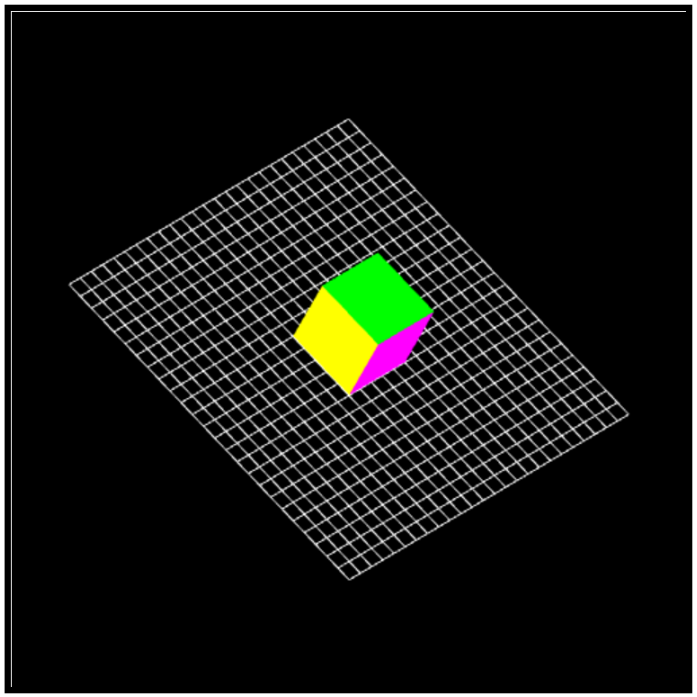
WebGL & OpenGL ES

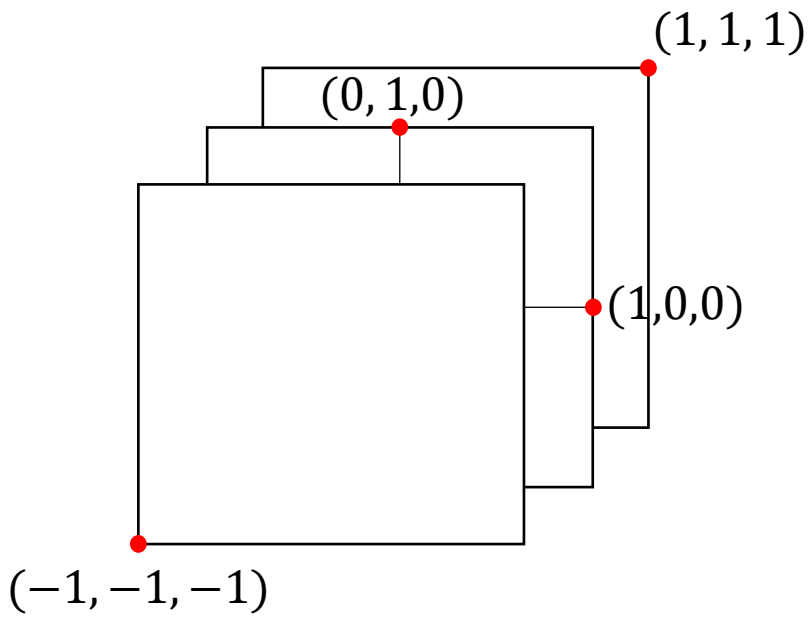
Uvod u 3D



Ortogonalna projekcija

Normirane koordinate

- sve što u normiranim koordinatama izlazi izvan raspona $[-1, 1]$ je odrezano, tj. ne iscrtava se (vrijedi i za z koordinatu!)



- spremnik dubine manje vrijednosti z koordinate interpretira kao bliže, a veće z koordinate kao dalje
- napomena: normirane koordinate čine lijevi koordinatni sustav!

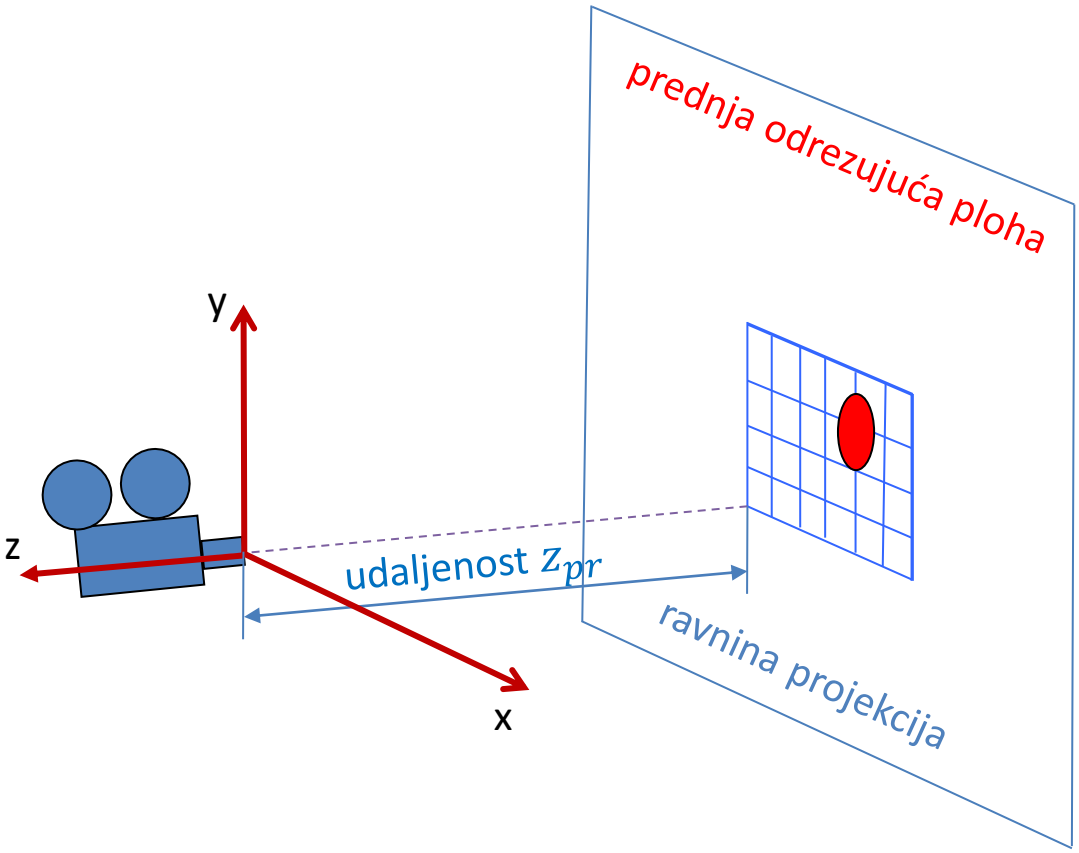
Ortogonalna projekcija

- zapravo nam ne treba – OpenGL ES automatski odbacuje z-koordinatu i u ViewPort iscrtava ono što je u rasponu normiranih koordinata
- ipak može biti korisna u svrhu skaliranja: možemo slobodno „sagraditi” scenu ne razmišljajući o dimenzijama objekata
- možemo zadati raspon koordinata koje želimo preslikati u normirane koordinate

