OSG 多点触控自然用户接口框架①

吴学毅

(西安理工大学 印刷包装与数字媒体学院, 西安 710048)

摘 要: OSG 多点触控自然用户接口框架是在 Windows 多点触控技术基础上, 将触控事件的管理和处理与 OSG 的事件处理机制相结合, 形成了 OSG 的多点触控运行框架. 在与用户交互过程中产生手指触屏的原始数据, 根据这些原始数据定义所需的手势, 并将其与 OSG 中交互事件处理机制相结合, 完成利用多点触控对三维场景的交互操作. 基于以上原理, 分析了三维用户交互中主要的操作任务, 定义了符合三维空间操作认知的多点触控交互手势, 设计了相关手势的识别算法, 并通过实例应用的开发验证了这一原理和设计成果的正确性和可行性.

关键词: 多点触控; OSG; 自然用户接口; 三维应用; 手势识别

Multi-Touch Natural User Interface Framework in OSG

WU Xue-Yi

(Faculty of Printing Packaging Engineering and Digital Media Technology, Xi'an University of Technology, Xi'an 710048, China)

Abstract: OSG multi-touch natural user interface framework is based on the Windows multi-touch technology, in which the management and processing of touch event combines with OSG event processing mechanism, and forms the OSG multi-touch operation framework. According to the raw data that is produced by user's finger through touching screen, it defines the gestures needed, and combines it with interactive event processing in OSG to complete multi-touch for interactive operation in 3d scene. Based on the above framework, this paper analyzes the major operation tasks in 3D user interaction, defines the interactive gestures that conform to cognition of the 3D spatial operations, and designs the recognition algorithm of the related gestures. Finally, through developing an application example, the correctness and feasibility of design and cognition of multi-touch natural user interface framework in OSG are verified.

Key words: multi-touch; OSG; natural user interface; 3D application; gestures recognition

自然用户接口(Natural User Interface, NUI)是一种允许用户使用来自于其他系统的知识、方法与系统进行交互的用户接口[1]. 多点触控技术使得人与机器的交互变得更加便捷、自然, 已被广泛作为平板电脑、智能手机等移动设备很普遍的用户接口[2], 近年来也被应用于三维应用的交互控制与操作. OSG(OpenSceneGraph, 简称 OSG)作为一种先进的开源三维图形引擎, 因其具有优异的场景图组织、支持主流三维技术、移植性好、开发快速等特点而被广泛地应用于三维图形开发领域[3].

多点触控的研究主要集中在相关硬件设备的研制、手势的定义、识别以及支持多点触控软件框架 3 个方面. 目前三个最流行的多点触控软件框架包括:开源框架 TUIO(Tangible User Interface Objects), Microsoft 的 Windows7 SDK 和 PQ labs 多点触控 SDK[4]. TUIO 为多点触摸设备和软件定义了一个通用的协议和 API, 允许传输抽象的交互表面描述, 包括触摸事件和物体状态. TUIO 不提供手势分析. Windows7 的 SDK 和 PQ labs 仅提供了原始触控数据或者基本的手势支持, 不足以应对非常复杂的应用. 人们在多点触控手势识别上进行了大量研究, 如使用图像处理进行手势识别[5]等, 在二维空间取得了较好成果, 但针对三维空间图形应用的技术尚不成熟[6]. 2011 年 OSG 发布 3.0 版本开始支持多点触控软件框架,

① 基金项目:陕西省教育厅科研计划(2013JK1155)

收稿时间:2016-06-27;收到修改稿时间:2016-08-18 [doi:10.15888/j.cnki.csa.005687]

仅提供了窗口多点触控功能注册、多点触控事件处理、多点触控原始数据的相关类. 对于满足复杂三维空间交互操作的多点触控手势定义和实现至今仍无完整支持.

本文主要研究了 OSG 中多点触控自然用户接口的应用框架, 根据三维交互任务的需要分析了三维交互中所需的操作手势, 结合对 OSG 中有关多点触控原始数据的研究, 扩展了 OSG 中的手势识别类型, 利用 OSG 中事件处理机制和漫游器操作类, 实现了基于多点触控的三维交互操作, 并结合应用实例验证了上述框架的可行性、有效性和可扩展性.

