



HLC

2º DAW

The background of the slide features a dark blue gradient. On the left, a blue and green geometric shape is partially visible. Two futuristic cars are shown in motion: a blue car on the left leaving a thick white smoke trail, and a red car on the right leaving a large, billowing orange and yellow smoke trail. The text is centered and white, providing information about the Unity Asset Store.

Asset Store

La Asset Store es una tienda de recursos de todo tipo:

Código
Texturas
Materiales
Modelos
Animaciones
Utilidades...

De pago y gratuitos.

Unity ofrece muchos packs de ejemplo como parte de sus tutoriales y documentación:

Standard Assets
2D Game Kit
3D Game Kit
Particles Pack
UI sample

Si hemos entrado con nuestra cuenta de Unity podremos acceder a:

Favoritos
Mi Perfil

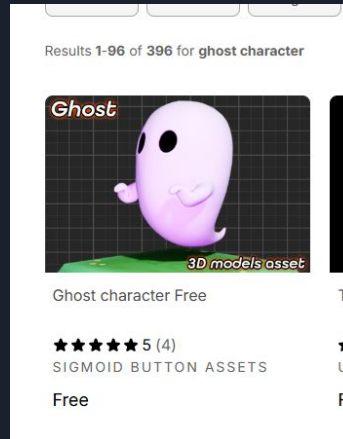
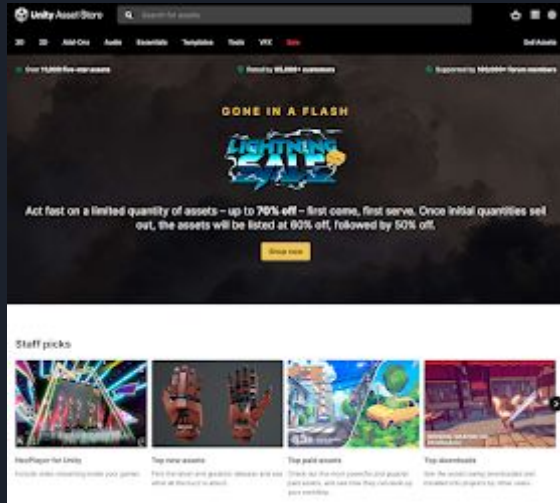
Assets ya descargados o comprados

Todas estas opciones están disponibles tanto en la web como en el propio Unity, a través de la ventana
Asset Store (Window/Asset Store)

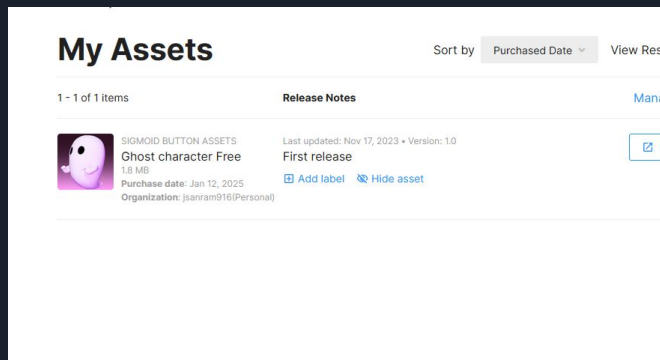
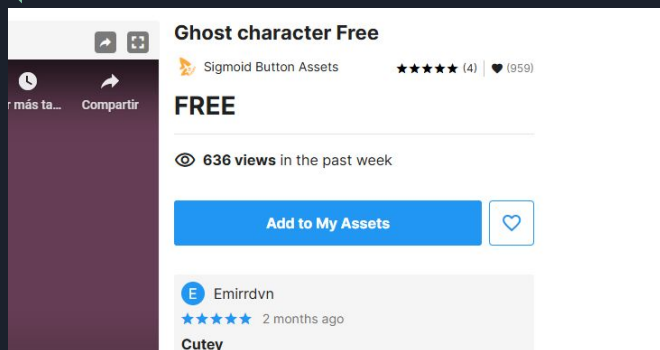
Asset Store

<https://assetstore.unity.com/>

Iremos a la web de Asset Store y buscaremos un elemento, por ejemplo: *Ghost character*



