Blob World

CS175 Final Project - Alex Pedersen

In this project I make a game with many colorful, amorphous blobs that animate pleasingly and collide into each other. In the game, you can control one blob with your mouse to affect the scene.

I was inspired to make a "blob" based project using WebGL by this website (https://blobmixer.14islands.com/) and all (https://www.juicebox.work/) the (https://womp.com/) trendy (https://tryegress.com/) blobs (https://www.persana.ai/) I've been seeing on marketing websites.

For this project I've focused on two interesting extensions we did not discuss in class: 1. realistic noise, and 2. basic collision detection and deformation.

Setup

I decided to use Three.js (https://threejs.org) for this project because it provides nice vector primitives as well as much helpful scaffolding for tracking meshes, materials etc.

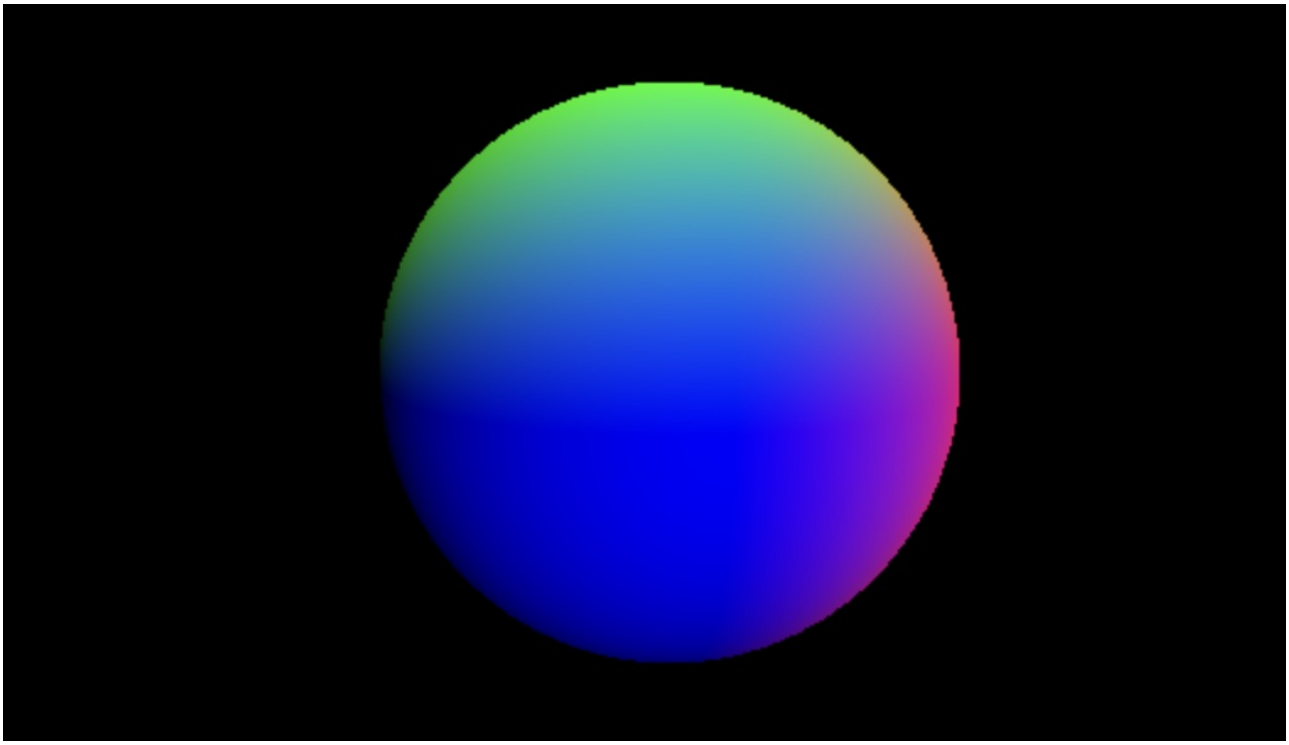
To begin I set up three global objects:

1. The renderer element, which is essentially a wrapper around HTML canvas
2. The scene which plays a similar role to our node tree from class
3. A 'perspective camera' from Three.js

For the initial "blob" I added a simple Three.js sphere object to the scene with a custom material. I added a vertex shader to calculate each vertice's position in object coordinates and used this variable in the fragment shader to vary color to create a basic gradient.

I also set up an animation loop and used Three.js object properties to vary the sphere's rotation on each paint.

