# Computer Graphics 2021
# Preliminary Exercise 2: Reading Meshes from OBJ

**Prof. Dr. Martin Hering-Bertram**
**Hochschule Bremen**

**The problems need to be solved by groups of 2 or 3 students during the laboratory sessions. Every project results in a presentation. The grade is based on these projects, there is no written exam.**

**Laboratory Problem:**
Implement an algorithm in C++ for sorting complex numbers by their absolute values. A complex number is composed of a real part x and an imaginary part y. Its value is

$$z = x + iy, \tag{1}$$

Where $i$ denotes the square root of *-1*. The absolute value is

$$|z| = \sqrt{x^2 + y^2} \tag{2}$$

To solve the problem, you may subdivide it into the following tasks:

1. Use the Priority Queue example as a starting point.
2. Define a class for complex numbers. This may contain methods for input and output to the screen, as well as a comparison operator. You do not need to implement arithmetic operations.
3. Write a testing function and validate the correctness of your approach

Can you accelerate your approach, e.g. getting rid of any square root computation?

**Homework Problems:**

**Problem 1: Reading Points and Triangles from OBJ**
**(a)** Implement a function to read points and triangles from an object file like tetra.obj containing the following data:

```
v  0.0  0.0  0.0
v  8.0  0.0  0.0
v  0.0  8.0  0.0
v  0.0  0.0  8.0
f  1 2 4
f  2 3 4
f  3 1 4
f  3 2 1
```

Here, the keys `v` and `f` denote vertices and faces (triangles and polygons), respectively. A vertex is defined by three floating point numbers denoting its x, y, and z-coordinates. Triangles use three vertex indices for each face, starting with 1 for the first vertex. You can assume for now that all faces are triangles.

For parsing the obj example, you can adapt the following code:

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
#include <vector>

void ReadData( string fname){ //fname = "F:\\CG21\\MeshOpenGL\\mesh1.obj";
    ifstream file( fname);
    if (!file){
        cout << "error opening file" << endl;
        return;
    }
    string key;
    float x, y, z;
    while( file){
        //getline( file, line);
        file >> key >> x >> y >> z;
        cout << key <<", "<< x <<", "<< y <<", "<< z << endl;
    }
    file.close();
}
```

**(b)** Develop a class **Vertex** and a class **Triangle**. Since the number of points and triangles is not known before reading the file, static arrays are not a good choice. Instead, vectors may be used:

```
vector <Vertex> points;
vector <Triangle> tris;
…
points.push_back( …);
```

**Problem 2: Rendering Triangle Meshes**

(a) Replace the function **DrawCylinder()** in **oglwidget.cpp** by one drawing the triangle mesh. It may contain the following code fragments:

```
glBegin( GL_TRIANGLES);
for( i=0; i<tris.size(); i++){
    …
    glNormal3fv( …);
    glVertex3fv( …);
    glVertex3fv( …);
    glVertex3fv( …);
}
glEnd();
```

Before a triangle can be drawn, the three points need to be copied from the vertex array. Note that the OBJ face indices start with 1, i.e. all indices need to be decremented before using them. Also, for every triangle its normal vector needs to be calculated for shading (use flat shading). The normal vector is obtained by the cross product of two triangle edges, for example like this:

```
void cross( float c[3], float a[3], float b[3]){ // c = a cross b
    c[0] = a[1]*b[2] - a[2]*b[1];
    c[1] = a[2]*b[0] - a[0]*b[2];
    c[2] = a[0]*b[1] - a[1]*b[0];
}
```