2024-2
고급프로그래밍
프로젝트

Dune 1.5

한남대학교 컴퓨터공학과

제출

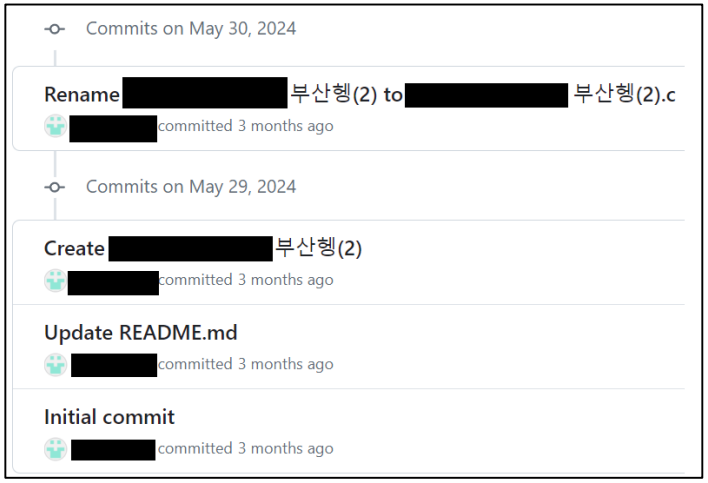
- **중간 제출**
  - 5) 시스템 메시지까지 하면 만점
  - ~11월 5일(화) 23:59
  - 소스 파일(zip 압축) +
  - 1)~5)에 구현된 기능들을 확인 가능한 실행 영상 제출
  - engine.c 상단에 주석으로 어디까지 했는지 설명

- **최종 제출**
  - ~12월 10일(화) 23:59
  - github repository URL 제출
    - private으로 설정, 12월 16일(월) 이후 public으로 전환
    - commit log가 제대로 작성되지 않은 경우 채점하지 않음.
    - 도중에 문제가 생긴 경우 스크린샷 제출
    - README에
      - “00학번(년도만) 홍길동” 이라고 쓰고,
      - 1)~11) 중 어디까지 했는지,
      - 길찾기, 시야 등을 어떻게 구현했는지 알고리즘 설명

commit 예시

- 적절한 로그→

- 의미 없는 로그 (채점 안 함)



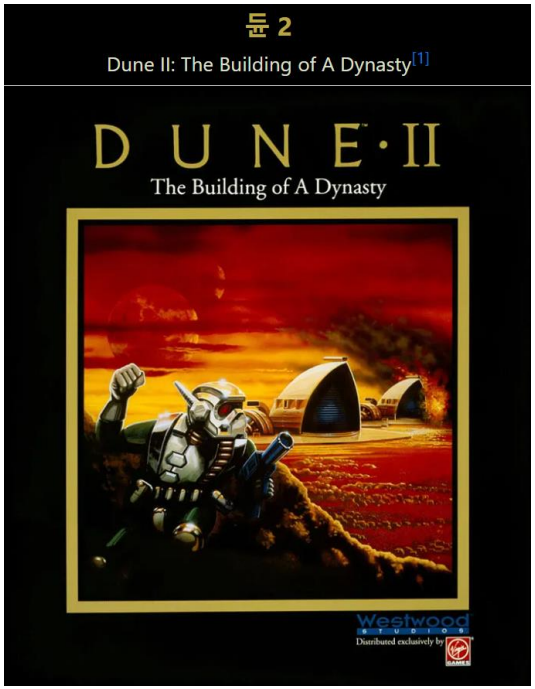
RTS

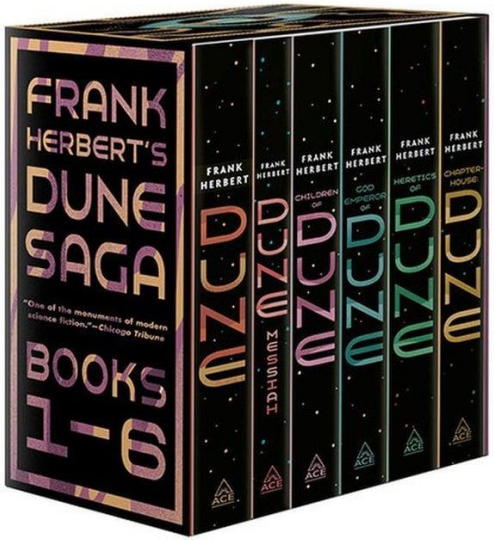
- **RTS(Real-Time Strategy)**
  - 실시간 전략 게임, 실시간 전략 시뮬레이션 게임
  - 주어진 맵에서 실시간으로
  - 매장된 자원 채취
  - → 건물 짓기
  - → 건물에서 유닛 생산
  - → 적 유닛 & 건물 파괴

