# OSW 과제

## 보고서: Tetromino 게임 코드 분석

**2023080128 ICT융합학부 황병주**

## 목차 :



## 1. 원본에서 수정본을 만들 때 체크리스트들을 어떻게 수정했는지

**체크리스트 1: 테트리스 게임의 배경음악 설정**

원본:

```
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')

```

수정본:

```python
while True:
    # 배경 음악을 랜덤으로 선택하여 재생
    if random.randint(0, 1) == 0:
        pygame.mixer.music.load('Hover.mp3')
    else:
        pygame.mixer.music.load('Our_Lives_Past.mp3')
    pygame.mixer.music.play(-1, 0.0)

    # 게임 시작 시간을 저장
    start_ticks = pygame.time.get_ticks()

    # 게임을 실행
    runGame(start_ticks)

    # 배경 음악을 정지
    pygame.mixer.music.stop()

    # 게임 종료 화면을 보여주기
    showTextScreen('Over :(')
```

**체크리스트 2: 시작화면 수정하기**

원본:

```python
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.', BASICFONT, TEXTCOLOR)
    pressKeyRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2) + 100)
    DISPLAYSURF.blit(pressKeySurf, pressKeyRect)

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

수정본:

```python
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()
```

**체크리스트 3: "2023080128 byeongjoo hwang" 띄우기**

원본:


수정본:


**체크리스트 4: 일시정지 화면 수정하기**

원본:

```
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('Paused') # 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
```

수정본:

```python
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
```

**체크리스트 5: 게임 오버 화면 수정하기**

원본:

```python
showTextScreen('Game Over')
```

수정본:

```python
showTextScreen('Over :(')
```

**체크리스트 6: 게임 경과 시간을 초 단위로 왼쪽 상단에 표시하기**

추가내용:

```python
elapsed_time = (pygame.time.get_ticks() - start_ticks) / 1000  # 초 단위로 변환
time_text = BASICFONT.render(f'Play Time: {int(elapsed_time)}sec', True, YELLOW)
DISPLAYSURF.blit(time_text, (10, 10))  # 화면 왼쪽 상단에 타이머 표시
```

**체크리스트 7: 블록 생성 시 고유의 색상을 가지도록 수정**

수정내용:

원래 **getNewPiece**함수가 **'color': random.randint(0, len(COLORS)-1)** 를 통해 색상을 랜덤으로 지정했었다. 이를 수정하여 각 모양마다 정해진 색상을 가지도록 수정해보자!

(아쉽게도 COLORS는 사용이 안 되는 튜플이 되었다.)

먼저 **PIECES** 딕셔너리를 수정하여 각 모양에 대한 색상을 고정으로 fix하는 방식으로 접근했다.

```python
PIECES = {'S': {'shape': S_SHAPE_TEMPLATE, 'color': GREEN}, #색상을 랜덤이 아니라, 정해놓기 위해 사전을 수정
          'Z': {'shape': Z_SHAPE_TEMPLATE, 'color': RED},
          'J': {'shape': J_SHAPE_TEMPLATE, 'color': BLUE},
          'L': {'shape': L_SHAPE_TEMPLATE, 'color': YELLOW},
          'I': {'shape': I_SHAPE_TEMPLATE, 'color': WHITE},
          'O': {'shape': O_SHAPE_TEMPLATE, 'color': PURPLE},
          'T': {'shape': T_SHAPE_TEMPLATE, 'color': LIGHTRED}}
```

딕셔너리의 값과 키가 변형되었으므로, 함수들의 PIECES딕셔너리에 대한 접근(인자로 받는 부분!) 또한 수정해주어야 했다.

수정되는 함수들:

**drawPiece**

```python
def drawPiece(piece, pixelx=None, pixely=None):#PIECES[piece['shape']]['shape'][piece['rotation']]으로 접근해야 함
    shapeToDraw = PIECES[piece['shape']]['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, PIECES[piece['shape']]['color'], pixelx + (x * BOXSIZE), pixely + (y * BOXSIZE))

```

**drawNextPiece**

```python
def drawNextPiece(piece): #마찬가지
    # 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)
```

**isValidPosition**

```python
def isValidPosition(board, piece, adjX=0, adjY=0): #PIECES 사전 구조가 변경되었기 때문에, 이에 맞게 접근 방식을 수정
    # 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']]['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
```

### addToBoard

