Visualizing Lake Fred

Mohammed Mowla

Agenda

1 – Data Processing
2 – Creating the Model
3 – 3D Printing
4 – Website

Data Processing

Generating data that is usable in blender

Data Processing

- Python
- Libraries NumPy and Matplotlib
- Data retrieved from Vernier LabQuest 3 as a CSV (Comma Separated Value) file
- Multiple Assignments

```python
import numpy as np
import matplotlib.pyplot as plt

# Retrieving data from the file
data = np.loadtxt("lake-data-raw.csv", float, skiprows=1, delimiter=",")
x_vals, y_vals, z_vals = data[:, 0], data[:, 1], -data[:, 2]
```

Data Processing (Cont.)

- Filtering non-unique x and y values
- A set in Python holds only unique values
  - Ex Array: [11, 41, 53, 22, 11]
  - Ex Set: {11, 41, 53, 22}

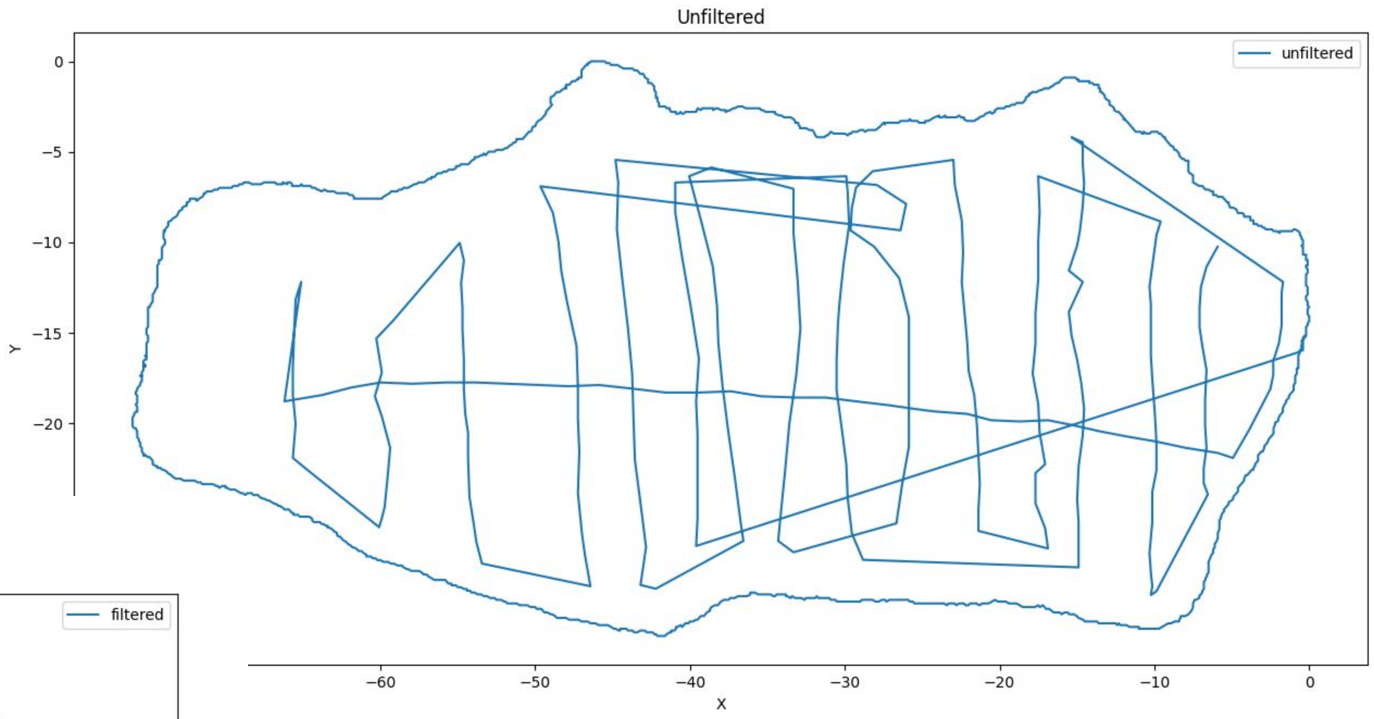
```python
# Removing values that hold insignificant x and y values
for i in range(len(x_vals)):
    curr_x, curr_y, curr_z = x_vals[i], y_vals[i], z_vals[i]

    # Check if the current x is unique
    if curr_x not in unique_x and curr_z == 0:
        x_filtered.append(x_vals[i])
        y_filtered.append(y_vals[i])
        z_filtered.append(z_vals[i])
    elif curr_z < 0:
        x_filtered.append(x_vals[i])
        y_filtered.append(y_vals[i])
        z_filtered.append(z_vals[i])
    # Check if the current y is unique
    if curr_y not in unique_y and curr_z == 0:
        x_filtered.append(x_vals[i])
        y_filtered.append(y_vals[i])
        z_filtered.append(z_vals[i])
    elif curr_z < 0:
        x_filtered.append(x_vals[i])
        y_filtered.append(y_vals[i])
        z_filtered.append(z_vals[i])

    # Adding current x and y values to sets
    unique_x.add(curr_x)
    unique_y.add(curr_y)
```