Dune

그림 출처:
나무위키, 위키백과 등

- **Dune 원작(1965)**
  - SF 소설의 시조새
- **Dune II 게임(1992)**
  - RTS(Real Time Strategy) 게임의 아버지(최초는 아님)
  - → Command and conquer, Warcraft I, II → Starcraft





개요

- **장소: 사막 행성 아라키스**
  - 우주에서 유일한 중요 자원(**스파이스**) 생산지

- **내용: 황제와 유력 가문들의 스파이스 쟁탈전**

아트레이드

하코넨

- **특수 유닛**
  - 샌드웜:사막의 포식자 & 스파이스의 생산자
  - 프레멘: 아라키스 행성의 원주민. 사막 특수부대라고 생각하면 됨
  - 사다우카: 황제의 정예병

Dune II

- ‘하베스터’가 스파이스 매장지에서 자원 채취
- 장판? 을 깔아야 건물을 지을 수 있음
- 장판이 없는 지형(사막)에 있는 유닛은 랜덤하게 샌드웜의 습격을 받음

TUTORIAL

Sleep()

- **Sleep(): 프로그램이 일정 시간 잠듦**
  - 예제) 1초마다 1 증가하는 변수
    - 참고) 가장 단순한 방법이라서 Sleep()을 사용하고 있지만, 이 방법으로는 정확한 시간을 제어하지는 못함

```c
#include <stdio.h>
#include <Windows.h>

int main(void) {
	int sec = 1;
	while (1) {
		printf("%d\n", sec);
		sec++;
		Sleep(1000); // ms
	}
}
```

```
1
2
3
4
5
6
7
...
```

Sleep()

- tick: 이 프로그램이 동작하는 최소 시간 단위(1 tick = 10 ms)

```
#include <stdio.h>
#include <Windows.h>
#define TICK 10  // ms
int sys_clock = 0;   // ms
int main(void) {
        int sec = 1;
        while (1) {
                // 3초마다 인사
                if (sys_clock % 3000 == 0) {
                        printf("안녕하세요\n");
                }
                // 1초마다 sec ++
                if (sys_clock % 1000 == 0) {
                        printf("sec: % d\n", sec);
                        sec ++;
                }
                Sleep(TICK); // 10ms 주기로 루프
                sys_clock += 10;
        }
}
```

```
안녕하세요
1
2
3
안녕하세요
4
5
6
안녕하세요
7
...
```

키 입력 받기

- **<conio.h>:_kbhit()**
  - 키 입력이 있는지 확인하는 함수
- **<conio.h>:_getch()**
  - 입력된 키를 전달 받는 함수(엔터 키가 입력될 때까지 기다리지 않음)
  - blocking function이므로 _kbhit()가 참일 때만 사용

```
...
int main(void) {
    while (1) {
        if (_kbhit()) {
            int key = _getch();
            if (key == 'q') { break; } // 'q'를 누르면 종료
        }
    Sleep(TICK); // loop 돌 때마다(TICK==10ms마다) 키 입력 확인
    }
}
```

연습문제

- ‘q’가 입력되고 나서 3초 후에 종료하기

키 타입 정의

- **수업에서 enum은 생략했는데, 어렵지 않으니 예제 검색해보기**
  - KEY 타입의 enum은 k_none, k_up, …, k_undef 중 하나의 값을 가짐
  - 실제로는 k_none부터 차례로 정수 0, 1, 2, …가 부여됨

```
typedef enum {
          k_none = 0, k_up, k_right, k_left, k_down,
          k_quit,
          k_undef, // 정의되지 않은 키 입력
} KEY;
```

방향키 입력 받기

- 다른 키('a', b', '1', '!', …)와 달리 2byte로 입력 된다.
  - **MSB(224) + LSB(72/80/75/77)**
- 일반 키는 1 byte, 방향키라면 2 byte를 읽어야 함. 첫번째 byte를 확인해서 구분

```
KEY get_key(void) {
          if (!_kbhit()) {  // 입력된 키가 있는지 확인
                    return k_none;
          }

          int byte = _getch();    // 입력된 키를 전달 받기
          switch (byte) {
          case 'q': return k_quit;  // 'q'를 누르면 종료
          case 224:
                    byte = _getch();  // MSB 224가 입력 되면 1바이트 더 받음
                    switch (byte) {
                    case 72: return k_up;
                    case 75: return k_left;
                    case 77: return k_right;
                    case 80: return k_down;
                    default: return k_undef;
                    }
          default: return k_undef;
          }
}
```

