Računarska grafika

JavaFX - animacija

Uvod

- Animacija - pokretna crtana slika
  - crtana – ručno ili automatski (računarskim programom)
- Pokretna slika je iluzija – stvara se brzimređanjem sličnih slika
  - čovekov vid ima ograničenje – razlikuje 24 slike u sekundi
  - posmatranjem promenljive scene u realnom svetu, čovek praktično uzorkuje 24 slike scene u sekundi
  - isti efekat – ako se na prikaznom uređaju prikažu 24 slike u sekundi
  - ako se sukcesivno prikazane slike relativno malo razlikuju, čovekov vid vrši interpolaciju između njih
    - interpolacija čini pokrete objekata u sceni kontinualnim
  - fenomen koristi najpre konvencionalni film, pa zatim TV slika
    - generišu pokretnu sliku prikazom serije relativno sličnih slika u kadru
  - U animiranom (crtanom) filmu se koristi serija crteža
    - sintetički (ručno ili automatski) stvorenih relativno sličnih slika

Animacija u JavaFX

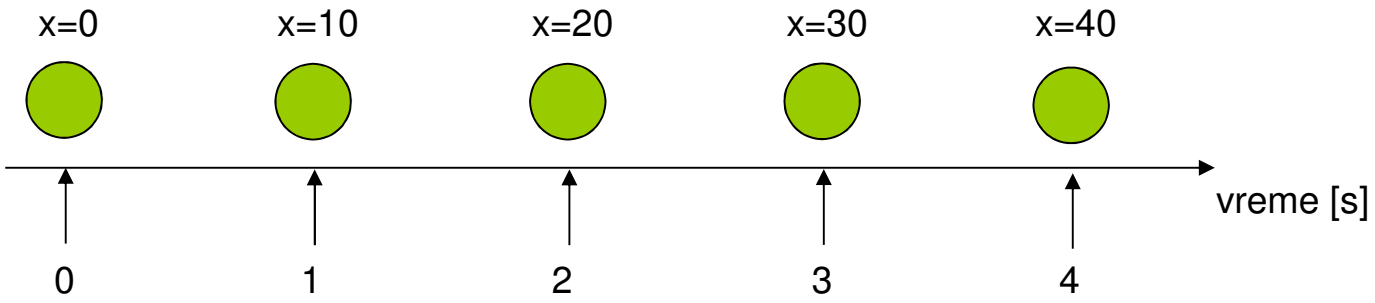
- Animacija u JavaFX:
  - prikaz promene vizuelnih svojstava/atributa čvora u vremenu
- Promena svojstava čvora:
  - geometrijska transformacija
    - efekat kretanja, promene veličine i/ili oblika čvora
  - promena boje i bilo kog drugog atributa koji određuje izgled čvora
- Kompletna promena u sceni
  - promenom svojstava/atributa željenih čvorova grafa scene
  - za svaki čvor se može definisati njegova animacija
- Napredovanje animacije u periodu trajanja
  - apstrahovano klasom `Timeline`

Ključne i interpolirane slike

- Neke slike u toku animacije su ključne slike (*key frames*)
  - eksplicitno se definišu
  - reprezentuju stanje animiranog čvora u nekom trenutku
  - apstrahovane su klasom `KeyFrame`
- U ključnim slikama atributi imaju zadate ključne vrednosti
  - vrednosti atributa su apstrahovane klasom `KeyValue`
- Slike između ključnih – automatskom sintezom
- Automatska sinteza
  - atributi se interpoliraju po nekom zakonu
  - `Interpolator` apstrahuje zakonitost po kojem se vrši interpolacija
  - interpolacija se vrši između tekućeg i sledećeg stanja ključne vrednosti

Animacija u JavaFX – sumarno

- Potrebno je:
  - da objekat vremenske linije (klase `Timeline`), sadrži sekvencu objekata ključnih slika (klase `KeyFrame`)
  - da svaki objekat ključne slike sadrži zbirku objekata ključnih vrednosti (klase `KeyValue`)
  - da svaki objekat ključne vrednosti sadrži objekat interpolatora (objekat klase `Interpolator`)
- Objektom vremenske linije se definiše animacija nekog čvora