```python
def addToBoard(board, piece): #PIECES 사전에 접근할 때 PIECES[piece['shape']]['shape'][piece['rotation']] 형식을 사용해야 함
    # 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']]['shape'][piece['rotation']][y][x] != BLANK:
                board[x + piece['x']][y + piece['y']] = piece['color']
```

이제 수정된 사전에 대한 참조방식의 오류는 해결되었다.

추가로 한 가지 함수를 더 수정해야 하는데,

color는 이미 RGB튜플이므로 COLORS사전을 사용하지 않고 다음과 같이 color를 그대로 사용하도록 변경해주었다.

### drawBox

```python
def drawBox(boxx, boxy, color, pixelx=None, pixely=None): #color를 그대로 사용하여 색상을 지정.color 변수는 이미 RGB 튜플이므로 COLORS 사전을 사용하지 않고 바로 사용할 수 있음
    # draw a single box (each tetromino piece has four boxes)
    if color == BLANK:
        return
    if pixelx == None and pixely == None:
        pixelx, pixely = convertToPixelCoords(boxx, boxy)
    pygame.draw.rect(DISPLAYSURF, color, (pixelx + 1, pixely + 1, BOXSIZE - 1, BOXSIZE - 1))
    pygame.draw.rect(DISPLAYSURF, BORDERCOLOR, (pixelx, pixely, BOXSIZE, BOXSIZE), 1)

```

**재밌어서 다음과 같은 색상을 좀 더 추가해주었다**

PURPLE = (128, 0, 128)
LIGHTPURPLE = (160, 32, 240)
ORANGE = (255, 165, 0)
LIGHTORANGE = (255, 165, 20)
CYAN = ( 0, 255, 255)
LIGHTCYAN = ( 20, 255, 255)
MAGENTA = (255, 0, 255)
LIGHTMAGENTA= (255, 20, 255)

## 2. 각 함수들의 역할에 대한 설명 +

# 3.전체적인 뼈대 함수의 호출 조건과 순서에 대한 설명

- `main()` **함수**: 게임을 초기화하고 게임 루프를 시작. 게임 루프는 시작화면을 표시하고, 게임을 실행하고, 게임 종료 화면을 표시하는 역할을 함. 프로그램 실행 시 첫 번째로 호출되며 사용자가 게임을 종료하거나 창을 닫을 때까지 지속적으로 호출. 게임 루프인 `runGame()` 함수를 호출하여 게임을 실행
- `runGame(start_ticks)` **함수**: 게임의 주요 루프를 실행. 게임 보드를 초기화하고, 블록의 움직임을 처리하고, 게임 화면을 갱신함 또한, 일시정지 기능과 게임 종료 조건을 처리. `main()` 함수에서 시작합니다.사용자가 게임을 시작하거나 일시정지를 해제할 때 호출.사용자가 게임을 종료하거나 게임 오버 조건이 발생했을 때 종료
- `showTextScreen(text)` **함수**: 대화식 화면을 표시하고 사용자의 입력을 기다림 시작화면, 일시정지화면, 게임 종료 화면을 모두 이 함수를 사용하여 구현. 시작화면, 일시정지화면, 게임 종료 화면을 표시할 때, 사용자의 입력을 기다릴 때 호출
- `calculateLevelAndFallFreq(score)` **함수**: 현재 점수를 기반으로 플레이어의 레벨과 블록이 떨어지는 속도를 계산
- `getNewPiece()` **함수**: 새로운 블록을 생성. 이때, 각 블록의 모양과 색상은 `PIECES` 사전에서 가져옴
- `addToBoard(board, piece)` **함수**: 현재 블록을 게임 보드에 추가
- `isValidPosition(board, piece, adjX=0, adjY=0)` **함수**: 현재 블록의 위치가 유효한지 검사. 블록이 보드 내에 있고 다른 블록과 충돌하지 않는 경우를 유효한 위치로 간주.
- `removeCompleteLines(board)` **함수**: 완전히 채워진 행을 제거하고, 위에 있는 블록을 아래로 이동
- `drawBoard(board)` **함수**: 게임 보드를 화면에 그린다. 이때, 각 블록의 색상은 해당 블록의 위치에 따라 지정.
- `drawStatus(score, level)` **함수**: 현재 점수와 레벨을 화면에 표시.
- `drawPiece(piece, pixelx=None, pixely=None)` **함수**: 현재 블록을 화면에 그림. 각 블록의 색상은 해당 블록의 모양에 따라 지정
- `drawNextPiece(piece)` **함수**: 다음에 나올 블록을 화면에 그림. 이 함수는 "Next:" 텍스트와 함께 다음 블록을 표시
- `checkForKeyPress()` **함수**: 사용자의 키 입력을 확인. 키 이벤트가 발생하면 해당 키를 반환.
- `checkForQuit()` **함수**: 게임 종료 여부를 확인. 사용자가 창을 닫거나 'Esc' 키를 누르면 게임종료
- `terminate()` **함수**: 게임을 종료합니다.
- `convertToPixelCoords(boxx, boxy)` **함수**: 게임 보드의 좌표를 화면 상의 좌표로 변환
- `makeTextObjs(text, font, color)` **함수**: 텍스트를 Surface 객체로 변환, 이 함수는 텍스트 렌더링에 사용
- `getBlankBoard()` **함수:**새로운 빈 게임 보드 데이터 구조를 생성하고 반환. 게임 보드의 가로와 세로 크기에 맞게 빈 보드를 생성. `runGame()` 함수 내에서 게임이 시작될 때, 새로운 게임 보드가 필요할 때 호출.
- `isOnBoard(x, y)` **함수:** 주어진 좌표가 게임 보드 내에 있는지 확인하여 True 또는 False를 반환. x와 y 좌표가 음수가 아니고, 보드의 가로와 세로 크기를 넘지 않는지 확인.떨어지는 조각의 위치를 확인할 때 사용.
- `isCompleteLine(board, y)` **함수:** 주어진 y 좌표의 라인이 블록으로 완전히 채워진 상태인지 확인하여 True 또는 False를 반환. 라인의 모든 열을 순회하며 빈 공간이 있는지 확인. 행이 꽉 찼는지 확인하여 행을 삭제할 때 사용
- `drawBox(boxx, boxy, color, pixelx=None, pixely=None)` **함수:**주어진 위치에 상자를 그리는 함수. 상자의 좌표와 색상을 사용하여 상자를 그리고 색상이 BLANK인 경우 아무것도 그리지 않음.

# 전체 완성코드 :

