Geometric Modeling: Meshes
Theory and Practice

CS 415: Game Development

Professor Eric Shaffer

ILLINOIS

Polygonal Meshes

In rendering, we typically will generate an image by simulating the reflection of light off surfaces

Rasterization engines most often use polygonal meshes to represent surfaces

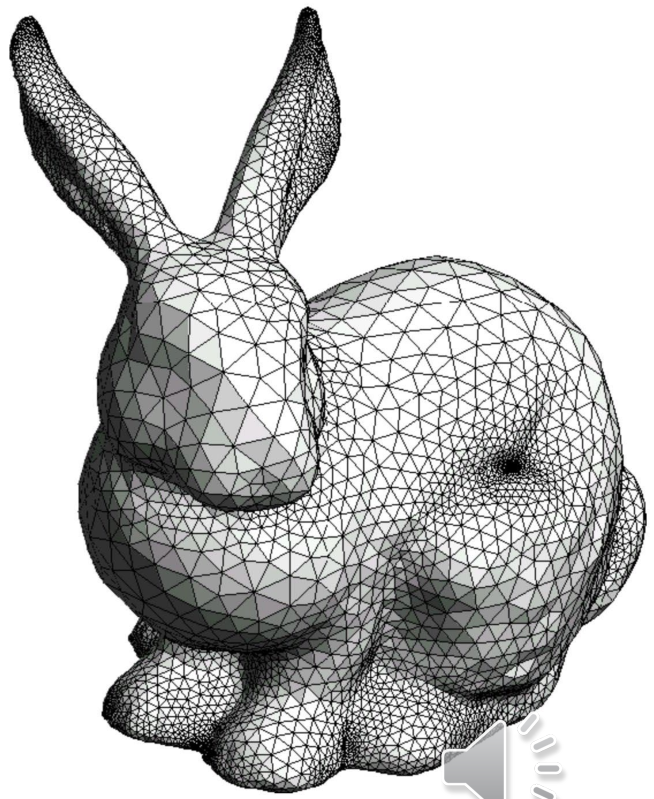
Modern GPUs are designed specifically to rasterize triangles

Many advantages to using triangles:

- Simplest 2D primitive…
- Any 2D polygon can be triangulated
- Can easily represent sharp surface features

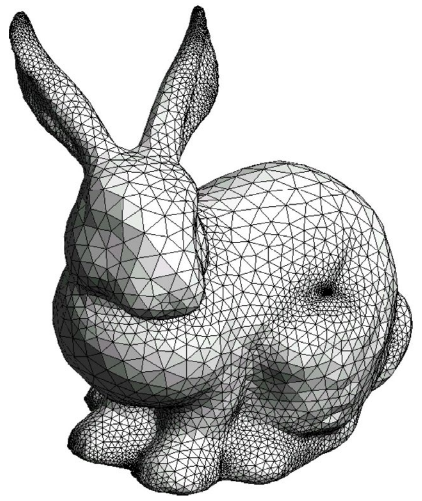
Why do we say 2D when we are rendering in 3D?

Any disadvantages you can think of?



ILLINOIS

Polygonal Meshes



A mesh is a network of triangles.
They connect to one another through shared vertices and edges to form a single continuous surface.

A mesh can be handled more efficiently than a collection of the same number of unrelated triangles.

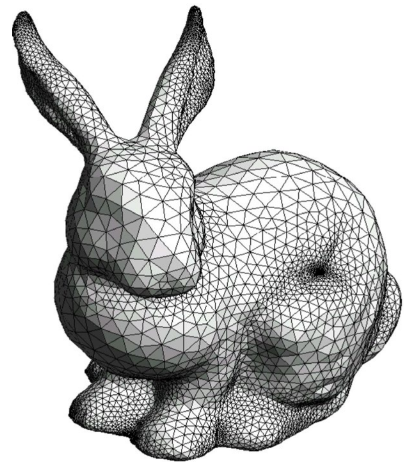
The minimum information required for a triangle mesh is:

- A set of triangles (triples of vertices)
- The positions (in 3D space) of their vertices

Many file formats/data structures store additional data at the vertices, edges, or face

- This data support texture mapping, shading, animation, and other operations.

