과제-5 **(Tetris) 완성된 테트리스 게임**

① 수행 과업

- 테트리스 게임의 완성.
- 점수 시스템과 게임 종료 기능 추가.

② 요구사항

1. 점수 시스템

- 블록을 쌓아 라인을 삭제할 때마다 점수 증가.
- 기능 요구사항
  - 삭제된 라인의 개수에 따라 점수 배점.
  - 화면에 현재 점수 표시.

2. 게임 종료

- 블록이 보드의 상단까지 쌓이면 게임 종료.
- 기능 요구사항:
  - 게임 종료 메시지 출력 및 화면 정지.

③ 스스로 학습

1. 게임 종료 처리
- 게임 종료 조건:
  - 새로운 블록이 생성될 때 이미 보드 상단에 고정된 블록이 있으면 게임 종료.
- 구현 아이디어:
  - 새로운 블록이 생성된 위치에서 check_collision()을 호출하여 충돌 여부를 확인.
  - 충돌 발생 시 게임 종료 처리.

2. 게임 진행 흐름
- 게임 루프
  - 블록 자동 하강 → 충돌 감지 → 고정 및 새로운 블록 생성.
  - 새로운 블록 생성 시 충돌 발생 → 게임 종료.
  - 삭제된 라인 처리 및 점수 업데이트.

3. 스스로 학습을 위한 질문
- 게임 종료 처리:
  - 새로운 블록이 생성되었을 때 게임 종료 조건을 어떻게 확인할 수 있을까?
- 코드 구조화
  - 이전에 구현한 각 기능(라인 삭제, 충돌 감지, 점수 계산)을 통합할 때 어떤 구조가 효율적일까?

④ 코드 구현 예제를 이해하여 설명하시오.

```
import pygame
import random
# Pygame 초기화
pygame.init()
# 화면 크기 설정
WIDTH, HEIGHT = 300, 600
ROWS, COLS = 20, 10
BLOCK_SIZE = WIDTH // COLS
# 색상 정의
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
BLUE = (0, 0, 255)
RED = (255, 0, 0)
# 블록 모양 정의 (I, O, T, S, Z, L, J)
SHAPES = [
    [[1, 1, 1, 1]],  # I
    [[1, 1], [1, 1]],  # O
    [[0, 1, 0], [1, 1, 1]],  # T
    [[0, 1, 1], [1, 1, 0]],  # S
    [[1, 1, 0], [0, 1, 1]],  # Z
    [[1, 0, 0], [1, 1, 1]],  # L
    [[0, 0, 1], [1, 1, 1]],  # J
]# 게임 보드 초기화
def create_board():
    return [[0 for _ in range(COLS)] for _ in range(ROWS)]
# 블록 회전 함수
def rotate_block(block):
    return [list(row) for row in zip(*block[::-1])]
# 블록 그리기 함수
def draw_block(screen, shape, x, y):
    for row_idx, row in enumerate(shape):
        for col_idx, cell in enumerate(row):
            if cell:
                pygame.draw.rect(
                    screen,
                    BLUE,
                    (x + col_idx * BLOCK_SIZE, y + row_idx * BLOCK_SIZE, BLOCK_SIZE,
BLOCK_SIZE),
                )
# 보드 그리기 함수
def draw_board(screen, board):
    for row_idx, row in enumerate(board):
        for col_idx, cell in enumerate(row):
            if cell:
                pygame.draw.rect(
                    screen,
                    RED,
                    (col_idx * BLOCK_SIZE, row_idx * BLOCK_SIZE, BLOCK_SIZE,
BLOCK_SIZE),
```

```
                    )
# 라인 삭제 함수
def clear_lines(board):
    cleared_rows = 0
    new_board = []
    for row in board:
        if all(cell == 1 for cell in row):  # 해당 행이 꽉 찬 경우
            cleared_rows += 1
        else:
            new_board.append(row)
    for _ in range(cleared_rows):
        new_board.insert(0, [0] * COLS)
    return new_board, cleared_rows
# 점수 계산 함수
def calculate_score(cleared_rows):
    score_table = {1: 100, 2: 300, 3: 500, 4: 800}
    return score_table.get(cleared_rows, 0)
# 충돌 감지 함수
def check_collision(board, shape, x, y):
    for row_idx, row in enumerate(shape):
        for col_idx, cell in enumerate(row):
            if cell:
                board_x = (x // BLOCK_SIZE) + col_idx
                board_y = (y // BLOCK_SIZE) + row_idx
                if board_x < 0 or board_x >= COLS or board_y >= ROWS:
                    return True
                if board_y >= 0 and board[board_y][board_x] != 0:
                    return True
    return False
# 블록 고정 함수
def place_block(board, shape, x, y):
    for row_idx, row in enumerate(shape):
        for col_idx, cell in enumerate(row):
            if cell:
                board_x = (x // BLOCK_SIZE) + col_idx
                board_y = (y // BLOCK_SIZE) + row_idx
                if 0 <= board_x < COLS and 0 <= board_y < ROWS:
                    board[board_y][board_x] = 1
def main():
    screen = pygame.display.set_mode((WIDTH, HEIGHT))
    pygame.display.set_caption("Tetris - 완성")
    clock = pygame.time.Clock()
    board = create_board()
    running = True
    current_block = random.choice(SHAPES)
    block_x, block_y = 4 * BLOCK_SIZE, 0
    drop_time = 0
    speed = 50
    total_score = 0
    while running:
```

```
        screen.fill(BLACK)
        # 시간 기반 블록 하강
        drop_time += clock.get_rawtime()
        clock.tick(30)
        if drop_time > speed:
            if not check_collision(board, current_block, block_x, block_y + BLOCK_SIZE):
                block_y += BLOCK_SIZE
            else:
                place_block(board, current_block, block_x, block_y)
                board, cleared_rows = clear_lines(board)
                total_score += calculate_score(cleared_rows)
                speed = max(50, speed - (cleared_rows * 20))  # 속도 증가
                current_block = random.choice(SHAPES)
                block_x, block_y = 4 * BLOCK_SIZE, 0
                # 게임 종료 조건
                if check_collision(board, current_block, block_x, block_y):
                    print("Game Over!")
                    running = False
            drop_time = 0
        # 이벤트 처리
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_UP:  # 블록 회전
                    rotated_block = rotate_block(current_block)
                    if not check_collision(board, rotated_block, block_x, block_y):
                        current_block = rotated_block
        # 키 입력 처리
        keys = pygame.key.get_pressed()
        if keys[pygame.K_LEFT]:
            if not check_collision(board, current_block, block_x - BLOCK_SIZE, block_y):
                block_x -= BLOCK_SIZE
        if keys[pygame.K_RIGHT]:
            if not check_collision(board, current_block, block_x + BLOCK_SIZE, block_y):
                block_x += BLOCK_SIZE
        if keys[pygame.K_DOWN]:
            if not check_collision(board, current_block, block_x, block_y + BLOCK_SIZE):
                block_y += BLOCK_SIZE
        # 보드 및 블록 그리기
        draw_board(screen, board)
        draw_block(screen, current_block, block_x, block_y)
        # 점수 표시
        font = pygame.font.SysFont("Arial", 24)
        score_text = font.render(f"Score: {total_score}", True, WHITE)
        screen.blit(score_text, (10, 10))
        pygame.display.flip()
    pygame.quit()
if __name__ == "__main__":
    main()
```

