

2016년도 포카전 AI게임

<포카워치>

버전

1.0.0.0

개발자

포스텍 컴공 14 최유정

포스텍 컴공 14 황일환

내용

개요.....	4
구성요소	4
필드와 점령지	4
팀	4
클래스.....	4
유닛	4
탄체	4
클래스	5
컴공과(DEP_CSE)	5
물리과(DEP_PHYS)	5
생명과(DEP_LIFE)	6
기계과(DEP_ME)	7
화학과(DEP_CHEM)	7
규칙.....	8
우선순위	8
공격	8
이동	8
리스폰	8
점령지.....	9
규칙 순서.....	9
프로그래밍 길라잡이	10
데이터 구조.....	10
protocol_data	10
protocol_unit.....	11
protocol_poison.....	12
protocol_petal	12
protocol_mushroom	13
명령	13
protocol_command.....	13
구현 흐름	14

초기 설정	14
턴	15
주의사항	15
시간초과	15
부록	16
protocol.h의 매크로와 함수들	16
inline int team_to_index(protocol_team t)	16
inline protocol_team index_to_team(int i)	16
inline protocol_team team_invert(protocol_team t)	16
inline int direction_to_dx(protocol_direction d)	16
inline int direction_to_dy(protocol_direction d)	16
inline bool state_kind_attack(protocol_state s)	16
inline bool state_kind_skill(protocol_state s)	16
inline protocol_direction state_to_direction(protocol_state s)	16
inline protocol_command direction_to_attackcommand(protocol_direction d)	16
inline protocol_command direction_to_skillcommand(protocol_direction d)	16
inline protocol_command dep_to_spawncommand(protocol_dep dep)	16
inline protocol_direction direction_flip(protocol_direction d)	16
inline protocol_direction direction_mirror(protocol_direction d)	16
inline void print_data(const protocol_data& d)	16
inline protocol_direction random_direction()	16
inline protocol_dep random_dep()	16
알림	17
버그리포트	17
업데이트와 관련하여	17

개요

본 게임은 포카전 시종목 게임으로써, 오버워치의 '점령전'을 모토로 하여 만들어진 게임이다.

구성요소

필드와 점령지

필드는 가로 21칸(MAP_WIDTH), 세로 15칸(MAP_HEIGHT)으로 이루어져 있다.

필드의 가운데에는 가로 5칸(POINT_X2 - POINT_X1), 세로 7칸(POINT_Y2 - POINT_Y1) 크기의 점령지가 존재한다.

팀

팀은 POSTECH팀(Team_POSTECH)과 KAIST팀(Team_KAIST)이 있다.

각 팀은 필드 상의 왼쪽, 오른쪽에서 시작하게 된다.

클래스

총 5개의 클래스가 있다.

각 클래스의 특징은 아래에 더 자세히 설명되어 있다.

유닛

각 팀은 3개의 유닛(UNIT_PER_TEAM)을 가진다.

각 팀은 3개의 유닛에 임의의 클래스를 배정하여 소환할 수 있다.

유닛은 필드 격자 위의 임의의 공간 위에 있을 수 있다. 단, 서로 겹칠 수는 없다.

유닛은 이동하거나 공격할 수 있다. 이에 대한 특성은 클래스에 따라 상이하다.

영웅 유닛의 경우 기술을 한 번 사용할 수 있다.

탄체


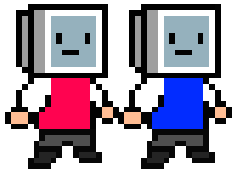

유닛의 공격이나 기술에 의해 탄체가 생겨날 수 있다.

탄체는 필드 격자 위의 임의의 공간 위에 있을 수 있다.

탄체는 유닛과 상호작용하여 피해나 치유를 줄 수 있다.



