class MainClass

import javax.media.j3d.Appearance;

import javax.media.j3d.Background;

import javax.media.j3d.BoundingSphere;

import javax.media.j3d.BranchGroup;

import javax.media.j3d.Canvas3D;

import javax.media.j3d.DirectionalLight;

import javax.media.j3d.Material;

import javax.media.j3d.Shape3D;

import javax.media.j3d.Transform3D;

import javax.media.j3d.TransformGroup;

import javax.vecmath.Color3f;

import javax.vecmath.Point3d;

import javax.vecmath.Quat4d;

import javax.vecmath.Vector3f;

import javax.media.j3d.TransparencyAttributes;

import com.sun.j3d.utils.behaviors.mouse.MouseRotate;

import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;

import com.sun.j3d.utils.behaviors.mouse.MouseZoom;

import com.sun.j3d.utils.geometry.Cylinder;

import com.sun.j3d.utils.geometry.Sphere;

import com.sun.j3d.utils.universe.SimpleUniverse;

import javax.swing.\*;

import java.util.\*;

import java.util.List;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.io.File;

import java.io.FileFilter;

import java.awt.\*;

import java.awt.event.\*;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.Date;

public class ApenPostProc extends JFrame {

private static final int ClarityParaSph = 50;

private static final int ClarityParaCylin = 20;

private JPanel NorthToolPanel, WestToolPanel, NTPLeftPanel, NTPRightPanel;

private JFrame homeframe;

private JButton AnimateTimeSeiresButton, CastSphereButton, LoadStatusButton, LoadTimeSeriesButton,

LoadForceChainButton, VelocityFieldButton, CleanUpButton;

private JButton CRDcastRoundButton;

private JLabel FileSettingLabel, RadiusLabel, ParticleNumberLabel, ScaleLabel, XCoorLabel, YCoorLabel, ZCoorLabel,

TransparencyLabel, ForceNumberLabel;

private JTextArea infoText;

private JScrollPane scroll;

private FileDialog fdOpen, fdSave;

private JFileChooser fdOpenFolder;

private File osFile, osFolder;

private int NumOfFilesTimeSeries = 0;

private int NumOfFilesForceChain = 0;

private int IndexVelocity = 0;

private Dialog dOpen, dRound;

private JButton J3DButton, J2DButton;

private int index3D2D = 3;

private JLabel CRDradiusl, CRDxl, CRDyl, CRDzl;

private JTextField RadiusText, ParticleNumberText, ScaleText, XCoorText, YCoorText, ZCoorText, TransparencyText,

ForceNumberText, CRDradiusc, CRDxc, CRDyc, CRDzc;

private GraphicsConfiguration config;

private Canvas3D canvasJ3D;

private SimpleUniverse simpUniv;

private BranchGroup bg, bgUp;

private BoundingSphere bounds;

private TransformGroup tgUp, tgScene, tg, tgCast, tgLoadStatus, tgLoadForceChain, tgCastSphere;

private DirectionalLight dl;

private ArrayList<Sphere> ArrListSphF1 = new ArrayList<Sphere>();

private ArrayList<TransformGroup> ArrListTransGroupF1 = new ArrayList<TransformGroup>();

private ArrayList<Transform3D> ArrListTrans3DF1 = new ArrayList<Transform3D>();

private ArrayList<Vector3f> ArrListVectorF1 = new ArrayList<Vector3f>();

private ArrayList<Cylinder> ArrListCylinderF2 = new ArrayList<Cylinder>();

private ArrayList<TransformGroup> ArrListTransGroupF2 = new ArrayList<TransformGroup>();

private ArrayList<Transform3D> ArrListTrans3DF2 = new ArrayList<Transform3D>();

private ArrayList<Vector3f> ArrListVectorF2 = new ArrayList<Vector3f>();

private Integer NumOfParticles = 0;

private ArrayList<Float> ColumnRadiusF1 = new ArrayList<Float>();

private ArrayList<ArrayList<Float>> radius = new ArrayList<ArrayList<Float>>();

private ArrayList<Float> ColumnXCoorF1 = new ArrayList<Float>();

private ArrayList<ArrayList<Float>> xcoor = new ArrayList<ArrayList<Float>>();

private ArrayList<Float> ColumnYCoorF1 = new ArrayList<Float>();

private ArrayList<ArrayList<Float>> ycoor = new ArrayList<ArrayList<Float>>();

private ArrayList<Float> ColumnZCoorF1 = new ArrayList<Float>();

private ArrayList<ArrayList<Float>> zcoor = new ArrayList<ArrayList<Float>>();

private ArrayList<Float> ColumnVelocity = new ArrayList<Float>();

private ArrayList<ArrayList<Float>> velocity = new ArrayList<ArrayList<Float>>();

private ArrayList<Float> ColumnX1CoorF2 = new ArrayList<Float>();

private ArrayList<Float> ColumnY1CoorF2 = new ArrayList<Float>();

private ArrayList<Float> ColumnZ1CoorF2 = new ArrayList<Float>();

private ArrayList<Float> ColumnX2CoorF2 = new ArrayList<Float>();

private ArrayList<Float> ColumnY2CoorF2 = new ArrayList<Float>();

private ArrayList<Float> ColumnZ2CoorF2 = new ArrayList<Float>();

private ArrayList<Float> ColumnForceValueF2 = new ArrayList<Float>();

private Font Calibri24 = new Font("Calibri", Font.PLAIN, 24);

private Font Calibri21 = new Font("Calibri", Font.PLAIN, 21);

private Font Calibri18 = new Font("Calibri", Font.PLAIN, 18);

public static void main(String[] args) throws Exception {

ApenPostProc demo = new ApenPostProc();

JFrame.setDefaultLookAndFeelDecorated(true);

JDialog.setDefaultLookAndFeelDecorated(true);

