Computer Graphics Exercise

October 3, 2024

1. Transform the TriangleOpengl project to output a triangle with a color gradient, creating a shaded effect. To achieve this, assign a different color to each vertex of the triangle, and the rendering pipeline, specifically the rasterizer subsystem, will linearly interpolate the colors between these vertices for each pixel inside the triangle.

2. Organize the project into independent modules, each contained in a .h and .cpp file. The main file will be responsible only for initializing the GLFW and GLAD libraries, and the game loop, while the modules will handle the initialization of graphic resources (shaders, VAO, geometries) and user interactions, promoting reusability in other projects.

3. For flexible geometry representation, we'll employ std::vector of glm::vec3 to store vertex positions. This enables us to create geometries of arbitrary shapes and sizes. The VAO management code will be adjusted to directly load data from these vectors. To utilize these features, ensure that you include the following headers in your project: #include <vector> and #include <glm/glm.hpp>

4. Create and render complex geometries such as a circle, a butterfly, and a heart through the definition of parametric curves. It is suggested to define a structure:

```
typedef struct {
  GLuint VAO; // VAO identifier
  GLuint VBO_G; // VBO identifier for vertex geometry
  GLuint VBO_C; // VBO identifier for vertex colors
  int nTriangles; // number of triangles that make up the geometry
  // Vertices and attributes
  vector<vec3> vertices; // structure containing vertex coordinates
  vector<vec4> colors; // structure containing vertex colors
  // Number of vertices
  int nv;
  int render;
} Figura;
```