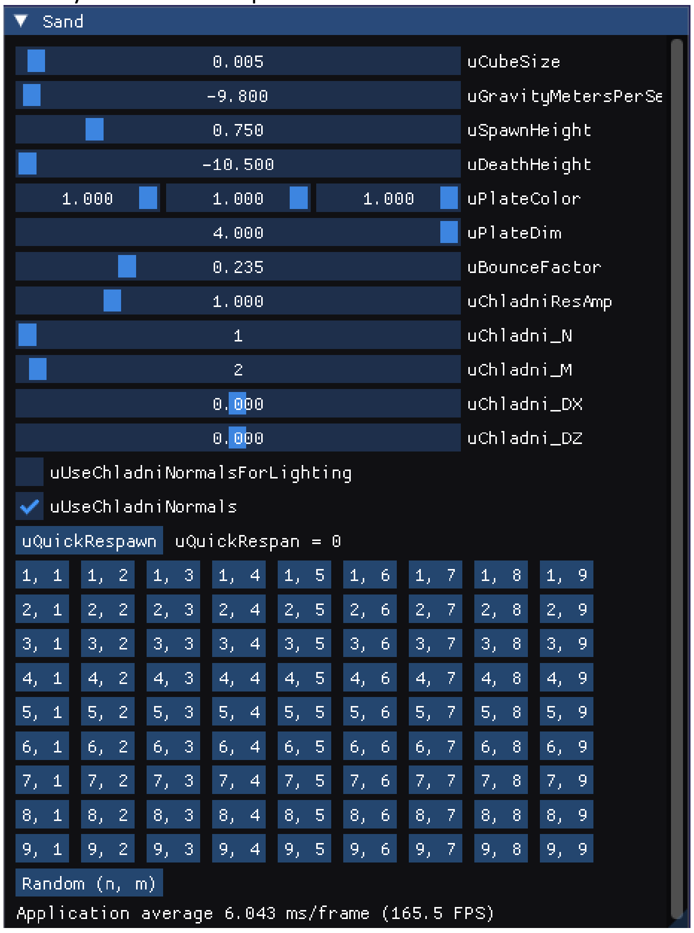
Craig Harris
harricra@oregonstate.edu
Final Project: Chladni Plate Simulation
[Video Link]

A copy of the accepted proposal is at the end of this document.

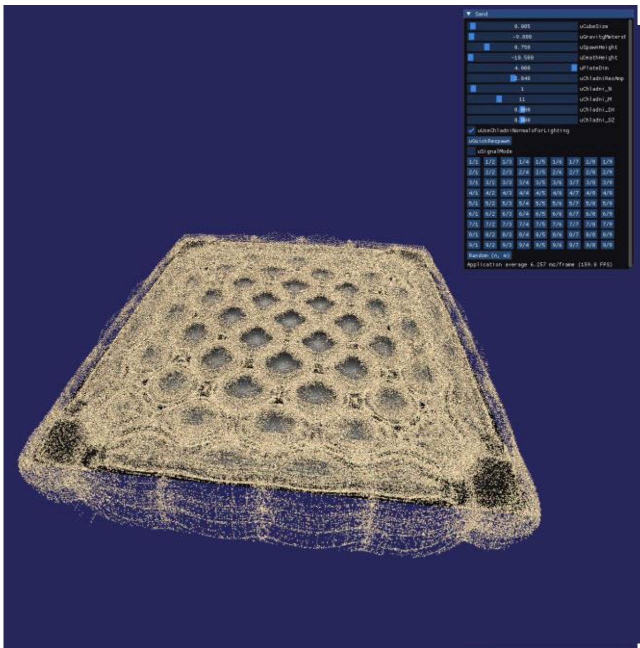
Description:

1. I began by rewriting the sample.cpp to use GLFW. I wanted to be able to use a GUI slider to quickly experiment with the different variables and could not get that working with glut.
2. In the code, several shader storage buffer objects (SSBOs) are set up to represent the sand particles. These hold data such as color, rotation, velocity, and position. The simulation in the linked video uses 128*128*128 = 2,097,152 particles which are updated every frame.
3. A quad is set up to represent the plate that the particles will bounce on.
4. The plate shader colors the quad with fragment lighting with the ability to toggle showing the Chladni resonances with color and bump mapping.
5. The sand compute shader updates each sand particle by checking if the particle intersects with the plate or if it should continue falling. If it intersects with the plate, the normal is used to calculate a standard reflective bounce vector, the amplitude of the resonance and the direction of the normal at the hit position are used to calculate a Chladni resonant vector. These two vectors are added together as the new velocity of the particle. Gravity is applied on each frame as well.
6. The sand render shader takes the particles positions as vertices, uses a geometry shader to turn them into cubes, then the fragment shader calculates the per fragment lighting,
7. These shaders are set up with uniform variables allowing for the user to change the size of the plate, choose whether to show the Chladni resonance amplitudes (with color and bump maps), change the sand particle size, and modify the N and M inputs to the Chladni function
   a. 

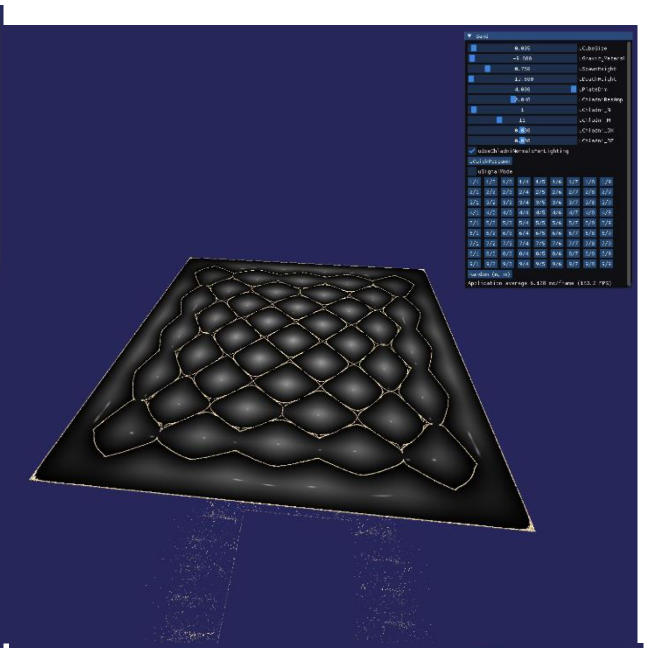
```
▼ Sand
          0.005                 uCubeSize
         -9.800                 uGravityMetersPerSe
          0.750                 uSpawnHeight
        -10.500                 uDeathHeight
 1.000     1.000     1.000      uPlateColor
          4.000                 uPlateDim
          0.235                 uBounceFactor
          1.000                 uChladniResAmp
            1                   uChladni_N
            2                   uChladni_M
          0.000                 uChladni_DX
          0.000                 uChladni_DZ
[ ] uUseChladniNormalsForLighting
[✓] uUseChladniNormals
uQuickRespawn  uQuickRespan = 0
1, 1  1, 2  1, 3  1, 4  1, 5  1, 6  1, 7  1, 8  1, 9
2, 1  2, 2  2, 3  2, 4  2, 5  2, 6  2, 7  2, 8  2, 9
3, 1  3, 2  3, 3  3, 4  3, 5  3, 6  3, 7  3, 8  3, 9
4, 1  4, 2  4, 3  4, 4  4, 5  4, 6  4, 7  4, 8  4, 9
5, 1  5, 2  5, 3  5, 4  5, 5  5, 6  5, 7  5, 8  5, 9
6, 1  6, 2  6, 3  6, 4  6, 5  6, 6  6, 7  6, 8  6, 9
7, 1  7, 2  7, 3  7, 4  7, 5  7, 6  7, 7  7, 8  7, 9
8, 1  8, 2  8, 3  8, 4  8, 5  8, 6  8, 7  8, 8  8, 9
9, 1  9, 2  9, 3  9, 4  9, 5  9, 6  9, 7  9, 8  9, 9
Random (n, m)
Application average 6.043 ms/frame (165.5 FPS)
```

8. The Chladni resonance function was modified from this website and this website.
   a. cos(n pi x / L) cos(m pi y / L) + cos(m pi x / L) cos(n pi y / L)

