

文章编号:1007-2780(2016)07-0708-06

基于 OpenGL 的三维模型点坐标拾取方法

施珂奕^{1,2}, 邓春健^{1,2*}, 邹 昆^{1,2}

(1. 电子科技大学 计算机科学与工程学院, 四川 成都 611731;

2. 电子科技大学 中山学院, 广东 中山 528402)

摘要:获得三维模型指定点的空间坐标是针对模型文件操作的基础。针对传统算法的不足提出一种获取所选点原始坐标的方法。在选点的过程中,该算法允许模型进行任意角度的旋转,以选取模型侧面及背面的点,并通过旋转矩阵对该点还原。算法完成从屏幕的二维像素坐标到模型的三维坐标,再到模型文件原始坐标的转换。实验表明,该方法可以准确获取模型的所有可视表面的点坐标,提高了拾取的灵活性。

关 键 词:OpenGL;三维模型;拾取;坐标

中图分类号:TP79 **文献标识码:**A **doi:**10.3788/YJYXS20163107.0708

Point pick-up in three-dimension model based on OpenGL

SHI Ke-yi^{1,2}, DENG Chun-jian^{1,2*}, ZOU Kun^{1,2}

(1. *School of Computer Science & Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China;*

2. *Zhongshan Institute, University of Electronic Science and Technology of China, Zhongshan 528402, China*)

Abstract: To pick-up a specified coordinate of a point is a basic operation for a 3D model file. Aimed at the shortages of traditional methods about 3D picking, a new method, which points at getting the original coordinate of specified point, is proposed. During picking, rotation at any angle is allowed. So points either on the side or on the back can be selected. And rotation matrix is used to restore points to their original state. The method transform the two-dimensional pixel coordinate to 3D model coordinate then to original model coordinate. All the points on visible surface can be picked precisely by this

收稿日期:2016-01-04; **修订日期:**2016-02-29.

基金项目:国家自然科学基金项目(No. 61302115, No. 61502088);广东省高等学校优秀青年教师培养计划项目(No. Yq2013204, No. Yq2013206);中山市产学研结合项目(No. 2013C2FC0014);电子薄膜与集成器件国家重点实验室中山分实验室开放基金(No. 412S0605)

Supported by National Natural Science Foundation of China(No. 61302115, No. 61502088); Training Program for Distinguished Young Teachers in Higher Education of Guangdong, China(No. Yq2013204, No. Yq2013206); Project on the Integration of Industry, Education and Research of Zhongshan, China(No. 2013C2FC0014)

* 通信联系人, E-mail: dengcj80@126.com

method and thus the flexibility is improved.

Key words: OpenGL; three-dimension model; pick-up; coordinate

1 引言

三维模型的交互控制是人机交互的重要功能,为了操作三维模型,常常需要获取图形特定部位的空间坐标。如何正确获取空间点的坐标变得尤为重要,其关系到是否能准确地执行图形操作。与二维图形的坐标获取不同,在三维环境下,通过鼠标等输入设备获取的二维屏幕坐标需要转换为三维空间坐标,而不能仅仅对二维屏幕坐标进行线性变换而获得。

为了解决点坐标拾取的问题,OpenGL 提供了选择模式和名字堆栈。在选择模式下,选择的基本单位为图元。目前,实际中常用三角面片的集合来近似地表现空间图形,即对原始模型的表面进行三角剖分,用离散的三角片来表示模型的所有表面。对于由三角面片构成的图形来说,最小的有效图元是单个三角面片。采用此方法时需对每个三角面片命名并压入名字堆栈。故在选择模式下可以定位到具体的三角片,然后通过该三角片的三个顶点坐标求出近似的点击坐标。此方法局限性很大,首先,为了描述一个稍微复杂的几何图形的表面信息,用到的三角面片数量动辄上千上万,将所有三角片压入名字堆栈会大幅降低系统性能,甚至可能溢出;其次,对于稍大的三角片,通过顶点坐标的平均值来确定点击点位置的坐标会有极大的误差。

