OSSProj 주간보고서

이번 주 주요 내용	• AWS 랭크 보여주기 방식 결정 • AWS 각 로컬컴퓨터에서 정상 작동 확인

교과목명	오픈소스소프트웨어프로젝트			담당교수	김동호 교수님
과제명	오픈소스를 활용한 테트리스 기능 개발			팀명	DoitDoit
회의일자	10/29	시간	16:00 - 17:30	장소	Webex

내용

1. AWS 개발 상황 점검

1) 고명섭 학생이 만든 AWS RDS와 VSCODE간의 연동을 각 로컬컴퓨터에서 확인 후 최종 Merge 성공

2) 이용자가 더 편하게 각 모드별 랭킹을 확인하기 쉽도록, 추가적으로 각 모드별 기록된 점수가 게임 모드 페이지에 표시되도록 결정

```
## 여기서부터 기록 저장
if difficulty == 1: ## normal
    cursor = tetris.cursor()
    name2 = chr(name[0]) + chr(name[1]) + chr(name[2])
    sql = 'INSERT INTO Normal (id, score) VALUES (%s,%s)'
    cursor.execute(sql, (name2, score))
    tetris.commit()
    cursor.close() ## tetris db insert
if difficulty == 2: ## hard
    cursor = tetris.cursor()
    name2 = chr(name[0]) + chr(name[1]) + chr(name[2])
    sql = 'INSERT INTO Hard (id, score) VALUES (%s,%s)'
    cursor.execute(sql, (name2, score))
    tetris.commit()
    cursor.close()
if difficulty == 3: ## ITem
    cursor = tetris.cursor()
    name2 = chr(name[0]) + chr(name[1]) + chr(name[2])
    sql = 'INSERT INTO Item (id, score) VALUES (%s,%s)'
    cursor.execute(sql, (name2, score))
    tetris.commit()
    cursor.close()
```

2. 피버모드 1차 완성

1) 기존의 점수구간별 설정에서, 블록 일정 개수 격파시 피버모드가 주어지도록 변경

2) 추가 회의를 통해 점수구간에 따라 피버시간이 점진적으로 오르도록 변경

3) 게임 루프가 돌 때 t0 시간을 측정한다. 이후 피버모드 발생 시 t1 시간을 측정하고, 둘의 차이로 피버모드 시간이 결정된다.

4) 기본 피버시간 27초를 Basic_Time 변수로 전환할 계획입니다.

```
# 콤보회수에 따른 피버타임

if 1500 > score >= 0:
    ADD = FEVERTIMER[0]
if 5000 > score >= 1500:
    ADD = FEVERTIMER[1]
if 15000 > score >= 5000:
    ADD = FEVERTIMER[2]
if 30000 > score >= 15000:
    ADD = FEVERTIMER[3]
if comboCounter > 4:
    t1 = time.time()
    mino = randint(1, 1)
    next_mino = randint(1, 1)
    next_fever = (c + fever_interval) * fever_score # 피버모드 점수 표시

    # fever time시 이미지 깜빡거리게
    if blink:
        screen.blit(pygame.transform.scale(ui_variables.fever_image,
                                           (int(SCREEN_WIDTH * 0.5), int(SCREEN_HEIGHT
                    (SCREEN_WIDTH * 0.1, SCREEN_HEIGHT * 0.1))
        blink = False
    else:
        blink = True
dt = t1 - t0
if dt >= 27+ADD:
    t0 = t1
    comboCounter =0
    mino = next_mino
    next_mino = randint(1, 7)
```

3. 모드별 담당자들의 개발방향과 의견논의

1) AWS 이후의 개발사항을 미리 점검하는 논의를 진행

2) 다음 진행할 사항인 Hard Mode의 장애물과, Normal Mode의 속도 조절을 어떤 식으로 구현할 지 논의하고자 함

3) 각 모드별 담당자마다 담당하는 부분에 관련된 코드를 리뷰하여 공유

4) 이후 모드에 들어갈 기능을 미리 구현해둔 소스코드를 가져와 사용방안에 대해 논의

김수빈 – Hard Mode 담당자

1) DIFFICULTY PAGE에서 selected 변수에 의해 모드가 결정된다.

```
elif page == DIFFICULTY_PAGE:
   # 난이도를 설정한다.
   DIFFICULTY_COUNT = 6
   DIFFICULTY_NAMES = ["easy", "NORMAL", "HARD", "PvP", "SPEED &
MINI", "REVERSE"]
   current_selected = selected
   for event in pygame.event.get():
```

