Graficación

Reporte Proyecto Final

Cd. Juárez, Chihuahua a 03 de junio del año 2020

Docente: Noé R. Rosales Morales

Integrantes:

Héctor Manuel Astorga Navarro

Nancy Alejandra Calderón Soto

Victor Manuel Segura Osua

Índice

Planteamiento del problema

A medida que avanza el tiempo, el poder crear videojuegos y/o animaciones ha estado cambiando la forma de ser accesible para un chico común y cualquiera, pues a veces no se tienen los recursos suficientes para crear algo así, nosotros quisimos crear un video juego clásico que se jugaba en los años 90s con PlayStation 1 o los súper Nintendo, creando una simulación de una carrera.

Teníamos que indagar demasiado pues es un video juego algo viejo y para poder recordar cómo eran los gráficos, las texturas, que era lo que este videojuego podía hacer y qué no.

Un video juego de muchos años pensamos que podría ser la solución para volver a crear una memoria a tiempos anteriores donde los videojuegos no tienen tanta “violencia” así que este es nuestro nuevo problema.

Definición del problema

Este problema podemos considerarlo un poco básico y sencillo al verlo por primera vez en cualquier plataforma de video juegos, pero la verdad es que no lo es pues se necesita mucha imaginación además de meterte dentro del juego para que veas un entorno más definido a los detalles que puedes agregar, pues estos son muy importante para la elaboración de animaciones y videojuegos así que con estas medidas ya implementadas queremos recrear tu niñez con este juego, que vuelvas a sentir como la adrenalina sube al jugarlo ese es nuestro objetivo para crear este proyecto.

Hipótesis

Para el problema planteado teníamos varias soluciones que a medida se fueron descartando, en si podíamos necesitar de muchas extensiones para poder crear este videojuego, el punto era que y cuales extensiones nos ayudarían a facilitar este problema.

Pensamos en la utilización de blender pues se nos hacía más sencillo a la hora de estar codificando. Además de atributos de texturas que es lo más importante para la creación de nuestro videojuego porque no queríamos que se viera como un juego antiguo casi original si no, algo antiguo con renovaciones de futuro, mucha mejor calidad y jugabilidad de aquel que lo obtenga.

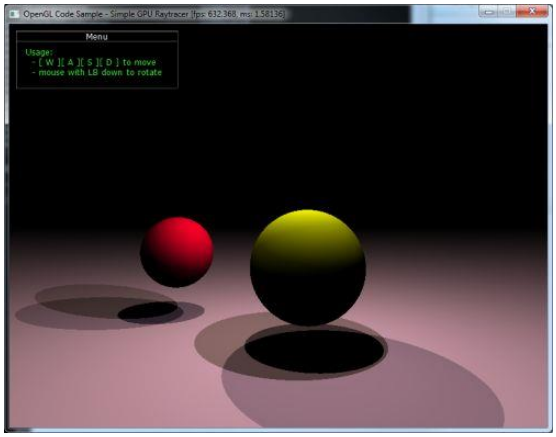
Cuando se programa también tiene que haber un tiempo real de movimiento para que la secuencia de lo que el jugador este haciendo concuerde cada segundo de lo que está pasando, además de crear perspectivas cuando este se esté moviendo no genere algún tipo de bug o la imagen se llegue a congelar para ellos se creó un storyborad para tener una secuencia de lo que puede estar pasando al momento de estarse moviendo y sea de buen ver para el jugador.

No todo se puede hacer tan fácil pues esta situación la planteamos para hacer en lenguaje c++ y no lo es todo, para poder trabajar con animaciones demás de crear video juego se requiere de tener una computadora al menos de gama media, pues al momento de renderizar el videojuego este te consumirá bastante memoria pues es algo muy pesado de transformar de código a interfaz y para esto probamos entre las diferentes cpus

de nuestro equipo pues ahí se puede ver cual puede soportar más la creación de este juego.

