

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“

**Тема:** “Анімація тривимірних об’єктів “

|  |  |  |  |
| --- | --- | --- | --- |
| Виконав  студент III курсу  групи КП-83  Василець Данило Андрійович  (*прізвище, ім’я, по батькові*)  варіант № 2 |  |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладачем  Шкурат Оксаною Сергіївною (*прізвище, ім’я, по батькові*) |

Київ 2021

**Варіант завдання**

**Завдання**:

Виконати анімацію тривимірної сцени за варіантом.

**Варіант: 2**

Анімація автомобіля car.obj. Рух коліс, пересування по екрану, обов’язкові повороти авто.

**Лістинг коду програми**

**Car.java**

**public** **class** Car **extends** JFrame{

**public** Canvas3D myCanvas3D;

**public** Car(){

**this**.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        myCanvas3D = **new** Canvas3D(SimpleUniverse.getPreferredConfiguration());

        SimpleUniverse simpleUniv = **new** SimpleUniverse(myCanvas3D);

        simpleUniv.getViewingPlatform().setNominalViewingTransform();

        createSceneGraph(simpleUniv);

        Utils\_anim.addLight(simpleUniv);

        OrbitBehavior ob = **new** OrbitBehavior(myCanvas3D);

        ob.setSchedulingBounds(**new** BoundingSphere(**new** Point3d(0.0,0.0,0.0),Double.MAX\_VALUE));

        simpleUniv.getViewingPlatform().setViewPlatformBehavior(ob);

        setTitle("Car animation");

        setSize(1700,1700);

        getContentPane().add("Center", myCanvas3D);

        setVisible(**true**);

    }

**public** **void** createSceneGraph(SimpleUniverse su){

        ObjectFile f = **new** ObjectFile(ObjectFile.RESIZE);

        BoundingSphere bs = **new** BoundingSphere(**new** Point3d(0.0,0.0,0.0),Double.MAX\_VALUE);

        BranchGroup carBranchGroup = **new** BranchGroup();

        Background carBackground = **new** Background();

        Scene carScene = **null**;

**try**{

            carScene = f.load("D:**\\**школа**\\**3 курс 2 сем**\\**маокг**\\**labs**\\**lab6**\\**resс**\\**models**\\**car.obj");

        }

**catch** (Exception e){

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

        }

        Hashtable carNamedObjects = carScene.getNamedObjects();

        TextureLoader textureCarLoad = **new** TextureLoader("D:**\\**школа**\\**3 курс 2 сем**\\**маокг**\\**labs**\\**lab6**\\**resс**\\**car.png", **null**);

        ImageComponent2D textureIm = textureCarLoad.getImage();

        Texture2D carTexture = **new** Texture2D(Texture2D.BASE\_LEVEL,Texture2D.RGB,

                textureIm.getWidth(),

                textureIm.getHeight());

        carTexture.setImage(0, textureIm);

        Appearance textureApp = **new** Appearance();

        textureApp.setTexture(carTexture);

        TextureAttributes textureAttr = **new** TextureAttributes();

        textureAttr.setTextureMode(TextureAttributes.REPLACE);

        textureApp.setTextureAttributes(textureAttr);

        Material mat = **new** Material();

        mat.setShininess(100.0f);

        mat.setSpecularColor(**new** Color3f(.7f,0.5f,0.1f));

        mat.setDiffuseColor(**new** Color3f(.6f,0.4f,0.2f));

        mat.setAmbientColor(**new** Color3f(.5f,0.3f,0.3f));

        mat.setEmissiveColor(**new** Color3f(.4f,0.2f,0.4f));

        textureApp.setMaterial(mat);

*// start animation*

        Transform3D startTransformation = **new** Transform3D();

        startTransformation.setScale(1.0/4);

        Transform3D combinedStartTransformation = **new** Transform3D();

        combinedStartTransformation.rotY(-3\*Math.PI/2);

        combinedStartTransformation.mul(startTransformation);

        TransformGroup carStartTransformGroup = **new** TransformGroup(combinedStartTransformation);

*// wheels*

**int** movesCount = 100;

**int** movesDuration = 500;

**int** startTime = 0;

*// wheel 1*

        Alpha w1RotAlpha = **new** Alpha(movesCount, Alpha.INCREASING\_ENABLE, startTime, 0, movesDuration,0,0,0,0,0);

        Shape3D wheel1 = (Shape3D) carNamedObjects.get("wheel1");

        TransformGroup wheelTG1 = **new** TransformGroup();

        wheelTG1.addChild(wheel1.cloneTree());

        Transform3D wheelRotAxis = **new** Transform3D();

        wheelRotAxis.set(**new** Vector3d(0, -0.1, -0.644));

        wheelRotAxis.setRotation(**new** AxisAngle4d(0, 0, -0.1, Math.PI/2));

        RotationInterpolator wheel1rot = **new** RotationInterpolator(w1RotAlpha, wheelTG1, wheelRotAxis,(**float**) 0.0f, (**float**) Math.PI\*2); *// Math.PI\*2*

        wheel1rot.setSchedulingBounds(bs);

        wheelTG1.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);

        wheelTG1.addChild(wheel1rot);

*// wheel 2*

        Alpha w2RotAlpha = **new** Alpha(movesCount, Alpha.INCREASING\_ENABLE, startTime, 0, movesDuration,0,0,0,0,0);

        Shape3D wheel2 = (Shape3D) carNamedObjects.get("wheel2");

        TransformGroup wheelTG2 = **new** TransformGroup();

        wheelTG2.addChild(wheel2.cloneTree());

