CIS 700:
Final Project Report

Emily Vo
Suzanne Knop
Andrea Lin

Link to Youtube demo:
https://www.youtube.com/watch?v=yWhc-ZxIrWA

Part 1: Design Document

Team: Andrea Lin, Suzanne Knop, Emily Vo

Introduction

What motivates this project?

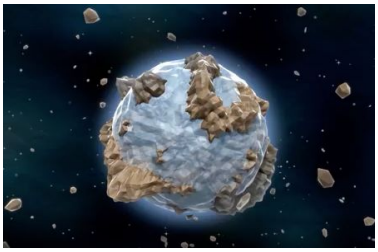
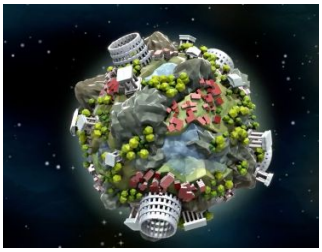
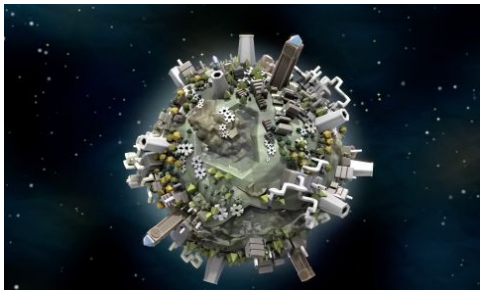
- A fun way to combine all of the skills we acquired over the course of the last few weeks (Noise, Shaders, Animation).
- It'll challenge us to make really stylized environments into a modular world.
- It's pretty cool. It's a good way to showcase our individual aesthetics and technical skills, but with our unified theme and focus on transitions, we will have a pretty polished final product.
- Learning how to use audio to procedurally drive the animation and movement, so that very little is hard-coded.

Goal

Create a group of related environments that is altered based on music data in the shader. We are planning to modularize the project by creating separate assets, which will be joined together in the end to create unique planets. These unique planets will be animated in interesting ways, using the music data to drive the animation. We will transition between each world using interesting animation!

Inspiration/reference:

- https://www.youtube.com/watch?v=EzsG1uqfDTQ
- https://www.youtube.com/watch?v=qARpMpHskTM
- https://www.youtube.com/watch?v=rmN8DHZYNCg

