Projitseerimine

Konstantin Tretjakov (kt@ut.ee)

26. september 2005

Eelmine kord

- Lineaarteisendused
  - Fikseeritud baasi korral on lineaarteisendused ekvivalentsed maatriksitega.
  - Teisenduse maatriksi veerud on baasivektorite kujutiste esitused algses baasis.
  - Teisenduste kompositsioonile ja summale vastab maatriksite korrutis ja summa.
  - Keerulisemaid maatrikseid on mugav konstrueerida elementaarsetest teisendustest nagu $\mathbf{R}(\phi), \mathbf{S}(a, b), \ldots$

Eelmine kord

- Ortogonaalsed lineaarteisendused:
  - Pööre $\mathbf{R}$ on ortogonaalne teisendus, seega $\mathbf{R}^{-1} = \mathbf{R}^T$.
  - Iga ortogonaalne teisendus on kas pööre või peegeldus või nende kombinatsioon.

Eelmine kord

- Affiinne ruum: $\langle \mathcal{P}, \mathcal{V} \rangle$.
- Affiinne teisendus:
  - Teisendab punkte punktideks ja vektoreid vektoriteks.
  - On lineaarne vektorite teisendus: $F(\mathbf{v}) = \mathbf{F}\mathbf{v}$.
  - Punktide korral rahuldab:

$$F(P + \vec{v}) = F(P) + F(\vec{v})$$

ehk $F(\mathbf{p}) = \mathbf{t} + \mathbf{F}\mathbf{p}$

Eelmine kord

- Affiinseid teisendusi esitatakse *homogeensete koordinaatide* abil:
  - Punktid: $\mathbf{p} = (p_x, p_y, 1)^T$
  - Vektorid: $\mathbf{v} = (v_x, v_y, 0)^T$
  - Teisendus:

$$\left(\begin{array}{cc|c} f_{11} & f_{12} & t_x \\ f_{21} & f_{22} & t_y \\ \hline 0 & 0 & 1 \end{array}\right)$$

Eelmine kord

- Nii punktide teisendus kui ka vektorite teisendus on vastava maatriksiga korrutis, (kuid punkt $(a, b, 1)^T$ ja vektor $(a, b, 0)^T$ teisendatakse erinevalt)
- Elementaarsed afiinsed teisendused: pööre, skaleerimine, peegeldus, *nihe*.
- Punktide teisendus $\mathbf{F}$ on sama mis koordinaadistiku ("kaamerafreimi") teisendus $\mathbf{F}^{-1}$.

Eelmine kord

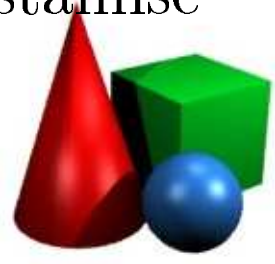
- OpenGL-is on süsteemi globaalse seisundi osaks nn. *model-view* maatriks, millega korrutatakse kõik punktid enne ekraanile väljastamist. Sellega antakse ette nii kaamera positsiooni kui ka objektide positsioone maailmas.
- Objekte saab kirjeldada hierarhiliselt, objekti renderdamiseks kasutatakse siis maatriksite stack-i.
- Peatusime sel hetkel, millal meil on olemas objektid kaamera freimis, ning me soovime neid juba ekraanile näitama hakata.

Seekord

- Planaarsed geomeetrilised projektsioonid
- Ortograafiline, kaldnurgaga ja perspektiivne projektsiooni teisendused homogeensetes koordinaatides.
- Perspektiiv ja interpoleerimine
- Perspektiiv ja varjud

Projektsioon

- *Projektsioon* on üldiselt suvaline teisendus mis kujutab punkte ühes (topoloogilises) ruumis teise ruumi peale.
  - Lineaaralgebras tähendab projektsioon mittepööratavat lineaarteisendust.
  - Geograafias tähendab projektsioon maakera kaartide tekitamise meetodit (tsilindrilised, mercator, etc).
  - Tehnilises joonises tähendab projektsioon kolmemõõtmeliste asjade paberil joonistamise viisi (isomeetriline, aksonomeetriline)



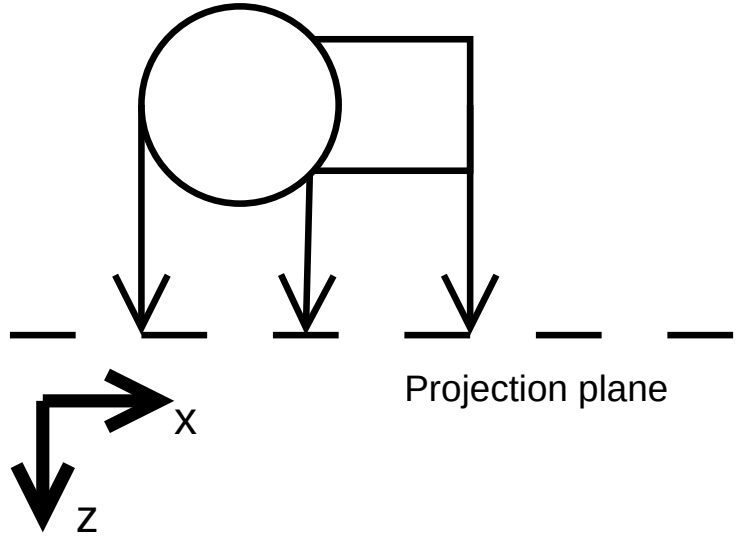
Projektsioon

- Meid huvitab siin *planaarne geomeetriline projektsioon* — teisendus, mis projitseerib *kolmemõõtmelise ruumi* punkte *tasandile sirgjoonte* abil.

Planaarsed geom. projektsioonid

- Paralleelne
  - Ortograafiline (ehk *perpendikulaarne*)
    - Ortogonaalne: *side/top/bottom*
    - Aksonomeetriline, isomeetriline.
  - Kaldnurgaga(?) (*oblique*): cavalier, cabinet.
- Perspektiivne
  - 1/2/3-punkti perspektiiv (kunstis)

Ortograafiline projektsioon



$$x' = x \qquad y' = y \qquad z' = 0$$

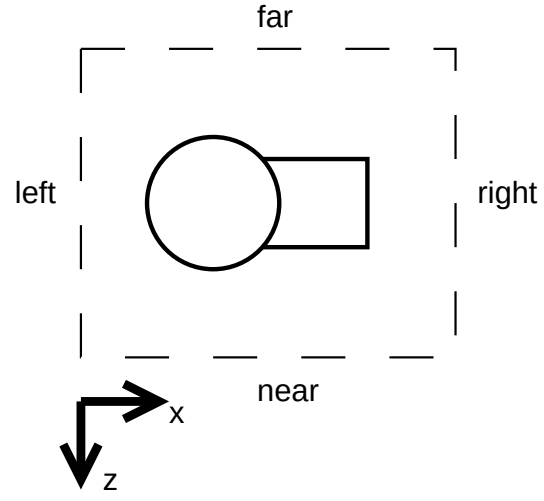
Ortograafiline projektsioon

- Loomulikult on ortograafiline projektsioon esitatav maatriksina homogeensetes koordinaatides:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Ortograafiline projektsioon

- Kuna ekraan pole lõpmata suur, peame me valima ruumis mingit ala, mida me ekraanile projitseerime. Seda ala määratakse nn. *pügamis-risttahuka (clipping box)* abil.



