Računarska grafika

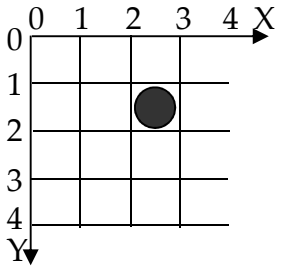
JavaFX - transformacije

Uvod

- Geometrijskim transformacijama objekata se menja njihov
  - položaj ← translacija
  - orijentacija ← rotacija
  - veličina, oblik ← skaliranje (Sx≠Sy), smicanje
- Geometrijske transformacije u Java FX biblioteci
  - primenjuju se na geometrijske primitive
- Primitive na koje se primenjuju transformacije mogu biti:
  - čvorovi scene
    - transformacija je atribut čvora, menja mu geometrijske karakteristike
  - primitive na kanvasu
    - transformacija je modalni atribut kanvasa
- Modalni atribut
  - primenjuje se tekuća vrednost na primitive koje se crtaju na kanvasu

Koordinate tačaka i piksela

- U JavaFX se koristi levi pravougli koordinatni sistem
- Koordinate tačke su realni brojevi dvostruke tačnosti (`double`)
- Pikseli se nalaze u ćelijama rešetke sa celobrojnim koord.
  - celobrojne koordinate označavaju gornji levi ugao ćelije piksela
- Prikazani piksel



  - ima (celobrojne) koordinate (2,1),
  - centar piksela se nalazi na (realnim) koordinatama (2.5,1.5)
  - sve tačke sa koordinatama 2≤x<3 i 1≤y<2 se preslikavaju u dati piksel

Transformacija koord. sistema

- Koordinatni sistem (KS) roditeljskog čvora
  - KS grupe u kojoj se nalazi objekat (primitiva-oblik ili složeni čvor-grupa)
- Na osnovnom nivou, grupa predstavlja koreni čvor scene
  - KS korenog čvora je KS scene kojoj je pridružen odgovarjući graf scene
- Svaki oblik ima svoj lokalni KS u kojem se zadaju koord. oblika
- Podrazumevano:
  - KS oblika se poklapa sa KS roditelja
- Prilikom transformacije objekta transformiše se njegov lokalni KS
  - objekat je vezan za njega
  - koordinate tačaka objekta u lokalnom KS se ne menjaju
  - menjaju se koordinate tačaka objekta u KS roditeljskog čvora
- Transformacija objekta se svodi na preslikavanje koordinata
  - iz njegovog lokalnog KS u KS roditeljskog čvora

Načini transformacija u JavaFX (1)

- Postoje dva načina za specificiranje transformacije čvora u JavaFX
  - (1) pomoću metoda za specificiranje elementarne transformacije čvora
    - transformacija čvora je njegovo svojstvo (*property*)
    - metode za transformacije upravljaju odgovarajućim svojstvom čvora
    - `setTranslateX()`, `setTranslateY()`, `setRotate()`, `setScaleX()`, `setScaleY()`
  - (2) pomoću objekata transformacija klasa `Translate`, `Rotate`, `Scale`, `Sheer` i `Affine`
    - klase su izvedene iz klase `Transform`, iz paketa `javafx.scene.transform`
    - klasa `Transform` je direktno izvedena iz klase `Object`
    - transformacija čvora je njegov atribut
    - pogodan način da se specificira sekvenca transformacija čvora

Načini transformacija u JavaFX (2)

- Postupak kada se primenjuje 2. način:
  - najpre se dohvati tekuća transformacija
    - metod čvora `getTransforms()`
    - vraća rezultat tipa `ObservableList<Transform>`
  - na dohvaćenu transformaciju se nadoveže nova sekvenca transform.
    - za vraćeni objekat se poziva `addAll()`
    - sa listom argumenata klasa potomaka `Transform`
  - transformacije u listi argumenata se primenjuju redosledom navođenja
- Ukoliko se primenjuju oba načina transformisanja čvora
  - prvo se vrši transformacija kojom se postavlja atribut čvora
  - zatim transformacija metodama koje postavljaju svojstva čvora

Translacija (1)

- Prvi način – metodima za postavljanje translacionih svojstava čvora:

```
setTranslateX(double tx)
setTranslateY(double ty)
```

  - `tx` i `ty` su pomeraji po *X* i *Y* osi koordinatnog sistema roditelja
  - to su koordinate koordinatnog početka lokalnog KS objekta u koordinatnom sistemu roditelja, posle translacije
- Drugi način – definisanjem atributa transformacije čvora
  - klasa kojom se opisuje atribut translacije je `Translate`
  - objekat klase `Translate` se može stvoriti konstruktorom klase:

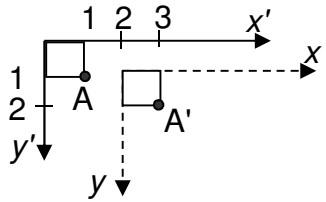
```
Translate(double tx, double ty)
```

  - ili statičkim metodom klase `Transform`:

```
static Translate translate(double tx, double ty)
```

Translacija (2)

- Transformacija:



- Matrična jednačina translacije:

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix} = \begin{vmatrix} x + tx \\ y + ty \\ 1 \end{vmatrix}$$

Primer translacije


```
Rectangle original = new Rectangle(5.0,5.0, 100.0,25.0);
original.setStroke(Color.GREEN); original.setFill(null);

Rectangle slika1 = new Rectangle(5.0,5.0, 100.0,25.0);
slika1.setStroke(Color.RED); slika1.setFill(null);
slika1.setTranslateX(90);
slika1.setTranslateY(30);

Rectangle slika2 = new Rectangle(5.0,5.0, 100.0,25.0);
slika2.setStroke(Color.BLUE); slika2.setFill(null);
Translate t = new Translate(25, 60);
slika2.getTransforms().addAll(t);
```

Pivot

- Za opis atributa transformacija rotacije, skaliranja i smicanja (za koje se koriste klase `Rotate`, `Scale` i `Sheer`, respektivno)
  - osim parametara lokalne transformacije (na primer, ugla za rotaciju)
  - mogu se definisati koordinate tačke pivota
- Pivot je tačka u odnosu na koju se transformiše lokalni KS objekta
- Matrična jednačina transformacije u odnosu na pivot (p_x,p_y)
  - globani KS (KS roditeljskog čvora) se translira u tačku pivota
  - uradi se zadata transformacija lokalnog koordinatnog sistema
  - globalni KS se vrati inverznom translacijom u originalni koord. početak
- Matrična jednačina ima sledeći oblik:
- $\mathbf{Q}' = (\mathbf{M_T}^{-1} * \mathbf{M_X} * \mathbf{M_T}) * \mathbf{Q} = \mathbf{M} * \mathbf{Q}$,
  - $\mathbf{Q}'$ je slika, a $\mathbf{Q}$ original tačke, vektori-kolone
  - $\mathbf{M_T}$ je matrica translacije globalnog KS u pivot, a $\mathbf{M_T}^{-1}$ inverzna matrica
  - $\mathbf{M_X}$ matrica transformacije lokalnog KS za pivot kao koord. početak

Rotacija (1)

- Prvi način – metod koji postavlja rotaciono svojstvo klase `Node`:

```
setRotate(double a)
```

- Objekat se rotira
  - za ugao od `a` stepeni u smeru kazaljke na časovniku
  - oko tačke (pivota) koja se nalazi u centru objekta
    - preciznije – u centru opisanog pravougaonika oko objekta
    - opisani pravougaonik – granice plana (*layout bounds*) objekta, pre rotacije
- Drugi način - definisanjem atributa transformacije posmatranog čvora
  - klasa kojom se opisuje atribut rotacije je `Rotate`
  - objekat klase `Rotate` se može stvoriti jednim od konstruktora klase:

```
Rotate(double a)
Rotate(double a, double px, double py)
```

    - gde je `a` ugao rotacije, a `px` i `py` koordinate pivota
  - ili statičkim metodima klase `Transform`:

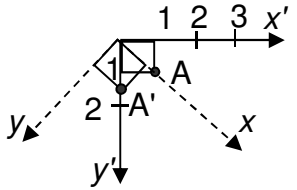
```
static Rotate rotate(double a)
static Rotate rotate(double a, double px, double py)
```

Rotacija (2)

- Matrična jednačina rotacije:

$$\mathbf{M_T} = \begin{vmatrix} 1 & 0 & -px \\ 0 & 1 & -py \\ 0 & 0 & 1 \end{vmatrix} \qquad \mathbf{M_R} = \begin{vmatrix} \cos(a) & -\sin(a) & 0 \\ \sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

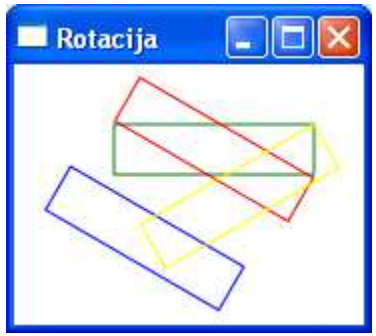
$$\mathbf{M_T^{-1}} = \begin{vmatrix} 1 & 0 & px \\ 0 & 1 & py \\ 0 & 0 & 1 \end{vmatrix}$$



$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} \cos(a) & -\sin(a) & px-px*\cos(a)+py*\sin(a) \\ \sin(a) & \cos(a) & py-px*\sin(a)-py*\cos(a) \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

$$= \begin{vmatrix} (x-px)*\cos(a) & -(y-py)*\sin(a) & + px \\ (x-px)*\sin(a) & (y-py)*\cos(a) & + py \\ 1 & & \end{vmatrix}$$

Primer rotacije




```
Rectangle original = new Rectangle(50.0,30.0, 100.0,25.0);
original.setStroke(Color.GREEN); original.setFill(null);

Rectangle slika1 = new Rectangle(50.0,30.0, 100.0,25.0);
slika1.setStroke(Color.RED); slika1.setFill(null);
slika1.setRotate(30);

Rectangle slika2 = new Rectangle(50.0,30.0, 100.0,25.0);
slika2.setStroke(Color.BLUE); slika2.setFill(null);
Rotate r1 = new Rotate(30);
slika2.getTransforms().addAll(r1);

Rectangle slika3 = new Rectangle(50.0,30.0, 100.0,25.0);
slika3.setStroke(Color.YELLOW); slika3.setFill(null);
Rotate r2 = new Rotate(-30,150,30);
slika3.getTransforms().addAll(r2);
```

Skaliranje (1)

- Prvi način - metodima koji postavljaju svojstva klase `Node`:

```
setScaleX(double sx)
setScaleY(double sy)
```

  - objekat se skalira koeficijentima (skala-faktorima) `sx` i `sy` u odnosu na pivot u centru opisanog pravougaonika oko objekta pre skaliranja
- Drugi način – definisanjem atributa transformacije čvora
  - klasa kojom se opisuje atribut skaliranja je `Scale`
  - objekat klase `Scale` se može stvoriti konstruktorima klase:

```
Scale(double sx, double sy)
Scale(double sx, double sx, double px, double py)
```

    - ili statičkim metodima klase `Transform`:

```
static Scale scale(double sx,double sy)
static Scale scale(double sx, double sx,
                   double px, double py)
```

    - gde su `sx` i `sy` koeficijenti skaliranja duž *X* i *Y* osa, a (`px`,`py`) – pivot

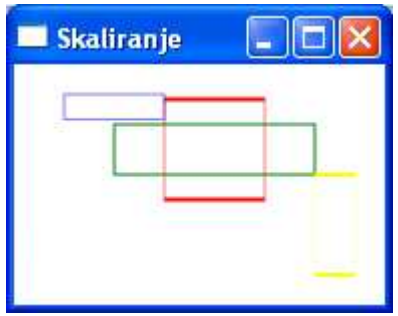
Skaliranje (2)

- Matrična jednačina skaliranja:

$$\mathbf{M}_T = \begin{vmatrix} 1 & 0 & -px \\ 0 & 1 & -py \\ 0 & 0 & 1 \end{vmatrix} \quad \mathbf{M}_S = \begin{vmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad \mathbf{M}_T^{-1} = \begin{vmatrix} 1 & 0 & px \\ 0 & 1 & py \\ 0 & 0 & 1 \end{vmatrix}$$

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} sx & 0 & px(1-sx) \\ 0 & sy & py(1-sy) \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix} = \begin{vmatrix} sx*x + px(1-sx) \\ sy*y + py(1-sy) \\ 1 \end{vmatrix}$$

Primer skaliranja




```
Rectangle original = new Rectangle(50.0,30.0, 100.0,25.0);
original.setStroke(Color.GREEN); original.setFill(null);

Rectangle slika1 = new Rectangle(50.0,30.0, 100.0,25.0);
slika1.setStroke(Color.RED); slika1.setFill(null);
slika1.setScaleX(0.5); slika1.setScaleY(2.0);

Rectangle slika2 = new Rectangle(50.0,30.0, 100.0,25.0);
slika2.setStroke(Color.BLUE); slika2.setFill(null);
Scale s1 = new Scale(0.5,0.5);
slika2.getTransforms().addAll(s1);

Rectangle slika3 = new Rectangle(50.0,30.0, 100.0,25.0);
slika3.setStroke(Color.YELLOW); slika3.setFill(null);
Scale s2 = new Scale(-0.2,-2.0,150,55);
slika3.getTransforms().addAll(s2);
```

Smicanje (iskošenje) (1)

- Smicanje čvora se može postići samo definisanjem atributa čvora
  - za razliku od translacije, rotacije i skaliranja
- Klasa kojom se opisuje atribut smicanja je `Shear`
- Objekat klase `Sheer` se može stvoriti konstruktorima klase:

```
Shear(double hx, double hy)
Shear(double hx, double hy, double px, double py)
```

  - ili statičkim metodima klase `Transform`:

```
static Shear shear(double hx, double hy)
static Shear shear(double hx, double hx,
                   double px,double py)
```

  - gde su `hx` i `hy` faktori smicanja duž *X* i *Y* osa, a (`xp`,`yp`) - pivot

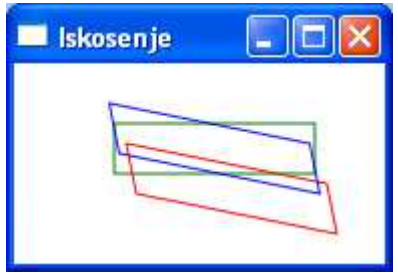
Smicanje (iskošenje) (2)

- Matrična jednačina smicanja ima sledeći oblik:

$$\mathbf{M}_T=\begin{vmatrix}1 & 0 & -px\\ 0 & 1 & -py\\ 0 & 0 & 1\end{vmatrix} \quad \mathbf{M}_H=\begin{vmatrix}1 & hx & 0\\ hy & 1 & 0\\ 0 & 0 & 1\end{vmatrix} \quad \mathbf{M}_T^{-1}=\begin{vmatrix}1 & 0 & px\\ 0 & 1 & py\\ 0 & 0 & 1\end{vmatrix}$$

$$\begin{vmatrix}x'\\ y'\\ 1\end{vmatrix} = \begin{vmatrix}1 & hx & -py*hx\\ hy & 1 & -px*hy\\ 0 & 0 & 1\end{vmatrix} * \begin{vmatrix}x\\ y\\ 1\end{vmatrix} = \begin{vmatrix}x + hx*y - py*hx\\ hy*x + y - px*hy\\ 1\end{vmatrix}$$

Primer smicanja




```
Rectangle original = new Rectangle(50.0,30.0, 100.0,25.0);
original.setStroke(Color.GREEN);
original.setFill(null);

Rectangle slika1 = new Rectangle(50.0,30.0, 100.0,25.0);
slika1.setStroke(Color.RED); slika1.setFill(null);
Shear s1 = new Shear(0.2,0.2);
slika1.getTransforms().addAll(s1);

Rectangle slika2 = new Rectangle(50.0,30.0, 100.0,25.0);
slika2.setStroke(Color.BLUE); slika2.setFill(null);
Shear s2 = Transform.shear(0.2,0.2,100.0,42.5);
slika2.getTransforms().addAll(s2);
```

Afina transformacija - razlog

- Složenu transformaciju čini sekvenca elementarnih
- Može se zadati tako što se:
  - metodom željenog čvora scene `getTransforms()` dohvati tekuća sekvenca elementarnih transformacija pridruženih čvoru
  - zatim se metodom `addAll(t1, t2, t3,...)` dodaju tekućoj sekvenci transformacije `t1, t2, t3,....`
- Loša strana ovog postupka
  - atribut transformacije se svodi na listu elementarnih transformacija
  - njih je potrebno svaki put množiti da bi se dobila kompozitna matrica
  - nije sigurno da postoji i "keširana" kompozitna matrica
- Postoji način da se kompozitna matrica jedanput izračuna i zada kao matrica složene transformacije
  - tada se u vreme iscrtavanja ovom matricom množe samo karakteristične tačke čvorova i tako dobijaju njihove slike
  - može da utiče na efikasnost crtanja

