Лабораторна робота №6

Тема: Анімація тривимірних об’єктів

Мета: Навчитися анімувати складні об’єкти тривимірної сцени.

Вимоги до засобів розробки програми

Мова написання – java.

Додаткові бібліотеки – java3d (інструкцію по встановленню див. в лабораторній роботі №4).

Теоретичні відомості

Java3D використовується для візуалізації тривимірних сцен.

Граф сцени (Scene graph) – розташування тривимірних об’єктів у деревовидній структурі, яке повністю визначає склад віртуального всесвіту, а також те, як всесвіт повинен бути відображений.

У будь-якого графа сцени єдиний Віртуальний Всесвіт(VirtualUniverse).

Крім геометрії об’єкту(вершин, ребер, нормалей), також можна змінювати його зовнішній вигляд. За зовнішній вигляд об’єкту відповідає клас Appearance.

За анімацію та взаємодію об’єктів сцени відповідають потомки класу Behavior.

В Java3D можливі такі геометричні перетворення як поворот, масштабування та переміщення. На основі цих перетворень і створюється анімація. Для взаємодії з об’єктами можуть використовуватися клавіатура, миша і т.д..

Для детального ознайомлення з можливостями Java3D зверніться до літературних джерел (див. пункт Література).

Приклади створення тривимірної сцени з анімацією

Опишемо програму для анімації хвилинної та годинної стрілок будильника.

1. Описуємо вікно програми та сцену, що буде видима спостерігачу

1.1. Описуємо сцену програми та позицію спостерігача

```
public Canvas3D myCanvas3D;
myCanvas3D = new Canvas3D(SimpleUniverse.getPreferredConfiguration());
        SimpleUniverse simpUniv = new SimpleUniverse(myCanvas3D);
simpUniv.getViewingPlatform().setNominalViewingTransform();
createSceneGraph(simpUniv);
```

1.2. Додаємо світло до сцени

```
BranchGroup bgLight = new BranchGroup();

BoundingSphere bounds = new BoundingSphere(new Point3d(0.0,0.0,0.0), 100.0);
Color3f lightColour1 = new Color3f(1.0f,1.0f,1.0f);
Vector3f lightDir1  = new Vector3f(-1.0f,0.0f,-0.5f);
DirectionalLight light1 = new DirectionalLight(lightColour1, lightDir1);
light1.setInfluencingBounds(bounds);
bgLight.addChild(light1);
simpUniv.addBranchGraph(bgLight);
```

1.3. Для навігації по сцені за допомогою миші, треба написати наступне

```
OrbitBehavior ob = new OrbitBehavior(myCanvas3D);
ob.setSchedulingBounds(new BoundingSphere(new Point3d(0.0,0.0,0.0),Double.MAX_VALUE));
simpUniv.getViewingPlatform().setViewPlatformBehavior(ob);
```

1.4. Закінчуємо опис вікна програми, вказавши його заголовок, розмір, позицію сцени у вікні та зробивши його видимим.

```
setTitle("Graphics Lab 6");
setSize(700,700);
getContentPane().add("Center", myCanvas3D);
setVisible(true);
```

2. Завантажуємо об’єкт до сцени (завантажуємо будильник з файлу alarm_clock.obj)

```
ObjectFile f = new ObjectFile(ObjectFile.RESIZE);
Scene s = null;
try
{
        s = f.load("alarm_clock.obj");
}
catch (Exception e)
{
```

```
            System.out.println("File loading failed:" + e);
}
```

2.1. В файлі з розширенням .obj зберігається геометрія сцени. Розглянемо фрагмент файлу:

```
#
# object minute_arrow
#

v  6.5112 20.3101 -0.0459
…
v  6.1467 20.2129 0.0497
# 120 vertices
```

Як видно після символу # вказана назва об’єкту, геометрія якого описується далі. Для анімації лише частини сцени необхідно знати назву того об’єкту, що буде рухатися. Як вже зрозуміло, її можна знайти у файлі .obj (але в такому випадку не можна бути впевненим, де саме розташований цей об’єкт – через складність описання геометрії) або переглянути його за допомогою графічного редактора тривимірних моделей (наприклад, 3ds Max).

2.2. Для виведення у консоль назв всіх об’єктів у java програмі можна скористатися наступним кодом:

```
Hashtable namedObjects = s.getNamedObjects();
Hashtable namedObjectsT = table.getNamedObjects();
 Enumeration enumer = namedObjectsT.keys();
 String name;
while (enumer.hasMoreElements())
{
            name = (String) enumer.nextElement();
            System.out.println("Name: "+name);
 }
```

2.3. Виділимо зі сцени хвилинну та годинну стрілки (у редакторі вони названі як minute_arrow та hour_arrow).

```
Shape3D minute_arrow = (Shape3D) namedObjects.get("minute_arrow");
Shape3D hour_arrow = (Shape3D) namedObjects.get("hour_arrow");
```

2.4. Всі інші об’єкти додаємо у сцену.

Об’єкти додаються у групу TransformGroup, для якої визначаються трансформації Transform3D (повернення сцени на кут або рух).

```
Transform3D tfObject = new Transform3D();
tfObject.rotY(Math.PI/3); //повертаємо будильник відносно осі y
tfObject.setScale(1.0/6);  //зменшуємо зображення в 6 разів
tfObject.setTranslation(new Vector3d(0.5f,0.0f,-0.4f)); // переміщуємо об’єкт по осі x та осі z
TransformGroup tgObject = new TransformGroup(tfObject);

//додаємо об’єкти в сцену повним їх перебором
Shape3D[] clock = new Shape3D[] {(Shape3D) namedObjects.get("sphere02"), clock_obod, (Shape3D)
namedObjects.get("hammer"),(Shape3D) namedObjects.get("alarm_arrow"),(Shape3D)
namedObjects.get("twelve"),(Shape3D) namedObjects.get("three"),(Shape3D)
namedObjects.get("six"),(Shape3D) namedObjects.get("nine"),(Shape3D)
namedObjects.get("cylinder07"),(Shape3D) namedObjects.get("cylinder06"),(Shape3D)
namedObjects.get("rectangle03"), dial, (Shape3D) namedObjects.get("sphere03")};
 for (Shape3D shape:clock){
        tgObject.addChild(shape.cloneTree());
 }
//можна модифікувати рядки коду таким чином, щоб не створювати додатковий масив, -  додати всі
об’єкти, імена яких не minute_arrow та hour_arrow
```

2.5. Зробимо циферблат будильника білим, а його корпус – помаранчевим.

Як було раніше вказано(див. Теоретичні відомості), за зовнішній вигляд відповідає клас Appearance.

```
Appearance whiteApp = new Appearance();
setToMyDefaultAppearance(whiteApp,new Color3f(1.0f,1.0f,1.0f));
Shape3D dial = (Shape3D) namedObjects.get("dial"); // циферблат
dial.setAppearance(whiteApp);
Appearance lightRedApp = new Appearance();
setToMyDefaultAppearance(lightRedApp,new Color3f(0.8f,0.1f,0.0f));
Shape3D clock_obod = (Shape3D) namedObjects.get("cylinder05"); // корпус
clock_obod.setAppearance(lightRedApp);

//метод для встановлення матеріалу відповідного кольору
public static void setToMyDefaultAppearance(Appearance app, Color3f col)
{
        app.setMaterial(new Material(col,col,col,col,150.0f));
 }
```

3. Анімація стрілок

Рух стрілки – це поворот навколо одного її кінця на 180^0. При чому, до того часу, коли годинна стрілка зробить 1 оберт, то хвилинна вже зробить 12.

