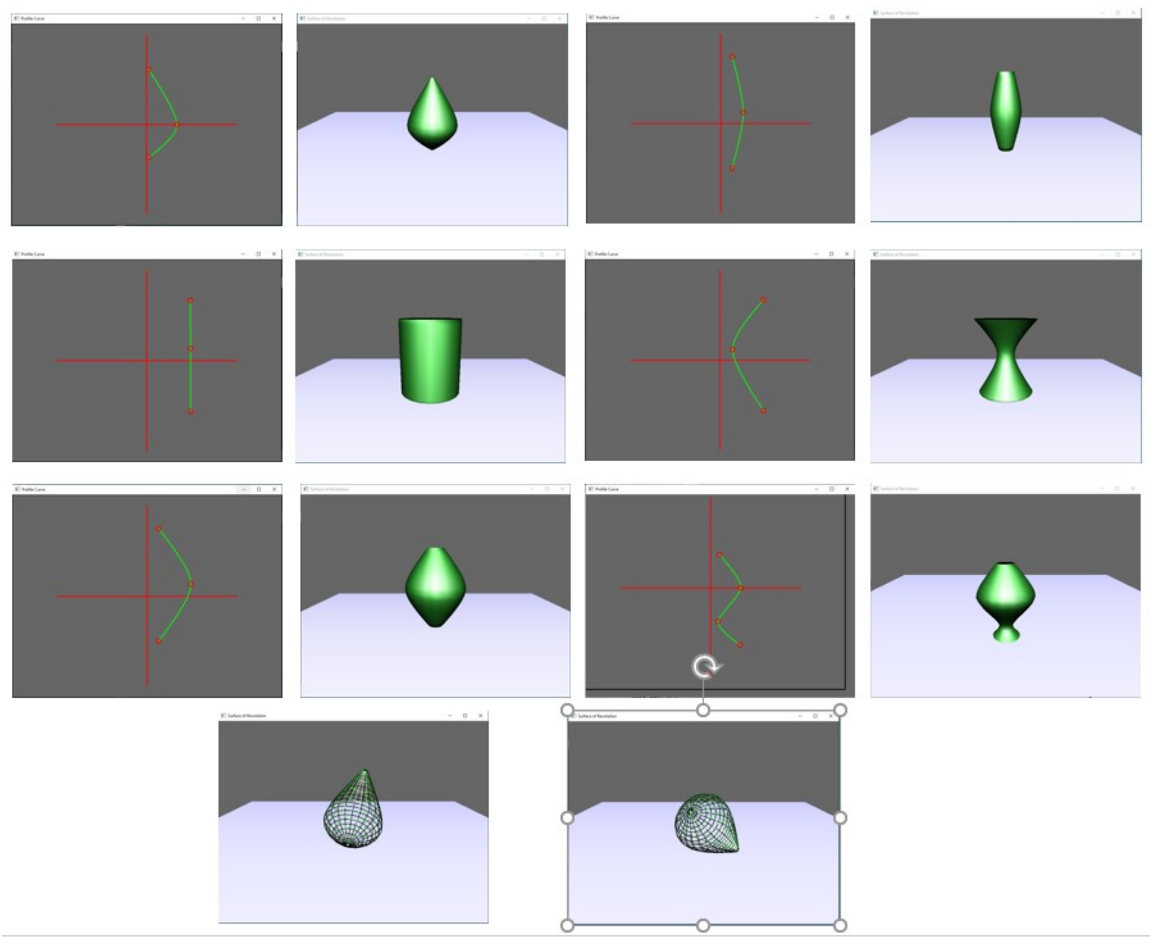
CPS511 Assignment 2:

Interactive Shape Modelling

Due Date: Nov. 9 11:59pm

You will implement an interactive shape modeling program using surfaces of revolution. This programming assignment will increase your knowledge of shape modeling, lighting, and shading. **You must do this assignment alone – no groups. Do not attempt to find source code on the web for this assignment. It will not help you and you risk extremely serious consequences.** Begin designing and programming early! This project is worth 12 percent of your mark. **If there is some part of the assignment you do not understand, please see me (or email me) as soon as possible and I will clarify the issue.**



Program Functionality Requirements:

<u>You must use the provided subdivision curve code as the profile curve.</u>

Your program should allow the user to design and build a shape using **surfaces of revolution**. Surfaces of revolution are generated by taking a smooth **profile curve**, defined using a spline or in this case a subdivision curve, copying its points and rotating the points around the y-axis by theta degrees (example: theta = 5 degrees). This copying and rotating process is repeated by incrementing theta until theta reaches 360 degrees. For example, from 0 degrees to 5, 10, 15, 20 , … 355. In my example, the increment is 5 degrees but it can be as small or as big as you like, creating a greater number or lesser number of curves, respectively. The curves are represented as an **array of points.** Quad polygons are then constructed by joining 2 successive points on the previous curve with its two corresponding points on the *current* curve. See the figures above. The bottom two figures show the surface or revolution drawn as lines (*wireframe*) and the surface has also been rotated.