1 OSG多点触控自然用户接口框架

OSG 是在 Windows 7 SDK 多点触控框架的基础上构建了自己的多点触控技术框架, 并完全继承了 Windows 7 SDK 中 WM_TOUCH 消息处理机制和主要的函数[6,7]. 在此基础上将多点触控事件和原始触控数据引入 OSG 的事件队列中进行管理和处理, 并将多点触控手势转变为鼠标事件类型, 在 OSG 刷新每一帧时使用事件适配器处理相应的事件, 完成三维场景对多点触控事件的响应. 其多点触控的框架如图 1 所示, 完成了从交互事件触发到交互结果显示的封闭处理.

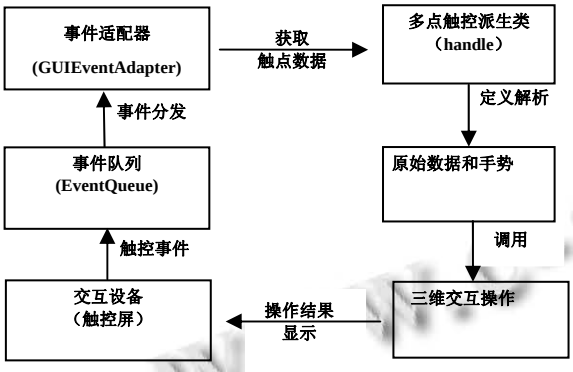


图 1 OSG 多点触控结构框架

交互设备是支持多点触控功能的触控屏, 作为触控事件的发起者, 同时也作为向用户展示触控操作结果的介质. 交互事件首先被 Windows 捕获, 并传递给 OSG 的事件队列[5]. 事件队列再将触控数据分发给 GUI 事件适配器. GUI 事件适配器是 OSG 中处理不同事件的主要类, 在提供给程序员的多点触控派生类事件处理函数 handle()中可以针对不同多点触控事件进行不同处理, 同时可以获取原始的多点触控数据. 然后根据手势与交互任务的映射关系, 调用三维漫游器类和设计好的三维交互操作, 即可完成三维空间的漫游和其他交互操作.

1.1 OSG 中多点触控事件处理

OSG 中对多点触控事件处理涉及 Graphics WindowWin32、EventQueue、GUIEventAdapter 等类, 其多点触控事件处理流程如图 2 所示.

