

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 №** **4**

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

|  |  |  |  |
| --- | --- | --- | --- |
| Виконав  студент III курсу  групи КП-83  Мар’яненко Роман Геннадійович  (*прізвище, ім’я, по батькові*)  варіант № 11 |  |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладачем  Шкурат Оксаною Сергіївною (*прізвище, ім’я, по батькові*) |

Київ 2020

**Варіант завдання**

За допомогою засобів, що надає бібліотека Java3D, побудувати тривимірний об’єкт. Для цього скористатися основними примітивами, що буде доцільно використовувати згідно варіанту: сфера, конус, паралелепіпед, циліндр. Об'єкт має складатися з 5-15 примітивів. Задати матеріал кожного примітиву, в разі необхідності накласти текстуру. В сцені має бути мінімум одне джерело освітлення.

Виконати анімацію сцени таким чином, щоб можна було розглянути об'єкт з усіх сторін. За бажанням можна виконати інтерактивні взаємодію з об'єктом за допомогою миші та клавіатури.

**Варіант:** Гантелі

**Лістинг коду програми**

|  |
| --- |
| Main.java |
| package com.company;  import com.sun.j3d.utils.applet.MainFrame;  import com.sun.j3d.utils.geometry.\*;  import com.sun.j3d.utils.universe.SimpleUniverse;  import javax.media.j3d.\*;  import javax.swing.\*;  import javax.vecmath.\*;  import java.applet.Applet;  import java.awt.\*;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  public class Main extends Applet implements ActionListener {  private final TransformGroup tg = new TransformGroup();  private final Transform3D t3d = new Transform3D();  private double angle = 0;  private final double velocity = 0.02;  public static void main(String[] args) {  MainFrame mf = new MainFrame(new Main(), 700, 700);  mf.setExtendedState(mf.getExtendedState() | JFrame.MAXIMIZED\_BOTH);  mf.run();  }  private Main() {  setLayout(new BorderLayout());  GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();  Canvas3D c = new Canvas3D(config);  add("Center", c);  SimpleUniverse universe = new SimpleUniverse(c);  Timer timer = new Timer(20, this);  timer.start();  universe.getViewingPlatform().setNominalViewingTransform();  universe.addBranchGraph(createBranchGroup());  }  @Override  public void actionPerformed(ActionEvent e) {  t3d.rotY(angle);  tg.setTransform(t3d);  angle += velocity;  }  private BranchGroup createBranchGroup() {  BranchGroup bg = new BranchGroup();  tg.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);  bg.addChild(tg);  createDumbbells(1.2, 0.6);  Color3f lightColor = new Color3f(new Color(64, 64, 64));  BoundingSphere ib = new BoundingSphere(new Point3d(0, 0, 0), 100);  Vector3f lightDirection1 = new Vector3f(-1, 1, -1);  DirectionalLight light1 = new DirectionalLight(lightColor, lightDirection1);  light1.setInfluencingBounds(ib);  bg.addChild(light1);  Vector3f lightDirection2 = new Vector3f(-1, 0, 0);  DirectionalLight light2 = new DirectionalLight(lightColor, lightDirection2);  light2.setInfluencingBounds(ib);  bg.addChild(light2);  AmbientLight al = new AmbientLight(new Color3f(new Color(100, 255, 255)));  al.setInfluencingBounds(ib);  bg.addChild(al);  return bg;  }  private void createDumbbells(double size, double distance) {  float x = (float)(size \* distance / 2);  createDumbbell(size, new Vector3f(x, 0, 0));  createDumbbell(size, new Vector3f(-x, 0, 0));  }  private void createDumbbell(double size, Vector3f initialVector) {  float x = initialVector.getX();  float y = initialVector.getY();  float z = initialVector.getZ();  // body  double bodyRadius = size \* 0.025;  double bodyHeight = size \* 0.6;  appendPrimitive(Dumbbell.getBody(bodyRadius, bodyHeight), initialVector);  // plates  double plateHeight = size \* 0.04;  double bigPlateRadius = size \* 0.2;  double middlePlateRadius = size \* 0.15;  double smallPlateRadius = size \* 0.1;  float bigPlateDistance = (float)(bodyHeight / 2 - 3.5 \* plateHeight);  float middlePlateDistance = bigPlateDistance + (float)plateHeight;  float smallPlateDistance = middlePlateDistance + (float)plateHeight;  appendPrimitive(Dumbbell.getPlate(bigPlateRadius, plateHeight), new Vector3f(x, y + bigPlateDistance, z));  appendPrimitive(Dumbbell.getPlate(bigPlateRadius, plateHeight), new Vector3f(x, y - bigPlateDistance, z));  appendPrimitive(Dumbbell.getPlate(middlePlateRadius, plateHeight), new Vector3f(x, y + middlePlateDistance, z));  appendPrimitive(Dumbbell.getPlate(middlePlateRadius, plateHeight), new Vector3f(x, y - middlePlateDistance, z));  appendPrimitive(Dumbbell.getPlate(smallPlateRadius, plateHeight), new Vector3f(x, y + smallPlateDistance, z));  appendPrimitive(Dumbbell.getPlate(smallPlateRadius, plateHeight), new Vector3f(x, y - smallPlateDistance, z));  // spikes  double spikeHeight = size \* 0.04;  float spikeDistance = (float)((bodyHeight + spikeHeight) / 2);  appendPrimitive(Dumbbell.getSpike(bodyRadius, spikeHeight), new Vector3f(x, y + spikeDistance, z));  appendPrimitive(Dumbbell.getSpike(bodyRadius, -spikeHeight), new Vector3f(x, y - spikeDistance, z));  }  private void appendPrimitive(Primitive primitive, Vector3f vector) {  TransformGroup bodyGroup = new TransformGroup();  Transform3D transform = new Transform3D();  transform.setTranslation(vector);  bodyGroup.setTransform(transform);  bodyGroup.addChild(primitive);  tg.addChild(bodyGroup);  }  } |

