МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота № 6

з дисципліни “Математичні та алгоритмічні основи комп’ютерної графіки”

Виконав	Зарахована
студент ІІІ курсу	“____” “____________” 20___ р.
групи КП-82	викладачем
Мельничук Олексій Геннадійович	Шкурат Оксаною Сергіївною
(*прізвище, ім’я, по батькові*)	(*прізвище, ім’я, по батькові*)

варіант № 12

Київ 2021

Тема:Анімація тривимірних об’єктів

Мета: Навчитися анімувати складні об’єкти тривимірної сцени.

Завдання

Виконати анімацію тривимірної сцени за варіантом.

Код програми

man.java

```
package lab6;

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.Map;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Toolkit;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Enumeration;

public class Bruh extends JFrame {
	public Canvas3D myCanvas3D;

	static BranchGroup root;

	static String assetPath = "D:\\eclipse_labs\\ffs\\src\\lab6\\resources\\";
	static String modelName = "belca.obj";
	static String bgName = "back.jpg";
	static String sword = "sword.obj";
	static String swordTex = "sword.png";
	static String man = "test.obj";

	Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();


	static Map<String, Shape3D> nameMap;

	static Scene sceneMan;
	static Scene[] sceneSwords = new Scene[8];
	static SimpleUniverse universe;


	private void configureWindow() {
		setTitle("Car Animation Example");
		setSize(1280, 640);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}

	private void configureCanvas() {
		myCanvas3D = new Canvas3D(SimpleUniverse.getPreferredConfiguration());
		System.out.printf("heigth: %s\n",
SimpleUniverse.getPreferredConfiguration().getBounds().getHeight());
		myCanvas3D.setDoubleBufferEnable(true);
		myCanvas3D.setSize(screenSize);
		getContentPane().add(myCanvas3D, BorderLayout.CENTER);
	}
	private void configureUniverse() {
		root = new BranchGroup();
		universe = new SimpleUniverse(myCanvas3D);
		universe.getViewingPlatform().setNominalViewingTransform();
	}

	public Bruh() {
		//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("idk get a job");
		setSize(700,700);
		getContentPane().add("Center", myCanvas3D);
		setVisible(true);
	}

	public static void main(String[] args)
	{
		Bruh alarmAnimation = new Bruh();
	}
		public static Scene getSceneFromFile(String location) throws IOException {
			ObjectFile file = new ObjectFile(ObjectFile.RESIZE);
			file.setFlags(ObjectFile.RESIZE | ObjectFile.TRIANGULATE | 
ObjectFile.STRIPIFY);
			return file.load(new FileReader(location));
		}


	//в цьому методі створюються об'єкти та додаються до сцени
	public void createSceneGraph(SimpleUniverse su)
	{
	BoundingSphere bs = new BoundingSphere(new Point3d(0.0,0.0,0.0),Double.MAX_VALUE);
	BranchGroup root = new BranchGroup();
	Background bg = new Background(new Color3f(-1.0f,-1.0f,1.0f));

		ObjectFile f = new ObjectFile(ObjectFile.RESIZE);

		sceneMan = null;

		try
		{
			for (int i = 0; i<8; i++)
			{
					sceneMan = getSceneFromFile(assetPath + man);

				sceneSwords[i] = f.load(assetPath + sword);
			}
		}
		catch (Exception e)
		{
			System.out.println("File loading failed:" + e);
		}

		Hashtable manParts = sceneMan.getNamedObjects();

		addAppearance(sceneMan);
	      Shape3D[] swordsShapes = new Shape3D[8];
		Transform3D[] tfSwords = new Transform3D[8];

		for (int i = 0; i<8; i++)
		{
			addAppearance(sceneSwords[i]);
			swordsShapes[i] = (Shape3D)
sceneSwords[i].getNamedObjects().get("sword");
			tfSwords[i] = new Transform3D();
		}

		float squared = 0.353f*5;
