Universidad Tecnológica de Santiago

(UTESA)

Realizado por

José Enrique Solís Acosta

(1-16-0632)

Presentado a

Iván Mendoza

Asignatura

Programación de videojuegos

Grupo

001

Tema

Tarea y actividad Semana 12

Dado el día 24 de abril en el año 2022, En la Ciudad de Santiago de los Caballeros, República Dominicana

ENLACES

- Link del repositorio: https://github.com/ESolis123/JuegoDeCartas.git
- Link de la carpeta donde se encuentan los capítulos del proyecto final:

https://github.com/ESolis123/JuegoDeCartas/tree/main/Documentos%20del%20proyecto%20final

CAPÍTULO III: DESARROLLO

3.1 Capturas de la Aplicación (Documentación completa del desarrollo, Scripts, Sprites, Prefabs e imágenes)

Menú inicio

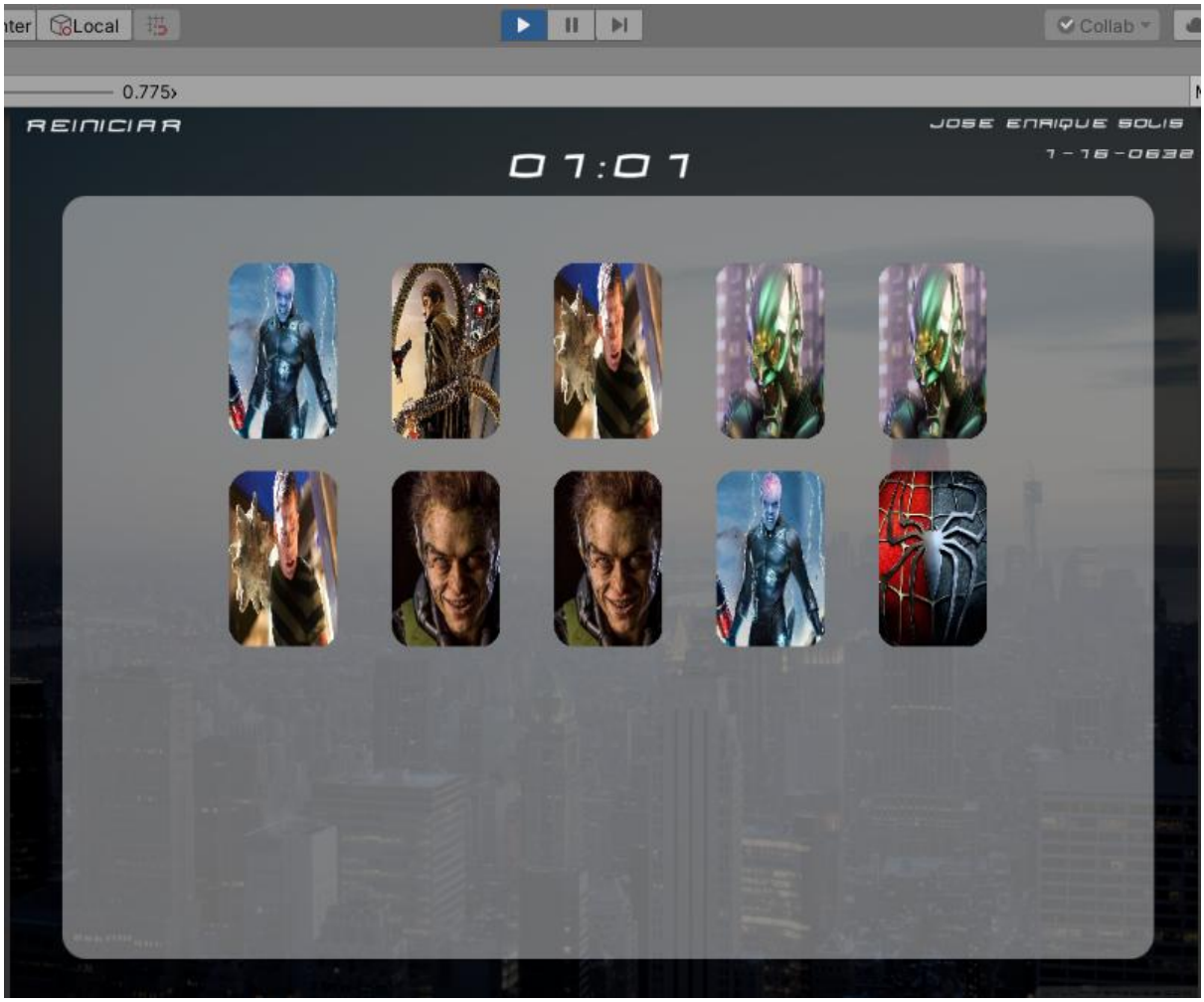
Menú configuraciones

Cambio de parámetros o settings

Información

Inicio de juego

Juego en progreso



Botón de pérdida e intentar de nuevo

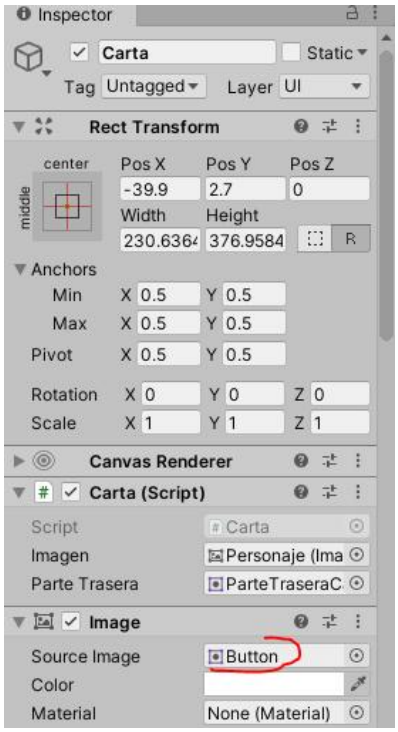
Botón de victoria

Sprites

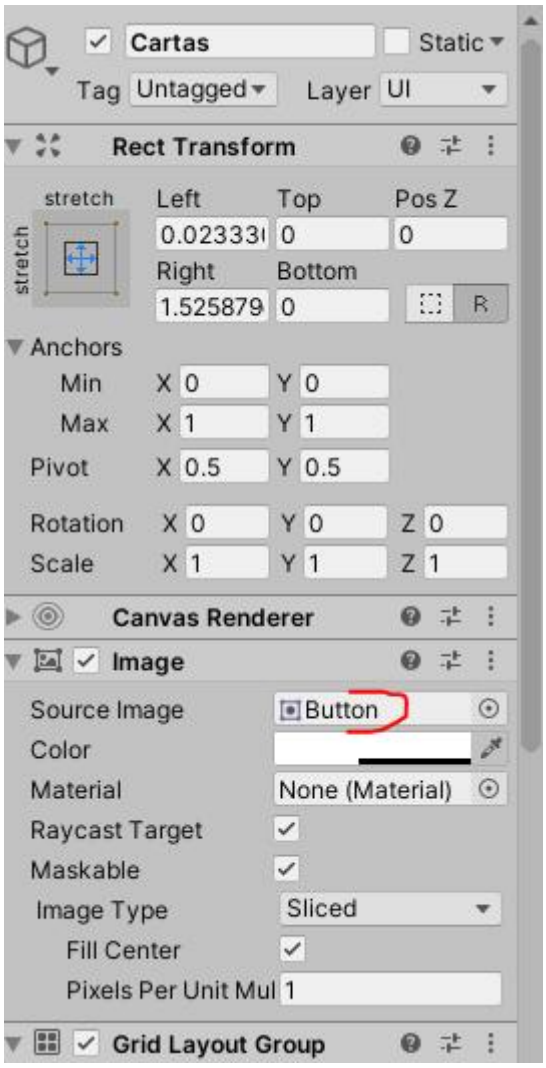
El único sprite creado particularmente para este proyecto fue este llamado botón o Button. Y se usa solamente como máscara de las cartas, para que la forma de estas sea ovalada, y para el fondo del tablero, pero con una opacidad más baja.

