오픈소스 과제

- 체크리스트 (과제)

1. 현재 테트리스 게임의 배경음악을 주어진 3개의 음악 중 1개가 재생되도록 수정 (V)
   음악 리스트를 지정 후 랜덤함수를 사용하여 세 곡 중 하나를 무작위로 설정. 그 후 로드 후 재생.
2. 상태창 이름을 학번_이름 으로 수정
   해당 부분 showTextScreen 부분을 수정하면 됨
3. 게임시작화면의 문구를 MY TETRIS으로 변경
   해당 부분 showTextScreen 부분을 수정하면 됨
4. 게임시작화면의 문구 및 배경색을 노란색으로 변경
   미리 각각의 RGB값에 따라 지정해 놓은 YELLOW색으로 TEXTCOLOR를 바꿔주면 됨
5. 게임 경과 시간을 초 단위로 표시 (새 게임 시작시 0으로 초기화 되어야 함)
6. 7개의 블록이 각각 고유의 색을 갖도록 코드를 수정하거나 추가
   처음에 든 생각 : 컬러도 딕셔너리를 지정하여(딕셔너리 이름 : COLORS), GetNewPiece함수에 'color'부분을
   COLORS[shape]로 수정하면 될 것이라고 생각함. 따라서 이미 지정된 PIECES딕셔너리와 연결시킬 것이었기 때문에 이와 같은 이름으로 딕셔너리를 지정하였음. 하지만 이렇게 코드를 추가하였음에도 불구 KeyError: (155, 0, 0) 와 같은 오류가 지속적으로 생김. 이는 코드에서 색상을 색상 이름 대신 RGB값으로 직접 사용하고 있었기 때문이었음.
   따라서 블럭을 그리는 함수인 drawBox함수의 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)) 부분의 COLORS[color], LIGHTCOLORS[color]부분을 'color'로 바꿔주면서 RGB값이 아닌 색상 이름을 직접 받아오도록 수정하여 이를 해결하였음.
7. 보고서 작성
   a. 1~6번의 과제를 코드의 어느 부분을 어떻게 수정했는지 설명
   b. 각 함수의 역할
   c. 함수의 호출 순서 및 호출 조건에 대한 설명

1~6번의 과제를 코드의 어느 부분을 어떻게 수정했는지 설명

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

```
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')
```

이 부분을

```
music_files = ['Platform_9.mp3', 'Hover.mp3', 'Our_Lives_Past
while True: # game loop
    selected_music = random.choice(music_files)
    pygame.mixer.music.load(selected_music)
    pygame.mixer.music.play(-1, 0.0)
```

이렇게 수정하였다. 다운로드한 음악의 이름들을 각각의 원소로 갖는 music_files란 이름의 리스트를 선언하고 random함수를 통해 랜덤한 음악을 선택하고 로드, 플레이를 통해 음악이 재생되게 수정하였다.

2. 상태창 이름을 학번_이름 으로 수정을 위해

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

이 부분을

```
pygame.display.set_caption('2021006453_KIMMINCHAE')
```

이렇게 수정함으로써 상단바에 학번_이름이 출력됨.

3. 게임 시작 화면의 문구를 MY TETRIS으로 변경은 기존

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

부분을

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

이렇게 수정하였다.

4. 게임시작화면의 문구 및 배경색을 노란색으로 변경을 위해

```
TEXTCOLOR = WHITE
TEXTSHADOWCOLOR = GRAY
#였던 기존의 텍스트 컬러를
TEXTCOLOR = LIGHTYELLOW
TEXTSHADOWCOLOR = YELLOW
#이렇게 수정함으로써 추후 TEXTCOLOR값이 인용될 때 노란색으로 바뀌도록 하였
```

5. 게임 경과 시간을 초 단위로 표시 (새 게임 시작시 0으로 초기화 되어야 함)를 위해서 먼저 함수를 선언, 작성해주었는데

```
def drawElapsedTime(elapsedTime):
    elapsedTimeSurf = BASICFONT.render('Play Time: %s' % int(
    #폰트를 통해 게임 경과 시간을 텍스트로 렌더링하는 코드
    elapsedTimeRect = elapsedTimeSurf.get_rect()
    #텍스트를 화면으로 가져오기 위한 코
    elapsedTimeRect.topright = (WINDOWWIDTH - 470, 20)
    #텍스트의 위치 설정
    DISPLAYSURF.blit(elapsedTimeSurf, elapsedTimeRect)
```

```
def runGame():
    #추가된 부분만 기술
    startTime = time.time()
    #time.time()은 프로그램이 실행된 시점(또는 특정 이벤트가 시작된 시점
    #초단위로 저장한다.
    while True:
        elapsedTime = time.time() - startTime
        #프로그램 시작 시점부터 현재까지의 경과 시간을 계산하는 코드.
        drawElapsedTime(elapsedTime)
```

```
        #함수를  호출.
```

이렇게 함수의 선언과 메인 실행 코드인 runGame함수 내의 코드 추가를 통해 Play Time을 출력 할 수 있었다.

