Algorithms for GIS

Spatial data: Models and representation

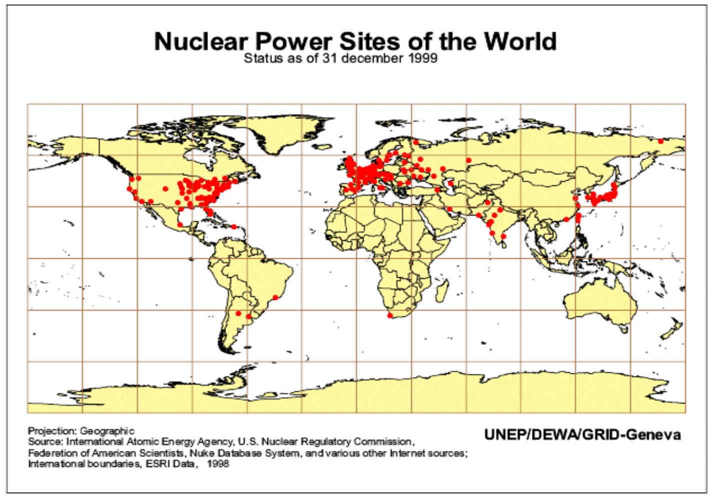
(part II)

Laura Toma

Bowdoin College

Spatial data representation

points


Nuclear Power Sites of the World
Status as of 31 december 1999
Projection: Geographic
Source: International Atomic Energy Agency, U.S. Nuclear Regulatory Commission, Federation of American Scientists, Nuke Database System, and various other Internet sources; International boundaries, ESRI Data, 1998
UNEP/DEWA/GRID-Geneva

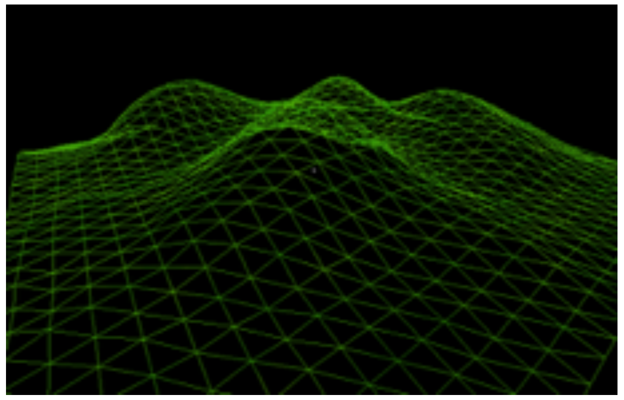
networks

graphs

planar maps

topological data structures (winged edge, half edge)

terrains



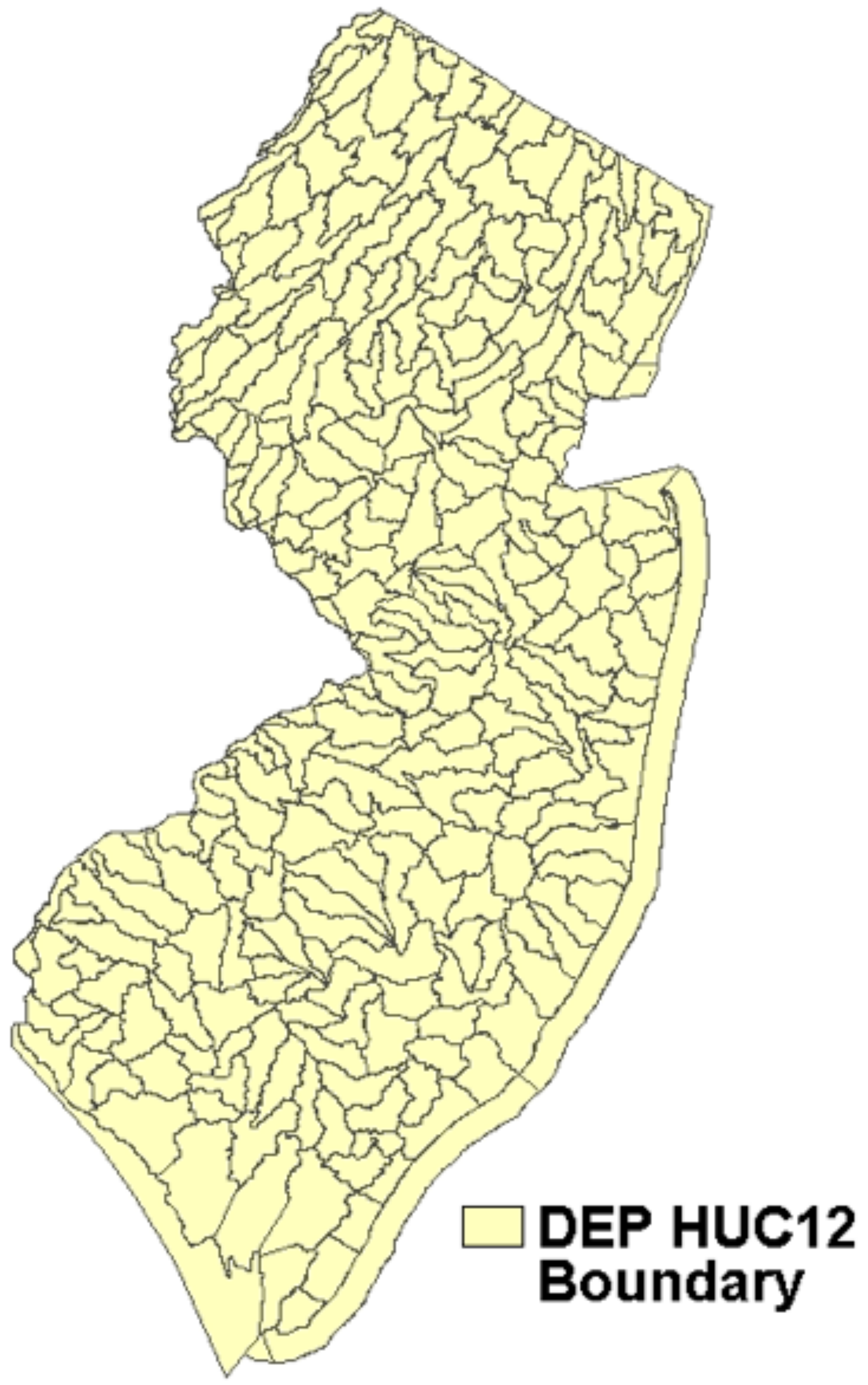
raster or TIN

- Last time we discussed networks and terrains
- Today we look at planar maps

Planar maps

digital hydrologic unit boundary layer (clipped at New Jersey political boundaries)

