Spacecraft support version 030115 By Vinka

Introduction

The spacecraft.dll module gives support to spacecraft that can be configured through initialisation file.
Most of the conventional spacecrafts will be supported by this module. You will be able to create your own spacecraft without the need to write a specific DLL.

Limitation

-maximum of 10 spacecrafts in the same scenario.
-maximum of 16 thrusters exhaust rendering for main engines
-maximum of 16 thrusters exhaust rendering for hover engines
-maximum of 16 thrusters exhaust rendering for retro engines
-maximum of 64 thrusters exhaust rendering for attitude engines
-maximum of 10 payloads
-maximum of 16 docking ports
-maximum of 8 animation sequence
-maximum of 128 components for the animation sequences
-maximum of 1024 groups for the animation sequences

Module Command keys :

'j' : jettison the payloads
Keys for animation sequence must be defined in the ini file (recognized keys are K, G, left-shift 0,1,2, …, 9

THE BEST WAY TO UNDERSTAND IT ALL, IS TO HAVE A LOOK AT THE EXAMPLE DELIVERED.

Files needed :

Modules\spacecraft.dll : the module implementing the generic spacecraft class
Config\spacecraft\spacecraft.cfg : the class configuration file

Config\spacecraft\<vessel name>.ini : the ini file which define the parameters of the spacecraft called "vessel name" in the scenario file (see below)

Scenario File format

The scenario file must define the ship as follow :
…
BEGIN SHIP
vessel name:spacecraft\spacecraft
…

The spacecraft module will search a ini file with the name of the vessel defined in the scenario file only in the config\spacecraft directory.

The following lines must be added to the scenario file

CONFIGURATION 0 (=launch, 1=in flight, 2=landed)

CURRENT_PAYLOAD (current payload not jettison yet)
SEQ 0 –2 0.0 (animation sequence 0 in state –2 (pause) at value 0.0)
SEQ 1 1 0.0 (animation sequence 1 in state 1 (run) at value 0.0)

The Load/Save scenarios seems to be working correctly

Initialisation File format

This file follow the format of the standard windows ini files. The following sections and items can/must be defined :

[LIFT]
lift function definition (if not defined, no lift will be produced)

- PT0=(-180,0)
- PT1=(-60,0)
- PT2=(-30,-0.2)
- PT3=(-1,0)
- PT4=(15,0.8)
- PT5=(20,1)
- PT6=(25,0.8)
- PT7=(60,0)
- PT8=(180,0)

PT0 to PT8 can be defined the angle must range from –180° to 180°, the second parameter is the lift coefficient

[CONFIG]
spacecraft configuration parameters section, most of these parameters correspond to the definition given by Martin schweiger in is documentation. Not all the parameters must be defined, other parameters will get default values when not specified.

- MESHNAME="atlantis" : the mesh name to be loaded
- SIZE=19.6 : the vessel radius size (m)
- EMPTY_MASS=104326 : the vessel empty mass (kg)
- FUEL_MASS=20000 : the vessel fuel mass (kg)
- MAIN_THRUST=53400 : the vessel main engine thrust (N)
- RETRO_THRUST=0 : the vessel retro engine thrust (N)
- HOVER_THRUST=0 : the vessel hover engine thrust (N)
- ATTITUDE_THRUST=7740 : the vessel attitude engine thrust (N)
- ISP=5000 : the fuel specific impulsion (N/kg/sec)
- PMI=(78.2,82.1,10.7) : the principal moment of inertia (kgm^2)
- CW_Z_POS=0.2 : the drag coefficient on the z axis when the vessel is moving forward
- CW_Z_NEG=0.5 : the drag coefficient on the z axis when the vessel is moving backward
- CW_X=1.5 : the drag coefficient on the x axis
- CW_Y=1.5 : the drag coefficient on the y axis
- CROSS_SECTION=(234.8,389.1,68.2) : the cross section in x,y,z axis (m^2)
- COG=8 : the center of gravity of the vessel above ground when landed (m)
- PITCH_MOMENT_SCALE=0.00001 : the pith moment scale
- BANK_MOMENT_SCALE=0.00002 : the bank moment scale