为了克服 OpenGL 的局限性,已有不少各具特色的解决办法,有的采取重绘式拾取法^[1],分别通过虚拟绘图和实际绘图的配合来帮助选取物体和点坐标。有的将模型分成若干四边形区域,建立网格,层层逼近点击部位所对应的图元,或者通过 OpenGL 的选择模式,选定图元,然后根据图元来求出交点坐标^[2-3]。有的通过对不同坐标系之间的转换和还原来求取指定点^[4-5]。有的首先获取三维模型的尺寸,然后根据尺寸建立一个特殊的正投影。在获取通过鼠标点击得到的屏幕然后,通过“针刺法”求得三维坐标。这种方法已能初步获取准确的坐标,但由于参数的特殊性,只能获取模型当前可以被操作者观察到部分的坐标,

无法做到全方位取点,也具有局限性^[6-9]。

针对以上方法的不足,本文提出了一种新的基于 OpenGL 的三维模型点坐标拾取的算法,该算法首先根据当前模型的尺寸信息设置合适的 OpenGL 视景体,以此显示模型。在 OpenGL 窗口,通过其提供的旋转函数,操作者可以选取模型表面上任意一点。根据模型尺寸、视景体尺寸和窗体信息,可以将通过鼠标的点击所获得的屏幕像素坐标转化为模型当前旋转状态的 x 轴与 y 轴坐标,再通过“针刺法”求得 z 轴坐标。之后,根据旋转矩阵,将所获得的坐标进行反向旋转,获得最终点坐标。这种算法允许自由地对模型进行旋转取点,避免了从特定方向取点的局限性,也保证了坐标的精准度。

2 基于 OpenGL 的三维模型点坐标拾取的算法

2.1 OpenGL 正投影环境的建立和参数设置

OpenGL 正投影环境的建立和参数设置是所有操作的基础。这里之所以采用正投影^[9]而非透视投影,是因为正投影不会对模型远处的部分进行缩短和收缩,从而可以准确地显示出模型的大小和度量,有利于提高选点的精确度。由于不同的模型的尺寸不同,如果使用相同的参数进行视景体配置,将可能会面临在窗口中显示的图形过小或过大的问题,这会严重影响到选点的范围和准确性,所以,针对模型的尺寸,采取动态设置参数的方法,对模型进行扫描,得到物体的真实三维坐标中,在 X 轴、 Y 轴和 Z 轴方向上的最大值和最小值,分别以 $x_{\min}, x_{\max}, y_{\min}, y_{\max}, z_{\min}, z_{\max}$ 来表示。在这里,令:

$$\begin{aligned} \text{left} &= -(x_{\max} - x_{\min}) \times 2; \\ \text{right} &= (x_{\max} - x_{\min}) \times 2; \\ \text{bottom} &= -(y_{\max} - y_{\min}) \times 2; \\ \text{top} &= (y_{\max} - y_{\min}) \times 2; \\ \text{near} &= -\sqrt{(\text{left} \times \text{left} + \text{bottom} \times \text{bottom} + (z_{\max} - z_{\min}) \times (z_{\max} - z_{\min}))}; \\ \text{far} &= \sqrt{(\text{left} \times \text{left} + \text{bottom} \times \text{bottom} + (z_{\max} - z_{\min}) \times (z_{\max} - z_{\min}))}; \end{aligned}$$

然后通过 OpenGL 提供的正投影视景体创建函数创建视景体:

```
glOrtho(left,right,bottom,top,near,far);
```

其中 near 和 far 取值的意义是:模型表面上,直线距离最大的两个点间的距离。这样设置可以保证,模型无论如何旋转,总可以将最上层的表面显示出来,而不会出现模型的上表面超出视景体界限的情况,避免图形显示不完全而导致的选点失败。

接下来利用选择模式来进一步提高选点精度。当所需选取的点处于模型边缘时,为了保证鼠标点击的部位是在模型上而非模型旁边的空白处,可以使用 OpenGL 提供的 GL_SELECT 模式,这样可以保证所获得数据的正确性。通过下面的函数进入选择模式:

```
glRenderMode(GL_SELECT);
```

