

Bienvenido, Alan Manuel Mendoza Arredondo

Administra tu información y las opciones de privacidad y seguridad a fin de que Google sea más relevante para ti. [Más información](#)

Privacidad y personalización
Para personalizar tu experiencia en Google, elige la

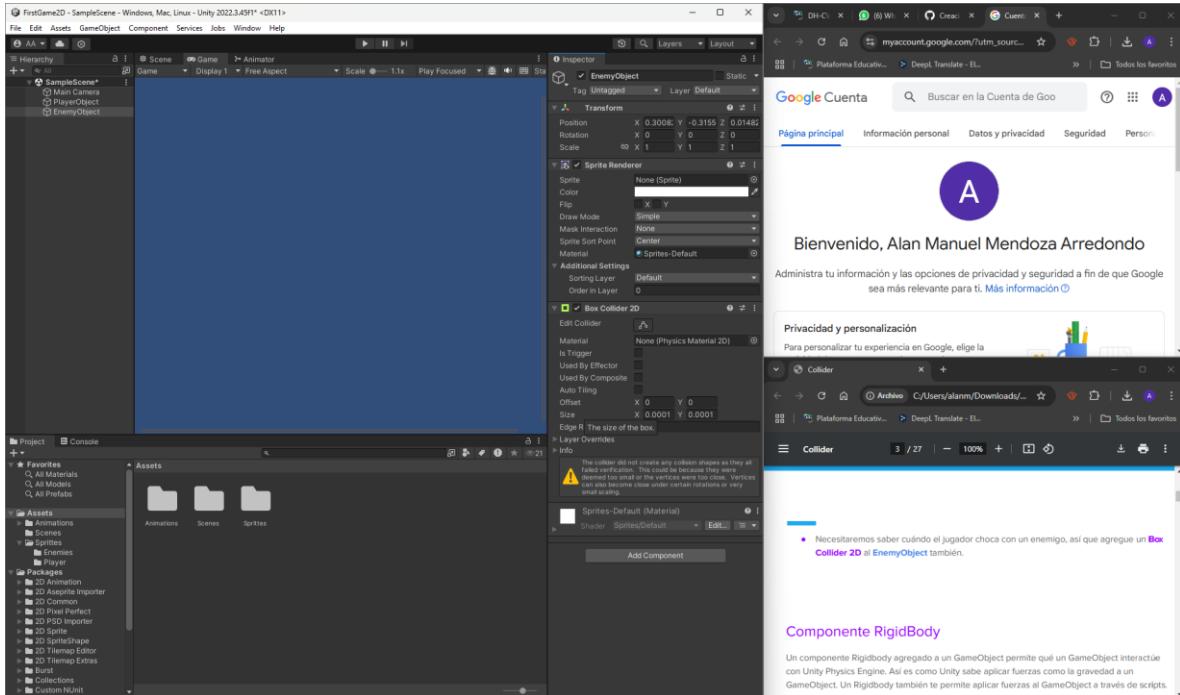
Collider

The collider did not create any collision shapes as they all failed verification. This could be because they were too small or the vertices were too close. Vertices can also become close under certain rotations or very small scaling.

Una aproximación de la forma de los objetos es usando un tipo de colisionador llamado "Primitive Collider" es menos intenso para el procesador. Hay dos tipos de colisionadores: primitivos en Unity 2D: **Box Collider 2D** y **Circle Collider 2D**.

Instrucciones

- Selecione **PlayerObject** y luego seleccione el botón **Add Component** en la vent
Inspector. Busque y seleccione **"Box Collider 2D"** para agregar un **Box Collider 2D** a **PlayerObject**



Bienvenido, Alan Manuel Mendoza Arredondo

Administra tu información y las opciones de privacidad y seguridad a fin de que Google sea más relevante para ti. [Más información](#)

Privacidad y personalización
Para personalizar tu experiencia en Google, elige la

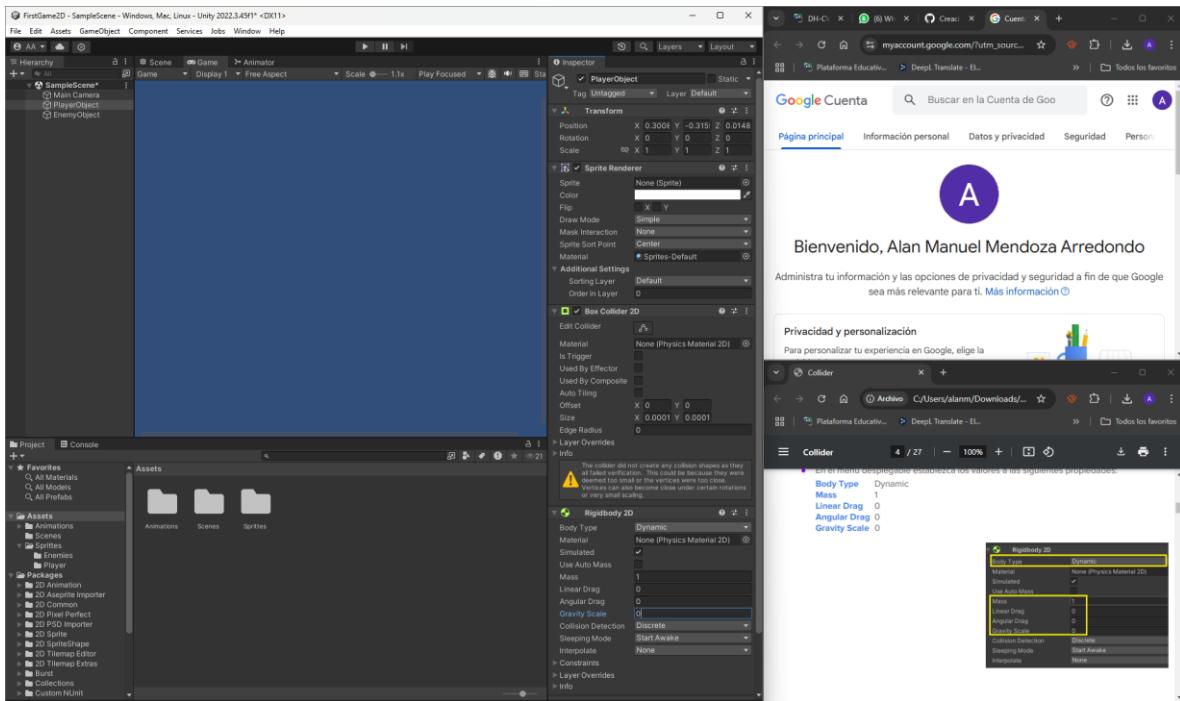
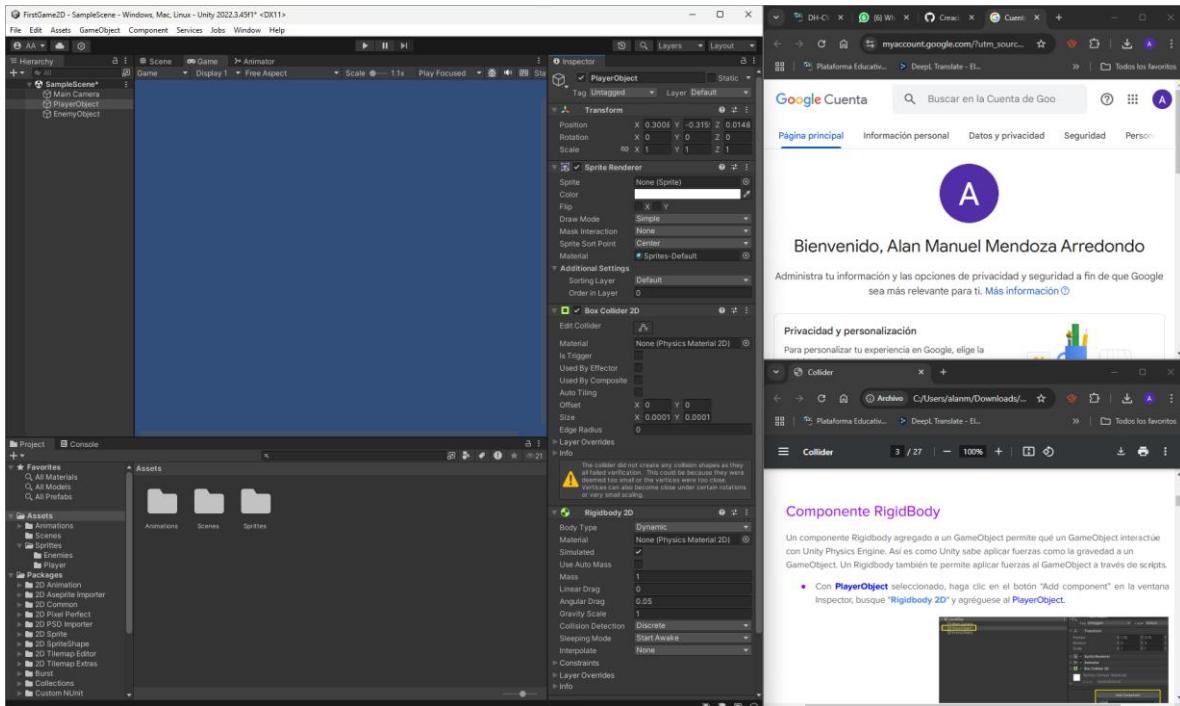
Collider

