

基于 MVC 模式的三维几何画板的设计

曾东海¹, 陈国华²

(1. 广东科学技术职业学院, 广东 广州 510640;
2. 广东药学院 计算机科学系, 广东 广州 510633)

摘 要:历史上,美国 Key Curriculum Press 公司最早提出了《几何画板》的概念模型并开发了一个具体的应用软件。该软件系统在 CAI 领域产生了巨大影响。《几何画板》虽然在表现平面几何教学方面有出色表现,但在空间(三维)几何表现方面则不尽人意,因为该软件是基于微软的 GDI 二维图形库来开发设计的。而空间几何关系的展现又是中小学数学教学中的重点和难点,最需要使用计算机辅助教学手段帮助学生直观地加以理解。笔者开发的《三维几何画板》以弥补《几何画板》在空间表现方面的不足,文中介绍了该软件体系结构中 MVC 设计模式的一种变体“文档-视图设计模式”的应用方法。说明了在某些情况下,为什么使用 MVC 模式的文档-视图变体可能会更加合适。并且详细阐述了文档-视图设计模式的实现方法及其各部分与模型-视图-控制器模式之间的对应关系。

关键词:设计模式;MVC 模式;文档-视图模式;软件体系结构

中图分类号:TP31

文献标识码:A

文章编号:1673-629X(2013)12-0240-04

doi:10.3969/j.issn.1673-629X.2013.12.057

Design of 3D Geometer's Sketchpad Based on MVC Mode

ZENG Dong-hai¹, CHEN Guo-hua²

(1. Guangdong Institute of Science and Technology, Guangzhou 510640, China;
2. Dept. of Computer Science, Guangdong Pharmaceutical Univ., Guangzhou 510633, China)

Abstract: In history, the Key Curriculum Press in USA is the first to propose a conceptual model of the Geometer's Sketchpad and develop a specific application software, which has had a tremendous impact in the field of CAI. Geometer's Sketchpad is excellent in the plane geometry teaching, but it is unsatisfactory in space (3D) geometry performance, because the design is based on Microsoft GDI 2D graphics library. Spatial geometric relationship is the emphasis and difficulty in mathematics teaching at middle and primary school, and it most needs to use computer-aided teaching methods to help students understand intuitively. Develop 3D Geometric Sketchpad to make up for the lack of space performance of the Geometer's Sketchpad, it has introduced the applications of a variant of MVC Design Pattern—the Doc/View Design Pattern. It has pointed out why in some cases the Doc/View Design Pattern is more suitable than the MVC Design Pattern. It has illustrated the implementing methods of the Doc/View Design Pattern in detail and had also pointed out the correspondences between the various parts and MVC pattern.

Key words: design pattern; MVC mode; Doc/View pattern; software system architecture

1 MVC 模式概述

在早期交互式程序设计过程中,如果不注意对数据处理功能和显示功能的耦合处理,常常会导致程序代码纷繁复杂、难以维护。其实,即便现在,很多 VB, Delphi 等使用 RAD 工具开发的程序仍有这类问题。甚至连使用当今热门的开发工具 C#, Java 等,如果设计人员不注意,有时候也会出现把业务逻辑写在显示模块中的现象。

模型-视图-控制器(MVC)是 20 世纪 80 年代由 Trygve Reenskaug^[1]等首创并第一个在 Smalltalk-80 环境下实现的一种软件设计模式,现在已得到广泛使用。它是一种将表示逻辑和业务逻辑分离的设计模式,旨在分离用户界面显示,用户输入控制和底层的信息应用。MVC 模式为许多交互式系统和具有图形用户接口的软件系统的应用框架提供了基础^[2-5]。微软的基础类库(MFC)本质上遵行 MVC 原则。

收稿日期:2013-01-19

修回日期:2013-04-26

网络出版时间:2013-08-28

基金项目:广东省自然科学基金项目(S2011010001841)

作者简介:曾东海(1968-),女,硕士,副教授,研究方向为软件工程等。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20130828.0837.024.html>

