python+pyGame 黑白棋游戏

作者：[guzhou\_diaoke](http://my.csdn.net/guzhou_diaoke)

用pygame-1.9.2和python3.32写的黑白棋小游戏

核心思想：处理棋盘64个格子里面棋子的颜色。规定黑棋先走，玩家先走或电脑先走随机选择，按照游戏规则（不清楚规则的可以百度下）。

主要难点1，按照规则，找出电脑或玩家可以落子的格子。

主要难点2，电脑下棋的算法。如果电脑在所有落子的选择中，有四个边角，可落子在边角，因为边角的棋子无法被翻转。如果没有边角，则选择可以翻转对手最多的位置落子。

为了学习起来更有趣，继续以游戏的方式来学习。

注：前几天学习了Invent Your Own Computer Games with Python 一书中的一些游戏例子程序，有一个字符界面的黑白棋程序，为了更便于操作和美观，我决定把它改为图形界面的程序，但多数代码修改或借用的原始程序。

1.绘制棋盘

import pygame, sys, random

from pygame.locals import \*

BACKGROUNDCOLOR = (255, 255, 255)

FPS = 40

# 退出

def terminate():

pygame.quit()

sys.exit()

# 初始化

pygame.init()

mainClock = pygame.time.Clock()

# 加载图片

boardImage = pygame.image.load('board.png')

boardRect = boardImage.get\_rect()

blackImage = pygame.image.load('black.png')

blackRect = blackImage.get\_rect()

whiteImage = pygame.image.load('white.png')

whiteRect = whiteImage.get\_rect()

# 设置窗口

windowSurface = pygame.display.set\_mode((boardRect.width, boardRect.height))

pygame.display.set\_caption('黑白棋')

# 游戏主循环

while True:

for event in pygame.event.get():

if event.type == QUIT:

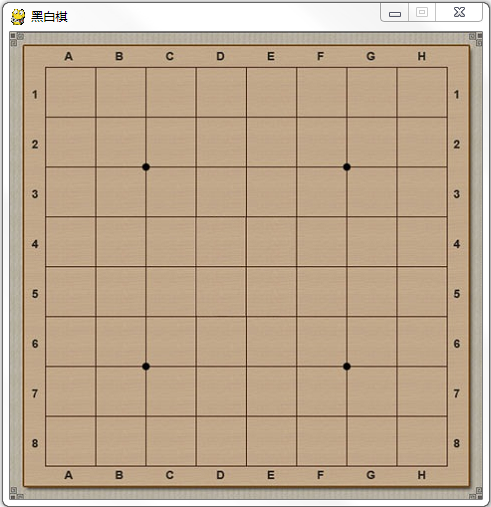
terminate()

windowSurface.fill(BACKGROUNDCOLOR)

windowSurface.blit(boardImage, boardRect, boardRect)

pygame.display.update()

mainClock.tick(FPS)

  
2.绘制棋子

1）黑白棋的规则，开局时先放置上黑白各两个棋子在中间

2）用一个8x8列表保存棋子

CELLWIDTH = 50

CELLHEIGHT = 50

PIECEWIDTH = 47

PIECEHEIGHT = 47

BOARDX = 35

BOARDY = 35

# 重置棋盘

def resetBoard(board):

for x in range(8):

for y in range(8):

board[x][y] = 'none'

# Starting pieces:

board[3][3] = 'black'

board[3][4] = 'white'

board[4][3] = 'white'

board[4][4] = 'black'

# 开局时建立新棋盘

def getNewBoard():

board = []

for i in range(8):

board.append(['none'] \* 8)

return board

mainBoard = getNewBoard()

resetBoard(mainBoard)

for x in range(8):

for y in range(8):

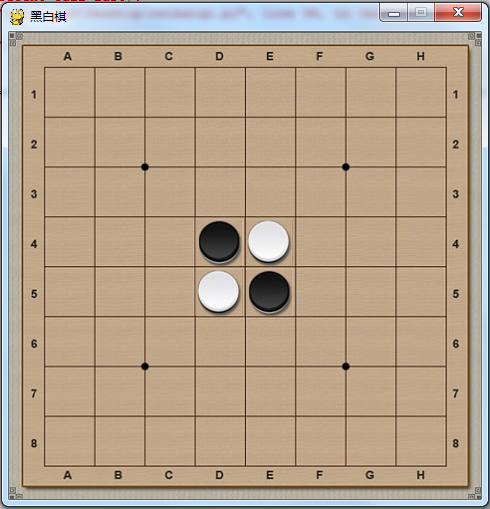
rectDst = pygame.Rect(BOARDX+x\*CELLWIDTH+2, BOARDY+y\*CELLHEIGHT+2, PIECEWIDTH, PIECEHEIGHT)

if mainBoard[x][y] == 'black':

windowSurface.blit(blackImage, rectDst, blackRect)

elif mainBoard[x][y] == 'white':

windowSurface.blit(whiteImage, rectDst, whiteRect)

