오픈소스 보고서

2020051085_여인범

GitHub repository : https://github.com/yib5518/osw

1. 조건 1~6에 맞춰 코드 수정하기

•조건1: 현재 테트리스 게임의 배경음악을 주어진 3개의 음악 중 1개가 재생되도록 수정

```
while True:
        if random.randint(0, 1) == 0:
            # 현재 테트리스 게임의 배경음악을 주어진 3개의 음악 중 1개가 재생되도록
수정     1번
            pygame.mixer.music.load('Platform_9.mp3')
        else:
            pygame.mixer.music.load('Platform_9.mp3')
        pygame.mixer.music.play(-1, 0.0)
        runGame()
        pygame.mixer.music.stop()
        showTextScreen('Game Over')
```

•조건2: 상태창 이름을 학번_이름 으로 수정

•조건3: 게임시작화면의 문구를 MY TETRIS으로 변경

```
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('2020051085_여인범') # 상태창 이름을 학번_이름
으로 수정     2번

    showTextScreen('MY TETRIS') # 게임시작화면의 문구를 MY TETRIS으로
변경      3번
```

•조건4: 게임시작화면의 문구 및 배경색을 노란색으로 변경

```
TEXTCOLOR = YELLOW
TEXTSHADOWCOLOR = YELLOW
COLORS      = (     BLUE,      GREEN,      RED,      YELLOW)
LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYELLOW)
assert len(COLORS) == len(LIGHTCOLORS)
```

TEXTCOLOR을 노란색으로 설정해서 TEXTCOLOR을 참조하고 있는 문구와 배경색들을 노란색으로 설정

•조건5: 게임 경과 시간을 초 단위로 표시

```
def runGame():
    board = getBlankBoard()
    lastMoveDownTime = time.time()
    lastMoveSidewaysTime = time.time()
    lastFallTime = time.time()
    startTime = time.time() # 시작 시간 기록으로 새 게임 시작시 0 으로
초기화    5 번
    movingDown = False
    movingLeft = False
    movingRight = False
    score = 0
    level, fallFreq = calculateLevelAndFallFreq(score)
```

중간 코드 생략

```
        drawBoard(board)
        # 게임 경과 시간을 초 단위로 표시   5 번
        drawStatus(score, level, time.time() - startTime)
        drawNextPiece(nextPiece)
        if fallingPiece != None:
            drawPiece(fallingPiece)

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

중간 코드 생략

```
def drawStatus(score, level, elapsedTime):
    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)
    # 게임 경과 시간을 초 단위로 표시 5 번
    elapsedTimeSurf = BASICFONT.render('Time: %s' % formatTime(elapsedTime),
True, TEXTCOLOR)
    elapsedTimeRect = elapsedTimeSurf.get_rect()
    elapsedTimeRect.topleft = (20, 20)
    DISPLAYSURF.blit(elapsedTimeSurf, elapsedTimeRect)
def formatTime(seconds):
    minutes = int(seconds // 60)
    seconds = int(seconds % 60)
    return f'{minutes:02}:{seconds:02}'
```

formatTime이라는 함수를 생성해서 함수를 분,초에 해당하는 변수 생성.

drawStatus함수 안에 게임 시작후 지난 시간을 표시하게 함

•조건6: 7개의 블록이 각각 고유의 색을 갖도록 코드를 수정하거나 추가

```
# 색 지정
SHAPE_COLORS = {     # 6 번을 해결하기 위함
    'S': BLUE,       # S 블록을 BLUE 로 변경
    'Z': GREEN,      # Z 블록을 GREEN 으로 변경
    'J': RED,        # J 블록을 RED 로 변경
    'L': YELLOW,     # L 블록을 YELLOW 로 변경
    'I': LIGHTGREEN, # I 블록을 LIGHTGREEN 으로 변경
    'O': LIGHTRED,   # O 블록을 LIGHTRED 로 변경
    'T': LIGHTBLUE   # T 블록을 LIGHTBLUE 로 변경
}
```

SHAPE_COLORS딕셔너리에 각 모양마다 색을 지정.

