Helicopter Sling Load

External Load Simulation for X-Plane

1 Introduction

Helicopter Sling Load (HSL) is a X-Plane 11 Plugin for the physical simulation of external loads. The purpose of this plugin is to provide an external load simulation that can be fully configured by 3rd party plugins. While it is possible for users to fly loads just with HSL, this is not the primary goal.

X-Plane is an ever changing environment and therefore plugins need to be maintained on a regular basis to keep working with the current versions. To make sure this plugin can survive even in a case where I cannot actively maintain it, I made it open source. All source code is available at https://github.com/kristian80/HSL and published using the GPL v3 license.

2 Installation

Extract the zip files and copy the HSL folder into your "X-Plane\Resources\Plugins" folder.

3 How to use this plugin

Using the GUI you can place objects in front of the helicopter and connect them with the rope. Then you can take off and release the load where you want to. The cargo will continue the physics computation even after release, so it will glide on the ground if you where not slow enough and it will drop if you release it mid flight. If you exceed the limitations of the rope, it will rupture and the cargo will fall down.

Plugins may place the cargo at any coordinates in the world, by accessing the corresponding datarefs and commands.

When connecting the cargo with a rope that is too short, the winch will extend it further. This way you do not have to worry about the helicopter being soaked towards the cargo.

You can use any objects within the X-Plane objects library as a cargo object. For this you need the library path of the objects. A simple method for finding nice objects is to download WED and create a dummy scenery. When opening the scenery you will be able to browse all installed X-Plane libraries, including all the custom scenery you have installed. Just hit the Update Objects button when finished. In case of an error, the plugin will be disabled. Be aware that the GUI is very sensitive to keyboard input and hitting CTRL+V often pastes the text twice.

4 How it works

The computation of a sling load is based on differential equations that cannot be solved analytically. Hence, HSL uses the corresponding difference equation for the numeric approximation. The accuracy of this numeric approximation is depending on the step size. If the step size would be infinitesimal small, the result would be precise.

As it turned out, for small loads the x-plane flight loop time resolution is on the limit even with four flight loops per frame and 100 fps. Furthermore, even small stutters resulted in unrealistic object behavior. Therefore, I have decoupled the physics computation from the x-plane main thread and all the computations run asynchronously in a separate thread. On each flight loop the information about the

helicopter position, speed and acceleration is updated. Using this information the plugin computes a virtual helicopter that continues to fly smoothly, even if your x-plane is currently stuttering. This way, I can use far smaller time steps for the computation than the flight loop would allow, resulting in an appropriate accuracy of the numerical approximation.

To suit high and low performance computers the plugin supports two modes for the computation:

High Performance Disabled: Recommended for CPUs with less cores. The physics thread uses the sleep command after each computation. The x-plane thread goes to sleep whenever waiting for the physics thread. This way the physics thread is prompting the Windows scheduler to perform other tasks in-between computations and resources are released as soon as one thread is waiting for the other one. As the time for the physics computation is magnitudes smaller than the minimum sleep time, the X-Plane flight loop will rarely have to wait for a physics computation to finish.

High Performance Enabled: Recommended for CPUs with lots of cores. Both, the physics thread and the x-plane flight loop will wait for the Mutex in a polling loop. If both run on separate cores, this will minimize the step size of the physics computation. Just try what gives the best result. The Windows scheduler is able to handle high CPU loads, so even with a low core count the High Performance mode might give better results.

For fire fighting the drops released by the bambi bucket are also computed in a separate thread. This thread runs synchronously to the flight loop thread, without blocking. If you have too many drops to be computed in-between flight loops, the drops will simply slow down. For performance reasons the number of drops that is drawn is limited, but they will still be considered for extinguishing fires.

