**CENTRO DE ENSEÑANZA TECNICA INDUSTRIAL**

VIDEOJUEGO DE CARRERAS PARA EL MANEJO DEL ESTRÉS EN ESTUDIANTES

(APLICACIÓN)

SUSTENTANTES:

EDGAR EDUARDO SANDOVAL AVALOS

ADRIAN SANCHEZ GONZALEZ

CARRERA:

INFORMATICA Y COMPUTACION

ASESOR:

MTRO. JOSE GUSTAVO ROJAS GARCIA (Asesor técnico)

MTRO. JUAN OLMEDA GONZÁLEZ (Asesor metodológico)

GUADALAJARA, JAL 30 DE JUNIO DE 2016

# 

# Resumen

## **I.1 Definición del problema**

En ocasiones los alumnos pueden presentar altos niveles de estrés generados por el gran número de actividades escolares que estos realizan.

Según algunos médicos del estrés de derivan enfermedades coronarias, problemas digestivos y trastornos menstruales en el caso de las mujeres.

El estrés escolar y sus molestias derivadas pueden llegar a afectar negativamente el rendimiento escolar del alumno.

## **I.2 Hipótesis**

Si se realiza una o varias actividades que alegren, distraigan o entretengan al alumno como un videojuego después del periodo de evaluación, fecha límite de sus actividades escolares etc. entonces disminuirán sus niveles de estrés por lo tanto el alumno estará saludable y dispuesto para seguir cumpliendo con sus actividades escolares.

## **I.3 Objetivos y alcances del proyecto**

Se pretende ayudar a los alumnos a asimilar el estrés mediante un videojuego en su computadora.

* Implementar la jugabilidad del género “arcade carreras” en el videojuego para que el usuario pueda disfrutar de una experiencia de carreras basándose en juegos existentes del mismo género.

## **I.4 Justificación**

Dados los avances tecnológicos y los distintos medios de entretenimiento usados actualmente ya sea cine, teatro, deportes, juegos, baile social, conciertos, espectáculos de comedia, animaciones, impresionistas, payasos, etc…, un videojuego es un gran medio de acceso a un público numeroso de usuarios, de esta manera es posible influir de cierta forma en la percepción que tienen los usuarios sobre algún tema por medio del entretenimiento.

Los videojuegos se han establecido actualmente como una excelente opción para el ocio y representan un medio alterno de entretenimiento.

Se propone un videojuego de carreras, por su fácil interacción con el jugador, permitiendo así que libere su estrés o su tensión acumulada o simplemente para entretener a quien lo usa.

Según el periódico “El informador” concluye en un estudio que los videojuegos mejoran los reflejos, impulsos, emociones, también logran tiempos de reacción más rápidos, así como decisiones más veloces y certeras sin tener en cuenta los conocimientos y experiencias previas de los jugadores. Así mismo consideran que los jugadores son constantes solucionadores de problemas que no dejan de buscar soluciones más fáciles y mejores a determinados desafíos y obstáculos.

Cada juego se diseña para que cualquiera tenga éxito, independientemente de sus destrezas y objetivos, pero así mismo para desafiar las capacidades de cada jugador sin llegar a sobrepasarlas.

## **I.5 Descripción del producto**

Será un videojuego de carreras “CETI Furious Turbo Buggy” presenta una versión de un videojuego de carreras. Este es un videojuego donde el objetivo es tener una forma fácil de jugar además de fácil interacción con el usuario de poder simular una carrera.

El videojuego tiene como meta entretener al jugador con carreras en pequeños automóviles de carreras mejor conocidos como “Karts”.

En el videojuego el jugador correrá en distintos mapas todos dentro del Centro de Enseñanza Técnica Industrial: Plantel Colomos. En las pistas, el jugador deberá seguir un camino indicando el inicio y el fin de la carrera esquivando obstáculos que los adversarios pueden dejar a su paso. Para esta versión habrá un mínimo de tres pistas.

Para el modo de un jugador, el jugador correrá las 3 diferentes pistas y en cada una se enfrentará a un enemigo, que al ganar la carrera este enemigo estará disponible para que el usuario pueda usar este personaje.

Para el modo multjugador, el jugador se unirá o dará alojo a una sala en al cual podrán unirse distintos usuarios que se encuentren en la red local y cuenten con su versión del videojuego, una vez iniciada la partida por el anfitrión los jugadores actuales en la sala pasaran a escoger un personaje y posteriormente se elegirá la pista en la cual se llevará a cabo la carrera.

## **I.6 Lista de requerimientos funcionales**

RF1.- El juego guarda el avance del usuario en un archivo.

RF2.- El juego carga la partida si es que existe.

RF3.- Controles de juego con el teclado.

RF4.- El avance del juego únicamente se guarda al final de una carrera.

RF5.- El multijugador solo se puede utilizar por medio de una red LAN

RF6.- El usuario no puede modificar texturas y/o calidad grafica

RF7.- La cámara debe seguir siempre al jugador

RF8.- Debe de haber cajas que contengan objetos para dar ventaja en la carrera

RF9.- La carrera debe terminar cuando un jugador cruce la meta 3 veces

## **I.7 Lista de requerimientos no funcionales**

RNF1 Al crear una nueva partida, la computadora genera un archivo codificado con el progreso del jugador.

RNF2 La pausa se activa con la tecla Esc.