클래스

컴공과(DEP_CSE)


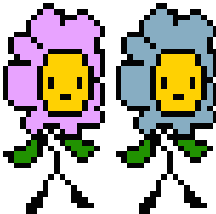
스플래시 이미지		
게임 내 이미지		
체력		
특징	스파크	매 10턴(CSE_SPARK_COOLTIME)마다 주변 1칸을 3턴(CSE_SPARK_STUN)동안 스톤시키는 전기 충격을 발사한다.
공격	블링크	공격 방향으로 4칸(CSE_BLINK_LENGTH) 즉시 이동한다. 쿨타임은 10턴(CSE_BLINK_COOLTIME)이다.
기술	스톰	자신 주변 3칸(CSE_STORM_RANGE) 이내의 적을 5턴(CSE_STORM_STUN)동안 스톤시킨다. 범위는 정사각형 모양이다.

물리과(DEP_PHYS)


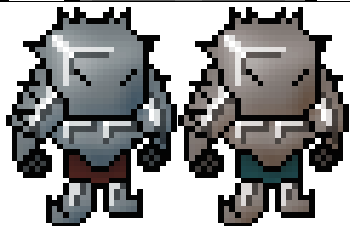
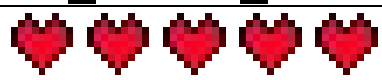
스플래시 이미지		
----------	--	--

게임 내 이미지		
체력		
공격	파동	공격 방향으로 있는 모든 적에게 1의 피해(PHYS_WAVE_DAMAGE)를 준다. 쿨타임은 4턴(PHYS_WAVE_COOLTIME)이다.
기술	블랙홀	기술 방향으로 가로 3칸 세로 3칸에 있는 모든 적들에게 10의 피해(PHYS_BLACKHOLE_DAMAGE)를 준다.

생명과(DEP_LIFE)

스플래시 이미지		
게임 내 이미지		
체력		
공격	꽃잎	공격 방향으로 꽃잎을 발사한다. 꽃잎은 발사된 방향으로 1턴에 1칸 움직인다. 아군 유닛이 꽃잎에 맞으면 1의 치유(LIFE_PETAL_HEAL)를 받고 스테인이 풀리며, 적군 유닛이 꽃잎에 맞으면 1의 피해(LIFE_PETAL_DAMAGE)를 받는다. 꽃잎이 유닛에 맞으면 그 자리에서 꽃잎은 사라진다. 쿨타임은 8턴(LIFE_PETAL_COOLTIME)이다.
기술	만개	모든 아군 유닛에게 10의 치유(LIFE_BLOSSOM_HEAL)를 주고 스테인을 푼다.

기계과(DEP_ME)

스플래시 이미지		
게임 내 이미지		
체력		
특징	가시	이 유닛 주변 1칸 안에 있는 모든 적에게 1의 피해 (ME_THORN_DAMAGE)를 준다.
		이 유닛은 2턴에 1칸만 움직일 수 있다.
공격	이 유닛은 공격할 수 없다.	
기술	교통사고	기술 방향으로 유닛이나 필드 끝에 부딪힐 때 까지 이동한다. 유닛에 부딪혀서 멈췄을 경우 그 유닛은 10의 피해(ME_ACCIDENT_DAMAGE)를 받는다.

화학과(DEP_CHEM)

스플래시 이미지		
----------	--	--

게임 내 이미지		
체력		
공격	독	공격 방향으로 4칸(CHEM_POISON_LENGTH)에 독을 방사한다. 독을 밟은 적군 유닛은 1의 피해(CHEM_POISON_DAMAGE)를 입는다. 독은 4턴(CHEM_POISON_SPAN)이 지나야 사라지며, 쿨타임은 6턴(CHEM_POISON_COOLTIME)이다.
기술	버섯	기술 방향으로 적군 유닛이 밟았을 때 10의 피해(CHEM_MUSHROOM_DAMAGE)를 주는 버섯을 하나 소환한다. 버섯은 적군 유닛에게 밟히는 순간 사라진다.

규칙

우선순위

우선순위는 교착 상황에서 어떤 유닛이 우선순위를 가지는가에 대한 규칙이다.

각 팀은 매 턴마다 다른 우선순위를 가진다.

