# 오픈소스 **SW**

과제중심수업 1주차 보고서

2019006608 최명학

# Introduction

테트로미노는 테트리스의 복제품으로 테트리스는 1984년 출시된 간단하면서도 나름의 다양한 규칙이 있는 게임입니다. 룰을 설명하려면 생각보다 까다롭습니다. 그런데 융통성이라곤 하나도 없는 컴퓨터에게 이해시키려면 세부적으로 입력해야 할 것이 매우 많습니다.
해당 오픈소스 소프트웨어를 활용하여 기존 테트리스 게임을 수정하는 과제를 수행함으로써 오픈소스의 개념과 이점을 이해할 수 있습니다. 기존의 코드를 분석하고 수정하는 과정을 통해 코드의 작동 원리를 좀 더 손쉽게 이해하고, 코딩 구조를 이해하며 기술을 향상시킬 수 있습니다.

## ⅰ 테트리스 게임 수정

### 1. 현재 테트리스 게임의 배경음악을 주어진 3개의 음악 중 1개가 재생되도록 수정

```
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('Tetromino')

    showTextScreen('Tetromino')
    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')
```

기존 메인함수입니다. tetrisb.mid, tetrisc.mid 파일이 존재하지 않아 원본 소스가 실행되지 않습니다.

먼저, 코드 내의 'tetrisb.mid' 와 tetrisc.mid 를 수업자료의 Hover.mp3, Platform_9.mp3, Our_Lives_Past.mp3 로 대체합니다. 기존 코드에는 0과 1의 정수 중 무작위로 하나를 출력하고, 0이면 tetrisb.mid 음악 파일, 그 외 tetrisc.mid 음악 파일을 실행합니다. 과제에서

주어진 음악파일은 3개이므로 0,1,2의 정수 중 무작위로 하나를 출력하고, 0이면 Hover.mp3, 1이면 Platform_9.mp3, 그 외 Our_Lives_Past.mp3 를 재생하도록 수정했습니다.

```
randomMusic = random.randint(0, 2)
while True: # game loop
    start_time = time.time()
    if randomMusic == 0:
        pygame.mixer.music.load('Hover.mp3')
    elif randomMusic == 1:
        pygame.mixer.music.load('Platform_9.mp3')
    else:
        pygame.mixer.music.load('Our_Lives_Past.mp3')
    pygame.mixer.music.play(-1, 0.0)
    runGame(start_time)
    pygame.mixer.music.stop()
    showTextScreen('Game Over')
```

## **2.** 상태창 이름을 학번_이름 으로 수정 **+**
## **3.** 게임시작화면의 문구를 **MY TETRIS**으로 변경

기존 메인함수에서 해당 내용을 다루는 부분을 찾을 수 있었습니다.
pygame.display.set_caption('Tetromino') 을
pygame.display.set_caption('2019006608_최명학') 으로

showTextScreen('Tetromino')을
showTextScreen('MY TETRIS') 로 변경하였습니다.

```
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('2019006608_최명학')#2

    showTextScreen('MY TETRIS')#3
```

## **4.** 게임시작화면의 문구 및 배경색을 노란색으로 변경

먼저, 게임 내 문구를 출력하는 함수를 찾았습니다.

```
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, TEXTCOLOR)
    titleRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2))
    DISPLAYSURF.blit(titleSurf, titleRect)

    # Draw the text
    titleSurf, titleRect = makeTextObjs(text, BIGFONT, TEXTSHADOWCOLOR)
    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()
```

해당 부분에서 titleSurf, titleRect = makeTextObjs(text, BIGFONT, TEXTCOLOR)
의 TEXTCOLOR 을 LIGHTYELLOW 로,

titleSurf, titleRect = makeTextObjs(text, BIGFONT, TEXTSHADOWCOLOR)를 YELLOW로 바꾸었습니다.

기존 필드에 정의된 RGB값을 바로 사용했습니다.

```
#                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)
```

이렇게 하면 게임을 실행 시 제목인 MY TETRIS라는 문구만 노란색으로 나옵니다.
체크리스트에는 게임 시작 부분의 문구만 노란색으로 변경하라고 되어 있었으나 LMS의 완성된 과제를 보여주는 PPT 이미지에서는 모든 문구 색상이 노란색으로 되어 있어,
필드의
TEXTCOLOR = WHITE
TEXTSHADOWCOLOR = GRAY

을 각각 LIGHTYELLOW, YELLOW 로 변경해 밝은 노란색을 메인 TEXT로, 그림자를 비교적 어두은 노란색으로 설정하여 모든 글씨와 그림자를 변경했습니다.

```
TEXTCOLOR = LIGHTYELLOW
TEXTSHADOWCOLOR = YELLOW
```

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

## **5-1.** 먼저 플레이 타임을 계산하기 위해 함수를 추가하기

플레이 타임 = 현재시간 - 시작 시간

함수를 추가하기 전, 함수에 사용하기 위해 코드 상단 필드에 start_time = time.time() 를 추가합니다.

```
import random, time, pygame, sys
from pygame.locals import *

FPS = 25
WINDOWWIDTH = 640
WINDOWHEIGHT = 480
BOXSIZE = 20
BOARDWIDTH = 10
BOARDHEIGHT = 20
BLANK = '.'
start_time = time.time()

MOVESIDEWAYSFREQ = 0.15
MOVEDOWNFREQ = 0.1

XMARGIN = int((WINDOWWIDTH - BOARDWI
```

```
def calculate_play_time(start_time):
    # 현재 시간을 가져옵니다.
    current_time = time.time()

    # 시작 시간과 현재 시간의 차이를 계산합니다.
    play_time = current_time - start_time

    return int(play_time)
```

시작 시간을 입력으로 받고, 경과된 시간을 반환합니다. 리턴 타입은 정수형으로 설정하여 초 단위로 보이게 합니다.

