Spacecraft support
Version : 4 (152315)
Author : Vinka
Orbiter Version compatibility : 2010 (100830)

Introduction
The spacecraft4.dll module gives support to spacecraft that can be configured through initialisation file.
Most of the conventional spacecraft's will be supported by this module. You will be able to create your own spacecraft without the need to write a specific DLL.

New features of this version
- OrbiterSound 4.0 support
- UMMU 3.0 support
- UCGO 3.0 support
- LUA support

New features or modified features are indicated with new in this documentation.
Obsolete features are indicated with obsolete

This new version should be compatible with scenarios using Spacecraft3 module. If you find an add-on where it is not the case please send me an e-mail, I'll take a look. I recommend to replace reference to “Spacecraft3” by “Spacecraft” into your scenario and to specify the “Spacecraft4” module in a Spacecraft.cfg file. Proceeding this way, you'll only have change the Spacecraft.cfg file to specify last Spacecraft<i> module version without re-editing all your scenarios

Limitation

- Obsolete – this limitation could be removed with latest version of Orbiter maximum of 10 spacecraft's 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 (each payload may be composed of up to 16 different meshes)
- maximum of 16 docking ports
- maximum of 32 animation sequence
- no limitation on components in an animation sequence (dynamically allocated)
- maximum of 1024 groups for all the animation sequences
- maximum of 16 attachment

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. In pause mode, left-shift 0,1 ...9 will pause then restart the motion. Using left-shift and left-ctrl 0,1,2,...,9 will restart the motion in the opposite direction.
Key 'K' is usually used for payload bay doors and G' for Gear.
newyou may use indifferently the keys 0-9 on the Numpad or on the keyboard.

modified because 'a' is now a standard command in Orbiter ctrl+'a' : Activate / Deactivate parent attachment management. A message will be displayed in the screen bottom left when activated. The name of the selected attachment and the name of the attached vessel (if any) is displayed. When active, cycle through attachment by using left shift Key + NumPad 4 (previous) or NumPad 6 (next). Use left shift Key + NumPad 0 to attach child (if any) / detach child (depending on the current status of the parent attachment)
Use left shift Key + NumPad 5 to toggle display of grapple points
newyou may also use the keys on the keyboard instead of the Numpad

'space bar' : Activate / Deactivate robotic arm. A message will be displayed in the screen bottom left when activated. The name of the selected robotic arm part is displayed. You can cycle through the different parts by using left shift Key / NumPad 4 (previous) or NumPad 6 (next). Use left shift Key + NumPad 8 or left shift Key + NumPad 2 to move the part in one direction or in the other direction. Releasing the key will stop motion. Use left shift Key + NumPad 0 to grapple or release the target.
newyou may also use the keys on the keyboard instead of the Numpad

newUMmu commands :
Left shift + 'N' : add New crew member to selected seat. If the seat is occupied by another crew member, the first available seat will be used.
Left shift + 'A' : toggle open/close Airlock (if airlock door animation is defined, it will be controlled by this key also)
Left shift + 'E' : EVA the crew member on the selected seat (if not empty)
Left shift + 'C' : Cycle through seats

obsolete – it is now managed directly by OrbiterLeftShift +NumPad * : activate/deactivate RCS mode obsolete – it is now managed directly by OrbiterLeftShift + NumPad . : activate/deactivate Control Surface mode

newUCGO commands :
Left shift + 'D' : cycle through cargos on Disk
Left shift + 'L' : Load selected cargo on disk to selected slot. If the selected slot is not empty, the first empty slot found will be used
Left shift + 'G' : Grapple cargo to selected slot. If the selected slot is not empty, the first empty slot found will be used
Left shift + 'R' : Release cargo from selected slot
Left shift + 'S' : cycle through Slots

newLUA commands :
To get control of a Spacecraft vessel, use the following LUA command
v=vessel.get_interface('<the name of the spacecraft>')

You now have the following specific Spacecraft LUA commands available :

- v:jettison() : jettison payload (=act like 'J' key)
- v:attach('<attachment name>') : if the named attachment is empty and a vessel is close enough, attach it. If the named attachment is full, detach the vessel attached to it (=act like the Lefst shift + '0' key in attachment mode)
- v:arm('<joint name>',angle) : move named joint to specified angle
- v:anim('<animation sequence name>',position) : move animation sequence to specified position (range [0,1])
- v:ucgo('load','<cargo configuration file name>',slot) : load the cargo at the specified slot (do not specify file name extension)
- v:ucgo('release',slot) : release cargo at the specified slot
- v:ucgo('grapple',slot) : grapple closest cargo at the specified slot
- v:ummu('add','<crew name>',age) : add crew member to the current selected seat
- v:ummu('eva','<crew name>') : initiate EVA for the named crew member
- v:ummu('airlock',<0|1>) : open (1) or close (0) airlock

