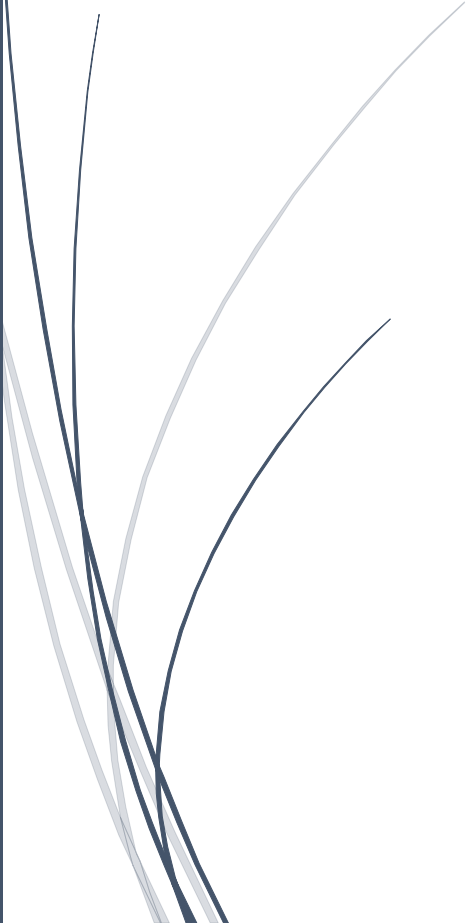
A dark blue vertical bar on the left side of the page. A blue arrow points to the right from the bar, containing the date.

17-11-2023

# DOCUMENTACIÓN – RA3 – APLICACIÓN 3D

NetGenius Company

Several thin, curved lines in dark blue and light grey originate from the bottom left and sweep upwards and to the right.

Adrián Peña Carnero, Álvaro González, Pablo  
Palacio.

NETGENIUS COMPANY

## Contenido

b) Se han reconocido las clases que permiten la captura, procesamiento y almacenamiento de datos multimedia. ....	2
c) Se han utilizado clases para la conversión de datos multimedia de un formato a otro. ....	4
d) Se han utilizado clases para construir procesadores para la transformación de las fuentes de datos multimedia. ....	4
e) Se han utilizado clases para el control de eventos, tipos de media y excepciones, entre otros. ....	4
f) Se han utilizado clases para la creación y control de animaciones. ....	5
g) Se han utilizado clases para construir reproductores de contenidos multimedia. ....	11
h) Se han depurado y documentado los programas desarrollados. ....	13

## b) Se han reconocido las clases que permiten la captura, procesamiento y almacenamiento de datos multimedia.

En nuestro proyecto no hemos utilizado ninguna clase relacionada con este punto.

Para la captura y almacenamiento de datos multimedia hay que crear un objeto vacío y crear un script como este:

```
using UnityEngine;
```

```
public class CameraCaptureScript : MonoBehaviour
```

```
{
```

```
    // Referencia a la textura donde mostraremos la imagen de la cámara
```

```
    public Renderer previewRenderer;
```

```
    void Start()
```

```
{
```

```
    // Iniciamos la captura de la cámara al inicio
```

```
    StartCameraCapture();
```

```
}
```

```
    void StartCameraCapture()
```

```
{
```

```
    // Verificamos si la cámara está disponible
```

```
    if (WebCamTexture.devices.Length > 0)
```

```
{
```

```

        // Tomamos la primera cámara disponible

        WebCamTexture webcamTexture = new
WebCamTexture(WebCamTexture.devices[0].name);

        // Mostramos la vista previa en el objeto Renderer

        previewRenderer.material.mainTexture = webcamTexture;

        // Iniciamos la captura de la cámara

        webcamTexture.Play();
    }
    else
    {
        Debug.LogError("No se encontraron cámaras disponibles.");
    }
}
}
}

```

Adjuntamos este script al objeto vacío y Seleccionamos el objeto en el que quieres mostrar en la vista previa de la cámara.

### c) Se han utilizado clases para la conversión de datos multimedia de un formato a otro.

Las operaciones de conversión de datos multimedia en Unity generalmente se realizan usando la clase `Texture2D` para imágenes y diferentes clases para operaciones de audio y video. Unity facilita la conversión de formatos utilizando métodos como `EncodeToPNG` para imágenes, que convierte texturas en matrices de bytes en formato PNG. La función `ConvertTextureToBytes` toma una textura y la convierte en una matriz de bytes. Esto puede resultar útil para almacenar imágenes en una base de datos, por ejemplo.