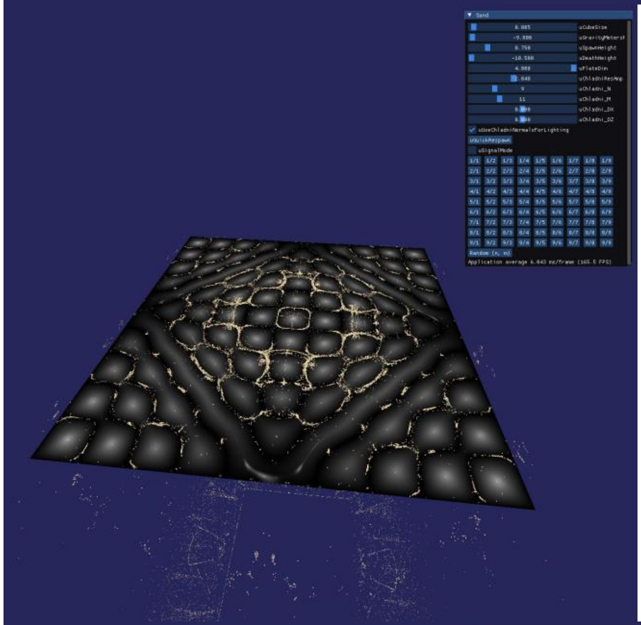
Screen Shots:



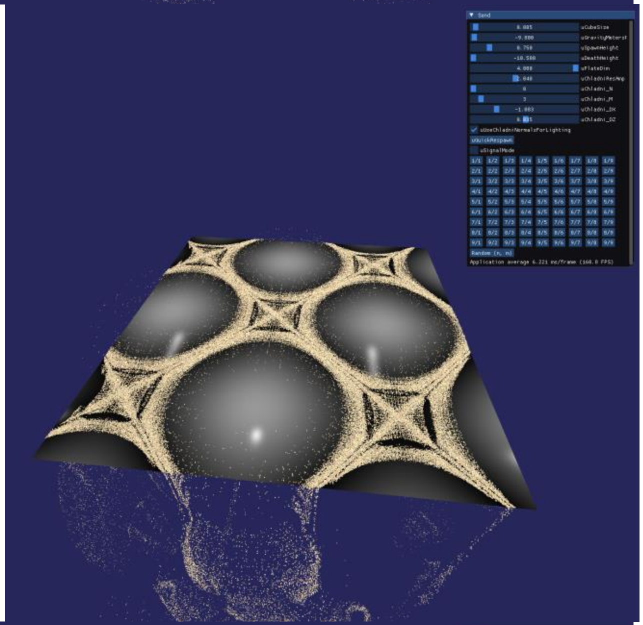
```
0.005      uCubeSize
-9.800     uGravityMetersF
0.750      uSpawnHeight
-10.500    uDeathHeight
4.000      uPlateDim
2.048      uChladniResAmp
1          uChladni_N
11         uChladni_M
0.000      uChladni_DX
0.000      uChladni_DZ
[✓] uUseChladniNormalsForLighting
[ ] uSignalMode
Application average 6.257 ms/frame (159.8 FPS)
```



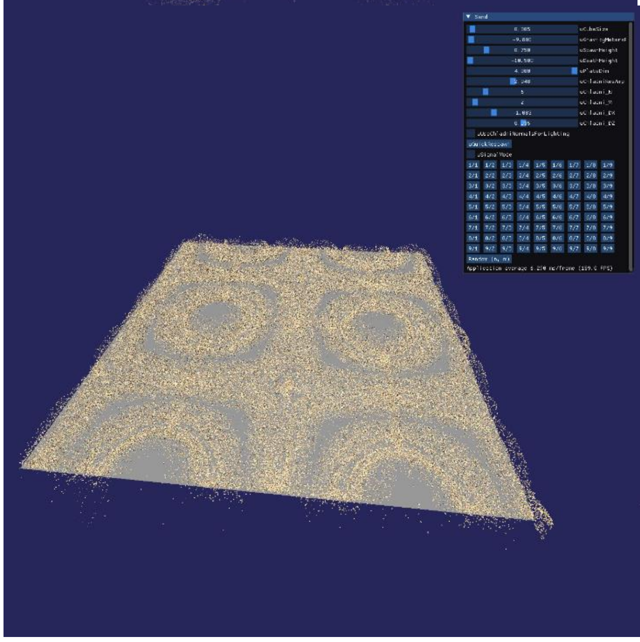
```
0.005      uCubeSize
-9.800     uGravityMetersF
0.750      uSpawnHeight
10.500     uDeathHeight
4.000      uPlateDim
2.048      uChladniResAmp
1          uChladni_N
11         uChladni_M
0.000      uChladni_DX
0.000      uChladni_DZ
[✓] uUseChladniNormalsForLighting
[ ] uSignalMode
Application average 6.128 ms/frame (153.2 FPS)
```