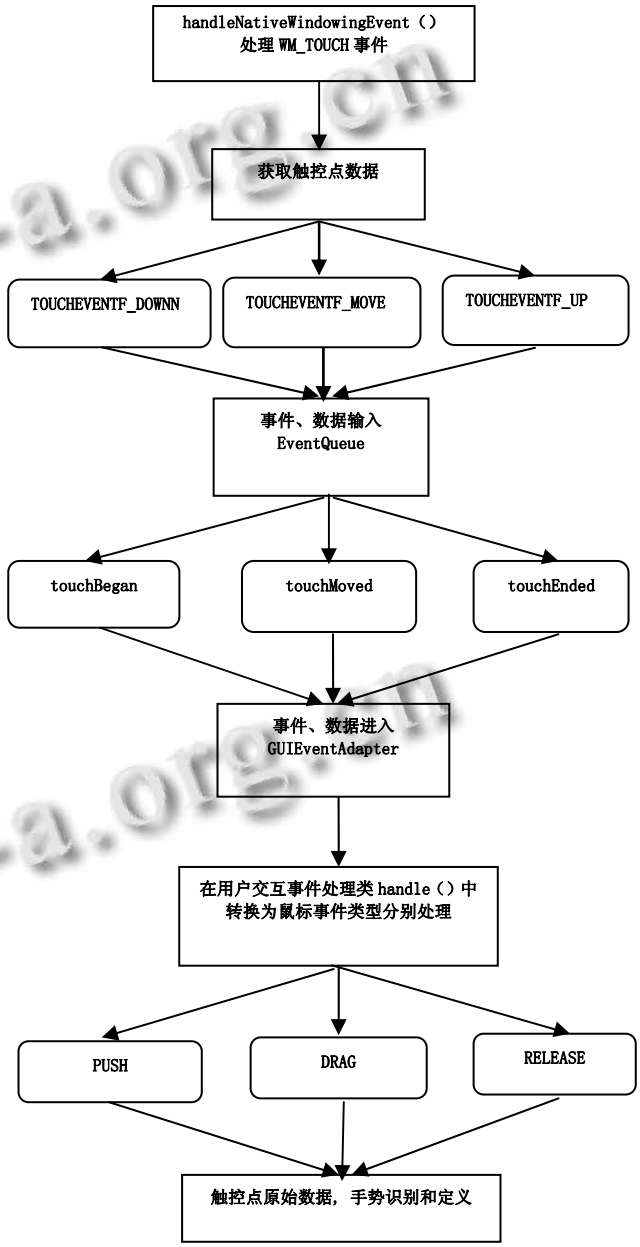


图 2 OSG 中多点触控事件处理流程

Windows 采用消息机制来完成事件的捕捉或处理, 而 OSG 则是在 GraphicsWindowWin32 类的

handleNativeWindowingEvent()函数中实现消息处理机制, 其中多点触控事件是通过对 WM_TOUCH 的捕获来处理的. 从 wParam 和 lParam 中获得触控数据, 并将触控事件类型分为 TOUCHEVENTF_DOWN、TOUCHEVENTF_MOVE 和 TOUCHEVENTF_UP3 种类型分别处理, 继而将事件和数据传入到 OSG 事件队列处理类. EventQueue 类可以理解为对交互事件进行添加和分发处理的集合, 它保存了一个 GUIEventAdapter 的队列, 并通过 touchBegan、touchMoved、touchEnded 方法向队列中新增触控事件和数据, 从而使得用户可以在自己的派生类中进行事件处理. 这一队列会在场景渲染的每一帧被遍历、执行和清理, 从而确保用户交互动作的顺利进行[6]. 在客户端, 通常由用户自定义一个派生自 GUIEventAdapter 的类, 并通过重载 handle 函数来处理各种事件的响应, 并将触控事件转化为鼠标的 PUSH、DRAG、RELEASE 事件类型进行不同处理, 而触控点原始数据也可传入其中, 为手势的识别和定义提供了数据基础.

1.2 多点触控数据流

在 OSG 中多点触控数据流是伴随着事件处理流程在不同阶段进行传递, 如图 2 所示, 并在不同阶段使用不同的数据结构进行存储.

在 GraphicsWindowWin32 中使用 Windows SDK 中通用多点触控数据结构体 TOUCHINPUT 来存储数据, 涉及触控点 ID(dwID)、触控状态(dwFlags)、触控点坐标(x, y)和时间等数据. 这些数据随事件处理流程被带入到 EventQueue 类中, 又随着不同事件的处理被传递到 GUIEventAdapter 类中. 在该类中使用 TouchData 类存储触控数据, 并提供了访问、遍历触控数据的函数. 其中存储的主要触控数据有触控点 ID(id)、触控状态(phase)、触控点坐标(x, y)和点击计数(tapCount)等. 每一触控事件发生时, OSG 均可获得这些数据, 在随后的事件发生时, 结构体中前一次事件的数据会丢失. 这些原始数据会被传递到用户自定义交互事件处理类中被用来处理多点触控事件. 在 OSG 中仅提供了一个多点触控轨迹球类 MultiTouchTrackballManipulator, 目前可以提供对 2 个手指的缩放、平移, 3 个手指同时点击回到家位置的简单多点触控操作支持, 尚不能满足三维空间复杂交互操作需求.

OSG 在底层提供了较为完善的多点触控交互功能框架, 但是在针对三维空间操作中的手势定义和功能实现上尚不完善. OSG 对多点触控手势的定义只有平移、缩放, 不能满足复杂应用三维空间人机交互的需要. OSG 未提供在原始数据基础上的手势识别算法, 另外由于三维交互任务复杂至今仍未对三维空间交互手势与任务间映射关系形成公认的结论, 因此, 解决三维空间的多点触控操作仍须进行大量研究.

2 三维交互任务与手势

由于三维空间交互操作的复杂性和需求的多样性, 目前对三维空间交互操作任务尚无明确的划分[8], 加之二维平面触控操作向三维空间进行映射存在操作性和纵深空间缺失问题, 因此对三维空间交互操作的二维触控手势尚无明确定义.