}

public ApenPostProc() {

homeframe = new JFrame("ApenPost");

GridBagLayout gblayout = new GridBagLayout();

homeframe.setLayout(gblayout);

setup3D();

setupSwing();

setupListener();

setupLayout();

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

homeframe.setSize((int) (dim.width \* 0.6), (int) (dim.height \* 0.66));

homeframe.setLocation((int) (dim.width \* 0.2), (int) (dim.height \* 0.2));

homeframe.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

homeframe.setVisible(true);

}

public void setup3D() {

config = SimpleUniverse.getPreferredConfiguration();

canvasJ3D = new Canvas3D(config);

simpUniv = new SimpleUniverse(canvasJ3D);

simpUniv.getViewer().getView().setFrontClipDistance(0.001);

simpUniv.getViewer().getView().setBackClipDistance(3000);

simpUniv.getViewingPlatform().setNominalViewingTransform();

bgUp = new BranchGroup();

bounds = new BoundingSphere(new Point3d(0, 0, 0), 1000);

Color3f bgColor = new Color3f(new Color(205, 205, 205));

Background background = new Background(bgColor);

background.setApplicationBounds(bounds);

bgUp.addChild(background);

Color3f lcolor = new Color3f(1f, 1f, 1f);

Vector3f lvector = new Vector3f(-1f, -1f, -1f);

dl = new DirectionalLight(lcolor, lvector);

dl.setInfluencingBounds(bounds);

bgUp.addChild(dl);

bg = new BranchGroup();

bgUp.addChild(bg);

tgUp = new TransformGroup();

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

tgUp.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);

bg.addChild(tgUp);

MouseRotate mouserotate = new MouseRotate();

mouserotate.setTransformGroup(tgUp);

bgUp.addChild(mouserotate);

mouserotate.setSchedulingBounds(bounds);

mouserotate.setFactor(0.005);

MouseTranslate mousetranslate = new MouseTranslate();

mousetranslate.setTransformGroup(tgUp);

bgUp.addChild(mousetranslate);

mousetranslate.setSchedulingBounds(bounds);

mousetranslate.setFactor(0.0002);

MouseZoom mousewheelzoom = new MouseZoom();

mousewheelzoom.setTransformGroup(tgUp);

bgUp.addChild(mousewheelzoom);

mousewheelzoom.setSchedulingBounds(bounds);

mousewheelzoom.setFactor(0.001);

tgScene = new TransformGroup();

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

tgScene.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);

tgUp.addChild(tgScene);

bgUp.setCapability(BranchGroup.ALLOW\_CHILDREN\_EXTEND);

bgUp.setCapability(BranchGroup.ALLOW\_CHILDREN\_WRITE);

bgUp.setCapability(BranchGroup.ALLOW\_CHILDREN\_READ);

bg.setCapability(BranchGroup.ALLOW\_CHILDREN\_EXTEND);

bg.setCapability(BranchGroup.ALLOW\_CHILDREN\_WRITE);

bg.setCapability(BranchGroup.ALLOW\_CHILDREN\_READ);

bg.setCapability(BranchGroup.ALLOW\_DETACH);

tgUp.setCapability(TransformGroup.ALLOW\_CHILDREN\_EXTEND);

tgUp.setCapability(TransformGroup.ALLOW\_CHILDREN\_WRITE);

tgUp.setCapability(TransformGroup.ALLOW\_CHILDREN\_READ);

dl.setCapability(DirectionalLight.ALLOW\_COLOR\_WRITE);

dl.setCapability(DirectionalLight.ALLOW\_STATE\_WRITE);

simpUniv.addBranchGraph(bgUp);

}

public void setupSwing() {

fdOpen = new FileDialog(homeframe, "Open", FileDialog.LOAD);

fdOpenFolder = new JFileChooser();

fdOpenFolder.setFileSelectionMode(JFileChooser.DIRECTORIES\_ONLY);

fdSave = new FileDialog(homeframe, "Save", FileDialog.SAVE);

AnimateTimeSeiresButton = new JButton("Animate");

AnimateTimeSeiresButton.setBackground(new Color(178, 209, 152));

AnimateTimeSeiresButton.setFont(Calibri21);

CastSphereButton = new JButton("Cast Sphere");

CastSphereButton.setBackground(new Color(178, 209, 152));

CastSphereButton.setFont(Calibri21);

LoadStatusButton = new JButton("Load Status");

LoadStatusButton.setBackground(new Color(178, 209, 152));

LoadStatusButton.setFont(Calibri21);

LoadTimeSeriesButton = new JButton("Load Time Series");

LoadTimeSeriesButton.setBackground(new Color(178, 209, 152));

LoadTimeSeriesButton.setFont(Calibri21);

LoadForceChainButton = new JButton("Load Force Chain");

LoadForceChainButton.setBackground(new Color(178, 209, 152));

LoadForceChainButton.setFont(Calibri21);

VelocityFieldButton = new JButton("Velocity Field");

VelocityFieldButton.setBackground(Color.GRAY);

VelocityFieldButton.setFont(Calibri21);

CleanUpButton = new JButton("Clean Up");

CleanUpButton.setBackground(new Color(178, 209, 152));

CleanUpButton.setFont(Calibri21);

infoText = new JTextArea("");

scroll = new JScrollPane(infoText);

scroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);

scroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_ALWAYS);

infoText.setFont(Calibri18);

infoText.setEditable(false);

infoText.setLineWrap(false);

infoText.setWrapStyleWord(false);

infoText.setBackground(new Color(252, 252, 252));

JFrame j0 = new JFrame();

dOpen = new Dialog(j0, "File Setting Area");

dOpen.setSize(200, 200);

dOpen.setResizable(false);

dOpen.setLayout(new GridBagLayout());

J3DButton = new JButton("3D");

J3DButton.setFont(Calibri24);