Uloga z-koordinate

- kad smo radili žičane modele z-koordinatu smo kod ortogonalne projekcije jednostavno samo odbacivali
- međutim, sada imamo dva problema:
  - spremnik dubine treba informaciju o z-koordinati
  - odrezujuće plohe: bliža na -1, a dalja na 1
- osim x i y moramo transformirati i z-koordinatu

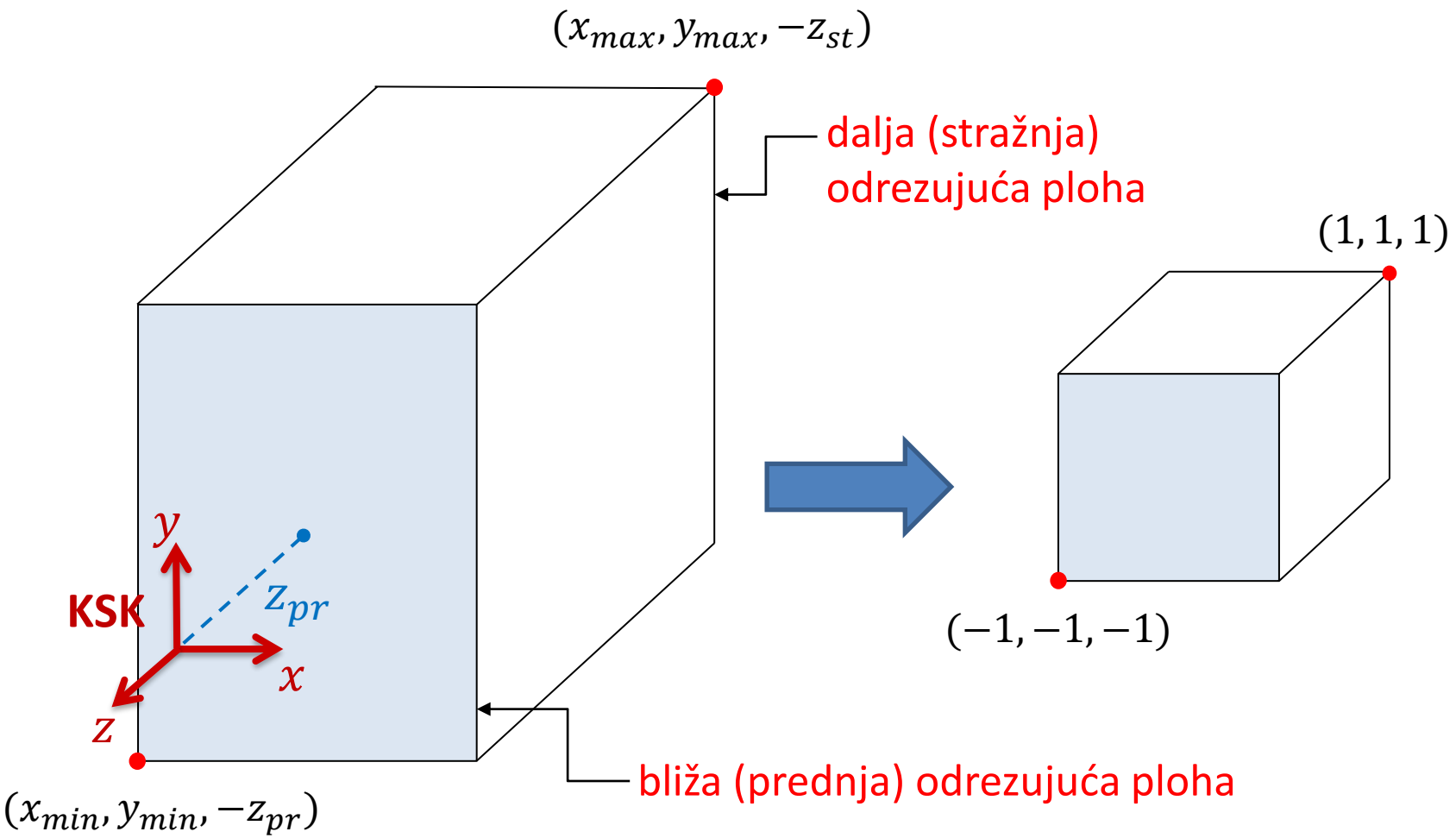
Koordinatni sustav kamere (KSK)



