Universal Cargo System for Orbiter

## Universal Cargo System for Orbiter (UCSO) is a cargo management system for Orbiter 2016, designed to replace UCGO. New cargoes can be added without any programming via configuration files. It provides an API for vessel authors to add UCSO support to their vessels.

Installation

### Simply unpack the ZIP file into the Orbiter root folder. You can use [Orbiter Addons Manager](https://www.orbithangar.com/searchid.php?ID=7213) developed by me to manage all of your add-ons, not just UCSO.

### You need [ShuttlePB UCSO](https://github.com/abdullah-radwan/ShuttlePB_UCSO/releases) to run the scenarios. Make sure to match the version.

Configuration

### The UCSO configuration file is ‘Config\UCSO\_Config.cfg’. You can open it with Notepad. All options are explained there.

Cargoes type

### There are 3 types of cargoes in UCS: Static, resource and unpackable cargoes. There is one example for each cargo in the default distribution.

### A static cargo, as its name employs, is static. It can’t be unpacked nor used by vessels. The example of this type is ‘CargoContainer’ cargo.

### A resource cargo contains a resource which can be used by vessels if the vessel supports. The resource types aren’t limited, and it’s not case-sensitive. You can include white spaces in the name.

### It’s recommended to use standard keywords below, so the vessel authors can use it also:

#### Fuel, Oxygen, SCRAM Fuel, APU Fuel, Food, Water, Hydrogen.

### The example cargo for this is ‘CargoFuel’ cargo, which contains 1000 kilograms of fuel.

### An unpackable cargo is a cargo which can be unpacked. When unpacked, the cargo will spawn an object (e.g. a satellite). There are 3 unpack modes: ’Landed’, ‘Delayed’, and ‘Manual’.

### Landed means the cargo will be automatically unpacked when it touches the ground. Delayed means the cargo will be unpacked when it reaches the set delay. Manual means the cargo has to be unpacked manually by an object (e.g. a vessel).

### The example for this is ‘CargoShuttlePB’ cargo, which spawns a ShuttlePB vessel after 25 seconds.

Cargo creation

### No programming is required to make a cargo.

### To create a cargo, you need to create a configuration file in the ‘Config\Vessels\UCSO’ folder. Copy the example for the cargo type you want, then modify it.

### For the mesh, you can use the default mesh, which is ‘UCSO\UCSO\_Cargo’. If you want to create a mesh, you must follow the following restrictions:

### The mesh bounding box must be 1.3mx1.3mx1.3m.

### The mesh bottom must be at -0.65 meter.

### The polygons should be as low as possible, as the cargoes can be massive (e.g. XR-05 can carry more than 300 cargoes). For a 1000 polygons mesh, the total polygons can be as high as 300,000 polygons! This is a frame rate killer. The recommended polygons are between 50 and 200, with one 512x512 or 1024x1024 texture.

### After finishing, place the mesh in ‘Meshes\UCSO’ folder, and the texture in ‘Textures\UCSO’ folder, then modify the configuration file accordingly. The default mesh Blender file can be found on [GitHub](https://github.com/abdullah-radwan/UCSO/blob/master/Sources/Meshes/UCSO_Cargo.blend).

### For the mass, UCSO will automatically add the container weight for the cargo (The default is 85 kilograms, which can be modified in UCSO configuration file).

### The configuration, mesh, and texture filenames must be unique, to avoid conflicts with other add-ons. For example, if your cargo is fuel and your name is Suzan, a good name is SZ12CargoFuel.

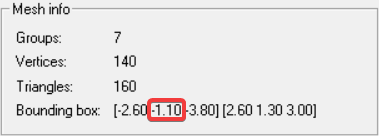
### If you’re making a resource cargo, use the standard resource keywords detailed above, so vessels can use it also.

### If you’re making an unpackable cargo, here are some notes:

### UCSO will add numbers to the spawn name if there are vessels with the same name.

### The spawn module is the vessel configuration file path as viewed from ‘Config\Vessels’ folder without ‘.cfg’. For example, DeltaGlider path is ‘Deltaglider’.

### The spawn height is the vessel height if released on the ground. To get it, open the vessel mesh with ‘Shipedit’ tool which can be found in ‘Orbitersdk\utils’ folder. The height is the absolute value for the negative value in the center in ‘Bounding box’. For ShuttlePB, this value is 1.10.