```
# 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 = '.'
start_ticks = pygame.time.get_ticks()

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)

#더 추가

PURPLE      = (128,   0, 128)
LIGHTPURPLE = (160,  32, 240)
ORANGE      = (255, 165,   0)
LIGHTORANGE = (255, 165,  20)
CYAN        = (  0, 255, 255)
LIGHTCYAN   = ( 20, 255, 255)
MAGENTA     = (255,   0, 255)
LIGHTMAGENTA= (255,  20, 255)

BORDERCOLOR = BLUE
BGCOLOR = BLACK
TEXTCOLOR = YELLOW
TEXTSHADOWCOLOR = YELLOW
COLORS      = (     BLUE,      GREEN,      RED,      YELLOW)
LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYELLOW)
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}

PIECES = {'S': {'shape': S_SHAPE_TEMPLATE, 'color': GREEN}, #색상을 랜덤이 아니라, 정해놓기 위해 사전을
          'Z': {'shape': Z_SHAPE_TEMPLATE, 'color': RED},
          'J': {'shape': J_SHAPE_TEMPLATE, 'color': BLUE},
          'L': {'shape': L_SHAPE_TEMPLATE, 'color': YELLOW},
          'I': {'shape': I_SHAPE_TEMPLATE, 'color': WHITE},
          'O': {'shape': O_SHAPE_TEMPLATE, 'color': PURPLE},
          'T': {'shape': T_SHAPE_TEMPLATE, 'color': LIGHTRED}}


# main 함수: 게임의 주요 루프를 실행하는 함수
# 전역 변수 FPSCLOCK, DISPLAYSURF, BASICFONT, BIGFONT을 사용함.
def main():
    global FPSCLOCK, DISPLAYSURF, BASICFONT, BIGFONT

    # Pygame 라이브러리를 초기화
```

```
    pygame.init()

    # 게임 루프의 프레임 속도를 제어하기 위한 Clock 객체를 생성
    FPSCLOCK = pygame.time.Clock()

    # 게임 창의 너비와 높이를 설정하여 화면을 생성
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))

    # 기본 폰트와 큰 폰트를 설정이 폰트는 게임 내에서 텍스트를 표시하는 데 사용됨
    BASICFONT = pygame.font.Font('freesansbold.ttf', 18)
    BIGFONT = pygame.font.Font('freesansbold.ttf', 100)