Interpolacija

- Na ključne vrednosti primenjuju se zakoni interpolacije određeni pridruženim objektima interpolatora
- Podrazumevano, primenjuju se linearni interpolatori
- Definišu tekuću vrednost odgovarajuće ključne vrednosti proporcionalno između dve sukcesivne vrednosti:

$$(V_x\text{-}Kv_i):(Kv_{i+1}\text{-}Kv_i)=(t_x\text{-}t_i):(t_{i+1}\text{-}t_i)$$

  - V_x – tekuća vrednost u trenutku t_x
  - Kv_i i Kv_{i+1} – sukcesivne ključne vrednosti između kojih se određuje V_x
  - t_i i t_{i+1} – trenuci kada atribut čvora ima ključne vrednosti Kv_i i Kv_{i+1}
  - t_x – proizvoljan trenutak između trenutaka t_i i t_{i+1}

Vrste animacije

- JavaFX podržava dve vrste animacije:
- (1) eksplicitno definisanje vremenske linije
  - definišu se pripadajući objekati vremenske linije
    - ključne slike, ključne vrednosti, interpolatori
- (2) "prelazi" – implicitno definišu vremensku liniju
  - definišu se samo početna i krajnja vrednost svojstva i trajanje
- Definisanje vremenske linije i pripadajućih objekata
  - složenije za korišćenje
  - preciznije upravljanje animacijom
- Definisanje prelaza
  - jednostavnije za korišćenje
  - manje precizno upravljanje animacijom

Animacija prelazima

- Definišu se:
  - početna i krajnja vrednost odgovarajućeg svojstva (atributa čvora)
  - trajanje animacije
- Sistem obezbeđuje stvaranje odgovarajuće vremenske linije
  - programer je rasterećen brige oko definisanja vremenske linije
  - viši nivo apstrakcije
  - manja fleksibilnost
- Prelazi se koriste za često korišćene animacije
  - animacije kretanja, promene boje ili transparentnosti
- Prelazi mogu biti sekvencijalno ili paralelno primenjeni

Hijerarhija klasa

- Animacija se apstrahuje klasom `Animation`
  - iz paketa `javafx.animation`
- Iz ove klase su izvedene klase
  - `Timeline` – apstrahuje animaciju definisanjem vremenske linije
  - `Transition` – apstrahuje animaciju definisanjem prelaza
- Konkretne klase za animaciju prelazima (izvedene iz klase `Transition`)
  - za animaciju kretanja: `TranslateTransition`, `RotateTransition` i `PathTransition`
  - za animaciju promene veličine (i oblika): `ScaleTransition`
  - za animaciju promene boje popunjavanja: `FillTransition`
  - za animaciju promene boje linije: `StrokeTransition`
  - za animaciju promene prozirnosti: `FadeTransition`
  - za kombinovanje prelaza sekvencijalno: `SequentialTransition`
  - za kombinovanje prelaza paralelno: `ParallelTransition`
  - za pauziranje (sekvencijalne) animacije određen period: `PauseTransition`

Pokretanje i izvršavanje animacije

- Formira se objekat animacije
  - animacije vremenskom linijom ili prelazom
- Animacija se pokreće pozivom metoda objekta animacije `play()`
  - metod samo pokreće animaciju
  - odmah vraća kontrolu pozivajućem programu
  - to omogućava i da se metod pozove ponovo za narednu animaciju
- Ako se metod `play()` pozove više puta u sekvenci
  - odgovarajuće animacije će se izvršavati u paraleli
  - važi bilo da se poziv odnosi na različite ili isti čvor koji se animira

Definisanje prelaza

- Prelazi su apstrahovani klasom `Transition`
  - klasa je izvedena iz klase `Animation`
- Potklase apstrahuju promene pojedinih svojstava čvora
- Animacija prelazom omogućava da se definišu samo
  - trajanje prelaza
  - čvor na koji se primenjuje prelaz
  - početna i krajnja vrednost svojstva koje se menja
    - ako se ne definiše početna vrednost – uzima se tekuća vrednost svojstva
- Kreiranje vremenske linije i odgovarajućih objekata – automatski
- Objekat prelaza sadrži objekat interpolatora
- Podrazumevani interpolator je `Interpolator.EASE_BOTH`
  - počinje animaciju sporo, ubrzava je, a kasnije usporava prema kraju

