Spatial data:
Models and representation

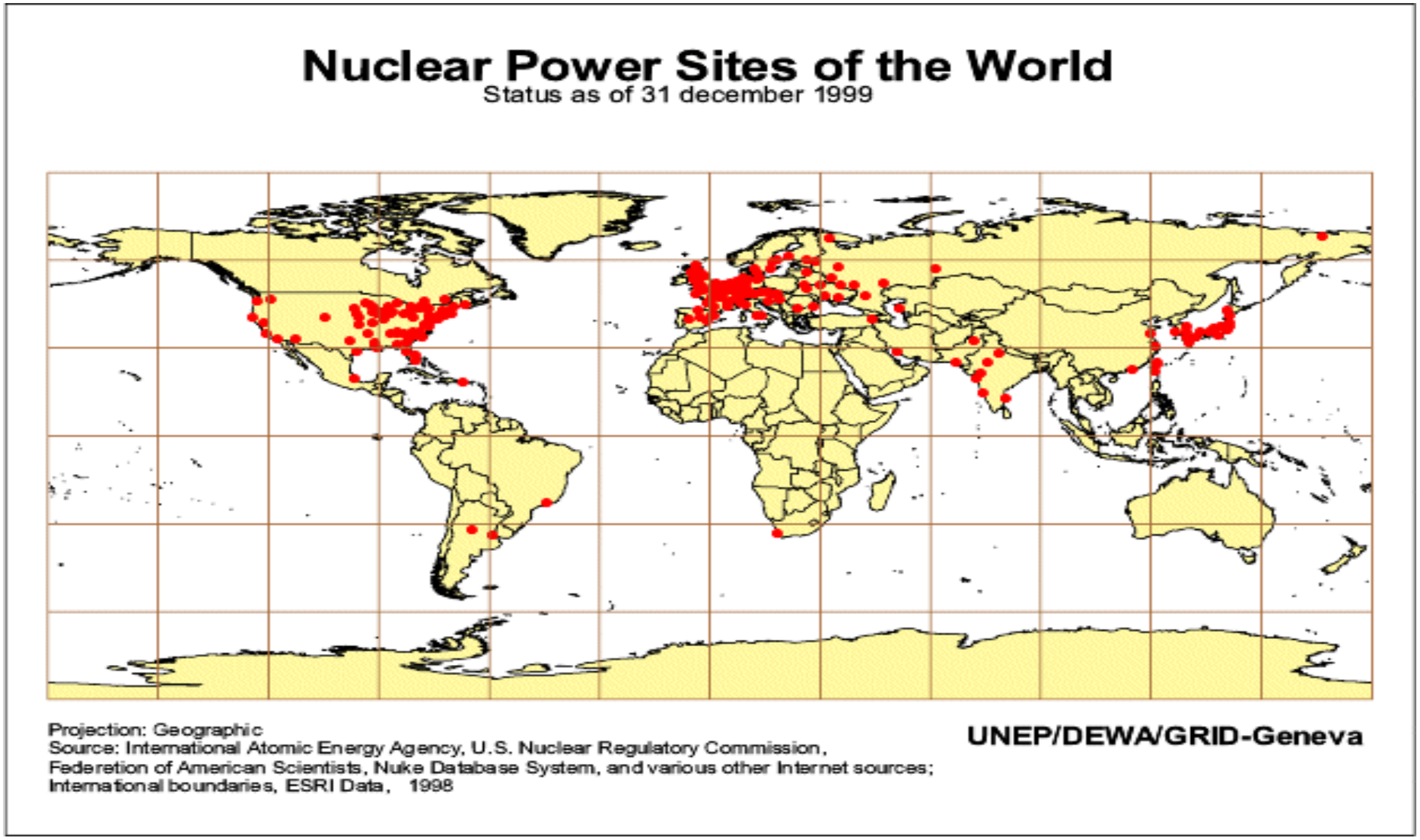
Laura Toma
csci 3225
Algorithms for GIS
Bowdoin College

Overview

- Spatial data
- Networks
- Terrains in GIS
  - Grids
  - TINs
  - Data structures for TINs
  - Grid or or TIN?
- Planar maps and meshes
  - Data structures for meshes

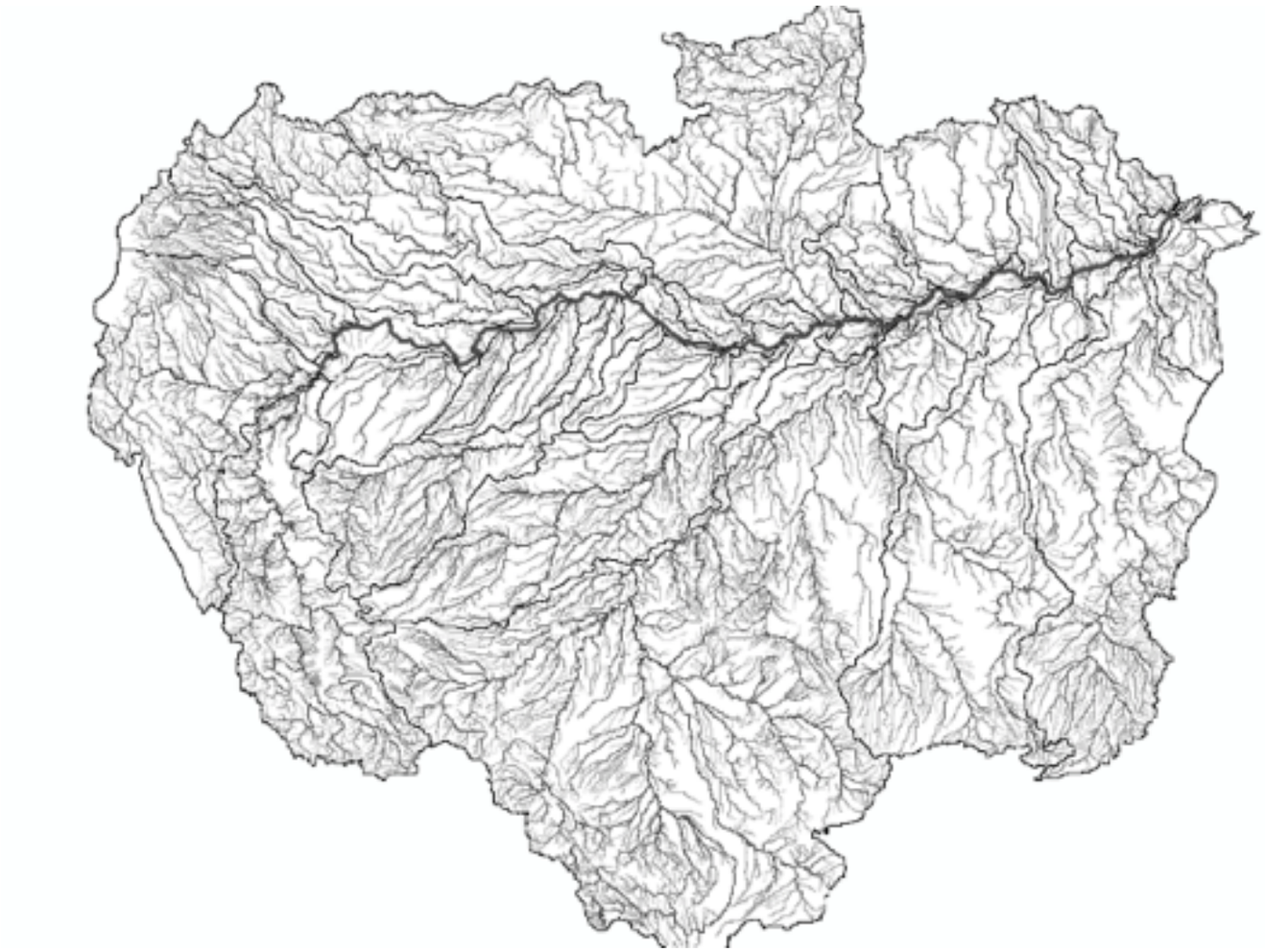
Spatial data in GIS

- Point data



Spatial data in GIS

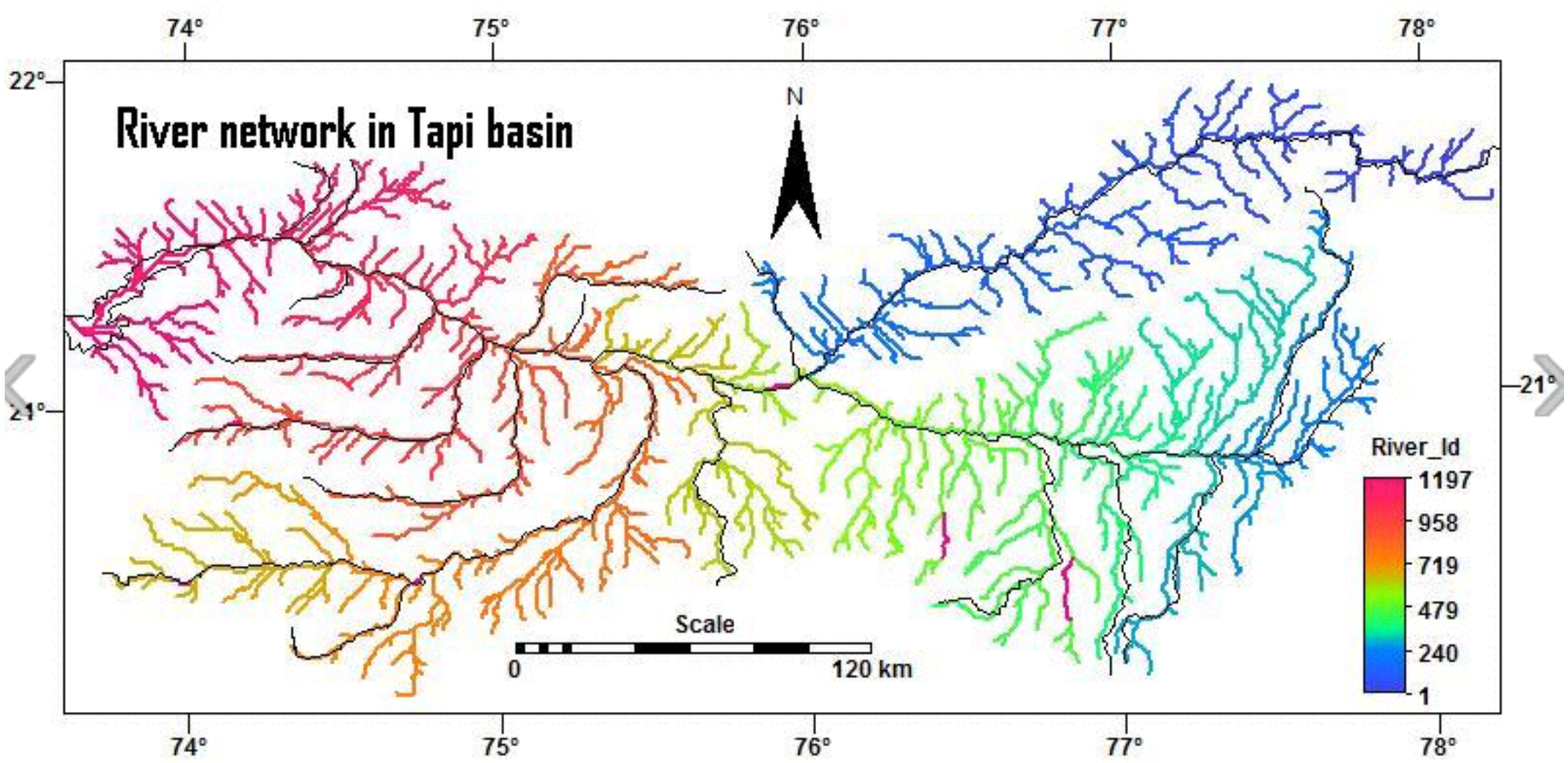
- Line data: NETWORKS



river network

Spatial data in GIS

- Line data: NETWORKS



river network

Spatial data in GIS

- Line data: NETWORKS

train/bus/road network

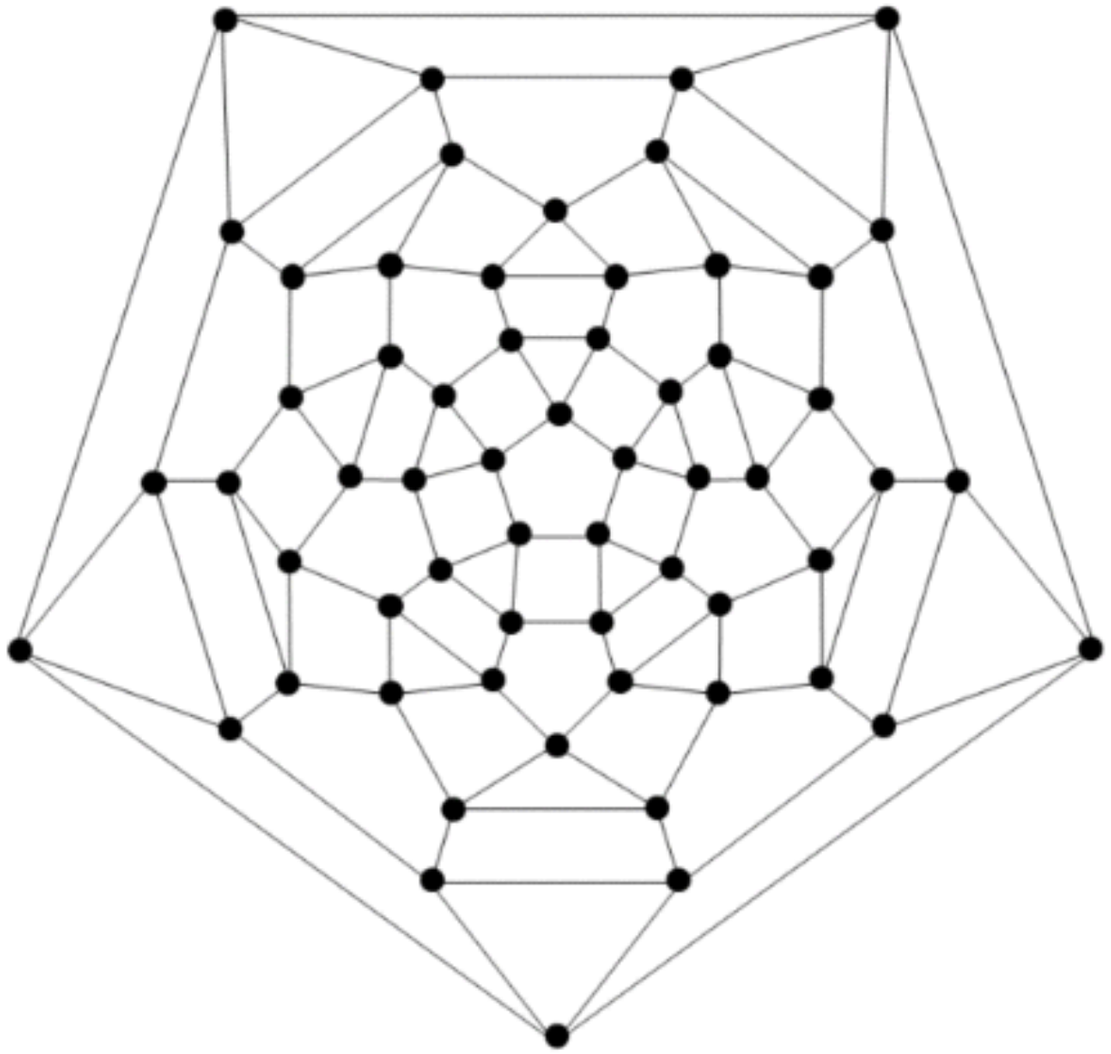
Spatial data in GIS

- Polygons: PLANAR MAPS

state boundaries

Spatial data in GIS

- Polygons: PLANAR MAPS

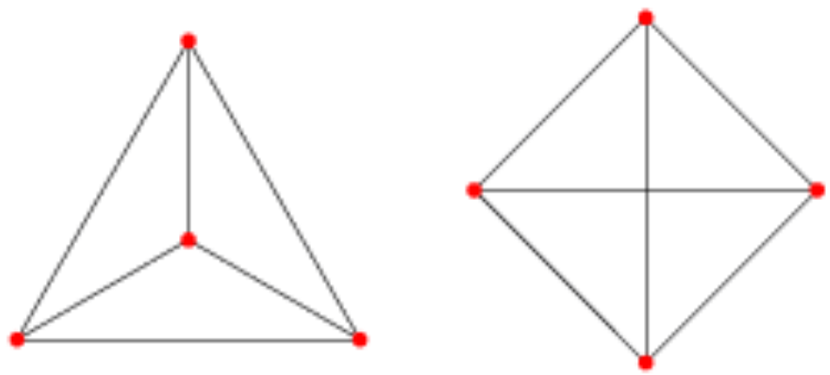


state boundaries

Detour

Definition:

A graph is called ***planar*** if it can be drawn in the plane such that no two edges intersect except at their endpoints. Such a drawing is called a planar embedding of the graph.



