Multistage launcher support
Version : 050630
Author : Vinka
Orbiter Version compatibility : 050216

Orbiter
Atlantis-Launch
000013.6 004 p=65.00 [81.75] b=-179.98 [-0.02] h=141.80 [-98.80]

Introduction
The multistage.dll module gives support to multistage launcher that can be configured through text file (windows ini file format). Most of the conventional launcher will be supported by this module. You will be able to create your own multistage launcher without the need to write a specific DLL.

Last version new features
- load/save scenario should work (but you better quit Orbiter between save/load sessions or some CTD (crash to desktop) may occur)
- multiple instances of multistage vessels are possible in the same scenario
- support for DanSteph OrbiterSound version 3.0
- Orbiter Vessel2 type compatible
- all obsolete functions removed from the code
- some new features in autopilot

Description and Limitation
Boosters
Boosters can be attached only to the first stage, You can define up to 10 groups of boosters. A group can contain up to 10 boosters (each booster in the group as the same characteristics), use only different groups if you want to define different characteristics such as the thrust or the burning time. Boosters are automatically fired when the first stage engine level reach 95% of the nominal thrust or after a programmable delay. Boosters are automatically jettisoned when fuel mass is falling to 0.
Stages
Only one stage can be fired at a time, when the fuel mass of the reach 0, the

stage is jettison and the next stage (if any) is ignited. If the first stage is running out of fuel and boosters are still thrusting, the first stage separation will be delayed until all boosters are jettisoned. You can define up to 10 stages.

Interstages

An interstage can be defined between stages. The interstage is dropped after previous stage jettison and before next stage ignition.Up to 9 interstages can be defined.

Payloads

A maximum of 10 payloads can be defined. You can program the focus so that, at the defined payload jettison, the vessel focus goes to this payload giving you the control of the payload. This change of focus is not very eye friendly because the image jump to the new focus vessel with a different position and size. If a fairing is defined, the fairing must be jettison prior to payload jettison. During the payload separation phase, the last stage remains active. Therefore, you can change the attitude, rotation speed before jettisoning the payload. If you have kept enough fuel, you can even restart the stage main engine to change the orbit. The first payload that will be jettison must be placed on top of the payload stack.

Fairing

A fairing can protect the payloads. This fairing can be composed of 10 elements maximum.

Module Command keys

Three predefined keys are detected by the multistage vessel class :

- ‘f’ : jettison fairing
- ‘j’ : jettison stage, interstage, payload in the define sequence order
- 'p' : start the guidance program if any

Scenario File format

The following lines must be added to the scenario file(.scn in orbiter\scenarios) :

CONFIG_FILE <subdir\filename> the ini file to be read by the multistage module for proper definition of the multistage launcher.

- GUIDANCE_FILE <subdir\filename> the guidance file to be read by the multistage gnc module for launcher guidance. This file is not mandatory
- CONFIGURATION 0 (=launch), 1 (=in flight)
- CURRENT_BOOSTER x (=current booster group running)
- CURRENT_STAGE x (=current stage running)
- CURRENT_INTERSTAGE x (=current interstage running)
- CURRENT_PAYLOAD (=current payload not jettison yet)
- FAIRING 0 (=no fairing), 1 (=fairing present)

The following lines are generated automatically by the multistage.dll when saving scenario and are used to restore the current state when loading scenario. You should generally no worry about these.

