

오픈소스SW 12주차 과제 보고서

2023021585 이동원

1) 체크리스트

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

```
1 while True:
2     if random.randint(0, 1) == 0:
3         # 현재 테트리스 게임의 배경음악을 주어진 3개의 음악 중 1개가 재생되도록 수정
4         pygame.mixer.music.load('Hover.mp3')
5     else:
6         pygame.mixer.music.load('Platform_9.mp3')
7     pygame.mixer.music.play(-1, 0.0)
8     runGame()
9     pygame.mixer.music.stop()
10    showTextScreen('Game Over')
```

함수 load()안에 '파일명'을 수정하여 기능 구현

2. 상태창 이름을 학번_이름 으로 수정

```
1 pygame.display.set_caption('2023021585_이동원')
```

함수 set_caption()안에 '2023021585_이동원'을 집어넣어 기능 구현

3. 게임시작화면의 문구를 MY TETRIS으로 변경

```
1 showTextScreen('MY TETRIS') # 게임시작화면의 문구를 MY TETRIS으로 변경
```

함수 showTextScreen()안에 'MY TETRIS'넣어 기능 구현

4. 게임시작화면의 문구 및 배경색을 노란색으로 변경

```
1 # 게임시작화면의 문구 및 배경색을 노란색으로 변경
2 TEXTCOLOR = YELLOW
3 TEXTSHADOWCOLOR = YELLOW
```

문구, 배경색을 결정하는 변수의 값을 YELLOW로 바꾼다.

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

```
1 def formatTime(seconds):
2     minutes = int(seconds // 60)
3     seconds = int(seconds % 60)
4     return f'{minutes:02}:{seconds:02}'
```

```
1 def drawStatus(score, level, elapsedTime):
2     scoreSurf = BASICFONT.render('Score: %s' % score, True, TEXTCOLOR)
3     scoreRect = scoreSurf.get_rect()
4     scoreRect.topleft = (WINDOWWIDTH - 150, 20)
5     DISPLAYSURF.blit(scoreSurf, scoreRect)
6
7     levelSurf = BASICFONT.render('Level: %s' % level, True, TEXTCOLOR)
8     levelRect = levelSurf.get_rect()
9     levelRect.topleft = (WINDOWWIDTH - 150, 50)
10    DISPLAYSURF.blit(levelSurf, levelRect)
11    # 게임 경과 시간을 초 단위로 표시
12    elapsedTimeSurf = BASICFONT.render('Time: %s' % formatTime(elapsedTime), True, TEXTCOLOR)
13    elapsedTimeRect = elapsedTimeSurf.get_rect()
14    elapsedTimeRect.topleft = (20, 20)
15    DISPLAYSURF.blit(elapsedTimeSurf, elapsedTimeRect)
```

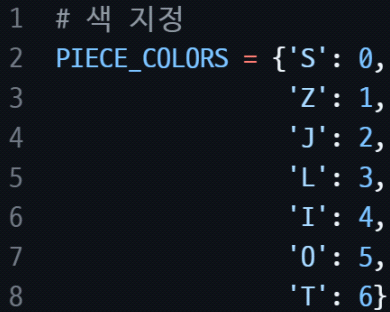
함수 formatTime()를 만들어 시간을 계산하는 기능을 구현한다.

함수 drawStatus()에 주석 아래 시간을 출력하는 기능을 구현한다.

6. 7개의 블록이 각각 고유의 색을 갖도록 코드를 수정하거나 추가

```
1 # 색 추가
2 ORANGE = (255, 165, 0)
3 LIGHTORANGE = (255, 200, 115)
4 INDIGO = (75, 0, 130)
5 LIGHTINDIGO = (165, 100, 255)
6 VIOLET = (148, 0, 211)
7 LIGHTVIOLET = (230, 150, 255)
```

