Pygame을 이용한 Tetromino 수정하기

학번 : 2021093363
학과 : 컬쳐테크놀로지전공
이름 : 이경환

1.1. 체크리스트 수정사항

다음의 여섯 가지의 수정사항과 그것들의 수정 방법을 작성한다. 수정한 파이썬 파일은 다음 깃허브 링크를 통하여 확인할 수 있다.

https://github.com/lgh0005/osw

```
157  def main():
158      global FPSCLOCK, DISPLAYSURF, BASICFONT, BIGFONT
159      pygame.init()
160      FPSCLOCK = pygame.time.Clock()
161      DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
162      DISPLAYSURF.fill(BACKGROUND)
163      BASICFONT = pygame.font.Font('freesansbold.ttf', 18)
164      BIGFONT = pygame.font.Font('freesansbold.ttf', 100)
165      pygame.display.set_caption('2021093363 Lee Kyeong Hwan')
166
167      showTextScreen('MY TETRIS')
168      music_list = ['Hover.mp3', 'Our_Lives_Past.mp3', 'Platform_9.mp3']
169      while True: # game loop
170          pygame.mixer.music.load(random.choice(music_list))
171          pygame.mixer.music.play(-1, 0.0)
172          runGame()
173          pygame.mixer.music.stop()
174          showTextScreen('Over :(')
```

그림 1 | 배경음, 상태창이름, 시작화면문구 및 배경색을 수정한 main 함수

1.1.1. 배경음악 재생

main 함수의 while 반복문에서 작성된 음악을 로드하는 pygame.mixer.music.load 함수에서 받는 인자를 리스트인 music_list에 3개의 노래 이름을 작성하였다. 그리고 그 리스트 내에 한 개를 무작위로 뽑게 하여 세 노래 중 하나의 노래를 선택하여 재생하도록 하였다. 그림 1의 168, 170번째 줄에서 확인할 수 있다.

1.1.2. 상태창 이름 변경

main 함수 내에 상태창 이름을 정하는 pygame.display.set_caption 함수에 인자로

본인의 학번과 이름을 문자열로 수정하였다. 그림 1의 165번째 줄에서 확인할 수 있다.

1.1.3. 게임 시작 화면 문구 및 배경색 변경

main 함수 내에 호출된 showTextScreen 함수는 문자열을 인자로 받아 그 문자열을 화면에 띄우는 함수이다. 이 문자열을 'MY TETRIS'로 수정하여 화면 문구를 수정사항에 맞게 변경하였다. 이는 그림 1의 167번째 줄에서 확인할 수 있다. 그리고 배경색을 나타내는 튜플 전역변수 BACKGROUND를 작성하고 Surface의 색을 채우는 fill 함수를 이용하여 배경색을 노란색으로 변경하였다. 이는 그림 1의 162번째 줄에서 확인할 수 있다.

```
177  def runGame():
178      global TIMER
179      # setup variables for the start of the game
180      board = getBlankBoard()
181      lastMoveDownTime = time.time()
182      lastMoveSidewaysTime = time.time()
183      lastFallTime = time.time()
184      movingDown = False # note: there is no movingUp variable
185      movingLeft = False
186      movingRight = False
187      score = 0
188      level, fallFreq = calculateLevelAndFallFreq(score)
189
190      fallingPiece = getNewPiece()
191      nextPiece = getNewPiece()
192
193      while True: # game loop
194          TIMER += 1
195          if fallingPiece == None:
196              # No falling piece in play, so start a new piece at the top
197              fallingPiece = nextPiece
198              nextPiece = getNewPiece()
199              lastFallTime = time.time() # reset lastFallTime
200
201              if not isValidPosition(board, fallingPiece):
202                  TIMER = 0
203                  return # can't fit a new piece on the board, so game over
```

