> 체크리스트(과제):

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

현재는 0과 1중 하나가 랜덤으로 생성(*random.randint(0, 1)*)되고 나오는 값에 따라 2개 중 하나의 음악이 재생(*pygame.mixer.music.load()*)된다.

원래 코드:

```
if random.randint(0, 1) == 0:
    pygame.mixer.music.load('tetrisb.mid')
else:
    pygame.mixer.music.load('tetrisc.mid')
```

주어진 노래는 3개이므로 랜덤으로 생성할 번호를 증가(random.randint(0, 2))시키고 재생되는 노래 함수에 파일을 바꾼다.

수정 코드:

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

☑ 2. 상태창 이름을 학번_이름 으로 수정

pygame.display.set_caption()은 상태창의 텍스트를 설정하는 것을 볼 수 있다.

원래 코드:

```
pygame.display.set_caption('Tetromino')
```

적절하게 이름을 수정했다.

수정 코드:

```
pygame.display.set_caption('2023015705 박윤하')
```

☑ 3. 게임시작화면의 문구를 MY TETRIS으로 변경

showTextScreen()은 화면에 텍스트를 출력하는 함수로 현재는 Tetromino를 매개변수로 보내 게임 시작화면(game Loop 전)에 출력하고 있다.

원래 코드:

```
showTextScreen('Tetromino')
```

문구를 MY TETRIS로 수정했다.

수정 코드:

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

☑ 4. 게임시작화면의 문구 및 배경색을 노란색으로 변경

TEXTCOLOR와 TEXTSHADOWCOLOR는 현재 각각 흰색과 회색을 저장해 매개변수로 여러 함수에 활용하고 있다.

원래 코드:

```
TEXTCOLOR = WHITE
TEXTSHADOWCOLOR = GRAY
```

게임시작 화면, 일시정지 화면, 게임오버 화면의 문구를 출력하는 함수인 showTextScreen()에서도 TEXTCOLOR와 TEXTSHADOWCOLOR를 사용하고 있다. 지정된 색깔을 변경해 모든 문구의 색과 배경색을 변경했다.

수정 코드:

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

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

pygame.time.get_ticks()는 현재 시각 값을 밀리미터(mm) 단위로 받아 올 수 있다. start는 게임 시작의 시각을 값으로 지정한다. ‘pygame.time.get_ticks() - start’ (현재시간 - 게임시작시간)으로 경과 시간을, 1000(mm, 즉 1초)를 나눠서 초 단위로 elapsed_time을 계산할 수 있다. ‘timer_message’는 경과 시간을 string타입으로 출력할 문구를 정리한다. ‘BASICFONT.render(timer_message, True, TEXTCOLOR)’에다 BASICFONT로 글자 스타일, 크기 맞게 timer_message 문구 렌더. ‘DISPLAYSURF.blit(..., (20, 20))‘로 DISPLAYSURF의 좌표 (20, 20)에 문구를 출력한다.

추가 코드:

```
start = pygame.time.get_ticks()
## << 생략 >> ##
elapsed_time = (pygame.time.get_ticks() - start) / 1000
timer_message = 'Play Time: ' + str(int(elapsed_time)) + 'sec'
## << 생략 >> ##
DISPLAYSURF.blit(BASICFONT.render(timer_message, True, TEXTCOLOR), (20, 20))
```

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

현재는 4개의 평범한 색과 연한 색을 각각 튜플로 묶어 블록에 랜덤으로 지정해서 그려내고 있다.

원래 코드:

```
COLORS      = (     BLUE,      GREEN,      RED,      YELLOW)
LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYELLOW)
## << 생략 >> ##
```

블록은 총 7개므로 색깔을 3개 더 추가해서 튜플에 저장했다. SHAPE_COLORS라는 딕셔너리를 또 선언해서 각 블록을 키(key), 고유의 색을 값(value)으로 각각 지정했다. 원래 getNewPiece 함수에서 새 블록을 생성할 때마다 색깔을 랜덤으로 지정했다. newPiece의 color 값을 SHAPE_COLORS에 맞는 색을 저장되게 수정했다. drawBox 함수에는 연한 색도 짝이 되는 색으로 다시 선언했다.

