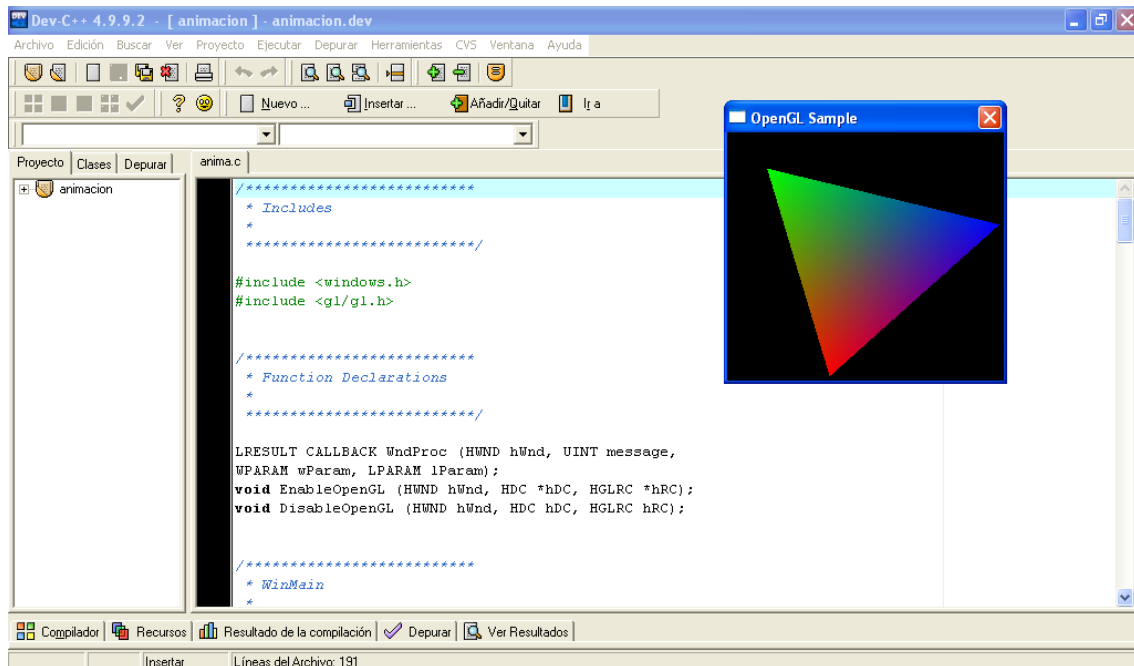


Nombre: León Arriaga Bryan Isaac.

Programación Gráfica en C.

Abrimos Dev C, y a continuación en el menú de archivos, damos abrir > nuevo > Proyecto y en vez de abrir consola de aplicación, abrimos multimedia.

Damos la aplicación de guardar el archivo y al ejecutar rengria que salri algo como esto:



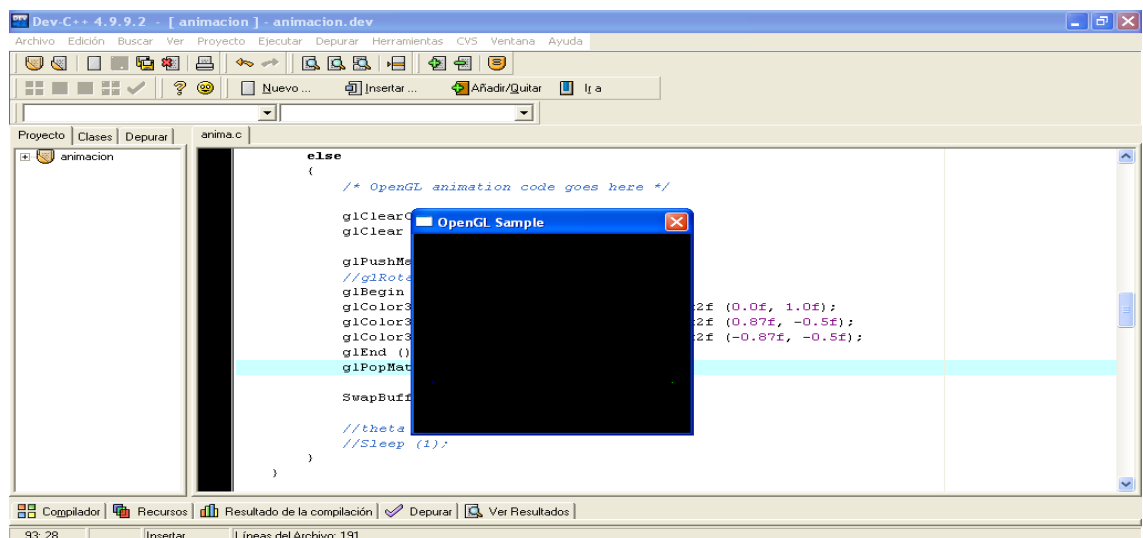
De acuerdo con la modificación de los archivos, es necesario que cambiemos el glBegin. En vez de un triángulo serán tres puntos.

