오픈소스SW 과제중심수업 보고서

중국학과 전공
2018049234 김민호

GitHub repository 주소 : https://github.com/mindol0113/osw-repository

1. 각 함수들의 역할

[메인 설정 코드]

```
# Tetromino (a Tetris clone)
# By Al Sweigart al@inventwithpython.com
# http://inventwithpython.com/pygame
# Released under a "Simplified BSD" license

import random, time, pygame, sys
from pygame.locals import *

FPS = 25
WINDOWWIDTH = 640
WINDOWHEIGHT = 480
BOXSIZE = 20
BOARDWIDTH = 10
BOARDHEIGHT = 20
BLANK = '.'
```

[키를 누르고 있는 동안 시간 상수 설정]

```
MOVESIDEWAYSFREQ = 0.15
MOVEDOWNFREQ = 0.1
```

[셋업 코드]

```
XMARGIN = int((WINDOWWIDTH - BOARDWIDTH * BOXSIZE) / 2)
TOPMARGIN = WINDOWHEIGHT - (BOARDHEIGHT * BOXSIZE) - 5

#                R    G    B
WHITE       = (255, 255, 255)
```

```
GRAY        = (185, 185, 185)
BLACK       = (  0,   0,   0)
RED         = (155,   0,   0)
LIGHTRED    = (175,  20,  20)
GREEN       = (  0, 155,   0)
LIGHTGREEN  = ( 20, 175,  20)
BLUE        = (  0,   0, 155)
LIGHTBLUE   = ( 20,  20, 175)
YELLOW      = (155, 155,   0)
LIGHTYELLOW = (175, 175,  20)

BORDERCOLOR = BLUE
BGCOLOR = BLACK
TEXTCOLOR = WHITE
TEXTSHADOWCOLOR = GRAY
COLORS      = (      BLUE,      GREEN,      RED,      YELLOW)
LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYELLOW)
assert len(COLORS) == len(LIGHTCOLORS) # each color must have light color
```

[조각 템플릿 설정]

```
TEMPLATEWIDTH = 5
TEMPLATEHEIGHT = 5

S_SHAPE_TEMPLATE = [['.....',
                     '.....',
                     '..OO.',
                     '.OO..',
                     '.....'],
                    ['.....',
                     '..O..',
                     '..OO.',
                     '...O.',
                     '.....']]

Z_SHAPE_TEMPLATE = [['.....',
                     '.....',
                     '.OO..',
                     '..OO.',
                     '.....'],
                    ['.....',
```

```
                       '..O..',
                       '.OO..',
                       '.O...',
                       '.....']]

I_SHAPE_TEMPLATE = [['..O..',
                       '..O..',
                       '..O..',
                       '..O..',
                       '.....'],
                      ['.....',
                       '.....',
                       'OOOO.',
                       '.....',
                       '.....']]

O_SHAPE_TEMPLATE = [['.....',
                       '.....',
                       '.OO..',
                       '.OO..',
                       '.....']]

J_SHAPE_TEMPLATE = [['.....',
                       '.O...',
                       '.OOO.',
                       '.....',
                       '.....'],
                      ['.....',
                       '..OO.',
                       '..O..',
                       '..O..',
                       '.....'],
                      ['.....',
                       '.....',
                       '.OOO.',
                       '...O.',
                       '.....'],
                      ['.....',
                       '..O..',
                       '..O..',
                       '.OO..',
```

```
                      '.....']]

L_SHAPE_TEMPLATE = [['.....',
                      '...O.',
                      '.OOO.',
                      '.....',
                      '.....'],
                     ['.....',
                      '..O..',
                      '..O..',
                      '..OO.',
                      '.....'],
                     ['.....',
                      '.....',
                      '.OOO.',
                      '.O...',
                      '.....'],
                     ['.....',
                      '.OO..',
                      '..O..',
                      '..O..',
                      '.....']]

T_SHAPE_TEMPLATE = [['.....',
                      '..O..',
                      '.OOO.',
                      '.....',
                      '.....'],
                     ['.....',
                      '..O..',
                      '..OO.',
                      '..O..',
                      '.....'],
                     ['.....',
                      '.....',
                      '.OOO.',
                      '..O..',
                      '.....'],
                     ['.....',
                      '..O..',
                      '.OO..',
```

```
                      '..O..',
                      '.....']]

PIECES = {'S': S_SHAPE_TEMPLATE,
          'Z': Z_SHAPE_TEMPLATE,
          'J': J_SHAPE_TEMPLATE,
          'L': L_SHAPE_TEMPLATE,
          'I': I_SHAPE_TEMPLATE,
          'O': O_SHAPE_TEMPLATE,
          'T': T_SHAPE_TEMPLATE}
```