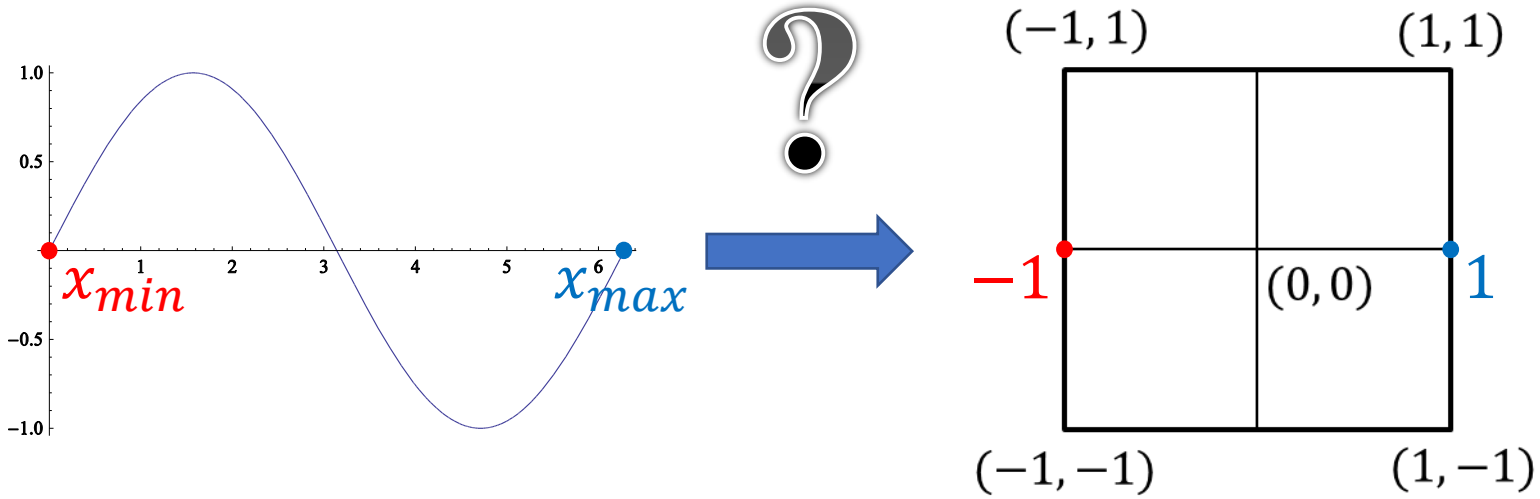
Prednja i stražnja odrezujuća ploha

- kad koristimo koordinatni sustav kamere obično biramo ravninu projekcije za prednju (bližu) odrezujuću plohu
- udaljenost do prednje odrezujuće plohe označavat ćemo sa z_{pr}
- moramo odabrati i stražnju (dalju) odrezujuću plohu na udaljenosti $z_{st} > z_{pr} > 0$
- u literaturi na engleskom z_{near} i z_{far}

Transformacija u normirane koordinate



Transformacija x-koordinate



- moramo preslikati $x_{min} \rightarrow -1$ i $x_{max} \rightarrow 1$, dakle:

$$-1 = s_x\, x_{min} + p_x \qquad (1)$$

$$1 = s_x\, x_{max} + p_x \qquad (2)$$

Transformacija x-koordinate

- ako od (2) oduzmemo (1) slijedi izraz za faktor skaliranja s_x

$$1 - (-1) = s_x\, x_{max} + p_x - (s_x\, x_{min} + p_x)$$

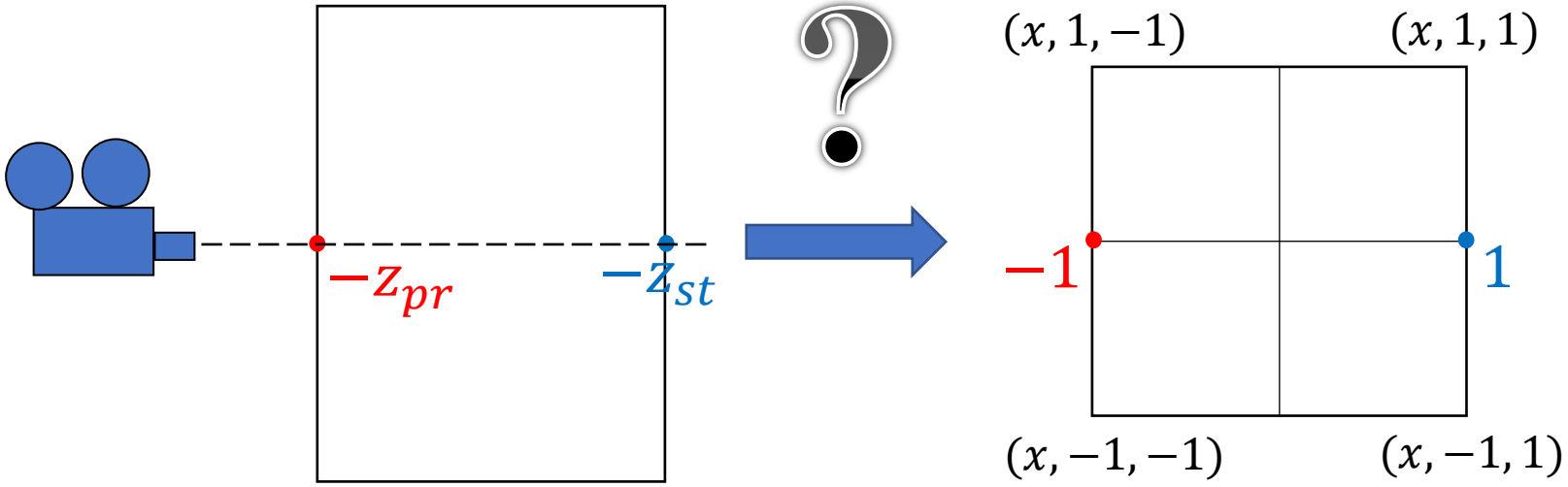
$$2 = s_x(\, x_{max} - \, x_{min})$$

$$s_x = \frac{2}{x_{max} - \, x_{min}}$$

- iz (1) slijedi izraz za pomak p_x

$$p_x = 1 - s_x\, x_{max} = \frac{x_{min} + x_{max}}{x_{min} - x_{max}}$$

Transformacija z-koordinate



- moramo preslikati $-z_{pr} \rightarrow -1$ i $-z_{st} \rightarrow 1$, dakle:

$$-1 = s_z(-z_{pr}) + p_z \quad (1)$$

$$1 = s_z(-z_{st}) + p_z \quad (2)$$

Transformacija z-koordinate

- ako od (2) oduzmemo (1) slijedi izraz za faktor skaliranja s_z

$$1 - (-1) = s_z(-z_{st}) + p_z - s_z(-z_{pr}) - p_z$$

$$2 = s_z(z_{pr} - z_{st})$$

$$s_z = \frac{2}{z_{pr} - z_{st}}$$

- iz (1) slijedi izraz za pomak p_z

$$p_z = 1 + s_z z_{st} = \frac{z_{pr} + z_{st}}{z_{pr} - z_{st}}$$

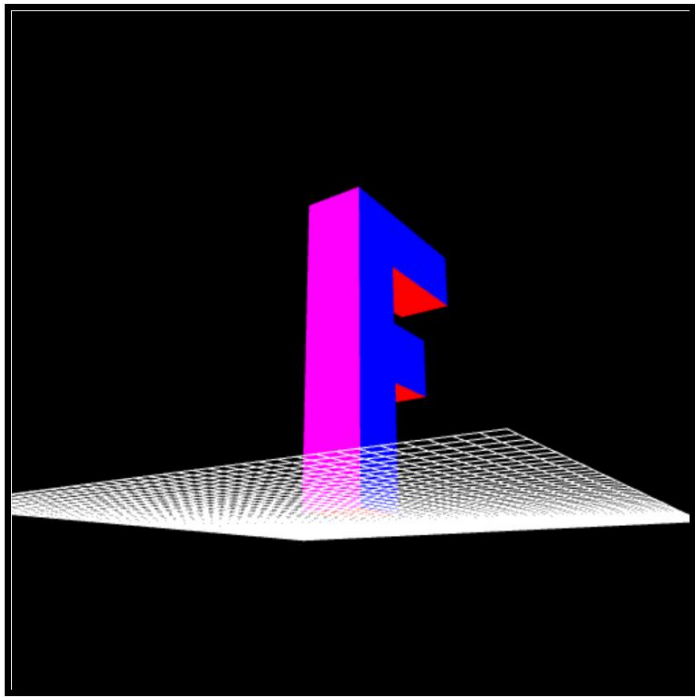
Ortogonalna projekcija u normirane koordinate

- matrica transformacije u homogenim koordinatama

$$\begin{bmatrix} s_x & 0 & 0 & p_x \\ 0 & s_y & 0 & p_y \\ 0 & 0 & s_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{x_{max} - x_{min}} & 0 & 0 & \frac{x_{min} + x_{max}}{x_{min} - x_{max}} \\ 0 & \frac{2}{y_{max} - y_{min}} & 0 & \frac{y_{min} + y_{max}}{y_{min} - y_{max}} \\ 0 & 0 & \frac{2}{z_{pr} - z_{st}} & \frac{z_{pr} + z_{st}}{z_{pr} - z_{st}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