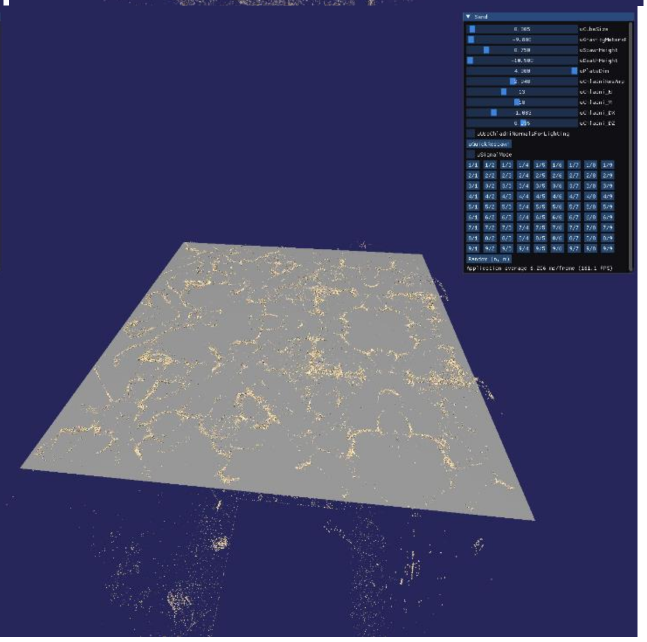
```
0.005      uCubeSize
-9.800     uGravityMetersF
0.750      uSpawnHeight
-10.500    uDeathHeight
4.000      uPlateDim
2.048      uChladniResAmp
9          uChladni_N
11         uChladni_M
0.000      uChladni_DX
0.000      uChladni_DZ
[✓] uUseChladniNormalsForLighting
[ ] uSignalMode
Application average 6.043 ms/frame (165.5 FPS)
```



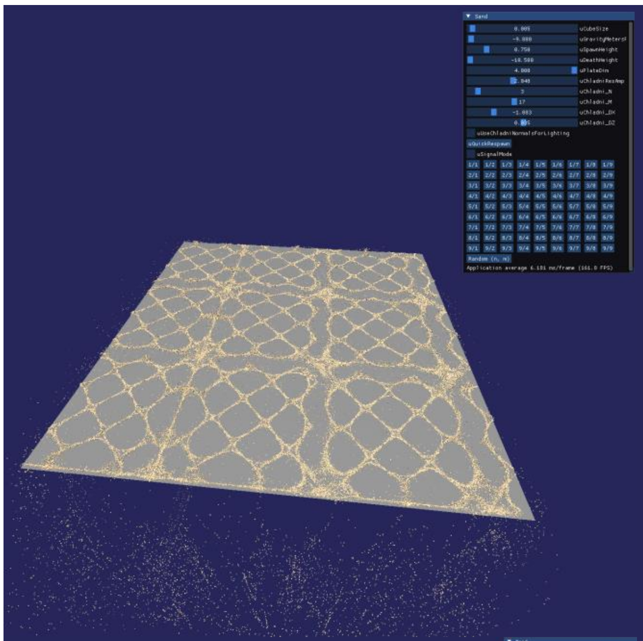
```
0.005      uCubeSize
-9.800     uGravityMetersF
0.750      uSpawnHeight
-10.500    uDeathHeight
4.000      uPlateDim
2.048      uChladniResAmp
0          uChladni_N
3          uChladni_M
-1.003     uChladni_DX
0.035      uChladni_DZ
[✓] uUseChladniNormalsForLighting
[ ] uSignalMode
Application average 6.221 ms/frame (160.8 FPS)
```



