

Computer Graphics

Lecture 13

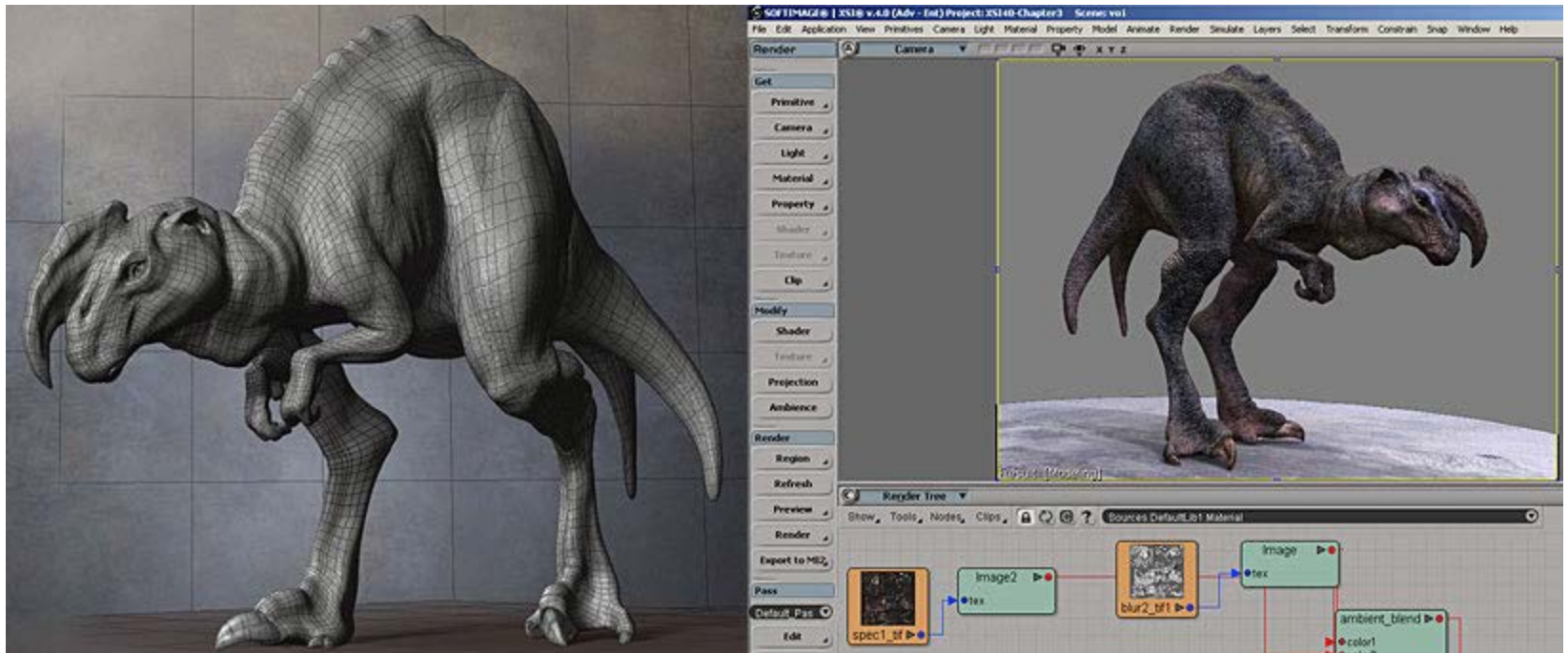
Building 3D Models

- What is 3D model?
 - 3D object using a collection of points in 3D space, connected by various geometric entities such as triangles, lines, curved surfaces, etc.



