|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| C:\Users\G-pos\Desktop\AIP2012\Resource\logo.png

|  |  |
| --- | --- |
| **2012 Science War AI 게임 ‘Diplomatics’ V0.14**통신 프로토콜 설명 | **작성** 이광무(iss300)  |
|  |  |

 |

1. **통신 패킷 규격**

Diplomatics는 서버-클라이언트 통신을 위해 구글이 개발한 ‘구글 프로토콜 버프’를 이용합니다. 본 프로토콜은 현재 공식적으로 **Java, C++, Python** 등을 지원하며, C# 등의 일부 언어는 비공식적으로 지원 중입니다.

 Diplomatics는 모든 통신에서 아래와 같은 양식의 패킷을 이용합니다.

|  |
| --- |
| **message** AIPacket{**int** header; |
| **int** cid; |
| **int** id *repeated*;  |
| **int** fromid *repeated*;  |
| **int** X *repeated*; |
| **int** Y *repeated*; |
| **char** types *repeated*;  |
| **string** aux *repeated*; |
| } |

 위 패킷을 ‘구글 프로토콜 버프’로 사용하시고자 하는 언어에 맞게 변환하여 이용하면 됩니다. ‘구글 프로토콜 버프’에 대한 자세한 정보는 다음 사이트에서 확인하실 수 있습니다.

<https://developers.google.com/protocol-buffers/docs/proto?hl=ko>

각 필드의 의미는 헤더나 통신 단계에 따라 다를 수 있습니다. 필드에 들어가는 값의 의미에 대해서는 다음 장에서 자세히 다뤄집니다.

1. **프로토콜**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **단계** | **헤더** | **클라이언트** | **방향** | **서버** |
| 핸드쉐이킹 | 1 | *안녕하세요! 저는* ***(Postech/Kaist) 소속****입니다.* ***CID(Client id)****를 보내주세요.* | **→** |  |
| 2 |  | **←** | *요청 받았습니다. 당신의 CID는 “아무개”입니다* |
| 3 | *좋습니다. 제 CID는 “아무개”입니다.* | **→** |  |
| 4 |  | **←** | Client가 준 CID와 서버에서 발급한 CID를 비교.차이가 없으면 Client에게 확인 완료를 통지. |
| (888) | **팀 이름**, **개발자 명단**, **각오**를 전송. | **→** |  |
| 턴처리 | 100 |  | **←** | 턴 개시!**패킷 데이터:** 맵 데이터 |
| 101 | AI 연산 후 명령서를 보냄.**패킷 데이터:** 명령서 | **→** |  |
| 102 |  | **←** | AI 명령서를 제대로 받았음을 통지.(이후 100을 통해 반복) |
| (888) | 게임 상에 띄울 메시지를 전송. | **→** |  |
| 종료 | 900 |  | **←** | 게임 끝 통지. |
| (901) | 게임 종료를 확인하였음을 서버에 통지. | **→** |  |

\* 위 표에서 괄호로 묶인 헤더의 경우 선택 사항으로, 서버에 전송하지 않아도 무방합니다.

* 1. **연결 시도**
	핸드쉐이킹 이전에 서버에 연결을 시도합니다. 연결 포트는 **12345**로 고정되어 있습니다.
	2. **핸드쉐이킹 단계** (헤더: 1 ~ 4)
	클라이언트가 서버에게 연결 수락을 요청하고, 클라이언트 AI의 정보를 교환하는 단계입니다.
* 헤더 **1 (**from **Client)**
	+ 클라이언트에게 소속 학교를 알리고 CID 발급을 요청합니다.

|  |  |
| --- | --- |
| **message** AIPacket{**int** header; | (= 1) |
| **int** cid; | (학교 번호. **1:** POSTECH, **2:** KAIST~~,~~ **~~3~~**~~: SNU~~) |
| **int** id *repeated*;  | (이하 미사용) |
| **…** |  |
| } |  |

* 헤더 **2 (**from **Server)**
	+ 서버에서 CID를 발급하여 전송합니다.

|  |  |
| --- | --- |
| **message** AIPacket{**int** header; | (= 2) |
| **int** cid; | (발급된 CID) |
| **int** id *repeated*;  | (이하 미사용) |
| **…** |  |
| } |  |

