Moving the Plane

Last updated July 12, 2018

Positioning Aircraft in X-Plane

This note describes how to position the user's aircraft or multiplayer aircraft in X-Plane. Positioning the user's aircraft can be broadly divided into two cases:

1. Positioning the user's aircraft once while the physics engine is running. *Example: positioning the user for an ILS approach in a training situation.*
2. Positioning the user's aircraft continuously while the physics engine is not running. *Example: visualizing flight data recorder information.*

Generally most datarefs for positioning the user's aircraft are contained in sim/flightmodel/position/.

A Note on Precision

All datarefs in this section are listed on the web page as either type int, float, or double. When a dataref is listed as float, it is internally stored in 32-bit floating point format. When a dataref is listed as double on the web page, it is internally stored in 64-bit double precision floating point format, but is accessible as both a float and a double for convenience. You may use either the double or float routines to write the dataref, depending on the precision you need.

Generally you will need to use double precision to work with latitude and longitude precisely. Because the local coordinate system moves to keep the aircraft near the origin, single precision should be adequate for cartesian XYZ values. The routines XPLMWorldToLocal and XPLMLocalToWorld work in double precision on the latitude-longitude side.

Positioning the User's Aircraft While the

Physics Engine is Running

While the physics engine is running, you may position the user's aircraft; it will instantly move and then continue in the direction it was flying (or the direction you specify). Do not continuously reposition the user's aircraft with the physics engine running; your positioning commands and the physics engine will thrash the aircraft around.

Positioning the Aircraft in Space

To place the user's aircraft, you position it in local coordinates (see ScreenCoordinates). This is a cartesian axis in meters. Use these datarefs:

sim/flightmodel/position/local_x
sim/flightmodel/position/local_y
sim/flightmodel/position/local_z

You can read or write these datarefs at any time. You cannot write these datarefs:

sim/flightmodel/position/latitude
sim/flightmodel/position/longitude
sim/flightmodel/position/elevation
sim/flightmodel/position/y_agl

X-Plane will calculate the latitude/longitude and elevations (AGL and MSL) for you from the local cartesian coordinates. If you need to place the aircraft based on latitude/longitude information, use the function XPLMWorldToLocal to convert coordinates.

Example: add 10 from sim/flightmodel/position/local_y; the aircraft hops 10 meters up in the air and then continues.

Orienting the Aircraft in Space

The aircraft's orientation is described by three rotations, "psi" (heading), "theta" (pitch), and "phi" (roll). However when the physics model is on, things are a bit more complicated. The datarefs

sim/flightmodel/position/psi
sim/flightmodel/position/theta
sim/flightmodel/position/phi

are read by the graphics engine to draw the aircraft. They are written by the physics model. The data ref

sim/flightmodel/position/q

is read and written by the physics model and is the **master** copy of the aircraft's orientation when the physics model is in. Every frame, the physics model updates q and then copies the values to psi, theta and phi. (Warning: datarefs are case sensitive; lower case q here is the quaternion aircraft rotation, while Q is a rotational rate of the aircraft.)

The dataref q here is a quaternion, stored as an array of four floats. Quaternions are mathematical constructs that (among other things) can fully describe an arbitrary rotation in 3-dimensional space. If you want to reorient the aircraft, you must write to q. Like local_x, local_y, and local_z, this will affect the sim immediately and the sim will continue from that position.

The quaternion is ordered {1, i, j, k}. The axes correspond to:

Quaternion	Euler	OpenGL
i	roll	-z
j	pitch	x
k	heading	-y

The math to convert from Euler to quaternion is:

```
psi' = pi / 360 * psi
theta' = pi / 360 * theta
phi' = pi / 360 * phi
q[0] =  cos(psi') * cos(theta') * cos(phi') + sin(psi') * sin(theta')
q[1] =  cos(psi') * cos(theta') * sin(phi') - sin(psi') * sin(theta')
q[2] =  cos(psi') * sin(theta') * cos(phi') + sin(psi') * cos(theta')
q[3] = -cos(psi') * sin(theta') * sin(phi') + sin(psi') * cos(theta')
```

The following datarefs cannot be written:

sim/flightmodel/position/alpha
sim/flightmodel/position/beta
sim/flightmodel/position/hpath
sim/flightmodel/position/vpath
sim/flightmodel/position/magpsi

They will be calculated based on the orientation you set, the aircraft's velocity, and the local wind conditions and magnetic variation.

Example: write {0.8, 0, 0, 0.6} to sim/flightmodel/position/q. The aircraft points roughly to the east-northeast.

