C++大作业论文

WEB GUI

基于HTML和javascript的通用图形界面

吴轩泰 于心一 刘嘉伦 黄逸凡

2021年12月

目录

1 背景

随着技术进步以及市场发展，日常生活中人们开始使用各种终端以及各种系统，这对程序的兼容性提出了不小的开发难度，再加上人们审美需求增加，对于应用程序的图形界面产生了越来越强的外观方面的需求。基于此，主流的方法有两种，一种是构建跨系统的虚拟机，在其上运行通用的程序。如 JVM；另一种是以动态网页为代表的，将浏览器作为程序的显示界面甚至是运行环境。

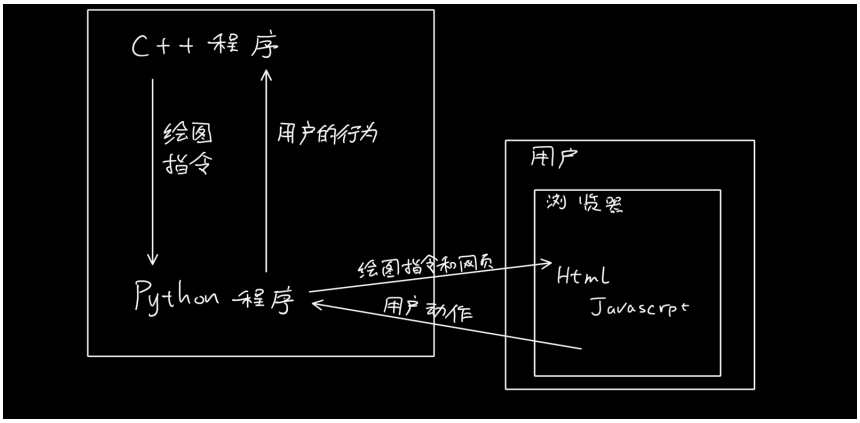
另一方面本小组的部分成员受其软件环境限制，无法安装任何图形界面库，也无法安装大部分开发工具。

因此我们设计了一套图形界面系统，借此简化了对图形界面的调用，使其结构不会影响到程序本身架构。同时也得以实现便捷部署以及异地查看。

2 主体架构

我们将一套系统分为三部分：主程序、python 中转服务器，以及浏览器的模版网页。其中主程序是图形界面的调用者，负责程序的全部逻辑；python 中转服务器负责将主程序的绘图指令转发给模版网页。模版网页执行绘图指令。

由于技术限制，我们无法从服务器直接向用户端发送请求，只能进行单向的请求，加上网络延时及连接不稳定性，我们把设计目标定位基于事件的程序，即图形界面的变动只是在用户执行操作时产生。例如常规的五子棋、中国象棋等游戏。但后期为了拓展其功能，也增加了定时更新画面的功能，得以实现动画效果。



3 运行流程

在服务器端开始运行主程序和 python 中转服务器后，用户在浏览器打开网址，服务器返回模版网页，其中的 javascript 代码开始运行，通过 POST 请求向服务器发送一条初始事件（UPDATE 事件）。python 中转服务器收到 POST 请求后把事件输出，重定向到本地的 fifo 文件中。主程序从 fifo 文件中读取事件，经过处理后将绘图指令输出（重定向）到本地 fifo 文件中，由中转服务器读取后作为 POST 请求的响应返回给用户端，用户端再响应。

整套系统的运行由若干次这样的循环组成。

4 输入和输出

由于采用了重定向，主程序的输入和输出是简单的文字信息，每次的输入是一条事件，输出是若干绘图指令，以 DONE 为结束，一条指令一行。指令和事件都是由类型和参数组成。

服务器向网页的通讯内容以 JSON 格式存储，是一个由绘图指令组成的列表。网页向服务器发送的请求负载是文本形式的事件内容。

事件	含义	实例
Update W H	屏幕宽度变为W ×H，其中 W 是宽，H 是高	Update 1024 768
Click X Y	在（X，Y）的位置用户进行了单击	Click 500 500
Key key	用户点击了编号为 key 的按键	Key 56
BOOK	你在之前预定了此时进行交流	BOOK
Prompt msg	用户在输入框输入了信息 msg	Prompt Hello {SPACE}World