```
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,
                'color': SHAPE_COLORS[shape]} # 7 개의 블록이 각각 고유의 색을
갖도록 코드를 수정하거나 추가 6 번
    return newPiece
```

getNewPiece안에 color값을 수정

```
# 7 개의 블록이 각각 고유의 색을 갖도록 코드를 수정하거나 추가   6 번
def drawBox(boxx, boxy, color, pixelx=None, pixely=None):
    if color == BLANK:
        return
    if pixelx == None and pixely == None:
        pixelx, pixely = convertToPixelCoords(boxx, boxy)

    # Check if the color is in COLORS list 6 번
    if color in COLORS:
        light_color = LIGHTCOLORS[COLORS.index(color)]
    else:
        # Set a default color or handle the error   6 번
        light_color = (255, 255, 255)  # white color as default

    pygame.draw.rect(DISPLAYSURF, color, (pixelx + 1, pixely + 1, BOXSIZE - 1,
BOXSIZE - 1))
    pygame.draw.rect(DISPLAYSURF, light_color, (pixelx + 1, pixely + 1,
BOXSIZE - 4, BOXSIZE - 4))
```

drawBox함수 안에 조건문을 추가하여 주어진 색상이 COLORS에 있으면, 그 색상의 밝은

버전을 LIGHTCOLORS에서 찾아 설정. 그 외에는 흰색을 설정. 그리고 pygame.draw.rect의 color값을 각각 color과 light_color로 바꿔 박스에 3d효과 부여

이상으로 조건 6개를 어떻게 바꿨는지 설명하였고 이제 각 함수의 역할을 설명하도록 하겠다.

2. 각 함수의 역할 설명

■ main 함수 전 부분

```
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)

BORDERCOLOR = BLUE
BGCOLOR = BLACK

# 게임시작화면의 문구 및 배경색을 노란색으로 변경   4번
TEXTCOLOR = YELLOW
TEXTSHADOWCOLOR = YELLOW
COLORS      = (     BLUE,      GREEN,      RED,      YELLOW)
```

```
LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYELLOW)
assert len(COLORS) == len(LIGHTCOLORS)

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..',
```

```
                    '.....'],
                   ['.....',
                    '.....',
                    '.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}

# 색 지정
SHAPE_COLORS = {     # 6번을 해결하기 위함
    'S': BLUE,       # S 블록을 BLUE로 변경
    'Z': GREEN,      # Z 블록을 GREEN으로 변경
    'J': RED,        # J 블록을 RED로 변경
    'L': YELLOW,     # L 블록을 YELLOW로 변경
    'I': LIGHTGREEN, # I 블록을 LIGHTGREEN으로 변경
    'O': LIGHTRED,   # O 블록을 LIGHTRED로 변경
    'T': LIGHTBLUE   # T 블록을 LIGHTBLUE로 변경
}
```

Main 함수 진입 전에 게임에 필요한 다양한 상수, 색상,화면 크기, 프레임 속도, 박스 크기 , 블록 모양 등 많은 것을 정의하는 부분이다.

```
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('2020051085_여인범') # 상태창 이름을 학번_이름
으로 수정     2번

    showTextScreen('MY TETRIS') # 게임시작화면의 문구를 MY TETRIS으로
변경      3번
    while True:
        if random.randint(0, 1) == 0:
```

```
            # 현재 테트리스 게임의 배경음악을 주어진 3 개의 음악 중 1 개가 재생되도록
수정     1 번
            pygame.mixer.music.load('Platform_9.mp3')
        else:
            pygame.mixer.music.load('Platform_9.mp3')
        pygame.mixer.music.play(-1, 0.0)
        runGame()
        pygame.mixer.music.stop()
        showTextScreen('Game Over')
```

다음으로는 코드를 실행하면 가장 먼저 돌아가는 main함수이다. global키워드를 써서 지역변수들을 전역변수로 선언. Pygame 라이브러리를 초기화, MY TETRIS로 게임 시작 화면을 표시, 배경음악 재생, runGame()함수 호출.

```
def runGame():
    board = getBlankBoard()
    lastMoveDownTime = time.time()
    lastMoveSidewaysTime = time.time()
    lastFallTime = time.time()
    startTime = time.time() # 시작 시간 기록으로 새 게임 시작시 0 으로
초기화    5 번
    movingDown = False
    movingLeft = False
    movingRight = False
    score = 0
    level, fallFreq = calculateLevelAndFallFreq(score)

    fallingPiece = getNewPiece()
    nextPiece = getNewPiece()

    while True:
        if fallingPiece == None:
            fallingPiece = nextPiece
            nextPiece = getNewPiece()
            lastFallTime = time.time()

            if not isValidPosition(board, fallingPiece):
                return

        checkForQuit()
        for event in pygame.event.get():
            if event.type == KEYUP:
                if (event.key == K_p):
                    # Pausing the game
                    DISPLAYSURF.fill(BGCOLOR)
                    pygame.mixer.music.stop()
                    showTextScreen('Paused')
                    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:
                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()

