Códigos

Este repositorio fue creado con el objetivo de hacer un sistema de AI básico.

A continuación podrás leer los códigos usados para que el juego pueda funcionar.

Enemigo AI:

1. Se creo un script con el nombre “EnemigoAI”
2. Se crearon las referencias para poder programarlo.

```
     // Para referenciar al jugador.
    public Transform jugador;
    // Para la navegación del enemigo.
    private NavMeshAgent agente;
    // La distancia mínima para detectar al jugador.
    private float distanciaMinima = 20.0f;
```

3. Le decimos que es lo que debe hacer una vez comenzado el juego.

```
private void Start()
    {
        agente = GetComponent<NavMeshAgent>();

// Se llama a la función para que el enemigo tenga un nuevo destino inicial
        ObtenerNuevoDestino();
    }
```

4. Ahora le decimos que distancia hay entre el jugador y el enemigo

```
private void Update()
    {
      // Se calcula la distancia entre el enemigo y el jugador.
        float distanciaJugador = Vector3.Distance(transform.position, jugador.position);

     // Aqui se ve si la distancia entre el enemigo y el jugador es menor que
     // La distancia mínima.
        if (distanciaJugador < distanciaMinima)
        {
            RaycastHit hit;   //Se agrego un raycast para detectar al jugador.
```

```
    // Aqui se realiza un raycast desde la posición del enemigo en dirección al jugador.
            if (Physics.Raycast(transform.position, jugador.position -
                transform.position, out hit, distanciaMinima))
            {
    // Si el raycast detecta al jugador con la etiqueta "Player"
                if (hit.collider.tag == "Player")
                {
    // Se actualiza la posición del objetivo para perseguir al jugador
                    agente.destination = jugador.position;
                }
            }
        }
        else
        {
     // Si el enemigo no tiene un camino el cual seguir.

            if (!agente.hasPath || agente.remainingDistance < 0.5f)
     // Se llama a la función para obtener un nuevo destino aleatorio.
                ObtenerNuevoDestino();
            }
        }
    }
```

5. Se creo la función ObtenerNuevoDestino para darle una dirección al enemigo.

```
private void ObtenerNuevoDestino()
    {
// Se obtiene una posición aleatoria en un rango de 20 unidades.
        Vector3 destino = Random.insideUnitSphere * 20.0f + transform.position;
// Se coloco para almacenar la información del NavMesh.
        NavMeshHit hit;

// Se obtiene una posición en el NavMesh que esté cerca del objetivo aleatorio.
        NavMesh.SamplePosition(destino, out hit, 20.0f, NavMesh.AllAreas);
// Se establece la nueva posición.
        agente.destination = hit.position;
}   }
```