El otro sprite es este para el manejador o handler del slider:

Prefab de las cartas

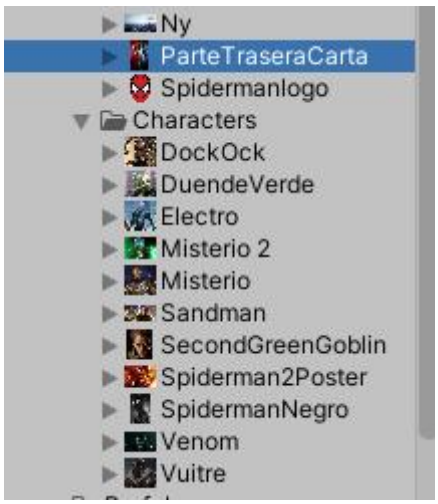


Gameobject de tablero



Imágenes

Las imágenes que se usan son las siguientes:



Fondo o background

Una imagen de la ciudad de Nueva York

La parte trasera de las cartas

Cuando están ocultas, se muestra la siguiente imagen:

Contenido de las cartas

Prefabs

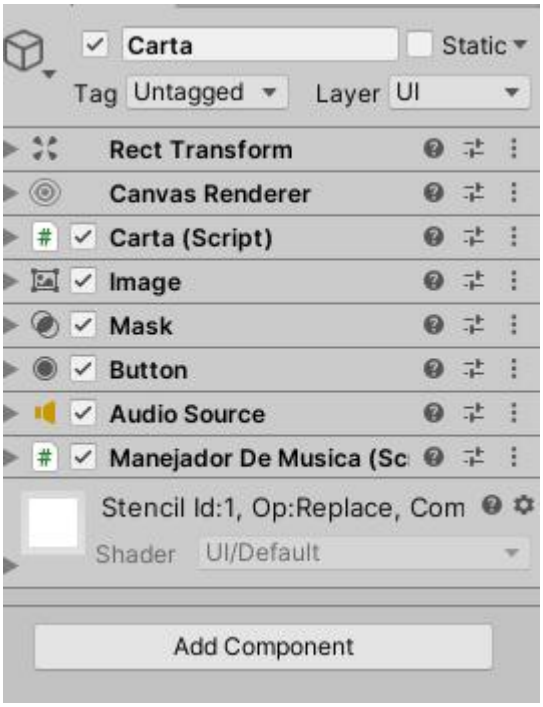
Hasta ahora hay un solo prefab, el de las cartas:

Está compuesto de la siguiente forma:

Carta contiene los siguientes componentes:

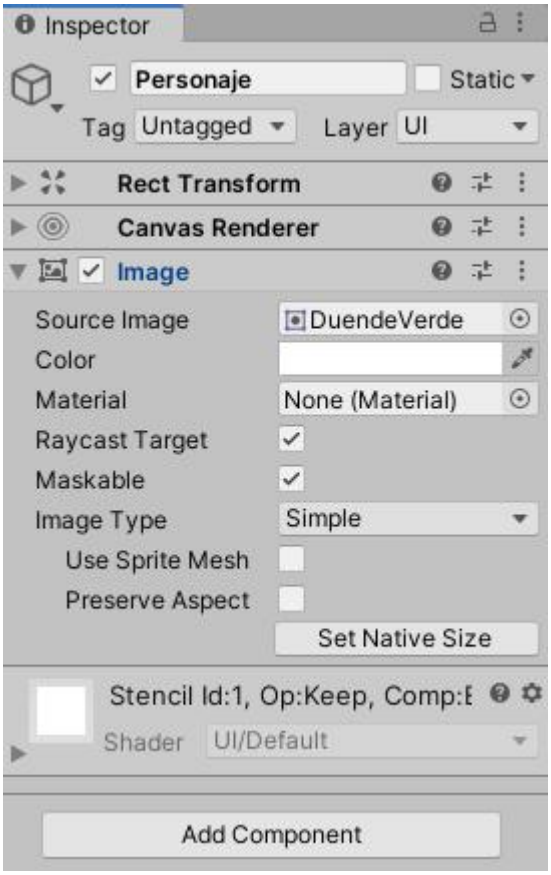
- **Carta (script):** Para el cambio entre el contenido (imagen de los villanos, héroes y demás) y la parte trasera. Además de las variables públicas que permiten saber al GameManager qué cartas han sido ya matcheadas y cuáles no y decidir si permitir mostrarlas o no.
- **Image**: La imagen de la máscara, osea el sprite button que describimos arriba.
- **Mask:** El componente que permite aplicar la máscara.
- **Button:** Cuando clickamos aquí mostramos la carta.

