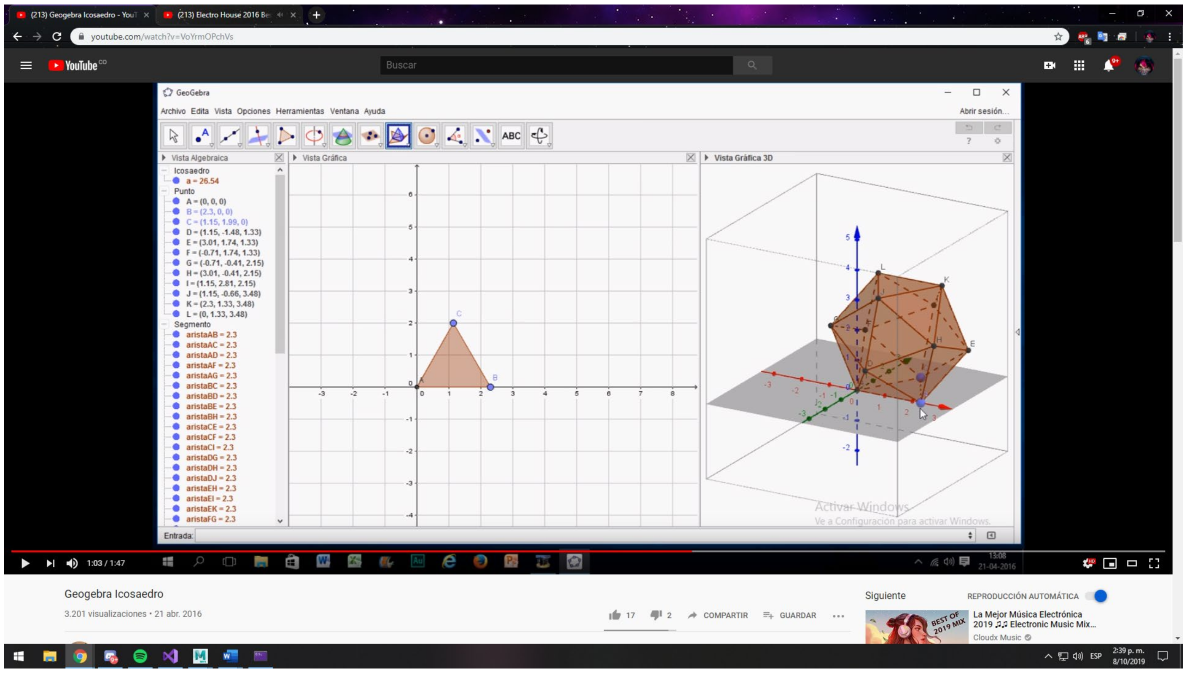
```
// Include standard headers
#include <stdio.h>
#include <stdlib.h>

// Include GLEW
#include <GL/glew.h>

// Include GLFW
#include <GLFW/glfw3.h>
GLFWwindow* window;

// Include GLM
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
using namespace glm;

#include <common/shader.hpp>

int main( void )
{
	// Initialise GLFW
	if( !glfwInit() )
	{
		fprintf( stderr, "Failed to initialize GLFW\n" );
		getchar();
		return -1;
	}

	glfwWindowHint(GLFW_SAMPLES, 4);
	glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
	glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
	glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE); // To make MacOS happy;
should not be needed
	glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);

	// Open a window and create its OpenGL context
	window = glfwCreateWindow( 1024, 768, "Tutorial 04 - Colored Cube", NULL,
NULL);
	if( window == NULL ){
		fprintf( stderr, "Failed to open GLFW window. If you have an Intel GPU,
they are not 3.3 compatible. Try the 2.1 version of the tutorials.\n" );
		getchar();
		glfwTerminate();
		return -1;
	}
	glfwMakeContextCurrent(window);

	// Initialize GLEW
	glewExperimental = true; // Needed for core profile
	if (glewInit() != GLEW_OK) {
		fprintf(stderr, "Failed to initialize GLEW\n");
		getchar();
		glfwTerminate();
		return -1;
	}

	// Ensure we can capture the escape key being pressed below
	glfwSetInputMode(window, GLFW_STICKY_KEYS, GL_TRUE);
	// Dark blue background
	glClearColor(0.0f, 0.0f, 0.4f, 0.0f);

	// Enable depth test
	glEnable(GL_DEPTH_TEST);
	// Accept fragment if it closer to the camera than the former one
	glDepthFunc(GL_LESS);

	GLuint VertexArrayID;
	glGenVertexArrays(1, &VertexArrayID);
	glBindVertexArray(VertexArrayID);

	// Create and compile our GLSL program from the shaders
	GLuint programID = LoadShaders( "TransformVertexShader.vertexshader",
"ColorFragmentShader.fragmentshader" );

	// Get a handle for our "MVP" uniform
	GLuint MatrixID = glGetUniformLocation(programID, "MVP");

	// Projection matrix : 45° Field of View, 4:3 ratio, display range : 0.1 unit
<-> 100 units
	glm::mat4 Projection = glm::perspective(glm::radians(45.0f), 4.0f / 3.0f,
0.1f, 100.0f);
	// Camera matrix
	glm::mat4 View       = glm::lookAt(
