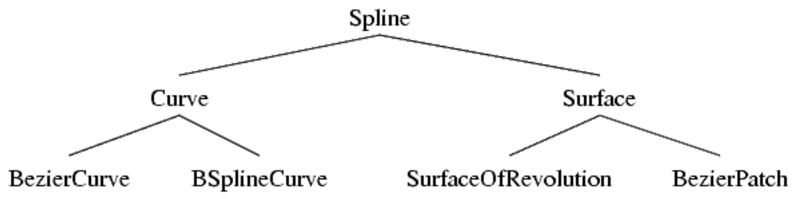
6.837 Intro to Computer Graphics
Assignment 8: Curves & Surfaces

In this assignment you will complete the implementation for a simple 2D spline editor. You will support both Bezier and BSplines, and convert between the two curves (for bicubic splines with exactly 4 control points). Once your curves are working you will move on to creating surfaces from these curves: surfaces of revolution and bicubic Bezier patches.

This assignment is not highly coupled with the previous assignments. We are giving you a significant new code base (the *curve_editor*). You'll just use your OpenGL previsualization (from assignment 3) to view the triangle meshes you create.

Tasks

- In this assignment you will implement a base class `Spline` from which you will derive `Curve`, `BezierCurve`, `BSplineCurve`, `Surface`, `SurfaceOfRevolution`, and `BezierPatch` as illustrated in the diagram below. Organize your code to allow reuse where possible and avoid copy-pasting similar code, as this makes it hard to debug later. See the spline_parser.C file for the expected constructors.



  In order to get your code to compile, you'll have to create some temporarily empty procedures in your Spline class. (Each is described below). Some of these may be pure virtual, depending on your implementation.

```
// FOR VISUALIZATION
virtual void Spline::Paint(ArgParser *args);

// FOR CONVERTING BETWEEN SPLINE TYPES
virtual void Spline::OutputBezier(FILE *file);
virtual void Spline::OutputBSpline(FILE *file);

// FOR CONTROL POINT PICKING
virtual int Spline::getNumVertices();
virtual Vec3f Spline::getVertex(int i);

// FOR EDITING OPERATIONS
virtual void Spline::moveControlPoint(int selectedPoint, float x, float y);
virtual void Spline::addControlPoint(int selectedPoint, float x, float y);
virtual void Spline::deleteControlPoint(int selectedPoint);

// FOR GENERATING TRIANGLES
virtual TriangleMesh* Spline::OutputTriangles(ArgParser *args);
```

- Implement the `Paint(ArgParser*)` function for the `Curve` classes. As in the example images below, you'll want to render the control points, the control polygon and the actual spline curve. The `-curve_tessellation <n>` command line argument will dictate the spacing of your samples --- where you evaluate Q(t) to approximate the spline. Useful OpenGL commands will be `glBegin(GL_POINTS)`, `glBegin(GL_LINES)`, `glPointSize(int)`, `glLineWidth(int)` and `glColor3f(float,float,float)`.

  For the first part of the assignment you may want to assume that your Bezier & BSpline curves have exactly 4 control points.

- Next, implement the conversion between Bezier and BSplines. That is, given a set of Bezier control points, find the (unique) set of BSpline control points that produce the same curve. As seen in lecture we can compute this by manipulating the matrix equation that describes the spline:

$$Q(t) = G_{bezier}B_{bezier}T = G_{bspline}B_{bspline}T$$

  Remember, Q is the curve, parameterized by t. T is the power basis, a vector of all the powers of t up to the degree of our curve (for this assignment $[t^3\ t^2\ t\ 1]^T$). B_{bezier} and $B_{bspline}$ are the spline basis matrix functions which are constant for each type of spline. And finally G_{bezier} and $G_{bspline}$ represent the control point geometry. Each control point is a column in the G matrix.

  The test cases below will help you verify that you've correctly solved the matrix equation to convert between these two types of splines. The command line option `-output_bezier <filename>` or `-output_bspline <filename>` will indicate which representation you should use when you save the splines. Implement the `OutputBezier(FILE *file)` and `OutputBSpline(FILE *file)` functions.

