Computer Graphics

Prof. Jibum Kim

Department of Computer Science & Engineering

Incheon National University

- **Shading (Surface rendering)**

- Lighting 은 material의 각 vertex 단위로 이루어 진다고 했다
- 각 vertex에 대해서 lighting 처리 후 색깔이 정해졌다
- 하지만, 앞 예제에서 보았듯이 polygon 의 내부의 색은 이미 칠해져 있다 (vertex 단위로 lighting이 동작하지만)
- 이런 경우 여러 개의 vertex로 이루어진 Polygon의 경우 polygon 내부 는 어떻게 색깔이 정해 질까?

- 이것을 처리하는 것을 shading 이라고 한다

즉, shading이란 각 정점의 색상이 정해졌을때 다각형 내부의 색을 칠하는 작업을 의미 한다

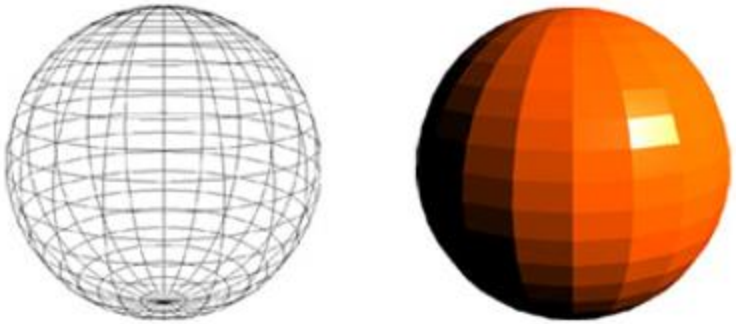
- Shading refers to the process of altering the color of a polygon in the 3D scene, based on its angle to lights and its distance from lights to create a photorealistic effect.

- **Flat shading**

- Flat shading은 매우 빠르고 간단한 방법으로 상수 shading이라고도 한다
- 이 방법에서는 주어진 하나의 다각형 전체를 동일한 색으로 칠한다
- Flat shading은 일방적으로 광원이 조금 먼 거리에 있다고 가정하고 하나의 다각형 표면이 모두 동일한 normal vector를 가지고 있다고 가정한다

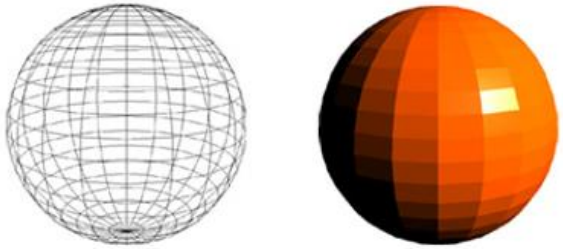
- Flat shading의 순서
- 1. 주어진 다각형에 대하여 다각형 내부의 중심점 (centroid)를 구한다
- (다각형을 구성하는 vertex의 위치를 평균하여 계산)
- 2. 이 중심점에서 normal vector (법선 벡터), 광원 벡터, 시점 벡터를 계산한다.
- 3. 앞에서 배운 Phong 조명 모델을 사용하여 이 vertex에서의 reflection을 계산하여 색을 정한다
- 4. 이 색으로 이 polygon면 내부를 모두 채운다

- OpenGL에서의 flat shading 사용방법
- glShadeModel(GL_FLAT);



- Flat shading은 Mach Band Effect 라고 불리는 현상이 생긴다
- 밑의 그림을 보면 각각의 사각형 내부는 완전히 동일한 색으로 별도의 경계선은 그리지 않았다. 하지만, 경계선이 있는 것 처럼 보인다

이 그림을 보면,
마치 각 영역의 경계선에 무언가 선이 그려져 있는 것처럼 느껴진다.
이 것이 마하 밴드 효과이다~!



