Com S 336
Project ideas and timetable

Overview

Projects should be done in pairs. Find a partner soon. **Use Piazza if you don't know anyone.** *If you want to have a group of 3, and there is a good reason, let me know. If you prefer to work alone, that may be possible if there is a corresponding group of 3 to balance it.*

Your project should normally be a focused investigation into a particular technique that we haven't covered in class, but if you have an idea for something you'd like to try, I'll listen. All proposals need to be approved by the instructor.

The idea is that you learn how a technique works, do something original with it, and explain it to the class. Ideally, this would be done by adapting examples that are already familiar to the other students. For reference, here is a summary of the main topics we plan to cover in class in the remainder of the term:

Perlin noise
Bump mapping
Frame buffer objects and render-to-texture
Shadow mapping

(If what you're doing depends on one of the topics above and you need a preview before we get to it, let me know, I can provide examples.)

Each group will give an 8-minute presentation. Deliverables include your code, your presentation slides, and references. In addition, each group member must submit a statement explaining their own contribution, specifying how the submitted code differs from the reference code (as applicable).

Deadlines

- **Tuesday, November 8**: Who you're working with and what ideas you are considering. Please send email about who your partner is and what you are thinking about doing. This is to try to prevent duplication. You can talk to me in person about ideas for projects too, of course.
- **Anyone who does not have a partner by Tuesday night will be assigned one at random!**
- **Monday, November 14**: a brief, concrete description of what you plan to do and the main references you expect to use. Please turn in a paragraph with 3 or so sentences describing what you propose to do and what your primary sources will be. Do enough reading that you can try to be somewhat specific about what you want to try to implement, not just "I want to investigate how skeletal animation works." Submit this on Canvas.
- **December 6, 8, and 14** - presentations (*Wednesday, December 14, 9:45 - 11:45, is our final exam slot*)
- **Friday, December 9** - all deliverables submitted (even if you are presenting the 14th)

Scheduling, and the guinea pig bonus

Presentation times will be determined by **`java.util.Random`**. However, if you *volunteer* to present on the first day (Tuesday, December 6) your group gets a 10% bonus added to your score.

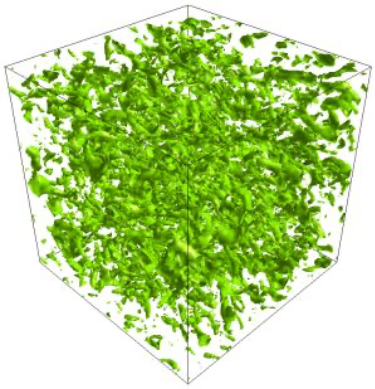
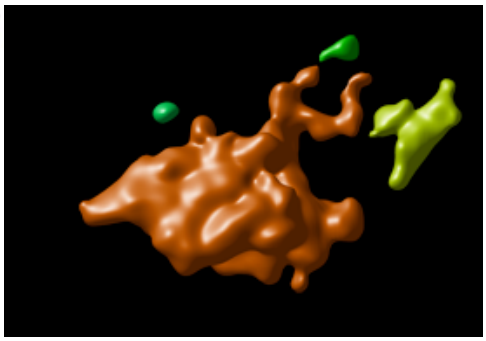
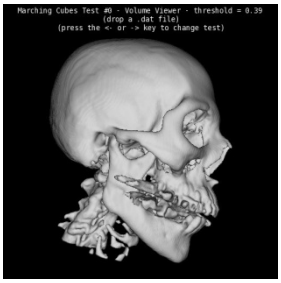
Some ideas ?

To get you started, here are a few examples of things that I think would probably lead to interesting projects. **Please do not limit yourself to this list, these are just off the top of my head!** There are many other things you might be more interested in.

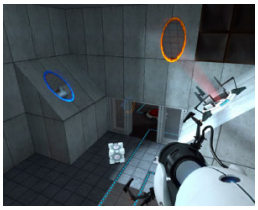
- The three.js examples directory is an incredible source of examples. The challenge with these is to make it into something original.
https://threejs.org/

- An excellent source of ideas to explore is the series of books GPU Gems (first 3 available online, scroll to bottom for table of contents)
https://developer.nvidia.com/gpugems/GPUGems/gpugems_pref01.html

- Another great source of ideas and resources is the web page associated with the book Real Time Rendering, http://www.realtimerendering.com/ .

Volumetric rendering (marching cubes, metaballs, isosurfaces)
All we can really render in the OpenGL pipeline is a polygonal mesh. What if all you have is a bunch of measurements or points in 3D space, such as from medical imaging or an implicit function? Some mechanism is needed to derive a mesh from that data. One important technique is the marching cubes algorithm, (but other techniques are possible too). E.g. see
http://users.polytech.unice.fr/~lingrand/MarchingCubes/accueil.html



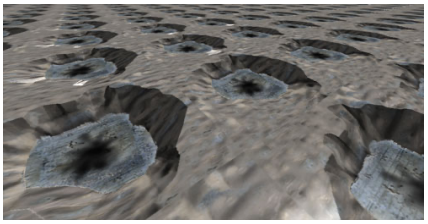
Rendering a "portal", as in the video game

This requires framebuffer objects and some calculations like those for mirrors.