Asset Store

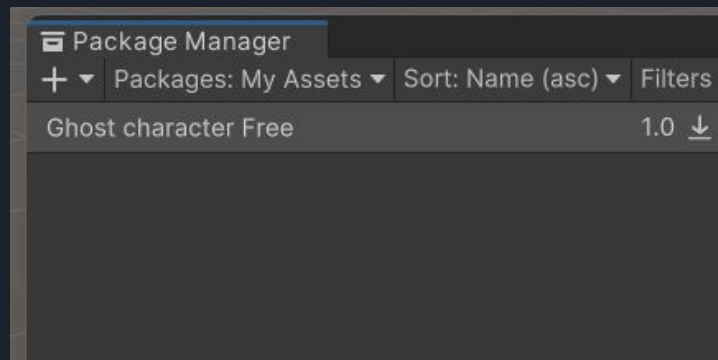


✓ Added to My Assets

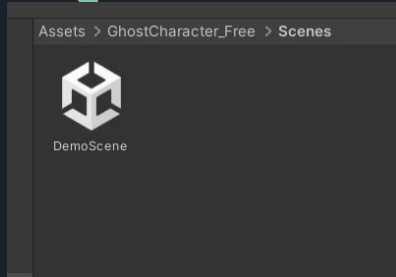


SIGMOID BUTTON ASSETS
Ghost character Free

Desde Unity, Window > Package Manager > My Assets. E importamos.



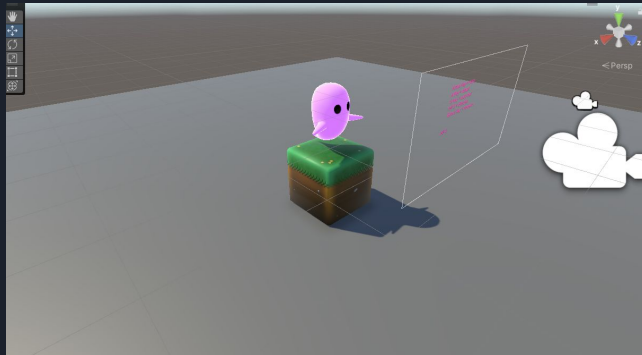
Asset Store



Abrimos la **escena** de demostración que nos trae el fantasma.

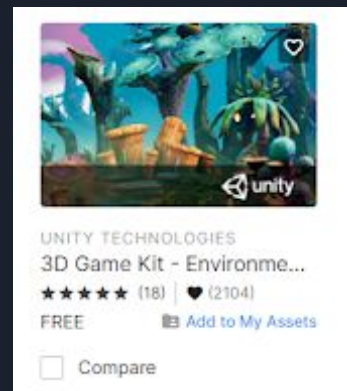
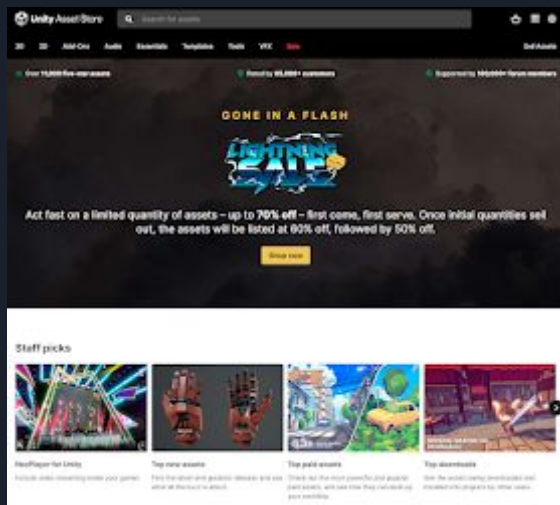
En Unity, una scene (escena) es un contenedor que organiza todos los elementos y configuraciones necesarios para una parte específica de nuestro juego o aplicación. Podemos pensar en una escena como un nivel, una pantalla o un estado del juego.

Por ejemplo, podríamos tener una escena para el menú principal, otra para el primer nivel del juego, otra para el nivel final, y así sucesivamente.



