

OSSP

TETRIS



제출일 : 2023-01-25

과 목 : 오픈소스sw프로젝트

담당 교수 : 나인섭

분 반 : 02

학 과 : 컴퓨터공학과

학 번 : 20184433

이 름 : 김서현



조선대학교
CHOSUN UNIVERSITY

주제 - 테트리스게임

개발 환경

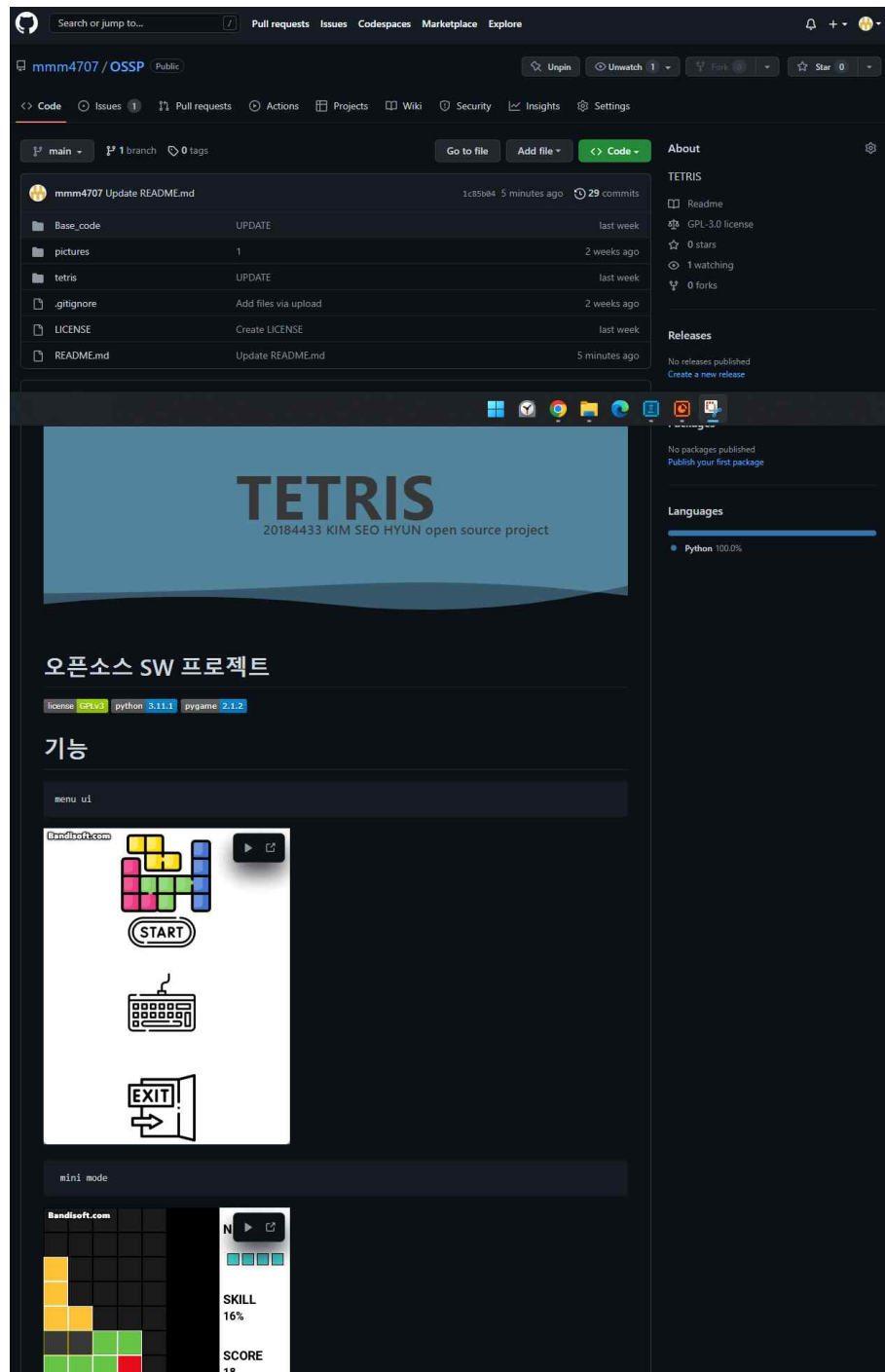
OS : Window 11

편집기 : VisualStudiocode 1.74.3

개발 언어 : Python 3.11.1

추가된 모듈 : Pygame 2.1.2 Pygame_menu4.3.6

Base_code : https://github.com/hbseo/OSD_game



기본 기능

우리가 아는 일반적인 테트리스 게임

종류의 블록이 랜덤하게 내려옵니다.

특정 키를 눌러 블록을 회전시킬 수 있습니다.

회전한 블록이 아래에 쌓입니다.

한 줄 가득 블록이 쌓이면 그 줄이 지워집니다.

게임이 시작할 때 배경음악이 시작된다

일시정지시 음악이 멈춤

최고 점수를 txt파일로 저장하여 게임을 다시 시작해도 최고 점수를 알 수 있다

그림자 : 현재 블록의 그림자를 알 수 있다

레벨 : 일정 목표를 완료하면 레벨이 올라간다

점수 : 레벨에 따라 점수가 오른다

목표 : 레벨을 올리기 위한 목표를 보여준다

게임 속도 : 레벨이 올라갈수록 블록이 빨리 떨어진다

다음 블록 미리보기 : 다음 떨어질 블록을 미리 확인할 수 있다

다시하기 : 게임이 끝나면 프로그램이 종료되지 않고 새 게임을 할 수 있다

Base_code License



GPL 라이선스 프로그램을 어떠한 목적으로든지 사용할 수 있다.