- Representation of 3D model

- The process of developing a mathematical representation of any three-dimensional surface of object via specialized software



Usage of a 3D model

- Medical
 - The medical industry uses detailed models of organs

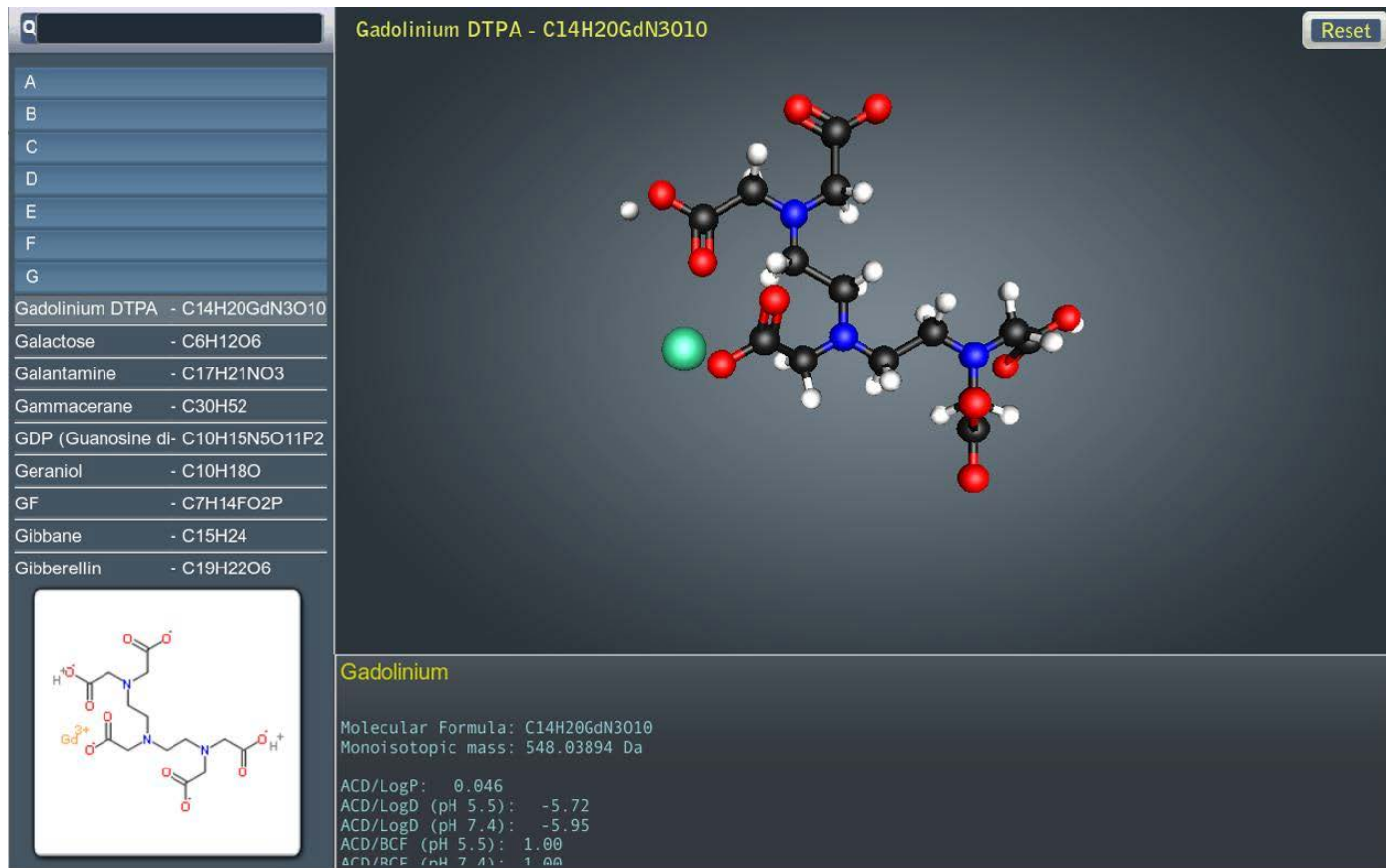


- Movies
 - The movie industry uses them as characters and objects for animated and real-life motion pictures



