

OSW Pygame 과제 보고서

ICT 융합학부 미디어테크놀로지전공 2023041921 정새움

GitHub Repository 링크: <https://github.com/aeioiie/osw/blob/main/README.md>

체크리스트 1~6 수행 방법

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

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

올려주신 음악 파일을 다운받아 pygame.mixer.music.load()에 파일 이름을 작성하였다.

원래는 random.randint(0, 1) 이었지만 음악 파일이 3개였기에 (0, 2)로 수정하였다.

2. 상태창 이름을 학번_이름 으로 수정, 게임 시작 화면의 문구를 MY TETRIS으로 변경

```
1 def main():
2     global FPSCLOCK, DISPLAYSURF, BASICFONT, BIGFONT
3     pygame.init()
4     FPSCLOCK = pygame.time.Clock()
5     DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
6     BASICFONT = pygame.font.Font('freesansbold.ttf', 18)
7     BIGFONT = pygame.font.Font('freesansbold.ttf', 100)
8     pygame.display.set_caption('2023041921_정새움')
9
10    showTextScreen('MY TETRIS')
```

main 함수에서 상태창의 이름과 게임 시작 화면의 문구를 나타내는 부분을 수정하였다.

3. 일시정지 화면, 게임 오버 화면

```
1 def main():
2     global FPSLOCK, DISPLAYSURF, BASICFONT, BIGFONT
3     pygame.init()
4     FPSLOCK = pygame.time.Clock()
5     DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
6     BASICFONT = pygame.font.Font('freesansbold.ttf', 18)
7     BIGFONT = pygame.font.Font('freesansbold.ttf', 100)
8     pygame.display.set_caption('2023041921_정새움')
9
10    showTextScreen('MY TETRIS')
11    while True: # game Loop
12        if random.randint(0, 2) == 0:
13            pygame.mixer.music.load('Hover.mp3')
14        elif random.randint(0, 2) == 1:
15            pygame.mixer.music.load('Platform_9.mp3')
16        else:
17            pygame.mixer.music.load('Our_Lives_Past.mp3')
18        pygame.mixer.music.play(-1, 0.0)
19        runGame()
20        pygame.mixer.music.stop()
21        showTextScreen('Over :(')
```

```
1 if (event.key == K_p):
2     # Pausing the game
3     DISPLAYSURF.fill(BG_COLOR)
4     pygame.mixer.music.stop()
5     showTextScreen('Get a rest!') # pause until a key press
6     pygame.mixer.music.play(-1, 0.0)
7     lastFallTime = time.time()
8     lastMoveDownTime = time.time()
9     lastMoveSidewaysTime = time.time()
```

```
1 pressKeySurf, pressKeyRect = makeTextObjs('Press any key to play! pause key is p', BASICFONT, TEXTCOLOR)
2 pressKeyRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2) + 100)
3 DISPLAYSURF.blit(pressKeySurf, pressKeyRect)
```

일시정지 화면에 'Get a rest!', 게임 오버 화면에 'Over :(', 그리고 첫 화면, 일시정지 화면, 게임 오버 화면 모두에 공통적으로 'Press any key to play! pause key is p'를 나타내기 위해 문구를 수정하였다.

4. 게임 시작 화면의 문구 및 배경색을 노란색으로 변경

```
1 BORDERCOLOR = BLUE
2 BGCOLOR = BLACK
3 TEXTCOLOR = YELLOW
4 TEXTSHADOWCOLOR = YELLOW
5 COLORS = (BLUE, GREEN, RED, YELLOW)
6 LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYELLOW)
```

게임 시작 화면의 문구와 배경 색을 모두 노란색으로 변경하기 위해 TEXTCOLOR와 TEXTSHADOWCOLOR을 모두 YELLOW로 설정하였다.

