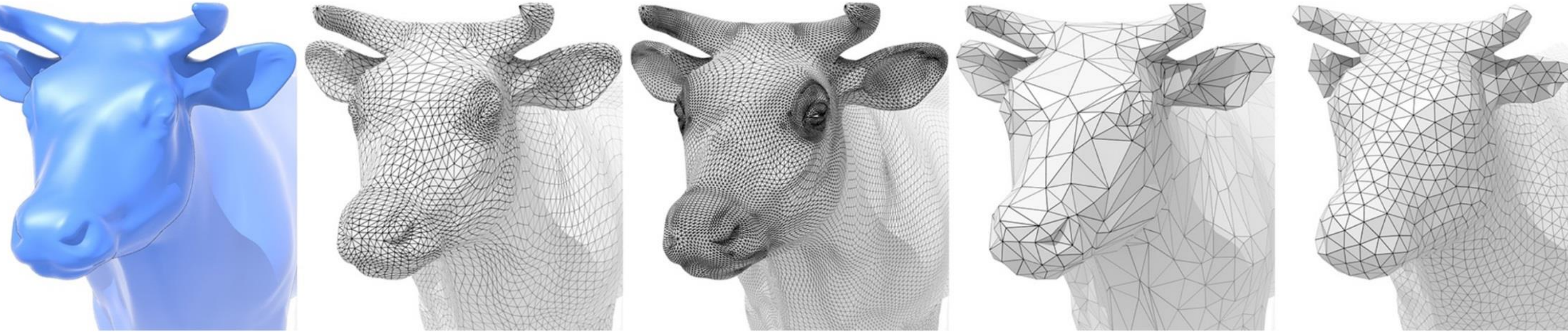
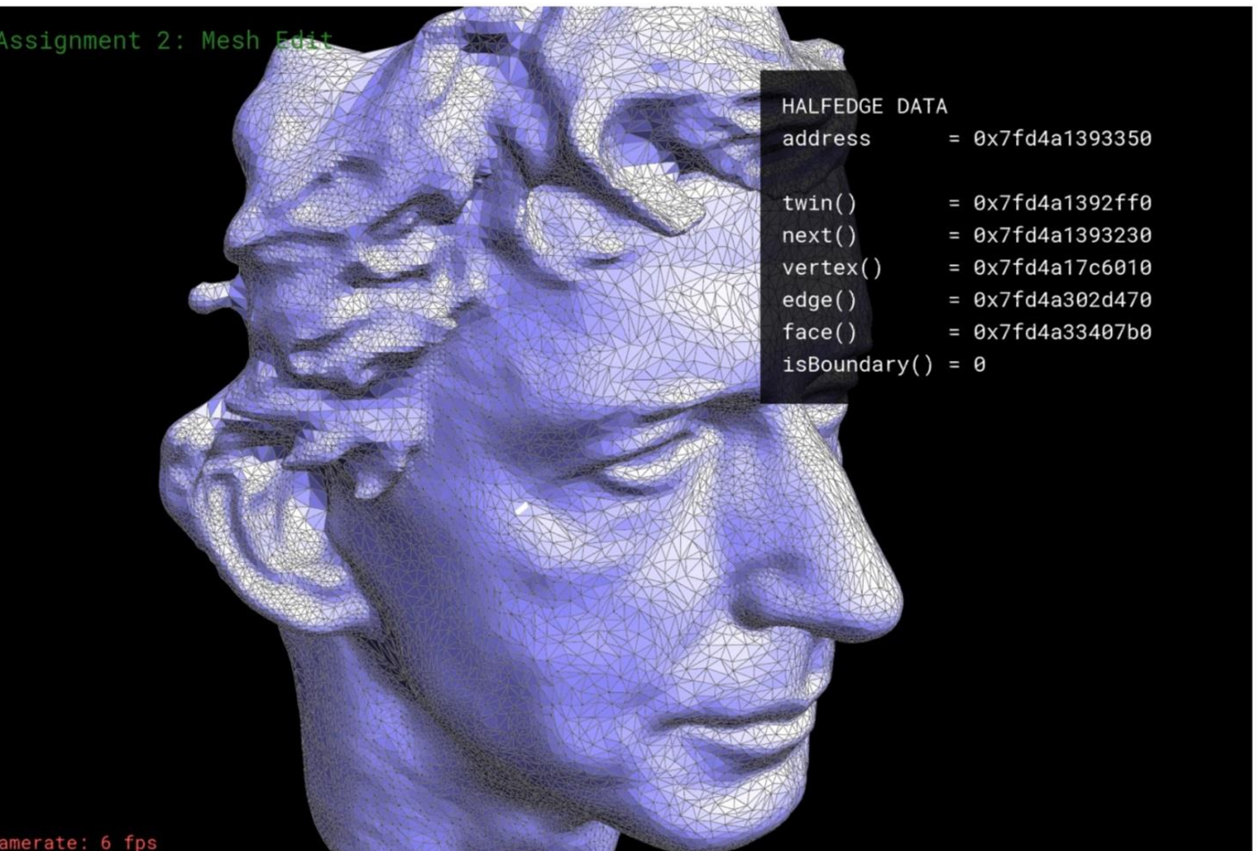
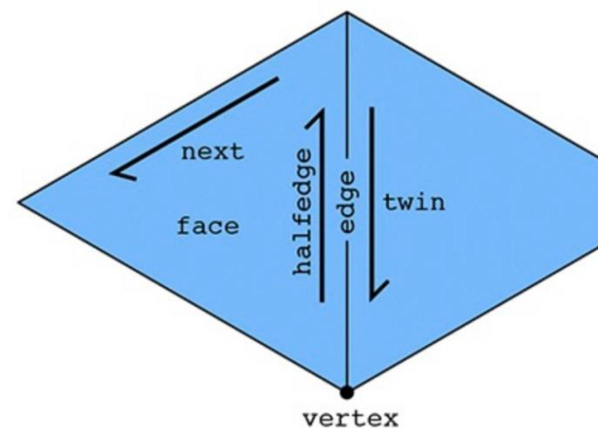
Geometry: curves, surfaces and Polygonal Meshes

NGUYEN THUY LINH





Assignment 2: Mesh Edit
HALFEDGE DATA
address = 0x7fd4a1393350
twin() = 0x7fd4a1392ff0
next() = 0x7fd4a1393230
vertex() = 0x7fd4a17c6010
edge() = 0x7fd4a302d470
face() = 0x7fd4a33407b0
isBoundary() = 0
amerate: 6 fps



next
face
halfedge
edge
twin
vertex

Last time: overview of geometry

- Many types of geometry in nature
- Demand sophisticated representations
- Two major categories:
  - IMPLICIT - “tests” if a point is in shape
  - EXPLICIT - directly “lists” points
- Lots of representations for both
- Today:
  - what is a surface, anyway?
  - nuts & bolts of polygon meshes

Geometry

Q: What is a “surface?”

A: Oh, it’s a 2-dimensional manifold.

Q: Ok… but what the heck is a *manifold*?

