# 8.2

1. import tkinter as tk
2. import math
3. def draw\_rect (x, y, w, a):
4. # -----------begin----------- #
5. coord = (x, y, x - w \* math.cos(math.radians(a)),
6. y + w \* math.sin(math.radians(a)),
7. # 使用给定的坐标和参数来计算矩形的四个顶点的坐标
8. x - math.sqrt(2) \* w \* math.sin(math.radians(a + 45)),
9. y - math.sqrt(2) \* w \* math.cos(math.radians(a + 45)),
10. x - w \* math.sin(math.radians(a)),
11. y - w \* math.cos(math.radians(a)))
12. # 使用正弦和余弦函数，根据矩形的宽度 w 和角度 a，计算得到相对于起始点 (x, y) 的偏移量。
13. canvas.create\_polygon(coord, fill='', outline='black')
14. # ------------end------------ #
15. if \_\_name\_\_ == '\_\_main\_\_':
16. root = tk.Tk()
17. root.geometry('1000x800')
18. canvas = tk.Canvas(root, bg="white", highlightthickness=1,
19. highlightbackground="tomato")
20. canvas.config(width=1000, height=700)
21. canvas.pack()
22. a = 0
23. # 实现边长增加5，旋转度数增加5
24. # for 循环来绘制一系列逐渐变大并逐渐旋转的矩形
25. for i in range(5, 301, 5):
26. draw\_rect( 400, 400, i, a)
27. canvas.update()
28. a += 5
29. root.mainloop()

# 8.3

1. import tkinter as tk
2. def draw\_canvas():
3. ############### Begin ##############
4. root = tk.Tk()
5. cv\_small = tk.Canvas(root, width=220, height=180)
6. cv\_small.pack()
7. cv\_small.delete("all") # 清除画布
8. cv\_small.create\_line(50, 30, 110, 90, fill='red', width=2) # 左臂
9. cv\_small.create\_line(170, 30, 110, 90, fill='red', width=2) # 右臂
10. cv\_small.create\_line(110, 160, 110, 90, fill='red', width=2) # 中心线
11. ############### End ################
12. root.mainloop()
13. draw\_canvas() # 调用函数绘制画布

# 8.4

1. import tkinter as tk
2. import math
3. def Y\_tree(depth, percent, left\_angle, right\_angle, start\_point, angle, length, canvas):
4. if depth == 0:
5. return
6. ################## Begin ####################
7. end\_point = (start\_point[0] + length \* math.cos(math.radians(angle)),
8. start\_point[1] - length \* math.sin(math.radians(angle)))
9. canvas.create\_line(start\_point[0], start\_point[1], end\_point[0], end\_point[1], width=2)
10. Y\_tree(depth - 1, percent, left\_angle, right\_angle, end\_point, angle + right\_angle, length \* percent, canvas) # 画右支
11. Y\_tree(depth - 1, percent, left\_angle, right\_angle, end\_point, angle - left\_angle, length \* percent, canvas) # 画左支
12. ################## End ######################
13. if \_\_name\_\_ == '\_\_main\_\_':
14. root = tk.Tk()
15. canvas = tk.Canvas(root, width=1200, height=1000)
16. canvas.pack()
17. start\_point = (300, 600)
18. angle = 45
19. length = 200
20. depth = 12
21. percent = 0.7
22. left\_angle = 30
23. right\_angle = 30
24. Y\_tree(depth, percent, left\_angle, right\_angle, start\_point, angle, length, canvas)
25. root.mainloop()

# 8.5

1. cv\_big = tk.Canvas(root, bg="white", highlightthickness=1, highlightbackground="LightBlue")
2. cv\_big.config(width=cv\_big\_width, height=cv\_big\_height)
3. cv\_big.place(x=30, y=10, anchor='nw')
4. # 创建红色边框的小画布
5. cv\_small\_width = cv\_big\_width // 4
6. cv\_small\_height = cv\_big\_height // 4
7. cv\_small = tk.Canvas(root, bg="white", highlightthickness=1, highlightbackground="red")
8. cv\_small.config(width=cv\_small\_width, height=cv\_small\_height)
9. cv\_small.place(x=30, y=cv\_big\_height - cv\_small\_height + 10, anchor='nw')

# 8．6

1. # Y形基本图案函数
2. def yPattern(cv\_small):
3. ######### Begin ############
4. cv\_small.delete("all") # 清除画布
5. cv\_small.create\_line(30, 10, 90, 70, fill='red', width=2) # 左臂
6. cv\_small.create\_line(150, 10, 90, 70, fill='red', width=2) # 右臂
7. cv\_small.create\_line(90, 140, 90, 70, fill='red', width=2)
8. ######### End ##############