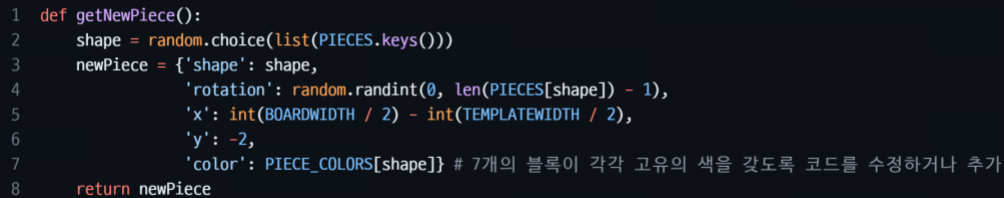
```
1 COLORS = (BLUE, GREEN, RED, YELLOW, ORANGE, INDIGO, VIOLET) # 원소 추가
2 LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYELLOW, LIGHTORANGE, LIGHTINDIGO, LIGHTVIOLET)
```



```

1 # 색 지정
2 PIECE_COLORS = {'S': 0,
3                 'Z': 1,
4                 'J': 2,
5                 'L': 3,
6                 'I': 4,
7                 'O': 5,
8                 'T': 6}

```



```

1 def getNewPiece():
2     shape = random.choice(list(PIECES.keys()))
3     newPiece = {'shape': shape,
4                 'rotation': random.randint(0, len(PIECES[shape]) - 1),
5                 'x': int(BOARDWIDTH / 2) - int(TEMPLATEWIDTH / 2),
6                 'y': -2,
7                 'color': PIECE_COLORS[shape]} # 7개의 블록이 각각 고유의 색을 갖도록 코드를 수정하거나 추가
8     return newPiece

```

3가지 색을 추가하고 집합에도 추가한다.

PIECE_COLORS 변수를 추가하여 각 블록마다 색을 지정한다.

함수 getNewPiece()안에 딕셔너리를 수정한다. ('color' : PIECE_COLORS[shape])

2) 각 함수의 역할

main : main() 함수는 게임의 메인 루프를 담당한다. pygame을 초기화하고, 화면, 폰트, 타이머 등을 설정한다. 게임의 시작 화면을 표시하고, 게임 음악을 재생하며, runGame() 함수를 통해 게임을 실행한다. 게임이 종료되면 음악을 멈추고 'Game Over' 화면을 표시한다.

runGame : runGame() 함수는 게임의 실행 루프를 설정한다. 보드를 초기화하고, 시간과 이동 상태를 초기화하며, 점수와 레벨을 초기화한다. 현재 떨어지는 블록과 다음 블록을 생성합니다. 게임 루프에서 현재 떨어지는 블록이 없으면, 다음 블록을 현재 블록으로 설정하고, 새로운 블록을 생성한다. 생성된 블록이 유효한 위치에 있지 않으면 게임을 종료하고 재시작한다. 키보드 이벤트에 따라 블록을 이동하거나 회전시키며, 일시 정지 기능도 처리한다. 블록이 움직이는 시간 간격을 확인하고, 유효한 위치에 있는지 확인하여 블록을 이동시킨다. 블록이 바닥에 도달하면 보드에 추가하고, 완성된 라인을 제거하여 점수를 업데이트한다. 화면을 그리기 위해 필요한 모든 요소를 업데이트하고, 화면을 업데이트한다.

formatTime : 게임 경과 시간을 초 단위로 표현한다.

makeTextObjs : 텍스트를 렌더링하여 서피스와 직사각형 객체를 반환한다.

terminate : 게임 종료 역할을 한다.

checkForKeyPress : 키 입력을 확인한다.

showTextScreen : 텍스트 화면을 표시한다.

checkForQuit : 게임 종료 이벤트를 처리한다.

calculateLevelAndFallFreq : 점수에 따라 레벨과 블록이 떨어지는 빈도를 계산한다.

getNewPiece : 새로운 블록을 생성하여 반환한다.

addToBoard : 블록을 보드에 추가한다.

getBlankBoard : 빈 보드를 생성하여 반환한다.

isOnBoard : 주어진 좌표가 보드 내에 있는지 확인한다.

isValidPosition : 주어진 블록이 보드 내의 유효한 위치에 있는지 확인한다.

isCompleterline : 주어진 줄이 완성되었는지 확인한다.

removeCompleteLines : 완성된 줄을 제거하고, 점수를 반환한다.

convertToPixelCoords : 보드좌표를 픽셀 좌표로 변환한다.

drawBox : 인자를 바탕으로 블록을 그린다.