4.1 What's included

- Gravity:
  - Using the X-Plane dataref. Yes, it should also work on Mars.
  - Rope forces: The rope is considered an single mass oscillator.
  - Stretch forces.
  - Damping forces according to the speed the rope is stretching.
- Air drag: The object is considered cuboid.
  - Object speed plus wind.
  - Air density depending on altitude and weather, using the X-Plane dataref.
  - CW value for each side of the cuboid.
  - Surface of each side.
  - Roll and pitch of the object are considered.
- Water drag:
  - Same as the Air drag, except that the object is always upright as it hits the ground.
  - Air and water drag are weighted according to the water line. This means that the wind might move a floating object.
- Water displacement:
  - Upward forces by the water displacement. If the object is lighter than water, it will swim.
  - Bambi buckets don't swim and fill with water instantly, according to the water line.
- Friction:
  - Static friction when the object is still on the solid ground.
  - Glide friction when the object is moving on the solid ground.
  - Damping by the operator: When lifting an object, the operators on winch or on the hook will usually try to stabilize the load. Otherwise the short rope would start hazardous oscillations.
  - Rope length where the operators start the stabilization.
  - Maximum force the operator may apply .

- For 3rd party plugins:
  - Plugin may apply additional forces (e.g. to simulate ground operator).
  - Plugin may rotate the cargo.
- Forces on the Helicopter:
  - Forces of the rope.
  - Momentum that is caused by the positioning of the winch.

4.2 What's missing

- Cargo rotation (except rotation by external plugin)
- Rotor downwash
- Ground impact:
  - When hitting the ground, the vertical speed is put to zero. No bouncing off the ground.
  - When hitting the ground, the object is always upright. No flipping over.
- There is no momentum on the load. All forces apply in the CoG, which is considered at the rope mounting point.
- Bambi bucket limited water flow when dropped into water
- Terrain steepness is not considered. For HSL the terrain is always flat.
- Cargo maximum forces. Cargo never breaks, no matter how hard it hits the ground.
- Full rope simulation, currently the rope is always straight.
- Fire temperature increase of the surroundings
- Fire turbulence simulation
- Fire in replay

5 Commands

Commands can be bound either to a hotkey or joystick button, or they can be called by 3rd party plug-ins using the X-Plane SDK.

Command	**Description**
`HSL/Winch_Up`	Move the winch up
`HSL/Winch_Down`	Move the winch down
`HSL/Winch_Stop`	Stop the winch
`HSL/Sling_Enable`	Enable slingline
`HSL/Sling_Disable`	Disable slingline
`HSL/Sling_Reset`	Reset slingline
`HSL/Load_Connect`	Connect the load
`HSL/Load_Release`	Release the load
`HSL/Load_On_Ground`	Place the load in front of the aircraft
`HSL/Load_On_Coordinates`	Place the load at given coordinates
`HSL/ToogleControlWindow`	Toggle Control Window
`HSL/UpdateObjects`	Update Objects
`HSL/Fire_On_Ground`	Place fire in front of the aircraft
`HSL/Fire_On_Coordinates`	Place fire at given coordinates
`HSL/Bucket_Release`	Release Water in Bambi Bucket

6 DataRefs

6.1 Writeable Values

6.1.1 Object Paths

Type	DataRef Name	Description
String (dynamic)	HSL/WinchObjectPath	Library or physical path (relative from XP folder) of the winch object. Library is tried first. Send command HSL/UpdateObjects to reload objects.
String (dynamic)	HSL/RopeObjectPath	Library or physical path (relative from XP folder) of the rope objects. Rope objects are not rotated, use spherical objects only. Library is tried first. Send command HSL/UpdateObjects to reload objects.
String (dynamic)	HSL/HookObjectPath	Library or physical path (relative from XP folder) of the hook object. Library is tried first. Send command HSL/UpdateObjects to reload objects.
String (dynamic)	HSL/CargoObjectPath	Library or physical path (relative from XP folder) of the cargo object. Library is tried first. Send command HSL/UpdateObjects to reload objects.