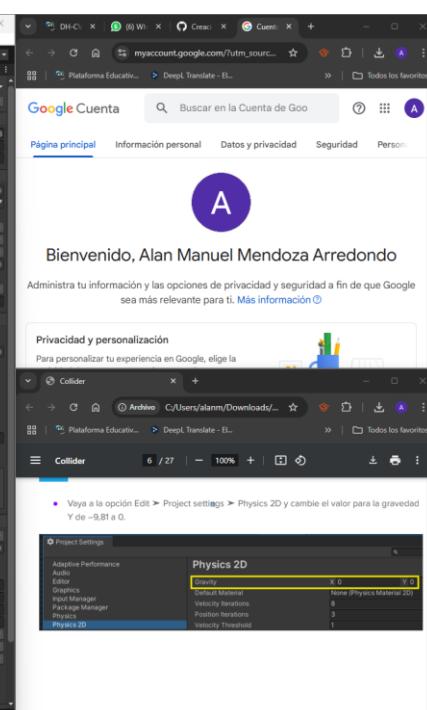
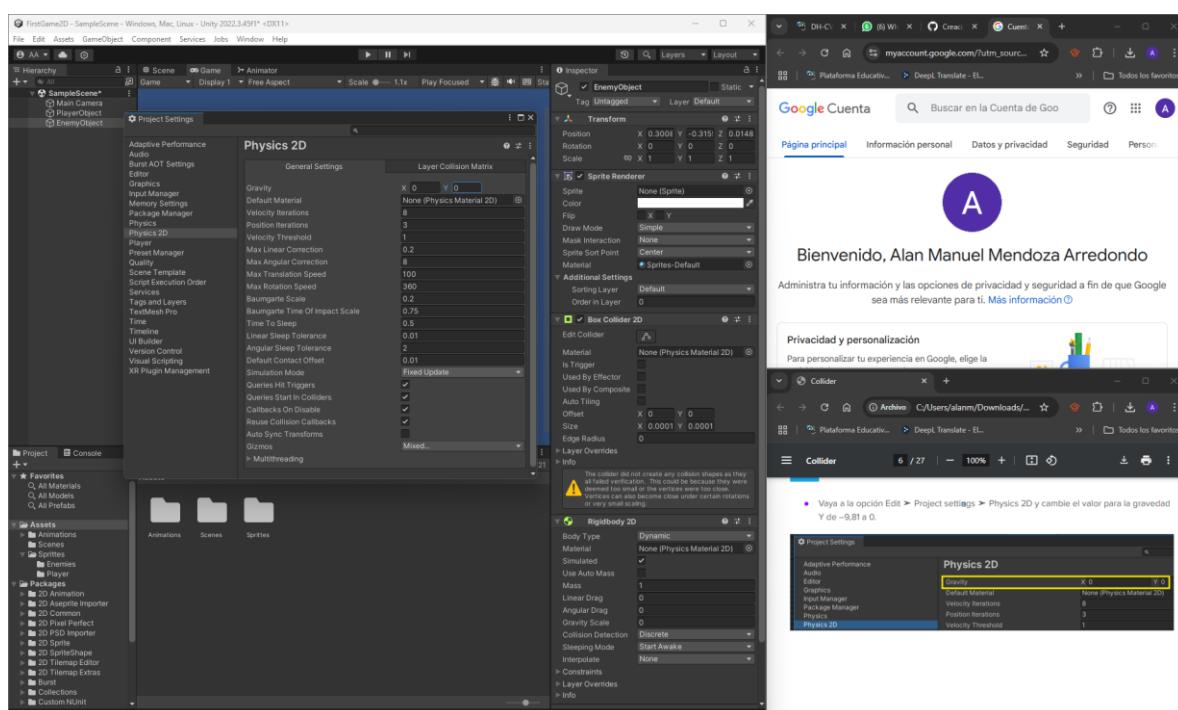
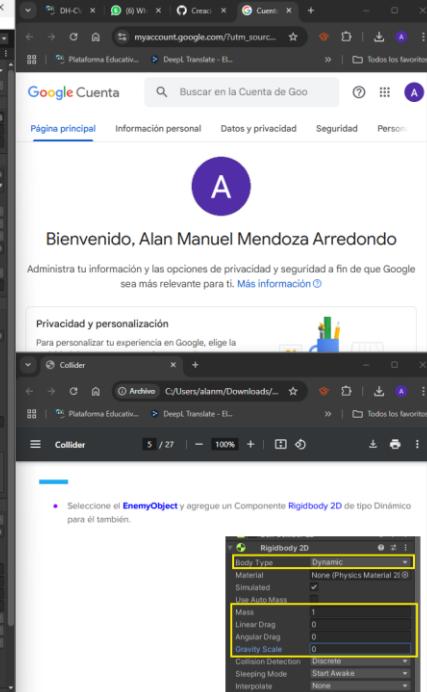
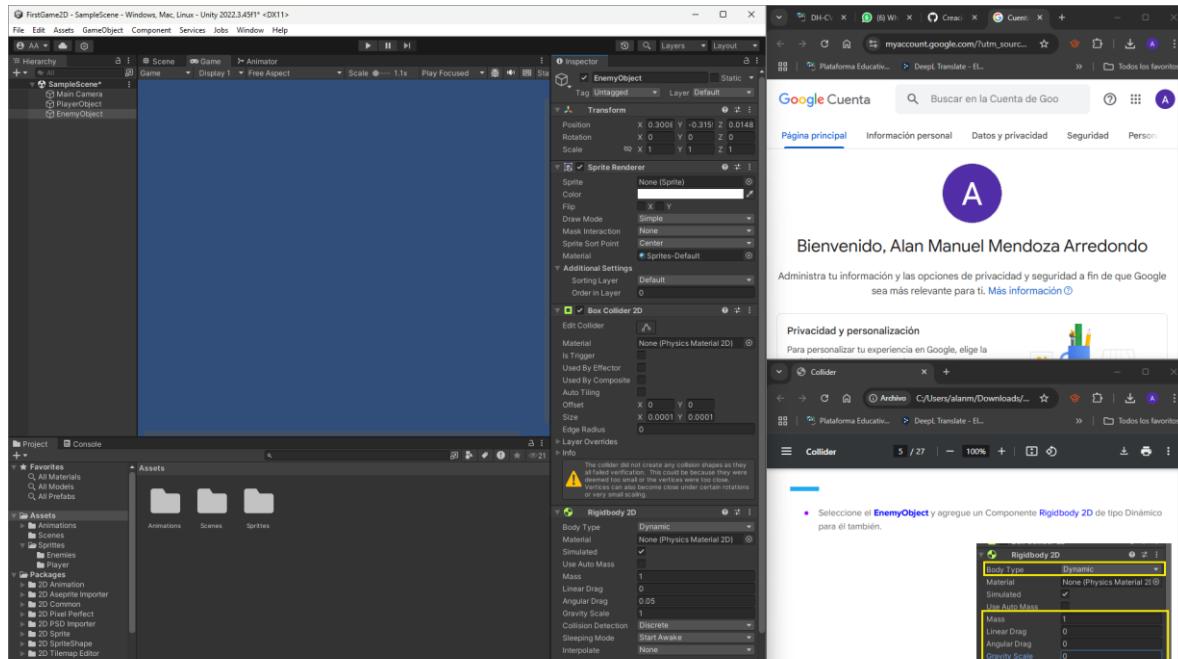
The collider did not create any collision shapes as they all failed verification. This could be because they were too small or the vertices were too close. Vertices can also become close under certain rotations or very small scaling.

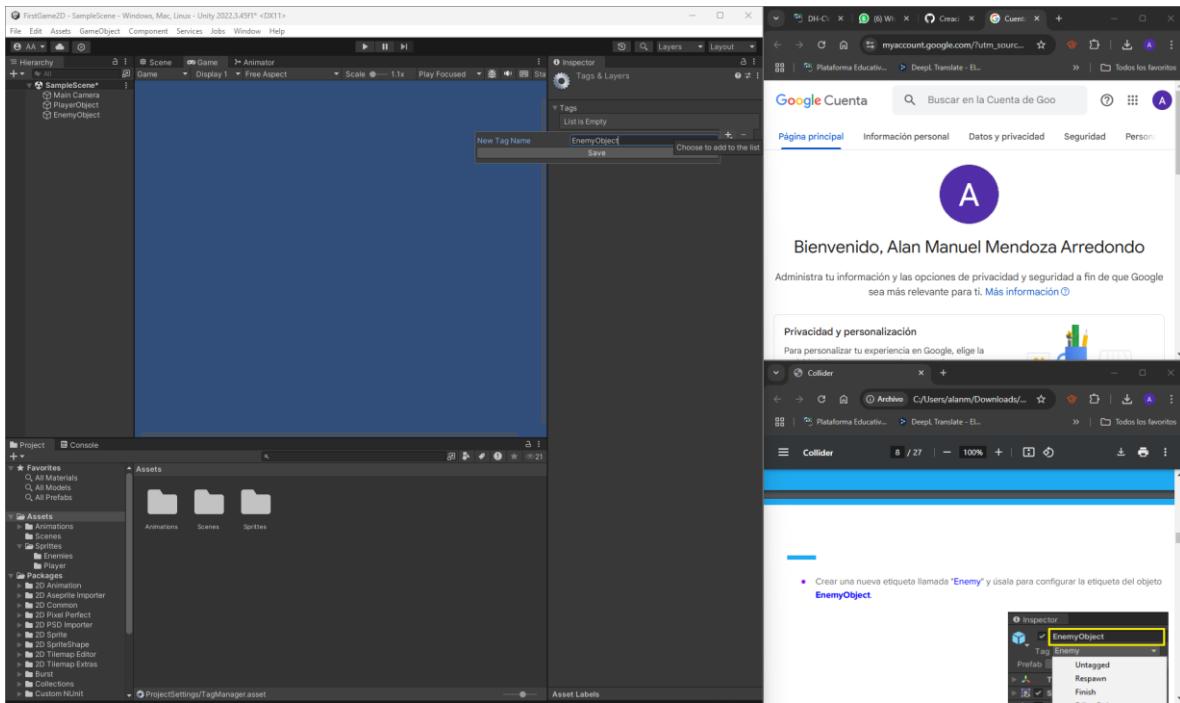
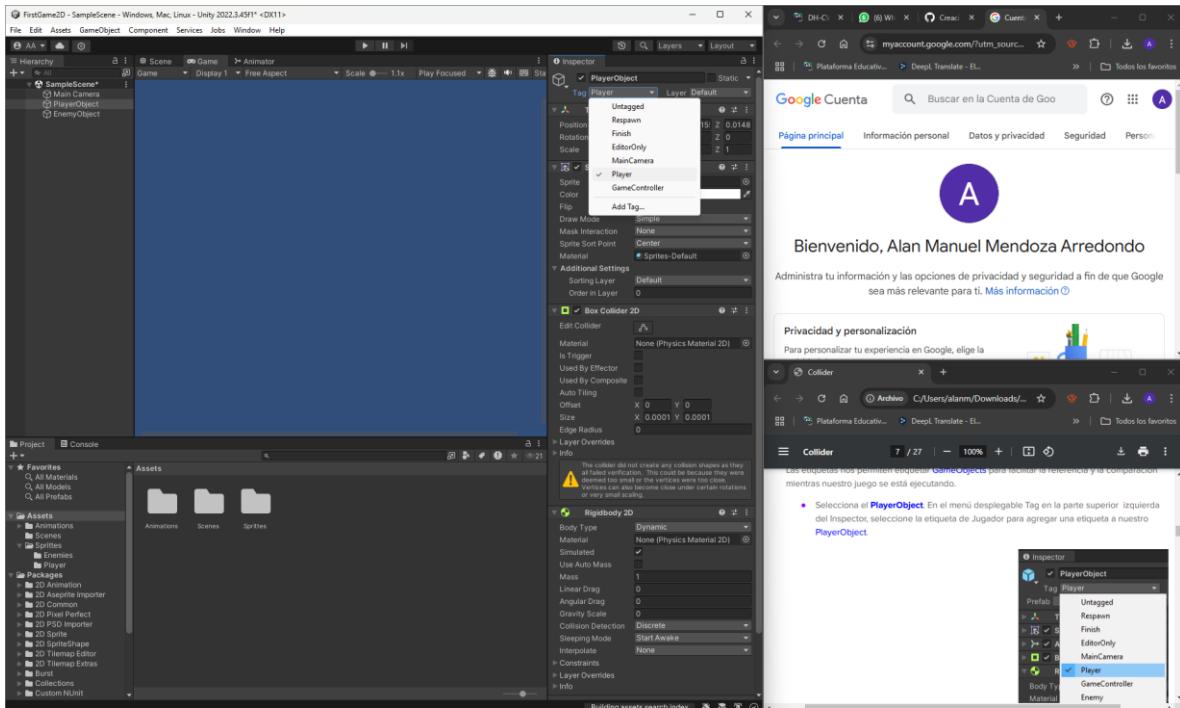
Necesitaremos saber cuándo el jugador choca con un enemigo, así que agregue un **Box Collider 2D** al **EnemyObject** también.

