오픈소스SW 과제중심수업 보고서

ICT융합학부 미디어테크놀로지 전공

2020086708 권문성

GitHub repository 주소 : https://github.com/331qjs/osw

1. 각 함수들의 역할

```
//글로벌 상수를 선언하고, 프로그램을 실행할 때 나타날 시작화면을 보여주는 기능을 하는 함수
게임을 하는 동안 'tetrisb.mid', 'tetrisc.mid' 음악 중 하나가 랜덤으로 나오고, runGame() 함수를 호출해 게임을 진행함. 게임이 끝나면 'Game Over' 텍스트가 화면에 나타남.
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: # game loop
        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')
```

```
def runGame():
    //변수선언생략//
        while True: # game loop
        if fallingPiece == None:  //떨어지는 조각이 착지한 후 없음으로 설정되므로
            fallingPiece = nextPiece //떨어지는 조각은 다음 조각을 받고
            nextPiece = getNewPiece()  //그 다음 조각은 새 조각을 받음
            lastFallTime = time.time() # reset lastFallTime
```

```
//일반적으로 조각의 일부가 이미 보드 위에 있음, 보드가 가득 차서 이 위치가 유효하지 않은 경우엔 게임 패배
            if not isValidPosition(board, fallingPiece):
                return # can't fit a new piece on the board, so game over
```

```
        checkForQuit()
        for event in pygame.event.get(): # event handling loop //사용자가 발생시킨 이벤트를 가져옴
            if event.type == KEYUP:  // 키가 올라갔을 때
                if (event.key == K_p): //사용자가 p를 누르면
                    DISPLAYSURF.fill(BGCOLOR)  //디스플레이 Surface를 비운다
                    pygame.mixer.music.stop()  //음악 중지
                    showTextScreen('Paused') # pause until a key press
                      //Paused 텍스트 표시하고 다시 키를 누를 때까지 기다림
```

```
                    pygame.mixer.music.play(-1, 0.0) //음악재생
                    lastFallTime = time.time()
                    lastMoveDownTime = time.time()
                    lastMoveSidewaysTime = time.time()
//화살표키나 WASD키 중 하나를 누르면 movingLeft, movingRight, movingDown 변수가 False로 설정됨
                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:  //키가 눌렸을 때
                //왼쪽 방향키나 a를 눌렀을 때 떨어지는 조각 왼쪽으로 한 칸 이동
                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()
```

```
                //오른쪽 방향키나 d를 눌렀을 때 떨어지는 조각 오른쪽으로 한 칸 이동
                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()
```

```
//위쪽 방향키나 w를 누르면 fallingPiece 딕셔너리의 'rotation' key의 값을 1 증가해 떨어지는 조각이 회전한다.
만약 회전된 위치가 보드에 있는 상자와 겹치기 때문에 유효하지 않으면 다시 1을 빼서 원래대로 다시 전환한다.
                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']])
```

```
//q를 누르면 위쪽 방향키나 w를 누른 경우와 반대방향으로 회전한다.
                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']])
```

```
 //아래쪽 방향키나 s를 누르면 떨어지는 조각이 평소보다 빨리 떨어진다.
                elif (event.key == K_DOWN or event.key == K_s):
                    movingDown = True
                    if isValidPosition(board, fallingPiece, adjY=1):
                        fallingPiece['y'] += 1
                    lastMoveDownTime = time.time()
```

```
// 스페이스바를 누르면 떨어지는 조각이 곧바로 바닥으로 떨어진다.
                elif event.key == K_SPACE: // 스페이스바를 누르면 떨어지는 조각이 곧바로 바닥으로 떨어진다.
                    movingDown = False
                    movingLeft = False
                    movingRight = False
                    for i in range(1, BOARDHEIGHT):
                        if not isValidPosition(board, fallingPiece, adjY=i):
                            break
                    fallingPiece['y'] += i - 1
```

```
        //사용자가 좌우로 이동하는 키를 0.15초 이상 누르면
        if (movingLeft or movingRight) and time.time() - lastMoveSidewaysTime > MOVESIDEWAYSFREQ:
            //왼쪽 방향키나 a를 계속 누른 상태로 있으면 떨어지는 조각이 왼쪽으로 계속 이동
            if movingLeft and isValidPosition(board, fallingPiece, adjX=-1):
                fallingPiece['x'] -= 1
            //오른쪽 방향키나 d를 계속 누른 상태로 떨어지는 조각이 오른쪽으로 계속 이동
            elif movingRight and isValidPosition(board, fallingPiece, adjX=1):
                fallingPiece['x'] += 1
            lastMoveSidewaysTime = time.time()
```

```
//아래쪽 방향키나 s를 0.15초 이상 계속 누른 상태로 있으면 떨어지는 조각이 아래쪽으로 계속 이동
        if movingDown and time.time() - lastMoveDownTime > MOVEDOWNFREQ and isValidPosition(board,
fallingPiece,
            fallingPiece['y'] += 1
            lastMoveDownTime = time.time()
```

```
        if time.time() - lastFallTime > fallFreq:
            if not isValidPosition(board, fallingPiece, adjY=1): //조각이 바닥에 도착했다면
                addToBoard(board, fallingPiece) //보드 데이터 구조의 조각 부분을 만들고
                score += removeCompleteLines(board) // 보드의 전체 줄을 지우고 위에 있는 상자들이 아래로 내려
가면 score 값을 1 증가
                level, fallFreq = calculateLevelAndFallFreq(score) //현재 레벨과 조각이 떨어지는 빈도를 업데이트
                fallingPiece = None
            else: //조각이 바닥에 도착하지 않았으면
                fallingPiece['y'] += 1 //한 칸 아래로 조각을 이동
                lastFallTime = time.time()
```

```
