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


```
showTextScreen('MY TETRIS')
while True: # game loop
    if random.randint(0, 1) == 0:
        pygame.mixer.music.load('Platform_9.mp3')
    else:
        pygame.mixer.music.load('Platform_9.mp3')
    pygame.mixer.music.play(-1, 0.0)
```

- 음악을 파일에 같이 넣고 music.load의 괄호 안을 고른 음악 파일 이름으로 수정하였다.
- 원래는 두 가지 중 랜덤으로 음악 하나를 실행시키는 것이나 여기선 둘 다 제목을 같게 해 하나만 실행되도록 설정되었다. 이 명령어는 음악 파일을 로드하는 역할을 한다.
- 게임이 실행되는 동안 음악이 나오므로 먼저 키를 눌러 게임이 시작되는 게 호출 조건이다. 본격적인 게임이 시작되면 바로 음악이 시작된다.

2. 상태창 이름을 학번_이름 으로 수정

```
pygame.display.set_caption('2023028686_YunDain')
```

- 괄호 안을 학번_이름으로 수정하여 상태창에 뜨도록 만들었다.
- 이 명령어는 Pygame 라이브러리를 사용하여 창의 제목을 설정한다. 이 함수 호출을 통해 Pygame 창의 제목 표시줄에 나타나는 텍스트를 지정할 수 있다.
- Pygame 실행과 동시에 작동한다.

3. 게임시작화면의 문구를 MY TETRIS으로 변경

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

- 괄호 안을 MY TETRIS로 수정했다.
- showTextScreen이란 함수를 만들어 화면 중앙에 글자가 나타나게 하였고 사용자가 키를 누를 때까지 해당 텍스트를 계속 표시하는 기능을 수행한다. 또, 이 코드는 텍스트에 그림자 효과를 준다. 그 외에 'Press any key to play! Pause key is p' 안내 문구를 첨부한다.
- 함수 코드는 아래에 첨부했다. text, BIGFONT, TEXTSHADOWCOLOR가 유효하고 WINDOWWIDTH와 WINDOWHEIGHT가 설정되고 DISPLAYSURF가 초기화되어 있어야 한다. 각 코드는 특정 조건이 만족될 때 호출되며, 순서대로 실행된다. makeTextObjs 함수를 사용하여 텍스트를 생성하고 위치를 설정한 후, DISPLAYSURF.blit 함수를 사용하여 화면에 그린

다. 마지막으로, 사용자가 키를 누를 때까지 대기하는 루프가 실행된다.

```
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()
```

4. 게임시작화면의 문구 및 배경색을 노란색으로 변경

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

- TEXTCOLOR와 TEXTSHADOWCOLOR를 LIGHTYELLOW와 YELLOW로 지정했다.
- 위에서 설명한 showTextScreen에서 쓰이는 변수로 화면에 뜨는 글자를 노란색으로 표시하는 역할을 한다.
- 변수 자체는 Pygame을 실행하면 값이 부여되며, showTextScreen 함수가 실행될 때 호출된다.

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

```
gameStartTime = time.time()
```

```
    elapsedTime = time.time() - gameStartTime
    drawStatus(score, level, elapsedTime)
```

```
def drawStatus(score, level, elapsedTime):
    # 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)

    elapsedTimeSurf = BASICFONT.render('Play Time: %s sec' % formatTime(elapsedTime), True, TEXTCOLOR)
    elapsedTimeRect = elapsedTimeSurf.get_rect()
    elapsedTimeRect.topleft = (WINDOWWIDTH - 600, 20)
    DISPLAYSURF.blit(elapsedTimeSurf, elapsedTimeRect)
```

- elapsedTime이란 변수를 만들어 score와 level과 같이 drawStatus로 그려지게 하고 게임이 시작된 시간을 왼쪽 화면에 뜨게 했다. 또한, 게임을 새로 시작하면 시간이 초기화되도

록 하였으며 단위를 sec로 표시하였다.

- 게임이 시작된 시간을 기록하고, 그 시간부터 현재까지의 경과 시간을 계산한다. 게임이 시작된 시점을 gameStartTime 변수에 저장한다. 현재 시각에서 gameStartTime을 빼서 경과 시간을 elapsedTime 변수에 저장한다. 이를 통해 게임이 시작된 후 얼마나 시간이 흘렀는지 알 수 있다. formatTime()로 elapsedTime을 형식화하여 문자열로 변환하고 BASICFONT.render()로 형식화된 텍스트를 렌더링하여 표면 객체를 생성한다. elapsedTimeSurf.get_rect()로 텍스트 표면의 사각형 객체를 생성하고 elapsedTimeRect.topleft = ()로 사각형 객체의 위치를 설정하며 DISPLAYSURF.blit(...): 텍스트 표면을 화면에 그린다.
- 함수 호출 순서는 getBlankBoard(), time.time(), calculateLevelAndFallFreq(score), time.time(), getNewPiece()이다. 조건은 각 호출이 필요한 변수와 모듈이 유효하게 정의되어 있어야 한다.

6. 7개의 블록이 각각 고유의 색을 갖도록 코드를 수정하거나 추가

```
def getNewPiece():
    # return a random new piece in a random rotation and color
    shape = random.choice(list(PIECES.keys()))
    COLOR = {'S' :  WHITE,
             'Z' : LIGHTGREEN,
             'J' : RED,
             'L' : GRAY,
             'I' : GREEN,
             'O' : YELLOW,
             'T' : BLUE}
    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[shape]}
    return newPiece
```

- COLOR로 딕셔너리를 만들어 각각 모양으로 key를 만들고 이에 고유 색을 부여했다. 그런 다음 밑의 newPiece의 'color': 부분을 COLOR[shape]로 수정하여 랜덤으로 지정된 모양의 고유 색을 부르게 수정했다.
- shape로 랜덤 모양을 지정하고, COLOR로 그 모양에 고유 색을 딕셔너리로 묶는다. newPiece로 딕셔너리를 만들어 모양, 회전 각도 랜덤, x 위치 고정, y 위치 고정, 고유 색 지정을 하여 새로운 조각을 만든다.
- shape로 먼저 랜덤 모양을 지정한 다음, COLOR 딕셔너리로 고유 색을 부여한다. 그런 다음, newPiece에서 앞의 shape를 호출, rotation을 랜덤으로 부여하고, 사진엔 없지만 위의 BOARDWIDTH와 TEMPLATEWIDTH 변수를 호출하여 x 위치를 고정시킨다. y도 -2로 고정시키고 color에서 앞의 shape 키에 맞는 COLOR의 value 값을 호출하여 이를 모두 딕셔너리로 묶는다. 그런 다음에 이를 리턴한다.

Github 링크

https://github.com/Yundaim/osw