Sistema de diálogos.

Generales

Este repositorio fue creado con el objetivo de hacer que un sistema de diálogo sea funcional, a este mismo se le agregó un script el cual es el encargado de la funcionalidad.

1. Se creo un script llamado “Diálogo”
2. A este se le agrego varias variables

```
    public GameObject Admiracion; //El signo de admiración para hscer la alerta.
    public GameObject Mouse; //El mouse que explica como activar.
    private bool isPlayerInRange;//Esta me va a indicar si el jugador esta cerca del rango.
    private bool didDialogueStart;//Nos va a indicar que el dialogo comenzo.
    private int lineIndex;//Este nos va a mostrar que linea de dialogo nos esta mostrando.
    public float typingTime = 0.05f; //Tiempo de los caracteres.

    public GameObject PanelDeDialogo; //Para tener una referencia al panel del dialogo.
    public TMP_Text TextoDeDialogo; //Para tener una referencia al texto de la ui.
    [SerializeField, TextArea(4, 6)]private string[] dialogueLines; //Aqui vamos a escribir lo que va a salir en el dialogo.
```

3. En el update se agrego el momento en el cual se comienza el dialogo y la manera en como pasar al dialogo siguiente, también se podrá detener el dialogo.

```
void Update()
    {
        if(isPlayerInRange && Input.GetButtonDown("Fire1")) //El dialogo va a comenzar cuando el player este en el area y presiona el click
        {
            if(!didDialogueStart)
            {
                StartDialogue();
            }
            else if(TextoDeDialogo.text == dialogueLines[lineIndex]) //Si esta mostrando toda la frase completa pasamos a la siguiente line
            {
                NextDialogueLine();
            }
            else //Podre detener el dialogo y adelantarlo.
            {
                StopAllCoroutines();
                TextoDeDialogo.text = dialogueLines[lineIndex];
            }
        }
    }
```

4. En esta área es donde se activan los paneles para mostrar el texto, los símbolos que activan el dialogo se desactivan mientras el dialogo pase y todo en el juego se detiene hasta que acabe el dialogo.

```
void StartDialogue()
    {
        didDialogueStart = true; //Le estamos diciendo que ya comenzo el dialogo.
        PanelDeDialogo.SetActive(true); //Se activa el panel para mostrar el texto.
        Admiracion.SetActive(false);//Se desactiva el signo de admiración cuando esta el dialogo.
        Mouse.SetActive(false);//Se desactiva el mouse cuando esta el dialogo.
        lineIndex = 0; // Para que siempre que reactivemos el dialogo comience de 0.
        Time.timeScale = 0f; //Todo se detiene completamente mientras esta el dialogo.

        StartCoroutine(ShowLine());
    }
```

5. Aquí se activa el segundo dialogo al momento de terminar el primero y aquí se sabe si hay mas diálogos o ya finalizo el dialogo (si ya finalizo se activan los símbolos y el tiempo regresa a la normalidad.).

```
private void NextDialogueLine()//Para poder pasar a la siguiente linea de dialogo.
    {
        lineIndex++; //Esto me ayuda a pasar a la siguiente linea.
```

```
        if(lineIndex < dialogueLines.Length) //si este es menor quiere decir que aun no llegamos a la ultima linea.
        {
            StartCoroutine(ShowLine());
        }
        else //Si no hay mas lineas el dialogo ah terminado.
        {
            didDialogueStart = false;
            PanelDeDialogo.SetActive(false);
            Admiracion.SetActive(true);//Se activa el signo de admiración.
            Mouse.SetActive(true);//Se activa el mouse.
            Time.timeScale = 1f; // Regresa el juego a la normalidad una vez acabado el dialogo.
        }
    }
```

6. Aquí se va a controlar como va a funcionar el dialogo, comenzamos sin texto pero se va a comenzar a escribir letra por letra y el tiempo que toma de una letra a otra.

```
private IEnumerator ShowLine()
    {
        TextoDeDialogo.text = string.Empty; //El texto va a comenzar vacio

        foreach(char ch in dialogueLines[lineIndex])//Se va a ir escribiendo caracter por caracter.
        {
            TextoDeDialogo.text += ch;
            yield return new WaitForSecondsRealtime(typingTime); //El tiempo de letra en letra.
        }
    }
```