* 헤더 **3 (**from **Client)**
	+ 제대로 CID를 전달받았는지 확인하기 위해. 발급된 CID를 서버에 다시 전송합니다.

|  |  |
| --- | --- |
| **message** AIPacket{**int** header; | (= 3) |
| **int** cid; | (발급된 CID) |
| **int** id *repeated*;  | (이하 미사용) |
| **…** |  |
| } |  |

* 헤더 **4 (**from **Server)**
	+ 서버에서 발급한 CID와 전송된 CID가 같을 경우, 제대로 서버에 연결되었음을 알리기 위해 헤더 4번으로 패킷을 전송합니다. 이후 서버가 보내는 패킷의 cid는 0으로 처리됩니다.

|  |  |
| --- | --- |
| **message** AIPacket{**int** header; | (= 4) |
| **int** cid; | (= 0) |
| **int** id *repeated*;  | (이하 미사용) |
| **…** |  |
| } |  |

* 헤더 **888**[[1]](#footnote-1) **(**from **Client) [선택]**
	+ 서버에게 본 클라이언트의 팀 이름, 제작자 명단, 각오를 전송합니다.
	이 단계는 선택 사항으로, 게임 시작 전까지 이 헤더로 전송된 데이터가 없을 경우 팀 이름을 **‘Unknown’**으로 처리합니다.
	또한 이 헤더를 통한 팀 정보 전송은 핸드쉐이킹 단계에서 단 한 번만 가능합니다.

|  |  |
| --- | --- |
| **message** AIPacket{**int** header; | (= 888) |
| **int** cid; | (발급받은 CID) |
| **int** id *repeated*;  | (미사용) |
| **…** |  |
| **string** aux *repeated*; | (인덱스 순으로 **팀 이름, 제작자 명단, 각오**) |
| } |  |

* 1. **턴 처리 단계** (헤더: 100 ~ 102)
	본 게임이 시작된 후 턴을 처리하는 단계입니다. 이 단계에서 AI의 연산이 적용됩니다.
	이 단계는 게임이 끝나기 전까지 **100→**101**→**102**→100→**101… 순으로 반복됩니다.
	 이 단계에서 100, 101번 헤더의 경우, ***repeated*** 처리된 필드를 **배열**로 취급합니다. n번 “정보 단위”의 특정 필드는 그 필드의 n번째 요소와 같은데, 가령 100번 헤더에서 n번 유닛의 id 필드 값은, id 필드의 n번째 요소와 같습니다.



그림 1. 패킷 구성 개념도

* 헤더 **100 (**from **Server)**
	+ 새로운 턴이 개시되었음을 알립니다.
	헤더 100의 정보 단위는 게임 상의 **“유닛”**으로, 직전 턴에 게임 판에 존재했던 유닛들의 정보를 클라이언트에게 전송합니다.

|  |  |
| --- | --- |
| **message** AIPacket{**int** header; | (= 100) |
| **int** cid; | (= 0) |
| **int** id *repeated*;  | (유닛의 고유 id) |
| **int** fromid *repeated*;  | (유닛의 소유주. **1:** 아군, **0:** 중립, **-1:** 적군) |
| **int** X *repeated*; | (유닛의 x 좌표) |
| **int** Y *repeated*; | (유닛의 y 좌표) |
| **char** types *repeated*;  | (유닛의 종류)\* |
| **string** aux *repeated*; | (미사용) |
| } |  |

\* 유닛의 종류는 영문 대문자 하나로 기록됩니다.
 **I:** Drill Tank, **A:** Artillery, **C:** Scout, **T:** Turret, **W:** Center (영역), **기타:** 장애물

* 헤더 **101 (**from **Client)**
	+ 헤더 101의 정보 단위는 **“명령서”**로써, AI 연산을 수행하여 명령서를 생성하고 전송합니다. 101번 헤더 패킷은 100번 헤더가 클라이언트로 전송된 후 **제한시간 1초** 안에 서버로 전송되어야 하며, 제한시간을 지나도 101번 패킷이 전송되지 않은 경우 해당 턴에 아무 동작도 하지 않습니다.
	 또한 적군 유닛에 대한 명령은 무시되며, 아군 유닛에 대한 명령 역시 게임 규칙에 따라 연산한 결과 무효 처리될 경우 수행되지 않을 수 있습니다.
	 명령서의 명령과 서버(게임)의 표시 형태가 상이할 수 있으므로, 정확히 명령서가 전달되었는지 여부를 조사하시려면 **개발자 모드[[2]](#footnote-2)** 이용하시기 바랍니다.

