Prototipo 2: Granja

1. Antes de empezar:
   - Crear un nuevo proyecto en Unity (selecciona el Template 3D)
   - Importar el paquete correspondiente al prototipo (Assets > Import Package > Custom Package).

2. Configuramos la escena:
   - Ahora añadimos una figura humana, 3 animales y un objeto comida al proyecto.
   - Renombra al personaje humano como Player.
   - Distribuye los personajes por la escena y modifica la escala del objeto comida para que se vea con facilidad.

3. Configuramos el control del personaje para moverlo de izquierda a derecha:
   - Creamos un script para el Player con nombre PlayerController y lo añadimos como componente del mismo.
   - Usamos el método Input.GetAxis:

```
public class PlayerController : MonoBehaviour
{
    public float horizontalInput;
    public float speed = 10.0f;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        horizontalInput = Input.GetAxis("Horizontal");
        transform.Translate(Vector3.right * horizontalInput * Time.deltaTime * speed);
    }
}
```

- Si ejecutamos podemos comprobar que el personaje se mueve de derecha a izquierda y puede salirse de los márgenes. Vamos a modificar el script para que el personaje se mantenga dentro del escenario.

```
public class PlayerController : MonoBehaviour
{
    public float horizontalInput;
    public float speed = 10.0f;
    public float xRange = 25;
    // Start is called before the first frame update
    void Start()
    {

    }
    // Update is called once per frame
    void Update()
    {
        horizontalInput = Input.GetAxis("Horizontal");
        transform.Translate(Vector3.right * horizontalInput * Time.deltaTime * speed);

        if (transform.position.x < -xRange)
        {
            transform.position = new Vector3(-xRange, transform.position.y,
transform.position.z);
        }
        if (transform.position.x > xRange)
        {
            transform.position = new Vector3(xRange, transform.position.y,
transform.position.z);
        }
    }
}
```

4. Vamos ahora a permitir que el personaje lance proyectiles: cuando se pulse la barra espaciadora se lanzará un proyectil a través de la escena y será destruido cuando abandone los límites de la escena.

- Creamos primero un script para controlar el movimiento del proyectil/comida y lo añadimos al objeto.

```
public class MoveForware : MonoBehaviour
{
    public float speed = 40;
    // Start is called before the first frame update
    void Start()
    {
    }
    // Update is called once per frame
    void Update()
    {
        transform.Translate(Vector3.forward * Time.deltaTime * speed);
    }
}
```

- Creamos un prefab a partir del proyectil que hemos configurado para poder reusarlo en cualquier momento:
  - Creamos un directorio Prefabs y arrastramos el objeto comida/proyectil. Escogemos **Original Prefab** cuando se nos de la opción.
  - Declaramos en el script del controlador del jugador **PlayerController** una variable **public GameObject projectilePrefab**
  - Seleccionamos al Player en la ventana de jerarquía y arrastramos el prefab hasta la variable que aparece ya en ella ventana inspector.
  - Añadimos en el script del jugador PlayerController código para comprobar la pulsación de la tecla espaciadora ***if (Input.GetKeyDown(KeyCode.Space))***

```
void Update()
{
    horizontalInput = Input.GetAxis("Horizontal");
    transform.Translate(Vector3.right * horizontalInput * Time.deltaTime * speed);

    if (transform.position.x < -xRange)
    {
        transform.position = new Vector3(-xRange, transform.position.y,
transform.position.z);
    }
    if (transform.position.x > xRange)
    {
        transform.position = new Vector3(xRange, transform.position.y,
transform.position.z);
    }
    if (Input.GetKeyDown(KeyCode.Space))
    {
        // Lanzamos proyectil
        Instantiate(projectilePrefab, transform.position,
projectilePrefab.transform.rotation);
    }
}
```

5. Vamos a convertir a los animales en prefabs de forma que los vamos a poder instanciar igual que al proyectil:

- Rotamos a los animales 180 grados para que estén frente al personaje.
- Les añadimos el script **MoveForward** para darles movimiento y editamos su propiedad velocidad.
- Finalmente los arrastramos al directorio **Prefabs.**

6. Debemos destruir los GameObjects que se salen de los límites del juego ya que ocupan memoria y recursos.

- En nuestro caso vamos a destruir los proyectiles una vez que se salen de los límites. Para ello creamos un nuevo script llamado **DestroyOutOfBounds** y lo añadimos como componente del proyectil:

```
public class DestroyOutOfBounds : MonoBehaviour
{
    private float topBound = 30;
    // Start is called before the first frame update
    void Start()
    {
    }
    // Update is called once per frame
    void Update()
    {
        if (transform.position.z > topBound)
        {
            Destroy(gameObject);
        }
    }
}
```

