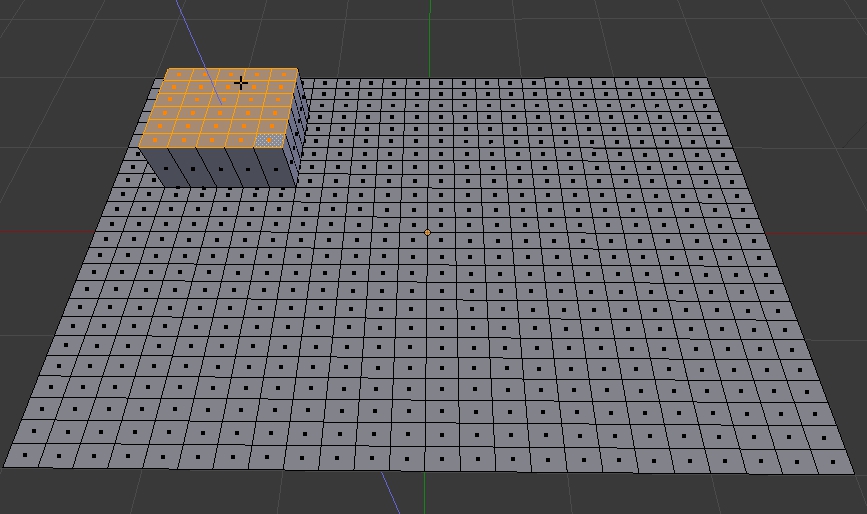
# Capítulo I: Diseño y modelos

## Diseño del Mapa

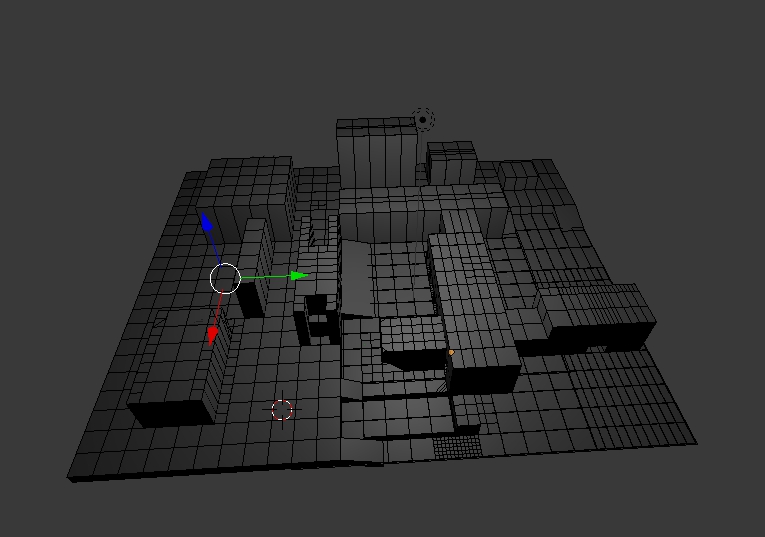
Para modelar el mapa y respectivamente los edificios se consideró tomar fotografías a todo el plantel, dado que este proceso llevaría bastante más tiempo para el modelado se optó por realizar los modelos basados en aproximaciones únicamente.

Los edificios se modelaron a partir de un plano base subdividido para facilitar un poco el trabajo de diseño tal y como se aprecia en la figura 1.1.



**Figura 1.1**

Este proceso se utilizó para todos los edificios e instalaciones para crear un único modelo que los contuviera a todos, dando el resultado mostrado en la figura 1.2.



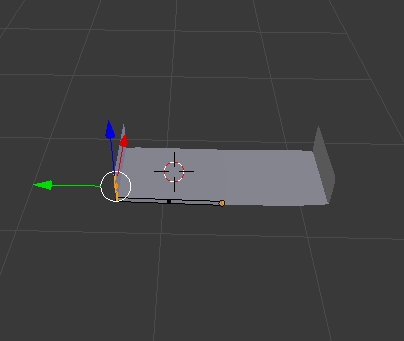
**Figura 1.2**

Después de tener el modelo completo, se diseñaron texturas para algunas zonas de este, solo para mejorar su efecto visual en juego, tal y como se muestra en la figura 1.3.

**/\* Edgar mete imagen de las texturas \*/**

## Modelado de pistas

Para diseñar las pistas se diseñó un fragmento únicamente de la pista apreciable en la figura 1.4.



**Figura 1.4**

Creado el fragmento, se duplicó varias veces a lo largo de una curva con el diseño del circuito para darle la forma deseada y se utilizó el mismo proceso para el diseño de las demás pistas tal y como se muestra en la figura 1.5.



**Figura 1.5**

## Modelado de coche y otros objetos

Para el 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 y libres.

# Capítulo II: Vistas y menús

## Menú principal

El menú principal, de opciones y de ajuste grafico se diseñaron usando canvas proporcionados por el propio editor de Unity, véase figura 2.1.



**Figura 2.1**

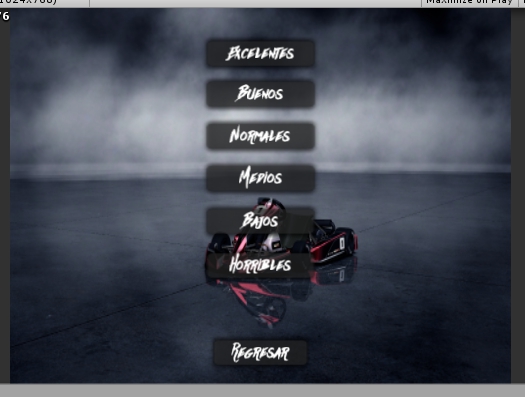
Cada uno de los botones maneja funciones similares cuando son presionados, estas ya están definidas por Unity para activar o desactivar un objeto cuando se presionan.