방향키 입력 받기

- 메인함수

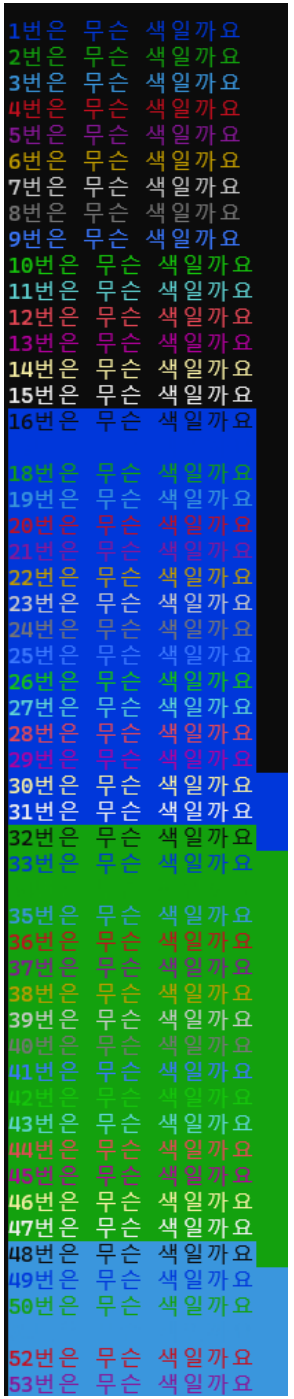
```
int main(void) {
        while (1) {
        KEY key = get_key();

        switch (key) {
        case k_quit:      printf("exiting..."); return 0;
        case k_up:        printf("up\n"); break;
        case k_left:      printf("left\n"); break;
        case k_right:     printf("right\n"); break;
        case k_down:      printf("down\n"); break;
        default: break;
        }

        Sleep(TICK);
        clock += 10;
        }
}
```

```
left
right
down
left
up
right
right
down
left
exiting...
```

문자 색상 변경

- **SetConsoleTextAttribute()**
  - https://dev-with-precious-dreams.tistory.com/5
  - https://blog.naver.com/sharonichoya/220874370397

```
for (int color = 0x00; color < 0xFF; color++) {
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), color);
    printf("%d번은 무슨 색일까요\n", color);
}
```



커서 이동시키기

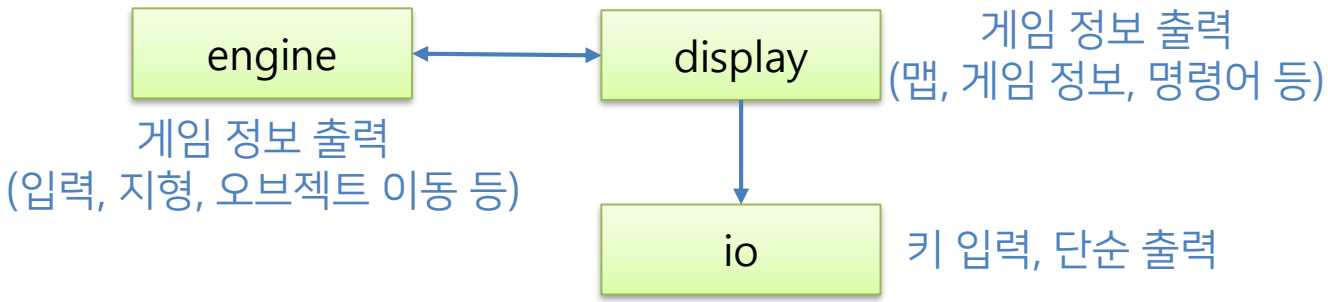
- SetConsoleCursorPosition()
  - https://blog.naver.com/dgsw102/221014320762

```
#include <Windows.h>
#include <conio.h>

// 문자 색상 변경
SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), color);

// 커서를 특정 위치로 이동시키기
COORD coord = { column, row }; // 행, 열 반대로 전달
SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
printf("출력할 내용");
```

SKELETON CODE

스켈레톤 코드

- **기본 구조**


engine
display
io
게임 정보 출력
(맵, 게임 정보, 명령어 등)
게임 정보 출력
(입력, 지형, 오브젝트 이동 등)
키 입력, 단순 출력

