# 테트리스 보고서

2021067710 이동규

**체크리스트**

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

```
whileTrue: # game loop
        ifrandom.randint(0, 1) ==0:
            pygame.mixer.music.load('Hover.mp3')
        else:
            pygame.mixer.music.load('Hover.mp3')
        pygame.mixer.music.play(-1, 0.0)
        runGame()
        pygame.mixer.music.stop()
        showTextScreen('Game Over')
```

Hover.mp3파일 로드해서 음악 삽입

게임 실행시hover음악이 아주 잘 들린다.

2. 상태창 이름을 학번_이름 으로 수정

```
efmain():
    globalFPSCLOCK, 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('2021067710 이동규')
```

하단에 set_caption에 학번이름을 넣고 상태창 수정

3. 게임시작화면의 문구를 MY TETRIS으로 변경

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

를 아래와 같이 변경

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

게임 오버시 표시

```
showTextScreen('Over :(')
```

로 변경

게임 시작 표기화면

```
  pressKeySurf, pressKeyRect=makeTextObjs('Press a key to play! pause key is p', BASICFONT, TEXTCOLOR)
```

로 변경

게임 일시 정지일시

```
showTextScreen('Paused')
```

를

```
 showTextScreen('Get a rest!')
```

로 변경

4. 게임시작화면의 문구 및 배경색을 노란색으로 변경

```
BORDERCOLOR=BLUE
BGCOLOR=BLACK
TEXTCOLOR=YELLOW
TEXTSHADOWCOLOR=YELLOW
```

Textcolor및 textshadowcolor를 둘다 yellow로 변경

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

```
defrunGame():
```

밑에

```
 startTime=time.time()
```

를 추가해 시작시간 기록 후 밑에 Whie True에

```
    whileTrue: # game loop
        currentTime=time.time()
        passed_time=startTime-currentTime
```

추가

DefdrawStatus에 passed_time추가

```
defdrawStatus(score, level, passed_time):
```

하단 defdrawStatus에 시간표기함수 추가

```
    # draw the time text
    passedTimeSurf=BASICFONT.render('Time: %.2f'%passed_time, True, TEXTCOLOR)  # 경과 시간 표시
    passedTimeRect=passedTimeSurf.get_rect()
    passedTimeRect.topleft= (WINDOWWIDTH-150, 0)
    DISPLAYSURF.blit(passedTimeSurf, passedTimeRect)
```

이후defrungame하단에 아래와 같이 passed_time추가

```
# drawing everything on the screen
        DISPLAYSURF.fill(BGCOLOR)
        drawBoard(board)
        drawStatus(score, level, passed_time)
```

6. 7개의 블록이 각각 고유의 색을 갖도록 코드를 수정하거나 추가

```
defgetNewPiece():
    # 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': random.randint(0, len(COLORS)-1)}
```

```
    returnnewPiece
```

이 함수의 'color' :부분에서 색을 선택한다는 것을 알아냈다

Color:random함수를 지우고 color함수의 첫번째인 파란색이 나와 모든 블록이 파란색으로 통일되게 나오나 코드를 바꾸고 디버깅을 해도 계속된 실패로 각각 고유의 색을 갖도록 코드를 구현하는데에 실패했다.

# 각 함수들의 설명 및 순서 + 호출조건

## 1. main()

**호출 시점**: 프로그램이 실행될 때 (if __name__ == '__main__': main()).

**호출 조건**: 프로그램 실행 시 자동으로 호출됨.

**역할**: 게임을 초기화하고 메인 게임 루프를 실행

1. pygame.init()을 호출하여 Pygame 을 초기화
2. 화면 크기, 제목 등을 설정하는 함수
3. 폰트와 사운드를 로드
4. showTextScreen('제목')호출-> 시작 화면 표시.
5. runGame()을 호출하여 게임을 실행
6. 게임이 종료되면 showTextScreen('Game Over')를 호출하여 게임 오버 화면을 표시
7. 게임종료 (terminate() 호출).

## 2. runGame()

**호출 시점**: 게임 시작 시 및 게임 오버 시.

**호출 조건**: main() 함수에서 showTextScreen('MY TETRIS') 호출, 게임 오버 시 showTextScreen('Over :(') 호출.

**역할**: 게임의 주요 루프를 실행하여, 게임의 상태를 업데이트하고 화면을 렌더링

1. 게임 보드와 도형을 초기화 (board, fallingPiece, nextPiece 등).
2. 점수와 레벨을 초기화.
3. 게임 루프과정
    - 키 입력을 처리합니다 (moveLeft, moveRight, rotate, fallingFast 등).
    - 위 과정을 통해 도형을 이동하거나 회전
    - 도형이 바닥에 도달했는지 확인하고, 도달했다면 보드에 추가
    - 완성된 행이 있는지 확인하고, 제거하고 새로운 도형 생성
    - 화면업데이트 (drawBoard, drawPiece, drawStatus 등).
    - 주기적으로 도형이 자동으로 떨어지도록 처리
4. 유효하지 않은 위치에 도형이 놓이면 게임을 종료한다. (return).