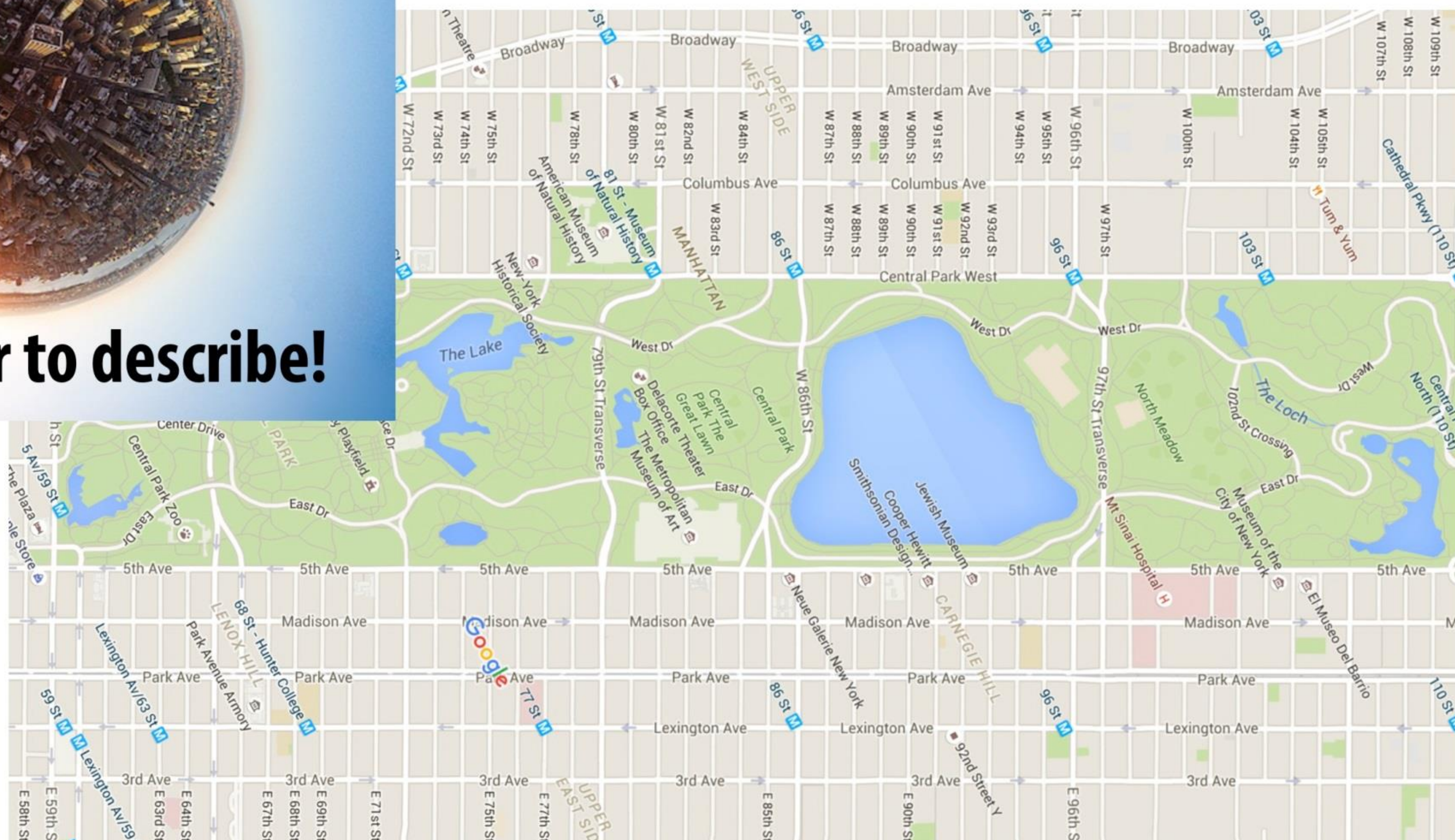
The Earth looks flat, if you get close enough

Can pretend we're on a grid:

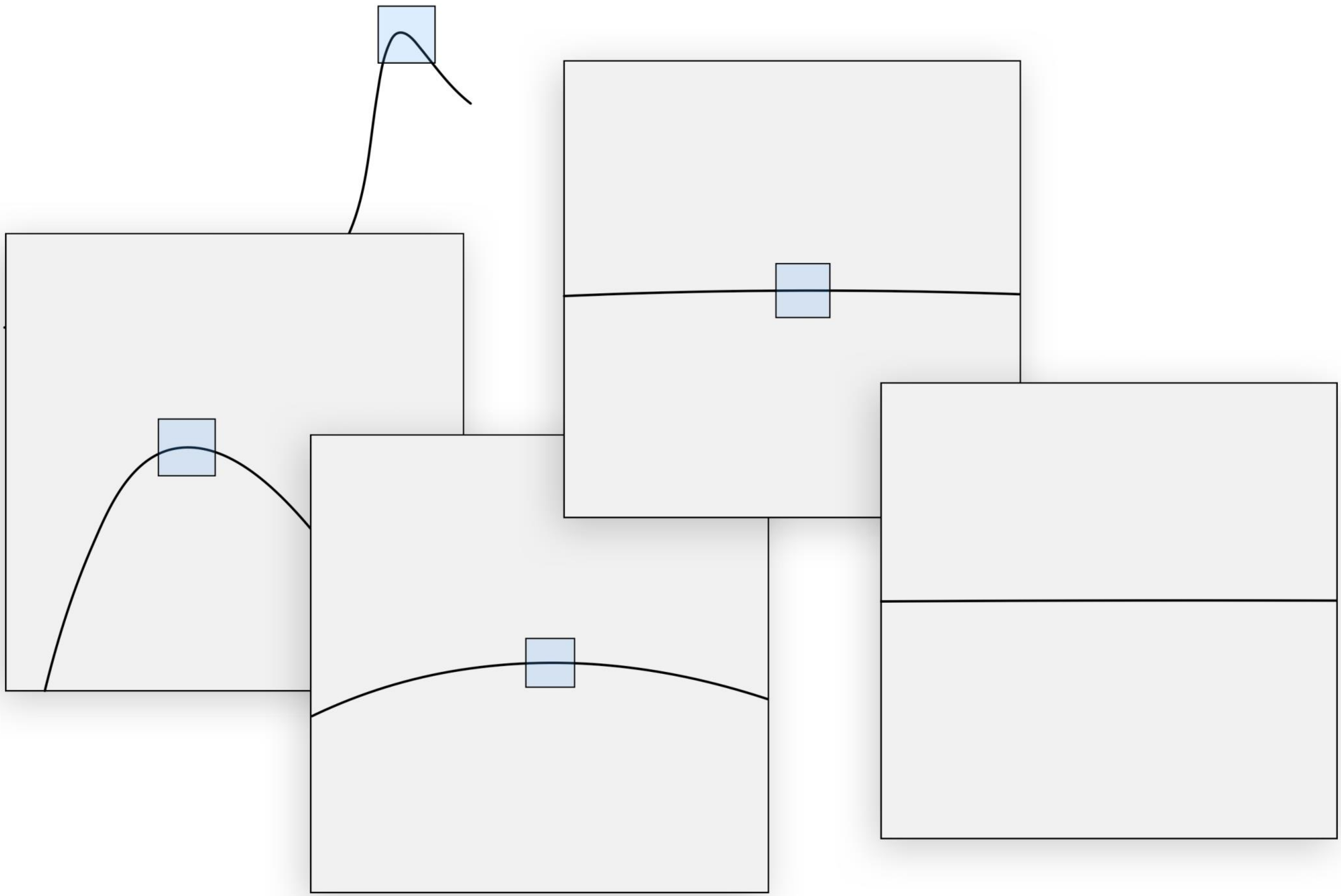
The Earth looks flat, if you get close enough

Much harder to describe!