- Science

- The science sector uses them as highly detailed models of chemical compounds



- Architecture
 - The architecture industry uses them to demonstrate proposed buildings and landscapes through Software Architectural Models



- Engineering
 - The engineering community uses them as designs of new devices, vehicles and structures as well as a host of other uses

Audi R8 e-tron

Thermomanagement
Thermal management
06/15

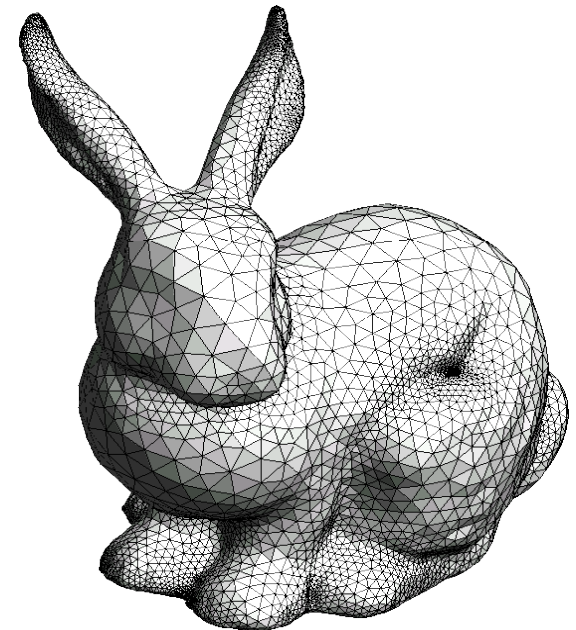


Classic models in CG

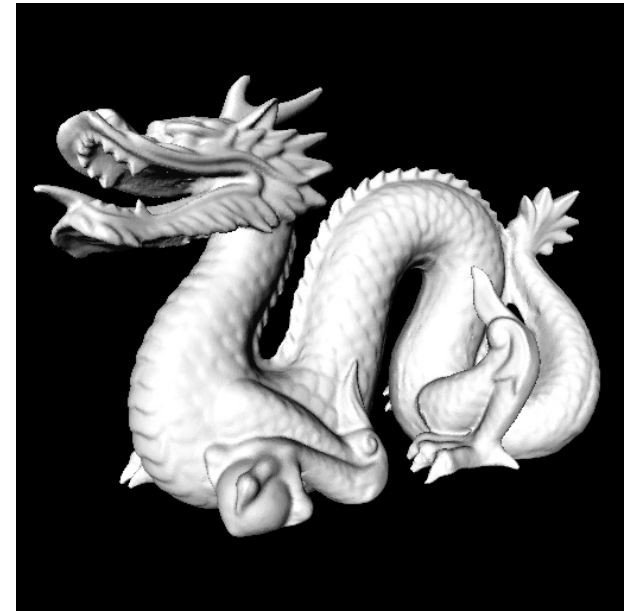
- Most common models use around the Graphics community for their research
 - The Utah Teapot
 - A mathematical model of an ordinary teapot, which appears solid, cylindrical and partially convex.
 - Created in 1975 by Martin Newell, a member of the pioneering graphics program at the University of Utah



- The Stanford Bunny
 - The Stanford Bunny is a computer graphics test model developed by Greg Turk and Marc Levoy in 1994 at Stanford University.
 - The Bunny consists of data describing 69,451 triangles determined by 3D scanning a ceramic figure of a rabbit.