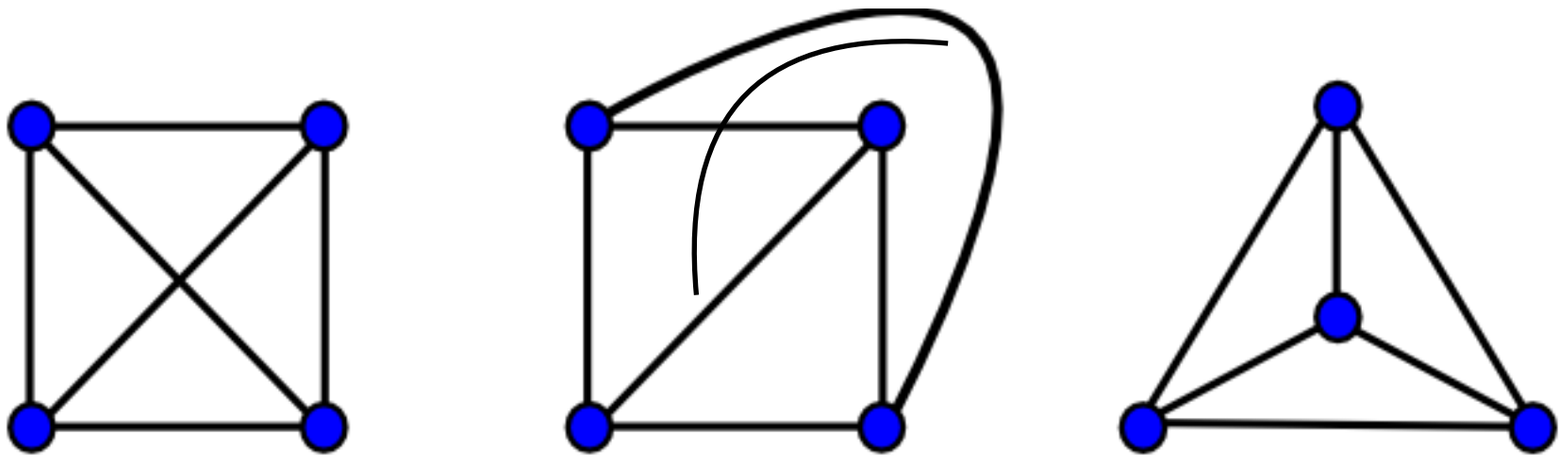
Two drawings of the same graph. Since there exists a planar embedding, the graph is planar.

Note: It's possible to draw non-planar embeddings of planar graphs.

Detour

Definition:

A graph is called ***planar*** if it can be drawn in the plane such that no two edges intersect except at their endpoints. Such a drawing is called a planar embedding of the graph.

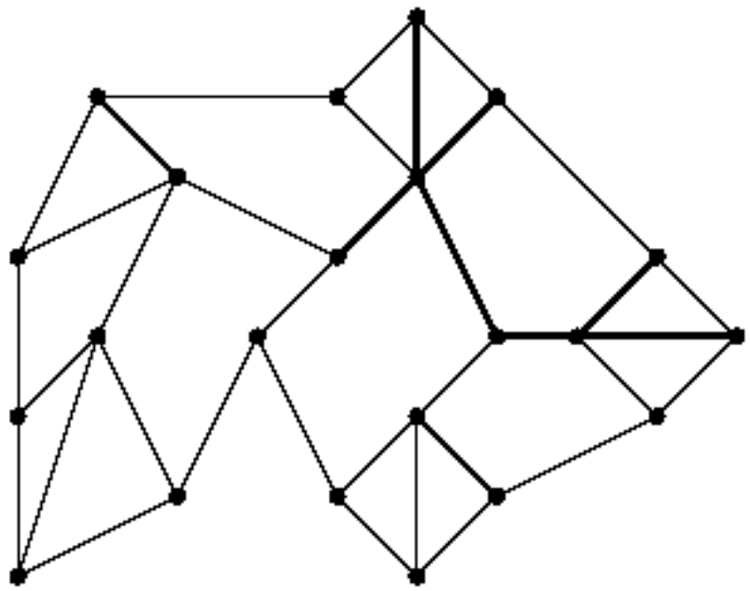


http://people.hofstra.edu/geotrans/eng/methods/img/planarnonplanar.png

Note: Edges can be represented as simple curves in the drawing

Detour

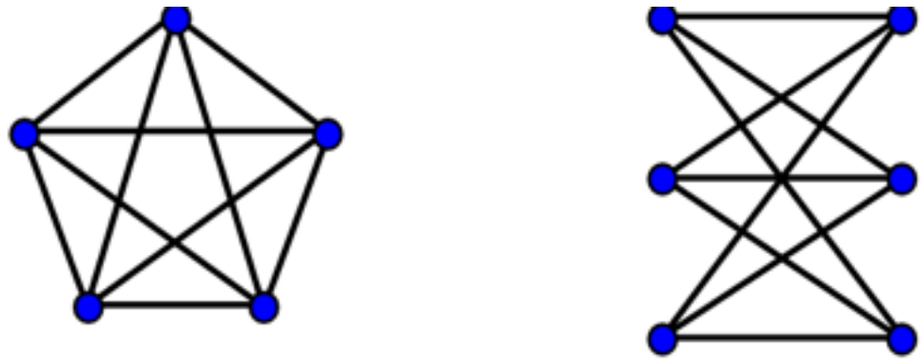
- A planar graph introduces a subdivision of the plane into regions called faces, which are polygons bounded by the graph's edges.



Detour

Some results:

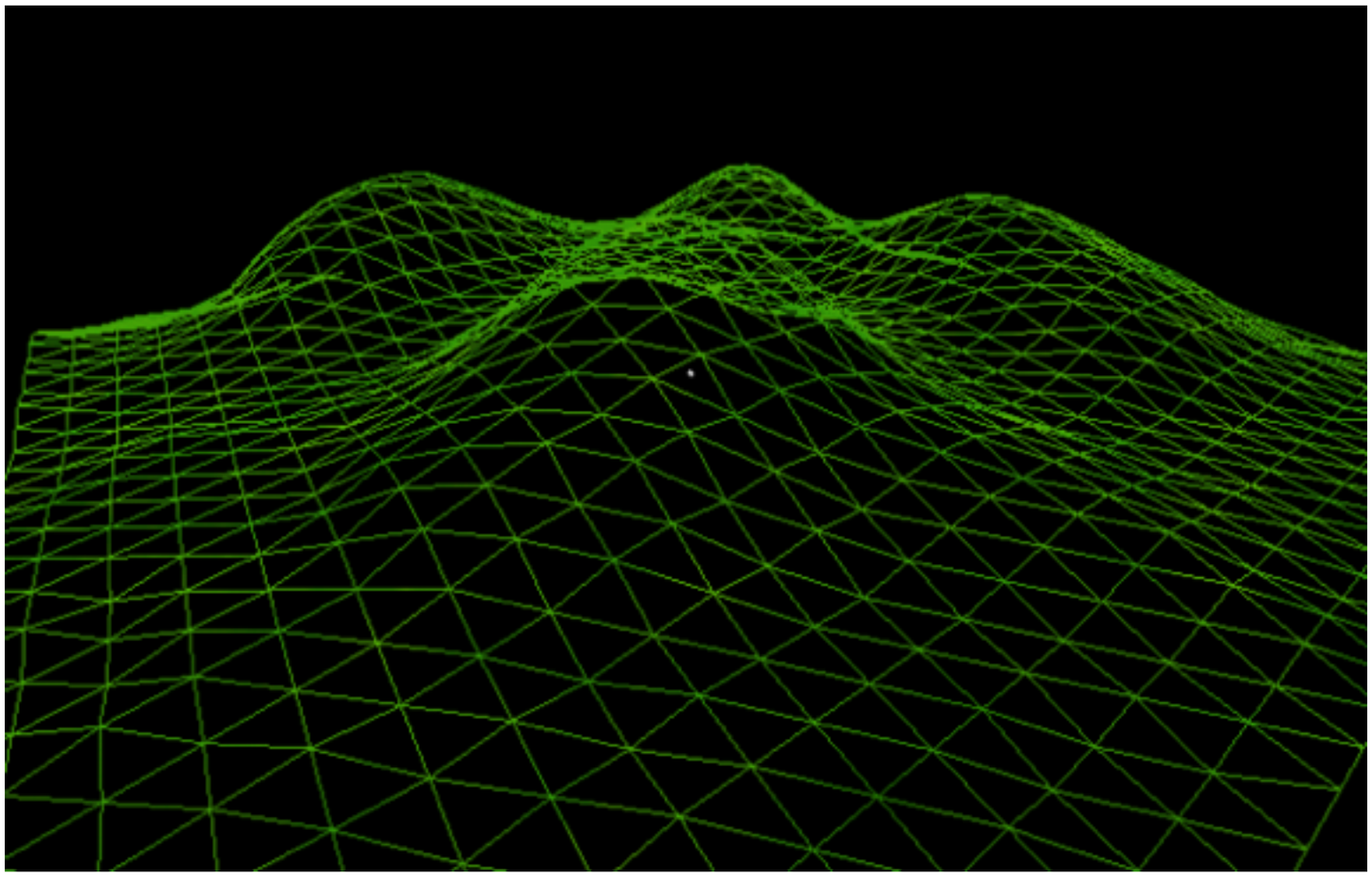
- Any planar graph has a planar straight-line drawing where edges do not intersect [Fary's theorem].
- A graph is planar iff it has no subgraphs isomorphic with K5 or K3,3 [Kuratowski's theorem].



- A graph is planar iff it has a dual graph.
- Any planar graph has at least one vertex of degree <= 5.
- There are a number of efficient algorithms for planarity testing that run in $o(n^3)$, but are difficult to implement.

Spatial data in GIS

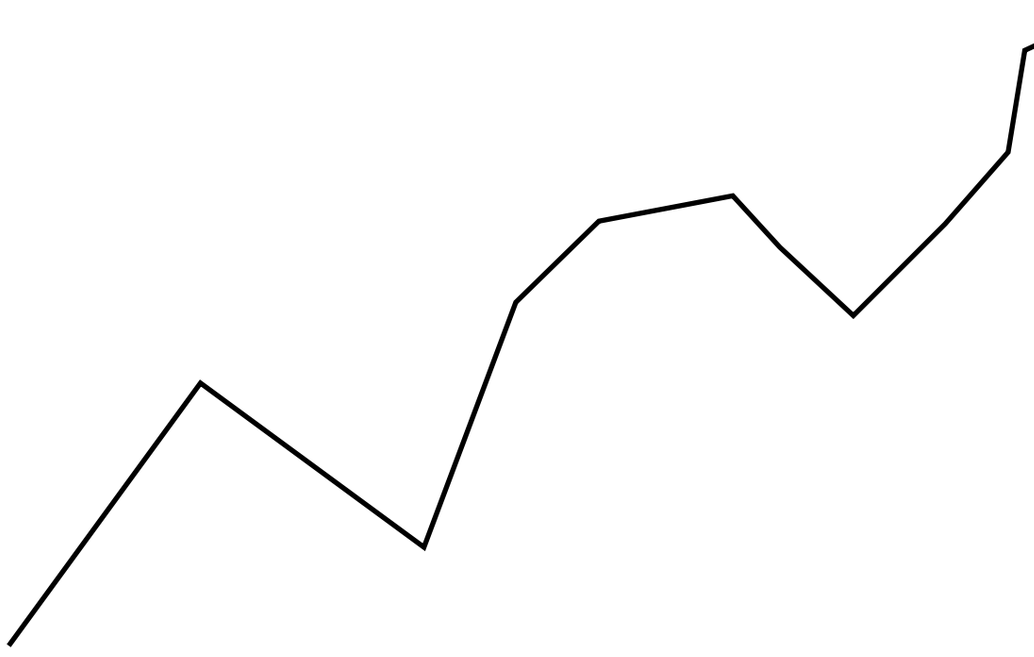
- TERRAINS
  - A surface such that any vertical line intersects it in at most one point (in CG terms: xy-monotone)



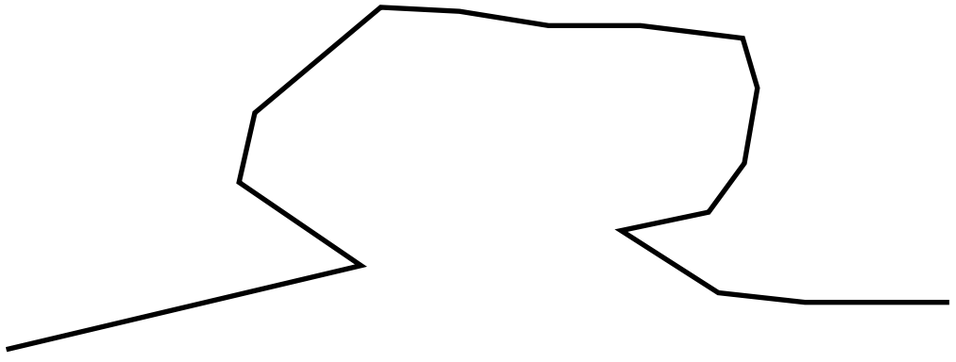
elevation

Detour

- A polygonal chain (polyline) is x-monotone if any vertical line intersects it at most once.
- A polyline is monotone wrt a line *l* if any line perpendicular to *l* intersects it at most once.



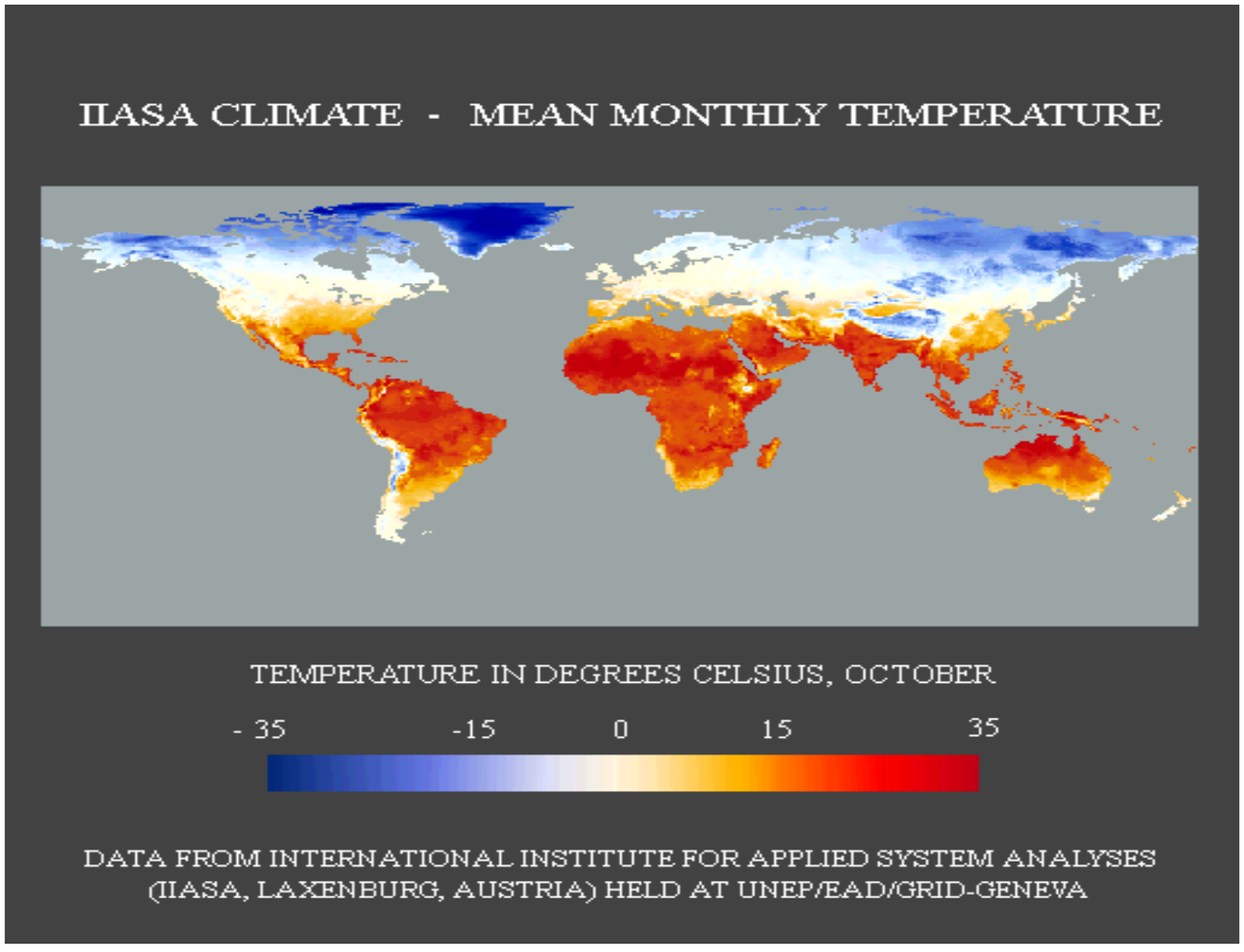
x-monotone



NOT x-monotone

Spatial data in GIS

- Terrains
  - A surface such that any vertical line intersects it in at most one point (in CG terms: xy-monotone)