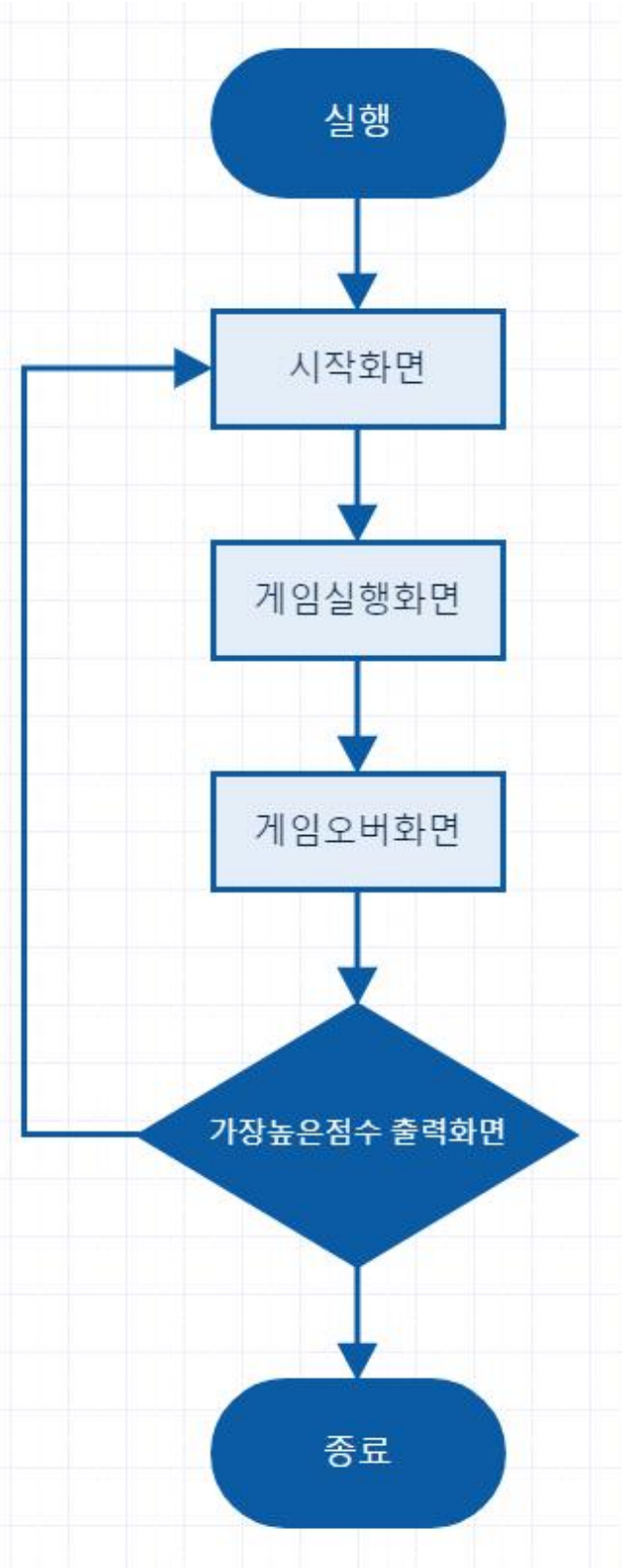
GPL 라이선스 프로그램의 소스 코드를 용도에 따라 변경 할 수 있다.

GPL 라이선스 프로그램을 판매/배포시 소스 코드도 요청하면 제공하여야 한다.

변경된 컴퓨터 프로그램 역시 프로그램의 소스 코드를 요청시 제공해야 한다.

변경된 컴퓨터 프로그램 역시 반드시 똑같은 GPL 라이선스를 취해야 한다.

Base code - flow chart



Base code – 클래스 구성 및 소스 분석

```
def run(self):
    pygame.init()
    icon = pygame.image.load('assets/images/icon.png')
    pygame.display.set_icon(icon)
    pygame.display.set_caption('Tetris')
    pygame.time.set_timer(pygame.USEREVENT, 500)
    start_sound = pygame.mixer.Sound('assets/sounds/Start.wav')
    start_sound.play()
    bgm = pygame.mixer.music.load('assets/sounds/bgm.mp3')
    while True:
        if self.check_reset():
            self.board.newGame()
            self.check_reset = False
            pygame.mixer.music.play(-1, 0.0)
        if self.board.game_over():
            self.screen.fill(BLACK)
            pygame.mixer.music.stop()
            self.board.GameOver()
            self.HighScore()
            self.check_reset = True
            self.board.init_board()
        for event in pygame.event.get():
            if event.type == QUIT:
                pygame.quit()
                sys.exit()
            elif event.type == KEYUP and event.key == K_p:
                self.screen.fill(BLACK)
                pygame.mixer.music.stop()
                self.board.pause()
                pygame.mixer.music.play(-1, 0.0)
            elif event.type == KEYDOWN:
                self.handle_key(event.key)
            elif event.type == pygame.USEREVENT:
                self.board.drop_piece()
            # self.screen.fill(BLACK)
            self.board.draw()
            pygame.display.update()
            self.clock.tick(30)
```

Tetris.py

108 line

테트리스게임을 실행하는 class

def __init__(self):

초기화 담당 메소드

def handle_key(self, event_key):

키보드 입력을 통한 동작 수행 메소드

def HighScore(self):

가장 높은 점수를 저장 하는 메소드

def run(self):

