https://github.com/geongeony/osw

목표

pygame을 이용해 '테트로미노' 게임을 구현한 파이썬 코드를 수정하여 게임 화면을 원하는 형태로 바꾼다.

0. 오류 수정
1. 배경음악을 주어진 3개의 음악 중 1개가 재생되도록 수정
2. 시작 화면 수정
3. 일시정지 화면 수정
4. 게임 오버 화면 수정
5. 게임 경과 시간을 초 단위로 표시
6. 7개의 블록이 각각 고유의 색을 갖도록 수정

각 함수들의 역할과 호출

main:

디스플레이를 설정하고 게임 화면을 생성한다.

타이틀을 띄우고 게임 루프를 실행하여 게임을 진행한다. 게임 루프는 runGame 함수를 호출한다.

게임 루프가 끝나면 프로그램을 종료한다.

runGame:

게임에 필요한 변수들을 초기화한다.

무한 루프를 실행하고, 루프 안에서는 사용자의 키보드 입력 처리, 블록 이동, 라인 삭제 등이 이루어진다. 테트리스 게임의 핵심 로직을 구현한다.

checkForQuit 함수를 호출하고, 종료 조건을 만족하면 게임을 종료한다.

checkForQuit: 게임 창 닫기, ESC키를 감지하고 게임을 종료하는 함수 'terminate'를 호출한다.

runGame 함수 안에서 블록을 생성하기 위해 getNewPiece 함수가 호출된다.

getNewPiece: 블록의 모양, 회전, 색상이 랜덤하게 설정된 새 블록을 생성한다.

getNewPiece 함수에서 생성된 블록은 addToBoard 함수가 게임 보드에 추가한다.

addToBoard: 현재 떨어지는 블록의 모양과 위치를 반영해 화면에 추가한다.

생성된 블록들이 보드에 추가되고 쌓여서 줄이 완성되면 removeCompleteLines 함수가 그 줄을 제거하는 역할을 한다.

removeCompleteLines: 줄이 완성되었는지 확인하는 'isCompleteLine' 함수로

완성된 줄을 찾으면 그 줄을 지우고 위의 블록들을 한 칸씩 아래로 내린다. 제거된 줄의 수를 반환하고, 그 수는 점수에 추가된다.

점수가 높아지면 게임 레벨과 블록이 내려오는 속도가 높아진다. runGame 함수에서 calculateLevelAndFallFreq 함수를 호출하고, level과 fallFreq 값을 받는다.

calculateLevelAndFallFreq: 현재 스코어로 레벨을 측정하고 레벨과 속도를 반환한다.

블록을 움직이라는 키 값을 받았을 때는 'isValidPosition' 함수를 호출하여 블록이 유효한 위치에 있는지 확인한다.
'drawBoard' 함수는 게임 보드의 경계선을 그리고 drawBox 함수를 호출해 각각의 게임 보드 상자를 그린다.

drawBox: 색과 위치를 매개변수로 받아서 특정 위치에 상자를 그린다.

'drawStatus' 함수는 runGame 안에서 변경된 점수, 레벨 등의 게임 상태를 화면에 그린다.
'drawNextPiece'는 다음에 나올 블록은 화면에 나타낸다. 블록을 그릴 때는 drawPiece 함수를 호출한다.

drawPiece: 블록의 정보를 받아서 drawBox 함수로 블록을 그린다.

ShowTextScreen:
main함수와 runGame함수 모두에게서 호출된다.
텍스트를 화면에 출력하고 'checkForKeyPress' 함수를 통해 키가 눌렸는지 확인될 때까지 기다린다.

과제 수행

0. 오류 수정

처음 코드를 그냥 실행하면 오류가 생긴다. 이는 음악 재생 과정에서 tetrisb.mid와 tetrisc.mid 중 하나를 재생해야 하는데, 파일에는 아무 음악도 저장되어 있지 않아서 생긴 오류다. 따라서 이미 저장되어 있는 'Hover.mp3', 'Our_Lives_Past.mp3', 'Platform_9.mp3' 중 1개가 재생되도록 하면 오류는 발생하지 않는다.

1. 배경음악을 주어진 3개의 음악 중 1개가 재생되도록 수정

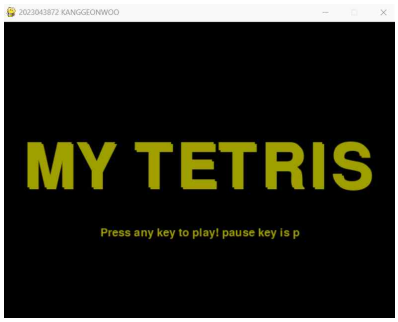
```
while True:
        playmusic = random.randint(0, 2)
    if playmusic == 0:
        pygame.mixer.music.load('Hover.mp3')
    elif playmusic == 1:
        pygame.mixer.music.load('Our_Lives_Past.mp3')
    else:
        pygame.mixer.music.load('Platform_9.mp3')
```

```
pygame.mixer.music.play(-1, 0.0)
```

0과 1 중에서 결정되었던 기존 코드에서 2까지 포함한 무작위 숫자를 생성하도록 수정하고, 재생되는 음악의 파일명을 바르게 적는다. 오류는 바로 고쳐졌고, 음악 재생도 잘 된다.