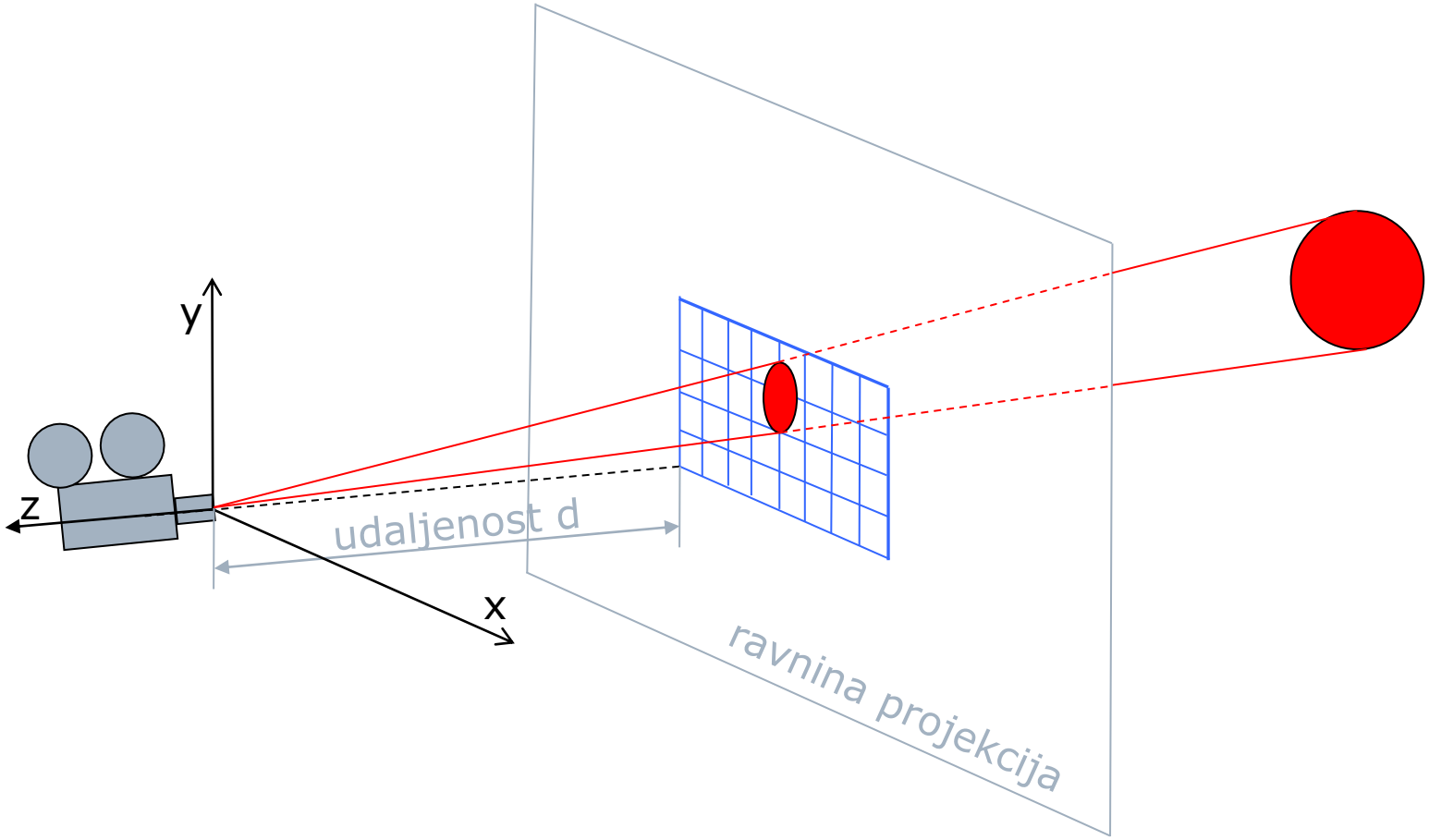
WebGL & OpenGL ES

Uvod u 3D

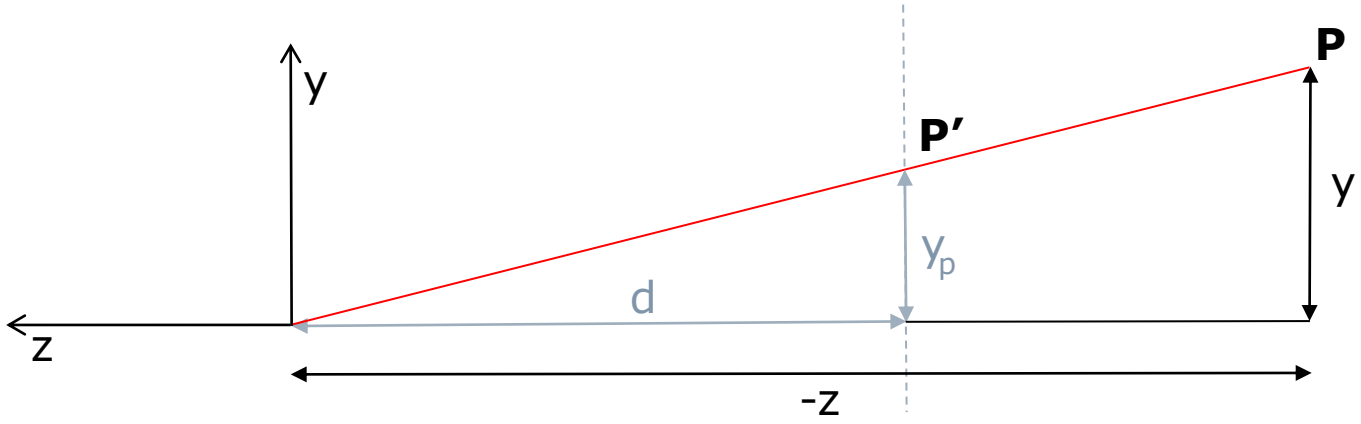


Perspektivna projekcija

Perspektivno projiciranje (1)



Perspektivno projiciranje (2)



Zbog sličnosti trokuta:

$$\frac{y_p}{d} = \frac{y}{-z} \quad \Rightarrow \quad y_p = -\frac{d}{z} y$$

Perspektivno projiciranje (3)

Za specijalni slučaj kad zrake izlaze iz (ili bolje reći ulaze u) centar projekcija koji je u ishodištu, a ravnina projekcija je paralelna sa *xy*-ravninom i nalazi se na udaljenosti *d* u smjeru negativne osi *z,* projicirane koordinate dane su sa:

$$x_p = -\frac{d}{z}x \qquad y_p = -\frac{d}{z}y$$

Napomena: Ako se koristi metoda z-spremnika za određivanje vidljivosti, onda se osim x_p i y_p mora pohraniti i (normirana) z-koordinata

Perspektivno projiciranje

- perspektivno projiciranje nije linearna transformacija!
- tek u kombinaciji sa svojstvima homogenih koordinata može se konstruirati odgovarajući matrični operator
- **OpenGL** podržava takav pristup jer program za sjenčanje vrhova automatski interpretira tip `vec4` kao homogene koordinate

Homogene koordinate

- homogene koordinate $[x, y, z, w]$ i $\left[\frac{x}{w}, \frac{y}{w}, \frac{z}{w}, 1\right]$ opisuju istu točku u Euklidskom prostoru i to točku $\left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}\right)$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = w \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \\ \frac{z}{w} \\ 1 \end{bmatrix} \Rightarrow točka\left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}\right)$$

Perspektivno projiciranje pomoću homogenih koordinata

- perspektivno projiciranje

$$x_p = -\frac{d}{z}\,x \qquad y_p = -\frac{d}{z}\,y$$

- dijeljenje sa $-z$ možemo postići tako da ga stavimo kao četvrtu komponentu

$$\begin{bmatrix} d\,x \\ d\,y \\ d\,z \\ -z \end{bmatrix} = -z \begin{bmatrix} -\frac{d}{z}\,x \\ -\frac{d}{z}\,y \\ -d \\ 1 \end{bmatrix} \Rightarrow točka\left(-\frac{d}{z}\,x, -\frac{d}{z}\,y, -d\right)$$

Matrični oblik u homogenim koordinatama

- možemo konstruirati 4x4 matricu koja djeluje na homogene koordinate te ako se rezultat interpretira kao točka u euklidskom prostoru ta matrica obavlja perspektivno projiciranje

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ w_p \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} d\,x \\ d\,y \\ d\,z \\ -z \end{bmatrix} = -z \begin{bmatrix} -\frac{d}{z}\,x \\ -\frac{d}{z}\,y \\ -d \\ 1 \end{bmatrix}$$

Automatska implementacija u programu za sjenčanje vrhova

- transformirana koordinata vrha u programu za sjenčanje pohranjuje se u predefiniranu varijablu

```
gl_Position = vec4(x, y, z, w);
```

- pošto se radi o homogenim koordinatama preostale koordinate automatski se podijele s parametrom skaliranja `w` i vrh se konačno iscrtava na poziciji

$$\left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}\right)$$

Primjer 8.2. – vertex shader

- iz `RG-primjer8-2-homogene-koordinate.html`

```
#version 300 es
in vec2 a_vrhXY;
uniform mat2 u_mTrans;
uniform float u_boja;

void main() {
  gl_Position = vec4(u_mTrans * a_vrhXY, 0,
    u_boja); // normira se tako da je w = 1
}
```

