X-SIM DEVELOPER GUIDELINES

Table of content

 20.2.2013

Programming Guideline for simulators that use X-Sim

1. Introduction

This document will describe the guideline how a developer of a simulator or dashboard is able to implement his hardware and software into the core of X-Sim. You will get details of the main engine of X-Sim, how it works and what you get for your own implementation. It is designed for programmers, interface developers and plugin writers. It is not an end consumer document. Please read the current software license to get involved with the legal usage. A commercial usage of any X-Sim version without a license is not possible.

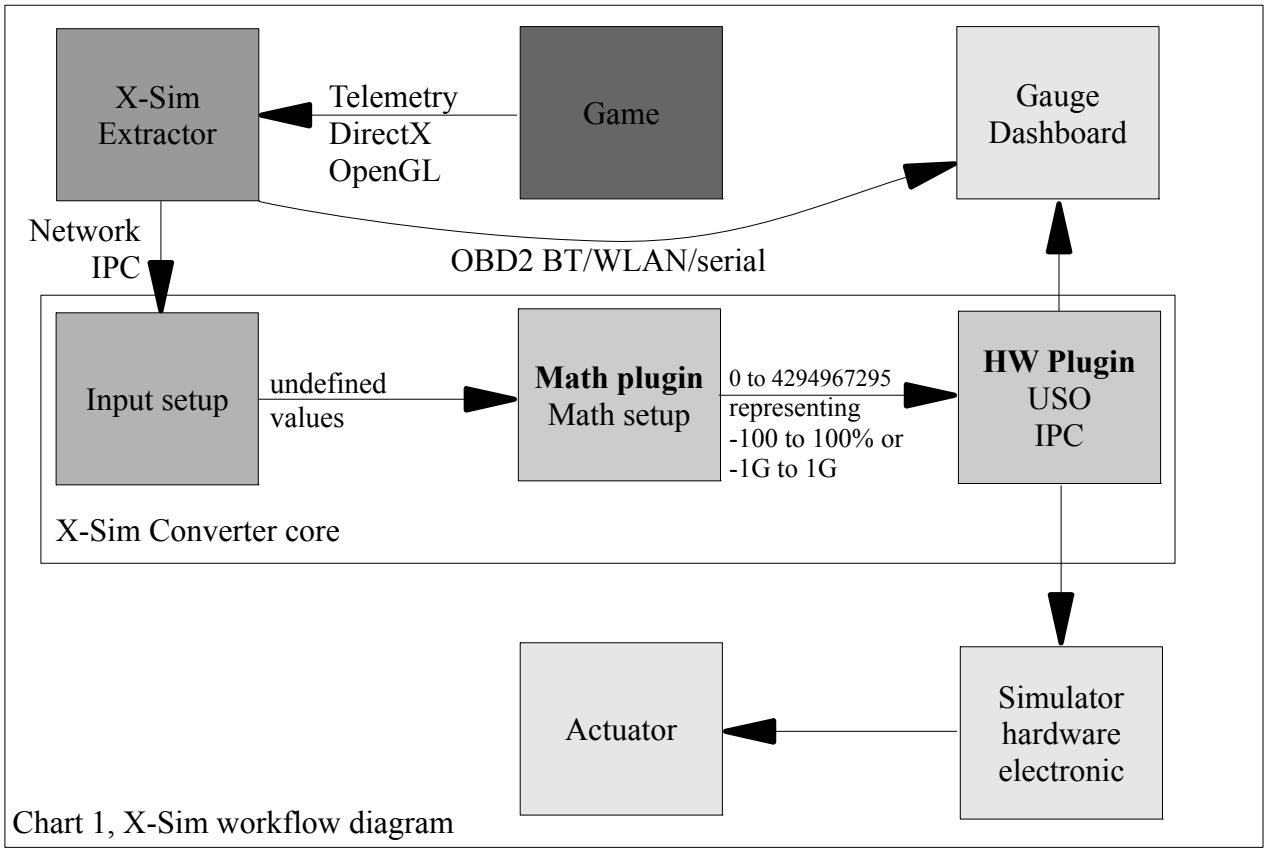

Chart 1, X-Sim workflow diagram

2. Guidelines for an integration

The next chapter will help you to integrate your simulator or dashboard into X-Sim with the option of an own user interface. We recommend to hold this guideline for a professional integration. Of course you can use another way of integration but if you like the response from the community you should give them the user friendly way. A hack from behind might get very complex for most of the users. Developers get support in source code, samples and programming help. If you have problems with integration or you need support for your language, you might contact the forum or the author for support. A workaround of the basics guidelines will make you much more work.

