Compreheisive Client AI実装ガイド

本文書について

本文書は、CC[1]を用いてheis対戦用AIを実装するためのガイドです。

必要な前提知識

フィールド上のユニットを操作するための関数・データなどにはC++のSTL(以下参照)が使用されています。
したがって、これらSTLの知識があると、よりスムーズにAIを実装できます。

- std::string: 文字列
- std::vector: 可変長の配列

実装対象

AIは、CC内のクラスで表現されています。
そのクラスを実装することで、AIを実装できます。

実装対象のクラス名とファイル名を以下に記載します。

- CUserAI: HeisClient_CC/src/AI/user_AI.[cpp, h]
- COpponentAI: HeisClient_CC/src/AI/opponent_AI.[cpp, h] (ローカルモードを利用する場合)

メンバ変数

各実装対象クラス内には、デフォルトで"m_commander"というメンバ変数が登録されております。
フィールド上のユニットは、このメンバ変数を通じて操作されますので、削除はしないでください。

その他、必要に応じてメンバ変数を追加してもかまいません。

処理の流れ

ユーザは、自チームの各ユニットがターンごとに取る行動を実装する必要があります。
すべての行動は、"AI_main"というメンバ関数に記述する必要があります。

"AI_main"関数は、C言語やC++のmain関数のようなものと思っていただいて構いません。
自チームのターンが来ると"AI_main"関数が実行され、関数を抜けるとターンが終了します。

以下に、実装例として「可能な限りすべてのユニットを行動させる」という処理のひな型を記載します。

```
void CUserAI::AI_main(const JSONRecvPacket_Board& board_pkt)
{
    std::string my_team_name = board_pkt.turn_team;
    while (m_commander->get_all_actable_infantry_ids(my_team_name).size() != 0)
{
        // 各ユニットの行動を記述
    }
}
```

利用できる関数

void AI_main(const JSONRecvPacket_Board& board_pkt)

AIのメイン処理です。
前章で述べた通り、1ターンで行うすべての行動をこの関数内に記述します。

引数board_pktには、ターン開始時にサーバから送信される「盤面」JSONに含まれている情報が含まれている構造体です。
「盤面」JSONについては、heisで通信するJSONの仕様書をご覧ください。

なお、board_pktのメンバ名はすべて「盤面」JSONの各フィールド名と合わせてあります。
また、board_pktの各メンバの型も「盤面」JSONの各フィールドの型と対応させてあり、

- 数値 -> int型、double型などの数値型
- 文字列 -> std::string型
- 真偽値 -> bool型
- オブジェクト -> 構造体
- 配列 -> std::vector型

の対応関係があります。

以下、すべてm_commanderから呼び出す関数です。

std::vector<std::string> get_all_movable_infantry_ids(const std::string& team_name)

std::vector<std::string> get_all_attackable_infantry_ids(const std::string& team_name)

std::vector<std::string> get_all_actable_infantry_ids(const std::string& team_name)

それぞれ、現時点で移動・攻撃・行動(移動 or 攻撃)可能な兵士のIDを検索する関数です。
条件に該当する兵士がいればそれらのIDが含まれた配列が返され、いない場合は空の配列が返されます。

引数team_nameには検索対象のチーム名を指定します。
例えば、敵のチーム名を指定してget_all_attackable_infantry_idsを呼び出せば、次の敵チームのターン開始時に自チームのユニットに攻撃できてしまう敵ユニットを検索できます。

void move(const std::string& id, const BoardPosition dst_pos)

void attack(const std::string& id, const BoardPosition dst_pos)

それぞれ、指定したIDの兵士に移動・攻撃をさせる関数です。
idに兵士IDを、dst_posに攻撃先の座標を指定します。

BoardPositionは構造体で、盤面の座標(2次元座標)を表します。
メンバxとyを持ちます。

なお、移動・攻撃ができない座標や兵士IDを指定した場合、エラーメッセージがコンソールに表示され、
移動・攻撃は行われません。

std::vector<BoardPosition> find_movable_position(const std::string& id)

std::vector<BoardPosition> find_attackable_position(const std::string& id)

それぞれ、指定したIDの兵士が移動・攻撃可能なマスを検索する関数です。
idに検索対象の兵士IDを指定します。

条件に該当するマスがある場合、それらのマスの座標が含まれた配列が返され、ない場合は空の配列が返されます。

BoardPosition get_position(const std::string& id)

int8_t get_hp(const std::string& id)

uint8_t get_action_remain(const std::string& id)

それぞれ、指定したIDの兵士の位置・HP・残り行動回数を取得する関数です。
idに検索対象の兵士IDを指定します。

注意事項

- マルチスレッドや外部プログラムなどを使用して実装した際の動作は確認しておりません。
  可能であればシングルスレッドで、外部のプログラムやスクリプトを使用せずに実装することを推奨します。
- CCにすでに実装されている関数やクラスのうち、本文書に記載されている以外のものを使用することは避けてください。
  対戦が正常に進行できなくなったり、CCが異常動作する恐れがあります。

1. heis総合クライアント"Compreheisive Client"の略。 ↩