Algo Final Statement

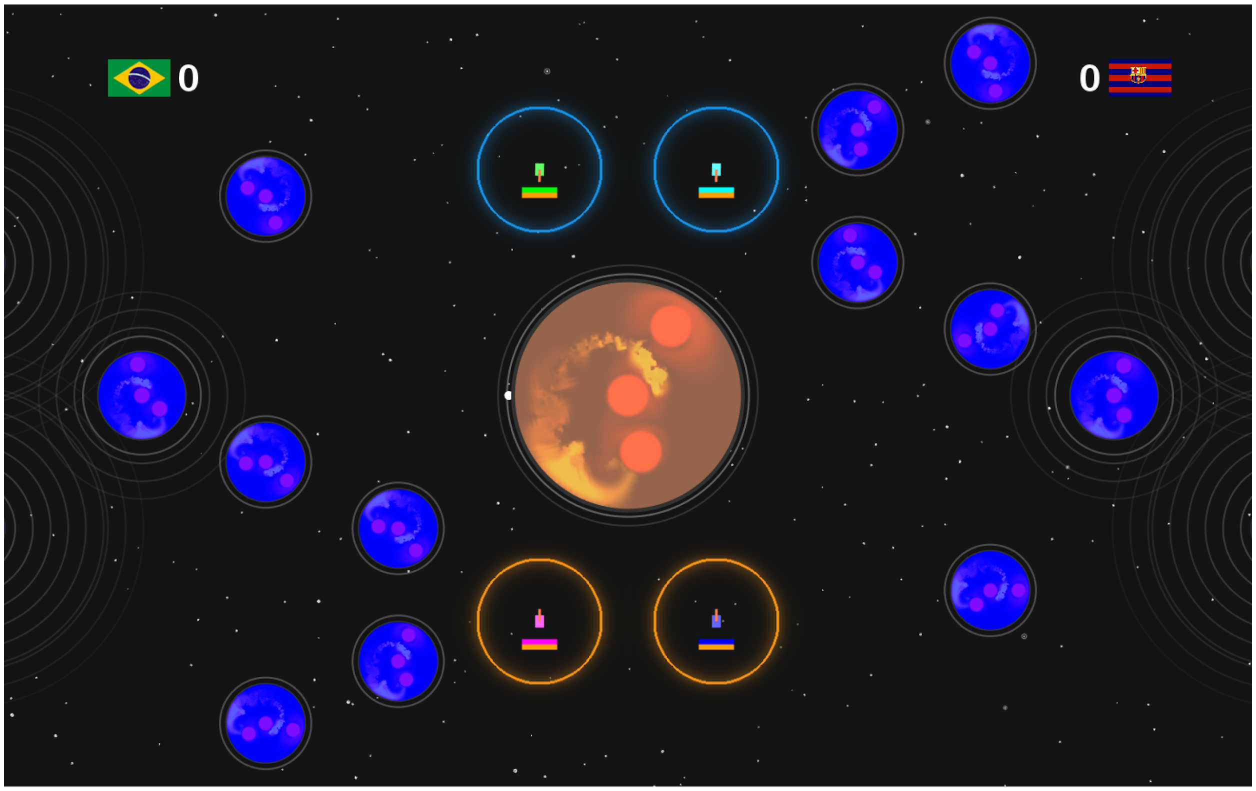
Note: This game uses OF 7 and it uses modified files from ofxFXFluid, which are included in my repo.

Anthony and I originally signed up for Algorithmic Animation because we had a really ugly looking game with clunky functionality, and we thought the class would teach us the skills we needed to make the game prettier and smoother. It taught us skills to make the game function better, like vector forces and using atan2 for mapping the joystick on the controller to the rotation of the player. Before Algo, my only experience with C++/OF was making this game, so we struggled a bit. We were already using forces and atan2, but it took us forever to sort of understand it, so because of algo, we now have a better understanding of how our game and how box2d work. It taught us various animation skills to make the game look much better.