RNF3 El usuario no puede abandonar de una carrea multijugador.

RNF4 Las cajas en que contiene el mapa deben dejar de existir cuando no se usan, de modo que al utilizarse debe de volver a crearse.

RNF5 Al únicamente al término de la carrera se puede guardar la partida.

# IV Desarrollo del proyecto

En esta sección se explican las características técnicas del programa además de los avances y cambios que van surgiendo conforme el programa termina de ser desarrollado.

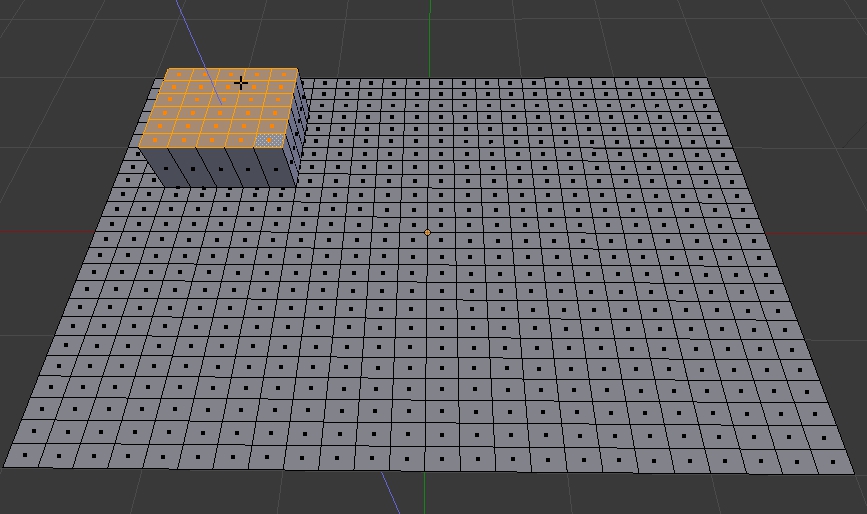
# IV.1 Diseños y Modelos

El diseño y el modelado forma parte fundamental en un videojuego ya que finalmente es lo que se le muestra al jugador en primera instancia a partir de ahí puede influir en la decisión de jugar o no el juego.

### **IV.1.1 Diseño del mapa**

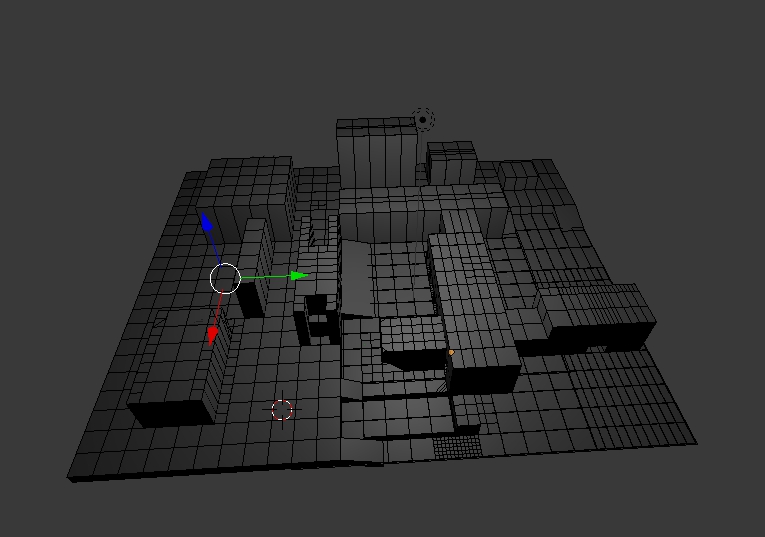
Para el modelado del mapa y de los edificios se optará por tomar medidas aproximadas para su construcción; puesto que se había considerado tomar fotografías a todo el plantel, pero este proceso tomaría demasiado tiempo.

Los edificios se modelarán a partir de un plano base subdividido para facilitar el trabajo de diseño tal y como se aprecia en la figura 4.1.



*Figura 4.1 Plano subdivido*

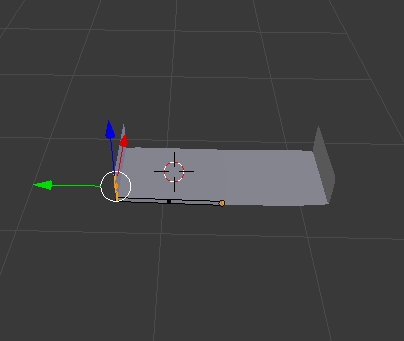
El proceso explicado en la parte superior se utilizará para todos los edificios e instalaciones para crear un modelo único que los contenga a todos, dando el resultado mostrado en la figura 4.2.



*Figura 4.2 Resultado final del mapa*

### **IV.1.2 Modelado de pistas**

Para el diseño de las pistas únicamente se creará un fragmento, el cual se muestra en la figura 4.3.



