Visualize ComplexCities

Programming Paradigms & Algorithms

Chair of Information Architecture
ETH Zürich

March 1, 2013

OO vs. Procedural

Algorithms

Recursion

Convex Hull

Voronoi

Shortest Path

Evolutionary Algorithms

Object-Oriented vs. Procedural Programming

There exist two main programming paradigms, Object-oriented programming and procedural programming.

Object-oriented: ”(The focus) [. . .] is to break down a programming task into objects that expose behavior (methods) and data (members or attributes) using interfaces.”

Procedural: ”The focus [. . .] is to break down a programming task into a collection of variables, data structures, and subroutines [. . .]”

Source: `http://en.wikipedia.org/wiki/Comparison_of_programming_paradigms`

Procedural Programming

In procedural programming, we divide the program code into different function, variables and data structures. It is, what we were doing in the last lecture.
The functions were designed to work on the variables.

Object-Oriented Programming (OO)

- In object oriented programming, we bundle the variables and functions in different objects. The objects work than on their own data-structures.
- Very important programming paradigm for Blender, since all the properties of blender objects can be accessed through python classes.

OO continued

A class is definition starts with the keyword *class*, followed by the class name.

- The content of the class has to be intended.
- The class normally has a constructor function, with which the initial values can be set.
- The variables of the class can be accessed within the class by using the *self* keyword and then the variable name.

```
class className:
    def __init__(self, initParameters):
        statements

    def classFunctionName1( parameters ):
        statement(s)
```

Class Human

```
class Human:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def name():
        return self.name

    def hasBirthday():
        self.age += 1

    def age():
        return self.age
```

Using an Object

When we have coded the class, we can use it.

```
# create a human.
arti = Human("Arthur", 42)
# Get the age.
currAge = arti.age()
# Increase the age.
arti.hasBirthday()
# Check the age again.
newAge = arti.age()
```

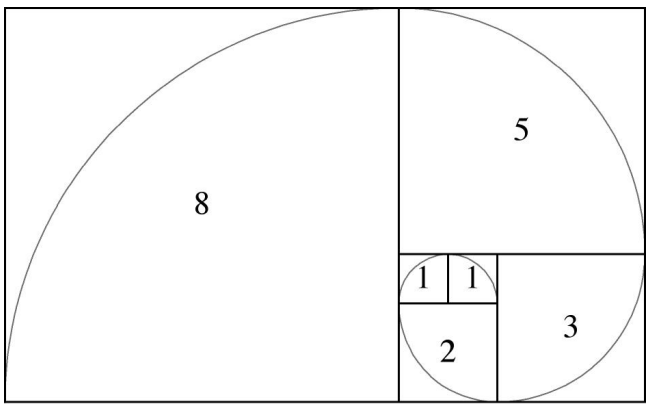
Algorithms

Definition from Wikipedia:

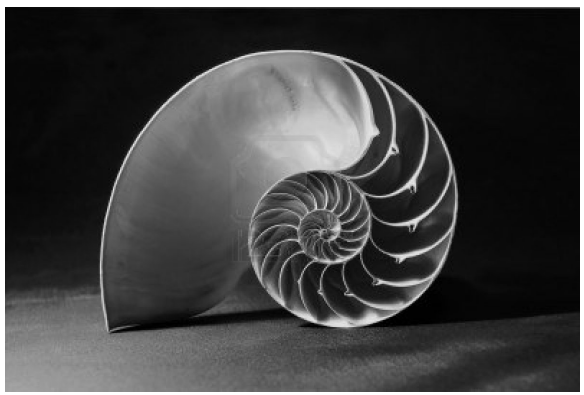
”[. . .] an algorithm is an effective method expressed as a finite list of well-defined instructions for calculating a function.”

Fibonacci Series

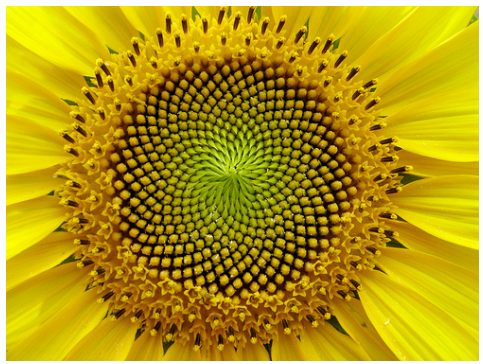
Fibonacci appears in many biological settings. For example in shells or sunflowers:



http://mathyear2013.blogspot.ch/



http://www.loremipsum.at/



http://de.123rf.com/

Recursion

What is the output of the following code?

```
def fib(n):
    if (n == 0):
        return 1
    elif (n == 1):
        return 1
    else:
        return fib(n - 1) + fib(n - 2)

print(fib(5))
```

Sorting

Speed is very important when solving problems. This can be seen, when we for example want to sort a list of things.