### 3. makeTextObjs(text, font, color)

**호출 시점**: 게임 루프 내에서.

**호출 조건**: main() 함수에서 텍스트 화면 이후 호출됨.

**역할**: 텍스트와 이미지,폰트를 받아 화면에 출력

1. 텍스트 준비 / 폰트 로딩 / 텍스트 랜더링/ 이미지 반환

### 4. terminate()

**호출 시점**: 게임 시작 시 및 새로운 빈 보드가 필요할 때.

**호출 조건**: runGame() 함수 내에서 보드를 초기화할 때 호출됨.

**역할**: 게임을 종료합니다.

1. pygame.quit()을 호출하여 Pygame 을 종료합니다.
2. sys.exit()를 호출하여 프로그램을 종료합니다.

### 5. checkForKeyPress()

**호출 시점**: 점수 변경 시.

**호출 조건**: runGame() 함수 내에서 점수 변경 시 호출됨.

**역할**: 키가 눌렸는지 확인한다

1. 모든 Pygame 이벤트를 체크한다. (pygame.event.get()).
2. 키보드 이벤트가 있는지 확인 (KEYUP).
3. 키보드 이벤트가 있으면 해당 키 코드를 반환합니다.
4. 없다면 None 을 반환합니다.

### 5. showTextScreen(text)

**호출 시점**: 새로운 블록이 필요할 때.

**호출 조건**: runGame() 함수 내에서 현재 블록이 떨어질 수 없을 때 호출됨.

**역할**: 주어진 텍스트를 화면 중앙에 표시하고, 키가 눌릴 때까지 대기합니다.

1. 화면을 지웁니다 (DISPLAYSURF.fill(BGCOLOR)).
2. makeTextObjs 를 호출하여 텍스트 서피스를 만듭니다.
3. 텍스트를 화면 중앙에 렌더링합니다 (DISPLAYSURF.blit).
4. 화면을 업데이트합니다 (pygame.display.update()).
5. 사용자가 키를 누를 때까지 대기합니다 (checkForKeyPress).

### 7. checkForQuit()

**호출 시점**: 이벤트 루프 내에서.

**호출 조건**: runGame() 및 checkForKeyPress() 함수 내에서 호출됨.

**역할**: 종료 이벤트가 발생했는지 확인하고, 발생하면 게임을 종료합니다.

**실행 과정**:

1. 모든 Pygame 이벤트를 체크합니다 (pygame.event.get()).
2. QUIT 이벤트가 있는지 확인합니다 (QUIT).
3. 있으면 terminate 를 호출하여 게임을 종료합니다.

8. **calculateLevelAndFallFreq(score)**

**호출 시점**: 키 입력 대기 시.

**호출 조건**: showTextScreen() 함수 내에서 호출됨.

**역할**: 점수를 기반으로 레벨과 도형이 떨어지는 주기를 계산합니다.

1. 점수를 기반으로 레벨을 계산합니다 (score // 10).
2. 레벨에 따라 도형이 떨어지는 속도를 결정합니다 (0.27 - (level * 0.02)).

9. **getNewPiece()**

**호출 시점**: 블록 이동 및 회전 시.

**호출 조건**: runGame() 함수 내에서 블록 이동 및 회전 시 호출됨.

**역할**: 새로운 도형을 생성합니다.

1. 무작위로 새로운 도형을 선택합니다 (random.choice(PIECES.keys())).
2. 초기 위치와 회전을 설정합니다 ('x': int(BOARDWIDTH / 2) - int(TEMPLATEWIDTH / 2)).

10. **addToBoard(board, piece)**

**호출 시점**: 블록이 고정될 때.

**호출 조건**: runGame() 함수 내에서 블록이 고정될 때 호출됨.

**역할**: 현재 도형을 보드에 추가합니다.

1. 도형의 각 블록을 보드에 고정시킵니다.
2. 보드의 해당 위치를 도형의 색상으로 설정합니다.

**11. getBlankBoard()**

**호출 시점**: 라인 완성 시.

**호출 조건**: runGame() 함수 내에서 블록이 고정된 후 호출됨.

**역할**: 빈 게임 보드를 생성합니다.

1. 빈 리스트를 만들어 게임 보드를 초기화합니다.
2. 보드의 각 행을 빈 상태로 채웁니다 (BLANK).

