Undo/Redo for Qt Tree Model

eryar@163.com

Abstract. Qt contains a set of item view classes that use a model/view architecture to manage the relationship between data and the way it is presented to the user. The separation of functionality introduced by this architecture gives developers greater flexibility to customize the presentation of items, and provides a standard model interface to allow a wide range of data sources to be used with existing item views. Model 3D aided design software such as AVEVA Plant/PDMS, Marine use the architecture to manage the design data source. The article demonstrate the Undo/Redo on the Qt Tree model.

Key Words. Model/View, MVC pattern, Undo/Redo, Tree Model

1. Introduction

现代稍微大型一点的软件，要处理的数据量通常会比较大。这时就需要有一个唯一的数据源，且会对这个数据源中的数据进行增、删、改的操作。如果没有统一的数据源，数据会随意地被创建和删除，且创建和删除的用户界面也不统一，不利于软件管理。基于唯一的数据源，并在这个基础上提供统一的增删改接口，不仅有利于软件数据管理，还有利于事务的处理，即 Undo/Redo 功能。若引入脚本语言，如 Tcl 或 Python，甚至可实现脚本命令对数据的操作，为程序增加二次开发功能。

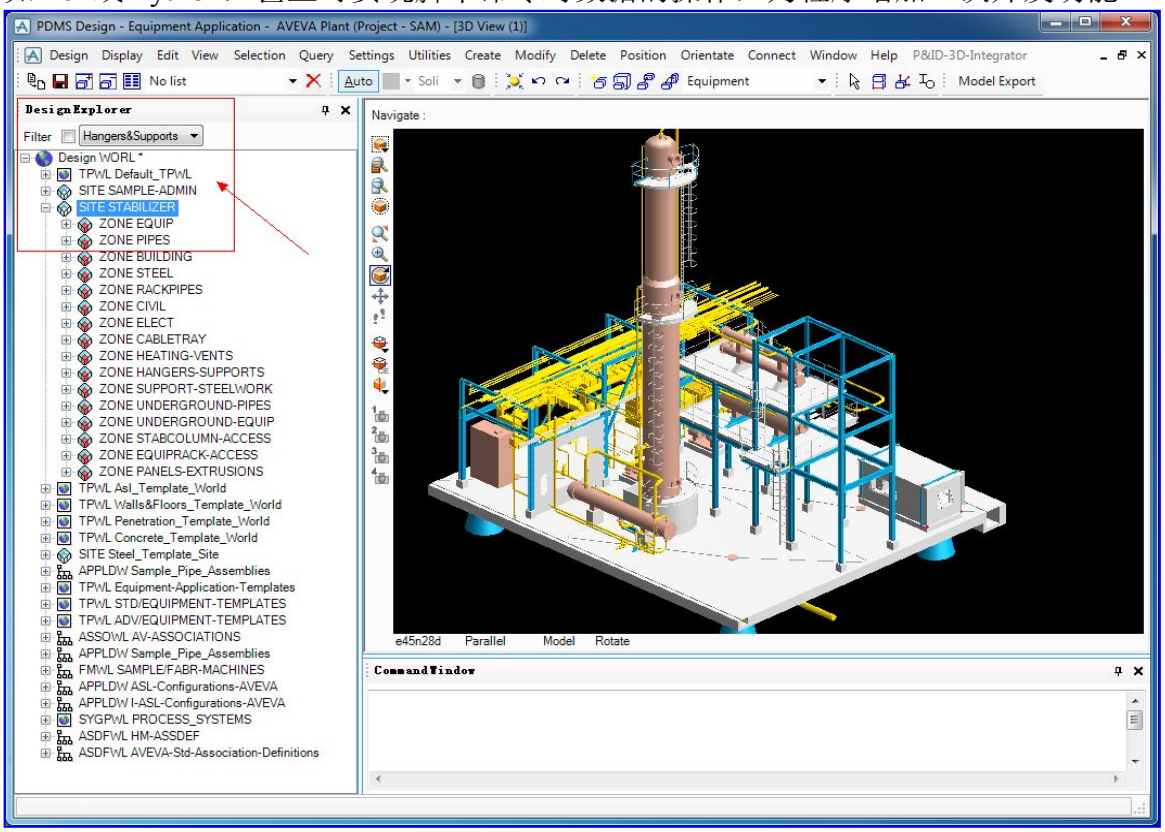

Figure 1. AVEVA Plant/PDMS Software

如上图1所示为英国剑桥大学CADCENTRE出品的AVEVA Plant/PDMS软件。PDMS使用了统一的数据源，左边的导航树及右边的三维模型都是这个数据源的一种展示方式，可以在导航树上创建及删除数据；也可以在三维视图中进行交互，方便地创建及修改模型。这种方式与GoF描述的Observer观察者模式相似，即：