  
3.鼠标事件

1）鼠标操纵

2）轮流走棋

# 谁先走

def whoGoesFirst():

if random.randint(0, 1) == 0:

return 'computer'

else:

return 'player'

turn = whoGoesFirst()

if turn == 'player':

playerTile = 'black'

computerTile = 'white'

else:

playerTile = 'white'

computerTile = 'black'

for event in pygame.event.get():

if event.type == QUIT:

terminate()

if turn == 'player' and event.type == MOUSEBUTTONDOWN and event.button == 1:

x, y = pygame.mouse.get\_pos()

col = int((x-BOARDX)/CELLWIDTH)

row = int((y-BOARDY)/CELLHEIGHT)

if makeMove(mainBoard, playerTile, col, row) == True:

if getValidMoves(mainBoard, computerTile) != []:

turn = 'computer'

windowSurface.fill(BACKGROUNDCOLOR)

windowSurface.blit(boardImage, boardRect, boardRect)

if (turn == 'computer'):

x, y = getComputerMove(mainBoard, computerTile)

makeMove(mainBoard, computerTile, x, y)

savex, savey = x, y

# 玩家没有可行的走法了

if getValidMoves(mainBoard, playerTile) != []:

turn = 'player'

windowSurface.fill(BACKGROUNDCOLOR)

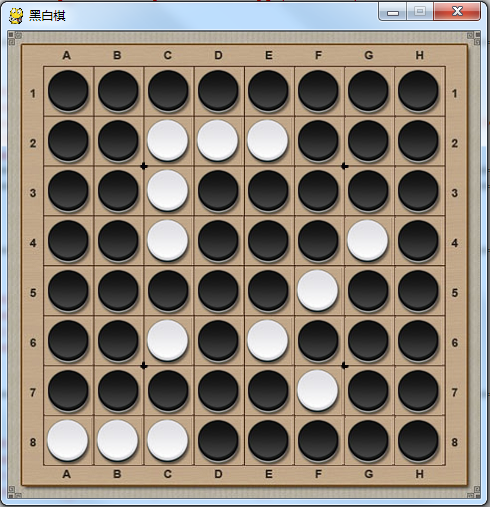
windowSurface.blit(boardImage, boardRect, boardRect)

4.游戏规则

1）是否允许落子

2）落子后的翻转

3）电脑AI走法



前几天不会玩时总被电脑虐，如今略微会玩了，图为我执黑，大胜电脑AI，看来这个AI算法并不咋地啊……

完整程序：

import pygame, sys, random

from pygame.locals import \*

BACKGROUNDCOLOR = (255, 255, 255)

BLACK = (255, 255, 255)

BLUE = (0, 0, 255)

CELLWIDTH = 50

CELLHEIGHT = 50

PIECEWIDTH = 47

PIECEHEIGHT = 47

BOARDX = 35

BOARDY = 35

FPS = 40

# 退出

def terminate():

pygame.quit()

sys.exit()

# 重置棋盘

def resetBoard(board):

for x in range(8):

for y in range(8):

board[x][y] = 'none'

# Starting pieces:

board[3][3] = 'black'

board[3][4] = 'white'

board[4][3] = 'white'

board[4][4] = 'black'

# 开局时建立新棋盘

def getNewBoard():

board = []

for i in range(8):

board.append(['none'] \* 8)

return board

# 是否是合法走法

def isValidMove(board, tile, xstart, ystart):

# 如果该位置已经有棋子或者出界了，返回False

if not isOnBoard(xstart, ystart) or board[xstart][ystart] != 'none':

return False

# 临时将tile 放到指定的位置

board[xstart][ystart] = tile

if tile == 'black':

otherTile = 'white'

else:

otherTile = 'black'

# 要被翻转的棋子

tilesToFlip = []

for xdirection, ydirection in [ [0, 1], [1, 1], [1, 0], [1, -1], [0, -1], [-1, -1], [-1, 0], [-1, 1] ]:

x, y = xstart, ystart

x += xdirection

y += ydirection

if isOnBoard(x, y) and board[x][y] == otherTile:

x += xdirection

y += ydirection

if not isOnBoard(x, y):

continue

# 一直走到出界或不是对方棋子的位置

while board[x][y] == otherTile:

x += xdirection

y += ydirection

if not isOnBoard(x, y):

break

# 出界了，则没有棋子要翻转OXXXXX

if not isOnBoard(x, y):

continue

# 是自己的棋子OXXXXXXO

if board[x][y] == tile:

while True:

x -= xdirection

y -= ydirection

# 回到了起点则结束

if x == xstart and y == ystart:

break

# 需要翻转的棋子

tilesToFlip.append([x, y])

# 将前面临时放上的棋子去掉，即还原棋盘

board[xstart][ystart] = 'none' # restore the empty space

# 没有要被翻转的棋子，则走法非法。翻转棋的规则。

if len(tilesToFlip) == 0: # If no tiles were flipped, this is not a valid move.