6. “7개의 블록이 각각 고유의 색을 갖도록 코드를 수정하거나 추가”를 보고 처음엔 이미 블럭 모양이 딕셔너리 처리 되어 실행되고 있었기 때문에 이와 같이 컬러도 딕셔너리를 지정하여(딕셔너리 이름 : COLORS), 그 후 GetNewPiece함수에 'color'부분을 COLORS[shape]로 수정하면 될 것이라고 생각함.

   ```
   COLORS = {
       'S': GREEN,
       'Z': RED,
       'J': BLUE,
       'L': ORANGE,
       'I': CYAN,
       'O': YELLOW,
       'T': PURPLE
   }
   ```

   이미 지정된 PIECES 딕셔너리와 연결 시킬 것이었기 때문에 위 사진과 같이 딕셔너리를 지정하였음. 하지만 이렇게 코드를 추가하였음에도 불구 KeyError: (155, 0, 0) 와 같은 오류가 지속적으로 생김. 이는 코드에서 블럭 색상을 받아올 때 색상 이름 대신 RGB값으로 직접 사용하고 있었기 때문이었음.

   따라서 블럭을 그리는 함수인 drawBox함수의

   ```
   pygame.draw.rect(DISPLAYSURF, COLORS[color], (pixelx + 1,



   pygame.draw.rect(DISPLAYSURF, LIGHTCOLORS[color], (pixelx
                                                      pixely
                                                      BOXSIZE
                                                      BOXSIZE
   pygame.draw.rect(DISPLAYSURF, color, (pixelx + 1,
   ```

```
pygame.draw.rect(DISPLAYSURF, color, (pixelx + 1,
                                      pixely + 1,
                                      BOXSIZE - 4,
                                      BOXSIZE - 4))
```

COLORS[color], LIGHTCOLORS[color]해당 부분을

color로 수정하면서 RGB값이 아닌 색상 이름을 직접 받아오도록 수정하여 이를 해결하였음.

각 함수의 역할

▼ makeTextObjs

텍스트를 화면에 그리기 위한 함수.

```
def makeTextObjs(text, font, color):
    surf = font.render(text, True, color)
    return surf, surf.get_rect()
```

▼ getNewPiece

getNewPiece 함수는 새로운 테트리스 블록 조각을 생성하여 반환. 이 함수는 랜덤한 새로운 블록 조각의 모양과 각 모양 별 지정 색상을 반환함.

```
def getNewPiece():
    # 랜덤한 블록 모양과 지정 색상 반환
    shape = random.choice(list(PIECES.keys()))
    newPiece = {'shape': shape,
                'rotation': random.randint(0, len(PIECE
                'x': int(BOARDWIDTH / 2) - int(TEMPLATE
                'y': -2, # 보드 위에서 시작 (0보다 작은 값)
                'color': COLORS[shape]}
    return newPiece
```

▼ checkForkeyPress

이 함수는 키보드 입력을 감지하여 처리하는 함수.

KEYDOWN 이벤트인 경우, 루프의 다음 반복으로 건너뛰며, KEYUP일 경우에 그 키 코드를 리턴한다. 즉 키를 눌렀다가(KEYDOWN) 뗐을 때(KEYUP) 해당 키 코드를 반환한다.

```
def checkForKeyPress():
    # Go through event queue looking for a KEYUP event.
    # Grab KEYDOWN events to remove them from the event
    checkForQuit()

    for event in pygame.event.get([KEYDOWN, KEYUP]):
        if event.type == KEYDOWN:
            continue
        return event.key
    return None
```