在选择模式下绘制图形,绘制完成后用以下函数切换回渲染模式并检查返回值:

```
glRenderMode(GL_RENDER);
```

如果返回值为 1,则说明在模型上出现了一次点击,否则该次点击没有命中模型。

2.2 鼠标选点坐标的获取

通过鼠标点击,获取的二维坐标实际上是屏幕像素的坐标,窗口左上角为原点,向右为 x 轴的正方向,向下为 y 轴的正方向,如图 1 所示。

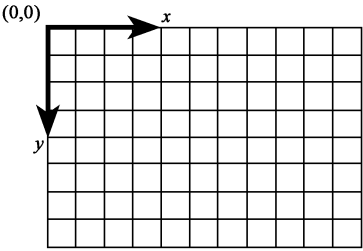


图 1 屏幕坐标系
Fig. 1 Coordinate system of screen

按上一小节的参数设置后,模型将显示在窗口的中间部位。将三维模型显示在二维平面的过程,实际上是把从观察者的位置所能看到的物体所有的表面经过缩放变换显示在屏幕上。所以,屏幕的二维像素坐标系与物体当前状态的 xOy 平面之间存在一种线性关系。知道了这一点,我们就能通过线性变换求出所选点的真实坐标 (x',y') 。

所获取的屏幕像素坐标设为 $(x_{\text{pixel}},y_{\text{pixel}})$,以 width 和 height 分别表示当前窗口的宽和高。以 x 坐标转换为例,由上一小节的参数设置可知,物体真实坐标的原点 $(0,0)$ 被显示在窗口的正中间,向左为 x 轴的负方向,向右为 x 轴的正方向。但屏幕坐标系的原点是位于窗口的左上角,向右为 x 轴正方向。所以,将 x_{pixel} 减去窗口宽的一半,这样,物体真实坐标系原点的位置就被移动到窗口最左侧,两个坐标系的原点就重合了。接下来就是按照一定比例进行缩放。由前面的参数设置可知,屏幕在 x 轴方向上,可以显示的容量为 $\text{right-left} = 4 \times (x_{\text{max}} - x_{\text{min}})$,即物体在 x 轴方向上两点间距离最大值的 4 倍。所以,易得缩放比例系数为 $\text{width}/(4 \times \text{right})$ 。

对于 y 轴坐标,其基本原理与 x 轴类似,但需要注意的是,屏幕坐标的 y 轴是以下为正方向,而物体真实坐标系是以上为正方向。

根据以上分析,即可得出下面的一组公式(1)和(2),得出坐标 (x',y') :

$$x' = \frac{x_{\text{pixel}} - \frac{\text{width}}{2}}{\frac{\text{width}}{4 \times \text{right}}} \tag{1}$$

$$y' = -\frac{\frac{\text{height}}{2} - y_{\text{pixel}}}{\frac{\text{height}}{4 \times \text{top}}} \tag{2}$$

坐标 (x',y') 表示的是模型在当前旋转状态下,鼠标点击部位 x 轴和 y 轴的模型坐标,所以在用针刺法求取 z 轴坐标 z' 之前,必须对读入的模型文件中的数据作相同的旋转处理,使模型文件中的数据与窗口中显示的模型处于相同的旋转状态。这里,可以利用旋转矩阵对模型数据进行处理。旋转矩阵是一个 4×4 的矩阵,将旋转矩阵左乘一个点,效果即为将该点按旋转矩阵的参数进行旋转。如公式(3)所示。

$$\begin{pmatrix} m[0] & m[4] & m[8] & m[12] \\ m[1] & m[5] & m[9] & m[13] \\ m[2] & m[6] & m[10] & m[14] \\ m[3] & m[7] & m[11] & m[15] \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \\ H \end{pmatrix} = \begin{pmatrix} X/H \\ Y/H \\ Z/H \\ 1 \end{pmatrix} = \begin{pmatrix} X_{\text{final}} \\ Y_{\text{final}} \\ Z_{\text{final}} \\ 1 \end{pmatrix} \tag{3}$$

