

Terrains, Triangle Meshes and Heightmaps

February 11th 2009

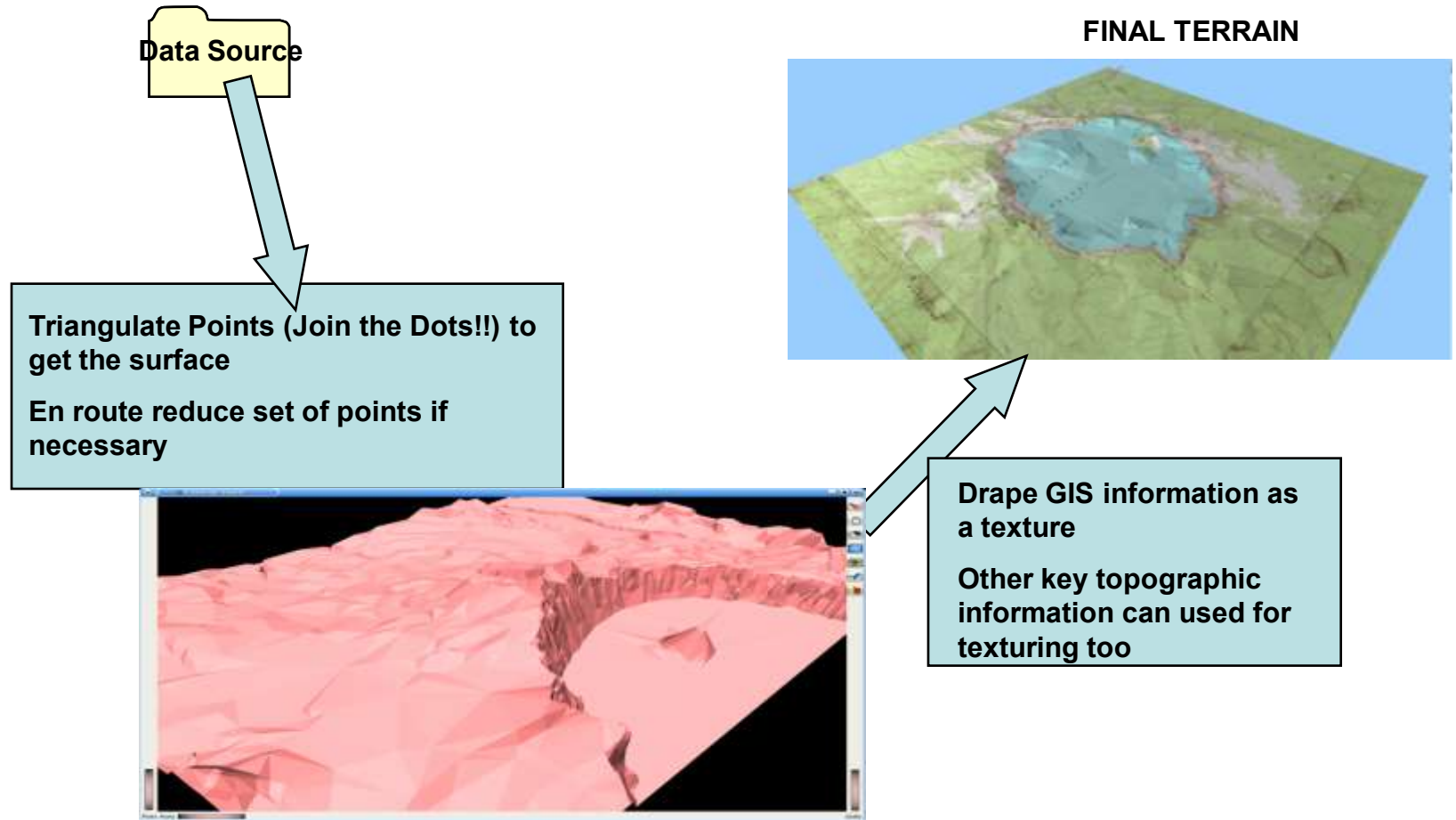
MAE 574, Virtual Reality Applications and
Research

Instructor: Govindarajan Srimathveeravalli

Terrains in VR

- Most VR applications that involves an ‘outdoor’ experience requires some sort of terrain
 - Flight simulators
 - Battlefield simulators
 - Architectural and heritage sites
 - Urban planning and simulation
 - Driving simulators
 - Etc.
 - Non VR applications
 - Meshes form the input to most D.E and P.D.E problem spaces

