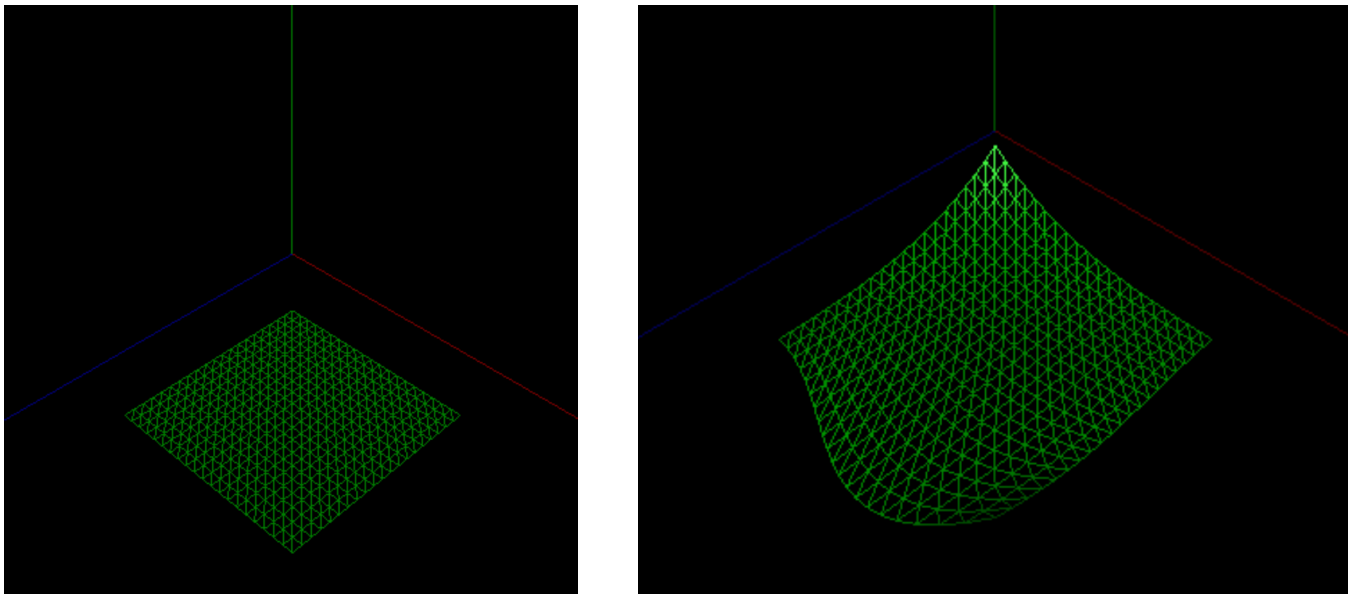
Assignment 5 Writeup Ryan Shendler

My program has two main functions. The first is to render a wireframe Bezier surface patch based on the position of 16 control points. Four of these control points can be moved, which changes the shape of the Bezier surface. The second function is to implement a lighting model which can render the Bezier surface using either flat or smooth shading.

For the first part of my program, I used the global array "points" to store the current locations of the 16 control points. I labeled these control points from 0 to 15 and included menu options that allow the user to change the X, Y, and Z coordinates of control points 0, 5, 11, and 14. To do this, a user needs to right-click to open the menu, hover over "Control Point 0," "Control Point 5," "Control Point 11," and "Control Point 14" to open the corresponding sub-menu and then select the option to either increase or decrease the X, Y, or Z coordinate of that control point. Each time a control point's location is changed, my program automatically redraws the Bezier surface with its new shape.