Changing the Aircraft's Velocity

While the sim is running, you can change the aircraft's velocity, both linearly and rotationally. The linear velocity of the aircraft is controlled by three values:

sim/flightmodel/position/local_vx
sim/flightmodel/position/local_vy
sim/flightmodel/position/local_vz

This determines both the aircraft's direction (in 3-d space) and its speed. Units are meters per second. This vector is in the world coordinate system; a velocity along the X axis moves the aircraft east no matter which way the aircraft is heading. The datarefs

sim/flightmodel/position/groundspeed
sim/flightmodel/position/indicated_airspeed
sim/flightmodel/position/indicated_airspeed2
sim/flightmodel/position/true_airspeed
sim/flightmodel/position/vh_ind
sim/flightmodel/position/vh_ind_fpm
sim/flightmodel/position/vh_ind_fpm2

are all derived from this one velocity vector. Please note that the values alpha, beta, hpath and vpath are also affected by this velocity vector; if the velocity vector points along the X axis but the aircraft is oriented along the Z axis, then the aircraft will be in a very heavy state of yaw.

Example: write 20 into sim/flightmodel/position/local_vy. The aircraft is launched up in the air.

You can also spin the aircraft by writing to the datarefs P, Q, and R, which are the rotation rates of the aircraft in degrees per second. Positive P rolls the aircraft to the right; Q pitches the aircraft up; positive R yaws the aircraft to the right.

sim/flightmodel/position/P
sim/flightmodel/position/Q
sim/flightmodel/position/R

You cannot write to the angular momentum datarefs N, M, and L; these are written by the flight model every frame.

You also cannot write to P_dot, Q_dot, or R_dot (angular accelerations) or the linear accelerations (local_ax, local_ay, and local_az).

Example: write a value of 20 to sim/flightmodel/position/Q. The aircraft rotates upward like Dr. Dre's car.

WARNING: The interaction of P, Q, and R vs. Prad, Qrad and Rrad is pretty sim specific. I do not recommend attempting to change the rotation rate of the aircraft.

Applying Arbitrary Forces to the Aircraft (10.30 and later)

The datarefs mentioned here are only available in X-Plane 10.30 and later.

X-Plane 10.30 allows you to add arbitrary forces to the aircraft; the sum of all forces from X-Plane and plugins are considered in total to determine the behavior of the aircraft. (For example, if you apply a force pushing your aircraft to the right while it is on the ground, it may actually roll to the right, even if you push on the CG, because the tires will 'stick' and tip the airplane over.)

The datarefs for plugin forces are meant to be additive and shared between plugins:

– Every frame, X-Plane will reset these datarefs to zero.

– In your flight-loop callback, _add_ the amount of force you want to apply by reading the datarefs, adding, and writing the results back.

– Because the results are additive, you can easily apply multiple distinct forces to the plane.

– Because the results are additive, multiple plugins can add force at the same time. For example, an ACF plugin can implement a custom engine by adding force while a push-back plugin pushes back on the airplane.

The coordinate system for adding force is the aircraft coordinate system – location is in meters with 0,0,0 at the normal CG of the aircraft; if the CG has been customized by the user, you do not need to change your physics – X-Plane accounts for this automatically after considering plugin force.

There are three force datarefs:

sim/flightmodel/forces/fside_plug_acf
sim/flightmodel/forces/fnrml_plug_acf
sim/flightmodel/forces/faxil_plug_acf

The units are Newtons – fside is the X xis, fnrml is the Y axis, and faxil is the Z axis. Note that the positive Z axis points to the back of the plane, so most engines apply NEGATIVE force to faxil_plug_acf.

There are also 3 rotational moment datarefs – the units are newton-meters:

sim/flightmodel/forces/L_plug_acf
sim/flightmodel/forces/M_plug_acf
sim/flightmodel/forces/N_plug_acf

L is a roll moment (positive is right roll), M is pitch (positive is nose up) and N is yaw (positive = yaw right/clockwise from above the ACF. Because you can apply moments, you can apply forces to arbitrary locations on the aircraft. Given a force vector Fx, Fy, Fz applied at a location X,Y,Z, the additions to these six datarefs are:

```
fside+=Fx;L+= Fx*Y - Fy*X;
fnrml+=Fy;M+= Fz*Y - Fy*Z;
faxil+=Fz;N+= Fz*X - Fx*Z;
```

This makes some sense intuitively: L (right-roll) increases when you apply a right force above the aircraft (Fx * Y) and decreases when you apply an up force to the right side of the aircraft (Fy * X).

Positioning the User's Aircraft While the Physics Engine is Not Running

The dataref

sim/operation/override/override_planepath

Controls whether X-Plane's physics engine is controlling the user's aircraft. It contains one element for each aircraft. You can set the first element to one to disable the physics model.

WARNING: Do not use this dataref to disable X-Plane's control of the AI aircraft; use the XPLMMultiplayer API XPLMDisableAIForPlane described below instead. Future SDKs may not support more than one element in sim/operation/override/override_planepath.

Once you disable the physics model, the following things change:

– The quaternion rotation is no longer used or copied to psi, theta and phi. Instead, change the aircraft's orientation directly by writing to psi, theta, and phi.

– The aircraft will basically hold any position and orientation you set.

– Derived variables like indicated_airspeed and alpha will not be updated.

