2nd International AIWolf Competition Regulation[1]

2020/04/02 Ver 1.2.0

1. Werewolf game rules for the 2nd International AIWolf Competition

1.1 Number and Types of Roles

Games are composed of 15 players or 5 players. Depending on the size of the game, the role distribution is 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 the Roles

1.2.1 Roles of the Villager Team:

There are four kinds of roles in the villager team: Villager, Seer, Medium, and Bodyguard.

Villager: Has no 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 eliminated from the game by voting, the Medium is informed of whether that player was a werewolf or a human. If there is no Medium in the game, the alignment of players eliminated by voting is not revealed.

[1] 6th AIWolf Competition

Bodyguard: At the end of each day, the bodyguard can choose one other player to “Guard.” That player is not affected by the werewolves’ attack. The bodyguard does not receive any feedback about the result of this action (i.e., whether he successfully protected an attack or the werewolves did not attack).

1.2.2 Roles of the Werewolf Team:

There are two kinds of roles in the villager team: Werewolf and Possessed.

Werewolf: At the end of each day (except the first), the werewolves vote on a human player. The player with the highest number of votes is Attacked. Also, only the werewolves can use the “whisper” channel to exchange information while hidden from human players.

Possessed: The Possessed has no abilities, like the Villager. However, the Possessed wins with the Werewolf team. Seers and Mediums identify the Possessed as human. Also, the possessed can NOT use the "whisper" channel.

1.3 Conversation between agents

1.3.1 Turn system

Conversation happens in turns. Each player can broadcast one message at each turn. The order in which the players broadcast within a turn is random.[2] Also, a player may choose not to broadcast a message.

In a day, a player may broadcast up to 10 messages. This limit does not include "Skip" and "Over" messages.

The "talk" phase ends when every player broadcasts "Over," or when every player broadcasts "Skip" three times in a row. The "talk" phase also ends when 20 turns have passed.

1.3.2 Conversation on the first day

On the first day, no conversation takes place.

1.3.3 Werewolf Whisper

The werewolves can use the “whisper” channel only after the elimination vote and before the attack vote. Although there is no attack vote on the first day, the whisper channel can still be used. The werewolves can use “whisper” to discuss strategies such as fake role claims.

If there is only one werewolf in the game, no whisper takes place. Please be careful that the

[2] In previous competitions, message exchange within a turn was synchronous: all the agents submitted the turn's message at the same time, and all messages were revealed at the same time. However, in this competition, the message exchange is not synchronous. Each agent receives all the messages sent up until their time to broadcast. Please be careful of the difference.

whisper method is not called in this situation.

1.4 Voting and Revoting

The vote to eliminate a player from the game happens after the “talk” phase. The server informs the werewolves, seers, and bodyguards about the eliminated player so they can use this information to make their decision. Other players are informed the following day.

If there is a tie, the vote is repeated one time. In this case, there is no extra conversation. If there is still a tie in the second vote, then the eliminated player is chosen randomly from the tied players. During a revote, all players have a vote, and they can vote in anyone, not just the players tied in the first vote.

The werewolf attack vote happens similarly. One revote is allowed in case of a tie, and there is no whisper before the revote.

1.5 Special Abilities

1.5.1 Seer

During the night phase, the Seer can choose a player. The seer receives information about whether that player is a human or a werewolf. The Seer receives the name of the player eliminated by voting before choosing the divination target. It is possible to do divination on the first day.

1.5.2 Medium

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

1.5.3 Bodyguard

The bodyguard can choose any player other than themselves to protect. That player is not affected by the werewolves’ attack. The bodyguard can choose a player that has been eliminated, but nothing happens in that case. The bodyguard receives the name of the player eliminated by voting before choosing the protection target.

2. Preliminary Contest and Final Contest

2.1 Testing:

Before the Preliminary contest, participants can conduct test sessions. The server executes practice games a few times every day with all the registered teams, and check if all agents can run without errors. We strongly recommend that you use this to test if your agent runs correctly on the server.

2.2 Preliminary Contest:

In the preliminary contest, we repeat the “Game Procedure” described below until each registered team participates in a minimum number of games. The teams with the highest winning rate are selected to be finalists. A total of 15 teams are selected for the finals from the preliminary contest.

(Note: The minimum number of games is 100 per team. However, we plan to exceed that limit and run as many games as time allows)

> **Game Procedure:** A set of 5 or 15 players are chosen randomly from the list of registered teams. Each player receives a random role from the list in section 1.1. The players in the winning team (village or werewolf) receive 1 point. This procedure is repeated 100 times with the same set of selected players.

Participants in the preliminary contest must submit source code that allows the agent to play any role (see section 3 for more information about submission). Also, teams 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 set of 100 games, 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/. After you create your account, you can register for the competition.

3.2 Player Submission

Submit the files that implement an agent capable of playing all roles described in section 1.1 of this document. The files to be submitted depend on the programming language used.

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”. This may cause problems when running your agent. So avoid including files that use these names in your jar file.

The recommended Java version is 11. If you use versions other than that, your agent may not run correctly.

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.

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 disqualified.

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

To do this, 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 path “team_foo/start_bar.py”. Please register your starting script as “start_bar.py”. In short, include all files inside a directory with your team name, as files in other directories may not be executed, and may result in disqualification for your team.

