2021059834 이현우 실습 과제 (보고서) https://github.com/LeeHyunWoo02/osw

▼ 체크리스트 설명

▼ 1. 현재 테트리스 게임의 배경음악을 주어진 3개의 음악 중 1개가 재생되도록 수정하라

```
def main():
        ## 중간 생략 ##
    while True: # game loop
        if random.randint(0, 1) == 0:
            pygame.mixer.music.load('Hover.mp3') # 노래
        else:
            pygame.mixer.music.load('Hover.mp3')
        ## 중간 생략 ##
```

- main( ) 함수의 whlie 반복문에 있는 pygame.mixer.music.load 코드를 수정 함.

▼ 2. 상태창 이름을 학번_이름으로 수정

```
def main():
   ## 중간 생략 ##

   pygame.display.set_caption('2021059834 이현우') # 상단바

   ## 중간 생략 ##
```

- main( ) 함수의 pygame.display.set_caption( ) 을 수정함.
- pygame.display.set_caption( ) : 타이틀의 텍스트를 결정하는 함수

▼ 3. 게임시작화면의 문구를 “ MY TETRIS “로 변경

```
def main():
                ## 코드 중간 생략 ##
```

```
    showTextScreen('MY TETRIS') # 게임 시작화면 문구 변경

                ## 코드 생략 ##
```

- main( ) 함수의 shoTextScreen( ) 을 수정

▼ 4. 게임시작화면의 문구 및 배경색을 노란색으로 변경

```
## 코드 생략##

BORDERCOLOR = BLUE
BGCOLOR = BLACK
TEXTCOLOR = YELLOW # 게임시작화면의 문구 노란색으로 변경
TEXTSHADOWCOLOR = YELLOW # 게임시작화면 문구의 배경색을 노란색

## 코드 생략 ##
```

- TEXTCOLOR 변수를 수정함으로써 시작화면의 텍스트 색을 변경
- TEXTSHADOWCOLOR 변수를 YELLOW로 수정함으로써 텍스트의 배경색을 노란색으로 변경

▼ 5. 게임 경과 시간을 초 단위로 표시 (새 게임 시작시 0으로 초기화 되어야 함)

```
def runGame():
                ## 코드 생략 ##
    startTime = time.time() # 게임 시작 시간 기록 추가시킴

                ## 코드 생략 ##

    while True: # game loop

                ## 중간 코드 생략 ##


        elapsedTime = time.time() - startTime
        # 게임 경과 시간 계산 - 초 단위
```

```
        drawElapsedTime(elapsedTime)
                    # 함수 호출하여 화면에 표시함

                        ## 중간 코드 생략 ##
```

- runGame( ) 함수 내에 게임 시작 시간을 기록할 변수 startTime을 추가시켜 time.time()을 활용하여 초기화 시킴
- gameloop 내에 elapsedTime 이라는 변수를 만들어 게임 경과 시간을 초 단위로 계산 한 값을 저장함.

  ```
  def drawElapsedTime(elapsedTime): # 경과 시간을 화면에 표
      elapsedTimeSurf = BASICFONT.render('Time: %s' % 
      elapsedTimeRect = elapsedTimeSurf.get_rect()
      elapsedTimeRect.topright = (WINDOWWIDTH - 180, 2
      DISPLAYSURF.blit(elapsedTimeSurf, elapsedTimeRec
  ```

  -> 게임경과시간을 화면에 표시 시키는 함수 추가 작

- drawElapsedTime 함수를 호출하여 화면에 표시 시킴

▼ 6. 7개의 블록이 각각 고유의 색을 갖도록 코드를 수정 하거나 추가

```
#                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)
ORANGE      = (255, 165,   0)
LIGHTORANGE = (255, 200,  40)
```

```
PURPLE      = (128,   0, 128)
LIGHTPURPLE = (160,  32, 240)
CYAN        = (  0, 255, 255)
LIGHTCYAN   = ( 32, 255, 255)
# 색을 더 추가함
# because 7개의 블록 각각에 고유한 색상을 부여하기 위해서임. 기존

COLORS      = (BLUE, GREEN, RED, YELLOW, ORANGE, PURPLE
LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYE
assert len(COLORS) == len(LIGHTCOLORS)
```