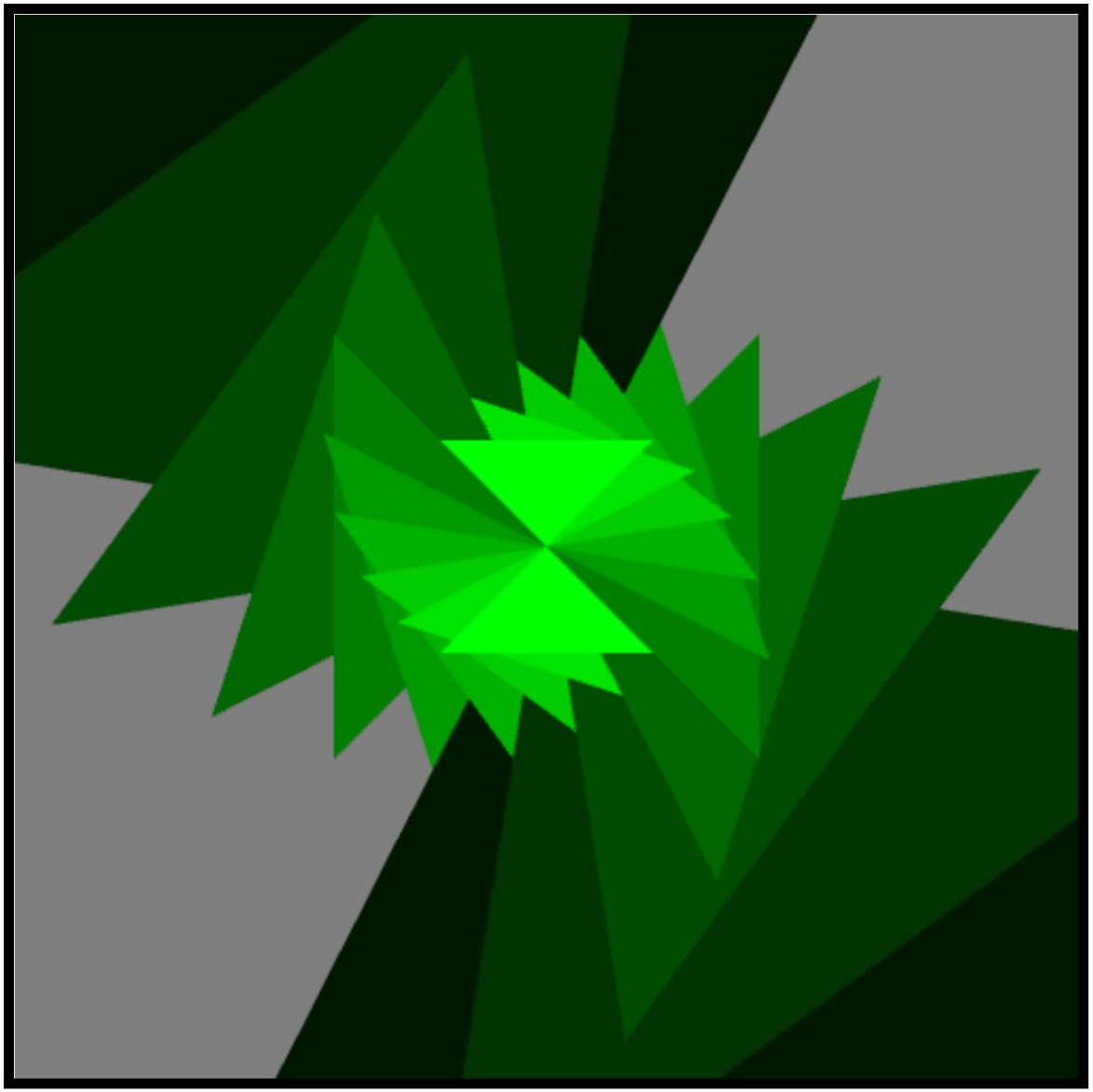
Primjer 8.2. – fragment shader

- iz `RG-primjer8-2-homogene-koordinate.html`

```
#version 300 es
precision highp float
uniform float u_boja;
out vec4 bojaPiksela;

void main() {
  bojaPiksela = vec4(0, u_boja, 0, 1);
}
```

Primjer 8.2.



Matrični oblik u homogenim koordinatama

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ w_p \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} d\,x \\ d\,y \\ d\,z \\ -z \end{bmatrix} = -z \begin{bmatrix} -\frac{d}{z}\,x \\ -\frac{d}{z}\,y \\ -d \\ 1 \end{bmatrix}$$

- problem s ovom jednostavnom projekcijom je da je z_p uvijek jednak $-d$
- nedostaci:
  - ne možemo se služiti spremnikom dubine
  - d je ograničen na interval od -1 do 1

Perspektivna projekcija u normirane koordinate

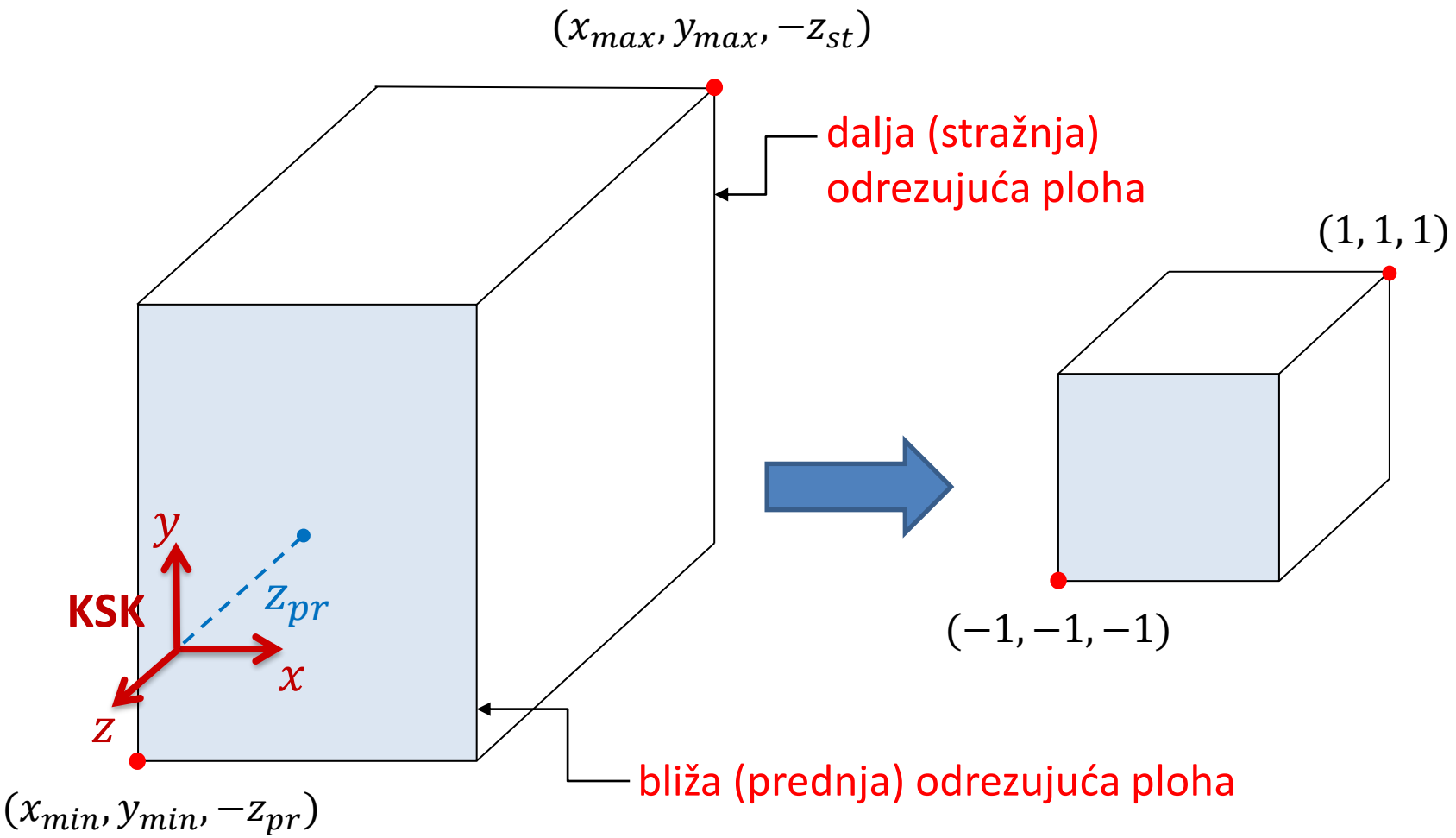
- na sličan način kako je opisano kod ortogonalne projekcije provodi se transformacija koordinata u normirane koordinate
- opet se koordinate sadržane u rasponu