        Transform3D wheelRotAxis2 = **new** Transform3D();

        wheelRotAxis2.set(**new** Vector3d(0, -0.101, 0.52));

        wheelRotAxis2.setRotation(**new** AxisAngle4d(0, 0, -0.1, Math.PI/2));

        RotationInterpolator wheel2rot = **new** RotationInterpolator(w2RotAlpha, wheelTG2, wheelRotAxis2,(**float**) 0.0f, (**float**) Math.PI\*2); *// Math.PI\*2*

        wheel2rot.setSchedulingBounds(bs);

        wheelTG2.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);

        wheelTG2.addChild(wheel2rot);

*// wheel 3*

        Alpha w3RotAlpha = **new** Alpha(movesCount, Alpha.INCREASING\_ENABLE, startTime, 0, movesDuration,0,0,0,0,0);

        Shape3D wheel3 = (Shape3D) carNamedObjects.get("wheel3");

        TransformGroup wheelTG3 = **new** TransformGroup();

        wheelTG3.addChild(wheel3.cloneTree());

        Transform3D wheelRotAxis3 = **new** Transform3D();

        wheelRotAxis3.set(**new** Vector3d(0, -0.1, -0.625));

        wheelRotAxis3.setRotation(**new** AxisAngle4d(0, 0, -0.1, Math.PI/2));

        RotationInterpolator wheel3rot = **new** RotationInterpolator(w3RotAlpha, wheelTG3, wheelRotAxis3,(**float**) 0.0f, (**float**) Math.PI\*2); *// Math.PI\*2*

        wheel3rot.setSchedulingBounds(bs);

        wheelTG3.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);

        wheelTG3.addChild(wheel3rot);

*// wheel 4*

        Alpha w4RotAlpha = **new** Alpha(movesCount, Alpha.INCREASING\_ENABLE, startTime, 0, movesDuration,0,0,0,0,0);

        Shape3D wheel4 = (Shape3D) carNamedObjects.get("wheel4");

        TransformGroup wheelTG4 = **new** TransformGroup();

        wheelTG4.addChild(wheel4.cloneTree());

        Transform3D wheelRotAxis4 = **new** Transform3D();

        wheelRotAxis4.set(**new** Vector3d(0, -0.101, 0.535));

        wheelRotAxis4.setRotation(**new** AxisAngle4d(0, 0, -0.1, Math.PI/2));

        RotationInterpolator wheel4rot = **new** RotationInterpolator(w4RotAlpha, wheelTG4, wheelRotAxis4,(**float**) 0.0f, (**float**) Math.PI\*2); *// Math.PI\*2*

        wheel4rot.setSchedulingBounds(bs);

        wheelTG4.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);

        wheelTG4.addChild(wheel4rot);

*//car body*

        TransformGroup sceneGroup = **new** TransformGroup();

        sceneGroup.addChild(wheelTG1);

        sceneGroup.addChild(wheelTG2);

        sceneGroup.addChild(wheelTG3);

        sceneGroup.addChild(wheelTG4);

        TransformGroup tgBody = **new** TransformGroup();

        Shape3D carBodyShape = (Shape3D) carNamedObjects.get("platinum1");

        carBodyShape.setAppearance(textureApp);

        tgBody.addChild(carBodyShape.cloneTree());

        sceneGroup.addChild(tgBody.cloneTree());

        TransformGroup whiteTransXformGroup = Utils\_anim.translate(

                carStartTransformGroup,

**new** Vector3f(0.0f,-0.2f,0.5f));

        TransformGroup whiteRotXformGroup = Utils\_anim.rotate(whiteTransXformGroup, **new** Alpha(10,5000));

        carBranchGroup.addChild(whiteRotXformGroup);

        carStartTransformGroup.addChild(sceneGroup);

        BoundingSphere bounds = **new** BoundingSphere(**new** Point3d(100.0,300.0,100.0),Double.MAX\_VALUE);

        BufferedImage BackImg = **null**;

**try** {

            BackImg = ImageIO.read(**new** File("resс/back.jpg"));

        } **catch** (IOException e) {

        }        TextureLoader starsTexture = **new** TextureLoader(BackImg, **this**);

        Dimension dim = java.awt.Toolkit.getDefaultToolkit().getScreenSize();

**double** dim\_height = dim.getHeight();

**double** dim\_width = dim.getWidth();

        Background car\_back\_img = **new** Background(starsTexture.getScaledImage((**int**) dim\_width, (**int**) dim\_height));

        car\_back\_img.setApplicationBounds(bounds);

        carBranchGroup.addChild(car\_back\_img);

        carBackground.setApplicationBounds(bounds);

        carBranchGroup.addChild(carBackground);

        carBranchGroup.compile();

        su.addBranchGraph(carBranchGroup);

    }

}

**Utils\_anim.java**

**public** **class** Utils\_anim {

**static** 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;

    }

**static** TransformGroup rotate(Node node, Alpha alpha){

        TransformGroup xformGroup = **new** TransformGroup();

        xformGroup.setCapability(

                TransformGroup.ALLOW\_TRANSFORM\_WRITE);

        RotationInterpolator interpolator =

**new** RotationInterpolator(alpha,xformGroup);

        interpolator.setSchedulingBounds(**new** BoundingSphere(

**new** Point3d(0.0,0.0,0.0),1.0));

        xformGroup.addChild(interpolator);

        xformGroup.addChild(node);

**return** xformGroup;

    }

**public** **static** **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**