Polygonal Meshes



Meshes being surface-like can be formalized as constraints on the mesh topology

- Topology - the way the triangles connect together, without regard for the vertex positions
- Many algorithms will only work on a mesh with predictable connectivity.
- The simplest and most restrictive requirement on the topology of a mesh is to be a manifold.
- A manifold mesh is “watertight”
  - It has no gaps and separates the space on the inside of the surface from the space outside
  - It also looks like a surface everywhere on the mesh
  - We’ll leave the precise definitions to the mathematicians;
  - The term manifold comes from the mathematical field of topology

ILLINOIS

Polygonal Meshes

Many algorithms assume that meshes are manifold
(it's always a good idea to verify this property to prevent crashes or infinite loops if you are handed a malformed mesh as input)

Check by verifying the following conditions:

- Every edge is shared by exactly two triangles
- Every vertex has a single, complete loop of triangles around it.

Sometimes, it's necessary to allow meshes to have boundaries.
Such meshes are not manifolds—They are not necessarily watertight.
(We can relax the requirements of a manifold mesh to those for a manifold with boundary without causing problems for most mesh processing algorithms)

The relaxed conditions are
- Every edge is used by either one or two triangles.
- Every vertex connects to a single edge-connected set of triangles.

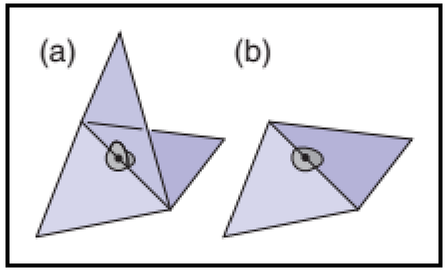


Figure 12.1. Non-manifold (a) and manifold (b) interior edges.

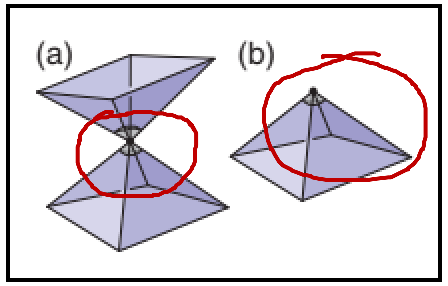


Figure 12.2. Non-manifold (a) and manifold (b) interior vertices.

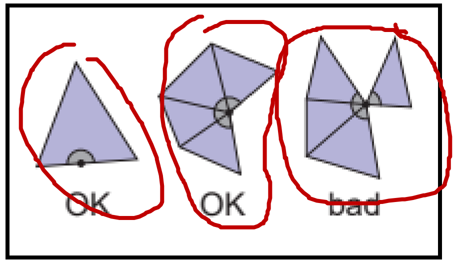


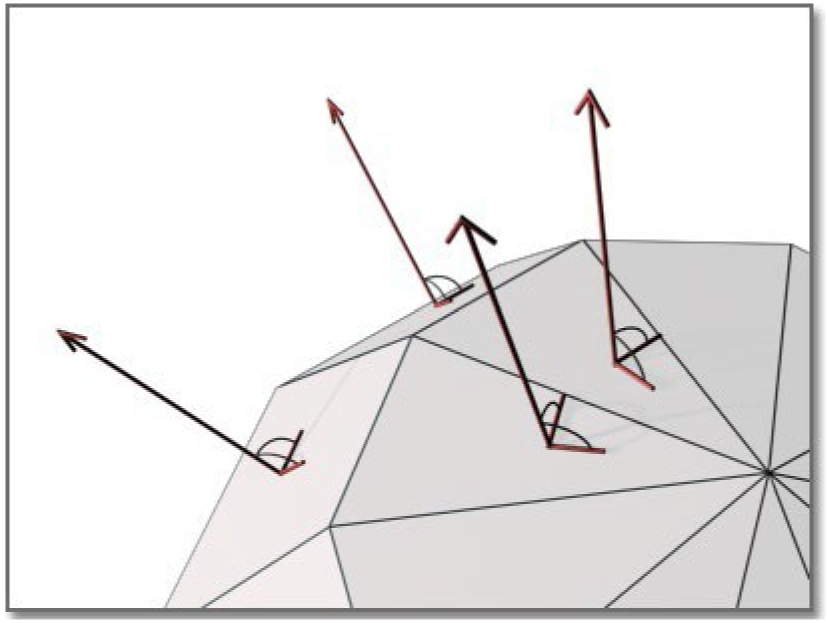
Figure 12.3. Conditions at the edge of a manifold with boundary.