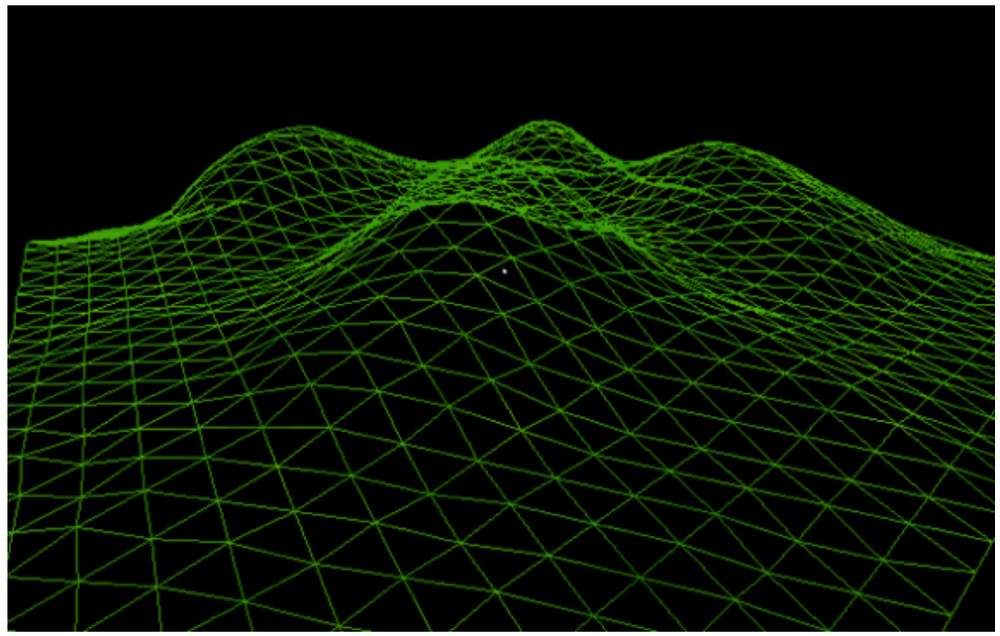


temperature

Spatial data in GIS applications

networks

terrains



planar maps

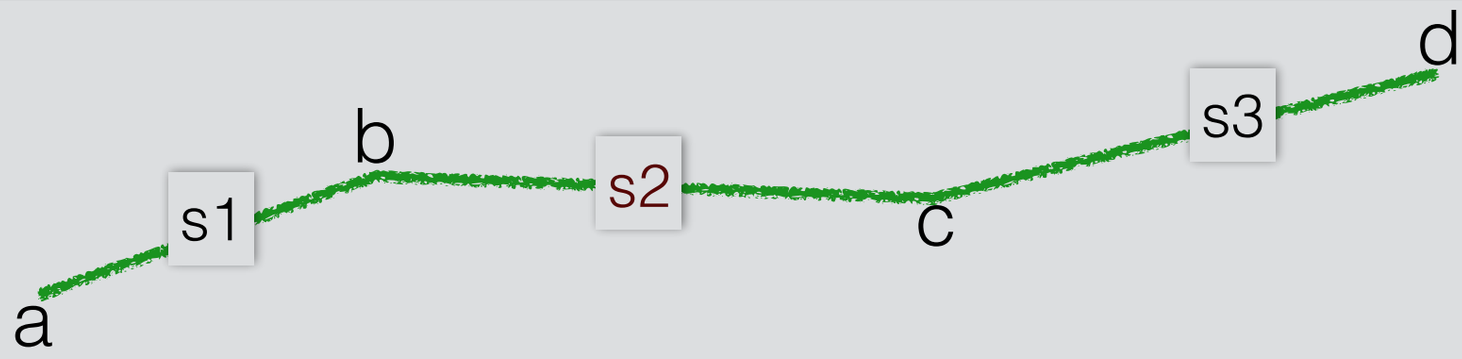
Suppose we have to solve a problem on networks/maps/terrains: what is a good representation?

Networks

Data structures for networks

- What do we expect to do with a network?
  - traverse paths
  - find all vertices adjacent to a given vertex,…

- First attempt
  - list of points, each point stores its coordinates
  - list of segments, each segment stores pointers to its vertices



- points = { a:(x_a, y_a), b:(x_b, y_b), c:(x_c, y_c), d:(x_d, y_d) }
- segments = { *(a, b), (b,c), (c,d)* }
- Assume you want to traverse a path starting at point *a*:
  - search through the segment list looking for a segment with startpoint *a*; you find *(a,b)*
  - search through the segment list looking for another segment with startpoint *b*; you find *(b,c)*

Data structures for networks

- What do we expect to do with a network?
  - traverse paths
  - find all vertices adjacent to a given vertex,…

- First attempt
  - list of points, each point stores its coordinates
  - list of segments, each segment stores pointers to its vertices

- Not efficient!
  - need a data structure that allows to traverse paths efficiently

Data structures for networks

- What do we expect to do with a network?
  - traverse paths
  - find all vertices adjacent to a given vertex,…

- Second attempt
  - a network is a graph!
  - use adjacency list (matrix, if dense)

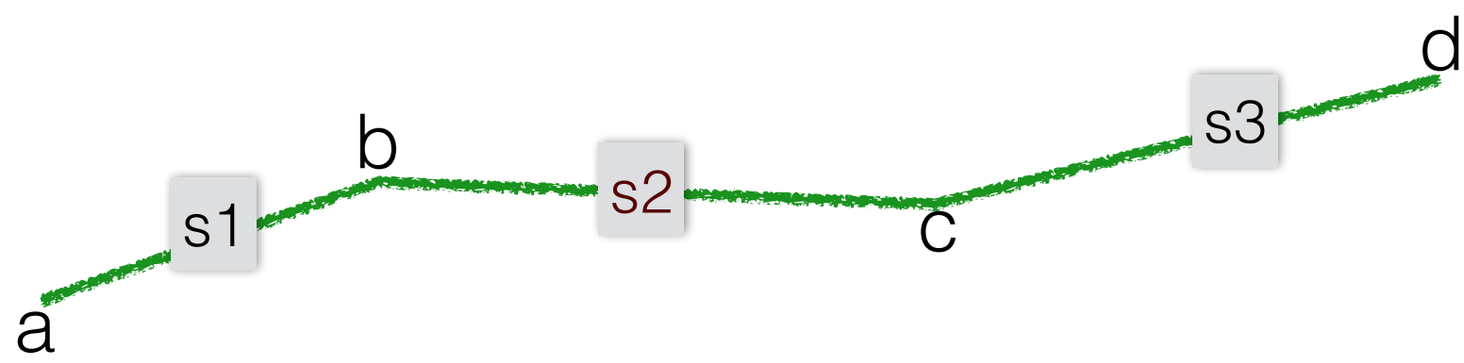
- Usually raw data comes without explicit topology
  - spaghetti data/ data structure (data is like spaghetti, no structure and they all mingle together)
  - need to build the adjacency list corresponding to raw data

Data structures for networks

Assume you download raw data for a (directed) network as a file, which contains a set of edges and their endpoints.

1. Describe how you would build an adjacency list from it.
2. Estimate the amount of memory you need for your structure.

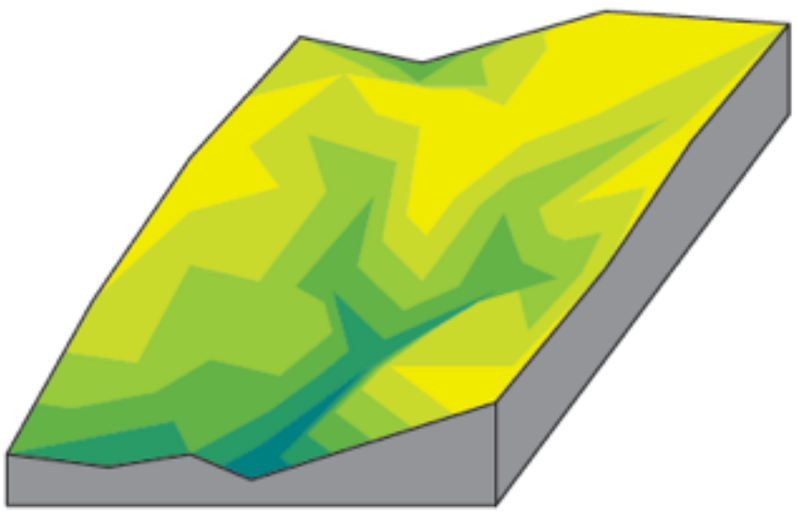
Analyze function of |V} vertices and |E| edges.



- file = { (x_a, y_a), (x_b, y_b), (x_c, y_c), (x_d, y_d), (x_b, y_b), (x_c, y_c), (x_c, y_c), (x_d, y_d) }

Terrains

Digital terrain models



watershed maps

risks of
floods, landslides,
eruptions, erosion

visibility maps

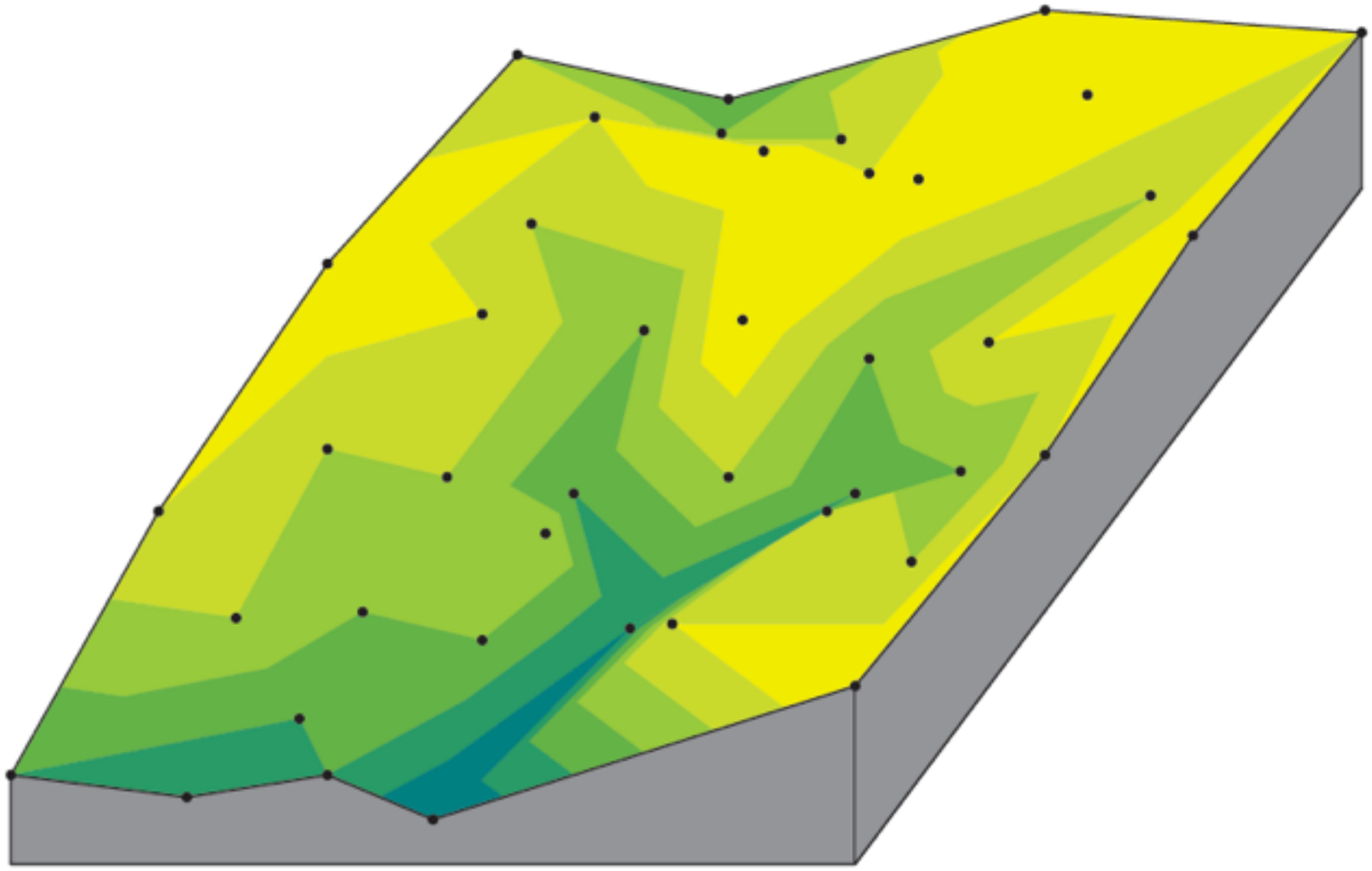
costs/benefits of
roads, buildings, dikes,
hydropower plants,
solar power plants

least-cost paths

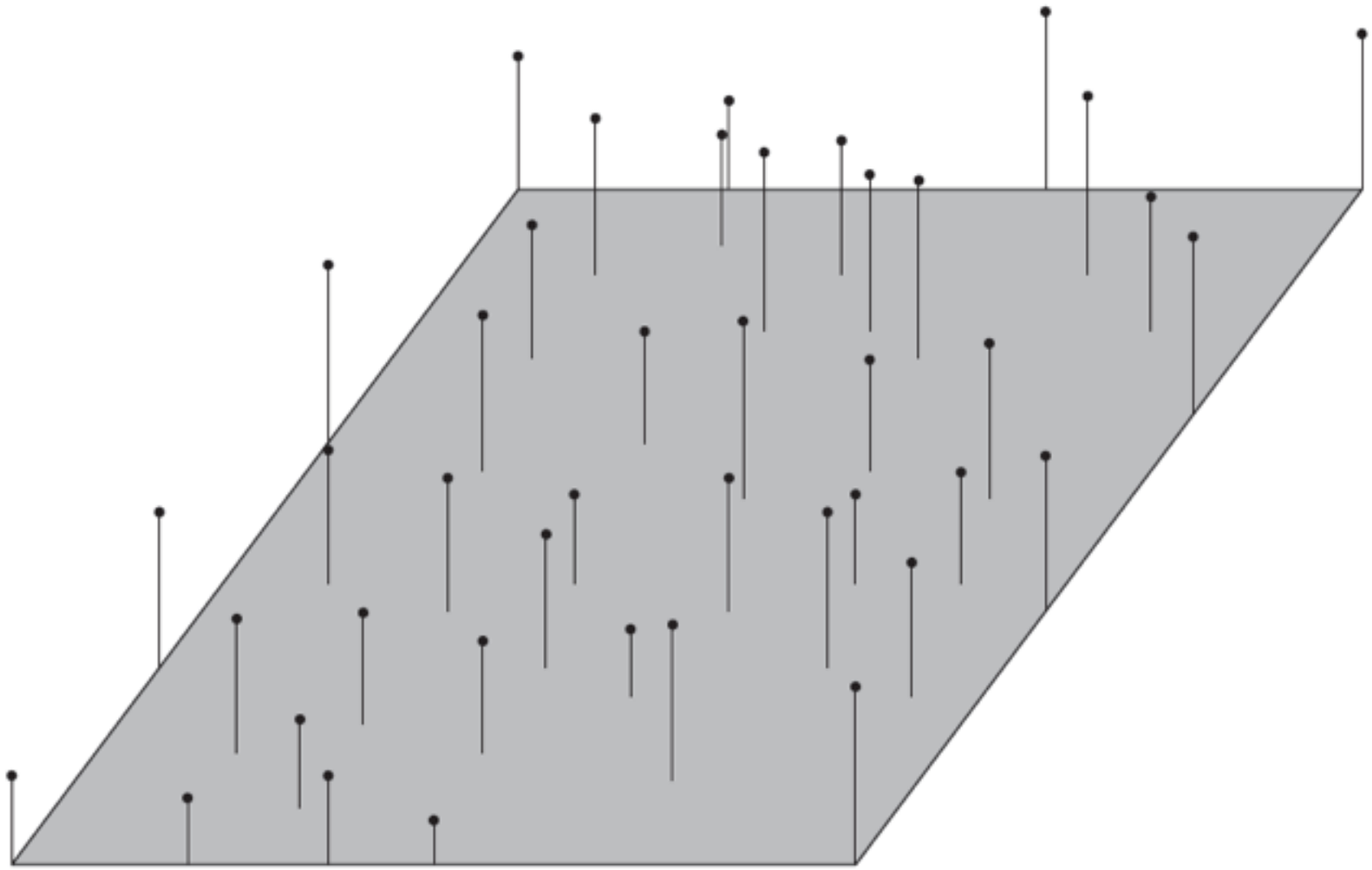
possible locations of
(pre-)historical roads
and settlements

