OSSP 주간보고서

과목	오픈소스 소프트웨어 프로젝트 (SCS4045-01)
담당교수	김동호 교수님
팀명	추박김 (CHUPARKIM)
주제	테트리스 버전업
기간	04/26-05/02

CHUPARKIM wiki 🧠

home

dev rule

meeting log

회의록: https://github.com/CSID-DGU/2021-OSSPC-CHUPARKIM-5/wiki/meeting-log

1. 피드백 후 방향 잡기

- 제안서 발표 후 교수님의 피드백: ’무엇을 하려는 지 정확히 모르겠다’, ‘완성본을 실행할 때 프로그램이 버벅거릴 것 같다’, ‘할 일의 양이 적어 보인다’
- 조교님의 피드백: ‘구체적으로 완성된 모습이 어떻게 될 지 모호하다’

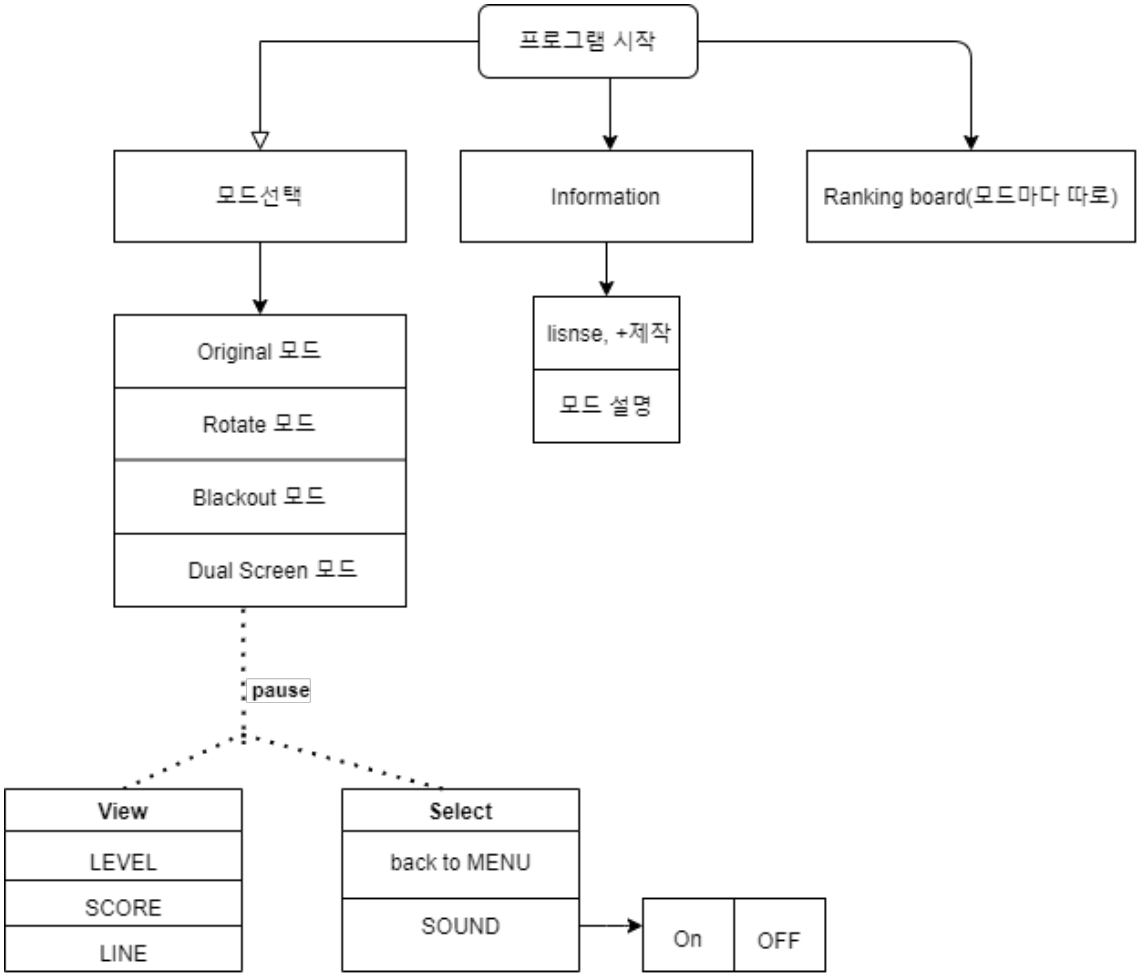
➜ 추가할 rotate, blackout, quater 모드의 구현에 긴 기간이 걸릴 것이라고 예상했지만 코드 분석을 통해 예상기간이 훨씬 단축될 것으로 판단함
➜ 구체적인 모드 구현 방식과 추가할 기능들에 대해 회의 진행

