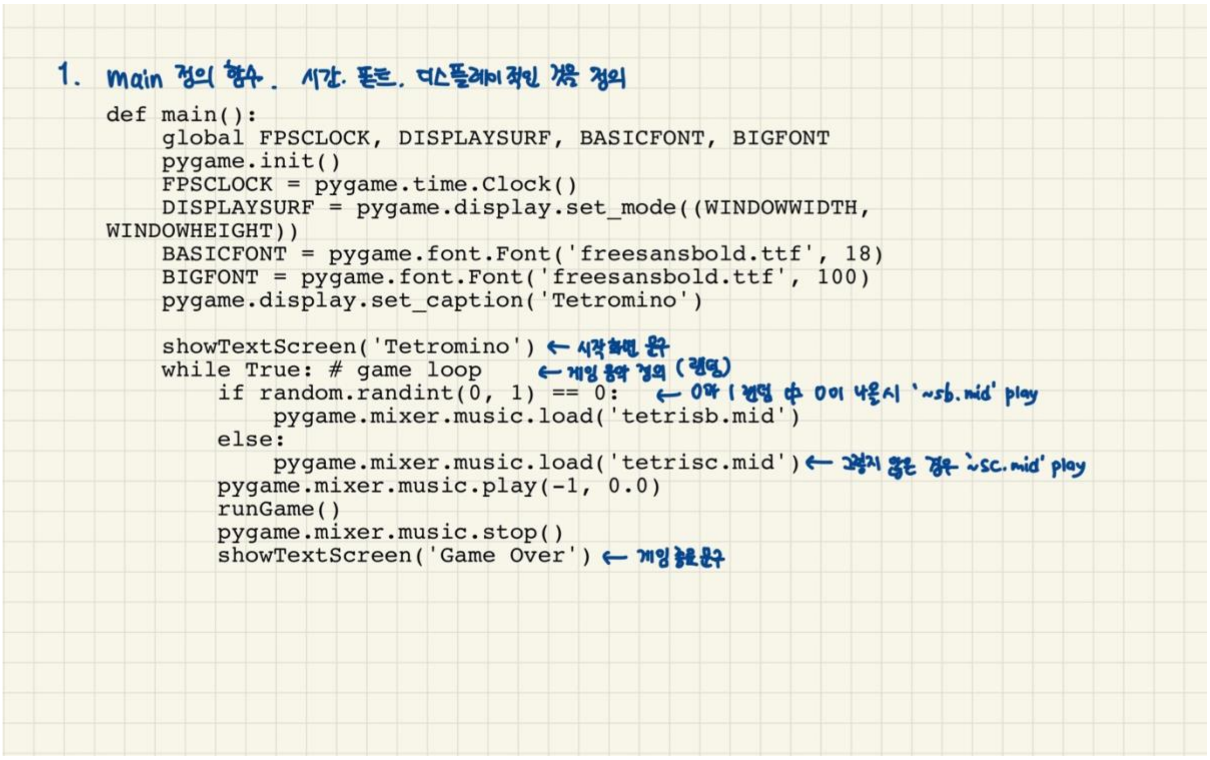
오픈소스SW 과제중심수업 보고서

ICT융합학부 컬쳐테크 전공
2020004257 김혜원

GitHub repository 주소 : https://github.com/hynnk0/osw_/tree/main

1. 각 함수들의 역할 & 함수의 호출 순서 또는 호출 조건



1. main 정의 함수. 시간, 폰트, 디스플레이 적인 것을 정의

```
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') ← 시작화면 문구
    while True: # game loop     ← 게임 음악 정의 (랜덤)
        if random.randint(0, 1) == 0:   ← 0과 1 랜덤 中 0이 나올시 '~sb.mid' play
            pygame.mixer.music.load('tetrisb.mid')
        else:
            pygame.mixer.music.load('tetrisc.mid')← 그렇지 않은 경우 '~sc.mid' play
        pygame.mixer.music.play(-1, 0.0)
        runGame()
        pygame.mixer.music.stop()
        showTextScreen('Game Over') ← 게임 종료문구
```