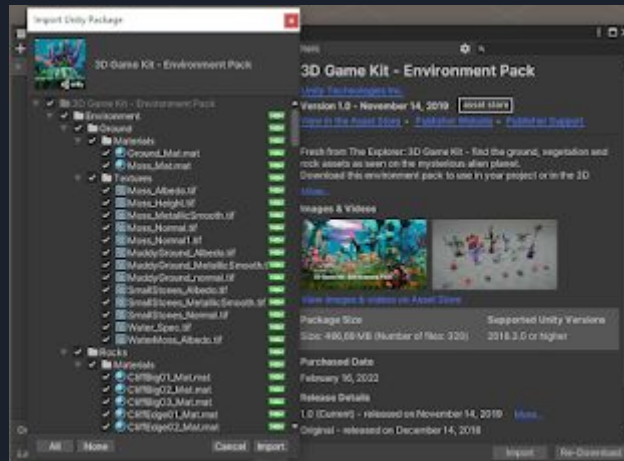
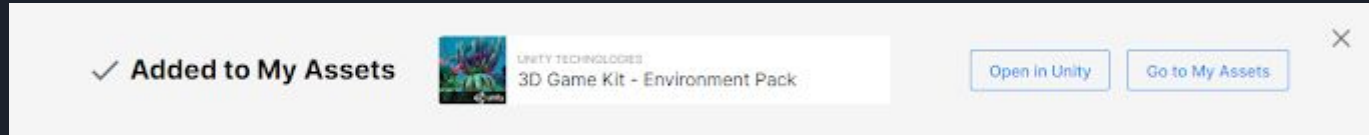
Asset Store

De nuevo, buscaremos un elemento: 3D Game Kit Environment y acotamos los filtros de búsqueda para que sean gratuitos. Elegiremos, por ejemplo este:



Asset Store

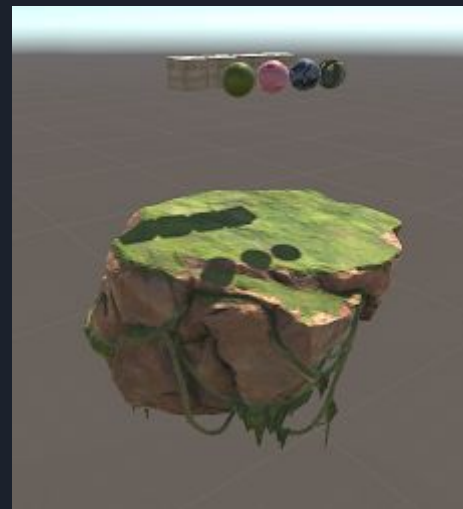
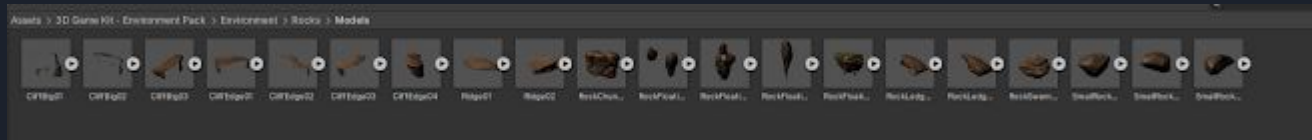
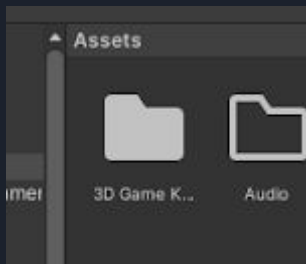
Haremos click en Add to My Assets y luego en Open in Unity. Tras esto, podremos darle a Download en la nueva ventana que nos aparece en Unity.



Podemos importar únicamente contenido seleccionado del paquete descargado.

Prefabs

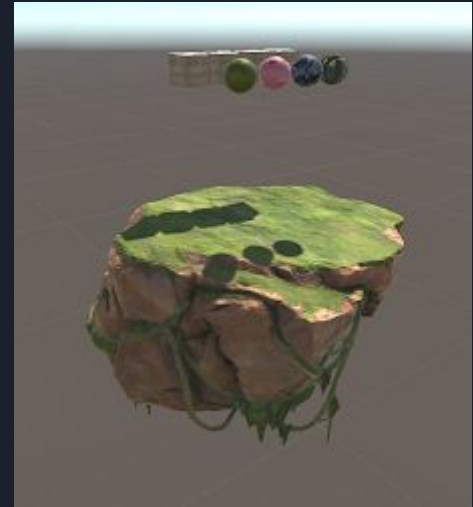
Una vez descargado e importado lo tendremos disponible en Assets