2. 협업 규칙

- 변수명, 함수명은 자유롭게 정한다. 하지만 새롭게 선언한 경우 반드시 주석처리를 통해 해당 변수나 함수의 목적을 밝히도록 한다.
- 버그 발생, 기능 추가 등의 내용에 대해 issue를 생성하여 의견을 주고받는다.
- merge하기 위해서는 항상 pull request를 거친다. pull request가 있는 경우 해당 issue를 보고 반드시 피드백을 남긴다.
- 회의는 매주 수요일과 일요일에 정기적으로 진행한다. 회의 시작 후 10분간은 현재 작업 상황에 대해 개략적으로 이야기하고 시작한다. 회의는 특별한 issue가 없어도 반드시 참석하는 것으로 하고, 회의 후에는 매번 스케줄을 re-mapping한다. 그리고 이를 반드시 구글 공유문서에 기록한다.

과목	오픈소스 소프트웨어 프로젝트 (SCS4045-01)
담당교수	김동호 교수님
팀명	추박김 (CHUPARKIM)
주제	테트리스 버전업
기간	05/03-05/09

1. 플로우차트



- 메뉴 선택은 마우스로 선택하는 방식으로 한다.
- 기존의 네 개의 테트리스를 동작하는 quarter screen mode ➜ 두 개의 테트리스를 동작하는 dual screen으로 compact down
- Base source인 PYTRIS에서는 pause시 아무것도 보이지 않았지만, pause시 level, score, line 값이 보이도록 한다.
- pause시 메뉴 선택으로 돌아갈 수 있도록 하고, sound on/off 기능을 추가한다.
- mode가 늘어났으므로 구체적으로 어떠한 모드인지 설명해주는 Information menu를 추가한다.

- Base source인 PYTRIS에서 메인 화면의 좌측상단에 3위까지만 보여줬던 스코어 보드를 따로 열람할 수 있도록 해당 메뉴를 추가한다. 스코어 보드는 mode별로 따로 표시되게끔 한다. Mode가 4개이기 때문에 스코어 보드도 4개이다.
- dual screen mode의 경우 두 개의 테트리스가 별개로 스코어를 갖는다고 표시되지만, 두 테트리스의 점수를 합산하여 하나의 스코어로 계산 되도록한다.
- rotate mode의 구현을 위하여 테트리스를 오른쪽 방향으로만 회전시킬 수 있도록 한다.
- original mode와 rotate모드는 배경을 삽입하여 회전하는 사용자가 방향성을 정확하게 인지할 수 있도록 한다. 이미지는 오래된 TV이미지를 사용하여 TV안에 테트리스가 실행되고 있는 것으로 한다.
- black out mode를 가볍게 구현한 결과 단순히 배경을 칸을 인식할 수 있는 격자를 없앤다고 해도 공간개념에 혼란을 줘서 게임의 재미를 높이려는 목표를 달성하기 어렵다. 따라서 시점이 매번 떨어지는 block에 고정되도록 한다.
- 게임의 전체적인 theme는 retro로 과거 DOS 컴퓨터 화면의 느낌을 motive로 한다. 이에 맞는 배경, 폰트, 색조, 음악을 구현한다.