POSTECH팀은 짝수 턴에, KAIST팀은 홀수 턴에 우선순위를 가진다.

같은 팀이라면 번호가 작은 유닛이 우선순위를 가진다.

공격

쿨타임이 다 지나기 전에는 공격을 다시 할 수 없다.

기술은 쿨타임과 관계 없이 사용할 수 있다.

이동

유닛에게 내려진 명령이 이동이라면 유닛은 해당 방향으로 한 칸 움직인다.

단, 기계과 유닛은 두 턴마다 한 칸 움직일 수 있다.

유닛은 필드 밖으로 나가도록 움직일 수 없다.

유닛은 서로 겹치도록 움직일 수 없다.

유닛은 서로를 통과하여 움직일 수 없다.

두 유닛이 한 자리로 움직이려고 할 때, 우선순위가 높은 유닛만이 그 자리로 가게 된다.

리스폰

유닛은 클래스마다 다른 체력을 가지며, 체력이 0이 되면 유닛은 죽게 된다.

유닛이 죽으면 10턴의 리스폰 시간(**RESPAWN_COOLTIME**) 후 다시 원래 위치에서 생성된다.

리스폰 시간이 다 되면 유닛은 원래 처음 생성되었던 자리에서 리스폰 된다.

리스폰 될 때 클래스는 최후로 받았던 리스폰 명령에 따른다.

리스폰 되는 순간 리스폰 되는 자리에 다른 유닛이 있을 경우 그 유닛은 즉사한다.

리스폰이 되고 나서 5턴(**INVINCIBLE_SPAN**) 동안은 무적상태가 된다.

매 5번째 리스폰(**HERO_PERIOD**) 마다 리스폰된 유닛은 영웅이 된다.

점령지

점령지의 소유자가 없는 상태에서 점령지에 한 팀의 유닛만 올라가 있다면 그 팀의 점령 게이지가 차게 된다.

점령지의 소유자가 어떤 팀인 상태에서 그 팀이 아닌 유닛만 점령지에 올라가 있다면 점령지에 올라간 팀의 점령 게이지가 차게 된다.

점령지의 점령 게이지를 모두 채우면(**POINT_TURN_OWN**) 점령 게이지를 모두 채운 팀이 점령지를 소유하게 된다.

점령지를 소유하고 있다면 승리 게이지가 차게 된다.

승리 게이지를 모두 채우고(**POINT_TURN_WIN**) 추가시간이 없다면 게임에서 승리하게 된다.

승리 게이지가 가득 찬 순간에 점령지의 소유자가 아닌 팀이 점령지에 올라와 있다면 추가시간(**POINT_TURN_EXTRA**)이 주어진다.

추가시간은 점령지의 소유자가 아닌 팀이 점령지 위에 있다면 계속 주어진다.

추가시간은 점령지의 소유자가 아닌 팀이 점령지 위에 없다면 줄어든다.

승리 여부에 상관 없이 300턴(**TURN_MAX**)이 지나면 게임은 종료된다. 이 경우 턴이 종료되는 순간 점령지를 소유하고 있는 팀이 승리하게 된다.

규칙 순서

규칙은 매 턴마다 명령을 조합하여 평가되며, 규칙이 적용되는 순서는 다음과 같다. 특별한 명시가 없는 한 한 규칙 순서 안에서의 규칙 적용은 동시에 일어난다. 예를 들면, 8번 규칙 순서를 적용할 때 두 물리 과 유닛이 서로를 바라보고 공격을 해서 서로에게 피해를 주고 동시에 죽을 수도 있다는 뜻이다. *표시가 있는 규칙 순서에서는 그 규칙을 적용했을 때 죽는 유닛이 발생할 수 있으며, 죽은 유닛은 다음 규칙 순서로 넘어가지 않음을 뜻한다.

