# Računarska grafika - vežbe

## 7 – JavaFX 3D
## materijal

# Ispitivanje materijala (1/3)

Primer programa za ispitivanje osobina materijala.

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.SceneAntialiasing;
import javafx.scene.Group;
import javafx.scene.PerspectiveCamera;
import javafx.scene.PointLight;
import javafx.scene.shape.Sphere;
import javafx.scene.paint.Color;
import javafx.scene.paint.PhongMaterial;
import javafx.scene.image.Image;
```

# Ispitivanje materijala (2/3)

```
public class Materijal extends Application {
  @Override public void start(Stage prozor) {
    Sphere lopta = new Sphere(100);
    lopta.setTranslateZ(400);
    PhongMaterial mat = new PhongMaterial();
    mat.setDiffuseColor(Color.CYAN);
    mat.setDiffuseMap(new Image("slike/kapi.jpg"));
//  mat.setSpecularColor(Color.CYAN);
//  mat.setSpecularColor(Color.WHITE);
    mat.setSpecularColor(Color.YELLOW);
    mat.setSpecularPower(1);
    mat.setSpecularMap( new Image("slike/VertikalnePruge.png" ) );
//  mat.setSpecularMap( new Image("slike/CrnoBelo.jpg" ) );
    mat.setBumpMap(new Image("slike/normale.jpg"));
    mat.setSelfIlluminationMap(new Image("slike/CrvenoZel.jpg" ));
    lopta.setMaterial(mat);
```

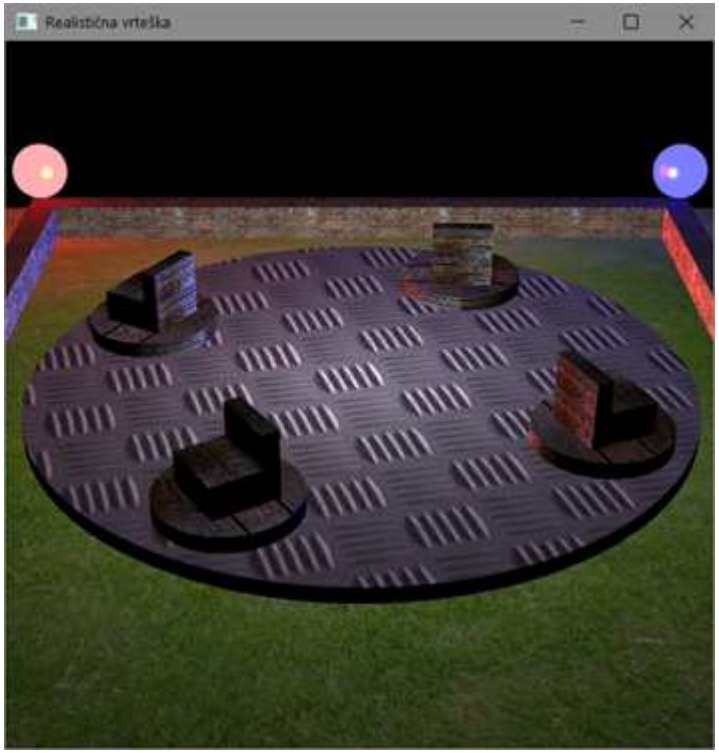
# Ispitivanje materijala (3/3)

```
    PointLight svetlo = new PointLight();
    svetlo.setTranslateX(100);
    svetlo.setTranslateY(100);
    svetlo.setTranslateZ(-200);
    Group koren = new Group(lopta, svetlo);
    SceneAntialiasing nazupčenost=SceneAntialiasing.BALANCED;
    Scene scena = new Scene(koren, 270, 220, true, nazupčenost);
    PerspectiveCamera kamera=new PerspectiveCamera(true);
    kamera.setNearClip(0.1);
    kamera.setFarClip(1000);
    scena.setCamera(kamera);
    prozor.setScene(scena);
    prozor.setTitle("Materijal");
    prozor.show();
  }
  public static void main(String[] args) { launch(args); }
}
```

# Zadatak 1: Realistična vrteška

Dopuniti rešenje zadatka „Vrteška“ osobinama materijala i dodatnim osvetljenjem scene, prema navedenim zahtevima:

- vrteška se nalazi na travnatoj podlozi;
- oko vrteške je zidić od cigala sa tri strane (pozadi, levo i desno);
- platforma je od metala;
- stolice sa podijumima su od dasaka;
- na krajevima zidića pozadi su dve svetleće kugle, plava – desno i crvena – levo.



# Rešenje 1: Realistična vrteška (1/6)

