2021008759 고제성 오픈소스 과제중심 1주차 보고서

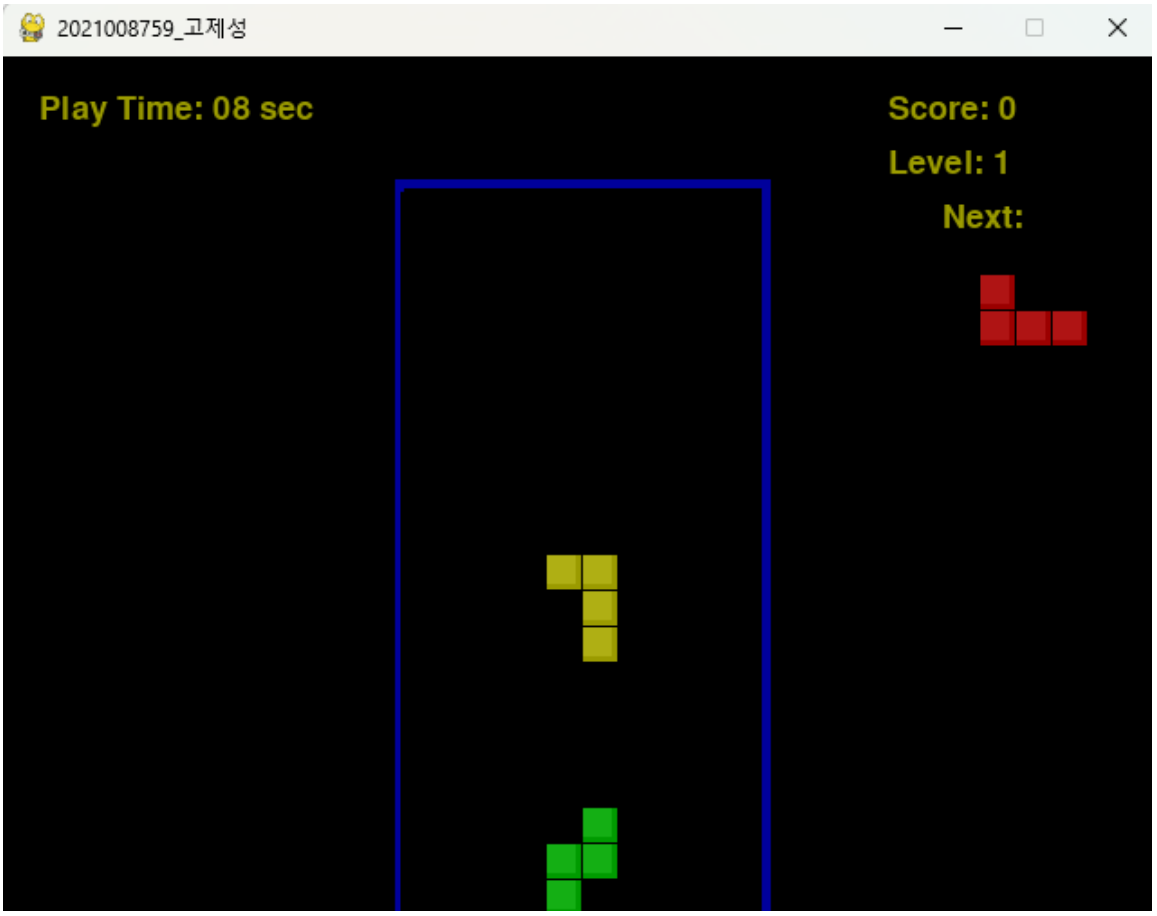
깃허브 주소 : https://github.com/kojesung/osw_TETROMINO

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

```
def main():
    global FPSCLOCK, DISPLAYSURF, BASICFONT, BIGFONT#해당 변수들을 초기화하고 게임을 시작함
    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('2021008759_고제성')#####################상태 창 이름 수정

    showTextScreen('MY TETRIS')###############문구를 띄우는 함수인데 시작할 때 띄우는 부분이니 이 부분을 수정해 시작화면 문구 변경
    while True: # game loop
        startTimer()#############게임 시작할 때마다 시작 시간 초기화하기 위해서 사용
        if random.randint(0, 1) == 0:
            pygame.mixer.music.load('Hover.mp3')#########################배경음악 설정
        else:
            pygame.mixer.music.load('Hover.mp3')#########################배경음악 설정
        pygame.mixer.music.play(-1, 0.0)
        runGame()
        pygame.mixer.music.stop()
        showTextScreen('Game Over')
```