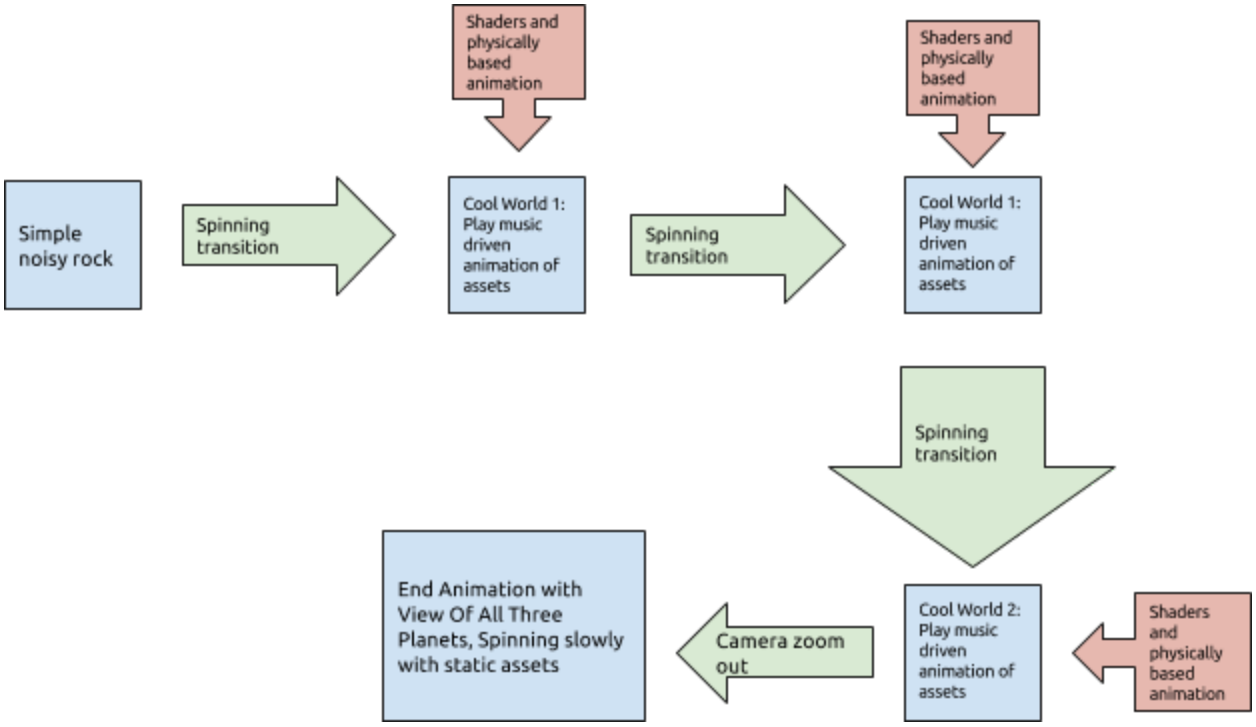
Specification

- Procedural Asset Generation - modularized components will be built that can be easily spawned and combined with others to create interesting environments for each planet
- Music Visualization - animated planet and shader features based on music (color, movement, shape/size)
- Transitions between planets - camera animation
- Post-processing shaders that add character and that react to the music.

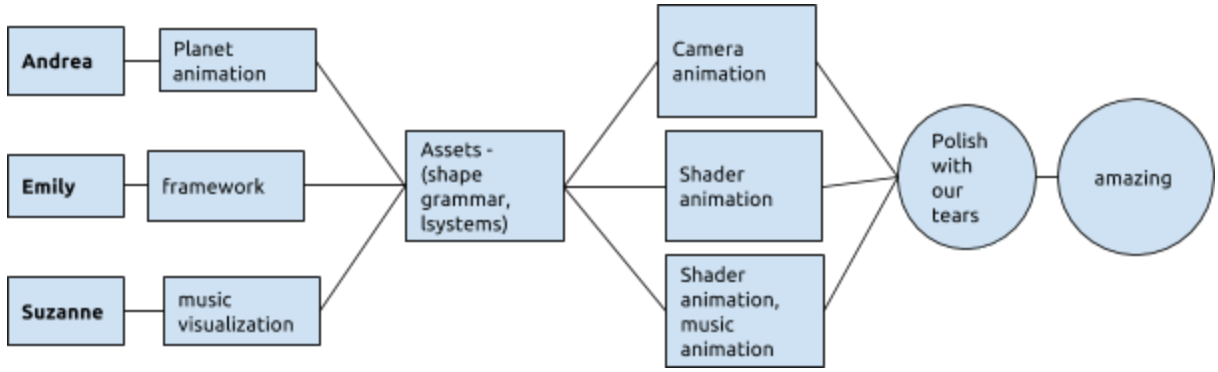
Techniques

- Toolbox functions to generate assets
- Music visualization techniques
    - Autocorrelation pitch detection algorithm
        - http://www.phy.mtu.edu/~suits/autocorrelation.html
        - https://developer.microsoft.com/en-us/microsoft-edge/testdrive/demos/webaudiotuner/
- Camera transformations for transitions
- Post-processing shaders
    - Dots shader: using sin & cos to create a dot pattern
    - RGB shift/chromatic aberration: separate out R, G, and B values from a pixel
    - Scanlines: use square functions and mods to create strips of evenly spaced discoloration

Design: Program Flow



Design: Team Workflow



Timeline (modified for actual progress)