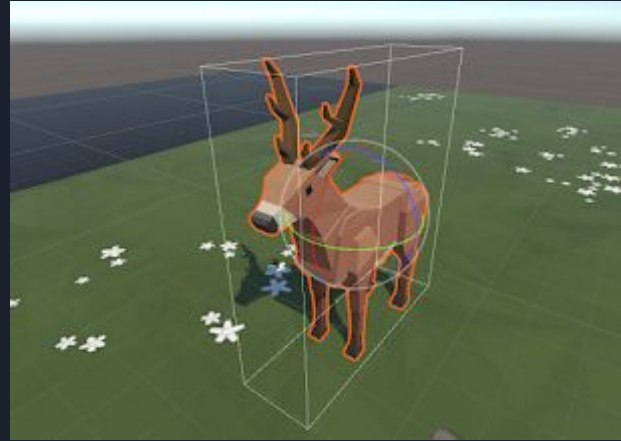
Prefabs

Haremos un juego en el que cuando una esfera roce el terreno, aparezca como amigo y si es cuadrado, como enemigo.



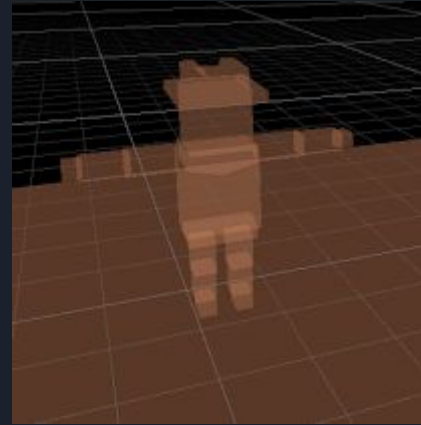
Proyecto. Creación de un juego real

Con los conocimientos básicos que tenemos en Unity y C#, vamos a crear un juego completo, con movimiento de jugadores, disparos y mapa.



Proyecto. Creación de un juego real

En primer lugar, vamos a crear un nuevo proyecto, en el que vamos a importar un conjunto de recursos que encontraremos en Moodle. **Prototype-2-Starter-File**
En este juego, un granjero disparará comida a los animales para eliminarlos.

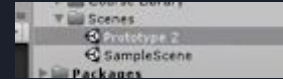


Proyecto. Creación de un juego real

Importamos el paquete descargado
Añadimos la escena Assets>Scenes>Prototype 2

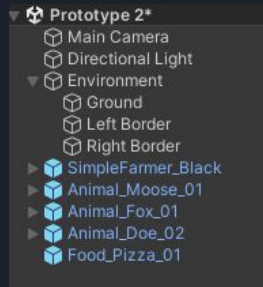


Podemos eliminar la escena de muestra Sample Scene



Si nos aparece algún elemento en rosa: Edit>Rendering>Materials>Convert

Desde Assets>Course Library > Humans, podemos añadir un granjero a nuestro plano.
Desde Assets>Course Library > Animals > Forest, por ejemplo, añadimos 3 animales.
Y también desde Assets>Course Library > Food añadiremos un trozo de pizza.





Proyecto. Creación de un juego real

Vamos a añadir movimiento lateral al personaje del granjero. Al cual vamos a renombrar desde la vista de jerarquía a **Jugador**.

Para poder usar este movimiento lateral crearemos un script en una carpeta que vamos a crear:

Assets>CourseLibrary>Scripts>ControlJugador.cs. Este script se lo asignaremos (arrastrando por ejemplo) a nuestro Jugador.

Con esto, ya podemos mover nuestro granjero, pero se sale por los lados del mapa.

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

public class ControlJugador : MonoBehaviour
{
    public float desplazamientoHorizontal;
    public float velocidad = 10.0f;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

        desplazamientoHorizontal = Input.GetAxis("Horizontal");
        //Mover al jugador horizontalmente
        transform.Translate(Vector3.right *
desplazamientoHorizontal * Time.deltaTime * velocidad);

    }
}
```



Proyecto. Creación de un juego real

Para solucionar esto, le pondremos un rango de límites laterales a nuestro código.

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

public class ControlJugador : MonoBehaviour
{

    public float desplazamientoHorizontal;
    public float velocidad = 10.0f;
    public float rangoX = 20.0f;

    // Start is called before the first frame update
    void Start()
    {

    }

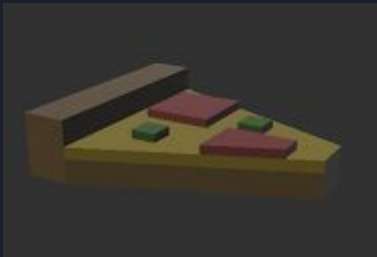
    // Update is called once per frame
    void Update()
    {

        if(transform.position.x < -rangoX){
            transform.position = new Vector3 (-rangoX, transform.position.y,
            transform.position.z);
        }
        if(transform.position.x > rangoX){
            transform.position = new Vector3 (rangoX, transform.position.y,
            transform.position.z);
        }
        desplazamientoHorizontal = Input.GetAxis("Horizontal");
        //Mover al jugador horizontalmente
        transform.Translate(Vector3.right * desplazamientoHorizontal *
        Time.deltaTime * velocidad);

    }
}
```

