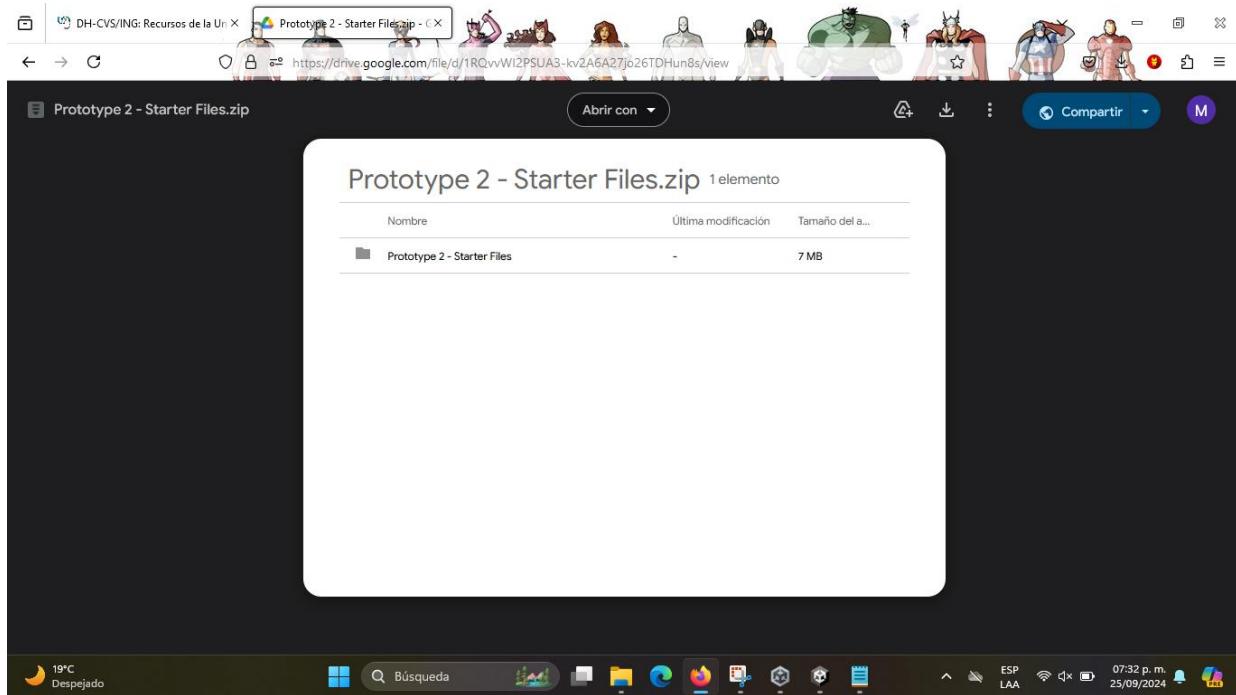


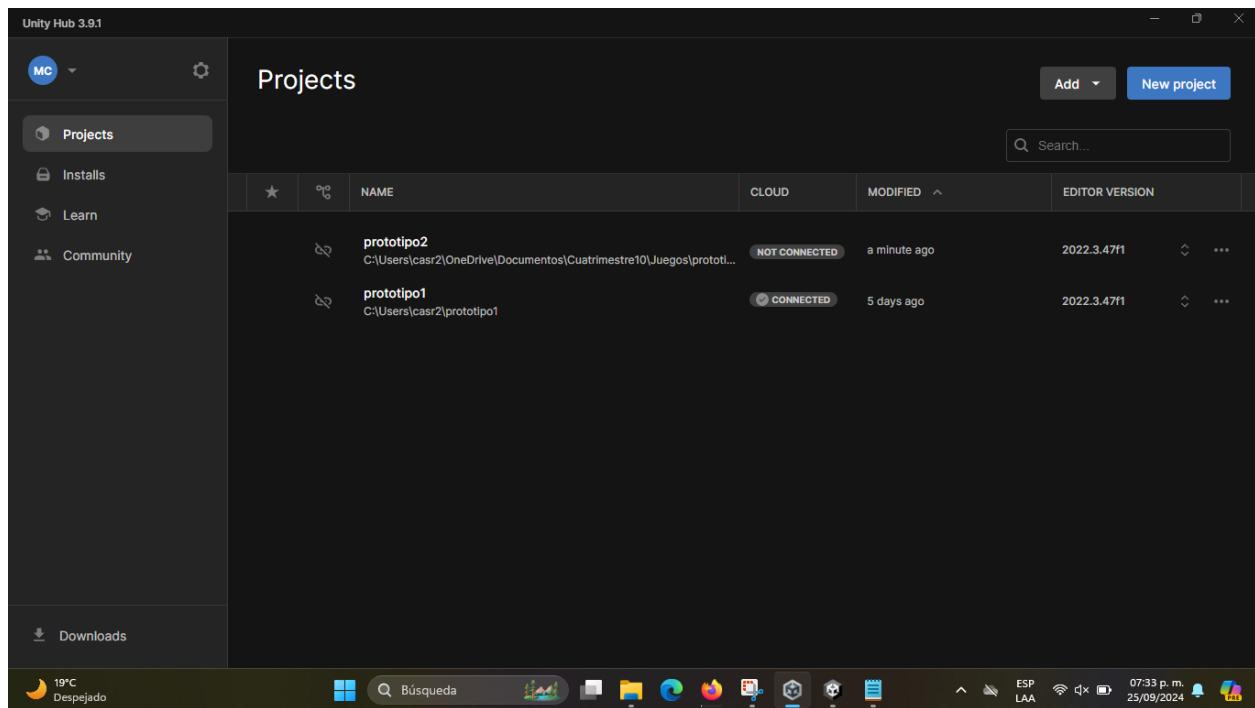
PROTOTIPO 2

Alimentar a los animales

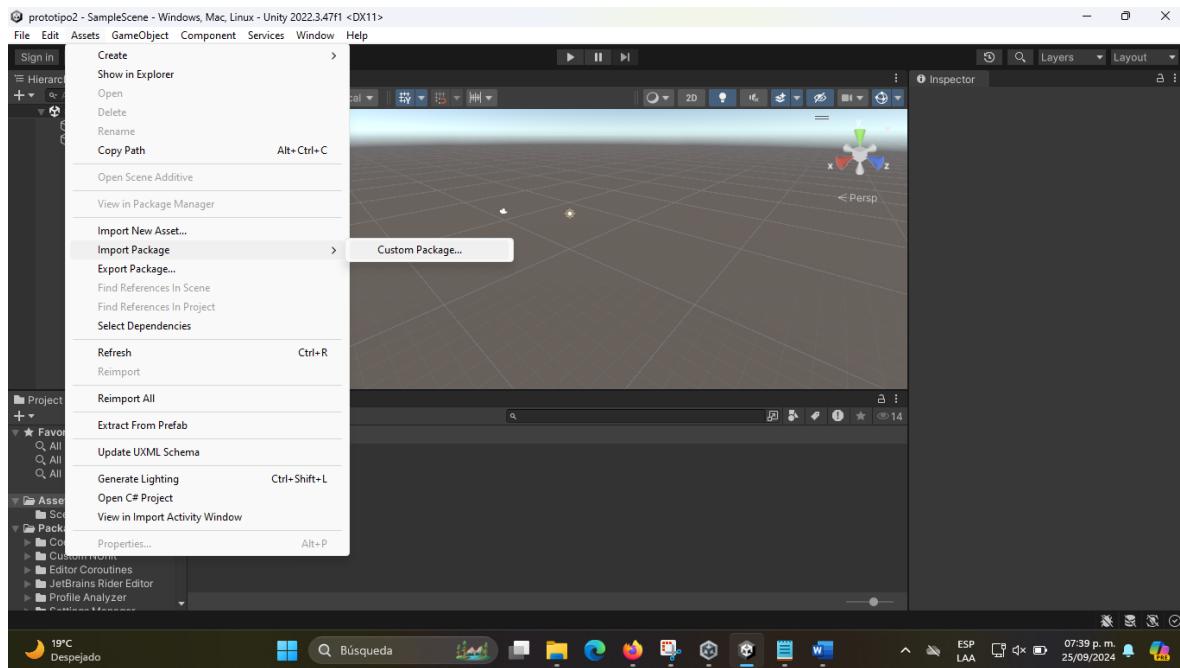
Primero se tiene que descargar el paquete proporcionado por el profesor, se descomprime y se almacena en archivos.



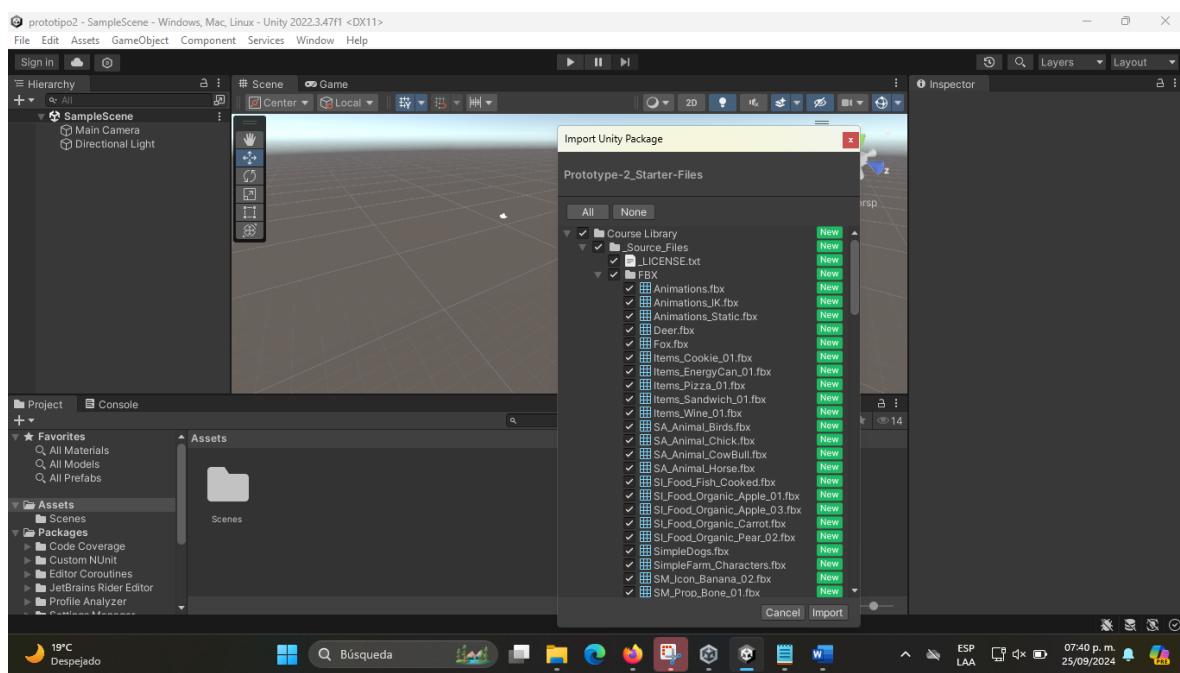
Se crea el proyecto en Unity Hub, nombrado como prototipo2