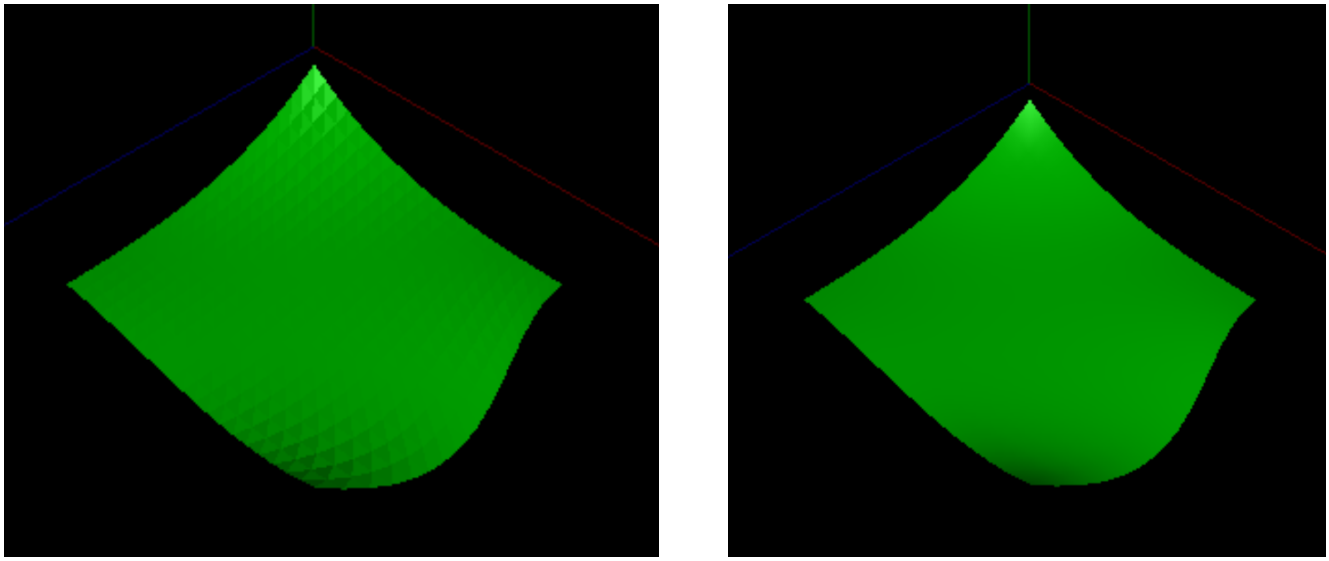


Bezier curve before and after several control points are moved

The Bezier curve is drawn by the function drawBezier(), which iterates through u and v from 0 to 1 in steps of 0.05. For each combination of u and v, my program calculates the corresponding surface point using the function calcPoint(). Once all of the surface points have been calculated, my program constructs a mesh of triangles, with each of the surface points being used as a vertex for one or more triangles. The vector "indices", which is found in drawBezier(), stores the indices for the vertices of each triangle, as well as the surface normal for that triangle. The surface normal for each triangle is calculated in the function calcNormal(), which calculates the normalized surface normal for a triangle, given the coordinates of its vertices. In order to actually draw the Bezier patch, my program then renders each of these triangles that are visible. In order to determine if a triangular face is visible, I defined the function cull(), which takes the dot product of the viewing vector and the surface normal of that triangle. If the dot product is greater than 0, then the triangle is considered a back face and is therefore not rendered. During the process of rendering the triangular mesh, my program uses the function vertNormal() to calculate the vertex normal for each vertex in the mesh. These normals are not normalized, but I used glEnable(GL_NORMALIZE) to automatically normalize all vertex normals before lighting calculations are performed.

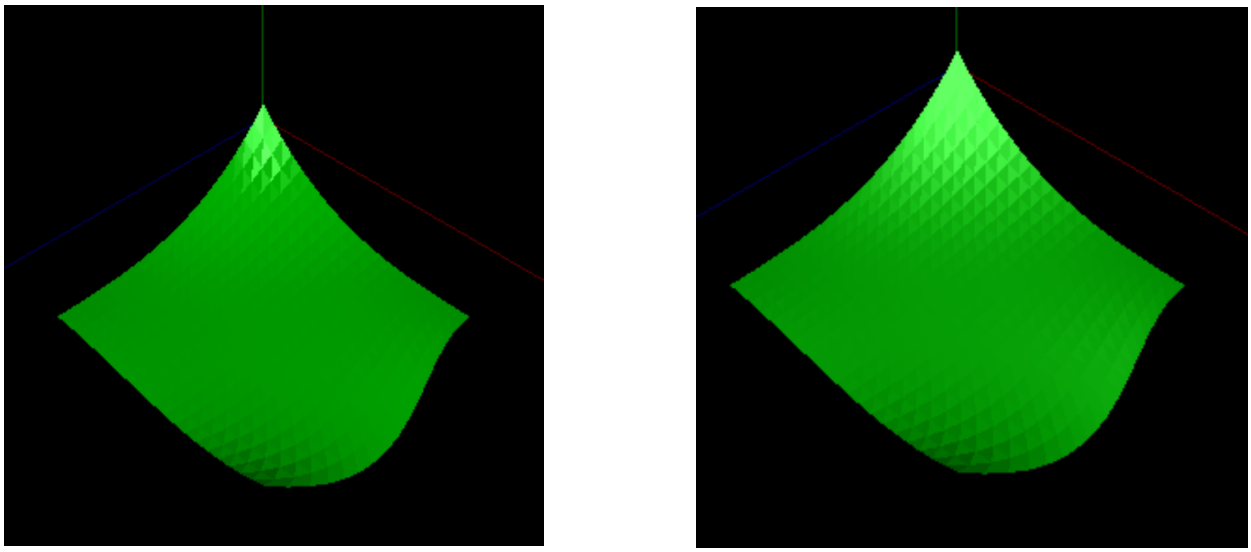
For the lighting portion of my program, I used a single light source, GL_LIGHT0. This light source was initially located at the point (50, 50, 50) and shone at the origin, but users have the option of moving the light source along the Y-direction by using the menu options "Light+" and "Light-". In a similar fashion, users can also use the menu options "Camera+" and "Camera-" to move the camera in the Z-direction. Users have three different options for how the Bezier surface is displayed: wireframe, flat shading, or smooth shading. The user can change the current display mode by using the menu option "Display Mode." Internally, the program uses the

