오픈소스SW 과제중심수업1 보고서

2023026944 김동건

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

원본 코드의 169~172번 라인을 다음과 같이 수정하여 3개의 음악 중 1개가 재생되도록 수정하였다. 0부터 2까지의 랜덤한 번호를 뽑고, 이를 music_no라는 변수에 저장한다. 그리고 이 값에 따라 3가지 음악 중 1가지 음악이 재생되도록 코드를 변경하였다.

원본 코드:

```
if random.randint(0, 1) == 0:
    pygame.mixer.music.load('tetrisb.mid')
else:
    pygame.mixer.music.load('tetrisc.mid')
pygame.mixer.music.play(-1, 0.0)
```

수정 코드:

```
music_no = random.randint(0, 2)
if music_no == 0:
    pygame.mixer.music.load('Hover.mp3')
elif music_no == 1:
    pygame.mixer.music.load('Our_Lives_Past.mp3')
else:
    pygame.mixer.music.load('Platform_9.mp3')
pygame.mixer.music.play(-1, 0.0)
```

2. 상태창 이름을 '학번_이름' 으로 수정

main() 함수의 시작 부분(165번 라인)에 존재하는 pygame.display.set_caption()의 인자값을 'Tetromino'에서 '2023026944_김동건'으로 수정하였다.

원본 코드:

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

수정 코드:

```
pygame.display.set_caption('2023026944_김동건')
```

3. 게임시작화면의 문구를 MY TETRIS로 변경

showTextScreen() 함수의 설명을 보면, 이 함수는 a 키가 입력될 때 까지 스크린의 가운데에 크기가 큰 Text를 표시하는 함수라고 작성되어 있다. main() 함수 내 showTextScreen() (167번 라인)의 인자값으로 'Tetromino' 를 'MY TETRIS'로 수정하였다.

원본 코드:

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

수정 코드:

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

4. 게임시작화면의 문구 및 배경색을 노란색으로 변경

showTextScreen()의 역할이 a 키가 입력될 때 까지 스크린의 가운데에 크기가 큰 Text를 표시하는 역할이므로 이 역시 showTextScreen() 함수를 수정하여 변경할 수 있다. 함수 내 titleSurf, titleRect, pressKeySurf, pressKeyRect에 들어가 있는 makeTextObjs()의 인자값으로 text, font, color가 들어간다. 게임시작화면의 문구와 배경색을 노란색으로 변경하는 것이 목표이므로 color에 들어가는 인자를 수정해야 한다. 이 인자값으로 각각 TEXTCOLOR, TEXTSHADOWCOLOR가 넘어가므로 TEXTCOLOR를 LIGHTYELLOW, TEXTSHADOWCOLOR를 YELLOW로 변경하면 된다. 또한 YELLOW는 (155, 155, 0), LIGHTYELLOW는 (175, 175, 20)의 RGB 값을 갖는 상수로 선언되어 있다.

원본 코드(38~39번 라인):

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

수정 코드:

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

5. 게임 경과 시간을 초 단위로 표시

우선, runGame()의 변수 선언 부분에 다음과 같은 코드를 추가한다.

```
startTime = time.time()
playTime = 0
```

다음으로, game loop에 해당하는 while문 맨 첫 번째 줄에 다음 코드를 추가한다. 이는 startTime으로부터 얼마나 시간이 지났는지를 playTime에 대입시켜 준다.

```
playTime = (int) (time.time() - startTime)
```

그리고, drawStatus 함수를 수정한다. 새로 넘겨받을 매개변수를 추가하고, 그것을 출력할 코드를 새로 작성한다.

```
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)

    timeSurf = BASICFONT.render('Play Time: %s sec' % playTime, True,
TEXTCOLOR)
    timeRect = timeSurf.get_rect()
    timeRect.topleft = (WINDOWWIDTH - 550, 20)
    DISPLAYSURF.blit(timeSurf, timeRect)
```

마지막으로, runGame() 내 drawStatus() 함수의 인자로 playTime을 추가로 넘기도록 수정한다.

```
drawStatus(score, level, playTime)
```

6. 7개의 블록이 각각 고유의 색을 갖도록 코드를 수정하거나 추가

getNewPiece() 함수를 보면, newPiece에 color에 대한 value로 COLOR의 길이만큼 랜덤한 정수를 대입하고 있다. 이를 i라고 할때, COLOR[i]와 LIGHTCOLOR[i] 색상을 이용하여 테트로미노가 그려짐을 drawBox()를 통해 알 수 있다.