Program Requirements

1. Use the provided skeleton code, which includes subdivision curve code to use as your profile curve. The skeleton code creates two windows (one for the profile curve and one for the surface of revolution). Interactively position (with the mouse) several control points to create a smooth profile curve. The provided code allows you to select one of 3 control points and move it around with the mouse. You can also delete control points and add new control points. See the mouse handling code for the 2D window. Set a maximum number of control points.

   Once the user has finished shaping the profile curve, then automatically your program then makes a copy of this curve and rotates it around the y-axis by a small amount and joins the corresponding points of the two curves to form quad polygons. Your program should repeat this copying and rotating process, forming quads with the current curve points and the previous until you have rotated a copied curve all the way around the y-axis and joined the last copied curve with the first curve to complete the surface of revolution. In effect, you are creating a mesh of quads. However, if the first control point of your profile curve is exactly on the y-axis (see cone example above) then you would need to form triangles between the current and previous curve points at the top. **Note: you may skip this "control point exactly on the y-axis" functionality and always ensure the control point is a small (positive) distance away from the y-axis. This keeps all polygons as quads and the shape will still looked "closed" at the top if the first and last control points are close enough to the y-axis. This also simplifies texture mapping in assignment 3.**

2. The finished 3D surface of revolution should be displayed in a separate window. This code is already provided, and a teapot is drawn initially. You are to fill in the empty functions/methods in the provided code (or write your own functions) and create variables and data structures needed. Then delete the teapot and draw your surface of revolution.

3. You must be able to use the mouse to interactively rotate your surface of revolution around the y-axis in the 3D window and around the x-axis (and keep the camera fixed). This will allow you to view your shape from different angles. You can also achieve this, if you prefer, by provide camera control such that the camera is on an imaginary hemisphere that covers your whole 3D world. You then provide the ability to move the camera around this hemisphere (i.e. control elevation and azimuth of the camera) using the mouse. Note: you might want to use a key to reset the orientation of your surface (or reset the initial camera view). I suggest the ‘r’ key.

4. You must be able to use the mouse (right mouse button and scroll wheel) and zoom in/out in the 3D window.

5. You must use lighting and shading to draw your surface of revolution. This means you need to calculate normal vectors for each vertex correctly. I have provided some hints in the skeleton code on how to do this. Basically, you should write a method that generates a quad normal vector for each quad. See my comments in *computeQuadNormals()* and in *computeVertexNormals()*. Then for each vertex you will need to average the normal of the 4 quads that share the vertex and normalize the resulting normal vector.

6. You must provide a key that changes the draw mode to “*wireframe*”, where you are drawing the edges of the quads. See the bottom two figures above. Use another key to flip back to solid shading.

7. Provide a ‘help’ key that when pressed tells the user how use your user interface (keys, mouse etc). You can simply write out the help instructions on a console window or on one of the two render windows.

Optional Bonus (1 mark)

1) Use VAOs/VBOs and a vertex+fragment shader to draw your surface and create more realistic lighting/shading. That is, do your lighting calculations in the fragment shader. (0.5 marks).
2) Allow the user more control over the shape of the surface of revolution. For example, provide the ability to create small bumps or patterns of indented regions etc. (0.5 to 1 mark)
3) Create a 3rd rendering style for your surface that draws the surface in wireframe mode but does not draw any backward facing quad edges. This will make your wireframe rendering look “cleaner”. Hint: the vector from the reference point to the camera position has a negative dot product with any back facing quad/vertex normal (0.5 marks)

The TA will judge the difficulty of your added functionality when deciding on bonus marks. **The maximum bonus for the assignment is 1 mark.**

NOTE

1. Once you have built and displayed a 3D shape, you may want to write the data structures to a file. In assignment 3, you will be replacing some of your simple bot parts with the more elaborate shapes you have created here. You will need to recreate your data structures for these parts in your assignment 3 code. You could write code that reads your files and recreates the data structures. Alternatively, you could just save the 3 or 4 profile curve control points for each saved shape and incorporate your assignment 2 code into your assignment 3 code (the part that generates the surface of revolution from the profile curve).

Grading (Out of 12 marks)

Final surface of revolution is correctly constructed and rendered. This implies normal vectors are constructed correctly.	8 marks
Render surface in wireframe mode.	2 marks
Interactive rotation of surface around x,y axes or via camera control	1 mark
Interactive zoom in/out via mouse wheel or when holding right mouse button down	1 mark
Bonus	1 marks
Total	13 marks

Program Submission

Use D2L to submit your assignment. Submit all your source files. You may use C, C++, or java for your program. Zip everything up into one file. **Do not include executable files.** If your program runs under Windows, include a README file describing how to compile your program. If you want to inform the TA about your program (special features, bonus work etc) include this information in program comments and the README file. Include all makefiles (for Linux) or solution+project files (for example, if you used Visual Studio). **It is your responsibility to ensure the TA has enough information so that he can, with little effort, compile and run your program**. I am being flexible in terms of your programming language and operating systems choice so you must try to meet me halfway. If the TA has trouble compiling your program, he will have the discretion to deduct marks and/or he will ask you to compile and run your program in his presence.