오픈소스SW 과제중심수업 보고서

ICT융합학부 미디어테크 전공
2020094802 김인석

GitHub repository 주소 : https://github.com/ingssg/osw_repo

1. 각 함수들의 역할

```
1. # Tetromino (a Tetris clone)
2. # By Al Sweigart al@inventwithpython.com
3. # http://inventwithpython.com/pygame
4. # Creative Commons BY-NC-SA 3.0 US
5.
6. import random, time, pygame, sys
7. from pygame.locals import *
8.
9. FPS = 25
10. WINDOWWIDTH = 640
11. WINDOWHEIGHT = 480
12. BOXSIZE = 20
13. BOARDWIDTH = 10
14. BOARDHEIGHT = 20
15. BLANK = '.'
```

ㄴ테트로미노 게임에 사용되는 상수들이다. 각 박스는 너비, 높이, 폭을 정의하고 있다.

```
17. MOVESIDEWAYSFREQ = 0.15
18. MOVEDOWNFREQ = 0.1
```

ㄴ플레이어가 방향키를 누른 상태에서의 피스 움직임의 시간이다.

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

ㄴ여백을 계산하는 상수이다.

```
#             R    G    B
WHITE       = (255, 255, 255)
GRAY        = (185, 185, 185)
BLACK       = (  0,   0,   0)
PINK        = (255,   0 , 221)
LIGHTPINK   = (255, 217, 250)
PURPLE      = (128,  65, 217)
LIGPURPLE   = (209, 178, 255)
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,  GRAY,      PINK,    PURPLE)
LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYELLOW, WHITE, LIGHTPINK, LIGPURPLE)
assert len(COLORS) == len(LIGHTCOLORS) # each color must have light color

TEMPLATEWIDTH = 5
TEMPLATEHEIGHT = 5

S_SHAPE_TEMPLATE = [['.....',
                     '.....',
                     '..00.',
                     '.00..',
                     '.....'],
                    ['.....',
                     '..0..',
                     '..00.',
                     '...0.',
                     '.....']]
```

ㄴ색상 정보와 블록의 모양새를 정의해준다.

```
                     '.....',
                     '.000.',
                     '..0..',
                     '.....'],
                    ['.....',
                     '..0..',
                     '.00..',
                     '..0..',
                     '.....']]

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}
```

ㄴ블록의 모양을 딕셔너리로 정의해준다.

```
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('2020094802 KIMINSEOK')

    showTextScreen('MY TETRIS')
    while True: # game loop
        if random.randint(0, 2) == 0:
            pygame.mixer.music.load('Hover.mp3')
        elif random.randint(0, 2) == 1:
            pygame.mixer.music.load('Our_Lives_Past.mp3')
        else:
            pygame.mixer.music.load('Platform_9.mp3')
        pygame.mixer.music.play(-1, 0.0)
        runGame()
        pygame.mixer.music.stop()
        showTextScreen('Over :(')
```

ㄴmain함수이며, 글로벌 상수를 생성하고 프로그램이 실행되면 나타나는 시작화면을 표시하는 작업을 한다. 또한 어떤 배경음악을 재생할지를 랜덤으로 결정해준다.(Hover, Our_Lives_Past, Platform 중 1개) 결정이 되면 runGame()을 호출한 후 게임을 시작하고 플레이어가 지게 된다면 runGame()에서 다시 main()으로 돌아가 배경음악을 중지하고 게임 오버 문구를 스크린에 띄운다.

```
def runGame():
    # setup variables for the start of the game
    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()

    start_time = time.time()
```

ㄴrunGame() 함수이고 게임 시작 값에 대한 몇몇 변수를 초기화하는 과정이다.

```
while True: # game loop

    if fallingPiece == None:
        # No falling piece in play, so start a new piece at the top
        fallingPiece = nextPiece
        nextPiece = getNewPiece()
        lastFallTime = time.time() # reset lastFallTime

        if not isValidPosition(board, fallingPiece):
            return # can't fit a new piece on the board, so game over

    checkForQuit()
```

