

Architecture

This is a work in progress, many things can be improved, the UI is just a draft and it should be implemented using TextMesh Pro once that the gameplay is finished.

--Glossary--

Game entity: a physical object that interacts with other entities (Asteroid, Rocket, Spaceship)

Stats: speed, max ammunition, lifetime...

EntityDatas: ScriptableObjects that contain the stats and references to the prefabs needed to create an EntityState. They are similar to the Model in a MVC architecture.

EntityStates: GameObjects classes that are created using EntityDatas. They handle their interactions with other EntityStates, they also contain their current stats (that are initialized with the EntityDatas stats). They are similar to the Controller in a MVC architecture.

--Architecture explanation--

For the entities and guns, I choose to separate their 3D models and art in prefabs, their stats and references to these 3D models in EntityDatas and their behaviors in EntityStates.

This allows me to have the art separated from the stats values. The heavyweight asset bundles made of prefabs, materials and 3D models are separated from the lightweight asset bundles made just of ScriptableObjects. In this way, I can balance, tweak the game and add more content easily.

This architecture respects the single responsibility principle creating something similar to a MVC architecture. Every EntityState is the only responsible for its behavior, they don't know who has created them or how.

To communicate EntityStates with other systems that are not EntityStates I use Pub/Sub system, they send messages but they don't know who receives them. This messages system, allows me to avoid the use of patterns like Singleton or static classes that couple the code. Every part of the game is totally independent of the others, allowing me to create unit tests easily.

**Best regards,
Michael**