1. 우선순위
2. 각 유닛에게 명령 전달
3. 이동
4. 컴공 공격에 의한 이동

5. *기계 기술에 의한 이동
6. *리스폰
7. *기술
8. *공격 (컴공의 스파크, 기계의 가시 포함)
9. *탄체 충돌 판정 (꽃잎, 독, 버섯)
10. 점령지

프로그래밍 길라잡이

데이터 구조

AI를 구현할 때에는 ai.cpp에 있는 다음 두 함수 안에 작성하면 된다.

```
void Ai::aiInit(void)
void Ai::ai(protocol_data info)
```

여기서 protocol_data는 현재 턴에 대한 정보를 담고 있으며, 이 정보를 참고해서 다음 턴에 내릴 결정을 하면 된다. 필요한 구조체와 각종 매크로에 대한 정의는 protocol.h에서 확인할 수 있다.

protocol_data

```
typedef struct {
    protocol_unit unit[UNIT_NUM_MAX];
    protocol_poison poison[POISON_NUM_MAX];
    protocol_petal petal[PETAL_NUM_MAX];
    protocol_mushroom mushroom[MUSHROOM_NUM_MAX];
    protocol_team owner;
    int own[TEAM_NUM_MAX];
    int win[TEAM_NUM_MAX];
    int extra;
    int elapsed;
} protocol_data;
```

unit	유닛에 대한 정보를 담고 있다. 인덱스 상으로 0, 1, 2는 TEAM_POSTECH, 3, 4, 5는 TEAM_KAIST이다.
poison	독 탄체에 대한 정보를 담고 있다.
petal	꽃잎 탄체에 대한 정보를 담고 있다.
mushroom	버섯 탄체에 대한 정보를 담고 있다.
owner	현재 점령지를 점령한 팀이 누구인지 나타낸다. 아무도 점령하지 않았다면 TEAM_NULL이다.
own	점령 게이지 수치를 나타낸다. 인덱스 상으로 0은 TEAM_POSTECH의 것이고, 1은 TEAM_KAIST의 것이다.
win	승리 게이지 수치를 나타낸다. 인덱스 상으로 0은 TEAM_POSTECH의 것이고, 1은 TEAM_KAIST의 것이다.
extra	추가 시간 수치를 나타낸다.

elapsed	게임 시작으로부터 몇 턴이 지났는지를 나타낸다.
---------	----------------------------

protocol_team

TEAM_NULL	아무 의미도 가지지 않는다.
TEAM_POSTECH	POSTECH팀
TEAM_KAIST	KAIST팀

protocol_unit

<pre>typedef struct { protocol_team team; protocol_dep dep; int x, y; protocol_state state; int health; bool hero; int cooltime; int respawn; int stun; int invincible; } protocol_unit;</pre>	
team	유닛의 팀을 나타낸다.
dep	유닛의 클래스를 나타낸다.
x, y	유닛의 물리적 위치를 나타낸다. 왼쪽 아래를 0,0 으로 생각한다.
state	유닛의 상태를 나타낸다.
health	유닛의 체력을 나타낸다.
hero	유닛이 영웅인지 아닌지를 나타낸다.
cooltime	유닛의 쿨타임을 나타낸다. 0보다 클 때엔 공격할 수 없다.
respawn	유닛의 리스폰 시간을 나타낸다.
stun	유닛의 스톤 시간을 나타낸다. 0보다 클 때엔 어떤 명령도 내릴 수 없다.
invincible	유닛의 무적 시간을 나타낸다. 0보다 클 때엔 어떤 피해도 받지 않는다.

protocol_dep

DEP_NULL	아무 의미도 가지지 않는다.
DEP_CSE	컴공과
DEP_PHYS	물리과
DEP_LIFE	생명과학
DEP_ME	기계과
DEP_CHEM	화학과

protocol_state

STATE_NULL	아무 의미도 가지지 않는다.
------------	-----------------

STATE_IDLE	유닛이 아무것도 하지 않음을 뜻한다. 움직이는 명령 역시 유닛을 이 상태로 만든다.
STATE_DEAD	유닛이 죽어있음을 뜻한다. 유닛이 이 상태일 경우 유닛의 클래스와 리스폰 시간 외의 다른 정보는 아무런 의미도 가지지 않는다.
STATE_ATTACK_RIGHT STATE_ATTACK_UP STATE_ATTACK_LEFT STATE_ATTACK_DOWN	유닛이 특정 방향으로 공격을 행하고 있음을 나타낸다.
STATE_SKILL_RIGHT STATE_SKILL_UP STATE_SKILL_LEFT STATE_SKILL_DOWN	유닛이 특정 방향으로 기술을 사용하고 있음을 나타낸다.
STATE_STUN	유닛이 스톤에 걸려있음을 나타낸다.

