IAC-14-B6.2.5

To the Interface and Beyond, Putting the "mmm" back into MMI

Mr R Sachasiri,
Geo-Informatics & Space Technology Development Agency (GISTDA), Thailand ravit@gistda.or.th

Mr Andy Armitage
Terma B.V., The Netherlands, aba@terma.com

Ms P Techavi, Mr J Plaidoung, Mr A Witts
Geo-Informatics & Space Technology Development Agency (GISTDA), Thailand,
ptechavi@eoc.gistda.or.th , jayranon@eoc.gistda.or.th, a.witts@yahoo.com

Whatever we may see in the movies, spacecraft operations are not supposed to be exciting. To some, this means that user interaction with a spacecraft control system must necessarily be difficult, intricate and tedious. We completely disagree. Smooth, automated operation is not at all equivalent to boring, or completely non-interactive. In our everyday lives we interact with technology that not only works smoothly but has also been designed with aesthetic appeal in mind. This ensures that the user experience starts and remains satisfying. It is only because of a lack of competition and commercial imperative that this trend has scarcely touched the spacecraft operations business. In the past few years we have invested to develop an underlying telemetry and commanding engine, that gives access to spacecraft monitoring information at the highest levels of performance. Interrelated information can be retrieved and presented to the user immediately on demand. Our customers and partners have invested to customize user interfaces for their own missions, to a level that begins to approach what see in the movies. The technology we used was QML (Qt Meta Language) and we would like to show a flavour of what QML makes possible for spacecraft operations. This brings an interesting possibility: to wrap a monitoring and control system (the "engine") under a fully mission-customized user interface (the "skin") with an appearance and behaviour exactly as the end user or customer desires.

INTRODUCTION & BACKGROUND

This paper describes the results of an ongoing collaboration between Terma B.V. of the Netherlands and the Geo-Informatics and Space Technology Development Agency (GISTDA) of Thailand.

The project known as the Versatile Operating System for Spacecraft Control & Administration (VOSSCA) aimed to perform a thorough modernisation of the existing ground control facilities at the Sri Racha ground station, with a view to supporting future missions as well as the currently operating Earth Observation satellite THEOS1.

The goals of this project included a complete re-invention of the user interfaces to fit with the GISTDA future requirements and vision. It also aimed to enable and further develop the local Thai capabilities in this technology domain.

GOALS & MOTIVATIONS

At the very start of the VOSSCA project, it was clear that the customer and users wanted a modern system to "look cool". The immediate questions were:

(1) What do you mean by "cool"?
(2) What are the real goals apart from appearance?

These questions were immediately answered by reference to science fiction movies. We should absolutely not dismiss these responses, because sci-fi movies are an invaluable source of inspiration for engineers.

We do however have to examine them seriously and critically. Different movies devote different resources and analysis to the "realism" of what they are trying to show; a thoughtful, slow-paced story might include shots that linger on how a space ship is operated, whereas an action movie maintains its pace by moving on very quickly. Both serve their purpose.

We show some examples below:

Fig I: The genius engineer speaks to his self-designed home computer to call up a holographic wall of information, which he physically rearranges to arrive at the design of a robotic suit. The meaning of the display is not exactly clear to us, but that is irrelevant because in the story, he designed it for himself , not for us.

Fig II: During a lengthy interplanetary mission, a crew member interacts with an autonomous intelligent on-board computer assistant. Technology malfunction is important to the story-line, so we must believe the system is infallible. Great care and attention has been spent on effects that are credible even on close examination

Fig III: An operator sits at a touch-enabled multimedia mission console which supports her to plan the targets and deployment of deadly robotic drones.

Fig IV: The interface supports audio and video communication with colleagues, maps, list of available resources, localised weather, time of sunrise and sunset.

The ultimate goal of a movie is clear: to move the story forwards in a short time, while engaging the audience. Very often when we look closely at stills from a movie, we see that the illusion would not last more than a few seconds. But we also see inspiration and imagination. A real, usable mission control system interface has to be used continuously from day to day, for a useful purpose.