Proyecto. Creación de un juego real

Trabajaremos ahora en que el trozo de pizza se mueva (dispare) hacia adelante (eje Z). Para ello crearemos un script llamado AvancePizza y se lo añadiremos a nuestro objeto en el juego. Podemos probarlo y ver cómo avanza sola, pero se pierde en el infinito eje Z.



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

public class AvancePizza : MonoBehaviour
{
    public float velocidad = 40.0f;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        transform.Translate(Vector3.forward *
            Time.deltaTime * velocidad);
    }
}
```



Proyecto. Creación de un juego real

Ahora vamos a crear Prefabs. Los prefabs son instanciables, con lo que, si queremos que cada vez que le demos a la barra espaciadora se lance un trozo de pizza, se creará un objeto (instancia) de ella en cada ocasión.

Para ello modificaremos sus copias y las guardaremos como prefabs originales.

Por ejemplo, lo que tenemos hasta ahora de pizza, es el prefab original, pero nosotros lo hemos modificado añadiéndole un script. Queremos que al darle a la barra espaciadora aparezca este mismo modificado, no uno nuevo sin script.

💡 ¿Qué es un prefab?

<https://docs.unity3d.com/es/530/Manual/Prefabs.html#:~:text=El%20prefab%20act%C3%BAa%20como%20una,ajustes%20para%20cada%20instancia%20individualmente.>

Proyecto. Creación de un juego real

Para crear un prefab, en este caso de pizza, vamos a crear una carpeta nueva Assets>Prefabs y vamos a arrastrar el objeto Food_Pizza_01 de la jerarquía al esta carpeta.



Y lo sustituiremos por el original. “Original Prefab”. Ya tenemos este objeto en nuestra carpeta con el script incorporado.

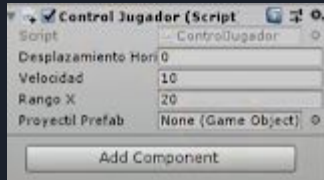
Vamos ahora a incorporar este prefab a nuestro Jugador, para que pueda lanzarlo. Nos vamos al script ControlJugador.cs y creamos el atributo **proyectorPrefab**:

```
public class ControlJugador : MonoBehaviour
{
    public float desplazamientoHorizontal;
    public float velocidad = 10.0f;
    public float rangoX = 20.0f;
    public GameObject proyectilPrefab;

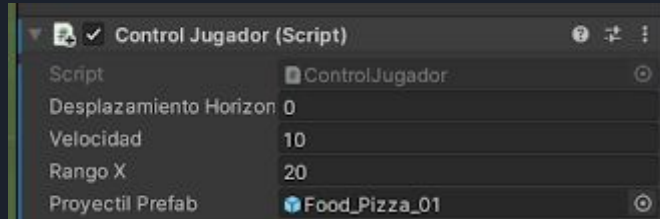
    // Start is called before the first frame update
    void Start()
    ...
}
```


Proyecto. Creación de un juego real

Nos fijamos en que ahora nuestro Jugador tiene una variable nueva en el panel lateral:



Arrastraremos ahí (donde aparece None) el prefab de pizza recién creado.





Proyecto. Creación de un juego real

Continuamos haciendo que al pulsar la barra espaciadora, el granjero lance trozos de pizza. Para ello modificamos nuestro código añadiendo un nuevo método. Aunque se siguen perdiendo en el infinito. Es decir, nunca desaparecen esas nuevas instancias.

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

public class ControlJugador : MonoBehaviour
{
    public float desplazamientoHorizontal;
    public float velocidad = 10.0f;
    public float rangoX = 20.0f;
    public GameObject proyectilPrefab;
    // Start is called before the first frame update
    void Start()
    {

    }

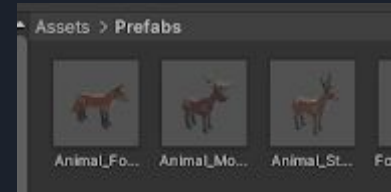
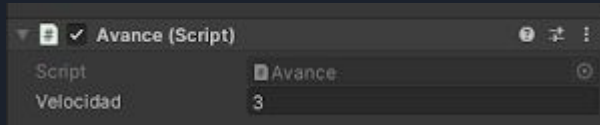
    // Update is called once per frame
    void Update()
    {
        if(Input.GetKeyDown(KeyCode.Space)){
            //lanzamiento de pizza
            Instantiate(proyectilPrefab, transform.position,
            proyectilPrefab.transform.rotation);
        }
        if(transform.position.x < -rangoX){
            transform.position = new Vector3(-rangoX, transform.position.y,
            transform.position.z);
        }
        if(transform.position.x > rangoX){
            transform.position = new Vector3(rangoX, transform.position.y,
            transform.position.z);
        }
        desplazamientoHorizontal = Input.GetAxis("Horizontal");
        //Mover al jugador horizontalmente
        transform.Translate(Vector3.right * desplazamientoHorizontal *
        Time.deltaTime * velocidad);
    }
}
```