Del segundo else, vamos a cambiar lo siguiente:

```
//glRotatef (theta, 0.Of, 0.Of, 1.Of);  
//theta += 1.Of;  
//Sleep (1);
```

Lo unico que cambió es que le agregamos dos diagonales invertidas. Y el comando glBegin (GL_triangles);→ glBegin (GL_POINTS); Si ejecutamos obtenemos los puntos.

Aunque hagamos una impresión de la pantalla, los puntos no se verán.



Cómo desplazar puntos:

Es necesario agregar una variable y declararla en el programa anterior:

La parte de azul es lo que cambiaremos en el programa:

```
glBegin (GL_POINTS);  
  
x=1;  
  
glColor3f (x, 0.0f, 0.0f);  glVertex2f (0.5f, 0.5f);  
  
glColor3f (0.0f, 1.0f, 0.0f);  glVertex2f (0.87f, -0.5f);  
  
glColor3f (0.0f, 0.0f, 1.0f);  glVertex2f (-0.87f, -0.5f);  
  
glEnd ();
```

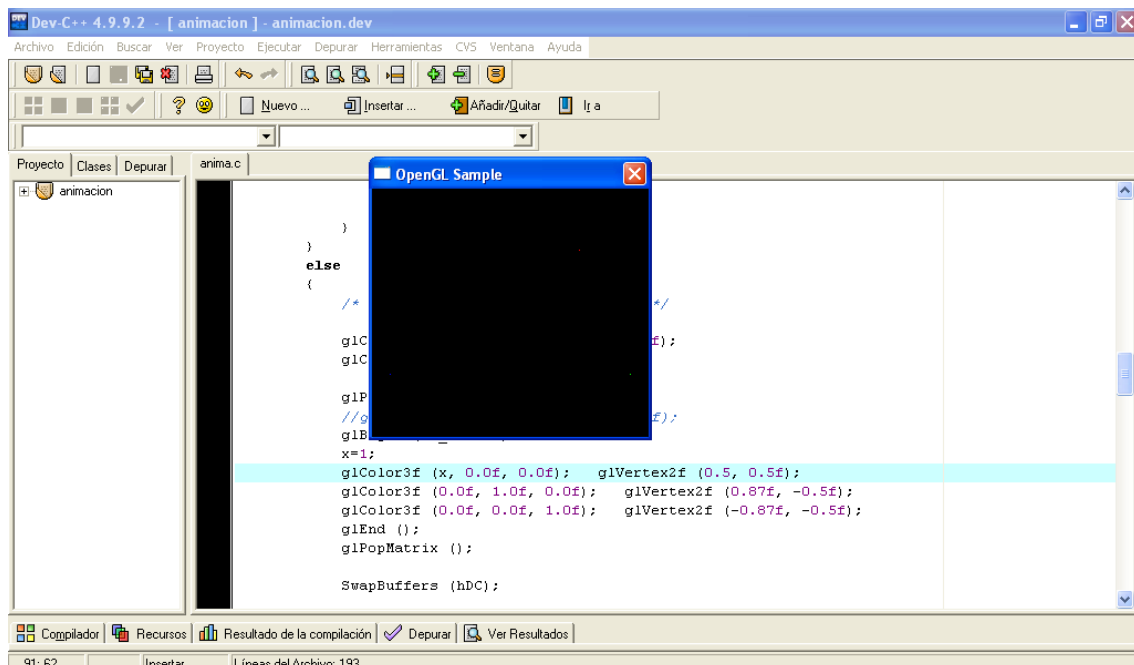
Y en la declaración de variables, esta es la parte que cambia:

```
BOOL bQuit = FALSE;
```

```
float theta = 0.0f;
```

```
int x;
```