								glm::vec3(10,3,-3), // Camera
is at (4,3,-3), in World Space
								glm::vec3(0,0,0), // and looks
at the origin
								glm::vec3(0,1,0)  // Head is
up (set to 0,-1,0 to look upside-down)
						   );
	// Model matrix : an identity matrix (model will be at the origin)
	glm::mat4 Model      = glm::mat4(1.0f);
	// Our ModelViewProjection : multiplication of our 3 matrices
	glm::mat4 MVP        = Projection * View * Model; // Remember, matrix
multiplication is the other way around

	// Our vertices. Tree consecutive floats give a 3D vertex; Three consecutive
vertices give a triangle.
	// A cube has 6 faces with 2 triangles each, so this makes 6*2=12 triangles,
and 12*3 vertices
	static const GLfloat g_vertex_buffer_data[] = {

		//punta
		0.0f,1.34f,3.51f,//L
		2.32f,1.34f,3.51f,//K
		1.16f,-0.67f,3.51f,//J

		0.0f,1.34f,3.51f,//L
		2.32f,1.34f,3.51f,//K
		1.16f,2.84f,2.17f,//I

		0.0f,1.34f,3.51f,//L
		1.16f,-0.67f,3.51f,//J
		-0.71f,-0.41f,2.15f,//G
```

```
0.0f,1.34f,3.51f,//L
-0.71f,-0.41f,2.15f,//G
-0.71f,1.74f,1.33f,//F

0.0f,1.34f,3.51f,//L
-0.71f,1.74f,1.33f,//F
1.16f,2.84f,2.17f,//I

//centro

2.32f,1.34f,3.51f,//K
1.16f,-0.67f,3.51f,//J
3.01f,-0.41f,2.15f,//H

2.32f,1.34f,3.51f,//K
3.01f,-0.41f,2.15f,//H
3.01f,1.74f,1.33f,//E

1.16f,-0.67f,3.51f,//J
3.01f,-0.41f,2.15f,//H
1.15f,-1.48f,1.33f,//D

1.16f,-0.67f,3.51f,//J
-0.71f,-0.41f,2.15f,//G
1.15f,-1.48f,1.33f,//D

-0.71f,-0.41f,2.15f,//G
-0.71f,1.74f,1.33f,//F
0.0f,0.0f,0.0f,//A

-0.71f,-0.41f,2.15f,//G
0.0f,0.0f,0.0f,//A
1.15f,-1.48f,1.33f,//D

-0.71f,1.74f,1.33f,//F
0.0f,0.0f,0.0f,//A
1.15f,1.99f,0.0f,//C

-0.71f,1.74f,1.33f,//F
1.16f,2.84f,2.17f,//I
1.15f,1.99f,0.0f,//C

1.16f,2.84f,2.17f,//I
1.15f,1.99f,0.0f,//C
3.01f,1.74f,1.33f,//E


2.32f,1.34f,3.51f,//K
1.16f,2.84f,2.17f,//I
3.01f,1.74f,1.33f,//E


//punta abajo
2.3f,0.0f,0.0f,//B
3.01f,1.74f,1.33f,//E
3.01f,-0.41f,2.15f,//H

2.3f,0.0f,0.0f,//B
```

```
        3.01f,-0.41f,2.15f,//H
        1.15f,-1.48f,1.33f,//D

        2.3f,0.0f,0.0f,//B
        1.15f,-1.48f,1.33f,//D
        0.0f,0.0f,0.0f,//A

        2.3f,0.0f,0.0f,//B
        0.0f,0.0f,0.0f,//A
        1.15f,1.99f,0.0f,//C

        2.3f,0.0f,0.0f,//B
        1.15f,1.99f,0.0f,//C
        3.01f,1.74f,1.33f,//E




};

// One color for each vertex. They were generated randomly.