# 8.7

1. # Y形分形参数更新函数
2. def update\_params\_Y(params\_text):
3. new\_str = params\_text.get("1.0", "end-1c") # 获取文本框中的分形参数字符串
4. new\_params = new\_str.split('\n')
5. new\_params = [x.split('=')[1].strip() for x in new\_params] # 获取参数值
6. # 参数更新
7. FinishLevel\_Y = int(new\_params[0])
8. TreePercent\_Y = float(new\_params[1])
9. LeftAngle\_Y = int(new\_params[2])
10. RightAngle\_Y = int(new\_params[3])
11. InitPoint\_Y = tuple(map(int, new\_params[4][1:-1].split(',')))
12. InitAngle\_Y = int(new\_params[5])
13. InitLong\_Y = int(new\_params[6])
14. return FinishLevel\_Y, TreePercent\_Y, LeftAngle\_Y, RightAngle\_Y, InitPoint\_Y, InitAngle\_Y, InitLong\_Y

# 8.8

import math

#################### Begin #######################

def yPattern(cv\_small):

    cv\_small.delete("all")  # 清除画布

    cv\_small.create\_line(30, 10, 90, 70, fill='red', width=2)  # 左臂

    cv\_small.create\_line(150, 10, 90, 70, fill='red', width=2)  # 右臂

    cv\_small.create\_line(90, 140, 90, 70, fill='red', width=2)

def Y\_tree(depth, percent, left\_angle, right\_angle, start\_point, angle, length, canvas):

    if depth == 0:

        return

    end\_point = (start\_point[0] + length \* math.cos(math.radians(angle)),

                 start\_point[1] - length \* math.sin(math.radians(angle)))

    canvas.create\_line(start\_point, end\_point, fill='LightBlue4', width=2)

    Y\_tree(depth - 1, percent, left\_angle, right\_angle, end\_point, angle + right\_angle, length \* percent,

           canvas)  # 画右枝

    Y\_tree(depth - 1, percent, left\_angle, right\_angle, end\_point, angle - left\_angle, length \* percent,

           canvas)  # 画左枝

##################### End ########################

# 8.9

import tkinter as tk

def radio\_frame(frame, shape\_var, ft1):

    # 添加单选框到 Frame组件

    rb\_Y = tk.Radiobutton(frame, text="Y形分形 Y\_Fractal", variable=shape\_var, value='Y', bg="white", font=ft1)

    rb\_Y.pack(anchor='w')

    rb\_Triangle = tk.Radiobutton(frame, text="三角形分形 Triangle\_Fractal", variable=shape\_var, value='Triangle',bg="white", font=ft1)

    rb\_Triangle.pack(anchor='w')

    rb\_Rect = tk.Radiobutton(frame, text="矩形分形 Rect\_Fractal", variable=shape\_var,value='Rect', bg="white",font=ft1)

    rb\_Rect.pack(anchor='w')

    ################# End ####################

    # 将默认选项设置为 Y 形分形

    rb\_Y.select()

def multiline\_text\_box(root,ft3):

    # 创建一个多行文本框

    params\_text = tk.Text(root, bg="white", highlightthickness=1, highlightbackground="LightBlue")

    params\_text.configure(font=ft3, spacing1=15)

    params\_text.place(x=20, y=300, width=480, height=310, anchor='nw')

    # 初始化分形参数并输出

    params\_Y = "your\_params\_Y"  # 替换为实际的分形参数字符串

    params\_text.insert(tk.INSERT, params\_Y)

    ################# Begin ##################

def bottom\_box(root,ft3):

    # 创建底部输出框

    output\_text = tk.Text(root, bg="white", highlightthickness=1,highlightbackground="LightBlue")

    output\_text.configure(font=ft3, spacing1=8)

    text = " 结果输出：<分形绘制参数的字符串形式输出>"

    output\_text.insert(tk.INSERT, text)

    output\_text.place(x=20, y=800, width=1240, height=80, anchor='sw')

