SIMULATION

Cellular Automata

Going back to the very initial work on my cloud demo, I realized that the way it was implemented is logically flawed which resulted in poor performance. This revelation comes after understanding how SilverLining SDK implemented the Cellular Automata in their cloud system. What I've done wrong is that I've implemented the entire cloud space as a simulation grid. So I'm basically looping through the entire simulation grid in each render frame, checking whether each grid point has a cloud or not.

How SilverLining does this is by encapsulating the simulation grid for each cloud instead of the entire sky (cloud layer). Their cloud is generated using an exponential cloud generator that will output different cloud sizes based on a Plank equation. [1] So technically, you will get different sizes of clouds ranging in an exponential fashion. After that, SilverLining uses a cloud distributor to randomly place the different clouds sizes in the cloud layer specified by the users in the initial setup. As such, their method does not involve any physical computation of fluid dynamics. However, they do allow their clouds to 'grow' a few time steps using the cellular automata method in order to produce puffy clouds instead of a spherical solid block of clouds.

Figure 1 : Simulation grid covers the entire sky region (mine)

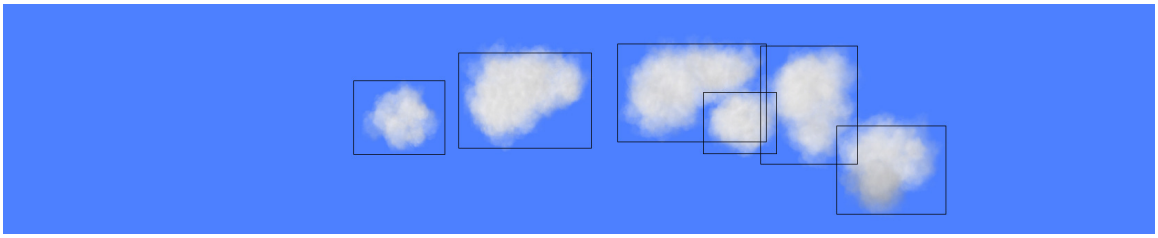


Figure 2: Conceptual portrayal of simulation grid for each cloud (SilverLining)

Also, SilverLining allows user to choose whether to simulate cloud growth. However, this feature doesn't seem to work when I turn it on. There are ways to make it work; either executes the cloud growth in the GPU after a few render frames as suggested by Mark Harris [2] instead of iterating through the many many arrays of clouds or do something like how 2XL enable physics for the eco-system based on the vehicle. We can allow the cloud growth to execute for a few selected different clouds or those nearest to the helicopter? This is just one of the possible ways that I can improve the cloud growth.

Sho Chin Hwa
gary@tqstudios.com.sg

Coupled map lattice

A simplification to the Navier-Stokes Equation is to use a discrete representation of it as suggested by [3]. This method, I believe is one area where I can research and make improvement on. According to the author, this method is easy to implement and the computational cost is small. It would even be better if I could run this simulation in GPU, which is my ultimate direction.

Here, I will attempt to explain the conceptual idea behind this method, or at least what I understand. I will break down the steps into three parts.

Part 1: Implement the method in C++ code.
Part 2: Visually represent the simulation using DirectX.
Part 3: Implement the method in HLSL.

According to the paper, cloud formation is simulated by applying *6 processes* for each time steps.

1. Viscosity and Pressure Effect
2. Advection
3. Diffusion of Water Vapor
4. Thermal Diffusion
5. Buoyancy
6. Phase Transition

The variables that are being affected by these processes at each time steps are

1. Velocity vector, $v(x, y, z)$
2. Water vapor, $w_v(x, y, z)$
3. Water droplets, $w_l(x, y, z)$
4. Temperature, $E(x, y, z)$

Viscosity and Pressure Effect

The variable affected by this process is the velocity vector, $v(x, y, z)$. Viscosity will cause velocity diffusion, meaning a more viscous fluid will dampen the velocity vector. The rate at which it effects the velocity diffusion is parameterized by a viscosity ratio k_v.

The pressure effect is a little absorbed. It is stated that the pressure effect requires the concept of the conversation of mass. Which means the fluid is incompressible and the amount of mass in one lattice is conversed throughout the entire lattice. So, based on this understanding, let's say one lattice has 1 unit of mass and the rest of the lattice points have none. This 1 unit of mass will be transported to different lattice points at different time steps by the velocity vector. But at any time steps, the total amount of mass in that fluid is 1 unit.

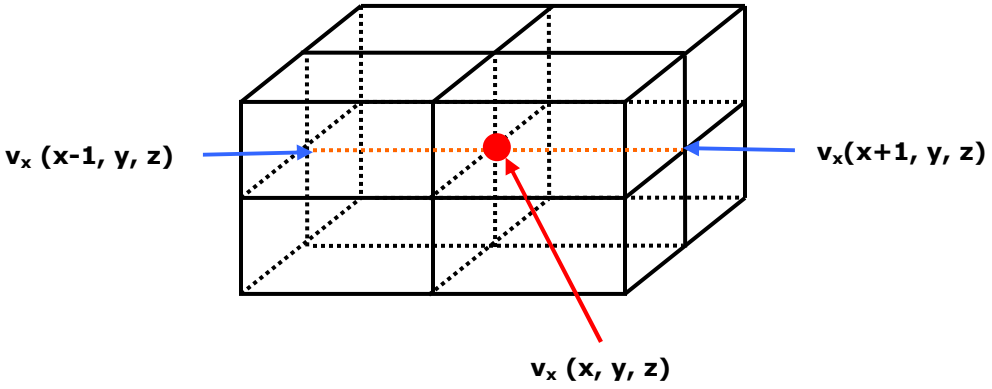
The paper mentioned that if we incorporate the pressure variables in the cloud modeling, it will results in a complicated modeling. As such, they introduce the

concept of discrete version of $grad(divv)$. $divv$ is the mass flow of each lattice and $grad(divv)$ is the flow caused by the gradient of the mass flow around the lattices.

Sounds complicated? Perhaps a diagram at this point will help illustrate this concept and hopefully, I got it right. Assume a portion of the lattice looks something like the diagram below. The term for $grad(divv)_x$ is given by the following equation.