그림 2 | 게임 경과 시간을 추가하여 수정한 runGame 함수

1.1.4. 게임 경과 시간 초 단위 표시

게임 경과 시간을 표시하기 위하여 먼저 전역변수 TIMER를 0으로 초기화하여 선언한다. 그리고 시간을 얻기 위하여 게임을 실행하는 함수인 runGame 함수 내에 게임이 시작되기 위한 while 반복문 내에 TIMER가 1씩 계속 증가하도록 한다. 그림 2에 178번째 줄과 194번째 줄에서 이를 확인할 수 있다. 추가로 게임이 종료되는 조건에 TIMER를 0으로 설정함으로써 게임을 다시 시작할 때 게임 경과 시간이 0이 되도록 한다. 그림 2의 202번째 줄에서 이를 확인할 수 있다.

```
485  def drawStatus(score, level, myTime):
486      # draw the score text
487      scoreSurf = BASICFONT.render('Score: %s' % score, True, TEXTCOLOR)
488      scoreRect = scoreSurf.get_rect()
489      scoreRect.topleft = (WINDOWWIDTH - 150, 20)
490      DISPLAYSURF.blit(scoreSurf, scoreRect)
491
492      # draw the level text
493      levelSurf = BASICFONT.render('Level: %s' % level, True, TEXTCOLOR)
494      levelRect = levelSurf.get_rect()
495      levelRect.topleft = (WINDOWWIDTH - 150, 50)
496      DISPLAYSURF.blit(levelSurf, levelRect)
497
498      # draw the time text
499      timeSurf = BASICFONT.render('Play Time: %s' % myTime, True, TEXTCOLOR)
500      timeRect = timeSurf.get_rect()
501      timeRect.topleft = (WINDOWWIDTH - 600, 20)
502      DISPLAYSURF.blit(timeSurf, timeRect)
```

그림 3 | 게임 경과 시간을 표시할 인자를 추가한 drawStatus 함수

그리고 상태표시를 하는 drawStatus 함수에 이 TIMER 인자를 받을 수 있도록 매개변수를 하나 추가하고 이 시간을 표시할 수 있도록 drawStatus 함수를 그림 3과 같이 수정한다. 그림 3의 498번째 줄부터 502번째 줄처럼 코드를 추가한다.

이후 초당 프레임을 나타내는 변수 FPS를 이용하여 계속 증가하는 TIMER에 FPS를 나누고 이를 정수로 캐스팅하면 게임 경과 시간을 얻을 수 있다. 따라서 runGame 함수를 하단의 그림 4와 같이 294번째 줄을 다음과 같이 수정한다.

```
291         # drawing everything on the screen
292         DISPLAYSURF.fill(BGCOLOR)
293         drawBoard(board)
294         drawStatus(score, level, int(TIMER/FPS))
295         drawNextPiece(nextPiece)
296         if fallingPiece != None:
297             drawPiece(fallingPiece)
298 
299         pygame.display.update()
300         FPSCLOCK.tick(FPS)
```

그림 4 | 수정된 drawStatus 함수를 호출하는 runGame 함수

1.1.5. 블록 고유의 색 가지기

```
24  #               R    G    B
25  WHITE       = (255, 255, 255)
26  GRAY        = (185, 185, 185)
27  BLACK       = (  0,   0,   0)
28  RED         = (255,   0,   0)
29  GREEN       = (  0, 255,   0)
30  BLUE        = (  0,   0, 255)
31  YELLOW      = (155, 155,   0)
32  LIGHTYELLOW = (85, 85, 0)
33  BACKGROUND = (55, 55, 0)
34  MAGENTA     = (255, 0, 255)
35  AQUA        = (0, 255, 255)
36 
37  BORDERCOLOR = BLUE
38  BGCOLOR = BLACK
39  TEXTCOLOR = YELLOW
40  TEXTSHADOWCOLOR = LIGHTYELLOW
41  COLORS      = (BLUE, GREEN, RED, YELLOW, WHITE, MAGENTA, AQUA)
```

그림 5 | 색을 나타내는 전역변수 및 색을 나열한 튜플 변수

블록의 고유색을 튜플(COLORS)로 작성한다. 일곱 가지의 색은 파랑(BLUE), 초록(GREEN), 빨강(RED), 노랑(YELLOW), 하양(WHITE), 자주색(MAGENTA), 하늘색(AQUA)이며 이들은 일곱 가지의 블록에 대하여 고유색을 나타낸다. (그림 5)

