**Informe de hallazgos y validación del framework “Unity ETH UQ” para el proyecto:**

**Desarrollo de un videojuego haciendo uso del framework “Unity ETH UQ”.**

**Dival Mauricio Hoyos Castro**

**Julián David Serna Echeverri**

2016

Facultad de Ingeniería – Programa de Ingeniería de Sistemas y Computación

Universidad del Quindío

**Contenido**

[**Introducción** 4](#_Toc460889819)

[**Implementación de funcionalidades** 5](#_Toc460889820)

[**Módulo Audio** 5](#_Toc460889821)

[**Reproducir un audio** 5](#_Toc460889822)

[**Reproducir un efecto** 6](#_Toc460889823)

[**Reproducir un efecto de forma cíclica** 6](#_Toc460889824)

[**Modificar el volumen de la música** 7](#_Toc460889825)

[**Modificar el volumen de los efectos** 8](#_Toc460889826)

[**Observaciones del módulo** 9](#_Toc460889827)

[**Módulo Util** 10](#_Toc460889828)

[**Manejo de Timer** 10](#_Toc460889829)

[**Definir si la plataforma en la que se ejecuta la aplicación es móvil** 11](#_Toc460889830)

[**Modificar la visibilidad de un Game Object** 12](#_Toc460889831)

[**Obtener un Game Object hijo a partir de su nombre y su padre** 12](#_Toc460889832)

[**Observaciones del módulo** 13](#_Toc460889833)

[**Módulo Components** 13](#_Toc460889834)

[**Soporte de idiomas a imagen** 14](#_Toc460889835)

[**Soporte de idiomas a texto** 14](#_Toc460889836)

[**Reproducción de audio a un toggle** 16](#_Toc460889837)

[**Soporte de idiomas a botón** 17](#_Toc460889838)

[**Reproducción de audio a un botón** 18](#_Toc460889839)

[**Modificar el valor de progreso de una barra de progreso** 20](#_Toc460889840)

[**Animar la barra de progreso cuando se modifica su valor** 20](#_Toc460889841)

[**Cargar una escena** 20](#_Toc460889842)

[**Al cargar una escena crear un panel sobre el canvas y mostrar un cargador** 21](#_Toc460889843)

[**Obtener el lenguaje del sistema** 21](#_Toc460889844)

[**Observaciones del módulo** 22](#_Toc460889845)

[**Módulo AppSystem** 23](#_Toc460889846)

[**Inicializar el sistema de analíticas** 23](#_Toc460889847)

[**Registrar un evento en el sistema de analíticas** 24](#_Toc460889848)

[**Registrar una pantalla en el sistema de analíticas** 24](#_Toc460889849)

[**Obtener y modificar variables registradas en el sistema de analíticas** 25](#_Toc460889850)

[**Observaciones del módulo** 25](#_Toc460889851)

# **Introducción**

Inicialmente el presente proyecto de grado estaba delimitado por la elaboración de un videojuego para la plataforma Android, que permitiera evaluar los componentes *util*, *display*, *audio* y *analythics* del proyecto “Framework de desarrollo de aplicaciones y juegos para sistemas IOS y Android en Unity con sistema de analíticas de uso y marketing” desarrollado por la empresa ETHEREAL GF y cuyo nombre clave es “Unity ETH UQ”. Pero en el trascurso de la etapa inicial del proyecto, la cual consistió en identificar las características, requerimientos y funcionalidades de dicho framework, Andrés Carvajal, administrador de proyecto de Ethereal GF informó al equipo de trabajo que debido a que la versión 5 de Unity 3D soporta elementos de interfaz de usuario el framework fue modificado y el módulo *display* fue reemplazado por el módulo *components*. Así mismo Andrés Herrera, desarrollador de Ethereal GF quien es el encargado de la mejora continua del módulo de analíticas informó al equipo de trabajo que dicho módulo había sido actualizado a la versión 2, lo cual implicaba que la administración de analíticas ya no se realizara en el módulo *analytics* sino en el módulo *appSystem*. Quedando así el presente proyecto delimitado por la elaboración de un videojuego para la plataforma Android, que permitiera evaluar los componentes *util*, *components*, *audio* y *appSystem* del framework “Unity ETH UQ”.

De acuerdo a las funcionalidades de los anteriores módulos, identificadas en la primera fase del proyecto, las cuales se listan en la Tabla 1, se implementó cada una de ellas en el videojuego para probar su funcionamiento como se explica en el transcurso de este documento.

|  |  |
| --- | --- |
| **Módulo** | **Funcionalidad** |
| Audio | Reproducir un audio |
| Reproducir un efecto |
| Reproducir un efecto de forma cíclica |
| Modificar el volumen de la música |
| Modificar el volumen de los efectos |
| Util | Manejo de Timer |
| Definir si la plataforma en que se ejecuta la aplicación es móvil |
| Modificar la visibilidad de un game object |
| Obtener un game object hijo a partir de su nombre y su padre |
| Components | Soporte de idiomas a imagen |
| Soporte de idiomas a texto |
| Reproducción de audio a un toggle |
| Soporte de idiomas a botón |
| Reproducción de audio a un botón |
| Modificar el valor de progreso a una barra de progreso |
| Animar la barra de progreso cuando se modifica su valor |
| Registrar una pantalla para llamarla cuando se usa la tecla de retroceso |
| Cargar una escena |
| Al cargar una escena crear un panel sobre el canvas y mostrar un cargador |
| Obtener el lenguaje del sistema |
| AppSystem | Inicializar el sistema de analíticas |
| Registrar un evento en el sistema de analíticas |
| Registrar una pantalla en el sistema de analíticas |
| Obtener y modificar variables registradas en el sistema de analíticas |

*Tabla 1. Funcionalidades del framework*

# **Implementación de funcionalidades**

A continuación se describe la implementación de cada una de las funcionalidades en el videojuego, sus observaciones en caso de haberlas y sus conclusiones, con el objetivo de validar el funcionamiento de cada funcionalidad.