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

```
1 def runGame():
2     # setup variables for the start of the game
3     board = getBlankBoard()
4     lastMoveDownTime = time.time()
5     lastMoveSidewaysTime = time.time()
6     lastFallTime = time.time()
7     gameStartTime = time.time() # 게임 시작 시간 기록
8     movingDown = False # note: there is no movingUp variable
9     movingLeft = False
10    movingRight = False
11    score = 0
12    level, fallFreq = calculateLevelAndFallFreq(score)
13
14    fallingPiece = getNewPiece()
15    nextPiece = getNewPiece()
16
17    while True: # game loop
18        currentTime = time.time() # 현재 시간 업데이트
19        elapsedTime = int(currentTime - gameStartTime) # 경과 시간 계산
```



```
1 DISPLAYSURF.fill(BG_COLOR)
2     drawBoard(board)
3     drawStatus(score, level, elapsedTime)
4     drawNextPiece(nextPiece)
5     if fallingPiece != None:
6         drawPiece(fallingPiece)
```




```
1 def drawStatus(score, level, elapsedTime):
2     # draw the score text
3     scoreSurf = BASICFONT.render('Score: %s' % score, True, TEXT_COLOR)
4     scoreRect = scoreSurf.get_rect()
5     scoreRect.topleft = (WINDOWWIDTH - 150, 20)
6     DISPLAYSURF.blit(scoreSurf, scoreRect)
7
8     # draw the level text
9     levelSurf = BASICFONT.render('Level: %s' % level, True, TEXT_COLOR)
10    levelRect = levelSurf.get_rect()
11    levelRect.topleft = (WINDOWWIDTH - 150, 50)
12    DISPLAYSURF.blit(levelSurf, levelRect)
13
14    # 경과 시간을 화면에 표시
15    timeSurf = BASICFONT.render('Play Time: %s sec' % elapsedTime, True, TEXT_COLOR)
16    timeRect = timeSurf.get_rect()
17    timeRect.topright = (WINDOWWIDTH - 450, 20)
18    DISPLAYSURF.blit(timeSurf, timeRect)
```


먼저 runGame() 함수 내에서 게임 시작 시간 기록을 위해 game.Start.time이라는 변수를 time.time()으로 초기화한다. while문 안에 currentTime = time.time()을 추가하여 현재 시간을 업데이트 하는 변수를 새로 만들고, elapsedTime = int(currentTime - gameStartTime)을 추가하여 경과 시간을 계산한다. 또한 runGame()의 아래 부분에 drawStatus 함수에 elapsedTime 인자를 추가한다. drawStatus() 함수를 정의하는 부분에도 elapsedTime의 인자를 추가해야 한다.

화면에 표시되는 점수, 레벨 그리고 경과 시간이 모두 같은 디자인이므로 같은 함수 내에 같은 형태로 경과 시간을 나타내는 부분을 추가한다. 이 때 경과 시간을 나타내는 Play Time은 Score, Level과 다르게 왼쪽에 위치하므로 time.Rect.topright를 사용하며, x좌표의 적절한 위치를 찾아 추가한다. y좌표의 위치는 Score과 같은 위치로 하였다.


6. 7개의 블록이 각각 고유의 색을 갖도록 코드를 수정하거나 추가



```
1  #           R   G   B
2  WHITE      = (255, 255, 255)
3  GRAY       = (185, 185, 185)
4  BLACK      = (  0,   0,   0)
5  RED        = (155,   0,   0)
6  LIGHTRED   = (175,  20,  20)
7  GREEN      = (  0, 155,   0)
8  LIGHTGREEN = ( 20, 175,  20)
9  BLUE       = (  0,   0, 155)
10 LIGHTBLUE  = ( 20,  20, 175)
11 YELLOW     = (155, 155,   0)
12 LIGHTYELLOW = (175, 175,  20)
13 ORANGE     = (255, 128,   0)
14 PURPLE     = (127,   0, 255)
15 PINK       = (255,   0, 255)
16 LIGHTORANGE = (255, 164,  32)
17 LIGHTPURPLE = (153,  51, 255)
18 LIGHTPINK  = (255,  51, 255)
```



```
1  PIECES = {'S': {'shape': S_SHAPE_TEMPLATE, 'color': BLUE},
2           'Z': {'shape': Z_SHAPE_TEMPLATE, 'color': GREEN},
3           'J': {'shape': J_SHAPE_TEMPLATE, 'color': RED},
4           'L': {'shape': L_SHAPE_TEMPLATE, 'color': YELLOW},
5           'I': {'shape': I_SHAPE_TEMPLATE, 'color': ORANGE},
6           'O': {'shape': O_SHAPE_TEMPLATE, 'color': PURPLE},
7           'T': {'shape': T_SHAPE_TEMPLATE, 'color': PINK}}
```



```
1  COLORS      = (BLUE, GREEN, RED, YELLOW, ORANGE, PURPLE, PINK)
2  LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYELLOW, LIGHTORANGE, LIGHTPURPLE, LIGHTPINK)
3  assert len(COLORS) == len(LIGHTCOLORS) # each color must have light color
```

```

1  def getNewPiece():
2      # return a random new piece in a random rotation and color
3      shape = random.choice(list(PIECES.keys()))
4      newPiece = {'shape': shape,
5                  'rotation': random.randint(0, len(PIECES[shape]['shape']) - 1), # ['shape'] 추가
6                  'x': int(BOARDWIDTH / 2) - int(TEMPLATEWIDTH / 2),
7                  'y': -2, # start it above the board (i.e. less than 0)
8                  'color': COLORS.index(PIECES[shape]['color'])} # 고유 색상 인덱스 사용
9      return newPiece
10
11 def addToBoard(board, piece):
12     piece_shape = PIECES[piece['shape']]['shape'][piece['rotation']]
13     for x in range(TEMPLATEWIDTH):
14         for y in range(TEMPLATEHEIGHT):
15             if piece_shape[y][x] != BLANK:
16                 board[x + piece['x']][y + piece['y']] = piece['color']
17