Flat shading의 예

- **https://www.dropbox.com/s/2gnmwnqsagnokg0/flat_shading.txt?dl=0**

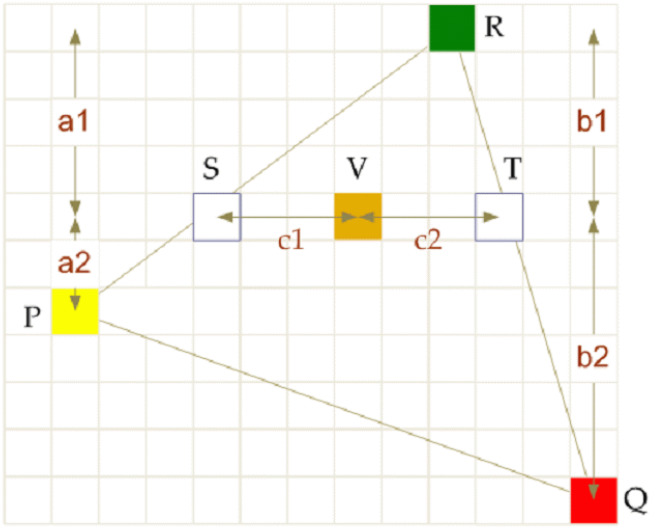
- 두번째 Shading 방법은 Gouraud shading 이다
- 이 shading 방법은 양방향 선형보간 (bilinear interpolation)방법을 사용한다
- Bilinear interpolation 방법은 barycentric coordinates을 사용한다. 배우기 전에 먼저 barycentric coordinates을 계산하는 방법을 다시 공부해보자

- 무게 중심 좌표 계산 방법
- Revisit barycentric coordinates

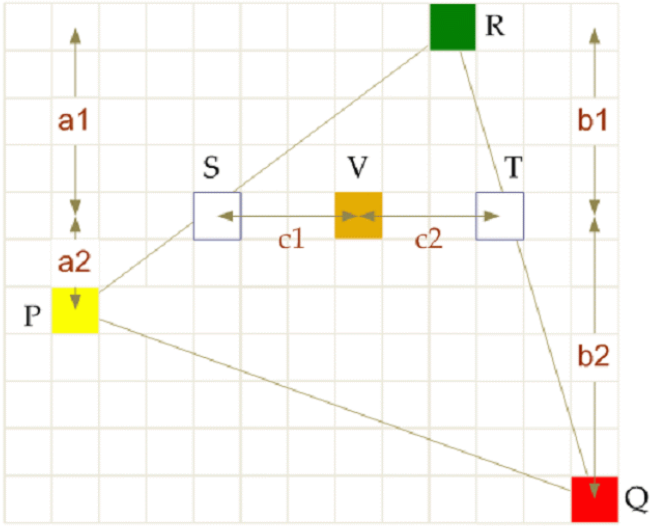
- 선분 PQ 위의 점 V의 무게 중심 좌표
- $V = \alpha P + \beta Q$, 단, $0 \le \alpha, \beta \le 1, \alpha + \beta = 1$
- α 는 P의 영향력, β는 Q의 영향력.
- V가 P에서 멀어질수록 α 는 줄어들어야 함, 이만큼 β는 늘어나야 한다
- 이를 이용, 가중치 비율을 선분의 길이 비율로 표현 가능
- $\alpha : \beta = |VQ| : |VP|$, 즉, $\alpha = \frac{|VQ|}{|VQ|+|VP|} = \frac{|VQ|}{|PQ|}, \beta = \frac{|VP|}{|VQ|+|VP|} = \frac{|VP|}{|PQ|}$

- **Bilinear interpolation (양방향 선형보간)**

- 세 개의 vertex P, Q, R로 이루어지는 polygon에서 P, Q, R의 색이 주어진 경우 polygon 내부의 점 (예: V)에서의 색을 보간하고자 한다
- 이 경우 무게 중심 좌표와 양방향 선형 보간을 이용하여 polygon 내부의 색을 보간할 수 있다



- 양방향 선형 보간
- P, Q, R의 색이 주어짐. 목적: Polygon 내부 V의 색을 보간
- 1. Y 방향 보간
- 1) P와 R을 이용하여 S의 색을 보간
- 2) R과 Q를 이용하여 T의 색을 보간

