

오픈소스\_SW 12주차  
과제

ICT 융합학부 2021024566 전재연

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

수정 A.

```
def main():  
    global FPSCLOCK, DISPLAYSURF, BASICFONT, BIGFONT, startTime  
    pygame.init()  
    FPSCLOCK = pygame.time.Clock()  
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))  
    BASICFONT = pygame.font.Font(pygame.font.get_default_font(), 18)  
    BIGFONT = pygame.font.Font(pygame.font.get_default_font(), 100)  
    pygame.display.set_caption('2021024566_JEONJAEYEON')  
  
    showTextScreen('MY TETRIS')  
    while True: # game loop  
        startTime = time.time() # 게임 시작할 때 startTime 초기화  
        if random.randint(0, 1) == 0:  
            pygame.mixer.music.load('Hover.mp3')  
        else:  
            pygame.mixer.music.load('Platform_9.mp3')  
            pygame.mixer.music.play(-1, 0.0)  
        runGame()  
        pygame.mixer.music.stop()  
        showTextScreen('Game Over')
```

main 메서드가 실행되면 random 함수가 실행되어 0 또는 1을 반환하게 됩니다.

그 수를 기준으로 if문이 실행되어 0이 출력되면 Hover.mp3가,  
1이 출력되면 Platform\_9.mp3가 실행됩니다.

2. 상태창 이름을 학번\_이름 으로 수정

1번의 메인메서드 코드 사진을 참고하겠습니다.

상태창을 설정하는 코드인,

pygame.display.set\_caption()에 저의 학번\_이름을 입력하여 상태창에 제 학번과 이름이 출

력 될 수 있도록 수정하였습니다

### 3. 게임시작화면의 문구를 MY TETRIS으로

변경

#### A. 1번의 메인메서드 코드 사진을

참고하겠습니다. 첫 화면을 출력하는

showTextScreen() 에 MY\_TETRIS를 입력하여 게임시작화면에 해당 문구가 출력되도록 수정 하였습니다.

### 4. 게임시작화면의 문구 및 배경색을 노란색으로 변경

```
#           R   G   B
WHITE      = (255, 255, 255)
GRAY       = (185, 185, 185)
BLACK      = (  0,   0,   0)
RED        = (155,   0,   0)
LIGHTRED   = (175,  20,  20)
GREEN      = (  0, 155,   0)
LIGHTGREEN = ( 20, 175,  20)
BLUE       = (  0,   0, 155)
LIGHTBLUE  = ( 20,  20, 175)
YELLOW     = (155, 155,   0)
LIGHTYELLOW = (175, 175,  20)

BORDERCOLOR = BLUE
BGCOLOR = BLACK
TEXTCOLOR = YELLOW
TEXTSHADOWCOLOR = YELLOW
COLORS = ( BLUE, GREEN, RED, YELLOW)
LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYELLOW)
assert len(COLORS) == len(LIGHTCOLORS) # each color must have light color
```

글자 색깔과 글자의 배경을 설정하는

TEXTCOLOR = YELLOW

TEXTSHADOWCOLOR = YELLOW를 모두 YELLOW로 수정하였습니다

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

```
while True: # game loop
    startTime = time.time() # 게임 시작할 때 startTime 초기화
    if random.randint(0, 1) == 0:
        pygame.mixer.music.load('Hover.mp3')
    else:
        pygame.mixer.music.load('Platform_9.mp3')
    pygame.mixer.music.play(-1, 0.0)
    runGame()
    pygame.mixer.music.stop()
    showTextScreen('Game Over')

def runGame():
    global startTime
    board = getBlankBoard()
    lastMoveDownTime = time.time()
    lastMoveSidewaysTime = time.time()
    lastFallTime = time.time()
    movingDown = False
    movingLeft = False
    movingRight = False
    score = 0
    level, fallFreq = calculateLevelAndFallFreq(score)

    fallingPiece = getNewPiece()
    nextPiece = getNewPiece()

    while True: # game loop
        playTime = int(time.time() - startTime) # playTime을 초 단위로 계산
        if fallingPiece == None:
            fallingPiece = nextPiece
            nextPiece = getNewPiece()
            lastFallTime = time.time()
            if not isValidPosition(board, fallingPiece):
                .

    DISPLAYSURF.fill(BGCOLOR)
    drawBoard(board)
    drawStatus(score, level, playTime) # playTime을 drawStatus에 전달
    drawNextPiece(nextPiece)
    if fallingPiece != None:
        drawPiece(fallingPiece)
```

```
def drawStatus(score, level, playTime):
    scoreSurf = BASICFONT.render('Score: %s' % score, True, TEXTCOLOR)
    scoreRect = scoreSurf.get_rect()
    scoreRect.topleft = (WINDOWWIDTH - 150, 20)
    DISPLAYSURF.blit(scoreSurf, scoreRect)

    levelSurf = BASICFONT.render('Level: %s' % level, True, TEXTCOLOR)
    levelRect = levelSurf.get_rect()
    levelRect.topleft = (WINDOWWIDTH - 150, 50)
    DISPLAYSURF.blit(levelSurf, levelRect)

    playTimeSurf = BASICFONT.render('PlayTime: %s sec' % playTime, True, TEXTCOLOR)
    playTimeRect = playTimeSurf.get_rect()
    playTimeRect.topright = (WINDOWWIDTH - 490, 20)
    DISPLAYSURF.blit(playTimeSurf, playTimeRect)
```