|  |  |
| --- | --- |
| **message** AIPacket{**int** header; | (= 101) |
| **int** cid; | (발급된 CID) |
| **int** id *repeated*;  | (명령서가 적용될 유닛의 고유 id) |
| **int** fromid *repeated*;  | (*아래 참조*)*\** |
| **int** X *repeated*; | (*아래 참조*)*\** |
| **int** Y *repeated*; | (*아래 참조*)*\** |
| **char** types *repeated*;  | (명령의 종류)\* |
| **string** aux *repeated*; | (미사용) |
| } |  |

\* 명령의 종류에 따라 fromid, X, Y 필드에 입력될 값은 다음과 같습니다.

|  |  |  |  |
| --- | --- | --- | --- |
| types | fromid | X, Y | 기능 |
| M | (미사용) | 목적지 좌표 | 유닛을 목적지(X, Y)로 **이동(공격)** |
| D | 지원할 유닛 ID | (미사용) | 유닛(fromid)의 **방어를 지원**하도록 함 |
| H | (미사용) | (미사용) | 현재 위치를 고수(**방어**) |
| C | (미사용) | 목적지 좌표 | 기병을 **기병 추가 이동**시 목적지(X, Y)로 이동 |
| A | 지원할 유닛 ID | 목적지 좌표 | 유닛(fromid)의 목적지(X, Y)로의 **이동을 지원** |

모든 명령에서 **id** 필드에 명시된 유닛이 명령 수행의 주체입니다.

* 헤더 **102 (**from **Server)**
	+ 서버가 클라이언트의 명령서 패킷(헤더 101)을 수신하였음을 알립니다.

|  |  |
| --- | --- |
| **message** AIPacket{**int** header; | (= 102) |
| **int** cid; | (= 0) |
| **int** id *repeated*;  | (이하 미사용) |
| **…** |  |
| } |  |

* 헤더 **888 (**from **Client) [선택]**
	+ 서버(게임)에 띄울 메시지를 비동기적으로 전송합니다. 서버에 출력되는 메시지는 일반적인 온라인 게임에서 채팅 내용이 표시되는 방식과 같습니다. 기본적으로 메시지는 aux의 0번째 요소에 있는 것으로 간주되고, 이 외의 인덱스는 무시됩니다.

|  |  |
| --- | --- |
| **message** AIPacket{**int** header; | (= 888) |
| **int** cid; | (발급받은 CID) |
| **int** id *repeated*;  | (미사용) |
| **…** |  |
| **string** aux *repeated*; | (0번째 인덱스에 메시지) |
| } |  |

* 1. **게임 종료 단계** (헤더: 900, 901)
	게임 종료 처리를 하는 단계입니다.[[3]](#footnote-3)
1. **맺음**

 Diplomatics 통신 프로토콜에 대한 문의사항은 <http://gpos.postech.ac.kr> 에 마련된 전용 Q&A 게시판을 이용해주시기 바랍니다.

1. 888번 헤더는 게임 전반에 걸쳐 비동기적 통신을 하는데 사용되는 헤더입니다. [↑](#footnote-ref-1)
2. 대기 화면에서 ‘D’ 키를 눌러 게임을 시작. Space(혹은 Enter)를 눌러 턴을 넘기고, Space를 누른 상태에서 ‘F’ 키를 누르고 있으면 턴이 빠르게 진행됩니다. [↑](#footnote-ref-2)
3. V0.14를 기준으로, 새로운 게임을 시작하기 위해 서버(게임) 프로그램을 종료한 후에 재시작해야 합니다. 즉 게임 종료 후에 후속 처리가 필요하지 않으므로 이 단계를 고려하지 않아도 좋습니다. [↑](#footnote-ref-3)