오픈소스SW 과제중심수업 보고서

ICT융합학부 미디어테크 전공
2018045641 오윤석

GitHub repository 주소 : https://github.com/Yeunseok97/osw

1. 각 함수들의 역할

♠ main(과제)

```
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('2018045641 OHYEUNSEOK')

    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 :(')
```

main 함수는 FPSCLOCK, DISPLAYSURF, BASICFONT, BIGFONT와 같은 전역변수를 설정하고, 인 게임 화면상에 보이는 텍스트를 출력한다
(★과제) 메인화면 문구변경, pygame.display.set_caption 변경

또한 게임 시작 시 재생할 BGM을 무작위로 결정한 다음 runGame()을 호출 하여 게임을 시작한다 플레이어가 패배하면 runGame함수에서 main함수로 돌아가고 배경 음악이 중지되고 게임 오버를 알리는 문구가 화면 위에 표시됩니다.
(★과제) mp3.파일 Hover, Our_Lives_Past, Platform_9 다운받고 적용시키기

플레이어가 키를 누르면 게임 오버 화면을 표시하는 showTextScreen () 함수가 반환되고 게임은 다시 While True: 문으로 돌아가서 새 게임이 실행된다.

♠ run(1)

```
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
    startTime = time.time()
    score = 0
    level, fallFreq = calculateLevelAndFallFreq(score)
    fallingPiece = getNewPiece()
    nextPiece = getNewPiece()

    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
```

runGame함수는 실질적으로 게임을 실행하는 함수로써 먼저 run(1)에서 startTime, score, lastFallTime, lastMoveSidewaysTime, LastMoveDownTime,을 초기화한다. 보드판은getBlankBoard함수값 할당하고, 단계와, 떨어지는 속도는 calculateLevelAndFallFreq함수값 할당한다. 떨어지는 조각과 새로 생성할 조각은 getnewPiece함수값을 할당하여 생성한다.

게임 루프는 먼저 떨어지는 조각이 없으면 다음 조각이 현재 떨어지는 조각으로 할당되고 다음 조각은 getNewPiece에서 새로운 조각을 생성한다. 또한 lastFallTime은 현재시간으로 초기화한다.

그리고 조각이 보드판 맨 윗라인을 넘게되면 isValidPosition이 False값을 반환하게 되고 게임 오버가 된다.

♠ run(2)

```
checkForQuit()
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

    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()
```

다음은 게임 내 기능들이 포함된 부분으로 만약에 p 키가 눌렸으면 음악이 멈추고 pause기능이 실행되고, 좌, 우, 하단으로 이동할 수 있게 해준다. 또한 사용자가 조각을 이동할 때 키를 눌렀다가 떼면 그 방향으로 계속 이동하지 않는 것을 나타내어 한번 눌렀을 때 한 칸만 이동하게 한다.
또한 elif문에선 조각을 회전시킬 때 유효한지 확인해준다.

♠ run(3)

```
# 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
```

해당 부분은 위쪽 키를 눌렀을 때 조각을 다음 회전으로 회전시킨다. 반대로 q를 눌렀을 때는 반대방향으로 회전시켜준다. 또한 아래쪽 키를 누르면 조각을 한칸 밑으로 이동시킨다.
스페이스 키를 눌렀을 때는 조각을 바로 떨어뜨린다.

♠ run(4)

```
    # 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, startTime)
    drawNextPiece(nextPiece)
    if fallingPiece != None:
        drawPiece(fallingPiece)

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

해당 부분은 좌측 또는 우측으로 조각을 이동시킬 때 매번 한번씩 키를 입력해야하는 것이 아니라 꾹 누르고있어도 해당 방향으로 이동을 시킨다.
다음 부분은 사용자로부터 어떠한 입력을 받지 않아도, 조각이 알아서 밑으로 떨어지게한다. 만약 조각이 착지하지 않았다면, 계속 한칸씩 밑으로 내려보낸다.

♠ makeTextObjs

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

이 함수는 텍스트와, 폰트, 색깔이 주어지면 render함수를 호출하고 해당 텍스트에 대한 surf값과 rect값을 반환해주는 함수이다. 텍스트를 만들때마다 추가로 코드를 입력하지 않고 이 함수를 사용하면 된다.

♠ terminate

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

프로그램을 종료시키는 함수이다.

♠ checkForKeyPress

```
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
```

이 함수는 게임 내 키가 입력되었는지 확인하는데 필요한상황(일시정지, 시작 전 메인화면, 게임오버 화면)에 필요한 함수이다. keyup 이벤트를 찾기위해 이벤트 큐를 통과시키고 keydown 이벤트들은 이벤트 큐에서 제거한다. 먼저 checkForQuit 함수를 호출한 후 keyup이벤트가 이벤트큐에 없다면 none값을 반환한다.

♠ checkForQuit