- The Stanford Dragon
 - The Stanford Dragon is a computer graphics test model created with a Cyberware 3030 MS at Stanford University.
 - The Dragon consists of data describing 871,414 triangles determined by 3D scanning a real sculpture

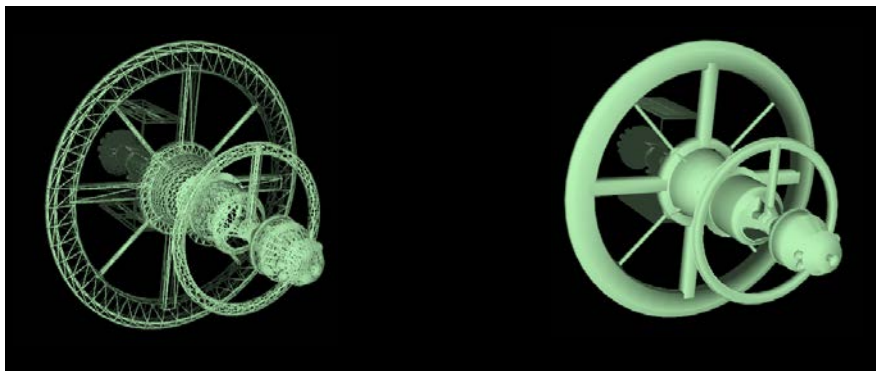


What is a mesh model

- Model represent by set of connected polygons (usually triangles)
- Can Represent by Vertices and Indexed Face Set
- Meshes are flexible and computers can render them so quickly. So the vast majority of 3D models today are built as textured polygonal models
- Since polygons are planar, curved surfaces can represent only by using many polygons

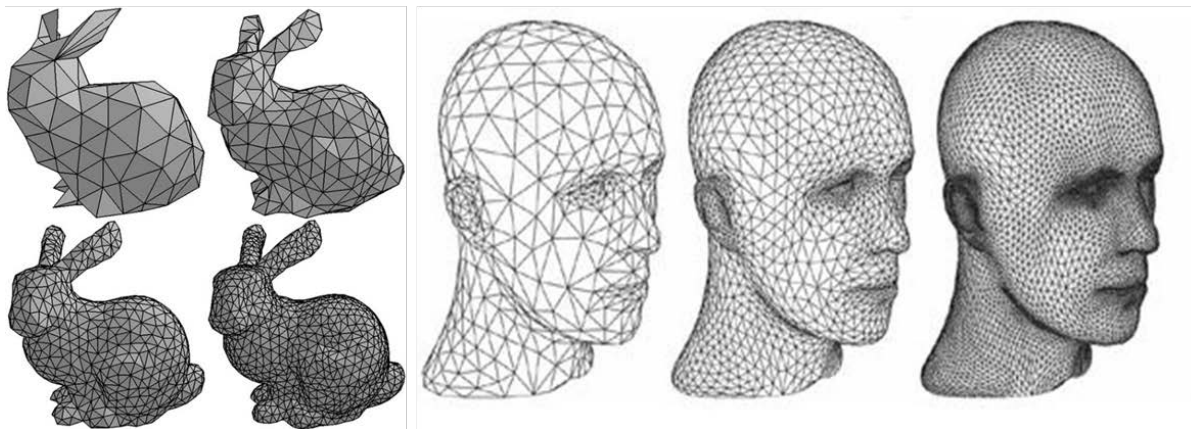
Modeling

3D Model with Triangles



12,000 – 15,000 Wireframe of Triangles

More triangles show fine details



Building a 3D model

Blender v2.78 (sub 0) OBJ File: "

www.blender.org

mtllib cube.mtl

o Cube

v -1.000000 -1.000000 1.000000

v -1.000000 1.000000 1.000000

v -1.000000 -1.000000 -1.000000

v -1.000000 1.000000 -1.000000

v 1.000000 -1.000000 1.000000

v 1.000000 1.000000 1.000000

v 1.000000 -1.000000 -1.000000

v 1.000000 1.000000 -1.000000

vn -1.0000 0.0000 0.0000

vn 0.0000 0.0000 -1.0000

vn 1.0000 0.0000 0.0000

vn 0.0000 0.0000 1.0000

vn 0.0000 -1.0000 0.0000

vn 0.0000 1.0000 0.0000

usemtl None

s off

f 1//1 2//1 4//1 3//1

f 3//2 4//2 8//2 7//2

f 7//3 8//3 6//3 5//3

f 5//4 6//4 2//4 1//4

f 3//5 7//5 5//5 1//5

f 8//6 4//6 2//6 6//6

Wavefront object file of a Cube

Wavefront .obj file

Vertex:

A vertex can be specified in a line starting with the letter "v".