```
// kod iz Vrteska.java
import javafx.scene.shape.Sphere;
import javafx.scene.image.Image;

public class RealisticnaVrteska extends Application {
   @Override public void start(Stage prozor) {
        Group okruženje=napraviOkruženje();
        Group vrteška = new Group();
        Group sve = new Group(okruženje,vrteška);
        PointLight svetlo=napraviSvetla(okruženje);
        // kod iz Vrteska.java
        scena.setFill(Color.BLACK);
        // kod iz Vrteska.java
    }
```

# Rešenje: Realistična vrteška (2/6)

```
private Group napraviOkruženje(){
  Box podloga = new Box(600,1,600);
  PhongMaterial matTrava = new PhongMaterial();
  matTrava.setDiffuseMap( new Image("trava1.jpg") );
  podloga.setMaterial(matTrava);

  PhongMaterial matZid = new PhongMaterial();
  matZid.setDiffuseMap( new Image("dugizid.png") );
  Box zadnjaOgrada = new Box(500,50,20);
  zadnjaOgrada.setTranslateZ(250);
  zadnjaOgrada.setMaterial(matZid);
  Box levaOgrada = new Box(20,50,500);
  levaOgrada.setTranslateX(-240);
  levaOgrada.setMaterial(matZid);
```

# Rešenje: Realistična vrteška (3/6)

```
Box desnaOgrada = new Box(20,50,500);
desnaOgrada.setTranslateX(240);
desnaOgrada.setMaterial(matZid);
PhongMaterial matLevaLampa = new PhongMaterial();
matLevaLampa.setDiffuseColor(Color.RED);
matLevaLampa.setSpecularColor(Color.YELLOW);
matLevaLampa.setSpecularPower(20);
matLevaLampa.setSelfIlluminationMap(new Image("crvenalampa.png"));
Sphere levaLampa = new Sphere(20);
levaLampa.setTranslateZ(250);
levaLampa.setTranslateY(-50);
levaLampa.setTranslateX(-240);
levaLampa.setMaterial(matLevaLampa);
```

# Rešenje: Realistična vrteška (4/6)