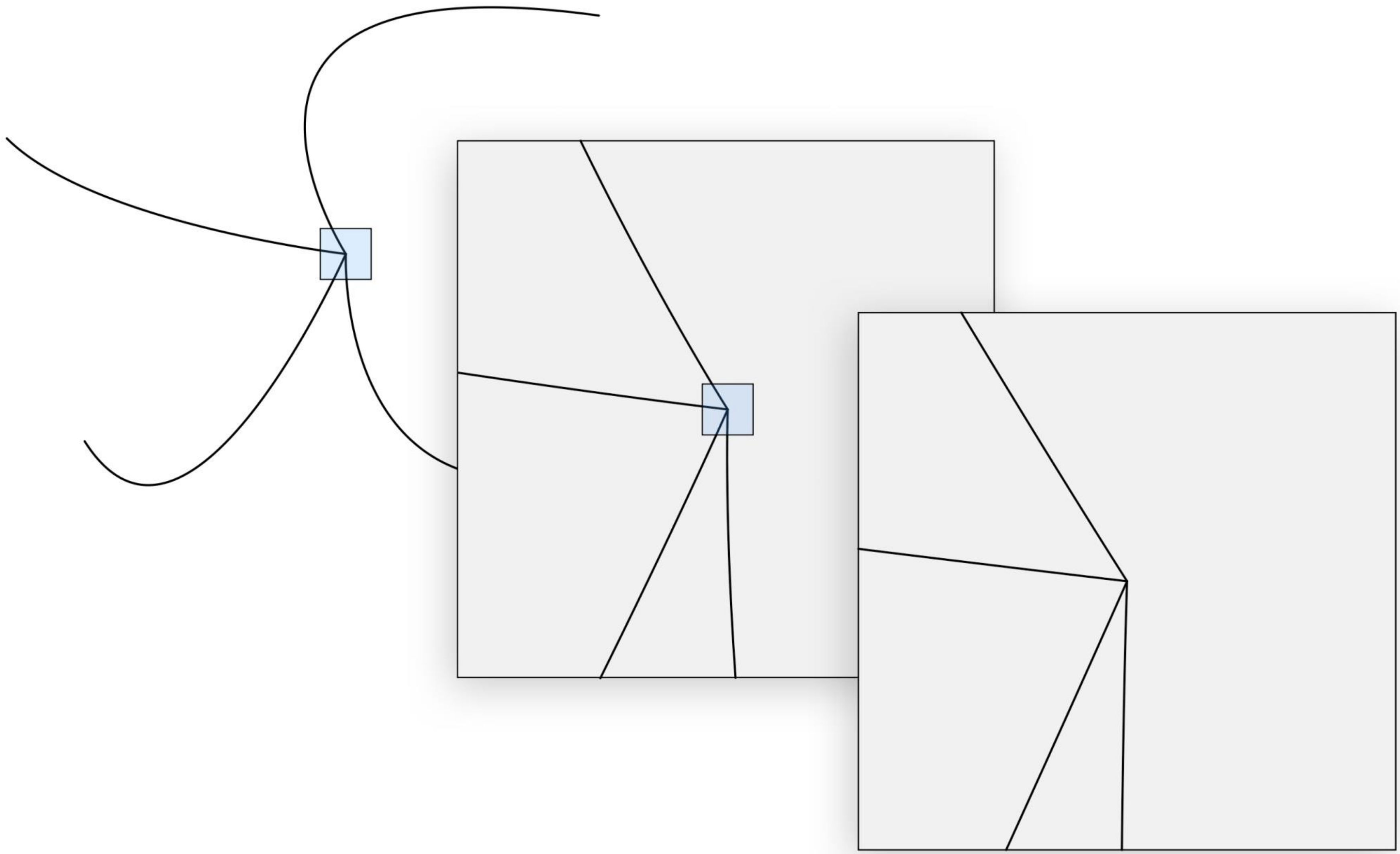
Can pretend we're on a grid:



A *smooth manifold* also looks flat close up

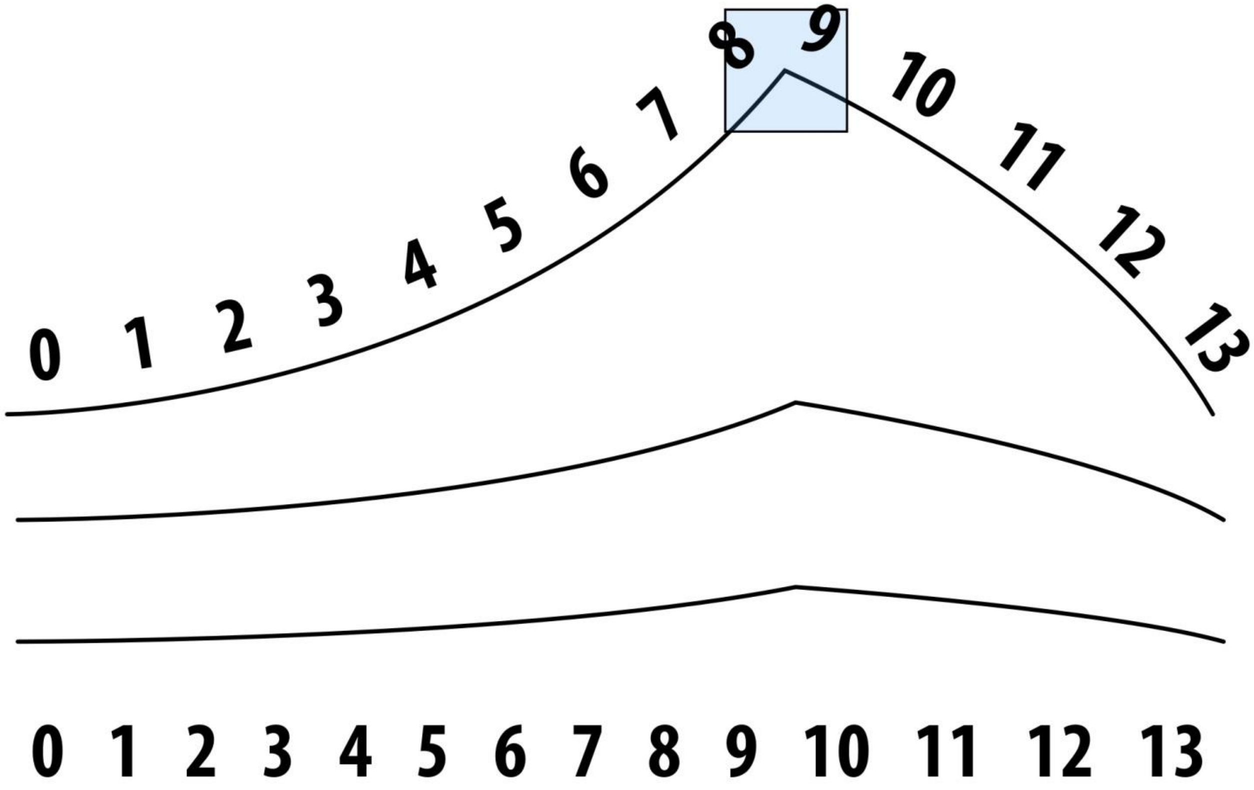


Not all curves are smooth manifolds



No matter how close we get, doesn't look like a single line!