    # 게임 창의 제목을 설정
    pygame.display.set_caption('2023080128 byeongjoo hwang')

    # 게임 시작 화면을 보여줌
    showTextScreen('MT TETRIS')

    # 게임 루프를 시작
    while True:
        # 배경 음악을 랜덤으로 선택하여 재생
        if random.randint(0, 1) == 0:
            pygame.mixer.music.load('Hover.mp3')
        else:
            pygame.mixer.music.load('Our_Lives_Past.mp3')
        pygame.mixer.music.play(-1, 0.0)

        # 게임 시작 시간을 저장
        start_ticks = pygame.time.get_ticks()

        # 게임을 실행
        runGame(start_ticks)

        # 배경 음악을 정지
        pygame.mixer.music.stop()

        # 게임 종료 화면을 보여주기
        showTextScreen('Over :(')


def runGame(start_ticks): #인자를 받도록 수정함.
    # 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)

    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()
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:
        # mering the piece sideways
        if (event.key == K_LEFT or event.key == K_a) and isValidPosition(board, fallingP
            fallingPiece['x'] -= 1
            movingLeft = True
            movingRight = False
            lastMoveSidewaysTime = time.time()

        elif (event.key == K_RIGHT or event.key == K_d) and isValidPosition(board, falli
            fallingPiece['x'] += 1
            movingRight = True
            movingLeft = False
            lastMoveSidewaysTime = time.time()

        # 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[falli
            if not isValidPosition(board, fallingPiece):
                fallingPiece['rotation'] = (fallingPiece['rotation'] - 1) % len(PIECES[f
        elif (event.key == K_q): # rotate the other direction
            fallingPiece['rotation'] = (fallingPiece['rotation'] - 1) % len(PIECES[falli
            if not isValidPosition(board, fallingPiece):
                fallingPiece['rotation'] = (fallingPiece['rotation'] + 1) % len(PIECES[f

        # 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

        # handle moving the piece because of user input
        if (movingLeft or movingRight) and time.time() - lastMoveSidewaysTime > MOVESIDEWAYSFRE(
            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(boar
            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)
        drawNextPiece(nextPiece)
        if fallingPiece != None:
            drawPiece(fallingPiece)

        elapsed_time = (pygame.time.get_ticks() - start_ticks) / 1000  # 초 단위로 변환
        time_text = BASICFONT.render(f'Play Time: {int(elapsed_time)}sec', True, YELLOW)
        DISPLAYSURF.blit(time_text, (10, 10))  # 화면 왼쪽 상단에 타이머 표시

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


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

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


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


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
    pressKeyRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2) + 100)
    DISPLAYSURF.blit(pressKeySurf, pressKeyRect)

