基于 Vega Prime 的三维选择的实现

韦 波[1]，李安明[2]
（1. 桂林工学院 土木工程系，广西 桂林 541004；2. 桂林市自来水公司 管网科，广西 桂林 541002）

摘 要：三维选择是三维信息系统、二维 GIS 与三维视景仿真技术结合等信息系统中重要的功能之一。三维选择的实现包括选择方式和选择工具的实现。给出了选择方式的实现方法，叙述了高亮显示、选择工具和选择工具图形绘制的实现方法。

关键词：GIS；Vega Prime；三维选择；选择方式；选择工具
中图分类号：P208　　**文献标志码**：B　　**文章编号**：1672-4623 (2009) 02-0004-03

Implementation of 3D Selection Based on Vega Prime

WEI Bo[1], LI Anming[2]
(1. Department of Civil Engineering, Guilin University of Technology, Guilin 541004, China;
2. Department of Pipe Network, Guilin Water Supply Co., Ltd., Guilin 541002, China)

Abstract: 3D selection is an important function in 3D and 2D combine with 3D vision simulation information system. The implementation of 3D selection includes selection mode and selection tool. It provides methods to realize the selection mode, highlight, selection tool and the drawing of selection tool.

Key words: GIS; vega prime; 3D selection; selection mode; selection tool

1 三维选择

三维选择与地理信息系统（GIS）软件中的二维选择相似，可以进行单选、多选（按住 Ctrl 键）、矩形选择、圆形选择、多边形选择、名称选择以及取消选择等操作，但与二维选择不同的是，三维选择还可以设定选择方式，包括对象节点方式（OBJECT）、几何节点方式（GEOMETRY）、DOF（Degrees of Freedom）节点方式、LOD(Levels of Detail)节点方式和NAMEDPART节点方式等[1-3]（见图 1）。

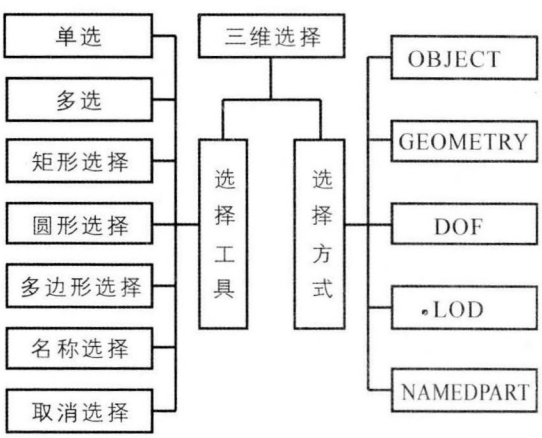

图 1 三维选择的内容

2 三维选择的实现

2.1 选择方式的实现

1）OBJECT 节点方式。应用于选择场景中的对象节点。使用 vpIsectorLOS 的 getHitObject 实现。

2）GEOMETRY 方式。应用于选择构建场景中对象的几何节点。使用 vpIsectorLOS 的 getHitNode 实现。

3）DOF 方式。应用于选择场景中的 DOF 节点。首先使用 vpIsectorLOS 的 getHitNode 获取几何节点，然后依次寻找该几何节点类型为 vsDOF::getStaticClassType 的父节点，如果存在，则该父节点为所需的 DOF 节点。

4）LOD 方式。应用于选择场景中的 LOD 节点。首先使用 vpIsectorLOS 的 getHitNode 获取几何节点，然后依次寻找该几何节点类型为 vsLOD::getStaticClassType 的父节点，如果存在，则该父节点为所需的 LOD 节点。

5）NAMEDPART 方式。应用于选择构建场景中对象的名字节点。首先使用 vpIsectorLOS 的 getHitNode 获取几何节点，然后依次寻找该几何节点的父节点，如果父节点的 getName 存在，且与对象节点的 getName 不一致，则该父节点为所需的名字节点。

收稿日期：2008-10-15
项目来源：广西教育厅面上科研项目（200708MS129）；广西自然科学基金项目（0833250）；学院青年教师科研扶持基金项目。

2.2　选择工具的实现

2.2.1　高亮显示的实现

高亮显示可有两种方法，分别通过 vrRenderStrategyBounds 和 vrRenderStrategyHigh- Light 策略来实现。VrRenderStrategyBounds 用于高亮对象的边界盒子，而 vrRenderStrategyHighLight 用于高亮对象本身。

