

## Reaalialikagrafiikan ohjelmointi – TTV14SP

Janne Koponen

15. joulukuuta 2015

## Kurssin sisältö

- 1 Vektorit
- 2 Matriisit
- 3 Lineaarikuvaukset
- 4 Kvaterniot
- 5 Koordinaatistomuunnokset
- 6 3D-objektit
- 7 3D-liukuhihna
- 8 Tekstuurit
- 9 Valaistusmallit
- 10 Shading
- 11 Varjojen reaalialkainen laskenta

## Aikataulu

- vko 44 Johdanto ja matriisien, vektoreiden ja kvaternioiden perusteita
- vko 45 Koordinaatistomuunnokset
- vko 46 3D-objektit, mallien lataaminen tiedostosta
- vko 47 3D-liukuhihna
- vko 48 Tekstuurit, valaistusmallit (ambient, distance falloff, Lambert, Oren-Nayar, Phong, Blinn-Phong), varjostimet
- vko 49 Gouraud shading, Phong shading, bump-mapping, normal-mapping, materiaalit
- vko 50 Harjoitustyö / Kurssin aikana tulleita toiveita (esim. cell shading)

## Kurssin suorittaminen

Harjoitustyö:

- “3d-demo” – Perspektiiviprojekti
- Kappaleen lataus tiedostosta
- Teksturointi
- Lambert- ja Phong-valaistusmalli
- Gouraud- ja Phong-shading, normal mapping

## Arvointi

- Kiitettävä (5): Opiskelija osaa määritellä, suunnitella, toteuttaa ja testata itsenäisesti kompleksisen 3D-grafiikkamoottorin käyttäen OpenGL-grafiikkarajapintaa. Opiskelija osoittaa asiantuntijuutta ja pystyy soveltamaan oppimaansa sekä kehittämään uutta tavoitteellisesti.
- Hyvä (3-4): Opiskelija osaa määritellä, suunnitella, toteuttaa ja testata itsenäisesti yksinkertaisen 3D-grafiikkamoottorin käyttäen OpenGL-grafiikkarajapintaa. Opiskelija toimii aloitteellisesti ja tavoitteellisesti annetuissa tehtävissä.
- Tyydyttävä (1-2): Opiskelija osaa määritellä, suunnitella, toteuttaa ja testata ohjatusti yksinkertaisen 3D-grafiikkamoottorin käyttäen OpenGL-grafiikkarajapintaa.

## Lähteitä

- Sheiner et al.: OpenGL Programming guide
- [https://en.wikibooks.org/wiki/OpenGL\\_Programming](https://en.wikibooks.org/wiki/OpenGL_Programming)
- <http://www.realtimerendering.com/>
- <https://open.gl>, <http://www.opengl-tutorial.org/> jne.

## Määritelmä

- Olkoon  $A, B \in \mathbb{R}^n$ . Tällöin voidaan piirtää suuntajana  $\overrightarrow{AB}$ .
  - Suuntajanan alkupiste on  $A$  ja loppupiste  $B$ .
  - Huomaa, että  $\overrightarrow{AB} \neq \overrightarrow{BA}$ .
  - Määritelmästä saadaan suuntajanalle pituus ja suunta.
- Määritellään vektori siten, että se edustaa kaikkien samansuuntaisten ja -suuruisten suuntajanojen joukkoa.
- Vektorilla on siis suunta ja suuruus, mutta ei tiettyä sijaintia.
  - Vektoreista voidaan tämän vuoksi valita kulloiseenkin tilanteeseen parhaiten sopiva kyseisestä joukosta.
- Vektoria merkitään yleensä  $\bar{a}$  tai  $\vec{a}$  tai  $a$ .
  - Käytetään jatkossa viimeistä notaatiota, eli vektoreita merkitään pienillä lihavoiduilla kirjaimilla.

## Vektorit koordinaatistossa

- Vektoreiden käsittely määritelmän mukaan voi olla hankala.
- Valitaan vektorille, eli vektoriluokalle, sellainen edustaja joka alkaa origosta  $O$ .
- Tällöin vektorijoukon edustaja on vektori  $\overrightarrow{Ox}$ .
  - Piste  $x$  kiinnittää tällöin yksikäsitteisesti kyseisen luokan vektoreiden edustajan.
  - Nyt voimme samaistaa avaruuden  $\mathbb{R}^n$  pisteen  $x$  ( $n$ -jonon) ja vektorin  $\mathbf{x}$ .

## Vektorin suunta ja suuruus

Olkoon annettu kaksi  $\mathbb{R}^n$ :n pistettä  $x$  ja  $y$ . Tällöin suuntajanan komponenttiesitykseksi saadaan

$$\overrightarrow{xy} = (y_1 - x_1 \ y_2 - x_2 \ \dots \ y_n - x_n).$$

Koska origon komponenttiesitys on  $(0 \ 0 \ \dots \ 0)$ , niin vektorin  $\mathbf{x}$  komponenttiesitykseksi saadaan

$$\mathbf{x} = (x_1 - 0 \ x_2 - 0 \ \dots \ x_n - 0) = (x_1 \ x_2 \ \dots \ x_n).$$

## Vektorioperaatiot

- Kaikki jatkossa esitettyt operaatiot määritellään yleisesti  $\mathbb{R}^n$  vektoreille, lukuunottamatta ristituloa.
- Ristitulo määritellään ainoastaan  $\mathbb{R}^3$  vektoreille.

Määritellään nollavektori  $\mathbf{0} = (0 \ 0 \ \dots \ 0)$ . Huomaa, että  $\mathbf{0}$  on ainoa vektori, jolla ei ole suuntaa.

Vektorin  $\mathbf{x}$  pituus, eli normi

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2} \quad (1)$$

Määritellään lisäksi, että pisteiden  $x$  ja  $y$  välinen etäisyys on

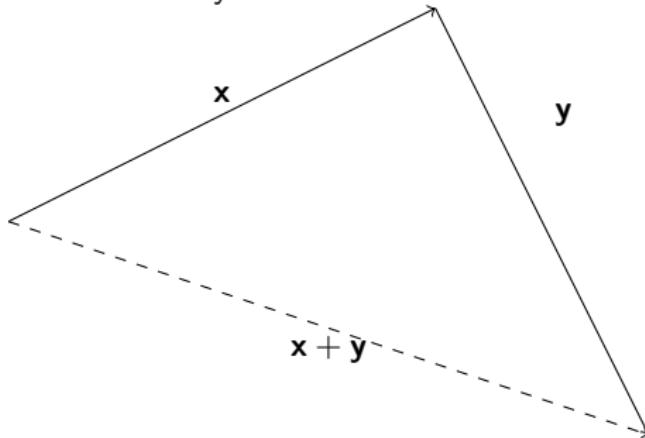
$$\|\overrightarrow{x-y}\| = \sqrt{y_1 - x_1, y_2 - x_2, \dots, y_n - x_n}.$$

## Yhteenlasku

Määritellään  $\mathbb{R}^n$  vektoreiden  $\mathbf{x}$  ja  $\mathbf{y}$  väliset operaatiot:

$$\mathbf{x} + \mathbf{y} = (x_1 + y_1 \ x_2 + y_2 \ \dots \ x_n + y_n) . \quad (2)$$

Geometrisesti yhteenlasku tarkoittaa "kahden vektorin peräkkäin asettamista".



## Skalaarilla kertominen

Skalaarilla tarkoitetaan  $\mathbb{R}$ :n lukua. Olkoon annettu skalaari  $a \in \mathbb{R}$  ja vektori  $\mathbf{x} \in \mathbb{R}^n$ . Määritellään näiden tulo

$$a\mathbf{x} = (ax_1 \ ax_2 \ \dots \ ax_n) \tag{3}$$

Geometrisesti tämä tarkoittaa vektorin pituuden kertomista sen suunnan säilyttäen.

Määritellään vektorin  $\mathbf{x}$  vastavektori  $-\mathbf{x} = -1\mathbf{x}$ .

## Vähennyslasku

Määritellään vektoreiden vähennyslasku käyttäen yhteenlaskua ja vektorin vastavektoria. Olkoon annettu kaksi vektoria  $\mathbf{x}$  ja  $\mathbf{y} \in \mathbb{R}^n$ . Tällöin

$$\mathbf{x} - \mathbf{y} = \mathbf{x} + (-\mathbf{y}) = (x_1 - y_1 \ x_2 - y_2 \ \dots \ x_n - y_n) . \quad (4)$$

# Harjoitukset VEKT.1

- ① Osoita, että  $\|\mathbf{x} - \mathbf{y}\| = \|\mathbf{y} - \mathbf{x}\|$ .
- ② Osoita, että määritelmän (4) askel  $\mathbf{x} + (-\mathbf{y}) = (x_1 - y_1 \ x_2 - y_2 \ \dots \ x_n - y_n)$  on oikein.
- ③ Osoita, että jos  $a \in \mathbb{R}$  ja  $\mathbf{x} \in \mathbb{R}^n$ , niin  $\|a\mathbf{x}\| = |a|\|\mathbf{x}\|$ .
- ④ Toteuta luokkamalli `vector_t<class T>`.
  - Toteuta kaikki edellä esitettyt laskuoperaatiot sopivilla operaattoreilla.

## Yksikkövektori

- Vektori  $\mathbf{x}$  on yksikkövektori, mikäli sen pituus  $\|\mathbf{x}\| = 1$ .
- Jokaisesta vektorista  $\mathbf{x} \neq \mathbf{0}$  voidaan muodostaa yksikkövektori kertomalla se skalaarilla  $\frac{1}{\|\mathbf{x}\|}$ . Tätä toimenpidettä kutsutaan normalisoinniksi.
- Normalisoinnissa vektorin suunta säilyy, mutta pituus voi muuttua.
- Yksikkövektori erotellaan tavallisesta vektorista asettamalla sen päälle merkintä  $\hat{\cdot}$ .
- Voidaan siis kirjoittaa

$$\hat{\mathbf{x}} = \frac{1}{\|\mathbf{x}\|} \mathbf{x}.$$

# $\sum$ -merkintä

Otetaan käyttöön summamerkintä  $\sum$ . Määritellään

$$\sum_{i=1}^n a_i = a_1 + a_2 + \dots + a_n .$$

Esimerkkejä:

$$\sum_{i=1}^n i = 1 + 2 + 3 + \dots + n ,$$

$$\sum_{i=1}^n i^2 = 1 + 4 + 9 + \dots + n^2 \text{ ja}$$

$$\sum_{i=1}^n \frac{1}{i} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} .$$

# $\sum$ -merkinnän ominaisuuksia

$\sum$ -merkinnälle pätee

$$c \sum_{i=1}^n a_i = \sum_{i=1}^n ca_i \text{ ja}$$

$$\sum_{i=1}^n b_i + \sum_{i=1}^n a_i = \sum_{i=1}^n (b_i + a_i) .$$

Huomaa erityisesti, että

$$(\sum_{i=1}^n b_i)(\sum_{i=1}^n a_i) = \sum_{i=1}^n \left( (\sum_{j=1}^n b_j) a_i \right) , \text{ mutta yleensä}$$

$$(\sum_{j=1}^n b_j)(\sum_{i=1}^n a_i) \neq \sum_{i=1}^n (b_j a_i) .$$

## Harjoituksia VEKT.2

- ① Laske  $\sum_{i=1}^5 i$ .
- ② Laske  $\sum_{i=1}^5 \frac{1}{i}$ .
- ③ Laske  $\sum_{i=1}^5 (i + \frac{1}{i})$ .
- ④ Olkoon annettu kaksi  $\mathbb{R}^3$ -vektoria  $\mathbf{x} = (1, 2, 3)$  ja  $\mathbf{y} = (3, 1, 0)$ . Laske  $\sum_{i=1}^3 (x_i y_i)$ .

## Pistetulo

Määritellään kahden vektorin  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$  pistetulo

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^n x_i y_i = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n. \quad (5)$$

Olkoon  $\mathbf{x}, \mathbf{y}$  ja  $\mathbf{z} \in \mathbb{R}^n$  sekä  $a \in \mathbb{R}$ . Tällöin pistetulolle pätee

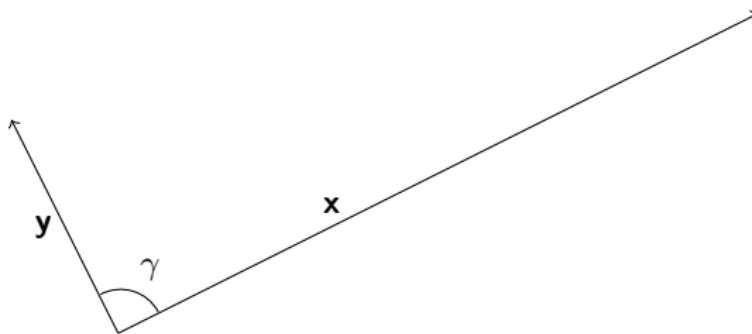
$$\begin{aligned} \mathbf{x} \cdot \mathbf{y} &= \mathbf{y} \cdot \mathbf{x}, \\ \mathbf{x} \cdot (\mathbf{y} + \mathbf{z}) &= \mathbf{x} \cdot \mathbf{y} + \mathbf{x} \cdot \mathbf{z}, \\ (a\mathbf{x}) \cdot \mathbf{y} &= a(\mathbf{x} \cdot \mathbf{y}) \text{ ja} \\ \mathbf{x} \cdot \mathbf{y} &= \|\mathbf{x}\| \|\mathbf{y}\| \cos \gamma, \end{aligned} \quad (6)$$

jossa  $\gamma$  on vektoreiden  $\mathbf{x}$  ja  $\mathbf{y}$  välinen kulma.

## Vektoreiden kohtisuorus

Olkoon vektorit  $\mathbf{x}$  ja  $\mathbf{y} \in \mathbb{R}^n$  ja  $\mathbf{x} \neq \mathbf{0} \neq \mathbf{y}$ . Jos  $\mathbf{x} \cdot \mathbf{y} = 0$ , niin tällöin  $\cos \gamma = 0$ .

Kosinifunktion ominaisuuksista tiedämme, että tällöin  $\gamma = -\frac{1}{2}\pi$  tai  $\gamma = \frac{1}{2}\pi$ , eli vektorit ovat toisiaan vastaan kohtisuorassa.



## Harjoituksia VEKT.3

- ① Laske vektorien  $\mathbf{x} = (2 \ 1 \ 3)$  ja  $\mathbf{y} = (1 \ 1 \ 1)$  pistetulo  $\mathbf{x} \cdot \mathbf{y}$  ja sen avulla vektoreiden välinen kulma.
- ② Laske vektorien  $\mathbf{x} = (1 \ 1 \ 0)$  ja  $\mathbf{y} = (0 \ 0 \ 1)$  pistetulo  $\mathbf{x} \cdot \mathbf{y}$  ja sen avulla vektoreiden välinen kulma.
- ③ Milloin  $\mathbf{x} \cdot \mathbf{x} > 0$ , milloin  $\mathbf{x} \cdot \mathbf{x} = 0$ , entä voiko olla  $\mathbf{x} \cdot \mathbf{x} < 0$ , jos voi, niin milloin?
- ④ Osoita, että  $\mathbf{x} \cdot (\mathbf{y} - \mathbf{z}) = \mathbf{x} \cdot \mathbf{y} - \mathbf{x} \cdot \mathbf{z}$
- ⑤ Osoita, että  $\|\mathbf{x}\| = \sqrt{\mathbf{x} \cdot \mathbf{x}}$ .
- ⑥ Lisää vektor\_t-luokkamalliin pistetulo. Käytä pistetulon laskentaan operaattoria  $*$ .
- ⑦ Lisää vektor\_t-luokkamalliin yksikkövektorin muodostus. Huomaa, että yksikkövektoria ei voi aina muodostaa.
- ⑧ Osoita, että jos  $\mathbf{x} \neq \mathbf{0}$  ja  $\hat{\mathbf{x}} = \frac{1}{\|\mathbf{x}\|} \mathbf{x}$ , niin  $\hat{\mathbf{x}}$  on yksikkövektori.

## Ristitulo

- Ristitulo voidaan määritellä ainoastaan  $\mathbb{R}^3$ :n vektoreille.
- Ristitulon avulla saadaan selville kahden vektorin määrittelemän tason kanssa kohtisuora suunta.

Olkoon  $\mathbf{x}$  ja  $\mathbf{y} \in \mathbb{R}^3$ . Tällöin niiden ristitulo

$$\mathbf{x} \times \mathbf{y} = (x_2y_3 - x_3y_2 \ x_3y_1 - x_1y_3 \ x_1y_2 - x_2y_1) . \quad (7)$$

Huomaa, että myös  $\mathbf{x} \times \mathbf{y}$  on myös vektori.

## Ristituloon ominaisuuksia

Olkoon  $\mathbf{x}, \mathbf{y}$  ja  $\mathbf{z} \in \mathbb{R}^3$  ja  $\gamma$  niiden välinen kulma. Tällöin

$$\|\mathbf{x} \times \mathbf{y}\| = \|\mathbf{x}\| \|\mathbf{y}\| \sin \gamma , \quad (8)$$

$$\mathbf{x} \times \mathbf{y} = -(\mathbf{y} \times \mathbf{x}) ,$$

$$\mathbf{x} \times \mathbf{x} = \mathbf{0} \text{ ja}$$

$$\mathbf{x} \times (\mathbf{y} + \mathbf{z}) = \mathbf{x} \times \mathbf{y} + \mathbf{x} \times \mathbf{z} .$$

## Harjoituksia VECT.4

- ① Osoita, että  $\mathbf{x} \times \mathbf{y} = -(\mathbf{y} \times \mathbf{x})$ .
- ② Osoita, että  $\mathbf{x} \times \mathbf{x} = \mathbf{0}$ .
- ③ Osoita, että  $\mathbf{x} \times (\mathbf{y} + \mathbf{z}) = \mathbf{x} \times \mathbf{y} + \mathbf{x} \times \mathbf{z}$ .
- ④ Olkoon annettu  $x = \begin{pmatrix} 1 & 2 & 1 \end{pmatrix}$ ,  $y = \begin{pmatrix} -1 & 0 & 0 \end{pmatrix}$  ja  $z = \begin{pmatrix} 2 & 2 & 1 \end{pmatrix}$ .  
Laske pinnan yksikkönormaali.
- ⑤ Lisää ristitulo vektor\_t-luokkamalliin.
- ⑥ Kirjoita funktio, joka laskee pinnan normaalilin kun sille syötetään kolme pistettä (homogeenisissa koordinaateissa).

## Nettilähteitä

- [http://blog.wolfire.com/2009/07/  
linear-algebra-for-game-developers-part-1/](http://blog.wolfire.com/2009/07/linear-algebra-for-game-developers-part-1/)
- <http://www.dickbaldwin.com/KjellTutorial/vectorIndex.html>
- <http://www.flipcode.com/archives/articles.shtml>

## Lineaarikuvauskset

- Kuvaus  $\mathbf{A} : \mathbb{R}^n \rightarrow \mathbb{R}^m$  on lineaarikuvaus jos:
  - 1)  $c\mathbf{A}(x) = \mathbf{A}(cx)$  ja
  - 2)  $\mathbf{A}(x) + \mathbf{A}(y) = \mathbf{A}(x + y)$  ,jossa  $c \in \mathbb{R}$  ja  $x, y \in \mathbb{R}^n$ .
- Lineaarikuvauksia ovat esimerkiksi
  - skaalaus,
  - kierto ja
  - "pienellä temppuilulla" myös siirrot.
- Matriisien avulla voidaan esittää lineaarikuvaukset avaruudesta  $\mathbb{R}^n$  avaruuteen  $\mathbb{R}^m$ .
- Lineaarikuvauxsiin palataan myöhemmin.

## Matriisin määrittely

Määritellään  $m \times n$ -matriisi lukutaulukkona, jossa on  $m$  riviä ja  $n$  saraketta. Merkitään jatkossa matriiseja isoilla lihavoiduilla kirjaimilla. Olkoon  $\mathbf{A}$   $m \times n$ -matriisi. Tällöin merkitään  $A \in \mathbb{R}^{m \times n}$ .

Olkoon esimerkiksi

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 1 & 4 \\ 2 & 4 \end{pmatrix}.$$

Tällöin  $\mathbf{A} \in \mathbb{R}^{3 \times 2}$ .

## Matriisin alkiot

Indeksoidaan matriisin alkiot yksinkertaistamaan määrittelyjä. Merkitään matriisin **A** yksittäistä alkiota  $a_{ij}$ :llä (tai  $a_{i,j}$ :llä jos pilkku on tarpeen).  $a_{ij}$  on matriisin  $i$ :nnen rivin  $j$ :s (tai  $j$ :nnen sarakkeen) alkio.

Olkoon esimerkiksi  $A \in \mathbb{R}^{3 \times 2}$ . Tällöin

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{pmatrix} .$$

## Matriisityyppejä

Olkoon  $\mathbf{A} \in \mathbb{R}^{m \times n}$ .

- Jos  $m = n$ , niin  $\mathbf{A}$ :n sanotaan olevan neliömatriisi.
  - Neliömatriisissa on yhtä monta saraketta ja riviä.
  - Neliömatriisit ovat keskeisessä asemassa tietokonegrafiikassa.
  - Neliömatriisia merkitään usein  $\mathbf{A} \in \mathbb{R}^{n \times n}$ .
- Jos  $n = 1$  sanotaan matriisia pystyvektoriksi.
  - Samaistetaan jatkossa  $\mathbb{R}^m$ :n vektorit ja  $\mathbb{R}^{m \times 1}$  matriisit.
- Jos  $m = 1$  sanotaan matriisia vaakavektoriksi.
  - Olisi mahdollista samaistaa myös vaakavektorit ja  $\mathbb{R}^l$ :n vektorit, mutta pysytään aiemmassa määritelmässä.
- Jos  $n = m = 1$ , niin samaistetaan  $\mathbf{A}$  skalaarin  $a_{1,1} \in \mathbb{R}$  kanssa.

## Diagonaalialkiot

Neliömatriisin **A** alkioita  $a_{ij}$ , joissa  $i = j$  kutsutaan **A**:n diagonaalialkioiksi.

Mikäli matriisin kaikki alkiot lukuunottamatta diagonaalialkioita ovat 0, sanotaan matriisia diagonaalimatriisiiksi.

Esimerkki

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

on diagonaalimatriisi.

Diagonaalimatriisi voidaan merkitä lyhyemmin

$$\begin{pmatrix} a_{1,1} & 0 & \dots & 0 \\ 0 & a_{2,2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{n,n} \end{pmatrix} = \begin{pmatrix} a_1 & 0 & \dots & 0 \\ 0 & a_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_n \end{pmatrix} = \text{diag}(a_1, a_2, \dots, a_n)$$

## Identiteetti- ja nollamatriisi

Määritellään  $\mathbb{R}^{n \times n}$  identiteetti- eli yksikkömatriisi

$$\mathbf{I} = \text{diag}(1, 1, \dots, 1) = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}.$$

Yksikkömatriisissa siis diagonaalialkiot ovat 1 ja kaikki muut alkiot ovat 0.

Määritellään nollamatriisi

$$\mathbf{0} = \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}.$$

Huomaa, että merkintää **0** käytetään sekä matriisille, että vektorille. Mikäli sekaannuksen vaaraa on, käytetään nollavektorille jatkossa merkintää  $\bar{0}$ .

## Harjoituksia MAT.1

- ① Matriisin alkiot kannattaa tallettaa yksiulotteisena taulukkona sarakkeittain. Toteuta tällä tavoin luokkamalli `matrix_t<class T>`, joka
  - asettaa oletusmuodostimessa matriisin arvoksi yksikkömatriisin ja
  - sisältää metodit asettamaan arvoksi nolla- ja identiteettimatriisit.

## Matriisien yhteenlasku

Matriisien yhteenlasku määritellään tapahtuvaksi alkioittain. Olkoon  $\mathbf{A}$  ja  $\mathbf{B} \in \mathbb{R}^{m \times n}$ . Tällöin

$$\mathbf{A} + \mathbf{B} = \begin{pmatrix} a_{1,1} + b_{1,1} & a_{1,2} + b_{1,2} & \dots & a_{1,n} + b_{1,n} \\ a_{2,1} + b_{2,1} & a_{2,2} + b_{2,2} & \dots & a_{2,n} + b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} + b_{m,1} & a_{m,2} + b_{m,2} & \dots & a_{m,n} + b_{m,n} \end{pmatrix}.$$

## Matriisin kertominen skalaarilla

Matriisin kertominen skalaarilla tapahtuu kertomalla kukin matriisin alkio skalaarilla. Olkoon  $\mathbf{A} \in \mathbb{R}^{m \times n}$  ja  $c \in \mathbb{R}$ . Tällöin

$$c\mathbf{A} = \begin{pmatrix} ca_{1,1} & ca_{1,2} & \dots & ca_{1,n} \\ ca_{2,1} & ca_{2,2} & \dots & ca_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ ca_{m,1} & ca_{m,2} & \dots & ca_{m,n} \end{pmatrix}.$$

## Matriisien vähennyslasku

Matriisien vähennyslasku määritellään tapahtuvaksi alkioittain. Olkoon  $\mathbf{A}$  ja  $\mathbf{B} \in \mathbb{R}^{m \times n}$ . Tällöin

$$\mathbf{A} - \mathbf{B} = \begin{pmatrix} a_{1,1} - b_{1,1} & a_{1,2} - b_{1,2} & \dots & a_{1,n} - b_{1,n} \\ a_{2,1} - b_{2,1} & a_{2,2} - b_{2,2} & \dots & a_{2,n} - b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} - b_{m,1} & a_{m,2} - b_{m,2} & \dots & a_{m,n} - b_{m,n} \end{pmatrix}.$$

## Harjoituksia MAT.2

- 1 Olkoon  $\mathbf{A}$  ja  $\mathbf{B} \in \mathbb{R}^{m \times n}$ . Osoita, että

$$\mathbf{A} - \mathbf{B} = \mathbf{A} + (-1)\mathbf{B} .$$

- 2 Olkoon  $\mathbf{A}$  ja  $\mathbf{B} \in \mathbb{R}^{m \times n}$ . Osoita, että

$$c(\mathbf{A} + \mathbf{B}) = (c\mathbf{A}) + (c\mathbf{B}) .$$

- 3 Olkoon  $\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$ ,  $\mathbf{B} = \begin{pmatrix} 3 & 2 \\ 1 & 3 \\ 1 & 2 \end{pmatrix}$  ja  $c = 4$ . Laske  $\mathbf{A} + c\mathbf{B}$ .

- 4 Toteuta `matrix_t`-luokkamalliin matriisien yhteen- ja vähennyslasku sekä skalaarilla kertominen.

## Matriisin ja vektorin välinen kertolasku

Olkoon  $\mathbf{A} \in \mathbb{R}^{m \times n}$  ja  $\mathbf{x} \in \mathbb{R}^n$ . Määritellään tällöin tulo

$$\mathbf{Ax} = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n a_{1,i}x_i \\ \sum_{i=1}^n a_{2,i}x_i \\ \vdots \\ \sum_{i=1}^n a_{m,i}x_i \end{pmatrix}.$$

Tulosvektorin alkio paikassa  $i$  on

$$(\mathbf{Ax})_i = \begin{pmatrix} a_{i,1} \\ a_{i,2} \\ \vdots \\ a_{i,n} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

Huomaa, että tässä määriteltiin  $\mathbf{Ax}$ , mutta ei  $\mathbf{x}\mathbf{A}$ !

## Matriisien kertolasku (1/2)

Edellä esitetty matriisin ja vektorin välinen tulo saadaan erikoistapauksena matriisien välisestä kertolaskusta.