Bubblesort: Simplest sorting algorithm, very inefficient. Uses SImple pairwise comparisons to sort the list.

Quicksort: Most famous sorting algorithm, very efficient. Divide and conquer approach to sorting.

For a speed comparison of these algorithms check `http://www.youtube.com/watch?v=aXXWXz5rF64`.

Convex Hull

A convex hull is a convex envelope around a set of points. The algorithm does two times the same in 2D. First it calculates the upper part of the convex hull and then the bottom part.

Convex Hull ctd.

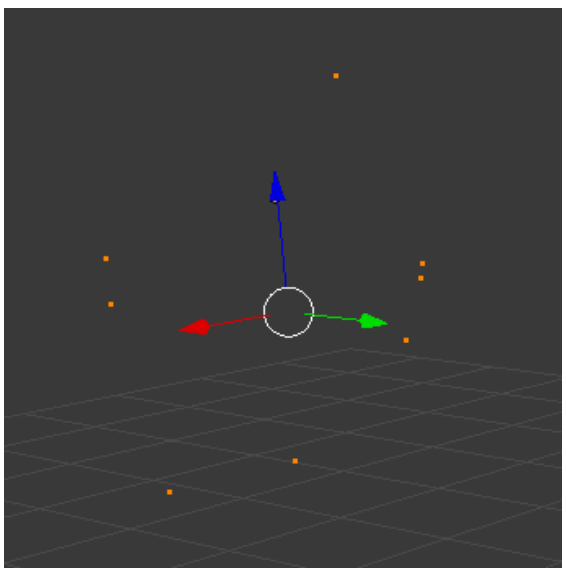
The algorithm for the upper half of the convex hull:

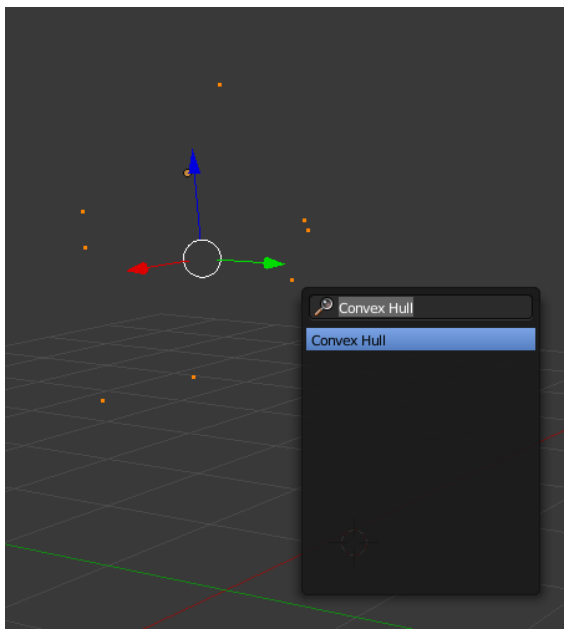
1. Sort the points according to the x-axis.
2. Choose the most left point.
3. Connect the last used point with the most left one not used.
4. If the last two connections form a left curve, delete the second oldest point of the connections. Go to 4.
5. If the current point is the most right one exit, otherwise go to 3

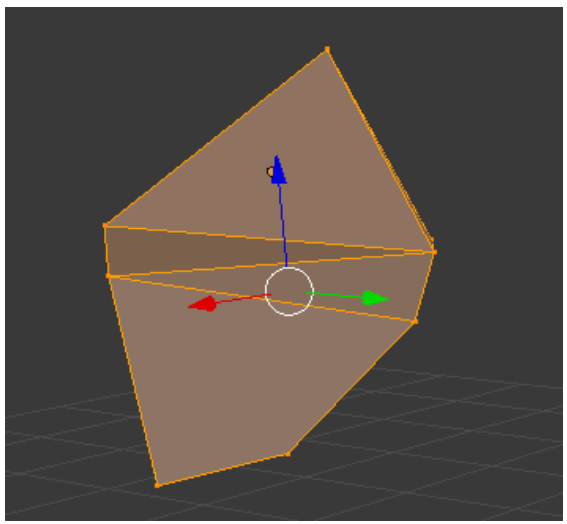
`http://www.youtube.com/watch?v=UlT6R0rz0i4`

Convex Hull in Blender

In edit mode, select the desired vertices, hit enter and choose *Convex Hull*.