*Figura 4.3 Fragmento de pista*

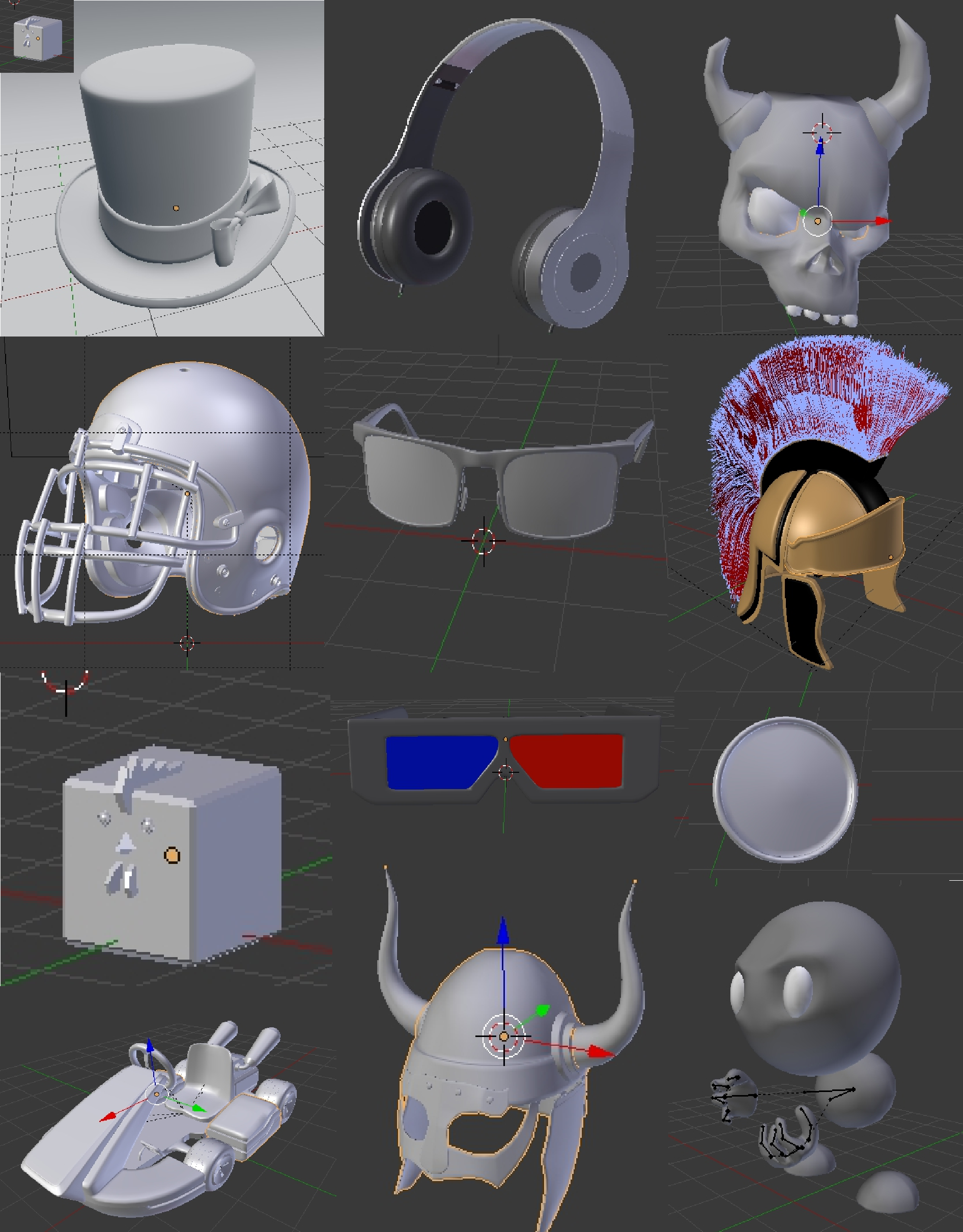
Una vez creado el fragmento, se duplicará varias veces a lo largo de una curva con el diseño del circuito para darle la forma deseada y se utilizará el mismo proceso para el diseño de las demás pistas tal y como se muestra en la figura 4.4.



*Figura 4.4 Curvas con pista adaptada*

### **IV.1.3 Modelado personajes**

Para el personaje, coche y otros objetos dentro del juego, se optó por buscar los diseños en internet y facilitar un poco la tarea de modificación o adaptación de estos al juego. Los distintos modelos utilizados son diseños gratuitos descargados desde la página [www.blendswap.com](http://www.blendswap.com).



*Figura 4.5 Modelos*

En la figura 4.5 se aprecian los objetos descargados para los personajes, para los objetos usados en las armas vease la figura 4.

# Capítulo IV.2: Vistas y menús

### **IV.2.1 Menú principal**

El menú principal de opciones y de ajuste gráfico se diseñará aplicando el uso de “canvas” proporcionados por el propio editor de Unity, véase figura 4.6.

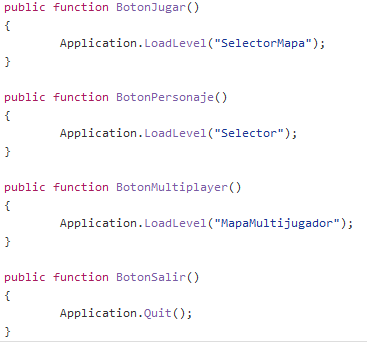


*Figura 4.6 Menú principal*

Cada uno de los botones manejará funciones similares siendo presionados, las cuales están definidas por Unity para activar o desactivar un objeto cunado se da clic sobre éste.

Código Menú Principal

Las funciones de los botones que cargan las escenas del juego mandan llamar directamente a la escena por su nombre, tal y como se aprecia a continuación.