- ROT_DRAG=(0.5,1.0,1.0) : the rotational drag
- WING_ASPECT=0.7 : the wing aspect
- WING_EFFECTIVENESS=2.5 : the wing effectiveness
- LAUNCH_PT1=(1,0,0), LAUNCH_PT2=(-1,0,0), LAUNCH_PT3 = (0,0,1) : the coordinates of the touchdown points in launch configuration
- LAND_PT1=(1,0,0), LAND_PT2=(-1,0,0), LAND_PT3 = (0,0,1) : the coordinates of the touchdown points in landing configuration. If you want both launch and land to be the same, just define LAND_PT1,2,3. The landing configuration is automatically selected after launch when a altitude of 100 meter is reached.
- FOCUS=1 : specify which payload will get the focus when jettisoned, to keep the focus on the mothercraft specify –1. Payloads valid values start at 0 and goes to number of payloads-1.
- VISIBLE=1 : specify if the internal mesh rendering must be done in cockpit view
- CAMERA=(0,0,0) : specify the position of the point of view in cockpit view (should correspond to the cockpit position in the mesh). The default value will set the cockpit at the front of vessel.

[DOCK_0]
…
[DOCK_15]
docking ports list (maximum 16 docking ports per vessel)

- POS=(0,2.44,10.44) : the docking port position
- DIR=(0,1,0) : the docking port approach direction
- ROT=(0,0,-1) : the docking port longitudinal rotation alignment vector

[EX_MAIN_0]
…
[EX_MAIN_15]
main engine exhaust rendering definition sections (as many section as engines rendering are required)

- OFF=(-2.05,3.45,-14.2) : the offset of the rendering
- DIR=(-0.050, 0.099, -0.994) : the direction of the rendering
- LENGTH=4 : the length of the rendering at full thrust
- WIDTH=0.5 : the width of the rendering at full thrust

[EX_RETRO_0]
…
[EX_RETRO_15]
same as EX_MAIN but for retro engines

[EX_HOVER_0]
…
[EX_HOVER_15]
same as EX_MAIN but for hover engines

[EX_ATT_0]
…
[EX_ATT_63]

attitude engines exhaust rendering definition sections

- OFF=(0,1.6,17) : the offset of the rendering
- DIR=(0,1,0) : the direction of the rendering
- LENGTH=0.3 : the length of the rendering at full thrust
- WIDTH=0.3 : the width of the rendering at full thrust
- ROT_AXIS=X : the rotation axis (X, Y or Z)
- ROT_CW=1 : the rotation direction (0 or 1)
- LIN_AXIS=Y : the translational axis (X, Y or Z)
- LIN_CW=1 : the translational direction (0 or 1)

Note that the ROT_ parameters or the LIN_ parameters may be missing as the attitude engine can only be used for translation or rotation respectively.

[ANIM_SEQ_0]
…
[ANIM_SEQ_7]
animation sequence definition sections, this corresponds to the definition of the ANIM_SEQ given by Martin in the sdk documentation

- KEY=3 : the keypad key that will start the animation sequence. Recognized keys are K (usually used for payload bay open/close), G (for gear up/down), left-shift 0, 1, 2, …, 9
- DURATION=10 : the duration of the sequence
- REPEAT : repeat mode (0=no repeat (default), 1=repeat by restarting cycle (0->1;0->1;0->…), 2=repeat by reversing cycle (0->1->0->1->0->…)

In normal mode, the animation sequence will generate a value from 0.0 to 1.0 while in running mode (1). The value will change in 'duration' seconds. Once the value 1.0 is reached, the simulation will go in pause mode (-2). Restarting the animation by pressing a key will turn the pause mode (-2) in running mode (-1). The value will now change from 1.0 to 0.0. Pressing the key will in running mode will revert the running mode (1->-1 or –1->1). In the scenario file you specify the sequence number, the running mode (1,2,-1,-2) and the initial value between 0.0 and 1.0. The mesh design must always represent the moving parts in one of the extreme position. If you want to have the original mesh position at Orbiter start, you should configure the scenario file with <seq num> 2 0.0. If you want the other extreme position, you should configure the scenario file with <seq num> -2 1.0

The same key can be used to start/stop one or more sequences.