protocol_poison

<pre>typedef struct { bool valid; protocol_team team; int x, y; int span; } protocol_poison;</pre>	
valid	독이 실제로 존재하는지를 나타낸다. false일 경우 구조체에 들어있는 값은 아무 의미도 가지지 않는다.
team	독이 어느 팀에서 만들어졌는지를 나타낸다.
x, y	독의 물리적 위치를 나타낸다.
span	독의 존재 시간을 나타낸다.

protocol_petal

<pre>typedef struct { bool valid; protocol_team team; int x, y; protocol_direction direction; } protocol_petal;</pre>	
valid	꽃잎이 실제로 존재하는지를 나타낸다. false일 경우 구조체에 들어있는 값은 아무 의미도 가지지 않는다.
team	꽃잎이 어느 팀에서 만들어졌는지를 나타낸다.
x, y	꽃잎의 물리적 위치를 나타낸다.
span	꽃잎이 어느 방향으로 진행하고 있는지를 나타낸다.

protocol_direction

DIRECTION_NULL	아무 의미도 가지지 않는다.
DIRECTION_RIGHT	오른쪽

DIRECTION_UP	위
DIRECTION_LEFT	왼쪽
DIRECTION_DOWN	아래

protocol_mushroom

<pre>typedef struct { bool valid; protocol_team team; int x, y; } protocol_mushroom;</pre>	
valid	버섯이 실제로 존재하는지를 나타낸다. false일 경우 구조체에 들어있는 값은 아무 의미도 가지지 않는다.
team	버섯이 어느 팀에서 만들어졌는지를 나타낸다.
x, y	버섯의 물리적 위치를 나타낸다.

명령

다음 턴에 내릴 결정은 다음과 같은 함수로 할 수 있다. 한 유닛에게는 단 하나의 명령만 내릴 수 있으며, 가장 나중에 내려진 명령이 전달된다. 또한, 규칙을 위반하는 명령은 적용되지 않는다. (스턴 상태의 유닛에게 이동 명령, 영웅이 아닌 유닛에게 기술 사용 명령 등)

<pre>void Ai::move(int i, protocol_direction x) void Ai::attack(int i, protocol_direction x) void Ai::skill(int i, protocol_direction x) void Ai::spawn(int i, protocol_dep x) void Ai::command(int i, protocol_command x)</pre>	
move	i번째 유닛에게 x방향으로 이동하라는 명령을 내린다.
attack	i번째 유닛에게 x방향으로 공격하라는 명령을 내린다.
skill	i번째 유닛에게 x방향으로 기술을 사용하라는 명령을 내린다.
spawn	i번째 유닛에게 x클래스로 리스폰 하도록 명령을 내린다. 이 명령은 유닛이 죽어있을 때에만 효과가 있다.
command	i번째 유닛에게 직접 명령을 내린다.

protocol_command

COMMAND_NULL	아무 의미도 가지지 않는다.
COMMAND_MOVE_RIGHT COMMAND_MOVE_UP COMMAND_MOVE_LEFT COMMAND_MOVE_DOWN	유닛에게 지정한 방향으로 이동하라는 명령을 내린다.
COMMAND_ATTACK_RIGHT COMMAND_ATTACK_UP COMMAND_ATTACK_LEFT COMMAND_ATTACK_DOWN	유닛에게 지정한 방향으로 공격하라는 명령을 내린다.
COMMAND_SKILL_RIGHT COMMAND_SKILL_UP	유닛에게 지정한 방향으로 기술을 사용하라는 명령을 내린다.