### d) Se han utilizado clases para construir procesadores para la transformación de las fuentes de datos multimedia.

Cuando se necesita realizar transformaciones específicas en datos multimedia, se pueden crear clases dedicadas para actuar como procesadores. Por ejemplo, un procesador de imágenes podría aplicar filtros o efectos visuales. En este caso, se podría tener una clase `ImageProcessor` que realiza una transformación simple, como convertir una imagen a blanco y negro.

### e) Se han utilizado clases para el control de eventos, tipos de media y excepciones, entre otros.

Unity pasa el control a un script de forma intermitente al llamar a ciertas funciones que están declaradas dentro de él. Una vez que una función ha terminado de ejecutarse, el control se pasa de nuevo a Unity. Estas funciones se conocen como funciones de eventos, ya que son activadas por Unity en respuesta a los eventos que ocurren durante el juego. Unity usa un esquema de nombres para identificar qué función llamar para un evento en particular. Por ejemplo, ya habrá visto la función de actualización (llamada antes de que ocurra una actualización de marco) y la función de inicio (llamada justo antes de la primera actualización de marco del objeto). Muchas más funciones de eventos están disponibles en Unity; la lista completa se puede encontrar en la página de referencia del script para la clase `[MonoBehaviour]` ([../ScriptReference/ MonoBehaviour.html](#)) junto con los detalles de su uso. Los siguientes son algunos de los eventos más comunes e importantes.

Control de Eventos:

En Unity, el control de eventos se basa en el uso de delegados y eventos. Los delegados son tipos que representan referencias a métodos y los eventos son una forma especial de delegado que proporciona una capa adicional de seguridad y abstracción. Los eventos permiten que diferentes partes de tu código se comuniquen sin acoplarse directamente.

Tipos de Media:

Unity ofrece diversas clases para manejar diferentes tipos de media, como imágenes, audio y video. Algunas de las clases más comunes son:

Imágenes (Texturas): Texture2D es ampliamente utilizada para manipular imágenes. Puedes cargar, modificar y mostrar imágenes en tu juego utilizando esta clase.

Audio: AudioClip es la clase principal para trabajar con sonido en Unity. Puedes adjuntar clips de audio a objetos en tu escena y controlar su reproducción.

Video: Unity proporciona la clase VideoPlayer para reproducir videos en tiempo real. Puedes cargar videos desde archivos locales o transmitirlos desde la web.

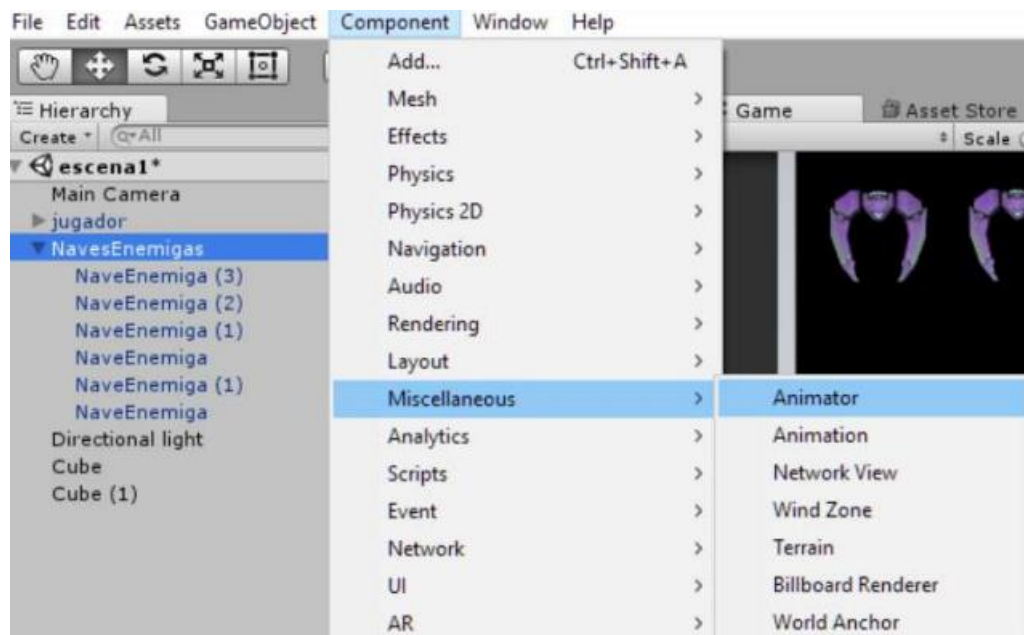
Excepciones:

En Unity, puedes utilizar bloques try-catch para manejar excepciones y controlar errores de manera efectiva. Esto es especialmente importante al realizar operaciones que pueden generar excepciones, como la división por cero.