What about sharp corners?



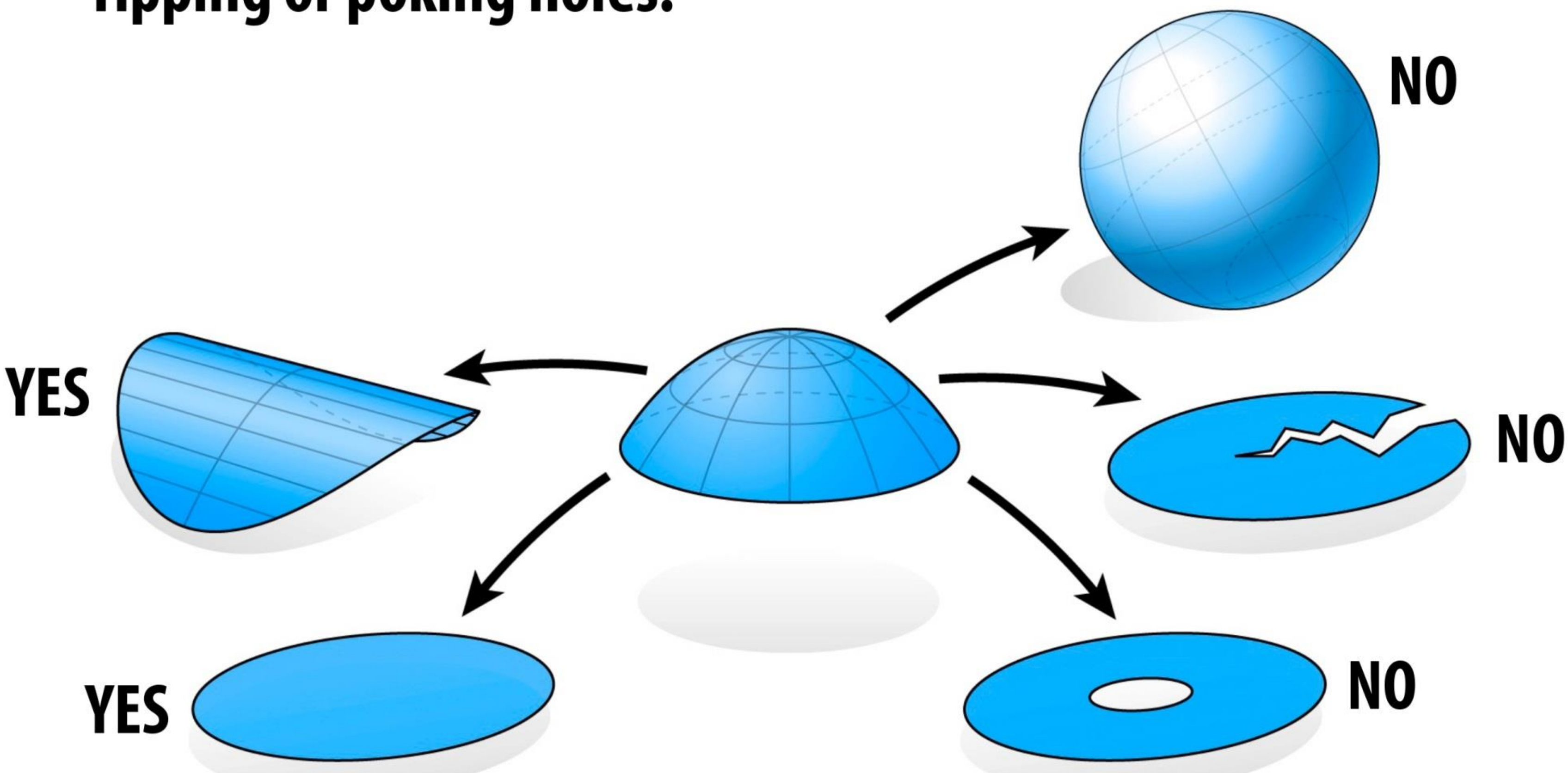
Can easily be flattened *into* a line.

Can still assign coordinates (just like Manhattan!)

...But is it a manifold?

