2019 학년도 2 학기

오픈소스소프트웨어프로젝트

프로젝트 분석 보고서

< 파이썬을 활용한 테트리스 게임 구현 >

열심히만들조 MDJ (1 조)

2013111550 경영학전공 최석진

2017110177 영어통번역학전공 김연진

2017110175 영어통번역학전공 최성연

목차

I. 프로젝트명

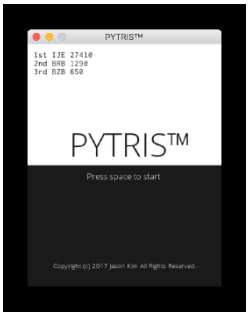
본 프로젝트는 파이썬을 활용한 테트리스 게임 구현으로, 프로젝트 명은 'PYTRIS'이다.

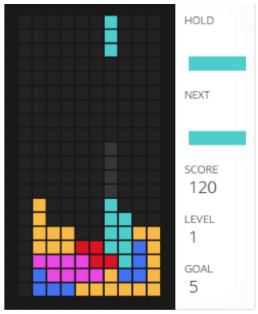
II. 프로젝트 URL

https://github.com/k0626089/PYTRIS

III. 프로젝트 설명

파이썬 모듈 중 하나인 pygame을 활용해 제작된 테트리스 게임 'PYTRIS'이며, mino.py와 pytris.py로 구성되어 있다.





mino.py는 테트리스의 블록 모델을, pytris.py는 main과 각종 함수를 내장한다. 또한 이전 버전인 mino.old.py와 pytris.old.py가 함께 들어있다. 게임에 사용된 효과음, 이미지, 폰트 등은 assets에 담겨있다.

사용된 라이선스는 MIT License이며, 이는 복제, 수정, 배포의 권한이 허용된 라이선스이다.

해당 프로젝트는 누구나 손쉽게 테트리스를 개발하고 즐길 수 있도록 하기 위한 오픈소스 소프트웨어 프로젝트이다. 기존 코드는 pygame을 활용하여 알아보기 쉽고 간결한 코드로 제작되었다는 장점이 있으나 해당 코드가 지닌 단점을 보완하고, 디자인적 측면을 강화하여 시각적 효과를 갖춘 테트리스를 만들 예정이다. 또한 개발 과정에서 github를 활용하여 효율적인 팀워크와 개발 능력을 향상시킬 것이다.

IV. 프로젝트 주요 기능

- 누구나 쉽고 편리하게 테트리스를 즐기도록 한다.
- 코드를 간략화하여 다른 사람이 알아보기 쉽도록 코드의 대중성을 높인다.

V. 소스코드 라인 수

아래 사진에서 볼 수 있듯이, 해당 프로젝트 소스코드의 총 라인 수는 1286줄이다.

```
kimyeonjin@kimyeonjin-VirtualBox:~/Desktop/ossp$ cloc PYTRIS/
      12 text files.
      12 unique files.
      45 files ignored.

http://cloc.sourceforge.net v 1.60  T=0.22 s (23.1 files/s, 7268.3 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Python                           5            176            108           1286
-------------------------------------------------------------------------------
SUM:                             5            176            108           1286
-------------------------------------------------------------------------------
```

VI. 소스코드 분석

1) 모듈

모듈	기능
pygame	비디오 게임 작성용으로 설계된 크로스 플랫폼 Python 모듈(license: GNU GPL)
operator	연산 시 사용되며, 본 프로젝트에서는 리스트 정렬기준을 위해 사용
random	테트리스 모형(tetrimino) 랜덤으로 사용
from pygame.locals import*	pygame 내부 모듈 사용

2) 변수

- matrix = [[0 for y in range(height + 1)] for x in range(width)] : 테트리스를 그리는 보드
- width, height : 보드의 크기 결정
- framerate : 블록이 떨어지는 속도를 결정
- done : 메인 루프를 돌리는 변수(초깃값:False)
- pause : 멈춤을 확인하는 변수 (초깃값:False)

- start : 게임 시작을 확인하는 변수 (초깃값:False)
- mino : 테트리스 블록 모양을 결정, 랜덤으로 숫자 할당
- dx = 3, dy =0 : 보드에서의 초기 위치

3) Main 함수

1. loop 내 총 4가지 스크린이 구현되었다.

