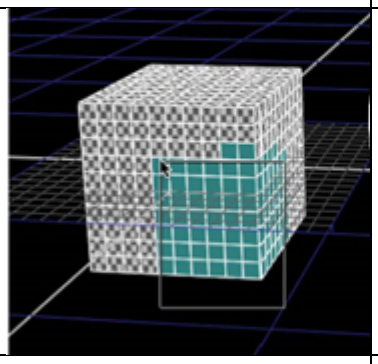
CS3242 Modeling Lab Assignment Summary

Name:__Bastian Morath_____ Student No: A0195628N_____

Tasks	*Main*	*Optional*
Computing Normal Vectors (main, 20 marks) Compute all normal vectors of each triangle and store them into `nlist` with the same index. Uncomment the code "`glNormal3dv`" in the `draw()` function so that your mesh will be drawn with correct shading. The normal computations should be done in the function `readfile()`.	20	
Compute Angle Statistics (main, 30 marks) For each triangle, it has a maximum angle and a minimum angle. This task is to record the frequencies of them into the two arrays `statMaxAngle` and `statMinAngle`. Each of the array has 18 buckets for angles from 0 to 180 degrees, i.e. each bucket has a 10-degree range. For example, if there are three triangles with minimum angles 9, 3 and 55, then `statMinAngle[0] = 2` and `statMinAngle[5] = 1`. The statistic will be printed at the end of `computeStat()`.	30	
Write an OBJ file (main, 20 marks) Just write the current mesh into another obj file. If you press "O", the file main.cpp will call your `writefile()`. You only have to produce the "v" and "f" lines. You should be able to read back your file with `readfile()`.	20	
Read Some Other Type of Files Other Than OBJ (optional, 20 marks) There are other files such as STL, 3DS, etc. Some of them are also quite easy to be read also. Implement some new member functions to read these files. We suggest you to add a new command in the keyboard function in `main.cpp` to call your new reading function(s) to test out this new reading functionality.		20
Implement `enext()`, `sym()` (20 marks, main) As mentioned in the lecture notes, they could be separate functions outside of the class.	20	
Implement `org()`, `dest()` (20 marks, main) As mentioned in the lecture notes, they should be member functions of the class.	20	
Implement `fnext()` (80 marks, main) After reading an OBJ file, link up all the triangles by setting up the `fnlist`. In order to prove your implementation is right, you could implement one of the followings to show that your code works.	80	
Compute the Number of Components (20 marks, optional) An OBJ file may contain a few pieces of surfaces, and the number of pieces is the same as the definition of the number of components in a graph. You may simply achieve this by BFS or DFS like a graph. However, by being in the same component we only mean that their triangles are connected by `fnlist`. Namely, two triangles can be in two different components if they intersect each other but they are not connected through some vertices and edges.		20
Implement `orientTriangles()` (20 marks, optional) Two triangles are in the same orientation if their version 0 normal vectors are in the same side of the surface by the right hand rule. If they are not in the same orientation, you can rectify that by "flipping" one of the triangle. By flipping a triangle, it can be done by exchanging two entries of the vertex indices in the `flist` (and the corresponding entries in the `fnlist`). You can orient all the triangles in the same surface by a BFS/DFS traversal with flipping. However, there is a possibility that meshes are NOT orientable, namely, you cannot flip all the triangles to face the same side of the surface. The simplest example is the		20

Möbius Strip. For such cases, your `orientTriangles()` will return FALSE and leave the mesh “unoriented”.		
Compute Vertex Normal Vectors for Smooth Shading (10 marks, optional) OpenGL can render the Gouraud and Phong shadings by averaged vertex normal vectors. Create a new vector array inside your class and name it `vnlist`. It stores all the vertex normal vectors that is the average of those of surrounding triangles. And add a keyboard function in your `main.cpp` so that you can choose to toggle flat or smooth shadings.		10
Visualize boundary edges (10 marks, optional) Color the edges shared by only exactly one triangle by red.		10
Implementing Selection of Triangle by User Marquee (20 marks, optional) Select a portion of your mesh by dragging a rectangle in your windows. Set up a bit vector (or bitset) `tselect` in your class to record the selected triangles. 		20
Total	190	100

For optional tasks, we expect you only to try %80 of them, so you can omit one task of 20 marks or two tasks of 10 marks.

Final “Quests” (Choose only one of the categories)

1. Subdivision
    - Barycentric/Loop
    - Partial
    - Mesh Relaxation
2. Self-intersection
    - Detection
    - Speed up
    - Object binary operations/Simple CSG
3. Registrations
    - ICP
    - Speedup
    - P2P, P2S, P2?
    - Visualizing error between two models
4. Decimation
    - Cluster/decimation
    - Quality control
    - Progressive mesh
5. Thickening
    - With different cap types
    - Avoid self-intersection
6. Remeshing
7. Inspection
    - Given two similar meshes, calculate and visualize their errors

Final Submission form

Name: _Bastian Morath _____ Student number: _A0195628N_____

Task Checklist

Put a tick in the box if you have finished that task.

Tasks	*Check*	
Computing Normal Vectors (main, 20 marks)	X	
Compute Angle Statistics (main, 30 marks)	X	
Write an OBJ file (main, 20 marks)	X	My method automatically writes either .obj or .off file depending on the filename!
Read Some Other Type of Files Other Than OBJ (optional, 20 marks)	X	I can also read in .off files. Note: .off files index the triangles from 0, while .obj indexes from 1. Took care of that.
Implement `enext()`, `sym()` (20 marks, main)	X	
Implement `org()`, `dest()` (20 marks, main) .	X	
Implement `fnext()` (80 marks, main)	X	
Compute the Number of Components (20 marks, optional)	X	I also created a method that colors the different components with different colors
Implement `orientTriangles()` (20 marks, optional)	X	
Compute Vertex Normal Vectors for Smooth Shading (10 marks, optional)	X	
Visualize boundary edges (10 marks, optional)	X	If edges are drawn, the mesh is not shown, so the edges are easier to see ☺
Implementing Selection of Triangle by User Marquee (20 marks, optional)		

Final Task

Topics: __Barycentric and Loop Subdivision, Coloring of components, and some Keyboard UI for various things (please see below)

What I did in this project, apart from the default main tasks:

- Barycentric subdivision by pressing the ‘B’-key.
- Loop subdivision by pressing the ‘L’-key. The user can choose between two different beta-values when calculating the new even vertices (by pressing key 8 or 9, respectively, before applying the subdivision)
- Read in .off files (only triangles, not arbitrary polygons). The program automatically detects if a .off or a .obj filename is provided.
- Write current object to an .obj or .off file (automatically chooses the file-type depending on the provided filename) by pressing the ‘O’-key and entering a filename
- Compute the number of components. Also, we can color the different components with unique colors (by pressing the ‘C’-key)
- Draw the edges of the object red by pressing the ‘E’-key. The polygons are not shown in this case, so that the edges are easier to see
- Compute Vertex normals
- Iterate through a (predefined) list of object files by pressing any of the keys 1-7