Ortograafiline projektsioon

- Pügamisristtahuka tahkude sisaldavaid tasandeid nimetatakse *pügamistasanditeks (clipping planes)*. Nimedeks siis *left, right, top, bottom, near, far clipping plane*.
- Osutub et on mugav teisendada ruumi nii et pügamistahukas teisendaks risttahukaks

$$\{-1 \leq x \leq 1, -1 \leq y \leq 1, -1 \leq z \leq 1\}$$

- Siis on pärast projitseerimist ekraani koordinaadid alati -1 ja 1 vahel (need on nn. *normalized device coordinates*).

Ortograafiline projektsioon

- Seega enne projitseerimist teostame meie pügamistahuka *normaliseerimist*:

$$x' = \frac{2}{x_\mathrm{r} - x_\mathrm{l}} \left(x - \frac{x_\mathrm{r} + x_\mathrm{l}}{2} \right)$$

$$y' = \frac{2}{y_\mathrm{t} - y_\mathrm{b}} \left(y - \frac{y_\mathrm{t} + y_\mathrm{b}}{2} \right)$$

$$z' = \frac{2}{z_\mathrm{n} - z_\mathrm{f}} \left(z - \frac{z_\mathrm{n} + z_\mathrm{f}}{2} \right)$$

Ortograafiline projektsioon

$$x' = \frac{2}{x_\mathrm{r} - x_\mathrm{l}} x - \frac{x_\mathrm{r} + x_\mathrm{l}}{x_\mathrm{r} - x_\mathrm{l}}$$
$$y' = \frac{2}{y_\mathrm{t} - y_\mathrm{b}} y - \frac{y_\mathrm{t} + y_\mathrm{b}}{y_\mathrm{t} - y_\mathrm{b}}$$
$$z' = \frac{2}{z_\mathrm{n} - z_\mathrm{f}} z - \frac{z_\mathrm{n} + z_\mathrm{f}}{z_\mathrm{n} - z_\mathrm{f}}$$

Ortograafiline projektsioon

- Maatrikskujul:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \left(\begin{array}{ccc|c} \frac{2}{x_\mathrm{r}-x_\mathrm{l}} & 0 & 0 & -\frac{x_\mathrm{r}+x_\mathrm{l}}{x_\mathrm{r}-x_\mathrm{l}} \\ 0 & \frac{2}{y_\mathrm{t}-y_\mathrm{b}} & 0 & -\frac{y_\mathrm{t}+y_\mathrm{b}}{y_\mathrm{t}-y_\mathrm{b}} \\ 0 & 0 & \frac{2}{z_\mathrm{n}-z_\mathrm{f}} & -\frac{z_\mathrm{n}+z_\mathrm{f}}{z_\mathrm{n}-z_\mathrm{f}} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- Tegelikult me ei taha z koordinaati päris ära kaotada. Seda läheb vaja peidetud tahkude eemaldamisel.

Ortograafiline projektsioon

- Olgu siis eelnevalt tuletatud maatriks $\mathbf{P}_{\text{ort}}$. Kui me korrutame meie punkte selle maatriksiga, saame me automaatselt punkte, mille (x, y) koordinaate saab kasutada kohe ekraani peal joonistamiseks, ning z koordinaati peidetud tahkude eemaldamiseks.
- Niiet koos projitseerimisega on meie maailma punktide kogu transformatsioon nüüd kujul

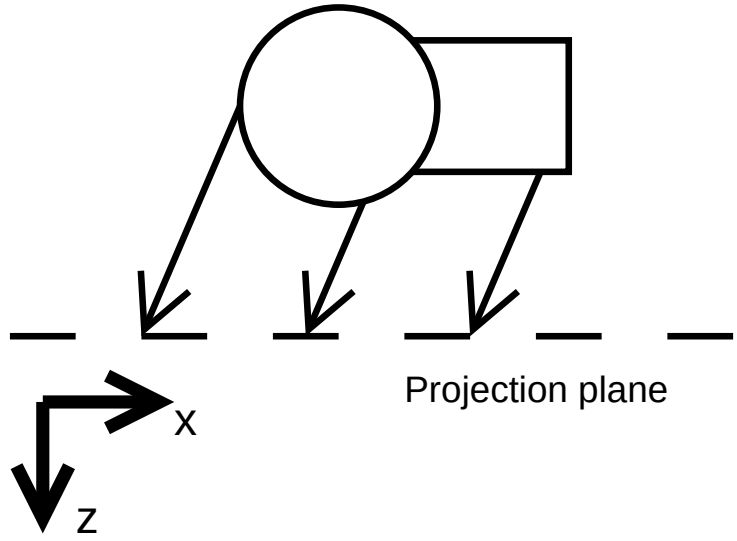
$$\mathbf{P}_{\text{ort}}\mathbf{VMx}_i$$

Ortograaf. projektsioon: OpenGL

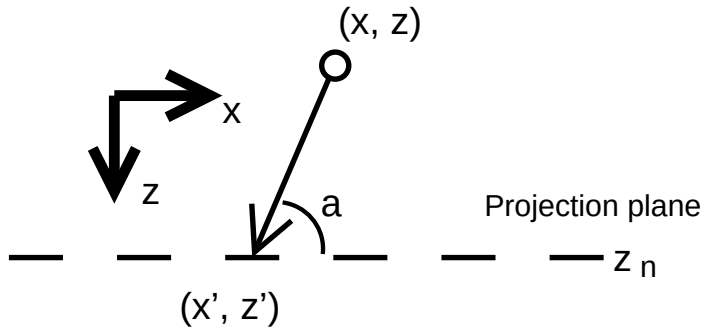
```
glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(left,right,bottom,top,dNear,dFar);
glMatrixMode(GL_MODELVIEW);
```

Oblique projection

- Ortograafilise projektsiooni üldistus on kaldnurgaline(?) (*oblique*) projektsioon:



Oblique projection



$$x' = x - (z_n - z)\cot\alpha$$

$$y' = y - (z_n - z)\cot\beta$$

Oblique projection

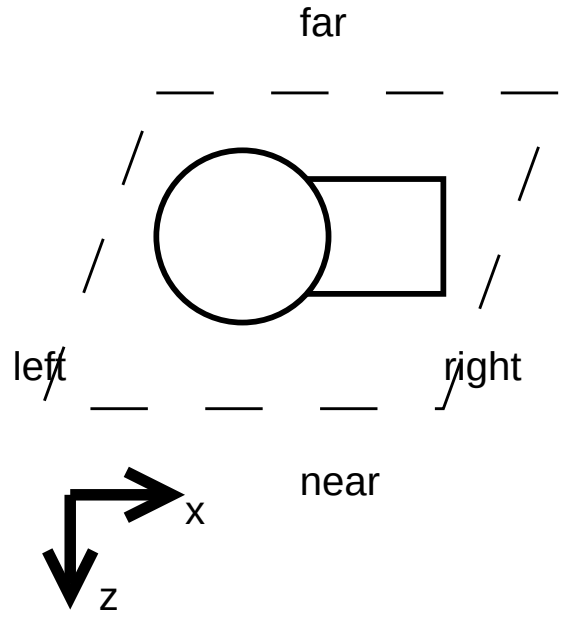
- Maatrikskujul:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \left(\begin{array}{ccc|c} 1 & 0 & \cot\alpha & -z_n\cot\alpha \\ 0 & 1 & \cot\beta & -z_n\cot\beta \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- Seda maatriksit tähistame edaspidi **H**-ga.