Olkoon  $\mathbf{A} \in \mathbb{R}^{m \times n}$  ja  $\mathbf{B} \in \mathbb{R}^{n \times p}$ . Määritellään tällöin tulo

$$\begin{aligned}\mathbf{AB} &= \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} & \dots & b_{1,p} \\ b_{2,1} & b_{2,2} & \dots & b_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n,1} & b_{n,2} & \dots & b_{n,p} \end{pmatrix} \\ &= \begin{pmatrix} \sum_{i=1}^n a_{1,i} b_{i,1} & \sum_{i=1}^n a_{1,i} b_{i,2} & \dots & \sum_{i=1}^n a_{1,i} b_{i,p} \\ \sum_{i=1}^n a_{2,i} b_{i,1} & \sum_{i=1}^n a_{2,i} b_{i,2} & \dots & \sum_{i=1}^n a_{2,i} b_{i,p} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^n a_{m,i} b_{i,1} & \sum_{i=1}^n a_{m,i} b_{i,2} & \dots & \sum_{i=1}^n a_{m,i} b_{i,p} \end{pmatrix}.\end{aligned}$$

## Matriisien kertolasku (2/2)

Määritelmästä seuraa suoraan, että  $\mathbf{AB} \in \mathbb{R}^{m \times p}$ , eli tulomatriisin rivien lukumäärän määrää  $\mathbf{A}$ :n rivien lukumäärä ja sarakkeiden lukumäärän määrää  $\mathbf{B}$ :n sarakkeiden lukumäärä. Lisäksi on huomioitava, että  $\mathbf{A}$ :ssa on oltava yhtä monta saraketta kuin  $\mathbf{B}$ :ssä on rivejä.

Tulomatriisin riville  $i$ , sarakkeeseen  $j$  tulee kertolaskussa arvo

$$(\mathbf{AB})_{ij} = \begin{pmatrix} a_{i,1} \\ a_{i,2} \\ \vdots \\ a_{i,n} \end{pmatrix} \cdot \begin{pmatrix} b_{1,j} \\ b_{2,j} \\ \vdots \\ b_{n,j} \end{pmatrix} = \sum_{k=1}^n a_{i,k} b_{k,j} .$$

Huomaa, että yleensä ei päde  $\mathbf{AB} = \mathbf{BA}$ .

## Matriisin transpoosi

Olkoon  $\mathbf{A} \in \mathbb{R}^{m \times n}$ . Tällöin matriisin  $\mathbf{A}$  transpoosi

$$\mathbf{A}^T = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{pmatrix}^T = \begin{pmatrix} a_{1,1} & a_{2,1} & \dots & a_{m,1} \\ a_{1,2} & a_{2,2} & \dots & a_{m,2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1,n} & a_{2,n} & \dots & a_{m,n} \end{pmatrix} \in \mathbb{R}^{n \times m}$$

muodostetaan vaihtamalla matriisin rivit ja sarakkeet keskenään.

## Matriisilaskujen ominaisuuksia

Olkoon  $\mathbf{A}$ ,  $\mathbf{B}$  ja  $\mathbf{C}$  matriiseja ja  $k$  skalaari. Mikäli laskutoimitukset ovat määriteltyjä, niin

$$\mathbf{A} + \mathbf{0} = \mathbf{A},$$

$$\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A},$$

$$\mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C},$$

$$c(\mathbf{A} + \mathbf{B}) = c\mathbf{A} + c\mathbf{B},$$

$$(c\mathbf{A})\mathbf{B} = \mathbf{A}(c\mathbf{B}),$$

$$(c\mathbf{A})\mathbf{B} = c(\mathbf{AB}),$$

$$\mathbf{A}\mathbf{0} = \mathbf{0},$$

$$\mathbf{0}\mathbf{A} = \mathbf{0},$$

$$\mathbf{AI} = \mathbf{A},$$

$$\mathbf{IA} = \mathbf{A},$$

$$\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C},$$

$$\mathbf{A}(\mathbf{B} + \mathbf{C}) = (\mathbf{AB}) + (\mathbf{AC}),$$

$$\mathbf{A} = (\mathbf{A}^T)^T \text{ ja}$$

$$(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T.$$

## Harjoituksia MAT.3

- ➊ Muodosta matriisitulo  $\begin{pmatrix} 2 & 4 \\ 1 & 1 \\ 4 & 2 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}$ .
- ➋ Totea edellisen tehtävän matriiseilla, että kaava  $\mathbf{AB} = (\mathbf{B}^T \mathbf{A}^T)^T$  pätee.
- ➌ Olkoon  $\mathbf{A} \in \mathbb{R}^{3 \times 3}$ . Osoita kaava  $\mathbf{AI} = \mathbf{A}$  oikeaksi.
- ➍ Totea edellisen paikkansapitävyys tapauksessa  $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 5 \\ 4 & 2 & 3 \end{pmatrix} \mathbf{I}$
- ➎ Olkoon  $\mathbf{A}$  ja  $\mathbf{B} \in \mathbb{R}^{2 \times 2}$  sekä  $c \in \mathbb{R}$ . Osoita oikeaksi kaava  $(c\mathbf{A})\mathbf{B} = c(\mathbf{AB}) = \mathbf{A}(c\mathbf{B})$ .
- ➏ Olkoon  $\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 2 & 2 \end{pmatrix}$ ,  $\mathbf{B} = \begin{pmatrix} 2 & 4 \\ 1 & 7 \end{pmatrix}$  ja  $c = -1$ . Totea edellisen harjoituksen kaavan paikkansapitävyys.
- ➐ Lisää luokkamalliin `matrix_t` tuki matriisiin ja vektorin kertolaskulle. Huomioi oikeat dimensiot.
- ➑ Lisää luokkamalliin `matrix_t<class T>` tuki matriisien kertolaskulle. Huomioi oikeat dimensiot.
- ➒ Lisää luokkamalliin `matrix_t<class T>` tuki transpoosin muodostamiselle.

## Määritelmä

Olkoon  $\mathbf{A} \in \mathbb{R}^{n \times n}$ , siis neliömatriisi. Mikäli tällöin on olemassa matriisi  $\mathbf{A}^{-1}$  siten, että

$$\mathbf{AA}^{-1} = \mathbf{I},$$

niin  $\mathbf{A}^{-1}$  on tällöin  $\mathbf{A}$ :n käänteismatriisi.

Mikäli kyseinen matriisi on olemassa, niin

- pätee  $\mathbf{AA}^{-1} = \mathbf{I} = \mathbf{A}^{-1}\mathbf{A}$ ,
- $\mathbf{A}^{-1}$  on yksikäsitteinen sekä
- matriisin  $\mathbf{A}$  sanotaan olevan kääntyvä (tai säännöllinen).

HUOM: Käänteismatriisi voidaan määritellä myös matriisille, joka ei ole neliömatriisi, mutta tällöin edelliset tulokset eivät yleisesti pidä paikkaansa.

Käsitellään jatkossa vain neliömatriiseja, mikäli muuta ei mainita.

## Determinantti

Edellinen määritelmä ei antanut juurikaan työkaluja käänteismatriisiin muodostamiseksi, tai edes sen olemassaolon selvittämiseksi. Tähän ongelmaan ratkaisun tarjoaa matriisin determinantti.

Olkoon  $\mathbf{A} \in \mathbb{R}^{n \times n}$ . Määritellään  $\mathbf{A}$ :n determinantti rekursiivisesti

$$|\mathbf{A}| = \sum_{j=1}^n (-1)^{i+j} a_{ij} |\mathbf{A}^{\{ij\}}| ,$$

jossa indeksin  $i \in \{1, \dots, n\}$  voi kiinnittää mielivaltaisesti. Sanotaan, että determinantti on kehitetty rivin  $i$  suhteen. Lisäksi  $\mathbf{A}^{\{ij\}}$  on sellainen  $\mathbf{A}$ :n alimatriisi, josta on poistettu  $i$ :s rivi ja  $j$ :s sarake.

Määritellään samalla, että  $1 \times 1$  matriisin ( $a$ ) determinantti on  $a$ .

## Esimerkki $2 \times 2$ -matriisin determinantti

Olkoon  $\mathbf{A} \in \mathbb{R}^{2 \times 2}$ . Tällöin sen determinantti

$$\begin{aligned} |\mathbf{A}| &= \sum_{j=1}^n (-1)^{i+j} a_{ij} |\mathbf{A}^{\{ij\}}| \\ &= (-1)^{1+1} a_{1,1} a_{2,2} + (-1)^{1+2} a_{1,2} a_{2,1} \\ &= a_{1,1} a_{2,2} - a_{1,2} a_{2,1} \end{aligned}$$

Olkoon

$$\mathbf{A} = \begin{pmatrix} 3 & 5 \\ 1 & 2 \end{pmatrix}.$$

Tällöin

$$|\mathbf{A}| = 3 \cdot 2 - 1 \cdot 5 = 1.$$

## Esimerkki $3 \times 3$ -matriisin determinantti

Olkoon  $\mathbf{A} \in \mathbb{R}^{3 \times 3}$ . Tällöin sen determinantti

$$\begin{aligned} |\mathbf{A}| &= \sum_{j=1}^n (-1)^{i+j} a_{ij} |\mathbf{A}^{\{ij\}}| \\ &= a_{1,1} |\mathbf{A}^{\{1,1\}}| - a_{1,2} |\mathbf{A}^{\{1,2\}}| + a_{1,3} |\mathbf{A}^{\{1,3\}}| \\ &= a_{1,1}(a_{2,2}a_{3,3} - a_{3,2}a_{2,3}) - a_{1,2}(a_{2,1}a_{3,3} - a_{3,1}a_{2,3}) + a_{1,3}(a_{2,1}a_{3,2} - a_{3,1}a_{2,2}) \end{aligned}$$

## Determinantin ominaisuuksia

- $|\mathbf{A}| \neq 0 \Leftrightarrow \mathbf{A}$  on kääntyvä.
- Jos  $\mathbf{A}$  on kääntyvä, niin  $|\mathbf{A}^{-1}| = \frac{1}{|\mathbf{A}|}$ .
- Jos jokin  $\mathbf{A}$ :n sarake tai rivi koostuu pelkästään nollista, niin  $|\mathbf{A}| = 0$ .

$$|\mathbf{AB}| = |\mathbf{A}| |\mathbf{B}|$$

$$|\mathbf{A}^T| = |\mathbf{A}|$$

$$|\mathbf{I}| = 1$$

## Harjoituksia MAT.4

- ① Osoita  $2 \times 2$ -matriisille  $|\mathbf{AB}| = |\mathbf{A}| |\mathbf{B}|$ .
- ② Osoita  $2 \times 2$ -matriisille  $|\mathbf{A}^T| = |\mathbf{A}|$ .
- ③ Osoita  $3 \times 3$ -matriisille  $|\mathbf{I}| = 1$ .
- ④ Olkoon  $\mathbf{A}$  ja  $\mathbf{B} \in \mathbb{R}^{n \times n}$  ja käännyviä. Päteekö yleisesti  $\mathbf{A}^{-1}\mathbf{B}^{-1} = (\mathbf{BA})^{-1}$ ? Perustele väitteesi?
- ⑤ Olkoon  $\mathbf{A} \in \mathbb{R}^{n \times n}$  ja käännyvä. Osoita että  $(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T$ .
- ⑥  $2 \times 2$  determinantin laskemiseksi määritelmän mukaan tarvitaan 2 kertolaskua ja  $3 \times 3$  determinantin laskemiseksi tarvitaan vastaavasti  $2 \cdot 3 = 6$  kertolaskua. Arvioi determinantin määritelmän avulla, montako kertolaskua tarvitaan  $4 \times 4$  determinantin laskemiseksi.
- ⑦ Lisää luokkamalliin `matrix_t` matriisin determinantin laskenta. Muista tarkistaa, että kyseessä on neliömatriisi.

## Käänteismatriisin muodostaminen

Olkoon  $\mathbf{A} \in \mathbb{R}^{n \times n}$ . Määritellään sen alkion  $a_{ij}$  kofaktori

$$c_{ij} = (-1)^{i+j} |A^{\{ij\}}| .$$

Tällöin voidaan siis muodostaa matriisi

$$\mathbf{C} = \begin{pmatrix} c_{1,1} & c_{1,2} & \dots & c_{1,n} \\ c_{2,1} & c_{2,2} & \dots & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n,1} & c_{n,2} & \dots & c_{n,n} \end{pmatrix} .$$

Matriisin  $\mathbf{A}$  adjungoitunut matriisi  $\text{adj}\mathbf{A} = \mathbf{C}^T$ . Tämän avulla voidaan laskea käänteismatriisi

$$\mathbf{A}^{-1} = \frac{1}{|\mathbf{A}|} \text{adj}\mathbf{A} . \tag{9}$$

## Ortogonaalinen matriisi

Matriisin  $\mathbf{A} \in \mathbf{R}^{n \times n}$  sanotaan olevan ortogonaalinen jos

$$\mathbf{A}\mathbf{A}^T = \mathbf{I} = \mathbf{A}^T\mathbf{A}.$$

Mikäli matriisi tiedetään ortogonaaliseksi, on sen käänteismatriisi helppo laskea, koska

$$\mathbf{A}^{-1} = \mathbf{A}^T.$$

## Harjoituksia MAT.5

- ① Tarkasta, että  $2 \times 2$  yksikkömatriisin käänteismatriisi on todella se itse.
- ② Olkoon  $\mathbf{A} = \begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix}$ . Laske  $\mathbf{A}^{-1}$ .
- ③ Olkoon  $\mathbf{A}$ , kuten edellä ja  $\mathbf{B} = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}$ . Laske  $\mathbf{A}^{-1}\mathbf{B}^{-1}$  ja  $(\mathbf{BA})^{-1}$ .
- ④ Matriisi  $\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$  on ortogonaalinen. Laske  $\mathbf{A}^{-1}$ .
- ⑤ Onko yksikkömatriisi ortogonaalinen?
- ⑥ Lisää luokkamalliin `matrix` -käänteismatriisiin laskenta.

## Perusteet

Lineaarinen yhtälöryhmä on muotoa

$$\left\{ \begin{array}{l} a_{1,1}x_1 + a_{1,2}x_2 + \dots + a_{1,n}x_n = y_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \dots + a_{2,n}x_n = y_2 \\ \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \dots + a_{n,n}x_n = y_n , \end{array} \right.$$

joka voidaan esittää matriisien ja vektorien avulla muodossa

$$\mathbf{Ax} = \mathbf{y} , \tag{10}$$

jossa  $\mathbf{A} \in \mathbb{R}^{n \times n}$  ja  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ . Tässä vektori  $\mathbf{x}$  on tuntematon.

## Ratkaiseminen käänteismatriisin avulla

Lähdetään liikkeelle yhtälöstä (10)

$$\mathbf{A}\mathbf{x} = \mathbf{y} .$$

Oletetaan, että  $\mathbf{A}$  on käännyvä<sup>1</sup>, jolloin edellinen voidaan kertoa (vasemmalta)  $\mathbf{A}^{-1}$ :llä,

$$\begin{aligned}\mathbf{A}^{-1}\mathbf{A}\mathbf{x} &= \mathbf{A}^{-1}\mathbf{y} \\ \Rightarrow \mathbf{x} &= \mathbf{A}^{-1}\mathbf{y} .\end{aligned}$$

- Menetelmä on käytännössä erittäin raskas isoilla matriiseilla.

---

<sup>1</sup>Mikäli  $\mathbf{A}$  ei ole käännyvä, yhtälöryhmällä on joko  $\infty$  tai ei yhtään ratkaisua. Mikäli taas  $\mathbf{A}$  on käännyvä, niin yhtälöryhmällä on täsmälleen yksi ratkaisu.

## Määritelmä

Kuten jo matriisien yhteydessä määriteltiin, niin

- kuvaus  $\mathbf{A} : \mathbb{R}^n \rightarrow \mathbb{R}^m$  on lineaarikuvaus jos:
  - 1)  $c\mathbf{A}(x) = \mathbf{A}(cx)$  ja
  - 2)  $\mathbf{A}(x) + \mathbf{A}(y) = \mathbf{A}(x + y)$  ,jossa  $c \in \mathbb{R}$  ja  $x, y \in \mathbb{R}^n$ .
- $\mathbb{R}^n$  on kuvaksen  $\mathbf{A}$  lähtöjoukko ja
- $\mathbb{R}^m$  on kuvaksen  $\mathbf{A}$  maalijoukko.
- Kuvaus  $\mathbf{A}$  kuvailee yhden tai useamman lähtöjoukon alkion yhdeksi maalijoukon alkioksi.
- Kaikille maalijoukon alkioille ei aina kuvaudu yhtään lähtöjoukon alkiota.

## Origon kuvautuminen

Olkoon  $\mathbf{A} : \mathbb{R}^n \rightarrow \mathbb{R}^m$  lineaarikuvaus. Kiinnitetään skalaari  $c = 0$  ja mielivaltainen  $\mathbf{x} \in \mathbb{R}^n$ . Merkitään  $\mathbf{y} = \mathbf{A}(\mathbf{x}) \in \mathbb{R}^m$ . Nyt voidaan kirjoittaa

$$\begin{aligned}\mathbf{A}(\mathbf{0}) &= \mathbf{A}(c\mathbf{x}) \\ &= c\mathbf{A}(\mathbf{x}) \\ &= c\mathbf{y} \\ &= 0\mathbf{y} \\ &= \mathbf{0} \in \mathbb{R}^m.\end{aligned}$$

Edellisen perusteella lineaarikuvaussa lähtöjoukon origo kuvautuu aina maalijoukon origoksi.

## Onko $f$ lineaarikuvaus? Esimerkki 1

Onko  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$  lineaarikuvaus kun  $f(\mathbf{x}) = x_1 + x_2$  ?

Kiinnitetään  $c \in \mathbb{R}$  sekä  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$ . Tällöin

$$cf(\mathbf{x}) = c(x_1 + x_2) = (cx_1) + (cx_2) = f((cx_1, cx_2)^T) = f(c(x_1, x_2)^T) = f(c\mathbf{x}),$$

eli kuvaus toteuttaa määritelmän 1. ehdon.

$$f(\mathbf{x} + \mathbf{y}) = x_1 + y_1 + x_2 + y_2 = \underbrace{x_1 + x_2}_{=f(\mathbf{x})} + \underbrace{y_1 + y_2}_{=f(\mathbf{y})} = f(\mathbf{x}) + f(\mathbf{y}) ,$$

joten kuvaus toteuttaa myös määritelmän 2. ehdon ja on siten lineaarikuvaus.

## Onko $f$ lineaarikuvaus? Esimerkki 2

Onko  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$  lineaarikuvaus kun  $f(\mathbf{x}) = x_1 x_2$  ?

Valitaan  $\mathbf{x} = (0, 1)$  ja  $\mathbf{y} = (1, 0)$ . Tällöin

$$f(\mathbf{x}) + f(\mathbf{y}) = (0 \cdot 1) + (1 \cdot 0) = 0 ,$$

mutta

$$f(\mathbf{x} + \mathbf{y}) = f((1, 1)^T) = 1 \cdot 1 = 1 ,$$

joten kuvaus ei täytä lineaarikuvauskelta vaadittavaa 2. ehtoa ja ei siten ole lineaarikuvaus.

## Harjoituksia LIN.1

- ① Onko  $f : \mathbb{R} \rightarrow \mathbb{R}$  lineaarikuvaus kun  $f(x) = 0$  ?
- ② Onko  $f : \mathbb{R} \rightarrow \mathbb{R}$  lineaarikuvaus kun  $f(x) = x^2$  ?
- ③ Onko  $f : \mathbb{R} \rightarrow \mathbb{R}$  lineaarikuvaus kun  $f(x) = x + 1$  ?
- ④ Onko  $f : \mathbb{R} \rightarrow \mathbb{R}$  lineaarikuvaus kun  $f(x) = 2x$  ?
- ⑤ Onko  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$  lineaarikuvaus kun  $f(\mathbf{x}) = x_1$  ?
- ⑥ Onko  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$  lineaarikuvaus kun  $f(\mathbf{x}) = \|\mathbf{x}\|$  ?
- ⑦ Onko  $f : \mathbb{R} \rightarrow \mathbb{R}^2$  lineaarikuvaus kun  $f(x) = (x, x)^T$  ?
- ⑧ Onko  $f : \mathbb{R} \rightarrow \mathbb{R}^2$  lineaarikuvaus kun  $f(x) = (x, x^2)^T$  ?
- ⑨ Onko  $f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  lineaarikuvaus kun  $f(\mathbf{x}) = (x_2, x_1)^T$  ?

Osoita määritelmän mukaan lineaarikuvaukseen ominaisuudet tai esitä vastaesimerkki.

## Avaruuden kantavekttorit

Avaruuden  $\mathbb{R}^n$  kantavektoreiksi kutsutaan koordinaattiakseleiden suuntaisia yksikkövektoreita  $\mathbf{e}_1 = (1 \ 0 \ 0 \ \dots \ 0)^T$ ,  $\mathbf{e}_2 = (0 \ 1 \ 0 \ \dots \ 0)^T$ ,  $\dots$ ,  $\mathbf{e}_n = (0 \ 0 \ \dots \ 0 \ 1)^T$ .

Mikä tahansa  $\mathbb{R}^n$ :n vektori  $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$  voidaan esittää kantavektoreiden painotettuna summana

$$\mathbf{x} = x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + \dots + x_n\mathbf{e}_n .$$

## Lineaarikuvauksen ja matriisin yhteys (1/3)

Olkoon annettu lineaarikuvaus  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ . Tällöin

$$\begin{aligned} f(\mathbf{x}) &= f(x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + \cdots + x_n\mathbf{e}_n) \\ &= f(x_1\mathbf{e}_1) + f(x_2\mathbf{e}_2) + \cdots + f(x_n\mathbf{e}_n) \\ &= x_1 \underbrace{f(\mathbf{e}_1)}_{\text{merk. } = \mathbf{f}_1} + x_2 \underbrace{f(\mathbf{e}_2)}_{\text{merk. } = \mathbf{f}_2} + \cdots + x_n \underbrace{f(\mathbf{e}_n)}_{\text{merk. } = \mathbf{f}_n}, \end{aligned}$$

joten lineaarikuvaus on määritetty yksikäsitteisesti, kun tunnetaan kuinka se kuvaa kantavektorit.

## Lineaarikuvauksen ja matriisin yhteys (2/3)

Lasketaan edelleen

$$\begin{aligned}
 f(\mathbf{x}) &= x_1 \mathbf{f}_1 + x_2 \mathbf{f}_2 + \cdots + x_n \mathbf{f}_n \\
 &= x_1(f_{1,1} \ f_{2,1} \ \dots \ f_{m,1})^T + x_2(f_{1,2} \ f_{2,2} \ \dots \ f_{m,2})^T + \cdots + x_n(f_{1,n} \ f_{2,n} \ \dots \ f_{m,n})^T \\
 &= \begin{pmatrix} x_1 f_{1,1} + x_2 f_{1,2} + \cdots + x_n f_{1,n} \\ x_1 f_{2,1} + x_2 f_{2,2} + \cdots + x_n f_{2,n} \\ \vdots \\ x_1 f_{m,1} + x_2 f_{m,2} + \cdots + x_n f_{m,n} \end{pmatrix} \\
 &= \mathbf{F}\mathbf{x} ,
 \end{aligned}$$

jossa

$$\mathbf{F} = \begin{pmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,n} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ f_{m,1} & f_{m,2} & \cdots & f_{m,n} \end{pmatrix} .$$

## Lineaarikuvaukseen ja matriisin yhteys (3/3)

Edellisen perusteella siis kaikki lineaarikuvaukset  $\mathbb{R}^n \rightarrow \mathbb{R}^m$  voidaan esittää matriiseina.

Toisaalta matriisin ja vektorin tulon ominaisuuksien perusteella tiedämme, että matriisin kertominen vektorilla toteuttaa lineaarikuvauksen ominaisuudet.

Tämän vuoksi lineaarikuvaukset  $\mathbb{R}^n \rightarrow \mathbb{R}^m$  ja matriisit  $\mathbb{R}^{n \times m}$  voidaan samaistaa.

## Lineaarikuvausten yhdistäminen

Olkoon annettu lineaarikuvauksen  $A : \mathbb{R}^n \rightarrow \mathbb{R}^m$  ja  $B : \mathbb{R}^m \rightarrow \mathbb{R}^l$ , sekä mielivaltainen  $\mathbf{x} \in \mathbb{R}^n$ . Olkoon lisäksi  $\mathbf{A}$  ja  $\mathbf{B}$  annettuja lineaarikuvauksia vastaavat matriisit.

Tällöin

$$B(A(\mathbf{x})) = \mathbf{B}\mathbf{A}\mathbf{x} .$$

## Harjoituksia LIN.2

Laske

- ① lineaarikuvausta  $A : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  vastaava matriisi, kun  $A(\mathbf{x}) = (x_1, x_2)^T$ .
- ② lineaarikuvausta  $B : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  vastaava matriisi, kun  $B(\mathbf{x}) = (2x_1, x_1 + x_2)^T$ .
- ③ lineaarikuvausta  $C : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  vastaava matriisi, kun  $C(\mathbf{x}) = (x_2, x_1)^T$ .
- ④ lineaarikuvausta  $D : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  vastaava matriisi, kun  $D(\mathbf{x}) = (x_2 + x_1, x_2 - x_1)^T$ .
- ⑤ yhdistetty kuvaus  $B(D(\mathbf{x}))$  ja sitä vastaava matriisi.
- ⑥ edellisten tehtävien kuvausten arvo kun  $\mathbf{x} = (2, -1)^T$ .

## Nettilähteitä

- <https://en.wikipedia.org/wiki/Quaternion>
- [https://en.wikipedia.org/wiki/Quaternions\\_and\\_spatial\\_rotation](https://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation)
- <http://www.cprogramming.com/tutorial/3d/quaternions.html>
- <http://www.euclideanspace.com/mathematics/algebra/realNormedAlgebra/quaternions>

## Määritelmä

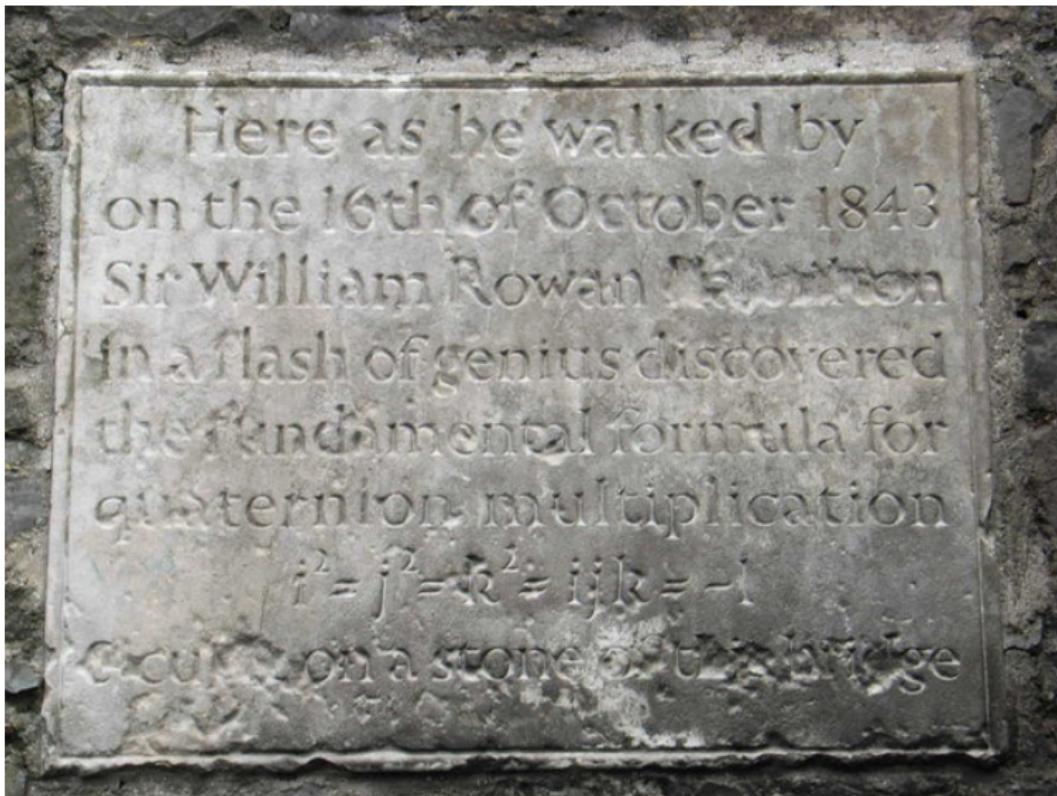
- Kvaterniot  $\mathbb{H} = \mathbb{R}^4$  varustettuna yhteenlaskulla, skalarilla kertomisella sekä kvaternioiden välisellä kertolaskulla.
- Määritellään kanta  $\bar{1} = (1 \ 0 \ 0 \ 0)^T$ ,  $i = (0 \ 1 \ 0 \ 0)^T$ ,  
 $j = (0 \ 0 \ 1 \ 0)^T$  ja  $k = (0 \ 0 \ 0 \ 1)^T$ .
- Kvaternionio  $q = (a \ b \ c \ d)^T \in \mathbb{H}$  voidaan kirjoittaa muodossa  
 $q = a\bar{1} + bi + cj + dk = a + bi + cj + dk$ .