## **Módulo Audio**

La totalidad de las funcionalidades de este módulo fueron implementadas en el videojuego, como lo indica la Tabla 2. A continuación se detalla el proceso de implementación.

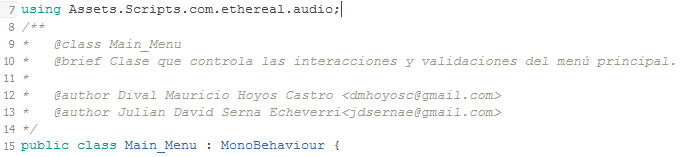
|  |  |
| --- | --- |
| **Funcionalidad** | **Implementación exitosa** |
| Reproducir un audio | Si |
| Reproducir un efecto | Si |
| Reproducir un efecto de forma cíclica | Si |
| Modificar el volumen de la música | Si |
| Modificar el volumen de los efectos | si |

*Tabla 2. Funcionalidades implementadas del módulo audio*

### **Reproducir un audio**

Su implementación se realizó de tal manera que al iniciar el menú principal se reproduce una canción, la cual se repite mientras el juego esté abierto actuando como música de fondo. Para su implementación se realizaron los siguientes pasos:

* En la carpeta “*Resources”* del videojuego se alojó el archivo a reproducir.
* Se importó el módulo *audio* en el script donde se iba a dar la instrucción de reproducir el audio como se muestra en la línea 7 de la Ilustración 1.
* En el método “*Start”* del script se invocó la función *“PlayMusic”* de la clase “*EthAudio”* a la cual se le ingresa como parámetro la ruta del archivo a reproducir dentro de la carpeta *“Resources”* como se muestra en la Ilustración 2.



*Ilustración 1. Importación del módulo audio*



*Ilustración 2. Invocación de la función PlayMusic*

#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que ejecuta un sonido de fondo mientras la aplicación está en ejecución.

### **Reproducir un efecto**

Su implementación se realizó de tal manera que cuando el personaje principal realiza un ataque con lanza se reproduce un efecto de sonido y también cuando el cocodrilo ataca y mata al personaje principal. Esto se llevó a cabo realizando los siguientes pasos:

* En la carpeta “Resources*”* del videojuego se alojó el archivo a reproducir.
* Se importó el módulo *audio* en el script donde se iba a dar la instrucción de reproducir el efecto como se muestra en la línea 7 de la Ilustración 1.
* En la sección del script “Character\_Controller” donde se realizó la validación del ataque con lanza se invocó la función “PlayEffect” de la clase “*EthAudio”* a la cual se le ingresa como parámetro la ruta del archivo a reproducir dentro de la carpeta *“Resources”* como se muestra en la Ilustración 3, de esta manera cada que el personaje realiza un ataque con lanza se reproduce el efecto.

C:\Users\Dival\Downloads\14248796_10210391184194112_1334224247_n.png

*Ilustración 3. Invocación de la función PlayEffect*

#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que reproduce el efecto de sonido una única vez cada que se realiza el ataque con lanza.

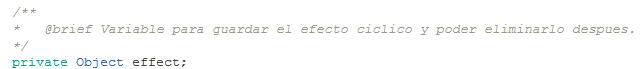
### **Reproducir un efecto de forma cíclica**

Su implementación se llevó a cabo en diferentes secciones del videojuego, las cuales fueron:

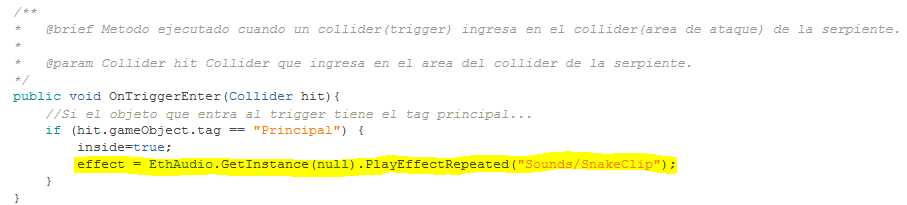
* Cuando el personaje está cerca de una serpiente se reproduce el audio de la serpiente atacando cíclicamente hasta que esté la mate o se aleje.
* Cuando el personaje está cerca de un oso se reproduce el audio de la serpiente atacando cíclicamente hasta que esté lo mate o se aleje.
* Cuando el personaje está cerca de un tigre se reproduce el audio de la serpiente atacando cíclicamente hasta que esté lo mate o se aleje.

Para su implementación en cada sección se realizaron los siguientes pasos:

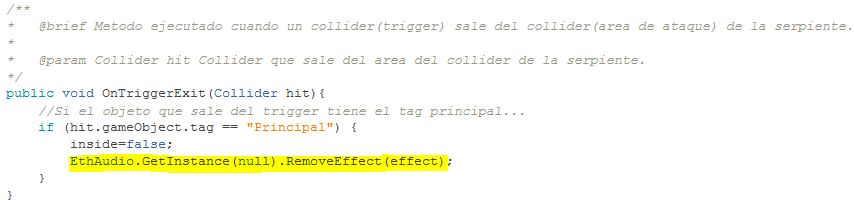
* En la carpeta “Resources*”* del videojuego se alojó el archivo a reproducir.
* Se importó el módulo *audio* en el script donde se iba a dar la instrucción de reproducir el efecto como se muestra en la línea 7 de la Ilustración 1.
* Se creó una variable privada en el script para guardar el efecto que se va a reproducir cíclicamente para poderlo parar posteriormente como se muestra en la Ilustración 4.
* En el método que notifica cuando el personaje principal está cerca del rival se utilizó el método *“PlayEffectRepeated”* de la clase *“EthAudio”* que recibe como parámetro la ruta del sonido dentro de la carpeta *“Resources”* como se muestra en la Ilustración 5.
* Dicho método se le asigna a la variable creada en el paso tres ya que este retorna el clip que se está reproduciendo.
* En el método que notifica cuando el personaje se aleja se cancela la reproducción del efecto utilizando el método *“RemoveEffect”* de la clase *“EthAudio”* que recibe como parámetro el efecto a eliminar como se muestra en la Ilustración 6.