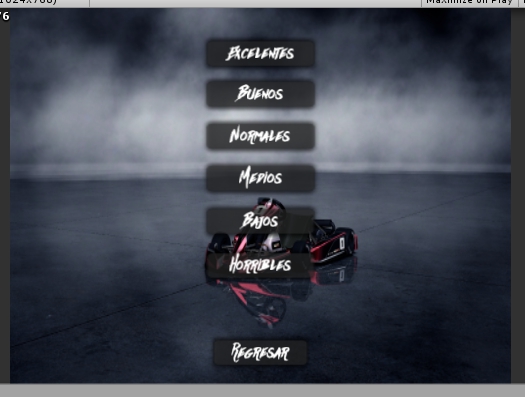


*Figura 4.7 Código para cargar las escenas*

En la figura 4.7 se realizan las funciones que cargan el selector de mapa, el selector de personaje, el menú multijugador y de salida del juego.

### **IV.2.2 Menú de gráficos**

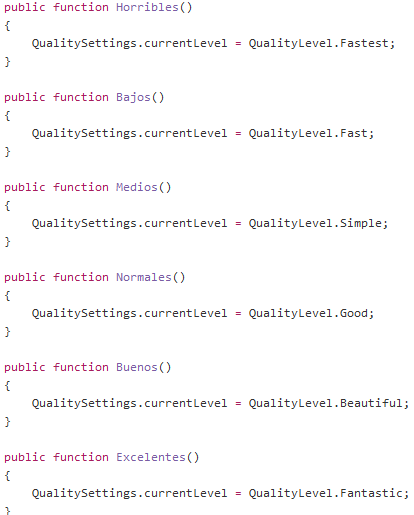
Como se aprecia en la figura 4.8, se pueden ajustar los gráficos desde el mismo juego.



*Figura 4.8 Menú Ajustes*

Para que se realice la acción mencionada en la parte superior, cada botón utiliza un código como el que se muestra a continuación.

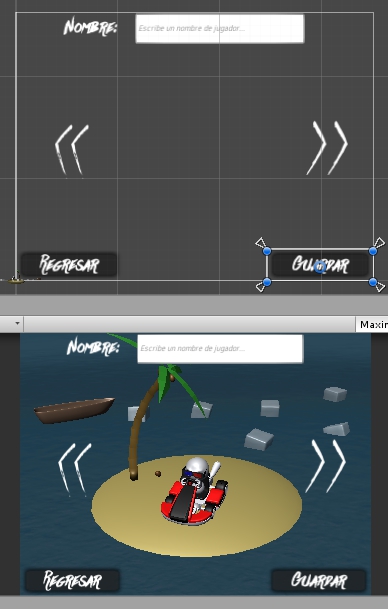
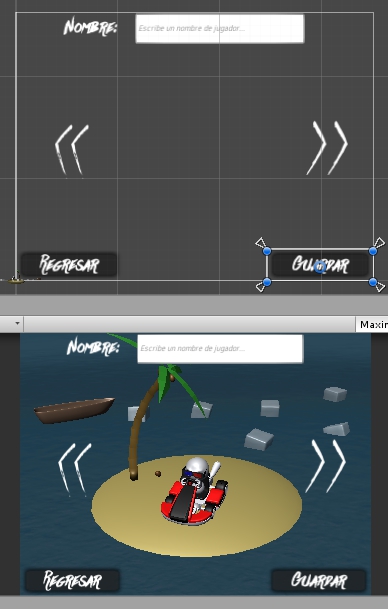
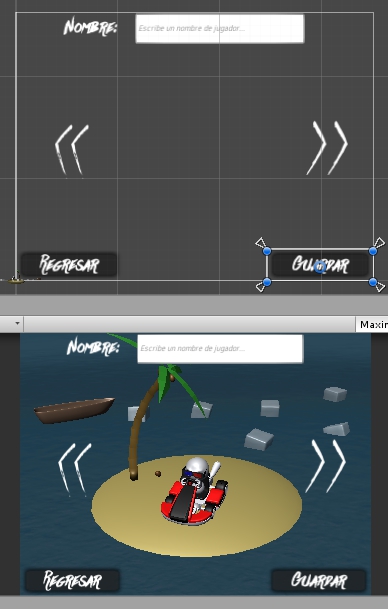
Código Menú Ajustes



*Figura 4.9 Código Menú Ajustes*

En la figura 4.9 se pueden apreciar las funciones que ajustan el valor de los gráficos en uno, ya predefinido por Unity. De manera similar las demás funciones ajustan a distintos valores ya definidos para lograr otro efecto en los gráficos del juego.

### **IV.2.3 Selector de personaje**

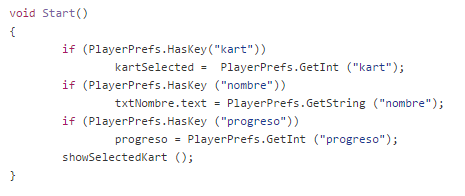


*Figura 4.10 Selector de Personaje*

El selector de personaje se creará agregando canvas a un terreno en 3D, véase figura 2.3

Código Selector Personaje

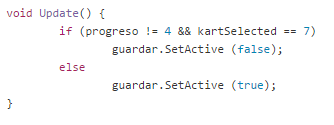
Al cargar la escena se realiza una inicialización de campos referentes a las preferencias del jugador en este caso el personaje seleccionado y su nombre, éstos son requeridos para futuras operaciones como por ejemplo mostrar los personajes disponibles (ver figura 4.13), el código mostrado en la parte inferior realiza esa función



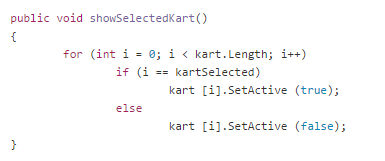
*Figura 4.11 Start Selector Personaje*

La figura 4.11 muestra la inicialización del selector, este procedimiento se llama por defecto cada que se inicia la escena del selector.