Для здійснення руху об’єкту необхідно:

1. Додати його у групу трансформації (TransformGroup).

2. Визначити необхідні перетворення для здійснення руху потрібним чином (у Transform3D). За замовчанням поворот відбувається навколо осі x.
3. Задати час руху (за допомогою об’єкту класу Alpha).
4. Визначити рух за допомогою потомку класу Interpolator (обертання – клас RotationInterpolator).

   Опишемо рух хвилинної стрілки.

```
TransformGroup tgmMinArrow = new TransformGroup();
   tgmMinArrow.addChild(minute_arrow.cloneTree());//додаємо хвилинну стрілку до групи трансформації
Transform3D minArrowRotationAxis = new Transform3D();
minArrowRotationAxis.rotZ(Math.PI/2); //вказуємо, що рух буде проходити навколо осі Z

int timeStart = 2000; //обертання стрілки відбудеться через 2 секунди після початку виконання програми
int noStartRotationsMin = 24;  //всього рух відбудеться 24 рази (тобто хвилинна стрілка відрахує 24 години)
int timeRotationMin = 3000;//1 оберт займе 3 секунди

Alpha minRotationAlpha = new Alpha(noStartRotationsMin,
                    Alpha.INCREASING_ENABLE,
                    timeStart,0,timeRotationMin,0,0,0,0,0);

RotationInterpolator minArrRotation = new RotationInterpolator(
                    minRotationAlpha, tgmMinArrow,
                    minArrowRotationAxis,(float) Math.PI*2,0.0f); //опис руху стрілки

minArrRotation.setSchedulingBounds(bounds);
//дозволяємо трансформації об’єкту у групі
tgmMinArrow.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
tgmMinArrow.addChild(minArrRotation);
```

Зрозуміло, що рух годинної стрілки описується так само, але вона повинна за той же час зробити 2 оберти, один оберт повинен займати 36 секунд.

4. Для більшої реалістичності додамо до сцени стіл та накладемо на нього текстуру.

```
table = f.load("table.obj");//завантажуємо модель стола
TextureLoader textureLoad = new TextureLoader("table.jpg",null);// завантажуємо зображення текстури
ImageComponent2D textureIm = textureLoad.getImage();
Texture2D aTexture = new Texture2D(Texture2D.BASE_LEVEL,Texture2D.RGB,
```

```
                                    textureIm.getWidth(),
                                    textureIm.getHeight());//створюємо текстуру
        aTexture.setImage(0,textureIm);//додаємо в текстуру зображення

        //визначаємо зовнішній вигляд текстури
        Appearance textureApp = new Appearance();
        textureApp.setTexture(aTexture);
        TextureAttributes textureAttr = new TextureAttributes();
        textureAttr.setTextureMode(TextureAttributes.REPLACE);
        textureApp.setTextureAttributes(textureAttr);
        //визначаємо характеристики матеріалу
        Material mat = new Material();
        mat.setShininess(120.0f);
        mat.setSpecularColor(new Color3f(0.7f,0.2f,0.1f));
        textureApp.setMaterial(mat);

        Shape3D tableObj = (Shape3D) namedObjectsT.get("table");//виокремлюємо стіл зі сцени
        tableObj.setAppearance(textureApp); //накладаємо текстуру на об'єкт
```

5. Відображення сцени у вікні програми

```
        BranchGroup theScene = new BranchGroup();
        theScene.addChild(tgArrows);//додаємо стрілки у вікно сцени
        theScene.addChild(tgObject);//додаємо інші частини будильнику
        theScene.addChild(tgObjectTable);//додаємо стіл
        //встановлюємо фонове зображення
        Background bg = new Background(new Color3f(1.0f,1.0f,1.0f));
        bg.setApplicationBounds(bounds);
        theScene.addChild(bg);

        theScene.compile();
        su.addBranchGraph(theScene);//додавання сцени у віртуальний всесвіт
```

