Using MS Flight Simulator™ scenery engine as a visual system for a real world Flight Simulator

by

Geert Rolf

Nordic Simulator Center, Lillestrøm, Norway
/
Private museum for Retired and Aged Computers,
Winssen, The Netherlands
www.bejaardecomputers.nl

Introduction

This story is about replacing Night Vision System of a Singer Link Miles DC-10 Level B Simulator by MicroSoft™ Flight Simulator™ to be used as a Daylight Vision System (DVS) for cockpit view. This story will take you through project history, technical overview of the interface, real time requirements and major changes we made during development.

Project history

In 2004 Continental Airlines offered a retired DC-10 Simulator for free in order to create space for a new simulator in their Houston training facility. A group of Norwegian flightsimmers led by Jan Fjeld joined together to finance transport and formed a working group called Nordic Simulator Center. They made an agreement with the Skedsmo Videregående Skole (SVS), the aviation education department, where the simulator is placed. Work on rebuilding this Sim started in 2004 after shipping 12 metric tons of hardware to Lillestrøm near Oslo, Norway. [1]

Being a private computermuseum, mainly focussed on Digital Equipment Corporation PDP-11 minicomputers, we (me and museum associate Warner Krelekamp) got involved in the beginning of 2008. Not only after implementing a major hardware upgrade involving replacement of two out of three PDP-11s by newer models and subsequently adapting the software to accept a significant change in hardware architecture, the new configuration worked in the fall of 2009. The DC-10 by that time could be flown blindly using Instruments and AutoPilot for automatic landing. [2]

This DC-10 Simulator still was equiped with its original Night Vision System (NVS), implemented by software on the third PDP-11, a bay of complex hardware and two monitors on top of the cockpit. Some, not all, of the airports defined in the Simulator were covered by specific views defined in NVS. The state of hardware, however, was very bad. One of the monitors was broken. Diagnostic software showed activity in the NVS bay by flashing rows of leds, but nothing happened on the monitors. The manuals showed that the procedures for debugging and tuning NVS were complex and required special tools. Expectations for success were low. The best NVS can give you is runway lighting, accurate runway and taxiways and clouds in gray. All agreed we wanted better.

Original configuration with NVS

The Norwegian DC-10 Simulator has two computers (CPU-A and CPU-B) running the Simulator software and a third computer (CPU-C) to for the Night Vision System (NVS).

A parrallel data channel between CPU-A and CPU-C is used to send packets with position information to NVS. These packets are 32 words (=64 bytes) in size and are sent 20 times a second.
Each packet contains a.o. position in X, Y and Z:

- X: distance right (+) and left (-) of runway using runway threshold as centrepoint.
- Y: is forward (+) or backwards (-) on runway centreline.
- Z: is feet above groundlevel (AGL). At high altitude, Z is silently converted to MSL (Mean Sea Level).

The X and Y coordinates are represented by 24 bit numbers with an accuracy of 1/16th foot. So, we see a hard limit: approximately 86 NM left, right, forward or backward of the runway you run off the visual scale. The DC-10 Simulator has no limit flying all over the world, but out of visual range the visual will turn black, which isn't that odd for a Night Vision System.

Other items in the position data:

- yaw, rate of yaw
- pitch, rate of pitch
- roll, rate of roll
- speeds for X, Y, Z direction
- code for airport/runway used as point of reference
- aircraft lights (on/off)
- 16 bytes unused (64 bytes total)

Coupling between CPU-A and CPU-C consists of two flat cables interfacing two DR11-C parrallel interfaces, one in each machine. Protocol is based on A/B status on these interfaces indicating status like “ready for next databuffer”, “ready for next word”, and “previous word received”. The design of the interface lacks to clear a status when reading the status, forcing the programmer to use interrupts. Simply polling the device status implies inevetible race-conditions. Using interrupts, data is transfered word by word causing > 650 interrupts per second a considerable load for hardware dating from the mid 70s of previous century.

The DC-10 Simulator runs 20 cycles per second. CPU-A requires NVS system to respond in time at the 50 msec timelimit. Getting the position data out of CPU-C and into a more powerful computer is clearly required.