		float length = 0.5f*5;


		tfSwords[0].setTranslation(new Vector3d(length, 0.0f, 0.0f));
		tfSwords[1].setTranslation(new Vector3d(squared,0.0f,squared));
		tfSwords[2].setTranslation(new Vector3d(0.0f, 0.0f, length));
		tfSwords[3].setTranslation(new Vector3d(-squared,0.0f,squared));
		tfSwords[4].setTranslation(new Vector3d(-length, 0.0f, 0.0f));
		tfSwords[5].setTranslation(new Vector3d(-squared, 0.0f,-squared));
		tfSwords[6].setTranslation(new Vector3d(0.0f,0.0f,-length));
		tfSwords[7].setTranslation(new Vector3d(squared,0.0f,-squared));

		TransformGroup[] tgSwordsTranslate = new TransformGroup[8];
		TransformGroup[] tgSwordsRotate = new TransformGroup[8];
		TransformGroup[] tgSwordsRotate2 = new TransformGroup[8];














	int movesCount = 100; // moves count
	int movesDuration = 500; // moves for 0,3 seconds
	int startTime = 0; // launch animation after timeStart seconds

	      Alpha swordRot = new Alpha(movesCount, Alpha.INCREASING_ENABLE, startTime, 0,
movesDuration,0,0,0,0,0);

	Transform3D taleRotAxisSmall = new Transform3D();
	taleRotAxisSmall.set(new Vector3d(0.0, 0.0, 0.0));//rotation axis location
```

```
        taleRotAxisSmall.setRotation(new AxisAngle4d(0.0, 0.0, 0.0, Math.PI/8));

              for (int i = 0; i<8; i++)
              {//
                     tgSwordsTranslate[i] = new TransformGroup();
                     tgSwordsRotate[i] = new TransformGroup();
                     tgSwordsRotate2[i] = new TransformGroup();


tgSwordsTranslate[i].setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
                     tgSwordsRotate[i].setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);

tgSwordsRotate2[i].setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);




                     tgSwordsRotate2[i].addChild(swordsShapes[i].cloneTree());




                     tgSwordsRotate[i] = rotate(tgSwordsRotate2[i], swordRot,
taleRotAxisSmall);


                     tgSwordsTranslate[i].setTransform(tfSwords[i]);
                     tgSwordsTranslate[i].addChild(tgSwordsRotate[i]);

              }





            TransformGroup swords = new TransformGroup();

              for (int i = 0; i<8; i++)
              {
                  swords.addChild(tgSwordsTranslate[i]);
              }

//
        TransformGroup whiteTransXformGroup = translate(
                   swords,
                new Vector3f(0, 0, 0));//MOVE FROM AXIS

        int movesCount2 = 100; // moves count
        int movesDuration2 = 8000; // moves for 0,3 seconds
        int startTime2 = 50; // launch animation after timeStart seconds


        Transform3D taleRotAxisBig = new Transform3D();
        taleRotAxisBig.set(new Vector3d(0.0, 0.0, 0.0));//rotation axis location
        taleRotAxisBig.setRotation(new AxisAngle4d(0, 1, 0, Math.PI/2));
        Alpha newa = new Alpha(movesCount2, Alpha.INCREASING_ENABLE, startTime2, 0,
movesDuration2,0,0,0,0,0);
        TransformGroup whiteRotXformGroup = rotate(whiteTransXformGroup,  newa,
taleRotAxisBig);
        root.addChild(whiteRotXformGroup);



              TransformGroup tgMan = new TransformGroup();
```

```
                Shape3D head1 = (Shape3D) manParts.get("head");
                Shape3D head2 = (Shape3D) manParts.get("helmet");
                Shape3D head3 = (Shape3D) manParts.get("eyes");
                TransformGroup tgHead = new TransformGroup();
                tgHead.addChild(head1.cloneTree());
                tgHead.addChild(head2.cloneTree());