Повний текст програми:

```
import javax.vecmath.*;

import com.sun.j3d.utils.universe.*;
import javax.media.j3d.*;
import com.sun.j3d.utils.behaviors.vp.*;
import com.sun.j3d.utils.image.TextureLoader;
import javax.swing.JFrame;
import com.sun.j3d.loaders.*;
import com.sun.j3d.loaders.objectfile.*;
import java.util.Hashtable;
import java.util.Enumeration;
public class AlarmAnimation extends JFrame
{

 public Canvas3D myCanvas3D;


 public AlarmAnimation()
 {

  //механізм для закриття вікна та виходу з програми
  this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  //параметри перегляду сцени за замовчанням
```

```
 myCanvas3D = new Canvas3D(SimpleUniverse.getPreferredConfiguration());
 //створення SimpleUniverse (віртуального всесвіту)
 SimpleUniverse simpUniv = new SimpleUniverse(myCanvas3D);
 //положення глядача за замовчанням
 simpUniv.getViewingPlatform().setNominalViewingTransform();
 //створення сцени
 createSceneGraph(simpUniv);
 //додання світла у сцену
 addLight(simpUniv);

 //наступні рядки дозволяють навігацію по сцені за допомогою миші
 OrbitBehavior ob = new OrbitBehavior(myCanvas3D);
 ob.setSchedulingBounds(new BoundingSphere(new Point3d(0.0,0.0,0.0),Double.MAX_VALUE));
 simpUniv.getViewingPlatform().setViewPlatformBehavior(ob);
 //параметри вікна програми
 setTitle("Alarm clock");
 setSize(700,700);
 getContentPane().add("Center", myCanvas3D);
 setVisible(true);
}

public static void main(String[] args)
{
          AlarmAnimation alarmAnimation = new AlarmAnimation();
}

//в цьому методі створюються об'єкти та додаються до сцени
public void createSceneGraph(SimpleUniverse su)
{
         // завантаження файлу .obj.
 ObjectFile f = new ObjectFile(ObjectFile.RESIZE);
 Scene alarmScene = null;
 Scene tableScene = null;
 try
 {
         alarmScene = f.load("alarm_clock.obj");
         tableScene = f.load("table.obj");
 }
 catch (Exception e)
 {
  System.out.println("File loading failed:" + e);
 }

 //створення трансформаційної групи для завнтаженого будильника
 Transform3D tfAlarm = new Transform3D();
 tfAlarm.rotZ(0);
 tfAlarm.rotY(Math.PI/3);
 tfAlarm.setScale(1.0/6);
 tfAlarm.setTranslation(new Vector3d(0.5f,0.0f,-0.4f));
 TransformGroup tgAlarm = new TransformGroup(tfAlarm);
 //створення трансформаційної групи для завнтаженого стола
 Transform3D tfObjectTable = new Transform3D();
 tfObjectTable.setTranslation(new Vector3d(0.0f,-0.61f,0.0f));
 TransformGroup tgObjectTable = new TransformGroup(tfObjectTable);
 tgObjectTable.addChild(tableScene.getSceneGroup());

 //отримання імен об'єктів, з яких складаються будильник та стіл
 //та вивід в консоль назв об'єктів, з яких складається стіл
```

```
  Hashtable alarmNamedObjects = alarmScene.getNamedObjects();
  Hashtable tableNamedObjects = tableScene.getNamedObjects();
  Enumeration enumer = tableNamedObjects.keys();
  String name;
  while (enumer.hasMoreElements())
  {
   name = (String) enumer.nextElement();
   System.out.println("Name: "+name);
  }

  //завантаження зображення для ткстури
  TextureLoader textureLoad = new TextureLoader("table.jpg",null);
  ImageComponent2D textureIm = textureLoad.getImage();
  //створення текстури
  Texture2D woodTexture = new Texture2D(Texture2D.BASE_LEVEL,Texture2D.RGB,
                          textureIm.getWidth(),
                          textureIm.getHeight());
  woodTexture.setImage(0,textureIm);

  //створення матеріалу з текстурою
  Appearance textureApp = new Appearance();
  textureApp.setTexture(woodTexture);
  TextureAttributes textureAttr = new TextureAttributes();
  textureAttr.setTextureMode(TextureAttributes.REPLACE);
  textureApp.setTextureAttributes(textureAttr);
  Material mat = new Material();
  mat.setShininess(120.0f);
  mat.setSpecularColor(new Color3f(0.7f,0.2f,0.1f));
  textureApp.setMaterial(mat);
  //накладання матеріалу столу
  Shape3D tableObj = (Shape3D) tableNamedObjects.get("table");
  tableObj.setAppearance(textureApp);

  //пофарбуємо циферблат у білий колір
  Appearance whiteApp = new Appearance();
  setToMyDefaultAppearance(whiteApp,new Color3f(1.0f,1.0f,1.0f));
  Shape3D dial = (Shape3D) alarmNamedObjects.get("dial");
  dial.setAppearance(whiteApp);
  //пофарбуємо корпус будильника у помаранчевий колір
  Appearance lightRedApp = new Appearance();
  setToMyDefaultAppearance(lightRedApp,new Color3f(0.8f,0.1f,0.0f));
  Shape3D clock_obod = (Shape3D) alarmNamedObjects.get("cylinder05");
  clock_obod.setAppearance(lightRedApp);

  Shape3D minute_arrow = (Shape3D) alarmNamedObjects.get("minute_arrow");
  Shape3D hour_arrow = (Shape3D) alarmNamedObjects.get("hour_arrow");
  //додавання всіх об'єктів будильника, крім хвилинної та годинної стрілки, у сцену
  Shape3D[] clock = new Shape3D[] {(Shape3D) alarmNamedObjects.get("sphere02"), clock_obod,
                  (Shape3D) alarmNamedObjects.get("hammer"),(Shape3D)
alarmNamedObjects.get("alarm_arrow"),
                  (Shape3D) alarmNamedObjects.get("twelve"),(Shape3D) alarmNamedObjects.get("three"),
                  (Shape3D) alarmNamedObjects.get("six"),(Shape3D) alarmNamedObjects.get("nine"),
                  (Shape3D) alarmNamedObjects.get("cylinder07"),(Shape3D)
alarmNamedObjects.get("cylinder06"),
                  (Shape3D) alarmNamedObjects.get("rectangle03"), dial, (Shape3D)
alarmNamedObjects.get("sphere03")};
  for (Shape3D shape:clock){
        tgAlarm.addChild(shape.cloneTree());
```

```
    }

    //група трансформації годинної стрілки
    TransformGroup tgmHourArrow = new TransformGroup();
    tgmHourArrow.addChild(hour_arrow.cloneTree());

    //повернемо площину обертання на 90 градусів відносно осі Z
    Transform3D hourArrowRotationAxis = new Transform3D();
    hourArrowRotationAxis.rotZ(Math.PI/2);

    int timeStart = 2000; //стрілка почне рух через 2 секунди після запуску програми
    int noRotationsHour = 2;  //кількість обертів
    int timeRotationHour = 36000;//час одного оберту

    //Alpha для руху годинної стрілки
    Alpha hourRotationAlpha = new Alpha(noRotationsHour,
                                Alpha.INCREASING_ENABLE,
                                timeStart,
                                0,timeRotationHour,0,0,0,0,0);
    //обертання годинної стрілки
    RotationInterpolator hourArrRotation = new RotationInterpolator(

hourRotationAlpha,tgmHourArrow,

hourArrowRotationAxis,(float) Math.PI*2,0.0f);

    BoundingSphere bounds = new BoundingSphere(new Point3d(0.0,0.0,0.0),Double.MAX_VALUE);
    hourArrRotation.setSchedulingBounds(bounds);
    tgmHourArrow.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
    tgmHourArrow.addChild(hourArrRotation);

   //група трансформації хвилинної стрілки
    TransformGroup tgmMinArrow = new TransformGroup();
    tgmMinArrow.addChild(minute_arrow.cloneTree());
   //повернемо площину обертання на 90 градусів відносно осі Z
    Transform3D minArrowRotationAxis = new Transform3D();
    minArrowRotationAxis.rotZ(Math.PI/2);

          int noStartRotationsMin = 24;  //всього рух відбудеться 24 рази
    int timeRotationMin = 3000;//1 оберт займе 3 секунди
    Alpha minRotationAlpha = new Alpha(noStartRotationsMin,
                            Alpha.INCREASING_ENABLE,
                            timeStart,0,timeRotationMin,0,0,0,0,0);
    //обертання хвилинної стрілки
    RotationInterpolator minArrRotation = new RotationInterpolator(
                                minRotationAlpha, tgmMinArrow,
                                minArrowRotationAxis,(float) Math.PI*2,0.0f);

    minArrRotation.setSchedulingBounds(bounds);
    tgmMinArrow.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
    tgmMinArrow.addChild(minArrRotation);

    //розміщуємо будильник
    Transform3D tfRotor = new Transform3D();
    tfRotor.rotY(Math.PI/3);//розміщуємо у напівоберті до користувача
    tfRotor.setScale(1.0/6);//зменшуємо будильник у 6 разів
    tfRotor.setTranslation(new Vector3d(0.5f,0.0f,-0.4f));//переміщуємо
    TransformGroup tgAlarmArrows = new TransformGroup(tfRotor);
```

```
  tgAlarmArrows.addChild(tgmHourArrow);
  tgAlarmArrows.addChild(tgmMinArrow);
  //створення сцени
  BranchGroup theScene = new BranchGroup();
  theScene.addChild(tgAlarmArrows);//додаємо стрілки
  theScene.addChild(tgAlarm);//додаємо будильник
  theScene.addChild(tgObjectTable);//додаємо стіл
  //створюємо білий фон
  Background bg = new Background(new Color3f(1.0f,1.0f,1.0f));
  bg.setApplicationBounds(bounds);
  theScene.addChild(bg);

  theScene.compile();

  //додаємо сцену до віртуального всесвіту
  su.addBranchGraph(theScene);
 }
 //метод для генерації зовнішньої поверхні
 public static void setToMyDefaultAppearance(Appearance app, Color3f col)
 {
  app.setMaterial(new Material(col,col,col,col,150.0f));
 }
 //метод для додавання освітлення
 public void addLight(SimpleUniverse su)
 {

  BranchGroup bgLight = new BranchGroup();

  BoundingSphere bounds = new BoundingSphere(new Point3d(0.0,0.0,0.0), 100.0);
  Color3f lightColour1 = new Color3f(1.0f,1.0f,1.0f);
  Vector3f lightDir1  = new Vector3f(-1.0f,0.0f,-0.5f);
  DirectionalLight light1 = new DirectionalLight(lightColour1, lightDir1);
  light1.setInfluencingBounds(bounds);

  bgLight.addChild(light1);
  su.addBranchGraph(bgLight);
 }
}
```

Результат роботи програми:

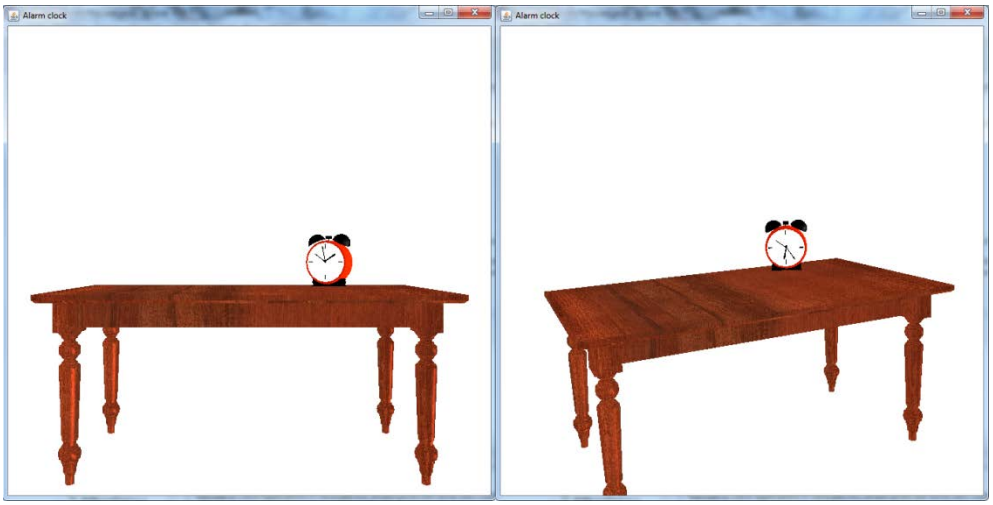

Рис.1. Результат роботи програми для анімації будильника

У вищеописаній програмі рух стрілок відбувався незалежно від дій користувача через 2 секунди після завантаження програми.

Опишемо приклад анімації об’єкту після вибору його користувачем.

В якості складного об’єкту оберемо рояль. Нехай після вибору користувачем кришки (над клавіатурою та над струнами) роялю зачиняються чи відчиняються. Пропустимо опис завантаження моделі, виставлення світла, створення вікна програми, виокремлення з складного об’єкту великої та малої кришки (аналогічно до попереднього прикладу).

1. Визначимо рух малої кришки роялю.

```
TransformGroup smallTransXformGroup = translate(
                small_cover,
                new Vector3f(0.1f,0.0f,0.7f));
Alpha small_cover_alpha = new Alpha(1,2000);
small_cover_alpha.setStartTime(Long.MAX_VALUE); //рух не почнеться, поки користувач не вибрав кришку

float startAngle1 = 0.0f;
float endAngle1 =(float)Math.PI/4;

 //закриття кришки – поворот її від 0 до 45 градусів
TransformGroup smallRotXformGroup =
     rotate(smallTransXformGroup,small_cover_alpha,startAngle1,endAngle1);

Alpha small_cover_alpha1 = new Alpha(1,2000);
small_cover_alpha1.setStartTime(Long.MAX_VALUE);

//відкриття кришки – поворот її від 0 до -45 градусів
TransformGroup smallRotXformGroup1 =
          rotate(smallRotXformGroup,small_cover_alpha1,startAngle1,-endAngle1);

 Transform3D coverObject = new Transform3D();
 coverObject.rotY(5*Math.PI/6); //початкова позиція складного об’єкту, роль в напівоберті до глядача
TransformGroup tgCoverObject = new TransformGroup(coverObject);
 tgCoverObject.addChild(smallRotXformGroup1);
TransformGroup tgCoverSmall = translate(tgCoverObject, new Vector3f(-0.312f,-0.023f,0.75f));
```

Аналогічно описується рух великої кришки.

2. Для здійснення руху після вибору об’єкта мишею створимо клас, що буде реагувати на позицію миші.

```
//класу передаються сцена, її границі, а також масив змінних часу(alphas), які визначають початок руху
PickingObjects po = new PickingObjects(myCanvas3D,theScene,bounds,alphas);
// додавання об’єкту класу у сцену
```

```
theScene.addChild(po);
//фрагмент описання класу для вибору
public class PickingObjects extends PickMouseBehavior
{ …
public void updateScene(int xpos, int ypos)
 {
Shape3D pickedShape = null;
      pickCanvas.setShapeLocation(xpos,ypos);
      PickResult pResult = pickCanvas.pickClosest();//вибір найближчого до миші об'єкта
      if (pResult != null)
      {
                pickedShape = (Shape3D) pResult.getNode(PickResult.SHAPE3D);
       }

      if (pickedShape != null) // якщо обрано об'єкт
      {
                if (pickedShape.getUserData()=="cover_small")//якщо обрано малу кришку
                {
                    if (move1==0){
                         //якщо кришка ще не закривалась – закриваємо її, рух почнеться при виборі об'єкта
                         alphas[0].setStartTime(System.currentTimeMillis()-alphas[0].getTriggerTime());
                         System.out.println("The object "+pickedShape.getUserData()+" has been selected.");
                         move1++;
                     }
                else{
                     if (move1 >0 ){
                         //якщо кришка закривалась – відкриваємо її, рух почнеться при виборі об'єкта
                         alphas[1].setStartTime(System.currentTimeMillis()-alphas[1].getTriggerTime());
                         System.out.println("The object "+pickedShape.getUserData()+" has been selected.");
                         move1=-1;
       }}
          }
      ….
      }
```