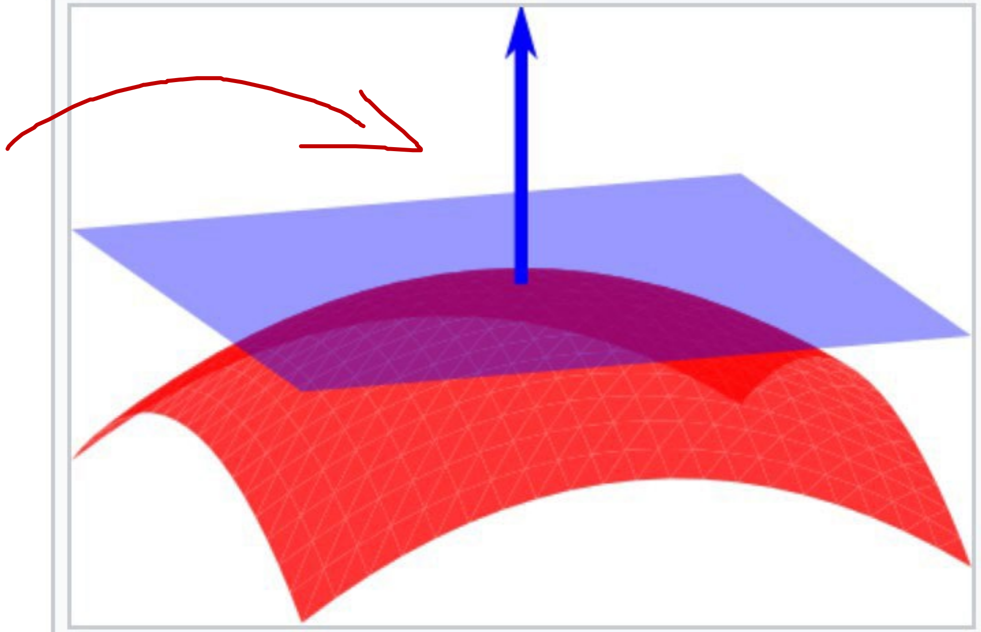
Vocabulary: Surface Normals

A normal is a vector that is perpendicular to object

Each triangle in a surface mesh has outward facing normal

The normal is just the vector perpendicular to the triangle

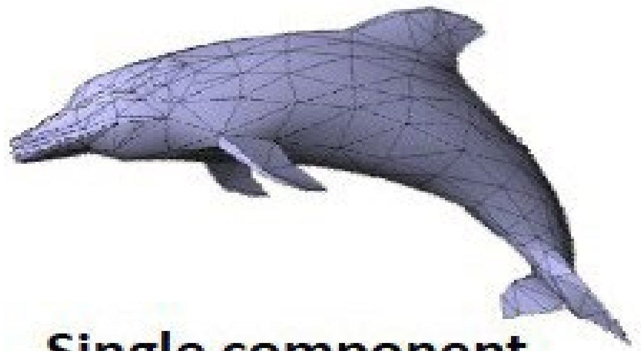


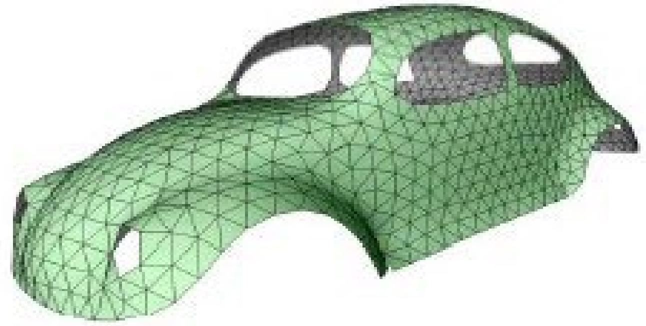


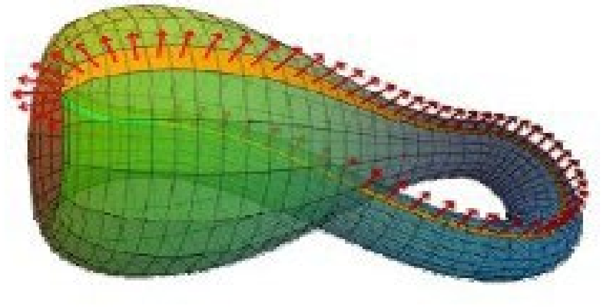
A normal to a surface at a point is the same as a normal to the tangent plane to the surface at the same point.