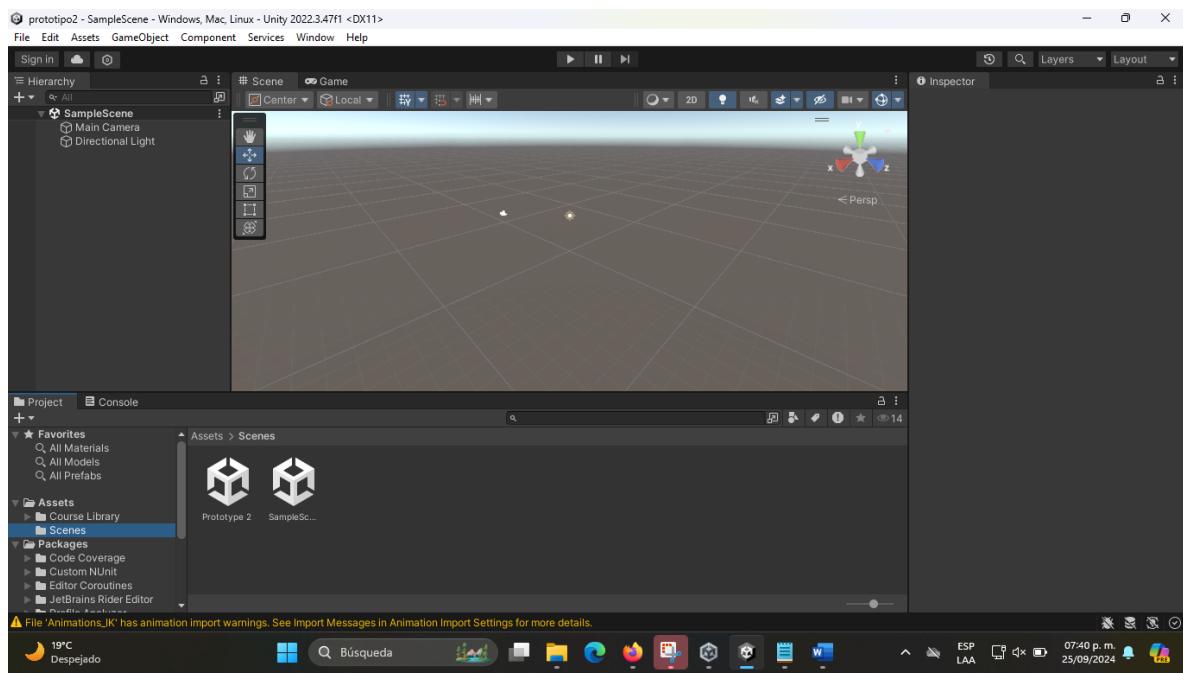
Una vez que se crea el proyecto se **importa** el paquete desde >**Assets > Import Package > Custom Package**.



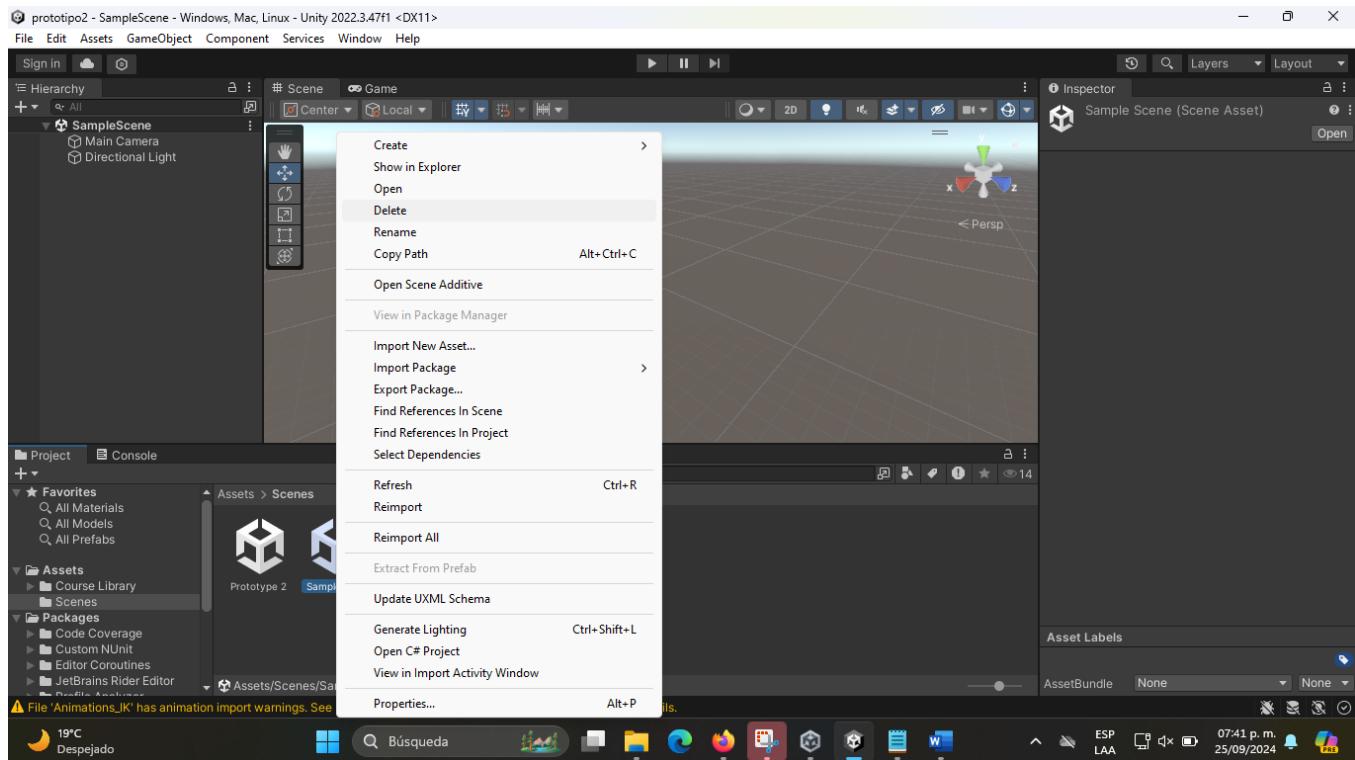
Selecciona el archivo que se descomprimió anteriormente, das clic en **Import** de la siguiente pantalla.



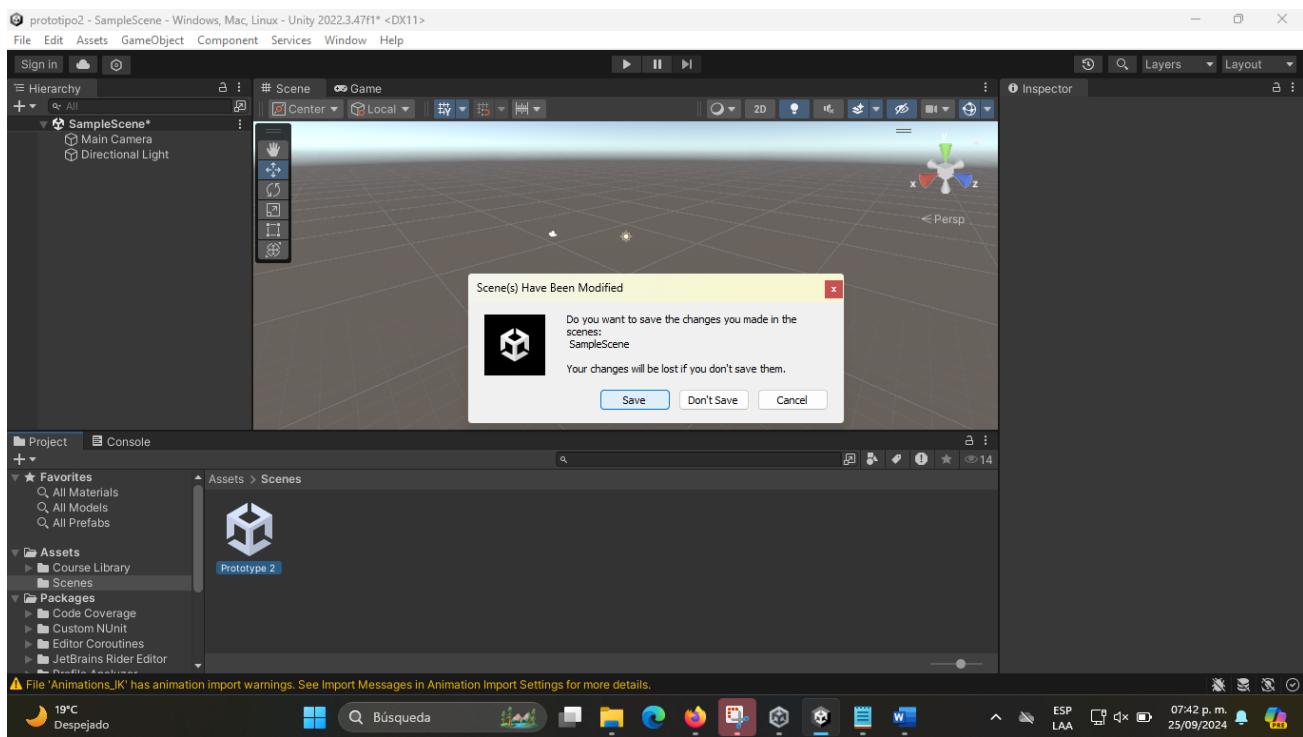
Se muestra esta contenido y posterior se eliminara el archivo **SampleScreen**