그리고, 이 함수를 def runGame(): 에 추가합니다.

```
        # drawing everything on the screen
        DISPLAYSURF.fill(BGCOLOR)
        drawBoard(board)
        play_time = calculate_elapsed_time(start_time)###############6
        drawStatus(score, level,play_time)
        drawNextPiece(nextPiece)
        if fallingPiece != None:
            drawPiece(fallingPiece)

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

## **5-2.** 출력한 플레이 타임을 화면에 출력하기

아래는 기존 drawStatus 함수입니다.

```
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)
```

이 함수에 매개변수 play_time 을 추가하고, 다른 스탯과 동일하게 화면에 보이도록 코드를 추가합니다.

```
    # draw the elapsed time
    timeSurf = BASICFONT.render('Play Time: %s sec'% play_time, True, TEXTCOLOR)
    timeRect = timeSurf.get_rect()
    timeRect.topleft = (WINDOWWIDTH - 600, 80)
    DISPLAYSURF.blit(timeSurf, timeRect)
```

timeRect.topleft = (WINDOWWIDTH - 600, 80)로 설정하여 화면의 좌측에 위치하도록 하였습니다.

함수의 매개변수가 추가되었으므로, 기존의 runGame() 함수에 전달인자를 추가합니다.

```
# drawing everything on the screen
DISPLAYSURF.fill(BGCOLOR)
drawBoard(board)
play_time = calculate_elapsed_time(start_time)###############6
drawStatus(score, level,play_time)
drawNextPiece(nextPiece)
if fallingPiece != None:
    drawPiece(fallingPiece)

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

## **5-3** 게임이 재시작 되면 **0**초로 초기화 하기

여기까지 작성하면 정상적으로 경과 시간이 출력되지만 게임오버 후 재시작해도 0초로 돌아가지 않고 그대로 흐른 시간을 출력합니다. 다시 0초로 돌아가게 하려면 기존의 경과 시간 계산 함수에서, 시작 시간을 게임이 재시작 된(다시 메인함수가 실행된) 기준으로 다시 초기화 해주어야 합니다.

```
pygame.mixer.music.play(-1, 0
runGame(start_time)
pygame.mixer.music.stop()
showTextScreen('Game Over')
```

메인함수 안에 runGame() 메소드에 매개변수와 인자를 추가합니다.

```
def runGame(start_time):
```

```
def main():
    global FPSCLOCK, DISPLAYSURF, BASICFONT,
    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((W
    BASICFONT = pygame.font.Font('freesansbo
    BIGFONT = pygame.font.Font('freesansbolo
    pygame.display.set_caption('2019006608_ㅊ

    showTextScreen('MY TETRIS')#3

    randomMusic = random.randint(0, 2)
    while True: # game loop
        start_time = time.time()
        if randomMusic == 0:
            pygame.mixer.music.load('Hover.m
        elif randomMusic == 1:
            pygame.mixer.music.load('Platfor
        else:
            pygame.mixer.music.load('Our_Liv
        pygame.mixer.music.play(-1, 0.0)
        runGame(start_time)
        pygame.mixer.music.stop()
        showTextScreen('Game Over')
```

그리고 게임 루프가 시작될 때, start_time = time.time()으로 다시 정의해줍니다. 그럼 게임이 재시작할때마다 시작 시간이 다시 정의되고, 정상적으로 경과시간이 다시 계산됩니다.

```
    randomMusic = random.randint(0, 2)
    while True: # game loop
        start_time = time.time()
        if randomMusic == 0:
            pygame.mixer.music.load('Hover.mp3')
        elif randomMusic == 1:
            pygame.mixer.music.load('Platform_9.mp3')
        else:
            pygame.mixer.music.load('Our_Lives_Past.mp3')
        pygame.mixer.music.play(-1, 0.0)
        runGame(start_time)
        pygame.mixer.music.stop()
        showTextScreen('Game Over')


def runGame(start_time):
    # 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
```

