辽 宁 科 技 大 学

课程设计报告

设计题目： 局域网五子棋设计

学院、系： 电信学院计算机系

专业班级： 计算机15-2班

学生姓名： 韦博文、索维

指导教师： 杨松波

成 绩：

2018年 8 月 30 日

**目录**

一．概述 .....................................................1

二．设计介绍 .................................................1

三、流程图 ...................................................2

四、程序实现 .................................................2

1. 功能测试 .................................................8
2. 总结 ......................................................9
3. 参考文献 ...................................................9
4. **概述**
5. 目的：本次课程设计是计算机网络原理课程之后的实践加强环节，通过本设计的训练使学生加深对网络协议的理解，并实践制定通信规则，同时通过实践，进一步掌握套接字编程方法，为大型网络编程打下坚实基础。
6. 内容：通过对传输层协议TCP和UDP和客户/服务器数据通信的原理及方式的掌握，自选任意一种开发工具，利用Winsock控件编写出指定题目的交互信息的软件。
7. 题目：局域网五子棋设计
8. 主要完成任务：实现局域网两台计算机的交互（五子棋的实现）

二**、设计介绍**

（一）游戏规则：

一、阳线和阴线

1、阳线：棋盘上可见的横纵直线。

2、阴线：棋盘上无实线连接的隐形斜线。

二、在棋阳线和阴线的任意一条线上形成的有5个或5个以上的同色棋子不间隔地紧紧相连。

五连：在棋盘上形成的5个同色棋子的“连”。

1. 实现过程：

一、制定通信规范。

二、画出棋盘、棋子，棋盘用画线完成，棋子用画点完成，首先添加一个PaintBox

1 画线 PaintBox.line (x1,y1) – (x2,y2)

2 画点 PaintBox.FillColor = RGB(r,g,b)

PaintBox.FillStyle = VbFSSolid

PaintBox.Circle (x,y) ,radius

3清除 PaintBox.Cls

三、定义一个逻辑变量表示是否可以下棋子，定义一个二维数组存储棋盘上是否有棋子。定义一个变量表示哪一方。

四、处理鼠标单击事件，根据逻辑变量的值判断是否处理，处理时将鼠标单击点坐标转换为棋盘坐标，判断棋盘数组中是否有棋子，无棋子则修改数组、画棋子、发送数据并判断胜负。

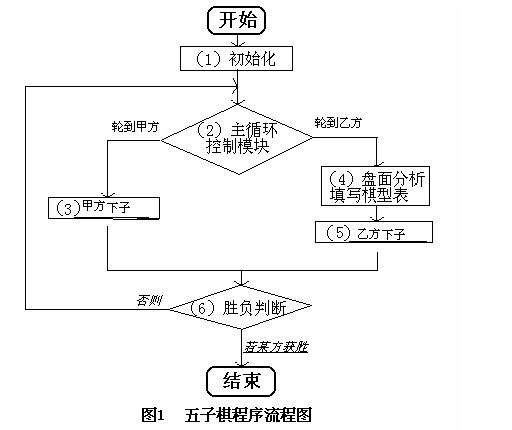
五、接收数据处理：修改棋盘数组、画棋子并判断胜负。

（三）使用语言及编译器

语言：Python

编译器：Pycharm

三**、流程图**



**四、程序实现**

1、服务端窗口实现：

# 初始化棋盘

self.QP = []

for i in range(self.num):

self.QP.append([-1] \* self.num)

tk = Tk()

tk.geometry(str((self.num + 1) \* self.mesh + 2 \* self.px) + 'x' + str((self.num + 1) \* self.mesh + self.py + self.px))

tk.title('五子棋')

# 构造棋盘界面

self.asdf = Canvas(tk, width=(self.num + 1) \* self.mesh + 2 \* self.px, height=(self.num + 1) \* self.mesh + self.py + self.px)

self.asdf.place(x=0, y=0)

self.asdf.create\_rectangle(0, 0, (self.num + 1) \* self.mesh + 2 \* self.px, (self.num + 1) \* self.mesh + self.py + self.px, fill="#d7a605")

self.canvas = Canvas(tk, width=str((self.num + 1) \* self.mesh), height=str((self.num + 1) \* self.mesh))

self.canvas.place(x=self.px, y=self.py)

self.canvas.create\_rectangle(self.mesh - 20, self.mesh - 20, self.mesh \* self.num + 20, self.mesh \* self.num + 20, fill="#d7a65b")

for i in range(self.num):

self.canvas.create\_line(self.mesh, self.mesh \* (i + 1), self.mesh \* self.num, self.mesh \* (i + 1))

self.canvas.create\_line(self.mesh \* (i + 1), self.mesh, self.mesh \* (i + 1), self.mesh \* self.num)

self.canvas.bind("<Button-1>", self.callback)