$$[x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{pr}, z_{st}]$$

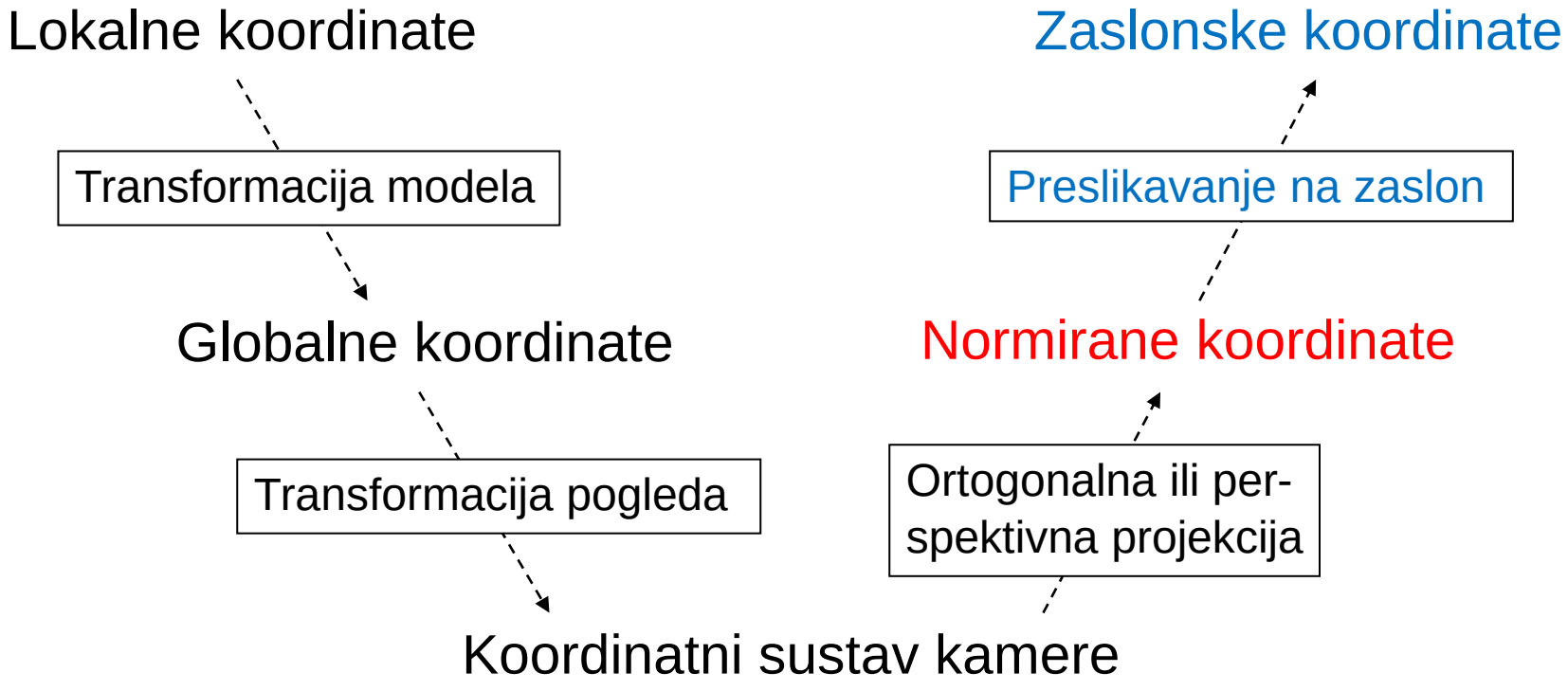
pri čemu su z_{pr} i z_{st} udaljenosti do prednje i stražnje odrezujuće plohe transformiraju u raspon

$$[-1, 1] \times [-1, 1] \times [-1, 1]$$

Transformacija u normirane koordinate



Koordinatni sustavi i transformacije u 3D grafici - OpenGL



Transformacije x i y koordinate

- perspektivno projicirane koordinate x_p i y_p uvrštavamo u formule za transformaciju u normirane koordinate koje smo već izveli za ortogonalnu projekciju

$$x' = \frac{2}{x_{max} - x_{min}} x_p + \frac{x_{min} + x_{max}}{x_{min} - x_{max}}$$

$$y' = \frac{2}{y_{max} - y_{min}} y_p + \frac{y_{min} + y_{max}}{y_{min} - y_{max}}$$

Transformacije x i y koordinate

- uvrstimo x_p sa $d = z_{pr}$ i preformuliramo:

$$x' = \frac{2}{x_{max} - x_{min}} \frac{z_{pr}}{-z} x + \frac{x_{min} + x_{max}}{x_{min} - x_{max}} \frac{-z}{-z}$$

$$x' = \frac{1}{-z} \left(\frac{2\, z_{pr}}{x_{max} - x_{min}} x + \frac{x_{min} + x_{max}}{x_{max} - x_{min}} z \right)$$

- za transformiranu koordinatu y' dobijemo:

$$y' = \frac{1}{-z} \left(\frac{2\, z_{pr}}{y_{max} - y_{min}} y + \frac{y_{min} + y_{max}}{y_{max} - y_{min}} z \right)$$

Transformacije x i y koordinate

- znamo kako realizirati dijeljenje sa $-z$ u homogenim koordinatama: moramo postaviti $w' = -z$
- u matričnom zapisu to znači da posljednji redak matrice tranformacije mora biti $[0, 0, -1, 0]$
- u zagradama imamo linearnu kombinacije x i z te linearnu kombinaciju y i z, što također možemo napisati kao prvi i drugi redak matrice transformacije koja je prikazana na sljedećem slajdu

Perspektivna projekcija u normirane koordinate

- matrica transformacije u homogenim koordinatama – nedostaje transformacija za z'