定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新。

对应上面的软件，即对于唯一的数据源这个对象，导航树及三维视图都依赖于他。当数据源这个对象中有数据的增删改时，导航树及三维视图都会得到通知并自动更新了。

图1右下角的CommandWindow中可以输入命令，即PML，也可以对数据源进行修改。引入Observer模式，即可实现这些功能。

Qt中包含了一系列的Model/View类，这些类都是基于MVC架构的，这种将数据与视图分开的处理，使同一个数据源可以不同的视图中进行展示，且不需要修改数据源的结构。Qt中基于Command命令模式实现了一个Undo/Redo的框架，将Undo/Redo框架应用于Model/View的类，即可得到与AVEVA Plant/PDMS中类似的功能。

本文主要对Qt中的model/view及undo/redo框架进行介绍，及如何将Undo/Redo框架应用到model/view中去，实现对以层次方式组织数据的树模型进行undo/redo，进而实现一个三维CAD工厂设计的原型。

2. Tree Model

软件中使用层次方式来组织数据，可护展性好。如 OpenCASCADE 中的 OCAF 框架的数据也是树形方式组织：

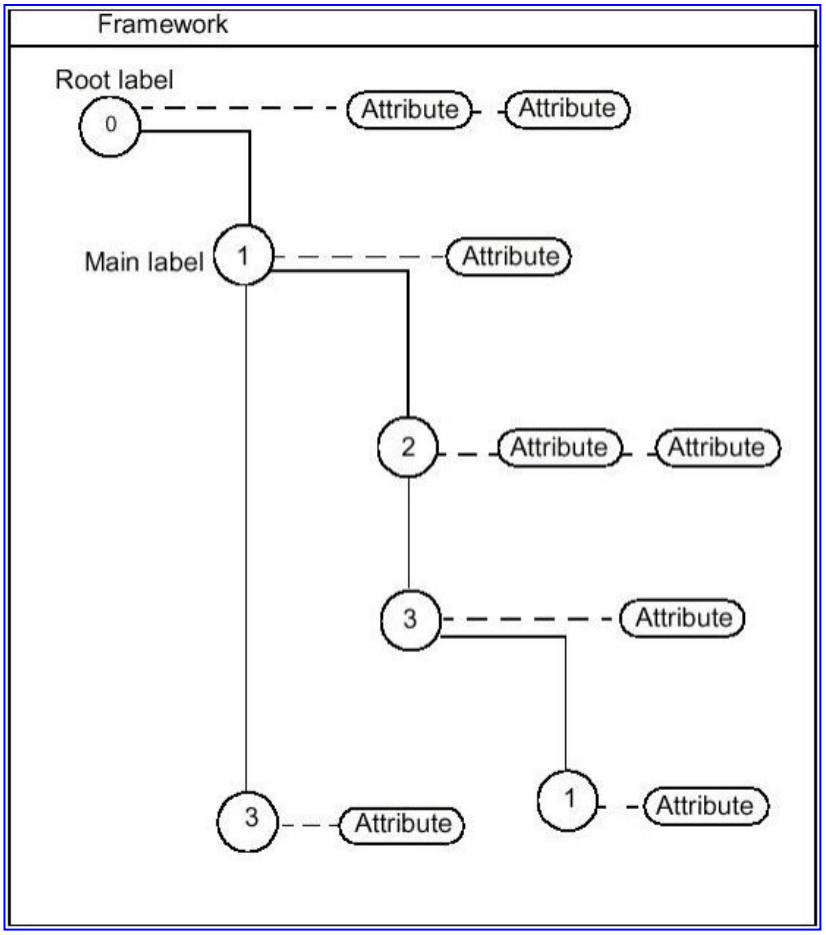

Figure 2.1 OpenCASCADE OCAF data tree

每个 Label 中可以存放属性，还包含子 Label，这样就提供了一个通用的程序框架，灵活使用，可以用一种统一的方式来存放数据。有一个缺点就是每个 Label 中的属性类型只能包含一种，这种方式相对于 AVEVA 的自定义属性 UDA 来说，就显得很不方便了，需要提前对数据的存储进行规划，扩展起来稍有不便。

AVEVA 中对每个数据 Element 都提供了自定义属性（User Definable Attribute），每种类型的属性（数值型、字符串型等）都可以包含任意数量，没有限制。所以可扩展性更好。