Whatever our reaction to these scenes, when we examine them closely, we see that the amazing visuals are developed with the best available care, talent, imagination and budget for one goal: to support the story line. In the same way we have to approach real-life interface development with focus on the key question: "How will we help the operator do his or her job?"

PUBLIC ENGAGEMENT

To generate public and political support even the most conservative of agencies understand that looking cool is something they have to consider. The genuine excitement of their missions has to be conveyed to a public expecting the latest advanced technology. They invest heavily in their web and other media presence. The reality does not always match the appearance, as illustrated below:

Fig V: ESOC Main Control Room. Despite appearances, the wall displays are mainly static illuminated posters, and the touch enabled tablet devices mounted on the operator consoles are only used for diary, contacts and email. The actual spacecraft control user interface is antiquated

OVERCOMING OBSTACLES

We could ask why so many mission control systems in use today seem so old-fashioned? It has become especially obvious in recent years, because of the appearance of smart phones and tablet devices. There are no simple reasons but it is worthwhile to understand them, because they apply in a much wider context to the space industry.

Market Size

For mobile devices, we see products aimed at an audience of tens or hundreds of millions. The product interface is also a platform for additional information products; apps, games, music, movies, and advertising. The sensors of the device such as self-location are used by open-ended applications that can generate additional revenue for the developers even if the end user pays little or nothing.

By contrast the spacecraft ground control system is typically a one-off relatively high value purchase made by one of a few hundred institutions around the world, each of whom employ a small engineering and operations team. While there is a demand for add-ons, and extensions, this is not on the same scale, and the end user would like to avoid costs by implementing some of them himself.

Political Environment

Historically, complete space systems have been developed with a very high input from Government Agencies, both in funding and technical control. While such support is an excellent way to incubate a fledgling industry, there comes a stage of maturity when both taxpayers and industry would be better served by a market- and demand- oriented approach.

Developments should be initiated by companies at their own risk to steal advantage over their competitors. Industry- initiated projects have a higher impetus for return on investment, and because of this pressure, usually have a much faster time to market.

It is increasingly recognised, especially by newcomers to the space business, that a transition from government-led to industry-led approach is inevitable.

Engineering Conservatism

The government-funded approach tends, over a long period, to foster a highly conservative, compliance-based, risk-averse engineering culture.

Engineers are also sometimes guilty of disguising complacency as technical judgement, for example by saying "user interface is not important if the underlying system is right". On the contrary, a poor user interface can render a good system unusable.

A very typical development approach is to defer user interfaces until the end of a project, while focussing on kernel performance and functionality. This is risky; it can be discovered far too late that getting information out of the system, and visible to the user, is somehow too expensive!

Prediction and Priorities

Fortunately, we can break out of the limited-market, government funding traps; technology has improved to the point where we do not need vast development budgets. The possibilities evolve over time, so a challenging development today could be perfectly feasible one year from now. Planning to defer a feature based on solid predictions that it will be much easier in the near future is not lazy or complacent, it is perfectly justifiable. In the meantime we can spend resources on something else.

All organisations have limited resources and have to prioritise to where they are most needed, what is a worthwhile development today, and what has a chance to generate a return on investment in reasonable time.

THE VISION

The GISTDA and Terma vision used inspiration from science fiction movies, long heritage in flight operations and software development. We took a completely fresh look at technologies available today at reasonable cost, and evaluated how that could be integrated with our modern flight operations kernel.

In this we found that being relatively small and agile organisations was a distinct advantage.

TECHNOLOGY CHOICE: QT & QML

Our main technology choice was "Qt Meta Language" (QML). This choice was an evolution from our previous choice of Qt for our underlying spacecraft monitoring kernel, CCS5. Qt and QML are obviously easy to integrate in the same software.

Qt was previously chosen by Terma B.V. for a variety of reasons, mainly: portability and performance. We also had a preference to rely on only one main external library with clear support and license conditions.

QML

QML is a language supported by the "Qt Quick" module of Qt designed to support scripted user interfaces in a modern smoothly-animated touch-enabled style. It is portable to embedded devices as well as Windows, Linux and MacOS and recently extended to iOS and Android. High graphical performance is assured by use of OpenGL in ES or desktop editions, and it is allowed to embed OpenGL shaders directly in a QML script.

