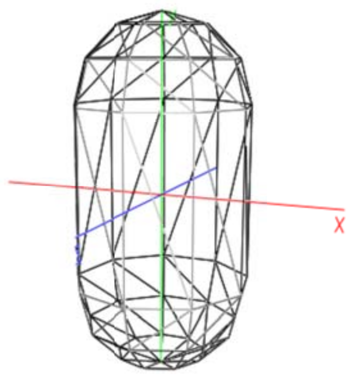
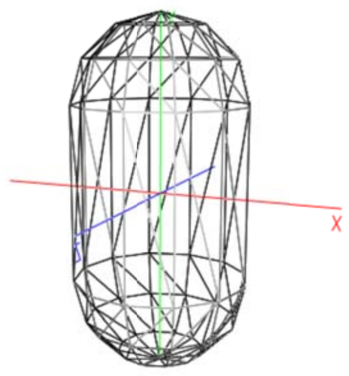
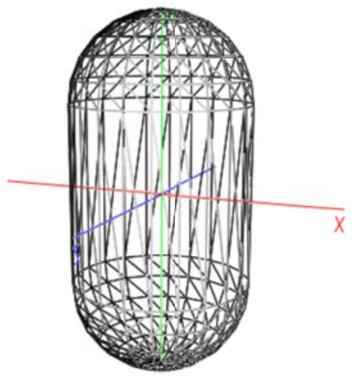
Programing Assignment #2

In this assignment you will practice 3D vector algebra and you will draw a parameterized 3D capsule using line primitives. Below is a picture of how your capsule should look like at different resolutions:







Note: Some versions of OpenGL support quadrilaterals (with GL_QUADS) to form primitives, however, GL_QUADS are not supposed to be available in the core profile, so stick with triangles or lines. See for ex.: https://www.opengl.org/sdk/docs/man/html/glDrawArrays.xhtml

For this assignment you will use the support code **`glutapp3d.7z`**. You will see that glutapp3d uses a class `SoAxis` that encapsulates an axis object that can be rotated with the arrow keys. You will create a similar class but called `SoCapsule` for drawing your capsule object. Implement your class in new so_capsule.h/.cpp files. Ask for help if you need assistance with Visual Studio.

Requirement 1 (45%): Implement class SoCapsule to draw your wire capsule

Your class has to be implemented in files `so_capsule.h` and `so_capsule.cpp`. It also has to have methods `init()`, `build()`, and `draw()`, following the same structure of the example class `SoAxis`. Method `build()` will make a wireframe capsule centered at the origin and with the main axis along the Y axis. The parameter list should take the following inputs:

```
void SoCapsule::build ( float len, float rt, float rb, int nfaces );
```

- **`len`** : the length of the "tube section" of the capsule along the (vertical) Y axis,
- **`rt`** : the radius of the top semi-sphere,
- **`rb`** : the radius of the bottom semi-sphere,
- **`nfaces`** : the number of faces approximating the curved faces.

Note that the capsule is divided in three parts: the tube section, the top semi-sphere, and the bottom semi-sphere. Parameter `len` tells the length of the tube section.

Parameter `nfaces` tells how many faces to create around the tube. Parameter `nfaces` will also be used to specify how many segments should be used to approximate the arcs of the semi-spheres. In this way parameter `nfaces` will control the overall resolution of the capsule.

Parameters `rt` and `rb` will allow you to make capsules with different appearances, for example if `rt` is (almost) zero and `rb` is greater than zero, the obtained shape will look more like a drop of water than a capsule.

To draw the capsule you will have to use some trigonometry and determine mathematically where the vertices of the capsule will be situated in the 3D environment given the described parameterization. The vertices will then have to be put in arrays in the correct order, so that all the needed lines that will represent the capsule can be properly defined.

Make sure you use a color different than pure red, blue or green (the colors used by `SoAxis`). Your capsule object will be reused in other assignments, so do your best to do a good implementation. Note that only 3D lines are required for drawing the capsule in this assignment.

When you draw your capsule make sure that you get the capsule to move together with the included 3D axes. In that way you can inspect the capsule from all points of view.

Requirement 2 (45%): Correct parameterization of the capsule.

Each parameter sent to your build() method has to be correctly taken into account when you draw the shape of the capsule. In order to test your implementation you will use the following keys:

- '**q**' : increment the number of faces (by one)
- '**a**' : decrement the number of faces (by one)
- '**w**' : increment the top face radius (by a small value)
- '**s**' : decrement the top face radius (by a small value)
- '**e**' : increment the bottom face radius (by a small value)
- '**d**' : decrement the bottom face radius (by a small value)

Requirement 3 (10%): Overall quality.

To receive full credits you have to present a good implementation of your project and submit a well-organized and commented source code. Extensions beyond the minimum are encouraged and will be taken into account.

Grading:

1) (5%) the SoCapsule class is correctly designed
2) (20%) your implementation draws a capsule on the screen
3) (5%) the capsule moves together with the axis and can be seen from any point of view
4) (15%) the lines approximating the top and bottom semi-spheres are correct
5) (45%) the parameters correctly change the capsule and the key controls are correct
6) (10%) overall quality

Submission:
Please present and submit your PA as usual.

Note: To better understand what the projection matrix is doing in glutapp3d.7z, check also the projection.exe application in “gltutors-projection.7z”, and be sure to read the readme file inside it!