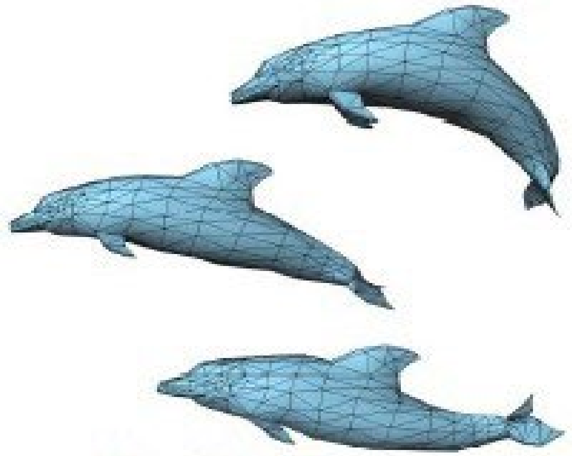
Courtesy Wikipedia

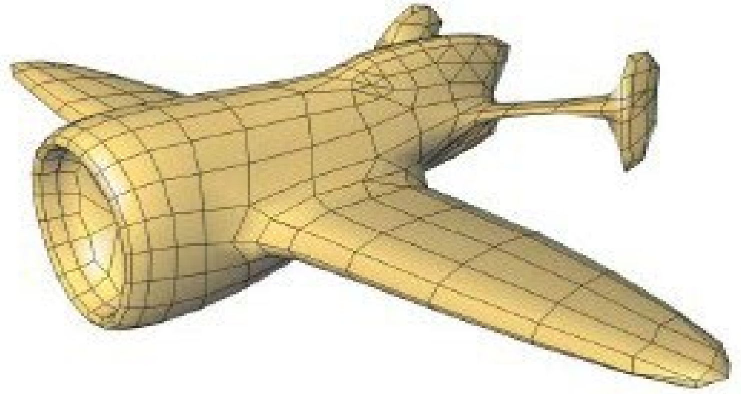
Vocabulary: Surface Mesh Properties

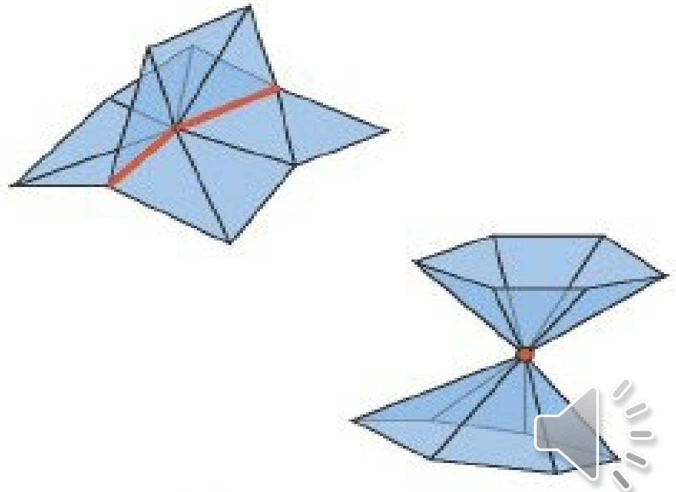

Single component, closed, triangular, orientable manifold


With boundaries


Not orientable

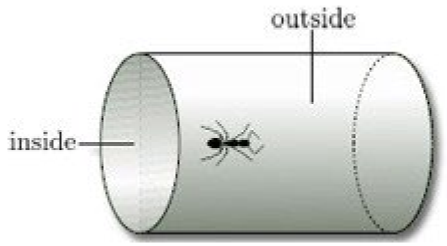

Multiple components

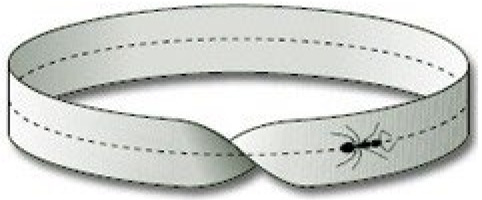

Not only triangles


Non manifold

Orientability

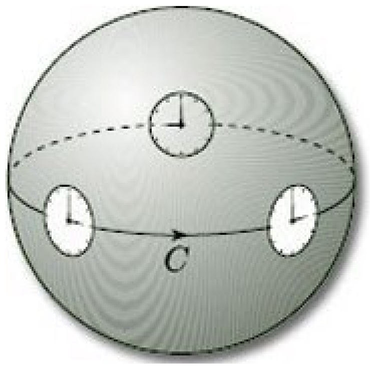
Not that relevant to this course….but it is interesting….