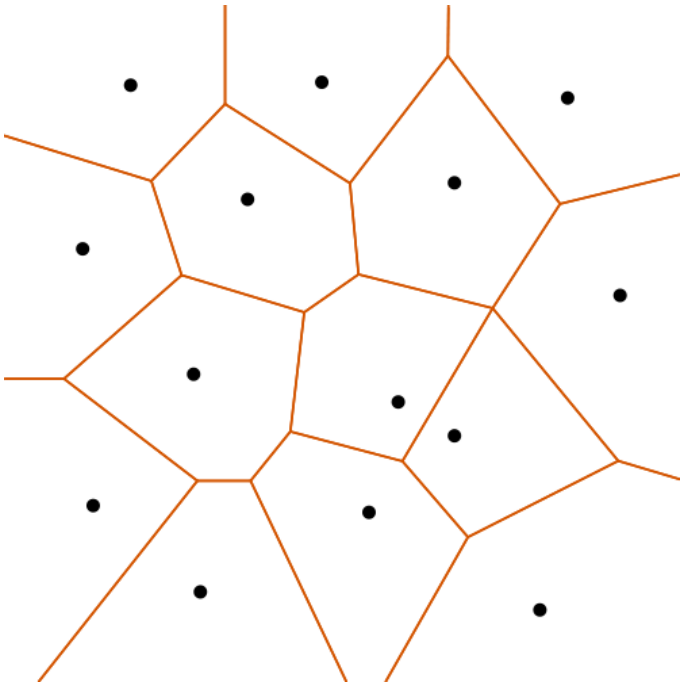






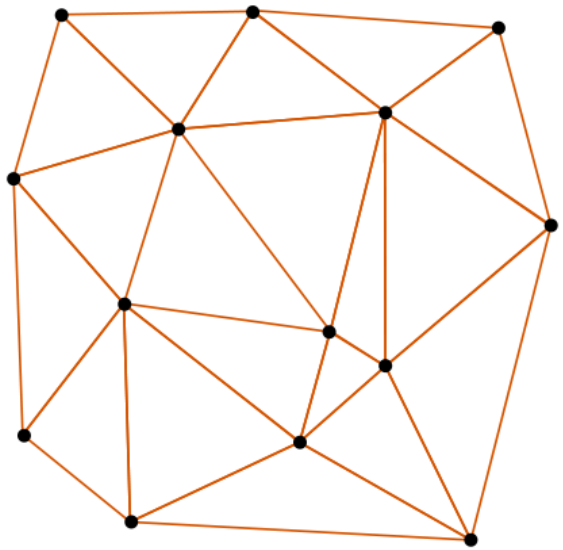
Voronoi & Delaunay Diagrams

Voronoi diagramm



Picture source: `http://www.wikipedia.org/`

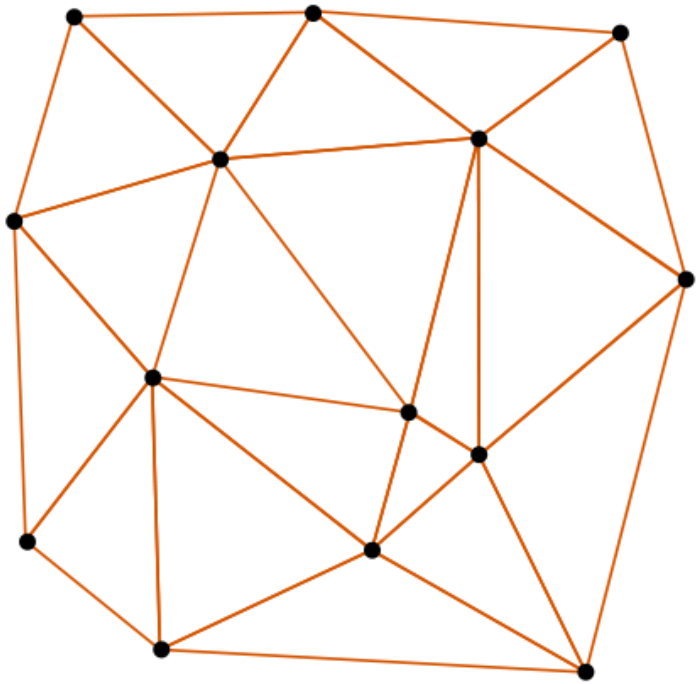
Delaunay triangulation

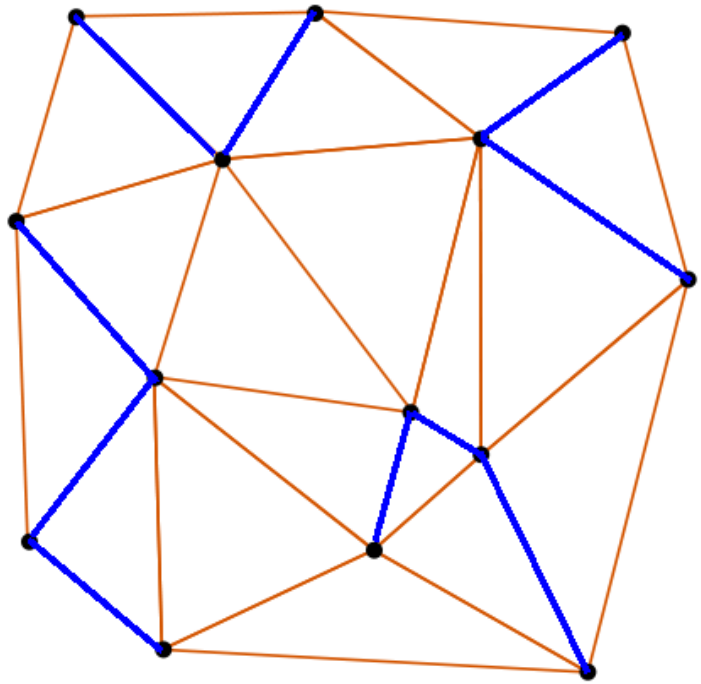


Delaunay Triangulation

The Delaunay triangulation maximizes the minimum angle of all the triangles. This means it tries to avoid narrow triangles. This gives a nice set of polygons we can use in visualization and rendering. It is also possible to get the nearest neighbors from Delaunay triangulation.