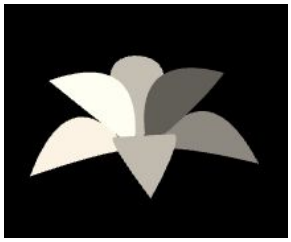
	Group goal	Andrea's Tasks	Emily's Tasks	Suzanne's Tasks
Week 1 (due 4/18)	Have framework completed with modularized code for each world and their assets. Want to make it easy to reuse assets and add to them, while also making it easy to generate new and interesting planets. The framework will be focused on also making sure that animating things based on music and time will be easy. We will also begin working on assets and aim to have at least one of them complete for each person in our group. We also want to create our transitions between each world. **Features:** ● Pregenerated assets that spawn on the worlds. ● Using noise to generate our basic terrains.	**Sphere animation** (spinning for transitions between each world). Spinning while despawning assets simultaneously. **Asset development** ○ Develop terrain (water potentially) ○ plant ○ Mountain **Camera Controls** create a class which can control the camera movements (zoom in, zoom out, pan)	Framework: ● World Interfaces ● Asset Interfaces ● Interfaces will make it easy for importing new assets, spawning new assets, and accessing sphere geometric features ● Interfaces will make it easy to call an "animate" or "tick" function for each asset ● App and each function will keep track of time and duration ● Complete flower asset using toolbox functions and flower planet using the framework I implemented ● Made a simple world for group mates to use as an example on how to make worlds from the framework ● Implemented simple rotations for the worlds. ● Implemented proper alignment with the normals for each asset, but allows for local transformations between multiple meshes in a single asset.	**Audio extraction:** Use the Web Audio API to load mp3 files and create audio nodes. **Audio analysis:** Use the analyzer node and byte frequency data to get the amplitude as the song is playing. Make this function available and modularized.
Week 2 (due 4/25)	**Goals for Demo**: Data will influence the appearance and movement of the planets and their assets. We will showcase a rough version of the entire combined animation, including basic planets with their assets spawned onto them.	● Camera movement / controls ● work on assets for water world ● Continue working on water world (procedural color, displacement)	● Complete procedural crystal geometry generation ● Shade crystals with animated iridescence ● Updated framework so that rotating a planet will not affect	**Audio analysis:** Implement a pitch detection algorithm, and use this to get color data which can be used for various things (such as the background, or for

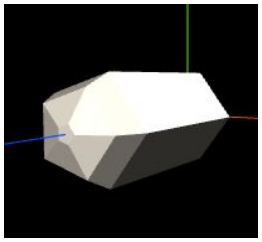
			the alignment of an asset. That is, it will still remain aligned with the normal correctly. ● Basic movement using CPU calculates of assets along the normal. ● Fixed issues with mesh loading in the node modules. ● Implemented functions that assist in resetting matrices of the mesh while maintaining transformation - useful for asset generation and placement.	later shaders). Also, extract BPM info from the audio which can be passed to worlds to influence how fast they spin.
Week 3 (5/3)	**Goals for Demo:** Integration & polishing. Planets and individual animations should be complete at this point, so all work will be towards having the final, polished animation. We hope to choreograph transitions with audio.	● Transitions between planets ● Water world assets (koi) ● Background (extra geometry, background color)	**Music:** ● Fixed functionality of picking and choosing current song. ● Passed the music data directly to the world classes so they could use the data in whatever way is appropriate for their look and feel. ● Implemented the animation using the music data for each planet (flower, crystal, water). Data was used either as an offset from the point on the sphere, or the size of the asset itself. **Geometry and Choreography:** ● Continued to fix alignment issues of assets with the normal when rotating and accelerating a planet. ● Fixed the camera class to use a proper lerp. It calculates a normalized t value instead of a fixed step for camera movement. This allows for cameras	**Post-processing Shaders:** Make a post-processing shader framework to make it easy to add new shaders, and create a variety of shaders that react to the audio information as it is playing.

			to appropriate reach their destination, but also with a specified time. ● Implemented all of the planet choreography with camera transitions. ● Implemented crystal explosion on the CPU.	