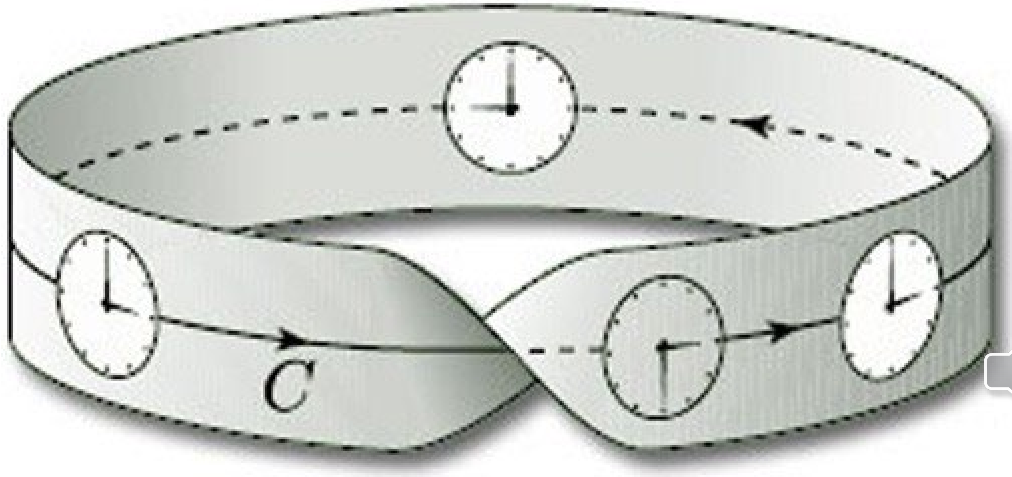




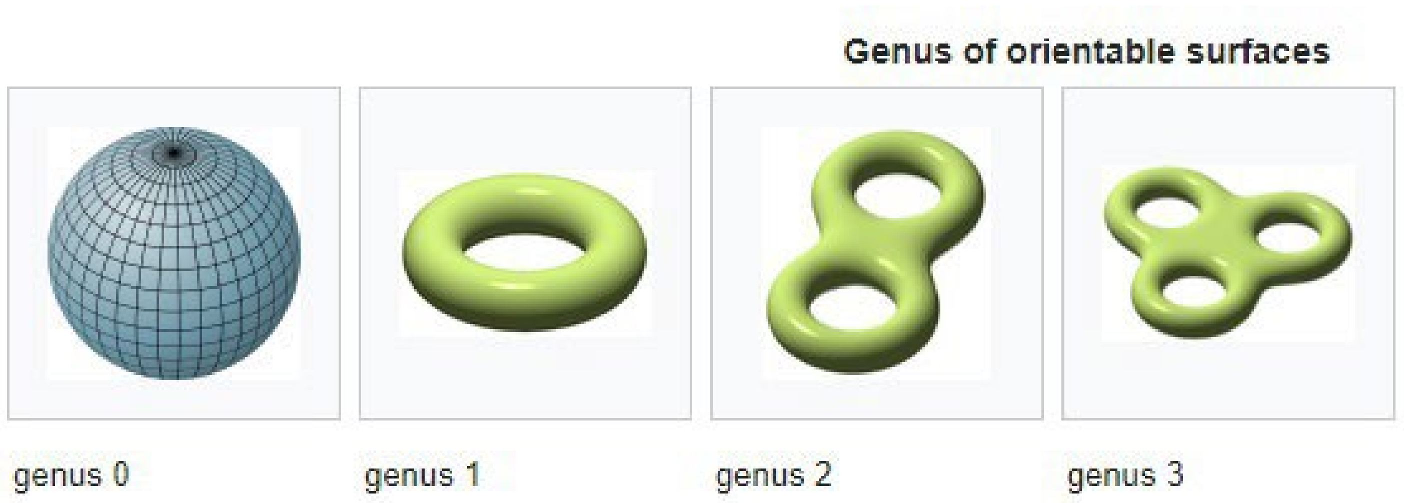
Imagine sliding a clock face around a surface.