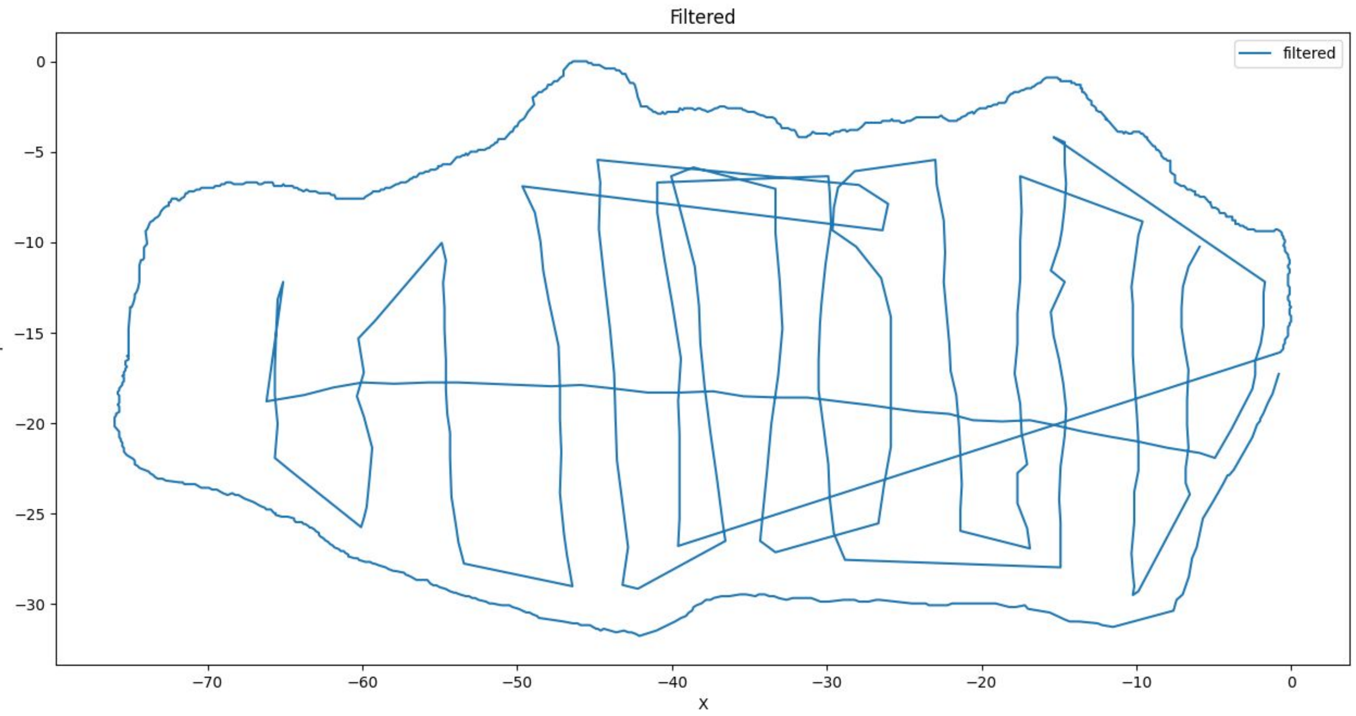
Data Processing (Cont.)

- Scaling the data
  - Why?
  - Ex: Distance between two adjacent points is $7.19 * 10^{-5}$
- The distance between each data point is extremely small.
- Using Blender to visualize these extremely small distance values results in a model that is nearly imperceptible.
- Point total: 1771 to 1480
  - 291 Removed

