CS 283: Assignment 4—Final Project

Ravi Ramamoorthi

As in CS 184, the final project is a chance to showcase what you've learned in the course, to develop a project of your choice. We provide some options here, as in assignment 2. However, note that (as with any of the other assignments), you can also propose a topic of your choosing, based on either a research project, or a desired implementation project. You are also welcome to extend one of your previous assignments (or do one of the components you didn't do in homework 3). The options noted below are deliberately underspecified, to give some flexibility, and allow you to use your creativity.

As in previous assignments, the submission is to be done by creating a website, including full documentation and examples (or a PDF), and including links to videos of the system running (if a real-time application). You should do this project in groups of two, but as before, this is not strictly required. As always e-mail the grader to submit the assignment.

The following are some of the ideas for the project. Note that you only need to do one of the following (or another of your own choosing). The first two correspond to the last two assignments given in the initial offering in Fall 09; you may want to take a brief look at the Fall 09 instantiation of CS 294-13 for more details.

- *Subdivision Surfaces:* This project aligns most closely with the geometric modeling aspect of homework 3. Subdivision is a new and very popular paradigm for shapes with complex topology. Given a control mesh, it creates a smooth limit surface. The basic idea is to implement Catmull Clark Subdivision with exact evaluation of the limit surface per Jos Stam's Siggraph 98 paper on "Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values."

  The first part is to write a basic subdivision routine. As with all subdivision schemes, you should operate on arbitrary input meshes with complex topology. You may want to start with basic Catmull Clark. Thereafter, include ways of "pushing the surface to the limit" after some number of subdivisions, per Stam's work. Then, develop code to compute tangents, normals and curvatures (Gauss and Mean) at the vertices. See if you can include some color coded plots, as in his paper. The input-output interface is up to you, but could for example, read OBJ or OFF files for geometry. You could use as examples many of the meshes in assignment 3.

  In the second part, we ask you to write code to move specified points of the control mesh to make the surface as smooth as possible—this typically involves minimizing surface area or integral squared curvature, but define a suitable error metric. You can use any suitable optimization method, and can use off the shelf code for this purpose (if in doubt write your own conjugate gradient descent solver from Numerical Recipes or Shewchuk's website). Your optimization can make use of discrete approximations to the minimized function but your visualizations (part 1 or the paragraph above) should be exact. Show some interesting surfaces that you make by optimizing various energy functions, including combinations of surface area and curvature. This second part is difficult, and grading will be based largely on apparent effort, so don't be too discouraged if not fully successful.

  You will need to carefully demonstrate this assignment, including example images with different parameters, shaded and wireframe examples, the control mesh and a few steps of subdivision, animations and still images for results of optimization and visualizations for gradients and curvatures.

- *Physical Simulation:* Physical simulation is a key component of modern computer graphics. This is a very open-ended assignment that asks you to implement any simulation (cloth, fluids, rigid, flexible bodies). Since the assignment is so open-ended, part of your goal will be to document a convincing writeup, showing the various steps towards creating your desired simulation. Grading will be based on apparent effort and quality of final results. You should prepare a video (still frames likely computed offline) of your final results and link it in from your website.

- *Animation:* (This is similar to what I gave as an option in CS 184, but since nobody did that then...) The goal is to build a basic inverse kinematics system. You may want to start with the analytic formulae for a two-joint arm, but you would ideally like to have a general solver for at least four joints with different lengths. The root should be stationary (attached to an immobile base). There should be some interactive or offline mechanism to specify the goal point in 3D (or create a procedural animation), and the IK solver should select joint angles to reach the desired goal. For submission, make a video that clearly shows your system working (and any problems if relevant), where the goal path is something complicated, that tests the full range of motion (a circle or large arc or figure-eight, not just a line). It would also be nice if you gracefully handled configurations that cannot be achieved (are out of reach). Make sure to try to cover the full range of the arm's operating space in terms of joint angles. Clearly, a full video documenting everything that was done is crucial. Note that the rendering can be very simple, reusing homework 3 from CS 184, homeworks 1 or 2 in this course, or any other relevant code; we want to focus on the animation.

- *Imaging:* Just like physical simulation, image acquisition and manipulation, and computational photography, are areas of growing interest in computer graphics. In this project, you should implement some of the basic imaging algorithms. As with physical simulation this is largely open-ended, and grading will be based on documentation and apparent effort. One suggestion is to do the HDR imaging method from Debevec 97 (if you don't want to take your own photographs, you can probably find some images on the web), and create a high-dynamic range image that can be displayed with any of the recent tonemapping algorithms. Another suggestion is to focus on texture synthesis, implementing at least the basic Efros and Freeman 01 method for image quilting, and at least one major extension.