That is followed by (x,y,z[,w]) coordinates. W is optional and defaults to 1.0

```
v -1.000000 -1.000000 1.000000
v -1.000000 1.000000 1.000000
v -1.000000 -1.000000 -1.000000
v -1.000000 1.000000 -1.000000
v 1.000000 -1.000000 1.000000
v 1.000000 1.000000 1.000000
v 1.000000 -1.000000 -1.000000
v 1.000000 1.000000 -1.000000
```

Faces:

Faces are defined using lists of vertex, texture and normal indices.

Polygons such as quadrilaterals can be defined by using more than three vertex/texture/normal indices.

```
usemtl None
s off
f 1//1 2//1 4//1 3//1
f 3//2 4//2 8//2 7//2
f 7//3 8//3 6//3 5//3
f 5//4 6//4 2//4 1//4
f 3//5 7//5 5//5 1//5
f 8//6 4//6 2//6 6//6
```

Vertex Indices:

A valid vertex index matches the corresponding vertex elements of a previously defined vertex list.

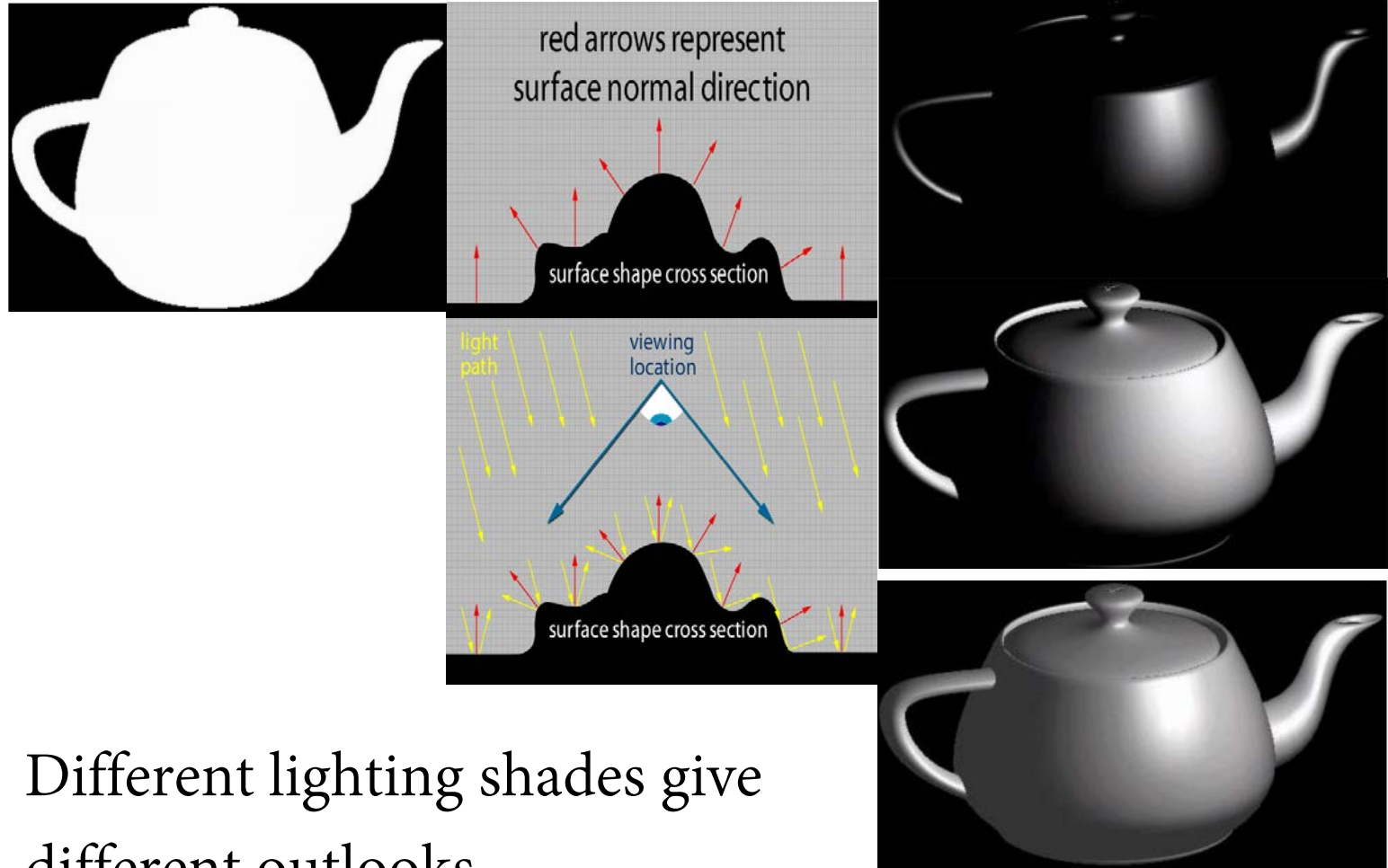
```
v -1.000000 -1.000000 1.000000
v -1.000000 1.000000 1.000000
v -1.000000 -1.000000 -1.000000
v -1.000000 1.000000 -1.000000
v 1.000000 -1.000000 1.000000
v 1.000000 1.000000 1.000000
v 1.000000 -1.000000 -1.000000
v 1.000000 1.000000 -1.000000
```

