Ch11: Modelling

Solid object

Solid object: finite, 3D, rigid, closed, finitely describable, determinable boundary

Geometric modeling: the generation of abstract description of 3D objects

- **Point-based**
- **Surface-based (b-reps, polygon mesh)**
- **Constructive**

Spatial-Partitioning Representation

- **Cell decomposition (voxels)**
- **Spatial occupancy enumeration**
- **Octrees**
- **Binary Space Partition**

Constructive Solid Geometry (CSG)

Directed acyclic graph (DAG): share the same component

Regularized Boolean Set Operations

After boolean set operations between solid objects, remove the dangling faces (regularized)

Polygon meshes

Polygon mesh: geometric objects with flat faces and straight edges

- Faces: the boundary of solid objects/spaces
- Edges: the boundary of faces
- Vertices: the boundary of edges
- Normals, texture coordinates, colors, shading coefficients, etc.

Polygon: simple, convex, flat

Representing polygon meshes

- Vertex list: locations of the vertices, geometric information
- Edge list: indexes into end vertices of edges, topological information
- Face list: indexes into vertices and normal list, topological information
- Normal list: directions of the normal vectors, orientation info

Euler's Formula: V – E + F = 2

- V: # of vertices
- E: # of edges
- F: # of faces

This implies: vertex list and face list are enough (any two of three is enough)

3D File Formats

3D file format: store information about 3D models as plain text or binary data
Popular formats: STL, OBJ, FBX, COLLADA, VRML, X3D

STL: neutral 3D formats, triangular mesh
OBJ: natural 3D formats, smooth curves and NURBS
FBX: proprietary file format
COLLADA: neutral 3D formats
VRML: web, polygonal mesh
X3D: based on VRML, add NURBS

Question:

a) What is Euler's formula for a convex polyhedron? Explain your variables.

[4 marks]

(a) Euler's formula: V + E – F = 2. V represents the number of vertices, E represents the number of edges, F represents the number of faces.

Question:

a) This question is about modelling.

[5 marks]

i) Name three classes of methods for geometric modelling.

(3 marks)

ii) Name four file formats for 3D models.

(2 marks)

(1) point-based, surface-based, constructive methods
(2) OBJ, X3D, VRML, COLLADA, STL, FBX

Ch12: Geometric Transformations

Scaling
Uniform scaling | Non–uniform scaling
Scaling in matrix form:
$\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$
— Not preserve lengths
— Not preserve angles
— Not a rigid body transformations

Rotation （物体逆时针转动）
$\begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$
— preserve lengths
— Preserve angles
— Rigid body transformations

Translation
— preserves lengths
— Preserves angles
— Rigid body transformation

2D shearing
$\begin{bmatrix} 1 & \frac{1}{tan(\theta)} \\ 0 & 1 \end{bmatrix}$ shear along x: $\begin{bmatrix} 1 & sh_x \\ 0 & 1 \end{bmatrix}$ shear along y: $\begin{bmatrix} 1 & 0 \\ sh_y & 1 \end{bmatrix}$

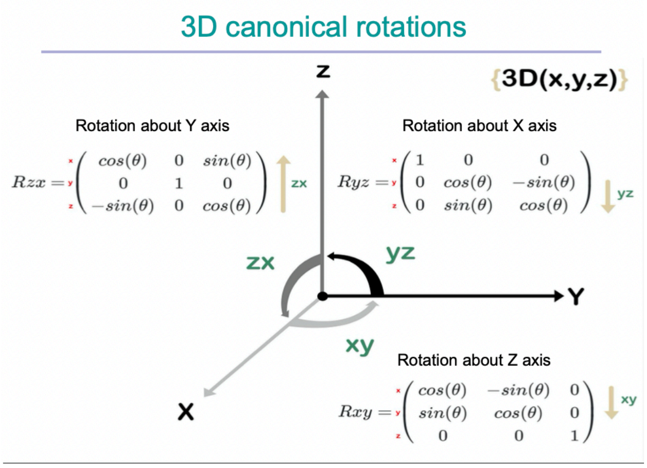
Only linear 2D transformations can be represented with a 2x2 matrix

