**Project title:** All-Terrain ATV

**Group or individual member(s)**

1. Jacob Alldredge

2. Kyle West

**Project URL (Where your project can be run. (ie., Github url). You must upload all of your project code to your GitHub account.**

<https://github.com/kyle-west/CS312-ATV> // Repository link

<https://kyle-west.github.io/CS312-ATV/> // in browser version

**1) What did you accomplish for this project based on your project proposal? What did you want to accomplish? Give details. Why did you select this project?**

**Terrain**

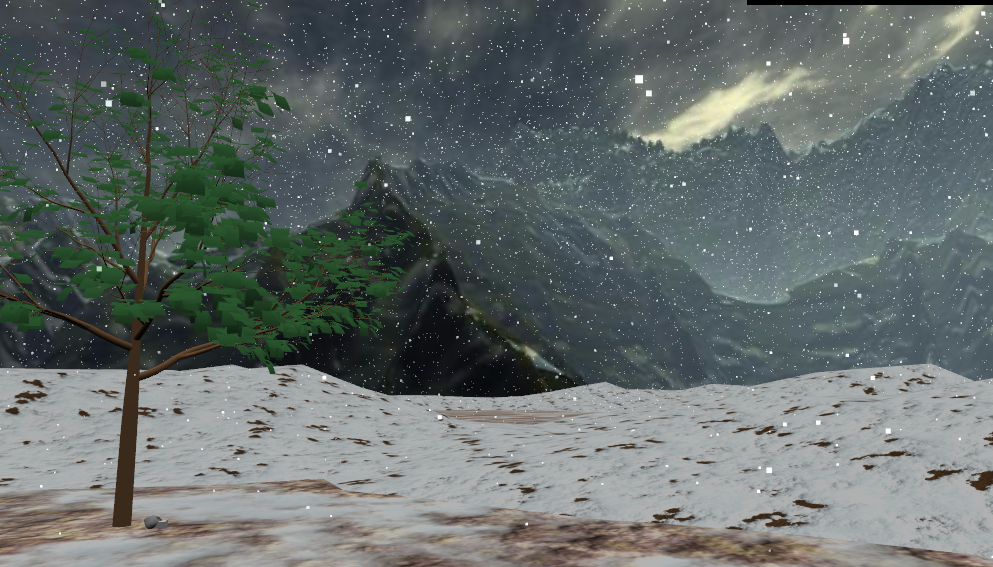
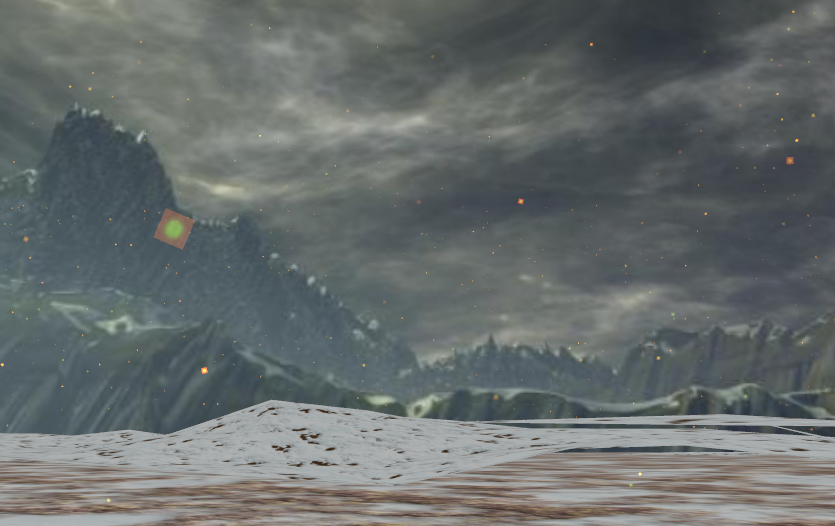
For our project we ended up with a pseudo random terrain. What we did was create a system of heightmaps that use grayscale to determine what the height of each of the vertices should be. The whiter the particle on the heightmap the greater the y value of the vertices.

We then created an array of references to the different heightmaps which are selected from randomly to create a grid of 64 heightmaps so that on each load the world is assembled at random (note that one of the maps we used spells out our name).

**Particles**

We two particle engines so that we could simulate two types of weather conditions. The first engine when called to run simulates snow, and the second simulates an environment with fireflies floating around. For each particle system you are able to set the colors for each particle as well as the color before it dies, this is how we created our firefly effect.

Particle systems also have a gravity vector so that you can create the falling animation like with the snow, and we also used it to implement wind effects that are controllable through the GUI.