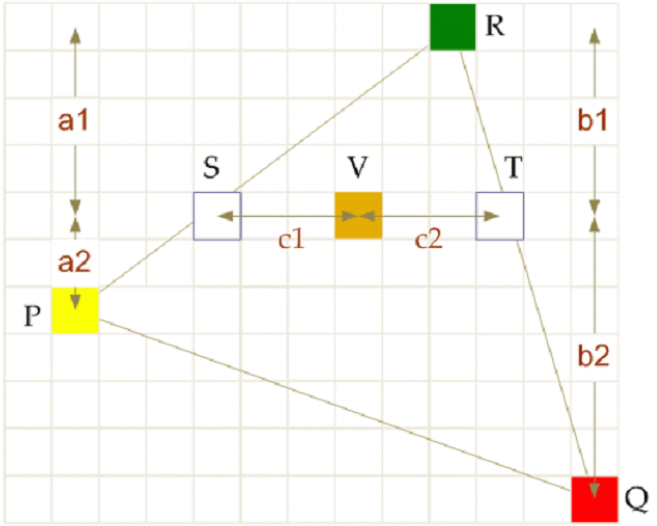


- 2. X 방향 보간
- S와 T를 이용하여 V의 색을 보간

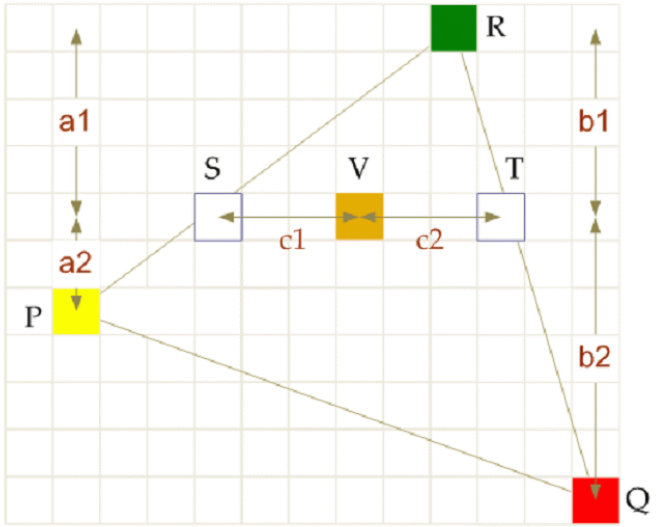
- 1. Y 방향 보간
- 1) P와 R을 이용하여 S의 색을 보간

- 닮은 꼴 삼각형을 이용

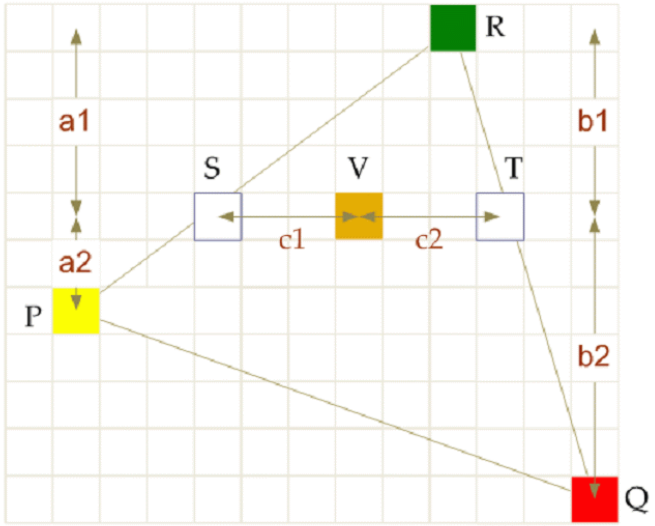
- $S(\text{의색}) = \frac{a1}{a1+a2} P(\text{색}) + \frac{a2}{a1+a2} R(\text{색})$



- 1. Y 방향 보간
- 2) R과 Q를 이용하여 T의 색을 보간
- 닮은 꼴 삼각형을 이용
- $T(\text{의색}) = \frac{b1}{b1+b2} Q(\text{색}) + \frac{b2}{b1+b2} R(\text{색})$



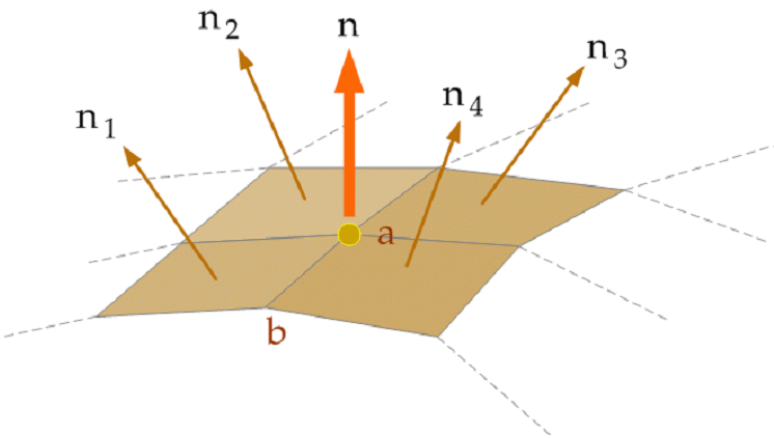
- **2. S와 T의 보간된 색을 이용 V의 색 보간**
- $\boldsymbol{V}(\text{의색}) = \frac{\boldsymbol{c2}}{\boldsymbol{c1}+\boldsymbol{c2}}\boldsymbol{S}(\text{색}) + \frac{\boldsymbol{c1}}{\boldsymbol{c1}+\boldsymbol{c2}}\boldsymbol{T}(\text{색})$