Повний текст програми:

```
import javax.vecmath.*;
import com.sun.j3d.utils.universe.*;
import javax.media.j3d.*;
import com.sun.j3d.utils.behaviors.vp.*;
import com.sun.j3d.utils.image.TextureLoader;
import javax.swing.JFrame;
import com.sun.j3d.loaders.*;
import com.sun.j3d.loaders.objectfile.*;
import java.util.Hashtable;
import java.util.Enumeration;

public class PianoInteraction extends JFrame
{
 public Canvas3D myCanvas3D;


 public PianoInteraction()
 {

        //механізм для закриття вікна та виходу з програми
          this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          //параметри перегляду сцени за замовчанням
```

```
            myCanvas3D = new Canvas3D(SimpleUniverse.getPreferredConfiguration());
            //створення SimpleUniverse (віртуального всесвіту)
            SimpleUniverse simpUniv = new SimpleUniverse(myCanvas3D);
            //положення глядача за замовчанням
            simpUniv.getViewingPlatform().setNominalViewingTransform();
            //створення сцени
            createSceneGraph(simpUniv);
            //додання світла у сцену
            addLight(simpUniv);
            //параметри вікна програми
            setTitle("Piano interaction");
            setSize(700,700);
            getContentPane().add("Center", myCanvas3D);
            setVisible(true);
}

public static void main(String[] args)
{
            PianoInteraction pianoInteraction = new PianoInteraction();
}

//в цьому методі створюються об'єкти та додаються до сцени
public void createSceneGraph(SimpleUniverse su)
{
          // завантаження файлу .obj.
 ObjectFile f = new ObjectFile(ObjectFile.RESIZE);
 Scene pianoScene = null;
 try
 {
          pianoScene = f.load("piano.obj");
 }
 catch (Exception e)
 {
  System.out.println("File loading failed:" + e);
 }

 //створення групи трансформації для роялю
 Transform3D tfPiano = new Transform3D();
 tfPiano.rotY(5*Math.PI/6);
 TransformGroup tgPiano = new TransformGroup(tfPiano);

 //отримання імен об'єктів, з яких складається рояль
 Hashtable pianoNamedObjects = pianoScene.getNamedObjects();

 Enumeration enumer = pianoNamedObjects.keys();
 String name;
 //створення білого матеріалу - для білих клавіш
 Appearance whiteApp = new Appearance();
 setToMyDefaultAppearance(whiteApp,new Color3f(1.0f,1.0f,1.0f));
 //створення чорного матеріалу - для чорних клавіш
 Appearance blackApp = new Appearance();
 setToMyDefaultAppearance(blackApp,new Color3f(0.0f,0.0f,0.0f));
 //створення сірого матеріалу - для кришок роялю
 Appearance greyApp = new Appearance();
 setToMyDefaultAppearance(greyApp,new Color3f(0.1f,0.1f,0.1f));
 //додавання об'єктів у сцену
 int keyCount = 0;//номер клавіші у хроматичній гамі з 12 клавіш (від білої фа до білої сі)
 int keyNumber = 0; //номер клавіші роялю
```

```
while (enumer.hasMoreElements())
{
  name = (String) enumer.nextElement();

  if (name.substring(0, 3).equals("key")){ //якщо об'єкт клавіша
        Shape3D keyOriginal = (Shape3D) pianoNamedObjects.get(name);
        //додаємо її у сцену
        Shape3D key = (Shape3D)keyOriginal.cloneNode(true);
        key.setPickable(false);//клавіші недоступні для вибору мишею
        tgPiano.addChild(key);
        //визначаємо номер клавіші на роялі
        keyNumber = Integer.parseInt(name.substring(3, name.length()));
        //визначаємо номер клавіші у хроматичній гамі
        keyCount = keyNumber%12;
        //розфарбовуємо клавіші
        switch ( keyCount){
                case 0: {
                        key.setAppearance(whiteApp);
                        break;
                }
                case 1: {
                        key.setAppearance(whiteApp);
                        break;
                }
                case 2: {
                        key.setAppearance(blackApp);
                        break;
                }
                case 3: {
                        key.setAppearance(whiteApp);
                        break;
                }
                case 4: {
                        key.setAppearance(blackApp);
                        break;
                }
                case 5: {
                        key.setAppearance(whiteApp);
                        break;
                }
                case 6: {
                        key.setAppearance(blackApp);
                        break;
                }
                case 7: {
                        key.setAppearance(whiteApp);
                        break;
                }
                case 8: {
                        key.setAppearance(whiteApp);
                        break;
                }
                case 9: {
                        key.setAppearance(blackApp);
                        break;
                }
                case 10: {
                        key.setAppearance(whiteApp);
```

```
                        break;
                    }
                    case 11: {
                        key.setAppearance(blackApp);
                        break;
                    }
            }

     }
     else{
            //якщо об'єкт не є клавішею, кришкою(малою чи великою), то додаємо його у сцену
                    if (!name.equals("stick") & !name.equals("cover_big") & !name.equals("cover_small")){
                    Shape3D nextObject = (Shape3D) pianoNamedObjects.get(name);
                    nextObject.setPickable(false);
                        tgPiano.addChild(nextObject.cloneTree());
            }
            }
      //виведення імен об'єктів у консоль
      System.out.println("Name: "+name);
    }
    //виокремлюємо велику кришку
    Shape3D big_cover = (Shape3D)((Shape3D) pianoNamedObjects.get("cover_big")).cloneTree();
    big_cover.setUserData("cover_big");
    big_cover.setAppearance(greyApp);
    //виокремлюємо малу кришку
    Shape3D small_cover = (Shape3D)((Shape3D) pianoNamedObjects.get("cover_small")).cloneTree();
    small_cover.setUserData("cover_small");
    small_cover.setAppearance(greyApp);
    //виокремлюємо палицю, що підтримує велику кришку
    Shape3D stick = (Shape3D)((Shape3D) pianoNamedObjects.get("stick")).cloneTree();
    stick.setUserData("stick");
    stick.setAppearance(greyApp);
    //рух малої кришки
    //переносимо центр повороту
    TransformGroup smallTransXformGroup = translate(
                    small_cover,
                    new Vector3f(0.1f,0.0f,0.7f));
   //рух кришки не починається без додаткових дій користувача
    Alpha small_cover_alpha = new Alpha(1,2000);
    small_cover_alpha.setStartTime(Long.MAX_VALUE);
    //поворот кришки - закриття
    float startAngle1 = 0.0f;
    float endAngle1 =(float)Math.PI/4;
    TransformGroup smallRotXformGroup =
      rotate(smallTransXformGroup,small_cover_alpha,startAngle1,endAngle1);
    //рух кришки - відкриття
    //рух кришки не починається без додаткових дій користувача
    Alpha small_cover_alpha1 = new Alpha(1,2000);
    small_cover_alpha1.setStartTime(Long.MAX_VALUE);
    TransformGroup smallRotXformGroup1 =
                  rotate(smallRotXformGroup,small_cover_alpha1,startAngle1,-endAngle1);
    Transform3D coverObject = new Transform3D();
    coverObject.rotY(5*Math.PI/6);
    TransformGroup tgSmallCoverObject = new TransformGroup(coverObject);
    tgSmallCoverObject.addChild(smallRotXformGroup1);
    //перенос кришки на потрібну позицію
    TransformGroup tgCoverSmall = translate(
                        tgSmallCoverObject,
```

```
        new Vector3f(-0.312f,-0.023f,0.75f));

