Administrivia

• Google form distributed for grading issues

• Final work outlined soon
  • Final homework
  • Final midterm
  • Final project grading standards
Today’s question

How to represent objects
Polygonal meshes

• Standard representation of 3D assets

• Questions:
  • What data and how stored?
  • How generate them?
  • How color and render them?
Data structure

• Geometric information
  • Vertices as 3D points

• Topology information
  • Relationships between vertices
  • Edges and faces
Normals and shading – shading equation

- Light equation
  - k terms – color of object
  - L terms – color of light
- Ambient term - $ka \cdot La$
  - Constant at all positions
- Diffuse term - $kd \cdot (n \cdot l)$
  - Related to light direction
- Specular term - $(v \cdot r)^q$
  - Related to light, viewer direction

$$L_o = k_a L_a + \left( k_d (n \cdot l) + k_s (v \cdot r)^q \right) \frac{L_i}{r^2}$$
Phong exponent

- Powers of $\cos (v \cdot r)^q$
  - $v$ and $r$ normalized
- Tightness of specular highlights
- Shininess of object
Normals and shading

- Face normal
  - One per face

- Vertex normal
  - One per vertex. More accurate

- Interpolation
  - Gouraud: Shade at vertices, interpolate
  - Phong: Interpolate normals, shade
Texture mapping

• Vary color across figure
• $ka$, $kd$ and $ks$ terms

• Interpolate position inside polygon to get color

• Not trivial!
• Mapping complex
Bump mapping

• “Texture” map of
  • Perturbed normals (on right)
  • Perturbed height (on left)
Summary – full polygon mesh asset

• Mesh can have vertices, faces, edges plus normals
• Material shader can have
  • Color (albedo)
  • Phong coefficient q
  • Normal map
  • Texture map
  • Bump map
  • Height map
How create 3D asset?

• Model by hand
• Model by procedure
• Model by scanning

• Mix all three
  • By hand control B-spline surface procedure
  • Take pictures for texture map, bump map
Constructive Solid Geometry (CSG)

- Volume based
- Supports physical and simulation of objects
- Heavily used in industry for precision and flexibility
- Can output polygonal mesh for Unity asset
Boolean operations on primitives

• Union
• Intersection
• Difference
• (and scaling)

• Rectangular blocks
• Spheres
• Cylinders
Easy CSG intro: Tinkercad

• [https://www.tinkercad.com](https://www.tinkercad.com)

• Free
• Easy
• Online tutorials
• Can add own procedural object code in Javascript!
CSG tree

• Unevaluated CSG object represented as tree

• How determine if point is inside object?
CSG tree

• Recursive procedure

```cpp
bool isMember(Point p, CSGnode u) {
    if (u.isLeaf)
        return u.primitiveMemberTest(p);
    else if (u.isUnion)
        return isMember(p, u.left) || isMember(p, u.right);
    else if (u.isIntersect)
        return isMember(p, u.left) && isMember(p, u.right);
    else if (u.isDifference)
        return isMember(p, u.left) && !isMember(p, u.right);
}
```
CSG problems: boundary issues

- Operation produces 2d glitch
- ??

2-dimensional

\[
\begin{align*}
A & \quad B \\
(a) & \quad (b) & \quad (c)
\end{align*}
\]
CSG problems: boundary issues

• Operation produces 2d glitch

• Definitions
  • Interior int(A) – surrounded by A
  • Exterior ext(A) – no A adjacent
  • Boundary bnd(A) – adjacent to both
  • Closure(A) = int(A) union bnd(A)

• A* = closure(interior(A))

• A op* B = closure(int(A op B))
Polygonal meshes

• Represents boundary of object

• 2D manifold
  • Neighborhood of vertex is 2d

• Constraints:
  • No t-junctions
  • Only 2 faces/edge
  • No points inside polygon
Meshlab

• Polygonal mesh editor
• Free
• View, edit, clean up meshes
• Many sophisticated algorithms
Meshes as planar graphs

- Euler’s formula
- \( v - e + f = 2 \)
Meshes as planar graphs

• Euler’s formula
• \( v - e + f = 2 \)

• Gives upper bounds on # of edges and faces

![Diagram](image)
Data structure again

• Face—vertex representation

• What can you find easily?
Data structure again

• Face—vertex representation

• What can you find easily?
  • Traverse vertices on face
  • Traverse faces from vertex

• What’s hard to find?
Data structure again

• Face—vertex representation
  
  • What can you find easily?
    • Traverse vertices on face
    • Traverse faces from vertex
  
  • What’s hard to find?
    • Adjacent faces?
    • Traverse vertices nearby systematically
Winged edge representations

- DECL - doubly-connected edge list
- Stores directed half-edges
- Flexible, supports easier updates
Winged edge representations

• Vertex v has coordinates plus one link to incident edge
• Face f has link to one half edge
• Edge (origin u, destination v) has
  • e.org: e’s origin
  • e.twin: e’s opposite twin half-edge
  • e.left: the face on e’s left side
  • e.next: the next half-edge after e in counterclockwise order about e’s left face
  • e.prev: the previous half-edge to e in counterclockwise order about e’s left face (that is, the next edge in clockwise order).
Winged edge representations

• What is ...

• e.dest: e’s destination vertex
Winged edge representations

• What is ...

• e.dest: e’s destination vertex

  e.dest ← e.twin.org
Winged edge representations

• What is ...

• e.right: the face on e’s right side
Winged edge representations

• What is ...

• e.right: the face on e’s right side

\[ e.\text{right} \leftarrow e.\text{twin.left} \]
Winged edge representations

• What is ...

• e.onext: the next half-edge that shares e’s origin that comes after e in counterclockwise order

\[ e.\text{onext} \leftarrow e.\text{prev.twin} \]
Winged edge representations

• What is ...

• the previous half-edge that shares e’s origin that comes before e in counter-clockwise order

\[ \text{e.oprev} \leftarrow \text{e.twin.next} \]
Winged edge representations

- Question: how traverse f in ccw order?
Winged edge representations

• Question: how traverse f in ccw order?

```c
faceVerticesCCW(Face f) {
    Edge start = f.incident;
    Edge e = start;
    do {
        output e.org;
        e = e.next;
    } while (e != start);
}
```
Winged edge representations

• Question: how traverse all vertices that are neighbors of v in cw order?
Winged edge representations

• Question: how traverse all vertices that are neighbors of v in cw order?

```cpp
def vertexNeighborsCW(Vertex v):
    Edge start = v.incident;
    Edge e = start;
    do {
        output e.dest; // formally: output e.twin.org
        e = e.org; // formally: e = e.twin.next
    } while (e != start);
```
In class exercise

Given vertex $v$ in a cell complex of a 2-manifold, the link of $v$ is defined to be the edges that bound the faces that are incident to $v$, excluding the edges that are incident to $v$ itself. Present a procedure (in pseudocode) that, given a vertex $v$ of a DCEL, returns a list $L$ consisting of the half edges of $v$’s link ordered counterclockwise about $v$. For example, in the figure below, a possible output would be $\langle e_1, \ldots, e_{11} \rangle$. (Any cyclic permutation would be correct.)