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

Files needed :

Modules\spacecraft4.dll : the module implementing the generic spacecraft class
Config\spacecraft\spacecraft.cfg : the class configuration file(or name it spacecraft4.cfg if you want a specific class)

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
…

(or vessel name:spacecraft\spacecraft4 if you prefer to use the specific spacecraft4 class)

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)
obsolete RCS – Orbiter is using RCSMODE : you must replace RCS 0 by RCSMODE 0 and RCS 1 by RCSMODE 1 (rotation) or RCSMODE 2 (translation)
~~RCS 0 (0=RCS deactivated, 1=RCS activated)~~
obsolete CTRL_SURFACE – Orbiter is using AFCMODE : you must replace CTRL_SURFACE 0 by AFCMODE 0 and CTRL_SURFACE 1 by AFCMODE 7 (all surfaces active)
~~CTRL_SURFACE   1 (0=control surfaces locked at zero angle, 1=control surfaces active)~~
newSEAT 2:John Doe (the seat 2 is occupied by member crew John Doe). Note that this must be coherent with UMmu saved parameters

Initialisation File format

This file follow the format of the standard windows ini files. The definition of each parameter is very short as this is very often a literal translation of the same parameter in Orbiter SDK. You should read Orbiter SDK documentation to have more information about all these parameters. The following sections and items can/must be defined :

[TEXTURE_LIST]
You must declare the texture files that will be used for engine exhaust rendering, this list is preloaded in the dll.

- TEX_x : the texture name that will be used in one of the ENG_TEX field. x stands for 1, 2, … up to 16.

Note that this texture library is static and common to all spacecraft vessels. This means that a maximum of 16 textures can be used for all spacecraft vessels but different vessels may use the same texture by using the same name.

[PARTICLESTREAM_x]
Definition of a particle stream. “x” stands for 1,2, … up to 13. Particle stream parameters are those defined in the orbiter SDK (see it for details). You must define a particle stream section before using it in the ini file.
Three particule streams are predefined and can be used. They are named “contrail”,”exhaust” and “rcs” (they are based on the sample definition of Orbiter sample code).

- NAME : the name associated with the particle stream, this name will be used to associate the particle stream with engines. Three default particle stream are already defined in the dll : exhaust, contrail and rcs.
- SRCSIZE : see orbiter API doc for details
- SRCRATE : see orbiter API doc for details
- V0 : see orbiter API doc for details
- SRCSPREAD : see orbiter API doc for details
- LIFETIME : see orbiter API doc for details

- GROWTHRATE : see orbiter API doc for details
- ATMSLOWDOWN : see orbiter API doc for details
- LTYPE : see orbiter API doc for details. Possible values are EMISSIVE, DIFFUSE
- LEVELMAP : see orbiter API doc for details. Possible values are LVL_LIN, LVL_FLAT, LVL_SQRT, LVL_PLIN, LVL_PSQRT
- LMIN : see orbiter API doc for details
- LMAX : see orbiter API doc for details
- ATMSMAP : see orbiter API doc for details. Possible values are ATM_FLAT, ATM_PLIN, ATM_PLOG
- AMIN : see orbiter API doc for details
- AMAX : see orbiter API doc for details
- TEX : texture file name for particle stream rendering

Note that this particle stream library is static and common to all spacecraft vessels. This means that a maximum of 13 particle streams can be used (+3 predefined) for all spacecraft vessels but different vessels may use the same particle stream by using the same name.