事件列表

指令	含义	实例
Rect X Y W H color	在（X，Y）位置绘制尺寸为 W×H 的矩形，颜色为 color	Rect 0 0 1024 768 red
Circ X Y R color	在(X,Y)为圆外接正方形的左上角绘制半径为 R 的圆，颜色为 color	Circ 5 5 5 blue
CLRS	将屏幕上所有图形清空	CLRS
Text X Y size color content	在（X，Y）为外接矩形左上角，字高度为 size，颜色为 color 打出 content	Text00 50 #0000ff Hello{TAB}World
Tri X Y L R H deg color	以(X,Y)为外接矩形左上角绘制颜色为 color 的三角形	Tri 30 30 10 10 20 0 blue
Timer t	在 t 毫秒后发送 BOOK 事件，以期在 t 毫秒后更新屏幕内容	Timer 5
Alert msg	发送内容为 msg 的弹窗提醒	Alert Bow{SPACE}Tie{SPACE}Are{SPACE}COOL
Title title	将网页的名称设为 title	Title Best{SPACE}Game{SPACE}Ever
Prompt msg	弹出输入框，输入的提示信息为 msg	Prompt 输入你的择偶标准
DONE	告知本批绘图指令完毕	DONE

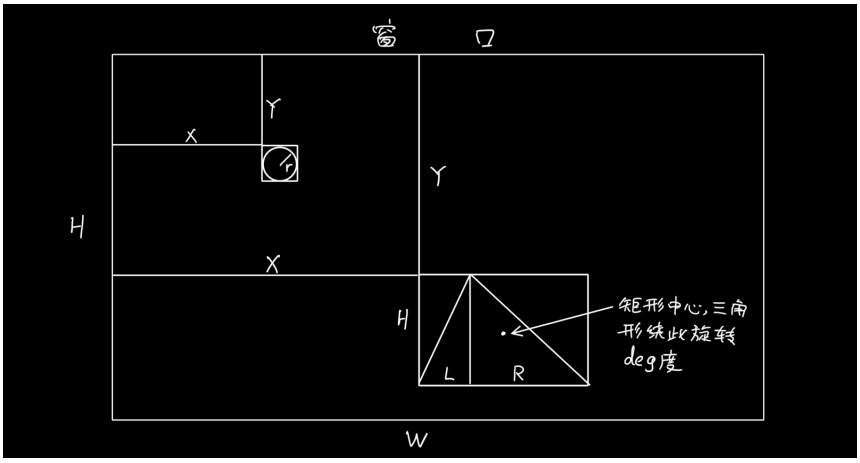
绘图指令的格式与含义

4.1 其他

4.1.1 坐标系统

坐标一般用（X，Y）表示，X 是点距左边框的距离，Y 是点距右边框的距离。

4.1.2 绘图部分的更多细节



4.2 文本信息格式

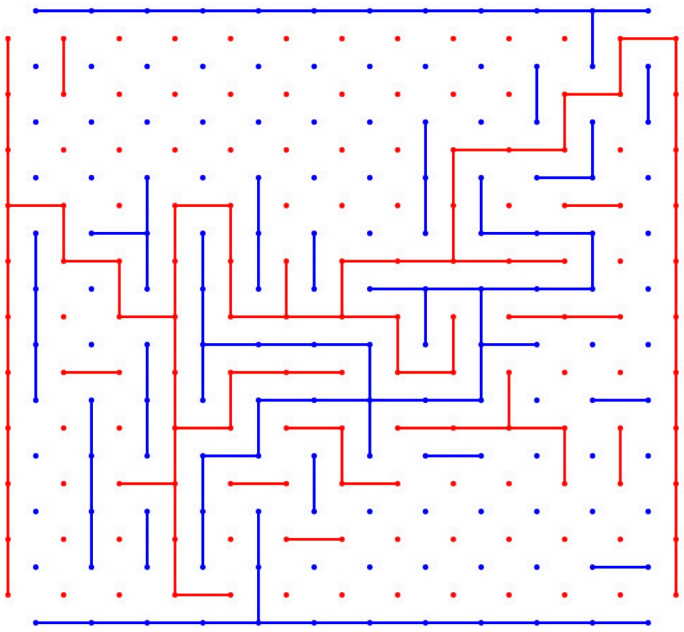
由于绘图指令以空格分割参数，以换行分割指令，因此绘图指令中的文本内容里要用 {SPACE} 代替空格，用 {ENTER} 代替换行，用 {TAB} 代替制表符。

5 实际应用

5.1 Cut_Games

5.1.1 介绍

此游戏由西方著名游戏 HEX 演变而来，棋盘为一个方格点阵，其中横坐标为奇、纵坐标为偶的是蓝点；纵坐标为偶，横坐标为奇的为红点。红方、蓝方轮流连线，不得交叉，红方连红点，蓝方连蓝点。红方从左连至右赢，蓝方从上连到下赢。这是一个典型的零和博弈问题，不存在平局（一方的连通等价于另一方的一个割，并不存在双方都无法联通的情况）。为了游戏平衡，先手红方的方向要略长。



