仓储仿真三维图元管理平台

张世辰,张翔立,黄金国
(华中科技大学 机械科学与工程学院 工业工程系,湖北 武汉 430074)

摘 要:在仓储仿真项目的开发中,图元的创建和三维场景的布置是重要的步骤。为了简化仓储仿真项目的开发,提出一种仓储三维仿真图元管理的平台方案。该平台模型由4个关键模块组成。并在WPF下实现了关键模块,结果表明采用该平台的相应模块能够实现仿真图元对象的快速创建、仿真图元的高效管理,方便地为开发基于WPF平台的仓储仿真项目提供合适的图元对象。因此该平台能够简化仓储仿真项目的开发步骤,节省开发时间。

关键词:仓储仿真;三维仿真;图元管理

中图分类号:TP3 **文献标识码**:A **文章编号**:1674-7720(2013)19-0088-03

The platform for the management of 3-D graphics primitive in storage simulation

Zhang Shichen, Zhang Xiangli, Huang Jinguo
(Department of Industrial Engineering, School of Mechanical Science & Engineering, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: The creation of 3-D graphics and arrangements of 3-D scene are important to the development of the storage simulation project. In order to simplify the arrangements of 3-D scene, decline the time spent in layout, this article has presented a design scheme of the platform for the management of storage 3-D simulation graphics primitive. The platform consists of four modules. And most of the key modules have been implemented under WPF. The result proves that corresponding modules have realized the fast creation and efficient management of graphics primitive, as well as efficient and simple means to provide the 3-D graphics primitive used in the development of storage simulation project based on WPF. So this platform can simplify the development and save a considerable time of storage simulation project.

Key words: storage simulation; 3-D model; graphics primitive management

计算机仿真是应用电子计算机对系统的结构、功能和行为以及参与系统控制的人的思维过程和行为进行动态地模仿,进而得出数量指标,为决策者提供有关这一过程或系统的定量分析结果,作为决策的理论依据。

以仓库仿真为例,仓库管理人员可实时掌握货物信息,直观地了解当前仓库的状态。相较于二维图表,三维模型在仿真方面有着很大的优势。三维的图元需要有图形用户界面GUI(Graphic User Interface)的支持,在Windows系统平台上从事图形用户界面程序开发的工具历经了Win32->MFC(及同类产品)->ActiveX/COM/Visual Basic->Windows Forms的变迁,从2007年开始微软推出了新一代GUI开发工具Windows Presentation Foundation,并把它作为未来十年Windows平台GUI开发的主要技术。

1 平台设计思想

1.1 平台设计的出发点

在仓储物流系统中,包括入库台、传送带、叉车、堆垛机、AVG小车、货架、货物、缓冲区临时堆场等资源。仓储仿真应用既需要展现实际仓库动作的动态信息(例如:堆垛机的实时运动,货架上是否有货物,当前入库、出库的货物);也要能够显示仓库的静态信息(例如货物的数量信息、供应商/客户信息等)。因此,仿真应用的图元不仅仅包含要显示的模型信息,从面向对象的角度要把应用中的图元对象化,增加相应的属性使之能够描述

现实中的仓库资源[1]。

开发仿真应用首先需要创建图元，一般通过三维软件建模获得，根据图形用户界面——GUI 的标准不同，直接使用或者进行转换生成符合要求的图元文件，此时的图元只具有几何信息，为了实现应用中的功能，还需要增加相应的属性，使之成为一个完整的图元对象，得到所有图元对象后需要设置每个模型的位置，即完成场景布置。完成这些步骤往往需要多个工具的参与，工具的切换增加了开发的繁琐，更重要的是这些步骤中缺乏对已有图元的管理，对于一些相似的图元，重复制作浪费时间和精力，故需要将这些图元管理起来，建立图元库。因此，本文提出的平台模型将这些步骤封装成相应模块，并与其他模块联合简化开发步骤。