- Next, generalize your implementation of Bezier & BSpline curves to handle input with more than 4 control points. By considering only a window of 4 control points at one time, the spline will be locally cubic. For BSplines this generalization is very elegant: simply slide a window over one control point at a time. However, for Bezier curves the solution is a little less elegant. Two neighboring curves will only share one control point and thus the G^1 or C^1 continuity at the joint depends on the neighboring control points. A BSpline with n+3 control points is composed of n 4-point BSplines. A Bezier curve with 3*n+1 control points is composed of n 4-point Bezier curves.

  You're only required to implement the Bezier to BSpline and BSpline to Bezier conversion for curves with exactly 4 control points. Converting curves with > 4 control points is worth extra credit. (There are different ways to do it.)

- To hook up the control point editing we've provided, you'll need to give the `SplineParser::Pick(...)` and `SplineParser::PickEdge(...)` functions access to your Curve data structures. You'll do this by providing the `getNumVertices()` and `getVertex(int i)` functions.

  Implement the `moveControlPoint(...)`, `addControlPoint(...)` and `deleteControlPoint(...)` functions to complete the curve editing implementation. Notice that by adding or deleting a control point, you will not have a round number of Bezier control point windows. You may choose to disallow adding or deleting control points on Bezier curves, or implement a control point duplication strategy to compensate. Also you may wish to disallow deletions if there are exactly 4 control points.

  Once completed, the left mouse button is used to move a control point (the mouse down click must be within 10 pixels of a control point), the middle mouse button adds a control point on an edge (if the mouse down click is within 10 pixels of an edge) and the right mouse button deletes a control point (again if there is one within 10 pixels of the down click). Hitting the 's' key will save the geometry both in the spline format and the triangle mesh format (as specified on the command line).

- Next implement the SurfaceOfRevolution class. You may assume the curve will always spin around the y axis (x = 0, z = 0). We've provided the `TriangleMesh` and `TriangleNet` classes to help you generate regularly tessellated quadrilateral patches of triangles (when `OutputTriangles(...)` is called). To use the `TriangleNet` class, simply specify the number of faces you'd like along each dimension. Then you simply specify the vertex positions for the grid corners (note this is one bigger in each dimension.) Two command line arguments control the tessellation of a Surface of Revolution: `-curve_tessellation <n>` and `-revolution_tessellation <n>`.

  You'll probably find the `TriangleMesh::Merge(const TriangleMesh &m)` function useful for combining multiple meshes into a single file.

- Finally, implement a 4x4 Bezier Patch with 16 contol points. Instead of having just a single parameter *t* (as with the curves), you'll be given two parameters, *t* and *s*. First use *t* to interpolate along the 4 parallel 4-point Bezier curves of the patch. Then use the *s* parameter to interpolate in the perpendicular direction between those 4 resulting points. The command line argument `-patch_tessellation <n>` controls the tessellation of the patch.

  You do not need to implement 3 dimensional editing of the control point grid for either the `SurfaceOfRevolution` or `BezierPatch`. (This is mostly a user interface issue, and has no "good" solution).

- Now you should be able to process the original teapot geometry (taken from *Jim Blinn's Corner: A Trip Down the Graphics Pipeline*) which consists of several Bezier curves swept around the y axis, and a collection of 4x4 Bezier patches. You'll just output a collection of polygons that can be rendered in your raytracer. Unfortunately the model is missing the bottom and has other nasty non-watertight intersection issues, so we can't render it with refraction.

Ideas for Extra Credit

- Implement a visualization of de Casteljau's algorithm for Bezier Splines and the mechanical construction subdivion for BSplines.
- Implement an interpolating spline and compare to the corresponding Bezier & BSpline curves.
- Extend your Bezier to BSpline and BSpline to Bezier conversions to handle curves with more than 4 control points. Note there are some tricky issues here with degrees of freedom and the number of control points.
- Implement Gouraud shading for curved objects. You'll need to store the ideal normal with each vertex when you create an .obj file, and parse this information when the file is loaded by your raytracer.

