Računarska grafika

JavaFX - mreža,
tekstura, podscena

Pojam proizvoljne geometrije

- JavaFX nudi samo 3 konkretne potklase geometrijskih tela:
  - za kvadar (`Box`), valjak (`Cylinder`) i loptu (`Sphere`)
  - klase su izvedene iz klase `Shape3D`
- Da bi se formirale površi sa proizvoljnom geometrijom
  - potrebno je definisati mrežu trouglova koja određuje površ
  - površ može biti zatvorena (telo) ili otvorena
- Da bi se površ prikazala, potrebne su sledeće informacije :
  - niz temena površi
  - nizu koordinata karakterističnih tačaka teksture
  - niz stranica (trouglova)
    - stranice su određene indeksima temena i tačaka teksture

Trougaona mreža i prikaz mreže

- Proizvoljno geometrijsko telo obrazovano od mreže trouglova
  - tipa `MeshView` izvedene klase iz `Shape3D`
- Klasa `TriangleMesh` – definisanje mreže trouglova
- Objektu klase `MeshView` se postavlja mreža trouglova
- Obe klase nalaze se u paketu `javafx.scene.shape`
- Formiranje proizvoljne površi (tela) tipa `MeshView`:
  - stvore se objekti klase `TriangleMesh` i klase `MeshView`
  - dohvata se zbirka postojećih temena (prazna) i dodaju nova temena
  - dohvata se zbirka postojećih i dodaju nove teksturne koordinate
  - dohvata se zbirka postojećih i dodaju nove stranice (trouglovi)
  - postavi se objektu klase `MeshView` formirani objekat klase `TriangleMesh`

Obrazac za formiranje površi

- Površ se formira na sledeći način:

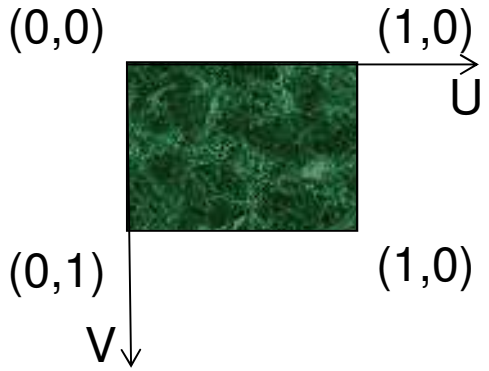
```
TriangleMesh mreža = new TriangleMesh();
mreža.getPoints().addAll(temena);
mreža.getTexCoords().addAll(teksturneKoordinate);
mreža.getFaces().addAll(stranice);
MeshView telo = new MeshView();
telo.setMesh(mreža);
```

Temena površi (tela)

- Temena se zadaju nizom njihovih koordinata
- Koordinate su tipa `float`
- Svako teme je predstavljeno sa 3 koordinate u nizu
- Koordinate se navode po redosledu: x, y, z
- Temena se navode uzastopno: x1, y1, z1, x2, y2, z2, …
- Nebitan je redosled temena u nizu
- Površi treba definisati u lokalnom koordinatnom sistemu
- Centar površi treba da bude u koordinatnom početku
  - po analogiji sa “standardnim” telima koja su tako defiisana
  - olakšava manipulisanje svim površima/telima u sceni

Tekstura

- Teksture su matrice piksela kojima se boji mreža trouglova
- Definišu se:
  - na osnovu učitane slike ili programski
  - u 2D (U-V) koordinatnom sistemu
- U-V koordinatni sistem
  - opseg U i V koordinata teksture je od 0.0 do 1.0
  - kordinata U raste sleva-udesno
  - koordinata V raste odozgo-naniže
- Koordinate četiri ugaona temena teksture, počevši od gornjeg levog ugla, u smeru kazaljke na časovniku:
  - (0,0), (1,0), (1,1), (0,1)



Teksturne koordinate

- Potrebno je zadati tačke teksture u koje se preslikavaju temena
- Koordinate se zadaju nizom njihovih koordinata
- Koordinate su tipa `float`
- Koordinate se zadaju u U-V koordinatnom sistemu
- Svaka relevantna tačka teksture je predstavljena sa 2 koordinate u nizu
- Koordinate se navode po redosledu: u, v
- Tačke teksture se navode uzastopno: u1, v1, u2, v2, …
- Nebitan je redosled tačaka teksture u nizu

