

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  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**

**Завдання**: Виконати анімацію тривимірної сцени за варіантом.

**Варіант:** Анімація павука «чорна вдова» black\_widow.obj. Рух ніжок, пересування по екрану.

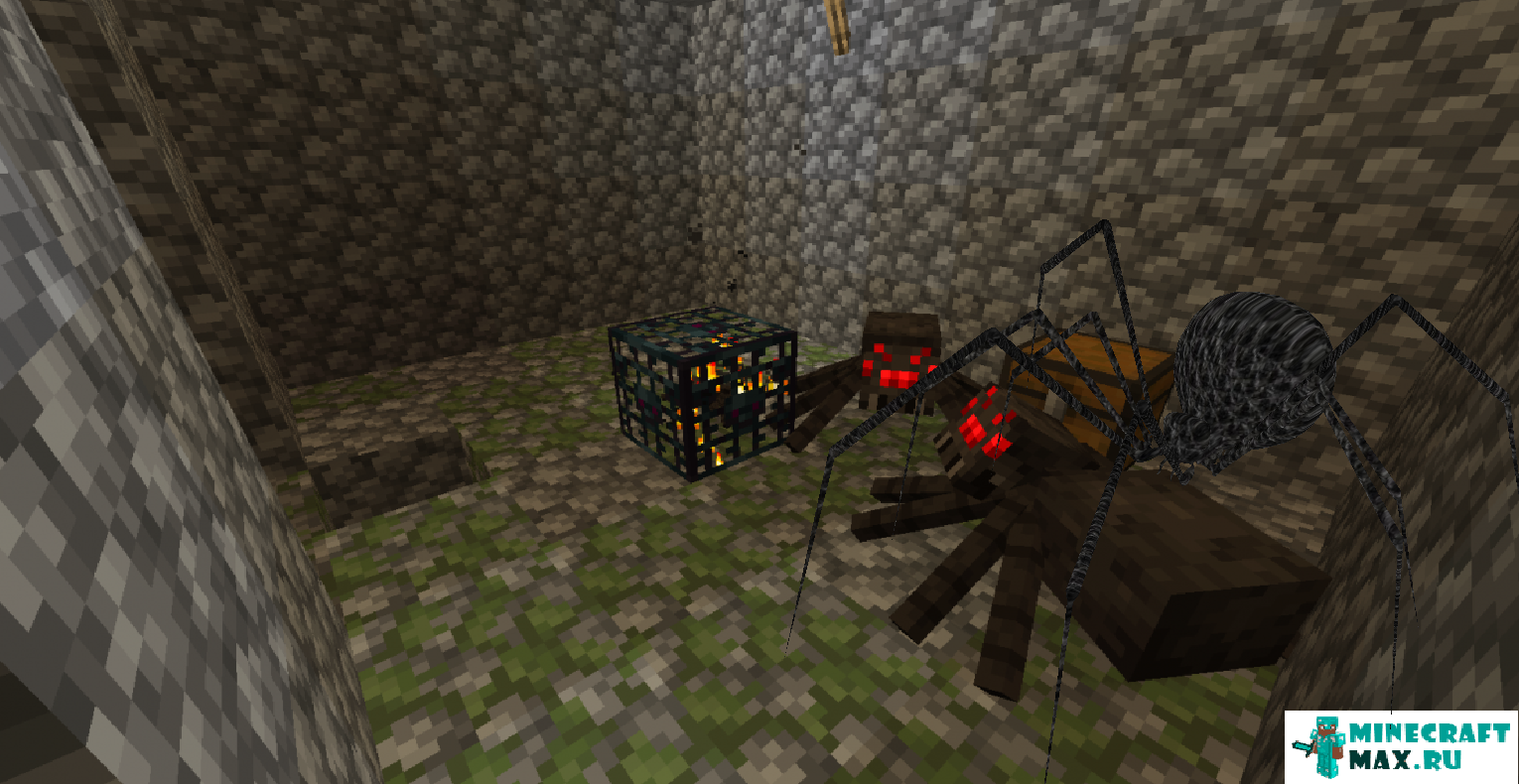
**Лістинг коду програми**

|  |
| --- |
| Main.java |
| package com.company;  import javax.vecmath.\*;  import com.sun.j3d.utils.image.TextureLoader;  import com.sun.j3d.utils.universe.\*;  import javax.media.j3d.\*;  import javax.swing.JFrame;  import com.sun.j3d.loaders.\*;  import com.sun.j3d.loaders.objectfile.\*;  import java.awt.\*;  import java.io.FileReader;  import java.io.IOException;  import java.util.Hashtable;  import java.util.Map;  public class Main extends JFrame{  private static Canvas3D canvas;  private static SimpleUniverse universe;  private static BranchGroup root;  private static TransformGroup blackWidow;  private static Scene spiderScene;  private final static String \_\_ASSETS\_\_ = System.getProperty("user.dir") + "\\src\\assets\\";  public Main() throws IOException {  setUpWindow();  canvas = new Canvas3D(SimpleUniverse.getPreferredConfiguration());  canvas.setDoubleBufferEnable(true);  getContentPane().add(canvas, BorderLayout.CENTER);  SimpleUniverse universe = new SimpleUniverse(canvas);  universe.getViewingPlatform().setNominalViewingTransform();  BranchGroup root = new BranchGroup();  setUpBackground(root, "background.png");  setUpLight(root);  blackWidow = getModelTransformGroup("black\_widow.obj", "black\_widow.jpg", "blkw\_body", "leg", 8);  root.addChild(blackWidow);  root.compile();  universe.addBranchGraph(root);  }  public static void main(String[] args) {  try {  Main window = new Main();  Animation spiderMovement = new Animation(blackWidow);  canvas.addKeyListener(spiderMovement);  window.setVisible(true);  } catch (Exception e) {  System.out.println(e.getMessage());  }  }  private void setUpWindow() {  setTitle("Lab6");  setExtendedState(JFrame.MAXIMIZED\_BOTH);  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  }  private void setUpLight(BranchGroup root) {  BoundingSphere bounds = new BoundingSphere();  bounds.setRadius(100);  DirectionalLight dirlight = new DirectionalLight(new Color3f(1, 1, 1), new Vector3f(-1, -1, -1));  dirlight.setInfluencingBounds(bounds);  root.addChild(dirlight);  AmbientLight amblight = new AmbientLight(new Color3f(1, 1, 1));  amblight.setInfluencingBounds(new BoundingSphere());  root.addChild(amblight);  }  private void setUpBackground(BranchGroup root, String path) {  TextureLoader t = new TextureLoader(\_\_ASSETS\_\_ + path, canvas);  Background background = new Background(t.getImage());  BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);  background.setApplicationBounds(bounds);  background.setImageScaleMode(Background.SCALE\_FIT\_ALL);  root.addChild(background);  }  private void addAppearance(Shape3D shape, String path) {  TextureLoader loader = new TextureLoader(\_\_ASSETS\_\_ + path, "RGP", new Container());  Texture texture = loader.getTexture();  Appearance appearance = new Appearance();  appearance.setTexture(texture);  shape.setAppearance(appearance);  }  public TransformGroup getModelTransformGroup(  String modelPath,  String texturePath,  String bodyGroupName,  String legGroupName,  int legGroupsCount  ) throws IOException {  ObjectFile loader = new ObjectFile(ObjectFile.RESIZE);  spiderScene = loader.load(\_\_ASSETS\_\_ + modelPath);  Hashtable spider = spiderScene.getNamedObjects();  TransformGroup sceneTg = new TransformGroup();  TransformGroup bodyTg = new TransformGroup();  initializeModelPart(bodyTg, sceneTg, spider, bodyGroupName, texturePath);  final int movesDuration = 500;  BoundingSphere bs = new BoundingSphere(new Point3d(0, 0, 0), Double.MAX\_VALUE);  for (int i = 0; i < legGroupsCount; i++) {  TransformGroup legTg = new TransformGroup();  initializeModelPart(legTg, sceneTg, spider, legGroupName + (i + 1), texturePath);  int startTime = i < legGroupsCount / 2  ? i \* movesDuration / legGroupsCount  : (legGroupsCount - i + legGroupsCount / 2 - 1) \* movesDuration / legGroupsCount;  Alpha alpha = new Alpha(Integer.MAX\_VALUE, Alpha.INCREASING\_ENABLE, startTime, 0, movesDuration, 0, 0, 0, 0, 0);  Transform3D rotate = new Transform3D();  rotate.setRotation(new AxisAngle4d(0, 1, 0, 0));  float rotationAngle = (float) Math.PI / 6 \* (i < legGroupsCount / 2 ? 1 : -1);  RotationInterpolator interpolator = new RotationInterpolator(alpha, legTg, rotate, 0, rotationAngle);  interpolator.setSchedulingBounds(bs);  legTg.addChild(interpolator);  }  Transform3D startTransformation = new Transform3D();  Transform3D rotY = new Transform3D();  rotY.rotY(Math.PI / 3);  Transform3D rotX = new Transform3D();  rotX.rotX(Math.PI / 6);  Transform3D scale = new Transform3D();  scale.setScale(0.45);  startTransformation.mul(rotX);  startTransformation.mul(rotY);  startTransformation.mul(scale);  TransformGroup rootTg = new TransformGroup(startTransformation);  rootTg.addChild(sceneTg);  rootTg.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);  return rootTg;  }  private void initializeModelPart(TransformGroup part, TransformGroup sceneTg, Hashtable table, String groupName, String texturePath) {  part.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);  Shape3D body = (Shape3D)table.get(groupName);  addAppearance(body, texturePath);  part.addChild(body.cloneTree());  sceneTg.addChild(part);  }  } |