Matrica afine transformacije

- Za slučaj 2D transformacija objekta:

```
| x'|    | mxx  mxy  tx |    |x|    |mxx*x + mxy*y + tx|
| y'| = | myx  myy  ty | * |y| = |myx*x + myy*y + ty|
| 1 |    |  0    0    1 |    |1|    |          1          |
```

- Klasa kojom se opisuje atribut složene transformacije je `Affine`
- Objekat klase se stvara konstruktorom `Affine()`
  - objekat transformacije predstavlja matricu identiteta
  - metodima oblika `setMxx(double mxx)`, ..., `setTy(double ty)` se zadaju odgovarajući elementi matrice afine transformacije
  - metodom `void append(Transform am)` klase `Affine` se superponiraju transformacije `Translate`, `Rotate`, `Scale`, `Shear`
- Drugi način stvaranja – statičkim metodom klase `Transform`:

```
static Affine affine(double mxx, double myx, double mxy,
                     double myy, double tx,  double ty)
```

Računanje matrice transformacije

- Atribut afine transformacije predstavlja preslikavanje
  - iz transformisanog lokalnog KS objekta u KS roditeljskog čvora
  - matrica transformacije se množi vektorima-kolonama tačaka da bi se one iz lokalnog KS transformisale u KS roditelja
- Transformisanje objekta:
  - pre transformacije lokalni KS se poklapa sa KS roditeljskog čvora
  - na objekat se primenjuje sekvenca transformacija: *t1*, *t2*,..., *tn*
- Pri preslikavanju iz lokalnog KS u KS roditeljskog čvora
  - transformacije se vrše obrnutim redosledom: *tn*,..., *t2*, *t1*
- Kada se tačka predstavlja vektorom-kolonom
  - vrši se prekonkatenacija matrica elementarnih transformacija
- Kompozitna matrica afine transformacije: $\mathbf{A}=\mathbf{T}_1*\mathbf{T}_2*...*\mathbf{T}_n$
  - redosled matrica u proizovdu odgovara redosledu transformacija kojima se transformiše objekat

Primer afine transformacije (1)

```
Rectangle original = new Rectangle(5.0,5.0, 100.0,25.0);
Circle tačka1 = new Circle(105.0, 30.0, 2.0);
original.setStroke(Color.GREEN); original.setFill(null);

Rectangle slika1 = new Rectangle(5.0,5.0, 100.0,25.0);
slika1.setStroke(Color.RED); slika1.setFill(null);
slika1.setStrokeWidth(8);
Translate t = new Translate(40,30);
Rotate r = new Rotate(45);
Affine a1 = new Affine();
a1.append(t); a1.append(r);
System.out.println(a1.getMxx()+" "+a1.getMxy()+" "+a1.getTx());
System.out.println(a1.getMyx()+" "+a1.getMyy()+" "+a1.getTy());
slika1.getTransforms().addAll(a1);
```

Primer afine transformacije (2)

```
Rectangle slika2 = new Rectangle(5.0,5.0, 100.0,25.0);
slika2.setStroke(Color.BLUE); slika2.setFill(null);
slika2.setStrokeWidth(2);
double k=Math.sqrt(2)/2;
Affine a2=Transform.affine(k, k, -k, k, 40, 30);
slika2.getTransforms().addAll(a2);
Circle tačka2 = new Circle(92.875, 125.175, 2.0);
tačka2.setFill(Color.YELLOW);
```

Matrica a1:

```
0.7071067811865476 -0.7071067811865475 40.0
0.7071067811865475 0.7071067811865476 30.0
```

Translacija u lokalnom sistemu

- Metodi koji postavljaju svojstva plana (*layout*) čvora:
  `setLayoutX(double x)` i `setLayoutY(double y)`
- Metodi koji postavljaju translaciona svojstva čvora:
  `setTranslateX(double x)` i `setTranslateY(double y)`
- Prvi par metoda translira objekat u njegovom lokalnom KS
  - loklani KS objekta je vezan za KS roditeljskog čvora
- Drugi par metoda translira objekat u KS roditeljskog čvora
  - lokalni KS objekta vezan za objekat
- Najpre se primenjuje transformacija čvora u roditeljskom KS, a zatim translaciono pomeranje definisano planom čvora
- Korišćenje
  - svojstva plana se koriste za pozicioniranje čvora u roditeljskom KS
  - transformacija se tipično koristi za pomeranje čvora u animaciji