COMMAND_SKILL_LEFT COMMAND_SKILL_DOWN	
COMMAND_SPAWN_CSE COMMAND_SPAWN_PHYS COMMAND_SPAWN_LIFE COMMAND_SPAWN_ME COMMAND_SPAWN_CHEM	유닛에게 지정한 클래스로 리스폰하라는 명령을 내린다.

구현 흐름

초기 설정

초기 설정은 Ai::aiInit 에 들어가는 명령을 모두 포함한다. 초기 설정에는 피벗과 클래스 선택이 있다. 초기 설정 단계는 블라인드로, 상대에 대한 정보를 전혀 모른 채 결정을 내려야 한다.

피벗

인공지능을 개발할 때, 어느 팀을 기준으로 인공지능을 개발할지 선택할 수 있다. 인공지능 프로그램이 서버에 접속할 때 기준 팀과 다른 팀으로 지정되었다면 자동으로 명령과 입력이 뒤집혀서 들어간다. 피벗은 다음 함수로 선택할 수 있다. 명시하지 않으면 피벗은 POSTECH팀이 된다.

```
void Ai::setPivot(protocol_team t)
```

위에서 기술했듯이, POSTECH팀 기준으로 개발한다면 0, 1, 2번째 유닛을 자신 팀으로 생각하면 된다. 또한, 시작 위치는 왼쪽이라고 가정하여 개발하면 된다. 만약 KAIST팀을 기준으로 개발한다면, 3, 4, 5번째 유닛을 자신 팀으로, 시작위치를 오른쪽으로 생각하여 개발하면 된다. protocol_data의 owner, own, win 등도 같은 방법으로 읽으면 된다.

클래스 선택

게임을 시작할 때 어떤 클래스를 가지고 시작할지 선택한다. 다음 함수를 사용하면 된다.

```
void Ai::CharacterInit(int i, protocol_dep x)
```

반드시 자신 팀의 세 유닛의 클래스를 명시해주어야 한다. 명시하지 않았을 경우의 행동은 정의되어있지 않다.

예제

다음은 POSTECH팀 기준으로 처음에 기계과, 물리과, 화학과를 선택하는 코드이다.

```
void Ai::aiInit(void)
{
    // Example on POSTECH side
    setPivot(Team_POSTECH);

    CharacterInit(0, DEP_ME);
    CharacterInit(1, DEP_PHYS);
    CharacterInit(2, DEP_CHEM);
}
```

다음은 KAIST팀 기준으로 처음에 모든 유닛을 컴공과로 선택하는 코드이다.

```
void Ai::aiInit(void)
{
    // Example on KAIST side
```

```

        setPivot(TEAM_KAIST);

        CharacterInit(3, DEP_CSE);
        CharacterInit(4, DEP_CSE);
        CharacterInit(5, DEP_CSE);
    }

```

턴

초기 설정이 완료되면 게임이 시작되고, 매 턴 경기의 정보와 함께 서버가 클라이언트에게 다음 명령을 요구한다. 인공지능은 경기의 정보를 종합하여 자신 팀 세 유닛에게 명령을 내리면 된다. 위에서 언급된 데이터 구조와 명령을 참고하여 인공지능을 작성하면 된다.

예제

아래는 POSTECH팀 기준으로 다음과 같은 인공지능을 구현한 코드이다.

1. 중심과 멀리 있다면 일단 중심으로 간다.
2. 중심에 있다면 무작위 방향으로 움직이거나 공격한다.
3. 죽었다면 무작위 클래스로 리스폰한다.

```

void Ai::ai(protocol_data info)
{
    print_data(info);

    // Example of random AI on POSTECH side
    for (int i = 0; i < UNIT_PER_TEAM; i++) {
        if (info.unit[i].state == STATE_DEAD) {
            spawn(i, random_dep());
        }
        else if (info.unit[i].x < MAP_WIDTH / 2 - 2) {
            move(i, DIRECTION_RIGHT);
        }
        else if (info.unit[i].x >= MAP_WIDTH / 2 + 2) {
            move(i, DIRECTION_LEFT);
        }
        else {
            switch (rand() % 2) {
                case 0: attack(i, random_direction()); break;
                case 1: move(i, random_direction()); break;
            }
        }
    }
}

```

주의사항

시간초과

만약 인공지능의 처리 시간이 서버의 턴보다 길어지거나, 혹은 다른 네트워크적 이유로 인공지능의 다음 명령이 서버에 전송되지 않았을 때 서버는 이전 턴의 명령을 그대로 사용한다.