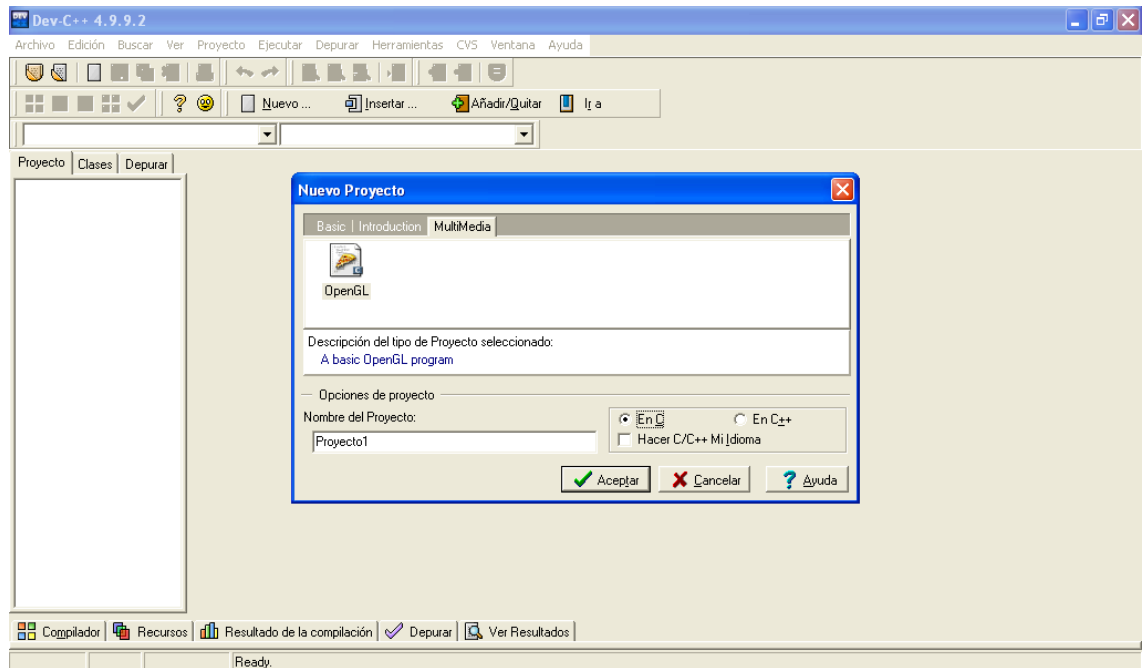
Compilamos y ejecutamos:



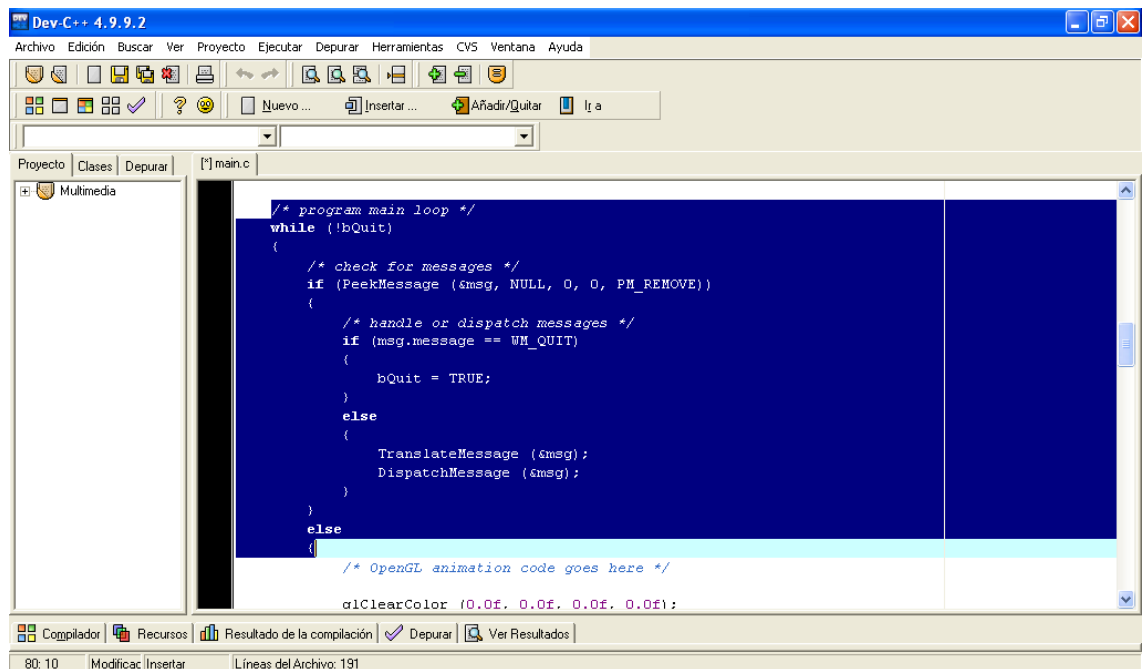
Lamentablemente no se alcanzan a ver los puntos. Más adelante trabajaremos con más colores y con otro tipo de formas.

Desde aquí comienza la verdadera tarea.

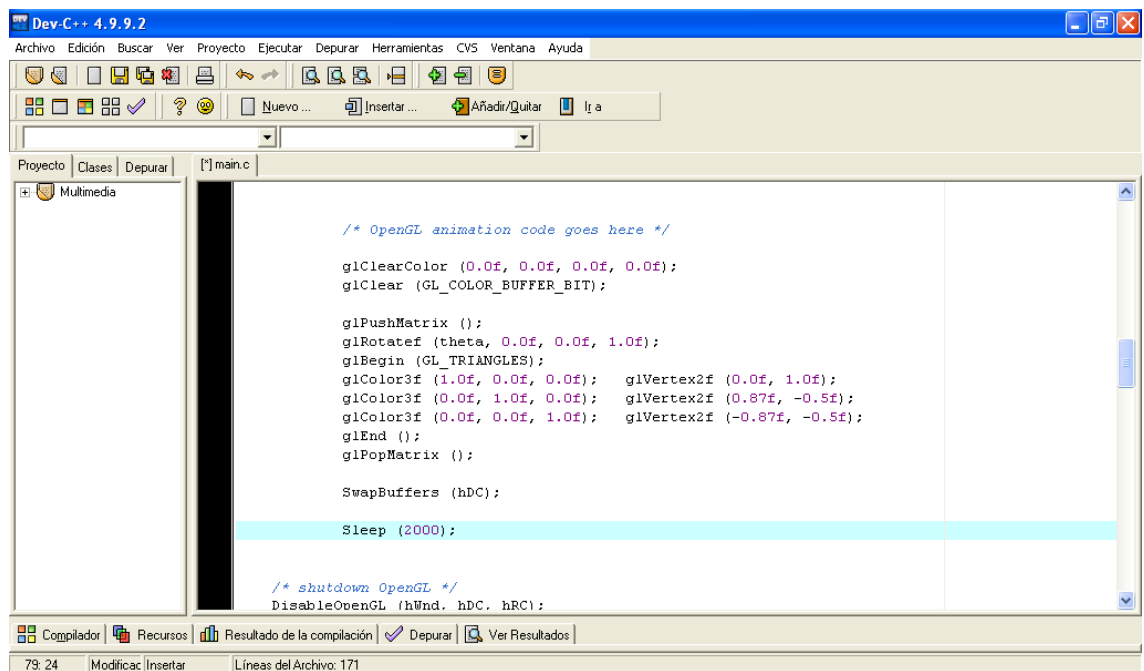
1. Abrimos un nuevo programa un la parte multimedia con lenguaje C.