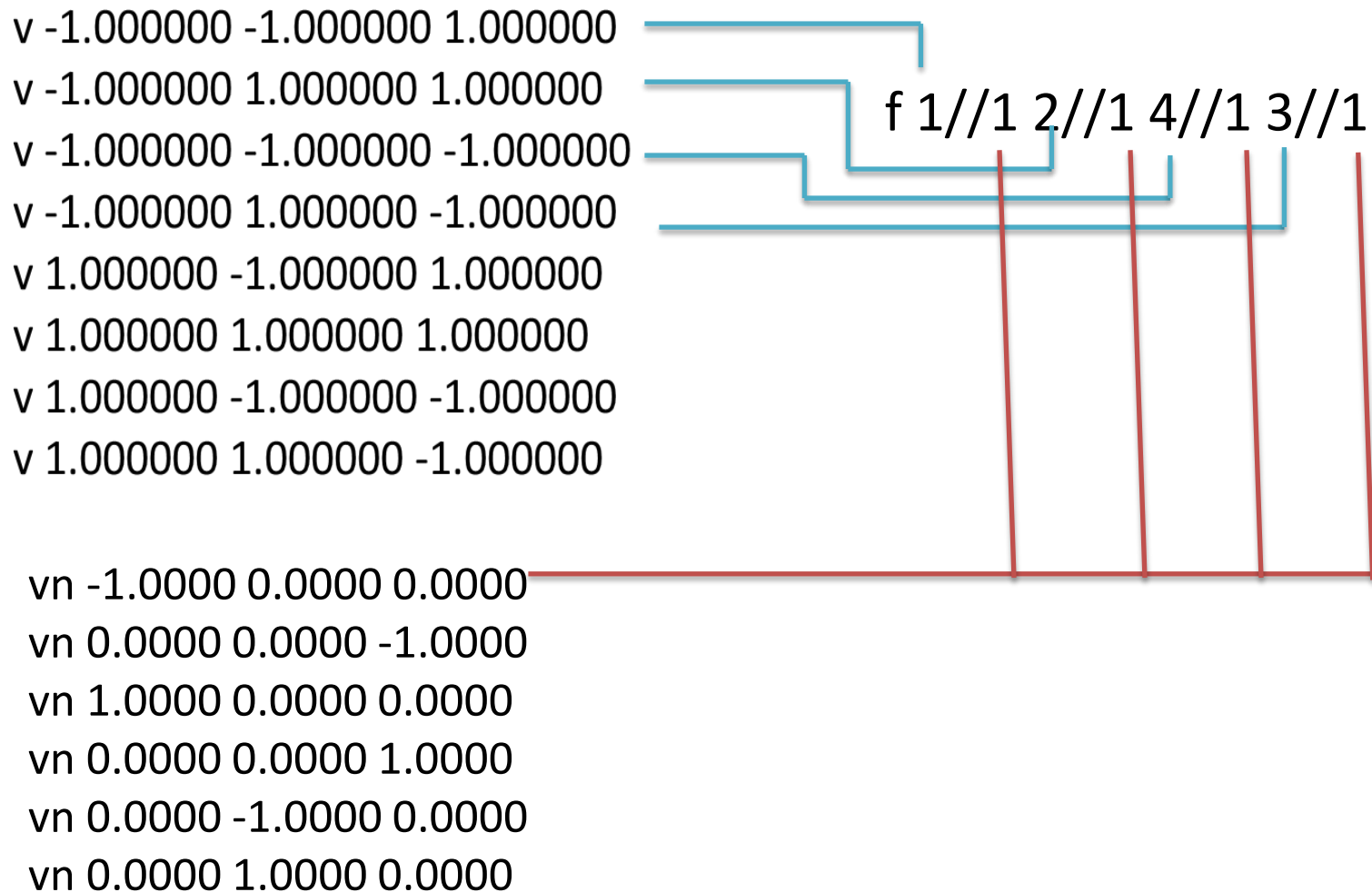
f 1//1 2//1 4//1 3//1

Vertex Normal Indices:

- A vertex can be specified in a line starting with the letter "vn".
- Optionally, normal indices can be used to specify normal vectors for vertices when defining a face.
- To add a normal index to a vertex index when defining a face, one must put a second slash after the texture coordinate index and then put the normal index.
- A valid normal index starts from 1 and matches the corresponding element in the previously defined list of normals. Each face can contain three or more elements.