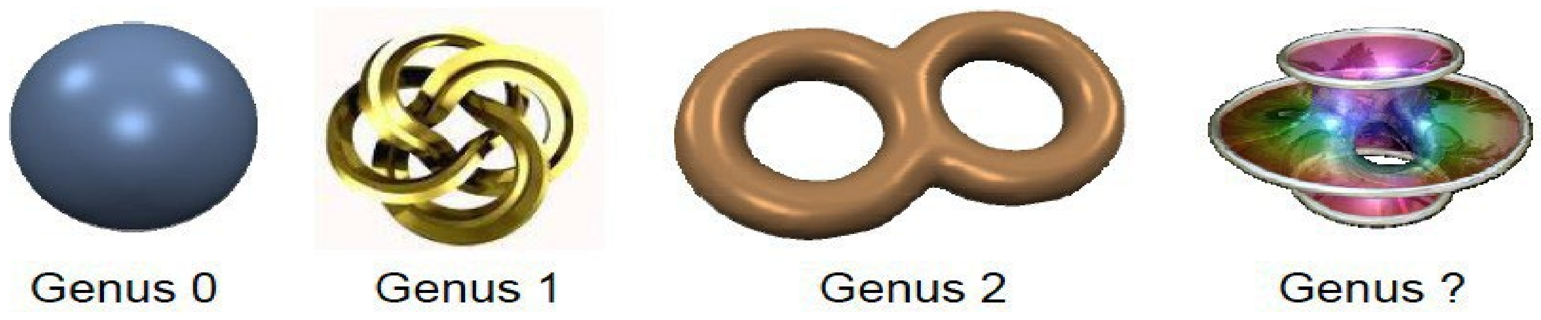
On an orientable surface the clock face will return to starting point and appear the same. (Sphere)
On a non-orientable surface it will be a mirror image of the original clock face (Moebius Strip)
…there is no consistent definition of clockwise on a non-orientable surface.

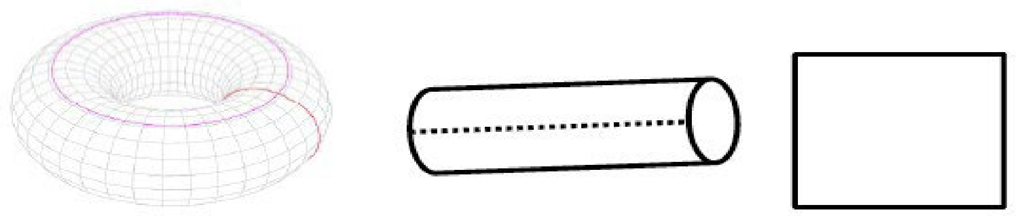




Genus


Genus of orientable surfaces
genus 0
genus 1
genus 2
genus 3


Genus 0
Genus 1
Genus 2
Genus ?



Euler Characteristic

For a closed (no boundary), manifold, connected surface mesh:

$$V-E+F=2(1-G)$$

- V = number of vertices
- E = number of edges
- F = number of faces
- G = genus (number of holes in the surface)

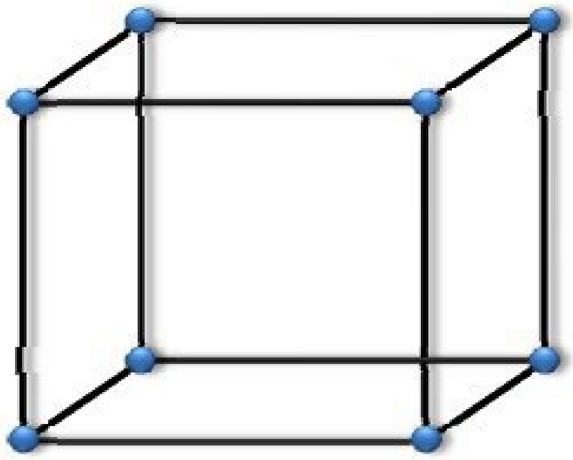
A 2-manifold is a surface (locally like a plane)