## Yhteenlasku

- Olkoon  $q_1, q_2 \in \mathbb{H}$ .
- Kahden kvaternion *summa*

$$\begin{aligned} q_1 + q_2 &= (a_1 + b_1 i + c_1 j + d_1 k) + (a_2 + b_2 i + c_2 j + d_2 k) \\ &= (a_1 + a_2) + (b_1 + b_2)i + (c_1 + c_2)j + (d_1 + d_2)k \end{aligned}$$

## Kannan elementtien tulo



## Kannan elementtien tulo

- Identiteetti  $i^2 = j^2 = k^2 = ijk = -1$  määrittelee tulon.
- Esimerkiksi  $-k = (ijk)k = ijk^2 = ij(-1)$ .
- Muut tapaukset:

$$ij = k \qquad ji = -k$$

$$jk = i \qquad kj = -i$$

$$ki = j \qquad ik = -j$$

## Tulo

- Olkoon  $q_1, q_2 \in \mathbb{H}$ .
- Kahden kvaternion *tulo*

$$\begin{aligned} q_1 q_2 &= (a_1 + b_1 i + c_1 j + d_1 k)(a_2 + b_2 i + c_2 j + d_2 k) \\ &= (a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2) \\ &\quad + (a_1 b_2 + b_1 a_2 + c_1 d_2 - d_1 c_2)i \\ &\quad + (a_1 c_2 - b_1 d_2 + c_1 a_2 + d_1 b_2)j \\ &\quad + (a_1 d_2 + b_1 c_2 - c_1 b_2 + d_1 a_2)k \end{aligned}$$

- Tämä määritelmä on suora seuraus kannan alkioiden tulon määritelmästä.

## Kvaternioiden ominaisuuksia

Algebrallisesti kvaterniot muodostavat *vinokunnan*<sup>2</sup>: Olkoon  $x, y, z \in \mathbb{H}$ . Tällöin

- ①  $x + (y + z) = (x + y) + z$ ,
- ② on olemassa  $0 \in \mathbb{H}$  s.e kaikilla  $x$  pätee  $x + 0 = x$ ,
- ③ kaikille  $x$  on olemassa  $(-x)$  s.e  $x + (-x) = 0$ ,
- ④  $x + y = y + x$ ,
- ⑤  $x(y + z) = xy + xz$ ,
- ⑥  $x(yz) = (xy)z$ ,
- ⑦ on olemassa  $1$  s.e kaikille  $x$  pätee  $1x = x$  ja
- ⑧ kaikille  $x \neq 0$  on olemassa  $x^{-1}$  s.e  $xx^{-1} = 1$ .

Huomaa että yleisesti ei päde

- $xy = yx$ , sillä esimerkiksi  $ij = k \neq -k = ji$ .

---

<sup>2</sup>Käytetään myös nimistä *jakorengas*

## Kvaternioiden ominaisuuksia

Koska kvaterniot muodostavat vinokunnan, pätee niille:

- 0 ja 1 ovat yksikäsiteisiä.
- $x + y = x + z \Rightarrow y = z$ .
- Jos  $x \neq 0$ , niin  $xy = xz \Rightarrow y = z$ .
- $-x = x$ .

Lisäksi tulon määritelmästä seuraa:

- $x1 = x$ .
- $x^{-1}x = xx^{-1} = 1$ .

Huomaa että yleisesti ei päde  $xy^{-1} = y^{-1}x$ , joten ei pidä kirjoittaa  $\frac{x}{y}$ .

## Jakaminen skalaari- ja vektoriosaan

- Kvaternion  $q = (a, b, c, d) \in \mathbb{H}$  voidaan jakaa *reaaliosaan*  $a \in \mathbb{R}$  ja *vektoriosaan*  $\mathbf{v} = (b \quad c \quad d)^T \in \mathbb{R}^3$ .
- Tällöin yhteenlasku  $(a_1 \quad \mathbf{v}_1)(a_2 \quad \mathbf{v}_2) = (a_1 + a_2 \quad \mathbf{v}_1 + \mathbf{v}_2)$ , ja
- kertolasku  $(a_1 \quad \mathbf{v}_1)(a_2 \quad \mathbf{v}_2) = (a_1 a_2 - \mathbf{v}_1 \cdot \mathbf{v}_2 \quad a_1 \mathbf{v}_2 + a_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2)$ .

## Konjugaatti, normi ja käänteisalkio

- Olkoon  $q = (a \quad \mathbf{v})$ . Tällöin  $q^* = (a \quad -\mathbf{v})$  on  $q$ :n *konjugaatti*.
- *Normi*  $\|q\| = \sqrt{q^* q} = \sqrt{a^2 + \|\mathbf{v}\|^2} = \sqrt{a^2 + b^2 + c^2 + d^2}$ .
- Olkoon  $q \neq 0$ . Tällöin käänteisalkio

$$q^{-1} = \frac{q^*}{qq^*} = \frac{q^*}{a^2 + b^2 + c^2 + d^2} .$$

- Siis jos  $\|q\| = 1$ , niin  $q^{-1} = q^*$ .

## Polaarimuoto ja kvaternion potenssi

- Kvaternionille voidaan johtaa yleistetty Eulerin kaava

$$e^{\hat{\mathbf{n}}\theta} = (\cos \theta \quad \hat{\mathbf{n}} \sin \theta) \in \mathbb{H}.$$

- Olkoon  $q = (a \quad \mathbf{v}) \in \mathbb{H}$ .
- *Polaarimuoto*

$$q = \|q\| e^{\hat{\mathbf{n}}\theta},$$

jossa

$$\theta = \arccos \frac{a}{\|q\|}, \text{ ja } \hat{\mathbf{n}} = \frac{1}{\|\mathbf{v}\|} \mathbf{v}$$

- *Kvaternion potenssi*

$$q^\alpha = \|q\|^\alpha e^{\hat{\mathbf{n}}\theta\alpha}$$

## Motivaatio (1/2)

Tavoitteena on määritellä järjestelmä, jonka avulla pisteitä voidaan siirtää koordinaatistosta toiseen ilman, että on tarvetta laskea välivaiheita eri koordinaatistoissa.

Pyritään määrittelemään edellisenkaltainen järjestelmä lineaarikuvausten avulla siten, että koordinaatistomuunnokset eri koordinaatistojen välillä voidaan kuvata yhdistetyillä lineaarikuvauskilla siten, että kaikille pisteille voidaan käyttää samaa yhdistettyä lineaarikuvausta.

Koska lineaarikuvaukset samaistettiin matriisien kanssa voidaan edellinen toteuttaa matriisien avulla<sup>3</sup> ja yhdistetyt lineaarikuvaukset voidaan muuntaa matriisien kertolaskuksi.

$$\begin{aligned} \mathbf{A}(\mathbf{B}(\mathbf{C}(x))) &= \underbrace{\mathbf{ABC}}_Q \mathbf{x} \\ \mathbf{A}(\mathbf{B}(\mathbf{C}(y))) &= Q\mathbf{y} \end{aligned}$$

---

<sup>3</sup>Edellyttää tietenkin että se voidaan toteuttaa lineaarikuvausten avulla.

## Motivaatio (2/2)

Ongelma: Lineaarikuvaukset kuvaavat origon origoksi  $\Rightarrow$  Siirtoa ei voi kuvata lineaarikuvauksella.

Ratkaisu: Valitaan käytettävä avaruus siten, että origo ei kuulu kyseiseen avaruuteen.

- ① Upotetaan  $\mathbb{R}^2$   $\mathbb{R}^3$ :een siten, että asetetaan kolmas koordinaatti aina arvoon 1.
- ② Upotetaan  $\mathbb{R}^3$   $\mathbb{R}^4$ :ään siten, että asetetaan neljäs koordinaatti aina arvoon 1.

Tällä "tempulla" voidaan myös siirrot kuvata lineaarikuvauksella.

Esimerkkejä  $\mathbb{R}^2$ :ssa

$$\begin{aligned}(0, 0) &\sim (0, 0, 1)^T \\ (x_1 \ x_2) &\sim (x_1 \ x_2 \ 1)^T\end{aligned}$$

Esimerkkejä  $\mathbb{R}^3$ :ssa

$$\begin{aligned}(0 \ 0 \ 0) &\sim (0 \ 0 \ 0 \ 1)^T \\ (x_1 \ x_2 \ x_3) &\sim (x_1 \ x_2 \ x_3 \ 1)^T\end{aligned}$$

## Määritelmä

- Edellä käytettiin vain homogenisia koordinaatteja, joissa "ylimääräinen" koordinaatti oli aina 1.
- Laajennetaan avaruutta s.e ylimääräinen koordinaatti  $w$  voi olla mikä tahansa reaaliluku  $w \neq 0$ .
- Samaistetaan homogeninen koordinaatti

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ w \end{pmatrix} \sim \begin{pmatrix} x_1/w \\ x_2/w \\ \vdots \\ x_n/w \end{pmatrix}, \quad (11)$$

jossa  $w \neq 0$ .<sup>4</sup>

- [https://upload.wikimedia.org/wikipedia/commons/3/34/Affine\\_transformations.ogv](https://upload.wikimedia.org/wikipedia/commons/3/34/Affine_transformations.ogv)
- Tätä hyödynnetään jatkossa perspektiivimuunnoksen yhteydessä.

---

<sup>4</sup>Tarkkaan ottaen olisi samaistettava  $(x_1 \ x_2 \ \dots \ x_n)$  ja  $\mathbb{R}^{n+1}$  suora  
 $\{(wx_1 \ wx_2 \ \dots \ wx_n \ w) : w \in \mathbb{R}\}$ .

Siirto  $\mathbb{R}^2$ :ssa

Määritellään siirtomatriisi

$$\mathbf{T}(\Delta x_1, \Delta x_2) = \begin{pmatrix} 1 & 0 & \Delta x_1 \\ 0 & 1 & \Delta x_2 \\ 0 & 0 & 1 \end{pmatrix}.$$

Matriisi riippuu siis kahdesta parametrista  $\Delta x_1$  ja  $\Delta x_2$ .

## Esimerkki

Siirretään pistettä  $(2 \ 0)$  vektorin  $(1 \ 2)$  verran.

Käytetään homogeenisia koordinaatteja:

$$(2 \ 0) \sim (2 \ 0 \ 1)^T .$$

Siirtomatriisi  $T(1, 2) = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}$ . Lasketaan siirron tulos

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \cdot 1 + 0 \cdot 0 + 1 \cdot 1 \\ 2 \cdot 0 + 0 \cdot 1 + 1 \cdot 2 \\ 2 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 \end{pmatrix} .$$

Siirretty koordinaatti on siis  $(3 \ 2)$ .

## Siirtomatriisin käänteismatriisi $\mathbb{R}^2$ :ssa

Olkoon annettu siirtomatriisi  $\mathbf{T}(\Delta x_1, \Delta x_2) \in \mathbb{R}^{n \times n}$ . Tällöin sen käänteismatriisi

$$\mathbf{T}(\Delta x_1, \Delta x_2)^{-1} = \begin{pmatrix} 1 & 0 & -\Delta x_1 \\ 0 & 1 & -\Delta x_2 \\ 0 & 0 & 1 \end{pmatrix}.$$

Todistus:

$$\begin{aligned} \mathbf{T}(\Delta x_1, \Delta x_2)\mathbf{T}(\Delta x_1, \Delta x_2)^{-1} &= \begin{pmatrix} 1 & 0 & \Delta x_1 \\ 0 & 1 & \Delta x_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -\Delta x_1 \\ 0 & 1 & -\Delta x_2 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & -\Delta x_1 + \Delta x_1 \\ 0 & 1 & -\Delta x_2 + \Delta x_2 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \mathbf{I}. \end{aligned}$$

## Harjoituksia HOMOG.1

- ① Olkoon  $\mathbf{x} = (3 \ -1 \ 1)^T$ . Laske  $\mathbf{T}(2, 2)\mathbf{x}$ .
- ② Olkoon  $\mathbf{x}$  kuten edellä. Laske  $\mathbf{T}(1, -2)\mathbf{T}(1, 2)\mathbf{x}$ .
- ③ Olkoon  $\mathbf{x}$  kuten edellä. Etsi matriisi  $\mathbf{A} \in \mathbb{R}^{3 \times 3}$ , jolle  $\mathbf{A}\mathbf{x} = (0 \ 0, \ )^T$ . VIHJE: Etsi sopiva siirtomatriisi ja tarkasta että se toteuttaa annetun yhtälön.

Kierto  $\mathbb{R}^2$ :ssa

Vektoria kierretään origon ympäri kulman  $\theta$  verran kiertomatriisilla

$$\mathbf{R}(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Kiertomatriisi  $\mathbf{R}(\theta)$  on ortogonaalinen, joten sen käänteismatriisi on  $\mathbf{R}(\theta)^T$ .

## Esimerkki – Yhdistetty siirto ja kierto

Siirretään pistettä  $(2 \ 0)$  vektorin  $(1 \ 2)$  verran ja kierretään sen jälkeen  $\frac{\pi}{2}$  radiaania.

Käytetään homogeenisia koordinaatteja:

$$(2 \ 0) \sim (2 \ 0 \ 1)^T .$$

$$\text{Siirtomatriisi } T(1, 2) = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix} \text{ ja kiertomatriisi } R\left(\frac{\pi}{2}\right) = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Lasketaan kokonaismuunnosmatriisi

$$\begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & -1 & -2 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} .$$

Muunnettu homogeeninen koordinaatti on siten

$$\begin{pmatrix} 0 & -1 & -2 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \cdot 0 + 0 \cdot -1 + 1 \cdot -2 \\ 2 \cdot 1 + 0 \cdot 0 + 1 \cdot 1 \\ 2 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 \end{pmatrix} = \begin{pmatrix} -2 \\ 3 \\ 1 \end{pmatrix} .$$

Skaalaus  $\mathbb{R}^2$ :ssa

Määritellään skaalausmatriisi

$$\mathbf{S}(k_1, k_2) = \begin{pmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Jossa  $k_1$  on skaalaus  $x_1$ -akselin suunnassa ja  $k_2$   $x_2$ -akselin suunnassa. Mikäli skaalaus on 0 jonkin akselin suuntaan, saadaan kaikki pisteet projisoitua toiselle koordinaattiakselille. Mikäli skaalauskerroin on negatiivinen, saadaan aikaan peilaus toisen koordinaattiakselin suhteen.

## Harjoituksia HOMOG.2

- ① Laske edellisen esimerkin tehtävä siten, että suoritat ensin kierron ja sitten siirron.
- ② Olkoon  $\mathbf{x} = (1 \ -4 \ 1)^T$ . Laske  $\mathbf{R}(\frac{\pi}{3})\mathbf{T}(-1, 4)\mathbf{x}$ .
- ③ Olkoon  $\mathbf{x}$  kuten edellä. Laske  $\mathbf{T}(-1, 4)\mathbf{R}(\frac{\pi}{3})\mathbf{x}$ .
- ④ Osoita, että kiertomatriisi on ortogonaalinen. VIHJE: Laske että  $\mathbf{R}(\theta)\mathbf{R}(\theta)^T = \mathbf{I}$ .
- ⑤ Onko skaalausmatriisilla käänteismatriisia?
- ⑥ Peri matriisi- ja vektoriluokkamallista 2d-homogenisten koordinaattien käsittelyyn omat luokkansa ja toteuta matriisiin kierto, siirto ja skaalaus.

Siirto  $\mathbb{R}^3$ :ssa

Määritellään siirtomatriisi

$$\mathbf{T}(\Delta x_1, \Delta x_2, \Delta x_3) = \begin{pmatrix} 1 & 0 & 0 & \Delta x_1 \\ 0 & 1 & 0 & \Delta x_2 \\ 0 & 0 & 1 & \Delta x_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Vastaavasti sen käänteismatriisi

$$\mathbf{T}(\Delta x_1, \Delta x_2, \Delta x_3)^{-1} = \begin{pmatrix} 1 & 0 & 0 & -\Delta x_1 \\ 0 & 1 & 0 & -\Delta x_2 \\ 0 & 0 & 1 & -\Delta x_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Kierto  $\mathbb{R}^3$ :ssa

Kierto on  $\mathbb{R}^3$ :ssa monimutkaisempi kuin  $\mathbb{R}^2$ :ssa, koska kierron akseli ei ole rajoittunut yhteen mahdolliseen suuntaan. Itse asiassa  $\mathbb{R}^3$ :ssa mahdollisia suuntia on  $\infty$  monta.

Vektoria kierretään  $x_1$ -akselin ympäri ympäri kulman  $\theta$  verran kiertomatriisilla

$$\mathbf{R}_1(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Kiertomatriisi  $\mathbf{R}_1(\theta)$  on ortogonaalinen, joten sen käänneismatriisi on  $\mathbf{R}_1(\theta)^T$ .

Muut kierrot  $\mathbb{R}^3$ :ssa

Kierto  $x_2$ -akselin ympäri saadaan toteutettua matriisilla

$$\mathbf{R}_2(\theta) = \begin{pmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

ja  $x_3$ -akselin ympäri matriisilla

$$\mathbf{R}_3(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Myös nämä kiertomatriisit ovat ortogonaalisia.

Skaalaus  $\mathbb{R}^3$ :ssa

Määritellään skaalausmatriisi

$$\mathbf{S}(k_1, k_2, k_3) = \begin{pmatrix} k_1 & 0 & 0 & 0 \\ 0 & k_2 & 0 & 0 \\ 0 & 0 & k_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Jossa  $k_1$  on skaalaus  $x_1$ -akselin,  $k_2$   $x_2$ -akselin ja  $k_3$   $x_3$ -akselin suunnassa.

## Harjoituksia HOMOG.3

- ① Piirrä tikku-ukko ruutupaperille siten, että sen vasemman jalan kärki on koordinaatiston origossa ja korkeus on 10 ruutua. Merkitse nivelten ja pään keskipisteen koordinaatit ruutupaperille. Laske seuraavien operaatioiden yhteinen muunnosmatriisi:
  - Skaalaa tikku-ukko 3 ruutua korkeaksi säilyttäen mittasuhteet.
  - Siirrä tikku-ukkoa  $x_1$ -suunnassa 4 ruutua vasemmalle ja  $x_2$ -suunnassa 3 ruutua ylös.
  - Kierrä ukkoa vastapäivään  $45^\circ$ , eli  $\frac{\pi}{4}$  radiaania.
- ② Piirrä edellisen tehtävän tikku-ukko muunnetuin koordinaatein. Laske muunnetut koordinaatit esimerkiksi käyttäen matriisi-luokkaa tai jotain matriisien laskennan osaavaa ohjelmistoa.
- ③ Peri matriisi- ja vektoriluokkamallista 3d-homogenisten koordinaattien käsitteilyyn omat luokkansa ja toteuta matriisiin kierrot, siirto ja skaalaus.

## Kierto mielivaltaisen akselin ympäri

Kierto mielivaltaisen yksikkövektorin  $\hat{\mathbf{n}}$  ympäri saadaan matriisilla  $\mathbf{R}_{\hat{\mathbf{n}}}(\theta) =$

$$\begin{pmatrix} n_1^2(1 - \cos \theta) + \cos \theta & n_1 n_2(1 - \cos \theta) - n_3 \sin \theta & n_1 n_3(1 - \cos \theta) + n_2 \sin \theta & 0 \\ n_1 n_2(1 - \cos \theta) + n_3 \sin \theta & n_2^2(1 - \cos \theta) + \cos \theta & n_2 n_3(1 - \cos \theta) - n_1 \sin \theta & 0 \\ n_1 n_3(1 - \cos \theta) - n_2 \sin \theta & n_2 n_3(1 - \cos \theta) + n_1 \sin \theta & n_3^2(1 - \cos \theta) + \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Tässä siis kiertoakseli  $\hat{\mathbf{n}} = (n_1 \ n_2 \ n_3)^T$  ja  $n_1^2 + n_2^2 + n_3^2 = 1$ . Kiertosuunta on oikeakätisessä koordinaatistossa vastapäivään.

## Aito kvaternio (pure quaternion)

- *Aito kvaternio* on kvaternio jonka reaaliosa on 0.
- Aito kvaterio on siis muotoa  $(0 \quad \mathbf{v})$ .
- $\mathbb{R}^3$  vektorit ja aidot kvaterniot voidaan samaistaa.
- Tätä hyödyntäen voidaan  $\mathbb{R}^3$  vektoreita kiertää käyttäen kvaternioiden kertolaskua.

## Kierto kvaternioiden avulla

- Olkoon  $v = (0 \quad \mathbf{v}) \in \mathbb{H}$ , jossa  $\mathbf{v} \in \mathbb{R}^3$  on kierrettävä vektori.
- Olkoon lisäksi  $q = (\cos \frac{\phi}{2} \quad \sin \frac{\phi}{2} \hat{\mathbf{u}})$ , jossa  $\|\hat{\mathbf{u}}\| = 1$ .
- Kierretty kvaternionio  $v' = (0 \quad \mathbf{v}') = qvq^{-1}$ ,
- jossa kierretty vektori  $\mathbf{v}' = \mathbf{R}(-\phi, \hat{\mathbf{u}})\mathbf{v}$ .

## Skaalaus mielivaltaisessa suunnassa

Skaalaus mielivaltaisen yksikkövektorin  $\hat{\mathbf{n}}$  suunnassa saadaan matriisilla  $\mathbf{S}_{\hat{\mathbf{n}}}(k) =$

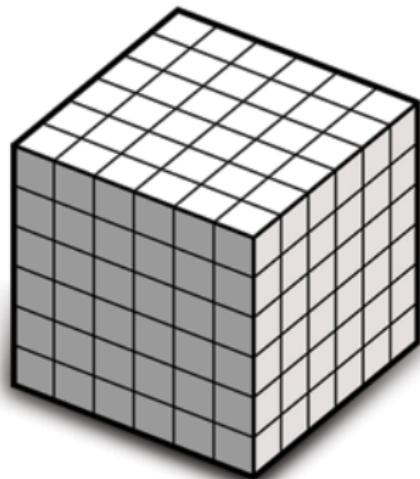
$$\begin{pmatrix} 1 + (k-1)n_1^2 & (k-1)n_1n_2 & (k-1)n_1n_3 & 0 \\ (k-1)n_1n_2 & 1 + (k-1)n_2^2 & (k-1)n_2n_3 & 0 \\ (k-1)n_1n_3 & (k-1)n_2n_3 & 1 + (k-1)n_3^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Tässä siis skaalaussuunta  $\hat{\mathbf{n}} = (n_1 \ n_2 \ n_3)^T$  ja  $n_1^2 + n_2^2 + n_3^2 = 1$ .

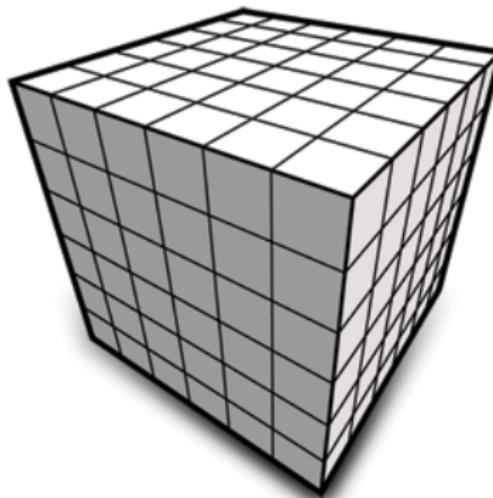
## Yleistä

- Tarkastellaan seuraavaksi kuinka 3d-maailmankoordinaateissa olevat kappaleet projisoidaan 2d-tasolle (näytölle).
- Erilaisia projektioita:
  - Perspektiiviprojektio – Realistinen vaikutelma
  - Isometrinen projektio – Aksonometrinen, säilyttää samansuuntaisuuden
  - Vinoprojektiot – Cabinet, Cavalier jne.
  - Ja paljon muita...