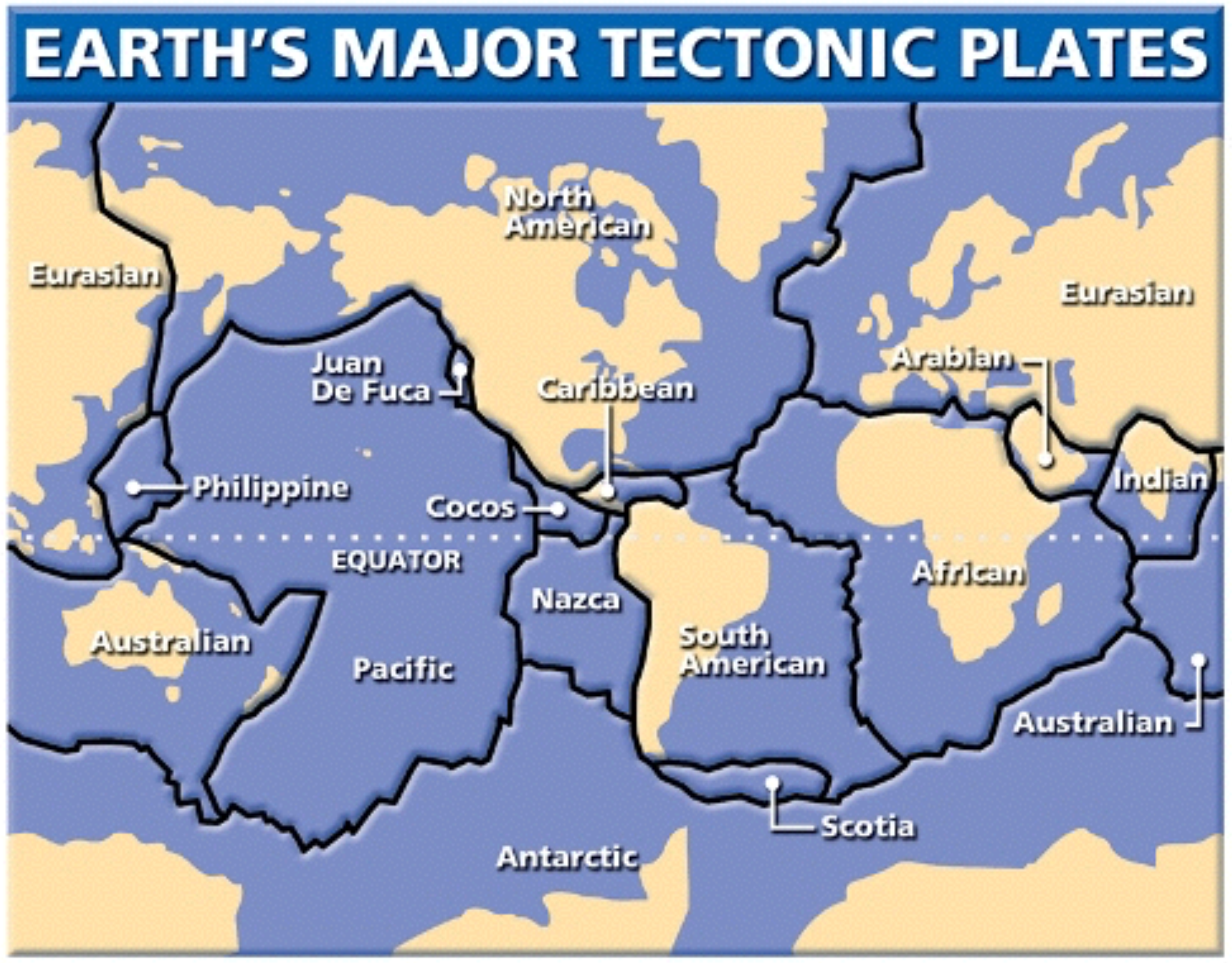
www.state.nj.us



Planar maps

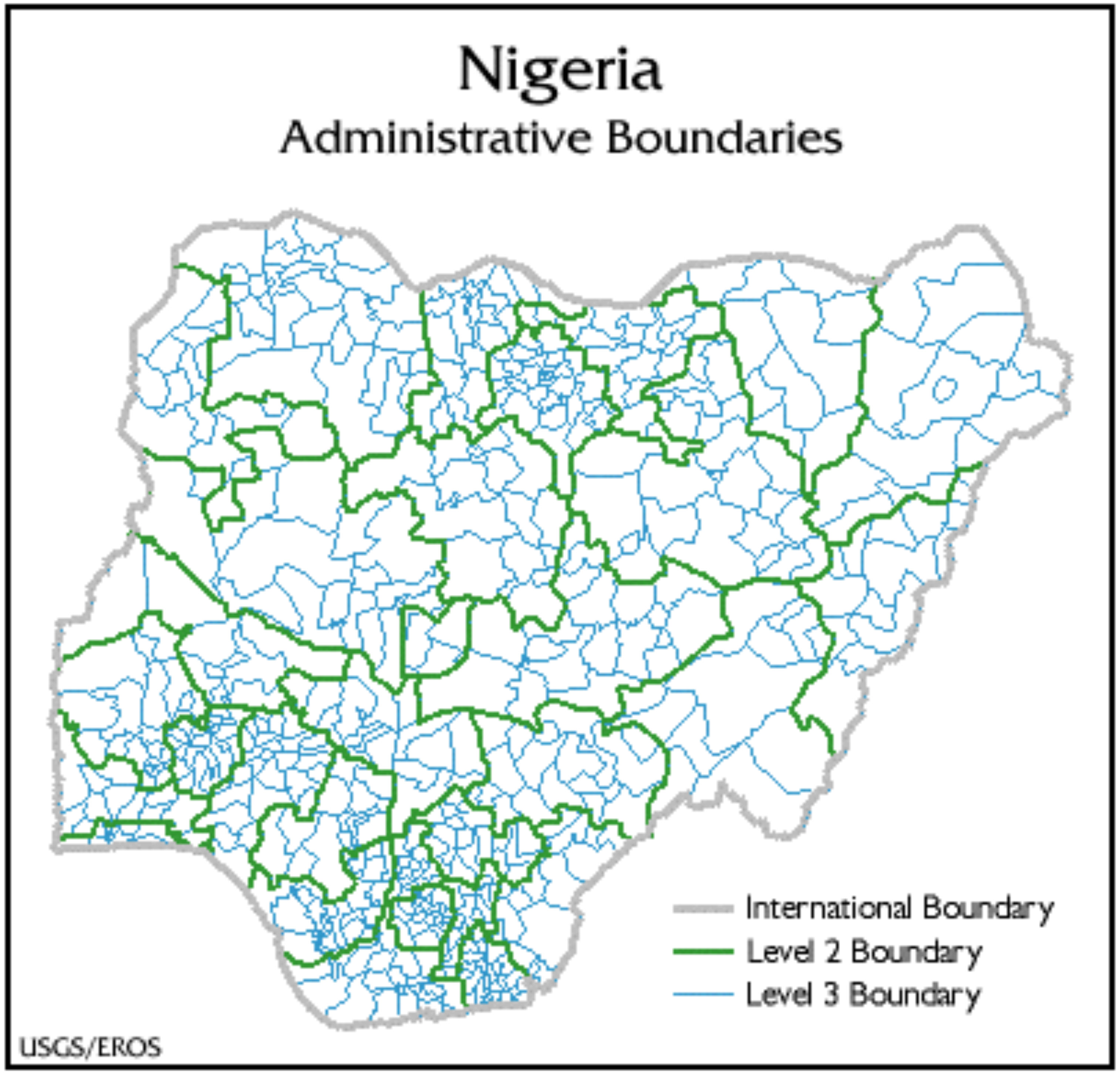
cadastral data

Planar maps

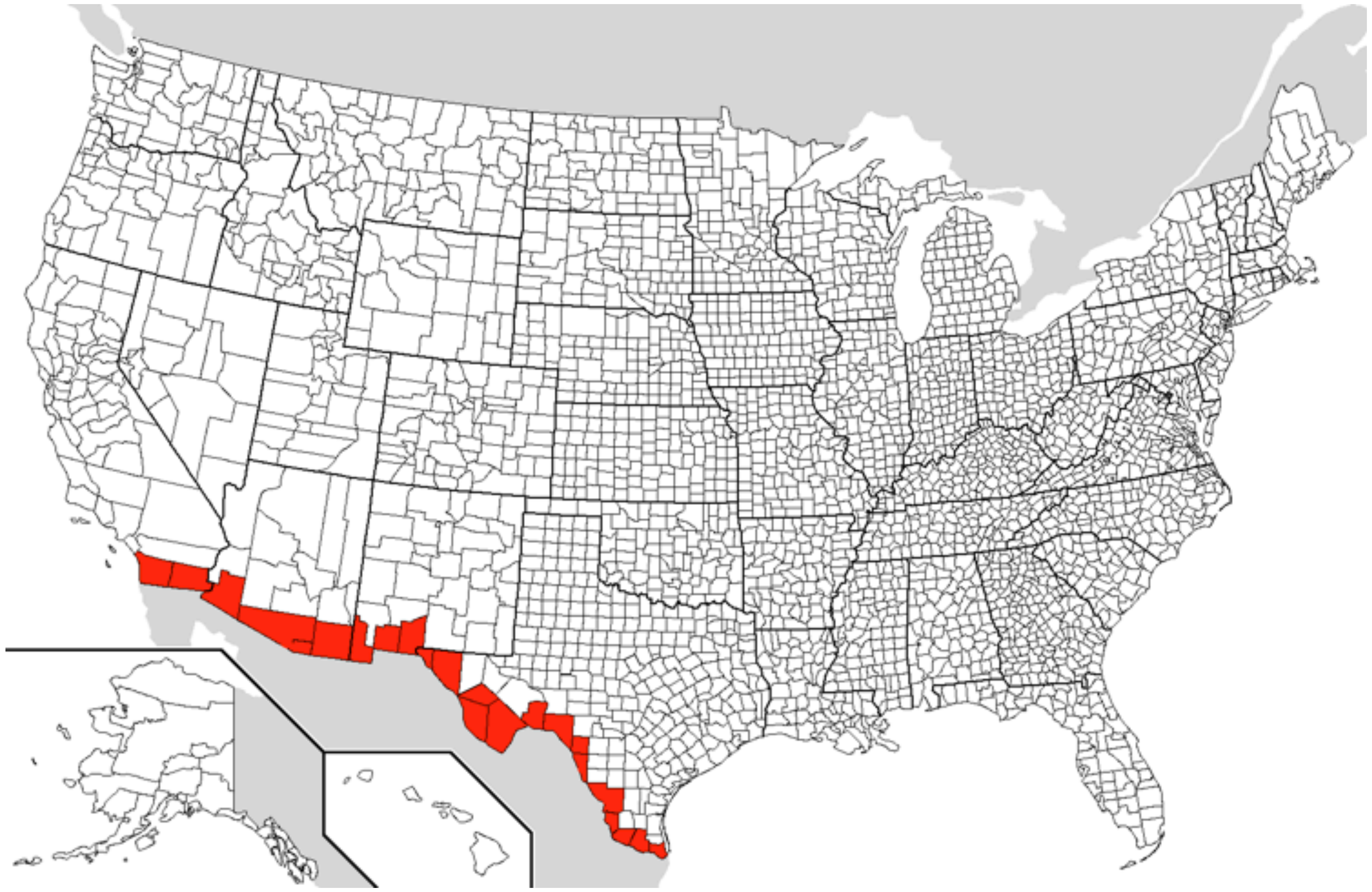


http://www.myinterestingfacts.com/wp-content/uploads/2014/11/Plate-Tectonics-Facts.jpg

Planar maps


Nigeria
Administrative Boundaries
International Boundary
Level 2 Boundary
Level 3 Boundary
USGS/EROS

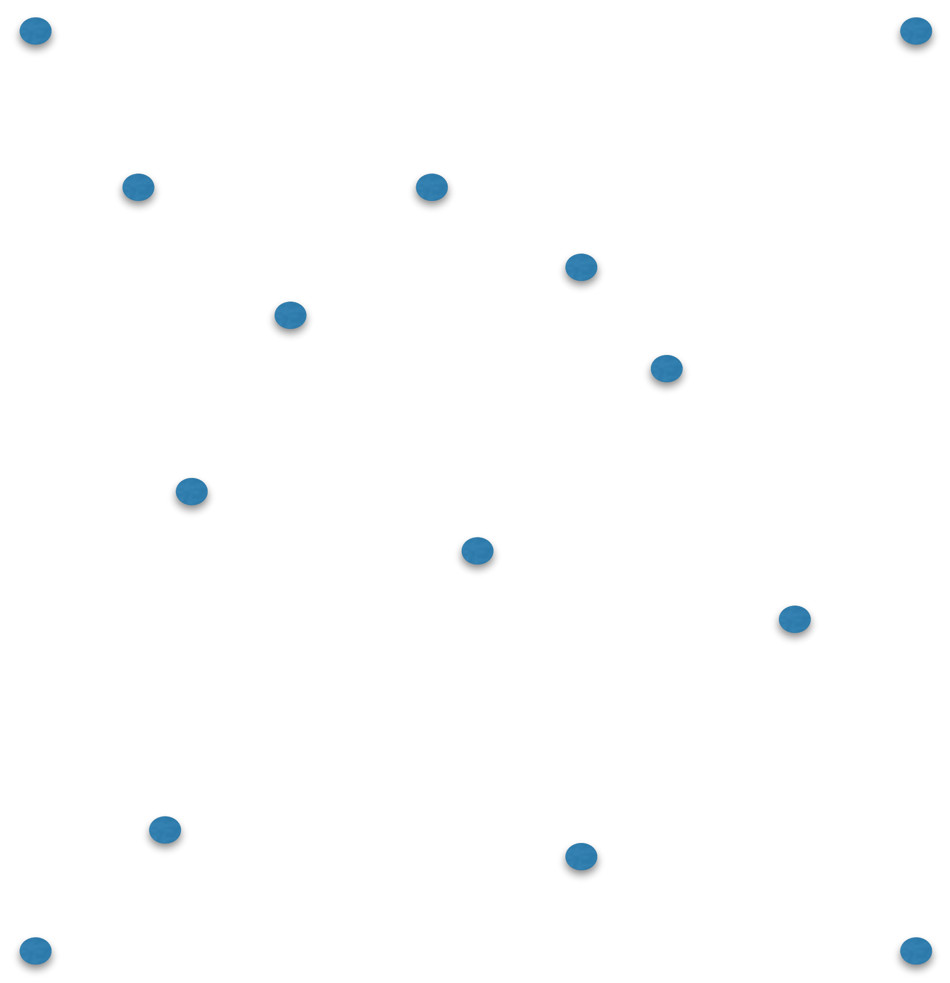
Planar maps



http://mapsof.net/uploads/static-maps/mexico_us_border_counties.png

Planar maps

- **Triangulations**: a special type of planar map
- Let P be a set of points in the plane.
- A triangulation T(P) is a partition of the plane into regions such that all regions are triangles.