Prelaz prozirnosti

- Postiže se efekat postepenog nestajanja/pojavljivanja čvora
- U toku zadatog trajanja animacije postepeno se smanjuje ili povećava vrednost svojstva neprozirnosti (*opacity*) čvora
- Opisuje se klasom `FadeTransition` koja ima sledeća svojstva:
  - trajanje (klase `Duration`) – određuje vreme ciklusa animacije
  - čvor (klase `Node`) – određuje objekat čije se svojstvo menja
  - početna vrednost (`fromValue`) svojstva neprozirnosti
  - poslednja vrednost (`toValue`) svojstva neprozirnosti
    - određuje apsolutnu krajnju vrednost
  - promena vrednosti (`byValue`) svojstva neprozirnosti
    - određuje krajnju vrednost atributa relativno u odnosu na početnu vrednost
    - može biti pozitivna ili negativna
    - alternativa poslednjoj vrednosti (poslednja vrednost ima prioritet)
  - podrazumevana početna vrednost je tekuća vrednost neprozirnosti

Prelaz prozirnosti – zadavanje

- Objekat prelaza prozirnosti klase `FadeTransition` stvara se konstruktorom sa argumentima
  - vremenom `t` klase `Duration` i
  - objektom `čvor` klase `Node` na koji će se primeniti
- Zatim se postave početna i krajnja vrednost za neprozirnost
  - metodi `setFromValue()` i `setToValue()` / `setByValue()`
  - u opsegu `0.0` (potpuna prozirnost) do `1.0` (potpuna neprozirnost)
  - ako se vrednosti zadaju izvan opsega biće ograničene na opseg [0,1]
- Animacija se pokreće metodom `play()`

```
FadeTransition pt = new FadeTransition(t, čvor);
pt.setFromValue(pocetnaVrednost);
pt.setToValue(krajnjaVrednost);
// alternatvno: pt.setByValue(promenaVrednosti);
pt.play();
```

Primer prelaza prozirnosti

```
Circle krug1 = new Circle(75.0, 25.0, 20.0);
krug1.setFill(Color.RED);

Duration t = Duration.seconds(5);
FadeTransition nestanak = new FadeTransition(t, krug1);
nestanak.setFromValue(1.0);
nestanak.setToValue(0.1);

Circle krug2 = new Circle(175.0, 25.0, 20.0);
krug2.setFill(Color.BLUE);
FadeTransition pojava = new FadeTransition(t, krug2);
pojava.setFromValue(0.1);
pojava.setByValue(0.9);

nestanak.play();
pojava.play();
```

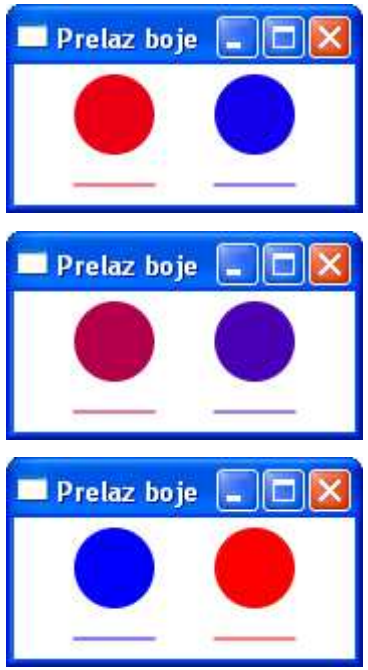
Prelaz boje

- Vrste prelaza boje:
  - prelaz boje popunjavanja – opisuje se klasom `FillTransition`
  - prelaz boje linije – opisuje se klasom `StrokeTransition`
- Menja se vrednost boje popunjavanja/linije od početne do krajnje
- Obe klase – svojstva kao i klasa `FadeTransition`

```
FillTransition pb= new FillTransition(t, čvor);
// ili:
// StrokeTransition pb= new StrokeTransition(t, čvor);
pb.setFromValue(pocetnaBoja);
pb.setToValue(krajnjaBoja);
pb.play();
```

