Mesh Manipulation

Drawable

Steps:

i. Allocate global memory for the model

```
Drawable * model;
```

ii. In create context create a drawable object using one the constructors

- Loading an obj file

```
model = new Drawable("suzanne.obj");
```

- Creating a model by defining the vertices, normals, uvs

```
model = new Drawable(floorVertices , floorUVs , floorNormals );
```

iii. Inside the mainloop

```
model->bind();
model->draw();
```

Use in the vertex shader:

```
layout(location = 0) in vec3 vertexPosition_modelspace;
layout(location = 1) in vec3 vertexNormal_modelspace;
layout(location = 2) in vec2 vertexUV;
```

```
vector<vec3> floorVertices = {
        vec3(-20.0f, -1, -20.0f),
        vec3(-20.0f, -1,  20.0f),
        vec3(20.0f,  -1,  20.0f),
        vec3(20.0f,  -1,  20.0f),
        vec3(20.0f,  -1, -20.0f),
        vec3(-20.0f, -1, -20.0f),

};

vector<vec2> floorUVs = {
        vec2(0.0f, 0.0f),
        vec2(0.0f, 1.0f),
        vec2(1.0f, 1.0f),
        vec2(1.0f, 1.0f),
        vec2(1.0f, 0.0f),
        vec2(0.0f, 0.0f),
};

vector<vec3> floorNormals = {
        vec3(0.0f, 1.0f, 0.0f),
        vec3(0.0f, 1.0f, 0.0f),
        vec3(0.0f, 1.0f, 0.0f),
        vec3(0.0f, 1.0f, 0.0f),
        vec3(0.0f, 1.0f, 0.0f),
        vec3(0.0f, 1.0f, 0.0f)
};
```

Task 1: Scene Creation

1.1 Construct a plane (x-z) using Drawable

1.2 Visualize the plane

1.3 Load the heart model “heart.obj” as Drawable

1.4 Visualize the heart model

Task 2: Translate and rotate the plane using the keyboard

2.1 Translate the plane in the ±y direction using the keyboard (use keys I, K)

2.2 Rotate the plane around the z direction

Tips:

- For the plane translation use the *planeY* variable
- For the plane rotation use the *planeAngle* variable

  Use the aforementioned variables to create the proper model matrix. You will need a translation and a rotation matrix.

- Do not forget to comment out Task 1.3

Task 3: Assign different colors to the model vertices based on their position with respect to the plane (below or above)

The equation of a plane with nonzero normal vector $n = (a, b, c)$ passing through the point $\boldsymbol{x_0} = (x_0, y_0, z_0)$ is

$$n \cdot (x - \boldsymbol{x_0}) = 0 \quad (1)$$

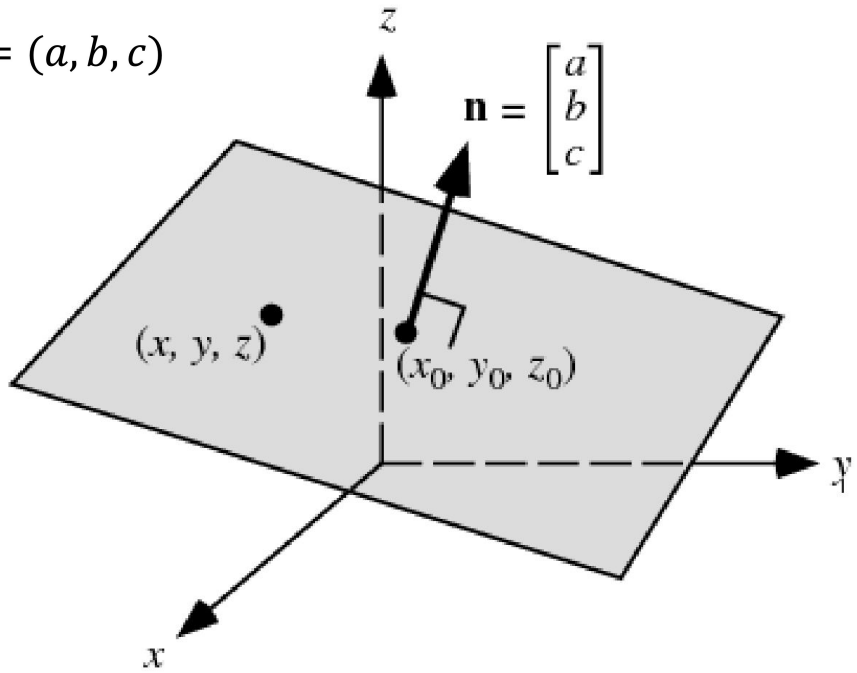
where $x = (x, y, z)$

Plugging this point into the general equation of a plane gives:

$$ax + by + cz + d = 0 \quad (2)$$

where $\mathrm{d} = -ax_0 - by_0 - c\,z_0$

To check if a vertex is above or below the plane, we use equation (2) and if the result is greater than zero, the point is above the plane, else it is below.



Task 3: Assign different colors to the model vertices based on their position with respect to the plane (below or above)

3.1 Calculate the coefficients (a, b, c, d) of the plane

3.2 Change the color of the fragments according to the position of the vertex with respect to the plane.

3.2.a VS : propagate vertex coordinates (world space) to fragment shader

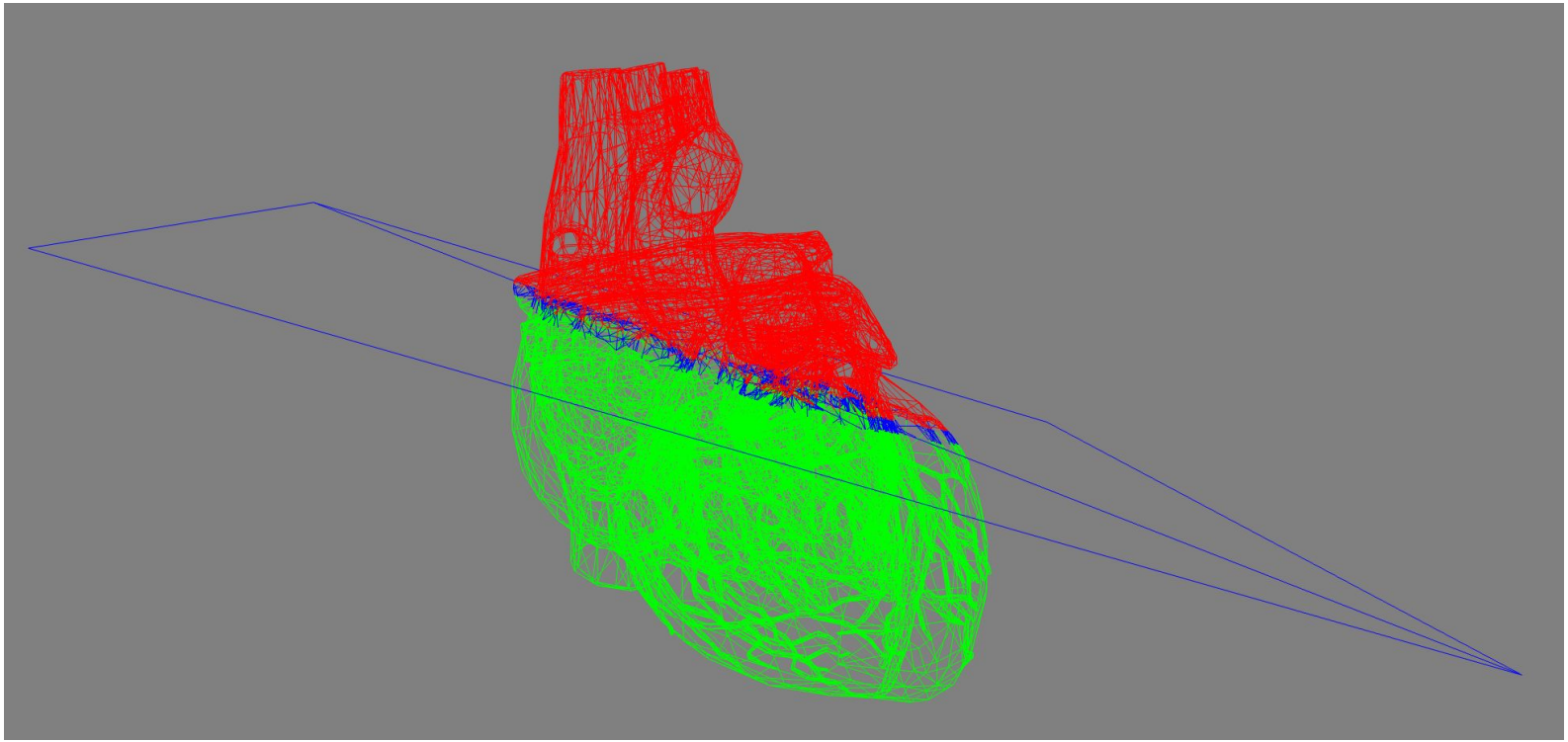
3.2.b VS: calculate vertex position in world space

