# Undo,Redo功能设计文档说明书

## 功能需求

Canvas中元素的各种操作支持有限步骤的撤销、重做功能。具体操作暂定包括：单元素重命名，单元素删除，单元素添加，单元素移动，单条线删除，单条画线。**多元素的操作后续再考虑，目前看元素和关系的添加和删除必须考虑，不然操作不连贯会出问题。例如，如果多元素删除不考虑的话，当一个重命名操作入栈后，删除对应元素，则Undo操作就失去目标了。**

## 设计方案

### 2.1 数据结构

1，Undo,Redo两个固定大小的栈，目前默认大小各为100，既支持100步撤销、重做，已经能满足大部分需求，无限步骤的撤销、重做业务上没啥意义，还浪费内存。

2，ICommand接口，内含Undo, Redo两个动作，用来封装所有的单元素执行和回撤动作，每次执行后，要把对应文档置为Dirty。

3，UndoRedoManager,全局单例，用来管理Undo、Redo两个固定栈和协同界面Undo、Redo按钮事件等操作。其中每个文档设置Undo,Redo固定栈各一个，以ModelDocument为Key,{Undo，Redo}为Value,用Dict存储。当文档切换时，根据CurrentDocument切换到对应的固定栈,并重置界面按钮状态。

### 2.2 算法设计

1， 所有单元素完成操作的程序点，将相关上下文数据封装到ICommand中，然后Push进入Undo栈（由于是固定栈，所以当元素个数大于固定值时(默认100)，自动替换栈底元素），同时清空Redo栈。

2， 当点击Undo按钮时，Pop出Undo栈栈顶ICommand元素，然后执行Undo接口，回归对应操作，最后将ICommand元素Push进入Redo栈。

3， 当点击Redo按钮时，Pop出Redo栈栈顶ICommand元素然后执行Redo接口，执行对应操作，最后将ICommand元素Push进入Undo栈。

该操作与第二步对称。

1. 当Undo栈由空变不空时，Undo按钮Enable;当Undo栈由不空变空时，Undo按钮Disable;当Redo栈由空变不空时，Redo按钮Enable;当Redo栈由不空变空时，Redo按钮Disable;以上四种操作，由UndoRedoEvent的四种事件通知主界面
2. 在单元素的6种操作完成点，根据不同的操作类型封装ICommand接口所需的各种信息：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 操作 | 元素类型 | 进度 | 编码人 | 备注 |
| ElementRenameCommand | MoveDsElement | 完成 | 卢琪 |  |
| MoveOpElement | 完成 | 卢琪 |
| MoveRsElement | 完成 | 卢琪 |
| ElementAddCommand | MoveDsElement | 施工中 | 卢琪 | 对应删除元素时的关系动作还未处理 |
| MoveOpElement | 施工中 | 卢琪 |
| ElementDeleteCommand | MoveDsElement | 施工中 | 卢琪 | 对应删除元素时的关系动作还未处理 |
| MoveOpElement | 施工中 | 卢琪 |
| RelationAddCommand |  |  |  |  |
| RelationDeleteCommand |  |  |  |  |
| ElementMoveCommand | MoveDsElement | 完成 | 卢琪 | 编辑状态下的单元素移动操作。 |
| MoveOpElement | 完成 | 卢琪 |
| MoveRsElement | 完成 | 卢琪 |

6, 具体的Command类的构造函数，Undo,Redo执行逻辑依赖与具体的业务，同时要谨慎选择Command类的插入点。

### 2.3 代码实现

代码位于命名空间Citta\_T1.Core.UndoRedo下，

FixedCommandStack.cs, UndoRedo固定栈,双链表实现

ICommand.cs，命令类接口口, 约定Undo撤销，Redo重做

UndoRedoEvent.cs, 通知主界面撤销，重做按钮状态的事件

UndoRedoManager.cs, 统筹管理Undo,Redo行为

Command目录，实现各种具体命令类的业务逻辑

其他各种操作的代码插入点，具体操作具体分析。