UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

EJERCICIOS DE CLASE Nº 01

NOMBRE COMPLETO: CARBAJAL REYES IRVIN JAIR

Nº de Cuenta: 422042084

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 15 DE FEBRERO DE 2025

CALIFICACIÓN: _____

EJERCICIOS DE SESIÓN:

1. Actividades realizadas. Una descripción de los ejercicios y capturas de pantalla de bloques de código generados y de ejecución del programa

2. Problemas presentados. Listar si surgieron problemas a la hora de ejecutar el código

3. Conclusión:
    a. Los ejercicios de la clase: Complejidad, explicación
    b. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias.

1) Actividades

A. Cambiar el color de fondo de la pantalla entre rojo, verde y azul de forma cíclica y solamente mostrando esos 3 colores con un periodo de lapso adecuado para el ojo humano

Primero, se implementó dentro de la función main una variable para registrar el tiempo de inicio en el cual se ejecutó el programa. Es importante resaltar que esta variable es conveniente declararla fuera del ciclo ***!glfwWindowShouldClose(mainWindow)*** para que los tiempos no se alteren demasiado. Además se declaró una variable que indicará el color dependiendo de su valor, en nuestro caso 0 representa el rojo, 1 representa el verde y 2 representa el azul.

```
170        // Guardar tiempo de inicio
171        auto startTime = std::chrono::high_resolution_clock::now();
172        int color = 0;
```

Luego, dentro del ciclo antes mencionado, se declara otra variable de tiempo pero esta vez almacenará el tiempo actual, esto para después obtener a partir de una diferencia entre el tiempo actual (***currentTime***) y el tiempo de inicio (***startTime***) el tiempo efectivo que ha transcurrido en la ejecución.

```
180        // Calcular el tiempo transcurrido
181        auto currentTime = std::chrono::high_resolution_clock::now();
182        std::chrono::duration<double> tiempo = currentTime - startTime;
```

Una vez que almacenamos el tiempo efectivo de ejecución transcurrido en segundos, podemos usar un condicional para que una vez que haya transcurrido dos segundos, sumar una unidad a la variable ***color*** para que represente la siguiente tonalidad, además de reiniciar nuestro *timer* igualando el tiempo de inicio (***startTime***) con el tiempo actual (***currentTime***).

```
183        if (tiempo.count() >= 2.0) {
184            // Cambiar el color cada 2 segundos
185            color = color++;  // 0 -> rojo, 1 -> verde, 2 -> azul
186            startTime = currentTime;
187        }
```

Finalmente, colocamos los condicionales para que se le asigne el color de fondo correspondiente al valor de ***color*** asignado.

```
190         if (color == 0) {
191             glClearColor(1.0f, 0.0f, 0.0f, 1.0f); // Rojo
192         }
193         else if (color == 1) {
194             glClearColor(0.0f, 1.0f, 0.0f, 1.0f); // Verde
195         }
196         else if (color == 2) {
197             glClearColor(0.0f, 0.0f, 1.0f, 1.0f); // Azul
198         }
199         else {
200             color = 0;
201         }
202
203         glClear(GL_COLOR_BUFFER_BIT);
```

Notamos que una vez que ***color*** llega al valor 2, a la variable se le asigna nuevamente el valor 0 para que represente la tonalidad inicial, en nuestro caso rojo.

B. Dibujar de forma simultánea en la ventana 1 cuadrado y 1 rombo separados.

Para este ejercicio, primero fue necesario determinar la cantidad de triángulos que se necesitan, en este caso son cuatro, por lo que necesitamos declarar doce vértices, así que ese número lo colocamos como parámetro de la función ***glDrawArrays***.

```
208         glDrawArrays(GL_TRIANGLES,0,12);
```