[메인 함수 - 주 기능()에서는 더 많은 글로벌 상수를 생성하고 프로그램을 실행할 때 나타나는 시작 화면을 표시하는 작업을 처리합니다.]

```
def main():
    global FPSCLOCK, DISPLAYSURF, BASICFONT, BIGFONT
    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
    BASICFONT = pygame.font.Font('freesansbold.ttf', 18)
    BIGFONT = pygame.font.Font('freesansbold.ttf', 100)
    pygame.display.set_caption('Tetromino')

    showTextScreen('Tetromino')
    while True: # 메인 게임 루프
        if random.randint(0, 1) == 0:
            pygame.mixer.music.load('tetrisb.mid')
        else:
            pygame.mixer.music.load('tetrisc.mid')
        pygame.mixer.music.play(-1, 0.0)
        runGame()
        pygame.mixer.music.stop()
        showTextScreen('Game Over')
```

[실제 게임의 코드는 모두 runGame()이다. 여기서 메인() 기능은 어떤 배경음악을 재생할지를 무작위로 결정(tetrisb.mid 또는 tetrisc.mid MIDI 음악 파일)하고 runGame()을 호출하여 게임을 시작한다. 플레이어가 지면 게임()이 메인()으로 돌아가 백그라운드 음악을 중지하고 게임을 화면에 표시한다.