analysis of animal
behaviour and evolution

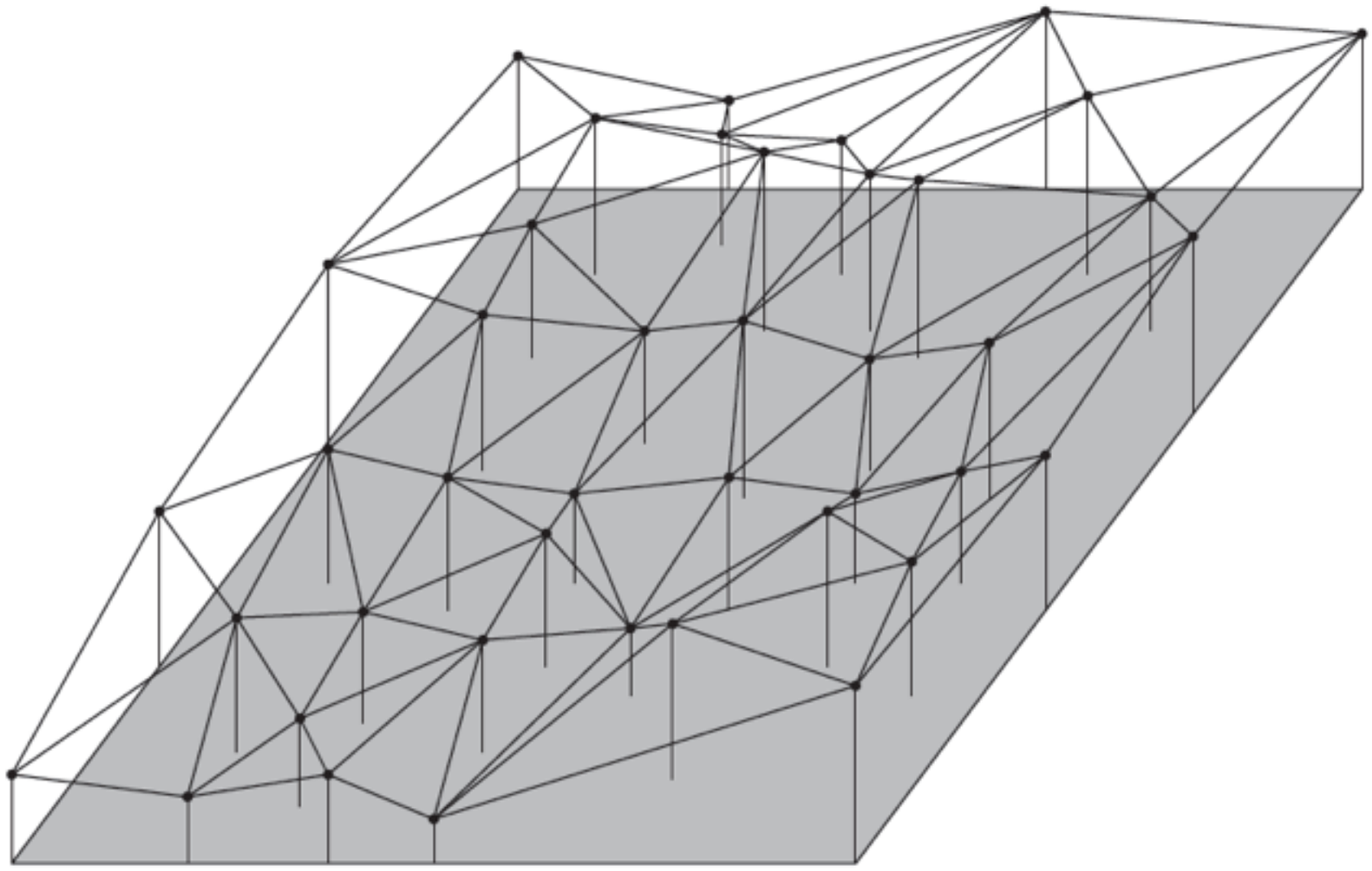
Digital terrain models



Digital terrain models

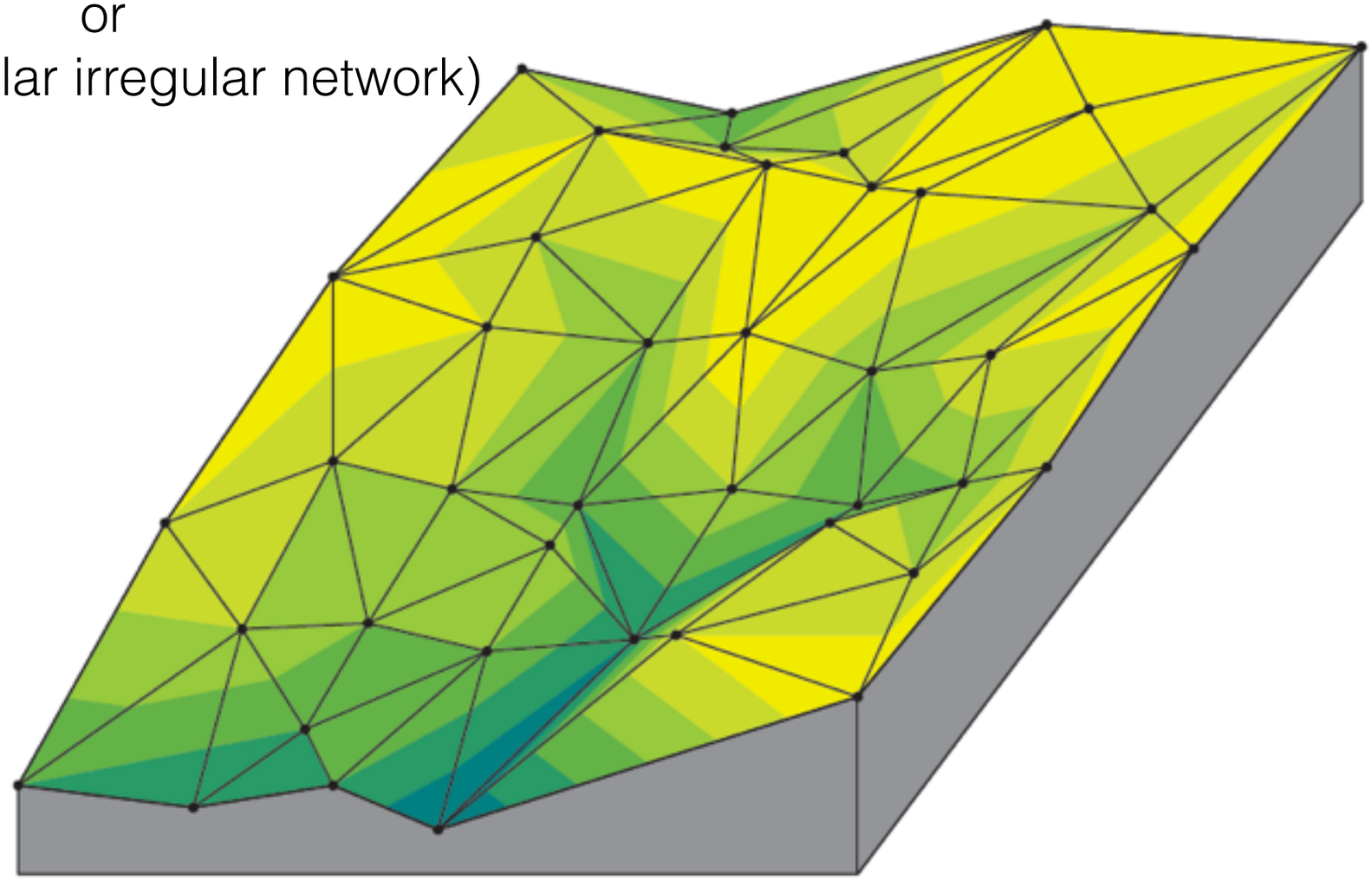


Digital terrain models



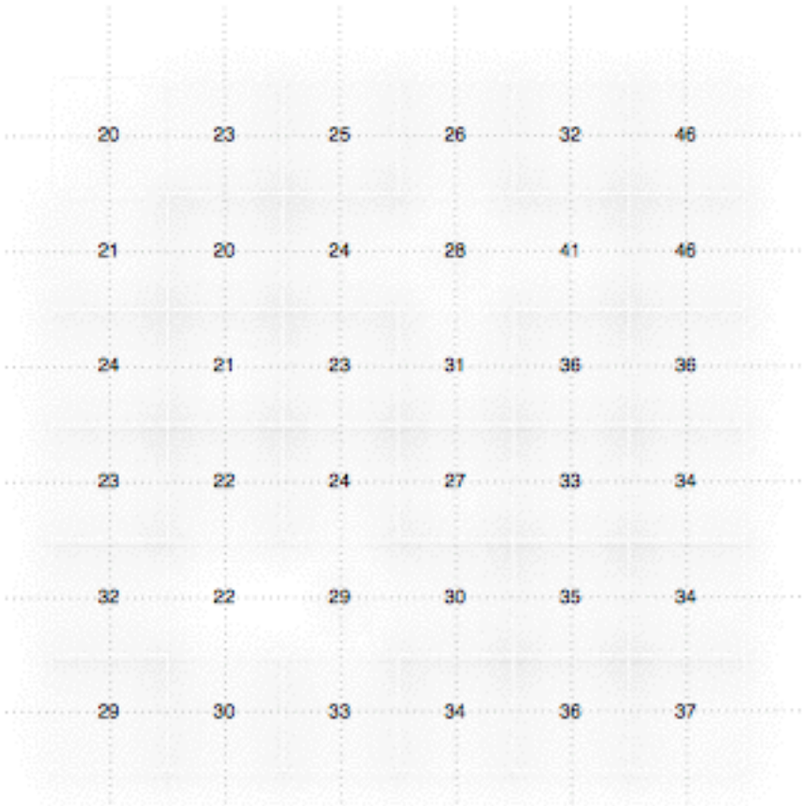
Digital terrain models

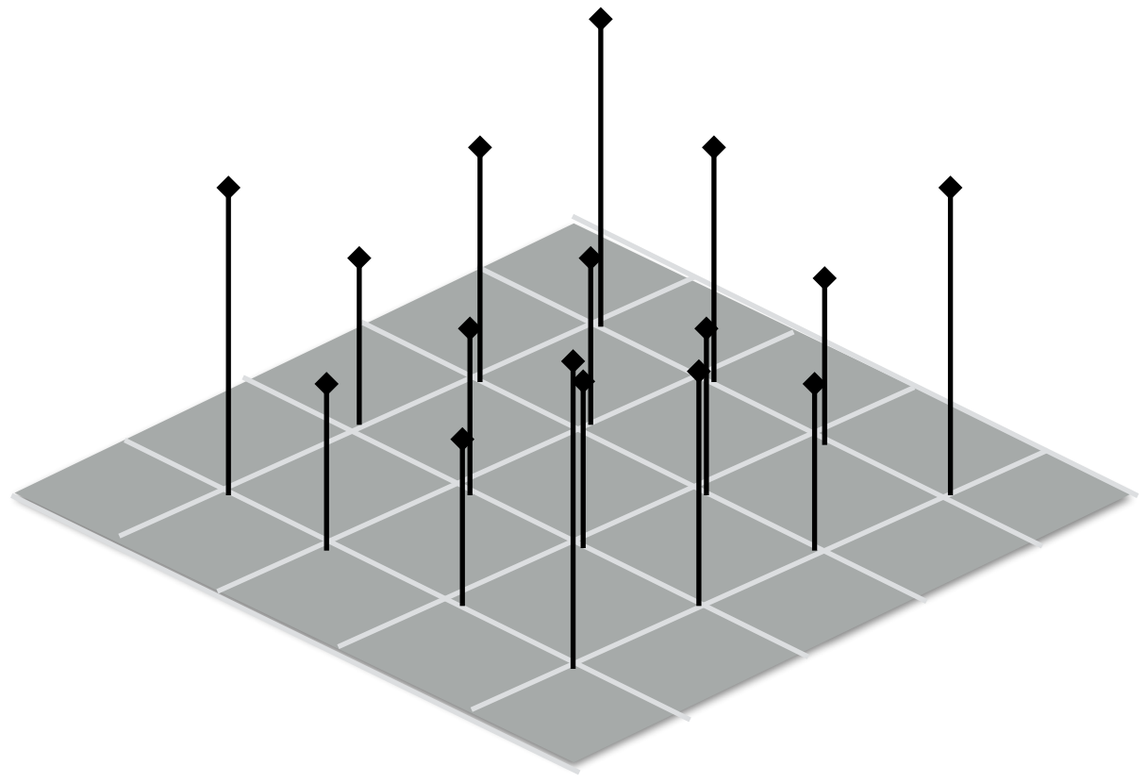
triangular mesh
or
TIN (triangular irregular network)



Digital terrain models

raster/grid



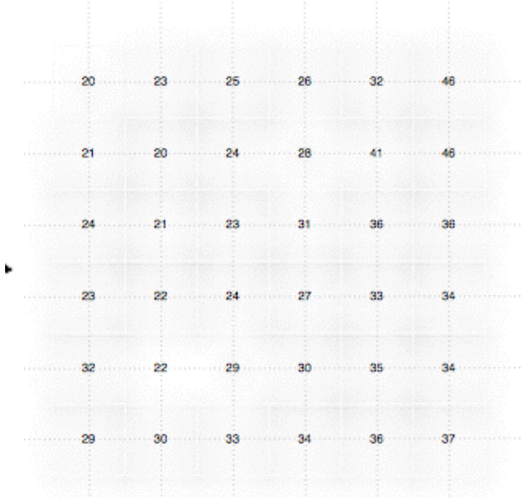


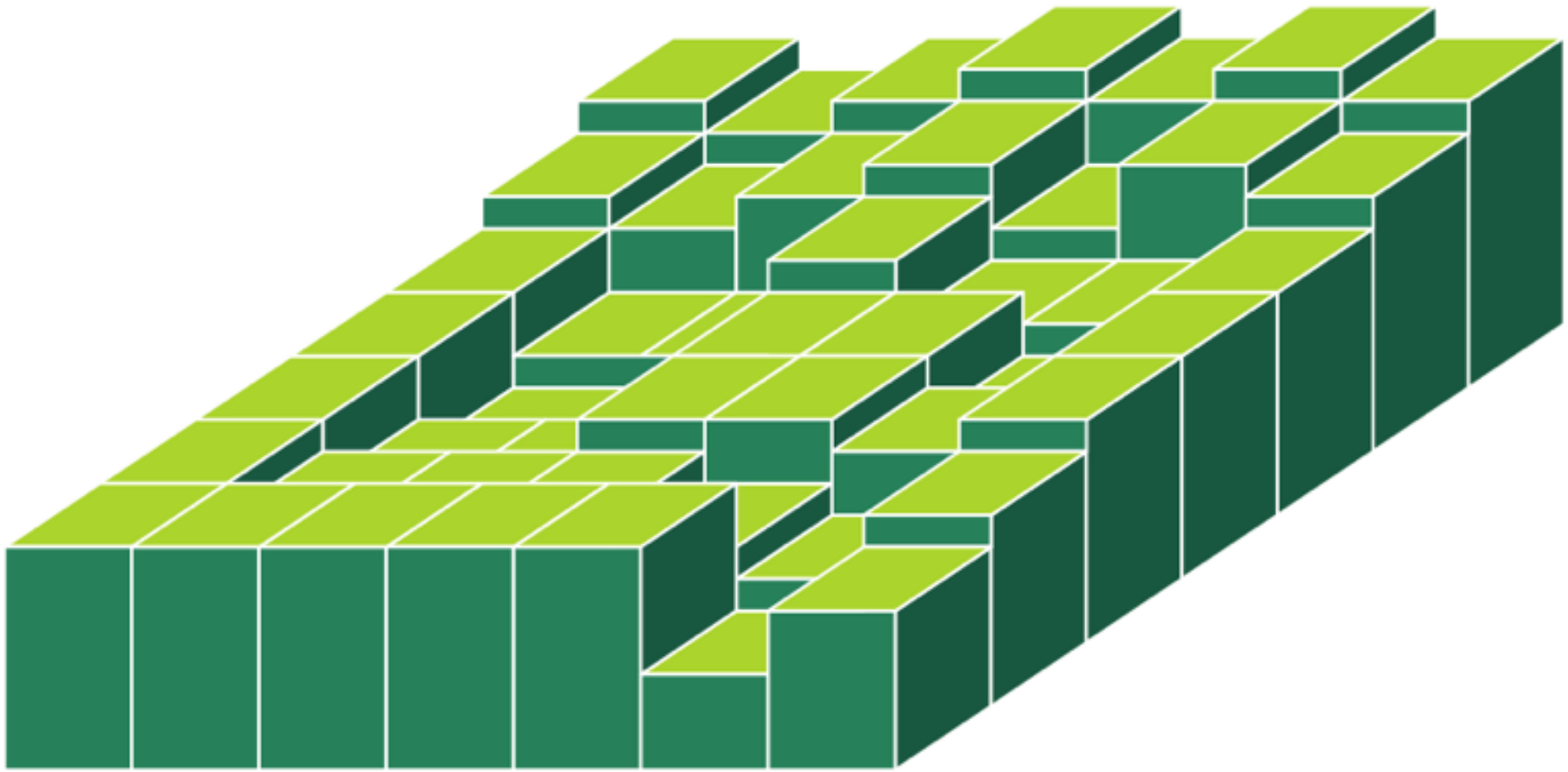
- Grid = 2D array of values
- Grids need an interpolation method

