NAME

About The Project

This project is about the love between the people, to record the history in a such small object and

Description

Getting Started

Algorithms

It's worth noting that this algorithm is just a simple example to demonstrate the idea and it won't be the final version. In future versions, I will try to implement more complex algorithms to increase information density and endurance.

Step demonstration

1. Convert file to hash with **shake_256**

```
with open(file_path + file_af, 'rb') as f:
    hash_obj = hashlib.shake_256()
    while True:
        chunk = f.read(4096)
        if not chunk:
            break
        hash_obj.update(chunk)  # Update hash chunk
hash_value_1 = hash_obj.digest(512) #hash with 512 length
```

2. Add file size in front of hash

```
sizebinary = '{0:b}'.format(os.path.getsize(file_path + file_af))

binary_hash_value = ''.join(format(byte, '08b') for byte in hash_value_1)

binary_hash_value = str(sizebinary) + str(binary_hash_value)
```

Convert file size in to binary and add in front of the hash value.

3. Cut the hash value in to "blcok"

```
def cut_len(list, len):
    output = []
    runtime_list = []
    counter = 0
    for i in list:
        runtime_list.append(i)
        counter += 1
        if counter == len:
            counter = 0
            output.append(runtime_list)
            runtime_list = []

    output.append(runtime_list)
    return output
```

Example

```
string = 1010100100001010001111100011 #hash value
len = 4 #cut length
output = [1010,1001,0000,1010,0011,1110,0011] #output format
```

4. Convert binay to int

```
def bin_int(list):
    output = []
    for i in list:
        i.reverse()
        runtime_int = 0
        runtime_mm_counter = 1
        for n in i:
            runtime_int += int(n) * runtime_mm_counter
            runtime_mm_counter *= 2
        output.append(runtime_int)
    return output
```

Example

```
list = [1010,1001,0000,1010,0011,1110,0011]
output = [10,9,0,10,3,14,3] #output format
```

5. Convert list into point Cordinates

```
def Paterrn_Cull_methed(list):
    runtime_counter = 0
    xCor = []
    yCor = []
    bool = False
    for i in list:
        if bool == False:
            xCor.append(i)
            bool = True
        else:
            yCor.append(i)
            bool = False
    if len(xCor) > len(yCor):
        yCor.append(xCor[-1])
    output = [xCor,yCor]
    return output
```

This code split the elements of the input list into two lists, one containing the elements at odd positions(X Cordinate) in the input list and the other containing the elements at even positions(Y Cordinate) in the input list.
If the length of the Y coordinate is not equal to the length of the X coordinate, a new element equal to the last value of X will be added to the Y coordinate.

Example

```
list = [10,9,0,10,3,14,3]
output = [[10,0,3,3],[9.10,14,3]] #output format
```

6. Output Cordinate to Grasshopper

```
def outputCorOne(list,lit):
    with open(file_path + '.txt','w') as Cor:
        counter = 0
        while counter < len(list[0]):
            Cor.writelines(str(list[0][counter]) + ',' + str(counter * lit) + ',' + str(li
            counter += 1
    Cor.close()
```

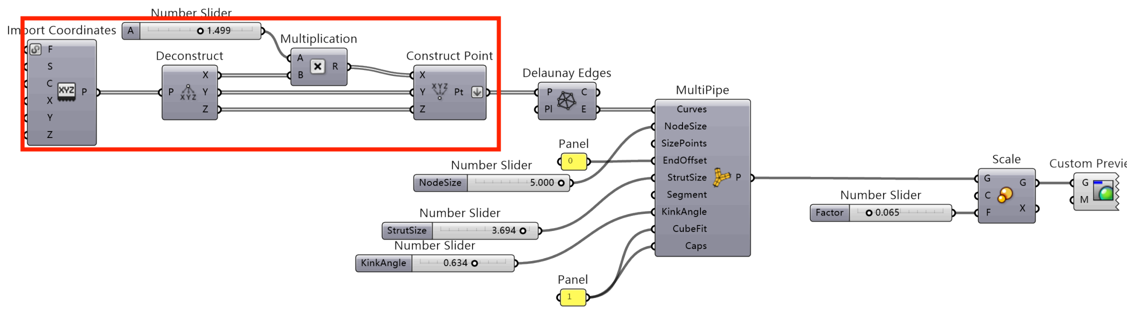
Output Cordinate to grasshopper and add Z coordinates witch is monotone increasing.