对于三维交互任务, Bowman[9] 认为在 3D/VR 交互系统存在 4 种普遍的交互任: Navigation, Manipulation, Selection/Pick, System Control. 而田丰等将其进一步划分为场景浏览、实体选择、实体释放、实体操作、场景数据访问和实体数据访问 6 种操作[10]. 作者认为以上对交互任务的划分可看作是三维交互操作的元操作任务, 除此之外, 由于应用目的不同还存在大量更为复杂的交互操作任务, 如三维地理空间中的距离丈量、通视、地图线路规划等. 当然, 这些复杂交互操作任务, 最终仍可分解为多个元操作任务的组合. 本文参考 Microsoft 触控手势定义[11], 结合对三维空间交互操作任务的分析和人们使用手势操作习惯的认识, 总结了表 1 中三维操作任务和对应的多点触控手势定义. 所有操作总体上可分为针对场景对象和被关注实体对象, 对于场景的操作是通过改变视点来实现, 而对于实体的操作前提是已有实体被拾取, 继而对其进行几何变换操作. 这种手势及操作方式继承了使用鼠标进行三维空间交互操作的一般过程.

表 1 给出了在通用三维交互应用中的主要操作任务和手势定义, 在一些更为专业化的交互中还会存在更多复杂操作. 一方面可以通过将一些简单手势组合为符合交互操作语义的复杂手势集合来实现复杂操作[12,13], 另一方面可以定义其他未使用手势来表达复杂操作.

表 1 三维操作任务及对应的多点触控手势定义

操作对象类型	任务类型	具体操作	手势	手势定义
场景主动浏览/实体控制	视点平移/实体平移	左/右平移		单指沿水平方向向左或向右滑动
		前/后平移(纵深方向)		一个手指按住不动, 另一手指在垂直方向上向上或向下滑动
		升/降操作		单指沿垂直方向向上或向下滑动
	场景缩放/实体缩放	放大		使用2个手指由收缩滑动分开
		缩小		使用2个手指收缩滑动到一起
	场景旋转/实体旋转	绕X轴旋转/俯仰		双指并拢沿顺时针或逆时针方向画弧
		绕Y轴旋转/改变方向		双指按压并沿顺时针或逆时针方向旋转
		绕Z轴旋转/滚动		一指按压, 另一指绕按压手指顺时针或逆时针方向画弧
场景/实体的拾取、释放	拾取/释放	拾取		单指快速在同一位置点击2次
		释放		单指在被拾取对象上点击1次
场景/实体数据访问	对象数据输入/输出	拾取+单击		先单指双击拾取对象, 再次单击显示对象的属性输出/输入界面
场景/实体范围选择	划定一个区域	系统控制+多次单击		由菜单列表指定选定区域类型, 用单指点击主要特征点给出封闭区域
场景操作	视点返回	家位置		三指同时按下

3 手势的识别

OSG 在每一帧处理多点触控事件时均会向用户自定义多点触控事件处理类的 handle()函数中传递多点触控数据, 包括触控点 ID(id)、触控状态(phase)、触控点坐标(x, y)和点击计数(tapCount)等数据. 每个接触触控屏的手指都会被赋以唯一的 ID, 每个手指在按下触控屏时, 其 phase 为 TOUCH_BEGAN, 随后会变成 TOUCH_MOVED, 在手指离开触控屏时会变为 TOUCH_ENDED. 由于触控屏的灵敏性, 会出现多个手指的 phase 不同步的现象. 同时, 对于触控坐标点的位置不能按绝对坐标值进行判断, 可以设置坐标灵敏度阈值(coTh)和角度灵敏度阈值(anTh). 这样只要 2 次敲击屏幕坐标位置在阈值范围内, 即可认为是点击了的同一位置[12].