[ANIM_COMP_0]
…
[ANIM_COMP_127]
animation component definition sections.

FOR ROTATION :

- SEQ=0 : the animation sequence containing this component
- GROUPS=91, 92, 93 : the mesh group list that must be animated
- RANGE=(0.0,1.0) : the range value when the animation takes place within the sequence (between 0 and 1)
- TYPE=ROTATION : specify that the animation is a rotation

- ROT_PNT=(-2.8,1.35,0.0) : the point of rotation (one point where the axis of rotation passes through it)
- ROT_AXIS=(0,0,1) : the rotation axis
- ANGLE0=0. : the angle of rotation that is applied from 0 to begin of range
- ANGLE=160. : the angle of rotation (negative value for rotation in other direction) that is applied from end of range to 1. A linear transformation of the angle value is computed from angle0 to angle within the range defined.

FOR TRANSLATION

- SEQ=3 : the animation sequence containing this component
- GROUPS=6 : the mesh group list that must be animated
- RANGE=(0.,1.) : the range value when the animation takes place within the sequence (between 0 and 1)
- TYPE=TRANSLATE : specify that the animation is a translation
- SHIFT0=(0,0,0) : the translation value that is applied from 0 to begin of range on the three axis X, Y, Z.
- SHIFT=(0,-0.8,0) : the translation value that is applied from end of range to 1. A linear transformation of the translation components is computed from begin of range to end of range.

FOR SCALING

- SEQ=3 : the animation sequence containing this component
- GROUPS=6 : the mesh group list that must be animated
- RANGE=(0.,1.) : the range value when the animation takes place within the sequence (between 0 and 1)
- TYPE=SCALE : specify that the animation is a scaling
- SCALE0=(1, 1,1) : the original values of scaling that are applied from 0 to begin of range on the 3 axis X, Y, Z.
- SCALE=(0.1,0.1,1) : the final values of scaling that are applied from end of range to 1, a linear transformation of the values is performed from the value of scale0 to scale in the range defined.
- REF=(-0.340,-3.1855,0) : the reference point to apply the scaling (fixed point where the mesh seems to be scaled around).

[PAYLOAD_0]
…
[PAYLOAD_9]
payloads definition sections

- MESHNAME=Carina : the mesh name of the payload for rendering
- NAME=Carina-1 : the name of the vessel that will be created when the payload is released
- OFF=(0,1,2) : the offset position of the payload in the spacecraft
- MASS=1200 : the total mass of the payload
- MODULE=Carina : the vessel class of the payload
- SPEED=(0,1,0) : the release speed (relative to the spacecraft axis) in m/s
- ROT_SPEED=(0.3,0,0) : the release rotation speed (relative to the spacecraft axis) in rad/s

Composition of movements :
the animation components are transformed in the order of their declaration in the ini file. Suppose you have solar panel made of 3 parts, you should rotate the outer part first then the middle part (and the outer part) then the inner part (and the middle part and the outer part). As a general rule if you want to compose movements declare first the components with the less groups and last the components with the more groups. It is a little bit hard to explain this, looking at the example Animations will maybe make it more understandable.

Examples

The following examples can be downloaded with the spacecraft add-on

Spacecraft – space shuttle : The space shuttle Atlantis in orbital configuration. The parameters for atmospheric flight and for gear and payload bay animation are the same as the original source code from Martin Schweiger. The payload bay contains three small payloads that can be jettison (J key). The payload bay is commanded with 'K' key and the landing gear by 'G' key.