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


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


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
```

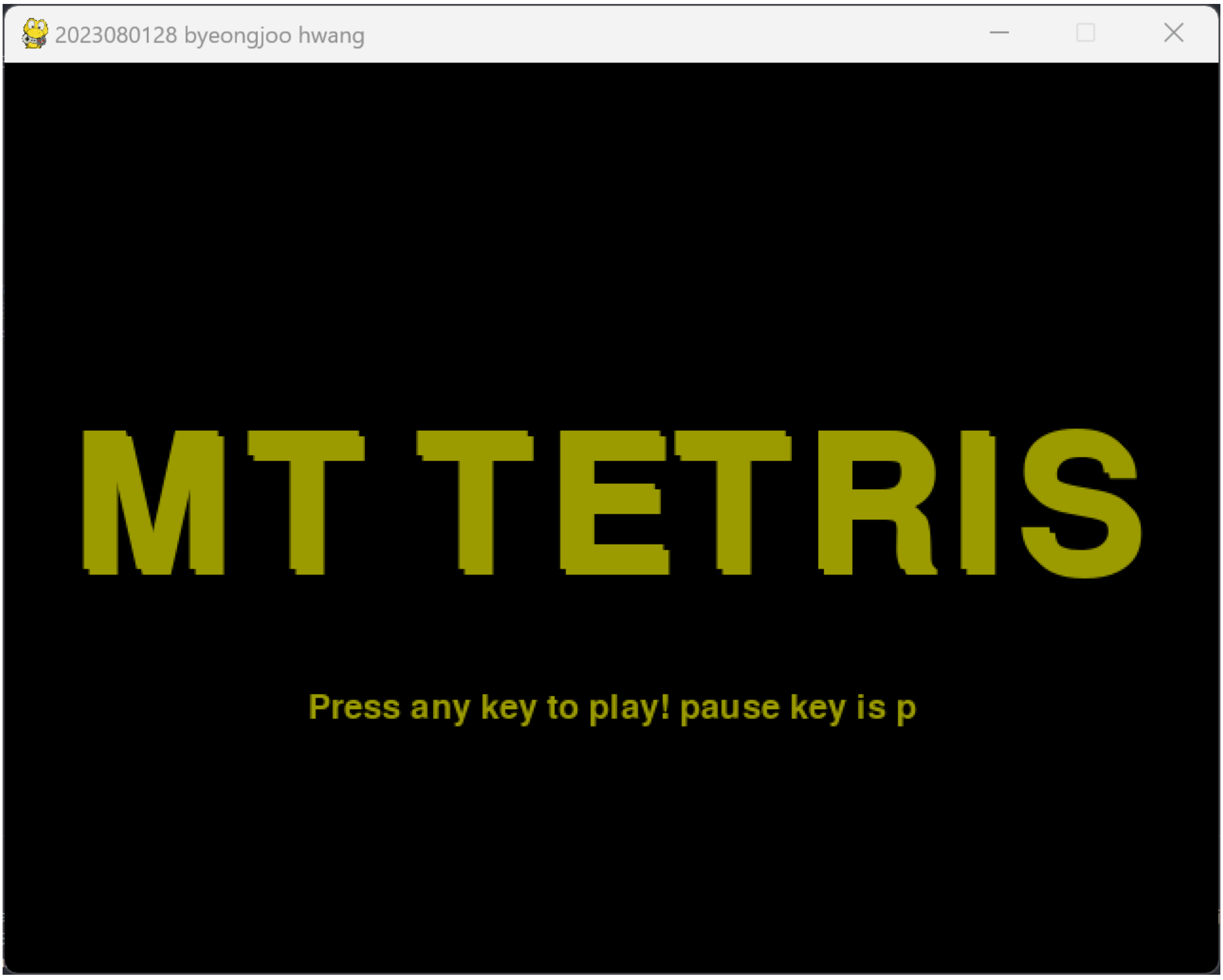
```
# def getNewPiece():
#     # return a random new piece in a random rotation and color
#     shape = random.choice(list(PIECES.keys()))
#     newPiece = {'shape': shape,
#                 'rotation': random.randint(0, lenP(IECES[shape]) - 1),
#                 'x': int(BOARDWIDTH / 2) - int(TEMPLATEWIDTH / 2),
#                 'y': -2, # start it above the board (i.e. less than 0)
#                 'color': random.randint(0, len(COLORS)-1)}
#     return newPiece

def getNewPiece(): # 색상을 정하기 위한 함수의 수정
    # 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]['shape']) - 1),
                'x': int(BOARDWIDTH / 2) - int(TEMPLATEWIDTH / 2),
                'y': -2, # start it above the board (i.e. less than 0)
                'color': PIECES[shape]['color']}
    return newPiece



# 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']

def addToBoard(board, piece): #PIECES 사전에 접근할 때 PIECES[piece['shape']]['shape'][piece['rotat
    # 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']]['shape'][piece['rotation']][y][x] != BLANK:
                board[x + piece['x']][y + piece['y']] = piece['color']