MVC 设计模式的出现不仅实现了数据处理功能模块和显示功能模块的分离,同时它还提高了应用系统的可维护性、可扩展性、可移植性和组件的可复用性。尽管 MVC 设计模式很早就提出,但在具体项目的开发中引入 MVC 却并不容易。要将数据处理代码和显示代码完全分离很多时候难以实现。

MVC 设计模式包括三个组成部分,分别是:Model (模型) 组件、View (视图) 组件和 Controller (控制器) 组件。表 1 是对这些组件的一个简单描述。

表 1 MVC 设计模式

组件	描述
模型 (Model)	模型是应用程序的主体部分。表示业务数据,或者业务逻辑,用于封装数据对象
视图 (View)	视图是应用程序中与用户界面相关的部分,是用户看到并与之交互的界面。作为模型的一种表示,显示数据对象的当前状态
控制器 (Controller)	控制器就是根据用户的输入执行相应操作的接口,它用来操作模型 (Model) 和数据对象,控制用户界面数据的显示和 Model 对象状态的更新

三个组件之间的关系如图 1 所示。

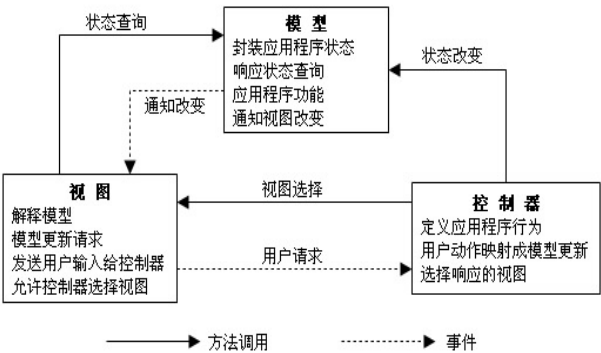


图 1 MVC 组件类型的关系和功能

使用 MVC 有其不可替代的优势:

可以将表示层和处理层相分离,表示层代码变化不会影响模型层和控制器层代码。这样就提高了系统的可靠性,也使得系统更加易于维护和修改。

可以用不同的视图来访问相同的数据服务。提高了系统的可重用性。降低了系统的生命周期成本。

尽管使用 MVC 设计模式有许多好处,但严格按这种设计模式进行系统设计在某些情况下会有很多困难,甚至不一定合适。这时需要用它的各种变体。

2 MFC 的 Doc/View (文档-视图) 设计模式

MFC (Microsoft Foundation Class) 的文档-视图设计模式可以看作是 MVC 的一种变体。它放松了视图与控制器的隔离,将窗口显示与事件处理紧密交织在一起,也就是将 MVC 中的视图与控制器的职责结合到

一个组件“视图”之中,实现了系统的用户接口。而文档组件则对应于 MVC 中的模型组件。

在有些情况下,使用 MVC 模式的文档-视图变体可能会更加合适。例如,在开发交互式图形设计系统 (类似于 CAD 系统) 的时候,就很难将视图功能与控制器功能相分离。假如你用鼠标选中了一个对象,并想对它进行拖动处理。被选中的对象需要改变它的外观-以一种“被选中色”(如银色)进行显示。对象选择本身是一个控制器动作,它不会导致模型的变动。被选中的对象将仅仅用作下一个控制器动作(拖动)的输入,只是它的外观必须改变-用被选中色进行显示。在一个严格的 MVC 模型中,这种纯粹的控制行为当然应该由控制器自身来完成。但改变显示颜色似乎又是一种“视图行为”。由控制器自身来完成这种视图行为当然就破坏了 MVC 模型的纯粹性。如果选择让控制器与视图进行协作来完成这一行为,就将增加一些不必要的实现开销。在这种情况下,如果把视图与控制器的功能统一在一个组件之中,即使用文档-视图模型,则该问题可以得到圆满解决。其代价是:将失去输入输出功能的独立性。

使用 MFC 的文档-视图模型,主要有如下几方面的特点需要关注^[6-7]:

(1) 文档对象主要用于存取和处理数据,视图则负责用不同的方式对数据加以表象。

(2) 文档对象、视图对象和框架窗口对象与一个文档模板类相关联,该文档模板类的基类对象是 CDocTemplate,并在应用程序初始化时进行构造。

(3) 一个应用程序通常只有一种文档模板,若需要也可以有多个文档模板。文档实例由文档模板生成并可以有多个视图与之关联。不过在某一时刻只能有一个活动的文档、视图和框架。

