Exercise 2

Visualize ComplexCity
Chair of Information Architecture
ETH Zürich

March 1, 2013

1 Slides

Go through the slides of the presentation.

a) Run the code snippets of the following slides and *print* all corresponding outputs: Slides 7, 8, 11

b) Construct a convex hull, a Voronoi diagram and a shortest path in Blender.

2 Programming

In this exercise, we will program Bubblesort. We will use the sorting algorithm to sort the pixels of a picture.

a) Program a function *bubble* which sorts a list. As hint us the following pseudocode:

```
# bubblesort, sorting a list liste
def bubblesort(liste):
  while somethingChanged:
    for i in range(0, len(liste)-1):
      if (i-th element > (i+1)-th element):
        swap the i-th element with the (i+1)-th
        element of liste
  return liste
```

Test your program with a list of numbers, e.g.

```
liste = [99,4,42,66,265,222,1]
print(liste)
newListe = bubblesort(liste)
print(newListe)
```

b) Now we are going to adapt our *bubblesort* function, so that it can sort pixels from a picture according to their brightness. Since a picture has three values per image, we need to change the *if* statement in the code of exercise a. Instead of comparing only numbers, we compare the sum of the RGB values.

```
# bubblesortPicture, sorting a list liste
def bubblesortPicture(liste):
  while somethingChanged:
    for i in range(0, len(liste)-1):
          i-th RGBsum = liste[i][0] +
            liste[i][1] + liste[i][2]
          (i+1)-th RGBsum = liste[i+1][0] +
            liste[i+1][1] + liste[i+1][2]
      if (i-th RGBsum > (i+1)-th RGBsum):
        swap the i-th element with the (i+1)-th
        element of liste
  return liste
```

With the function adapted, we can now load a **small** picture and sort it with the algorithm.

```
# read in a picture and get the pixels as a list.
import Image
pic = Image.open("someSmallImage.png")
a = list(pic.getdata())

# Apply bubblesortPicture to the list.
newA = bubblesortPicture(a)

# Set the pixels in the picture to the sorted ones.
pic.putdata(newA)

# Show the picture.
pic.show()
```

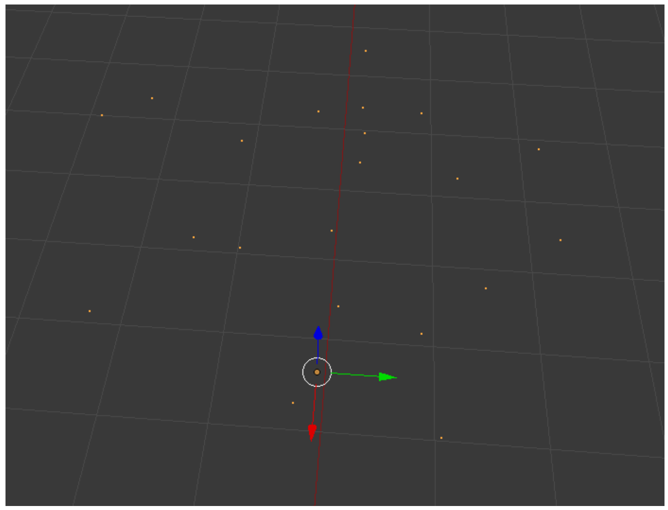
c) To see what time difference a good algorithm can have to naive implementations, compare the following with the time your sorting algorithm is running. This uses the sorting algorithm which is integrated into Python:

```
# read in a picture and get the pixels as a list.
import Image
pic2 = Image.open("someSmallImage.png")
a = list(pic2.getdata())

# Apply the Python internal sorting function to
# the list.
newA = sorted(a, key=lambda data:data[0] +
  data[1] + data[2])

# Set the pixels in the picture to the sorted ones.
pic2.putdata(a)

# Show the picture.
pic2.show()
```