- 7개의 블록 각각의 고유한 색상을 부여하기 위해서는 기존 4개의 색상만으로는 부족하기 때문에 색을 더 추가 시킴.

```
Shape_color_index = {shape : i for i, shape in enumerat
# 이걸 작성함으로써 각 모양을 키 그에 맞는 인덱스를 값으로 구성 된
# 이렇게 만들어두면 getNewPiece 함수에서 색 설정을 모양마다 설정
def getNewPiece():
    # 새로운 모양을 랜덤으로 생성하고, 색을 랜덤으로 지정해줌.
    shape = random.choice(list(PIECES.keys()))

    newPiece = {'shape': shape,
                'rotation': random.randint(0, len(PIECE
                'x': int(BOARDWIDTH / 2) - int(TEMPLATE
                'y': -2, # start it above the board (i.
                'color': Shape_color_index[shape] } # 색
    return newPiece
```

- Shape_color_index : 각 모양을 키로 설정, 각 모양에 맞는 인덱스를 값으로 구성 된 딕셔너리를 추가함으로써 getNewPiece()에서 모양마다 색 부여를 할 수 있음.
- newPiece에 값을 부여할 때 color : hape_color_index[shape] 함으로써 색상 인덱스를 설정 함.
    - 예를 들어 Shape가 T면 6이 설정 됨.

▼ 함수 분석

▼ main( )

```
def main():
    global FPSCLOCK, DISPLAYSURF, BASICFONT, BIGFONT
    pygame.init() # Pygame 모듈 초기화
    FPSCLOCK = pygame.time.Clock()
        # 게임의 프레임 속도를 제어하는 시계 객체를 생성
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH,
        # 게임의 메인 디스플레이를 설정
    BASICFONT = pygame.font.Font('freesansbold.ttf', 18
    BIGFONT = pygame.font.Font('freesansbold.ttf', 100)
    pygame.display.set_caption('2021059834 이현우') # 상단

    showTextScreen('MY TETRIS') # 게임 시작화면 문구 변경
    while True: # game loop
        if random.randint(0, 1) == 0:
            pygame.mixer.music.load('Hover.mp3') # 노래
        else:
            pygame.mixer.music.load('Hover.mp3')
        pygame.mixer.music.play(-1, 0.0) # 음악을 무한 반복
        runGame()
        pygame.mixer.music.stop() # 게임이 종료 되면 음악 중
        showTextScreen('Game Over')
```

