과제 중심 수업 1주차 보고서 2023031694 김서현

1. 배경 음악이 주어진 3개 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('2023031694 KIMSEOHYEON')
    music_files = ['Hover.mp3', 'Our_Lives_Past.mp3', 'Platform_9.mp3']

    # 음악 파일 중 하나를 임의로 선택
    music_file = random.choice(music_files)

    # 선택된 음악 파일을 로드하여 재생
    pygame.mixer.music.load(music_file)
    pygame.mixer.music.play(-1, 0.0)


    showTextScreen('MY TETRIS')
    while True:
            runGame()
            pygame.mixer.music.stop()
            showTextScreen('Over :(')
```

오픈소스 코드 초반에 정의된 main 함수 하위에 music_files = ['Hover.mp3', 'Our_Lives_Past.mp3', 'Platform_9.mp3'] 코드를 추가하여 정의한 후 music_file = random.choice(music_files) 코드를 추가해 music_files 안에 랜덤하게 한 개를 실행할 수 있도록 해줍니다. 이 때 음악 파일을 재생하기 위하여 pygame.mixer.music.load(music_file)과 pygame.mixer.music.play(-1, 0.0) 코드를 추가합니다. pygame.mixer.music.load(music_file) 코드는 music_file 변수에 저장된 음악 파일을 로드합니다. 즉, 해당 음악 파일을 메모리에 로드하여 재생할 준비를 합니다. pygame.mixer.music.play(-1, 0.0) 코드는 로드된 음악 파일을 재생합니다. 첫 번째 인수 -1은 음악을 반복 재생하도록 지정합니다. 두 번째 인수 0.0은 재생을 시작할 시간을 지정합니다. 여기서는 음악을 처음부터 재생하므로 0.0으로 설정됩니다. pygame.mixer.music.load() 함수와 pygame.mixer.music.play() 함수는 모두 pygame 라이브러리의 mixer 모듈에서 제공하는 함수입니다.

함수 호출 순서를 정리해보자면, 다음과 같습니다. 먼저 pygame.init() 함수가 호출되어 pygame 라이브러리가 초기화됩니다. 그 다음에는 음악 파일들의 경로가 정의된 리스트인 music_files 변수가 정의됩니다. 그 후에 random.choice() 함수를 사용하여 music_files 리스트에서 임의의 음악 파일을 선택합니다. 선택된 음악 파일은 pygame.mixer.music.load() 함수를 사용하여 메모리에 로드됩니다. 마지막으로 선택된 음악 파일이 pygame.mixer.music.play() 함수를 사용하여 반복 재생됩니다. 또한, 음악이 재생되기 위해서는 주피터의 /tetromino/ 파일 안에 해당 음악 파일 3개가 있어야 합니다.

2. 상태창 이름을 학번 이름으로 바꾸기

```
pygame.display.set_caption('2023031694 KIMSEOHYEON')
```

```
def main():
    global FPSCLOCK, DISPLAYSURF, BASICFONT, BIGFONT
```

pygame.init() 하위에 pygame.display.set_caption('2023031694 KIMSEOHYEON') 으로 코드를 수정합니다.

pygame.display.set_caption은 pygame 라이브러리에서 사용되는 함수로, 창의 제목을 설정하는 데 사용됩니다. 이 함수는 Pygame 창의 제목을 변경하고, 이러한 변경은 창 표시 표시줄에 표시됩니다. 따라서 기존의 Tetromino를 학번과 이름으로 수정할 수 있습니다.

3. 게임 시작 화면 문구 바꾸기

```
showTextScreen('MY TETRIS')
```

또한 화면에 뜨는 이름을 showTextScreen 코드를 수정하여 'MY TETRIS' 로 바꿔주었습니다. 주어진 텍스트를 화면 중앙에 표시하고, 플레이어가 키를 누를 때까지 대기합니다.