Objetivo

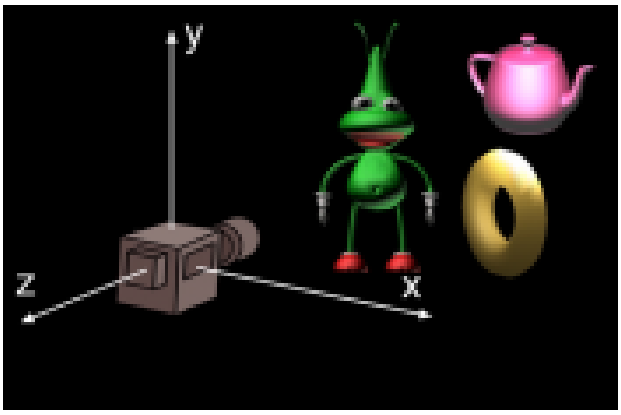
El implementar interfaces con un diseño detallado en cada cosa, en cada carretera, movimiento e incluso personajes, puede recrear la memoria y satisfacción de estar dentro del juego, poder sentir como avanzas a medida que te mueves, a medida que vas pasando cada uno de los niveles pues queremos que te sientas como un jugador único puesto que los video juegos para eso son, para sentirnos bien con nosotros y espejarnos del mundo de afuera, además de recrear la sensación de estar en la niñez aun siendo un joven o incluso un adulto que quiera recrear aquella escena cuando jugo este juego, pues queremos destrozar los límites de la imaginaciones y sensaciones para que cualquiera que juegue nuestro proyecto se sienta libre por consiguiente crearemos un buen ambiente para el jugador que hará , tal vez no popular, pero si recomendable el juego en caso de futuros diversos.



Marco teórico

Nos basamos en muchas actividades básicas para la trasformación de este proyecto pues iniciamos el cómo creamos una simple clase hasta como mover y configurar texturas puesto que las herramientas más básicas hacen la diferencia.

Claro empezando nuestro conocimiento es como crear ventanas en opengl pero... para empezar ¿qué es opengl?



Opengl es una librería grafica que permite la utilización de más de 200 órdenes para generar aplicaciones interactivas en 3D, la librería está diseñada para ser independiente de la plataforma y del sistema operativo, por ello no contiene ordenes de gestión de ventana ni de dispositivos de entrada, para realizar estas operaciones se necesitan otras librerías como la GLUT.

¿Qué es GLUT?

La librería GLUT es un paquete auxiliar para construir aplicaciones de ventanas además de incluir algunas primitivas geométricas auxiliares, lo bueno este paquete es que el mismo código nos sirve para Windows, así como también para LUNIX.

También lo que se podría decir otra librería que lograremos implementar y comprender seria la librería SFML

SFML es una biblioteca multiplataforma (Windows, Mac OS X y Linux) escrita en C++ y totalmente orientada a objetos para el desarrollo de aplicaciones multimedia enfocada en el desarrollo de videojuegos 2D.

La actual versión estable es la 1.6, pero en este artículo vamos a hablar de SFML 2 que se encuentra en Release Candidate

desde el 15 de abril y desde entonces no se ha parado de pulir y mejorar. Nos encontramos en una versión bastante madura con muchas novedades.

No obstante, tenemos módulos en la librería de SFML:

- **System**: Este es el módulo básico de SFML, nos proporciona clases útiles para el desarrollo como manejo sencillo de hilos, control del tiempo, plantillas para manejar vectores, streams, cadenas, utf, etc.
- **Window**: El módulo window nos sirve para el manejo de la ventana de nuestra aplicación usando este módulo se puede crear un contexto OpenGL en el que dibujar directamente desde OpenGL.
- **Graphics**: Nos proporciona un contexto especial sobre Window donde se puede "dibujar". Además, nos proporciona varias clases útiles para el manejo de imágenes, texturas (imágenes vivas en la tarjeta gráfica), colores, sprites, textos y figuras 2D como círculos, rectángulos y formas convexas.
- **Audio**: Nos proporciona varias clases para trabajar con el audio, dos tipos de audio hay en SFML por defecto Sound que es un archivo de sonido corto que se carga en memoria y Music que son archivos largos de audio que se van reproduciendo en Stream. Soporte para sonido 3D.
- **Network**: Colección de clases que nos facilita la creación de aplicaciones en red. Cuenta con clases para el manejo de http, ftp, packet, socket, etc.

