Računalna grafika – laboratorijske vježbe 2

Damir Horvat

Fakultet organizacije i informatike, Varaždin

Sadržaj

Crtanje elipse

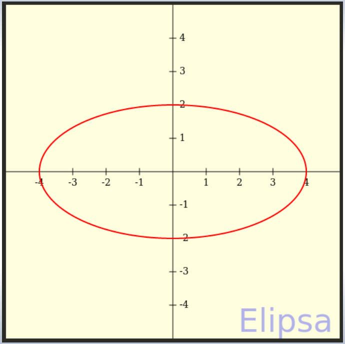
Primjer 1. – Crtanje elipse

- Pomoću `GKS` klase nacrtajte elipsu zadanu parametarskim jednadžbama

$$x = a\cos t, \quad y = b\sin t.$$

- Stavite $a = 4$ i $b = 2$.
- Što se događa s elipsom kada se mijenjaju dimenzije canvasa, posebno kada se ne čuva omjer širine i visine?
- Stavite $a = b = 3$, mijenjajte dimenzije canvasa i mijenjajte granice na koordinatnim osima. Koji problem se može javiti vezano uz vjerni prikaz kružnice?

Screenshot elipse



Nadogradnja `GKS` klase s novim konstruktorima

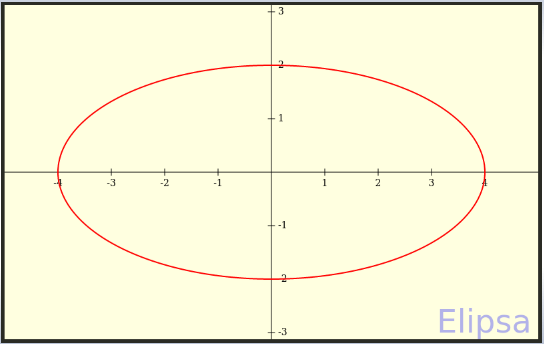
Novi konstruktor u `GKS` klasi

Dodajte konstruktor `GKS(platno, xmin, xmax)` koji radi sljedeće:

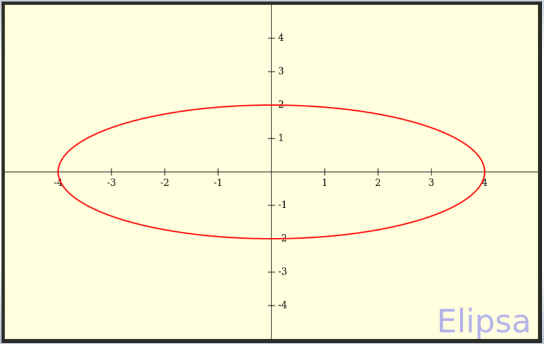
- faktor s_x izračuna po formuli
- $s_y = -s_x$
- pomak p_x izračuna po formuli
- $p_y = \frac{h}{2}$

Možete modificirati postojeći konstruktor jer `JavaScript` dozvoljava nedefinirane varijable ili možete novi konstruktor kreirati preko statične metode.

Screenshot elipse s novim konstruktorom



Screenshot elipse sa starim konstruktorom



Implementacija `MT2D` klase

Primjer 2. – Implementacija `MT2D` klase

- Implementirajte klasu `MT2D` matričnih reprezentacija geometrijskih transformacija u 2D.
- Pripadne metode moraju generirati matrice transformacija u homogenim koordinatama, dakle 3×3 matrice.
- Implementirajte metode za translaciju, skaliranje, rotaciju oko ishodišta, zrcaljenja s obzirom na koordinatne osi, identitetu, smicanje duž koordinatnih osi.

Ideja implementacije `MT2D` klase

- Klasa neka ima jedan (privatni) atribut `matrica` u koji će se svaki put iznova spremati novogenerirana matrica određene transformacije.
- Omogućite da se iz drugih klasa može preuzeti vrijednost atributa `matrica`.
- Konstruktor `MT2D` klase neka atribut `matrica` postavi na jediničnu matricu.
- Svaki put kada se pozove metoda koja generira matricu neke geometrijske transformacije, ta metoda mijenja vrijednost atributa `matrica` u vrijednost koju je ona izgenerirala.