J3DButton.setBackground(new Color(22, 79, 79));

J2DButton = new JButton("2D");

J2DButton.setFont(Calibri24);

J2DButton.setBackground(new Color(151, 235, 235));

FileSettingLabel = new JLabel("File Setting", JLabel.CENTER);

FileSettingLabel.setFont(Calibri21);

RadiusLabel = new JLabel("Radius Col:", JLabel.CENTER);

RadiusLabel.setFont(Calibri18);

ParticleNumberLabel = new JLabel("Par Num:", JLabel.CENTER);

ParticleNumberLabel.setFont(Calibri18);

ScaleLabel = new JLabel("Scale:", JLabel.CENTER);

ScaleLabel.setFont(Calibri18);

XCoorLabel = new JLabel("X Col:", JLabel.CENTER);

XCoorLabel.setFont(Calibri18);

YCoorLabel = new JLabel("Y Col:", JLabel.CENTER);

YCoorLabel.setFont(Calibri18);

ZCoorLabel = new JLabel("Z Col:", JLabel.CENTER);

ZCoorLabel.setFont(Calibri18);

TransparencyLabel = new JLabel("Transparency:", JLabel.CENTER);

TransparencyLabel.setFont(Calibri18);

ForceNumberLabel = new JLabel("Force Num:", JLabel.CENTER);

ForceNumberLabel.setFont(Calibri18);

RadiusText = new JTextField("");

ParticleNumberText = new JTextField("");

ScaleText = new JTextField("");

XCoorText = new JTextField("");

YCoorText = new JTextField("");

ZCoorText = new JTextField("");

TransparencyText = new JTextField("");

ForceNumberText = new JTextField("");

JFrame j1 = new JFrame();

dRound = new Dialog(j1, "Round Setting");

dRound.setSize(200, 200);

dRound.setResizable(false);

dRound.setLayout(new GridBagLayout());

CRDradiusl = new JLabel("Radius:", JLabel.CENTER);

CRDradiusl.setFont(Calibri18);

CRDxl = new JLabel("x:", JLabel.CENTER);

CRDxl.setFont(Calibri18);

CRDyl = new JLabel("y:", JLabel.CENTER);

CRDyl.setFont(Calibri18);

CRDzl = new JLabel("z:", JLabel.CENTER);

CRDzl.setFont(Calibri18);

CRDcastRoundButton = new JButton("Cast");

CRDcastRoundButton.setBackground(new Color(150, 205, 205));

CRDcastRoundButton.setFont(Calibri24);

CRDradiusc = new JTextField("");

CRDxc = new JTextField("");

CRDyc = new JTextField("");

CRDzc = new JTextField("");

dRound.add(CRDradiusl, new GBC(0, 0).setFill(GBC.BOTH).setWeight(0, 0).setInsets(25, 20, 0, 0));

dRound.add(CRDradiusc, new GBC(1, 0).setFill(GBC.BOTH).setWeight(1, 0).setInsets(25, 5, 0, 20));

dRound.add(CRDxl, new GBC(0, 1).setFill(GBC.BOTH).setWeight(0, 0).setInsets(0, 20, 0, 0));

dRound.add(CRDxc, new GBC(1, 1).setFill(GBC.BOTH).setWeight(0, 0).setInsets(0, 5, 0, 20));

dRound.add(CRDyl, new GBC(0, 2).setFill(GBC.BOTH).setWeight(0, 0).setInsets(0, 20, 0, 0));

dRound.add(CRDyc, new GBC(1, 2).setFill(GBC.BOTH).setWeight(0, 0).setInsets(0, 5, 0, 20));

dRound.add(CRDzl, new GBC(0, 3).setFill(GBC.BOTH).setWeight(0, 0).setInsets(0, 20, 0, 0));

dRound.add(CRDzc, new GBC(1, 3).setFill(GBC.BOTH).setWeight(0, 0).setInsets(0, 5, 0, 20));

dRound.add(CRDcastRoundButton, new GBC(0, 4, 2, 1).setInsets(5, 0, 25, 0));

}

public void setupLayout() {

NorthToolPanel = new JPanel();

NorthToolPanel.setBackground(new Color(235, 235, 235));

homeframe.add(NorthToolPanel, new GBC(1, 0).setFill(GBC.BOTH).setIpad(0, -20).setWeight(0, 0));

WestToolPanel = new JPanel();

WestToolPanel.setBackground(new Color(235, 235, 235));

homeframe.add(WestToolPanel, new GBC(0, 0, 1, 3).setFill(GBC.BOTH).setIpad(0, 0).setWeight(0, 1));

homeframe.add(canvasJ3D, new GBC(1, 1).setFill(GBC.BOTH).setIpad(20, 10).setWeight(1, 1));

homeframe.add(scroll, new GBC(1, 2).setFill(GBC.BOTH).setIpad(0, 100).setWeight(0, 0));

WestToolPanel.setLayout(new GridBagLayout());

WestToolPanel.add(LoadStatusButton, new GBC(0, 0).setFill(GBC.BOTH).setWeight(1, 1));

WestToolPanel.add(LoadTimeSeriesButton, new GBC(0, 1).setFill(GBC.BOTH).setWeight(1, 1));

WestToolPanel.add(AnimateTimeSeiresButton, new GBC(0, 2).setFill(GBC.BOTH).setWeight(1, 1));

WestToolPanel.add(CastSphereButton, new GBC(0, 3).setFill(GBC.BOTH).setWeight(1, 1));

WestToolPanel.add(LoadForceChainButton, new GBC(0, 4).setFill(GBC.BOTH).setWeight(1, 1));

WestToolPanel.add(VelocityFieldButton, new GBC(0, 5).setFill(GBC.BOTH).setWeight(1, 1));

WestToolPanel.add(CleanUpButton, new GBC(0, 6).setFill(GBC.BOTH).setWeight(1, 1));