중략

```
elif event.key == K_RIGHT:
    pygame.key.set_repeat(0)
```

```
    if selected < DIFFICULTY_COUNT - 1:
       # next difficulty select
       ui_variables.click_sound.play()
       selected = selected + 1
    elif event.key == K_LEFT:
         pygame.key.set_repeat(0)
         if selected > 0:
            # previous difficulty select
            ui_variables.click_sound.play()
            selected = selected - 1
    if event.key == K_SPACE:
      pygame.key.set_repeat(0)
      if 0 <= selected < 3:
      # start game with selected difficulty
      ui_variables.click_sound.play()
      start = True
      init_game(DEFAULT_WIDTH, DEFAULT_HEIGHT, selected)
```

2) 설정된 selected는 init_game에서 game_difficulty(difficulty) 변수로 사용되고, 게임을 시작하는 코드에서 framerate로 사용된다.

```
def init_game(board_width, board_height, game_difficulty):
    global width, height, matrix, matrix_2P, difficulty, framerate

    width = board_width
    height = board_height

    matrix = [[0 for y in range(board_height + 1)] for x in
range(board_width)]
    matrix_2P = [[0 for y in range(board_height + 1)] for x in
range(board_width)]

    difficulty = game_difficulty
    framerate = STARTING_FRAMERATE_BY_DIFFCULTY[difficulty]
```

```
# Game Screen
elif start:
 중략
 # Set speed
    if not game_over:
       keys_pressed = pygame.key.get_pressed()
       if keys_pressed[K_DOWN]:
          pygame.time.set_timer(pygame.USEREVENT, framerate * 1)
       else:
          pygame.time.set_timer(pygame.USEREVENT, framerate * 5)
```

3) 따라서 속도만 빨랐던 기존 hard mode의 속도를 기본 속도로 수정하고, 장애물이 나오도록 구현하기 위해서는 PvP mode의 attack 코드를 참고하여 hard mode에서 실행시킬 수 있

는 새로운 함수를 추가할 계획

```
for i in range(2, max_score, attack_interval):
    if score > i * attack_score and score < (i * attack_score +
300):  # 1000~1300,2500~2800,4000~4300
       if blink:
         screen.blit(pygame.transform.scale(ui_variables.pvp_annoying_
image,(int(SCREEN_WIDTH * 0.4), int(SCREEN_HEIGHT * 0.9))),
(SCREEN_WIDTH * 0.5, SCREEN_HEIGHT * 0))  # 이미지 깜빡거리게
         blink = False
       else:
         blink = True
```

고다희 – Normal Mode 속도 조절 담당자

기존 PINTRIS에서 start screen 내부에 page를 나누는 방식으로 구현한 점을 유지, 페이지를 추가해 Normal Mode에서만 속도 조절이 가능한 것이 아닌 모든 모드에서 가능하게 변경

1) 현재 설정된 페이지

```
START_PAGE, MENU_PAGE, HELP_PAGE, SETTING_PAGE, DIFFICULTY_PAGE = 0,
10, 11, 12, 20
```

⇒ 기존에 모드 선택 페이지였던 DIFFICULTY_PAGE를 MODE_PAGE로 변경하고, 게임 시작 속도 설정하는 페이지로 DIFFICULTY_PAGE를 추가한다.

* 페이지 흐름의 변화

```
Start page <-> Menu Page <-> Diffculty Page -> Start
```

⇒ Start page <-> Menu Page <-> Mode Page -> Difficulty Page -> Start

2) 기존에 모드 선택 페이지에서 K_SPACE 이벤트가 발생(모드 선택) 시, init_game 함수가 호출되며 게임이 실행되고, 이때의 difficulty는 easy_difficulty, normal_difficulty, hard_difficulty의 3가지로 나눠져 있다.

