Matematikken bag 3D computergrafik

Rolf Fagerberg
Lektor i datalogi
Institut for Matematik og Datalogi, SDU

UNF 11. november 2015

3D grafik

3D computergrafik bruges bla. til:

- Computerspil
- Animerede film
- Speciel effects i almindelige film
- Træningssimulatorer (pilot, kaptajn, soldat,. . .)
- Visualisering (data, arkitektur, . . .)

3D grafik

3D computergrafik bruges bla. til:

- Computerspil
- Animerede film
- Speciel effects i almindelige film
- Træningssimulatorer (pilot, kaptajn, soldat,. . .)
- Visualisering (data, arkitektur, . . .)

Det er de samme principper for 3D grafik som bruges i alle disse sammenhænge. Disse er aftenens emne.

Virtuelle objekter

Vi vil gerne definere objekter i det 3-dimensionelle rum ($\vec{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3$).

Virtuelle objekter

Vi vil gerne definere objekter i det 3-dimensionelle rum ($\vec{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3$).

I den virkelige verden ser vi normalt overfladen af objekter $\Rightarrow$ vi vil gerne definere overflader i rummet.

Virtuelle objekter

Vi vil gerne definere objekter i det 3-dimensionelle rum ($\vec{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3$).

I den virkelige verden ser vi normalt overfladen af objekter $\Rightarrow$ vi vil gerne definere overflader i rummet.

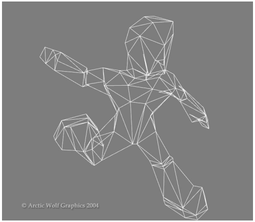
Tre punkter i rummet definerer altid en plan i rummet, samt indhegner et trekantet udsnit af denne.

Virtuelle objekter

Vi vil gerne definere objekter i det 3-dimensionelle rum ($\vec{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3$).

I den virkelige verden ser vi normalt overfladen af objekter $\Rightarrow$ vi vil gerne definere overflader i rummet.

Tre punkter i rummet definerer altid en plan i rummet, samt indhegner et trekantet udsnit af denne. Så trekanter vil være den basale byggesten, som sættes sammen til overflader af de ønskede virtuelle objekter.

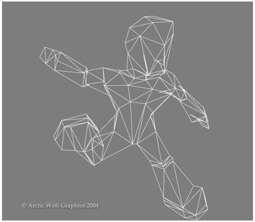


Virtuelle objekter

Vi vil gerne definere objekter i det 3-dimensionelle rum ($\vec{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3$).

I den virkelige verden ser vi normalt overfladen af objekter $\Rightarrow$ vi vil gerne definere overflader i rummet.

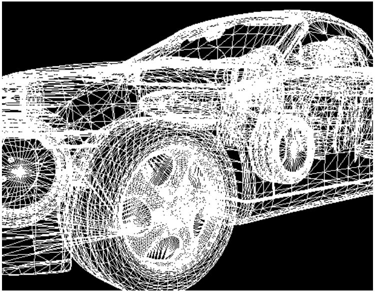
Tre punkter i rummet definerer altid en plan i rummet, samt indhegner et trekantet udsnit af denne. Så trekanter vil være den basale byggesten, som sættes sammen til overflader af de ønskede virtuelle objekter.

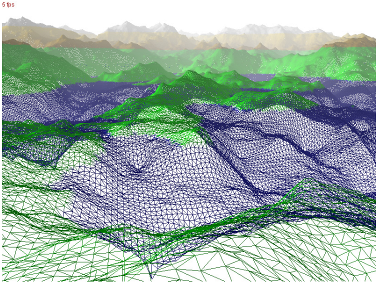


Model = samling af punkter i rummet samt info om hvordan de hænger sammen tre og tre (modellens trekanter).

Virtuelle objekter

Flere eksempler:





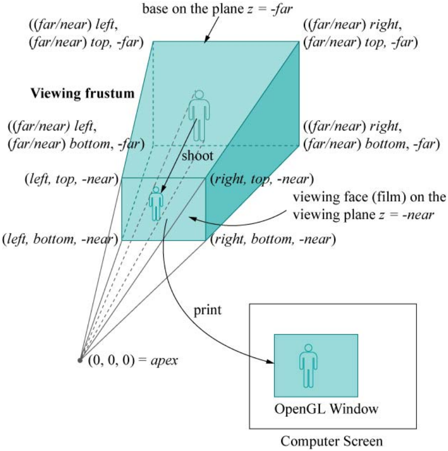
Rendering af virtuelle objekter i 3D

Rendering = generering af et billede ("fotografering").

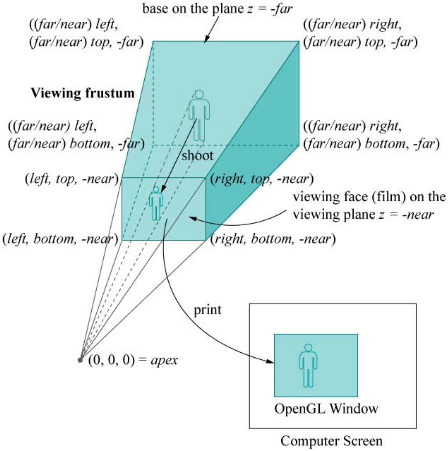
Skridtene i rendering:

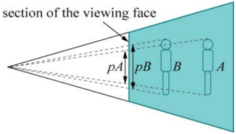
- Definer modeller som samlinger af trekanter.
- Flyt modellerne rundt i rummet (*transformation*), dvs. opstil scenen.
- Flyt alle trekanter i scenen til skærmen (*projektion*) på en måde som simulerer fotografering.
- Tilføj farver til de pixels på skærmen som dækkes af de projicerede trekanter (*shading*).

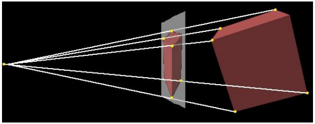
Perspektivisk projektion



Perspektivisk projektion







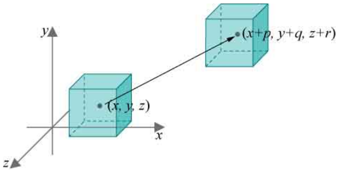
Flytte objekter

We har brug for at flytte/transformere vores objekter i rummet.

Flytte objekter

We har brug for at flytte/transformere vores objekter i rummet.

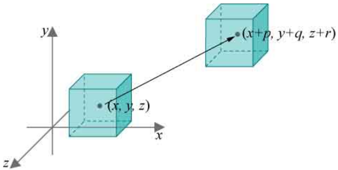
- En model (kasse, bil, bygning, person, . . .) er defineret i én position (ofte rundt om centrum i koordinatsystemet), men skal bruges i en anden position i scenen.
- Måske i en anden størrelse end oprindeligt defineret.
- Måske i adskillige positioner i samme scene (by med huse og biler).
- Måske i forskellige positioner i forskellig scener/frames (animation).