```
pressKeySurf, pressKeyRect = makeTextObjs('Press any key to play! Pause key is P', BASICFONT, TEXTCOLOR)
```

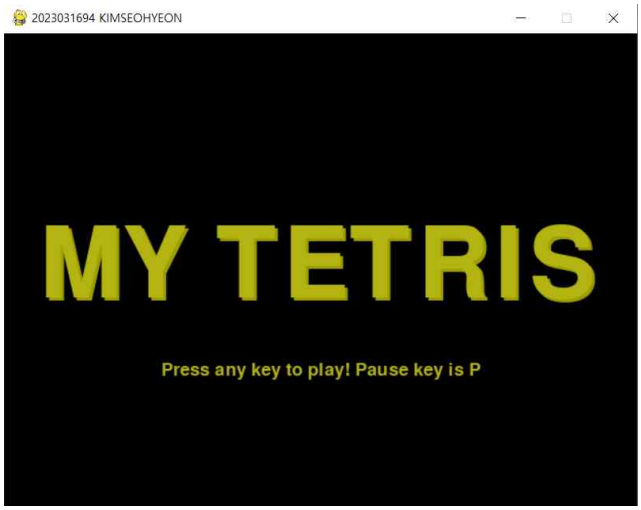
makeTextObjs 함수를 호출하여 주어진 텍스트("Press any key to play! Pause key is P"), 폰트(BASICFONT), 그리고 텍스트 색상(TEXTCOLOR)을 사용하여 텍스트 서페이스(pressKeySurf)와 그에 대한 사각형(pressKeyRect) 객체를 생성합니다. 그리고 텍스트를 화면 중앙 아래쪽에 배치하기 위해 pressKeyRect.center를 설정합니다. DISPLAYSURF.blit(pressKeySurf, pressKeyRect)를 사용하여 생성된 텍스트 서페이스를 화면에 그립니다.

4. 노란색으로 변경

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

오픈소스 코드 초반의 # 하위에서 텍스트 컬러와 텍스트섀도우컬러를 각각 라이트 옐로우와 옐로우 컬러로 바꿔줍니다.

해당 코드를 모두 추가하고 수정하면 다음과 같은 시작 화면이 나타납니다.



+ 일시 정지 화면과 게임 오버 화면

```
showTextScreen('Get a rest!') # pause until a key press
```

def runGame(): 코드 하위에서 *# Pausing the game* 파트에 있는 showTextScreen('') 코드에서 괄호 안을 Get a rest! 로 수정합니다.

runGame() 함수는 Tetris 게임의 핵심 로직을 실행하는 함수입니다. 이 함수는 게임 루프를 포함하고 있으며, 플레이어의 입력을 처리하고 게임 보드를 갱신하여 플레이어에게 게임 화면을 제공합니다. showTextScreen 함수는 화면에 텍스트를 보여줍니다.

```
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('Over :(')
```

오픈소스 코드에서 *#game loop* 에 대한 부분을 찾으면 While True 문 하위 else 에 걸려있는 showTextScreen() 코드에서 괄호 안을 'Over :(' 로 수정하면 게임이 종료될 때 해당 텍스트가 나타나게 됩니다. While True: 문은 게임의 주 루프입니다. 이 루프는 게임이 종료될 때까지 계속 실행됩니다. if 및 else 구문은 게임의 배경 음악을 설정하는 부분입니다. 해당 조건문으로 선택된 음악은 pygame.mixer.music.load() 을 통해 게임의 배경음악이 되고, pygame.mixer.music.stop() 을 통해 멈추게 됩니다. 게임과 같이 음악이 종료되고 화면에는 오버 표시가 뜨게 됩니다.

5. 게임 경과 시간을 초 단위로 표시

```
def runGame():
    # 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()

    startTime = time.time()
```

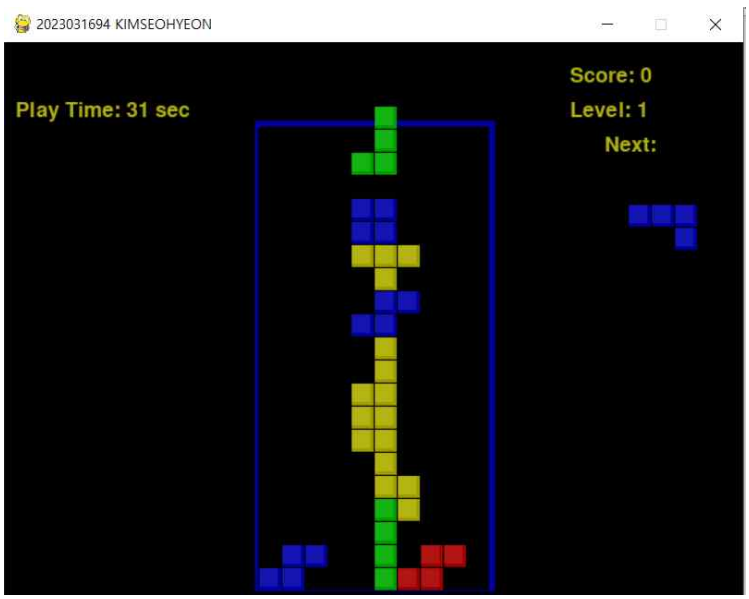
def rungame(): 함수의 시작 부분에 startTime = time.time() 코드를 추가합니다. 해당 코드는 Python의 time 모듈을 사용하여 현재 시간을 초 단위로 측정하고, 이를 startTime 변수에 저장하는 기능을 수행합니다. 이는 게임이 시작될 때마다 경과 시간 측정을 초기화하기 위함입니다. startTime 변수를 현재 시간으로 설정하여 게임이 시작된 시점을 기록합니다. time.time() 함수는 현재 시간을 반환합니다. 또한, startTime 값을 기준으로 현재 시간에서 startTime을 뺀 값을 사용하여 경과 시간을 계산할 수 있습니다.