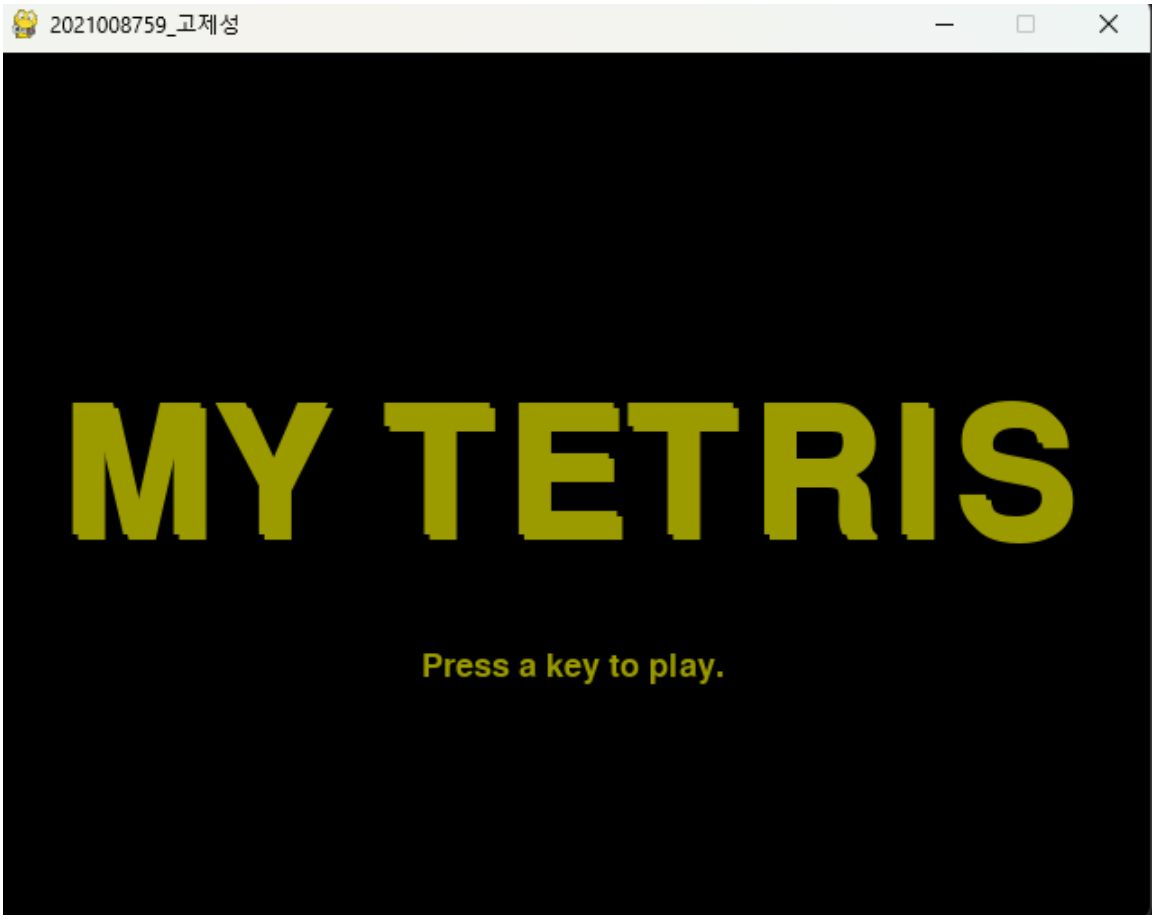
2. **상태창 이름을 학번_이름 으로 수정**

   main() 함수 안에서 pygame.display.set_caption() 안에 인자로 학번_이름 지정하여 수정하였음



3. **게임시작화면의 문구를 MY TETRIS으로 변경**

   main() 함수 안에서 게임이 시작될 때 호출되는 ShowTextScreen()안에 'MY TETRIS'를 넣어 시작 문구 변경



4. **게임시작화면의 문구 및 배경색을 노란색으로 변경**

   YELLOW로 고정된 변수 선언해두고

```
TEXTCOLOR = YELLOW##############################글자 색 노란색으로 수정
TEXTSHADOWCOLOR = YELLOW########################글자 그림자 색 노란색으로 수정
```