Flytte objekter

We har brug for at flytte/transformere vores objekter i rummet.

- En model (kasse, bil, bygning, person,...) er defineret i én position (ofte rundt om centrum i koordinatsystemet), men skal bruges i en anden position i scenen.
- Måske i en anden størrelse end oprindeligt defineret.
- Måske i adskillige positioner i samme scene (by med huse og biler).
- Måske i forskellige positioner i forskellig scener/frames (animation).

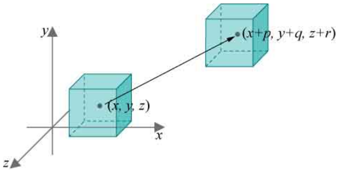


Flytte model ⇔ flytte trekanter ⇔ flytte hjørnepunkter

Flytte objekter

We har brug for at flytte/transformere vores objekter i rummet.

- En model (kasse, bil, bygning, person,...) er defineret i én position (ofte rundt om centrum i koordinatsystemet), men skal bruges i en anden position i scenen.
- Måske i en anden størrelse end oprindeligt defineret.
- Måske i adskillige positioner i samme scene (by med huse og biler).
- Måske i forskellige positioner i forskellig scener/frames (animation).



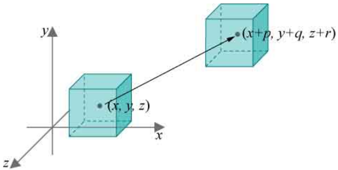
Flytte model ⇔ flytte trekanter ⇔ flytte hjørnepunkter

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = f(x, y, z)$$

Flytte objekter

We har brug for at flytte/transformere vores objekter i rummet.

- En model (kasse, bil, bygning, person,. . .) er defineret i én position (ofte rundt om centrum i koordinatsystemet), men skal bruges i en anden position i scenen.
- Måske i en anden størrelse end oprindeligt defineret.
- Måske i adskillige positioner i samme scene (by med huse og biler).
- Måske i forskellige positioner i forskellig scener/frames (animation).

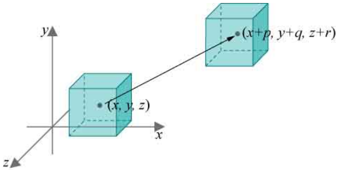


Flytte model ⇔ flytte trekanter ⇔ flytte hjørnepunkter

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = f(x, y, z)$$

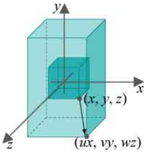
Opgave: find funktionerne vi skal bruge

Translation



$$f(x,y,z) = \begin{pmatrix} x+p \\ y+q \\ z+r \end{pmatrix}$$

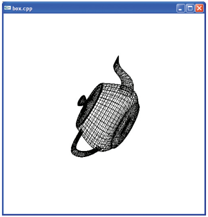
Skalering



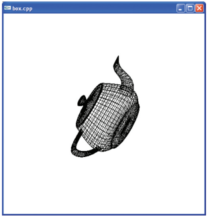
$$f(x, y, z) = \begin{pmatrix} u \cdot x \\ v \cdot y \\ w \cdot z \end{pmatrix}$$

(Ofte $u = v = w$, uniform skalering).

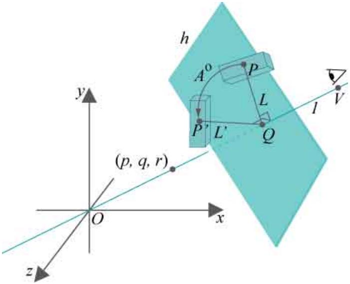
Rotation



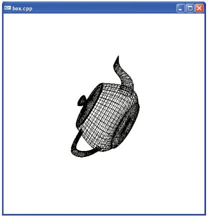
Rotation



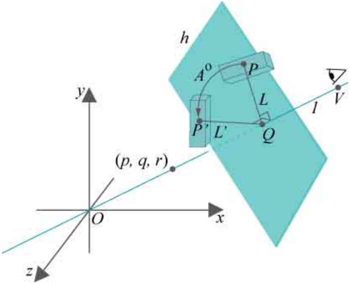
Generelt: rotation omkring en linie gennem (0,0,0):



Rotation



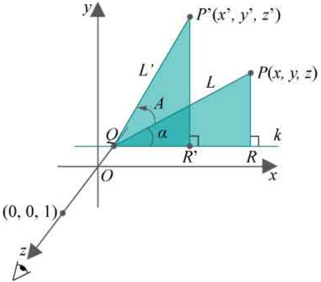
Generelt: rotation omkring en linie gennem (0,0,0):



$$f(x, y, z) = \begin{pmatrix} ? \\ ? \\ ? \end{pmatrix}$$

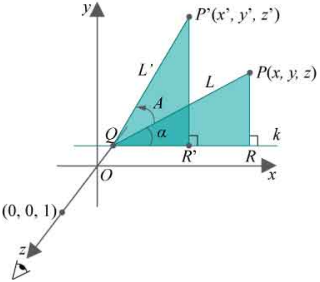
Rotation

Simpler tilfælde: Rotation om z-aksen.



Rotation

Simpler tilfælde: Rotation om z-aksen.



Brug formel for rotation i 2D:

$$f(x, y, z) = \begin{pmatrix} x\cos\phi - y\sin\phi \\ x\sin\phi + y\cos\phi \\ z \end{pmatrix}$$

Rotation

Tilsvarende: Rotation om x-aksen og y-aksen.

$$f(x,y,z) = \begin{pmatrix} x \\ y\cos\phi - z\sin\phi \\ y\sin\phi + z\cos\phi \end{pmatrix}$$

$$f(x,y,z) = \begin{pmatrix} z\sin\phi + x\cos\phi \\ y \\ z\cos\phi - x\sin\phi \end{pmatrix}$$

Mellemspil

$$2 + 3 + 4 =$$

Mellemspil

$$2 + 3 + 4 = 9$$

Mellemspil

$$2 + 3 + 4 = 9$$

$$2 + (3 + 4) \text{ eller } (2 + 3) + 4?$$

Mellemspil

$$2 + 3 + 4 = 9$$

$$2 + (3 + 4) \text{ eller } (2 + 3) + 4?$$

Ligegyldigt, plus er associativ:

$$a + (b + c) = (a + b) + c$$

Mellemspil

$$2 \cdot 3 \cdot 4 =$$

Mellemspil

$$2 \cdot 3 \cdot 4 = 24$$

Mellemspil

$$2 \cdot 3 \cdot 4 = 24$$

$2 \cdot (3 \cdot 4)$ eller $(2 \cdot 3) \cdot 4$?

