Winter 2018: CSI4130
Assignment 2

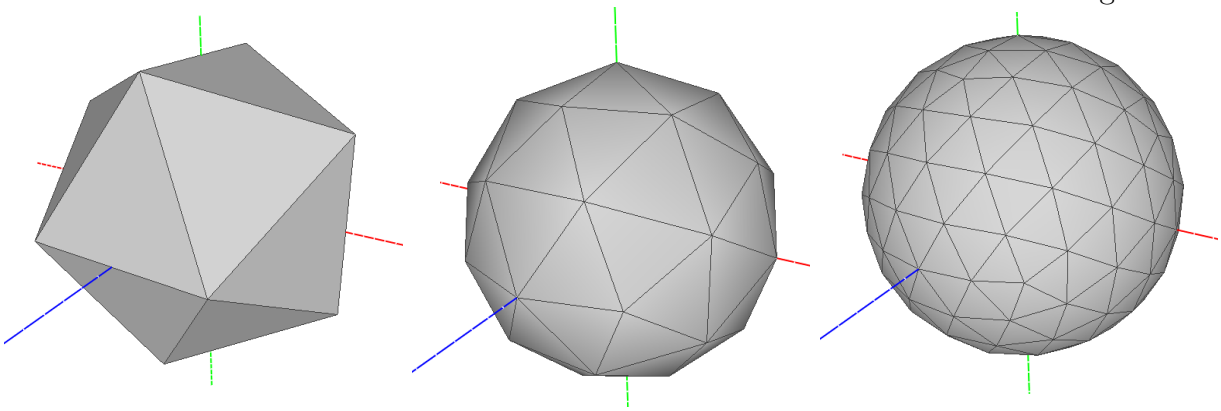
Due: Wednesday, March 2^{nd}, 2018, 11:00 pm on Virtual Campus
University of Ottawa - Université d'Ottawa

Jochen Lang

1 Curves and Surfaces [10 in total]

This assignment is based on a modified solar laboratory (lab03/lab04) which can be downloaded with this assignment. You should use this code as a start either in the C++/GLUT version or in the Java/JOGL version. The modification is in the reduction of the number of vertices for the sphere and a slight change in shape class hierarchy to make things easier to implement for this assignment.

You will need to find a better way to draw the primitives. The lab used fixed size tesselations (surface meshes of a given size). We would like to have a sphere and a torus primitive that allows us to draw these with a different number of vertices. We will use two different strategies.



1.1 Icosahedron as Sphere Approximation

The sphere primitive is currently drawn as an icosahedron approximating the sphere. We can make this approximation more accurate by a displaced subdivision process. We will borrow some ideas from Loop subdivision. Loop subdivision splits triangular faces into 4 new faces with each subdivision. Here, we can simply find the new center position as the average of the three vertices and then displace the center.

1.1.1 One level of subdivision [3]

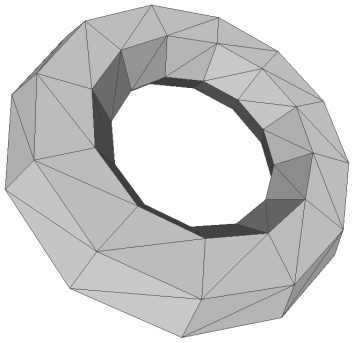
Calculate the new center vertices for all faces and replace each face with the four new faces. Each vertex will have to be placed on the sphere, i.e., after calculating the center of the face, normalize the coordinate for a length of 1.

1.1.2 Any level of subdivision [2]

Your sphere primitive should accept as a parameter the subdivision level. Level 0 is no subdivision, i.e., the icosahedron, level 1 is your solution to Question 1.1.1, level 2 is the algorithm of Question 1.1.1 applied on the result again.

1.2 Torus

For the torus, we use a different strategy by using a definition of the torus in parametric form. We will be using the notion that a torus can be thought of as a set of disks with their center on another disk. A tesselation can be obtained by placing a vertex at fixed angular intervals and then forming quads with neighboring vertices.



1.2.1 Parameterized torus [5]

A parameterized description of a torus is

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f(\alpha,\beta) \\ g(\alpha,\beta) \\ h(\alpha,\beta) \end{bmatrix} = \begin{bmatrix} (r_o + r_i cos(\alpha))cos(\beta) \\ (r_o + r_i cos(\alpha))sin(\beta) \\ r_i sin(\alpha) \end{bmatrix}. \tag{1}$$

The radius r_o is the radius of the circle on which the disk centers are placed. The radius of the disks is r_i. The angle α determines the position on the inner disk while β determines the position on the outer disk.

Your parameters for the torus shape must be r_i, r_o, and the integer number of division for the inner n_i and outer disk n_o. Then $\alpha = i\frac{2\phi}{n_i}$ where $i = 0 \ldots n_i - 1$ and β is calculated correspondingly. As a result, you can form a quad betwen neighboring vertices. For example, with $n_i = 6$ and $n_o = 12$, the vertices $(i = 0, o = 0)$, $(i = 1, o = 0)$, $(i = 0, o = 1)$ and $(i = 1, o = 1)$ form a quad. This quad can be split along the diagonal into two triangles.

1.2.2 Triangle strips [3] (Bonus)

As a bonus question (the bonus only counts towards the assignments), turn the triangles into (multiple) triangle strips and provide code to render the torus with `GL_TRIANGLE_STRIP`.

2 Submission

Feel free to add additional source files as required but please do not submit project definitions or other IDE files to Virtual Campus. The files that you submit have to be sufficient for your program.

2.1 C++ Project

Your assignment submission must consist of a *zip archive*. The following is a (partial) list of files[1]
Do not introduce any non-standard C/C++ features (no windows includes!).

Filename	
`common/shader.h`	Loading and compiling shaders.
`common/shader.cpp`	
`assign2/buttons.h`	The shape helper class (similar to basic_shape)
`assign2/buttons.cpp`	
`assign2/screensaver.cpp`	The top-level glut main program (similar to interaction).
`assign1/buttons.vs`	Vertex shader
`assign1/buttons.fs`	Fragment shader

2.2 Java JOGL Project

In case of JOGL, I would like to receive your assignment in a zip archive with the following layout.

Filename	
`Shader/src/shader/Shader.java`	Loading and compiling shaders.
`Assign1/src/buttons/Buttonshape.java`	The shape helper class (similar to basic_shape).
`Assign1/src/buttons/Buttons.java`	Top-level GLCanvas (similar to Interaction.java).
`assign1/src/starshape/Main.java`	Top-level window and main.
`Assign1/shader/buttons.vs`	Vertex shader
`Assign1/shader/buttons.fs`	Fragment shader

[1] I use the forward slash for directories but windows use is fine.