2. 협업 규칙 업데이트

- 메인 저장소: 오픈소스 프로젝트 github main repository (upstream)
- 내 저장소: github 각자 계정에 있는 repository (메인 저장소로부터 fork한 저장소, origin)
- 로컬 저장소: 각자 컴퓨터에 생성한 git repository file
- 작업 공간: 실제 소스코드가 있는 workspace

  메인 저장소를 내 저장소로 fork한다. 그러면 내 원격 저장소에 저장되어 작업할 수 있는 환경이 마련된다.
  로컬 저장소에 clone한 후, origin과 upstream을 설정한다.
  git remote add origin [url]
  git remote add upstream [url]

- main: 배포 가능한 상태만을 관리한다.
- develop:
  기능 개발을 위한 branch들을 병합하기 위해 사용한다.
  모든 기능이 추가되고 버그가 수정되어 배포가능한 안정적인 상태라면 develop branch를 main branch에 merge한다.
- feature: 작업할 때에는 각자 새로운 branch에서 작업한다.
  개발이 완료되면 develop branch로 merge하여 팀원들과 공유한다.
  branch 이름은 [이름/기능요약] 형식을 사용한다. ex) YJ/blackout
  새로운 기능 개발 및 버그 수정이 필요할 때마다 develop branch로부터 분기한다.

- Github wiki, readme 활용에 대해 wiki는 팀의 활동 내용을 중심으로 작성하고, readme는 사용자를 고려하여 작성하도록 함

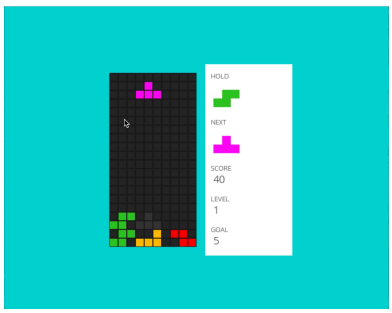
3. 프로젝트 진행 상황

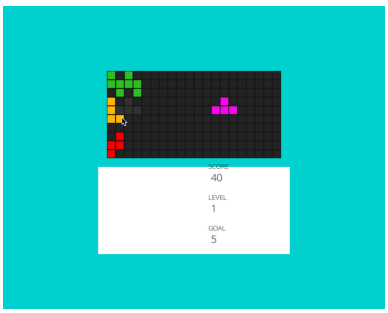
- 보드 사이즈 조절
  레트로 테마 적용을 위해 사이즈 조절

- Rotate mode

```
game_key = ( # 회전 횟수에 따른 게임 키 조정
    (K_RIGHT, K_LEFT),
    (K_DOWN, K_UP),
    (K_LEFT, K_RIGHT),
    (K_UP, K_DOWN),
)
```

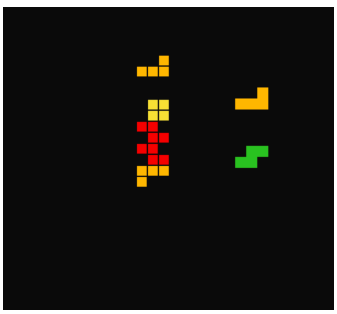
```
# Draw board 칸그리기
for x in range(width):
    for y in range(height):
        if g_type == 0:  # 0회전
            dx = 17 + 188 + block_size * x
            dy = 17 + 113 + block_size * y
        elif g_type == 1:  # 1회전 = 270도 회전
            dx = 17 + 188 + block_size * (height - y - 1)
            dy = 17 + 113 + block_size * x
        elif g_type == 2:  # 2회전 = 180도 회전
            dx = 17 + 188 + 170 + block_size * (width - x)
            dy = 17 + 113 + block_size * (height - y - 1)
        elif g_type == 3:  # 3회전 = 90도 회전
            dx = 17 + 188 + block_size * y
            dy = 17 + 113 + 170 + block_size * (width - x - 1)

        draw_block(dx, dy, ui_variables.t_color[matrix[x][y + 1]
```





