FlowFree 设计文档

一、 游戏简介

FlowFree 是一款基于 android、iPhone/iPad os 的手游。这是一款操作简单，惹人喜爱的益智类游戏。这个游戏的基本规则是用一条折线连接两个颜色相同的端点，游戏中有许多对端点，需要将所有的端点连线匹配，并且任意两条折线不能相交，同时还要保证匹配完所有的端点后折线将整个游戏盘填满，不能有多余的空格。对于这个游戏，每一关都只存在唯一方案。

二、 实验简介

用 QT 在电脑上实现 FlowFree 中最简单的模式。

三、 基本思路和算法

首先将整个游戏看作一个对象(Game)，它至少拥有游戏盘(board)、难度(board_size)、关卡(current_level)三个成员变量分别记录整个游戏、选择难度和关卡。因为游戏随时在变化所以需要实时重绘，因此我们需要保存整个游戏的当前情况，即每条折线的路径。可以为每种颜色用一个栈(linePath[])来存储它的路径。

FlowFree 这款游戏做的比较精美用户体验极佳，在连线的时候会以手指为圆心绘出一个颜色透明的大圆圈提醒用户，并且连线也是中间一条实心细线周围一条实心透明粗线，非常有艺术感。不仅可以以初始的两个颜色端点为起点还可以以当前已绘部分中的某点为起点继续连线，在当前线条“切断”另外已经完成的线条时，会发出断线的提示声，在完成一条线时，会发出连通的提示声。你可以在连线的过程中恢复到以前走过的步骤(仅限按下手指时到目前的步骤，按下手指之前的步骤无法恢复)，恢复到以前步骤的同时游戏局面也随之恢复。

为了实现上面这些极佳的用户体验，我们还需要实时备份之前的游戏局面，这里也可以用另外一个栈(linePathBackup[])来存储路径。同时还需要一个表(win[])来记录每种颜色是否连通，当每对端点都连通的时候需要判断整个棋盘是否填满。

在我的代码实现中，用 start()函数开始游戏，在游戏开始时需要调用 preprocess()进行相应的数据初始化。我重写了 *paintEvent*(QPaintEvent*)函数，在这个函数中调用 paintBackground(QPaintEvent*)绘制好游戏盘，调用 paintEllipse(QPaintEvent*)绘制多对不同颜色的端点，调用 paintPolyline()实时绘制当前已有的部分连线，如果函数对象指针不为空则表明当前用户已按下左键(手指触摸屏幕)，需要调用函数对象指向的 paintEllipseLighter(QRect)在指定的方格内绘制透明大圆圈提醒用户。

我重写了 *mousePressEvent*(QMouseEvent*) 响应用户按下鼠标左键，调用 getCurrent(QMouseEvent*)得到当前的坐标和颜色信息，调用 backup()初始化备份数据，获取要绘制的颜色 last_color(与 current_color 同值)。在重写的 *mouseMoveEvent*(QMouseEvent*)中调用 getCurrent(QMouseEvent*)得到当前的坐标和颜色信息，判断坐标是否合法，如果不在游戏盘内视为不合法。通过调用 legal(const QPoint&)判断是否合法，若合法是前进还是后退，该函数为 0、-1、1 的三出口函数分别对应后退、不合法、前进三种情况。对于不合法请求不做任何处理；对前进请求通过对比 current_color 和

last_color 判断是否“切断”其他颜色已有连线，若切断需要调用 clean(int, int)进行处理和备份，并且调用 push_back(const QPoint&)更新当前路径；对后退的请求，需要调用 clean(int, QPoint, int)进行处理和调用 backupRestore()进行备份恢复。

用户松开鼠标时，重写的 *mouseReleaseEvent*(QMouseEvent*)被调用，进行相应的发声处理、判断用户是否过关。如果用户过关调用 QMessageBox::question 询问用户下一步操作。倘若用户关闭游戏调用重写的 *closeEvent*(QCloseEvent*)进行相应的数据处理。

其中一些比较重要的函数在代码中有简练的注释解释函数意义。整个游戏实现中，路径的存储最为关键，影响整个程序的鲁棒性以及是否容易调试。我的代码中采用 QVector<QPoint>实现栈来储存路径，因为路径是按照先绘先进后绘后进这正好符合栈的特性。而备份路径也同理，因为路径消除是按照后绘先消，而路径恢复是先消后恢复，这也符合栈的特性。用栈来存储信息方便了路径的存储和消除操作。再用二维数组 board 辅助记录游戏盘颜色的信息，这样方便判断移动是否合法、判断该点是否应当恢复。

四、 总结

完成这个大作业不仅学习了 QT 中不少新东西，还结合之前学习的 c++面向对象知识，加强对面向对象的理解和一些软件机制的理解。过程中尝试使用函数对象，需要用函数指针指向成员函数，这个和函数指针指向普通的函数大不一样，通过查阅资料学习了 c++的一些以前不知道的知识。在清除 warning【-Wreorder】时，复习了相关的 c++语法知识，认识到 c++的严密性。总之完成这个大作业不仅是对 QT 的学习还有对 c++和面向对象的复习，对它们有了新的认识，更增强了自学能力。