                tgHead.addChild(head3.cloneTree());

        Transform3D locateHead = new Transform3D();
        locateHead.setTranslation(new Vector3d(0.0, 1.3757, -0.1));
        tgHead.setTransform(locateHead);

        TransformGroup movedHead = new TransformGroup();

        movedHead.addChild(tgHead);

            Alpha headAlpha = new Alpha(1, Alpha.INCREASING_ENABLE, startTime, 0, 5000
,0,0,0,0,0);
            Transform3D headRotAxis = new Transform3D();

            headRotAxis.set(new Vector3d(0.0, 1.3757, -0.05));
            headRotAxis.setRotation(new AxisAngle4d(0.0, 0.0, 1.0, Math.PI/2));

            RotationInterpolator headRotat = new RotationInterpolator(headAlpha, movedHead,
headRotAxis, 0.0f,(float) Math.PI/32);
        headRotat.setSchedulingBounds(bs);
        movedHead.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        movedHead.setTransform(headRotAxis);
        movedHead.addChild(headRotat);



        tgMan.addChild(movedHead);



                Shape3D hand1 = (Shape3D) manParts.get("hands");
                Shape3D hand2 = (Shape3D) manParts.get("palms");
                TransformGroup tgHands = new TransformGroup();
                tgHands.addChild(hand1.cloneTree());
                tgHands.addChild(hand2.cloneTree());

                TransformGroup movedHand = new TransformGroup();
                Transform3D locateHand = new Transform3D();
                locateHand.setTranslation(new Vector3d(0.0, 1.2957, -0.10));
                movedHand.setTransform(locateHand);

                movedHand.addChild(tgHands);



            Alpha handAlpha = new Alpha(2, Alpha.INCREASING_ENABLE, startTime, 0, 5000
,0,0,0,0,0);
            Transform3D handRotAxis = new Transform3D();

            handRotAxis.setRotation(new AxisAngle4d(0.0, 0.0, 1.0, Math.PI/2));
           RotationInterpolator handRotat = new RotationInterpolator(handAlpha, tgHands,
handRotAxis, 0.0f,(float)Math.PI * 2);
           handRotat.setSchedulingBounds(bs);
```

```
          tgHands.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
          tgHands.addChild(handRotat);

        tgMan.addChild(movedHand);




           Shape3D boots = (Shape3D) manParts.get("boots");
           Shape3D chest = (Shape3D) manParts.get("chest");
           Shape3D skirt = (Shape3D) manParts.get("skirt");
           TransformGroup tgboots = new TransformGroup();
           TransformGroup tgchest = new TransformGroup();
           TransformGroup tgskirt = new TransformGroup();
           tgboots.addChild(boots.cloneTree());
           tgchest.addChild(chest.cloneTree());
           tgskirt.addChild(skirt.cloneTree());

           tgMan.addChild(tgboots);
           tgMan.addChild(tgskirt);
           tgMan.addChild(tgchest);

           root.addChild(tgMan);






            root.compile();
            su.addBranchGraph(root);


        }


          private 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;
          }

          private TransformGroup rotate(Node node, Alpha alpha, Transform3D taleRotAxis){
              TransformGroup xformGroup = new TransformGroup();
              xformGroup.setCapability(
                      TransformGroup.ALLOW_TRANSFORM_WRITE);

              RotationInterpolator interpolator =
                      new RotationInterpolator(alpha,xformGroup, taleRotAxis,  0.0f,
(float) Math.PI*2);

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

              xformGroup.addChild(interpolator);
              xformGroup.addChild(node);