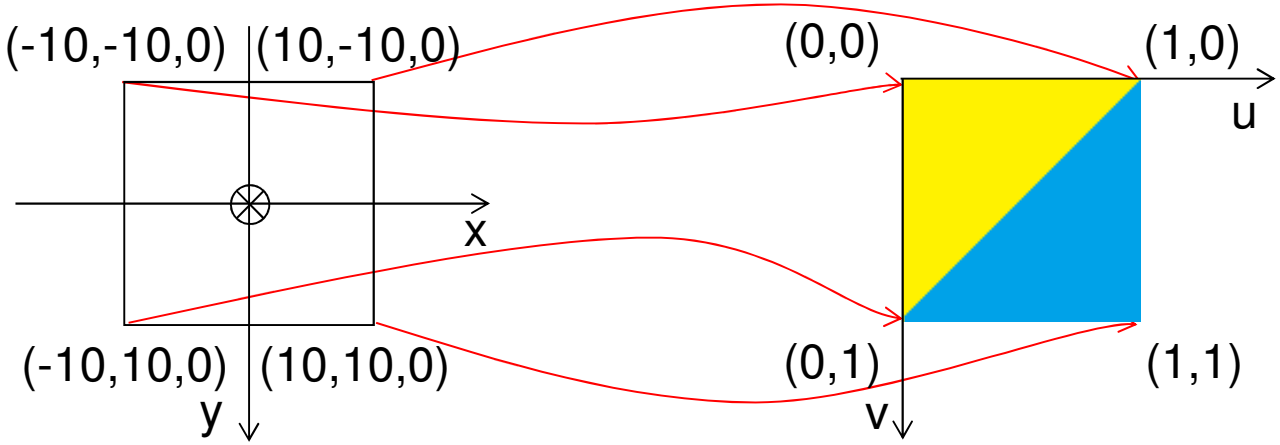
Stranice – trouglovi u mreži

- Stranice se zadaju nizom temena trougova
- Teme trougla je predstavljeno parom indeksa:
  - indeks temena trougla u nizu temena
  - indeks u nizu teksturnih koordinata u koje se preslikava teme trougla
- Indeksi se navode redosledom obilaska temena
  koji određuje normalu na površ po pravilu desne zavojnice
- Svaka stranica mreže se predstavlja sa dva trougla (šest temena)
- Jedan trougao predstavlja spoljnu, a drugi unutrašnju površ stranice
- Jedan trougao u mreži se predstavlja sa 12 indeksa:
  - 2 trougla $\times$ 3 temena $\times$ 2 indeksa

Primer – mreža kvadrata (1)

- Kvadrat je zadat temenima:

i teksturom:

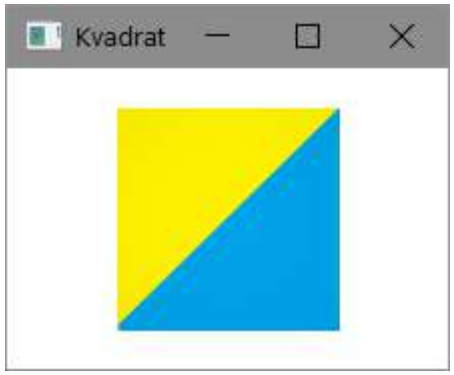


Koordinate temena	Koordinate teksture	Stranice
v0 = (-10, -10, 0) v1 = (-10, 10, 0) v2 = (10, 10, 0) v3 = (10, -10, 0)	t0 = (0, 0) t1 = (0, 1) t2 = (1, 1) t3 = (1, 0)	$f0_{lice}$ = (0, 0, 1, 1, 3, 3) - žuta $f0_{naličje}$ = (0, 0, 3, 3, 1, 1) - žuta $f1_{lice}$ = (1, 1, 2, 2, 3, 3) - plava $f1_{naličje}$ = (1, 1, 3, 3, 2, 2) - plava

Primer – mreža kvadrata (2)

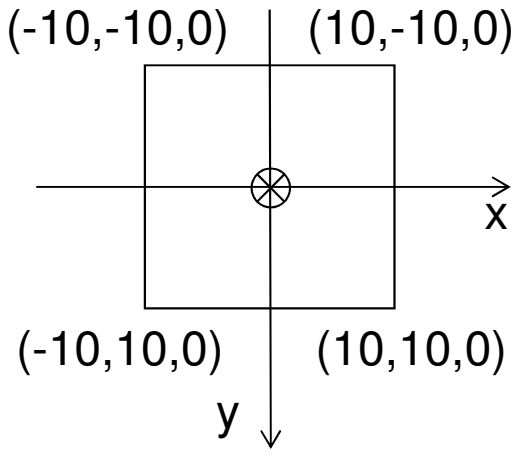
```
...
import javafx.scene.shape.MeshView;
import javafx.scene.shape.TriangleMesh;
import javafx.scene.paint.PhongMaterial;
import javafx.scene.image.Image;

public class Mreza extends Application {
  @Override public void start(Stage prozor) {
    MeshView kvadrat = napraviKvadrat();
    // kod za postavljanje kamere, scene, prozora
  }
```