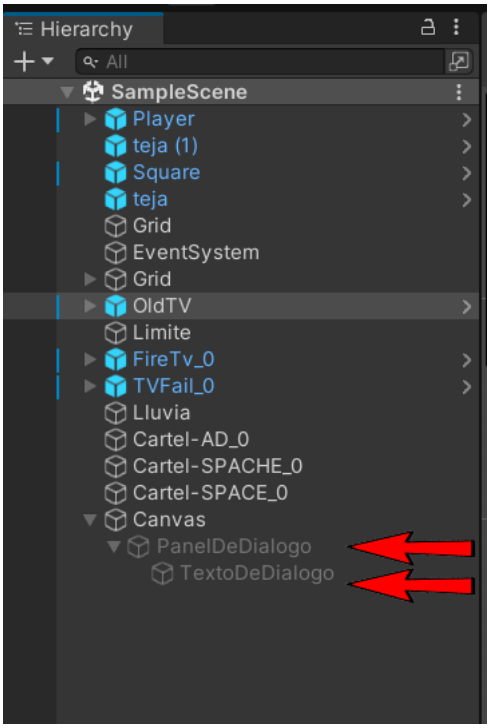
7. Este se activa cuando el jugador este dentro del área del objeto el cual va a tener el dialogo, lo cual va a activar los símbolos que indican el siguiente paso para activar el dialogo.
   Dentro de esta área se va a poder activar el dialogo.
   Se le dijo al script que solo el objeto que pase sobre el y tenga la etiqueta “Player” podrá activarlo.

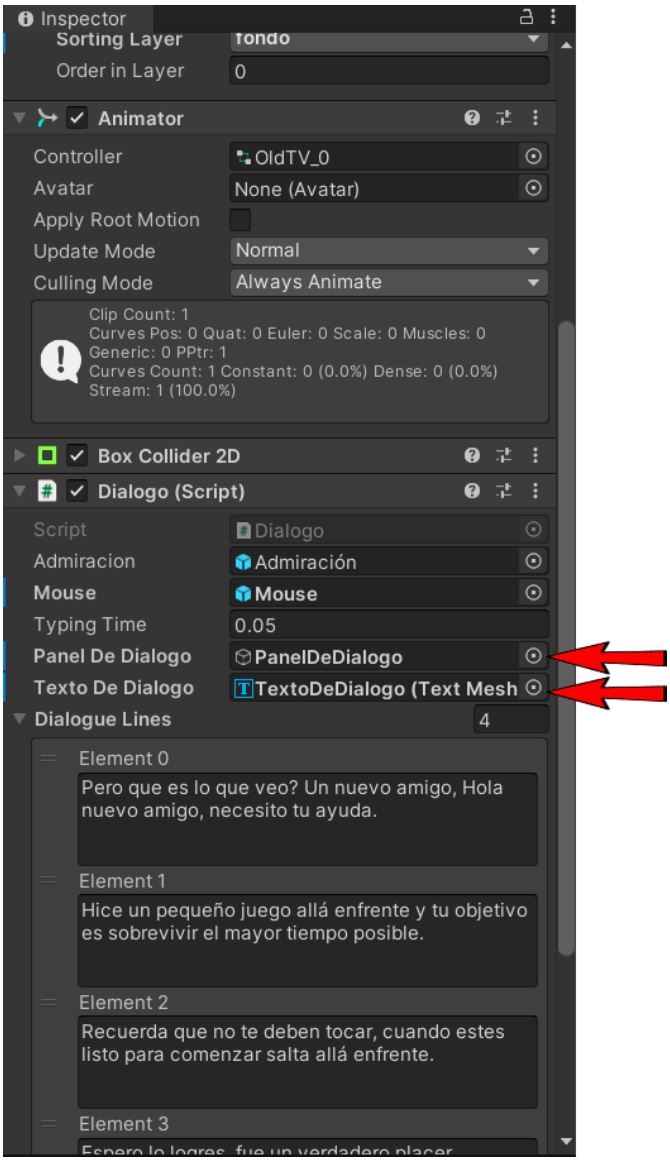
```
private void OnTriggerEnter2D(Collider2D collision) //Cuando pasen por el trigger este se va a activar.
    {
        if(collision.gameObject.CompareTag("Player")) // Solo el que tenga el tag de player lo puede activar.
        {
            isPlayerInRange = true;
            Admiracion.SetActive(true);//Se activa el simbolo de admiración al momento de estar cerca de la misión.
            Mouse.SetActive(true);//Se activa el mouse al momento de estar cerca de la misión.
        }

    }
```