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

public function Boton() {  
    Application.LoadLevel("Escena");  
}

Este tipo de función carga la escena del modo historia, multijugador, selector de jugador y de mapa.

## Menú Grafico



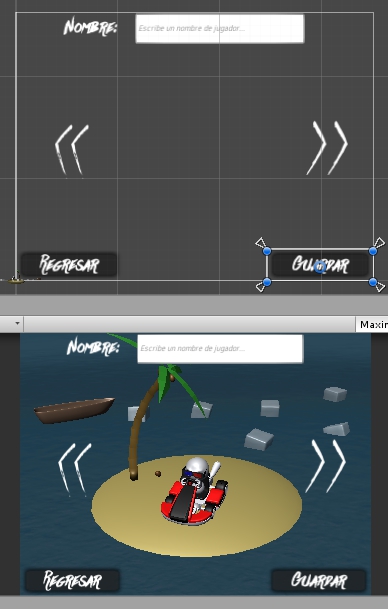
**Figura 2.2**

Como se aprecia en la figura 2.2, se pueden ajustar los gráficos desde el mismo juego, para esto cada botón utiliza un código como el que se muestra a continuación.

public function GraficosBuenos() {  
    QualitySettings.currentLevel = QualityLevel.Good;  
}

Esta función ajusta el valor de los gráficos en uno ya predefinido por Unity como “bueno”, de manera similar las demás funciones ajustan a distintos valores ya definidos para lograr otro efecto en los gráficos del juego;

## Selector de personaje



**Figura 2.3**

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

public void showSelectedKart() {  
    for (int i = 0; i < kart.Length; i++)  
        if (i == kartSelected)  
            kart [i].SetActive (true);  
        else  
            kart [i].SetActive (false);  
}

La función anterior cambia entre los distintos personajes definidos en un arreglo que los contiene a todos, la variable kartSelected contiene el número del personaje seleccionado actualmente.

public void next() {  
    kartSelected = (kartSelected + 1 + kart.Length) % kart.Length;  
    showSelectedKart ();  
}

La función anterior calcula el número del personaje y llama a la función que lo muestra, esta cambia el valor al siguiente personaje. Basada en esta función se define una que calcula el valor anterior con la misma estructura.

void Start() {  
    if (PlayerPrefs.HasKey("kart"))  
        kartSelected =  PlayerPrefs.GetInt ("kart");  
    if (PlayerPrefs.HasKey ("nombre"))   
        txtNombre.text = PlayerPrefs.GetString ("nombre");  
    if (PlayerPrefs.HasKey ("progreso"))   
        progreso = PlayerPrefs.GetInt ("progreso");  
    showSelectedKart ();

}

Para cargar los datos del usuario que contienen el nombre, el carro seleccionado y el progreso se utiliza la clase PlayerPrefs, esta carga una variable con una etiqueta ya definida al guardar.

## Selector de mapa



**Figura 2.4**

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

****

**Figura 2.5**

En la figura 2.5 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. El procedimiento seleccionarMapa funciona recibiendo el número de pista o nivel del botón presionado y guarda en qué nivel iniciara el jugador cuando se cargue la pista.

# Capítulo III: Programación

## 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 3.1**

En el código anterior (figura 3.1) 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.

var x Number = Input.GetAxis("Horizontal");

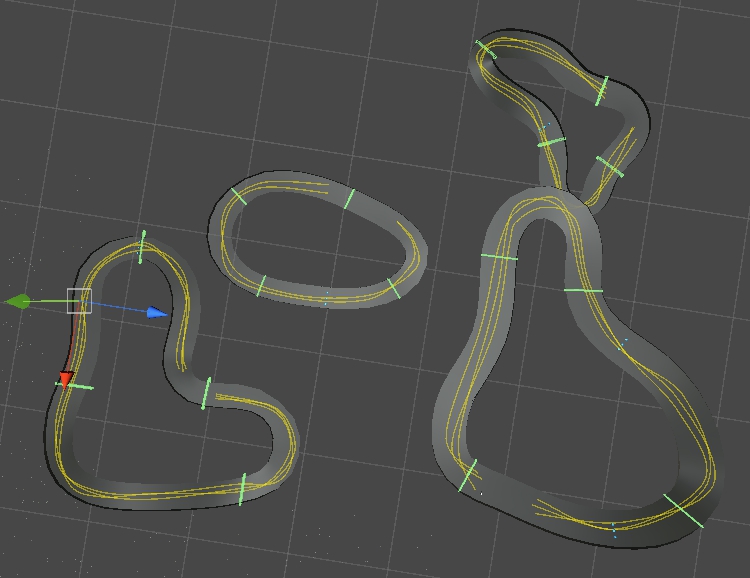
var y: Number = Input.GetAxis("Vertical");

this.GetComponent.<Rigidbody>().velocity += y \* transform.forward \* forwardSpeed;

this.GetComponent.<Rigidbody>().AddTorque(transform.up \* x \* steerAngle, ForceMode.Acceleration);

El código anterior 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.

## Checkpoints



**Figura 3.**

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 3.x 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.

Existen 3 distintos puntos de control en el juego, estos se detallaran a continuación:

1. Meta: Este punto de control como su nombre lo indica funciona como línea de inicio/fin para el jugador.