                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):
                    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_DOWN or event.key == K_s):
                    movingDown = True
                    if isValidPosition(board, fallingPiece, adjY=1):
                        fallingPiece['y'] += 1
                    lastMoveDownTime = time.time()

                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

        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()

        if time.time() - lastFallTime > fallFreq:
            if not isValidPosition(board, fallingPiece, adjY=1):
                addToBoard(board, fallingPiece)
                score += removeCompleteLines(board)
                level, fallFreq = calculateLevelAndFallFreq(score)
                fallingPiece = None
            else:
                fallingPiece['y'] += 1
                lastFallTime = time.time()

        DISPLAYSURF.fill(BGCOLOR)
        drawBoard(board)
        # 게임 경과 시간을 초 단위로 표시   5번
        drawStatus(score, level, time.time() - startTime)
        drawNextPiece(nextPiece)
        if fallingPiece != None:
            drawPiece(fallingPiece)

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

runGame()은 게임의 중요한 로직들을 처리한다. 보드, 시간 관련 변수, 블록을 이동시키는 변수, 점수, 레벨등을 초기화하고 블록들을 생성, 게임이 종료될 때까지 블록의 하,좌,우 이동과 회전, 점수 계산등을 처리

```
def drawStatus(score, level, elapsedTime):
    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)
    # 게임 경과 시간을 초 단위로 표시 5 번
    elapsedTimeSurf = BASICFONT.render('Time: %s' % formatTime(elapsedTime),
True, TEXTCOLOR)
    elapsedTimeRect = elapsedTimeSurf.get_rect()
    elapsedTimeRect.topleft = (20, 20)
    DISPLAYSURF.blit(elapsedTimeSurf, elapsedTimeRect)
```

drawStatus()는 화면에 점수, 레벨 경과 시간을 렌더링한다.

```
def formatTime(seconds):
    minutes = int(seconds // 60)
    seconds = int(seconds % 60)
    return f'{minutes:02}:{seconds:02}'
```

formatTime()는 시간을 분과 초로 변환하여 String값으로 반환.

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

makeTextObjs()는 주어진 텍스트들을 지정된 폰트와 색상으로 렌더링.

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

terminate()는 게임을 종료시키는 함수이다.

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

checkForKeyPress()는 checkForQuit함수를 호출하여 게임을 종료하는 상황이 발생했는지 확인하고 종료. 전반적인 키 입력을 감지하고 이벤트가 발생했을때의 코드 반환.

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

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

```
    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()
```

showTextScreen()은 텍스트 화면을 표시하는 함수이다. 주어진 텍스트를 화면 중앙에 배치하고 게임을 멈췄을 때 "Press a key to play."문장을 생성. 사용자가 다시 키를 입력할때까지 대기.

```
def checkForQuit():
    for event in pygame.event.get(QUIT):
        terminate()
    for event in pygame.event.get(KEYUP):
        if event.key == K_ESCAPE:
            terminate()
        pygame.event.post(event)
```

checkForQuit()는 게임을 종료시키는 이벤트가 일어났을 때 게임을 종료시키는 함수이다.

```
def calculateLevelAndFallFreq(score):
    level = int(score / 10) + 1
    fallFreq = 0.27 - (level * 0.02)
    return level, fallFreq
```

calcilateLevelAndFallFreq()는 점수에 따라 레벨과 블록 낙하 시간을 짧게 만들어 게임을 더 어렵게 한다.

```
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,
                'color': SHAPE_COLORS[shape]} # 7 개의 블록이 각각 고유의 색을
갖도록 코드를 수정하거나 추가 6 번
    return newPiece
```

getNewPiece()는 새로운 블록을 생성하고 블록의 모양, 회전 상태, 블록의 초기 x좌표와 y좌표를 설정하고 색상을 부여하는 역할을 한다.

```
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']
```

addToBoard함수는 현재 떨어지는 블록을 게임 보드에 추가하는 역할을 한다. 보드의 x좌표

와 y좌표를 계산하여 해당하는 위치의 보드를 블록의 색상으로 갱신한다.

```
def getBlankBoard():
    board = []
    for i in range(BOARDWIDTH):
        board.append([BLANK] * BOARDHEIGHT)
    return board
```

getBlankBoard()는 빈 보드를 초기화하고 보드의 각 역을 생성하고 최종적으로 게임에 사용되는 비어있는 보드를 반환한다.