Steps to Terrain Generation



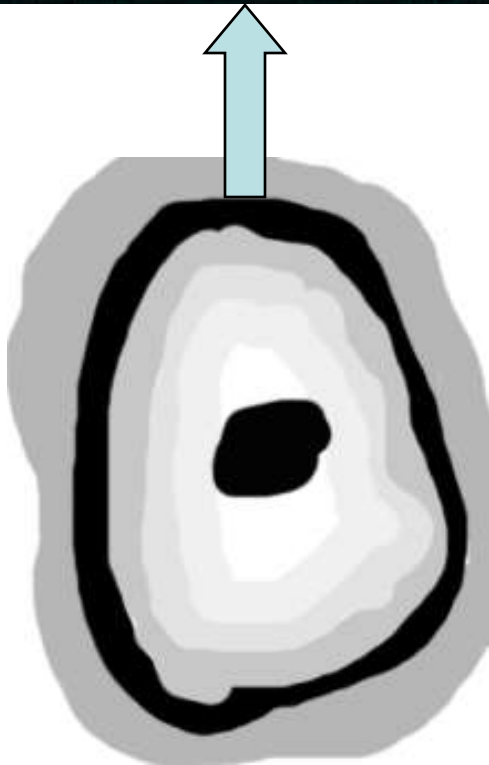
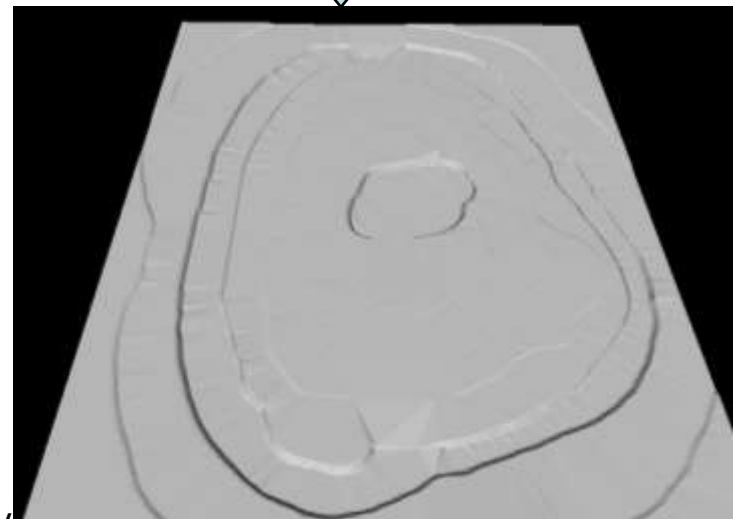
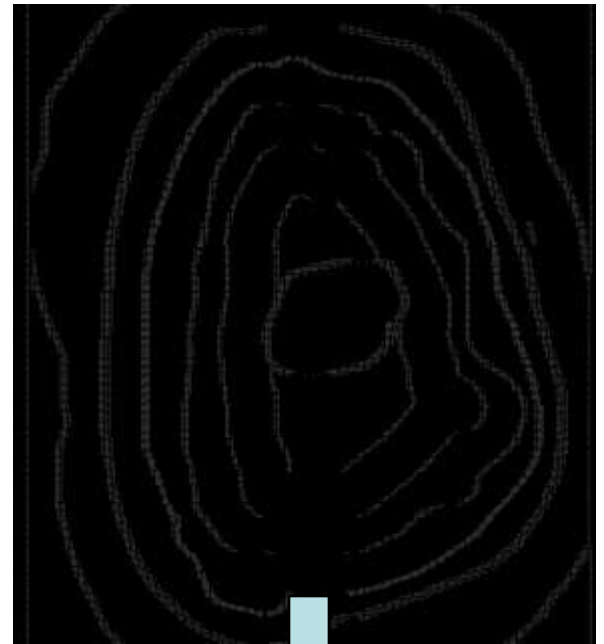
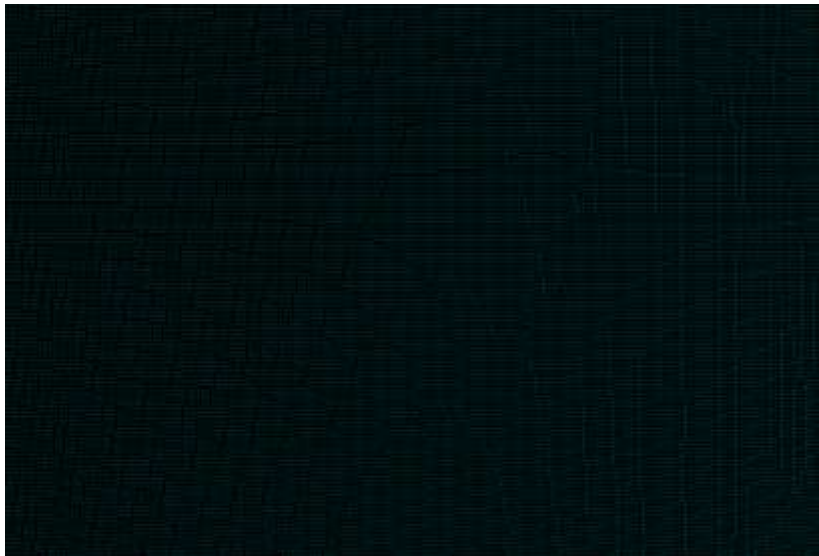
Data Sources


- DEM (Digital Elevation Model)
 - Free raster format
- DLG (Digital Line Graph)
 - Sometimes free vector format
- NED (National Elevation Data)
 - More recent format, available free for most of USA
- SRTM (Shuttle Radar Topography Map)
 - Free
- TIN (Triangulated Irregular Network)
 - Post triangulated mesh information, not a very common format
- Greyscale
 - Images encoding height information into the gray/intensity scale
 - All maps are essentially square/rectangular and have height encoded onto a scale of 0-255
- Etc.