- **Audio Source:** Para disparar el sonido de la carta cuando gira.
- **Manejador de música:** para mutear el Audio source, si los settings lo indican.



Y **Personaje**, por otro lado, los siguientes:

Image: La parte delantera de la carta.



Scripts

Los scripts son los siguientes:

- **GameManager:** Este se encarga de repartir las casas, de hacer las comparaciones entre las cartas que están en el tablero, verificar si hay más de dos que están reveladas, y compararlas a ver si son iguales. Se encarga de descartar las cartas que ya han sido matcheadas y reveladas, además de terminar el juego.
- **Helper:** Este contiene una función genérica para no destruir los objetos. Hasta ahora sólo se está usando para el objeto de música.

```
1 reference | You, 6 hours ago | 1 author (You)
public class Helper : MonoBehaviour
{
    1 reference
    public static void NoDestruir (GameObject objeto, string tag)
    {
        var objs = GameObject.FindGameObjectsWithTag(tag);

        if (objs.Length > 1)
        {
            Destroy(objeto.gameObject);
        }       You, 6 hours ago • Escena inicio, Manejador de musica …

        DontDestroyOnLoad(objeto.gameObject);
    }
}
```

- **ManejadorDeMusica:** En este se controla el muteo de los Audiosource, ya sean de música o de efectos. De música sólo debe haber uno por escena, y de efectos todas las cartas tienen uno. En caso de que se haya elegido mutear los AudioSource de música, la variable booleana y estática música se hace false, y lo mismo con la de efectos. En la función Mutear se va verificando el estado de estas variables y se le aplica a los AudioSource.

```
6 references | You, 6 hours ago | 1 author (You)
public class ManejadorDeMusica : MonoBehaviour
{
    4 references | 4 references
    public static bool efectos = true, musica = true;
    3 references
    public Tipo tipo;

    5 references
    public enum Tipo
    {
        1 reference
        Efectos,
        2 references
        Musica
    }
       You, 6 hours ago • Escena inicio, Manejador de musica …
    0 references
    void Start()
    {
        if(tipo.Equals(Tipo.Musica))
            Helper.NoDestruir(gameObject, tipo.ToString());
    }

    0 references
    void Update()
    {
        Mutear();
    }

    1 reference
    void Mutear()
    {
        var tipos = new Dictionary<Tipo, bool>()
        {
            {Tipo.Efectos, efectos},
            {Tipo.Musica, musica},
        };

        GetComponent<AudioSource>().mute = !tipos[tipo];
    }
}
```

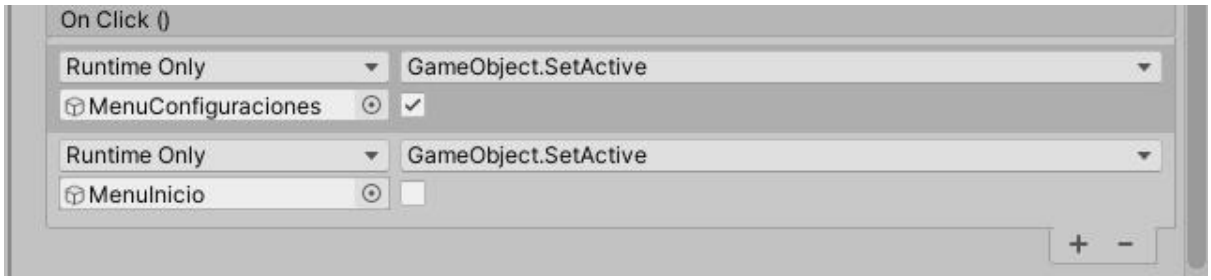
MenuConfiguraciones: Se encarga de asignar estas variables booleanas que mencionábamos y de cambiar el texto de los botones dependiendo de sus estados.

```
0 references | You, 2 hours ago | 1 author (You)
public class MenuConfiguraciones : MonoBehaviour
{
    1 reference | 1 reference
    public TextMeshProUGUI botonApagarMusica, botonApagarEfectos;

    0 references
    void Update()
    {
        botonApagarMusica.text = ManejadorDeMusica.musica ? "Apagar musica" : "Encender musica";
        botonApagarEfectos.text = ManejadorDeMusica.efectos ? "Apagar efectos" : "Encender efectos";
    }

    0 references
    public void ApagarMusica() => ManejadorDeMusica.musica = !ManejadorDeMusica.musica;

    0 references
    public void ApagarSonido() => ManejadorDeMusica.efectos = !ManejadorDeMusica.efectos;
}
```