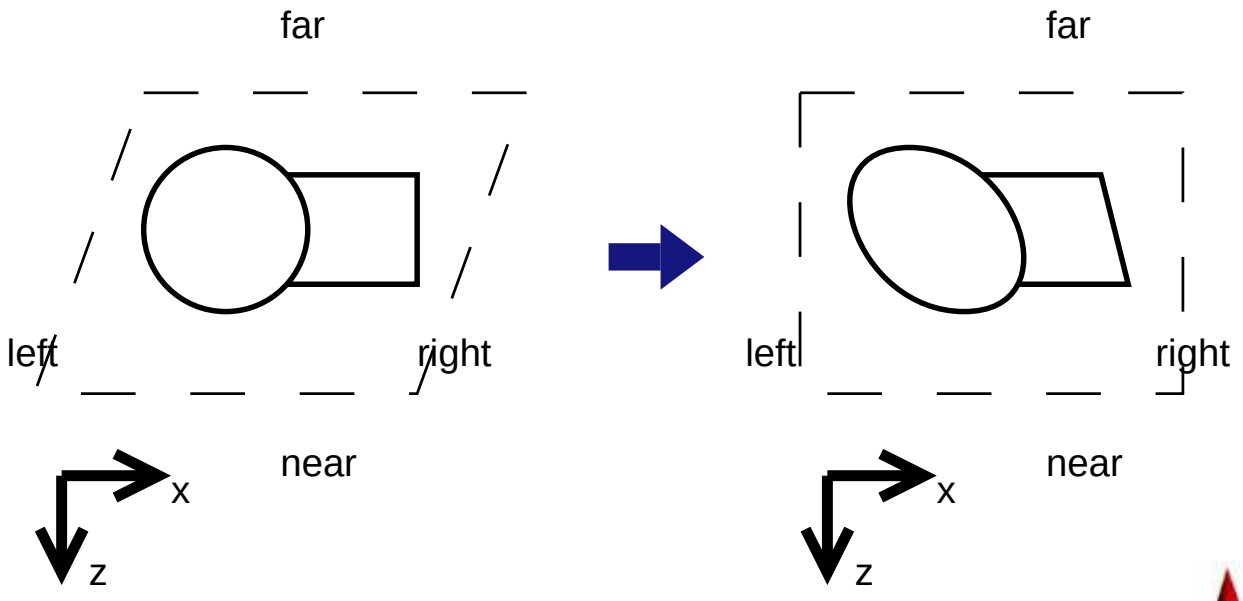
Oblique projection

- Täpselt samamoodi nagu ortogonaalsel juhul saame me määrata pügamis-*rööptahuka*:



Oblique projection

- Pärast transformatsiooni **H** muutub rööptahukas risttahukaks:



Oblique projection

- Saadud risttahuka saame nüüd normaliseerida täpselt nagu tegime ortogonaalse projektsiooni korral. Seega kokku on kaldnurgalise projektsiooni transformatsioon:

$$\mathbf{P}_{\text{obl}} = \mathbf{P}_{\text{ort}}\mathbf{H}$$

Oblique projection: OpenGL

```
glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(left,right,bottom,top,dNear,dFar);
  float H[] = {     1,      0,  0,  0,
                    0,      1,  0,  0,
                cot(a),cot(b),1,  0,
                    0,      0,  0,  1 };
  glMultMatrixf(H);
glMatrixMode(GL_MODELVIEW);
```

Homogeensed koordinaadid

- Ennem defineerisime punktide esitust *homogeensetes koordinaatides* kui kujutust

$$(x, y, z) \to (x, y, z, 1)$$

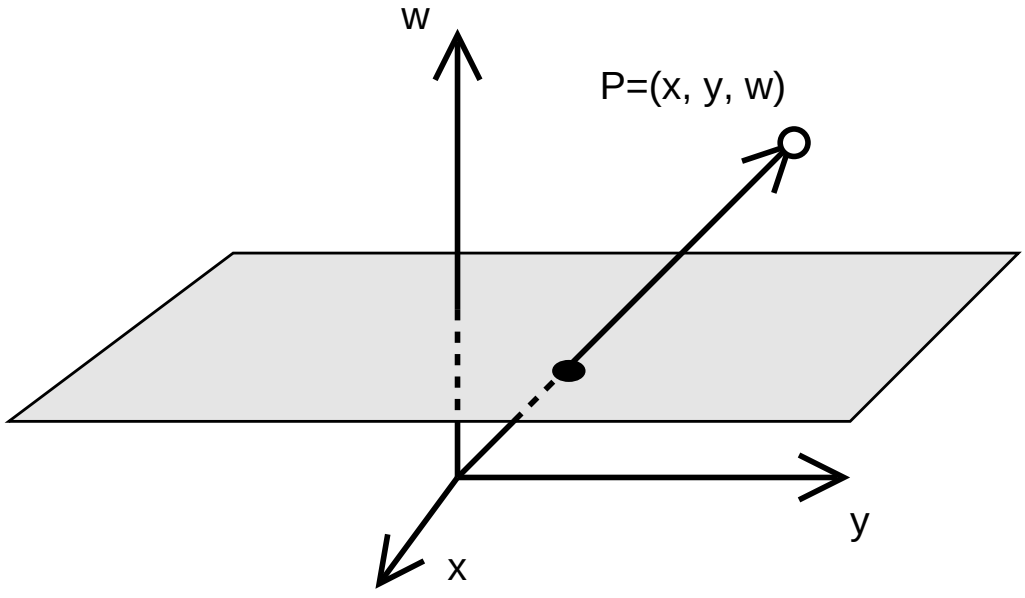
- Defineerime nüüd vastupidist kujutust: punktile (x, y, z, w) homogeensetes koordinaatides paneme vastavusse ruumi $\mathbb{R}^3$ punkti $\left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}\right)$.
- Selline esitus kõvasti aitab meile perspektiivse teisenduse juures.

