Računarska grafika - vežbe

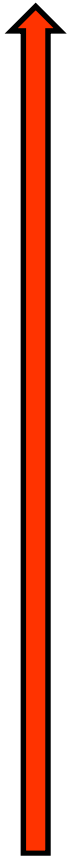
5 – JavaFX 3D
scena, objekti, kamera, svetlo

Test 3D scene (1/2)

Program za test 3D scene:

```
import javafx.application.Application;
import javafx.application.ConditionalFeature;
import javafx.application.Platform;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.text.Text;
import javafx.stage.Stage;

public class Test3Dscene extends Application {
    @Override public void start(Stage prozor) {
        boolean podržana = Platform.isSupported(
                                        ConditionalFeature.SCENE3D);
        prikazRezultata(prozor, podržana);
    }
    public static void main(String[] args) { launch(args); }
```

Test 3D scene (2/2)



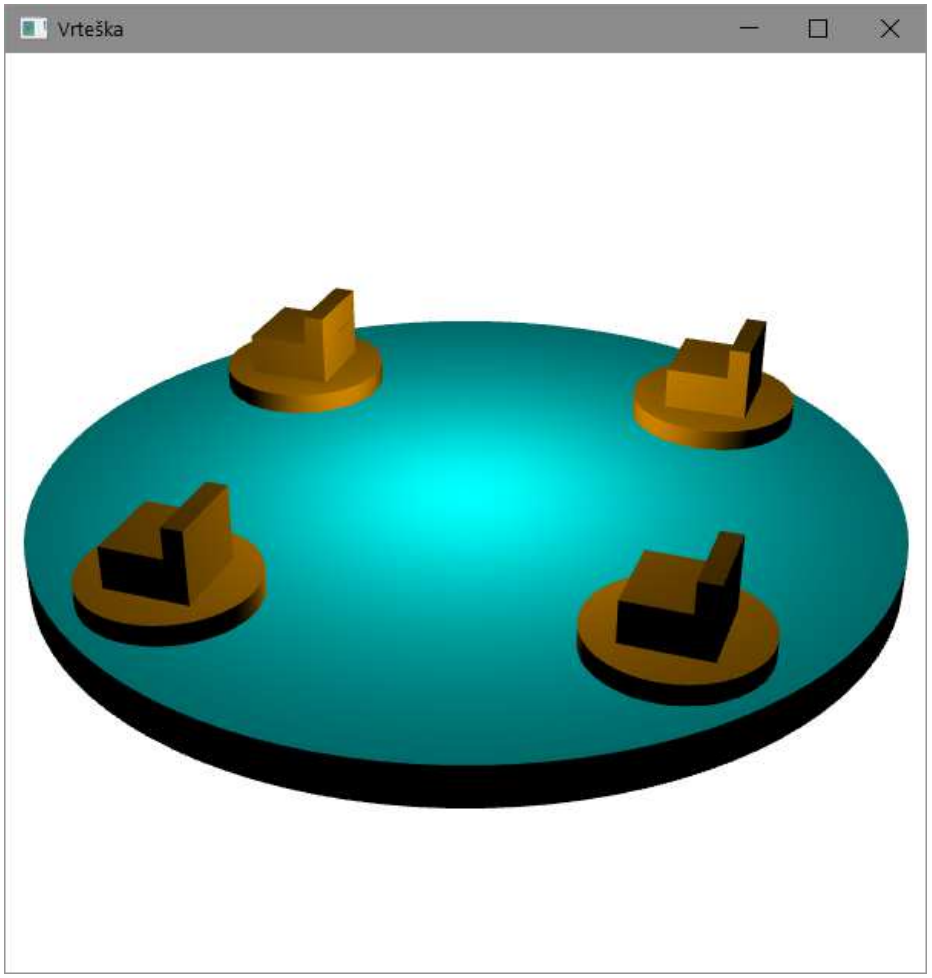
```
    private void prikazRezultata(Stage prozor,
                                 boolean podržana){
       String t;
       if (podržana) t= "3D scena je podržana.";
       else t="3D scena nije podržana.";
       Text tekst = new Text(50, 30t);
       Group koren = new Group();
       koren.getChildren().addAll(tekst);
       Scene scena = new Scene(koren, 300, 50);
       prozor.setTitle("Test 3D scene");
       prozor.setScene(scena);
      prozor.show();
    }

}
```

Zadatak 1: Vrteška (1/2)