New Code Base for this Assignment

You may modify these files as you wish, but you should only need to create the new classes described above.

- vectors.h
- matrix.h
- matrix.C
- arg_parser.h
- glCanvas.h
- glCanvas.C
- spline_parser.h
- spline_parser.C
- triangle_mesh.h
- triangle_mesh.C
- main.C

- Makefile (for Athena Linux)

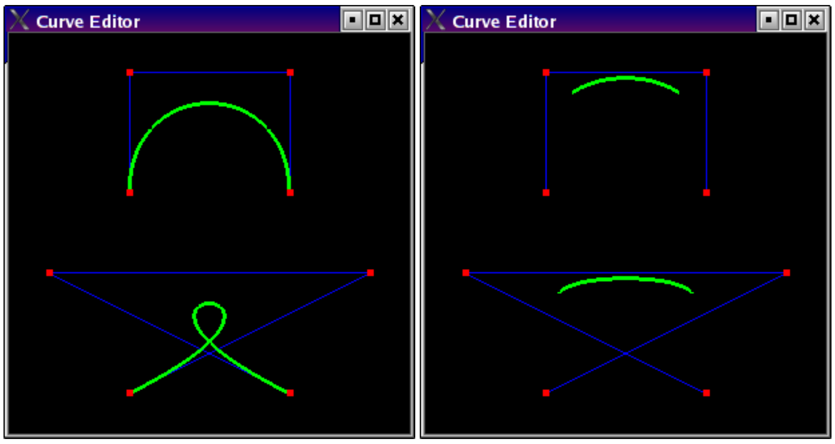
Input Files

- spline8_01_bezier.txt
- spline8_02_bpline.txt
- spline8_03_bezier.txt
- spline8_04_bspline.txt
- spline8_05_bspline_dups.txt
- spline8_06_torus.txt
- spline8_07_vase.txt
- spline8_08_bezier_patch.txt
- spline8_09_teapot.txt

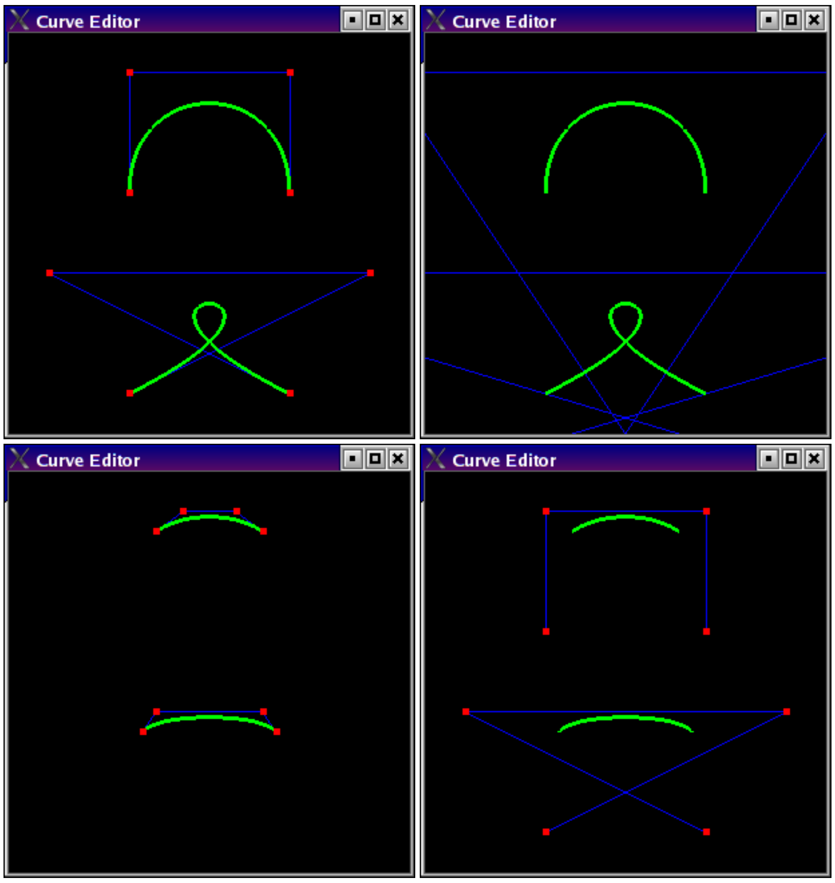
- scene8_06_torus_low.txt
- scene8_06_torus_high.txt
- scene8_07_vase_low.txt
- scene8_07_vase_high.txt
- scene8_08_bezier_patch_low.txt
- scene8_08_bezier_patch_med.txt
- scene8_08_bezier_patch_high.txt
- scene8_09_teapot_low.txt
- scene8_09_teapot_high.txt
- scene8_10_transparent_vase.txt
- scene8_11_reflective_teapot.txt