              return xformGroup;
          }


         private TextureLoader getTextureLoader(String path) throws IOException {
              ClassLoader classLoader = Thread.currentThread().getContextClassLoader();
              var textureResource = classLoader.getResource(path);
              if (textureResource == null) {
                  throw new IOException("Couldn't find texture: " + path);
              }
              return new TextureLoader(textureResource.getPath(), myCanvas3D);
          }

             private void addAppearance(Scene s){
                    nameMap = s.getNamedObjects();
                    for (String name : nameMap.keySet()) {
                           Shape3D car = nameMap.get(name);

                           Appearance carAppearance = new Appearance();
                           carAppearance.setTexture(getTexture(assetPath + "png//" + name +
".png"));
                           TextureAttributes texAttr = new TextureAttributes();
                           texAttr.setTextureMode(TextureAttributes.COMBINE);
                           carAppearance.setTextureAttributes(texAttr);
                           carAppearance.setMaterial(getMaterial());

                           car.setAppearance(carAppearance);
                    }

             }


             Texture getTexture(String path) {
                    TextureLoader textureLoader = new TextureLoader(path, myCanvas3D);
                    Texture texture = textureLoader.getTexture();
                    texture.setBoundaryModeS(Texture.WRAP);
                    texture.setBoundaryModeT(Texture.WRAP);
                    texture.setBoundaryColor(new Color4f(1.0f, 1.0f, 0.0f, 0.0f));
                    return texture;
             }
             private void addOtherLight() {
                    Color3f directionalLightColor = new Color3f(Color.BLACK);
                    Color3f ambientLightColor = new Color3f(Color.WHITE);
                    Vector3f lightDirection = new Vector3f(-1F, -1F, -1F);

                    AmbientLight ambientLight = new AmbientLight(ambientLightColor);
                    DirectionalLight directionalLight = new
DirectionalLight(directionalLightColor, lightDirection);

                    Bounds influenceRegion = new BoundingSphere();

                    ambientLight.setInfluencingBounds(influenceRegion);
                    directionalLight.setInfluencingBounds(influenceRegion);
                    root.addChild(ambientLight);
                    root.addChild(directionalLight);
             }


             Material getMaterial() {
                    Material material = new Material();


                 Color3f ambient = new Color3f(1.0f, 1.0f, 1.0f);
                 Color3f diffuse = new Color3f(1.0f, 1.0f, 1.0f);
                 Color3f specular = new Color3f(1.0f, 1.0f, 1.0f);


//             material.setAmbientColor(new Color3f(new Color(221, 221, 221)));
//             material.setDiffuseColor(new Color3f(new Color(200, 200, 200)));
//             material.setSpecularColor(new Color3f(new Color(200, 200, 200)));

                    material.setAmbientColor(ambient);
                    material.setDiffuseColor(diffuse);
                    material.setSpecularColor(specular);
                    material.setShininess(1f);
                    material.setLightingEnable(true);
                    return material;
             }



       //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);
       }
}
```