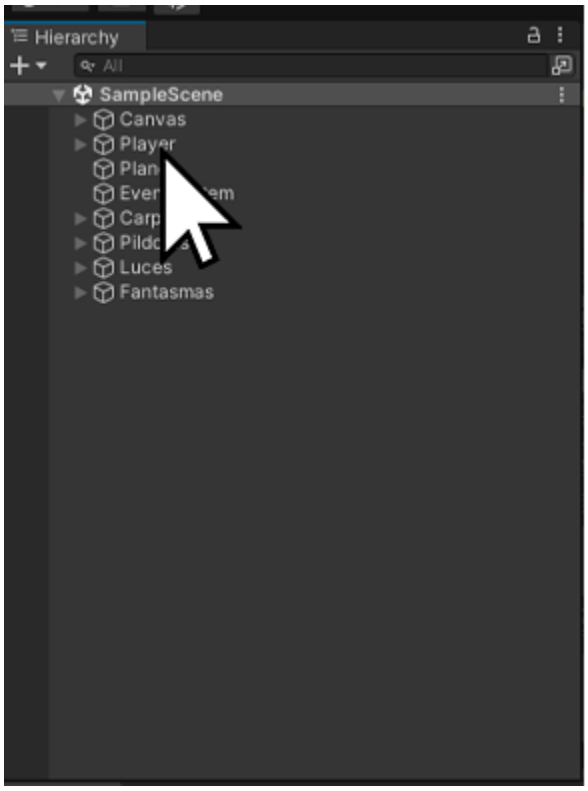
6. Así se ve el código completo

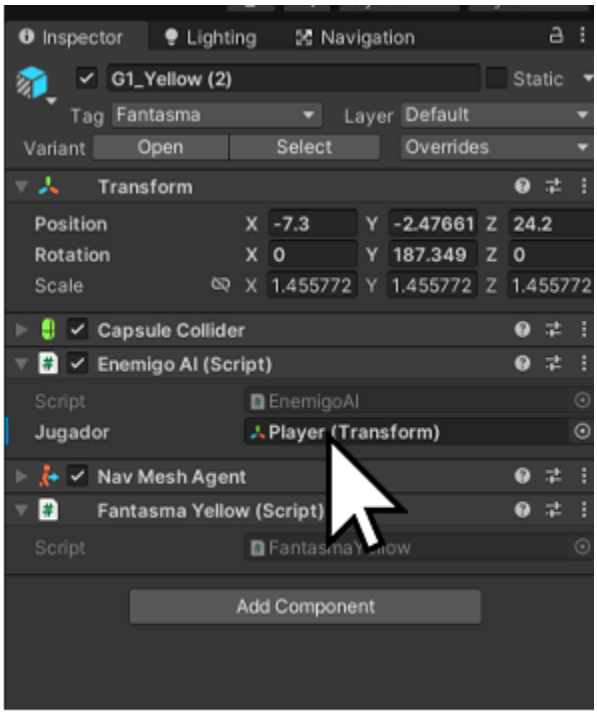
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;

public class EnemigoAI : MonoBehaviour
{
```

```
    // Para referenciar al jugador.
    public Transform jugador;
    // Para la navegación del enemigo.
    private NavMeshAgent agente;
    // La distancia mínima para detectar al jugador.
    private float distanciaMinima = 20.0f;
private void Start()
    {
        agente = GetComponent<NavMeshAgent>();

// Se llama a la función para que el enemigo tenga un nuevo destino inicial
        ObtenerNuevoDestino();
    }
private void Update()
    {
      // Se calcula la distancia entre el enemigo y el jugador.
        float distanciaJugador = Vector3.Distance(transform.position, jugador.position);

     // Aqui se ve si la distancia entre el enemigo y el jugador es menor que
     // La distancia mínima.
        if (distanciaJugador < distanciaMinima)
        {
            RaycastHit hit;   //Se agrego un raycast para detectar al jugador.

    // Aqui se realiza un raycast desde la posición del enemigo en dirección al jugador.
            if (Physics.Raycast(transform.position, jugador.position -
                transform.position, out hit, distanciaMinima))
            {
    // Si el raycast detecta al jugador con la etiqueta "Player"
                if (hit.collider.tag == "Player")
                {
    // Se actualiza la posición del objetivo para perseguir al jugador
                    agente.destination = jugador.position;
                }
            }
        }
        else
        {
     // Si el enemigo no tiene un camino el cual seguir.

            if (!agente.hasPath || agente.remainingDistance < 0.5f)
     // Se llama a la función para obtener un nuevo destino aleatorio.
                ObtenerNuevoDestino();
            }
        }
    }
private void ObtenerNuevoDestino()
    {
// Se obtiene una posición aleatoria en un rango de 20 unidades.
        Vector3 destino = Random.insideUnitSphere * 20.0f + transform.position;
// Se coloco para almacenar la información del NavMesh.
        NavMeshHit hit;