→ main 함수는 게임의 초기 설정을 수행하고 게임 루프를 실행하는 역할을 함

▼ runGame()

```
def runGame():
     # 여기까지 초기 설정
    board = getBlankBoard() #빈 게임 보드 설정
    lastMoveDownTime = time.time() # 타이머 초기화
    lastMoveSidewaysTime = time.time() # 타이머 초기화
    lastFallTime = time.time() # 타이머 초기화
    movingDown = False # 조각의 이동 상태 초기화
```

```
    movingLeft = False
    movingRight = False
    score = 0 # 점수 0으로 초기화
    level, fallFreq = calculateLevelAndFallFreq(score)
        # 현재 레벨과 조각이 떨어지는 빈도를 계산

    fallingPiece = getNewPiece() # 현재 떨어지고 있는 조각 생
    nextPiece = getNewPiece() # 다음 조각 생성



    while True: # 게임 루프
        if fallingPiece == None:
      # 만약에 떨어지고 있는 조각이 없으면
            nextPiece = getNewPiece()  # 새로운 조각을 시작
            lastFallTime = time.time() # 마지막 떨어진 시간

            if not isValidPosition(board, fallingPiece)
                return # 새 조각이 보드에 맞지 않으면 게임료
        checkForQuit()
        for event in pygame.event.get(): # 이벤트 처리
            if event.type == KEYUP: # 키를 뗏을 때의 처리 코
                if (event.key == K_p):
                    # Pausing the game
                    DISPLAYSURF.fill(BGCOLOR)
                    pygame.mixer.music.stop()
                    showTextScreen('Paused') # pause un
                    pygame.mixer.music.play(-1, 0.0)
                    lastFallTime = time.time()
                    lastMoveDownTime = time.time()
                    lastMoveSidewaysTime = time.time()
                elif (event.key == K_LEFT or event.key =
                    movingLeft = False
                elif (event.key == K_RIGHT or event.key
                    movingRight = False
                elif (event.key == K_DOWN or event.key =
                    movingDown = False
```

```
            elif event.type == KEYDOWN:
                # 키를 눌렀을 때의 처리 코드
                if (event.key == K_LEFT or event.key ==
                    fallingPiece['x'] -= 1
                    movingLeft = True
                    movingRight = False
                    lastMoveSidewaysTime = time.time()

                elif (event.key == K_RIGHT or event.key
                    fallingPiece['x'] += 1
                    movingRight = True
                    movingLeft = False
                    lastMoveSidewaysTime = time.time()

                # rotating the piece (if there is room
                elif (event.key == K_UP or event.key ==
                    fallingPiece['rotation'] = (falling
                    if not isValidPosition(board, falli
                        fallingPiece['rotation'] = (fal
                elif (event.key == K_q): # rotate the o
                    fallingPiece['rotation'] = (falling
                    if not isValidPosition(board, falli
                        fallingPiece['rotation'] = (fal

                # making the piece fall faster with the
                elif (event.key == K_DOWN or event.key =
                    movingDown = True
                    if isValidPosition(board, fallingPi
                        fallingPiece['y'] += 1
                    lastMoveDownTime = time.time()

                # move the current piece all the way do
                elif event.key == K_SPACE:
                    movingDown = False
                    movingLeft = False
                    movingRight = False
                    for i in range(1, BOARDHEIGHT):
                        if not isValidPosition(board, f
```

```
                        break
                fallingPiece['y'] += i - 1

        # 족가 이동 처리
        if (movingLeft or movingRight) and time.time()
                #왼쪽 또는 오른쪽으로 이동
            if movingLeft and isValidPosition(board, fa
                fallingPiece['x'] -= 1
            elif movingRight and isValidPosition(board,
                fallingPiece['x'] += 1
            lastMoveSidewaysTime = time.time()

        if movingDown and time.time() - lastMoveDownTim
               # 아래로 이동
            fallingPiece['y'] += 1
            lastMoveDownTime = time.time()

        # 조각 떨어뜨리기
        if time.time() - lastFallTime > fallFreq:
            # see if the piece has landed
            if not isValidPosition(board, fallingPiece,
                addToBoard(board, fallingPiece)
                score += removeCompleteLines(board)
                level, fallFreq = calculateLevelAndFall
                fallingPiece = None
            else:
                fallingPiece['y'] += 1
                lastFallTime = time.time()
                        # 조각이 자동으로 떨어지는 타이밍을 체

        #화면 그리기
        DISPLAYSURF.fill(BGCOLOR)
        drawBoard(board)
        drawStatus(score, level)
        drawNextPiece(nextPiece)
        #게임 보드, 점수, 레벨, 다음조각을 화면에 그림
        if fallingPiece != None:
            drawPiece(fallingPiece)
```

```
            # 현재 떨어지고 있는 조각을 화면에 그림
        pygame.display.update()
        FPSCLOCK.tick(FPS)
```

→ 전체 게임의 흐름을 관리하며, 사용자 입력, 조각의 이동 및 회전, 점수 계산, 화면그리기 등 게임의 주요 기능을 수행함.