```
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
```

이 함수는 프로그램을 종료할 수 있는 이벤트들을 호출한다. 이벤트 큐 안에 quit 이벤트가 있거나 ESC키(keyup 이벤트의)가 입력되면. terminate함수를 호출하여 언제든지 프로그램을 종료시킨다.

♠ showTextScreen(과제)

```
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 any key to play! pause key is p', BASICFONT, TEXTCOLOR)
    pressKeyRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2) + 100)
    DISPLAYSURF.blit(pressKeySurf, pressKeyRect)

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

시작화면, 게임오버화면을 분리해서 함수를 만드는 대신에 showTextScreen이라는 함수를 사용하여 간소화하였다. 사용자가 임의로 텍스트를 입력하면 makeTextObjs함수 사용하여 문구를 출력해준다. while문을 통해 checkForKeyPress함수에서 반환값이 none이라면 pygame.display.update()와 FPSCLOCK.tick()을 계속 호출한다. 쉽게말해 사용자가 키를 입력하기 전까지 화면에 사용자가 입력해놓은 텍스트가 유지된다는 소리이다.

(★과제)Press any key to play1 pause key is p 문구로 교체

♠ calculateLevelAndFallFreq

```
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
```

이 함수는 단계와 조각의 떨어지는 속도를 설정한 함수이다.
사용자가 1줄을 꽉채워 완성하면 1점을 얻고, 10점마다 단계가 올라가는데 뒤에 +1을 해놓은 것은 score가 시작 시 0이어서 +1이 없다면 0단계가 된다. 시작 단계가 0단계가 아닌 1단계이기 때문에 +1을 추가한 것이다. 떨어지는 속도는 레벨에 비례하여 점점 더 빨라지게 되어있다.

♠ getNewPiece(과제)

```
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':  (list(PIECES.keys()).index(shape))%len(COLORS)}
```

이 함수는 새로운 조각을 만드는 만드는 함수이다.
먼저 임의로 조각을 고르기 위해 가능한 모든 모양의 리스트를 shapes.keys를 호출한다. 또한 dict_keys의 값은 리스트 내의 값이 아니므로 random.choice를 사용하려면 다시 리스트화 시켜야한다. 이후 random.choice를 통해 조각을 무작위로 반환한다.
조각의 회전의 키 값은 조각이 회전 가능한 수보다 1 작은 임의의 정수이다. 그래서 같은 조각이 항상 같은 모양으로 떨어지는 것이 아닌 다른 모양으로 떨어지는 것이다.
x값은 조각이 항상 정 중앙에서 떨어지도록 설정한 것이고, y값은 조각이 떨어질 때 미리 두 칸을 차지하고 떨어지는 것이 아니라 화면 밖에서부터 1칸씩 차례대로 보여야 하므로 -2 값을 넣은 것이다. 0 값이 보드의 맨 위의 행 값이다.
조각의 색은 먼저 생성한 랜덤한 조각의 위치를 index를 이용하여 찾은 후 COLORS 리스트길이로 나머지 연산을 해주면 조각의 색이 고정된다. COLORS의 리스트에 색이 모자라 추가하였다.

```
COLORS      = (     BLUE,      GREEN,      RED,      YELLOW,      PURPLE,      PINK,      BROWN)
LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYELLOW, LIGHTPURPLE, LIGHTPINK, LIGHTBROWN)
assert len(COLORS) == len(LIGHTCOLORS) # each color must have light color
```

assert len의 조건으로 인해 lightcolor 리스트의 색도 추가하였다.
(★과제)조각이 랜덤하게 색을 받는 것이 아닌 각각의 고유색을 가지게함.

♠ addToBoard

```
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']
```

이 함수는 조각이 떨어져서 고정된 조각이 차지하는 공간에 대한 데이터를 표현한다. 현재 조각이 떨어지는 중이면 조각은 데이터 구조에 표시되지 않는다.

♠ getBlankBoard

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

이 함수는 새로운 보드 데이터 구조를 생성한다

♠ def isOnBoard

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

이 함수는 현재 보드 상에 표시된 유효한 x,y좌표값을 확인해주는 함수이다. x,y좌표 값이 둘 다 0보다 크지않고 boardwidth, boardheight상수보다 작으면 함수는 True값을 반환한다.

♠ isValidPosition

```
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
```

해당 함수는 보드판과 조각의 데이터구조를 주고, 조각의 칸과 보드판의 칸이 하나라도 겹치지 않을 때 True값을 반환한다. 조각이 떨어질 때 빈 공간이 있으면 계속 이동 가능하게 떨어지게해주고 해당 공간에 블록이 쌓여져 있으면 통과하지 못하고 멈추게 해주는 함수이다.