스켈레톤 코드

- **자료 구조**
  - 맵: layer 0(map[0])은 지형, layer 1(map[1])은 오브젝트
  - POSITION: 맵 상의 위치를 나타냄

```
#define N_LAYER 2
#define MAP_WIDTH  60
#define MAP_HEIGHT 18
char map[N_LAYER][MAP_HEIGHT][MAP_WIDTH] = { 0 };
```

```
typedef struct {
        int row, column;
} POSITION;
```

  - KEY: 입력 키 정의
  - DIRECTION: 이동 방향

```
typedef enum {
        k_none = 0, k_up, k_right, k_left, k_down,
        k_quit,
        k_undef, // 정의되지 않은 키 입력
} KEY;
```

```
typedef enum {
        d_stay = 0, d_up, d_right, d_left, d_down
} DIRECTION;
```

스켈레톤 코드

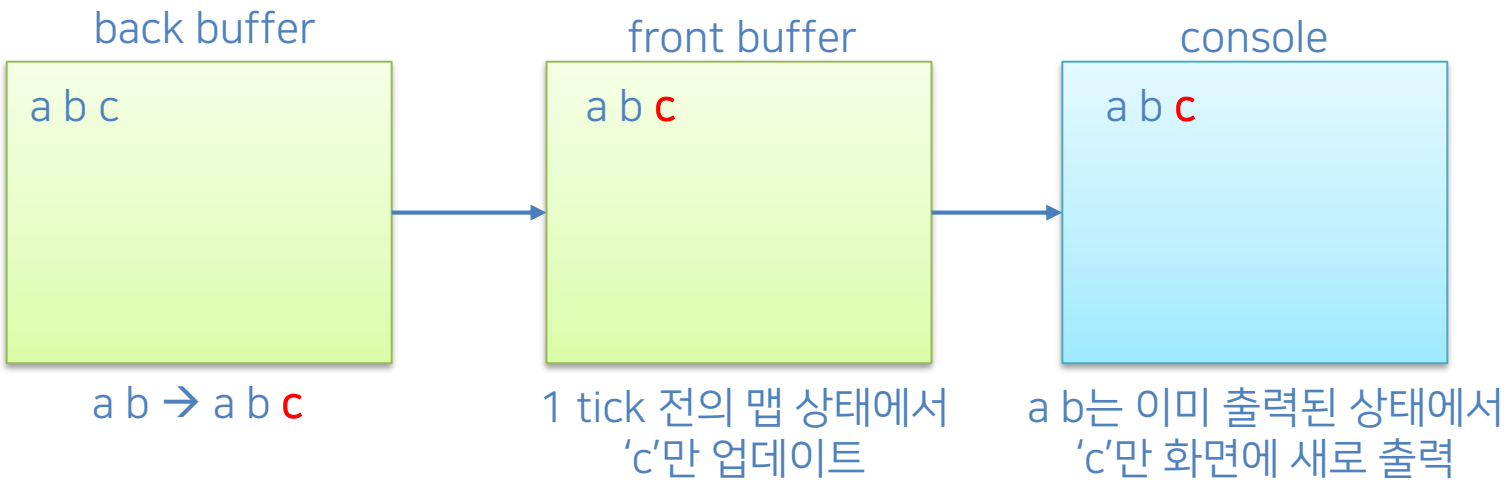
- **자료 구조**

```
typedef struct {
        int spice;      // 현재 보유한 스파이스
        int spice_max;  // 스파이스 최대 저장량
        int population; // 현재 인구 수
        int population_max;  // 수용 가능한 인구 수
} RESOURCE;
```

```
// 대강 만들어 봤음. 기능 추가하면서 각자 수정할 것
typedef struct {
        POSITION pos;               // 현재 위치(position)
        POSITION dest;              // 목적지(destination)
        char repr;                  // 화면에 표시할 문자(representation)
        int move_period;            // '몇 ms마다 한 칸 움직이는지'를 뜻함
        int next_move_time;         // 다음에 움직일 시간
} OBJECT_SAMPLE;
```