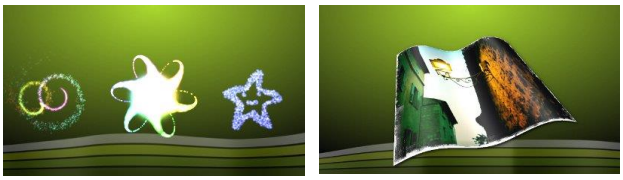

Fig VI: QML particles and shader demos

The QML language is descriptive rather than imperative; we encode what we want to see. This creates a clean separation between the description of what we want to see, and the way underlying data is acquired and processed.

In QML we bind the state of the user interface to data values that may change. If the bound data changes, the user interface is automatically updated, with transitions if desired. Ideally, a QML user interface can be written without any imperative code. It is allowed to incorporate call-back functions in JavaScript syntax, but the files are definitively not JavaScript.

The QML classes include layout classes for automatically arranging data across the available space, this includes grids and arbitrary geometric paths.

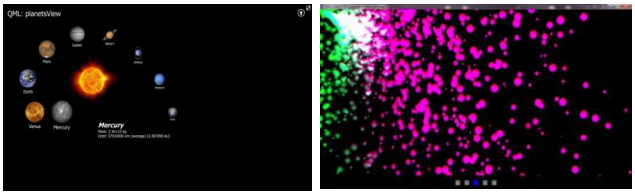

Fig VII: A simple path demo script showing the planets and a scripted shader demonstration

QML Plugins

QML has a number of interesting add-on plugins, for example, WebKit allows embedded web content, Qt Charts supports simple charts, there is support for mapping XML structures to visual objects, and if desired, a set of standard UI components that look like conventional user interface objects.

The Qt3D (experimental) plugin allows simple animated presentation of 3D objects. This is interesting for space applications because it allows the assignment of a 3D model ”mesh” to a QML object and then to animate using bindings. Obviously these can include simple representations of the spacecraft or subsystem structures. We found it especially interesting because we did not have to make major investments in 3D modelling: it was possible to download or export 3D models from the free version of Sketchup.

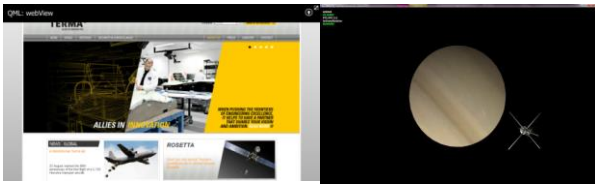

Fig VIII: An embedded web viewer using Qt WebKit, and a trivial Qt3D animation from telemetry

Note: Qt3D should not be considered equivalent to sophisticated 3D modelling provided by game engines, and in the version we prototyped it still has some minor issues (see “Alternatives to QML” and “Lessons Learned” below).

System Integration

To animate QML user interfaces with spacecraft related data, we had to implement custom classes and expose them to QML. These equate to the classes in the underlying control system, such as telemetry parameters, or mission database, or archive database objects. These classes had to be developed and documented as QML extensions to the underlying kernel. One of the main advantages it that the information being animated is directly held in memory in Qt objects, and the Qt Quick integration allows bindings to be set up with the properties of arbitrary Qt objects.

Alternatives to QML

Game engines were considered, either instead of or complementary to QML. The game engines considered were Unity3D and Unreal, with Scaleform as 3D development tool. Not surprisingly they were vastly superior for development, modelling and animation of full 3D environments, however the method of data exchange with the underlying kernel would have required heavy investment or clumsy integration. If game engines will be considered in future it is more likely to be for public exhibition type of displays.

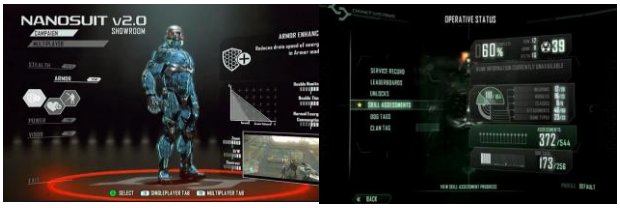

Fig IX: User interfaces using Scaleform with Unity3D

Windows Presentation Foundation (WPF) was also considered, but rejected partly because it is completely Windows oriented, whereas QML is portable to other platforms. The VOSSCA development proceeded with QML.

