UNIVERSIDAD NACIONAL DE LOJA

ÁREA DE ENERGÍA Y RECURSOS RENOVABLES

CARRERA DE INGENIERÍA DE LA COMPUTACIÓN

TRABAJO FINAL DE ASIGNATURA

PROGRAMACIÓN ORIENTADA A OBJETOS

DOCENTE:

ING. ROBERTH FIGUEROA DÍAZ M.SC

ALUMNO:

ROY EMMANUEL LEÓN CASTILLO

PARALERO: 2 “A”

AGOSTO 2020

LOJA-ECUADOR

CREACIÓN DE VIDEOJUEGO EN LA PLATAFORMA DE DESARROLLO UNITY

Resume. – El presente documento contiene en lo que respecta al informe final del proyecto de programación que consistió en la creación de un videojuego en la plataforma de desarrollo **Unity,** con la finalidad de aprender el desarrollo de un videojuego.

El videojuego fue desarrollado en el lenguaje C# ya que Unity trabaja con tal lenguaje.

Previo a su desarrollo se realizó un diagrama de clases para tener una idea de cómo iría estructurado.

INTRODUCCIÓN

A medida que transcurre el tiempo son creados varios videojuegos y cada vez la forma en que los desarrollan va evolucionando.

La finalidad de hacer este proyecto fue entrar en el mundo de creación de videojuegos y ver o comprender como se implementa la programación orientada a objetos en ella.

OBJETIVO

- Crear un Videojuego que entretenga al usuario o por lo menos lo tenga pegado a la pantalla un rato.

ANALISÍS

¿Qué es Unity?

Es un motor de desarrollo o motor de juegos creada por la empresa Unity Technologies.

Lenguaje de programación utilizado.

C#. – Es un lenguaje de programación diseñado por la conocida compañía Microsoft. Fue estandarizado hace un tiempo por la ECMA e ISO dos de las organizaciones más importantes a la hora de crear estándares para los servicios o productos. El lenguaje de programación C# está orientada a objetos.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

Palabras claves:

- **Programación orientada a objetos**
- **Unity**
- **MonoBehaviour**

METODOLOGÍA

1. **Diagrama de clases**
   - **1.1. Escenas**

Enlace:
https://app.lucidchart.com/invitations/accept/b6d02115-547c-498b-bd68-7fd6ad55f90d

En este diagrama de clase se definió como estarán conectadas las escenas.

Para iniciar sección se creó un controlador para devuelva **true** si los datos ingresados con correctos, si los son nos envía a Menú Principal.

En Menú Principal podemos tener acceso a cualquier nivel dependiendo del progreso que tenga el usuario.

Dentro de cada escena se tiene una opción para poder pasar el juego.

Una vez acabado un nivel pasa al siguiente y si ya no existen más niveles regresa al Menú Principal.

- **1.2. Player**

Enlace:

https://app.lucidchart.com/invitations/accept/7fc96888-8ddb-454f-bb00-6d70f7eb932f

En este diagrama de clase se define las funciones que podrá hacer el jugador dentro del juego.

- **1.3. Objetivos**

Enlace:

https://app.lucidchart.com/invitations/accept/c0fa3efa-fb5a-4659-8b54-3f02a60517ca

En este diagrama se define los objetivos que habrá y sus funcionamientos.

Algo importante a tomar en cuenta es que a pesar de que los hijos tienen la misma función y variables se los deja en cada uno ya que a la hora entrar a un nivel el mensaje y donde se muestra cambiará dependiendo del hijo.

1.4. Objetos del Escenario

Enlace:

https://app.lucidchart.com/invitations/accept/7621d05f-0ef1-4f6a-804f-c25c577970a1

En este diagrama definimos las funciones que tendrán unos objetos dentro de las escenas.

Algo interesante hacer notar es que en C# un método de una variable abstracta que va a ser heredado se le pone el antecesor virtual (se utiliza para modificar un método) y la clase la cual hereda se le pone el antecesor override.

2. Código

Dentro de los códigos existen variables tipo GameObject, TMP_Text, entre otros, que son propios de Unity los cuales nos permite llamar a objetos que aparecen en el juego.

