Universidad Nacional de San Agustin
Facultad de Ingenieria, Produccion y Servicios
Escuela Profesional de Ciencia de la Computacion

CONTROLADORES DE ENTRADA DE PRIMERA Y TERCERA PERSONA

Integrantes: Kelvin Paul Pucho Zevallos

Profesor: Diego Alonso Iquira Becerra

Fecha de realización: 23 de mayo de 2022
Fecha de entrega: 23 de mayo de 2022

Arequipa - Perú

Índice de Contenidos

Índice de Figuras

Índice de Códigos

1. Creación de Objetos

La escena SceneMain cuenta con 3 camaras, 2 collider para cambiar de primera persona a tercera persona o viceversa. Con Towers para considerar un efecto de collider cuando el obstáculo es sobrepuesto en la camara. Y un objeto capsula que sera mi player.

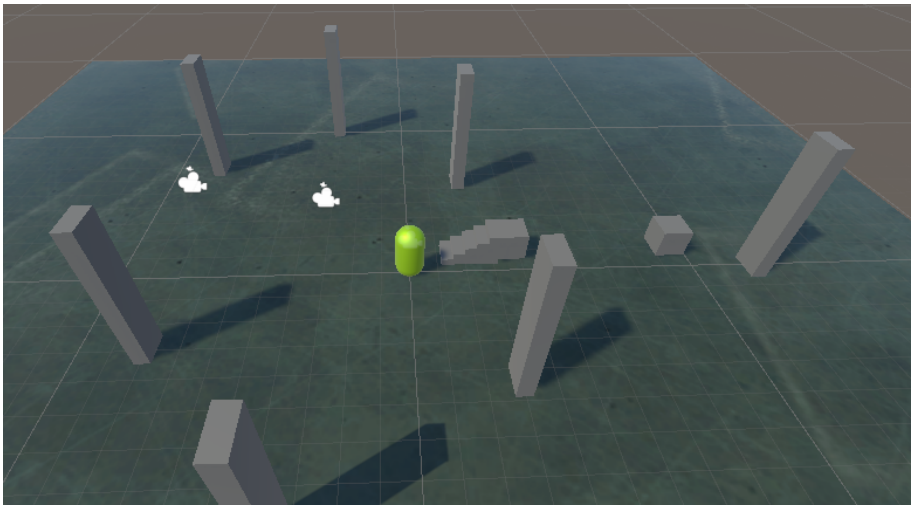

Figura 1: Escenario

1.1. Vista de Juego

La cámara primero enfoca a la cápsula como tercera persona, cuando el usuario se encuentra con un collider la cámara cambiara a primera persona o a una cámara fija.

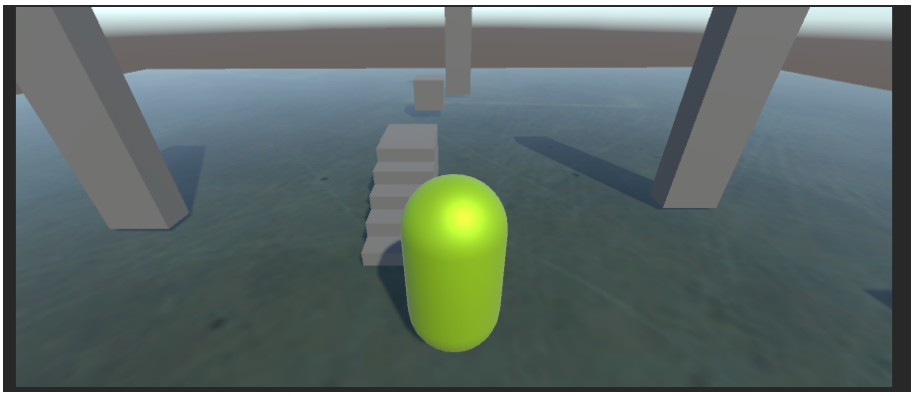

Figura 2: Vista del Juego

La escena contara con las tres cámaras como se ve en la imagen siguiente:

Figura 3: Camaras

La segunda cámara utilizo el componente de character controller para interactuar con un collider.