3.2.c FS: get vertex position from fragment shader

3.2.d FS: get the coefficients of the plane from main program (uniform vec4)

3.2.e FS: find on which side of the plane is the vertex and apply different colors (red >0.02, green <-0.02, else blue)

Task 3: Assign different colors to the model vertices based on their position with respect to the plane (below or above)



Task 3: Assign different colors to the model vertices based on their position with respect to the plane (below or above)

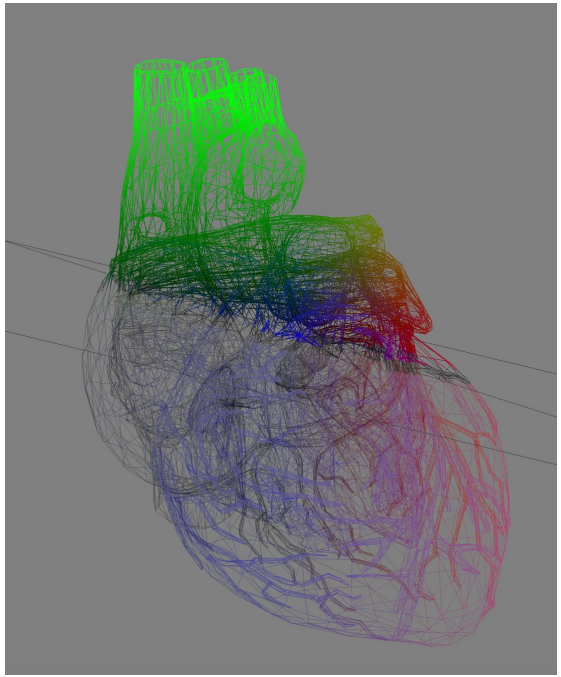
3.3 FS: color the model based on the vertex position and make the lower part vanish party (use alpha).