부록

protocol.h의 매크로와 함수들

```
inline int team_to_index(protocol_team t)
```

팀 정보를 인덱스로 바꾼다. TEAM_POSTECH이면 0, TEAM_KAIST이면 1을 반환한다. protocol_data의 own, win 등을 참조할 때 사용한다.

```
inline protocol_team index_to_team(int i)
```

인덱스를 팀 정보로 바꾼다. 위 함수의 역함수이다.

```
inline protocol_team team_invert(protocol_team t)
```

팀 정보를 뒤집는다. TEAM_POSTECH이면 TEAM_KAIST를, TEAM_KAIST이면 TEAM_POSTECH을 반환한다.

```
inline int direction_to_dx(protocol_direction d)
```

방향 정보를 토대로 x좌표 값을 반환한다. 왼쪽일 경우 -1, 오른쪽일 경우 1, 그 외의 경우는 0이다.

```
inline int direction_to_dy(protocol_direction d)
```

방향 정보를 토대로 y좌표 값을 반환한다. 아래일 경우 -1, 위일 경우 1, 그 외의 경우는 0이다.

```
inline bool state_kind_attack(protocol_state s)
```

상태가 공격하는 상태인지를 참/거짓으로 반환한다.

```
inline bool state_kind_skill(protocol_state s)
```

상태가 기술을 사용하는 상태인지를 참/거짓으로 반환한다.

```
inline protocol_direction state_to_direction(protocol_state s)
```

상태에서 방향 정보를 추출한다. (STATE_ATTACK_RIGHT, STATE_SKILL_LEFT 등) 방향 정보가 없는 상태일 경우 DIRECTION_NULL을 반환한다.

```
inline protocol_command direction_to_attackcommand(protocol_direction d)
```

방향 정보를 공격 명령으로 바꾼다. DIRECTION_RIGHT을 넣으면 COMMAND_ATTACK_RIGHT을 반환한다.

```
inline protocol_command direction_to_skillcommand(protocol_direction d)
```

방향 정보를 기술 사용 명령으로 바꾼다.

```
inline protocol_command dep_to_spawncommand(protocol_dep dep)
```

클래스 정보를 클래스 리스폰 명령으로 바꾼다.

```
inline protocol_direction direction_flip(protocol_direction d)
```

방향을 뒤집는다. 오른쪽은 왼쪽으로, 위는 아래로 바뀐다.

```
inline protocol_direction direction_mirror(protocol_direction d)
```

방향을 반전한다. 오른쪽과 왼쪽만 바뀌고 위, 아래는 바뀌지 않는다.

```
inline void print_data(const protocol_data& d)
```

주어진 protocol_data를 cmd에 출력한다.

```
inline protocol_direction random_direction()
```

4방향 중 임의의 방향을 반환한다.

```
inline protocol_dep random_dep()
```

5개 클래스 중 임의의 클래스를 반환한다.

알림

버그리포트

최유정(yjchoi0606@postech.ac.kr)

항일환(dlf0325@postech.ac.kr)

업데이트와 관련하여

본 게임 특성상, 유닛 밸런스 조정과 디버그를 위해 상기된 값들(쿨타임, 피해량 등)에 대한 몇 번의 업데이트가 있을 수 있습니다. 또한, 그래픽과 사운드가 미완성이므로 리소스 관련 업데이트도 있을 예정입니다. 이 부분을 숙지하여 착오 없으시길 바랍니다.