Summary

Tournament Structure

This new Competitive Track follows a round robin structure between agents. In each pairing:

- The agents will play a fixed set of levels together in order only once.
- The agents play levels by taking alternating shots
- The agents cannot restart the level, losing a level is a 0 score for both agents.
- Each agents score total from these levels will be added to an overall score for that agent

The winner of the final competition round will be the AIBIRDS 2015 Champion of the Competitive Track.

Details

Setup

- Run new_track_server.jar with the time setting as before
  e.g.: java -jar new_track_server.jar -t 60 (starts the server with the time limit 60 mins)

- Make sure you install both plugins (both can be found in the folder “plugins”) into chrome
  1. plugin_gameinteraction
  2. **and** plugin_videorecording

- Connect 2, 4, 6 or 8 agents

- Make sure to open the appropriate amount of chrome windows (1 to 4)

- You have two options to pair each agent:
  - Manually pair each agent by clicking the pair buttons.
  - Immediately clicking start will randomize the first pairings

  Either way all subsequent pairing will be randomized

Description

The competitive track is substantially different to the standard track. In the competitive track two agents compete against each other by playing the same level with alternating shots. The agent who scores the winning shot gets all the points of the level. Because of this, it is important who makes the first shot.

Bidding:

Agents make concealed bids for the right to make the first shot. Agents have 30 seconds to analyse a level and to make a bid. Bids can be positive or negative integers, or 0. Negative integer represents a bid to make the second shot. If an agent does not make a bid within the time limit, the bid is considered to be 0.

The agent with the higher bid wins and makes the first shot. In case both agents make the same bid, the agent who submitted its bid first is considered to have the higher bid. An agent *wins the bidding* if it made a positive bid and has the higher bid, or if it made a negative bid and has the lower bid. Note that both agents can win the bidding.

Example:

If an agent who won the bidding also scores the winning shot, that agent will pay the positive amount of points of its bid to the other agent. If the agent who scores the winning shot did not win the bidding, no payment is made. For example, agent A won the bid by bidding 15,000 points. If agent A scores the winning shot and the score is 33,000 points, then agent A receives 33,000 points but has to pay 15,000 points to agent B. In a different match, agent C has the lower bid by bidding -10,000 points and makes the second shot. If agent C scores the winning shot and the score is 25,000 points, then agent C receives 25,000 points but has to pay 10,000 points to agent D who made the first shot.

Playing/Running:

When it is the agent's turn, the agent has 30 seconds to take screenshots at any time, zoom in and out at any time, and submit one shot. If no shot is submitted during this time, the game server will automatically make a useless shot (for example shoot backwards). After a shot has been made (or after the 30 seconds are over), the server will wait for another 15 seconds for the game to become static. If the game is not yet over, it will then be the other agents turn.

During an agents turn, the competing agent cannot interact with the game. In particular it cannot take screenshots or make a shot.

Competition Structure:

In each competition round, agents are assigned to a group of up to 8 agents and have to compete against each agent of the same group. The order in which agents are paired up is selected randomly.

- Each competition round consists of a fixed set of levels and each agent plays the same levels against all other agents of the same group.
- All agents of the same group are playing simultaneously. Agents will be started only once per competition round and then play against all other agents of their group. Two minutes after all games of a group are played, agents will be stopped.
- The number of levels we use in each competition round depends on the number of participants. The levels used in the competition will not be known in advance to any participant.
- During each pairing of two agents, the fixed set of levels is played in the same order and each level is played only once.
- The overall score of an agent in a competition round is the sum of all points the agent received during that competition round, that is the sum of points of all pairings with all other agents of the same group.
- We will have a dynamically updated leader board for each group where all agents are ranked according to their current overall score. Agents can query the current leaderboard of their group at any time.
- Depending on the number of participants, there may be only one competition round. If there are multiple rounds, the top ranked agents of each group move to the next competition round.
- The winner of the final competition round will be the AIBIRDS 2015 Champion of the Competitive Track.

Turns and Stages effect on Commands

The most significant change to the protocol is that the server now has a state. This state determines what commands an agent is able to use at certain times.

- See getCompeitionstage and getTurn in the description for how an agent can determine what stage it is in.

When the server is in a specific state all disabled commands will return an ERR response (0). In addition to the competition stage the server state is also determined by which agent's turn it currently is in each pair. An agent who waiting for their turn will be in a waiting state. During the waiting states most commands other than information queries are disabled.

Commands available by Stage

While "**Bidding**", and **your turn**, your agent can send the following commands:

- doScreenShot
- clickInCentre
- fullyZoomOut
- fullyZoomIn
- bid

While "**Running**", and **your turn**, your agent can send the following commands:

- doScreenShot
- clickInCentre
- fullyZoomOut
- fullyZoomIn
- cshoot
- pshoot
- cFastshoot
- pFastshoot
- wasteBird

At any time an agent can send the following commands:

- getState
- getCompetitionState
- getTurn
- getOpponent
- getLevels
- getOpponentScore
- getScoreBoard
- getMyRoundScore

The Flow of a Game

- All agents start in the connecting stage
- The start button is pressed and each agent is paired with another agent.
- A game is created for each pair
- (Connecting to Loading) The games load the first level
- (Loading to Bidding) The game detects the level has been loaded
- The first agent has 30 seconds to make its bid
    - If no bid is made the default bid is 0
    - Once the agent has made its bid or the time runs out is the other agents turn
- The second agent has 30 seconds to make its bid
- (Bidding to Running) The game determines which agent bid higher and lets them shoot first
- The first agent has 30 seconds to make its shot
    - If no shot is made the wasteBird shot will be made for the agent
- (Running to PostShot) Once a shot has been made the server takes 15 seconds to check if the level was solved
    - (PostShot to Loading) If the level was solved the game loads the next level
    - (PostShot to Running) If the level was NOT solved it is now the second agents turn to shoot
    - (PostShot to Finished) If the final level was solved the pairing between these two agents is finished
- (Finished to Loading) Once all pairs of agents have finished their games the global scoreboard is updated, the agents are re-paired in a round robin (everyone plays everyone once)

- Once every agent has played every other agent the competition ends.
- The winner is the agent with the highest overall score.