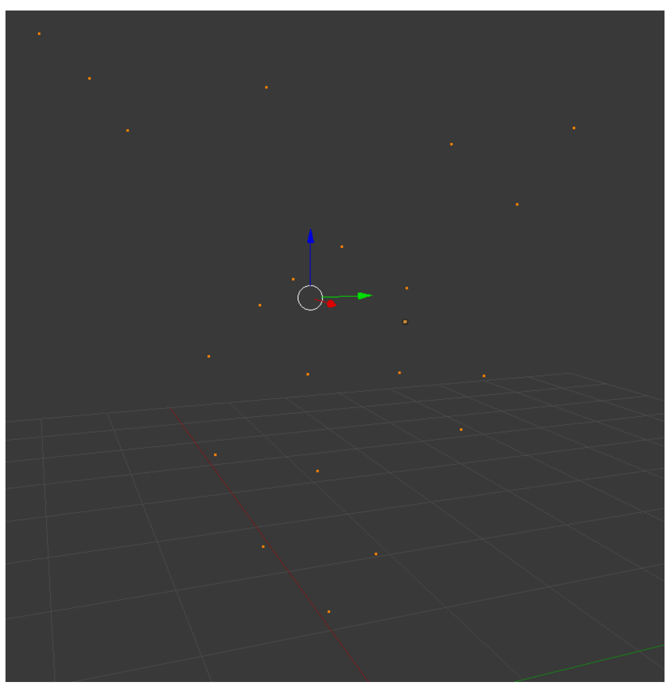
3 Blender

We will construct a Delaunay triangulation using Blender.

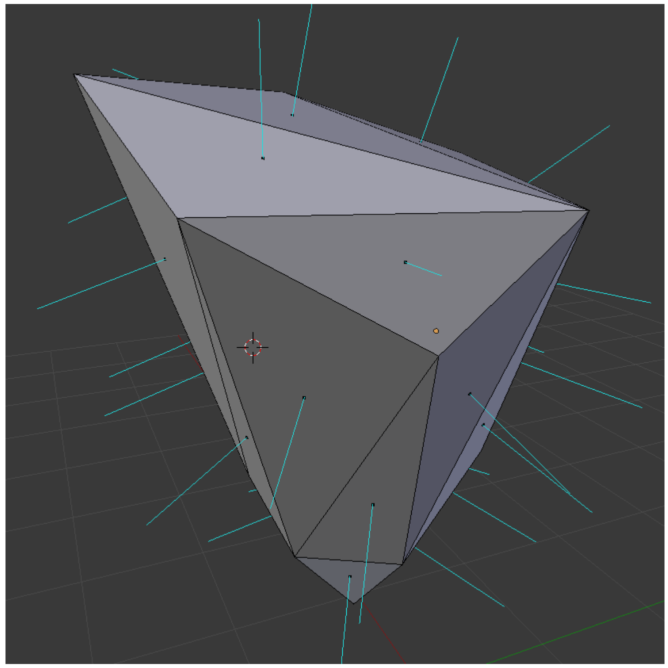
a) Add some points into an empty scene. The points should all be on the x-y-plane.



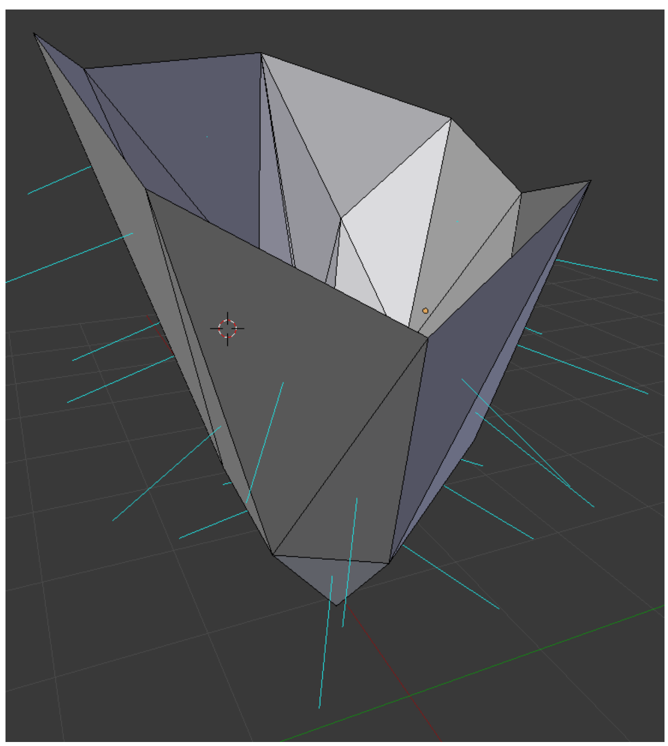
b) Use *Proportianal Editing* to project the points on a convex hull as depicted in slide 18.



c) Construct the convex hull around the projected points.



d) Delete all the faces and corresponding edges, where the normal of the face points upwards.



e) When you now project the points back into a plain (set scaling in z-direction to zero) and change the viewport shading to wireframe, you can see the delaunay triangulation we just constructed.