프로그램을 시작하는 부분 이미지로드, 배경음

악재생이 있는 메소드

```
class Piece:
    O = (((0,0,0,0,0), (0,0,0,0,0),(0,0,1,1,0),(0,0,1,1,0),(0,0,0,0,0)),) * 4

    I = (((0,0,0,0,0), (0,0,0,0,0), (0,2,2,2,2), (0,0,0,0,0), (0,0,0,0,0)),
          ((0,0,0,0,0), (0,0,2,0,0), (0,0,2,0,0), (0,0,2,0,0), (0,0,2,0,0)),
          ((0,0,0,0,0), (0,0,0,0,0), (2,2,2,2,0), (0,0,0,0,0), (0,0,0,0,0)),
          ((0,0,2,0,0), (0,0,2,0,0), (0,0,2,0,0), (0,0,2,0,0), (0,0,0,0,0)))

    L = (((0,0,0,0,0), (0,0,3,0,0), (0,0,3,0,0), (0,0,3,3,0), (0,0,0,0,0)),
          ((0,0,0,0,0), (0,0,0,0,0), (0,3,3,3,0), (0,3,0,0,0), (0,0,0,0,0)),
          ((0,0,0,0,0), (0,3,3,0,0), (0,0,3,0,0), (0,0,3,0,0), (0,0,0,0,0)),
          ((0,0,0,0,0), (0,0,0,3,0), (0,3,3,3,0), (0,0,0,0,0), (0,0,0,0,0)))

    J = (((0,0,0,0,0), (0,0,4,0,0), (0,0,4,0,0), (0,4,4,0,0), (0,0,0,0,0)),
          ((0,0,0,0,0), (0,4,0,0,0), (0,4,4,4,0), (0,0,0,0,0), (0,0,0,0,0)),
          ((0,0,0,0,0), (0,0,4,4,0), (0,0,4,0,0), (0,0,4,0,0), (0,0,0,0,0)),
          ((0,0,0,0,0), (0,0,0,0,0), (0,4,4,4,0), (0,0,0,4,0), (0,0,0,0,0)))

    Z = (((0,0,0,0,0), (0,0,0,5,0), (0,0,5,5,0), (0,0,5,0,0), (0,0,0,0,0)),
          ((0,0,0,0,0), (0,0,0,0,0), (0,5,5,0,0), (0,0,5,5,0), (0,0,0,0,0)),
          ((0,0,0,0,0), (0,0,5,0,0), (0,5,5,0,0), (0,5,0,0,0), (0,0,0,0,0)),
          ((0,0,0,0,0), (0,5,5,0,0), (0,0,5,5,0), (0,0,0,0,0), (0,0,0,0,0)))

    S = (((0,0,0,0,0), (0,0,6,0,0), (0,0,6,6,0), (0,0,6,0,0), (0,0,0,0,0)),
          ((0,0,0,0,0), (0,0,0,0,0), (0,0,6,6,0), (0,6,6,0,0), (0,0,0,0,0)),
          ((0,0,0,0,0), (0,6,0,0,0), (0,6,6,0,0), (0,0,6,0,0), (0,0,0,0,0)),
          ((0,0,0,0,0), (0,0,6,6,0), (0,6,6,0,0), (0,0,0,0,0), (0,0,0,0,0)))

    T = (((0,0,0,0,0), (0,0,7,0,0), (0,0,7,7,0), (0,0,7,0,0), (0,0,0,0,0)),
          ((0,0,0,0,0), (0,0,0,0,0), (0,7,7,7,0), (0,0,7,0,0), (0,0,0,0,0)),
          ((0,0,0,0,0), (0,0,7,0,0), (0,7,7,0,0), (0,0,7,0,0), (0,0,0,0,0)),
          ((0,0,0,0,0), (0,0,7,0,0), (0,7,7,7,0), (0,0,0,0,0), (0,0,0,0,0)))

    PIECES = {'O': O, 'I': I, 'L': L, 'J': J, 'Z': Z, 'S': S, 'T': T}

    T_COLOR = [yellow, cyan, orange, blue, red, green, pink, (55, 55, 55)]
```

Piece.py

67 line

테트리스 블록의 모양을 만드는 class

def __init__(self, piece_name=None):

초기화 부분 랜덤패키지를 이용해 랜덤으로 블록생성하는 메소드

def __iter__(self):

초기화 담당 메소드

def rotate(self, clockwise=True):

블록 모양을 바꿔주는 메소드

```

def draw(self):
    now = datetime.datetime.now()
    nowTime = now.strftime('%H:%M:%S')
    self.screen.fill(BLACK)
    for x in range(self.width):
        for y in range(self.height):
            x_pix, y_pix = self.pos_to_pixel(x, y)
            pygame.draw.rect(self.screen, (26,26,26),
                             (x_pix, y_pix, self.block_size, self.block_size))
            pygame.draw.rect(self.screen, BLACK,
                             (x_pix, y_pix, self.block_size, self.block_size),1)
    self.draw_blocks(self.piece, dx=self.piece_x, dy=self.piece_y)
    self.draw_blocks(self.board)
    pygame.draw.rect(self.screen, WHITE, Rect(250, 0, 350, 450))
    self.draw_next_piece(self.next_piece)
    next_text = pygame.font.Font('assets/Roboto-Bold.ttf', 18).render('NEXT', True, BLACK)
    skill_text = pygame.font.Font('assets/Roboto-Bold.ttf', 18).render('SKILL', True, BLACK)
    skill_value = pygame.font.Font('assets/Roboto-Bold.ttf', 16).render(str(self.skill)+'%', True, BLACK)
    score_text = pygame.font.Font('assets/Roboto-Bold.ttf', 18).render('SCORE', True, BLACK)
    score_value = pygame.font.Font('assets/Roboto-Bold.ttf', 16).render(str(self.score), True, BLACK)
    level_text = pygame.font.Font('assets/Roboto-Bold.ttf', 18).render('LEVEL', True, BLACK)
    level_value = pygame.font.Font('assets/Roboto-Bold.ttf', 16).render(str(self.level), True, BLACK)
    goal_text = pygame.font.Font('assets/Roboto-Bold.ttf', 18).render('GOAL', True, BLACK)
    goal_value = pygame.font.Font('assets/Roboto-Bold.ttf', 16).render(str(self.goal), True, BLACK)
    time_text = pygame.font.Font('assets/Roboto-Bold.ttf', 14).render(str(nowTime), True, BLACK)
    self.screen.blit(next_text, (255, 20))
    self.screen.blit(skill_text, (255, 120))
    self.screen.blit(skill_value, (255, 145))
    self.screen.blit(score_text, (255, 200))
    self.screen.blit(score_value, (255, 225))
    self.screen.blit(level_text, (255, 275))
    self.screen.blit(level_value, (255, 300))
    self.screen.blit(goal_text, (255, 350))
    self.screen.blit(goal_value, (255, 375))
    self.screen.blit(time_text, (255, 430))