Properties of linear transformation (properties of **affine transformations**)
— **origin** maps to origin
— **Lines** map to lines
— **Parallel lines** remain parallel
— **Ratios** are preserved
— Closed under **composition**

Homogeneous coordinates
Translation

$$Translation = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

3D rotation



Reverse rotations

$R^{-1}(\theta) = R(-\theta) = R^T(\theta)$

Rigid body transformation: rotation | translation

Non-rigid body transformation: scaling | shearing

3D Translation

– preserves lengths, angles, areas, volumes

3D Scaling

– not preserve lengths, angles, areas, volumes

3D Rotation

– preserves lengths, angles, areas, volumes

3D Shearing

– not preserves lengths, angles, areas, but preserve volumes

Question:

b) This question is about geometric transformations.

[11 marks]

i) Consider Model A and Model B in **Figure 1**. Give a chain of the basic transformation matrices for translation, scaling and rotation which, when post-multiplied by the homogeneous coordinates of the vertices of Model A, will transform the vertices of Model A into their corresponding vertices of Model B, such that the vertex at $(-1,1)$ of Model A is transformed to the vertex at $(c,0)$ of Model B.

(7 marks)

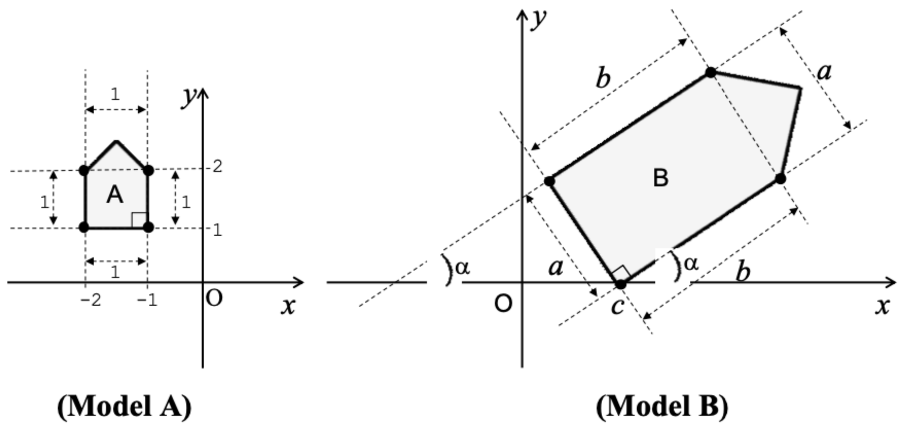


Figure 1.

ii) Compute the composite 2D transformation matrix for the transformations found in question i) above.

(4 marks)

(1) First, translate A to the origin (the bottom right point on the origin).

$$T(1,-1) = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

Second, non-uniform scaling A with factor a on x-direction and b on y-direction

$$S(a,b) = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Third, rotate A with $\theta = -(\frac{\pi}{2} - \alpha) = \alpha - \frac{\pi}{2}$

$$R(\alpha - \tfrac{\pi}{2}) = \begin{bmatrix} \cos(\alpha - \frac{\pi}{2}) & -\sin(\alpha - \frac{\pi}{2}) & 0 \\ \sin(\alpha - \frac{\pi}{2}) & \cos(\alpha - \frac{\pi}{2}) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \sin\alpha & \cos\alpha & 0 \\ -\cos\alpha & \sin\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Fourth, translate A to point c (the bottom right point on the c point)

$$T(c,0) = \begin{bmatrix} 1 & 0 & c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(2) $M = T(c,0) \cdot R(\alpha - \frac{\pi}{2}) \cdot S(a,b) \cdot T(1,-1)$