```

```

1  def isValidPosition(board, piece, adjX=0, adjY=0):
2      piece_shape = PIECES[piece['shape']]['shape'][piece['rotation']]
3      for x in range(TEMPLATEWIDTH):
4          for y in range(TEMPLATEHEIGHT):
5              isAboveBoard = y + piece['y'] + adjY < 0
6              if isAboveBoard or piece_shape[y][x] == BLANK:
7                  continue
8              if not isOnBoard(x + piece['x'] + adjX, y + piece['y'] + adjY):
9                  return False
10             if board[x + piece['x'] + adjX][y + piece['y'] + adjY] != BLANK:
11                 return False
12     return True

```

```

1 def drawPiece(piece, pixelx=None, pixely=None):
2     shapeToDraw = PIECES[piece['shape']]['shape'][piece['rotation']]
3     if pixelx == None and pixely == None:
4         # if pixelx & pixely hasn't been specified, use the location stored in the piece data structure
5         pixelx, pixely = convertToPixelCoords(piece['x'], piece['y'])
6
7     # draw each of the boxes that make up the piece
8     for x in range(TEMPLATEWIDTH):
9         for y in range(TEMPLATEHEIGHT):
10             if shapeToDraw[y][x] != BLANK:
11                 drawBox(None, None, piece['color'], pixelx + (x * BOXSIZE), pixely + (y * BOXSIZE))
12
13
14 def drawNextPiece(nextPiece):
15     # draw the "next" text
16     nextSurf = BASICFONT.render('Next:', True, TEXTCOLOR)
17     nextRect = nextSurf.get_rect()
18     nextRect.topleft = (WINDOWWIDTH - 120, 80)
19     DISPLAYSURF.blit(nextSurf, nextRect)
20     # draw the "next" piece
21     drawPiece(nextPiece, pixelx=WINDOWWIDTH-120, pixely=100)

```

7개 블록이 각각 고유의 색을 가져야 하므로 원래 COLOR에 있는 색만으로는 부족하기 때문에 ORANGE, PURPLE, PINK의 RGB 값을 찾아 정의하고 COLOR 튜플 안에 추가해주었다. 또한 각 색깔들은 연한 색깔을 반드시 가져야 하므로 LIGHTORANGE, LIGHTPURPLE, LIGHTPINK도 LIGHTCOLOR 튜플 안에 추가하였다.

PIECES 딕셔너리에서 'color' 값을 임의로 추가하였다. S는 BLUE, Z는 GREEN, J는 RED, L은 YELLOW, I는 ORANGE, O는 PURPLE, 그리고 T는 PINK의 색을 갖도록 하였다. PIECES의 구조를 변경하였기 때문에 이에 맞게 블록 정보에 접근하는 모든 코드를 수정해야 한다.

getNewPiece()에서는 random.randint(0, len(PIECES[shape]) - 1) 부분을 random.randint(0, len(PIECES[shape]['shape']) - 1)로, drawPiece()에서는 shapeToDraw = PIECES[piece['shape']][piece['rotation']] 부분을 shapeToDraw = PIECES[piece['shape']]['shape'][piece['rotation']]로 수정하였다.

또한 코드의 가독성을 높이기 위해 addToBoard()에 piece_shape = PIECES[piece['shape']]['shape'][piece['rotation']] 변수를 추가하고, if문의 조건인 if PIECES[piece['shape']][piece['rotation']][y][x] != BLANK: 부분을 if piece_shape[y][x] != BLANK:로 수정하였다. 마찬가지로 isValidPosition()에 piece_shape = PIECES[piece['shape']]['shape'][piece['rotation']]를 추가하고, if isAboveBoard or PIECES[piece['shape']][piece['rotation']][y][x] == BLANK: 부분을 if isAboveBoard or piece_shape[y][x] == BLANK:로 수정하였다.

마지막으로 올바른 형식의 데이터가 함수에 전달되도록 drawNextPiece()의 인자를 piece에서 nextPiece로 변경하였다.