각 함수의 호출 순서 및 호출 조건에 대한 설명

게임 시작 시 변수들과 함수들을 초기화 하는 코드

1. board = getBlankBoard() : 빈 보드를 초기화한다.
2. lastMoveDownTime = time.time(), lastMoveSidewaysTime = time.time(), lastFallTime = time.time() : 각 움직임의 시간을 초기화.
3. movingDown = False, movingLeft = False, movingRight = False : 움직임 상태 변수를 초기화.
4. score = 0 : 점수를 초기화.
5. level, fallFreq = calculateLevelAndFallFreq(score) : 현재 점수에 기반하여 레벨과 낙하 빈도를 계산.
6. startTime = time.time() : 게임 시작 시간을 기록함.
7. fallingPiece = getNewPiece(), nextPiece = getNewPiece() : 새로운 조각을 생성.

while문을 통한 반복 (game loop)

1. elapsedTime = time.time() - startTime : 경과 시간을 계산.
2. if fallingPiece == None : 현재 떨어지는 조각이 없을 경우를 조건으로 설정
    - fallingPiece = nextPiece : 다음 조각을 현재 조각으로 정의함.

- nextPiece = getNewPiece() : 새로운 다음 조각을 생성.
- lastFallTime = time.time() : 마지막 낙하 시간을 초기화.
- if not isValidPosition(board, fallingPiece) : 새 조각이 유효한 위치에 배치될 수 없는 경우 게임 오버로 종료함.

이벤트 처리 (키 입력)

for event in pygame.event.get() :

- if event.type == KEYUP : 키를 뗀 경우
  - if event.key == K_p : 'P' 키로 게임을 일시 중지
  - 방향키(K_LEFT, K_a, K_RIGHT, K_d, K_DOWN, K_s)에 따라 이동 상태 변수를 False로 설정
- if event.type == KEYDOWN : 키를 누른 경우
  - 방향키(K_LEFT, K_a, K_RIGHT, K_d)에 따라 조각을 좌우로 이동시키고, 이동 상태 변수를 True로 설정
  - if event.key == K_UP or event.key == K_w : 위쪽 화살표나 'W' 키로 조각을 회전.
  - if event.key == K_q : 'Q' 키로 조각을 반대 방향으로 회전
  - if event.key == K_DOWN or event.key == K_s : 아래쪽 화살표나 'S' 키로 조각을 빠르게 떨어뜨림.
  - if event.key == K_SPACE : 스페이스바로 조각을 한 번에 바닥까지 떨어뜨림

사용자 입력에 따른 조각 이동

1. if (movingLeft or movingRight) and time.time() - lastMoveSidewaysTime > MOVESIDEWAYSFREQ : 일정 시간이 지나면 좌우로 조각을 이동
   - if movingLeft and isValidPosition(board, fallingPiece, adjX=-1) : 왼쪽 이동이 가능한지 확인하고 이동
   - if movingRight and isValidPosition(board, fallingPiece, adjX=1) : 오른쪽 이동이 가능한지 확인하고 이동

2. if movingDown and time.time() - lastMoveDownTime > MOVEDOWNFREQ and isValidPosition(board, fallingPiece, adjY=1) : 일정

시간이 지나면 조각을 아래로 이동시킴

조각 자동 낙하 매커니즘에 대한 설명

1. if time.time() - lastFallTime > fallFreq : 일정 시간이 지나면 조각을 자동으로 떨어뜨림
    - if not isValidPosition(board, fallingPiece, adjY=1) : 조각이 더 이상 떨어질 수 없는 경우
        - addToBoard(board, fallingPiece) : 보드에 조각을 추가
        - score += removeCompleteLines(board) : 완성된 줄을 제거하고 점수를 업데이트
        - level, fallFreq = calculateLevelAndFallFreq(score) : 새로운 점수에 기반하여 레벨과 낙하 빈도를 다시 계산함
        - fallingPiece = None : 현재 떨어지는 조각을 제거
    - else : 조각을 한 칸 아래로 이동

화면을 그리는 함수에 대한 간단 설명

1. DISPLAYSURF.fill(BGCOLOR) : 화면을 배경색으로 채움
2. drawBoard(board) : 보드를 그림
3. drawStatus(score, level) : 점수와 레벨을 표시
4. drawNextPiece(nextPiece) : 다음 조각을 그린다
5. if fallingPiece != None : 현재 떨어지는 조각을 그린다
6. drawElapsedTime(elapsedTime) : 경과 시간을 표시
7. pygame.display.update() : 화면을 갱신
8. FPSCLOCK.tick(FPS) : 프레임 속도를 유지