♠ isCompleteLine, ♠ removeCompleteLines

```
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
```

isCompleteLine함수는 모든 공간이 남김없이 조각으로 메워지면 컴플리트 상태로 간주하여. for문이 각 공간의 행을 통과한다. 만약 공간이 BLANK값이면 False값을 반환한다. 다음.removeCompleteLines는 컴플리트 상태가 된 줄을 없애고 스코어값을 증가시킨다. 또한 해당줄이 없어지고 난 뒤 그 위에 조각들을 모두 y-1값으로 이동시키고 맨 위에 있던 칸을 비운다.

♠ convertToPixelCoords

```
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))
```

해당 함수는 보드판 상에 주어진 칸의 x,y좌표를 픽셀좌표로 변환해준다.

♠ drawBox

```
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))
```

위 함수는 각 단일 칸을 그리는 함수이다. 각 칸의 x,y좌표값을 받아 보드판에 그려져야할 위치에 그린다.

♠ drawBoard

```
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])
```

위 함수는 게임 내 테두리 밑 배경을 설정하며 drawbox함수를 호출하여 사용자에게 보이는 전체적인 화면을 설정한다

♠ drawStatus(과제)

```
def drawStatus(score, level, startTime):
    # 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)

    # 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)

    # draw the time text
    timeSurf = BASICFONT.render('Playtime: %d sec' %(time.time()- startTime), True, TEXTCOLOR)
    timeRect = timeSurf.get_rect()
    timeRect.topright = (WINDOWWIDTH - 500, 20)
    DISPLAYSURF.blit(timeSurf, timeRect)
```

위 함수는 게임 내 화면에 보이는 점수, 단계 진행시간을 표시해주는 함수이다

(★과제) 좌측 상단에 게임이 진행된 시간을 표시해주는데 먼저runGame함수에 starTime = time.time()를 초기화하여 게임이 새로 진행될 때마다 시간초가 0으로 바뀔 수 있도록 하고 display해주는 곳의 값에 startTime을 추가해준 후 현재 함수에서 time.time()으로 현재 시간을 다시 받은후 startTime을 빼주면 시간초가 증가하게 된다.

♠ drawPiece

```
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))
```

위 함수는 게임 내 떨어지는 조각과 다음에 나올 조각을 그리는데 사용되는 함수이다.

♠ drawNextPiece

```
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)
```

해당 함수는 drawpiece 함수를 호출하여 다음에 떨어지는 조각을 그려서 보여주는 함수이다.

2. 함수의 호출 순서 또는 호출 조건

먼저 모든 함수들의 정의가실행되고 프로그램의 주요 부분을 시작하기 위해 메인 함수가 호출된다.

main() ->

pygame.init()메서드를 호출하여 라이브러리를 초기화 ->

display.set_mode메서드를 호출하여 게임스크린의 크기를 지정 ->

FPSCLOCK에 게임 루프의 주기를 결정할 pygame.time.Clock() 객체를 생성->

폰트를 설정하고 좌측상단에 띄워질 캡션을 설정 ->

다음 showTextscreen함수를 호출하여 메인화면에 띄울 문구를 띄움 ->

showTextscreen()

텍스트 그림자 그리기

titleSurf, titleRect에 makeTextObjs함수 호출

render함수 호출 및 surf, rect값 반환

titleSurf, titleRect 이미지 복사

텍스트 그리기

titleSurf, titleRect에 makeTextObjs함수 호출(상세내용 생략)

위치 지정

titleSurf, titleRect 이미지 복사

일시 정지시 키 입력 문구 그리기

pressKeySurf, pressKeyRect에 makeTextObjs함수 호출(상세내용 생략)

위치 지정

pressKeySurf, pressKeyRect 이미지 복사

(위 문구는 사용자로부터 종료를 제외한 어떤 키를 입력받으면 사라지고 다음 명령실행)

(키입력 후) bgm 3개중 1개가 랜덤으로 실행 ->

runGame함수를 호출하여 실질적인 게임을 실행 ->

runGame함수에서 빠져나오면 음악 중지 ->

게임오버 문구 출력 ->

runGame() ->

(게임시작)

getBlankBoard 함수를 호출하여 보드 데이터 구조 생성 ->

lastMoveDownTime, lastMoveSidewaysTime, lastFallTime 현재시간으로 초기화 ->

movingDown, movingLeft, movingRight, False값으로 초기화 -> score 0으로 초기화 ->

level, fallFreq에 calculateLevelAndFallFreq(score)값 할당 ->

FallingPiece, nextPiece에 getNewPiece()값 할당 ->

(게임루프)

while True문으로 게임 루프 틀 생성 ->

fallingpiece가 없다면 fallingpiece에 nextPiece 값 지정 ->

nextPiece에 getNewPiece함수값 할당 ->

getNewPiece()