Greyscale Images

- Consider TGA Images
- Commonly used format,
 - We already have a parser for it !
- Has a header and data as a single array, binary format.
- Handles 8,24 or 32 bit data.
 - Greyscale is 8 bit or 1 byte of color information
- Non greyscale data can be converted to it using a simple relation

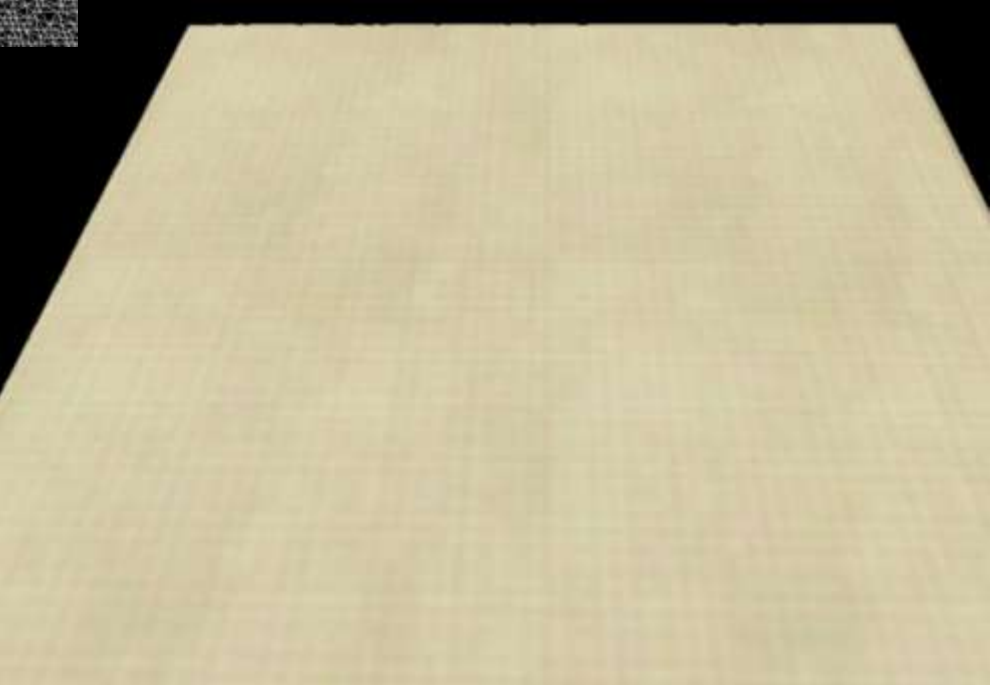
$$Gscl = 0.3 * R + 0.59 * G + 0.11 * B$$





Surface Triangulation is not
restricted to terrains

Can be used to represent other
surfaces such as paper for example



How to Triangulate?

- Generating the meshes has to be tied into some sort of 'cost equation' or 'target'
 - Simplest means is joining all points in a grid
 - Typically grids consists of many thousands of points
 - What if we don't have a grid of points ?
 - Polynomial interpolation!
 - What if point data is repetitive or is over represented ?
 - Decimation, but How?
 - » Some simple Data metric
 - » Sampling rate problems (change in elevation vs grid size)
 - Use of triangle face sets vs triangle strips
 - Delaunay triangulation

Properties of Meshes

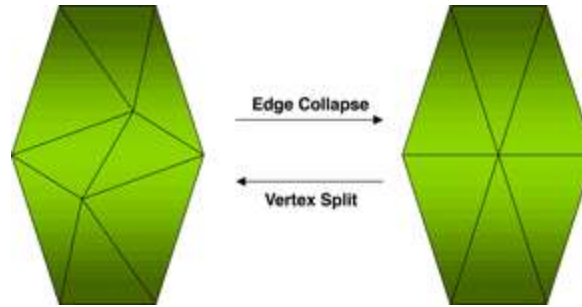
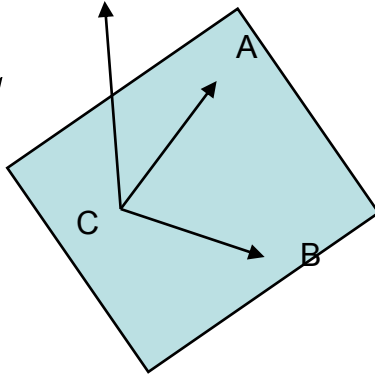
- **Solidity**
 - Faces are closed, encloses a finite volume
- **Connectedness**
 - Traverse from any vertex to any other
- **Simple**
 - No holes etc.
- **Planarity**
 - All polygons making up the mesh are planar
- **Convexity**
 - Line connecting any two points on mesh lie within the object itself