회전 횟수를 나타내는 g_type이라는 변수를 새로 만들었고 Draw board 함수 내에서 회전 횟수를 기준으로 그려지는 방향이 다르게 구현
또, 회전 횟수에 따라 사용자가 인지하는 방향이 달라져서 게임 진행 키가 바뀌게 되므로 game_key 튜플 안에 넣어서 각 방향마다 알맞는 방향키로 게임할 수 있게 수정

- Blackout mode



배경 및 전체적인 디자인을 모두 black으로 변경하고 locx, locy 변수를 추가하여 시점 고정 구현

- Dual screen mode

- 코드가 버벅거릴 거라는 교수님의 피드백에 대해 진지하게 고민되어야 함. pygame의 메모리 사용문제를 구글링해보고 해당 문제를 겪은 일취월장팀의 ohmytris 코드를 분석해 볼 필요가 있음.

➜ Base source인 PYTRIS는 기능에 대한 부분은 모듈화가 잘되어있지만 loop이 돌아가는 실질적으로 main역할을 하는 부분에서 기능별로 모듈화가 되어있지 않음. 무분별하게 남용되는 if 문으로 프로그램이 돌아가기 때문에 while문이 계속해서 반복되는 loop이 한 번 실행될 때 시간이 오래 걸림. 순차적인 주석 처리 과정을 통해 ohmytris의 코드를 분석한 결과 함수가 콜 되는 부분이 상당부분 엉켜 있는 것 같음. 본 팀의 프로젝트 역시 프로그램을 구성하는데 있어서 main 부분이 길어질 수밖에 없음. 따라서 최대한 기능을 분리해 메인 부분을 모듈화 하고, 새로 추가할 모드를 구현할 때 최대한 기능들을 모듈화해서 main loop을 간소화해서 개발할 필요가 있음. 인터페이스 구현 단계에서 파이썬 파일을 따로 해서 상수들과, 함수부분, main 부분을 분리할 것도 고려해야 됨.

4. 현재 issue 및 추후 일정

- Rotate mode

사용자가 받아들이는 시점을 기준으로 원하는 방향을 방향키로 입력하면 블록이 따라가는 방식을 채택하였을 때 1회전, 3회전 시에 “K_UP(위 방향키)”이 블록 이동을 위해 쓰이게 됨 ➜ 회전키 변경 필요 (다른 모드와 혼동을 주지 않기 위해 전부 변경하는 쪽으로)
게임 진행 키 변경으로 인해 사운드 재생에서 오류 발생
블록이 왼쪽 이동을 위해 K_LEFT를 입력하였을 때 "사운드1" 재생되어야 하는데 rotate.py 코드의 경우 2회전 상태에서 K_LEFT를 입력하였을 때 실제 블록은 오른쪽으로 이동하나 동일하게 "사운드1"이 재생됨
사이드바 텍스트 위치 수정 필요 ➜ 보드 크기 감소 or 전체 게임창 크기 증가

- Blackout mode

현재 고정되어 있는 시점의 위치 수정 필요
사이드바에 표시되는 next mino까지 blackout

- Rotate mode + blackout mode를 결합한 새로운 mode 개발 고려 ➜ lab형식으로 구현 시도해 볼 것

- 인터페이스 source 자료 수집

상준: 회전키 변경, 사운드 수정, 사이드바 위치 수정, main 모듈화, 인터페이스 source 자료수집
예진: blackout mode 보드판 중앙으로 시점 고정 위치 변경, 스코어보드까지 blackout, main 모듈화, 인터페이스 source 자료수집
정현: dual Screen mode 구현완성, main 모듈화, 인터페이스 source 자료수집