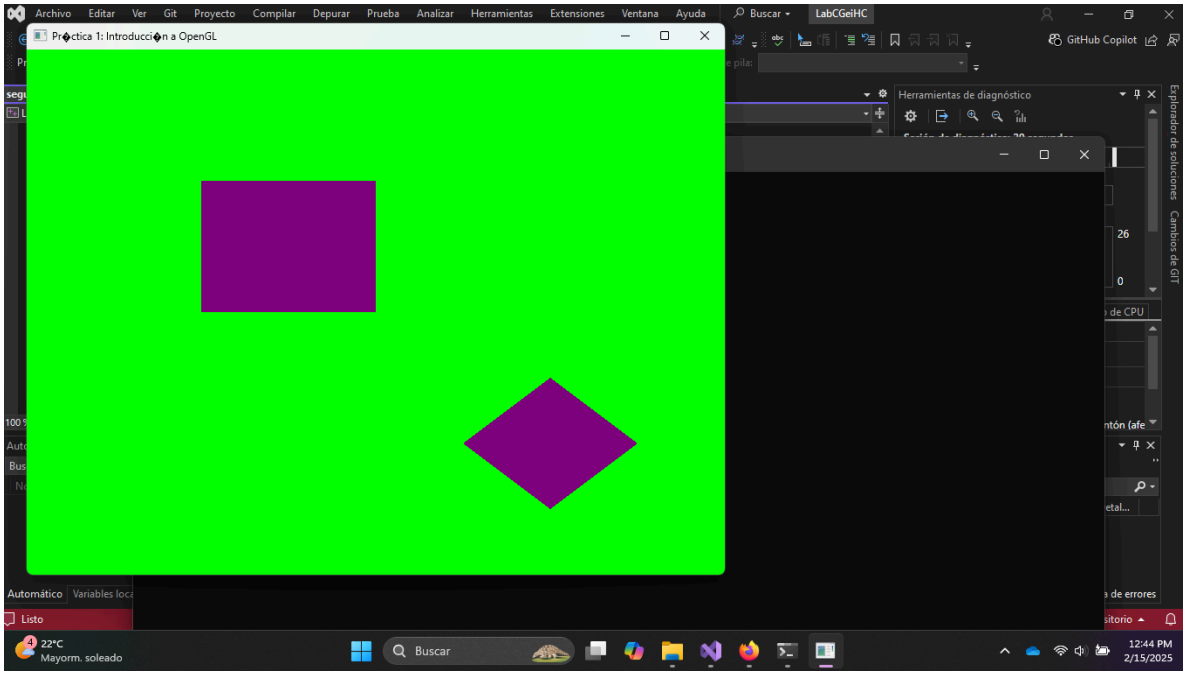
Luego, declaramos los vértices de cada uno de los triángulos que necesitamos, dos para el cuadrado y dos para el rombo, tomando en cuenta que su ubicación no se intercepte entre ambas figuras.

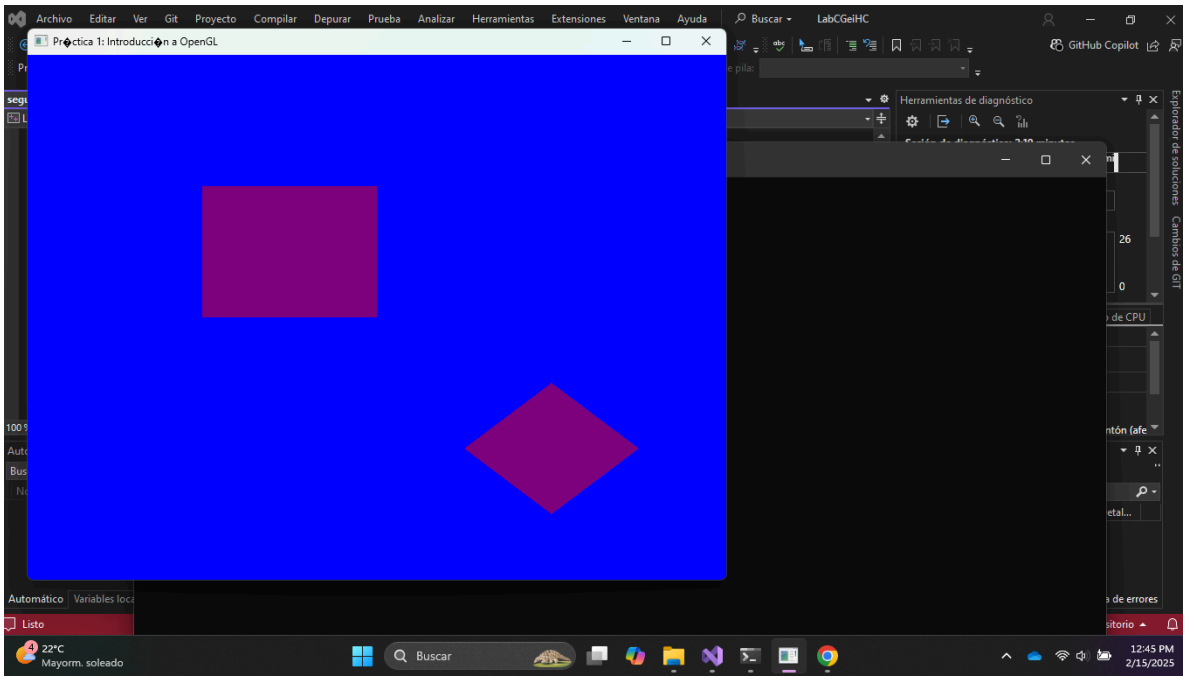
```
31  void CrearTriangulo()
32  {
33      GLfloat vertices[] = {
34          0.0f, 0.0f,0.0f,
35          0.0f, 0.5f, 0.0f,
36          -0.5f, 0.0f, 0.0f,//triangulo 1
37          -0.5f,0.0f, 0.0f,
38          0.0f, 0.5f,0.0f,
39          -0.5f,0.5f, 0.0f, //triangulo 2
40          0.5f,-0.25f, 0.0f,
41          0.25f,-0.5f, 0.0f,
42          0.75f,-0.5f, 0.0f, //triangulo 3
43          0.25f,-0.5f, 0.0f,
44          0.75f,-0.5f, 0.0f,
45          0.5f,-0.75f,0.0f //triangulo 4
46      };
```