6.1.2 Winch

Type	DataRef Name	Description
Float Array[3]	HSL/Winch/VectorWinchPosition	Position of the winch in aircraft coordinates.
Double	HSL/Winch/WinchSpeed	Up/Down speed of the winch [m/s]

6.1.3 Rope

Type	DataRef Name	Description
Double	HSL/Rope/RopeLengthStart	Length of the rope upon plugin start or reset
Double	HSL/Rope/RopeLengthNormal	Deployed length of the rope, without expansion
Double	HSL/Rope/RopeDamping	Damping factor of the rope. Has to be between zero (undamped) and one (critical damping).
Double	HSL/Rope/RopeK	Force applied when the rope is expanded to twice its normal size [N].
Double	HSL/Rope/RuptureForce	Force that causes the rope to rupture.
Double	HSL/Rope/MaxAccRopeFactor	Not used anymore.
Double	HSL/Rope/OperatorForce	Force of the operator to dampen the oscillation.
Double	HSL/Rope/OperatorLength	Rope length when the operator start dampening the oscillation.

6.1.4 Hook

Type	DataRef Name	Description
Double	HSL/Hook/Height	Height of the hook object when drawing the hook on the ground [m]. Only for visualization.
Double	HSL/Hook/Mass	Mass of the hook [kg]
Float Array[3]	HSL/Hook/Size	Dimensions of the hook: Depth/Width/Height [m]. Only for physics, not for visualization.
Float Array[3]	HSL/Hook/Rotation	Rotation of the hook [deg]. Only for visualization.
Float Array[3]	HSL/Hook/CW	Aerial resistance of the hook surfaces: Front/Side/Top.
Double	HSL/Hook/FrictionGlide	Glide friction of the hook, when on solid ground. Relative value [N/N]: Horizontal resistance force resulting from vertical pressure force.
Double	HSL/Hook/FrictionStatic	Static friction of the hook, when on solid ground. Relative value [N/N]: Maximum horizontal resistance force resulting from vertical pressure force.
Float Array[3]	HSL/Hook/ExternalForce	External forces applied on the hook (e.g. by an operator) [N].

6.1.5 Cargo

Type	DataRef Name	Description
Double	HSL/Cargo/SetLatitude	Latitude value when placing a new cargo at coordinates. Use for command HSL/Load_On_Coordinates.
Double	HSL/Cargo/SetLongitude	Longitude value when placing a new cargo at coordinates. Use for command HSL/Load_On_Coordinates.
Double	HSL/Cargo/Height	Height of the cargo object when drawing the cargo on the ground [m]. Only for visualization.
Float Array[3]	HSL/Cargo/RopeOffset	Offset of rope in relation to the origin of the cargo object [m]. Only for visualization.
Float Array[3]	HSL/Cargo/Rotation	Rotation of the cargo [deg]. Only for visualization.
Double	HSL/Cargo/Mass	Mass of the cargo [kg]
Float Array[3]	HSL/Cargo/Size	Dimensions of the cargo: Depth/Width/Height [m]. Only for physics, not for visualization.
Float Array[3]	HSL/Cargo/CWFront	Aerial resistance of the cargo surfaces: Front/Side/Top.
Double	HSL/Cargo/FrictionGlide	Glide friction of the cargo, when on solid ground. Relative value [N/N]: Horizontal resistance force resulting from vertical pressure force.
Double	HSL/Cargo/FrictionStatic	Static friction of the cargo, when on solid ground. Relative value [N/N]: Maximum horizontal resistance force resulting from vertical pressure force.
Integer	HSL/Cargo/IsBambiBucket	Tells the plugin if the cargo is a bambi bucket, either 0 or 1. Bambi buckets don't swim and fill up their volume when lowered into water.
Integer	HSL/Cargo/BambiBucketReleaseWater	Tells the plugin to release the water in the bambi bucket (either 0 or 1).
Float Array[3]	HSL/Cargo/ExternalForce	External forces applied on the cargo(e.g. by an operator) [N].