이어서 무작위 블록을 불러오는 getNewPiece 함수를 수정한다. 색을 지정할 지역

변수 myColor를 설정하고, 일곱 가지의 블록 각각에 대하여 고유색을 지정한다. 이후 조각의 정보를 저장하는 딕셔너리 자료 newPiece의 키 중 'color'의 값을 myColor로 설정함으로써 블록 고유색을 지정한다. 하단의 그림 6의 367번째 줄에서 380번째 줄은 고유색 지정 조건문을 나타내며 387번째 줄은 딕셔너리 키인 'color'의 값을 myColor로 설정한 것이다. 즉, COLORS 튜플 내 원소들 순서대로 각각 'S자 블록', 'Z자 블록', 'J자 블록', 'L자 블록', 'I자 블록', 'O자 블록', 'T자 블록'의 고유색을 지정한다.

```
364  def getNewPiece():
365      # return a random new piece in a random rotation and color
366      shape = random.choice(list(PIECES.keys()))
367      myColor = ''
368      if shape == 'S':
369          myColor = COLORS[0]
370      if shape == 'Z':
371          myColor = COLORS[1]
372      if shape == 'J':
373          myColor = COLORS[2]
374      if shape == 'L':
375          myColor = COLORS[3]
376      if shape == 'I':
377          myColor = COLORS[4]
378      if shape == 'O':
379          myColor = COLORS[5]
380      if shape == 'T':
381          myColor = COLORS[6]
382
383      newPiece = {'shape': shape,
384                  'rotation': random.randint(0, len(PIECES[shape]) - 1),
385                  'x': int(BOARDWIDTH / 2) - int(TEMPLATEWIDTH / 2),
386                  'y': -2, # start it above the board (i.e. less than 0)
387                  'color': myColor}
388      return newPiece
```

그림 6 | 각 블록이 고유의 색을 가지도록 수정한 getNewPiece 함수

2.1. 함수의 역할

이 파일 안에 정의되어있는 함수는 총 21개로 이들 중 게임 실행에 직접 관여하는 함수와 화면에 이미지를 띄우는 함수를 구별하여 간략히 설명한다. 추가로 Pygame에 있는 여러 함수 중 핵심적인 것들을 설명한다.

2.1.1. 게임 실행 함수

main() : 기본적인 게임의 화면인 DISPLAYSURF, 설정된 FPS만큼 화면을 업데이트

할 FPSCLOCK, 문장을 화면에 띄울 글자 폰트들을 지정한다. 무한 반복문 내에서 이들은 화면을 계속하여 업데이트하여 게임 내의 객체들을 화면에 띄워 움직일 수 있도록 한다.

runGame() : 무한 반복문 내에 게임 내용에 해당하는 것들을 모두 업데이트한다. 그리고 event를 받는 반복문을 통해 키보드의 input을 받는다. 그 외 게임 내용에 해당하는 여러 조건을 계속하여 판정한다.

terminate(), checkforQuit(), checkForKeyPress() : 사용자의 input을 통한 event를 처리하는 함수로 사용자가 키보드를 누르거나 사용자가 게임을 끄려고 할 때 이를 처리하는 함수이다.

그 외 게임에서 등장하는 블록들을 움직이거나 회전, 특정 조건에서 블록이 사라지거나 생성되는 것에 관여하는 함수들은 다음과 같으며 역할을 간단히 서술하면 다음과 같다.

calculateLevelAndFallFreq(score) : 점수에 비례하여 블록이 떨어지는 주기를 빠르게 하는 역할을 한다.
getNewPiece() : 다음 내려올 새 블록을 무작위로 불러오며 블록의 정보를 딕셔너리 자료로 반환한다.
addToBoard(board, piece) : 보드의 격자에 조건에 맞는 칸에 색을 칠함으로써 블록이 움직이게 보이도록 한다.
getBlankBoard(), isOnBoard(x, y) : 비어있는 보드를 생성하며 보드 위에 있는지 판별한다.
isValidPosition(board, piece, adjX=0, adjY=0) : 보드 내에 있으며 다른 블록이랑 닿았는지 판별한다.
isCompleteLine(board, y) : 한 줄이 완성되었는지 판별한다.
removeCompleteLines(board) : 완성된 한 줄을 보드에서 삭제한다.
convertToPixelCoords(boxx, boxy) : 보드의 격자에 맞는 위치를 반환한다.

2.1.2. 이미지를 띄우는 함수

이 함수들은 공통적으로 이미지를 불러오고 그 이미지에 해당하는 위치를 처리하기 위하여 rect를 얻는다. 그 후 게임 윈도우 위에 이미지를 띄우며 그 이미지의 이동은

기본적으로 rect의 좌상향점을 기준으로 x 좌표와 y 좌표를 통해 움직인다.