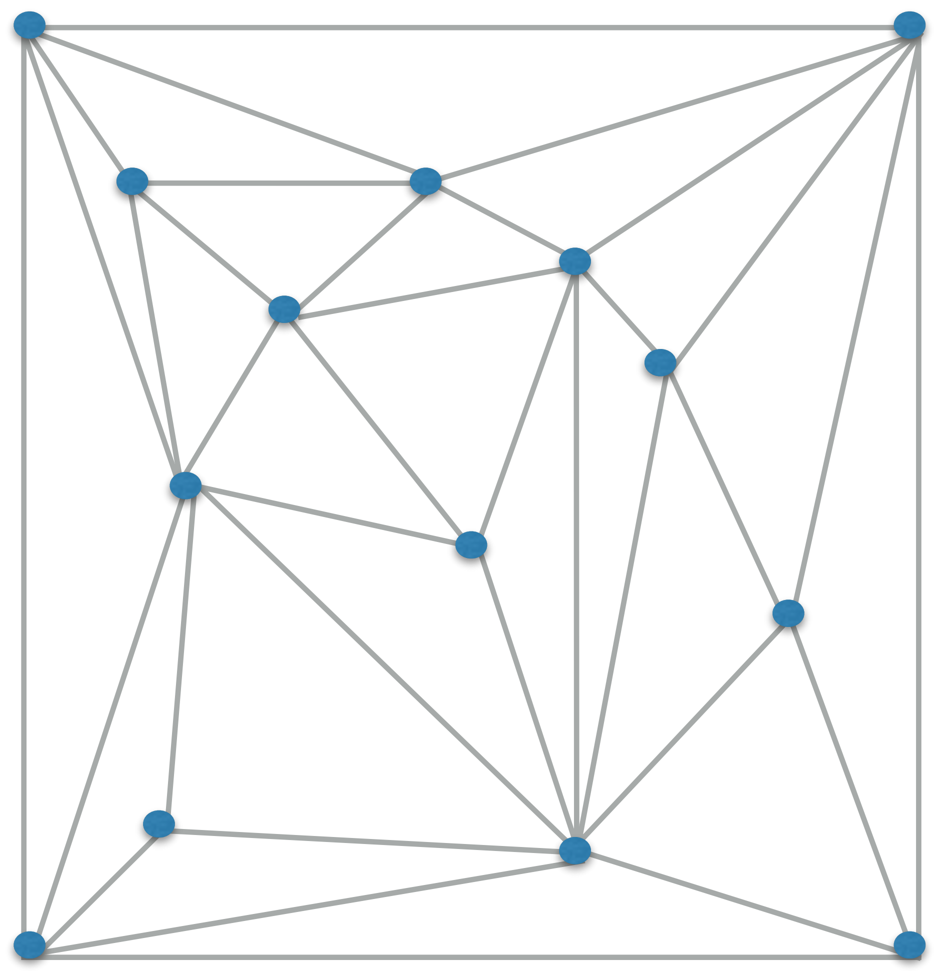


Planar maps

- **Triangulations**: a special type of planar map

- Let P be a set of points in the plane.

- A triangulation T(P) is a partition of the plane into regions such that all regions are triangles.

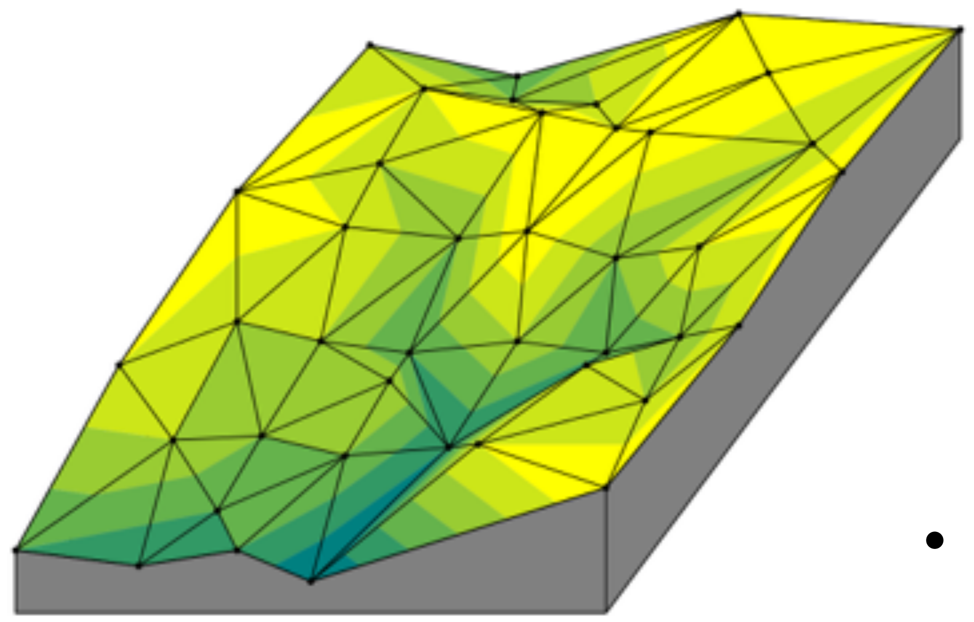


one possible triangulation of P

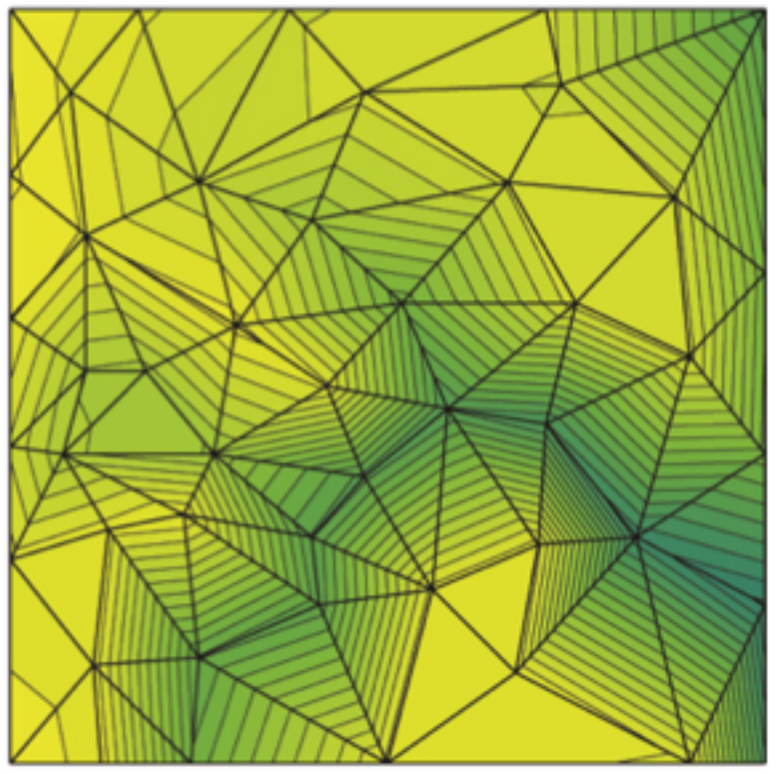
Data structures for planar maps

- What do we expect to do on a map?
    - walk from edge to a neighboring edge
    - walk from face to a neighbor face
    - walk along the boundary of a face
    - …
- In order to be efficient, a data structure must store the topology
- Studied in the context of **meshing**
    - triangulations vs TIN terrains (triangular meshes)
    - planar maps vs polygonal meshes

triangular terrain meshes



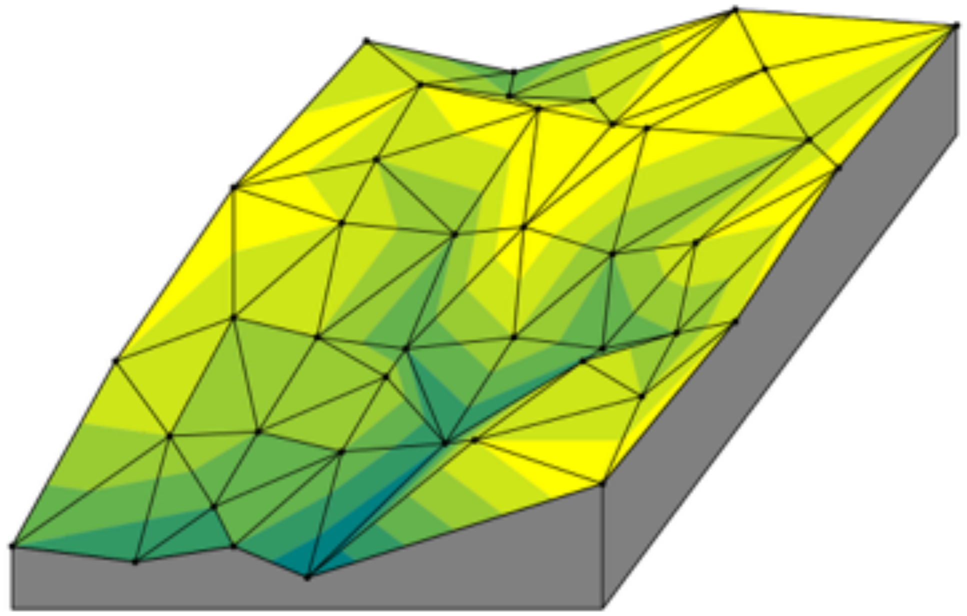
- The 2D projection of a triangulated terrain is a triangulation.

