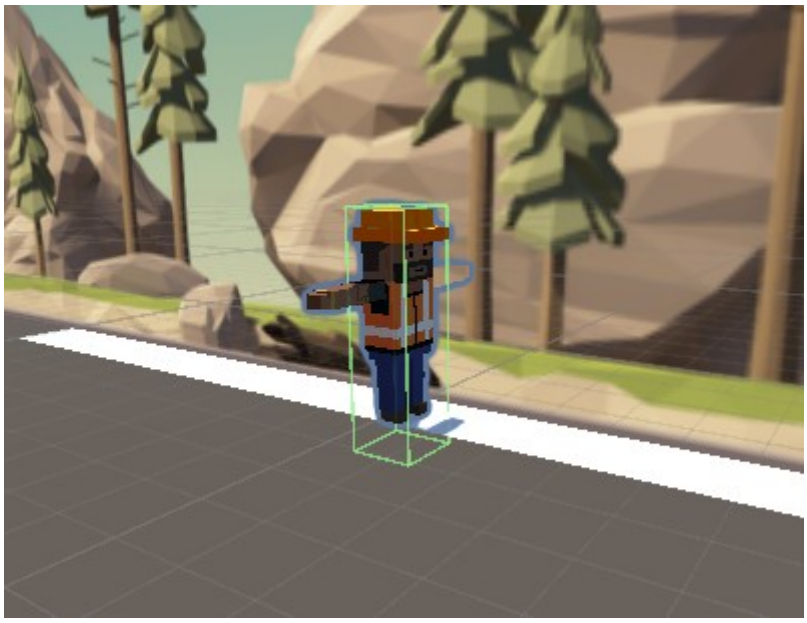
Prototipo 3

1. Lo primero que debemos hacer es configurar un nuevo proyecto, importar los archivos iniciales y elegir un fondo para el juego:

- Abre Unity Hub y crea un proyecto "Prototipo03" vacío en el directorio de trabajo con la versión adecuada de Unity e importa el paquete **Prototype 3 Starter Files**.
- Una vez en el proyecto abre la escena **Prototype 3** y elimina la escena **Sample Scene** sin guardar.
- Seleccione el objeto **Background object** en la ventana de jerarquía, luego en el componente **Sprite Renderer > Sprite**, seleccione una de las imágenes para el fondo: **Background_City**, **Background_Nature**, o **Background_Town**.

2. Ahora seleccionamos un personaje para el jugador:

- Desde la ventana de proyecto **Course Library > Characters**, arrastra un personaje a la ventana de jerarquía, cámbiela el nombre a "**Player**" y luego gíralo en el eje Y para mirar hacia la derecha.
- Añade al personaje un componente **Rigid Body** y un componente **Box Collider** (ajusta el tamaño del Box Collider al personaje).



- Crea una carpeta llamada **Scripts** en la ventana de proyectos para guardar los scripts. Dentro crea un script llamado **PlayerController** y añádelo al personaje.
- Vamos a editar el script y añadir código para hacer saltar al personaje cada vez que se pulse la tecla de espacio:

```
public class PlayerController : MonoBehaviour
{
	private Rigidbody playerRb;
	// Start is called before the first frame update
	void Start()
	{
		playerRb = GetComponent<Rigidbody>();
	}

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

```
            if (Input.GetKeyDown(KeyCode.Space))
            {
                playerRb.AddForce(Vector3.up * 100, ForceMode.Impulse);
            }
        }
}
```

- Necesitamos darle al jugador un salto perfecto ajustando la fuerza del salto, la gravedad del escena y la masa del personaje. Vamos a modificar el script y despues podemos ajustar las variables desde la ventana del inspector:

```
public class PlayerController : MonoBehaviour
{
    private Rigidbody playerRb;
    public float jumpForce;
    public float gravityModifier;
    // Start is called before the first frame update
    void Start()
    {
        playerRb = GetComponent<Rigidbody>();
        //playerRb.AddForce(Vector3.up * 100);
        Physics.gravity *= gravityModifier;
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            playerRb.AddForce(Vector3.up * jumpForce, ForceMode.Impulse);
        }
    }
}
```

- Ahora mismo, cada vez que pulsamos la barra espaciadora, el personaje salta independientemente de si está en el suelo o no. Debemos evitar este comportamiento. Para ello vamos a introducir una estructura **if** para preguntar antes de saltar si estamos en el suelo:

```
public class PlayerController : MonoBehaviour
{
    private Rigidbody playerRb;
    public float jumpForce;
    public float gravityModifier;
    public bool isOnGround = true;
    // Start is called before the first frame update
    void Start()
    {
        playerRb = GetComponent<Rigidbody>();
        //playerRb.AddForce(Vector3.up * 100);
        Physics.gravity *= gravityModifier;
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space) && isOnGround)
        {
            playerRb.AddForce(Vector3.up * jumpForce, ForceMode.Impulse);
            isOnGround = false;
        }
    }
    private void OnCollisionEnter(Collision collision)
```