3 文档-视图模式的实现方法

为了体验文档-视图设计模式的价值,以《三维几何画板》的设计为例,来说明它在一个具体软件设计过程中的实现方法。《三维几何画板》不仅可以制作立体几何课件,也可以制作平面几何课件,因为空间包含了平面。甚至有时也可用于制作某些物理、化学等教学课件。它是基于 MFC 的文档-视图设计模式来设计的一个应用程序,这类 Windows 应用程序一般主要由四个大类构成:

(1) 应用程序类:代表一个应用程序的入口,它负责对应用程序进行初始化,并创建与之相关联的框架窗口。

(2) 框架窗口类:Windows 下的应用程序一般都需要在控制台上开一个属于自己的窗口,它的所有输入/

输出信息和用户交互接口都将限制在该窗口之中。

(3) 文档类: 主要用来定义、存储并管理应用程序的数据, 包括从永久性存储(通常指磁盘文件)中加载数据或将其存储到永久性存储介质中。

(4) 视图类: 视图用来显示文档中的数据, 并作为用户和文档之间全部输入、选择和文档编辑的中介物。通常与单一文档相关联。但一个文档可能与多个视图相关联。

它们之间的关系一般如图 2 所示。

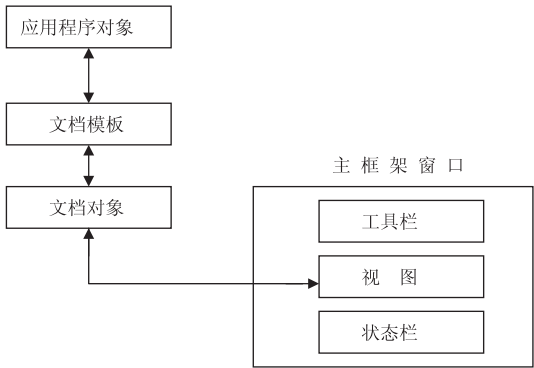


图 2 MFC 的文档-视图模式下四个主要类之间的关系

其中应用程序类和框架窗口类一般都由开发平台提供的“精灵”自动生成, 开发人员极少需要对它们进行修改。

绝大多数情况下, 用户只需要处理文档类对象和视图类对象的代码。

3.1 文档类

文档对象用于存储一些常见的几何形体对象, 包括二维和三维的对象。其中二维的几何对象包括点、线、矩形、多边形、圆、椭圆和其他圆锥及三角函数曲线、自由手绘曲线等。三维几何对象则包含长方体、四棱锥以及常见的二次曲面(如: 球面、椭球面、圆锥面、环面、圆柱面等), 也包括旋转面、八面体、十二面体及由函数生成的自由曲面等。此外, 还包括标注文字和轮廓文字, 以及通过 3D Studio 和 AutoCAD 创建的对象^[8-9]。

这些对象的结构如图 3 所示。

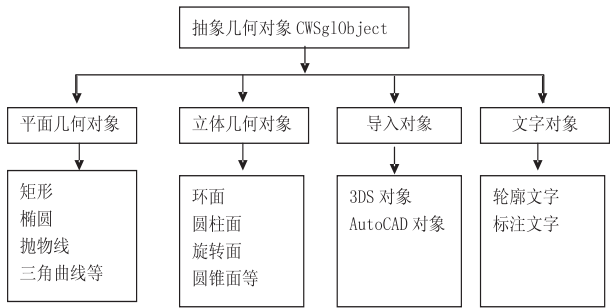


图 3 类层次结构

而文档类对象本身则是抽象几何对象的一个容

器, 它包含一个抽象几何对象数组:

```
typedef CTypedPtrArray<CObArray, CWSglObject * > CWSglObArray; // 数组定义

class CWSGeom3dDoc : public CDocument
{
protected: // create from serialization only
    CWSGeom3dDoc();
    CWSglObArray m_WSglObArray; // 数组成员
    .....
}
```