- MET : mission elapsed time when scenario was saved
- STAGE_STATE : current stage state (ignited, delayed ignition ...)
- STAGE_IGNITION_TIME : recorded MET to compute ignition delay
- GNC_RUN : guidance program run flag
- GNC_PROGRAM : guidance program current state
- GNC_STEP : guidance program current step
- GNC_MET : guidance program MET when saving scenario
- GNC_AUTO_JETTISON : flag auto jettison
- GNC_ENG_TIME : recorded MET at start of execution of engine(...) command
- GNC_ROLL_KILL : current status of the roll program
- GNC_ROLL_PITCH : recorded MET at start of pitch in the roll program

- GNC_ROLL_TIME : recorded MET at start of the roll program
- GNC_PITCH_TIME : recorded MET at start of pitch(...) command

Initialisation File format

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

[MISC]

miscellanous parameters section (this section is not mandatory)

- COG : center above ground (in meters), use to display the launcher at a certain altitude above ground. Usefull to place it on a launch table.
- GNC_DEBUG : if set to 1, will display the guidance parameters on screen (debug string)
- TELEMETRY : if set to 1, a file called “telemetry.txt” will be generated in Orbiter root directory. This is a text file with fields separated by <tab>. It can be imported directly in EXCEL for example. The recorded fields are : met (mission elapsed time), altitude, vertical speed, horizontal speed, longitude, latitude, mass, thrust.
- FOCUS : can be set between 1 and the number of payloads defined. If this parameter is defined when the specified payload is jettison. The focus will change from the multistage vessel to the newly created payload. This will generally change the point of view and the distance of view, you will observe a discontinuity of the rendering in Orbiter window.

[TEXTURE_LIST]

You must declare the texture files that will be used for engine exhaust rendering, this list is preloaded in the dll and shared by all instances of multistage vessels.

- TEX_x : the texture name that will be used in one of the ENG_TEX field. x stands for 1, 2, … up to 16. The texture name you specify must correspond to an existing file in the texture directory of orbiter

[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. Particle streams are preloaded in the dll and shared by all instances of multistage vessels.
Three particule streams are predefined and can be used without [Particulestream] section. They are named “contrail”, “exhaust”and “rcs” (they are based on the sample definition of the Atlantis and Deltaglider sample code).

- NAME : the name associated with the particle stream, this name will be used in booster and stage section to associated the particle stream to engines.
- 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

[BOOSTER_x]

booster group definition section where x stands for 1, 2, 3 ... up to 10. There can be no hole in the sequence. Booster section is not mandatory (as there exists some launcher without boosters).

Mandatory parameters

- N : number of booster in the group
- MESHNAME : mesh name root , booster meshes must be <meshname>_1.msh, <meshname>_2.msh, ..., <meshname>_N.msh. You must supply a mesh for each booster with the proper orientation for each booster.
- OFF : (x,y,z) vector position of booster mesh offset in launcher (meters)
- HEIGHT : booster height in meter
- DIAMETER : booster diameter in meter
- THRUST: booster thrust in N
- EMPTYMASS : booster empty mass in kg
- FUELMASS : booster fuel mass in kg
- BURNTIME : booster burn time in sec
- ANGLE : the initial angle that the OFF vector will be turned around Z axis for effective booster offset representation. All boosters in the group will be at the same distance from the Z axis and will be equally spaced around the Z axis.

Unmandatory parameters

- BURNDELAY : booster delayed time in sec before booster ignition (time start when first stage has reach 95% of thrust level). If this parameter is not defined, booster is ignited at launch.
- SPEED : translation speed of 1st booster jettison (in m/s), for other boosters, the speed vector is rotated to take into account the booster position angle in x-y plane.
- ROT_SPEED : rotation speed of 1st booster jettison (in rad/s), for other boosters, the rotation speed vector is also rotated.
- MODULE : name of the vessel class to booster will received when jettisoned. By default the module class will be stage which require the stage.dll to be installed (this come with multistage.dll).
- ENG_i: (x,y,z) vector position of exhaust mesh for thrust rendering. 'i' stands for 1, 2, ... up to 4, allowing up to 4 engines per booster. There can be no hole in the sequence. If present, the parameter ENG_DIAMETER must be defined. By default, only one engine will be rendered on the Z axis of the booster.
- ENG_DIAMETER : the diameter of the engine for exhaust thrust rendering
- ENG_TEX : texture file name for engine texture rendering (the file must exist in the \texture subdirectory)
- ENG_PSTREAM1 : name of particle stream attached to the engine. The name must be defined in a PARTICLESTREAM_x section or it must be one of the predefined streams (contrail,exhaust or rcs).
- ENG_PSTREAM2 : name of a second (optional) particle stream attached to the engine

- CURVE_i: (t,l) defines the booster thrust level versus time (t is for time in seconds , l is for thrust level in percent of maximum thrust). Thrust level is interpolated between specified times. The profile can contain a maximum of 10 points (i stands for 1, 2 .. up to 10). If no curve is defined then 100% thrust is applied all the time.