8. Este se desactiva cuando el jugador sale del área del objeto, lo cual desactiva los símbolos que indican el siguiente paso y ya no se puede iniciar el dialogo.

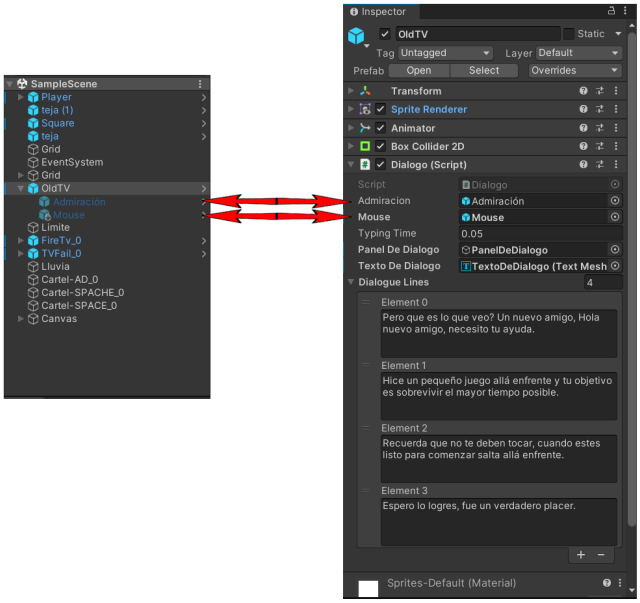
```
private void OnTriggerExit2D(Collider2D collision) //Cuando salga del trigger este se va a desactivar.
    {
        if (collision.gameObject.CompareTag("Player")) // Solo el que tenga el tag de player lo puede desactivar
        {
            isPlayerInRange = false;
            Admiracion.SetActive(false);//Se desactiva el simbolo de admiración al momento de alejarse de la misión.
            Mouse.SetActive(false);//Se desactiva el mouse.
        }
    }
```

Al momento de tener el código, en el unity se va a jalar el panel y el texto a sus cuadros correspondientes.