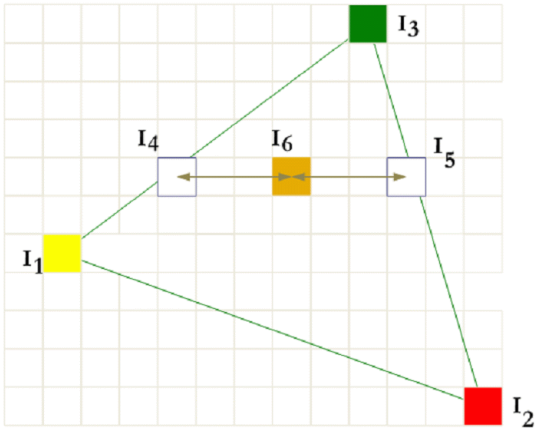
- **Gouraud shading (구로 쉐이딩)**

- **Gouraud shading**
- **Flat shading과 다르게 polygon 내부를 서로 다른 색으로 채우는 방법**
- **Smooth shading이라고도 불린다**
- **Polygon의 vertex에서의 색을 계산한 후 양방향 선형 보간 을 통하여 polygon 내부의 색을 칠함**
- **Polygon을 이루는 vertex에서의 색을 계산시 각 vertex에서의 법선벡터가 필요하다. 하지만, 법선 벡터란 vertex 단위가 아닌 면 단위로 결정된다. 또한, vertex는 일반적으로 면이 교차하는 위치에 존재 한다**
- **어떤 정점에서의 법선벡터를 어떻게 계산할 까?**

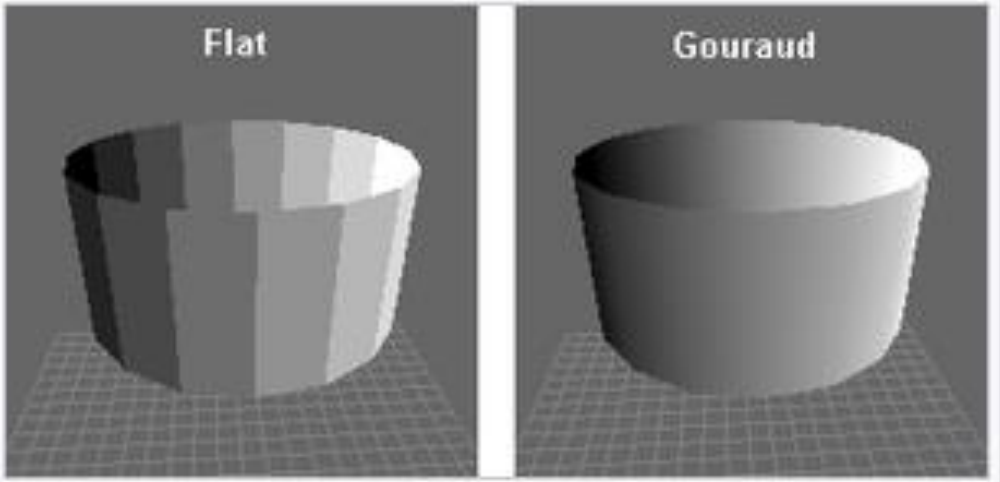
- Vertex에서의 법선 벡터 구하기 (근사화 하기)
- 그 vertex에 인접한 면들의 법선 벡터를 구하고 이를 평균 낸다
- 즉, 해당 vertex를 공유하는 면의 개수에 따라 식은 달라진다
- (즉, 이 vertex에 인접한 면에 대한 정보가 필요하다)
- 예: vertex a에서의 법선 벡터 (vertex a에 인접한 면 4개) $n = \frac{n_1 + n_2 + n_3 + n_4}{|n_1 + n_2 + n_3 + n_4|}$
- 예: vertex b에서의 법선 벡터 (vertex b에 인접한 면 2개) $\boldsymbol{n = \frac{n_1 + n_4}{|n_1 + n_4|}}$



- 각 vertex의 법선 벡터를 구하고 그 vertex에서의 색이 계산되면 polygon 내부의 색을 양방향 선형 보간으로 계산한다
- 이렇게 양방향 선형 보간을 사용하면 polygon 내부 화소들의 색이 서서히 변하게 된다



- **Gouraud shading에서 다각형 내부의 음영이 보다 부드럽게 이어진다. 하지만, 단점으로는 선형 보간 계산 때문에 flat shading 보다 오랜 시간이 필요하다**



Comparison of flat shading and Gouraud shading.