*Ilustración 4. Creación de la variable para guardar el efecto*



*Ilustración 5. Utilización del método “PlayEffectRepeated” de la clase “EthAudio”*



*Ilustración 6.* Utilización del método *“RemoveEffect”* de la clase *“EthAudio”*

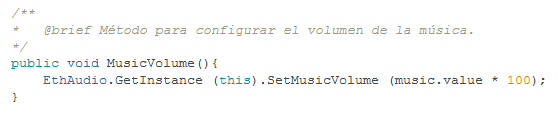
#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que reproduce el efecto de sonido cíclicamente hasta que el personaje se aleja del rival.

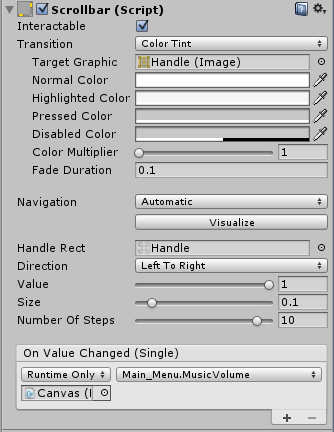
### **Modificar el volumen de la música**

Su implementación se llevó a cabo en el menú de configuraciones, de tal manera que cuando el usuario desplace la barra que indica el volumen de la música se haga uso de la funcionalidad del framework y el volumen aumente o disminuya según el desplazamiento del usuario. Esto se llevó a cabo realizando los siguientes pasos:

* Se creó un “*ScrollBar”*  que sería con el cual el usuario interactuaría.
* Se importó el módulo *audio* en el script donde se iba a dar la instrucción de reproducir el audio como se muestra en la línea 7 de la Ilustración 1.
* Se creó un método en el script *“Main\_Menu”* que hiciera uso del método *“SetMusicVolume”* de la clase “*EthAudio”,* al cual se le ingresó el valor del “*ScrollBar”*, el cual se muestra en la Ilustración 7.
* Se le asignó al “*ScrollBar”*  en su propiedad *“On Value Change”* el método creado en el paso anterior como se muestra en la Ilustración 8.



*Ilustración 7. Método que utiliza el método SetMusicVolume de la clase EthAudio*



*Ilustración 8. Asignación del método que modifica el volumen de la música al scroll*

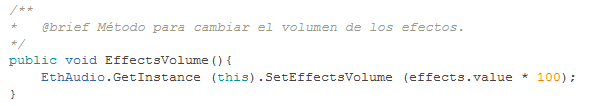
#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que al desplazar el scroll el volumen de la música de fondo se modifica.

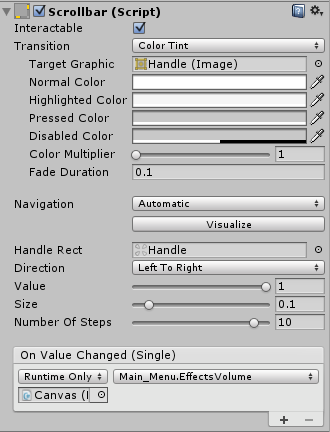
### **Modificar el volumen de los efectos**

Su implementación se llevó a cabo en el menú de configuraciones, de tal manera que cuando el usuario desplace la barra que indica el volumen de los efectos se haga uso de la funcionalidad del framework y el volumen aumente o disminuya según el desplazamiento del usuario. Esto se llevó a cabo realizando los siguientes pasos:

* Se creó un “*ScrollBar”*  que sería con el cual el usuario interactuaría.
* Se importó el módulo *audio* en el script donde se iba a dar la instrucción de reproducir el audio como se muestra en la línea 7 de la Ilustración 1.
* Se creó un método en el script *“Main\_Menu”* que hiciera uso del método *“SetEffectsVolume”* de la clase “*EthAudio”,* al cual se le ingresó el valor del “*ScrollBar”*, el cual se muestra en la Ilustración 9.
* Se le asignó al “*ScrollBar”*  en su propiedad *“On Value Change”* el método creado en el paso anterior como se muestra en la Ilustración 10.



*Ilustración 9. Método que utiliza el método SetEffectsVolume de la clase EthAudio*



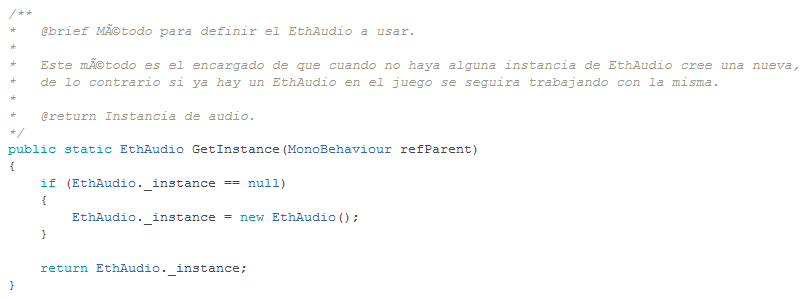
*Ilustración 10. Asignación del método que modifica el volumen de los efectos al scroll*

#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que al desplazar el scroll el volumen de los efectos se modifica.

### **Observaciones del módulo**

Para obtener la instancia actual de la clase EthAudio se utiliza el método *“GetInstance”,* el cualrecibe un parámetro que nunca es usado, por lo cual se sugiere la eliminación de este parámetro en el código fuente del framework. En la Ilustración 11 se muestra el método que debería ser modificado.