Futbol Forever is a post-apocalyptic soccer game. Things got bad on earth. Real bad. The game is a look into what the future of sports may hold for us when we can no longer inhabit our home planet. The game makes uses of space physics such as gravity wells around planets, and a lack of gravity while floating in space.

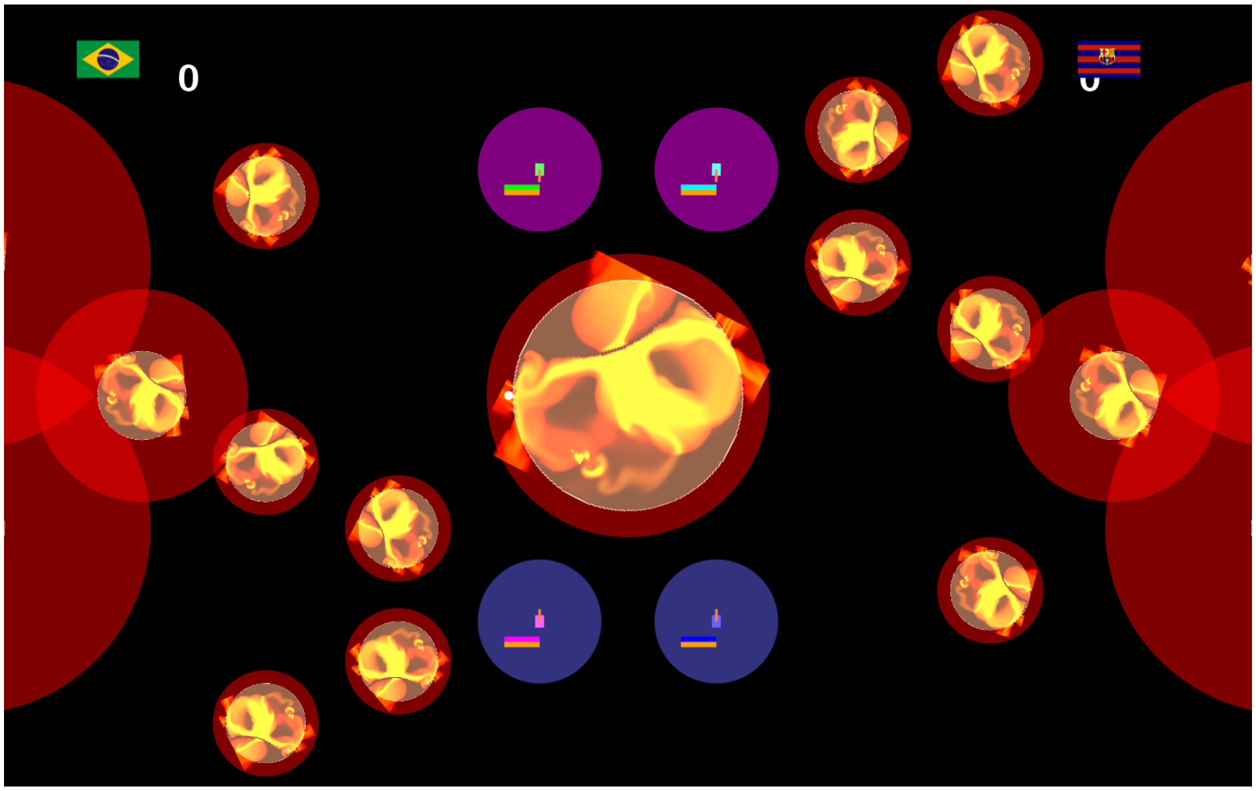
One of the ugliest things about our game was the solid colored planets. We discussed how we wanted it to look, and we ultimately decided on making a gassy planet. Initially we thought that it would be too taxing on the program to do this, so we thought we might make a video of a gassy planet. I started with some fluid dynamics called ofxMSA, but it was a tremendous mess. Then I came across a plugin called ofxFxFluid which was actually made by a second year DT student, Patricio Gonzalez. This proved to be a challenge that took quite a bit of time to implement. Many hours. However, it was worth it in the end, and I learned a bit about textures and shaders, and *a lot* about FBOs, which are awesome. The fluid plugin was based on this article from a book called GPU gems. I tried to read it because I wanted to understand how the plugin was working (or not working in my case). It was really hard to understand though, because I don't have a math background. Either way though, I feel like I had a confusing intro to openGL and the next time I try to learn it will be easier (that's how it was when I learned programming anyway).

