Edward Yang, Yuning Zheng
CEE 101S
Professor Derek Fong
18 August 2017

Final Project Writeup (Powder Game)

Purpose

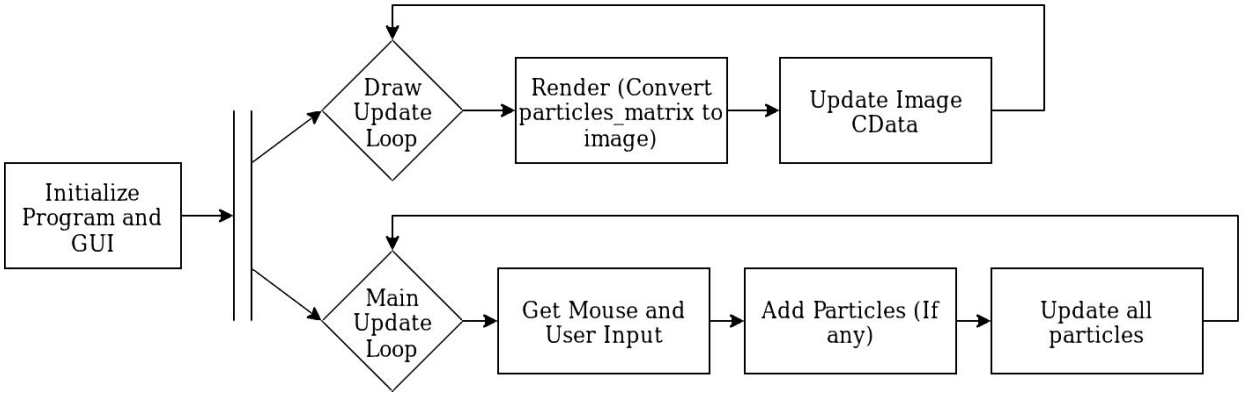
The main purpose of our project was to try and replicate a very basic version of Powder Game, which is essentially a particle simulation sandbox. In doing so, we were trying to see just how well MATLAB could cope with both high frame rate graphics (which it does not normally do well at), and extremely parallelizable operations on thousands of particles at once (which it should be able to do well at).

General Overview of Components

Our main data structures were the cache of actual particles and a matrix of positions and particles at each position. The reason why we had both of these data structures at the same time was because the update function would be orders of magnitude faster than if only one of them was used. If only the particle cache was used, lookup times for neighbors would be enormous, while if only the value matrix was used, there would be a lot of unnecessary iterations and calculations involved.

We had two (presumably asynchronous) timers that executed the main program update loop and the draw loop, which updated the particles and rendered the particles, respectively. In the update loop, user added particles would first be added, and then the simulation run a certain amount of times to get a final update. In the draw loop, the particles were converted into a pixel matrix by indexing a color matrix, which was then set as the CData of the image handle, which was far faster than rerunning imshow.

The GUI served as the main user interface, and allowed the user to change various aspects of their input and the particles within the simulation.



GUI

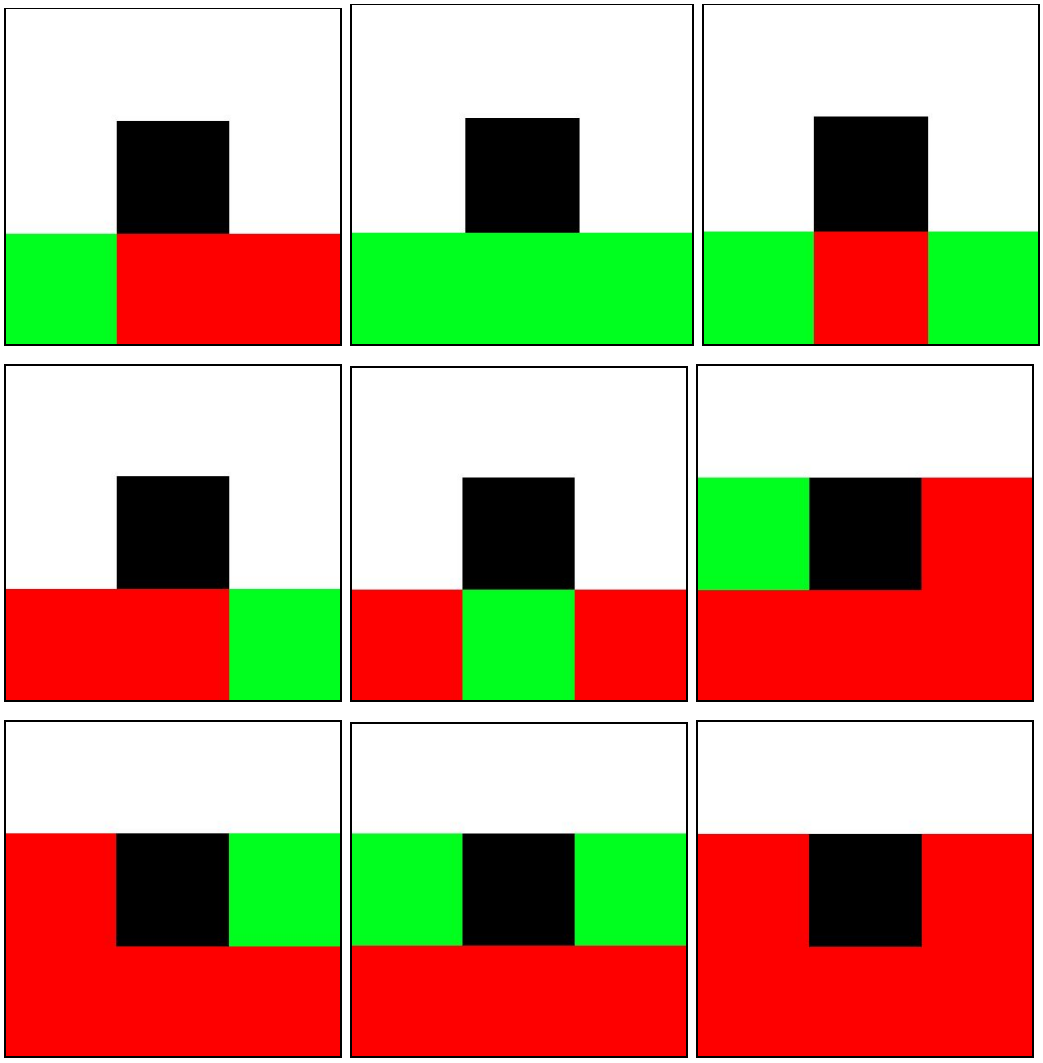
The GUI allows the user to change the type of particle, density of the particle circle, etc., interrupt or continue the flow of time, and clear the simulation when necessary. It aimed to both appear as unintrusive to the user as possible and to allow easy changing of program properties.