Homogeensed koordinaadid

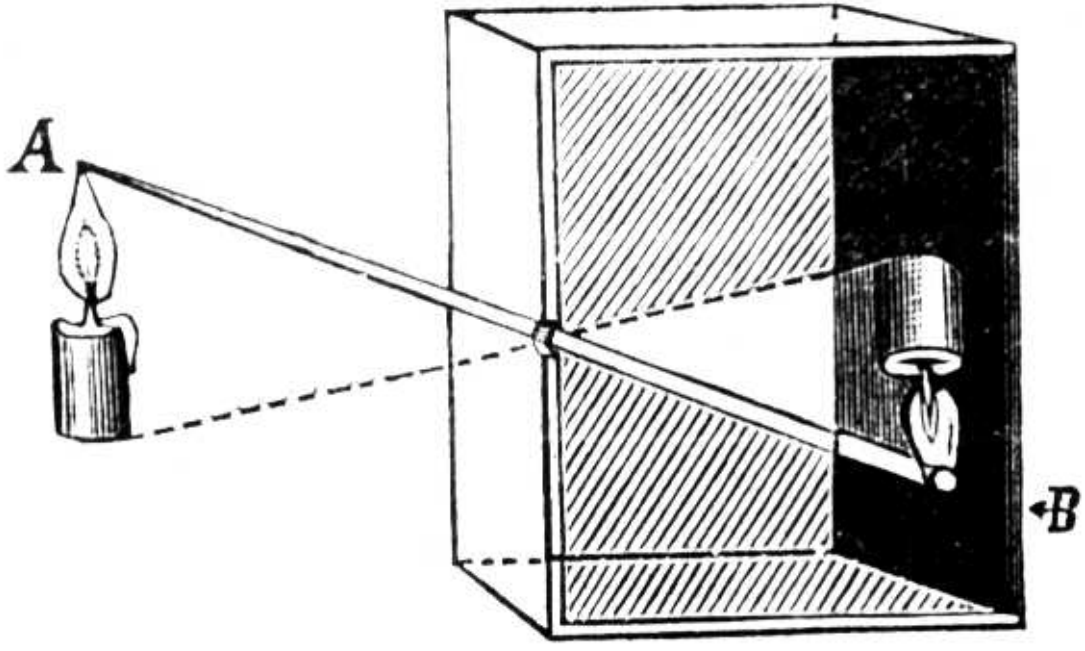
- Nii vastab ühele ruumi $\mathbb{R}^3$ punktile lõpmata palju punkte homogeenses ruumis (terve sirge). Homogeensetele punktidele, millel $w = 0$ ei vasta aga ühtegi punkti ruumis $\mathbb{R}^3$ (selline punkt on projektiivgeomeetrias tuntud kui lõpmatuspunkt (*point at infinity*))

Homogeensed koordinaadid

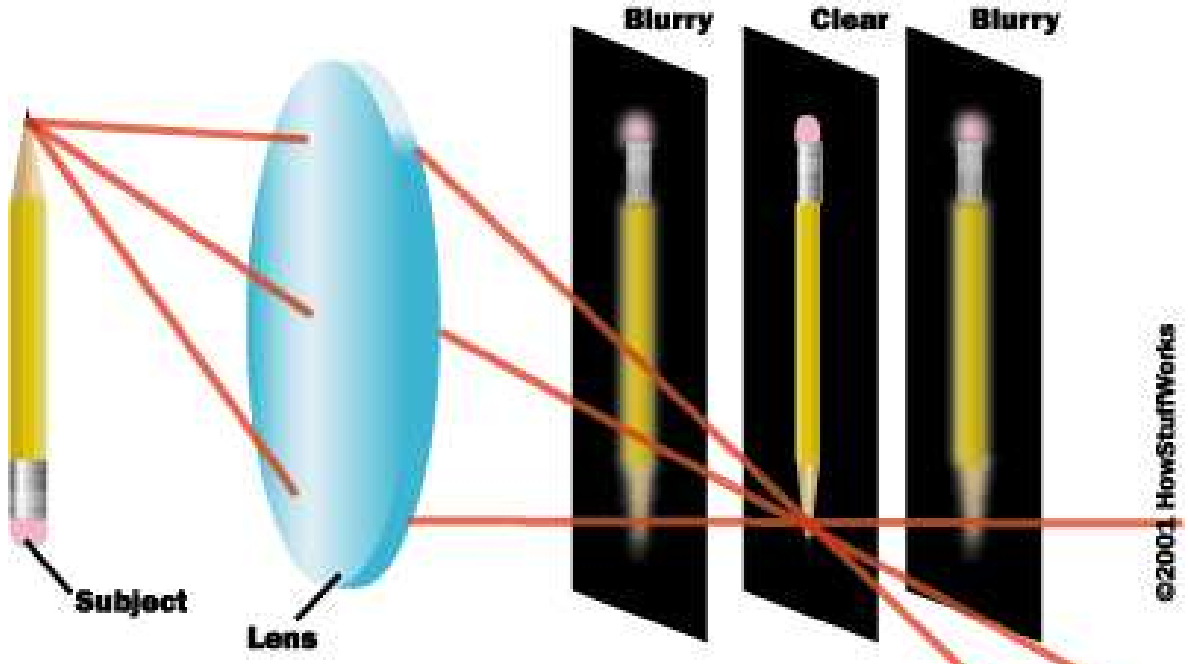
- Kahemõõtmeliste punktide korral analoogselt $(x, y, w) \leftrightarrow \left(\frac{x}{w}, \frac{y}{w}\right)$:



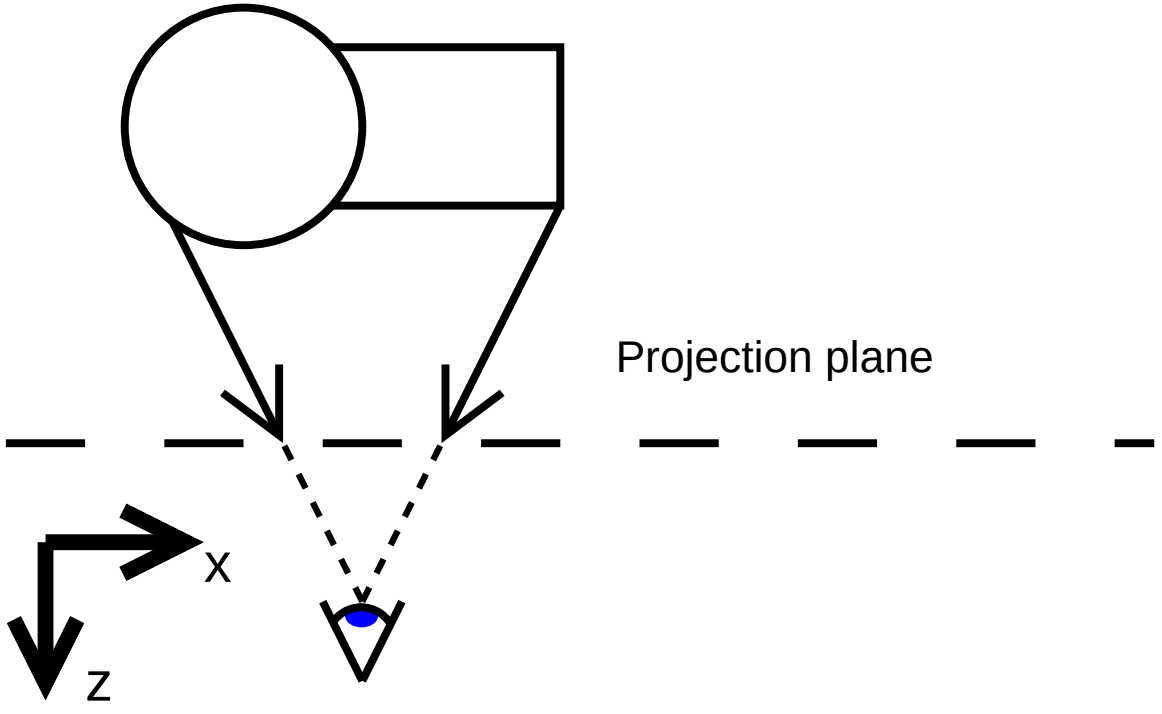
Camera obscura



Tegelik kaamera

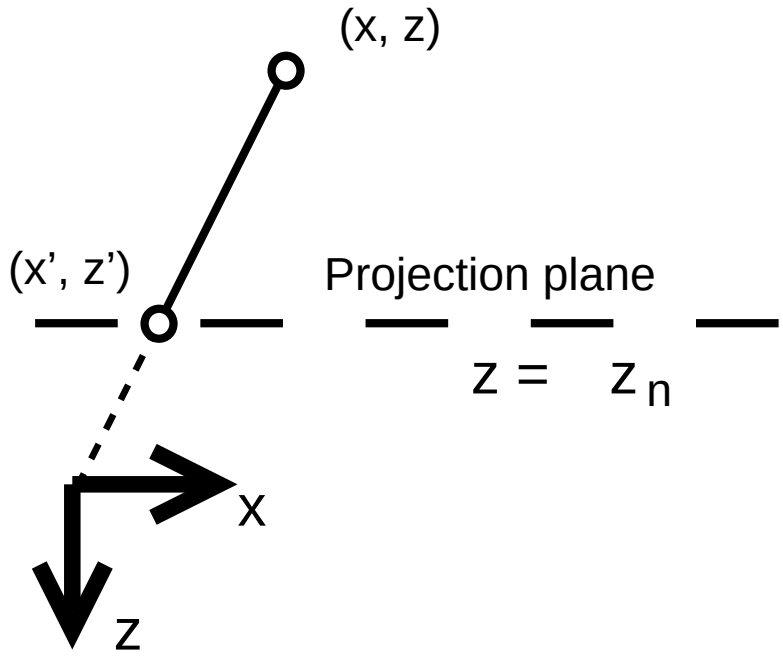


Mida meie tegema hakkame



Perspektiivne projektsioon

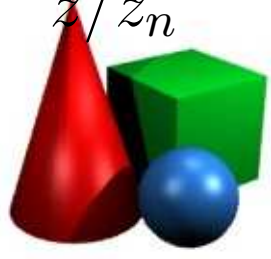
- Olgu projektsiooni keskpunktiks koordinaadistiku keskpunkt. Projitseerime tasandile $z = z_n$.



$$x' = \frac{z_n}{z}x = \frac{x}{z/z_n}$$

$$y' = \frac{z_n}{z}y = \frac{y}{z/z_n}$$

$$z' = z_n = \frac{z}{z/z_n}$$



Perspektiivne projektsioon

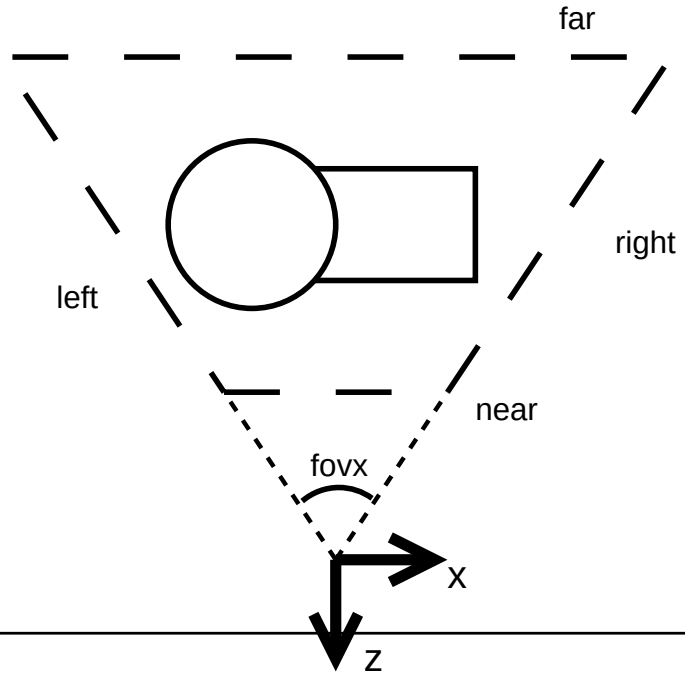
- Vaatleme järgmist kujutust homogeensetes koordinaatides:

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 1/z_n & 0 \end{array}\right) \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