[STAGE_x]

Stage definition section where x stands for 1, 2, 3 ... up to 10. There can be no hole in the sequence. The STAGE_1 section is mandatory.

Mandatory parameters
- MESHNAME : the mesh name of the stage
- OFF : (x,y,z) vector position of stage mesh offset in launcher (meters)
- HEIGHT : stage height in meter
- DIAMETER : stage diameter in meter
- THRUST: stage thrust in N
- EMPTYMASS : stage empty mass in kg
- FUELMASS : stage fuel mass in kg
- BURNTIME : stage burn time in sec

Unmandatory parameters
- IGNITE_DELAY : the delay before ignition of stage after separation of previous stage/interstage element in seconds
- SPEED : translation speed for stage jettison (in m/s)
- ROT_SPEED : rotation speed for stage jettison (in m/s)
- MODULE : name of the vessel class the stage will received when jettisoned. By default, a “stage” class vessel is created.
- PITCHTHRUST: stage pitch axis attitude thrust in N. If not specified a default value is computed. Thrust is considered to be applied by two thrusters of the specified force located at one meter from the axis of rotation. If you want to simulate a thruster at a difference distance you have to multiply the distance by the real thrust and to use this value as the thrust value.
- YAWTHRUST: stage yaw axis attitude thrust in N. If not specified a default value is computed.
- ROLLTHRUST: stage roll axis attitude thrust in N. If not specified a default value is computed.
- ENG_i : (x,y,z) vector position of exhaust mesh for thrust rendering. ‘i’ stands for 1, 2, 3 ... up to 32, allowing up to 32 main engines per stage. There can be no hole in the sequence. If present, the parameter ENG_DIAMETER must be defined. By default, only one engine will be rendered on the Z axis of the stage
- ENG_DIAMETER : this parameter is only mandatory if ENG_i is defined. By default, the exhaust scale is half the diameter of the stage.
- ENG_TEX : texture file name for engine texture rendering (the file must exist in the \texture subdirectory)
- ENG_PSTREAM1 : name of particle stream attached to the engine
- ENG_PSTREAM2 : name of a second (optional) particle stream attached to the engine

[SEPARATION_xy]

interstage definition section where xy stands for 12, 23, 34 ... up to 910.
There can be no separation section. Separation_xy can only exist if a Stage_y exists.

Mandatory parameters
- MESHNAME : the mesh name of the interstage
- OFF : (x,y,z) vector position of interstage mesh offset in launcher (meters)
- HEIGHT : interstage height in meter
- DIAMETER : interstage diameter in meter
- EMPTYMASS : interstage mass in kg

Unmandatory parameters
- SEPARATION_DELAY : the delay before separation after previous stage burn out in second
- SPEED : translation speed for interstage jettison (in m/s)
- ROT_SPEED : rotation speed for interstage jettison (in rad/s)
- MODULE : name of the vessel class to be created when interstage is jettisoned. Default class is “stage”.