Example

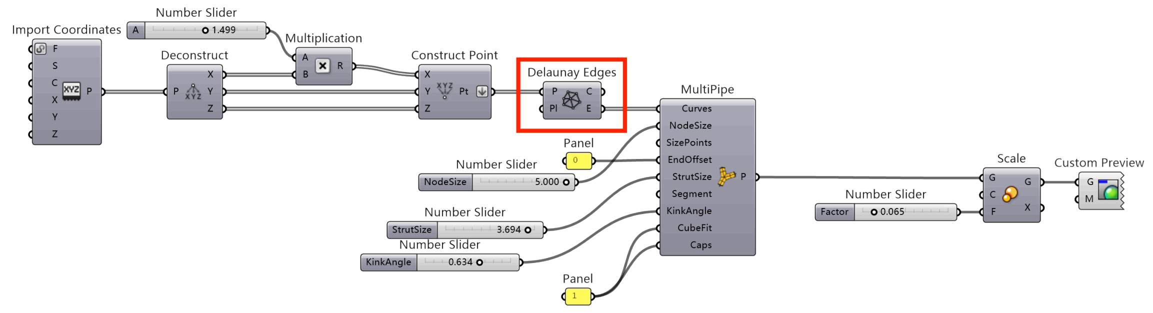
```
list = [[10,0,3,3],[9.10,14,3]]
```

```
10,9,0
0,10,1
3,14,2
3,3,3
```