# 8.10

第一关

1. def run\_Y\_tree(cv\_big,output\_text):
2. # Y形分形树函数
3. def Y\_tree(depth, percent, left\_angle, right\_angle, start\_point, angle, length, canvas):
4. if depth == 0:
5. return
6. end\_point = (start\_point[0] + length \* math.cos(math.radians(angle)),
7. start\_point[1] - length \* math.sin(math.radians(angle)))
8. canvas.create\_line(start\_point, end\_point, fill='LightBlue4', width=2)
9. # canvas.update()
10. # canvas.after(1)
11. Y\_tree(depth - 1, percent, left\_angle, right\_angle, end\_point, angle + right\_angle, length \* percent,
12. canvas) # 画右枝
13. Y\_tree(depth - 1, percent, left\_angle, right\_angle, end\_point, angle - left\_angle, length \* percent,
14. canvas) # 画左枝
15. # print(start\_point,end\_point)
16. # 清除画布
17. cv\_big.delete("all")
18. # 将初始参数传入函数
19. Y\_tree(FinishLevel\_Y, TreePercent\_Y, LeftAngle\_Y, RightAngle\_Y, InitPoint\_Y, InitAngle\_Y, InitLong\_Y, cv\_big)
20. output\_text.delete("1.0", "end")
21. # 在底部的结果输出框输出分形树的绘制参数
22. text = " 结果输出：<分形绘制参数的字符串形式输出>\n finishLevel={}, treePercent={}," \
23. " leftAngle={}, rightAngle={}, initPoint={}, initAngle={}, initLong={}" \
24. .format(FinishLevel\_Y, TreePercent\_Y, LeftAngle\_Y, RightAngle\_Y, InitPoint\_Y,
25. InitAngle\_Y, InitLong\_Y)
26. print(text)
27. output\_text.insert(tk.INSERT, text)

第二关

def Y\_pattern(cv\_small):

cv\_small.delete("all") # 清除画布

cv\_small.create\_line(50, 30, 110, 90, fill='red', width=2) # 左臂

cv\_small.create\_line(170, 30, 110, 90, fill='red', width=2) # 右臂

cv\_small.create\_line(110, 160, 110, 90, fill='red', width=2) # 中心线

# 三角形基本图案函数

def Triangle\_Pattern(cv\_small):

cv\_small.delete("all") # 清除画布

cv\_small.create\_polygon(80, 130, 20, 10, 140, 10, outline='red', fill='white', width=2)

# 方形基本图案函数

def Rect\_Pattern(cv\_small):

cv\_small.delete("all") # 清空画布

cv\_small.create\_rectangle(35, 10, 155, 120, outline='red', fill='white', width=2)

第三关

1. # Y形分形参数更新函数
2. def update\_params\_Y(params\_text):
3. new\_str = params\_text.get("1.0", "end-1c") # 获取文本框中的分形参数字符串
4. new\_params = new\_str.split('\n')
5. new\_params = [x.split('=')[1].strip() for x in new\_params] # 获取参数值
6. # 参数更新
7. FinishLevel\_Y = int(new\_params[0])
8. TreePercent\_Y = float(new\_params[1])
9. LeftAngle\_Y = int(new\_params[2])
10. RightAngle\_Y = int(new\_params[3])
11. InitPoint\_Y = tuple(map(int, new\_params[4][1:-1].split(',')))
12. InitAngle\_Y = int(new\_params[5])
13. InitLong\_Y = int(new\_params[6])
14. return FinishLevel\_Y, TreePercent\_Y, LeftAngle\_Y, RightAngle\_Y, InitPoint\_Y, InitAngle\_Y, InitLong\_Y
15. # 三角形分形参数设置
16. def update\_params\_Triangle(params\_text):
17. new\_str = params\_text.get("1.0", "end-1c") # 获取文本框中的分形参数字符串
18. new\_params = new\_str.split('\n')
19. new\_params = [x.split('=')[1].strip() for x in new\_params] # 获取参数值
20. # 参数更新
21. FinishLevel\_tri = int(new\_params[0])
22. TreePercent\_tri = float(new\_params[1])
23. InitPoint\_tri = tuple(map(int, new\_params[2][1:-1].split(',')))
24. InitAngle\_tri = int(new\_params[3])
25. InitLong\_tri = int(new\_params[4])
26. return FinishLevel\_tri, TreePercent\_tri, InitPoint\_tri, InitAngle\_tri, InitLong\_tri
27. # 方形参数更新函数
28. def update\_params\_Rect(params\_text):
29. new\_str = params\_text.get("1.0", "end-1c") # 获取文本框中的分形参数字符串
30. new\_params = new\_str.split('\n')
31. new\_params = [x.split('=')[1].strip() for x in new\_params] # 获取参数值
32. # 参数更新
33. FinishLevel\_Rect = int(new\_params[0])
34. LeftPoint\_Rect = tuple(map(int, new\_params[1][1:-1].split(',')))
35. RightPoint\_Rect = tuple(map(int, new\_params[2][1:-1].split(',')))
36. InitAngle\_Rect = int(new\_params[3])
37. return FinishLevel\_Rect, LeftPoint\_Rect, RightPoint\_Rect, InitAngle\_Rect