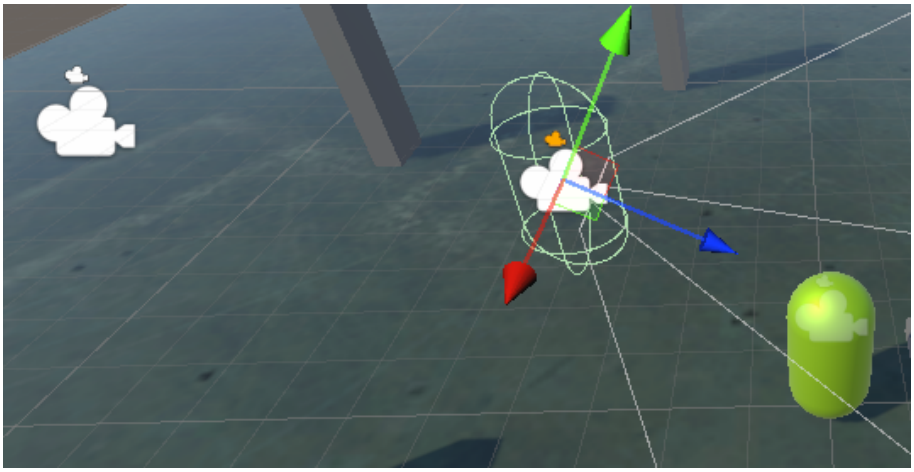

Figura 4: Camaras con Character Controller

1.2. Hierarchy

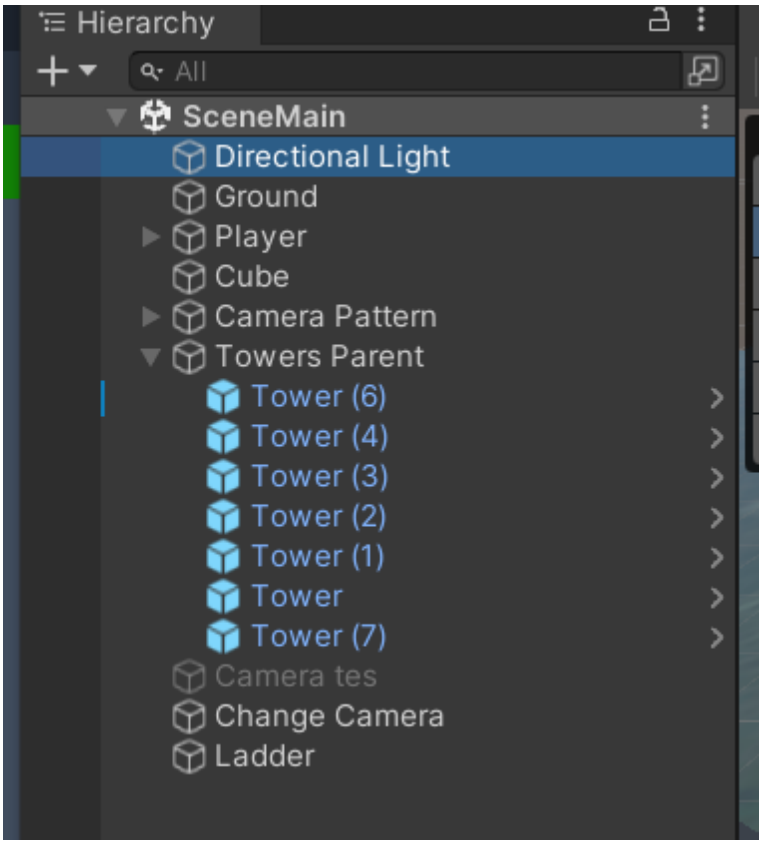

Figura 5: Hierarchy

Los objetos son:

- Ground
- Player: Capsula
- Cameras: Fixed Camera, Main Camera y Change Camera
- Towers(prefabs)
- Ladder

2. Assets

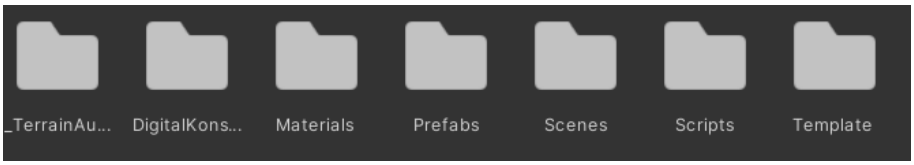

Figura 6: Assets

3. Materials

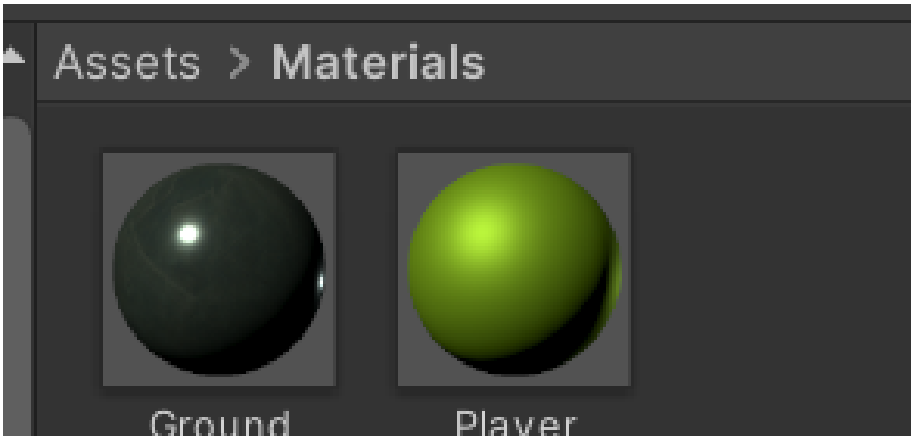

Figura 7: Materials

4. Inspector del Objeto Player

- **Inspector del Player** El borde azul indica las variables publicas que se declara en la clase CapsuleController.