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


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


# 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):
```

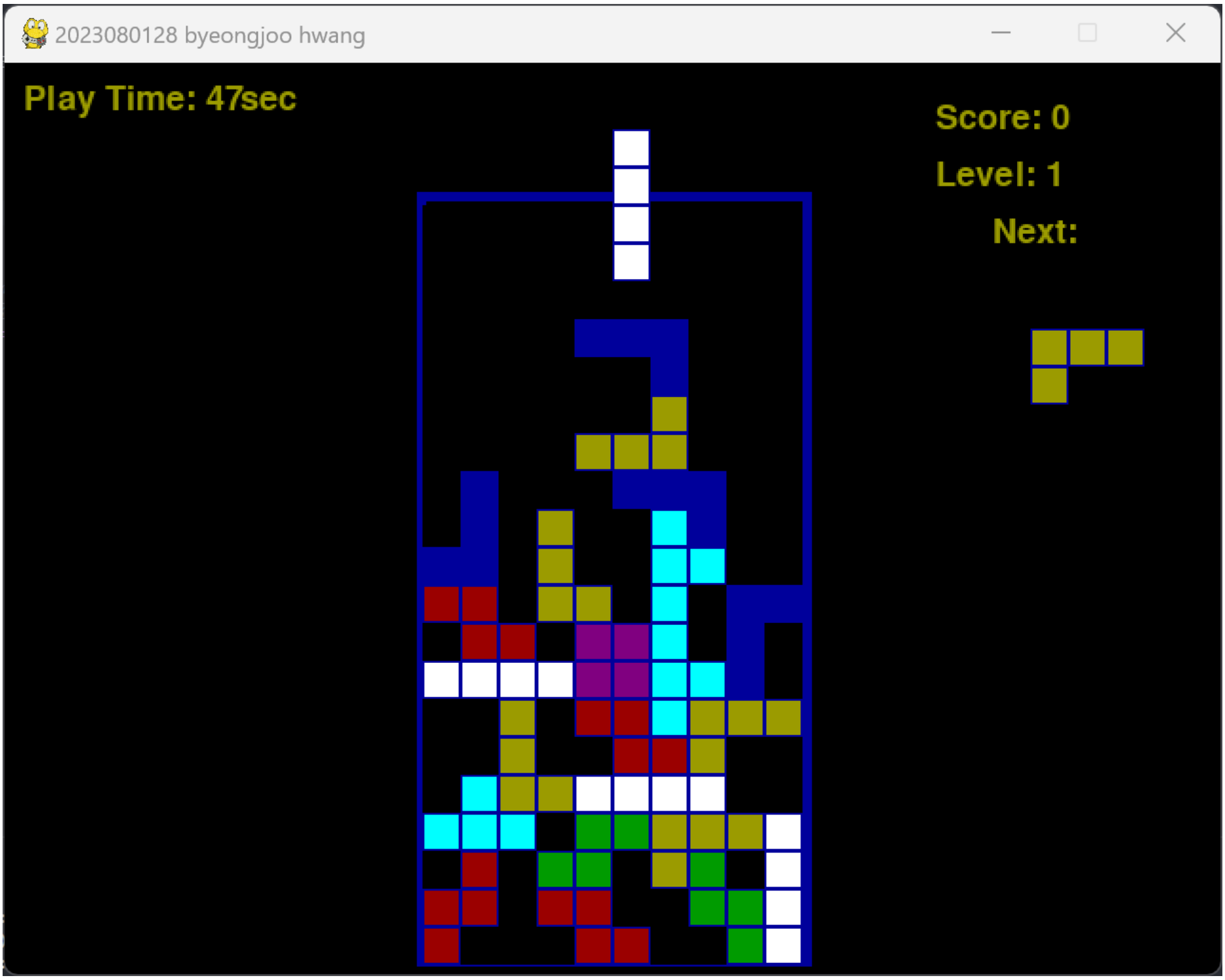
```
#                 return False
#             if board[x + piece['x'] + adjX][y + piece['y'] + adjY] != BLANK:
#                 return False
#     return True

def isValidPosition(board, piece, adjX=0, adjY=0): #PIECES 사전 구조가 변경되었기 때문에, 이에 맞게 접근
    # 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']]['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


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 n
    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


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))