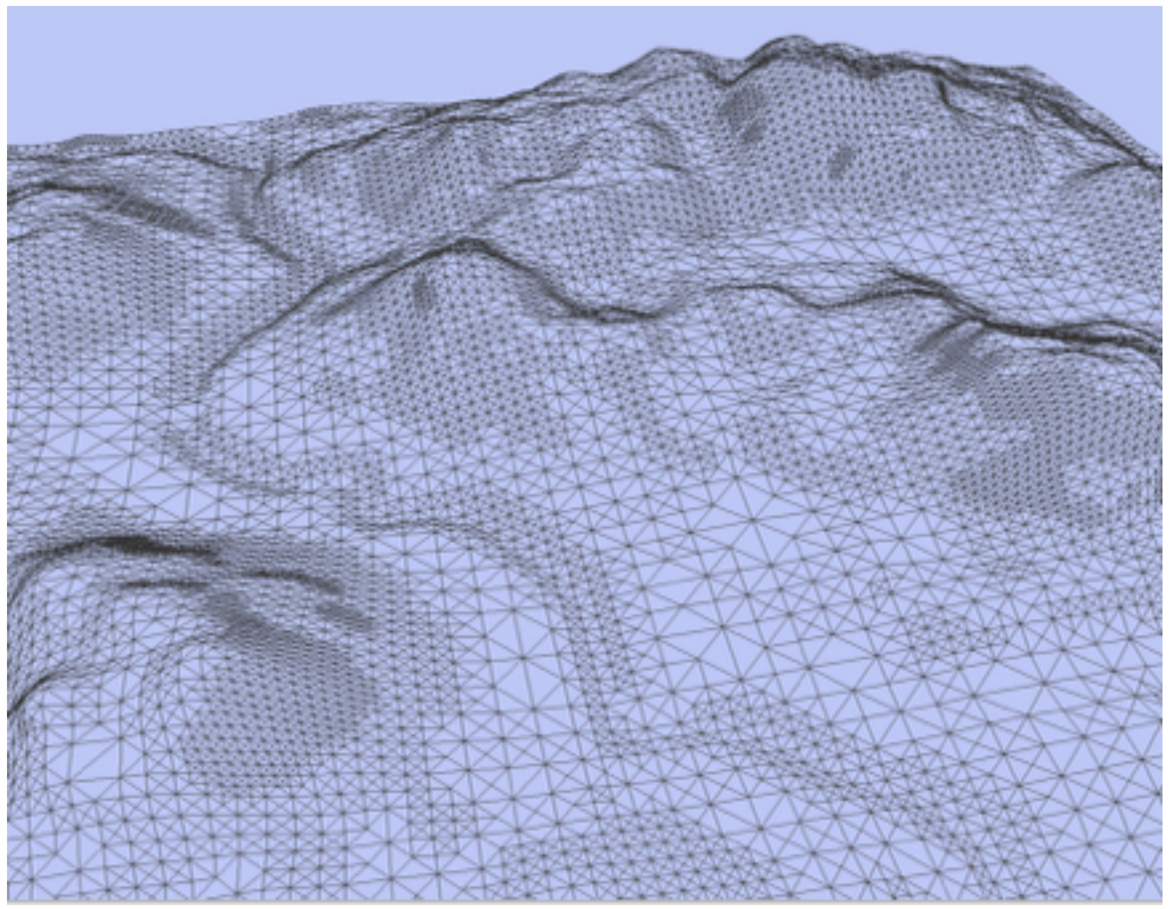


triangulations

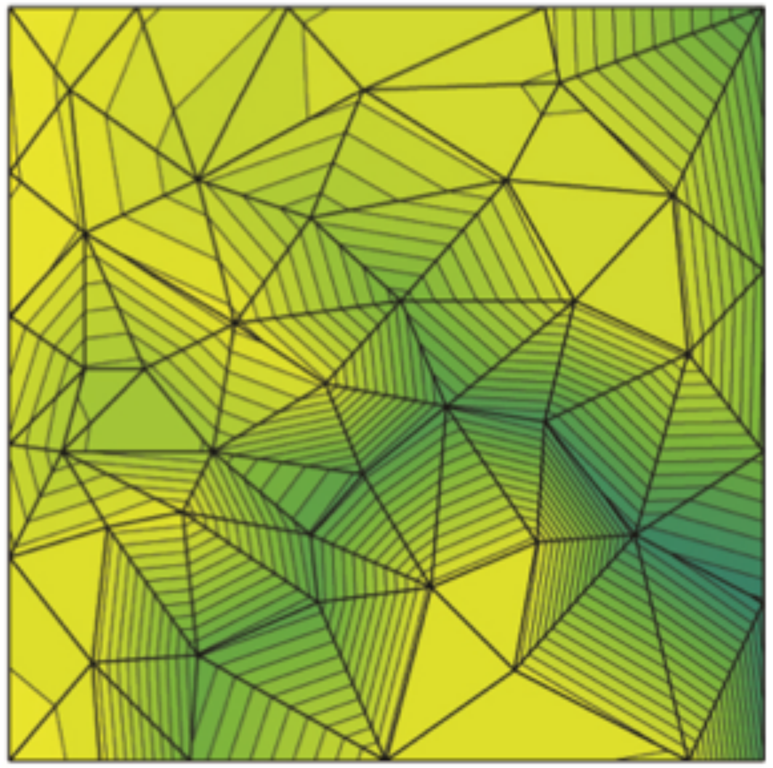
terrain meshes

triangular terrain meshes

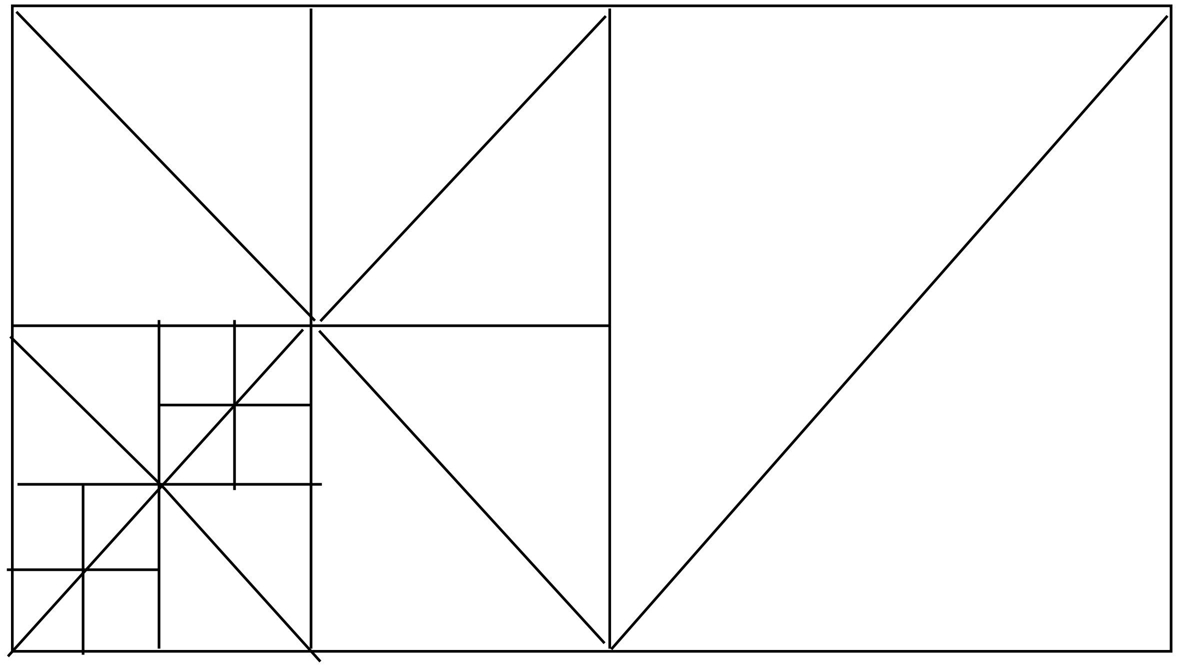




http://pre09.deviantart.net/bcb0/th/pre/f/2007/264/e/a/terrain_mesh_by_sordith.jpg