Simulator manufacturers tend to split simulators from visual systems in order to give their customers options for chosing different visual systems. The clean interface between simulator and visual system is a key issue in this project. Link Miles NVS is one on the list of visual systems that do not meet todays requirements. Both FAA and JAA, the aviation authorities, accept the restricted visual systems but have issued guidelines for keeping scenery details low for the benefit of acceptable performance. [3]

Singer Link Miles DC-10 Level B Simulator at SVS Flylinjen school, Lillestrøm, Norway.

Original computer configuration.

Three |d|i|g|i|t|a|l| PDP 11/45 minicomputers.

Most to the right is CPU-C, extended and under test.

Up in the cockpit.

Three seats for captain, first officer and flight engineer.

Left side shows part of the Instructor Station.

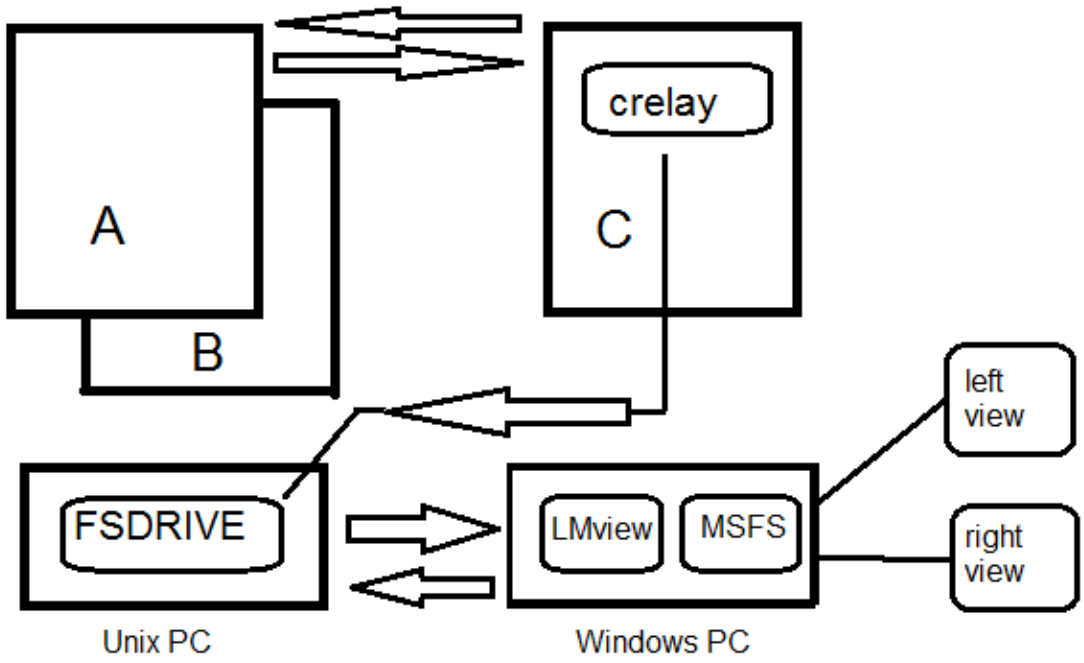
Outline of new DVS system architecture

Getting the raw data out of CPU-C into a platform that is powerful enough to process it for Microsoft FlightSimulator, is a first task. The architecture in mind is:

1. **Relay**. Add an ethernet card to CPU-C, run one of the later RT-11 operating system with a TCP/IP stack and send the data over the network to a Unix machine.
2. **Convert.** On a Unix machine write the programs to process the data and make a connection to the Windows PC with Microsoft FlightSimulator.
3. **Display.** On the Windows PC write the interface program to set FlightSimulator to the right position.

At first it was not decided if, after experimental phase, the programs of (2) and (3) could be integrated in one single program in (3). We already had a Unix PC running at the DC-10 to connect all console serial lines to the Internet for remote work. Unix has a full C programming environment and as (2) is an intermediate it has the best opportunities for debugging and not disturbing any realtime constraints at MS-FS by extra load.

DVS system, as currently in use, has several components outlined in the picture below:



- CPU-A & B run the Simulator and CPU-A sends position data to CPU-C
- CRELAY receives the position data and sends it to the Unix-PC via network
- FSDRIVE receives the position data, converts it and sends it to the Windows PC.
- LMview receives the converted position data and sets the aircraft in MSFS accordingly.