Kolokvijum K2 15/16, zadatak prerađen za JavaFX

Napisati program za crtanje i animaciju scene prikazane na slici primenom grafičke biblioteke JavaFX. Scenu čini vrteška sa 4 stolice. Osnova vrteške (podijum) je disk poluprečnika 200, visine 20, koji se okreće oko vertikalne ose u centru, konstantnom brzinom u smeru suprotnom od kretanja kazaljke na časovniku, posmatrano odozgo, tako da pun krug napravi za 10s. Svaka od stolica nalazi se na zasebnom disku (postolju) poluprečnika 40, visine 10, koji se okreću konstantnom brzinom od 1/3 obrtaja u sekundi u smeru kretanja kazaljki na časovniku, posmatrano odozgo. Stolice se crtaju kao dva kvadra. Sedište je kvadratne osnove, dužine stranice 40 i visine 20, dok je naslon visine 20 i debljine 10. Sedište sa postoljem je narandžaste, a podijum tirkizne boje. Belo svetlo je na vertikalnoj osi podijuma, iznad podijuma na visini 100 (merenoj od centra diska podijuma). Koristi se projekcija sa perspektivom. Kameru postaviti približno u položaj koji bi proizveo prikazanu sliku.

Zadatak 1: Vrteška(2/2)



Rešenje: Vrteška (1/8)

```
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.paint.PhongMaterial;
import javafx.scene.paint.Color;
import javafx.scene.PerspectiveCamera;
import javafx.scene.PointLight;
```

Rešenje: Vrteška (2/8)

```
import javafx.scene.transform.Rotate;
import javafx.scene.transform.Translate;
import javafx.animation.Animation;
import javafx.animation.RotateTransition;
import javafx.animation.Interpolator;
import javafx.event.EventHandler;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;
import javafx.util.Duration;

public class Vrteska extends Application {

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

Rešenje: Vrteška (3/8)

```
private Group napraviStolicu(){
    PhongMaterial mat = new PhongMaterial();
    mat.setDiffuseColor(Color.ORANGE);
    Cylinder postolje = new Cylinder(40,10,100);
    postolje.setMaterial(mat);
    Box sediste = new Box(40,20,40);
    sediste.setMaterial(mat);
    sediste.setTranslateY(-15);
    Box naslon = new Box(40,20,10);
    naslon.setMaterial(mat);
    naslon.setTranslateY(-35);
    naslon.setTranslateZ(15);
    Group stolica =new Group();
    stolica.getChildren().addAll(postolje,sediste,naslon);
    return stolica;
}
```

Rešenje: Vrteška (4/8)

```
private RotateTransition napraviRotaciju(Duration vreme,
                                         Group obj, int smer){
   RotateTransition rot = new RotateTransition(vreme, obj);
   rot.setAxis(Rotate.Y_AXIS);
   rot.setToAngle(smer*360);
   rot.setInterpolator(Interpolator.LINEAR);
   rot.setCycleCount(Animation.INDEFINITE);
   return rot;
}
```

Rešenje: Vrteška (5/8)

```
private void napraviVrtećiPodijum(Group g){
   Cylinder podijum = new Cylinder(200,20,100);
   PhongMaterial mat = new PhongMaterial();
   mat.setDiffuseColor(Color.CYAN);
   podijum.setMaterial(mat);
   g.getChildren().addAll(podijum);
   RotateTransition rot=napraviRotaciju(
                          Duration.millis(10000),g,-1);
   rot.play();
}
```

Rešenje: Vrteška (6/8)

```
private void napraviGrupuVrtećihStolica(Group g){
   double u=0;
   for(int i=0; i<=3; i++) {
      Group stolica = napraviStolicu();
      stolica.setTranslateY(-15);
      stolica.setTranslateX(Math.cos(u)*150);
      stolica.setTranslateZ(Math.sin(u)*150);
      g.getChildren().addAll(stolica);
      RotateTransition rot=napraviRotaciju(
                           Duration.millis(3000), stolica, 1);
      rot.play();
      u+=Math.PI/2;
   }
}
```

Rešenje: Vrteška (7/8)

```
private PointLight napraviSvetlo(Group g){
   PointLight svetlo = new PointLight();
   svetlo.setTranslateY(-100);
   svetlo.setColor(Color.WHITE);
   g.getChildren().addAll(svetlo);
   return svetlo;
}

private PerspectiveCamera napraviKameru() {
   PerspectiveCamera kamera = new PerspectiveCamera(true);
   kamera.setFarClip(1000.0);
   Translate t = new Translate(0, 0, -800.0);
   Rotate rX = new Rotate(-30.0, Rotate.X_AXIS);
   kamera.getTransforms().addAll(rX,t);
   return kamera;
}
```

Rešenje: Vrteška (8/8)