ㄴ떨어지고있는 피스가 없으면 다음 피스로 대체하고 lastFallTime을 리셋한다. 새 피스로 대체할 수 없으면 게임오버이다.

```
        for event in pygame.event.get(): # event handling loop
            if event.type == KEYUP:
                if (event.key == K_p):
                    # Pausing the game
                    DISPLAYSURF.fill(BGCOLOR)
                    pygame.mixer.music.stop()
                    showTextScreen('Get a rest!') # pause until a key press
                    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
```

ㄴ이벤트 처리 루프는 플레이어가 블록을 회전하거나 이동하거나 게임을 일시 중지하면 처리된다. 만약 플레이어가 p키를 누르면 게임이 일시 중지되고 음악을 중지한다. 또한 일시 중지 상태의 텍스트를 표시하고 플레이어가 키를 눌러 계속할 때까지 기다린다. 플레이어가 키를 누르면 배경음악을 다시 시작하고 시간 변수를 다시 현재 시간으로 재설정한다. 또한 화살표 키나 wasd 중 하나를 누르면 이동 변수가 False로 다시 설정된다.

```
elif event.type == KEYDOWN:
    # moving the piece sideways
    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()
```

ㄴ왼쪽 화살표 키나 a를 누른 상태에서 movingleft변수는 True로 설정되고 떨어지는 블록이 왼쪽과 오른쪽으로 모두 이동하지 않도록 하기 위해 MovingRight변수는 False로 설정되고, 마지막 lastMoveSidewaysTime이 현재 시간으로 업데이트 된다. 이러한 변수는 플레이어가 화살표 키를 누르고 있으면 계속 이동할 수 있도록 설정된다. elif 부분도 방향만 오른쪽이고 내용은 거의 유사하다.

```
        # rotating the piece (if there is room to rotate)
        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']])
```

ㄴ위쪽화살표 키나 w키는 다음 회전으로 회전시킨다. q는 다른 방향으로 회전시키게끔 한다.

```
# making the piece fall faster with the down key
elif (event.key == K_DOWN or event.key == K_s):
    movingDown = True
    if isValidPosition(board, fallingPiece, adjY=1):
        fallingPiece['y'] += 1
    lastMoveDownTime = time.time()
```

ㄴ아래쪽 화살표나 s를 누르면 한 칸 아래로 블록을 이동시키고 movingDow은 True로 설정되고 lastMoveDownTime은 현재 시간으로 재설정된다.

```
# move the current piece all the way down
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
```

ㄴ스페이스키를 누르면 movingDown, movingLeft, movingRight변수가 모드 False가 되며 착륙할 수 있는 범위까지 떨어진다.

```
# handle moving the piece because of user input
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()
```

ㄴ사용자가 키를 0.15초 이상 누른 경우 조건식은 참이 되고 그렇다면 사용자가 화살표키를

누르지 않았어도 왼쪽 또는 오른쪽으로 떨어지는 블록을 이동해야한다.

```
        # let the piece fall if it is time to fall
        if time.time() - lastFallTime > fallFreq:
            # see if the piece has landed
            if not isValidPosition(board, fallingPiece, adjY=1):
                # falling piece has landed, set it on the board
                addToBoard(board, fallingPiece)
                score += removeCompleteLines(board)
                level, fallFreq = calculateLevelAndFallFreq(score)
                fallingPiece = None
            else:
                # piece did not land, just move the piece down
                fallingPiece['y'] += 1
                lastFallTime = time.time()
```

ㄴ블록이 하강하는 속도는 lastFallTime변수에 의해 추적된다. 마지막으로 블록이 한 칸 아래로 떨어진 이후 충분한 시간이 경과하면 블록을 한 칸씩 떨어뜨릴 수 있다. 블록이 착륙했다면 점수판을 새로 설정한다. 아니라면 y 위치를 한 칸 아래로 설정하고 lastFallTime을 현재 시간으로 재설정한다.

```
        # drawing everything on the screen
        DISPLAYSURF.fill(BGCOLOR)
        drawBoard(board)
        drawStatus(score, level)

        pt = time.time() - start_time
        drawTime(pt)

        drawNextPiece(nextPiece)
        if fallingPiece != None:
            drawPiece(fallingPiece)

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

ㄴ화면에 모든 것을 나타내 준다. 점수나 레벨, 경과 시간, 다음 피스의 모양등을 업데이트한다.

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

ㄴtext, font, color이 주어지면 render()를 호출하고 surf, surf.get_rect()를 반환한다.

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

ㄴ게임을 종료시킨다.