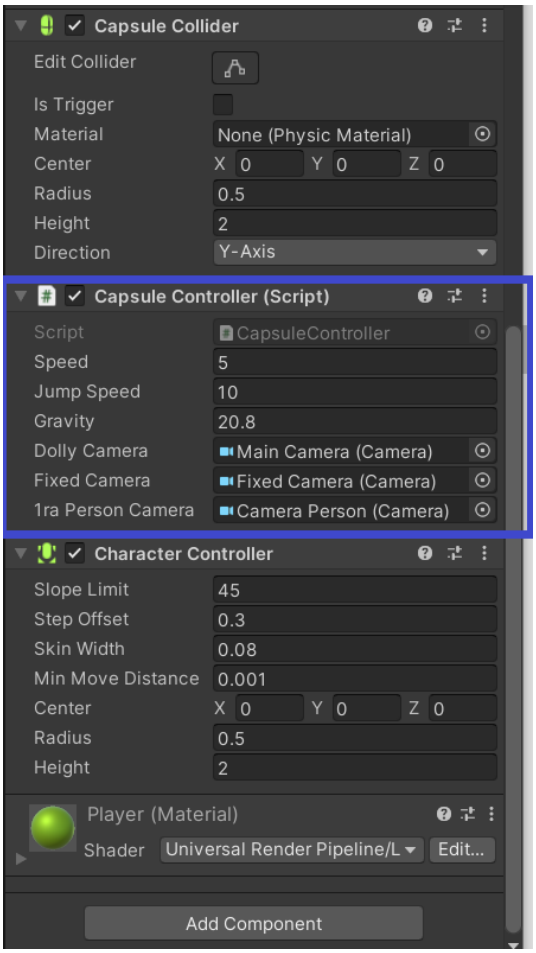

Figura 8: Capsule Collider, Capsule Collider script, Character Controller

- **Inspector de la Main Camera**

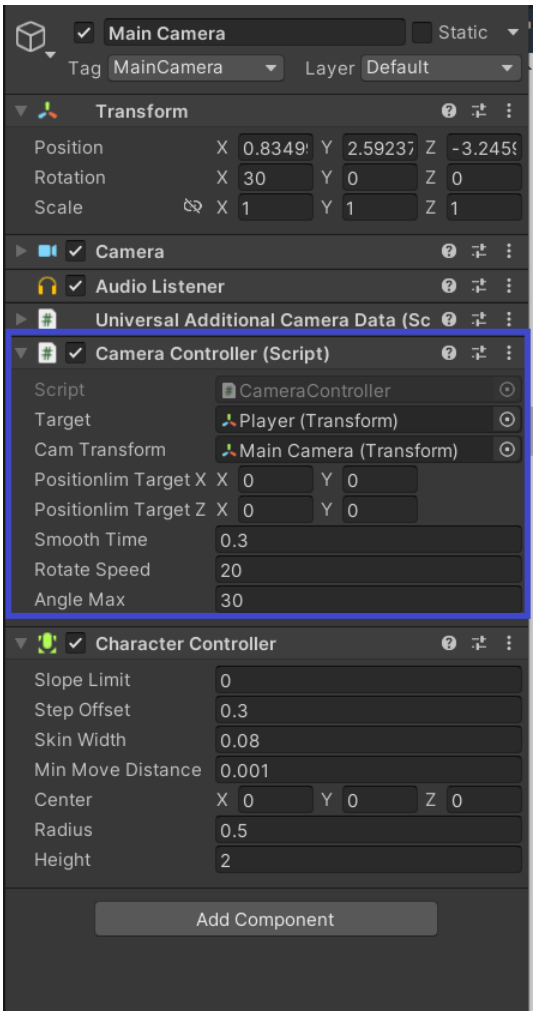

Figura 9: Camera Controller (script), Character Controller

Las variables publicas que son de tipo float son para rotar con velocidad y cierto la cámara cuando esta no enfoca al player (Capsule)

5. Codigos

5.1. Capsule Controller

Este código controla los movimientos de mi objeto capsula ademas de que posee funciones que detectan el evento cuando mi objeto colisiona con otro.

Código 1: CapsuleController.cs

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
```

```
5  public class CapsuleController : MonoBehaviour
6  {
7      public float speed = 5.0f;
8      public float jumpSpeed = 5.0f;
9      public float gravity = 20.8f;
10     private Vector3 moveDirection;
11     private CharacterController chCtrl;
12
13     // cameras
14     public Camera dollyCamera;
15     public Camera fixedCamera;
16     public Camera _1raPersonCamera;
17
18
19     // Start is called before the first frame update
20     void Start()
21     {
22         chCtrl = GetComponent<CharacterController>();
23     }
24
25     // Update is called once per frame
26     void FixedUpdate()
27     {
28         if (chCtrl.isGrounded)
29         {
30             moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0.0f, Input.GetAxis("Vertical"));
31             moveDirection = transform.TransformDirection(moveDirection);
32             moveDirection *= speed;
33             if (Input.GetButton("Jump"))
34             {
35                 moveDirection.y = jumpSpeed;
36             }
37         }
38         moveDirection.y -= (gravity * Time.deltaTime);
39         chCtrl.Move(moveDirection * Time.deltaTime);
40     }
41
42     private void OnTriggerEnter(Collider other)
43     {
44         if (other.gameObject.CompareTag("Test"))
45         {
46
47             SwitchOfCamera();
48         }
49         if (other.gameObject.CompareTag("ChangeCamera"))
50         {
51             Debug.Log("change 1ra person");
52             SwitchOfCamera1raPerson();
53         }
54     }
55
```