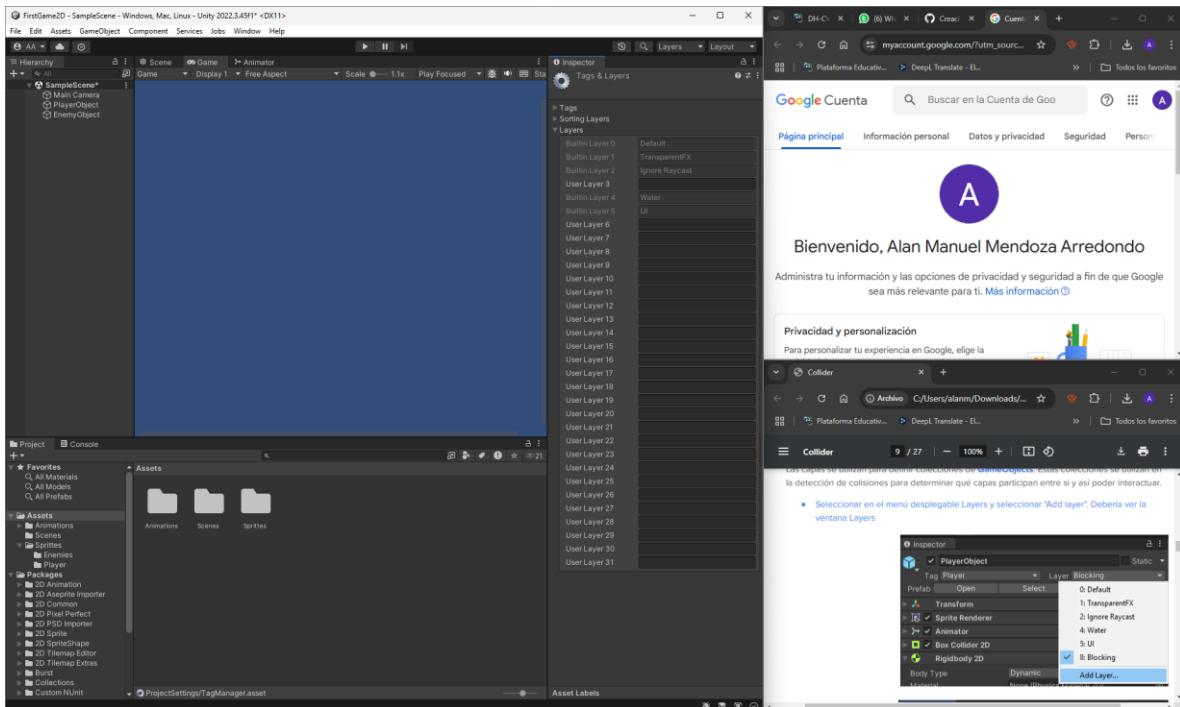
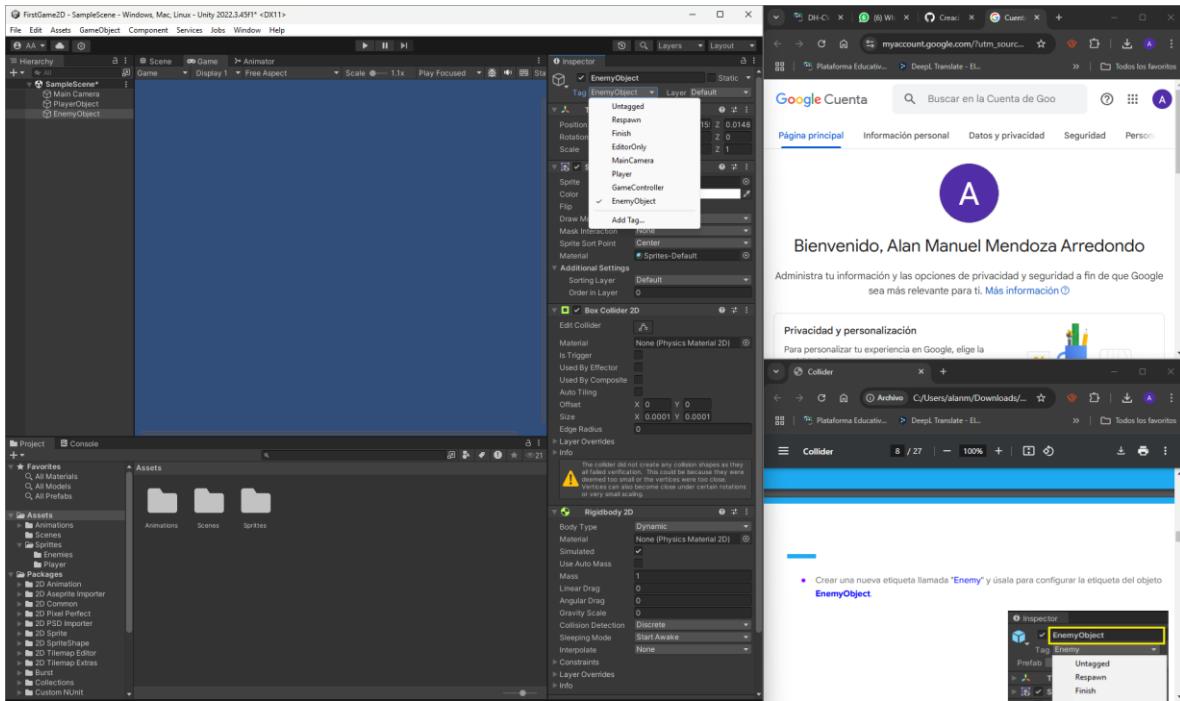
Componente RigidBody

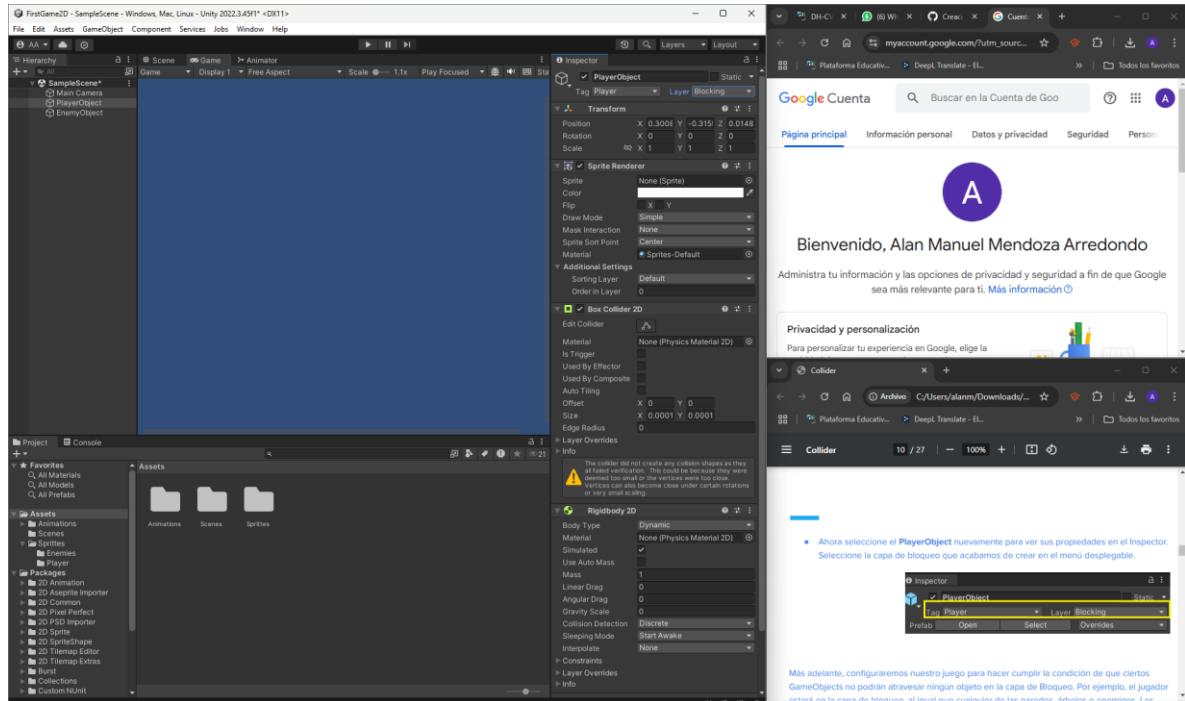
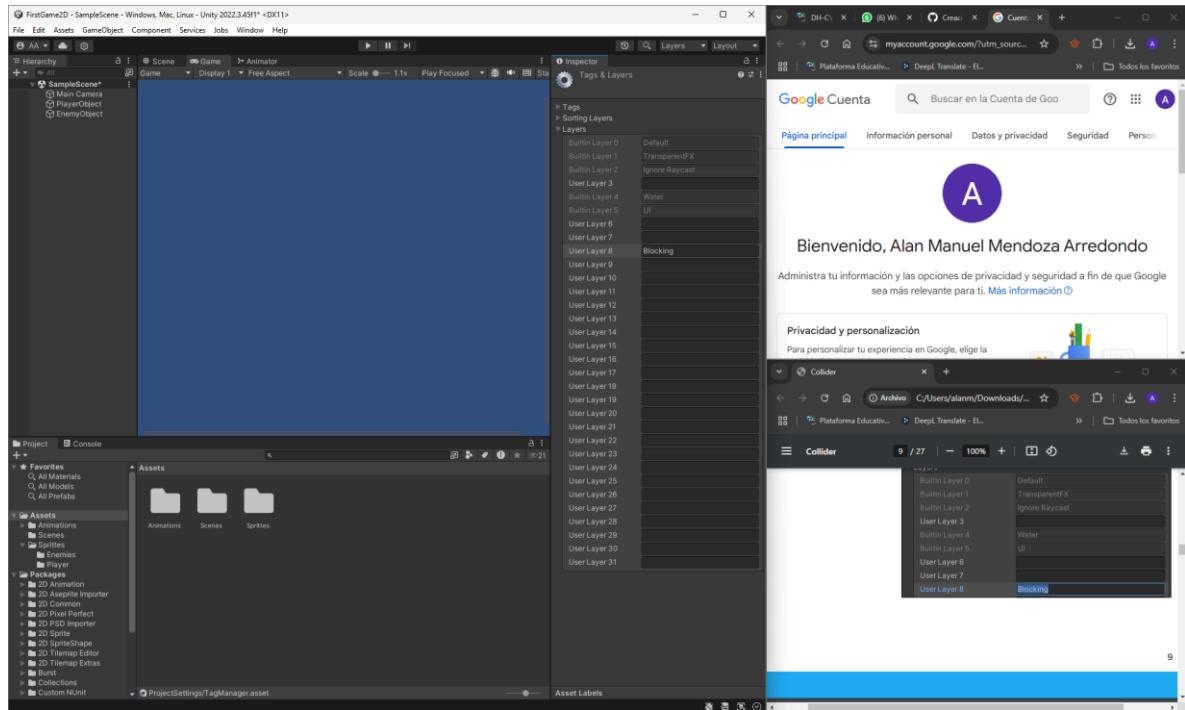
Un componente **Rigidbody** agregado a un **GameObject** permite que un **GameObject** interactúe con Unity Physics Engine. Así es como Unity sabe aplicar fuerzas como la gravedad a un **GameObject**. Un **Rigidbody** también te permite aplicar fuerzas a un **GameObject** a través de scripts.

