2021011394 김예준

오픈소스SW 12주차 과제중심수업

과제 깃허브 링크: https://github.com/dpwns5889/osw

1) 1~6번의 과제의 코드 수정 부분

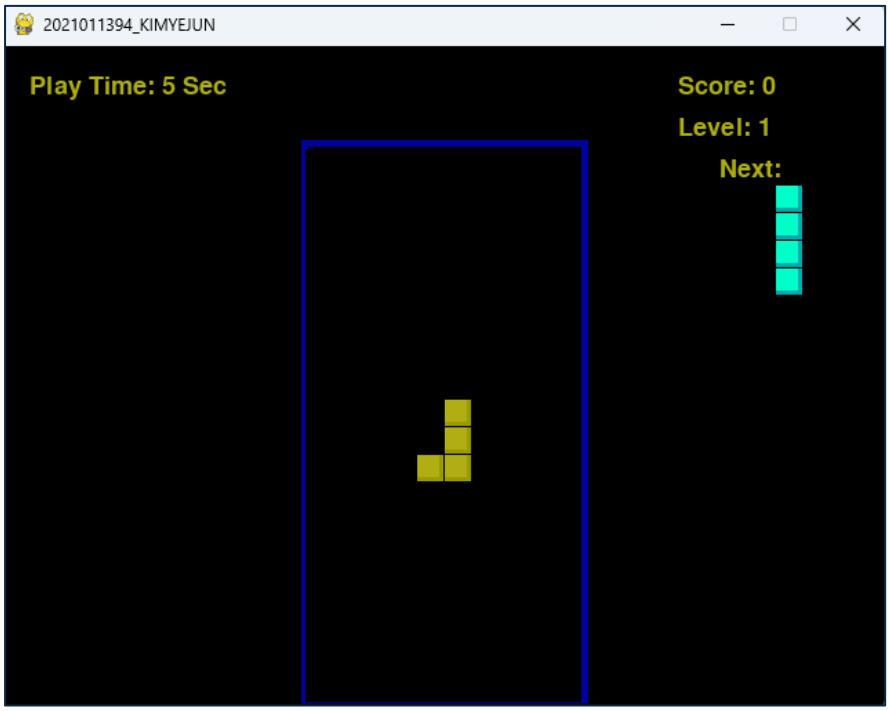
1. 현재 테트리스 게임의 배경음악을 주어진 3개의 음악 중 1개가 재생되도록 수정하기

```
while True: # game loop
    music_count = random.randint(0, 3) #랜덤 정수(0,1,2)를 담는 변수를 선언해서 값에 따라 노래를 재생.
    if music_count == 0:
        pygame.mixer.music.load('Hover.mp3')
    elif music_count == 1:
        pygame.mixer.music.load('Our_Lives_Past.mp3')
    else:
        pygame.mixer.music.load('Platform_9.mp3')
    pygame.mixer.music.play(-1, 0.0) #무한 반복
    runGame()
    pygame.mixer.music.stop()
    showTextScreen('Over :(')
```

main()함수에서 random.randint()를 (0, 2)에서 (0, 3)으로 바꿔서 결과값이 #0, 1, 2 3개로 나올 수 있게 했습니다. 이후 if문을 이용해 각 값에 해당하는 노래를 load()하여 랜덤으로 3가지 노래가 재생되도록 만들었습니다.

2. 상태창 이름을 학번_이름으로 수정하기

```
pygame.display.set_caption('2021011394_KIMYEJUN') # Tetromino -> 학번, 이름
```



main()함수에서 set_caption()의 인자를 기본 'Tetromino'에서 '학번_이름'으로 수정해 상태창에 표시되는 문구를 변경했습니다.

3. 게임시작화면의 문구를 MY TETRIS으로 변경하기

```
showTextScreen('My Tetris') # Tetromino -> My Tetris
```

main() 함수에서 showTextScreen()의 인자를 기본의 'Tetromino'에서 'My_Tetris'로 수정해 시작화면에 표시되는 문구를 수정했습니다.