第四关

1. import tkinter as tk
2. def add\_fractal\_ne(frame,shape\_var,ft3):
3. # 按钮【添加】的回调函数，将分形图案对应的函数模块增加到相关字典
4. def add\_function():
5. # 获取分形名称，更改单选框名字
6. name\_fractal = text\_name.get('1.0', 'end-1c') # 文本框控件中第一个字符的位置是 1.0,'end-1c'返回的输入中不包含换行符
7. # 获取分形单选框的值
8. fractal\_type = text\_radio.get('1.0', 'end-1c')
9. # 创建单选框
10. rb = tk.Radiobutton(frame, text=name\_fractal, variable=shape\_var, value=fractal\_type, bg="white", font=ft3)
11. rb.pack(anchor='w')
12. # 添加后关闭窗口
13. small\_root2.destroy()
14. # 弹出一个添加分形代码的窗口
15. small\_root2 = tk.Tk()
16. small\_root2.geometry("1000x700+200+50")
17. small\_root2.title("新增分形图案类型")
18. # 分形名称，读取为单选框的名字
19. lab\_name = tk.Label(small\_root2, text="分形名称：", font=ft3)
20. lab\_name.place(x=40, y=10, width=100, height=30, anchor='nw')
21. text\_name = tk.Text(small\_root2, highlightthickness=1, highlightbackground="LightBlue")
22. text\_name.configure(font=ft3)
23. text\_name.place(x=140, y=10, width=340, height=30)
24. # 分形单选框的值，读取为单选框的值
25. lab\_radio = tk.Label(small\_root2, text="分形英文简写：", font=ft3)
26. lab\_radio.place(x=520, y=10, width=140, height=30, anchor='nw')
27. text\_radio = tk.Text(small\_root2, highlightthickness=1, highlightbackground="LightBlue")
28. text\_radio.configure(font=ft3)
29. text\_radio.place(x=660, y=10, width=300, height=30)
30. # 分形参数初始化，读取的代码可以直接运行
31. lab\_param = tk.Label(small\_root2, text="请输入分形参数初始化的代码：", font=ft3)
32. lab\_param.place(x=40, y=50, width=280, height=30, anchor='nw')
33. text\_param = tk.Text(small\_root2, highlightthickness=1, highlightbackground="LightBlue")
34. text\_param.configure(font=ft3, spacing1=5)
35. text\_param.place(x=40, y=85, width=440, height=140)
36. # 分形参数字符串形式输出，后续读取代码到分形参数输出字典str\_params\_dict
37. lab\_str = tk.Label(small\_root2, text="请输入分形参数字符串形式输出的代码：", font=ft3)
38. lab\_str.place(x=40, y=230, width=360, height=30, anchor='nw')
39. text\_str = tk.Text(small\_root2, highlightthickness=1, highlightbackground="LightBlue")
40. text\_str.configure(font=ft3, spacing1=5)
41. text\_str.place(x=40, y=265, width=440, height=140)
42. # print(text\_str)
43. # 参数更新，后续读取代码到参数更新字典update\_params\_dict
44. lab\_up = tk.Label(small\_root2, text="请输入分形参数更新的代码：", font=ft3)
45. lab\_up.place(x=40, y=410, width=260, height=30, anchor='nw')
46. text\_up = tk.Text(small\_root2, highlightthickness=1, highlightbackground="LightBlue")
47. text\_up.configure(font=ft3, spacing1=5)
48. text\_up.place(x=40, y=445, width=440, height=145)
49. # 输入基本图案函数，后续读取代码到基本图案绘制函数字典pattern\_dict
50. lab\_pattern = tk.Label(small\_root2, text="请输入基本图案绘制函数：", font=ft3)
51. lab\_pattern.place(x=520, y=50, width=240, height=30, anchor='nw')
52. text\_pattern = tk.Text(small\_root2, highlightthickness=1, highlightbackground="LightBlue")
53. text\_pattern.configure(font=ft3, spacing1=5)
54. text\_pattern.place(x=520, y=85, width=440, height=240)
55. # 输入分形树函数，后续读取代码到分形树绘制函数字典tree\_dict
56. lab\_tree = tk.Label(small\_root2, text="请输入分形树绘制函数：", font=ft3)
57. lab\_tree.place(x=520, y=330, width=220, height=30, anchor='nw')
58. text\_tree = tk.Text(small\_root2, highlightthickness=1, highlightbackground="LightBlue")
59. text\_tree.configure(font=ft3, spacing1=5)
60. text\_tree.place(x=520, y=365, width=440, height=230)
61. # 按钮【添加】
62. button\_add = tk.Button(small\_root2, text="添加", command=add\_function, font=ft3)
63. button\_add.place(x=450, y=610, width=100, height=35, anchor='nw')