//화면에 보드, score, level, 새로운 조각, 떨어지는 조각을 모두 표시되도록 함
        DISPLAYSURF.fill(BGCOLOR)
        drawBoard(board)
        drawStatus(score, level)
        drawNextPiece(nextPiece)
        if fallingPiece != None:
            drawPiece(fallingPiece)

        pygame.display.update() //디스플레이 Surface가 실제로 화면에 나타남
        FPSCLOCK.tick(FPS) //약간의 일시정지
```

```
def makeTextObjs(text, font, color): //텍스트에 대한 Surface 및 Rect 객체 반환
    surf = font.render(text, True, color)
    return surf, surf.get_rect()

def terminate(): //게임 종료
    pygame.quit()
    sys.exit()
```

```
//CheckForQuit() 함수를 호출하고, 이벤트 큐에서 KEYUP, KEYDOWN 이벤트를 꺼내고 KEYDOWN 이벤트는
무시한다. 이벤트 큐에 KEYUP 이벤트가 없으면 함수는 None을 반환
def checkForKeyPress():
    checkForQuit()
    for event in pygame.event.get([KEYDOWN, KEYUP]):
        if event.type == KEYDOWN:
            continue
        return event.key
    return None
```

```
//화면에 보여질 텍스트 설정
def showTextScreen(text):
    //텍스트 그림자
    titleSurf, titleRect = makeTextObjs(text, BIGFONT, TEXTSHADOWCOLOR)
    titleRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2))
    DISPLAYSURF.blit(titleSurf, titleRect)

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

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

     //사용자가 키보드를 누를 때까지 화면 유지
    while checkForKeyPress() == None:
        pygame.display.update()
        FPSCLOCK.tick()
```

```
//이벤트 큐에 QUIT 이벤트가 있거나 Esc 키의 KEYUP 이벤트가 있으면 프로그램 종료
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
```

```
//score가 10점이 될 때마다 level이 1씩 증가하고 조각이 더 빨리 떨어지게 함
def calculateLevelAndFallFreq(score):
    level = int(score / 10) + 1
    fallFreq = 0.27 - (level * 0.02)
    return level, fallFreq
```

```
// 무작위의 모양과 색으로 새로운 조각을 보드 위쪽에 생성
def getNewPiece():
    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
```

```
//바닥에 떨어져있는 조각의 데이터 구조를 가져와서 해당 상자를 보드 데이터 구조에 추가
def addToBoard(board, piece):
    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']
```

```
//새 빈 보드 데이터 구조를 만들고 반환
def getBlankBoard(): //
    # create and return a new blank board data structure
    board = []
    for i in range(BOARDWIDTH):
        board.append([BLANK] * BOARDHEIGHT)
    return board
```

```
// 값으로 받은 x, y가 보드에 존재하는 유효한 값인지 확인하는 함수
def isOnBoard(x, y):
    return x >= 0 and x < BOARDWIDTH and y < BOARDHEIGHT
```

```
////조각의 상자가 보드에 있고 보드에 있는 상자와 겹치지 않으면 True 반환
def isValidPosition(board, piece, adjX=0, adjY=0):
    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: //조각의 공간이 보드 위에 있거나 공백이면 코드를 계속 진행
                continue
            if not isOnBoard(x + piece['x'] + adjX, y + piece['y'] + adjY): // 조각의 상자가 보드에 없으면 False 반환
                return False
            if board[x + piece['x'] + adjX][y + piece['y'] + adjY] != BLANK: //조각의 상자가 위치한 보드의 공간이 비어 있지 않으면 False 반환
                return False
    return True
```

```
//y에 해당하는 줄이 공백없이 전부 채워져 있으면 True 반환
def isCompleteLine(board, y):
    for x in range(BOARDWIDTH):
        if board[x][y] == BLANK:
            return False
    return True
```

```
 //보드에서 가득 찬 줄을 제거하고 그 위에 있는 줄을 모두 아래로 이동한 후 제거된 줄의 수를 반환
def removeCompleteLines(board):
    numLinesRemoved = 0
    y = BOARDHEIGHT - 1 # start y at the bottom of the board
    while y >= 0:
        if isCompleteLine(board, y):
            for pullDownY in range(y, 0, -1):
                for x in range(BOARDWIDTH):
                    board[x][pullDownY] = board[x][pullDownY-1]
            for x in range(BOARDWIDTH):
                board[x][0] = BLANK
            numLinesRemoved += 1
        else:
            y -= 1 # move on to check next row up
    return numLinesRemoved
```

```
//보드의 상자 좌표를 픽셀 죄표로 변환하는 함수
def convertToPixelCoords(boxx, boxy):
 return (XMARGIN + (boxx * BOXSIZE)), (TOPMARGIN + (boxy * BOXSIZE))
```

```
//화면에 하나의 상자를 그리는 함수
def drawBox(boxx, boxy, color, pixelx=None, pixely=None):
    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):
     //보드의 모서리 그리기
    pygame.draw.rect(DISPLAYSURF, BORDERCOLOR, (XMARGIN - 3, TOPMARGIN - 7, (BOARDWIDTH * BOXSIZE) + 8, (BOARDHEIGHT * BOXSIZE) + 8), 5)
     //보드의 배경 그리기
    pygame.draw.rect(DISPLAYSURF, BGCOLOR, (XMARGIN, TOPMARGIN, BOXSIZE * BOARDWIDTH, BOXSIZE * BOARDHEIGHT))
    //보드에 각각의 상자 그리기
    for x in range(BOARDWIDTH):
        for y in range(BOARDHEIGHT):
            drawBox(x, y, board[x][y])
```

```
//점수와 레벨을 화면에 나타내는 함수
def drawStatus(score, level):
    scoreSurf = BASICFONT.render('Score: %s' % score, True, TEXTCOLOR)
    scoreRect = scoreSurf.get_rect()
    scoreRect.topleft = (WINDOWWIDTH - 150, 20)
    DISPLAYSURF.blit(scoreSurf, scoreRect)

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