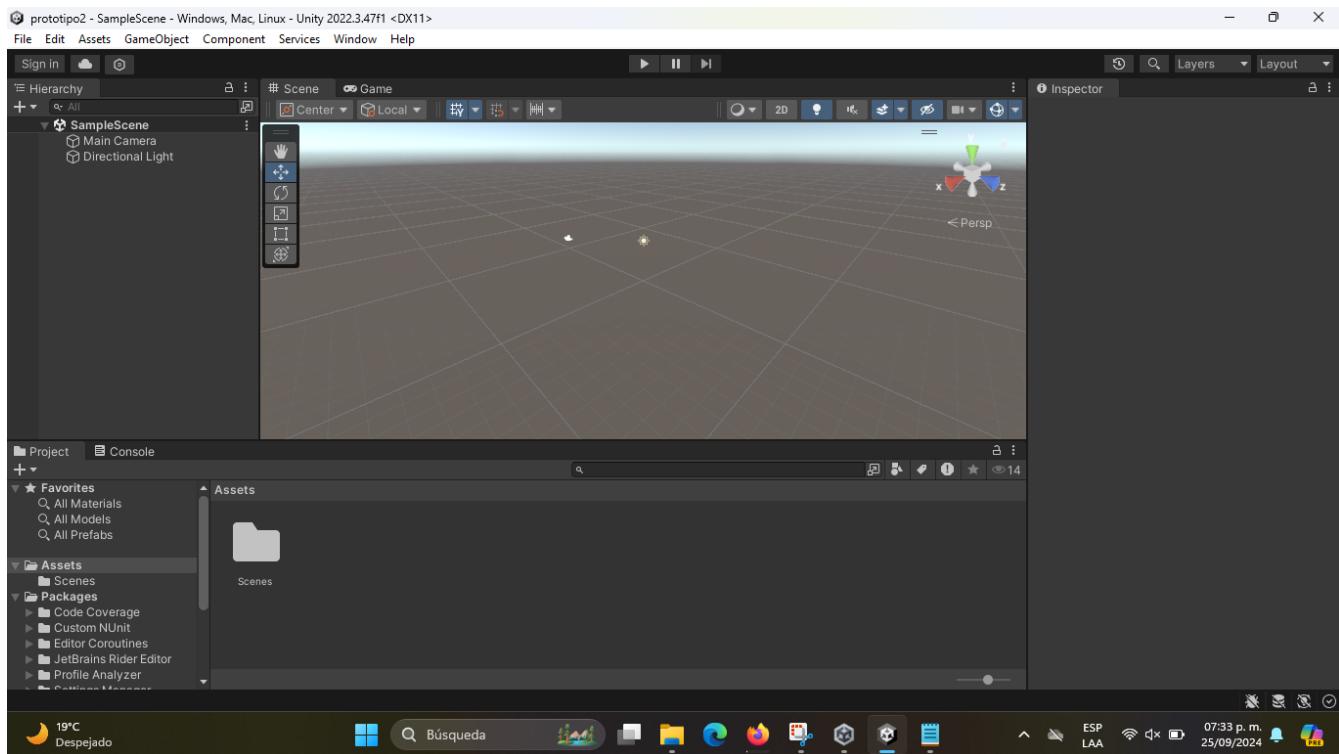
Eliminar **SampleScreen**, clic derecho sobre el paquete y dar **Delete**



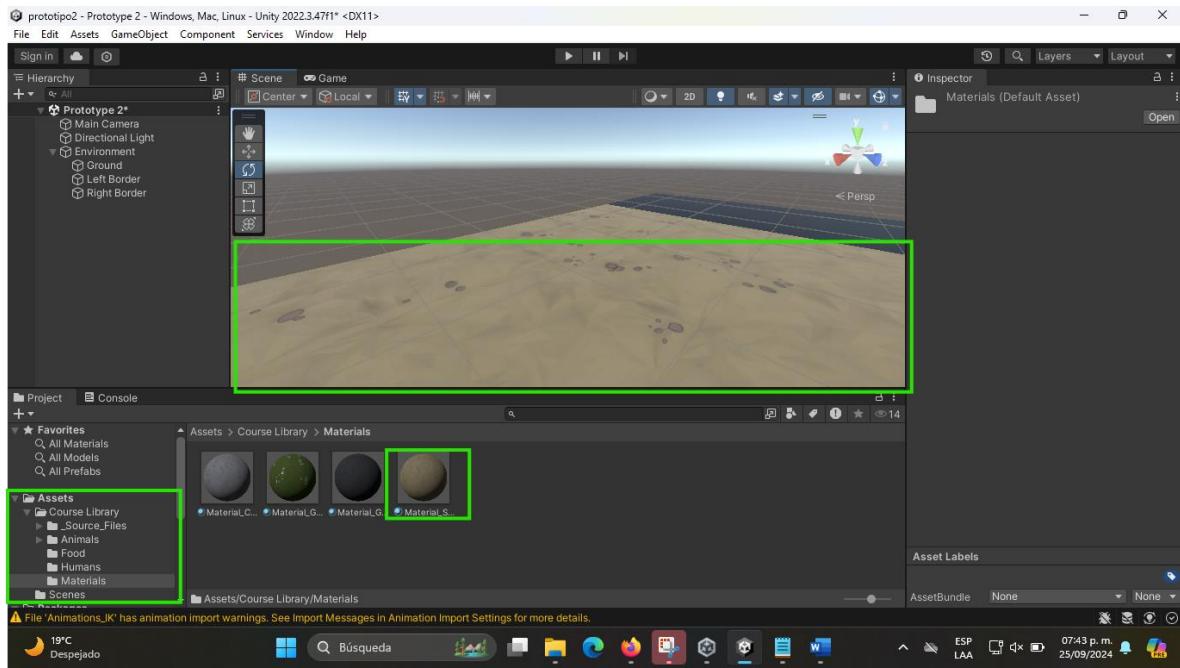
Dar clic en save para guardar los cambios



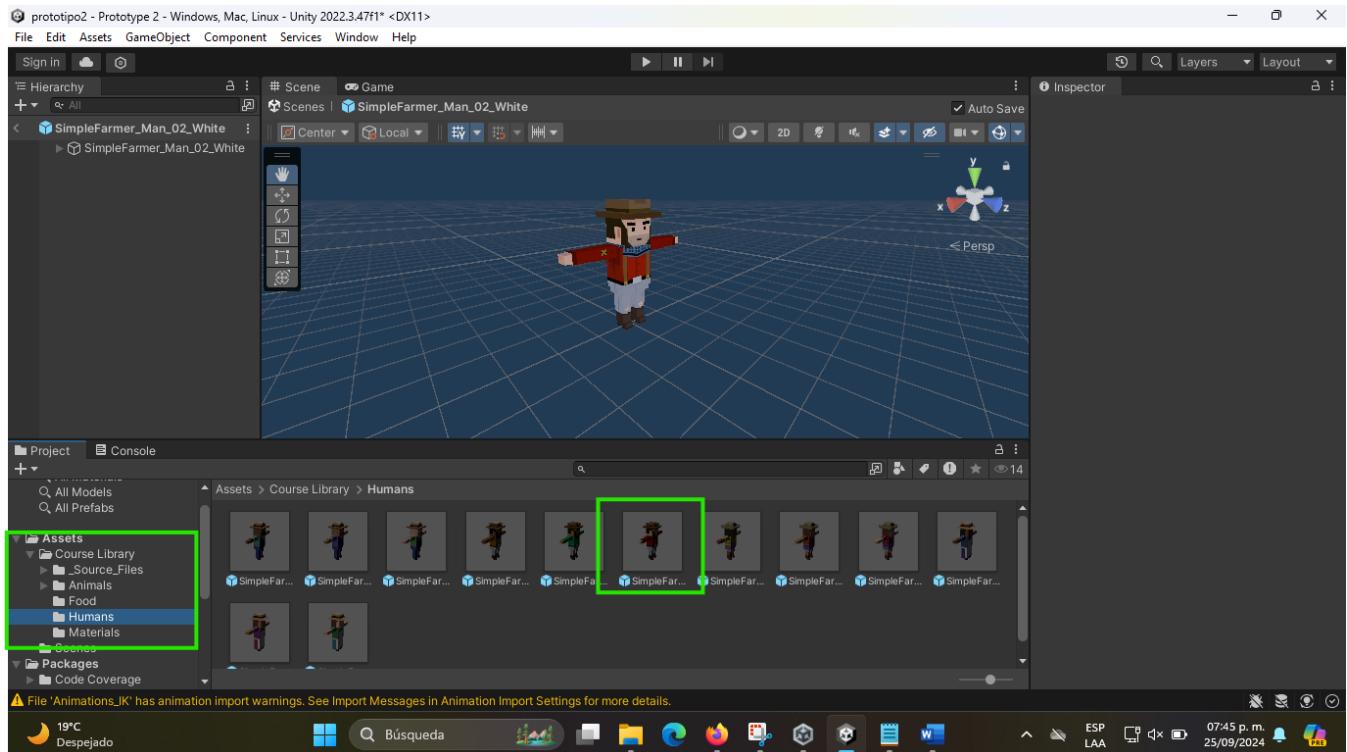
Elegir escenario desde la capeta **Assets>Scenes**