**12. isOnBoard(x, y)**

**호출 시점**: 화면에 블록을 그릴 때.

**호출 조건**: drawBox() 및 drawPiece() 함수 내에서 호출됨.

**역할**: 좌표가 보드 내에 있는지 확인합니다.

1. 주어진 좌표가 보드의 범위 내에 있는지 검사합니다 (0 <= x < BOARDWIDTH 및 y < BOARDHEIGHT).

**13. isValidPosition(board, piece, adjX=0, adjY=0)**

**호출 시점**: 화면에 박스를 그릴 때.단일 박스 그리기.

**호출 조건**: drawBoard() 및 drawPiece() 함수 내에서 호출됨.

**역할**: 도형이 현재 위치에 놓일 수 있는지 확인합니다.

1. 도형의 각 블록이 유효한 위치에 있는지 검사합니다.
2. 보드의 범위를 벗어나지 않는지 확인합니다.
3. 보드의 다른 블록과 충돌하지 않는지 확인합니다.

**14. isCompleteLine(board, y)**

**호출 시점**: 매 프레임마다.

**호출 조건**: runGame() 함수 내에서 매 프레임마다 호출됨.

**역할**: 주어진 행이 완성되었는지 확인합니다.

1. 주어진 행의 모든 셀이 채워져 있는지 검사합니다.
2. 모든 셀이 채워져 있으면 True 를 반환합니다.

**15. removeCompleteLines(board)**

**호출 시점**: 매 프레임마다.

**호출 조건**: runGame() 함수 내에서 매 프레임마다 호출됨.

**역할**: 완성된 행을 제거하고 점수를 갱신합니다.

1. 보드를 스캔하여 완성된 행을 찾습니다.
2. 완성된 행을 제거합니다.
3. 제거된 행 위의 모든 블록을 한 줄씩 아래로 내립니다.
4. 제거된 행의 수만큼 점수를 갱신합니다.

**16. convertToPixelCoords(boxx, boxy)**

**호출 시점**: 화면에 블록을 그릴 때.

**호출 조건**: runGame() 및 drawNextPiece() 함수 내에서 호출됨.

**역할**: 게임 보드 좌표를 픽셀 좌표로 변환합니다.

1. 보드 좌표를 화면에 그릴 수 있는 픽셀 좌표로 변환합니다.
2. 각 블록의 좌표를 계산하여 화면에 적절한 위치를 찾습니다.

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

**호출 시점**: 다음 블록을 그릴 때.

**호출 조건**: runGame() 함수 내에서 매 프레임마다 호출됨.

**역할**: 주어진 좌표에 상자를 그립니다.

1. 주어진 좌표에 상자를 그리고, 색상을 지정합니다.
2. 보드 좌표를 픽셀 좌표로 변환합니다 (convertToPixelCoords 호출).
3. 상자를 화면에 그립니다 (pygame.draw.rect).

### 18. drawBoard(board)

**호출 시점**: 게임 종료 시.

**호출 조건**: checkForQuit() 및 기타 종료 조건 발생 시 호출됨.

**역할**: 게임 보드를 화면에 그립니다.

1. 보드의 각 셀을 검사합니다.
2. 채워진 셀을 화면에 그립니다 (drawBox 호출).

### 19. drawStatus(score, level, passed_time)

**호출 시점**: 새로운 게임 조각이 다음에 나타날 것을 표시할 필요가 있을 때
**호출 조건**: 게임 화면에 다음 조각을 미리 보여줄 때와 현재 조각이 보드에 배치된 후, 다음 조각을 미리 준비할 때.
**역할**: 현재 점수, 레벨, 경과 시간을 화면에 그립니다.

1. 화면의 특정 위치에 점수를 텍스트로 렌더링합니다 (makeTextObjs 호출).
2. 텍스트를 화면에 표시합니다 (DISPLAYSURF.blit).

### 20. drawPiece(piece, pixelx=None, pixely=None)

**호출 시점**: 현재 화면에 특정 조각을 그려야 할 때 호출됩니다.
**호출 조건**: 새 조각이 나타날 때 + 위치변경할때

**역할**: 주어진 도형을 화면에 그립니다.

1. 도형의 각 블록을 검사합니다.
2. 각 블록을 화면에 그립니다 (drawBox 호출).

### 21. drawNextPiece(piece)

**호출 시점**: 게임의 상태 정보(점수, 레벨, 경과 시간 등)를 업데이트해야 할 때 호출됩니다.
**호출 조건**: 점수 변경될 때 + 레벨 변경될 때 + 경과 시간이 업데이트 될 때

**역할**: 다음 도형을 화면에 그립니다.

1. 다음 도형을 화면의 특정 위치에 그립니다.
2. 도형의 각 블록을 그립니다 (drawBox 호출).