Posteriormente mientras se navega por los personajes disponibles se realiza una validación para que el jugador no pueda seleccionar el último personaje sin antes haber finalizado la pista final el siguiente código realiza esa función.



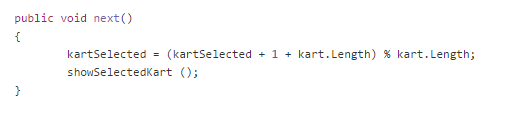
*Figura 4.12 Update Selector Personaje*



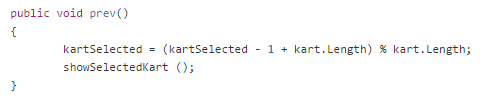
*Figura 4.13 Función para mostrar personajes*

El procedimiento de la figura 4.12 revisa que el progreso del jugador se cumpla para poder mostrar el botón de selección en el último personaje.

El procedimiento de la figura 4.13 cambia entre los distintos personajes definidos en un arreglo que los contiene a todos, la variable kartSelected contiene el número del personaje seleccionado actualmente.



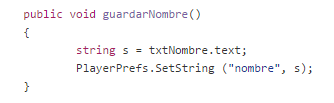
*Figura 4.14 Función next en Selector Personaje*



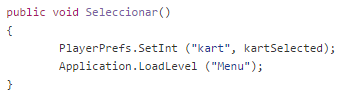
*Figura 4.15 Función prev en Selector Personaje*

Los procedimientos “*next*” y “*prev*” (ver figura 4.14 y 4.15) se encuentran asociadas a un botón en la escena respectivamente, estás funciones calculan el siguiente índice del personaje o el anterior dependiendo el caso y muestran nuevamente el personaje seleccionado con el procedimiento “Update”. Ver figura 4.12.

Finalmente para guardar los cambios se utiliza una caja de texto ubicada en la parte superior de la pantalla y el botón ubicado en la parte inferior derecha de la pantalla para guardar el nombre de jugador que se utilizará en el modo multijugador y el personaje seleccionado (ver figura 4.10), finalmente al guardar los cambios el jugador es redirigido a la pantalla de menú principal el código es mostrado a continuación



*Figura 4.16 Función guardar nombre en Selector Personaje*



*Figura 4.17 Función Seleccionar en Selector Personaje*

En las figuras 4.16 y 4.17 se puede ver el código usado para guardar los datos del usuario como el nombre y el kart que seleccionó, estos se guardan con un índice para su posterior lectura.

### **IV.2.4 Selector de Mapa**

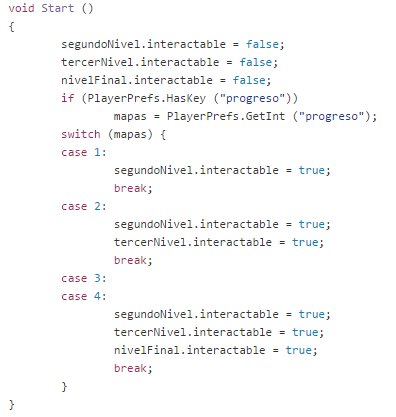


*Figura 4.18 Seleccionar Mapa*

En la imagen anterior se observa el selector de mapas o de niveles (figura 4.18), este permite al usuario entrar a un determinado nivel a partir de su progreso en el juego.

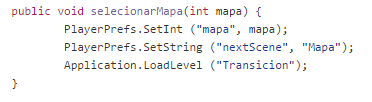
Código Selector Mapa

Al cargar la escena se realiza una validación para conocer a que niveles tiene acceso, una vez obtenido el dato se desactivan todos los botones que no pertenezcan en el conjunto de niveles superados



*Figura 4.19 Método Start Seleccionar Mapa*

En la figura 4.19 se aprecia el código del selector, el procedimiento “Start” se ejecuta en cuanto la escena se carga por lo tanto en este definimos que por default el segundo, tercer y último nivel estarán bloqueados, en caso de encontrar un progreso de jugador guardado con PlayerPrefs entonces este activara los siguientes mapas.



*Figura 4.20 Método seleccionar Mapa*

El código anterior (Figura4.20) pertenece al procedimiento seleccionar mapa, éste funciona recibiendo el número de pista o nivel del botón presionado y guarda en qué nivel iniciara el jugador cuando se cargue la pista.