Todo Scripts creado hereda MonoBehaviour el cual nos da la opción de llamar métodos propios de clase, tal como Update(Aquí va lo que se ejecuta mientras el juego corre), Start(Aquí se ejecuta todo lo que debe ejecutarse cuando inicia la escena), OnTriggerEnter(Cuando un objeto toca otro objeto), se la puede quitar, pero dejarías de tener ese beneficio, así que primero hay que ver si nuestra clase necesita de esos métodos.

Las variables TMP_InputField sirve para que el usuario pueda ingresar las palabras que se necesita, y TMP_Text para mostrar un mensaje, El botón salir nos permitirá cerrar el juego.

También hay clase tipo Vector3, que nos permite acceder a todas las cosas que Unity permite hacer con un objeto 3D, este tipo de clases nos permite mover dentro de la escena.

Se creo un código que nos sirve para pedir el nombre de Usuario y contraseña, así como uno que sirve para comprobar si los datos que ingreso el usuario son correctos y así permitirle o no el ingreso.

Para el menú principal se creo un código el cual nos permite ingresar al juego y/o a cualquiera de sus niveles

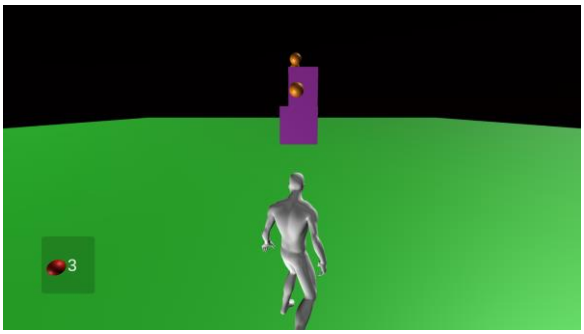


Imagen de la Escena 1.

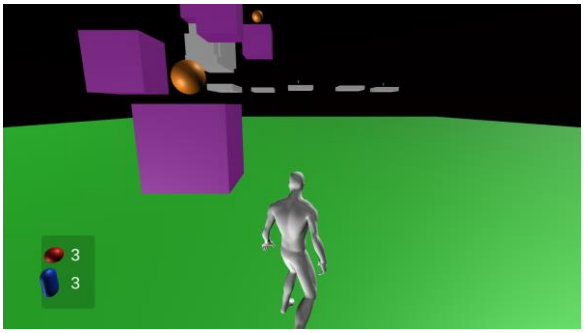


Imagen de la Escena 2.

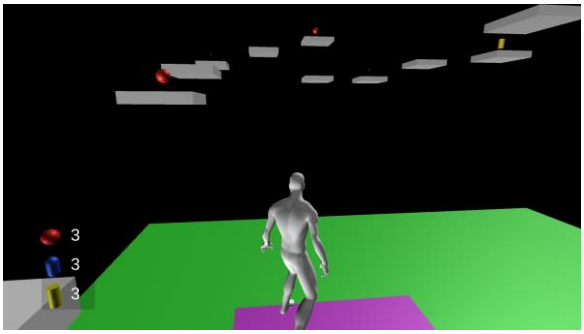


Imagen de la Escena 3.

CREACIÓN DE VIDEOJUEGO EN LA PLATAFORMA DE DESARROLLO UNITY

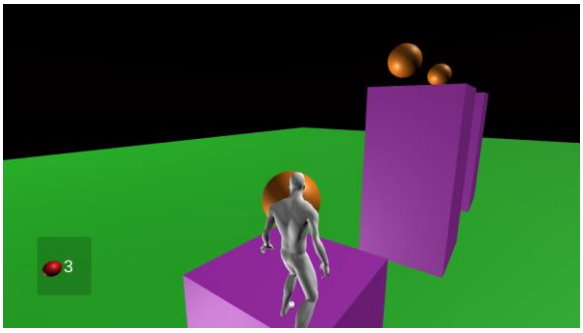
Los niveles se los guarda en un archivo txt dentro de cada uno de ellos para ser leídos en menú principal y así tener accesos a ellos.

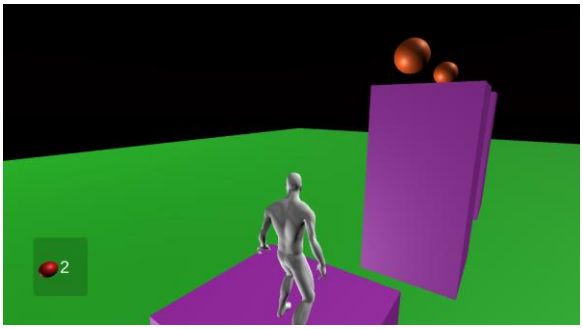
En los scripts de cada escena se llama a la clase encargada de los objetos de la escena para que realicen sus funciones.