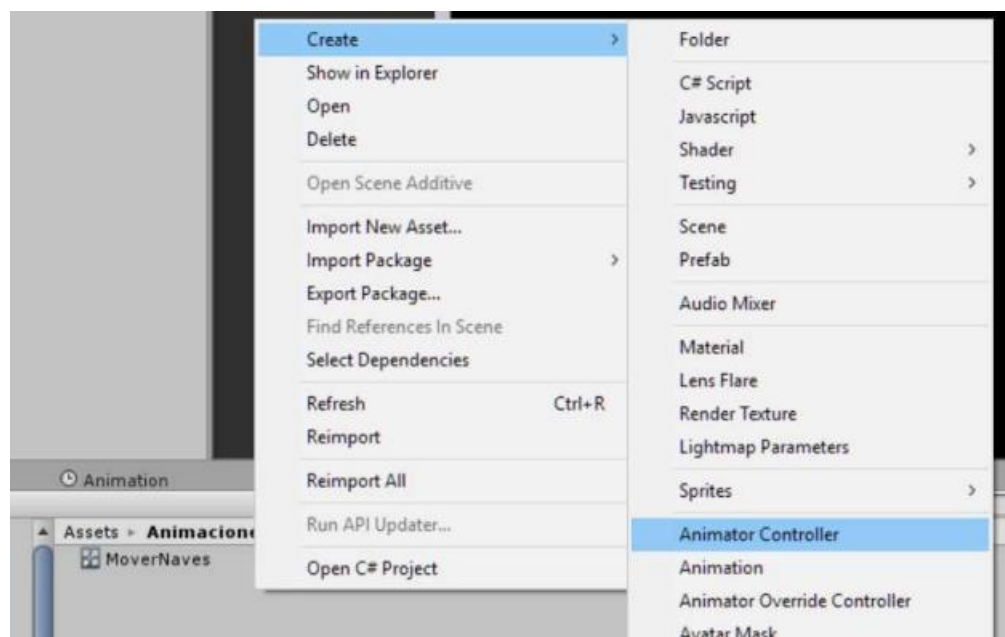
## f) Se han utilizado clases para la creación y control de animaciones.

En nuestro caso ninguno de los tres hemos realizado ninguna clase para la creación y control de animaciones, pero no obstante te muestro como podríamos realizarlo:

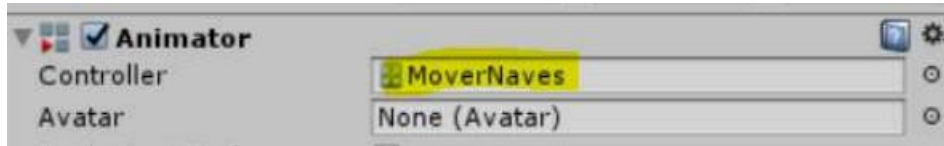
Al agregar efectos visuales a los elementos de la escena mediante animaciones mejora la calidad visual y proporciona dinámicas más entretenida. Se trata de un componente esencial en cualquier videojuego. Unity ofrece un animador integral con una máquina de estados que facilita la creación de animaciones detalladas, las cuales pueden ser implementadas a través de scripts en nuestra escena.



