetDirection;
public Vector2 targetCharacterDirection;
```

If there's a parent object controlling motion, such as a Character Controller.
Yaw rotation will affect this object instead of the camera if set. This is why we define a private GameObject for it.

```
private  GameObject characterBody;
```

At start the target direction is set to the camera's initial orientation. If needed the target direction for the character body is set to its initial state

```
void Start()
{
targetDirection = transform.localRotation.eulerAngles;
if (characterBody) targetCharacterDirection =
characterBody.transform.localRotation.eulerAngles;
}
```

In the Update function we make sure the cursor is locked when set and we allow the desired target value based on the clamp script. Mouse input is raw for a more sensitive mice movement.

```
void Update()
{
Screen.lockCursor = lockCursor;
var targetOrientation = Quaternion.Euler(targetDirection);
var targetCharacterOrientation = Quaternion.Euler(targetCharacterDirection);
var mouseDelta = new Vector2(Input.GetAxisRaw("Mouse X"), Input.GetAxisRaw("Mouse
Y"));
```

Scale input and sensitivity setting with the smoothing value. The movement uses smoothMouse for good movement and uses absolute mouse movement value for point zero.

```
mouseDelta = Vector2.Scale(mouseDelta, new Vector2(sensitivity.x * smoothing.x,
sensitivity.y * smoothing.y));
_smoothMouse.x = Mathf.Lerp(_smoothMouse.x, mouseDelta.x, 1f / smoothing.x);
_smoothMouse.y = Mathf.Lerp(_smoothMouse.y, mouseDelta.y, 1f / smoothing.y);
_mouseAbsolute += _smoothMouse;
```

Clamp and apply local x value so it's not affected by world transforms.

```
if (clampInDegrees.x < 360)
_mouseAbsolute.x = Mathf.Clamp(_mouseAbsolute.x, -clampInDegrees.x * 0.5f,
clampInDegrees.x * 0.5f);
var xRotation = Quaternion.AngleAxis(-_mouseAbsolute.y, targetOrientation *
Vector3.right);
transform.localRotation = xRotation;
```

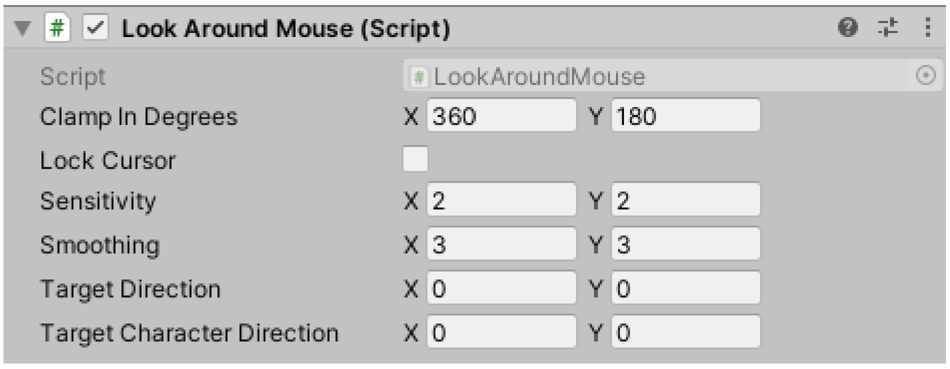
Clamp and apply the global y value.

```
if (clampInDegrees.y < 360)
_mouseAbsolute.y = Mathf.Clamp(_mouseAbsolute.y, -clampInDegrees.y * 0.5f,
clampInDegrees.y * 0.5f);
transform.localRotation *= targetOrientation;
```

If there is a character body that acts as a parent to the camera we use:

```
if (characterBody)
{
var yRotation = Quaternion.AngleAxis(_mouseAbsolute.x,
characterBody.transform.up);
characterBody.transform.localRotation = yRotation;
characterBody.transform.localRotation *= targetCharacterOrientation;
}
else
{
var yRotation = Quaternion.AngleAxis(_mouseAbsolute.x,
transform.InverseTransformDirection(Vector3.up));
transform.localRotation *= yRotation;
}
}
}
```

Simply add this script to the main camera to. You can change the settings of it to your likings in the Inspectors view. Now you can look around during the ride on the path.



Look Around Mouse (Script)

Script	LookAroundMouse			
Clamp In Degrees	X	360	Y	180
Lock Cursor	☐			
Sensitivity	X	2	Y	2
Smoothing	X	3	Y	3
Target Direction	X	0	Y	0
Target Character Direction	X	0	Y	0

You have now a camera that will move over a path you create and you can look around during this movement. You can use this any way you want. Imagine shooting or taking pictures while moving on this path. It's all up to your imagination. I'll hope to meet you all in the next issue. Have fun!

All of the assets used in this issue can be downloaded here: https://rp-interactive.nl/magazine.html

All scripts used in issue 04 are here for you.

RailCameraScript.

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

public class RailCamera : MonoBehaviour
{
///////Rail shooter 2D  Script by René Pol RP-Interactive.nl© 01-07-2021//////////

public int MovingDirection;
public float EndLeft, EndRight, EndUp, EndDown;
public float CameraSpeed;
public Image ClearedImage;

void Start()
{
ClearedImage.gameObject.SetActive(false);
}

void Update()
{
if(MovingDirection == 1)
{
transform.Translate(Vector2.left * CameraSpeed * Time.deltaTime);
}
if (MovingDirection == 2)
{
transform.Translate(Vector2.right * CameraSpeed * Time.deltaTime);
}
if (MovingDirection == 3)
{
transform.Translate(Vector2.up * CameraSpeed * Time.deltaTime);
}
if (MovingDirection == 4)
{
transform.Translate(Vector2.down * CameraSpeed * Time.deltaTime);
}
if(transform.position.x < EndLeft)
{
MovingDirection = 0;
ClearedImage.gameObject.SetActive(true);
}
}
}
```