```
//조각의 모양, 위치, 회전, 색에 따라 조각의 상자를 그리는 함수
def drawPiece(piece, pixelx=None, pixely=None):
    shapeToDraw = PIECES[piece['shape']][piece['rotation']]
    if pixelx == None and pixely == None:
        pixelx, pixely = convertToPixelCoords(piece['x'], piece['y'])

    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))
```

```
//Next라는 텍스트를 화면에 나타내고, 다음으로 떨어질 조각 그리기
def drawNextPiece(piece):
    nextSurf = BASICFONT.render('Next:', True, TEXTCOLOR)
    nextRect = nextSurf.get_rect()
    nextRect.topleft = (WINDOWWIDTH - 120, 80)
    DISPLAYSURF.blit(nextSurf, nextRect)
    drawPiece(piece, pixelx=WINDOWWIDTH-120, pixely=100)
```

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

2. 함수의 호출 순서 또는 호출 조건

1. main() 함수 내에서 showTextScreen('Tetromino') 함수 호출

2. showTextScreen(Text) 함수 내에서 makeTextObjs(text, BIGFONT, TEXTSHADOWCOLOR) 2번, makeTextObjs('Press a key to play.', BASICFONT, TEXTCOLOR) 1번으로 makeTextObjs(text, font, color) 함수 총 3번 호출됨,
while의 조건으로 checkForKeyPress() 호출됨

3. checkForKeyPress()의 함수 내에서 checkForQuit() 호출됨

4. checkForQuit() 함수 내에서 terminate() 호출됨

5. 다시 main()으로 돌아와서 whil True: 내에 있는 runGame() 호출

< runGame()내에서 일어나는 함수 호출 순서 >

6. getBlankBoard() 함수 호출됨

7. calculateLevelAndFallFreq(score) 호출됨

8. fallingPiece와 nextPiece 변수에 값을 할당하기 위해 각각 getNewPiece() 함수 호출됨

9. 만약 fallingPiece 변수의 값이 None이면, nextPiece 값 할당을 위해 getNewPiece() 호출
그 이후 isValidPosition(board, fallingPiece) 함수 호출됨

10. checkForQuit() 함수 호출

11. checkForQuit() 함수 내에서 terminate() 함수 호출됨

for event in pygame.event.get()가 성립할 때

12. 이벤트 타입이 KEYUP이고, p키를 누르면, showTextScreen('Paused') 함수 호출됨

이벤트 타입이 KEYDOWN이면

실행될 코드의 조건확인을 위해 if문에서
13. isValidPosition(board, fallingPiece, adjX=-1) 함수 총 2번 호출됨
14. isValidPosition(board, piece, adjX=0, adjY=0) 내에서 if not문 조건에
isOnBoard(x + piece['x'] + adjX, y + piece['y'] + adjY) 호출

15. 위쪽 방향키 또는 w를 눌렀을 때,
if not문에서 isValidPosition(board, fallingPiece) 호출
16. isValidPosition(board, piece, adjX=0, adjY=0) 내에서 if not문 조건에
isOnBoard(x + piece['x'] + adjX, y + piece['y'] + adjY) 호출

17. q를 눌렀을 때, if not문에서 isValidPosition(board, fallingPiece) 호출
18. isValidPosition(board, piece, adjX=0, adjY=0) 내에서 if not문 조건에
isOnBoard(x + piece['x'] + adjX, y + piece['y'] + adjY) 호출

19. 스페이스바를 눌렀을 때
if not 문에서 isValidPosition(board, fallingPiece, adjY=i) 호출
20. isValidPosition(board, piece, adjX=0, adjY=0) 내에서 if not문 조건에
isOnBoard(x + piece['x'] + adjX, y + piece['y'] + adjY) 호출

