1 Design Patterns

The game uses Godot's build-in architectural design pattern, which, compared to a classical ECS, uses Nodes instead of entities with components, favours inheritance over composition, while still allowing the later to be used on a higher level, and, similar to systems in ECS, uses what the Godot developers call servers and lets Nodes have their own logic in the form of scripts.

This architectural pattern was mainly chosen because Godot was build around this pattern, and thus being well integrated with Godot's design philosophy, making it very userfriendly and leightweight, while still providing good performance. Moreover, as there is no destinction between scenes and prefabs - everything is a node in godot - it become easier to combine nodes and therefore to reuse parts of the game. Considering the complexity of the game there should not be any performance issues compared to a traditional ECS approach, however if optimization was required in the future, it would easily be possible to do so, for instance by only processing visible objects or by directly interfacing with Godot's low level servers for critical game code.

This design pattern can best be found in the Client project, and in a future version it would be possible to also find it in the Services project if a server-side map was added.

Another pattern I used is ..