  BoundingSphere bounds = (BoundingSphere)tgSmallCoverObject.getBounds();
  //рух великої кришки
  //перенос центру повороту
  TransformGroup bigCovTransXformGroup = translate(
                big_cover,
                new Vector3f(0.0f,0.0f,0.0f));
  //рух кришки не починається без додаткових дій користувача
  Alpha big_cover_alpha = new Alpha(1,2000);
  big_cover_alpha.setStartTime(Long.MAX_VALUE);
  //рух великої кришки - закриття
  float startAngle2 = 0.0f;
  float endAngle2 =(float)(Math.PI/5.47);
  TransformGroup bigCovRotXformGroup =
    rotateY(bigCovTransXformGroup,big_cover_alpha,startAngle2,endAngle2);
  //рух кришки не починається без додаткових дій користувача
  Alpha big_cover_alpha1 = new Alpha(1,2000);
  big_cover_alpha1.setStartTime(Long.MAX_VALUE);
  //рух великої кришки - відкриття
  TransformGroup bigCovRotXformGroup1 =
              rotateY(bigCovRotXformGroup,big_cover_alpha1,startAngle2,-endAngle2);

  Transform3D coverBigObject = new Transform3D();
  coverBigObject.rotY(5*Math.PI/6);
  TransformGroup tgCoverBigObject = new TransformGroup(coverBigObject);
  tgCoverBigObject.addChild(bigCovRotXformGroup1);
  //перенос кришки на потрібну позицію
  TransformGroup tgCoverBig = translate(tgCoverBigObject,new Vector3f(0.0f,0.0f,0.0f));
  //створення загальної сцени
  BranchGroup theScene = new BranchGroup();
  theScene.addChild(tgPiano);//додаємо рояль
  theScene.addChild(tgCoverSmall);//додаємо малу кришку
  theScene.addChild(tgCoverBig);//додаємо велику кришку
 //створюємо білий фон
  Background bg = new Background(new Color3f(1.0f,1.0f,1.0f));
  bg.setApplicationBounds(bounds);
  theScene.addChild(bg);
  //масив змінних часу(alphas), які визначають початок руху
  Alpha[] alphas = new Alpha[4];
  alphas[0] = small_cover_alpha;
  alphas[1] = small_cover_alpha1;
  alphas[2] = big_cover_alpha;
  alphas[3] = big_cover_alpha1;
  //об'єкт класу, що реагує на вибір частини роялю мишею
  PickingObjects pickingObjects = new PickingObjects(myCanvas3D,theScene,bounds,alphas);
  theScene.addChild(pickingObjects);
  theScene.compile();
  //додавання сцени до віртуального всесвіту
  su.addBranchGraph(theScene);
 }
 //метод для генерації зовнішньої поверхні
 public static void setToMyDefaultAppearance(Appearance app, Color3f col)
 {
  app.setMaterial(new Material(col,col,col,col,150.0f));
 }
 //метод для додавання освітлення
 public void addLight(SimpleUniverse su)
```

```
{

  BranchGroup bgLight = new BranchGroup();

  BoundingSphere bounds = new BoundingSphere(new Point3d(0.0,0.0,0.0), 100.0);
  Color3f lightColour1 = new Color3f(1.0f,1.0f,1.0f);
  Vector3f lightDir1  = new Vector3f(-1.0f,0.0f,-0.5f);
  DirectionalLight light1 = new DirectionalLight(lightColour1, lightDir1);
  light1.setInfluencingBounds(bounds);

  bgLight.addChild(light1);
  su.addBranchGraph(bgLight);
}

//метод для обертання об'єкту
TransformGroup rotate(Node node,Alpha alpha, float startAngle, float endAngle){

            Transform3D trans = new Transform3D();
            trans.setTranslation(new Vector3d(-0.1f,0.2f,0.0f));

            Transform3D rot = new Transform3D();
            rot.rotZ(Math.PI/2);
            rot.mul(trans);

          TransformGroup xformGroup = new TransformGroup();
  xformGroup.setCapability(
            TransformGroup.ALLOW_TRANSFORM_WRITE);

  RotationInterpolator interpolator =
        new RotationInterpolator(alpha,xformGroup,rot,startAngle,endAngle);

 interpolator.setSchedulingBounds(new BoundingSphere(
                new Point3d(0.0,0.0,0.0),1.0));

  xformGroup.addChild(interpolator);

  xformGroup.addChild(node);

  return xformGroup;

}
//-------------------------------------------------//

//метод для переміщення об'єкту
TransformGroup translate(Node node,Vector3f vector){

   Transform3D transform3D = new Transform3D();
   transform3D.setTranslation(vector);
   TransformGroup transformGroup =
                          new TransformGroup();
   transformGroup.setTransform(transform3D);

   transformGroup.addChild(node);
   return transformGroup;
}
//-------------------------------------------------//
//метод для обертання об'єкту навколо осі z
TransformGroup rotateY(Node node,Alpha alpha, float start_angle,float end_angle){
```

```
            Transform3D trans = new Transform3D();
             trans.setTranslation(new Vector3d(0.7f,0.0f,0.0f));
            Transform3D rot = new Transform3D();
             rot.rotX(Math.PI/2);
             rot.mul(trans);
            TransformGroup xformGroup = new TransformGroup();
   xformGroup.setCapability(
              TransformGroup.ALLOW_TRANSFORM_WRITE);

   RotationInterpolator interpolator =
         new RotationInterpolator(alpha,xformGroup,rot,start_angle,end_angle);
   interpolator.setSchedulingBounds(new BoundingSphere(
                  new Point3d(0.0,0.0,0.0),1.0));
   xformGroup.addChild(interpolator);
   xformGroup.addChild(node);

   return xformGroup;

  }
}

//----------------------//
import javax.media.j3d.*;
import com.sun.j3d.utils.picking.*;
import com.sun.j3d.utils.picking.behaviors.*;
import com.sun.j3d.utils.geometry.*;

public class PickingObjects extends PickMouseBehavior
{

  //масив змінних часу(alphas), які визначають початок руху при виборі об'єкта мишею
  public Alpha[] alphas;
  //значення для визначення руху - відкриття чи закриття кришки
  public int move1;
  public int move2;

  public PickingObjects(Canvas3D pCanvas, BranchGroup root, Bounds pBounds,
                  Alpha[] alphs)
  {
   super(pCanvas,root,pBounds);
   setSchedulingBounds(pBounds);
   alphas = alphs;
   move1 = 0;
   move2 = 0;
  }


  public void updateScene(int xpos, int ypos)
  {
   Shape3D pickedShape = null;
   pickCanvas.setShapeLocation(xpos,ypos);
   PickResult pResult = pickCanvas.pickClosest();//вибір об'єкту
   if (pResult != null)
   {
    pickedShape = (Shape3D) pResult.getNode(PickResult.SHAPE3D);
   }
```

```
if (pickedShape != null)//було вибрано об'єкт
{
 if (pickedShape.getUserData()=="cover_small")//обраний об'єкт - мала кришка
 {
          if (move1==0){
                    //закриваємо кришку якщо вона відкрита
                    alphas[0].setStartTime(System.currentTimeMillis()-alphas[0].getTriggerTime());
                    System.out.println("The object "+pickedShape.getUserData()+" has been selected.");
                    move1++;
          }
          else{
          if (move1 >0 ){
                    //відкриваємо кришку, якщо вона закрита
                    alphas[1].setStartTime(System.currentTimeMillis()-alphas[1].getTriggerTime());
                    System.out.println("The object "+pickedShape.getUserData()+" has been selected.");
                    move1=-1;
          }}
 }
 else
 {
  if (pickedShape.getUserData()=="cover_big")//обраний об'єкт - велика кришка
  {
        if (move2==0){
                  //закриваємо кришку якщо вона відкрита
                  alphas[2].setStartTime(System.currentTimeMillis()-alphas[2].getTriggerTime());
                  System.out.println("The object "+pickedShape.getUserData()+" has been selected.");
                  move2++;

        }
        else{
                  if (move2 >0 ){
                            //відкриваємо кришку, якщо вона закрита
                             alphas[3].setStartTime(System.currentTimeMillis()-alphas[3].getTriggerTime());
                             System.out.println("The object "+pickedShape.getUserData()+" has been selected.");
                             move2 = -1;
                  }}}}}}}
```