# 中间的文字

self.a = Label(tk, text=self.key[self.tag], fg=self.color[self.tag], bg='green', font=("Times", "14", "bold"))

self.b = Label(tk, text="走棋", fg=self.color[self.tag], bg='green', font=("Times", "14", "bold"))

self.a.place(x=2 \* self.px + 60 + 10 + 90, y=(self.py - self.high) / 2 + 4)

self.b.place(x=(self.num + 1) \* self.mesh + self.px - self.wide - self.px - 10 - 42 - 90, y=(self.py - self.high) / 2 + 4)

t1 = threading.Thread(target=self.acceptMessage, args=(self.server,self.x,self.y,self))

t1.start()

tk.mainloop()

1. 客户端窗口实现：

# 初始化棋盘

self.QP = []

for i in range(self.num):

self.QP.append([-1] \* self.num)

# 构造棋盘界面

self.asdf = Canvas(frame4, width=(self.num + 1) \* self.mesh + 2 \* self.px,

height=(self.num + 1) \* self.mesh + self.py + self.px)

self.asdf.place(x=0, y=0)

self.asdf.create\_rectangle(0, 0, (self.num + 1) \* self.mesh + 2 \* self.px,

(self.num + 1) \* self.mesh + self.py + self.px, fill="#d7a65b")

self.canvas = Canvas(frame4, width=str((self.num + 1) \* self.mesh), height=str((self.num + 1) \* self.mesh))

self.canvas.place(x=self.px, y=self.py)

self.canvas.create\_rectangle(self.mesh - 20, self.mesh - 20, self.mesh \* self.num + 20,self.mesh \* self.num + 20, fill="#d7a65b")

for i in range(self.num):

self.canvas.create\_line(self.mesh, self.mesh \* (i + 1), self.mesh \* self.num, self.mesh \* (i + 1))

self.canvas.create\_line(self.mesh \* (i + 1), self.mesh, self.mesh \* (i + 1), self.mesh \* self.num)

self.canvas.bind("<Button-1>", self.callback)

# 中间的文字

self.a = Label(frame4, text=self.key[self.tag], fg=self.color[self.tag], bg='#d7a65b',

font=("Times", "14", "bold"))

self.b = Label(frame4, text="走棋", fg=self.color[self.tag], bg='#d7a65b', font=("Times", "14", "bold"))

self.a.place(x=2 \* self.px + 60 + 10 + 90, y=(self.py - self.high) / 2 + 4)

self.b.place(x=(self.num + 1) \* self.mesh + self.px - self.wide - self.px - 10 - 42 - 90,

y=(self.py - self.high) / 2 + 4)

frame3.pack(side=TOP)

frame4.pack(side=BOTTOM)

frame5.pack(side=TOP)

frame6.pack(side=BOTTOM)

frame1.pack(side=LEFT)

frame2.pack(side=RIGHT)

root.mainloop()

1. 点击棋盘时双端显示的方法：

# 画棋子

def callback(self, event):

if self.te:

color = ["black", "black"]

data = str(event.x) + ":" + str(event.y)

self.client.send(data.encode("utf-8"))

x = round(event.x / self.mesh) - 1

y = round(event.y / self.mesh) - 1

errorX = self.mesh \* (x + 1) - event.x

errorY = self.mesh \* (y + 1) - event.y

dis = (errorX \*\* 2 + errorY \*\* 2) \*\* 0.5

if self.QP[x][y] == -1 and dis < self.K / 2 \* self.mesh and self.stop == 0:

self.a.config(text=self.key[(self.tag + 1) % 2], fg=color[(self.tag + 1) % 2])

self.QP[x][y] = self.tag

self.canvas.create\_oval(self.mesh \* (x + 1) - self.Qr, self.mesh \* (y + 1) - self.Qr,

self.mesh \* (x + 1) + self.Qr, self.mesh \* (y + 1) + self.Qr,

fill=self.color[self.tag])

v = [[0, 1], [1, 0], [1, 1], [1, -1]]

for i in v:

x1, y1 = x, y

while x1 < self.num - 1 and x1 > 0 and y1 > 0 and y1 < self.num - 1:

x1 += i[0]

y1 += i[1]

count = 0

while x1 <= self.num - 1 and x1 >= 0 and y1 >= 0 and y1 <= self.num - 1:

if self.QP[x1][y1] == self.tag:

count += 1

if count == 5:

self.win()

else:

count = 0

x1 -= i[0]

y1 -= i[1]

self.tag = (self.tag + 1) % 2

self.tagx, self.tagy = x, y

self.te = 0

def callbackServer(self, eventx, eventy):