UCSO API

### To support UCSO in your vessel, you need to use the C++ API.

### First, you need to set your project. If not already done, you need to import Orbiter SDK properties. In your project .’vcxproj’ file, add the following lines:

<ImportGroup Condition="'$(Configuration)'=='Release'" Label="PropertySheets">

<Import Project="$(ProjectDir)’RELATIVE PATH’resources\Orbiter vessel.props" />

</ImportGroup>

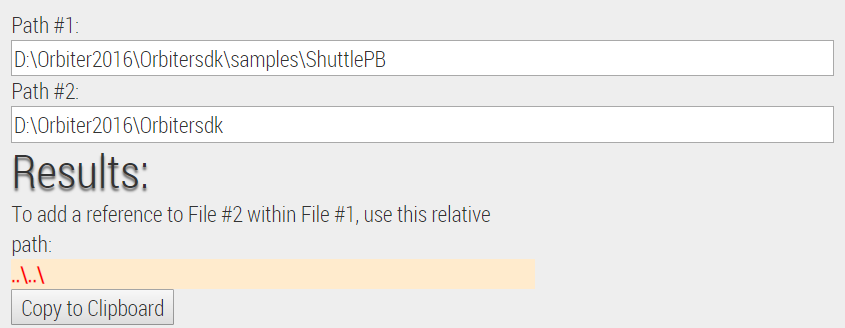
<ImportGroup Condition="'$(Configuration)'=='Debug'" Label="PropertySheets">

<Import Project="$(ProjectDir)’RELATIVE PATH’resources\Orbiter vessel.props" />

<Import Project="$(ProjectDir)’RELATIVE PATH’resources\Orbiter debug.props" />

</ImportGroup>

### You need to replace ‘RELATIVE PATH’ with the relative path from your project folder to ‘Orbitersdk’ folder. You can use [this calculator](https://www.stevebreese.com/Relative-Path-Calculator). Insert the project folder path in the first box, and the ‘Orbitersdk’ folder in the second box, then copy the first result and replace ‘RELATIVE PATH’ with it. Here is an example with the ShuttlePB project:

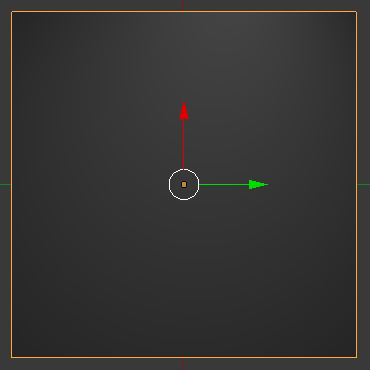


### The full path should be "$(ProjectDir)..\..\resources\Orbiter vessel.props". Do that for all projects if you have multiple projects in one solution.

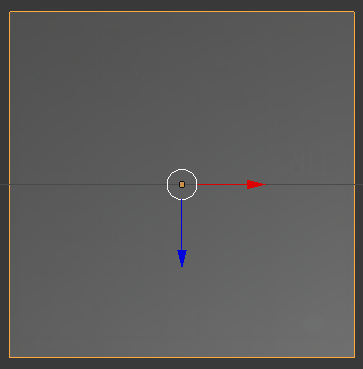
### Use UCSO by including ‘UCSO\_API.h’ file and linking against ‘UCSO\_API.lib’ file. The API methods are explained in ‘UCSO\_API.h’ file.

Attachment points

### The attachment point for UCSO cargo is at the center of the cargo bottom. You need to set your attachment point accordingly.



### For ShuttlePB UCSO, the attachment is set below the vessel, and it’s rotated 180 degrees so it’s Y-axis points down. You need to rotate it properly, as the release velocity if released in space will be added in Y-axis. If it’s rotated up, the cargo will pass through the vessel.



### A very handy tool is [Mesh Wizard](https://www.orbithangar.com/searchid.php?ID=2740), which will show you the attachment point axis. In order to run the program, you need to download [MSVBVM50.DLL](https://www.dll-files.com/msvbvm50.dll.html) and place it in the program folder, then open the command prompt as an administrator in the program folder, and execute the following commands:

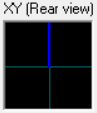
regsvr32 COMCTL32.OCX

regsvr32 MSFLXGRD.OCX

h

### Then open the program, it should open without problems. Open the ROT vector calculator from the ‘Calculator’ menu. For a normal attachment, the DIR is 0 0 1, and the ROT is 0 1 0.

### The XY or the rearview blue line represents the direction of release velocity. If it points up, the cargo will move up, and vice versa.



Credits

### [Fred18](https://www.orbiter-forum.com/member.php?u=8871) for the ground release rotation and touchdown points code.

### [Woo482](https://www.orbiter-forum.com/member.php?u=195) for the ground release location code.

### [Hasnat Ahmed Khan](https://sketchfab.com/3d-models/container-92bd84031ebc4ddcbf3b3d3689c4bf31) for the cargo 3D model. The model was modified.

About

### UCSO is a free opensource cargo management system for Orbiter. It’s under GPL 3 license. The source code can be found on the [UCSO GitHub repository](https://github.com/abdullah-radwan/UCSO).

### All contribution are appreciated.

### Copyright © Abdullah Radwan