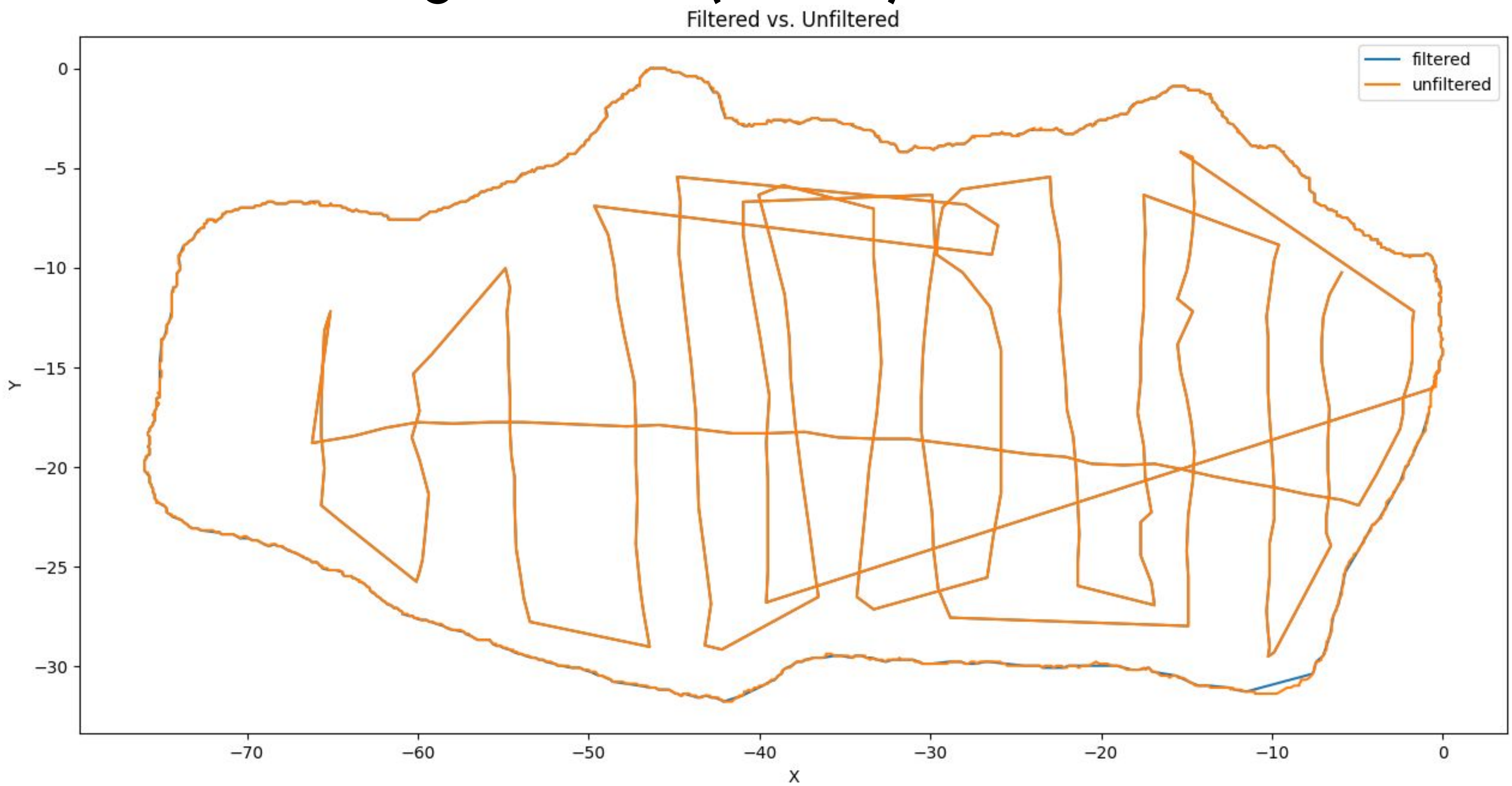
```python
# Maximum x and y to scale
max_x = max(x_vals)
max_y = max(y_vals)

# Scaling points to be usable in Blender
scale_mult = 10000
x_filtered = [scale_mult*(x - max_x) for x in x_filtered]
y_filtered = [scale_mult*(y - max_y) for y in y_filtered]
x_vals = [scale_mult*(x - max_x) for x in x_vals]
y_vals = [scale_mult*(y - max_y) for y in y_vals]


file = open('lake_data_processed.txt', 'w')
for x in range(len(x_filtered)):
    file.write(str(x_filtered[x]))
    file.write(',')
    file.write(str(y_filtered[x]))
    file.write(',')
    file.write(str(z_filtered[x]))
    file.write('\n')
file.close()
```

Data Processing Results





Data Processing Results (Cont.)



Creating the Model

Creating the 3D model in Blender Using Delaunay Triangulation

Creating the Model : Blender

- Blender is an open-source 3D computer software tool
  - Supports modeling, rigging, animation, simulations, rendering,…
  - Includes scripting through Python!
- The most obvious use case for blender is to create 3D models for video games.
- There are scientific use cases and libraries in Blender
  - Ex: BlenderGIS
    - Easily import (satellite) maps, displacement maps and geometry like buildings

How Do You Make a 3D Model?

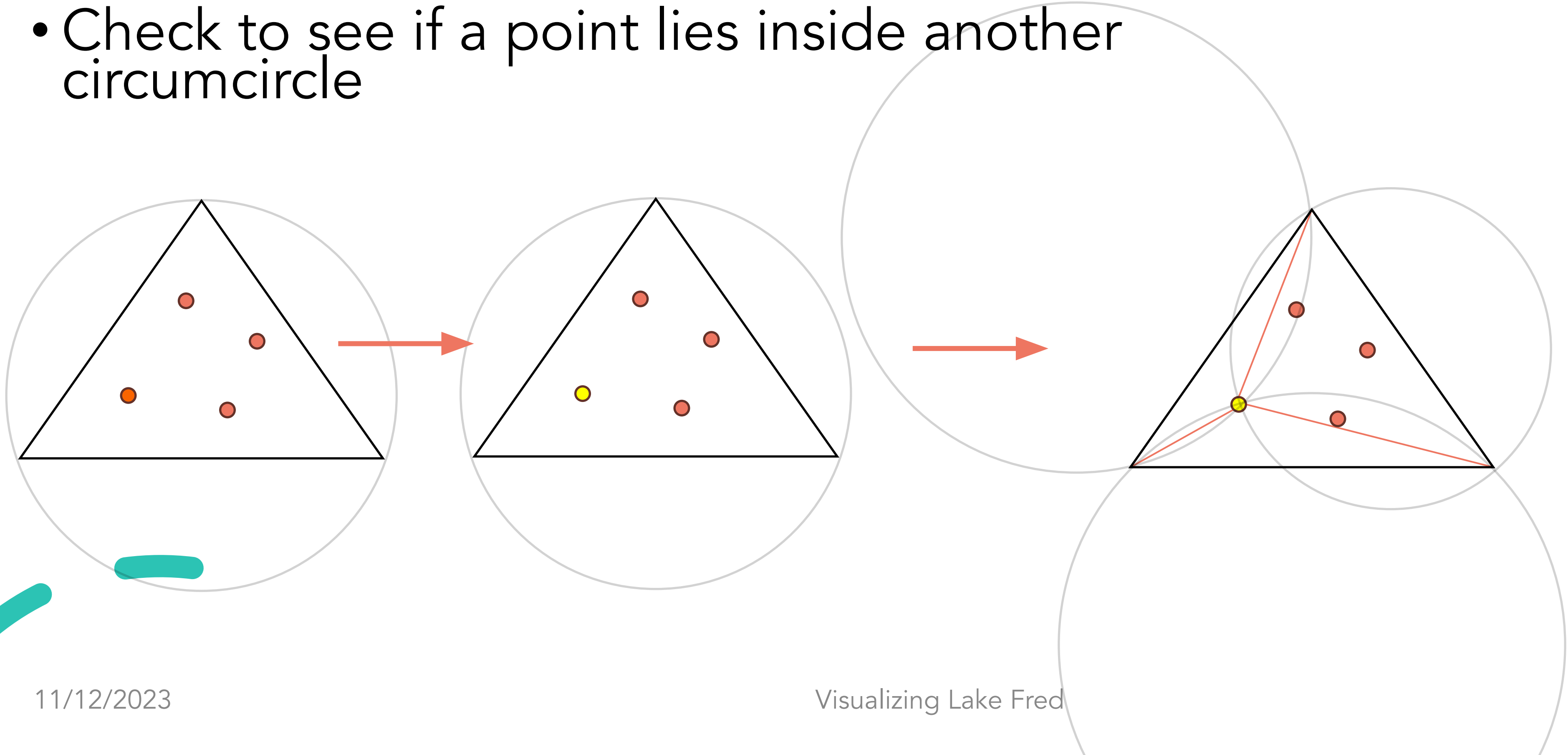
- Identified that we have a point cloud (A discrete set of data points in 3D space)
- How do we take a point cloud and make something from it?
- Triangulation!