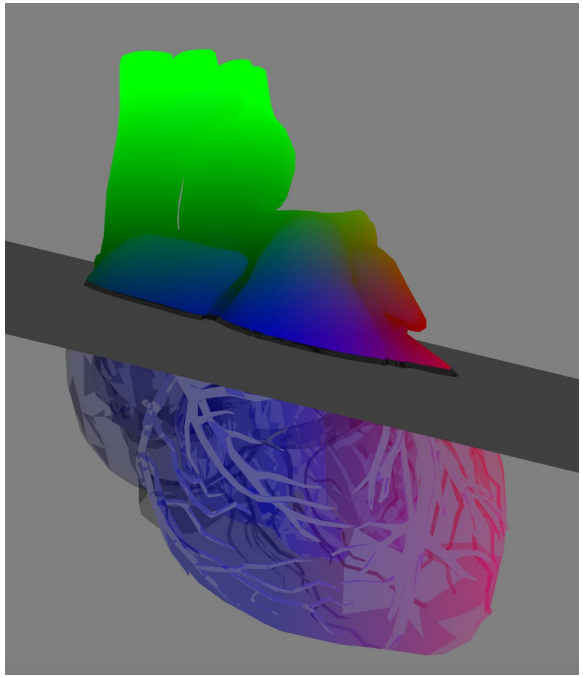
```
// Task 3.3: blend must be enabled
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

** Do not forget to comment out Task 3.2.e FS

Task 3: Assign different colors to the model vertices based on their position with respect to the plane (below or above)

3.4 Add a wireframe toggle action that will check the current polygon mode and change it to either FILL or LINE.





Task 4: Description

- Separate the object into two parts using the position of the plane and adjust the displacement of the two halves using the keyboard.
- The detachment offset will be adjusted from the keyboard.
- The vertex coordinates must be transformed so that the coordinates are moved away from the plane.
- This transformation will cause undesired stretching of the intermediate edges. Solve this problem by discarding the fragments that are between the two halves.

Task 4: Subtasks

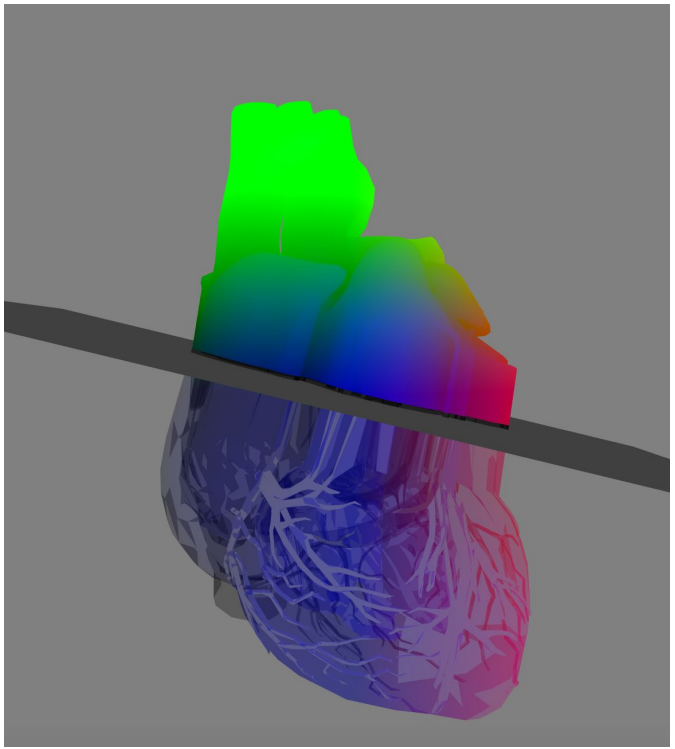
4.1.a Change the detachment coefficient using U, O keys. The detachment coefficient must not have a negative value.

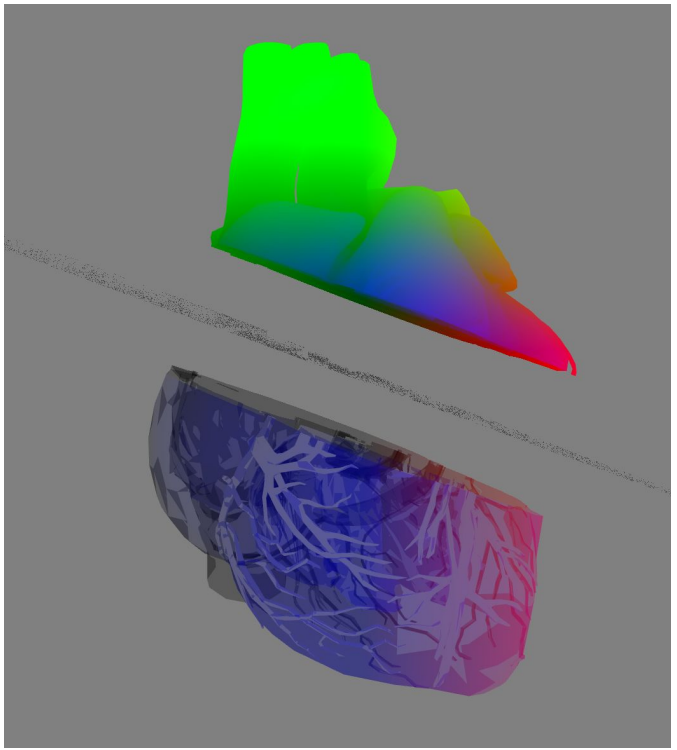
4.1.b Calculate and transmit the detachment offset to the GPU

4.1.c Displace the coordinates above the plane by the detachmentDisplacement and the coordinated below by -detachmentDisplacement

4.1.d Discard the fragments that are between the two halves

Task 4.1.c and 4.1.d





Task 5: Use a new shader program to render the plane

5.1 Add shader files to the CMakeLists.txt and rebuild the project. The shader files should be visible in the SolutionExplorer

5.2 Load the new shader program and allocate memory for the View-Projection

5.3 Use the new shader program to draw the plane. Use red color to see make sure that you are using the correct shader program. (Alter the code from Task 2)