## Isometrisen vs. perspektiiviprojektio

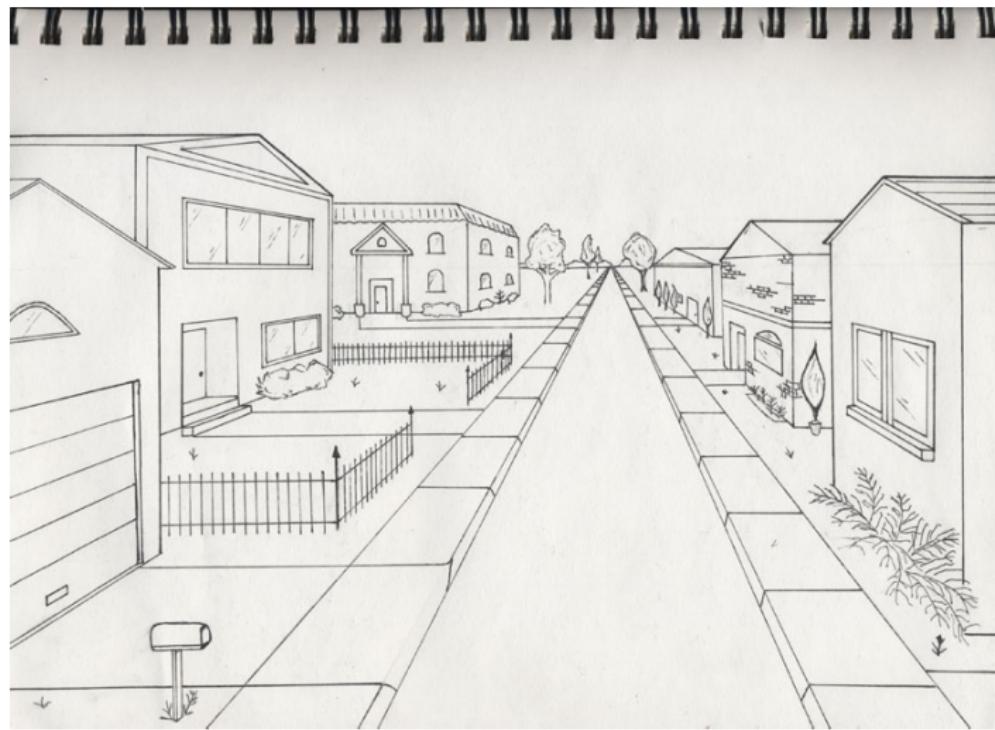


**isometric projection**



**perspective projection**

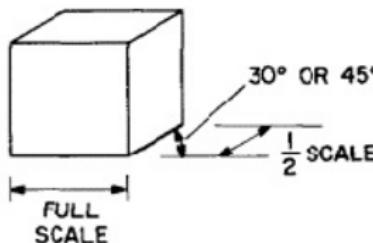
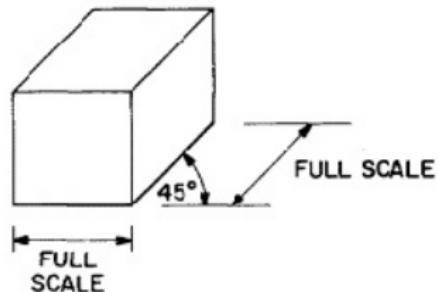
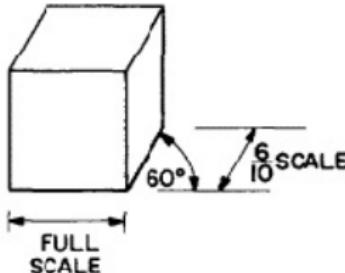
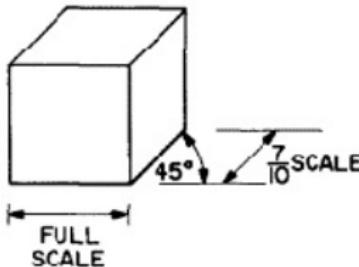
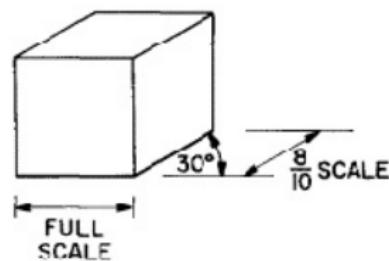
## Perspektiiviprojektilo



## Isometrisen projektio



# Vinoprojektiot



CAVALIER DRAWING

CABINET DRAWING

## 3d-maailmasta 2d-tasokuvaksi

Yleinen periaate perspektiivi- ja isometriselle projektiolle:

- ① Kiinnitä kameran ominaisuudet:
  - Paikka, orientaatio
  - Kuvasuhde
  - Projektion tyyppi ja projektion parametrit = Kuvatilavuus (view volume)
- ② Lasketaan muunnos jolla kuvatilavuus kuvailee suorakulmaiseksi särmioksi jonka kulmapisteet ovat ( $\pm 1 \ \pm 1 \ 0$  tai  $-1$ ).
- ③ Muunnetaan kulmapisteet tällä muunnoksella.
- ④ Leikataan näkymä särmioon.
- ⑤ Piirretään särmioon sisältö filmitasolle.

## Kameran parametrit perspektiiviprojektiossa

- $\mathbf{x}_{cam}$  = kameran posisio
- $\mathbf{u}_{cam}$  = kameran yläsuunnan määrävä yksikkövektori
- $\mathbf{n}_{cam}$  = kameran filmitason yksikkönormaali
- $\theta_x, \theta_y$  = Näkymän kulma
- $d_{near}, d_{far}$  = Etu- ja takaleikkaustason etäisyys kamerasta

## Kameramuunnoksen laskeminen perspektiiviprojektiossa

- ➊ Siirretään kamera origoon muunnoksella  $\mathbf{T}^{-1}(\mathbf{x}_{cam}) = \mathbf{T}(-\mathbf{x}_{cam})$ .
- ➋ Kierretään kamera s.e  $\mathbf{n}_{cam}$  osoittaa negatiivisen z-akselin suuntaan ja  $\mathbf{u}_{cam}$  osoittaa positiivisen y-akselin suuntaan muunnoksella  $\mathbf{M}$ . Pitää siis löytää matriisi  $\mathbf{M}$  joka kuvailee vektorin  $\mathbf{n}_{cam} \mapsto (0 \ 0 \ -1)$  ja vektorin  $\mathbf{u}_{cam} \mapsto (0 \ 1 \ 0)$ . Palataan tähän ongelmaan myöhemmin.
- ➌ Skaalataan takaleikkaustaso neliöksi  $(\pm 1 \ \pm 1 \ -1)$  Tämä tehdään skaalaamalle ensin x- ja y-koordinaatit arvoon  $d_{far}$  ja sen jälkeen siirretään x, y, z-koordinaatit halutuiksi. Saadaan matriisit  $S_{xy} = \text{diag}(\cot(\theta_x), \cot(\theta_y), 1, 1)$  ja  $S_{xyx} = \text{diag}(1/d_{far}, 1/d_{far}, 1/d_{far}, 1)$ . Kuvatilavuuden etuosan muodostaa nyt neliö  $(\pm k \ \pm k \ -k)$ , jossa  $k = d_{near}/d_{far}$ .
- ➍ Muodostetaan perspektiiviprojektion matriisi

$$\mathbf{D} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{k-1} & \frac{k}{k-1} \\ 0 & 0 & -1 & 0 \end{pmatrix}.$$

- ➎ Kameramuunnos saadaan siis matriisilla  $\mathbf{DS}_{xyx}S_{xy}\mathbf{MT}(-\mathbf{x}_{cam})$ . Tällöin kuvatilavuus on kuvaillut suorakulmaiseen särmiöön  $(\pm 1 \ \pm 1 \ 0 \text{ tai } -1)$ .

## Muunnos $\mathbf{M}$

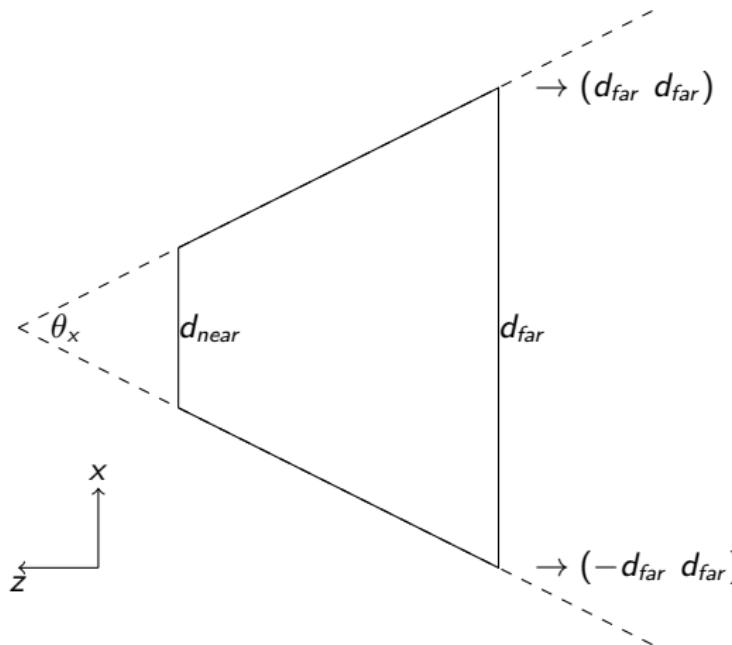
- Kameran suunta on kiinnitetty yksikkövektorien  $\mathbf{u}$  ja  $\mathbf{n}$  avulla.
- Kameran muunnettua suuntaa on kiinnitetty vastaavasti kantavektorien  $\mathbf{e}_2$  ja  $-\mathbf{e}_3$ .
- Tavoitteena löytää  $\mathbf{M}$  s.e.  $\mathbf{Mu} = \mathbf{e}_2$  ja  $\mathbf{Mn} = -\mathbf{e}_3$ .
- Tämä ei kuitenkaan vielä kiinnitä matriisia  $\mathbf{M}$  yksikäsittisesti.
- Lasketaan apuvektorit  $\mathbf{v} = \mathbf{u} \times \mathbf{n}$  ja  $-\mathbf{e}_1 = -\mathbf{e}_2 \times \mathbf{e}_3$ .
- Vaaditaan lisäksi että  $\mathbf{Mv} = -\mathbf{e}_1$ .
- Kirjoitetaan matriisimuodossa

$$\mathbf{M} (\mathbf{v} \quad \mathbf{u} \quad \mathbf{n}) = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix},$$

josta saadaan ratkaistua

$$\mathbf{M} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} (\mathbf{v} \quad \mathbf{u} \quad \mathbf{n})^T = (-\mathbf{v} \quad \mathbf{u} \quad -\mathbf{n})^T$$

## Harjoituksia – Muunnos $S_{xy}$



(Ex 1) Johda muunnos  $S_{xy}$ . VIHJE: Johda muunnos erikseen x- ja y-akseleiden suuntaan ja yhdistää ne.

Diagonaalimatriisi  $\text{diag}(k_x \ k_y \ k_z \ 1)$  skaalaa x koordinaattia kertoimella  $k_x$ , y koordinaattia kertoimella  $k_y$  jne.

## Muunnos D

- Olkoon meillä piste  $\mathbf{x} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$  homogeenisissa koordinaateissa. Lasketaan

$$\mathbf{Dx} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{k-1} & \frac{k}{k-1} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ \frac{z-k}{k-1} \\ -z \end{pmatrix} \sim \begin{pmatrix} x \\ y \\ -x/z \\ -y/z \\ \frac{-1}{k-1} + \frac{k}{z(k-1)} \end{pmatrix}.$$

- Periaatteessa z-koordinaatti ei enää kiinnosta, paitsi käytettäessä z-puskuroinnissa.

## Kameran parametrit vinoprojektiossa

- $\mathbf{x}_{cam}$  = kameran posisio
- $\mathbf{u}_{cam}$  = kameran yläsuunnan määräävä yksikkövektori
- $\mathbf{n}_{cam}$  = kameran filmitason yksikkönormaali
- $w_x, w_y$  = Näkymän korkeus ja leveys

## Kameramuunnoksen laskeminen vinoprojektiossa

- Position ja orientaation muunnos kuten perspektiiviprojektiossa.
- Skaalataan maailmankoordinaatisto sopivaksi.
- Kuvausmatriisi

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & -k \cos \alpha & 0 \\ 0 & 1 & -k \sin \alpha & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

jossa  $k$  ja  $\alpha$  määrävät projektion tyypin.

- Cavalier-projektilo:  $k = 1$ ,  $\alpha = 30^\circ$  tai  $\alpha = 45^\circ$
- Cabinet-projektilo:  $k = 0.5$ ,  $\alpha = 30^\circ$  tai  $\alpha = 45^\circ$

## Kameramuunnoksen laskeminen isometrisessä projektiossa

- Position ja orientaation muunnos kuten perspektiiviprojektiossa.
- Skaalaus kuten vinoprojektiossa.
- Kuvausmatriisi

$$\mathbf{P} = \frac{1}{\sqrt{6}} \begin{pmatrix} \sqrt{3} & 0 & -\sqrt{3} & 0 \\ 1 & 2 & -1 & 0 \\ \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} .$$

## Lisätietoja

- [https://en.wikipedia.org/wiki/3D\\_projection](https://en.wikipedia.org/wiki/3D_projection)
- [https://en.wikipedia.org/wiki/Isometric\\_projection](https://en.wikipedia.org/wiki/Isometric_projection)
- [https://en.wikipedia.org/wiki/OblIQUE\\_projection](https://en.wikipedia.org/wiki/OblIQUE_projection)

## Harjoituksia

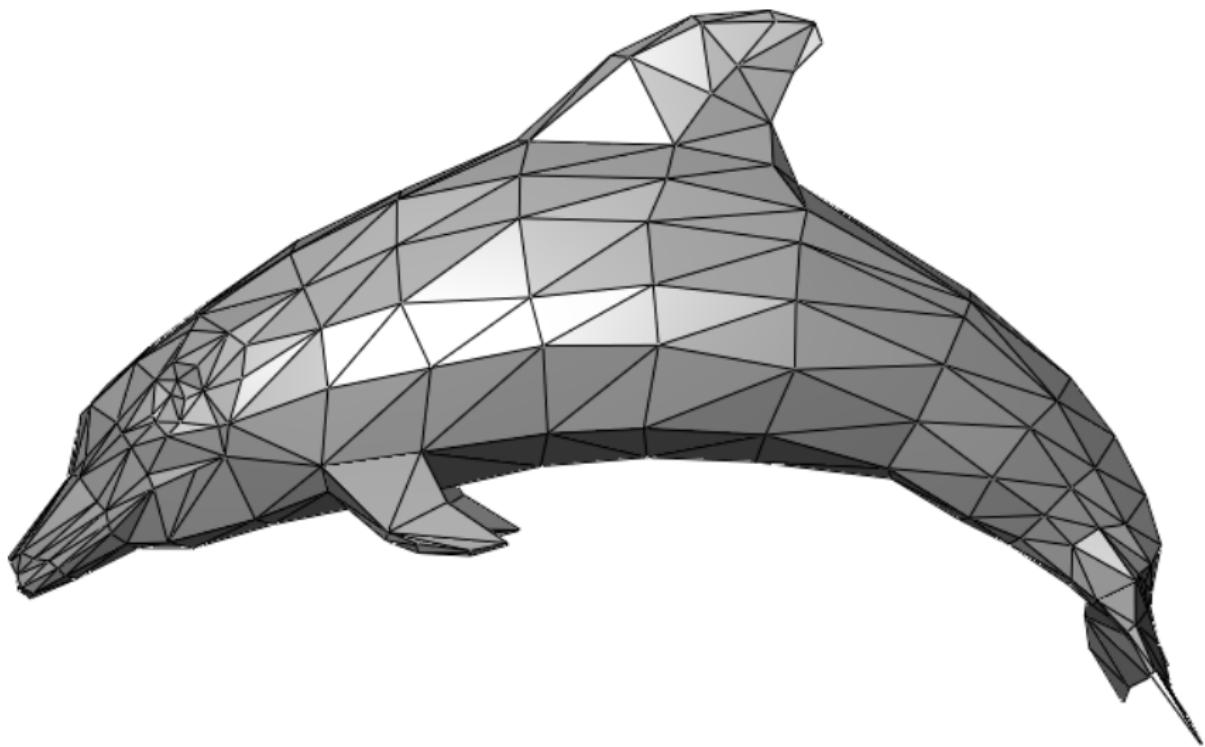
- (Ex 2) Lisää matriisiluokkaan perspektiivimuunnoksen vaatimat matriisit.
- (Ex 3) Tutki miten saat perspektiivimuunnoksen toteutettua OpenGL:llä ja kirjoita yksinkertainen testiohjelma.

## Perusteet

Yleensä 3D-kappaleet piirretään siten että:

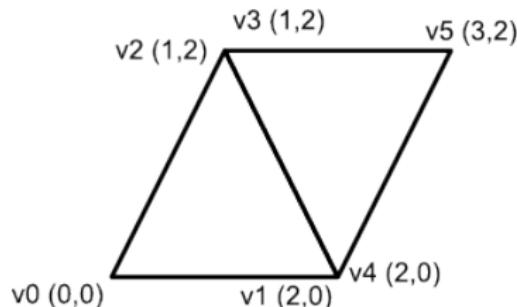
- Vain piirrettävän kappaleen pinta piirretään.
- Piirrettävä pinta jaetaan tasomaisiin monikulmioihin, usein kolmioihin (triangle mesh).
- Yleensä kolmioiden kärkipisteet talletetaan erillisen listaan (vertexlist).
  - Kun useampi kolmio jakaa yhteen kärkipisteen talletetaan se listaan vain kerran.
  - Kolmioiden kulmapisteiden indeksit talletetaan toiseen listaan (indexlist).
  - On myös mahdollista tallettaa kärkipisteet kolmioittain, jolloin indeksilistaa ei käytetä.
- Kolmioinnin päällä käytetään usein jotain tehosteita kuten esimerkiksi tekstuuriointia, valaistusmalleja jne., joilla pinta saadaan näyttämään luonnollisemmalta.

## Kolmiointi



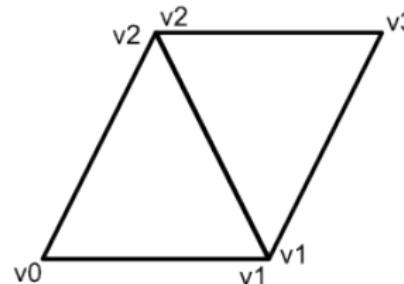
## Indeksointi

Without indexing



[0,0, 2,0, 1,2, 1,2, 2,0, 3,2]

With indexing



[0,1,2, 2,1,3]  
[0,0, 2,0, 1,2, 3,2]

Vertices  
reused  
twice

## Objektien luonti

- Periaatteessa kappaleet voi luoda kirjoittamalla kolmointi ja indeksointi koodiin.
  - Kärkipisteet kirjoitetaan suoraan listana tai
  - kärkipisteet lasketaan jollain algoritmilla.
  - Monimutkaisissa kappaleissa yleensä epäkäytännöllistä.
- Tavallisesti kappaleet suunnitellaan erillisillä piirtotyökaluilla, kuten esimerkiksi  tai  AUTODESK 3DS MAX .
- Siirto piirto-ohjelmasta ulkoiseen sovellukseen tapahtuu tiedoston kautta.
- Tarkastellaan tässä yhteydessä OBJ-tiedostotyyppiä.

## Yleistä

- Tekstipohjainen
- Avoin
- Kärkipisteiden positiot
- uv-koordinaatit
- Pinnan normaalit
- Indeksointi **vastapäivään**
- Dokumentti esimerkiksi  
[http://www.cs.utah.edu/~boulos/cs3505/obj\\_spec.pdf](http://www.cs.utah.edu/~boulos/cs3505/obj_spec.pdf).

## Kuutio (1/3)

```
# http://people.sc.fsu.edu/~jburkardt/data/obj/cube.obj
# cube.obj
#
g cube

#vertices
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
```

## Kuutio (2/3)

```
#vertex normals
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
```

## Kuutio (3/3)

```
#face elements
f 1//2 7//2 5//2
f 1//2 3//2 7//2
f 1//6 4//6 3//6
f 1//6 2//6 4//6
f 3//3 8//3 7//3
f 3//3 4//3 8//3
f 5//5 7//5 8//5
f 5//5 8//5 6//5
f 1//4 5//4 6//4
f 1//4 6//4 2//4
f 2//1 6//1 8//1
f 2//1 8//1 4//1
```

## Harjoituksia

(Ex 4) Tutustu OBJ-tiedostomuotoon ja toteuta kirjasto jonka avulla saat luettua obj-tiedostoja omiin ohjelmiisi. Tee kirjastoon ensin rajoitettu tuki obj-tiedostoiille ja varaudu myöhemmin lisäämään siihen tuki esimerkiksi materiaaleille.

Esimerkkitiedostoja löytyy esimerkiksi sivulta

<http://people.sc.fsu.edu/~jb Burkardt/data/obj/obj.html>.

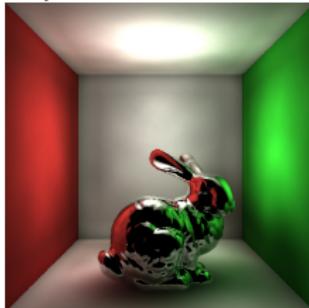
## Taustaa

- 3D-graafikan piirron eri vaiheita kuvataan usein liukuhihnnana (pipeline).
- Piirrettävä data liikkuu liukuhihnnalla eteenpäin ja kussakin askellessa dataa "jalostetaan" askelen verran lähemmäksi lopullista kuvaa.
- Lähtödata

```
short face_indices[16301][6] = [
    {1538, 2410, 1101 ,0,1,2 }, {6196 ,1101 ,4,5 }, {696, 704, 101 ,6,7,5 },
    {1192, 117, 1113 ,8,9,10 }, {1192, 117, 1113 ,11,12,13 }, {540, 1192, 1113 ,14,8,10 },
    {704, 696, 398 ,7,6,15 }, {704, 696, 398 ,16,5,16 }, {1605, 772, 4540 ,17,18,19 },
    {1192, 540, 1857 ,8,14,13 }, {1192, 540, 1857 ,18,3,20 }, {713, 173, 1260 ,3,12,11 },
    {173, 1192, 1857 ,12,8,13 }, {173, 1192, 1857 ,13,1260 ,20,3,11 }, {804, 4128, 4120 ,21,22,23 },
    {1133, 804, 4120 ,24,21,23 }, {1133, 804, 4120 ,25,27,28 }, {4140, 1534, 4136 ,25,27,28 },
    {4531, 4140, 4136 ,26,25,26 }, {4531, 4140, 4136 ,27,28,29 }, {4515, 1609, 4485 ,30,31,32 },
    {710, 1396, 889 ,33,34,35 }, {710, 1396, 889 ,36,37,38 }, {2046, 1905, 768 ,39,40,41 },
    {2453, 5699, 5698 ,42,43,44 }, {2453, 5699, 5698 ,45,46,47 }, {768, 1905, 1393 ,41,40,45 },
    {1393, 1490, 4112 ,45,46,47 }, {1393, 1490, 4112 ,48,49,50 }, {190, 4281, 839 ,46,51,52 },
    {4112, 1490, 839 ,47,46,52 }, {4112, 1490, 839 ,48,49,53 }, {897, 8879, 3286 ,54,55,56 },
    {920, 4291, 1937 ,57,58,59 }, {920, 4291, 1937 ,60,61,62 }, {4915, 4914 ,63,64,65 },
    {1093, 4537, 840 ,66,67,68 }, {1093, 4537, 840 ,69,70,71 }, {59, 4158 ,70,71,72 },
    {7248, 7323, 7322 ,73,74,75 }, {7248, 7323, 7322 ,76,77,78 }, {5289, 5266 ,79,80,81 },
    {4197, 4217, 4258 ,82,83,84 }, {4197, 4217, 4258 ,85,86,87 }, {4536, 4535 ,88,89,90 },
    {4497, 4498, 4535 ,91,92,93 }]
```



- Lopullinen kuva



## Lähtödata

- Vertex-data:
  - Positio
  - Normaali
  - Väri
  - $u, v$ -koordinaatit tekstuurille, normal-mapille jne.
  - jne.
- Kolmiodata
  - Indeksit vertex-dataan
  - Materiaali
- Valaistus

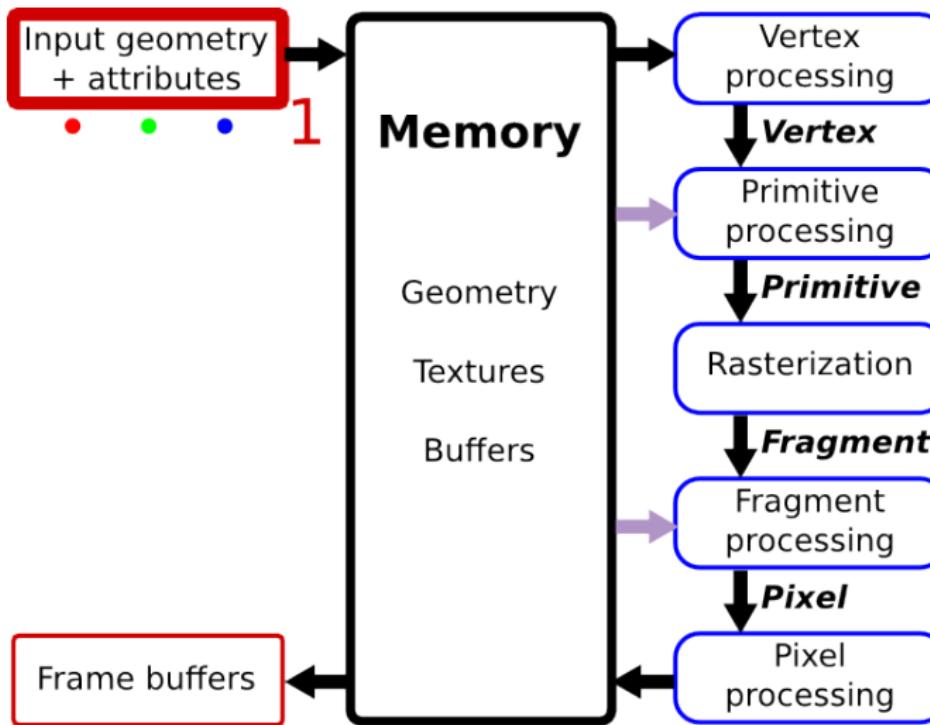
## Materiaalit

- Väri
- Tekstuuri
- Normal mapping
- Valaistusmalli:
  - Ambient
  - Diffuusi
  - Peili
  - jne.