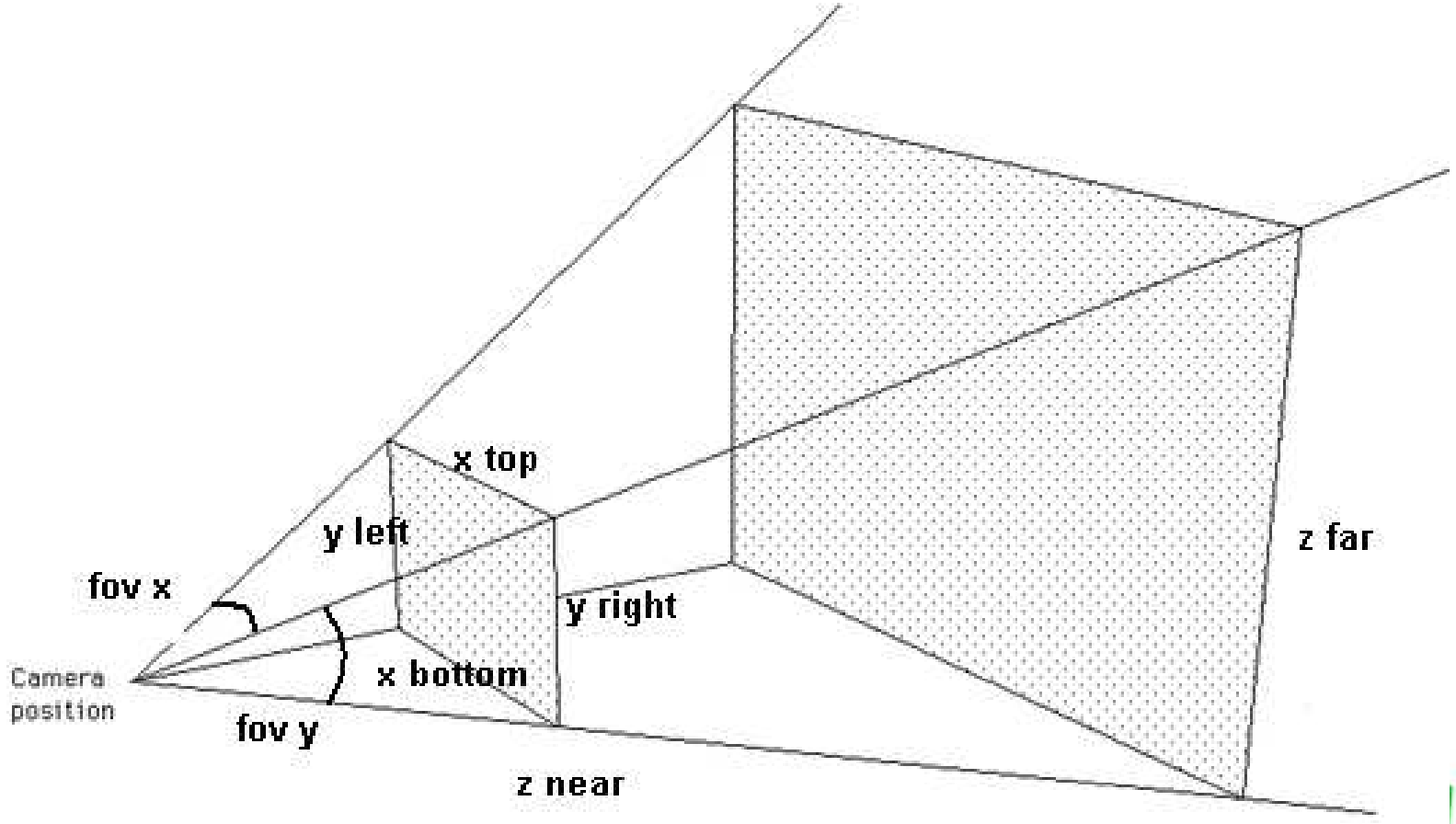
- Ta kujutab punkti $(x, y, z, 1)^T$ punktiks $(x, y, z, z/z_n)^T$ ja seega teostabki vajaliku projektsiooni.

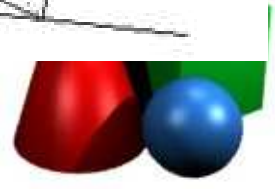
View frustum

- Analoogselt ortograafilisele projektsioonile peame määrama nähtava ruumi ala. Seekord on selle ala kujuks *pügatud püramiid (frustum)*:



View frustum





View frustum normalization

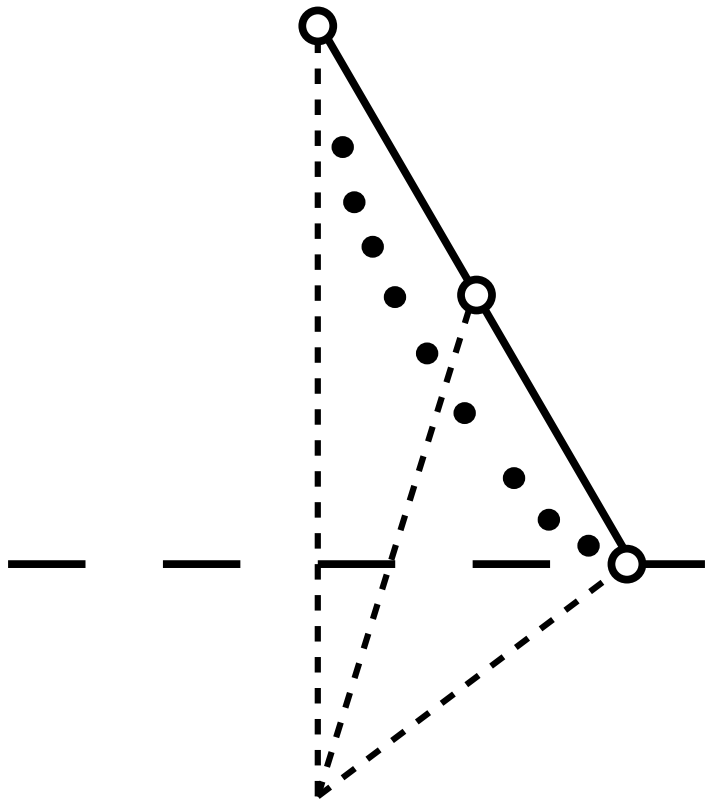
- Proovime teha sama triki nagu ortogonaalse ja kald-projektsioonide puhul — normaliseerime frustum-i $2 \times 2 \times 2$ kuubiks koordinaatide keskel.
- x ja y koordinaatidega on lihtne — lihtsalt skaleerime neid pärast projitseerimist nii et ristkülik $\{x_\mathrm{l} \leq x \leq x_\mathrm{r}, y_\mathrm{b} \leq y \leq y_\mathrm{t}\}$ muutuks ristkülikuks $\{-1 \leq x \leq 1, -1 \leq y \leq 1\}$

Sügavuse säilitamine

- z koordinaadiga on aga praegu probleem. Nimelt kujutisega $(x, y, z) \rightarrow \left(\frac{x}{z/z_n}, \frac{y}{z/z_n}, z_n\right)$ oleme me kaotanud informatsiooni sügavuse kohta. Tuleb kuidagimoodi sügavusinformatsiooni ikkagi säilitada.
- Kas $(x, y, z) \rightarrow \left(\frac{x}{z/z_n}, \frac{y}{z/z_n}, z\right)$ on hea lahendus?

Sügavuse säilitamine

- Ei, sest sirged kujutatakse kõverateks.