ANALYSIS APPROACH

The project started with a systematic User Experience (UX) approach. This takes human behaviour, organisational hierarchies, relationships and technology factors into account to compose the best User Experience.

It became evident that the system interface was not only relevant to the user of the system but also to visitors, as the system serves as a tool for outreach. The VOSSCA goals were set to include:

(1) Include both 2D and 3D synoptic displays
(2) Allow commanding from a Mission Control Dashboard
(3) Interface to pass scheduling to include both activity as well as telecommand management
(4) Operator schedule monthly and daily schedule management with notification capabilities.
(5) Report and data visualization module for analysis of satellite system data trends.
(6) Incorporate one or other novel user interface device

Storyboard Composition

Sequences of user actions can be analysed and converted into a “story” which summarises the sequence in a set of rough pictures, and allows it to be

discussed, analysed and modified before it is implemented. This is obviously an approach inherited from movie production.

The VOSSCA project was preceded by an Operational Process Study (OPS), which performed some simple story-boarding, an excerpt of which is shown in the following figure:

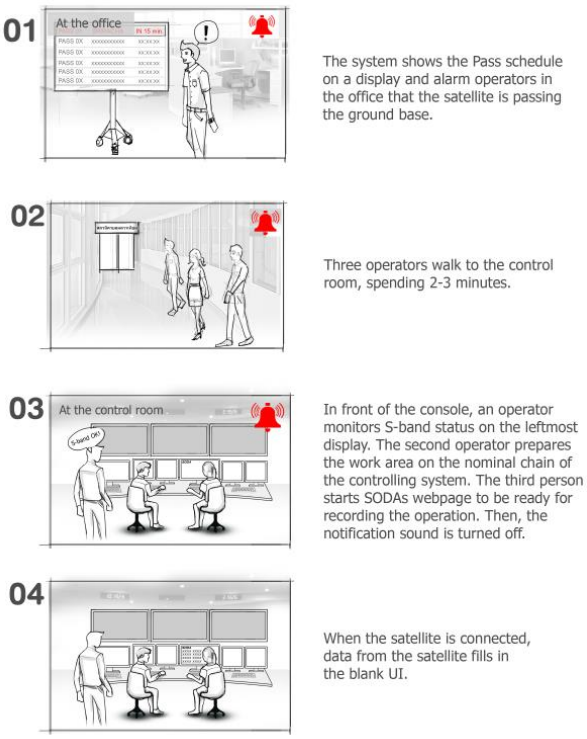

Fig X: Story-board showing the start of a satellite pass

The suitability of this approach depends on the cost multiplier between detailed design and implementation. Producing a detailed storyboard for a several scenarios can be expensive. If the technology used and the developers are able to rapid prototype different scenarios on a real device, then it is more cost effective to develop a number of user interface prototypes, select the best ones, and then improve them in a series of iterations. The QML scripting language is amenable to this iterative prototyping approach.

User Interviews versus Observation

Listening to, and collecting user input and opinions about their typical tasks is an important aspect of the analysis. However it is important to be aware of the potential pitfalls of this approach! These stem from the fact that the operators are only human.

Quite often a verbal or written description of how operators use a system does not match what they actually do! They do not intend to mislead, but they are sometimes only relating whatever is foremost on their mind at the time of the interview; a problem they had specifically that morning for example. This can result in distorted priorities.

Some operators are extremely focussed on a specific irritation of the system they currently use. For example "I would like to increase the size of the font in this log window". While it is useful to collect this kind of input, this is often not specifically useful or relevant for the new system, for example if the window does not exist in the new system at all.

We must also be aware that not all spacecraft operators are natural innovators. They may be able to relate accurately what they do. They may also lack insight or imagination as to how they could do it better. Recall that the best user interfaces were not designed by asking a focus group how they would like their systems to be in future.