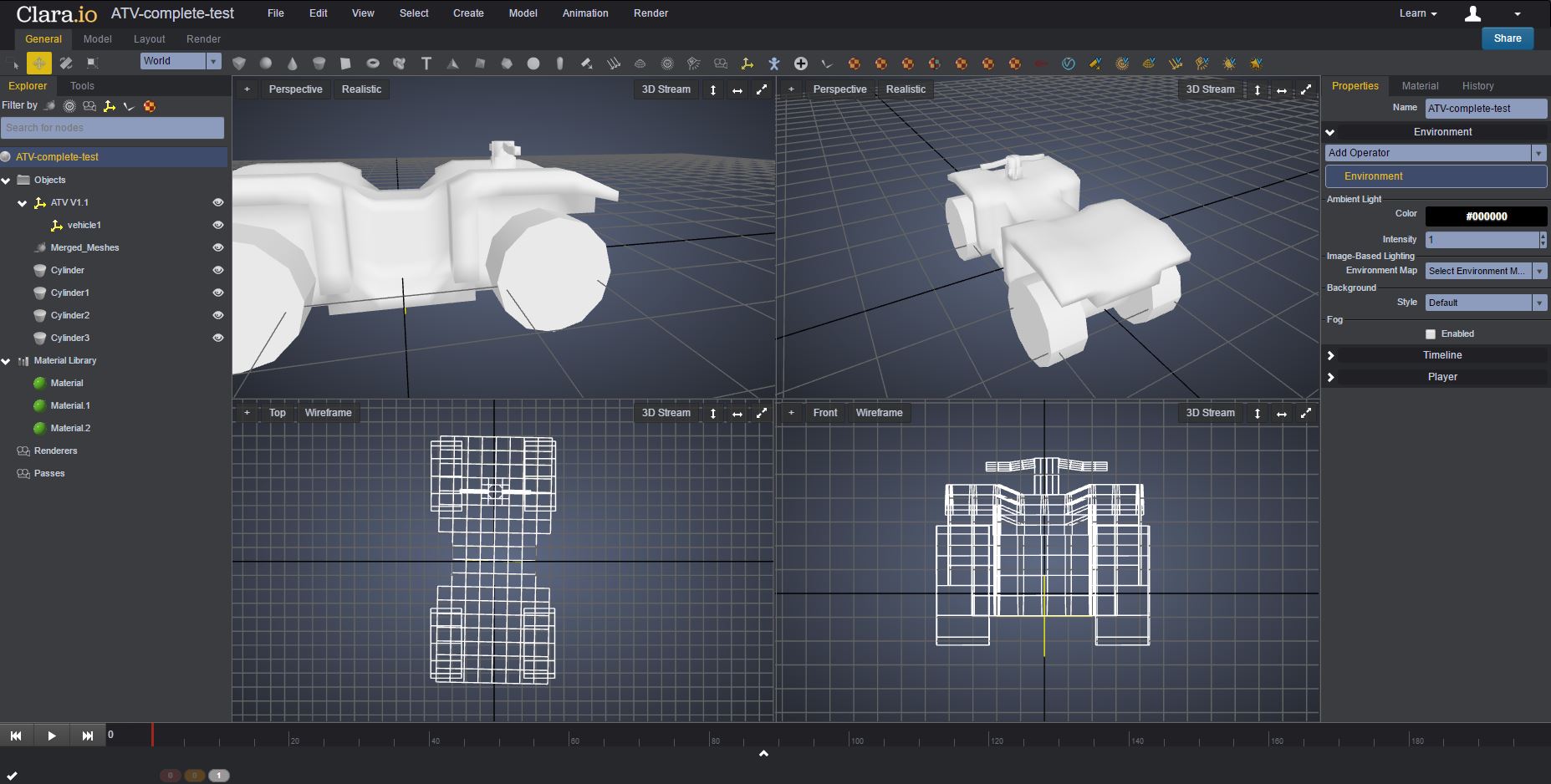
**GUI**

We used the dat.gui package to implement a GUI in our project. It can be found in the upper right hand corner of the screen, and is used to dynamically adjust variables such as which particle system is running and the direction of the wind.

**ATV**

The ATV that we use in our project was created by us in Blender and Clara.io. We ran into issues applying physics imposters to our wheel meshes because they were simply too complex so we ended up replacing them with simple cylinder meshes. We ran into issues with our physics engines which is explained further below, so we ended up applying physics to a sphere and having our ATV mesh mimic the speed and angle of the sphere.

The mesh was given what we are calling a “visual imposter.” Because the imported mesh uses a rotation vector as a rate rather than an angle, we had to make a box behave as as the atv should, then we make the ATV mesh a child of the box. We used animations triggered by the user’s input to make it feel more real to use.