- ✓ Pause screen 게임 일시 정지 화면
- ✓ Game screen 게임 실행 화면
- ✓ Game over screen 게임 오버 화면
- ✓ Start screen 프로그램 시작 화면

2. Pause screen

line 318~342

```
if pause:
    for event in pygame.event.get():
        if event.type == QUIT:
            done = True
        elif event.type == USEREVENT:
            pygame.time.set_timer(pygame.USEREVENT, 300)
            draw_board(next_mino, hold_mino, score, level, goal)

        elif event.type == KEYDOWN:
            erase_mino(dx, dy, mino, rotation)
            if event.key == K_ESCAPE:
                pause = False
```

for event in pygame.event.get()이란 사용자가 발생시킨 이벤트를 가져와 for문을 수행하는 기능이며 반복문 안의 조건문에 따라 게임이 진행된다.

QUIT를 선택(창 닫기)한 경우: done을 True로 반환해 루프를 나와 pygame실행을 중단한다.

USEREVENT(start에서 esc 누름)의 경우: draw_board()로 현재블록과 hold 블록을 그려낸 후 PAUSED라는 글을 보인다.

KEYDOWN(다시 esc 누름)의 경우: erase_mino()로 블록을 지운다. Esc를 누른 경우 pause에 False를 할당한다.

3. Game screen

line 344~356

```
344     elif start:
345         for event in pygame.event.get():
346             if event.type == QUIT:
347                 done = True

348             elif event.type == USEREVENT:
349                 # Set speed
350                 if not game_over:
351                     keys_pressed = pygame.key.get_pressed()
352                     if keys_pressed[K_DOWN]:
353                         pygame.time.set_timer(pygame.USEREVENT, framerate * 1)
354                     else:
355                         pygame.time.set_timer(pygame.USEREVENT, framerate * 10)
356
```

QUIT를 선택(창 닫기): done을 True로 반환해 루프를 나와 pygame실행을 중단한다.

USEREVENT(game_over가 아닌 경우)를 통해 게임 속도를 조정한다. Down버튼 press 여부를 통해 속도가 조정된다.

line 357~368

```
357                 # Draw a mino
358                 draw_mino(dx, dy, mino, rotation)
359                 draw_board(next_mino, hold_mino, score, level, goal)
360
361                 # Erase a mino
362                 if not game_over:
363                     erase_mino(dx, dy, mino, rotation)
364
365                 # Move mino down
366                 if not is_bottom(dx, dy, mino, rotation):
367                     dy += 1
368
```

draw_mino()와 draw_board()로 블록과 게임 보드를 구현, game_over가 아닌 경우 erase_mino()로 블록 지우기를 실행한다. is_bottom()이 아닌 경우 dy의 값을 1씩 증가시킨다.

line 369~389

```
369                 # Create new mino
370                 else:
371                     if hard_drop or bottom_count == 6:
372                         hard_drop = False
373                         bottom_count = 0
374                         score += 10 * level
375                         draw_mino(dx, dy, mino, rotation)
376                         draw_board(next_mino, hold_mino, score, level, goal)
377                         if is_stackable(next_mino):
378                             mino = next_mino
379                             next_mino = randint(1, 7)
380                             dx, dy = 3, 0
381                             rotation = 0
382                             hold = False
383                         else:
384                             start = False
385                             game_over = True
386                             pygame.time.set_timer(pygame.USEREVENT, 1)
387                     else:
388                         bottom_count += 1
389
```