추가/수정 코드:

① 색깔 추가하고 7개의 블록이 각각 고유의 색 지정

```
CYAN        = (0, 155, 155)
LIGHTCYAN   = (20, 175, 175)
ORANGE      = (155, 27, 0)
LIGHTORANGE = (175, 47, 20)
PURPLE      = (128, 0, 128)
LIGHTPURPLE = (148, 20, 148)

COLORS      = (BLUE, GREEN, RED, YELLOW, CYAN, ORANGE, PURPLE)
LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYELLOW, LIGHTCYAN, LIGHTORANGE, LIGHTPURPLE)

SHAPE_COLORS = {
    'S': GREEN,
    'Z': RED,
    'J': BLUE,
    'L': ORANGE,
    'I': CYAN,
    'O': YELLOW,
    'T': PURPLE }
```

② getNewPiece 함수 수정

```
def getNewPiece():
    ## << 생략 >> ##
    newPiece = {
                ## << 생략 >> ##
                'color': SHAPE_COLORS[shape] }
    return newPiece
```

③ drawBox 함수 수정

```
def drawBox(boxx, boxy, color, pixelx=None, pixely=None):
    ## << 생략 >> ##
    light_color = LIGHTCOLORS[COLORS.index(color)]
```

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

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

> 함수의 설명 + 함수 호출조건, 순서 설명:

i. main(): 프로그램이 실행될 때 시작화면을 보여준다. 배경음악을 재생하고 runGame()로 게임 시작한다. 게임이 끝날 때 배경음악을 멈추고 게임오버 화면을 보여준다. 무한반복(while True) 속에서 showTextScreen 통해 아무 키를 누르면 게임이 처음부터 다시 시작된다.

ii. runGame()

: 실제 게임을 실행하는 함수이다. 변수 초기화 후에 fallingPiece 변수에 사용자가 움직일 수 있는 떨어지는 블록, nextPiece 변수에는 다음으로 나올 블록을 getNewPiece() 함수(랜덤 블록 생성) 통해 지정한다.

게임 루프(while True)속에 떨어지는 블록이 없다면 nextPiece를 fallingPiece에 넣고 nextPiece에는 새로운 블록을 넣는다. lastFallTime 변수로 떨어지는 데에 시간을 정하기 위해(속도 조정 fallFreq) 현재 시각(time.time())으로 초기화한다. isValidPosition() 함수로 블록이 게임판을 넘어갔는지 확인하고, 넘었으면 runGame()을 리턴해 게임을 끝낸다.

for event in pygame.event.get()는 키를 누르는 이벤트(떨어지는 블록 움직이거나 회전시킬 때, 일시정지 시킬 때)를 처리한다. p를 누를 때 게임 일시정지, 화살표나 wasd 눌러 맞는 방향으로 움직이거나 회전한다. 스페이스(space) 키를 누르면 떨어지는 블록은 바로 밑으로 착지한다. isValidPosition() 함수로 블록이 최대한 떨어질 수 있는 공간을 확인한다. lastFallTime 변수로 블록이 떨어지는 rate를 측정하고 있다.

isValidPosition가 true면 블록이 착지했다는 것이다. 즉, addToBoard() 호출해서 다른 블록이 위에 착지할 수 있게 board 상태를 업데이트해준다. removeCompleteLines()는 완성된 줄을 지워주고 지워진 줄의 계수를 점수에 추가한다. 점수에 따라 블록 떨어지는 rate와 레벨이 바뀌기 때문에 calculateLevelAndFallFreq() 호출해 레벨과 속도를 업데이트시킨다. 떨어지는 블록이 없으므로 fallingPiece = None하고 다음 블록으로 게임 진행한다. 만약 떨어지는 블록이 착지하지 않았다면 착지할 때까지 Y position을 1씩 줄인다.

drawBoard, drawStatus, drawNextPiece, drawPiece는 화면에 실제 게임을 구현하는 함수로 마지막에 pygame.display.update()로 display를 FPSCLOCK.tick(FPS) (잠깐의 텀) 마다 업데이트한다.

iii. checkForKeyPress()

: 이벤트 queue에 KEYUP, KEYDOWN 이벤트를 확인하고 KEYDOWN은 queue에서 뺀다.

iv. showTextScreen(text)

: 제네릭 함수. 받는 매개변수로 개임시작화면, 게임오버화면, 일시정지화면 문구롤 사용되고 추가로 “Press a key to play.”를 출력한다.

v. checkForQuit(): esc 아니면 KEYUP 이벤트를 다시 queue에 저장한다.

vi. calculateLevelAndFallFreq(score): 레벨과 떨어지는 속도를 계산하는 함수

vii. getBlankBoard() : board 초기화

https://github.com/yhpark04/osw