Primer prelaza boje

```
Circle krug1 = new Circle(50.0, 25.0, 20.0);
Circle krug2 = new Circle(120.0, 25.0, 20.0);
Line linija1 = new Line(30,60, 70,60);
Line linija2 = new Line(100,60, 140,60);

Duration t = Duration.seconds(5);
FillTransition pbk1 = new FillTransition(t, krug1);
pbk1.setFromValue(Color.RED); pbk1.setToValue(Color.BLUE);
StrokeTransition pbl1 = new StrokeTransition(t, linija1);
pbl1.setFromValue(Color.RED); pbl1.setToValue(Color.BLUE);

FillTransition pbk2 = new FillTransition(t, krug2);
pbk2.setFromValue(Color.BLUE); pbk2.setToValue(Color.RED);
StrokeTransition pbl2 = new StrokeTransition(t, linija2);
pbl2.setFromValue(Color.BLUE); pbl2.setToValue(Color.RED);

pbk1.play(); pbl1.play();
pbk2.play(); pbl2.play();
```



Prelazi transformacijama

- Prelazom translacije se menja položaj objekta
  - opisuje se klasom `TranslateTransition`
- Prelazom rotacije se menja orijentacija objekta
  - opisuje se klasom `RotateTransition`
- Prelazom skaliranja se menja veličina, ali i oblik objekta, ako koeficijenti skaliranja nisu jednaki
  - opisuje se klasom `ScaleTransition`
- Ne postoji prelaz smicanja
- Prelazima se pri stvaranju zadaju: trajanje animacije `t` (tipa `Duration`) i `čvor` (tipa `Node`)
- Zatim se postavljaju karakteristike prelaza
  - početne i završne vrednosti odgovarajućih svojstava

Prelaz translacijom

- Početna vrednost koordinata čvora se postavljaju metodima:
  - `setFromX()` i `setFromY()`
- Završna vrednost se može postavljati na dva načina:
  - zadavanjem apsolutne vrednosti metodima: `setToX()` i `setToY()`
  - zadavanjem relativne vrednosti u odnosu na početnu metodima: `setByX()` i `setByY()`
- Prelaz se obavlja tako što se koordinatni početak lokalnog KS čvora translira za odgovarajući (apsolutno ili relativno) zadati pomeraj
- Efekat je odgovarajuće translatorno pomeranje čvora u KS scene

Prelazi rotacijom i skaliranjem

- Prelaz rotacijom
  - početna vrednost ugla se postavlja metodom: `setFromAngle()`
  - završna vrednost se može postavljati na dva načina:
    - zadavanjem apsolutne vrednosti metodom `setToAngle()`
    - zadavanjem relativne vrednosti u odnosu na početnu `setByAngle()`
  - pozitivan ugao rotacije se zadaje u smeru kazaljke sata
  - pivot tačka oko koje se rotira čvor se nalazi u njegovom centru

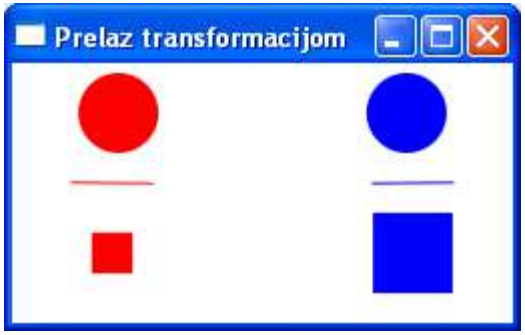
- Prelaz skaliranjem
  - početne vrednost faktora skaliranja se postavljaju metodima: `setFromX()` i `setFromY()`
  - završna vrednost se može postavljati zadavanjem:
    - apsolutne vrednosti metodima: `setToX()` i `setToY()`/`setByY()`
    - pivot tačka se nalazi u centru čvora