```
def checkForKeyPress():
    # Go through event queue looking for a KEYUP event.
    # Grab KEYDOWN events to remove them from the event queue.
    checkForQuit()

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

ㄴCheckForQuit()를 호출한 후 KEYUP 및 KEYDOWN 이벤트를 꺼낸다.

```
def showTextScreen(text):
    # This function displays large text in the
    # center of the screen until a key is pressed.
    # Draw the text drop shadow
    titleSurf, titleRect = makeTextObjs(text, BIGFONT, YELLOW)
    titleRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2))
    DISPLAYSURF.blit(titleSurf, titleRect)

    # Draw the text
    titleSurf, titleRect = makeTextObjs(text, BIGFONT, LIGHTYELLOW)
    titleRect.center = (int(WINDOWWIDTH / 2) - 3, int(WINDOWHEIGHT / 2) - 3)
    DISPLAYSURF.blit(titleSurf, titleRect)

    # Draw the additional "Press a key to play." text.
    pressKeySurf, pressKeyRect = makeTextObjs('Press any key to play! pause key is p', BASICFONT, YELLOW)
    pressKeyRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2) + 100)
    DISPLAYSURF.blit(pressKeySurf, pressKeyRect)

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

ㄴ텍스트 파라미터에 전달된 텍스트를 모두 그리는 함수이다. 또한 특정 문구를 추가로 표시한다. 텍스트의 폰트나 색상을 설정할 수 있다. 이는 시작화면, 게임오버화면, 일시 중지 화면에도 사용된다. 또한 사용자가 키를 누를 때까지 텍스트가 화면에 유지되도록 한다.

```
def checkForQuit():
    for event in pygame.event.get(QUIT): # get all the QUIT events
        terminate() # terminate if any QUIT events are present
    for event in pygame.event.get(KEYUP): # get all the KEYUP events
        if event.key == K_ESCAPE:
            terminate() # terminate if the KEYUP event was for the Esc key
        pygame.event.post(event) # put the other KEYUP event objects back
```

ㄴ프로그램이 종료되는 모든 이벤트를 처리할 수 있게한다.

```
def calculateLevelAndFallFreq(score):
    # Based on the score, return the level the player is on and
    # how many seconds pass until a falling piece falls one space.
    level = int(score / 10) + 1
    fallFreq = 0.27 - (level * 0.02)
    return level, fallFreq
```

ㄴ스코어를 받아 레벨을 결정한 후 반환한다. fallFreq는 블록의 하강 속도를 결정하여 난이도를 어렵게 한다.

```
def getNewPiece():
    # return a random new piece in a random rotation and color
    listofPIECES = list(PIECES.keys())
    shape = random.choice(listofPIECES)
    index = 0
    for i in range(0,7):
        if shape == listofPIECES[i]:
            index = i

    newPiece = {'shape': shape,
                'rotation': random.randint(0, len(PIECES[shape]) - 1),
                'x': int(BOARDWIDTH / 2) - int(TEMPLATEWIDTH / 2),
                'y': -2, # start it above the board (i.e. less than 0)
                'color': index % 7}
    return newPiece
```

ㄴPIECES들 중에서 랜덤으로 하나를 받은 후 위치와 색상을 설정한 새로운 블록을 반환한다.

```
def addToBoard(board, piece):
    # fill in the board based on piece's location, shape, and rotation
    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']
```

ㄴpiece를 받아와서 board에 추가해주는 역할을 한다.