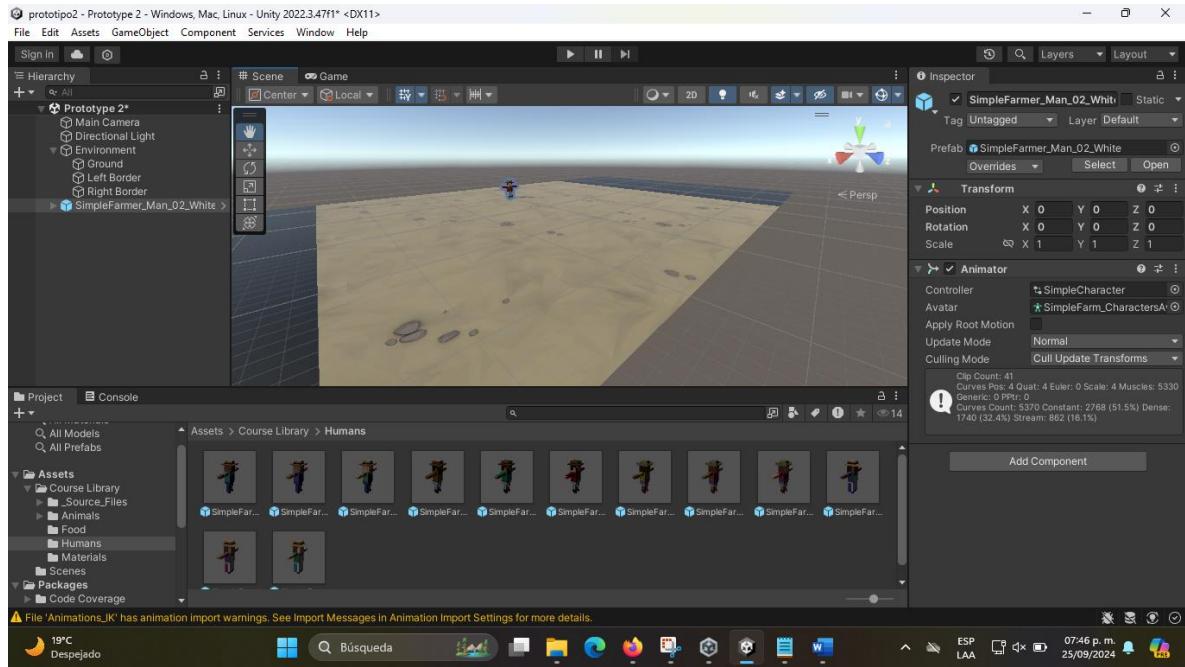
Arrastrar elemento al área izquierda de Prototype 2



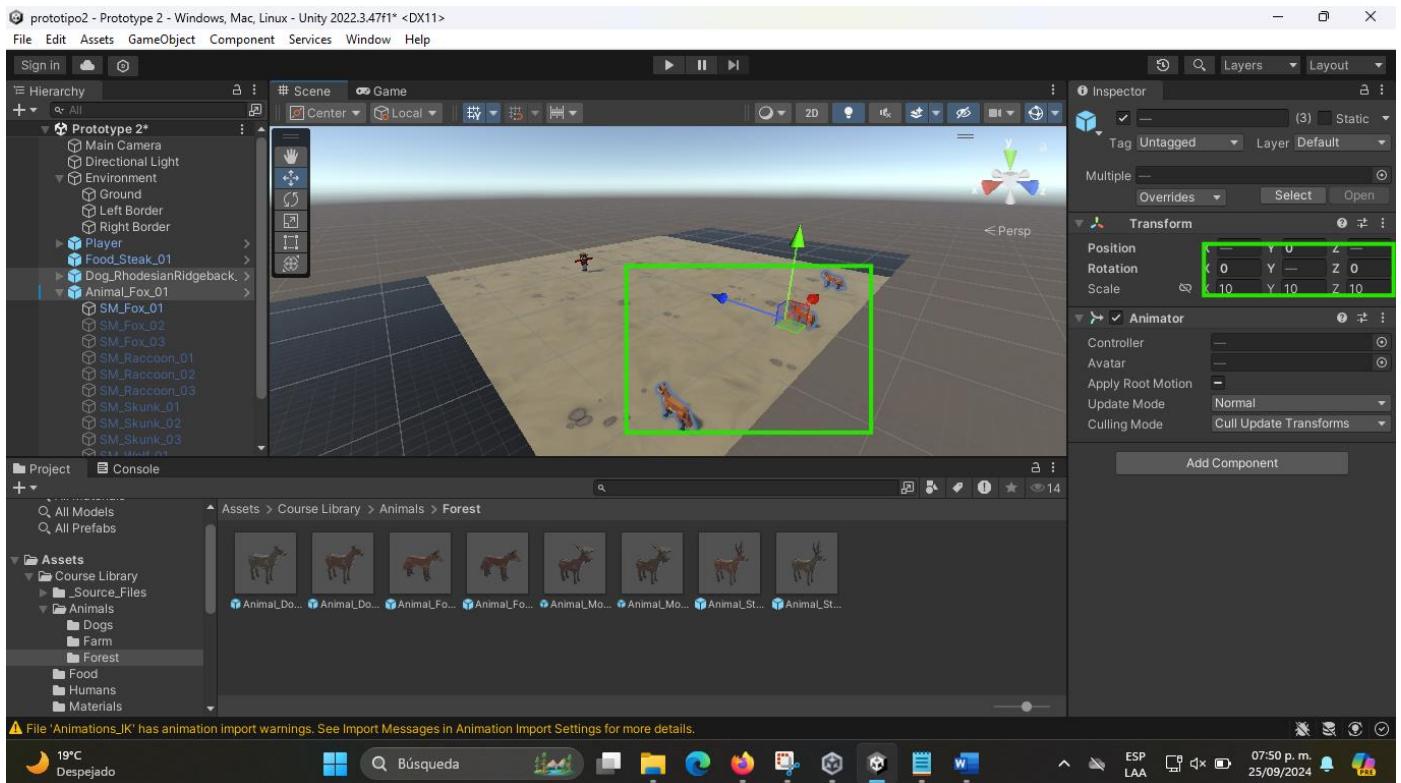
Elegir personaje, igual arrastrar elemento al área de Prototype2



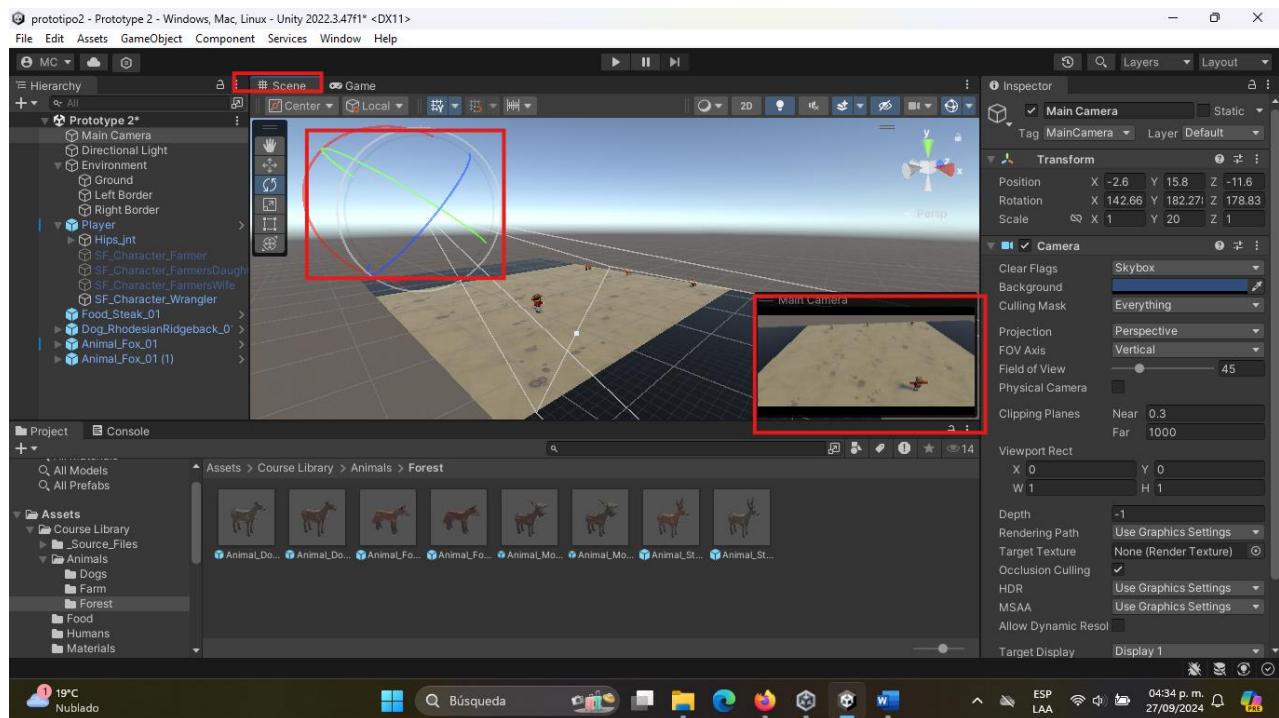
Resultado esperado



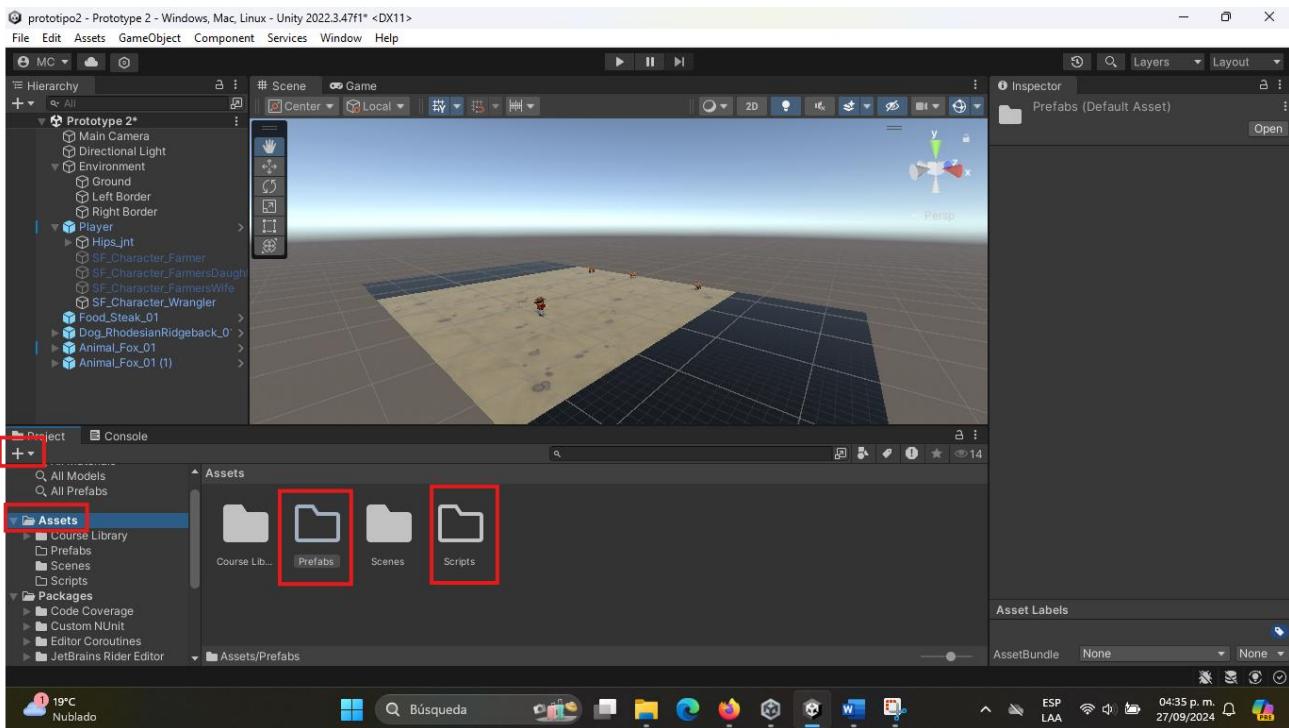
Agregar animales de la carpeta **>Assets>Animals**, arrastrar los 3 animales preferidos al área de **Prototype2**