1.2 平台的功能分析

仿真图元的加载。单独开发出绘图模块工作量较大，现有的三维软件造型功能已十分完整，并能支持如 obj 等中间图形文件，使用三维软件（pro/e、SolidWorks、3DMax 等）创建图元，生成的图形文件需要由平台的相应模块将其转换为平台可用图元。

仿真图元的对象化。被模拟的对象除了几何信息外，应具有很多特有的属性，比如货物的基本信息，因此平台中需要创建图元对象，使之更贴近现实仓储资源。

仿真图元对象的编辑与管理。平台需要实现图元对象的存储及属性修改以满足相似图元的重复利用提高开发效率，对于已保存的图元，平台也需要提供基本管理的功能模块[2]。

图元输出功能，平台的最终目的是为仓储仿真应用提供图元，因此需要输出完整的图元文件。该图元文件中包含了图元的几何信息确保模型的准确外形；包含了坐标属性使模型显示在场景的正确位置；包含图元的属性定义，使图元对象携带现实仓库中资源的信息。

2 平台结构及模块分析

2.1 平台结构

平台采用典型的三层架构，图 1 是平台的内部结构，表示层负责把仿真数据和操作工具展示给用户。在该平台中，表示层的主要作用是显示三维模型，提供属性的可视化编辑等工具，管理数据层的图元。

业务逻辑层中需要实现表示层中各个工具的功能，为表示层展现的场景提供模型数据，调用和处理数据层的图元信息。

数据层使用数据库软件来保存三维图元的属性信息，图元对象是在程序运行的时候创建出来的，保存图元对象就是将图元对象的所有属性保存数据库表中，通过相应模块能够将这些信息快速还原为图元对象，通过数据库操作，能够修改表中图元的属性信息，提高几何相似图元的利用率。

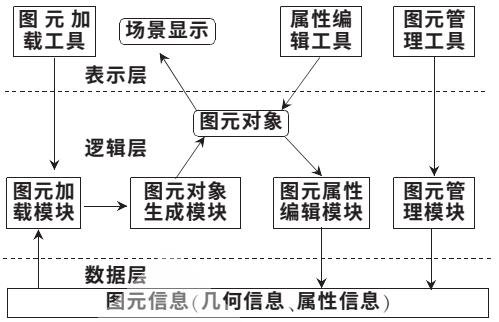

图 1 平台内部结构

2.2 模块分析

逻辑层是平台的核心层，参照图 1 平台内部结构，逻辑层按照功能分为 4 个模块[3]。

（1）图元加载模块

图元加载工具调用加载模块的相应功能加载图元，图元的加载有两种方式：一种是从外部导入三维图形软件生成的中间文件，并将其转换为平台可用的图元；另一种方式是从数据库中获取保存的图元。加载模块将图元的几何信息传递给图元对象模块。

（2）图元对象生成模块

该模块生成一个图元对象，此对象的几何属性来自图元加载模块，同时将几何信息显示于表示层场景中，此时的对象只具有几何属性，只能在场景中显示，不能承载信息，故还需要使用图元编辑模块完成属性编辑。

（3）图元属性编辑模块

表示层的属性编辑工具调用属性编辑模块的相应功能对前一模块生成的图元对象进行属性的创建和编辑，完成属性编辑的图元对象存入数据库中。

（4）图元管理模块

此模块的作用主要有两个：对数据库中的图元进行管理，比如图元的删除、重命名、以及部分属性的直接修改；图元的输出，将数据库中的图元对象输出，为脱离本平台的仿真应用的开发提供可用图元。