```
56  private void SwitchOfCamera1raPerson()
57  {
58      if (dollyCamera.enabled)
59      {
60          Debug.Log("change Camera of 1ra Person");
61          // fixed camera
62          _1raPersonCamera.enabled = true;
63          _1raPersonCamera.GetComponent<AudioListener>().enabled = true;
64
65          // dolly camera
66          dollyCamera.enabled = false;
67          dollyCamera.GetComponent<AudioListener>().enabled = false;
68
69      }
70
71      else if (_1raPersonCamera.enabled)
72      {
73          Debug.Log("change Dolly Camera");
74
75          // dolly camera
76          dollyCamera.enabled = true;
77          dollyCamera.GetComponent<AudioListener>().enabled = true;
78
79          // fixed camera
80          _1raPersonCamera.enabled = false;
81          _1raPersonCamera.GetComponent<AudioListener>().enabled = false;
82
83      }
84
85  }
86
87  private void SwitchOfCamera()
88  {
89      if (dollyCamera.enabled)
90      {
91          Debug.Log("change Fixed Camera");
92          // fixed camera
93          fixedCamera.enabled = true;
94          fixedCamera.GetComponent<AudioListener>().enabled = true;
95
96          // dolly camera
97          dollyCamera.enabled = false;
98          dollyCamera.GetComponent<AudioListener>().enabled = false;
99
100     }
101
102     else if (fixedCamera.enabled)
103     {
104         Debug.Log("change Dolly Camera");
105
106         // dolly camera
```

```
107             dollyCamera.enabled = true;
108             dollyCamera.GetComponent<AudioListener>().enabled = true;
109 
110             // fixed camera
111             fixedCamera.enabled = false;
112             fixedCamera.GetComponent<AudioListener>().enabled = false;
113 
114         }
115 
116 
117     }
118 
119 }
```

- Variables de Cámaras declaradas
- Start: iniciar la variable de tipo Character Controller
- Fixed Update : En cada frame se realiza el desplazamiento y si el usuario realiza el salto del objeto se agrega el movimiento al eje y para luego caer de acuerdo a la gravedad.
- OnTriggerEnter: Para detectar las colisiones
- SwitchOfCamera y SwitchOfCamera1raPerson: Funciones llamadas a partir de las colisiones para cambiar el modo de cámara.

5.2. Camera Controller

Este código es para que la cámara siga al objeto definido mediante la diferencia de sus posiciones para hallar su distancia. Y cuando detecta una colisión que obstaculice la vista del usuario genere una mejor vista.

Código 2: CameraController.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  
5  public class CameraController : MonoBehaviour
6  {
7      //camera will follow this target
8      public Transform target;
9      //camera transform
10     public Transform camTransform;
11     //offset between camera and target
12     private Vector3 offset;
13     private Vector3 oldCameraPosition;
14     private Vector3 oldTargetPosition;
15     private Vector3 oldoffset;
16 
17     public Vector2 positionlimTargetX;
```

```
18    public Vector2 positionlimTargetZ;
19    //change this value to get desired smoothness
20    public float smoothTime = 0.3f;
21    // this value will change at the runtime depending on target movement