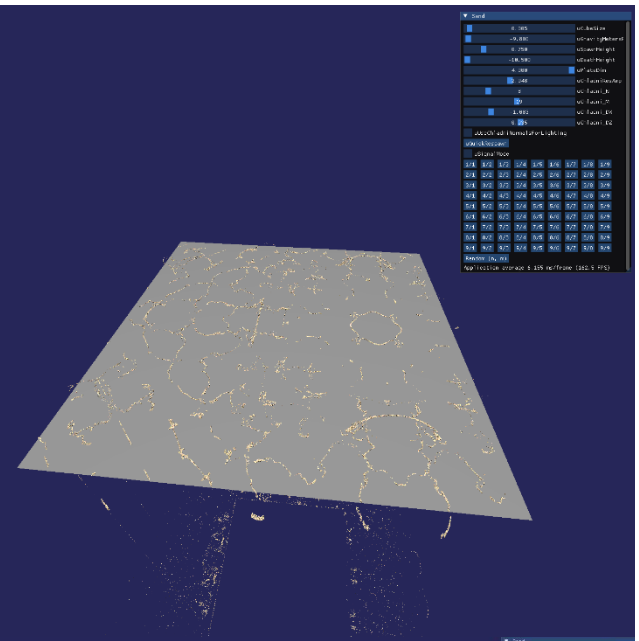
```
0.005      uCubeSize
-9.800     uGravityMetersF
0.750      uSpawnHeight
-10.500    uDeathHeight
4.000      uPlateDim
2.048      uChladniResAmp
5          uChladni_N
2          uChladni_M
1.003      uChladni_DX
0.035      uChladni_DZ
[ ] uUseChladniNormalsForLighting
[ ] uSignalMode
Application average 6.270 ms/frame (159.6 FPS)
```



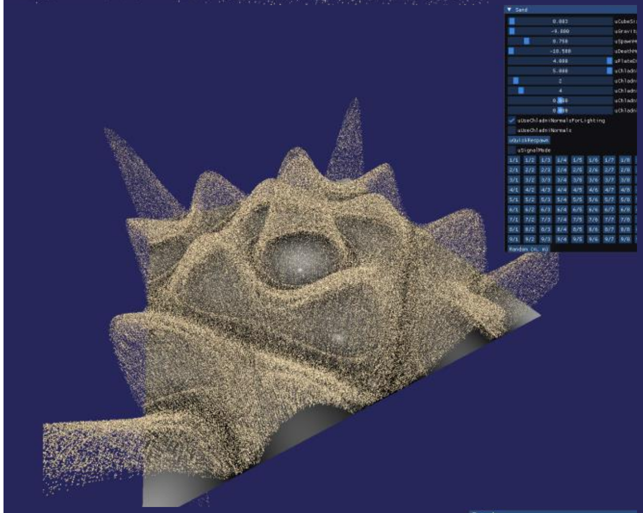
```
0.005      uCubeSize
-9.800     uGravityMetersF
0.750      uSpawnHeight
-10.500    uDeathHeight
4.000      uPlateDim
2.048      uChladniResAmp
13         uChladni_N
18         uChladni_M
1.003      uChladni_DX
0.035      uChladni_DZ
[ ] uUseChladniNormalsForLighting
[ ] uSignalMode
Application average 6.206 ms/frame (161.1 FPS)
```