    - nearest neighbor
    - linear
    - bilinear, splines, krigging, IDW, etc

20	23	25	26	32	46
21	20	24	28	41	46
24	21	23	31	36	36
23	22	24	27	33	34
32	22	29	30	35	34
29	30	33	34	36	37

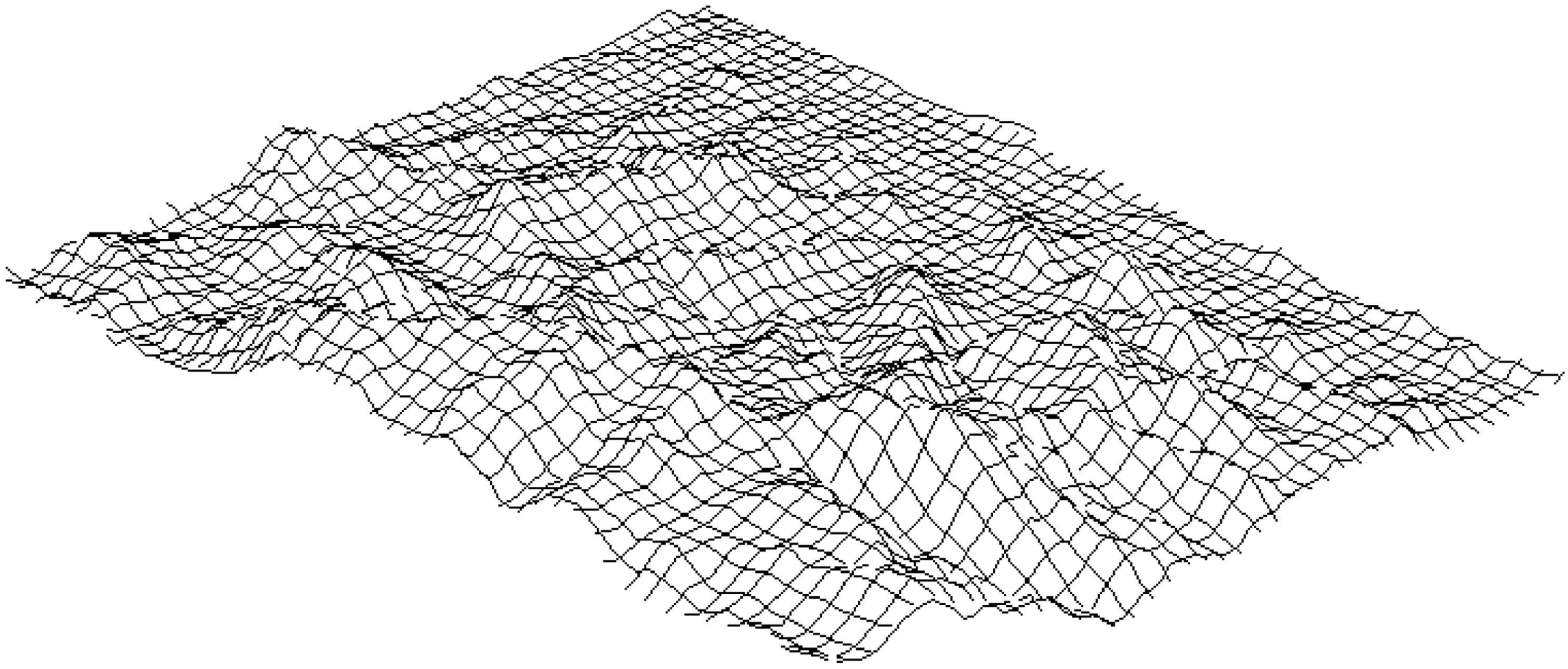
Grids with nearest neighbor interpolation



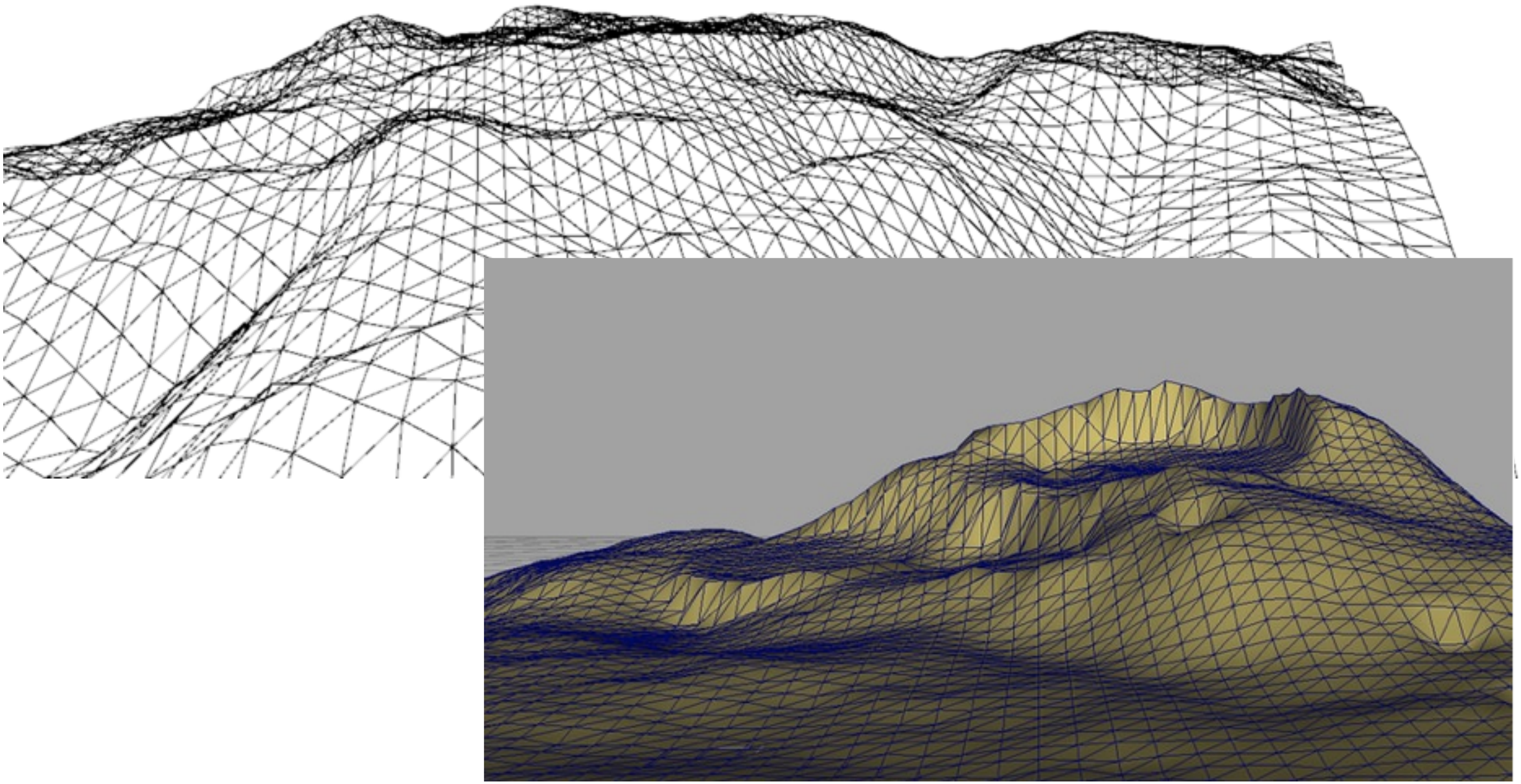


Grids with linear interpolation

20	23	25	26	32	46
21	20	24	28	41	46
24	21	23	31	36	36
23	22	24	27	33	34
32	22	29	30	35	34
29	30	33	34	36	37



Grids (+ linear interpolation) can be viewed as TINs



Grids

- Grid data easy to obtain from aerial imagery
  - rasters = images (image: grid of color pixels)
  - massive amounts of aerial imagery available
  - SAR interferometry: by combining Synthetic Aperture Radar (SAR) images of the same area it is possible to generate elevation maps

Grid elevation data sources

- USGS
  - http://earthexplorer.usgs.gov

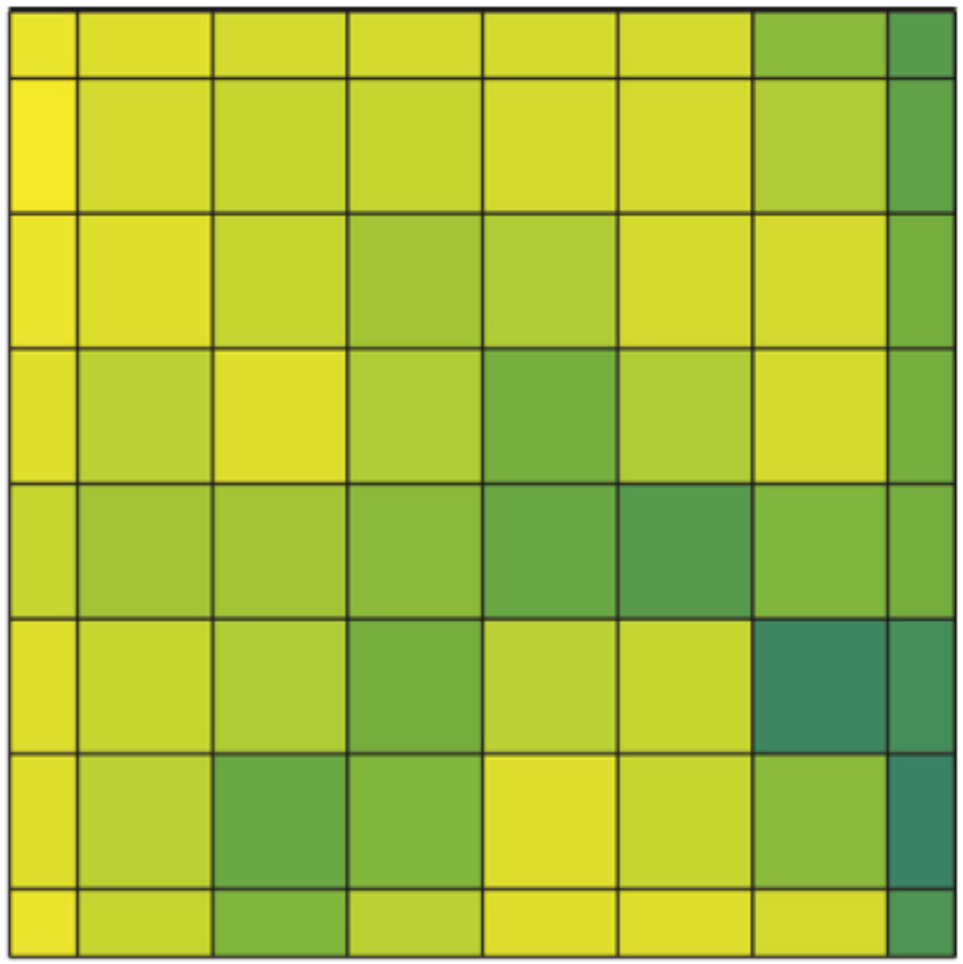
- SRTM 90m elevation data for entire world
  - http://srtm.csi.cgiar.org/SELECTION/inputCoord.asp
  - can download tiles anywhere in the world
  - SRTM 30m data available for the entire USA (50+GB)

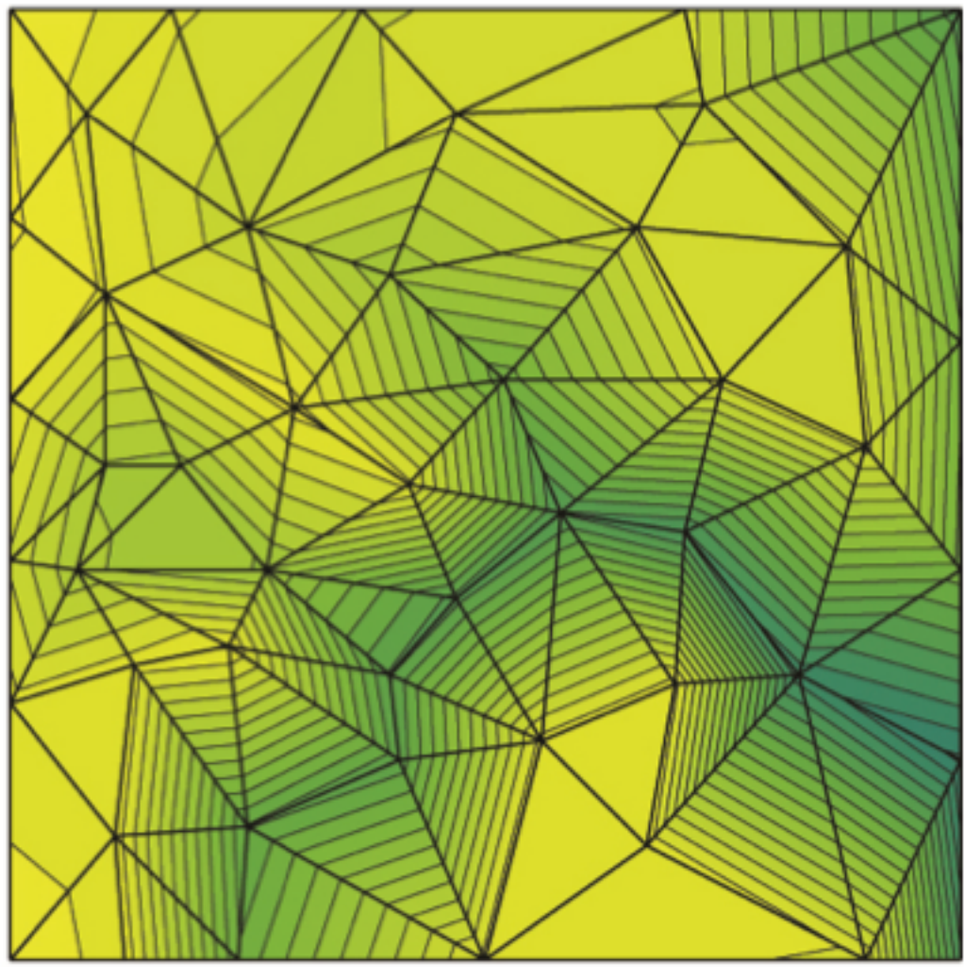
- … many others

Grid ascii format

```
ncols         391
nrows         472
xllcorner     271845
yllcorner     3875415
cellsize      30
NODATA_value  -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 781 773 766 761 756 751 745 738 730 723 716 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 775 767 759 754 750 746 741 735 728 721 714 709 705 700 696 694 693 692
691 690 691 692 696 701 707 714 722 728 732 733 731 726 721 718 718 722 729 737 746 755 761 764 762 760 760 759 754 748 741 733 725 718
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 770. . . . . . .
```

Digital terrain models: Grid or TIN ?





Digital terrain models: Grid or TIN ?