At this point I had the following output:

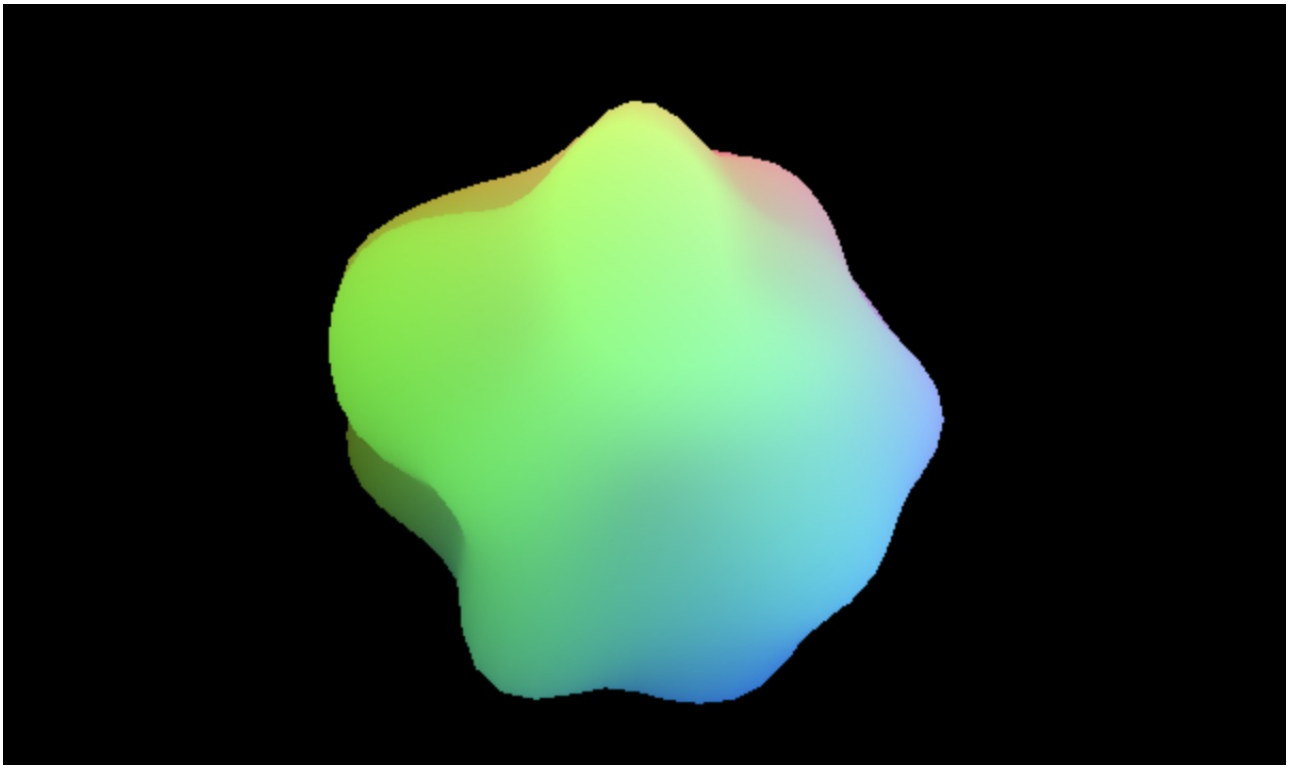


Noise

From here I wanted to make the blobs actually blobby rather than spherical. I did some reading about how this is can be accomplished with noise here (https://thebookofshaders.com/11/). I ultimately decided on simplex noise due to its efficiency. I found a canonical implementation here (https://gist.github.com/patriciogonzalezvivo/670c22f3966e662d2f83) for WebGL, so I introduced this into my shader.

I use the noise function by passing the three position components as the three arguments. This gives me a unique "variation" factor for each vertex which I add to its position in the shader. I also tweaked the color calculation in the fragment shader to make it more muted and relaxing.

At this point my blob looked like this:



Collision

Movement

In order to have collision there needs to be movement, so my first step was to assign the blob a velocity vector and use this vector to update its position at each frame in the animation loop.

From here, I introduced a bounding box vector. On each animation frame, I check if the position of the blob exceeds the limits of the frame. If so, I negate the corresponding velocity component.

Blob Collision

In order to implement blob collision, I added a second blob with all the same properties as the first one. From here, I implemented blob collision in a similar manner to bounding box collision: I check if the distance between blobs is less than the blob's radius (I don't use diameter because I want some overlap). If the blobs have collided I compute the vector of their collision from their positions. I use this vector to calculate the portion of each blob's previous velocity in the direction of the collision, which gives impulse magnitude. Finally, I update each blob's velocity in the opposite direction of the collision by the appropriate impulse value.