在变换中,点一般采用齐次坐标的方式来表示,通常情况下,变换后点的第 4 个分量 H 一般不是 1,为了用正常化的齐次坐标表示变换后的点,需要用 H 去除 4 个分量,即得到正常化齐次坐标。对模型中所有的点都左乘同一个旋转矩阵,即完成模型的旋转。为了获取窗口所显示模型的旋转矩阵,可以使用 OpenGL 提供的以下函数:

`glGetFloatv(GL_MODELVIEW_MATRIX, m);`

其中 m 被声明为 `GLfloat m[16]`,获取的旋转矩阵存放在数组 m 中,且矩阵中元素的排列方式按照公式(1)中所示排列。

在旋转后的模型中,求取所有 x 轴坐标等于 x' 且 y 轴坐标等于 y' 的点,并统计它们的 z 轴坐标,其中 z 轴坐标值最大的点即为所选的点,因为操作者所看到的点总是面向操作者方向的最上面的点,将其记为 z' 。至此,已获得了物体在某种旋转状态下,所选取的某点的坐标 (x', y', z') 。

2.3 获取原始模型坐标点

为了将 (x', y', z') 还原成原始状态下的坐标,需要对该点进行反向旋转。在介绍如何反向旋转之前,首先分析一下旋转矩阵。由于在屏幕坐标系下, z 轴是垂直于屏幕向外的,绕 z 轴旋转的效果是模型在屏幕上顺时针或逆时针旋转,并不会显露出新的模型部分,对取点来说并无帮助,故对绕 z 轴旋转的情况不做分析。以下主要分析物体绕 x 轴旋转和绕 y 轴旋转的情况。

假设物体当前的状态是绕 x 轴旋转 α 度,则旋转函数和相应的旋转矩阵 M_x 的构成如公式(4):

`glRotatef( $\alpha$ , 1.0f, 0.0f, 0.0f);`

$$M_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (4)$$

假设物体当前的状态是绕 y 轴旋转 β 度,则旋转函数和相应的旋转矩阵 M_y 的构成如公式(5):

`glRotatef( $\beta$ , 0.0f, 1.0f, 0.0f);`

$$M_y = \begin{pmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (5)$$

以上两个旋转矩阵是绕 x 轴或 y 轴旋转变换的基础,当同时绕 x 轴和 y 轴旋转的时候,仅仅需要将两个矩阵相乘,其相乘的顺序取决于旋转函数执行的顺序,对实际的旋转效果无影响。公式(6)所示矩阵是先绕 x 轴旋转 α 度、再绕 y 轴旋转 β 度的旋转矩阵:

$$M_{xy} = \begin{pmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ \sin\alpha\sin\beta & \cos\alpha & \sin\alpha\cos\beta & 0 \\ \cos\alpha\sin\beta & -\sin\alpha & \cos\alpha\cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (6)$$

反向旋转实际上等同于将模型绕 x 轴旋转 $-\alpha$ 度,再绕 y 轴旋转 $-\beta$ 度。而己知公式(7)和(8):

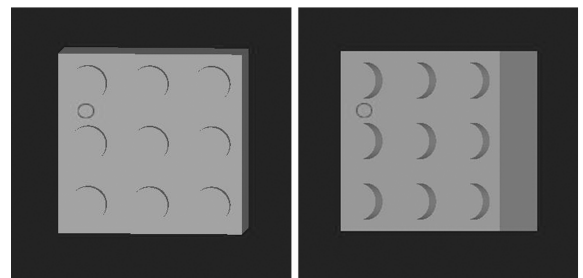
$$\sin(-\alpha) = -\sin(\alpha), \quad (7)$$

$$\cos(-\alpha) = \cos(\alpha). \quad (8)$$

所以,经过分析可知,欲达到反向旋转的效果,实际上只需要将旋转矩阵中所有正弦函数取反即可。观察 M_x 、 M_y 以及 M_{xy} ,可以发现,只需要将旋转矩阵 m 中的 $m[2]$ 、 $m[6]$ 、 $m[8]$ 、 $m[9]$ 取反,即可形成一个相对于原旋转矩阵的逆向旋转矩阵 $M_{\bar{x}}$ 。之后,按照 2.2 所述的矩阵与点乘法,将 $M_{\bar{x}}$ 左乘 $(x', y', z', 1)$,并得到一个正常化齐次坐标 $(X_{\text{final}}, Y_{\text{final}}, Z_{\text{final}})$,此即为所选点在物体真实坐标系的原始状态下的坐标。

