Artificial Intelligence Competition

Context

You find yourself against your will in an epic war of humongous proportions, in which the only goal is to win over your enemies. Hopefully, you are accompanied by 6 of your brothers-in-arms and have limitless supplies of snowballs, your favorite weapon. To win, you must use your tactical skills and teamwork to **exterminate the enemy** or **bring the flag back to your base**.

Game Description

Snowball fights are played by two teams of 7 players in a predefined map. The map is finite and is riddled with walls that let you take cover and hide. You must implement the behavior of the seven players that you control. Your players can undertake actions that will allow you to either eliminate the enemy players or to bring the flag back to your starting point. A game lasts a maximum of 5 minutes, after which the winning team is determined by the sum of the remaining health of all the players. The game runs at 30 frames per second, which means 33 milliseconds per frame and a total of 9000 frames in a single game.

There are three ways to end a game:

- Capture the flag in the center of the map and bring it back to your starting point. The flag can be picked up only after either 20% of the players have fallen in combat or after half of the game duration, whichever happens first.
- Eliminate all the players in the other team by throwing snowballs at them. Be careful, as friendly fire can hurt your own players.
- If the game isn't already over after 5 minutes, the team that has the highest sum of remaining health points wins.

Your team will be disqualified if your program crashes. There will be a tie if both teams crash at the same time. There will also be a tie if, at the end of the game, both teams have exactly as many remaining health points.

Additional challenge: in the second part of the competition, a thick blizzard strikes the map and reduces player visibility to a radius of 225 distance units!

World Description

World Map

The dimensions of the world map are variable. The dimensions use floating-point values, meaning that the world is continuous rather than discrete. It is therefore possible to move to any specific point on the map. The map contains walls, which can be of any polygonal shape, convex or concave. Walls block players and snowballs and cannot be seen through. All the players of the same team start at the same location when the game begins.

Team

A team is a collection of players. The union of the areas visible to each player in it makes up the area that is visible to the team as a whole.

Player

A player starts with 100 hit points, and can execute several predefined actions (more on that later). The walking speed of a player depends on its remaining health: the more a player is damaged, the slower it moves. Players have a field of view of 360 degrees (meaning they see everything around them), but cannot see through walls. The property getters of non-visible enemy players return garbage. There is no collision between players. Each player is identified by a unique ID.

Player State

A player has a state, which contains a state type, the current action, and the queued action. The state type lets you know what the player is generally doing if he's visible (for instance, is moving, throws a snowball, is idle). It's the only state information available on enemy players. The queued action is an action that has been sent to the server but has yet to be acted upon. It is therefore better to wait until it has been treated before sending more.

Snowball

A snowball travels in a straight line through the map. Its speed depends on its throw distance and its damage depends on its speed. A snowball thrown far away will therefore be faster and more potent. A player is hit by a snowball if it comes inclusively within 15 units of her; the snowball disappears and the player is damaged. If multiple players find themselves within colliding distance at the same simulation frame, they are all damaged. Snowballs cannot pass through walls. If a snowball is visible, its orientation, speed, destination and damage are accessible. Snowballs are identified with a unique ID.

Flag

The flag is at the center of the level and is always visible. When it is picked up by a player, the player is slowed down and becomes visible to all players. The flag does not collide with anything.

Player Actions

MoveAction

This action lets a player move to a new destination. The **shortest path** to the destination, taking walls into account, will be followed. The path is computed by the server and is not made available on the client side. The player's speed depends on its remaining health.

ThrowAction

This action lets you throw a snowball to a given destination. When a player starts throwing, she needs to charge up for some time. This time depends on the throw distance. The throw distance

is clamped between 50 and 600 units. The speed and damage of the snowball also depends on the throw distance.

IdleAction

This action tells the player to stop whatever she's doing.

PickFlagAction

This action lets a player pick up the flag. The flag can only be picked up once 20% of players have fallen, or less than half of the game time remains. A player can pick up the flag if she's inclusively within a radius of 10 units of it. When the flag is picked up, the carrier becomes visible to the other team and her speed is cut in half. The flag cannot be taken from another player: the player must either drop it voluntarily with the DropFlagAction, or will release it if killed.