```
    Sphere desnaLampa = new Sphere(20);
    desnaLampa.setTranslateZ(250);
    desnaLampa.setTranslateY(-50);
    desnaLampa.setTranslateX(240);
    PhongMaterial matDesnaLampa = new PhongMaterial();
    matDesnaLampa.setDiffuseColor(Color.BLUE);
    matDesnaLampa.setSpecularColor(Color.YELLOW);
    matDesnaLampa.setSpecularPower(20);
    matDesnaLampa.setSelfIlluminationMap(new Image("plavaLampa.png"));
    desnaLampa.setMaterial(matDesnaLampa);
    Group okruženje=new Group(podloga, zadnjaOgrada, levaOgrada,
                              desnaOgrada, levaLampa, desnaLampa);
    return okruženje;
}
```

# Rešenje: Realistična vrteška (5/6)

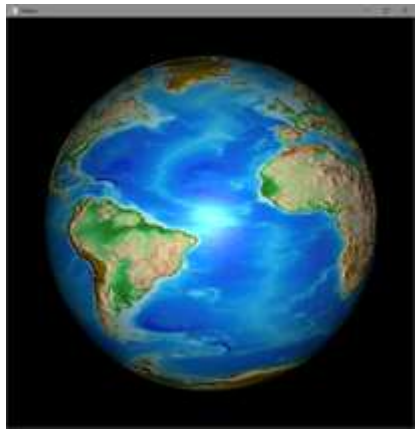
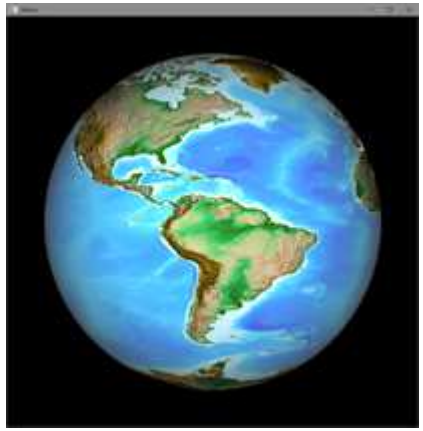
```
    private void napraviVrtećiPodijum(Group g){ // stari kod
      PhongMaterial mat = new PhongMaterial();
      mat.setDiffuseMap( new Image("metal.jpg"));
      podijum.setMaterial(mat);
      // kod iz Vrteska.java
    }
    private Group napraviStolicu(){// stari kod
      PhongMaterial mat = new PhongMaterial();
      mat.setDiffuseMap( new Image("daske.jpg"));
      postolje.setMaterial(mat);
      sediste.setMaterial(mat);
      naslon.setMaterial(mat);
      // kod iz Vrteska.java
}
```

# Rešenje: Realistična vrteška (6/6)

```
private PointLight napraviSvetla(Group svetla){
  PointLight svetlo = new PointLight();
  svetlo.setTranslateY(-100);
  svetlo.setColor(Color.WHITE);
  PointLight levaLampa = new PointLight(Color.RED);
  levaLampa.setTranslateZ(250);
  levaLampa.setTranslateY(-50);
  levaLampa.setTranslateX(-240);
  PointLight desnaLampa = new PointLight(Color.BLUE);
  desnaLampa.setTranslateZ(250);
  desnaLampa.setTranslateY(-50);
  desnaLampa.setTranslateX(240);
  svetla.getChildren().addAll(svetlo,levaLampa,desnaLampa);
  return svetlo;
}
```

# Zadatak 2: Globus (1/2)

Napisati program koji iscrtava rotirajući globus. Globus ima vizuelno reljefnu mapu kopna i odsjaj samo od vodenih površina. Pritiskom tastera miša se zaustavlja rotacija, a otpuštanjem nastavlja. Ulaskom kurzora u prozor, globus se samo-osvetljava, ali samo na vodenim površinama. Program se pokreće u režimu slike preko celog ekrana (*full screen mode*), a pritiskom tastera ESC prelazi u režim prozora.

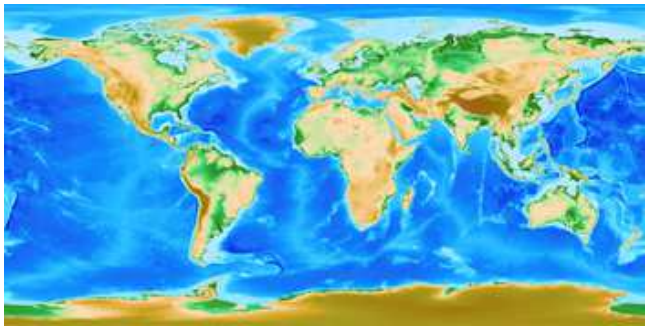




# Zadatak 2: Globus (2/2)

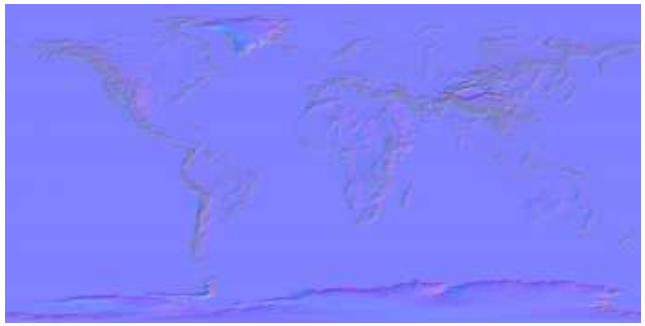
Na raspolaganju su sledeće slike u fajlovima:

(preuzete sa sajta: http://www.wthr.us/2013/05/23/bored-then-create-a-planet/ )

zemlja-mapa.jpg



zemlja-normale.jpg

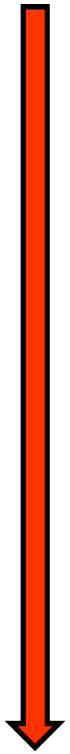


zemlja-sjaj.jpg

zemlja-svetlo.jpg

# Rešenje: Globus (1/6)

Prilagođen program, interakcija: http://stackoverflow.com/questions/19621423/javafx-materials-bump-and-spec-maps