Sügavuse säilitamine

- Osutub et hea lahendus on kujutus

$$(x, y, z) \rightarrow \left(\frac{x}{z/z_n}, \frac{y}{z/z_n}, A + \frac{B}{z} \right)$$

- Kui $B < 0$ siis säilitab ta sügavuse informatsiooni ses mõttes et

$$A + \frac{B}{z_1} \leq A + \frac{B}{z_2} \qquad \Leftrightarrow \qquad z_1 \leq z_2$$

Sügavuse säilitamine

- Konstante A ja B saab valida enam-vähem suvaliselt, kuid mugav oleks kui punktid mis asuvad tasandil $z = z_n$ (*near clipping plane*) läheksid üle punktideks tasandil $z' = 1$ ning need mis asuvad tasandil $z = z_f$ kujutaks punktideks tasandil $z' = -1$. Nendest tingimustest saame:

$$A + \frac{B}{z_n} = 1 \quad A + \frac{B}{z_f} = -1$$

$$A = \frac{-(z_f + z_n)}{z_f - z_n} \quad B = \frac{2 z_n z_f}{z_f - z_n}$$

Homogeensetes koordinaatides

- Kujutus $(x, y, z) \to \left(\frac{x}{z/z_n}, \frac{y}{z/z_n}, A + \frac{B}{z}\right)$ on homogeensetes koordinaatides esitatav kui

$$(x, y, z, 1) \to \left(\frac{x}{z/z_n}, \frac{y}{z/z_n}, A + \frac{B}{z}, 1\right)$$

mis on samaväärne

$$(x, y, z, 1) \to (z_n x, z_n y, Az + B, z)$$

Homogeensetes koordinaatides

- Seega

$$\left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}, 1\right) \rightarrow \left(\frac{x}{w}z_n, \frac{y}{w}z_n, A\frac{z}{w} + B, \frac{z}{w}\right)$$

  mis on samaväärne

$$(x, y, z, w) \rightarrow (z_n x, z_n y, Az + Bw, z)$$

- Seega saime lineaarkujutust! (sellest järeldubki et ta säilitab sirgeid).

Projektsiooni maatriks

- Maatrikskujul:

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \left(\begin{array}{ccc|c} z_n & 0 & 0 & 0 \\ 0 & z_n & 0 & 0 \\ 0 & 0 & A & B \\ \hline 0 & 0 & 1 & 0 \end{array}\right) \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

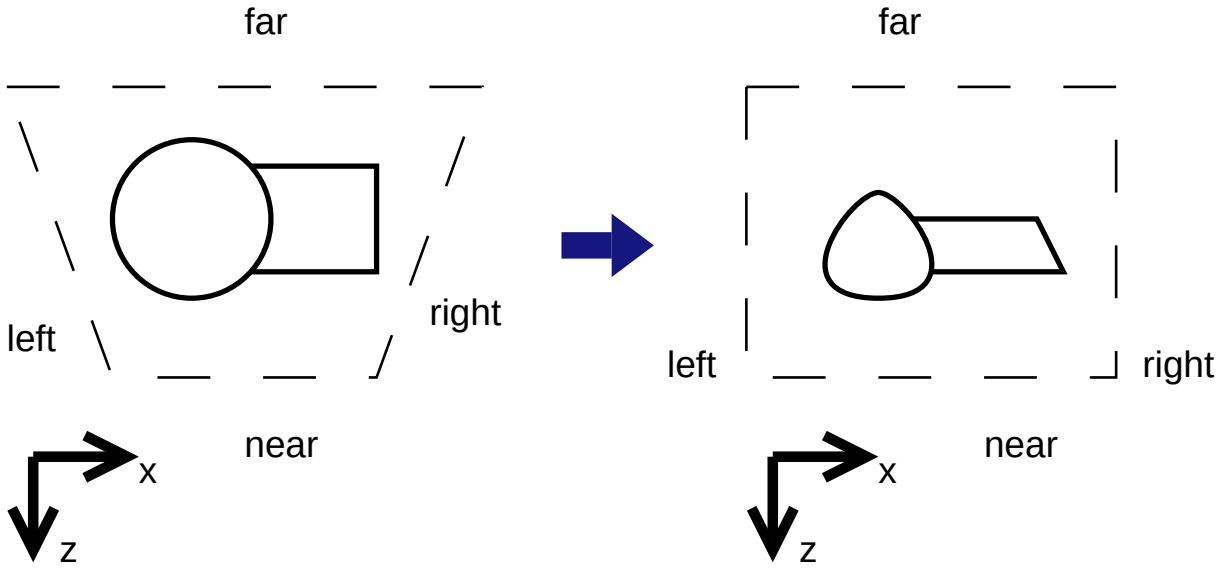
Projektsiooni maatriks

- Kui rakendame ka x ja y normaliseerimist saame kokku maatriksi:

$$\mathbf{P}_{\text{persp}} = \left(\begin{array}{cc|c|c} \frac{2z_n}{x_r - x_l} & 0 & 0 & 0 \\ 0 & \frac{2z_n}{y_t - y_b} & 0 & 0 \\ 0 & 0 & \frac{-(z_f + z_n)}{z_f - z_n} & \frac{2z_n z_f}{z_f - z_n} \\ \hline 0 & 0 & 1 & 0 \end{array}\right)$$

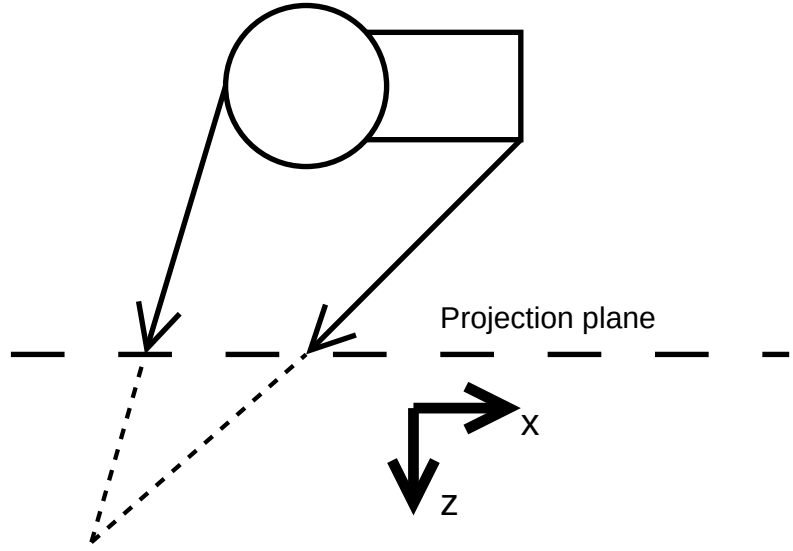
Projektsiooni maatriks

- Olemegi saavutanud frustum-i normaliseerimise:



Mitteortogonaalne perspektiiv

- Vaatlesime juhtumi kus projektsiooni tasand on perpendikulaarne vaate suunaga. Mõnikord on vaja projitseerida ka niimoodi:



Mitteortogonaalne perspektiiv

- Mitteortogonaalse juhtumi saame viia üle ortogonaalsele kui rakendame teisendust **H**, millega normaliseerisime kaldnurkse projektsiooni clipping-rööptahuka. Sellel üldisemal juhul on teisenduse maatriks kujul:

$$\mathbf{P}_{\text{persp}} = \left(\begin{array}{cccc} \frac{2z_n}{x_r-x_l} & 0 & \frac{x_r+x_l}{x_r-x_l} & 0 \\ 0 & \frac{2z_n}{y_t-y_b} & \frac{y_t+y_b}{y_t-y_b} & 0 \\ 0 & 0 & \frac{-(z_f+z_n)}{z_f-z_n} & \frac{2z_n z_f}{z_f-z_n} \\ \hline 0 & 0 & 1 & 0 \end{array}\right)$$

Perspektiiv OpenGL-is

```
glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glFrustum(left,right,bottom,top,dNear,dFar);
glMatrixMode(GL_MODELVIEW);
```

Perspektiiv OpenGL-is

```
glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluPerspective(fovY,aspect,dNear,dFar);
glMatrixMode(GL_MODELVIEW);
```

Märkused: Field-of-view

- Frustum-i nurk (FOV):
  - "Tavaliste" piltide renderdamisel peab FOV olema mõistlikult keskmine. 30 kraadi on ok valik.
  - Suur FOV tekitab nn. *fish-eye* efekti. Lähedased objektid on oluliselt suuremad kui kaugemad.
  - Väike FOV on rohkem sarnane ortograafilise projektsioonile. Imiteerib "zoom"-i.