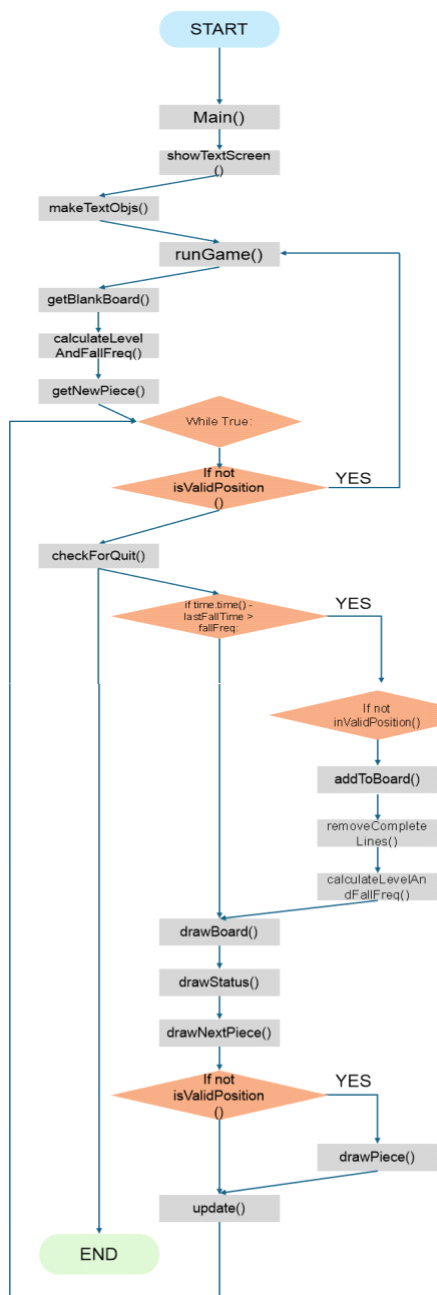
drawBoard : 보드를 그린다.

drawStatus : 게임의 상태 (점수, 레벨, 경과 시간)를 그린다.

drawPiece : 블록을 그리기 위해 drawBox 인자값을 세팅한다.

drawNextPiece : 다음에 나올 블록을 그린다.

3) 함수의 호출 순서 및 호출 조건에 대한 설명



프로그램 실행을 하게 되면 main함수부터 실행이 된다.

이후 main함수 내에 있는 실행문들이 차례차례 실행된다. 먼저 showTextScreen이 호출되고 showTextScreen함수 내에 있는 실행문인 makeTextObjs함수가 호출되어 게임시작화면에 문구를 띄어준다. 이후 main함수 안에 있는 runGame함수가 실행된다.

runGame함수 내에 있는 실행문들인 getBlankBoard, calculateLevelAndFall

Freq, getNewPiece함수가 실행이 되어 게임의 정보들을 받아온다.

이후 조건문에서 isValidPosition함수가 False라면 runGame함수를 종료시키고 showTextScreen함수를 실행시켜

‘Game Over’를 화면에 띄운 후 while문에 의해 다시

runGame함수를 실행한다. 조건문에서 걸리지 않는다면

checkForQuit함수가 실행되어 프로그램 종료 키 여부를 확인한다.

종료 확인이 되었다면 프로그램을 종료하고 아니라면 계속

이어진다. 이후에는 조작에 관한 실행문들이 이어진다. 그 중 if

time.time() - lastFallTime > fallFreq 조건문을 만족하고 not

isValidPosition을 만족한다면 addToBoard,

removeCompleteLines, calculateLevelAndFallFreq

함수가 호출이 되어 화면에 띄울 정보들을 업데이트 하여

drawBoard, drawStatus, drawNextPiece함수를 통해 프로그램

내에서 그린다. 이후 조건문 if not is ValidPosition을 통해

만족시키면 drawPiece함수를 호출하고 만족시키지 않는다면

update함수를 통해 프로그램 내에서 그린 것들을 화면에 띄운다.

update가 끝나면 while True에 의해 특정 상황을 제외하고 무한 반복이 된다.

4) github 주소

<https://github.com/DongWonLee2/osw>