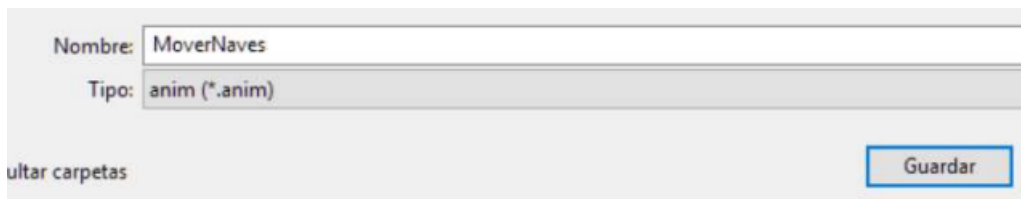
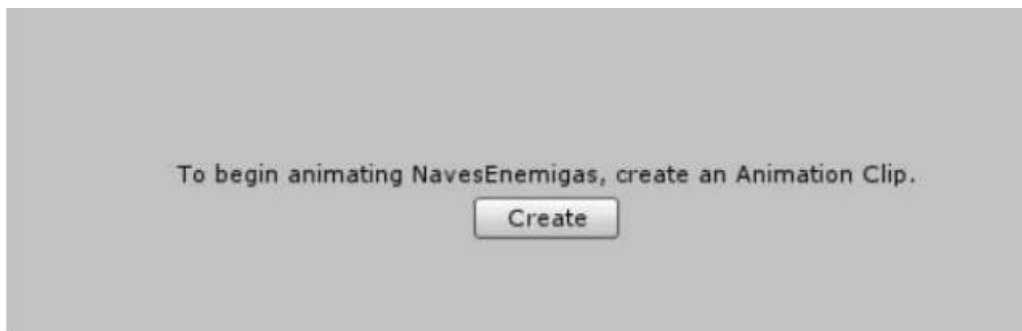
En nuestros Assets creamos “Animator Controler” para ello , creamos una carpeta llamada Animaciones para tenerlo organizado.



Le llamamos como queramos en este caso MoverNaves y seleccionamos el objeto creado llamado NavesEnemigas en la jerarquía y añadimos el Animator Controller a su componente Controller.



Ahora simplemente al seleccionar la pestaña Animation le damos a créate AnimationClip y lo guardamos como MoverNaves.



Y ahora seria animarlo y configurarlo como nosotros queramos, aquí te dejo una visión general del sistema de animación:

Unity cuenta con un sistema de animación avanzado llamado Mecanim, que ofrece:



Un flujo de trabajo sencillo y configuraciones de animación para todos los elementos de Unity, como objetos, personajes y propiedades.

Soporte para clips de animación importados y animaciones creadas dentro de Unity.

Retargeting de animación humanoide, lo que permite aplicar animaciones de un modelo de personaje a otro.

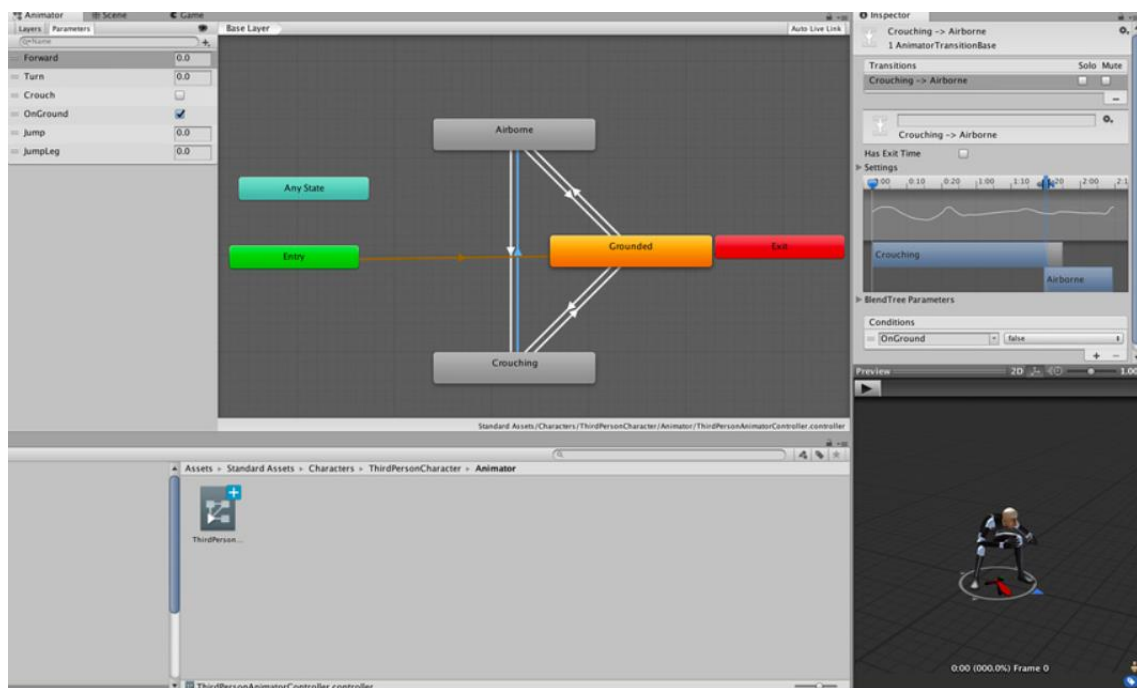
Un proceso simplificado para alinear clips de animación.

