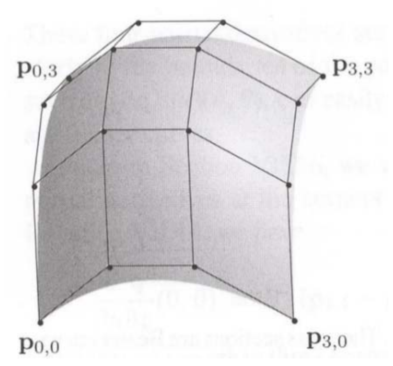
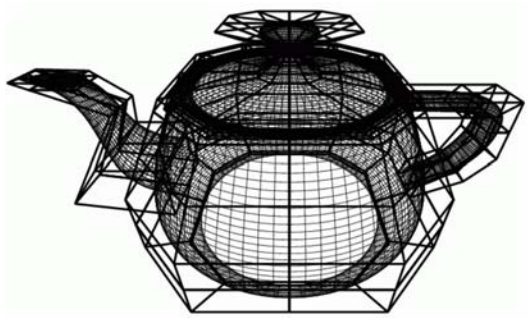
Computer Graphics

Lab Assignment #2 – Parametric Surfaces

In this assignment you will build methods to display curved objects based on parametric Bézier patches.



->



Requirement 1 (60%) – Display the "Utah Teapot" with Bézier Patches

Read the data file "teapotCGA.bpt" which defines the coordinates of the control points for each Bézier patch around a teapot model, and correctly display the result.

For this requirement you have to:
=> Add the functionality to display or not the control polyhedron cage.
=> Generate correct normal vectors such that the shading of your model looks correct.
=> Add the functionality to display or not all used normal vectors.
=> Allow the user to interactively change the resolution / number of triangles in the surface (to control this you may just use keyboard keys or add a slider to the user interface).

I suggest you approach your implementation one step at a time: first create your own classes and/or functions to draw one smooth Bézier surface from 16 control points, and once this is working fine, then load the dataset in order to display the entire model. Then work on the remaining functionality.

As in the first lab assignment, the recommended approach is to, every time you evaluate points in the surface, store triangles and normal vectors in the arrays of GsModel for display in the scene graph. To visualize the normal vectors and the control polygon you can use the SnLines scene graph node.

The selected data file, which was uploaded to Cat Courses, is from this website:
http://www.holmes3d.net/graphics/teapot/

Additional relevant information can be found here:

https://en.wikipedia.org/wiki/Utah_teapot
https://www.scratchapixel.com/lessons/advanced-rendering/bezier-curve-rendering-utah-teapot

Requirement 2 (20%) – Extensions

Implement one of the following options:

Option 1: Implement your own scene node to send the triangles and normals to OpenGL, and then for your next requirement below you will compare if you can achieve a scene graph node faster than using GsModel. I will be explaining in the lab how to create your own scene node.

Option 2: If you have not done yet shadows in your lab 1, here you have the chance again. I suggest you use a projection matrix to create a ground shadow by projecting all triangles of the scene to the ground plane. Shadow solutions based on shaders can be a topic for the final project.

Option 3: Add the option to let the user manipulate the vertices of the control polyhedron interactively. You may either implement your own solution to translate clicked points in the screen to vertex manipulation, or you can make use of the SnManipulator class. Just make sure you achieve something that works well. Study the "cameratest" example (which is included in the new sig version 0930) to learn how to compute a ray traversing the scene from the point the user clicked the mouse.

Requirement 3 (10%) – Explanation and Evaluation

As before, your submission must contain a readme file (.txt,.doc,.pdf) where you will **explain the main points** of your project, where to find the key parts in the code, and all other relevant information. This should not be long but it should provide enough information giving a good idea of the main points of your implementation.

You will also evaluate your project by writing in your readme file some **evaluation information including both text and a table.** The table will summarize with numbers the results of your evaluation. Examples of possible evaluations are to:
- Measure how much time each module of your project takes for different number of triangles.
- Measure how many frames per second you achieve in different computers and in different number of triangles approximating your surfaces.
- Compare the difference in performance when using two different approaches/implementations for a given part of your project.

The results should be summarized in a simple table and you also have to say **where in your code you implemented your measurement functions** which provided your numbers. Even if the measurement code is not used in your final version, you have to leave your measurement functions and calls there, just comment the calls which are not being used in your final version but leave them there. This will be important to check which exact operations are include in each computed time, etc.

Requirement 4 (10%) – Overall Quality

As always, your project has to look good and be well implemented with an organized code. Focus on achieving results that are interesting and correct.

Grading and Submission

Please read files grading.txt and rules.txt posted at CatCourses.

Additional submissions notes:

1. Make sure your submitted project compiles.
2. Run “cleanall.bat” before creating your zip file to submit
3. Include your project files but do not include sig

   Thanks!