Project 3

Due: 8 May 2017

Edited as of 2:30pm on 2 May (see Option 1)
Edited as of 4:50pm on 7 May (see Option 1)

This is the assignment that will count for 20% of your "5th assessment" grade. The remainder will come from your score on the prior assessment that you choose to re-do. (The re-done assessment will *also* be averaged with the old score on that assessment, boosting the score in that assignment as well.)

For this implementation, you will choose one of the following four options.

Option 1: Mesh rendering

In an environment with either camera control (buttons, sliders, etc) or automatic movement that rotates the camera around the object or rotates the object (so as to view the object from all angles): display a polygon-rendered mesh.

Intro version: Render a simple mesh of three quadrilaterals in a short strip (use the coordinates posted online).

Full version: Render the Utah teapot (use the coordinates posted online). This will require managing a set of indices into a shared vertex array. Some of the points need to be reflected about various axes to generate the entire teapot. The teapot should be visible at a reasonable scale and (with controls or camera movement) from all/most angles.

Edited to add: The data for the teapot is grouped into "patches" of 16 points—to be clear, this represents a 4x4 grid of points (hence a 3x3 grid of quadrilaterals). The data for the intro strip is 8 points in a 4x2 grid* (so, three quads). I've edited the point and patch data to clarify this a little. You should get it displaying as solid triangles with lighting and different normals (although points and line strips could be useful for debugging!) but can get full credit even if the normals are not quite right, especially in the reflected parts.

***Edited again to add:** The 4x2 grid is row-major; the first four points are the first row and the last four points are the second row. It's a lot like

the first half of a patch (which was intentional). If you plot the points of the strip by hand you should see how they form a strip.

Option 2: Scene graph

Model and render a cluster of primitive objects in a scene, where subordinate objects are modeled by their transformation matrix *relative to* their "parent" object.

Intro version: Model and render a "house" where the roof is a cube rotated 45 degrees to make a peaked roof; the roof object should be subordinate to the base house object, so somewhere should be a single composed transformation matrix that, if changed, would move/rotate/scale the entire house as a unit. (This change does not need to be doable via HTML controls; it's ok if it requires editing the js source.)

Full version: The root scene will have a house (two parts), a tree (two parts), and a humanoid figure (very loosely construed; at least six parts, with the torso being the parent of the head and four limbs). All objects may be rendered using only cubes and cylinders, even if this is... somewhat stylised.

Option 3: Texture map

Use texture mapping to display an external 2D image in a 3D rendered model.

Intro version: Access a GIF or PNG file and map it onto a single (two-triangle) rectangle that is floating in the 3D space. The camera should be at an angle relative to the rectangle, and a perspective transform should be used, so we can see the mapped image distorted and 3D-looking.

Full version: The 2D image should be wrapped around a cylinder (think Coke can). Ok if ends of cylinder are solid grey or whatever. Camera should be controllable or should automatically move to display most/all angles of the object.

Option 4: Parametric curves

Use numeric solving on parametric functions to display a relatively smooth curve.

Intro version: Render a semicircle as a parametric curve where u ranges from 0 at one end to 1 at the other. Clicking a point in the canvas should move either the centre of the circle or one of the endpoints of the semicircle to that point. The centre and both endpoints of the semicircle should be drawn as visible, large points.

Full version: Render a Bézier curve with four control points. The control points should be drawn as visible, large points. There should be some way to select each of the four control points (HTML buttons are fine, doesn't need to be fancy) and once selected, clicking in the canvas should move the selected control point (and update the curve accordingly).

Rubric

The first ten points (of 20) are for doing the intro version and documenting it well enough that I can see what you've done.

The remaining ten points are assigned to upgrading that into the full version. Partial credit is available.

Handing it in

I do want you to hand in all the files for this, including a readme; but the readme should also include the URL that I can use to run/test your code. The handin command is

```
handin cmsc381 proj3 dirname/
```

The project is due **at 5:30pm** on the due date, i.e. the end of the exam period.