Drawing/Rendering

To render everything as quickly as possible, all efforts were made to vectorize the parts that could be vectorized. The colors of each of the values were stored in a matrix, and this was used to allow a vectorized index of the values with the values from the particles_matrix, and then those indexed color values were then transformed so that they matched the correct format for images. The reason why setting CData was used as opposed to just using imshow again is because setting CData is far more efficient—none of the initialization work has to be repeated.

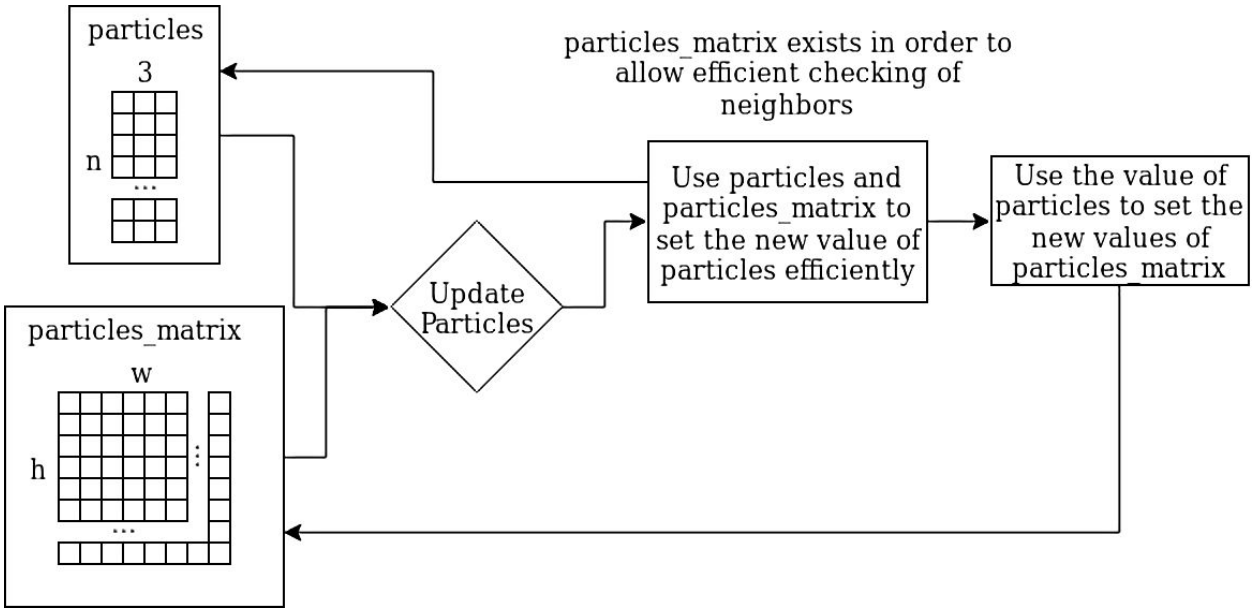
Particle Update Function

The basic logic goes as follows: for liquids, if the bottom neighbors are open, then go to one of the open ones, if not, then go to an open side position. The same logic applied to sand, excepting the move to the side part. This leads to the following logical cases to test for:



Where red is an occupied position and green is a movable position.

The final particle update function, move_particles_v2, arguably make this project what it is. It essentially has the same logic as move_particles, except all of the ifs and elses are entirely vectorized. This is something that MATLAB can do very well, but it also introduced some unnecessary computations that could be avoided if MATLAB normal loops were fast enough to use like in other languages. Basically, for each liquid particle, if the bottom is open, then move to a bottom spot, if it is not, then move to an open side. Solid particles were similar, except they wouldn't move to a side if the bottom was full.



Vectorization

This is probably the most time consuming part (development-wise, a 5 hour marathon) of the program. Each of the if/else logic statements from move_particles was converted to a vectorized form to be applied to particles and particles_matrix. These were then combined to allow the program to run far faster than before (>= 30 times faster on 100k particles, and likely higher factors as the number of particles increase) and essentially keep up with the frame rate.

Unresolved Issues/Conclusion

There are still issues left with this project, such as the realisticness of the simulation being not very close to reality. However, this would essentially require a whole different program strategy: one based on actual physics (i.e. navier-stokes), and which might not necessarily run as well. This was not exactly feasible within the time-frame, as neither of us had any experience with that type of particle system. It would definitely be a project to look at doing in the future though.

Source Code:
https://github.com/efyang/powdergame-matlab