AIWolf International Competition 2019 Regulations

2019/05/11 Ver 1.3.0

1. Werewolf Game Rules for the 2019 International Competition

1.1 Number and Types of Roles

In this competition, games will be composed of 15 players or 5 players. For each game, the role distribution will be as follows:

15 player game: 8 Villagers, 1 Seer, 1 Medium, 1 Bodyguard, 3 Werewolves, 1 Possessed
5 player game: 2 Villagers, 1 Seer, 1 Werewolf, 1 Possessed

1.2 Description of Roles

1.2.1 Roles of the Villager Team:

The Roles of the villager team are as follows: Villager, Seer, Medium, Bodyguard.

Villager: Has no particular special abilities.

Seer: At the end of each day, the seer can choose one player to “Divine”. The seer will know whether this player is a werewolf or a human.

Medium: When a player is voted and eliminated from the game, the Medium will know whether this player is a werewolf or a human. (Otherwise, players who are voted are not revealed until the end of the game)

Bodyguard: At the end of each day, the bodyguard can choose one other player to “Guard”. This player will not be affected by the werewolves’ attack. In this competition, the Bodyguard will not receive any feedback about the result of this action (if the werewolves missed intentionally, or if they prevented an attack, etc).

1.2.2 Roles of the Werewolf Team:

Werewolf: At the end of every day, each werewolf will vote on a player of the villager team to “Attack”. The player with the highest number of votes will be attacked. Additionally, The werewolves have access to the “whisper” channel that can only be heard by other werewolves.

Possessed: The possessed has the same abilities as a Villager (i.e. no abilities), but is aligned with the Werewolf Team. Seers and Mediums will identify the possessed as Human.

1.3 Conversation between agents

1.3.1 Turn system

The conversation happens in turns. Each player is able to broadcast one message at each turn. However, the player can also choose to not broadcast a message (SKIP or OVER). The server will collect the messages from each player and send it to all other players. The order of the messages is random, therefore the order of the messages in a single turn has no special meaning.

Each player can broadcast a maximum of 10 messages each day. However, SKIP and OVER are not counted against this limit.

The server will end the day phase when all players broadcast the OVER message, when all players broadcast the SKIP message three turns in a row, or when 20 turns have passed.

1.3.2 Conversation on the first day

There is no conversation on the Day Phase of the first day. (Also known as Night Start rule)

1.3.3 Werewolf Whisper

The werewolves can use the whisper action after the voting of the Day Phase is concluded, and before voting for the Attack target. The whisper happens using a turn system identical to the Day Phase conversation. In the first day of the game, Although there is no attack on the first day of the game, the Werewolves can use whisper during the Night Phase of the first day. For example, this can be used to discuss general strategy such as fake ComingOuts.

If there is only one werewolf in the game, there will be no whisper stage. Please be careful that the whisper method will not be called in this situation.

1.4 Voting and Re-Voting

The voting to eliminate a player wil happen end of the Day phase. The identity of the agent that was eliminated is available for use of the Seer, Bodyguard, and Werewolves, so that they can make their night phase decisions.

If the voting results in a simple majority, that player is eliminated. In case of a tie, a single re-vote will be performed. No conversation happens before the re-voting. During the re-vote, any player can be the target of a vote (not just the tied players). If there is still a tie after the re-vote, one of the tied players (chosen randomly) will be eliminated.

The werewolf attack target will be decided by an identical voting process, including a single re-vote in case of a tie. The whisper action cannot be used before a re-vote.

1.5 Special Abilities

1.5.1 Seer

The seer can choose a player during the night phase, and will receive information about whether that player is a human or a werewolf. The seer will be informed of the result of the voting stage before being asked to choose a divination target. The seer is able to perform divination during the first day.

1.5.2 Medium

The medium receives information about whether the player eliminated by the voting stage was a human or a werewolf. As there is no voting on the first day, the medium will receive no information on that day.