The next statements are the result of years of development in robotic and simulator technology. Most of them make direct sense to you, some are the result of the feedback of X-Sim members that do not have programming skills but want an easy usable software. Simple keep in mind that your followers can give you information's about bugs and the quality of the interfaces.

2.1. Basic guideline for simulator development with X-Sim

The main suggestion of this document is very simple.

a.) Use X-Sim as input telemetry scaler and build a simulator that works with -1G to 1G in the directions you can support. X-Sim will offer you an output of 0 to 4294967295 (32bit unsigned integer) which represent -1G to 1G or better noted as -100% to 100% of your actuator movement. 2147483648 is the zero position. Do NOT pass through the telemetry 1:1 and do NOT try to solve the scaling by yourself, you cannot reach this goal with low resources. X-Sim offers statistic tools, scalers and washouts to get this working for you.

b.) The end user must have access to change his own profile or make your standard setup better for his personal feelings. For simple 2DOF/3DOF kinematics you can use the math section and store the kinematics as a hardware standard .rn2 profile for your customer. Is this your case you have only to write a hardware plugin which will take the -100% to 100% of X-Sim and move an actuator without the need of an own math plugin and without the need of an own GUI inside of X-Sim. Your motion controller must solve the movement with PID and dynamics control in realtime. If you use a complex simulator like a steward platform you must program a kinematics math plugin. This for two reasons. The first reason is to avoid a complex user math setup by adding your own GUI into X-Sim and the second reason is to avoid the usage of an 8bit non floating point microprocessor for complex math work.

2.2. Basic guideline for dashboard/gauge development with X-Sim

Dashboard/gauge developer have fixed output values that can be used. This type of export get a 1:1 pass through telemetry data. The hardware interface plugin will offer you all this data and you can write your own hardware access code into them. Another option is the powerful OBD2 interface. It does not use the converter application. You are able to make a dashboard that is also compatible with real cars telemetry. All newer cars own OBD2. The common interface ELM327 is supported worldwide. The limitation of the X-Sim OBD2 is the low support of many different telemetry output (for now only speed, rpm, coolant, gear). The type of the gauge values is always integer.

2.3. Possible X-Sim software and hardware output interface overview:

X-Sim offers many ways to implement a hardware or a software.

Here are the most common ones listed:

- hardware plugin interface, bidirectional, high resolution
- USO, universal serial port and TCP/IP/UDP output with only one direction
- IPC, inter process communication with memory mapped files, one direction
- OBD2 interface for gauge hardware with LAN/WLAN/Bluetooth/RS232/RS485

2.4. Basic guideline for developing a motion interface

Suggested PID and dynamics counts per second of an actuator interface (feedback loops):
Flight simulation - 30Hz
Car simulation with low power actuators - 100Hz
Car simulation with vibration option - 1000Hz

Suggested minimum resolution of reachable positions:
Flight simulation - 4096 positions - 12bit ADC if using a pot
Car Simulation - 256/1024 positions - 8bit/10bit ADC if using a pot

Suggested programming requirement for X-Sim:
An own interface plugin with autodetection.

3. Description of the X-Sim core engine

The software is in most cases used as a simple input scaler of game data until it reaches your hardware. It solves the problem of unknown and instable sizes of input values. It is not recommended to use the software as a 1:1 pass through because you cannot fight against the unknown size of the different input values. The size of a speed or acceleration value will change with the car you use and with the map you drive. For example a F1 car has much more acceleration than a ralley car. Otherwise a F1 car does not have a itense road feeling like you have in a ralley car.