Model surface normal





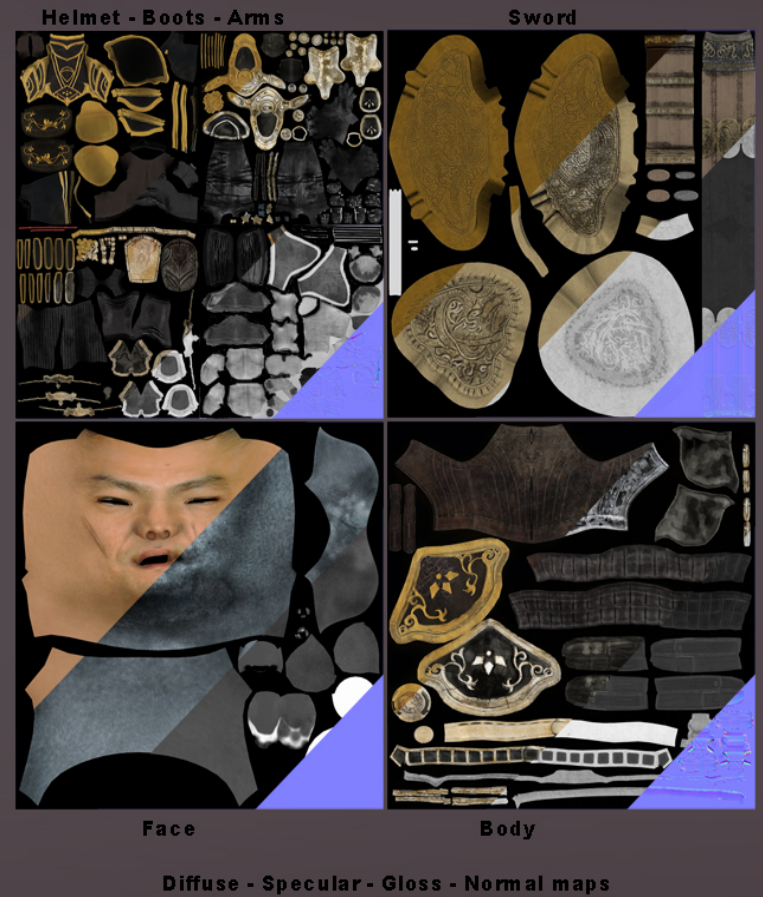
Texture maps



Normal + OCC



Final



Face format

The diagram illustrates the mapping of vertex coordinates to face format indices. It shows four vertex coordinates on the left: vt 0.0 0.0, vt 0.0 1.0, vt 1.0 0.0, and vt 1.0 1.0. Blue lines connect these to the corresponding indices in the face format string 'f 3/1/3 8/4/3 7/3/3'. Specifically, 'vt 0.0 0.0' connects to '3', 'vt 0.0 1.0' connects to '1', 'vt 1.0 0.0' connects to '7', and 'vt 1.0 1.0' connects to '3'.

```
vt 0.0 0.0  
vt 0.0 1.0  
vt 1.0 0.0  
vt 1.0 1.0
```

f 3/1/3 8/4/3 7/3/3

Each face can contain three or more elements with the format of

f v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3

Summary

cube.obj with materials and documentation

#

mtllib cube-textures.mtl

g cube

v1 is origin: back lower left (bll)

v2=fll, v3=bul, v4=ful, v5=blr, v6=flr, v7=

v 0.0 0.0 0.0 v1 = back lower left (origin)

v 0.0 0.0 1.0 v2 = front lower left

v 0.0 1.0 0.0 v3 = back upper left

v 0.0 1.0 1.0 v4 = front upper left

v 1.0 0.0 0.0 v5 = back lower right