Sample Output

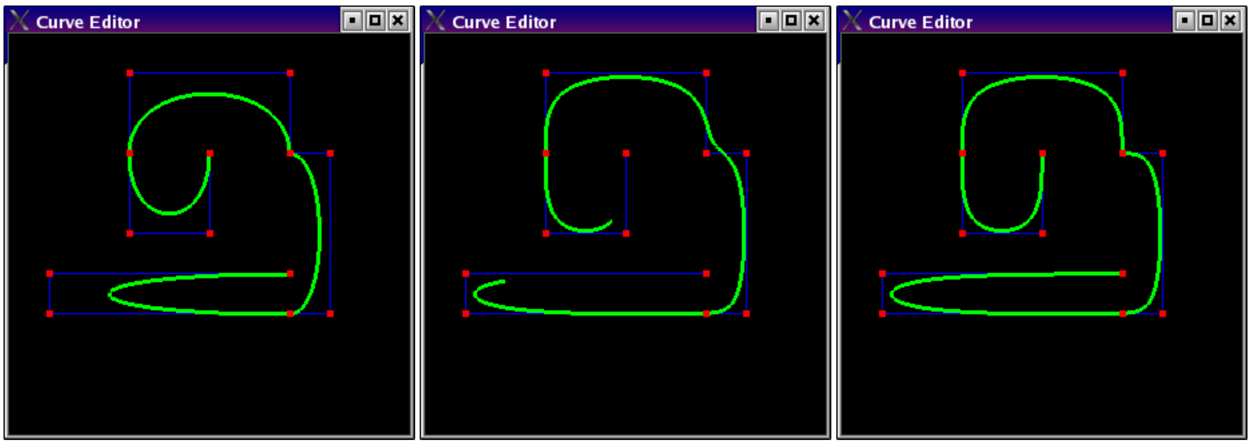
```
curve_editor -input spline8_01_bezier.txt -gui -curve_tessellation 30
curve_editor -input spline8_02_bspline.txt -gui -curve_tessellation 30
```



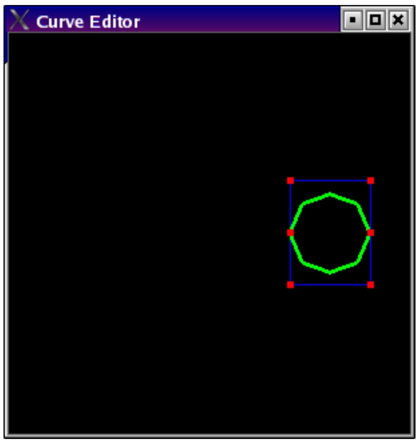
```
curve_editor -input spline8_01_bezier.txt -output_bezier output8_01_bezier.txt
curve_editor -input spline8_01_bezier.txt -output_bspline output8_01_bspline.txt
curve_editor -input spline8_02_bspline.txt -output_bezier output8_02_bezier.txt
curve_editor -input spline8_02_bspline.txt -output_bspline output8_02_bspline.txt
curve_editor -input output8_01_bezier.txt -gui -curve_tessellation 30
curve_editor -input output8_01_bspline.txt -gui -curve_tessellation 30
curve_editor -input output8_02_bezier.txt -gui -curve_tessellation 30
curve_editor -input output8_02_bspline.txt -gui -curve_tessellation 30
```