Mellemspil

$$2 \cdot 3 \cdot 4 = 24$$

$$2 \cdot (3 \cdot 4) \text{ eller } (2 \cdot 3) \cdot 4?$$

Ligegyldigt, gange er associativ:

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

Mellemspil

$$2 - 3 - 4 =$$

Mellemspil

$$2 - 3 - 4 = ?$$

$2 - (3 - 4)$ eller $(2 - 3) - 4$?

Mellemspil

$$2 - 3 - 4 = ?$$

$$2 - (3 - 4) \text{ eller } (2 - 3) - 4?$$

Ikke ligegyldigt, minus er ikke associativ:

$$2 - (3 - 4) = 3$$

$$(2 - 3) - 4 = -5$$

Mellemspil

Plus, gange, minus. . . :

$$\text{tal og tal} \rightarrow \text{nyt tal.}$$

Mellemspil

Plus, gange, minus. . . :

$$\text{tal og tal} \rightarrow \text{nyt tal.}$$

Plus, gange, minus,. . . på andre matematiske objekter?

Mellemspil

Plus, gange, minus. . . :

$$\text{tal og tal} \to \text{nyt tal.}$$

Plus, gange, minus,. . . på andre matematiske objekter?

F.eks. vektorer har et plus:

$$\begin{pmatrix} 2 \\ 5 \\ 7 \end{pmatrix} + \begin{pmatrix} 1 \\ 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 2+1 \\ 5+3 \\ 7+4 \end{pmatrix} = \begin{pmatrix} 3 \\ 8 \\ 11 \end{pmatrix}$$

Mellemspil

Plus, gange, minus. . . :

$$\text{tal og tal} \to \text{nyt tal.}$$

Plus, gange, minus,. . . på andre matematiske objekter?

F.eks. vektorer har et plus:

$$\begin{pmatrix}2\\5\\7\end{pmatrix} + \begin{pmatrix}1\\3\\4\end{pmatrix} = \begin{pmatrix}2+1\\5+3\\7+4\end{pmatrix} = \begin{pmatrix}3\\8\\11\end{pmatrix}$$

$$\begin{pmatrix}2\\5\\7\end{pmatrix} \oplus \begin{pmatrix}1\\3\\4\end{pmatrix} = \begin{pmatrix}2+1\\5+3\\7+4\end{pmatrix} = \begin{pmatrix}3\\8\\11\end{pmatrix}$$

Mellemspil

Plus, gange, minus. . . :

$$\text{tal og tal} \to \text{nyt tal.}$$

Plus, gange, minus,. . . på andre matematiske objekter?

F.eks. vektorer har et plus:

$$\begin{pmatrix}2\\5\\7\end{pmatrix} + \begin{pmatrix}1\\3\\4\end{pmatrix} = \begin{pmatrix}2+1\\5+3\\7+4\end{pmatrix} = \begin{pmatrix}3\\8\\11\end{pmatrix}$$

$$\begin{pmatrix}2\\5\\7\end{pmatrix} \oplus \begin{pmatrix}1\\3\\4\end{pmatrix} = \begin{pmatrix}2+1\\5+3\\7+4\end{pmatrix} = \begin{pmatrix}3\\8\\11\end{pmatrix}$$

(vektor + vektor = ny vektor)

Mellemspil

Plus, gange, minus... :

$$\text{tal og tal} \to \text{nyt tal}.$$

Plus, gange, minus,... på andre matematiske objekter?

F.eks. vektorer har et plus:

$$\begin{pmatrix}2\\5\\7\end{pmatrix} + \begin{pmatrix}1\\3\\4\end{pmatrix} = \begin{pmatrix}2+1\\5+3\\7+4\end{pmatrix} = \begin{pmatrix}3\\8\\11\end{pmatrix}$$

$$\begin{pmatrix}2\\5\\7\end{pmatrix} \oplus \begin{pmatrix}1\\3\\4\end{pmatrix} = \begin{pmatrix}2+1\\5+3\\7+4\end{pmatrix} = \begin{pmatrix}3\\8\\11\end{pmatrix}$$

$$(\text{vektor} + \text{vektor} = \text{ny vektor})$$

Har vektorer et gange?

Matricer

Vektor = sekvens (søjle eller række) af tal: $\begin{pmatrix} 2 \\ 5 \\ 7 \end{pmatrix}$ eller $(2, 5, 7)$.

Matricer

Vektor = sekvens (søjle eller række) af tal: $\begin{pmatrix} 2 \\ 5 \\ 7 \end{pmatrix}$ eller $(2, 5, 7)$.

Nyt matematisk objekt:

Matricer

Vektor = sekvens (søjle eller række) af tal: $\begin{pmatrix} 2 \\ 5 \\ 7 \end{pmatrix}$ eller $(2, 5, 7)$.

Nyt matematisk objekt:

Matrix = firkant af tal:

$$\begin{bmatrix} 1 & 3 & 4 & 1 \\ 2 & 5 & 6 & 7 \\ 9 & 1 & 1 & 0 \end{bmatrix}$$

Ovenstående er en 3×4 matrix.

Matricer

Vektor = sekvens (søjle eller række) af tal: $\begin{pmatrix} 2 \\ 5 \\ 7 \end{pmatrix}$ eller $(2, 5, 7)$.

Nyt matematisk objekt:

Matrix = firkant af tal:

$$\begin{bmatrix} 1 & 3 & 4 & 1 \\ 2 & 5 & 6 & 7 \\ 9 & 1 & 1 & 0 \end{bmatrix}$$

Ovenstående er en 3×4 matrix.

Kan vi lave et plus og et gange for matricer?