Modelling the Flow of the Game

It is important for an agent to be able to model the game. This requires the agent to keep track of the change in competition stages. The agent can then determine when information becomes available such as when the global scoreboard is updated.

For the example of the provided naive agent implements the following:

- **Connecting** The game starts at this stage, once the game changes to either **Loading** or **Bidding** the agent can request: getLevels, getOpponent. It also knows its current level will be the first level returned by getLevels.

- **Bidding** Once the bidding stage has finished the agent can request getTurn() to discover whether or not it will be going first. This can be used to determine if the agent won its bid

- **PostShot/Running** If either of these stages changes back to **Loading** or **Bidding** then the current level has been completed. The agent knows the next level will be the next level returned by getLevels. The agent can request checkMyScore to get its own score and checkScore(opponent) to get its opponents score.

- **Finished** If this stage changes to either **Loading** or **Bidding** the agent knows the next round has started. The agent will be playing the first level again and can request getOpponent to see who it is playing against. The scoreboard will now be updated will the results of each game in the round and the agent can request getScoreBoard to see how its new opponent did in the previous round.

Protocol Changes

Description of New Commands

Bid

Amount is a 4 byte signed integer, negative bids are treated as bids to NOT go first, positive bids are treated as bids to go first.

MID	Amount

Return 1 or 0

1: The server has accepted the bid

0: The server has declined the bid (it is either not your turn, not in the bidding stage or you have already bid).

New shot command “Waste Bird”

MID

Return 4 byte integer: 1 or 0.

1: The server has wasted a bird for you

0: The server has not wasted a bird for you

Configure Message

An agent still needs to send this message to tell the server its team id. The response from the server will be the same as normal but the information is irrelevant (read and discard).

Query Configuration

Get Opponent

MID

Return:

Opponent ID

The agent will be returned with a single integer representing its current opponents ID. This id can be used to lookup their score in the global scoreboard.

Get Levels

MID

Return:

Level 1	Level 2	Level 3

Return is a byte array of fixed length (4 * 3). each four bytes represents a level number that will be played. Each pair of agents will play through level 1,2,3.

Query State Commands

Get Competition stage

MID

Return:

Competition stage

Returns 4 bytes representing an integer that maps to one of the following stages: Connecting(0), Bidding(1), Running(2), PostShot(3), Loading(4), Waiting(5), Finished(6)

Get Turn

MID

Return 4 byte integer: 1 or 2.
1: It is your turn
2: It is not your turn

Query Score Commands

Get Opponent Score

MID

Return:

Score on Level 1	Score on Level 2	Score on Level 3

Returns a byte array of fixed size (4 * 3) of the scores achieved by your opponent on each level in the current pairing.

Get My Round Score

MID

Return:

Score on Level 1	Score on Level 2	Score on Level 3

Returns a byte array of fixed size (4 * 3) of the scores achieved by the requesting agent on each level in the current pairing.

Get ScoreBoard

MID

Return:

agentId1,agentId2,…agentId8	Round1	Round2	...	Round7

Roundn:

agent1 score: 1,2,3	agent2 score: 1,2,3	….	agent8 score: 1,2,3

Returns:
The first section of the byte array contains the ids of each agent that is connected to the server. This section is of size 8 (the number of agents) * 4 (four-bytes integer id). The ids are numbers 0-7 indicating the order in which the agents connected to the server. these ids are consistent with the ids returned by getOppoent.
(However to indicate non present agents a 0 is zero, therefore the agent ids in this section are incremented by 1 and are in the range 1-8. Please decrement to get the id referred to by getOpponent)

The next 7 sections of the byte array represent the rounds. these sections take a total fixed size of 7 * 8 * 3 * 4. Each round is of size 8 (agents) * 3 (score for each level) * 4 (integer representing the score).

Summary of Removed Commands

11-30 Query Messages	
13	Get Best Scores
23	Get my score
14	Get the current level
31-50 In-Game Action Messages	
33	Sequence of shots [Safe mode]
43	Sequence of shots [Fast mode]
51-60 Level Selection Messages	
51	Load a level
52	Restart a level

Summary of New Commands

MID	Request	Format(byte[])	Return	Format(byte[])
53	bid	[53][Amount] Amount: 4 bytes integer	four bytes array. 0 or 1. 1 bid accepted. 0 rejected.	[bid_ok]
54	getOpponent	[54]	Four bytes array. converted to integer is the agent's current opponents ID	[ID]
55	getCompetitionstage	[55]	Four bytes array. converted to integer is the ordinal of the current competition stage	[Competition Stage]
56	getTurn	[56]	Four bytes array. 1 your turn. else not your turn.	[turn]
57	getOpponentScore	[57]	3 * 4 bytes array. agents current opponents score	[score1][score2] [score3]
58	getScoreBoard	[58]	8 * 5 byte array then 7 * 8 * 4 * 3 byte array. each (id+1) then each round, each agent,	[id1]..[id8] [round1 -> [agent1 -> [score1] [score2] [score3]]]

			each level -> score	
59	getLevels	[59]	3 * 4 byte array. an integer representing the level of poached eggs the agents will play on	[level1][level2] [level3]
60	wasteBird	[60]	OK/ERR	[1]/[0]
61	getMyRoundScore	[61]	3 * 4 bytes array. agents current score	[score1][score2] [score3]