스켈레톤 코드

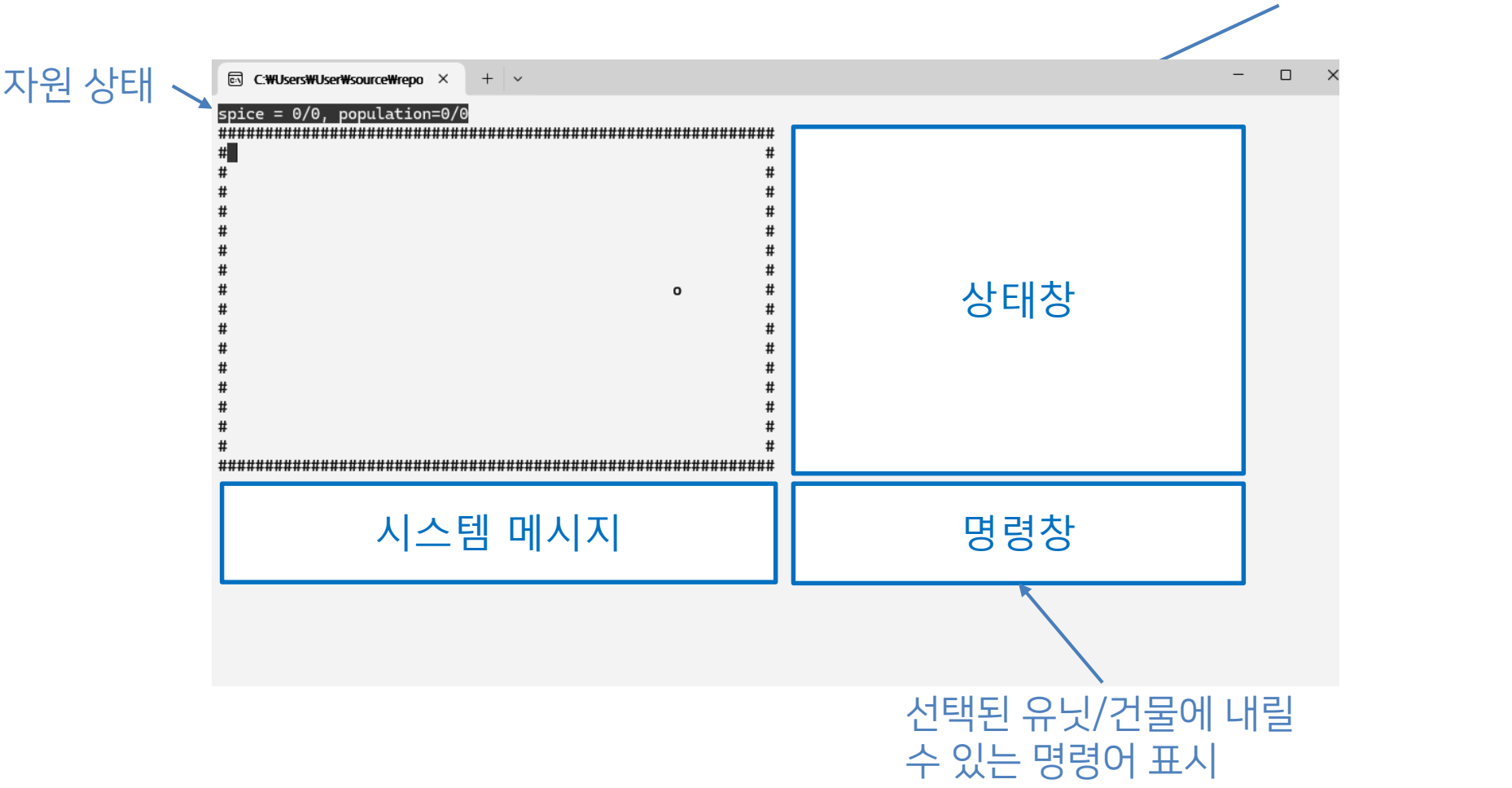
  - 게임화면 전체를 매번(예: 10ms마다) 다시 쓰면 화면이 깜박임
  - 입출력은 일반 연산보다 훨씬 느린 작업이기 때문
- **더블 버퍼링**
  - 화면 전체를 다시 출력하는 게 아니라 back buffer에 먼저 기록하고,
  - front buffer와 비교해서 다른 부분만 업데이트 & 출력한다.
    - 참고: https://codevang.tistory.com/39
    - 영상 압축에도 비슷한 원리가 활용됨(Key frame)