Matricer

Plus for matricer:

$$\begin{bmatrix}1 & 6 & 4\\2 & 5 & 7\\9 & 1 & 1\end{bmatrix} + \begin{bmatrix}3 & 2 & 1\\4 & 3 & 2\\5 & 4 & 3\end{bmatrix} = \begin{bmatrix}1+3 & 6+2 & 4+1\\2+4 & 5+3 & 7+2\\9+5 & 1+4 & 1+3\end{bmatrix} = \begin{bmatrix}4 & 8 & 5\\6 & 8 & 9\\14 & 5 & 4\end{bmatrix}$$

(Matrice + matrice = ny matrice)

Matricer

Gange for matricer:

$$\begin{bmatrix} 1 & 6 & 4 \\ 2 & 5 & 7 \\ 9 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 2 & 1 \\ 4 & 3 & 2 \\ 5 & 4 & 3 \end{bmatrix} = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix}$$

$$(\text{Matrice} \cdot \text{matrice} = \text{ny matrice})$$

Matricer

Gange for matricer:

$$\begin{bmatrix} 1 & 6 & 4 \\ 2 & 5 & 7 \\ 9 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 2 & 1 \\ 4 & 3 & 2 \\ 5 & 4 & 3 \end{bmatrix} = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix}$$

(Matrice $\cdot$ matrice $=$ ny matrice)

$$\begin{bmatrix} 1 & 6 & 4 \\ \cancel{2} & \cancel{5} & \cancel{7} \\ 9 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 2 & \cancel{1} \\ 4 & 3 & \cancel{2} \\ 5 & 4 & \cancel{3} \end{bmatrix} = \begin{bmatrix} ? & ? & ? \\ ? & ? & 33 \\ ? & ? & ? \end{bmatrix}$$

$$33 = 2 \cdot 1 + 5 \cdot 2 + 7 \cdot 3$$

Matricer

Gange for matricer:

$$\begin{bmatrix} 1 & 6 & 4 \\ 2 & 5 & 7 \\ 9 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 2 & 1 \\ 4 & 3 & 2 \\ 5 & 4 & 3 \end{bmatrix} = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix}$$

$$(\text{Matrice} \cdot \text{matrice} = \text{ny matrice})$$

$$\begin{bmatrix} 1 & 6 & 4 \\ 2 & 5 & 7 \\ 9 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 2 & 1 \\ 4 & 3 & 2 \\ 5 & 4 & 3 \end{bmatrix} = \begin{bmatrix} ? & ? & ? \\ ? & ? & 33 \\ ? & ? & ? \end{bmatrix}$$

$$33 = 2 \cdot 1 + 5 \cdot 2 + 7 \cdot 3$$

$$\begin{bmatrix} 1 & 6 & 4 \\ 2 & 5 & 7 \\ 9 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 2 & 1 \\ 4 & 3 & 2 \\ 5 & 4 & 3 \end{bmatrix} = \begin{bmatrix} ? & ? & ? \\ ? & ? & 33 \\ ? & 25 & ? \end{bmatrix}$$

$$25 = 9 \cdot 2 + 1 \cdot 3 + 1 \cdot 4$$

Matricer

Gange for matricer:

$$\begin{bmatrix} 1 & 6 & 4 \\ 2 & 5 & 7 \\ 9 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 2 & 1 \\ 4 & 3 & 2 \\ 5 & 4 & 3 \end{bmatrix} = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix}$$

$$(\text{Matrice} \cdot \text{matrice} = \text{ny matrice})$$

$$\begin{bmatrix} 1 & 6 & 4 \\ 2 & 5 & 7 \\ 9 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 2 & 1 \\ 4 & 3 & 2 \\ 5 & 4 & 3 \end{bmatrix} = \begin{bmatrix} ? & ? & ? \\ ? & ? & {\color{red}33} \\ ? & ? & ? \end{bmatrix}$$

$$33 = 2 \cdot 1 + 5 \cdot 2 + 7 \cdot 3$$

$$\begin{bmatrix} 1 & 6 & 4 \\ 2 & 5 & 7 \\ 9 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 2 & 1 \\ 4 & 3 & 2 \\ 5 & 4 & 3 \end{bmatrix} = \begin{bmatrix} ? & ? & ? \\ ? & ? & 33 \\ ? & {\color{red}25} & ? \end{bmatrix}$$

$$25 = 9 \cdot 2 + 1 \cdot 3 + 1 \cdot 4$$

Man kan vise at gange for matricer er associativt: $A \cdot (B \cdot C) = (A \cdot B) \cdot C$.

Tilbage til at flytte objekter

Flytte model $\Leftrightarrow$ flytte trekanter $\Leftrightarrow$ flytte hjørnepunkter

Vi skal bruge funktioner $f(x, y, z) = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$

Tilbage til at flytte objekter

Flytte model $\Leftrightarrow$ flytte trekanter $\Leftrightarrow$ flytte hjørnepunkter

Vi skal bruge funktioner $f(x,y,z) = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$

Enhver matrice kan give en sådan funktion på flg. måde:

$$f\left(x,y,z\right) = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1x + 2y + 3z \\ 4x + 5y + 6z \\ 7x + 8y + 9z \end{pmatrix}$$

Tilbage til at flytte objekter

Flytte model $\Leftrightarrow$ flytte trekanter $\Leftrightarrow$ flytte hjørnepunkter

Vi skal bruge funktioner $f(x, y, z) = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$

Enhver matrice kan give en sådan funktion på flg. måde:

$$f(x, y, z) = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1x + 2y + 3z \\ 4x + 5y + 6z \\ 7x + 8y + 9z \end{pmatrix}$$

Spørgsmål: kan vores ønskede transformationer (transformation, skalering, rotation, projektion) udtrykkes ved matricer?

Tilbage til at flytte objekter

Kan alle vores ønskede transformationer blive udtrykt ved matricer?

$$f(x, y, z) = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1x + 2y + 3z \\ 4x + 5y + 6z \\ 7x + 8y + 9z \end{pmatrix}$$

Tilbage til at flytte objekter

Kan alle vores ønskede transformationer blive udtrykt ved matricer?

$$f(x, y, z) = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1x + 2y + 3z \\ 4x + 5y + 6z \\ 7x + 8y + 9z \end{pmatrix}$$

Hvad skulle vi få ud af det?

Tilbage til at flytte objekter

Kan alle vores ønskede transformationer blive udtrykt ved matricer?

$$f(x, y, z) = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1x + 2y + 3z \\ 4x + 5y + 6z \\ 7x + 8y + 9z \end{pmatrix}$$

Hvad skulle vi få ud af det?

Recall: Matrix gange er associativt: $A \cdot (B \cdot C) = (A \cdot B) \cdot C$.

Tilbage til at flytte objekter

Kan alle vores ønskede transformationer blive udtrykt ved matricer?

$$f(x,y,z) = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1x + 2y + 3z \\ 4x + 5y + 6z \\ 7x + 8y + 9z \end{pmatrix}$$

Hvad skulle vi få ud af det?

Recall: Matrix gange er associativt: $A \cdot (B \cdot C) = (A \cdot B) \cdot C$. Derfor:

$$A \cdot (B \cdot (C \cdot (E \cdot (F \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix})))) = ((((A \cdot B) \cdot C) \cdot E) \cdot F) \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Tilbage til at flytte objekter

Kan alle vores ønskede transformationer blive udtrykt ved matricer?

$$f(x,y,z) = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1x + 2y + 3z \\ 4x + 5y + 6z \\ 7x + 8y + 9z \end{pmatrix}$$

Hvad skulle vi få ud af det?