# 6. 7개의 블록이 각각 고유의 색을 갖도록 코드를 수정하거나 추가

## 6-1 색상 추가하기

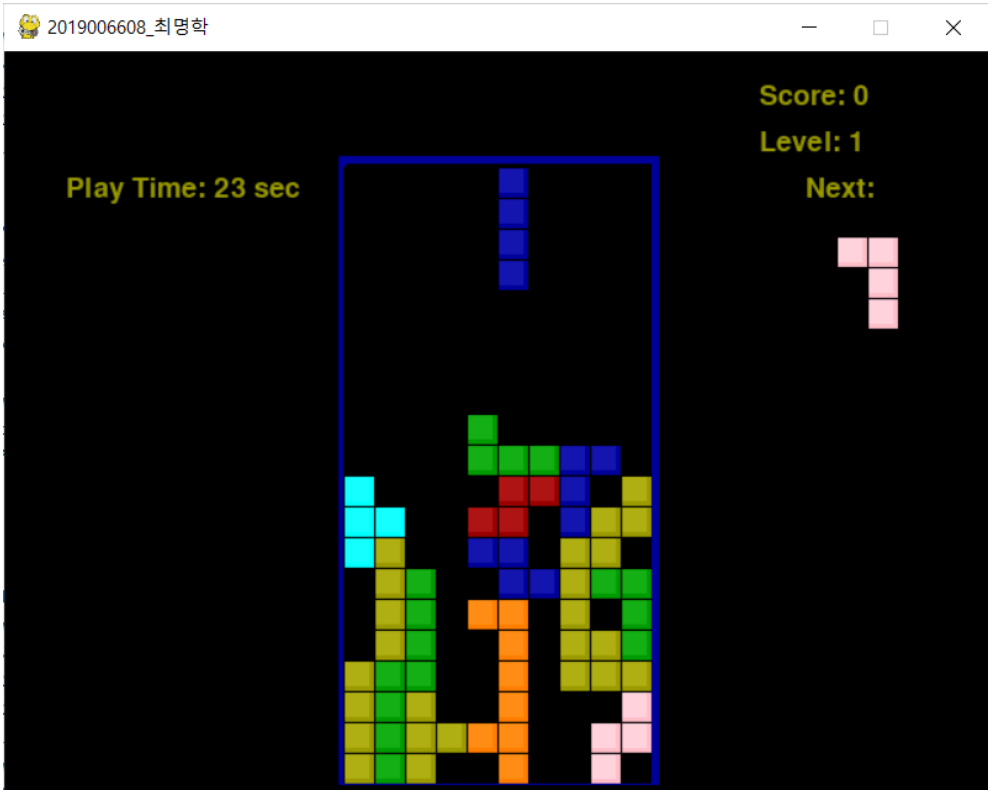
기존 소스코드에는 4가지 색상과 그 색상의 밝은 타입의 4가지 색상이 저장되어있습니다. 3가지의 색상을 먼저 추가합니다.

```
# 새로운 색상 추가
CYAN = (  0, 255, 255)
ORANGE = (255, 128,   0)
PINK = (255, 192, 203)
LIGHTCYAN = (  20, 255, 255)
LIGHTORANGE = (255, 142, 20)
LIGHTPINK = (255, 212, 223)
```

블록은 밝은 색과 어두운 색으로 구성되기 때문에 반드시 LIGHTCOLOR 도 추가해야 합니다.

```
COLORS      = (BLUE,      GREEN,      RED,      YELLOW,      CYAN,      ORANGE,      PINK)
LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYELLOW, LIGHTCYAN, LIGHTORANGE, LIGHTPINK)
assert len(COLORS) == len(LIGHTCOLORS) # each color must have light color
```

실행시켜보면, 정상적으로 7가지 색깔이 나오는 것을 확인 할 수 있습니다.




## **6-2** 블록의 모양에 따라 고유한 색상 부여하기

7가지 색상을 표현했지만, 색은 아직 무작위로 설정되어 있습니다.

이는 getNewPiece() 함수에서 확인할 수 있습니다.

```
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': random.randint(0, len(COLORS)-1)}
    return newPiece
```

해당 코드는 새로운 피스를 생성하는 함수로 무작위 모양과 색상, 회전상태, 초기위치를 반환합니다. 색상을 나타내는 'color': random.randint(0, len(COLORS)-1) 부분은 COLORS의 길이에서 1을 뺀 정수 (6) 와 0 사이의 무작위 정수로 색상이 정해지는 것을 확인할 수 있습니다.