3.3 Source Code Submission

Agents selected to the finalist tournament must submit their source code. When submitting the finalist version of your agent, submit the source code of all files used. The method of submission will be detailed after the preliminary contest. Agents submitted without source code will be disqualified.

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. .NET agents may read data from the .dll files submitted. Python agents may read data from the files included in the .zip archive that was submitted.

5. About running player programs

In this competition, we will use the aiwolf-ver0.6.x server available at http://aiwolf.org/en/server. We will use the instructions listed under “Running AIWolf Server” in the same webpage.

To create a player that can join the game, prepare a java 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 implements 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)

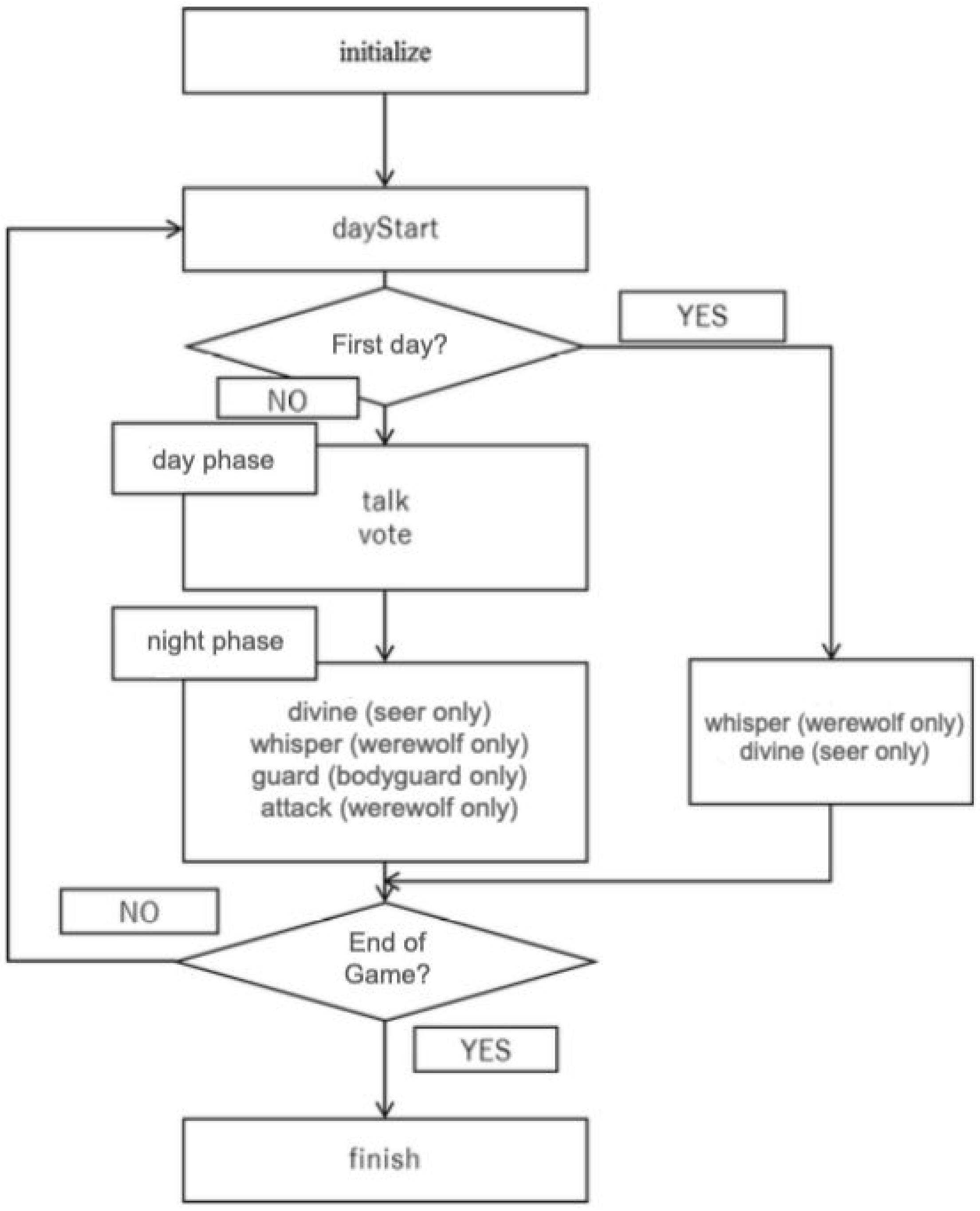
whisper(): This method is only called for werewolf agents. The messages returned for this method are not displayed to the non-werewolf players. In this competition, the message must be constructed using the class org.aiwolf.client.lib.ContentBuilder, which will guarantee that it follows the defined protocol. (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 an 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 email'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 source 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 committee: 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 disqualified. 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 disqualification. Please be careful.

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

2019/02/09 -- ver 1.0.0 -- Translated from the 4th AI Wolf competition
2019/05/09 -- ver 1.1.0 -- Clarification about the practice contest, clarification about reading from files, other minor changes.
2020/03/29 -- ver 1.2.0 -- Updated rules for 2020 (non-synchronous broadcast, Java 11)