*Ilustración 11. Método GetInstance que utiliza un parámetro innecesario*

## **Módulo Util**

La totalidad de las funcionalidades de este módulo fueron implementadas en el videojuego, como lo indica la Tabla 3. A continuación se detalla el proceso de implementación.

|  |  |
| --- | --- |
| **Funcionalidad** | **Implementación exitosa** |
| Manejo de Timer | Si |
| Definir si la plataforma en que se ejecuta la aplicación es móvil | Si |
| Modificar la visibilidad de un game object | Si |
| Obtener un game object hijo a partir de su nombre y su padre | Si |

*Tabla 3. Funcionalidades implementadas del módulo util*

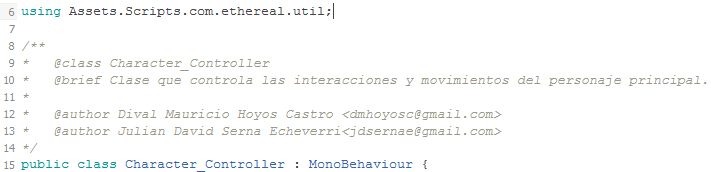
### **Manejo de Timer**

Los EthTimer tienen como funcionalidad iniciar una espera y posteriormente ejecutar un método, los cuales fueron utilizados en las siguientes secciones del videojuego:

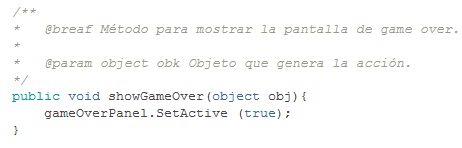
* Después de iniciar la animación de ataque con cerbatana esperar medio segundo y generar el dardo.
* Un segundo después de iniciar un ataque la variable auxiliar que define si el personaje está realizando un ataque cambie.
* Un segundo y medio después del personaje morir muestre la pantalla de game over.
* Hacer que el cocodrilo despierte después de cinco segundos de haber sido aturdido.

Para su implementación en cada sección se realizaron los siguientes pasos:

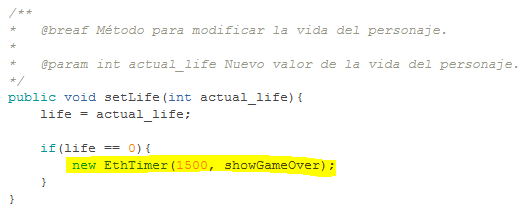
* Se importó el módulo *util* en el script donde se iba a dar la instrucción de reproducir el audio como se muestra en la línea 6 de la Ilustración 12.
* Se creó un método que realice la acción deseada cuando pase el tiempo del EthTimer. En la Ilustración 13 se muestra el ejemplo del método para mostrar el menú de game over.
* En la sección pertinente del script se genera el EthTimer indicándole el tiempo en milisegundos que desea esperar antes de ejecutar el método que se ingresa como segundo parámetro. En la imagen 14 se muestra la creación del EthTimer para mostrar la ventana de game over.



*Ilustración 12. Importación del módulo util*



*Ilustración 13. Método para mostrar la ventana de game over*



*Ilustración 14. Generación del EthTimer para mostrar la ventana de game over*

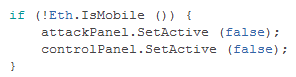
#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que al crearse el EthTimer en las diferentes secciones después del tiempo especificado se ejecuta el método ingresado.

### **Definir si la plataforma en la que se ejecuta la aplicación es móvil**

Su implementación se llevó a cabo de tal forma que si la plataforma sobre la que se ejecuta la aplicación no es móvil los controles de movimiento y ataque no se muestren en pantalla debido a que se usarán las teclas. Esto se llevó a cabo realizando los siguientes pasos:

* Se importó el módulo *util* en el script donde se iba a dar la instrucción de reproducir el audio como se muestra en la línea 6 de la Ilustración 12.
* En el método *“Start”* del Script se realiza la validación utilizando el método *“IsMobile”* de la clase *“EthUtils”* como se muestra en la Ilustración 15.



*Ilustración 15. Validación utilizando el método “IsMobile” de la clase* *“EthUtils”*

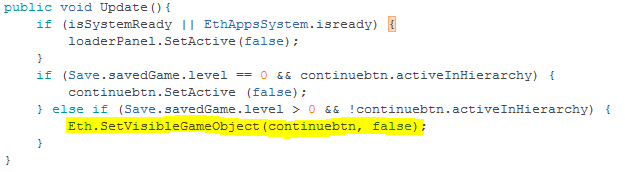
#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que los controles de movimiento y ataque no se muestran cuando la aplicación no se ejecuta en un dispositivo móvil.

### **Modificar la visibilidad de un Game Object**

Su implementación se llevó a cabo en el botón continuar del menú principal, de tal manera que si el usuario no tiene una partida iniciada en un nivel superior al tutorial este botón este oculto. Esto se llevó a cabo realizando los siguientes pasos:

* Se importó el módulo *util* en el script donde se iba a dar la instrucción de reproducir el audio como se muestra en la línea 6 de la Ilustración 12.
* En el método *“Update”* de la clase *“Main\_Menu”* se realizó la validación si existía una partida iniciada se ocultara el botón continuar utilizandoel método *“SetVisibleGameObject”* de la clase “EthUtils”, el cual recibe como parámetros el objeto al cual se le cambia la visibilidad y el estado de dicha visibilidad, como se muestra en la Ilustración 16.



*Ilustración 16. Validación para ocultar el botón continuar*

#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que cuando el usuario no tiene una partida iniciada en un nivel superior al tutorial el botón continuar no se muestra en el menú principal.

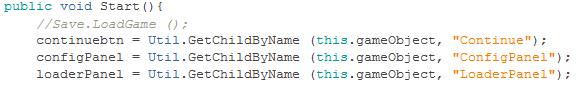
### **Obtener un Game Object hijo a partir de su nombre y su padre**

Esta funcionalidad se utilizó en diferentes secciones del videojuego las cuales fueron:

* Para que cuando un rival muera, este obtenga su collider y lo cambie a trigger para que el personaje pueda continuar su camino.
* Para obtener el botón continuar, la ventana de configuración y el cargador para modificar su visibilidad al iniciar el menú.

Para su implementación en cada sección se realizaron los siguientes pasos:

* Se importó el módulo *util* en el script donde se iba a dar la instrucción de reproducir el audio como se muestra en la línea 6 de la Ilustración 12.
* En el método “*Start”* de la clase *“Main\_Menu”* se hizo uso del método *“GetChildByName”* de la clase “EthUtils” al que se le ingresan como parámetro el Game Object padre y el nombre del hijo a buscar, como se muestra en la Ilustración 17 para obtener los elementos del menú principal.