```
@Override public void start(Stage prozor) {
   Group vrteška = new Group();
   napraviGrupuVrtećihStolica(vrteška);
   napraviVrtećiPodijum(vrteška);
   PointLight svetlo=napraviSvetlo(vrteška);
   EventHandler<KeyEvent> r = d -> {
      KeyCode k=d.getCode();
      if (k==KeyCode.SPACE) svetlo.setLightOn(!svetlo.isLightOn());
   };
   PerspectiveCamera kamera=napraviKameru();
   SceneAntialiasing glatko=SceneAntialiasing.BALANCED;
   Scene scena = new Scene(vrteška, 600, 600, true, glatko);
   scena.setCamera(kamera);
   scena.addEventHandler(KeyEvent.KEY_PRESSED, r);
   prozor.setTitle("Vrteška");
   prozor.setScene(scena); prozor.show();
} }
```

Zadatak 2: upravljanje kamerom (1/4)

Napisati program koji formira scenu od 3 objekta:
kvadra, valjka i lopte i upravlja kamerom komandama prema sledećoj tabeli:

Ulaz	Efekat	Opis akcije
Mouse Left Button Drag Up	Sphere Forward	Rotiranje nosača (grupe u kojoj je kamera) oko X ose u smeru desne zavojnice
Mouse Left Button Drag Down	Sphere Backward	Rotiranje nosača (grupe u kojoj je kamera) oko X ose u suprotnom smeru desne zavojnice
Mouse Left Button Drag Left	Sphere Left	Rotiranje nosača (grupe u kojoj je kamera) oko Y ose u smeru desne zavojnice
Mouse Left Button Drag Right	Sphere Right	Rotiranje nosača (grupe u kojoj je kamera) oko Y ose u suprotnomsmeru desne zavojnice

Zadatak 2: upravljanje kamerom (2/4)

Ulaz	Efekat	Opis akcije
Mouse Right Button Drag Right	Pan Right	Transliranje kamere u pravcu pozitivne X ose lokalnog koordinatnog sistema
Mouse Right Button Drag Left	Pan Left	Transliranje kamere u pravcu negativne X ose lokalnog koordinatnog sistema
Mouse Right Button Drag Down	Pan Down	Transliranje kamere u pravcu pozitivne Y ose lokalnog koordinatnog sistema
Mouse Right Button Drag Up	Pan Up	Transliranje kamere u pravcu negativne Y ose lokalnog koordinatnog sistema
Mouse Wheel Forward	Zoom In	Transliranje kamere u pravcu pozitivne Z ose lokalnog koordinatnog sistema
Mouse Wheel Backward	Zoom Out	Transliranje kamere u pravcu negativne Z ose lokalnog koordinatnog sistema

Zadatak 2: upravljanje kamerom (3/4)

Ulaz	Efekat	Opis akcije
HOME	Reset	Postavljanje kamere i nosača (grupe u kojoj je kamera) u početnu poziciju i orijentaciju
PG UP	Carrier Roll Right	Rotiranje nosača (grupe u kojoj je kamera) oko Z ose u smeru desne zavojnice
PG DN	Carrier Roll Left	Rotiranje nosača (grupe u kojoj je kamera) oko Z ose u suprotnom smeru desne zavojnice
→	Yaw Right	Rotiranje kamere oko Y ose lokalnog koordinatnog sistema u smeru desne zavojnice
←	Yaw Left	Rotiranje kamere oko Y ose lokalnog koordinatnog sistema u smeru leve zavojnice
↑	Tilt Up	Rotiranje kamere oko X ose lokalnog koordinatnog sistema u smeru desne zavojnice
↓	Tilt Down	Rotiranje kamere oko X ose lokalnog koordinatnog sistema u smeru leve zavojnice

Zadatak 2: upravljanje kamerom (4/4)

Ulaz	Efekat	Opis akcije
+	Roll Right	Rotiranje kamere oko Z ose lokalnog koordinatnog sistema u smeru desne zavojnice
-	Roll Left	Rotiranje kamere oko Z ose lokalnog koordinatnog sistema u smeru leve zavojnice
W	Move Foreward	Translacija nosača duž pozitivne Z ose
S	Move Backward	Translacija nosača duž negativne Z ose
A	Move Left	Translacija nosača duž negativne X ose
D	Move Right	Translacija nosača duž pozitivne X ose
Z	Move Up	Translacija nosača duž negativne Y ose
B	Move Down	Translacija nosača duž pozitivne Y ose

Rešenje: upravljanje kamerom (1/12)

```
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;
import javafx.scene.transform.Translate;
import javafx.scene.input.KeyEvent;
import javafx.scene.input.KeyCode;
import javafx.scene.input.MouseEvent;
import javafx.scene.input.ScrollEvent;
import javafx.event.EventHandler;
```

Rešenje: upravljanje kamerom (2/12)

