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 1

NOMBRE COMPLETO: Gomez Enríquez Agustin

Nº de Cuenta: 317031405

GRUPO DE LABORATORIO: 3

GRUPO DE TEORÍA: 5

SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 19 – agosto - 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**

```
166         // Loop principal
167         while (!glfwWindowShouldClose(mainWindow)) {
168             glfwPollEvents();
169
170             // Cambiar fondo cada COLOR_PERIOD
171             double tNow = glfwGetTime();
172             if (tNow - tLast >= COLOR_PERIOD) {
173                 bgIndex = (bgIndex + 1) % 3;
174                 tLast = tNow;
175             }
176
177             float r = (bgIndex == 0) ? 1.0f : 0.0f;
178             float g = (bgIndex == 1) ? 1.0f : 0.0f;
179             float b = (bgIndex == 2) ? 1.0f : 0.0f;
180
181             glClearColor(r, g, b, 1.0f);
182             glClear(GL_COLOR_BUFFER_BIT);
```

Para generar el cambio de colores nos vamos al loop donde vamos a cambiar el fondo cada 2 segundos (de manera que es apreciable al ojo humano). Este cambio va a ser cíclico, es decir: rojo, verde, azul y se repite constantemente.

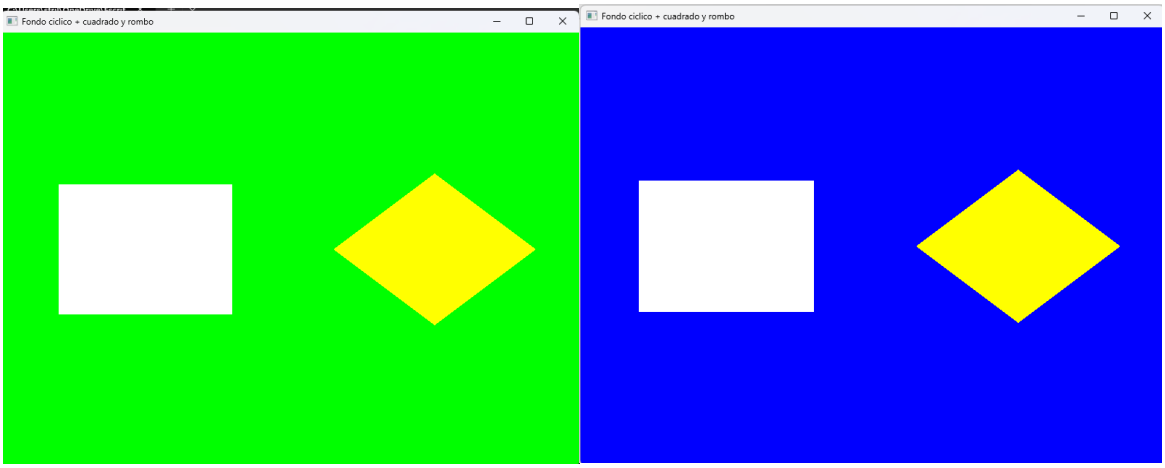
```
63  // ===== Constructores de formas 2D (llenadas con triángulos) =====
64  static void agregarCuadrado(std::vector<GLfloat>& verts, float cx, float cy, float tam) {
65      // cuadrado axis-aligned centrado en (cx,cy) con lado = tam
66      float m = tam * 0.5f, x0 = cx - m, x1 = cx + m, y0 = cy - m, y1 = cy + m;
67      // triángulo 1
68      verts.insert(verts.end(), { x0,y0, x1,y0, x1,y1 });
69      // triángulo 2
70      verts.insert(verts.end(), { x0,y0, x1,y1, x0,y1 });
71  }
72
73  static void agregarRombo(std::vector<GLfloat>& verts, float cx, float cy, float r) {
74      // "rombo" (diamante): un cuadrado rotado 45°, definido por sus 4 vértices cardinales
75      // top, right, bottom, left
76      float xt = cx, yt = cy + r;
77      float xr = cx + r, yr = cy;
78      float xb = cx, yb = cy - r;
79      float xl = cx - r, yl = cy;
80      // Dos triángulos: (top, left, bottom) y (top, right, bottom)
81      verts.insert(verts.end(), { xt,yt,  xl,yl,  xb,yb });
82      verts.insert(verts.end(), { xt,yt,  xr,yr,  xb,yb });
83  }
```