Результат роботи програми:

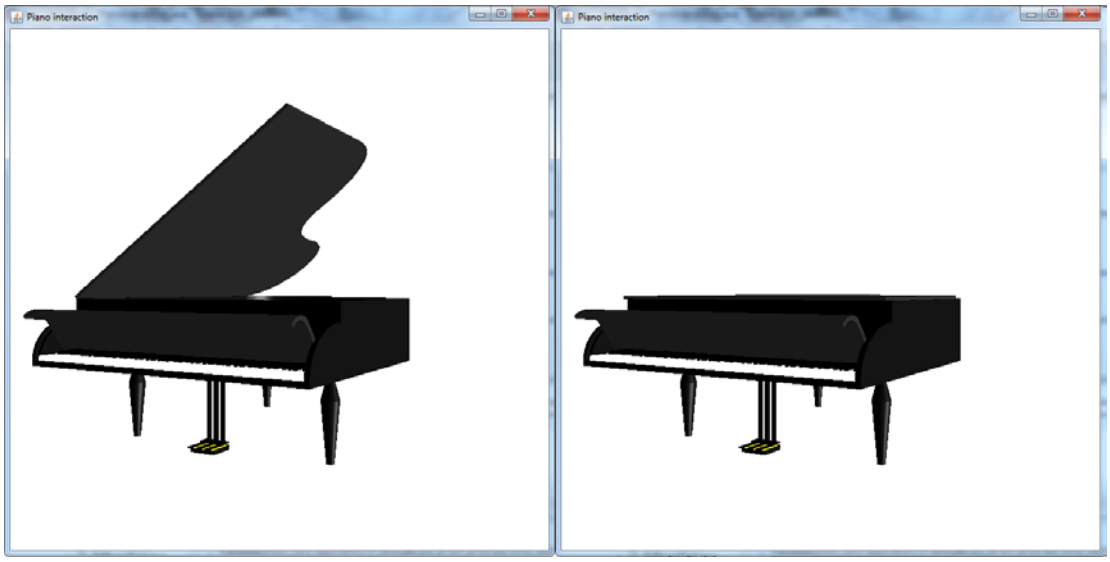

Рис. 2. Рух верхньої кришки роялю

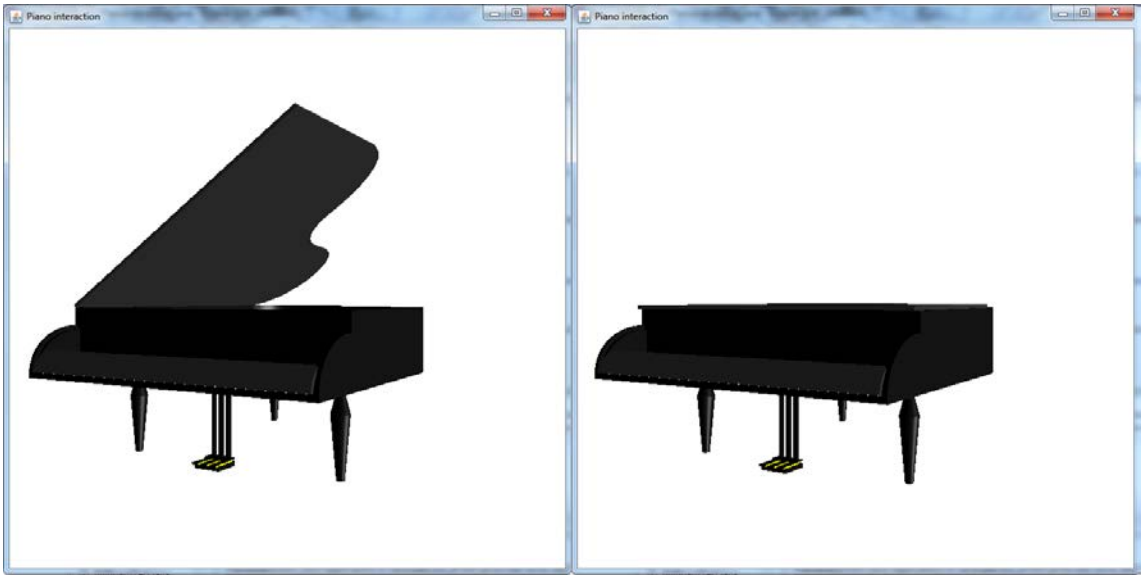

Рис. 3. Рух нижньої та верхньої кришки роялю

У попередніх двох програмах описувалась анімація при повороті об’єкта на кут. Тому приведемо приклад руху по прямій.

Попередньо завантажимо модель таргана. Він буде рухатися зверху екрану до низу.

1. Опишемо рух

```
TransformGroup sceneGroup = new TransformGroup();
sceneGroup.addChild(s.getSceneGroup()); //об’єкт додано у групу трансформації
Transform3D tCrawl = new Transform3D();
tCrawl.rotY(-Math.PI/2);

long crawlTime = 10000;//час, за який тарган переповзе екран
   Alpha crawlAlpha = new Alpha(1,
         Alpha.INCREASING_ENABLE,
         0,
         0, crawlTime,0,0,0,0,0);

   float crawlDistance = 9.0f; //відстань, на яку просунеться об’єкт
   PositionInterpolator posICrawl = new PositionInterpolator(crawlAlpha,
               sceneGroup,tCrawl, -9.0f, crawlDistance);
   BoundingSphere bs = new BoundingSphere(new Point3d(0.0,0.0,0.0),Double.MAX_VALUE);
   posICrawl.setSchedulingBounds(bs);
   sceneGroup.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);//дозвіл на здійснення
трансформацій об’єкту у групі
   sceneGroup.addChild(posICrawl);
   tgObject.addChild(sceneGroup);
    theScene.addChild(tgObject);
```

Повний текст програми:

```
import javax.vecmath.*;
```

```
import com.sun.j3d.utils.universe.*;

import javax.media.j3d.*;

import com.sun.j3d.utils.behaviors.vp.*;
import com.sun.j3d.utils.image.TextureLoader;

import javax.swing.JFrame;
import com.sun.j3d.loaders.*;
import com.sun.j3d.loaders.objectfile.*;
import java.util.Hashtable;
import java.util.Enumeration;

public class Cockroach extends JFrame
{

 //The canvas to be drawn upon.
 public Canvas3D myCanvas3D;


 public Cockroach()
 {

          //механізм для закриття вікна та виходу з програми
            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            //параметри перегляду сцени за замовчанням
            myCanvas3D = new Canvas3D(SimpleUniverse.getPreferredConfiguration());
            //створення SimpleUniverse (віртуального всесвіту)
            SimpleUniverse simpUniv = new SimpleUniverse(myCanvas3D);
            //положення глядача за замовчанням
            simpUniv.getViewingPlatform().setNominalViewingTransform();
            //створення сцени
            createSceneGraph(simpUniv);
            //додання світла у сцену
            addLight(simpUniv);

            //наступні рядки дозволяють навігацію по сцені за допомогою миші
            OrbitBehavior ob = new OrbitBehavior(myCanvas3D);
            ob.setSchedulingBounds(new BoundingSphere(new Point3d(0.0,0.0,0.0),Double.MAX_VALUE));
            simpUniv.getViewingPlatform().setViewPlatformBehavior(ob);
            //параметри вікна програми
            setTitle("Cockroach");
            setSize(700,700);
            getContentPane().add("Center", myCanvas3D);
            setVisible(true);
 }


 public static void main(String[] args)
 {
           Cockroach cockroach = new Cockroach();
 }

 //в цьому методі створюються об'єкти та додаються до сцени
 public void createSceneGraph(SimpleUniverse su)
 {
          // завантаження файлу .obj.
```

```
ObjectFile f = new ObjectFile(ObjectFile.RESIZE);
Scene cockroachScene = null;
try
{
        cockroachScene = f.load("ROACH.obj");
}
catch (Exception e)
{
 System.out.println("File loading failed:" + e);
}
//створення трансформаційної групи для завантаженого об'єкту
Transform3D scaling = new Transform3D();
scaling.setScale(1.0/6);
Transform3D tfRoach = new Transform3D();
tfRoach.rotX(Math.PI/2);
tfRoach.mul(scaling);
TransformGroup tgRoach = new TransformGroup(tfRoach);

TransformGroup sceneGroup = new TransformGroup();
sceneGroup.addChild(cockroachScene.getSceneGroup());
//отримання імен об'єктів, з яких складається тарган
//та вивід в консоль назв об'єктів
Hashtable roachNamedObjects = cockroachScene.getNamedObjects();
Enumeration enumer = roachNamedObjects.keys();
String name;
while (enumer.hasMoreElements())
{
 name = (String) enumer.nextElement();
 System.out.println("Name: "+name);
}
//пофарбуємо тіло таргана у світло-зелений колір
Appearance lightApp = new Appearance();
setToMyDefaultAppearance(lightApp,new Color3f(0.3f,0.3f,0.1f));
Shape3D body = (Shape3D) roachNamedObjects.get("roach_body");
body.setAppearance(lightApp);
BoundingSphere bounds = new BoundingSphere(new Point3d(120.0,250.0,100.0),Double.MAX_VALUE);
BranchGroup theScene = new BranchGroup();
Transform3D tCrawl = new Transform3D();
tCrawl.rotY(-Math.PI/2);

long crawlTime = 10000;//час, за який тарган переповзе екран
   Alpha crawlAlpha = new Alpha(1,
        Alpha.INCREASING_ENABLE,
        0,
        0, crawlTime,0,0,0,0,0);

float crawlDistance = 9.0f; //відстань, на яку просунеться об’єкт
PositionInterpolator posICrawl = new PositionInterpolator(crawlAlpha,
                sceneGroup,tCrawl, -9.0f, crawlDistance);

BoundingSphere bs = new BoundingSphere(new Point3d(0.0,0.0,0.0),Double.MAX_VALUE);
posICrawl.setSchedulingBounds(bs);
sceneGroup.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
sceneGroup.addChild(posICrawl);
tgRoach.addChild(sceneGroup);
theScene.addChild(tgRoach);//додавання об'єкту у сцену
//створюємо білий фон
Background bg = new Background(new Color3f(1.0f,1.0f,1.0f));
```

```
  bg.setApplicationBounds(bounds);
  theScene.addChild(bg);

  theScene.compile();

 //додаємо сцену до віртуального всесвіту
  su.addBranchGraph(theScene);
 }

 //метод для генерації зовнішньої поверхні
 public static void setToMyDefaultAppearance(Appearance app, Color3f col)
 {
  app.setMaterial(new Material(col,col,col,col,150.0f));
 }
 //метод для додавання освітлення
 public void addLight(SimpleUniverse su)
 {

  BranchGroup bgLight = new BranchGroup();

  BoundingSphere bounds = new BoundingSphere(new Point3d(0.0,0.0,0.0), 100.0);
  Color3f lightColour1 = new Color3f(1.0f,1.0f,1.0f);
  Vector3f lightDir1  = new Vector3f(-1.0f,0.0f,-0.5f);
  DirectionalLight light1 = new DirectionalLight(lightColour1, lightDir1);
  light1.setInfluencingBounds(bounds);

  bgLight.addChild(light1);
  su.addBranchGraph(bgLight);
 }
}
```