Proyecto. Creación de un juego real


Vamos a hacer que nuestros animales avancen, al igual que las pizzas, pero en este caso hacia el granjero. Para ello vamos a reutilizar nuestro script **AvancePizza.cs**. Así que, teniendo buenas prácticas, vamos a renombrarlo a **Avance.cs**.

Para cambiar el nombre de un script no es suficiente con: botón secundario>rename, sino que también hay que abrir dicho archivo en VS Code y modificar el nombre de la clase.

Una vez esto, se lo añadimos como componente a los 3 animales. Nos aparecerá en Inspector. Pero le cambiamos la velocidad a 3 en lugar de 40.



Una vez hecho esto, vamos a crear un prefab por cada animal, de la misma manera que lo hicimos con el trozo de pizza.



Proyecto. Creación de un juego real

En este punto los animales vienen a por nosotros, y podemos lanzar pizzas para destruirlos. La idea es que aunque los destruyamos, continúen apareciendo animales aleatoriamente, por eso los hemos instanciado. Igualmente, tanto los trozos de pizza como los 3 animales, siguen saliéndose del plano. Para ello vamos a crear el script DestruirFueraEscena.cs

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

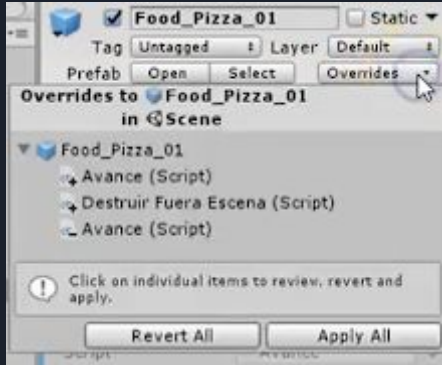
public class DestruirFueraEscena : MonoBehaviour
{
    float limiteSuperior = 30;
    float limiteInferior = -10;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if(transform.position.z > limiteSuperior ||
transform.position.z < limiteInferior){
            Destroy(gameObject);
        }
    }
}
```

Proyecto. Creación de un juego real

Una vez hecho este script, lo arrastramos a Food_Pizza_01 en el panel de jerarquía. Pero así sólo se lo hemos añadido al primer trozo de pizza, entonces debemos darle a



Apply All desde Overrides para que también se le aplique a todas sus nuevas instancias.

En el caso de los 3 animales es diferente. Se lo debemos añadir a los prefabs. Para ello simplemente los seleccionamos y le damos a Add Component y lo buscamos.



Proyecto. Creación de un juego real

Seguimos mejorando nuestro juego. Ahora queremos que al pulsar Intro también podamos lanzar galletas.

Añadimos a nuestra escena (panel de jerarquía) el objeto Food_Cookie_01.

Le añadimos a este objeto los scripts Avance.cs y DestruirFueraEscena.cs

Una vez hecho, lo creamos como prefab.

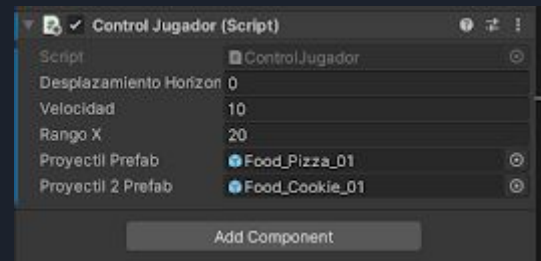
Tenemos, entonces, que modificar el script ControlJugador, añadiéndole el atributo:

```
public GameObject proyectil2Prefab;
```

y el método:

```
if(Input.GetKeyDown(KeyCode.Return)){  
    //lanzamiento de galleta  
    Instantiate(proyectil2Prefab, transform.position,  
    proyectil2Prefab.transform.rotation);  
}
```

Una vez modificado, arrastramos el prefab de la galleta al componente de Jugador (como hicimos anteriormente con la pizza).

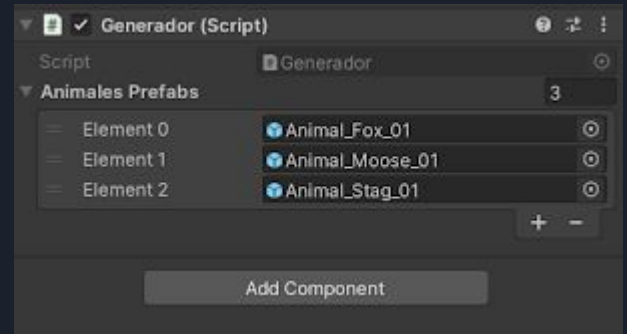
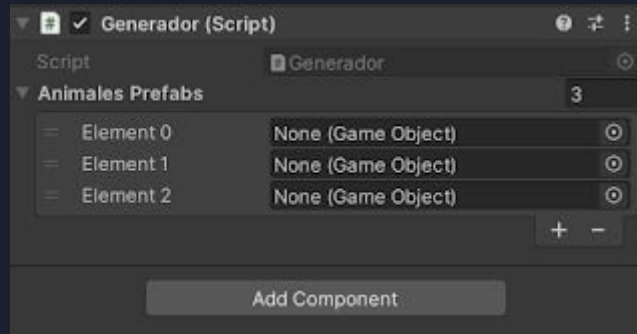
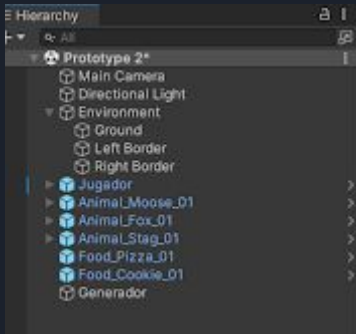


Proyecto. Creación de un juego real

Para generar animales de manera aleatoria, vamos a añadir un nuevo objeto a la escena. Create Empty (desde la vista de jerarquía) y le llamamos Generador. También crearemos un script llamado Generador.cs, el cual abriremos con VS Code y le añadiremos este atributo:

```
public GameObject[] animalesPrefabs;
```

Una vez hecho, seleccionaremos el objeto Generador y modificaremos en el panel derecho (Inspector) el tamaño del Array a 3. Y en cada uno de los elementos arrastraremos el prefab de cada animal.



Proyecto. Creación de un juego real

Para que aparezca un animal aleatorio (uno de los 3) en una posición aleatoria (eje X dentro de los rangos -20,20) al pulsar una tecla (S por ejemplo), modificaremos de nuevo el código:



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

```
public class Generator : MonoBehaviour {
```

```
    public GameObject[] animalesPrefabs;  
    private float rangoXGenerator = 20f;  
    private float posZGenerator = 20f;
```

```
    // Start is called before the first frame update
```

```
    void Start()
```

```
{
```

```
}
```

```
    // Update is called once per frame
```

```
    void Update()
```

```
{
```

```
        if(Input.GetKeyDown(KeyCode.S)){
```

```
            GenerarAnimalAleatorio();
```

```
        }
```

```
}
```

```
    void GenerarAnimalAleatorio(){
```

```
        int indexAnimal = Random.Range(0,animalesPrefabs.Length);
```

```
        Vector3 posicionGenerator = new
```


```
Vector3(Random.Range(-rangoXGenerator, rangoXGenerator), 0, posZGenerator);
```

```
        //Generar un animal
```

```
        Instantiate(animalesPrefabs[indexAnimal], posicionGenerator,  
animalesPrefabs[indexAnimal].transform.rotation);
```