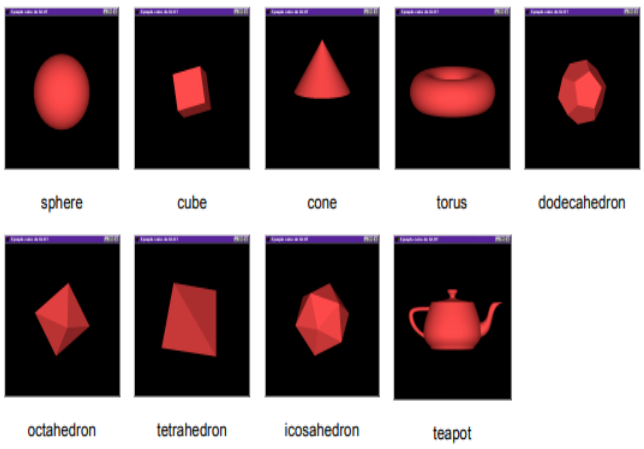
Puede que todo esto sea sencillo pero la verdad es que tiene su complejidad pues además de aprender los conceptos básicos se tuvo que aprender las primitivas geométricas básicas para poder crear nuestro proyecto pues de aquí se deriva todas las escenas del proyecto, usamos tal cual las funciones glut claro que hay una gran cantidad, así como...

- glutWireSphere(radius, slices, stacks), glutSolidSphere(radius, slices, stacks)
- glutWireCube(size), glutSolidCube(size)
- glutWireCone(base, height, slices, stacks), glutSolidCone(base, height, slices, stacks)
- glutWireDodecahedron(void), glutSolidDodecahedron(void)
- glutWireOctahedron(void), glutSolidOctahedron(void)
- glutWireTetrahedron(void), glutSolidTetrahedron(void)

Que claro son las funciones básicas para poder crear figuras dentro del opengl y a la vez crear el entorno de un juego o animación básica para su uso, no obstante, no solo se usan figuras y ya también tenemos la capacidad de implementar colores a los que son las figuras pues una animación sin colores no se definirá muy bien lo cual sentencias sencillas como:

- glColor3f(0.5f, 0.5f, 0.5f);
- glutWireTeapot(0.5);
- glBegin(GL_LINES);

Nos facilitan el modo de poder cambiar las cosas a manera de que vayamos avanzando en las creaciones básicas del proyecto.

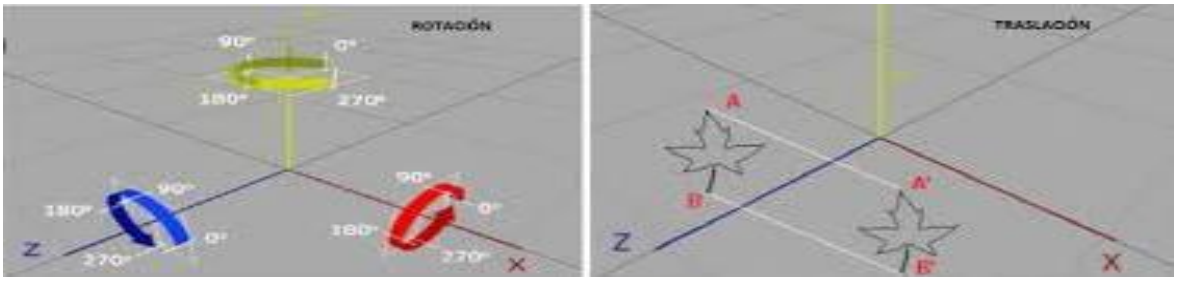


Para poder hacer este proyecto no solo tuvimos que aprender de colores y figuras, también se tuvo que aprender sobre la traslación que bien es fácil con una sencilla sentencia:

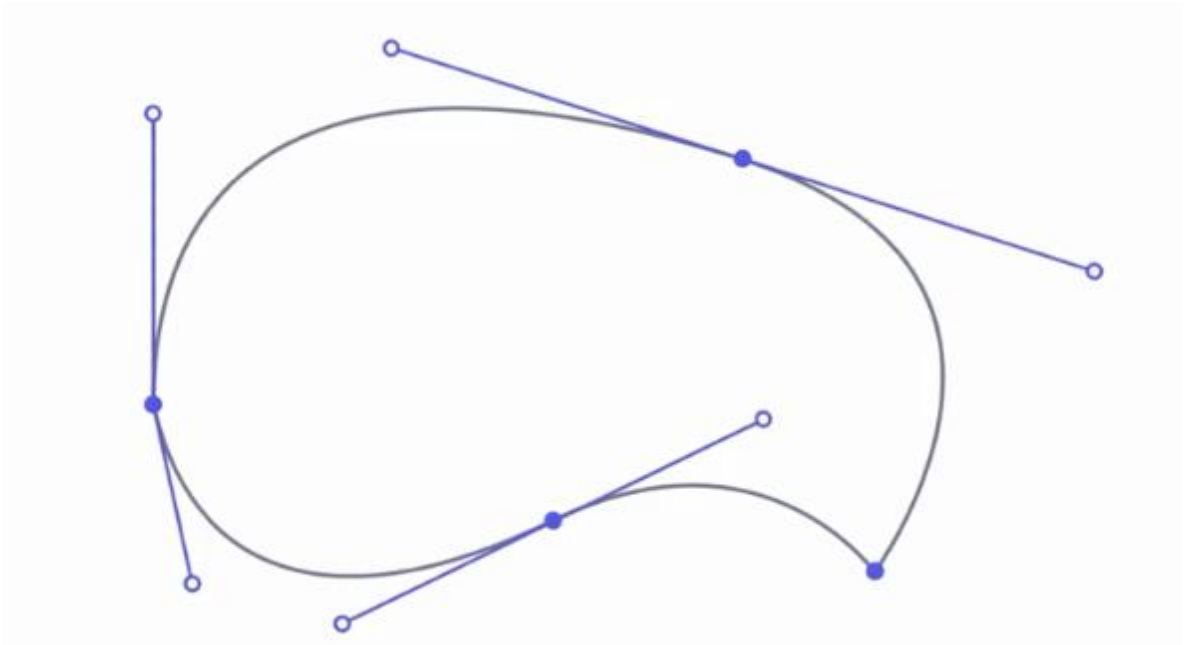
- glTranslatef(x, y, z)

Pues no solo la tranlacion se nos hace suficiente para hacer todo esto por lo cual se tuvo que aprender además de la traslación el escalado para agrandar las figuras.

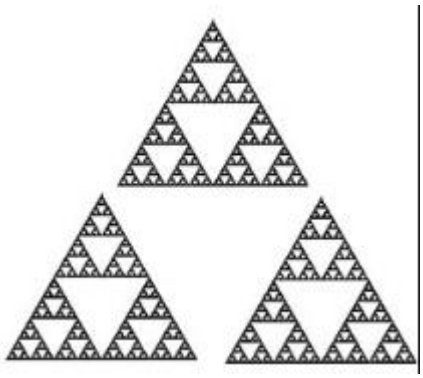
- glScalef(sx, sy, sz)



Además de utilizar lo que es las curvas Bézier las cuales nos definen una forma perfecta de lo que queramos transformar mediante figuras con líneas.



No obstante, tuvimos más conocimientos más profundos sobre como codificar en el opengl llegando a si a los fractales...



Un fractal es un objeto geométrico cuya estructura básica es irregular además de que se repite en diferentes escalas. Esto nos ayuda a poder crear formas irregulares y a la vez varias para poder crear un entorno deseado.

Ahora si viene lo que nos interesa, lo que es la animación en 3D lo cual esto es lo más importante para poder hacer la interfaz e interacción del juego con el jugador, no usamos todos los **Tipos de animación 3D**, pero si hay que saber de lo básico y de lo que es cada uno de los tipos.

Cel-Shaded

Es un tipo de renderización no fotorrealista diseñada para hacer que los gráficos por computadora parezcan dibujados a mano. Las sombras planas se usan comúnmente para imitar el estilo de los cómics o dibujos animados.