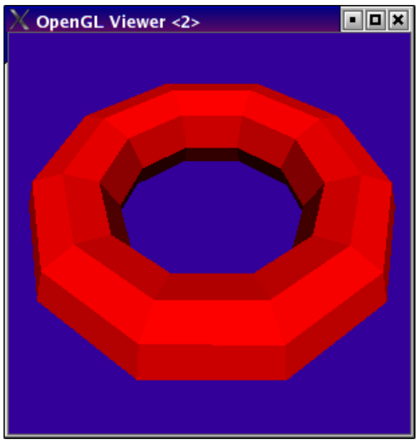


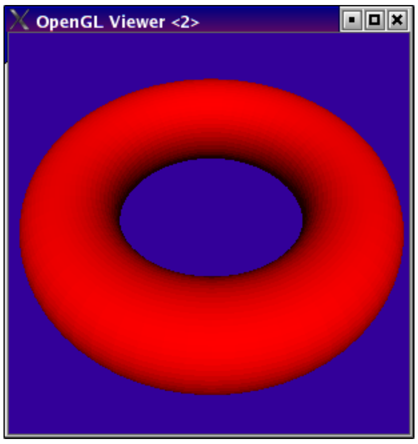
```
curve_editor -input spline8_03_bezier.txt -gui -curve_tessellation 30
curve_editor -input spline8_04_bspline.txt -gui -curve_tessellation 30
curve_editor -input spline8_05_bspline_dups.txt -gui -curve_tessellation 30
```



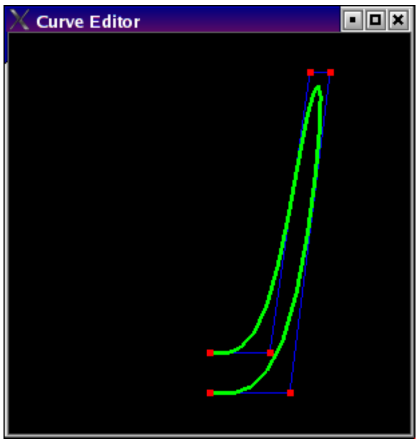
```
curve_editor -input spline8_06_torus.txt -curve_tessellation 4 -gui
curve_editor -input spline8_06_torus.txt -curve_tessellation 4 -revolution_tessellation 10 -output torus_low.obj
curve_editor -input spline8_06_torus.txt -curve_tessellation 30 -revolution_tessellation 60 -output torus_high.obj
raytracer -input scene8_06_torus_low.txt -gui -size 300 300
raytracer -input scene8_06_torus_high.txt -gui -size 300 300
```