- OpenGL에서의 shading
- Gouraud shading 사용시
- glShadeModel(GL_SMOOTH);
- Flat shading 사용시
- glShadeModel(GL_flat);

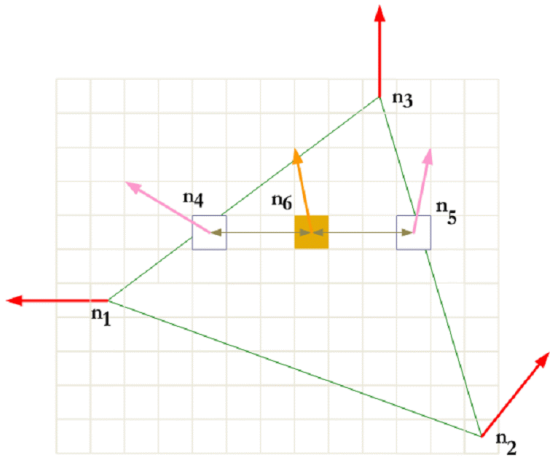
- **https://www.dropbox.com/s/87pkyzn20e0xsnd/shading_1.txt?dl=0**

- **glutSolidTeapot (1.0);**

- **Phong shading (퐁 쉐이딩)**

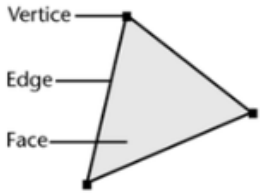
- **Phong shading (퐁 쉐이딩)**
- **Phong에 의해 제안된 쉐이딩 방법**
- **퐁 쉐이딩은 vertex의 법선 벡터를 보간한다**
- **먼저, vertex의 법선 벡터를 보간하여 개별 화소마다의 법선 벡터를 계산한 후, 결과 법선 벡터를 기준으로 화소마다 조명 모델을 가하여 반사광의 세기를 구한다**

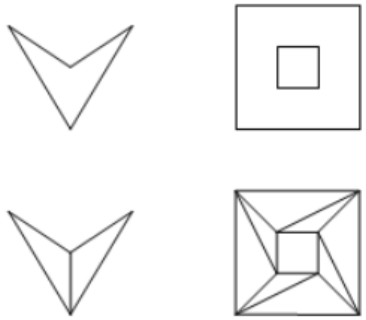
- 법선 벡터의 보간은 구로 쉐이딩의 색 보간과 동일하게 **양방향 선형 보간**을 사용
- 벡터 보간시에는 벡터의 x, y, z 성분 별로 보간이 행해진다
- 벡터 (2, 4, 6)을 지닌 화소와 벡터 (4, 4, 8)을 지닌 화소의 중간점에 있는 화소에서의 벡터는 (3, 4, 7)이 된다
- 예: n6에서의 법선 벡터를 보간하기 위하여
- 1. n1, n3 법선 벡터 이용: n4 법선 벡터 보간
- 2. n3, n2 법선 벡터 이용: n5 법선 벡터 보간
- 3. n4, n5 법선 벡터 이용: n6 법선 벡터 보간



- **Modeling in 3D Space**

- **Polygon: polygons are straight-sided shapes (3 or more sides), defined by vertices and the straight lines that connect them (edges)**
- **Face: polygon의 내부 영역**
- **Polygon의 기본 요소: vertices, edges, face**
- **Polygon은 가장 단순하면서 더 많이 사용되는 2D 물체임**
- **우리가 사용하는 polygon은 simple and planar polygon임**
- **Convex polygon은 OpenGL에서 GL_POLYGON으로 한번에 그릴 수 있다. Non-convex polygon인 경우에는 convex한 polygon으로 나누어서 그린다 (tessellation 이용)**