→ 게임의 메인 루프를 실행하기 위해 실행

▼ makeTextObjs

```
def makeTextObjs(text, font, color):
    surf = font.render(text, True, color) # 텍스트를 렌더링
    return surf, surf.get_rect()
    # surf => 렌더링된 텍스트가 포함된 표면 객체
    # surf.get_rect() => 표면 객체의 사각현 객체
```

→ 주어진 텍스트를 특정 폰트와 색상으로 렌더링하고, 그 결과로 생성된 표면 객체와 그 표면 객체의 사각형 객체를 반환하는 함수.

▼ terminate()

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

▼ checkForKeyPress()

```
def checkForKeyPress():

    checkForQuit()
    # 게임을 종료하는 이벤트 ( 예: 창 닫기 버튼 클릭 )

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

→ 키보드 입력을 확인하고, 키가 눌렸다고 올라오는 이벤트가 발생했을 때 해당 키를 반환하는 함수.

▼ showTextScreen()

```
def showTextScreen(text):
        #텍스트 그림자 그리기
    titleSurf, titleRect = makeTextObjs(text, BIGFONT, 
    titleRect.center = (int(WINDOWWIDTH / 2), int(WINDO
    DISPLAYSURF.blit(titleSurf, titleRect)

        # 텍스트 그리기
    titleSurf, titleRect = makeTextObjs(text, BIGFONT, 
    titleRect.center = (int(WINDOWWIDTH / 2) - 3, int(W
    # 실제 텍스트를 그림자 텍스트 보다 약간 위쪽에 위치시킴.
    DISPLAYSURF.blit(titleSurf, titleRect)

    # 추가 텍스트 그리기 " Press a ~~ "
    pressKeySurf, pressKeyRect = makeTextObjs('Press a 
    pressKeyRect.center = (int(WINDOWWIDTH / 2), int(WI
    DISPLAYSURF.blit(pressKeySurf, pressKeyRect)

    while checkForKeyPress() == None: # 키 입력 대기
        pygame.display.update() # 키 입력이 있을 때까지 화면
        FPSCLOCK.tick()
```

→ 화면 중앙에 큰 텍스트를 표시하고, 키 입력이 있을 때까지 해당 텍스트를 유지하는 기능.

→ 게임 시작 화면이나 일시 정지 화면 등에 유용하게 사용 가능

▼ calculateLevelAndFallFreq(score)

```
def calculateLevelAndFallFreq(score):
    level = int(score / 10) + 1 # 레벨 계산
    fallFreq = 0.27 - (level * 0.02) # 기본 낙하주기 + 0.2
```

```
    # level이 1 올라갈 때마다 주기를 0.02초씩 줄임
    return level, fallFreq
```

→ 게임 점수에 따라 플레이어의 레벨과 블록이 한 칸 떨어지는데 걸리는 시간을 계산하여 반환하는 함수.

→ 레벨에 따라 블록이 떨어지는 속도를 조절하여 게임의 난이도를 동적으로 조정하는 역할

▼ getNewPiece( )

```
def getNewPiece():

    # Shape_color_index = {shape: i for i, shape in enu
    # 각 모양에 대해 고유한 인덱스를 할당하여 딕셔너리 생성

    shape = random.choice(list(PIECES.keys()))
    # PIECES 딕셔너리의 키 중 하나를 무작위로 선택하여 shape에 
    newPiece = {'shape': shape,
                'rotation': random.randint(0, len(PIECE
                'x': int(BOARDWIDTH / 2) - int(TEMPLATE
                'y': -2, # start it above the board (i.
                'color': random.randint(0, len(COLORS)-
    # 'color': Shape_color_index[shape]}
    # 각 모양마다 고유한 색상 인덱스를 할당 받게 됨.