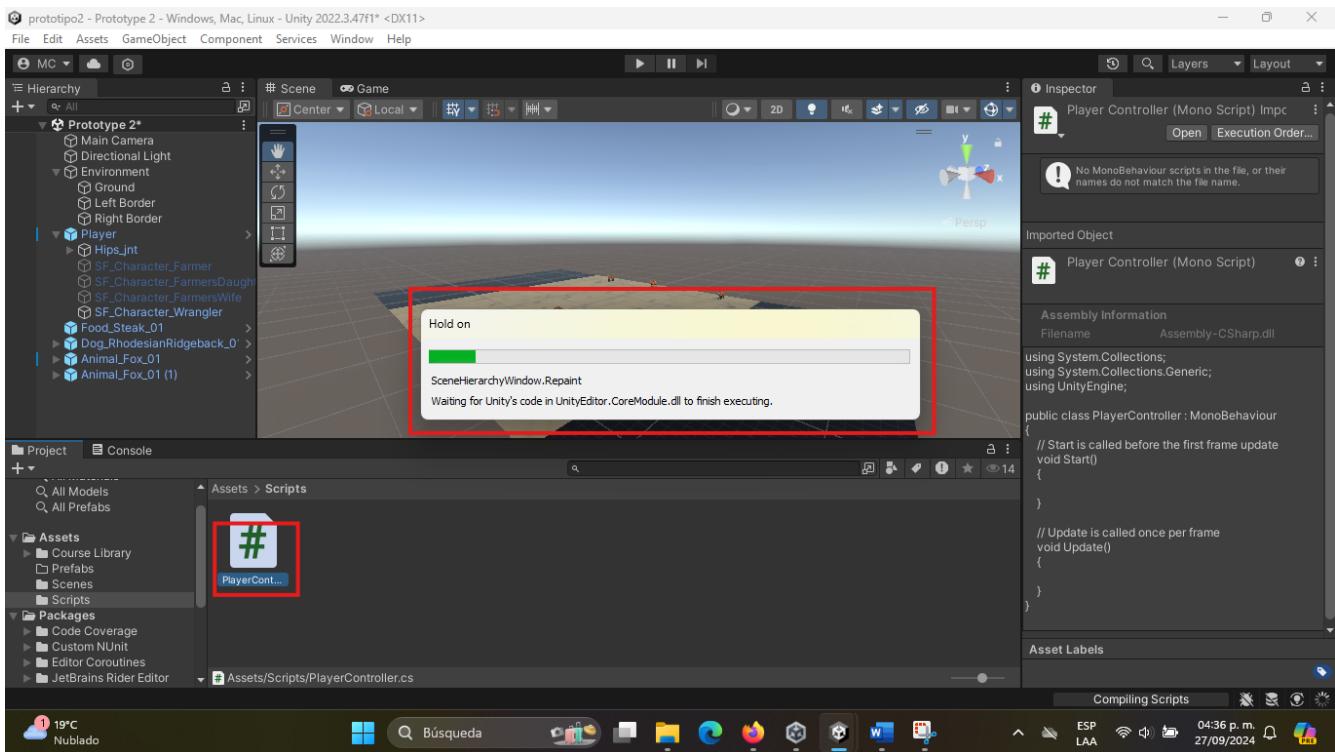
Modificar la cámara en un ángulo para visualizar todo



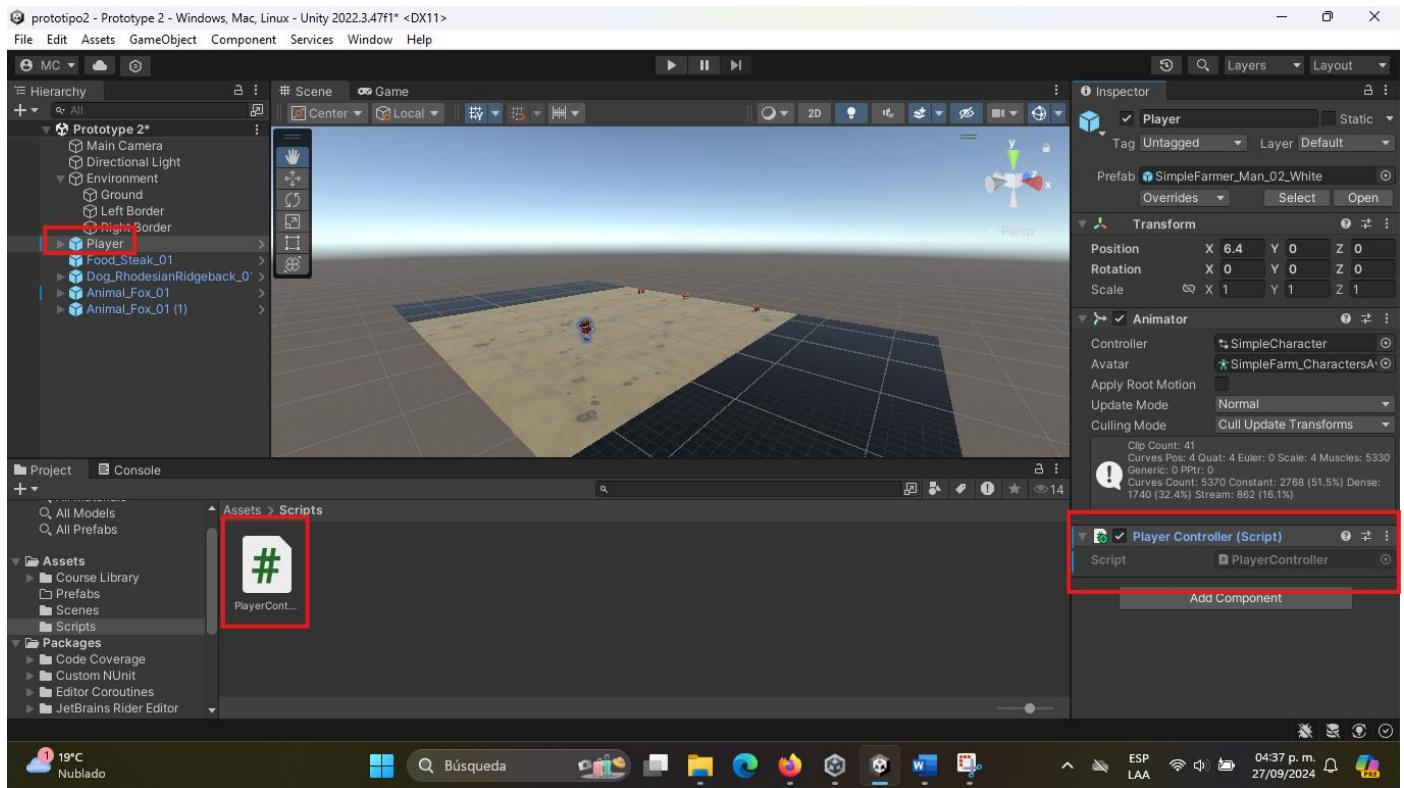
Agregar las carpetas **Prefabs** y **Scripts** en Assets



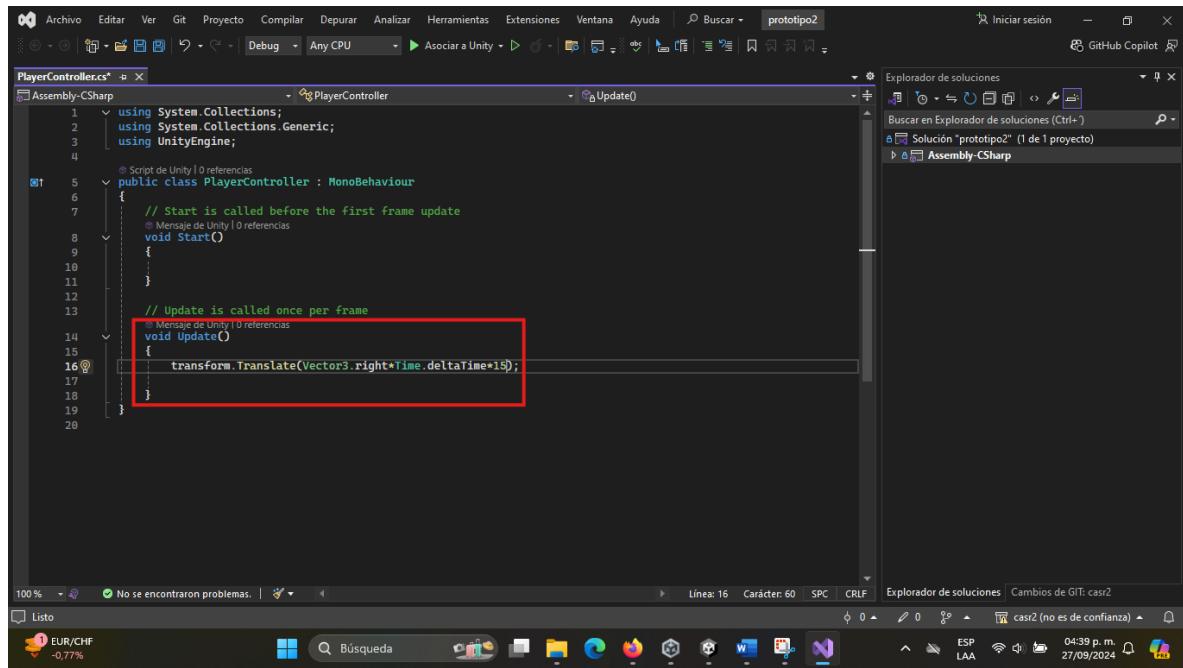
En la carpeta **Scripts**, seleccionar y dar clic en el signo + [agregar] y seleccionar **Script>C#**