3 结果与讨论

本文通过两个 stl 格式的模型文件对上述方法进行验证,实验环境为 windows 7 系统,IDE 为 Visual Studio 2012,编程语言为 C++。图 2(a)和图 2(b)分别是在不同旋转状态下,在模型的相同部位(即圆圈内)取点。结果如图 2(c)所示,考虑到手动点击会存在不可避免的误差,可以发现,

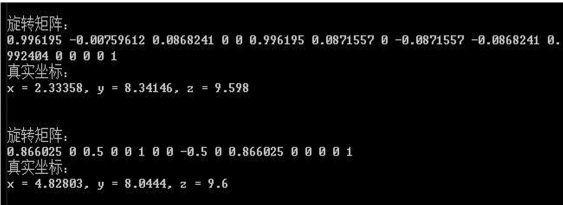


(a) 同部位取点状态 1

(a) Pickup in same site 1

(b) 同部位取点状态 2

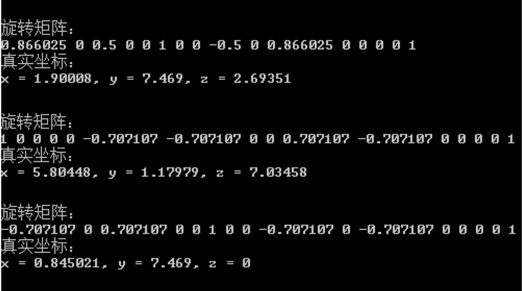
(b) Pickup in same site 2



(c) 拾取结果
(c) Result of pickup

图 2 模型表面同部位取点

Fig. 2 Pick-up in model surface on same site



(e) 取点结果
(e) Result of pickup

图 3 曲面及背面取点

Fig. 3 Pick-up incurved surface and back side

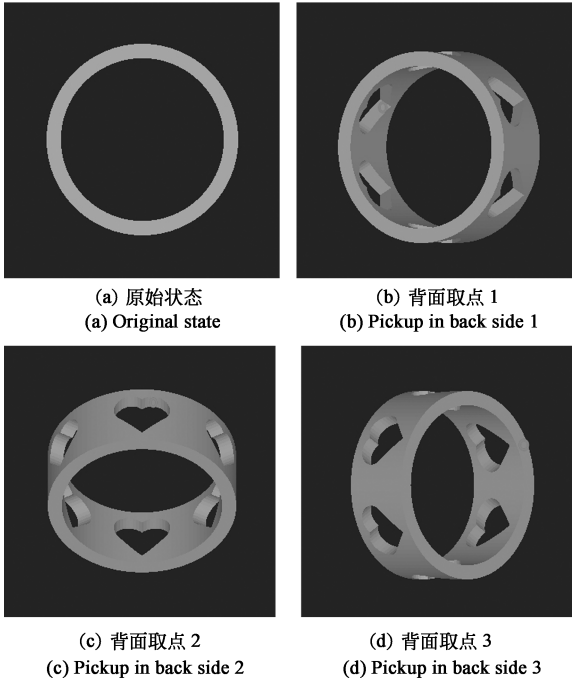
取点结果基本相同。

图 3 展示的是一个带有心形镂空的圆环，模型的初始状态如图 3(a) 所示，是一个平置圆环的俯视图，其侧面背面均不可见。图 3(b) 与图 3(c) 展示的是通过旋转，分别在圆环镂空的心形的上侧和下侧取点。图 3(d) 展示的是在圆环底部取点，拾取结果如图 3(e) 所示。

4 结 论

本文研究了一种基于 OpenGL 的三维模型点坐标获取的方法，解决了模型在不同旋转状态下，获取指定点坐标的问题，对其中的关键问题进行了深入的研究，分析并解释了旋转矩阵的特点，提出了一种有效的原始坐标点的获取方法。