```
 # 경과 시간 측정 및 업데이트
playTime = time.time() - startTime

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

playTime += FPSCLOCK.get_rawtime() / 1000  # 경과 시간을 초 단위로 변환하여 누적
drawPlayTime(playTime)  # 경과 시간을 화면에 표시
```

따라서 *# drawing everything on the screen* 상위에 Play Time = time.time() - startTime을 추가하여 게임 시작 후 경과 시간인 Play Time을 현재 시간에서 startTime을 뺀 값으로 정의하여 사용합니다. 또한, 하위에는 playTime += FPSCLOCK.get_rawtime() / 1000 코드는 각 프레임마다 FPSCLOCK.get_rawtime()을 호출하여 이전 프레임과 현재 프레임 사이의 경과 시간을 밀리초 단위로 가져옵니다. 가져온 밀리초 값을 1000으로 나누어 초 단위로 변환합니다. 변환된 값을 playTime 변수에 누적하여 총 경과 시간을 계산합니다.drawPlayTime(playTime) 코드는 누적된 playTime 값을 화면에 표시합니다. 이 함수는 playTime을 받아 텍스트 형태로 화면의 적절한 위치에 나타냅니다.

```
def drawPlayTime(playTime):
    # 경과 시간을 화면에 표시
    playTimeSurf = BASICFONT.render('Play Time: {} sec'.format(int(playTime)), True, TEXTCOLOR)
    playTimeRect = playTimeSurf.get_rect()
    playTimeRect.topleft = (10, 50)
    DISPLAYSURF.blit(playTimeSurf, playTimeRect)
```

또한, 아래에 drawPlayTime 함수를 새롭게 추가해 줍니다. 해당 함수에서는 BASICFONT.render를 사용하여 경과 시간을 텍스트로 렌더링합니다. int(playTime)을 사용하여 경과 시간을 정수로 변환합니다. playTimeRect.topleft를 (10, 50)으로 설정하여 텍스트가 화면에 표시되는 위치를 조정하고 DISPLAYSURF.blit(playTimeSurf, playTimeRect)를 사용하여 렌더링된 텍스트를 화면에 그립니다. 위의 코드를 모두 추가하여 수정하면 오른쪽과 같은 화면이 나타나게 됩니다.



6. 7개의 블록이 각각 고유의 색 갖도록 하기

PIECE 딕셔너리 밑에 PIECE_COLORS 딕셔너리를 만들어 각 피스에 할당되는 고유의 색을 지정해줍니다.

```
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}

PIECE_COLORS = {'S': GREEN,
                'Z': RED,
                'J': BLUE,
                'L': YELLOW,
                'I': WHITE,
                'O': GRAY,
                'T': LIGHTBLUE}
```

```
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': PIECE_COLORS[shape]}  # 지정된 색 사용
    return newPiece
```

다음으로는 getnewpiece 함수 하위에 있던 기존의 랜덤 색 지정을 고정된 색을 사용하도록 바꿔줍니다. 'color': random.randint(0, len(COLORS)-1) 코드를 'color': PIECE_COLORS[shape]로 수정하여 PIECE_COLORS 딕셔너리에서 키로 현재 선택된 피스의 모양인 shape를 사용하여 해당 피스의 색을 가져옵니다.

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

또한, drawBox 함수 하위에 있던 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)) 코드를 pygame.draw.rect(DISPLAYSURF, color, (pixelx + 1, pixely + 1, BOXSIZE - 1, BOXSIZE - 1))
pygame.draw.rect(DISPLAYSURF, color, (pixelx + 1, pixely + 1, BOXSIZE - 4, BOXSIZE - 4))로 수정합니다. 이전 코드에서는 COLORS와 LIGHTCOLORS 리스트에서 특정 인덱스 color에 해당하는 색상을 가져와 사용했습니다. 반면에 새로운 코드에서는 함수의 매개변수 color로 직접 전달된 값이 사용됩니다. 따라서 이전 코드에서는 미리 정의된 색상 리스트에서 색상을 가져와 사용했지만, 새로운 코드에서는 함수 호출시 전달된 색상을 사용하여 상자를 그립니다.

```
drawBox(None, None, PIECE_COLORS[piece['shape']], pixelx + (x * BOXSIZE), pixely + (y * BOXSIZE))
```

그리고 def drawPiece 함수 하위에 drawBox 코드에서 기존에 piece['color'] 로 설정된 부분을 PIECE_COLORS[piece['shape']] 로 바꾸어 지정된 색이 나타나도록 해줍니다. drawBox 함수는 게임 화면에서 하나의 박스(블록)을 그리는 함수입니다. 각 피스는 여러 개의 박스로 이루어져 있으며, drawBox 함수는 각각의 박스를 그리는 역할을 합니다. 해당 코드를 모두 수정하면 각 피스들이 PIECE_COLORS 딕셔너리에 지정된 색상으로 나타나게 됩니다.

GitHub https://github.com/seo0917/osw