// Se obtiene una posición en el NavMesh que esté cerca del objetivo aleatorio.
```

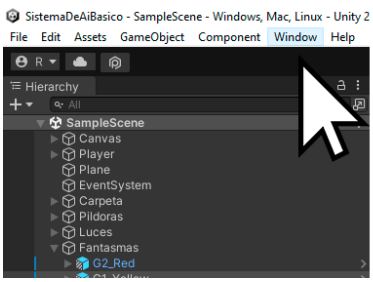
```
        NavMesh.SamplePosition(destino, out hit, 20.0f, NavMesh.AllAreas);
// Se establece la nueva posición.
        agente.destination = hit.position;
    }
}
```

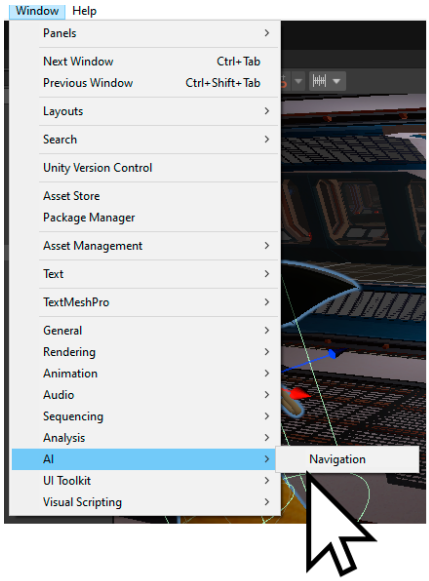
7. Le agregamos el script al enemigo y en el inspector, buscamos el script y donde dice “Jugador” arrastramos al player de la jerarquía y lo colocamos ahi.



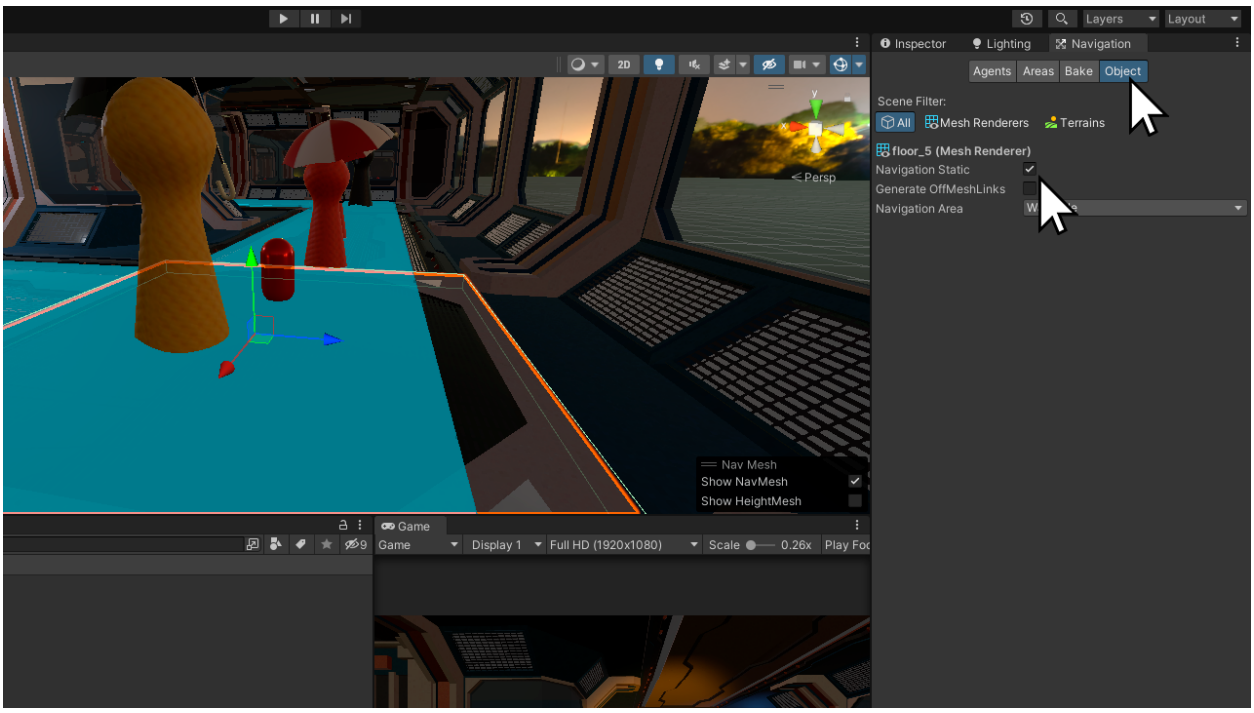


8. Una vez se creo el código, debes entrar a la pestaña Windows, AI, Navigation.

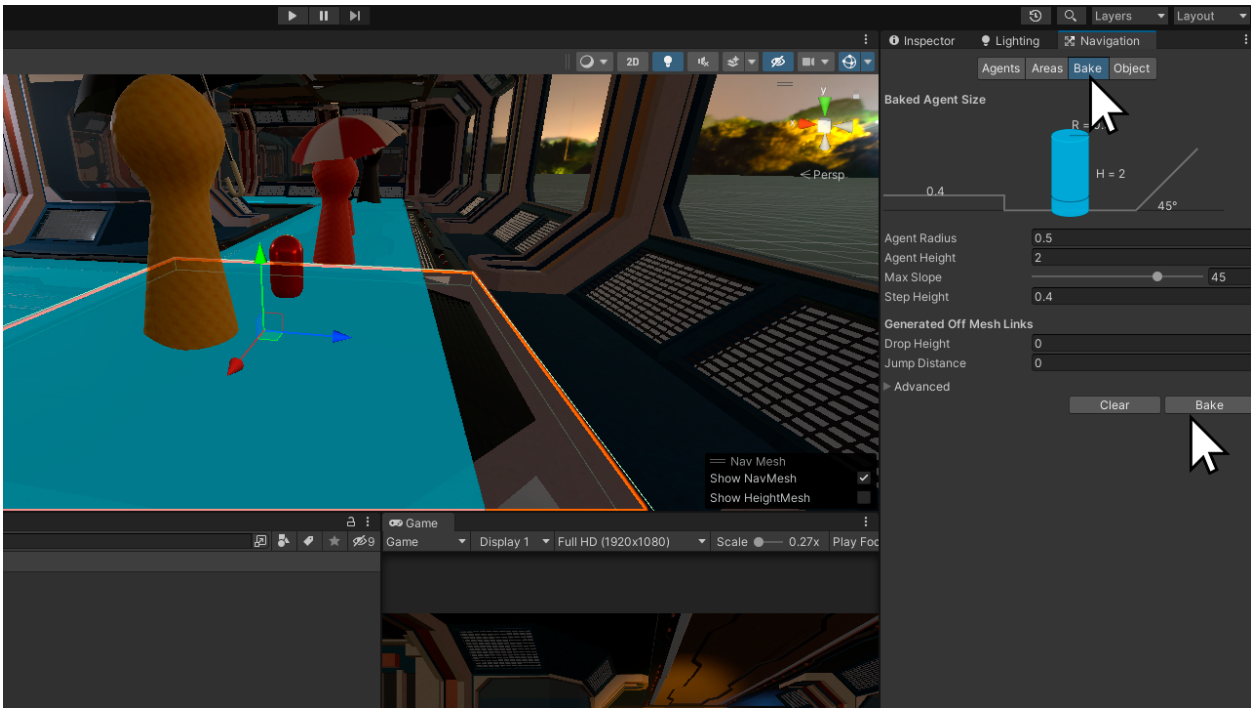




9. Vamos a seleccionar el suelo donde queremos que puedan moverse los enemigos.
10. En la parte derecha nos va a salir una opción llamada “Navigation”, dentro de ella presionamos donde dice “Object”, ahi vamos a presionar “Navigation Static”.



11. Una vez echo esto presionamos donde dice “Bake” y nuevamente mas abajo presionamos “Bake”, veras como el suelo seleccionado se pone azul.



12. Todas las partes que no estén azules, los enemigos no podrán pasar por ahí.

FantasmaBlack:

1. Al fantasma negro se le agrego un script extra, cuando toque al jugador este perderá automáticamente.

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

public class FantasmaBlack : MonoBehaviour
{
//Cuando el fantasma entre en un trigger este se activara
    private void OnTriggerEnter(Collider collision)
    {
//Si la colisión trae la etiqueta "Player".
        if(collision.gameObject.tag == "Player")
        {
//La escena se reiniciara.
```

```
            SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
        }
    }
}
```