【Peer 평가 Check List】(Tetris) 완성된 테트리스 게임

설명자 Explainer		평가자 Evaluator		날짜 date	
☐ 전혀 설명하지 못함	☐ 기본적인 설명은 했으나 부족함		☐ 완벽하게 이해하고 명확히 설명함		

【기능 이해 및 코드 해석 능력】

1. 블록 회전 구현 이해

☐ rotate_block 함수가 블록을 90도 회전시키는 방법(행과 열 변환)을 정확히 설명했는가?

☐ 회전 후 충돌 여부를 감지하고 회전을 취소하는 조건을 이해했는가?

2. 라인 삭제 및 점수 시스템

☐ clear_lines 함수가 꽉 찬 행을 삭제하고, 위의 행을 한 칸씩 아래로 이동시키는 과정을 이해했는가?

☐ 삭제된 줄의 개수에 따라 점수를 계산하는 방식(calculate_score 함수)을 명확히 설명했는가?

3. 충돌 감지 및 블록 고정

☐ check_collision 함수가 블록이 경계나 고정된 블록과 충돌하는 경우를 어떻게 감지하는지 설명했는가?

☐ 블록이 고정된 후 새로운 블록을 생성하는 흐름을 정확히 이해했는가?

4. 게임 종료 처리

☐ 게임 종료 조건(새로운 블록이 생성된 위치에서 충돌 발생)을 명확히 이해하고 설명했는가?

☐ 게임 종료 시 화면 출력("Game Over!")이 어떻게 이루어지는지 설명했는가?

5. 게임 종료 처리

☐ 삭제된 줄의 누적 개수에 따라 블록 속도가 점진적으로 빨라지는 로직을 이해했는가?

☐ speed 값이 감소하여 블록 하강 속도가 빨라지는 과정을 설명했는가?

【설명 능력】

6. 명확성과 논리성

☐ 동료가 이해하기 쉽도록 프로그램의 주요 기능과 동작 원리를 논리적으로 설명했는가?