Spacecraft – Deltaglider : The Delta glider in orbital configuration. The parameters for the atmospheric flight and animations are the same as the original source code from Martin Schweiger. 'K' key commands the nose cone, 'G' key commands the landing gear, left-shift 0 key commands the air lock.

Spacecraft – Deltaglider landed : The Delta glider landed at the Cape. This is equivalent to the original scenario "Cape Canaveral" of Orbiter. 'K' key commands the nose cone, 'G' key commands the landing gear, left-shift 0 key commands the air lock.

Spacecraft – SCA747 : you should download the required files and instruction from directly from htttp://gdhp.tripod.com/Orbiter (by Damir "slat" Gulesich)

Spacecraft – Horizontal take off Vertical landing : The Delta glider on the space shuttle landing facility ready to take off. Going over 100 m altitude will arm the vertical landing.

Spacecraft – Vertical take off Horizontal landing : The Delta glider landed vertically, ready for take off. Going over 100 m altitude will arm the horizontal landing.

Spacecraft – Multiple docking : two deltagliders docked to a station module with 2 docking ports. All three vessels are defined from spacecraft.dll.

Spacecraft – Animations : a space platform demonstrating various animations possibilities.
shift+0 : antenna rotation. The antenna is rotating permanently over 360° (repeat mode 1)
shift+1 : robotic arm activation. Demonstrat composition of movements (translation of the base and rotations of the arms). The movement is permanently cycling from one end to the other (repeat mode 2)
shift+2 : solar panel deployment. Demonstrate composition of movements (rotations only)
shift+3 : inflatable structure. Demonstrate the use of the scale function and composition of movements (translation and scaling).

Distribution :

The distribution is taking advantage of the orbiter support for sub-directories. no file will be placed in the root directories. This should result in better classification and in no existing file being replaced.

Base package (always required)
module\spacecraft.dll : module implementation of the spacecraft vessel class
config\spacecraft\spacecraft.cfg : spacecraft vessel class configuration file
add-on docs\spacecraft.pdf : this documentation file

Example 1
scenarios\spacecraft – space shuttle.scn
config\spacecraft\sts-101.ini
config\spacecraft\maqsat-r.cfg
config\spacecraft\maqsat-g.cfg
config\spacecraft\maqsat-b.cfg
meshes\spacecraft\maqsat-r.msh
meshes\spacecraft\maqsat-g.msh
meshes\spacecraft\maqsat-b.msh

Example 2
scenarios\spacecraft – deltaglider.scn
config\spacecraft\deltaglider.ini

Example 3
scenarios\spacecraft – deltaglider landed.scn
config\spacecraft\gl-01.ini

Example 4
No more include in this package, download it from htttp://gdhp.tripod.com/Orbiter

Example 5
scenarios\spacecraft – horizontal takeoff vertical landing.scn
config\spacecraft\gl-HV.ini

Example 6
scenarios\spacecraft – vertical takeoff horizontal landing.scn
config\spacecraft\gl-VH.ini

Example 7
scenarios\spacecraft – Multiple docking.scn
config\spacecraft\deltaglider01.ini
config\spacecraft\deltaglider02.ini
config\spacecraft\module1A.ini

Example 8
scenarios\spacecraft – Animations.scn
config\spacecraft\platform.ini
meshes\spacecraft\platform.msh

For developers
The source code can be obtained upon request.
OrbiterSDK\samples\spacecraft.c, spacecraft.h : the source code of the spacecraft.dll module

Credits/Copyright

Space shuttle, DeltaGlider : from the original Orbiter package
SCA747 add-on by Damir "slat" Gulesich
Source code : all source code by Vinka inspired from the various sample code by Martin Schweiger.

Thanks to Christoph Kopp for beta testing this version.

Known bug

There is not much protection against wrong ini file therefore this will result in Orbiter crash. My advice is to start from a working ini file and modify it gradually.

Support

Any questions, help, request or bug report to vinka@swing.be

What will come next

- User requests (I know some were not satisfied in this release).
- Own requests coming from new add-ons using spacecraft.dll