v 1.0 0.0 1.0 v6 = front lower right

v 1.0 1.0 0.0 v7 = back upper right

v 1.0 1.0 1.0 v8 = front upper right

vn 0.0 0.0 1.0

vn 0.0 0.0 -1.0

vn 0.0 1.0 0.0

vn 0.0 -1.0 0.0

vn 1.0 0.0 0.0

vn -1.0 0.0 0.0

vt 0.0 0.0 # vt=1 is upper left of texture

vt 0.0 1.0 # vt=2 is lower left of texture

vt 1.0 0.0 # vt=3 is upper right of texture

vt 1.0 1.0 # vt=4 is lower right of texture

remember, syntax is v/vt/vn

g back face

usemtl buffy-gray

f 1/4/2 7/1/2 5/2/2

f 1/4/2 3/3/2 7/1/2

g left face

usemtl buffy-blue

f 1/2/6 4/3/6 3/1/6

f 1/2/6 2/4/6 4/3/6

g top face

usemtl buffy-red

f 3/1/3 8/4/3 7/3/3

f 3/1/3 4/2/3 8/4/3

g right face

usemtl buffy-green

f 5/4/5 7/3/5 8/1/5

f 5/4/5 8/1/5 6/2/5

g bottom face

usemtl buffy-red

f 1/2/4 5/4/4 6/3/4

f 1/2/4 6/3/4 2/1/4

g front face

usemtl buffy-gray

f 2/2/1 6/4/1 8/3/1

f 2/2/1 8/3/1 4/1/1



T-rex low poly game version

Tris : 11,700

Arun aggarwal



T-rex low poly game version

Tris : 11,700

Arun aggarwal

<https://www.artstation.com>



T-rex low poly game version

Tris : 11,700

Arun aggarwal