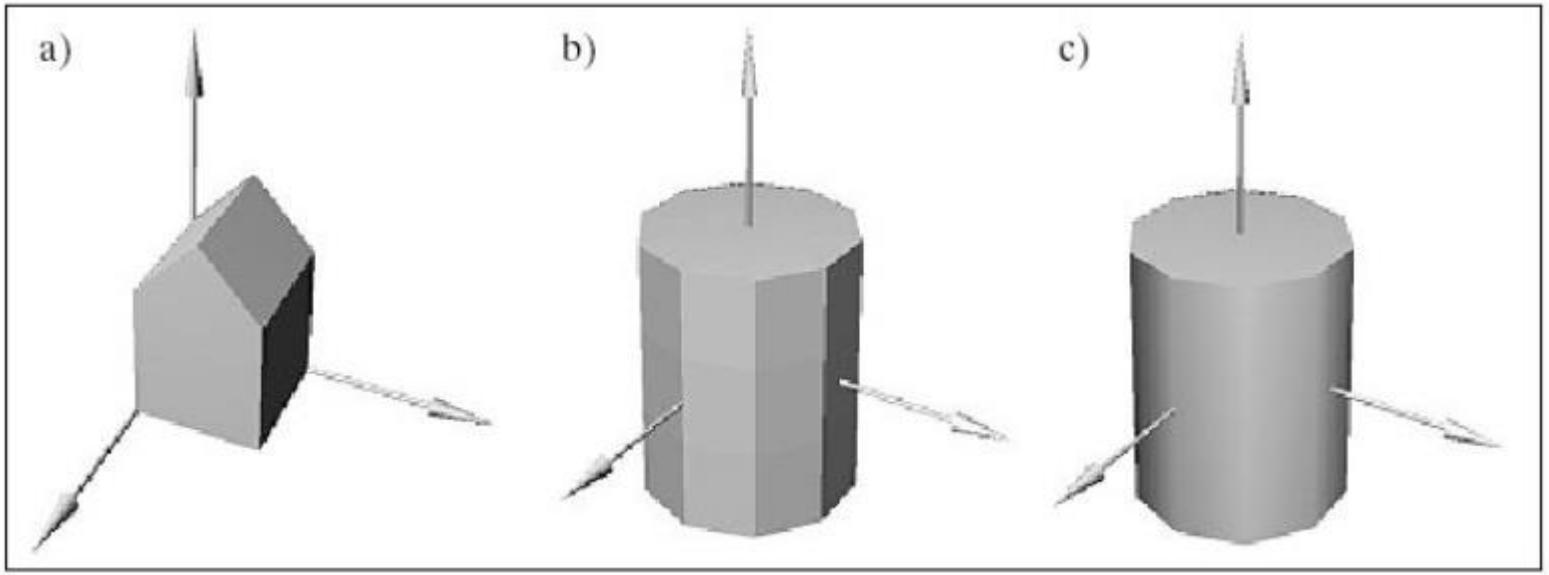


Tessellation

- **Polygonal meshes (simply, meshes, 다각형 메쉬) are simply collection of polygons, or faces that fit together to form the skin of the object**
- **현재 컴퓨터 그래픽스에서 solid shape을 표현하는 표준적인 방법이다**
- **물체가 solid하다는 것의 의미는?**
- **The object is considered to be solid of the polygonal faces fit together to enclose space (closed surface)**

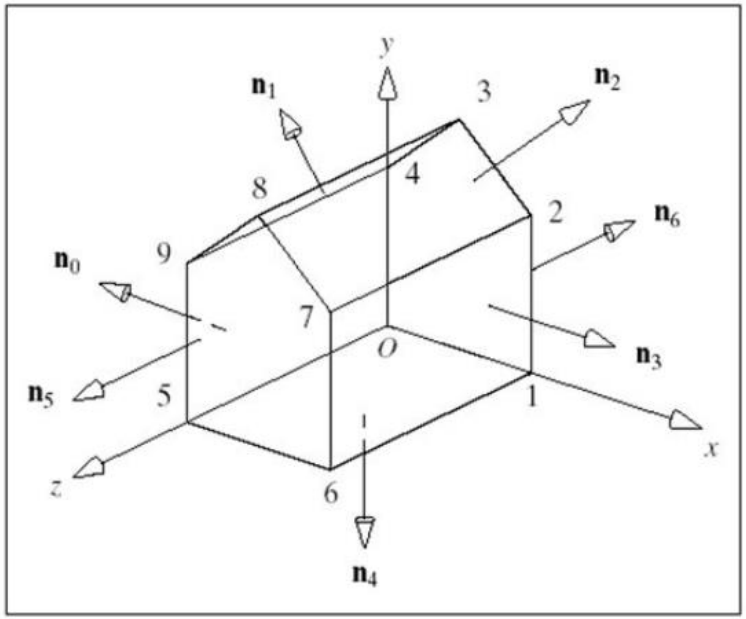
- A polygonal mesh is given by a list of polygons, along with information about the direction in which each polygon is facing
- The important directional information is often simply the outward-pointing normal vector to the plane of the face

- 일부 object는 polygonal mesh로 완벽히 표현 가능
- 예: flat한 face를 갖는 물체 (a)
- 근사적으로 밖에 표현할 수 없는 objects들도 있음(b, c)



- **Mesh example**

- **The basic barn (외양간?): it has seven polygonal faces and a total of 10 vertices**
- **It has seven normal vectors**



