# Ingame Ship Builder

--- SINCRESS ---

This is the documentation of the **Ingame Ship Builder** asset for Unity3D. This document describes the example scene, the provided assets and the scripts of the framework. The code is self-documenting and the scripts are well commented so the documentation only provides a high level overview. Thank you for selecting the *Ingame Ship Builder* for your project*,* I hope you enjoy using this package and adding features to it.

## 1. Features

This asset provides solution for building ships from modular components ingame and for converting them into a physically simulated model of a spaceship (which has colliders and rigidbodies and exists within Unity's physics engine) which the player can control either by using mouse and keyboard or only the keyboard. Furthermore, this ship model can easily be expanded on by adding other modules such as ship equipment, weapon systems and other. The builder is also simple to use and to add features to, and your models can be converted into modular hulls and components for ingame building easily.

Here's what this asset provides:

- Flight scene where you can test built ships
- Rigidbody physics driven ship model
- Modular ship scripting enables easy expansion and adding of features
- Two control modes for players: mouse and keyboard flight
- Engine kill for turning the engines on and off (gliding)
- Supercruise speed with visual effect
- 3D heads up display with inertia
- Game proven mechanics and a well-polished framework
- Two textured ship models with a dozen components/modules

- Ship builder scene with UI controls
- Three stages of construction: load or create a new ship, build and save or test
- Binary saving and loading of ship creations
- Hardpoint system for building where *components* are placed on *hulls*
- Quick to setup, easy to use

# 2. Flight Mechanics

Let’s follow up with a short overview of the main classes which correspond to the principal components mentioned above:

The **Ship** class is the main part which ties all the ship components together. It’s used for communication between the ship components and holding data principal to this ship instance. There is a ScriptableObject called **ModelInfo** which contains information related to the specific ship model, such as thrust values, ship model name and the distance at which the camera follows the ship. This is where you would put other model paremeters such as armor value, cargo space, equipment slots and similar. Other variables are also held in the *Ship* class if they should be commonly accessed by other game features. Currently it connects the **ShipEngines** and the **ShipCockpitControls** components. The Ship class is where you would want to tie in your main ship functionality and add other ship features such as ShipWeapons, ShipPowerups or others.

**ShipEngines** deals with applying forces to the ship as a result of player’s throttle and steering input. It also defines how the ship handles: the speed and maneuovrability of the vessel. The player can seamlessly apply both mouse and keyboard steering input through the *ShipCockpitControls* class and the other controls are tied to the keyboard. Joystick or other types of control can be added easily.

**ShipCockpitControls** is the class responsible for handling player input and notifiying other components. The *useMouseInput* variable is used to distinct which flight mode is currently engaged. Along with mouse steering controls and keyboard controls for yaw, pitch and roll, this class processes other player input, such as toggling the engines and the supercruise speed.

## 3. Scenes

The **Builder** scene has the ship constructor interface and logic. The user interface has a left side menu with three stages: ***select hull***, ***add parts*** and ***test flight***. Each panel has controls relevant for the various build stages. The **Canvas** uses a BuildController script which controls the transitions between panels.

First Panel is responsible for automatically loading the autosaved draft from the previous build. You can also manually load a ship creation from the Ships folder, or load up an empty hull to create a new ship. The second panel offers a dropdown with components which can be attached and removed from selected hardpoints. To select a hardpoint, click on it. **Right mouse** controls the orbit camera. Third panel contains the Test Flight button which opens up the Flight scene with your created ship. Here you can also save your creation to the Ships folder, with the specified name. It’s easy to add your own custom design or more features to each of the panels.

The **Flight** scene contains the flight model for ships as well as a HUD. The main camera is in chase mode behind the player ship and can be switched to orbit mode. An ambiental starfield is attached. The **Canvas** displays the 2D user interface: the mouse cursor and static crosshair. There is also a 3D Heads up display which shows speed and throttle as well as flight mode information. An **AsteroidSpawner** is used to provide a backdrop to the flight scene, by spawning random asteroid prefabs which are also rigidbodies. There is a **Main Camera** with associated pursuit camera properties, as well as the orbit camera controls.

## 4. Model configuration

A hull uses **hardpoints** to determine where a component is placed. These are prefabs which the *ShipConstructionHull* script will pick up as children of the Hull which can be selected by clicking on them. Once a component has been selected to be attached to the hardpoint, its position and rotation are copied from the hardpoint. This is why it’s important that your components are made in such a way that the **origin of the model is at the point where it will be attached to the hull**. To do this in Blender, for example, simply place origin in the middle of the coordinate system (move the object to 0,0,0) and then drag the mesh to where you feel it should attach to the hull. Nested components are not supported by this asset but can be easily added if you wish to do so. Also, it should be easy to allow the user to rotate components or scale them.

I hope you find this package helpful and easy to use in your project. If you have questions or feedback please contact me at sincress@gmail.com. Please rate this Asset on the Unity Asset Store, it means a lot.

Sincress