*Ilustración 17. Obtención de Game Objects hijos del canvas del menú principal*

#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que se inicializan los elementos y se pueden cambiar sus propiedades de ahí en adelante.

### **Observaciones del módulo**

La utilización de las funcionalidades de este módulo no tuvo ningún problema, sin embargo se aconseja que para la clase *“EthUtils”* concuerde el nombre del archivo con el de la clase en el código, ya que el archivo se llama “EthUtils” y la clase en el código se llame “Eth”.

## **Módulo Components**

Para este módulo no fue posible implementar la totalidad de las funcionalidades, como lo indica la Tabla 4 el componente *“progressBar”* no pudo ser implementado. A continuación se detalla el proceso de cada funcionalidad.

|  |  |
| --- | --- |
| **Funcionalidad** | **Implementación exitosa** |
| Soporte de idiomas a imagen | Si |
| Soporte de idiomas a texto | Si |
| Reproducción de audio a un toggle | Si |
| Soporte de idiomas a botón | Si |
| Reproducción de audio a un botón | Si |
| Modificar el valor de progreso a una barra de progreso | No |
| Animar la barra de progreso cuando se modifica su valor | No |
| Cargar una escena | Si |
| Al cargar una escena crear un panel sobre el canvas y mostrar un cargador | Si |
| Obtener el lenguaje del sistema | Si |

*Tabla 4. Funcionalidades implementadas del módulo util*

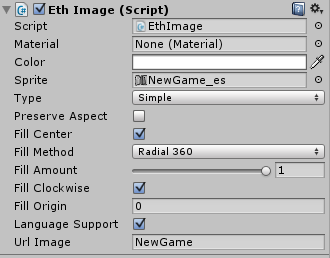
### **Soporte de idiomas a imagen**

Su implementación se llevó a cabo en diferentes secciones del videojuego, las cuales fueron:

* Imagen de fondo de los botones de nuevo juego y continuar del menú principal.
* Imagen de fondo del menú de configuración.
* Imagen de fondo de los botones del menú en juego.
* Imagen de fondo del menú de game over.
* Imagen de fondo de los botones del menú de game over.

Para su implementación en cada sección se realizaron los siguientes pasos:

* Se creó un Game Object vacío y se le agregaron los componentes *“Rect Transform”* y *“Canvas Renderer”.*
* Se le agregó el componente “Eth Image” y se configuró el componente dependiendo la necesidad del programador.
* Se agregaron dos texturas en la carpeta *“Resources”* del juego, el nombre de la que corresponde al idioma español termina en “\_es” y el nombre de la que corresponde al idioma ingles tiene el mismo nombre que la anterior pero su terminación es “\_en”.
* Al componente *“Eth Image”* se le activa la propiedad *“Language Support”* y en la propiedad *“Url Image”* se pone el nombre que corresponde a las dos imágenes sin la terminación “\_es” o “\_en”. En la Ilustración 18 se muestra la configuración de idiomas para el fondo del botón nuevo juego.



*Ilustración 18. Configuración de idiomas para la imagen de fondo del botón nuevo juego*

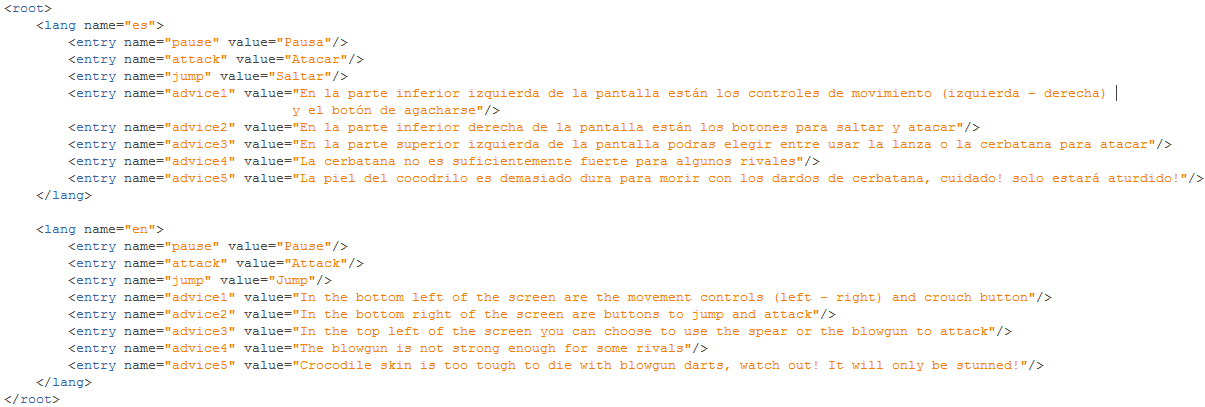
#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que la imagen de fondo cambia de acuerdo al idioma seleccionado.

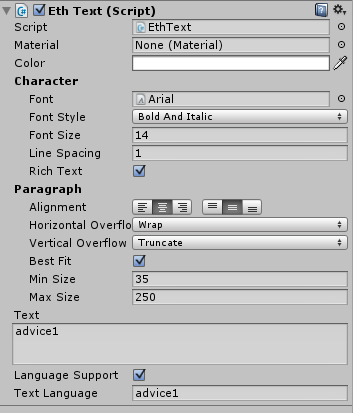
### **Soporte de idiomas a texto**

Su implementación se llevó a cabo en cada uno de los consejos que se dan al usuario en el nivel tutorial. Para su implementación se realizaron los siguientes pasos:

* Se creó un archivo XML que sirve como diccionario de datos para el soporte de idiomas, el cual debe estar en la carpeta *“Resources”* del juego y cuya estructura se muestra en la Ilustración 19.
* Se creó un Game Object vacío y se le agregaron los componentes *“Rect Transform”* y *“Canvas Renderer”.*
* Se le agregó el componente “Eth Text” y se configuró el componente dependiendo la necesidad del programador.
* Al componente *“Eth Text”* se le activó la propiedad *“Language Support”* y en la propiedad *“Text Language”* se puso el valor del texto alojado en el archivo XML como se muestra en la Ilustración 20.