I didn't have too much trouble, getting it to work in a separate app, but importing into our game was very troublesome.



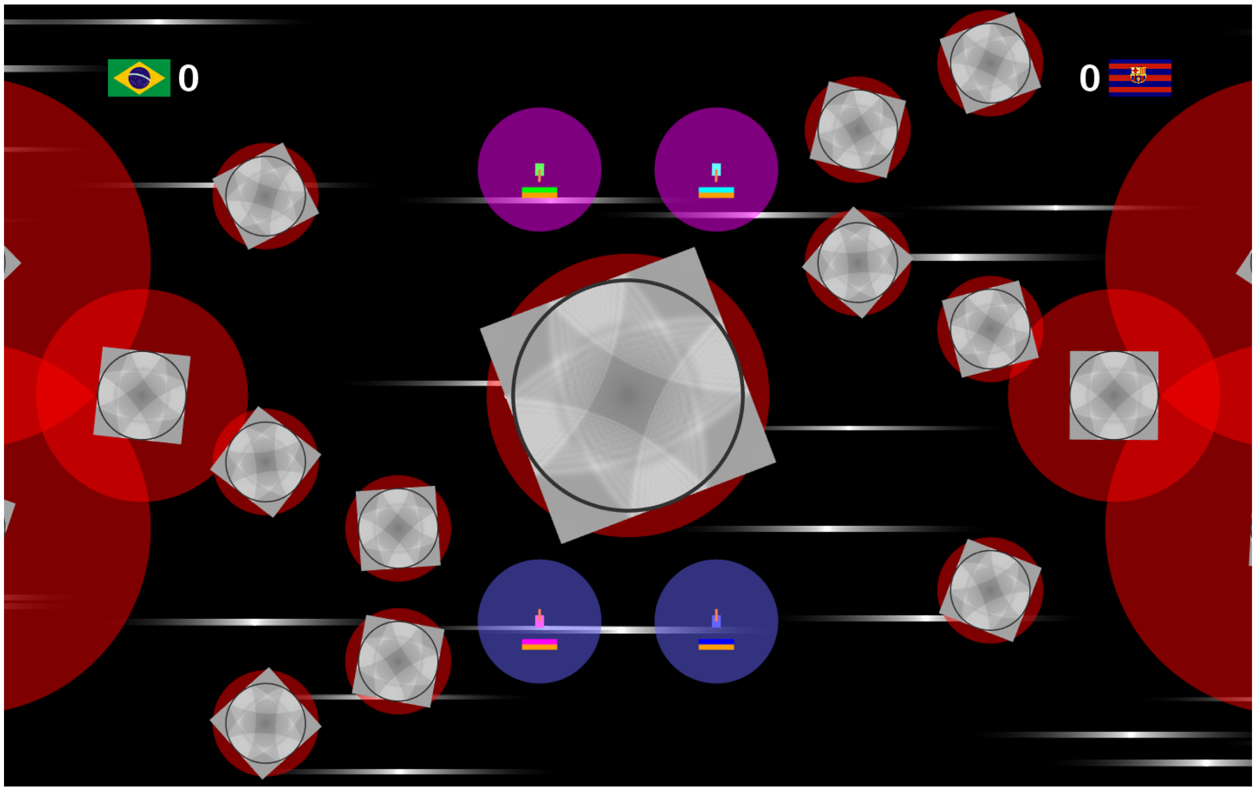
This is how the looked at first. After spending hours trying to get it to work, I found Patricio and he sat down with me and we actually edited the openGL blend settings in his plugin together. It was awesome to be able to get one on one help from the creator. I'm going to included the files we edited in my repo in case you need it when you look at our game.