Recall: Matrix gange er associativt: $A \cdot (B \cdot C) = (A \cdot B) \cdot C$. Derfor:

$$A \cdot (B \cdot (C \cdot (E \cdot (F \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix})))) = ((((A \cdot B) \cdot C) \cdot E) \cdot F) \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Dette kan spare beregninger: 3D model = mange trekanter = mange punkter. Alle punkter i model skal gennem de samme transformationer. Men $(((A \cdot B) \cdot C) \cdot E) \cdot F$ skal kun beregnes én gang.

Kan vores transformationer udtrykkes ved matricer?

Kan vores transformationer udtrykkes ved matricer?

- Skalering:

$$f(x, y, z) = \begin{pmatrix} u \cdot x \\ v \cdot y \\ w \cdot z \end{pmatrix}$$

Kan vores transformationer udtrykkes ved matricer?

- Skalering:

$$f(x, y, z) = \begin{pmatrix} u \cdot x \\ v \cdot y \\ w \cdot z \end{pmatrix} = \begin{bmatrix} u & 0 & 0 \\ 0 & v & 0 \\ 0 & 0 & w \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Kan vores transformationer udtrykkes ved matricer?

- Skalering:

$$f(x,y,z) = \begin{pmatrix} u \cdot x \\ v \cdot y \\ w \cdot z \end{pmatrix} = \begin{bmatrix} u & 0 & 0 \\ 0 & v & 0 \\ 0 & 0 & w \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

- Rotation vinkel ϕ om z-aksen:

$$f(x,y,z) = \begin{pmatrix} x\cos\phi - y\sin\phi \\ x\sin\phi + y\cos\phi \\ z \end{pmatrix}$$

Kan vores transformationer udtrykkes ved matricer?

- Skalering:

$$f(x, y, z) = \begin{pmatrix} u \cdot x \\ v \cdot y \\ w \cdot z \end{pmatrix} = \begin{bmatrix} u & 0 & 0 \\ 0 & v & 0 \\ 0 & 0 & w \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

- Rotation vinkel ϕ om z-aksen:

$$f(x, y, z) = \begin{pmatrix} x \cos\phi - y \sin\phi \\ x \sin\phi + y \cos\phi \\ z \end{pmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Kan vores transformationer udtrykkes ved matricer?

- Skalering:

$$f(x,y,z) = \begin{pmatrix} u \cdot x \\ v \cdot y \\ w \cdot z \end{pmatrix} = \begin{bmatrix} u & 0 & 0 \\ 0 & v & 0 \\ 0 & 0 & w \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

- Rotation vinkel ϕ om z-aksen:

$$f(x,y,z) = \begin{pmatrix} x\cos\phi - y\sin\phi \\ x\sin\phi + y\cos\phi \\ z \end{pmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

- Translation?

$$f(x,y,z) = \begin{pmatrix} x + p \\ y + q \\ z + r \end{pmatrix} = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Kan vores transformationer udtrykkes ved matricer?

- Skalering:

$$f(x,y,z) = \begin{pmatrix} u \cdot x \\ v \cdot y \\ w \cdot z \end{pmatrix} = \begin{bmatrix} u & 0 & 0 \\ 0 & v & 0 \\ 0 & 0 & w \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

- Rotation vinkel ϕ om z-aksen:

$$f(x,y,z) = \begin{pmatrix} x\cos\phi - y\sin\phi \\ x\sin\phi + y\cos\phi \\ z \end{pmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

- Translation?

$$f(x,y,z) = \begin{pmatrix} x+p \\ y+q \\ z+r \end{pmatrix} = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

  Umuligt. For translation gælder $f(0,0,0) \neq (0,0,0)$, men alle funktioner induceret af matricer har $f(0,0,0) = (0,0,0)$.

Homogenene koordinater

Gå til 4D:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Homogenene koordinater

Gå til 4D:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Og tilbage:

$$\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \rightarrow \begin{pmatrix} x/w \\ y/w \\ z/w \end{pmatrix}$$

Homogenene koordinater

Translationer (i 3D) kan nu blive udtrykt ved matricer:

$$\begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x+p \\ y+q \\ z+r \\ 1 \end{pmatrix}$$

Homogenene koordinater

Translationer (i 3D) kan nu blive udtrykt ved matricer:

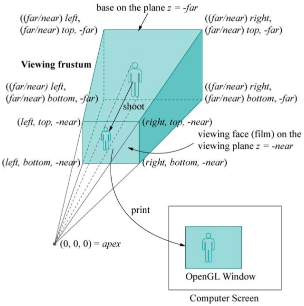
$$\begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x+p \\ y+q \\ z+r \\ 1 \end{pmatrix}$$

Alle 3x3 matricer er stadig til rådighed (inkl. skalering og rotation):

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 4 & 5 & 6 & 0 \\ 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} 1x+2y+3z \\ 4x+5y+6z \\ 7x+8y+9z \\ 1 \end{pmatrix}$$

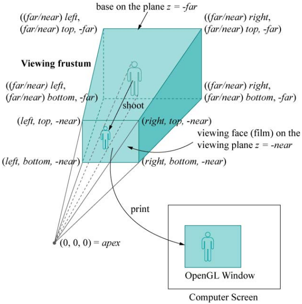
Projektion

Perspektivisk projektion:



Projektion

Perspektivisk projektion:



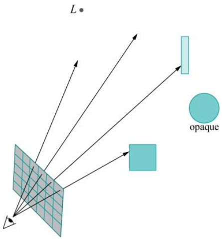
Udrykt ved en 4x4 matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z/d \end{pmatrix} \to \begin{pmatrix} xd/z \\ yd/z \\ d \end{pmatrix}$$

Shading

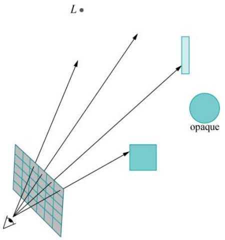
Shading

Shading = find color values at pixels of screen (when rendering a virtual 3D scene).



Shading

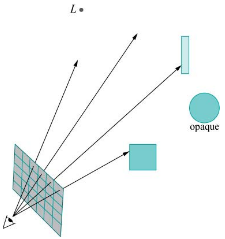
Shading = find color values at pixels of screen (when rendering a virtual 3D scene).



Same as finding color value for the closest triangle on the ray of the pixel (assuming this is an opaque object, and air is clear).

Shading

Shading = find color values at pixels of screen (when rendering a virtual 3D scene).



Same as finding color value for the closest triangle on the ray of the pixel (assuming this is an opaque object, and air is clear).

Core objective: Find color values for intersection of a ray with a triangle.