为了完成表 1 中设计的多点触控手势识别, 结合 OSG 中获得的触控数据及对触控数据生命周期的分析, 在已有触控数据存储结构体基础上增加了手指旋转角度(angle), 同时为了识别单指双击操作还增加了

P(x,y)、time 用于存储单指点击结束时手指单击的位置和时间[13]. 则相应的识别算法如下:

设data为存储当前帧多点触控数据的结构体,last为存储上一帧多点触控数据的结构体. P_i 为当前帧触点i的坐标, $P_i^{'}$ 为前一帧触点i的坐标, i=1, 2, …为触控点个数. $\Delta x_i = P_i^{'}.x - P_i.x$, 为前后两帧同一手指在x方向的坐标差. $\Delta y_i = P_i^{'}.y - P_i.y$, 为前后两帧同一手指在y方向的坐标差. $\Delta(P_i^{'}, P_i)$ 为第i个触点前后2帧距离差. $\eta = P_2 - P_1$ 为当前帧第一和第二个触点间距离, $\eta^{'} = P_2^{'} - P_1^{'}$ 为前一帧第一和第二个触点间距离. θ为单指触点到坐标原点形成线段与Y轴间的夹角. Num为phase等于TOUCH_ENDED的触控点计数.

(1) 获取系统当前时间T, 获得当前帧多点触控数据data.

(2) 若当前帧data的触控点数为1且上一帧last的触控点数为1, 则触控类型为单指触控. ①若 $\Delta x_i > \Delta y_i$, 则当前手势为左/右平移操作; ②相反, 则当前手势为升/将操作; ③若有对象处于被拾取状态且 $\Delta(P_1^{'}, P_1)$ <coTh, 则当前手势为释放拾取对象操作; ④若data.phase等于TOUCH_ENDED, time=T, P=P1(一次单击手势结束, 记录当前时间为判断下一次单击手势作准备); ⑤若 data.phase 等于 TOUCH_ENDED 且 $\Delta(P, P_1)$ <coTh, T- time<1秒（以1秒作为单指双击时间的间隔）, 则当前手势为单指双击操作.

(3) 若当前帧 data 的触控点数为 2 且上一帧 last 的触控点数为 2,则触控类型为双指触控. ①若 $\Delta(P_1^{'}, P_1)$ <coTh 且 $\Delta x_2 < \Delta y_2$, 则当前手势为前/后平移操作; ②若$|\eta - \eta^{'}|$>1, 则当前手势为缩放操作; ③若 η < 两指并拢最大阈值, 且第二指的phase=TOUCH_BEGAN, 则 angle=0; 第二指的phase=TOUCH_MOVED, 则计算 θ, 并使 angle 等于前后 2 帧 θ 之差, 则当前手势为绕 X 轴旋转操作; ④若第一、二指的 angle 均不为 0, 则当前手势为绕 Y 轴旋转操作; ⑤若 $\Delta(P_1^{'}, P_1)$ <coTh 且第二指的 angle 均不为 0, 则当前手势为绕 Z 轴旋转操作.

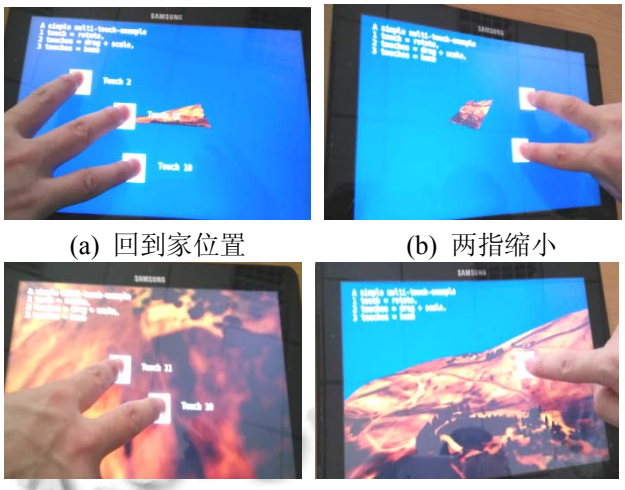
(4) 若当前帧 data 的触控点数为 3 且上一帧 last 的触控点数为 3, 则触控类型为三指触控, 则场景视点回到家位置.

(5) last=data. 若 phase 等于 TOUCH_ENDED, 则 Num++.

(6) 若 Num=data 的触控点个数, 则一次完整的触控过程完成, last=NULL.