Part 2: Results

Assets:
Assets are placed procedurally on planets, and move/warp based on audio input

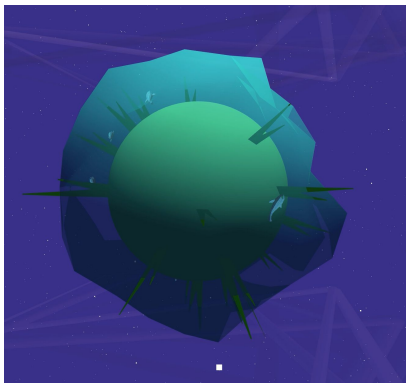

Flower

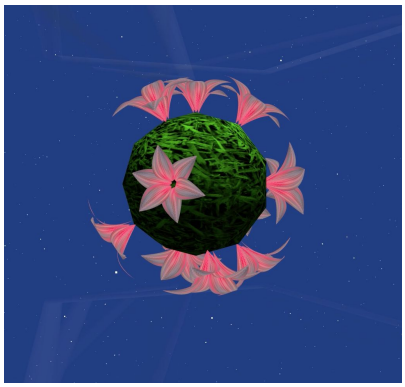

Crystal

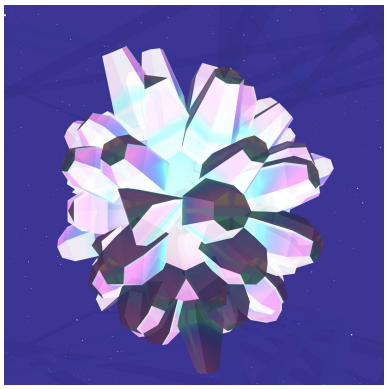

Koi


Seaweed

Planets:
Three worlds created with various procedural elements: water, flower, and crystal world





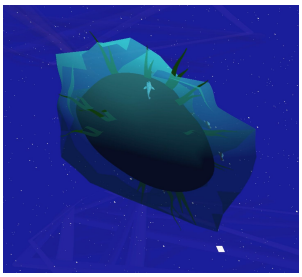


Water World: fbm-based water, procedural coloring, seaweed assets created procedurally, koi fish color based on pseudo-random noise function

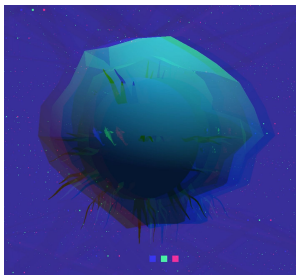
Flower World: Flowers are procedurally generated with toolbox functions, as individual petals. The placement of the petals, spawning, and alignment with normals is procedural.

Crystal World: The crystals are procedurally generated by parametrically determining the points of a 2D shape, and extruding it. There was variance in the number of points, the size of the bevel, the size of the crystal, and the number of segments. The crystals and planets are iridescent, and the uv's also are offset by time-based noise.

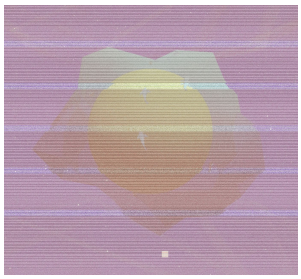
Post-processing Shaders:
Shaders react to audio data and can be easily paired with different songs, changing when the song changes.



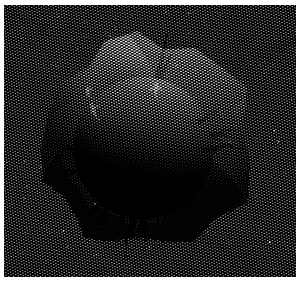
Vertical roll and warp/distortion - used time & audio data



Chromatic aberration (RGB shift) - used audio data



Scanlines, grain, and brightness / distortion glitches - used audio data



Dots (grayscale high contrast & full color saturated)

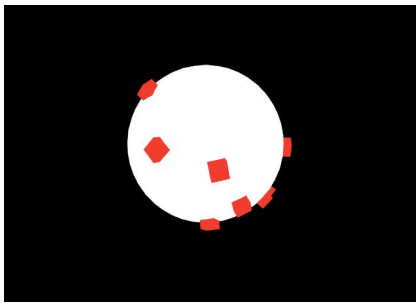
Audio Analysis:

- Real-time amplitude (volume) analysis using the Web Audio API framework.
- Real-time frequency (pitch) analysis using the Web Audio API framework
- Conversion of pitch into 6-digit hex color
- Beat detection using web-audio-beat-detector

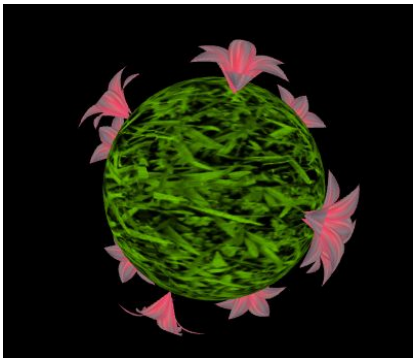
Part 3: Evaluation

How well did you do? What parameters did you tune along the way? Include some WIP shots that compare intermediate results to your final. Explain why you made the decisions you did.