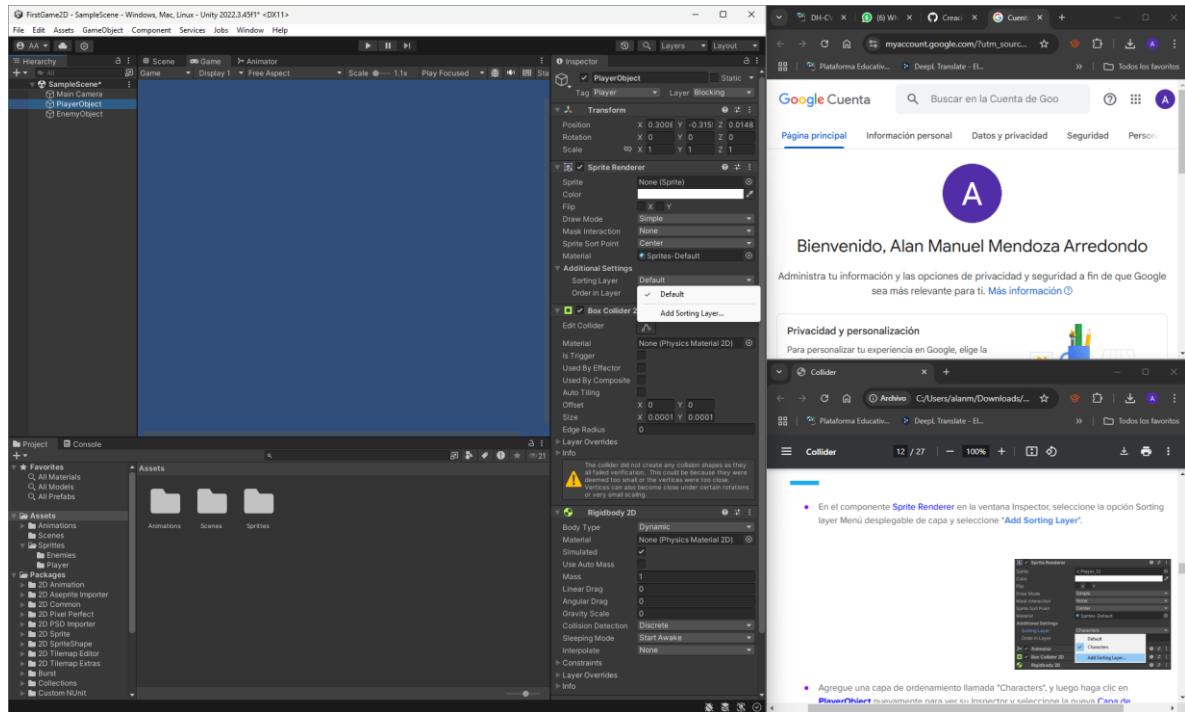












Bienvenido, Alan Manuel Mendoza Arredondo

Administrá tu información y las opciones de privacidad y seguridad a fin de que Google sea más relevante para ti. [Más información](#)

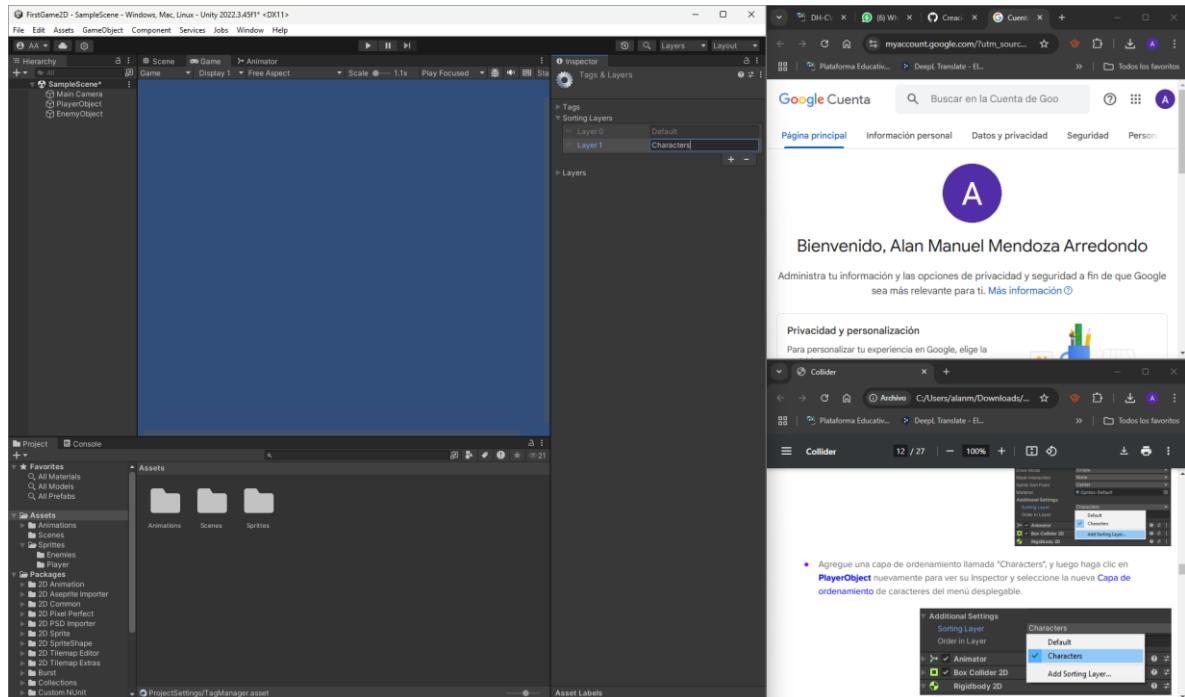
Privacidad y personalización

Para personalizar tu experiencia en Google, elige la

Collider

- En el componente **Sprite Renderer** en la ventana **Inspector**, seleccione la opción **Sorting layer** Menú desplegable de capa y seleccione **"Add Sorting Layer"**.

Agregue una capa de ordenamiento llamada "Characters", y luego haga clic en **PlayerObject** nuevamente para ver su Inspector y seleccione la nueva **Capa de ordenamiento** de caracteres del menú desplegable.



Bienvenido, Alan Manuel Mendoza Arredondo

Administrá tu información y las opciones de privacidad y seguridad a fin de que Google sea más relevante para ti. [Más información](#)

Privacidad y personalización

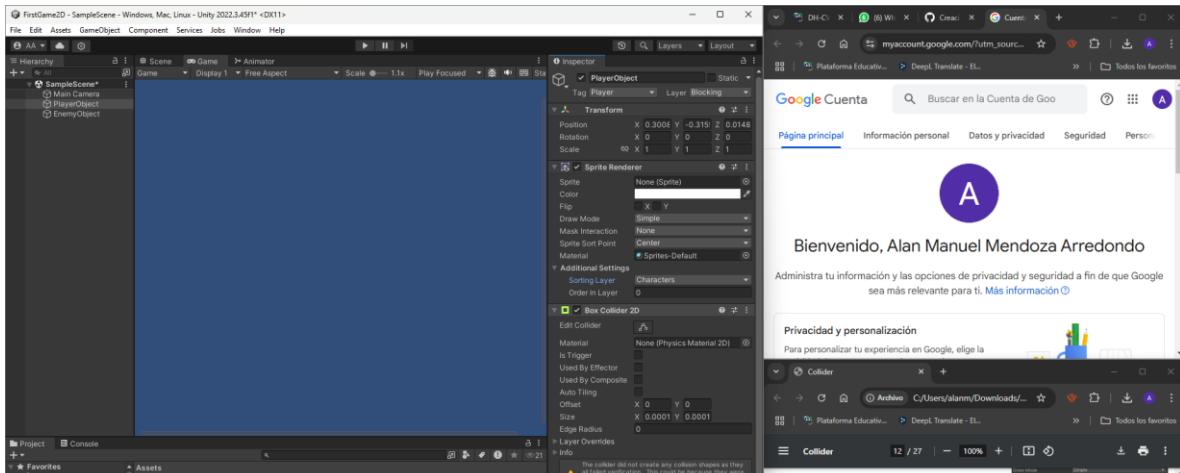
Para personalizar tu experiencia en Google, elige la

Collider

- Agregue una capa de ordenamiento llamada "Characters", y luego haga clic en **PlayerObject** nuevamente para ver su Inspector y seleccione la nueva **Capa de ordenamiento** de caracteres del menú desplegable.

Additional Settings

Sorting Layer	Order in Layer	Characters
Default	Animator	<input checked="" type="checkbox"/> Characters
	Box Collider 2D	Add Sorting Layer...
	Rigidbody 2D	



Bienvenido, Alan Manuel Mendoza Arredondo

Administrá tu información y las opciones de privacidad y seguridad a fin de que Google sea más relevante para ti. [Más información](#)

Privacidad y personalización

Para personalizar tu experiencia en Google, elige la

Collider

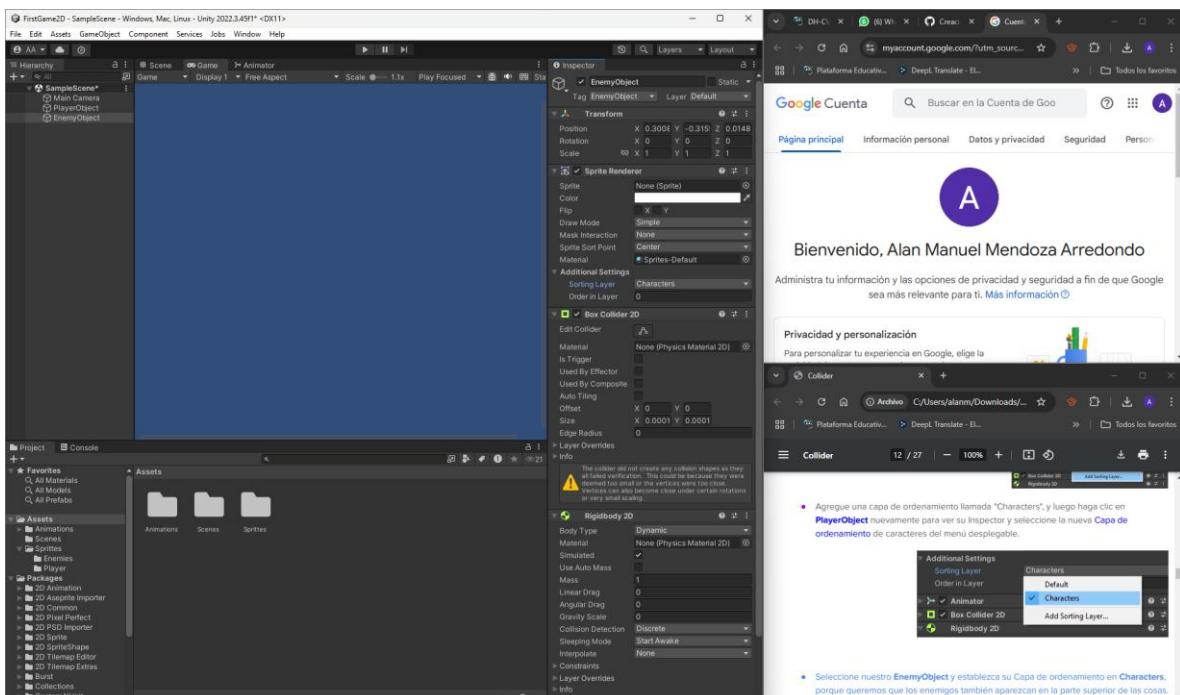
Agree una capa de ordenamiento llamada "Characters", y luego haga clic en **PlayerObject** nuevamente para ver su inspector y seleccione la nueva **Capa de ordenamiento** de caracteres del menú desplegable.

Additional Settings

Sorting Layer Characters

Order in Layer

Animator Box Collider 2D Rigidbody 2D



Bienvenido, Alan Manuel Mendoza Arredondo

Administrá tu información y las opciones de privacidad y seguridad a fin de que Google sea más relevante para ti. [Más información](#)

Privacidad y personalización

Para personalizar tu experiencia en Google, elige la

Collider

Agree una capa de ordenamiento llamada "Characters", y luego haga clic en **PlayerObject** nuevamente para ver su inspector y seleccione la nueva **Capa de ordenamiento** de caracteres del menú desplegable.