Bullet Script

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

public class Bullet : MonoBehaviour
{
///////Rail shooter 2D  Script by René Pol RP-Interactive.nl© 01-07-2021/////////

void Start()
{
Destroy(this.gameObject, 1f);
}
}
```

CrossHair Script.

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

public class Crosshair : MonoBehaviour
{
///////Rail shooter 2D  Script by René Pol RP-Interactive.nl© 01-07-2021/////////

public float BulletAmmo = 25;
public GameObject BulletImpact;
public AudioClip Shoot, Empty;
public Transform Cameraspot;

void Start()
{
Cursor.visible = false;
}

public void ShootBullet()
{
if(BulletAmmo > 0)
{
AudioSource.PlayClipAtPoint(Shoot, new Vector3(Cameraspot.position.x,
Cameraspot.position.y, -10));
Instantiate(BulletImpact, transform.position, Quaternion.identity);
BulletAmmo -= 1;
}
else
{
AudioSource.PlayClipAtPoint(Empty, new Vector3(Cameraspot.position.x,
Cameraspot.position.y, -10));
}
}
```

```
void Update()
{
if (Input.GetMouseButtonDown(0))
{
ShootBullet();
}
Vector2 mouseCursorPos = Camera.main.ScreenToWorldPoint(Input.mousePosition);
transform.position = mouseCursorPos;
}
}
```

Enemy01 Script.

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

public class Enemy01 : MonoBehaviour
{
///////Rail shooter 2D  Script by René Pol RP-Interactive.nl© 01-07-2021/////////

public AudioClip Hurt;
public Transform Cameraspot;

void OnCollisionEnter2D(Collision2D other)
{
if (other.gameObject.tag == ("Bullet"))
{
AudioSource.PlayClipAtPoint(Hurt, new Vector3(Cameraspot.position.x,
Cameraspot.position.y, -10));
Destroy(other.gameObject);
Destroy(this.gameObject);
}
}
}
```

Follow Path Script.

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

public class FollowPath : MonoBehaviour
{

////// Camera follows path 3D  René Pol 01-07-2021//////////////////

public float speed = 0.5f;
public Transform pathParent;
Transform targetPoint;
int index;
```

```
void OnDrawGizmos()
{
Vector3 from;
Vector3 to;
for (int a = 0; a < pathParent.childCount; a++)
{
from = pathParent.GetChild(a).position;
to = pathParent.GetChild((a + 1) % pathParent.childCount).position;
Gizmos.color = new Color(1, 0, 0);
Gizmos.DrawLine(from, to);
}
}

void Start()
{
index = 0;
targetPoint = pathParent.GetChild(index);
}

void Update()
{
transform.position = Vector3.MoveTowards(transform.position, targetPoint.position,
speed * Time.deltaTime);
if (Vector3.Distance(transform.position, targetPoint.position) < 0.1f)
{
index++;
index %= pathParent.childCount;
targetPoint = pathParent.GetChild(index);
}
}
}
```

LookAroundMouse Script.

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

public class LookAroundMouse : MonoBehaviour
{

////// Camera follows path 3D  René Pol 01-07-2021///////////////////

Vector2 _mouseAbsolute;
Vector2 _smoothMouse;

public Vector2 clampInDegrees = new Vector2(360, 180);
public bool lockCursor;
public Vector2 sensitivity = new Vector2(2, 2);
public Vector2 smoothing = new Vector2(3, 3);
public Vector2 targetDirection;
public Vector2 targetCharacterDirection;
private  GameObject characterBody;
```

```
void Start()
{
targetDirection = transform.localRotation.eulerAngles;
if (characterBody) targetCharacterDirection =
characterBody.transform.localRotation.eulerAngles;
}

void Update()
{
Screen.lockCursor = lockCursor;
var targetOrientation = Quaternion.Euler(targetDirection);
var targetCharacterOrientation = Quaternion.Euler(targetCharacterDirection);
var mouseDelta = new Vector2(Input.GetAxisRaw("Mouse X"), Input.GetAxisRaw("Mouse
Y"));
mouseDelta = Vector2.Scale(mouseDelta, new Vector2(sensitivity.x * smoothing.x,
sensitivity.y * smoothing.y));
_smoothMouse.x = Mathf.Lerp(_smoothMouse.x, mouseDelta.x, 1f / smoothing.x);
_smoothMouse.y = Mathf.Lerp(_smoothMouse.y, mouseDelta.y, 1f / smoothing.y);
_mouseAbsolute += _smoothMouse;
if (clampInDegrees.x < 360)
_mouseAbsolute.x = Mathf.Clamp(_mouseAbsolute.x, -clampInDegrees.x * 0.5f,
clampInDegrees.x * 0.5f);
var xRotation = Quaternion.AngleAxis(-_mouseAbsolute.y, targetOrientation *
Vector3.right);
transform.localRotation = xRotation;
if (clampInDegrees.y < 360)
_mouseAbsolute.y = Mathf.Clamp(_mouseAbsolute.y, -clampInDegrees.y * 0.5f,
clampInDegrees.y * 0.5f);
transform.localRotation *= targetOrientation;
if (characterBody)
{
var yRotation = Quaternion.AngleAxis(_mouseAbsolute.x,
characterBody.transform.up);
characterBody.transform.localRotation = yRotation;
characterBody.transform.localRotation *= targetCharacterOrientation;
}
else
{
var yRotation = Quaternion.AngleAxis(_mouseAbsolute.x,
transform.InverseTransformDirection(Vector3.up));
transform.localRotation *= yRotation;
}
}
}
```