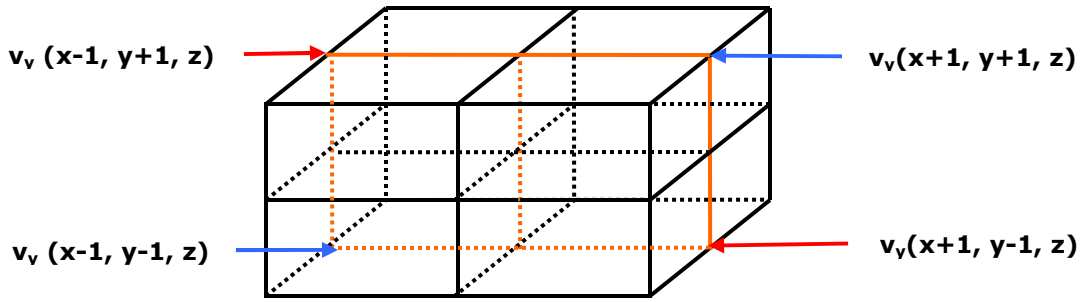
$$grad(divv)_x = A + B \text{ where}$$

$$A = \frac{1}{2}\left[v_x(x+1, y, z) + v_x(x-1, y, z) - 2v_x(x, y, z)\right]$$



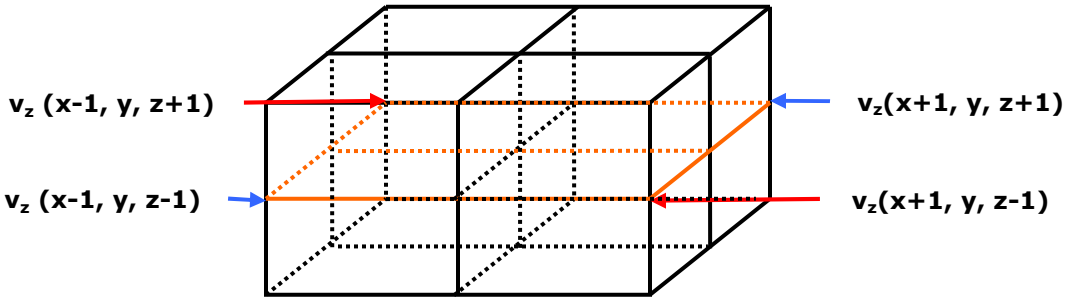
$$B = \frac{1}{4}(C + D) \text{ where}$$

$$C = v_y(x+1, y+1, z) + v_y(x-1, y-1, z) - v_y(x-1, y+1, z) - v_y(x+1, y-1, z)$$



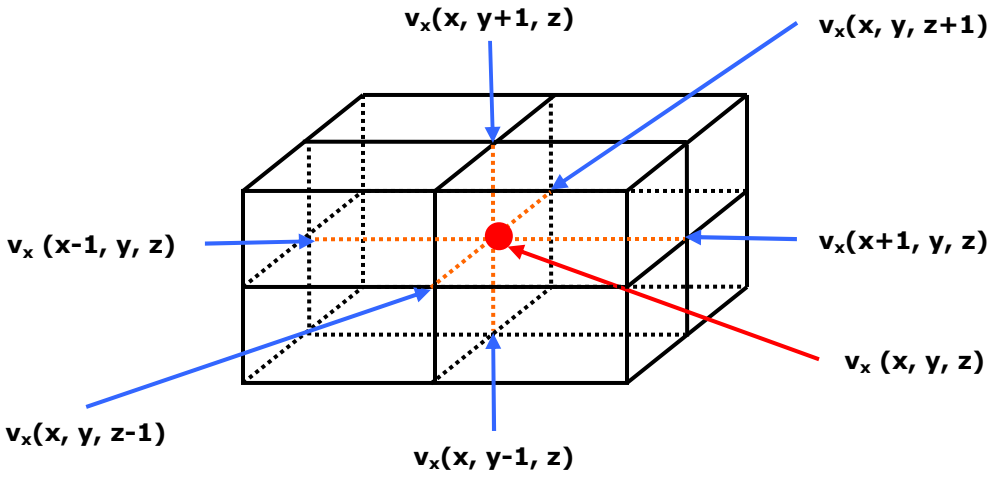
Sho Chin Hwa
gary@tqstudios.com.sg

$$D = v_z(x+1, y, z+1) + v_z(x-1, y, z-1) - v_z(x-1, y, z+1) - v_z(x+1, y, z-1)$$



Next, we will calculate the change in velocity, $\Delta v_x(x, y, z)$ by using the following equation:

$$\Delta v_x(x, y, z) = \frac{1}{6}\begin{bmatrix} v_x(x+1, y, z) + v_x(x-1, y, z) \\ + v_x(x, y+1, z) + v_x(x, y-1, z) \\ + v_x(x, y, z+1) + v_x(x, y, z-1) \\ - 6v_x(x, y, z) \end{bmatrix}$$



Sho Chin Hwa
gary@tqstudios.com.sg

And finally to find the velocity vector of each lattice point at the next time step $v_x^*(x, y, z)$, we use the following equation:

$$v_x^*(x, y, z) = v_x(x, y, z) + k_v \Delta v_x(x, y, z) + k_p \, grad(divv)_x$$

where k_v is the viscosity ratio and

k_p is the coefficient of the pressure effect

Right now, I'm not too sure what are the correct coefficients to be used in the equation and I'm only guessing that it is clamped between the values of 0 and 1.

Advection

Using the velocity vector from the 1st process, we will simulate the process of advection. This is done by transporting (advecting) all the 3 variable states – water vapor, water droplets and temperature to its new position based on the velocity vector using the following equation.

$$(x + v_x, y + v_y, z + v_z) = (l + \delta x, m + \delta y, n + \delta z) \text{ where}$$

l, m, n are the integer portions and $\delta x, \delta y, \delta z$ are the fractional portions of $(x + v_x, y + v_y, z + v_z)$.

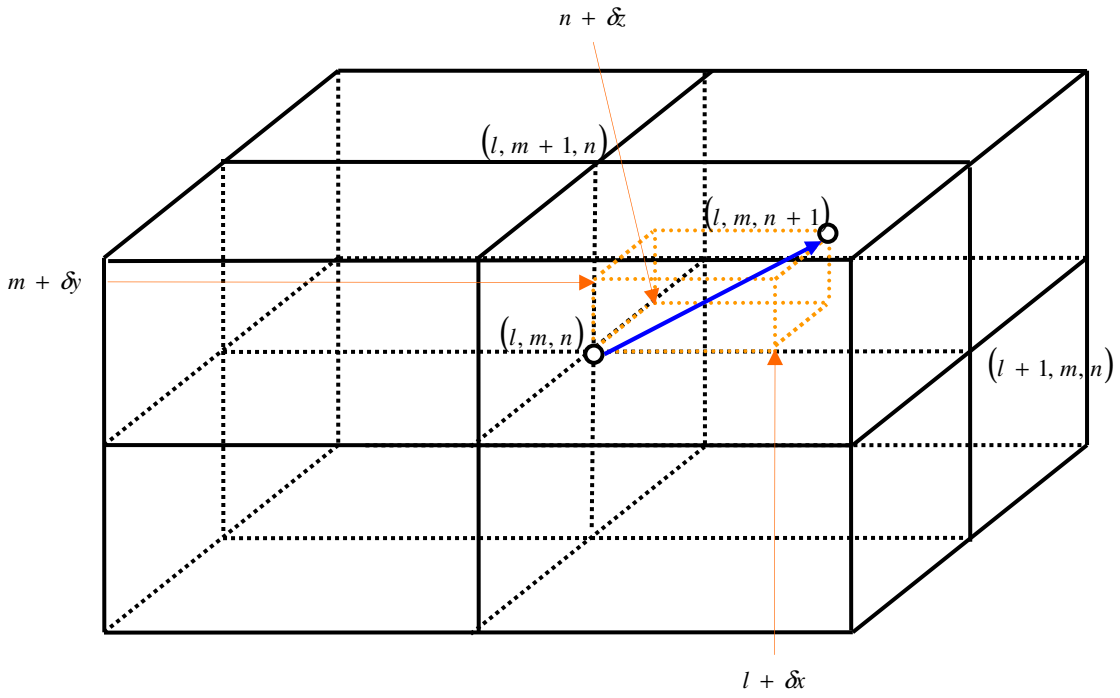
According to the paper, the simulation will become unstable if the following condition is not satisfied.

$$v_x, v_y, v_z < 1.0$$

This means that advection will only transport variable states from one grid point to any one of the nearest 8 blocks of grid lattice containing that grid point itself. Right now, I'm testing the simulation method by setting initial velocity between the range of 0 and 1. Not too sure how the value should be changed but the paper stated that cloud-like shapers are not formed under which the condition is not satisfied.