```
    }
```

```
}
```



Proyecto. Creación de un juego real

Mejoraremos nuestro juego para que los animales vayan apareciendo cada cierto intervalo, y no tener que estar pulsando la tecla S. Por ejemplo, que pasados 2", vayan apareciendo cada 1,5".

Crearemos un método y lo invocaremos cada cierto tiempo.

💡 **Cómo funciona InvokeRepetating**
<https://docs.unity3d.com/ScriptReference/MonoBehaviour.InvokeRepeating.html>

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

```
public class Generador : MonoBehaviour {  
  
    public GameObject[] animalesPrefabs;  
    private float rangoXGenerador = 20f;  
    private float posZGenerador = 20f;  
    private float retardoInicial = 2.0f;  
    private float intervaloGeneracion = 1.5f;  
  
    // Start is called before the first frame update  
    void Start()  
    {  
        InvokeRepeating("GenerarAnimalAleatorio", retardoInicial,  
            intervaloGeneracion);  
    }  
  
    void GenerarAnimalAleatorio(){  
        int indexAnimal = Random.Range(0,animalesPrefabs.Length);  
        Vector3 posicionGenerador = new  
Vector3(Random.Range(-rangoXGenerador, rangoXGenerador), 0,  
posZGenerador);  
        //Generar un animal  
        Instantiate(animalesPrefabs[indexAnimal], posicionGenerador,  
animalesPrefabs[indexAnimal].transform.rotation);  
    }  
}
```

Proyecto. Creación de un juego real

En este momento los animales “atraviesan” al granjero y los trozos de pizza/galleta a los animales, es decir, no se detecta ninguna colisión.

Para ello seleccionaremos uno de nuestros animales desde el panel de jerarquía y le añadiremos el componente Box Collider. Aunque tenemos que ajustar su caja de colisión.



Proyecto. Creación de un juego real

Una vez ajustada su caja, le aplicamos este componente a todas sus instancias (prefabs), Overrides>Apply All.

Ahora vamos a seleccionar ese Prefab (no el objeto de jerarquía) y en su componente Box Collider marcaremos ☒ Is Trigger, para indicarle que es un disparador y podamos actuar cuando un objeto lo colisione (como por ejemplo hacerlo desaparecer).

Hacemos lo mismo con los otros dos animales.

También con la pizza y con la galleta, aunque estos podemos hacerlo directamente desde los prefabs.





Proyecto. Creación de un juego real

Para que las colisiones tengan efecto real, tenemos que *añadir física* a los elementos, hacerlos *cuerpos rígidos*.

Desde los prefabs (no objetos) añadiremos el componente Rigidbody y desmarcaremos la casilla Use Gravity para evitar que se hunda.



Proyecto. Creación de un juego real

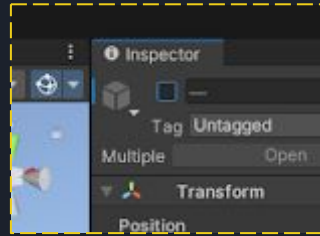
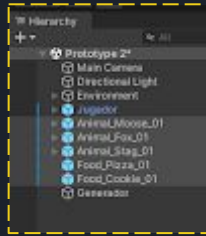
Una vez hecho esto, vamos a hacer que los elementos desaparezcan cuando entren en colisión. Para ello creamos el script **DetectorColisiones** y se lo añadimos a los 5 prefabs.

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

public class DetectorColisiones : MonoBehaviour
{
    private void OnTriggerEnter(Collider other){
        Destroy(gameObject);
    }
}
```

Proyecto. Creación de un juego real

Para que los trozos de comida y los animales no aparezcan desde el principio, los seleccionaremos y los *deshabilitaremos* del Inspector.



Para ir finalizando el juego, haremos que aparezca un mensaje en la consola cuando alguno de los animales llegue al final. Para ello abriremos el script `DestruirFueraEscena.cs`



Proyecto. Creación de un juego real

Y modificamos el método update() a:

```
void Update()
{
    if(transform.position.z > limiteSuperior){
        Destroy(gameObject);
    }else if( transform.position.z < limiteInferior){
        Destroy(gameObject);
        Debug.Log("Fin de partida");
    }
}
```


Proyecto. Creación de un juego real

Ahora vemos cómo los animales aparecen de manera aleatoria y cuando alguno consigue llegar al final de su recorrido, aparece el siguiente mensaje:

