


Računarska grafika

JavaFX – 2D oblici



Geometrijske 2D primitive

- Osnovna klasa – apstrakcija geom. oblika: `Shape`
- Linije:
 - prava linija (segment, duž), izlomljena linija
- Mnogouglovi:
 - pravougaonik, proizvoljan mnogougao
- Ovali:
 - krug, elipsa, luk
- Krive:
 - kvadratna i kubna Bezejeova
- Napomena:
 - sve primitive osim linija podrazumevano su popunjene

Složeni oblici

- Putanja
 - proizvoljna linijska otvorena ili zatvorena putanja
- SVG putanja
 - putanja definisana standardom SVG
- Kombinovani oblici – skupovne operacije
 - unija
 - presek
 - razlika

Linija

- Linija = pravolinijski segment, duž
- Zadavanje:
 - argumenti konstruktora – koordinate početne i krajnje tačke

```
Line(double pX, double pY, double kX, double kY);
```

- podrazumevani konstruktor, naknadno svojstava linije

```
Line linija = new Line();
```

```
linija.setStartX(pX);
```

```
linija.setStartY(pY);
```

```
linija.setEndX(kX);
```

```
linija.setEndY(kY);
```

Izlomljena linija

- Zadavanje:

- niz parametara promenljive dužine, smenjuju se X i Y koordinate temana

```
Polyline(double... koordinate)
```

- dodavanje niza X, Y koordinata na listu prethodnih

```
Polyline polyline = new Polyline();  
polyline.getPoints().addAll(new Double[] {  
    x1,y1, x2,y2, x3,y3 });
```

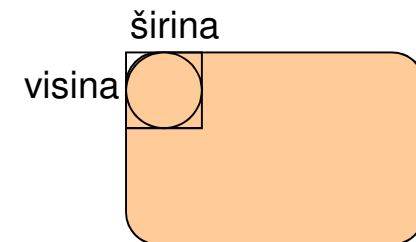
Pravougaonik

- Stranice paralelne osama
- Zadavanje:
 - konstruktor sa zadatim gornjim levim uglom, širinom i visinom
`Rectangle(double x, double y, double š, double v)`
 - podrazumevani konstruktor, naknadno postavljanje svojstava
`Rectangle pravougaonik = new Rectangle();`
`pravougaonik.setX(x);`
`pravougaonik.setY(y);`
`pravougaonik.setWidth(š);`
`pravougaonik.setHeight(v);`

Zaobljeni pravougaonik

- Definiše se pravougaonik, a zatim zaobljenje
- Zaobljenje se definiše pravougaonikom u uglu, opisanim oko elipse koja određuje krivinu
- Luk elipse od 90° se crta umesto ugla pravougaonika
- Metodi kojima se određuje širina i visina elipse:

```
void setArcWidth(double širina)  
void setArcHeight(double visina)
```



Mnogougao

- Zatvorena izlomljena linija
- Klasa `Polygon`
- Zadaje se na iste načine kao i izlomljena linija
- Poslednje teme se automatski spaja sa prvim
- Može biti proizvoljan
 - konveksan
 - konkavan
 - samopresecajući

Krug

- Zadavanje:

- argumenti konstruktora – koordinate centra i poluprečnik

```
Circle(double cX, double cY, double r);
```

- argumenti konstruktora – samo poluprečnik, naknadno centar

```
Circle krug = new Circle(r);
```

```
krug.setCenterX(x);
```

```
krug.setCenterY(y);
```

- podrazumevani konstruktor – naknadno centar i poluprečnik

```
krug.setRadius(r);
```

Elipsa

- Zadavanje:

- argumenti konstruktora – centar, poluprečnici X i Y

```
Ellipse(double cX, double cY, double rX, double rY);
```

- argumenti konstruktora – poluprečnici, naknadno centar

```
Ellipse elipsa = new Ellipse(double rX, double rY);
```

```
elipsa.setCenterX(x);
```

```
elipsa.setCenterY(y);
```

- podrazumevani konstruktor – naknadno centar i poluprečnici

```
elipsa.setRadiusX(r);
```

```
elipsa.setRadiusY(r);
```

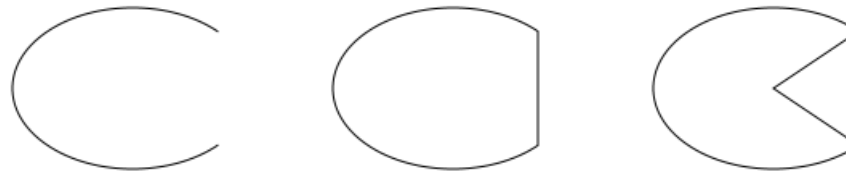
Luk

- Deo elipse između dve tačke na njoj
- Zadavanje:
 - argumenti konstruktora – elipsa + početni ugao i ugao luka
 - referentni smer: suprotno od kazaljke, mereno od +X ose
 - podrazumevani konstruktor, naknadno parametri elipse + uglovi

```
Arc luk = new Arc();  
// ... kao za elipsu  
luk.setStartAngle(pU);  
luk.setLength(kU);
```

Tipovi luka

- Tipovi:
 - otvoren (OPEN) - podrazumevano,
 - odsečak (CHORD),
 - isečak (ROUND)

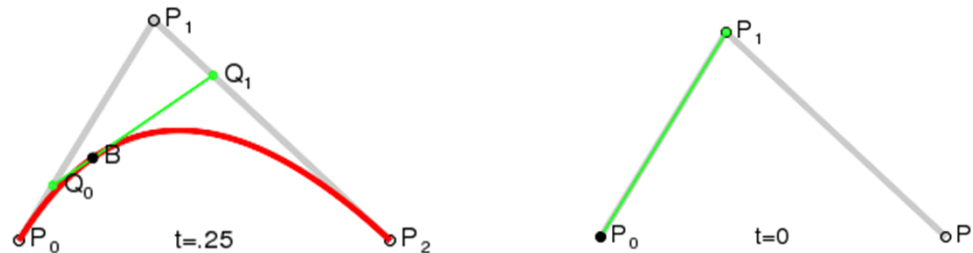


- Zadavanje:

```
luk.setType(ArcType.TIP_LUKA);
```

Kvadratna kriva (1)

- Kvadratna Bezejeva (orig. *Bézier*) kriva
 - određena sa 3 tačke:
 - 2 krajnje i jednom kontrolnom “privlačećom” tačkom
 - prave koje spajaju početnu/krajnju tačku sa kontrolnom su tangente na krivu u početnoj/krajnjoj tački
 - kriva je linearna interpolacija između dve tačke
 - prva je linearna interpolacija između početne i kontrolne tačke
 - druga je linearna interpolacija između kontrolne i krajnje tačke



JavaFX - 2D oblici

24.02.2017.

Kvadratna kriva (2)

- Zadavanje:

- argumenti konstruktora – koord. početne, kontrolne i krajnje tačke

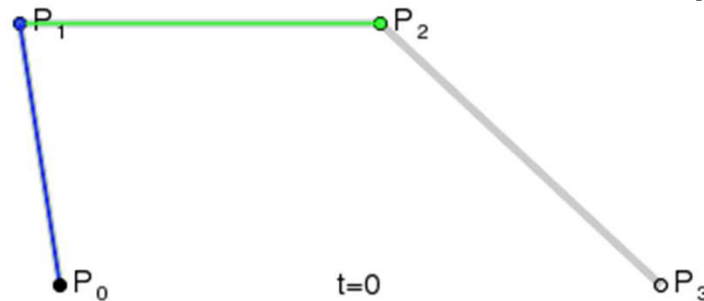
```
QuadCurve(double x1, double y1,  
           double xk, double yk,  
           double x2, double y2);
```

- podrazumevani konstruktor, naknadno zadavanje tačaka

```
QuadCurve kriva = new QuadCurve();  
kriva.setStartX(x1);           kriva.setStartY(y1);  
kriva.setControlX(xk);        kriva.setControlY(yk);  
kriva.setEndX(x2);            kriva.setEndY(y2);
```

Kubna kriva

- Kubna Bezejeova kriva – 2 kontrolne privlačeće tačke



- Zadavanje:
 - analogna dva načina kao za kvadratnu krivu
 - metodi za postavljanje kontrolnih tačaka
 - `kriva.setControlX1(xK1);`
 - `kriva.setControlY1(yK1);`
 - `kriva.setControlX2(xK2);`
 - `kriva.setControlY2(yK2);`

Putanja (1)

- Složen linijski oblik koji se sastoji od sekvence
 - elementarnih geometrijskih otvorenih primitiva:
 - linijski segment (proizvoljan, horizontalan, vertikalni), luk, kriva
 - pomeraja u tačku (bez crtanja)
 - zatvaranja putanje pravom linijom
- Koncept sastavljanja putanje
 - izvrši se pomeraju njenu početnu tačku (postaje tekuća)
 - zatim se elementi dodaju, tako što se odgovarajuća geometrijska primitiva crta od tekuće do zadate tačke



Putanja (2)

- Zadavanje:

- konstruktor sa proizvoljnim brojem elemenata putanje

```
Path(PathElement... elementi)
```

- podrazumevani konstruktor i naknadno dodavanje elemenata

```
Path putanja = new Path();
```