This is one of the funniest accidents that has ever come out of my code. It was from trying to make the gassy planet. That poor soul…

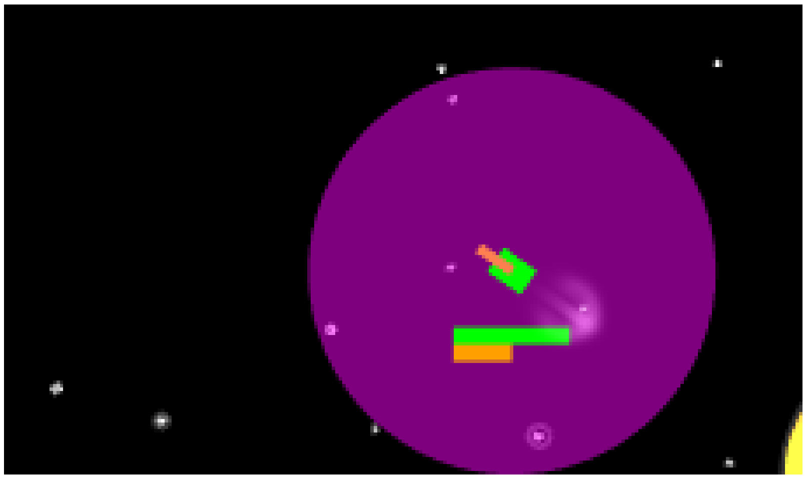


The plugin uses FBO, so I am only doing the calculations for the planets one time, and then I can draw it as many times as I want without slowing down the framerate. You were totally right. Drawing it small and scaling it up worked perfectly. In order to make it less obvious that the planets are exactly the same. I randomly rotated FBOs and set the large planet in the center to a different color. For some reason, setting ofSetRectMode broke the plugin (as you see above), so instead, I figured out a way to access the FBO element using pointers and I was able to change the anchor point of rotation, so that they would rotate from the center and therefore stay on the planets. I wanted to slow down the gas, but I couldn't figure out a way to do that that allowed the gas to fill up the chambers fully.

I tried to use another of Patricio's effects called ofxRipple to make the gravity wells, but after an hour or so of no progress I decided to cut my losses. Below is what it looked like when I stopped.



After I got the planets looking great, I wanted to add an animation for when the player boosts. At first I was going to use a particle system, but then I decided that it would look more realistic if it looked like smoke, and the liquid I had been using before. I had to make 4 separate FBOs for this, because every player needs his or her own animation since they won't be pressing the boost button at the same time. I'm actually still having a really frustrating problem with this. Below is an image of what the smoke boost looks like for players 2, 3, and 4. It animates beautifully and I'm really really happy with it. However, it doesn't work properly for player 1. It is just spits out a static fbo of smoke that looks terrible (On Anthony's computer it just pits out a solid black FBO, so I'm not sure how it will appear on yours). This makes no sense to me because the FBOs are in the player class, so I cannot figure out why it would be different. Oh well.



This project does not stop just because the final is over. My next plan is to make the controls a little simpler by using atan2 to help the player rotate around the planet. We also will continue working on the visual aspect. We need a start screen that displays the controls, and we need reset functionality.

Anthony worked on the background and gravity wells, which also both look better now, but you can read about that in his paper.

You need wireless xbox 360 controllers to play the game, as well as a wireless receiver for windows (and a mac driver for that)

The controls of the game are:
A: Boost
B: Attract (ball or your teammate if they get stuck in orbit)
X: Shoot
Left Joystick: direction
Left and Right Bumper: Rotate around plannet.

Thanks for the great semester!