이 소스코드에서 사용된 이미지를 띄우는 함수들은 다음과 같으며 역할을 간단히 서술하면 다음과 같다.

makeTextObjs(text, font, color) : 텍스트를 입력받고 font에 해당하는 글꼴, color에 해당하는 색을 이미지로 얻고 이에 대한 rect를 반환한다.
showTextScreen(text) : makeTextObjs 함수를 이용하여 제목, 제목에 대한 그림자, 사용자 인터페이스 등과 같은 텍스트 요소들을 화면에 띄운다.
drawBox(boxx, boxy, color, pixelx=None, pixely=None) : 다음 내려올 블록을 표시할 박스를 그린다. 블록을 표시하기 위하여 격자로 나누고 그 격자에 해당 블록에 맞는 칸에 색을 칠한다.
drawBoard(board) : 블록이 화면에 보일 보드를 그린다.
drawStatus(score, level, myTime) : 사용자 인터페이스 중 점수, 레벨, 게임 경과 시간을 표시하는 텍스트를 그린다.
drawPiece(piece, pixelx=None, pixely=None) : 블록을 보드 위에 그린다.
drawNextPiece(piece) : 다음 내려올 블록을 그린다.

2.1.3. Pygame 모듈과 그 내부 함수

Pygame은 2D 또는 PyOpenGL 모듈을 함께 사용하여 3D 플랫폼 콘텐츠 또는 게임을 제작하는데 사용되는 GNU LGPL Version 2.1 라이선스 아래에 Python 언어로 작성된 모듈이다. 게임을 제작할 때 필요한 작용을 하는 함수를 작성하면 다음과 같다.

pygame.init() : Pygame 모듈을 초기화하는 함수이다.
screen.blit(Surface, (x, y)) : Pygame으로 생성한 윈도우 위에 이미지를 좌상향점을 기준으로 x 좌표, y 좌표에 해당 이미지를 띄운다.
pygame.quit() : Pygame 모듈을 종료한다. Pygame으로 생성한 윈도우가 종료된다.
pygame.time.Clock() : Pygame에서 게임의 프레임을 관여하는 함수이다. 이와 함께 tick 함수를 사용하여 게임의 화면 업데이트를 할 수 있다.
pygame.event.get() : 사용자의 Input을 받거나 다른 특정 조건들을 큐에서 얻는다. 키보드나 마우스, 특정 시간 등이 event가 될 수 있다.
pygame.draw : Pygame에서 윈도우 위에 도형을 그릴 수 있다. 사각형, 다각형, 원, 타원, 직선 등을 그릴 수 있다.

pygame.mixer.music : 음악을 재생하거나 정지, 로드 등을 할 수 있는 소리와 관련된 클래스이다.
pygame.sprite : 게임 내에 등장하는 모든 객체의 특성과 행동들을 위한 클래스이다. 객체들 간의 충돌, 같은 객체들을 처리하는 그룹, 객체의 행동 등 게임에서 등장하는 요소들의 특성들을 나타내게 해준다.
pygame.math : 특히 벡터값을 얻는데 주로 쓰이는 클래스이다. 이를 통해 게임에서 등장하는 요소들을 움직임을 구현할 수 있다.

3.1. 함수의 호출 순서 및 호출 조건

조건문 'if __name__ == "__main__"'과 해당 파일에서 작성자가 정의한 함수들의 전반적인 함수들의 호출 되는 순서 및 그것들이 호출되는 조건을 설명한다.

3.1.1. if __name__ == '__main__' 조건문

해당 프로그램 내부의 이름(__name__)이 '__main__'인 경우 즉 실행하려는 파이썬 파일이 모듈이 아님을 나타내는 조건문이다. 이 조건문이 작성된 파일을 직접 실행한 경우 __name__에 __main__이라는 값이 입력되어 이 조건문에 대한 실행으로 main 함수가 호출되는데 이 파일 내에서 다른 파이썬 파일을 임포트한 경우 그것들의 실행을 방지한다.