|  |
| --- |
| Animation.java |
| package com.company;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.awt.event.KeyEvent;  import java.awt.event.KeyListener;  import javax.media.j3d.\*;  import javax.swing.Timer;  import javax.vecmath.\*;  class KeyCodes {  final static int ROTATE\_UP = 38;  final static int ROTATE\_DOWN = 40;  final static int ROTATE\_LEFT = 37;  final static int ROTATE\_RIGHT = 39;  final static int MOVE\_UP = 87;  final static int MOVE\_DOWN = 83;  final static int MOVE\_LEFT = 65;  final static int MOVE\_RIGHT = 68;  }  public class Animation implements ActionListener, KeyListener {  private TransformGroup group;  private Transform3D transform3D = new Transform3D();  private float x = 0;  private float y = 0;  private float dx = 0.02f;  private float dy = 0.02f;  private float da = 0.05f;  private boolean mu = false;  private boolean md = false;  private boolean ml = false;  private boolean mr = false;  private boolean ru = false;  private boolean rd = false;  private boolean rl = false;  private boolean rr = false;  Animation(TransformGroup group) {  this.group = group;  this.group.getTransform(this.transform3D);  Timer timer = new Timer(20, this);  timer.start();  }  private void Move() {  if (ml) x -= dx;  if (mr) x += dx;  if (mu) y += dy;  if (md) y -= dy;  transform3D.setTranslation(new Vector3f(x, y, 0));  Transform3D rotation = new Transform3D();  if (ru) rotation.rotX(-da);  if (rd) rotation.rotX(da);  if (rl) rotation.rotY(da);  if (rr) rotation.rotY(-da);  transform3D.mul(rotation);  group.setTransform(transform3D);  }  @Override  public void actionPerformed(ActionEvent e) {  Move();  }  @Override  public void keyPressed(KeyEvent ev) {  switch (ev.getKeyCode()) {  case KeyCodes.MOVE\_UP -> mu = true;  case KeyCodes.MOVE\_DOWN -> md = true;  case KeyCodes.MOVE\_LEFT -> ml = true;  case KeyCodes.MOVE\_RIGHT -> mr = true;  case KeyCodes.ROTATE\_UP -> ru = true;  case KeyCodes.ROTATE\_DOWN -> rd = true;  case KeyCodes.ROTATE\_LEFT -> rl = true;  case KeyCodes.ROTATE\_RIGHT -> rr = true;  }  }  @Override  public void keyTyped(KeyEvent e) {}  @Override  public void keyReleased(KeyEvent ev) {  switch (ev.getKeyCode()) {  case KeyCodes.MOVE\_UP -> mu = false;  case KeyCodes.MOVE\_DOWN -> md = false;  case KeyCodes.MOVE\_LEFT -> ml = false;  case KeyCodes.MOVE\_RIGHT -> mr = false;  case KeyCodes.ROTATE\_UP -> ru = false;  case KeyCodes.ROTATE\_DOWN -> rd = false;  case KeyCodes.ROTATE\_LEFT -> rl = false;  case KeyCodes.ROTATE\_RIGHT -> rr = false;  }  }  } |

**Результат**

****

****