FantasmaYellow:

1. Al fantasma amarillo se le agrego un script extra, cuando toque al jugador 2 veces este perderá automáticamente.

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

public class FantasmaYellow : MonoBehaviour
{
// Para contar cuántas veces ha sido tocado el objetivo.
    private int tocadoCount = 0;

//Cuando el fantasma entre en un trigger este se activara
    private void OnTriggerEnter(Collider collision)
    {
        if (collision.gameObject.tag == "Player")
        {
// Se incrementa el contador cada vez que se toca el objeto.
            tocadoCount++;

// Comprueba si se ha tocado el objeto dos veces o más
            if (tocadoCount >= 2)
            {
//La escena se reiniciara.
                SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
            }
        }
    }
}
```

FantasmaRed:

1. Al fantasma rojo se le agrego un script extra, cuando toque al jugador 5 veces este perderá automáticamente.

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

public class FantasmaYellow : MonoBehaviour
{
// Para contar cuántas veces ha sido tocado el objetivo.
    private int tocadoCount = 0;

//Cuando el fantasma entre en un trigger este se activara
    private void OnTriggerEnter(Collider collision)
    {
        if (collision.gameObject.tag == "Player")
        {
// Se incrementa el contador cada vez que se toca el objeto.
            tocadoCount++;

// Comprueba si se ha tocado el objeto cinco veces o más
            if (tocadoCount >= 5)
            {
//La escena se reiniciara.
                SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
            }
        }
    }
}
```