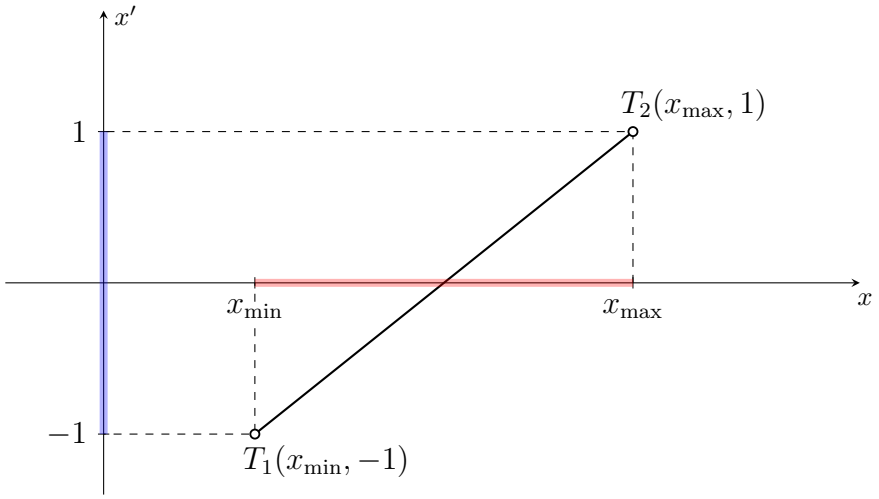
Normalizirane koordinate i ortogonalna projekcija

`OrtogonalnaProjekcija(xmin, xmax, ymin, ymax, zmin, zmax)`

Želimo da se u koordinatnom sustavu kamere kvadar

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

preslika na kocku $[-1, 1] \times [-1, 1] \times [-1, 1]$ tako da se odbacivanjem z koordinate dobije ortogonalna projekcija. Za segmente $[x_{\min}, x_{\max}]$ i $[y_{\min}, y_{\max}]$ postupamo na analogni način kao i u dvodimenzionalnom slučaju.



Tražimo jednadžbu pravca kroz dvije točke $T_1(x_{\min}, -1)$ i $T_2(x_{\max}, 1)$.

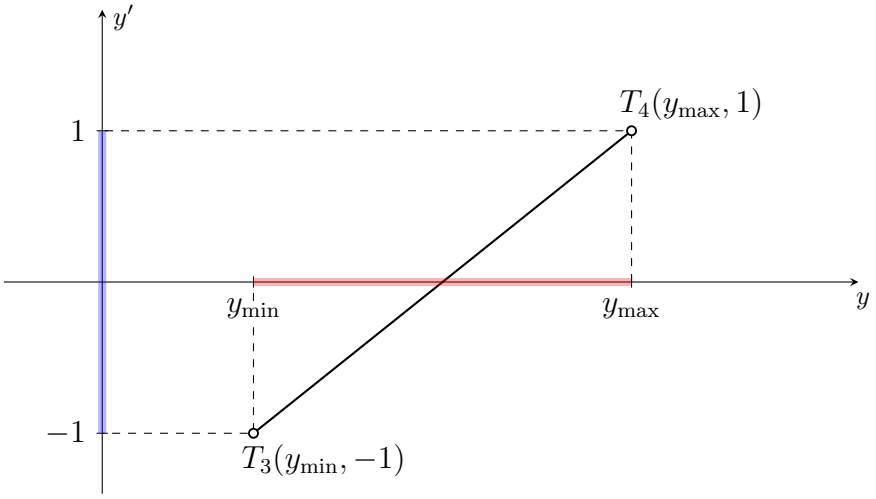
$$\begin{aligned} x' - (-1) &= \frac{1 - (-1)}{x_{\max} - x_{\min}}(x - x_{\min}) \\ x' + 1 &= \frac{2}{x_{\max} - x_{\min}}(x - x_{\min}) \\ x' &= \frac{2}{x_{\max} - x_{\min}} \cdot x - \frac{2}{x_{\max} - x_{\min}} \cdot x_{\min} - 1 \end{aligned}$$

Uvedemo oznake

$$s_x = \frac{2}{x_{\max} - x_{\min}}, \quad t_x = -s_x \cdot x_{\min} - 1 = \frac{x_{\min} + x_{\max}}{x_{\min} - x_{\max}}$$

pa slijedi

$$x' = s_x \cdot x + t_x.$$



Tražimo jednadžbu pravca kroz dvije točke $T_3(y_{\min}, -1)$ i $T_4(y_{\max}, 1)$.

$$y' - (-1) = \frac{1 - (-1)}{y_{\max} - y_{\min}}(y - y_{\min})$$

$$y' + 1 = \frac{2}{y_{\max} - y_{\min}}(y - y_{\min})$$

$$y' = \frac{2}{y_{\max} - y_{\min}} \cdot y - \frac{2}{y_{\max} - y_{\min}} \cdot y_{\min} - 1$$