[AERODYNAMICS]

- MODEL : possible values are “shuttle”, “dglider” or “capsule”. If you know nothing at aerodynamics, this is the only parameters that you must specify. All other parameters related to aerodynamics will take default values according to the model chosen. For “shuttle”, Orbiter default values given for space shuttle Atlantis will be used. For “dglider”, Orbiter default values given for delta glider will be used. For “capsule”, another set of default values are used. If you want to replace default values by a specific values, you must specify one or more of the following parameters :
- VAIRFOIL_ATTACK : attack point (x,y,z) in vessel coordinates of the vertical air foil
- VAIRFOIL_CHORD : chord length of the vertical air foil
- VAIRFOIL_AREA : vertical air foil area
- VAIRFOIL_ASPECT : vertical air foil
- VAIRFOIL_LIFT_1 (up to _32) : (aoa,cl,cm) lift profile for vertical air foil, the first element (aoa) is the angle of attack and must be defined between -180° and +180°, the second element (cl) is the lift coefficient, the third element is the moment coefficient
- VAIRFOIL_EFFICIENCY : vertical air foil efficiency factor (b^2/S where b is the wing span and S the wing area)
- VAIRFOIL_WAVE_DRAG : maximum drag coefficient in transonic
- VAIRFOIL_CD0 : drag coefficient at 0 angle of attack
- HAIRFOIL_ATTACK : attack point (x,y,z) in vessel coordinates of the horizontal air foil
- HAIRFOIL_CHORD : chord length of the horizontal air foil
- HAIRFOIL_AREA : horizontal air foil area
- HAIRFOIL_ASPECT : horizontal air foil
- HAIRFOIL_LIFT_1 (up to _32) : (beta,cl) lift profile for horizontal air foil, the first element (beta) is the bank angle and must be defined between -180° and +180°, the second element (cl) is the lift coefficient
- HAIRFOIL_EFFICIENCY : horizontal air foil efficiency factor (b^2/S where b is the wing span and S the wing area)
- HAIRFOIL_WAVE_DRAG : maximum drag coefficient in transonic
- HAIRFOIL_CD0 : drag coefficient at 0° bank angle
- ELEVATOR_ATTACK : attack point of the elevator
- ELEVATOR_AREA : elevator control surface area
- ELEVATOR_LIFT : elevator maximum change in lift when fully oriented
- ELEVATOR_ANIM : sequence number for animation of elevator

- RUDDER_ATTACK : attack point of the rudder
- RUDDER_AREA : rudder control surface area
- RUDDER_LIFT : rudder maximum change in lift when fully oriented
- RUDDER_ANIM : sequence number for animation of rudder
- AILERON_ATTACK : attack point of the aileron
- AILERON_AREA : aileron control surface area
- AILERON_LIFT : aileron maximum change in lift when fully oriented
- AILERON_RIGHT_ANIM : sequence number for animation of the right aileron
- AILERON_LEFT_ANIM : sequence number for animation of the left aileron
- ELEVATOR_TRIM_ATTACK : attack point of the elevator trim
- ELEVATOR_TRIM_AREA : elevator trim control surface area
- ELEVATOR_TRIM_LIFT : elevator trim maximum change in lift when fully oriented
- ELEVATOR_TRIM_ANIM : sequence number for animation of elevator trim
- RUDDER_TRIM_ATTACK : attack point of the rudder trim
- RUDDER_TRIM_AREA : rudder trim control surface area
- RUDDER_TRIM_LIFT : rudder trim maximum change in lift when fully oriented
- RUDDER_TRIM_ANIM : sequence number for animation of rudder trim
- FLAPS_ATTACK : attack point of the flaps
- FLAPS_AREA : flaps control surface area
- FLAPS_LIFT : flaps maximum change in lift when fully oriented
- FLAPS_ANIM : sequence number for animation of flaps
- SPEEDBRAKE_ATTACK : speed brake attack point
- SPEEDBRAKE_DRAG : speed brake maximum change in drag when fully deployed
- SPEEDBRAKE_ANIM : sequence number for animation of speed brake

General information about Aerodynamic (crude approximation)
ATTACK is the point where the aerodynamic force is applied, usually this point should be aft of the center of gravity. Set something like (0,0,-x)
CHORD is the length of the wing (in z direction) or the vessel length (V) or height (H) (for lifting body)
*AREA is the surface of the wing (seen from above) or vessel area (for lifting body = width * length (V) = height * length (H)*
ASPECT is the wing aspect ~= wing span^2 / wing area (for lifting body = width/length)
EFFICIENCY, use same value as aspect if you don't know or better Clift/Cdrag if given
ELEVATOR is responsible for lift control
RUDDER is responsible for yaw control
AILERON is responsible for roll control
FLAPS may increase the lift at low speed (reducing the stall speed)
SPEEDBRAKE may increase the drag, therefore reducing the speed faster

[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). The sphere defined must enclosed the vessel completely.
- 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)