```
def getBlankBoard():
    # create and return a new blank board data structure
    board = []
    for i in range(BOARDWIDTH):
        board.append([BLANK] * BOARDHEIGHT)
    return board
```

ㄴ새로운 비어있는 보드 데이터구조를 생성하고 반환한다.

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

ㄴ받은 x,y좌표가 유효한 값인지 확인하는 함수로 두 좌표 모두 0보다 작거나 BOARDWIDTH 및 BOARDHEIGHT 상수보다 크거나 같으면 True를 반환한다.

```
def isValidPosition(board, piece, adjX=0, adjY=0):
    # Return True if the piece is within the board and not colliding
    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
```

ㄴ 블록이 보드 내에 있고 충돌하지 않는 경우 True를 반환하는 기능이다.

```
def isCompleteLine(board, y):
    # Return True if the line filled with boxes with no gaps.
    for x in range(BOARDWIDTH):
        if board[x][y] == BLANK:
            return False
    return True
```

ㄴ빈 칸이 없는 블록이 채워진 경우 True를 반환한다.

```
def removeCompleteLines(board):
    # Remove any completed lines on the board, move everything above them down, and return the number of complete lines.
    numLinesRemoved = 0
    y = BOARDHEIGHT - 1 # start y at the bottom of the board
    while y >= 0:
        if isCompleteLine(board, y):
            # Remove the line and pull boxes down by one line.
            for pullDownY in range(y, 0, -1):
                for x in range(BOARDWIDTH):
                    board[x][pullDownY] = board[x][pullDownY-1]
            # Set very top line to blank.
            for x in range(BOARDWIDTH):
                board[x][0] = BLANK
            numLinesRemoved += 1
            # Note on the next iteration of the loop, y is the same.
            # This is so that if the line that was pulled down is also
            # complete, it will be removed.
        else:
            y -= 1 # move on to check next row up
    return numLinesRemoved
```

ㄴ위 함수는 전달된 보드 데이터 구조에서 모든 전체 선을 찾아 선을 제거한 이후 해당 선 위의 모든 상자를 한 행 아래로 이동하고 제거된 선의 수를 반환한다.

```
def convertToPixelCoords(boxx, boxy):
    # Convert the given xy coordinates of the board to xy
    # coordinates of the location on the screen.
    return (XMARGIN + (boxx * BOXSIZE)), (TOPMARGIN + (boxy * BOXSIZE))
```

ㄴ보드의 박스 좌표를 픽셀 좌표로 변환한다.

```
def drawBox(boxx, boxy, color, pixelx=None, pixely=None):
    # draw a single box (each tetromino piece has four boxes)
    # at xy coordinates on the board. Or, if pixelx & pixely
    # are specified, draw to the pixel coordinates stored in
    # pixelx & pixely (this is used for the "Next" piece).
    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))
```

ㄴ화면에 단일 상자를 그린다. 이 때 보드 좌표에 대한 값을 수신할 수 있고 만약 설정하지 않으면 기본적으로 없음으로 설정된다. 또한 박스 전체를 색으로 채울 수 없기에 조금 더 작은 박스가 위에 그려진다.

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

    # fill the background of the board
    pygame.draw.rect(DISPLAYSURF, BGCOLOR, (XMARGIN, TOPMARGIN, BOXSIZE * BOARDWIDTH, BOXSIZE * BOARDHEIGHT))
    # draw the individual boxes on the board
    for x in range(BOARDWIDTH):
        for y in range(BOARDHEIGHT):
            drawBox(x, y, board[x][y])
```

ㄴ보드의 테두리와 보드의 모든 박스에 대한 그리기 기능을 호출하는 역할이다.

```
def drawStatus(score, level):
    # draw the score text
    scoreSurf = BASICFONT.render('Score: %s' % score, True, YELLOW)
    scoreRect = scoreSurf.get_rect()
    scoreRect.topleft = (WINDOWWIDTH - 150, 20)
    DISPLAYSURF.blit(scoreSurf, scoreRect)

    # draw the level text
    levelSurf = BASICFONT.render('Level: %s' % level, True, YELLOW)
    levelRect = levelSurf.get_rect()
    levelRect.topleft = (WINDOWWIDTH - 150, 50)
    DISPLAYSURF.blit(levelSurf, levelRect)

def drawTime(pt):
    # draw the play time text
    timeSurf = BASICFONT.render('Play Time: %d sec' % pt, True, YELLOW)
    timeRect = timeSurf.get_rect()
    timeRect.topleft = (WINDOWWIDTH - 600, 20)
    DISPLAYSURF.blit(timeSurf, timeRect)
```