2. Al aparecer el código, es necesario quitar el *min lop* hasta el segundo else del código inicial y llave de inicio y lsa dos llaves finales.



3. Quitamos el theta y arreglamos el sleep a(2000)



```
/* OpenGL animation code goes here */

glClearColor (0.0f, 0.0f, 0.0f, 0.0f);
glClear (GL_COLOR_BUFFER_BIT);

glPushMatrix ();
glRotatef (theta, 0.0f, 0.0f, 1.0f);
glBegin (GL_TRIANGLES);
glColor3f (1.0f, 0.0f, 0.0f);   glVertex2f (0.0f, 1.0f);
glColor3f (0.0f, 1.0f, 0.0f);   glVertex2f (0.87f, -0.5f);
glColor3f (0.0f, 0.0f, 1.0f);   glVertex2f (-0.87f, -0.5f);
glEnd ();
glPopMatrix ();

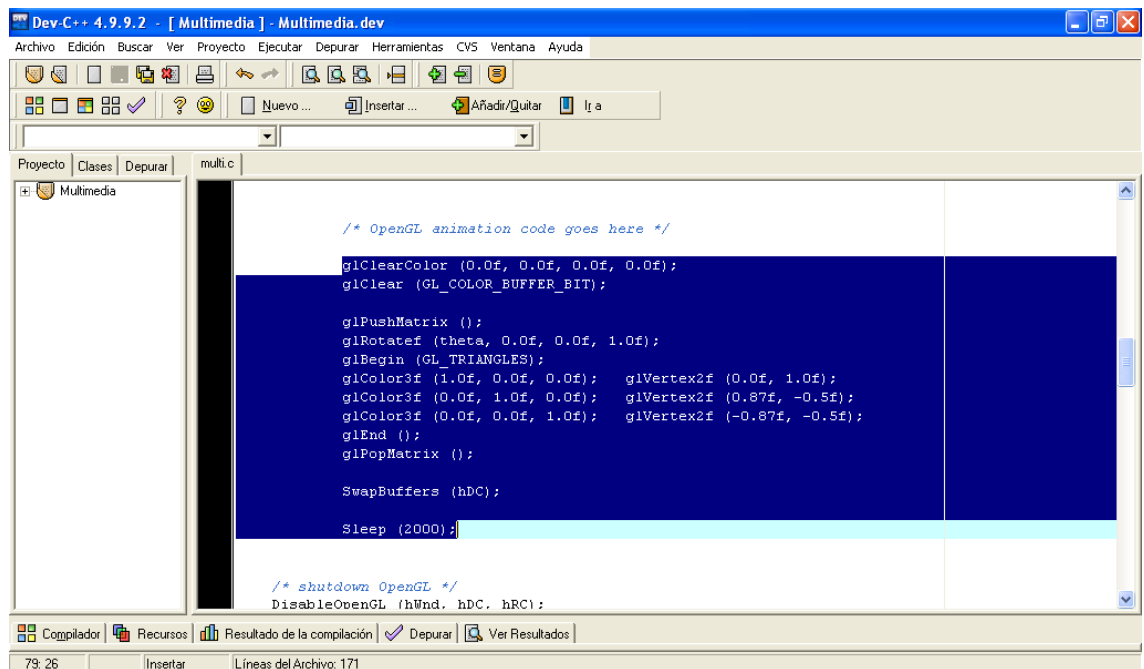
SwapBuffers (hDC);

Sleep (2000);

/* shutdown OpenGL */
DisableOpenGL (hWnd, hDC, hRC);
```