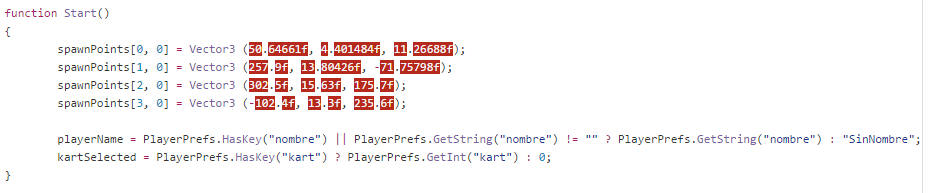
### **4.2.5 Multijugador**



*Figura 4.21 Multijugador*

En esta escena se requiere dinamismo en los componentes en cuanto a poder aparecer y desaparecer botones y cajas de texto dependiendo la situación, para esta escena los scripts no son dibujados en un canvas desde el inicio estos son generados al momento de cargar la escena. Véase figura 4.21.

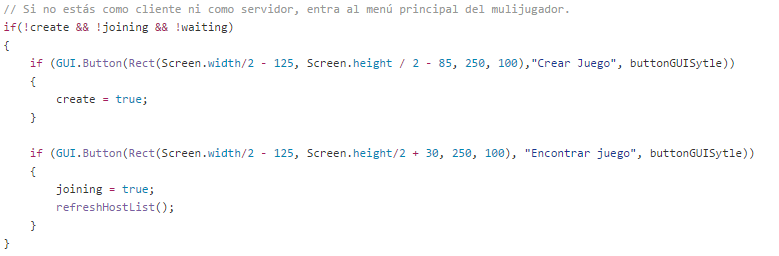
Código Multijugador



*Figura 4.22 Start del Multijugador*

En el código anterior se inicializan los vectores de aparición de los jugadores y en el orden en el que aparecerán además se accede al nombre guardado en caso de existir y el personaje seleccionado también en caso de existir.

La primera pantalla que se muestra, contiene solamente 2 botones para crear partida o unirse a la partida, a continuación, el código.



*Figura 4.22 Start del Multijugador*

La sección de código anterior se ejecuta en el método OnGUI de Unity, crea y asigna una especie de condición “OnClick” para la ejecución de un código en caso de que el botón sea presionado, en caso de crear partida cambia la bandera créate de lo contrario cambia la bandera joining y busca los host en la red.

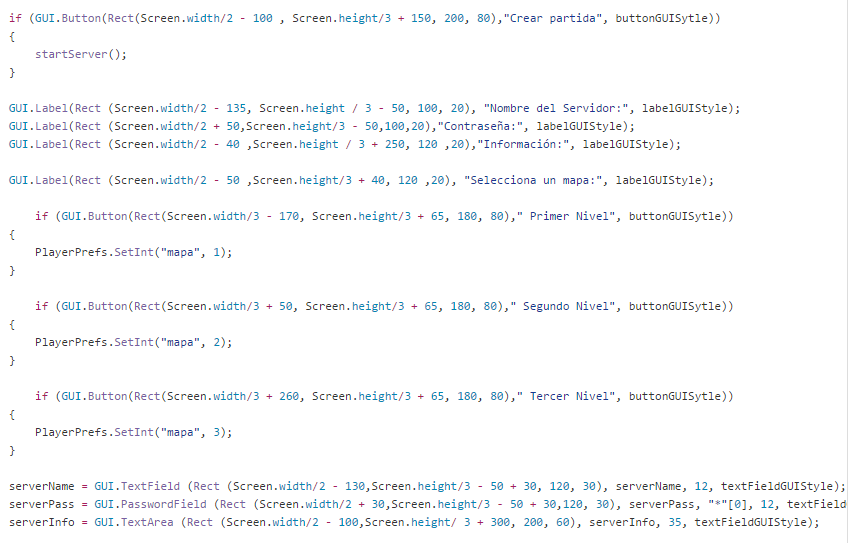
En caso de ser creador de partida se mostrará la siguiente pantalla.



*Figura 4.23 Crear partida de multijugador*

En la figura anterior se muestra como luce la pantalla para crear partidas, está cuenta con dos botones a las esquinas superiores los cuales te permiten regresar al menú principal o volver a la pantalla anterior.

También muestra dos cajas te texto para colocar el nombre de la partida y la contraseña de la partida respectivamente, ambos opcionales en caso de que no se cuente con ese valor se colocaran los valores por defecto por la función MasterServer.RegisterHost(), ver capitulo IV.4



*Figura 4.24 Código Crear partida de multijugador*

La sección de código anterior también es ejecutada en el método OnGUI y muestra como es que se construye la pantalla de crear partida ver figura 4.23.



*Figura 4.25 Unirse a partida de multijugador*

En la figura anterior se muestra como luce la pantalla para unirse a las partidas, está cuenta con dos botones a las esquinas superiores los cuales te permiten regresar al menú principal o volver a la pantalla anterior.

En caso de que no se haya encontrado ninguna partida mostrará un mensaje como “Ninguna partida encontrada” además de un botón para intentar refrescar la lista de posibles servidores.