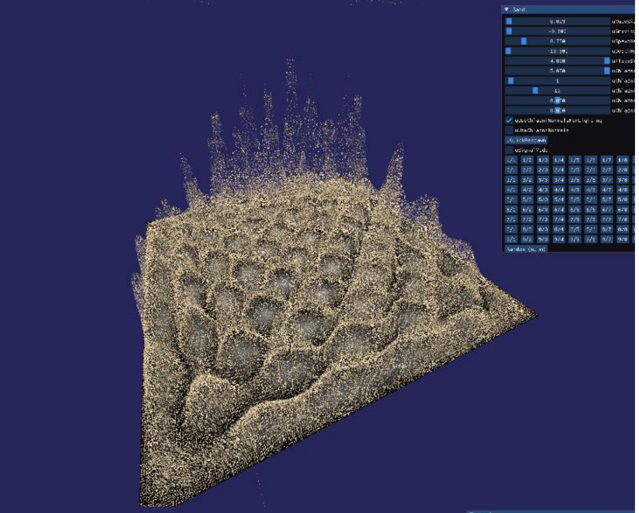
```
0.005      uCubeSize
-9.800     uGravityMetersP
0.750      uSpawnHeight
-10.500    uDeathHeight
4.000      uPlateDim
2.040      uChladniResAmp
3          uChladni_N
17         uChladni_M
-1.083     uChladni_DX
0.005      uChladni_DZ
[ ] uUseChladniNormalsForLighting
[ ] uSignalMode
Application average 6.181 ms/frame (161.8 FPS)
```



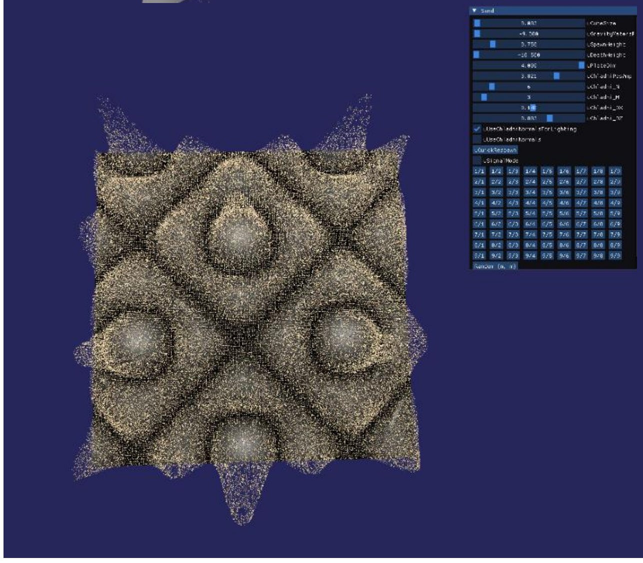
```
0.005      uCubeSize
-9.800     uGravityMetersP
0.750      uSpawnHeight
-10.500    uDeathHeight
4.000      uPlateDim
2.040      uChladniResAmp
5          uChladni_N
19         uChladni_M
-1.083     uChladni_DX
0.035      uChladni_DZ
[ ] uUseChladniNormalsForLighting
[ ] uSignalMode
Application average 6.155 ms/frame (162.5 FPS)
```



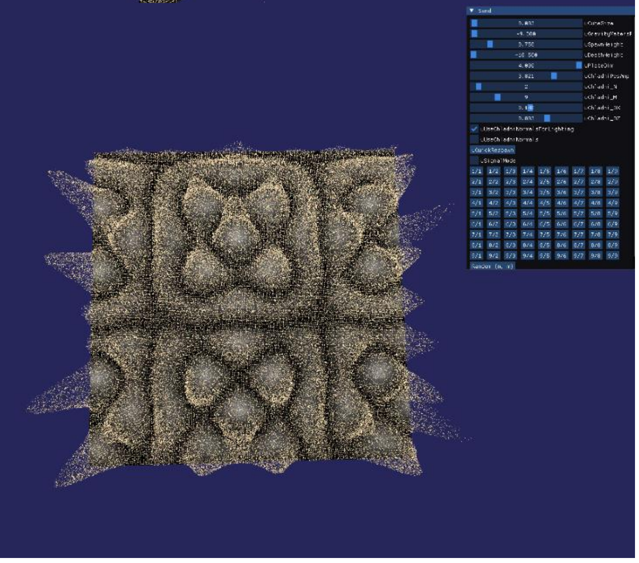
```
0.002      uCubeSi
-9.800     uGravit
0.750      uSpawnH
-10.500    uDeathH
4.000      uPlateD
5.000      uChladn
2          uChladn
4          uChladn
0.000      uChladn
0.000      uChladn
[✓] uUseChladniNormalsForLighting
[ ] uUseChladniNormals
[ ] uSignalMode
```



```
0.003      uCubeSi
-9.800     uGravit
0.770      uSpawnH
-10.500    uDeathH
4.000      uPlateD
5.000      uChladn
1          uChladn
11         uChladn
0.000      uChladn
0.000      uChladn
[✓] uUseChladniNormalsForLighting
[ ] uUseChladniNormals
[ ] uSignalMode
```