- newMAIN_THRUST_ANIM=-1 : if you want to associate an animation with the value of main thrust (animation value [0,1] is directly associated to main thrust level [0,1]
- newMAIN_THRUST_ANIM_SPEED=-1 : if you want to associate an animation with the value of main thrust : animation value is calculated with a rate of change based on the thrust level. At thrust level 0, the rate is null and the animation value does not changed, at thrust level 1, the animation sequence will be played from 0 to 1 in the specified duration for the animation (maximum frequency) and repeated again. This mode is useful to animate a propeller for example.
- MAIN_TEX=texture name in the library to render main thrust
- MAIN_PSTREAM1=first particle stream name in the library to render main thrust
- MAIN_PSTREAM2=second particle stream name in the library to render main thrust
- RETRO_THRUST=0 : the vessel retro engine thrust (N)
- newRETRO_THRUST_ANIM=-1 : see MAIN_THRUST_ANIM, same for retro. Note that if both MAIN_THRUST_ANIM and RETRO_THRUST_ANIM have the same animation sequence number, the animation mapping will be different : main thrust level [0,1] will be map to animation [0.5,1] and retro thrust level [0,1] will be map to animation [0.5,0].
- newRETRO_THRUST_ANIM_SPEED=-1 : see MAIN_THRUST_ANIM_SPEED, same for retro. Note that if both MAIN_THRUST_ANIM_SPEED and RETRO_THRUST_ANIM_SPEED have the same animation sequence number, the main thrust will be counted as positive speed and the retro thrust will be counted as negative speed (propeller will rotate in both direction according to speed).
- RETRO_TEX=texture name in the library to render retro thrust
- RETRO_PSTREAM1=first particle stream name in the library to render retro thrust
- RETRO_PSTREAM2=second particle stream name in the library to render retro thrust
- HOVER_THRUST=0 : the vessel hover engine thrust (N)
- newHOVER_THRUST_ANIM=-1 : see MAIN_THRUST_ANIM, same for hover.
- newHOVER_THRUST_ANIM_SPEED=-1 : see MAIN_THRUST_ANIM_SPEED, same for hover.
- HOVER_TEX=texture name in the library to render main thrust
- HOVER_PSTREAM1=first particle stream name in the library to render hover thrust
- HOVER_PSTREAM2=second particle stream name in the library to render hover thrust
- ATTITUDE_THRUST=7740 : the vessel attitude engine thrust (N), thrust is considered to be applied by thrusters located at 1 meter from the rotation axis, you should adapt the engine thrust according to the real radius.
- ATT_TEX=texture name in the library to render attitude thrust
- ATT_PSTREAM1=first particle stream name in the library to render attitude thrust
- ATT_PSTREAM2=second particle stream name in the library to render attitude thrust
- REENTRY_PSTREAM=particle stream name in the library to render atmospheric re-entry
- 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 pitch moment scale
- BANK_MOMENT_SCALE=0.00002 : the bank moment scale : obsolete – but kept for backward compatibilty – replaced by YAW_MOMENT_SCALE
- newYAW_MOMENT_SCALE=0.00002 : the yaw moment scale (replace bank_moment_scale)

- ROT_DRAG=(0.5,1.0,1.0) : the rotational drag
- TRIM : the trim value
- 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 mother craft 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.

[LANDING_GEAR]
- bug correctionWHEEL_FRICTION=0.04 : the surface friction coefficient associated with the longitudinal axis when wheels are fully deployed.
- bug correctionBELLY_FRICTION=0.4 : the surface friction coefficient associated with the lateral axis and with longitudinal axis when wheels are not fully deployed (belly landing)
- bug correctionBRAKE_FORCE=200000. : the maximum wheels brake force in Newton when wheels are fully deployed. When wheels are not deployed, brake can't be used and value is set to 0.
- SEQ=-1 : the animation sequence associated with the landing gear. When an animation sequence is defined for landing gear surface friction and brake force are modified when wheels are fully deployed. The animation sequence value for fully deployed mode must be 1 (not 0). If there is no animation sequence associated, the parameters are set according to normal landing mode (landing gear deployed).

[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=CCW, 1=CW (clock wise)
- LIN_AXIS=Y : the translational axis (X, Y or Z)
- LIN_CW=1 : the translational direction (0=negative axis or 1= positive axis)

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_1]
...
animation sequence definition sections, this corresponds to the definition of the ANIM_SEQ given by Martin in the sdk documentation

- newNAME : sequence name (use max 32 character). This parameter is only useful to identify the sequence if you intend to use LUA script
- INIT_POS : the animation value between [0,1] which correspond to the position of the mesh group as it is specified in the mesh file (default is 0). If you want to start the simulation with a different position, this position must be specified in the scenario file (.scn) and not in the ini file.
- 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 the specified initial 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

- PAUSE : (default value is 0). Setting this value to 1 will indicate that in normal mode, pressing a key will pause motion. Restarting of motion can be done in the direction 0->1 by using left-shift + key and in the opposite direction (1->0) by using left-shift & left-ctrl + key.