[FAIRING]
Fairing definition section. There can be no fairing

Mandatory parameters
- N : number of fairing elements
- MESHNAME : mesh name root , fairing meshes must be <meshname>_1.msh, <meshname>_2.msh, ..., <meshname>_N.msh and must be properly oriented with the launcher
- OFF : (x,y,z) vector position of first fairing mesh offset in launcher (meters)
- ANGLE : the initial angle that the OFF vector will be turned around Z axis for effective fairing offset
- HEIGHT : fairing total height in meter
- DIAMETER : fairing total diameter in meter
- EMPTYMASS : fairing total empty mass in kg

Unmandatory parameters
- SPEED : translation speed for fairing 1st element jettison (in m/s)
- ROT_SPEED : rotation speed for fairing 1st element jettison (in rad/s)
- MODULE : name of the vessel class to be created when fairing is jettisoned. By default “stage” class.

[PAYLOAD_x]
payload definition section where x stands for 1, 2, 3 ... up to 10. There can be no hole in the sequence. There can be no payload section.

Mandatory parameters
- MESHNAME : mesh name of payload. It has been found that all payloads must have a different mesh name or Orbiter will crash. 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 : (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)
- HEIGHT : payload height in meter
- DIAMETER : payload diameter in meter

- MASS : payload mass in kg
- MODULE : 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

Unmandatory parameters
- SPEED : translation speed for payload jettison (in m/s)
- ROT_SPEED : rotation speed for payload jettison (in rad/s)
- RENDER : if set to 1, the payload will be rendered even when fairing is still on the launcher. This is usefull if the fairing is not completely hidding the payload.

Sound parameters
Support for OrbiterSound version 3 is available in the sound section

[SOUND]
- 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_CONDITIONNING : specify the sound file to be played for air conditionning
- 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

Atmospheric parameters
The following parameters are computed according to the following approximations :
The launcher is always approximate by a cylinder :
- The mass of the cylinder is the sum of the mass of the components.
- The height of the cylinder is the height occupied by the components.
- The diameter of the cylinder is computed to have an equivalent volume of the total volume of the components.
- The PMI are those of the cylinder
- The cross sections are those of the cylinder
- The drag coefficients are computed according to arbitrary values
- The rotational drag is computed according to arbitrary values

Parameters are updated each time a part of the launcher is jettisoned.
No wing effect is supported
No lift effect is supported
Future version may allow user to specify its own atmospheric parameters for better accuracy.

Guidance Autopilot
Introduction
Multistage is supporting a guidance system which is configurable through a text file.

This is a preliminary version of the software. I know much functions have still not been tested or implemented.

How to use it

The guidance system is foreseen to automate all the launch. It must be started before the launch by typing the ‘P’ key. It can be turned off by pushing again the ‘P’ key but in this case it will not be possible to turn it on again.
The guidance file format is detailed hereafter, the guidance file to be used must be specified in the scenario file by adding a line like the following one in the ship section :

```
  GUIDANCE_FILE Config\multistage\shuttle.txt
```

The guidance file

Each line as the following format :

<mission elapsed time>=<command>(param1,param2,…)

The mission elapsed time is specified in seconds and negative values can be used for operation before launch (main engine start sequence, sound play).

The following commands are supported:

engine(starting_thrust, final_thrust, thrusting_time)
This will control the thrust of the main engine varying it from the value indicated in “starting_thrust” to the value indicated in “final_thrust” in a time duration defined in “thrusting_time”. Thrust is indicated in percent and must be between 0 and 100.

roll(init_pitch_time,init_pitch_angle,heading_target,pitch_target,pitch_mode)
This will control the roll program. The launcher is pitched in the direction indicated by “pitch_mode” (1=pitch up, -1=pitch_down) until it reach the “init_pitch_angle”(in °), then the pitch guidance loop is closed and the pitch is controlled to reach the pitch target. When the “ init_pitch_time” is elapsed, the roll and yaw guidance are closed to reach the proper heading target. This is a complex manoeuvre and setting bad parameters can lead to a catastrophic roll. Take care that the next command must not be issued before the roll is complete.

pitch(starting_pitch,final_pitch,pitch_time)
This will control the pitch program. The launcher is pitched from the “starting_pitch” to the “final_pitch” in a time duration set by “pitch_time”
Note that the pitch segments should join gently together (that is the final_pitch of the previous segment should be the starting_pitch of the next segment)

aoa(aoa_angle)
This will control the angle of attack to “aoa_angle” between the vessel zero angle of attack axis (usually the z axis) and the speed vector.

spin(rate)
This will command the roll up to the specified roll rate (°/sec).You may use positive and negative value to have left or right roll

attitude(pitch,heading,roll)

This will command the vessel attitude to the specified pitch, heading and roll all expressed in degree in the range from -180° to 180° and measured in LVLH (local vertical, local horizontal coordinates system).