DUNE 1.5

1) 준비

- **화면 배치**
  - Intro(), Outro()는 알아서 꾸미기



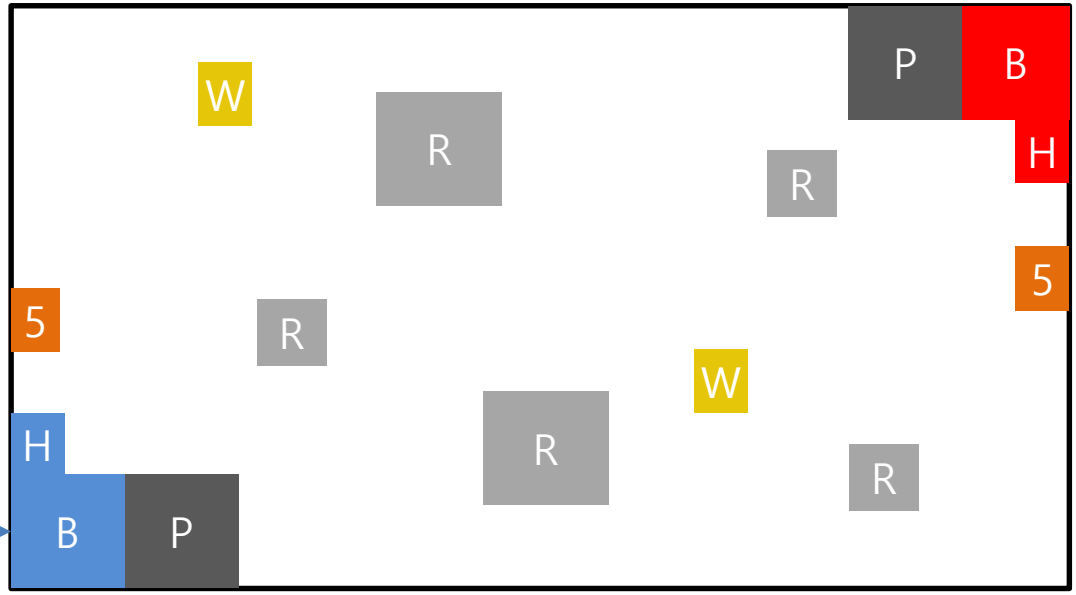
- **display()의 하위 함수 작성**

```
void display(
    RESOURCE resource,
    char map[N_LAYER][MAP_HEIGHT][MAP_WIDTH],
    CURSOR cursor)
{
    display_resource(resource);
    display_map(map);
    display_cursor(cursor);
    // display_system_message()
    // display_object_info()
    // display_commands()
    // ...
}
```

예시일 뿐임. 수정 가능

초기 상태

- **표시**
  - 아트레이디스(플레이어)는 푸른색, 하코넨(AI)은 빨간색, 샌드웜은 황토색, 장판은 검은색, 스파이스는 주황색, 기타 지형은 회색
  - **B**: 본진(Base), **W**: 샌드웜, **H**: 하베스터, **S**: 스파이스 매장지(매장량 표시. 1~9), **P**: 장판(Plate), **R**: 바위(Rock)
- **초기 배치**
  - 각각 좌하단, 우상단에 1 본부(2x2), 1 하베스터, 가까운 스파이스(5), 2x2 장판
  - 중립 샌드웜 2, 1X1/2X2 바위 적당히 배치
  - 건물, 지형(B, P, S, R)은 layer 0, 유닛(H, W)은 layer 1
- **지형**
  - **사막**: 기본 지형(빈 칸). 건물을
  - 지을 수 없음
  - **장판(2x2)**: 장판 위에 건설 가능
  - **바위**: 샌드웜은 통과할 수 없음



건물 아래는 장판
(부서지면 장판(P)이 노출됨)

건물 & 유닛 목록

- **건물 단축키 확인**

진영	건물	설명	건설비용	내구도	명령어
공통	본진(Base)		없음	50	H: 하베스터 생산
	장판(P: Plate)	건물 짓기 전에 깔기	1	없음	없음
	숙소(D: Dormitory)	인구 최대치 증가(10)	2	10	없음
	창고(G: Garage)	스파이스 보관 최대치 증가(10)	4	10	없음
아트레이디스	병영(B: Barracks)	보병 생산	4	20	보병 생산(S: Soldier)
	은신처(S: Shelter)	특수유닛 생산	5	30	프레멘 생산(F: Fremen)
하코넨	투기장(A: Arena)	투사 생산	3	15	투사 생산(F: Fighter)
	공장(F: Factory)	특수유닛 생산	5	30	중전차 생산(T: heavy Tank)

	유닛	생산비용	인구수	이동주기	공격력	공격주기	체력	시야	명령어
공통	하베스터	5	5	2000	없음	없음	70	0	H: Harvest, M: Move
아트레이디스	프레멘	5	2	400	15	200	25	8	M: 이동, P: 순찰
	보병	1	1	1000	5	800	15	1	
하코넨	투사	1	1	1200	6	600	10	1	
	중전차	12	5	3000	40	4000	60	4	
중립	샌드웜	없음	없음	2500	무한대	10000	무한대	무한대	없음