3.1.2. 파일 내 함수들의 호출 순서 및 조건

해당 파일 실행 시 3.1.1.의 내용에 의하여 먼저 main 함수가 실행된다. main 함수 내에서 가장 먼저 showTextScreen 함수를 호출하여 화면에 'MY TETRIS' 문자열과 'Press any key to play! pause key is p' 문자열을 화면에 띄운다. 이때 while 무한 반복문 내에 작성된 runGame 함수를 호출한다.

runGame 함수 내에서 무한 반복문 안에 특정 event를 받아오는 for 반복문이 있다. 사용자가 키보드로 p키를 누를 경우, 게임이 일시 정지되며 이를 알리는 문구를 showTextScreen('Get a rest!') 함수를 통해 화면에 띄운다. 그리고 checkForQuit 함수를 통해 사용자로부터 게임을 종료할 경우 checkForQuit 안에 terminate 함수로 인하여 게임이 종료된다. 그 외 getNewPiece 함수로 지속적으로 다음 내려올 블록을 생성하며, isValidPosition 함수로 게임이 끝나는 조건 및 블록의 회전과 이동을 판정한다. 그리고 drawBoard, drawStatus, drawNextPiece 함수들은 화면에 사용자가 알아야 할 정보 및 플레이 화면을 화면에 그린다. 즉 runGame이 호출되면 이 함수 내의

while 반복문 내에서 isValidPositin 함수를 통해 지속적으로 게임을 판정하며 drawBoard, drawStatus, drawNextPiece 함수로 화면에 실시간으로 업데이트 되는 정보들을 표시하는 순서를 띤다.

그 외 이 파일의 게임적 요소에 대한 함수들의 호출을 살펴보면 블록이 떨어진 시간 간격이 블록 주기보다 길 경우에 대한 판정으로부터 시작된다. 이 조건이 충족되는 경우 블록이 놓여있는지 판별한다. 만일 놓인 블록으로 인하여 한 줄이 완성된 경우 removeCompleteLines 함수를 통해 그 줄을 삭제시키며, 레벨과 떨어지는 주기를 calculateLevelAndFallFreq(score) 함수를 통해 재지정한다. 반대로 블록이 놓이지 않은 상태이면 블록의 y 값을 1씩 증가시키고 마지막으로 놓인 시간을 계속 측정한다. 그리고 a, s, d 키 또는 좌, 하 ,우 키를 통해 블록을 내리거나 회전시킬 수 있다. 이들의 작동 조건과 전반적인 함수들의 호출 순서를 정리하면 하단의 그림 7과 같다.

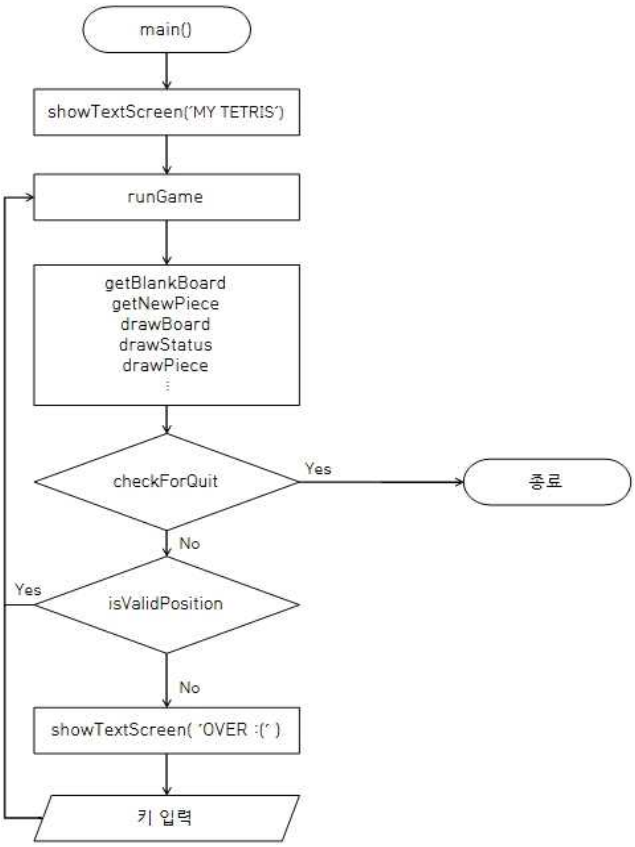

그림 7 | Tetromino 게임 파일의 함수 호출 순서도

4.1. 참고 문헌

[1] Pygame Document 웹페이지 https://www.pygame.org/docs/
[2] Pygame GitHub 웹페이지 https://github.com/pygame
[3] “파이썬으로 배우는 게임 개발 입문편”, “히로세 츠요시”, “제 1판”, Jpup, p.234
[4] “ゲーム開發ではじめるPython3”, “大西 武”, “工學社”, p.32 ~ 33