Результат роботи програми:

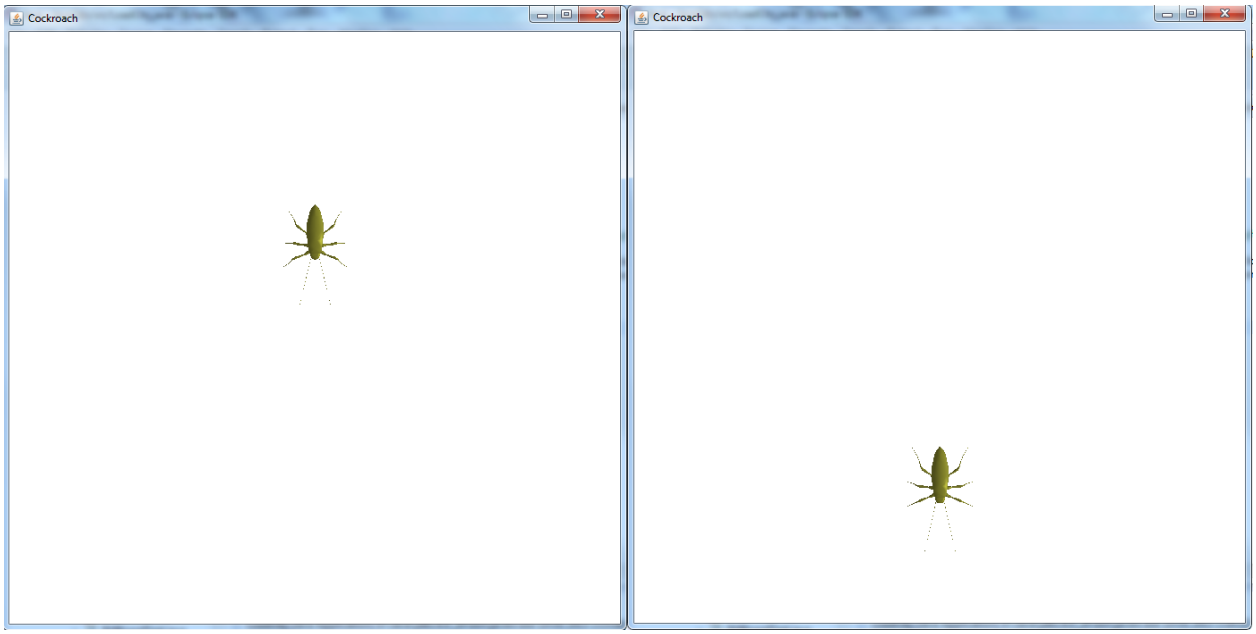

Рис. 4. Рух таргана

Увага! Дана програма приведена у якості демонстрації даного виду трансформацій і не є прикладом виконання лабораторної роботи у повному обсязі.

Завдання

Виконати анімацію тривимірної сцени за варіантом.

Варіанти завдань

1. Анімація павука «чорна вдова» black_widow.obj. Рух ніжок, пересування по екрану.
2. Анімація автомобіля car.obj. Рух коліс, пересування по екрану, обов’язкові повороти авто.
3. Анімація риби fish.obj. Риба повинна рухати плавцями, хвостом, головою, рухатися по екрану.
4. Анімація гусака goose.obj. Гусак повинен рухати ногами, ходити по екрану, з поворотами .
5. Анімація вертольоту helicopter.obj. У вертольота повинні рухатися обидва гвинти, вертоліт повинен пересуватися по екрану.
6. Анімація жука «божа корівка» ladybug.obj. Жук повинен рухати лапками, пересуватися по екрану.
7. Анімація одноокого циклопа Майка (із мультфільму) mike.obj. Повинен рухати руками і ногами, пересуватися по екрану.
8. Анімація тренеру покемонів (із мультфільму) pokemon_trainer.obj. Повинен рухати руками, покемони повинні вилітати з рук, рухатися по екрану та повертатися в руки.
9. Анімація білка Скрата (із мультфільму) scrat.obj. Горіх повинен рухатися по екрану, білка – за ним; білка повинна рухати руками або ногами, хвостом.
10. Анімація воїна з сокирою warrior.obj. Воїн повинен взяти в руки сокиру, рухати головою, руками, сокирою.

Звіт повинен містити

1. Титульний аркуш.
2. Постановку завдання.
3. Хід роботи.
4. Результат роботи у вигляді скріншотів програми

Література

1. Selman,D. Java 3D Programming [Text] / Selman D. — NY : Manning. — 2002. — 400 p.
2. Davidson, A. Killer Game Programming in Java [Text] / A. Davidson. — CA : O'REILLY. — 2005. — 970 p.
3. Java 3D Implementation Documentation.

Контрольні запитання

1. Які основні види трансформацій об'єктів у Java3d ви знаєте?
2. Назвіть основні об'єкти тривимірної сцени.
3. Яка структура .obj файлу?
4. Як виокремити частину складного об'єкта у окремий об'єкт?
5. Як ви знаєте, по замовчанню об'єкт рухається вздовж або навколо осі x. Які перетворення необхідно зробити в коді програми, щоб рух відбувався навколо осі z або вздовж осі y?
6. Як накласти текстуру лише на частину складного об'єкта?