Qt 中基于 MVC 模式的 model/view 架构提供了一些预定义的模型及视图，如列表模型及对应的视图，树模型及树视图等。在 model/view 架构中，为 model 提供了统一的访问数据的方式，即 Model Index。对于列表 model，通过 QModelIndex::row() 即可访问。对于表格 model，需要 QModelIndex::row()和 QModelIndex::column()。对于树 model，除了上述两个函数，还需要提供 QModelIndex::parent()来对父节点进行访问。如下图 2.2 所示：

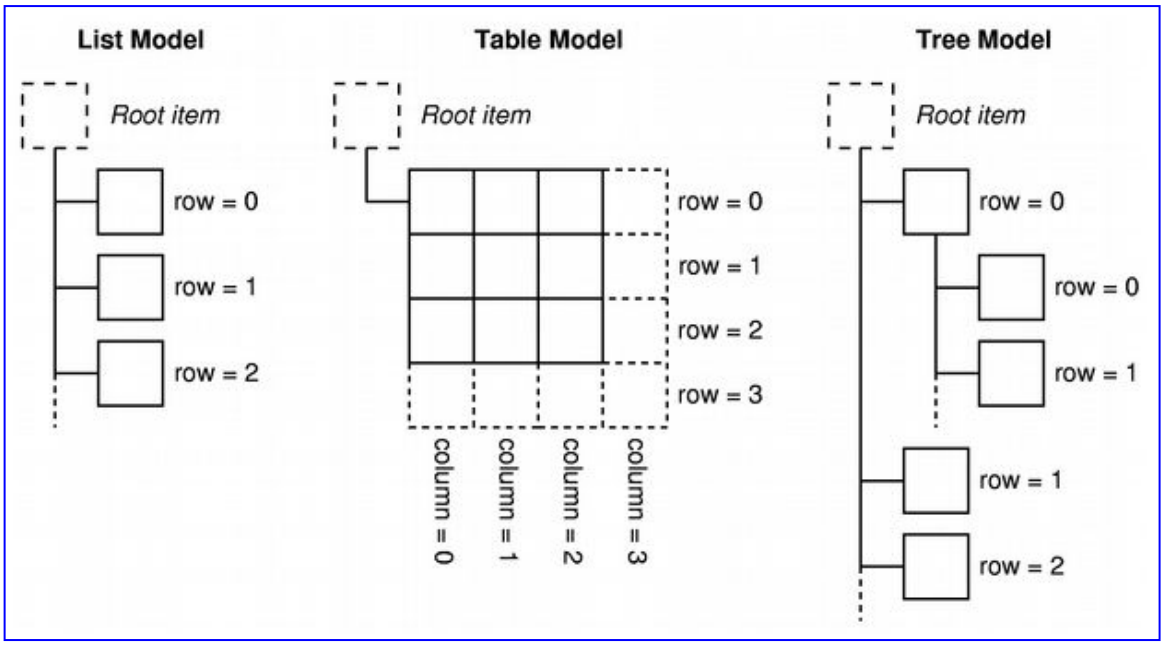

Figure 2.2 Schematic view of Qt's Models

若需要对列表及表格 model 应用 Undo 框架，还是很方便的，只要记住数据操作的对应的 row 或 row 及 column 即可。若要对树形 model 进行 undo，原理也是一样的，即在 redo 中生成数据在 undo 中恢复数据。

3. Key Points

将 undo 框架与 model/view 结合有两种方式：

- 通过在 undo/redo 命令中调用 QAbstractItemModel 的 API 来改变底层数据；
- 自定义 model 使其创建命令并推送到 undo 栈 Stack 中去；

第一种方式看起来要容易些，且可与任意 model 结合。The first approach seems simpler, as it works with any model and gives the undo framework total control --- you can create as many different types of commands as you want and each of them can call a number of methods from the model API in their undo() and redo() routines.

当 model 中有 delete 操作时，会将 index 内部的数据删除。对于这种情况，需要注意。Qt 给出了两个解决办法：

There are two things to remember. The first is that, if you delete items, rows or columns, you should save the data of all the items getting deleted(including children) somewhere so that you can later revert the deletion. The second is that it would be nice to be able to set meaningful descriptions of commands pushed on the stack. To do that you can provide methods in the model, that are called each time a command is pushed onto the stack, which should each return a description based on the parameters of the command being executed.