조각 랜덤조각 리스트에서 결정 ->

newPiece가 어떤 회전방향으로 떨어질지 결정,

떨어지는 x값은 화면 정 중앙, y값은 보드판 맨 윗부분보다 -2만큼 위쪽에서 등장

각 조각의 색은 종류별로 고정된 색으로 출력

newPiece 반환

lastFallTime 시간 재지정

만약 isValidPosition의 조건이 맞지 않으면 반환값 없음.

checkForQuit함수 호출

이벤트 큐 안에 quit 이벤트가 있거나 ESC키(keyup 이벤트의)가 입력되면.
terminate함수를 호출하여 언제든지 프로그램을 종료

(이벤트 핸들링 루프)

for 문으로 이벤트 핸들링 루프 생성->

이벤트 타입이 KEYUP일 때 ->

(게임 일시중지)

사용자로부터 p키를 입력받음 ->

재생중인 음악 중지 ->

shotextscreen함수로 paused 문구 생성 ->

lastFallTime, lastMoveDownTime, lastMoveSidewaysTime을 time.time으로 초기화 ->

다른 입력키 좌, 우 아래 입력받으면 False값 반환 ->

이벤트 타입이 KEYDOWN일 때 ->

(조각이동)

사용자로부터 좌 키나 a키를 입력받고 isValidPosition의 조건이 충족 ->

왼쪽으로 -1만큼 이동 ->

movingLeft에 True값 지정 ->

movingRight에 False값 지정 ->

lastMoveSidewaysTime time.time()으로 초기화 ->

사용자로부터 우 키나 d키를 입력받고 isValidPosition의 조건이 충족 ->

오른쪽으로 -1만큼 이동 ->

movingLeft에 False값 지정 ->

movingRight에 True값 지정 ->

lastMoveSidewaysTime time.time()으로 초기화 ->

(블록회전)

사용자로부터 위쪽 키나 w키를 입력받은 경우 ->

fallingpiece딕셔너리에서 rotation의 키 값 1 증가시키고 rotation될 수 있는 값으로 나머지연산 ->

회전이 유효하지 않은 경우는 이미 다른 조각이 위치해야 있으면 유효하지 않으므로 rotation 값을 입력받아도 다시 원래회전으로 전환 ->

(블록회전 반대)

사용자로부터 q값 입력 받았을 때 ->

fallingpiece딕셔너리에서 rotation의 키 값 1 감소시키고 rotation될 수 있는 값으로 나머지연산 ->

회전이 유효하지 않은 경우는 이미 다른 조각이 위치해야 있으면 유효하지 않으므로 rotation 값을 입력받아도 다시 원래회전으로 전환 ->

(하단이동)

사용자로부터 아래쪽 키 또는 s키 입력받았을 때

movingDown값에 True 할당

isValidPosition함수에서 확인했을 때 값이 유효한 경우 fallinigpiece 현재 y값에 1값을 더함.

lastMoveDownTime 현재시간으로 초기화

(바닥으로 바로이동)

사용자로부터 스페이스키 입력받음 ->

movingDown, movingLeft, movingmRight 값에 False값 지정 ->

isValidPosition값이 유효하지 않을때까지 아래쪽으로 이동. ->

(키를 고정하고 있어도 그대로 명령수행)

movingLeft 또는 movingRight값과 현재시간 - lastMoveSidewaysTime이 MOVESIDEWAYSFREQ보다 클 때 ->

좌측으로 계속이동 ->

우측으로 계속 이동 ->

movingDown과 현재시간 - lastMoveDownTime이 MOVEDOWNFEWQ와 isValidPosition의 값보다 클 때 ->

하단으로 계속 이동 ->

(사용자가 추가적인 키 입력 이외에 조각이 자연스럽게 떨어지게 함)

만약 현재시간 - lastFallTime 값이 fallFreq보다 클 때 ->

만약 조각이 다 떨어진 경우 ->

addToBoard함수 호출 ->

score에 removeCompleteLines 값을 더하고 그 결과 할당 ->

numLinesRemoved 값에 0을 할당하고 ->

y의 값을 1 뺀다 ->

y값이 0보다 같거나 클때까지 반복한다. ->

level, fallFreq에 calculateLevelAndFallFreq(scrore)값 할당

fallingPiece에 None값 할당

아닌 경우 ->

하단으로 1씩 계속 이동 ->

lastFallTime을 time.time()으로 초기화

배경화면 색깔적용하여 채우기

drawboard 함수 호출 ->

drawStatus (score, level, time) 호출 ->

drawStatus

(score, level, time)+surf, rect, rect.topleft 혹은 right값에 각각, score, level, time.time() - starTime값을 출력한다

drawNextPiece함수 다음조각값을 호출 ->

만약 떨어지는 조각이 없다면 ->

drawpiece함수를 호출하여 새로 떨어질 조각을 그린다.