```

Board.py

337 line

화면을 구성하는 class

def __init__(self, screen):

초기화 담당 메소드

def init_board(self):

변수 초기화 담당 메소드

def generate_piece(self):

모양 정의 메소드

def nextpiece(self):

다음에 나올 블록을 그려주는 메소드

def absorb_piece(self):

다음에 나올 블록을 없애주는 메소드

def block_collide_with_board(self, x, y):

블록 충돌 관련 메소드

def collide_with_board(self, dx, dy):

블록 충돌 관련 메소드

def can_move_piece(self, dx, dy):

블록이 움직일 수 있는 경우를 판단하는 메소드

def can_drop_piece(self):

아래로 한칸 내려가는것을 실행하는 메소드

def try_rotate_piece(self, clockwise=True):

블록을 회전하고 벽과 충돌하는것을 인식하는 메소드

def move_piece(self, dx, dy):

블록을 움직이는 메소드

```

def drop_piece(self):
블럭을 내리는 메소드
def full_drop_piece(self):
블럭을 완전히 밑으로 내리는 메소드
def rotate_piece(self, clockwise=True):
블럭을 회전시키는 메소드
def pos_to_pixel(self, x, y):
위치를 반환하는 메소드
def pos_to_pixel_next(self, x, y):
다음 위치 반환하는 메소드
def delete_line(self, y):
x좌표로 한줄블럭이 완성됐을때 y좌표도 계산해서 한번에 라인을 지
워주는 메소드
def delete_lines(self):
x좌표로 한줄블럭이 완성됐을때 라인을 지워주는 메소드
def level_speed(self):
레벨별 스피드를 조절하는 메소드
def game_over(self):
게임이 종료됐음을 반환하는 메소드
def draw_blocks(self, array2d, color=WHITE, dx=0, dy=0):
블럭 모양을 만들어주는 메소드
def draw_next_piece(self, array2d, color=WHITE):
다음 블럭 모양을 만들어주는 메소드
def draw(self):
보드 내 필요한 내용들을 넣어주는 메소드
def pause(self):
게임을 일시정지시켜주는 메소드
def GameOver(self):
게임오버를 알려주는 메소드
def newGame(self):
새로운 게임을 시작하는 메소드
def HS(self, txt="no"):
가장 높은 점수를 보여주는 메소드
def ultimate(self):
조건이 완료되었을 때 스킬을 발동시켜주는 메소드