1.5.3 Bodyguard

The bodyguard can choose a player during the night phase, and that player will not be affected by the werewolf's attack. The bodyguard can choose an eliminated player, but nothing will happen in that situation. The bodyguard will be informed of the result of the voting stage before being asked to choose a protection target.

2. Preliminary Contest and Final Contest

The competition will happen in two stages: The preliminary contest and the final contest.

2.1 Practice Contest:

Before the Preliminary contest, we will conduct daily practice contests among the registered agents. Every day, each agent will participate on at lease one set (100 games) of 5-player game and one set of 15-player game.

In the case where there are less than 15 agents registered, sample agents will be added to the practice contest. The logs of the practice contests can be downloaded from the contest registration page.

2.2 Preliminary Contest:

In the preliminary contest, repeating the trials described below until each registered team passes the prescribed number of "Games", and the winner will be selected by order of average winning rate. The preliminary contest will select a total of 15 finalist teams.

(Note: The number of Games in one trial is 100. We will try to run as many trials as possible, depending on the available time)

> **The Procedure of One Trial:** From the list of registered teams, a set of 5 or 15 players are chosen at random. The roles of the players are selected randomly (see section 1.1 for a list of roles). The players in the winning team (villager or werewolf) receive 1 point per game. The one trial has 100 games with the selected set of players.

Participants in the preliminary contest must submit the executable files that allows the agent to play as any role (see section 3 for more information about submission). Also, a participating team may be composed of multiple people.

2.3 Final Contest:

The 15 players selected in the preliminary contest will participate in a final set of Games. The role of each player will be randomized for every game. The winning rate will not be weighted by role. Also, in every game, the ID of the players will be randomized.

3. How to submit an agent

3.1 Team Registration:

To register a team in this competition, you must make an account in the competition webpage: http://contest.aiwolf.org/en. After you create your account, you can register for the competition.

Additionally, the logs of the daily practice contests will be available in the above webpage as well.

3.2 Player Submission

Submit the files that implement an agent capable of playing all roles described in section 1 of this document. The files to be submitted depend on the programming language used.

Additionally, besides the executable files submited during the registration, the submission of the agent's source code is also necessary. The method for submitting the final source code will be announced separately.

3.2.1 Java Agent

File to submit: jar archive

For teams creating a Java agent, you may submit a single jar file containing all your code. If you are using libraries for machine learning or other things, please include them in your jar file. It will automatically create the necessary classpaths.

Also, include any data files that you may need in the jar archive, and read them from there. For example, if you need to read the file /data/foo.txt inside the jar archive, you can use *“InputStream is = getClass().getClassLoader().getResourceAsStream(“data.foo.txt”)”* to create an input stream that reads your data.

Please be aware that there may be conflicts if you include the following files in your jar file “aiwolf-client.jar”, “aiwolf-server.jar”, “aiwolf-common.jar”, “aiwolf-viewer.jar”, “jsonic-xxxx.jar” included in the AIWolf platform our project provides. This may cause problems when running your agent. So avoid including these files in your jar file.

3.2.2 C# Agent

File to submit: dll file, zip archive

If you create a single dll file, please register your dll file directly. If you create multiple dll files, please submit all of them as a single zip archive.

However, if you submit a zip archive, make sure to name the file with your player class as “teamname.dll”. For example, if your team name is “tori”, your player class needs to be inside a file named “tori.dll”, or it will not run. If you are including a single dll, there are no restrictions on the filename.

Note that while you can include data files and library files in your zip file, if your agent tries to read data from any file not included in the registered zip file, it will be ground for disclassification of the team.

3.2.3 Python Agent

File to submit: zip archive

Teams submitting agents in python should register a zip file. If the contest server is not able to open your zip file and run your agent, your agent will be disclassified.

When registering your agent to the contest, make sure to indicate the starting script.