firstmainclass.java

```
package lab5;

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

import java.awt.Color;
import javax.media.j3d.*;
import javax.media.j3d.Material;
import javax.vecmath.*;
import javax.media.j3d.Background;

import com.sun.j3d.loaders.*;
import com.sun.j3d.loaders.objectfile.ObjectFile;
import com.sun.j3d.loaders.lw3d.Lw3dLoader;
import com.sun.j3d.utils.image.TextureLoader;
import java.awt.*;
import java.io.FileReader;
import java.io.IOException;
import java.util.Map;
import javax.swing.JFrame;

public class FirstMainClass extends JFrame {
	static SimpleUniverse universe;
	static Scene scene;
	static Map<String, Shape3D> nameMap;
	static BranchGroup root;
	static Canvas3D canvas;


	static String assetPath = 
"C:\\Users\\datru\\Desktop\\study2021\\maokg\\lab5\\res\\";
	static String modelName = "camaro2.obj";
	static String bgName = "garage.jpg";

	static TransformGroup wholeCar;
	static Transform3D transform3D;

	public FirstMainClass() throws IOException {
		configureWindow();
		configureCanvas();
		configureUniverse();
		addModelToUniverse();
		setCarElementsList();
		addAppearance();
		addImageBackground();
		addLightToUniverse();
		addOtherLight();
		ChangeViewAngle();
		root.compile();
		universe.addBranchGraph(root);
	}

	private void configureWindow() {
		setTitle("Car Animation Example");
		setSize(760, 640);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}

	private void configureCanvas() {
		canvas = new Canvas3D(SimpleUniverse.getPreferredConfiguration());
		canvas.setDoubleBufferEnable(true);
```

```
        getContentPane().add(canvas, BorderLayout.CENTER);
    }

    private void configureUniverse() {
        root = new BranchGroup();
        universe = new SimpleUniverse(canvas);
        universe.getViewingPlatform().setNominalViewingTransform();
    }

    private void addModelToUniverse() throws IOException {
        scene = getSceneFromFile(assetPath + modelName);
        root = scene.getSceneGroup();
    }

    private void addLightToUniverse() {
        Bounds bounds = new BoundingSphere();
        Color3f color = new Color3f(96 / 255f, 96 / 255f, 96 / 255f);
        Vector3f lightdirection = new Vector3f(0f, -1f, 0f);
        DirectionalLight dirlight = new DirectionalLight(color, lightdirection);
        dirlight.setInfluencingBounds(bounds);
        root.addChild(dirlight);
    }

    private void printModelElementsList(Map<String, Shape3D> nameMap) {
        for (String name : nameMap.keySet()) {
            System.out.printf("Name: %s\n", name);
        }
    }

    private void setCarElementsList() {
        nameMap = scene.getNamedObjects();

        // Print elements of your model:
        printModelElementsList(nameMap);

        wholeCar = new TransformGroup();
        transform3D = new Transform3D();
        transform3D.setScale(new Vector3d(1, 1, 1));
        wholeCar.setTransform(transform3D);


        for (String name : nameMap.keySet()) {
            root.removeChild(nameMap.get(name));
            wholeCar.addChild(nameMap.get(name));
            wholeCar.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        }

        root.addChild(wholeCar);
    }

    Texture getTexture(String path) {
        TextureLoader textureLoader = new TextureLoader(path, canvas);
        Texture texture = textureLoader.getTexture();
        texture.setBoundaryModeS(Texture.WRAP);
        texture.setBoundaryModeT(Texture.WRAP);
        texture.setBoundaryColor(new Color4f(0.0f, 1.0f, 0.0f, 0.0f));
        return texture;
    }

    Material getMaterial() {
        Material material = new Material();


      Color3f ambient = new Color3f(1.0f, 1.0f, 1.0f);
```

```
           Color3f diffuse = new Color3f(1.0f, 1.0f, 1.0f);
           Color3f specular = new Color3f(1.0f, 1.0f, 1.0f);