```

내 소스 깃허브 주소

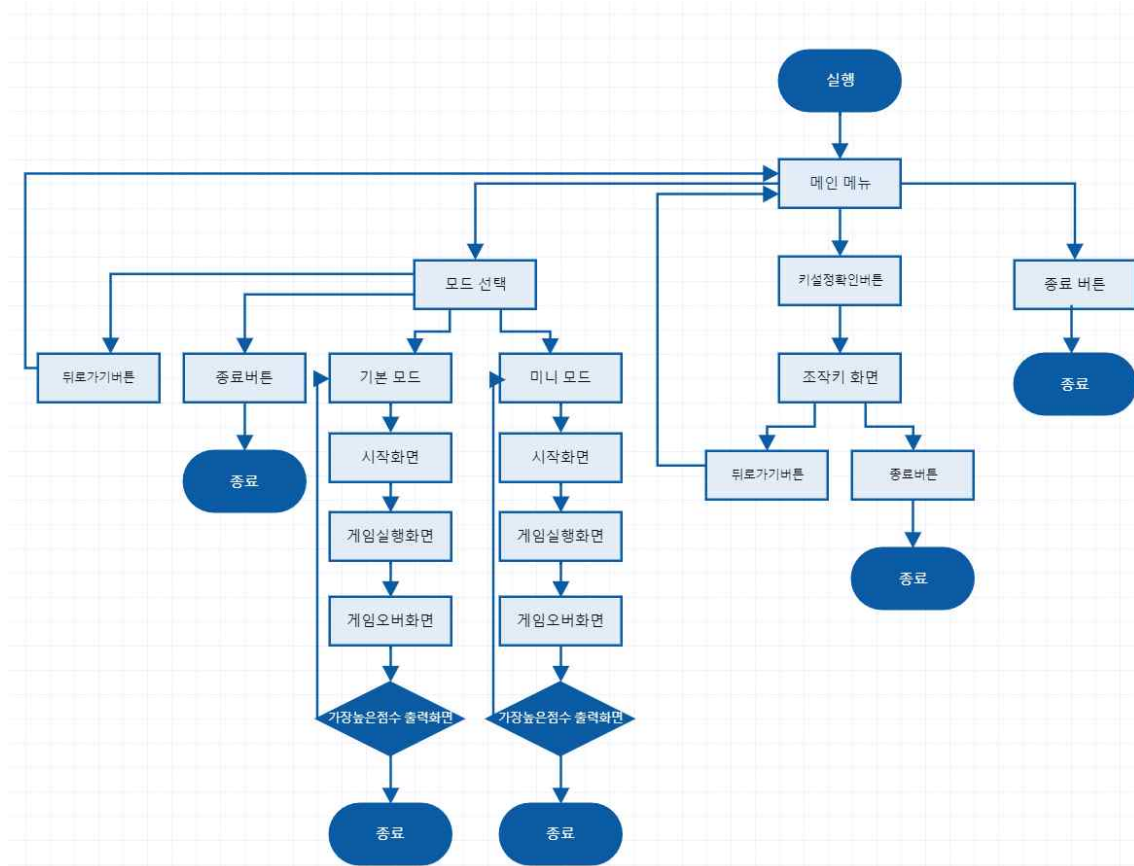
<https://github.com/mmm4707/OSSP>

The screenshot shows a GitHub repository page for a project named 'TETRIS'. The repository is owned by 'mmm4707' and is currently on the 'main' branch. The page displays a list of files and folders: 'Base_code' (UPDATE, last week), 'pictures' (1, 2 weeks ago), 'tetris' (UPDATE, last week), '.gitignore' (Add files via upload, 2 weeks ago), 'LICENSE' (Create LICENSE, last week), and 'README.md' (Update README.md, last week). Below the file list, there is a section for the 'README.md' file, which features a large yellow banner with the word 'TETRIS' in bold, followed by '20184433 KIM SEO HYUN open source project'. Underneath the banner, the text '오픈소스 SW 프로젝트' (Open Source SW Project) is displayed. At the bottom of the README section, there are tags for 'license: GPLv3', 'python: 3.11.1', and 'pygame: 2.1.2'.