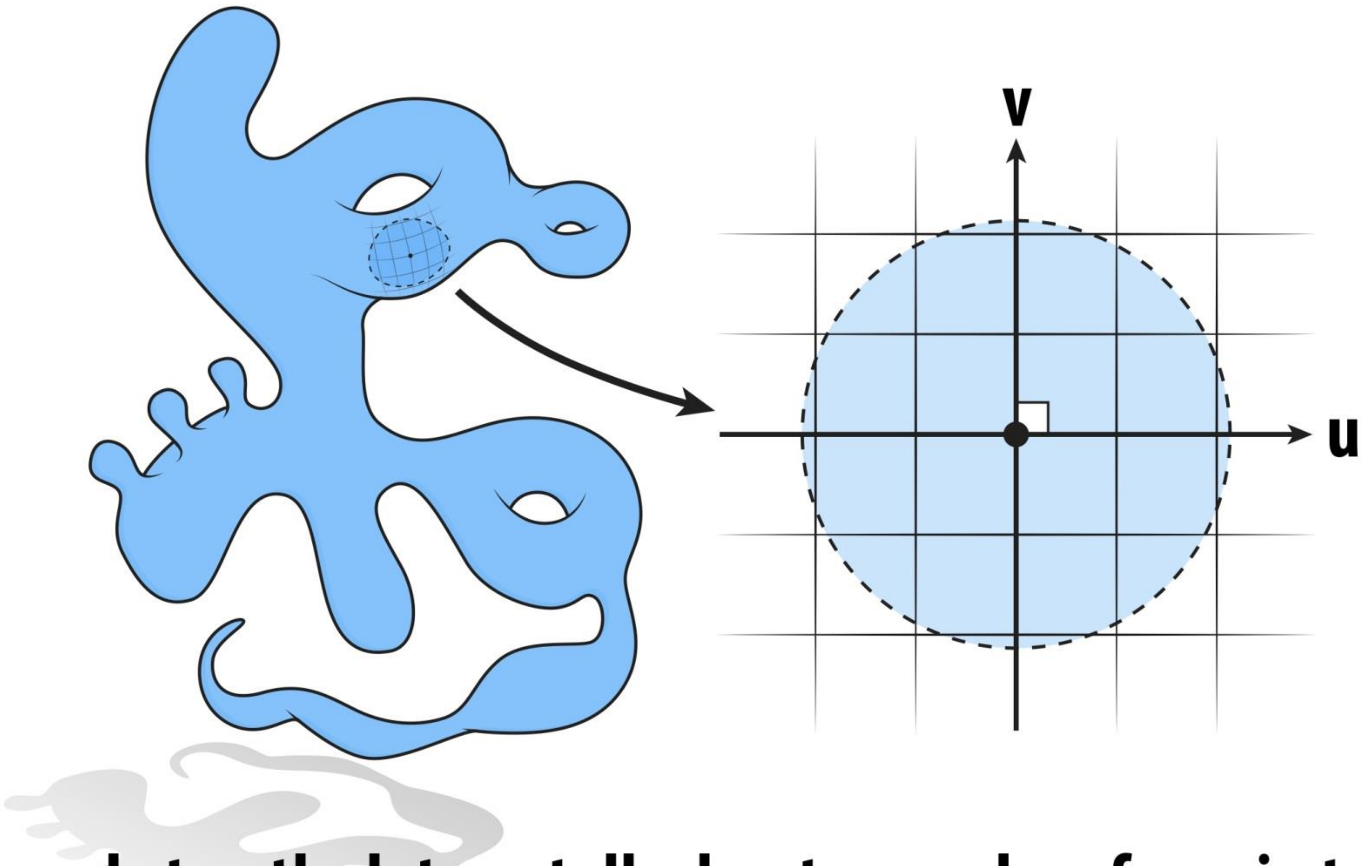
Definition of a manifold

- *"A subset S of R^m is an n-manifold if every point p in S is contained in a neighborhood that can be mapped bijectively and continuously (both ways) to the open ball in R^n."*
- In other words: each little piece can be made flat without "ripping or poking holes."



Why is the manifold property valuable?

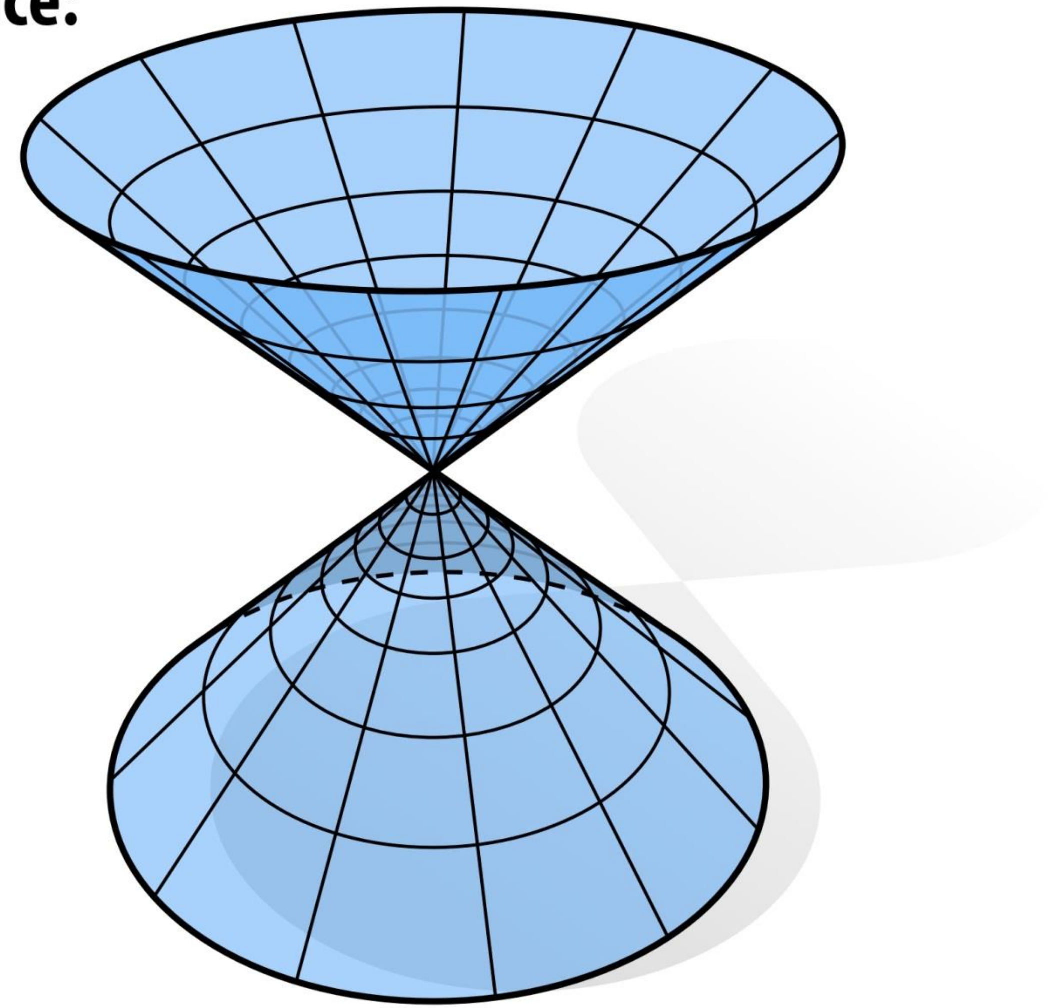
- **Makes life simple: all surfaces look the same (at least locally).**
- **Gives us coordinates! (At least locally.)**



- **More abstractly, lets us talk about curved surfaces in terms of familiar tools: vector calculus & linear algebra.**

Isn't every shape manifold?

- **No, for instance:**

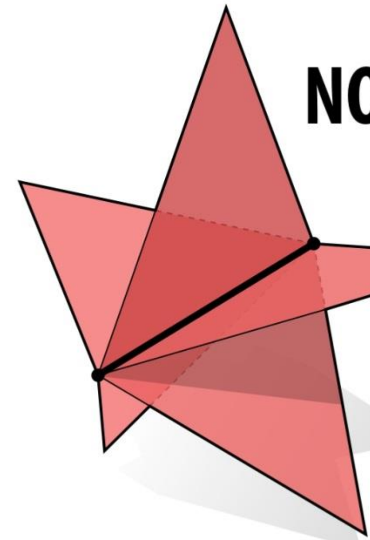


No way to put a (simple) coordinate system on the center point!

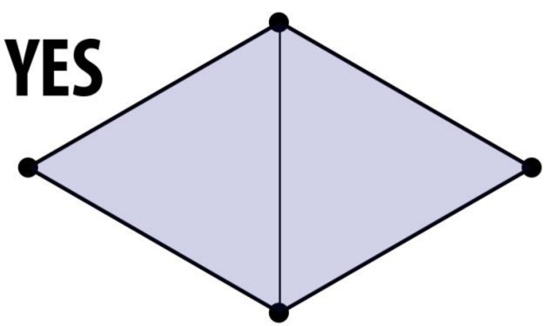
What about discrete surfaces?