# global tag, tagx, tagy, a

color = ["white", "white"]

x = round(eventx / self.mesh) - 1

y = round(eventy / self.mesh) - 1

errorX = self.mesh \* (x + 1) - eventx

errorY = self.mesh \* (y + 1) - eventy

dis = (errorX \*\* 2 + errorY \*\* 2) \*\* 0.5

if self.QP[x][y] == -1 and dis < self.K / 2 \* self.mesh and self.stop == 0:

self.a.config(text=self.key[(self.tag + 1) % 2], fg=color[(self.tag + 1) % 2])

self.QP[x][y] = self.tag

self.canvas.create\_oval(self.mesh \* (x + 1) - self.Qr, self.mesh \* (y + 1) - self.Qr,

self.mesh \* (x + 1) + self.Qr, self.mesh \* (y + 1) + self.Qr,

fill=color[self.tag])

v = [[0, 1], [1, 0], [1, 1], [1, -1]]

for i in v:

x1, y1 = x, y

while x1 < self.num - 1 and x1 > 0 and y1 > 0 and y1 < self.num - 1:

x1 += i[0]

y1 += i[1]

count = 0

while x1 <= self.num - 1 and x1 >= 0 and y1 >= 0 and y1 <= self.num - 1:

if self.QP[x1][y1] == self.tag:

count += 1

if count == 5:

self.win()

else:

count = 0

x1 -= i[0]

y1 -= i[1]

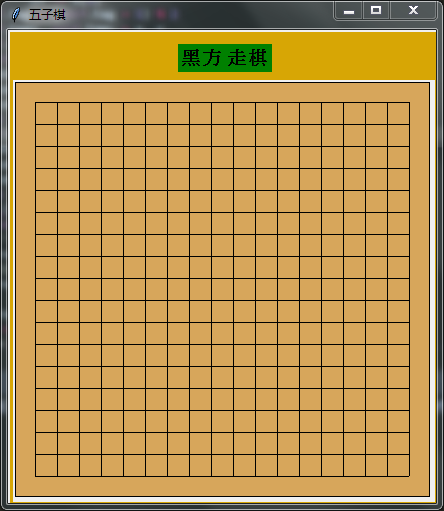
self.tag = (self.tag + 1) % 2

self.tagx, self.tagy = x, y

self.te = 1

1. **功能测试**

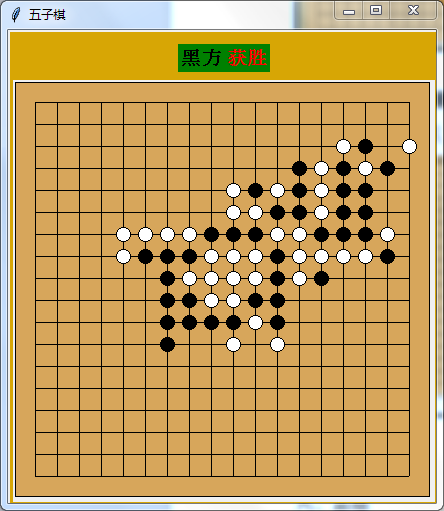
服务端窗口：

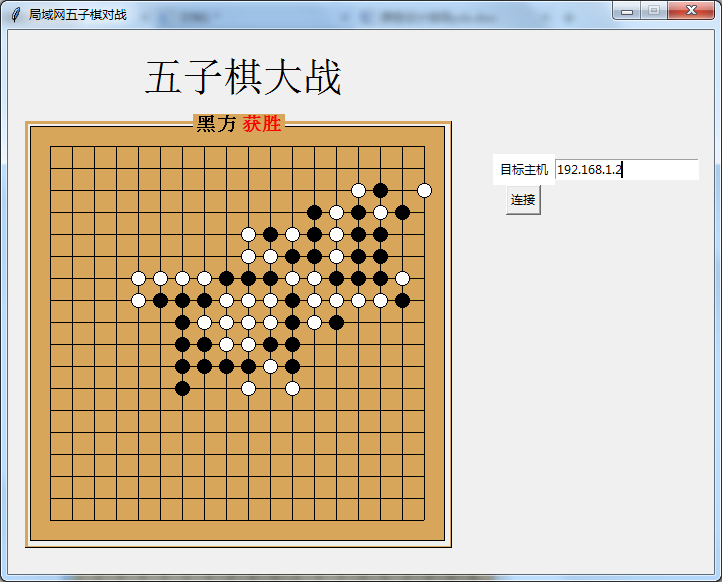


客户端窗口：



游戏进行及结果：





**六、总结**

本次实验时间虽短，但收获颇丰。由于C++及Java语言是最初学习的高级语言，现在基本忘记，所以用了自己比较熟悉的Python语言进行设计。还好遇见问题虽多但慢慢克难攻坚也都悉数解决。整个实验完成后自己对计算机网络有了新的了解。在此之前学习的课程实现的无论是简单的程序也好小游戏也好都只是单机版的，只能在一个电脑上独自使用。在此次设计过程中确实体会到了网络编程的强大之处，他可以使我们我们不仅仅单纯的和计算机进行交互，还可以和局域网内的其他用户进行交流通信。

**七、参考文献**

1、CSDN博客

2、Python中文社区

3、tkinter官方使用文档