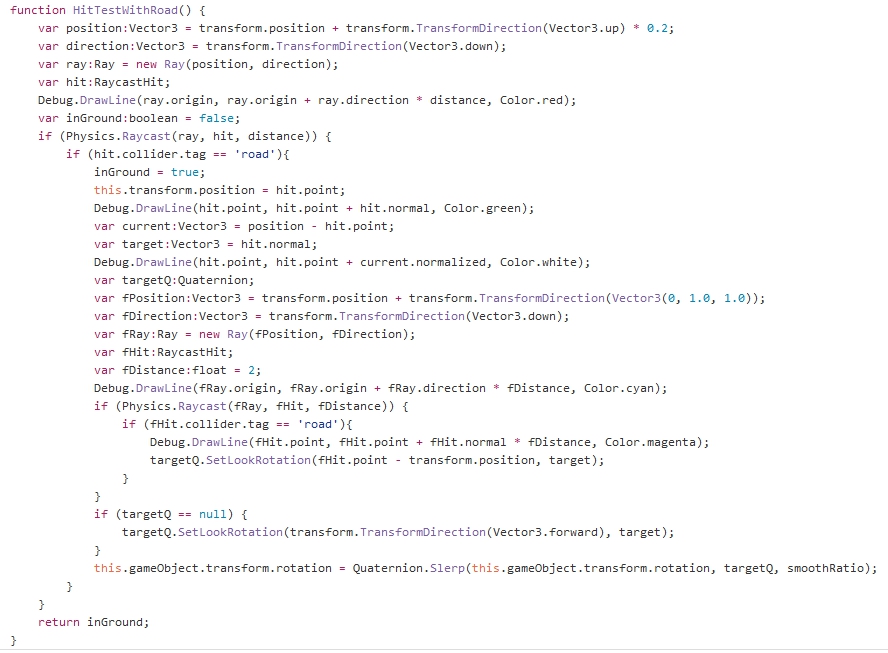
También muestra dos cajas te texto para colocar el nombre de la partida y la contraseña de la partida respectivamente, ambos opcionales en caso de que no se cuente con ese valor se colocaran los valores por defecto por la función MasterServer.RegisterHost(), ver capitulo IV.4

# Capítulo IV.3: Modo un jugador

Como se mencionó anteriormente el videojuego posee una modalidad para un jugador.

### **IV.3.1 Programación movimiento del carro**

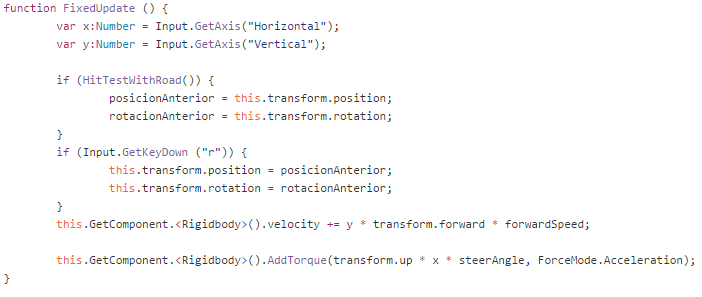
El movimiento del carro se basa en que el objeto que se maneja se adhiera automáticamente a la malla más cercana cuya etiqueta sea la correcta.



*Figura 4. Funcion para adherir el carro al camino*

En el código anterior (figura 4.) se puede ver la función principal del movimiento, esta busca un objeto con el cual colisionar que use la etiqueta “road” y respectivamente obtiene la posición del objeto en el cual se colocó el script para hacer que se acerque y se mantenga en la pista.

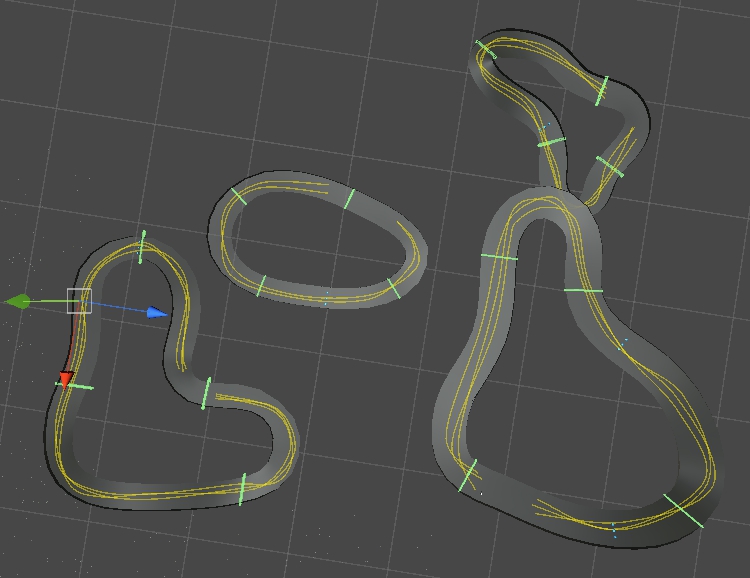
Para obtener los movimientos del jugador se toman las teclas definidas como “Vertical” y “Horizontal”, estas ajustan la velocidad y la torsión del carro, tal y como se muestra en el siguiente código.



*Figura 4. Función FixedUpdate Movimiento un jugador*

El código anterior (figura 4.) muestra que para avanzar se usa la variable “y” multiplicada por otras variables propias de los objetos que contienen el componente Rigidbody, estas representan el ángulo de avance y la velocidad de este, así mismo para obtener el giro del carro se multiplica la variable “x” por una variable que representa el ángulo de giro y otra de aceleración.