Uvedemo oznake

$$s_y = \frac{2}{y_{\max} - y_{\min}}, \quad t_y = -s_y \cdot y_{\min} - 1 = \frac{y_{\min} + y_{\max}}{y_{\min} - y_{\max}}$$

pa slijedi

$$y' = s_y \cdot y + t_y.$$

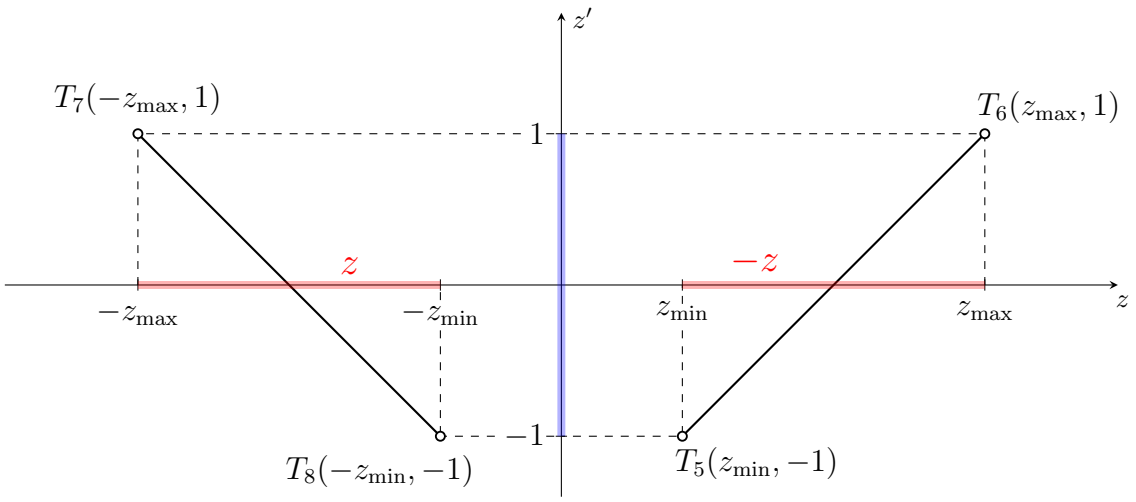
Što se tiče segmenta $[z_{\min}, z_{\max}]$ postupamo na malo drukčiji način uz sljedeći dogovor. U koordinatnom sustavu kamere z-os predstavlja smjer gledanja kamere. Svi objekti (ili dijelovi objekata) čija je "udaljenost" od kamere između $z_{\min}$ i $z_{\max}$ bit će vidljivi na ekranu. Stoga je razumno pretpostaviti da je $z_{\min} \geqslant 0$ i $z_{\max} \geqslant 0$. Naravno, ako je neki od tih brojeva manji od nule, tada zapravo želimo da se na ekranu prikaže dio 3D scene koji se nalazi iza kamere. To neće biti nikakva programska greška, osim što možda nije logično da prikazujemo dio scene koji kamera ne obuhvaća. Uz takav dogovor je $z_{\min} \geqslant 0$ i $z_{\max} \geqslant 0$ jer z-koordinatu interpretiramo kao "udaljenost" od kamere. Ovdje pod pojmom udaljenosti od kamere ne mislimo na standardnu euklidsku udaljenost, nego samo na udaljenost u smjeru z-osi, tj. na udaljenost u smjeru gledanja kamere, što je zapravo apsolutna vrijednost z-koordinate.

Međutim, u našem koordinatnom sustavu kamere, kamera se nalazi u ishodištu i gleda prema negativnom dijelu z-osi. To zapravo znači da svi objekti koji se nalaze ispred kamere u koordinatnom sustavu kamere imaju negativne z-koordinate. Stoga je uz ovakav dogovor zapravo

$$-z \in [z_{\min}, z_{\max}]$$

jer bismo inače prikazivali objekte iza kamere, tj. dobili bismo prazni canvas ukoliko se iza kamere ne bi nalazilo ništa, što i jest najčešći slučaj u jednostavnijim 3D scenama u kojima nema puno objekata.

Stoga na z' i $-z$ primjenjujemo istu formulu kao što smo to napravili ranije za x' i x te y' i y.