4. Se guarda el programa y se corre. Si aparece un triángulo un rato esta bien.

5. Solo se trabajara con esta parte del código.



```
/* OpenGL animation code goes here */

glClearColor (0.0f, 0.0f, 0.0f, 0.0f);
glClear (GL_COLOR_BUFFER_BIT);

glPushMatrix ();
glRotatef (theta, 0.0f, 0.0f, 1.0f);
glBegin (GL_TRIANGLES);
glColor3f (1.0f, 0.0f, 0.0f);   glVertex2f (0.0f, 1.0f);
glColor3f (0.0f, 1.0f, 0.0f);   glVertex2f (0.87f, -0.5f);
glColor3f (0.0f, 0.0f, 1.0f);   glVertex2f (-0.87f, -0.5f);
glEnd ();
glPopMatrix ();

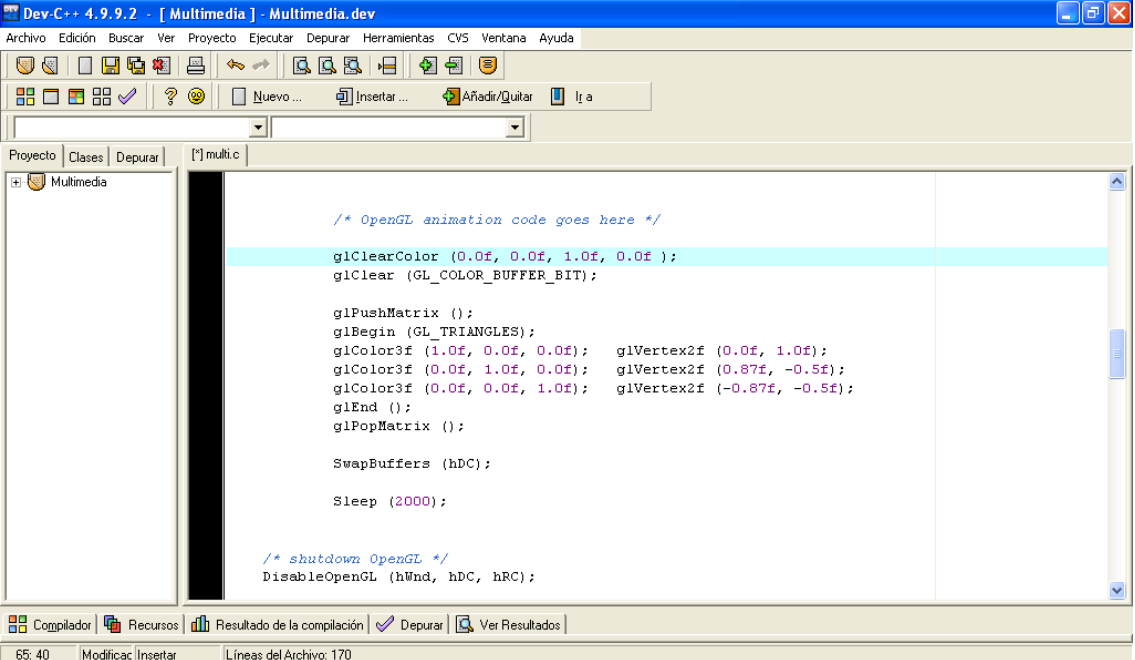
SwapBuffers (hDC);

Sleep (2000);

/* shutdown OpenGL */
DisableOpenGL (hWnd, hDC, hRC);
```

6. Debemos quitar el `glRotat`. Sirve para rotar.
7. La parte que dice `GLClearColor` sirve para cambiar el fondo. Contienen cuatro argumentos. Los primeros tres son el color. Por ahora es recomendable que el cuarto lo dejemos en cero.