# 12.1

第一关

#!usr/bin/python

import random

from PIL import Image, ImageTk

import math

import random

import time

class Chess:

    def \_\_init\_\_(self):

        # 设计五子棋运行的数据结构

        ########### Begin ###########

        # 请在此开编写你的的代码

        self.ChessData = [[{"Cstate": 0, "Cstep": 0} for j in range(19)] for i in range(19)]

        ########### End ###########

        print(self.ChessData)

        self.Currently\_step = 0   #棋子当前进行的步数

if \_\_name\_\_ == '\_\_main\_\_':

    chess = Chess()

  第二关

#!usr/bin/python

import random

from PIL import Image, ImageTk

import math

import random

import time

class Chess:

    def \_\_init\_\_(self):

        # 设计五子棋运行的数据结构

        self.ChessData = [[{"Cstate": 0, "Cstep": 0} for j in range(19)] for i in range(19)]

        self.Currently\_step = 0   #棋子当前进行的步数

    def main(self):

        i = random.randint(0, 18)

        j = random.randint(0, 18)

        # 使ChessDate[i][j]点的CState值轮流为1和2，其步骤计数保存于相应点的CStep中

        ########### Begin ###########

        # 请在此开编写你的的代码

        for color in range(1, 3):

            self.ChessData[i][j]['Cstate'] = color

            self.Currently\_step += 1

            self.ChessData[i][j]['Cstep'] = self.Currently\_step

        ########### End ###########

if \_\_name\_\_ == '\_\_main\_\_':

    chess = Chess()

    chess.main()

# 12.2

第一关

a=0

b=1

a+=1

第二关：

from tkinter import \*

from PIL import Image, ImageTk

from Step1\_c import \*

def cross():

    for i in range(19):  # 横线

        point = [[40, 37], [706, 37]]

        point[0][1] = 21 + i \* 34.3

        point[1][1] = 21 + i \* 34.3

        ########## Begin ##########

        # 请在此开始编写你的的代码

        if i==0 or i==18:

            canvas.create\_line(point, fill="#C18B5F", width=2.5)

        canvas.create\_line(point, fill="#C18B5F", width=1.5)

        ########## End ##########

    for i in range(19):  # 竖线

        point = [[41, 21], [41, 640]]

        point[0][0] = 41 + i \* 37

        point[1][0] = 41 + i \* 37

        ########## Begin ##########

        # 请在此开始编写你的的代码

        if i==0 or i==18:

            canvas.create\_line(point, fill="#C18B5F", width=2.5)

        canvas.create\_line(point, fill="#C18B5F", width=1.5)

        ########## End ##########

    for x in (4, 9, 14):

        for y in (4, 9, 14):

            if x == y == 9:

                radius = 9

            else:

                radius = 6

            canvas.create\_oval(x \* 36.9 + 41 - radius / 2,

                               y \* 34.3 + 21 - radius / 2,

                               x \* 36.9 + 41 + radius / 2,

                               y \* 34.3 + 21 + radius / 2,

                               fill='#BE875E', outline="")

if \_\_name\_\_ == '\_\_main\_\_':

    cross()

    canvas.pack()

    root.mainloop()

# 12.3

from Step2\_c import \*

ChessData = [[{"Cstate": 0, "Cstep": 0} for j in range(19)] for i in

             range(19)]  # 初始化棋局

# 图片创建

blackch = Image.open("/data/workspace/myshixun/finalchess/image/黑子-小.png").resize((40, 40))

blackch = ImageTk.PhotoImage(blackch)

whitech = Image.open("/data/workspace/myshixun/finalchess/image/白子-小.png").resize((40, 40))

whitech = ImageTk.PhotoImage(whitech)

# 复原棋局

def resetlast():

    file = open('/data/workspace/myshixun/finalchess/data.txt', 'r+', encoding='utf-8')

    data = file.readlines()

    rtu = []

    for i in data:

        rtu.append(i.strip())

    for i in range(0, 19):

        j = 0

        rtu1 = rtu[i].split("\t")

        for x in rtu1:

            ChessData[i][j]['Cstate'] = int(x)

            j += 1

    for i in range(19, 38):

        j = 0

        rtu1 = rtu[i].split("\t")

        for x in rtu1:

            ChessData[i - 19][j]['Cstep'] = int(x)

            j += 1

    file.close()

    for x in range(19):

        for y in range(19):

            # 判断棋子的颜色，用create\_image（）函数创建棋子图片

            ########## Begin ##########

            # 请在此开始编写你的的代码

            if ChessData[x][y]['Cstate'] == 1:  # 白棋

                canvas.create\_image(x \* 36.8 + 41 - 14, y \* 34.6 + 21, image=whitech, anchor=W)

            elif ChessData[x][y]['Cstate'] == 2:  # 黑棋

                canvas.create\_image(x \* 36.8 + 41 - 14, y \* 34.6 + 21, image=blackch, anchor=W)

            ########## End ##########

                # 如果该位置已经下过黑子，并且步数不为0，则在该位置显示黑子的步数

            if ChessData[x][y]['Cstep'] != 0 and ChessData[x][y]['Cstate'] == 2:

                reviewnum = Label(root, text=str(ChessData[x][y]['Cstep']), bg='#000', fg='#fff',

                                  font=("黑体", 8), width=1, height=1)

                reviewnum.place(x=x \* 36.8 + 41 - 7, y=y \* 34.6 + 21 - 9, anchor='nw')

            # 如果该位置已经下过白子，并且步数不为0，则在该位置显示白子的步数

            elif ChessData[x][y]['Cstep'] != 0 and ChessData[x][y]['Cstate'] == 1:

                reviewnum = Label(root, text=str(ChessData[x][y]['Cstep']), bg='#fff', fg='#000',

                                  font=("黑体", 8), width=1, height=1)

                reviewnum.place(x=x \* 36.8 + 41 - 7, y=y \* 34.6 + 21 - 9, anchor='nw')

if \_\_name\_\_ == '\_\_main\_\_':

    cross()

    resetlast()

    canvas.pack()

    root.mainloop()

# 12.4

from Step0 import \*

# 按钮重新开始

def refresh():

    canvas.create\_image(750, 145, image=photos2, anchor=W)

    canvas.create\_image(870, 145, image=photos2, anchor=W)

    photoqizi = Image.open("/data/workspace/myshixun/finalchess/image/棋盘-空.png").resize((730, 700))

    photoqizi = ImageTk.PhotoImage(photoqizi)

    canvas.create\_image(10, 350, image=photoqizi, anchor=W)

    cross()

    if chess.Ovalone != 0:

        canvas.delete(chess.Ovalone)

        # 重置棋盘信息

        ########### Begin ###########

        # 请在此开始编写你的的代码

        canvas.delete(chess.Ovalone)

        ########### End ###########

    chess.Currently\_step = 0

    chess.Gameover = 0

    chess.Depth = 0

    chess.player = 0  # 轮到下棋的标志，1=下，0=不下

    chess.computer = 0  # 轮到下棋的标志，1=下，0=不下

    chess.myColor = 0  # 玩家选择的棋子颜色

    chess.computercolor = 0  # 电脑的棋子颜色

    chess.player2color = 0  # 玩家2的棋子颜色

    if chess.WinFLAG == 1:

        chess.resultshow(root, canvas, whitech, blackch)

    if chess.Chess\_Mode == 1:

        chess.playing(1, root, canvas, blackch, whitech, photos1, photos2)

        # chess.curlocation(root, player=1, computer=0,Current\_Player=1)

    elif chess.Chess\_Mode == 2:

        chess.playing(2, root, canvas, blackch, whitech, photos1, photos2)

    elif chess.Chess\_Mode == 0:

        chess.playing(0, root, canvas, blackch, whitech, photos1, photos2)

        # chess.curlocation(root, player=1, computer=1,Current\_Player=2)

    if chess.Ovalone != 0:

        canvas.delete(chess.Ovalone)

    chess.WinFLAG = 0

    root.update()

# 重新开始按钮

photorefresh = Image.open("/data/workspace/myshixun/finalchess/image/重新开始.png").resize((100, 50))

photorefresh = ImageTk.PhotoImage(photorefresh)

btn\_reset = Button(root, text='重新开始', font='Arial,12', width=85, height=35,

                   image=photorefresh, command=refresh, bg='#FFA500', bd=0)

btn\_reset.place(x=850, y=220)

if \_\_name\_\_ == '\_\_main\_\_':

    cross()

    resetlast()

    canvas.pack()

    root.mainloop()

