AIC2021: Game description

1 Lore

The human race just discovered fire. Now, several clans fight for the dominance of a given region.

2 The Game

The aim of the game is to control the given zone either militarily or technologically. More precisely, the main objective of each faction is to either destroy the opponent's base or to discover the wheel. If none of these conditions is satisfies after 2000 game rounds, the winner is chosen according to the following criteria (in order):

1. The team that has more technological level.
2. The team that spent more resources on technologies.
3. The team whose sum of resources plus the value of all of its units is the highest.
4. Randomly.

3 Economy

Each team will have access to three types of resources: food, wood and stone. These resources are shared between all units. Each team starts with 100 units of each resource and receives two additional units of each at the beginning of each round. It is also possible to get additional resources with workers (at a rate of 5 resources/round) or with special types of structures.

4 Map

The map consists of a grid of dimensions between 20×20 and 50×50. Each unit occupies exactly one tile and all tiles of the map are accessible by every unit except if there is a mountain, water, or another unit. Each tile may also contain a Food, Wood or Stone deposit. These deposits can be mined by workers to obtain the corresponding resource at a rate of 5 resources/round.

Each unit has a finite vision range, which means that it can only sense the tiles that are close enough to its location. The vision range depends on the unit type (Section 6). Additionally, units cannot sense tiles whose illumination is less than their detection level. Moreover, they cannot detect anything on those tiles, and they can't perform certain kinds of actions on them (rock painting, gathering resources, etc.). For more info on this mechanic, see Section 5.

Each team starts with exactly one Base and it is guaranteed that all maps are symmetric. This symmetry may be horizontal, vertical or rotational. All tiles of a given map will have a fixed offset between 0 and 1000 (more precisely, the tile that would be $(0, 0)$ is marked as $(offset_x, offset_y)$ instead). This way units cannot guess the map edges without exploring.

5 Illumination

Each tile of the map has an illumination level that changes depending on the nearby light sources. Light sources can be one of the following:

- Structures (including the Base).
- Fires created by units.
- Torches carried by units.

Each light source illuminates nearby tiles with an intensity that decreases with the distance to the source and the opacity of the tiles. Tiles with mountains have opacity 100 (light can't pass through), forests have opacity 2, and all other tiles have opacity 1. The illumination of each tile is the maximum of the illumination induced by each light source. For more information about the exact formula, check Section 11.

6 Units and Structures

Each unit has a unique identifying number (ID) chosen randomly between 1 and 10.000. Whenever a new unit is created, there is a period of 10 turns in which the unit is still in construction and can't move, attack or perform any other action. Teams cannot control more than 75 units simultaneously.

There are six types of units: Workers, Explorers, Trappers, Axemen, Spearmen and Wolves. The parameters of each of these types are as indicated in the following table. All distances are shown in squared units. For instance, a unit with 12 attack range at $(0, 0)$ can attack a unit at $(2, 1)$ since $2^2 + 1^2 \leq 12$, but it cannot attack a unit at $(2, 3)$ since $2^2 + 3^2 > 12$. A $(*)$ indicates that this unit follows a special mechanic (see Section 8).

	Worker	Explorer	Trapper	Axeman	Spearman	Wolf
Wood Cost	10	30	20	40	100	30
Stone Cost	20	10	30	80	40	30
Food Cost	20	15	20	60	60	140
Maximum Health	10	10	15	70	20	120
Attack	4	–	–$^{(*)}$	15	5	2
Attack range	5	–	2	5	18	2
Minimum attack range	0	–	1	0	9	0
Attack cooldown	2	–	20	4	2	1
Vision range	25	50	16	20	32	32
Movement cooldown	2	1	3	4	4	2
Movement range	2	2	2	2	2	2
Detection level	11	5	11	10	10	$8^{(*)}$
Torch throw range$^{(*)}$	2	18	2	2	2	2

Besides the units above, there are additional unmovable units called structures. With the exception of the Base, all of them can be built by workers, although some of them require certain technologies.