//        material.setAmbientColor(new Color3f(new Color(221, 221, 221)));
//        material.setDiffuseColor(new Color3f(new Color(200, 200, 200)));
//        material.setSpecularColor(new Color3f(new Color(200, 200, 200)));

              material.setAmbientColor(ambient);
              material.setDiffuseColor(diffuse);
              material.setSpecularColor(specular);
              material.setShininess(1f);
              material.setLightingEnable(true);
              return material;
       }

       private void addAppearance() throws IOException {
              for (String name : nameMap.keySet()) {
                     Shape3D car = nameMap.get(name);

                     Appearance carAppearance = new Appearance();
                     carAppearance.setTexture(getTexture(assetPath + "png//" + name +
".png"));
                     TextureAttributes texAttr = new TextureAttributes();
                     texAttr.setTextureMode(TextureAttributes.COMBINE);
                     carAppearance.setTextureAttributes(texAttr);
                     carAppearance.setMaterial(getMaterial());

                     car.setAppearance(carAppearance);
              }

       }

       private void addColorBackground() {
              Background background = new Background(new Color3f(Color.CYAN));
              BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0),
100.0);
              background.setApplicationBounds(bounds);
              root.addChild(background);
       }

       private void addImageBackground() {
              TextureLoader t = new TextureLoader(assetPath + bgName, canvas);
              Background background = new Background(t.getImage());
              background.setImageScaleMode(Background.SCALE_FIT_ALL);
              BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0),
100.0);
              background.setApplicationBounds(bounds);
              root.addChild(background);
       }

       private void ChangeViewAngle() {
              ViewingPlatform vp = universe.getViewingPlatform();
              TransformGroup vpGroup = vp.getMultiTransformGroup().getTransformGroup(0);
              Transform3D vpTranslation = new Transform3D();
              Vector3f translationVector = new Vector3f(0.0F, 0.0F, 6F);
              vpTranslation.setTranslation(translationVector);
              vpGroup.setTransform(vpTranslation);
       }

       private void addOtherLight() {
              Color3f directionalLightColor = new Color3f(Color.BLACK);
              Color3f ambientLightColor = new Color3f(Color.WHITE);
              Vector3f lightDirection = new Vector3f(-1F, -1F, -1F);
```

```
		AmbientLight ambientLight = new AmbientLight(ambientLightColor);
		DirectionalLight directionalLight = new
DirectionalLight(directionalLightColor, lightDirection);

		Bounds influenceRegion = new BoundingSphere();

		ambientLight.setInfluencingBounds(influenceRegion);
		directionalLight.setInfluencingBounds(influenceRegion);
		root.addChild(ambientLight);
		root.addChild(directionalLight);
	}

	public static Scene getSceneFromFile(String location) throws IOException {
		ObjectFile file = new ObjectFile(ObjectFile.RESIZE);
		file.setFlags(ObjectFile.RESIZE | ObjectFile.TRIANGULATE |
ObjectFile.STRIPIFY);
		return file.load(new FileReader(location));
	}

	// Not always works
	public static Scene getSceneFromLwoFile(String location) throws IOException {
		Lw3dLoader loader = new Lw3dLoader();
		return loader.load(new FileReader(location));
	}

	public static void main(String[] args) {
		try {
			FirstMainClass window = new FirstMainClass();
			AnimateCar carMovement = new AnimateCar(wholeCar, transform3D, window);
			window.addKeyListener(carMovement);
			window.setVisible(true);
		} catch (IOException ex) {
			System.out.println(ex.getMessage());
		}
	}
}
```