## Valaistus

- Valonlähteen tyyppi
  - Pistemäinen valo
  - Suunnattu valo ( $\approx$ pistemäinen valo kaukana)
  - Spotti ( $\approx$ suunnattu pistemäinen valo)
- Väri

## Liukuhihna – Yksinkertaistettu malli



## Verteksien käsittely

- Sisään: Verteksi
  - Koordinaatti
  - Väri
  - $u, v$ -koordinaatit
  - Normaali
  - ...
- Verteksin siirto, kierto ym.
- Perspektiivimuunnos ym.
- Valaistuslaskenta (verteksille)
- Ulos: Muokattu verteksi

## Primitiivien käsittely

- Sisään: Verteksit ja indeksointi
- Muodostetaan primitiivi kärkipisteistä ja indekseistä
- Backface culling
- Jako pienempiin primitiiveihin
- Leikkaus kuvatilavuuteen
- Ulos: Primitiivi / useampia primitiivejä

## Rasterointi

- Sisään: Primitiivi
- Lasketaan primitiivin pikseleiden paikat
- Interpoloidaan kärkipisteille annetut arvot primitiivin yli
  - Koordinaatti
  - $u, v$ -koordinaatit
  - Väriarvot
  - Normaali
  - ...
- Ulos: Fragmentteja (primitiivin pikselit)

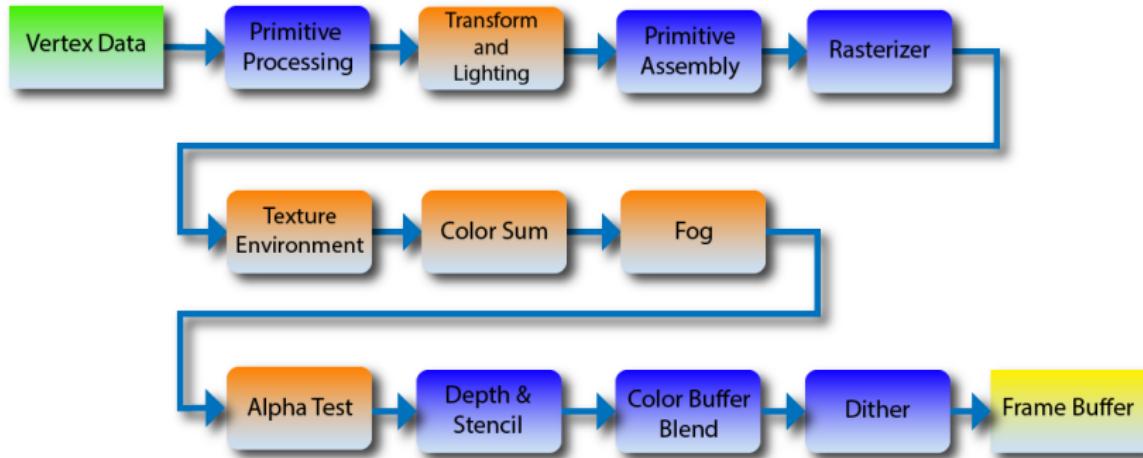
## Fragmenttien käsittely

- Sisään: Fragmentti
  - Koordinaatti
  - $u, v$ -koordinaatit
  - Väriarvot
  - Normaali
  - ...
- Muokataan fragmentin väri
  - Teksturointi
  - Valaistusmalli
- Ulos: Muokattu fragmentti
  - Koordinaatit
  - Väri

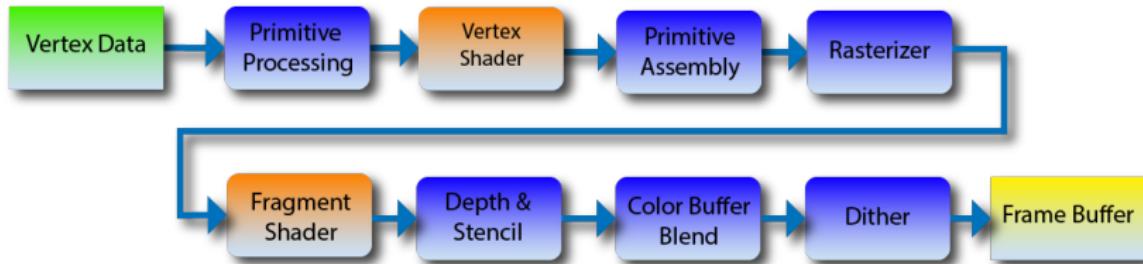
## Pikslien käsittely

- Sisään: Pikselin väriarvo
- Z-puskurointi
- Stencil-buffer
- $\alpha$ -blending
- Ulos: Pikseli

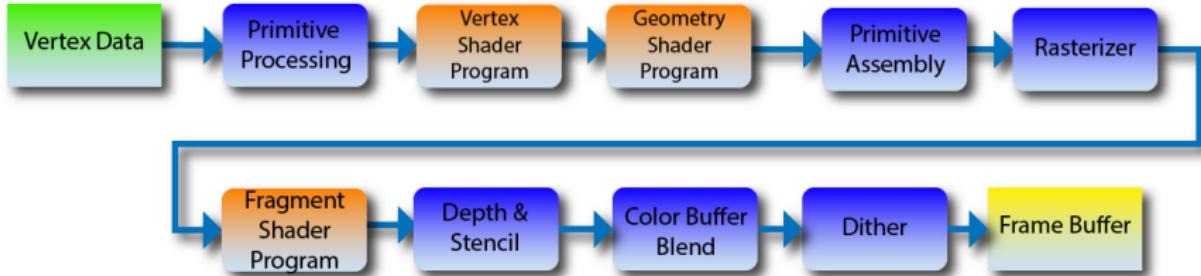
# Liukuhihna – OpenGL 1.0



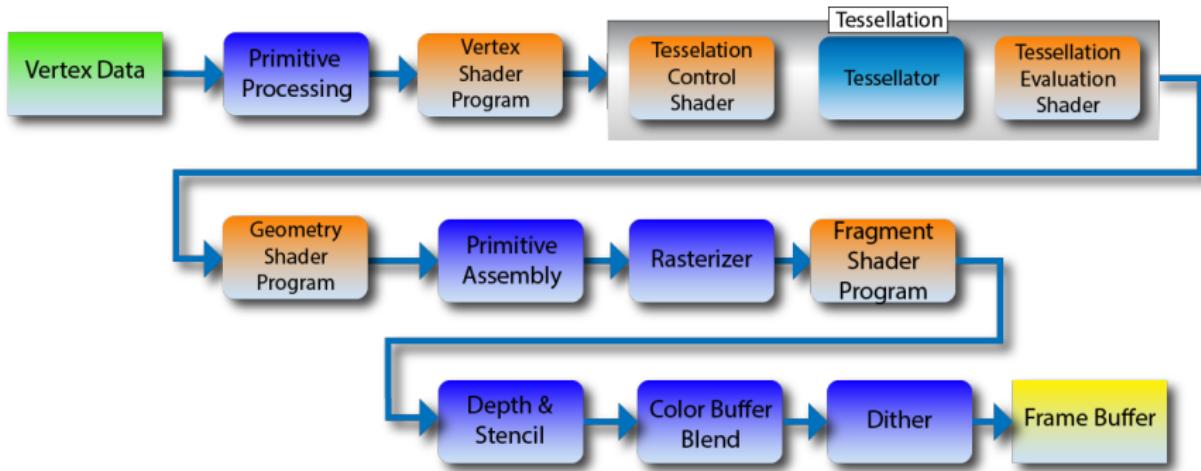
# Liukuhihna – OpenGL 2.0



## Liukuhihna – OpenGL 3.2



# Liukuhihna – OpenGL 4.0



## Shaderit

- Ohjelmoitava osa OpenGL-liukuhihnaa
- Vertex shader, Geometry shader, Fragment shader
- Tessellation control ja Evaluation shader, OpenGL 4.0 tai `ARB_tessellation_shader`
- Compute shader, OpenGL 4.3 tai `ARB_compute_shader`
- Ohjelmointi tapahtuu C-kieltä muistuttavalla GLSL-kielellä.
- `http://www.kickjs.org/example/shader_editor/shader_editor.html`

## GLSL

- C-kieltä muistuttava kieli.
- Aloituskohta `void main()`.
- Tuki funktioille, paluuarvo `return`-avainsanalla.
- `float, double, int, uint, bool, vec*, dvec*, ivec*, uvec*, bvec*, mat*, dmat*, sampler*`
- `struct`
- `const, in, out, uniform, buffer, (Compute shaders:shared)`
- `if-else, switch-case, ? :`
- `for, while, do-while, continue, break`
- `discard` - Vain fragment shaderissa. Poistutaan shaderista ja pikseli jätetään piirtämättä.
- Laaja joukko matemaattisia funktioita. Monilta osin laajemmin kuin C:ssä, ei sisällä kuitenkaan `rand()`-funktiota.
- <https://www.opengl.org/sdk/docs/man/html/>
- Ei esikääntääjää → Esimerkiksi `#include` ei toimi.
- `#define` toimii!

## Esimerkkejä shadereiden ohjelmoinnista

- [http://www.kickjs.org/example/shader\\_editor/shader\\_editor.html](http://www.kickjs.org/example/shader_editor/shader_editor.html)
- Katsotaan esimerkki "Head (unlit)"
- Toteutetaan tiilikuutio
- Toteutetaan partikkelpilvi
- Toteutetaan teepannuun yksinkertainen valaistus ja "hytkyminen".

## Harjoituksia

- (Ex 5) Toteuta varjostimilla ohjelma joka piirtää joka neljännen sarakkeen ja joka neljännen rivin. Muut rivit piirretään mustalla.
- (Ex 6) Toteuta varjostimilla ohjelma joka *uv*-koordinaattien avulla luo tiilipinnan.
- (Ex 7) Toteuta varjostimilla ohjelma, joka piirtää kappaleen raidallisena s.e raidat ovat *xz*-tasossa.

Malliratkaisut ovat seuraavilla kalvoilla, älä kurki ennen kuin olet yrittänyt itse keksiä ratkaisut!

## Ruudukko fragment shaderilla

```
1 precision highp float;
2
3 varying vec2 uv;
4 uniform sampler2D tex;
5
6 void main(void)
7 {
8     gl_FragColor = vec4(texture2D(tex,uv).xyz, 1.0);
9     float mx=mod(gl_FragCoord.x , 4.0);
10    float my=mod(gl_FragCoord.y , 4.0);
11    if((mx<3.0) && (my<3.0)) {
12        //discard;
13        gl_FragColor = vec4(0.0, 0.0, 0.0, 1.0);
14    }
15 }
```

## Tiilikuvio fragment shaderilla

```
1 precision highp float;
2
3 varying vec2 uv;
4
5 uniform sampler2D tex;
6
7 void main(void)
{
8     const float brickW=0.33;
9     const float brickH=0.1;
10    const float mortarX=0.1*brickW;
11    const float mortarY=mortarX;
12
13    float mx=mod(uv.x+0.5*floor(uv.y/brickH) , brickW);
14    float my=mod(uv.y , brickH);
15    if((mx>mortarX) && (my>mortarY)) {
16        gl_FragColor = vec4(0.9, 0.1, 0.1, 1.0);
17    } else {
18        gl_FragColor = vec4(0.9, 0.9, 0.9, 1.0);
19    }
20}
```

## Raidallinen objekti

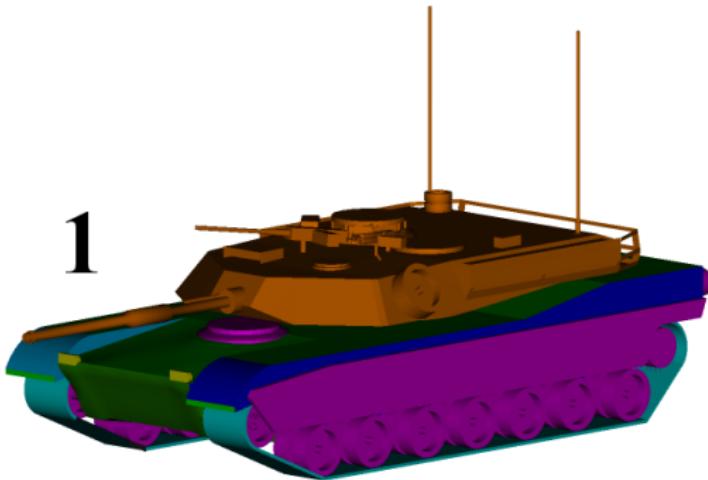
```
1 attribute vec3 vertex;
2 uniform mat4 _mvProj; // model-view-projection matrix
3 uniform float _time; // time in seconds
4 varying vec4 pos;
5
6 void main(void) {
7     vec3 s=vec3(0.5*(2.1+sin(vertex.y*5.0+0.005*_time)),1.0, ←
8         0.5*(2.1+sin(vertex.y*5.0+0.005*_time)));
9     vec3 vtx=vertex*s;
10    gl_Position = _mvProj * vec4(vtx, 1.0);
11    pos = gl_Position;
12 }
13
14 precision highp float;
15 uniform float _time;
16 varying vec4 pos;
17
18 void main(void)
19 {
20     float r=pos.y-0.001*_time;
21     if(mod(5.0*r, 2.0)>1.0) {
22         gl_FragColor = vec4(0.5, 0.5, 0.5,1.0);
23     } else {
24         gl_FragColor = vec4(1.0, 0, 0, 1.0);
25     }
26 }
```

## Taustaa

- Reaalimaailman kappaleet ovat harvoin yksivärisiä.
- Periaatteessa olisi mahdollista muodostaa kappaleet siten että kukaan pinnan kolmio olisi yksiväri ja näiden kolmioiden avulla muodostettaisiin kappaleen väritys.  
⇒ Kolmioiden määrä kasvaisi tarpeettoman suureksi.
- Ratkaisu: Levitetään bittikarttakuva (tekstuuri) kappaleen pinnalle, jolloin pinnan väritksellä voidaan luoda illusio yksityiskohtaisemmasta mallista.

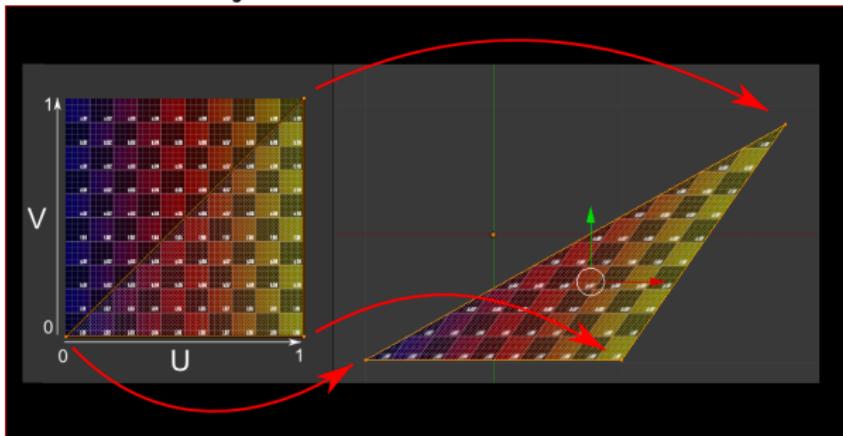


## Esimerkki

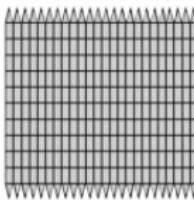
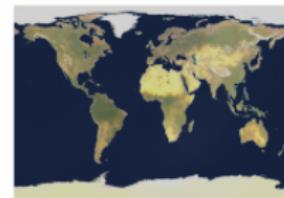


## uv-koordinaatit

- Tekstuuri kiinnitetään kaappaleen pinnalle vertekseittäin.
- Kullekin verteksille annetaan *uv*-koordinaatti, jossa  $(u, v) \in [0, 1] \times [0, 1]$ .
- Tekstuuri venytetään piirrettävän primitiivin päälle käyttäen *uv*-koordinaatteja.



# Esimerkki

**3-D Model** $p = (x, y, z)$ **UV Map** $p = (u, v)$ **Texture**

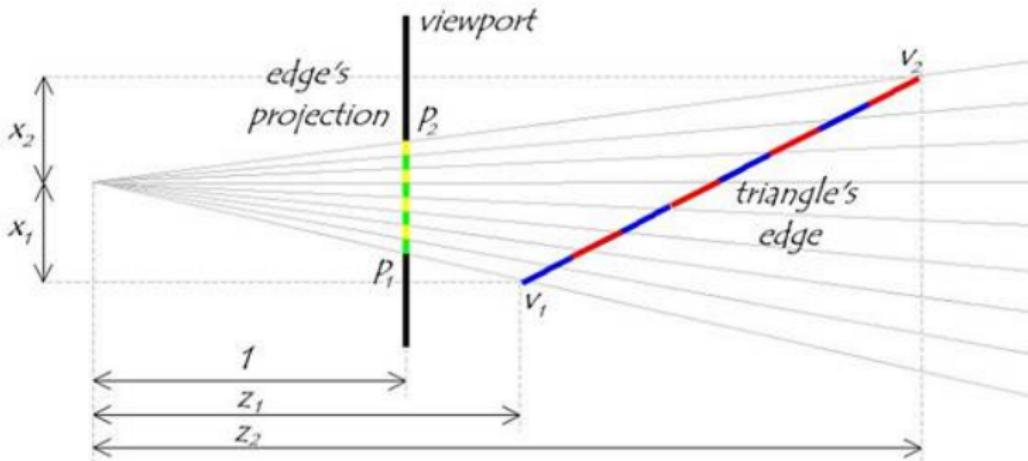
## Miten tekstuuri venytetään?



- Jos venytys tehdään näytön koordinaatistossa lineaarisesti, voidaan saada vääräistynyt lopputulos.
- $u(\alpha) = (1 - \alpha)u_0 + \alpha u_1$   
 $v(\alpha) = (1 - \alpha)v_0 + \alpha v_1, \alpha \in [0, 1].$

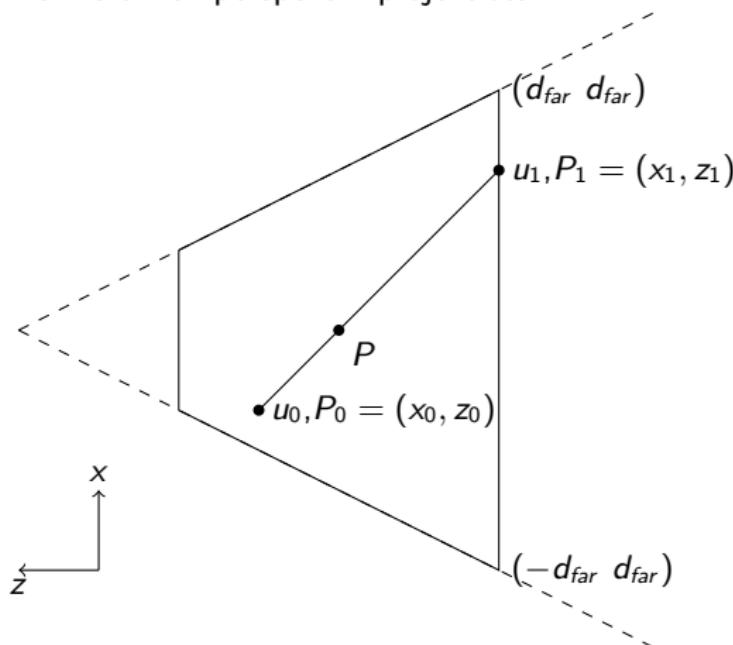
## Mikä meni pieleen?

- Jos piirrettävä pinta ei ole samansuuntainen kuvatason kanssa ei kuvatason pikselin projektiion koko piirrettävällä pinnalla ole vakio.



## Perspektiivikorjattu interpolointi (1/3)

- Tilanne ennen perspektiiviprojektiota:



- Olkoon  $\beta \in [0, 1]$ . Merkitään  $P = (1 - \beta)P_0 + \beta P_1$ .

## Perspektiivikorjattu interpolointi (2/3)

- Olkoon  $\beta \in [0, 1]$ . Merkitään  $P = (1 - \beta)P_0 + \beta P_1$ .
- Tällöin tekstuurikoordinaatille pisteessä  $P$  pätee  $u_P = (1 - \beta)u_0 + \beta u_1$ .

- Perspektiivimuunnoksen  $\mathbf{D} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{k-1} & \frac{k}{k-1} \\ 0 & -1 & 0 \end{pmatrix}$  jälkeen<sup>5</sup>

$$\mathbf{D}P_0 = \begin{pmatrix} x_0 & \frac{z_0 - k}{k-1} & -z_0 \end{pmatrix}^T \Rightarrow \hat{x}_0 = -\frac{x_0}{z_0}$$

$$\begin{aligned} \mathbf{D}P &= \left( (1 - \beta)x_0 + \beta x_1, \frac{(1 - \beta)z_0 + \beta z_1 - k}{k-1}, -(1 - \beta)z_0 - \beta z_1 \right)^T \\ &\Rightarrow \hat{x}_P = -\frac{(1 - \beta)x_0 + \beta x_1}{(1 - \beta)z_0 + \beta z_1} \end{aligned}$$

$$\mathbf{D}P_1 = \begin{pmatrix} x_1 & \frac{z_1 - k}{k-1} & -z_1 \end{pmatrix}^T \Rightarrow \hat{x}_1 = -\frac{x_1}{z_1}$$

- Valitaan  $\alpha$  s.e  $\hat{x}_P = (1 - \alpha)\hat{x}_0 + \alpha\hat{x}_1$ .

---

<sup>5</sup>homogeenisissa koordinaateissa

## Perspektiivikorjattu interpolointi (3/3)

- Valitaan  $\alpha$  s.e  $\hat{x}_P = (1 - \alpha)\hat{x}_0 + \alpha\hat{x}_1$ .
- Kirjoitetaan tämä auki (ja kerrotaan  $-1$ :llä).

$$\frac{(1 - \beta)x_0 + \beta x_1}{(1 - \beta)z_0 + \beta z_1} = (1 - \alpha)\frac{x_0}{z_0} + \alpha\frac{x_1}{z_1}$$

- Ratkaistaan  $\beta$ :

$$\beta = \frac{\alpha z_0}{\alpha z_0 + (1 - \alpha)z_1}.$$

- Saatiin siis interpolointikaavat

$$\begin{aligned} u(\alpha) &= (1 - \beta)u_0 + \beta u_1 \\ v(\alpha) &= (1 - \beta)v_0 + \beta v_1 . \end{aligned}$$

- OpenGL laskee  $uv$ -koordinaateille<sup>6</sup> interpoloinnin näin, joten yleensä tätä ei tarvitse laskea itse.

---

<sup>6</sup>ja kaikille muillekin interpoloitaville suureille

## Teksturoinnin tehostaminen

- Jos teksturoitu kappale on kaukana (=pieni) ei teksturoinnin tarvitse olla korkearesoluutioinen.
- Periaatteessa tarkasta tekstuurista voidaan näytteistää sopiva pienemmän resoluution tekstuuri.
  - Jos näytteistys tehdään nopeasti (lähimmän pikselin menetelmällä) on tuloksena joissain tapauksissa moire-ilmiö.
  - Jos näytteistys tehdään paremmilla menetelmissä, joudutaan yhtä piirrettävää pikseliä kohden lukemaan useita pikseleitä tekstuurista.
- Mipmapping: Talletetaan tekstuurit usealla resoluutiolla, jolloin skaalaus voidaan laskea etukäteen.
  - Jos skaalataan  $\frac{1}{2}$ :een, vaaditaan lisätilaa vain  $\frac{1}{3}$  alkuperäisestä tilasta.
  - Perustelu: Olkoon alkuperäisen tekstuurin koko  $n \times n = n^2$  pikseliä. Tällöin ensimmäisen skaalatun tekstuurin koko on  $\frac{n}{2} \times \frac{n}{2}$ , toisen skaalatun tekstuurin koko  $\frac{n}{2^2} \times \frac{n}{2^2}$  jne. Kokonaistila jonka mipmapping vaatii on siis (enimmillään)  
$$\sum_{k=0}^{\infty} \frac{n}{2^k} \times \frac{n}{2^k} = n^2 \sum_{k=0}^{\infty} 2^{-2k} = n^2 \frac{4}{3}.$$

## Moire-ilmiö

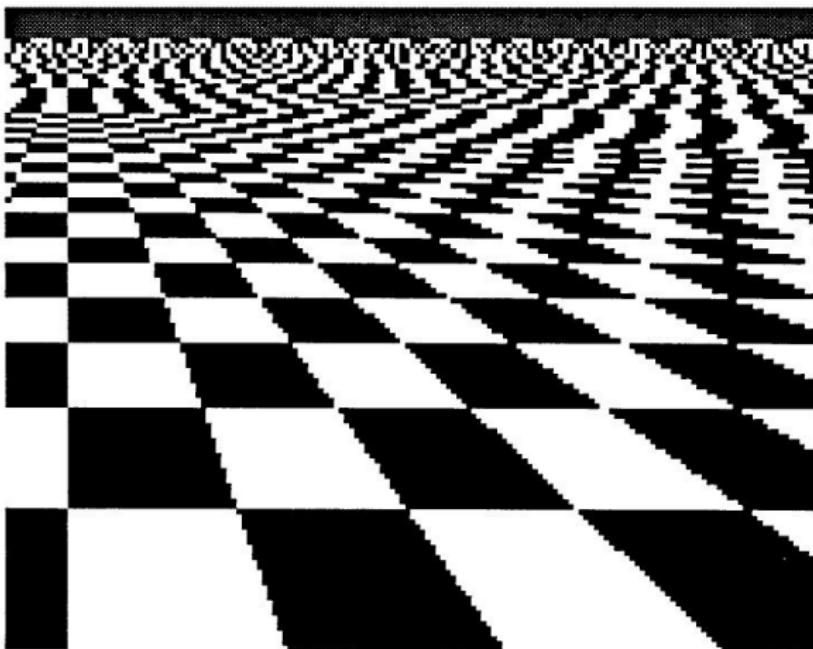
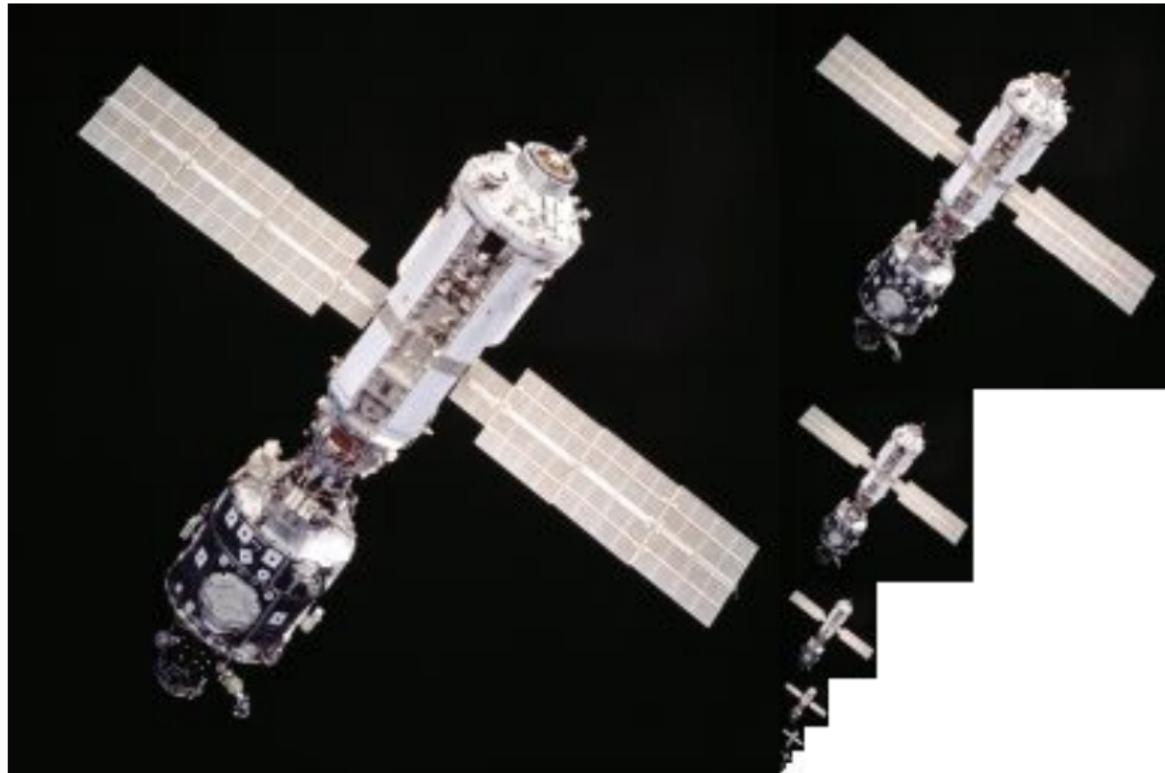
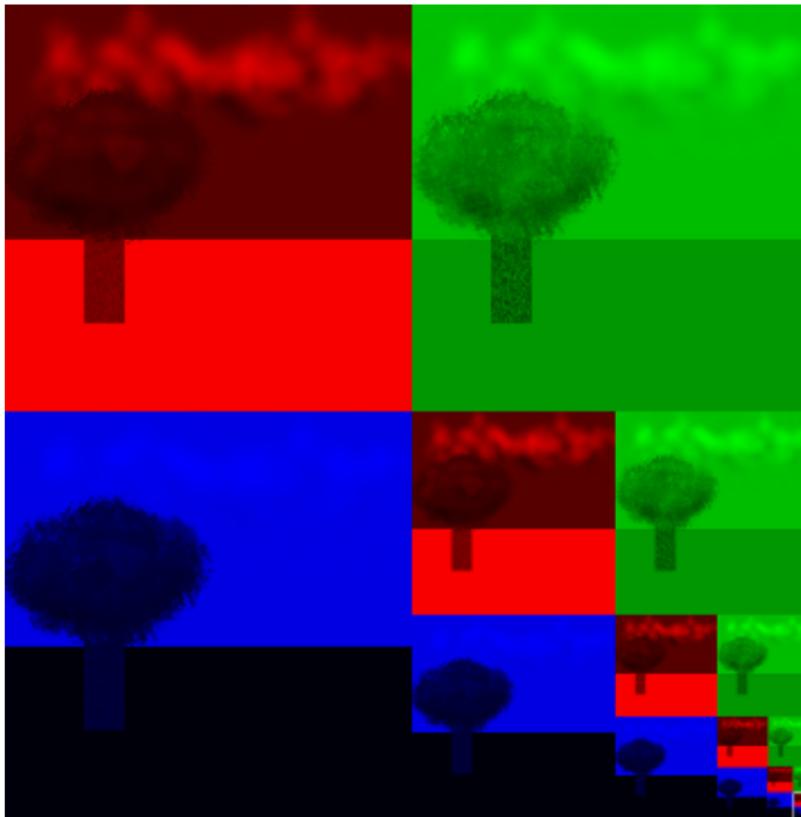


Figure 3.2: *Texture mapping with point sampling, with more extreme compression. Note moire near horizon.*

## Esimerkki



## Esimerkki



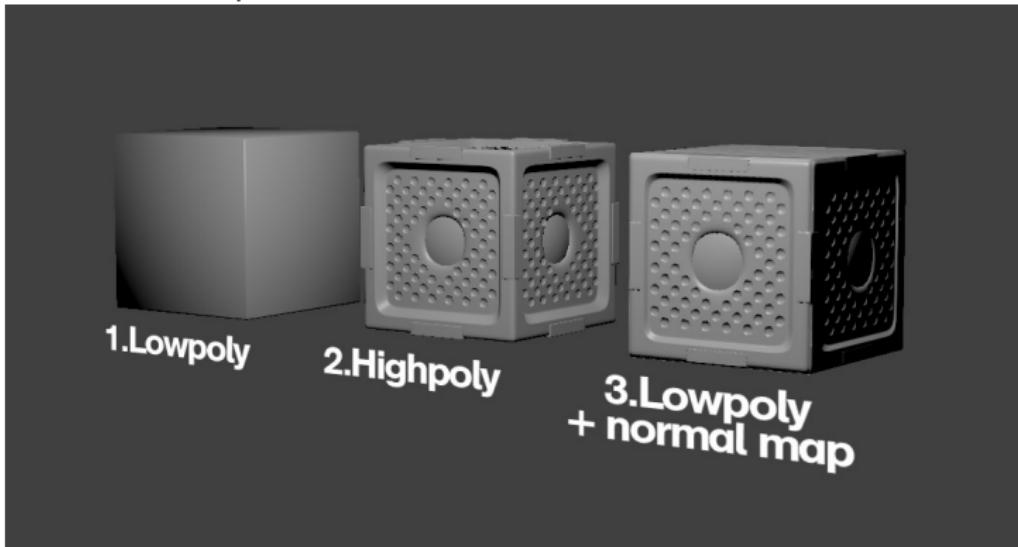
## OpenGL:ssä

- <http://www.opengl-tutorial.org/beginners-tutorials/tutorial-5-a-textured-cube/>

```
1 unsigned char * data;
2 fill data...
3
4 GLuint textureID;
5 glGenTextures(1, &textureID);
6
7 glBindTexture(GL_TEXTURE_2D, textureID);
8
9 glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0,
10   GL_BGR, GL_UNSIGNED_BYTE, data);
11
12 data voidaan vapauttaa...
13
14 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
15 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
16 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
17 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
18   GL_LINEAR_MIPMAP_LINEAR);
19 glGenerateMipmap(GL_TEXTURE_2D);
```

## Taustaa

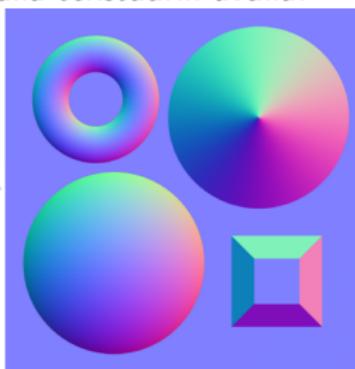
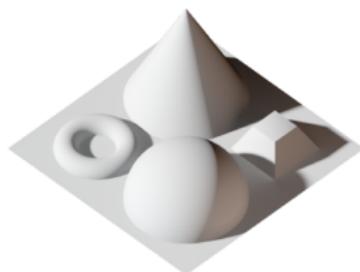
- Useat valaistusmallit hyödyntävät pinnan normaalitietoa.
- Reaalimaailman materiaaleissa pinnan normaali voi vaihdella paljon "tasaisessakin" pinnassa.