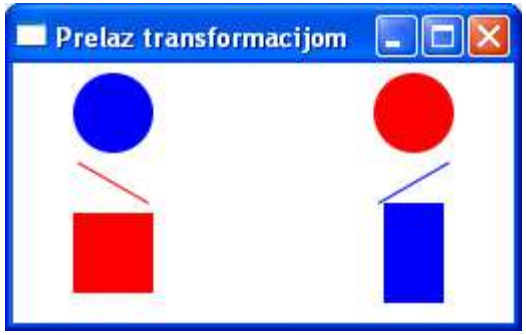
Primer prelaza transformacijom (1)

```
Circle krug1 = new Circle(50.0, 25.0, 20.0);
krug1.setFill(Color.RED);
Circle krug2 = new Circle(200.0, 25.0, 20.0);
krug2.setFill(Color.BLUE);
Line linija1 = new Line(30,60, 70,60);
linija1.setStroke(Color.RED);
Line linija2 = new Line(180,60, 220,60);
linija2.setStroke(Color.BLUE);
Rectangle pravoug1 = new Rectangle(30.0, 75.0, 40.0, 40.0);
pravoug1.setFill(Color.RED);
Rectangle pravoug2 = new Rectangle(180.0, 75.0, 40.0, 40.0);
pravoug2.setFill(Color.BLUE);

Duration t = Duration.seconds(5);
TranslateTransition ptk1 = new TranslateTransition(t, krug1);
ptk1.setFromX(0.0); ptk1.setToX(150.0);
TranslateTransition ptk2 = new TranslateTransition(t, krug2);
ptk2.setByX(-150.0);
```

Primer prelaza transformacijom (2)

```
RotateTransition prl1 = new RotateTransition(t, linija1);
prl1.setFromAngle(0); prl1.setToAngle(30);
RotateTransition prl2 = new RotateTransition(t, linija2);
prl2.setByAngle(-30);

ScaleTransition psp1 = new ScaleTransition(t, pravoug1);
psp1.setFromX(0.5); psp1.setFromY(0.5);
psp1.setToX(1.0); psp1.setToY(1.0);
ScaleTransition psp2 = new ScaleTransition(t, pravoug2);
psp2.setByX(-0.25); psp2.setByY(0.25);

ptk1.play(); ptk2.play();
prl1.play(); prl2.play();
psp1.play(); psp2.play();
```





Prelaz po putanji

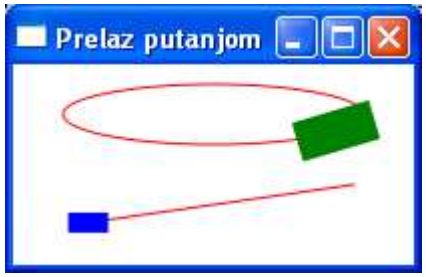
- Vrši se pomeranje čvora po putanji
- Putanja je definisana geometrijskim oblikom
  - centar animiranog čvora se nalazi na putanji
  - moguća promena orijentacije čvora tokom kretanja po putanji
- Prelaz po putanji se opisuje klasom `PathTransition`
- Pri stvaranju objekta se zadaju:
  - trajanje (tipa `Duration`)
  - čvor (tipa `Node`)
  - putanja (tipa `Shape`)
- Putanja može biti:
  - proizvoljna zatvorena geometrijska primitiva (oblik) kao što je krug, elipsa, luk ili pravougaonik
  - otvorena primitiva kakva je linija ili putanja

Orijentacija pri prelazu po putanji

- Podrazumevano, čvor ne menja orijentaciju tokom prelaza
- Može se zahtevati i da se njegova orijentacija prilagođava putanji
  - tada je vektor uspravnosti
    (*Y* osa lokalnog koordinatnog sistema)
    uvek normalan na tangentu putanje
- Promena orijentacije se zadaje metodom objekta putanje:
  `setOrientation(PathTransition.OrientationType o)`
  - orijentacija `o` je vrednost tipa nabrajanja:
    `PathTransition.OrientationType.ORTHOGONAL_TO_TANGENT`
    `PathTransition.OrientationType.NONE` (podrazumevano)

Primer prelaza po putanji

```
Ellipse putElipsa = new Ellipse(100.0, 25.0, 75.0, 15.0);
putElipsa.setStroke(Color.RED); putElipsa.setFill(null);
Line putLinija = new Line(30,80, 170,60);
putLinija.setStroke(Color.RED);
Rectangle pravoug1 = new Rectangle(40.0, 20.0, Color.GREEN);
Rectangle pravoug2 = new Rectangle(20.0, 10.0, Color.BLUE);

Duration t = Duration.seconds(5);
PathTransition pp1 = new PathTransition(t, putElipsa, pravoug1);
pp1.setOrientation(PathTransition.OrientationType.
                                        ORTHOGONAL_TO_TANGENT);
PathTransition pp2 = new PathTransition(t, putLinija, pravoug2);
pp2.setOrientation(PathTransition.OrientationType.NONE);

pp1.play();
pp2.play();
```



