1. 음악실행

변경 전 코드

```
    while True: # game loop
        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)
        runGame()
        pygame.mixer.music.stop()
        showTextScreen('Game Over')
```

변경 후 코드

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

        pygame.mixer.music.play(-1, 0.0)
        runGame()
        pygame.mixer.music.stop()
        showTextScreen('Game Over')
```

2) 각 함수의 역할

1. **main()**
   - 게임의 초기 설정을 하고 게임 루프를 실행합니다.
   - 초기화된 Pygame 객체를 사용하여 화면을 설정하고, 글꼴을 로드하며, 초기 텍스트 화면을 보여줍니다.
   - 게임 루프에서 무작위로 음악을 선택하여 재생하고 게임을 실행합니다.

2. **runGame()**
   - 실제 게임을 실행하는 메인 게임 루프를 포함합니다.
   - 게임 보드를 초기화하고, 테트리스 조각을 생성하며, 게임 상태를 업데이트하고 그립니다.
   - 사용자 입력(키보드)을 처리하여 조각을 이동하거나 회전시킵니다.

3. **makeTextObjs(text, font, color)**
   - 텍스트 표면과 해당 텍스트의 사각형 객체를 생성하여 반환합니다.

4. **terminate()**
   - Pygame을 종료하고 프로그램을 종료합니다.

5. **checkForKeyPress()**
   - 이벤트 큐에서 KEYUP 이벤트를 검사하고, 해당 키 코드를 반환합니다.
   - 게임을 일시정지하거나 메뉴 화면에서 키 입력을 기다리는 데 사용됩니다.

6. **showTextScreen(text)**
   - 중앙에 큰 텍스트를 표시하고, 사용자가 키를 누를 때까지 기다립니다.
   - 게임 시작 화면이나 게임 오버 화면에 사용됩니다.

7. **checkForQuit()**
   - QUIT 이벤트나 Esc 키 입력을 검사하여 프로그램을 종료합니다.

8. **calculateLevelAndFallFreq(score)**
   - 점수를 기반으로 현재 레벨과 조각이 떨어지는 주기를 계산합니다.

9. **getNewPiece()**
   - 무작위로 새로운 테트리스 조각을 생성하여 반환합니다.

10. **addToBoard(board, piece)**
    - 현재 조각의 위치와 모양을 보드에 추가합니다.

11. **getBlankBoard()**
    - 빈 테트리스 보드를 생성하여 반환합니다.

12. **isOnBoard(x, y)**
    - 주어진 좌표가 보드 내에 있는지 확인합니다.

13. **isValidPosition(board, piece, adjX=0, adjY=0)**
    - 조각이 주어진 위치에 유효한지(충돌하지 않는지) 확인합니다.

14. **isCompleteLine(board, y)**
    - 주어진 행이 완전히 채워졌는지 확인합니다.

15. **removeCompleteLines(board)**
    - 완전히 채워진 행을 제거하고, 제거된 행 수를 반환합니다.

16. **convertToPixelCoords(boxx, boxy)**
    - 보드의 xy 좌표를 화면의 픽셀 좌표로 변환합니다.

17. **drawBox(boxx, boxy, color, pixelx=None, pixely=None)**
    - 단일 테트리스 박스를 화면에 그립니다.

18. **drawBoard(board)**
    - 보드와 개별 박스를 화면에 그립니다.

19. **drawStatus(score, level)**
    - 현재 점수와 레벨을 화면에 그립니다.

20. **drawPiece(piece, pixelx=None, pixely=None)**
    - 주어진 위치에 테트리스 조각을 그립니다.

21. **drawNextPiece(piece)**
    - "Next" 텍스트와 다음 조각을 화면에 그립니다.

3) 함수의 호출 순서 및 호출 조건

1. **프로그램 시작**
   - `if __name__ == '__main__': main()`
     - 프로그램이 시작되면 `main()` 함수가 호출됩니다.

2. **main()**
   - Pygame을 초기화하고 게임 화면과 글꼴을 설정합니다.
   - `showTextScreen('Tetromino')`을 호출하여 시작 화면을 보여줍니다.

- 무작위로 음악을 선택하고 재생합니다.
- `runGame()`을 호출하여 게임을 실행합니다.
- 게임이 종료되면 `showTextScreen('Game Over')`를 호출하여 게임 오버 화면을 보여줍니다.

3. **runGame()**
- 게임 보드와 조각을 초기화합니다.
- 메인 게임 루프를 시작합니다.
  - `checkForQuit()`을 호출하여 게임 종료 여부를 확인합니다.
  - Pygame 이벤트를 처리하여 조각을 이동하거나 회전시킵니다.
  - 조각의 이동, 회전, 낙하를 처리합니다.
  - 보드 상태를 업데이트하고 그립니다.

4. **게임 루프에서 호출되는 함수들**
- `getBlankBoard()`, `getNewPiece()`, `calculateLevelAndFallFreq()`, `isValidPosition()`, `addToBoard()`, `removeCompleteLines()`, `drawBoard()`, `drawPiece()`, `drawStatus()`, `drawNextPiece()` 등 다양한 함수가 게임의 상태를 관리하고 화면을 업데이트하기 위해 호출됩니다.

5. **이벤트 처리**
- 키 입력에 따라 `checkForKeyPress()`, `checkForQuit()`, `terminate()` 등의 함수가 호출됩니다.

6. **게임 종료**
- `terminate()` 함수가 호출되어 Pygame을 종료하고 프로그램이 종료됩니다.

각 함수는 특정한 조건에서 호출되며, 게임의 흐름에 따라 필요에 따라 호출됩니다. `main()` 함수가 전체 게임 흐름을 관리하며, `runGame()` 함수가 실제 게임 플레이를 처리합니다.

2.
1)
변경전 코드
```
    showTextScreen('Tetromino')
```

변경후 코드
```
pygame.display.set_caption('2023031058_이재원')