- Surfaces made of, e.g., triangle are no longer *smooth*.
- But they can still be *manifold:*
  - two triangles per edge (no “fins”)
  - every vertex looks like a “fan”

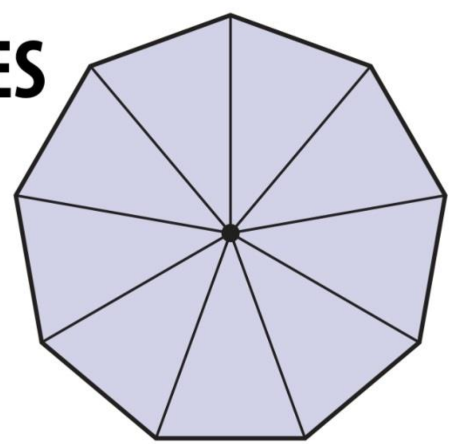
NO

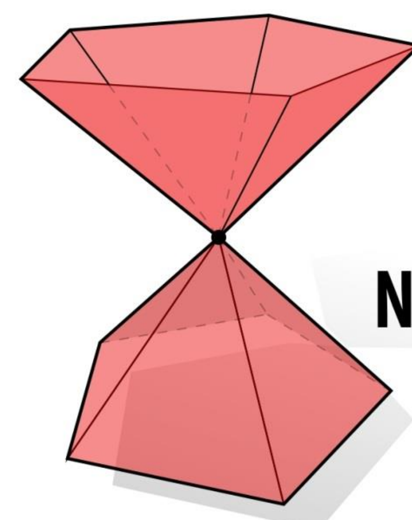


YES



YES



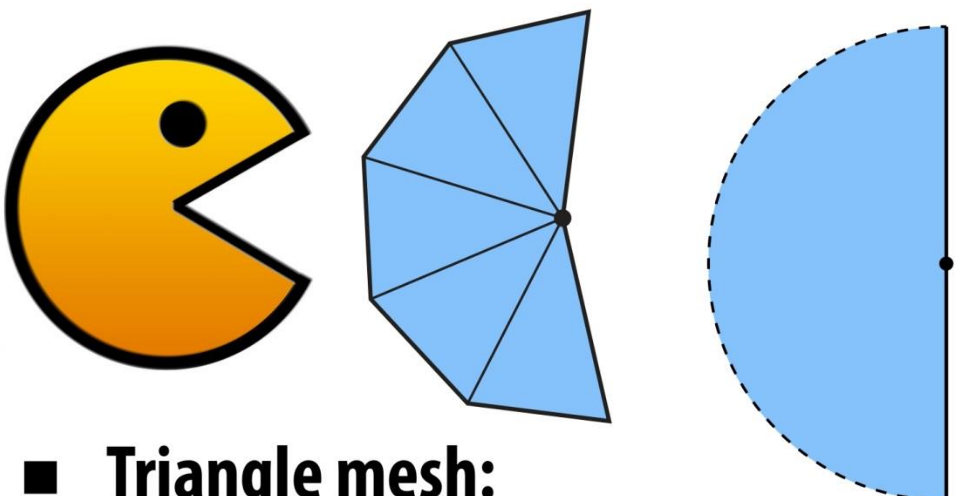


NO

- Why? *Simplicity.*
  - no special cases to handle
  - keeps data structures (reasonably) simple)

What about boundary?

- The boundary is where the surface "ends."
- E.g., waist & ankles on a pair of pants.
- Locally, looks like a *half* disk
- Globally, each boundary forms a loop



- Triangle mesh:
  - one triangle per boundary edge
  - boundary vertex looks like "pacman"

Polygonal Meshes

Modeling with basic shapes (cube, cylinder, sphere, etc) too primitive

Difficult to approach realism

Polygonal meshes:

- Collection of polygons, or faces, that form “skin” of object
- Offer more flexibility
- Models complex surfaces better
- Examples:
  - Human face
  - Animal structures
  - Furniture, etc

Polygonal Meshes

Have become standard in CG

OpenGL

- Good at drawing polygon
- Mesh = sequence of polygons

Simple meshes exact. (e.g barn)

Complex meshes approximate (e.g. human face)

Later: use shading technique to smoothen

Non-solid Objects

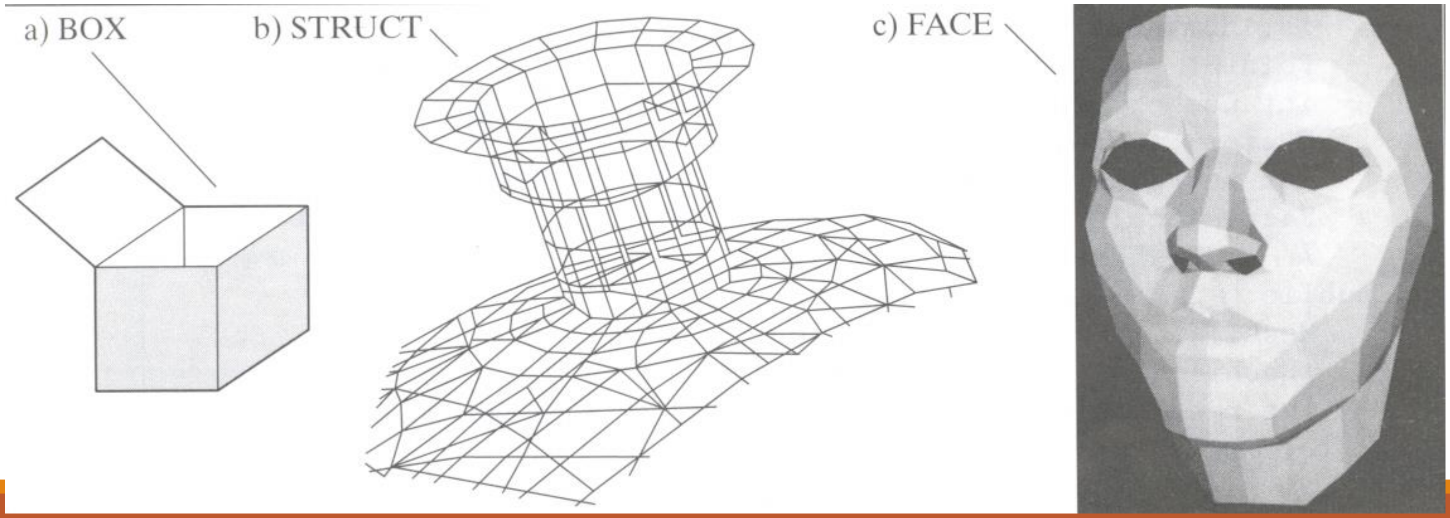
Examples: box, face

Visualize as infinitely thin skin

Meshes to approximate complex objects

Shading used later to smoothen

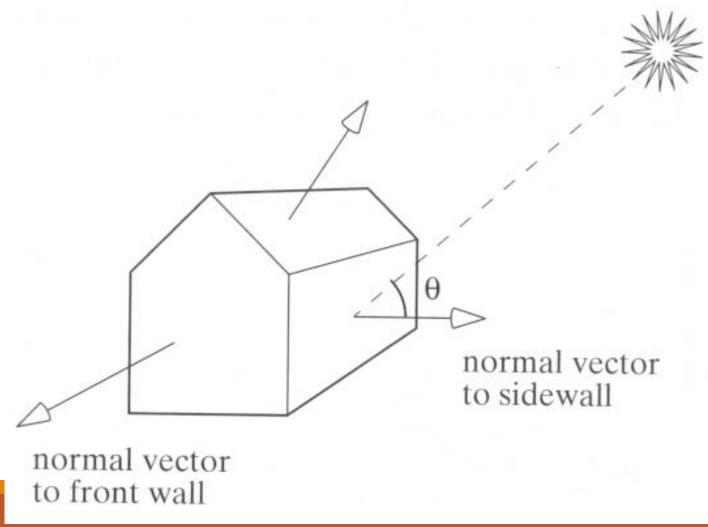
Non-trivial: creating mesh for complex objects (CAD)