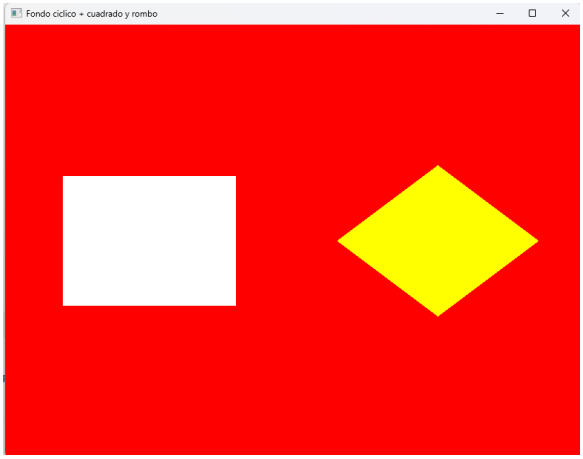
Para la cuestión de las figuras, creamos nuestras funciones para cuadrado y rombo. Aquí es lo estamos trabajando por medio de triángulos, de manera que al unirlos podemos generar nuestras dos guras y solo ajustamos su posición.

```
184         // Dibujar formas
185         glUseProgram(program);
186
187         // Cuadrado (blanco)
188         glUniform3f(uColor, 1.0f, 1.0f, 1.0f);
189         glBindVertexArray(vaoSquare);
190         glDrawArrays(GL_TRIANGLES, 0, (GLsizei)(vSquare.size() / 2));
191         glBindVertexArray(0);
192
193         // Rombo (amarillo)
194         glUniform3f(uColor, 1.0f, 1.0f, 0.0f);
195         glBindVertexArray(vaoDiamond);
196         glDrawArrays(GL_TRIANGLES, 0, (GLsizei)(vDiamond.size() / 2));
197         glBindVertexArray(0);
198
199         glfwSwapBuffers(mainWindow);
```

Regresando a nuestro loop, mandamos a llamar a nuestras figuras que para fines fáciles de ubicar se dejaron en color blanco. Cada uno se forma de lado derecho e izquierdo para que sea fácil de apreciar

Ejecución



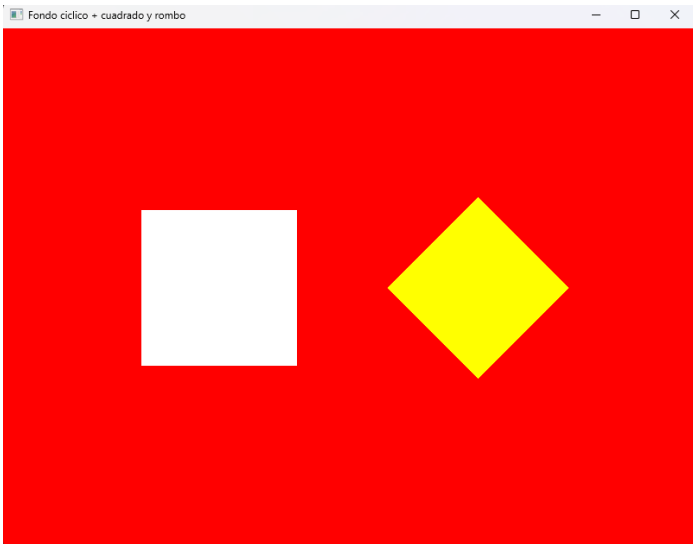


1. **Problemas presentados. Listar si surgieron problemas a la hora de ejecutar el código**

Tuve problemas para hacer que pareciera mas un cuadrado que un rectángulo y vi que esto influye por los shaders o por como manejamos la creación de la ventana así que opte por la siguiente solución:

```
120     //glViewport(0, 0, BufferWidth, BufferHeight);
121     int side = (BufferWidth < BufferHeight) ? BufferWidth : BufferHeight;
122     int x = (BufferWidth - side) / 2;
123     int y = (BufferHeight - side) / 2;
124     glViewport(x, y, side, side);
```

Con esta modificación al viewport, aseguramos que aunque la ventana sea 800x600, el área de dibujo será un cuadrado centrado, evitando que el "cuadrado" se vea aplastado. Y la ejecución seria de la siguiente manera:



Conclusión

Al ser un primer acercamiento al mundo de OpenGL pude comprender las geometrías, un mediano acercamiento a cómo trabajan los shaders y lo que mas se me dificulto fue la parte del loop, pues trabajar con las dimensiones de las figuras y hacer que se realizaran esos pequeños cambios en ejecución los fui ajustando al tanteo hasta conseguir que se mostrara correctamente. Nota: siento que, como buena introducción a esto, vendría bien un tutorial (ya que no pude estar en las primeras clases).