Delaunay Triangulation

- Definition: Delaunay Triangulation (DT)
  - For a set $\{p_i\}$ of discrete points p_i in a general position is a triangulation such that no point is inside the circumcircle of any triangle in the DT.
  - In Algebraic Geometry and Computational Geometry:
    - General Position (Points Only): An arrangement of points where no three points are colinear (Lie in a straight line)
  - A circumcircle is a circle that passes through all the vertices of a given polygon. In our case a triangle
- Several Types of Algorithms for computing DT exist
  - Flip Algorithms
  - Divide and Conquer
  - Sweephull (Used by SciPy implementation from Qhull library)
  - Incremental (Explained Next)

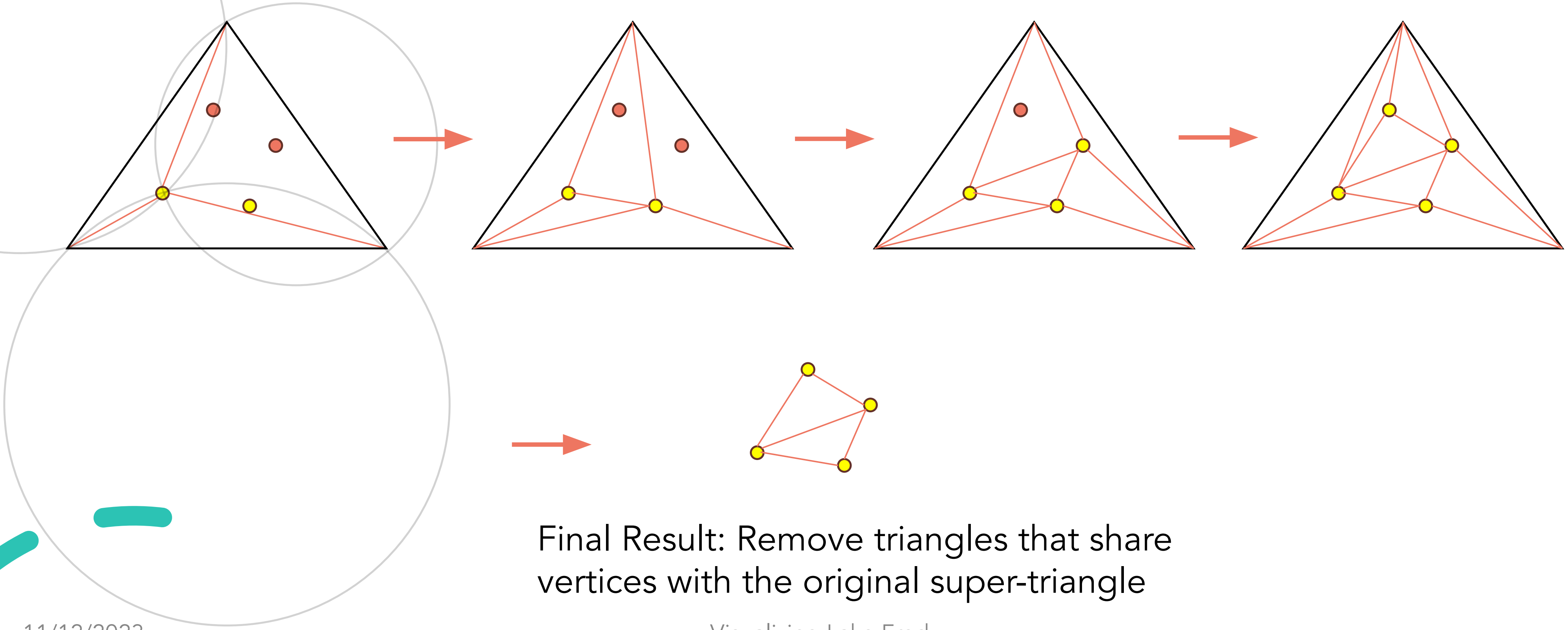
Incremental DT : Bowyer-Watson Algorithm

- Let's look at 4 points
- Create a super-triangle the contains all points
  - Then add one point at a time.
- Check to see if a point lies inside another circumcircle



Bowyer-Watson Algorithm (Cont.)