return False

return tilesToFlip

# 是否出界

def isOnBoard(x, y):

return x >= 0 and x <= 7 and y >= 0 and y <=7

# 获取可落子的位置

def getValidMoves(board, tile):

validMoves = []

for x in range(8):

for y in range(8):

if isValidMove(board, tile, x, y) != False:

validMoves.append([x, y])

return validMoves

# 获取棋盘上黑白双方的棋子数

def getScoreOfBoard(board):

xscore = 0

oscore = 0

for x in range(8):

for y in range(8):

if board[x][y] == 'black':

xscore += 1

if board[x][y] == 'white':

oscore += 1

return {'black':xscore, 'white':oscore}

# 谁先走

def whoGoesFirst():

if random.randint(0, 1) == 0:

return 'computer'

else:

return 'player'

# 将一个tile棋子放到(xstart, ystart)

def makeMove(board, tile, xstart, ystart):

tilesToFlip = isValidMove(board, tile, xstart, ystart)

if tilesToFlip == False:

return False

board[xstart][ystart] = tile

for x, y in tilesToFlip:

board[x][y] = tile

return True

# 复制棋盘

def getBoardCopy(board):

dupeBoard = getNewBoard()

for x in range(8):

for y in range(8):

dupeBoard[x][y] = board[x][y]

return dupeBoard

# 是否在角上

def isOnCorner(x, y):

return (x == 0 and y == 0) or (x == 7 and y == 0) or (x == 0 and y == 7) or (x == 7 and y == 7)

# 电脑走法，AI

def getComputerMove(board, computerTile):

# 获取所以合法走法

possibleMoves = getValidMoves(board, computerTile)

# 打乱所有合法走法

random.shuffle(possibleMoves)

# [x, y]在角上，则优先走，因为角上的不会被再次翻转

for x, y in possibleMoves:

if isOnCorner(x, y):

return [x, y]

bestScore = -1

for x, y in possibleMoves:

dupeBoard = getBoardCopy(board)

makeMove(dupeBoard, computerTile, x, y)

# 按照分数选择走法，优先选择翻转后分数最多的走法

score = getScoreOfBoard(dupeBoard)[computerTile]

if score > bestScore:

bestMove = [x, y]

bestScore = score

return bestMove

# 是否游戏结束

def isGameOver(board):

for x in range(8):

for y in range(8):

if board[x][y] == 'none':

return False

return True

# 初始化

pygame.init()

mainClock = pygame.time.Clock()

# 加载图片

boardImage = pygame.image.load('board.png')

boardRect = boardImage.get\_rect()

blackImage = pygame.image.load('black.png')

blackRect = blackImage.get\_rect()

whiteImage = pygame.image.load('white.png')

whiteRect = whiteImage.get\_rect()

basicFont = pygame.font.SysFont(None, 48)

gameoverStr = 'Game Over Score '

mainBoard = getNewBoard()

resetBoard(mainBoard)

turn = whoGoesFirst()

if turn == 'player':

playerTile = 'black'

computerTile = 'white'

else:

playerTile = 'white'

computerTile = 'black'

print(turn)

# 设置窗口

windowSurface = pygame.display.set\_mode((boardRect.width, boardRect.height))

pygame.display.set\_caption('黑白棋')

gameOver = False

# 游戏主循环

while True:

for event in pygame.event.get():

if event.type == QUIT:

terminate()

if gameOver == False and turn == 'player' and event.type == MOUSEBUTTONDOWN and event.button == 1:

x, y = pygame.mouse.get\_pos()

col = int((x-BOARDX)/CELLWIDTH)

row = int((y-BOARDY)/CELLHEIGHT)

if makeMove(mainBoard, playerTile, col, row) == True:

if getValidMoves(mainBoard, computerTile) != []:

turn = 'computer'

windowSurface.fill(BACKGROUNDCOLOR)

windowSurface.blit(boardImage, boardRect, boardRect)

if (gameOver == False and turn == 'computer'):

x, y = getComputerMove(mainBoard, computerTile)

makeMove(mainBoard, computerTile, x, y)

savex, savey = x, y

# 玩家没有可行的走法了

if getValidMoves(mainBoard, playerTile) != []:

turn = 'player'

windowSurface.fill(BACKGROUNDCOLOR)

windowSurface.blit(boardImage, boardRect, boardRect)

for x in range(8):

for y in range(8):

rectDst = pygame.Rect(BOARDX+x\*CELLWIDTH+2, BOARDY+y\*CELLHEIGHT+2, PIECEWIDTH, PIECEHEIGHT)

if mainBoard[x][y] == 'black':

windowSurface.blit(blackImage, rectDst, blackRect)

elif mainBoard[x][y] == 'white':

windowSurface.blit(whiteImage, rectDst, whiteRect)

if isGameOver(mainBoard):

scorePlayer = getScoreOfBoard(mainBoard)[playerTile]

scoreComputer = getScoreOfBoard(mainBoard)[computerTile]

outputStr = gameoverStr + str(scorePlayer) + ":" + str(scoreComputer)

text = basicFont.render(outputStr, True, BLACK, BLUE)

textRect = text.get\_rect()

textRect.centerx = windowSurface.get\_rect().centerx

textRect.centery = windowSurface.get\_rect().centery

windowSurface.blit(text, textRect)

pygame.display.update()

mainClock.tick(FPS)