    # 새로운 조각을 딕셔너리 형태로 생성함.
    # rotation : 해당 모양의 가능한 회전 수 중 하나를 무작위 선
    # x,y : 새로운 조각의 좌표. x는 보드의 중앙 y는 보드위에 위치
    # color : 가능한 색상 중 하나를 무작위로 선택
    return newPiece
```

→ 게임 내에서 새로운 조각을 생성하는 역할

▼ addToBoard(board, piece)

```
def addToBoard(board, piece):

    for x in range(TEMPLATEWIDTH): # 반복문을 통한 조각 탐색
        for y in range(TEMPLATEHEIGHT):
```

```
            if PIECES[piece['shape']][piece['rotation']]
            # 비어있지 않은 셀 확인
                board[x + piece['x']][y + piece['y']] =
                # 보드 업데이트
```

→ 게임 보드 위에 주어진 조각을 놓는 역할

→ 플레이어가 조작하는 조각을 보드에 올바르게 추가함으로써, 게임의 진행 상황을 업데이트하고, 조각을 쌓아 완전한 줄을 만들어 제거 할 수 있게 함.

- 반복문을 통한 탐색
  - 각 매개변수는 조각의 템플릿 크기를 나타냄. 두 변수를 사용하는 반복문을 통해 조각의 모든 셀을 순회함.
- 비어있지 않은 셀 확인
  - 조각의 현재 회전 상태에서 x,y 위치에 있는 셀이 비어있지 않은 경우, 해당 셀을 보드에 추가해야 함.
- 보드 업데이트
  - 조각의 위치를 기준으로 보드의 해당 위치에 조각의 색상을 설정

▼ getBlankBoard( )

```
def getBlankBoard():
    board = [] # 빈 보드 생성
    for i in range(BOARDWIDTH): # 크기만큼 반복하여 각 열을
        board.append([BLANK] * BOARDHEIGHT)
    return board
```

→ 새로운 빈 게임 보드를 생성하고 반환하는 역할을 함.

▼ isOnBoard(x, y)

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

→ 주어진 좌표가 게임 보드의 범위 내에 있는지 여부 확인

▼ isValidPosition(board, piece, adjX = 0, adjY = 0)

```
def isValidPosition(board, piece, adjX=0, adjY=0):
    for x in range(TEMPLATEWIDTH): # 블록의 각 셀을 순회
        for y in range(TEMPLATEHEIGHT):
            isAboveBoard = y + piece['y'] + adjY < 0 # 
            if isAboveBoard or PIECES[piece['shape']][p
                continue # 빈 셀인지 확인
            if not isOnBoard(x + piece['x'] + adjX, y +
                return False #isOnBoard() 현재 셀이 게임 보
            if board[x + piece['x'] + adjX][y + piece['
                return False # 현재 셀의 위치에 다른 블록이 
    return True
```

→ 주어진 조각이 게임 보드 내에 올바르게 위치해 있는지, 다른 블록과 충돌하지 않는지를 확인 함.

→ 플레이어가 블록을 이동하거나 회전시킬 때마다, 이 함수를 사용하여 그러한 동작이 가능한지를 판단할 수 있음.

→ 게임의 규칙을 유지하고, 플레이어가 보드 밖으로 블록을 이동시키거나 다른 블록과 부적절하게 겹치게 하는 것을 방지함

▼ removeCompleteLines(board)

```
def removeCompleteLines(board):

    numLinesRemoved = 0 # 삭제된 줄의 수를 0으로 초기화
    y = BOARDHEIGHT - 1  # 게임 보드의 가장 아래 줄 부터 검사

    while y >= 0:
        if isCompleteLine(board, y): # 함수를 호출하여 현재
            # 이 함수는 현재 줄이 모두 블록으로 채워져 있는지 검사
            for pullDownY in range(y, 0, -1): # for 문을
                for x in range(BOARDWIDTH):
                    board[x][pullDownY] = board[x][pull
            for x in range(BOARDWIDTH):
                board[x][0] = BLANK # 가장 윗줄은 빈줄로 설
            numLinesRemoved += 1 # 줄 삭제 횟수 업데이트
        else:
```