However, make sure that your start script is inside a directory with your team name. For example, if your team name is registered as “team_foo” and your starting script name is “start_bar.py”, your zip file should contain the “team_foo/start_bar.py” path. In short, please include all scripts under a directory with your team name.

Note that while you can include data files and library files in your zip file, if your agent tries to read data from any file not included in the registered zip file, it will be ground for disclassification of the team.

4. Forbidden activities

During the competition, the following activities are forbidden to the agents. Agents that do not follow these restrictions, or agents that throw errors during the contest, will be eliminated. In certain cases the organizers might contact the team directly to find a solution, at the organizer's discretion.

- Broadcasting a message that cannot be created by the ContentBuilder
- Writing to files (reading files is permitted under certain conditions)
- Connecting to a network
- Creating new threads
- Executing a program as a separate process
- Taking more than 100ms to respond to a request from the server (small deviations from this time limit might be tolerated)

About reading data from files: Your agent may only read data from the files that you have uploaded when registering to the contest. Java agents may read data from the .jar file that was registered. Python and .NET agents may read data from the files included in the .zip archive that was registered.

5. About running player programs

In this competition, we will use the aiwolf-ver0.5.x server available at http://aiwolf.org/en/server. Please refer the instructions listed under "Running AIWolf Server" in the same webpage to carry out games in your environment..
To create a player that can join the game, prepare a program that inherits the *org.aiwolf.common.data.Player* Interface inside *AIWolfCommon.jar*. Or prepare a program in .NET or Python that can communicate with the server in the same manner as the Interface above.

5.1 Methods that need to be implemented in the Player Interface (Java)

A class inheriting the Player interface needs to implement 11 methods. These methods are divided in the following 4 groups:

- Methods for organizing information: *initialize, update, dayStart, finish*
- Methods for targeted actions: *vote, attack, guard, divine*
- Methods for dialogue: *talk, whisper*
- Methods for naming: *getName*

5.1.1 Methods for organizing information (initialize, update, dayStart, finish)

These methods are used to process information, and do not need to return anything.

initialize(GameInfo, GameSetting): This method is called once at the start of the game. The arguments are the current state of the game *GameInfo*, as well as the information about the setup of the game (number of players and roles) *GameSetting*.

update(GameInfo): Is called before each invocation of all other methods (except *initialize*), to update the agent's information about the game state. When called before *finish* it will also provide the full list of players and their roles.

dayStart(): It is called once before each day phase.

finish(): It is called after the game is finished.

5.1.2. Methods for targeted actions (vote, attack, guard, divine)

Methods that must return the ID of the agent to be targeted by the action. In the case of *Attack, Guard, Divine*, these methods will only be called on agents with the respective roles.

vote(): Return the player to be voted out of the game on that day phase.
attack(): Only for werewolves. Return the player to be voted for attack on that night phase.
guard(): Only for the bodyguard. Return the player to be protected on that night phase.
divine(): Only for the seer. Return the player to be investigated on that night phase.

5.1.3. Methods for dialogue (talk, whisper)

These methods must return the message to be broadcast (in String format). *whisper* is only called for werewolf players.

talk(): Returns a message that will be broadcast for all active players. In this competition, the message must be constructed using the class org.aiwolf.client.lib.ContentBuilder, which will guarantee that it follows the defined protocol. (See section 5.3)