Delaunay & Nearest Neighbors

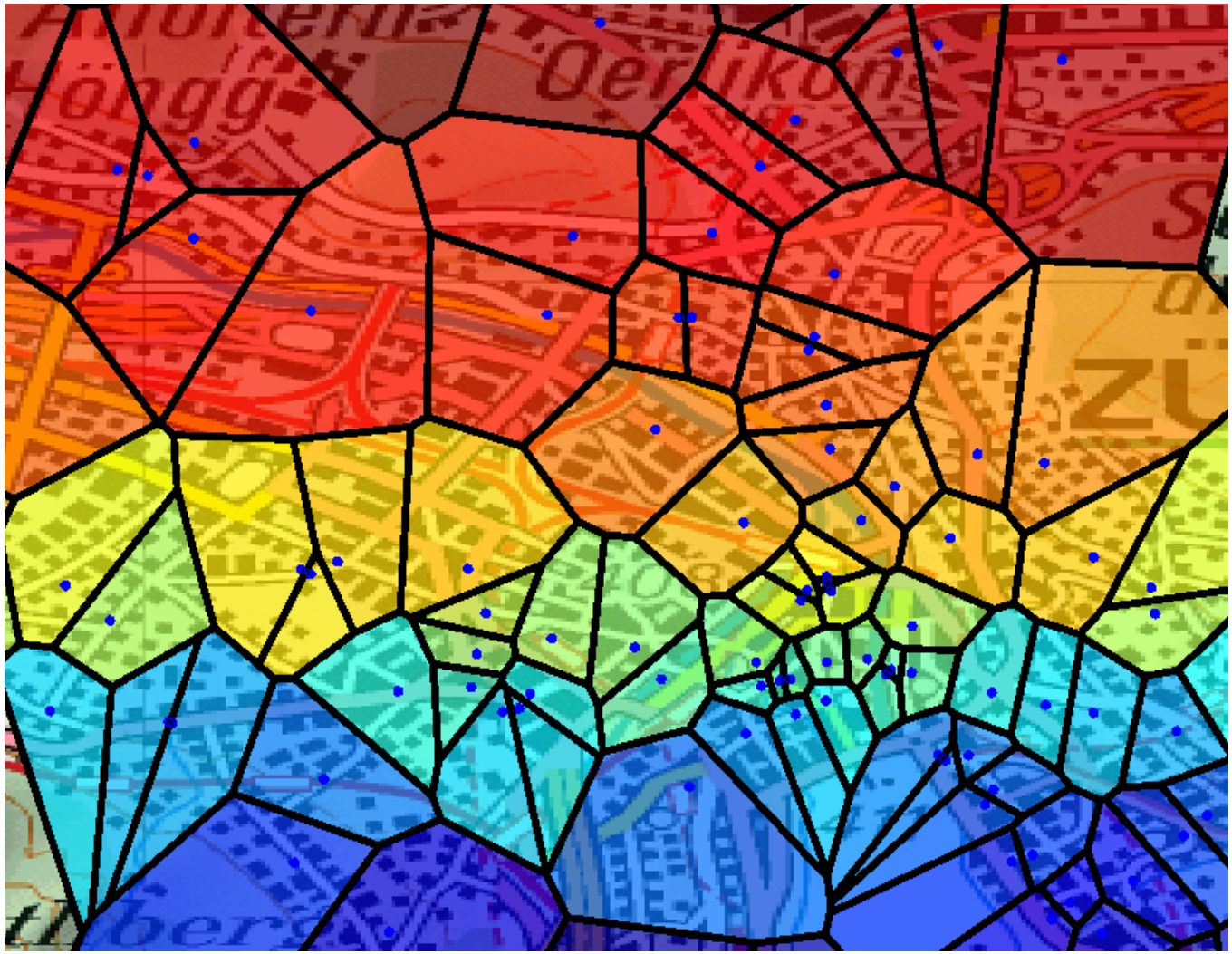




Voronoi Diagram

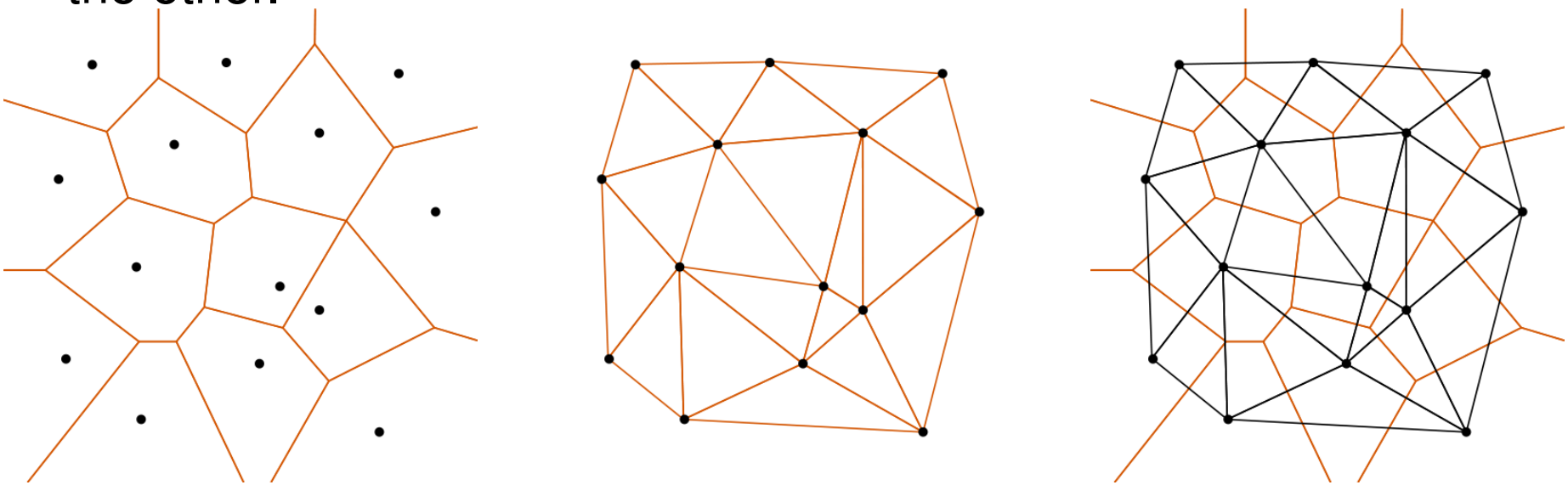
The Voronoi diagram shows the area of each point, where it is the nearest to. This can be used to show for example the catchment area of stores in a city.