3.1. The telemetry interface core

You will find sample source for game plugins in the X-Sim folders. All telemetry data are collected together with the extractor.exe application. You can combine all values of the different tools but you can only send some of them to the converter.exe application. The motion driver will provide you the gui interface to build such a data packet. Normally you use the standard button because you use one of the drivers and not a combination.

3.1.1. Game plugins

Game plugins are different from game to game. Not all plugins can offer you all 6DOF values and is limited to the values a game studio will give you. The game plugins get support for the game wizard. This means games are automatically detected, their configuration will get changed for a telemetry output and the plugin is loaded and setup for you. The values are physical telemetry data in m/s or G but are multiplied with a factor like 10000 because X-Sim does not use floats for the below reasons. The game wizard can be modified and updated with the script files in the plugin directory of X-Sim (own tutorial available).

3.1.2. Graphic driver

If the game developer do not deliver a telemetry interface you need other tools to get the telemetry data. The graphic driver is a DirectX and OpenGL reader which will read out your camera position and look direction. With this information all the vehicle speeds and accelerations in linear and rotation movement is recalculated. This driver has the best telemetry output but is limited to the supported DirectX and OpenGL version. Games which use only pixelshaders and avoid open standards by calculating all needed information in their own pixel shaders cannot work. The values do not have a known physical size like m/s. It is a value that changes equal to physical effects but is not scaled to a norm. It is like pixel/s instead of m/s. X-Sim will scale it to your hardware.
Application note: “X-Sim Extractor Graphic Driver Guide”

3.1.3. Force feedback driver

The third telemetry alternative is using a connected force feedback wheel. The force feedback driver is a DirectX force feedback reader. In older versions this tool was known as an external software called “Yoda” which is now implemented into one GUI. It reads the game calls to DirectX to your force feedback input device (i.e. wheel) and show each of them. This will not give you the one effect you feel on a force feedback wheel where all this command build a combined effect, it will give you a bunch of values before they are combined to the effect you feel on the force feedback wheel. In most cases the lateral value is worth full for a re-usage in a simulator. The values do not have a known physical size like m/s. It is a value that changes equal to physical effects but is not scaled to a norm. X-Sim will scale it to your hardware.
Application note: Video tutorial of “Yoda in Action”

3.1.4. Memory driver

As forth tool X-Sim delivers a more complex solution with some extra tutorials. The memory driver will read special positions of a game where telemetry data are located. In older versions of X-Sim this tool is known as force injector. You can find fitting profiles in the online database and a cheat engine tutorial in the documentation of X-Sim. As you can read the word "cheat engine" it is not an official way of a game studio, it is a Tool that you can use as a private person but must ask the game developer if you use it commercial. This tool is limited to one patch version of a game. With the next patch version of the game the telemetry data are located in another memory area. The values you get are mostly physical float values that are multiplied with i.e. 10000 like in the plugin driver.
Application note: Manual of X-Sim – Pointer Tutorial

3.2. The kinematic core

Kinematics can be solved with a small math plugin. There are sample source codes in the X-Sim installation folders which can be used for a quick entry. Do not put dynamics or PID into a math plugin. PID and dynamics (controlled PID to motion) are part of a realtime processor. Kinematics (calculation target points without thinking about motion feedbacks of the motor) must be calculated with a 32bit floating point processor that is mostly not available in a PID controller.

3.2.1. Simple math profiles

It is possible to avoid programming own math plugins by using the math setup of X-Sim. There you can add, multiply and scale between each axis as you like. However it is not possible to make kinematics for a steward platform which needs bigger formula calculations. You will find in the manual many different plugin descriptions for simulator usage. The community tells us that it is possible to use the math setup for simulators up to 3DOF with combined or separated dependencies from input values to actuators.