Perlin Fire

E.g. see

http://developer.download.nvidia.com/SDK/10/direct3d/Source/PerlinFire/doc/PerlinFire.pdf



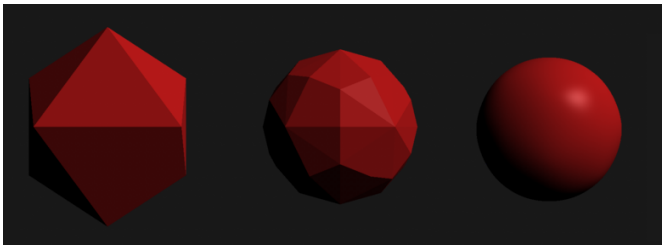
Parallax occlusion mapping

E.g. see

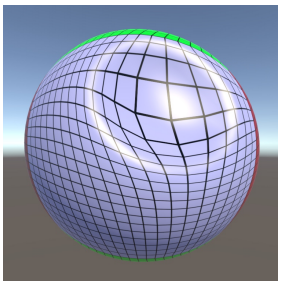
https://www.gamedev.net/articles/programming/graphics/a-closer-look-at-parallax-occlusion-mapping-r3262/

Tessellation shaders

You'll need OpenGL 4 for this, so it would have to be done in C++...



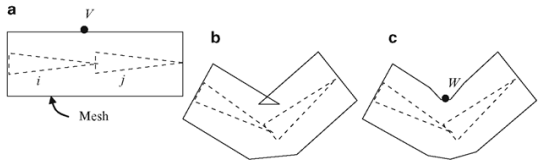
Catmull-Clark subdivision
E.g. see
https://graphics.stanford.edu/~mdfisher/subdivision.html



Mesh deformation
E.g., see
http://catlikecoding.com/unity/tutorials/mesh-deformation/

Water
There's a lot that can be done here, so define the scope of your project carefully.
http://habib.wikidot.com/
http://vterrain.org/Water/

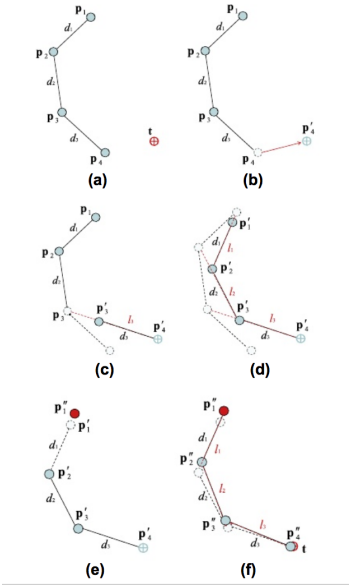
Generating fur, hair, etc.
http://developer.download.nvidia.com/SDK/10/direct3d/Source/Fur/doc/FurShellsAndFins.pdf



Skeletal animation

E.g. see

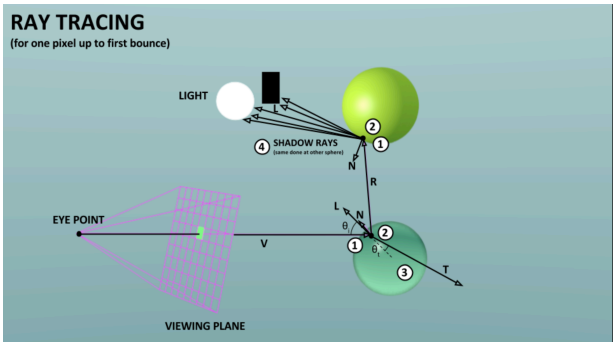
http://what-when-how.com/advanced-methods-in-computer-graphics/skeletal-animation-advanced-methods-in-computer-graphics-part-3/



Inverse kinematics (FABRIK)

E.g. see

http://www.academia.edu/9165835/FABRIK_A_fast_iterative_solver_for_the_Inverse_Kinematics_problem



Basic ray tracing

https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-ray-tracing/implementing-the-raytracing-algorithm

More ideas?

Particle systems
Cloth simulation
Billboarding
Non-photorealistic shading, toon shading
Motion blur
What is the stencil buffer and what is it used for?
Simulating a 3D texture in WebGL (and why)
Ambient occlusion
Using a geometry shader (Requires OpenGL 4, so would have to be done in C++)
How animation is done in practice
How to lay out texture coordinates in modeling software such as Blender or 3dsMax (this could have a big learning curve...)
If you have an interest in software design problems, an interesting problem would be how to extend CS336Object to include a specification of a model and how it should be rendered, in terms of surface properties and lights (along the lines of what three.js does, but way less complicated?)
Compute shaders (Requires OpenGL 4, so would have to be done in C++) - using the GPU as a tiny supercomputer...
Voronoi shattering
How does a physics engine work, and can we implement a simple one ourselves?