ㄴdrawStatus()함수는 화면 모서리의 오른쪽 상단에 나타나는 Score : 및 Level : 정보에 대한 텍스트를 렌더링하는 역할이며 drawTime()함수는 화면 모서리 왼쪽 상단의 Play Time : 정보에 대한 텍스트를 렌더링하는 역할이다.

```
def drawPiece(piece, pixelx=None, pixely=None):
    shapeToDraw = PIECES[piece['shape']][piece['rotation']]
    if pixelx == None and pixely == None:
        # if pixelx & pixely hasn't been specified, use the location stored in the piece data structure
        pixelx, pixely = convertToPixelCoords(piece['x'], piece['y'])

    # draw each of the boxes that make up the piece
    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))
```

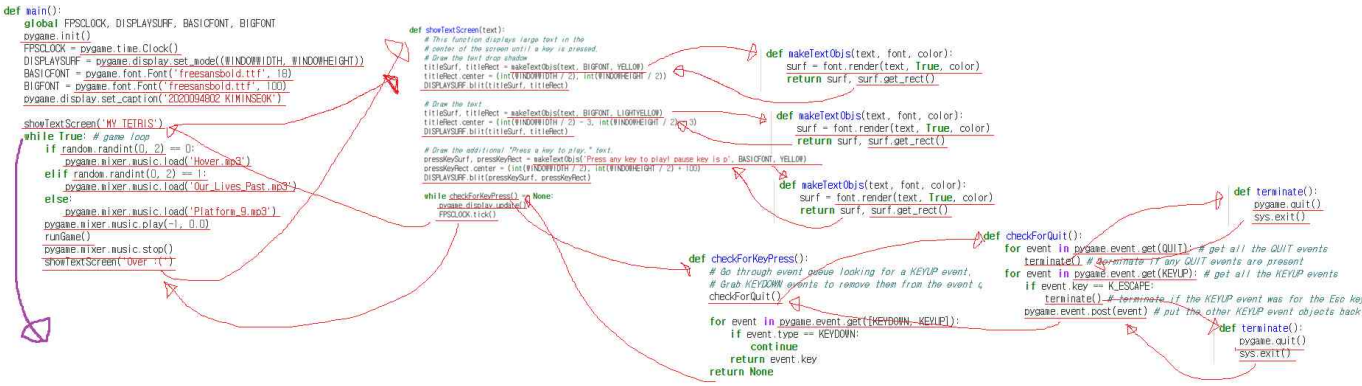