2) 커서 & 상태창

- 커서는 방향키로 이동
- 투명함: 지형/유닛은 그대로 두고 색깔로만 표시

- **방향키 더블클릭**
  - 각 방향키를 짧게 두 번 누르면 여러 칸 이동(거리는 적당히 조절하기)
- **선택(스페이스바)**
  - 오브젝트 위에서 스페이스바를 누르면 선택하고, 상태 & 명령 표시
    - 예) 빈 지형을 선택하면 상태창에 '사막 지형'의 정보를 표시할 것
  - 오브젝트가 선택된 상태에서 다른 오브젝트를 선택하면 즉시 대상 전환
- **취소(ESC 키)**
  - 선택 취소
  - 상태창 비우기

3) 중립 유닛

어떻게 구현했는지
README에 설명

- **샌드웜**
  - 동족(샌드웜)을 제외한, 가장 가까운 유닛을 향해 천천히 움직임
  - 일반 유닛을 만나면 잡아 먹음
  - 가끔 배설을 함 → 스파이스 매장지 생성
    - 생성 주기, 매장량은 랜덤. 플레이해 보면서 적절히 밸런싱할 것
- **Bonus1)**
  - 뱀 게임처럼 유닛을 잡아먹으면 길어지고, 배설을 하면 짧아지도록 만들기
    - 참고: https://kocaine.tistory.com/entry/c%EC%96%B8%EC%96%B4-%EB%B1%80%EA%B2%8C%EC%9E%84-%EC%86%8C%EC%8A%A4%EC%BD%94%EB%93%9C
- **Bonus2)**
  - Layer 2(공중) 추가
    - layer 2의 공중 유닛/오브젝트는 layer 1 오브젝트 위에 겹쳐질 수 있음
    - 사막독수리 구현: 목적 없이 날아다니는 중립 유닛
    - 모래 폭풍 구현(2x2)
      - 일정 주기로 생성되고, 빠르게 돌아다니다 사라짐, 지나치는 유닛 소멸, 건물 반파
    - 채점해야 하니까 처음 시작할 때 독수리, 모래폭풍 한 개씩 미리 만들어 두기

4) 유닛 1기 생산

- **명령어(단축키)**
  - 모든 단축키는 대소문자 무관
  - 아군 건물을 선택하면
  - 선택된 건물 종류에 따라서
  - 명령창에 명령어를 표시하고,
  - 해당 키가 입력되면 생산 시작
  - 잘못된 키 입력 시 선택 상태 유지,
  - ESC 키로 선택 취소
- **예)**
  - 본진(B)을 선택하면 상태 & 명령어 표시
  - ‘H’를 누르면 (자원이 충분하면) 즉시 본진 옆에 하베스터 1기 생산
  - 시스템 메시지 출력
    - 예) “A new harvester ready” / “Not enough spice”
- **Bonus)**
  - 생산에 일정 시간이 걸림 & 선택하면 남은 시간 표시
  - ‘x’로 취소 가능

5) 시스템 메시지

- **형식: 메시지 로그**
  - 창 크기는 고정되어 있고,
  - 메시지가 쌓이면 스크롤이 올라감
  - 즉 오래된 메시지는 위로 올라가다가 사라지고, 최신 메시지는 가장 아래에 표시

- **시스템 메시지를 출력하는 경우 예시)**
  - 스타크래프트 참고
  - 특정 작업을 직접 지시한 경우
    - 예) 하베스터가 스파이스 채취를 시작합니다. XXX Ready! 등등
  - 수행할 수 없는 명령을 입력한 경우
    - 예) 사막 지형에 건물 짓기, 건물 위로 이동하기, 생산 자원이 모자란 경우, 인구 수용량이 모자란 경우 등등
  - 아군 유닛이 공격당한 경우 등등

- **메시지 텍스트**
  - 알아서 적당히…

6) 건설

- **커서 변환**
  - 2x2 장판/건물 건설 시 2x2 크기로 변환, 취소하거나 건설 시작하면 다시 1x1
  - 맵을 벗어나지 않도록 boundary check