DropFlagAction

This action lets the player drop the flag. Your team wins if the flag is dropped inclusively within 10 units of your team's starting point.

Implementing a Client

A client needs to subclass *Gang* and implement the *compute* method. This method is called on each frame (at an interval of 33 milliseconds). The return value is the list of actions that your players should execute. At most one action per player can be started on any given frame. If the method takes too long to execute, your players will simply carry on doing whatever they were doing. It is therefore not necessary to return every 33 milliseconds if you need more time. Refer yourself to the default implementations **MyGang.java** or **mygang.py** for concrete examples. Algorithms for determining speeds, damage, charge-up times and such are described in the source's documentation (the most useful ones for that matter being the player documentation and the ThrowAction documentation).

Technical Details

To lauch a snowball fight, you must start the server and connect two clients to it. Clients must be programmed in Java or Python.

Server

To start the server, open a terminal window in the *server* directory. This directory should contain at least *AIServer.jar*. To run it, execute this command:

java -jar AIServer.jar map.txt

To run the server with the blizzard and reduced visibility (for day 2), run this command instead:

java -jar AIServer.jar map.txt a

If you see the snowstorm, then you are in advanced (blizzard) mode. When the server is

running, all you have to do is to connect two clients to start the game. When the game is over, you have to run a new server instance to start a new game.

Client

To connect a client to the server, you must first create a project in the IDE of your choice.

(Note: you can provide the name of your team as the first parameter to the main function.)

Java

Once your Java project has been set up, you must include the *AIClient.jar* library in your project (from *client/java*). Then, add Main.java and MyGang.jar to your project from *client/java/src*. You must then implement the *compute* method in MyGang. To connect to a server, run the project from *Main.java*. Accessing non-public and non-protected methods is strictly prohibited. You should note have to use setters on *World* objects in your implementation.

Python

The Python client source can be found in *client/python*. It has three modules: *world.py*, *actions.py*, and *utilities.maths.py*. You must implement the *compute* method in *mygang.py*. To connect to the server, execute main.py. Accessing members whose names start with _ or __ is strictly forbidden. The *World* object is publicly read-only and must not be modified through shady ways.

Documentation

The Java and Python environment documentation can be found at:

client/java/doc
client/python/doc (constants are documented right in the code itself)

Many helper methods were made available for your convenience. Use them!! (Mainly the ones in *utilities.maths*, *Player*, *Snowball*, *Map*)

Correction

Each team will fight each other team one time, and the final score will be decided with the number of victories. The number of draws will be used to break an eventual tie. If there's still a tie, the team with the least average victory time will win.

The levels used for correction will be similar to the one provided, but not identical. The number of walls and the dimensions of the map will be similar. All games will be played with 7 players on each side. The levels will always be symmetrical.

Scores for the first run (on the first day) and the second run (on the second day) are distinct and independent.

tl;dr

- 7 vs. 7 snowball fights.
- Your seven players start at the same location. There is no collision between players.
- Players can throw snowballs, move, pick up the flag, drop the flag, or do nothing.
- The speed of a player depends on its health points. The speed and damage of a snowball depends on how far it is thrown.
- Friendly fire hurts your players.
- The map coordinates are continuous.
- The game runs at 30 simulation frames per second (33 milliseconds per turn). If you take longer than 33ms to react, the only consequence is skipping that turn.
- The levels have walls. Players can't pass through them, and they block visibility.
- The more damage a player has taken, the slower she moves.
- The advanced mode reduces visibility to a radius of 225 units around your players.

Annex 1 - Navigating the UI

- Rotate: left-click drag
- Move around: right-click or middle-click drag
- Zoom: mouse wheel
- Team Colors: hold Ctrl

Annex 2 - Level Configuration

The provided level is entirely configurable. You can modify the size of the map, the number of players, the starting positions, the flag position and the positions, shapes and number of walls.

Mandatory commands

m <map width>,<map height>
p <number of teams> <number of players per team> [<starting X location of team 1>,<starting Y position of team 1>, [...]]
f <flag X>,<flag Y>

Optional and repeatable commands

w [list of <x>,<y> of wall vertices in CLOCKWISE order, each separated by a space]

See *server/map.txt* for an example.