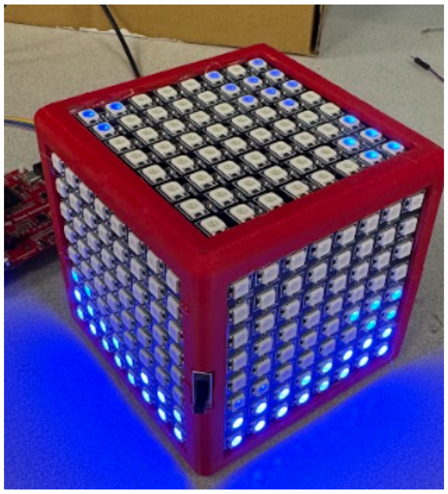
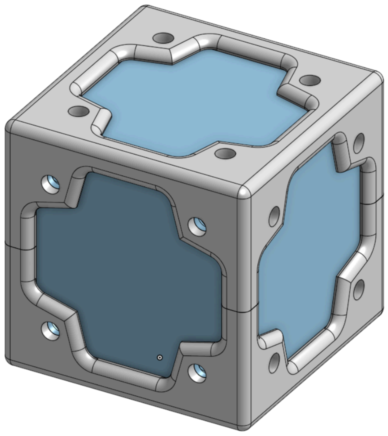
Flui-Cube
ELEC 327 Final Project
Allen Mikhailov

The FluiCube is a cube with an LED grid on each side that displays a fluid simulation that will update and whose gravity is dictated by the orientation of the device. It is powered by a battery so it is standalone and portable.

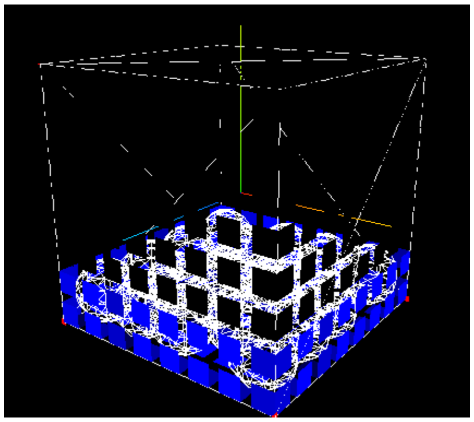


I was inspired by an image of a fluid simulation pendulum and I wanted to make a three dimensional version. I chose a cube because I knew it would be the easiest shape to work with and I have a lot of experience making 3d printed cubes. So I came to call it the FluiCube as a combination of fluid and cube.



The fluid pendant (left) and a past project of mine (right)

I knew that the simulation would be the hardest part of this project. With the processing power of the msp0 a full fledged 3d fluid simulation would be practically impossible. So I went with more of a particle simulation instead. Debugging the simulation on the processor would be an almost impossible challenge so a website was created to test the simulation in an easier to debug environment. Three JS was used to create the graphics and Emscripten is used to compile the c code for the simulation to webassembly so it can be run by the website. This was done so transferring the simulation from the website and onto the device would be as easy as copying the file and removing a few Emscripten flags.

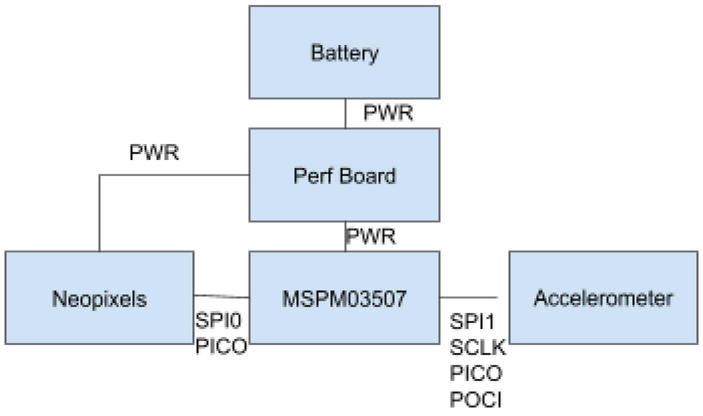


The simulation consists of 30 particles that when close together apply a repelling force to each other. This force is computed by $f = \frac{k}{d+s}$, where “f” is the force applied, “k” is the repulsion constant which can be used to tune the simulation, “d” is the distance between the two particles, “s” is just an extremely small number used to prevent divide by 0 errors. Due to the slow speed of division and square root, the Fast inverse square root algorithm, used in quake 1999, is used to compute $\frac{1}{d+s}$. Particles that are over a certain distance from each other do not have the repelling force applied as this keeps the particles from all sticking to the edges. There is also bounds detection to prevent the particles from going out of the box. A gravitational force is applied to every particle in the simulation which forces the particles down. The gravitational force is also the only input of the simulation which is intended to be set by the accelerometer in the simulation. Damping is also applied to each particle as their velocity is multiplied by 0.95 each frame inorder to keep the simulation stable. Each particle has a size however this has no real effect on the simulation and is for visuals on the website. These rules keep the simulation stable and allow it to attempt to mimic a fluid. The simulation runs on O(n^2) time with n being the number of particles.

The values for the brightness for the leds are calculated by the particles themselves during the simulation step. After each particle has been moved their distance from the walls on each of the three axes are calculated. These values multiplied by a brightness constant are applied to a 3x3 square of pixels on each face. The buffer for these values is actually 10x10 as while the LED grids are 8x8 having the extra one element padding means we do not have to check the bounds when adding to the brightness.

The simulation still has much room for optimization and improvement. Originally it was planned to do more micro optimization once the simulation was working on the device however it seems that is not needed to achieve the actual effect as increasing the timestep allowed the simulation to appear much faster, while still being relatively stable.

The electrical component of the device is relatively simple as the only inputs are from an accelerometer and the only output is to a chain of led grids which just requires a single signal wire. Sadly the accelerometer on my pcb did not function so I was forced to switch to the simon boards given out in class as they were able to replicate the function of my pcb. The protocol to the neopixel led grid could be replicated by the SPImodule on the processor and the simon board already had an accelerometer on it that worked. The accelerometer is communicated with the SPI protocol. To handle power distribution I soldered the ground and power lines of the neopixel connection, simon board, and battery together on a perf board. The battery connection is interrupted by a three position slide switch to allow it to turn on and off. The led grids are connected in series in order: Simon Board → Front → Top → Right → Right → Bottom → Back → Left. The left LED grid is not connected as of now as attaching it would prevent me from flashing the device as I forgot to add a hole for that on the 3d model.



The frame of the device is made of two 3d printed parts. These are the top and bottom parts of the cube. Each face has an indent for the led grid to sit on top of. The bottom half also has a place for the battery to sit on as it is the heaviest component. The two halves join together through a Male-female interlock. The simon board and perfboard are placed on one of the ledges. There is a slot for a switch to fit into as well. This is the second version of the model as the first one was incorrectly dimensioned. Everything is held together with hot glue. Link to CAD: https://cad.onshape.com/documents/76eacfd5498cf4e60ec3783c/w/4f7267ae187bece282b08e1e/e/543ca1d104634286252ed4cd?renderMode=0&uiState=681c2c0c5429d957af8e340d