- **MenuInicio:** Sólo contiene las funciones de Iniciar el juego, osea cambiar a la Scene Juego, o de cerrar la aplicación.

```
public class MenuInicio : MonoBehaviour
{
    0 references
    public void Iniciar() => SceneManager.LoadScene("Juego");
    0 references
    public void Salir() => Application.Quit();
}
```

Las demás cosas que se necesitan en el Menú inicio se controlan directamente, mediante el componente Button en el editor. Como por ejemplo en este botón de mostrar las configuraciones:



- **MenuParametros:** Con este cambiamos los settings del juego. Estos valores se modifican en la interfaz de usuario mediante Sliders.

```
3 references | You, 1 second ago | 1 author (You)
public class MenuParametros : MonoBehaviour
{
    4 references | 4 references | 4 references
    public static int maxCartas = 16, minCartas = 8, tiempo = 30;
    2 references | 2 references | 2 references
    public Slider cartasMinSlider, cartasMaxSlider, tiempoSlider;
    1 reference | 1 reference | 1 reference
    public TextMeshProUGUI actualMaxCartas, actualMinCartas, actualTiempo;
    0 references
    void Start()
    {
        cartasMaxSlider.value = maxCartas;
        cartasMinSlider.value = minCartas;
        tiempoSlider.value = tiempo;
    }
    0 references
    public void CambiarMaximoDeCartas() => maxCartas = (int) cartasMaxSlider.value;
    0 references
    public void CambiarMinimoDeCartas() => minCartas = (int) cartasMinSlider.value;
    0 references
    public void CambiarTiempo() => tiempo = (int) tiempoSlider.value;
    1 reference
    private void ActualizarValores()
    {
        actualTiempo.text = tiempo.ToString();
        actualMaxCartas.text = maxCartas.ToString();
        actualMinCartas.text = minCartas.ToString();
    }
    0 references
    private void Update() => ActualizarValores();      You, 1 second ago • Uncommitted c
}
```

- **Carta:** Para mostrar la carta, para ocultarla, para reproducir el sonido del giro, también contiene la función para asignarle la parte delantera y trasera.

```
public class Carta : MonoBehaviour
{
    3 references
    public Image imagen;
    1 reference
    public Sprite parteTrasera;

    3 references
    public string personaje => parteDelantera.name;

    [HideInInspector]
    4 references | 5 references
    public bool estaGirada, estaLista;
    3 references
    GameManager gameManager;
    3 references
    private Sprite parteDelantera;

    0 references
    private void Start()
    {
        gameManager = FindObjectOfType<GameManager>();
        Ocultar();
    }

    1 reference
    public void AsignarSprite(Sprite sprite, int tipo)
    {
        imagen.sprite = sprite;
        parteDelantera = sprite;
    }
```

```
public void LlamarGiro()      You, last month • Funcionamiento basico _
{
    Girar();
}

1 reference
private void Girar()
{
    if (!estaLista && !estaGirada && gameManager.numeroDeCartasGiradas < gameManager.limiteCartasGiradas)
    {
        Mostrar();
        PlayClip();
    }
}

1 reference
private void PlayClip()
{
    AudioSource audioSource = GetComponent<AudioSource>();
    audioSource.loop = false;
    audioSource.playOnAwake = false;
    audioSource.Play();
}

3 references
public void Ocultar()
{
    estaGirada = false;
    imagen.sprite = parteTrasera;
}
```

```
1 reference
private void Mostrar()
{
    estaGirada = true;
    imagen.sprite = parteDelantera;
}

2 references
public void Descartar()
{
    estaLista = true;
}
```

- **MusicaDeFondo:** Para reproducir la música de fondo. Tiene la opción de elegir entre un número random de clips, pero hasta ahora sólo tiene uno solo en el editor.

```
0 references | You, 6 hours ago | 1 author (You)
public class MusicaDeFondo : MonoBehaviour
{
    2 references
    public AudioClip[]  clips;
    5 references
    AudioSource fuenteAudio;

    0 references
    void Start()
    {
        fuenteAudio = GetComponent<AudioSource>();
        PlayClip(clips[Random.Range(0, clips.Length)]);
    }

    1 reference
    public void PlayClip(AudioClip clip)
    {
        if(!fuenteAudio.isPlaying)
        {
            fuenteAudio.clip = clip;
            fuenteAudio.loop = true;
            fuenteAudio.Play();
        }
    }       You, last month • Diseño, sonidos, musica y ma
}
```