# 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
```

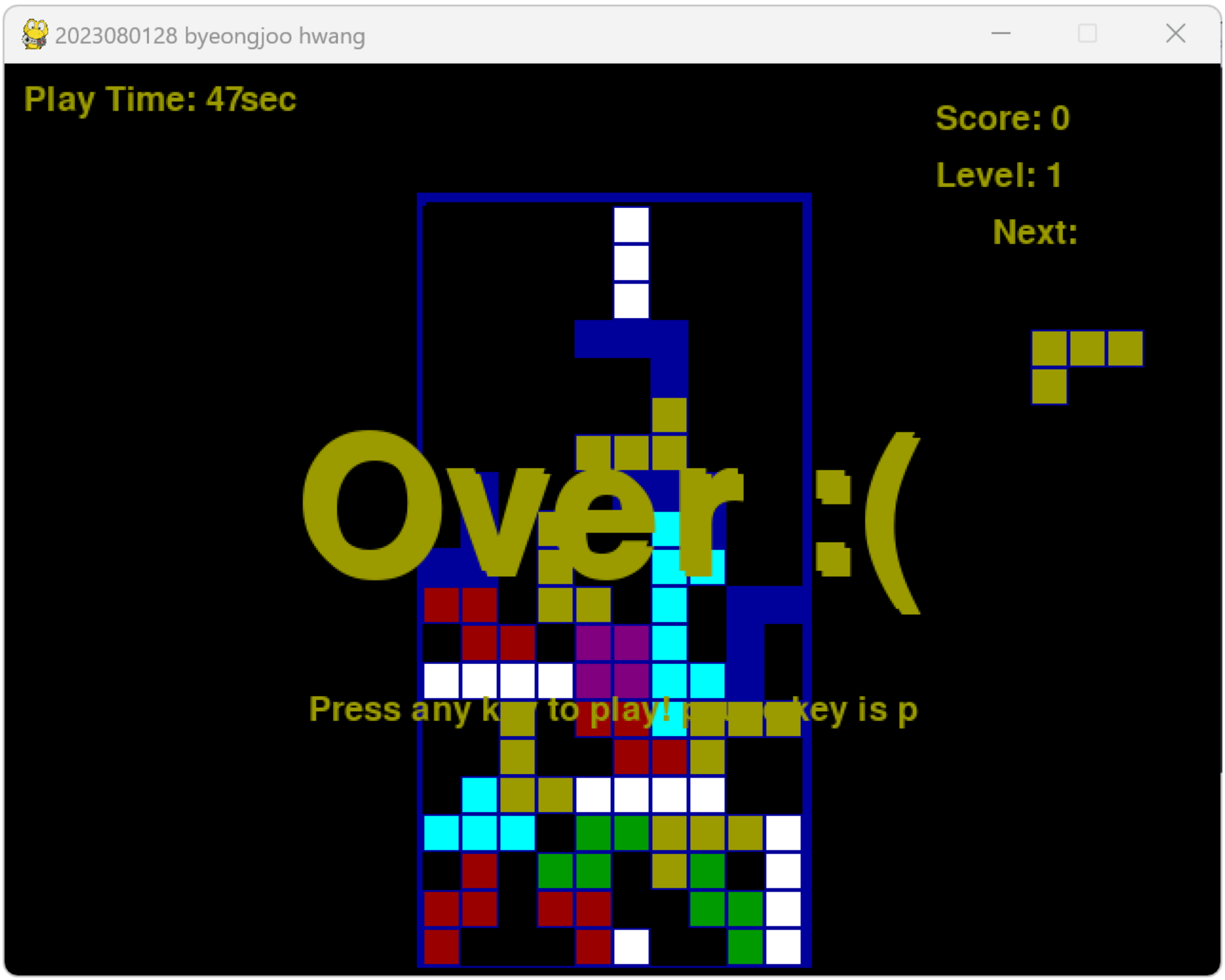
```
#     # are specified, draw to the pixel coordinates stored in
#     # pixelx & pixely (this is used for the "Next" piece).
#     if color == BLANK:
#         return
#     if pixelx == None and pixely == None:
#         pixelx, pixely = convertToPixelCoords(boxx, boxy)
#     pygame.draw.rect(DISPLAYSURF, COLORS[color], (pixelx + 1, pixely + 1, BOXSIZE - 1, BOXSIZE
#     pygame.draw.rect(DISPLAYSURF, LIGHTCOLORS[color], (pixelx + 1, pixely + 1, BOXSIZE - 4, BO

def drawBox(boxx, boxy, color, pixelx=None, pixely=None): #color를 그대로 사용하여 색상을 지정.color
    # draw a single box (each tetromino piece has four boxes)
    if color == BLANK:
        return
    if pixelx == None and pixely == None:
        pixelx, pixely = convertToPixelCoords(boxx, boxy)
    pygame.draw.rect(DISPLAYSURF, color, (pixelx + 1, pixely + 1, BOXSIZE - 1, BOXSIZE - 1))
    pygame.draw.rect(DISPLAYSURF, BORDERCOLOR, (pixelx, pixely, BOXSIZE, BOXSIZE), 1)

def drawBoard(board):
    # draw the border around the board
    pygame.draw.rect(DISPLAYSURF, BORDERCOLOR, (XMARGIN - 3, TOPMARGIN - 7, (BOARDWIDTH * BOXSIZ

    # fill the background of the board
    pygame.draw.rect(DISPLAYSURF, BGCOLOR, (XMARGIN, TOPMARGIN, BOXSIZE * BOARDWIDTH, BOXSIZE *
    # draw the individual boxes on the board
    for x in range(BOARDWIDTH):
        for y in range(BOARDHEIGHT):
            drawBox(x, y, board[x][y])


def drawStatus(score, level):
    # 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)