6.1.6 Fire

Type	DataRef Name	Description
String (dynamic)	HSL/Fire/FireAircraftPath	Physical path (relative from XP folder) of the AI aircraft that is used to draw the fire.
Double	HSL/Fire/WaterRadius	Within this radius, water drops are counting towards extinguishing the fire [m].
Double	HSL/Fire/StrengthStart	Start value for water required to extinguish the fire [kg].
Double	HSL/Fire/StrengthMax	Maximum value for water required to extinguish the fire [kg].
Double	HSL/Fire/StrengthIncrease	Fire increasing over time,water weight increase per second [kg/s].
Double	HSL/Fire/SetLatitude	Latitude value when placing a new fire at coordinates. Use for command HSL/Fire_On_Coordinates.
Double	HSL/Fire/SetLongitude	Longitude value when placing a new fire at coordinates. Use for command HSL/Fire_On_Coordinates.
Double	HSL/Fire/SetElevation	Elevation over ground when placing a new fire at coordinates. Use for command HSL/Fire_On_Coordinates.
Integer	HSL/Fire/UpdatePositions	When 1: Recalculate the fire positions. XP world coordinate positioning is not precise when the aircraft is far away. Hence, it is advisable to re-calcuate the position when the aircraft gets closer.
Integer	HSL/Fire/RemoveFires	When 1, all fires are removed.

6.2 ReadOnly Values

6.2.1 Calculated Values

Type	DataRef Name	Description
Double	HSL/Calculated/FrameTime	Time between flight loops.
Double	HSL/Calculated/NewRopeLength	Current rope length with expansion.
Double	HSL/Calculated/RopeStretchRelative	Relative value of the expansion (0 = no expansion, 1 = twice the size)
Double	HSL/Calculated/RopeForceScalar	Scalar value of the rope force [N]
Double	HSL/Calculated/RopeStretchSpeed	Speed of the expanding rope [m/s]. Negative value for contracting rope.
Double	HSL/Calculated/RopeCorrectedD	Damping value for the current rope load = 2 * D * sqrt(m/K)
Float Array[3]	HSL/Calculated/VectorHelicopterPosition	Position of the winch in OpenGL coordinates.
Float Array[3]	HSL/Calculated/VectorHookPosition	Position of the end of the rope, no matter what's connected.

6.2.2 Rope

Type	DataRef Name	Description
Integer	HSL/Rope/RopeRuptured	1 if the rope is ruptured, 0 otherwise.

6.2.3 Hook

Type	DataRef Name	Description
Integer	HSL/Hook/Connected	1 if the hook is connected to the rope, 0 otherwise. Hook only disconnects on rope rupture.
Integer	HSL/Hook/FollowOnly	1 if the cargo is connected, 0 otherwise.
Integer	HSL/Hook/DrawingEnabled	1 if the hook is currently drawn, 0 otherwise.
Integer	HSL/Hook/InstancedDrawing	1 if instanced drawing is enabled.

6.2.4 Cargo

Type	DataRef Name	Description
Float Array[3]	HSL/Cargo/Position	Position of the cargo.
Integer	HSL/Cargo/Connected	1 if the cargo is connected to the rope, 0 otherwise.
Integer	HSL/Cargo/FollowOnly	1 if the cargo is neither connected nor placed on terrain, 0 otherwise.
Integer	HSL/Cargo/DrawingEnabled	1 if the cargo is drawn, 0 otherwise.
Integer	HSL/Cargo/InstancedDrawing	1 if instanced drawing is enabled.
Double	HSL/Cargo/BambiBucketWaterWeight	Weight of the water in the bambi bucket [kg]
Double	HSL/Cargo/BambiBucketWaterLevel	Water level in the bambi bucket, between 0 and 1. This is computed, but may be written too.

6.2.5 Fire

Type	DataRef Name	Description
Integer	HSL/Fire/CreateFailed	Fire creation failed.
Double	HSL/Fire/Count	Number of active fires.
Float Array[10]	HSL/Fire/FireStrength	Strength of the currently active fires, in water weight [kg].

7 License