static const GLfloat g_color_buffer_data[] = {
        0.583f,  0.771f,  0.014f,
        0.609f,  0.115f,  0.436f,
        0.327f,  0.483f,  0.844f,
        0.822f,  0.569f,  0.201f,
        0.435f,  0.602f,  0.223f,
        0.310f,  0.747f,  0.185f,
        0.597f,  0.770f,  0.761f,
        0.559f,  0.436f,  0.730f,
        0.359f,  0.583f,  0.152f,
        0.483f,  0.596f,  0.789f,
        0.559f,  0.861f,  0.639f,
        0.195f,  0.548f,  0.859f,
        0.014f,  0.184f,  0.576f,
        0.771f,  0.328f,  0.970f,
        0.583f,  0.771f,  0.014f,

        0.609f,  0.115f,  0.436f,
        0.327f,  0.483f,  0.844f,
        0.822f,  0.569f,  0.201f,
        0.435f,  0.602f,  0.223f,
        0.310f,  0.747f,  0.185f,
        0.597f,  0.770f,  0.761f,
        0.559f,  0.436f,  0.730f,
        0.359f,  0.583f,  0.152f,
        0.483f,  0.596f,  0.789f,
        0.559f,  0.861f,  0.639f,
        0.195f,  0.548f,  0.859f,
        0.014f,  0.184f,  0.576f,
        0.771f,  0.328f,  0.970f,

        0.583f,  0.771f,  0.014f,
        0.609f,  0.115f,  0.436f,
        0.327f,  0.483f,  0.844f,
        0.822f,  0.569f,  0.201f,
        0.435f,  0.602f,  0.223f,
        0.310f,  0.747f,  0.185f,
			0.597f,  0.770f,  0.761f,
			0.559f,  0.436f,  0.730f,
			0.359f,  0.583f,  0.152f,
			0.483f,  0.596f,  0.789f,
			0.559f,  0.861f,  0.639f,
			0.195f,  0.548f,  0.859f,
			0.014f,  0.184f,  0.576f,
			0.771f,  0.328f,  0.970f,

			0.583f,  0.771f,  0.014f,
			0.609f,  0.115f,  0.436f,
			0.327f,  0.483f,  0.844f,
			0.822f,  0.569f,  0.201f,
			0.435f,  0.602f,  0.223f,
			0.310f,  0.747f,  0.185f,
			0.597f,  0.770f,  0.761f,
			0.559f,  0.436f,  0.730f,
			0.359f,  0.583f,  0.152f,
			0.483f,  0.596f,  0.789f,
			0.559f,  0.861f,  0.639f,
			0.195f,  0.548f,  0.859f,
			0.014f,  0.184f,  0.576f,
			0.771f,  0.328f,  0.970f,
			0.820f,  0.883f,  0.371f,
			0.583f,  0.771f,  0.014f,
			0.609f,  0.115f,  0.436f,
			0.327f,  0.483f,  0.844f,
			0.822f,  0.569f,  0.201f,
		};

		GLuint vertexbuffer;
		glGenBuffers(1, &vertexbuffer);
		glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
		glBufferData(GL_ARRAY_BUFFER, sizeof(g_vertex_buffer_data), 
g_vertex_buffer_data, GL_STATIC_DRAW);

		GLuint colorbuffer;
		glGenBuffers(1, &colorbuffer);
		glBindBuffer(GL_ARRAY_BUFFER, colorbuffer);
		glBufferData(GL_ARRAY_BUFFER, sizeof(g_color_buffer_data), 
g_color_buffer_data, GL_STATIC_DRAW);

		do{

			// Clear the screen
			glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

			// Use our shader
			glUseProgram(programID);

			// Send our transformation to the currently bound shader, 
			// in the "MVP" uniform
			glUniformMatrix4fv(MatrixID, 1, GL_FALSE, &MVP[0][0]);

			// 1rst attribute buffer : vertices
			glEnableVertexAttribArray(0);
			glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
			glVertexAttribPointer(
                    0,                  // attribute. No particular reason for 0,
but must match the layout in the shader.