- **Temporizador:** Para controlar el tiempo, y para mostrar los botones de victoria o pérdida.

```
public class Temporizador : MonoBehaviour
{
    0 references
    public TextMeshProUGUI BotonDeIntentarDeNuevo;
    5 references
    private float tiempo;
    2 references | 2 references
    int minutos, segundos;
    2 references
    TextMeshProUGUI texto;

    4 references
    GameManager gameManager;
    0 references
    void Start()        You, last month • Juego terminado …
    {
        gameManager = FindObjectOfType<GameManager>();
        tiempo = MenuParametros.tiempo;
        texto = GetComponent<TextMeshProUGUI>();
    }
```

```
1 reference
void MostrarTiempo()
{
    tiempo -= Time.deltaTime;
    minutos = (int) tiempo/60;
    segundos = (int) tiempo%60;
    texto.text = $"{minutos.ToString("00")}:{segundos.ToString("00")}";
}

1 reference
void TerminarJuego()
{
    if(tiempo <=0)
    {
        gameManager.PresentarBotonDeIntentarDeNuevo("Perdiste \n intenta de nuevo");
        gameManager.juegoEnProceso = false;
    }
}

0 references
void Update()
{
    if(gameManager.juegoEnProceso)
    {
        MostrarTiempo();
        TerminarJuego();
    }
}
```

- **TextoMenu:** Este es para cambiar el color de los botones cuando se les pasa por encima con el puntero del mouse.

```
0 references | You, 6 hours ago | 1 author (You)
public class TextoMenu : MonoBehaviour
{
    2 references
    Color colorPrevio;
    4 references
    TextMeshProUGUI componenteTexto;

    0 references
    void Start()
    {
        componenteTexto = GetComponentInChildren<TextMeshProUGUI>();
    }

    0 references
    public void OnMouseOver()
    {
        Debug.Log($"Onselect Nombre: {name}");
        colorPrevio = componenteTexto.color;
        componenteTexto.color = Color.yellow;
    }       You, 6 hours ago • Escena inicio, Manejador de musica …

    0 references
    public void OnMouseExit()
    {
        Debug.Log($"OnDeselect Nombre: {name}");
        componenteTexto.color = colorPrevio;
    }
}
```

3.2 Prototipos

- **Lo-Fi**, con sólo la escena del juego, la música de fondo y la opción de reiniciar.
- **Hi-Fi**, con las dos escenas: Inicio, con los menús de inicio, configuraciones, y parámetros, además de la escena del juego ya descrita.

3.3 Perfiles de Usuarios

El público objetivo es uno que está comprendido entre las edades de 8 en adelante. Se eligió 8 ya que las imágenes de algunos personajes pueden resultar inapropiadas para un público menor a esta edad. Y no hay una edad tope, ya que cualquier persona puede disfrutar del juego.

3.4 Usabilidad

Según los reportes de las pruebas, hasta el momento dos de familiares cercanos, el juego es fácilde usar, comprender y es entretenido.

3.5 Test

Prueba 1

Sexo	Femenino
Edad	52
Nivel de estudios	Licenciada
Aficiones	Leer, Ver novelas y series

Tareas	**Puntuación**
Jugabilidad	5
Dificultad	3
Control del personaje	- No hay personajes
Guía del usuario	4
La información proporcionada por el juego	4
Diseño visual 2	5
La coherencia 4	5

Prueba 2

Sexo	Femenino
Edad	19
Nivel de estudios	Universitaria
Aficiones	Leer, ver series y jugar juegos en el celular

Tareas	**Puntuación**
Jugabilidad	5
Dificultad	2
Control del personaje	-No hay personajes
Guía del usuario	3
La información proporcionada por el juego	3
Diseño visual 2	4
La coherencia 4	4

3.6 Versiones de la Aplicación

La versión beta va a ser el mismo prototipo Lo-fi, y la Alpha la Hi-Li.