Märkused: Perspective division

- Kuna perspektiivne projektsioon eksplitsiitselt kasutab homogeensete koordinaatide w komponendi, siis pärast punktide teisendamist ja projitseerimist jagatakse kõik koordinaadid w komponendiga. Seda sammu nimetatakse *perspective division*.
- 1. Teisendus: $\mathbf{x}_i \rightarrow \mathbf{P}_{\text{persp}}\mathbf{VMx}_i$
  2. Perspective division: $(x, y, z, w) \rightarrow (x/w, y/w, z/w, 1)$

Viewport transform

- Projitseerimine annab välja punkte *normalizeeritud koordinaatides*, s.t. koordinaadid on -1 ja 1 vahel. Neid punkte ekraanile joonistamisel kasutatakse juba ekraani enda koordinaate (kus punkt koordinaatidega $(0, 0)$ on tihti üleval vasakul). Seda punktide viimast teisendust nimetatakse *viewport*-transformatsiooniks.

$$x_{\text{win}} = \frac{\text{screen width}}{2}(x_{\text{norm}} + 1)$$

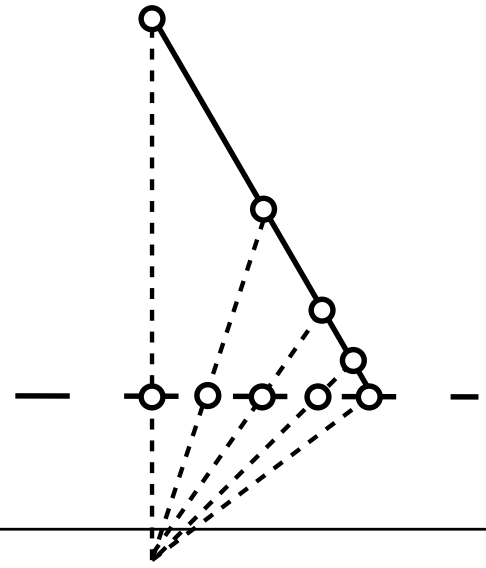
$$y_{\text{win}} = \frac{\text{screen height}}{2}(1 - y_{\text{norm}})$$

Graphics pipeline

1. Vertex transforms $\Leftarrow$ Done
2. Face culling & clipping
3. Rasterization
4. Shading
5. Visibility tests & blending

Perspective interpolation

- Kui me kasutame perspektiivset projektsiooni, siis lineaarne interpoleerimine ekraanil *ei vasta* lineaarsele interpoleerimisele maailmas. Seda tuleb arvestada eelkõige tekstuuri koordinaatide interpoleerimisel.



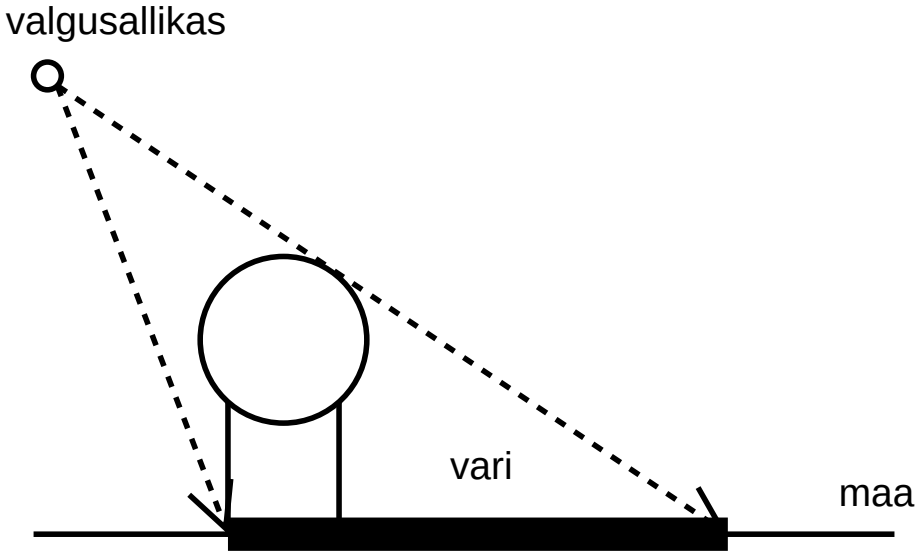
Perspective interpolation

- Olgu punktiga $P_1 = (x_1, y_1, z_1)$ seotud mingi atribuuti väärtus a_1, punktiga $P_2 = (x_2, y_2, z_2)$ – väärtus a_2, ning neid ühendaval sirglõigul muutub atribuut lineaarselt. Olgu punkti P_1 perspektiivne projektsioon P_1^*, punkti P_2 perspektiivne projektsioon — P_2^*. Olgu $P_t^* = tP_1^* + (1-t)P_2^*$ ning olgu P_t sirglõiku $[P_1, P_2]$ punkt mille projektsioon on P_t^*. Saab näidata et atribuuti väärtus a_t selles punktis rahuldab

$$\frac{a_t}{z_t} = t\frac{a_1}{z_1} + (1-t)\frac{a_2}{z_2}$$

Perspective shadows

- Perspektiivset projektsiooni saab mugavalt kasutada lihtsate varjude tekitamiseks



Perspective shadows

- Leiame projektsioonimaatriksit $\mathbf{P}_{\text{shad}}$ mis projitseerib asju maa tasandile.
- Joonistame maad
- Joonistame objekti
- Lülitame musta värvi sisse ning joonistame objekti, teisendatud projektsioonimaatriksiga. See tekitab maa peal sobiva varju.
- Antud juhul võib võtta kõige lihtsama projektsioonimaatriksi sest mingit sügavust siin säilitada pole vaja.

Kokkuvõte

- Erinevad projektsioonid on esitatavad lineaarteisendustena homogeensetes koordinaatides: $\mathbf{P}_{\text{ort}}$ $\mathbf{P}_{\text{obl}}$ $\mathbf{P}_{\text{persp}}$
- Projektiivseid teisendusi võib kasutada varjude realiseerimiseks.
- Lineaarne interpoleerimine perspektiivselt projitseeritud punktide peal ei vasta lineaarsele interpoleerimisele punktide originaalidel.
- Järgmine kord siis tahkude kärpimisest ja peidetud pindade eemaldamisest.

Küsimused?