Animacija vremenskom linijom

- Koraci definisanja vremenske linije:
  - definisanje objekata interpolatora
  - definisanje ključnih vrednosti svojstava ciljnih objekata
  - formiranje objekata ključnih slika koji sadrže ključne vrednosti
  - stvaranje objekta vremenske linije sa ključnim slikama
  - postavljanje svojstava animacije
- Nakon ovih koraka - pokretanje animacije
  - metodom `play()` vremenske linije

Definisanje interpolatora

- Objekat klase `Interpolator`
- Definiše zakon po kojem se menja ključna vrednost
- Na raspolaganju su i sledeći bibliotečki interpolatori:
  - `LINEAR` – linearna (ravnomerna) animacija
  - `EASE_IN` – postepeno se ubzava animacija
  - `EASE_OUT` – postepeno se usporava animacija
  - `EASE_BOTH` – najpre se postepeno ubrzava, a zatim usporava
  - `DISCRETE` – skokovita promena nakon isteka vremena
- Za `EASE` interpolacije se koristi algoritam iz specifikacije SMIL 3.0
  - sa faktorom ubrzanja/usporenja 0.2.
- Podrazumeva se linearni interpolator: `Interpolator.LINEAR`
- Mogu se definisati i specifični interpolatori

Definisanje ključnih vrednosti

- Objekat klase `KeyValue` obuhvata:
  metu, krajnju vrednost, interpolator
  - meta je ciljno svojstvo na koje se odnosi ključna vrednost
  - krajnja vrednost je vrednost ciljnog svojstva na kraju intervala animacije
  - interpolator služi za računanje vrednosti svojstva u intervalu animacije
- Interval završava kada menjana vrednost dostigne krajnju vrednost
- Konstruktori ključnih vrednosti:

```
KeyValue(WritableValue<T> meta, T krajnjaVrednost)
KeyValue(WritableValue<T> meta, T krajnjaVrednost,
                                Iterpolator interpolator)
```

  - prvi oblik podrazumeva primenu podrazumevanog interpolatora

Primer ključnih vrednosti

- Stvaranje 4 ključne vrednosti
- Za dve ključne vrednosti je meta svojstvo X-translacije čvora kruga
  - prva definiše početnu translaciju u pravcu X ose `0`
  - druga definiše krajnju translaciju `150`
- Za dve ključne vrednosti je meta boja popune istog čvora kruga
  - treća definiše početnu crvenu, a četvrta krajnju plavu boju
- Za drugu ključnu vrednost se primenjuje se `EASE_BOTH` interpolator
  - početno ubrzanje i završno usporenje animacije

```
Circle krug = new Circle(25.0, 25.0, 20.0);
KeyValue mesto0 = new KeyValue(krug.translateXProperty(), 0.0);
KeyValue mesto1 = new KeyValue(krug.translateXProperty(),150.0,
                                                Interpolator.EASE_BOTH);
KeyValue boja0 = new KeyValue(krug.fillProperty(), Color.RED);
KeyValue boja1 = new KeyValue(krug.fillProperty(), Color.BLUE);
```

Definisanje ključnih slika (1)

- Objekat klase `KeyFrame`
  - formira se grupisanjem ključnih vrednosti promenljivih svojstava
- Skup ključnih vrednosti određuje stanje čvora
  - stanje je okarakterisano datim vrednostima u ključnoj slici
- Potrebna je i informacija o trenutku ključne slike
  - zadaje se relativno, kao vremenski pomeraj tipa `Duration` u odnosu na trenutak početka animacije
- Ključnoj slici se može dodeliti i ime
  - ime može da se koristi za pozicioniranje na ključnu sliku u animaciji
- Ključnoj slici se može pridružiti i rukovalac događajem akcije
  - događaj se dešava u trenutku dolaska animacije u ključnu sliku

Definisanje ključnih slika (2)