$$= \begin{bmatrix} 1 & 0 & c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sin\alpha & \cos\alpha & 0 \\ -\cos\alpha & \sin\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sin\alpha & \cos\alpha & 0 \\ -\cos\alpha & \sin\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a & 0 & a \\ 0 & b & -b \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a\sin\alpha & b\cos\alpha & a\sin\alpha - b\cos\alpha \\ -a\cos\alpha & b\sin\alpha & -a\cos\alpha - b\sin\alpha \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a\sin\alpha & b\cos\alpha & a\sin\alpha - b\cos\alpha + c \\ -a\cos\alpha & b\sin\alpha & -a\cos\alpha - b\sin\alpha \\ 0 & 0 & 1 \end{bmatrix}$$

Ch13: Color & Lighting

Cones:
L or R — red light
M or G — green light
S or B — blue light

Metamers: a given perceptual sensation of color derives from the stimulus of all three cone types.

Perception:
– varies form people to people
– Affected by adaptation
– Affected by surrounding colors

Color Perception
– Hue: distinguishes between colors
– Saturation: how far the color is from a gray of equal intensity
– Lightness: the perceived intensity of a reflecting object

HSV color model

Color Models
• Additive (RGB)

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

• Subtractive (CMYK) — K = min (C, M, Y)

Question:

d) What are the four printing primary colours used in printing? Give the relations between the four printing primary colours and the three additive primary colours.

[4 marks]

(d) C – cyan, M – magenta, Y – yellow, K – black.
The relations are:
[C, M, Y] = 1 – [R, G, B]
K = min[C, M ,Y]; [C, M, Y] = [C, M, Y] – K

Lighting

- Illumination: the transport of energy from light sources to surfaces
- Lighting: compute the luminous intensity
- Shading: assign colors to pixels

Illumination (physically based, empirical)

Two components of lighting

- **Light sources**: spectrum, geometric attributes, attenuation
- **Surface properties**: reflectance spectrum, subsurface reflectance, geometric attributes

Light models: Emission + Scattering + Reception

Light source modeling

Ambient light sources:

1. No spatial or directional characteristics
2. **Illuminates all surfaces equally**
3. Amount reflected depends on **surface properties**

$I = K_a \cdot I_a, \quad I_a = L(P_0)$ = L (constant)

Distant light sources:

1. **Direction is constant** for all surfaces
2. All rays of light from the source are **parallel**

$\vec{L}(P, P_0) = L \cdot \vec{l}$

Point light sources:

1. emits **light equally in all directions**
2. **direction** to the light from a point **differs** for different points
3. The intensity of illumination is **proportional** to the **inverse square of distance**

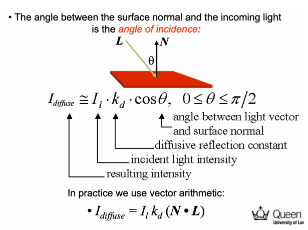
$$\vec{L}(P, P_0) = \frac{L(P_0) \cdot \vec{l}}{|P - P_0|^2}, \ \vec{l} = \frac{P - P_0}{|P - P_0|}$$

Area light sources:

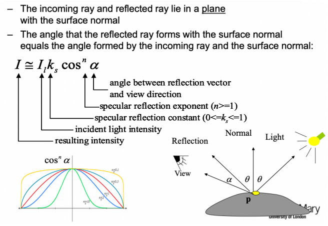
1. 2D emissive surface
2. Capable of generating **soft shadows**

Reflection (diffuse + specular)

Diffuse reflection: very rough surface, incoming ray is equally likely to be reflected at any direction

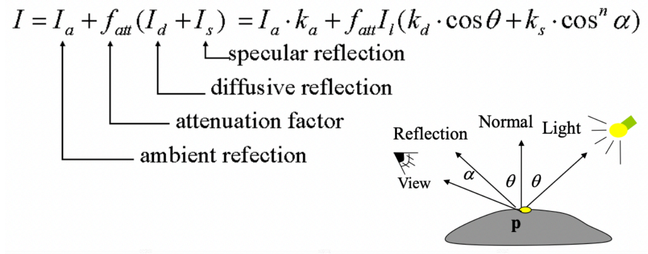


Specular reflection: very smooth, a light shining on a specular surface causes a **bright spot**