NorthToolPanel.setLayout(new GridBagLayout());

NorthToolPanel.setLayout(new GridBagLayout());

NorthToolPanel.add(FileSettingLabel,

new GBC(0, 0, 6, 1).setFill(GBC.BOTH).setIpad(0, 0).setInsets(10, 0, 5, 0).setWeight(0, 1));

NorthToolPanel.add(RadiusLabel, new GBC(0, 1).setFill(GBC.BOTH).setIpad(0, 0).setWeight(1, 0));

NorthToolPanel.add(ParticleNumberLabel, new GBC(0, 2).setFill(GBC.BOTH).setIpad(0, 0).setWeight(1, 1));

NorthToolPanel.add(ScaleLabel, new GBC(0, 3).setFill(GBC.BOTH).setInsets(0, 0, 15, 0).setIpad(0, 0).setWeight(1, 1));

NorthToolPanel.add(RadiusText, new GBC(1, 1).setFill(GBC.BOTH).setIpad(0, 0).setWeight(1, 1));

NorthToolPanel.add(ParticleNumberText, new GBC(1, 2).setFill(GBC.BOTH).setIpad(0, 0).setWeight(1, 1));

NorthToolPanel.add(ScaleText, new GBC(1, 3).setFill(GBC.BOTH).setInsets(0, 0, 15, 0).setIpad(0, 0).setWeight(1, 1));

NorthToolPanel.add(XCoorLabel, new GBC(2, 1).setFill(GBC.BOTH).setIpad(0, 0).setWeight(1, 1));

NorthToolPanel.add(YCoorLabel, new GBC(2, 2).setFill(GBC.BOTH).setIpad(0, 0).setWeight(1, 1));

NorthToolPanel.add(ZCoorLabel, new GBC(2, 3).setFill(GBC.BOTH).setInsets(0, 0, 15, 0).setIpad(0, 0).setWeight(1, 1));

NorthToolPanel.add(XCoorText, new GBC(3, 1).setFill(GBC.BOTH).setIpad(0, 0).setWeight(1, 1));

NorthToolPanel.add(YCoorText, new GBC(3, 2).setFill(GBC.BOTH).setIpad(0, 0).setWeight(1, 1));

NorthToolPanel.add(ZCoorText, new GBC(3, 3).setFill(GBC.BOTH).setInsets(0, 0, 15, 0).setIpad(0, 0).setWeight(1, 1));

NorthToolPanel.add(TransparencyLabel, new GBC(4, 1).setFill(GBC.BOTH).setIpad(0, 0).setWeight(1, 1));

NorthToolPanel.add(ForceNumberLabel, new GBC(4, 2).setFill(GBC.BOTH).setIpad(0, 0).setWeight(1, 1));

NorthToolPanel.add(TransparencyText, new GBC(5, 1).setFill(GBC.BOTH).setIpad(0, 0).setInsets(0, 0, 0, 25).setWeight(1, 1));

NorthToolPanel.add(ForceNumberText, new GBC(5, 2).setFill(GBC.BOTH).setIpad(0, 0).setInsets(0, 0, 0, 25).setWeight(1, 1));

}

public void setupListener() {

VelocityFieldButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (IndexVelocity == 0) {

IndexVelocity = 1;

VelocityFieldButton.setBackground(new Color(178, 209, 152));

System.out.println(IndexVelocity);

}

else if (IndexVelocity == 1) {

IndexVelocity = 0;

VelocityFieldButton.setBackground(Color.GRAY);

System.out.println(IndexVelocity);

}

}

});

AnimateTimeSeiresButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (NumOfFilesTimeSeries != 0) {

float vmax = 0;

float vmin = 0;

if (IndexVelocity == 1) {

ArrayList<Float> temp = new ArrayList<Float>();

for (int i = 0; i < velocity.size(); i++) {

float m = findMin(velocity.get(i));

temp.add(m);

}

vmin = findMin(temp);

ArrayList<Float> temp2 = new ArrayList<Float>();

for (int i = 0; i < velocity.size(); i++) {

float m = findMax(velocity.get(i));

temp2.add(m);

}

vmax = findMax(temp2);

}

for (int j = 0; j < NumOfFilesTimeSeries; j++) {

for (int i = 0; i < NumOfParticles; i++) {

if (IndexVelocity == 1) {

float vthis = velocity.get(j).get(i);

float f = (0 + (vthis - vmin) \* (5 - 0) / (vmax - vmin));

Appearance appearance = new Appearance();

Material material1 = new Material();

if (f >= 0 & f <= 1) {

Color3f c = new Color3f(new Color(255 - (int) (255 \* f), 0, 255));

material1.setDiffuseColor(c);

appearance.setMaterial(material1);

}

if (f >= 1 & f <= 2) {

Color3f c = new Color3f(new Color(0, (int) (255 \* (f - 1)), 255));

material1.setDiffuseColor(c);

appearance.setMaterial(material1);

}

if (f >= 2 & f <= 3) {

Color3f c = new Color3f(new Color(0, 255, 255 - (int) (255 \* (f - 2))));

material1.setDiffuseColor(c);

appearance.setMaterial(material1);

}

if (f >= 3 & f <= 4) {

Color3f c = new Color3f(new Color((int) (255 \* (f - 3)), 255, 0));

material1.setDiffuseColor(c);

appearance.setMaterial(material1);

}

if (f >= 4 & f <= 5) {

Color3f c = new Color3f(new Color(255, 255 - (int) (255 \* (f - 4)), 0));

material1.setDiffuseColor(c);

appearance.setMaterial(material1);

}

ArrListSphF1.get(i).setAppearance(appearance);

}

Transform3D t3D = new Transform3D();

t3D.setTranslation(

new Vector3f(xcoor.get(j).get(i), ycoor.get(j).get(i), zcoor.get(j).get(i)));