Most of the time, the new position is not exactly located at the grid points and thus, have to be distributed into the 8 adjacent lattices using a weighted value of the initial states.

Sho Chin Hwa
gary@tqstudios.com.sg



Diffusion of Water Vapor

After transporting the water vapor variables to its new position using the velocity vector, we will then simulate the process of diffusion. Imagine if you drop a blob of ink into a clear pool of water, the ink will slowly diffuse into the surrounding. This phenomenon is similar in the air as the water vapor will diffuse into its surrounding. The discrete version of equation used is as follows:

$$w_x^*(x, y, z) = w_x(x, y, z) + k_{d,w}\Delta w_x(x, y, z) \text{ where}$$

$k_{d,w}$ is the diffusion coefficient for water vapor and Δw_x is the difference of water vapor per time step calculated using the following method:

$$\Delta w_v(x, y, z) = \frac{1}{6}\begin{bmatrix} w_v(x+1, y, z) + w_v(x-1, y, z) \\ + w_v(x, y+1, z) + w_v(x, y-1, z) \\ + w_v(x, y, z+1) + w_v(x, y, z-1) \\ - 6w_v(x, y, z) \end{bmatrix}$$

Sho Chin Hwa
gary@tqstudios.com.sg

Thermal Diffusion

This process acts similarly as the diffusion of water vapor but acts on the temperature state variable instead.

$$E^*(x, y, z) = E(x, y, z) + k_{d,E}\Delta E(x, y, z) \text{ where}$$

$k_{d,E}$ is the coefficient of thermal diffusion and ΔE is the difference in the temperature per time step calculated using the following method:

$$\Delta E(x, y, z) = \frac{1}{6}\begin{bmatrix} E(x+1, y, z) + E(x-1, y, z) \\ + E(x, y+1, z) + E(x, y-1, z) \\ + E(x, y, z+1) + E(x, y, z-1) \\ - 6E(x, y, z) \end{bmatrix}$$

Current work

I am in the process of writing the simulator under the DXUT framework. The basic framework is up – creation of simulation grid, start and pause of simulation with some real-time variable tweaking. First, I will test the velocity vector field because this is the most important variable in this simulation. If it is not working correctly, the other processes such as advection and diffusion might not execute accordingly.

I'm currently tweaking the viscosity and pressure effect as it is giving me some unreliable data under certain conditions. It could be possible that I've mixed up the x, y and z axis used in the paper and mine when coding the algorithm, thus resulting in erratic velocity data. (Already fixed)

Coded and tested the following process:

1. Viscosity and Pressure Effect
2. Advection
3. Diffusion of Water Vapor
4. Thermal Diffusion

Still working on:

1. Buoyancy
2. Phase Transition

The following is the screen shot of the very basic cloud simulator now – showing the velocity field at different time steps.

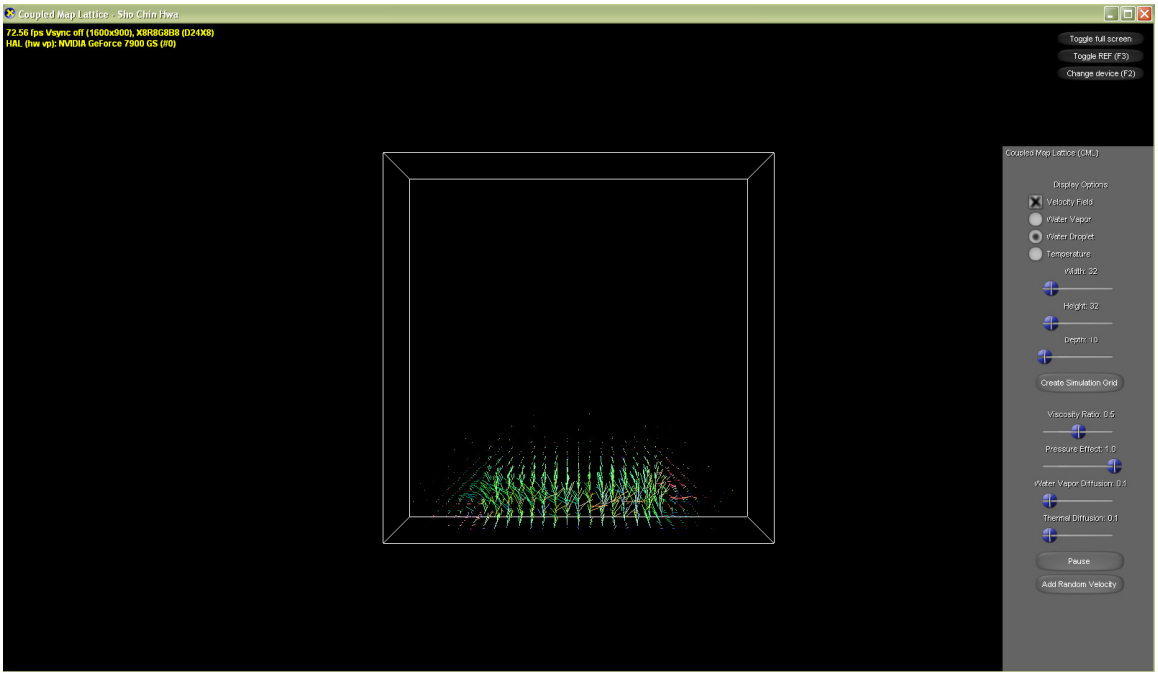


Figure 3: Velocity field at initial time step (constrained to a small portion at the lower grid space with velocity moving up)