	Base	Settlement	Barracks	Farm	Sawmill	Quarry
Wood cost	$-^{(*)}$	75	150	150	0	150
Stone cost	$-^{(*)}$	75	150	150	150	0
Food cost	$-^{(*)}$	0	0	0	150	150
Maximum Health	400	75	200	50	50	50
Attack	40	–	–	–	–	–
Attack range	18	–	–	–	–	–
Minimum attack range	0	–	–	–	–	–
Attack cooldown	2	–	–	–	–	–
Vision range	50	25	25	16	16	16
Detection level	0	8	8	11	11	11
Luminous intensity	20	16	16	16	16	16

6.1 Deer

Deer are neutral units that provide additional food. Each deer has 15 health points and 6 turns of movement cooldown. Whenever a deer dies, a food deposit is generated at its position with 500 units of food. However, this deposit loses 2 units of food each round. The food at these deposits can be gathered by workers.

The deer behavior is as follows: each round, each tile has a probability of $1/1000$ of spawning a deer whenever its illumination is smaller than 6. Deer move randomly as long as they stay on tiles with less than 6 illumination. If the deer does not have any adjacent tile with less than 6 illumination, it moves to the closest one on the map. If there is none, it moves randomly.

7 Movement, Attack and Cooldowns

Each unit has a movement and an attack cooldown. They can only move or attack when their respective cooldown is inferior to 1. In particular, if both cooldowns are inferior to 1, the unit can attack and move on the same turn.

Every time that the unit attacks or moves, it adds cooldown to its respective value. The amount added is the one given in the table in Section 6. If the unit moves in a diagonal direction, the movement cooldown added is multiplied by 1.4142, which is approximately $\sqrt{2}$. At the beginning of every turn, both cooldowns decrease by 1.

Units cannot attack through mountains. More precisely, if the segment that goes from the center of the unit's tile to the center of the attacking target passes through the interior of a tile with a mountain, the attack cannot be performed. However, it is allowed that the segment passes through the boundary of such tiles. For instance, a unit at $(0, 0)$ can attack a unit at $(1, 1)$ even if there is an obstacle at $(0, 1)$.

8 Special Mechanics

Whenever a unit kills another, its team receives half of the resources that the other unit was carrying.

All non-structure and non-wolf units can carry a torch and illuminate nearby tiles. To light a torch, the unit must be standing next to an adjacent light source (which may be a fire, a structure, or

another unit carrying a torch), and pay 10 units of wood. Torches last for 150 rounds. Units may also throw their torch at nearby tiles, creating a fire that lasts for the torch's remaining rounds. All torches and fires have a luminous intensity of 16.

Each unit also has the following mechanics:

Workers: They can build settlements, barracks (requires the *Military Training* technology), farms (requires the *Jobs* technology), sawmills (requires the *Jobs* technology), and quarries (requires the *Jobs* technology). Workers can gather resources from nearby sources as long as the source's illumination is greater than the worker's detection level, at a rate of 5 resources/round (if there is more than one resource at a given tile, it gets 5 units of each). A worker can carry up to 100 of each resource, and they must deposit them at settlements or bases to share them with the whole team.

Explorers: They can't attack, but they can throw their torch at a long distance.

Trappers: Whenever they attack, they create a trap at the target tile that kills anyone that steps on it. They can also disarm other traps.

Wolves:: They can always detect units inside their vision range, independently from the illumination. However, they cannot perform rock art or carry torches.

Base: It can create workers, explorers, trappers and wolves (requires the *Domestication* technology). It can research new technologies and workers can deposit their resources on it. If it dies, its team loses the game.

Settlements: It can create workers, explorers, trappers and wolves (requires the *Domestication* technology). Workers can deposit their resources on it.

Barracks: It can create Axemen and Spearmen.

Farms, Sawmills and Quarries: They provide one additional unit of food, wood and stone respectively.

9 Communication and Vision

Each unit (with the exception of wolves) cannot detect anything outside of its vision radius or anything on tiles with an illumination inferior to its detection level. If a tile is inside their vision radius and their illumination is high enough, its information can be sensed using the *UnitController* methods (check Section 13 for more information).