Simple observation and recording, with the occasional question, while a user goes about their normal job may be the best way to collect accurate information about the true user interaction sequence.

Direct Measurement

It is also possible to directly measure user interactions with eye-tracking, that measures which parts of the screen the user pays most attention to. This has the advantage of eliminating bias or prejudice of the developer from measuring the effectiveness of the interface. This approach is often used to measure the effectiveness and usage patterns of major web sites, especially ones that contain advertising. It will feature increasingly on tablets and smart phones, but due to the small scale of the project, has not been used in the VOSSCA project so far.

IMPLEMENTATION

We needed to separate our implementation into two different types of display, standard and mission-specific.

Standard Parameter Displays

With standard displays we wanted to provide a common way to view sets of data such as telemetry parameters that would look and feel familiar in all missions where our system was used. We also wanted to make sure our new system supported at least the complete equivalent of what was available in historical systems.

In effect we wanted to give the system a new "house style". If the end user does not have the time or resources to develop their own mission-specific QML, a pleasant and modern-looking default interface had to be available, that could do a little more than the traditional displays could.

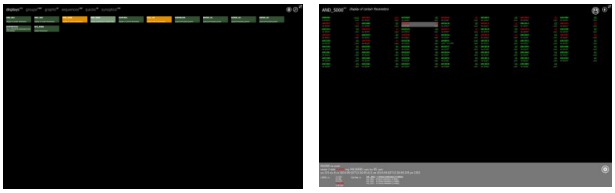

Fig XI: Top level and detailed views of telemetry parameter status, with focus on parameters in alarm

These generic displays are modelled roughly after the functionality and some aspects of the appearance of historical systems, but using the features available in a modern library as well as dynamism. The look-and-feel was based loosely on Windows Phone and "Metro" themes.

The generic views present telemetry in a mission-neutral way, so simply lay out rows of grouped parameters in summary or detail. A tabbed window can be maximized to fill up the current screen, and the user can tap or can swipe horizontally between different tabs. There are no scroll-bars as such, instead the user swipes the view. To focus on a particular group the user taps on the display, and it flips over to show more detail. Importantly the system overview summarises any alarms at a lower level, so the user is guided in advance to a subsystem where there is a potential problem.

Unlike previous systems we support dynamic on-the-fly definition of new displays, the user merely has to drag & drop a parameter or group of parameters from another display to create a new display. If saved, these new displays become part of the user's own personal display list.

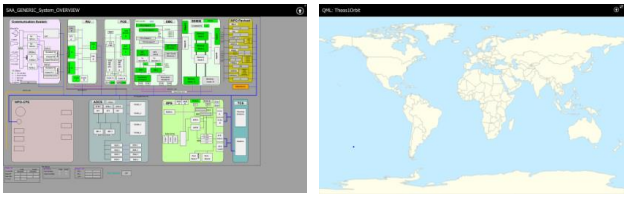

Fig XII: Animated 2D schematics, a synoptic picture display and SVG map showing orbit track

The system also integrates the traditional synoptic picture animation system, so it is possible to click on different 2D schematics showing animated status in the traditional way. The new system improves on the traditional one by supporting completely dynamic scaling using pinches and touch interaction to tap between different pictures.

Mission Custom Displays

Mission custom displays are scripts developed for a specific mission; for example this might include images of a spacecraft itself or a dedicated layout of animated telemetry parameters for the specific mission.