Crea una clase para cada objetivo que se ponga y se llama a su función getCorrecto para saber si ya no existen más objetivos y poder realizar más funciones.

Con un script para pausa cambiamos el tiempo en que corre el juego para así detenerlo

Para los objetivos lo que hacemos es contar los que tengan en el mismo nombre, mostrar el número y cuando el personaje choca con uno de ellos actualizar el número





(como vemos en la esquina inferior izquierda el número se actualizo al chocar con el objetivo), cuando llega a cero devuelve el valor **true** que será ocupado por el código correspondiente de la escena.

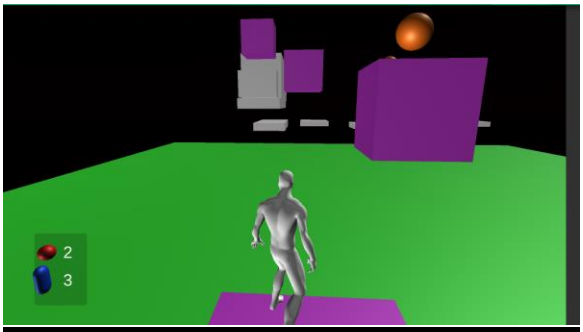
Cuando los objetivos llegan a cero el código llama a uno de los tres hijos (el que corresponda al objetivo, son: Esfera (Objetivo Rojo), Capsula (Objetivo Azul) y Cilindro (Objetivo Amarillo)) y muestra un mensaje en panta de que el objetivo N ha sido completado.

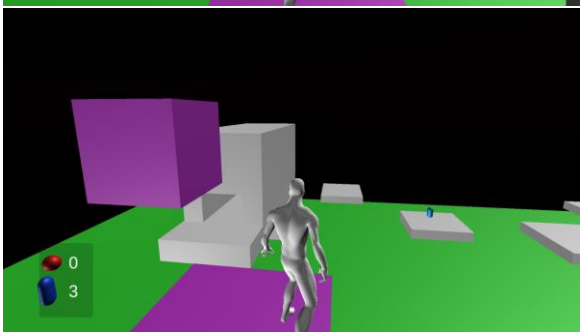
Existe un código abstracto para dar función a unos objetos dentro de la escena, tiene 2 hijos.

El primero:

Destruye un objeto.

CREACIÓN DE VIDEOJUEGO EN LA PLATAFORMA DE DESARROLLO UNITY

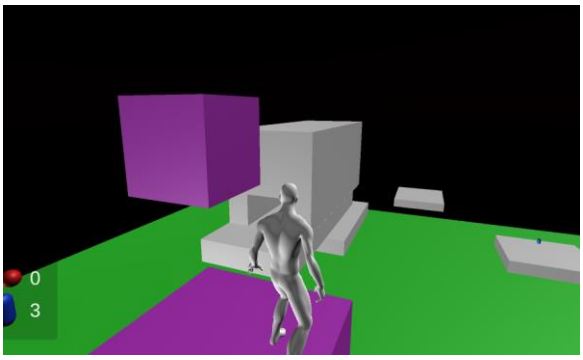


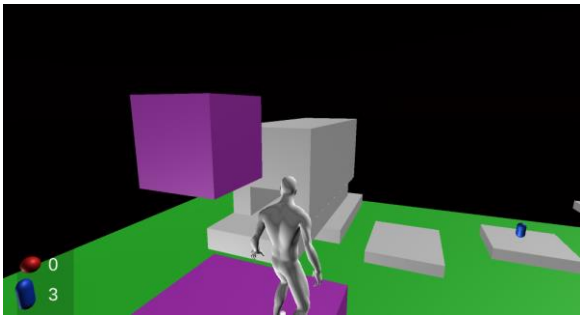


Como podemos observar el cubo que tapaba ese camino se destruyó (Por el contenido del código). En este caso el código de la escena lo condiciona cuando los objetivos rojos.

El segundo:

Permite que una plataforma se dirija de un lado a otro.

