

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота № 4**

з дисципліни “ МАОКГ”

|  |  |  |
| --- | --- | --- |
| Виконав  студент III курсу  групи КП-82  Залізний Захар Юрійович  (*прізвище, ім’я, по батькові*)  Варіант № 7 |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2021 р.  викладачем  Шкурат О. С.  (*прізвище, ім’я, по батькові*) |

Київ-2021

**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. В

**Результат виконання програми :**



**Код програми:**

|  |
| --- |
| **IceBall.java** |
| package lab4.src;  import com.sun.j3d.utils.geometry.Cone;  import com.sun.j3d.utils.geometry.Cylinder;  import com.sun.j3d.utils.geometry.Primitive;  import com.sun.j3d.utils.geometry.Sphere;  import javax.media.j3d.Appearance;  import javax.media.j3d.Material;  import javax.vecmath.Color3f;  public class IceBall {  public static Cylinder getLeg(float radius, float height, Color3f emmissiveColor) {  Appearance ap = *getXMassBallsAppearence*(emmissiveColor);  return new Cylinder(radius, height, ap);  }  public static Cone getHorn(float radius, float height, Color3f emmissiveColor) {  Appearance ap = *getXMassBallsAppearence*(emmissiveColor);  return new Cone(radius, height, ap);  }  public static Sphere getSphere(float radius, Color3f emissiveColor) {  int primflags = Primitive.*GENERATE\_NORMALS* + Primitive.*GENERATE\_TEXTURE\_COORDS*;  return new Sphere(radius, primflags, *getXMassBallsAppearence*(emissiveColor));  }  private static Appearance getXMassBallsAppearence(Color3f emissive) {  Appearance ap = new Appearance();  Color3f ambient = new Color3f(0.2f, 0.15f, .15f);  Color3f diffuse = new Color3f(1.2f, 1.15f, .15f);  Color3f specular = new Color3f(0.0f, 0.0f, 0.0f);  ap.setMaterial(new Material(ambient, emissive, diffuse, specular, 1.0f));  return ap;  }  } |

|  |
| --- |
| **IceCream.java** |
| package lab4.src;  import javax.media.j3d.\*;  import javax.swing.\*;  import javax.vecmath.Color3f;  import javax.vecmath.Point3d;  import javax.vecmath.Vector3f;  import com.sun.j3d.utils.geometry.Cone;  import com.sun.j3d.utils.geometry.Cylinder;  import com.sun.j3d.utils.geometry.Sphere;  import com.sun.j3d.utils.universe.SimpleUniverse;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  public class IceCream implements ActionListener{  private TransformGroup iceCreamGroup;  private Transform3D trans = new Transform3D();  private float angle = 1.0f;  public static void main(String[] args) {  new IceCream();  }  public IceCream() {  Timer timer = new Timer(50, this);  timer.start();  BranchGroup scene = createSceneGraph();  SimpleUniverse u = new SimpleUniverse();  u.getViewingPlatform().setNominalViewingTransform();  u.addBranchGraph(scene);  }  private BranchGroup createSceneGraph() {  BranchGroup objRoot = new BranchGroup();  iceCreamGroup = new TransformGroup();  iceCreamGroup.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  buildIceCream();  objRoot.addChild(iceCreamGroup);  BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, -0.8, 0.0),100.0);  Color3f light1Color = new Color3f(0.5f, 0.5f, 0.4f);  Vector3f light1Direction = new Vector3f(4.0f, -7.0f, 12.0f);  DirectionalLight light1 = new DirectionalLight(light1Color, light1Direction);  light1.setInfluencingBounds(bounds);  objRoot.addChild(light1);  // встановлюємо навколишнє освітлення  Color3f ambientColor = new Color3f(1.0f, 1.0f, 1.0f);  AmbientLight ambientLightNode = new AmbientLight(ambientColor);  ambientLightNode.setInfluencingBounds(bounds);  objRoot.addChild(ambientLightNode);  return objRoot;  }  private void buildIceCream() {  float width = 0.5f, height = 0.5f;  createHorn(width/2, height, 0, 0, 0, new Color3f(0.541f, 0.302f, 0.117f));  float radius = (float) (width / (2 \* ( 1 + Math.*sin*(Math.*PI*/3))));  createBall(radius, radius, height / 2, .0f, new Color3f(1f, 0.631f, 0.8f));  createBall(radius, -(width - 3 \* radius), height / 2, radius, new Color3f(0.38f, 0.945f, 0.7059f));  createBall(radius, -(width - 3 \* radius), height / 2, -radius, new Color3f(1f, 0.886f, 0.412f));  createBall( 1.2f \* radius, 0, height / 2 + radius, 0, new Color3f(1f, 1f, 1f));  createLeg(width/1.5f, 0.001f, 0, -height/2, 0, new Color3f(0.992f, 0.972f, 0.886f));  }  private void createBall(float radius, float x, float y, float z, Color3f emissive) {  TransformGroup tg = new TransformGroup();  Transform3D transform = new Transform3D();  Sphere cone = IceBall.*getSphere*(radius, emissive);  Vector3f vector = new Vector3f(x, y, z);  transform.setTranslation(vector);  tg.setTransform(transform);  tg.addChild(cone);  iceCreamGroup.addChild(tg);  }  private void createHorn(float radius, float height, float x, float y, float z, Color3f emissive) {  TransformGroup tg = new TransformGroup();  Transform3D transform = new Transform3D();  Cone cone = IceBall.*getHorn*(radius, height, emissive);  Vector3f vector = new Vector3f(x, y, z);  transform.setTranslation(vector);  transform.rotX(Math.*PI*);  tg.setTransform(transform);  tg.addChild(cone);  iceCreamGroup.addChild(tg);  }  private void createLeg(float radius, float height, float x, float y, float z, Color3f emissive) {  TransformGroup tg = new TransformGroup();  Transform3D transform = new Transform3D();  Cylinder cylinder = IceBall.*getLeg*(radius, height, emissive);  Vector3f vector = new Vector3f(x, y, z);  transform.setTranslation(vector);  tg.setTransform(transform);  tg.addChild(cylinder);  iceCreamGroup.addChild(tg);  }  @Override  public void actionPerformed(ActionEvent e) {  trans.rotY(angle);  iceCreamGroup.setTransform(trans);  angle += 0.05;  }  } |