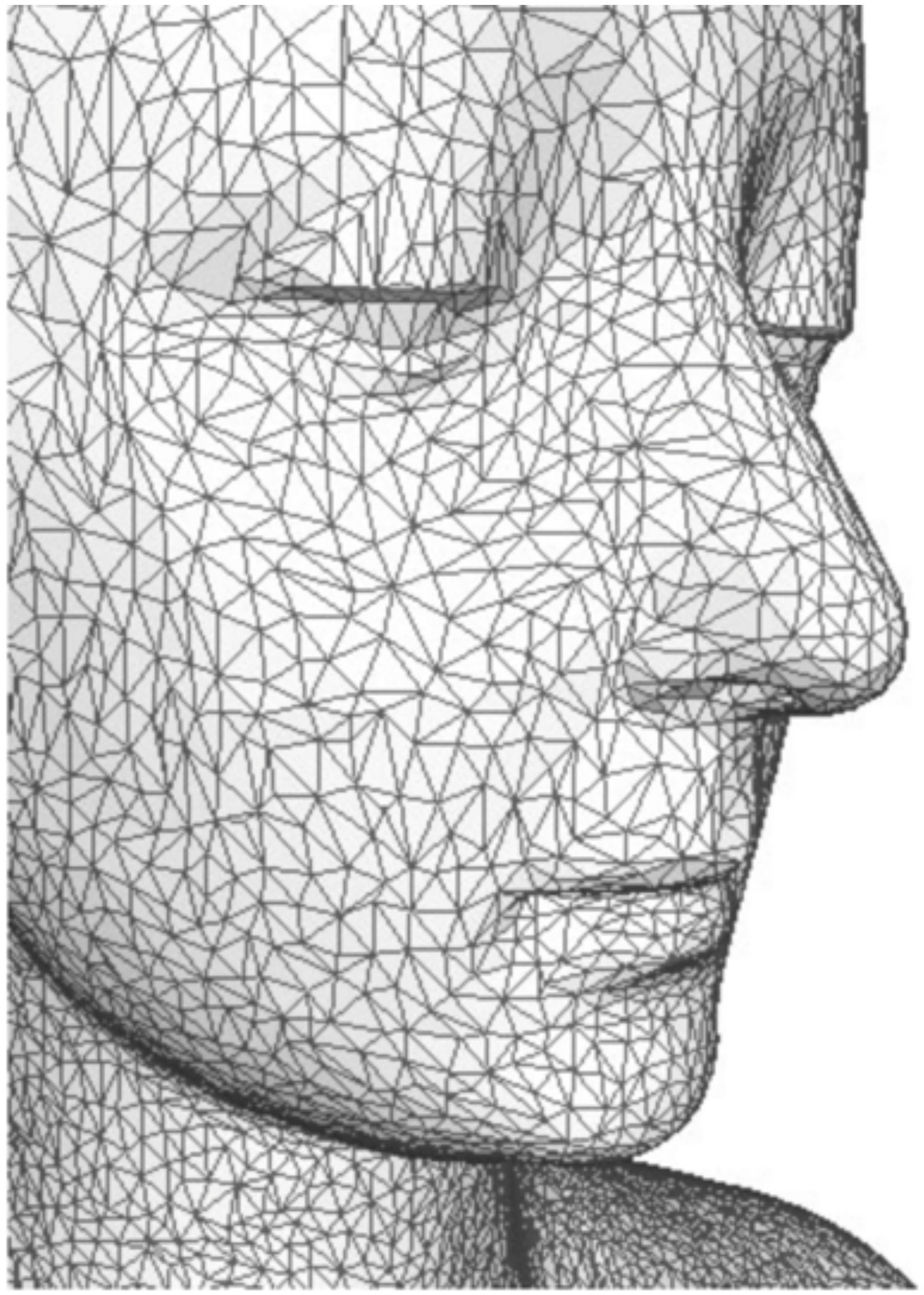
triangulations



planar maps

Meshing

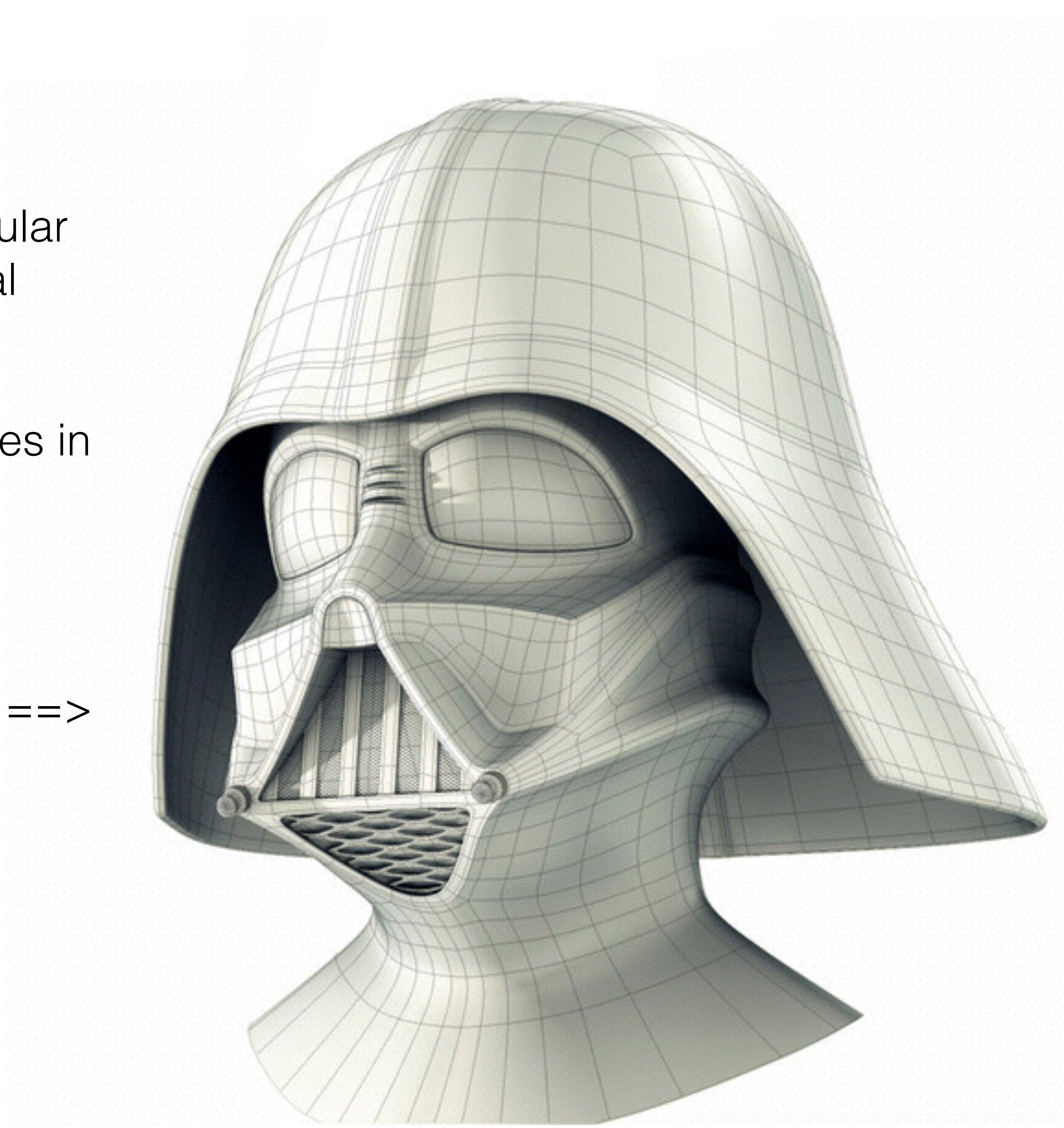
- Meshes are not necessarily triangular
  - rectangular meshes, hexagonal meshes, ..
- Used to represent arbitrary surfaces in 3D (not necessary terrains)
- Big research area



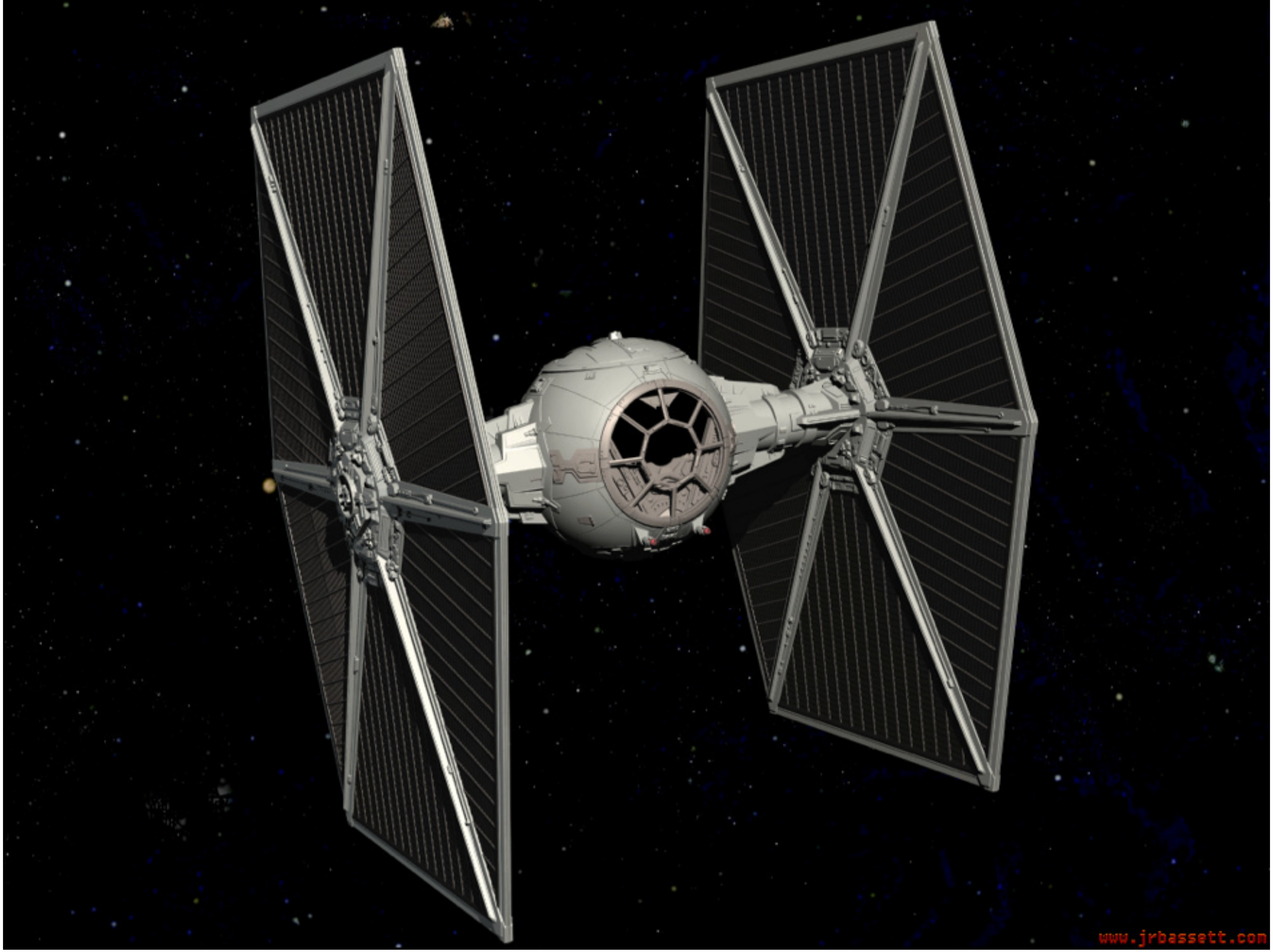
http://glasnost.itcarlow.ie/~powerk/GeneralGraphicsNotes/meshes/polygon_mesh_images/trike.jpg

Meshing

- Meshes are not necessarily triangular
  - rectangular meshes, hexagonal meshes, ..
- Used to represent arbitrary surfaces in 3D (not necessary terrains)
- Big research area
- On large meshes speed is critical ==> need efficient data structures