# 12.5

from Step0 import \*

# 人机模式（机白）

def peocomwhite():

    if chess.Chess\_Mode == 3:

        chess.Chess\_Mode = 1

        ########## Begin ###########

        # 请在此开始编写你的的代码

        chess.playing(1, root, canvas, blackch, whitech, photos1, photos2)

        ########### End ###########

    else:

        chess.Chess\_Mode = 1

        refresh()

# 人机模式（机黑）

def peocomblack():

    if chess.Chess\_Mode == 3:

        chess.Chess\_Mode = 2

        ########### Begin ###########

        # 请在此开始编写你的的代码

        chess.playing(2, root, canvas, blackch, whitech, photos1, photos2)

        ########### End ###########

    else:

        chess.Chess\_Mode = 2

        refresh()

# "双人模式"

def doublepeople():

    if chess.Chess\_Mode == 3:

        chess.Chess\_Mode = 0

        ########### Begin ###########

        # 请在此开始编写你的的代码

        chess.playing(0, root, canvas, blackch, whitech, photos1, photos2)

        ########### End ###########

    else:

        chess.Chess\_Mode = 0

        refresh()

def startgame():

    win = Tk()

    win.title("开始游戏")

    win.geometry('500x300+100+100')

    Label(win, text="五子棋", width=120, height=2).place(x=-175, y=50)

    Button(win, text="人机模式（机白）", command=peocomwhite, width=20, height=2).place(x=170, y=100)

    Button(win, text="人机模式（机黑）", command=peocomblack, width=20, height=2).place(x=170, y=150)

    Button(win, text="双人模式", command=doublepeople, width=20, height=2).place(x=170, y=200)

# 创建一个主目录菜单，也被称为顶级菜单

main\_menu = Menu(root)

gamefile = Menu(main\_menu, tearoff=False)

gamefile.add\_command(label="开始游戏", command=startgame)

main\_menu.add\_cascade(label="游戏", menu=gamefile)  # 在主目录菜单上新增"文件"选项，并通过menu参数与下拉菜单绑定

root.config(menu=main\_menu)

if \_\_name\_\_ == '\_\_main\_\_':

    cross()

    canvas.pack()

    root.mainloop()

# 12.6

1. def curlocation(player, computer, Current\_Player):
2. if player == 1 and computer == 0: # 玩家为黑棋，电脑为白棋
3. if Current\_Player == 1: # 落子方为黑方
4. ########### Begin ###########
5. # 请在此开始编写你的的代码
6. print("当前落子方为：玩家")
7. else:
8. print("当前落子方为：电脑")
9. ########### End ###########
10. elif player == 0 and computer == 1: # 玩家为白棋，电脑为黑棋
11. if Current\_Player == 1:
12. ########### Begin ###########
13. # 请在此开始编写你的的代码
14. print("当前落子方为：电脑")
15. else:
16. print("当前落子方为：玩家")
17. ########### End ###########
18. elif player == 1 and computer == 1: # 玩家1为黑棋，玩家2为白棋
19. print("当前落子方为：玩家1")
20. else:
21. print("当前落子方为：玩家2")
22. # 当玩家为黑棋，电脑为白棋时 ，且落子放为黑方
23. curlocation(1, 0, 1)
24. # 当玩家为白棋，电脑为黑棋时 ，且落子放为黑方
25. curlocation(0, 1, 1)

# 12.7

1. from Step2 import \*
2. # 鼠标左击落子事件定义
3. class Player:
4. def PointNextMove(chessData):
5. global playerx, playery
6. playerx = float(format(chessData.x))
7. playery = float(format(chessData.y))
8. if chess.player == 1:
9. if chess.myColor == 2 and chess.computercolor == 1:
10. chess.playgame\_black(root, playerx, playery, canvas, blackch, whitech, result, photos1, photos2)
11. if chess.myColor == 1 and chess.computercolor == 2:
12. chess.playgame\_white(root, playerx, playery, canvas, blackch, whitech, result, photos1, photos2)
13. if chess.myColor == 2 and chess.player2color == 1:
14. chess.doublepeople(root, playerx, playery, canvas, blackch, whitech, result, photos1, photos2)
15. print("落子成功！")
16. canvas.bind('<Button-1>', PointNextMove)
17. class HumanPlayer(Player):
18. def NextMove(self):
19. # 调用player\_location()方法
20. ########## Begin ###########
21. # 请在此开始编写你的的代码
22. chess.player\_location()
23. ########### End ###########
24. class RobotPlayer(Player):
25. def NextMove(self):
26. # 调用chess.ai\_location()方法
27. ########## Begin ###########
28. # 请在此开始编写你的的代码
29. chess.ai\_location()
30. ########### End ###########
31. if \_\_name\_\_ == '\_\_main\_\_':
32. cross()
33. canvas.pack()
34. root.mainloop()