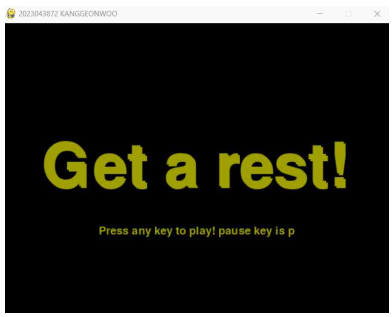
2. 시작 화면 수정

먼저 main 함수에서 창의 이름을 설정하는 pygame.display.set_caption('Tetromino')를 학번 이름으로 바꿔준다. 바로 아래에 showTextScreen 함수에 주는 문자열을 'MY TETRIS'로 바꾸면 타이틀 텍스트가 바뀐다. 밑에 부제목처럼 쓰여 있던 'Press a key to play.'는 시작하거나 일시 정지 했을 때, 게임이 종료됐을 때처럼 showTextScreen 함수를 호출했을 때 항상 있어야 하는 안냇말이다. 따라서 showTextScreen 함수에서 바꿔줄 수 있다. 과제 목표에서 모든 글자색이 노란색이기 때문에 전체 TEXTCOLOR와 TEXTSHADOWCOLOR를 모두 YELLOW로 바꾸면 시작화면이 완성된다.



3. 일시정지 화면 수정

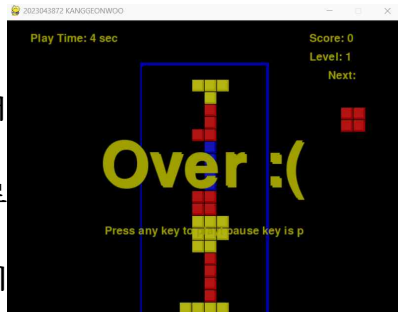
나머지는 시작 화면 수정 과정에서 완료했고, 'Paused'를 'Get a rest!'로 바꾸기만 하면 된다. pause는 테트리스 게임 도중 발생하는 이벤트이다. runGame에서 showTextScreen('Paused')를 showTextScreen('Get a rest!')로 바꿔준다.



4. 게임 오버 화면 수정

마찬가지로 'Game Over'를 'Over :('로 바꾸기만 하면 된다. main 함수의 runGame 함수가 종료됐을 때 showTextScreen의 인자를 바꿔준다.

5, 게임 경과 시간을 초 단위로 표시



runGame 함수의 맨 앞에 startTime = time.time()을 써서 게임 시작 시간을 기록한다.
게임 루프 내부에서 PlayTime = time.time() - startTime로 플레이 타임을 계산하고,
화면에 플레이 타임을 띄우기 위해 drawStatus 함수에 PlayTime 인자를 추가하여 호출한다.

```
def drawStatus(score, level, PlayTime):
    # 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)
    PlayTimeSurf = BASICFONT.render('Play Time: %d sec' % PlayTime, True, TEXTCOLOR)
    PlayTimeRect = levelSurf.get_rect()
    PlayTimeRect.topleft = (40, 20)
    DISPLAYSURF.blit(PlayTimeSurf, PlayTimeRect)
```

score, level과 똑같은 형식으로 PlayTime의 코드를 써주고, 출력되는 문자열 부분만 과제 목표에 맞게 수정한다. PlayTime은 원래 실수이지만 목표는 초 단위로 표시하는 것이므로 %d로 포맷한다. 이제 경과 시간이 초 단위로 표시되고 게임 오버된 후 다시 실행할 때 다시 runGame이 작동해서 0초로 초기화는 되지만 일시정지 하는 동안에도 시간이 흐른다.
이를 해결하기 위해 먼저 runGame 함수에서 pausedTime = 0으로 일시정지한 시간을 초기화한다. 플레이 타임을 계산하기 전에

```
if pausedTime != 0:
        startTime += time.time() - pausedTime
        pausedTime = 0
```

위의 코드를 쓰면 일시정지 했던 시간만큼 최종 결과값에 보정이 된다.

6. 7개의 블록이 각각 고유의 색을 갖도록 수정

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

각 모양마다 고유의 색을 지정해주는 딕셔너리를 만든다.
이미 COLORS라는 이름이 위에서 사용됐지만 딱히 다른 이름이 생각나는 게 아니라

‘COLORS_’ 라고 이름을 붙였다.
처음에는 LIGHTGREEN, LIGHTBLUE, LIGHTRED를 썼었는데, GREEN, BLUE, RED와 구분이 안 될 정도여서 WHITE, GRAY, PINK로 지정해줬다.
분홍색은 기존에 정의가 되어있지 않았던 색이라 PINK = (175, 0, 175) 코드를 추가했다.
이제 지정된 색을 블록을 그리는 함수에 주면 된다. ‘getNewPiece’ 함수에서 color의 값을 랜덤이 아니라 shape에 맞는 색을 반환하도록 수정한다. 'color': COLORS[shape]
화면에 실제로 그리는 함수 ‘drawBox’에서는 정수값 대신 색이 인자로 들어왔으므로 COLORS[color]와 LIGHTCOLORS[color]를 둘 다 color로 바꿔준다.

결과

0-1. 처음에 났던 오류는 음악 파일을 찾을 수 없는 것이었고, 배경음악을 주어진 3개의 음악 중 1개가 재생되도록 수정한 후 제대로 작동이 되었다.
2-4. 시작 화면, 일시정지 화면, 게임 오버 화면을 목표와 똑같이 그리는 데 성공했다.
5. 게임이 시작될 때 시간과 게임을 진행한 시간, 그리고 일시정지를 지속한 시간을 계산해서 게임 경과 시간을 정확히 표시할 수 있었다.
6. 7개의 블록들에 미리 색을 지정해주고 각각 고유의 색을 가지게 되었다.