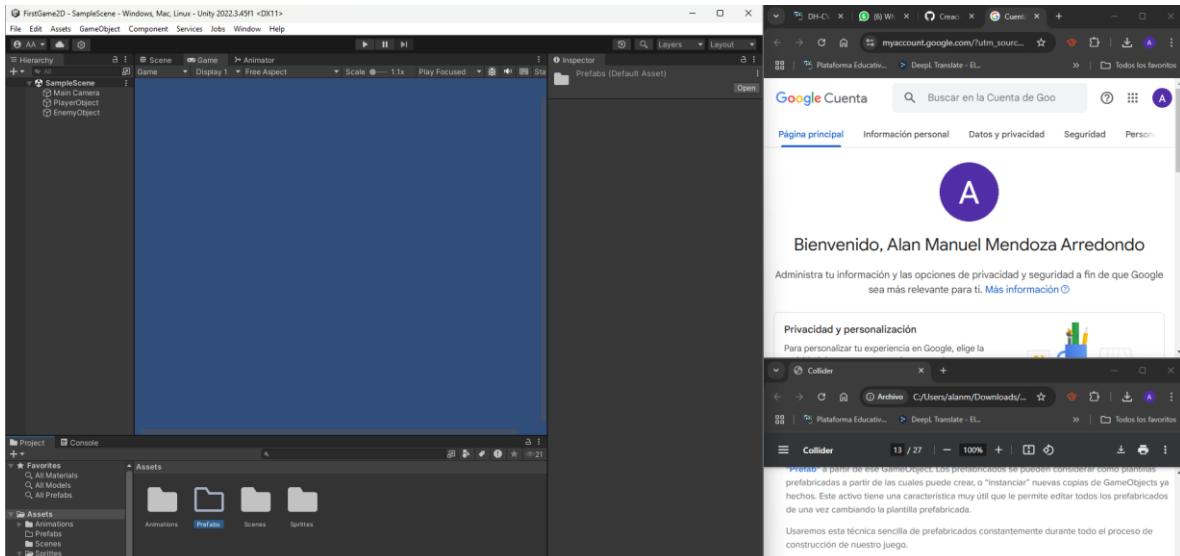
Additional Settings

Sorting Layer Characters

Order in Layer

Animator Box Collider 2D Rigidbody 2D

Selección nuestro **EnemyObject** y establezca su Capa de ordenamiento en **Characters**, porque queremos que los enemigos también aparezcan en la parte superior de las cosas.



Bienvenido, Alan Manuel Mendoza Arredondo

Administrá tu información y las opciones de privacidad y seguridad a fin de que Google sea más relevante para ti. [Más información](#)

Privacidad y personalización

Para personalizar tu experiencia en Google, elige la

Collider

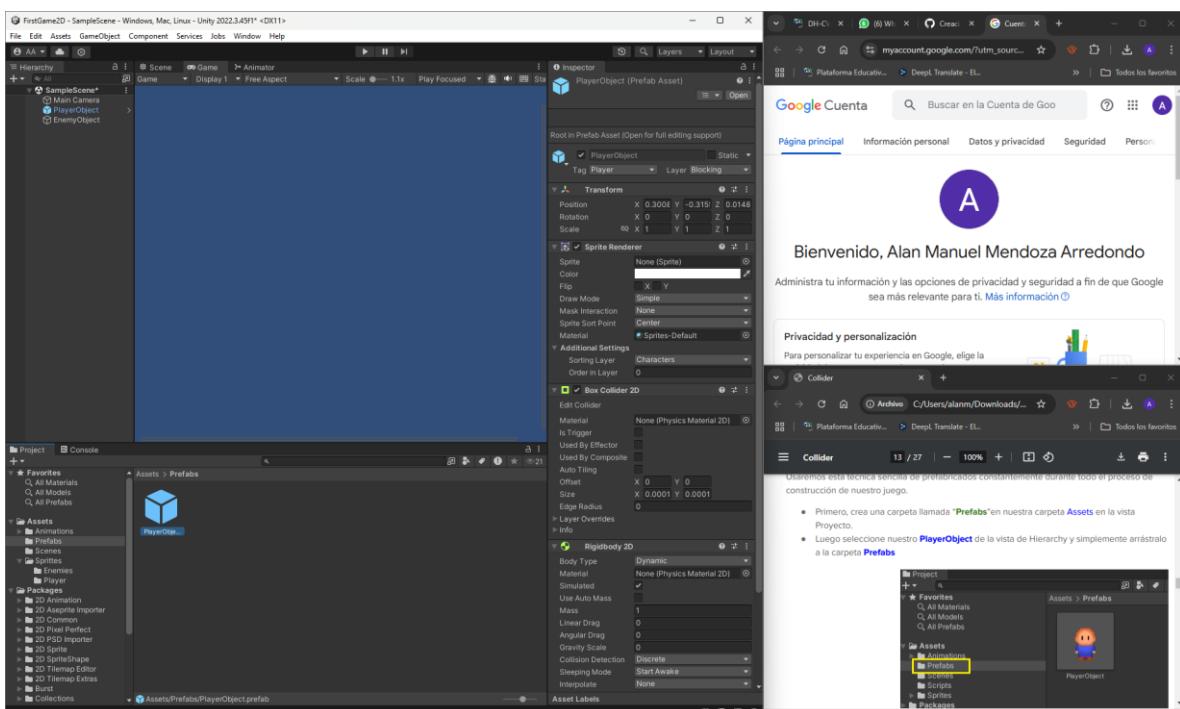
Puedes arrastrar y soltar este GameObject. Los prefabricados se pueden considerar como plantillas prefabricadas a partir de las cuales puedes crear, o "instanciar" nuevas copias de GameObjects ya hechos. Este activo tiene una característica muy útil que le permite editar todos los prefabricados de una vez cambiando la plantilla prefabricada.

Usaremos esta técnica sencilla de prefabricados constantemente durante todo el proceso de construcción de nuestro juego.

- Primero, crea una carpeta llamada **'Prefabs'** en nuestra carpeta **Assets** en la vista Proyecto.
- Luego selecciona nuestro **PlayerObject** de la vista de Hierarchy y simplemente arrástralo a la carpeta **Prefabs**.

Project

Assets > Favorites
Assets > All Materials
Assets > All Models
Assets > All Prefabs



Bienvenido, Alan Manuel Mendoza Arredondo

Administrá tu información y las opciones de privacidad y seguridad a fin de que Google sea más relevante para ti. [Más información](#)

Privacidad y personalización

Para personalizar tu experiencia en Google, elige la

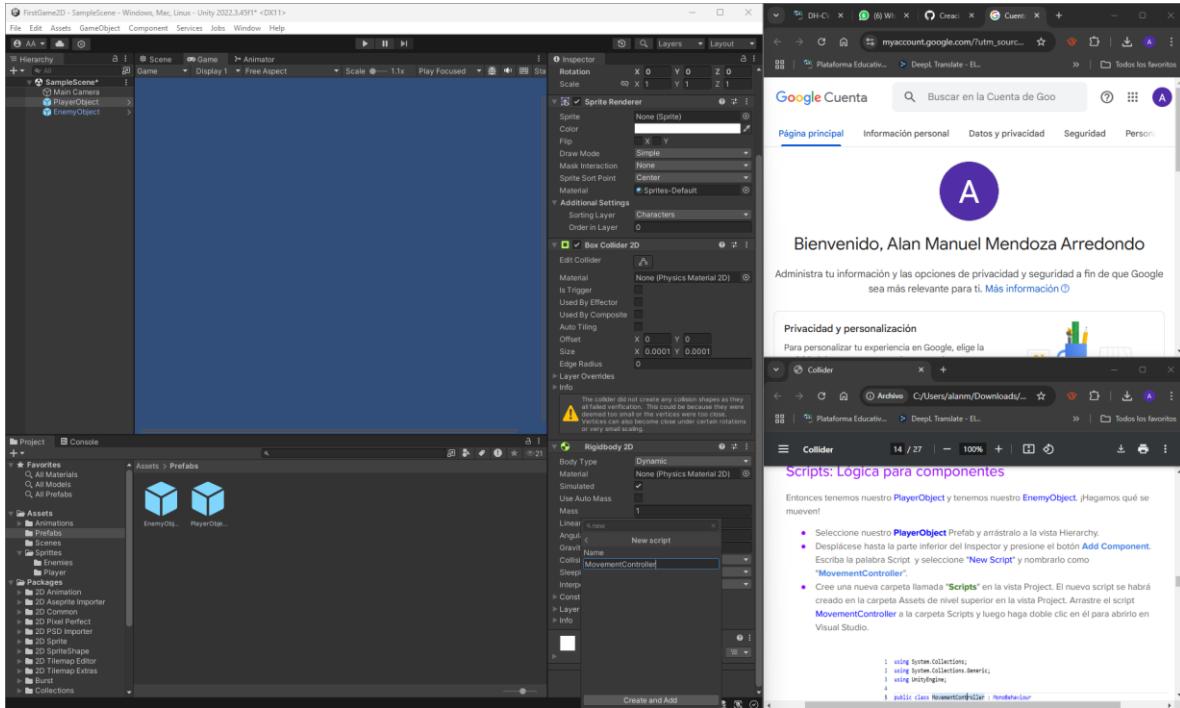
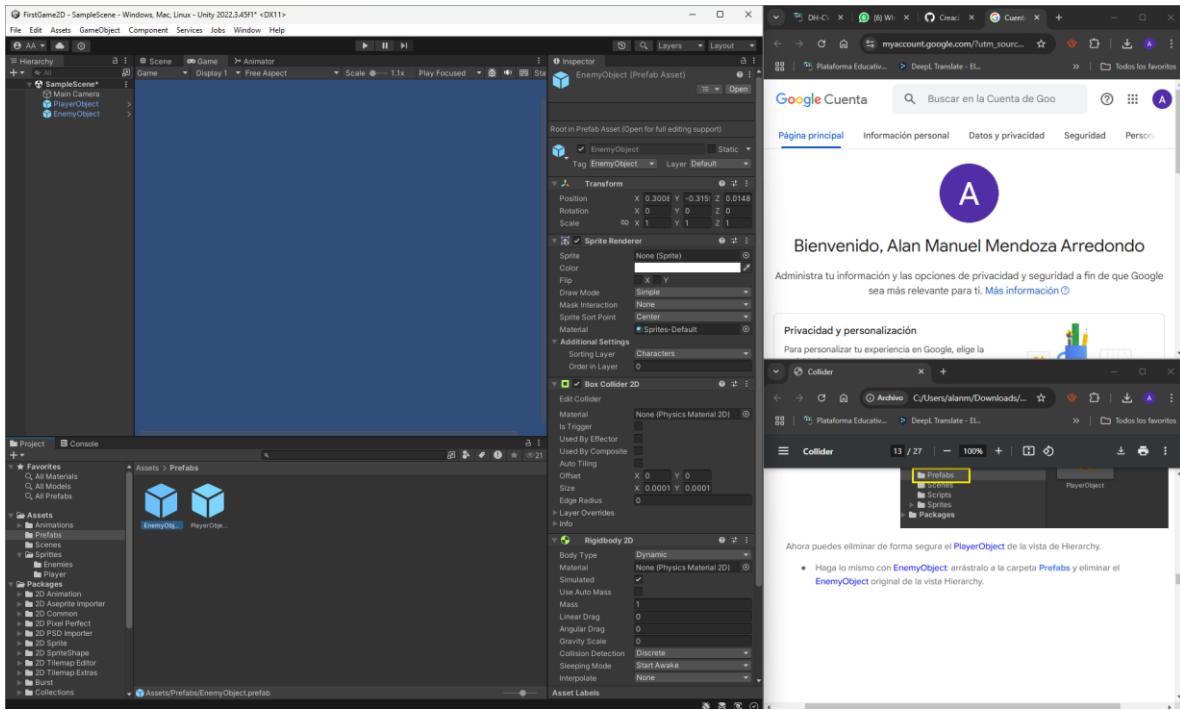
Collider

Usaremos esta técnica sencilla de prefabricados constantemente durante todo el proceso de construcción de nuestro juego.