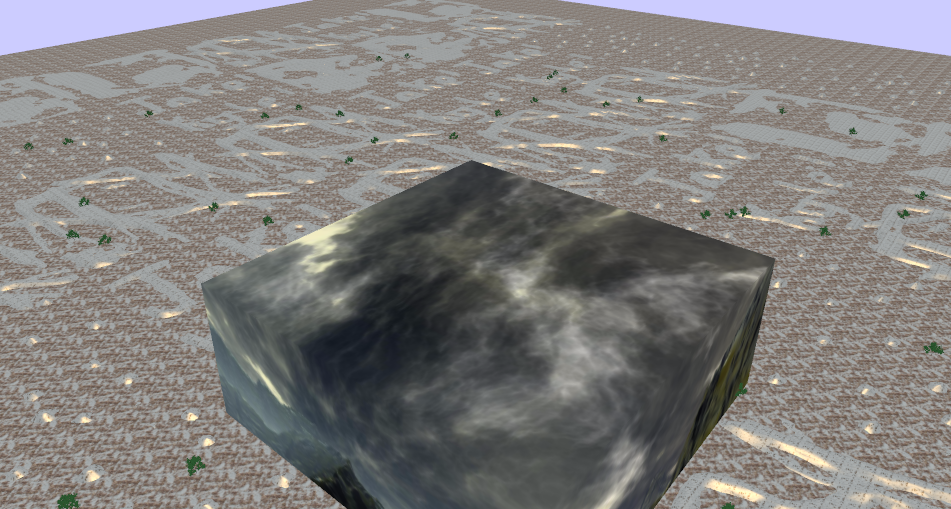
**Additional Meshes**

To help fill in our environment, we borrowed some prebuilt meshes from Clara.io in the form of a tree and rock. These meshes are then randomly placed at an X and Z coordinate throughout our world. The rocks are then scaled randomly to create the effect of multiple rock meshes to provide variety to the world environment.

**Skybox**

Kyle created a skybox from pictures that he borrowed from the internet. We are using a mountainous area scenery which we have grown to like. It follows the player so that it appears to be infinitely in the distance.



**Physics**

For our physics engine, we used Cannonjs. Our initial decision to use this engine was based on the fact that the core goal of our project was to have a randomized terrain that objects would react realistically with and this was achieved with Cannonjs. We have explained below some of the issues that arose because of this choice along with our workaround to the problem. It is possible that if we had originally started with Oimojs as our engine, we could have had an actual car controller where the friction of the wheels carried the player, but it simply wasn’t possible because of the exclusiveness of both engines.

**Music/Sounds**

Kyle wrote our theme music. It was done in GarageBand and loops infinitely. The music can be disabled through the GUI or just mute your computer. The ATV sounds were borrowed from another game. The code increases and decreases the playback speed of the engine according to the speed of the player. It sounds more realistic this way.

**2) What did you NOT accomplish that you indicated in your project proposal? Why?**

The only aspect of our project that we were not able to master was the realistic vehicle controller. This issue arose when we discovered that the physics engines for babylonjs were mutually exclusive in the technology that we needed. This is to say that Cannonjs had the vehicle engine and movement that we needed but not the vehicle joints. Oimojs had all of the vehicle joints that we needed to assemble the ATV, but when the assembled vehicle collided with our heightmaps, it would crash our program. We worked around this by applying physics to a sphere which interacted appropriately with our heightmaps and then making our ATV mesh effectively child of the sphere. This way we simulated realistic physics, but it is not a true vehicle.

**3) Did you need to change your project? Why?**

There was no need to make any significant change to our project, except our change from threejs to Babylonjs. The reason for this change has to deal with the physical interaction between our vehicle and the randomized heightmaps. In threejs we were able to import a vehicle, and it behaved realistically with physics and controls. However, Threejs does not handle plane physics; our vehicle would drive up on invisible slopes and drive through the generated terrain, so we needed a change.

Babylonjs allowed us to create heightmaps and almost perfectly handles collisions with objects against it. This was the only significant change we made to our project.

**4) Provide any open source libraries/code used in your project.**

· Babylon.slim.js

· Dat.gui.min.js

**5) Provide any references or websites used in your project. I want URLs. If you copied code from any site, you must indicate that site.**

<http://cdn.babylonjs.com/wwwbabylonjs/Scenes/minority-race/>

We used this code to create engine sounds for our ATV.

<https://clara.io/view/ac16a020-e8e1-4bef-a7be-721f0dbd36d9>

This is where our tree mesh can be found.

<https://clara.io/view/0f8f50d5-d0f7-438c-b634-c92aa595baa3>

This is where we got our rock mesh from.

<http://3delyvisions.co/skf1.htm>

This is where we got the skybox images from.

**6) Self-grade your overall project. Things to think about: Did you finish the project? Is it professional? Can others use it and understand what it does?**

A

**7) Other comments, Issues that you had while completing your project, images or notes**

We also made the loading screen by injecting HTML into the loading screen text and using JavaScript and CSS to make it look nice and update the user on the loadings progress.