Imena metoda i njihove matrice

- Identiteta – postavlja matricu transformacije na jediničnu matricu

```
identitet()
```

- Translacija za vektor (p_x, p_y)

```
pomakni(px, py)
```

- Skaliranje s faktorima s_x i s_y

```
skaliraj(sx, sy)
```

- Matrica identitete

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Matrica translacije

$$\begin{bmatrix} 1 & 0 & p_x \\ 0 & 1 & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

- Matrica skaliranja

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Imena metoda i njihove matrice

- Zrcaljenje na osi x

```
zrcaliNaX()
```

- Zrcaljenje na osi y

```
zrcaliNaY()
```

- Rotacija oko ishodišta za navedeni kut u stupnjevima

```
rotiraj(kut)
```

- Smicanje s faktorima $\operatorname{tg}\alpha$ i $\operatorname{tg}\beta$

```
smicanje(alpha, beta)
```

- Matrice zrcaljenja na koordinatnim osima

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Matrica rotacije

$$\begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Matrica smicanja

$$\begin{bmatrix} 1 & \operatorname{tg}\beta & 0 \\ \operatorname{tg}\alpha & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Primjeri implementacija nekih metoda

```
class MT2D {
  constructor() {
   //code
  }

  //transformacije
  pomakni(px, py) {
  let m = [[1,0,px],[0,1,py],[0,0,1]];
    this._matrica = m;
  }
  ...
}
```

`trans(m)` metoda

Primjer 3. – `trans(m)` metoda

- U klasu GKS dodajte metodu `trans(m)` kojom se zadaje matrica transformacije koja se primjenjuje prije crtanja u globalnim koordinatama.
- To je zapravo transformacija iz lokalnih u globalne koordinate – po defaultu postaviti na identitet, tj. jediničnu matricu.

Ideja implementacije `trans(m)` metode

- Klasi `GKS` treba dodati još jedan (privatni) atribut `matrica` u kojemu će se čuvati matrica transformacije i njegovu vrijednost po defaultu odmah postaviti na jediničnu matricu.
- Metoda `trans(m)` zapravo treba iz objekta `m` tipa `MT2D` preuzeti vrijednost atributa `matrica` i tu vrijednost pridružiti atributu `matrica` u klasi `GKS`.
- U metodama `postaviNa` i `linijaDo` prije transformacije točaka u koordinate piksela, treba na zadane točke primijeniti matricu transformacije koja se nalazi u atributu `matrica`, a tek nakon toga tako transformirane točke prevesti u koordinate piksela.

Računanje slike vektora odnosno radijvektora

- Neka je $f : V \to V$ linearni operator, a $\mathcal{B}$ neka baza za vektorski prostor V.
- Neka je $F_{\mathcal{B}}$ matrica operatora f u bazi $\mathcal{B}$, a $X_{\mathcal{B}}$ koordinatna matrica vektora $x \in V$ u bazi $\mathcal{B}$.
- Tada je koordinatna matrica vektora $f(x)$ u bazi $\mathcal{B}$ matrica $Y_{\mathcal{B}}$ za koju vrijedi

$$Y_{\mathcal{B}} = F_{\mathcal{B}} \cdot X_{\mathcal{B}}.$$

- Nama je ovdje dovoljno da se ograničimo na kanonsku bazu. Stoga sliku neke točke pri nekoj linearnoj transformaciji dobijemo tako da matricu te transformacije pomnožimo s koordinatnom matricom te točke, tj. pripadnog radijvektora. Pritom koordinatne matrice vektora zapisujemo u obliku jednostupčanih matrica.

Računanje slike vektora odnosno radijvektora

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_{02} \\ a_{12} \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \left[\begin{array}{cc|c} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ \hline 0 & 0 & 1 \end{array}\right] \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

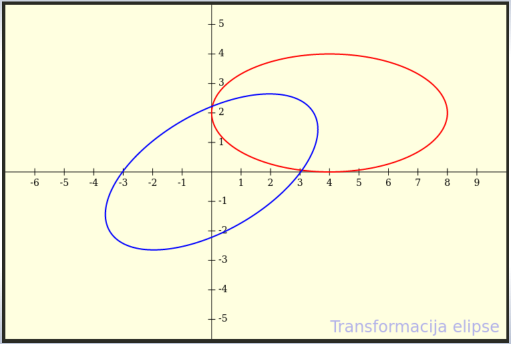
$$\begin{aligned} x' &= a_{00}x + a_{01}y + a_{02} \\ y' &= a_{10}x + a_{11}y + a_{12} \end{aligned}$$

$$\begin{aligned} x_{\text{pix}} &= s_x \cdot x' + p_x \\ y_{\text{pix}} &= s_y \cdot y' + p_y \end{aligned}$$