- Jos tästä vaihtelua mallinnetaan tarkasti, voi objektiin kolmioiden määrä kasvaa liian suureksi.

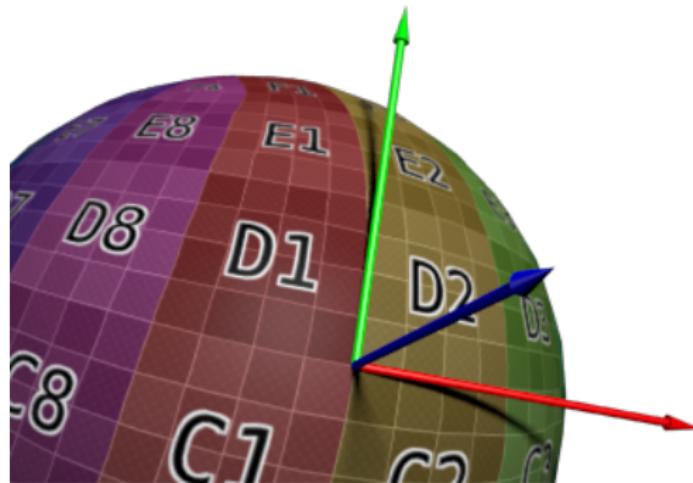
## Tekniikka silmän huijaamiseksi

- Lasketaan valaistus kuten kappaleessa olisi pienempiä muotoja.
- Korjataan pinnan normaalit tekstuurin avulla.



- $n = 2 \text{ color} - 1$
- Ongelma: Missä koordinaatistossa normaali on annettu?

## Tangenttikoordinaatisto



**n** = Kolmion yksikkönormaali (sininen)

**t** = Tangenti (yksikkövektori) *u*-akselin suuntaan (punainen)

**b** = Bitangenti (yksikkövektori) ,  $\mathbf{b} = \mathbf{t} \times \mathbf{n}$  (vihreä)

$\mathbf{T}_{TBN} = \begin{pmatrix} \mathbf{t}_x & \mathbf{b}_x & \mathbf{n}_x \\ \mathbf{t}_y & \mathbf{b}_y & \mathbf{n}_y \\ \mathbf{t}_z & \mathbf{b}_z & \mathbf{n}_z \end{pmatrix}$ , muuntaa normaalilin tangenttikoordinaatistosta

objektiin koordinaatistoon.

Koska  $\mathbf{T}_{TBN}$  on ortogonaalinen, on sen käänneismatriisi  $\mathbf{T}_{TBN}^T$ .

## Siirrosmatriisi muunnoksissa

- Yleisesti muunnosmatriisi voidaan kirjoittaa muodossa  $\begin{pmatrix} \mathbf{A} & \mathbf{m} \\ \mathbf{0} & 1 \end{pmatrix}$ .
- Tutkitaan muunnosmatriisiin kertolaskua

$$\begin{pmatrix} \mathbf{A} & \mathbf{m} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{B} & \mathbf{n} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{AB} + \mathbf{m0} & \mathbf{An} + \mathbf{m1} \\ \mathbf{0B} + \mathbf{10} & \mathbf{0n} + \mathbf{1} \cdot \mathbf{1} \end{pmatrix} = \begin{pmatrix} \mathbf{AB} & \mathbf{An} + \mathbf{m} \\ \mathbf{0} & 1 \end{pmatrix}.$$

⇒ Siirrososa **n** tai **m** ei vaikuta vasemmalla ylhäällä olevaan  $3 \times 3$  lohkoon.

## Pinnan normaali muunnoksissa

- Olkoon annettu kolmion normaali  $\mathbf{n} \in \mathbb{R}^3$  objektiin koordinaatistossa ja olkoon vastaavasti sitä kohden kohtisuora vektori  $\mathbf{t}$ .
- Tällöin  $\mathbf{t} \cdot \mathbf{n} = 0$ .
- Olkoon käännyvä muunnosmatriisi  $\mathbf{A} \in \mathbb{R}^{3 \times 3}$ .
- Jotta  $\mathbf{n}'$  olisi muunnetun pinnan normaali, on oltava voimassa

$$\begin{aligned} 0 &= \mathbf{n}' \cdot \mathbf{At} \\ &= \mathbf{n}'^T \mathbf{At} \\ &= (\mathbf{A}^T \mathbf{n}')^T \mathbf{t} \\ &= (\mathbf{A}^T \mathbf{n}') \cdot \mathbf{t}. \end{aligned}$$

- Tämä pätee kaikilla  $\mathbf{t}$  kun valitaan  $\mathbf{n}'$  s.e  $\mathbf{A}^T \mathbf{n}' = \mathbf{n}$ .
- Koska  $\mathbf{A}^T$  on käännyvä, niin  $\mathbf{n}' = (\mathbf{A}^T)^{-1} \mathbf{n}$ .

## Toteutus (1/2)

- <http://learnopengl.com/#!Advanced-Lighting/Normal-Mapping>
- Vertexshaderissa:

$$\mathbf{t}' = \mathbf{Nt}$$

$$\mathbf{b}' = \mathbf{Nb}$$

$$\mathbf{n}' = \mathbf{Nn}_{vertex}$$

$$\mathbf{T}_{TBN} = \dots$$

$$\mathbf{x}'_{vertex} = \mathbf{Mx}_{vertex}$$

$$\mathbf{x}'_{light} = \mathbf{T}_{TBN}^T \mathbf{x}_{light}$$

$$\mathbf{x}'_{view} = \mathbf{T}_{TBN}^T \mathbf{x}_{view},$$

jossa  $\mathbf{M}$  on mallin muunnosmatriisi ja  $\mathbf{N} = \text{sub}_{33}(\mathbf{M})^{T^{-1}}$ . Tässä  $\text{sub}_{33}$  palauttaa parametrimatriisin sarakkeiden 1...3 rivit 1...3.

## Toteutus (2/2)

- Fragmentshaderissa:

$$\mathbf{n}_{frag} = 2 \text{texture}(u, v) - 1$$

$$\mathbf{d}_{light} = \mathbf{x}'_{light} - \mathbf{x}'_{frag}$$

$$\mathbf{d}_{view} = \mathbf{x}'_{view} - \mathbf{x}'_{frag},$$

jossa  $\mathbf{n}_{frag}$  on fragmentin normaali,  $\mathbf{d}_{light}$  on suuntavektori valonlähteeseen ja  $\mathbf{d}_{view}$  on suuntavektori kameraan.

- Tarvittaessa normalisoiti edelliset ennen käyttöä.
- Tässä muunnettiin valon ja kameran positio tangenttiavaruuteen ja laskettiin normaalit siellä. Yleensä tällainen lähestymistapa riittää yksinkertaisille valaistusmalleille.

## Menetelmiä

- Tekstuurien avulla voidaan muokata kuva realistisemman näköiseksi.
- Esimerkiksi valonlähteen aiheuttamat valot ja varjot voidaan laskea etukäteen tekstuureihin ja käyttää niitä osana pelin valaistusmallia (light mapping).
  - Valonlähteen on pysytävä paikallaan.
  - Riippuen valaistusmallista voidaan joissakin tapauksissa useamman valonlähteen vaikutus laskea samaan tekstuuriin.
  - Kirkkaus tai väri voi muuttua.
- Valaistuksen parametreja voidaan hallita fragmenttatasolla tekstuurien avulla, esimerkiksi pinnan peilimäisyyttä (gloss mapping).
- Simuloidaan pinnan syvyyden aiheuttamia perspektiiviefektejä (Parallax mapping).

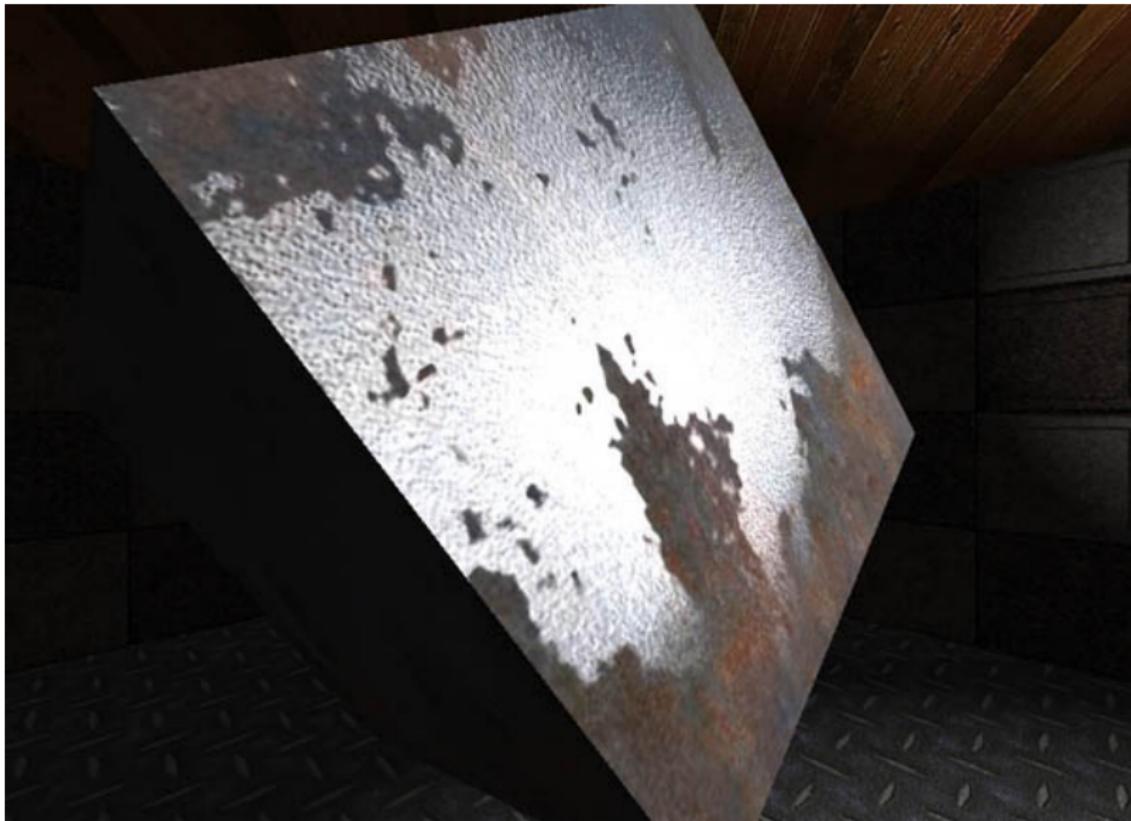
## Light mapping



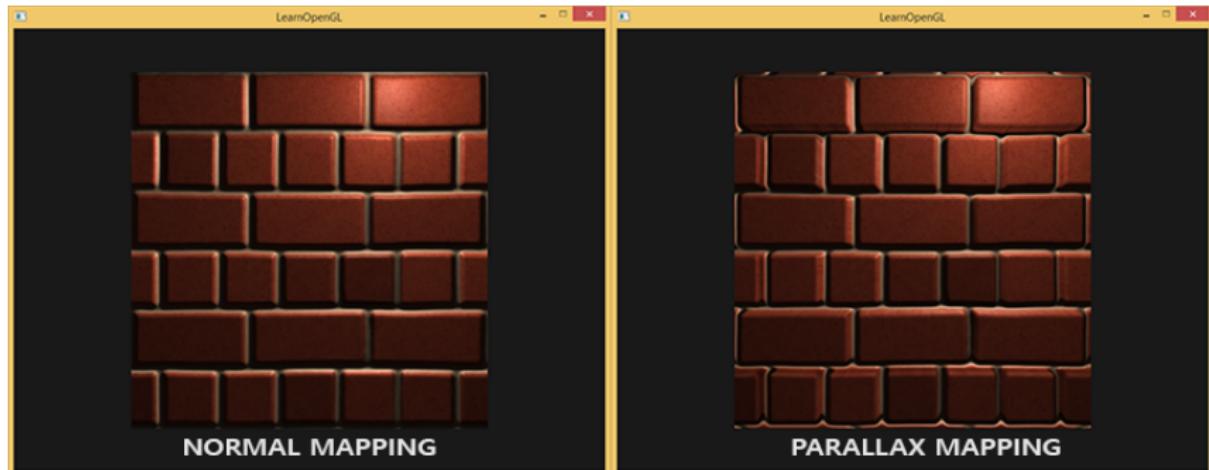
Tekstuurit

Muita tekstuuriinnoin sovelluksia

## Gloss mapping



## Parallax mapping



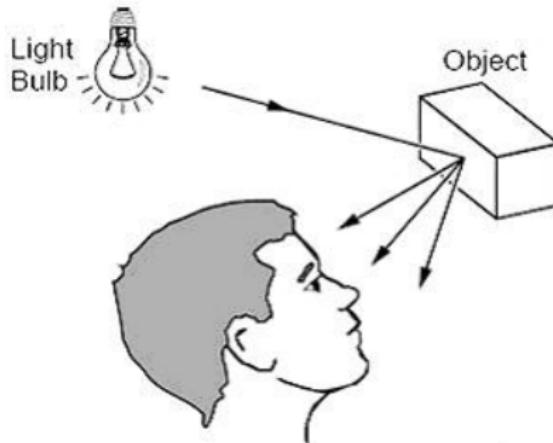
<http://www.learnopengl.com/#!Advanced-Lighting/Parallax-Mapping>

## Harjoituksia

- (Ex 8) Kirjoita ohjelma joka piirtää pyörivän teksturoidun kuution.
- (Ex 9) Lisää edelliseen normal mapping ja hyödynnä sitä Lambertin valaistusmallissa. Lambertin valaistusmallissa valon intensiteetti kullekin värikanavalle  $I_D = \max(\mathbf{n} \cdot \mathbf{d}_{light}, 0)c_DL$ , jossa  $\mathbf{n}$  on pinnan normaali,  $\mathbf{d}_{light}$  on valon suuntavektori,  $c$  on pinnan heijastuskerroin (väri) ja  $I_L$  on valon intensiteetti.

## Taustaa

- Silmä havaitsee valoa jonka avulla ihminen muodostaa käsityksensä ympäröivän maailman geometriasta.
  - Silmä ei havaitse esineen väriä, muotoa jne.
  - Aivot tuottavat em. informaation silmän havaitsemasta kuvasta.
- Havaittava valo on usein heijastunut jostain kappaleesta.



- ⇒ Realistiselta näyttävän grafiikan tuottamiseksi tarvitaan valaistusmalli.
- Erilaisille materiaaleille tarvitaan erilaisia valaistusmalleja (tai ainakin parametrisoitavia malleja).

## Erilaisia valaistusmalleja

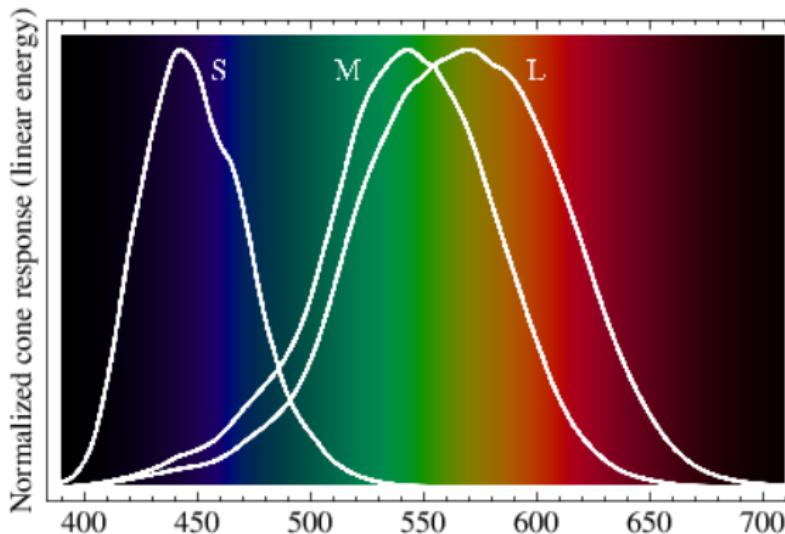
- Ambient
  - Valaisee kaikkia pinnan pisteitä samalla voimakkuudella riippumatta pinnan normaalista.
  - Valolla ei ole suuntaa.
- Lambert
  - Olettaa pinnan diffuusiksi.
  - Pinnan kirkkauteen vaikuttaa valon suunnan ja pinnan normaalin välinen kulma.
- Phong
  - Myös peiliheijastus huomioidaan valaistuksessa.
- Oren-Nayar
  - "Realistiksempi vaihtoehto Lambertin mallille."
- Heidrich-Seidel
  - "Anisotrooppinen Phong"
- <http://www.sunandblackcat.com/tipFullView.php?l=eng&topicid=30>

## Esimerkki

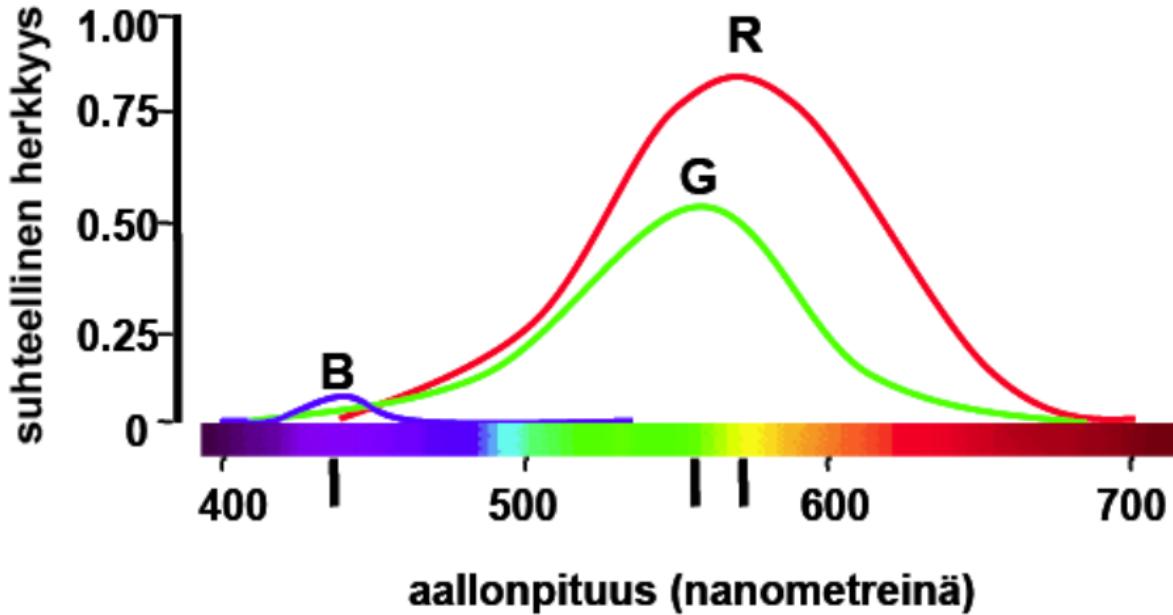


# Väri

- Ihmissilmä havaitsee valoa jonka aallonpituuus on karkeasti 400...700 nm.
- Silmässä on hämäränäköön erikoistuneiden sauvasolujen lisäksi kolmenlaisia tappisoluja:
  - L-tyyppin tappisolut reagoivat punaiseen valoon ja ovat herkimmillään noin 650 nm aallonpituuudella, 63%.
  - M-tyyppin tappisolut reagoivat vihreään valoon ja ovat herkimmillään noin 530 nm aallonpituuudella, 36%.
  - S-tyyppin tappisolut reagoivat siniseen valoon ja ovat herkimmillään noin 460 nm aallonpituuudella, 1%.



## Silmän herkkyyys eri aallonpituuksille



## Metameria

- Metamerialla tarkoitetaan ilmiötä, jossa kaksi objektiota näyttää yhdessä valossa samanvärisiltä, mutta toisessa valaistuksessa erivärisiltä.
- Ilmiö johtuu siitä, että tappisolut ovat herkkiä laajalle alueelle spektriä.
- Esimerkiksi valaistaessa valkoista ja sinistä kappaletta sinisellä valolla, näyttävät kumpikin sinisiltä.
- Yleensä tietokonegrafiikassa jätetään tämä ilmiö huomioimatta ja käytetään yksinkertaisempia malleja joissa oletetaan tappisolujen herkistyneen vain yhdelle aallonpituuudelle.

## RGB

- Kun oletetaan että erityyppiset tappisolut ovat herkistyneet kukaan vain yhdelle aallonpituuudelle, voidaan väri kuvata kolmella toisistaan riippumattomalla parametrilla.
- Valaistuksen laskennassa käytetään usein punaisen (R), vihreän (G) ja sinisen (B) intensiteettejä kuvaamaan väriä.
- Tällaisen parametroinnin etuna on se, että kukaan värikanava voidaan laskennassa käsitellä yleensä erillisenä. Näin toimitaan jatkossa, kaavat esitetään vain yhdelle kanavalle.
- Valaistuksen laskennassa käytetään additiivista mallinnusta, jossa eri valonlähteiden vaikutus summataan lopputulokseen.

## Suunnattu valo

- Approksimoi äärettömän kaukana olevaa pistemäistä valonlähdettä.
- Valon intensiteetti on vakio kaikkialla.
- Valolla on suunta.
- Parametrointi:

$\mathbf{d}_{light}$  = Valon suunta

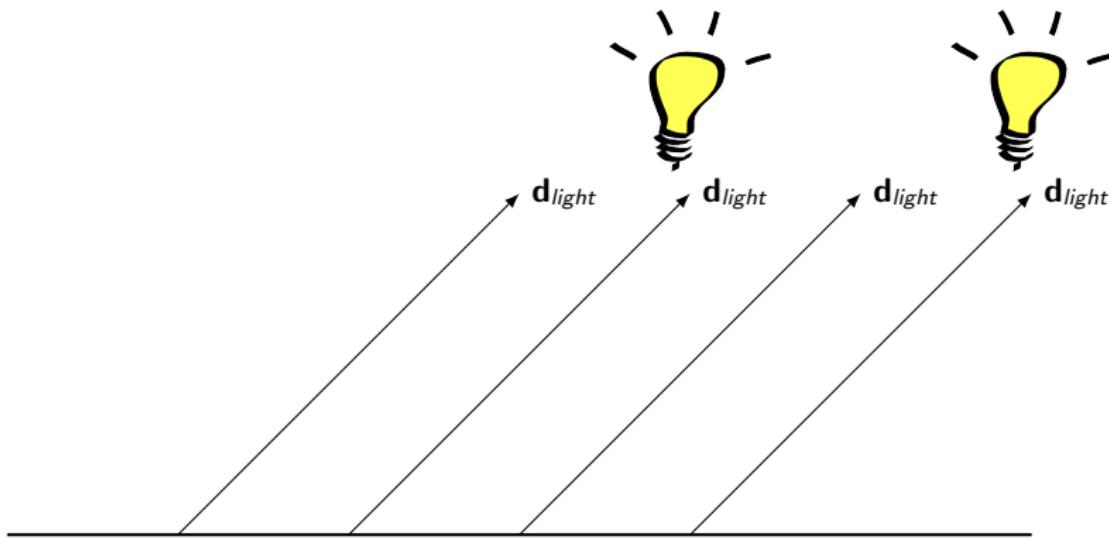
$c$  = Valon kirkkaus (yhdellä kanavalla)

- Intensiteetti ja suunta:

$$I = c$$

$$\mathbf{d}_{light} = \mathbf{d}_{light}$$

## Suunnatun valon parametrit



## Pistemäinen valo

- Pistemäinen valonlähde säteilee kaikkiin suuntiin samalla voimakkuudella.
- Valon intensiteetti vähenee verrannollisesti etäisyyden neliöön.