Triangle Contd.

$$P(s, t) = \sum + \sum s + \sum t$$

$$a = 4 - \sum$$

$$b = 3 - \sum$$

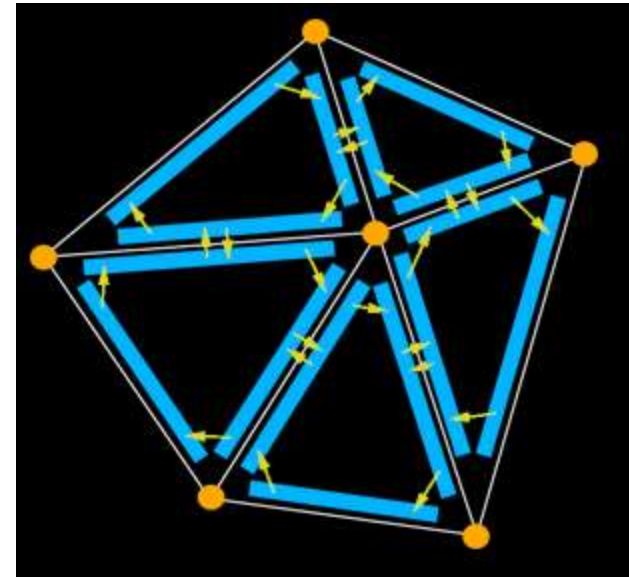


•Desirables

- Well form triangles (Maximize all angles)
- Represent surface accurately with minimum possible entities

Typical Questions?

- Which faces use this vertex?
- Which edges use this vertex?
- Which faces border this edge?
- Which edges border this face?
- Which faces are adjacent to this face?



Images and info courtesy Flipcode

Make a plane,

Useful for number of purposes.

- How would one construct this plane ?
- What is the generalized equation of plane in 3D ?

Some Common Requirements

- Minimalism
 - We want surface represented with least possible number of points and triangles
- Accuracy
 - When reducing the points or triangles we do not want to lose any topographic information
 - Mesh must retain features at edges etc.
- Robustness
 - For ease of rendering triangles rendered must be well shaped.

Doing Triangle Strips

- Extracting data from grayscale height maps not accurate
- Good enough for most applications though
- Iterate through the image
 - Counter clockwise
- Floating point errors
- A number of things can be done on top of this
 - Elimination of extra points based on 'some' metric
 - Generation of normal information!
- How many triangles in your mesh?
 - $2n - 2 - k$ (n :- total number of points, k :- points on convex hull)