main 코드에 startTime을 현재시간-현재시간=0을 이용해 0으로 초기화 합니다

후에 runGame 코드에서 time-startTime으로 현재 플레이되는 시간을 sec단위로  
측정하고 이를 int타입으로 설정하여 소수점을 지웁니다  
그렇게 측정한 값을 drawStatus에 반환 할 수 있도록 하고

drawStatus에서

화면 좌측에 위치 시킬 수 있도록 좌표를 계산하여 플레이 타임을 표시합니다

플레이가 종료되고 다시 새로운 게임이 시작되면 다시 메인함수가 호출되므로  
playTime은 0 으로 다시 초기화 됩니다

6. 7개의 블록이 각각 고유의 색을 갖도록 코드를 수정하거나 추가

```
BLOCKCOLORS = {
    'S': 0,
    'Z': 1,
    'J': 2,
    'L': 3,
    'I': 0,
    'O': 1,
    'T': 2
}

def getNewPiece():
    # return a new piece with a fixed color based on its shape
    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': BLOCKCOLORS[shape]}
    return newPiece
```

BLOCKCOLORS라는 디렉토리를 생성하여 getNewPiece의 color 필드에서 블록의 색깔을 고정시켜 초기화 합니다.

```
def drawBox(boxx, boxy, color, pixelx=None, pixely=None):
    if color == BLANK:
        return
    if pixelx == None and pixely == None:
        pixelx, pixely = convertToPixelCoords(boxx, boxy)
    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))
```

그 후 runGame()이 실행되면 drawPiece가 실행되고 이에따라 drawBox가 실행되며 getNewPiece에서 생성했던 블록이 화면에 나타나게됩니다.

7. 각함수의 고유한 역할과 함수의 호출순서 및 호출조건

주요 함수 고유한 역할에 대하여 서술하겠습니다.

main(): 이 함수는 게임의 진입점입니다. Pygame을

초기화하고 게임 루프를 시작합니다. 게임 루프에서는

게임을 실행하고, 게임이 종료되면 "Game Over"

화면을 표시합니다.

runGame():

runGame() 함수는 게임의 주요 로직을 담당합니다. 이 함수는 게임의 메인 루프를 실행하며, 다음과 같은 일을 수행합니다:

게임 초기화: 보드를 초기화하고, 점수를 초기화하며, 첫 번째 Tetromino 블록과 그 다음 Tetromino 블록을 설정합니다.

이벤트 처리: 사용자 입력을 처리하여 Tetromino 블록을 이동하고 회전시키는 등의 작업을 수행합니다.

블록 이동 및 시간 관리: Tetromino 블록이 아래로 이동하고, 시간이 흐름에 따라 게임 속도를 조절합니다.

화면 업데이트: 게임 보드, 점수, 레벨 등을 업데이트하고, 화면에 그려줍니다.

게임 종료 조건 확인: Tetromino 블록이 화면 위쪽으로 나가면 게임을 종료하고, "Game Over" 화면을 표시합니다.

makeTextObjs(text, font, color): 이 함수는 주어진

텍스트, 폰트 및 색상으로 텍스트 오브젝트를

생성합니다.

terminate(): 이 함수는 게임을 종료합니다.

Pygame을 종료하고 시스템을 종료합니다.

checkForKeyPress(): 이 함수는 사용자 입력을

대기합니다. 사용자가 키를 누를 때까지 대기하고, 키

이벤트를 반환합니다.

showTextScreen(text): 이 함수는 대기 화면을

표시합니다. 화면 가운데에 큰 텍스트를 표시하고,

사용자 입력을 대기합니다.



checkForQuit(): 이 함수는 게임 종료를 확인합니다.

윈도우를 닫거나 ESC 키를 누를 때 게임을

종료합니다.

calculateLevelAndFallFreq(score): 이 함수는 현재

점수를 기반으로 플레이어의 레벨과 블록이 떨어지는

속도를 계산합니다.

getNewPiece():

getNewPiece() 함수는 새로운 Tetromino 블록을 생성합니다. 이 함수는 다음과 같은 일을 수행합니다:

Tetromino 블록의 모양을 무작위로 선택합니다. 색깔은 위의 BLOCKCOLORS에서 지정한 색 깔로 고정합니다.