# 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
#         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 * BOXS

def drawPiece(piece, pixelx=None, pixely=None):#PIECES[piece['shape']]['shape'][piece['rotation
    shapeToDraw = PIECES[piece['shape']]['shape'][piece['rotation']]
```

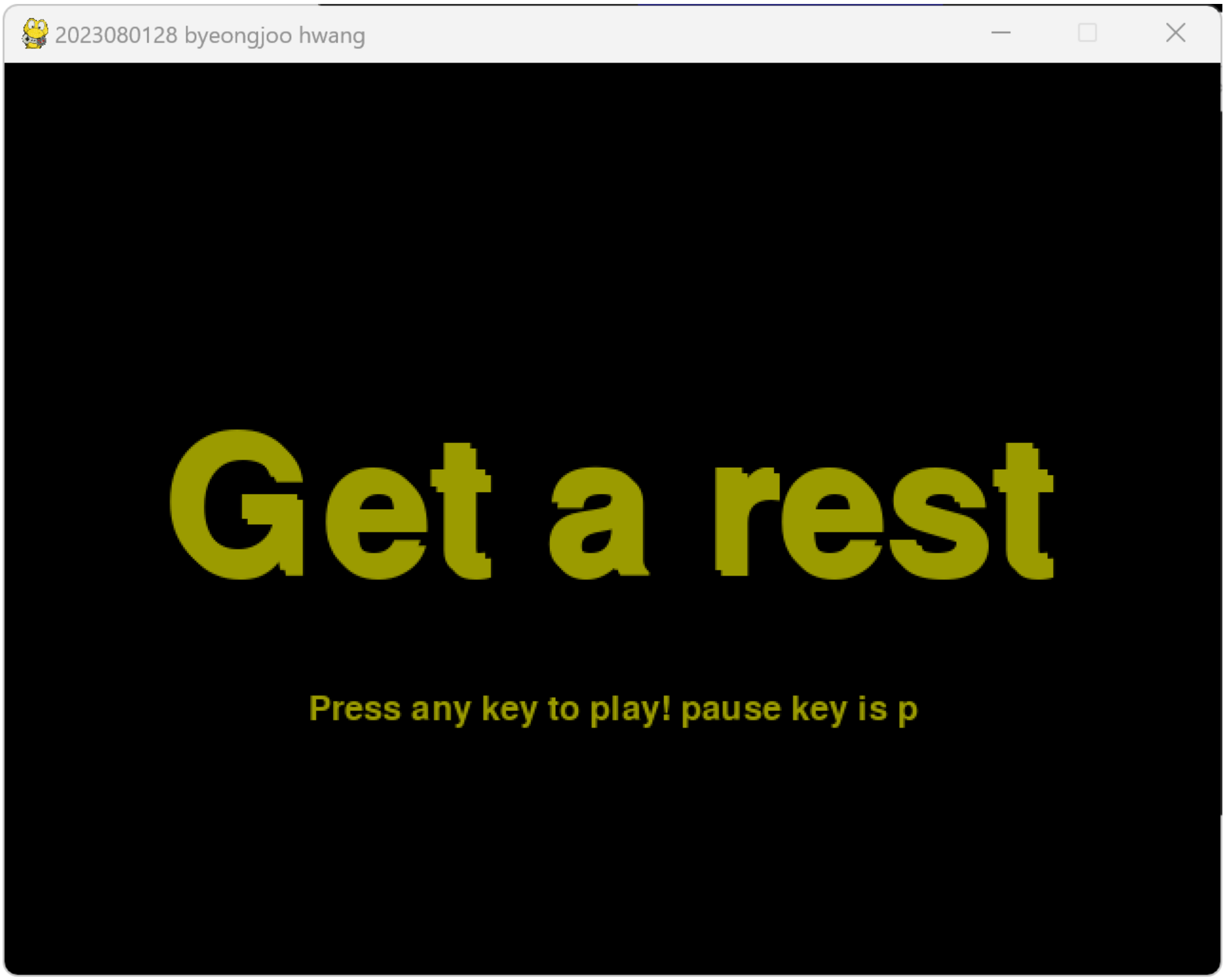
```
    if pixelx == None and pixely == None:
        # if pixelx & pixely hasn't been specified, use the location stored in the piece data st
        pixelx, pixely = convertToPixelCoords(piece['x'], piece['y'])

    # 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, PIECES[piece['shape']]['color'], pixelx + (x * BOXSIZE), pix

# 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)

def drawNextPiece(piece): #마찬가지
    # 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()
```

## 실행화면


2023080128 byeongjoo hwang
MT TETRIS
Press any key to play! pause key is p


2023080128 byeongjoo hwang
Play Time: 47sec
Score: 0
Level: 1
Next:


2023080128 byeongjoo hwang
Play Time: 47sec
Score: 0
Level: 1
Next:
Over :(
Press any k to play! key is p


2023080128 byeongjoo hwang
Get a rest
Press any key to play! pause key is p