### **IV.3.1 Programación puntos de control**

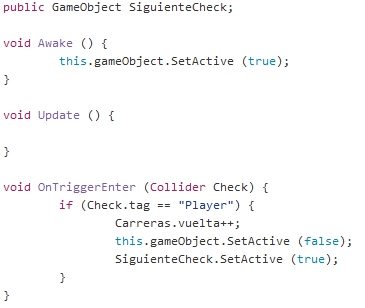


*Figura 4. Puntos de control de las 4 pistas*

Los checkpoints o puntos de control son programados por pista y siguiendo una secuencia de activación en el sentido de la pista, en la figura 4. podemos observar todos los puntos de control utilizados en cada una de las pistas. Cada checkpoint funciona como trigger, al cruzar un objeto este se activa ejecutando una acción.

Código Checkpoints

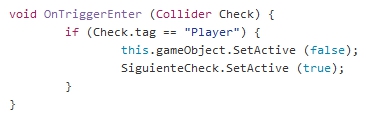
El primer punto de control es la meta, este sirve como punto de inicio/fin para el jugador, su principal utilidad es activar el siguiente checkpoint y contar las vueltas del jugador.



*Figura 4. Codigo meta*

En la figura 4. Se aprecia el código de la meta del usuario, este maneja un objeto de referencia que es el siguiente punto de control, el procedimiento “Awake” se llama justo al cargar la escena, este activa la meta por defecto como el primer punto de control a cruzar, el procedimiento “OnTirggerEnter” define la acción que se hara cuando un objeto definido como trigger sea atravesado por otro, en este caso el procedimiento verifica que el objeto que atravesó la meta sea el jugador, incrementa las vueltas que lleva el jugador, activa el siguiente checkpoint y desactiva la meta para obligar al jugador a atravesar el siguiente checkpoint.

El checpoint es un objeto distinto a la meta cuya función es solo activar el siguiente checkpoint para lograr así un control de activación en la pista.

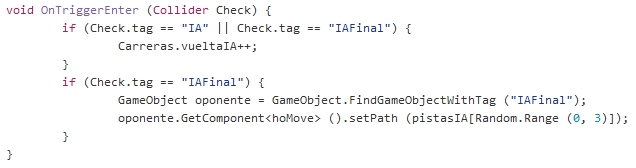


*Figura 4. Codigo checkpoint*

De manera similar a la meta el checkpoint cuenta con el procedimiento

OnTriggerEnter (vease figura 4.) que cuando es llamado activa el siguiente checkpoint de la variable “SiguienteCheck” y se desactiva a si mismo.

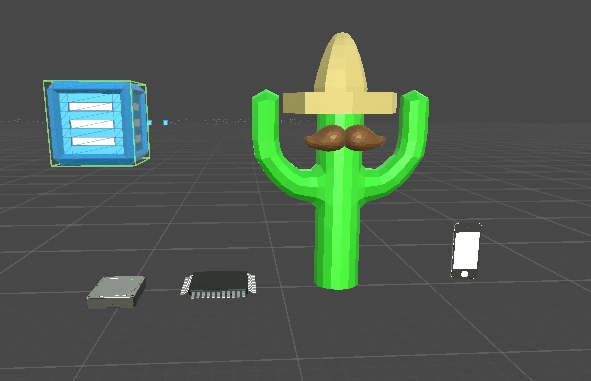
El último punto de control es la meta de los oponentes, de manera similar a la meta del jugador este cuenta las vueltas de los oponentes para determinar si alguno termino antes que el jugador, este no cuenta con un checkpoint siguiente, además este cuenta con una validación extra para cuando cruza el oponente final.



*Figura 4. Codigo metaIA*

Como se puede ver en la figura 4. El checkpoint valida cuando cruza un objeto con el tag “IA” que es un oponente cualquiera y respectivamente “IAFinal” siendo este el oponente final, cuando es el caso del oponente final este le reasigna una ruta aleatoria para seguir.

### **IV.3.2 Programación armas/obstáculos**

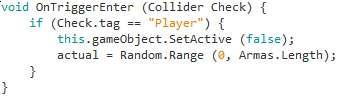


*Figura 4. Obstaculos/armas*

*/\* Referenciar a la parte de objetos descargados \*/*

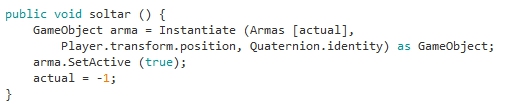
Los objetos utilizados como obstaculos (figura 4.) son asignados al jugador cuando este colisiona con la caja o cubo que los contiene, apreciable en la parte superior izquierda de la figura 4. , esta fue programada con un trigger que regresa un objeto aleatorio y se lo asigna al usuario.

Código objetos

  
*Figura 4. Codigo trigger de asignacion*

En la figura anterior se puede apreciar el procedimiento trigger que asigna un objeto aleatorio si el jugador es el que colisiona con la caja, los objetos se encuentran en un arreglo de nombre “Armas” y se obtiene la posicion para asignarla a la varibale “actual”.

  
*Figura 4. Funcion Update del objeto*

  
*Figura 4. Funcion soltar*

En la figura 4. se puede apreciar la funcion Update, esta se llama cada frame del juego y valida si fue precionada la tecla de espacio, cuando es precionada esta tecla se llama a la funcion soltar (ver figura 4.), esta instancia el objeto arma justo detrás del jugador y cambia el arma actual a -1, valor que se definio como si no tuviese ningun arma el jugador.

Los objetos que funcionan como armas contienen un colisionador de acuerdo a su forma, cuando un objeto colisiona con el ususario o con un oponente, este reducira su velocidad.