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

- main() 함수의 while문을 다음과 같이 수정

```
while True: # game loop
        choice = random.randint(0, 2)
        if choice == 0:
            pygame.mixer.music.load('Hover.mp3')
        elif choice == 1:
            pygame.mixer.music.load('Our_Lives_Past.mp3')
        else:
            pygame.mixer.music.load('Platform_9.mp3')
```

2. 상태창 이름을 학번_이름으로 수정

- main() 함수의 pygame.display.set_caption('Tetromino') 부분을 다음과 같이 수정

```
pygame.display.set_caption('2021046117_신승민')
```

3. 게임시작화면의 문구를 MY TETRIS으로 변경

- main() 함수의 showTextScreen('Tetromino') 부분을 다음과 같이 수정

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

4. 게임시작화면의 문구 및 배경색을 노란색으로 변경

TEXTCOLOR = WHITE

TEXTSHADOWCOLOR = GRAY

- 해당 코드를 다음과 같이 수정

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

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

- def runGame() 함수 수정 내용

startTime = time.time() 추가

drawing everything on the screen 하단에

elapsedTime = time.time() – startTime를 추가하고

drawStatus(score, level)을 drawStatus(score, level, elapsedTime)로 수정

- def drawStatus() 함수 수정 내용

drawStatus(score, level)을 drawStatus(score, level, elapsedTime)로 수정하고

```
    timeSurf = BASICFONT.render('Time: %s' % int(elapsedTime), True, TEXTCOLOR)
    timeRect = timeSurf.get_rect()
    timeRect.topleft = (20, 20)
    DISPLAYSURF.blit(timeSurf, timeRect)
```

를 새로 추가

6. 7개의 블록이 각각 고유의 색을 갖도록 코드를 수정하거나 추가

- 기존 색상 선언 부분을 다음과 같이 변경

```
WHITE       = (255, 255, 255)
GRAY        = (185, 185, 185)
BLACK       = (  0,   0,   0)
RED         = (155,   0,   0)
GREEN       = (  0, 155,   0)
BLUE        = (  0,   0, 155)
YELLOW      = (155, 155,   0)
CYAN        = (  0, 255, 255)
MAGENTA     = (255,   0, 255)
ORANGE      = (255, 128,   0)
```

- 해당 코드를 삭제

COLORS = (BLUE, GREEN, RED, YELLOW)

LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYELLOW)

assert len(COLORS) == len(LIGHTCOLORS) # each color must have light color

- PIECE_COLORS 새로 선언

PIECE_COLORS = {'S': GREEN,

'Z': RED,

'J': BLUE,

'L': YELLOW,

'I': CYAN,

'O': ORANGE,

'T': MAGENTA}

- def getNewPiece() 함수 수정 내용

'color': random.randint(0, len(COLORS)-1)을

'color': PIECE_COLORS[shape]}로 수정

- def drawBox() 함수 수정 내용

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)) 해당 코드 두 줄을

pygame.draw.rect(DISPLAYSURF, color, (pixelx + 1, pixely + 1, BOXSIZE - 1, BOXSIZE - 1))로 대체

7. 각 함수의 역할 설명

main() 함수

- 게임 루프에 필요한 변수들을 전역 변수로 선언
- 텍스트의 폰트를 설정하고 상태창 이름을 출력
- 시작 화면과 종료 화면 통제
- 무작위로 배경 음악을 재생

runGame() 함수

- 게임 시작을 위해 필요한 변수들을 선언
- 블록들의 이동과 회전, 낙하 속도를 통제
- 게임의 상태를 화면에 출력

makeTextObjs() 함수

- 입력받는 인자를 통해 텍스트를 출력

terminate() 함수

- 게임을 종료하는 목적으로 사용

checkForKeyPress() 함수

- 입력된 키가 KEYDOWN일 경우 계속 진행
- 입력된 키가 KEYUP일 경우 해당 키 반환
- 두 경우 모두 아니면 None 반환

showTextScreen() 함수

- 텍스트의 위치를 화면 중앙으로 설정하고 그림자를 추가
- Press a key to play를 화면 중앙 아래에 출력
- 키 입력을 받으면 다음 동작으로 넘어감

checkForQuit() 함수

- 모든 종료 이벤트를 판단해 terminate() 함수를 호출
- 사용자가 esc 키를 눌렀을 경우에도 terminate() 함수 호출

calculateLevelAndFallFreq() 함수

- 플레이어의 점수를 기반으로 레벨을 산출
- 레벨이 높아질수록 블록이 떨어지는 속도가 늘어남

getNewpiece() 함수

- 무작위 형태와 회전 상태의 블록을 리턴

addToBoard() 함수

- 떨어지는 블록의 형태와 회전 상태, 위치에 따라 하단에 쌓이도록 보드를 업데이트함

getBlankBoard() 함수

- 비어있는 보드를 만들고 이를 리턴함

isOnBoard() 함수

- 보드의 위치가 플레이 가능 영역에 있는지 판단

isValidPosition() 함수

- 블록이 보드 위에 존재하는지 판단

isCompleteLine() 함수

- 주어진 줄이 완전히 채워졌는지 확인

removeCompleteLines() 함수

- 완전히 채워진 줄을 제거, 윗줄을 한줄씩 아래로 이동
- 제거된 줄의 숫자를 리턴

convertToPixelCoords() 함수

- 보드 좌표를 픽셀 좌표로 변환

drawBox() 함수

- 주어진 좌표에 블록을 그린다

drawBoard() 함수

- 보드 주변에 테두리를 그린다
- 보드의 배경을 채운다
- 보드 상태를 화면에 출력

drawStatus() 함수

- 현재 점수, 레벨, 경과 시간을 화면에 표시

drawPiece() 함수

- 그릴 블록의 모양과 회전을 결정

drawNextPiece() 함수

- 다음에 나올 블록의 모양과 회전을 결정

8. 함수의 호출 순서 및 호출 조건에 대한 설명

게임을 실행하면 main()이 가장 먼저 호출되고, 이후 showTextScreen() 함수가 호출된다.

checkForKeyPress() 함수를 통해 입력을 확인하면 다시 main()으로 복귀한다.

이어서 runGame() 함수가 호출되고 getBlankBoard() 함수를 통해 비어 있는 보드를 출력한다.

calculateLevelAndFallFreq() 함수를 통해 level과 fallFreq 변수를 할당하고,

getNewPiece()로 fallingPiece와 nextPiece 변수를 할당한다.

isValidPosition()을 통해 배치의 가능성을 판단하고 불가능하다면 리턴한다.

이 경우 main() 함수로 복귀하여 showTextScreen() 함수를 통해 Game Over 문구를 출력한다.

checkForQuit() 함수는 terminate() 함수를 호출하고 이 경우 게임이 종료된다.

블록이 바닥에 닿은 경우 addToBoard() 함수를 통해 이를 보드에 반영하고,

removeCompleteLines() 함수가 isCompleteLine() 함수를 통해 보드의 한 줄이 완전히 채워졌는지

여부를 확인하고 이어서 calculateLevelAndFallFreq() 함수가 호출되어 해당 결과를 반영한다.

drawBoard(), drawStatus(), drawNextPiece() 함수들은 화면을 업데이트하기에 지속적으로 호출된다.

떨어지는 블록이 있다면 drawPiece() 함수가 호출되어 이를 그려준다.

9. 깃허브 링크

https://github.com/ssm3842/osw