**海南大学计算机科学与技术学院**

**《计算机图形学》课内实验报告二**

****

**班 级：计算机科学与技术2021-3班**

**成 员： 李季鸿 20213002624**

**报告名称： 实验报告二**

**指导老师： 高新瑞**

**完成日期： 2023年9月21日**

**《计算机图形学》实验报告二**

**学生姓名：李季鸿 班级：21级计科3班 学号：20213002624**

**实验地点：9-202 指导教师：高新瑞**

**实验日期：**2023.9.21  **实验课时：2学时**

**实验环境：**Windows 10+JDK1.8+记事本+IntelliJ IDEA

**一、实验目的**

本实验旨在通过使用Java 3D编程，以一个简单的点的显示程序为例，达到以下目的：

1. 了解并熟练掌握Java 3D编程：通过编写和理解Java 3D程序，学习如何创建和操作三维图形场景。

2. 了解Java 3D基本图形功能：通过创建和显示点集，掌握Java 3D的基本图形绘制功能，包括顶点坐标和点的颜色设置。

3. 了解并熟悉Java 3D的场景式管理：通过创建分支组（BranchGroup）和添加行为（Behavior），学习如何使用Java 3D来管理三维场景，包括旋转、缩放和平移等操作。

1. 通过球体和长方体组合体程序对Java场景式管理实践：虽然代码中没有提到球体和长方体组合体，但可以扩展该程序，以实践更复杂的三维图形对象的创建和管理，例如球体和长方体的组合，从而深入理解Java 3D的场景管理功能。

通过完成这些目标，参与者将获得有关Java 3D编程和三维图形场景管理的基本知识和实践经验。

**二、实验过程**

## 实验内容：

（1）代码

1. */\*\**
2. \* \\* Created with IntelliJ IDEA.
3. \* \\* @ProjectName: 例3.2 点的显示程序实例
4. \* \\* @FileName: DisplayPoints
5. \* \\* @author: li-jihong
6. \* \\* Date: 2023-09-21 13:57
7. \*/
8. import com.sun.j3d.utils.applet.MainFrame;
9. import com.sun.j3d.utils.behaviors.mouse.MouseRotate;
10. import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;
11. import com.sun.j3d.utils.behaviors.mouse.MouseZoom;
12. import com.sun.j3d.utils.universe.SimpleUniverse;
13. import javax.media.j3d.\*;
14. import javax.vecmath.Color3f;
15. import javax.vecmath.Point3d;
16. import java.applet.Applet;
17. import java.awt.\*;
18. public class DisplayPoints extends Applet {
19. public DisplayPoints() {
20. setLayout(new BorderLayout());
21. GraphicsConfiguration gc = SimpleUniverse.getPreferredConfiguration();
22. Canvas3D c = new Canvas3D(gc);
23. add("Center", c);
24. BranchGroup BranchGroupScene = createBranchGroup();
25. SimpleUniverse u = new SimpleUniverse(c);
26. u.getViewingPlatform().setNominalViewingTransform();
27. u.addBranchGraph(BranchGroupScene);
28. }
29. public static void main(String[] args) {
30. new MainFrame(new DisplayPoints(), 450, 450);
31. }
32. public BranchGroup createBranchGroup() {
33. BranchGroup BranchGroupRoot = new BranchGroup();
34. BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);
35. Color3f bgColor = new Color3f(1.0f, 1.0f, 1.0f);
36. Background bg = new Background(bgColor);
37. bg.setApplicationBounds(bounds);
38. BranchGroupRoot.addChild(bg);
39. TransformGroup transformgroup = new TransformGroup();
40. transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);
41. transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);
42. BranchGroupRoot.addChild(transformgroup);
43. MouseRotate mouserotate = new MouseRotate();
44. mouserotate.setTransformGroup(transformgroup);
45. BranchGroupRoot.addChild(mouserotate);
46. mouserotate.setSchedulingBounds(bounds);
47. MouseZoom mousezoom = new MouseZoom();
48. mousezoom.setTransformGroup(transformgroup);
49. BranchGroupRoot.addChild(mousezoom);
50. mousezoom.setSchedulingBounds(bounds);
51. MouseTranslate mousetranslate = new MouseTranslate();
52. mousetranslate.setTransformGroup(transformgroup);
53. BranchGroupRoot.addChild(mousetranslate);
54. mousetranslate.setSchedulingBounds(bounds);
55. Shape3D shapepoints = new Shape3D();
56. *//定义顶点坐标*
57. float vertexes[] = {.5f, 0.6f, 0.0f, -0.5f, 0.6f, 0.0f,
58. 0.5f, 0.05f, 0.0f, -0.5f, 0.05f, 0.f,
59. -0.5f, -0.7f, 0.0f, 0.5f, -0.7f, 0.1f,};
60. *//定义点颜色*
61. float pointcolors[] = {1.0f, 0.f, 0.0f, 0.0f, 1.f, 0.0f,
62. 0.0f, 0.f, 1.0f, 1.0f, 1.0f, 0.f,
63. 0.0f, 1.0f, 1.f, 1.f, 0.f, 1.0f};
64. int vCount = 6;
65. PointArray points = new PointArray(vCount, PointArray.COORDINATES |
66. PointArray.COLOR\_3);
67. points.setCoordinates(0, vertexes);
68. points.setColors(0, pointcolors);
69. Appearance app = new Appearance();
70. *//定义点的属性*
71. PointAttributes pointsattributes = new PointAttributes();
72. *//定义点的大小*
73. pointsattributes.setPointSize(70.0f);
74. *//如有下面这项，空间点显示为圆球形；如无，空间点显示为正方形*
75. pointsattributes.setPointAntialiasingEnable(false);
76. app.setPointAttributes(pointsattributes);
77. shapepoints.setGeometry(points);
78. shapepoints.setAppearance(app);
79. transformgroup.addChild(shapepoints);
80. BranchGroupRoot.compile();
81. return BranchGroupRoot;
82. }
83. }
84. 结果截图