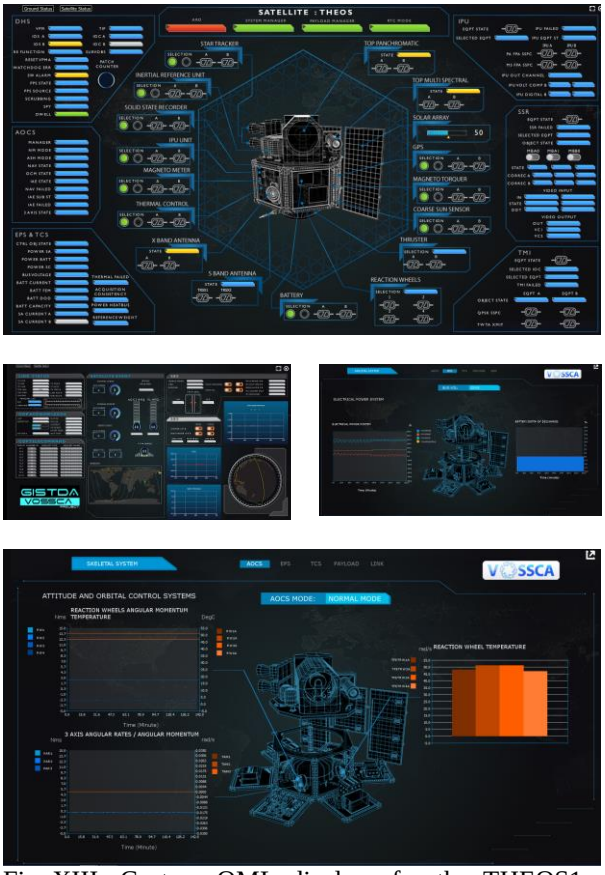

Fig XIII: Custom QML displays for the THEOS1 satellite

These custom displays can be called up from the standard display or, if desired, can be used to completely replace the standard displays. In this case however the mission is completely free to implement their own style.

NOVEL USER INPUT METHODS

The use of innovative input methods is part of the GISTDA vision for VOSSCA, and so we explored several options outlined here.

Domain-Specific Safety Concerns

Safety, criticality and protection of a satellite are paramount in space operations, so it is especially important to prevent mistaken or inaccurate user input from causing damage to a spacecraft. The potential limitations of each type of device were considered. While motion sensing devices may not be accurate enough for direct spacecraft operations, the possibilities for contact-free user interaction may be well-suited for public displays.

Collecting user input is a compromise between accuracy, expressiveness and leaving the user free of encumbrances or extra equipment, to express himself precisely in the most natural way.

Touch Screen versus Mouse

Our most obvious input change has been to allow touch-enabled user input. Note that keyboard and mouse should not be too-quickly dismissed as old-fashioned! A mouse allows us to control user interfaces accurately and with buttons, allow different means of expression.

We tested using first generation Windows Surface Pro 2 tablet PC and using conventional Dell desktop PC's with 27" Dell touch-screen monitors. In the latter case the touch input is detected via a USB connector. Both configurations used Windows 8.1.This demonstrated quite well the kind of issues that can be seen when transferring between devices with different physical sizes and different resolutions.

Screen Resolution

On-screen virtual buttons, where the user can tap by fingertip, must be big enough to be sure that the correct button has been clicked or tapped. There are many variables - not only the device input and output resolution, but also the size of the fingertip.

We can also fall into unforeseen traps when developing for different touch screens. A mouse moves on a different surface with a different scale for the mouse motion. With a touch screen, the hand must physically move to the point of input and then tap. If we design a user interface with a button at the top right, very easy to reach on a small tablet screen, on a 40" touch-enabled surface, the user has to lean an uncomfortable distance to hit the same button.

While it is possible to write device and resolution-independent QML it is very easy to forget, and costs some additional effort; these issues can only be fully rooted out by testing on the final target device.

Mobile Device as Remote Controller

One solution can be to use a tablet as a local control device for one or more wall-mounted displays. In the simplest approach this can be set up by simply connecting a wall-mounted monitor to the tablet HD connector, and working in duplicated screen mode. Otherwise more sophisticated desktop sharing software is required.

Fig XIV: Tablet as input device

Motion Detection

Two types of motion detection were investigated Kinect and Leap Motion. Both are available at very moderate cost.

Fig XV: Kinect (left) and Leap Motion (right)

Kinect

This is a full-body motion sensor originally developed as a game controller for XBOX. The user is expect to be at a distance. It can be used in near mode for half-body detection at 70cm to 1m, and 2-3m for full body motion detection. To use Kinect at a suitable distance from the screen, all objects on the screen must have sufficient size to be visible; this means that such a display has to be explicitly designed for this kind of use. For these reasons, Kinect may be suited for public display but is not suited for direct use by a mission control operator.

