Seasoned Cube: An Interactive Environment

Aisha Peters
Student
Computer Science
University of Alaska Fairbanks
Fairbanks, Alaska, USA
ampeters@alaska.edu

ABSTRACT

This paper covers the process of rendering an interactive scene with seasons and weather using WebGL. The standard methods of particle engines and particle appearance are used for precipitation effects with a slight focus on realism for rendering. Users are able to control the scene through a GUI to alter the season and trigger precipitation…

KEYWORDS

Interactive, Particle Engine, Three.js, GUI

ACM Reference Format:
Aisha Peters. 2018. Seasoned Cube: An Interactive Environment. In *Proceedings of ACM SIGPLAN Conference on Programming Languages (PL'18).* ACM, New York, NY, USA, 5 pages.

1 INTRODUCTION

The goal of this project was to make an interactive scene that changes season and weather using the WebGL framework three.js. The scene created consists of a few trees, both deciduous and coniferous, upon a cube that acts as the ground. There are options to change the season to spring, summer, autumn, or winter through a GUI. Each season has an additional option to trigger precipitation, which is generated using a particle system. Precipitation type and magnitude is dependent upon the current season. For example, summer has lighter rainfall than autumn and only winter has snowfall. There is also a fog effect that is triggered during autumn and winter precipitation.

PL'18, January 01–03, 2018, New York, NY, USA
2018.

2 PREVIOUS RESEARCH

2.1 Seasonal Precipitation

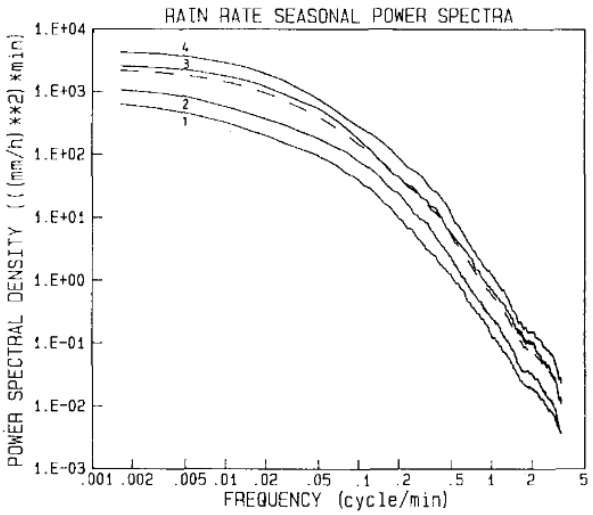


Figure 1: 49-year power spectral density and its seasonal variability [3] [1) Jan-Feb-Mar, 2) Apr-May-Dec, 3) June-July-Nov, 4) Aug-Sept-Oct]

The rate of rainfall changes throughout the year, typically with the highest rate per season occurring in the months of August thru October, and the least being in January thru May [3]. This means that starting in spring, rainfall gradually becomes heavier throughout the year till being at its heaviest in autumn, before it is replaced by snowfall in

the coldest months. These rates are to be considered when rendering fixed seasonal effects in an application.

2.2 Particle System

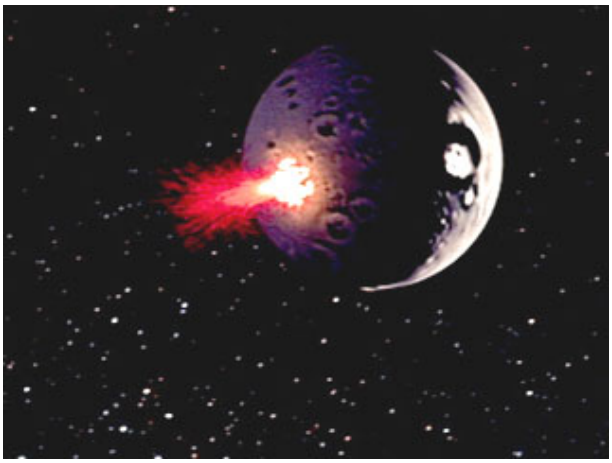


Figure 2: Particle effect by William Reeves in film Star Trek II: The Wrath of Khan [Paramount Pictures]

In computer graphics, precipitation is generally rendered using a particle system. Particle systems were coined in 1983 by William Reeves [6], and are used to model dynamic fluid or amorphous objects. The main component of a particle system is the particle engine. This is used to generate particles, which can be any type of graphical object such as sprites or 3D models. Each generated particle has attributes that may change like position, velocity, or acceleration. Altering the attributes can create many effects, such as falling, path following, exploding, or randomly traveling particles. The combined effect of the particle movement is used to simulate a larger entity, like smoke, dust clouds, or rainfall.

3 PROBLEM ANALYSIS

3.1 Changing Seasons

There are four seasons which affect the color of a nature scene. Each season has an affiliated hue based on what has been observed, such as vivid shades of green in the summertime and orange and yellow in autumn. The exact colors are dependent upon seasonal perception [7], which is not something a graphical program has any control over. Typically, seasonal changes are implemented by changing the assets used to a version that has the seasonal effects already applied such as changing leaf color, adding a layer of snowfall, and removing leaves.

3.2 Rendering Rain and Snow

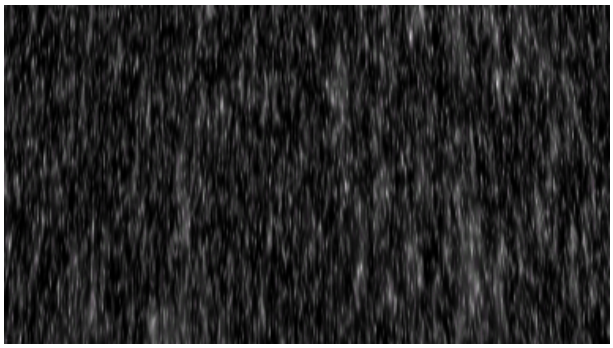


Figure 3: Example of rain rendered with particle engine [9]

In order to render reasonable raindrops, distortion from their motion has to be considered when being viewed by the camera [4]. This causes raindrops to be elongated instead of being a simple drop of water, and is usually done by rendering streaks from a particle engine to represent the raindrops instead of a waterdrop shape [9]. There is also the attenuation of visibility from heavy rainfall to consider. This is the result from evaporating vapors which requires some fog to simulate in graphics [2].

Snowfall, unlike rain, does not have the same shape for each flake. This makes rendering a bunch of uniform particles inaccurate for snow representation. A method to make the effect more realistic is to use a cluster of particle systems with each generating a different type of snowflake [8]. Snow

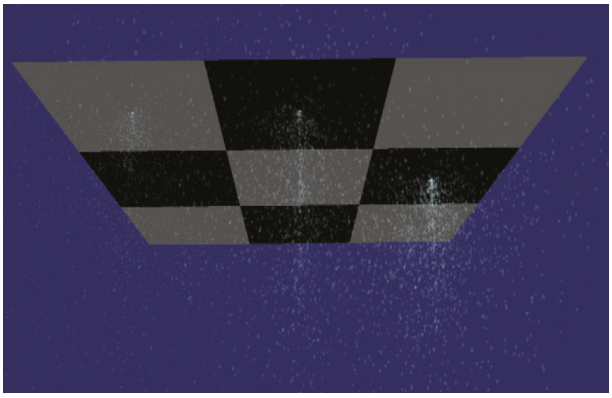


Figure 4: Snowflake clusters [8]

also accumulates over time on surfaces, which can be simulated by using particle collision to detect when snow has landed on a surface [5].

4 PROPOSED SOLUTION

4.1 Three.js

Three.js was created by Ricardo Cabello in 2010 [1]. It abstracts API calls which makes WebGL development simpler, thus speeding up the process required to make a working program. Features include camera controllers, premade geometry, ready to use lighting, and easy scene management. Since this project was to be completed within a limited amount of time, Three.js was ideal for allowing time to focus on the main goals.

4.2 GUI for Managing Changes

Since this project was meant to be interactive, all changes to the scene are managed within a GUI. For this purpose, dat.GUI was used because it is lightweight and compatible with Three.js. Using the GUI, the program allows the user to pick the current season from a dropdown menu. Their selection changes which preset color is stored in the variables used for each object's material color, which will then be re-rendered into the scene. For precipitation, a checkbox is used to trigger it on or off. The type of precipitation is determined by the current season setting.

4.3 Particle Settings

A particle engine is used to generate rain and snow in the scene. For rain, an elongated thin sphere represents each particle. For snow, each particle is a randomly scaled sprite. The season setting effects whether having precipitation active triggers rain, heavy rain, or snow. Heavy rain and snow act as a trigger to generate fog into the scene.

5 RESULTS AND ANALYSIS

5.1 Interaction

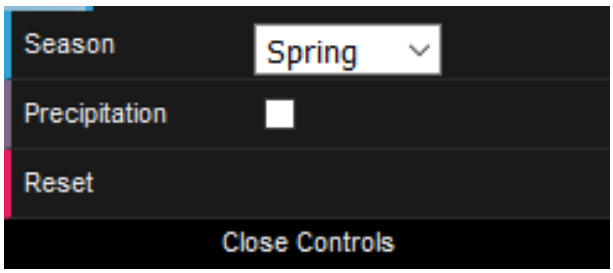


Figure 5: GUI used in project

Using dat.GUI, a simple interface is available to the user for making any changes to the way the scene is rendered. Implementation using this method was time efficient, as a custom GUI was not necessary for the project. Values are stored in the GUI objects which upon input update variables used by the code setting up the scene objects. These changes are then automatically rendered during the next call to the program's update function. There is also camera control available through interaction with the mouse, which Three.js provides functionality for. Limits were set to the camera zoom to prevent the user from going outside of the rendering space.

