Exercise 1: Buffers and Meshs

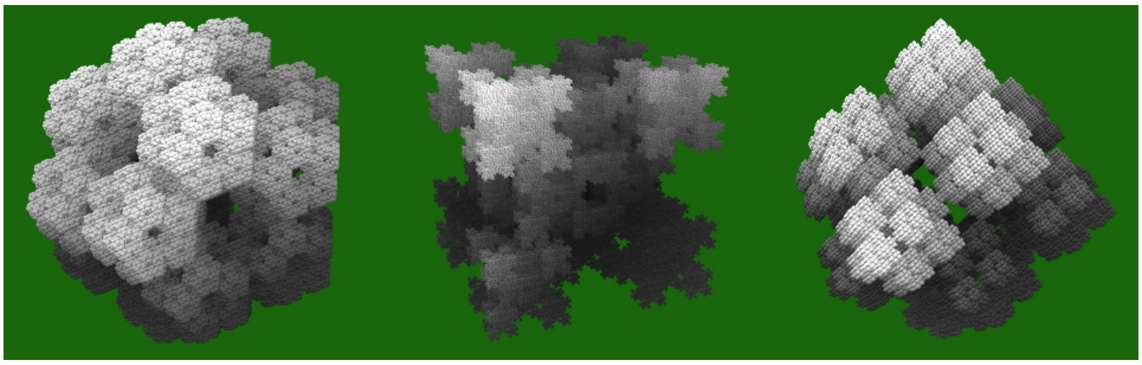
Figure 1: Dodecahedron fractal which is drawn using vertex and index buffers. Changing a parameter in the Dodecahedron vertex definition results in various shapes: regular dodecahedron, endo dodecahedron and rhombic dodecahedron

In this and most following exercises there is a demo program and a framework. The framework will be the same all the time and is going to grow from task to task.

Do not expect that the exercise sheets are fully self contained. You will find many useful informations in the source code and header files, including `TODO: ...` comments which mark all places where something must be added.

Task 1: Implement the Buffer Class

Buffers are one of two different resources in graphics programming (the other one are textures). With buffers you can do many different things. Among them, they are used to define triangle meshes (vertices and indices) and deliver parameters to shader programs (uniforms).

In the framework ***buffer.cpp/.h*** contains a class to handle this buffers. Fill in the TODOs of buffer creation and data upload, as well as the different bindings.

Task 2: Vertex Format

In order to use buffers for vertices their format must be described. A vertex format defines which parts of a buffer are used for which input to the shader and how data is interpreted. There is a vertex format class in ***vertexformat.cpp/.h*** which needs to be completed.

Note: The OpenGL object which contains the vertex format is called Vertex Array Object (VAO). They can be used in two different ways. In the old usage they also contained the buffer bindings itself. They became independent of the buffer binding with ARB_vertex_attrib_binding which is core since 4.3.

Task 3: Fractal Rendering

At the current state the window shows nothing than a green background. The fractal is already generated, but not submitted into buffer objects. Create three buffers for the fractal (positions, normals and indices) and a uniform buffer for the transformation matrices. Finally, establish the binding in the main-loop and make the draw call. Then, the leftmost fractal from Figure 1 should appear.

Note: To get the other exiting shapes have a look in `fractalgen.cpp`

Theory

Be prepared to answer the following questions.

Q1 What is a buffer resource?

Q2 How is the vertex format working?

Q3 What is the index buffer?

Q4 What is an uniform buffer?