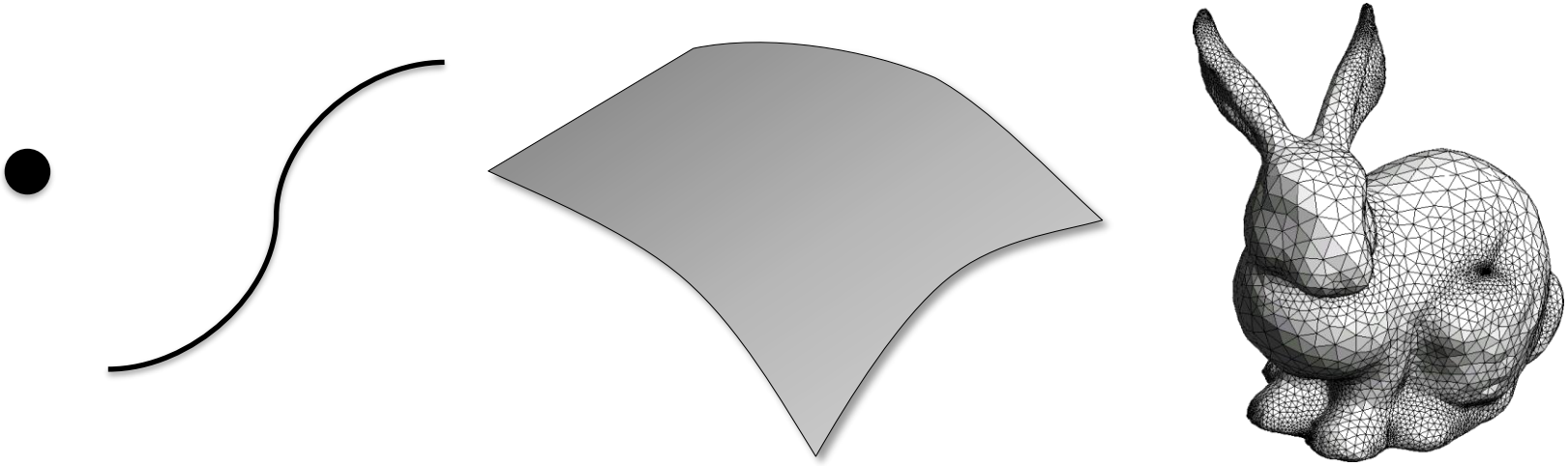
Geometry and Textures I

Prof. Dr. Markus Gross

ETH zürich

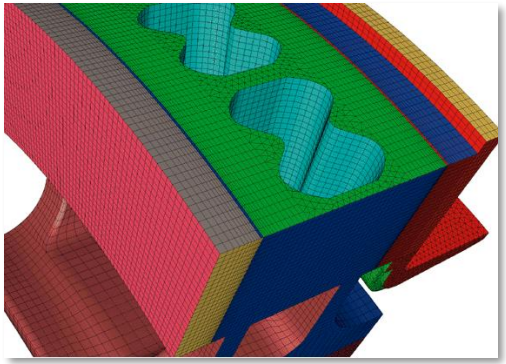
cgl

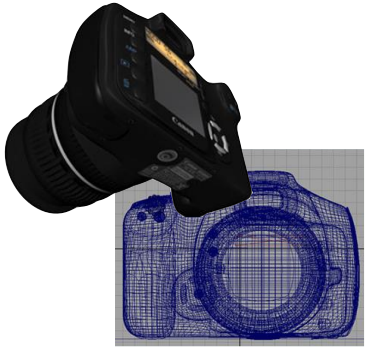
Geometry in Graphics



Applications

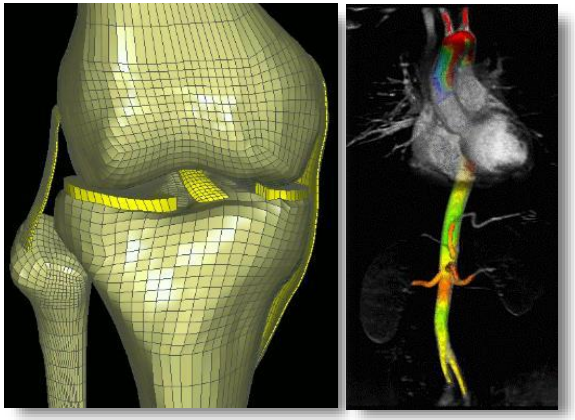
Games/Movies



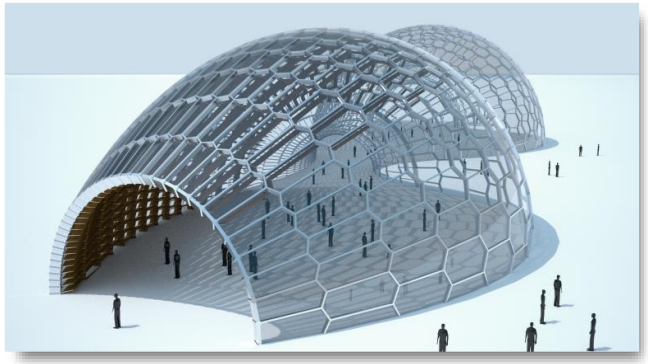


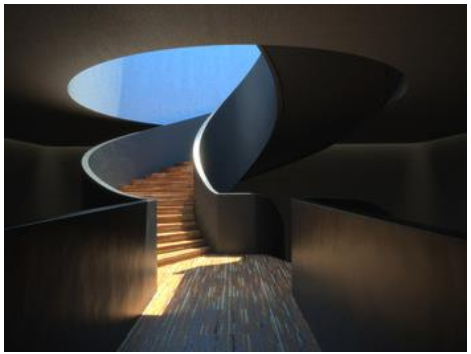
Engineering/Product design

Applications



Medicine/Biology





Architecture

Sources of Geometry

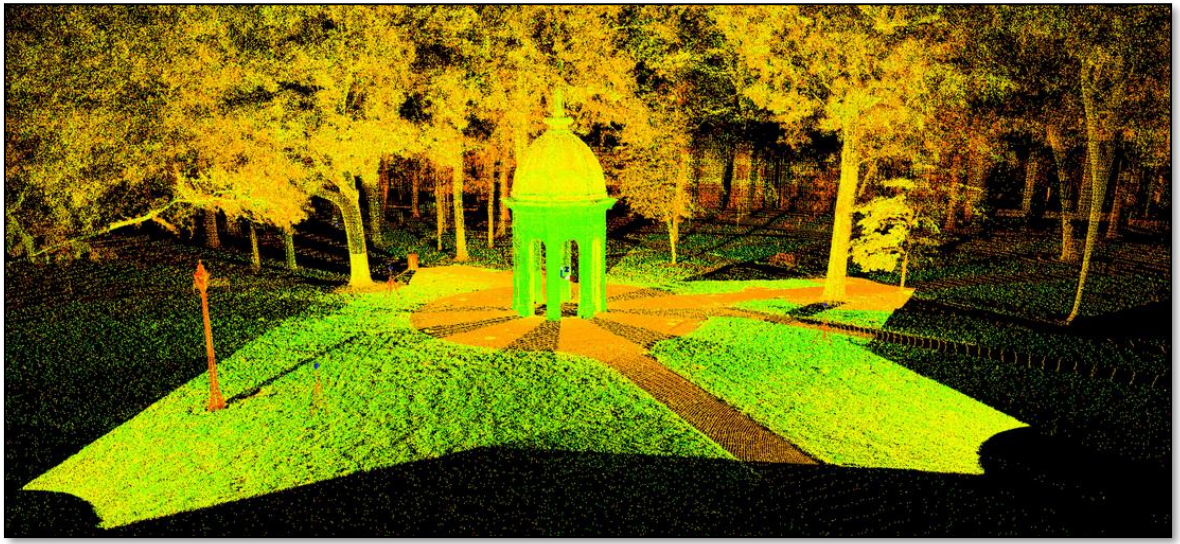
- Acquired real-world objects

3D Scanning