另外，对于绝大多数模型文件，表面信息都被转化为三角图元或网格，在微观上看，面对的都是单个有限平面。所以视觉上的曲面或不规则面对本方法精度的影响很小。对于本方法，影响取点精度的主要因素有两个，一个是显示设备的分辨率，另一个是由手动点击带来的误差。在运用中，当遇到模型表面过于复杂，不便于选点，则可通过放大来解决，之后对获取的点乘以缩放系数即可解决。还可能遇到图形平移的情况，可通过反向移动的方法解决，比较简单，就不在此处细讲。



参 考 文 献：

[1] 郝继贵,张国全,唐大林,等. 一种新的基于 OpenGL 重绘式物体拾取方法[J]. 微型机与应用, 2005, 24(10): 53-55. ZHU J G, ZHANG G Q, TANG D L, *et al.* A new redrawing method for object pickup based on OpenGL[J]. *Microcomputer & Its Applications*, 2005, 24(10): 53-55. (in Chinese)

[2] 何健鹰,徐强华,游佳. 基于 OpenGL 的一种三维拾取方法[J]. 计算机工程与科学, 2006, 28(1): 45-46, 70. HE J Y, XU Q H, YOU J. A 3-D picking method based on OpenGL[J]. *Computer Engineering & Science*, 2006, 28(1): 45-46, 70. (in Chinese)

[3] 姚继权,李晓懿. 计算机图形学人机交互中三维拾取方法的研究[J]. 工程设计学报, 2006, 13(2): 116-120. YAO J Q, LI X H. Research on 3-dimension pick-up of human-computer interaction in computer graphics[J].

Journal of Engineering Design, 2006, 13(2): 116-120. (in Chinese)

- [4] 毕硕本,陆良虎,毕胜杰,等. 史前聚落遗址三维拾取与精确分布的研究[J]. 测绘通报,2013(10):15-19.
BI S B, LU L H, BI S J, *et al.* Research on three-dimensional pickup and precise distribution of prehistoric settlement site[J]. *Bulletin of Surveying and Mapping*, 2013(10): 15-19. (in Chinese)
- [5] 叶帅,游有鹏,邢永彦. 基于 OpenGL 的三维模型交互控制研究[J]. 机械设计与制造工程,2015,44(1):15-18.
YE S, YOU Y P, XING Y Y. Study on the interactive control of three-dimensional models based on OpenGL[J]. *Machine Design and Manufacturing Engineering*, 2015, 44(1): 15-18. (in Chinese)
- [6] 刘力强,周明全,耿国华. 一种平行透视下的三维拾取方法[J]. 西北大学学报:自然科学版,2002,32(1):39-42.
LIU L Q, ZHOU M Q, GENG G H. A 3D picking method based on parallel perspective model [J]. *Journal of Northwestern University: Natural Science Edition*, 2002, 32(1): 39-42. (in Chinese)
- [7] 冯宇平,戴明,孙立悦,等. 图像自动拼接融合的优化设计[J]. 光学精密工程,2010,18(2):470-476.
FENG Y P, DAI M, SUN L Y, *et al.* Optimized design of automatic image mosaic[J]. *Optics and Precision Engineering*, 2010, 18(2): 470-476. (in Chinese)
- [8] 祝敏. 基于 Open GL 的 LED 灯光情景仿真[D]. 哈尔滨:哈尔滨理工大学,2007.
ZHU M. The simulation of the LED lighting scene based on OpenGL[D]. Harbin: Harbin University of Science and Technology, 2007. (in Chinese)
- [9] 夏果,吕国强,胡跃辉. 一种真三维显示数据生成的方法[J]. 液晶与显示,2009,24(2):273-277.
XIA G, LV G Q, HU Y H. Data generation method of true 3D volumetric display[J]. *Chinese Journal of Liquid Crystals and Displays*, 2009, 24(2): 273-277. (in Chinese)

作者简介:施珂奕(1991—),男,江苏东台人,硕士研究生,研究方向为增材制造关键技术研究、数字图像处理技术。