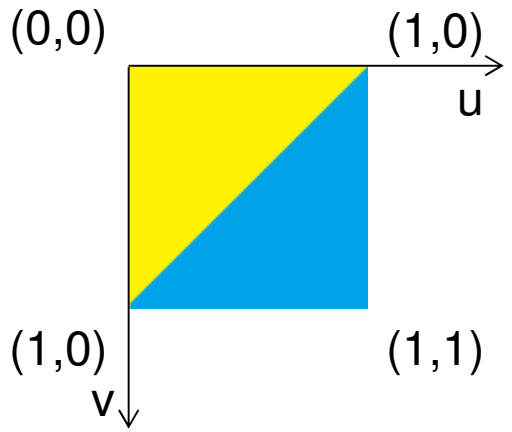
Primer – mreža kvadrata (3)

```
public MeshView napraviKvadrat() {
  // Temena: za svako se daju X, Y i Z koordinata
  float[] temena = {
    -10f, -10f, 0f,  // v0
    -10f,  10f, 0f,  // v1
     10f,  10f, 0f,  // v2
     10f, -10f, 0f   // v3
  };
```



Primer – mreža kvadrata (4)

```
// Teksturne koordinate
float[] teksturneKoordinate = {
   0.0f, 0.0f,   // t0
   0.0f, 1.0f,   // t1
   1.0f, 1.0f,   // t2
   1.0f, 0.0f    // t3
};
```

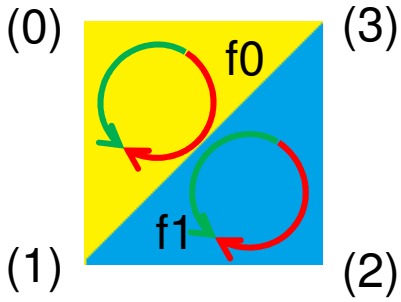


Primer – mreža kvadrata (5)

```
// Stranice: po 2 indeksa za svako teme,
// 3 temena za trougao,
// 2 trougla za stranicu (lice i naličje)
int[] stranice = {
  0, 0, 1, 1, 3, 3, // iv0,it0,iv1,it1,iv3,it3 (f0 lice)
  0, 0, 3, 3, 1, 1, // iv0,it0,iv3,it3,iv1,it1 (f0 naličje)
  1, 1, 2, 2, 3, 3, // iv1,it1,iv2,it2,iv3,it3 (f1 lice)
  1, 1, 3, 3, 2, 2  // iv1,it1,iv3,it3,iv2,it2 (f1 naličje)
};
```

lice

naličje



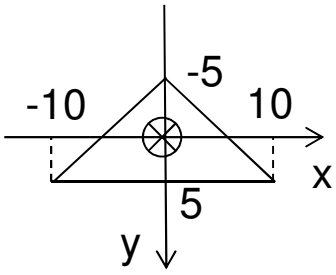
Primer – mreža kvadrata (6)

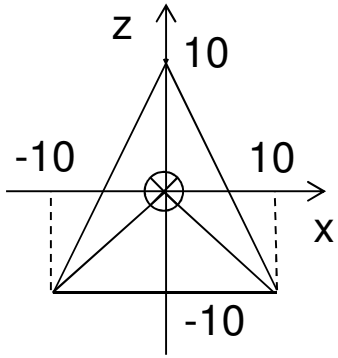
```
    TriangleMesh mreža = new TriangleMesh();
    mreža.getPoints().addAll(temena);
    mreža.getTexCoords().addAll(teksturneKoordinate);
    mreža.getFaces().addAll(stranice);
    MeshView kvadrat = new MeshView();
    kvadrat.setMesh(mreža);

    PhongMaterial mat = new PhongMaterial();
    mat.setDiffuseMap(new Image("ZutoPlaviKvadrat.png"));
    kvadrat.setMaterial(mat);
    return kvadrat;
}
```

Primer – mreža tetraedra (1)

