```c
/* tutoriel 001 (compact, structuré)
 * ouvrir une fenetre
 * - http://www.opengl-tutorial.org/beginners-tutorials/tutorial-1-opening-a-window/
 * - http://www.glfw.org/docs/latest/
 * - https://www.opengl.org/sdk/docs
 * gcc code001_s.c glmath.c -o code001 -lGL -lGLEW -lglfw
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <GL/glew.h>
#include <GLFW/glfw3.h>

#include "glutils.h"
#include "glmath.h"


// ----------------------------------------------------------------------

struct Application
{
    GLuint program_id;
    GLuint vertex_array_id;
    GLuint vertex_buffer_id;
    mat4 mvp_matrix;
    GLuint mvp_matrix_id;
    GLFWwindow* window;
} app;

// ----------------------------------------------------------------------

int init_window()
{
    if ( !glfwInit() ) return 1;

    glfwWindowHint(GLFW_SAMPLES, 4);
    glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
    glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);

    app.window = glfwCreateWindow( 800, 450, "code 003", NULL, NULL);
    if ( app.window == NULL )
    {
        glfwTerminate();
        return 2;
    }

    glfwMakeContextCurrent(app.window);
    glewExperimental = 1;
    if (glewInit() != GLEW_OK)
    {
        glfwTerminate();
        return 3;
    }

    return 0;
}

int init_app()
{
    char vertex_shader[] =
        "#version 330 core                                          \n"
        "layout(location = 0) in vec3 vertexPosition_modelspace;  \n"
        "uniform mat4 MVP;                                          \n"
        "void main()                                                \n"
        "{                                                          \n"
        "  gl_Position = MVP * vec4(vertexPosition_modelspace,1); \n"
        "}                                                          \n";
    char fragment_shader[] =
        "#version 330 core                                          \n"
        "out vec3 color;                                            \n"
        "void main()                                                \n"
        "{                                                          \n"
        "  color = vec3(1,0,0);                                     \n"
        "}                                                          \n";
    app.program_id = load_shaders( vertex_shader, fragment_shader );
    if (app.program_id == 0) return 1;

```

```
    app.mvp_matrix_id = glGetUniformLocation(app.program_id, "MVP");

    mat4 projection_matrix =
        mat4_perspective(45., 800./450., .1, 100.);
    mat4 view_matrix =
        mat4_lookAt(vec3_init(4,3,3), vec3_init(0,0,0), vec3_init(0,1,0));
    mat4 model_matrix =
        mat4_identity();
    app.mvp_matrix =
        mat4_product(
            mat4_product(projection_matrix, view_matrix),
            model_matrix
        );

    glGenVertexArrays(1, &app.vertex_array_id);
    glBindVertexArray(app.vertex_array_id);

    static const GLfloat g_vertex_buffer_data[] =
    {
        -1.0f, -1.0f, 0.0f,
        1.0f, -1.0f, 0.0f,
        0.0f,  1.0f, 0.0f,
    };

    glGenBuffers(1, &app.vertex_buffer_id);
    glBindBuffer(GL_ARRAY_BUFFER, app.vertex_buffer_id);
    glBufferData(
        GL_ARRAY_BUFFER,
        sizeof(g_vertex_buffer_data),
                g_vertex_buffer_data,
                GL_STATIC_DRAW
               );

    return 0;
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glUseProgram(app.program_id);

    glUniformMatrix4fv(
        app.mvp_matrix_id,
        1,
        GL_FALSE,
        &app.mvp_matrix.matrix[0][0]
    );

    glEnableVertexAttribArray(0);
    glBindBuffer(GL_ARRAY_BUFFER, app.vertex_buffer_id);
    glVertexAttribPointer(
        0,
        3,
        GL_FLOAT,
        GL_FALSE,
        0,
        (void*)0
    );
    // Draw the triangle !
    glDrawArrays(GL_TRIANGLES, 0, 3);
    glDisableVertexAttribArray(0);
}

void mainloop()
{
    glClearColor(0.0f, 0.0f, 0.4f, 0.0f);

    glfwSetInputMode(app.window, GLFW_STICKY_KEYS, GL_TRUE);

    do
    {
        display();
        glfwSwapBuffers(app.window);
        glfwPollEvents();
    }
    while ( glfwGetKey(app.window, GLFW_KEY_ESCAPE ) != GLFW_PRESS
        && !glfwWindowShouldClose(app.window)
    );
}

```

```
159 int main(void)
160 {
161     if ( init_window() != 0 || init_app() != 0 ) return 1;
162 
163     mainloop();
164 
165     glDeleteBuffers(1, &app.vertex_buffer_id);
166     glDeleteProgram(app.program_id);
167     glDeleteVertexArrays(1, &app.vertex_array_id);
168 
169 
170     glfwTerminate();
171 
172     // that's all folks!
173     return 0;
174 }
175 
176 
177 
```