Notably, this simulation assumes all the blobs have equal mass and there is no inertia, but this is ok for my purpose.

Deformation

I also want the blobs to deform when they collide. To approximate this behavior, I have the blobs deform away from each other when their centers are close.

To implement this behavior I added a uniform variable in my vertex shader to track the world positions of each blob. In the shader, I calculate the distance between each vertex and the other blob and update the vertex position accordingly (the closer the vertex, the more I subtract in the direction of the other blob). I use

the smoothsetp function to produce a gradient effect.

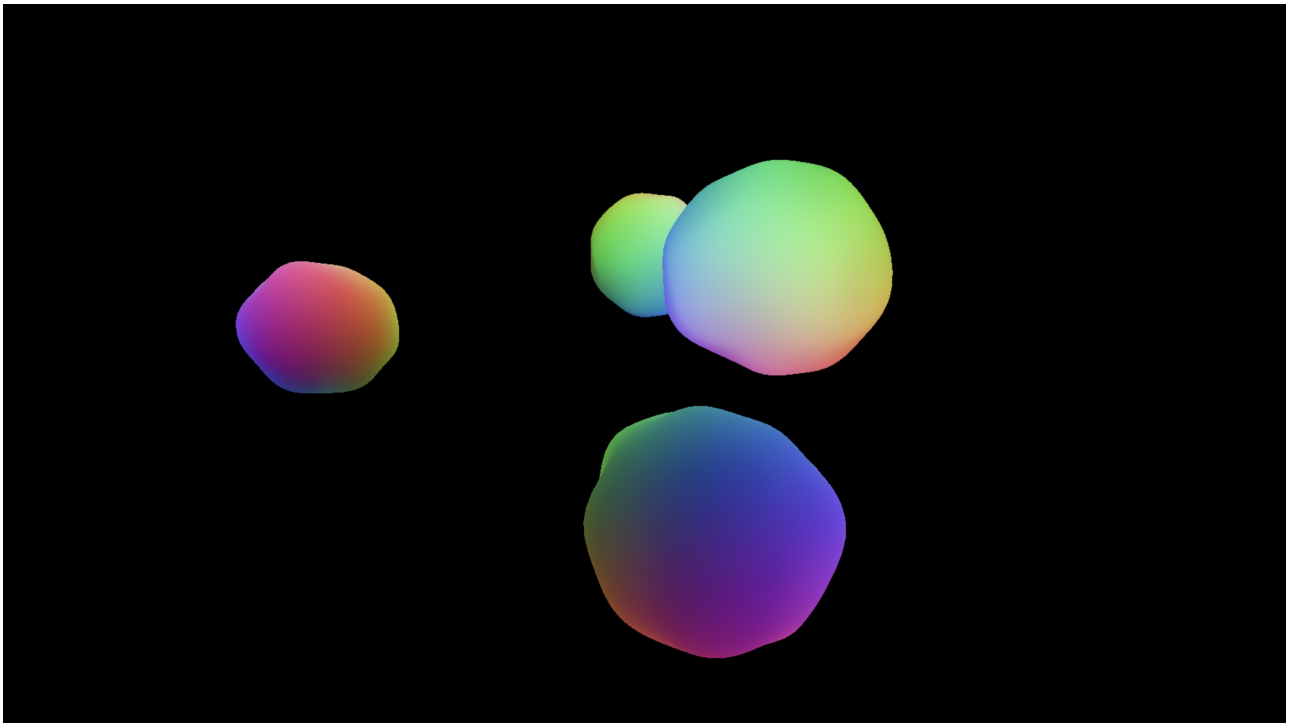
Final Touches

First, I wanted to add more blobs. To accomplish this goal, I

- Replaced many of my blob variables with arrays.
- Updated the animation loop to apply updates in a loop. Collision detection becomes n^2, but this seems ok for this number of blobs on my machine.
- Replaced the blob position variables in the vertex shader with an array of positions. I iterate over this array and calculate deformation much like before.

Finally, I wanted one blob to follow my mouse movements. To accomplish this behavior, I updated the animation loop so that the first blob's velocity is reset to some scaled vector of the difference between mouse position and blob position in X and Y (Z remains unconstrained).

Here is a screenshot of the final result:



Resources

- https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene
- https://thebookofshaders.com/11/
- https://discoverthreejs.com/book/first-steps/animation-loop/
- https://gist.github.com/patriciogonzalezvivo/670c22f3966e662d2f83