Getting data out of CPU-C

We already had positive experience with Alan Baldwin's FTP for Small Systems. We use it for filetransfer and transferring diskimages on CPU-A. FTP for Small Systems runs under one of the lastest RT-11 v5.4 SJ versions. The DC-10 Simulator software uses customised RT-11 v2, an early version of this basic operating system not capable for running TCP/IP..

As I work about 1000 km from the Norwegian simulator, testing real-time applications requires me to have the Sim overhere... anyway, something had to be invented. An old PDP 11/10 with 16 KB core memory served as a Simulator-Simulator with a small program to send out 32 words of dummy data at the rate of 20 times a second. Using same hardware (DR11-C head-to-head with another DR11-C) coupled to an PDP 11/44 the same configuration was available as in Norway.

The FTP package for Small Systems is for PDPs that have max 48KB of memory. The package does not feature a programming interface for TCP/IP as the machine is very small. So, I turned to using another version of TCP/IP for RT11 v5.3 XM: run on bigger machines using max 4MB of memory. Here you load the TCP/IP stack as a background task and use it from application programs in the foreground. This implementation has a neat programming interface implemented in a library of routines.
The experiments to use this TCP/IP implementation, to collect the packet of data and send it to a Unix machine failed. The first two packets were received properly, but the programs crashed. No doubt the enormous stream of interrupts (> 650/sec) were killing.

Timeline: April 2010.
A program (called "iwhere") written in C programming language to run on CPU-C (are you still with me?) prints position data on the screen for every 80 packets. Now we can see position data, but it is not yet out of CPU-C!

Desperately, I dropped the TCP/IP solution and decided to keep things as simple as possible and write all software myself:

- send position data packets as UDP broadcasts, hence no acknowledgements, no flow control, no handshakes or other interaction.
- No ARP address resolution (broadcast has, known, fixed MAC address)
- pretty straightforward: get 32 words, assemble the UDP/IP packet and kick the Ethernet card.

You don't win beauty contests with this kind of programming, but we needed it to work. So "CRELAY" was born and does its work since that day. One thing it changed to the datapackets: it uses a free, unused word to store the PSN, Packet Serial Number. We do want to know if packets get lost on the way.

Timeline: early August 2010.
Crelay works on CPU-C. The "ppos" program running on Unix printed the position info and made a capture. The flight crew of Jan Fjeld, Vidar Eggen and Magnus Wennevold flew a Houston Circuit blindly using instruments and data was captured.

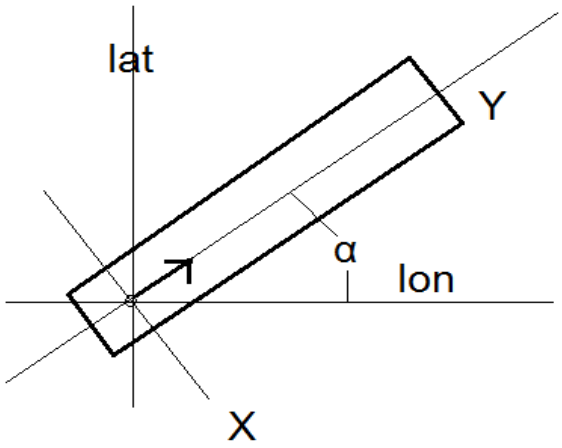
Mission accomplished: data finally got out of CPU-C!

First steps in converting data and driving MSFS

The program under development is called FSDRIVE and it reads the UDP broadcasts and converts it to a format suitable for MSFS. I studied the capabilities of the LUA programming interface that comes with FSUIPC by Peter Dowson [4]. It suited our needs and one of the demo programs has been used as a starting point for our application.

LMview, as the program is called, accepts position info as a string with values for lat, lon, pitch, heading and roll and sets the aircraft accordingly. After experimenting we set MSFS in *slew mode*, hence disabling the flight engine of the aircraft we use in MSFS.