3 平台的关键模块实现

3.1 图元加载模块

XAML 是 WPF 技术中专门用于设计 UI 的语言，用 ViewPort3D 标签定义一个三维场景，其中几何信息定义在标记 <GeometryModel3D.Geometry> 的子标记 <MeshGeometry3D> 中，其中主要由 Positions、TriangleIndices 这两个属性决定模型的形状，Positions 属性的值为一系列点的三维坐标，每 3 个点构成 1 个三角形面作为构成模型表面的最小单元，Positions 属性中按照 3 个坐标一组，分别对应第 i 个点（i=0，1，2，3，…），TriangleIndices 属性保存 Positions 中点的索引 i，每 3 个索引为一组，构成 1 个三角形面，简单的三维模型表面分割成三角形时，Positions 的点坐标及 TriangleIndices 中索引的数量不多，但遇到稍微复杂的图元，此时 Positions 中的点以及 TriangleIndices 中索引的数量将大大增加。这将大大增加

XAML 代码的篇幅。

obj 文件作为大多数三维软件支持的输出文件，包含了模型的全部几何信息。obj 文件定义三维模型的方式与 positions 属性不同，不能直接使用，在 WPF 中使用展现三维模型通常需要使用微软提供的 Blend 软件的图形导入功能，使用 Blend 软件在 WPF 项目中导入 obj 文件就是按照上文描述的这种规则将表面划分成很多的三角形，并将三角形的顶点坐标按照指定的规则写入 Positions 属性中，造成 XAML 代码的大量增加[4]。因此在 Blend 中加载 obj 文件的方式并不理想。

由于 Positions 属性中点的坐标冗长，通过改变坐标达到改变模型形状或位置的做法并不可取，通过三维软件重新编辑 obj 文件相对方便，因此完全可以将 Positions 及 TriangleIndices 的属性赋值放在逻辑层完成。在平台界面上通过图元加载工具通过文件对话框选择相应的 obj 文件，此时加载模块调用转换功能将 obj 文件的几何信息提取，生成一系列点的坐标，在逻辑层将这些点的数据添加到 Positions 属性中，这样在 XAML 代码中就避免了加入 ModelVisual3D 标签，这种方式既保留了 Blend 软件加载 obj 文件的方便性又减少了 XAML 代码的篇幅，使其便于维护。

3.2 图元对象生成模块

obj 文件中的几何信息被转换为符合 Positions 属性值规则的点的集合后赋给 GeometryModel3D 的 Geometry 属性，并将此 GeometryModel3D 对象赋给 ModelVisual3D 对象的 Content 属性，该 ModelVisual3D 对象将被添加到一个继承自 ModelVisual3D 的 ModelVisual3DWithName 类的 Children 属性中。Viewport3D 对象的 Children 属性添加 ModelVisual3D 类型对象。此时完成了一个图元对象的创建，此对象只包含几何信息等一些内置属性，再设置好相机、光线等属性即可显示三维模型。

3.3 图元属性编辑模块

(1)内置属性的编辑

内置属性主要包含位置信息、颜色信息。位置信息只反映出了模型的空间坐标及姿态，ModelVisual3D 对象中通过 Transform 属性来改变模型的位置属性、旋转属性、放缩属性；GeometryModel3D.Meterial 属性表示模型的材质属性。

Transform 属性值是 Transform 类型，TranslateTransform3D 继承自 Transform 类，TranslateTransform3D 类有 3 个表示位置信息的属性 OffsetX、OffsetY、OffsetZ。每个都有相应的依赖属性：OffsetXProperty、OffsetYProperty、OffsetZProperty。依赖属性就是一种可以自己没有值，并能够通过 Binding 从数据源获得值的属性[5]。因此可以将模型的依赖属性 OffsetXProperty 等属性绑定到界面上 Slider（滚动条）控件的 Value 属性，这样就可以控制模型的位置。自定义的图元对象 ModelVisual3DWithName 有一个 location 属性，表示模型固定点的空间坐标，该点为模型边界立方体的固定顶点[5]。通过该点的坐标，可以确定模型的空间位置，进而确定模型间的相对位置。其余内置属性的编辑方式可以参照上述过程。

(2)自定义属性的编辑

ModelVisual3D 对象拥有最基本的三维模型的属性，比如上文提到的颜色、位置等属性。但在实际应用中，针对不同的模型还需要用到一些自定义属性。比如说货物的重量属性、生产日期属性、客户属性等。