Grid

- Pros:
  - implicit topology
  - implicit geometry
  - simple algorithms
  - readily available in this form
- Cons:
  - uniform resolution ==> space waste
  - space becomes prohibitive as resolution increases

TIN

- Pros:
  - variable resolution
  - potentially space efficient
- Cons:
  - need to built and store topology
  - stored topology takes space
  - more complex programming (pointers..);

Digital terrain models: Grid or TIN ?

Consider a 300km by 300km area to be represented as a grid/raster. How big (how many points) is the grid at:

A. 100m resolution

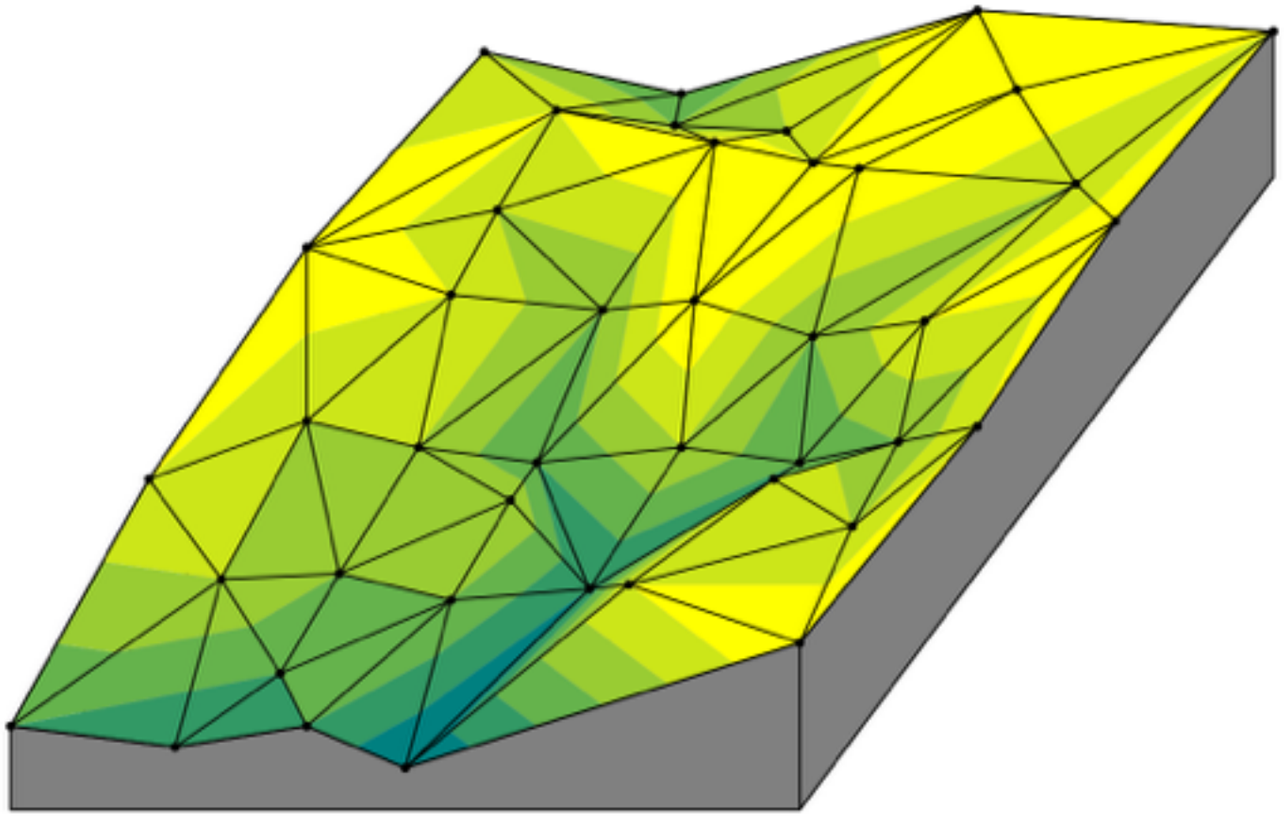
B. 10m resolution

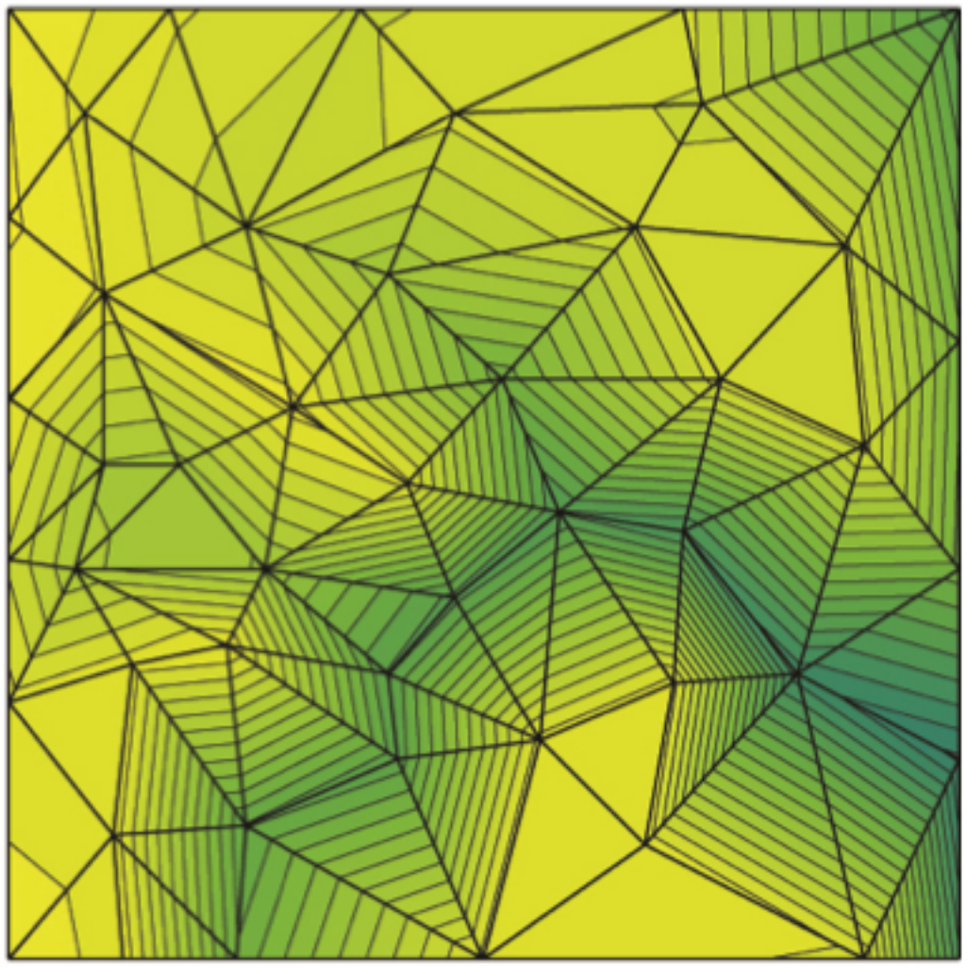
C. 1m resolution

Compute space requirement for a grid of 100x100=10,000 points; and for a TIN of 1000 points.

- How do we store a TIN ???

Data structures for TINs





- What do we expect to do on a TIN?
  - walk along an edge/triangle path
  - given an edge, find the two faces that are adjacent to this edge
  - walk along the boundary of a face (triangle)
  - find all edges and all triangles incident to a point

A good data structure for TINs should do these fast

Data structures for TINs

- Simplified versions of more general structures

Edge-based

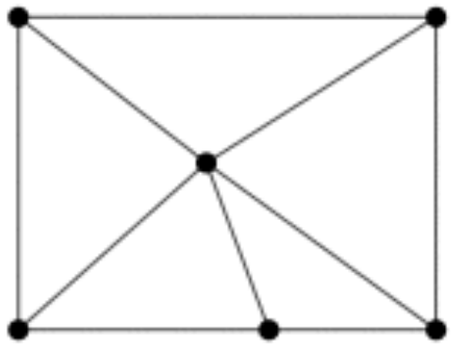
- arrays of vertices, edges, triangles
- every vertex stores:
  - its coordinates
- every edge stores:
  - 2 references to its adjacent vertices
  - 2 references to its adjacent triangles
- every triangle stores:
  - 3 references to its 3 edge

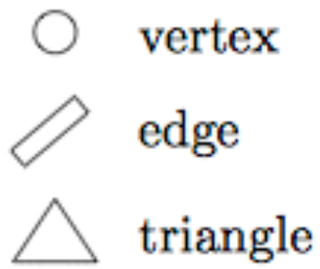
Triangle-based

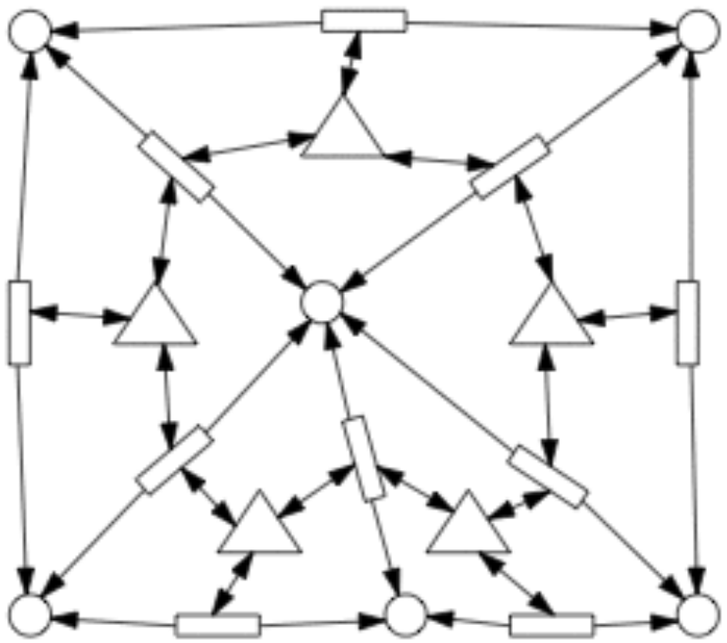
- arrays of vertices and triangles (edges are not stored explicitly)
- every vertex stores:
  - its coordinates
- every triangles stores:
  - 3 references to its incident vertices
  - 3 references to its adjacent triangles

- Note: CGAL uses triangle-based

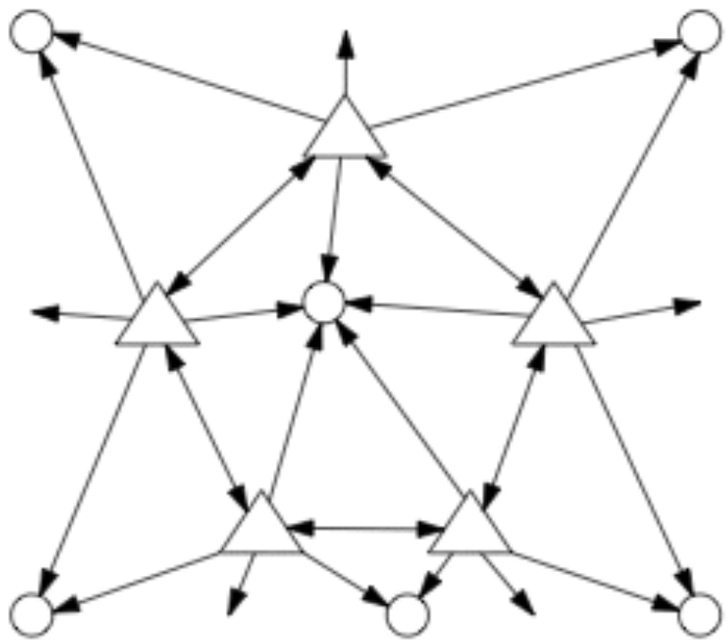
Data structures for TINs







edge-based



triangle-based

Assume we have a triangulation with n points. How much memory do we need to store it in a topological structure?

- edge-based
- triangle-based

Assume we have a triangulation with n points. How much memory do we need to store it in a topological structure?

- edge-based
- triangle-based

- The first issue is : what is the number of edges and triangles?
  - We can get bounds using Euler’s formula

Detour

Euler formula:

The following relation exists between the number of edges, vertices and faces in a connected planar graph: v - e + f = 2.

- Notes:
  - v-e+f -c =1 for c connected components
  - also true for any convex polyhedral surface in 3D
  - the Euler characteristic X= v - e + f is an invariant that describes the shape of space; it is X=2 for planar graphs, convex polyhedra,etc; can be extended to topological spaces.

n = nb of points in the TIN

Detour

- From Euler formula we know n-e+f = 2
- Furthermore, each triangle has 3 edges and each edge is in precisely 2 triangles (assuming the outside face as a triangle). This means 3f = 2e.
- We get:
  - the number of faces in a triangulation with n vertices is f = 2n-4
  - the number of edges in a triangulation with n vertices is e = 3n-6
- If the outside face is not triangulated it can be shown that
  - $e < 3n-6$, $f < 2n-4$
- Intuition: Given n points, the planar graph with largest number of edges and faces is a complete triangulation.

Theorem:

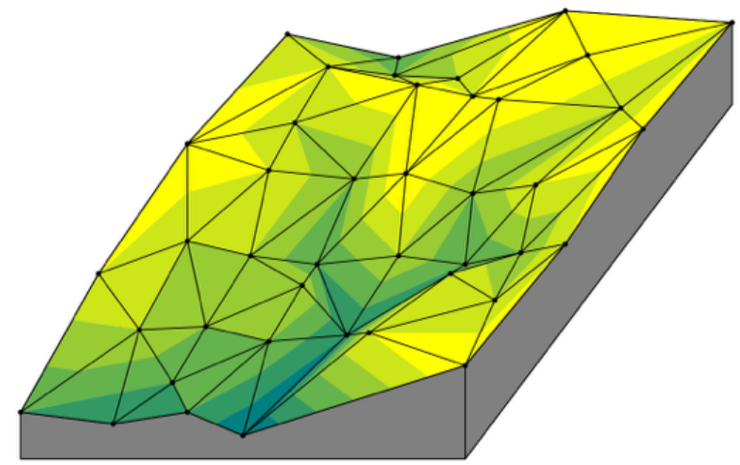
A triangulation with n vertices has at most 3n-6 edges and at most 2n-4 faces.

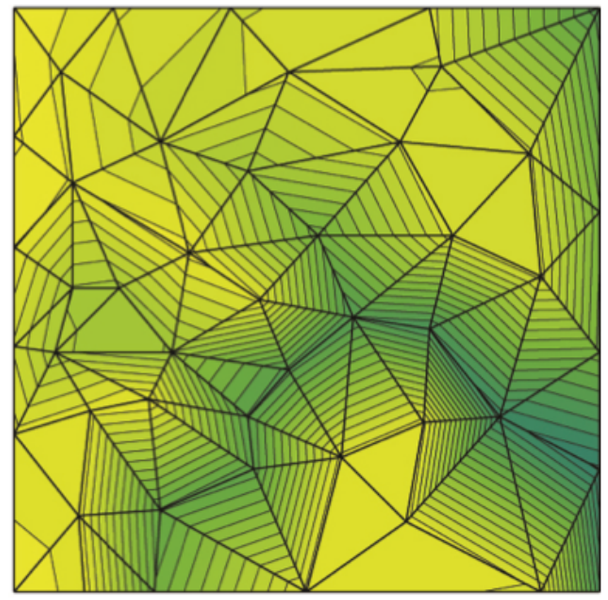
Assume we have a triangulation with n points. How much memory do we need to store it in a topological structure?