새로운 블록, mino를 생성한다. hard_drop 혹은 bottom_count의 값이 6인 경우 hard_drop을 False로, bottom_count를 0으로 초기화한다. 또한 score 값을 해당 level의 10배하여 증가한다. 뒤이어 if_stackable()로 다음 블록을 쌓을 수 있는 지 판단하여 함수를 실행한다. bottom_count의 값이 6이 아니거나 hard_drop이 아닌 경우 bottom_count의 값을 1 증가시키고 아래 390~403라인의 제어문이 시행된다.

line 390~403

```
390                 # Erase line
391                 erase_count = 0
392                 for j in range(21):
393                     is_full = True
394                     for i in range(10):
395                         if matrix[i][j] == 0:
396                             is_full = False
397                     if is_full:
398                         erase_count += 1
399                         k = j
400                         while k > 0:
401                             for i in range(10):
402                                 matrix[i][k] = matrix[i][k - 1]
403                             k -= 1
```

위 398라인의 코드 실행 후, 한 행이 채워지면 해당 줄을 지우는 기능이 실행된다. earse_count에 0을 대입한 후 중첩된 반복문과 이중 리스트를 활용하여 다 채워진 줄의 여부를 판단, 있는 경우 erase_count를 1씩 증가시킨다. 다 채워진 줄을 지운 후 matrix를 한 칸 씩 내린다.

이 후 erase_count의 값에 따라 효과음이 실행되고, 각 값에 따른 추가 점수가 부여된다.

line 417~422

```
417                 # Increase level
418                 goal -= erase_count
419                 if goal < 1 and level < 15:
420                     level += 1
421                     goal += level * 5
422                     framerate = int(framerate * 0.8)
```

해당 레벨의 목표인 goal의 값은 erase_count의 증가값만큼 감소한다. 만약 goal의 값이 1미만, level의 값이 15미만이라면 level 1증가, goal은 (level * 5)만큼 증가된다. 블록의 속도 또한 조정된다.

line 424~437

```
424             elif event.type == KEYDOWN:
425                 erase_mino(dx, dy, mino, rotation)
426                 if event.key == K_ESCAPE:
427                     ui_variables.click_sound.play()
428                     pause = True
429                 # Hard drop
430                 elif event.key == K_SPACE:
431                     ui_variables.drop_sound.play()
432                     while not is_bottom(dx, dy, mino, rotation):
433                         dy += 1
434                     hard_drop = True
435                     pygame.time.set_timer(pygame.USEREVENT, 1)
436                     draw_mino(dx, dy, mino, rotation)
437                     draw_board(next_mino, hold_mino, score, level, goal)
```

ESC키를 누르고 뗄 경우 효과음과 함께 게임이 일시 정지된다. SPACE키를 누르고 뗄 경우 효과음과 함께 Hard drop이 실행되며, 이를 통해 블록이 빠르게 내려간다.

line 438~452

```
438                 # Hold
439                 elif event.key == K_LSHIFT or event.key == K_c:
440                     if hold == False:
441                         ui_variables.move_sound.play()
442                         if hold_mino == -1:
443                             hold_mino = mino
444                             mino = next_mino
445                             next_mino = randint(1, 7)
446                         else:
447                             hold_mino, mino = mino, hold_mino
448                         dx, dy = 3, 0
449                         rotation = 0
450                         hold = True
451                     draw_mino(dx, dy, mino, rotation)
452                     draw_board(next_mino, hold_mino, score, level, goal)
```

왼쪽 Shift키 혹은 c를 누르고 뗄 경우 hold가 실행된다. 이 때 현재 블록인 mino는 hold_mino에 저장되며 next_mino의 블록이 mino로, 새로운 랜덤 블록이 next_mino로 대입된다.

