

GALAXY TRUCKER

Presentation and Design Choices

Developed by:

Alessandro Pasquetto
Lorenzo Orlandi
Gabriele Pedesini
Stefano Molteni

MVC PATTERN

Server	Client
<p>Model</p> <p>Contains most of the game logic and is responsible for maintaining the state and information related to the various ongoing matches.</p> <p>Controller</p> <p>Handles the orchestration of the different game phases, and manages the requests and responses of the various users.</p>	<p>View</p> <p>Whether it's a TUI or a GUI, it contains the logic for creating the user interface and managing all possible user interactions.</p> <p>Model</p> <p>A more lightweight model whose purpose is to maintain a consistent local state on the client side, in order to minimize the amount of data exchanged with the server.</p>

On both the server and client sides, dedicated classes are implemented to manage communication using Socket and RMI protocols.



IMPLEMENTED FUNCTIONALITIES

Rules

- ✓ Basic rules
- ✓ Complete rules

Connection

- ✓ Socket
- ✓ RMI

User Interface

- ✓ TUI
- ✓ GUI

Additional Functionalities

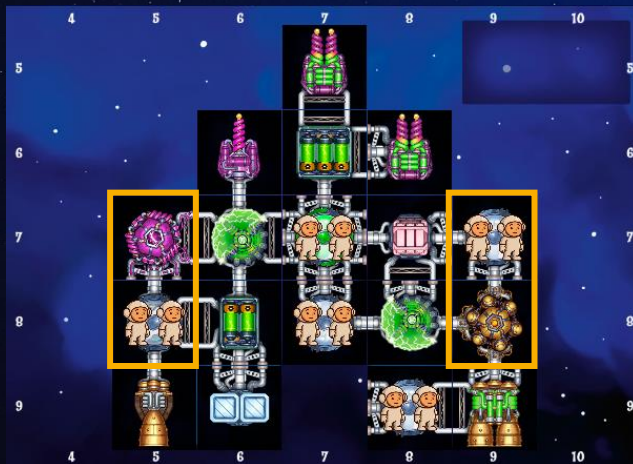
- ✓ Demo travel
- ✓ Multiple games
- ✓ Resilience
- ✗ Persistence

DEMO TRAVEL

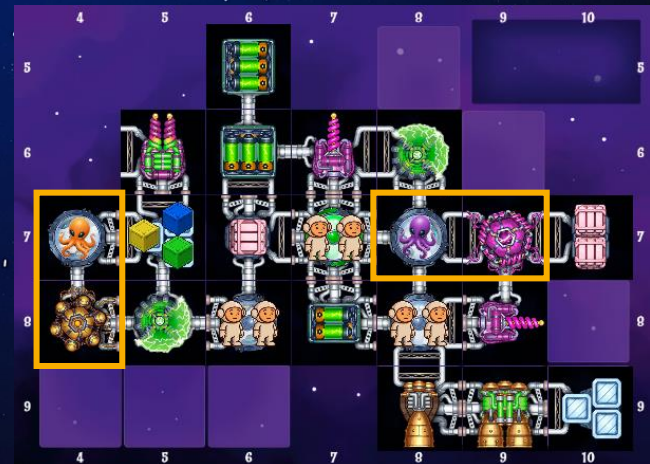
To manage the distinction between demo travels and level 2 travels, we introduced a **level attribute** in the *Game* class.

This attribute allows us to define the correct behaviour to apply in various gameplay situations where there is a functional difference between the two types of games.

Demo travel



Level 2 travel



MULTIPLE GAMES

To support the coexistence of multiple games running simultaneously on the same server machine, we introduced a helper class called *GameManagerMaps*. This class contains a **map of GameManager** instances, each associated with a specific **game ID** as the key. Each game is therefore linked to its own *GameManager*, which is responsible for storing useful information about the game.

Every communication between client and server is labelled with the ID of the game the user is currently participating in. This enables the system to forward the message to the correct *GameManager*.

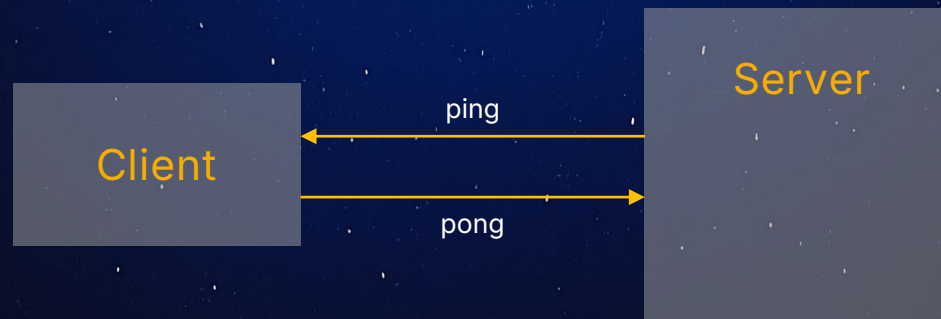


RESILIENCE (1)

A **ThreadPinger** sends a ping every second to all connected clients (both via Socket and RMI).

The client is considered **disconnected** if it doesn't respond to two consecutive pings.

If a client does not receive a ping from server for 3 seconds, it will be forwarded to connection page.



RESILIENCE (2)

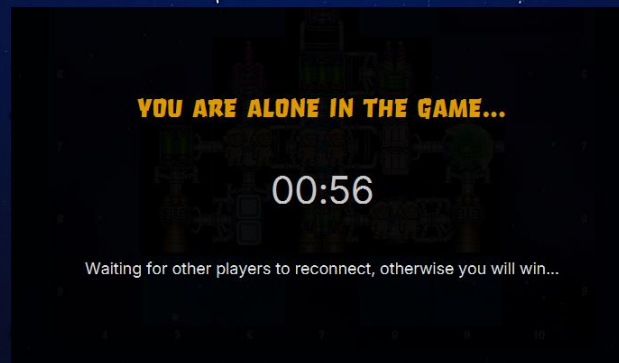
Disconnection is handled differently based on the game phase:

- Lobby Phase

The disconnected player is removed from the lobby list.

- Game Phase

Player is moved from the active players list to a disconnected list. If only one active player remains, the game enters **freeze mode**. Disconnected players are excluded from "ready" checks, and the game proceeds with **automatic decisions** for them.



←
Freeze mode

RESILIENCE (3)

Reconnection

If a player reconnects during a game, they are restored to the active list.

If they were part of an ongoing event, a specific reconnection handler is invoked (each event card provides one), ensuring the player can resume participation, including re-sending pending questions.