3.2 视图类

绘制三维图形的视图类对象主要是在需要更新用户接口时将显示列表绘制出来:

```
CGLDispList KDDispList[100]; // 显示列表定义
void CWSGeom3dView::OnDrawGL()
{
    KDDispList[0].Draw(); // 显示列表绘制
}
```

3.3 控制器方法

由于在 MFC 的文档-视图模型中没有单独的控制器类, 所有与控制器有关的方法只能表现为视图类的成员函数。这些方法主要表现为对鼠标和键盘的消息响应。

左鼠标按下事件: OnLButtonDown (UINT nFlags, CPoint point);

鼠标移动事件: OnMouseMove (UINT nFlags, CPoint point);

左鼠标抬起事件: OnLButtonUp (UINT nFlags, CPoint point);

键盘按下事件: OnKeyDown (UINT nChar, UINT nRepCnt, UINT nFlags);

右鼠标按下事件: OnRButtonDown (UINT nFlags, CPoint point);

键盘抬起事件: OnKeyUp (UINT nChar, UINT nRepCnt, UINT nFlags)

响应这些消息的结果主要是创建将在视图中绘制的显示列表, 创建方法如下:

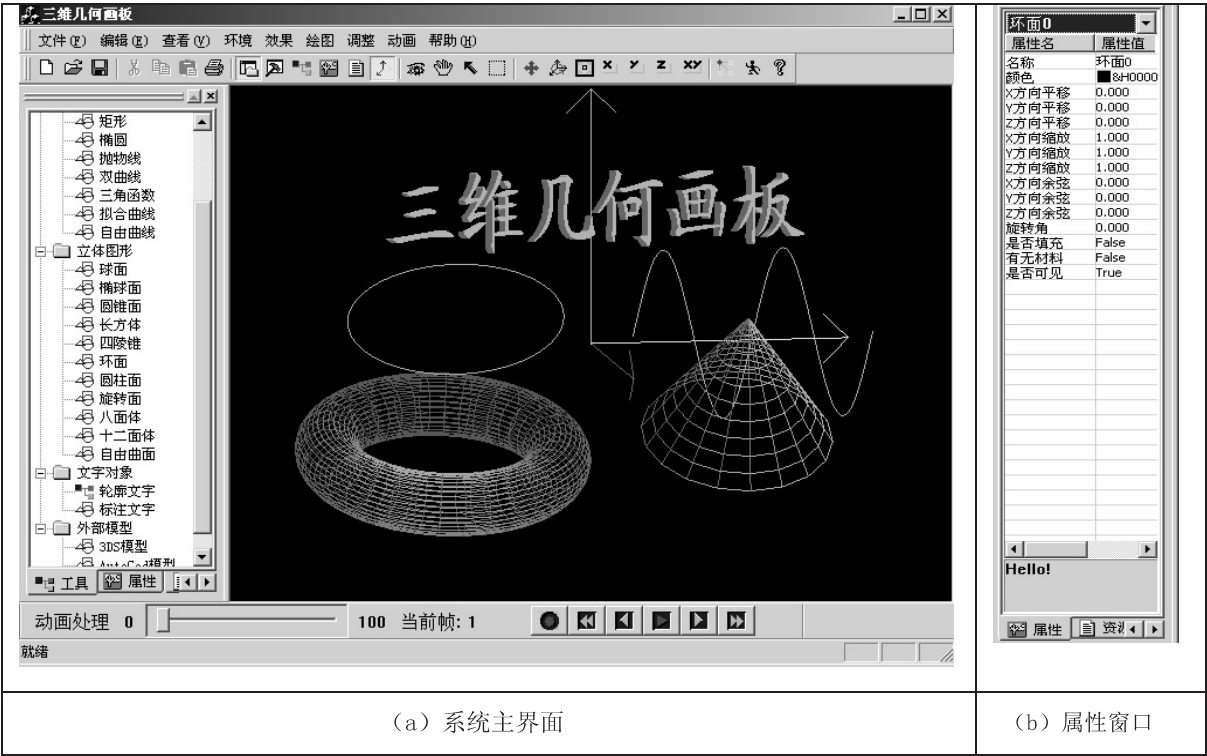
```
void CWSGeom3dView::BuildKDDispList (BOOL bImmediateExec, int ithFrame)
{
    CWSGeom3dDoc * pDoc = (CWSGeom3dDoc *) GetDocument();
    KDDispList[ithFrame].StartDef (bImmediateExec); // 开始定义显示列表
    if (pDoc != NULL)
    {
        CTypedPtrArray<CObArray, CWSglObject * > & glObList = pDoc->m_WSglObArray;
```

```
int i;
for(i=0;i<glObList.GetSize();i++)
{
    glObList.GetAt(i)->draw(ithFrame); //循环往显示列表中绘制几何对象
}

KDDisplist[ithFrame].EndDef(); //显示列表定义结束
}
```