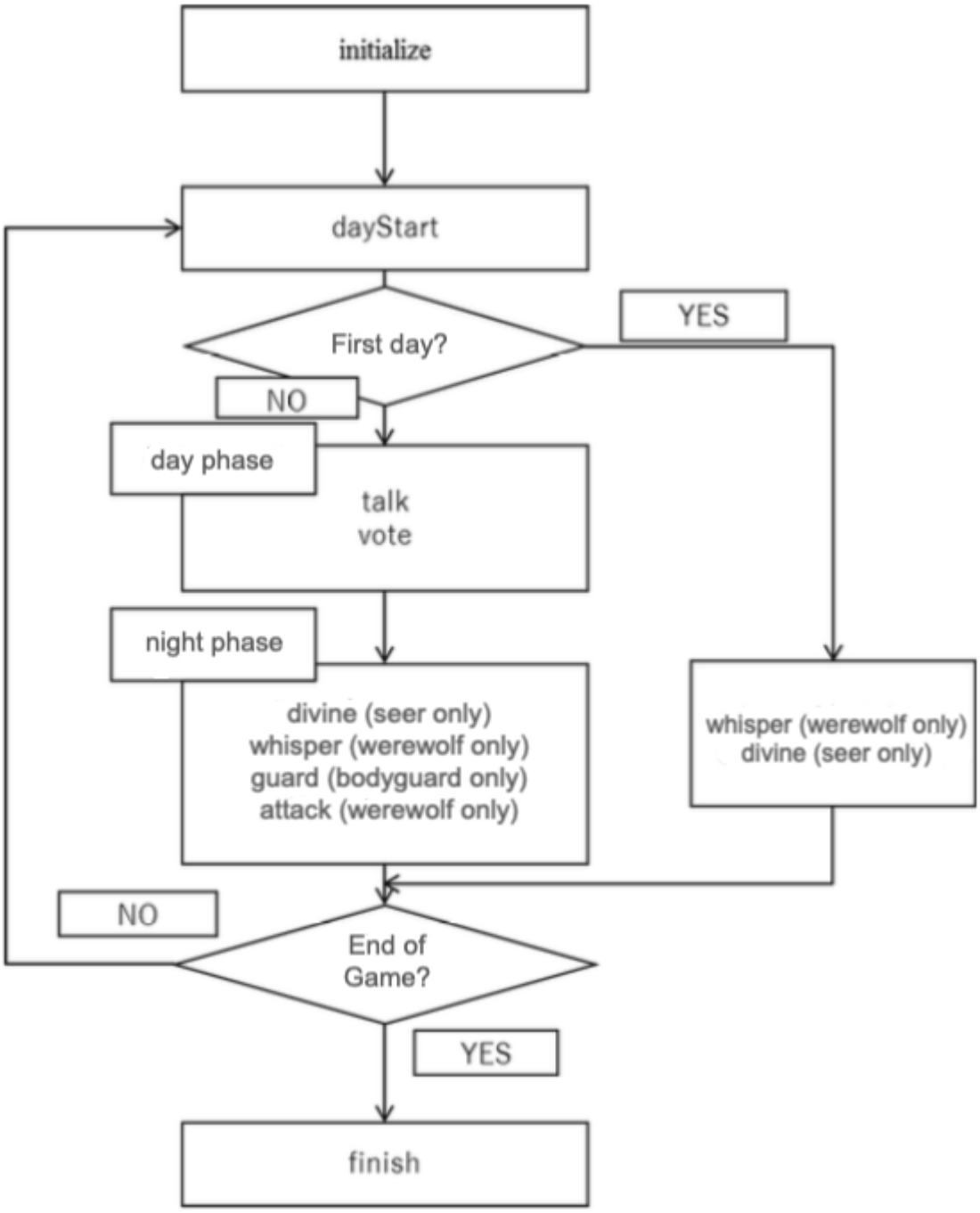
whisper(): This method is only called for werewolf agents. The messages returned for this method are not displayed to the non-werewolf players. In this competition, the message must be constructed using the class org.aiwolf.client.lib.ContentBuilder, which will guarantee that it follows the defined protocol. (See section 5.3)

5.1.4. Naming Methods (getName)

getName(): Returns the name of the player (in String format). The name of the player is displayed on the Game's log. Please return the team name that you registered during your registration for the competition. If you return a different name, this may result in disclassification.

5.2. Timing for invoking each method.

The methods described in the previous section (except for getName) are called following the flow described below. While the flow does not include the “update” method, it is called before every other method, with the exception of initialize.