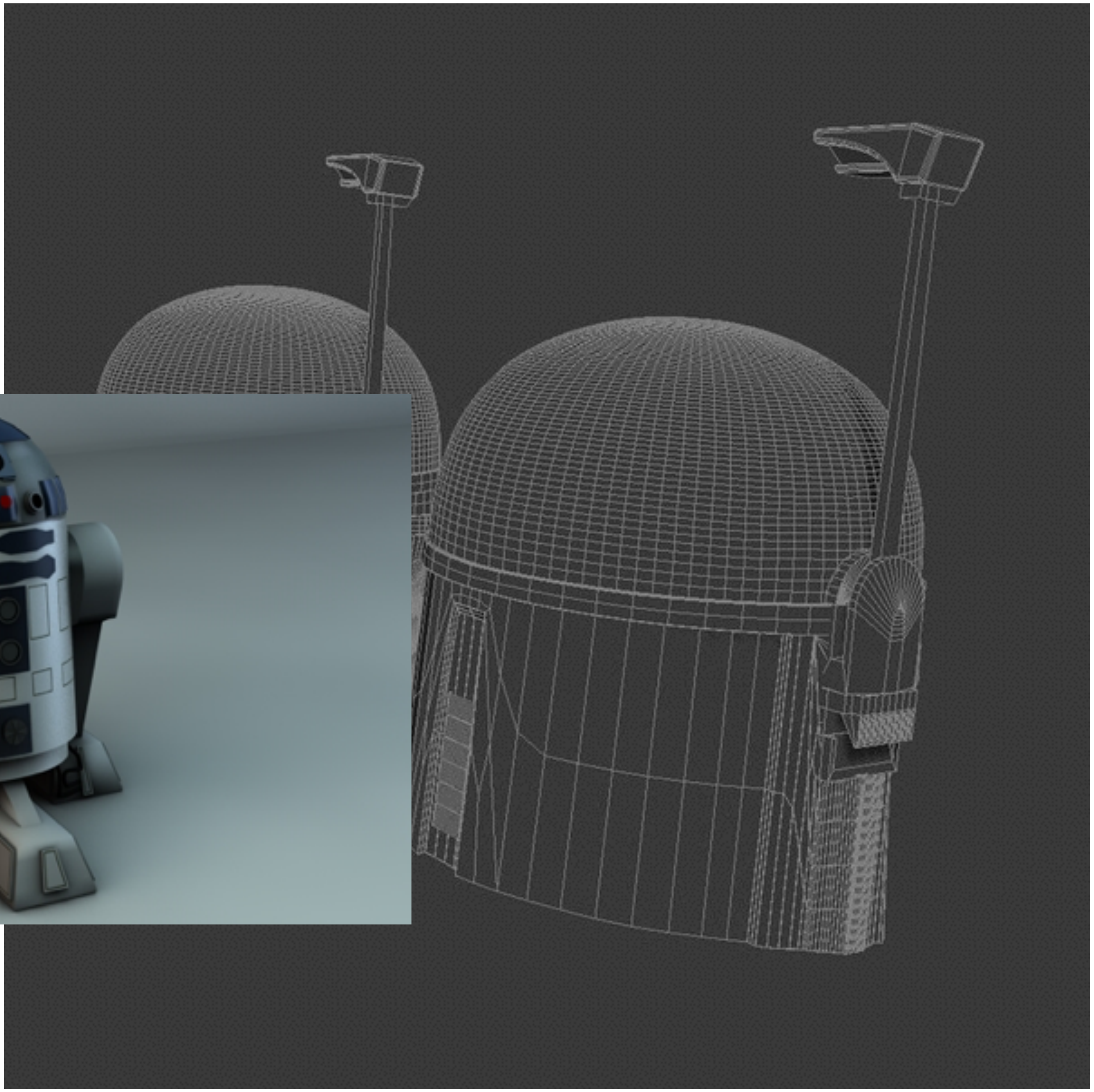


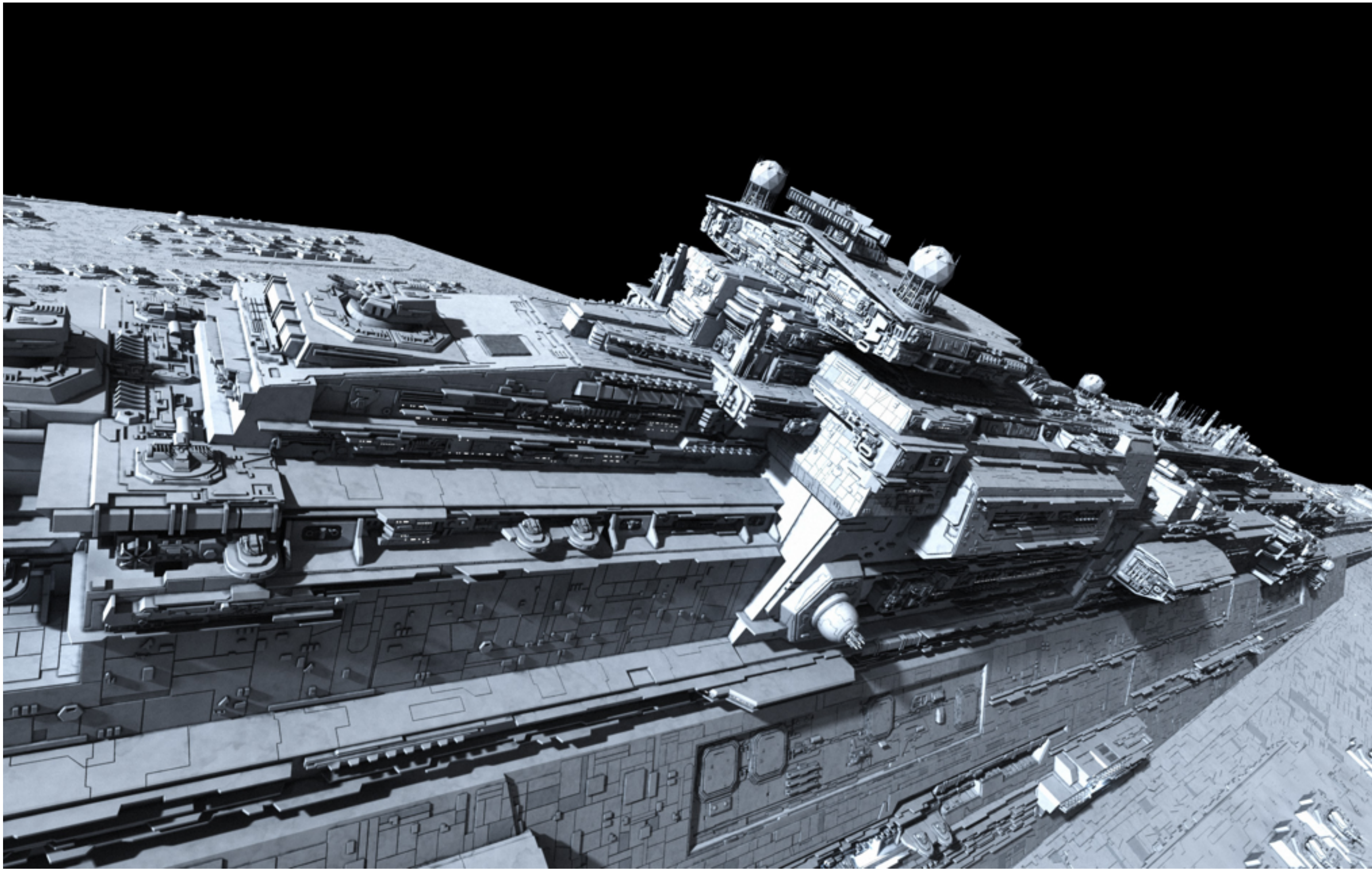
http://previewcf.turbosquid.com/Preview/2014/05/25__02_45_53/Vader_Wireframe.jpg5a0d1bb5-8efa-406f-bd61-31f60014f57eLarge.jpg



TIE fighter http://www.jrbassett.com/images/TFg3d0.JPG

https://0.s3.envato.com/files/316771.jpg


http://previewcf.turbosquid.com/Preview/2014/07/07__15_44_27/mesh%202.jpg42209060-b29e-4876-8cd3-8ce8339529e4Large.jpg



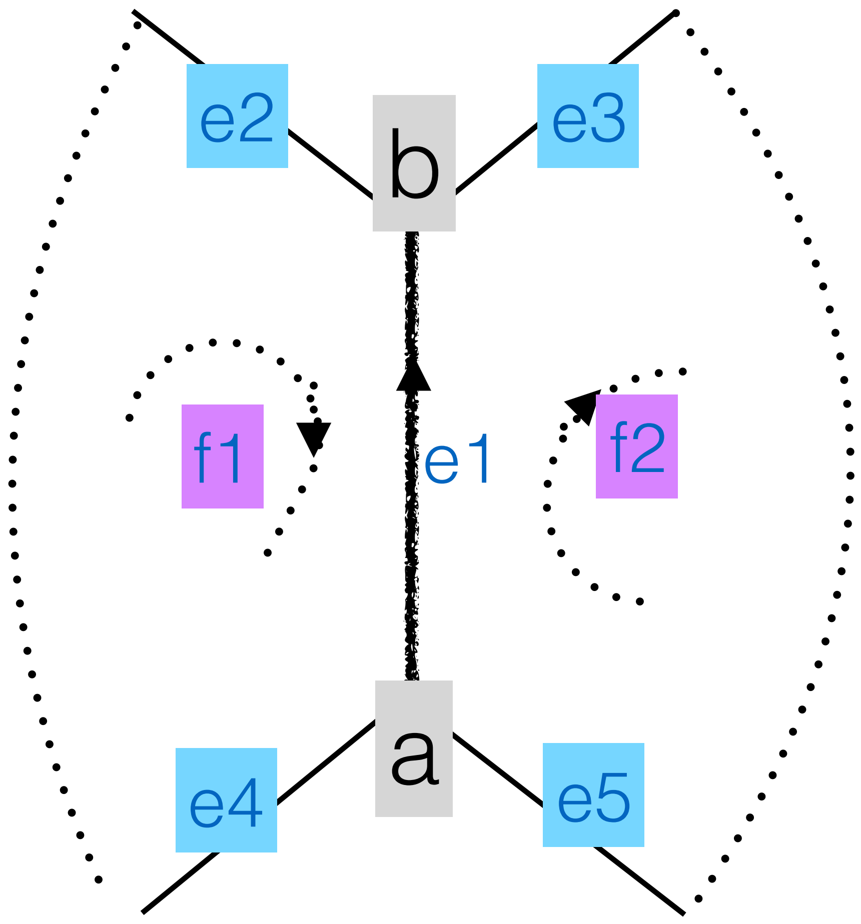
Data structures for polygonal meshes

- 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?

Winged-edge data structure

- 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*.

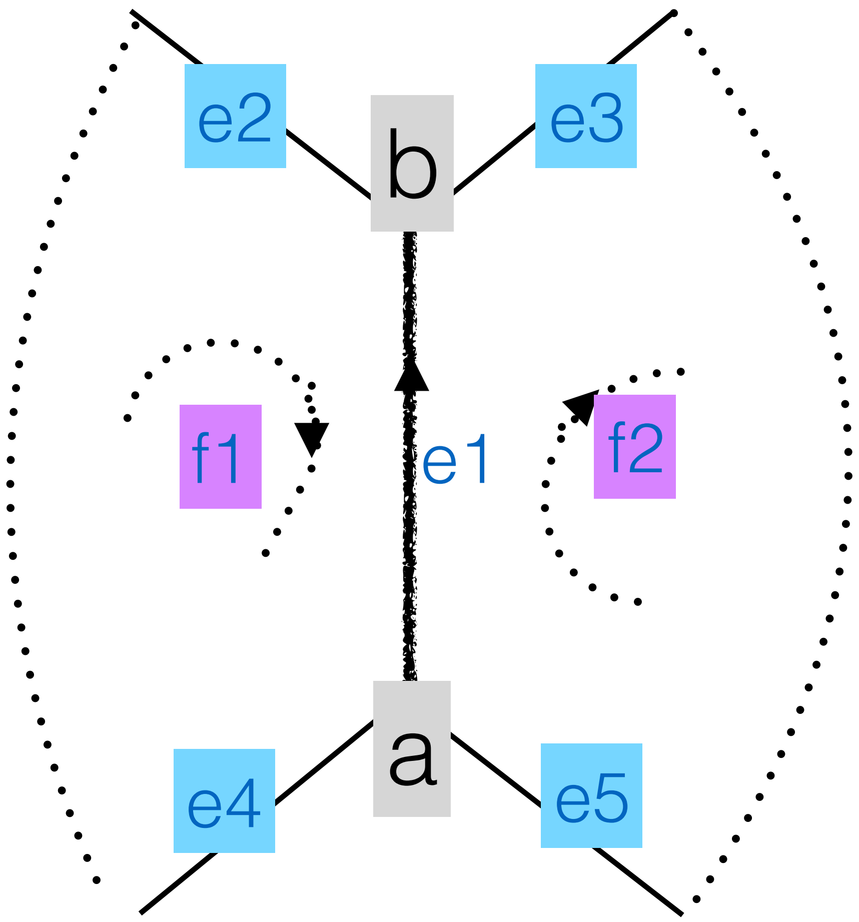


- Questions: size..? how many bytes per edge?…total?