4 基于多点触控的OSG三维交互应用

通过对 OSG 多点触控解决框架和三维空间交互任务的分析, 利用所设计的多点触控手势识别算法, 在 VC++的 Win32 控制台框架下使用 OSG3.1 版本作为三维交互引擎和多点触控操作平台, 开发了如图 3 所示的三维多点触控交互平台, (a)图是三个手指同时点击使三维地图场景视点回到家位置, (b)、(c)图是 2 个手指的操作三维地图场景缩小、放大, (d)图是对三维地图场景进行了平移的操作. 实践表明, 对 OSG 多点触控框架的认识是正确的, 所设计的三维空间多点触控手势识别算法是有效的, 可满足较为通用的三维空间多点触控交互操作的需要, 体现了多点触控人机交互技术更加简单、方便、高效、直观的特点.



(a) 回到家位置　(b) 两指缩小

(c) 两指放大　(d) 一指平移

图 3　多点触控手势对三维地图对象操作

5 结语

目前在 OSG 中常见的人机交互方式仍是鼠标、键盘, 操作方式不够人性化, 多点触控交互方式向现实中的双手操作方式实现了有效靠拢, 尤其是向自然用户接口提供了底层支持[14]. 但仍有很多问题亟待进一步深入研究: (1)由于多点触控操作是在二维平面上进行, 而三维空间操作本质上存在 6 个自由度, 如何解决二者之间的差异, 设计出规范、为大家公认的交互手势; (2)由于二维平面操作需要向三维空间操作映射, 二者存在操作粒度不统一的问题[15]. 如何使二者的操作能够协调一致值得研究; (3)由于存在多种多点触控

解决方案, 用户对手势的认知不统一, 急需研究一种中间件技术来解决二者的差异问题.

参考文献

1 Norman DA. Natural user interfaces are not natural. Interactions, 2010, (17): 6–10.

2 Olafsdottir H, Appert C. Multi-touch gestures for discrete and continuous control. International Working Conference on Advanced Visual Interfaces (AVI). Como, Italy. May 2014.

3 王锐,钱学雷.OpenSceneGraph 三维渲染引擎设计与实践.北京:清华大学出版社,2009.

4 Yao JL, Fernando T, et al. A multi-touch natural user interface framework. 2012 International Conference on Systems and Information (ICSAI 2012). 2012. 499–504.

5 Pal P, Vikas S, Raju P. Design and implementation of low-cost multi-touch surface computing using image processing. IOSR Journal of Computer Engineering, 2014, 16(2): 48–56.

6 Hlavacek F, Kovarova A. Multi-touch interaction techniques to control 3D objects on a smartphone screen. Proc. of the 16th International Conference on Computer Systems and Technologies (CompSysTech '15). NY, USA. 2015. 350–357.

7 Kiriaty Y. MultiTouch capabilities in Windows 7, MSDN Magazine. 2009. http://msdn.microsoft.com/en-us/magazine/ee336016.aspx.

8 姚垚.基于多点触控平台的的三维可视化系统交互设计与评价研究[硕士学位论文].郑州:解放军信息工程大学,2012.

9 Bowman DA, Kruijff E, LaViola JJ, et al. Mine, Ivan Poupyrev. 3D user interface design: Fundamental techniques, theory, and practice. In: Akeley K, ed. SIGGRAPH 2000 Course Notes 36. Addison Wesley, 2000. http://www.siggraph.org.

10 田丰,戴国忠.三维交互任务的描述和结构设计.软件学报,2002,13(11):2099–2105.

11 Windows Touch Gestures Overview. https://msdn.microsoft.com/en-us/library/dd940543(VS.85).aspx.

12 迟健男,等.多点触摸人机交互技术综述.智能系统学报,2011,6(1):28–37.

13 张国华,等.基于多点触摸的交互手势分析与设计.计算机应用研究,2010,27(5):1737–1739,1752.

14 Wang CB, Dai JQ. 3D multi-touch recognition based virtual interaction. 2010 3rd International Congress on Image and Signal Processing (CISP2010). 2010. 1478–1481.

15 Ohnishi T, Lindeman R. Multiple multi-touch touchpads for 3D selection. IEEE Symposium on 3D User Interfaces 2011. 2011. 115–116.