- **아무것도 선택하지 않은 상태에서는**
  - 명령창에 B: Build 표시
  - 'B' 키를 입력하면 명령창에 건설 가능한 건물 목록
  - 단축키로 선택, 또는 ESC 키로 취소
  - 건물을 선택하면 건물 크기에 맞게 커서 크기를 조절하고, 커서를 움직여서 스페이스바를 한번 더 누르면 즉시 건설
  - **장판이 먼저 깔려 있어야 그 위에 다른 건물 건설 가능

- **Bonus)**
  - 생산에 일정 시간이 걸림 & 건설 중이면 깜박임 & 선택하면 남은 시간 표시
  - 'x'로 취소 가능

7) 유닛 목록 구현

- **유닛 목록을 배열, 구조체, 연결리스트 등으로 구현**
  - 보유 가능한 유닛 수의 최댓값은 supply 최댓값과 같음

8) 유닛 행동-하베스터

- **각 유닛은**
  - 마지막으로 내려진 명령을 계속 수행, 임무 완료 시 제자리 대기
- **하베스터**
  - 공격능력 없음. 느림.
  - 명령어: H: Harvest(수확), M: move(이동)
- **예)**
  - 하베스터(H) 선택 → 상태창, 명령어 표시
    - 상태창에는 이름, 체력 등 + 간단한 그림, 명령어는 H: Harvest
  - → 'H' 키 입력(잘못된 키 입력 시 선택 상태 유지)
  - → 스파이스 매장지 선택(잘못된 위치 선택 시 'H' 명령 입력 상태 유지)
  - → 매장지에서 스파이스를 수확
    - 스파이스 매장지에서 수확에 걸리는 시간은 적당히 조절할 것(3~5초 정도?)
    - 다른 하베스터가 있으면 차례를 기다림
  - → 다른 명령이 없으면 수확한 스파이스를 본진에 배달, 매장지←→본진 왕복
    - 한번에 수확하는 스파이스는 2~4
    - 본진에 배달하면 보유 자원량 증가(화면에 표시)
    - 단 창고 수용량을 초과한 스파이스는 버려짐
    - 매장량이 떨어지면 본진 옆에서 대기

9) 유닛 행동-보병, 특수유닛

- **각 유닛은**
  - 마지막으로 내려진 명령을 계속 수행
  - 임무 완료 시 제자리 대기

- **일반 명령**
  - M: 이동 (Move)
    - 포인트를 지정하면 그 곳으로 이동
    - 건물을 지정하면 건물 앞으로 이동해서 공격
  - P: 순찰(Patrol)
    - 포인트를 지정하면 현재 위치와 지정한 위치를 왕복

10) 전투

- **적 유닛이 시야 안에 들어오면**
  - 대기(임무 없음)/이동/순찰 중, 적 유닛이 시야에 들어오면
  - 인접한 칸로 이동 후 전투 시작
  - 공격 주기마다 공격력만큼 공격
  - 적 체력 감소
  - 체력 0이 되면 파괴
  - 유닛 특수능력이 있으면 사용

- **Bonus)**
  - 유닛 사정거리(<시야) 각각 다르게 설정, 원거리 전투 구현, 총알 표현

11) AI

- **하코넨 진영 스크립트 작성**
  - 어떤 행동 원리를, 어떻게 구현했는지 github README 파일에 설명
  - 지능(?) 수준에 따라 평가
  - AI가 내리는 명령들은 시스템 메시지를 출력하지 않음

- **예)**
  - 건물, 유닛 초기에 모두 배치, 건설/생산 없음, 유닛 행동 x
  - 건물, 유닛 초기에 모두 배치, 건설/생산 없음, 유닛은 랜덤하게 행동
  - 건물, 유닛 초기에 초기 배치, 건설/생산 없음, 유닛에 단순한 명령을 내림
  - 건설/생산 명령을 적절히 내림, 유닛은 랜덤하게 행동
  - 건설/생산을 내림, 유닛에 단순한 명령을 내림
  - 플레이어 유닛의 움직임에 따라 전술적 명령을 내림
  - 플레이어의 움직임에 따라 전술적 명령을 내림
  - 스크립트에 따라 전략적 명령을 내림
  - ML/Deep Learning으로 학습시킨 모델을 적용해서 명령을 내림

추가 기능은 언제나 환영

- **추가된 기능 적어 주기**

- **밸런스도 적당히 맞추기**
  - 채집, 전투가 너무 길어지지 않도록
  - 경향성만 지켜서: 아트레이디스(싸고 빠름), 하코넨(비싸고, 느리고, 강함)

- **문제 설명이 애매한 부분은**
  - 자의적으로 판단해서 설정하고,
  - engine.c 상단 주석 또는 README 파일에 기재