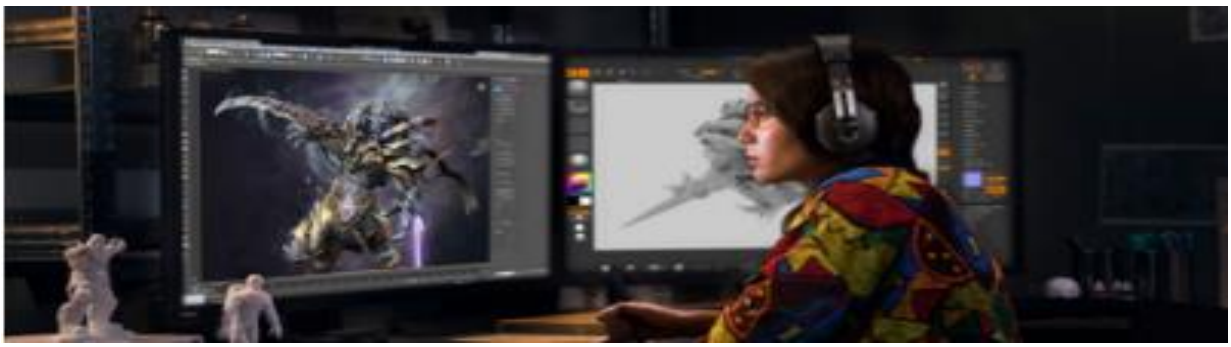
Morph

En este caso no se trata de modificar una imagen sino la forma del objeto en tres dimensiones. Este cambio continuo puede utilizarse en animación para representar deformaciones o crear efectos visuales. Otro uso, más técnico, consiste en suavizar las transiciones entre diferentes representaciones de un mismo objeto cuando éstas tienen diferente nivel de detalle.

Skeletal

Se crea una representación simplificada del cuerpo del personaje, análogo a un esqueleto o a un stickman. En personajes humanos y animales, muchas partes del modelo de esqueleto corresponden a la ubicación real de los huesos, pero la animación del modelo de esqueleto skeletal animation es también utilizada para animar otras cosas, como expresiones faciales.

Ahora bien, nos entramos a lo último que aprendimos que es la animación controlada por el usuario o llamada animación digital lo cual consiste en crear imágenes en movimiento lo cual hacer que el que corra el juego o lo pruebe se pueda mover a través del entorno de la interfaz de la programación creando una perspectiva como en nuestra vida real, así como todos los juegos ya creados en el mundo.



Desarrollo

Para nuestro proyecto comenzamos con un tamaño formal de lo que es la ventana de interacción lo cual pensamos que las dimensiones normales o básicas para poder operar el juego

```
int width = 1024;
int height = 768;
int roadW = 2000;
int segL = 50; //Tamano del segmento
float camD = 0.84; //Profundidad de la camara
```

Para nuestro juego como es un simulador de carreras basado en un juego de los años 90s entonces tuvimos que ejemplificar el modelo de diseño con algo similar lo cual aquí tenemos un ejemplo de lo que es el diseño del juego a crear.

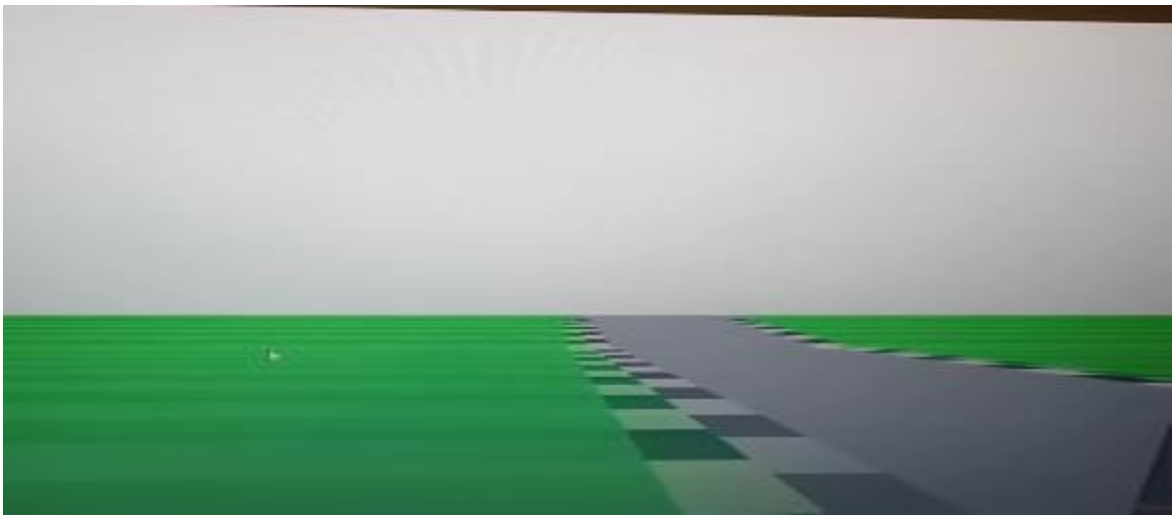
Ya una vez visto el detalle empezamos a crear los centros de línea de 3d y las coordenadas de la pantalla.

```
struct Line
{
    float x, y, z; //Centro de la linea en 3d
    float X, Y, W; //Coredenadas de la pantalla
    float curve, spriteX, clip, scale;
    Sprite sprite;

    Line()
    {
        spriteX = curve = x = y = z = 0;
    }

    void project(int camX, int camY, int camZ)
    {
        scale = camD / (z - camZ);
        X = (1 + scale * (x - camX)) * width / 2;
        Y = (1 - scale * (y - camY)) * height / 2;
        W = scale * roadW * width / 2;
    }
```

```
    void drawSprite(RenderWindow& app)
    {
        Sprite s = sprite;
        int w = s.getTextureRect().width;
        int h = s.getTextureRect().height;

        float destX = X + scale * spriteX * width / 2;
        float destY = Y + 4;
        float destW = w * W / 266;
        float destH = h * W / 266;

        destX += destW * spriteX; //offsetX
        destY += destH * (-1);    //offsetY

        float clipH = destY + destH - clip;
        if (clipH < 0) clipH = 0;

        if (clipH >= destH) return;
        s.setTextureRect(IntRect(0, 0, w, h - h * clipH / destH));
        s.setScale(destW / w, destH / h);
        s.setPosition(destX, destY);
        app.draw(s);
    }
};
```