pygame.display.set_caption('2023031058_이재원')
```
2)함수설명
이 함수는 Pygame 창의 제목을 설정합니다. 즉, 게임 창의 상단 표시줄에 표시될 제목을 설정하는 역할을 합니다. 여기서는 '2023031058_이재원 - Tetromino'으로 설정되어 있습니다.
3)함수 호출 순서 및 호출 조건에 대한 설명
이 함수는 main() 함수에서 호출됩니다.
Pygame 초기화 이후에 호출되며, 게임 창의 제목을 설정하기 위해 사용됩니다.
이 함수는 pygame.init() 다음에 호출되므로, Pygame을 초기화한 후에 실행되어야 합니다.
따라서 main() 함수의 시작 부분에 위치하여야 합니다.

4.
1)
수정전

```
def showTextScreen(text):
    # This function displays large text in the
    # center of the screen until a key is pressed.
    # Draw the text drop shadow
    titleSurf, titleRect = makeTextObjs(text, BIGFONT, TEXTSHADOWCOLOR)
    titleRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2))
    DISPLAYSURF.blit(titleSurf, titleRect)

    # Draw the text
    titleSurf, titleRect = makeTextObjs(text, BIGFONT, TEXTCOLOR)
    titleRect.center = (int(WINDOWWIDTH / 2) - 3, int(WINDOWHEIGHT / 2) - 3)
    DISPLAYSURF.blit(titleSurf, titleRect)

    # Draw the additional "Press a key to play." text.
    pressKeySurf, pressKeyRect = makeTextObjs('Press a key to play.', BASICFONT,
TEXTCOLOR)
    pressKeyRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2) + 100)
    DISPLAYSURF.blit(pressKeySurf, pressKeyRect)

    while checkForKeyPress() == None:
        pygame.display.update()
        FPSCLOCK.tick()
```

수정후(함수의 역할 포함)

```
def showTextScreen(text):
    # 이 함수는 화면 중앙에 큰 텍스트를 표시하고
    # 키가 눌릴 때까지 기다립니다.
    # 텍스트 그림자 그리기
    titleSurf, titleRect = makeTextObjs(text, BIGFONT, TEXTSHADOWCOLOR)
    titleRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2))
    DISPLAYSURF.blit(titleSurf, titleRect)

    # 텍스트 그리기
    titleSurf, titleRect = makeTextObjs(text, BIGFONT, YELLOW)  # 노란색 텍스트 색상
    titleRect.center = (int(WINDOWWIDTH / 2) - 3, int(WINDOWHEIGHT / 2) - 3)
    DISPLAYSURF.blit(titleSurf, titleRect)