                    3,                  // size
                    GL_FLOAT,           // type
                    GL_FALSE,           // normalized?
                    0,                  // stride
                    (void*)0            // array buffer offset
              );

              // 2nd attribute buffer : colors
              glEnableVertexAttribArray(1);
              glBindBuffer(GL_ARRAY_BUFFER, colorbuffer);
              glVertexAttribPointer(
                    1,                                // attribute. No particular
reason for 1, but must match the layout in the shader.
                    3,                                // size
                    GL_FLOAT,                         // type
                    GL_FALSE,                         // normalized?
                    0,                                // stride
                    (void*)0                          // array buffer offset
              );

              // Draw the triangle !
              glDrawArrays(GL_TRIANGLES, 0, 20*3); // 12*3 indices starting at 0 ->
12 triangles

              glDisableVertexAttribArray(0);
              glDisableVertexAttribArray(1);

              // Swap buffers
              glfwSwapBuffers(window);
              glfwPollEvents();

       } // Check if the ESC key was pressed or the window was closed
       while( glfwGetKey(window, GLFW_KEY_ESCAPE ) != GLFW_PRESS &&
                 glfwWindowShouldClose(window) == 0 );

       // Cleanup VBO and shader
       glDeleteBuffers(1, &vertexbuffer);
       glDeleteBuffers(1, &colorbuffer);
       glDeleteProgram(programID);
       glDeleteVertexArrays(1, &VertexArrayID);

       // Close OpenGL window and terminate GLFW
       glfwTerminate();

       return 0;
}
```



(213) Geogebra Icosaedro - YouT
(213) Electro House 2016 Be
youtube.com/watch?v=VoYrmOPchVs
YouTube
Buscar
GeoGebra
Archivo Edita Vista Opciones Herramientas Ventana Ayuda
Abrir sesión...
Vista Algebraica
Vista Gráfica
Vista Gráfica 3D
Entrada:
1:03 / 1:47
Activar Windows
Ve a Configuración para activar Windows.
Siguiente
REPRODUCCIÓN AUTOMÁTICA
17
2
COMPARTIR
GUARDAR
La Mejor Música Electrónica 2019 ♫♫ Electronic Music Mix...
Cloudx Music
BEST OF 2019 MIX
2:39 p. m.
8/10/2019
ESP

Icosaedro
- a = 26.54

Punto
- A = (0, 0, 0)
- B = (2.3, 0, 0)
- C = (1.15, 1.99, 0)
- D = (1.15, -1.48, 1.33)
- E = (3.01, 1.74, 1.33)
- F = (-0.71, 1.74, 1.33)
- G = (-0.71, -0.41, 2.15)
- H = (3.01, -0.41, 2.15)
- I = (1.15, 2.81, 2.15)
- J = (1.15, -0.66, 3.48)
- K = (2.3, 1.33, 3.48)
- L = (0, 1.33, 3.48)

Segmento
- aristaAB = 2.3
- aristaAC = 2.3
- aristaAD = 2.3
- aristaAF = 2.3
- aristaAG = 2.3
- aristaBC = 2.3
- aristaBD = 2.3
- aristaBE = 2.3
- aristaBH = 2.3
- aristaCE = 2.3
- aristaCF = 2.3
- aristaCI = 2.3
- aristaDG = 2.3
- aristaDH = 2.3
- aristaDJ = 2.3
- aristaEH = 2.3
- aristaEI = 2.3
- aristaEK = 2.3
- aristaFG = 2.3

Geogebra Icosaedro

3.201 visualizaciones • 21 abr. 2016