# 당현송 과제중심수업 1주차

```
while True: # game loop
    if random.randint(0, 1) == 0:
        pygame.mixer.music.load('Hover.mp3')
    else:
        pygame.mixer.music.load('Hover.mp3')
```

체크리스트 1. 기존 코드에 tetrisb.mid이라는 부분을 음악 파일명인 Hover.mp3로 수정한 뒤 음악 파일을 같은 디렉토리로 옮기는 방식으로 해결

```
def main():
    global FPSCLOCK, DISPLAYSURF, BASICFONT, BIGFONT
    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
    BASICFONT = pygame.font.Font('freesansbold.ttf', 18)
    BIGFONT = pygame.font.Font('freesansbold.ttf', 100)
    pygame.display.set_caption('2021022379 DANGHYEONSONG')
```

체크리스트 2. 해당 코드 마지막 부분 Tetromino를 학번, 이름인 2021022379 DANGHYEONSONG으로 수정

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

체크리스트 3. 화면 표시 문구를 MY TETRIS로 변경하기 위해 showTextScreen 부분을 수정

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

체크리스트 4. 게임 시작화면의 문구, 문구의 그림자색을 노랑으로 변경하기 위해 색깔 수정.

```python
def drawStatus(score, leve
    # 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)

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

(function) render: Any

```python
while True: # game loop
    playTime = int(time.time()-startTime)
    if fallingPiece == None:
        # No falling piece in play, so start a new piece at the top
        fallingPiece = nextPiece
        nextPiece = getNewPiece()
        lastFallTime = time.time() # reset lastFallTime

        if not isValidPosition(board, fallingPiece):
            return # can't fit a new piece on the board, so game over
```

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

체크리스트 5. playTime에 대해 정의하고 그 후 playTime을 이용한 계산을 통해 초단위로 게임시간을 표시하고  playTime = int(time.time()-startTime)으로 새게임이 시작될때 시간이 초기화된다.

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

체크리스트 6. 7개의 블록이 고유의 색을 갖도록 blockcolors라는 디렉토리를 추가하여 위에 생성한 clolors 디렉토리와 매핑될 수 있도록 drawbox에서 매핑.

```
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
```

```
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))
```

이후에는 blockclolor 디렉토리에 각각에 맞는 인덱스를 고정해주면 고유한 블록의 색깔이 지정.

체크리스트 7. 함수의 고유한 역할과 호출순서 및 호출조건

**주요 함수의 역할**

1. **main() 함수**

- **역할:** 게임의 초기 설정을 수행하고, 메인 게임 루프를 시작합니다.
- **호출 순서 및 조건:**
  - 스크립트가 직접 실행될 때 (**if __name__ == '__main__': main()**).
  - 이 함수는 Pygame을 초기화하고, 디스플레이와 폰트를 설정한 후, 메인 게임 루프를 시작합니다.

2. **runGame() 함수**

- **역할:** 게임의 주요 논리를 처리하고, 게임 루프를 통해 게임 상태를 업데이트하고 렌더링합니다.
- **호출 순서 및 조건:**
  - **main()** 함수 내에서 호출됩니다.
  - 게임이 시작되면, 메인 게임 루프에서 계속 호출됩니다.

    - 게임 루프에서 키 입력을 처리하고, 조각 이동, 충돌 감지, 라인 제거 등의 작업을 수행합니다.

3. **drawBoard(board) 함수**
    - **역할:** 게임 보드를 화면에 그립니다.
    - **호출 순서 및 조건:**
        - **runGame()** 함수 내에서 호출됩니다.
        - 매 프레임마다 게임 보드를 업데이트하고 렌더링할 때 호출됩니다.

**함수 호출 순서 및 조건**

1. **main() 함수**
    - 게임이 실행될 때 (**if __name__ == '__main__': main()** 조건에 의해 호출).
    - Pygame을 초기화하고, 디스플레이를 설정하며, **runGame()** 함수를 호출합니다.
    - 게임 루프에서 배경 음악을 재생하고, 게임이 종료되면 게임 오버 화면을 표시합니다.
2. **runGame() 함수**
    - **main()** 함수 내에서 호출됩니다.
    - 게임 보드 초기화, 조각 생성, 타이머 설정 등의 초기 설정을 수행합니다.
    - 게임 루프를 통해 지속적으로 호출되며, 다음 작업들을 수행합니다:
        - 키 입력 이벤트 처리
        - 조각 이동 및 회전
        - 충돌 감지 및 조각 고정
        - 완성된 라인 제거
        - 점수 및 레벨 업데이트
        - 화면 업데이트 및 게임 보드 그리기 (**drawBoard(board)** 함수 호출)
        - 게임 오버 조건 감지
3. **drawBoard(board) 함수**
    - **runGame()** 함수 내에서 호출됩니다.
    - 매 프레임마다 호출되어, 현재 게임 보드 상태를 화면에 그립니다.

- 보드 경계를 그리며, 각 셀을 순회하면서 셀이 비어 있지 않으면 해당 위치에 박스를 그립니다.

https://github.com/danga011/osw