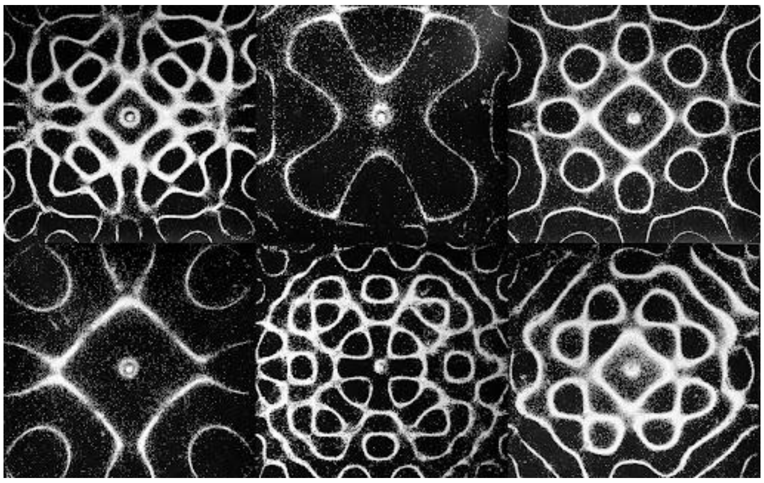
```
0.002      uCubeSize
-9.800     uGravityMetersP
0.750      uSpawnHeight
-10.500    uDeathHeight
4.000      uPlateDim
3.821      uChladniPosAmp
6          uChladni_N
3          uChladni_M
0.140      uChladni_DX
0.003      uChladni_DZ
[✓] uUseChladniNormalsForLighting
[ ] uUseChladniNormals
[ ] uSignalMode
```



```
0.002      uCubeSize
-9.800     uGravityMetersP
0.750      uSpawnHeight
-10.500    uDeathHeight
4.000      uPlateDim
3.821      uChladniPosAmp
2          uChladni_N
9          uChladni_M
0.140      uChladni_DX
0.003      uChladni_DZ
[✓] uUseChladniNormalsForLighting
[ ] uUseChladniNormals
[ ] uSignalMode
```

Craig Harris
harricra@oregonstate.edu
Final Project Proposal: Chladni Plate Shader

Description: Chladni plates are used to show the modes of vibration on a rigid surface. Activating the resonance was first done by sliding a violin bow across a metal plate, but today it's typically done by attaching the metal plate to a speaker driver. When sand is covering the top of the plate, and the plate is resonating, the sand falls into the gaps where no movement is occurring (node) as the vibrating part of the plate (antinode) pushes the sand away. When the resonance is changed, the sand flows to the new pattern of nodes/antinodes.


Different patterns created on Chladni plates

Plan: To recreate sand moving in patterns as noted above. There are three main hurtles I see:

1. Recreate the pattern in a fragment shader. I have found some equations online that should make this fairly simple. There could be added complexity for adding variability of the shape of the plate (circle, violin, etc) or the point at which the plate is being activated.
2. Simulate the depth and imperfections of the system -- the sand groups into piles but is not perfectly within the pattern, some lines are thinner/wider, some sand is still sitting in the antinode area.. This part can likely be done with vertex displacement, noise, and per-frag lighting. I might play with shadows too.
3. The trickiest part I would like to implement is the transition between patterns; when one resonance is switched to another, the sand has to move to the new nodes. I've purposefully not looked at outside resources yet as I want to attempt this on my own for a bit. I think some sort of motion vector pass could be used. This video shows an example of switching between the resonances.

If that all goes smoothly, I want to look at modifying it for 3 dimensions and adding some sort of color mapping.