Leonhard Euler
1707 – 1783

ILLINOIS

Euler Characteristic for Closed 2-Manifold Polygonal Meshes

$$V + F - E = \boxed{\chi}$$ Euler characteristic

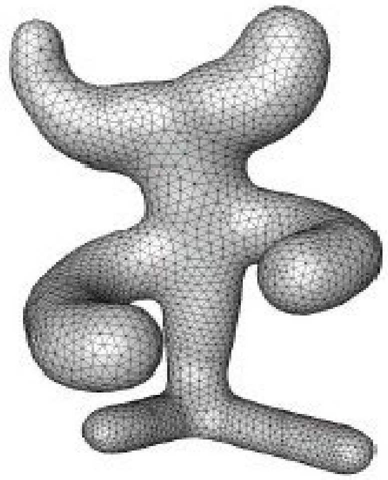


$V = 8$

$E = 12$

$F = 6$

$\chi = 8 + 6 - 12 = 2$



$V = 3890$

$E = 11664$

$F = 7776$

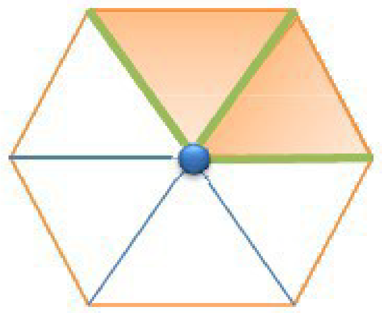
$\chi = 2$

..and if they are triangle meshes

- *Triangle* mesh statistics

  $E \approx 3V$

  $F \approx 2V$

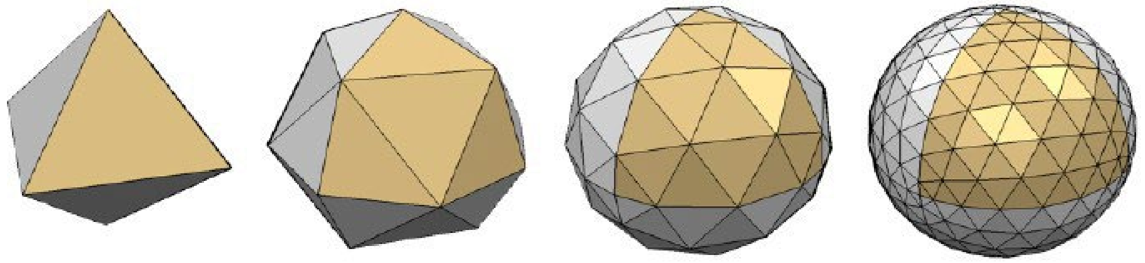


- Avg. valence ≈ 6

  Show using Euler Formula

Aside from being totally interesting on their own, these formulas are useful for computing memory usage

Mesh Data Structures



Need to store

- Geometry
- Connectivity

Can be used as file formats or internal formats

Considerations

- Space
- Efficient operations

Mesh processing has different requirements than rendering

- Example: Deforming a mesh when simulating physics…

Mesh Data Structure: Face Set (STL)

- face:
  - 3 positions

Triangles		
$x_{11}\ y_{11}\ z_{11}$	$x_{12}\ y_{12}\ z_{12}$	$x_{13}\ y_{13}\ z_{13}$
$x_{21}\ y_{21}\ z_{21}$	$x_{22}\ y_{22}\ z_{22}$	$x_{23}\ y_{23}\ z_{23}$
...	...	...
$x_{F1}\ y_{F1}\ z_{F1}$	$x_{F2}\ y_{F2}\ z_{F2}$	$x_{F3}\ y_{F3}\ z_{F3}$

$\boxed{36\ \text{B/f}} = 72\ \text{B/v}$

no connectivity!

Designed in 1987 for stereolithography (3D printing technology)

No explicit information about which vertices are shared by which triangles
Consider how efficiently we could:

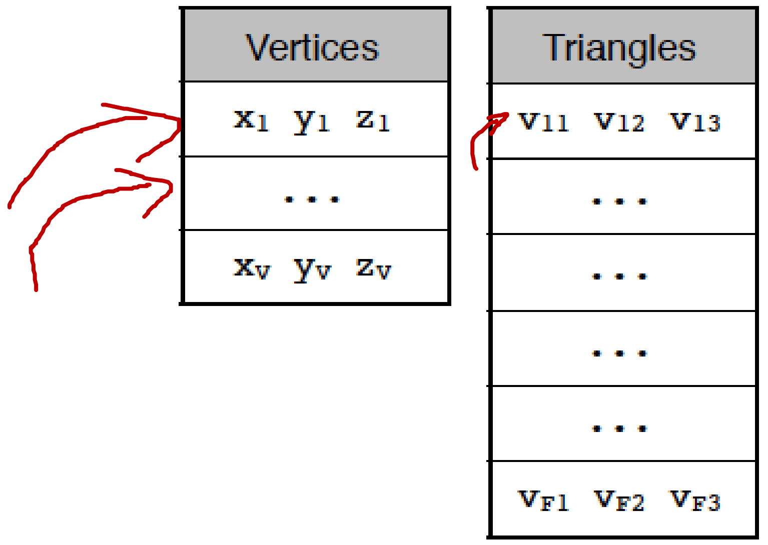
- Deform mesh by moving vertices
- Lookup vertex neighbor information

We will be focusing on triangle meshes…but all of these structures generalize to other polygons…storage requirements will change.