- Teniendo seleccionado el proyectil en la ventana de jerarquía seleccionamos **Apply all** en el menú desplegable **Overrides** para que el cambio se aplique al prefab.
- También debemos destruir a los animales que se salen por el límite inferior. Podemos modificar el script **DestroyOutOfBounds** para que destruya también a los objetos que sobrepasen este límite.

```
public class DestroyOutOfBounds : MonoBehaviour
{
    private float topBound = 30;
    private float lowerBound = -10;
    // Start is called before the first frame update
    void Start()
    {
    }
    // Update is called once per frame
    void Update()
    {
        if (transform.position.z > topBound)
        {
            Destroy(gameObject);
        }
        else if (transform.position.z < lowerBound)
        {
            Destroy(gameObject);
        }

    }
}
```

7. En este punto vamos a programar que los animales aparezcan aleatoriamente. Cuando se presione la tecla **s** se generará un animal que avanzará hacia el jugador:
   - Creamos un objeto vacío ***Empty object*** y lo llamamos ***SpawnManager***.
   - A continuación creamos un script y lo llamamos ***SpawnManager*** y lo añadimos al objeto vacío. El código que debe contener para empezar debe ser:

```
public class SpawnManager : MonoBehaviour
{
    public GameObject[] animalPrefabs;
    public int animalIndex;
    private float spawnRangeX = 20;
    private float spawnPosZ = 20;
    // Start is called before the first frame update
    void Start()
    {
    }
        // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.S))
        {
            Vector3 spawnPos = new Vector3(Random.Range(-spawnRangeX, spawnRangeX),0,
spawnPosZ);
            int animalIndex = Random.Range(0, animalPrefabs.Length);
            Instantiate(animalPrefabs[animalIndex], spawnPos,
animalPrefabs[animalIndex].transform.rotation);
        }
    }
}
```

- En la ventana de inspección del objeto vacío debemos ver el script y la lista vacía correspondiente al array declarado. Lo llenamos con los prefabs de los animales que tenemos creados desde la ventana de proyecto (no desde la ventana de jerarquía).
- Según leemos en el código, cada vez que se presione la tecla **s** se generará un animal en la parte superior de la pantalla.

8. Modificaciones finales y gestión de colisiones:
   - Vamos a modificar el juego para que los animales aparezcan automáticamente cada pocos segundos sin necesidad de pulsar una tecla.
   - A continuación vamos a añadir colliders a los prefabs de los animales para poder destruirlos con los proyectiles.
   - Finalmente vamos a añadir un mensaje de Game Over si algún animal sobrepasa la posición del personaje.

Vamos a empezar creando un nuevo método para generar los animales:

- Dentro del script SpawnManager creamos un método nuevo **void SpawnRandomAnimal() {}** debajo del método Update()

```
void Update()
{
   /* if (Input.GetKeyDown(KeyCode.S))
    {
        Vector3 spawnPos = new Vector3(Random.Range(-spawnRangeX, spawnRangeX),0,
spawnPosZ);
        int animalIndex = Random.Range(0, animalPrefabs.Length);
        Instantiate(animalPrefabs[animalIndex], spawnPos,
animalPrefabs[animalIndex].transform.rotation);
    }*/
}
void SpawnRandomAnimal()
{
    int animalIndex = Random.Range(0, animalPrefabs.Length);
```

```
    Vector3 spawnpos = new Vector3(Random.Range(-spawnRangeX,spawnRangeX), 0,
spawnPosZ);
    Instantiate(animalPrefabs[animalIndex], spawnpos,
animalPrefabs[animalIndex].transform.rotation);
}
```

- Para generar los animales a intervalos modificamos el método start del script:

```
public class SpawnManager : MonoBehaviour
{
    public GameObject[] animalPrefabs;
    public int animalIndex;
    private float spawnRangeX = 20;
    private float spawnPosZ = 20;
    private float startDelay = 2;
    private float spawnInterval = 1.5f;
    // Start is called before the first frame update
    void Start()
    {
        InvokeRepeating("SpawnRandomAnimal", startDelay, spawnInterval);
    }...
```

- Para detectar las colisiones debemos añadir a cada animal un **Box Collider** y ajustarlo para que ocupe su volumen aproximadamente. Debemos asegurarnos que el componente Box Collider tienen marcado el checkbox **Is Trigger.**
- Este proceso deberemos repetirlo para el proyectil. Además, a éste deberemos añadirle un componente **Rigid Body** y debemos desmarcar el uso de la gravedad para que los proyectiles vuelen.
- Una vez que tenemos los **Box Colliders** con sus **triggers** marcados debemos escribir un script para destruir los objetos con los impactos. Lo creamos con el nombre **DetectCollisions** y añadimos el siguiente código después del métdo Update y antes de la llave final:

```
void OnTriggerEnter(Collider other)
{
    Destroy(gameObject);
    Destroy(other.gameObject);
}
```

- Una vez lo tenemos, debemos añadir este script a cada animal.

9. Finalización del juego: podemos hacer que finalice el juego cuando algún animal sobrepasa la posición del personaje y mostrar un mensaje de Game Over