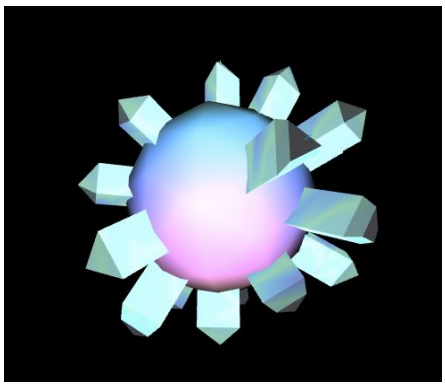
***Emily*:**

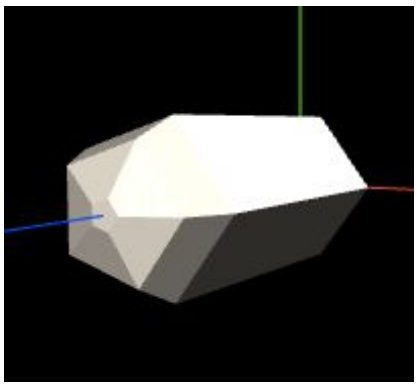


For week 1, I think I made good progress because I was able to generate a simple world using my own framework. I spent a lot of time assuring that my base code was easy to extend so that a planet can have any type of animation, assets, etc. I didn't do too much tweaking for the creation of the flowers.



I wanted to make peruvian lilies, so when I was tweaking the toolbox functions, I essentially knew what kind of shape I wanted to go for instead of testing to see what looks good. I use an ease function to get the curvature of the flower petal, as well as the actual shape of the flower, but kept testing normal quadratic equations before realizing that the shape of the ease function was better. I was quickly able to get the flower planet working after adding texturing as well.



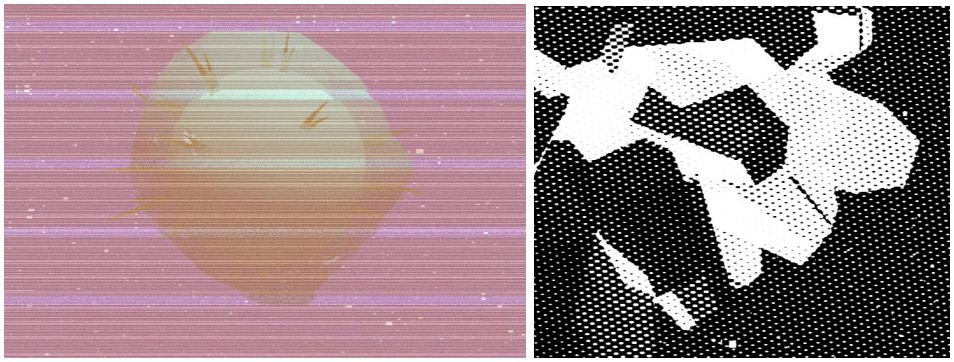


For week 2, I focused on the creation of another planet. I wanted more natural looking structures, so I varied the number of points in the 2D shape the crystal was extruded from, as well as the size and the number of bevels. I added an iridescent shader, but didn't think to add transparency to the material as well as noise until I looked more at opal materials. I did those tweaks in week 3. I added very very basic animation along the normal (essentially displacing with the sin of time).