플레이어가 키를 누르면 게임을 화면에 표시하는 ShowTextScreen() 기능이 반환된다. 게임 루프는 169라인에서 다시 시작점으로 돌아가 또 다른 게임을 시작할 것이다.]

```
def runGame():
    # 게임 시작에 대한 설정 변수
    board = getBlankBoard()
    lastMoveDownTime = time.time()
    lastMoveSidewaysTime = time.time()
    lastFallTime = time.time()
    movingDown = False # note: there is no movingUp variable
    movingLeft = False
    movingRight = False
    score = 0
    level, fallFreq = calculateLevelAndFallFreq(score)

    fallingPiece = getNewPiece()
    nextPiece = getNewPiece()

    while True: # 메인 게임 루프이다.
        if fallingPiece == None:
            # 더 이상 조각이 없어 새로운 게임을 시작한다.
            fallingPiece = nextPiece
            nextPiece = getNewPiece()
            lastFallTime = time.time() # lastFallTime을 리셋한다.

            if not isValidPosition(board, fallingPiece):
                return # 새 조각을 맞출 수 없어 새로운 게임을 시작한다.

        checkForQuit()
        for event in pygame.event.get(): # event handling loop
            if event.type == KEYUP:
                if (event.key == K_p):
                    # 게임 일시정지
                    DISPLAYSURF.fill(BGCOLOR)
                    pygame.mixer.music.stop()
                    showTextScreen('Paused') # 키보드 키가 눌러질 때까지 일시정지
                    pygame.mixer.music.play(-1, 0.0)
                    lastFallTime = time.time()
                    lastMoveDownTime = time.time()
                    lastMoveSidewaysTime = time.time()
                elif (event.key == K_LEFT or event.key == K_a):
                    movingLeft = False
                elif (event.key == K_RIGHT or event.key == K_d):
```

```
                movingRight = False
            elif (event.key == K_DOWN or event.key == K_s):
                movingDown = False

        elif event.type == KEYDOWN:
            # 조각을 사이드로 이동시킨다.
            if (event.key == K_LEFT or event.key == K_a) and
isValidPosition(board, fallingPiece, adjX=-1):
                fallingPiece['x'] -= 1
                movingLeft = True
                movingRight = False
                lastMoveSidewaysTime = time.time()

            elif (event.key == K_RIGHT or event.key == K_d) and
isValidPosition(board, fallingPiece, adjX=1):
                fallingPiece['x'] += 1
                movingRight = True
                movingLeft = False
                lastMoveSidewaysTime = time.time()

            # 조각을 변경한다.
            elif (event.key == K_UP or event.key == K_w):
                fallingPiece['rotation'] = (fallingPiece['rotation'] + 1) %
len(PIECES[fallingPiece['shape']])
                if not isValidPosition(board, fallingPiece):
                    fallingPiece['rotation'] = (fallingPiece['rotation'] - 1) %
len(PIECES[fallingPiece['shape']])
            elif (event.key == K_q): # rotate the other direction
                fallingPiece['rotation'] = (fallingPiece['rotation'] - 1) %
len(PIECES[fallingPiece['shape']])
                if not isValidPosition(board, fallingPiece):
                    fallingPiece['rotation'] = (fallingPiece['rotation'] + 1) %
len(PIECES[fallingPiece['shape']])

            # 다운 키를 누를 경우 조각이 더욱 빨리 떨어지도록 한다.
            elif (event.key == K_DOWN or event.key == K_s):
                movingDown = True
                if isValidPosition(board, fallingPiece, adjY=1):
                    fallingPiece['y'] += 1
                lastMoveDownTime = time.time()
```

```
                # 현재의 조각을 완전히 내린다.
                elif event.key == K_SPACE:
                    movingDown = False
                    movingLeft = False
                    movingRight = False
                    for i in range(1, BOARDHEIGHT):
                        if not isValidPosition(board, fallingPiece, adjY=i):
                            break
                    fallingPiece['y'] += i - 1

        # 유저의 입력에 따라 조각의 이동을 제어한다.
        if (movingLeft or movingRight) and time.time() - lastMoveSidewaysTime > MOVESIDEWAYSFREQ:
            if movingLeft and isValidPosition(board, fallingPiece, adjX=-1):
                fallingPiece['x'] -= 1
            elif movingRight and isValidPosition(board, fallingPiece, adjX=1):
                fallingPiece['x'] += 1
            lastMoveSidewaysTime = time.time()

        if movingDown and time.time() - lastMoveDownTime > MOVEDOWNFREQ and isValidPosition(board, fallingPiece, adjY=1):
            fallingPiece['y'] += 1
            lastMoveDownTime = time.time()

        # 정해진 시간에 맞춰 조각을 떨어뜨린다.
        if time.time() - lastFallTime > fallFreq:
            # 조각이 떨어졌을 경우의 상황을 보여준다.
            if not isValidPosition(board, fallingPiece, adjY=1):
                # 떨어지는 조각을 보드에 맞춘다.
                addToBoard(board, fallingPiece)
                score += removeCompleteLines(board)
                level, fallFreq = calculateLevelAndFallFreq(score)
                fallingPiece = None
            else:
                # 조각이 제대로 합체되지 않을 경우 그대로 보드 위에 올린다.
                fallingPiece['y'] += 1
                lastFallTime = time.time()

        # 디스플레이 코드
        DISPLAYSURF.fill(BGCOLOR)
        drawBoard(board)
```

```
        drawStatus(score, level)
        drawNextPiece(nextPiece)
        if fallingPiece != None:
            drawPiece(fallingPiece)

        pygame.display.update()
        FPSCLOCK.tick(FPS)


def makeTextObjs(text, font, color):
    surf = font.render(text, True, color)
    return surf, surf.get_rect()


def terminate():
    pygame.quit()
    sys.exit()


def checkForKeyPress():
    # 이벤트 대기열에서 KEYUP 이벤트를 찾도록 한다.
    # KEYDOWN 이벤트를 캡처하여 이벤트 대기열에서 제거한다.
    checkForQuit()

    for event in pygame.event.get([KEYDOWN, KEYUP]):
        if event.type == KEYDOWN:
            continue
        return event.key
    return None


def showTextScreen(text):
    # 이 기능은 키를 누를 때까지 화면 중앙에 큰 텍스트를 표시한다.
    # 텍스트 드롭 섀도를 그린다.