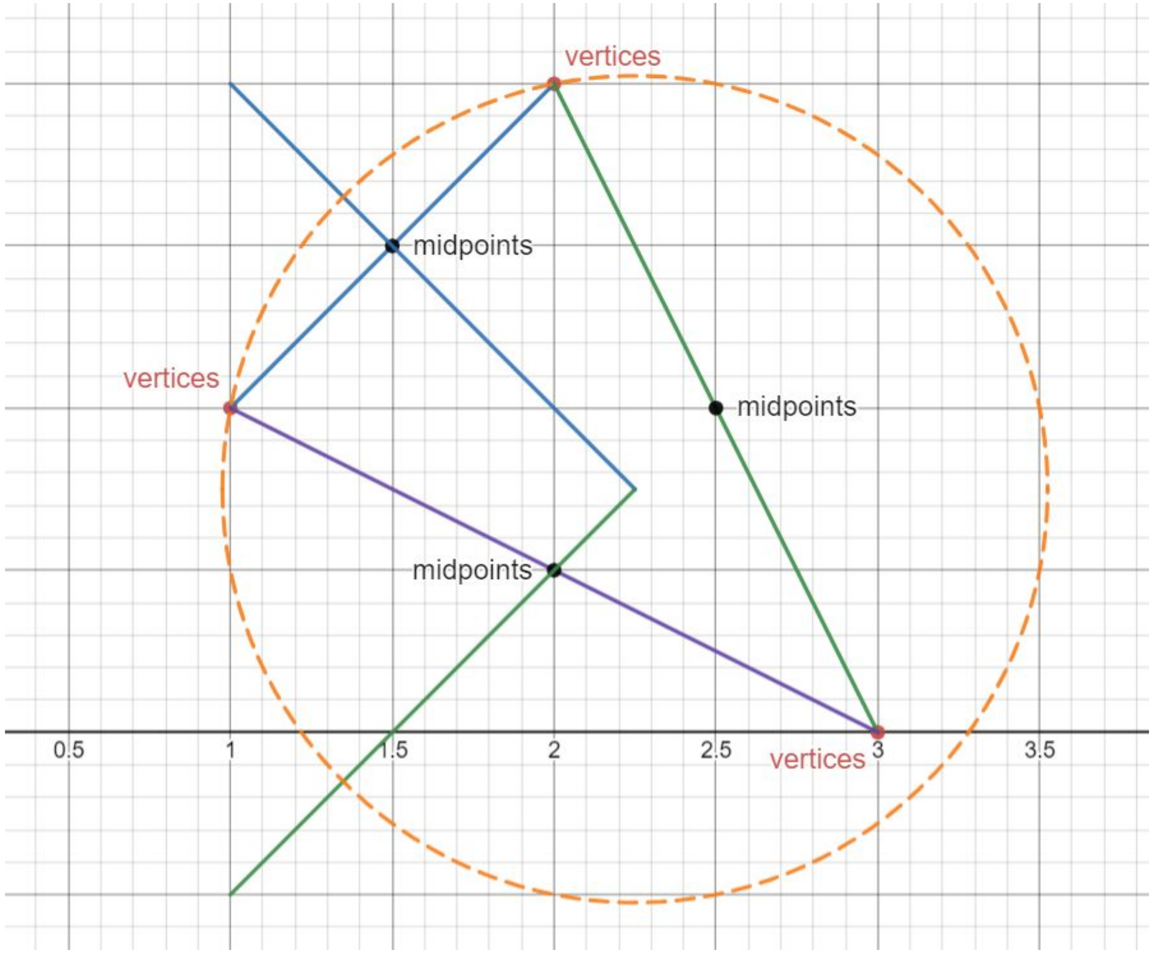
- Continue adding points and checking if points lie inside circumcircles to which they don't belong



Final Result: Remove triangles that share vertices with the original super-triangle

Bowyer-Watson Algorithm (Cont.)

- Now how would you implement this in a computer algorithm?
  - It's obvious how you see which points lie inside a circumcircle but, how do you figure that out algebraically / programmatically?
- There are two approaches to solving this problem.
- First approach: Find the intersection of the lines that are perpendicular to the midpoint of the sides of the triangle.



Bowyer-Watson Algorithm (Cont.)

- Downfalls of the first approach:
  - Edge cases where the slope between two vertices are either zero or undefined.
- **Second approach**: Compute the circumcenter differently.
  - The distance between each vertex and the unknown circumcenter is equal.
- Let the circumcenter be (x, y).
- Distance from point A, B, C to circumcenter be:

$$D_A = \sqrt{(Ax - x)^2 + (Ay - y)^2}$$
$$D_B = \sqrt{(Bx - x)^2 + (By - y)^2}$$
$$D_C = \sqrt{(Cx - x)^2 + (Cy - y)^2}$$

Note: Here Ax, Bx, Cx are all knowns

Bowyer-Watson Algorithm (Cont.)