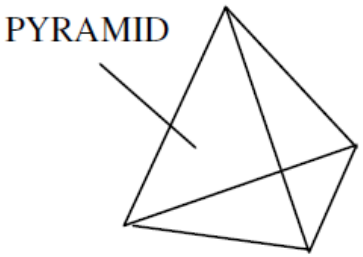
- 3개의 separate list로 mesh 표현
- **Vertex list:** vertex들의 위치 정보
- **Normal list:** 각 face의 orientation 정보 (normalize된 normal vector)
- **Face list :** face에 속하는 vertex들과 각 vertex에 해당하는 normal vector
- 여기서 각 face는 flat 하므로 동일함

vertex	x	y	z
0	0	0	0
1	1	0	0
2	1	1	0
3	0.5	1.5	0
4	0	1	0
5	0	0	1
6	1	0	1
7	1	1	1
8	0.5	1.5	1
9	0	1	1

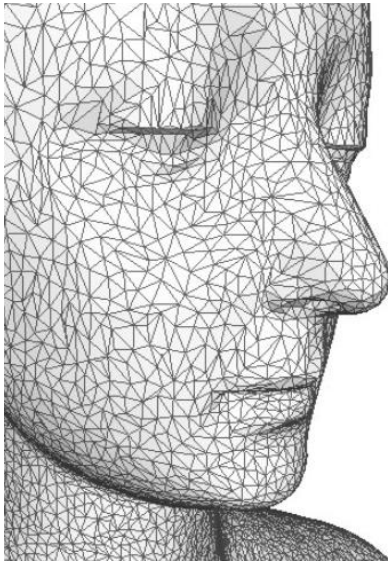
normal	n_x	n_y	n_z
0	−1	0	0
1	−0.707	0.707	0
2	0.707	0.707	0
3	1	0	0
4	0	−1	0
5	0	0	1
6	0	0	−1

face	vertices	associated normal
0 (left)	0,5,9,4	0,0,0,0
1 (roof left)	3,4,9,8	1,1,1,1
2 (roof right)	2,3,8,7	2,2,2,2
3 (right)	1,2,7,6	3,3,3,3
4 (bottom)	0,1,6,5	4,4,4,4
5 (front)	5,6,7,8,9	5,5,5,5,5
6 (back)	0,4,3,2,1	6,6,6,6,6

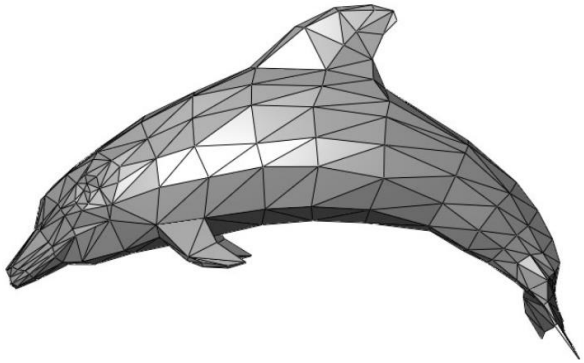
- **Mesh의 특징들**
- **1. Solidity: 각 face가 유한한 양의 공간을 enclose하는 solid object를 표현**
- **2. Connected: 모든 face가 다른 face와 적어도 하나의 edge를 공유**
- **3. Convexity: mesh가 convex한 object를 표현**
- **4. Planarity: 각 face가 planar polygon (각 face의 vertices가 한 평면에 있음)**
- **5. Simplicity: solid object를 표현하면서 hole (구멍)이 없는 것**



- **컴퓨터 그래픽스에서 사용하는 가장 일반적인 mesh는 triangular mesh로 모든 face가 삼각형들인 mesh를 의미한다**



Triangular mesh로 모델링한 사람 얼굴



Triangular mesh로 모델링한 돌고래

- 3D mesh File format
- 확장자 .3ds, .ply, .stl, .obj, .vtk,….
- 예: obj file format
- 3D mesh를 표현하는
- https://en.wikipedia.org/wiki/Wavefront_.obj_file
- http://www.andrewnoske.com/wiki/OBJ_file_format
- https://dirsig.cis.rit.edu/docs/new/obj.html

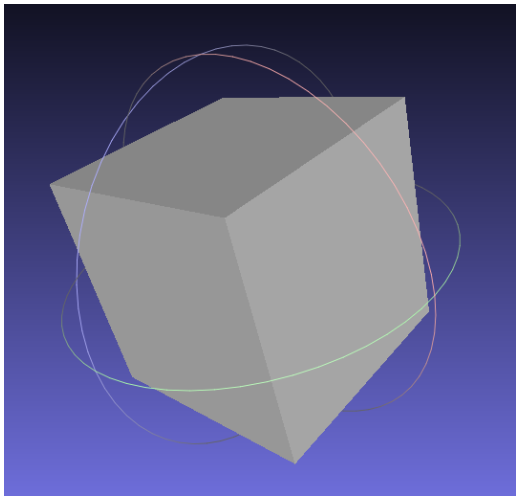
- obj mesh 파일 포맷의 예
- # comment line (무시됨)
- # vertex 위치 정보 (x, y, z)
- v x y z
- …
- # vertex의 texture 정보 (u, v). 0에서 1사이 값으로. (s, t)와 유사
- vt u v
- …
- # vertex의 법선 벡터 정보
- vn x y z
- …
- # face의 정보 (vertex 들의 index 정보 줌)
- f v1 v2 v3
- …