Sources of Geometry

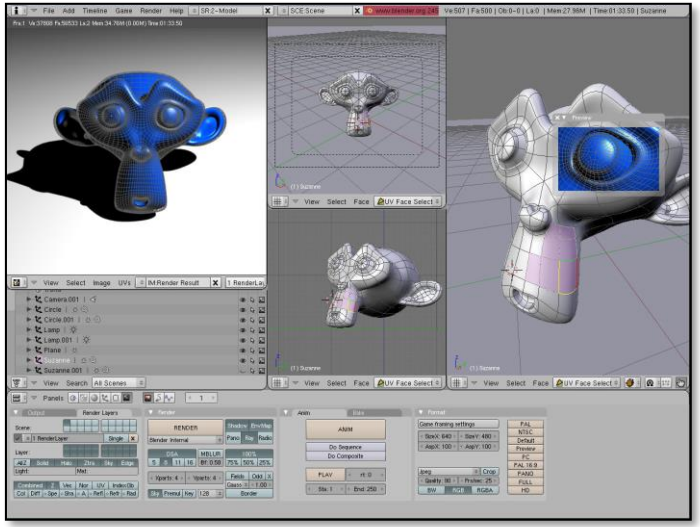
- Acquired real-world objects

Point Clouds



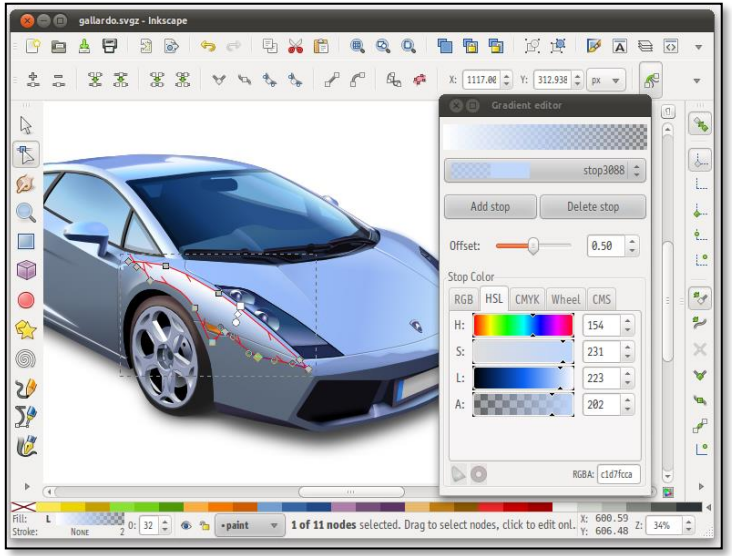
Sources of Geometry

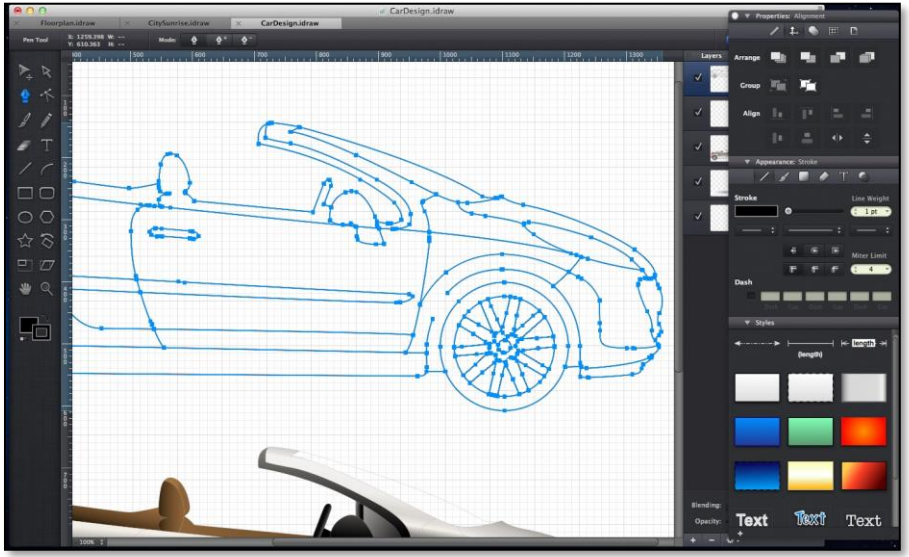
- Digital 3D modeling



Sources of Geometry

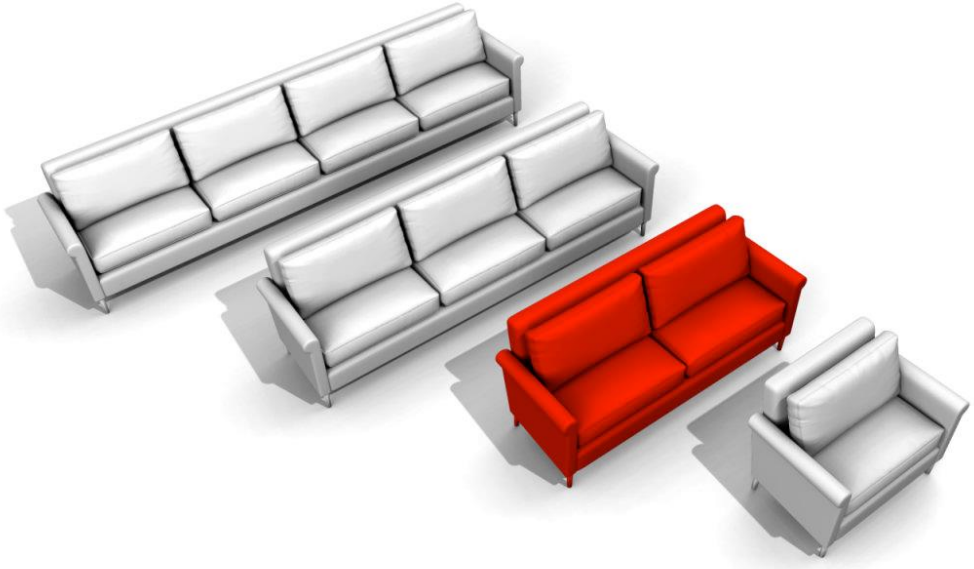
- Digital 3D modeling





Sources of Geometry

- Procedural Modeling

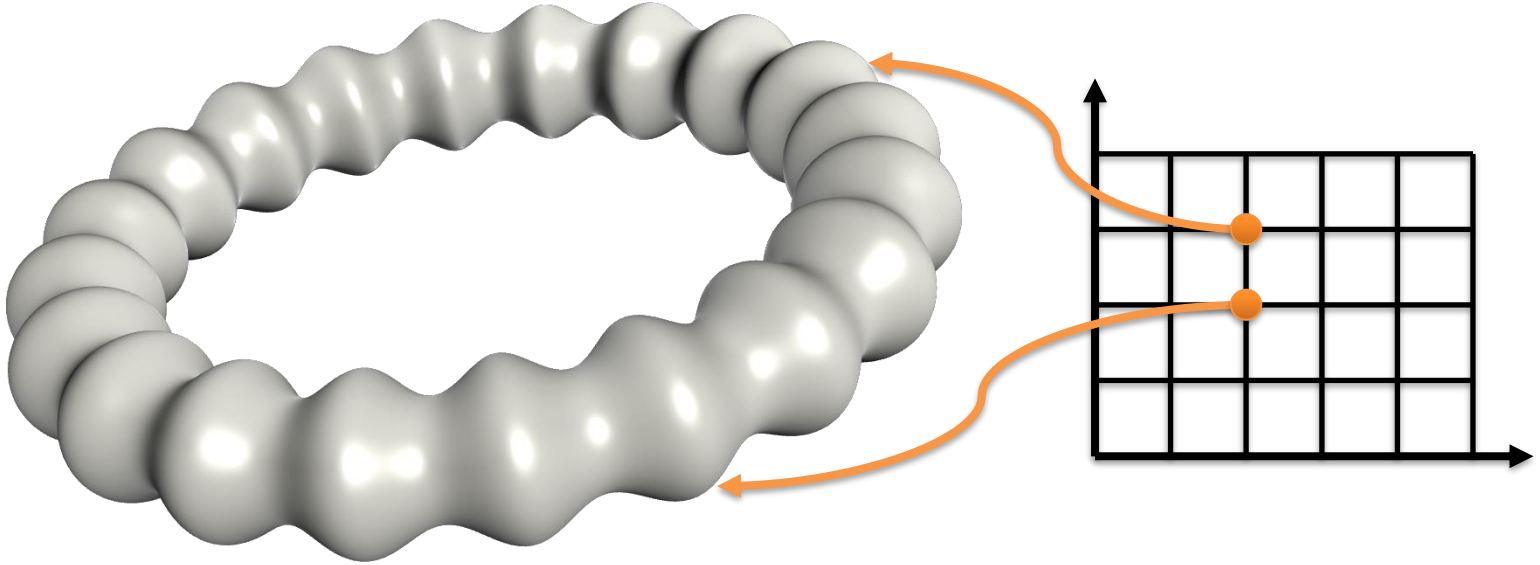


Geometry Representations

- Considerations
  - Storage
  - Acquisition of shapes
  - Creation of shapes
  - Editing shapes
  - Rendering shapes

