March 3, 2023

Dr. Brian Ricks
6001 Dodge Street
Omaha, NE 68182

Dear Dr. Ricks:

The first month of my project has been productive. I spent the first several weeks reading about partial derivatives, navier-stokes, vector fields, and reviewing linear algebra. In hindsight, I should have started programming while reading. I have found that implementing the equations has been the most beneficial for understanding them.

I started implementing the shader code listed in [3]. This consisted of translating the code from HLSL to GLSL. I quickly ran into issues trying to debug the code and understand what I was implementing. I abandoned this approach and looked for a simpler step-by-step approach. I found [5] which provides a clearer explanation of the equations involved in fluid simulation and how to implement them. It also provides reference to [6] which I will use for the final report. I have decided to follow the basic structure from [5] because it divides the simulation into manageable steps. This structure from [5] is outlined below:

```
initialize color field, c
initialize velocity field, u

while(true):
    u_a := advect field u through itself
    d := calculate divergence of u_a
    p := calculate pressure based on d, using jacobi iteration
    u := u_a - gradient of p
    c := advect field c through velocity field u
    draw c
    wait a bit
```

Currently I am rendering the color field on a quad that fills the screen. My color field is stored in a framebuffer that is repeatedly used in the render loop. My code for the CPU is not included in this report but I will post all source code to github for inspection later this month. Inside the fragment shader I have implemented the velocity field as a static equation that determines the velocity based on the fragment's x and y coordinates. The next step is to represent the velocity field as a texture that can be changed over time. Once this is done, I will implement support for altering the velocity field with the mouse. Finally, I will complete the other steps of the algorithm: calculating divergence, pressure, and advecting the velocity field through itself.

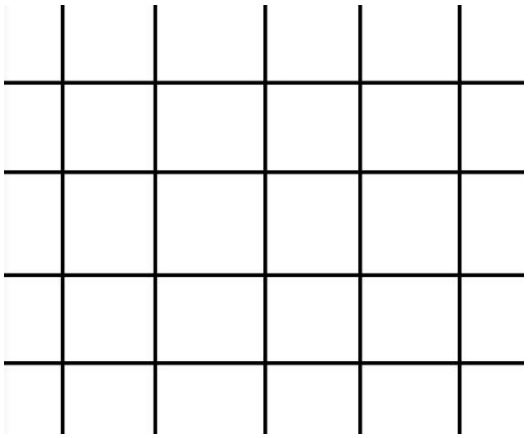
Listed below is the simple fragment shader currently used as well as several images from running different static velocity fields on a starting grid image.