내 소스 License



내 소스 flow chart



기존 소스와 차별점: 목록 및 출력물 비교

기존 소스에서는 게임 기능만 구현되고 메뉴부분이 없고 조작키 설명 부분도 없어 사용자 입장에서 많이 불편절하다고 느꼈습니다.

그래서 사용자를 위한 메뉴 gui, 조작키설명부분을 구현하였고 간단한 테트리스를 원하는 사용자를 위해 미니모드를 추가해 보았습니다.

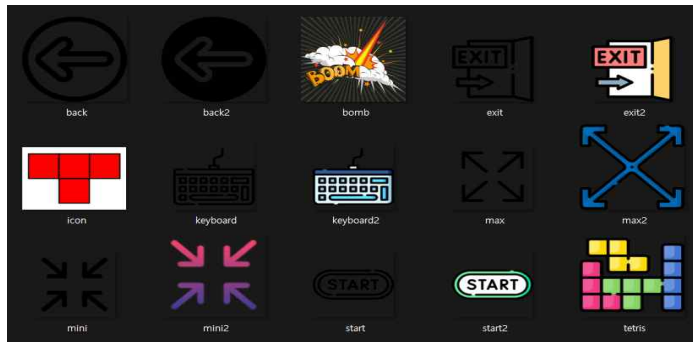
그리고 작동시켜보니 특정 상황에서 효과음이 출력되지 않는 점 블럭이 충돌되면서 정상작동이 되지않는 부분, 그림자가 이상하게 출력되는 부분등 많은 문제점이 발견되어 모두 수정 완료하였습니다.

run.py라는 파일을 만들어서 132줄의 코드를 추가하고 기존 실행파일이었던 tetris.py에서 run.py로 실행파일을 바꿨습니다.

그리고 다양한 오류를 Board.py파일에서 바꿨고 주석도 달아주었습니다.

Board.py를 베이스로 mini_Board.py파일을 만들어서 미니모드도 추가해주었습니다.

메뉴 아이콘 저장 부분



아이콘을 다운받은 사이트 : <https://www.flaticon.com/>

사용한 폰트 : roboto font , gmarket font

Roboto 폰트 다운받은 사이트 : <https://fontmeme.com/ktype/roboto-font/>

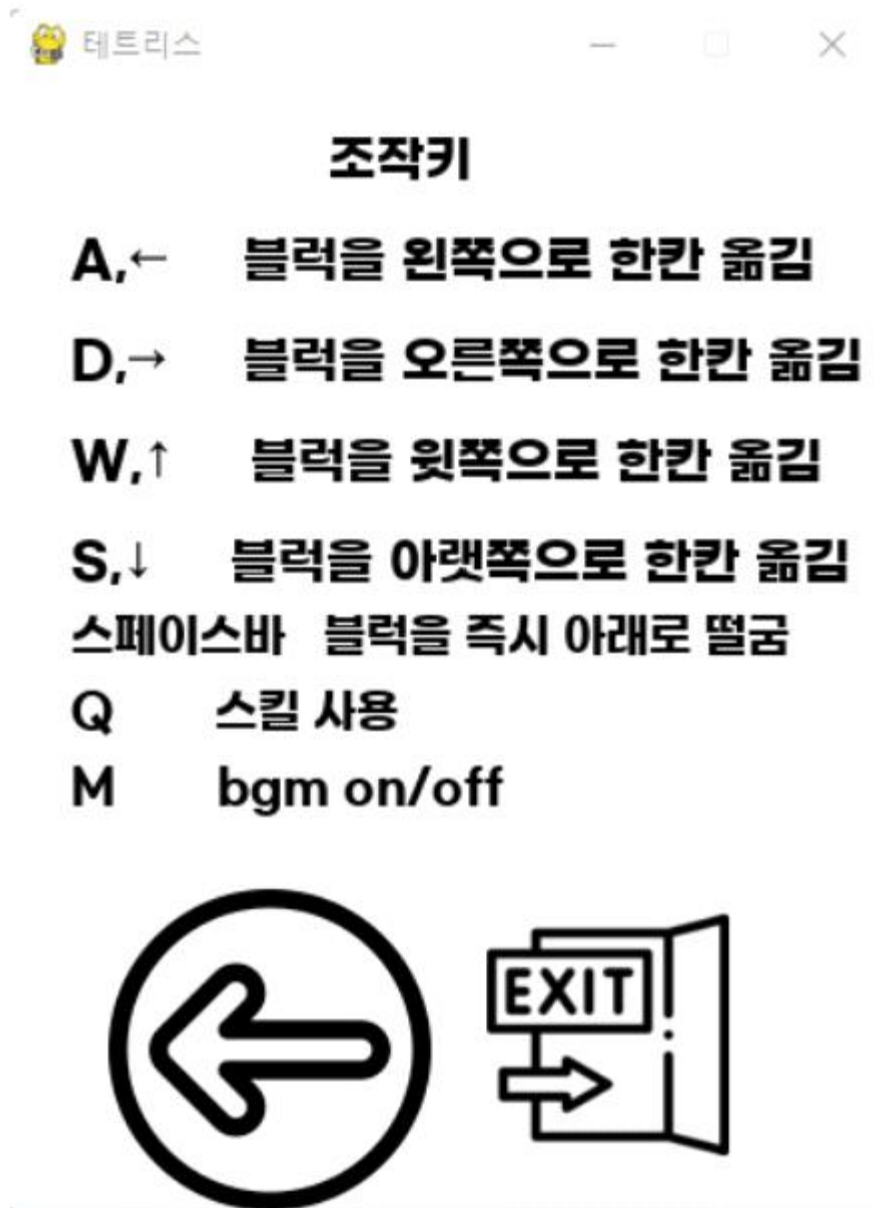
Gmarket 폰트 다운받은 사이트: <https://corp.gmarket.com/>