The program “replay” was written to read a captured file and send the broadcasts at correct intervals onto the network. As mentioned before, I'm about 1000 km from the Norwegian Sim and realtime testing cannot be done over long distance with network delays on the Internet.
Conversion of runway coordinates to Longitude/Latitude as required by the LUA program “LMview”, needs some help from sources at the Internet. Mathematics like those involved when calculating distances on the earth-globe are non-trivial. E.g. to convert X and Y in first quadrant is shown in picture below:



$$\text{Latitude} = \text{Latitude}_{centre} - \cos(\alpha) * X + \sin(\alpha) * Y$$

$$\text{Longitude} = \text{Longitude}_{centre} + \cos(\alpha) * Y + \sin(\alpha) * X$$

Where α is recalculated from runway heading and X and Y converted from distance in feet to fractions of the Longitude and Latitude. Note: one degree in Longitude is static, one degree in Latitude isn't (largest at equator and smallest at the poles). Here Haversine algorithms have been applied, taken from the Internet. Want to know more? See [5]

Timeline: August, 23 2010
Replay of Houston flight visible on MSFS in my workshop. First edition, still with bad performance on ground, installed in Norway one day later. [6]

This milestone was a fact and a lot of work was ahead: tuning, refinements based on the experiences during replaying captured flights overhere and testflights flown in Norway.

Refinements, lessons learned and spin-off since August 2010

With the Norwegian crew flying the DC-10 at least once a week and me having time every now and then we nevertheless made steady progress since the August 2010 milestone.

Before going through the events, I must mention the spin-off project called “Partition 11”, that is the effort to add new airports to the DC-10 Simulator on the #11 partition of the system (ZIP) disk. This effort is about getting the required data of runways, beacons, navaids into the old style compact datatables for CPU-A. Clearly, Daylight Vision System challenges us to fly from A to B! After Oslo Gardermoen, other airports followed: Bergen, Stavanger, Trondheim, Kobenhavn, Goteborg, Stockholm and Amsterdam Schiphol.

(I) Ground performance

The shaky performance on the ground improved by not using *slew mode* on ground in MSFS. LMview, the LUA program that we use to set aircraft position in MSFS, was changed to take aircraft lights and slew mode as extra in each input record. FSDRIVE can now set slew mode automatically when passing 300ft AGL. This option is barely used after the real fix of the ground performance issue. Setting slew mode in MSFS disables the flight engine of the aircraft you use in MSFS: you only set position.

Movements of 1/20th foot converted into Longitude and Latitude are significant in > 7th decimal. After changing FSDRIVE to use double precision (64 bit) floating point for its calculations, ground performance became smooth.

(II) FSDRIVE rebuilt from synchronous to asynchronous

Classic straightforward C programs wait when reading input from network: program continues when data is available. This synchronous scenario cycles through the following steps:

1. Read/wait for input from CPU-C
2. Convert the data
3. Send the data to the PC with MSFS
4. Read/wait for the acknowledge from PC with MSFS

It was noticed when MSFS is busy (e.g. loading scenery) it blocks FSDRIVE and you get a disturbed view: a little hold followed by a quick jump forward. Also on take-off at high speed, the view got lagged behind realtime.
We need asynchronous setup here. The key for this is the great *select* systemcall in Unix: we can detect which I/O channel has data for us without blocking the program. Used on input channels only, it kind of signals us what work need to be done:

1. setup the *select* system-call, wait until anything happens.
   You get out of the systemcall when:
   - there is input data from MSFS, or
   - there is input data from CPU-C, or
   - there is input data from both MSFS and CPU-C
2. when data from MSFS (=acknowledge from previous data) mark it “ready”
3. when data is available from CPU-C
   - read it
   - convert it

- check MSFS is marked “ready”
    - If so, send the data to MSFS and mark it “busy”.
    - If not, drop the data.

Whenever MSFS is busy it will not be overloaded by a backlog of data. You may see a glitch in visual, but the next datapacket is back on track: as realtime as possible.

(III) Loading scenery: pause on aircraft holding

When you start the DC-10 Simulator runway 24R at Los Angelos is your starting point. When you choose e.g. Oslo as initial position it starts to move your aircraft to Oslo in quick steps across the globe. Obviously, MSFS loads scenery for these steps and gets overloaded. A change is made to FSDRIVE to detect your aircraft is holding and to stop sending packets to MSFS exept sending one packet every 10 seconds. This proved suitable to get scenery loaded quickly.