*Ilustración 19. Estructura del archivo XML que sirve como diccionario de datos*



*Ilustración 20. Asignación de las propiedades “Language Support” y “Text Language”*

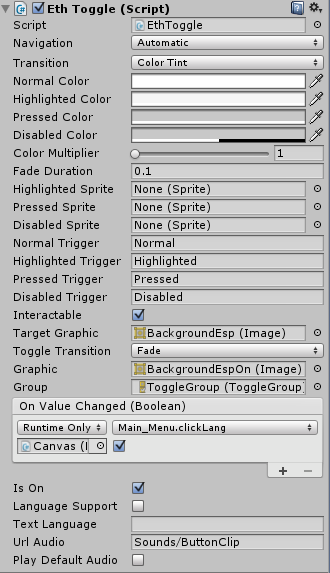
#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que el texto del consejo cambia de acuerdo al consejo a mostrar.

### **Reproducción de audio a un toggle**

Su implementación se llevó a cabo en el menú de configuraciones en la selección de idioma, donde al dar clic en alguno de los botones toggle se reproduce un efecto de sonido. Para su implementación se realizaron los siguientes pasos:

* Se creó un Game Object vacío y se le agregaron los componentes *“Rect Transform”* y *“Canvas Renderer”.*
* Se le agregó el componente “Eth Toggle” y se configuró el componente dependiendo la necesidad del programador.
* Al componente *en la propiedad “Url Audio”* se le ingresa la ubicación del efecto dentro de la carpeta *“Resources”* del juego como se muestra en la Ilustración 21.



*Ilustración 20. Asignación de las propiedades “Language Support” y “Text Language” del texto*

#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que al dar clic sobre el botón toggle se reproduce el efecto asignado.

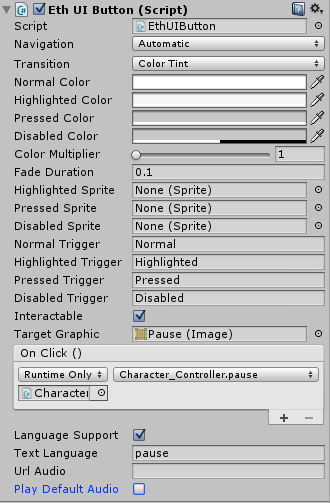
### **Soporte de idiomas a botón**

La implementación de esta funcionalidad se llevó a cabo en diferentes secciones del videojuego, las cuales fueron:

* Los botones de salto y ataque.
* El botón de pausa.

Para su implementación en cada sección se realizaron los siguientes pasos:

* Se creó un Game Object vacío y se le agregaron los componentes *“Rect Transform”* y *“Canvas Renderer”.*
* Se le agregó el componente “Eth UI Button” y se configuró el componente dependiendo la necesidad del programador.
* A este game object se le agregó un Game object hijo vacío y a este se le agregaron los componentes *“Rect Transform”*, *“Canvas Renderer”*  y *“Eth Text” y se le asignó el nombre “Text”.*
* En el botón (Primer Game Object creado), se activó la propiedad *“Language Support”* y en la propiedad *“Text Language”* se puso el valor del texto alojado en el archivo XML como se muestra en la Ilustración 21.



*Ilustración 21. Asignación de las propiedades “Language Support” y “Text Language” del botón*

#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que el texto del botón cambia de acuerdo al idioma seleccionado.

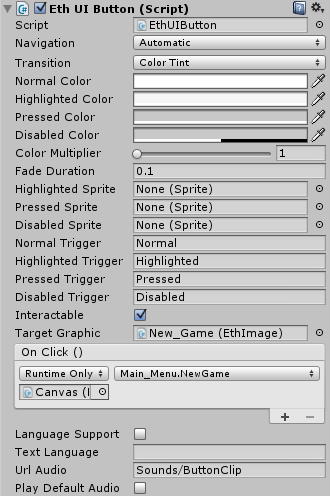
### **Reproducción de audio a un botón**

La implementación de esta funcionalidad se llevó a cabo en diferentes secciones del videojuego, las cuales fueron:

* Todos los botones del menú principal.
* Todos los botones del menú de configuraciones.
* Todos los botones del menú en juego.
* Todos los botones del menú de game over.

Para su implementación en cada sección se realizaron los siguientes pasos:

* Se creó un Game Object vacío y se le agregaron los componentes *“Rect Transform”* y *“Canvas Renderer”.*
* Se le agregó el componente “Eth UI Button” y se configuró el componente dependiendo la necesidad del programador.
* En el componente *“Eth UI Button”* a la propiedad *“Url Audio”* se le asigno la dirección del efecto a reproducir dentro de la carpeta *“Resources”* como se muestra en la Ilustración 22*.*



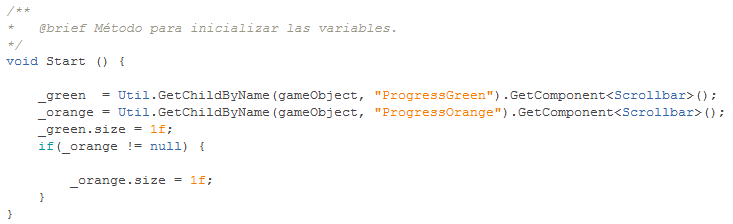
*Ilustración 22. Asignación de la propiedad “Url Audio” del botón*

#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que al hacer clic en el botón se reproduce el efecto especificado.

### **Modificar el valor de progreso de una barra de progreso**

Su implementación no se pudo llevar a cabo debido a que el script *“EthUIProgressbar”* en el método *“Start”* en las líneas 35 y 36, mostradas en la Ilustración 23, hacen uso de dos Game Objects propios del último juego donde Ethereal GF implementó el framework.



*Ilustración 23. Líneas del script “EthUIProgressbar” que son propias de otro juego*

#### **Conclusiones de la funcionalidad**

Al usar Game Objects propios de otro juego, este script se convierte en una solución adaptada y no una parte funcional del framework. Por ende se recomienda eliminar el script o corregir el código para dar soporte al componente progressbar.