ㄴdrawPiece()기능은 피스로 전달되는 피스 데이터 구조에 따라 피스의 상자를 그리는 데에 사용된다. 이 기능은 낙하물과 Next 피스를 그릴 때 사용된다.

```
def drawNextPiece(piece):
    # draw the "next" text
    nextSurf = BASICFONT.render('Next:', True, YELLOW)
    nextRect = nextSurf.get_rect()
    nextRect.topleft = (WINDOWWIDTH - 120, 80)
    DISPLAYSURF.blit(nextSurf, nextRect)
    # draw the "next" piece
    drawPiece(piece, pixelx=WINDOWWIDTH-120, pixely=100)


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

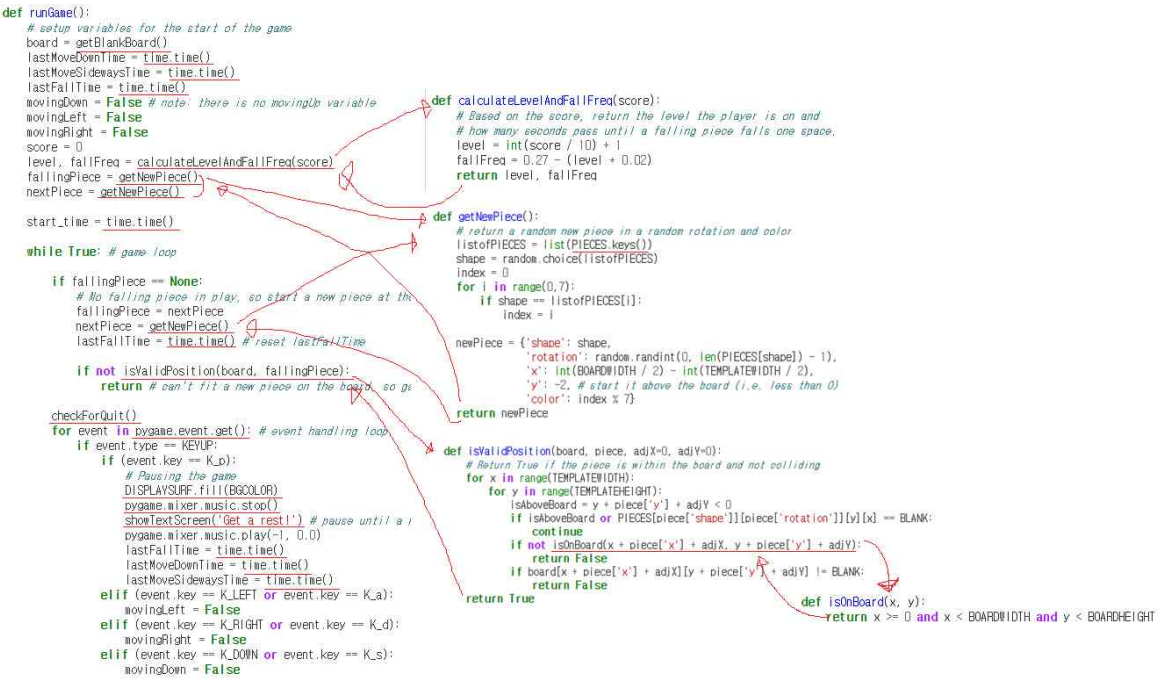
ㄴ위 함수는 화면 오른쪽 상단에 Next 피스를 그리는 함수이며, drawPiece()함수를 호출하고 drawPiece() pixelx 및 픽셀 매개 변수에 대한 인수를 전달하며 이를 수행한다.
그 아래 부분은 main()함수를 호출하여 프로그램의 주요 부분을 시작하는 부분이다.