- Primero, crea una carpeta llamada **'Prefabs'** en nuestra carpeta **Assets** en la vista Proyecto.
- Luego selecciona nuestro **PlayerObject** de la vista de Hierarchy y simplemente arrástralo a la carpeta **Prefabs**.

Project

Assets > Favorites
Assets > All Materials
Assets > All Models
Assets > All Prefabs



Bienvenido, Alan Manuel Mendoza Arredondo

Administrá tu información y las opciones de privacidad y seguridad a fin de que Google sea más relevante para ti. [Más información](#)

Privacidad y personalización

Para personalizar tu experiencia en Google, elige la

Collider

Assets / Collider / Scripts / PlayerObject

Ahora puedes eliminar de forma segura el PlayerObject de la vista de Hierarchy.

- Haga lo mismo con EnemyObject: arrástrelo a la carpeta **Prefs** y elimínelo del EnemyObject original de la vista Hierarchy.

Bienvenido, Alan Manuel Mendoza Arredondo

Administrá tu información y las opciones de privacidad y seguridad a fin de que Google sea más relevante para ti. [Más información](#)

Privacidad y personalización

Para personalizar tu experiencia en Google, elige la

Collider

Assets / Collider / Scripts / PlayerObject

Scripts: Lógica para componentes

Entonces tenemos nuestro PlayerObject y tenemos nuestro EnemyObject. ¡Hagamos qué se mueve!

- Selecciónnate nuestro PlayerObject Prefab y arrástreló a la vista Hierarchy.
- Desplázese hasta la parte inferior del Inspector y presione el botón **Add Component**. Escriba la palabra Script y seleccione **New Script** y nombrarlo como **"MovementController"**.
- Cree una nueva carpeta llamada **"Scripts"** en la vista Project. El nuevo script se habrá creado en la carpeta Assets de nivel superior en la vista Project. Arrástrale el script **MovementController** a la carpeta Scripts y luego haga doble clic en él para abrirlo en Visual Studio.

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class MovementController : MonoBehaviour
6 {

```

Unity Editor:

Google Cuenta:

Visual Studio Code:

```

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

public class MovementController : MonoBehaviour
{
    public float movementSpeed = 3.8f;
    Vector2 movement = new Vector2(0);
    Rigidbody2D rb2D;
    void Start()
    {
        rb2D = GetComponent<Rigidbody2D>();
    }
    // Update is called once per frame
    void Update()
    {
    }
}

```

Google Cuenta:

Visual Studio Code:

```

public class MovementController : MonoBehaviour
{
    //Velocidad de los personajes
    public float movementSpeed = 3.8f;
    //Referencia al Rigidbody2D del Player o Enemy
    Vector2 movement = new Vector2(0);
    //Referencia a Rigidbody2D
    Rigidbody2D rb2D;
    // Start is called before the first frame update
    void Start()
    {
        //Establece el componente Rigidbody2D enlazado
        rb2D = GetComponent<Rigidbody2D>();
    }
}

```

The screenshot shows the Unity Editor interface on the left and a Google Account page on the right.

Unity Editor (Left):

- MovementController.cs:**

```

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

public class MovementController : MonoBehaviour
{
    public float movementSpeed = 3.0f;
    Vector2 movement = new Vector2(0, 0);
    Rigidbody rb2D;

    void Start()
    {
        rb2D = GetComponent<Rigidbody>();
    }

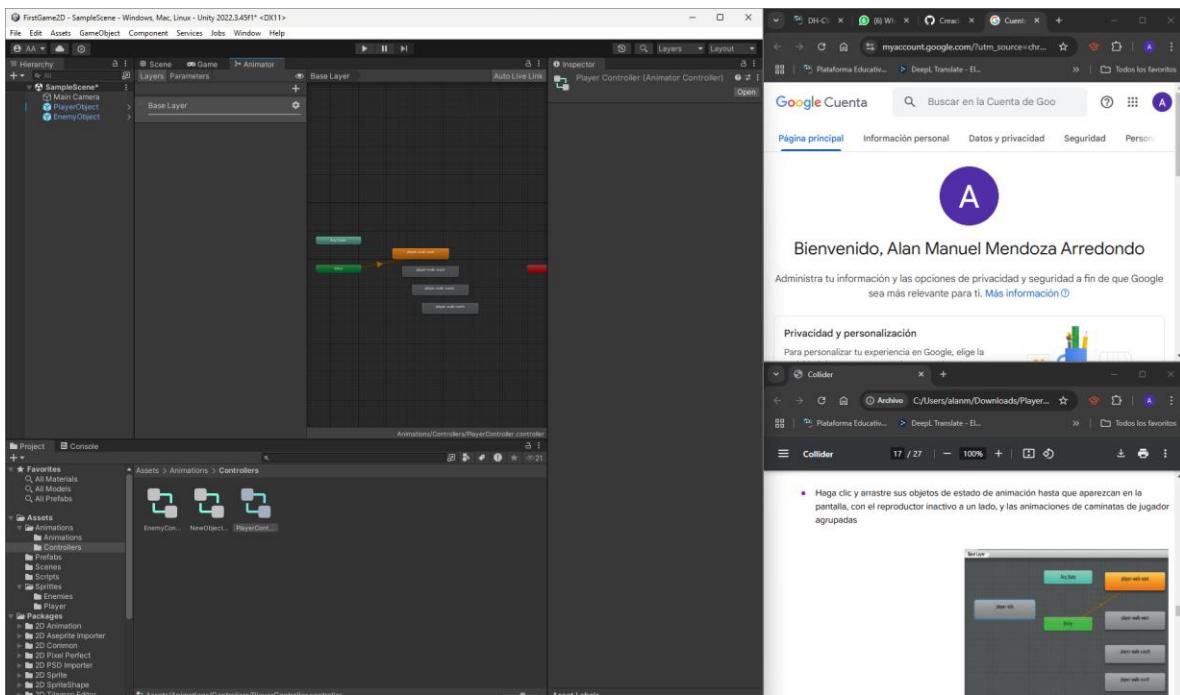
    // Update is called once per frame
    void Update()
    {
        movement.x = Input.GetAxisRaw("Horizontal");
        movement.y = Input.GetAxisRaw("Vertical");
        movement.Normalize();
        rb2D.velocity = movement * movementSpeed;
    }