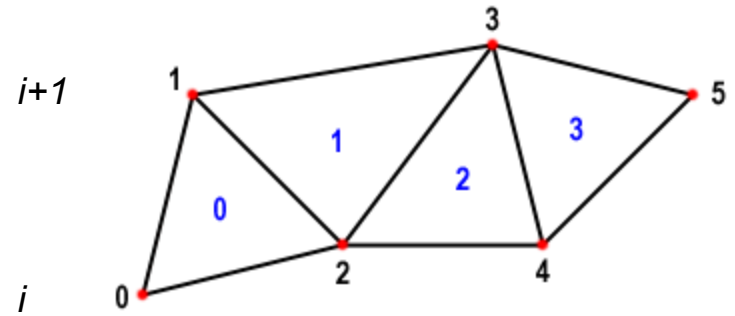
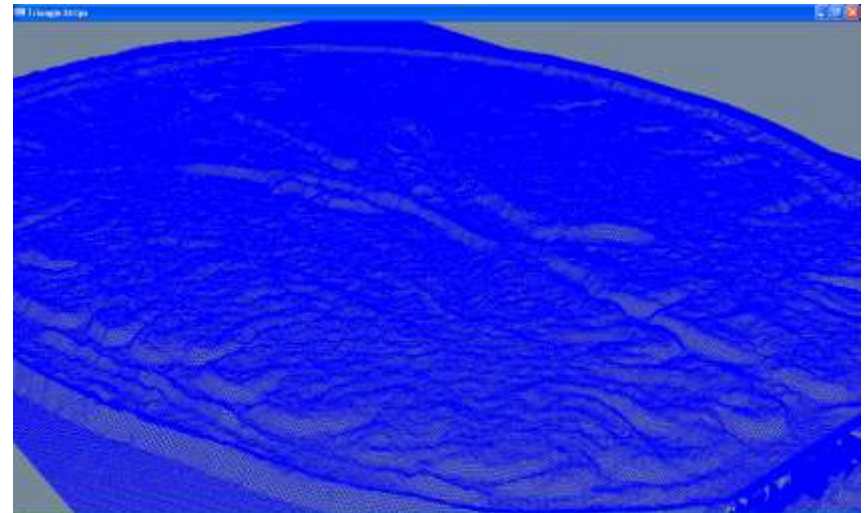
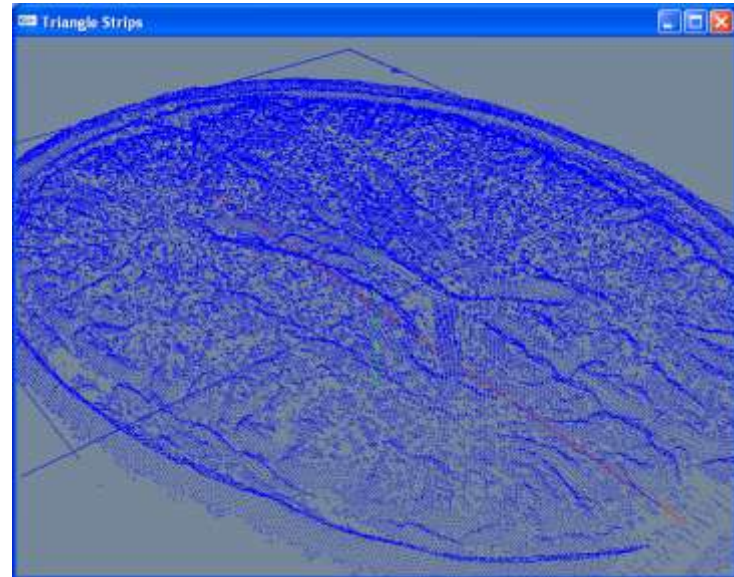
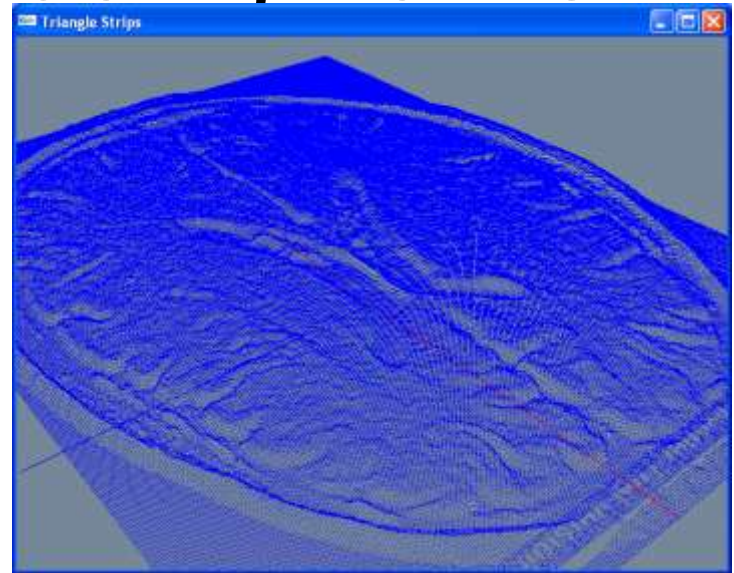


Fig. 1: Triangle strip



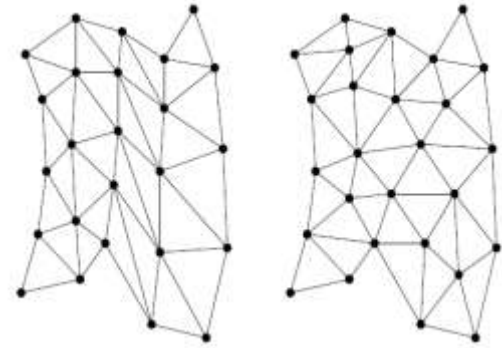
Decimation – Reducing Points

- Some very simple techniques can be used to reduce the number of points in our mesh
 - F.e. Compare with neighbors and eliminate points based on some threshold
 - Fit a plane for a selection of three points and check to see all other points that lie on that plane and eliminate them (Garland and Heckbert)
- New problem: No longer a regular grid (Structured grid of data)
- Generating a triangle mesh is no longer straight forward.
- Have to choose what three points have to be joined to form each triangle



Intro. To Delaunay (Info here courtesy. J Shewchuck)

- Key difference between structured and unstructured
 - Determination of neighbors of a node
 - Simple addition/arithmetic in case former, complex in case of latter
- Some considerations
 - Internal and External boundaries
 - Piecewise geometry, stuff that is linear and can actually be discretized
 - Floating values can be evil



Structured and Unstructured meshes, Same problem different solution sets



Figure 1.5: Los Angeles Basin.