Primjer 4. – transformacije elipse

- Definirajte globalni koordinatni sustav u kojemu je $x \in [-7, 10]$, a granice na y-osi se određuju automatski simetrično s obzirom na ishodište tako da jedinične dužine na koordinatnim osima budu uvijek iste duljine.
- Pomoću odgovarajućih geometrijskih transformacija na istoj slici nacrtajte sljedeće krivulje:
  - elipsu s poluosima $a = 4$ i $b = 2$ paralelnim s koordinatnim osima te središtem u točki $(4, 2)$
  - elipsu s poluosima $a = 4$ i $b = 2$ sa središtem u ishodištu, a velika os elipse zatvara kut od 30° s pozitivnim dijelom x-osi

Screenshot 4. primjera



Neki dijelovi koda

```
var gks = //stavi neki implementirani GKS konstruktor
var mat = new MT2D();

//generiraj translaciju za vektor (4,2)
mat.pomakni(4,2);
//primijeni translaciju na vrhove prije crtanja
gks.trans(mat);
//nacrtaj translatiranu elipsu
elipsa(gks,4,2);
```

Kompozicija geometrijskih transformacija

- Sjetimo se iznimno važnog svojstva linearnih operatora za cjelokupno čovječanstvo

Kompozicija linearnih operatora

Uz odabrane baze, kompoziciji linearnih operatora odgovara produkt njihovih matričnih prikaza.

- Upravo je to glavna motivacija za onako "čudno" množenje matrica.

Kompozicija geometrijskih transformacija

- Specijalno, neka su f i g dvije geometrijske transformacije u ravnini, a F i G njihovi matrični zapisi u homogenim koordinatama.
- Kompoziciji $f \circ g$ odgovara matrica FG u homogenim koordinatama.
- Kompozicija geometrijskih transformacija nije komutativna operacija, tj. općenito je
$$f \circ g \neq g \circ f$$
- Isto tako, produkt matrica nije komutativna operacija, tj. općenito je
$$FG \neq GF$$

Kompozicija geometrijskih transformacija

- Dakle, bitan je redoslijed izvođenja geometrijskih transformacija. Redoslijed izvođenja se obavlja zdesna ulijevo s obzirom na zapis kompozicije funkcija.
- U složenoj geometrijskoj transformaciji $f \circ g$, najprije se na objektu izvede transformacija g, a tek nakon toga na novotransformiranom objektu se izvede transformacija f.
- U složenoj geometrijskoj transformaciji $g \circ f$, najprije se na objektu izvede transformacija f, a tek nakon toga na novotransformiranom objektu se izvede transformacija g.

Kompozicija geometrijskih transformacija

- Kompozicija geometrijskih transformacija je asocijativna operacija, tj.

$$(f \circ g) \circ h = f \circ (g \circ h).$$

  Stoga možemo pisati bez zagrada: $f \circ g \circ h$.

- Množenje matrica je asocijativna operacija, tj.

$$(FG)H = F(GH).$$

  Stoga možemo pisati bez zagrada: FGH.

`mult(m)` metoda u `MT2D` klasi

Primjer 5. – `mult(m)` metoda

- U klasu `MT2D` dodajte mogućnost kompozicije geometrijskih transformacija, tj. množenja odgovarajućih matrica transformacija preko `mult(m)` metode.
- Metoda `mult(m)` množi trenutnu matricu transformacije zdesna s matricom `m`.

Novosti u implementaciji klase `MT2D`

- U klasu treba dodati metodu `mult(m)` koja će trenutnu vrijednost atributa `matrica` zamijeniti s produktom `matrica*m`.
- Svaka od implementiranih metoda koja generira matricu određene transformacije više na izlazu ne smije prebrisati trenutnu vrijednost atributa `matrica` svojom generiranom matricom, već mora trenutnu vrijednost atributa `matrica` s desne strane pomnožiti svojom generiranom matricom.

Novosti u implementaciji klase MT2D

- Na taj način je omogućena kompozicija transformacija u `MT2D` klasi. Pritom, zadnje napisana transformacija u kodu se zapravo u stvarnosti prva izvršava na nekom objektu, dok se transformacija koja je napisana prva u kodu zapravo izvršava posljednja. To je zbog toga što obavljamo množenje s desne strane.
- Metoda `identitet()` i dalje na izlazu samo vraća jediničnu matricu jer ćemo pomoću te metode moći zaboraviti sve dosadašnje kompozicije transformacija u slučaju da želimo krenuti iz početka s novim lancem transformacija.