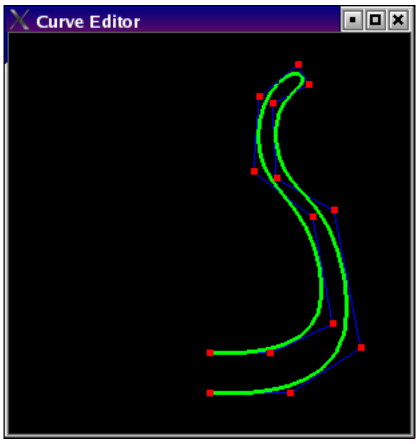


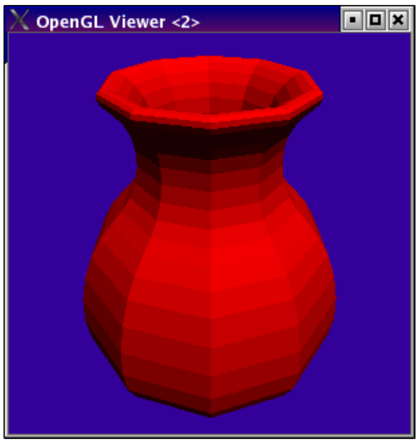


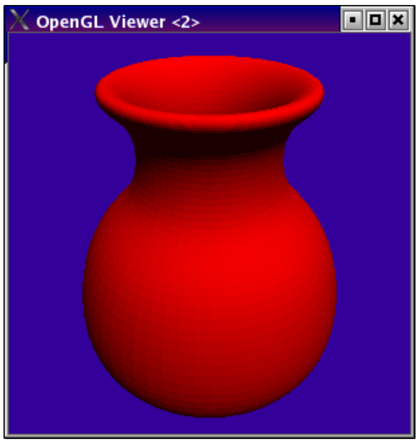


```
curve_editor -input spline8_07_vase.txt -curve_tessellation 4 -output_bspline output8_07_edit.txt -gui
curve_editor -input output8_07_edit.txt -curve_tessellation 4 -revolution_tessellation 10 -output vase_low.obj
curve_editor -input output8_07_edit.txt -curve_tessellation 10 -revolution_tessellation 60 -output vase_high.obj
raytracer -input scene8_07_vase_low.txt -gui -size 300 300
raytracer -input scene8_07_vase_high.txt -gui -size 300 300
```

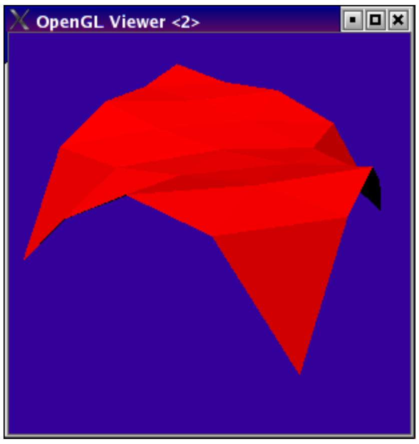


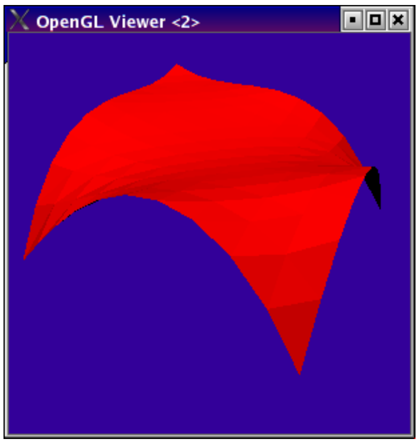


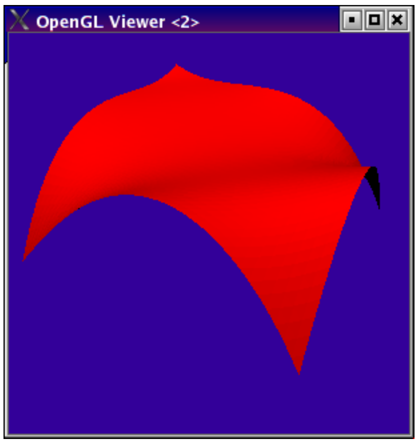




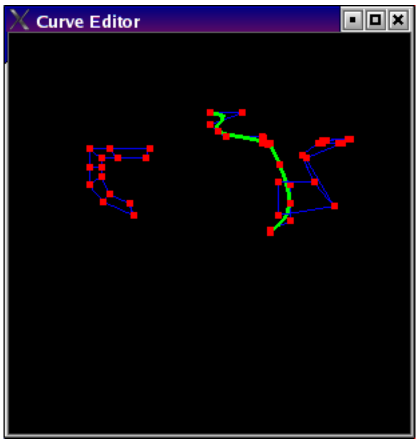
```
curve_editor -input spline8_08_bezier_patch.txt -gui
curve_editor -input spline8_08_bezier_patch.txt -patch_tessellation 4 -output patch_low.obj
curve_editor -input spline8_08_bezier_patch.txt -patch_tessellation 10 -output patch_med.obj
curve_editor -input spline8_08_bezier_patch.txt -patch_tessellation 40 -output patch_high.obj
raytracer -input scene8_08_bezier_patch_low.txt -gui -size 300 300
raytracer -input scene8_08_bezier_patch_med.txt -gui -size 300 300
raytracer -input scene8_08_bezier_patch_high.txt -gui -size 300 300
```