- Konstruktori ključne slike:

```
KeyFrame(Duration t, KeyValue... kVred)
KeyFrame(Duration t, String ime, KeyValue... kVred)
KeyFrame(Duration time,
         EventHandler<ActionEvent> naKraju, KeyValue... kVred)
KeyFrame(Duration time, String ime,
         EventHandler<ActionEvent> naKraju, KeyValue... kVred)
KeyFrame(Duration t, String ime,
         EventHandler<ActionEvent> naKraju,
         Collection<KeyValue> zbirkaKljučnihVrednosti)
```

- Prvi konstruktor definiše
  - trenutak i skup ključnih vrednosti
- Drugi i poslednja dva konstruktora definišu i ime ključne slike
- Poslednja tri konstruktora definišu i rukovalac za obradu događaja akcije

Primer ključnih slika

- Stvaranje početne i krajnje ključne slike pomoću prethodno formiranih ključnih vrednosti
  - prva ključna slika je u početnom trenutku animacije
  - druga ključna slika je nakon 5 sekundi

```
Duration t0 = Duration.ZERO,
         t1 = Duration.seconds(5);
KeyFrame početnaSlika = new KeyFrame(t0, mesto0, boja0);
KeyFrame krajnjaSlika = new KeyFrame(t1, mesto1, boja1);
```

Definisanje vremenske linije (1)

- Objekat klase `Timeline`
  - formira se dodavanjem definisanih ključnih slika
- Ključne slike se mogu dodati pri stvaranju ili kasnije
  - redosled dodavanja je nebitan
    - ključne slike se uređuju prema trenutku prikaza (definisan u svakoj od njih)
- Nakon formiranja v. linije mogu se postavljati svojstva njenog objekta
- Na kraju se može pokrenuti animacija
  - pozivom njegovom metodom `play()`
- Zatim se ključne slike mogu menjati čak i dok traje animacija
  - efekti postaju vidljivi tek nakon zaustavljanja i ponovnog pokretanja animacije
- Objekti ključnih slika vremenske linije se mogu dohvatiti
  - metod: `ObservableList<KeyFrame> getKeyFrames()`

Definisanje vremenske linije (2)

- Konstruktori vremenske linije:

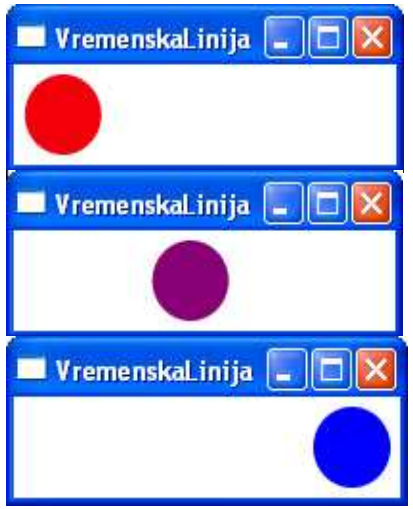
```
Timeline()
Timeline(KeyFrame... ključneSlike)
Timeline(double učestanostSlike)
Timeline(double učestanostSlike, KeyFrame... ključneSlike)
```

- Prvi konstruktor stvara praznu vremensku liniju
  - sa podrazumevanom optimalnom učestanošću osvežavanja slike (*frame rate*)
  - optimalna učestanost – prema proceni izvršnog okruženja
- Drugi i četvrti definišu početni skup ključnih slika vremenske linije
- Treći i četvrti definišu željene učestanosti osvežavanja slike
- Primer:
  - stvaranje vremenske linije sa dve ključne slike i pokretanje animacije:

```
Timeline vl = new Timeline(početnaSlika, krajnjaSlika);
vl.play();
```

Primer animacije vremenskom linijom

```
Circle krug = new Circle(25.0, 25.0, 20.0);
krug.setFill(Color.RED);

KeyValue mesto0 = new KeyValue(krug.translateXProperty(), 0.0);
KeyValue boja0 = new KeyValue(krug.fillProperty(), Color.RED);
KeyValue mesto1 = new KeyValue(krug.translateXProperty(), 150.0,
                                                Interpolator.EASE_BOTH);
KeyValue boja1 = new KeyValue(krug.fillProperty(), Color.BLUE);

Duration t0 = Duration.ZERO, t1 = Duration.seconds(5);

KeyFrame početnaSlika = new KeyFrame(t0, mesto0, boja0);
KeyFrame krajnjaSlika = new KeyFrame(t1, mesto1, boja1);

Timeline vl = new Timeline(početnaSlika, krajnjaSlika);

vl.play();
```