   ShowTextScreen함수에서 고정된 값으로 사용

```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()
```

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

```python
# 새로운 함수 추가: 시작 시간을 저장하는 전역 변수
startTime = None

# 게임이 시작될 때 시작 시간을 초기화하는 함수
def startTimer():
    global startTime
    startTime = time.time()

# 경과 시간을 초 단위로 반환하는 함수
def getElapsedTime():
    global startTime
    if startTime is None:
        return 0
    return int(time.time() - startTime)

# 경과 시간을 게임 화면에 표시(그리는)하는 함수 뒤에 다른 정보들 그리는 함수 호출하는 곳에서 함께 호출됨
def drawTimer():
    # 경과 시간을 초 단위로 가져옴
    elapsedTime = getElapsedTime()
    # 초를 분과 초로 변환
    minutes = elapsedTime // 60###분 단위도 추가하고 싶으면 추가할 수 있음
    seconds = elapsedTime % 60
    # 시간을 화면에 출력
    timeSurf = BASICFONT.render('Play Time: {:02d} sec'.format(seconds), True, TEXTCOLOR)##pygame에서 제공하는 메서드로 렌더링
    timeRect = timeSurf.get_rect()##사각형 크기의 객체로 반환
    timeRect.topleft = (20, 20)#위치 설정
    DISPLAYSURF.blit(timeSurf, timeRect)#앞서 선언한 위치에 렌더링하도록 지정
```

게임이 시작될 때 시간을 초기화하는 startTimer()함수는 main()함수 안에서 호출됨

시간을 표시하는 drawTimer()함수는 게임이 실행되는 while loop안에서 drawBoard(), drawStatus(), drawNextPiece 등 게임에 필요한 요소들을 그리는 곳에서 함께 호출됨

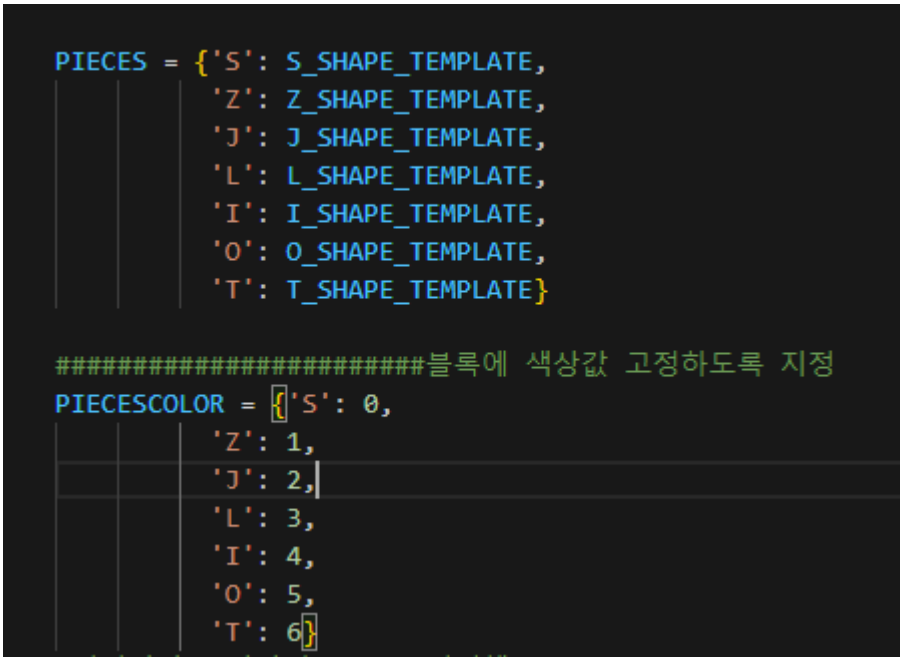
```
    # drawing everything on the screen
    DISPLAYSURF.fill(BGCOLOR)
    drawBoard(board)
    drawStatus(score, level)
    drawNextPiece(nextPiece)
    if fallingPiece != None:
        drawPiece(fallingPiece)

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

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

   일반 색상은 기본색, LIGHT색상은 배경색으로 지정하는 코드임을 확인했고 블럭마다 고유의 색상을 가지기 위해서는 3가지 색상을 추가해야되기 때문에 PURPLE, ORANGE, PINK색상을 LIGHT색과 함께 추가 후 딕셔너리를 만들어 블럭별로 매칭시킴

   → getNewPiece() 함수에서랜덤으로 바뀌는 shape에 매칭되는 색상을 고정시켜 지정해줌