ILLINOIS

Mesh Data Structure: Indexed Face Set (OBJ)

- vertex:
  - position
- face:
  - vertex indices



Vertices
x_1 y_1 z_1
. . .
x_v y_v z_v

Triangles
v_{11} v_{12} v_{13}
. . .
. . .
. . .
. . .
v_{F1} v_{F2} v_{F3}

12 B/v + 12 B/f = 36 B/v

no neighborhood info

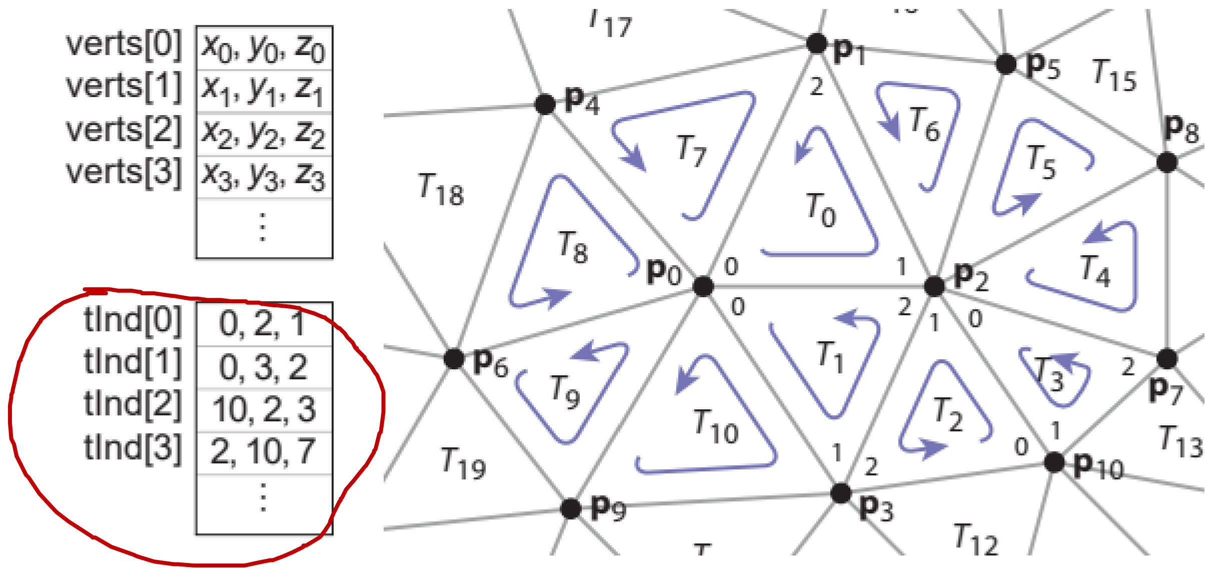
The OBJ file format is a popular storage format for meshes

Developed by Wavefront Technologies…now part of AutoDesk

Text files with .obj extension

```
# List of geometric vertices
v 0.123 0.234 0.345
v …
v …
# List of triangles
f 1 3 4
f 2 4 5
…
```

Indexed Face Set: Example



verts[0]	x_0, y_0, z_0
verts[1]	x_1, y_1, z_1
verts[2]	x_2, y_2, z_2
verts[3]	x_3, y_3, z_3
	⋮

tInd[0]	0, 2, 1
tInd[1]	0, 3, 2
tInd[2]	10, 2, 3
tInd[3]	2, 10, 7
	⋮

For each triangle

The vertices are listed in ccw order when viewed from outside

Many data structures/file formats use this convention

Allows for unambiguous computation of outward facing normal vectors for each triangle

Questions

- How easy is it for artists to work directly with surface meshes?
- How important is it for artists to work easily with meshes?
- What would be an alternative surface representation?

- What are the advantages compared to meshes?

- What are the disadvantages compared to meshes?