### **Animar la barra de progreso cuando se modifica su valor**

Debido a que el script *“EthUIProgressbar”* utiliza Game Objects de otro juego y no se puede implementar el componente, esta funcionalidad tampoco se puede implementar.

#### **Conclusiones de la funcionalidad**

Esta funcionalidad es derivada del componente “EthUIScrollbar”, por ende se recomienda modificar el código fuente para arreglar su funcionalidad o eliminar el componente y todos sus derivados.

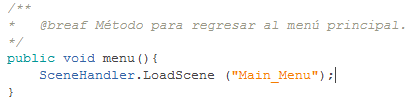
### **Cargar una escena**

La implementación de esta funcionalidad se llevó a cabo en diferentes secciones del videojuego, las cuales fueron:

* Para cargar el nivel tutorial al iniciar una nueva partida.
* Al superar un nivel, para cargar el siguiente.

Para su implementación en cada sección se realizaron los siguientes pasos:

* Se utilizó el método *“LoadScene”* de la clase *“SceneHandler”* donde se debe cargar una escena y se le asignó como parámetro el nombre de la escena a cargar como se muestra en la Ilustración 24.



*Ilustración 24. Utilización del método “LoadScene” de la clase “SceneHandler”*

#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que al cumplir con las condiciones establecidas se realiza el cambio de escena.

### **Al cargar una escena crear un panel sobre el canvas y mostrar un cargador**

La implementación de esta funcionalidad se llevó a cabo en la carga de las escenas más pesadas, las cuales fueron:

* Transición del menú principal al nivel tutorial.
* Transición del nivel tutorial al primer nivel.
* Transición del primer nivel al segundo nivel.
* Transición del segundo nivel al tercer nivel.

Para su implementación se llevaron a acabo los siguientes pasos:

* Se creó un prefab en la carpeta *“Resources”* del juego al cual se le asignó una imagen de fondo y una imagen con animación, cuyo nombre fue *“Loader”*.
* A los métodos *“LoadScene”* que correspondan a las transiciones mencionadas anteriormente se les agregó un segundo parámetro que corresponde al canvas de la pantalla, que es donde se mostrará el cargador, como se muestra en la Ilustración 25.



*Ilustración 25. Implementación del loader en la carga de escanas*

#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que mientras se carga la escena se muestra el cargador.

### **Obtener el lenguaje del sistema**

La implementación de esta funcionalidad se llevó a cabo al iniciar el menú principal del videojuego, donde se valida si la variable que contiene el lenguaje ya tiene uno asignado, de lo contrario se toma el idioma del sistema. Para su implementación se realizaron los siguientes pasos:

* En el método *“Start”* del script *“Main\_Menu”* se realizó la validación y en caso de cumplirla se utilizó el método “GetSystemLanguage” de la clase *“Util”* como se muestra en la Ilustración 26.



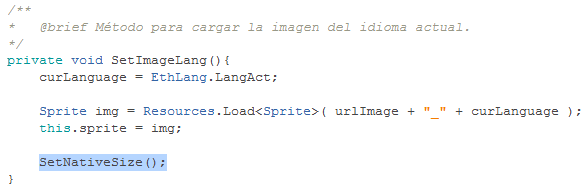
*Ilustración 26. Utilización del método “*GetSystemLanguage*” de la clase “Util”*

#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que al iniciar el juego se configura un idioma.

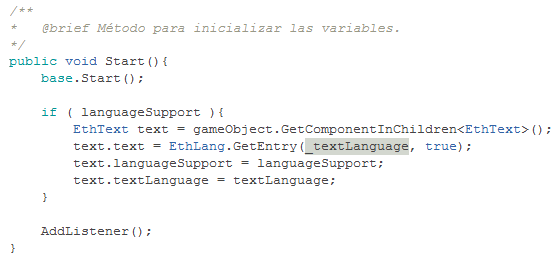
### **Observaciones del módulo**

Al configurar el soporte de idiomas a una imagen, esta se renderiza y no conserva el tamaño asignado al Game Object que la contiene, el equipo de trabajo encontró que la clase *“EthImage”* en el método *“SetImageLang”* en su última línea ejecuta el método *“etNativeSize”, el cual se muestra en la Ilustración 27,* lo cual generaba esta situación. Se recomienda modificar el método para evitar esta situación.



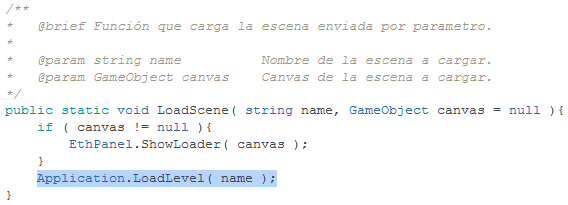
*Ilustración 27. Código que genera que la imagen con soporte de idiomas no conserve su tamaño*

Al configurar el soporte de idiomas de un componente *“EthUIButton”* el texto no cambia dependiendo del idioma, lo cual es debido a que en el método *“Start”* se utiliza la variable privada *“\_textLanguage”* para obtener los datos del diccionario, como se muestra en la Ilustración 28, cuando se debería usar la variable publica *“textLanguage”*.



*Ilustración 28. Variable que no permite el soporte de idiomas para el botón*

El método “LoadScene” del script “SceneHandler” hace uso de la instrucción “Application.LoadLevel” de unity, como se muestra en la Ilustración 29, la cual tiene una sobrecarga y puede recibir el nombre o el índice de la escena a cargar. Sería útil que el framework también tuviera dicha sobrecarga para permitir ingresar el índice de la escena.