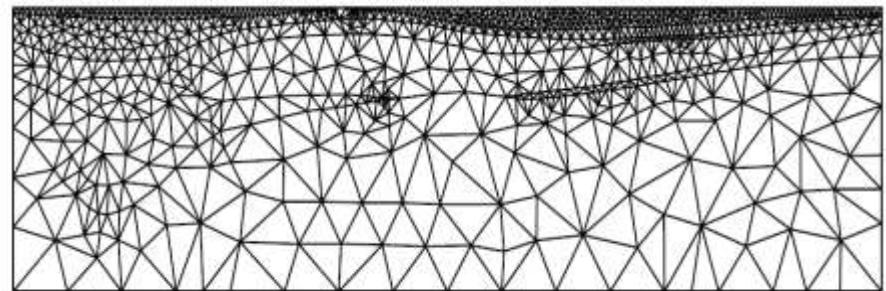
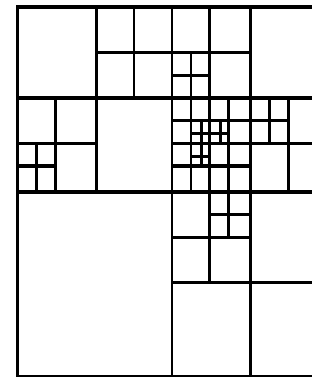
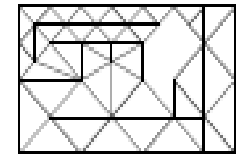
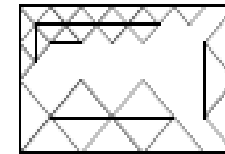
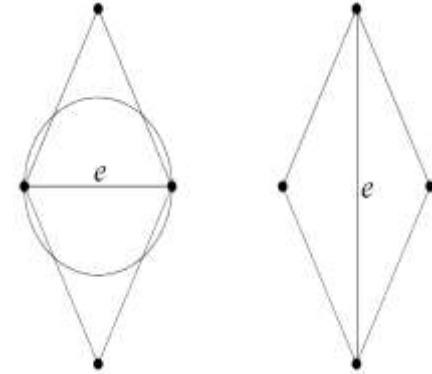
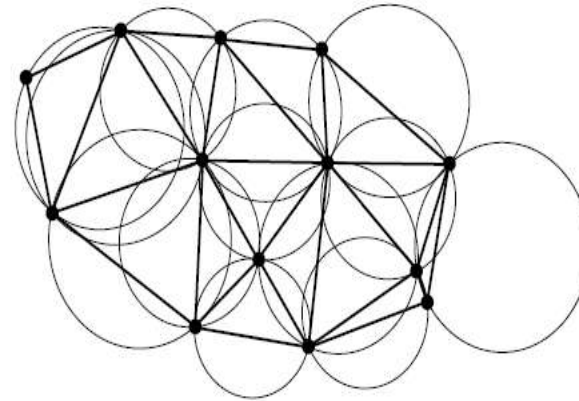


Figure 1.7: Unstructured mesh of Los Angeles Basin.

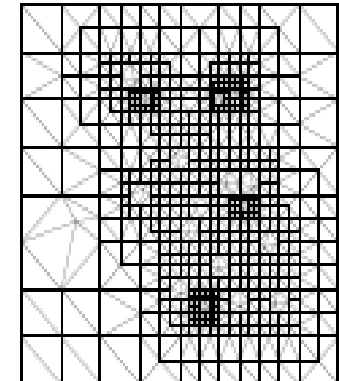
Delaunay Contd.

- Premise:

- Maximizes the minimum angle of triangles it generates
- Is a triangulation T of vertex set V such that none of the triangles intersect each other and T forms the convex hull of V
- For any set of points u, v (forming an edge) from the set V there should exist atleast one circumcircle that is empty
 - Realize an infinite number of circles can be constructed using a two points.
- A triangle formed from V is said to be delaunay



(a)

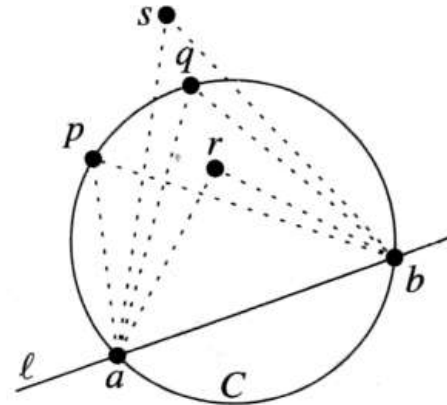


(b)

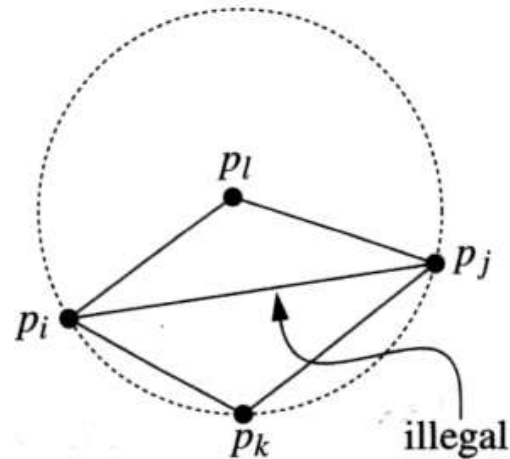
Images Courtesy Shewchuck

Thale's Theorem

- How to determine the maximum of the minimum angle of a triangulation T ?
- How to determine when and where to flip?
- Works out to the same as a circumcircle of any edge