16. 아래쪽 방향키 또는 s를 눌렀을 때, if문에서
isValidPosition(board, fallingPiece, adjY=1) 호출

for event in pygame.event.get() 반복이 끝난 후

(movingLeft or movingRight) and time.time() - lastMoveSidewaysTime > MOVESIDEWAYSFREQ이면,
21. if 조건문들에서 isValidPosition(board, fallingPiece, adjX=-1) 호출 (총 2번)
22. isValidPosition(board, piece, adjX=0, adjY=0) 내에서 if not문 조건에
isOnBoard(x + piece['x'] + adjX, y + piece['y'] + adjY) 호출

23. if 문에서 조건확인을 위해 isValidPosition(board, fallingPiece, adjY=1) 호출
24. isValidPosition(board, piece, adjX=0, adjY=0) 내에서 if not문 조건에
isOnBoard(x + piece['x'] + adjX, y + piece['y'] + adjY) 호출

time.time() - lastFallTime > fallFreq이면,
25. if not문에서 조건확인을 위해 isValidPosition(board, fallingPiece, adjY=1) 호출
26. isValidPosition(board, piece, adjX=0, adjY=0) 내에서 if not문 조건에
isOnBoard(x + piece['x'] + adjX, y + piece['y'] + adjY) 호출

27. 위의 20번 조건이 참이라면, addToBoard(board, fallingPiece) 호출

28. 위의 20번 조건이 참이라면,
score 변수 값을 변화하기 위해 removeCompleteLines(board) 호출

29. removeCompleteLines(board)함수 내의
if 조건문에서 isCompleteLine(board, y) 호출

30. 위의 20번 조건이 참이라면,
level, fallFreq 변수 값을 할당하기 위해 calculateLevelAndFallFreq(score) 호출

31. drawBoard(board) 호출

32. drawStatus(score, level) 호출

33. drawNextPiece(nextPiece) 호출

34. drawNextPiece(piece) 함수 내에서
drawPiece(piece, pixelx=WINDOWWIDTH-120, pixely=100) 호출

35. drawPiece(piece, pixelx=None, pixely=None) 함수 내에서
pixelx와 pixely 모두 None일 경우, pixelx와 pixely 변수의 값에
convertToPixelCoords(piece['x'], piece['y']) 호출

36. for x in range(TEMPLATEWIDTH):
for y in range(TEMPLATEHEIGHT):에서
shapeToDraw[y][x] 이 비어있는 것이 아니라면
drawBox(None, None, piece['color'], pixelx + (x * BOXSIZE),
pixely + (y * BOXSIZE)) 호출

37. drawBox(boxx, boxy, color, pixelx=None, pixely=None) 내에서 pixelx와 pixely 모두 None일 경우, pixelx와 pixely 변수의 값에 convertToPixelCoords(piece['x'], piece['y']) 호출

38. fallingPiece가 None이 아니라면 drawPiece(fallingPiece) 호출

39. drawPiece(piece, pixelx=None, pixely=None) 함수 내에서 pixelx와 pixely 모두 None일 경우, pixelx와 pixely 변수의 값에 convertToPixelCoords(piece['x'], piece['y']) 호출

40. for x in range(TEMPLATEWIDTH):
    for y in range(TEMPLATEHEIGHT):에서 shapeToDraw[y][x] 이 비어있는 것이 아니라면 drawBox(None, None, piece['color'], pixelx + (x * BOXSIZE), pixely + (y * BOXSIZE)) 호출

41. drawBox(boxx, boxy, color, pixelx=None, pixely=None) 내에서 pixelx와 pixely 모두 None일 경우, pixelx와 pixely 변수의 값에 convertToPixelCoords(piece['x'], piece['y']) 호출

runGame() 실행 후

42. main()에서 마지막으로 showTextScreen('Game Over') 호출