    # 배경색을 노란색으로 변경
    DISPLAYSURF.fill(YELLOW)

    # 추가 텍스트 "Press a key to play." 그리기
    pressKeySurf, pressKeyRect = makeTextObjs('Press a key to play.', BASICFONT,
TEXTCOLOR)
    pressKeyRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2) + 100)
    DISPLAYSURF.blit(pressKeySurf, pressKeyRect)

    while checkForKeyPress() == None:
        pygame.display.update()
        FPSCLOCK.tick()
```

3)
showTextScreen('MY TETRIS') 함수는 게임 시작 화면을 표시하는 함수입니다. 이 함수는 게임이 시작될 때 호출되며, 다음과 같은 단계를 거쳐 실행됩니다.
큰 텍스트를 중앙에 표시하고 키 입력을 기다립니다.
텍스트를 그림자와 함께 표시합니다.
배경을 노란색으로 채웁니다.
"Press a key to play."와 같은 부가 텍스트를 추가로 그립니다.
키 입력을 대기합니다. 사용자가 키를 누를 때까지 화면이 유지됩니다.
이 함수는 사용자가 키를 누를 때까지 계속해서 대기하므로, 사용자가 게임을 시작하고자 할 때까지 화면이 유지됩니다. 사용자가 키를 누르면 이 함수가 종료되고, 게임이 실제로 시작됩니다.

5.
1,2)
수정 전:
```
def runGame():
```

수정 후:
```
def runGame():
    # 기존 코드 생략
    # 게임 시작 시간 초기화
    startTime = time.time()
    while True: # game loop
        # 기존 코드 생략
        # 현재 시간 계산
        currentTime = time.time()
        # 게임 경과 시간 계산 (초 단위)
        elapsedTime = int(currentTime - startTime)
        # 기존 코드 생략
```

3)
위 코드에서는 다음과 같은 함수 실행 순서와 조건이 있습니다:
1.runGame() 함수가 호출되면 게임 루프가 시작됩니다.
2.startTime = time.time()를 사용하여 새 게임 시작 시간을 초기화합니다.
3.게임 루프 내에서 기존 게임 로직이 실행됩니다.
4.게임 루프 내에서 현재 시간(currentTime)을 가져와서 경과 시간(elapsedTime)을 계산합니다.
5.elapsedTime 변수에는 현재 시간과 시작 시간 간의 차이가 초 단위로 저장됩니다.
6.게임 경과 시간을 화면에 표시하는 코드가 있습니다. 이 부분은 코드에 직접적으로 포함되어 있지는 않지만, 경과 시간을 표시하는 기능이 있다고 가정합니다.
게임 루프가 반복됩니다.

6.
변경 전
```
#                R     G     B
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)

BORDERCOLOR = BLUE
BGCOLOR = BLACK
TEXTCOLOR = WHITE
TEXTSHADOWCOLOR = GRAY
COLORS      = (      BLUE,      GREEN,      RED,      YELLOW)
LIGHTCOLORS = (LIGHTBLUE, LIGHTGREEN, LIGHTRED, LIGHTYELLOW)
assert len(COLORS) == len(LIGHTCOLORS) # each color must have light color
```

변경 후(+함수 역할)

3)
이제 수정한 후의 코드에서 함수 호출 순서와 호출 조건에 대해 설명하겠습니다.

1. **runGame() 함수 호출**:
   - `main()` 함수에서 먼저 호출됩니다.
   - 이 함수는 게임의 메인 루프를 실행합니다.

2. **checkForQuit() 함수 호출**:
   - 게임 루프에서 사용자의 종료나 일시 중지를 감지합니다.
   - 종료 또는 일시 중지 이벤트가 발생하면 해당 이벤트를 처리하고 게임을 종료하거나 일시 중지합니다.

3. **pygame.event.get() 호출**:
   - 사용자 입력 이벤트를 가져오는 함수입니다.
   - 사용자의 키보드 입력 및 종료 이벤트를 처리합니다.

4. **사용자 입력에 대한 이벤트 처리**:
   - 키보드 입력에 따라 게임 요소가 업데이트됩니다.
   - 키보드 입력 이벤트가 발생하면, 해당 이벤트에 맞게 게임 요소가 이동, 회전 등이 처리됩니다.

5. **게임 요소 업데이트**:
   - 사용자의 입력에 따라 게임 요소가 업데이트됩니다.
   - 블록의 이동, 회전, 블록이 떨어지는 동작이 여기에 포함됩니다.

6. **화면 갱신**:
   - 게임 요소가 업데이트되면, 그에 따라 화면도 업데이트됩니다.
   - 변경된 게임 상태를 화면에 렌더링하여 사용자에게 보여줍니다.

7. **FPS 설정 및 화면 업데이트**:
   - 게임 루프의 마지막 단계에서 FPS를 설정하고 화면을 업데이트합니다.
   - FPS는 게임의 속도를 제어하며, 게임이 일정한 속도로 실행되도록 합니다.

새로운 색상이 추가되어, 게임의 색상 팔레트가 확장되었습니다. 이로 인해 게임의 시각적 다양성이 증가하고, 블록들이 각각 고유의 색상을 가지게 됩니다.