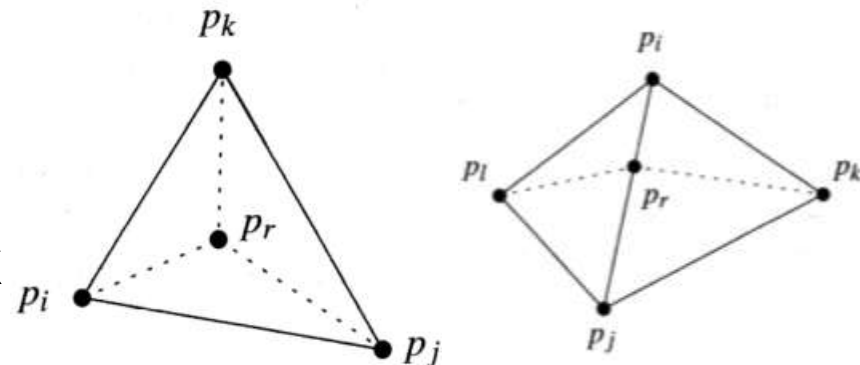
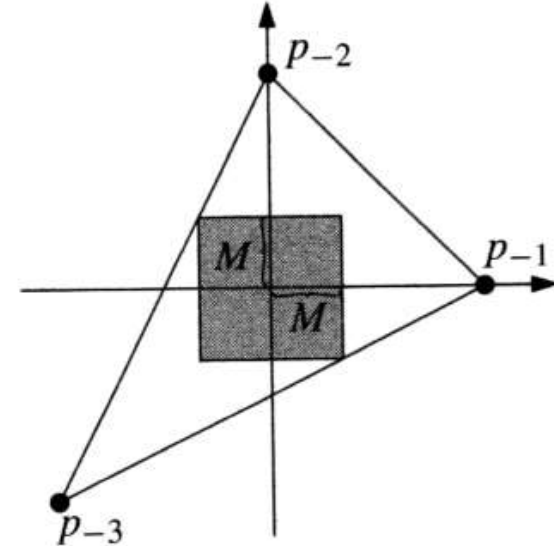
$$\angle arb > \angle apb = \angle aqb > \angle asb.$$



Images Courtesy: Glenn Eguchi

Bourke's Algorithm for Triangulation

1. Initialize the vertex list, an edge list and an empty triangle list
2. Determine a super triangle that will enclose the entire vertex list
3. Incrementally add vertices from the list to the mesh
4. Determine on which triangle the added vertex lies on
5. Draw edges from the vertex to complete the triangle
6. Perform edge flips to make all triangles and edges delaunay
7. Gather new edges to form triangles and add them to the triangle list
8. Repeat until all values in vertex list are exhausted.



Garland and Heckbert's Greedy Insertion (Taken from their implementation of Scape)

```
SCAN_TRIANGLE(Triangle T):  
  plane ← FIND_TRIANGLE_PLANE(T)  
  best ← nil  
  maxerr ← 0  
  forall points  $p$  inside triangle  $T$  do  
    err ←  $|H(p) - \text{INTERPOLATE\_TO\_PLANE}(p, \text{plane})|$   
    if err > maxerr then  
      maxerr ← err  
      best ←  $p$   
  T.heapptr ← HEAP_CHANGE(T.heapptr, maxerr, T)  
  T.candpos ← best
```

MESH_INSERT(Point p , Triangle T): Insert a new vertex in triangle T , and update the Delaunay mesh

```
INSERT(Point  $p$ , Triangle  $T$ ):  
  mark  $p$  as used  
  MESH_INSERT( $p$ ,  $T$ )    % incremental Delaunay triangulation  
  forall triangles  $U$  adjacent to  $p$  do  
    SCAN_TRIANGLE( $U$ )
```

```
GREEDY_INSERT():  
  initialize mesh to two triangles with the height field grid corners as vertices  
  forall initial triangles  $T$  do  
    SCAN_TRIANGLE( $T$ )  
  while not GOAL_MET() do  
     $T \leftarrow \text{HEAP\_DELETE\_MAX}()$   
    INSERT( $T.candpos$ ,  $T$ )
```

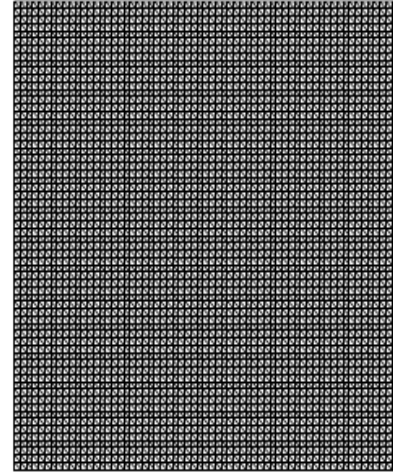


Figure 1: Uniform grid triangulation of 65×65 height field H .