Tetromino 블록의 초기 위치를 설정합니다. 일반적으로 화면 상단 중앙에 위치하도록 설정됩니다.

생성된 Tetromino 블록을 NewPiece로 반환하면 runGame에서 이를 받아

```
fallingPiece = getNewPiece()
```

```
nextPiece = getNewPiece()
```

이렇게 떨어질 블록과 다음블록을 생성합니다.

addToBoard(board, piece): 이 함수는 떨어지는

조각을 보드에 추가합니다.

getBlankBoard(): 이 함수는 빈 보드를 생성하여

반환합니다.

isOnBoard(x, y): 이 함수는 주어진 좌표가 보드

내에 있는지 확인합니다.

isValidPosition(board, piece, adjX=0, adjY=0): 이

함수는 조각의 위치가 유효한지 확인합니다.

`isCompleteLine(board, y)`: 이 함수는 주어진 `y`

위치의 줄이 완전한 줄인지 확인합니다.

`removeCompleteLines(board)`: 이 함수는 완전한

줄을 제거하고, 위에 있는 블록들을 아래로

이동시킵니다.

`convertToPixelCoords(boxx, boxy)`: 이 함수는

보드의 좌표를 화면 상의 좌표로 변환합니다.

`drawBox(boxx, boxy, color, pixelx=None,`

`pixely=None)`: 이 함수는 박스를 그립니다. 박스의

위치와 색상이 주어지면 해당 위치에 박스를

그립니다.

`drawBoard(board)`: 이 함수는 전체 보드를 그립니다.

`drawStatus(score, level, playTime)`:

`drawStatus()` 함수는 각각의 상태를 화면에 렌더링하는 역할을 합니다.



점수를 나타내는 Score는 runGame()에서  
score += removeCompleteLines(board) 이 로직을 통해 score를 반환받아 화면에  
렌더링 합니다. 이는 한 줄이 완성되었을 때 1씩 올라갑니다

게임의 난이도를 나타내는 level은 runGame()에서  
level, fallFreq = calculateLevelAndFallFreq(score) 로직을 통해 초기화되고 drawStatus는  
이를 받아 렌더링 합니다  
level이 올라갈수록 블록이 떨어지는 속도가 증가합니다

플레이타임을 나타내는 PlayTime은 runGame에서 플레이타임이  
초기화되어 drawStatus에서 렌더링합니다

drawPiece(piece, pixelx=None, pixely=None): 이

함수는 떨어지는 조각을 그립니다.

drawNextPiece(piece): 이 함수는 다음에 떨어질

조각을 표시합니다.

이를 바탕으로 게임시작을 기준으로 함수의 호출순서 및 호출조건을 서술하겠습니다

main() 함수 호출:

프로그램이 시작되면 main() 함수가 호출됩니다.

main() 함수는 게임을 초기화하고 runGame() 함수를 호출하여 게임을 실행합니다.

게임이 종료되면 "Game Over" 화면이 표시됩니다.

runGame() 함수 내부 호출:

runGame() 함수 내에서는 다음과 같은 함수들이 호출됩니다:

getBlankBoard(): 보드를 초기화합니다.

calculateLevelAndFallFreq(score): 현재 점수를 기반으로 레벨과 Tetromino 블록의 떨어지는 속도를 계산합니다.

playTime = int(time.time() - startTime): 해당로직으로 플레이타임을 계산하여 반환합니다.

getNewPiece(): 현재 Tetromino 블록과 그 다음 Tetromino 블록을 설정합니다.

checkForQuit(): 사용자가 게임을 종료하려고 할 때 확인합니다.

이벤트 처리 및 게임 루프:

runGame()을 통해서 이벤트 처리를 위한 루프가 실행됩니다.

사용자의 입력에 따라 Tetromino 블록이 이동하거나 회전합니다.

게임 루프가 반복되면서 Tetromino 블록이 아래로 이동하고 화면이 업데이트됩니다.

Tetromino 블록 이동 및 화면 업데이트:

Tetromino 블록은 사용자의 입력 또는 일정한 시간 간격으로 자동으로 아래로 이동합니다.

Tetromino 블록의 이동, 회전 및 화면 업데이트를 처리하는 함수들이 호출됩니다.

게임 종료 조건 확인:

Tetromino 블록이 화면 위쪽으로 나가거나 보드가 가득 찼을 때 게임을 종료합니다.  
main()에 의하여 "Game Over" 화면을 표시하고, 사용자의 입력을 기다립니다.

깃허브주소

<https://github.com/gdbs1107/osw.git>