Una vista previa práctica de clips de animación, transiciones e interacciones, permitiendo a los animadores trabajar de manera más independiente antes de la implementación del código del juego.

Manejo de interacciones complejas entre animaciones mediante una herramienta visual de programación.

Capacidad para animar diferentes partes del cuerpo con lógicas distintas.

Funcionalidades de capas (layering) y enmascaramiento (masking).



Y el flujo de trabajo de la Animación:

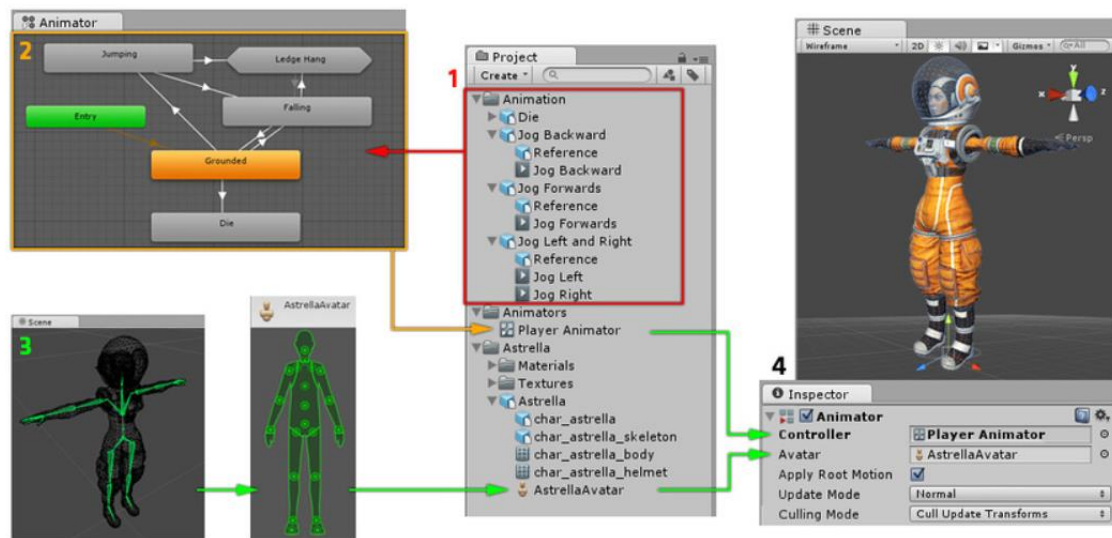
El sistema de animación de Unity se basa en el concepto de "Animation Clips" que contienen información sobre cómo ciertos objetos deben cambiar su posición, rotación u otras propiedades a lo largo del tiempo. Estos clips son grabaciones lineales individuales, creadas por artistas o animadores con herramientas de terceros como Autodesk® 3ds Max® o Autodesk® Maya®, o provenientes de estudios de captura de movimiento u otras fuentes.

Estos clips se organizan en un sistema estructurado similar a un diagrama de flujo llamado "Animator Controller". Este actúa como una "Máquina de Estados" que controla qué clip debe reproducirse en un momento dado y cuándo deben cambiar o mezclarse las animaciones.

Un Animator Controller simple podría contener uno o dos clips, como para controlar un powerup o animar una puerta. En cambio, un Animator Controller más avanzado podría contener muchas animaciones humanoides para las acciones de personajes principales, mezclándose para proporcionar movimientos fluidos mientras el jugador se desplaza por la escena.

El sistema de animación de Unity también cuenta con características especiales para manejar personajes humanoides, permitiendo retargeting de animaciones humanoides desde cualquier fuente (como captura de movimiento, Asset Store u otras bibliotecas de animación de terceros) hacia tu propio modelo de personaje, así como ajustar definiciones musculares. Estas características especiales son posibles gracias al sistema de Avatares de Unity, donde los personajes humanoides se asignan a un formato interno común.

Estos elementos, los "Animation Clips", el "Animator Controller" y el "Avatar", se combinan en un GameObject a través del componente Animator. Este componente hace referencia a un Animator Controller.



En este diagrama podemos ver:

En resumen, el proceso de animación en Unity implica los siguientes pasos:

Se importan clips de animación desde una fuente externa o se crean dentro de Unity, como animaciones humanoides capturadas en movimiento.

Estos clips se organizan en un Animator Controller, que actúa como un conjunto de estados conectados, representados como nodos en una vista visual en la ventana Animator. Este Animator Controller es un activo en la ventana del Proyecto.