Once you reenable the physics model, the aircraft will continue based on 'q', the quaternion rotation, not the orientation you wrote with psi, theta and phi.

Example: write 1 to sim/operation/override/override_planepath[0]. Then write 40 to sim/flightmodel/position/theta and 100 to sim/flightmodel/position /indicated_airspeed. The aircraft will pitch up 40 degrees and indicate 100 knots, but the stall horn will not go off. Then write 40 to sim/flightmodel/position/alpha. The stall horn will sound.

WARNING: The interaction with the instrument system when physics are disabled is a bit complex and may be subject to change.

Transitioning From Disabled to Enabled Flight Model

When you re-enable the flight model, the sim will continue based on the velocity vector and quaternion q. So if you have set up the aircraft based on phi, psi, and theta for orientation, and some kind of speed, you will need to set the velocity vector to give the aircraft speed along its current heading, and conform the quaternion q to the current rotations. Once you do this, re-enabling the flight model should be relatively smooth.

Moving the Aircraft A Long Way (Around the Earth)

(This is a paste from an email to the dev list…it needs cleanup.)

Starting with X-Plane 850 beta 9, plugins now can position the plane anywhere in the world. Here are some details (all datarefs are in sim/flightmodel/position/):

1. You do not ever have to override the flight model to move the plane. If you alter local_x, local_y and local_z the plane jumps instantly.

2. To move the plane a small amount, simply change local_x, local_y, and local_z.

3. To move the plane a LONG way, you must change local_x, local_y and local_z as well

as latitude and longitude. *Use XPLMWorldToLocal or XPLMLocalToWorld to get the "other" coordinate system (depending on whether your positioning is based on cartesian or geographic coords). This will only work in 850 beta 9 and newer.*

4. Don't position the plane inside the earth. Enough said. 😉
5. You do not need to, should not, and cannot write to lat_ref and lon_ref. The sim will pick this value for you as it processes scenery.
6. The orientation of the plane is in cartesian coordinates. So if you position the plane from latitude 45N to latitude 45S, the plane will be rotated 90 degrees relative to the horizon. *This is because the horizon has rotated 90 degrees and the plane has not.*
    1. The coordinate shift happens after you write your datarefs. So…you'll need to calculate the change in attitude/rotation yourself (and write the 'q' variable – see the SDK tech notes for more on positioning the plane) as you write xyz, anticipating the horizon angle changing.
7. In the future it may not be necessary to write lat/lon and xyz, but it won't be harmful. Please always write CONSISTENT results – e.g. don't write a lat/lon that are different from xyz (based on the current XPLMWorldToLocal and XPLMLocalToWorld conversions).

Controlling the Other Aircraft

X-Plane also features up to nineteen other multiplayer aircraft. Generally they are controlled with the sim/multiplayer/position datarefs. Before controlling the aircraft, your plugin should call XPLMAcquireAircraft. This gives your plugin exclusive rights to these multiplayer aircraft. You will lose control of the aircraft when your plugin is disabled, so call XPLMAcquireAircraft whenever your plugin is enabled. If XPLMAcquireAircraft fails, it probably means another plugin is using multiplayer aircraft.

Once you have control of the aircraft, you can disable X-Plane's control over their position by calling XPLMDisableAIForPlane. The aircraft will now hold still until you reposition it.

Controlling AI Aircraft in X-Plane 6, and 7

(This also applies to X-Plane 8.00 – 8.40)

Up through v8.40, AI aircraft were controlled by simple calculations constituting an "AI

Flight Model". This can be overridden by setting the Sim/operation/override /override_planepath dataref for the aircraft you wish to control. When this dataref is set, the aircraft will not move on its own – you will need to provide a continuous set of positions and rotations, updated once per frame. These are controlled by sim/multiplayer/position/* datarefs found here.

Controlling AI Aircraft in X-Plane 8 and 9

AI aircraft feature a full flight model for the aircraft it is currently flying. The AI controls the aircraft via autopilot inputs. Just like v8.40, setting the Sim/operation/override /override_planepath dataref will override this behavior. As such, you will need to provide the same stream of positions and rotations for that aircraft, updated once per aircraft.

In v8.60, the AI aircraft feature a full flight model and are controlled in the same manner by the AI – via autopilot inputs. However, as of v8.60 we have a new dataref available, Sim/operation/override/override_plane_ai_autopilot. Setting this dataref allows us to disable the AI control of the aircraft, while retaining the full flight model. You are then able to control the aircraft by setting the auto-pilot datarefs for that aircraft, just as the AI would do on its own. See sim/multiplayer/autopilot/ and sim/multiplayer/controls/ (see Resources/plugins/DataRefs.txt for available datarefs).

Contents

Developer resources for the X-Plane flight simulator

Never miss an update

Sign up for email updates below to get the latest X-Plane news in your inbox.

you@domain.com

SIGN UP

We will never sell or share your email address; we'll only use it to send you X-Plane-related emails. (More info in our privacy policy.)

You'll receive about one email a month, and you can unsubscribe at any time.