*Ilustración 28. Utilización de la instrucción “Application.LoadLevel” de unity en el framework*

## **Módulo AppSystem**

La totalidad de las funcionalidades de este módulo fueron implementadas en el videojuego, como lo indica la Tabla 5. A continuación se detalla el proceso de implementación.

|  |  |
| --- | --- |
| **Funcionalidad** | **Implementación exitosa** |
| Inicializar el sistema de analíticas | Si |
| Registrar un evento en el sistema de analíticas | Si |
| Registrar una pantalla en el sistema de analíticas | Si |
| Obtener y modificar variables registradas en el sistema | Si |

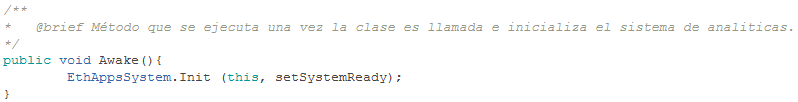
*Tabla 5. Funcionalidades implementadas del módulo appSystem*

Este módulo permite mediante una cuenta registrada en el sistema de Ethereal GF, al cual se accede mediante la URL <http://ethgame.com/WebFrontEndv2/dashboard.html>, registrar un videojuego al cual se le asigna un identificador que posteriormente es usado para iniciar el sistema en la aplicación se pueda tener acceso a información que permita el análisis de datos y toma de decisiones sobre la aplicación, por ejemplo si ningún usuario ha llegado a un nivel determinado analizar si hay un error que no permita el avance.

### **Inicializar el sistema de analíticas**

La implementación de esta funcionalidad se realizó de tal manera que cuando se ejecuta el juego, antes de mostrar el menú principal se inicie el sistema de analíticas, es por esto que antes de mostrar el menú principal se muestra un cargador. Para su implementación se realizaron los siguientes pasos:

* En el método “*Awake*” del script “*Main\_Menu*” se llamó el método “*Init*” de la clase “*EthAppSystem*”, como se muestra en la Ilustración 29, al cual se le ingresaron dos parámetros, el componente actual y un método que se ejecuta una vez se inicia el sistema de analiticas, el cual quita el cargador y muestra el menú principal.



*Ilustración 29. Llamado del método Init de la clase EthAppsSystem*

#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que al iniciar el juego se registra una nueva descarga o una nueva sesión en caso de ya estar registrada la descarga en la plataforma de analíticas de Ethereal GF.

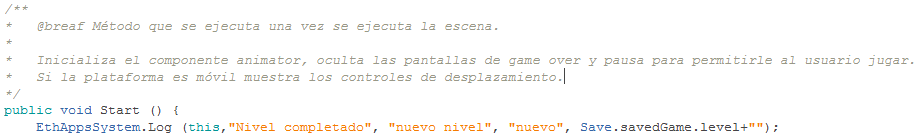
### **Registrar un evento en el sistema de analíticas**

La implementación de esta funcionalidad se realizó en diferentes secciones del juego con la finalidad de reportar diferentes eventos al sistema, las cuales fueron:

* Registrar el evento de iniciar un nuevo juego.
* Registrar cuando se abre el menú de configuraciones.
* Registrar cuando se inicia un nivel.
* Registrar cuando se completa un nivel.

Para su implementación en cada sección se realizaron los siguientes pasos:

* En la sección de código necesaria, se realizó el llamado al método “*Log*” de la clase “*EthAppsSystem*”, como se muestra en la ilustración 30, y se le ingresaron cuatro parámetros string que sirven para definir filtros y poder realizar un buen análisis de los eventos regsitrados.



*Ilustración 30. Llamado del método Log de la clase EthAppsSystem*

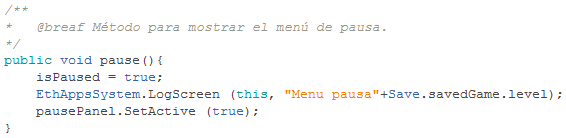
#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que al ejecutarse el método donde está incluida la instrucción “Log”, se registra el evento en la plataforma de analíticas de Ethereal GF.

### **Registrar una pantalla en el sistema de analíticas**

La implementación de esta funcionalidad se realizó de tal manera que cuando se abra el menú de configuraciones o de pausa en un nivel se registra en el sistema de analíticas. Para su implementación se realizaron los siguientes pasos:

* En la sección de código necesaria, se realizó el llamado al método “*LogScreen*” de la clase “*EthAppsSystem*”, como se muestra en la ilustración 31, y se le ingresa el nombre con el que se identificara esa pantalla.



*Ilustración 31. Llamado del método LogScreen de la clase EthAppsSystem*

#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que al ejecutarse el método donde está incluida la instrucción “LogScreen”, se registra la pantalla en la plataforma de analíticas de Ethereal GF.

### **Obtener y modificar variables registradas en el sistema de analíticas**

La implementación de esta funcionalidad se realizó de tal manera que cuando se inicia el juego, se obtiene la variable que indica el nivel actual del jugador y cuando se supera un nivel se modifica el valor de la variable. Para su implementación se realizaron los siguientes pasos:

* En el método “*Awake*” del script “*Main\_Menu*” se llamó al método “*GetVariable*” de la clase “*EthAppsSystem*”, como se muestra en la ilustración 32, y se le ingresa el nombre de la variable a obtener, dicho valor se le asignó al nivel actual del usuario.
* En la validación de cuando el jugador supera un nivel se llamó el método “*ChangeStateVariable*” de la clase “*EthAppsSystem*”, como se muestra en la ilustración 33, y se le ingresan el script actual, el nombre de la variable y el nuevo valor como parámetros.



*Ilustración 32. Llamado del método GetVariable de la clase EthAppsSystem*



*Ilustración 33. Llamado del método ChangeStateVariable de la clase EthAppsSystem*

#### **Conclusiones de la funcionalidad**

Al implementar esta funcionalidad se obtuvieron los resultados esperados ya que al obtener el valor de una variable este quedaba guardado en el juego, así mismo cuando en el juego se modifica una variable del sistema esta quedaba guardada en la plataforma de Ethereal GF.

### **Observaciones del módulo**

A pesar de la documentación de las clases de este módulo, la interpretación de las funcionalidades es compleja, por lo cual se requirió el asesoramiento de Andrés Herrera. Además en la plataforma solo se actualizan los datos cada 24 horas, lo que hace que no se pueda verificar al 100% si los registros se están realizando de forma correcta.