5.2 Changing the Scene

Since custom assets are not used for this project, changes are made directly to the material of the

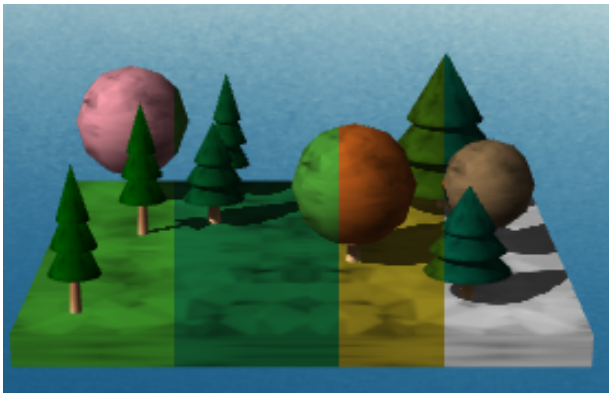


Figure 6: Cross section of each season

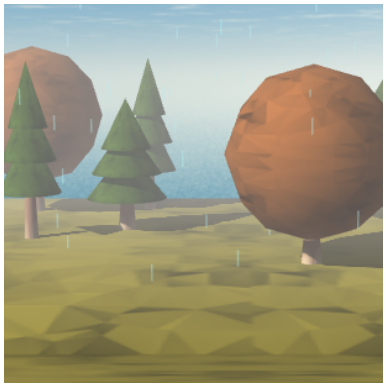
trees and ground. This would cause issues with a larger scale project with all of the work being done every update call, but for a small project like this one this approach works well. No new code is implemented, as the material is given a variable that stores the color to use, which is altered through the GUI.

5.3 Rain and Snow

Due to time constraints, a separate particle engine was set up for the rain and snow rather than having a single engine with multiple particle types. However, this is not something that effects the user as the resulting effects are the same regardless. It does add to the time spent on computations, but as this is a small project the impact is not problematic. Snow is represented by a sprite to give it the impression of a snowflake, whereas raindrops are drawn lines. The rate at which rain moves was increased to give a more realistic appearance.

6 FUTURE WORK AND CONCLUSION

Continued work on this project would be to create seasonal assets to load into the scene, add in more objects such as grass tufts and flowers, implement wind effects, have a dynamic day and night cycle, and use collision to trigger splash effects for raindrops. The only issue encountered while working



(a) Rain

(b) Snow

Figure 7: Seasonal effects in scene

on this project was implementing a dynamic particle engine to generate both rain and snow. This too would be implemented during future work.

For this interactive project, a GUI is ideal for user input as it provides an easily understood interface for them for interact with. Since this project draws objects rather than preloading a set of seasonal assets, seasonal change is best implemented through user input to change how the scene is rendered via color. Precipitation is done using a particle engine as is normally practiced, which can be triggered through user input to give them more control of the scene. For slight realism, raindrops are streak-like shapes to factor in their movement when viewed from a stationary camera and snowflakes are random sizes to give variance.

REFERENCES

[1] Brian Danchilla. 2012. Three. js Framework. In *Beginning WebGL for HTML5*. Springer, 173–203.

[2] NR Donaldson and RE Stewart. 1993. Fog induced by mixed-phase precipitation. *Atmospheric research* 29, 1-2 (1993), 9–25.

[3] E Dutton and H Dougherty. 1979. Year-to-year variability of rainfall for microwave applications in the USA. *IEEE Transactions on Communications* 27, 5 (1979), 829–832.

[4] Kshitiz Garg and Shree K. Nayar. 2006. Photorealistic Rendering of Rain Streaks. *ACM Trans. Graph.* 25, 3 (July 2006), 996–1002. https://doi.org/10.1145/1141911.1141985

[5] Benjamin Neukom, Stefan Müller Arisona, and Simon Schubiger. 2018. Real-time GIS-based snow cover approximation and rendering for large terrains. *Computers & Graphics* 71 (2018), 14–22.

[6] William T Reeves. 1983. Particle systems - a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics (TOG)* 2, 2 (1983), 91–108.

[7] Tsuyoshi Shimmura, Tomoya Nakayama, Ai Shinomiya, and Takashi Yoshimura. 2017. Seasonal changes in color perception. *General and comparative endocrinology* (2017).

[8] Jian Tan and Xiangtao Fan. 2011. Particle system based snow simulating in real time. *Procedia Environmental Sciences* 10 (2011), 1244–1249.

[9] Yoann Weber, Vincent Jolivet, Guillaume Gilet, and Djamchid Ghazanfarpour. 2015. A multiscale model for rain rendering in real-time. *Computers & Graphics* 50 (2015), 61–70.