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

[ANIM_COMP_0]
[ANIM_COMP_1]
...
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=ROTATE : 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
- ANGLE=160. : the angle of rotation (negative value for rotation in other direction)
- PARENT= : the ANIM_COMP number of the parent if any. The parent must always be declared before the child.

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
- SHIFT=(0,-0.8,0) : the translation value that is applied.
- PARENT= : the ANIM_COMP number of the parent if any. The parent must always be declared before the child.

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
- SCALE=(0.1,0.1,1) : the scale values that are applied.
- REF=(-0.340,-3.1855,0) : the reference point to apply the scaling (fixed point where the mesh seems to be scaled around).
- PARENT= : the ANIM_COMP number of the parent if any. The parent must always be declared before the child.

FOR ROBOTIC ARM (specific)
For the grapple of the robotic arm, you must define an animation component without reference to mesh groups. The transformation will be applied to a triangle. You must defined the 3 points coordinates by using :
- TIP_1=(-2.26,1.71,-6.5) : first point coordinates
- TIP_2=(-2.26,1.71,-7.5) : second point coordinates

- TIP_3=(-2.26,2.71,-6.5) : third point coordinates

These coordinates will be used to move dynamically the attachment position associated with the grapple in accordance with the robotic arm motion. Except for the GROUPS parameter that is replaced by TIP_1, TIP_2 and TIP_3, all other parameters of the animation component must be specified.

Composition of movements :
the animation components are now managed with reference to parent components (fully compliant with the Orbiter definition). This reference to parent is specially useful if you want to use the robotic arm feature.

[PAYLOAD_0]
...
[PAYLOAD_9]
payloads definition sections

- MESHNAME=Carina : mesh name of payload. You can define up to 5 meshes for one payload, they must be declared in the same string, with each mesh name separated by a ; (semi-column). Ex: mesh1;mesh2
- OFF=(0,10,-5) : (x,y,z) vector position of payload mesh offset in launcher (meters). If you have declared more than one mesh for the payload, you must declare the same number of offset in the same order. Offset positions are separated by ; (semi-columns). Ex: (0,0,25); (0,0,27)
- MASS=1200 : payload mass in kg
- MODULE=carina : module name to be called at vessel create. This is the config file name of the vessel type of the payload.
- NAME : the name that will be given to the vessel at payload release time
- SPEED : translation speed for payload jettison relative to spacecraft (in m/s)
- ROT_SPEED : rotation speed for payload jettison relative to spacecraft (in rad/s)

[SOUND]
The sound section gives support to OrbiterSound version 4.0

- MAIN_THRUST : specify the sound file to be played for main thrust
- HOVER_THRUST : specify the sound file to be played for hover thrust
- RCS_THRUST_ATTACK : specify the sound file to be played for RCS thrust
- RCS_THRUST_SUSTAIN : specify the sound file to be played for RCS thrust
- AIR_CONDITIONING : specify the sound file to be played for air conditioning
- COCKPIT_AMBIENCE_1 : specify the sound file to be played for cockpit ambience
- COCKPIT_AMBIENCE_2 : specify the sound file to be played for cockpit ambience
- COCKPIT_AMBIENCE_3 : specify the sound file to be played for cockpit ambience
- COCKPIT_AMBIENCE_4 : specify the sound file to be played for cockpit ambience
- COCKPIT_AMBIENCE_5 : specify the sound file to be played for cockpit ambience
- COCKPIT_AMBIENCE_6 : specify the sound file to be played for cockpit ambience
- COCKPIT_AMBIENCE_7 : specify the sound file to be played for cockpit ambience
- COCKPIT_AMBIENCE_8 : specify the sound file to be played for cockpit ambience
- COCKPIT_AMBIENCE_9 : specify the sound file to be played for cockpit ambience
- newMODE_ROTATION :
- newMODE_TRANSLATION :
- newMODE_ATTITUDEOFF :
- newWIND_AIRSPEED :