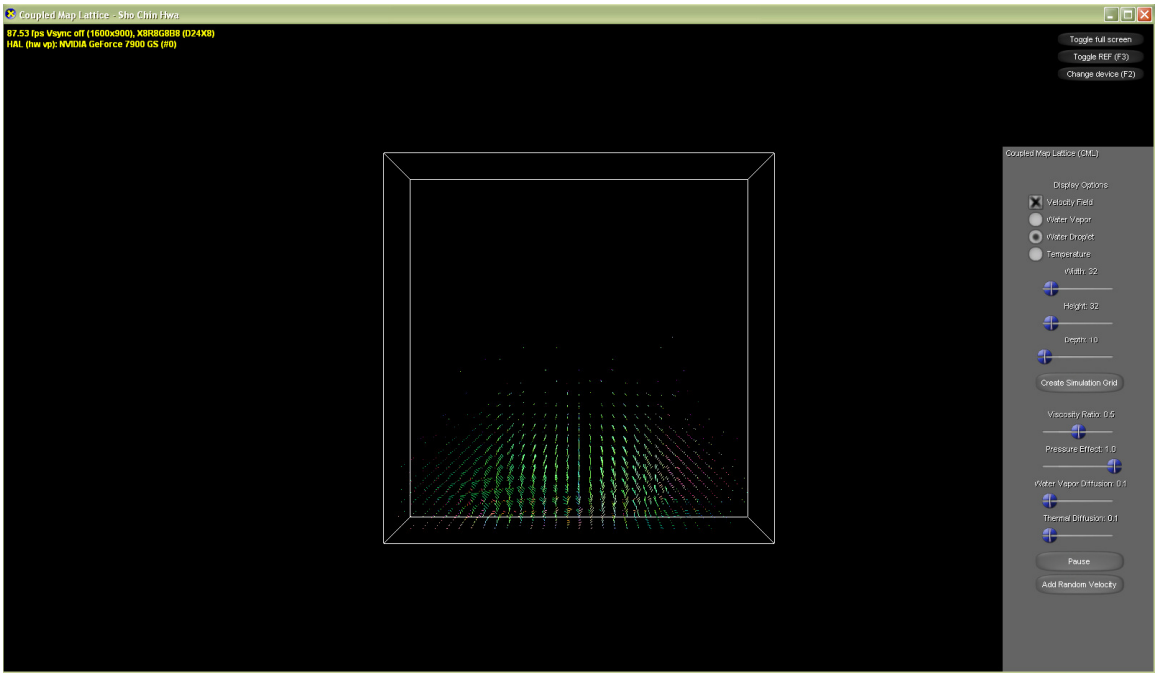


Figure 4: Velocity field after a few time steps (under viscosity and pressure effect)