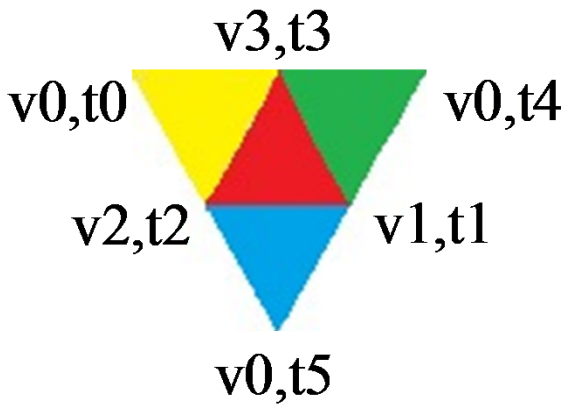
- Opis tetraedra:
  - osnovica (crvena) u ravni y=5
  - vrh na Y-osi u tački y=-5
  - prednja (plava) stranica
    ivica na osnovici paralelna X-osi, x ∈ [-10,10], z=-10, y=5
  - bočna desna (zelena) i bočna leva (žuta) stranica
    imaju zajedničko teme u tački (0, 5, 10)





Primer – mreža tetraedra (2)

- Temena i tekstura

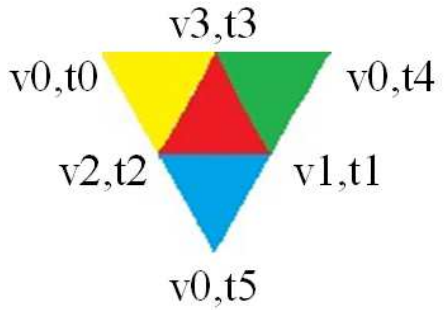


Teksturirana mreža

Alternativna tekstura

Primer – mreža tetraedra (3)

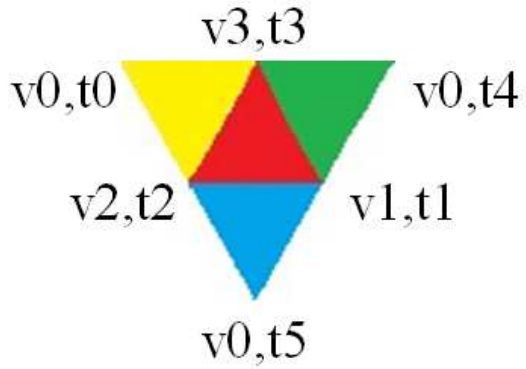
- Koordinate temena i teksture
- Stranice
  - indeksi temena i tekstura



Koordinate temena	Koordinate teksture	Stranice tetraedra
v0 = (0, -5, 0) v1 = (10, 5, -10) v2 = (-10, 5, -10) v3 = (0, 5, 10)	t0 = (0.00, 0.00) t1 = (0.75, 0.50) t2 = (0.25, 0.50) t3 = (0.50, 0.00) t4 = (1.00, 0.00) t5 = (0.50, 1.00)	$f0_{lice}$ = (0, 5, 2, 2, 1, 1) - plava $f0_{naličje}$ = (0, 5, 1, 1, 2, 2) - plava $f1_{lice}$ = (1, 1, 3, 3, 0, 4) - zelena $f1_{naličje}$ = (1, 1, 0, 4, 3, 3) - zelena $f2_{lice}$ = (0, 0, 3, 3, 2, 2) - žuta $f2_{naličje}$ = (0, 0, 2, 2, 3, 3) - žuta $f3_{lice}$ = (1, 1, 2, 2, 3, 3) - crvena $f3_{naličje}$ = (1, 1, 3, 3, 2, 2) - crvena

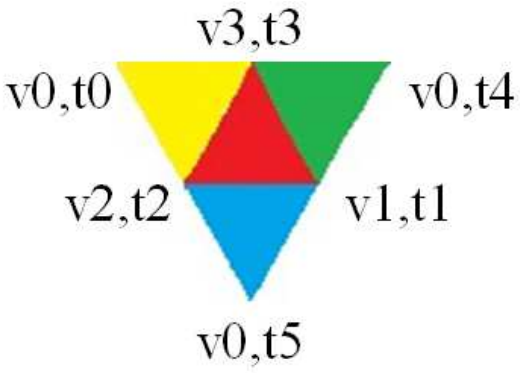
Primer – mreža tetraedra (4)

```
import javafx.scene.shape.MeshView;
import javafx.scene.shape.TriangleMesh;
import javafx.scene.paint.PhongMaterial;
...
public MeshView napraviTetraedar() {
// Koordinate temena: za svako teme se daje x, y, z
   float[] temena = {
        0f, -5f,   0f, // v0
       10f,  5f, -10f, // v1
      -10f,  5f, -10f, // v2
        0f,  5f,  10f  // v3
   };
```