What is a Polygonal Mesh

Polygonal mesh given by:

- Polygon list
- Direction of each polygon
- Represent direction as normal vector
- Normal vector used in shading
- Normal vector/light vector determines shading

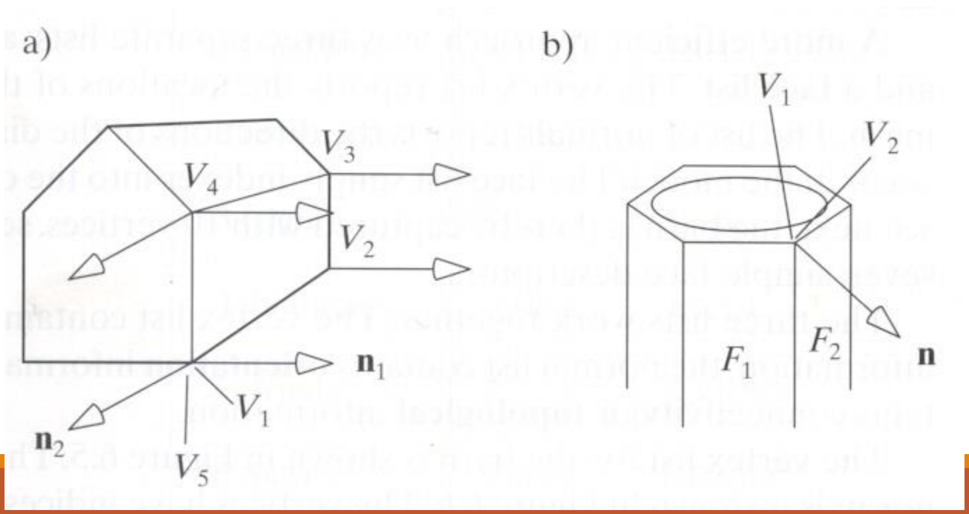


Vertex Normal

Use vertex normal instead of face normal

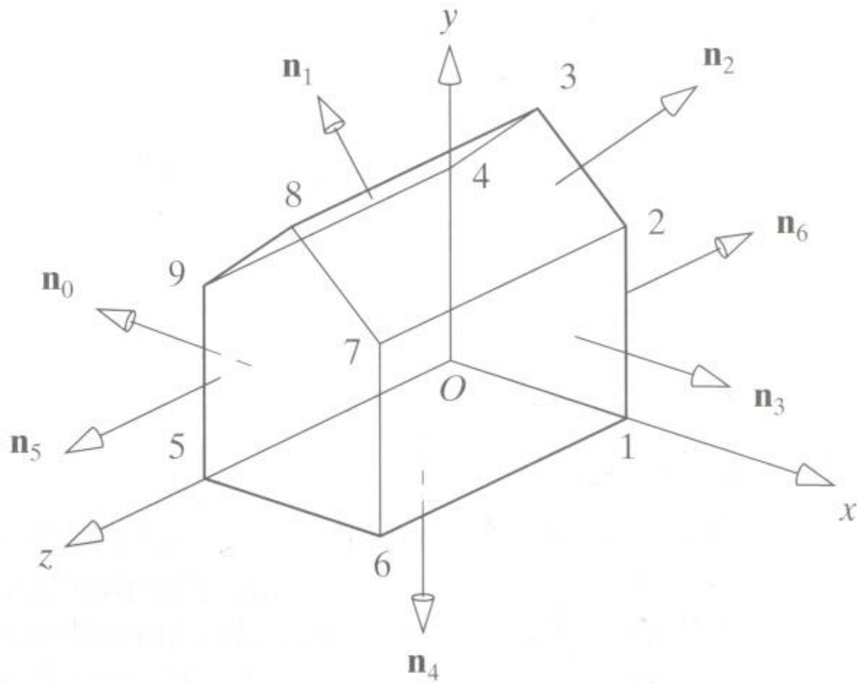
See advantages later:

- Facilitates clipping
- Shading of smoothly curved shapes
- Flat surfaces: all vertices associated with same **n**
- Smoothly curved surfaces: *V1*, *V2* with common edge share **n**



Defining Polygonal Mesh

Use barn example below:



Defining Polygonal Mesh

Three lists:

- Vertex list: distinct vertices (vertex number, Vx, Vy, Vz)
- Normal list: Normals to faces (normalized nx, ny, nz)
- Face list: indexes into vertex and normal lists. i.e. vertices and normals associated with each face

Face list convention:

- Traverse vertices counter-clockwise
- Interior on left, exterior on right

Newell Method for Normal Vectors

Martin Newell at Utah (teapot guy)

Normal vector:

- calculation difficult by hand
- Given formulae, suitable for computer
- Compute during mesh generation

Simple approach used previously:

- Start with any three vertices V1, V2, V3
- Form two vectors, say V1-V2, V3-V2
- Normal: cross product (perp) of vectors

Newell Method for Normal Vectors

Problems with simple approach:

- If two vectors are almost parallel, cross product is small
- Numerical inaccuracy may result
- Newell method: robust
- Formulae: Normal N = (mx, my, mz)

$$m_x = \sum_{i=0}^{N-1} \left(y_i - y_{next(i)}\right)\left(z_i + z_{next(i)}\right)$$

$$m_y = \sum_{i=0}^{N-1} \left(z_i - z_{next(i)}\right)\left(x_i + x_{next(i)}\right)$$

$$m_z = \sum_{i=0}^{N-1} \left(x_i - x_{next(i)}\right)\left(y_i + y_{next(i)}\right)$$

Newell Method Example

Example: Find normal of polygon with vertices

P0 = (6,1,4), P1=(7,0,9) and P2 = (1,1,2)

Solution:

Using simple cross product:

((7,0,9)-(6,1,4)) X ((1,1,2)-(6,1,4)) = (2,-23,-5)

Using Newell method, plug in values result is the same:

Normal is (2, -23, -5)

Meshes in Programs

Class *Mesh*

Helper classes

- VertexID
- Face

Mesh Object:

- Normal list
- Vertex list
- Face list

Use arrays of pt, norm, face

Dynamic allocation at runtime

Array lengths: numVerts, numNormals, numFaces

Meshes in Programs

Face:

- Vertex list
- Normal vector associated with each face
- Array of index pairs

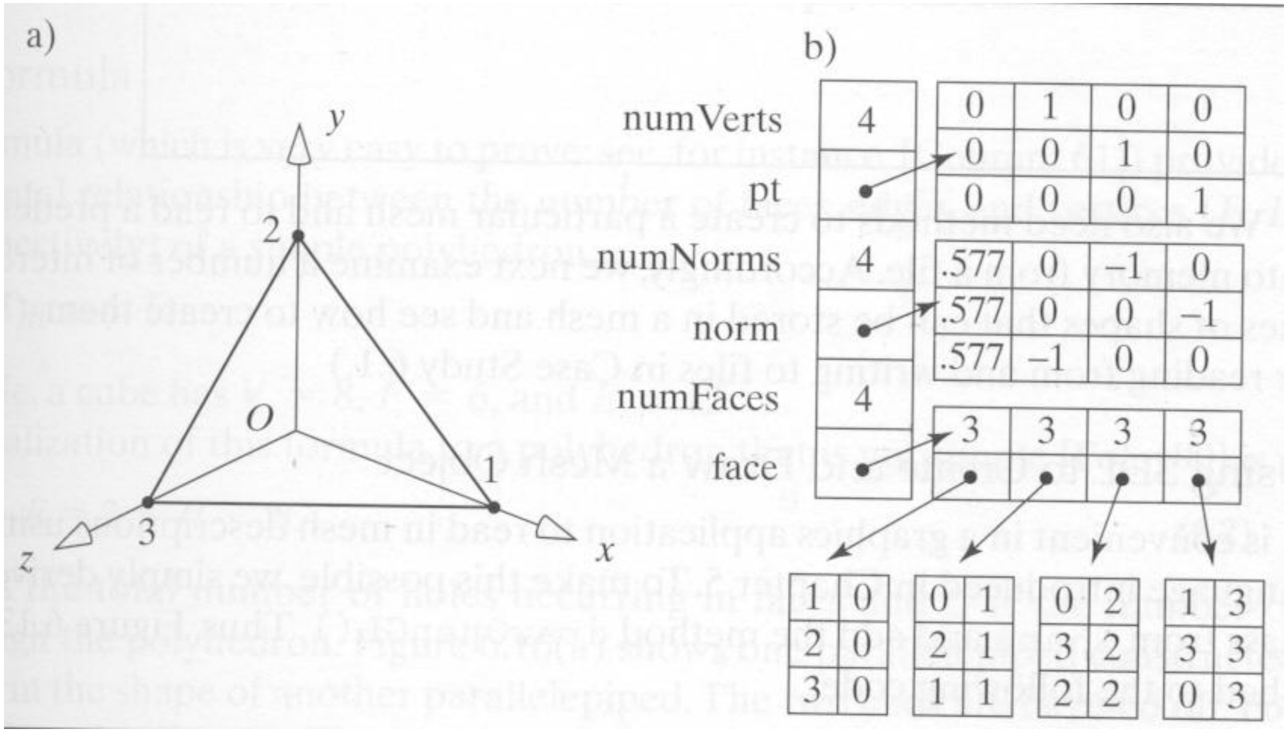
Example, vth vertex of fth face:

- Position: pt[face[f].vert[v].vertIndex]
- Normal vector: norm[face[f].vert[v].normIndex]

Organized approach, permits random access

Meshes in Programs

Tetrahedron example



Meshes in Programs

Data structure:

```
// ############### Vertex ID #####################
class VertexID
   public:
                    int vertIndex;   // index of this vertex in the vertex list
                    int normIndex;   // index of this vertex's normal
}
// ############### Face #####################
class Face
   public:
                    int nVerts;      // number of vertices in this face
                    VertexID *vert;   // the list of vertex and normal indices
                    Face( ){nVerts = 0; vert = NULL;}   // constructor
                    -Face( ){delete[] vert; nVerts = 0;  // destructor
};
```

Meshes in Programs

```
// ###############   Mesh ######################
class Mesh{
  private:
        int numVerts;        // number of vertices in the mesh
        Point3 *pt;          // array of 3D vertices
        int numNormals;      // number of normal vertices for the mesh
        Vector3 *norm;       // array of normals
        int numFaces;        // number of faces in the mesh
        Face *face;          // array of face data
        //… others to be added later
  public:
        Mesh( );             // constructor
        ~Mesh( );            // destructor
        int readFile(char *fileName);   // to read in a filed mesh
        ….. other methods….
}
```

Drawing Meshes Using OpenGL

Pseudo-code:

```
for(each face f in Mesh)
{
        glBegin(GL_POLYGON);
        for(each vertex v in face f)
        {
          glNormal3f(normal at vertex v);
          glVertex3f(position of vertex v);
        }
        glEnd( );
}
```

Drawing Meshes Using OpenGL

Actual code:

```
Void Mesh::draw( )          // use openGL to draw this mesh
{
   for(int f = 0;f < numFaces;f++)
   {
         glBegin(GL_POLYGON);
            for(int v=0;v<face[f].nVerts;v++)        // for each one
            {
                        int in = face[f].vert[v].normIndex;  // index of this normal
                        int iv = face[f].vert[v].vertIndex;    // index of this vertex
                        glNormal3f(norm[in].x, norm[in].y, norm[in].z);
                        glVertex3f(pt[iv].x, pt[iv].y, pt[iv].z);
            }
        glEnd( );
    }
}
```

Drawing Meshes Using SDL

Scene class reads SDL files

Accepts keyword *Mesh*

Example:

- Pawn stored in mesh file pawn.3vn
- Add line:
  - Push translate 3 5 4 scale 3 3 3 mesh pawn.3vn pop

More on Meshes

Simple meshes easy by hand

Complex meshes:

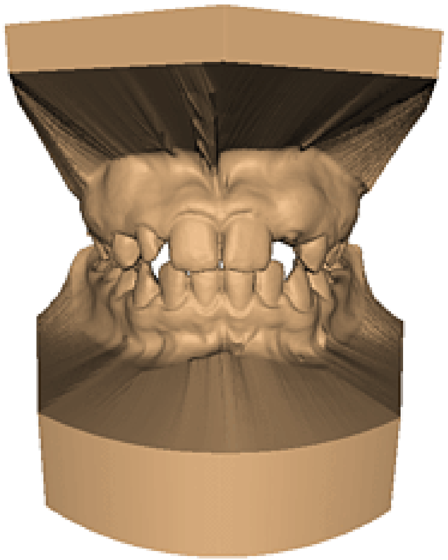
- Mathematical functions
- Algorithms
- Digitize real objects

Libraries of meshes available

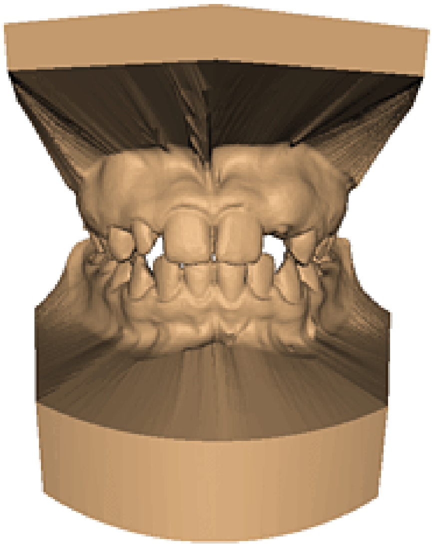
Mesh trends:

- 3D scanning
- Mesh Simplification

3D Simplification Example



Original: 424,000 triangles



60,000 triangles (14%).

1000 triangles (0.2%)

(courtesy of Michael Garland and Data courtesy of Iris Development.)

References

Hill, 6.1-6.2