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)

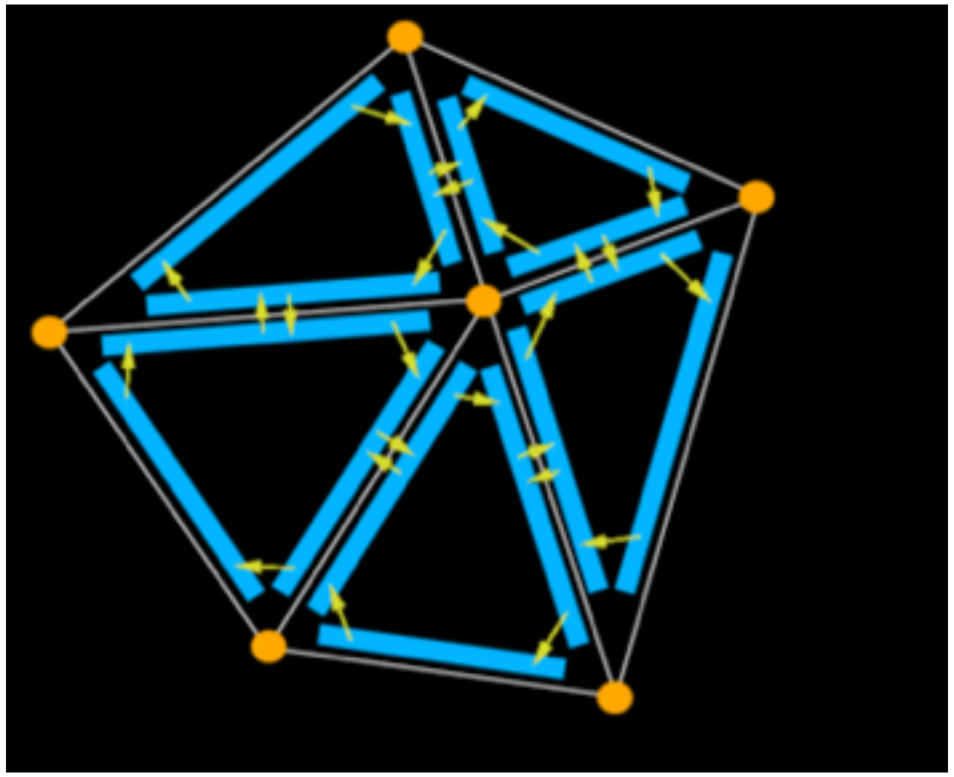
- Does not work for surfaces with holes
  - can be fixed e.g. by being careful how you orient the boundary…
- Links:
  - http://www.baumgart.org/winged-edge/winged-edge.html



Half-edge data structure

- 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 (eg 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

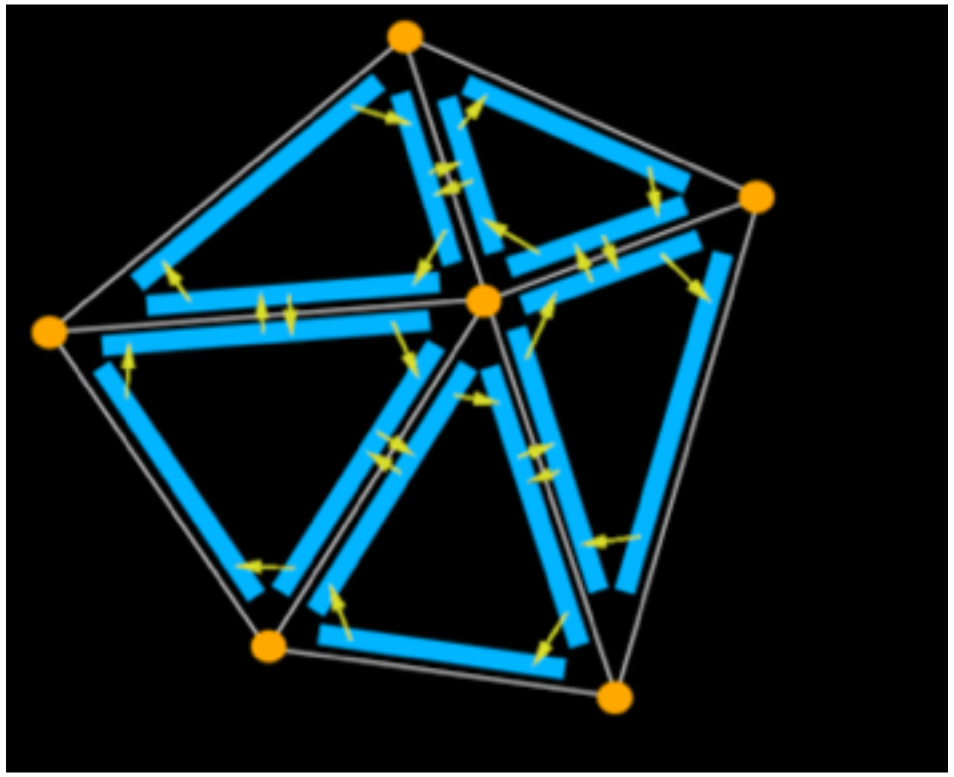


http://www.flipcode.com/archives/The_Half-Edge_Data_Structure.shtml

- Questions: 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

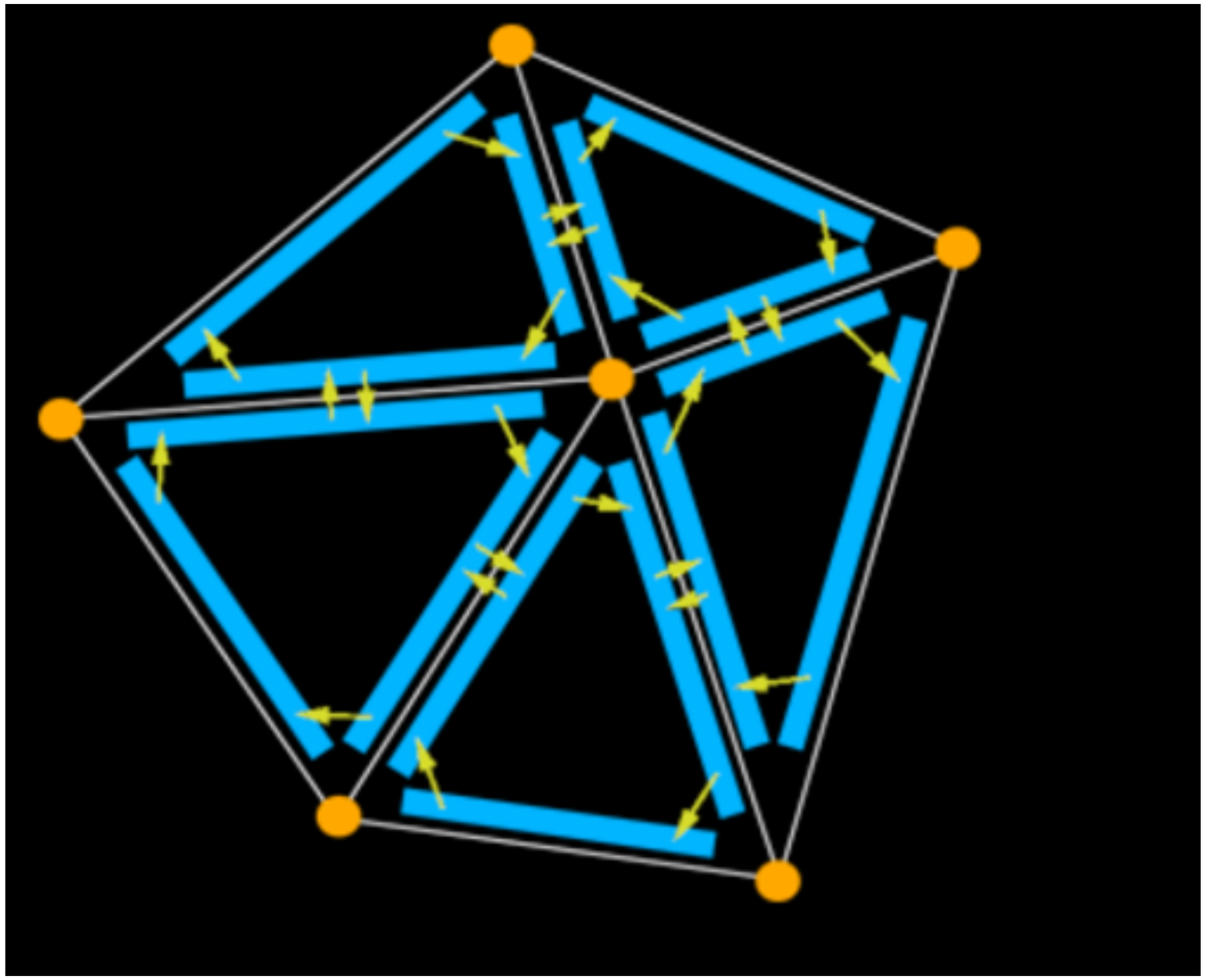
- Iterate the edges along a face f

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

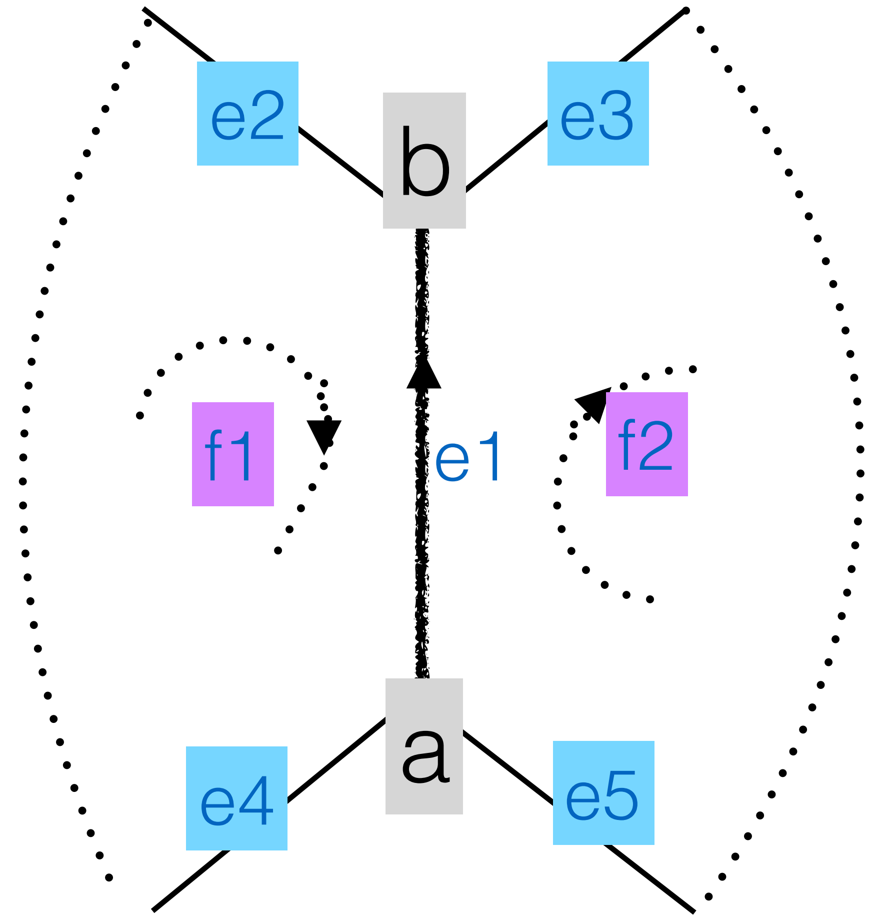
- Iterate over the edges adjacent to a vertex v