line 453~486

```
453                 # Turn right
454                 elif event.key == K_UP or event.key == K_x:
455                     if is_turnable_r(dx, dy, mino, rotation):
456                         ui_variables.move_sound.play()
457                         rotation += 1
458                     # Kick
459                     elif is_turnable_r(dx, dy - 1, mino, rotation):
460                         ui_variables.move_sound.play()
461                         dy -= 1
462                         rotation += 1
```

```
483                     if rotation == 4:
484                         rotation = 0
485                     draw_mino(dx, dy, mino, rotation)
486                     draw_board(next_mino, hold_mino, score, level, goal)
```

up키 혹은 x를 누르고 뗄 경우, is_turnable_r()을 통해 우회 가능 여부를 판단하여 파라미터에 따라 우측 회전을 실행하고 rotation의 값을 1씩 증가시킨다. 블록은 총 네 방향이므로 만일 rotation의 값이 4라면 rotation의 값을 0으로 초기화한다. 블록의 좌측 회전 또한 동일한 알고리즘이 적용된다.

line 520~526

```
520                 # Move left
521                 elif event.key == K_LEFT:
522                     if not is_leftedge(dx, dy, mino, rotation):
523                         ui_variables.move_sound.play()
524                         dx -= 1
525                     draw_mino(dx, dy, mino, rotation)
526                     draw_board(next_mino, hold_mino, score, level, goal)
```

왼쪽 키를 누른 경우 is_leftedge()를 통해 블록이 왼쪽 끝에 위치해 있는 지 여부를 판단, 그렇지 않다면 왼쪽으로 한 칸 씩 이동한다. 오른쪽 키를 누른 경우 동일한 알고리즘이 적용되어 블록이 오른쪽으로 한 칸 씩 이동하게 된다.

4. Game over screen

line 538~546 , 548~550

```
538     elif game_over:
539         for event in pygame.event.get():
540             if event.type == QUIT:
541                 done = True
542             elif event.type == USEREVENT:
543                 pygame.time.set_timer(pygame.USEREVENT, 300)
544                 over_text_1 = ui_variables.h2_b.render("GAME", 1, ui_variables.white)
545                 over_text_2 = ui_variables.h2_b.render("OVER", 1, ui_variables.white)
546                 over_start = ui_variables.h5.render("Press return to continue", 1, ui_variables.white)

548                  draw_board(next_mino, hold_mino, score, level, goal)
549                  screen.blit(over_text_1, (58, 75))
550                  screen.blit(over_text_2, (62, 105))
```

QUIT를 선택(창 닫기)하여 done을 True로 반환해 루프를 나와 pygame실행을 중단한다. USEREVENT를 통해 “GAME”, “OVER”를 각각 over_text_1,2로 선언하여 screen.blit을 활용, 게임오버 창에 해당 텍스트를 띄운다. 같은 방식으로 이니셜과 언더바를 출력한다.

line 564~566

```
564                 if blink:
565                     screen.blit(over_start, (32, 195))
566                     blink = False
```

blink를 활용한 조건문을 통해 게임오버창의 이니셜과 언더바가 깜빡인다.

line 577~583

```
577             elif event.type == KEYDOWN:
578                 if event.key == K_RETURN:
579                     ui_variables.click_sound.play()
580 
581                     outfile = open('leaderboard.txt','a')
582                     outfile.write(chr(name[0]) + chr(name[1]) + chr(name[2]) + ' ' + str(score) + '\n')
583                     outfile.close()
```

게임오버 화면에서 엔터키를 눌렀을 때 'leaderboard.txt'라는 외부 텍스트 파일에 유저가 입력한 이니셜이 작성된다.

line 613~618

```
613                 elif event.key == K_RIGHT:
614                     if name_location != 2:
615                         name_location += 1
616                     else:
617                         name_location = 0
618                     pygame.time.set_timer(pygame.USEREVENT, 1)
```

게임오버 화면에서 엔터키가 아닌 오른쪽방향키를 눌렀을 때 이니셜 커서가 오른쪽으로 한 칸 이동된다. 613라인의 K_RIGHT와 name_location을 수정하여 좌우 방향키를 불렀을 때 커서가 옆으로 가도록, 상하 방향키를 눌렀을 때 알파벳이 바뀐다.

5. Start screen

line 641~682

```
else:
    for event in pygame.event.get():
        if event.type == QUIT:
            done = True
        elif event.type == KEYDOWN:
            if event.key == K_SPACE:
                ui_variables.click_sound.play()
                start = True

            if not start:
                pygame.display.update()
                clock.tick(3)
```

QUIT를 선택(창 닫기): done을 True로 반환해 루프를 나와 pygame실행을 중단한다.