- newREENTRY_AIRSPEED :
- newLAND_TOUCHDOWN :
- newGROUND_ROLL :
- newWHEELBRAKE :
- newCRASH_SOUND :
- newDOCKING :
- newUNDOCKING :
- newRADIOLANDCLEARANCE :
- newDOCKING_RADIO :
- newUNDOCKING_RADIO :
- newRADAR_APPROACH :
- newRADAR_CLOSE :
- newRETRO_THRUST :
- newUSER_THRUST :
- newCOUNTDOWN_WHENTAKEOFF :

[VC]
This is for virtual cockpit mode
MESHNAME="atlantis": specify the mesh of the cockpit, if the cockpit mesh is inside the vessel mesh, just put the same name as the one you have specified in the [config] section
MFD_LEFT=125 : mesh group where Orbiter standard left MFD will be drawn. See Orbiter SDK documentation for restriction on this mesh group (must be a rectangle)
MFD_RIGHT=126 mesh group where Orbiter standard right MFD will be drawn.
HUD=127 : mesh group where Orbiter HUD will be displayed
HUD_SIZE=0.252 : screen size (in meter) for HUD display. See Orbiter SDK for details.
HUD_CENTER=(-0.5479,2.325,15.777) : coordinates of HUD center on the mesh. See Orbiter SDK for details.

[PARENT_ATTACH_0]
[CHILD_ATTACH_0]
...
use [PARENT_ATTACH_i] to specify a parent attachment, use [CHILD_ATTACH_i] to specify a child attachment. The total number of attachment (child and parent added) per spacecraft is 16.
attachment management
- NAME=string : the name of the attachment as it will be displayed. For example 'cargo bay front'. If omitted the name will be the attachment number
- POS=(x,y,z) : the position of the attachment in vessel coordinates
- DIR=(x,y,z) : the direction of the attachment (normalized vector)
- ROT=(x,y,z) : the orientation of the attachment (normalized vector, must be perpendicular to DIR vector)
- LOOSE=<0|1> : specify if loose attachment allowed (1)
- ID="XS : 8 characters string to specify an identification of the child attachment (this field is only used with child attachment)
- RANGE=<value> : the maximum distance in meter between the parent attachment and the child attachment to consider that attachment between both vessels is allowed (this field is only used with parent attachment)
- newSPEED=(0,1,0) : the release speed of the child when the child is detached (this field is only used with parent attachment). SPEED is in m/s over the 3 axes. If not defined, the speed will be zero.

- newROT_SPEED=(0,1,0) : the release rotation speed of the child when the child is detached (this field is only used with parent attachment). ROT_SPEED is in rad/s over the 3 axes. If not defined, the rotation speed will be zero.

Note on displaying grapple points : this is an undocumented feature of Orbiter but it seems that only child attachment with there ID name starting with a 'G' are displayed.
Note on using SPEED, ROT_SPEED : these values are expressed in vessel axes and do not take into account the tip orientation for the robotic arm. So, do not use it with the tip attachment on the robotic arm or the release will be weird. It is unlikely that a robotic arm would be able to provide translation and rotation speed at release

[ROBOTIC_ARM]
robotic arm definition. You can define up to 8 joints.

- JOINT_0_NAME="shoulder yaw" : name of first joint that will be displayed on screen when this joint is selected
- JOINT_0_SEQ=5 : reference to associated animation sequence. This animation sequence must be defined with parameters INIT_POS and DURATION. A KEY parameter is not required as the joint will be manipulated with the NumPad (see module command keys paragraph)
- JOINT_0_RANGE=(-120,200) : angular position of joint that will be displayed on screen for information
- JOINT_1_NAME,JOINT_1_SEQ...JOINT_7_NAME,JOINT_7_SEQ : other joints
- GRAP_SEQ=10 : reference to associated animation sequence with grapple. This animation sequence is associated with a special animation component (see ANIM_COMP definition)
- GRAP_ATTACH=3 : reference to associated attachment with grapple. This attachment must be defined like the other attachments and must be a parent attachment.