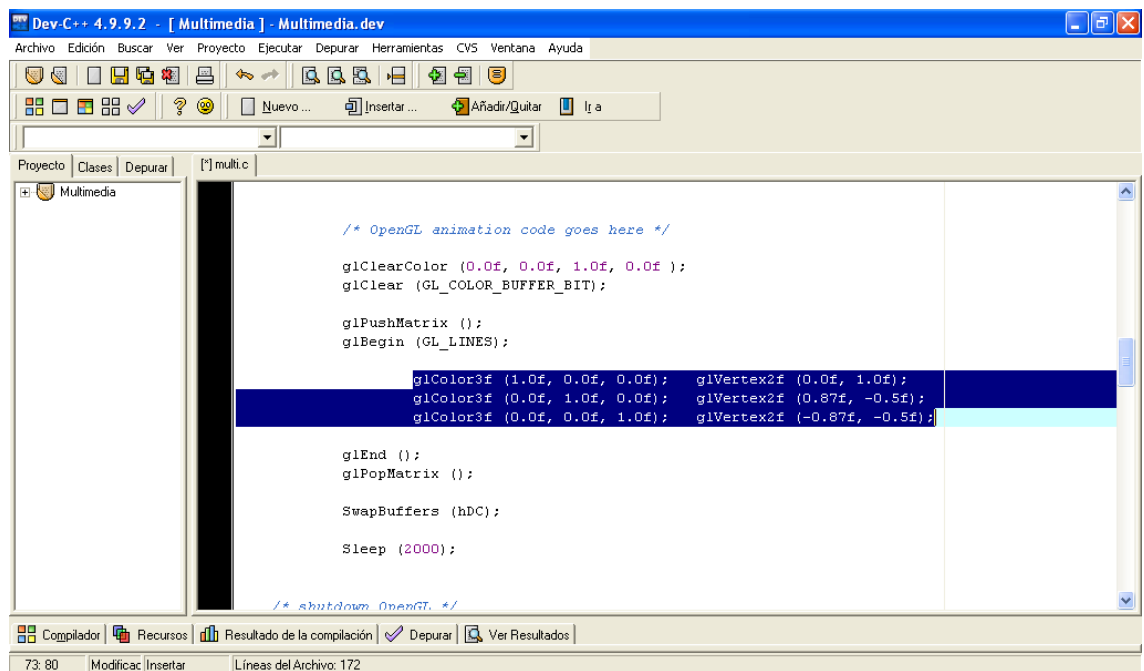
Los colores se clasifican en Red, Green, Blue. Es decir que si queremos una pantalla de cierto color, solo llenamos el espacio con un numero uno. Si queremos una pantalla azul deberá de verse de la siguiente manera:



```
/* OpenGL animation code goes here */  
  
glClearColor (0.0f, 0.0f, 1.0f, 0.0f );  
glClear (GL_COLOR_BUFFER_BIT);  
  
glPushMatrix ();  
glBegin (GL_TRIANGLES);  
glColor3f (1.0f, 0.0f, 0.0f);   glVertex2f (0.0f, 1.0f);  
glColor3f (0.0f, 1.0f, 0.0f);   glVertex2f (0.87f, -0.5f);  
glColor3f (0.0f, 0.0f, 1.0f);   glVertex2f (-0.87f, -0.5f);  
glEnd ();  
glPopMatrix ();  
  
SwapBuffers (hDC);  
  
Sleep (2000);  
  
/* shutdown OpenGL */  
DisableOpenGL (hWnd, hDC, hRC);
```

8. El comando `GLClear`, permite borrar el buffer de la memoria. Se deja tal cual.
9. El comando `Glpush` y `GLpop` trabajan en par. Y siempre se configura dentro de ellos.
10. Los comandos `Glbegin` y `Glend` siempre van juntos. En el `glbegin` es necesario escribir el argumento que queremos que la animación tenga. En este caso escribiremos, `LINES`.