# 12.8

def ChessCheck(file):

    ChessData = [[{"Cstate": 0, "Cstep": 0} for j in range(19)] for i in

                 range(19)]

    file = open(file, 'r+', encoding='utf-8')

    data = file.readlines()

    rtu = []

    for i in data:

        rtu.append(i.strip())

    for i in range(0, 19):

        j = 0

        rtu1 = rtu[i].split("\t")

        for x in rtu1:

            ChessData[i][j]['Cstate'] = int(x)

            j += 1

    for i in range(19, 38):

        j = 0

        rtu1 = rtu[i].split("\t")

        for x in rtu1:

            ChessData[i - 19][j]['Cstep'] = int(x)

            j += 1

    # 1.判断水平方向是否连续五子

    for x in range(15):

        for y in range(19):

            if ChessData[x][y]['Cstate'] == 1 and ChessData[x + 1][y]['Cstate'] == 1 and \

                    ChessData[x + 2][y]['Cstate'] == 1 and ChessData[x + 3][y]['Cstate'] == 1 and \

                    ChessData[x + 4][y]['Cstate'] == 1:

                return print("白棋胜利！")

            if ChessData[x][y]['Cstate'] == 2 and ChessData[x + 1][y]['Cstate'] == 2 and \

                    ChessData[x + 2][y]['Cstate'] == 2 and ChessData[x + 3][y]['Cstate'] == 2 and \

                    ChessData[x + 4][y]['Cstate'] == 2:

                return print("黑棋胜利！")

    ########### Begin ###########

    # 请在此开始编写你的的代码

    # 2.判断垂直是否连续五子

    # 3.判断左斜向是否连续五子

    # 4.判断右斜向是否连续五子

    ########### End ###########

    # 5.判断是否是平局

    avanum = 0

    for x in range(19):

        for y in range(19):

            if ChessData[x][y]['Cstate'] == 0:  # 判断棋盘中是否有空位，有空位则非平局

                break

            elif ChessData[x][y]['Cstate'] != 0:

                avanum += 1

    if avanum == 365:

        return print("平局！")

    return print('白棋胜利！')

if \_\_name\_\_ == '\_\_main\_\_':

    ChessCheck('/data/workspace/myshixun/finalchess/data.txt')

# 12.9

1. def stu\_review(chess,root,Label): # 按钮复盘
2. if chess.WinFLAG == 1:
3. ################ Begin ###################
4. for x in range(19):
5. for y in range(19):
6. if chess.ChessData[x][y]['Cstep'] != 0 and \
7. chess.ChessData[x][y]['Cstate'] == 2:
8. reviewnum = Label(root, text=str(
9. chess.ChessData[x][y]['Cstep']),
10. bg='#000', fg='#fff',font=("黑体", 8),
11. width=1, height=1)
12. reviewnum.place(x=x \* 36.8 + 41 - 7, y=y \*
13. 34.6 + 21 - 9, anchor='nw')
14. chess.Relabel[x][y] = reviewnum
15. elif chess.ChessData[x][y]['Cstep'] != 0 and \
16. chess.ChessData[x][y]['Cstate'] == 1:
17. reviewnum = Label(root, text=str(
18. chess.ChessData[x][y]['Cstep']), bg='#fff',
19. fg='#000',font=("黑体", 8), width=1, height=1)
20. reviewnum.place(x=x \* 36.8 + 41 - 7, y=y \*
21. 34.6 + 21 - 9, anchor='nw')
22. chess.Relabel[x][y] = reviewnum
23. ################ End #####################
24. elif chess.WinFLAG == 0:
25. # show\_info('信息提示！','对局结束后才能复盘噢！')
26. print('信息提示！','对局结束后才能复盘噢！')
27. chess.ifreview = 1
28. root.update()

# 12.10

1. def stu\_concelreview(chess, root):
2. ############ Begin #############
3. if chess.WinFLAG == 1:
4. chess.showHistory = True
5. for x in range(19):
6. for y in range(19):
7. if chess.ChessData[x][y]['Cstep'] != 0 and chess.Relabel[x][y] != 0:
8. chess.Relabel[x][y].destroy()
9. chess.Relabel[x][y] = 0
10. elif chess.WinFLAG == 0:
11. chess.showHistory = False
12. print('信息提示！','在复盘后使用噢！')
13. ############ End ###############
14. root.update()