```
        {
            isOnGround = true;
        }
    }
```

3. Una vez que el personaje salta, vamos a introducir obstáculos que se muevan hacia él:

- Desde **Course Library** > **Obstacles**, agrega un obstáculo y cámbiale el nombre a "**Obstacle**" y colócalo donde debería aparecer.
- Añade al obstáculo un componente **Rigid Body** y un **Box Collider** (edita los límites del colisionador para que se ajusten al obstáculo).
- Crea una nueva carpeta "**Prefabs**" y arrastra el obstáculo de la escena a la carpeta como **Original Prefab**.
- Crea un nuevo script "**MoveLeft**", aplícalo al obstáculo y ábrelo.
- En el script **MoveLeft**, escribe el código para modificar su **Translate** a la izquierda según una variable de velocidad. Guarda las características del prefab (Overrides).
- Aplique el script **MoveLeft** al fondo también.
- Si guardamos y damos Play veremos como el obstáculo y el fondo avanzan hacia la izquierda.

```
public class MoveLeft : MonoBehaviour
{
    private float speed = 10;
    // Start is called before the first frame update
    void Start()
    {
    }

    // Update is called once per frame
    void Update()
    {
        transform.Translate(Vector3.left * Time.deltaTime * speed);

    }
}
```

4. De manera similar al proyecto anterior, necesitamos crear un objeto vacío **SpawnManager** que cree instancias de obstáculos a partir de su Prefab:

- Crea un nuevo objeto vacío "**SpawnManager**" y luego aplícale un nuevo script **SpawnManager**.
- En el script **SpawnManager**, declara un nuevo **public GameObject obstaclePrefab;**, luego asigna tu prefab del obstáculo a la nueva variable en la ventana del inspector.
- Declara un vector **private Vector3 spawnPos** apuntando a la posición donde se debe generar el obstáculo.
- En el método **Start**, escribe el código que crea las instancia de un nuevo obstáculo prefabricado, luego puedes eliminar el prefab de la escena y probar a ejecutar.

```
public class SpawnManager : MonoBehaviour
{
    public GameObject obstaclePrefab;
    private Vector3 spawnPos = new Vector3(25, 0, 0);
    private float startDelay = 2;
    private float repeatRate = 2;
    // Start is called before the first frame update
    void Start()
    {
        //Instantiate(obstaclePrefab, spawnPos, obstaclePrefab.transform.rotation);
        InvokeRepeating("SpawnObstacle", startDelay, repeatRate);
    }
```

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

    }
    void SpawnObstacle()
    {
        Instantiate(obstaclePrefab, spawnPos, obstaclePrefab.transform.rotation);
    }
}
```

5. Ahora vamos a repetir el fondo para que se mueva hacia la izquierda a la misma velocidad que los obstáculos y parezca que no deja de pasar ante nosotros en un bucle:

- Crea un nuevo script llamado **RepeatBackground** y añádelo al objeto de fondo.

Para repetir el fondo y proporcionar la ilusión de un mundo que pasa rápidamente, necesitamos restablecer la posición del objeto de fondo para que encaje perfectamente.

- Declara en el script una nueva variable privada **private Vector3 startPos;**
- En el método Start(), establece la variable startPos en su posición inicial real asignándole = **transform.position;**
- En el método Update(), escribe una estructura **if** para restablecer la posición si se mueve una cierta distancia.

```
private Vector3 startPos;
void Start()
	{
		startPos = transform.position;
	}
void Update()
	{
		if (transform.position.x < startPos.x - 50)
		{
			transform.position = startPos;
		}
	}
```

El fondo se repite cada pocos segundos, pero la transición está bastante mal conseguida. Necesitamos hacer que el fondo se repita de manera perfecta y fluida con algunas variables nuevas. Vamos a modificar el script **RepeatBackground:**

- Agregar un componente **Box Collider** al fondo
- Declara una nueva variable **private float repeatWidth**
- En el método **Start**, obtén el ancho del **BoxCollider**, divídelo por 2.
- Incorpora la variable **repeatWidth** en la función **repeat**

```
public class NewBehaviourScript : MonoBehaviour
{
    private Vector3 startPos;
    private float repeatWidth;
    // Start is called before the first frame update
    void Start()
    {
        startPos = transform.position;
        repeatWidth = GetComponent<BoxCollider>().size.x / 2;
    }

    // Update is called once per frame
    void Update()
    {
        //if (transform.position.x < startPos.x - 40)
```

```
        if (transform.position.x < startPos.x - repeatWidth)
        {
            transform.position = startPos;
        }
    }
}
```

6. Cuando el jugador choca con un obstáculo, queremos activar un estado de "Game Over" que detenga todo. Para hacerlo, necesitamos una forma de etiquetar y distinguir todos los objetos del juego con los que puede chocar el jugador:

- En la ventana del inspector, agrega una etiqueta de "**Ground**" al objeto Ground y una etiqueta de "**Obstacle**" al prefab de los obstáculos (**tienes que crear las etiquetas primero**).
- En el script **PlayerController**, declara una nueva variable **public bool gameOver;**
- En el método **OnCollisionEnter** del script agrega la estructura **if-else** para probar si el jugador chocó con el "**Ground**" o con un "**Obstacle**".
- Si choca con el suelo", establecemos **isOnGround = true**, y si choca con un obstáculo, establecemos **gameOver = true.**

```
public class PlayerController : MonoBehaviour
{
    private Rigidbody playerRb;
    public float jumpForce;
    public float gravityModifier;
    public bool isOnGround = true;
    public bool gameOver = false;
    // Start is called before the first frame update
    void Start()
    {
        playerRb = GetComponent<Rigidbody>();
        //playerRb.AddForce(Vector3.up * 100);
        Physics.gravity *= gravityModifier;
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space) && isOnGround)
        {
            playerRb.AddForce(Vector3.up * jumpForce, ForceMode.Impulse);
            isOnGround = false;
        }
    }
    private void OnCollisionEnter(Collision collision)
    {
        //isOnGround = true;
        if (collision.gameObject.CompareTag("Ground"))
        {
            isOnGround = true;
        }
        else if (collision.gameObject.CompareTag("Obstacle"))
        {
            gameOver = true;
            Debug.Log("Game Over!");
        }
    }
}
```

Hemos agregado una variable booleana que activa el estado de **gameOver** que parece funcionar, pero el fondo y los objetos continúan moviéndose. Necesitamos que el script **MoveLeft** se comunique con el script **PlayerController** y detener el movimiento una vez que el jugador dispare o active el **gameOver**.

- En el script **MoveLeft,** declara una nueva variable **private PlayerController playerControllerScript;**
- En el método **Start()**, inicializalo buscando el objeto **Player** y obteniendo el componente **PlayerController**.
- Introduce en una estructura **if** el método **transform.Translate** comprobando si el juego ha terminado

```
public class MoveLeft : MonoBehaviour
{
    private float speed = 10;
    private PlayerController playerControllerScript;
    // Start is called before the first frame update
    void Start()
    {
        playerControllerScript =
GameObject.Find("Player").GetComponent<PlayerController>();
    }

    // Update is called once per frame
    void Update()
    {
        //transform.Translate(Vector3.left * Time.deltaTime * speed);
        if (playerControllerScript.gameOver == false)
        {
            transform.Translate(Vector3.left * Time.deltaTime * speed);
        }
    }
}
```

El fondo y los obstáculos dejan de moverse cuando **gameOver == true**, pero el script **SpawnManager** no deja de generar un ejército de obstáculos. Necesitamos comunicarnos con el script **Spawn Manager** y decirle que se detenga cuando termine el juego.

- En el script **SpawnManager**, obtén una referencia al script **playerController** usando la misma técnica que para el script **MoveLeft** en los pasos anteriores.
- Agrega una condición para crear instancias de objetos solo si **gameOver == false**

```
public class SpawnManager : MonoBehaviour
{
    public GameObject obstaclePrefab;
    private Vector3 spawnPos = new Vector3(25, 0, 0);
    private float startDelay = 2;
    private float repeatRate = 2;
    private PlayerController playerControllerScript;
    // Start is called before the first frame update
    void Start()
    {
        //Instantiate(obstaclePrefab, spawnPos, obstaclePrefab.transform.rotation);
        InvokeRepeating("SpawnObstacle", startDelay, repeatRate);
        playerControllerScript =
GameObject.Find("Player").GetComponent<PlayerController>();
    }

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

    }
    void SpawnObstacle()
    {
        //Instantiate(obstaclePrefab, spawnPos, obstaclePrefab.transform.rotation);
        if (playerControllerScript.gameOver == false)
        {
```

```
            Instantiate(obstaclePrefab, spawnPos, obstaclePrefab.transform.rotation);
        }
    }
}
```

7. Por último, debemos destruir los obstáculos que vamos saltando cuando salen de un cierto límite como hacíamos con los animales del prototipo 2:

- En el script **MoveLeft**, en el método **Update()**; escribimos una estructura **if** para destruir los obstáculos si su posición es menor que una variable que vamos a llamar **leftBound** .

```
public class MoveLeft : MonoBehaviour
{
    private float speed = 10;
    private PlayerController playerControllerScript;
    private float leftBound = -15;
    // Start is called before the first frame update
    void Start()
    {
        playerControllerScript =
GameObject.Find("Player").GetComponent<PlayerController>();
    }

    // Update is called once per frame
    void Update()
    {
        //transform.Translate(Vector3.left * Time.deltaTime * speed);
        if (playerControllerScript.gameOver == false)
        {
            transform.Translate(Vector3.left * Time.deltaTime * speed);
        }
        if (transform.position.x < leftBound && gameObject.CompareTag("Obstacle"))
        {
            Destroy(gameObject);
        }
    }
}
```