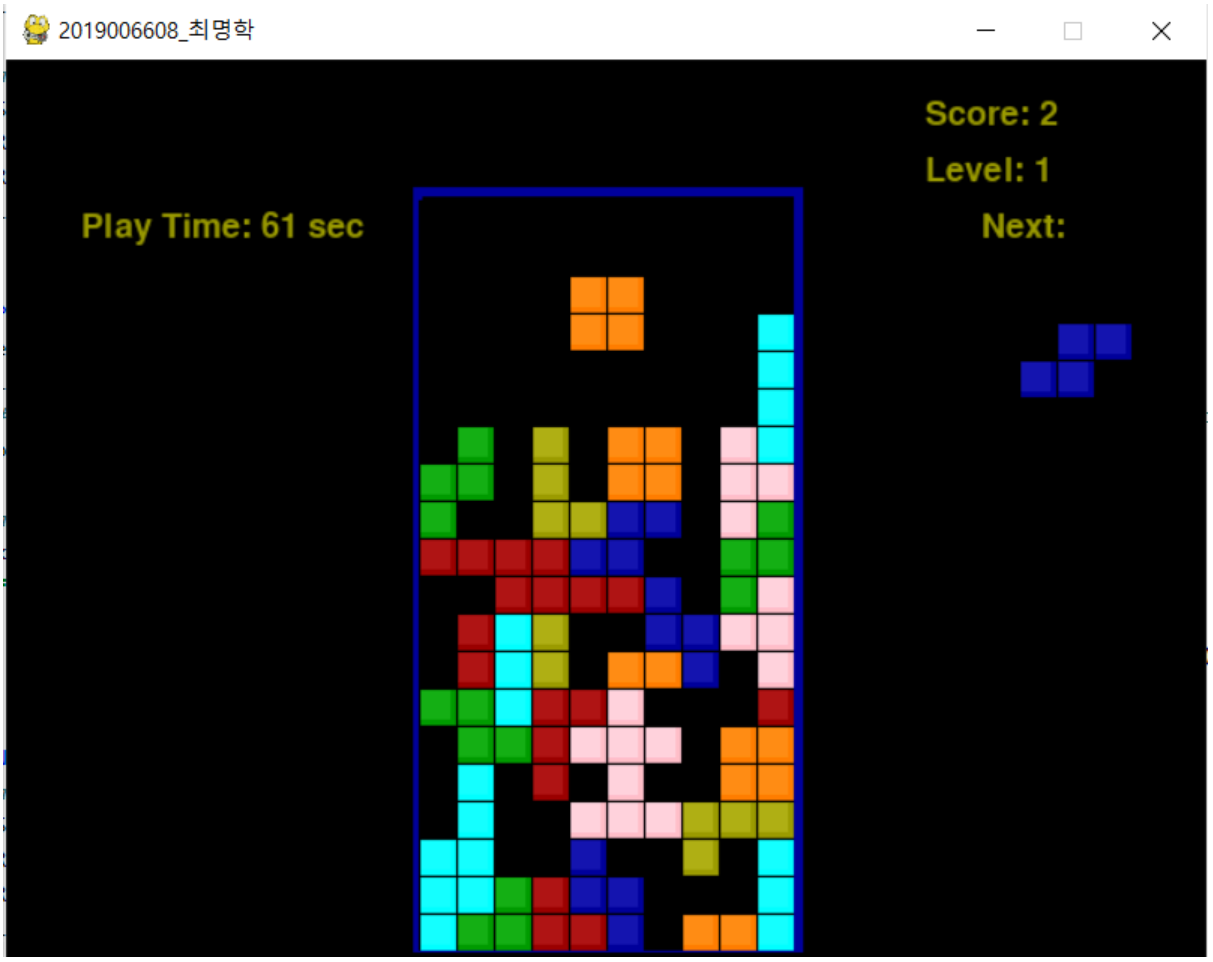
떠오른 방법은 shape이 정해졌을 때, shape 모양에 따라 0부터 6까지 고유의 정수를 반환한 값을 color 에 적용시키는 것이었습니다. 그러기 위해 먼저 shape 에 따라 0부터 6까지의 정수를 반환하는 함수를 만들었습니다.

```
def assignColor(shape):
    if shape == 'S':
        return 0
    elif shape == 'Z':
        return 1
    elif shape == 'J':
        return 2
    elif shape == 'L':
        return 3
    elif shape == 'I':
        return 4
    elif shape == 'O':
        return 5
    elif shape == 'T':
        return 6
```

그 후 getNewPiece() 를 수정했습니다. assignColor() 에서 반환된 정수를 newPiece의 color 로 정의했습니다.

```
def getNewPiece():
    # return a random new piece in a random rotation and color
    shape = random.choice(list(PIECES.keys()))
    color = assignColor(shape)
    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': color}
    return newPiece
```

실행 시켜보면, 각 블록마다 고유한 색상을 가지는 것을 확인할 수 있습니다.




## ii ) 각 함수의 역할

def main(): *수정한 함수

```
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('2019006608_ChoiMyeongHak')#2

    showTextScreen('MY TETRIS')#3

    randomMusic = random.randint(0, 2)
    while True: # game loop
        start_time = time.time()
        if randomMusic == 0:
            pygame.mixer.music.load('Hover.mp3')
        elif randomMusic == 1:
            pygame.mixer.music.load('Platform_9.mp3')
        else:
            pygame.mixer.music.load('Our_Lives_Past.mp3')
        pygame.mixer.music.play(-1, 0.0)
        runGame(start_time)
        pygame.mixer.music.stop()
        showTextScreen('Game Over')
```

pygame을 활용해 테트리스 게임을 만드는 메인함수입니다. 게임에 필요한 전역 변수를 설정하고, pygame 라이브러리를 초기화 합니다.

게임의 주요 화면을 설정합니다. 게임의 프레임을 제어하고, 게임창의 크기와 폰트 그리고 제목(상태창 : '2019006608_ChoiMyeongHak')을 설정합니다

게임 시작 화면에 텍스트를 표시하고, 랜덤 음악을 재생합니다.

메인 게임 루프를 관리합니다. 게임은 계속해서 실행되고, 게임이 시작된 시간을 기록하고 실제 게임을 실행합니다. 게임이 끝나면 음악을 중지하고 게임 오버 화면을 표시합니다.