new[UMMU]
Ummu definition

CREW_SIZE=4: specify the maximum number of crew members on board of the ship.
AIRLOCK_POS=(0,0,10.5) : position of the center of the airlock in ship coordinates.
AIRLOCK_SIZE=(2,2,4) : size of the airlock along ship axes.
AIRLOCK_ANIM=11 : number of the animation sequence associated with airlock door. Do not specify or set value to -1 if you don't have airlock animation
EVA_POS=(0,2.5,10.5) : position of crew member for EVA in ship coordinates
EVA_ATT=(0,0,0) : attitude of crew member for EVA (angles in degrees)
SPACE_SUIT_MESH="SomeSuit" : mesh name to use instead of the default space suit from UMMU
SEAT_<i>_GRPS=27,111 : groups number in the mesh that represent the UMmu seated in the vessel (if any)

new[UCGO]
UCGO definition

RELEASE_SPEED=0.1 : release speed of cargo in space in m/s
MAXIMUM_LOAD=5000 : maximum cargo mass in kg that the vessel can carrying
GRAPPLE_DISTANCE=50 : grapple distance radius in meter from center of ship
GROUND_RELEASE_POSITION=(0,0,4) : global release position on ground
SLOT_<i>_POS=(0,0,0) : position of cargo slot <i> (where i=0,1, .. 39) in vessel coordinates
SLOT_<i>_ATT=(0,0,0) : attitude of cargo slot <i> (where i=0,1, .. 39) (angles in degrees)

Examples

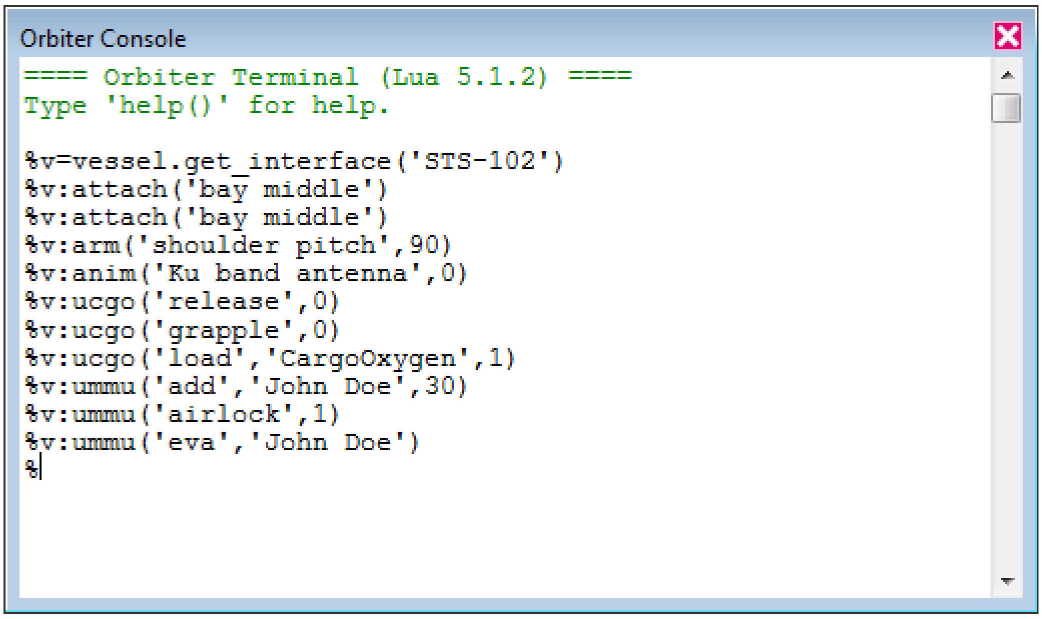
The following examples can be downloaded with the spacecraft add-on, their purpose is to illustrate the possibilities of spacecraft3.dll, they are probably more useful for developers than for end users.

Animations
a space platform demonstrating various animations possibilities.
Left shift+0 : antenna rotation. The antenna is rotating permanently over 360° (repeat mode 1)
Left shift+1 : robotic arm activation. Demonstrate 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)
Left shift+Left ctrl+2 : solar panel deployment. Demonstrate composition of movements (rotations only), Left shift+2 to fold, Left shift+Left ctrl+2 to unfold panels
Left shift+3 : inflatable structure. Demonstrate the use of the scale function and composition of movements (translation and scaling).
Left shift+4 : rotation of the arms

Atlantis
The space shuttle Atlantis in orbital configuration. The payload bay contains three small payloads that can be jettison (J key). Various ejection speed (translation & rotation) are demonstrated. The payload bay opening is commanded with 'K' key and the landing gear by 'G' key. At second payload jettison, the focus change to the payload (as specified in ini file), use F3 and select sts-101 to return the focus on to the shuttle.