Voronoi Diagram & Flower Shops



Connection between Voronoi & Delaunay

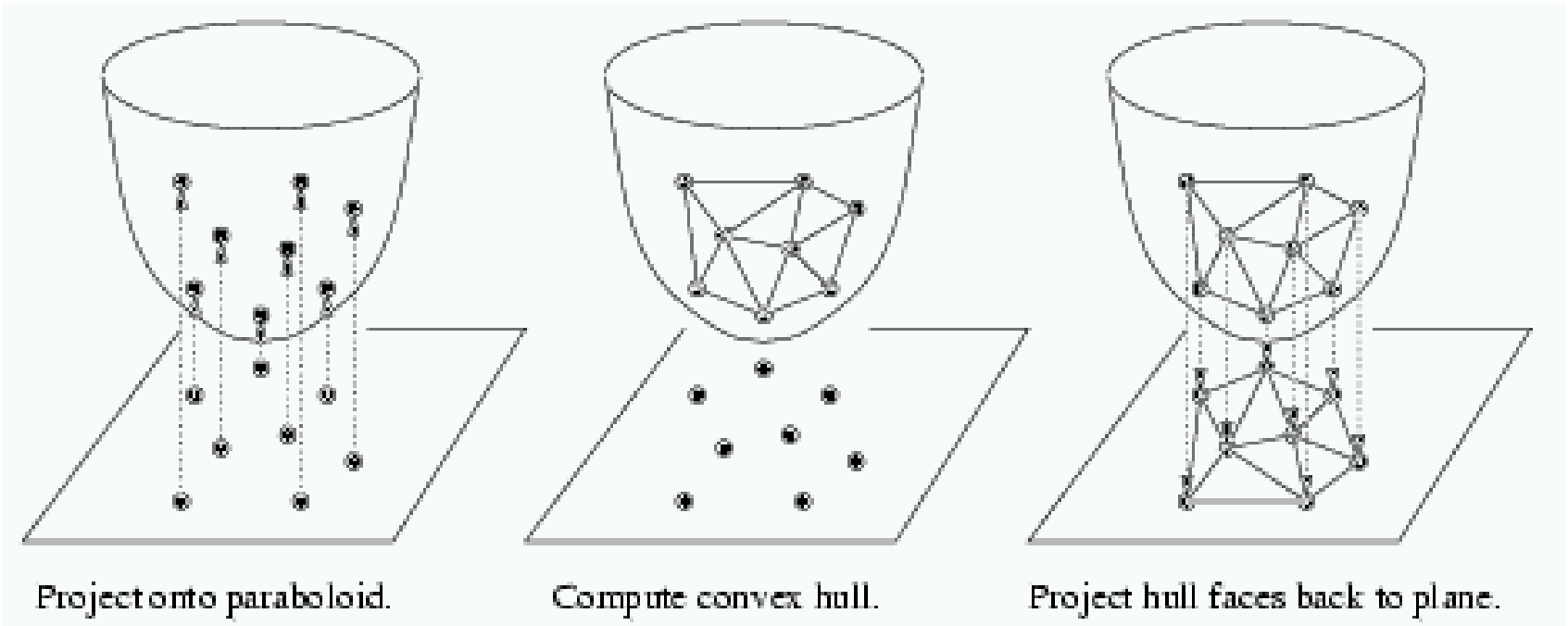
The two are dual. This means that one can be constructed from the other.



Picture source: `http://www.wikipedia.org/`

Construct a Delaunay Triangulation

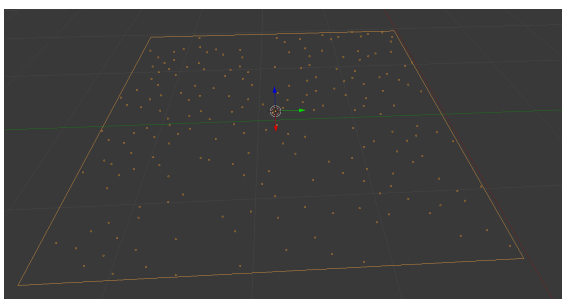
To construct a Voronoi diagram we first construct the Delaunay triangulation, because it is less expensive to construct.