ArrListTransGroupF1.get(i).setTransform(t3D);

}

try {

Thread thread = Thread.currentThread();

thread.sleep(80);

} catch (InterruptedException e1) {

e1.printStackTrace();

}

}

updateInfoText("Showing Animation Successful!");

}

else {

updateInfoText("Showing Animation Flaied! Please reload a folder that consists of a time series.");

}

}

});

CastSphereButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

dRound.setLocation(homeframe.getX() + homeframe.getWidth() / 2 - dRound.getWidth() / 2,

homeframe.getY() + homeframe.getHeight() / 2 - dRound.getHeight() / 2);

dRound.setVisible(true);

dRound.setAlwaysOnTop(true);

}

});

LoadStatusButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (TransparencyText.getText().length() != 0) {

try {

if (Float.valueOf(TransparencyText.getText()) < 0

|| Float.valueOf(TransparencyText.getText()) > 1) {

System.out.println("Please enter a transparency parameter that is between 0 and 1.");

return;

}

}

catch (NumberFormatException ex) {

System.out.println("Please enter the right format [NUMBER].");

return;

}

}

NumOfParticles = 0;

fdOpen.setMultipleMode(false);

fdOpen.setVisible(true);

String dirPath = fdOpen.getDirectory();

String fileName = fdOpen.getFile();

if (dirPath == null || fileName == null) {

return;

}

osFile = new File(dirPath, fileName);

try {

ColumnRadiusF1.clear();

ColumnXCoorF1.clear();

ColumnYCoorF1.clear();

ColumnZCoorF1.clear();

ColumnVelocity.clear();

BufferedReader br = new BufferedReader(new FileReader(osFile));

String line = br.readLine();

for (int i = 0; i < Integer.valueOf(ParticleNumberText.getText()); i++) {

String line1 = line.replaceAll("\\s+", " ");

String[] numbers = line1.split(" ");

if (numbers[0].length() != 0) {

ColumnRadiusF1.add(Float.valueOf(numbers[0]));

ColumnXCoorF1.add(Float.valueOf(numbers[Integer.valueOf(XCoorText.getText()) - 1]));

ColumnYCoorF1.add(Float.valueOf(numbers[Integer.valueOf(YCoorText.getText()) - 1]));

if (index3D2D == 3) {

ColumnZCoorF1.add(Float.valueOf(numbers[Integer.valueOf(ZCoorText.getText()) - 1]));

}

float xv = Float.valueOf(numbers[Integer.valueOf(9)]);

float yv = Float.valueOf(numbers[Integer.valueOf(10)]);

float zv = Float.valueOf(numbers[Integer.valueOf(11)]);

float v = (float) Math.sqrt(Math.pow(xv, 2) + Math.pow(yv, 2) + Math.pow(zv, 2));

ColumnVelocity.add(v);

NumOfParticles++;

line = br.readLine();

} else {

ColumnRadiusF1.add(Float.valueOf(numbers[1]));

ColumnXCoorF1.add(Float.valueOf(numbers[Integer.valueOf(XCoorText.getText())]));

ColumnYCoorF1.add(Float.valueOf(numbers[Integer.valueOf(YCoorText.getText())]));

if (index3D2D == 3) {

ColumnZCoorF1.add(Float.valueOf(numbers[Integer.valueOf(ZCoorText.getText())]));

}

float xv = Float.valueOf(numbers[Integer.valueOf(10)]);

float yv = Float.valueOf(numbers[Integer.valueOf(11)]);

float zv = Float.valueOf(numbers[Integer.valueOf(12)]);

float v = (float) Math.sqrt(Math.pow(xv, 2) + Math.pow(yv, 2) + Math.pow(zv, 2));

ColumnVelocity.add(v);

NumOfParticles++;

line = br.readLine();

}

}

br.close();

} catch (IOException ex) {

throw new RuntimeException("Read File Failed!");

}

NumOfFilesTimeSeries = 0;

radius.clear();

xcoor.clear();

ycoor.clear();

zcoor.clear();

velocity.clear();

LoadStatus();

updateInfoText("Loading Status Successful!");

}

});

LoadTimeSeriesButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

fdOpen.setMultipleMode(true);

fdOpen.setVisible(true);

File[] fs = fdOpen.getFiles();

if (fs.length == 0) {

return;

} else {

NumOfFilesTimeSeries = fs.length;

for (int j = 0; j < NumOfFilesTimeSeries; j++) {

try {

NumOfParticles = 0;

ArrayList<Float> cRadius = new ArrayList<Float>();

ArrayList<Float> cXCoor = new ArrayList<Float>();

ArrayList<Float> cYCoor = new ArrayList<Float>();

ArrayList<Float> cZCoor = new ArrayList<Float>();

ArrayList<Float> cvel = new ArrayList<Float>();

BufferedReader br = new BufferedReader(new FileReader(fs[j]));

String line = br.readLine();

for (int i = 0; i < Integer.valueOf(ParticleNumberText.getText()); i++) {

String line1 = line.replaceAll("\\s+", " ");

String[] numbers = line1.split(" ");

if (numbers[0].length() != 0) {

cRadius.add(Float.valueOf(numbers[0]));

cXCoor.add(Float.valueOf(numbers[Integer.valueOf(XCoorText.getText()) - 1]));

cYCoor.add(Float.valueOf(numbers[Integer.valueOf(YCoorText.getText()) - 1]));

if (index3D2D == 3) {

cZCoor.add(Float.valueOf(numbers[Integer.valueOf(ZCoorText.getText()) - 1]));

}

float xv = Float.valueOf(numbers[Integer.valueOf(9)]);

float yv = Float.valueOf(numbers[Integer.valueOf(10)]);

float zv = Float.valueOf(numbers[Integer.valueOf(11)]);

float v = (float) Math.sqrt(Math.pow(xv, 2) + Math.pow(yv, 2) + Math.pow(zv, 2));

cvel.add(v);