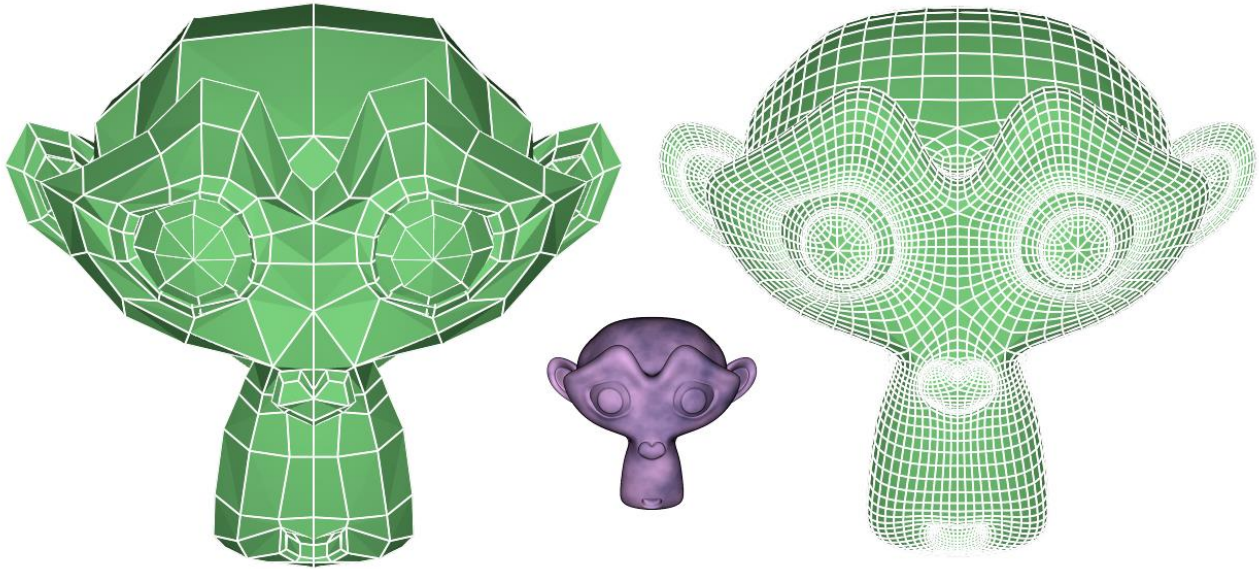
Geometry Representations

- Parametric surfaces



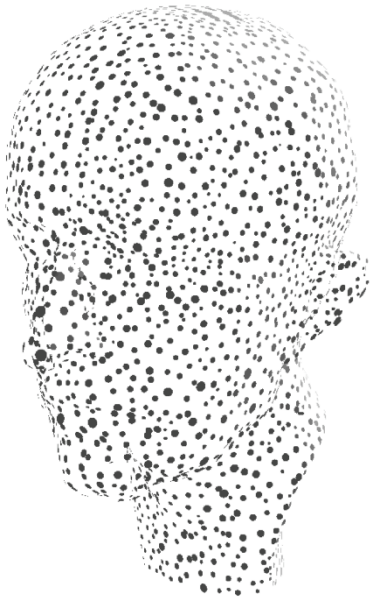
Geometry Representations

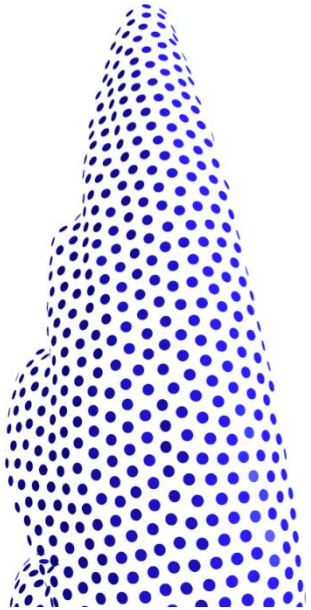
- Subdivision surfaces



Geometry Representations

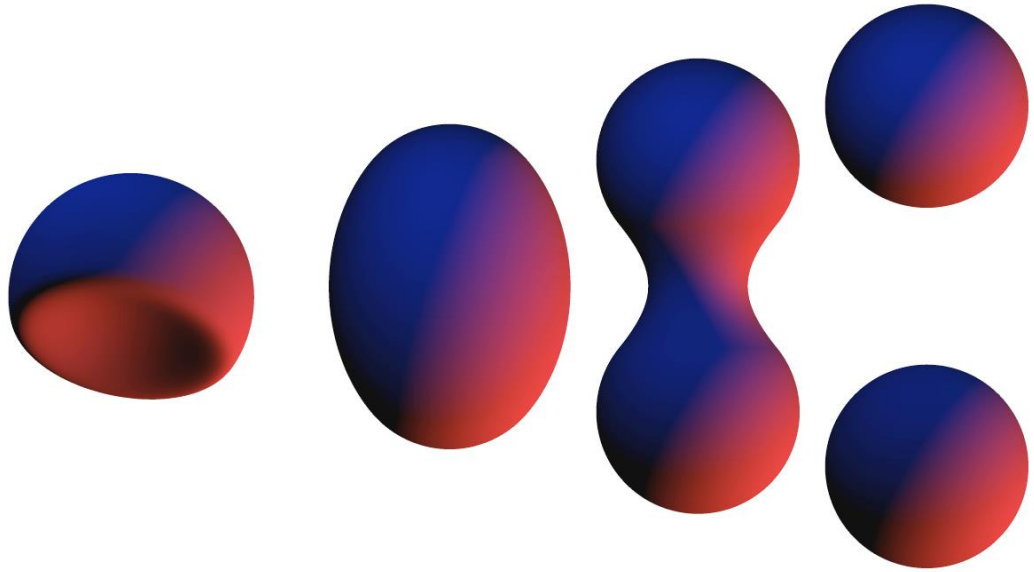
- Point set surfaces



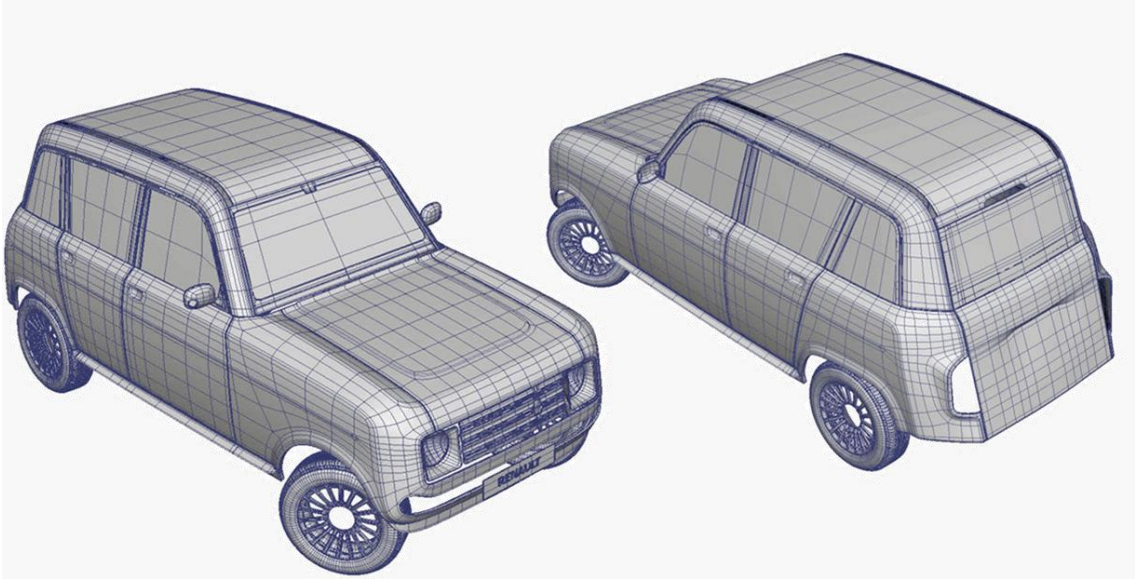
Geometry Representations

- Implicit surfaces



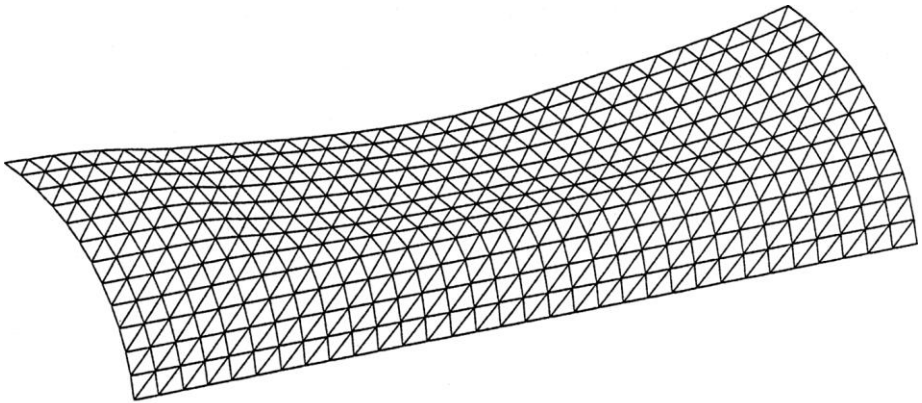
Geometry Representations

- Polygonal Meshes



Geometry Representations

- Polygonal Meshes
  - Boundary of objects
  - Geometry + connectivity

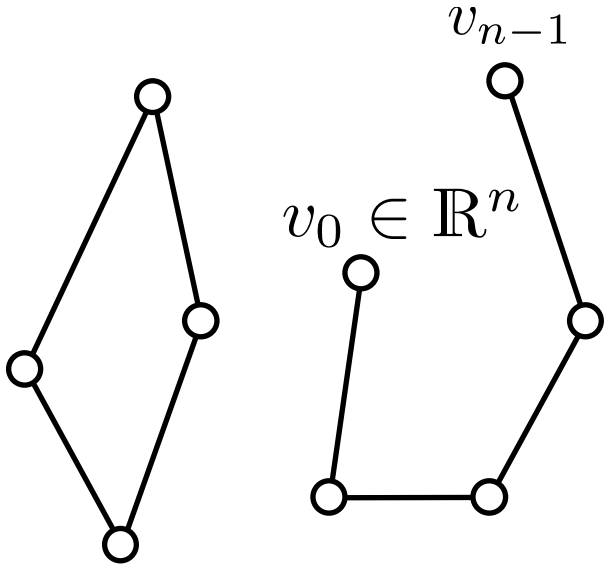


Geometry Representations

- Polygonal Meshes
  - Fast rendering with optimized GPU pipeline

Polygonal Mesh

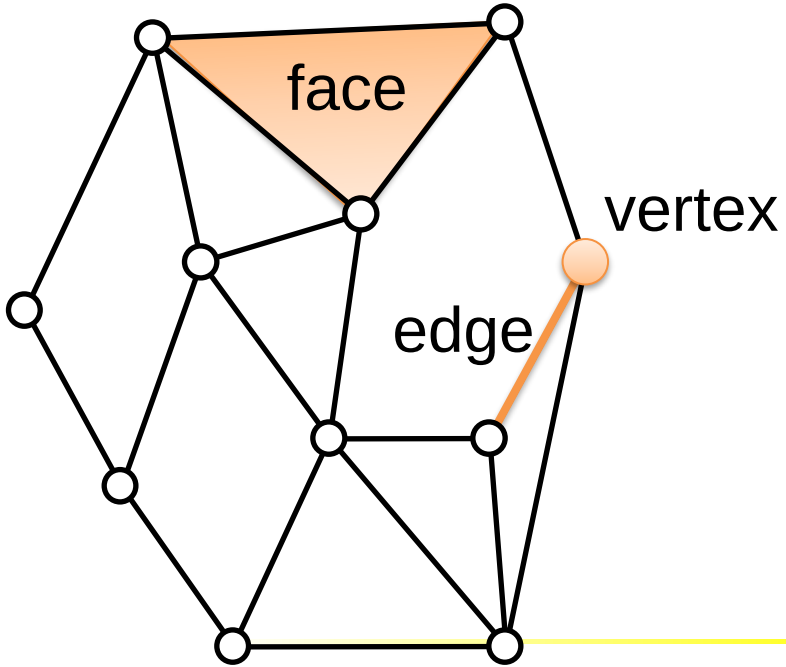
- What is a polygon



- Vertices