KEYDOWN(키를 누른 이벤트): 만약 누른 키가 space라면 start에 true를 할당한다.

이벤트가 아직 없는 경우(else의 일부): start의 초깃값으로 false가 들어가 있으므로 이벤트가 아직 없는 경우 제어문을 실행하게 된다.

4) 함수

1. 함수 공통 부분

항목	공통 내용
공통된 함수 매개변수	x에 dx를 y에 dy를 mino에 숫자, r에 방향을 대입한다.
'is_'로 시작하는 함수 (is_turnable_l, is_turnable_r 제외)	먼저 grid의 값이 0이 아닌 지 확인한다. 0이 아니라면 if 문을 실행하게 된다.

def draw_mino(x,y,mino,r) line 152~170

```
152  # Draw a tetrimino
153  def draw_mino(x, y, mino, r):
154      grid = tetrimino.mino_map[mino - 1][r]
155
156      tx, ty = x, y
157      while not is_bottom(tx, ty, mino, r):
158          ty += 1
159
160      # Draw ghost
161      for i in range(4):
162          for j in range(4):
163              if grid[i][j] != 0:
164                  matrix[tx + j][ty + i] = 8
165
166      # Draw mino
167      for i in range(4):
168          for j in range(4):
169              if grid[i][j] != 0:
170                  matrix[x + j][y + i] = grid[i][j]
```

기능) tetrimino 그리기

- Grid에 mino_map의 모양과 방향을 선택한 리스트를 넣는다. is_bottom()이 아닌 경우 ty의 값을 1씩 증가한다.
- ghost 그리기 : for문 내 matrix의 인덱스에 8을 대입한다.
-mino 그리기 : for문으로 matrix의 인덱스에 grid 인덱스 값을 대입한다.

def erase_mino(x,y,mino,r) line 173~187

```
def erase_mino(x, y, mino, r):
    grid = tetrimino.mino_map[mino - 1][r]

    # Erase ghost
    for j in range(21):
        for i in range(10):
            if matrix[i][j] == 8:
                matrix[i][j] = 0

    # Erase mino
    for i in range(4):
        for j in range(4):
            if grid[i][j] != 0:
                matrix[x + j][y + i] = 0
```

기능) tetrimino 지우기

- Grid에 mino_map의 모양과 방향을 선택한 리스트를 넣는다.
- ghost 지우기 : for문으로 draw_mino로 matrix에 8이 들어간 부분들을 모두 0으로 바꾼다.
-mino 지우기 : for문으로 현재 내려오고 있던 블록을 찾아 0으로 지운다

def is_bottom(x,y,mino,r) line 189~200

	기능) 블록이 바닥에 있으면 true를 반환하는 함수

```
def is_bottom(x, y, mino, r):
    grid = tetrimino.mino_map[mino - 1][r]

    for i in range(4):
        for j in range(4):
            if grid[i][j] != 0:
                if (y + i + 1) > 20:
                    return True
                elif matrix[x + j][y + i + 1] != 0 and matrix[x + j][y + i + 1] != 8:
                    return True

    return False
```

- 테트리스를 그려내고자 하는 위치가 전체 높이를 벗어났는지 확인한다. 벗어나지 않았다면 matrix로 찾고자 하는 행과 열이 0과 8이 아닌 수로 채워져 있는지 확인한다. 이에 해당하지 않으면 false를 반환한다.

def is_leftedge(x,y,mino,r) line 203~214

```
def is_leftedge(x, y, mino, r):
    grid = tetrimino.mino_map[mino - 1][r]

    for i in range(4):
        for j in range(4):
            if grid[i][j] != 0:
                if (x + j - 1) < 0:
                    return True
                elif matrix[x + j - 1][y + i] != 0:
                    return True

    return False
```