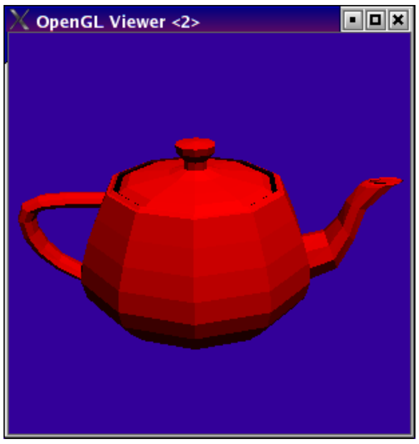


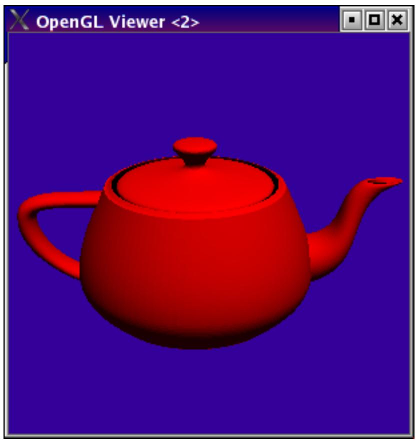




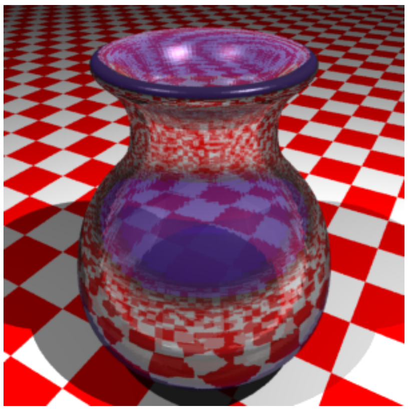
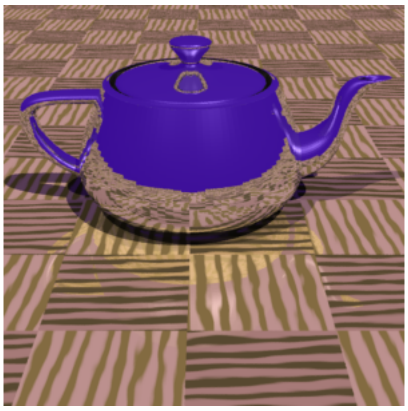
```
curve_editor -input spline8_09_teapot.txt -curve_tessellation 4 -gui
curve_editor -input spline8_09_teapot.txt -patch_tessellation 4 -curve_tessellation 4 -revolution_tessellation 10 -output teapot_low.obj
curve_editor -input spline8_09_teapot.txt -patch_tessellation 30 -curve_tessellation 30 -revolution_tessellation 100 -output teapot_high.
raytracer -input scene8_09_teapot_low.txt -gui -size 300 300
raytracer -input scene8_09_teapot_high.txt -gui -size 300 300
```







```
curve_editor -input output8_07_edit.txt -curve_tessellation 20 -revolution_tessellation 100 -output vase_very_high.obj
raytracer -input scene8_10_transparent_vase.txt -output output8_10.tga -grid 30 30 30 -size 300 300 -bounces 4 -shade_back -jittered_sampl
raytracer -input scene8_11_reflective_teapot.txt -output output8_11.tga -grid 50 30 30 -size 300 300 -bounces 4 -shade_back -jittered_samp
```

See the main Assignments Page for submission information.