For week 3, the choreography was implemented by me. I had to tweak the camera movements that were from week 1 because they had a fixed time step size. I found it way more effective to specify a time interval of movement, as well as a certain distance, rather than hoping the planet will get to where you need to in some duration of time. I used a simple lerp, but basically normalized the value of time to be within the specified interval. This made the choreography, movement, and timing of the camera transitions to be a lot more precise. From there, I just assured there was a decent duration to appreciate the look of the planets and their animations, and transitions with simple camera zooms. For the crystal, I made them explode before transitioning to make it more interesting.
I also helped with the music data extraction. I updated the classes so they can access the music analyzer, and from there, you can read from the music buffer for every tick. Each asset was mapping to some point in the buffer, and either updated the scale or offset along the normal based on the frequency read back. I had to do some tweaking to the way I implemented this so that these animations could be consistent and occur, but while a planet is rotating as well. I did the animation for each planet.

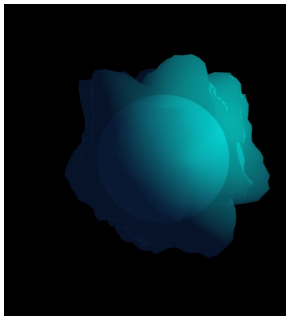
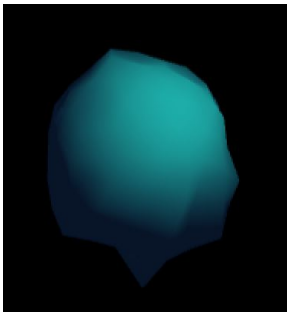
***Suzanne*:**

Retrieving color from pitch information went through a lot of iterations - since there is only one frequency, I could only easily get a grayscale color, but that isn't what I wanted. I tried a few ways to convert it to RGB by alternating which component I was changing (R, G, or B), or to just use it as an offset to an existing color. I eventually settled on converting the frequency to a 6-digit hex from which I could create a nice color. Then, the problem became that the color changed too quickly and too rapidly, so I fine-tuned this by taking in the previous color and blending this with the new color, which gave me more subtle color changes, and more nice, muted colors overall.

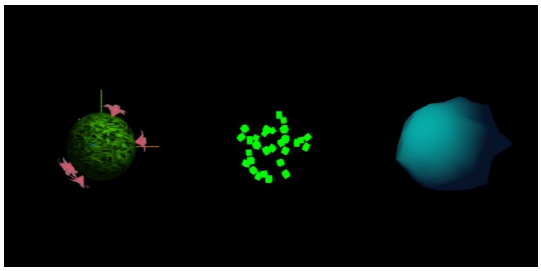
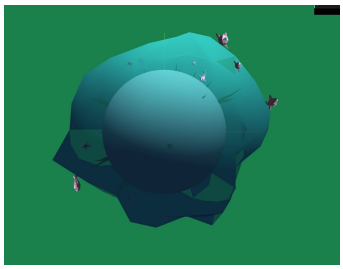


The "retro" shader had a lot components and fine-tuning that went into it as well - I decided to do scanlines, grain, small distortions, and also a few brightness, light, and warmth adjustments to make it seem more "faded." It was difficult deciding which components should be affected by audio information and which shouldn't, and I settling on just changing the extra-bright glitch strips that switch around when the amplitude rose above a certain threshold. Initially I had also tried to change the strength of the grain according to the audio, but this appeared too chaotic and flashy for the viewer. I had a similar decision to make for the dots shader - I originally tried to make the dots grow with the amplitude but this was too distracting and flashing, so I left it out.

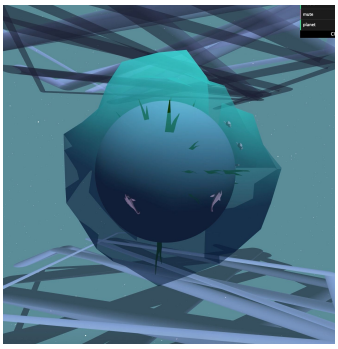
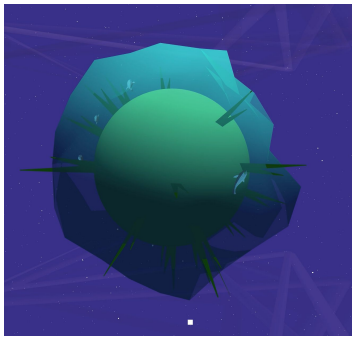
***Andrea*:**