基于上述思想，实现了 Tree Model 与 Undo Framework 结合的一个小例子：

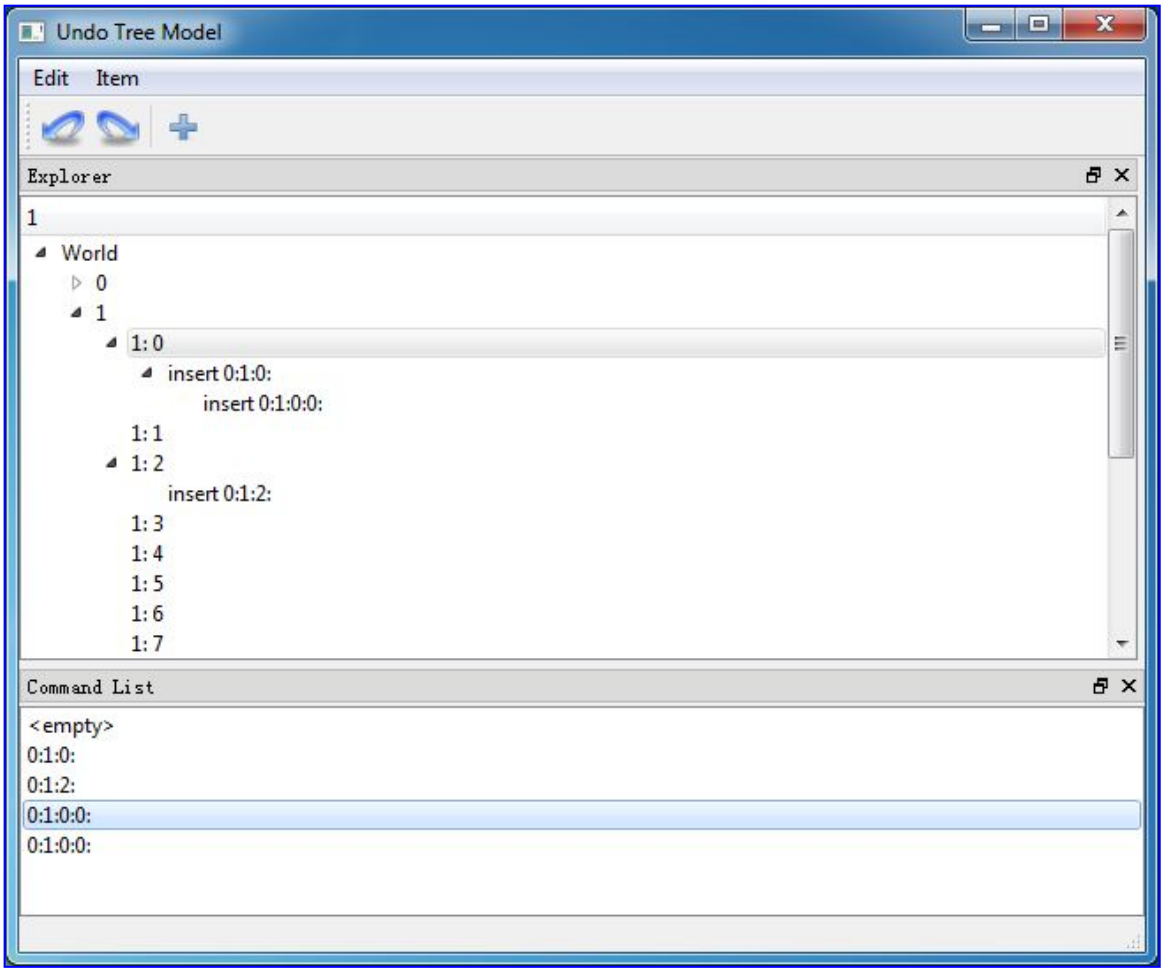

Figure 3. Undo/Redo and Tree Model

4. Conclusion

Qt 中基于 Command 命令模式的 Undo 框架，需要对数据操作及恢复仔细考量。尤其对于有删除数据的操作，若以 QModelIndex 作为参考依据，则会出错。对于此 Qt 提供了两个解决思路。

基于 Qt 的 model/view 及 undo 框架，也可以方便实现 OpenCASCADE 中的 OCAF 框架许多类似的功能，且 Qt 的代码可读性更高。若要快速开发程序，可以考虑使用 Qt 的 model/view 框架。

感谢晓天的建议。

5. References

1. Using Undo/Redo with Item Views. http://doc.qt.digia.com/qq/qq25-undo.html
2. OpenCASCADE. OCAF User's Guide. 2014
3. GoF. Design Patterns-Element of Reusable Object-Oriented Software. 1995