- Parametrointi:

$\mathbf{x}_{light}$  = Valon positio

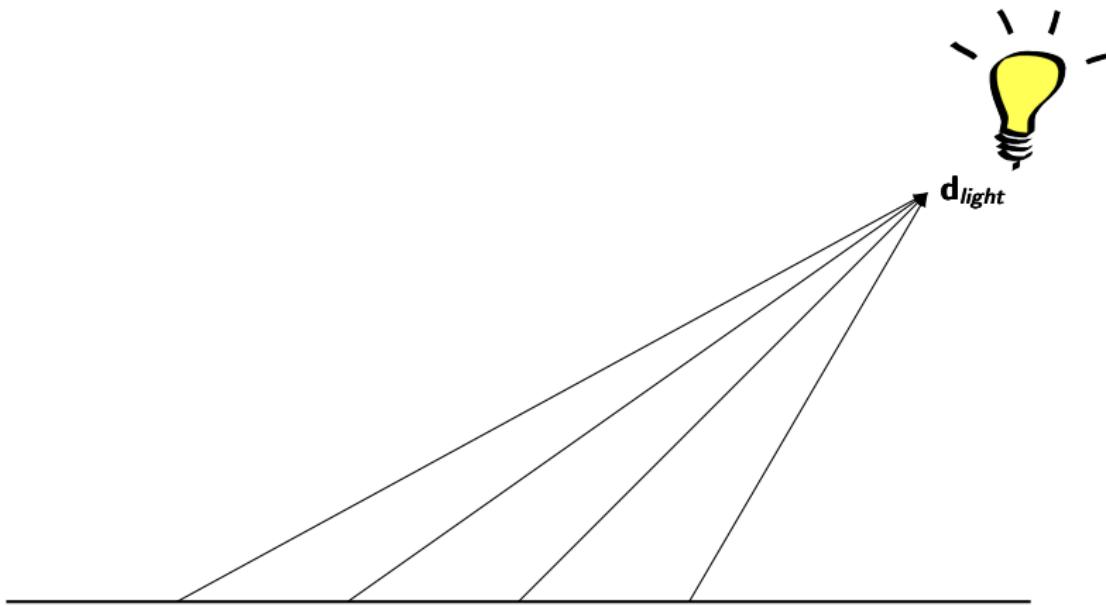
$c$  = Valon kirkkaus (yhdellä kanavalla)

- Intensiteetti ja suunta:

$$I(\mathbf{x}) = \frac{c}{C_1 + C_2 \|\mathbf{x}_{light} - \mathbf{x}\| + C_3 \|\mathbf{x}_{light} - \mathbf{x}\|^2}.$$

$$\mathbf{d}_{light}(\mathbf{x}) = \mathbf{x}_{light} - \mathbf{x}$$

## Pistemäisen valon parametrit



# Spotti

- Kuten pistemäinen valo, mutta intensiteetti riippuu myös kulmasta.
- Kulmariippuvuus on myös parametroitavissa.
- Parametrointi:

$\mathbf{x}_{light}$  = Valon positio

$\mathbf{d}_{dir}$  = Valon keskiakselin suunta

$c$  = Valon kirkkaus (yhdellä kanavalla)

$f(\mathbf{d}_{light}, \mathbf{d}_{dir})$  = Valon intensiteetin suuntariippuvuus. Esimerkiksi

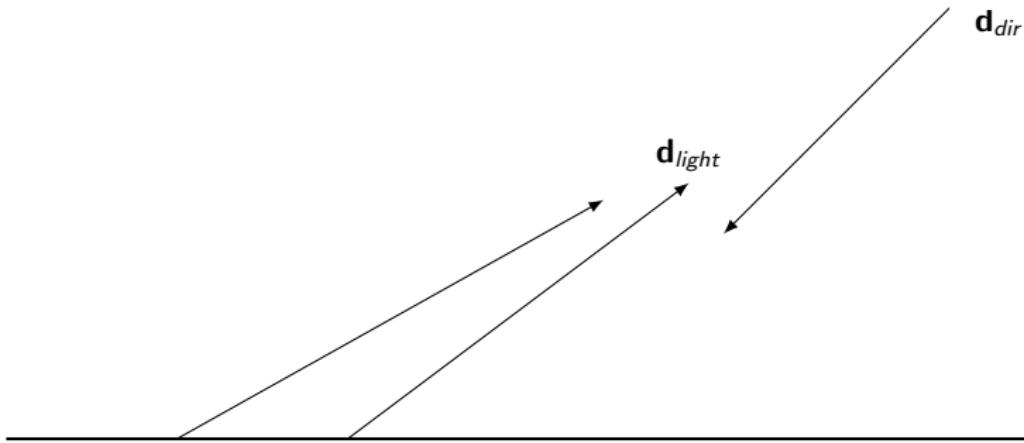
$$f(\mathbf{d}_{light}, \mathbf{d}_{dir}) = \max(-\mathbf{d}_{light} \cdot \mathbf{d}_{dir}, 0)^\alpha.$$

- Intensiteetti ja suunta:

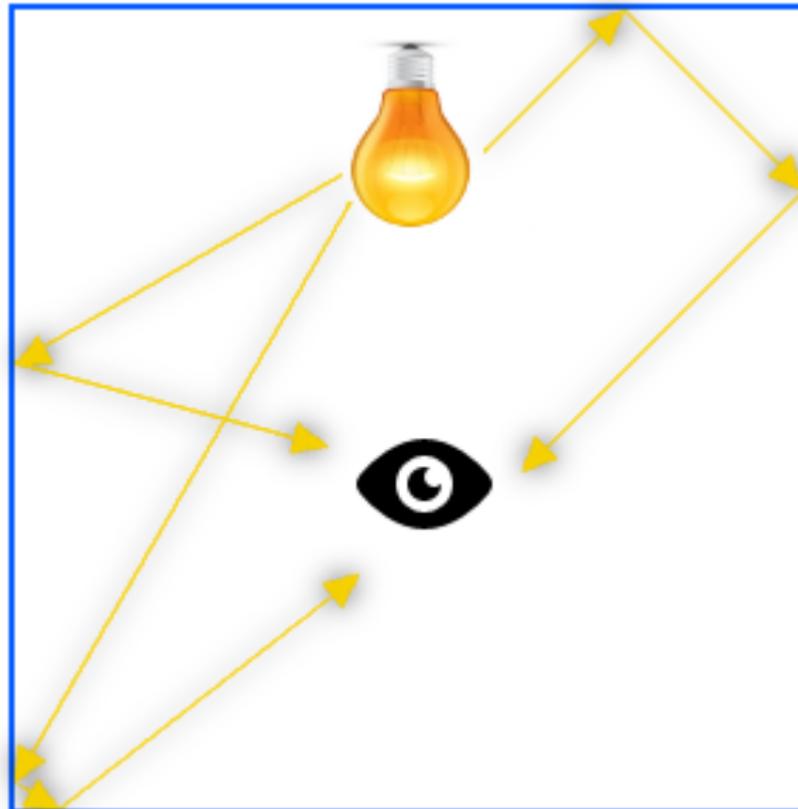
$$I(\mathbf{x}) = \frac{c f(\mathbf{d}_{light}, \mathbf{d}_{dir})}{C_1 + C_2 \|\mathbf{x}_{light} - \mathbf{x}\| + C_3 \|\mathbf{x}_{light} - \mathbf{x}\|^2},$$

$$\mathbf{d}_{light}(\mathbf{x}) = \mathbf{x}_{light} - \mathbf{x}$$

## Spotin parametrit

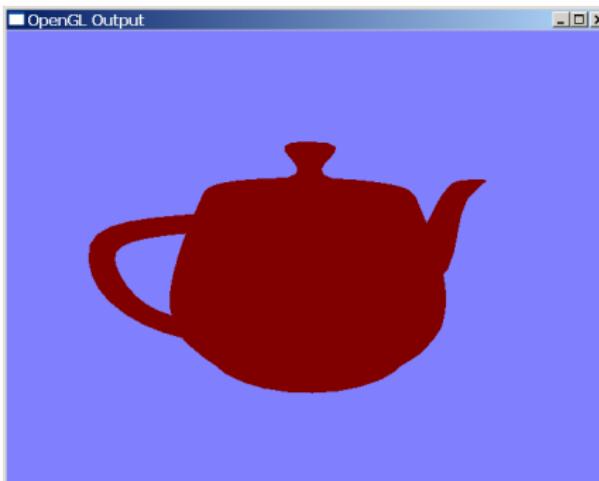
 $d_{dir}$  $d_{light}$ 

## Periaate

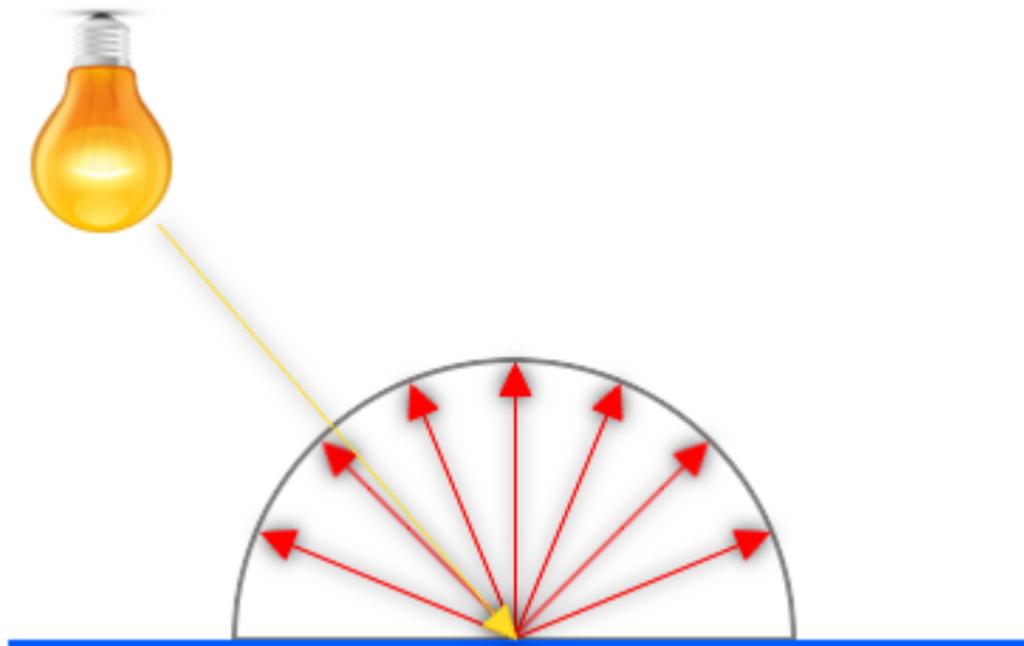


## Yleistä

- "Yleisvalo", käytetään yleensä osana muita valaistusmalleja.
- Ilman ambient-termiä muut valaistusmallit generoivat usein liian suuren kontrastin kuvaan.
- Valaisee kaikkia pinnan pisteitä samalla voimakkuudella riippumatta pinnan normaalista.
- Valonlähteellä ei ole paikka tai suuntaa.
- Intensiteetti kullekin värikanavalle  $I_A = c_A I_{L,A}$ .



## Periaate

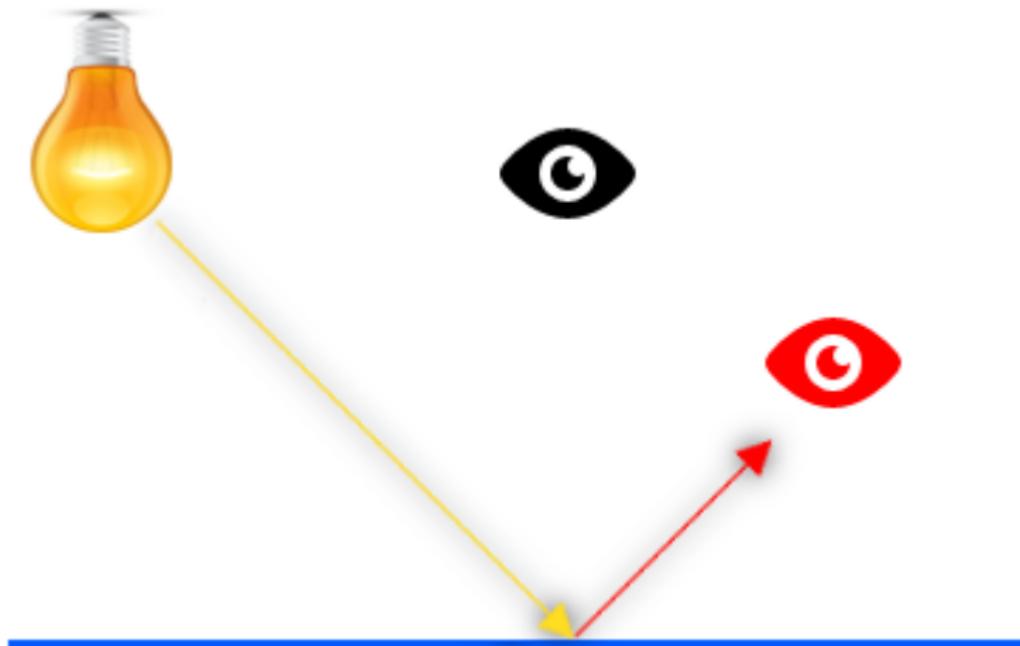


## Yleistä

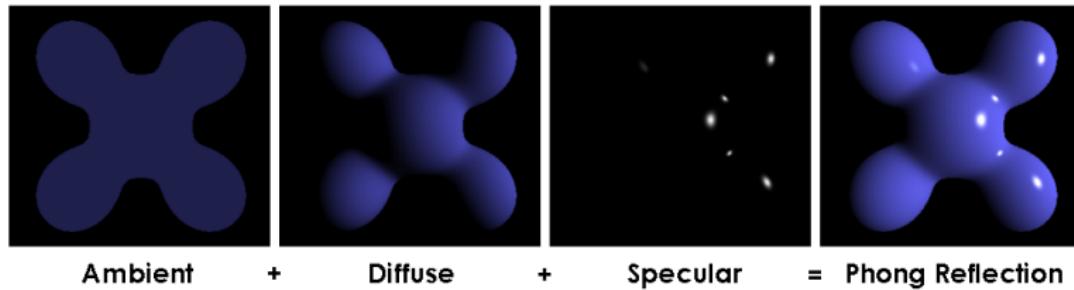
- Olettaa pinnan diffuusiksi.
- Pinnan kirkkauteen vaikuttaa valon suunnan ja pinnan normaalinvälinen kulma.
- Intensiteetti kullekin värikanavalle  $I_D = (\mathbf{n} \cdot \mathbf{d}_{light}) c_D I_{L,D}$ .
- Usein käytetään yhdessä ambient-termin kanssa, jolloin  $I_{A,D} = I_A + I_D$ .



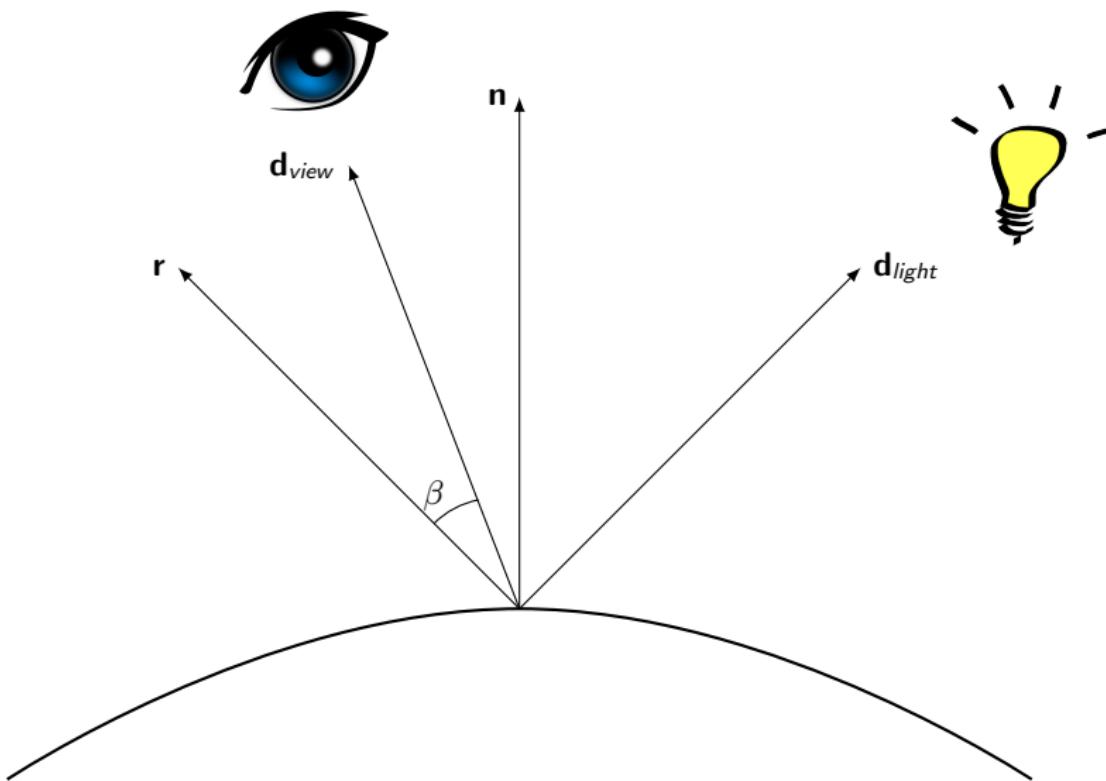
## Periaate



## Termit



# Vektorit

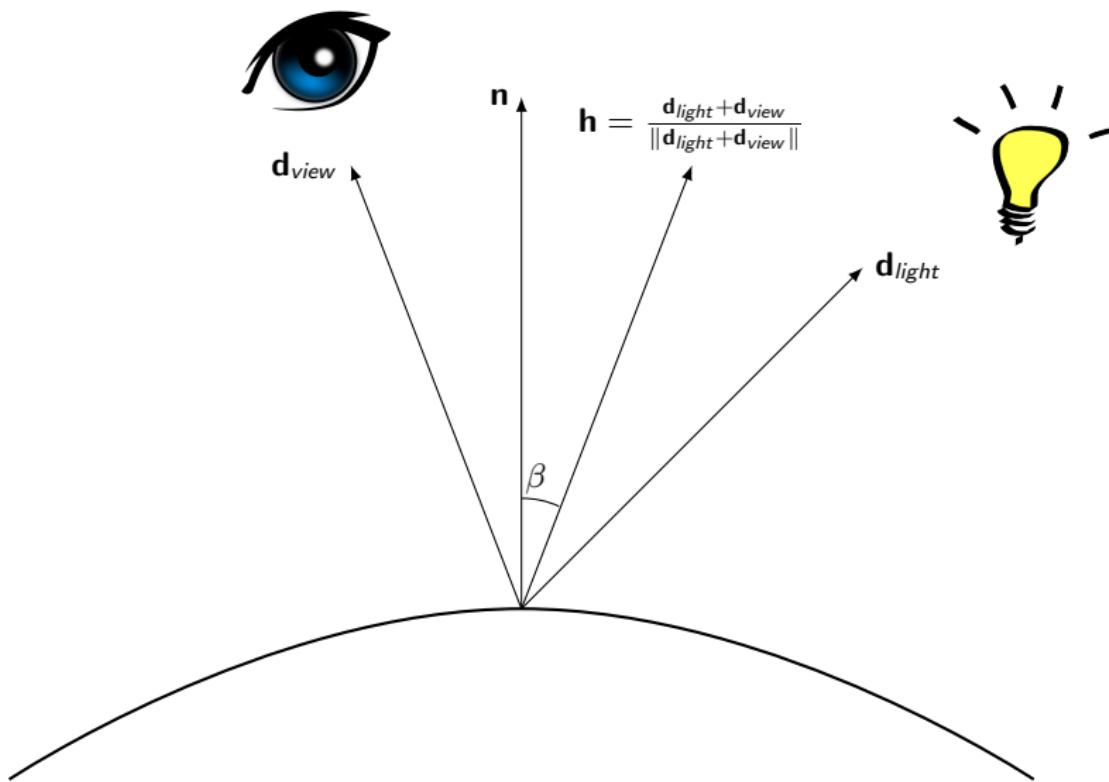


## Yleistä

- Huomioi myös peiliheijastuksen.
- Intensiteetti kullekin värikanavalle  $I_P = I_A + I_D + (\underbrace{\mathbf{r} \cdot \mathbf{d}_{view}}_{=\cos \beta})^\alpha c_S I_{L,S}$ , jossa  $\mathbf{r} = 2(\mathbf{d}_{light} \cdot \mathbf{n})\mathbf{n} - \mathbf{d}_{light}$ .



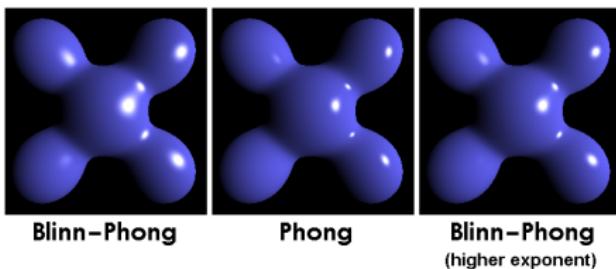
# Blinn-Phong vektorit



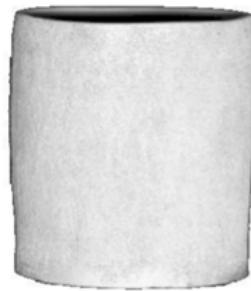
## Blinn-Phong

- Saadaan optimoitua nopeammaksi kuin Phong-malli, jos käytössä on ortografinen projektio ja valonlähde ovat kaukana pinnasta.
  - Tällöin  $\mathbf{h}$  on vakio riippumatta pinnan kaareutumisesta.
- Intensiteetti kullekin värikanavalle  $I_P = I_A + I_D + (\underbrace{\mathbf{n} \cdot \mathbf{h}}_{=\cos \beta})^\alpha c_S I_{L,S}$ , jossa

$$\mathbf{h} = \frac{\mathbf{d}_{light} + \mathbf{d}_{view}}{\|\mathbf{d}_{light} + \mathbf{d}_{view}\|}.$$



## Lambertin malli ei ole täydellinen



Real Image



Lambertian Model

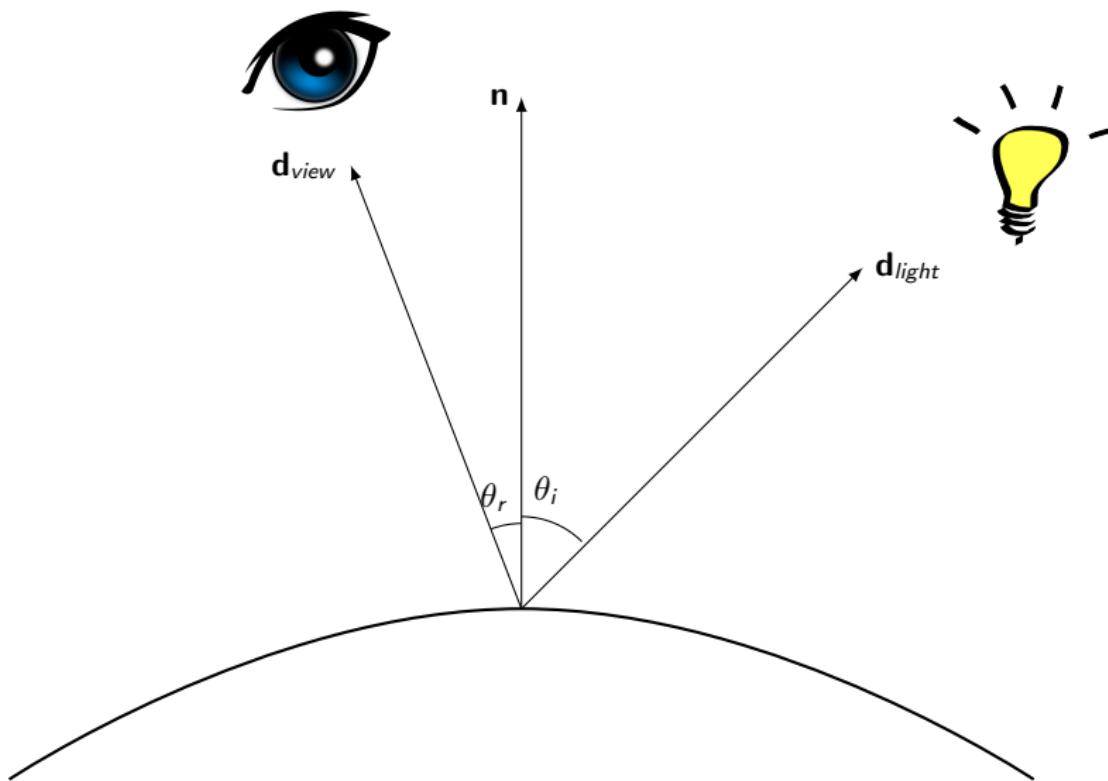


Oren-Nayar Model

## Malli

- Lambertin malli antaa liian tumman heijastuksen valoisen alueen reunille.
- Kuun oletettiin olevan diffuusi, mutta havainnot eivät sopineet Lambertin malliin.
- Malli olettaa pinnan koostuvan mikrourista, joiden seinämät toteuttavat Lambertin mallin.
- Huomattavasti monimutkaisempi kuin Lambertin malli.

## Mallin vektorit



# Malli

- $\sigma^2$  = karheuden varianssi. Jos  $\sigma^2 = 0$ , saadaan Lambertin malli.
- $\alpha = \max(\theta_r, \theta_i)$
- $\beta = \min(\theta_r, \theta_i)$
- $A = 1 - \frac{1}{2} \frac{\sigma^2}{\sigma^2 + 0.57}$
- $B = 0.45 \frac{\sigma^2}{\sigma^2 + 0.09}$

$$L_D = \frac{c_D I_{L,D}}{\pi} \cos \theta_i (A + B \max(0, \cos(\theta_i - \theta_r)) \sin \alpha \tan \beta)$$

## Materiaalien määrittely

- Tuki OpenGL:n materiaaleille vanhentui versiossa 3.0.
- Samankaltaista parametriointia voidaan kuitenkin käyttää edelleen myös shadereihin perustuvien valaistuslaskentojen kanssa.
- Periaatteessa kyse on vain valaistusmallien parametrioinnin yhtenäistämisestä.
  - Phongin valaistusmallia parametroimalla saadaan periaatteessa muut (ambient ja Lambert) mallit erikoistapauksena.
  - Ohjelmassa on kuitenkin hyvä käyttää mahdollisimman yksinkertaista mallia laskennan määrään minimoimiseksi.
- Myös OBJ-tiedostotyypissä on tuki materiaaleille, tutkitaan jatkossa sitä.

## OBJ-materiaalikirjasto

- <http://paulbourke.net/dataformats/mtl/>
- Tiedoston laajenne .MTL.
- OBJ-tiedostossa materiaalikirjasto luetaan komennolla `mtllib tiedostonnimi`.
- Materiaalikirjastosta valitaan materiaali käyttöön komennolla `usemtl matriaalinnimi`.
- Parametrointi Phong valaistusmallille, teksturoinnille, bumb-mappingille (josta saadaan laskettua normal mapping) jne.

## Esimerkki materiaaliparameterista

```
# define a material named 'Colored'  
newmtl Colored  
Ka 1.000 1.000 1.000 #ambient  
Kd 1.000 1.000 1.000 #diffuse  
Ks 0.100 0.100 0.100 #specular  
Ns 10.0           #phong exponent (0.0-1000.0)
```

## Esimerkki teksturoinnista

```
newmtl Textured
map_Ka lenna.tga      # the ambient texture map
map_Kd lenna.tga      # the diffuse texture map (most of the time, it
                      # will be the same as the ambient texture map)
map_Ks lenna.tga      # specular color texture map
map_Ns lenna_spec.tga# specular highlight component
map_d lenna_alpha.tga# the alpha texture map
map_bump lenna_bump.tga #some implementations use 'map_bump' instead
                      # of 'bump' below
```

## Harjoituksia

(Ex 10) Toteuta materiaalituki OBJ-kirjastoosi.