NumOfParticles++;

line = br.readLine();

} else {

cRadius.add(Float.valueOf(numbers[1]));

cXCoor.add(Float.valueOf(numbers[Integer.valueOf(XCoorText.getText())]));

cYCoor.add(Float.valueOf(numbers[Integer.valueOf(YCoorText.getText())]));

if (index3D2D == 3) {

cZCoor.add(Float.valueOf(numbers[Integer.valueOf(ZCoorText.getText())]));

}

float xv = Float.valueOf(numbers[Integer.valueOf(10)]);

float yv = Float.valueOf(numbers[Integer.valueOf(11)]);

float zv = Float.valueOf(numbers[Integer.valueOf(12)]);

float v = (float) Math.sqrt(Math.pow(xv, 2) + Math.pow(yv, 2) + Math.pow(zv, 2));

cvel.add(v);

NumOfParticles++;

line = br.readLine();

}

}

radius.add(cRadius);

xcoor.add(cXCoor);

ycoor.add(cYCoor);

zcoor.add(cZCoor);

velocity.add(cvel);

br.close();

} catch (IOException ex) {

throw new RuntimeException("Read Folder Failed!");

}

}

updateInfoText("Loading Folder Successful!");

}

}

});

LoadForceChainButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (TransparencyText.getText().length() != 0) {

try {

if (Float.valueOf(TransparencyText.getText()) < 0

|| Float.valueOf(TransparencyText.getText()) > 1) {

System.out.println("Please enter a transparency parameter that is between 0 and 1.");

return;

}

}

catch (NumberFormatException ex) {

System.out.println("Please enter the right format [NUMBER].");

return;

}

}

fdOpen.setMultipleMode(false);

fdOpen.setVisible(true);

String dirPath = fdOpen.getDirectory();

String fileName = fdOpen.getFile();

if (dirPath == null || fileName == null) {

return;

}

osFile = new File(dirPath, fileName);

try {

ColumnX1CoorF2.clear();

ColumnY1CoorF2.clear();

ColumnZ1CoorF2.clear();

ColumnX2CoorF2.clear();

ColumnY2CoorF2.clear();

ColumnZ2CoorF2.clear();

ColumnForceValueF2.clear();

BufferedReader br = new BufferedReader(new FileReader(osFile));

String line = br.readLine();

for (int i = 0; i < Integer.valueOf(ForceNumberText.getText()); i++) {

String line1 = line.replaceAll("\\s+", " ");

String[] numbers = line1.split(" ");

if (numbers[0].length() != 0) {

ColumnX1CoorF2.add(Float.valueOf(numbers[0]));

ColumnY1CoorF2.add(Float.valueOf(numbers[1]));

ColumnZ1CoorF2.add(Float.valueOf(numbers[2]));

ColumnX2CoorF2.add(Float.valueOf(numbers[3]));

ColumnY2CoorF2.add(Float.valueOf(numbers[4]));

ColumnZ2CoorF2.add(Float.valueOf(numbers[5]));

ColumnForceValueF2.add(Float.valueOf(numbers[6]));

line = br.readLine();

} else {

ColumnX1CoorF2.add(Float.valueOf(numbers[1]));

ColumnY1CoorF2.add(Float.valueOf(numbers[2]));

ColumnZ1CoorF2.add(Float.valueOf(numbers[3]));

ColumnX2CoorF2.add(Float.valueOf(numbers[4]));

ColumnY2CoorF2.add(Float.valueOf(numbers[5]));

ColumnZ2CoorF2.add(Float.valueOf(numbers[6]));

ColumnForceValueF2.add(Float.valueOf(numbers[7]));

line = br.readLine();

}

}

br.close();

} catch (IOException ex) {

throw new RuntimeException("Read File Failed!");

}

LoadForceChain();

updateInfoText("Loading Force Chain Successful!");

}

});

VelocityFieldButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

}

});

CleanUpButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

bg.detach();

tgUp.removeAllChildren();

bgUp.addChild(bg);

updateInfoText("Clean Up Successful!");

}

});

J3DButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (index3D2D == 3) {

;

} else if (index3D2D == 2) {

index3D2D = 3;

J3DButton.setBackground(new Color(22, 79, 79));

J2DButton.setBackground(new Color(151, 235, 235));

ZCoorText.setEditable(true);

} else {

;

}

}

});

J2DButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (index3D2D == 2) {

;

} else if (index3D2D == 3) {

index3D2D = 2;

J2DButton.setBackground(new Color(22, 79, 79));

J3DButton.setBackground(new Color(151, 235, 235));

ZCoorText.setEditable(false);

} else {

;

}

}

});

CRDcastRoundButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

bg.detach();

Appearance appearance = new Appearance();

Material material = new Material();

Color3f c = new Color3f(new Color(3, 168, 158));

material.setDiffuseColor(c);

appearance.setMaterial(material);

TransparencyAttributes ta = new TransparencyAttributes();

ta.setTransparencyMode(1);

ta.setTransparency(0.8f);

appearance.setTransparencyAttributes(ta);

tgCastSphere = new TransformGroup();

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

Sphere sph = new Sphere(Float.valueOf(CRDradiusc.getText()), 1, ClarityParaSph, appearance);

tgCastSphere.addChild(sph);

Transform3D t3D = new Transform3D();

Vector3f vec = new Vector3f(Float.valueOf(CRDxc.getText()), Float.valueOf(CRDyc.getText()),

Float.valueOf(CRDzc.getText()));

t3D.setTranslation(vec);

tgCastSphere.setTransform(t3D);

tgUp.addChild(tgCastSphere);

bgUp.addChild(bg);

updateInfoText("Casting Sphere Successful!");

dRound.setVisible(false);