5.3. Valid Broadcast Strings

In this competition, all dialogue among agents must be composed of strings that can be created

by the org.aiwofl.client.lib.ContentBuilder class. There are 23 kinds of messages, listed below.

- estimate：X believes that Y's role is Z. (EstimateContentBuilder)
- comingout：X declares that Y's role is Z.(ComingoutContentBuilder)
- divination：X divines Y. (DivinationContentBuilder)
- divined：X divined Y, and the result was Z (Human or Wolf) (DivinedResultContentBuilder)
- identified：X used the medium power, and identified Y as Z (Human or Wolf) (IdentContentBuilder)
- guard：X protects Y. (GuardCandidateContentBuilder)
- guarded：X protected Y. (GuardedAgentContentBuilder)
- vote：X votes for Y. (VoteContentBuilder)
- voted ：X voted for Y. (VotedContentBuilder)
- attack：X votes for Y to be attacked.（AttackContentBuilder）
- attacked ：X attacked Y. (AttackedContentBuilder)
- agree：X agrees with broadcast T. (AgreeContentBuilder)
- disagree：X disagrees with broadcast T. (DisagreeContentBuilder)
- request：X requests Y to do Z (RequestContentBuilder)
- inquire：X inquires Y about Z (InquiryContentBuilder)
- because：X states sentence Z because of reason Y (BecauseContentBuilder)
- and：X states A and B and … (AndContentBuilder)
- or：X states at least one of A or B or … (OrContentBuilder)
- xor：X states either A or B (XorContentBuilder)
- not：X negates Y (NotContentBuilder)
- day：X states that Y happened on day T (DayContentBuilder)
- over：I have nothing else to say (if all players broadcast OVER, the Talk Phase ends)(OverContentBuilder)
- skip：I will say nothing now (even if all other players broad cast OVER, the Talk Phase does not end) (SkipContentBuilder)

5.4. About the Player class package

Please create a unique Player class. Avoid just rewriting the sample classes such as org.aiwolf.sample.player.SampleRoleAssignPlayer.

Also, please include your Player Class in an independent package. The recommended package naming convention is to use the reverse order of your e-mail's address. (For example, if your e-mail address is contestant@example.com, and your Player class is MyPlayer, the header of your class would probably look like this:

```
package com.example.contestant;
Import org.aiwolf.sample.lib.AbstractRoleAssignPlayer;

public class MyPlayer extends AbstractRoleAssignPlayer {

}
```

In this case, when submitting your agent code, you should write the following in the "class path" section:

```
com.example.contestant.MyPlayer
```

6. Using other Programming Languages

Besides Java, we will consider entries in Python and .NET. When using these languages, please check their respective libraries. Using TCP-IP libraries for socket communication, and making sure that the correct messages are passed back and forth between server and client, it is possible to participate in the game. When in doubt, please contact the organizing comittee: gm@googlegroups.com

However, please note that the contest will be executed in a linux machine, so if you do your development in a different environment, make sure that your code is portable. Clients that cannot run on the server environment will be automatically disclassified. For example, in a previous contest one player used a system-dependent JSON library for python that did not run on the linux server, and caused disclassification. Please be careful.

We strongly recommend that you register an initial version of your agent early and participate in the practice contests.

7. Changes

These regulations are subject to change at any time. Changes will be announced at the project page (http://aiwolf.org/en), the development mailing list (aiwolfdev@googlegroups.com) or our Slack Channel (https://aiwolfen.slack.com).

8. Change History

2019/05/11 -- ver 1.3.0 -- Matching with Japanese version 1.3 -- minor fixes.
2019/05/09 -- ver 1.1.0 -- Clarification about the practice contest, clarification about reading from files, other minor changes.
2019/02/09 -- ver 1.0.0 -- Translated from the 4th AI Wolf competition rules by Claus Aranha