Agregar script al personaje arrastrando al nombre de Player en el área de prototype2



Agregar el contenido al script [transform.Translate(Vector3.right*Time.deltaTime*15)]

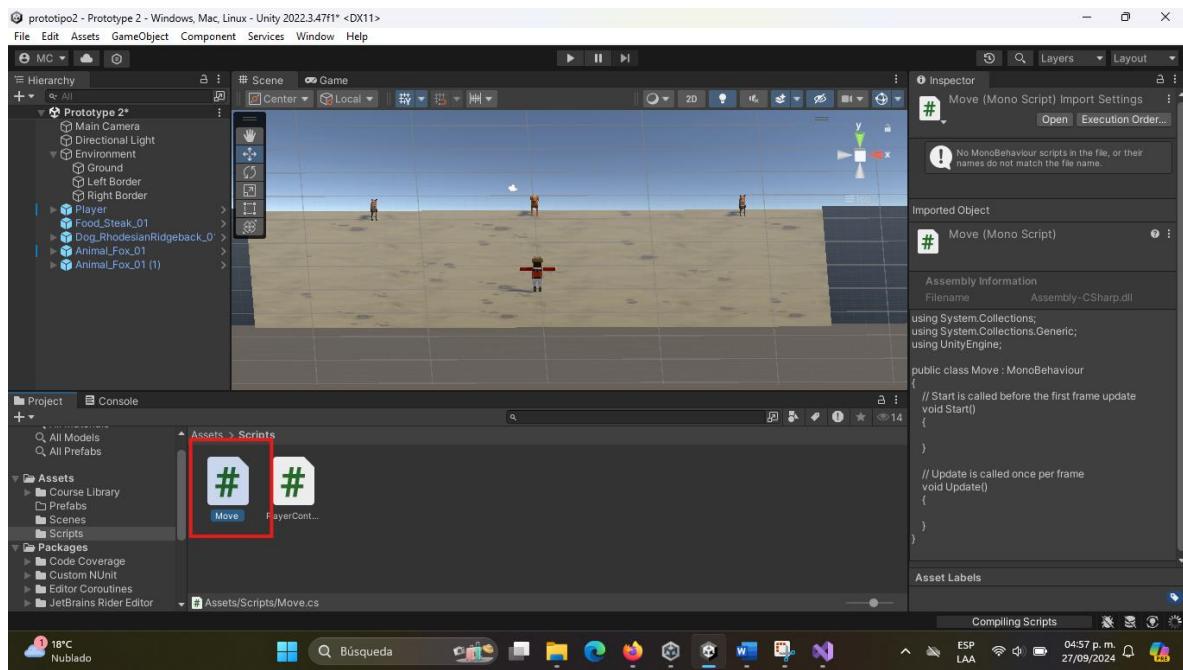


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerController : MonoBehaviour
{
    void Start()
    {
    }

    void Update()
    {
        transform.Translate(Vector3.right*Time.deltaTime*15);
    }
}
```

Crear el script Move de la misma forma y añadirlo a los animales y el personaje.



Agregar al script **Move** el siguiente código [transform.Translate(Vector3.forward*Time.deltaTime*10);]

The screenshot shows the Unity Editor's code editor with the file 'Move.cs' open. The code defines a MonoBehavior class named 'Move'. It includes a 'Start()' method and an 'Update()' method. The 'Update()' method contains the line 'transform.Translate(Vector3.forward*Time.deltaTime*10);'. This line is highlighted with a red rectangular box. The rest of the code is standard Unity movement logic.

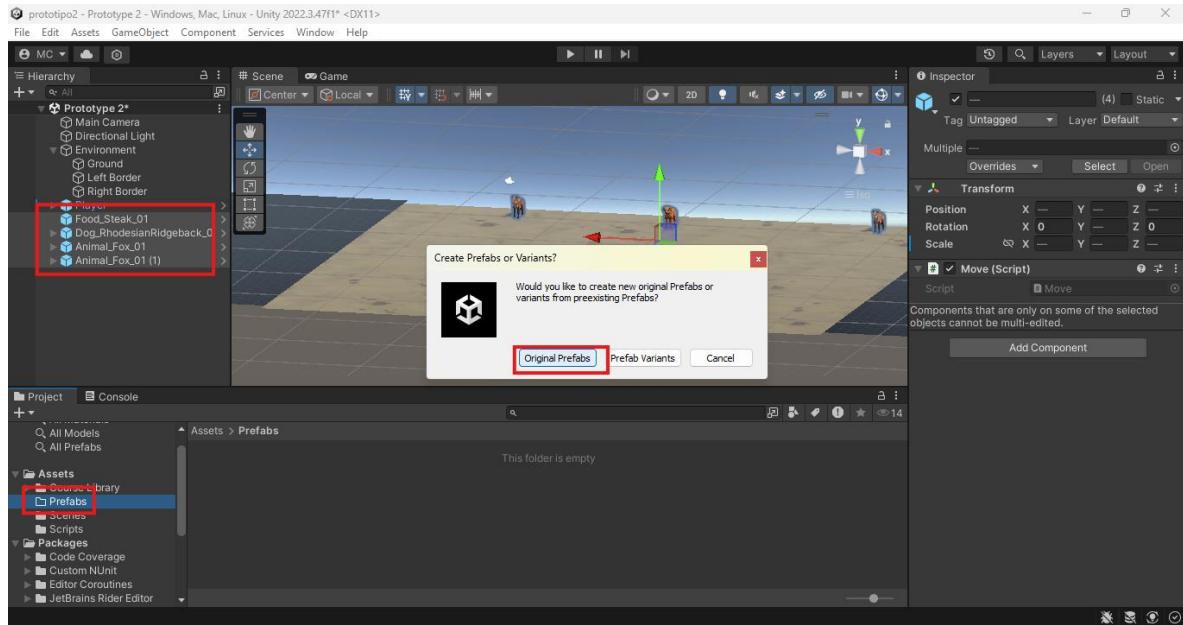
```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Move : MonoBehaviour
6 {
7     // Start is called before the first frame update
8     void Start()
9     {
10
11     }
12
13     // Update is called once per frame
14     void Update()
15     {
16         transform.Translate(Vector3.forward*Time.deltaTime*10);
17     }
18 }
19
20 }
```

Completar con el siguiente código

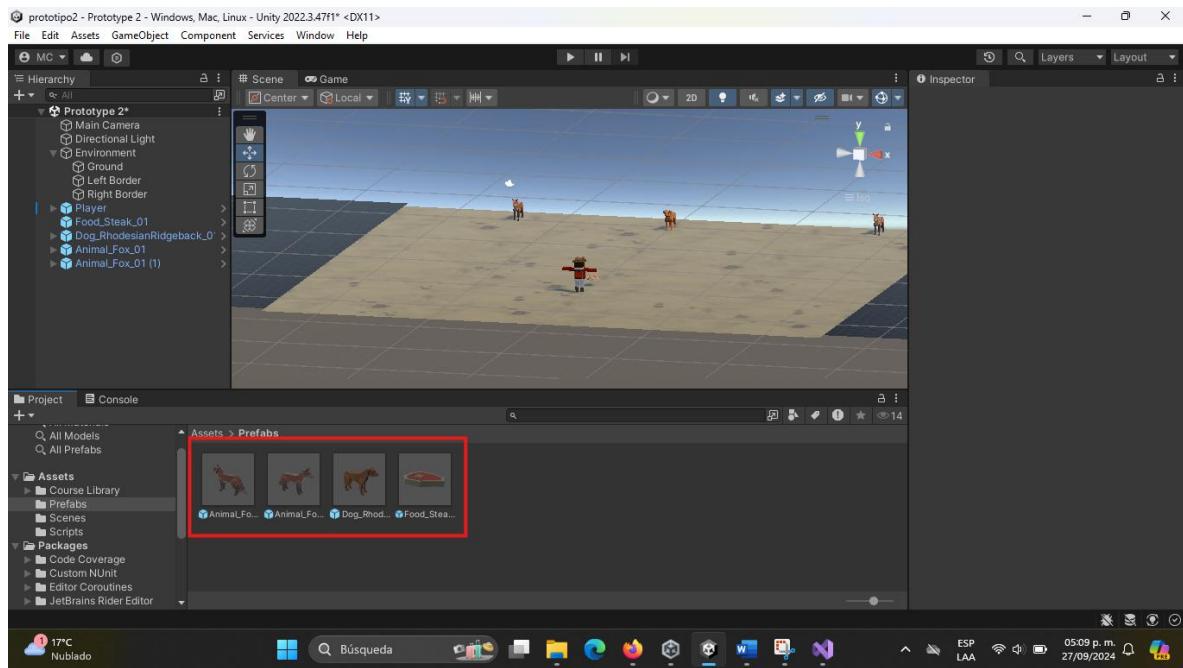
The screenshot shows the Unity Editor's code editor with the file 'Move.cs' open. The code defines a MonoBehavior class named 'Move'. It includes a 'Start()' method and an 'Update()' method. The 'Update()' method now contains additional logic: it checks if the object's position is greater than -17. If so, it destroys the game object. Otherwise, if the position is less than -17, it also destroys the game object. Finally, it moves the object forward. The code additions are highlighted with a red rectangular box.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Move : MonoBehaviour
6 {
7     // Start is called before the first frame update
8     void Start()
9     {
10
11     }
12
13     // Update is called once per frame
14     void Update()
15     {
16         if(transform.position.z > -17)
17         {
18             Destroy(gameObject);
19         }
20         else if(transform.position.z < -17)
21         {
22             Destroy(gameObject);
23         }
24         transform.Translate(Vector3.forward*Time.deltaTime*10);
25     }
26 }
27
28 }
```

Agregar cada uno de los elementos (animales, comida y personaje) a la carpeta de **Prefabs**.



Verificar que se encuentren dentro.