Ahora si ya nos inspiramos a lo grande que es la creación de texturas y los fondos para el juego.

```
    RenderWindow app(VideoMode(width, height), "Juego de carreras");
    app.setFramerateLimit(60);

    Texture t[50];
    Sprite object[50];
    for (int i = 1; i <= 7; i++)
    {
        t[i].loadFromFile("images/" + std::to_string(i) + ".png");
        t[i].setSmooth(true);
        object[i].setTexture(t[i]);
    }

    Texture bg;
    bg.loadFromFile("images/cielo.gif");
    bg.setRepeated(true);
    Sprite sBackground(bg);
    sBackground.setTextureRect(IntRect(0, 0, 5000, 411));
    sBackground.setPosition(-2000, 0);
```



Como todo jugador será necesario poner un “mando” con el cual puedas tu controlar el juego por lo cual le pusimos las teclas básicas del teclado que todo juego contiene para mejor movilidad.

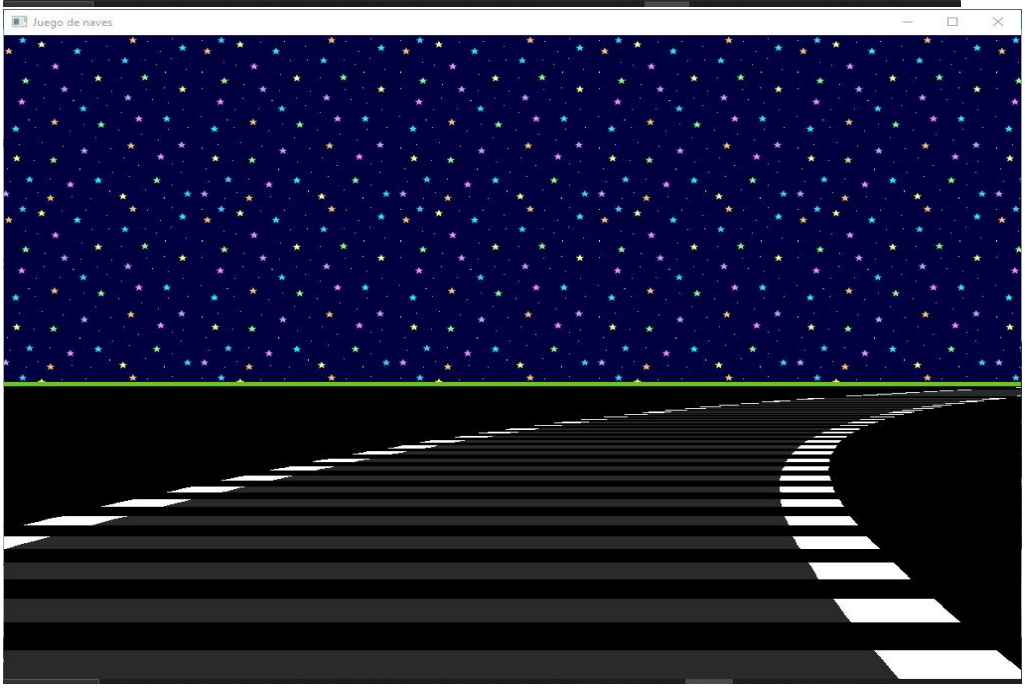
```
int speed = 0;

if (Keyboard::isKeyPressed(Keyboard::Right)) playerX += 0.1;
if (Keyboard::isKeyPressed(Keyboard::Left)) playerX -= 0.1;
if (Keyboard::isKeyPressed(Keyboard::Up)) speed = 200;
if (Keyboard::isKeyPressed(Keyboard::Down)) speed = -200;
if (Keyboard::isKeyPressed(Keyboard::Tab)) speed *= 3;
if (Keyboard::isKeyPressed(Keyboard::W)) H += 100;
if (Keyboard::isKeyPressed(Keyboard::S)) H -= 100;

pos += speed;
while (pos >= N * segL) pos -= N * segL;
while (pos < 0) pos += N * segL;

app.clear(Color(105, 205, 4));
app.draw(sBackground);
int startPos = pos / segL;
int camH = lines[startPos].y + H;
if (speed > 0) sBackground.move(-lines[startPos].curve * 2, 0);
if (speed < 0) sBackground.move(lines[startPos].curve * 2, 0);

int maxy = height;
float x = 0, dx = 0;
```

