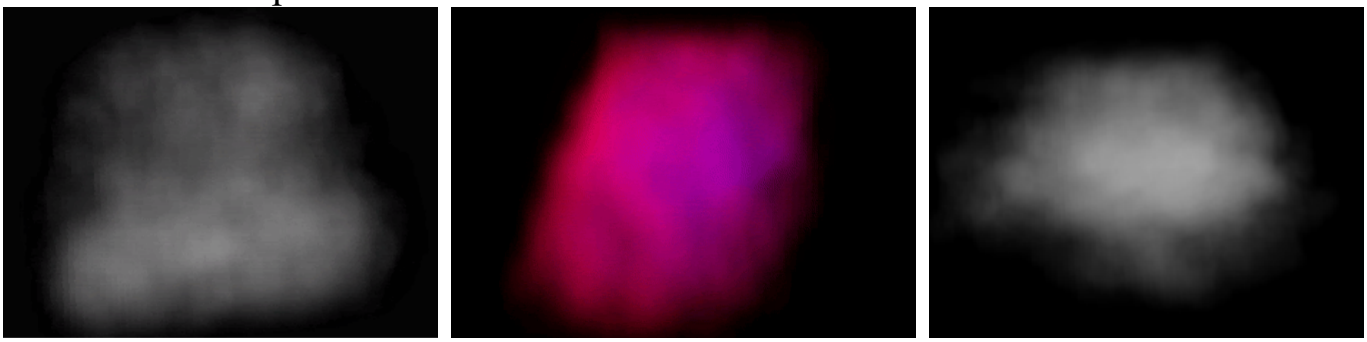
Cloud Sim Final Report

Project Members: Sayan Das, Andrew Wang, Mohak Jain, Anthony Zhou

Abstract

We created a real-time voxel-based simulation of clouds. Cloud simulation is an exciting and relevant topic due to its use in rendering realistic outdoor scenes, such as in games and movies. We incorporated a pseudo-physical model to dynamically render clouds in a voxel space, in which we incorporated cloud movement via advection, elliptical cloud spaces, smoothing via cloud density calculation, and rendering via THREE.JS's Lambertian model and ray marching to arrive at our final product. The final simulation goes through the process of a cloud forming and evolving over time and wind. We are proud of our very realistic-looking result, and we learned a great deal about the THREE.JS/WebGL tech stack we used along with the reinforcement of many of the concepts we were exposed to in this course.



Technical Approach

Our process was generally divided into simulation, rendering, and beautification. We started by

implementing a base physical model in our voxel space based on a 2000 paper from Dr. Tomoyuki Nishita's group at the University of Tokyo. We then moved into the process of rendering these voxels in a presentable and realistic way. Finally, we played with simulation parameters to create as realistic of a demonstration as possible.

Simulation

The 2000 Nishita paper suggests that an effective voxel-based pseudo-physical cloud model can be constructed by assigning each voxel in the space a set of three boolean values. These values are activation of phase transition from vapor to cloud (act), humidity (hum), and cloud (cld). Clouds are only rendered if the cld variable is set to true. At each time step, we calculate the value of these three variables using the following transition rules:

$$hum(i,j,k,t_{i+1}) = hum(i,j,k,t_i) \wedge \neg act(i,j,k,t_i), \quad (1)$$

$$cld(i,j,k,t_{i+1}) = cld(i,j,k,t_i) \vee act(i,j,k,t_i), \quad (2)$$

$$act(i,j,k,t_{i+1}) = \neg act(i,j,k,t_i) \wedge hum(i,j,k,t_i) \wedge f_{act}(i,j,k), \quad (3)$$

We begin by assigning each voxel's hum and act parameters at random and with cld as off everywhere in the space. We get the following basic result, where this rectangular prism is the entire voxel space:

With this base simulation done, we moved on to adding movement to our model.

Advection

Advection is movement by wind, and governed rather straightforwardly by a simple set of formulas at each time step:

$$hum(i,j,k,t_{i+1}) = \begin{cases} hum(i - v(z_k), j, k, t_i), & i - v(z_k) > 0 \\ 0, & \text{otherwise} \end{cases}, \quad (8)$$

$$cld(i,j,k,t_{i+1}) = \begin{cases} cld(i - v(z_k), j, k, t_i), & i - v(z_k) > 0 \\ 0, & \text{otherwise} \end{cases}, \quad (9)$$

$$act(i,j,k,t_{i+1}) = \begin{cases} act(i - v(z_k), j, k, t_i), & i - v(z_k) > 0 \\ 0, & \text{otherwise} \end{cases}, \quad (10)$$

The velocity function is specified by the animator, and one can implement a piecewise-linear wind function in order to move different parts of the cloud at different speeds. Our implementation is a unidirectional movement along the x-axis based on which z-coordinate each voxel is at.

An example of this is shown below. Notice how the cloud moves at different speeds and directions for different values of z:

Realism: Probabilistic extinction and ellipses

As we see above, the simulation currently results in all of our “clouds” reaching a steady state with no irregularities, forming and moving rather unrealistically. However, we want our model to be dynamic and continually evolving, which meant introducing a handful of random events.

First, we introduce a set of three probabilities that allow for our simulation to have random behavior: An extinction value, used to probabilistically turn the cloud boolean off, and an activation and humidity value, which probabilistically turn the activation and humidity of a voxel on. THroughout the project, we continuously tweaked these for more realistic clouds.

We then followed a suggestion from the Nishita paper to use ellipsoids to seed our clouds. We generated random, axis-aligned ellipsoids, ensuring that they

were both within our voxel space and not too large. We then used these ellipsoids to weigh the probabilities mentioned above, at a high level ensuring that the cloud boolean was more likely to turn on near the center of the ellipsoid. We calculated these probabilities additively, summing probabilities from all ellipsoids, and ensuring that voxels that were not in any ellipsoids were impossible to form clouds.

We determined that the below metric performed well and produced realistic clouds, making the edges of clouds sparse and the centers dense.

Weighting for Seeding Probability

$$dist = 1 - (\frac{i - e.xpos}{e.a})^2 - (\frac{j - e.ypos}{e.b})^2 - (\frac{k - e.zpos}{e.c})^2$$

```
If dist >= 0, wt = SQRT(1-dist), else wt = 0
```

Rendering: Density, Ray Marching, & Linear Color Interpolation