  $v_0, v_1, \ldots, v_{n-1}$
- Edges
  $\{(v_0, v_1), \ldots, (v_{n-2}, v_{n-1})\}$
- Planar and non-self-intersecting

Polygonal Mesh

- Set of connected polygons

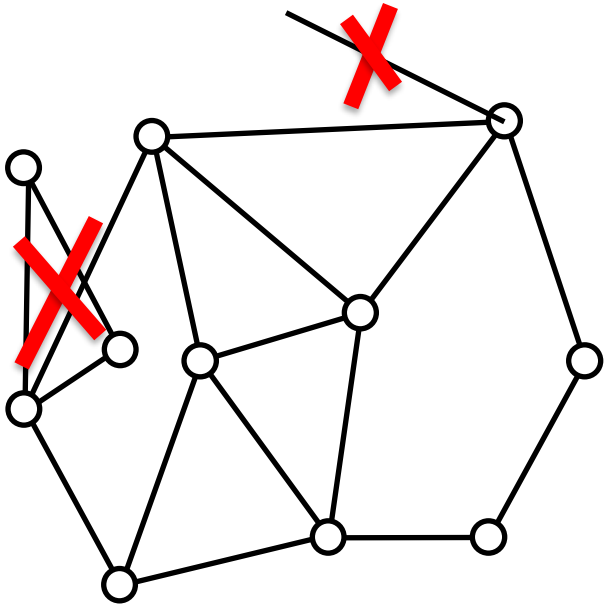


$$M = \langle V, E, F \rangle$$

vertices edges faces

Polygonal Mesh

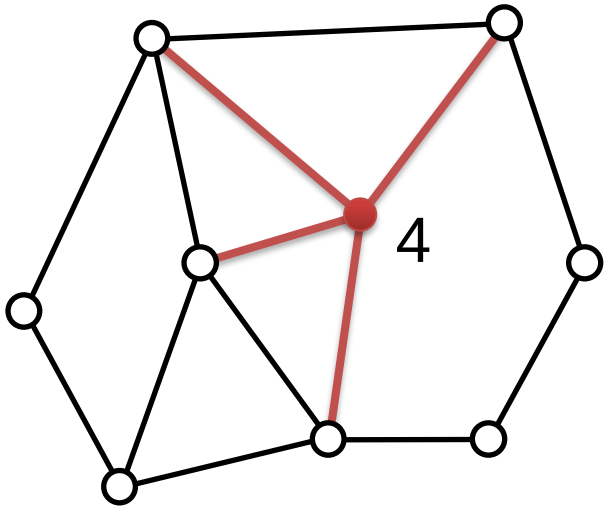
- Properties



- Every edge belongs to at least one polygon
- The intersection of two polygons in M is either empty, a vertex, or an edge

Polygonal Mesh

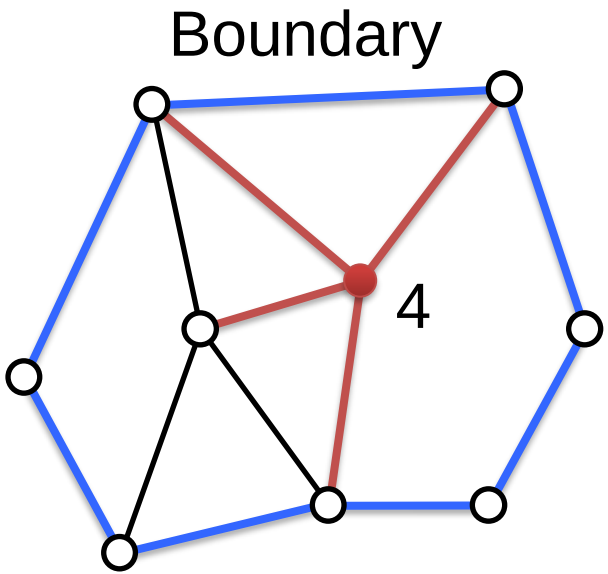
- Definitions



- **Vertex degree (Valence)**: Number of edges incident to a vertex

Polygonal Mesh

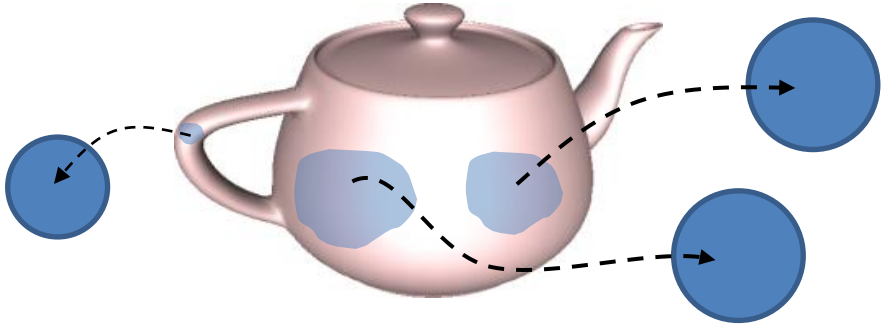
- Definitions



- **Vertex degree (valence)**: Number of edges incident to a vertex
- **Boundary**: the set of all edges that belong to only one polygon