7. Plot out the Point(Adjust the scale)



8. Delaunay_Edges



A 1.499

Panel 0

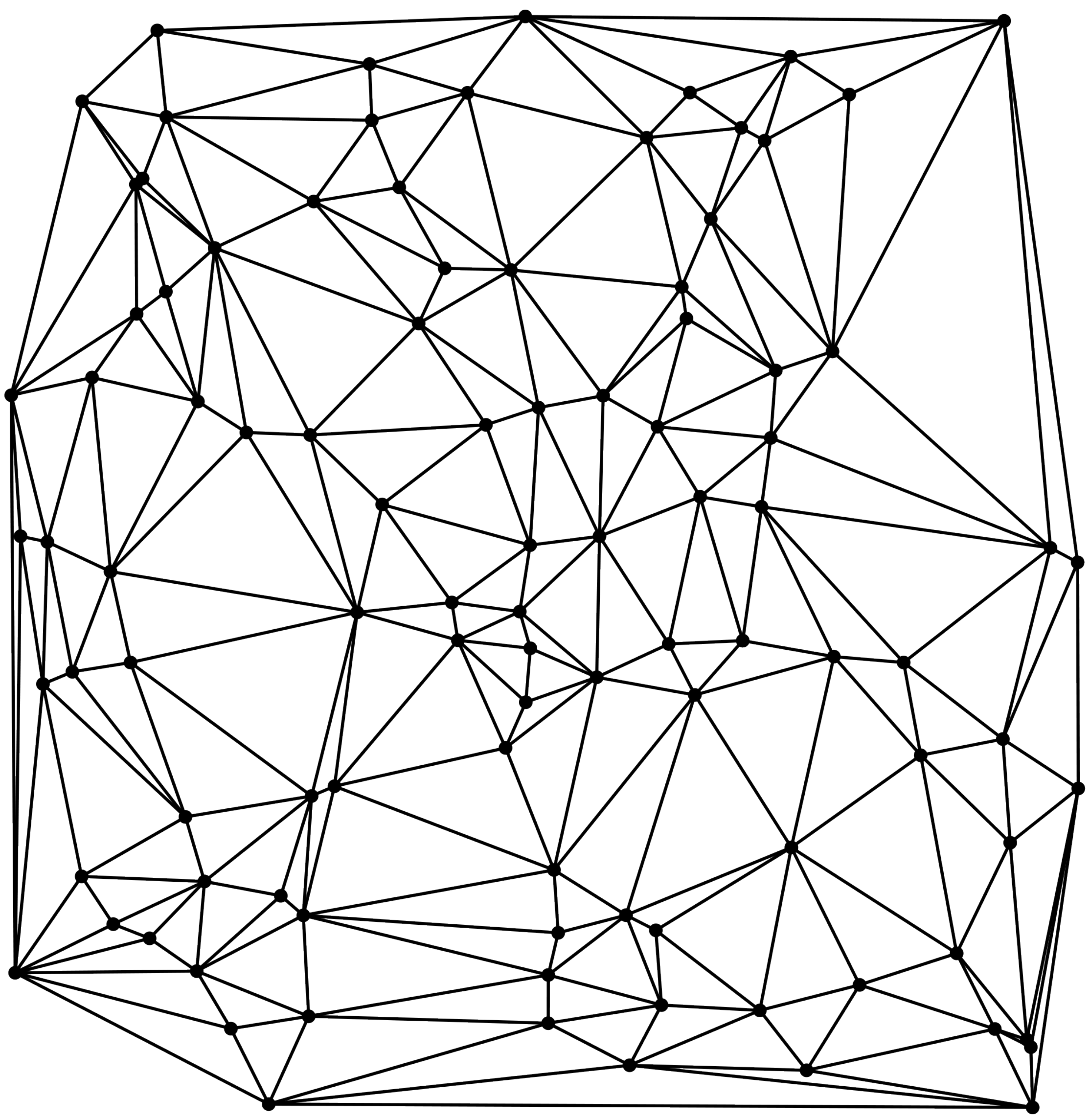
NodeSize 5.000

StrutSize 3.694

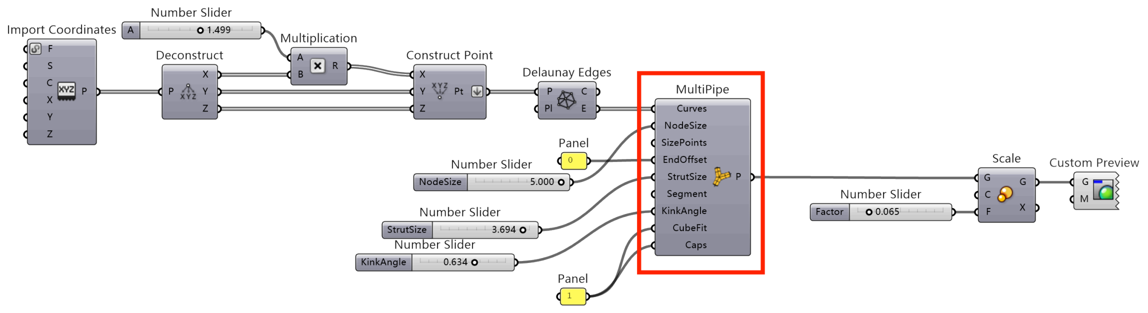
KinkAngle 0.634

Panel 1

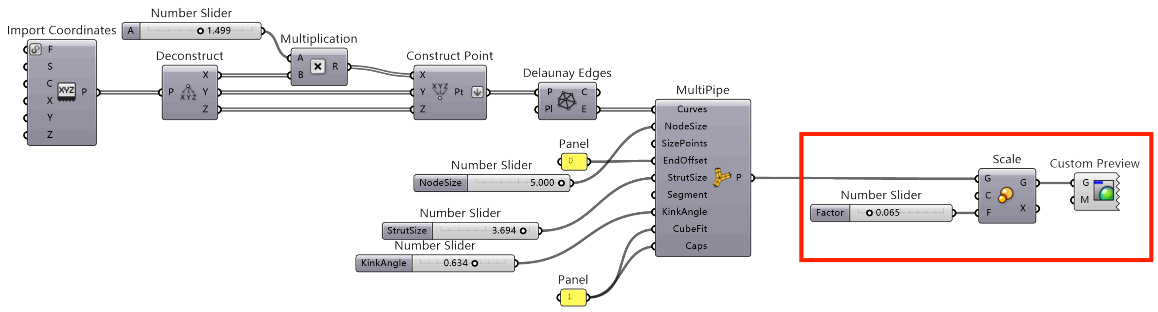
Factor 0.065



9. Use MultiPipe



10. Scale and output



TODO

1. Transfer to Blender rather than Rhino and Grasshopper
2. Use more complex algorithms to generate the model with higher information density and endurance.

Q&A

1. Why use HASH?

- Due to the ability of a hash to represent a complex file with a simple string, it is an effective way to reduce file size and make it possible to record information in a small area. Although there is a 100% chance of hash collisions occurring when implementing larger files, recording the file size in the model can substantially reduce the number of collisions. Additionally, we know that the files we store have some meaning, such as a video, photo or text. By adding this as a condition, I would say it is impossible for collisions to disturb the original information.

1.
2.