Sho Chin Hwa
gary@tqstudios.com.sg

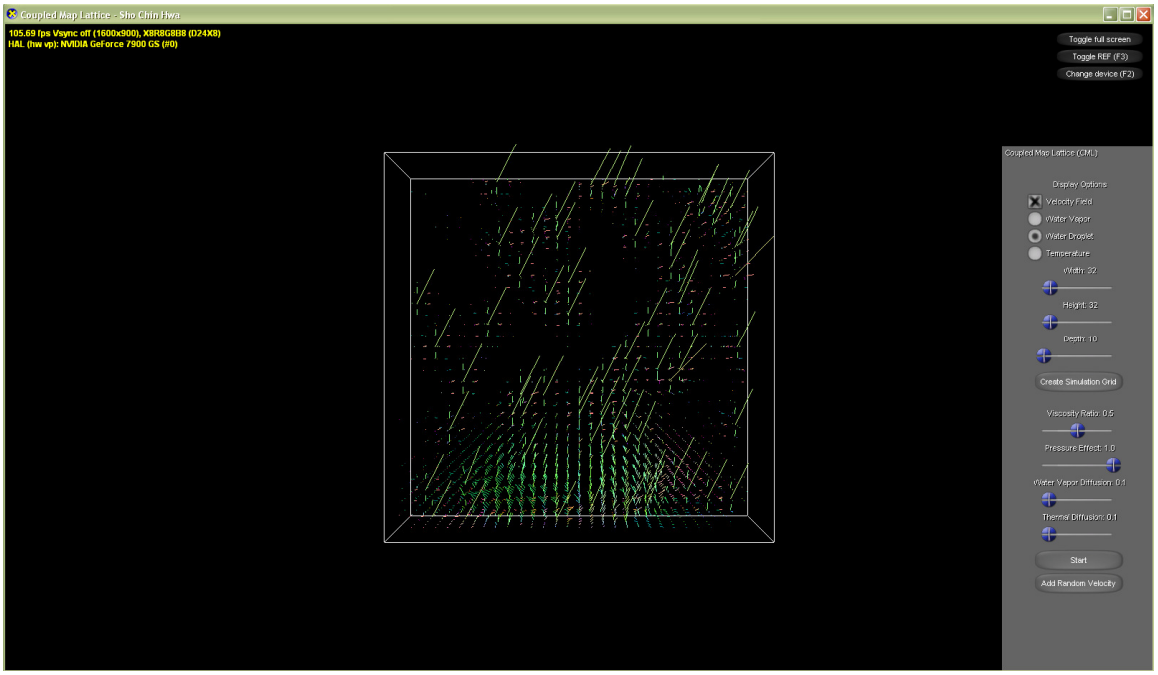


Figure 5: Randomly add velocity field into the space

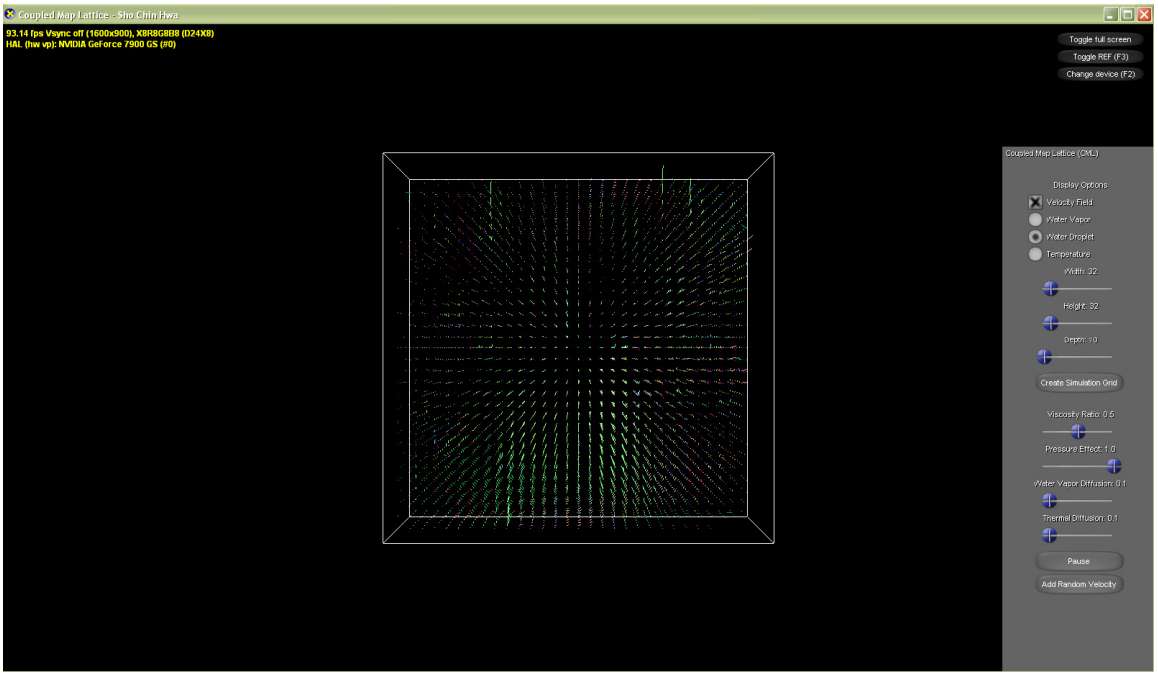


Figure 6: Velocity diffusion with the newly added velocity vector

Sho Chin Hwa
gary@tqstudios.com.sg

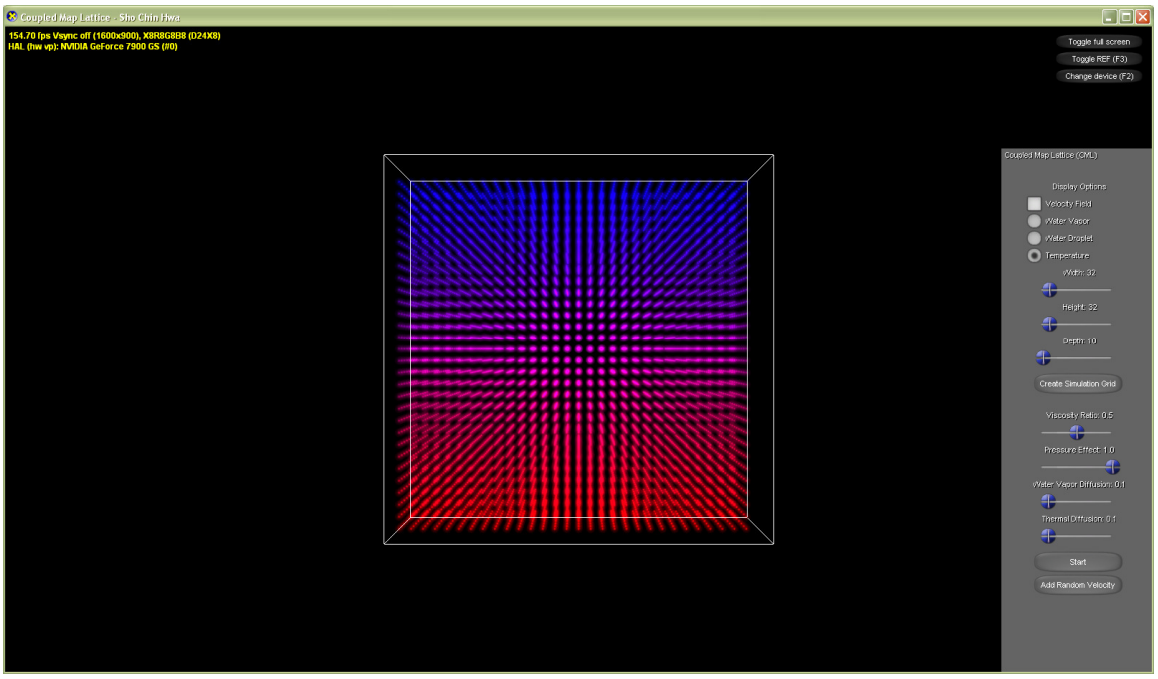


Figure 7: Initial temperature state (300K from ground and drop linearly towards the upper part)

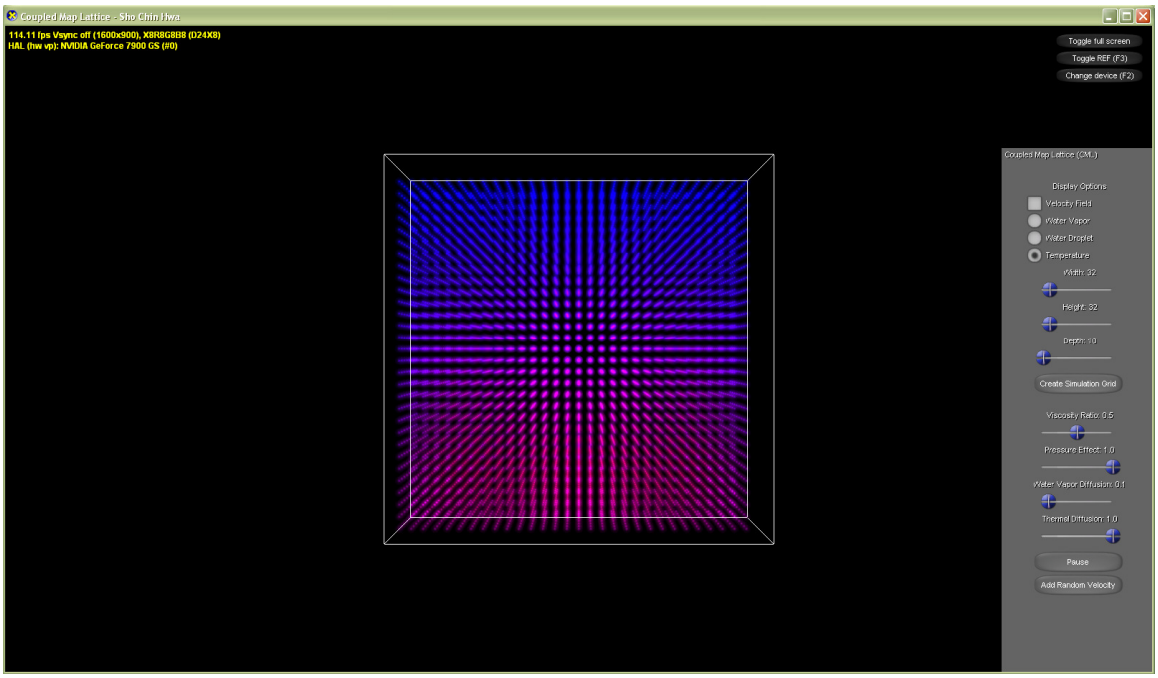


Figure 8: Heat diffusion being turned on (Notice that the ground is no longer as hot because temperature is diffused into the surrounding)