```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.SceneAntialiasing;
import javafx.scene.Group;
import javafx.scene.PerspectiveCamera;
import javafx.scene.image.Image;
import javafx.scene.paint.Color;
import javafx.scene.paint.PhongMaterial;
import javafx.scene.shape.Sphere;
```

# Rešenje: Globus (2/6)

```
import javafx.scene.transform.Rotate;
import javafx.animation.RotateTransition;
import javafx.animation.Interpolator;
import javafx.util.Duration;
import javafx.event.EventHandler;
import javafx.scene.input.MouseEvent;

public class Globus extends Application {

 // kod na sledećim slajdovima

public static void main(String[] args) { launch(args); }
}
```

# Rešenje: Globus (3/6)

```
@Override public void start(Stage prozor) {
    Sphere zemlja = new Sphere(400);
    PhongMaterial materijal = new PhongMaterial();
    materijal.setDiffuseMap( new Image("zemlja-mapa.jpg"));
    materijal.setBumpMap(new Image("zemlja-normale.jpg"));
    materijal.setSpecularMap( new Image("zemlja-sjaj.jpg" ));
    zemlja.setMaterial(materijal );
    Group koren = new Group(zemlja);
```

# Rešenje: Globus (4/6)

```
SceneAntialiasing glatko=SceneAntialiasing.BALANCED;
Scene scena = new Scene(koren, 1000, 1000, true, glatko);
scena.setFill(Color.BLACK);
PerspectiveCamera kamera = new PerspectiveCamera(true);
kamera.setTranslateZ(-2000);
kamera.setFarClip(2100);
scena.setCamera(kamera);

prozor.setScene(scena);
prozor.setTitle("Globus");
prozor.show();
prozor.setFullScreen(true);
```

# Rešenje: Globus (5/6)

```
RotateTransition rot = new RotateTransition(
                       Duration.seconds(30), zemlja );
rot.setAxis(Rotate.Y_AXIS);
rot.setFromAngle(360);
rot.setToAngle(0);
rot.setInterpolator(Interpolator.LINEAR);
rot.setCycleCount(RotateTransition.INDEFINITE);
rot.play();
```

# Rešenje: Globus (6/6)

```
    EventHandler<MouseEvent> r1 = dog -> rot.pause();
    EventHandler<MouseEvent> r2 = dog -> rot.play();
    scena.addEventHandler(MouseEvent.MOUSE_PRESSED, r1);
    scena.addEventHandler(MouseEvent.MOUSE_RELEASED, r2);