기능) 블록이 왼쪽 끝에 있으면 true를 반환하는 함수

- 테트리스를 그려내고자 하는 위치가 전체 너비에서 0(가장 왼쪽을 의미)보다 작은지를 확인한다. 작지 않다면 matrix로 찾고자 하는 행과 열이 0이 아닌 수로 채워져 있는지 확인한다. 이에 해당하지 않으면 false를 반환한다.

def is_rightedge(x,y,mino,r) line:217~228

```
def is_rightedge(x, y, mino, r):
    grid = tetrimino.mino_map[mino - 1][r]

    for i in range(4):
        for j in range(4):
            if grid[i][j] != 0:
                if (x + j + 1) > 9:
                    return True
                elif matrix[x + j + 1][y + i] != 0:
                    return True

    return False
```

기능) 블록이 오른쪽 끝에 있으면 true를 반환하는 함수

- 테트리스를 그려내고자 하는 위치가 전체 너비에서 9(가장 오른쪽을 의미)보다 큰지 확인한다. 크지 않다면 matrix로 찾고자 하는 행과 열이 0이 아닌 수로 채워져 있는지 확인한다. 이에 해당하지 않으면 false를 반환한다.

def is_turnable_r(x,y,mino,r) line 231~245

기능) 오른쪽으로 회전이 가능한 경우 true를 반환하는 함수

```
def is_turnable_r(x, y, mino, r):
    if r != 3:
        grid = tetrimino.mino_map[mino - 1][r + 1]
    else:
        grid = tetrimino.mino_map[mino - 1][0]

    for i in range(4):
        for j in range(4):
            if grid[i][j] != 0:
                if (x + j) < 0 or (x + j) > 9 or (y + i) < 0 or (y + i) > 20:
                    return False
                elif matrix[x + j][y + i] != 0:
                    return False

    return True
```

- 방향의 값이 리스트의 끝이 아니라면 grid를 오른쪽으로 회전하기 위해 r에 더하기 1을 한다. 끝이라면 grid에서 방향을 0으로 설정한다. 양쪽 너비와 위아래의 길이를 비교하여 보드의 크기를 벗어났다면 false를 반환한다. 벗어나지 않았다면 matrix로 찾고자 하는 행과 열이 0이 아닌 수로 채워져 있는지 확인한다. 이에 해당하지 않으면 true를 반환한다.

def is_turnable_l(x,y,mino,r) line 248~262

```
def is_turnable_l(x, y, mino, r):
    if r != 0:
        grid = tetrimino.mino_map[mino - 1][r - 1]
    else:
        grid = tetrimino.mino_map[mino - 1][3]

    for i in range(4):
        for j in range(4):
            if grid[i][j] != 0:
                if (x + j) < 0 or (x + j) > 9 or (y + i) < 0 or (y + i) > 20:
                    return False
                elif matrix[x + j][y + i] != 0:
                    return False

    return True
```

기능) 왼쪽으로 회전이 가능한 경우 true를 반환하는 함수

- 방향의 값이 0이 아니라면 grid를 왼쪽으로 회전하기 위해 r에 마이너스 1을 한다. 0이라면방향 리스트의 마지막인 3을 설정한다. 양쪽 너비와 위아래의 길이를 비교하여 보드의 크기를 벗어났다면 false를 반환한다. 벗어나지 않았다면 matrix로 찾고자 하는 행과 열이 0이 아닌 수로 채워져 있는지 확인한다. 이에 해당하지 않으면 true를 반환한다.

def is_stackable(mino) line 265~274

```
def is_stackable(mino):
    grid = tetrimino.mino_map[mino - 1][0]

    for i in range(4):
        for j in range(4):
            #print(grid[i][j], matrix[3 + j][i])
            if grid[i][j] != 0 and matrix[3 + j][i] != 0:
                return False

    return True
```