自定义的属性是某个图元所特有的属性，不必要为所有图元对象都加上这样的属性否则会在应用程序中产生一些不必要的内存空间浪费，于是在本模块中只将自定义的属性保存在数据库中[6]，并与特定的图元对象相关联。将自定义的属性添加到图元对象这一步骤在下一模块中完成。

3.4 图元管理模块

图元管理模块实现数据库中已有图元数据的管理。可以删除图元，修改图元的部分属性信息。对于需要输出的图元，为了实现其自定义属性，平台通过动态代码为每个图元定制一个类型。该类型继承自 ModelVisual3D 类，根据数据库中的图元信息添加自定义的 CLR 属性或依赖属性，最终将动态编译成.dll 文件保存。

在应用中，只需要引入所需图元对象的.dll 文件及名称空间并实例化该类就可以得到图元对象。可以通过该对象的 CLR 属性获得自定义的属性信息，将依赖属性绑定到相应的数据源，或者将此类型作为父类型，扩展方法、事件等操作。

4 已完成模块的测试

以一个钢卷仓库为例，使用该平台布置仓库场景。首先用 Pro/e 软件做出三维模型：立柱、横梁、钢索、钢卷、抓手。并导出各个模型的 obj 文件。通过平台图元加载模块加载这些 obj 文件，通过图元生成模块生成图元对象，显示场景如图 2 所示。

图 2 场景显示

使用平台的属性编辑模块调整各模型的相对位置及各个模型的材质，并将图元信息保存，再使用图元管理模块导出 dll 文件。在新的 WPF 项目中引用 dll 及相应名称空间，实例化各个图元及场景，效果如图 3 所示：布置好场景后后台可以控制场景中的每一个模型，获得

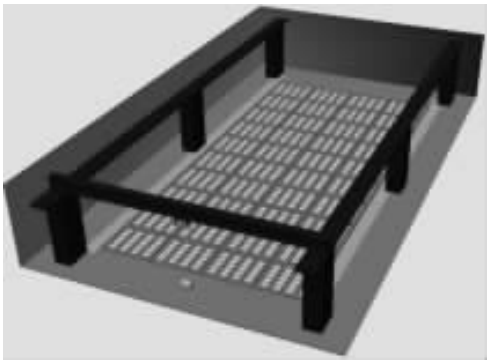

图3 完成布置后的场景

图元属性，根据逻辑需求实现模型的仿真动作。

本文提出了一个仓储仿真及三维图元管理平台的模型，并在WPF下实现了平台关键模块，通过对平台模块的测试可以看出这些模块可以简化仓储仿真的开发步骤，提高相似图元的利用率，节省开发时间。后期任务：增加仿真模块，即从数据库中获取指定场景的所有图元，自动完成场景布置，通过设定参数仿真场景，或者对于简单的项目直接将图元相应的属性与实时数据相关联，以完成场景的实时仿真。

参考文献

[1] 蔡靖，申婷婷，王海丹.基于Flexsim的某自动化仓储系统的仿真结果和分析[J].制造业自动化，2012，34(7)：107-122.

[2] 张建奇，李墨翰，郑伟.基于WPF的工厂物流管理系统界面设计[J].自动化技术与应用，2011，30(12)：17-20.

[3] 李成刚，冯静，凌玲.基于WPF的交互绘图系统的开发[J].微型机与应用，2011，30(6)：50-52.

[4] 张洪定，孟冬梅.基于Expression Blend4中文版WPF和Silverlight项目设计基础[M].北京：清华大学出版社，2011.

[5] 刘铁锰.深入浅出WPF[M].北京：中国水利水电出版社，2010.

[6] 石怡.WPF使用XAML实现对SQL Server数据库绑定的方法[J].电脑开发与应用，2011，24(10)：70-74.

(收稿日期：2013-06-05)

作者简介：

张世辰，男，1989年生，硕士研究生，主要研究方向：基于WPF平台的仓储仿真。