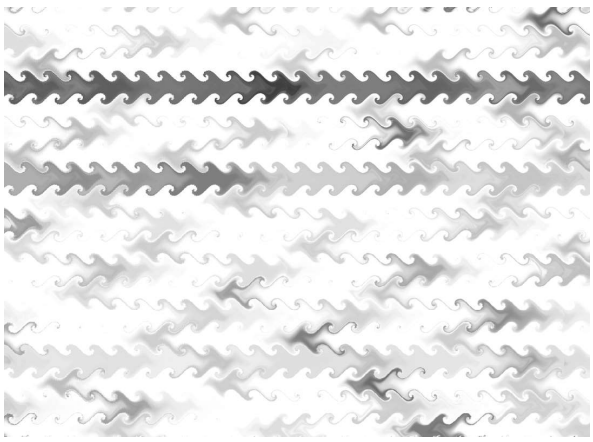
Sincerely,

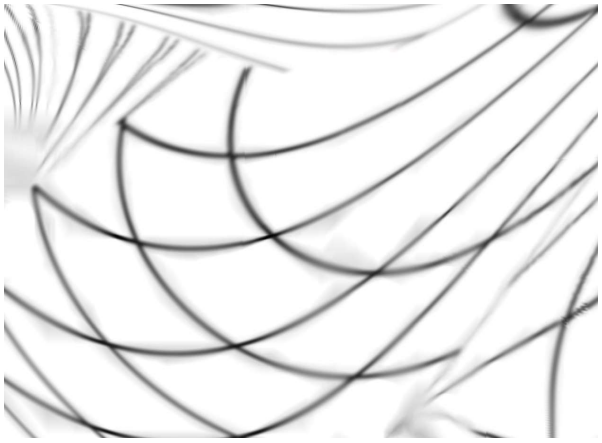
Philip Sigillito

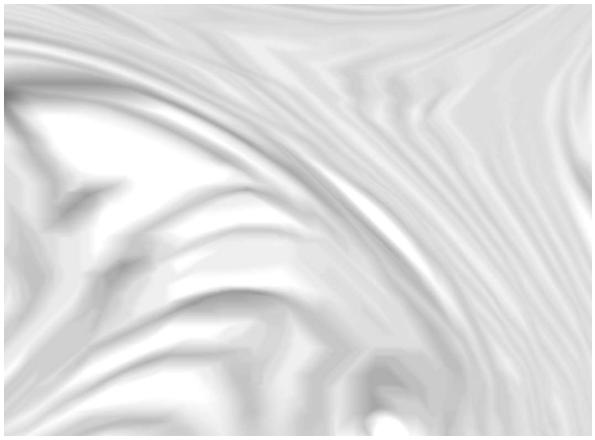
```
#version 330 core

out vec4 FragColor;
in vec2 TexCoords;
in vec4 gl_FragCoord;

uniform sampler2D screenTexture;

float pi = 3.1415;
vec2 centerScreen = vec2(800,400);

void main()
{
    //advect field scrolling right
    float advectX = -0.001;
    float advectY = 0.000;

    vec2 vectorFromCenter = gl_FragCoord.xy - centerScreen;

    //set static advection field,
    advectY = sin(2.0 * pi * gl_FragCoord.x)* vectorFromCenter.x *  0.05;
    advectX = sin(2.0 * pi * gl_FragCoord.y)* vectorFromCenter.y *  0.05;

    vec2 pastPos = vec2(TexCoords.x + advectX, TexCoords.y +advectY);
    FragColor = vec4(vec3(texture(screenTexture, pastPos)), 1.0)
}
```

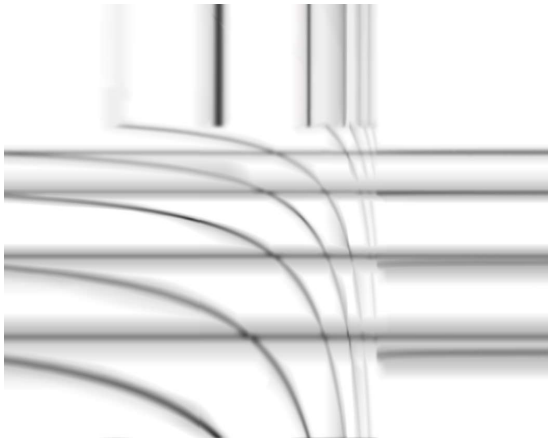
Starting grid pattern and various static velocity fields:











Works Cited/Relevant Resources:

[1] Stam, Jos. “Stable fluids.” *International Conference on Computer Graphics and Interactive Techniques* (1999).

[2] Stam, Jos.: Real-time fluid dynamics for games. In: Proceedings of the Game Developer Conference (2003)

[3] Martin Guay, Fabrice Colin, Richard Egli. Simple and Fast Fluids. GPU Pro, 2011, GPU Pro, 2, pp.433-444. ⟨inria-00596050⟩

[4] Nguyen, H. (2008). Chapter 38. In *GPU gems*. essay, Addison-Wesley.

[5] Wong, J. (n.d.). *Fluid simulation (with WebGL demo)*. Zero Wind :: Jamie Wong. Retrieved March 2, 2023, from https://jamie-wong.com/2016/08/05/webgl-fluid-simulation/

[6] Bridson, R., & Muller-Fischer, M. (2007, August 10). FLUID SIMULATION SIGGRAPH 2007 Course Note. Siggraph 2007.