2. run Game 정의 : 게임 보드, 좌우·상·하 움직임, 레벨 etc

```
def runGame():                                                 새 게임 시작
    # setup variables for the start of the game
    board = getBlankBoard()
    lastMoveDownTime = time.time()                                 시작 전 모두 초기화
    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: # game loop  ← 게임 반복 루프
        if fallingPiece == None:  ← 떨어지는 조각이 X : 다음 조각을 떨어뜨려라
            # 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): ← 보드에 새 조각 들어갈 자리 X : Game over
                return # can't fit a new piece on the board, so game over
```

2-1. 게임 멈추기

```
checkForQuit()
        for event in pygame.event.get(): # event handling loop → 게임 루프 돌아가는 中
            if event.type == KEYUP:
                if (event.key == K_p): → 'p' key를 누르면
                    # Pausing the game
                    DISPLAYSURF.fill(BGCOLOR)
                    pygame.mixer.music.stop() → 음악 stop
                    showTextScreen('Paused') # pause until a key press → ' ' 문구 띄움
                    pygame.mixer.music.play(-1, 0.0)
                    lastFallTime = time.time()
                    lastMoveDownTime = time.time()          ) → 시간 `paused'
                    lastMoveSidewaysTime = time.time()
                                                                Paused에서
                elif (event.key == K_LEFT or event.key == K_a): → LEFT or 'a' key를 누는 경우
                    movingLeft = False                          움직이는데 실패
                elif (event.key == K_RIGHT or event.key == K_d)
                    movingRight = False
                elif (event.key == K_DOWN or event.key == K_s):  ⇒ paused에서
                    movingDown = False                              다른 키 누는 경우
                                                                    움직임 실패
```

2-2. 조각 움직이기

```
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()
```

각각 좌측, 우측에 맞는 키를 입력한 경우 좌측→좌 우측→우 로 움직이게 하는 함수

2-3. 조각 rotation 돌리기 + 조각 빨리 떨어지게 하기 + $\overset{\text{한번에}}{\text{조각}}$ 다 떨어지게 하기

```
# 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']])
```

회전

회전X→원래

밀거나 회전시 유효한가

```
                # 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
```

바닥 찍게하기

2-4. 유저의 입력 결과에 따라 움직이기 + 조각 시간 다 지면 떨어트리기 + 화면에 보드·점수 등 나타내기

```
  # 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()

        # 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()

        # drawing everything on the screen
        DISPLAYSURF.fill(BGCOLOR)
        drawBoard(board)
        drawStatus(score, level)
        drawNextPiece(nextPiece)
        if fallingPiece != None:
            drawPiece(fallingPiece)

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

3. 텍스트 폰트 등 만들기 1

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

텍스트 만드는 단축 함수

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

terminate 함수

```
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
```

키 눌림 발생했는지 기다리기

4. 화면에 Text 띄우기

```
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, TEXTSHADOWCOLOR)
    titleRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2))
    DISPLAYSURF.blit(titleSurf, titleRect)

    # Draw the text
    titleSurf, titleRect = makeTextObjs(text, BIGFONT, TEXTCOLOR)
    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 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): # 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
```

5. 레벨과 그에 따른 떨어지는 속도 정하기 + 새로운 조각 떨어트리기 + 보드 추가하기 & 새로운 보드 로딩하기

```
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

def getNewPiece():
    # return a random new piece in a random rotation and color
    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, # start it above the board (i.e. less than 0)
                'color': random.randint(0, len(COLORS)-1)}
    return newPiece
```

새 피스 만들기

```
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']
```

보드에 피스 추가하기

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

새보드 데이터 구조 만들기

6.

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

보드 플레이 中

```
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
```

유효한 조각 움직임 범위

```
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


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
```

완료된 줄 조각 지우기

7.

```
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))
```

보드 좌표를 픽셀 좌표로 변경

```
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))
```

보드 or 화면에 상자 그리기

```
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])
```

스크린에 모두 그리기

8.

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

현재 스코어 & 상태 나타내기

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

레벨 표시

```
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))
```

보드 or 화면에 피스 그리기

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

다음 조각

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

main 함수 실행

9.

```
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}
```

모양

. = 빈칸

O = 상자