# 오픈소스SW 과제중심수업 보고서

ICT융합학부 미디어테크놀로지전공

2020015841 한정윤

https://github.com/jyoooon/osw

```
# 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 = '.'

MOVESIDEWAYSFREQ = 0.15
MOVEDOWNFREQ = 0.1
```

```
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)

# 각 블록에 고유한 색을 채워주기 위한 추가 색상
MOLA1           = (150, 150, 150)
LIGHTMOLA1      = (170, 170, 170)
MOLA2           = (200,    80,    40)
LIGHTMOLA2      = (220, 100,    60)
MOLA3           = ( 30,    50, 100)
LIGHTMOLA3      = ( 50,    70, 120)

BORDERCOLOR = BLUE
```

```
BGCOLOR = BLACK

TEXTCOLOR = YELLOW

TEXTSHADOWCOLOR = LIGHTYELLOW

COLORS = ( BLUE, GREEN, RED, YELLOW, MOLA1, MOLA2, MOLA3)

LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYELLOW, LIGHTMOLA1, LIGHTMOLA2, LIGHTMOLA3)

assert len(COLORS) == len(LIGHTCOLORS) # each color must have light color


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}

# 각 블록 모양에 고유의 색상을 지정해준다
PIECES_COLOR = {'S': 0,
                'Z': 1,
                'J': 2,
                'L': 3,
                'I': 4,
                'O': 5,
                'T': 6}
```

# 글로벌 상수를 생성하고 기본적인 설정을 해준다.

```
def main():
    global FPSCLOCK, DISPLAYSURF, BASICFONT, BIGFONT, PLAYTIME
    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('2020015841 HANJUNGYUN')


    showTextScreen('MY TETRIS')
```

# 노래를 랜덤으로 실행하고 실제 게임을 실행하는 runGame()을 선언해주며 게임이 끝나면 노래가 멈추고 'Over :(' 가 나오도록 한다.

```
    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')
        PLAYTIME = time.time()
        pygame.mixer.music.play(-1, 0.0)
        runGame()
        pygame.mixer.music.stop()
        showTextScreen('Over :(')
```

```
# 게임을 실행하는 실제 코드로 게임을 실행하기 전에 모두 초기화한다.
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)

# 떨어지는 블록과 다음 블록을 나타내준다
    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

        checkForQuit()
# 이벤트 처리 루프로 블록을 회전시키고 이동하고 게임을 멈출 때의 이벤트를 처리한다.
        for event in pygame.event.get(): # event handling loop
            if event.type == KEYUP:
# p를 누르면 게임을 잠시 멈춘다
                if (event.key == K_p):
                    # Pausing the game
                    DISPLAYSURF.fill(BGCOLOR)     # Get a rest!를 출력하며 화면을 가리고 음악을 멈춘다
                    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()
# 블록을 회전시킨다
                # 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']])
# q를 누르면 반대 방향으로 돌린다.
                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()
# 스페이스 키를 누르면 바로 아래로 떨어진다
                    # 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


        # 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()


# 피스가 자연스럽게 떨어지도록 만든다
        # 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()


# 화면에 score, level 보드 등을 그린다.
        # 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)


# 텍스트를 만드는 단축 함수이다
def makeTextObjs(text, font, color):
    surf = font.render(text, True, color)
    return surf, surf.get_rect()


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


# checkForKeyPress()에서 키 눌림 이벤트가 발생햇는지 기다린다
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


# 텍스트 스크린 함수이다
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! 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()


# Esc 키를 눌러서 게임을 종료하여야한다
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


# 점수에 따라 레벨을 계산하고 블록이 내려오는 속도를 조절한다.
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

# 무작위로 피스를 만든다
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': PIECES_COLOR[shape]}       # 앞서 지정해준 모양별 색상이다
    return newPiece


# 보드 데이터 구조에 피스를 추가한다
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']


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

```
# 파라미터로 넘겨준 x, y좌표가 보드에 있는지 검사한다
def isOnBoard(x, y):
    return x >= 0 and x < BOARDWIDTH and y < BOARDHEIGHT


# 블록의 x, y좌표를 받아서 블록의 데이터 구조 내에서 좌표를 더해 보고 결정한다.
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

# 블록으로 채워진 것이 한줄이 되었는지 검사한다
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


# 완성된 줄을 없애고 나머지 블록들을 아래로 한 칸씩 내려준다
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
```

```
# 게시판 좌표계를 픽셀 좌표계로 전환한다
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))


# 픽셀 좌표계에 블록을 그린다
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))


# 스크린에 보드 주변 테두리와 배경색을 채운다
```

```
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])


# 보드에 점수와 레벨 진행된 시간을 넣는다
def drawStatus(score, level):
    #draw the play time
    playtimeSurf = BASICFONT.render('Play Time: %d sec' % (time.time() - PLAYTIME), True, TEXTCOLOR)
    playtimeRect = playtimeSurf.get_rect()
    playtimeRect.topleft = (WINDOWWIDTH - 600, 20)
    DISPLAYSURF.blit(playtimeSurf, playtimeRect)

    # 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)


# 떨어지는 블록과 다음에 나올 블록을 그릴때 사용한다
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))
```

```
# 다음 피스를 그린다
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)



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