# Com S 336
# Project ideas and timetable

## Overview

**Projects will be done in groups of two or three. If you want to do yours alone you need to get permission from me. Find a partner asap**. *Use Piazza if you don't know anyone.*

Your project should be a focused investigation into a particular technique that we haven't covered in class, or it could be something else that I haven't thought about before. 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 a 7-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 19** : Please send email about who your partner is and what you are thinking about doing. This is to try to prevent too much duplication. You can talk to me in person about ideas for projects too, of course.
- **Anyone who does not have a partner by the 20th will be assigned one at random!**
- **December 10, 12, and 19** - presentations
- **Friday, December 13** - all deliverables submitted (even if you are presenting the 19th)

## Scheduling, incentives to present early

Any group that *volunteers* to present on the first day (Tuesday, December 10) gets a 10% bonus added to their score. Volunteers need to let me know by this coming Friday, November 16. All others will be assigned to one of the 3 available dates by `java.util.Random`.

## Some ideas?

To get you started, here are a few examples of things that I think would probably lead to nice 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.

- One excellent source of ideas is the library of example code for three.js. (Obviously, if you wanted to base your project on one of these examples, you'd need to take the technique and extend it or do something original with it.)
https://threejs.org/examples/

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

- Another resource 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

**Particle systems**

**Cloth simulation**

**Billboarding**

**Non-photorealistic shading, toon shading**

**Motion blur**

**What is the stencil buffer and what is it used for?**

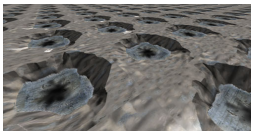
**Simulating a 3D texture in WebGL**

**Ambient occlusion**

**Using a geometry shader**
(Requires OpenGL 4, so would have to be done in C++)

**Rendering a "portal" (e.g. as in the video game)**

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

 **Parallax occlusion mapping**

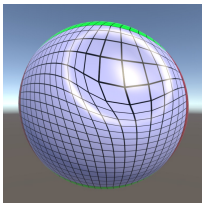
E.g. see
https://www.gamedev.net/articles/programming/graphics/a-closer-look-at-parallax-occlusion-mapping-r3262/

 **Tessellation shaders**

You'll need OpenGL 4 for this, so it would have to be done in C++...
http://prideout.net/blog/?p=48
http://codeflow.org/entries/2010/nov/07/opengl-4-tessellation/

 **Catmull-Clark subdivision**

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

 **Mesh deformation**

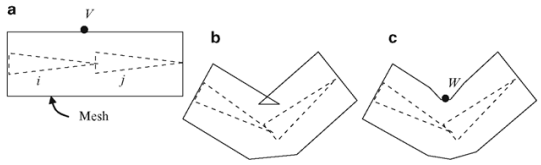
E.g., see
http://catlikecoding.com/unity/tutorials/mesh-deformation/

 **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/

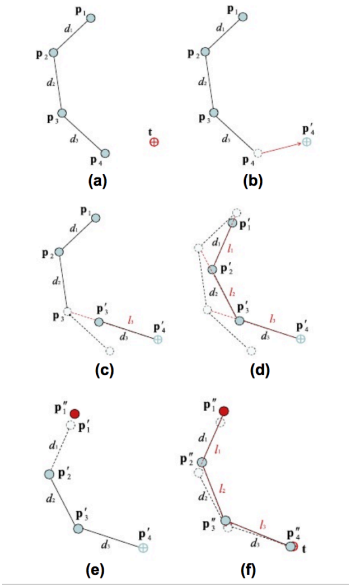
**Generating fur, hair, etc.**

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




**Bones (skeletal animation)**

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

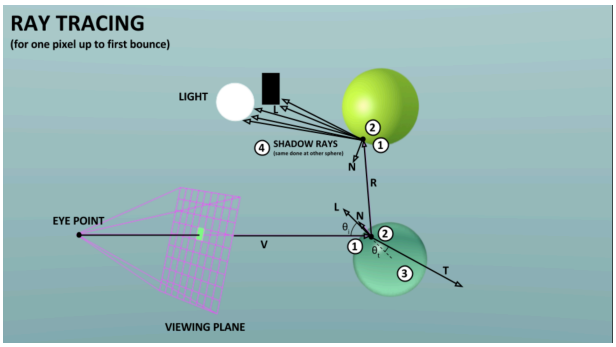



**Inverse kinematics (FABRIK)**

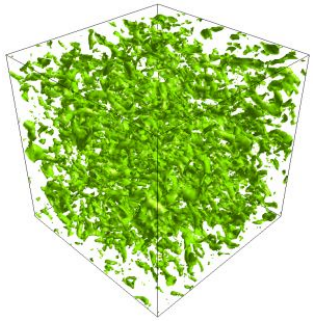
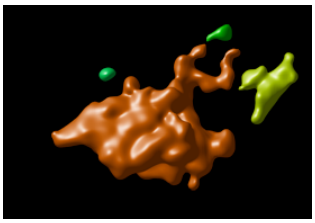
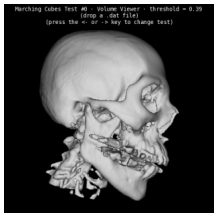
E.g. see
http://www.academia.edu/9165835/FABRIK_A_fast_iterative_solver_for_the_Inverse_Kinematics_problem

**Geometry instancing**

Geometry instancing is a method of rendering many copies of the same mesh fast by using only one draw call. This is done by using glDrawArraysInstanced() instead of glDrawArrays(), and in the vertex shader you are provided with a builtin variable gl_InstanceId to let you know what instance you are handling.
http://software.intel.com/en-us/articles/rendering-grass-with-instancing-in-directx-10/
https://www.saschawillems.de/?p=1852
https://keaukraine.github.io/webgl-fur/



**Basic ray tracing**
https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-ray-tracing/implementing-the-raytracing-algorithm
http://www.realtimerendering.com/raytracing/Ray%20Tracing%20in%20a%20Weekend.pdf

**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
https://threejs.org/examples/#webgl_marchingcubes