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

isOnBoard()는 x,y좌표가 게임 보드 안에 있는지 확인.

```
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:
                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
```

isValidPosition()는 현재 블록이 보드 경계를 벗어나지 않고 다른 블록과 겹치지 않는지를 확인. 모든 조건을 만족하면 True값 반환.

```
def isCompleteLine(board, y):
    for x in range(BOARDWIDTH):
        if board[x][y] == BLANK:
            return False
    return True
```

isCompleteLine()함수는 주어진 행들이 빈칸없이 블록으로 가득 찼는지 검사. 빈칸이 하나라도 있으면 False값 반환

```
def removeCompleteLines(board):
    numLinesRemoved = 0
    y = BOARDHEIGHT - 1
    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
    return numLinesRemoved
```

removeCompleteLines()함수는 블록으로 가득 찬 행을 제거하고 위의 행을 아래로 당기며 제거된 행의 수를 반환한다.

```
def convertToPixelCoords(boxx, boxy):
    return (XMARGIN + (boxx * BOXSIZE)), (TOPMARGIN + (boxy * BOXSIZE))
```

convertToPixelCoords()는 보드의 x,y좌표를 픽셀 좌표로 변환 후 반환.

```
def drawBox(boxx, boxy, color, pixelx=None, pixely=None):

    if color == BLANK:
        return
    if pixelx == None and pixely == None:
        pixelx, pixely = convertToPixelCoords(boxx, boxy)

    # Check if the color is in COLORS list 6번
    if color in COLORS:
        light_color = LIGHTCOLORS[COLORS.index(color)]
    else:
        # Set a default color or handle the error   6번
        light_color = (255, 255, 255)  # white color as default

    pygame.draw.rect(DISPLAYSURF, color, (pixelx + 1, pixely + 1, BOXSIZE - 1,
BOXSIZE - 1))
    pygame.draw.rect(DISPLAYSURF, light_color, (pixelx + 1, pixely + 1,
BOXSIZE - 4, BOXSIZE - 4))
```

drawBox()는 보드에서 하나의 블록을 만드는 역할. 블록의 x,y좌표, 색상, 블록을 그릴 픽셀 좌표를 계산.

```
def drawBoard(board):
    pygame.draw.rect(DISPLAYSURF, BORDERCOLOR, (XMARGIN - 3, TOPMARGIN - 7,
(BOARDWIDTH * BOXSIZE) + 8, (BOARDHEIGHT * BOXSIZE) + 8), 5)
    pygame.draw.rect(DISPLAYSURF, BGCOLOR, (XMARGIN, TOPMARGIN, BOXSIZE *
BOARDWIDTH, BOXSIZE * BOARDHEIGHT))
    for x in range(BOARDWIDTH):
        for y in range(BOARDHEIGHT):
            drawBox(x, y, board[x][y])
```

drawboard()는 보드를 그리는 역할. 보드의 테두리, 배경, 보드의 각 블록을 그림.

```
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))
```

drawPiece()는 piece딕셔너리에서 해당 모양의 블록을 가져오고 pixelx,pixely가 모두 None이면 픽셀 좌표를 계산. drawBox함수를 호출하여 해당 위치에 블록을 그림. 주어진 테트리스 블록의 모양을 그리는 역할

```
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)
```