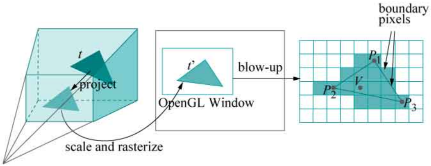
Shading

Core objective: Find color values for point at intersection of a ray with a triangle.

Shading

Core objective: Find color values for point at intersection of a ray with a triangle.

- Rendering is triangle-driven (`foreach triangle: render`).
- Triangles are simply (triples of) vertices until rasterization phase, where pixels of the triangle are found from pixels of the vertices.



So the actual rays are determined in the rasterization phase.

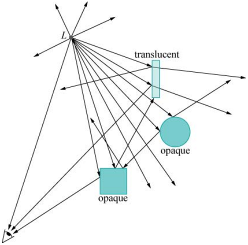
Modeling Light

Core objective: Find color values for intersection of a ray with a triangle.

Modeling Light

Core objective: Find color values for intersection of a ray with a triangle.

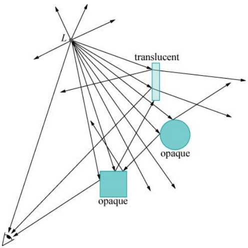
Model physical light (photons)



Modeling Light

Core objective: Find color values for intersection of a ray with a triangle.

Model physical light (photons)



Photons are

- Emitted from ligth sources.
- Reflected, absorbed, re-emitted, transmitted when hitting objects.

Modeling Light

Highly complex physical proces. Zillions of photons.

Can only be modeled to a certain degree mathematically (ongoing research expands on the available models).

Modeling Light

Highly complex physical proces. Zillions of photons.

Can only be modeled to a certain degree mathematically (ongoing research expands on the available models).

(Figure by Jason Jacobs)

Modeling Light

Realtime rendering additionally has severe time constraints. Framerate $\sim 30/\text{sec}$, screen size $\sim 10^6$ pixels $\Rightarrow$ few GPU cycles available for calculation per ray.

Modeling Light

Realtime rendering additionally has severe time constraints. Framerate $\sim 30/\text{sec}$, screen size $\sim 10^6$ pixels $\Rightarrow$ few GPU cycles available for calculation per ray.

Hence, *realtime* rendering (games, simulators) use quite rough light models.

Modeling Light

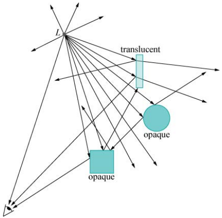
Realtime rendering additionally has severe time constraints. Framerate $\sim$ 30/sec, screen size $\sim 10^6$ pixels $\Rightarrow$ few GPU cycles available for calculation per ray.

Hence, *realtime* rendering (games, simulators) use quite rough light models.

Offline rendering (movies, visualization) can use more advanced light models (and also other rendering methods needing more time, such as ray tracing and radiosity).

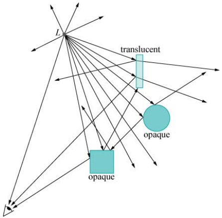
Phongs Lightning Model

A classic, simple model.



Phongs Lightning Model

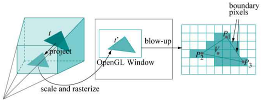
A classic, simple model.



- Models only opaque objects.
- Models only one level of light/surface interactions.
- Light/surface interaction is modeled by two simple submodels, diffuse and specular term.
- Models indirect light effects very crudely (ambient term).
- Light actually generated at surface can be added (emissive term).
- Occlusion is not modeled (all objects see all lights).

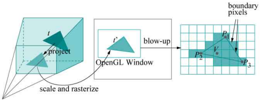
Shading models

So we have information in each vertex. How spread color calculation over entire triangle pixels?



Shading models

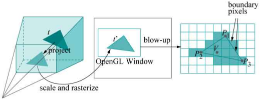
So we have information in each vertex. How spread color calculation over entire triangle pixels?



- Flat shading: Color calculated for one point is used for entire triangle.

Shading models

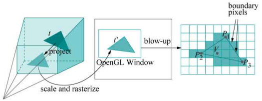
So we have information in each vertex. How spread color calculation over entire triangle pixels?



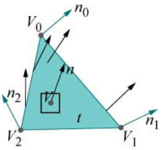
- Flat shading: Color calculated for one point is used for entire triangle.
- Smooth shading (aka. Gouraud shading): Colors calculated for three vertices are interpolated across the entire triangle (individually for each RGB-channel).

Shading models

So we have information in each vertex. How spread color calculation over entire triangle pixels?

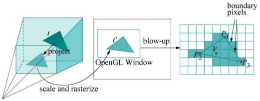


- Flat shading: Color calculated for one point is used for entire triangle.
- Smooth shading (aka. Gouraud shading): Colors calculated for three vertices are interpolated across the entire triangle (individually for each RGB-channel).
- Phong shading: Color calculation done for all points of pixels.

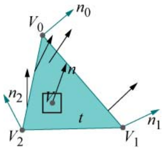


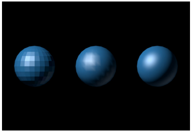
Shading models

So we have information in each vertex. How spread color calculation over entire triangle pixels?



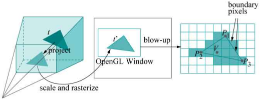
- Flat shading: Color calculated for one point is used for entire triangle.
- Smooth shading (aka. Gouraud shading): Colors calculated for three vertices are interpolated across the entire triangle (individually for each RGB-channel).
- Phong shading: Color calculation done for all points of pixels.



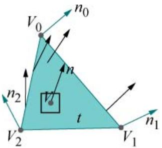


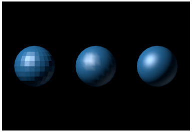
Shading models

So we have information in each vertex. How spread color calculation over entire triangle pixels?



- Flat shading: Color calculated for one point is used for entire triangle.
- Smooth shading (aka. Gouraud shading): Colors calculated for three vertices are interpolated across the entire triangle (individually for each RGB-channel).
- Phong shading: Color calculation done for all points of pixels.





Calculation time increases down the list.

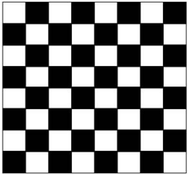
Textures

Texture $=$ $1/2/3$D data table.

Textures

Texture = 1/2/3D data table.

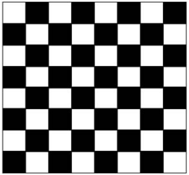
Often: (color values of) 2D picture.



Textures

Texture = 1/2/3D data table.

Often: (color values of) 2D picture.

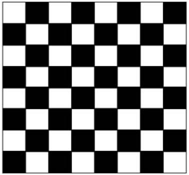


External file, or generated online inside program (animated textures), or renderede (offline or online) scene.

Textures

Texture = 1/2/3D data table.

Often: (color values of) 2D picture.