## def runGame(start_time): *수정한 함수

```
def runGame(start_time):
    # 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()
```

board : 빈 보드를 생성함
last… : 마지막으로 조각이 움직인 시간을 기록합니다.
moving…:조각이 움직이는 방향을 제어하는 boolean
score : 점수 저장
level : 현재 레벨
fallFreq : 조각이 떨어지는 빈도
fallingPiece : 현재 떨어지고 있는 조각
nextPiece : 다음 조각

```
  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
```

fallingPiece 가 None 인 경우, 새로운 조각을 생성합니다. 새로운 조각이 보드에 말이 안되게 배치되면 게임이 끝납니다.

```
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
```

KEYUP 이벤트 처리 : p를 누르면 일시정지, 방향키나 wasd 키로 블록 움직임 제어

```
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()

    # rotating the piece (if there is room to rotate)
    elif (event.key == K_UP or event.key == K_w):
        fallingPiece['rotation'] = (fallingPiece['rotation'] + 1) % len(PIECES[fallingPiece['shape']])
        if not isValidPosition(board, fallingPiece):
            fallingPiece['rotation'] = (fallingPiece['rotation'] - 1) % len(PIECES[fallingPiece['shape']])
    elif (event.key == K_q): # rotate the other direction
        fallingPiece['rotation'] = (fallingPiece['rotation'] - 1) % len(PIECES[fallingPiece['shape']])
        if not isValidPosition(board, fallingPiece):
            fallingPiece['rotation'] = (fallingPiece['rotation'] + 1) % len(PIECES[fallingPiece['shape']])

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

KEYDOWN 이벤트 처리 : 좌우 방향키, a,d 로 좌우 이동, 윗 키, w, q 키로 회전,아래 키, s로 빠르게 내리기, 스페이스바로 즉시 바닥으로 내리기

```
# handle moving the piece because of user input
if (movingLeft or movingRight) and time.time() - lastMoveSidewaysTime > MOVESIDEWAYSFREQ:
    if movingLeft and isValidPosition(board, fallingPiece, adjX=-1):
        fallingPiece['x'] -= 1
    elif movingRight and isValidPosition(board, fallingPiece, adjX=1):
        fallingPiece['x'] += 1
    lastMoveSidewaysTime = time.time()

if movingDown and time.time() - lastMoveDownTime > MOVEDOWNFREQ and isValidPosition(board, fallingPiece, adjY=1):
    fallingPiece['y'] += 1
    lastMoveDownTime = time.time()

# let the piece fall if it is time to fall
if time.time() - lastFallTime > fallFreq:
    # see if the piece has landed
    if not isValidPosition(board, fallingPiece, adjY=1):
        # falling piece has landed, set it on the board
        addToBoard(board, fallingPiece)
        score += removeCompleteLines(board)
        level, fallFreq = calculateLevelAndFallFreq(score)
        fallingPiece = None
    else:
        # piece did not land, just move the piece down
        fallingPiece['y'] += 1
        lastFallTime = time.time()
```

사용자 입력에 따라 조각을 이동시키고, 주기적으로 조각이 한칸씩 떨어지고 조각이 착지하면 보드에 추가하고, 1줄이 완성되면 줄을 제거하고 점수가 1점 상승

```
# drawing everything on the screen
DISPLAYSURF.fill(BGCOLOR)
drawBoard(board)
play_time = calculate_play_time(start_time)################6
drawStatus(score, level,play_time)################6
drawNextPiece(nextPiece)
if fallingPiece != None:
    drawPiece(fallingPiece)

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

화면 그리기 : 화면을 지우고, 보드를 그리고 점수, 레벨, 플레이 타임을 표시
다음 조각을 그림, 화면 업데이트 하고 프레임 조절

## def makeTextObjs(text, font, color):

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

텍스트와 폰트, 색상을 받아서 Pygame 의 텍스트 렌더링 표면과 그 직사각형 반환

## def terminate():

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

pygame 을 종료하고 시스템을 종료함

## def checkForKeyPress():

```
def checkForKeyPress():
    # Go through event queue looking for a KEYUP event.
    # Grab KEYDOWN events to remove them from the event queue.
    checkForQuit()

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

키보드 이벤트 큐를 순회하여 키업 이벤트를 찾고 키다운 이벤트는 제거

## def showTextScreen(text):

```
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, TEXTCOLOR)
    titleRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2))
    DISPLAYSURF.blit(titleSurf, titleRect)

    # Draw the text
    titleSurf, titleRect = makeTextObjs(text, BIGFONT, TEXTSHADOWCOLOR)
    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()
```

중앙에 큰 텍스트를 표시하고 사용자가 키를 누를 때까지 대기. 텍스트의 그림자와 추가적인 안내 텍스트 표시

## def checkForQuit():

```
def checkForQuit():
    for event in pygame.event.get(QUIT): # get all the QUIT events
        terminate() # terminate if any QUIT events are present
    for event in pygame.event.get(KEYUP): # get all the KEYUP events
        if event.key == K_ESCAPE:
            terminate() # terminate if the KEYUP event was for the Esc key
        pygame.event.post(event) # put the other KEYUP event objects back
```

게임 종료 이벤트와 Esc 키 이벤트로 프로그램을 종료하는 기능

## def calculateLevelAndFallFreq(score):