- $D_A = D_B = D_c$
- Therefore:
  $$\sqrt{(Ax-x)^2+(Ay-y)^2} = \sqrt{(Bx-x)^2+(By-y)^2} = \sqrt{(Cx-x)^2+(Cy-y)^2}$$
- Your two equations:
  - $D_A = D_B$
  - $D_A = D_c$
- Squaring both sides results in:
  - Eq1: $(Ax-x)^2+(Ax-x)^2 = (Bx-x)^2+(Bx-x)^2$
  - Eq2: $(Ax-x)^2+(Ax-x)^2 = (Bx-x)^2+(Bx-x)^2$
- Expand and simplify (skipping some parts):
  - Eq1: $Ax^2 + Ay^2 - Bx^2 - By^2 = 2x(Ax-Bx) + 2y(Ay-By)$
  - Eq2: $Ax^2 + Ay^2 - Cx^2 - Cy^2 = 2x(Ax-Cx) + 2y(Ay-Cy)$

SciPy Implementation of Delaunay Triangulation

- Several Types of Algorithms for computing DT exist
  - Flip Algorithms
  - Divide and Conquer
  - Sweephull (Used by SciPy implementation from Qhull library)
  - Incremental (Explained Previously)
- The SciPy implementation is what was used to create the model.

Creating the Model in Blender

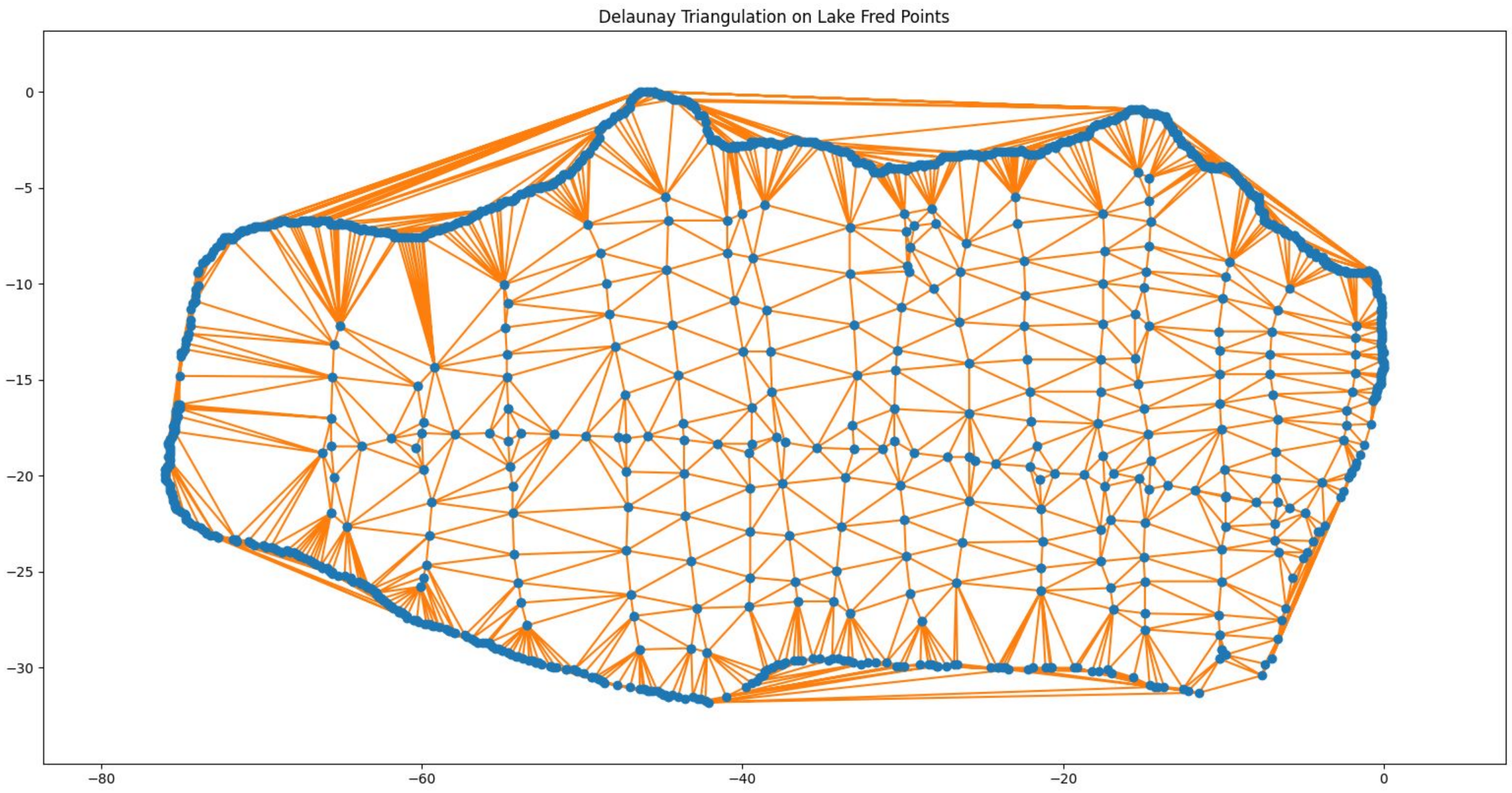
- Blender has an embedded Python interpreter.
- Blender also includes an interface for writing your scripts inside the application
- The Meat and Potatoes of the code:

```python
from bpy import data as D, context as C
from scipy.spatial import Delaunay
import numpy as np
```

```python
    def delaunay_2d(self):
        # Scipy implementation of 2D Delaunay Triangulation using
        triangulation = Delaunay(self.all_points[0:, :2])
        tri = triangulation.simplices
        self.faces = [list(face) for face in tri]
```

Creating the Model in Blender (Cont.)