```
            y -= 1
    return numLinesRemoved
```

→ 완성된 줄을 삭제하고 그 위에 있는 줄들을 아래로 당겨오는 역할을 함.

→ 삭제된 줄의 수를 반환

▼ convertToPixelCoords(boxx, boxy)

```
def convertToPixelCoords(boxx, boxy):
        # boxx => 게임 보드의 x 좌표 boxy => 게임 보드의 y 좌
    return (XMARGIN + (boxx * BOXSIZE)), (TOPMARGIN + (
    # XMARGIN + (boxx * BOXSIZE) => 게임 보드의 x 좌표를 픽
    # XMARGIN : 게임 보드가 화면에서 시작 되는 가로 방향의 여백
    # (boxx * BOXSIZE) : 게임 보드 내의 특정 블록이 차지하는 픽

    # TOPMARGIN + (boxy * BOXSIZE) => 게임 보드의 y 좌표를
    # TOPMARGIN : 게임 보드가 화면에서 시작되는 세로 방향의 여백
    # (boxy * BOXSIZE) : 게임 보드 내의 특정 블록이 차지한느 픽
```

→ 게임 보드 상의 좌표 (boxx, boxy)를 화면 상의 픽셀 좌표로 변환하는 역할을

▼ drawBox(boxx, boxy, color, pixelx = None, pixely = None)

```
def drawBox(boxx, boxy, color, pixelx=None, pixely=None
        # color : 블록의 색상
    if color == BLANK:
        return
    if pixelx == None and pixely == None:
        pixelx, pixely = convertToPixelCoords(boxx, box

    pygame.draw.rect(DISPLAYSURF, COLORS[color], (pixel
    pygame.draw.rect(DISPLAYSURF, LIGHTCOLORS[color], (
# rect => 함수를 사용하여 두 개의 사각형을 그림
```

→ 주어진 위치에 특정 색상의 블록을 그리는 역할을 함.

→ color가 Blank인 경우, 함수는 블록을 그리지 않고 바로 반환합니다.

▼ drawStatus(score, level)

```
def drawStatus(score, level):

    scoreSurf = BASICFONT.render('Score: %s' % score, T
    #render 메서드를 사용하여 점수와 레벨을 텍스트로 변환
    scoreRect = scoreSurf.get_rect()
    scoreRect.topleft = (WINDOWWIDTH - 150, 20)
    # 각각의 텍스트가 화면에서 어디에 위치할지를 결정
    DISPLAYSURF.blit(scoreSurf, scoreRect)
        #계산된 위치에 텍스트를 그립니다.

    levelSurf = BASICFONT.render('Level: %s' % level, T
    levelRect = levelSurf.get_rect()
    levelRect.topleft = (WINDOWWIDTH - 150, 50)
    DISPLAYSURF.blit(levelSurf, levelRect)
```

→ 게임의 점수와 레벨을 화면에 표시하는 함수 구현

▼ drawPiece(piece, pixelx = None, pixely = None)

```
def drawPiece(piece, pixelx=None, pixely=None):
    shapeToDraw = PIECES[piece['shape']][piece['rotatio
    if pixelx == None and pixely == None:
        pixelx, pixely = convertToPixelCoords(piece['x'

    for x in range(TEMPLATEWIDTH):
        # Templatewidth : 도형을 구성하는 2D 템플릿의 너비
        for y in range(TEMPLATEHEIGHT): # 높이
            if shapeToDraw[y][x] != BLANK:
                drawBox(None, None, piece['color'], pix
```

→ 특정 도형을 화면에 그리는 함수

▼ drawNextPiece(piece)