}

});

dRound.addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e) {

dRound.setVisible(false);

}

});

homeframe.addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e) {

System.exit(0);

}

});

}

public void updateInfoText(String info) {

Date d = new Date();

String s = null;

DateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

s = df.format(d);

if (infoText.getText().length() == 0) {

infoText.append("(" + s + ") " + info);

} else {

infoText.append("\n" + "(" + s + ") " + info);

}

}

public void LoadStatus() {

bg.detach();

tgUp.removeAllChildren();

tgLoadStatus = new TransformGroup();

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

tgLoadStatus.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);

ArrListSphF1.clear();

ArrListTransGroupF1.clear();

ArrListTrans3DF1.clear();

ArrListVectorF1.clear();

float vmax = 0;

float vmin = 0;

if (IndexVelocity == 1) {

vmin = findMin(ColumnVelocity);

vmax = findMax(ColumnVelocity);

}

for (int i = 0; i <= NumOfParticles - 1; i++) {

Appearance appearance = new Appearance();

if (IndexVelocity == 1) {

float vthis = ColumnVelocity.get(i);

float f = 0;

if (vmax != vmin) {

f = (0 + (vthis - vmin) \* (5 - 0) / (vmax - vmin));

}

if (vmax == vmin) {

f = 0;

}

Material material1 = new Material();

material1.setShininess(128);

if (f >= 0 & f <= 1) {

Color3f c = new Color3f(new Color(255 - (int) (255 \* f), 0, 255));

material1.setDiffuseColor(c);

appearance.setMaterial(material1);

}

if (f >= 1 & f <= 2) {

Color3f c = new Color3f(new Color(0, (int) (255 \* (f - 1)), 255));

material1.setDiffuseColor(c);

appearance.setMaterial(material1);

}

if (f >= 2 & f <= 3) {

Color3f c = new Color3f(new Color(0, 255, 255 - (int) (255 \* (f - 2))));

material1.setDiffuseColor(c);

appearance.setMaterial(material1);

}

if (f >= 3 & f <= 4) {

Color3f c = new Color3f(new Color((int) (255 \* (f - 3)), 255, 0));

material1.setDiffuseColor(c);

appearance.setMaterial(material1);

}

if (f >= 4 & f <= 5) {

Color3f c = new Color3f(new Color(255, 255 - (int) (255 \* (f - 4)), 0));

material1.setDiffuseColor(c);

appearance.setMaterial(material1);

}

} else if (IndexVelocity == 0) {

Material material1 = new Material();

material1.setShininess(128);

Color3f c = new Color3f(new Color(65, 125, 125));

material1.setDiffuseColor(c);

appearance.setMaterial(material1);

}

if (TransparencyText.getText().length() != 0) {

TransparencyAttributes ta = new TransparencyAttributes();

ta.setTransparencyMode(1);

ta.setTransparency(Float.valueOf(TransparencyText.getText()));

appearance.setTransparencyAttributes(ta);

}

ArrListSphF1.add(new Sphere(ColumnRadiusF1.get(i) \* Float.valueOf(ScaleText.getText()), 1, ClarityParaSph,

appearance));

ArrListTransGroupF1.add(new TransformGroup());

ArrListTrans3DF1.add(new Transform3D());

ArrListSphF1.get(i).getShape().setCapability(Shape3D.ALLOW\_APPEARANCE\_WRITE);

ArrListSphF1.get(i).getShape().setCapability(Shape3D.ALLOW\_APPEARANCE\_READ);

ArrListSphF1.get(i).getShape().setCapability(Shape3D.ALLOW\_APPEARANCE\_OVERRIDE\_WRITE);

ArrListSphF1.get(i).getShape().setCapability(Shape3D.ALLOW\_APPEARANCE\_OVERRIDE\_READ);

if (index3D2D == 3) {

ArrListVectorF1.add(new Vector3f(ColumnXCoorF1.get(i) \* Float.valueOf(ScaleText.getText()),

ColumnYCoorF1.get(i) \* Float.valueOf(ScaleText.getText()),

ColumnZCoorF1.get(i) \* Float.valueOf(ScaleText.getText())));

}

if (index3D2D == 2) {

ArrListVectorF1.add(new Vector3f(ColumnXCoorF1.get(i) \* Float.valueOf(ScaleText.getText()),

ColumnYCoorF1.get(i) \* Float.valueOf(ScaleText.getText()), 0));

}

ArrListTrans3DF1.get(i).setTranslation(ArrListVectorF1.get(i));

ArrListTransGroupF1.get(i).addChild(ArrListSphF1.get(i));

ArrListTransGroupF1.get(i).setTransform(ArrListTrans3DF1.get(i));

tgLoadStatus.addChild(ArrListTransGroupF1.get(i));

}

for (int i = 0; i <= NumOfParticles - 1; i++) {

ArrListTransGroupF1.get(i).setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);

ArrListTransGroupF1.get(i).setCapability(Shape3D.ALLOW\_APPEARANCE\_WRITE);

ArrListTransGroupF1.get(i).setCapability(Shape3D.ALLOW\_APPEARANCE\_READ);

ArrListTransGroupF1.get(i).setCapability(Appearance.ALLOW\_COLORING\_ATTRIBUTES\_WRITE);

ArrListTransGroupF1.get(i).setCapability(Appearance.ALLOW\_COLORING\_ATTRIBUTES\_READ);

}

tgUp.addChild(tgLoadStatus);

bgUp.addChild(bg);

}

public float findMin(ArrayList<Float> arl) {

float min = arl.get(0);

int index = 0;

for (int i = 1; i < arl.size(); i++) {

if (arl.get(i) < min) {

min = arl.get(i);

index = i;

}

}

return arl.get(index);