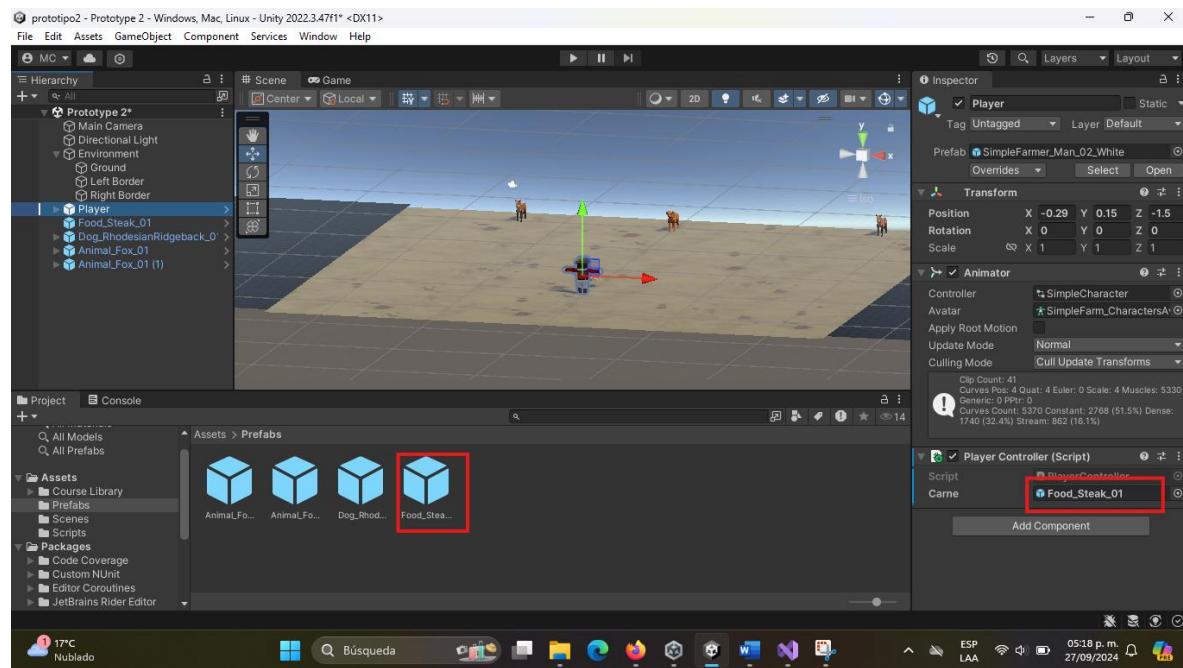


En el script **PlayerController** se agrega una variable pública y una funcionalidad para lanzar la comida.

The screenshot shows the Unity Editor's code editor with the file `PlayerController.cs` open. The code defines a class `PlayerController` with two methods: `Start()` and `Update()`. In the `Update()` method, line 26 contains the code `Instantiate(carne, transform.position, carne.transform.rotation);`, which is highlighted with a red box. The code is part of a larger script that handles player movement and food instantiation.

```
public GameObject carne;
void Start()
{
}
void Update()
{
    float hor = Input.GetAxis("Horizontal");
    if (transform.position.x > 24)
    {
        transform.position = new Vector3(24, //x
                                         transform.position.y, //y
                                         transform.position.z); //z
    }
    Instantiate(carne, transform.position,
               carne.transform.rotation);
    transform.Translate(Vector3.right*Time.deltaTime*8*hor);
}
```

Agregamos el objeto a la carne o comida



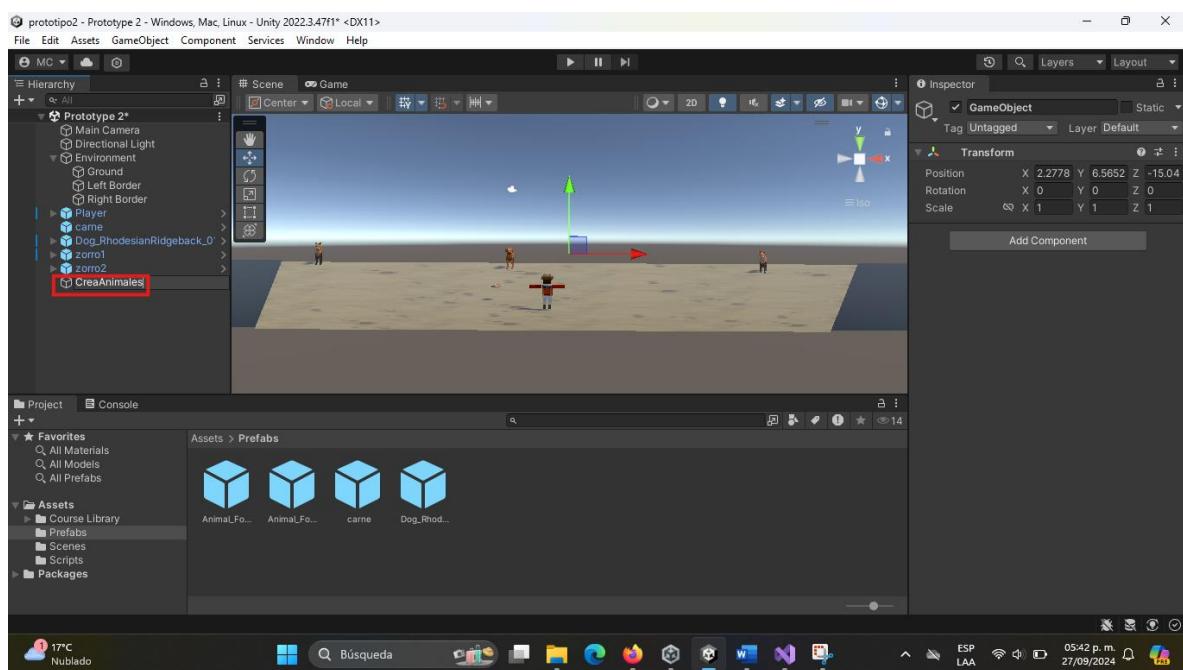
Realiza la siguiente modificación a la condición para el movimiento de la comida.

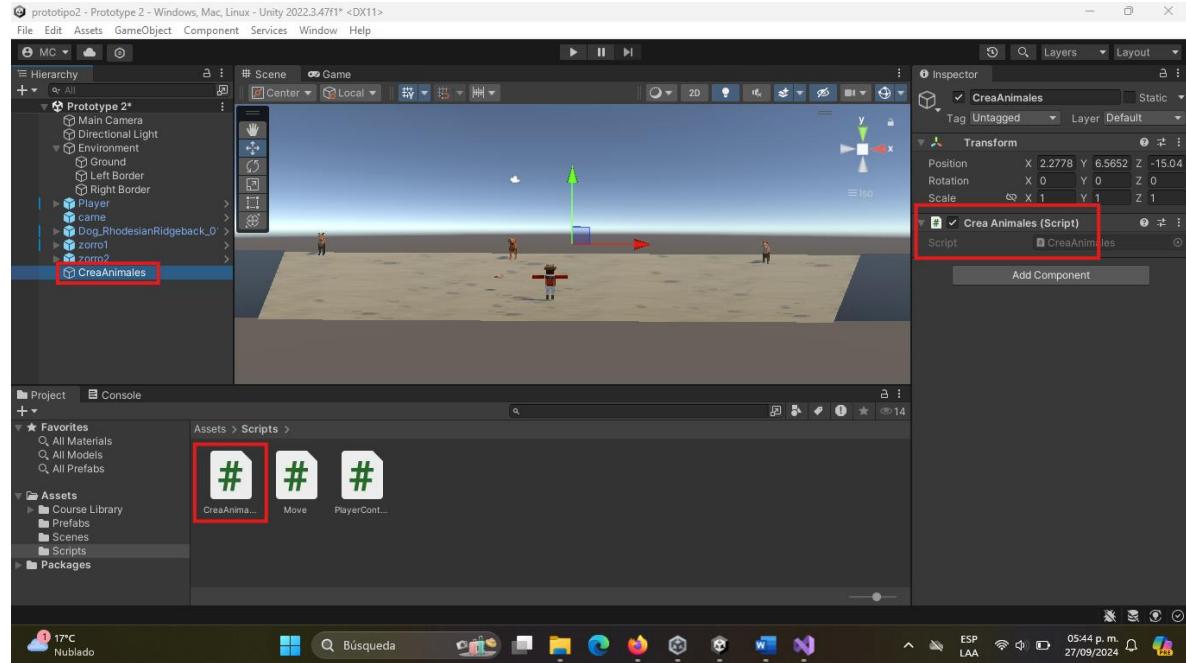
The screenshot shows the Visual Studio IDE interface with the following details:

- Top Bar:** Archivo, Editar, Ver, Git, Proyecto, Compilar, Depurar, Analizar, Herramientas, Extensiones, Ventana, Ayuda, Buscar, Iniciar sesión, GitHub Copilot.
- Solution Explorer:** Explorador de soluciones, Buscar en Explorador de soluciones (Ctrl+), Solución "prototipo2" (1 de 1 proyecto), Assembly-CSharp.
- Code Editor:** PlayerController.cs (Assembly-CSharp) file open. The code defines a Start() method and an Update() method. The Update() method contains logic for horizontal movement and space key instantiation. A red box highlights the instantiation code:

```
if (Input.GetKeyDown(KeyCode.Space))
{
    Instantiate(carne, transform.position,
    carne.transform.rotation);
}
```
- Status Bar:** Linea: 30, Carácter: 1, SPC, CRLF.
- Bottom Bar:** Elementos guardados, Búsqueda, Navegación, Cambios de GIT: cast2, 17°C Nublado.

Crear objeto vacío (**CreaAnimales**) en el área de **Prototype2**, dar clic derecho y agregar un objeto vacío.





