

Spring 2008
22C:251 Advanced OpenGL Techniques
Assignment 3

Due: Thursday, March 27th at 11:59pm

Goal: Learn to use render-to-texture and geometry shaders.

Note: You *may* work in groups of two for this project, but you will be expected to implement a larger variety of techniques.

Problem 1 (25 points): Use render into a texture to store your output for reuse in another pass. Use “ping-ponging” to bounce your output back and forth between rendering passes.

- **Part A:** Implement the “Game of Life” in a pixel shader. Every step in the game outputs a binary values (black/white) that are used as the input into the next step. For each step, a pixel’s output is determined by looking at its 8 neighbors from the previous step and using the following rules:

1. Any live (black) pixels with 0 or 1 live neighbors dies (white) due to loneliness.
2. Any live pixels with more than 3 live neighbors dies due to overcrowding.
3. Any live pixels with 2 or 3 live neighbors remains alive.
4. Any dead pixels with exactly 3 live neighbors comes to life.

Start using either a random binary image or load some examples I will post on the web page. Please slow the steps down so I can see them (say 5 or 10 steps per second).

Groups of two must also allow on-screen clicks to add 2×2 black (live) regions at those points.

- **Part B:** Implement a basic image processing program that loads an image and allows me to apply an arbitrary sequence of filters to it. These filters should be implemented as pixel shaders. Individuals must implement 3 of the following filters, groups must implement all of these filters:

1. Convert RGB to Luminance (grayscale) using: $L = 0.3 \times R + 0.59 \times G + 0.11 \times B$.
2. A 7×7 box filter centered at the current pixel (i.e., an average of 49 neighboring pixels).
3. A 3×3 median filter centered at the current pixel (convert to grayscale luminance first).
4. A 5×5 Gaussian filter centered at the current pixel. (See webpage for more details)
5. A 3×3 Sobel filter centered at the current pixel. (See webpage for more details)
6. A 5×5 Laplace filter centered at the current pixel. (See webpage for more details)
7. A Canny edge detector. (See webpage for more details)

Allow me to select the next filter to apply using either a menu or keystrokes, however if you bind these filters to keystrokes, please output text in the OpenGL window displaying the key controls.

Problem 2 (25 points): Write a program that loads a model from a file and displays it on screen, using a shader program to display *only the silhouette edges*. Silhouette edges are defined as edges where one of the adjacent triangles points away from the viewer (i.e., $N \cdot V > 0$) and the other adjacent triangle points towards the viewer (i.e., $N \cdot V \leq 0$).

- You will need to do this using a geometry shader. I suggest you pass in `GL_LINES` (e.g., `glBegin(GL_LINES)` or `glDrawElements(GL_LINES,...)`) and the geometry shader should output `GL_LINE_STRIP`. You would then need to set the maximum number of vertex outputs from the geometry shader to be 2.
- Your fragment shader can simply display the silhouettes in white.
- Remember, to use a geometry shader you *must* have a vertex shader.
- The tricky part of this problem is getting information about an edge's adjacent triangles. I will post some models that explicitly store edges and some code to read these models.

*Groups must also have a mode where the silhouettes are not computed from the eye's point of view but are computed from the **light's** point of view. You should write a different geometry shader that then extrudes these silhouettes into triangles in the direction away from the light (i.e., you are creating shadow polygons that could be used for shadow volumes). This geometry shader would not output a `GL_LINE_STRIP` but would rather output a `GL_TRIANGLE_STRIP` with a maximum 4 output vertices per input line.*

NOTE: Please include a README file describing the platform your program runs on as well how to compile your program. If you do not include all the models with your submission, tell me where the models must be for your program to find them.