22    private Vector3 velocity = Vector3.zero;
23
24
25    // translate camera
26    private bool cameraColision = false;
27    private bool cameraMove = true;
28
29    private float rotationSpeed = -60f;
30    private float oldRotationAngle;
31
32    public float rotateSpeed = 20.0f;
33    public float angleMax = 30.0f;
34
35    private Vector3 initialVector = Vector3.forward;
36
37
38    // Start is called before the first frame update
39    void Start()
40    {
41        offset = camTransform.position - target.position;
42    }
43
44    // Update is called once per frame
45    void LateUpdate()
46    {
47        //change rotation
48
49        if (cameraColision)
50        {
51
52            float rotateDegrees = 0f;
53            rotateDegrees -= rotateSpeed * Time.deltaTime;
54            Vector3 currentVector = transform.position - target.position;
55            currentVector.y = 0;
56            float angleBetween = Vector3.Angle(initialVector, currentVector) * (Vector3.Cross(
      ↪ initialVector, currentVector).y > 0 ? 1 : -1);
57            float newAngle = Mathf.Clamp(angleBetween + rotateDegrees, -angleMax, angleMax);
58            rotateDegrees = newAngle - angleBetween;
59            Debug.Log(rotateDegrees);
60            transform.RotateAround(target.position, Vector3.up, rotateDegrees* Time.deltaTime);
61            if (rotateDegrees < 100)
62            {
63                cameraColision = false;
64                cameraMove = false;
65                offset = camTransform.position - target.position;
66            }
67        }
```

```
68      else
69      {
70          if (cameraMove)
71          {
72              Vector3 targetPosition = target.position + offset;
73              positionlimTargetX = new Vector2(target.position.x - 2.0f, target.position.x + 2.0f);
74              positionlimTargetZ = new Vector2(0.0f, target.position.x + 2.0f);
75
76
77              camTransform.position = Vector3.SmoothDamp(transform.position, targetPosition, ref
    ↪ velocity, smoothTime);
78              oldCameraPosition = camTransform.position;
79              oldTargetPosition = target.position;
80              oldoffset = offset;
81              var angle1 = Vector3.Angle(transform.position, target.position);
82              oldRotationAngle = transform.rotation.eulerAngles.x;
83          }
84
85          else
86          {
87
88              if((target.position.x >= positionlimTargetX.y || target.position.x <= positionlimTargetX.
    ↪ x) || target.position.z >= positionlimTargetZ.y)
89              {
90                  Debug.Log("retornando posiiton de camara");
91                  Vector3 l = oldCameraPosition - oldTargetPosition; //new offset
92
93                  camTransform.position = target.position + l;
94
95                  cameraMove = true;
96              }
97          }
98      }
99
100 }
101
102 private void OnTriggerEnter(Collider other)
103 {
104
105     if (other.gameObject.CompareTag("TowerCollader"))
106     {
107         Debug.Log("camara chocando");
108         cameraColision = true;
109
110     }
111
112 }
113 }
```