4. 게임시작화면의 문구 및 배경색을 노란색으로 변경하기

```
TEXTCOLOR = LIGHTYELLOW #White -> Yellow
TEXTSHADOWCOLOR = YELLOW #Gray -> Yellow
```

TEXTCOLOR와 TEXTSHADOWCOLOR 각각 WHITE와 GRAY에서 LIGHTYELLOW와 YELLOW로 변경해서 showTextScreen()과 drawStatus(), drawNextPiece()에 사용되는 문구색과 문구의 배경색을 노란색으로 변경했습니다.

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

```
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
    start_ticks = pygame.time.get_ticks()
```

```
# drawing everything on the screen
DISPLAYSURF.fill(BGCOLOR)
drawBoard(board)
play_time = (pygame.time.get_ticks() - start_ticks) / 1000
drawStatus(score, level, play_time)
```

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

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

runGame의 루프가 시작하기 전에 start_ticks를 time.get_ticks()를 이용해 시작한 이후부터의 밀리초 값과, 루프가 끝날 때의 밀리초 값을 빼 그 값을 1000으로 나누어 시간인 play_time 값으로 사용했습니다.

이 play_time 변수를 점수와 레벨을 표시하는 drawStatus()함수에 매개변수로 전달해 시간을 업데이트 할 수 있도록 했습니다. 추가로 시간이 표시될 문구를 그린 다음에 (20, 20)에 위치하도록 했습니다.

6. 7개의 블록이 각각 고유의 색을 갖도록 코드를 수정하거나 추가하기

```
#               R    G    B
WHITE       = (255, 255, 255)
GRAY        = (185, 185, 185)
BLACK       = (  0,   0,   0)
RED         = (155,   0,   0)
LIGHTRED    = (175,  20,  20)
ORANGE      = (255, 170,   0)
LIGHTORANGE = (255, 190,  50)
YELLOW      = (155, 155,   0)
LIGHTYELLOW = (175, 175,  20)
GREEN       = (  0, 155,   0)
LIGHTGREEN  = ( 20, 175,  20)
CYAN        = (  0, 170, 175)
LIGHTCYAN   = (  0, 255, 200)
BLUE        = (  0,   0, 155)
LIGHTBLUE   = ( 20,  20, 175)
PURPLE      = (200,   0, 255)
LIGHTPURPLE = (220, 110, 255)

BORDERCOLOR = BLUE
BGCOLOR = BLACK
TEXTCOLOR = LIGHTYELLOW #White -> Yellow
TEXTSHADOWCOLOR = YELLOW #Gray -> Yellow
COLORS      = (RED     , ORANGE     , YELLOW     , GREEN     , CYAN     , BLUE     ,      PURPLE)
LIGHTCOLORS = (LIGHTRED, LIGHTORANGE, LIGHTYELLOW, LIGHTGREEN, LIGHTCYAN, LIGHTBLUE, LIGHTPURPLE)
```

블록에 사용될 색을 4가지에서 7가지로 늘리고 그 색들을 COLORS와 LIGHTCOLORS에 순서가 맞게 초기화 했습니다.

```
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}

COLOR = {'S': 0,
         'Z': 1,
         'J': 2,
         'L': 3,
         'I': 4,
         'O': 5,
         'T': 6}
```

```
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': COLOR[shape]}
    return newPiece
```

그리고 각 블록마다 정수 값을 가지게 했습니다. 이 값은 새 블록을 생성하는 getNewPiece()함수에서 블록을 각 키 값에 맞게 생성하는 데 사용합니다.

2) 각 함수의 역할

main(): 전반부는 게임을 실행하는 데 필요한 일부 변수를 선언한다. 폰트나 타이틀, 초기화면도 초기화하고 생성한다. 이후 while 루프 안에서는 노래를 재생하고 게임이 오버되어도 다시 재생할 수 있게 한다.

runGame(): 함수의 전반부는 각각의 게임을 초기화한다. 시간, 레벨, 블록들을 초기화한다. 루프 안에서는 블록이 없으면 생성한다. 이후 입력을 감지해서 블록의 이동, 드랍, 회전을 감지하고 실행한다. 이후 모든 입력의 결과를 화면에 반영한다.

makeTextObjs(text, font, color): 매개변수로 받은 text를 color에 맞게 색을 변경하고 입력받은 font로 폰트를 변경해 오브젝트로 만들어 반환한다.

terminate(): 게임을 종료한다.

checkForKeyPress(): 종료하는 입력이 있는지 검사하고 버튼이 올라오는 이벤트를 감지해 그 버튼을 반환한다.