对几何节点实施高亮显示策略使用 vrGeometry 的 setRenderStrategy 函数实现。对对象节点、DOF 节点、LOD 节点和名字节点，需要遍历其父节点，直至找到整个对象为止。遍历父节点使用 vsTraversalUser 辅助实现。

```
void vpSelection::setRenderStrategy(vsNode * root, vrRenderStrategy *strategy)
{
  vsTraversalUser<vrRenderStrategy*, vsTraversalLookUpNodeId> trav (strategy);
trav.addPreVisit (vsGeometry:: getStaticNodeId (), this, &vpSelection:: travFuncGeometry);
  trav.visit (root);
}
vsTraversal:: ResultvpSelection:: travFuncGeometry (vsNode*node, vrRenderStrategy * strategy)
{
vrGeometry*geometry=static_cast< vsGeometry*> (node) ->getGeometry ();
  geometry->setRenderStrategy (strategy);
  return vsTraversal:: RESULT_CONTINUE;
}
```

其中，vpSelection 为自定义的一个三维选择类，setRenderStrategy 和 travFuncGeometry 为 vpSelection 类中两个自定义的成员函数。前者实现遍历父节点，后者则实现高亮显示。

2.2.2　选择工具的实现

1）单选。根据设置的选择方式判断鼠标点击是否选中了对象。在判断之前，需要为 vpIsectorLOS 准备 4 种数据：Translate 参数、Rotate 参数、SegmentRange 参数和鼠标点击处的坐标。对于非 ORTHOGRAPHIC 投影，数据获取方法为：

①Translate 参数。获取 x, y, z 的值。

```
const vuMatrix<double> &viewMat = channel ->getViewMatrix ();
    x = viewMat [3] [0];
    y = viewMat [3] [1];
    z = viewMat [3] [2];
```

其中，channel 为场景通道。

② Rotate 参数。获取 h, p, r 的值，其中 r=0.0。

```
    vuVec3<float> mouse (mx, my, -1);
    vuVec3<float> vec;
    vuMatrix<float> projInv;
    projInv.invert (channel->getVrChannel () -> getProjection Matrix ());
    projInv.transformPoint (&mouse);
    channel->getVrChannel () -> getOffsetMatrixInverse () .transformPoint (&mouse);
    channel->getVrChannel () ->getViewMatrix (). transformPoint (&mouse);
    vec [0] = mouse [0] - x;
    vec [1] = mouse [1] - y;
    vec [2] = mouse [2] - z;
    h = vuRad2Deg (-vuArcTan (vec [0], vec [1]));
    p=vuRad2Deg (vuArcTan (vec [2], vuSqrt (
    vuSq (vec [0]) + vuSq (vec [1]))));
```

其中，mx、my 为鼠标点击处坐标，范围为 [-1，1]。

③ SegmentRange 参数。获取 range 的值。

```
    channel->getNearFar (&n, &f);
    range = 2 * f;
```

④ 鼠标点击处坐标。

```
    vpWindow *m_pWin = *vpWindow:: begin ();
    vrWindow:: Mouse mouse=m_pWin->
    getMouse ();
    mx=mouse.m_nx;
    my=mouse.m_ny;
```

其中，Vega Prime 的 Mouse 结构成员 m_nx 和 m_ny 的坐标范围正好为 [-1，1]。

2）多选（按住 Ctrl 键）。当按住 Ctrl 键时，鼠标点击执行多选。如果某对象已被选中，再次点击则取消选择。多选实质上是多次执行单选，但需要将被选择的对象存储起来，使用 CPtrList 实现。

3）矩形选择。即从按下至松开鼠标左键所移动的矩形区域。可以依据矩形的边界按照设定的搜索间距循环执行单选操作。多个对象被选中使用 CPtrList 存储。

4）圆形选择。先获取圆的外接矩形，然后执行矩形选择。为保证搜索点在圆形范围内，建立约束条件：搜索点到圆心的距离小于或等于圆的半径。

5）多边形选择。先获取多边形的外接矩形，然后执行矩形选择。矩形选择过程中搜索点必须落在多边形内，使用奇-偶规则 [4] 进行判断，程序基于 X-扫描线算法 [4] 实现。

6）名称选择。名称选择不属于鼠标操作的选择，是根据输入的节点名称选择对象。使用 vsNode 的 getName 获取节点名称比较判断。

7）取消选择。取消选择通过给对象高亮显示的策略赋予 NULL 实现：

```
geometry->setRenderStrategy (NULL);
```

2.2.3 选择工具图形的绘制

在执行矩形、圆和多边形选择时，鼠标移动时应同时绘制出相应的矩形、圆和多边形图形。由于绘制的图形不需要加入到场景中，而且是二维图形，因此使用 Vega Prime 的一个非官方插件 vpOverlay [5] 来实现。vpOverlay 包含有 6 个类，这里只使用 vpOverlay 2DlineStrip 一个类。