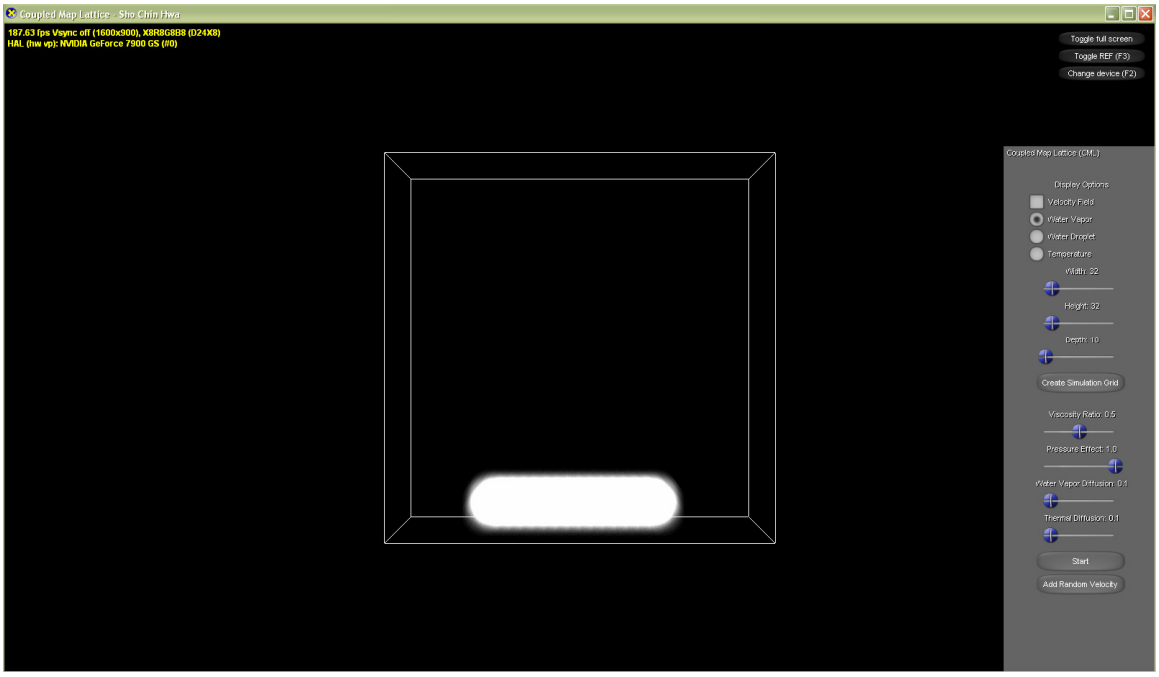


Figure 9: Bed of water vapor at the initial spot

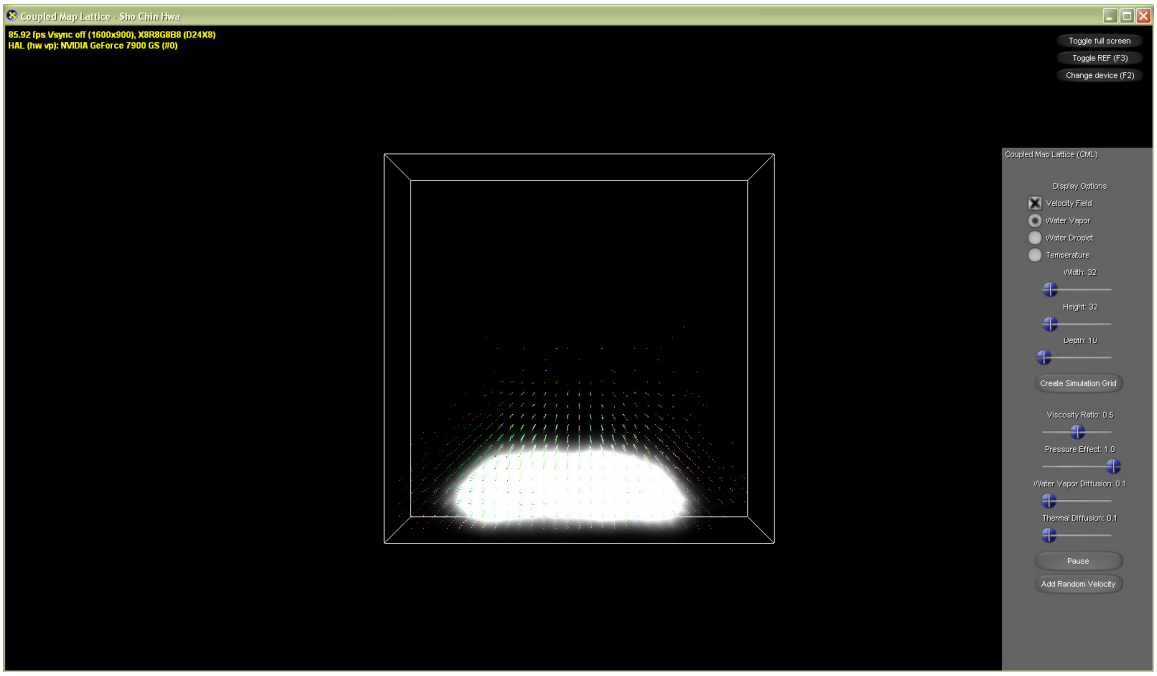


Figure 10: Advection of water vapor according to velocity field (Here velocity field is slowly diffusing outwards)