Upravljanje animacijom

- Animacijom se može upravljati
- Upravlja se pomoću metoda odgovarajućeg objekta animacije
  - animacija se može pokrenuti na više načina
  - može se odložiti pokretanje animacije
  - može se zaustaviti ili pauzirati
  - može se ponavljati u petlji
  - može se vraćati unazad
  - može joj se podešavati brzina
- Moguće je dohvatiti stanje animacije i njeno trajanje

Pokretanje animacije

- Nekoliko metoda `play()`:

```
play() // nastavlja animaciju, prvi put od početka
playFromStart()
playFrom(Duration vreme) // od tačke u vremenu
playFrom(String imeTačke)// od označene tačke datog imena
```

- Asinhrono dejstvo – kontrola se odmah vraća
- Više pokrenutih animacija – konkurentno se izvršava
- Postoje i metodi za pomeranje u tačku (bez animacije):

```
jumpTo(Duration vreme)
jumpTo(String imeTačke)
```

Obeležavanje tačaka

- Dva načina:
  - zadavanjem imena ključne slike

```
KeyValue ključnaVrednost = ...
KeyFrame kljucnaSlikaA =
    new KeyFrame(Duration.seconds(5), "A", ključnaVrednost);
```

  - definisanjem imenovanog trenutka u animaciji

```
Animation a = ...
a.getCuePoints().put("B", Duration.seconds(10));
a.getCuePoints().put("C", Duration.seconds(15));
```

Odlaganje, zaustavljanje, pauziranje

- Odlaganje početka za zadato vreme (u odnosu na pokretanje):

```
Animation a = ...
a.setDelay(Duration.seconds(5)); a.play();
```

- Zaustavljanje: metod `stop()`
  - nakon zaustavjanja animacija se vraća u početnu tačku
- Pauziranje (privremeno zaustvljanje): metod `pause()`
- Nastavljanje od tačke u kojoj je zaustavljena: metod `start()`
- Asinhrono dejstvo zaustavljanja i pauziranja, kao kod `play()`
- Oba metoda nemaju efekat ako animacija nije pokrenuta

Ponavljanje i vraćanje unazad

- Ponavljanje zadati broj ciklusa: metod `setCycleCount()`
  - podrazumevana vrednost je 1
  - vrednost `Animation.INDEFINITE` za beskonačnu petlju
- Vraćanje unazad: metod `setAutoReverse(Boolean vraćanje)`
  - u sukcesivnim ciklusima se vrši animacija unapred i unazad
  - za vrednost `vraćanje==true` u parnim ciklusina obavljati unazad
- Promene ovih svojstava u vreme trajanja animacije neće delovati pre zaustavljanja i ponovnog pokretanja animacije

Promena brzine

- Promena brzine: `metod setRate(double v)`
  - nominalna brzina je za `v=1.0` u smeru unapred
  - vrednost `v` predstavlja multplikativni faktor nominalne brzne
  - `v` može biti veće ili manje od `1`
  - znak `v` određuje smer animacije: za `v<0` animacija teče unazad
    - ima smisla tek nakon što je neka animacija već stigla u neku tačku
  - vrednost `v=0` onemogućava animaciju
- Invertovanje brzine: dohvatanje tekuće brzine i promena znaka

```
animacija.setRate(-1.0 * animacija.getRate());
```

  - animacija kreće unazad od tačke u koju je stigla

Događaj završetka animacije

- Moguća je obrada događaja završetka animacije
- Događaj završetka se implicitno dešava na kraju animacije
- Postavljanje rukovaoca završetka: metod `setOnFinished()`
- Na primer:

```
Animation a = ...
a.setOnFinished(e -> System.out.print("Gotovo."));
```

- Događaj se neće desiti:
  - ako se animacija zaustavi pomoću `stop()`
  - ako se program završi pre kraja animacije
  - ako je pokrenuta animacija sa beskonačnim brojem ciklusa