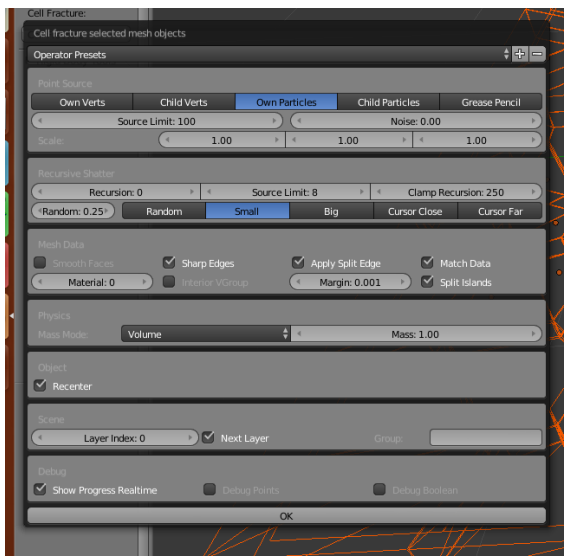


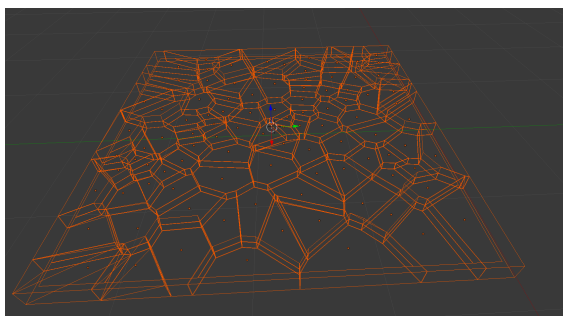
Picture source: `http://www.cs.wustl.edu/~pless/546/lectures/L13.html`

Voronoi in Blender

Enable the cell fracture addon. Then add a plane and emit particles from the plane. Afterwards click *Cell Structure* and then *ok*. The diagram is then in the second layer.







Shortest Path

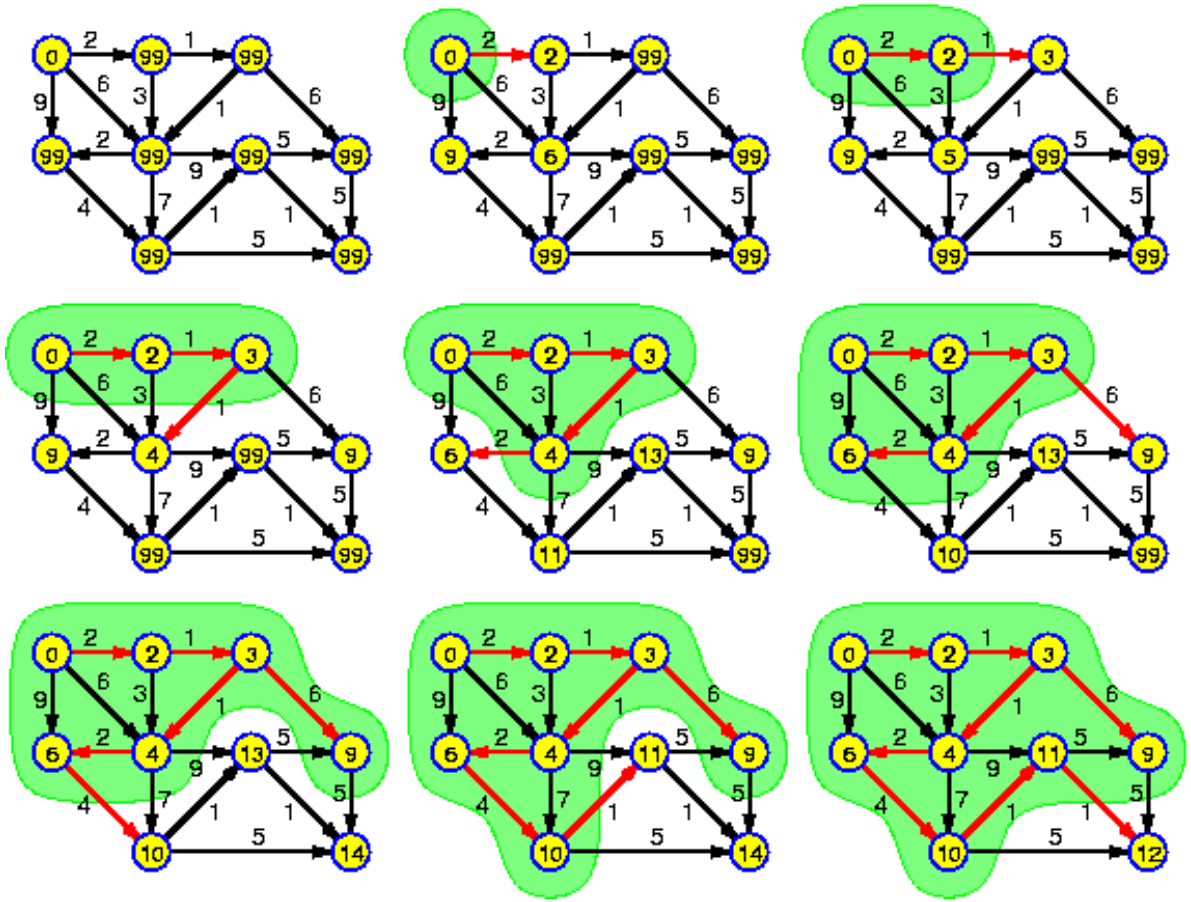
The most famous shortest path algorithm is the Dijkstra algorithm. It only works when all the distances are non-negative and there exists a path from the starting point to the destination.