External file, or generated online inside program (animated textures), or renderede (offline or online) scene.

But texture data can be interpreted as anything, e.g. normal vectors, light maps/shadow maps, heightfields,. . . .

Use of Textures

- Generate detailed graphical content hardly possible with triangles (such as clouds, skyboxes, plain pictures (posters, decals) on surfaces).
- Create illusion of structure, saving lots of triangles. Can be (low level) part of a level-of-detail scheme.
- Most of a game's graphical expression is via artwork using textures.
- Hold special-purpose data for use in rendering process.

Examples

(From All Things Designed)

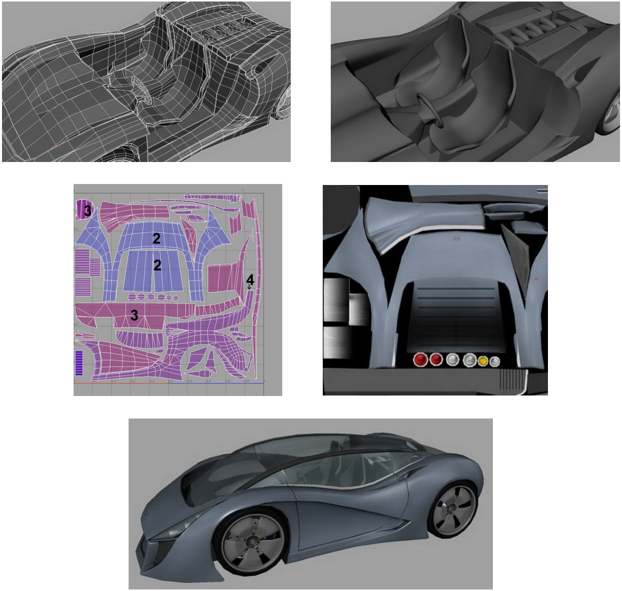
Examples

Examples

Examples

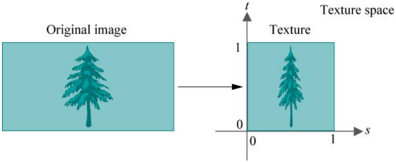
(From Sly Cooper)

Examples



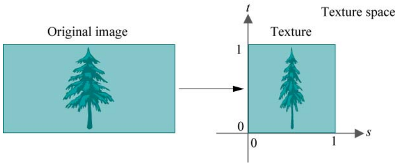
(Figures by Valentin Nadolu)

Texture Coordinates



Texture data get mapped to $[0;1]^{1,2,3}$ in texture space.

Texture Coordinates

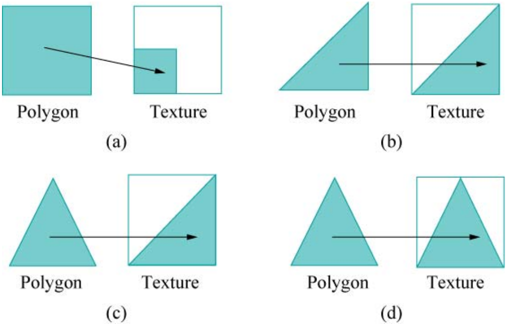


Texture data get mapped to $[0;1]^{1,2,3}$ in texture space.

Vertices can be associated with texture coordinates

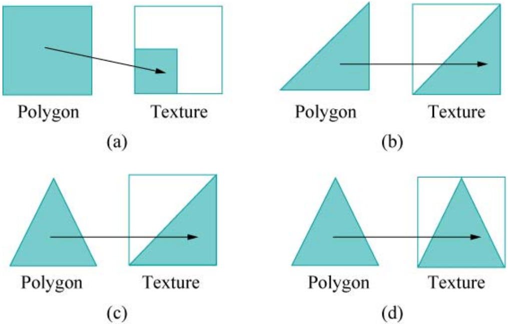
Texture Coordinates

Texture space points can be arbitrary:



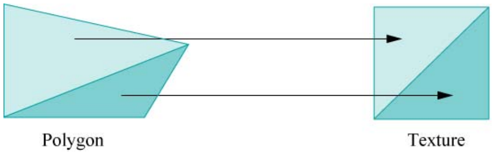
Texture Coordinates

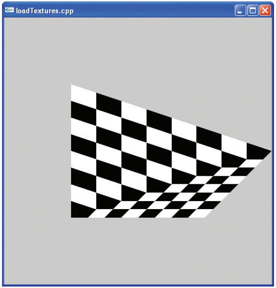
Texture space points can be arbitrary:



Points internally in triangle are associated with points in texture space using interpolation.

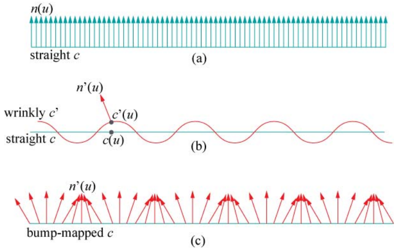
Interpolation Example





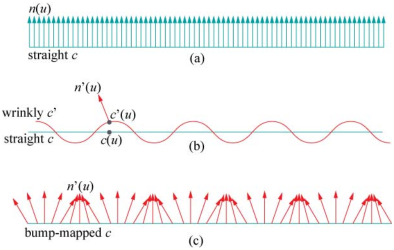
Texture Use: Bumpmapping

Store surface normals (or perturbation of normals) in texture.



Texture Use: Bumpmapping

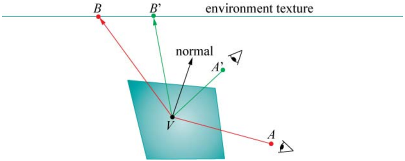
Store surface normals (or perturbation of normals) in texture.



(Figure by www.chromesphere.com)

Texture Use: Environment Mapping

Relections can see environment. Make part of shading calculation.



Environment Mapping

Easiest with Cube mapping:

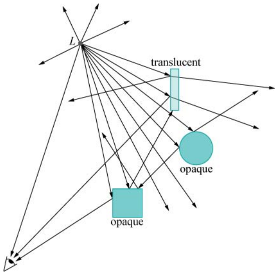
Six single textures. Can each be generated online by rendering from current center and saving framebuffer as texture.

Alternative Rendering Methods

- Standard GPU pipeline (OpenGL): real-time, but shading based on *local* effects. No shadows in basic pipeline (must be added by ad-hoc methods).
- Ray tracing: *Global* shading model particularly good at specular effects (shiny surfaces). Too computationally expensive to be real-time.
- Radiosity: *Global* shading model particularly good at diffuse effects (matte surfaces, indirect light). Too computationally expensive to be real-time. But well suited for storing results as textures (as diffuse light is not viewpoint dependent).