3.2.2. Math plugin

Complex motion platforms like a steward platform have a bunch of different dimensions like the dimension between each actuator. For this X-Sim offers a 6DOF spider plugin which collects all needed input values from your game input value and you can calculate in one big step the output values that can be directly send to your dynamics and PID hardware or which can be sent back to the math setup. The spider plugin should get pre-scaled values with a fitting math line before the spider plugin. You should work with 0 to 4294967295 representing -1G to 1G or 0 to 360° rotation as input value. As setup example of the spider plugin you have to add 6 axis to a new converter plugin. Then you have to add for each axis one of the needed input values with a G-Force plugin. Check them with a game that they are 0 to 4294967295. Then add to each axis the spider plugin.

3.3. The output core

X-Sim offers different types how you can interfacing with hardware:

- hardware plugin interface, bidirectional, high resolution
- USO, universal serial port and TCP/IP/UDP output with only one direction
- IPC, inter process communication with memory mapped files
- OBD2 interface for gauge hardware with LAN/WLAN/Bluetooth/RS232/RS485

3.3.1. Universal serial output USO

This powerfull dialog will help you with the first steps of communicating with a serial interface to your hardware. You can specify there what you like to send to your device with a small line. You can send your serial data over a serial port or network. You can choose unlimited RS232 or RS485 comport adapters or unlimited numbers of TCP/IP/UDP network protocols. The parser will read your wished output and send it to the selected output.
An example: The Parser Line S~a01~~a02~P will send a big "S" as start indicator, the axis 1 value out of the math setup, the axis 2 value and a stop indicator "P". This setup is a quick entry and should not be used as a final solution. It does not have a feedback and is unidirectional. The type of output is limited to 8bit, 16bit or 32bit values in different formats. Bit shifting is supported but a bit merge is not possible. It is a developer dialog for compatibility options.

3.3.2. Hardware interface plugin

A hardware plugin is a .dll file in the /interface plugin folder of X-Sim. You can load an example out of the /other stuff directory of X-Sim and program there an auto detection and communication routine. Complex interfaces must be programmed with this type of plugin. Your interface must provide a detection function for a user friendly overview. For example if you send "hello" it must say "I am here, I am version 1.4". After this scan routine you open each of the found devices and setup the low configuration your device need. You tell X-Sim all the available inputs like a position that can be reached. You have to fill out a routine for immediately access and one for a virtual access for synchronized execution. Last but not least you have a user GUI which opens if you double click the plugin in the interface setup. Here you can show realtime information of your interface and error detections. Normally your also include a dialog which can rename the axis to more common names. Only this plugins can solve high resolution output and user friendly auto detection. You have to select inside the synaptrix a direct output to your interface. For example, if you have setup axis 1 you have a last math value from 0 to 4294967295 which you like to send to your device. You have to set the direct output in the synaptrix to a fitting interface analogue input and your plugin will get this value. If you need only an 8bit value you must devide the input by 16777216 which will lower the resolution and reachable positions of your actuator.

3.3.3. OBD2 interface

You can access the extractor application with a bluetooth serial port, with a serial port or with WLAN if you enable the OBD2 feature in the settings menu. This is a 1:1 copy of the ELM327 interface for automotive usage. You can use the data sheets of the ELM327 chip to access the OBD2 interface. For now only the speed, the rpm count and the coolant is available. If you like more data you can ask the author of X-Sim to send them on not needed OBD2 commands. You can recalculate the acceleration from 0 to 100 at your own code by polling the speed value. If you choose this way your dashboard gets automatically automotive compatible and you can connect your dashboard to a real ELM327 interface in your car. Sample app: Tourque on android market.

3.3.4. IPC Inter process communication

This X-Sim interface is complex to setup and should not be used for a user friendly solution.
IPC interface in windows is also known as memory mapped files. Memory mapped files are available (import) in any windows language and generate a file in the memory where you can access from any program which knows the file name. This exchange is very fast but must be polled, a change will not be noticed by a windows event. In C++ you open the file with the C++ command CreateFileMapping or OpenFileMapping and access the file with MapViewOfFile. X-Sim offers memory slots in the dashboard section where you can export gauge values or you have a 6 axis export in the program setup that is i.e. used for the 3D Simulator simulation.