2. 함수의 호출 순서 또는 호출 조건

함수 호출 순서는 기본적으로 빨간밑줄이 그어진 것을 위에서 아래 순서로 보면 되며 화살표가 있을 시 따라 읽으면 된다. 다시 돌아오는 화살표는 그 함수의 실행이 끝나면 돌아오는 것으로 해석하면 된다.



보라색 화살표부분에서 만약 게임이 진행중일 때 0~2의 랜덤 숫자 중 0이 나오면 pygame.mixer.music.load('Hover.mp3')를 진행시키고 1이 나오면 pygame.mixer.music.load('Our_Lives_Past.mp3')를 진행 시키고 둘다 아니면 pygame.mixer.music.load('Platform_9.mp3')를 진행 시킨다.
runGame()의 실행부분은 다음과 같다.

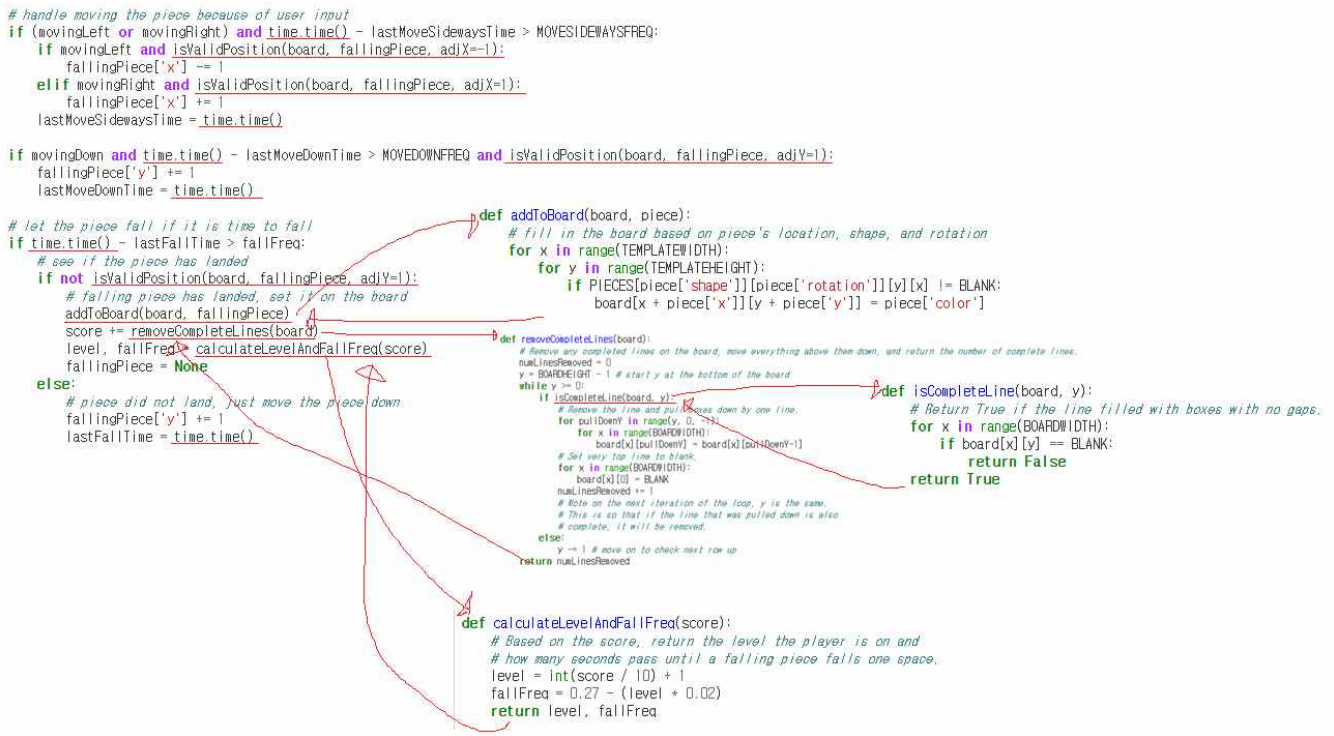


while문은 게임이 진행중일 때 만약 하강중인 블록이 있다면 if문 안의 함수가 실행되며 isValidPosition()함수를 실행시킨 후 isOnBoard()까지 검사한다. chectForQuit()과

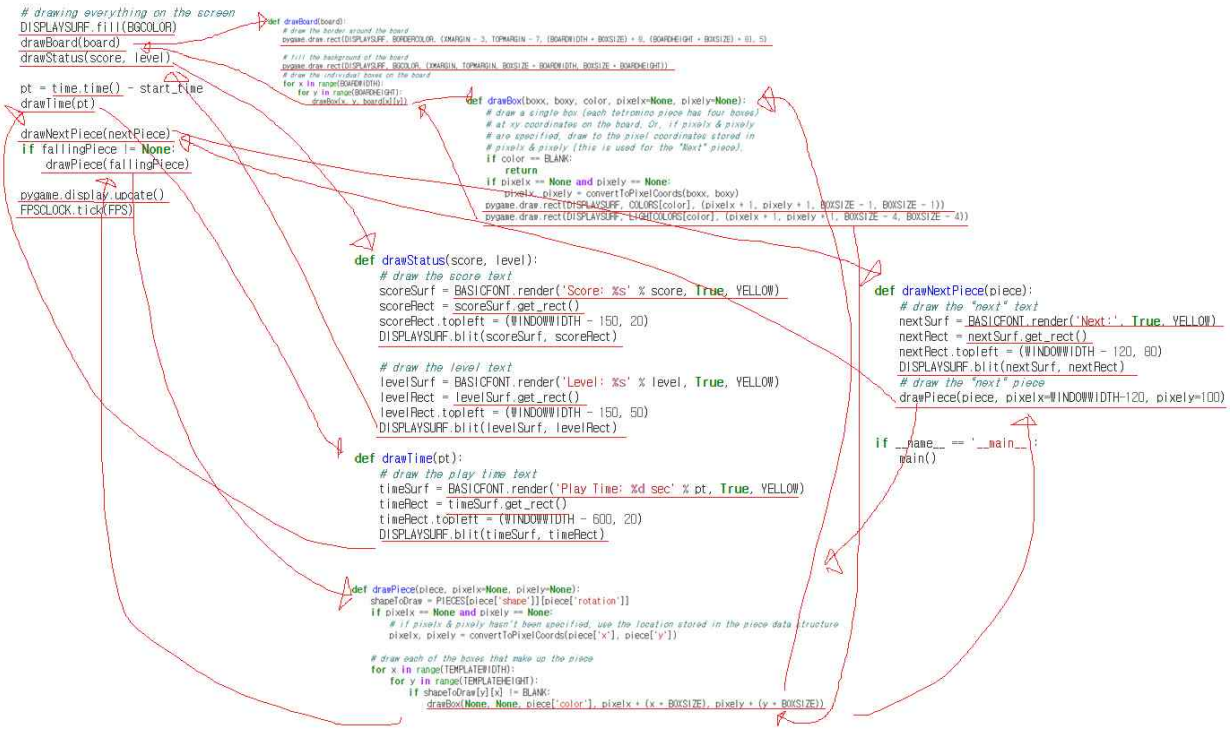
showTextScreen()은 main함수에서 다루었으므로 생략한다. 실행순서는 main함수에서와 같다.