Atlantis – Hst
The space shuttle Atlantis in orbit with the Hubble Space Telescope in the payload bay. Use the shuttle arm (see example below) to grapple Hubble, release Hubble from the shuttle and deploy it in space. Hubble is also animated : Left shift+1 : deploy high gain antenna, Left shift +2 : open hatch, Left shift+3 : deploy solar panels

Atlantis – RMS
The space shuttle Atlantis in orbital configuration. The payload bay is opened with two small payloads attached (use Left Ctrl + 'a' then Left shift NumPad4/NumPad6 to cycle through attachment and left shift NumPad0 to attach/detach payload). A third payload is already in space and the robotic arm is deployed. (use 'space bar' then left shift NumPad4/NumPad6 to cycle through joints, left shift NumPad2/Numpad8 to move joint, left shift NumPad0 to grapple / ungrapple when attachment 'rms' is selected).
new-UCGO There is also a UCGO cargo in the payload bay. 4 slots are defined. Use F1 key to switch to cockpit view. Use Left shift+'S' to cycle through the cargo slots. Use Left shift+'D' to cycle through cargo on disk. Use Left shift+'L' to load selected cargo on disk to selected slot. Use Left shift+'R' to release cargo from selected slot, use Left shift+'G' to grapple cargo to selected slot.
new-LUA script capability Check the LuaConsole in Orbiter Modules and start the scenario. Then open the console by selecting it with CTRL+F4



```
==== Orbiter Terminal (Lua 5.1.2) ====
Type 'help()' for help.

%v=vessel.get_interface('STS-102')
%v:attach('bay middle')
%v:attach('bay middle')
%v:arm('shoulder pitch',90)
%v:anim('Ku band antenna',0)
%v:ucgo('release',0)
%v:ucgo('grapple',0)
%v:ucgo('load','CargoOxygen',1)
%v:ummu('add','John Doe',30)
%v:ummu('airlock',1)
%v:ummu('eva','John Doe')
%
```

Atlantis Final Approach
The Atlantis Space Shuttle in final approach at KSC. The new aerodynamic model is implemented, use Left shift+ Numpad 1 to deploy air brake. Use F8 to change view to virtual cockpit 3D (Do not zoom using the mouse or Orbiter will CTD !)

Deltaglider final approach
The Deltaglider in final approach at KSC. The new aerodynamic model is implemented with control surface animations. Use 'G' key to deploy landing gear, Use Left shift+Numpad 4 to deploy air brake. Use F8 to change view to virtual cockpit 3D

Deltaglider landed
The Delta glider landed at the Cape.'K' key commands the nose cone, 'G' key commands the landing gear, Left shift + Numpad 0 key commands the air lock, Left shift + Numpad 1 commands the radiator, Left shift + Numpad 2 commands the escape ladder, Left shift + Numpad 3 commands the top hatch, Left shift + Numpad 4 commands the air brake.

Deltaglider
The Delta glider in orbital configuration. See previous example for the command keys.
RCS thrusters rendering with particle stream has been added
new-UMmu Use 'F1' to enter cockpit mode. UMmu commands are displayed on screen. Use Left shift+'N' to enter a new crew member. Use Left shift+'C' to cycle through seats. Use Left shift+'A' to open/close airlock (you must also open nose cone with 'K'). Use Left shift+'E' to EVA selected crew member (check that selected seat is not empty !).

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

Russian Dolls
This example illustrates the possibility to define payloads composed of multiple meshes. Each ship is carrying the smaller ones. Jettison the small ships ('J') then change the focus ('F3') and jettison again until all vessels are separated. You should end this scenario with 6 ships, all of class 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.

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\spacecraft4.dll : module implementation of the spacecraft vessel class
config\spacecraft\spacecraft(4).cfg : spacecraft vessel class configuration file
add-on docs\spacecraft.pdf : this documentation file

All add-ons based on spacecraft4.dll may be distributed freely with this base package included but you should preferably asks to download last spacecraft4 version.

For developers
The source code can be obtained upon request.

Credits/Copyright

Space shuttle, DeltaGlider, Hst : from the original Orbiter package
Source code : all source code by Vinka inspired from the various sample code by Martin Schweiger.

OrbiterSound 4.0, UMmu 3.0 and UCGO 3.0 are from Daniel Polli (aka Dansteph) and can be downloaded at http://orbiter.dansteph.com

Thanks to all beta testers of this version. Special thank to jacquesmomo who spent more hours using my add-ons than I used for developing it.

Known bugs

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.

When going to 3D cockpit mode if first action is zooming with the mouse Orbiter will crash.

Support

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