SimpleMeshView.java

```
 /*
  * Copyright (c) 2013, Oracle and/or its affiliates. All rights reserved.
  * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
  *
  * This code is free software; you can redistribute it and/or modify it
  * under the terms of the GNU General Public License version 2 only, as
  * published by the Free Software Foundation.  Oracle designates this
  * particular file as subject to the "Classpath" exception as provided
  * by Oracle in the LICENSE file that accompanied this code.
  *
  * This code is distributed in the hope that it will be useful, but WITHOUT
  * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
  * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
  * version 2 for more details (a copy is included in the LICENSE file that
  * accompanied this code).
  *
  * You should have received a copy of the GNU General Public License version
  * 2 along with this work; if not, write to the Free Software Foundation,
  * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
  *
  * Please contact Oracle, 500 Oracle Parkway, Redwood Shores, CA 94065 USA
  * or visit www.oracle.com if you need additional information or have any
  * questions.
  */

import javafx.application.Application;
import javafx.scene.*;
import javafx.scene.paint.Color;
import javafx.scene.paint.PhongMaterial;
import javafx.scene.shape.*;
import javafx.stage.Stage;

public class SimpleMeshViewer extends Application {

    Group root;
    PointLight pointLight;
    MeshView meshView;
    TriangleMesh triMesh;
    PhongMaterial material;
    final static float minX = -10;
    final static float minY = -10;
    final static float maxX = 10;
    final static float maxY = 10;

    static TriangleMesh buildTriangleMesh(int subDivX, int subDivY, float scale) {

        final int pointSize = 3;
        final int texCoordSize = 2;
        // 3 point indices and 3 texCoord indices per triangle
        final int faceSize = 6;
        int numDivX = subDivX + 1;
        int numVerts = (subDivY + 1) * numDivX;
        float points[] = new float[numVerts * pointSize];
        float texCoords[] = new float[numVerts * texCoordSize];
        int faceCount = subDivX * subDivY * 2;
        int faces[] = new int[faceCount * faceSize];

        // Create points and texCoords
        for (int y = 0; y <= subDivY; y++) {
            float dy = (float) y / subDivY;
            double fy = (1 - dy) * minY + dy * maxY;

            for (int x = 0; x <= subDivX; x++) {
                float dx = (float) x / subDivX;
                double fx = (1 - dx) * minX + dx * maxX;

                int index = y * numDivX * pointSize + (x * pointSize);
```

```
                points[index] = (float) fx * scale;
                points[index + 1] = (float) fy * scale;
                points[index + 2] = 0.0f;

                index = y * numDivX * texCoordSize + (x * texCoordSize);
                texCoords[index] = dx;
                texCoords[index + 1] = dy;
            }
        }

        // Create faces
        for (int y = 0; y < subDivY; y++) {
            for (int x = 0; x < subDivX; x++) {
                int p00 = y * numDivX + x;
                int p01 = p00 + 1;
                int p10 = p00 + numDivX;
                int p11 = p10 + 1;
                int tc00 = y * numDivX + x;
                int tc01 = tc00 + 1;
                int tc10 = tc00 + numDivX;
                int tc11 = tc10 + 1;

                int index = (y * subDivX * faceSize + (x * faceSize)) * 2;
                faces[index + 0] = p00;
                faces[index + 1] = tc00;
                faces[index + 2] = p10;
                faces[index + 3] = tc10;
                faces[index + 4] = p11;
                faces[index + 5] = tc11;

                index += faceSize;
                faces[index + 0] = p11;
                faces[index + 1] = tc11;
                faces[index + 2] = p01;
                faces[index + 3] = tc01;
                faces[index + 4] = p00;
                faces[index + 5] = tc00;
            }
        }

        TriangleMesh triangleMesh = new TriangleMesh(points, texCoords, faces);

        return triangleMesh;
    }

    private Group buildScene() {
        triMesh = buildTriangleMesh(5, 5, 20);

        material = new PhongMaterial();
        material.setDiffuseColor(Color.GOLD);
        material.setSpecularColor(Color.rgb(30, 30, 30));
        meshView = new MeshView(triMesh);
        meshView.setTranslateX(400);
        meshView.setTranslateY(400);
        meshView.setTranslateZ(20);
        meshView.setMaterial(material);

        //Set Wireframe mode
        meshView.setDrawMode(DrawMode.LINE);

        meshView.setCullFace(CullFace.BACK);

        pointLight = new PointLight(Color.ANTIQUEWHITE);
        pointLight.setTranslateX(150);
        pointLight.setTranslateY(-100);
        pointLight.setTranslateZ(-1000);

        root = new Group(meshView, pointLight);
        return root;
    }
```

```
    private PerspectiveCamera addCamera(Scene scene) {
        PerspectiveCamera perspectiveCamera = new PerspectiveCamera();
        scene.setCamera(perspectiveCamera);
        return perspectiveCamera;
    }

    @Override
    public void start(Stage primaryStage) {
        Scene scene = new Scene(buildScene(), 800, 800, true);
        scene.setFill(Color.rgb(10, 10, 40));
        addCamera(scene);
        primaryStage.setTitle("SimpleMeshViewer");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public static void main(String[] args) {
        // Remove this line once dirtyopts bug is fixed for 3D primitive
        System.setProperty("prism.dirtyopts", "false");
        launch(args);
    }
}
```