    Image samoOsvetljenje = new Image("zemlja-svetlo.jpg");
    EventHandler<MouseEvent> r3 = dog ->
        materijal.setSelfIlluminationMap(samoOsvetljenje);
    EventHandler<MouseEvent> r4 = dog ->
        materijal.setSelfIlluminationMap(null);
    scena.addEventHandler(MouseEvent.MOUSE_ENTERED, r3);
    scena.addEventHandler(MouseEvent.MOUSE_EXITED, r4);
}
```

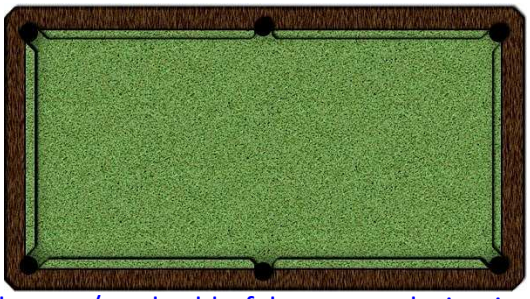
# Zadatak 3: Bilijarski sto (1/2)

Napisati program koji crta bilijarski sto sa kuglama i lusterom iznad stola. Gornja površina stola je samo prividno trodimenzionalna, koristiti sliku bilijarskog stola za teksturu ploče debljine 1 piksela podignutu na sto od dasaka. Pored lustera, scena se osvetljava i lampom na kameri. Tri kugle slučajne boje se slučajno raspoređuju na stolu, ali tako da se ne preklope. Predvideti sledeću interakciju:

- `Space` → paljenje/gašenje lustera
- `Enter` → zamena kugli na stolu
- `LeftMouseDrag` → kretanje kamere po sferi iznad scene, orjentacija prema centru
- `RightMouseDrag` → X-Y translacija u lokalnom sistemu kamere (pan)
- `MouseWheelScroll` → Z translacija kamere u lokalnom sistemu (zoom in/out)

# Zadatak 3: Bilijarski sto (2/2)

- Slika gornje površine stola:



http://hyunky.com/pool-table-felt-texture-design-inspiration-10

Slika dasaka za kutiju:

https://freestocktextures.com/photos-wood/

- Ciljna 3D scena:


# Rešenje: Bilijarski sto (1/16)

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.SceneAntialiasing;
import javafx.scene.Group;
import javafx.scene.shape.Cylinder;
import javafx.scene.shape.Box;
import javafx.scene.shape.Sphere;
import javafx.scene.paint.PhongMaterial;
import javafx.scene.paint.Color;
import javafx.scene.PerspectiveCamera;
import javafx.scene.PointLight;
import javafx.scene.transform.Rotate;
```

# Rešenje: Bilijarski sto (2/16)

```
import javafx.event.EventHandler;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;
import javafx.scene.input.MouseEvent;
import javafx.scene.input.ScrollEvent;
import javafx.scene.image.Image;
import java.util.Random;

public class Bilijar extends Application {

  // kod na sledećim slajdovima

  public static void main(String[] args) { launch(args); }
}
```

# Rešenje: Bilijarski sto (3/16)

```
private static final double DUŽINA_STOLA = 600.0;
private static final double ŠIRINA_STOLA = 400.0;
private static final double VISINA_STOLA = 100.0;
private static final double DEBLJINA_PLOČE = 1.0;
private static final double MARTINELA = 50.0;
private static final double POLUPREČNIK_ABAŽURA = 20.0;
private static final double VISINA_ABAŽURA = 20.0;
private static final double DEBLJINA_STAKLA = 1.0;
private static final double VISINA_LUSTERA = 150.0;
private static final double PREČNIK_KUGLE = 20.0;
private static final double DALJA_ODSECAJUĆA_RAVAN = 2000.0;
private static final double POČETNO_RASTOJANJE_KAMERE = -800.0;
```

# Rešenje: Bilijarski sto (4/16)

```
private final Rotate rx = new Rotate();
private final Rotate ry = new Rotate();
private final Rotate rz = new Rotate();
private double pozX;                private double pozY;
private double staraPozX;           private double staraPozY;
private PerspectiveCamera kamera;
private PointLight svetlo;
private final Group koren = new Group();
private final Group sto = new Group();
private final Group kugle = new Group();
private final Group luster = new Group();
private final Group nosač = new Group();
private static final Random RND = new Random();
```