Units run independently and they don't share memory. This means that objects detected by a given unit might not be detected by others. However, units can communicate in two ways. First, they can perform Rock Art and write a number from 0 to 100000 on top of their tile. This number can then be read by other units (even from the opposite team!). Reading and writing on a given tile can only be performed if the illumination of the tile is higher than the unit's detection level. Initially, all tiles are painted with 0.

The other way for units to communicate is using smoke signals. If a unit has intrinsic illumination or is carrying a torch, it can broadcast an integer using a smoke signal. This integer will be visible by all units on the map. A smoke signal can only be created by a given unit if both of its cooldowns are smaller than 1. If created, it adds 10 to both cooldowns. The signal will last until the unit's next turn.

Both painting and signaling through smoke signals can be performed using the methods at *UnitController*.

10 Technologies

Each team has a technological level that increases with the amount of technologies researched. Teams cannot research any technology that requires more technology level than what they currently have. To increase from level N to $N+1$, the team must research at least 3 technologies that require level N. Each technology gives passive bonuses for the team that researches it (they can be researched by both teams).

10.1 Available technologies

This is the list of available technologies, sorted by the required technological level. The research cost is indicated between brackets.

Level 0

- **Domestication** [300 food, 50 wood, 50 stone]: The base and settlements can create wolves.
- **Military training** [150 wood, 150 stone, 100 food]: Workers can build Barracks.
- **Boxes** [300 wood, 100 stone]: Units can carry up to 200 units of each resource.
- **Rafts** [500 wood, 100 stone]: Units can enter tiles with water when moving. However, exiting a tile with water duplicates the movement cooldown added. Units still cannot created on water.
- **Rock art** [100 food, 100 wood, 200 stone]: Units can draw any integer using Rock Art. Allows units to sense the opponent's technology level and technologies.
- **Coin** [500 food, 100 wood, 100 stone]: Each round, the team gets an additional resource of each.
- **Utensils** [100 wood, 300 stone]: Each round, workers can gather up to 10 resources of each.

Level 1

- **Sharpeners** [400 wood, 600 stone]: New axemen and spearmen 20% more attack power.
- **Cooking** [400 food, 400 wood, 200 stone]: All new units have 20% more maximum health.
- **Eugenics** [200 food, 400 stone, 400 wood]: New wolves have 0.5 less movement cooldown and 20 more maximum health points (it is applied before Cooking). Allows wolves to read smoke signals.
- **Navigation** [500 wood, 500 stone]: Units no longer get additional cooldown when exiting tiles with water.
- **Jobs** [800 food, 200 wood, 200 stone]: Workers can create farms, sawmills and quarries.
- **Oil** [500 food, 500 wood]: New torches and fires can last up to 250 rounds.
- **Vocabulary** [500 food, 250 wood, 250 stone]: Signaling with smoke signals adds 2 cooldown of each instead of 10.
- **Huts** [500 wood, 500 stone]: The maximum number of units increases by 25.
- **Tactics** [350 food, 350 wood, 350 stone]: Whenever a new unit is created, the team gets refunded 20% of its cost (rounded down).

Level 2

- **Crystals** [1500 stone]: All new units detection level decreases by 2.
- **Combustion** [1500 wood]: All torches and fires created by this team have 2 additional luminous intensity.
- **Schools** [3500 food]: This team's technological level is set to 3.
- **Poison** [1500 food]: Each round, the opponent receives 1 less resource of each.
- **Expertise** [500 food, 500 wood, 500 stone]: All new units have 20% less attack cooldown.
- **Flint** [1500 stone]: Units no longer require to be adjacent to another light source to light torches.
- **Houses** [750 stone, 750 wood]: The maximum number of units increases by 25.

Level 3

- **The wheel** [1500 food, 1500 wood, 1500 stone]: This team wins the game.

11 Illumination formula

Intuitively, given a light source F, the illumination at a tile $A = (x, y)$ is given by the maximum of all tiles A' adjacent to (x, y) that are closer to the light source, where the maximum is computed after subtracting a quantity proportional to their opacity $O(A')$.