```
#               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)
PURPLE      = (180, 85, 162)
LIGHTPURPLE = (200, 105, 182)
ORANGE      = (255, 127, 0)
LIGHTORANGE = (255, 147, 20)
PINK        = (255, 192, 203)
LIGHTPINK = (255, 212, 223)################블록 개수와 색의 개수 맞추기 위해 PURPLE,ORANGE,PINK(LIGHT도 각각)추가하였음
#색깔 정의
BORDERCOLOR = BLUE
BGCOLOR = BLACK
TEXTCOLOR = YELLOW##############################글자 색 노란색으로 수정
TEXTSHADOWCOLOR = YELLOW##############################글자 그림자 색 노란색으로 수정
COLORS      = (     BLUE,      GREEN,      RED,      YELLOW, PURPLE, ORANGE, PINK)################블록 개수와 맞추기 위해 튜플에 자료 추가
LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYELLOW, LIGHTPURPLE, LIGHTORANGE, LIGHTPINK)
assert len(COLORS) == len(LIGHTCOLORS) # each color must have light color
```



```
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}

########################블록에 색상값 고정하도록 지정
PIECESCOLOR = {'S': 0,
          'Z': 1,
          'J': 2,
          'L': 3,
          'I': 4,
          'O': 5,
          'T': 6}
```

```
def getNewPiece():
    # return a random new piece in a random rotation with fixed 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': PIECESCOLOR[shape]}###########색상값을 직접 지정해주기 위해 랜덤으로 설정된 도형에 맞춘 색깔 지정
    return newPiece
```

주요 함수 역할 및 함수의 호출 순서 및 호출 조건에 대한 설명

주요한 함수지만 위에서 설명하여 중복되는 함수들의 설명은 생략하겠습니다.

main()

```
def main():
    global FPSCLOCK, DISPLAYSURF, BASICFONT, BIGFONT  # 전역 변
    pygame.init()  # Pygame 초기화
    FPSCLOCK = pygame.time.Clock()  # Pygame Clock 객체 생성
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOW
    BASICFONT = pygame.font.Font('freesansbold.ttf', 18)  # 기
    BIGFONT = pygame.font.Font('freesansbold.ttf', 100)  # 큰
    pygame.display.set_caption('2021008759_고제성')  # 창 제목 설

    showTextScreen('MY TETRIS')  # 시작 화면 표시
    while True:  # 게임 루프
        startTimer()  # 게임 타이머 초기화
```

```
        if random.randint(0, 1) == 0:
            pygame.mixer.music.load('Hover.mp3')  # 배경 음악 로
        else:
            pygame.mixer.music.load('Hover.mp3')  # 배경 음악 로
        pygame.mixer.music.play(-1, 0.0)  # 배경 음악 재생
        runGame()  # 게임 실행
        pygame.mixer.music.stop()  # 배경 음악 정지
        showTextScreen('Game Over')  # 게임 오버 화면 표시

'''
전체 게임의 초기화 및 실행을 담당,
게임의 프레임 속도 조절하는 FPSCLOCK
게임 창 설정하는 DISPLAYSURF
폰트 설정하는 BASICFONT, VIGFONT
등을 초기화하고
게임을 while True:를 통해 루프 안에서 실행시키도록 함
'''
```

showTextScreen()

```
def showTextScreen(text):
    # 이 함수는 주어진 텍스트를 화면 중앙에 큰 글씨로 표시하고,
    # 사용자가 키를 누를 때까지 대기
    # 큰 텍스트 객체와 그림자 텍스트 객체를 생성
    titleSurf, titleRect = makeTextObjs(text, BIGFONT, TEXTSH
    titleRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGH
    DISPLAYSURF.blit(titleSurf, titleRect)