# Rešenje: Bilijarski sto (5/16)

```
private void napraviSto(){
  PhongMaterial matTelo = new PhongMaterial();
  matTelo.setDiffuseMap(new Image("daske.jpg"));
  Box telo = new Box(DUŽINA_STOLA,VISINA_STOLA,ŠIRINA_STOLA);
  telo.setMaterial(matTelo);

  PhongMaterial matPloča = new PhongMaterial();
  matPloča.setDiffuseMap(new Image("bilijarsto.jpg"));
  Box ploča = new Box(DUŽINA_STOLA,DEBLJINA_PLOČE,ŠIRINA_STOLA);
  ploča.setMaterial(matPloča);
  ploča.setTranslateY(-(VISINA_STOLA+DEBLJINA_PLOČE)/2);

  sto.getChildren().addAll(telo, ploča);
}
```

# Rešenje: Bilijarski sto (6/16)

```
private Sphere napraviKuglu(double r, Color boja){
  PhongMaterial mat = new PhongMaterial();
  mat.setDiffuseColor(boja);
  mat.setSpecularColor(Color.YELLOW);
  Sphere kugla = new Sphere(r,100);
  kugla.setMaterial(mat);
  return kugla;
}
```

# Rešenje: Bilijarski sto (7/16)

```
private void napraviSkupKugli(int n) {
  kugle.getChildren().clear();
  double ugao = 0;
  final double deltaUgla = 2*Math.PI/n;
  double odstupanjeUgla;
  double deltaTona = 360/n;
  double pocetniTon = 0;
  for(int i=0; i<n; i++) {
    double ton= RND.nextDouble()*deltaTona+pocetniTon+deltaTona/2;
    pocetniTon+=deltaTona;
    double zasićenje=RND.nextDouble()*0.3+0.7;
    double sjaj=RND.nextDouble()*0.3+0.7;
    Color boja = Color.hsb(ton, zasićenje, sjaj);
    Sphere kugla = napraviKuglu(PREČNIK_KUGLE/2.0, boja);
```

# Rešenje: Bilijarski sto (8/16)

```
    final double minRadius = PREČNIK_KUGLE*2;
    final double maxRadius = ŠIRINA_STOLA/2.0-MARTINELA;
    double radius=RND.nextDouble()*(maxRadius-minRadius)+minRadius;
    odstupanjeUgla=RND.nextDouble()*deltaUgla/2;
    kugla.setTranslateX(Math.cos(ugao+odstupanjeUgla)*radius);
    kugla.setTranslateZ(Math.sin(ugao+odstupanjeUgla)*radius);
    kugle.getChildren().add(kugla);
    ugao+=deltaUgla;
  }
  double pomeraj = -(VISINA_STOLA+PREČNIK_KUGLE)/2-DEBLJINA_PLOČE;
  kugle.setTranslateY(pomeraj);
}
```

# Rešenje: Bilijarski sto (9/16)

```
private void napraviKameru() {
  kamera = new PerspectiveCamera(true);
  kamera.setFarClip(DALJA_ODSECAJUĆA_RAVAN);
  kamera.setTranslateZ(POČETNO_RASTOJANJE_KAMERE);
}
```

# Rešenje: Bilijarski sto (10/16)

```
private void napraviNosačKamereSaLampom(){
  PointLight lampa = new PointLight();
  lampa.setColor(Color.WHITE);
  lampa.setTranslateX(kamera.getTranslateX());
  lampa.setTranslateY(kamera.getTranslateY());
  lampa.setTranslateZ(kamera.getTranslateZ());
  nosač.getChildren().addAll(kamera, lampa);
  rx.setAxis(Rotate.X_AXIS); rx.setAngle(-20);
  ry.setAxis(Rotate.Y_AXIS); ry.setAngle(-45);
  rz.setAxis(Rotate.Z_AXIS); rz.setAngle(0);
  nosač.getTransforms().addAll(rz, ry, rx);
}
```