Disable(<pitch|roll|jettison>)
This will command disable the specified program. It can be "pitch" program, "roll" program or "jettison" program. Disable(jettison) means that there will be no automatic booster or stage separation when fuel mass reach 0, all jettison command must be issued by the guidance program using the command jettison()

Playsound(<sound file name>)
This will play the specified sound file name on the radio (OrbiterSound is required).

fairing()
This will command the fairing jettison (equivalent to the key command 'F')

jettison()
This will command the jettison of a payload (or stage if applied earlier into the flight, but it is really a bad idea)

target(target_apogee)
Once called, this will constantly check for the current orbit parameters and will command a stop of all engines when the "target_apogee" (in km above surface) is reached

Rules of thumb for making a launcher

If you are making a real launcher, you can get a lot of information at http ://www.astronautix.com about dimension, weight and thrust. Otherwise, you should follow these rules of thumb :

- you should assume that a stage is composed at 9/10 of fuel and at 1/10 of empty mass.
- the total thrust (N) at launch must be at least equal to the total mass (kg) * 9.81
- using real values will require that you fly an optimized profile to reach proper orbit which is nearly impossible without an efficient automatic guidance. you can cheat on the launcher efficiency by increasing the specified thrust or by increasing the burn time.
- for meshes, center all meshes at (0,0,0). This point should correspond to the mesh center of gravity.

Examples

For this version I have consider that all add-ons related to Ariane 4 (meshes by me) and Ariane 5 (wonderfull meshes by Thomas Ruth (TomPA)) are Orbiter add-ons and are no more examples of programmation of multistage.dll.These add-ons are downloadable separately.

The following examples can be downloaded by developpers to see how multistage ini files works.

3-stages launcher

this poor quality meshes rendered launcher is delivered just for the purpose of demonstrating the different possibilities of the multistage module. It is composed of 3 families of boosters, each containing a pair of boosters. These 3 families of boosters will be dropped at 1 second interval. The 2nd stage is immediately mounted a top of 1st stage. The 3rd stage is mounted a top of an interstage between the 2nd and the 3rd stage. The fairing is composed of two pieces. A dual payload is configured. The burn time are very short so you will be able to see the all sequence of flight in 2 minutes but of course this will not be sufficient for reaching orbit. You can edit the demo.ini file to increase the burn time. Two identical launchers are defined in the scenario to illustrate the possibility to have more than one multistage vessel instance.

Atlantis space shuttle

this example illustrate the ability to configure the module to recreate the space shuttle. In this configuration the orbiter tank is considered as the first stage. The 2 SRB constitute the booster group. The orbiter is considered as the payload. The Atlantis module is activated at payload jettison by using spacecraft.dll. The SRB mesh had to be duplicated to work with multistage. A guidance file is also delivered with this example.

Credits/Copyright

3 stage example meshes : all meshes by Vinka. made quickly with Anim8or
Space shuttle : from the original Orbiter package, I only duplicated the SRB mesh
You are free to distribute spacecraft.dll, stage.dll and multistage.dll with your add-ons.
Thanks to BigJimW and Roger "Frying Tiger" Long for beta testing this version.

Known bug

There is not much protection against wrong ini file therefore this will result in Orbiter crash

Support

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

What will come next

- configurable atmospheric flight characteristics
- errors report in ini file
- ullage motor support
- more autopilot functions
- more realistic flight parameters