- Variables declaradas:

Código 3: Variables

```
//camera will follow this target
public Transform target;
//camera transform
public Transform camTransform;
//offset between camera and target
private Vector3 offset;
private Vector3 oldCameraPosition;
private Vector3 oldTargetPosition;
private Vector3 oldoffset;

public Vector2 positionlimTargetX;
public Vector2 positionlimTargetZ;
//change this value to get desired smoothness
public float smoothTime = 0.3f;
// this value will change at the runtime depending on target movement
private Vector3 velocity = Vector3.zero;


// translate camera
private bool cameraColision = false;
private bool cameraMove = true;

private float rotationSpeed = -60f;
private float oldRotationAngle;

public float rotateSpeed = 20.0f;
public float angleMax = 30.0f;

private Vector3 initialVector = Vector3.forward;
```

Las variables que se necesitan son las referencias al objeto y a la camara en si. Pero también use variables que contiene el Old como prefijos para almacenar su posición de un punto cuando este se mueva. Para que la camara tenga un movimiento mas suave y mas real use el metodo SmoothDamp de la clase del Vector3.
A partir del código comentado (translate camera) las variables que se usan son para poder rotar la camara.

- Funcion LateUpdate:

Código 4: LateUpdate

```
void LateUpdate()
{
   //change rotation

   if (cameraColision)
   {

      float rotateDegrees = 0f;
      rotateDegrees -= rotateSpeed * Time.deltaTime;
```

```
10          Vector3 currentVector = transform.position - target.position;
11          currentVector.y = 0;
12          float angleBetween = Vector3.Angle(initialVector, currentVector) * (Vector3.Cross(
        ↪ initialVector, currentVector).y > 0 ? 1 : -1);
13          float newAngle = Mathf.Clamp(angleBetween + rotateDegrees, -angleMax, angleMax);
14          rotateDegrees = newAngle - angleBetween;
15          Debug.Log(rotateDegrees);
16          transform.RotateAround(target.position, Vector3.up, rotateDegrees* Time.deltaTime);
17          if (rotateDegrees < 100)
18          {
19              cameraColision = false;
20              cameraMove = false;
21              offset = camTransform.position - target.position;
22          }
23      }
24      else
25      {
26          if (cameraMove)
27          {
28              Vector3 targetPosition = target.position + offset;
29              positionlimTargetX = new Vector2(target.position.x - 2.0f, target.position.x + 2.0f);
30              positionlimTargetZ = new Vector2(0.0f, target.position.x + 2.0f);
31
32
33              camTransform.position = Vector3.SmoothDamp(transform.position, targetPosition, ref
        ↪  velocity, smoothTime);
34              oldCameraPosition = camTransform.position;
35              oldTargetPosition = target.position;
36              oldoffset = offset;
37              var angle1 = Vector3.Angle(transform.position, target.position);
38              oldRotationAngle = transform.rotation.eulerAngles.x;
39          }
40
41          else
42          {
43
44              if((target.position.x >= positionlimTargetX.y || target.position.x <=
        ↪ positionlimTargetX.x) || target.position.z >= positionlimTargetZ.y)
45              {
46                  Debug.Log("retornando posiiton de camara");
47                  Vector3 l = oldCameraPosition - oldTargetPosition; //new offset
48
49                  camTransform.position = target.position + l;
50
51                  cameraMove = true;
52              }
53          }
54      }
55
56  }
```

Este código contiene una condición que separa en dos bloques cuando la cámara colisiona y cuando no colisiona:

- Cuando Colisiona: La cámara en cada frame cuando colisiona en este caso con un objeto tower lo que realiza es obtener la posición de la cámara respecto al objeto y el grado de rotación. Para obtener cuando de ángulo se quiere que rote la cámara se realiza los procedimiento que se ve en el código como es el Vector3.Angle y el Mathf.Clamp. Para obtener el grado de rotación y posteriormente se establece en el transform de la cámara en el método RotateAround. La condición es menor a 100 es cuando se desea que la cámara rote hasta cierto punto para luego establecer la condición de colisión en falso, como también la cámara en movimiento y volver a ubicar la nueva posición de la cámara respecto al objeto.
- Cuando no colisiona:
  Se tiene otras dos condiciones:
  * Cuando la cámara esta en movimiento en tercera persona se usa el método SmoothDamp pero se guarda las posiciones actuales de cada objeto y cámara.
  * Cuando la cámara no esta en movimiento significa que la rotación ya se genero por ello si el usuario avanza una cierta posición retorna la posición de la cámara en tercera persona, y se actualiza la nueva posición de la cámara basándose en las posiciones guardadas.

- Funcion OnTriggerEnter:

Código 5: OnTriggerEnter

```
private void OnTriggerEnter(Collider other)
{

   if (other.gameObject.CompareTag("TowerCollader"))
   {
      Debug.Log("camara chocando");
      cameraColision = true;

   }

}
```

Este método captura el objeto con cual colisiona la cámara y establece la variable cameraColision en true para rotar la camara.

Referencias

- **Grabación de Explicación:** https://drive.google.com/file/d/1VUxeZ7BQB31JFIKUiDOs6sm8AqiXilI6/view?usp=sharing
- https://www.youtube.com/watch?v=_QajrabyTJc
- https://docs.unity3d.com/ScriptReference/Vector3.Cross.html
- https://forum.unity.com/threads/third-person-camera-movement-script.858673/
- https://answers.unity.com/questions/1352954/how-to-limit-the-angle-of-a-camera-1.html
- https://answers.unity.com/questions/438836/limit-camera-rotation-with-rotatearound.html
- https://answers.unity.com/questions/1528262/how-can-i-change-camera-when-colliding-with-x-obje.html
- https://forum.unity.com/threads/switching-cameras-using-a-collider.528036/