One of the first concerns I had when making the water planet was how I wanted to visualize the water, and how to make a mesh believably watery. First I tried applying a single noise function, but it didn't give me enough satisfactory wavy-ness. I ended up implementing fractal brownian motion noise, but when I applied that to the surface of the mesh, there was a lot of unnecessary detail which didn't suit the liquid-like visualization I was aiming for. Ultimately, I decided to reduce the polycount of the mesh sphere. Rather than striving for super realistic water, I think that this decision made the water more stylized, and also more believable. I also altered the alpha channel of the material so there is a transparent quality to the water, and the base sphere underneath the water changes with respect to a global timer.

I also worked through deciding how to implement my assets into my planet, and which to include. I first thought about implementing a rock system, or improving the terrain of the base planet, but I ended up deciding to create assets that can move around. Thus, I ended up creating seaweed plants which move with the current (of the music), as well as koi fish which have some limited swimming abilities.

I worked on the setting up the basic camera controls and choreography, as well as the additional geometry in the rest of the environment. I ended up kind of happily stumbling upon this cool visualization of twirly-type things above and below the planet. This was simply achieved in the vertex shader, where changed the position of the mesh based on the uniform time variable divided by the y-position. Although interesting, the visualization was a bit distracting, so I changed the transparency to help it blend in more with the background rather than competing for attention in the foreground. I also worked on fine-tuning the color of the environment. Initially, there was too much greenish in the background, which made everything look like it was underwater. By altering the way that the audio was interpreted into an RGB value, I was able to achieve a more space-appropriate color range.

Part 4: Future Work

Given more time, what would you add/improve?

Emily's improvements:

- More interesting animations rather than just the music driven ones (e.g. the crystal explosion). Can make those animations specific to the planet type.
- More subtle details in the planets. The flower planet could have more flowers, bump mapping and the crystal planet could look as if it is growing out as clusters and have a more varied orientation.
- The music driven animations could be more idiosyncratic to the type of the planet. For example, the flowers could bounce and bloom rather than scale in size.
- The animations would be done primarily on the GPU because the CPU craps out so hard with the camera transitions, planet rotations, and asset orientation upsets.
- More planets that incorporate more complex procedural concepts, such as lsystems.

Suzanne's improvements:

- Smoother transition between colors generated from pitch changes
- Use generated color for more aspects of the project
- Figuring out how to easily stack multiple shaders without having multiple composers (currently the composer bugs out when I try to do this)
- Pre-analyze the "character" of a music in order to procedurally (instead of hard-coding-ly) assign shaders and worlds to a song.
- Use bpm information for more than just rotation speed

Andrea's improvements:

- Implement procedural koi swimming behavior, instead of just offsetting the position in the frag shader with a sine function.
- Work on the grammar rules for the seaweed - incorporate more variety in the creation of the plants, such as different number of sprouts, varying shapes, and different colors
- Incorporate sky boxes to make for a more immersive visual experience
- Continue working on background - add a field of falling stars, additional planets
- Offset the water in the water planet based on audio data, so that the waves correspond to the music playing at the time

Part 5: Acknowledgements

IQ's color palette function: http://iquilezles.org/www/articles/palettes/palettes.htm

Web Audio API: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API

web-audio-beat-detector - https://www.npmjs.com/package/web-audio-beat-detector

Autocorrelation (pitch detection) algorithm:
https://developer.microsoft.com/en-us/microsoft-edge/testdrive/demos/webaudiotuner/

Dots shader influence: https://threejs.org/examples/webgl_postprocessing.html

Post-processing grain:
https://www.reddit.com/r/opengl/comments/1rr4fy/any_good_ways_of_generating_film_grain_noise/

Post-processing scanlines:
https://gamedev.stackexchange.com/questions/64036/how-can-i-implement-a-scanline-effect

RGB shift influence: https://forum.unity3d.com/threads/rgb-split-shader.220068/

Vertical roll influence:
https://github.com/felixturner/bad-tv-shader/blob/master/BadTVShader.js

Music: Humble by Kendrick Lamar

Music: Wildcat by Ratatat