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} \frac{2\,z_{pr}}{x_{max}-x_{min}} & 0 & \frac{x_{max}+x_{min}}{x_{max}-x_{min}} & 0 \\ 0 & \frac{2\,z_{pr}}{y_{max}-y_{min}} & \frac{y_{max}+y_{min}}{y_{max}-y_{min}} & 0 \\ 0 & 0 & \gamma\ ? & \delta\ ? \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformacija z koordinate

- transformacija z-koordinate mora biti oblika

$$z' = \frac{1}{-z}(\gamma z + \delta) = -\gamma - \frac{\delta}{z}$$

- dvije nepoznanice γ i δ možemo odrediti iz uvjeta da se koordinata $-z_{pr}$ prednje odrezujuće plohe mora preslikati u -1, a $-z_{st}$ stražnje plohe u 1

$$-1 = -\gamma - \frac{\delta}{-z_{pr}} \qquad \& \qquad 1 = -\gamma - \frac{\delta}{-z_{st}}$$

Transformacija z koordinate

- rješavanjem dobivamo

$$\gamma = 1 + \frac{\delta}{z_{pr}} \quad \Rightarrow \quad 1 = -1 + \delta\left(\frac{1}{z_{st}} - \frac{1}{z_{pr}}\right)$$

$$\delta = \frac{2\, z_{pr}\, z_{st}}{z_{pr} - z_{st}}$$

$$\gamma = 1 + \frac{2\ z_{st}}{z_{pr} - z_{st}} = \frac{z_{pr} + z_{st}}{z_{pr} - z_{st}}$$

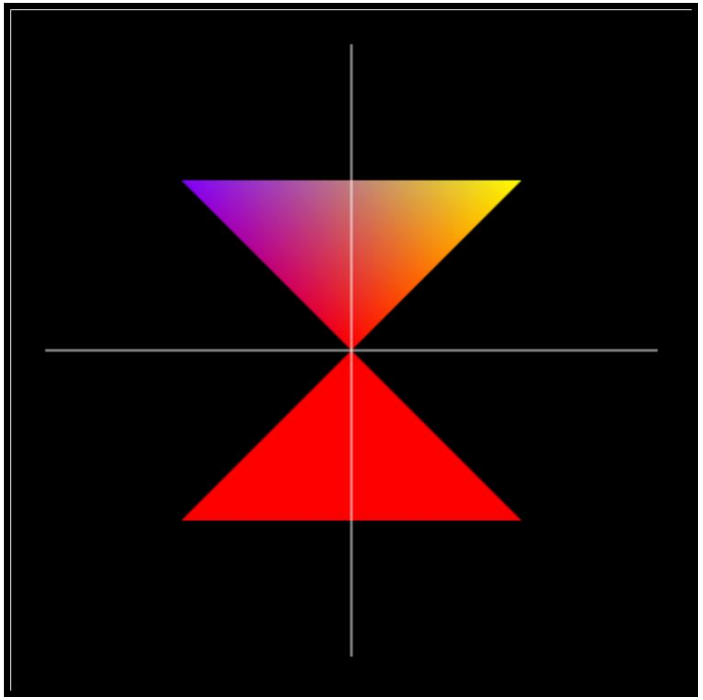
Perspektivna projekcija u normirane koordinate

- matrica transformacije u homogenim koordinatama

$$\begin{bmatrix} \frac{2\,z_{pr}}{x_{max}-x_{min}} & 0 & \frac{x_{max}+x_{min}}{x_{max}-x_{min}} & 0 \\ 0 & \frac{2\,z_{pr}}{y_{max}-y_{min}} & \frac{y_{max}+y_{min}}{y_{max}-y_{min}} & 0 \\ 0 & 0 & \frac{z_{pr}+z_{st}}{z_{pr}-z_{st}} & \frac{2\,z_{pr}\,z_{st}}{z_{pr}-z_{st}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

WebGL & OpenGL ES

Još malo o spremnicima…



Vertex
Array
Object

Vertex Array Object (VAO)

- OpenGL ES je stroj stanja: kod poziva rutina za iscrtavanje koristi se spremnik s podacima koji je posljednji vezan (engl. *bind*)
- da bi se omogućilo kreiranje i korištenje više spremnika s različitim skupovima podataka uveden je Vertex Array Object (VAO) koji pamti pojedina stanja
- prije iscrtavanje treba vezati odgovarajući VAO

Primjer 8.1.

- iz RG-primjer8-1-VAO.html
- skup podataka opisan s leptirVAO

```
leptirVAO = gl.createVertexArray();
gl.bindVertexArray(leptirVAO);
gl.bindBuffer(gl.ARRAY_BUFFER, gl.createBuffer());
gl.enableVertexAttribArray(GPUprogram1.a_vrhXYZ);
gl.enableVertexAttribArray(GPUprogram1.a_boja);
gl.vertexAttribPointer(GPUprogram1.a_vrhXYZ, 3,
  gl.FLOAT, false, 24, 0);
gl.vertexAttribPointer(GPUprogram1.a_boja, 3,
  gl.FLOAT, false, 24, 12);
gl.bufferData(gl.ARRAY_BUFFER, new
  Float32Array(vrhoviLeptira.flat()), gl.STATIC_DRAW);
```

Primjer 8.1.

- drugi skup podataka opisan s koordOsiVAO

```
koordOsiVAO = gl.createVertexArray();
gl.bindVertexArray(koordOsiVAO);
gl.bindBuffer(gl.ARRAY_BUFFER, gl.createBuffer());
gl.enableVertexAttribArray(GPUprogram1.a_vrhXYZ);
gl.vertexAttribPointer(GPUprogram1.a_vrhXYZ, 3,
  gl.FLOAT, false, 12, 0);
gl.bufferData(gl.ARRAY_BUFFER, new
  Float32Array(vrhoviKoordOsi.flat()), gl.STATIC_DRAW);
gl.vertexAttrib3f(GPUprogram1.a_boja, 1, 1, 1);
```

Primjer 8.1.

- iscrtavanja dva različita skupa podataka

```
// poveži sa spremnikom u kojem je leptir
gl.bindVertexArray(leptirVAO);
gl.drawArrays(gl.TRIANGLES, 0, vrhoviLeptira.length);

// poveži sa spremnikom u kojem su osi
gl.bindVertexArray(koordOsiVAO);
gl.drawArrays(gl.LINES, 0, vrhoviKoordOsi.length);
```