```
public class KameraUpravljanje extends Application {

    private static final double POCETNO_RASTOJANJE = -1000;

    private static final double BRZINA_TRANS = 1.0;
    private static final double BRZINA_ROT = 1.0;
    private static final double BRZINA_TOCKA = 1.0;
    private static final double CTRL_FAKTOR = 0.1;
    private static final double ALT_FAKTOR = 10.0;

    private static final Translate t = new Translate();
    private static final Rotate rx = new Rotate();
    private static final Rotate ry = new Rotate();
    private static final Rotate rz = new Rotate();

    private static double pozX, pozY;
    private static double staraPozX, staraPozY;
    private static double korakX, korakY, korakZ;
```

Rešenje: upravljanje kamerom (3/12)

```
@Override public void start(Stage prozor) {

    Cylinder valjak = new Cylinder(70,140,20);
    PhongMaterial matV = new PhongMaterial();
    matV.setDiffuseColor(Color.CYAN);
    valjak.setMaterial(matV);

    Box kvadar = new Box(140,140,50);
    PhongMaterial matK = new PhongMaterial();
    matK.setDiffuseColor(Color.BLUEVIOLET);
    kvadar.setMaterial(matK);
    kvadar.setTranslateX(-150);

    Sphere lopta = new Sphere(70,20);
    PhongMaterial matL = new PhongMaterial();
    matL.setDiffuseColor(Color.DARKBLUE);
    lopta.setMaterial(matL);
    lopta.setTranslateX(150);
```

Rešenje: upravljanje kamerom (4/12)

```
PointLight svetlo = new PointLight();
svetlo.setTranslateX(200);
svetlo.setTranslateY(-100);
svetlo.setTranslateZ(-200);

PerspectiveCamera kamera = new PerspectiveCamera(true);
kamera.setNearClip(0.1);
kamera.setFarClip(5000.0);
kamera.setTranslateZ(POCETNO_RASTOJANJE);

Group nosač = new Group(kamera);
rx.setAxis(Rotate.X_AXIS);
ry.setAxis(Rotate.Y_AXIS);
rz.setAxis(Rotate.Z_AXIS);
rx.setAngle(0);
ry.setAngle(0);
rz.setAngle(0);
nosač.getTransforms().addAll(t, rz, ry, rx);
```

Rešenje: upravljanje kamerom (5/12)

```
    Group koren = new Group(valjak, kvadar, lopta, svetlo, nosač);
    SceneAntialiasing glatko=SceneAntialiasing.BALANCED;
    Scene scena = new Scene(koren, 600, 600, true, glatko);
    scena.setCamera(kamera);
    obradaDogađaja(scena, kamera);
    prozor.setTitle("Kamera-upravljanje");
    prozor.setScene(scena);
    prozor.show();
}

private void obradaDogađaja (Scene scena, PerspectiveCamera kamera) {

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

Rešenje: upravljanje kamerom (6/12)

```
EventHandler<MouseEvent> r2 = dog -> {// obrada događaja vučenja miša
    staraPozX = pozX;                staraPozY = pozY;
    pozX = dog.getSceneX();          pozY = dog.getSceneY();
    korakX = (pozX - staraPozX);     korakY = (pozY - staraPozY);
    double modifikator = 1.0;
    if (dog.isControlDown()) modifikator = CTRL_FAKTOR;
    if (dog.isAltDown()) modifikator = ALT_FAKTOR;
    if (dog.isPrimaryButtonDown()) {
        ry.setAngle(ry.getAngle() - korakX*BRZINA_ROT*modifikator);
        rx.setAngle(rx.getAngle() + korakY*BRZINA_ROT*modifikator);
    }
    else if (dog.isSecondaryButtonDown()) {
        double x = kamera.getTranslateX();
        kamera.setTranslateX(x + korakX*BRZINA_TRANS*modifikator);
        double y = kamera.getTranslateY();
        kamera.setTranslateY(y + korakY*BRZINA_TRANS*modifikator);
    }
}; // obrada događaja vučenja miša
scena.addEventHandler(MouseEvent.MOUSE_DRAGGED, r2);
```