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

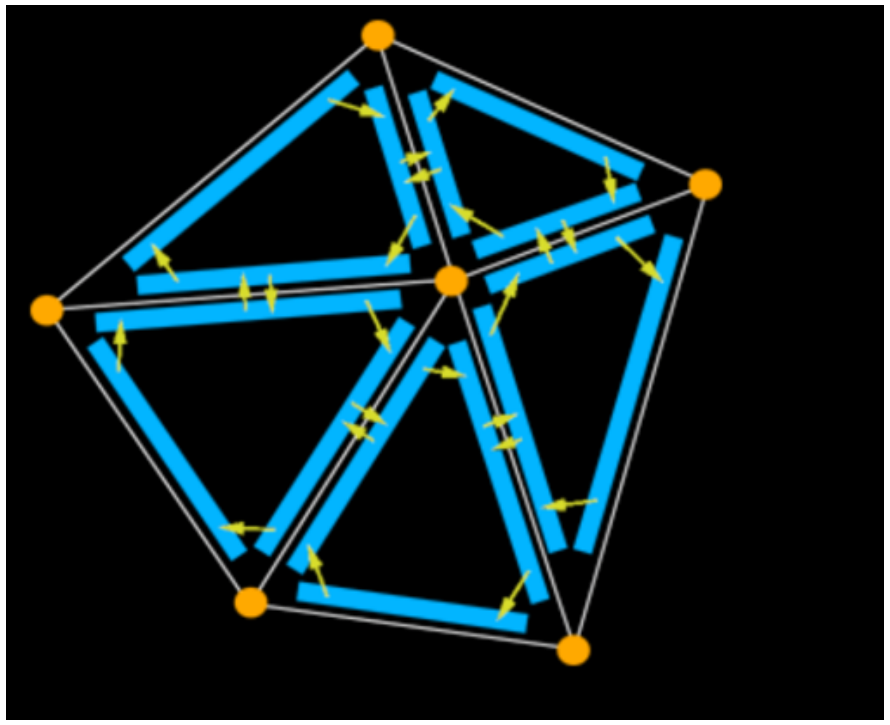
- are two faces adjacent?
- are two vertices adjacent?
- what are all faces that use this vertex ?



Topological structures for polygonal meshes

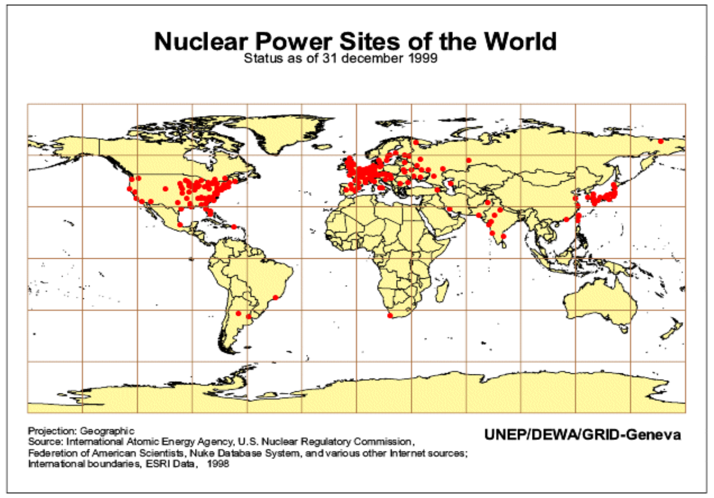


- Support basic adjacency operations fast
- But
    - use a lot of space !!!
    - need to be constructed from raw data
    - involve complex programming



Summary

points


Nuclear Power Sites of the World
Status as of 31 december 1999
Projection: Geographic
Source: International Atomic Energy Agency, U.S. Nuclear Regulatory Commission, Federation of American Scientists, Nuke Database System, and various other Internet sources; International boundaries, ESRI Data, 1998
UNEP/DEWA/GRID-Geneva

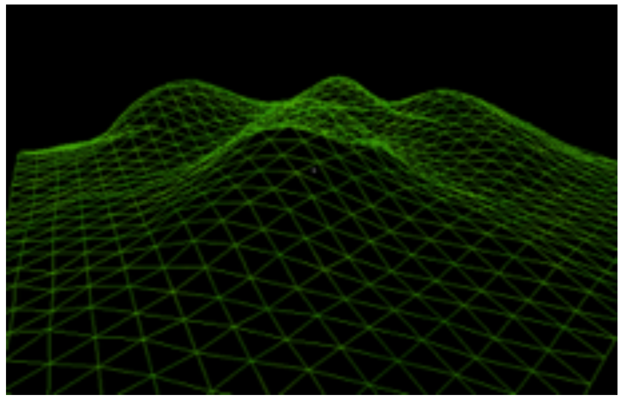
networks

graphs

planar maps

topological data structures
(winged edge, half edge)

terrains

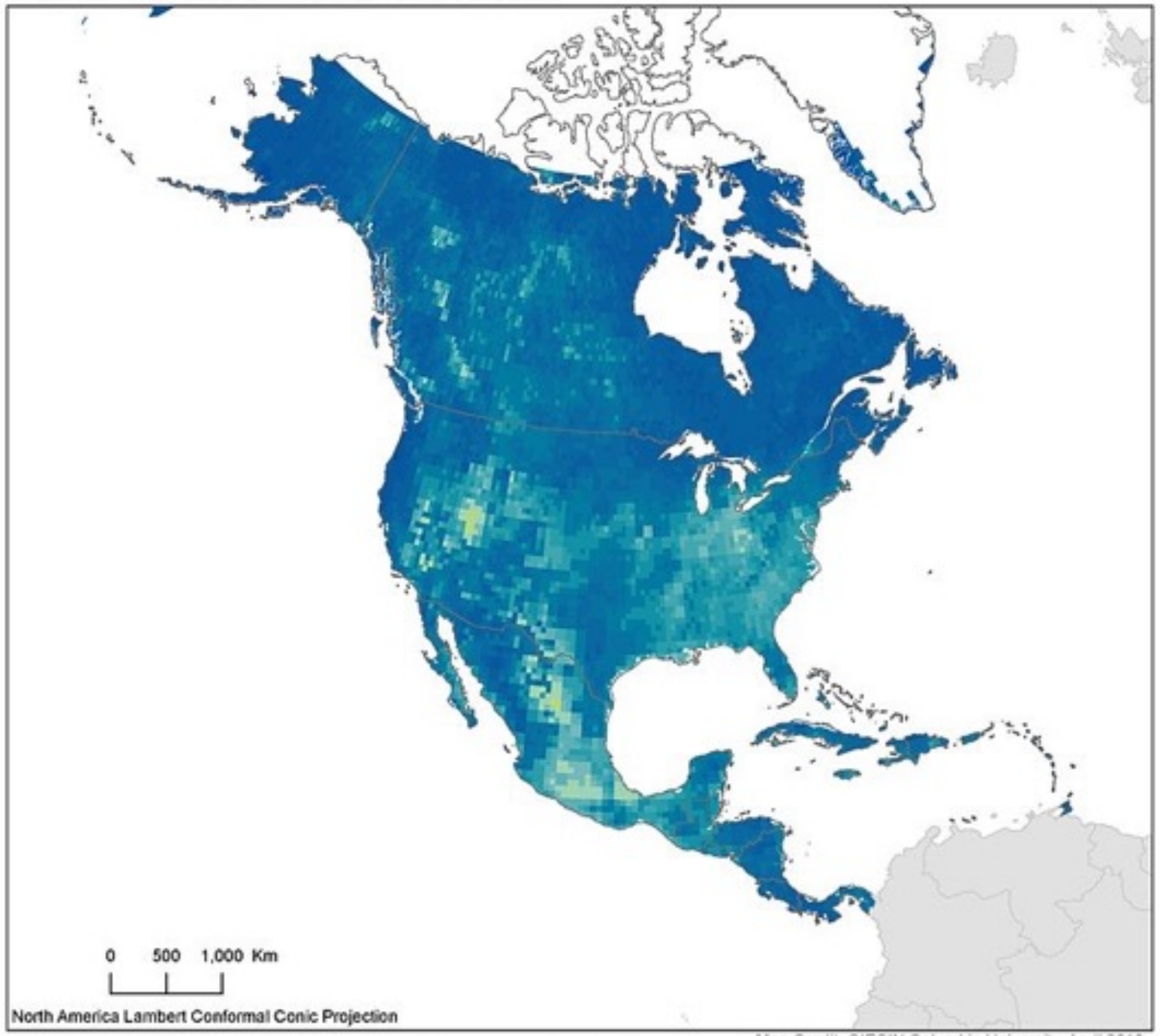


raster or TIN

- Two fundamentally different models
  - **Vector** data: points, lines, polygon + data structures
  - **Raster** data: matrix of values

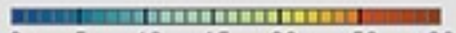
Global Annual Average PM$_{2.5}$ Grids from MODIS and MISR Aerosol Optical Depth (AOD), 2010: North America

Satellite-Derived Environmental Indicators



Map Credit: CIESIN Columbia University, April 2013.

Global Annual PM$_{2.5}$ Grids from MODIS and MISR Aerosol Optical Depth (AOD) data sets provide annual "snap shots" of particulate matter 2.5 micrometers or smaller in diameter from 2001–2010. Exposure to fine particles is associated with premature death as well as increased morbidity from respiratory and cardiovascular disease, especially in the elderly, young children, and those already suffering from these illnesses. The grids were derived from Moderate Resolution Imaging Spectroradiometer (MODIS) and Multi-angle Imaging SpectroRadiometer (MISR) Aerosol Optical Depth (AOD) data. The raster grid cell size is approximately 50 sq. km at the equator, and the extent is from 70°N to 60°S latitude.



Center for International Earth Science Information Network
EARTH INSTITUTE | COLUMBIA UNIVERSITY

Data Source: Battelle Memorial Institute, and Center for International Earth Science Information Network (CIESIN)/Columbia University. 2013. Global Annual Average PM2.5 Grids from MODIS and MISR Aerosol Optical Depth (AOD), 2001–2010. Palisades, NY: NASA Socioeconomic Data and Applications Center (SEDAC). http://sedac.ciesin.columbia.edu/data/set/sdei-global-annual-avg-pm2-5-2001-2010.

© 2013. The Trustees of Columbia University in the City of New York.

This document is licensed under a Creative Commons 3.0 Attribution License http://creativecommons.org/licenses/by/3.0/