CS434: Final Project Proposals

Project Proposal due March 10th (projects due Friday May 2nd)

1. Radiosity

Global illumination (and ray-tracing) provide alternate methods to generate realistic imagery. For example, radiosity produces a compact methodology for computing radiosity values per patch/triangle in a scene and producing very compelling imagery. For this assignment,

- You can solve for radiosities using a linear least squares solver (I can provide you wish sparse linear least squares optimization package (not mine, but developed by people at Stanford; however, we use it lots in our research and are very familiar with it). The computational cost of this method is high but once computed the scene can be reilluminated in several ways; or
- You can use a gather or shoot logic to compute the radiosity values incrementally.

As explained in class, radiosity computes a per patch radiosity value that depends on the radiance and irradiance of the entire scene. In the simplest sense, given N triangles, algorithm is $O(N^2)$. For this assignment, you are to implement some form of acceleration so that computations are possible even for models with N near a million triangles. Several options exists such as using octrees, adaptive subdivision, hemi-cubes, and other schemes. Your program should run on a single PC and also support dynamically changing the lighting conditions (once the radiosity computation is finished). Using GPU-based computing is an option and encouraged.

Additional features include supporting, aside for shadows, transparency, simulated reflections, and refraction, etc.

2. Photometric Stereo

Photometric stereo enables recovering the per-pixel normals of an observed scene. In this project, you can design a system to perform a photometric stereo reconstruction with unknown light directions. Further, you should integrate the normals so as to form a 2.5D surface.

3. Non-Photorealistic Rendering (NPR)

This project entails developing and implementing an NPR technique. The details can be found in the relevant papers. The tentative, but not all inclusive list, is the following:

- Exaggerated Shading: implement something akin to the Exaggerated Shading by Rusinkiewicz et al. 2006.

- Pen-and-Ink: implement some form of pen-and-ink using a precreated tonal map (e.g., textures of different density/darkness), Computer Generated Pen-and-Ink Illustration by Winkenbach et al. 1994.

- Contour/Silhouette: write a system to render the silhouettes/contours of arbitrary 3D models – you could require find polygonal tessellation but the contours should be smooth; Coherent Stylized Silhouettes, Kalnins et al. 2003.

- Using light transport/photometric stereo (with known lights) to obtain additional information for making an NPR.

4. Terrain Simulator

Implement a simple terrain editor and then various rendering engines for different part of the terrain: mountains, water/rivers, trees, plants, bushes, wind. For example enabling drawing rivers, pulling up mountains (idea: rivers could automatically form in between mountains), and put down trees using "tree paint" and bushes in open planes (idea: could be done using "sparse paint"). Water should appear to flow. You could also have some sort of wind that moves the trees. Optional: clouds in the sky.

Quite fun!

5. Soft-shadows

Implement the soft shadow rendering algorithm described in class. The rendering of the visibility masks should be done with HW support. CUDA implementation of the sampling location to grid and triangle to grid assignment a plus.

6. Motion tracker

Implement a simple motion tracker.

- Subject wears orange ping pong balls at joints (e.g. elbows, knees, wrists, feet, shoulders, hips, forehead).
- Subject wears dark clothing.
- Subject moves in front (on top of) green screen (green cloth).
- Subject acquired with two video cameras, e.g. two iPhones.
- Videos processed offline to detect trajectories of balls.
- Motion transferred to stick figure.

7. Hand tracker

Same as above but track single hand using colored tape circling finger joints.

8. Implement your own game

- 3D or 2D

- Freeware digital art (textures, characters, animation, sound), e.g. http://www.3dmodelfree.com/
- Score, at least 2 levels, at least 2 types of hazards
- Your own or freeware game engine (e.g. Unity, http://www.unity3d.com/)

9. Your Own Project

If you wish to propose and present your own final assignment, that is fine, **but you must provide a project description**.

10. Project Proposal Format

Your project proposal should be one-page long and should include the following sections:

a. title
b. summary of the objective
c. main components of the project to be developed
d. milestone for your mid-way project presentation
e. what your demo will consist of

If you make it a team project, each person should have a clear task in the project and students will receive independent grades. Please plan the project accordingly and include the distribution of tasks in the project proposal. Team projects would typically be two but a three person is possible – though we would advise against a three person team unless you have previous team experience.

Demo Day

Midway project presentations will be made in class during the week March 31st-April 4th. Final Project demo day is Friday May 2nd. Closer to that date we will setup a schedule via a democratic algorithm. Essentially, we will put up department-wide announcements, provide food and snacks, and have a demo-fest. Each student will present and demonstrate her/his assignment. You must also provide on that same day a CD/USB-key containing your source code, binaries, project and presentation – program must be executable from the CD/key. No extensions, no late penalties, no late passes, and no exceptions to this due date will be given.

If you have more questions, please see instructors.

Good luck!