Rešenje: upravljanje kamerom (7/12)

```
EventHandler<ScrollEvent> r3 = dog -> {// obrada dogadjaja točkića miša
    double modifikator = 1.0;
    korakZ = dog.getDeltaY();

    if (dog.isControlDown()) modifikator = CTRL_FAKTOR;
    if (dog.isAltDown()) modifikator = ALT_FAKTOR;

    double z = kamera.getTranslateZ();
    kamera.setTranslateZ(z + korakZ*BRZINA_TOCKA*modifikator);
};  // obrada dogadjaja točkića miša
scena.addEventHandler(ScrollEvent.SCROLL, r3);

EventHandler<KeyEvent> r4 = dog -> {// obrada događaja tastature
    KeyCode k=dog.getCode();
    double modifikator = 1.0;
    if (dog.isControlDown()) modifikator = CTRL_FAKTOR;
    if (dog.isAltDown()) modifikator = ALT_FAKTOR;
```

Rešenje: upravljanje kamerom (8/12)

```
switch (k) {
    case HOME:
        kamera.setTranslateZ(POCETNO_RASTOJANJE);
        kamera.setTranslateX(0);
        kamera.setTranslateY(0);
        kamera.setRotationAxis(Rotate.X_AXIS);
        kamera.setRotate(0);
        kamera.setRotationAxis(Rotate.Y_AXIS);
        kamera.setRotate(0);

        kamera.setRotationAxis(Rotate.Z_AXIS);
        kamera.setRotate(0);
        t.setX(0.0); t.setY(0.0); t.setZ(0.0);
        rx.setAngle(0); ry.setAngle(0);  rz.setAngle(0); break;
```

Rešenje: upravljanje kamerom (9/12)

```
case UP:
    if (kamera.getRotationAxis() != Rotate.X_AXIS) {
        kamera.setRotationAxis(Rotate.X_AXIS);
        kamera.setRotate(0);
    }
    else kamera.setRotationAxis(Rotate.X_AXIS);
    kamera.setRotate(kamera.getRotate() + modifikator); break;
case DOWN:
    if (kamera.getRotationAxis() != Rotate.X_AXIS) {
        kamera.setRotationAxis(Rotate.X_AXIS);
        kamera.setRotate(0);
    }
    else kamera.setRotationAxis(Rotate.X_AXIS);
    kamera.setRotate(kamera.getRotate() - modifikator); break;
```

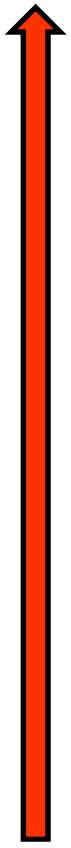
Rešenje: upravljanje kamerom (10/12)

```
case RIGHT:
    if (kamera.getRotationAxis() != Rotate.Y_AXIS) {
        kamera.setRotationAxis(Rotate.Y_AXIS);
        kamera.setRotate(0);
    }
    else kamera.setRotationAxis(Rotate.Y_AXIS);
    kamera.setRotate(kamera.getRotate() + modifikator); break;
case LEFT:
    if (kamera.getRotationAxis() != Rotate.Y_AXIS) {
        kamera.setRotationAxis(Rotate.Y_AXIS);
        kamera.setRotate(0);
    }
    else kamera.setRotationAxis(Rotate.Y_AXIS);
    kamera.setRotate(kamera.getRotate() - modifikator); break;
```

Rešenje: upravljanje kamerom (11/12)

```
case ADD:
    if (kamera.getRotationAxis() != Rotate.Z_AXIS) {
        kamera.setRotationAxis(Rotate.Z_AXIS);
        kamera.setRotate(0);
    }
    else kamera.setRotationAxis(Rotate.Z_AXIS);
    kamera.setRotate(kamera.getRotate() + modifikator); break;
case SUBTRACT:
    if (kamera.getRotationAxis() != Rotate.Z_AXIS) {
        kamera.setRotationAxis(Rotate.Z_AXIS);
        kamera.setRotate(0);
    }
    else kamera.setRotationAxis(Rotate.Z_AXIS);
    kamera.setRotate(kamera.getRotate() - modifikator); break;
```

Rešenje: upravljanje kamerom (12/12)



```
                case PAGE_UP: rz.setAngle(rz.getAngle()+modifikator); break;
                case PAGE_DOWN: rz.setAngle(rz.getAngle()-modifikator); break;

                case W: t.setZ(t.getZ() + BRZINA_TRANS*modifikator); break;
                case S: t.setZ(t.getZ() - BRZINA_TRANS*modifikator); break;
                case D: t.setX(t.getX() + BRZINA_TRANS*modifikator); break;
                case A: t.setX(t.getX() - BRZINA_TRANS*modifikator); break;
                case B: t.setY(t.getY() + BRZINA_TRANS*modifikator); break;
                case Z: t.setY(t.getY() - BRZINA_TRANS*modifikator); break;
            }
        }; // obrada događaja tastature
          scena.addEventHandler(KeyEvent.KEY_PRESSED, r4);
    }

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