    # 큰 텍스트 객체를 실제 텍스트 색상으로 다시 생성
    titleSurf, titleRect = makeTextObjs(text, BIGFONT, TEXTCO
    titleRect.center = (int(WINDOWWIDTH / 2) - 3, int(WINDOWH
    DISPLAYSURF.blit(titleSurf, titleRect)

    # "Press a key to play." 텍스트 객체를 생성
    pressKeySurf, pressKeyRect = makeTextObjs('Press a key to
    pressKeyRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHE
    DISPLAYSURF.blit(pressKeySurf, pressKeyRect)
```

```
    # 사용자가 키를 누를 때까지 대기
    while checkForKeyPress() == None:
        pygame.display.update()
        FPSCLOCK.tick()

#인자로 받은 값들을 렌더링시켜줌
while checkForKeyPress() == None:
    키가 눌릴때까지 update()호출하여 화면 업데이트,

checkForKeyPress()에서 호출하는 checkForQuit()는
QUIT이벤트 발생, ESC누름이 발생하면 게임 종료, 다른 KEYUP 발생 시 이벤트
```

runGame()

```
getBlankBoard()로 현재 보드 초기화
calculateLevelAndFallFreq(score)로 score를 통해 현재 레벨과 떨어지
getNewPiece()로 현재 및 새로운 블록의 모양, 회전상태, 색깔, 위치를 지정
while True:
    if fallingPiece == None:
        drawPiece()함수 호출 #새로운 블록의 모양을 담은 변수의 내용을
        if not isValidPosition(board, fallingPiece):
            return
'''
drawPiece()함수는 블록을 그리는 함수이고 해당 함수 안에 있는
convertToPixelCoords(piece['x'], piece['y']) 함수는 보드 좌표를 픽
'''
'''
isValidPosition()은 블록(piece)이 보드(board) 내에 있으며 다른 블록과
반복문을 사용하여 블록의 각 셀 검사:
    if 블록의 셀이 보드 내에 있는지
    if 블록의 셀이 이미 채워진 셀과 충돌하는지
    모든 셀이 유효하면 True 반환
'''
checkForQuit()
for event in pygame.event.get():
    KEYUP 이벤트 처리
        P 키: 게임을 일시 정지
        왼쪽, 오른쪽, 아래쪽 키: 해당 키가 떼어졌을 때,False로 설정하여
```

```
    KEYDOWN 이벤트 처리
        왼쪽 키: 블록 왼쪽으로 이동, 유효한 위치인지 확인한 후 이동##유효
        오른쪽 키: 블록 오른쪽으로 이동, 유효한 위치인지 확인한 후 이동##
        위쪽 키: 블록 회전, 회전 후 위치가 유효하지 않으면 원래 상태로##
        Q 키: 블록을 반대 방향으로 회전, 회전 후 위치가 유효하지 않으면
        아래쪽 키: 블록을 더 빠르게 아래로 이동, 이동 후 위치가 유효하지
        스페이스바: 블록을 즉시 아래로
블럭 좌 우로 움직이는 구문
if time.time() - lastFallTime > fallFreq:#떨어질 시간이 되면 블럭이
    if not isValidPosition(board, fallingPiece, adjY=1):
        addToBoard(board, fallingPiece)#보드에 고정
        score += removeCompleteLines(board)#지우는 보드만큼 점수에
'''
removeCompleteLines()는
isCompleteLine()함수로 지워지는지 판별하고 라인을 지우는 역할
'''

drawBoard(board) #보드의 현재 상태를 그림
drawStatus(score, level) #점수와 레벨 그림
drawNextPiece(nextPiece) #다음에 나올 블럭 그림
if fallingPiece != None:# fallingPiece가 None이 아니면 drawPiece
  drawPiece(fallingPiece)
drawTimer()#경과 시간을 화면에 표시
```

추가로 piece를 새로 그리거나 Board에 무언가를 추가하는 경우 drawBox()함수를 호출하여 그림

```
if color == BLANK:
  return
#color로 받은 인자 값이 BLANK('.')이 아닌 경우에 그림을 그림(빈공간 확인
if pixelx == None and pixely == None:
    pixelx, pixely = convertToPixelCoords(boxx, boxy)
#pixelx와 pixely가 None이면, convertToPixelCoords 함수를 사용하여
```