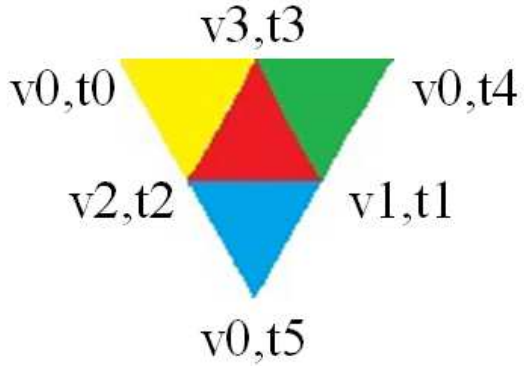
Primer – mreža tetraedra (5)

```
// Teksturne koord.: U, V koord. karakterističnih tačaka
        float[] teksturneKoordinate = {
            0.00f, 0.0f, // t0
            0.75f, 0.5f, // t1
            0.25f, 0.5f, // t2
            0.50f, 0.0f, // t3
            1.00f, 0.0f, // t4
            0.50f, 1.0f, // t5
        };
```



Primer – mreža tetraedra (6)

```
// Stranice: par indeksa za svako teme, 3 temena za trougao,
//           2 trougla za stranicu (lice i naličje)
   int[] stranice = {
      0, 5, 2, 2, 1, 1, // f0 lice
      0, 5, 1, 1, 2, 2, // f0 naličje
      1, 1, 3, 3, 0, 4, // f1 lice
      1, 1, 0, 4, 3, 3, // f1 naličje
      0, 0, 3, 3, 2, 2, // f2 lice
      0, 0, 2, 2, 3, 3, // f2 naličje
      1, 1, 2, 2, 3, 3, // f3 lice
      1, 1, 3, 3, 2, 2, // f3 naličje
   };
```

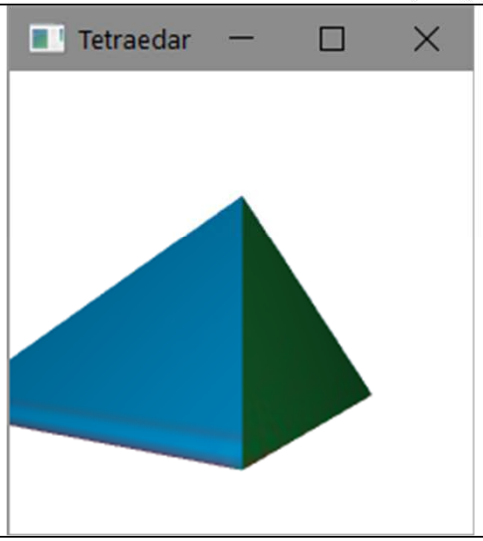
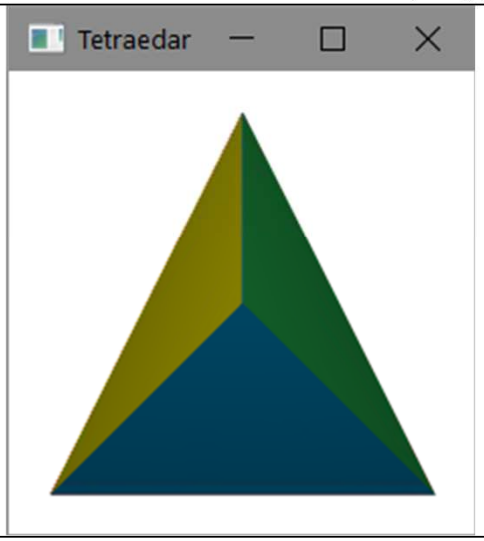


Primer – mreža tetraedra (7)

```
    TriangleMesh mreža = new TriangleMesh();
    mreža.getPoints().addAll(temena);
    mreža.getTexCoords().addAll(teksturneKoordinate);
    mreža.getFaces().addAll(stranice);
    MeshView tetraedar = new MeshView();
    tetraedar.setMesh(mreža);

    PhongMaterial mat = new PhongMaterial();
    mat.setDiffuseMap(new Image("TeksturaTetraedar.jpg"));
    tetraedar.setMaterial(mat);
    return tetraedar;
}
```