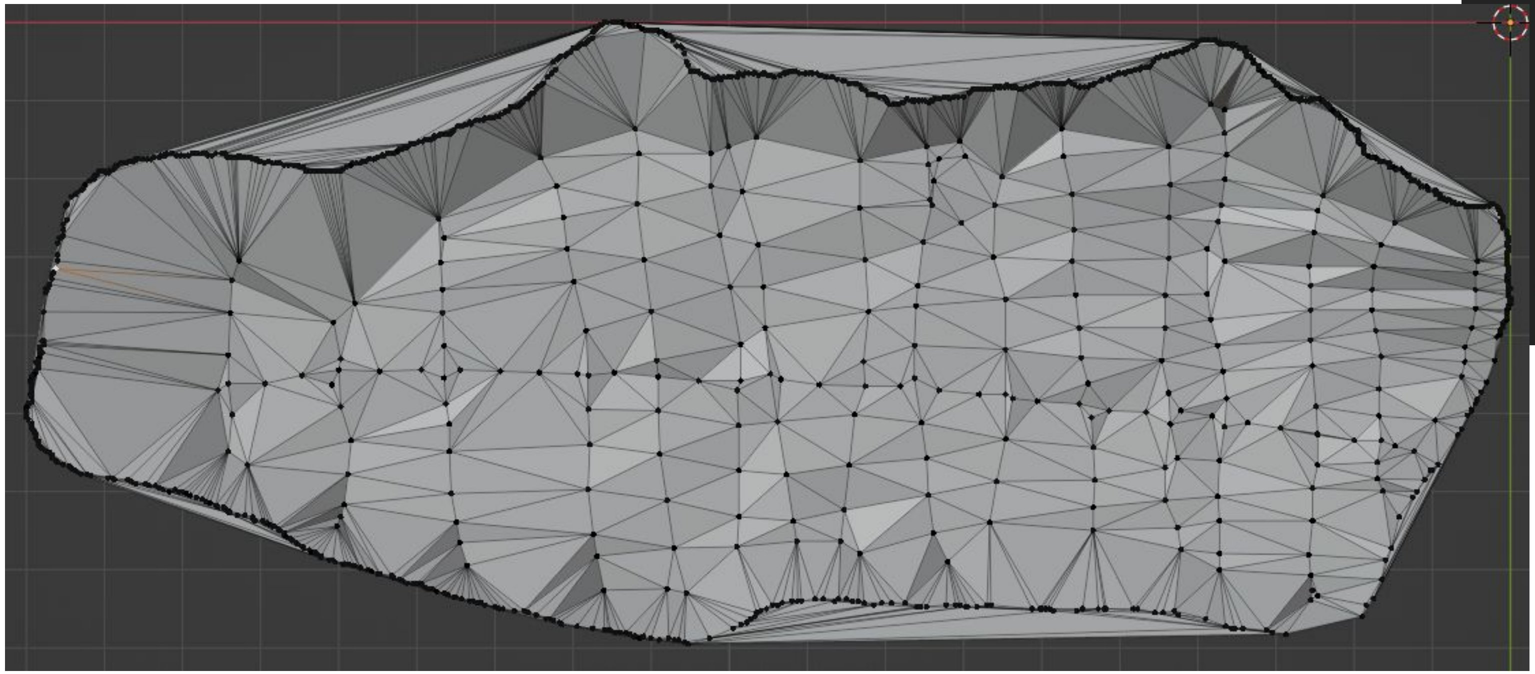
- Visualizing what SciPy is did:



Creating the Model in Blender (Cont.)

