DrawPlate 的一种高效实现方式

rayscc, zhl.rays@outlook.com

1 引言

DrawPlate 属于 DevTrack 的数据标记部分，对图像算法的逻辑分析与设计具有很好的辅助的作用。其需要实现的功能如下:

［1］实现画笔功能，可以对原始图像上的任一坐标格子进行覆盖绘制；

［2］实现橡皮擦功能，可以擦除画笔留下的痕迹；

［3］实现撤销和重做的功能。

注：坐标格子是绘制的最小单位。根据逻辑坐标，将多个位于一个矩形框内的像素点集合作为一个坐标格子。坐标格子的定义如下：Struct Cell{Point loc; Color clr;}。

2 具体操作

为了支持图像的放大与缩小，DrawPlate 中的绘制对象均以坐标点的形式进行存储（而不是以像素点集合的形式或是 Bitmap 的形式）。

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

图1 3*7的原始视图

0	0	1	1	1	0	0
0	0	2	2	0	0	0
0	0	0	2	0	0	0

图2 3*7的按步骤1绘制视图

0	0	1	1	3	0	0
0	0	2	3	3	0	0
0	0	0	2	0	0	0

图3 3*7的按步骤2绘制视图

为了平衡空间效率以及时间效率，我们在内存中只保留一张绘制视图、一张覆盖格子恢复子图以及一个回退栈，并将步骤号作为划分准则。绘制视图以二维数组的方式进行存储。如图 1 所示，为一张 3×7 的绘制视图。

在每一次绘制时，我们将判断每个需要绘制的坐标格子与之前同一坐标下的格子颜色是否相同，并将这些差异保存到覆盖格子恢复子图中，该子图将参与撤销与重做（值得一提的是，绘制视图中值为 0 的坐标格子将不参与差异对比）。例如，上述图 1 至图 3 中，坐标(1,5)、(2,4)和(2,6)连同它们各自的编号将被保存到覆盖格子恢复子图中的步骤 3 号位（步骤号与编号一一对应）。图 4 为覆盖恢复格子的生成过程。

在实现橡皮擦功能时，我们直接将绘制视图目标坐标所示位置设为 0 即可。

对于撤销与重做功能，只需创建一个回退栈，保存每次回退前将要被清除的坐标格子。

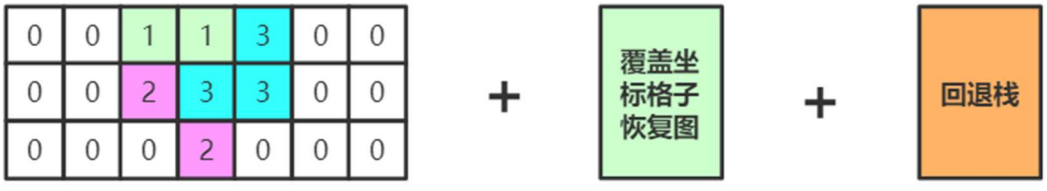

图4 需要的数据存储类型

撤销功能的实现

[1] 检查恢复格子子图(检查是否存在当前步骤号)，如果有，则先将恢复格子子图中恢复对应的格子颜色到当前绘制视图；若没有，则跳过。

[2] 在绘制视图中**搜索**与当前步骤号编号相同的格子，先将这些格子压入到回退栈，再根据这些格子对绘制视图对应位，置 0 清除。

[3] 将当前步骤号指向上一个（-1）。

值得一提的是，覆盖格子恢复子图将在最后清除全图的时候被清理，在撤销和重做过程中，改变的仅仅是步骤号指向。

重做功能的实现

[1] 从回退栈中，弹出一个栈顶元素，将该元素中包含的坐标格子绘制到绘制视图中；

[2] 将当前步骤号指向下一个（+1）；

[3] 恢复格子子图中恢复对应的格子颜色到当前绘制视图。

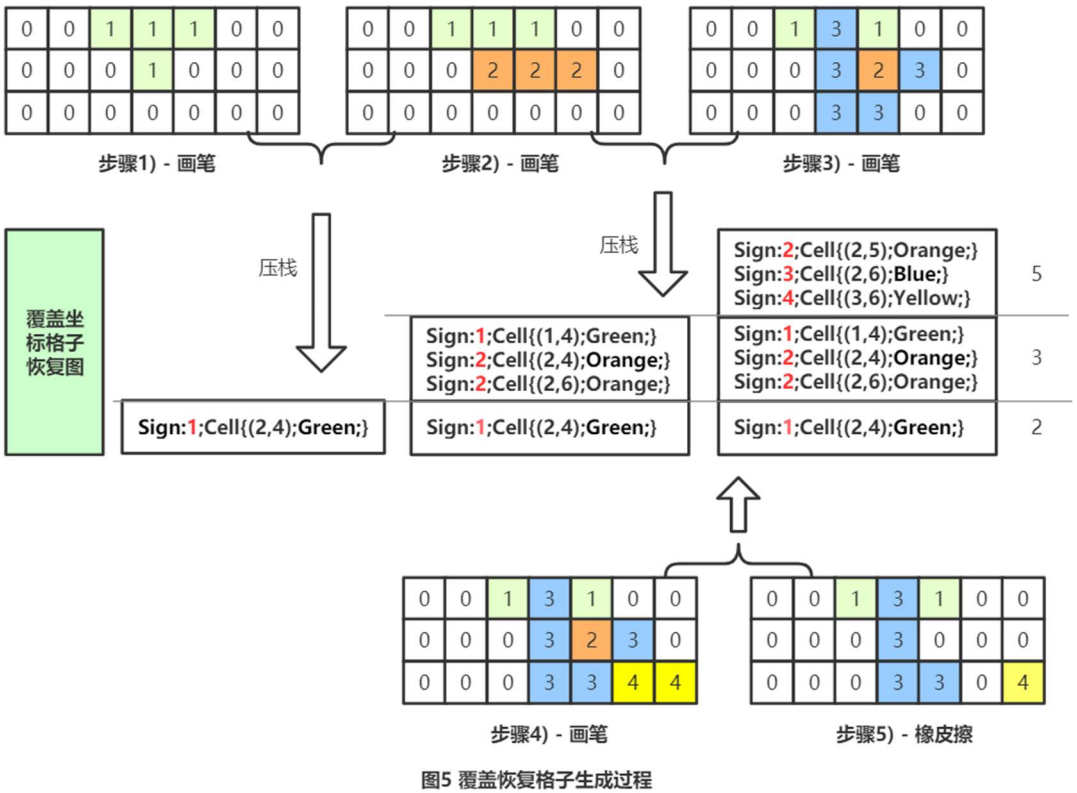

图5 覆盖恢复格子生成过程

如何在绘制视图中搜索指定编号的格子？

如何在一张具有上千个甚至上万个元素的集合中，快速找到具有相同数值的元素集合，是一件十分有趣的事情。考虑到绘图时使用者的习惯，绘制的数据具有很好的局部性，我们提出了一种结合了 BFS+DFS 的方法。相对于去穷举遍历绘制视图的全部元素，该方法更加

的高效。大致的过程如下：

[1] 从绘制视图的某一位置出发，采用 BFS 逐层向外搜索以求取得到第一个满足条件的格子坐标，记作(X_0, Y_0)；

[2] 从(X_0, Y_0)出发，采用 DFS（8 个方向搜索）求取获得连通的所有相同编号的格子。