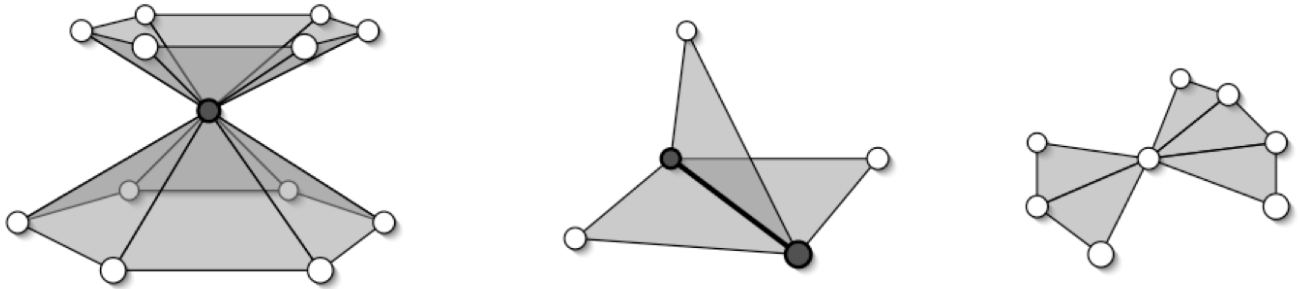
Manifolds

- Surface locally homeomorphic to a disk
- Closed manifolds divides space into two



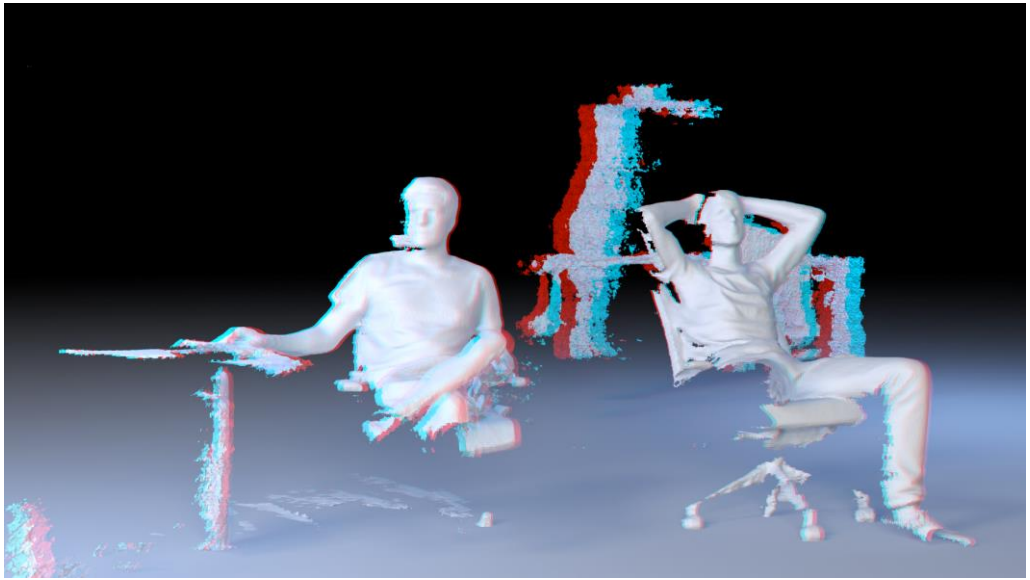
Manifold Mesh

- In a manifold mesh, not allowed:



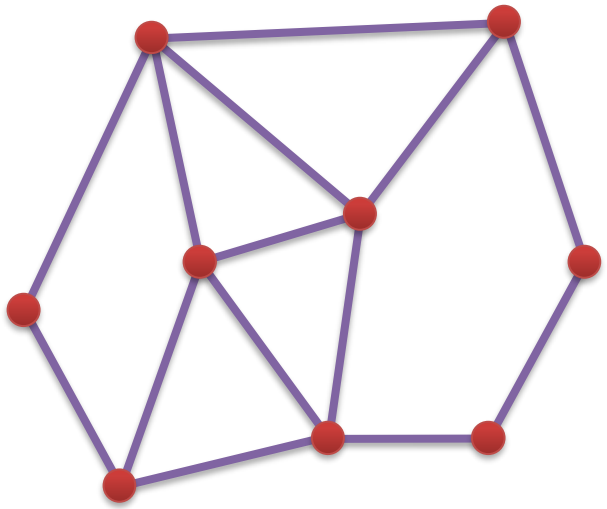
Manifold Mesh

- Real-world data is often non-manifold



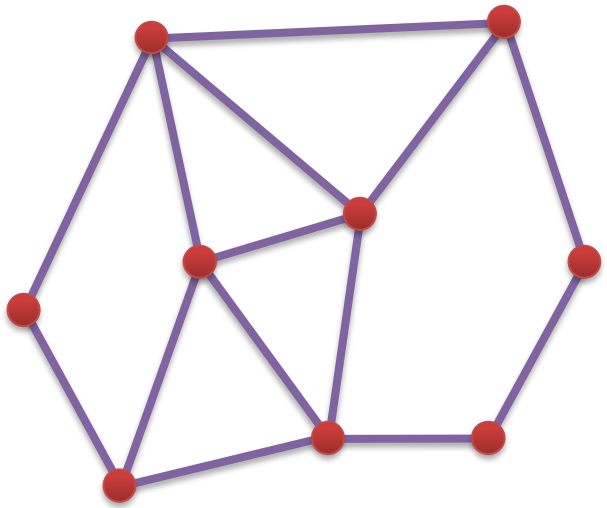
Mesh Data Structures

- Store geometry & topology
  - **Geometry**: vertex locations
  - **Topology**: how vertices are connected (edges/faces)
  - **Attributes**: Normal, color, etc.



Mesh Data Structures

- Operations to be supported
  - Rendering

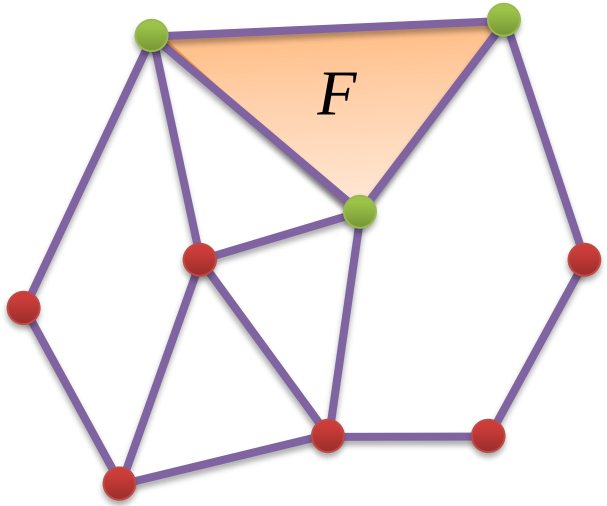


Mesh Data Structures

- Operations to be supported
  - Rendering
  - Geometry queries

    Example:

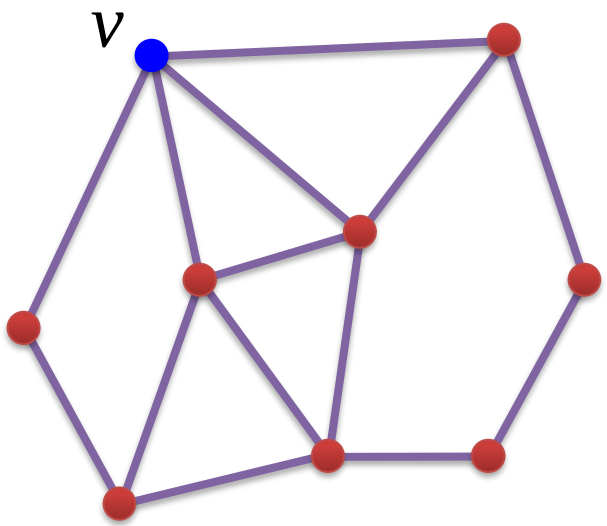
    What are the vertices of face F ?



Mesh Data Structures

- Operations to be supported
  - Rendering
  - Geometry queries
  - Modifications

    Example:
    Remove vertex v.