```
elif event.type == KEYDOWN:
    # moving the piece sideways
    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()

    # rotating the piece (if there is room to rotate)
    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']])

    # making the piece fall faster with the down key
    elif (event.key == K_DOWN or event.key == K_s):
        movingDown = True
        if isValidPosition(board, fallingPiece, adjY=1):
            fallingPiece['y'] += 1
        lastMoveDownTime = time.time()

    # move the current piece all the way down
    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
```

isValidPosition()함수도 마찬가지로 생략한다. 위 함수들은 키가 눌리고 있다면 실행되는 것으로 처음 if문부터 살피며 아래로 쭉 순서대로 실행시키며 내려간다. 이 때 3번째 나오는 elif문부터 선택적으로 함수가 실행되는데 플레이어가 어떤 키를 누르느냐에 따라 어떤 isValidPosition()함수가 실행되는지 달라진다. 여기서는 위화살표나 w를 눌렀을 때 혹은 q를 눌렀을 때 혹은 아래화살표나 s, 스페이스 키를 누르느냐에 따라 각 조건에 맞게 각 조건문 안에서의 함수가 실행되게 된다.



isValidPosition()함수는 생략하고 만약 좌우로 움직이고 있고 사용자가 0.15초이상 키를 눌렀을 경우 처음 if 문에서의 함수가 실행된다. 이 때 유효한 위치에 있는지 확인하는 isValidPosition()함수가 실행되게 된다. 만약 좌측으로 이동하고 있다면 1번만 실행될 것이고 우측이라면 한 번 더 실행될 것이다. 혹은 아래로 움직이고 있다면 1번 실행될 것이다.
만약 유효한 위치에 있지 않다면 addToBoard()함수와 removeCompleteLines(), calculateLevelAndFallFreq()함수가 실행된다.



마지막으로 drawBoard()와 drawStatus()가 실행된다. 이후 drawTime()이 실행되며 drawNextPiece가 실행된다. 여기서 drawPiece()가 실행되는 데 이 때 만약 비어있다면 drawBox()가 실행되고 아니라면 실행되지 않는다. 이후 하강 중이라면 drawPiece()를 한번 더 실행한 후 pygame.display.update()와 FPSCLOCK.tick()가 실행된다.