실행화면 1 - 메인 메뉴 부분

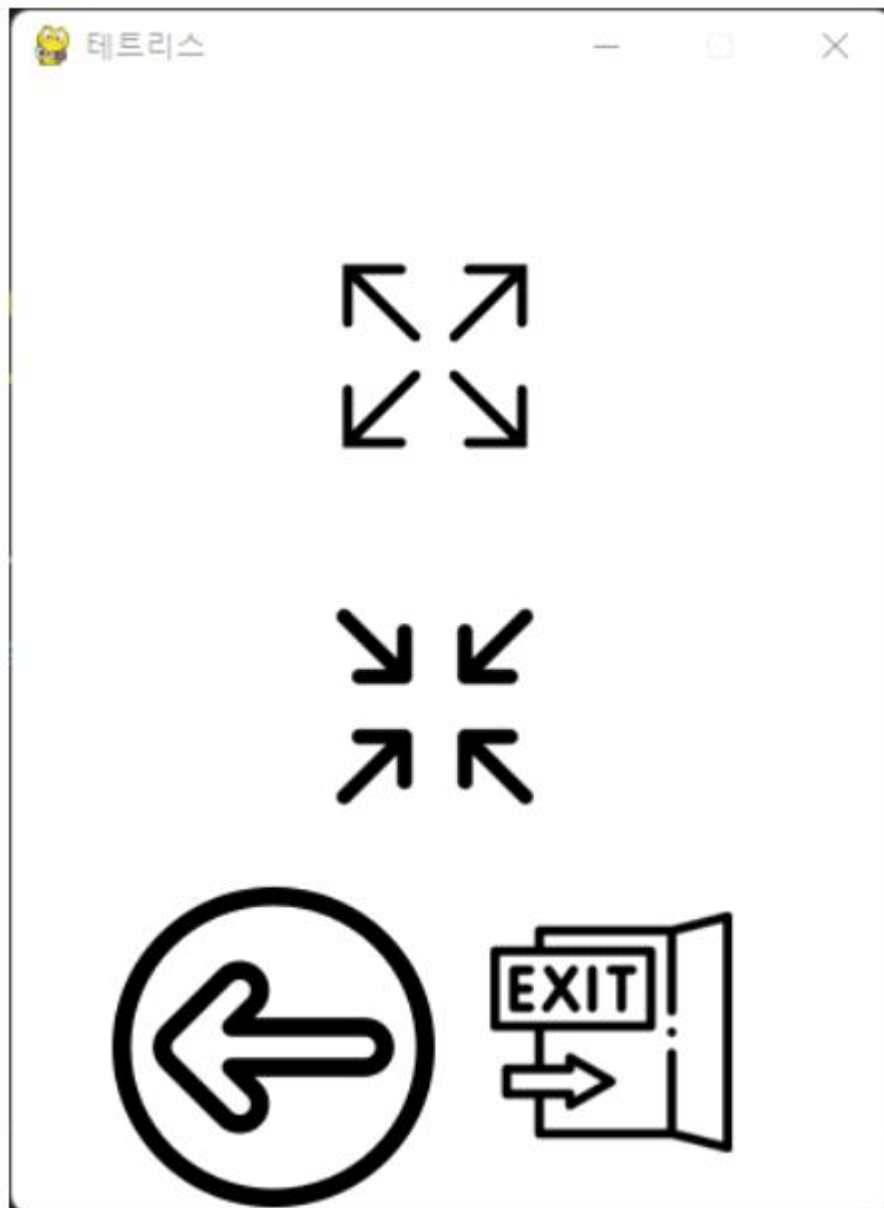
테트리스



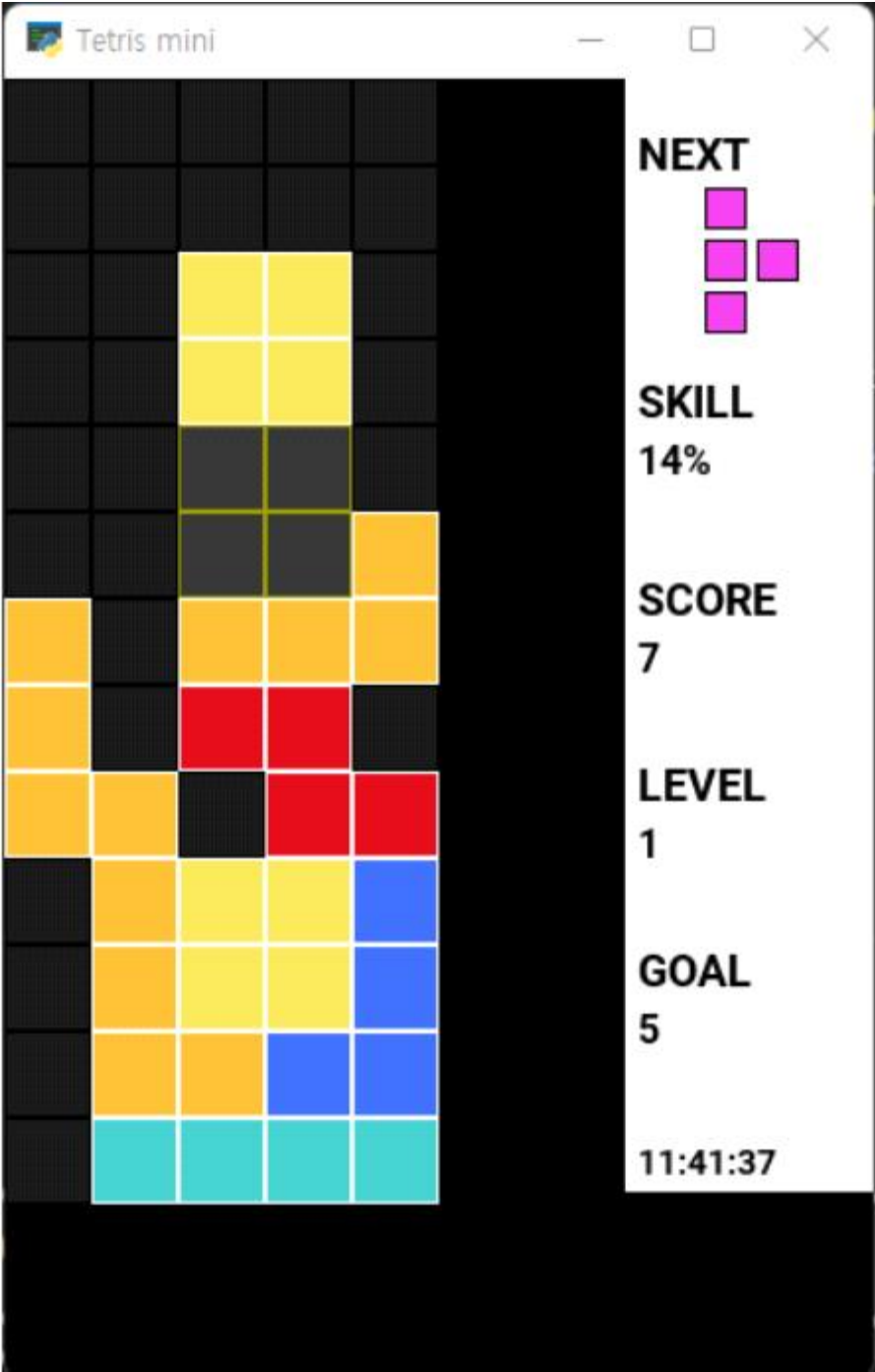
실행화면2 - 조작키 설명부분



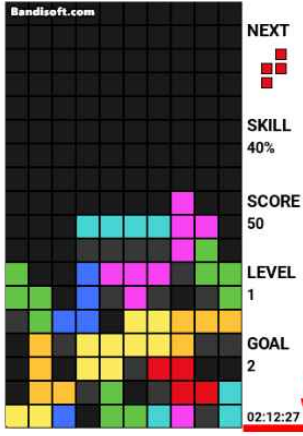
실행화면3 -모드 선택부분 (미니모드와 기본모드)



실행화면4 - 미니 모드 게임화면



실행화면5 - 그림자 버그수정, 현재시간에서 플레이시간으로 변경



기존 소스코드
그림자오류 발생

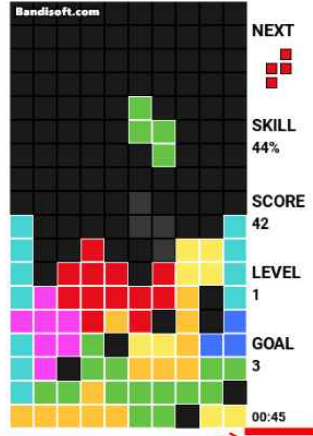
```
def draw_shadow(self, array2d, dx, dy):
    for y, row in enumerate(array2d):
        y += dy
        if y >= 2 and y < self.height:
            for x, block in enumerate(row):
                x += dx
                if block:
                    tmp = 1
                    while self.can_move_place(0, tmp):
                        tmp += 1
                    x_s, y_s = self.pos_to_pixel(x, y + tmp - 1)
                    pygame.draw.rect(self.screen, self.piece.f.COLOR[2],
                                   (x_s, y_s, self.block_size, self.block_size))
                    pygame.draw.rect(self.screen, BLACK,
                                   (x_s, y_s, self.block_size, 1))
```

← 현재시간을 나타내주는 소스코드

```
now = datetime.datetime.now() #현재시간을 나타내주는 소스코드
nowTime = now.strftime("%H:%M:%S")
```

← 플레이시간을 나타내주는 코드

```
sec = round(time.time() - self.start) #플레이한 시간을 나타내주는 코드
r_time = datetime.datetime.fromtimestamp(sec).strftime("%H:%M:%S")
```