# Rešenje: Bilijarski sto (11/16)

```
private void napraviLuster() {
  svetlo = new PointLight();
  svetlo.setTranslateY(-VISINA_LUSTERA);
  svetlo.setColor(Color.WHITE);

  PhongMaterial matAbažur = new PhongMaterial();
  matAbažur.setDiffuseColor(Color.DARKGOLDENROD);
  matAbažur.setSpecularColor(Color.RED);
  Cylinder abažur =
            new Cylinder(POLUPREČNIK_ABAŽURA,VISINA_ABAŽURA,20);
  abažur.setTranslateY(-VISINA_LUSTERA);
  abažur.setMaterial(matAbažur);
```

# Rešenje: Bilijarski sto (12/16)

```
    PhongMaterial matStaklo = new PhongMaterial();
    matStaklo.setDiffuseColor(Color.BLUE);
    matStaklo.setSpecularColor(Color.RED);
    matStaklo.setSelfIlluminationMap(
                              new Image("CrvenaLampa.png"));
    Cylinder staklo =
              new Cylinder(POLUPREČNIK_ABAŽURA,DEBLJINA_STAKLA,20);
    staklo.setTranslateY(-VISINA_LUSTERA+VISINA_ABAŽURA/2);
    staklo.setMaterial(matStaklo);

    luster.getChildren().addAll(svetlo, abažur, staklo);
  }
```

# Rešenje: Bilijarski sto (13/16)

```
private Scene napraviScenu() {
  EventHandler<KeyEvent> r = d -> { // pritisak tastera
    KeyCode k=d.getCode();
    switch (k) {
      case SPACE: svetlo.setLightOn(!svetlo.isLightOn()); break;
      case ENTER: kugle.getChildren().clear(); napraviSkupKugli(3);
    }
  }; // obrada događaja pritiska tastera

  EventHandler<MouseEvent> r1 = dog -> { // pritisak miša
            staraPozX = pozX = dog.getSceneX();
            staraPozX = pozY = dog.getSceneY();
  }; // obrada događaja pritiska miša
```

# Rešenje: Bilijarski sto (14/16)

```
EventHandler<MouseEvent> r2 = dog -> { // vučenje miša
  staraPozX = pozX; pozX = dog.getSceneX();
  staraPozY = pozY; pozY = dog.getSceneY();
  double korakX = (pozX - staraPozX), korakY = (pozY - staraPozY);
  if (dog.isPrimaryButtonDown()) {
    ry.setAngle(ry.getAngle() - korakX);
    rx.setAngle(rx.getAngle() + korakY);
  } else if (dog.isSecondaryButtonDown()) {
    double x = kamera.getTranslateX();
    kamera.setTranslateX(kamera.getTranslateX() + korakX);
    double y = kamera.getTranslateY();
    kamera.setTranslateY(kamera.getTranslateY() + korakY);
  }
}; // obrada događaja vučenja miša
```

# Rešenje: Bilijarski sto (15/16)

```
  EventHandler<ScrollEvent> r3 = dog -> { // točkić miša
    double korakZ = dog.getDeltaY();
    double z = kamera.getTranslateZ();
    kamera.setTranslateZ(z + korakZ);
   };  // obrada dogadjaja točkića miša
   SceneAntialiasing glatko=SceneAntialiasing.BALANCED;
   Scene scena = new Scene(koren, 1000, 600, true, glatko);
   scena.setFill(Color.BLACK);  scena.setCamera(kamera);
   scena.addEventHandler(KeyEvent.KEY_PRESSED, r);
   scena.addEventHandler(MouseEvent.MOUSE_PRESSED, r1);
   scena.addEventHandler(MouseEvent.MOUSE_DRAGGED, r2);
   scena.addEventHandler(ScrollEvent.SCROLL, r3);
   return scena;
 }
```

# Rešenje: Bilijarski sto (16/16)

```
 public void napraviGrafScene(){
    napraviSto();                napraviSkupKugli(3);
    napraviLuster();             napraviKameru();
    napraviNosačKamereSaLampom();
    koren.getChildren().addAll(sto, kugle, luster, nosač);
}

@Override public void start(Stage prozor) {
    napraviGrafScene();
    Scene scena = napraviScenu();
    prozor.setTitle("Bilijar");
    prozor.setScene(scena);
    prozor.show();
}
```