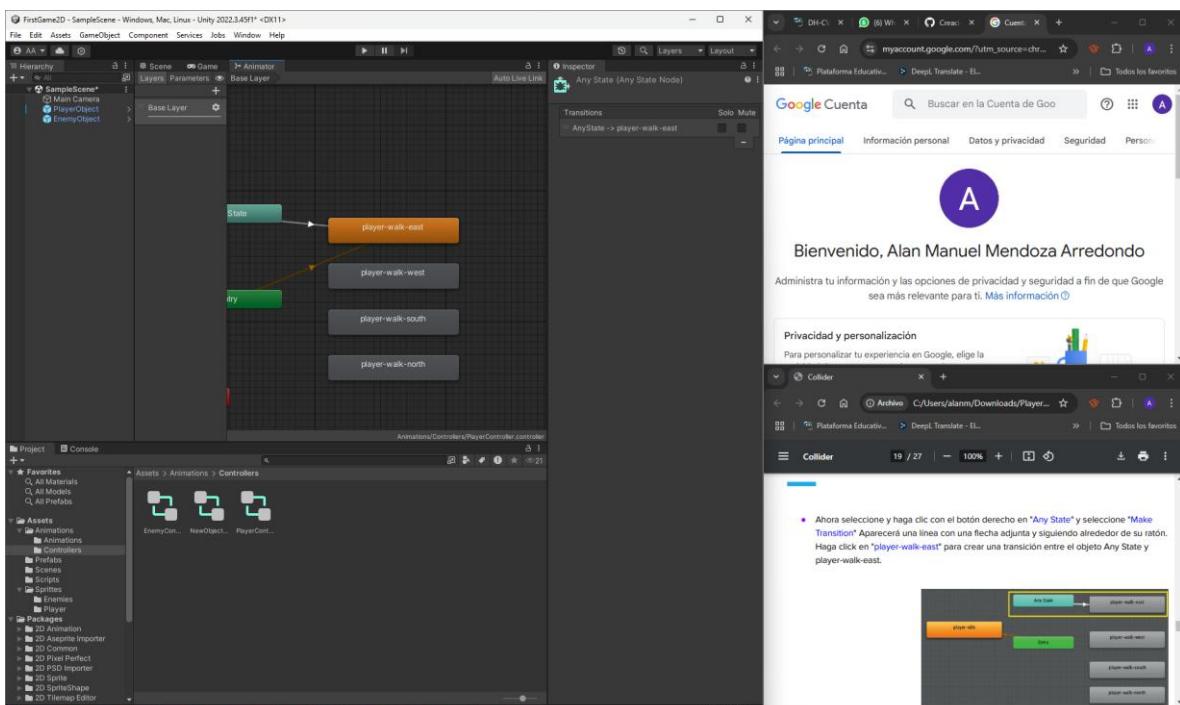
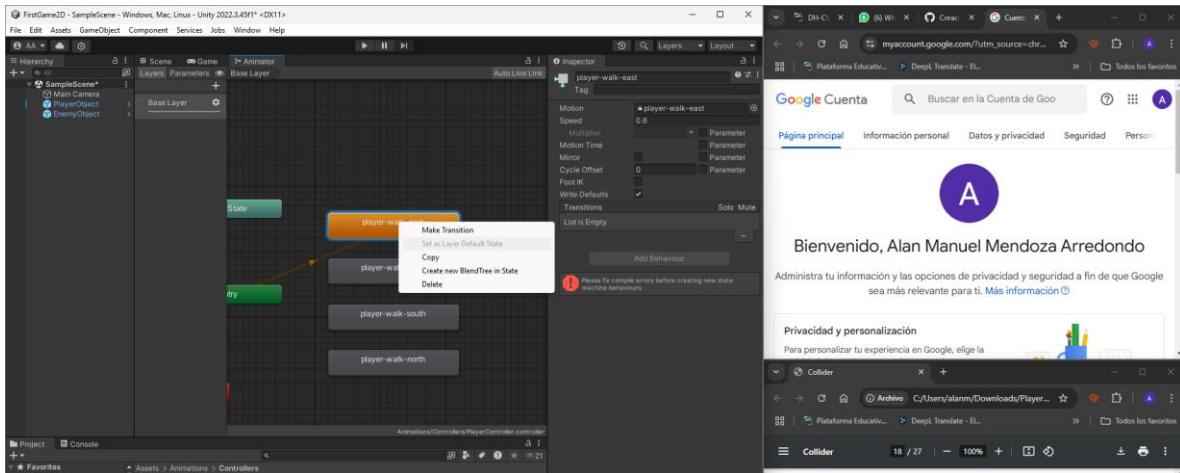
    // Update is called once per frame
    void FixedUpdate()
    {
        movement.x = Input.GetAxisRaw("Horizontal");
        movement.y = Input.GetAxisRaw("Vertical");
        movement.Normalize();
        rb2D.velocity = movement * movementSpeed;
    }
}

```
- Explorador de soluciones:** Shows the solution structure for "FirstGame2D".

Google Cuenta (Right):

- Página principal:** Bienvenido, Alan Manuel Mendoza Arredondo.
- Privacidad y personalización:** Para personalizar tu experiencia en Google, elige la configuración que más te convenga.





A

Bienvenido, Alan Manuel Mendoza Arredondo

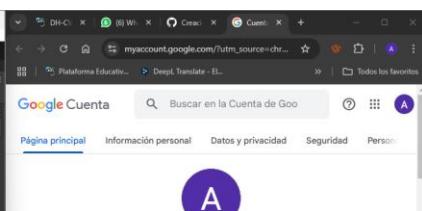
Administrá tu información y las opciones de privacidad y seguridad a fin de que Google sea más relevante para ti. [Más información](#)

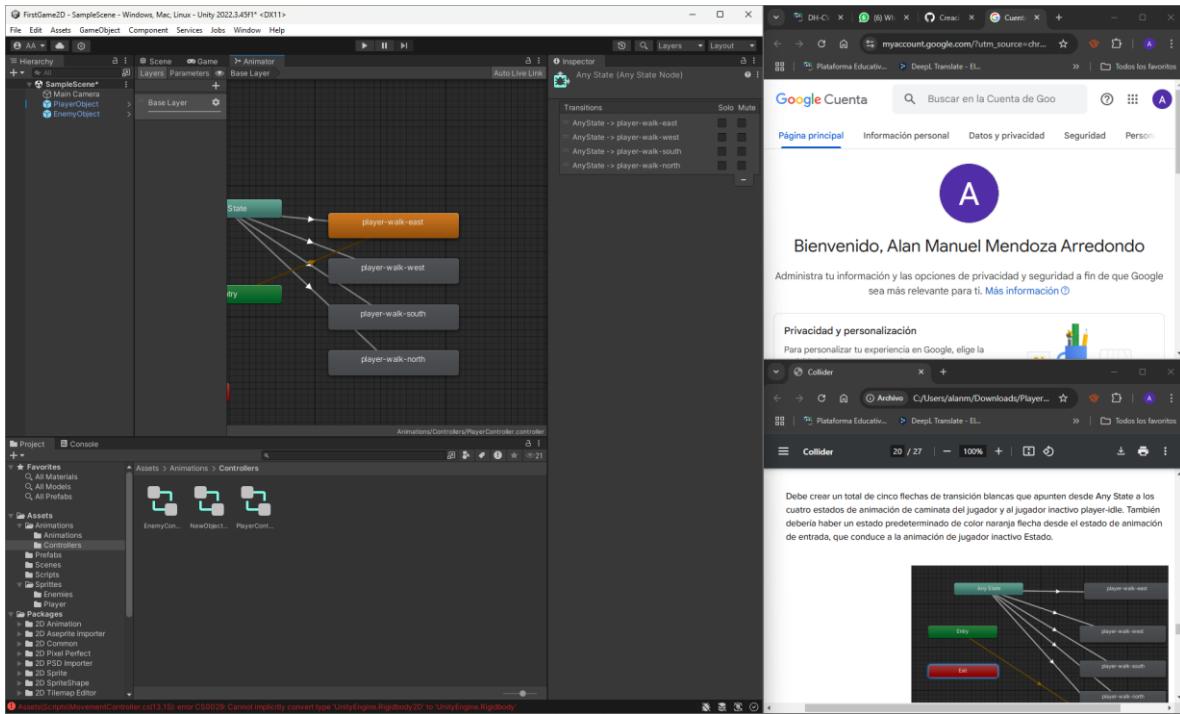
Privacidad y personalización

Para personalizar tu experiencia en Google, elige la

Collider

El color naranja indica que es el estado predeterminado para este Animator. Seleccione y, a continuación, haga clic con el botón derecho en el estado de animación "player-idle" y seleccione "Set as Layer Default State".





Bienvenido, Alan Manuel Mendoza Arredondo

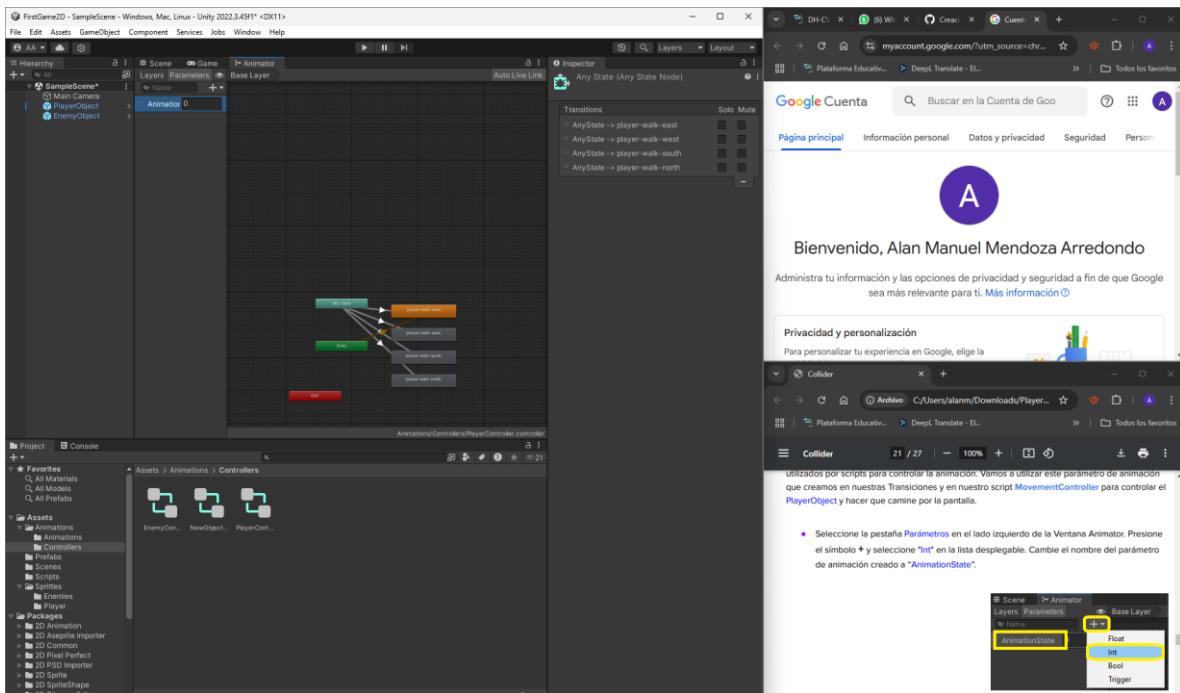
Administrá tu información y las opciones de privacidad y seguridad a fin de que Google sea más relevante para ti. [Más información](#)

Privacidad y personalización

Para personalizar tu experiencia en Google, elige la

Collider

Debe crear un total de cinco flechas de transición blancas que apunten desde Any State a los cuatro estados de animación de caminata del jugador y al jugador inactivo player-idle. También debería haber un estado predeterminado de color naranja flecha desde el estado de animación de entrada, que conduce a la animación de jugador inactivo Estado.



Bienvenido, Alan Manuel Mendoza Arredondo

Administrá tu información y las opciones de privacidad y seguridad a fin de que Google sea más relevante para ti. [Más información](#)

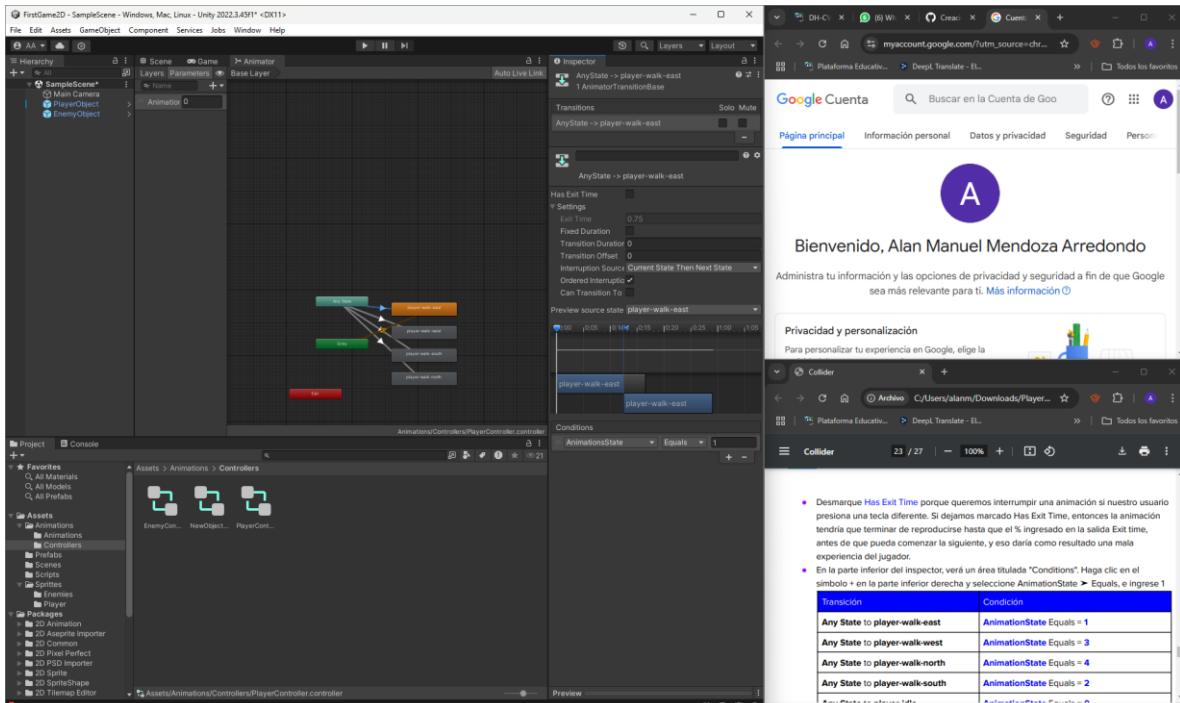
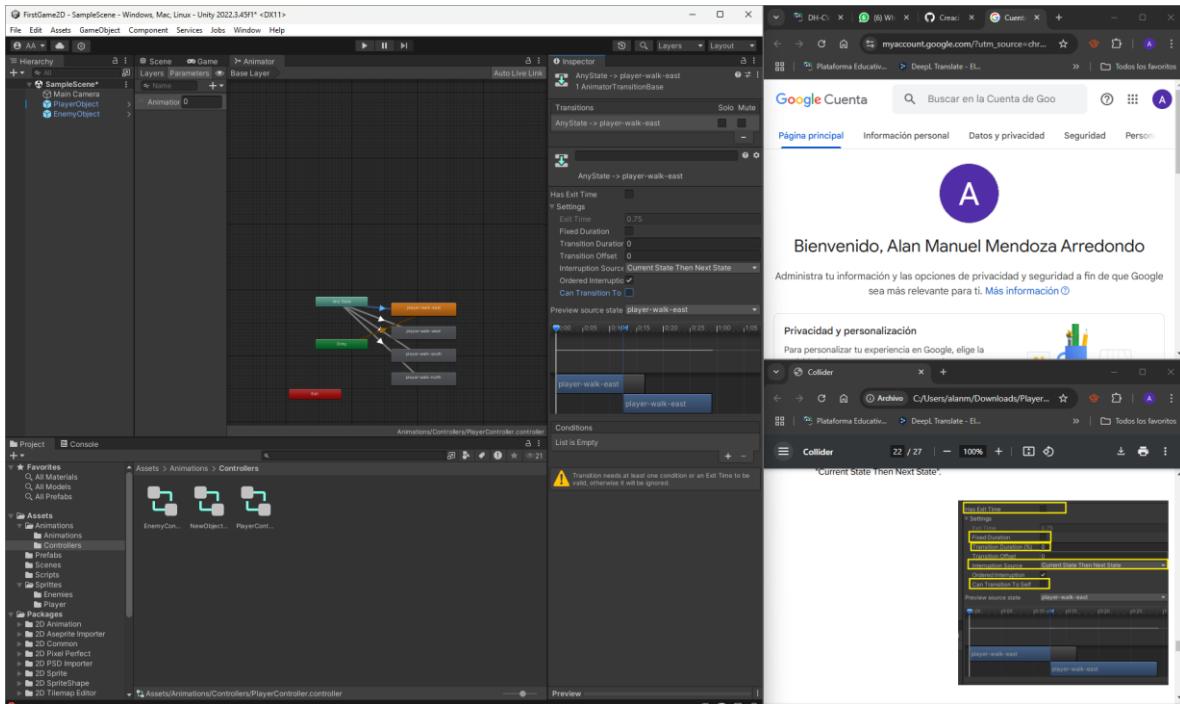
Privacidad y personalización

Para personalizar tu experiencia en Google, elige la

Collider

Utilizados por scripts para controlar la animación. Vamos a utilizar este parámetro de animación que creamos en nuestras Transiciones y en nuestro script *MovementController* para controlar el *PlayerObject* y hacer que camine por la pantalla.

- Selecione la pestaña **Parámetros** en el lado izquierdo de la Ventana Animator. Presione el símbolo '+' y seleccione 'int' en la lista desplegable. Cambie el nombre del parámetro de animación creado a 'AnimationState'.



Two screenshots of a Unity development environment showing the code for the `MovementController.cs` script and a corresponding Google account page.

Screenshot 1:

```

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

public class MovementController : MonoBehaviour
{
    public float movementSpeed = 3.0f;
    Vector2 movement = new Vector2(0, 0);
    Rigidbody2D rb2D;
    Animator animator;
    string animationState = "AnimationState";
}

enum CharStates
{
    walkFast = 1,
    walkWest = 3,
    walkSouth = 2,
    walkNorth = 4,
    idleSouth = 5
}

// Start is called before the first frame update
void Start()
{
    rb2D = GetComponent();
}

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

// Message de Unity 0 referencias
private void FixedUpdate()
{
    movement.x = Input.GetAxisRaw("Horizontal");
    movement.y = Input.GetAxisRaw("Vertical");

    movement.Normalize();
    rb2D.velocity = movement * movementSpeed;
}

```

Screenshot 2:

```

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

public class MovementController : MonoBehaviour
{
    public float movementSpeed = 3.0f;
    Vector2 movement = new Vector2(0, 0);
    Rigidbody2D rb2D;
    Animator animator;
    string animationState = "AnimationState";
}

enum CharStates
{
    walkFast = 1,
    walkWest = 3,
    walkSouth = 2,
    walkNorth = 4,
    idleSouth = 5
}

// Start is called before the first frame update
void Start()
{
    rb2D = GetComponent();
    animator = GetComponent();
}

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

// Message de Unity 0 referencias
private void FixedUpdate()
{
    movement.x = Input.GetAxisRaw("Horizontal");
    movement.y = Input.GetAxisRaw("Vertical");

    movement.Normalize();
    rb2D.velocity = movement * movementSpeed;
}

```

The second screenshot shows the `animator = GetComponent();` line highlighted with a red box, indicating the step where the `Animator` component is assigned to the variable.

Google Account Page:

Bienvenido, Alan Manuel Mendoza Arredondo

Administrá tu información y las opciones de privacidad y seguridad a fin de que Google sea más relevante para ti. [Más información](#)

Privacidad y personalización

Para personalizar tu experiencia en Google, elige la

Collider

// Start is called before the first frame update
void Start()
{
 //Establece el componente Rigidbody2D enlazado
 rb2D = GetComponent //Establece valor de componente Animator el objeto ligado
 animator = GetComponent<Animator>();
}

● Inicializamos en el método start el valor del componente Animator del objeto enlazado

// Start is called before the first frame update
void Start()
{
 //Establece el componente Rigidbody2D enlazado
 rb2D = GetComponent< //Establece valor de componente Animator el objeto ligado
 animator = GetComponent<Animator>();
}

The screenshot shows the Unity Editor's code editor with the file `MovementController.cs` open. The code implements a movement controller using Unity's Animation system. It defines an enum `CharStates` with values `walkEast`, `walkWest`, `walkSouth`, `walkNorth`, and `idleSouth`. The `Start()` method initializes the animator component. The `Update()` method is called once per frame to update the animation state based on movement direction. The `FixedUpdate()` method is called once per frame to handle input and move the character. A browser window in the background shows a Google account profile for Alan Manuel Mendoza Arredondo.

```

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

public class MovementController : MonoBehaviour
{
    public enum CharStates
    {
        walkEast = 1,
        walkWest = 3,
        walkSouth = 2,
        walkNorth = 4,
        idleSouth = 5
    }

    // Start is called before the first frame update
    void Start()
    {
        rb2D = GetComponent<Rigidbody2D>();
        animator = GetComponent<Animator>();
    }

    // Update is called once per frame
    void Update()
    {
        // References
        private void UpdateState()
        {
            if (movement.x > 0)
            {
                animator.SetInteger(animationState, (int)CharStates.walkEast);
            }
            else if (movement.x < 0)
            {
                animator.SetInteger(animationState, (int)CharStates.walkWest);
            }
            else if (movement.y > 0)
            {
                animator.SetInteger(animationState, (int)CharStates.walkNorth);
            }
            else if (movement.y < 0)
            {
                animator.SetInteger(animationState, (int)CharStates.walkSouth);
            }
            else
            {
                animator.SetInteger(animationState, (int)CharStates.idleSouth);
            }
        }

        // References
        private void FixedUpdate()
        {
            movement.x = Input.GetAxisRaw("Horizontal");
            movement.y = Input.GetAxisRaw("Vertical");

            movement.Normalize();
            rb2D.velocity = movement * movementSpeed;
        }
    }

    // Update is called once per frame
    void FixedUpdate()
    {
        // References
        void Update()
        {
            // References
            private void UpdateState()
            {
                if (movement.x > 0)
                {
                    animator.SetInteger(animationState, (int)CharStates.walkEast);
                }
                else if (movement.x < 0)
                {
                    animator.SetInteger(animationState, (int)CharStates.walkWest);
                }
                else if (movement.y > 0)
                {
                    animator.SetInteger(animationState, (int)CharStates.walkNorth);
                }
                else if (movement.y < 0)
                {
                    animator.SetInteger(animationState, (int)CharStates.walkSouth);
                }
                else
                {
                    animator.SetInteger(animationState, (int)CharStates.idleSouth);
                }
            }

            // References
            private void FixedUpdate()
            {
                MoveCharacter();
            }
        }
    }

    // References
    private void MoveCharacter()
    {
        movement.x = Input.GetAxisRaw("Horizontal");
        movement.y = Input.GetAxisRaw("Vertical");

        movement.Normalize();
        rb2D.velocity = movement * movementSpeed;
    }
}

```

This screenshot shows the same `MovementController.cs` file with some changes. The `Update()` method has been removed, and the logic has been moved into the `FixedUpdate()` method. The `MoveCharacter()` method now includes the normalization and speed scaling logic. The browser window in the background remains the same.

```

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

public class MovementController : MonoBehaviour
{
    public enum CharStates
    {
        walkEast = 1,
        walkWest = 3,
        walkSouth = 2,
        walkNorth = 4,
        idleSouth = 5
    }

    // Start is called before the first frame update
    void Start()
    {
        rb2D = GetComponent<Rigidbody2D>();
        animator = GetComponent<Animator>();
    }

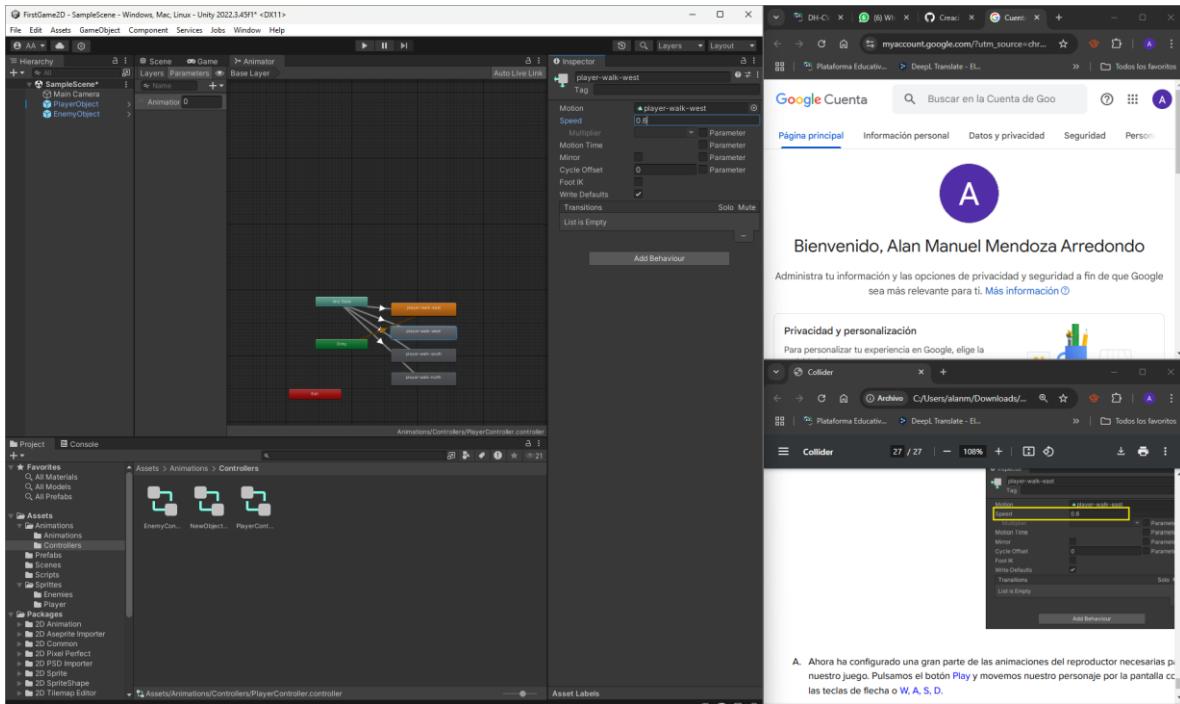
    // Update is called once per frame
    void FixedUpdate()
    {
        // References
        void Update()
        {
            // References
            private void UpdateState()
            {
                if (movement.x > 0)
                {
                    animator.SetInteger(animationState, (int)CharStates.walkEast);
                }
                else if (movement.x < 0)
                {
                    animator.SetInteger(animationState, (int)CharStates.walkWest);
                }
                else if (movement.y > 0)
                {
                    animator.SetInteger(animationState, (int)CharStates.walkNorth);
                }
                else if (movement.y < 0)
                {
                    animator.SetInteger(animationState, (int)CharStates.walkSouth);
                }
                else
                {
                    animator.SetInteger(animationState, (int)CharStates.idleSouth);
                }
            }

            // References
            private void FixedUpdate()
            {
                MoveCharacter();
            }
        }
    }

    // References
    private void MoveCharacter()
    {
        movement.x = Input.GetAxisRaw("Horizontal");
        movement.y = Input.GetAxisRaw("Vertical");

        movement.Normalize();
        rb2D.velocity = movement * movementSpeed;
    }
}

```



A. Ahora ha configurado una gran parte de las animaciones del reproductor necesarias para nuestro juego. Pulsamos el botón Play y movemos nuestro personaje por la pantalla con las teclas de flecha o W, A, S, D.