- edge-based
- triangle-based

Planar maps and meshes

Planar maps and meshes

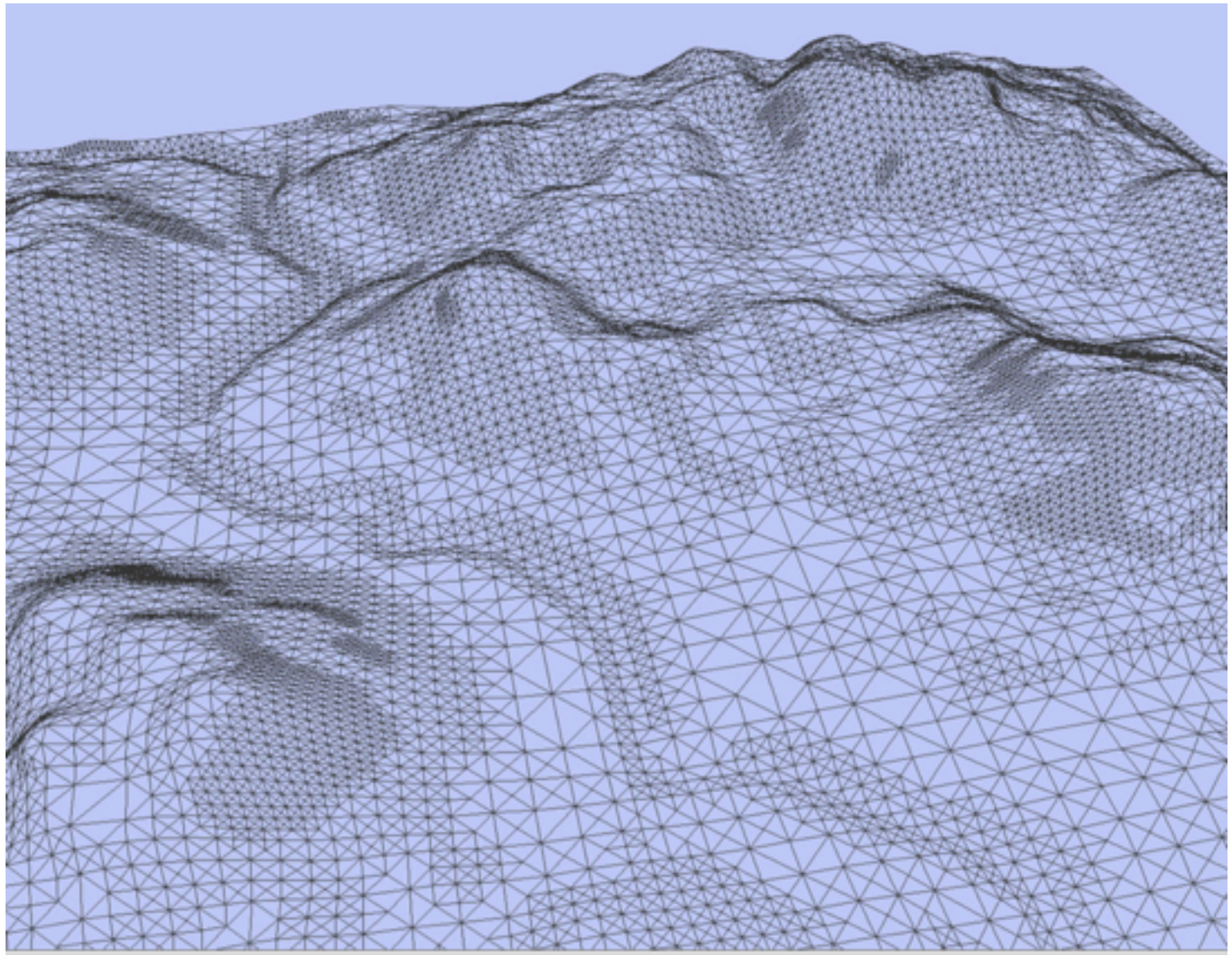




- Planar maps (2D) and terrain meshes (3D) are very similar.
  - the 2D projection of a terrain mesh is a planar map
- What is a good data structure for planar maps/meshes?

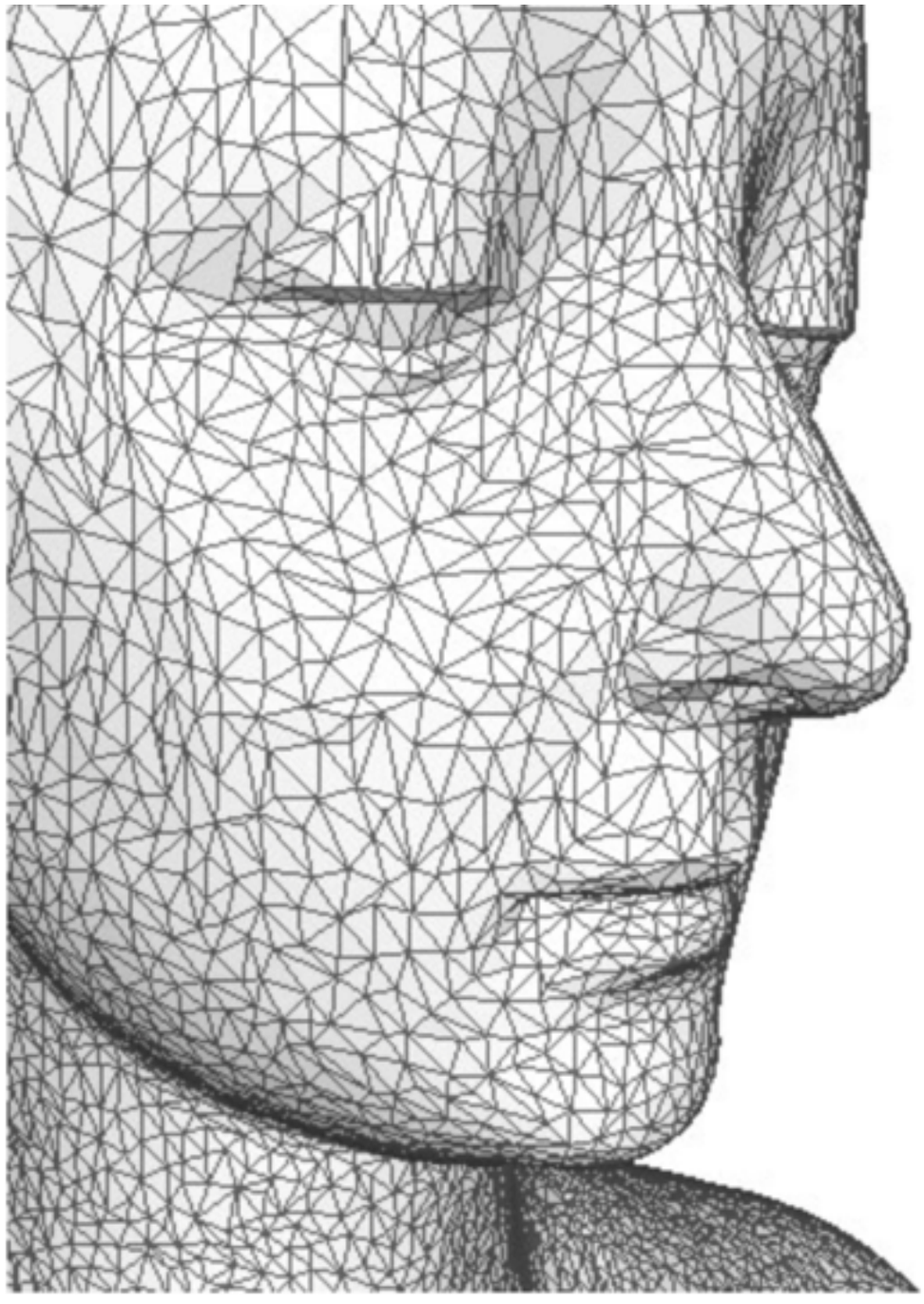
Detour: Meshing

- Big research area; Meshes used in modeling
- Big question:
  - Small angles cause numerical instability
  - Chose points to maximize minimum angle, trade off number of points with size of angle



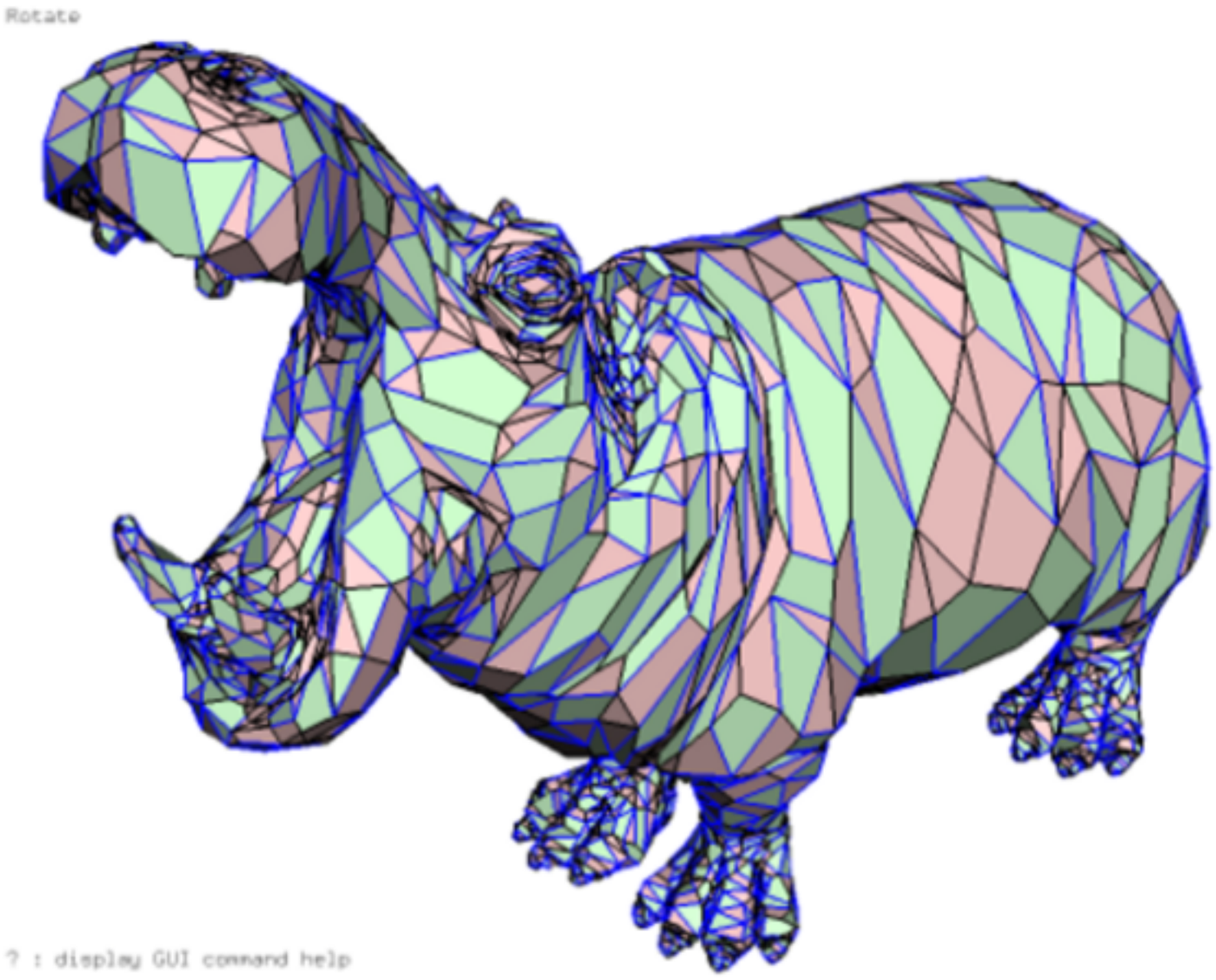
http://pre09.deviantart.net/bcb0/th/pre/f/2007/264/e/a/terrain_mesh_by_sordith.jpg

Meshes for arbitrary 3D surfaces



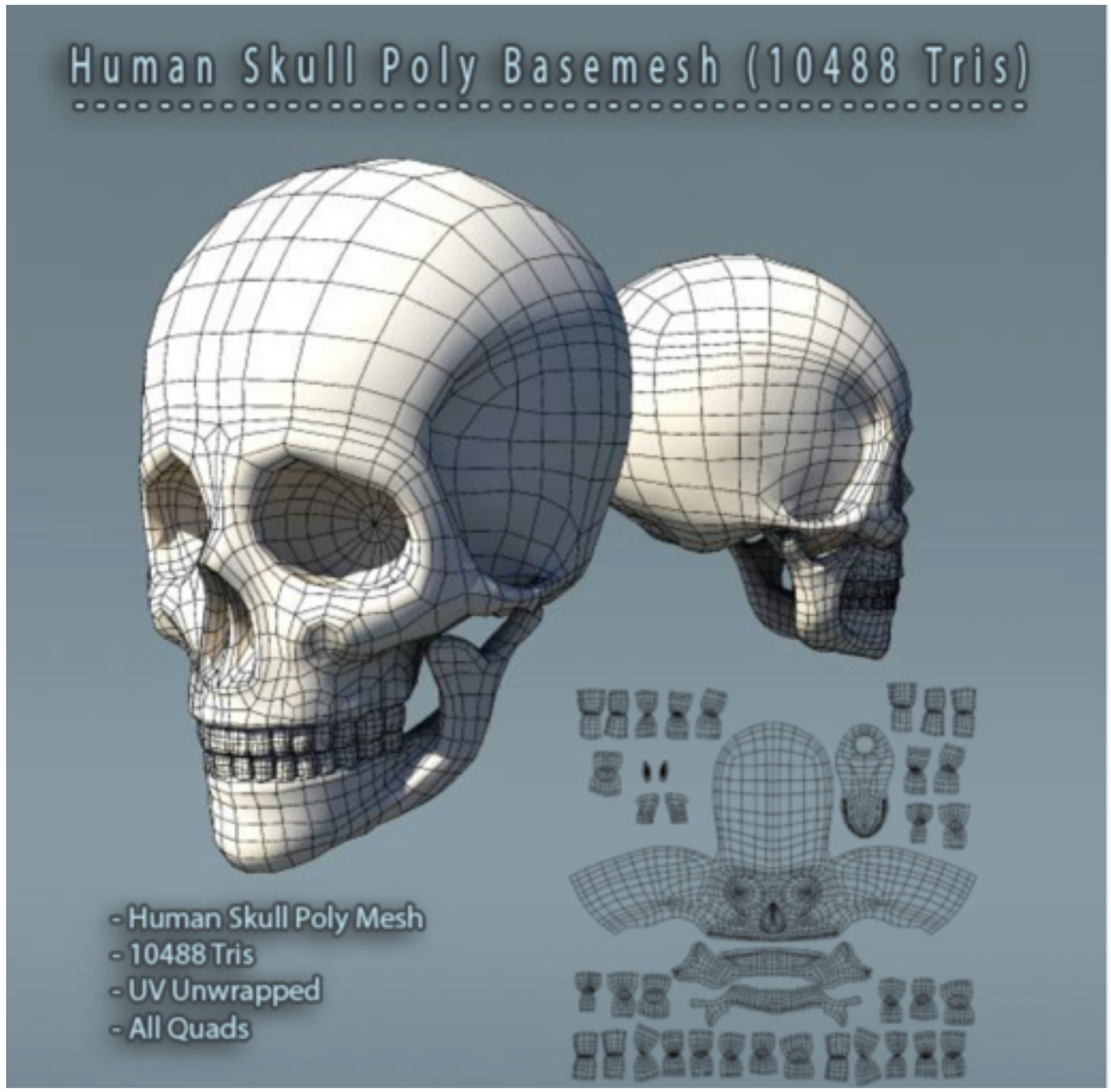
http://glasnost.itcarlow.ie/~powerk/GeneralGraphicsNotes/meshes/polygon_mesh_images/trike.jpg

Meshes for arbitrary 3D surfaces



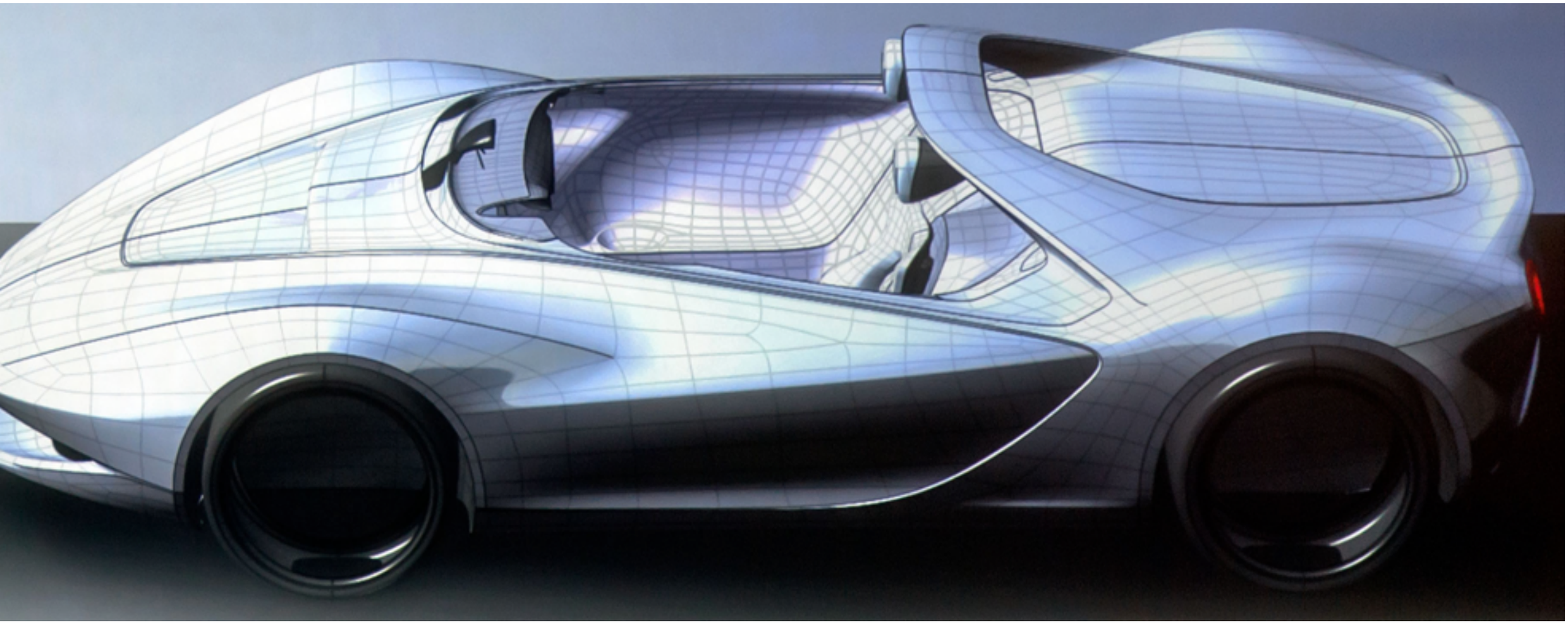
http://magsoft.dinauz.org/images/sourceCode/PPMCcap2.png