Mesh Data Structures

- Triangle List

Triangles		
Vertex Coord.	Vertex Coord.	Vertex Coord.
(x_0, y_0, z_0)	(x_1, y_1, z_1)	(x_2, y_2, z_2)
(x_3, y_3, z_3)	(x_4, y_4, z_4)	(x_5, y_5, z_5)
...	...	...

- Simple
- No connectivity
- Redundant
- STL file format

Mesh Data Structures

- Indexed Face Set

Triangles		
Vertex Index	Vertex Index	Vertex Index
1	2	0
...	...	...

Vertices	
Index	Coord.
0	(x_0, y_0, z_0)
1	(x_1, y_1, z_1)
2	(x_2, y_2, z_2)
…	...

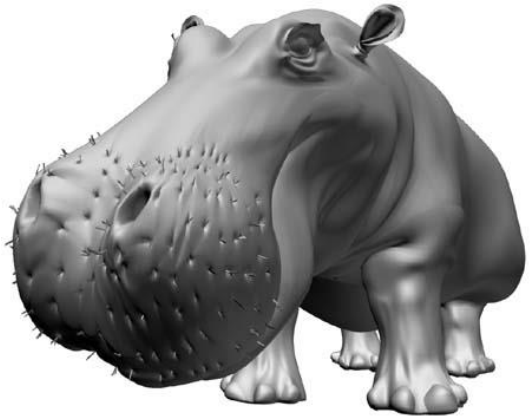
Mesh Data Structures

- Indexed Face Set
  - Avoids redundancy
  - OBJ, OFF, WRL file formats
  - Stores connectivity, but:
    - Costly geometric queries
    - Costly mesh modifications

Mesh Textures

- Enhancing realism via textures

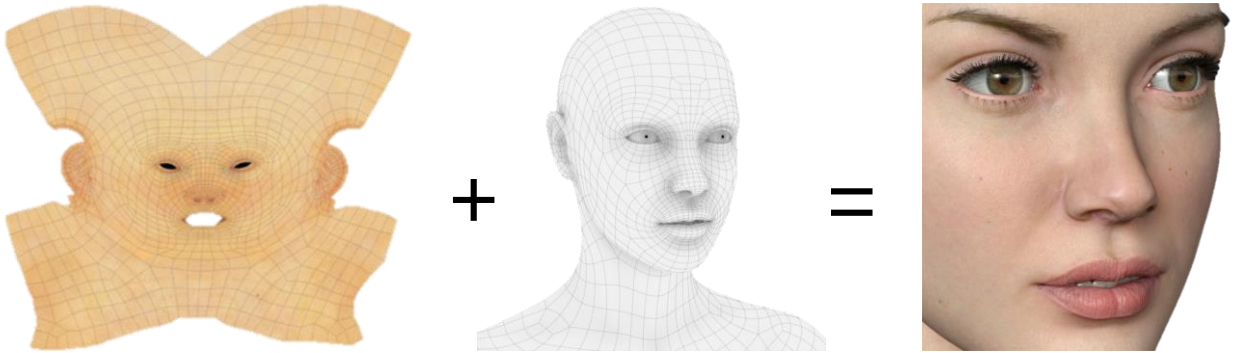
Lighting & Shading



Texture Mapping

Texture Mapping

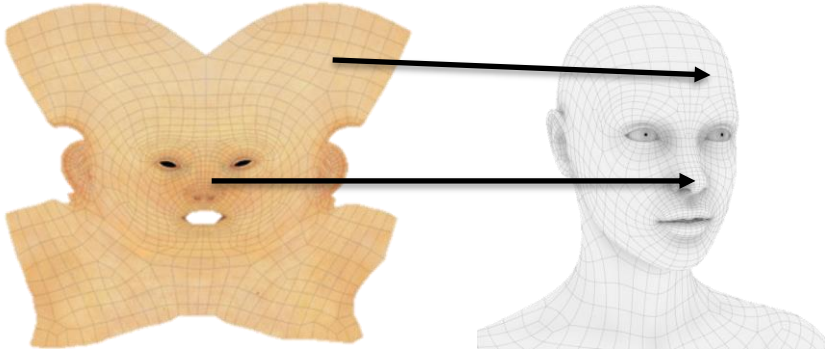
- Enhance details without increasing geometric complexity