5.1.2 实现

我们用一个二维数组记录点的连通情况以及边的占用情况（正如所见，每个空白格点上要么是红线，要么是蓝线）。编写了一个游戏类将其包装。

```
class cut_games{
        private:
                vector<vector<int> > nodes;//存储用数组
                int n,m;//棋盘尺寸
                int status;//记录游戏状态
                int find_root(int num)//并查集用，查根
                void merge(int num1,int num2)//并查集用，合并
                int check(int x,int y)//将向数组x，y对应的边连通
                vector<int> route()//求解得胜方的取胜最短路径
                int basic_length = 20;//网格宽度
                int offset_x = 0,offset_y = 0;//棋盘绘制偏移量
                void rect(int x,int y, int w, int h,int color)//绘制长方形
                void dot(int x,int y,int r,int color)//绘制圆形
                void draw_board()//绘制棋局
                void draw_connection(int x,int y,int stroke = -1)//绘制线段
                void draw_line()//绘制全部以连通边
                void draw_route()//绘制取胜路径
                void draw_game()//绘制整局游戏
        public:
                cut_games(int n,int m)//初始化
                void update_screen_size(int new_width,int new_height) //处理屏幕尺寸更新
                int click(int x,int y)//处理鼠标点击事件，选取屏幕x，y位置对应的边
}
```

Listing 1: 类的结构

用并查集维护点的连通性 循环方式实现的并查集，高效地维护了点的连通性，减少了回合间的响应时间。

bellman-ford 算法变体求解最短路径 基于性能要求不高，采用 bellman-ford 算法，按边更新最短路长度直至一轮循环不再更新为止。用 anc 数组记录每个点最短路的上一节点，借此求出最短路径。

处理鼠标点击事件 直观上来说，被点击对象应该是所有空白格点中距鼠标坐标最近的。因此每个空白格点的响应区域是一个 45° 放置的正方形。

对于一维的情况一般用 $\lfloor \frac{x}{basic\_length} \rfloor$ 来选取，但这里坐标轴发生了旋转，因此我们用以下公式实现了坐标到点击对象的转换。

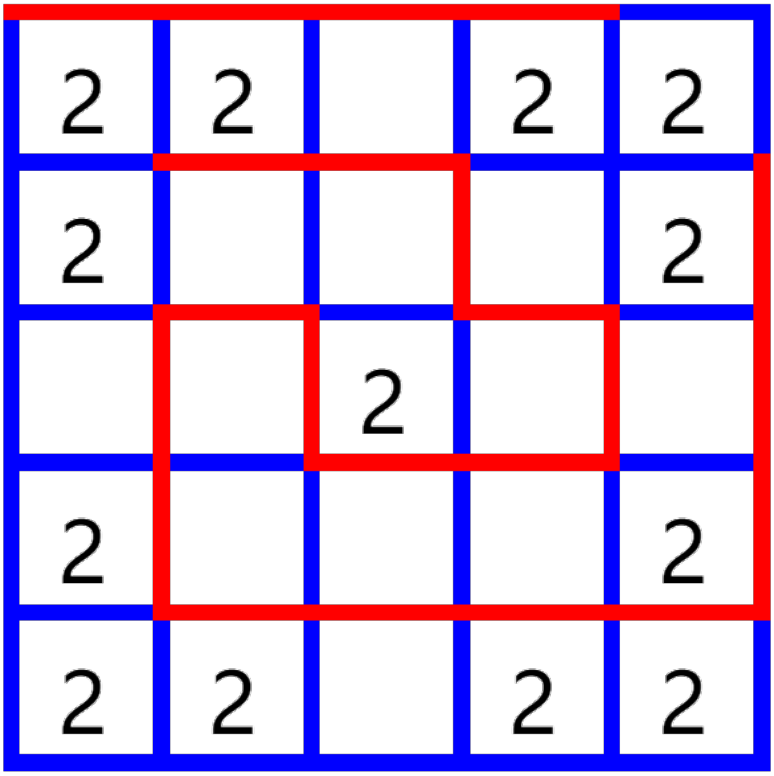
$$a = \lfloor \frac{x + y + basic\_length}{2 \cdot basic\_length} \rfloor$$

$$b = \lfloor \frac{x - y + basic\_length}{2 \cdot basic\_length} \rfloor$$

$$x' = a + b; y' = a - b$$

5.2 Careful String

5.2.1 游戏介绍



此游戏由日本著名益智游戏数回 (Slitherlin) 演变而来，游戏棋盘为一个正方形方格表格构成，玩家需要从左上角出发，画出一条连通左上角与右下角格点的，不间断、不交叉的线。而题目中会有一些方格填有数字 0，1，2，3 之一，作为限制条件。玩家需要保证填有数字 X 的方格的边界有且仅有 X 条边被连线。该游戏不但拥有划迷宫般的乐趣，还可以训练游戏者利用逻辑思维冷静推断，系统地抽其丝、剥其茧，从而使脑力得到全面提升。

