

UCSD CSE 167 Final Project: Shadertoy animation

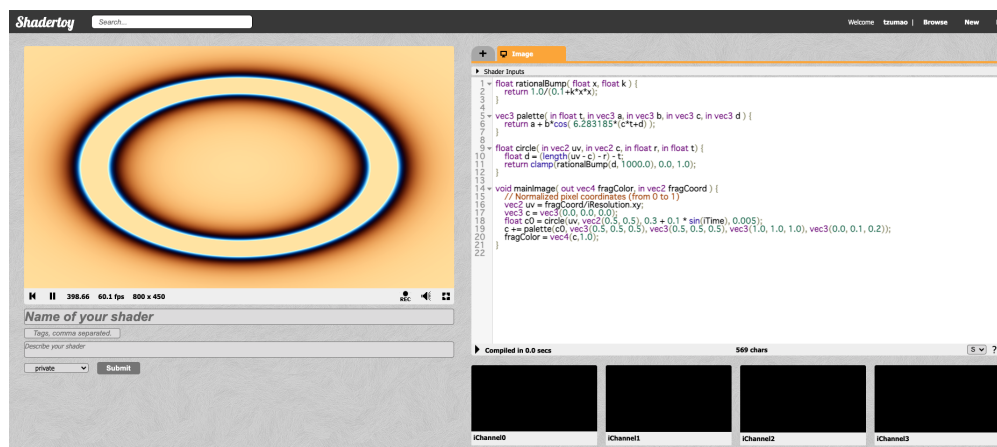


Figure 1: A random shader I made on Shadertoy.

In this last assignment/final project, we will explore on producing some animations by writing code. Instead of balboa, we will implement the animation in a very cool online service called [Shadertoy](#). In Shadertoy, all you have access to are a bunch of fragment shaders that draw to a quad. It is however a surprisingly powerful programming model that can be used to produce a wide variety of beautiful animations.

Before starting, take a look at the shaders other people wrote by clicking *Browse* at the top-right of the shadertoy site. Note that these are all open source, so you can try to read their code (some of them are really short!). Here are a few ones I like (in no particular order):

- [Seascape](#) by TDM.
- [Snail](#) by Inigo Quilez.
- [Hex Glitch](#) by igneus.
- [For the neon style enjoyers](#) by mrange.
- [Glass crash](#) by Dave.
- [Fractal land](#) by Kali.
- [Rainforest](#) by Inigo Quilez.
- [Starry Planes](#) by mrange.
- [Lover 2](#) by Fabrice Neyret.

Browse around and look at what other people created. Think about whether you can create the same.

To get started with shader art programming, here are two good Youtube videos by kishimisu. The first one introduces 2D shaders, and the second one introduces 3D shaders using raymarching.

- [An introduction to Shader Art Coding](#)
- [An introduction to Raymarching](#)

For the project, submit a shadertoy link (*please set the shadertoy project to UNLISTED, so that we do not spam the shadertoy browsing page, you can set it back to public after the class ends*) and a report documenting your journey: what have you tried, how did you reach your final art, etc. You will be graded

by both technical sophistication and artistic values: you can have something that is technically cool but artistically uninspiring, or you can have something that is artistically interesting, but does not have that much technical content. Your shader should be an animation, ideally with dynamic scenes instead of just panning cameras.

Below are some other resources that you can read/watch.

Storing and reading data across frames. Sometimes you may want to store information from your fragment shaders and read from others. You will do it by writing to a texture using an extra fragment shader, then reading it back in another shader. See [this shadertoy tutorial](#) for how to achieve that.

Live coding sessions. You can find some more coding sessions of how experts come up with their shaders on the internet. The Youtube channel The Art of Code also has some great content.

- [Shader Coding: Over the Moon - Part 1](#)
- [Shader Coding: Over the Moon - Part 2](#)
- [LiveCoding - The Universe Within - Part 1](#)
- [LiveCoding - The Universe Within - Part 2](#)

I also recommend the videos of Inigo Quilez, here are a few of them:

- [LIVE Coding "Happy Jumping"](#)
- [Live Coding "Sphere Gears" - Part 1](#)
- [Live Coding "Sphere Gears" - Part 2](#)
- [Painting a Landscape with Maths](#)

Particle systems. An easy way to get cool animation is to use something call particle systems, where you represent a list of particles with individual positions, velocities, and other states. The particles can influence each other to create collective behaviors. Here are a few example shadertoys that implement a particle system:

- [Interactive particles](#) by berelium.
- [Firework Show](#) by yibojiang.
- [Thermal particles](#) by wyatt.
- [Underwater Boids](#) by michael0804.

Mass spring systems. Mass spring systems are another kind of cool way to implement physics. Instead of a collection of particles, we build a graph of vertices and create forces to push and pull vertices that have an edge between them. Below are some cool shadertoys that implement mass spring systems

- [SpiderWeb](#) by middle.
- [Lazy Cloth](#) by 834144373.

Fluid dynamics. Fluids are another type of physical system that is relatively easy to implement (while the math is more involved), but can have very cool effects. A starter on this topic is Jos Stam's [Real-Time Fluid Dynamics for Games](#). [This playlist](#) contains some cool shadertoys on fluid dynamics. Correct fluid simulation is subtle and I do not guarantee the correctness of those shaders, but one good thing about art is that it does not have to be correct!

Rigid body simulation. Rigid body simulation is much harder in 3D than 2D. If you want to do this I recommend starting from 2D.

- [pool game physics](#) by archee.
- [2D Physics \(balls\)](#) by TDM.
- [physics engine](#) by archee.

Finally, [Shadertoy Unofficial](#) is a cool blog documenting some cool shadertoy tricks.