Tražimo jednadžbu pravca kroz dvije točke $T_5(z_{\min}, -1)$ i $T_6(z_{\max}, 1)$ i umjesto z pišemo $-z$.

$$z' - (-1) = \frac{1-(-1)}{z_{\max} - z_{\min}}(-z - z_{\min})$$

$$z' + 1 = \frac{2}{z_{\max} - z_{\min}}(-z - z_{\min})$$

$$z' = \frac{2}{z_{\min} - z_{\max}} \cdot z + \frac{2}{z_{\min} - z_{\max}} \cdot z_{\min} - 1$$

Uvedemo oznake

$$s_z = \frac{2}{z_{\min} - z_{\max}}, \quad t_z = s_z \cdot z_{\min} - 1 = \frac{z_{\min} + z_{\max}}{z_{\min} - z_{\max}}$$

slijedi

$$z' = s_z \cdot z + t_z.$$

Tražena transformacija zadana je u homogenim koordinatama matricom

$$\begin{bmatrix} s_x & 0 & 0 & t_x \\ 0 & s_y & 0 & t_y \\ 0 & 0 & s_z & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{x_{\max}-x_{\min}} & 0 & 0 & \frac{x_{\min}+x_{\max}}{x_{\min}-x_{\max}} \\ 0 & \frac{2}{y_{\max}-y_{\min}} & 0 & \frac{y_{\min}+y_{\max}}{y_{\min}-y_{\max}} \\ 0 & 0 & \frac{2}{z_{\min}-z_{\max}} & \frac{z_{\min}+z_{\max}}{z_{\min}-z_{\max}} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Napomena. Mogli smo transformaciju za z koordinatu napraviti na isti način kao što smo to napravili za x i y koordinate. Međutim, onda bismo svuda u kodu morali za z varijablu pisati negativne granice. Na primjer, umjesto

```
OrtogonalnaProjekcija(-19, 19, -12, 19, 5, 100)
```

trebali bismo pisati

```
OrtogonalnaProjekcija(-19, 19, -12, 19, -100, -5).
```

Sve je to zbog toga jer u našoj implementiranoj metodi `PostaviKameru` kamera gleda prema negativnom dijelu z-osi. Mogli smo u toj metodi staviti da z-os od kamere ima suprotnu orijentaciju od z-osi globalnog koordinatnog sustava (to je ovisilo o redoslijedu vektorskog množenja dva vektora) i u tom slučaju ne bismo morali mijenjati predznak z koordinati jer bi objekti

ispred kamere imali pozitivne z koordinate. Naravno, nismo trebali mijenjati predznak z koordinati niti u ovom našem slučaju, samo bismo onda morali u metodi `OrtogonalnaProjekcija` pisati negativne granice za z varijablu ako bismo željeli prikazati objekte koji su ispred kamere.

Čemu ova dulja diskusija? Zapravo se želi reći da je sve stvar pristupa i dogovora. Jedan predznak može bitno promijeniti cijelu priču, samo treba razumjeti matematičku pozadinu, a ne odmah reći da je to bug u programu. Prednost shadera jest u tome da oni o tome ništa ne znaju, mi im moramo sve pripremiti. Na koji način ćemo to pripremiti jest naš izbor pa onda možemo odabrati onaj pristup koji je nama draži. Nedostatak svega toga (ako možemo to tako reći) jest da moramo razumjeti neke matematičke principe i ne se izgubiti u različitim pristupima za rješavanje istog problema. Osobama kojima matematika nije jača strana to može predstavljati određeni problem. Međutim, zato postoje mnoge gotove biblioteke (poput `glMatrix` za `JavaScript`) koje se mogu koristiti za rješavanje takvih problema. Naravno, u tom slučaju moramo proučiti njihovu dokumentaciju i prilagoditi se onom pristupu kojeg one koriste. Dakle, izjava *"Objekti ispred kamere imaju negativne z koordinate."* može i ne mora biti istinita, sve ovisi o pristupu koji se koristi.

`WebGL` koristi sličan pristup prilikom korištenja metode `gl.viewport`. Ako samo direktno koristimo normalizirane koordinate i uključimo spremnik dubine, objekti koji imaju veće z koordinate se nalaze iza objekata koji imaju manje z koordinate. Drugim riječima, prilikom projiciranja na ekran, ravnina $z = -1$ je najbliža našem oku, a ravnina $z = 1$ je najudaljenija od našeg oka. Još zornije, ravnina $z = 0$ je naš ekran, ravnina $z = 1$ je unutar ekrana, a ravnina $z = -1$ je ispred ekrana. To je samo zorni opis, `WebGL`-u to ništa ne znači. On samo kocku $[-1, 1] \times [-1, 1] \times [-1, 1]$ pomoću `gl.viewport` prevodi u zaslonske koordinate. `WebGL` nema pojma o kameri, nema pojma o transformacijama, nema pojma o ortogonalnoj projekciji, nema pojma o perpsektivnoj projekciji. Što se tiče transformacija, `WebGL` jedino zna kako kocku $[-1, 1] \times [-1, 1] \times [-1, 1]$ prevesti u zaslonske koordinate. Naša briga je da 3D scenu, tj. dio 3D scene koji želimo prikazati na ekranu dovedemo unutar kocke $[-1, 1] \times [-1, 1] \times [-1, 1]$. Kako ćemo mi to napraviti, to je naša stvar. Različite vrste transformacija i projekcija koje smo do sada radili, pojam kamere koji smo uveli u 3D scenu su zapravo samo etape koje nam olakšavaju i zorno približavaju taj proces dovođenja željenog dijela 3D scene unutar kocke $[-1, 1] \times [-1, 1] \times [-1, 1]$.