Leap Motion

This system was invented for air-gesture interaction with personal computers while sitting in front of the screen. The device is set up just in front of the PC. Leap is in theory able to detect relatively fine gestures and individual fingers. The active range

was about 1 cubic foot around the device. It can detect some virtual gestures: circle, swipe, and tap

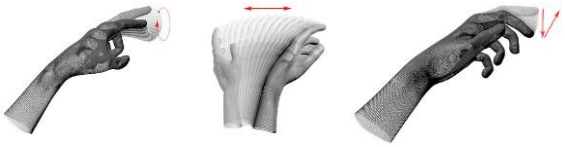

Fig XVI: Leap Motion Gesture Detection

Leap Motion comes with software to allow the device to emulate a computer mouse, but this does not support complex gestures. There are some early-stage developments to enable the full range of gestures to be detected in QML, for example the library QLeap.

In our experiments we found that Leap Motion seemed (at the time) to be an immature product. It was difficult to avoid inadvertent gestures from being picked up, and some intended gestures were were not interpreted accurately. Again, we encountered some unexpected issues: if the user leaned over the device, perhaps to look closely at a screen, parts of the face or head were detected!

We suspect that Leap Motion has matured somewhat as a product since our testing one year ago, and we may examine it again in future.

Speech & Voice Control

For the sake of completeness we mention the possibility of control using spoken commands. The goal would be to teach the system to recognise a limited number of voice commands, to check with the user whether the interpretation is correct, and then execute the command, perhaps by initiating a specific automation sequence.

We do not plan any explicit development, since this appears to be better supported by standard operating system tools such as Windows SAPI.

LESSONS LEARNED

The choice of QML seems to have been a good one, although it is not completely free of issues. Especially with the Qt3D plugin (which is explicitly stated to be experimental and subject to change) we have found issues for example the scene will only render correctly if it is the first QML file opened.

We have noticed that many issues are reported solved in Qt5.3 (we developed for Qt5.1) and so they may be solved by a simple upgrade. At the same time, Qt5.3 offers the possibility to deploy for Android and iOS, so there is a further impetus to upgrade.

It is all-too-easy to forget physical ergonomic issues when given access to a new display technology or novel input method. Also, if the development device differs even slightly from the deployment device, unforeseen issues often occur. It is also very easy to forget that humans come in different shapes and sizes as well as hardware.

It is absolutely essential for novel user interfaces to go through a cycle of development, trial with real users on the real deployment device, feedback followed by adjustments and improvements.

PROOF OF CONCEPT

It was decided in July 2014 to perform a live satellite verification test – to operate the THEOS-1 satellite in orbit from the VOSSCA system – both in terms of command and control.

During a pass of the satellite over our ground station the control was switched from the existing system to the VOSSCA system and the spacecraft was operated without error.

This proves that we have fulfilled the first objective, of creating a control system which can replace our existing system.

FURTHER DEVELOPMENTS

The VOSSCA project so far succeeded in the objectives to demonstrate impressive modernized graphical user interfaces, with touch screen interaction. The complete system was also able to demonstrate that it could take over control of the THEOS1 satellite.

In the longer term we must admit that the new QML implementation does not provide a fully immersive end-to-end user experience; for example all commanding is initiated via automated sequences instead of directly. This is an ongoing development that will need a number of iterations.

Qt5.3 has become available during the project, and this provides support for Android and iOS devices. Remote apps on these platforms could be explored.

Although the 3D display is workable, the Qt3D module is experimental and is not robust; this could be revisited using Qt5.3, or by introducing a game engine (with corresponding data exchange)

At the time of study, the Leap Motion product was somewhat immature and disappointing, we suspect that this has since improved, and should revisit it.

There has been some serious discussion about making a physical operator console similar to the one seen in Fig III.

Images are for educational purposes only, courtesy ESA, Time Warner, Universal Pictures, Marvel Studios, Leap Motion, Digia, Google, Excy.

None of these organizations are affiliated with Terma, nor does use of these images imply any endorsement

