COMP 557 - Fall 2020 - Assignment 3
Bezier Surfaces

Available 23:30 pm Tuesday 27 October

Due 23:30 pm Tuesday 10 November

Getting Started

The purpose of this assignment is to evaluate Bezier surfaces and to draw smooth objects using OpenGL.

The Bezier surfaces in this assignment are bicubic polynomials of order 3 in each parameter, thus each patch is defined by a 4 by 4 grid of control points. Unlike a triangle mesh, a Bezier surface patch is smooth, differentiable, and its shape is entirely specified by its 16 control points. However, to draw such a smooth surface in OpenGL we must approximate the smooth surface with small primitives (e.g., triangles). You will do this by evaluating the surface at regular intervals along the s and t directions, and drawing triangle strips.

While it is possible to use geometry shaders and or tesselation shaders in OpenGL to evaluate and draw surfaces, in the assignment you will implement your own code to evaluate the surface and its derivatives given a set of control points.

Provided Code

The provided runs, but does nothing but draw a world axis and a ground plane, and calls a collection of methods in the BezierPatchWork class that you need to complete. The sample code zip file has the following contents.

- A3App and BezierPatchwork java files, which live in the comp557.a3 package. You will need the jogl fat jar and vecmath jars as per the previous assignment.
- mintools.jar, which is updated with the arcball and shadow map solution from the previous assignment.
- Several Data files are included. The file testPatch.txt contains one single flat Bezier patch. The file testPatches.txt contains two Bezier patches. The third contains Bezier patches defining the Utah teapot. Change the constructor of A3App to run your application with different data files.

The code (BezierPatchWork.java) contains TODO comments in the places where you will need to add code to complete the assignment objectives specified below. The A3App code sets up the interface and adds a number of swing controls to a control window. You will use these controls to turn on or off or adjust the display of different things as specified in the objectives.

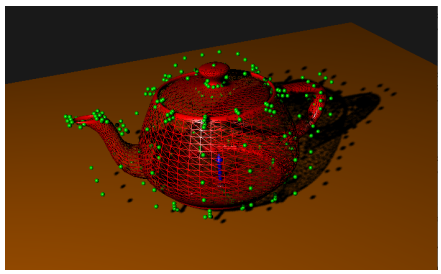
Bezier Patch Data

The teapot is composed of 32 Bezier patches, each with a 4 by 4 grid of control points for a total of 16 control points each. You will notice that the BezierPatchWork class already contains code for loading all this data into an array of Matrix4d objects. Each matrix contains the 16 control points of a given patch for a given axis (x, y, or z). For instance, Gx, the 4 by 4 matrix of x coordinates of control points of the first patch will be found in coordinatePatch[0][0], while the y coordinates are in coordinatePatch[0][1], and z in coordinatePatch[0][2]. This code is provided as a convenience to help you get started faster. But if you feel strongly about storing your control points in some other manner then feel free to change it!

Steps and Objectives

1. ***Control Points***

   The *drawControlPoints* method of BezierPatchWork is currently unfinished. Add code to draw the control points by drawing small green spheres of radius 0.05. Use the setkd method of the pipeline to set the colour. The "Display Control Points" checkbox in the controls window will let you enable drawing. Make sure it is checked when you are testing your code (it is on by default).



2. ***Surface***

The *draw* method of BezierPatchWork uses and vertex buffers that are set up in the init. It will evaluate the function with a regular spacing set by *subivisions* to set the positions of a set of triangle strips, which are then drawn. The buffers are reused to evaluate all the different patches on every single draw call (not the most computationally efficient choice, but a simple set up for learning).

Finish implementing the evaluate coordinate call to set the vertex positions to see the shape of the patch.

Note that the "Display Bezier Mesh" checkbox must be checked when you are testing your code (it is checked by default).

3. ***Tangents***

   Write code to compute the surface tangents (i.e., the derivatives in the s and t directions). Write code in the *drawSurfaceTangents* method to draw the s and t derivatives as red and green lines at the surface point (s,t) provided in the parameters.

   Note that the default state of the controls is not to call this method. Use the s and t sliders in the controls window to check that your tangent vectors make sense.

4. ***Normals***

   Write code to compute surface normals using your code that computes tangent vectors. Compute a normal for each surface point you evaluate in objective 2, and modify the patch drawing code so that the normal is set properly in the buffer.

5. ***Repair Bad Normals***

   Note that some surface points have bad normals, notably the top and bottom of the teapot. Why is this happening? Write some code to check for when this is happening, and use a nearby point of the surface to approximate the desired value of the bad normal. Note that your "fix" need not work in all cases, but should work in the case of the teapot data.

Written Questions

Optional

Up to two bonus marks will be given if you add extra polish to your assignment:

1. Create an interesting shape with Bezier patches in the format that can be loaded by this assignment. For one example, a Trefoil know using an approximately circular collection of 4 Bezier curves (like the teapot) swept along a Frenet frames or Bishop frame of some appropriate path function. For a second example, a vase that also resemble human faces, by creating a surface of revolution. Note that the sweep of your surface of revolution can again be 4 Bezier curves much like in the teapot.
2. Implement rational surfaces by using homogeneous coordinates for your points. Briefly describe your implementation in your readme and provide a data file for a sphere made of 8 cubic tensor product patches. Provide other examples should you be so inspired!

Finished?

Great! Be sure your name and student number appears is in the window title, in your readme, and in the top comments section of each of your source files.

Prepare a zip file with your source code folder and readme file if you made one. Only use a *zip* archive! Submit it via MyCourses. **DOUBLE CHECK** your submitted files by downloading them. You cannot receive any marks for assignments with missing or corrupt files!! **Treat deadlines as HARD**. If you submit multiple times, only your last submission will be graded

Note that you are encouraged to discuss assignments with your classmates, but not to the point of sharing code or answers. All work must be your own. Please talk to the TAs or the prof if the academic integrity policies are not clear.