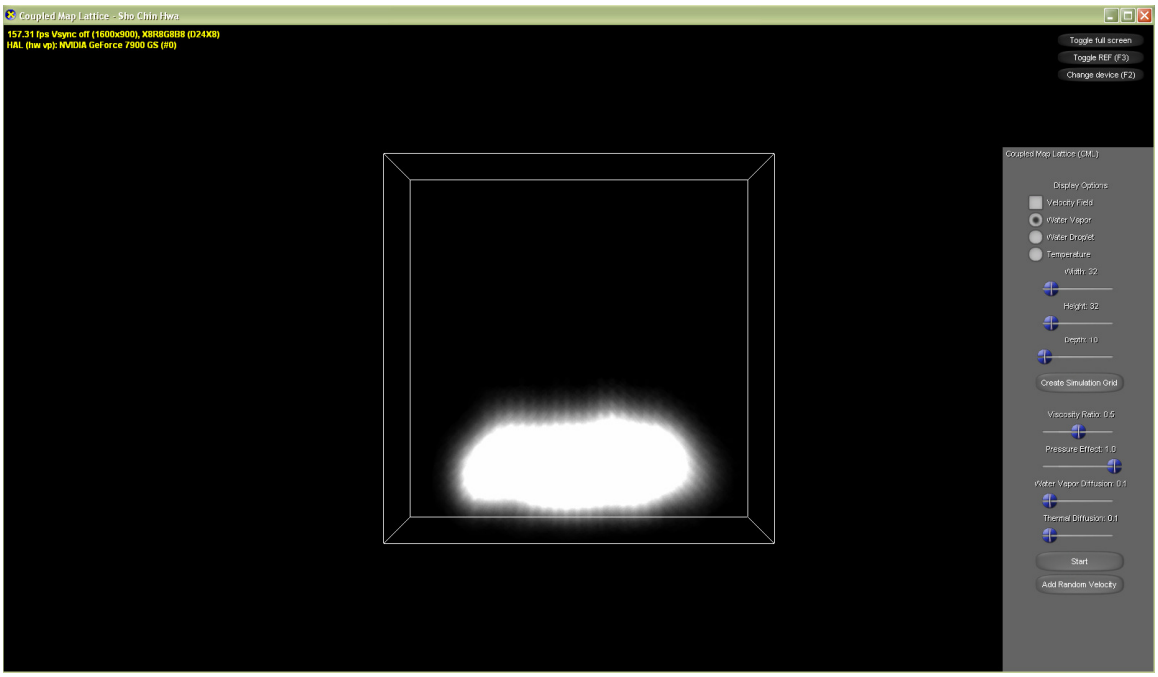


Figure 11: Advection of water vapor

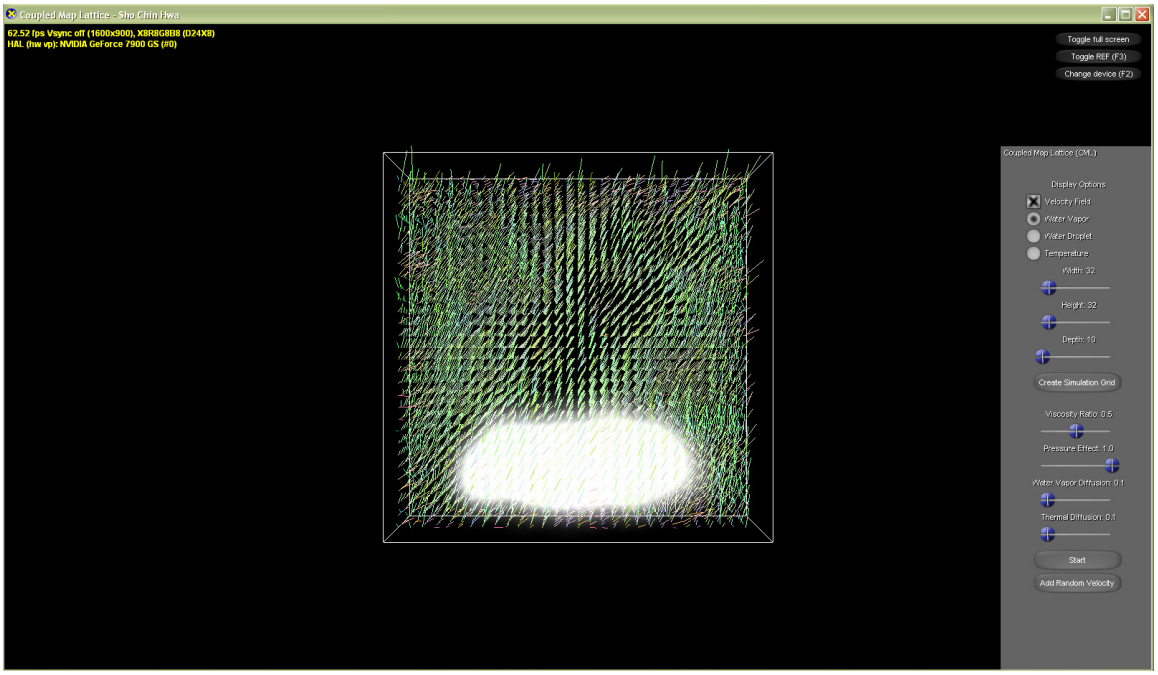


Figure 12: Advection of water vapor with velocity field

Sho Chin Hwa
gary@tqstudios.com.sg

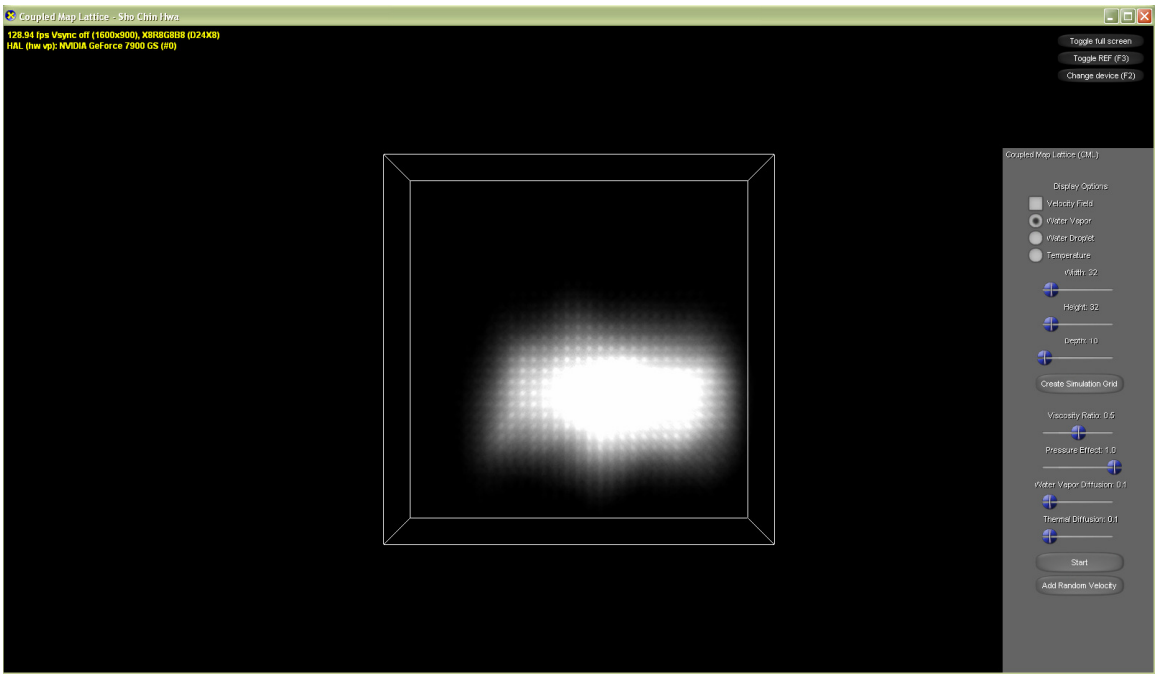


Figure 13: Advection of water vapor

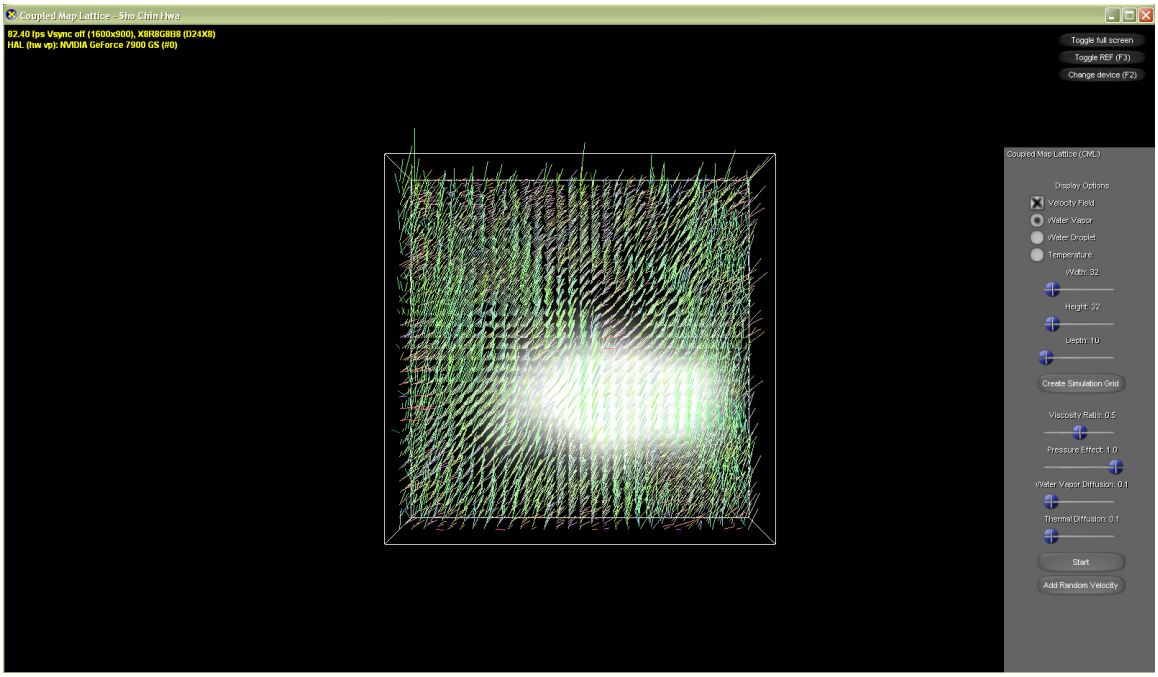


Figure 14: Advection of water vapor with velocity field

Sho Chin Hwa
gary@tqstudios.com.sg

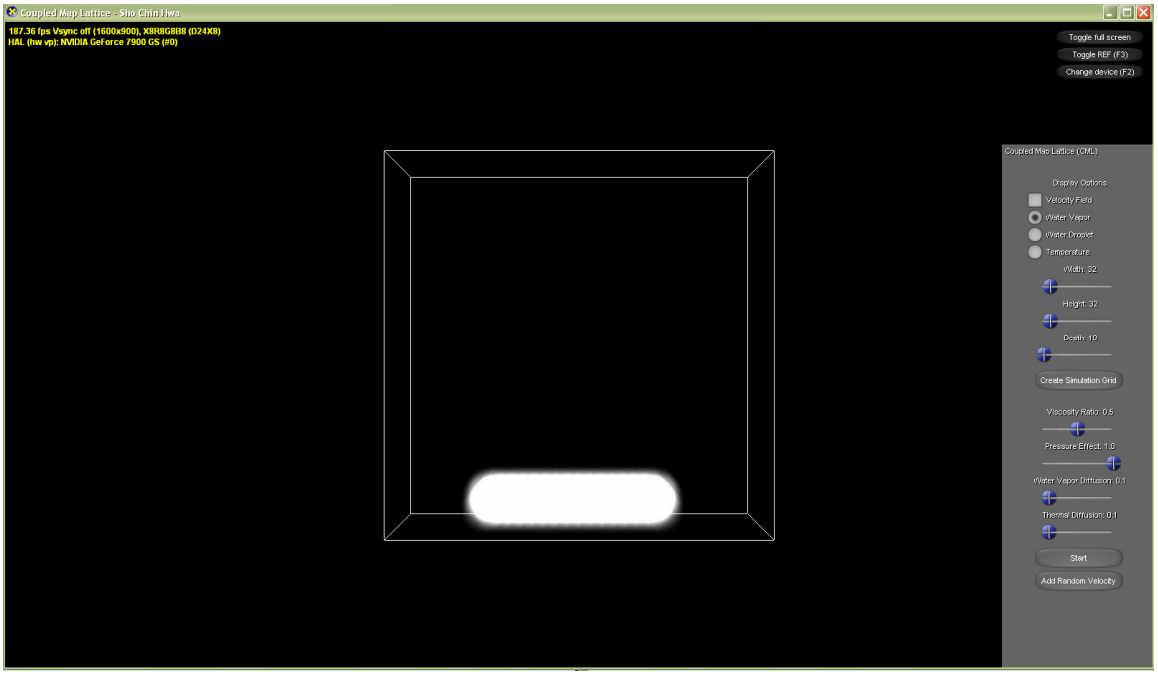


Figure 15: Initial water vapor without diffusion