우선, 전역변수에 PIECE가 dict의 형태로 선언되어 있음을 알 수 있다. 각 key 별로 대응되는 테트로미노가 있으므로, 각 테트로미노에 대응하는 색상을 저장하는 dictionary를 새로 선언해야 한다. 이 dictionary의 이름은 PIECECOLOR이다.

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

색상의 종류가 부족해 보이므로 임의의 색상 몇 개를 추가한다.

```
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)
PURPLE      = (102,   0, 204)
LIGHTPURPLE = (153,  51, 255)
ORANGE      = (255,  51,   0)
LIGHTORANGE = (255, 112,  77)
```

그리고 이 색상들을 COLORS, LIGHTCOLORS에 각각 대응되도록 삽입한다.

```
COLORS      = (    BLUE,     GREEN,     RED,     YELLOW,     ORANGE,
PURPLE,  GRAY)
LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYELLOW,
LIGHTORANGE, LIGHTPURPLE, WHITE)
```

마지막으로, getNewPiece() 함수의 newPiece의 color value로 PIECECOLOR[shape], 즉 각 테트로미노 별로 이미 정해진 색상을 return하도록 수정한다.

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

7. 각 함수의 역할

runGame(): 게임을 시작하고, 게임이 원활히 진행될 수 있도록 한다. 테트로미노가 일정한 범위를 벗어난 곳에 놓이면 함수 실행을 종료한다.

getNewPiece(): 7종류의 테트로미노 중 랜덤으로 하나 뽑아서 그 블록에 대한 정보를 반환한다.

isValidPosition(): 블록이 놓일 수 있는 곳인지 판단한다.

drawStatus(): 현재 점수, 레벨 등 게임 진행 정보를 화면에 텍스트로 출력한다.

removeCompleteLines(): 한 행이 블록으로 가득 차 있는지 확인하는 isCompleteLine()이 true를 반환하면, 해당 라인을 지우고 그 위의 블록을 모두 밑으로 내린다. 그리고 제거된 라인 수를 반환한다.

8. 함수의 호출 순서 및 호출 조건에 대한 설명

Main() 함수부터 실행한다. 게임에 필요한 변수들을 미리 설정하고, showTextScreen()을 통해 타이틀을 출력하고 아무 키나 눌렀다 뗄 때까지 대기한다. 만약 그러한 이벤트가 발생하면, showTextScreen() 함수는 종료되고 runGame()이 실행된다. Board와 처음 놓아야 할 블록, 다음에 나올 블록을 로드하고, while True문에 진입하여 다음과 같은 행동을 isValidPosition()이 true

가 될 때까지, 즉 보드에 새로운 블록을 놓을 수 없을 때까지 반복한다.

- 키보드의 키를 입력하는 것에 대한 이벤트 리스너를 상시 대기시킨다. P키를 누르면 다시 아무 키나 누를 때까지 Pause를 띄워 게임을 일시정지 시키고, W, A, S, D 혹은 상, 하, 좌, 우 키가 입력되면 블록을 이동시키거나 회전시킨다. SPACE 키를 누르면 블록이 즉시 맨 아래로 떨어진다.
- 일정 시간이 지날 때마다 현재 블록을 한 칸 씩 밑으로 내린다. 이때 isValidPosition()을 통해 블록이 더 이상 내려갈 수 없으면 addToBoard()를 통해 보드에 블록을 추가시킨다. 또한 removeCompleteLines()를 통해 블록으로 꽉 찬 라인이 있으면 그 라인의 블록을 모두 제거하고 점수를 획득한다. 그리고 calculateLevelAndFallFreq() 함수를 실행하여 현재 점수에 맞춰 레벨과 블록이 떨어지는 속도를 조절한다.