기능) tetrimino를 그릴 수 있으면 true를 반환함수

- 이중 for문을 활용해 grid의 값이 0인지 아닌지 확인한다. 그리고 matrix의 값이 0인지 아닌지 확인한다. matrix가 0이 아니라는 의미는 그 곳에 다른 값이 들어가 있다는 것이다. 즉 그 공간은 이미 채워져 있는 공간으로 테트리스를 그릴 수 없다. 만약 둘이 모두 0이 아니라면 False를 반환한다. 이에 해당하지 않으면 true를 반환한다.

def draw_block(x, y, color) line 68~79

```
68  def draw_block(x, y, color):
69      pygame.draw.rect(
70          screen,
71          color,
72          Rect(x, y, block_size, block_size)
73      )
74      pygame.draw.rect(
75          screen,
76          ui_variables.grey_1,
77          Rect(x, y, block_size, block_size),
78          1
79      )
```

기능) x와 y는 위치를 나타내고 color는 색을 나타낸다. 이후 코드에서 테트리스 블록과 배경의 색을 나타내는데 사용된다. 78번 라인에 1은 사각형의 선 크기를 나타낸다.

def draw_board(next, hold, score, level, goal) line 82~83

```
82  def draw_board(next, hold, score, level, goal):
83      screen.fill(ui_variables.grey_1)
```

기능) 게임화면 내에서 각각 (next - 다음에 나올 블록 / hold - 홀드한 블록 / score - 현재점수 / level - 현재 레벨 / goal - 목표 블록제거 횟수 및 목표까지 잔여 블록제거 횟수) 를 나타내기 위한 매개변수이다.

- screen.fill(ui_variables.grey_1)은 게임화면을 채색하는 pygame내 기능이다.

line 86~90

```
86      pygame.draw.rect(
87          screen,
88          ui_variables.white,
89          Rect(204, 0, 96, 374)
90      )
```

- 그 후 pygame.draw.rect()를 활용하여 양쪽 사이드 바를 그린다.

line 93~104

```
93      grid_n = tetrimino.mino_map[next - 1][0]
94
95      for i in range(4):
96          for j in range(4):
97              dx = 220 + block_size * j
98              dy = 140 + block_size * i
99              if grid_n[i][j] != 0:
100                 pygame.draw.rect(
101                     screen,
102                     ui_variables.t_color[grid_n[i][j]],
103                     Rect(dx, dy, block_size, block_size)
104                 )
```

기능) 이중 for문을 활용하여 게임화면 우측에 Next 블록을, 같은 구조로 게임 중 사용자가 Hold한 블록을 나타낸다.

- 93번 라인에서 grid_n으로 mino.py의 mino_map을 선언한 후 이중 반복문과 pygame.draw.rect를 활용하여 블록을 나타내고 있다.

이후의 코드에서는 score가 999999를 넘지 못하도록 설정한다. 또한 hold, next, level등 인자들 각각의 색을 선언한다. 그 후 screen.blit을 활용하여 색을 입힌 각각의 텍스트를 특정 위치에 배치한다.

```
146     for x in range(width):
147         for y in range(height):
148             dx = 17 + block_size * x
149             dy = 17 + block_size * y
150             draw_block(dx, dy, ui_variables.t_color[matrix[x][y + 1]])
```

- 뒤이어 이중 for문을 활용하여 게임의 board를 그리면서 draw_board 함수는 마무리된다.

5) ui_variables 클래스

line 22~26

```
22  class ui_variables:
23      # Fonts
24      font_path = "./assets/fonts/OpenSans-Light.ttf"
25      font_path_b = "./assets/fonts/OpenSans-Bold.ttf"
26      font_path_i = "./assets/fonts/Inconsolata/Inconsolata.otf"
```

pytris에 들어갈 font, sound를 변수로 선언한다. 해당 파일들은 개발자의 Github에서 다운받을 수 있다.

line 49~54

```
49      # Background colors
50      black = (10, 10, 10) #rgb(10, 10, 10)
51      white = (255, 255, 255) #rgb(255, 255, 255)
52      grey_1 = (26, 26, 26) #rgb(26, 26, 26)
53      grey_2 = (35, 35, 35) #rgb(35, 35, 35)
54      grey_3 = (55, 55, 55) #rgb(55, 55, 55)
```