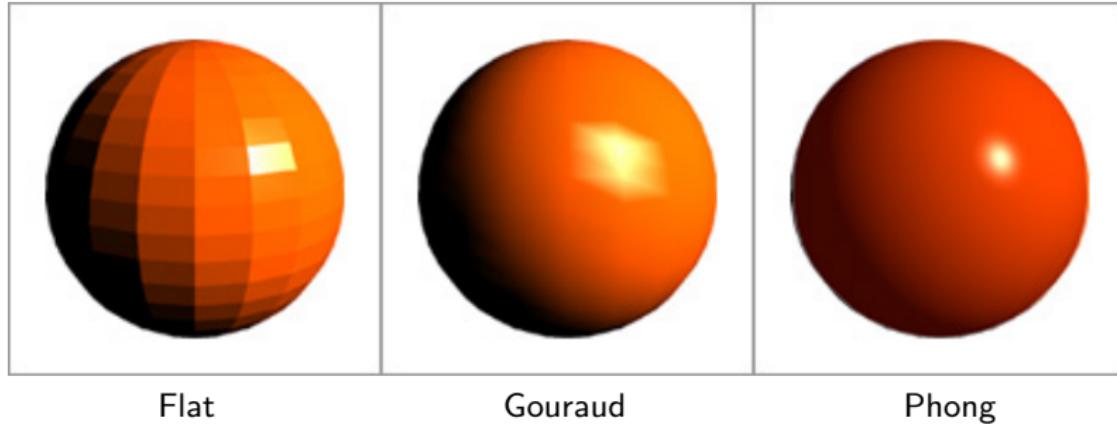
## Taustaa

- Varjostus (shading) on prosessi jolla primitiivin väritys muodostetaan.
- Kun tekstuuriointi jätetään huomioimatta, on piirron lähtödatan yksikkönä primitiivi.
- Primitiivin geometria on puolestaan määritelty kärkipisteidensä (verteksien) ja mahdollisesti primitiivin normaalilin avulla.
- Primitiivin geometriasta ja valaistustiedoista voidaan laskea väritys koko piirrettävän primitiivin alalle.
  - Väritys voidaan toteuttaa usealla tavalla:
  - Väritetään koko primitiivi samalla värellä (flat shading), valaistus lasketaan esimerkiksi primitiivin normaalilia käyttäen.
  - Lasketaan primitiivin kulmapisteille väriarvot ja interpoloidaan lineaarisesti kullekin piirrettävälle pikselille väriarvo (Gouraud shading).
  - Interpoloidaan normaalitietoa primitiivin yli kärkipisteiden normaalien avulla ja lasketaan väriarvo pikseleittäin (Phong shading<sup>7</sup>).

---

<sup>7</sup>Älä sekoita tästä Phongin valaistusmalliin!

## Vertailu



## Flat shading

- Koko primitiivi väritetään samalla väriillä.
- Mikäli valaistus lasketaan, käytetään sen laskennassa yleensä primitiivin normaalia.
- + Nopea
- + Tarvittava normaali on helppo laskea jälkikäteenkin mikäli se ei tule mallin mukana.
- Ei kovin realistisen näköinen.

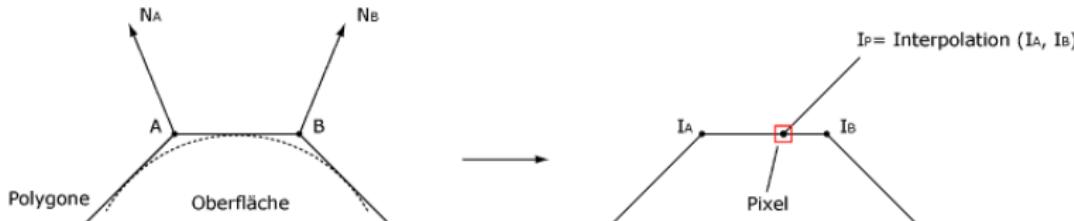
# Gouraud shading

<https://www.youtube.com/watch?v=ULMxym7PfTY>

Schattierung durch Interpolation

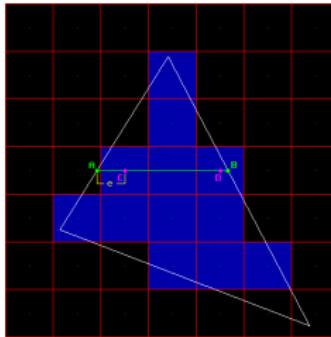
$I_A$  = Schattierungsfunktion ( $N_A$ )  
 $I_B$  = Schattierungsfunktion ( $N_B$ )

Gouraud-Shading



## Gouraud shading – laskenta

- Primitiivin kulmapisteiden väri interpoloidaan bilinearisesti primitiivin yli.
- Toteutus on helppo (esimerkiksi näin):



- ① Lasketaan kulmapisteiden väriarvot
- ② Interpoloidaan väriarvot jokaiselle vaakarasteririville pitkin primitiivin reunaa lineaarisesti.
- ③ Interpoloidaan väriarvot vaakariveittäin.

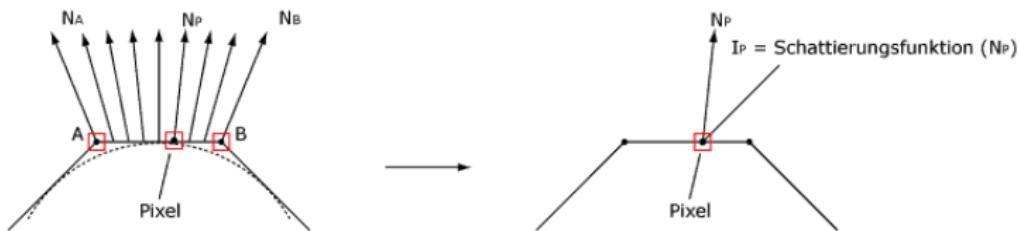
- + Suhteellisen realistinen
- + Melko kevyt toteuttaa
- Ei toimi kovin hyvin Phongin valaistusmallin kanssa.

# Phong shading

<https://www.youtube.com/watch?v=Seo5eo3GUQo>

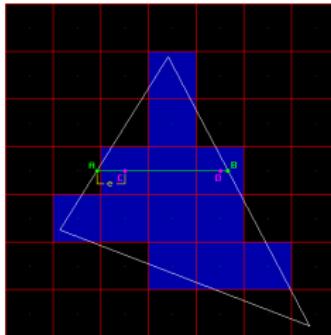
Schattierung durch Interpolation

## Phong-Shading



## Phong shading – laskenta

- Primitiivin kulmapisteiden normaalivektorit interpoloidaan primitiivin yli.
- Toteutus on helppo (esimerkiksi näin):



- ❶ Interpoloidaan normaalit jokaiselle vaakarasteririville pitkin primitiivin reunaa lineaarisesti.
- ❷ Interpoloidaan normaalit vaakariveittäin.
- ❸ Lasketaan valaistus kullekin pikselille.

+ Realistinen

- Laskennallisesti raskaampi kuin flat tai Gouraud shading.

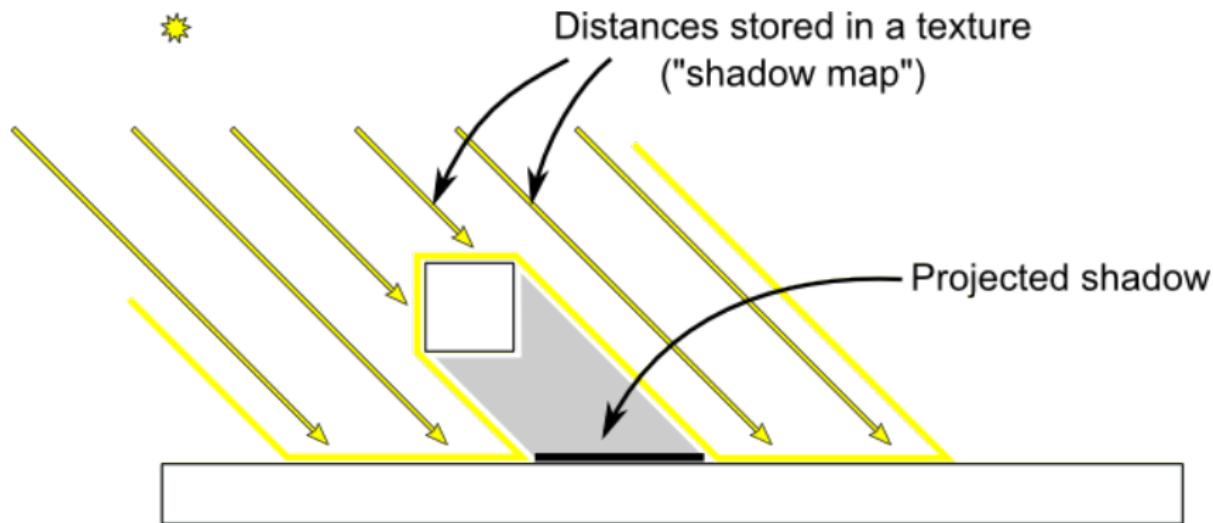
## OpenGL Tutoriaali

<http://ogldev.atspace.co.uk/>

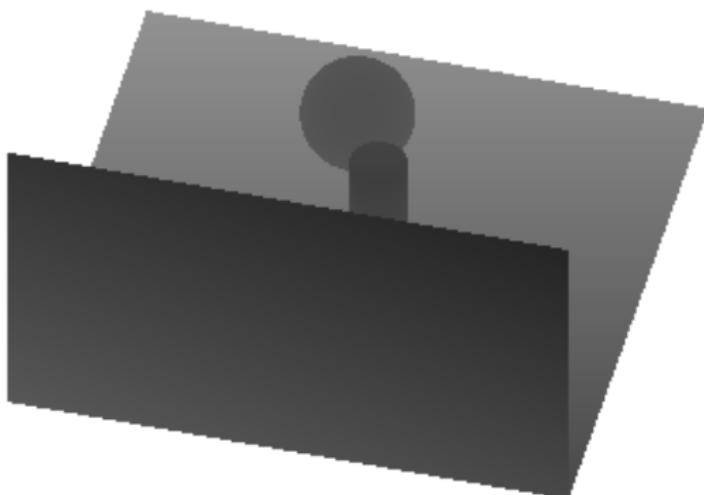
## Idea

- ① Piirretään kenttä tekstuuriin s.e kamera on valonlähteen paikalla.
  - Talletetaan lähimmän pinnan etäisyys kameraan.
  - Vain Z-puskuri, ei pinnan väritystä.
- ② Piirretään kenttä käyttäen normaalia kameraa.
  - Lasketaan pisteen *uv*-koordinaatit.
  - Piste on varjossa jos sen etäisyys valonlähteeseen on suurempi kuin edellä talletettu etäisyys valonlähteeseen.
  - Muuten piste on valaistu.
- Periaatteessa tällä algoritmilla saadaan aikaan reaalialkaiset varjot.
- Käytännön soveltamisessa on monia ongelmia.
- <http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/>

## Varjon syvyyskartan laskenta



## Varjon syvyyskartta



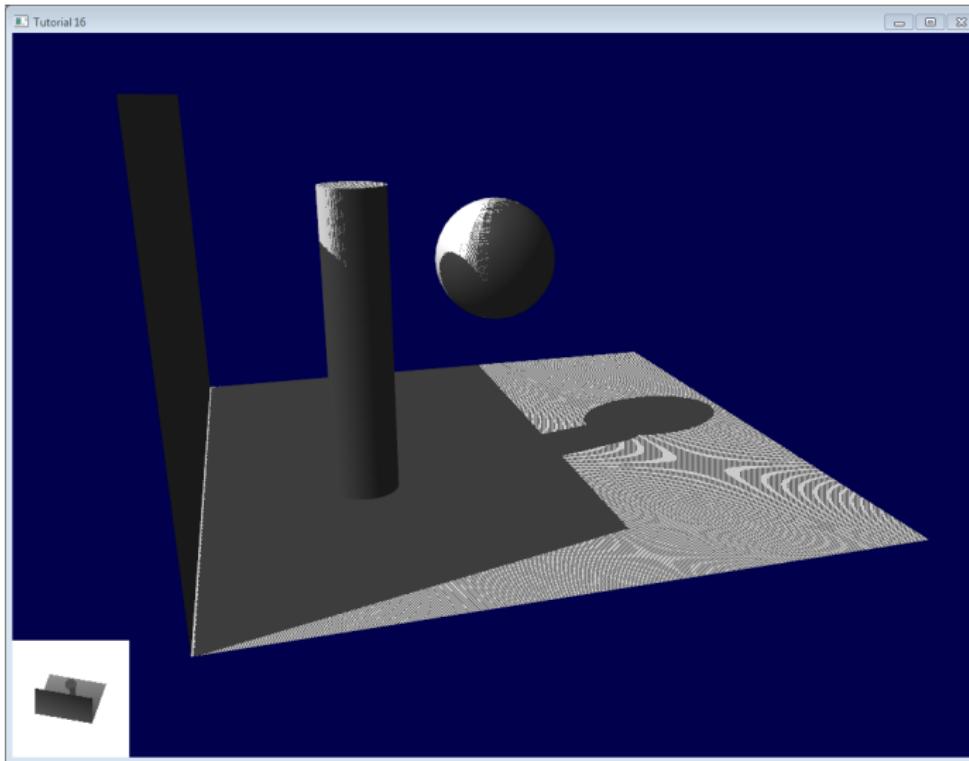
$$uv \sim xyz$$

- Idea:

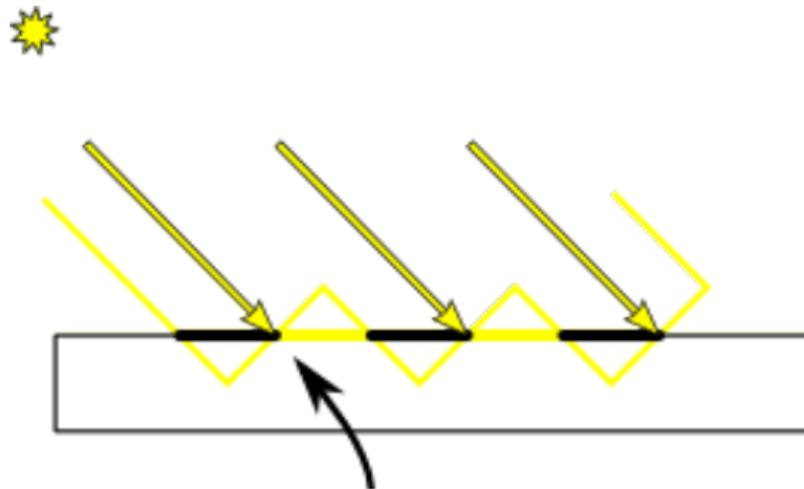
- Vertexshaderissa lasketaan verteksin positio
- näytön koordinaatistossa ja
- varjon koordinaatistossa (kamera valonlähteessä).
- Tämän jälkeen muunnetaan varjokoordinaatit  $([-1, 1])$   $uv$ -koordinaateiksi  $([0, 1])$ .

- $\text{biasMatrix} = \begin{pmatrix} 0.5 & 0.0 & 0.0 & 0.5 \\ 0.0 & 0.5 & 0.0 & 0.5 \\ 0.0 & 0.0 & 0.0 & 0.5 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$ .

## Tulos



## Miksi ei onnistunut?



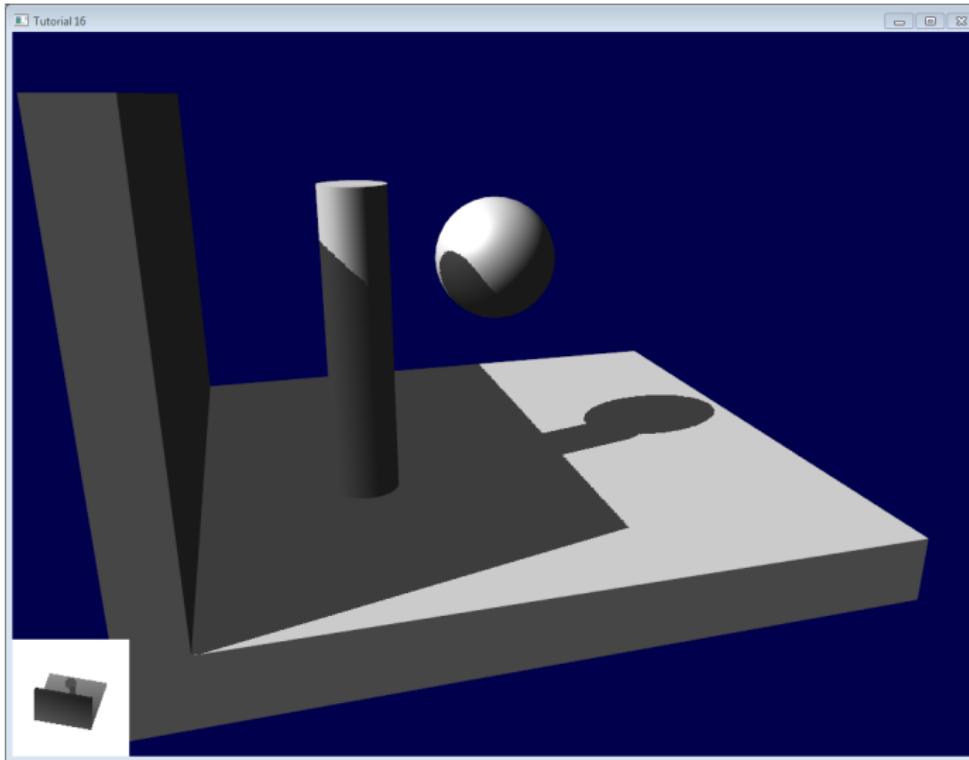
Lightmap pixels

## Korjaus

- "Siirretään valonlähdettä hieman taaksepäin"
- Koodissa tämä on helppo tehdä esimerkiksi fragmentshaderissa:

```
1 uniform sampler2DShadow shadowMap;
2 ...
3 visibility=texture(shadowMap, vec3(ShadowCoord.xy,
4     ShadowCoord.z/ShadowCoord.w-depthEps));
```

## Korjattu varjostus



## Renderöinti tekstuuriin

```
1 void InitDepthTexture() {
2     depthProgramID = LoadShaders( "DepthRTT.vs" , "DepthRTT.fs" );
3     depthMatrixID = glGetUniformLocation(depthProgramID , "depthMVP" );
4
5     glGenFramebuffers(1 , &FramebufferName);
6     glBindFramebuffer(GL_FRAMEBUFFER , FramebufferName);
7
8     glGenTextures(1 , &depthTexture);
9     glBindTexture(GL_TEXTURE_2D , depthTexture);
10    glTexImage2D(GL_TEXTURE_2D , 0 , GL_DEPTH_COMPONENT16 ,
11                 ShadowTextureSize , ShadowTextureSize , 0 ,
12                 GL_DEPTH_COMPONENT , GL_FLOAT , 0);
```

## Renderointi tekstuuriin

```
1 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
2 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
3 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
4     GL_CLAMP_TO_EDGE);
5 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
6     GL_CLAMP_TO_EDGE);
7 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_COMPARE_FUNC,
8     GL_EQUAL);
9 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_COMPARE_MODE,
10    GL_COMPARE_R_TO_TEXTURE);

11
12 glBindFramebuffer(GL_FRAMEBUFFER, GL_DEPTH_ATTACHMENT,
13    depthTexture, 0);
14 glDrawBuffer(GL_NONE);
15 if(glCheckFramebufferStatus(GL_FRAMEBUFFER) !=
16    GL_FRAMEBUFFER_COMPLETE) {
17     throw "InitDepthTexture failed";
18 }
19 }
```

## Renderöinti tekstuuriin

```
1 void RenderToTexture() {
2     glBindFramebuffer(GL_FRAMEBUFFER, FramebufferName);
3     glViewport(0,0,ShadowTextureSize,ShadowTextureSize);
4
5     glEnable(GL_CULL_FACE);
6     glCullFace(GL_BACK);
```

## Renderöinti tekstuuriin

```
1 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
2 glUseProgram(depthProgramID);
3
4 alpha+=0.01;
5
6 lightInvDir = glm::vec3(...); //Valon suunta
7 depthProjectionMatrix = glm::ortho<float>(-10,10,-10,10,-10,20);
8 depthViewMatrix = glm::lookAt(lightInvDir, glm::vec3(0,0,0),
9     glm::vec3(0,1,0));
10
11 depthModelMatrix = glm::mat4(1.0);
12 depthMVP=depthProjectionMatrix*depthViewMatrix*depthModelMatrix;
13
14 glUniformMatrix4fv(depthMatrixID, 1, GL_FALSE, &depthMVP[0][0]);
```

## Renderöinti tekstuuriin

```
1  glEnableVertexAttribArray(0);
2  glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
3  glVertexAttribPointer(...)
4  glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, elementbuffer);
5  glDrawElements(...);
6
7  glDisableVertexAttribArray(0);
8 }
```

## Renderöinti tekstuuriin

```
1 #version 330 core
2
3 layout(location = 0) in vec3 vertexPosition_modelspace;
4 uniform mat4 depthMVP;
5 void main(){
6     gl_Position =  depthMVP * vec4(vertexPosition_modelspace,1);
7 }
```

## Renderöinti tekstuuriin

```
1 #version 330 core
2
3 layout(location = 0) out float fragmentdepth;
4
5 void main(){
6     fragmentdepth = gl_FragCoord.z;
7 }
```

## Renderöinti näytölle

```
1 void renderer::Init(GLFWwindow* w) {
2     window=w;
3     glClearColor(0.1f, 0.1f, 0.1f, 1.0f);
4     glEnable(GL_DEPTH_TEST);
5     glDepthFunc(GL_LESS);
6     glEnable(GL_CULL_FACE);
7
8     glGenVertexArrays(1, &VertexArrayID);
9     glBindVertexArray(VertexArrayID);
```

## Renderöinti näytölle

```
1 Texture = loadDDS("uvmap.DDS");
2 loadOBJ("room_thickwalls.obj", vertices, uvs, normals);
3 indexVBO(vertices, uvs, normals, indices, indexed_vertices,
4     indexedUvs, indexed_normals);
5
6 glGenBuffers(1, &vertexbuffer);
7 glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
8 glBufferData(GL_ARRAY_BUFFER, indexed_vertices.size()
9     * sizeof(glm::vec3), &indexed_vertices[0], GL_STATIC_DRAW);
```

## Renderöinti näytölle

```
1 glGenBuffers(1, &uvbuffer);
2 glBindBuffer(GL_ARRAY_BUFFER, uvbuffer);
3 glBufferData(GL_ARRAY_BUFFER, indexedUvs.size()
4     * sizeof(glm::vec2), &indexedUvs[0], GL_STATIC_DRAW);
5 glGenBuffers(1, &normalbuffer);
6 glBindBuffer(GL_ARRAY_BUFFER, normalbuffer);
7 glBufferData(GL_ARRAY_BUFFER, indexedNormals.size()
8     * sizeof(glm::vec3), &indexedNormals[0], GL_STATIC_DRAW);
9 glGenBuffers(1, &elementbuffer);
10 glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, elementbuffer);
11 glBufferData(GL_ELEMENT_ARRAY_BUFFER, indices.size()
12     * sizeof(unsigned short), &indices[0], GL_STATIC_DRAW);
```

## Renderöinti näytölle

```
1 programID = LoadShaders( "ShadowMapping . vs" , "ShadowMapping . fs" ←
2   );
3 TextureID = glGetUniformLocation(programID , "myTextureSampler" );
4 MatrixID = glGetUniformLocation(programID , "MVP" );
5 DepthBiasID = glGetUniformLocation(programID , "DepthBiasMVP" );
6 ShadowMapID = glGetUniformLocation(programID , "shadowMap" );
7 DepthEpsID = glGetUniformLocation(programID , "depthEps" );
8 LightDirID = glGetUniformLocation(programID , "lightDir" );
9
10 }  
InitDepthTexture();
```

## Renderöinti näytölle

```
1 void renderer::Render(void) {
2     RenderToTexture();
3
4     glBindFramebuffer(GL_FRAMEBUFFER, 0);
5     glViewport(0,0,1024,768);
6     glEnable(GL_CULL_FACE);
7     glCullFace(GL_BACK);
8     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
9     glUseProgram(programID);
10
11    glm::mat4 ProjectionMatrix =
12        glm::perspective(1.0f, 4.0f / 3.0f, 0.1f, 100.0f);
13    glm::mat4 ViewMatrix =glm::lookAt(
14        glm::vec3(14,6,4), glm::vec3(0,1,0), glm::vec3(0,1,0));
15    glm::mat4 ModelMatrix = glm::mat4(1.0);
16    glm::mat4 MVP = ProjectionMatrix * ViewMatrix * ModelMatrix;
17
18    glm::mat4 biasMatrix(0.5, 0.0, 0.0, 0.0,
19                        0.0, 0.5, 0.0, 0.0,
20                        0.0, 0.0, 0.5, 0.0,
21                        0.5, 0.5, 0.5, 1.0);
22
23    glm::mat4 depthBiasMVP = biasMatrix*depthMVP;
```

## Renderöinti näytölle

```
1 glUniformMatrix4fv(MatrixID, 1, GL_FALSE, &MVP[0][0]);
2 glUniformMatrix4fv(DepthBiasID, 1, GL_FALSE,
3     &depthBiasMVP[0][0]);
4
5 glActiveTexture(GL_TEXTURE0);
6 glBindTexture(GL_TEXTURE_2D, Texture);
7 glUniform1i(TextureID, 0);
8 glUniform1f(DepthEpsID, depthEps);
9 glUniform3fv(LightDirID, 1, glm::value_ptr(lightInvDir));
10 glActiveTexture(GL_TEXTURE1);
11 glBindTexture(GL_TEXTURE_2D, depthTexture);
12 glUniform1i(ShadowMapID, 1);
```

## Renderöinti näytölle

```
1  glEnableVertexAttribArray(0);
2  glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
3  glVertexAttribPointer(...);
4  glEnableVertexAttribArray(1);
5  glBindBuffer(GL_ARRAY_BUFFER, uvbuffer);
6  glVertexAttribPointer(...);
7  glEnableVertexAttribArray(2);
8  glBindBuffer(GL_ARRAY_BUFFER, normalbuffer);
9  glVertexAttribPointer(...);
10 glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, elementbuffer);
11 glDrawElements(...);
```

## Renderöinti näytölle

```
1     glDisableVertexAttribArray(0);
2     glDisableVertexAttribArray(1);
3     glDisableVertexAttribArray(2);
4
5     glfwSwapBuffers(window);
6 }
```

## Renderöinti näytölle

```
1 #version 330 core
2
3 layout(location = 0) in vec3 vertexPosition_modelspace;
4 layout(location = 1) in vec2 vertexUV;
5 layout(location = 2) in vec3 vertexNormal_modelspace;
6
7 out vec2 UV;
8 out vec4 ShadowCoord;
9 out vec3 Normal;
10
11 uniform mat4 MVP;
12 uniform mat4 DepthBiasMVP;
13
14 void main(){
15     gl_Position = MVP * vec4(vertexPosition_modelspace,1);
16     ShadowCoord = DepthBiasMVP * vec4(vertexPosition_modelspace,1);
17     UV = vertexUV;
18     Normal=vertexNormal_modelspace;
19 }
```

## Renderöinti näytölle

```
1 #version 330 core
2
3 in vec2 UV;
4 in vec4 ShadowCoord;
5 in vec3 Normal;
6 layout(location = 0) out vec3 color;
7 uniform sampler2D myTextureSampler;
8 uniform sampler2DShadow shadowMap;
9 uniform float depthEps;
10 uniform vec3 lightDir;
11
12 void main(){
13     vec3 LightColor = vec3(1,1,1);
14     vec3 MaterialDiffuseColor = texture2D(myTextureSampler, UV).rgb;
15
16     float visibility = 0.2;
17     float dx=0.002f;
18     int N=10;
19     float diffuse=dot(normalize(Normal), normalize(lightDir));
20     float shadowz=ShadowCoord.z/ShadowCoord.w-depthEps;
21     visibility += 0.8f*max(0.0, diffuse)
22         *texture(shadowMap, vec3(ShadowCoord.xy, shadowz));
23     color = visibility * MaterialDiffuseColor * LightColor;
24 }
```