```
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
```

점수를 기반으로 현재 레벨에 따라 블록이 떨어지는 속도 계산 후 반환

## def assignColor(shape): *추가한 함수

```
def assignColor(shape):
    if shape == 'S':
        return 0
    elif shape == 'Z':
        return 1
    elif shape == 'J':
        return 2
    elif shape == 'L':
        return 3
    elif shape == 'I':
        return 4
    elif shape == 'O':
        return 5
    elif shape == 'T':
        return 6
```

주어진 모양에 따라 색상을 할당

## def getNewPiece():

```
def getNewPiece():
    # return a random new piece in a random rotation and color
    shape = random.choice(list(PIECES.keys()))
    color = assignColor(shape)
    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': color}
    return newPiece
```

새로운 테트리스 블록을 생성. 랜덤한 모양과 무작위 회전 상태로 생성함

## def addToBoard(board, piece):

```
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']
```

블록을 현지 위치와 모양, 회전 상태에 따라 보드에 추가

## def getBlackBoard():

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

빈 보드 반환

## def isOnBoard(x,y):

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

주어진 좌표가 보드 내에 있는지 확인하기 위한 함수

## def isValidPosition(board, piece, adjX=0, adjY=0):

```
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
```

블록이 보드 안에 있고 충돌하지 않는지 확인하는 함수

## def isCompleteLine(board, y):

```
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
```

특정 행이 블록으로 꽉 찼는지 확인하는 함수

## def removeCompleteLines(board):

```
def removeCompleteLines(board):
    # Remove any completed lines on the board, move everything d
    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 s
            # This is so that if the line that was pulled down i
            # complete, it will be removed.
        else:
            y -= 1 # move on to check next row up
    return numLinesRemoved
```

완성된 행을 제거하고 위의 블록을 한 줄 아래로 이동, 제거된 행의 수를 반환

## def convertToPixelCoords(boxx, boxy):

```
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))
```

보드의 xy좌표를 xy좌표로 반환하는 함수

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

```
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))
```

개별 블록을 화면에 그리는 함수

## def drawBoard():

```
def drawBoard(board):
    # draw the border around the board
    pygame.draw.rect(DISPLAYSURF, BORDERCOLOR, (XMARGIN - 3, TOPMARGIN - 7, (BOARDWIDTH * BOXSIZE) + 8, (BOARDHEIGHT * BOXSIZE) + 8), 5)

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

보드의 배경과 테두리, 각 블록을 그리는 함수

## def calculate_play_time(start_time): *추가한 함수

```
def calculate_play_time(start_time):
    # 현재 시간을 가져옵니다.
    current_time = time.time()

    # 시작 시간과 현재 시간의 차이를 계산합니다.
    play_time = current_time - start_time

    return int(play_time)
```

게임 시작 시간과 현재 시간의 차이를 계산하여 플레이 시간을 정수형으로 반환

## def drawStatus(score, level, play_time) *수정한 함수

```
def drawStatus(score, level, play_time):
    # draw the score text
    scoreSurf = BASICFONT.render('Score: %s' % score, True, TEXTCOLOR)
    scoreRect = scoreSurf.get_rect()
    scoreRect.topleft = (WINDOWWIDTH - 150, 20)
    DISPLAYSURF.blit(scoreSurf, scoreRect)

    # draw the level text
    levelSurf = BASICFONT.render('Level: %s' % level, True, TEXTCOLOR)
    levelRect = levelSurf.get_rect()
    levelRect.topleft = (WINDOWWIDTH - 150, 50)
    DISPLAYSURF.blit(levelSurf, levelRect)

    # draw the elapsed time
    timeSurf = BASICFONT.render('Play Time: %s sec' % play_time, True, TEXTCOLOR)
    timeRect = timeSurf.get_rect()
    timeRect.topleft = (WINDOWWIDTH - 600, 80)
    DISPLAYSURF.blit(timeSurf, timeRect)
```

## def drawPiece(piece, pixelx=None, pixely=None):

현재 점수, 레벨, 플레이 시간을 화면에 표시

```
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))
```

주어진 블록을 화면에 그림

## def drawNextPiece(piece):

```
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)
```

다음의 등장할 블록을 화면에 표시

# iii) 함수의 호출 순서와 조건

## 1. main 함수

main 함수는 게임의 시작점입니다.
호출 조건: 프로그램 실행 시.

- 호출 순서:
    1. pygame.init(): Pygame을 초기화합니다.
    2. FPSCLOCK = pygame.time.Clock(): 프레임 레이트 관리를 위한 시계 객체를 생성합니다.
    3. DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT)): 게임 화면을 설정합니다.
    4. BASICFONT = pygame.font.Font('freesansbold.ttf', 18): 기본 폰트를 로드합니다.
    5. BIGFONT = pygame.font.Font('freesansbold.ttf', 100): 큰 폰트를 로드합니다.
    6. pygame.display.set_caption('2019006608_ChoiMyeongHak'): 게임 창의 제목을 설정합니다.
    7. showTextScreen('MY TETRIS'): 시작 화면을 표시합니다.
    8. 랜덤한 음악 파일을 선택하여 재생합니다.
    9. runGame(start_time): 게임 루프를 시작합니다.
    10. pygame.mixer.music.stop(): 음악 재생을 중지합니다.
    11. showTextScreen('Game Over'): 게임 오버 화면을 표시합니다.