showTextScreen(text): 화면에 크게 text를 표시하고 아무 버튼의 KEYUP 이벤트를 감지되면 종료된다.

checkForQuit(): 종료하는 입력과 ESC 버튼의 KETUP이벤트를 감지한다. 감지하면 terminate() 함수를 실행해 게임을 종료한다.

calculateLevelAndFallFreq(score): score를 바탕으로 레벨을 조정하고 레벨에 따라 블록이 내려오는 속도를 변경한다.

getNewPiece(): 블록의 모양을 생성하고 그 블록의 정보를 반환한다.

addToBoard(board, piece): board에 piece의 정보를 저장한다.

getBlankBoard(): 쌓은 블록을 저장하는 리스트인 Board를 만들고 반환한다.

isOnBoard(x, y): x, y가 유효한 위치 값인지 검사한다.

isValidPosition(board, piece, adjX=0, adjY=0): isOnBoard()를 이용해 블록이 board 안에 있는지 검사한다. 또 블록이 다른 블록과 충돌하는지 여부도 검사한다. 보드에 있고 충돌하지 않으면 True를 반환한다.

isCompleteLine(board, y): board의 y줄이 블록으로 다 차 있는지 검사한다.

removeCompleteLines(board): board의 모든 y줄을 검사해 완성된 줄이 있다면 삭제하고 그 위의 블록을 아래로 내린다. 이후 삭제된 줄만큼 점수를 증가시킨다.

convertToPixelCoords(boxx, boxy): board의 좌표에 맞춰 화면의 좌표를 반환한다.

drawBox(boxx, boxy, color, pixelx=None, pixely=None): pixelx, pixely에 맞춰 박스를 시각적으로 그린다.

drawBoard(board): 테두리를 먼저 그린 후에 각각의 블록을 drawBox()를 이용해 그린다.

drawStatus(score, level, play_time): 화면에 점수, 레벨, 시간을 표시한다.

drawPiece(piece, pixelx=None, pixely=None): piece를 구성하는 박스를 pixelx, pixely에 맞게 그린다.

drawNextPiece (piece): 다음에 내려올 블록을 표시한다.

3) 함수의 호출 순서와 호출 조건

main(): 모든 함수의 정의가 끝나면 실행된다.

showTextScreen(): 게임이 실행됐을 때, 중지했을 때, 끝났을 때 실행된다.

makeTextObjs()

checkForKeyPress(): 키입력이 있는지 검사하기 위해 실행된다.

checkForQuit()

terminate()

runGame(): 게임이 종료되었을 때 버튼이 입력되면 실행된다.

runGame(): 게임이 실행되었을 때 최초 1회 이후에는 게임이 종료되었을 때 버튼이 입력되면 실행된다.

getBlankBoard(): 게임이 시작될 때 보드를 초기화하기 위해 1회만 실행한다.

calculateLevelAndFallFreq(): 게임이 시작될 때 초기화를 위해 1회 실행하고 이후 블록이 착지하면 실행된다.

- **게임 동안 아래 함수들이 반복해서 실행된다.**

getNewPiece(): 게임이 시작될 때 2회, 블록이 착지한 후에 실행된다.

isValidPosition(): 블록이 이동, 회전, 낙하, 착륙할 때 실행된다.

isOnBoard()

checkForQuit(): rungame()과 checkForKeyPress()에서 실행된다.

terminate(): 실행종료 입력이나 ESC 입력이 있을 때 실행된다.

addToBoard(): 블록이 착륙했을 때 실행된다.

removeCompleteLines(): 블록이 착륙한 뒤에 실행된다.

isCompleteLine(): 완성된 줄을 탐색할 때 실행된다.

drawBoard(): 블록의 이동이 다 끝난 후에 실행된다.

drawBox()

convertToPixelCoords(): 입력받은 pixelx와 pixely가 없다면 실행된다.

drawStatus(): 블록의 이동이 다 끝난 후에 실행된다.

drawNextPiece(): 블록의 이동이 다 끝난 후에 실행된다

drawPiece()

drawBox(): piece의 해당 칸이 blank가 아니라면 실행된다.

convertToPixelCoords(): 입력받은 pixelx와 pixely가 없다면 실행된다.