Meshes for arbitrary 3D surfaces



http://0.s3.envato.com/files/18578606/Skull_BaseMesh_590x590_JPG_PREVIEW.jpg

Meshes for arbitrary 3D surfaces



http://www.carbodydesign.com/media/2013/06/Pininfarina-Sergio-Concept-Polygonal-Mesh-3D-Model.jpg

Data structures for polygonal meshes

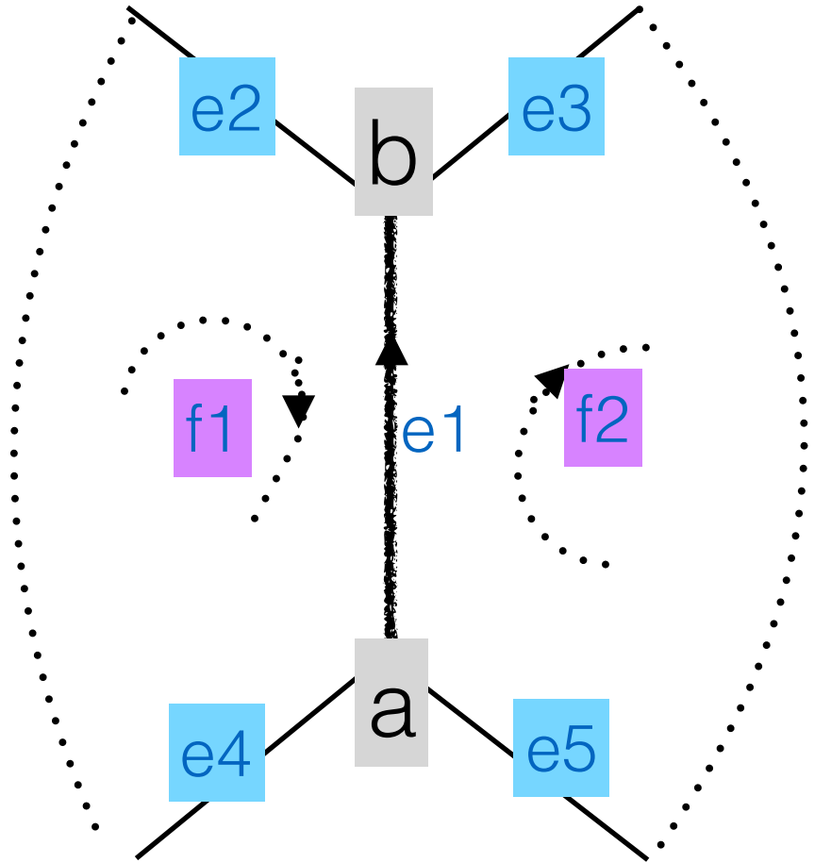
- First attempt:
  - list of vertices; each vertex stores in coordinates
  - list of face, each face storing pointers to its vertices

- Ok for some application (maybe), but does not store the topology
  - e.g. how do we walk from one face to another in this mesh?
  - which faces use this vertex?
  - which faces border this edge?
  - which edges border this face?
  - which faces are adjacent to this face?

Data structures for polygonal meshes

- Winged-edge data structure [Baumgart]
  - lists of vertices, edges and faces
  - each vertex stores:
    - its coordinates
    - a pointer to one edge incident to this vertex
  - each face stores:
    - a pointer to one edge along the boundary of this face
  - each edge stores
    - pointers to its two vertices
    - pointers to its left and right faces
    - predecessor and successor of this edge when traversing its left face
    - predecessor and successor of this edge when traversing its right face

Note: direction of an edge only used to establish left and right; each face oriented clockwise; the 4 edges are the *wings.*



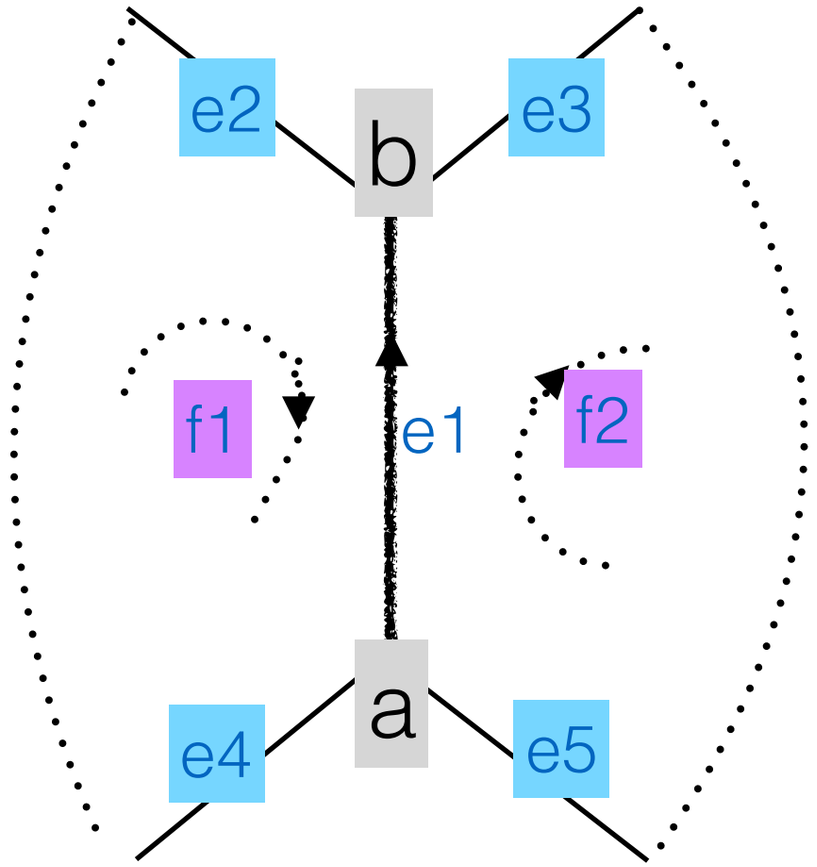
- Exercise: size..? how many bytes per edge?…total?

Winged-edge data structure

- Can answer adjacency queries
  - list edges and vertices on a face.
    - how?
  - list all edges incident on a vertex
    - how
  - are two faces adjacent?
    - how?
  - etc
- Not all queries are O(1)

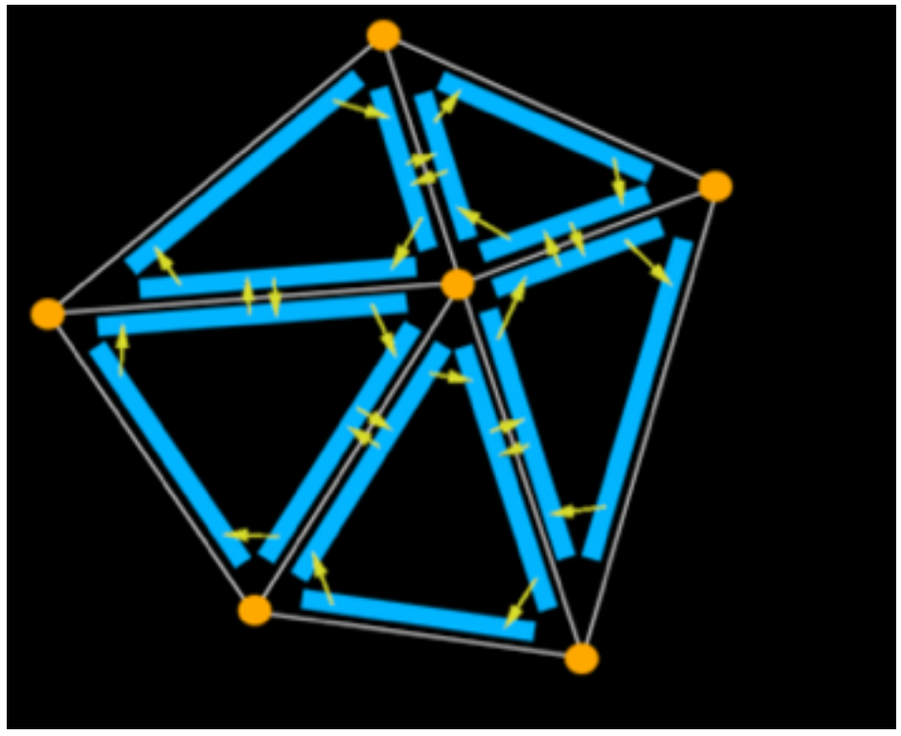
- Does not work for surfaces with holes
  - can be fixed e.g. by being careful how you orient the boundary…

- Links:
  - http://www.baumgart.org/winged-edge/winged-edge.html



Data structures for polygonal meshes

- Half-edge data structure
  - an edge = a pair of half edges
  - the half edges that border a face form a circular list
  - assume all faces are oriented the same way (say cw)
  - each vertex stores:
    - its coordinates
    - a pointer to exactly one half edge *starting* from this vertex
  - each face stores:
    - a pointer to one of the half-edges that borders it
  - each half-edge stores:
    - a pointer to the face it borders
    - a pointer to its endpoint
    - a pointer to twin half-edge
    - a pointer to next half-edge around the face

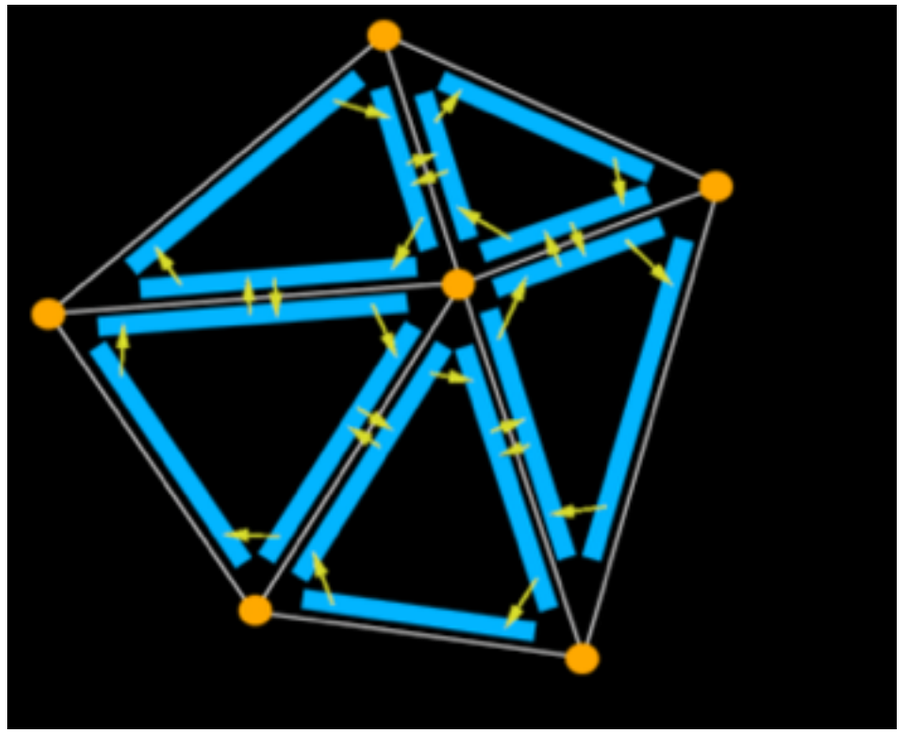


http://www.flipcode.com/archives/The_Half-Edge_Data_Structure.shtml

- Exercise: size..? how many bytes per edge?…total?

Half-edge data structure

```
//from http://www.flipcode.com/archives/The_Half-Edge_Data_Structure.shtml

struct HE_edge

{

        HE_vert* vert;   // vertex at the end of the half-edge
        HE_edge* pair;   // oppositely oriented adjacent half-edge
        HE_face* face;   // face the half-edge borders
        HE_edge* next;   // next half-edge around the face

};

struct HE_edge

{

        HE_vert* vert;   // vertex at the end of the half-edge
        HE_edge* pair;   // oppositely oriented adjacent half-edge
        HE_face* face;   // face the half-edge borders
        HE_edge* next;   // next half-edge around the face

};

 struct HE_vert {

        float x;
        float y;
        float z;

        HE_edge* edge;  // one of the half-edges emantating from the vertex

};

struct HE_face {

        HE_edge* edge;  // one of the half-edges bordering the face

};
```



http://www.flipcode.com/archives/The_Half-Edge_Data_Structure.shtml

Half-edge data structure

Can answer adjacency queries:

- Find the vertices of a half-edge e

e->vert, and e->twin->vert

- Find the two faces that border half-edge e

e->face, e->twin->face

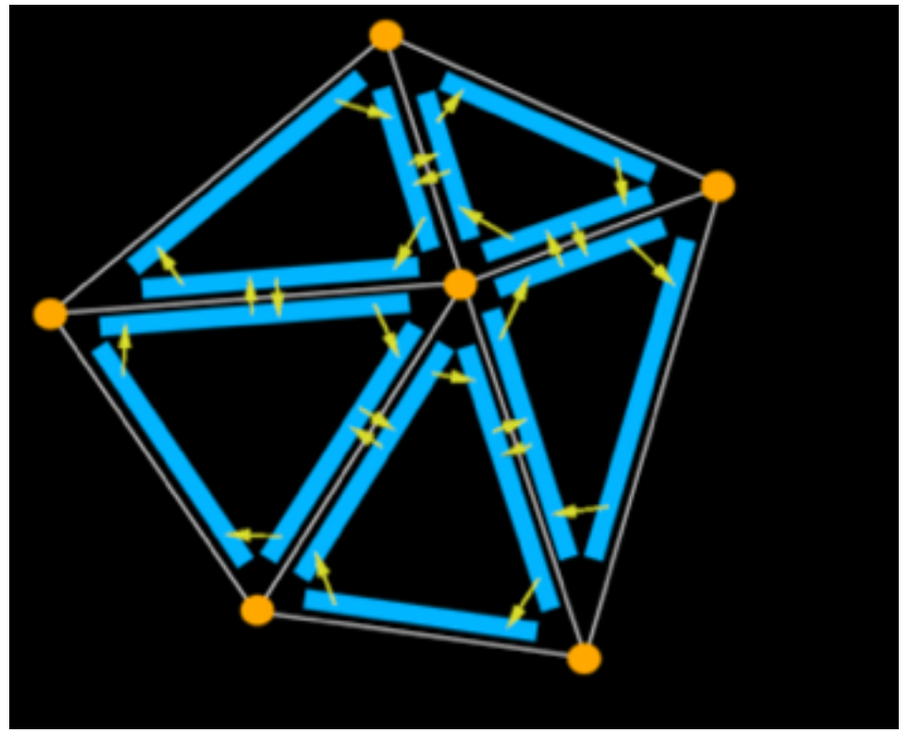
- Iterate the edges along a face f

```
e = f->edge
do {
        …
        e = e->next
} while (e != f->edge)
```

- Iterate over the edges adjacent to a vertex v

```
e = v->edge;
do {
        …
        e = e->next
} while (e != v->edge)
```

- are two faces adjacent?
- are two vertices adjacent?
- what are all faces that use this vertex ?



Topological structures for polygonal meshes

- Can answer basic adjacency queries fast (O(1) if possible)
- But
  - use a lot of space !!!
  - need to be constructed from raw data
  - involve more complex programming

Spatial data terminology

Vector model

Data stored as points, lines and polygons in an appropriate data structure

Raster model

Data stored as grid of values

networks

planar maps

terrains