El modelo del personaje, que está riggeado (en este caso, el astronauta "Astrella"), tiene una configuración de huesos específica que se mapea en el formato Avatar común de Unity. Este mapeo se almacena como un activo Avatar como parte del modelo del personaje importado.

Al animar el modelo del personaje, se le adjunta un componente Animator. En la vista del Inspector, se muestra el componente Animator con asignados tanto el Animator Controller como el Avatar. El animador utiliza estos elementos en conjunto para animar el modelo. La referencia al Avatar solo es necesaria al animar un personaje humanoide; para otros tipos de animación, solo se requiere un Animator Controller.

El sistema de animación de Unity incluye varios conceptos y terminología, y si surge la necesidad de entender algún término, se puede consultar el Glosario de Animación proporcionado por Unity.

## g) Se han utilizado clases para construir reproductores de contenidos multimedia.

Nosotros no hemos utilizado ninguna clase para construir ningún reproductor de contenido multimedia, pero le mostramos como podríamos construirlo y utilizarlo:

En Unity, puedes construir reproductores de contenidos multimedia utilizando clases. Aquí hay un resumen de cómo hacerlo:

### **Importa las Bibliotecas Necesarias:**

Asegúrate de importar las bibliotecas necesarias para trabajar con multimedia en Unity, como UnityEngine y otras bibliotecas específicas según tus necesidades.

### **Crea una Clase para el Reproductor:**

Define una clase específica para tu reproductor de contenidos multimedia. Puedes llamar a esta clase, por ejemplo, ReproductorMultimedia.

### **Añade Funcionalidades:**

Dentro de la clase ReproductorMultimedia, implementa funciones o métodos que te permitan cargar, reproducir, pausar, detener y gestionar otros aspectos relacionados con el contenido multimedia.

Ejemplo:

```
public class ReproductorMultimedia
{
    // Métodos para cargar, reproducir, pausar, detener, etc.
```

```

public void CargarContenido(string rutaContenido) { /* Implementación */ }

public void Reproducir() { /* Implementación */ }

public void Pausar() { /* Implementación */ }

public void Detener() { /* Implementación */ }

}

```

### Implementa la Lógica de Reproducción:

Dentro de cada método, implementa la lógica específica para cargar y reproducir el contenido multimedia. Esto puede incluir el uso de la clase MediaPlayer de Unity o cualquier otra biblioteca que estés utilizando.

### Integra con Unity:

Ahora, puedes crear un objeto en tu escena de Unity y adjuntarle el script del reproductor multimedia. Desde el Editor de Unity, podrás ajustar parámetros y llamar a los métodos del reproductor según sea necesario.

```

public class ControladorEscena : MonoBehaviour
{
    private ReproductorMultimedia reproductor;

    void Start()
    {
        reproductor = new ReproductorMultimedia();
        reproductor.CargarContenido("ruta/al/archivo.mp4");
        reproductor.Reproducir();
    }
}

```

Estos pasos proporcionan una estructura básica para construir un reproductor de contenidos multimedia en Unity utilizando clases y scripts de C#. Asegúrate de ajustar la implementación según tus necesidades específicas y las características multimedia que desees incluir.

## h) Se han depurado y documentado los programas desarrollados.

Nosotros solo hemos documentado en las clases brevemente lo que hemos desarrollado

Para depurar y comentar los programas desarrollados se haría así:

Inicia Unity y abre tu proyecto. Abre el script que desees depurar haciendo doble clic en él desde Unity. Posteriormente, dentro del editor de código, selecciona "Open C# Project" para abrir el proyecto en Visual Studio.

Examina tu código en busca de secciones críticas o áreas propensas a errores. Coloca puntos de interrupción haciendo clic en la barra izquierda del editor de código, junto a las líneas donde desees detener la ejecución.

Asegúrate de que Unity esté en ejecución con tu proyecto abierto. En Visual Studio, busca y haz clic en el botón "Attach to Unity" en la barra de herramientas de depuración.

Haz clic en el botón "Play" en Unity para ejecutar tu proyecto. Visual Studio detendrá la ejecución en los puntos de interrupción, permitiéndote inspeccionar variables y seguir el flujo del programa. Utiliza la ventana de depuración en Visual Studio para explorar variables y realizar correcciones en tiempo real.

Después de realizar correcciones, haz clic en el botón "Continue" en la barra de herramientas de depuración para reanudar la ejecución del programa.