⇒ DIFFICULTY_PAGE에서 게임 시작 시의 게임 하강 속도 레벨을 1~9로 선택할 수 있고, 속도 레벨이 한 칸 올라갈 때는 framerate에서 일정 값을 빼고, 내려갈 때는 더하는 걸로 시작 속도를 조절한다. (STARTING_FRAMERATE_BY_DIFFCULTY 변수 삭제 가능)

* init_game에서 framerate 변경하는 코드 수정

```
def init_game(board_width, board_height, game_difficulty):
중략
    difficulty = game_difficulty
    framerate = STARTING_FRAMERATE_BY_DIFFCULTY[difficulty]
```

⇒ framerate -= difficulty*speed_change (여기서 speed_change는 게임 중 레벨업에 따라서도 속도가 변동되기 때문에 최대 속도를 고려해서 설정)

* 기존: K_SPACE 이벤트 발생 시, 바로 게임 시작

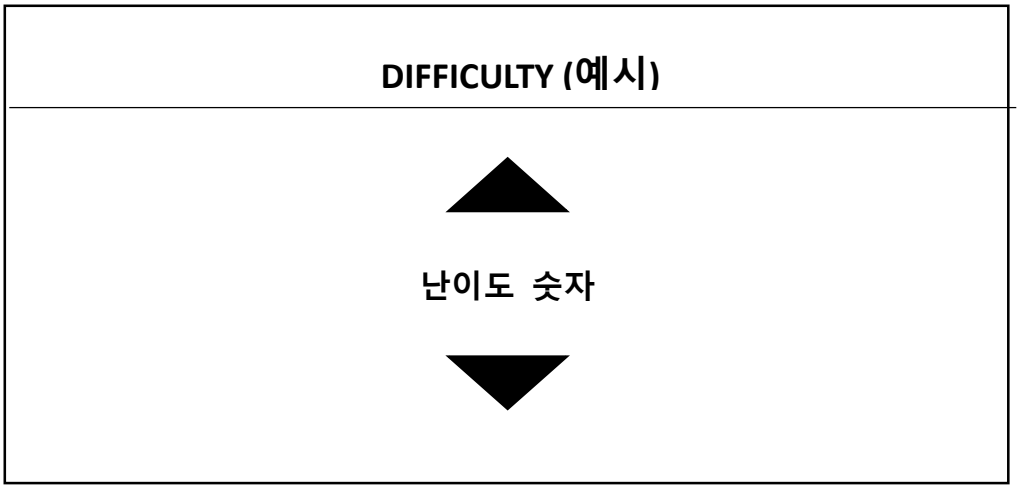
selected=0이면 easy, 1이면 normal, 2이면 hard 모드이고 각 모드에 따른 스피드가 정해져 있으며, 그 외에 pvp, reverse은 자동으로 normal 속도로 정해놨다.

```
if event.key == K_SPACE:
   pygame.key.set_repeat(0)
```

```
if 0 <= selected < 3:
  ui_variables.click_sound.play()
  start = True
  init_game(DEFAULT_WIDTH, DEFAULT_HEIGHT, selected)
```

⇒ 바로 init_game으로 설정해서 게임을 시작하는 게 아니라, page = DIFFICULTY_PAGE로 페이지 변경 후 다시 key event를 발생시켜 위쪽 화살표(K_UP)일 때는 difficulty를 1 증가, 아래쪽 화살표(K_DOWN)일 때는 1 감소시킨다. Difficulty 설정 후, K_SPACE가 들어오면 앞 페이지에서 설정된 selected에 따라서 모드를 읽고 init_game에 변경된 difficulty를 넣어 게임을 시작하는 방식으로 구현 예정

* 추가될 페이지 예시



난이도 숫자가 올라가는 부분에 대한 고민 필요

4. 추가 논의 사항 및 다음주 진행 예정 사항

1) 최종 Merge 전에 Open request를 로컬 컴퓨터로 미리 가져와서 확인 할 수 있는 방법
2) Hard Mode 구현 시작