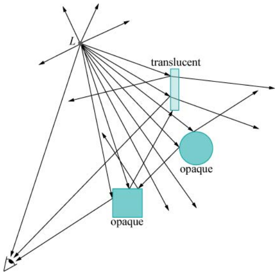
Ray Tracing

Follow photon paths to the eye.



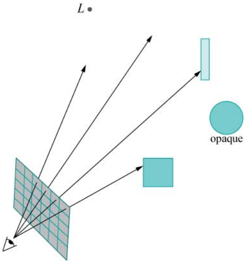
Ray Tracing

Follow photon paths to the eye.



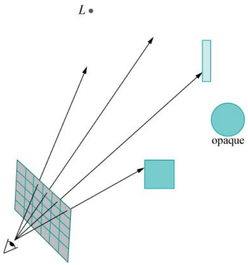
For efficiency, follow these in a backwards fashion, from the eye (only spend time on photons actually hitting the eye).

Ray Tracing Level 0



At end of rays: calculate colors by Phongs lighting model.

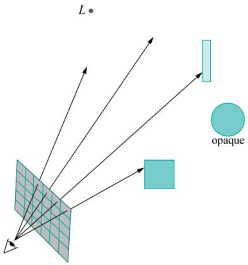
Ray Tracing Level 0



At end of rays: calculate colors by Phongs lighting model.

Same result as standard GPU pipeline.

Ray Tracing Level 0



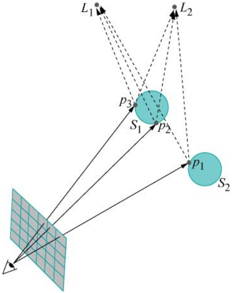
At end of rays: calculate colors by Phongs lighting model.

Same result as standard GPU pipeline.

Requires mechanism for fast determination of intersection points between rays and objects of the scene (e.g., store objects in data structures).

Ray Tracing Level 1

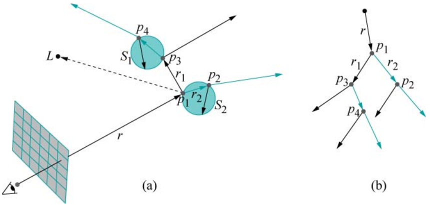
Add occlusion tests to light sources.



Gives shadows.

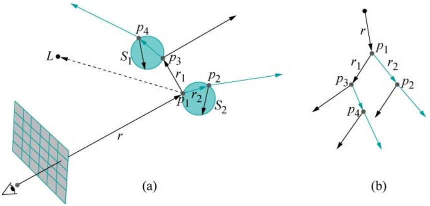
Ray Tracing Level 2+

Add reflection and transmission. Then *recurse*.



Ray Tracing Level 2+

Add reflection and transmission. Then *recurse*.



Note: simulating indirect light transfer between diffuse surfaces requires following many (approximating infinitely many) reflective rays from each ray intersection point in the recursive process.

Prohibitively costly. So ray tracing works best for glossy materials.

Ray Tracing Examples

(Figures by Bill Martin, RayScale, Daniel Pohl, NVIDIA)

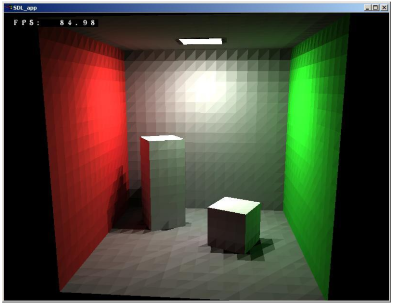
Radiosity

Model indirect light bouncing between purely diffuse (Lambertian) surfaces (of which some are light emitting).

(Figure by Jason Jacobs)

Patches

Start by patch-ifying the surfaces of the scene.



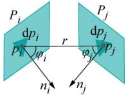
(Figure by Chuck Pheatt)

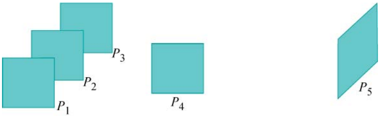
Entire patch will be considered to have same ligth value (radiosity/brightness) B_i.

Radiosity: photons emitted per time and per area.

Form Factors

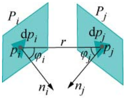
Form factor F_{ij}: measure of light transport between patch i and j.

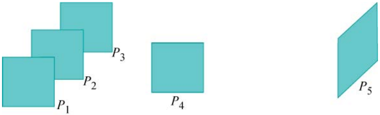




Form Factors

Form factor F_{ij}: measure of light transport between patch i and j.





(Technically: For F_{ij}: sum (integrate) contribution between (infinitesimal small areas around) all points on the two patches P_i and P_j.)

Radiosity Equation

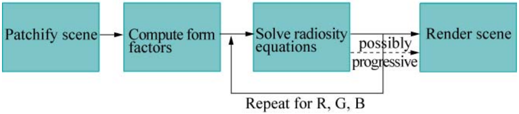
With M a specific $n \times n$ matrix (n is number of patches in schene) having entries depending on form factors and reflectance of patches, B the sought vector of brightness/radiosity values for patches and E the vector of emissive values for patches, one can prove:

$$MB = E$$

Using properties of the matrix M and results from matrix theory, it can be proven that the iterative process

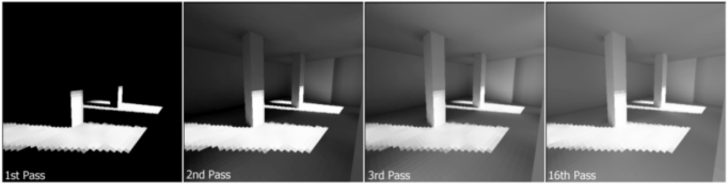
$$B_{i+1} = E + (I - M)B_i$$

for any start vector B_0 will converge to B. This is usually faster than directly solving $MB = E$ (by e.g. inverting M), and less memory is used.



Iterative Process

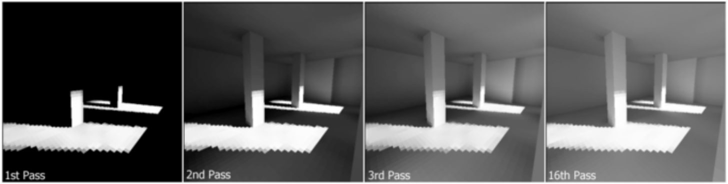
Here is the result of rendering a specific scene with B_1, B_2, B_3, B_{16}.



(Figure by Hugo Elias)

Iterative Process

Here is the result of rendering a specific scene with B_1, B_2, B_3, B_{16}.



(Figure by Hugo Elias)

The patching of the room may be refined based on one run of radiosity, increasing the resolution in areas with large variation in light values (edges of shadows, e.g.), and lowering the resolution in areas with small variation.