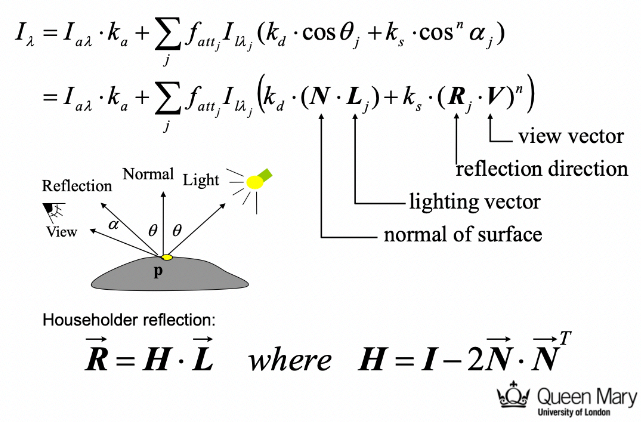


view-dependent

The Light Model: (ambient + reflection – (diffuse + specular) + emission + attenuation)



Multiple color light sources

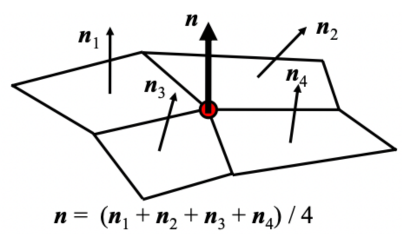


Shading Models
– Flat shading
– Smooth shading
– **Gourand** shading
– **Phong** shading

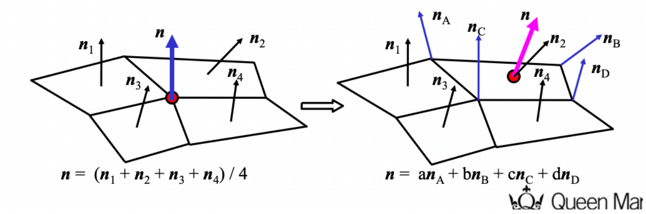
Flat shading: good for **coarse** preview of scenes
– problem: mach band effect: dark looks darker, light looks lighter
Solution: smooth shading — remove discontinuity

Gouraud shading: intensity interpolation shading

The normal at each vertex is the average of the normals of its adjacent faces



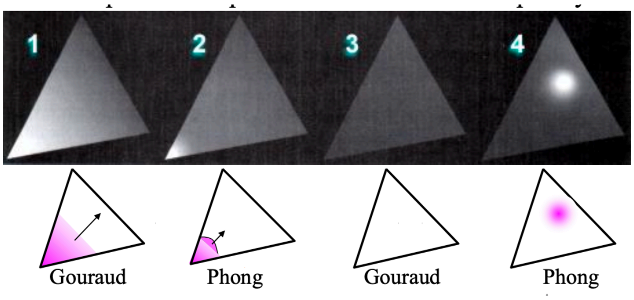
Phong shading: normal-vector interpolation shading



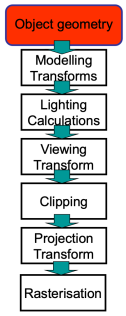
Gouraud shading problems

– may miss interior specular highlights if it's not at vertices

– may spread specular highlights along edges



Rendering Pipeline



- Object geometry (move models)
- Modelling Transforms
- Lighting calculations
- Viewing transform (move camera)
- Clipping
- Projection Transform
- Rasterization

Transformations (3)

- Modeling Transforms: from **object** coordinates to **world** coordinates
- Viewing Transforms: from **world** coordinates to **view reference** coordinates
- Projection Transforms: from **camera** coordinates to **window** coordinates