11. Los comandos que vienen en columnas:



```
/* OpenGL animation code goes here */

glClearColor (0.0f, 0.0f, 1.0f, 0.0f );
glClear (GL_COLOR_BUFFER_BIT);

glPushMatrix ();
glBegin (GL_LINES);

    glColor3f (1.0f, 0.0f, 0.0f);    glVertex2f (0.0f, 1.0f);
    glColor3f (0.0f, 1.0f, 0.0f);    glVertex2f (0.87f, -0.5f);
    glColor3f (0.0f, 0.0f, 1.0f);    glVertex2f (-0.87f, -0.5f);

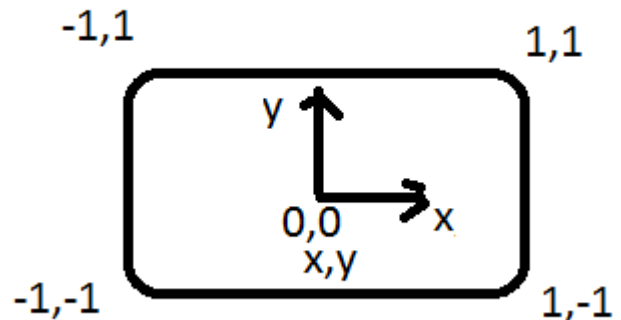
glEnd ();
glPopMatrix ();

SwapBuffers (hDC);

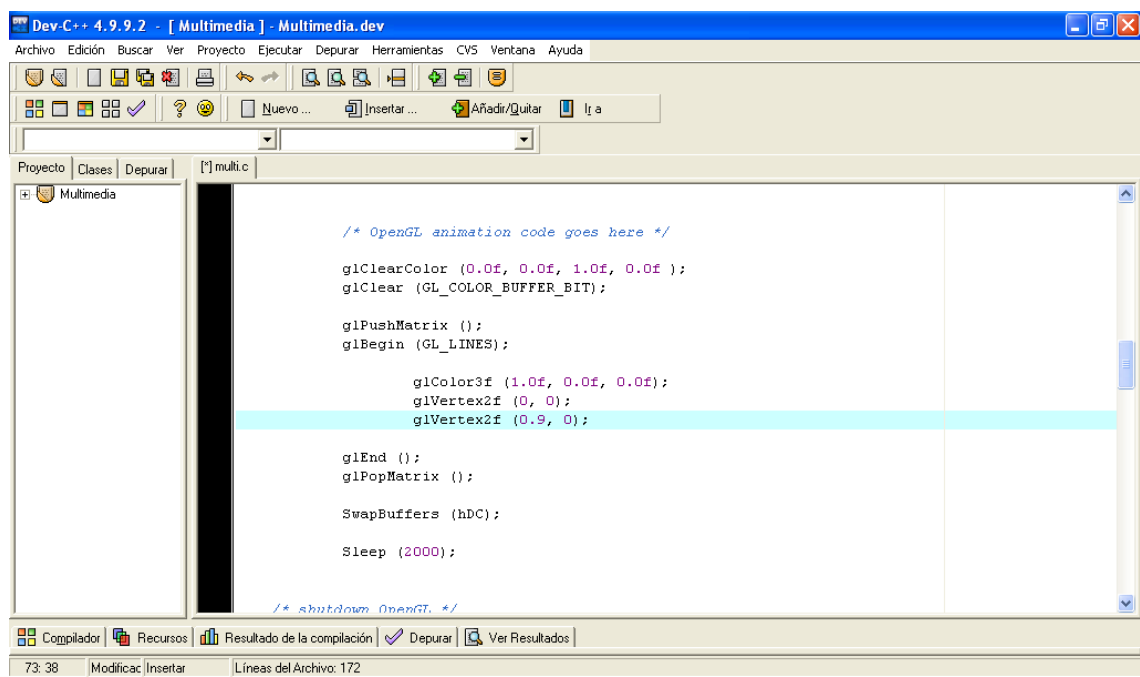
Sleep (2000);

/* shutdown OpenGL. */
```

La primera línea pertenece al color que la línea tiene. El formato de colores de fondo de pantalla es el mismo aquí. La segunda columna pertenece a los vértices que forman la línea. Su argumento pertenece al comando (x,y). Este punto es el punto inicial de una línea. Este es el espacio de trabajo.



12. Vamos a escribir una línea de la siguiente manera:



```
/* OpenGL animation code goes here */

glClearColor (0.0f, 0.0f, 1.0f, 0.0f );
glClear (GL_COLOR_BUFFER_BIT);

glPushMatrix ();
glBegin (GL_LINES);

    glColor3f (1.0f, 0.0f, 0.0f);
    glVertex2f (0, 0);
    glVertex2f (0.9, 0);

glEnd ();
glPopMatrix ();

SwapBuffers (hDC);

Sleep (2000);

/* shutdown OpenGL. */
```

Con esa información, podemos hacer la tarea.