I) Računalna 3D grafika iz računalne perspektive

Predrag Brođanac, V. gimnazija, Zagreb

Prikaz slike na zaslonu monitora često uzimamo zdravo za gotovo. Ne razmišljamo o načinu kako se primjerice trodimenzionalni objekti prikazuju na zaslonu monitora? Kako se dobije trodimenzionalnost u računalnim igrama?

U ne tako dalekim vremenima za prikaz grafike na računalu koristila se pravokutna projekcija. Nije postojala trodimenzionalnost, likovi su prikazivani plošno. Primjer takve računalne grafike susrećemo primjerice u igrici ***Super Mario***.

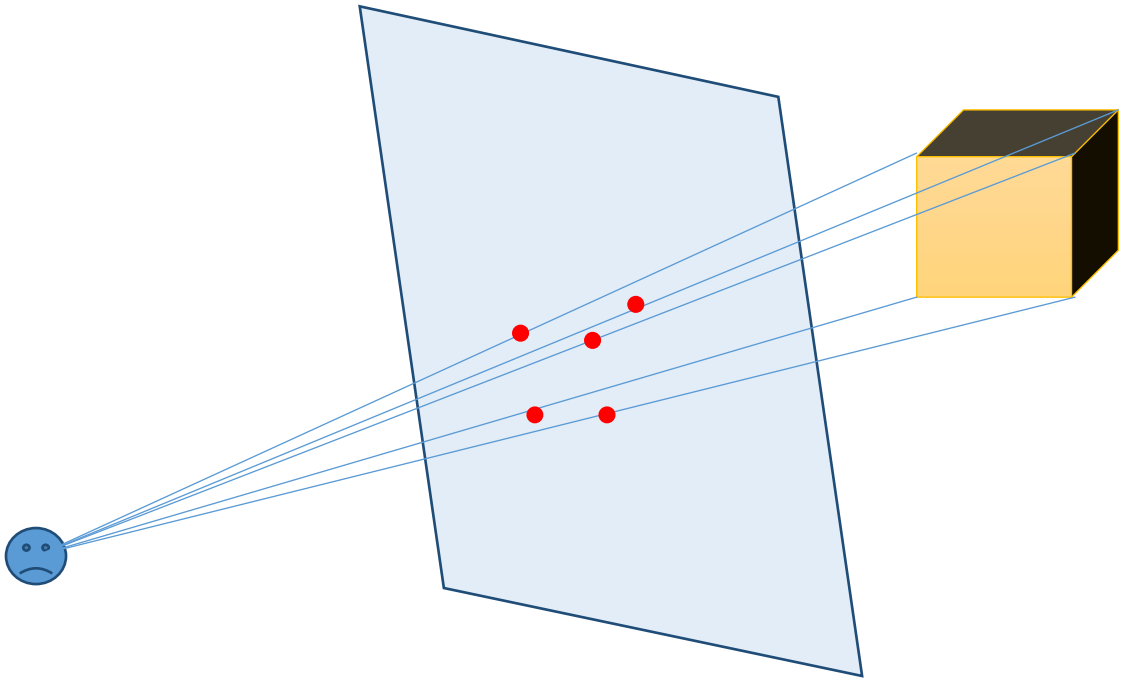
Slika 1. Pravokutna projekcija – Super Mario

U međuvremenu je računalna grafika napredovala pa tako u suvremenijim računalnim igrama susrećemo perspektivnu projekciju. Primjer igre u kojoj susrećemo perspektivnu projekciju jest ***Quake 4***.

Slika 2. Perspektivna projekcija – Quake 4

Općenito se postavlja pitanje kako dobiti trodimenzionalnost na zaslonu monitora? Kako trodimenzionalne objekte iz prostora prikazati na zaslonu dvodimenzionalnog ekrana?

Na dani problem možemo gledati na sljedeći način: na vertikalni stalak postavimo primjerice trodimenzionalnu rešetku kocke. Ispred kocke na nekoj udaljenosti postavimo staklo, a mi stanemo na nekoj udaljenosti ispred stakla tako da se staklo nalazi između nas i kocke. Na staklu pisaljkom zabilježimo mjesta na kojima vidimo vrhove trodimenzionalne rešetke kocke te dobivene točke spojimo linijama koje odgovaraju bridovima rešetke kocke. Grafički bismo to mogli prikazati sljedećom slikom:

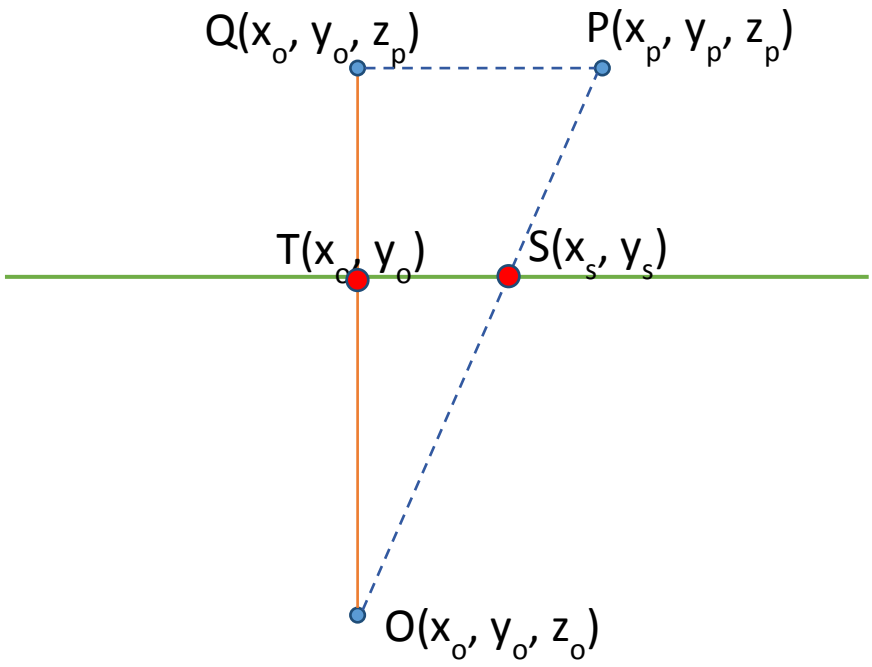


Pozicija točaka na staklu ovisit će o nizu parametara:

- Udaljenost gledatelja od stakla
- Udaljenost slike od stakla
- Pozicija (horizontalna/vertikalna) gledatelja s obzirom na sliku
- …

Staklo će u računalu zamijeniti zaslon monitora.

Promotrimo nešto pojednostavljenu situaciju. Neka imamo zamišljeni trodimenzionalni koordinatni sustav u kojem se nalazi staklo (zaslon), točka $P(x_p, y_p, z_p)$ koja se nalazi iza zaslona, a mi kao promatrači koji gledamo prema monitoru i točki P, promatramo iz točke $O(x_o, y_o, z_o)$. Zanima nas koje će koordinate ($S(x_s, y_s)$) na ekranu imati točka P iz perspektive nas kao promatrača.



Iz navedene skice lako uočavamo da je:

$$\frac{d(O,T)}{d(O,Q)} = \frac{d(T,S)}{d(Q,P)}$$

Odnosno:

$$d(T,S) = \frac{d(O,T) \cdot d(Q,P)}{d(O,Q)}$$

Iz navedene relacije slijedi:

$$|x_s - x_o| = \frac{|z_o| \cdot |x_p - x_o|}{|z_o + z_p|}$$

i

$$|y_s - y_o| = \frac{|z_o| \cdot |y_p - y_o|}{|z_o + z_p|}$$

Nakon ovih uvodnih razmatranja imamo sve spremno za ilustraciju projekcije u konkretnom programskom jeziku. Ilustraciju ćemo provoditi u programskom jeziku Python.

Za početak ćemo implementirati funkciju koja će koordinate točke iz trodimenzionalnog koordinatnog sustava pretvarati u pripadne koordinate u dvije dimenzije (točke ekrana). Neka su dani sljedeći parametri:

- xp, yp, zp – koordinate točke u trodimenzionalnom koordinatnom sustavu
- xo, yo, zo – koordinate točke iz koje gledamo
- S – širina ekrana u pikselima
- V – visina ekrana u pikselima

Funkcija će vraćati koordinate pripadne točke ekrana:

```
def iz3Du2D(xp, yp, zp, xo, yo, zo, S, V):
```

```
    xs = S / 2 + (zo * (xp - xo) / (zo + zp) + xo)
    ys = V / 2 - (zo * (yp - yo) / (zo + zp) + yo)
    return xs, ys
```

Pretpostavimo da imamo „platno za crtanje“ te naredbe Pythona koje nam omogućavaju crtanje osnovnih geometrijskih oblika na tom platnu:

- **create_line(x1, y1, x2, y2 [, parametri])** – crta dužinu s krajevima u točkama (x_1, y_1) i (x_2, y_2).
- **create_oval(x1, y1, x2, y2 [,parametri])** – crta elipsu koja je upisana u pravokutnik čiji je gornji lijevi vrh u točki s koordinatama (x_1, y_1), a donji desni u točki s koordinatama (x_2, y_2)

Na točku možemo gledati kao na elisu čija su velika i mala poluos jednake te neka je polumjer pripadne kružnice 2 piskela. Napišimo funkciju koja će zadanu točku *P (xp, yp, zp)* crtati na ekranu dimenzija *S x V* ako točku promatramo iz koordinate *O (xo, yo, zo)*:

```
def crtajTocku(xp, yp, zp, xo, yo, zo, S, V):
    xs, ys = iz3Du2D(xp, yp, zp, xo, yo, zo, S, V)
    PLATNO.create_oval(xs - 2, ys - 2, xs + 2, ys + 2, , fill = 'red')
    return
```

Nadalje napišimo funkciju koja će crtati dužinu *AB*, pri čemu je *A(xa, ya, za)* i *B(xb, yb, zb)*:

```
def linija(xa, ya, za, xb, yb, zb, xo, yo, zo, S, V):
    crtajTocku(xa, ya, za, xo, yo, zo, S, V )
    crtajTocku(xb, yb, zb, xo, yo, zo, S, V )
    x1, y1 = iz3Du2D(xa, ya, za, xo, yo, zo, S, V)
    x2, y2 = iz3Du2D(xb, yb, zb, xo, yo, zo, S, V)
    PLATNO.create_line(x1, y1, x2, y2)
    return
```

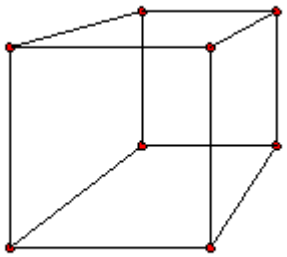
Sada imamo sve spremno za crtanje kocke koja se nalazi u prvom kvadrantu trodimenzionalnog sustava a čija je duljina stranice *a*:

```
def kocka(a, xo, yo, zo, S, V):
    linija(0, 0, 0, a, 0, 0, xo, yo, zo, S, V)
    linija(0, 0, 0, 0, 0, a, xo, yo, zo, S, V)
    linija(0, 0, 0, 0, a, 0, xo, yo, zo, S, V)
    linija(a, a, 0, 0, a, 0, xo, yo, zo, S, V)
    linija(a, a, 0, a, 0, 0, xo, yo, zo, S, V)
    linija(a, a, 0, a, a, a, xo, yo, zo, S, V)
    linija(a, 0, a, a, 0, 0, xo, yo, zo, S, V)
    linija(a, 0, a, a, a, a, xo, yo, zo, S, V)
    linija(a, 0, a, 0, 0, a, xo, yo, zo, S, V)
    linija(0, a, a, 0, a, 0, xo, yo, zo, S, V)
    linija(0, a, a, 0, 0, a, xo, yo, zo, S, V)
    linija(0, a, a, a, a, a, xo, yo, zo, S, V)
    return
```

Uz parametre:

- koordinate promatrača: *O(200, 150, 200)*
- dimenzije platna: 800 x 600

dobit ćemo sljedeću sliku:



Sličnu implementaciju mogli bismo napraviti i u drugim programskim jezicima, princip će biti skoro ekvivalentan, tj. uz neke razlike u naredbama programskog jezika.

Gornja razmatranja dala su nam kratak uvid u načine kako trodimenzionalne objekte projicirati na zaslon monitora. Ovo je abeceda svih programa koji unutar svog rada imaju trodimenzionalni prikaz. Trodimenzionalnost se dobiva generiranjem niza jednostavnih trodimenzionalnih objekata (najčešće trokuta). Kreiranje takvih složenih trodimenzionalnih objekata sastavljenih od trokuta s pripadnim bojama, njihovo kretanje, sjene i sl. nazivamo *renderiranjem*. Renderiranja kompleksnih trodimenzionalnih objekata u pozadini kriju znatan broj matematičkih operacija što zahtijeva veliku snagu centralnog procesora. Upravo se iz tih razloga u današnja računala ugrađuju grafičke kartice s vlastitim vrlo snažnim, matematičkim procesorima i radnim memorijama te se na taj način rasterećuje centralni procesor i centralna memorija, a renderiranja kompleksnih objekata znatno su brža i teku glatko.