또한 위와 같이 배경색과 블록의 색깔 역시 변수로 선언해주었다

6) tetrimino 클래스

line 1~10

```
1   class tetrimino:
2   ##########################################
3       mino_map = [
4           [
5               [
6                   [0, 0, 0, 0],
7                   [1, 1, 1, 1],
8                   [0, 0, 0, 0],
9                   [0, 0, 0, 0]
10              ],
```

7가지의 테트로미노 모형을 나타낸 모듈이다. 위와 같이 'tetrimino' 클래스로 작성되었으며 1부터 7까지 총 7개의 수로 서로 다른 7개의 블록을 나타낸다.

VII. 기존 프로젝트의 장단점 및 개선 사항

i. 장점

1. Pygame으로 작성되어 이해하기 쉽다.

Pygame은 PyQt5보다 쉬운 python GUI 개발툴이다. Pyqt는 라이브러리가 복잡하고, 내용이 방대하며, 어려운 부분이 많다. 하지만 Pygame은 캔버스와 그래픽 그리기, 다채널 사운드 처리, 창과 클릭 이벤트 처리, 충돌 감지 등의 복잡한 작업 없이도 여러 가지 GUI 지향 동작을 편리하게 다룰 수 있는 방편을 제공한다.

2. 최소한의 모듈로 프로그램을 구현하였다.

어려운 모듈이 아닌 Pygame, operator, random으로만 프로그램을 구현하였다. 따라서 프로그램이 더욱 간결하고 이해하기가 쉽다.

3. 소스 코드의 구성이 직관적이다.

코드 내 변수 및 함수의 이름을 직관적으로 설정하여 해당 요소에 대한 이해도를 높였다. 예를 들어 블록 그리기 함수 def draw_mino 와 같이 기능에 따라 이름을 직관적으로 정의하였다. 따라서 해당 함수와 변수가 어떠한 역할을 하고 있는지 쉽게 이해할 수 있다.

또한 변수와 함수들, 그리고 메인함수가 별도로 구분이 잘 되어있고 변수 부분 역시 폰트는 폰트, 음성파일은 음성파일, 색상은 색상 별로 충분한 여백과 함께 구분이 잘 되어있어 보기에 편하다.

4. 테트리스 블록을 효율적으로 구현하였다.

1부터 7까지의 숫자만을 활용하여 4*4 크기의 3D텐서로 테트리스 블록 모형을 만든 후, 이중 for문을 활용하여 색을 입히는 과정을 통해 테트로미노를 간략하게 구현했다.

ii. 단점

1. 파일이 2가지로만 구성되어 있다.

mino.py의 tetrimino를 제외한 모든 class와 함수가 한 페이지에 있어 코드 분석에 어려움이 있었다. 함수의 기능 별로 페이지를 나누면 코드에 대한 가독성이 더 좋았을 것으로 예상된다. 예를 들어 ui_variable같은 경우 따로 파이썬 파일을 만들어 import하고, main문이나 draw_board와 같은 중요 함수들을 기능별로 나누어 파일을 만들어 import했다면 파일이 더욱 간결하고 가독성도 향상되었을 것이다.

2. 사용자의 관점에서 보면 프로그램 설명이 부족하다.

프로젝트의 readme에는 프로그램의 기능이 자세하게 설명되어 있으나 프로그램만을 사용하는 사람의 입장에서는 설명이 부족하다. Soft_drop과 hard_drop, hold와 같이 좋은 기능이 많이 있지만 어떤 키를 눌러야 실행되는지 프로그램을 처음 본 사용자의 입장에서는 알 수 없다. 그러므로 게임을 시작하기 전 이와 같은 기능에 대한 설명 필요하다.

3. 소스 코드 내 오류가 존재한다.

게임 종료 후 변수의 초기화가 모두 수작업으로 진행되었다. 그러다 보니 개발자도 593, 594 라인에서 score를 두 번 초기화하는 오류를 범했다.