```
def drawNextPiece(piece):
        # Next : 텍스트 그리기
    nextSurf = BASICFONT.render('Next:', True, TEXTCOLO
    nextRect = nextSurf.get_rect()
    nextRect.topleft = (WINDOWWIDTH - 120, 80)
    DISPLAYSURF.blit(nextSurf, nextRect)

        #
    drawPiece(piece, pixelx=WINDOWWIDTH-120, pixely=100
```

→ 다음에 나올 도형을 화면에 표시하는 역할

- Next : 텍스트 그리기
  - **BASICFONT.render('Next:', True, TEXTCOLOR) : "Next: "라는 텍스트를 생성함.**

▼ 함수 구현 순서 및 호출 조건

1. **main()**
   a. 프로그램의 시작. 초기 설정을 수행하고 게임을 시작.
   b. 게임이 시작되면 자동으로 호출.
2. **runGame()**
   a. 게임의 메인 루프를 실행하며 게임의 전반적인 흐름을 제어
   b. main( ) 함수 내부에서 게임 루프를 실행 할 때 호출
3. **showTextScreen()**
   a. 게임 시작이나 종료 시 화면에 텍스트를 표시.
   b. main( ) 함수에서 게임 시작 전과 게임 오버 후에 호출
4. **checkForKeyPress( )**
   a. 키 입력을 확인 및 처리함
   b. 키보드 입력을 확인할 때 호출.
5. **getBlankBoard( )**
   a. 새로운 빈 게임 보드를 생성

b. 게임이 시작될 때 혹은 보드를 초기화 할 때 호출

6. **getNewPiece( )**
   a. 새로운 테트리스 조각을 생성
   b. 조각이 새로 생성 될 때마다 호출.
7. **addToBoard(board, piece)**
   a. 주어진 조각을 보드에 추가.
   b. 현재 블록을 게임 보드에 추가 할 때, 블록이 바닥에 닿거나 다른 블록에 닿을 때 호출.
8. **isOnBoard(x, y)**
   a. 주어진 좌표가 보드 안에 있는지 확인
   b. 블록의 위치가 게임 보드 안에 있는지 확인 할 때 호출
9. **isValidPosition(board, piece, adjX = 0, adjY = 0)**
   a. 조각이 보드 상에서 유효한 위치에 있는지 확인.
   b. 블록을 이동시키거나 회전시킬 때 호출.
10. **removeCompleteLines(board)**
    a. 완성된 라인을 제거하고 보드를 업데이트.
    b. runGame( ) 함수 내부에서 블록이 추가 된 후 호출.
11. **calculateLevelAndFallFreq(score)**
    a. 현재 점수를 기준으로 레벨과 조각이 떨어지는 속도를 계산
    b. runGame( ) 함수 내부에서 점수에 따라 레벨이 변할 때 호출
12. **makeTextObjs( )**
    a. 화면에 표시될 텍스트 객체를 생성할 때 사용.
    b. 텍스트 객체를 생성할 때 호출
13. **convertToPixelCoords(boxx, boxy)**
    a. 게임 보드 좌표를 화면 픽셀 좌표로 변환.
    b. 블록을 화면에 그릴 때 호출.
14. **drawBox(boxx, boxy, color, pixelx = None)**
    a. 지정된 색상의 박스를 그림. 게임 보드와 현재 블록을 그릴 때 사용

b. 개별 블록을 화면에 그릴 때 호출

15. **drawPiece(piece, pixelx = None, pixely = None)**
    a. 개별 조각을 화면에 그림.
    b. 블록을 이동시키거나 회전시킬 때 마다 호출.
16. **drawStatus(score, level)**
    a. 현재 점수와 레벨을 화면에 표시.
    b. runGame( ) 함수 내부에서 주기적으로 호출, 현재 점수와 레벨을 화면에 표시할 때 호출 함.
17. **terminate()**
    a. pygame.quit() 과 sys.exit() 을 통해 게임을 완전히 종료시킴
    b. 게임을 종료할 때 호출