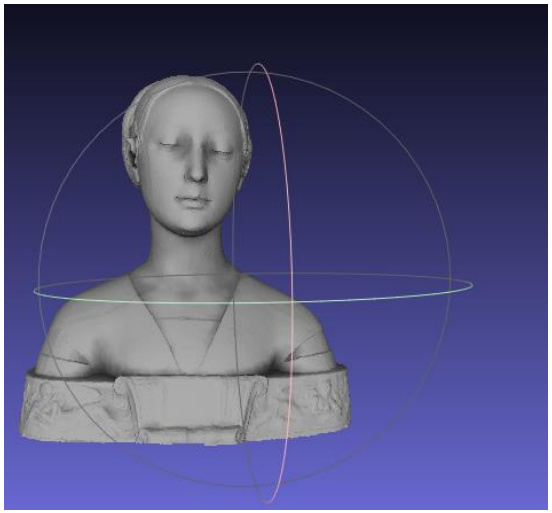
- https://www.dropbox.com/s/40yq3jzu71f8uy8/cube.obj?dl=0
- https://www.dropbox.com/s/acbya4b9t6nf22z/Laurana50k.ply?dl=0
- **https://www.meshlab.net/#download**
- **Stanford graphics lab**
- **http://graphics.stanford.edu/data/3Dscanrep/**

- 예: cube.obj : 각 face의 normal vector 정의, 단 vertex number는 1부터 시작

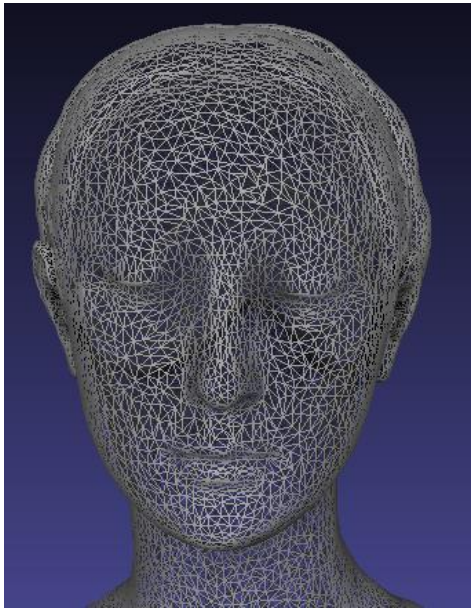
```
# Object cube.obj
# Vertices: 8
# Faces: 6
####
v  0.0  0.0  0.0
v  0.0  0.0  1.0
v  0.0  1.0  0.0
v  0.0  1.0  1.0
v  1.0  0.0  0.0
v  1.0  0.0  1.0
v  1.0  1.0  0.0
v  1.0  1.0  1.0

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

f 1 2 4 3
f 2 6 8 4
f 6 5 7 8
f 5 1 3 7
f 3 4 8 7
f 5 6 2 1
# 6 faces, 0 coords texture

# End of File
```

- Open source인 MeshLab을 이용하면 3D mesh를 쉽게 시각화할 수 있다
- 컴퓨터의 MeshLab 실행
- File누른 후 Import Mesh 누르고 메쉬 선택후 실행
- Meslab은 대부분의 3D mesh format 지원
- (좌) cube.obj (우) Laurana50k.ply





- **Rendering 모드 선택으로 Wireframe 모두로 triangular mesh를 볼 수 있음**



- 기본적으로 ‘obj’ 파일의 메쉬를 이용하면 3D 프린터로 출력 가능하다
- 3D 프린터는 물체의 밑단 부터 쌓아가는 방식이기 때문에 아래 왼쪽 메쉬와 같이 공중에 떠있는 부분이 있는 메쉬는 별도의 지지대를 3D 프린터 프로그램을 설정하여 받혀야 한다
- 7호관 422호의 3D 프린터 (Cubicon style)로 3D 프린터로 obj파일을 출력 가능
- http://www.3dcubicon.com/cubicon/sub22_3.php?goods_id=1804050001