#=====================================================================

# Gomoku

#=====================================================================

import **pygame** # class 0, **c0**

from Chessboard import **Chessboard** # class 2, **c2**

class **Gomoku**():# class 1, **c1**

def \_\_init\_\_(self):

**pygame.init**() #c0

self.**screen** = **pygame.display**.set\_mode((800, 600)) # **c0**

pygame.display.set\_caption("五子棋")

self.clock = pygame.time.Clock()

self.font=pygame.font.Font(r"C:\Windows\Fonts\consola.ttf",24)

self.going = True

self.**chessboard** = Chessboard() # **c2**

def loop(self): # **looping, not idle**

while self.going:

self.**update**()

self.**draw**()

self.**clock.tick**(60)

pygame.quit()

def **update**(self):

for e in **pygame.event**.get():

if e.type == pygame.QUIT:

self.going = False

elif e.type == pygame.**MOUSEBUTTONDOWN**:

self.**chessboard.handle\_key\_event**(e) # **c2**

def **draw**(self):

self.screen.fill((255, 255, 255))

self.screen.blit(self.font.render("FPS: {0:.2F}".format(self.clock.get\_fps()), True, (0, 0, 0)), (10, 10))

self.**chessboard.draw**(self.**screen**)

if self.chessboard.game\_over:

self.screen.blit(self.font.render("{0} Win".format("Black" if self.chessboard.winner=='b' else "White"), True, (0, 0, 0)), (500, 10))

**pygame.display**.update()

#=====================================================================

#=====================================================================

# Chessboard

#=====================================================================

import **pygame**# **c0**

class **Chessboard**: # **c2**

def **\_\_init\_\_**(self):

self.grid\_size = 26

self.start\_x, self.start\_y = 30, 50

self.edge\_size = self.grid\_size / 2

self.grid\_count = 19

self.piece = 'b'

self.winner = None

self.game\_over = False

self.grid = []

for i in range(self.grid\_count):

self.grid.append(list("." \* self.grid\_count))

def **handle\_key\_event**(self, e):

origin\_x = self.start\_x - self.edge\_size

origin\_y = self.start\_y - self.edge\_size

size = (self.grid\_count - 1) \* self.grid\_size + self.edge\_size\*2

**pos** = e.pos

if origin\_x<=pos[0]<=origin\_x+size and origin\_y<=pos[1]<=origin\_y+size:

if not self.game\_over:

x = pos[0] - origin\_x

y = pos[1] - origin\_y

r = int(y // self.grid\_size)

c = int(x // self.grid\_size)

if **self.set\_piece(r, c)**: # 不好的作法，焦點模糊

self.check\_win(r, c)

def set\_piece(self, r, c):

if **self.grid[r][c] == '.'**:

self.grid[r][c] = self.piece

if self.piece == 'b':

self.piece = 'w'

else:

self.piece = 'b'

return True

return False

def **check\_win**(self, r, c):

**n**\_count = self.get\_continuous\_count(r, c, **-1, 0**)

**s**\_count = self.get\_continuous\_count(r, c, **1, 0**)

**e**\_count = self.get\_continuous\_count(r, c, **0, 1**)

**w**\_count = self.get\_continuous\_count(r, c, **0, -1**)

**se**\_count = self.get\_continuous\_count(r, c, **1, 1**)

**nw**\_count = self.get\_continuous\_count(r, c, **-1, -1**)

**ne**\_count = self.get\_continuous\_count(r, c, **-1, 1**)

**sw**\_count = self.get\_continuous\_count(r, c, **1, -1**)

if (**n\_count+s\_count**+1>=5) or (e\_count+w\_count+1>=5) or **\**

(se\_count+nw\_count+1>=5) or (ne\_count+sw\_count+1>=5):

self.winner = self.grid[r][c]

self.game\_over = True

def get\_continuous\_count(self, r, c, **dr**, **dc**):

piece = self.grid[r][c]

result = 0

i = 1

while True:

new\_r = r + **dr \* i**

new\_c = c + **dc \* i**

if 0 <= new\_r < self.grid\_count and \

0 <= new\_c < self.grid\_count:

if self.grid[new\_r][new\_c] == piece:

result += 1

else:

break

else:

break

**i += 1**

return result

def **draw**(self, **screen**):

# 棋盤底色

**pygame.draw**.rect(**screen**, (185, 122, 87),

[

self.start\_x - self.edge\_size,

self.start\_y - self.edge\_size,

(self.grid\_count - 1) \* self.grid\_size + self.edge\_size \* 2,

(self.grid\_count - 1) \* self.grid\_size + self.edge\_size \* 2],

0)

for r in range(self.**grid\_count**):

y = self.start\_y + r \* self.grid\_size

pygame.draw.**line**(screen, (0, 0, 0),

[self.start\_x, y],

[self.start\_x + self.grid\_size \* (self.grid\_count - 1),

y], 2)

for c in range(self.**grid\_count**):

x = self.start\_x + c \* self.grid\_size

pygame.draw.**line**(screen, (0, 0, 0),

[x, self.start\_y],

[x, self.start\_y+self.grid\_size\*(self.grid\_count-1)],

2)

for r in range(self.**grid\_count**):

for c in range(self.grid\_count):

piece = self.**grid[r][c]**

if piece != '.':

if piece == 'b':

**color** = (0, 0, 0)

else:

**color** = (255, 255, 255)

x = self.start\_x + c \* self.grid\_size

y = self.start\_y + r \* self.grid\_size

pygame.draw.**circle**(screen, color, [x, y],

self.grid\_size // 2) # smaller

#=====================================================================

#=====================================================================

# game = **Gomoku**()

# **game.loop**()

#=====================================================================