Množenje matrica

- Neka je $A = [a_{ij}]$ matrica tipa (m, n).
- Neka je $B = [b_{ij}]$ matrica tipa (n, p).
- Tada je $AB = [c_{ij}]$ matrica tipa (m, p) i vrijedi

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, \quad i = 1, 2, \ldots, m, \quad j = 1, 2, \ldots, p.$$

Predložak za metodu `mult`

```
mult(m) {
    let m1 = [[0,0,0],[0,0,0],[0,0,0]];
    for (let i = 0; i < 3; i++) {
      for (let j = 0; j < 3; j++) {
        for (let k = 0; k < 3; k++) {
          //code
        }
      }
    }
    this._matrica = m1;
  }
```

Novi kod za metodu pomakni

- stari kod metode pomakni

```
pomakni(px, py) {
  let m = [[1,0,px],[0,1,py],[0,0,1]];
  this._matrica = m;
}
```

- novi kod metode pomakni

```
pomakni(px, py) {
  let m = [[1,0,px],[0,1,py],[0,0,1]];
  this.mult(m);
}
```

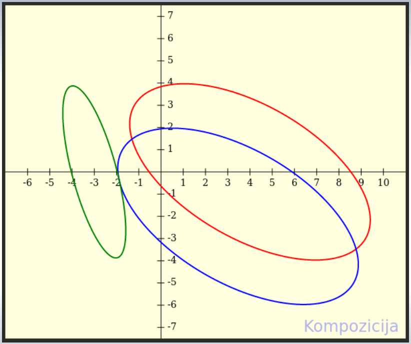
- Analogno se promijene kodovi ostalih metoda. Jedino metodu `identitet()` ostavimo nepromijenjenu.

Kompozicija transformacija

Primjer 6. – kompozicija transformacija

- Primjenom odgovarajućih kompozicija transformacija nacrtajte sljedeće elipse u navedenim bojama:
- crvenom bojom: elipsu s poluosima $a = 6$ i $b = 3$ koja je najprije zarotirana za -30° oko ishodišta, a nakon toga translatirana za vektor $(4, 0)$
- plavom bojom: elipsu s poluosima $a = 6$ i $b = 3$ koja je najprije translatirana za vektor $(4, 0)$, a nakon toga zarotirana za -30° oko ishodišta
- zelenom bojom: elipsu s poluosima $a = 4$ i $b = 1$ koja je najprije zarotirana za 75° oko ishodišta, nakon toga translatirana za 3 jedinice udesno i na kraju zrcaljena na osi y.

Screenshot 6. primjera



Dio koda za 6. primjer

```
    ...
var gks = //stavi neki implementirani GKS konstruktor
var mat = new MT2D();
    ...
kan.strokeStyle = "blue";
mat.identitet();
mat.rotiraj(-30);
mat.pomakni(4,0);
gks.trans(mat);
elipsa(gks,6,3);
    ...
```

Dodatak

Nadogradnja `GKS` klase s konstruktorom `EqualX`

Dodavanje konstruktora `EqualX` u `GKS` klasu

- Dodajte u `GKS` klasu konstruktor `EqualX` kao statičnu metodu oblika `static EqualX(canvas, w, h, xmin, xmax) {...}`
- Taj konstruktor uvijek postavlja na koordinatne osi jedinične dužine istih duljina u pikselima, bez obzira na dimenzije canvasa i veličinu prirodnog koordinatnog sustava.
- Raspon granica na y-osi automatski se određuje simetrično s obzirom na ishodište na temelju danih granica na x-osi i dimenzija canvasa tako da duljine jediničnih dužina budu jednake na obje osi.

Teoretska pozadina za konstruktor `EqualX`

Mora biti osigurano da je $s_x = -s_y$, tj.

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

iz čega slijedi

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

Stoga je

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

Implementacija konstruktora EqualX

```
class GKS {
    constructor(canvas, w, h, xmin, xmax, ymin, ymax) {
        //code
    }

    static EqualX(canvas, w, h, xmin, xmax) {
        //code
        return new GKS(canvas, w, h, xmin, xmax, ymin, ymax);
    }

   //ostale metode
}

//pozivanje staticne metode
var gks = GKS.EqualX(kan,w,h,xmin,xmax);
```

Nadogradnja `GKS` klase s konstruktorom `EqualY`

Dodavanje konstruktora `EqualY` u `GKS` klasu

- Dodajte u `GKS` klasu konstruktor `EqualY` kao statičnu metodu oblika `static EqualY(canvas, w, h, ymin, ymax) {...}`
- Taj konstruktor uvijek postavlja na koordinatne osi jedinične dužine istih duljina u pikselima, bez obzira na dimenzije canvasa i veličinu prirodnog koordinatnog sustava.
- Raspon granica na x-osi automatski se određuje simetrično s obzirom na ishodište na temelju danih granica na y-osi i dimenzija canvasa tako da duljine jediničnih dužina budu jednake na obje osi.

Teoretska pozadina za konstruktor `EqualY`

Mora biti osigurano da je $s_x = -s_y$, tj.

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

iz čega slijedi

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

Stoga je

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