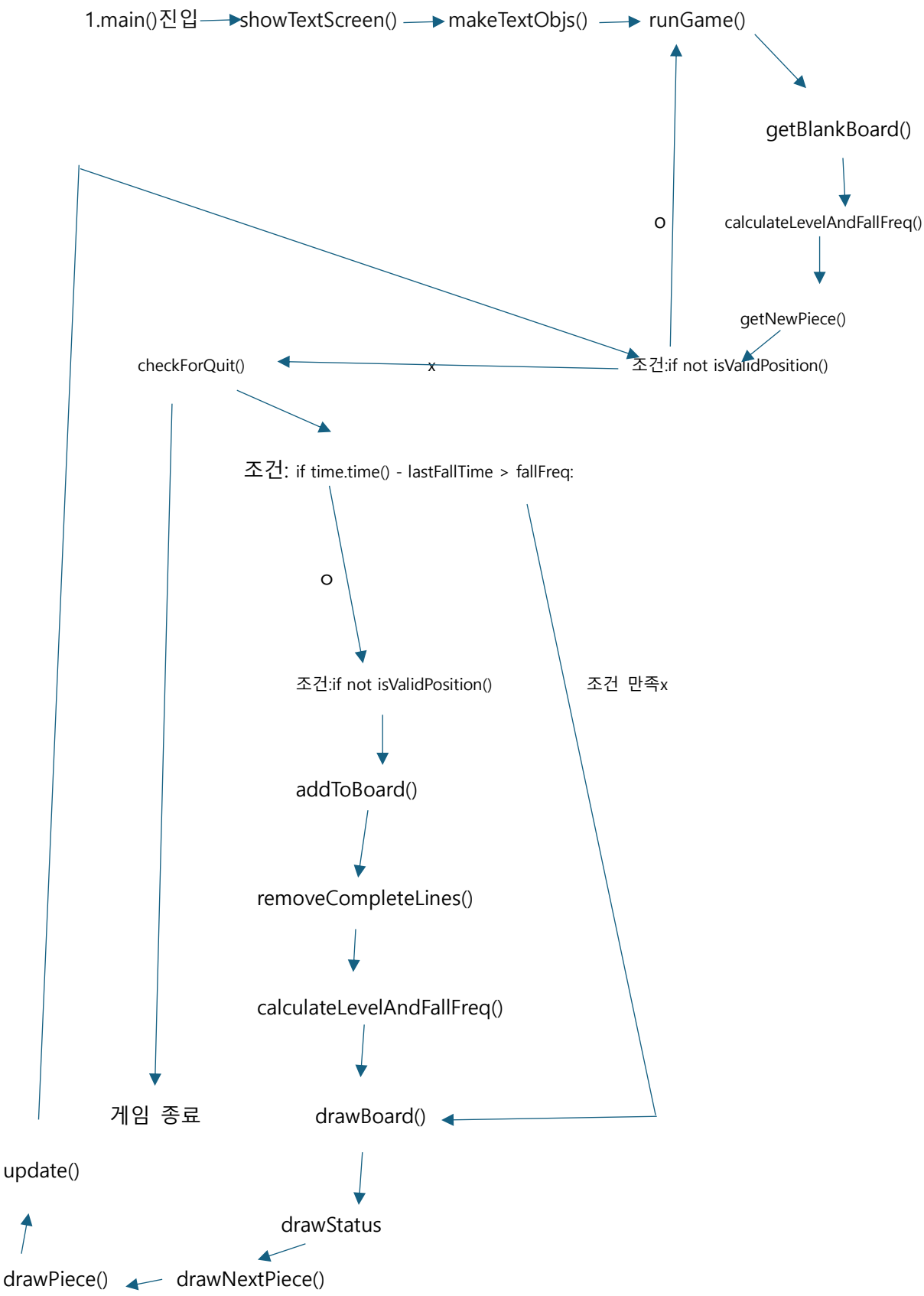
drawNextPoece()는 다음에 나올 테트리스 블록을 화면에 그림. drawPiece()를 호출하여 픽셀 좌표값에 다음에 나올 블록을 그림.

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

이 코드는 메인 함수의 시작을 의미. 해당 모듈이 import된 경우가 아닌 interpreter에서 직접 실행된 경우에 main()호출

3. 함수의 호출 순서 및 조건

   main()부터 시작. 이후 showTextScreen함수 호출 후 showTextScreen함수 안에 있는 makeTextObjs함수가 게임시작 문구를 만든다. 이후 main함수가 rungame함수를 실행하고 runGame()내의 getBlackBoard(), calculateLevelAndFallFreq(), getNewPiece()들이 게임을 위한 기저를 만들고 while True:로 반복문을 실행한다 반복문을 탈출하는 조건이 if not isValidPosition()이며 not isValidPostion()에 해당하면 runGame()을 끝내고 게임 오버 텍스트를 화면에 표시하며 이후에 반복문을 통해 게임을 다시 실행함. 만약 isValidPosition()이라면 checkForQuit()을 통해 종료 키가 입력되었는지를 확인하고 입력되었으면 종료, 아니면 계속 진행한다. 그 후 조건: if time.time() - lastFallTime > fallFreq을 만족하면 addToBoard(), removeCompleteLines(), calculateLevelAndFallFreq()를 실행하고 drawboard(), drawStatus(), drawNextPiece(), drawPiece(), update()를 통해 게임을 실행한다. 이후에는 반복문으로 인해 다시 조건문으로 가서 게임을 계속 실행한다.


1.main()진입
showTextScreen()
makeTextObjs()
runGame()
getBlankBoard()
calculateLevelAndFallFreq()
getNewPiece()
조건:if not isValidPosition()
o
x
checkForQuit()
조건: if time.time() - lastFallTime > fallFreq:
o
조건:if not isValidPosition()
조건 만족x
addToBoard()
removeCompleteLines()
calculateLevelAndFallFreq()
게임 종료
drawBoard()
update()
drawStatus
drawPiece()
drawNextPiece()