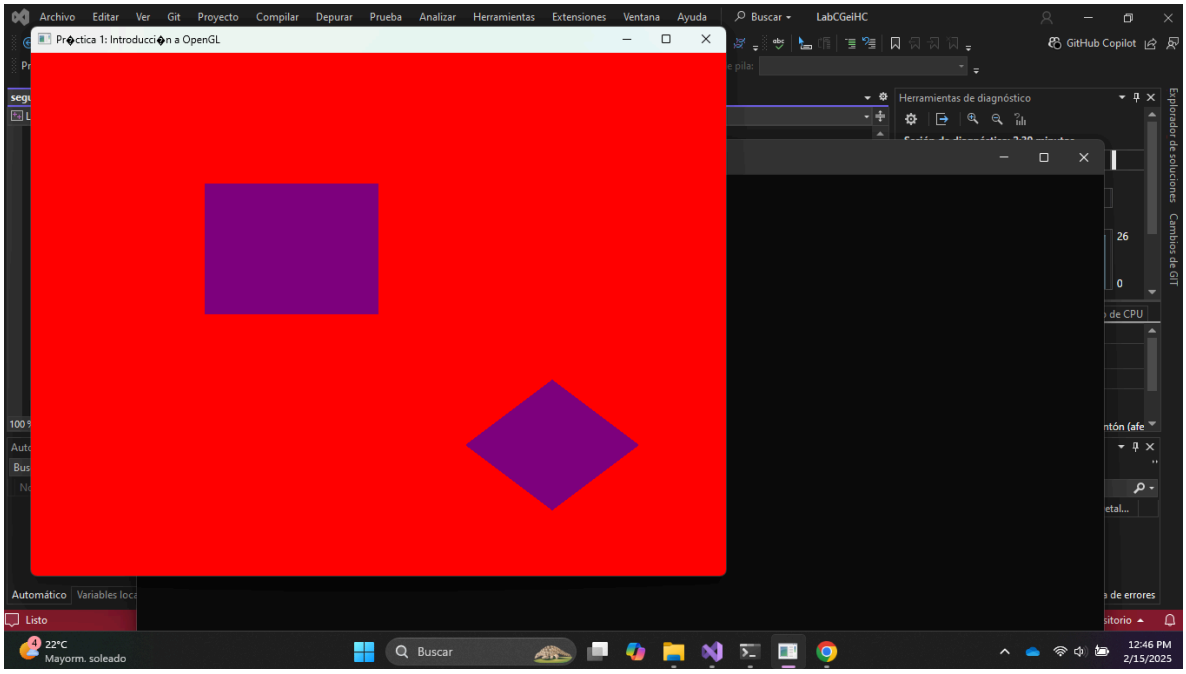
También fue necesario cambiar el color de los triángulos con el fin que su tonalidad no se pierda con alguna de las del fondo, se optó por el color morado.

```
24      void main()                                          \n\
25      {                                                    \n\
26          color = vec4(0.502f,0.0f,0.502f,1.0f);               \n\
27      }";
```

A continuación se muestran capturas de la ejecución:




Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Analizar Herramientas Extensiones Ventana Ayuda
Buscar
LabCGeiHC
GitHub Copilot
Pr@ctica 1: Introducci@n a OpenGL
Herramientas de diagnóstico
Automático
Variables loca
Listo
22°C
Mayorm. soleado
Buscar
12:45 PM
2/15/2025


Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Analizar Herramientas Extensiones Ventana Ayuda
Buscar
LabCGeiHC
GitHub Copilot
Pr@ctica 1: Introducci@n a OpenGL
Herramientas de diagnóstico
Automático
Variables loca
Listo
22°C
Mayorm. soleado
Buscar
12:46 PM
2/15/2025

2) Problemas presentados

Durante el desarrollo de los ejercicios donde se presentaron más problemas fue en la parte del cambio de color de fondo, ya que no se tenía el conocimiento de donde era más óptimo colocar los elementos del temporizador provocando que se cometieran errores como el colocar el inicio del temporizador dentro de un ciclo provocando retrasos. Además que se implementaron otras maneras de medir los intervalos de tiempo para el cambio de color pero resultaba muy tardada la ejecución del proyecto, sin embargo, se resolvió implementando dos variables para almacenar tanto el tiempo de inicio de ejecución como el tiempo actual para poder obtener intervalos de tiempo adecuados, lo que también nos permitió reiniciar el *timer* en el momento que creíamos adecuando y realizar los cambios de color.

3) Conclusiones

Se lograron satisfactoriamente todos los ejercicios de clase propuestos colocando doce triángulos que representan un cuadrado y un rombo, asignándole un color distinto a los que tomaba el fondo del programa, ya que este cambiaba cada dos segundos. La complejidad de los ejercicios me parece adecuada para una primera práctica del curso, presentándose algunas problemáticas provocadas por la falta de práctica en el lenguaje pero solucionables con el avance del curso. La explicación de las actividades fue clara y no se presentaron dudas acerca de los elementos involucrados para realizarlos.