(IV) LMview extensions

Two commands were built into LMview: one to retrieve MSFS's idea of the current position of the aircraft. This serves calibrating Initial Positions e.g. adjusting an aircraft at a jetway. Another command reports the MSFS version to make it possible to treat different versions of MS Flightsimulators differently.

(V) Fly beyond visual limit of 86Nm

To fly outside visual area, FSDRIVE keeps track of overflows (wrap-arounds) in the 24 bit numbers for X and Y. Extended_XY mode accounts for the wrap-arounds and calculates Lon/Lat accordingly. For instance, first wrap-around in Y at positive side, adds 86NM and accounts for the negative Y we now read. After passing through 0 it becomes a positive value and a second wrap-around adds 172NM, making the total offset 258NM.

Timeline January 2011:

testflight Paris CdG to Oslo Gardermoen.

To test the Extended_XY option, flight crew of Vidar Eggen and Magnus Wennevold flew Paris to Oslo. It was the first flight into Oslo [6] *and nearest starting point was Paris. In the second hour of flight a significant error in Longitude was noticed and INS read-outs done by Jan Fjeld were used to plot the real position and direct the flight crew into RW01R at Gardermoen. Visually, the aircraft flew over Danmark into Oslo Fjord, real Lat/Lon showed positions more to the west. When the Simulator switched to Oslo visual at ±45NM distance, view matched with reality.*

So, Extended_XY worked insufficiently over long distance!

(VI) Plan B for flying long distance

As position data contained sufficient unused space, a six-line change in DC-10 Sim software was made to add aircraft position in Latitude and Longitude to the position data.

Now we can chose what's best: FSDRIVE can work in Plan A mode (runway coordinates) or Plan B mode (aircraft Lon/Lat). Plan B turns out not accurate enough to make ILS landings.

```
; ---------- DVS PLAN B MODIFICATION ----------
MOV    NBLTAC,      48.(R5)     ; AIRCRAFT LAT POS
MOV    NBLTAC+2,    50.(R5)     ; 2ND WORD
MOV    NBLTAC+4,    52.(R5)     ; 3RD WORD
MOV    NBLNAC,      54.(R5)     ; AIRCRAFT LON POS
MOV    NBLNAC+2,    56.(R5)     ; 2ND WORD
MOV    NBLNAC+4,    58.(R5)     ; 3RD WORD
```

You need to use Plan A for landings, otherwise you are off runway when flying on instruments. (picture showing this effect below)

(picture by Magnus Wennevold)

On “visual” runway 17 at Bergen Flesland in plan B mode: the runway as “seen” by instruments is on the righthand side.

After several experiments, the final scheme used is: switch to B mode when climbing through 2000 ft AGL; return to Plan A when descending through FL100, or lower 4000 and 2000 when flying circuits. On switching modes we use 10 second smoothing algorithm to gradually smooth the difference in position, which is in the order of 1000 feet on long flights.

Timeline: April 2011.
<u>The proof of the pudding is eating.</u>
Magnus Wennevold, Jan Fjeld and Bjørn Kulterstad flew Paris to Oslo in the mixed Plan A/B implementation. Positions were accurate and only a small bump in visual was noticed when switching to Oslo visual.

(VII) Smoothing altitude (Z)
Ground level in the DC-10 Simulator is 15 feet AGL, assumed to be pilots eye-level. All airports defined for FSDRIVE have MSL of a sample aircraft used in MSFS. On hard landings and late take-offs the DC-10 Sim shows Z values of around 9 feet, smashing the aircaft to the ground in MSFS, while maintaining pitch in position data. FSDRIVE smooths the Z values below Z=15.

As Z data is always defined as AGL with current airport selected, it turns out that it is silently converted to MSL on high altitude. (Example on next page)

When getting close to your destination airport (±45 NM distance) the Link Miles simulator switches to the visual of that airport/runway and MSL to AGL is accounted for.
Unfortunately this happens moments before the actual switching of visual orientation and cannot be handled neatly at the same moment.
For example when approaching Oslo descending from high altitude, 680 foot is subtracted moments (0.5 seconds) before switching to Oslo visual. At the moment of switching FSDRIVE accounts for Oslo elevation and adds 680 feet.
A 50 second smoothing algorithm working on the calculated MSL (based on Z) has been implemented to avoid a bump in visual. It supresses the effect of going down 680 feet and jumping up 680 feet shortly after that.