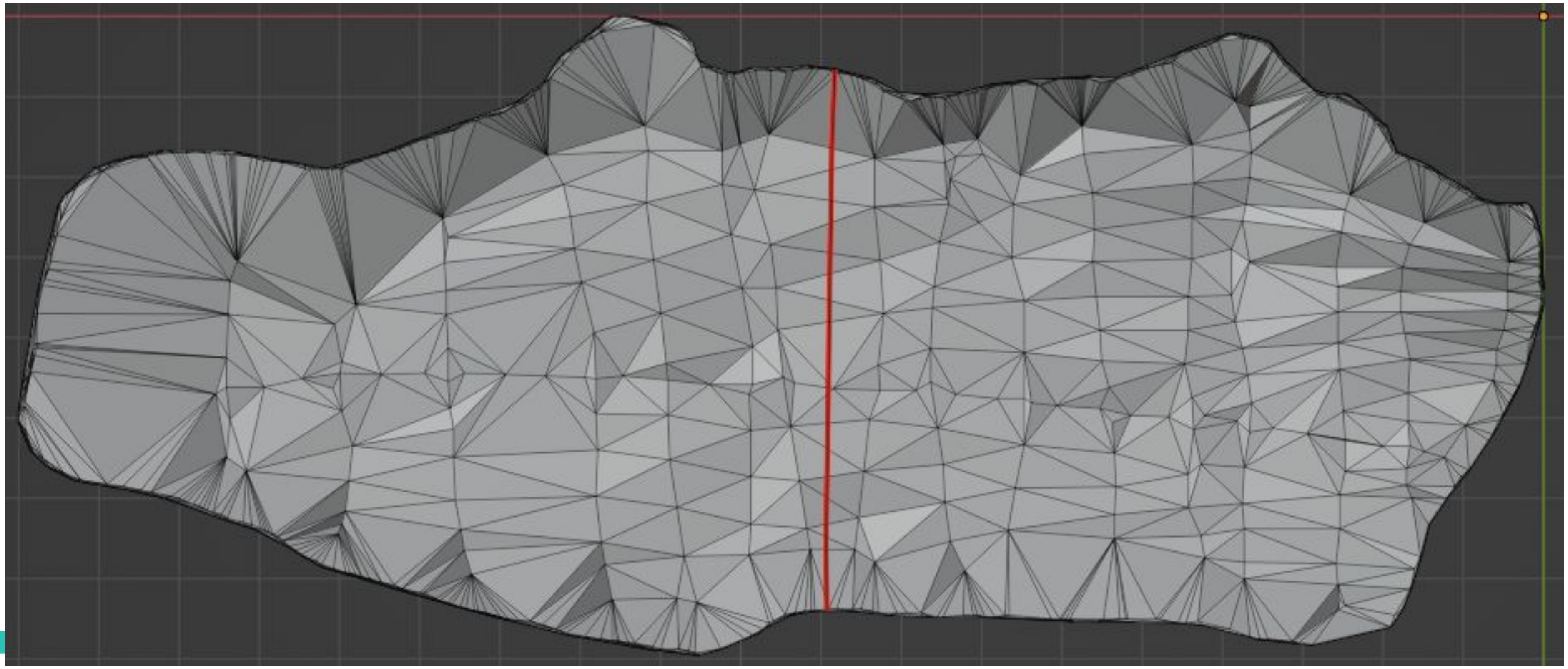
- Raised / lowered each point in Blender while keeping the same edges:

```
def create_mesh(self, data_name, obj_name):
    # Creating the mesh
    if len(self.faces) != 0:
        # Create a new mesh data block
        mesh_data = D.meshes.new(data_name)
        mesh_data.from_pydata(self.all_points, self.edges, self.faces)

        # Create the mesh object and link it to the scene
        mesh_obj = D.objects.new(obj_name, mesh_data)
        C.collection.objects.link(mesh_obj)

        # Update the scene
        C.view_layer.objects.active = mesh_obj
        mesh_obj.select_set(True)

        # Finally, update the mesh to display it
        mesh_data.update()

        return mesh_obj
    else:
        print("Faces Empty")
```



Creating the Model in Blender (Cont.)

- Removed unwanted triangles around perimeter

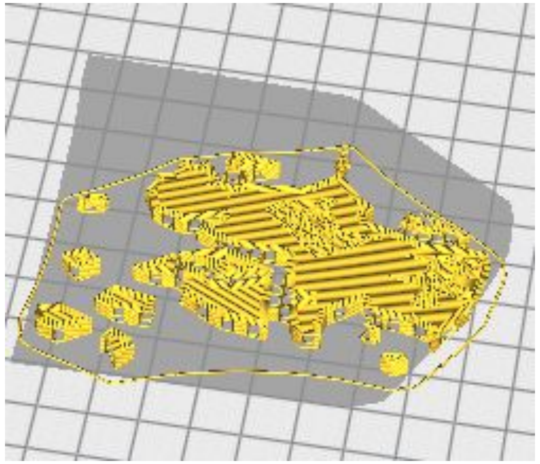


3D Printing

Printing the Model and Associated Issues

Issues With 3D Printing

- Exporting the entire lake model works perfectly fine, great even.
- Issues arise when wanting to create a larger model by splitting the lake.

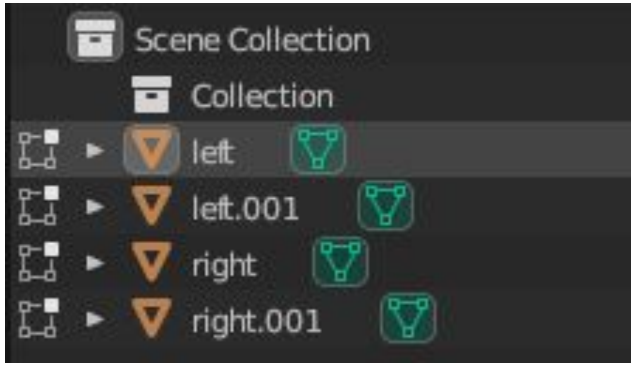


Split results in failed slice

Extruding then exporting caused red error in slicer.

Issues With 3D Printing (Cont.)

- Solution: Extrude all four parts
  - Splitting
    - Extruding the left and right splits in both up and down extrusions
- End result is 4 objects in blender



The blender objects representing 2 left extrusions (up and down) and 2 right extrusions (up and down)

Final 3D Prints

Handing out

Lake Website

View the Lake Model in a browser using Three.js

What is Three.js?

https://collyz.github.io/Personal-Website/
Lake Website

- Three.js is a 3D JavaScript library that tries to make it as easy as possible to put 3D content on the web.
- WebGL is not the same as Three.js.
    - WebGL is a very low-level system that only draws points, lights and triangles.
    - Three.js uses WebGL to do quite a few things but also handles other things such as lights, materials, textures among the many things it can do.

Sources

- Qhull - http://www.qhull.org/
- An implementation of Watson's Algorithm for computing 2-Dimensional Delaunay Triangulations [Sloan, Houlsby] https://www.newcastle.edu.au/__data/assets/pdf_file/0018/22482/07_An-implementation-of-Watsons-algorithm-for-computing-two-dimensional-Delaunay-triangulations.pdf

Thank you

Mohammed Mowla
mowlam1@go.stockton.edu

Interactive Lake Fred

Data Processing, 3D Print and Blender Files