4 使用文档-视图模式的实例

《三维几何画板》作为一个 Windows 系统下的立体几何动态课件制作工具,为广大教师制作三维课件提供了便利。使用《三维几何画板》可以利用鼠标简单、直观地创建一些常见的立体几何对象以及在三维空间中呈现的平面几何对象。此外,无论平面几何对象还是立体几何对象都可以作为动画单元,分别为它们定义轨迹动画或关键帧动画^[10-12]。操作方式如图 4 所示。



(a) 系统主界面

(b) 属性窗口

图 4 系统操作主界面

5 结束语

文中使用一个三维图形系统的开发过程作为案例,详细阐述了软件体系结构中 MVC 设计模式的变体文档-视图设计模式的应用方法。特别是这种设计模式在 CAI 软件平台开发中的应用。这种来自于工程设计一线的工作实践的总结性陈述,期望能对有志于 CAI 软件设计的同仁有一定帮助和借鉴作用。

参考文献:

[1] Reenskaug I T, Wold P, Lehne O A. Working with objects; the OOram software engineering method [M]. [s. l.]: Manning Publications Company, 1996.

[2] Donald H, Baker P. Computer graphics C version [M]. Beijing: Tsinghua University Press, 1998.

[3] 任中方, 张 华, 闫明松, 等. MVC 模式研究的综述 [J]. 计算机应用研究, 2004, 21 (10): 1-4.

[4] Hu Tianlei, Chen Gang. Adaptive XML to relational mapping: an integrated approach [J]. Journal of Zhejiang university sci-

ence A, 2008, 9 (6): 758-769.

[5] 张 原, 张 昭, 刘 蕊. 基于 MVC 设计模式的虚拟实现平台模块化设计 [J]. 计算机工程与科学, 2013, 35 (8): 125-129.

[6] 李 薇, 徐国标. OpenGL 3D 入门与提高 [M]. 成都: 西南交通大学出版社, 1998.

[7] 匡天君, 滕远道, 王 乘, 等. 基于 MFC 和 OpenGL 三维图形的开发 [J]. 微计算机信息, 2004 (6): 115-116.

[8] 廖朵朵, 张华军. OpenGL 三维图形程序设计 [M]. 北京: 星球地图出版社, 1996.

[9] 陈国华. 三维连续图形变换的一类算法与实现 [J]. 中国图象图形学报, 2001, 6A (12): 1240-1243.

[10] 陈国华. 三维几何画板的研究与设计 [J]. 现代教育技术, 2006, 16 (4): 66-69.

[11] 刘胜利. 几何画板与微型课件制作 [M]. 北京: 科学出版社, 2004.

[12] 陶维林. 几何画板实用范例教程 [M]. 北京: 清华大学出版社, 2001.

基于MVC模式的三维几何画板的设计

作者：曾东海， 陈国华， ZENG Dong-hai， CHEN Guo-hua

作者单位：曾东海, ZENG Dong-hai (广东科学技术职业学院, 广东 广州, 510640)， 陈国华, CHEN Guo-hua (广东药学院 计算机科学系, 广东 广州, 510633)

刊名：计算机技术与发展

ISTIC

英文刊名：Computer Technology and Development

年，卷(期)：2013(12)

本文链接：http://d.g.wanfangdata.com.cn/Periodical_wjtz201312057.aspx