    titleSurf, titleRect = makeTextObjs(text, BIGFONT, TEXTSHADOWCOLOR)
    titleRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2))
    DISPLAYSURF.blit(titleSurf, titleRect)

    # 텍스트 표현하기
    titleSurf, titleRect = makeTextObjs(text, BIGFONT, TEXTCOLOR)
    titleRect.center = (int(WINDOWWIDTH / 2) - 3, int(WINDOWHEIGHT / 2) - 3)
```

```
    DISPLAYSURF.blit(titleSurf, titleRect)

    # 추가적인 "Press a key to play." 텍스트를 구현한다.
    pressKeySurf, pressKeyRect = makeTextObjs('Press a key to play.', BASICFONT,
TEXTCOLOR)
    pressKeyRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2) + 100)
    DISPLAYSURF.blit(pressKeySurf, pressKeyRect)

    while checkForKeyPress() == None:
        pygame.display.update()
        FPSCLOCK.tick()


def checkForQuit():
    for event in pygame.event.get(QUIT): # 모든 QUIT events 입력
        terminate() # QUIT 이벤트의 경우 전환한다.
    for event in pygame.event.get(KEYUP): # 모든 KEYUP events 입력
        if event.key == K_ESCAPE:
            terminate() # KEYUP event가 ESC 키에 해당하도록 전환한다.
        pygame.event.post(event) # 다른 KEYUP event objects를 제공한다.


def calculateLevelAndFallFreq(score):
    # 점수를 기준으로 플레이어가 있는 레벨과 낙하물이 한 칸 떨어질 때까지 경과하는 시
      간을 반환한다.
    level = int(score / 10) + 1
    fallFreq = 0.27 - (level * 0.02)
    return level, fallFreq

def getNewPiece():
    # 무작위로 새로운 색깔의 조각을 제공한다.
    shape = random.choice(list(PIECES.keys()))
    newPiece = {'shape': shape,
                'rotation': random.randint(0, len(PIECES[shape]) - 1),
                'x': int(BOARDWIDTH / 2) - int(TEMPLATEWIDTH / 2),
                'y': -2, # 보드 위에서 시작하도록 한다. (i.e. less than 0)
                'color': random.randint(0, len(COLORS)-1)}
    return newPiece


def addToBoard(board, piece):
```

```
    # 피스의 위치, 모양, 회전 등에 따라 보드를 채운다.
    for x in range(TEMPLATEWIDTH):
        for y in range(TEMPLATEHEIGHT):
            if PIECES[piece['shape']][piece['rotation']][y][x] != BLANK:
                board[x + piece['x']][y + piece['y']] = piece['color']


def getBlankBoard():
    # 새 빈 보드 데이터 구조를 만들고 반환한다.
    board = []
    for i in range(BOARDWIDTH):
        board.append([BLANK] * BOARDHEIGHT)
    return board


def isOnBoard(x, y):
    return x >= 0 and x < BOARDWIDTH and y < BOARDHEIGHT


def isValidPosition(board, piece, adjX=0, adjY=0):
    # 부품이 보드 내에 있고 충돌하지 않는 경우 True를 반환한다.
    for x in range(TEMPLATEWIDTH):
        for y in range(TEMPLATEHEIGHT):
            isAboveBoard = y + piece['y'] + adjY < 0
            if isAboveBoard or PIECES[piece['shape']][piece['rotation']][y][x] ==
BLANK:
                continue
            if not isOnBoard(x + piece['x'] + adjX, y + piece['y'] + adjY):
                return False
            if board[x + piece['x'] + adjX][y + piece['y'] + adjY] != BLANK:
                return False
    return True

def isCompleteLine(board, y):
    # 줄이 공백이 없는 상자로 채워진 경우 True를 반환한다.
    for x in range(BOARDWIDTH):
        if board[x][y] == BLANK:
            return False
    return True
```

```
def removeCompleteLines(board):
    # 보드에서 완료된 라인을 제거하고 그 위에 있는 모든 라인을 아래로 이동한 다음 전체 라인 수를 반환한다.
    numLinesRemoved = 0
    y = BOARDHEIGHT - 1 # start y at the bottom of the board
    while y >= 0:
        if isCompleteLine(board, y):
            # 선을 제거하고 상자를 한 줄 아래로 당긴다.
            for pullDownY in range(y, 0, -1):
                for x in range(BOARDWIDTH):
                    board[x][pullDownY] = board[x][pullDownY-1]
            # 빈 곳에 탑 라인을 설정한다.
            for x in range(BOARDWIDTH):
                board[x][0] = BLANK
            numLinesRemoved += 1
            # 루프의 다음 반복에 대한 참고 사항으로, y는 동일하다.
            # 아래로 당겨진 라인도 완성되면 제거되도록 하기 위함이다.
        else:
            y -= 1 # 다음 단계의 체크
    return numLinesRemoved


def convertToPixelCoords(boxx, boxy):
    # 보드의 지정된 xy 좌표를 xy로 변환한다.
    # 화면 상의 위치 좌표
    return (XMARGIN + (boxx * BOXSIZE)), (TOPMARGIN + (boxy * BOXSIZE))


def drawBox(boxx, boxy, color, pixelx=None, pixely=None):
    # 싱글 박스 생성
    # 보드의 xy 좌표 또는 픽셀 x & 픽셀이 지정된 경우 픽셀 x & 픽셀에 저장된 픽셀 좌표로 그린다(이것은 "Next" 피스에 사용된다).
    if color == BLANK:
        return
    if pixelx == None and pixely == None:
        pixelx, pixely = convertToPixelCoords(boxx, boxy)
    pygame.draw.rect(DISPLAYSURF, COLORS[color], (pixelx + 1, pixely + 1, BOXSIZE - 1, BOXSIZE - 1))
    pygame.draw.rect(DISPLAYSURF, LIGHTCOLORS[color], (pixelx + 1, pixely + 1, BOXSIZE - 4, BOXSIZE - 4))
```

```
def drawBoard(board):
    # 테두리를 그린다.
    pygame.draw.rect(DISPLAYSURF, BORDERCOLOR, (XMARGIN - 3, TOPMARGIN - 7, (BOARDWIDTH * BOXSIZE) + 8, (BOARDHEIGHT * BOXSIZE) + 8), 5)

    # 배경 채우기
    pygame.draw.rect(DISPLAYSURF, BGCOLOR, (XMARGIN, TOPMARGIN, BOXSIZE * BOARDWIDTH, BOXSIZE * BOARDHEIGHT))
    # 각각 박스 생성
    for x in range(BOARDWIDTH):
        for y in range(BOARDHEIGHT):
            drawBox(x, y, board[x][y])


def drawStatus(score, level):
    # 점수 텍스트 생성
    scoreSurf = BASICFONT.render('Score: %s' % score, True, TEXTCOLOR)
    scoreRect = scoreSurf.get_rect()
    scoreRect.topleft = (WINDOWWIDTH - 150, 20)
    DISPLAYSURF.blit(scoreSurf, scoreRect)

    # 레벨 텍스트 생성
    levelSurf = BASICFONT.render('Level: %s' % level, True, TEXTCOLOR)
    levelRect = levelSurf.get_rect()
    levelRect.topleft = (WINDOWWIDTH - 150, 50)
    DISPLAYSURF.blit(levelSurf, levelRect)


def drawPiece(piece, pixelx=None, pixely=None):
    shapeToDraw = PIECES[piece['shape']][piece['rotation']]
    if pixelx == None and pixely == None:
        # 픽셀 x & 픽셀리가 지정되지 않은 경우 조각 데이터 구조에 저장된 위치를 사용한다.
        pixelx, pixely = convertToPixelCoords(piece['x'], piece['y'])

    # 조각을 만드는 각각의 박스들을 생성
    for x in range(TEMPLATEWIDTH):
        for y in range(TEMPLATEHEIGHT):
            if shapeToDraw[y][x] != BLANK:
                drawBox(None, None, piece['color'], pixelx + (x * BOXSIZE), pixely +
```