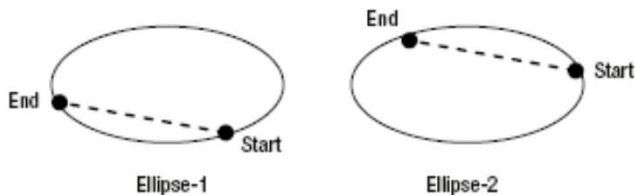
```
putanja.getElements().addAll(pomeraj, linija, luk,  
                               kriva, ..., zatvaranje);
```

Elementi putanje

- `MoveTo` (pomeranje u tačku)
- `LineTo` (segment sa proizvoljnim nagibom)
- `HLineTo` (horizontalni segment)
- `VLineTo` (vertikalni segment)
- `ArcTo` (luk)
- `QuadCurveTo` (kvadratna Bezejeova kriva)
- `CubicCurveTo` (kubna Bezejeova kriva)
- `ClosePath` (zatvaranje putanje)

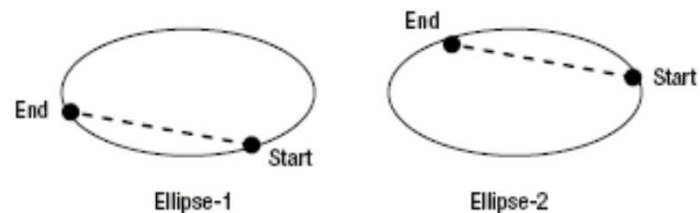
Luk – element putanje (1)

- Problem:
 - kroz tekuću i zadatu tačku moguće je provući 4 luka elipse



- svaka od dve elipse sa slike ima po dva moguća luka
- Rešenje – dva parametra koja će odrediti:
 - da li se uzima kraći ili duži luk
 - da li je smer luka od tekuće do zadate tačke
 - smer kazaljke na časovniku (eng. *clockwise*) – `true`
 - obrnut (eng. *counterclockwise*) – `false`

Luk – element putanje (2)



Tip luka	Elipsa	veliki	smer
veliki	elipsa1	true	false
veliki	elipsa2	true	true
mali	elipsa1	false	true
mali	elipsa2	false	false

- Moguća i rotacija X-ose ellipse luka

SVG putanja

- Standardna notacija SVG (*Scalable Vector Graphics*)
 - standard propisuje format XML fajla za vektorski i mešani vektorsko-rasterski opis 2D grafičkih oblika
 - održava konzorcijum W3C, dokumentacija: <http://www.w3.org/TR/SVG/>
- SVG putanja se definiše kao niska znakova
 - specificira geometrijske primitive koje ulaze u putanju
- Klasa `SVGPath` je izvedena iz klase `Shape`
- Zadavanje:
 - podrazumevanim konstruktorom `SVGPath()`, zatim se metodom `setContent()` postavlja SVG niska znakova

SVG komande i parametri

Kom	Parametri	Opis	JavaFX klasa
M	$(x, y) +$	pomeranje u tačku (x,y)	MoveTo
L	$(x, y) +$	linija do tačke (x,y)	LineTo
H	$x +$	horizontalna linija do x	HLineTo
V	$y +$	vertikalna linija do y	VLineTo
A	$(rx, ry, rotacija, veliki, smer, x, y) +$	luk elipse	ArcTo
Q	$(x1, y1, x, y) +$	kvadratna Bezjeova kriva	QuadCurveTo
T	$(x, y) +$	glatko nadovezana kv. kriva	
C	$(x1, y1, x2, y2, x, y) +$	kubna Bezjeova kriva	CubicCurveTo
S	$(x2, y2, x, y) +$	glatko nadovezana ku. kriva	
Z	-	zatvaranje putnje	ClosePath

Kombinovani oblik

- Kombinacije dva geometrijska oblika
- Skupovne operacije: unija, presek i razlika
- Operacije se primenjuju na skupove piksela koji pripadaju zadatim oblicima
- Oblici se kombinuju statičkim metodima klase `Shape`
 - unija: `Shape.union(oblik1, oblik2)`
 - presek: `Shape.intersect(oblik1, oblik2)`
 - razlika: `Shape.subtract(oblik1, oblik2) //oblik1-oblik2`

Kombinovanje oblika - primer

```
Circle krug1 = new Circle(25.0, 25.0, 20.0);  
Circle krug2 = new Circle(35.0, 25.0, 20.0);  
Circle krug3 = new Circle(80.0, 25.0, 20.0);  
Circle krug4 = new Circle(95.0, 25.0, 20.0);  
Circle krug5 = new Circle(135.0, 25.0, 20.0);  
Circle krug6 = new Circle(145.0, 25.0, 20.0);  
Shape unija = Shape.union(krug1, krug2);  
Shape presek = Shape.intersect(krug3, krug4);  
Shape razlika = Shape.subtract(krug5, krug6);
```