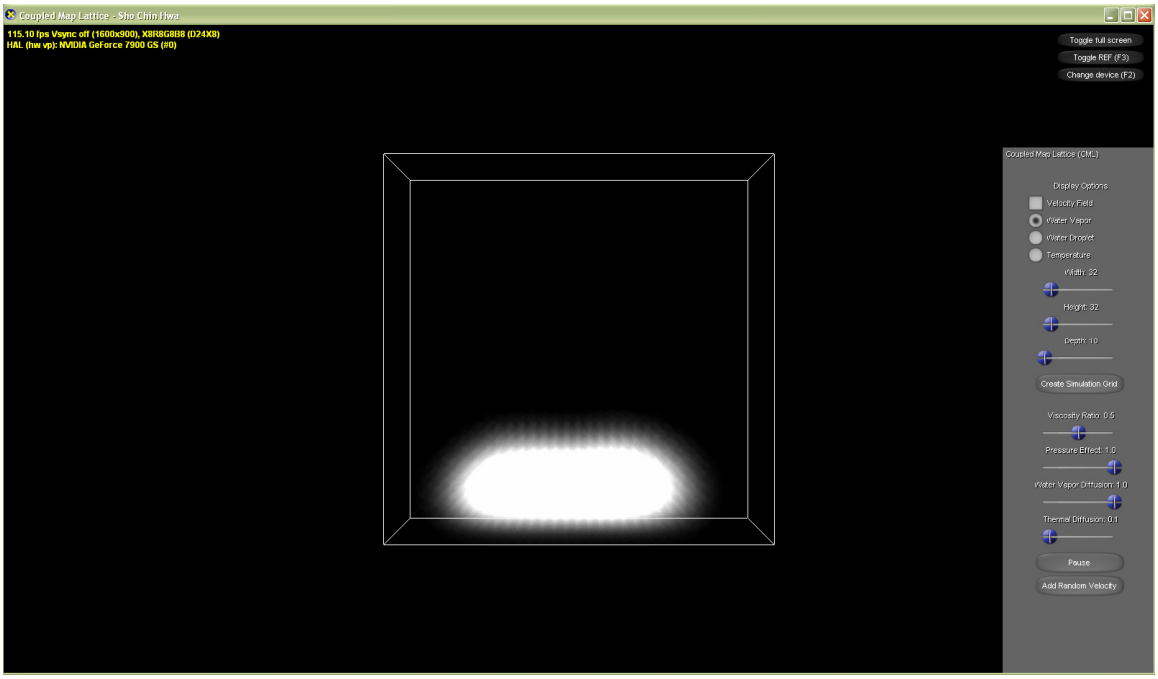


Figure 16: Diffusion of water vapor being simulated.

Sho Chin Hwa
gary@tqstudios.com.sg

REFERENCES

[1] The Size Distribution of Cumulus Clouds in Representative Florida Populations, Vernon G. Plank, 1968, Journal of Applied Meteorology.
[2] Real-time Cloud Simulation and Rendering, Dissertation 2003, Mark Jason Harris.
[3] A Method for Modeling Clouds based on Atmospheric Fluid Dynamics, R. Miyazaki, S. Yoshida, Y. Dobashi, T.Nishita, Proc. Pacific Graphics 2001

Sho Chin Hwa
gary@tqstudios.com.sg