Primer – mreža tetraedra (8)

bez rotacije	`tetraedar.setRotationAxis (Rotate.Y_AXIS); tetraedar.setRotate(45);`	`tetraedar.setRotationAxis (Rotate.X_AXIS); tetraedar.setRotate(90);`
		

Primer – mreža tetraedra (9)

- Druga (jednostavnija) mogućnost - skala boja
  - za slučaj da su stranice monolitno obojene

```
float[] teksturneKoordinate = {
            0.1f, 0.5f, // 0 crvena
            0.3f, 0.5f, // 1 zelena
            0.5f, 0.5f, // 2 plava
            0.7f, 0.5f  // 3 žuta
        };
```

Primer – mreža tetraedra (10)

- Stranice:

```
int[] stranice = {
            0, 2, 2, 2, 1, 2, // f0 lice
            0, 2, 1, 2, 2, 2, // f0 naličje
            1, 1, 3, 1, 0, 1, // f1 lice
            1, 1, 0, 1, 3, 1, // f1 naličje
            0, 3, 3, 3, 2, 3, // f2 lice
            0, 3, 2, 3, 3, 3, // f2 naličje
            1, 0, 2, 0, 3, 0, // f3 lice
            1, 0, 3, 0, 2, 0, // f3 naličje
        };
```

Podscena

- U 3D sceni može postojati
  - proizvoljan broj čvorova 3D objekata i izvora svetala
  - samo jedna kamera
- Ukoliko se želi više kamera (više pogleda na scenu)
  - potrebno je uvesti podscene
  - svakoj podsceni pridružiti njenu kameru
- Podscena je, kao i scena, kontejner grafa scene
- Opisana je klasom `SubScene`
  - izvedenom iz klase `Node`
  - definisanom u paketu `javafx.scene`

Parametri podscene

- Podscena ima svojstva i atribute kao i scena:
  - graf scene
  - visinu i širinu
  - boju pozadine
  - z-bafer
  - glatkoću ivica objekata (*antialiasing*)
  - vlastitu kameru
- Konstruktor podscene ima iste parametre kao i konstruktor scene
- Ako se ne postavi kamera podscene – podrazumevana paralelna
- Ako postoje 3D objekti u podsceni – podrazumevano svetlo
  - tačkasti izvor belog svetla na kameri (*headlight*)

Korišćenje podscene

- `SubScene` je izvedena iz `Node`
  - primercima podscene mogu da se koriste kao čvorovi u sceni
- Graf scene može sadržati više podscena
  - podscene se kombinuju u jednu scenu
- Kompozicija scene – i graf podscene može sadržati podscene
- Podscene se mogu koristiti da bi se kombinovala 2D i 3D grafika
  - neke podscene mogu biti 3D, dok druga mogu biti 2D
- Podscene imaju podrazumevano transparentnu pozadinu
  - moguće je vizuelno preklopiti njihov prikaz i dobiti jedinstvenu sliku

Primer podscene (1)

- Dve podscene od kojih se obrazuje jedna scena
- U prvoj podsceni je kvadar:
  - prikazan perspektivnom kamerom sa okom koje nije u nuli
    - rotiranom oko X ose za -30°
  - osvetljen odozgo svetlom boje CYAN
- U drugoj podsceni je lopta:
  - prikazana podrazumevanom paralelnom kamerom
  - osvetljena sleva svetlom boje MAGENTA

Primer podscene (2)

```
...
import javafx.scene.Scene;
import javafx.scene.SubScene;
import javafx.scene.SceneAntialiasing;
import javafx.scene.Group;
public class Podscena extends Application {
  @Override public void start(Stage prozor) {
    // definisanje i pozicioniranje kvadra, lopte i svetala
    PerspectiveCamera kamKva= new PerspectiveCamera(false);
    kamKva.setRotationAxis(Rotate.X_AXIS);
    kamKva.setRotate(-30);
```

Primer podscene (3)

```
Group grupaKva = new Group(kvadar, svetloKva);
SceneAntialiasing glatko=SceneAntialiasing.BALANCED;
SubScene scenaKva = new SubScene(
                          grupaKva, 230, 70, true, glatko);
scenaKva.setCamera(kamKva);
Group grupaLop = new Group(lopta, svetloLop);
SubScene scenaLop = new SubScene(
                          grupaLop, 230, 70, true, glatko);
Group koren = new Group();
koren.getChildren().addAll(scenaKva,scenaLop);
Scene scena = new Scene(koren, 230, 70, true, glatko);
...
```