각 함수들의 역할에 대한 설명

1. main()

- Pygame 라이브러리를 초기화하고 게임 윈도우, 시계, 폰트를 설정한다.
- 'MY TETRIS'라는 시작 화면을 표시한 후 주 게임 루프를 시작한다.
- 게임의 음악을 무작위로 선택하여 재생한다.
- runGame() 함수를 호출하여 게임을 실행하고, 게임이 종료되면 'Over :(' 화면을 표시한다.

2. runGame()

- 게임 시작에 필요한 변수를 초기화하고, 게임 보드를 준비한다.
- 사용자의 입력을 받아 처리하고, 테트리미노의 이동, 회전, 속도 조절을 관리한다.
- 테트리미노가 보드 바닥에 닿거나 다른 테트리미노 위에 안착하면 보드에 테트리미노를 고정시키고, 줄이 완성되었는지 확인하여 점수를 계산한다.
- 게임 화면을 업데이트하고, 점수, 레벨, 경과 시간을 화면에 표시한다.

3. showTextScreen(text)

- 입력된 텍스트를 화면 중앙에 크게 표시한다.
- 게임의 시작, 일시 정지, 게임 오버 시에 사용자에게 정보를 제공한다.
- 사용자가 키 입력을 할 때까지 화면을 계속 표시한다.

4. checkForQuit()

- Pygame 이벤트 큐를 검사하여 QUIT 이벤트(창을 닫는 등)가 있는지 확인한다.
- 있으면 **terminate()**를 호출하여 게임을 종료한다.
- ESC 키를 누른 경우에도 게임을 종료한다.

5. calculateLevelAndFallFreq(score)

- 사용자의 점수에 기반하여 게임 레벨을 결정하고, 테트리미노가 떨어지는 빈도를 조정한다.
- 레벨이 높아질수록 테트리미노는 더 빠르게 떨어진다.

6. getNewPiece()

- 사용 가능한 테트리미노 중에서 하나를 무작위로 선택하고, 시작 위치와 회전 상태를 초기화한다.
- 이 새로운 테트리미노는 게임 보드 상단에서 시작된다.

7. addToBoard(board, piece)

- 활성 테트리미노를 게임 보드에 추가한다.
- 테트리미노가 최종적으로 착지한 위치의 보드 셀을 업데이트한다.

8. getBlankBoard()

- 게임의 시작 시나 리셋 시에 사용된다.
- BOARDWIDTH x BOARDHEIGHT 크기의 빈 게임 보드를 생성한다.

9. isValidPosition(board, piece, adjX=0, adjY=0)

- 테트리미노가 주어진 위치에 유효하게 놓일 수 있는지 검사한다.
- 범위를 벗어나거나, 다른 테트리미노와 겹치는 경우 유효하지 않는다.

10. removeCompleteLines(board)

- 게임 보드를 검사하여 완성된 라인을 찾고, 그 라인을 제거한 후 위에 있는 모든 라인을 아래로 이동시킨다.
- 완성된 라인 수에 따라 점수를 계산한다.

11. drawBox(boxx, boxy, color, pixelx=None, pixely=None)

- 주어진 위치에 상자(게임 피스의 일부)를 그린다.
- 위치는 보드 좌표나 픽셀 좌표로 지정할 수 있다.

12. drawBoard(board)

- 전체 게임 보드와 그 경계를 그린다.
- 각각의 셀을 그리기 위해 **drawBox()**를 호출한다.

13. drawStatus(score, level, elapsedTime)

- 화면에 현재 점수, 게임 레벨, 경과 시간을 표시한다.

14. drawPiece(piece, pixelx=None, pixely=None)

- 화면에 테트리미노를 그린다.
- 주어진 테트리미노의 형태와 색상을 사용하여 각 상자를 그린다.

15. drawNextPiece(nextPiece)

- 화면에 다음에 나올 테트리미노를 'Next.' 라벨과 함께 표시한다.

16. makeTextObjs(text, font, color)

- 주어진 텍스트와 폰트, 색상을 사용하여 텍스트 오브젝트와 위치를 반환한다.

17. terminate()

- Pygame을 종료하고, 시스템을 종료한다.

18. checkForKeyPress()

- 사용자의 키 입력을 감지한다.
- 키 입력을 받으면 해당 키를 반환하고, 아니면 None을 반환한다.

19. convertToPixelCoords(boxx, boxy)

- 보드 상의 좌표를 픽셀 좌표로 변환하여, 화면 상의 정확한 위치를 계산한다.
- 이는 테트리미노를 정확한 위치에 그리기 위해 사용된다

함수의 호출 순서 및 호출 조건에 대한 설명