## 2. showTextScreen 함수

게임 시작 또는 종료 화면을 표시합니다.
호출 조건: 게임 시작 또는 종료 시.

- 호출 순서:
    1. makeTextObjs(text, BIGFONT, TEXTCOLOR): 텍스트 객체를 생성합니다.
    2. 텍스트를 화면 중앙에 그림.
    3. 'Press a key to play.' 텍스트를 추가로 그림.
    4. checkForKeyPress(): 키 입력을 대기합니다.
    5. Pygame 화면을 업데이트합니다.

## 3. checkForKeyPress 함수

키 입력을 확인합니다.
호출 조건: showTextScreen 함수에서 키 입력을 기다릴 때.

- 호출 순서:
    1. checkForQuit(): 종료 이벤트를 처리합니다.
    2. Pygame 이벤트 큐에서 KEYDOWN 및 KEYUP 이벤트를 확인합니다.
    3. KEYUP 이벤트가 발생하면 해당 키를 반환합니다.
    4. 그렇지 않으면 None을 반환합니다.

## 4. checkForQuit 함수

종료 이벤트를 처리합니다.

호출 조건: 키 입력 또는 이벤트 처리 시.

- 호출 순서:
    1. Pygame 이벤트 큐에서 QUIT 이벤트를 확인합니다.
    2. terminate()를 호출하여 게임을 종료합니다.
    3. Pygame 이벤트 큐에서 KEYUP 이벤트를 확인합니다.
    4. Esc 키가 눌리면 terminate()를 호출합니다.
    5. 다른 KEYUP 이벤트는 다시 이벤트 큐에 넣습니다.

## 5. runGame 함수

게임의 주요 루프를 실행합니다.
호출 조건: main 함수에서 게임이 시작될 때.

- 호출 순서:
    1. getBlankBoard(): 빈 보드를 생성합니다.
    2. 시간 관련 변수를 초기화합니다.
    3. 이동 관련 플래그 변수를 초기화합니다.
    4. 점수와 레벨을 초기화합니다.
    5. calculateLevelAndFallFreq(score): 초기 레벨과 블록 낙하 주기를 계산합니다.
    6. getNewPiece()를 두 번 호출하여 현재 블록과 다음 블록을 생성합니다.
    7. 게임 루프를 시작합니다.
        - 호출 조건: 매 프레임마다.
        - 호출 순서:
            1. fallingPiece가 없으면 새로운 블록을 생성하고 초기화합니다.
            2. checkForQuit(): 종료 이벤트를 처리합니다.
            3. Pygame 이벤트 큐를 처리하여 키 입력을 처리합니다.
            4. 블록의 이동 및 회전을 처리합니다.
            5. 블록이 자동으로 떨어지게 합니다.
            6. isValidPosition()을 호출하여 블록의 유효성을 검사합니다.
            7. addToBoard(board, fallingPiece): 블록을 보드에 추가합니다.
            8. removeCompleteLines(board): 완성된 라인을 제거하고 점수를 업데이트합니다.
            9. 화면을 업데이트합니다.

## 6. getNewPiece 함수

무작위 모양과 색상의 새로운 블록을 생성합니다.
호출 조건: 새로운 블록이 필요할 때.

- 호출 순서:
    1. 무작위 모양을 선택합니다.
    2. assignColor(shape): 모양에 따라 색상을 할당합니다.
    3. 새로운 블록 객체를 생성합니다.

## 7. assignColor 함수

모양에 따라 색상을 할당합니다.
호출 조건: 새로운 블록이 생성될 때.

- 호출 순서:
    1. 모양에 따라 고유한 색상 값을 반환합니다.

### 8. calculateLevelAndFallFreq 함수

점수에 따라 레벨과 블록 낙하 주기를 계산합니다.
호출 조건: 점수가 변경될 때.

- 호출 순서:
    1. 점수를 기반으로 레벨을 계산합니다.
    2. 레벨에 따라 낙하 주기를 계산합니다.

### 9. addToBoard 함수

블록을 보드에 추가합니다.
호출 조건: 블록이 최종 위치에 도달했을 때.

- 호출 순서:
    1. 블록의 각 셀을 보드에 추가합니다.

### 10. isValidPosition 함수

블록이 유효한 위치에 있는지 확인합니다.
호출 조건: 블록의 이동 또는 회전 시.

- 호출 순서:
    1. 블록이 보드 내에 있는지 확인합니다.
    2. 블록이 다른 블록과 충돌하지 않는지 확인합니다.

### 11. removeCompleteLines 함수

완성된 라인을 제거하고 점수를 업데이트합니다.
호출 조건: 블록이 최종 위치에 도달했을 때.

- 호출 순서:
    1. 완성된 라인을 제거하고 위의 블록을 아래로 이동시킵니다.
    2. 제거된 라인의 수를 반환합니다.

### 12. drawBoard 함수

보드를 화면에 그립니다.
호출 조건: 매 프레임마다.