Texture Geometry Rendering

Texture Mapping

- Issues
  - Mapping between texture and geometry



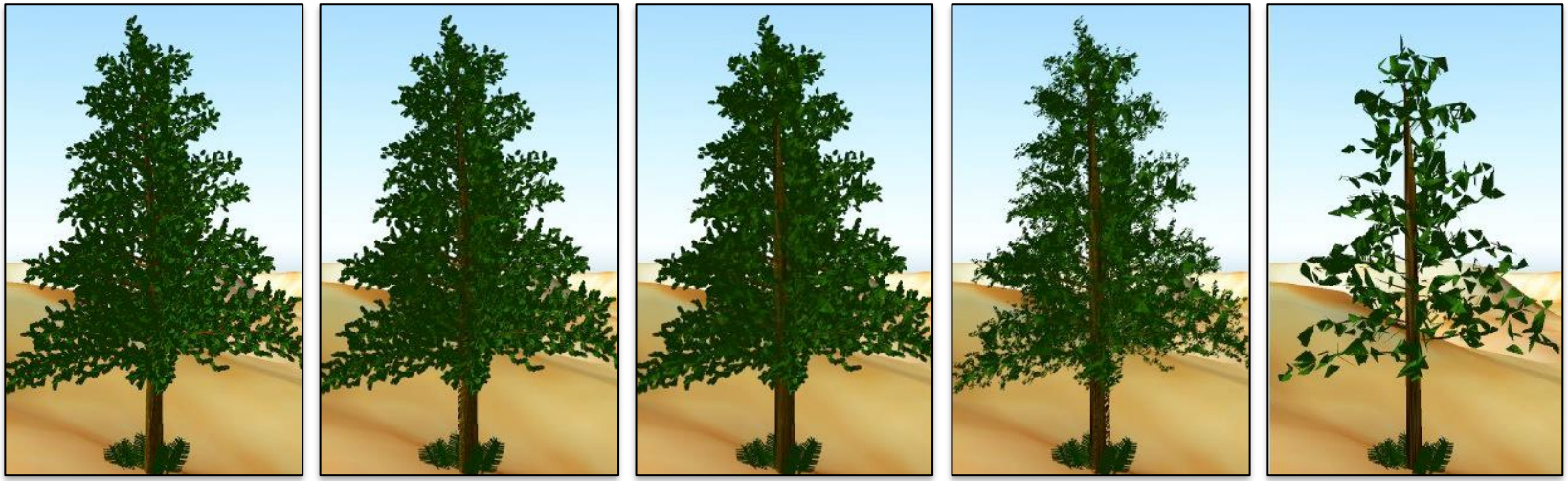
Texture Geometry

Texture Mapping

- Issues
  - Anti-aliasing and filtering

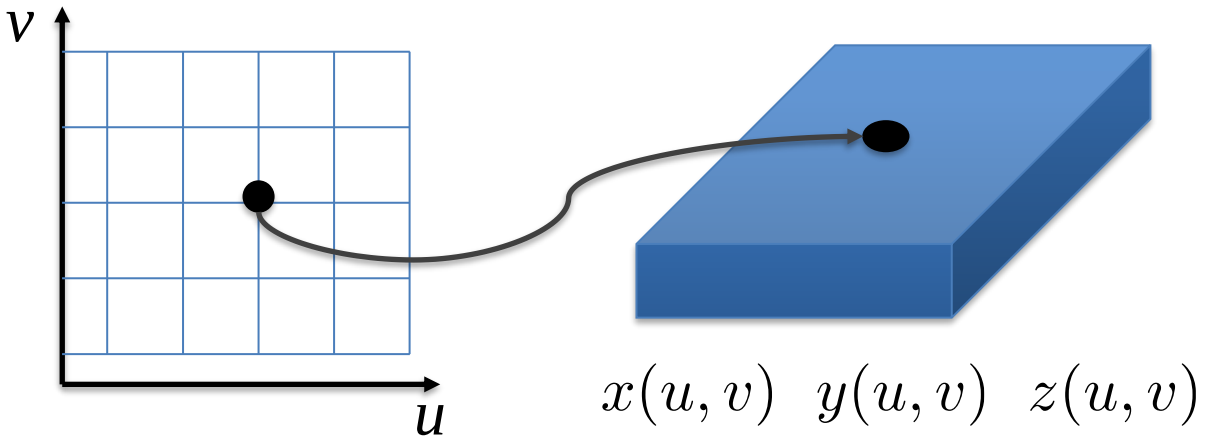
Texture Mapping

- Issues
  - Level-of-detail



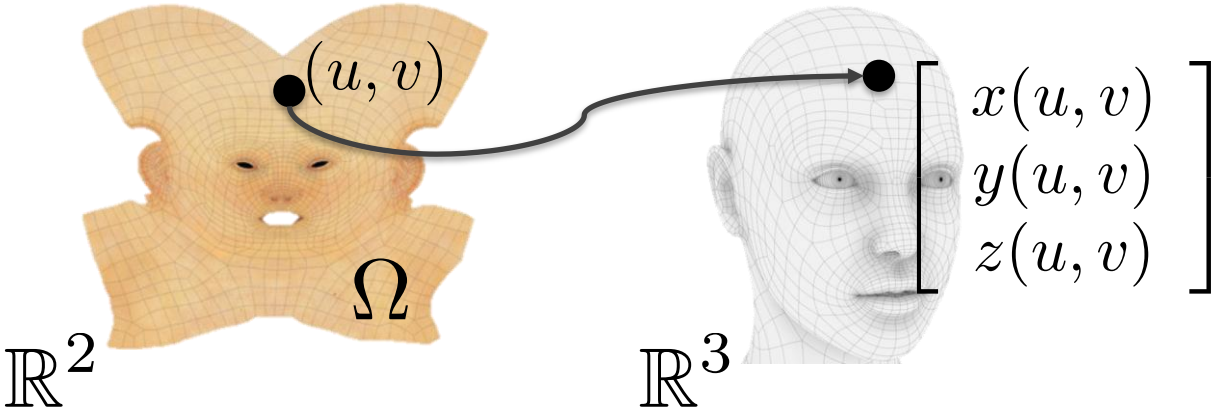
Texture Mapping

- One-to-one mapping between texture and geometry



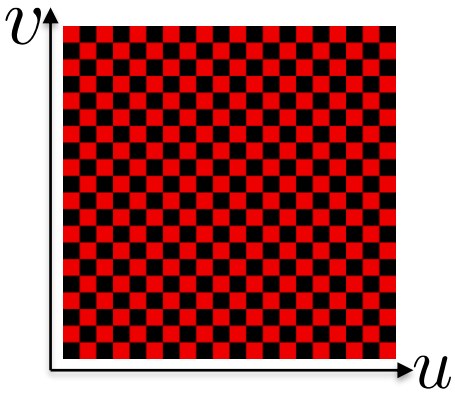
Texture Mapping

- Requires parameterization

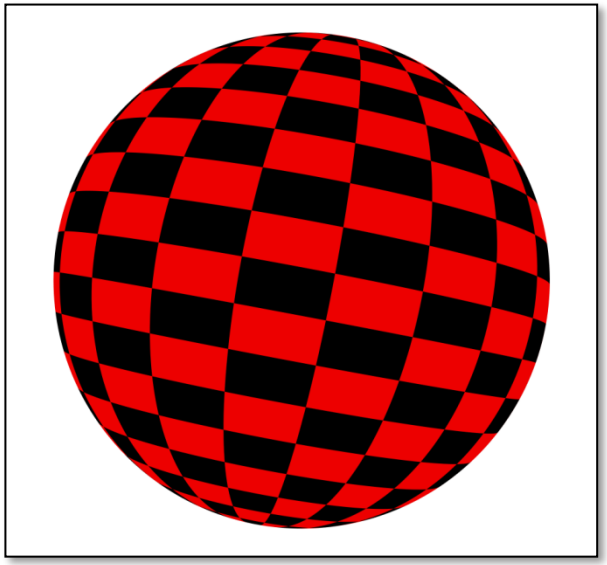


Parameterization

- Example: sphere

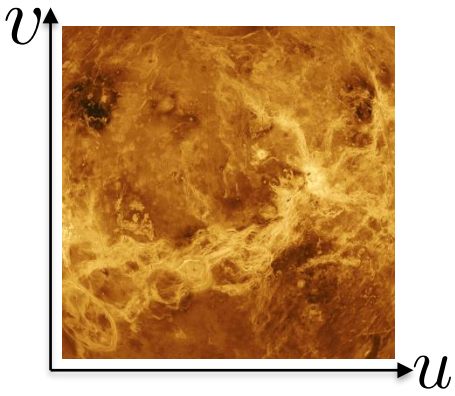


$$\begin{bmatrix} u \\ v \end{bmatrix} \rightarrow \begin{bmatrix} \sin(u)\sin(v) \\ \cos(v) \\ \cos(u)\sin(v) \end{bmatrix}$$



Parameterization

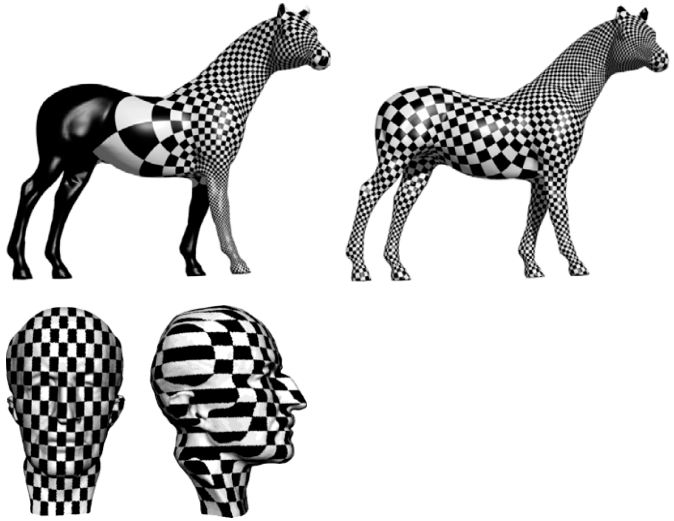
- Example: sphere



$$\begin{bmatrix} u \\ v \end{bmatrix} \rightarrow \begin{bmatrix} \sin(u)\sin(v) \\ \cos(v) \\ \cos(u)\sin(v) \end{bmatrix}$$

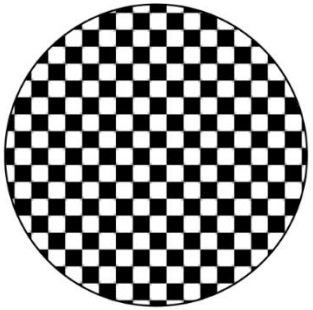
Parameterization

- Desirable properties
  - Low distortion
  - Bijective mapping
  - Efficient to compute