|  |  |
| --- | --- |
|  |  |
| 图2-1 使用反走样参数 | 图2-2 未使用反走样参数 |

**三、实验总结**

在本次实验中，我使用了Java 3D编程技术创建了一个简单的三维点的显示程序。这个实验的关键焦点是观察点反走样参数`setPointAntialiasingEnable`的开启和关闭对点的显示形状造成的影响。

1. 实验目的：本实验的主要目标是研究和理解Java 3D编程技术，以及探讨点抗锯齿效果参数在点的显示中的作用。

2. 实验步骤：在实验中，我按照以下步骤进行了操作：

- 首先，我创建了一个Java 3D程序，设置了三维图形的场景，包括窗口布局和视图配置。

- 然后，我创建了一个包含点集的三维场景，其中点的坐标和颜色分别由`vertexes`和`pointcolors`数组定义。

- 接着，我使用了`PointAttributes`来定义了点的属性，其中包括点的大小。

- 最重要的是，我进行了两次实验观察点的显示效果。一次是开启了点反走样效果，另一次是关闭了点反走样效果。

3. 结果比较：在观察实验结果时，我注意到`setPointAntialiasingEnable`参数的状态对点的显示形状产生了显著影响。当该参数开启时，点呈现为圆球形状，边缘更加圆滑；而当关闭该参数时，点呈现为正方形状，边缘更加锐利。这是因为点反走样效果的开启可以平滑点的边缘，使其看起来更加光滑。

4. 实验总结：通过这个实验，我不仅学会了如何使用Java 3D创建三维点集并控制其外观，还深入了解了点反走样效果参数在三维图形显示中的重要作用。这个实验为我提供了有关Java 3D编程技术的基础知识，并突出了反走样效果参数的重要性。根据实际需求，我可以选择开启或关闭反走样效果，以获得不同的视觉效果，从而更好地满足项目的要求。