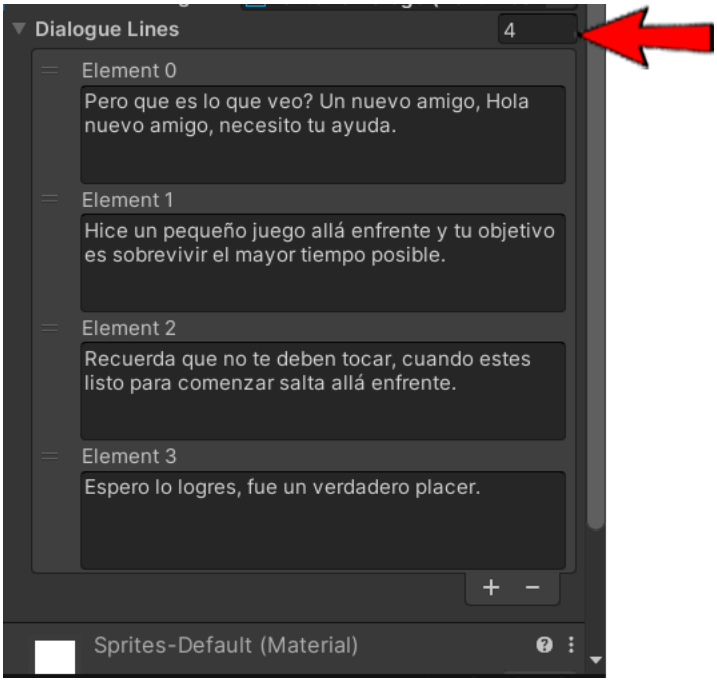

Hierarchy
All
SampleScene
Player
teja (1)
Square
teja
Grid
EventSystem
Grid
OldTV
Limite
FireTv_0
TVFail_0
Lluvia
Cartel-AD_0
Cartel-SPACHE_0
Cartel-SPACE_0
Canvas
PanelDeDialogo
TextoDeDialogo



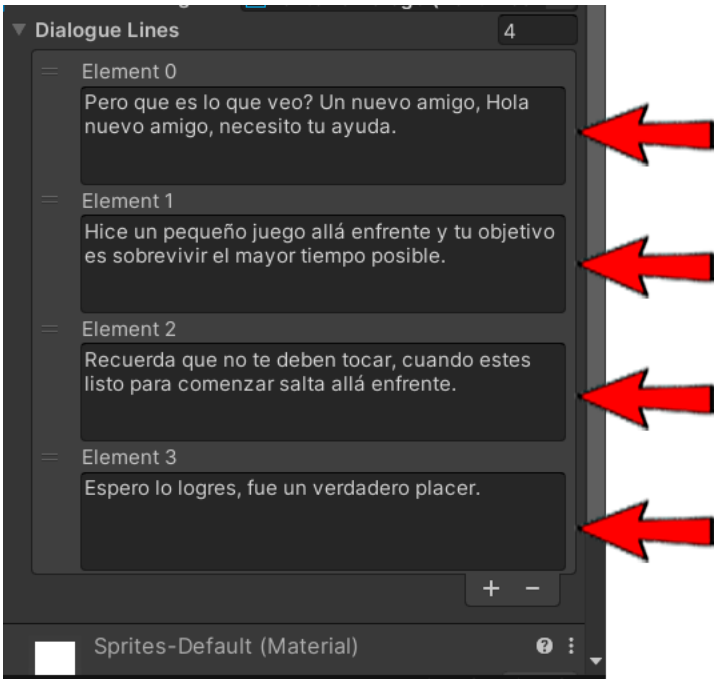
De igual manera se jalan las imágenes que se vayan a ocupar para que esta actúen como los símbolos que te avisan como sirve el juego.



En el siguiente recuadro escribe cuantos cuadros de texto deseas.



Comienza a escribir lo que deseas que diga.



Sistema de movimiento del player.

Este script fue creado con el objetivo de hacer que el Player pueda moverse dentro del juego y de esa manera activar los sistemas de dialogo.

1. Primero se agregaron las referencias para poder programarlo.}

```
//Movimiento
public float moveSpeed;
//Salto
private bool canDoubleJump;
public float jumpForce;
//Componentes
public Rigidbody2D rgb;
//Animaciones+
private Animator anim;
private SpriteRenderer theSR;
//Piso
private bool isGrounded;
public Transform groundCheckPoint;
public LayerMask whatIsGround;
```