Facts and figures

The version of MS FlightSimulator used is FS9, FS2004. FSX can be used as well, although it requires different tuning of runways and aircraft gate positions.

The LMview program, written in LUA is about 130 lines and most of it is similar to the client demo program.

CRELAY, is witten in C programming language with some patches in assembler. It is 907 lines of code in total.

FSDRIVE and supporting modules are all written in C, 1845 lines of code some of it disabled after being used in experiments.

Change of 6 lines in LMEXEC, the tailormade scheduler for the Simulator.

Hops in altitude during Paris to Oslo flight. Note every record represents 1/20th of a second.

```
14521 Z=29167.875 d=387.125
14537 Z=29077.562 d=-93.812
14585 Z=29001.438 d=-95.562
14665 Z=29135.812 d=96.500
14681 Z=29048.625 d=-95.438
14697 Z=28961.688 d=-95.375
...
26537 Z=32348.938 d=288.375
26568 area changed to: Amsterdam
...
52249 Z=21382.500 d=-680.938
52257 area changed to: Oslo Gardermoen
```

Elevation at Paris is 407 ft, Amsterdam 6 ft and Oslo 698 ft all including 15 ft extra for eye level. See text at previous page.

When switching visual it is interesting to see the difference between Plan A and B calculated positions:

Snapshot #1: a few minutes before encountering Oslo visual:

- Plan A position is N58.834799905859 E010.465865789265 calculated with X extended -1 times and Y extended -2 times with the threshold of RW18C at Amsterdam as reference.
- The Plan B position at that moment is N58.856427300255 E009.121486353627
- The distance between the two points is 77 km bearing about 270 degrees west.
- Note that we still fly in B mode, the west position.

Snapshot #2: shortly after visual switched to Oslo, still on descent above FL200:

- Plan A position is: N59.517515879511 E009.887259433111 from Oslo RW01R
- Plan B position is: N59.519500565668 E009.872246730042
- That makes 880 meter to the west, about 280 degrees.
- Note that we still fly in B mode, the west position.
- Transition to A mode happened later, descending through FL100.

Conclusion

Using runway coordinates for visual, as designed, is required for precision landings flying on instruments. By using mixed A/B mode we successfully pushed the old simulator beyond its limits. Discussions about the need of 40 FPS visual have been ongoing for months, but all agreed it works great with 20 frames per second.

So, at the end of this story enjoy the grand tour [8].

REFERENCES

1. Rolf, G., “Work in Progress: the Norwegian DC-10” published at toomuchfs.com http://www.toomuchfs.com/2mfs/showdoc.php?dsn=493

2. Rolf, G.,“En Simulator i fem hundre Kilobyte! (0,0005 GB)”, lecture slides FlightSim Lan Gardermoen 2009 (FLG-09), October 10, 2009. http://geerol.home.xs4all.nl/filestore/FLG-talk-75dpi.pdf

3. JAA, Leaflet No 13, Old visual systems and new visual scenes for FSTDs http://www.easa.eu.int/certification/flight-standards/doc/oeb-supporting-documents/tgl/AGM%20S6%20STD%20TGL%2013%20Old%20Vis%20Syst%20New%20Vis%20Scenes%20Feb%2008%20Print.pdf

4. Dowson, Peter, FSUIPC API for interfacing to Microsoft Flight Simulator. http://www.schiratti.com/dowson.html

5. Veness, Chris: Calculate distance, bearing and more between Latitude/Longitude points. http://www.movable-type.co.uk/scripts/latlong.html

6. Movie made using Fraps of landing at Houston RW8 from captured circuit (1st visual) http://geerol.home.xs4all.nl/filestore/DVS1-KIAH-ilsapp-RW08.wmv

7. First landing of the DC-10 at Oslo lufthavn Gardermoen – many to follow. http://www.youtube.com/watch?v=fuYlvF2Nfo0

8. The grand tour around the Norwegian DC-10 Simulator: movie by Vidar Eggen. http://www.youtube.com/watch?v=Y-DpcvY4aBk

1 2 3 4

5 6 7 8