}

public float findMax(ArrayList<Float> arl) {

float max = arl.get(0);

int index = 0;

for (int i = 1; i < arl.size(); i++) {

if (arl.get(i) > max) {

max = arl.get(i);

index = i;

}

}

return arl.get(index);

}

public void LoadForceChain() {

ArrListCylinderF2.clear();

ArrListTransGroupF2.clear();

ArrListTrans3DF2.clear();

ArrListVectorF2.clear();

bg.detach();

tgLoadForceChain = new TransformGroup();

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

tgLoadForceChain.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);

Appearance appearance = new Appearance();

Material material1 = new Material();

Color3f c = new Color3f(new Color(248, 108, 53));

material1.setDiffuseColor(c);

appearance.setMaterial(material1);

ArrListCylinderF2.clear();

ArrListTransGroupF2.clear();

ArrListTrans3DF2.clear();

ArrListVectorF2.clear();

for (int i = 0; i <= Integer.valueOf(ForceNumberText.getText()) - 1; i++) {

float x1 = ColumnX1CoorF2.get(i), y1 = ColumnY1CoorF2.get(i), z1 = ColumnZ1CoorF2.get(i),

x2 = ColumnX2CoorF2.get(i), y2 = ColumnY2CoorF2.get(i), z2 = ColumnZ2CoorF2.get(i),

force = ColumnForceValueF2.get(i), radius = Math.abs(force / 30000);

if (x1 == 0 && y1 == 0 && z1 == 0 && x2 == 0 && y2 == 0 && z2 == 0) {

break;

}

TransformGroup tg = myCylinder(x1, y1, z1, x2, y2, z2, radius, ClarityParaCylin, ClarityParaCylin, 15, appearance);

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

tgLoadForceChain.addChild(tg);

}

tgUp.addChild(tgLoadForceChain);

bgUp.addChild(bg);

}

public TransformGroup myCylinder(float x1, float y1, float z1, float x2, float y2, float z2, float radius,

int xclarity, int yclarity, int sphclarity, Appearance ap) {

float height = (float) Math.sqrt(Math.pow(x1 - x2, 2) + Math.pow(y1 - y2, 2) + Math.pow(z1 - z2, 2));

Cylinder cylin = new Cylinder(Math.abs(radius), height, 1, xclarity, yclarity, ap);

float a1 = 0, b1 = height / 2, c1 = 0, a2 = (x2 - x1) / 2, b2 = (y2 - y1) / 2, c2 = (z2 - z1) / 2;

float am = (a1 + a2) / 2, bm = (b1 + b2) / 2, cm = (c1 + c2) / 2;

float A = (float) Math.sqrt(Math.pow(am, 2) + Math.pow(bm, 2) + Math.pow(cm, 2));

float x0 = am / A, y0 = bm / A, z0 = cm / A;

Transform3D t3d = new Transform3D();

Quat4d q4d = new Quat4d(x0, y0, z0, 0);

t3d.setRotation(q4d);

Vector3f vec = new Vector3f((float) (0.5 \* (x1 + x2)), (float) (0.5 \* (y1 + y2)), (float) (0.5 \* (z1 + z2)));

t3d.setTranslation(vec);

TransformGroup tgCylin = new TransformGroup();

tgCylin.addChild(cylin);

tgCylin.setTransform(t3d);

Sphere sph1 = new Sphere(Math.abs(radius), 1, sphclarity, ap);

Transform3D t3dSph1 = new Transform3D();

Vector3f vecSph1 = new Vector3f(x1, y1, z1);

t3dSph1.setTranslation(vecSph1);

TransformGroup tgSph1 = new TransformGroup();

tgSph1.addChild(sph1);

tgSph1.setTransform(t3dSph1);

Sphere sph2 = new Sphere(Math.abs(radius), 1, sphclarity, ap);

Transform3D t3dSph2 = new Transform3D();

Vector3f vecSph2 = new Vector3f(x2, y2, z2);

t3dSph2.setTranslation(vecSph2);

TransformGroup tgSph2 = new TransformGroup();

tgSph2.addChild(sph2);

tgSph2.setTransform(t3dSph2);

TransformGroup tgbig = new TransformGroup();

tgbig.addChild(tgCylin);

tgbig.addChild(tgSph1);

tgbig.addChild(tgSph2);

return tgbig;

}

public static List<String> getFilesList(File file) {

List<String> result = new ArrayList<String>();

if (!file.isDirectory()) {

result.add(file.getAbsolutePath());

} else {

File[] directoryList = file.listFiles(new FileFilter() {

public boolean accept(File file) {

if (file.isFile() && file.getName().indexOf("txt") > -1) {

return true;

} else if (file.isFile() && file.getName().indexOf("dat") > -1) {

return true;

} else {

return false;

}

}

});

for (int i = 0; i < directoryList.length; i++) {

result.add(directoryList[i].getPath());

}

}

return result;

}

}

class GBC

import java.awt.GridBagConstraints;

import java.awt.Insets;

public class GBC extends GridBagConstraints {

public GBC(int gridx, int gridy) {

this.gridx = gridx;

this.gridy = gridy;

}

public GBC(int gridx, int gridy, int gridwidth, int gridheight) {

this.gridx = gridx;

this.gridy = gridy;

this.gridwidth = gridwidth;

this.gridheight = gridheight;

}

public GBC setAnchor(int anchor) {

this.anchor = anchor;

return this;

}

public GBC setFill(int fill) {

this.fill = fill;

return this;

}

public GBC setWeight(double weightx, double weighty) {

this.weightx = weightx;

this.weighty = weighty;

return this;

}

public GBC setInsets(int top, int left, int bottom, int right) {

this.insets = new Insets(top, left, bottom, right);

return this;

}

public GBC setIpad(int ipadx, int ipady) {

this.ipadx = ipadx;

this.ipady = ipady;

return this;

}

}