`WebGL` svaku točku $(x, y, z) \in [-1, 1] \times [-1, 1] \times [-1, 1]$ prevodi u zaslonske koordinate tako da zanemari z komponentu. Dakle, `WebGL` točku (x, y) prevodi u zaslonske koordinate na slični način kako smo to mi radili na prvim laboratorijskim vježbama. Pritom koristi parametre koji su navedeni u metodi `gl.viewport`. `WebGL` ne zna ništa o ortogonalnoj i perpsektivnoj projekciji. Naše formule za dovođenje 3D scene unutar kocke $[-1, 1] \times [-1, 1] \times [-1, 1]$ moraju osigurati da je na x i y komponentama normaliziranih koordinata provedeno ortogonalno ili perspektivno projiciranje. Danas smo pokazali kako mora izgledati takva matrica transformacije u slučaju ortogonalnog projiciranja, a sljedeći put ćemo pokazati kako izgleda takva matrica transformacije za slučaj perspektivnog projiciranja. `WebGL` komponentu z koristi za spremnik dubine kako bi odredio koji objekt je ispred ili iza nekog drugog objekta. Prilikom spremanja z komponente u spremnik dubine `WebGL` može provesti dodatne transformacije na z komponentu kako bi dobio novu vrijednost z' koju onda zapravo spremi u spremnik dubine. Spomenute transformacije su ovisne o implementaciji u hardveru, a cilj takvih transformacija je osigurati što je moguće veću preciznost spremnika dubine.

`OrtogonalnaProjekcijaX(xmin, xmax, ymin, ymax, zmin, zmax, w, h)`

Želimo da se u koordinatnom sustavu kamere kvadar

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

preslika na kocku $[-1,1] \times [-1,1] \times [-1,1]$ tako da se odbacivanjem z koordinate dobije ortogonalna projekcija pri čemu se čuvaju proporcije slike prilikom preslikavanja iz normaliziranih koordinata na pravokutnik (canvas) $[0,w] \times [0,h]$. Za čuvanje proporcije slike po potrebi se podešava segment $[y_{\min}, y_{\max}]$.

U ovom slučaju mora vrijediti

$$\frac{x_{\max} - x_{\min}}{y_{\max} - y_{\min}} = \frac{w}{h}$$

iz čega slijedi

$$y_{\max} - y_{\min} = \frac{h}{w}(x_{\max} - x_{\min}).$$

Definiramo broj

$$k = \frac{\frac{h}{w}(x_{\max} - x_{\min}) - (y_{\max} - y_{\min})}{2}.$$

Sada interval $[y_{\min}, y_{\max}]$ zamijenimo s intervalom $[y_1, y_2]$ pri čemu su

$$y_1 = y_{\min} - k, \quad y_2 = y_{\max} + k.$$

Na kraju pozovemo `OrtogonalnaProjekcija(xmin, xmax, y1, y2, zmin, zmax)`.

`OrtogonalnaProjekcijaY(xmin, xmax, ymin, ymax, zmin, zmax, w, h)`

Želimo da se u koordinatnom sustavu kamere kvadar

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

preslika na kocku $[-1,1] \times [-1,1] \times [-1,1]$ tako da se odbacivanjem z koordinate dobije ortogonalna projekcija pri čemu se čuvaju proporcije slike prilikom preslikavanja iz normaliziranih koordinata na pravokutnik (canvas) $[0,w] \times [0,h]$. Za čuvanje proporcije slike po potrebi se podešava segment $[x_{\min}, x_{\max}]$.

U ovom slučaju mora vrijediti

$$\frac{x_{\max} - x_{\min}}{y_{\max} - y_{\min}} = \frac{w}{h}$$

iz čega slijedi

$$x_{\max} - x_{\min} = \frac{w}{h}(y_{\max} - y_{\min}).$$

Definiramo broj

$$k = \frac{\frac{w}{h}(y_{\max} - y_{\min}) - (x_{\max} - x_{\min})}{2}.$$

Sada interval $[x_{\min}, x_{\max}]$ zamijenimo s intervalom $[x_1, x_2]$ pri čemu su

$$x_1 = x_{\min} - k, \quad x_2 = y_{\max} + k.$$

Na kraju pozovemo `OrtogonalnaProjekcija(x1, x2, ymin, ymax, zmin, zmax)`.