1）矩形的绘制。vpOverlay2DlineStrip 是用来绘制连续直线的，因此绘制矩形时需要进行一些改变，否则绘制出的是鼠标移动的轨迹。为了绘制出矩形，需要计算矩形的 4 个角点坐标，根据鼠标移动的当前点与最初按下鼠标位置的点可计算出。4 个角点坐标计算出来后，可调用 vpOverlay2DlineStrip 的 addVertex 按顺时针或逆时针方向依次添加角点坐标即可绘制出图形。为绘制出封闭的矩形，最初按下鼠标位置的点的坐标需要添加两次，一次是开始，第二次是最后。Vega Prime 的 Mouse 结构成员 m_nx 和 m_ny 的坐标范围是 [-1,1]，而 vpOverlay2DlineStrip 的坐标范围是 [0,1]，因此绘制图形前需要进行坐标转换：

```
X = (X+1) /2.0;
Y = (Y+1) /2.0;
```

图形绘制过程中遵循“擦-绘”的循环过程，清除原来的图形使用 vpOverlay2DlineStrip 的 vertexList.clear() 实现。要使图形可见设置 vpOverlay2DlineStrip 的 setEnable 为 TRUE。

2）圆的绘制。实现过程与矩形的绘制相似。由于 vpOverlay2DlineStrip 不能直接绘制圆或圆弧，所以通过其内接多边形逼近来实现。多边形的边是连续直线，所以仍然可以使用 vpOverlay2DlineStrip 来绘制图形。首先构建一个圆，以最初按下鼠标位置的点为圆心，通过鼠标移动的点与圆心坐标计算半径。Vega Prime 的 vrSphere 可以用来构建一个球面，除圆心坐标、半径外，还需要设定镶嵌度，也就是逼近圆的程度，当其取值大于 30 时圆弧看不出棱角，但若超过 70 则圆的动态绘制会明显慢。使用 vrSphere 的 getVertices 可获得全部顶点，这些顶点中处于圆内部的点必须去掉，只保留构建圆边界的顶点，可以通过计算顶点与圆心的距离与半径比较来进行筛选，把符合条件的顶点使用 vpOverlay2DlineStrip 的 addVertex 依次添加即可绘制出圆。

3）多边形的绘制。实现过程也与矩形的绘制过程相似，只是 vpOverlay2DlineStrip 的 vertexList 只保留依次按下鼠标的点的坐标，其中起始点的坐标在头、尾各存储一次。

2.2.4 鼠标键盘响应函数

使用选择工具必须使用鼠标或键盘，以接收鼠标或键盘的消息。Vega Prime 中可以自定义响应鼠标或键盘的函数：

1）声明函数，必须使用 static 类型。

```
static bool MyvrWinMsgHandlerFunc (
vrWindow *vrWin, vrWindow:: Message id,
int param1, int param2, int param3);
```

2）函数体。

```
bool CGL3DView:: MyvrWinMsgHandlerFunc (
vrWindow *vrWin, vrWindow:: Message id,
int param1, int param2, int param3) {...
vrWin->defaultVrWinMsgHandler (vrWin, id,
param1, param2, param3); //必须加入
return TRUE; }
```

3）设置自定义函数。

```
vpWindow *m_pWin = *vpWindow:: begin ();
m_pWin->setUserVrWinMsgHandler (
MyvrWinMsgHandlerFunc);
```

3 结 语

三维选择是三维信息系统、二维 GIS 与三维视景仿真技术结合等信息系统中重要的功能之一，通过三维选择的实现，可以进一步开发如图查属性、属性查图等二维 GIS 功能，从而完善开发系统。

参考文献

[1] MultiGen®. Vega Prime Programmer's Guide Version 1.2[Z]. 2003

[2] MultiGen®. MultiGen® Creator User's Guide Version 3.0 for Windows[Z].2004

[3] VRCHINA.http://www.vrchina.net/bbs/viewthread.php?tid=2828 [EB/OL].2008-09-05

[4] 陈传波，陆枫. 计算机图形学基础[M]. 北京：电子工业出版社，2005

[5] MultiGen®. http://www.multigen.com/cgi-bin/dl-plugins.cgi?url=http://www.multigen-paradigm.com/support/sc_files/ unsupported/vpOverlay1.13forVegaPrime1.2_Setup.exe [EB/OL]. 2008-09-30

第一作者简介：韦波，副教授，主要从事 GIS 技术集成应用研究。