Una vez creado los comandos, las texturas, las ventanas y líneas... como nuestro juego es de una simulación de carreras lo esencial es claro pintar una ruta la cual se va a ir moviendo y generando curvas a lo largo del camino simulando una carrera de verdad.

```
///////Dibujado del camino////////
for (int n = startPos; n < startPos + 300; n++)
{
    Line& l = lines[n % N];
    l.project(playerX * roadW - x, camH, startPos * segL - (n >= N ?
    x += dx;
    dx += l.curve;

    l.clip = maxy;
    if (l.Y >= maxy) continue;
    maxy = l.Y;

    Color Espacio = (n / 3) % 2 ? Color(0, 0, 0) : Color(0, 0, 0);
    Color Camellon = (n / 3) % 2 ? Color(0, 0, 0) : Color(255, 255, 2
    Color Camino = (n / 3) % 2 ? Color(0, 0, 0) : Color(42, 42, 42);
```

```
Color Espacio = (n / 3) % 2 ? Color(0, 0, 0) : Color(0, 0, 0);
Color Camellon = (n / 3) % 2 ? Color(0, 0, 0) : Color(255, 255, 2
Color Camino = (n / 3) % 2 ? Color(0, 0, 0) : Color(42, 42, 42);


Line p = lines[(n - 1) % N]; //Linea previemanete dibjada

drawQuad(app, Espacio, 0, p.Y, width, 0, l.Y, width);
drawQuad(app, Camellon, p.X, p.Y, p.W * 1.2, l.X, l.Y, l.W * 1.2)
drawQuad(app, Camino, p.X, p.Y, p.W, l.X, l.Y, l.W);
```

Aquí podemos apreciar un resultado en primera persona demás de la textura y encamino de la carrera, preferimos hacerlo de noche pues creemos que puede tener un poco más atracción hacia el juego.



Conclusión

Esta materia se podría decir que es algo tediosa pero satisfactoria pues al crear tu propia animación te crea, genera un cosquilleo de querer aprender más y más lo cual hace más grande tu hambre por aprender, este proyecto se nos hace bien y satisfactorio pues emplea todo lo aprendido en clase por consiguiente es el fruto de un gran esfuerzo, dedicación, desvelos para la creación de este proyecto además de que nos divertimos en equipo a la hora de generar ideas para poder plasmarlos a lo largo de las unidades no obstante la programación nos puede ayudar en muchas ocasiones en la vida y no solo por el código si no porque para poder generar algo hay que estar ahí, organizar tus ideas para después plantarlas con todo por consecuencia tendrás un buen trabajo si lo haces como si fuera un código.

Recomendaciones

Estudia y lucha por tus sueños, porque nadie más que tu podrás ser el creador de tu futuro.

Referencias

Antonio, J. (03 de 06 de 2020). *Graficacion*. Obtenido de https://joseantonioarguellopalacios.blogspot.com/2019/11/animacion-por-computadora.html

Digital, C. I. (03 de 06 de 2020). *Animacion 3D*. Obtenido de https://www.campusproducciondigital.com/blog/item/118-tecnicas-en-animacion-3d

M. Pauline Baker. (2006). En D. Heran, *Grarficos por computadora con OpenGL* (pág. 918). Madrid: PEARSON.

Morales, C. C. (03 de 06 de 2020). *Opengl Y Glut*. Obtenido de https://sis2430.files.wordpress.com/2009/11/tutorial-1.pdf