Результати роботи програми

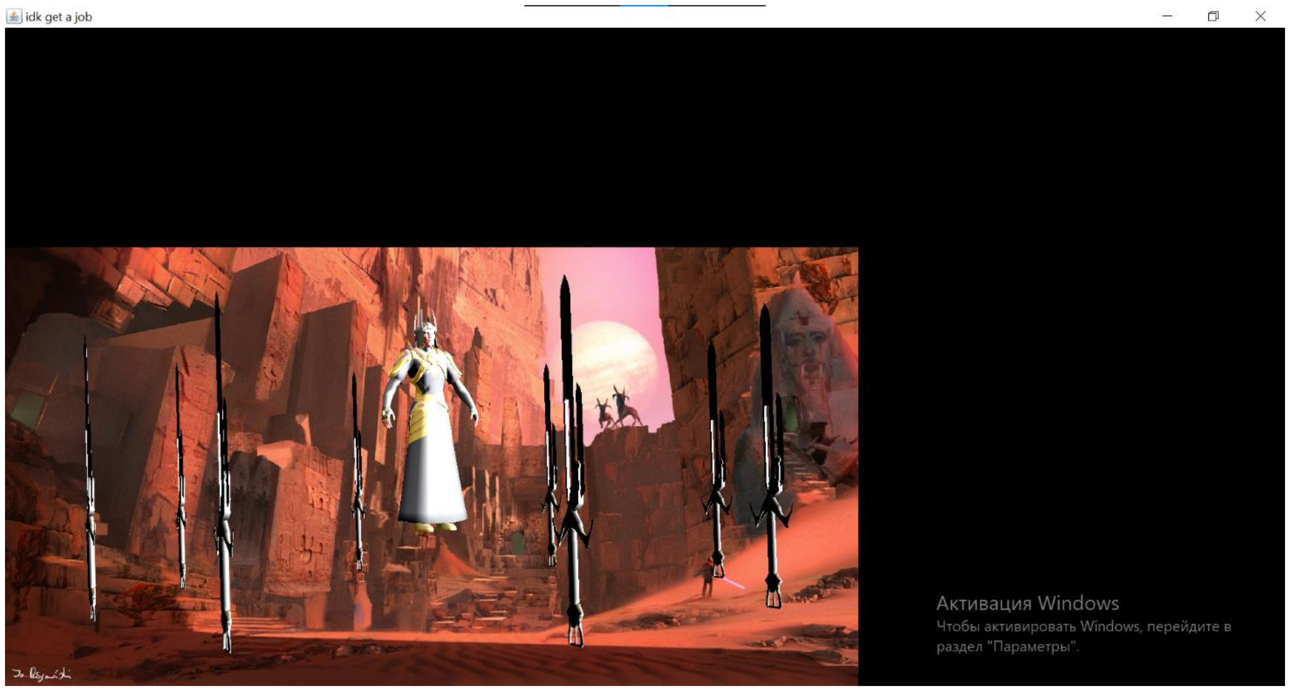

Рис.1. Мечі обертаються по колу

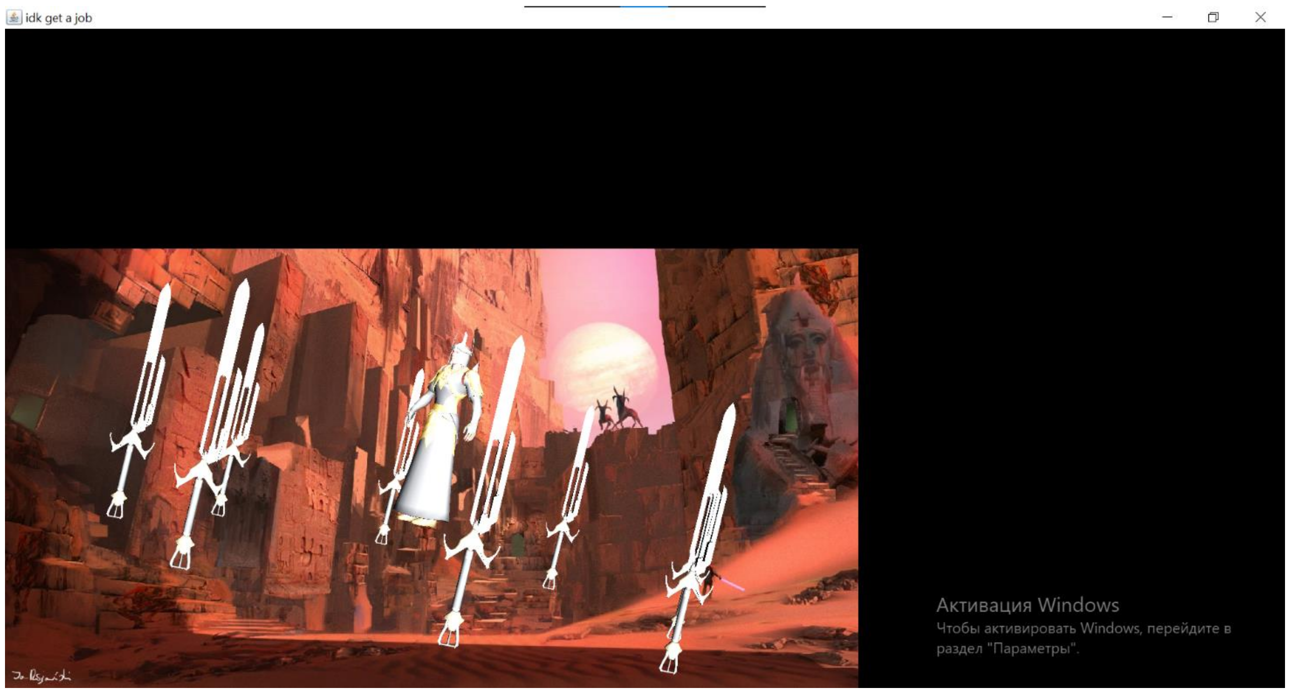

Рис.2. Мечі повертаються також по своїй осі, персонаж рухає головою

Висновки

Виконавши дану лабораторну роботу я навчився анімувати складні об’єкти тривимірної сцени, виділяти окремі об’єкти з .obj файлів.