Dijkstra

1. Label all graphs with infinite distance except for the first one and set all to ”unvisited”. Set the current location to the starting location.
2. Set the distance for all neighbors to the current distance plus the distance from the current location if it is smaller than the distance they already have.
3. Go to the location, which has the shortest distance by now. If it is the destination exit, otherwise set the last location to ”visited” and go to 2.

Dijkstra

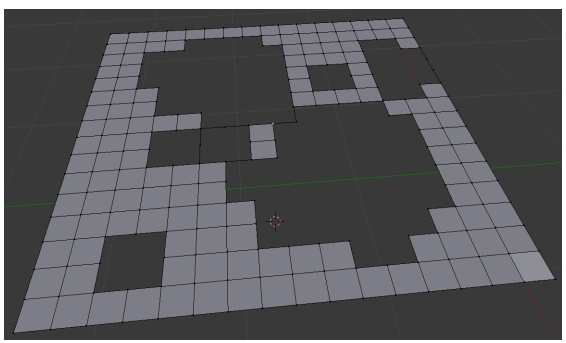
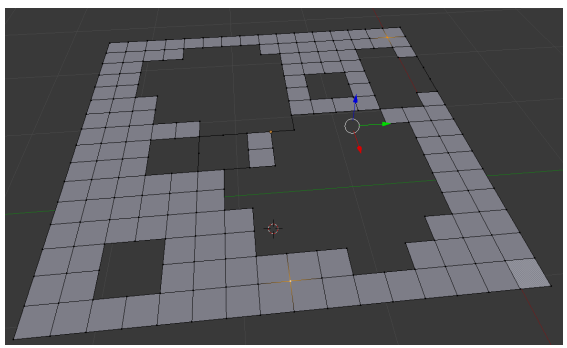
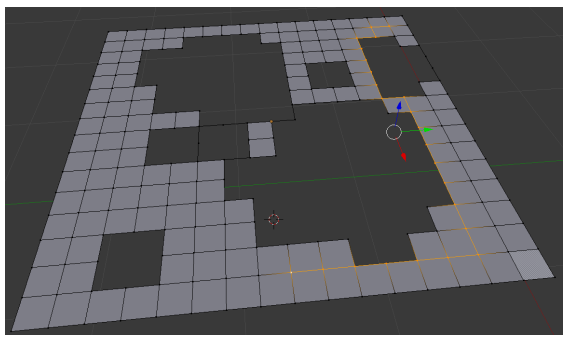
The most famous shortest path algorithm is the Dijkstra algorithm.



Picture source: `http://www.iekucukcay.com/`

Shortest Path in Blender

Add a new plane and make some wholes. In edit mode, select two vertices. Hit *w* and select *Select Vertex Path*.

Evolutionary Algorithms

Evolutionary algorithms are an optimization technique. The idea behind these algorithms comes from biological evolution. Basically, the method varies the current state and looks if an offspring is better then the current state.

Evolutionary Algorithms

One way how evolutionary algorithms work is the following:

1. Generate an random initial population.
2. Generate offspring of the current population by crossover, variation, ...
3. Add the offspring to the population. Delete the members of the population with the worst fitness.
4. If the solutions are not good enough, go to 2.

Evolutionary Algorithms

- An example of an evolutionary algorithm used for architecture can be found in `http://vimeo.com/15697593`.
- The drawback of this methods is, that they converge slow to a solution, i.e. the need a lot of steps to find a good solution.

Helpful Links

- http://docs.python.org/3/
- http://www.python-kurs.eu/klassen.php
- http://www.grasshopper3d.com/profiles/blogs/evolutionary-principles
- http://en.wikipedia.org/wiki/Sorting_algorithm