To define the exact formula, we need the following notation:

- Given a tile (x, y) and a direction $d \in \{N, S, E, W, NE, NO, SE, SW\}$ (where N denotes the North direction, S denotes South, etc.), we denote by $(x, y) + d$ the tile that's adjacent to (x, y) following direction d. For instance, $(4, 5) + N = (4, 6)$.
- Given a direction d, we denote by $\|d\|_D$ the *approximate norm* of d. Intuitively, the main directions (N, S, E, W) would have norm 1, while the remaining ones would have norm $\sqrt{2}$. However, to keep integer distances, we approximate them to 2 and 3 respectively. Thus, $\|d\|_D = 2$ for $d \in \{N, S, E, W\}$, and $\|d\|_D = 3$ for $d \in \{NE, SE, NW, SW\}$.
- Given a tile $A = (x, y)$, denote by $\|(x', y')\|_A$ the squared euclidean distance from (x, y) to (x', y').

With this notation, we can define the desired formula. Given a light source F at $A = (x_F, y_F)$ and intensity ℓ_F, the recursive formula that gives the intensity at $I_F(x, y)$ at (x, y) provided by F is the following:

$$I_F(x,y) = \begin{cases} \ell_F & \text{if } (x,y) = (x_F, y_F) \\ \max_d\{I_F((x,y)+d) - O((x,y)+d)\|d\|_D \mid \|(x,y)+d\|_A < \|(x,y)\|_A\} & \text{otherwise,} \end{cases}$$

where the maximum is chosen between all possible directions d, and $O((x, y)+d)$ denotes the opacity of tile $(x, y)+d$ (this value is 100 for mountains, 2 for forests and 1 for all other tiles). The following diagram shows how a map with no mountains or forests is illuminated by a light source of intensity 10.

			0	0	0	0	0			
	0	0	0	1	2	1	0	0	0	
	0	1	2	3	4	3	2	1	0	
0	0	2	4	5	6	5	4	2	0	0
0	1	3	5	7	8	7	5	3	1	0
0	2	4	6	8	10	8	6	4	2	0
0	1	3	5	7	8	7	5	3	1	0
0	0	2	4	5	6	5	4	2	0	0
	0	1	2	3	4	3	2	1	0	
	0	0	0	1	2	1	0	0	0	
			0	0	0	0	0			

If a tile is illuminated by several light sources, the illumination of the tile is the maximum of the illumination provided by each of these light sources.

12 Energy

Energy is an approximate indicator of the number of basic instructions that each unit performs. More precisely, each bytecode instruction performed by a given unit consumes one unit of energy (except internal operations of the methods provided in the documentation, these consume a constant amount of bytecode which is given in the documentation). For users not familiarized with Java, it is not necessary to know how bytecode works, however it is good to have in mind that it is somewhat proportional to the number of code instructions. The amount of energy consumed up to a certain instruction can be accessed at any time using the methods in *UnitController*. Whenever a unit surpasses the amount of energy allowed (currently set to 15000), the unit pauses and continues running the remaining instructions during its next turn.

13 User Instructions

Players must fill the *run* method of the *UnitPlayer* class, which is run independently by all units of the player's team. This function has a *UnitController* as input, which is used to give orders to the given unit and to get information of the visible tiles or the common array, among others (check the documentation for more details).

Run does the following: Whenever a new unit is created and finishes its construction period (10 turns), its *run* method is executed until either the unit finishes its turn by calling the *yield* method, or when it surpasses the amount of allowed energy. If a unit returns from its run method, it dies. Because of this it is suggested to keep its instructions inside a *while(true)* statement, and to finish each iteration with a call to *yield* (check *nullplayer* and *demoplayer*).

14 Implementation info.

You may skip this section if you're not enough familiarized with Java.

Each unit is run in an independent thread. Each of these threads gets reactivated every time the unit is scheduled (once every round, following the same relative order every round). For safety reasons, it is forbidden to use any method or class outside *java/lang*, *java/math* and *java/util* (and some of the sub-classes of these). It is also forbidden to use static variables (which include switch statements, since they internally do so).

Even though codes that break these rules are automatically detected the instrumenter, we are going to check all the finalists' codes manually to be sure that everyone is playing a fair game. If a team breaks any of these rules without informing any of the devs, it will be disqualified.