2. Aquí le decimos que el jugador va a poder moverse en los planos (x, y) a la velocidad deseada en el unity.
   Le decimos que detecte el suelo todo objeto que tenga la etiqueta “isGrounded” para así hacer que solo pueda saltar cuando lo este tocando y no en cualquier área.
   También se dio la opción de tener un doble salto, pero no se puede reactivar hasta que vuelva a tocar suelo.
   Y se activa una animación en el jugador depende la dirección en la cual se mueva.

```
void Update()
    {
        rgb.velocity = new Vector2(moveSpeed * Input.GetAxis("Horizontal"), rgb.velocity.y);

        isGrounded = Physics2D.OverlapCircle(groundCheckPoint.position, 0.2f, whatIsGround);

        if(isGrounded)
```

```
        {
            canDoubleJump = true;
        }

        if(Input.GetButtonDown("Jump"))
        {
            if (isGrounded)
            {
                rgb.velocity = new Vector2(rgb.velocity.x, jumpForce);

            }
            else
            {
                if(canDoubleJump)
                {
                    rgb.velocity = new Vector2(rgb.velocity.x, jumpForce);
                    canDoubleJump = false;
                }
            }

        }
        if(rgb.velocity.x < 0)
        {
            theSR.flipX = true;
        }
        else
            if(rgb.velocity.x > 0)
        {
            theSR.flipX = false;
        }
        anim.SetFloat("moveSpeed", Mathf.Abs(rgb.velocity.x));
        anim.SetBool("isGrounded", isGrounded);
    }
}
```

Fondo movimiento.

Se creo un script para que el fondo se pueda mover junto con el jugador y el fondo tenga movimiento dando mas vida al juego

1. Se agregaron las variables para hacer que este script funcione.

```
public Vector2 velocidadMovimiento; // para el movimiento del fondo.
    private Vector2 offset;
    private Material material; //Referencia al material.
    private Rigidbody2D jugadorRB;//referencia al jugador.
```

2. Se agrego un material a los fondos para que este fuera detectado por es script.

```
private void Awake()
    {
        material = GetComponent<SpriteRenderer>().material; //El material es el mismo que tiene el sprite renderer.
        jugadorRB = GameObject.FindGameObjectWithTag("Player").GetComponent<Rigidbody2D>();
    }
```

3. Se agrego la velocidad de movimiento que va atener el fondo al momento de moverse.

```
private void Update()
    {
        offset = (jugadorRB.velocity.x * 0.1f) * velocidadMovimiento * Time.deltaTime;//el offset es igual a la velocidad de time delta tim
        material.mainTextureOffset += offset;
    }
```

Lluvia.

Se agrego un script a un objeto vacío el cual va a generar objetos en este caso rocas de manera aleatoria las cuales van caer en un espacio especifico.

1. se colocaron las variables para que este script funcionara.

```
public GameObject[] Rocas;
    public float SegundosSpawn = 0.5f;
    public float MinPiedra;
    public float MaxPiedra;
```

2. Colocamos una rutina para que vayan apareciendo rocas en esa área especifica.

```
void Start()
    {
        StartCoroutine(RockSpawn());
    }
```

3. Aquí le decimos que las rocas van a salir de manera aleatoria en toda la zona donde esta colocado el objeto vacío y cuanto tiempo va a tardar para salir una roca distinta.
   Y después de cierto tiempo la roca se va a destruir para no saturar el juego.

```
IEnumerator RockSpawn()
    {
        while(true)
        {
            var wanted = Random.Range(MinPiedra, MaxPiedra);
            var position = new Vector3(wanted, transform.position.y);
            GameObject gameObject = Instantiate(Rocas[Random.Range(0, Rocas.Length)], position, Quaternion.identity);
            yield return new WaitForSeconds(SegundosSpawn);
            Destroy(gameObject, 5f);
        }
    }
```

Menú principal y rocas.

Aquí se generaron 2 scripts que van de la mano para hacer mas interactivo el juego.
Cuando una roca toca al jugador este automáticamente es llevado al menú de inicio.

1. Se hizo el script de rocas, la cual al tocar al jugador este va a ser mandado a la primera escena el cual es un menú principal creado para este juego.

```
private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.tag == "Player")
        {
            SceneManager.LoadScene(0);//Cada que una de las rocas toque al player, se va a reiniciar el juego.
        }
    }
```

2. El menú principal va a contar con dos botones uno dice jugar y el otro dice salir.
   Al presionar jugar va a llevar a la segunda escena la cual es el juego y se va a jugar con normalidad.
   Al presionar salir, el juego se va a cerrar.

```
public void jugar()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1); //Para cambiar de escena al juego.
    }

    public void salir()
    {
```

```
        Application.Quit(); //Para cerrar el juego.
    }
```