global integer “mode,” to keep track of the current display mode. A value of 0 means wireframe, 1 means flat shading, and 2 means smooth shading.



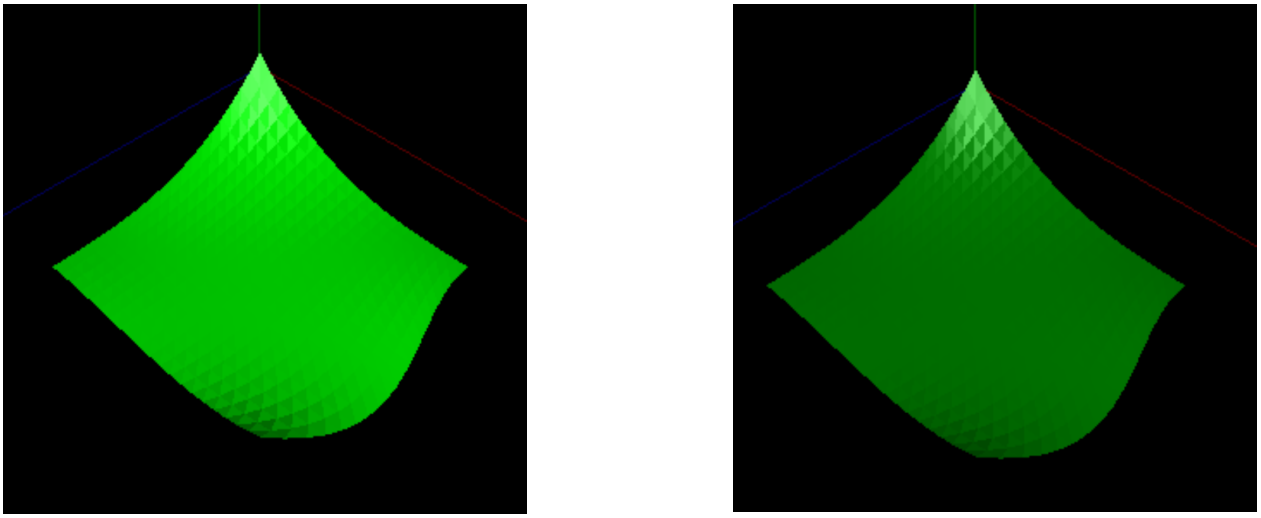
Bezier surface with flat shading (left) and smooth shading (right)

In addition to being able to change the display mode, users can also change the specular exponent and the diffuse value of the Bezier surface by using the menu option “Lighting.” Increasing the specular exponent will cause reflected white light to become more focused while decreasing it will have the opposite effect.



High specular exponent (left) and low specular exponent (right)

Increasing the diffuse value will make the surface have a more vibrant hue while decreasing it will make the surface appear duller.



High diffuse value (left) and low diffuse value (right)

In order to run my program, open the Assignment5.sln file in Visual Studio and build the project. All of the code is located in main.cpp. My answers to the theory questions are located in the file Assignment5Theory.pdf.