5.2.2 实现

处理鼠标点击事件 虽然通用图形界面已经提供了识别用户鼠标点击位置坐标的功能，但由于程序中格点建立的参数是像素坐标的形式，而在实际的游戏运行中，要求玩家每一次都精确地点击到格点所在坐标是不切实际的。

因此程序中建立了基于边框粗细（即所含像素点个数）的逻辑型格点识别函数，通过循环语句穷举比较鼠标点击坐标与所有格点坐标。若鼠标点击坐标在某一格点附近（基于绘制的表格的边框的粗细，即像素点个数），则函数将该格点坐标储存至一个全局变量中，并且函数返回值为真。若所有格点均不满足条件，则函数返回值为假。而基于鼠标点击事件识别，程序中建立了一个参数用于储存格点的点击情况（将被点击的格点的相应参数赋值为 1）。

判断点击两点后的状态 首先程序中建立了两个逻辑型二维数组 u[x][y],v[x][y]，分别储存坐标为（x ，y）的格点右方、下方的边的连接情况（真为已连接）。

程序中建立了判断点击的两点的位置关系的函数，通过比较玩家点击的两个格点的横纵坐标，处理了不同情况下玩家点击两个格点后程序的行为：

1. 若两个格点横纵坐标均不同，则函数判断为连线不合法，程序不输出任何值。
2. 若两个格点仅横/纵坐标相同，则函数判断为连线合法。
3. 若两个格点横纵坐标均相同，函数将取消格点的点击行为。

进而，程序建立了更为完善的连线处理函数：

1. 连线不合法：若两点间不存在完整的连线，程序不进行任何行为；若两点间存在完整的连线，则取消这条连线。
2. 连线合法：若两点间不存在完整的连线，则绘制一条以这两个点为端点的连线；若两点间存在完整的连线，则取消这条连线。

检查 程序通过构建的二维数组（前文提及的用于储存边的连接情况的二维数组）进行是否满足方格中数字的连接边数的判断，并标示出未满足条件的方格（将方格中数字颜色由红色改为黑色）。

程序通过构建新的逻辑型二维数组，依据连线情况，先将左上角的点坐标赋值为真，将其所在的一段连线的右/下方的点坐标赋值为真，循环进行赋值操作，若最终数组中对应右下方点坐标的值为真，则判断连线连通。

5.2.3 图形界面对游戏环境编写的优势

程序借助图形界面提供的不同颜色绘图机制，美化并丰富了游戏环境。同时程序以图形界面提供的矩形、圆形绘制为基础，进一步创建了矩形边框、圆边矩形等绘图函数，并以此编写多个的游戏界面，实现了初始 logo 界面、分级选关、关卡、通关界面等多样化的界面的绘制。

5.3 24_points

24 点

5.3.1 expression

为了使结果拥有更好的可读性，我们实现了 expression 类，通过递归的树状结构存储算式，在输出时根据运算优先级仅添加必要的括号。并根据二元运算逆运算的性质通过对树进行 DFS 递归进行左旋操作优化表达式结构，使输出结果更优。

5.3.2 求解

我们采用朴素的 DFS，枚举组合中两数及进行的运算，将最终结果与 24 进行考虑浮点误差的运算求解，在常见情形下效率符合实际需求。

5.4 Snakes

5.4.1 一些细节

用数组记录蛇身体坐标点，定期向前进方向移动。糖果的生成由随机生成器控制，概率受蛇身长度以及在场糖果数量控制。蛇身方向的改变由键盘控制，也可由鼠标点击控制，蛇头会转向最接近点击位置的方向。

5.4.2 动画实现

通过 BOOK 事件以及 Timer 指令实现定时通讯，但由于 POST 请求的 header 和 payload 在部分浏览器中并不同时发送而产生了服务器端的固定延时，加之网络质量有波动，实际效果往往会因场景而异。

6 结语

我们用 python 和网页等实现了一个借助浏览器的图形界面，基于其网络运行的特性，对于用户端软件环境几乎没有要求，可以实现一次部署，多次便捷地分享，相较于以前传统的发送可执行文件或是源代码，用户端的操作更简单。而对于开发者，对图形界面的操作被简化为了好上手的正常形式的输入输出，调用操作更为简单。图形界面实现了文字输入、弹窗提醒、更改标题等众多功能，几乎可作为大多数种类简单游戏的实现平台。除此之外，程序可拓展性强，可增加联机功能，成为可被实际使用的产品。

图形界面满足了设计之初简化开发流程以及优化用户端体验的宗旨。