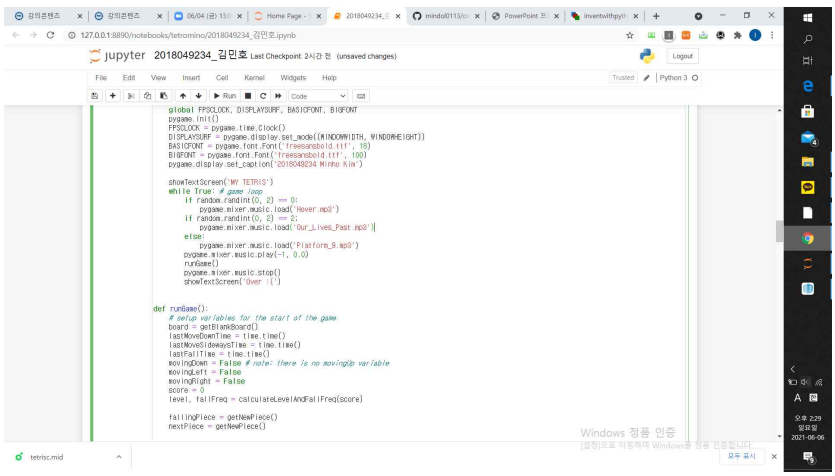
(y * BOXSIZE))

```
def drawNextPiece(piece):
    # "next" 텍스트 생성
    nextSurf = BASICFONT.render('Next:', True, TEXTCOLOR)
    nextRect = nextSurf.get_rect()
    nextRect.topleft = (WINDOWWIDTH - 120, 80)
    DISPLAYSURF.blit(nextSurf, nextRect)


    # next" 조각 생성
    drawPiece(piece, pixelx=WINDOWWIDTH-120, pixely=100)


if __name__ == '__main__':
    main()
```

2. 함수의 호출 순서 또는 호출 조건



수정사항들입니다. 하지만 실행이 잘 되지 않습니다. 문법 오류가 발생합니다.