변경된 프로그램
그림자오류가 생기지 않음

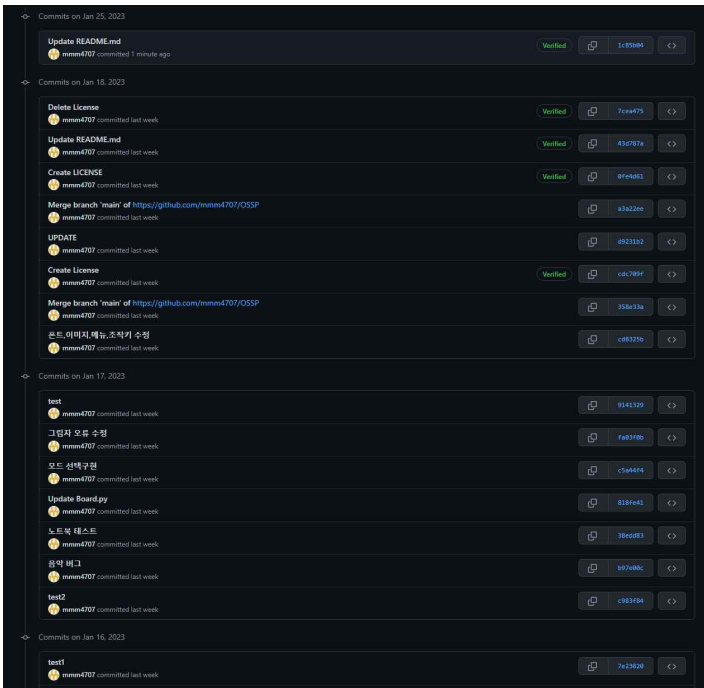
실행 화면은 github readme파일에 gif로 올려놓았습니다.

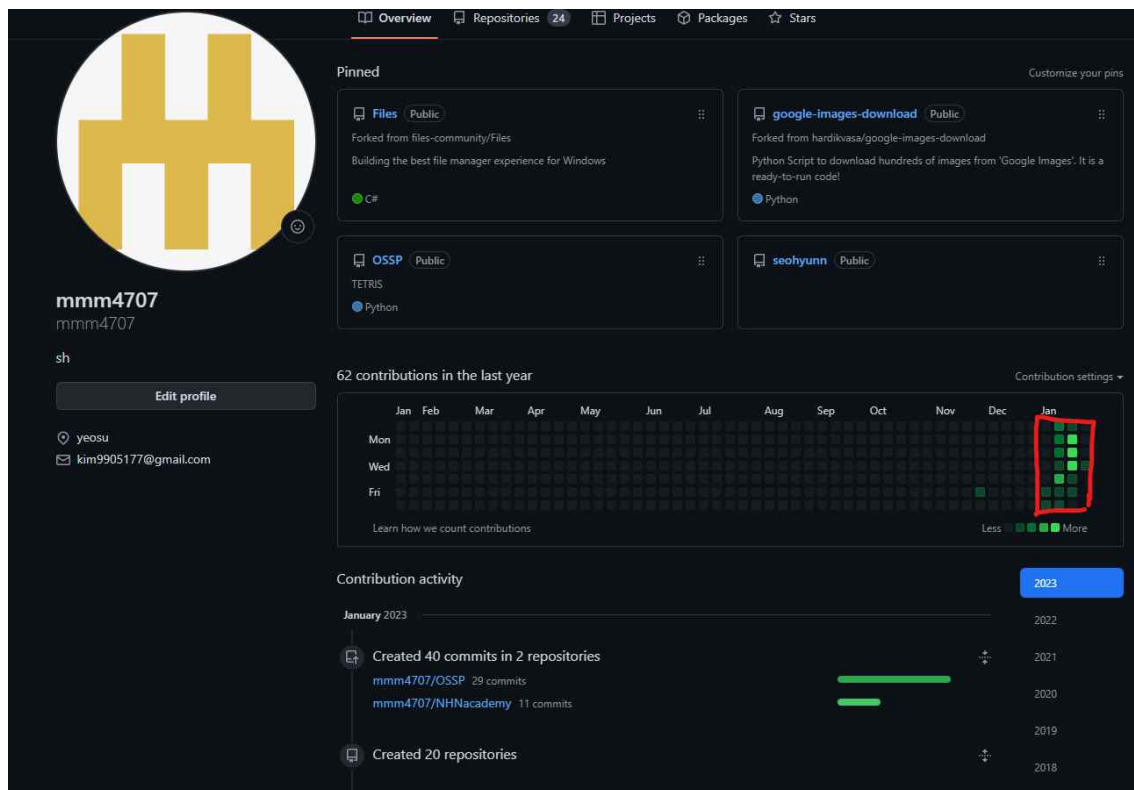
GITHUB LOG

총 28 COMMIT 과제가 시작된 이후로 하루에 1COMMIT 이상을 하였음

<https://github.com/mmm4707>

<https://github.com/mmm4707/OSSP/commits/main>





느낀점

처음에는 오픈소스프로젝트라 간단할 것이라 생각했습니다.

하지만 오픈소스주제를 정하는것, 오픈소스를 선정하는 것, 선정된 오픈소스를 실행하기 위한 환경을 구성하는 것, 오픈소스에 기능을 추가하고 수정하는것 어떤것 하나 쉬운과정이 없었습니다.

Nhn java아카데미를 병행하며 과제를 수행하느라 java와 python을 번갈아 수행해서 혼동스러운 순간도 있었습니다.

하지만 이번 프로젝트를 하면서 github 환경에 많이 익숙해져서 하루에 어떤것이든 1commit 이상을 하자는 목표도 생기게 되었고 python도 많이 공부하게 되어서 프로젝트를 다 마친 지금은 정말 뿌듯하고 후련합니다.