|  |
| --- |
| Dumbbell.java |
| package com.company;  import com.sun.j3d.utils.geometry.\*;  import javax.media.j3d.\*;  import javax.vecmath.\*;  public class Dumbbell {  private final static int flags = Primitive.GENERATE\_NORMALS + Primitive.GENERATE\_TEXTURE\_COORDS;  public static Cylinder getBody(double r, double h) {  return getCylinder(r, h, getBodyAppearence());  }  public static Cylinder getPlate(double r, double h) {  return getCylinder(r, h, getPlateAppearence());  }  public static Cone getSpike(double r, double h) {  return getCone(r, h, getBodyAppearence());  }  private static Cylinder getCylinder(double r, double h, Appearance a) {  return new Cylinder((float)r, (float)h, flags, a);  }  private static Cone getCone(double r, double h, Appearance a) {  return new Cone((float)r, (float)h, flags, a);  }  private static Appearance getBodyAppearence() {  Appearance ap = new Appearance();  Color3f emissive = new Color3f(0.4453f, 0.4453f, 0.4453f);  Color3f ambient = new Color3f(0f, 0f, 0f);  Color3f diffuse = new Color3f(1f, 1f, 1f);  Color3f specular = new Color3f(1f, 1f, 1f);  ap.setMaterial(new Material(ambient, emissive, diffuse, specular, 70f));  return ap;  }  private static Appearance getPlateAppearence() {  Appearance ap = new Appearance();  Color3f emissive = new Color3f(0.03f, 0.03f, 0.03f);  Color3f ambient = new Color3f(0.1f, 0.1f, 0.1f);  Color3f diffuse = new Color3f(0.1f, 0.1f, 0.1f);  Color3f specular = new Color3f(0.1f, 0.1f, 0.1f);  ap.setMaterial(new Material(ambient, emissive, diffuse, specular, 70f));  ap.setMaterial(new Material(ambient, emissive, diffuse, specular, 1.0f));  return ap;  }  } |

**Результат**

|  |  |
| --- | --- |
|  |  |
|  |  |