Assignment #3

Point Value: 110 points

Due Date: April 18th, 11:59pm

For this assignment you will write a program that will convert input from a Bézier surface representation to a polygonal representation and then display it with OpenGL.

1. **This assignment is due as indicted above. Projects turned in late will lose points as described in the class policies handout. This assignment should be done alone or in pairs. You may share ideas with other groups, but you may not share code.**

2. **You may develop on Unix, OS X, or Windows. The platform you use will be the one used to grade assignments. Keep in mind that there are slight variations due to OS versions, different libraries, and other factors, so you should verify that your code runs on the instructional machines appropriate for you platform choice.**

3. **We will be using the submit software for submission of this assignment. Instructions for using the submission software are here. You should include a README file that at the minimum contains the following information:**

   - Your (and your partner's) name
   - The platform your code runs on
   - The location of your source code (i.e. indicate who in your group has done the submission, and on what platform). Only one of the people in your group should submit the actual code. The other people should only submit the README file.

   All files needed to compile your code should appear in the submitted directory. It is your responsibility to make sure that they will compile and run properly.

   - Windows: The grader should be able to recompile your program by simply opening the project and rebuilding it from scratch.
   - Unix and OS X: The grader should be able to recompile your program simply by typing "make".

   You will also turn in some images. These should be named "image-nn.xxx" where nn is a number (e.g. 01, 02, 03...) and xxx is an appropriate extension (e.g. tif, jpg, ppm, etc.)

4. **Once you have your assignment working, you should also update your class web page to include an "Assignment 03" link to a page with some screenshots taken of your program running. Make sure that your images demonstrate all the features that you have implemented in your code. The images on your web page are the same ones that should be submitted with your code.**

5. **Optionally, you may earn some extra credit by also posting an animation captured showing your program in use. While you are responsible for figuring out the most appropriate method for doing this, the TAs and Professor might have some helpful suggestions. Poor quality video generated using a video camera pointed at your screen is not adequate. Also, files that are unreasonably large (*i.e.* poor choice of compression codec), unplayable on the grader's computer (*i.e.* a non-standard codec), or of very poor quality (*i.e.* bad selection of compression parameters) may receive little or no bonus credit.**

6. **This assignment will be graded in part by looking at your web page to see that you have posted examples demonstrating all required features. The images on your website should correspond to the ones you submitted. You may add extra images to your website after the deadline, but these images should clearly be marked as extra images that were not included in the submission.**

   If you work in a group, you should all link to the same web page and the web page should list your group members.

   Grading will include a few points for aesthetics and creativity as demonstrated on your web page.

7. ***Do not wait until the last minute to start this assignment.*** **As you may have learned with your raytracer, this assignment can go very smoothly if you start early and plan ahead. If you wait until the last minute to start, then you will be a sad, sad puppy (or kitten at your option).**

   Check the news group regularly for updates on the assignment or other clarification. We will assume that anything posted there is henceforth known to all.

8. **Submitting an image that was not generated by your code is considered cheating. Adding images to your website after the deadline without clearly marking them as added later is also cheating.**

9. **For this assignment, you will write a program that can convert Bézier patches to polygons and then display the result. It should:**

   - Read in a list of patch data from a file.
   - Tessellate/triangulate the patch using either a uniform or adaptive technique. (You have to implement both, your program will have a command-line flag to determine which is used.)
   - Open a window and use OpenGL to Render the object.
   - When "s" is pressed the program will toggle between flat and smooth shading.
   - When "w" is pressed the program will toggle between filled and wireframe mode.
   - *Optional: When "h" is pressed the program will toggle between filled and hidden-line mode.*
   - When the arrow keys are pressed the object will be rotated.

- When the shift+arrow keys are pressed the object will be translated.
- When launched the initial zoom will show the entire object. Pressing the +/- keys will zoom in/out.

10. **Your program will take two command line arguments with a third option parameter. These are: the input file name, the subdivision parameter, and a flag which determines if subdivision should be adaptive or uniform. An example command would look like:**

    % myprogram3 inputfile.bez 0.1 -a

11. **The input file contains a list of patches. The subdivision *parameter* should be interpreted as the step size in U and V for uniform subdivision, or as an error measure for adaptive subdivision. If the -a is present, then adaptive subdivision should be used, otherwise uniform.**

    The input file contains the control points for the patches and an output file name. Keep in mind that the control points are vectors, not scalars! Example input files will be posted on the class website.

12. **For uniform tessellation, quadrilateral polygons should be formed by taking *parameter*-sized steps in the U and V direction.**

13. **For adaptive triangulation, the error between the actual surface and the quadrilateral polygon should be less than *parameter*. The error should be distance evaluated at the midpoints of the polygon edges and the center of each quadrilateral. Techniques to avoid cracking should be used in the adaptive subdivision code. You should use a subdivision technique that yields triangles as discussed in class.**

14. **You must write the code to tesselate/triangulate the surfaces yourself. Do not use a utility library to do this for you!**

15. **The images you submit should demonstrate your features on the "teapot" and "arch" inputs. You should also use Maya (or some other program) to create at least one additional input file.**

    Maya can be used to create B-Spline surfaces that can be exported to files. Figure out how to get data out of Maya and converted to the format expected by your program. Or you can use a text editor to hand code a file.

16. **<u>Optional</u>: The use of OBJ input/output for this assignment is optional and will be worth some small amount of bonus points. OBJ is a simple 3D surface file format, described at http://www.royriggs.com/obj.html and elsewhere on the web. In it's simplest form, it is just a list of vertices followed by a list of faces (potentially just triangles).**

    If you implement OBJ *output* then your program should accept an optional command line argument that comes after the required ones. This is -o and a file name. If the -o option is given then no OpenGL window should be opened and instead the model should be written to the specified output file name.

If you implement OBJ *input* then your code should recognize that the input file has a .obj extension instead of .bez and directly display the polygons in the OBJ file.

17. **Optional: Add the ability to load multiple objects (for example a mix of .bez and .obj files) and display them each with different transformations. Implement a method for selecting one of the objects (for example, using a key to cycle through objects) and highlight the selected object. The arrow keys will now move the selected object. If you implement this option then you'll likely want to define some sort of ".scene" file that contains a list objects to load and transformations for each object. It you really want to have fun, add the ability to write out these scene files and pass them to your ray-tracer.**

18. **Optional: When "c" is pressed the program will do vertex color shading based on the Gaussian Curvature of the surface. If the surface is a Bézier surface then the curvature should be computed from the cubic surface's derivatives. If it is a polygon (.obj) surface then come up with some method for estimating the curvature from the polygons.**

19. **Optional: Refer to the example GLSL code posted on the website and implement a custom shader for your object display.**

20. **Questions should be posted to the news group or emailed to cs184.**

21. **For CS284a, the following features are not optional, but required:**

    - Load and display OBJ files
    - On press of "c" key display Gaussian Curvature of the surface