Parameterization

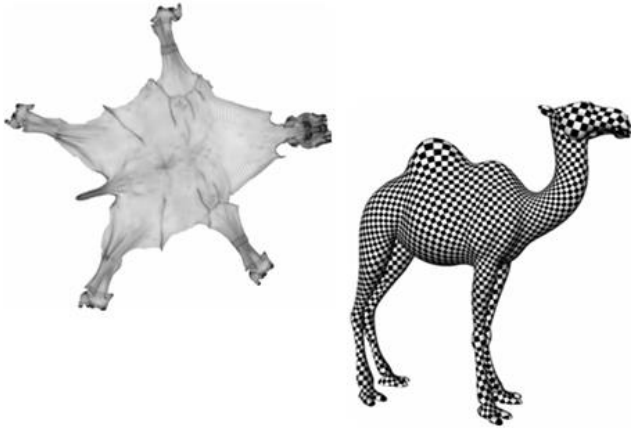
- Desirable properties – low distortion

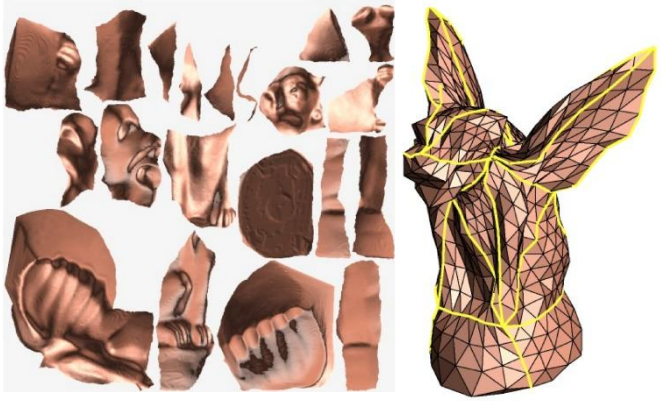


Texture Map

Texture Mapping

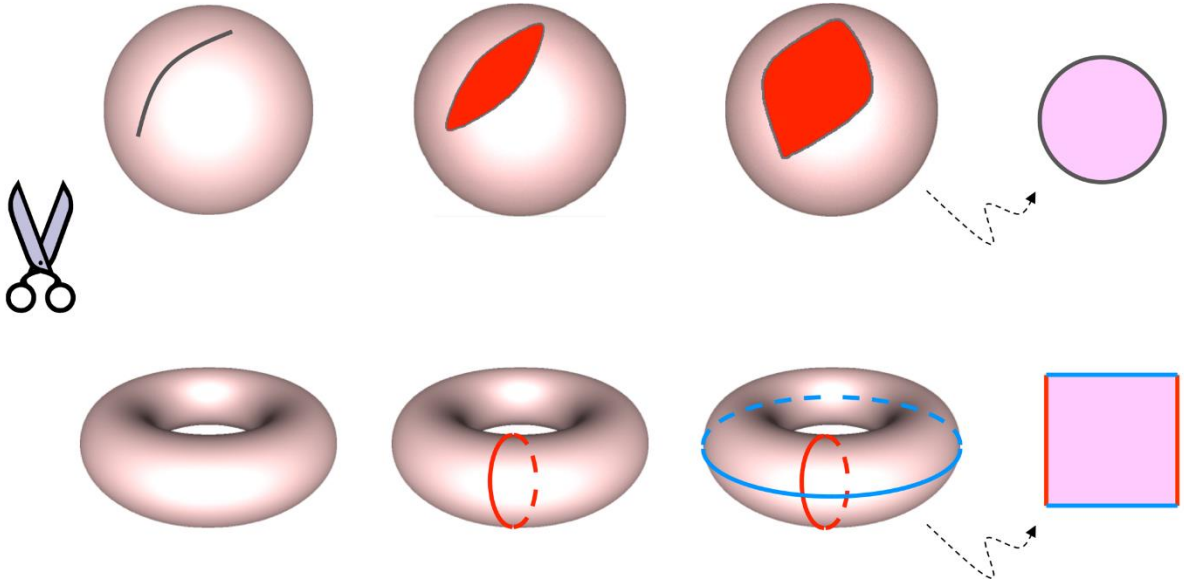
- Additional Issues
  - Finding cuts
  - Texture atlases





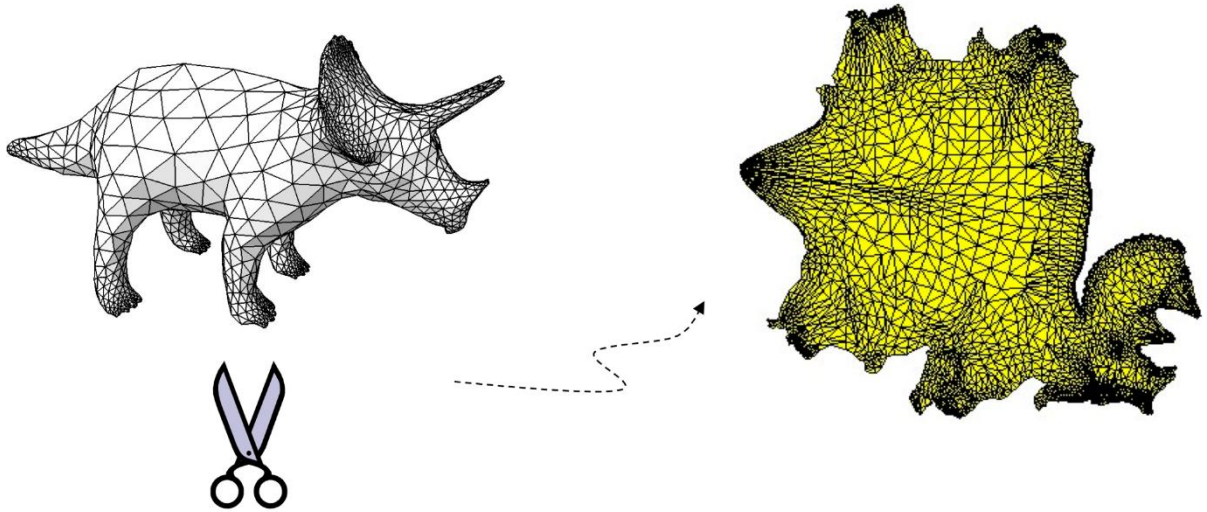
Texture Mapping

- Additional Issues - Finding cuts



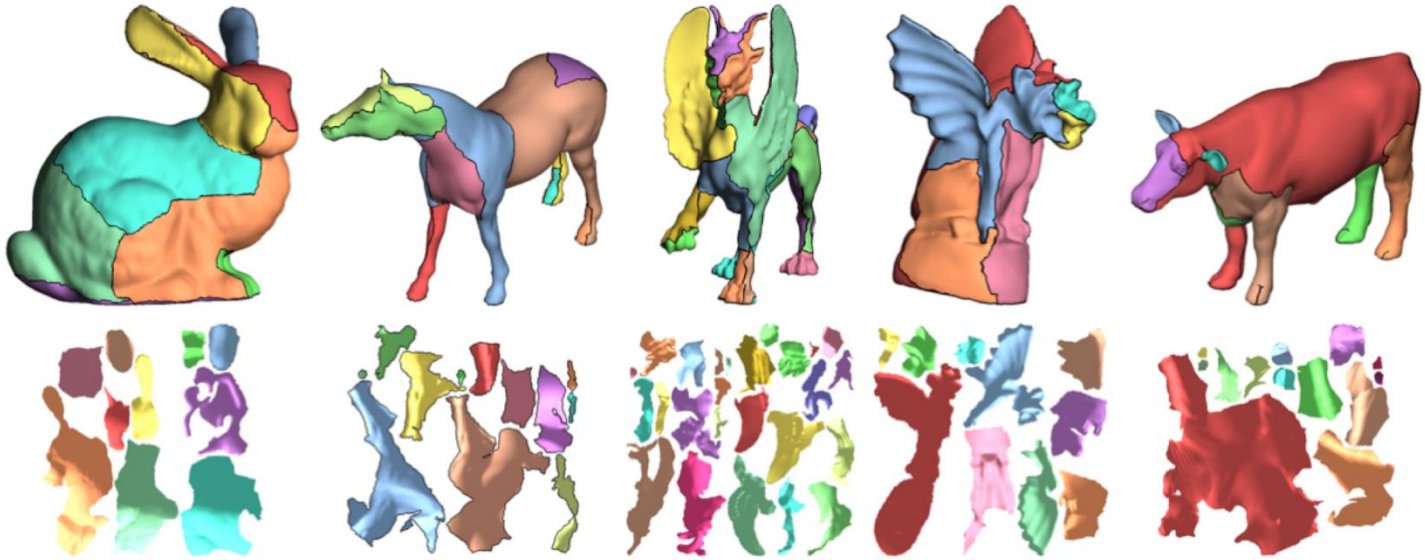
Texture Mapping

- Additional Issues - Finding cuts



Texture Mapping

- Additional Issues – Texture atlases



Texture Mapping

- OBJ Files

```
v 0.131171 -0.113469 0.178314
v 0.130945 -0.114951 0.182474
…
vt 0.538446 0.4275
vt 0.550132 0.41427
…
vn 0.609697 0.486474 0.625789
vn 0.799934 0.334347 0.498315
…
f 22/209/22 220/210/220
221/211/221
f 21/213/21 219/214/219
220/210/229
…
```

Vertex positions

Texture coordinates

Normals

Faces (triangles)
coordNr/texNr/normalNr

Texture Mapping

- OpenGL

Load and bind texture

```
loadImage(&texture_data);
glGenTextures(1, &texId);
glBindTexture(GL_TEXTURE_2D, texId);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB,
        w, h, 0, GL_RGB,
                 GL_UNSIGNED_BYTE,
        texture_data);
...
```

Fragment shader: texture lookup

```
uniform sampler2D texWood;
varying vec2 texCoords;      // [u, v]

void main(void) {
    gl_FragColor = texture2D(texWood,
    texCoords);
}
```

Texture Mapping

- WebGL

Load and bind texture

```
const texture = gl.createTexture();
gl.bindTexture(gl.TEXTURE_2D, texture);
const image = new Image();
image.onload = () => {
    gl.bindTexture(gl.TEXTURE_2D, texture);
    gl.texImage2D( gl.TEXTURE_2D, level,
    internalFormat, srcFormat, srcType,
    image);
    gl.generateMipmap(gl.TEXTURE_2D)};
```

Fragment shader: texture lookup

```
uniform sampler2D texWood;
varying vec2 texCoords;     // [u, v]

void main(void) {
    gl_FragColor = texture2D(texWood,
    texCoords);
}
```