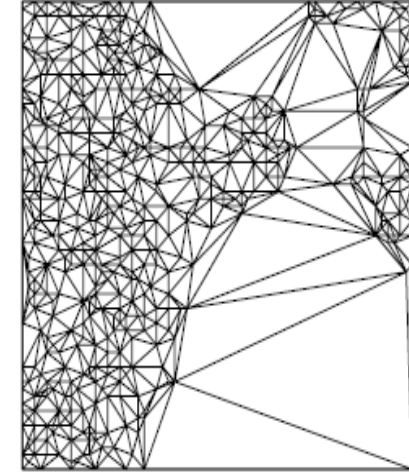
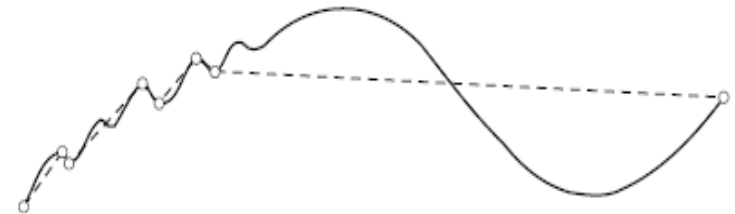


Figure 2: A triangulation TS using 512 vertices that approximates H ; an example of a triangulated irregular network.



Variations, Courtesy: Shewchuck and Bourke

1. PLSG :- Planar Straight Line Graphs, triangulations formed on them is constrained and hence not strictly Delaunay.

1. Constrained delaunay

2. Can be made delaunay by addition of extra vertices and dividing boundary edges

1. Conforming delaunay

2. Addition of Steiner points

3. Algorithms fall under following broad classes

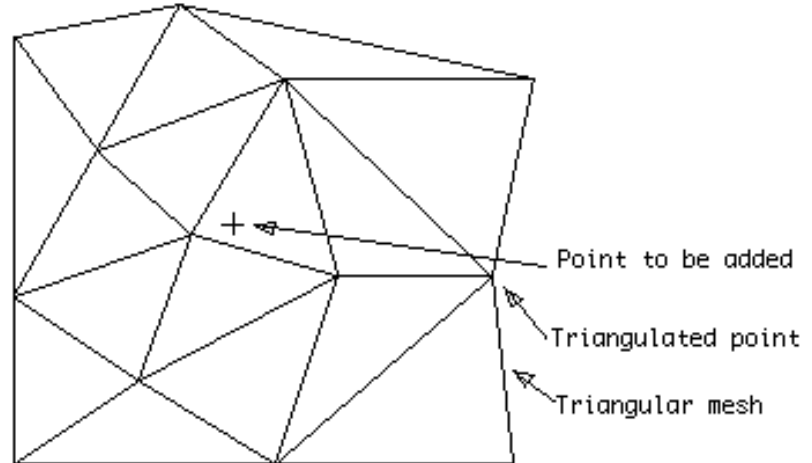
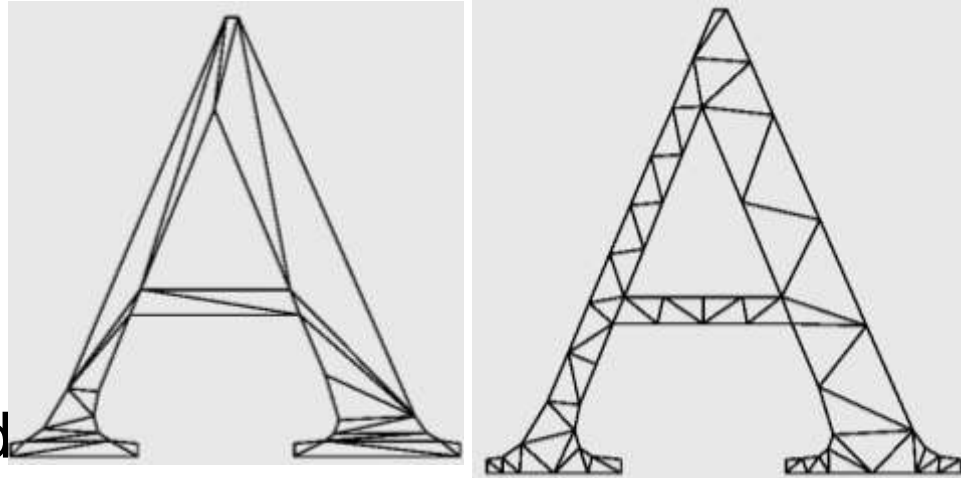
1. Divide and conquer

2. Advancing front

3. Tree based

4. Insertion

5. Plane Sweep



Example of Insertion