En el script **CreaAnimales** se agrega la funcionalidad para que los animales se muevan

```

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

public class CreaAnimales : MonoBehaviour
{
    public GameObject[] animales;
    void Start()
    {
        InvokeRepeating("CreateAnimal", 1, 1);
    }

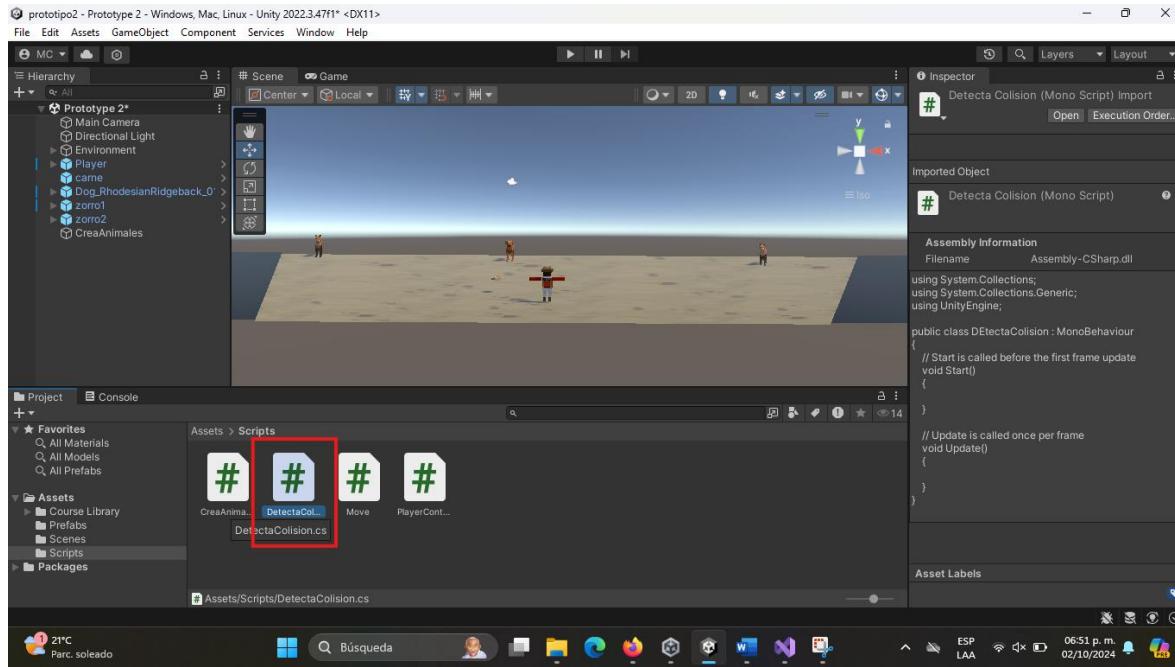
    void CreateAnimal()
    {
        int index = Random.Range(0, 3);
        int posX = Random.Range(-20, 20);

        animales[index].transform.position = new Vector3(posX,
            animales[index].transform.position.y,
            animales[index].transform.position.z);

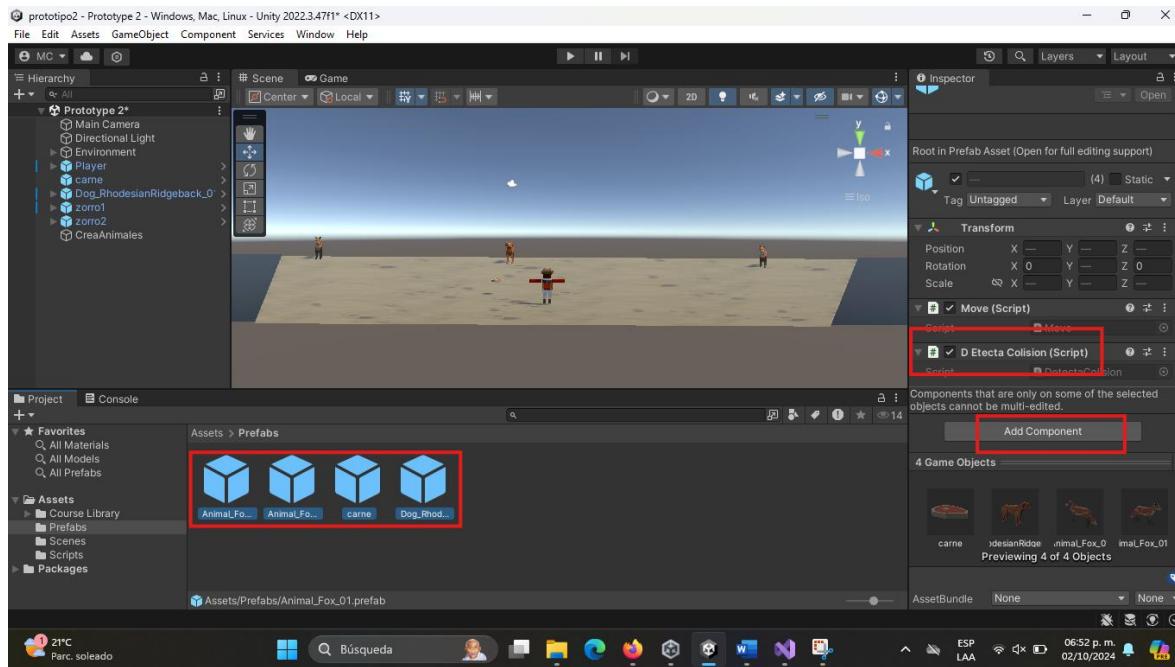
        Instantiate(animales[index], animales[index].transform.position,
            animales[index].transform.rotation);
    }
}

```

Se crea un nuevo script para destruir la comida al tocar los animales, llamado **DetectarCollision**.



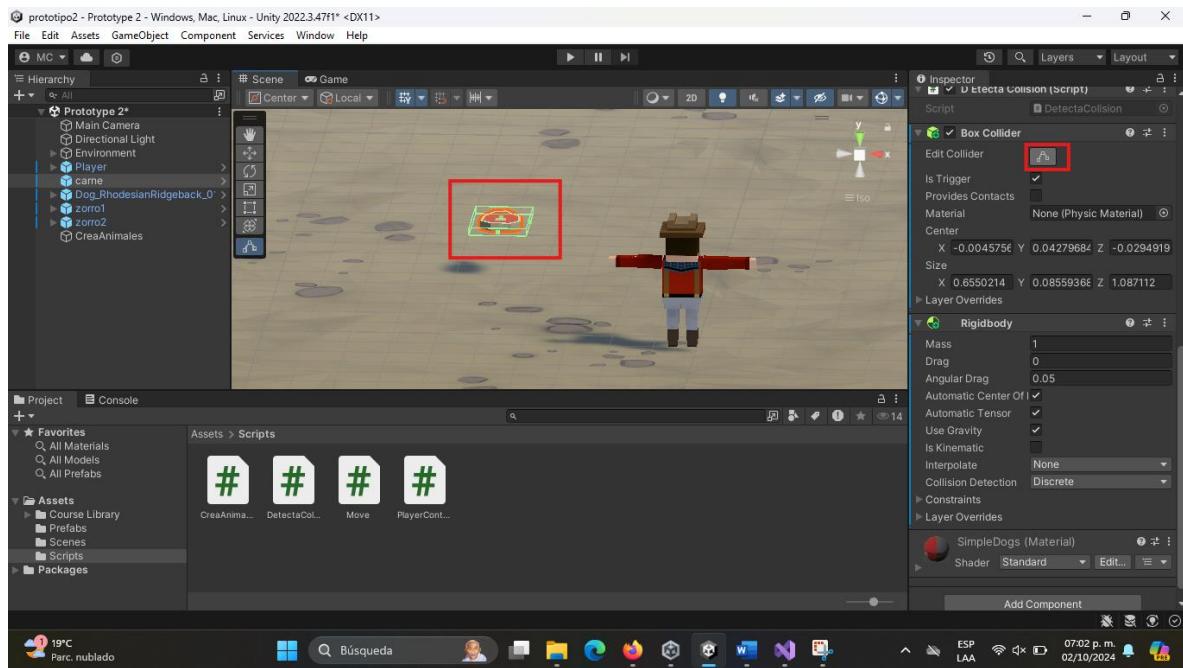
Se agrega a cada uno de los elementos del juego y el lado izquierdo se agrega un componente nuevo, clic en **AddComponent**.



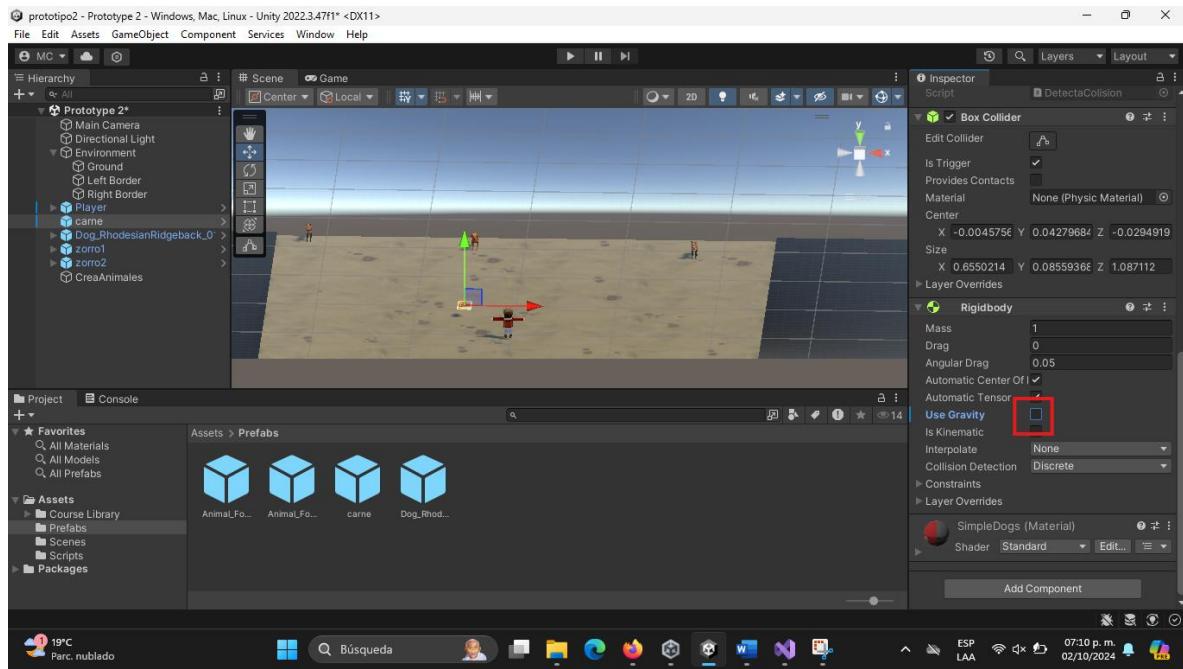
Seleccionar **Box Collider**, recuerda verificar que se agregue a los animales y la comida. (activar el **IsTrigger**)



Se selecciona el icono en la imagen y se cambia las dimensiones del Box Collider para estar del tamaño del elemento (Repetir con los animales)



Seleccionar la comida para activar en el lado izquierdo el elemento **Use Gravity**.



A la carne y los animales se les agrega el **Overrides>Apply All**.