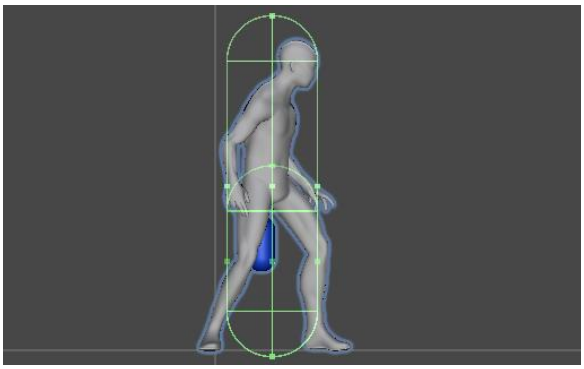




Para las funciones del personaje se tiene lo siguiente:

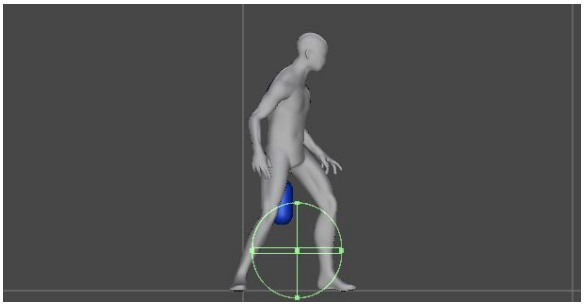
La interfaz mover.

El código que permite al personaje moverse, saltar y agacharse junto a las animaciones correspondientes, esto se puede hacer ya que se tiene definido la velocidad de movimiento, fuerza de salto y los límites del cuerpo gracias a las variables `Rigidbody, CapsuleCollider y GameObject.`

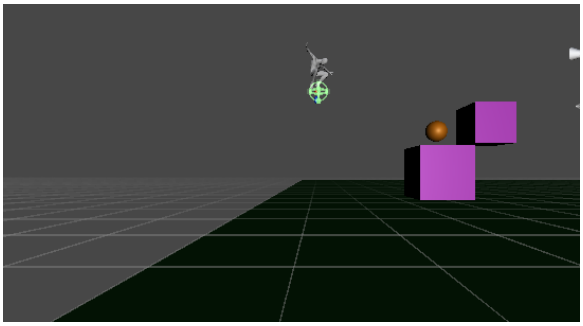


El código para saltar lo que hace es que cuando el personaje está en un lugar firme pueda saltar y sí está en el aire no.

Esto se puede gracias al Capsule Collider de sus pies que detenta cuando está en el piso.



CREACIÓN DE VIDEOJUEGO EN LA PLATAFORMA DE DESARROLLO UNITY



Este código permite saber si tenemos algo arriba de la cabeza, haciendo que cuando te agachas y hay un objeto arriba no te puedas levantar, aunque dejes de aplastar el botón (Dando en el contador 0 si no hay algo arriba y 1 si lo hay).

RESUMEN

En resumen, primero se definió como va a estar conformado el juego y sobre qué, después se creó un diagrama de clases para cada uno de ello.

En segundo se consulto en la API y videos todo lo que sirva para la creación de este.

El juego en sí es simple de entender el jugador debe recolectar los objetivos que se le pide, pero si cae tendrá que recorrer todo el camino, se debía añadir un poco de dificultad.

En el jugador se le implementó dentro del juego el script de las escenas puesto que él va a estar en todo momento dentro de ella.

CONCLUSIONES

- Concluyo que la lógica de programación orientada a objetos es muy útil ya que nos permite tener un código más ordenado y comprensible.
- Concluyo que Unity es un buen programa para iniciar en el desarrollo de videojuegos ya que encuentras una gran variedad de documentos y videos que sirven de apoyo.
- Concluyo que el juego puede llegar a ser entretenido aunque pueda alterar la paciencia del jugador.
- Concluyo que la investigación es muy importante en el proyecto puesto que te puedes encontrar varias soluciones para un mismo problema, pero eso no quiere decir que todos te sirvan.

RECOMENDACIONES

- Recomiendo que si inician en el ámbito de desarrollo de juego ocupen Unity.
- Recomiendo que primero lean, por lo menos lo básico, de la documentación que tiene Unity y ver sus videos.
- Recomiendo que sean constantes, ahí es cuando el código comenzará a ser tuyo y se te facilitará las cosas.

BIBLIOGRAFÍA

1. **¿Qué es Unity?, Dispone en:** https://openwebinars.net/blog/que-es-unity/
2. **Acerca de C#, Dispone en:** https://negociosyestrategia.com/blog/que-es-csharp/