- 호출 순서:
    1. 보드의 경계를 그립니다.
    2. 보드 배경을 채웁니다.
    3. 각 셀을 그립니다.

### 13. drawPiece 함수

블록을 화면에 그립니다.
호출 조건: 매 프레임마다, 새로운 블록이 생성될 때.

- 호출 순서:
    1. 블록의 각 셀을 그립니다.

### 14. drawStatus 함수

점수, 레벨 및 플레이 시간을 화면에 표시합니다.

호출 조건: 매 프레임마다.

- 호출 순서:
    1. 점수를 그립니다.
    2. 레벨을 그립니다.
    3. 플레이 시간을 그립니다.

### 15. drawNextPiece 함수

다음 블록을 화면에 그립니다.
호출 조건: 매 프레임마다.

- 호출 순서:
    1. 'Next:' 텍스트를 그립니다.
    2. 다음 블록을 그립니다.

### 16. makeTextObjs 함수

텍스트 표면과 직사각형을 생성합니다.
호출 조건: 텍스트를 화면에 그릴 때.

- 호출 순서:
    1. 텍스트를 렌더링합니다.
    2. 텍스트 표면과 직사각형 객체를 반환합니다.

### 17. terminate 함수

게임을 종료합니다.
호출 조건: 종료 이벤트가 발생할 때.

- 호출 순서:
    1. Pygame을 종료합니다.
    2. 시스템을 종료합니다.

### 18. convertToPixelCoords 함수

보드 좌표를 화면 좌표로 변환합니다.
호출 조건: 블록을 화면에 그릴 때.

- 호출 순서:
    1. 보드 좌표를 화면 좌표로 변환합니다.

### 19. drawBox 함수

개별 블록을 화면에 그립니다.
호출 조건: 블록을 화면에 그릴 때.

- 호출 순서:
    1. 개별 블록을 그립니다.

### 20. isOnBoard 함수

주어진 좌표가 보드 내에 있는지 확인합니다.
호출 조건: 블록의 이동 또는 회전 시.

- 호출 순서:
    1. 좌표가 보드 내에 있는지 확인합니다.

### 21. isCompleteLine 함수

특정 행이 완성되었는지 확인합니다.
호출 조건: 블록이 최종 위치에 도달했을 때.

- 호출 순서:
    1. 특정 행이 완성되었는지 확인합니다.

### 22. getBlankBoard 함수

빈 보드를 생성합니다.
호출 조건: 게임이 시작될 때.

- 호출 순서:
    1. 빈 보드를 생성합니다.

### 23. calculate_play_time 함수

플레이 시간을 계산합니다.
호출 조건: 게임 루프에서 시간 정보를 업데이트할 때.

- 호출 순서:
    1. 현재 시간을 가져옵니다.
    2. 시작 시간과 현재 시간의 차이를 계산합니다.

# Conclusion

해당 오픈소스 소프트웨어를 활용하여 기존 프로젝트를 수정하는 과제를 수행함으로써 오픈소스의 개념과 이점을 이해할 수 있었습니다. 기존의 코드를 분석하고 수정하는 과정을 통해 코드의 작동 원리를 좀 더 손쉽게 이해하고, 코딩 구조를 이해하며 기술이 향상됨을 느꼈습니다. 1,2,3,4,5 번 문제는 코드를 깊게 보지 않아도 충분히 해결할 수 있었으나, 6번 문제의 모양에 따른 고유 색상을 부여하는 과정에서 문제가 발생했습니다. 문제를 해결하는 과정에서 막히는 부분이 있었는데, AI에게 질문해도 잘 해결하지 못하는 모습을 보였습니다.

Dictionary를 수정해 모양마다 새로운 색상 key를 부여하라는 AI의 답변은 기존 함수들을 꼬이게 만들고, 수정해야 할 것이 너무 많아 더 쉬운 방법이 있지 않을까 고민했습니다. def getNewPiece() 함수를 분석했는데 무작위 모양과 무작위 색상을 부여하는 부분에서 color은 color 리스트의 길이에서 무작위 정수로 부여하는 것을 확인했고, 무작위 shape가 결정되면 그 shape 에 따라 정수를 정해주면 되지 않을까 라는 아이디어가 떠올라 필요한 함수를 추가하고 실행했더니 너무도 간단히 해결되는 것을 확인했습니다.

기존 코드의 구조와 순서를 분석해 문제인 부분을 찾고, 해당 부분을 기존 자료형을 해치지 않는 형태로 내가 원하도록 넣기 위해 고민하여 좋은 결과를 낼 수 있었던 것 같습니다. 오픈소스를 활용하여 내가 원하는 것을 쉽게 얻을 수 있지만, 온전히 내 것으로 만들려면 최소한의 분석과 코드에 대한 이해가 필요하다는 것을 느꼈습니다. 기존의 오픈소스 프로젝트를 수정하고 확장하는 과정에서 소프트웨어 개발에 대한 통찰력과 자신감을 얻을 수 있었던 좋은 경험이었습니다.

Github

https://github.com/cmhcmh112/osw

Reference
#테트로미노
https://inventwithpython.com/pygame/chapter7.html