Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

Exploration of collaborative environment technologies for maritime analysis

Étienne Vachon
Valérie Lavigne
DRDC Valcartier

Defence Research and Development Canada – Valcartier

Technical Memorandum
DRDC Valcartier TM 2011-233
April 2012

Canada

Exploration of collaborative environment technologies for maritime analysis

Étienne Vachon
Valérie Lavigne
DRDC Valcartier

Defence Research and Development Canada – Valcartier

Technical Memorandum
DRDC Valcartier TM 2011-233
April 2012

Abstract

The identification and tracking of Vessels of Interest (VOIs) require the collaboration of various analysts from the federal departments forming the Maritime Security Operations Centers, to collect and analyze information about this vessel, to understand its intentions and assess if it may represent a threat. In order to support collaborative work, the Intelligence Laboratory (ILAB) of the Intelligence and Information (I2) section was equipped with new advanced displays and interactions devices. For a period of eight months, a team worked to explore how these technologies could enhance collaborative analysis and shared situation awareness among analysts.

Existing multi-touch frameworks were explored and a multi-touch version of Google Earth was implemented. It can be used on a multi-touch table and with a wall display using gestures that are being captured by a Microsoft Kinect sensor. A maritime situation awareness portal including various VOI analysis tools mock-ups was developed and displayed on the knowledge wall. A simple application was also developed to display vessel information windows with various orientations. This exploration of advanced Human-Computer Interaction technologies produced insights on how to take advantage of these innovations in a maritime surveillance context as well as in other intelligence domain activities.

Résumé

L'identification et le suivi de navires d'intérêt demande la collaboration d'analystes variés provenant des départements fédéraux formant les Centre des opérations de la sûreté maritime afin de rassembler, interpréter, et présenter autant d'information que possible au sujet du navire, de comprendre ses intentions et d'évaluer s'il représente une menace. Afin de permettre le travail en collaboration, le laboratoire du renseignement (ILAB) de la section Renseignement et information (RI) a fait l'acquisition de nouveaux équipements d'affichage et d'interaction avancés. Pendant une période de huit mois, une équipe a exploré comment ces technologies pourraient améliorer le travail d'analyse collaboratif et le partage de la connaissance de la situation entre les analystes.

Les architectures logicielles existantes pour les applications multi-tactiles ont été explorées et une version multi-tactile de Google Earth a été implémentée. Elle peut être utilisée aussi bien sur une table multi-tactile qu'avec un affichage mural à l'aide de gestes captés par le senseur Kinect de Microsoft. Un portail d'éveil situationnel maritime incluant différentes maquettes d'outils d'analyse de navire d'intérêt a été développé afin d'être affiché sur le mur d'écrans. Une application simple a également été développée pour permettre l'affichage de fenêtres d'information sur les navires qui peuvent être orientées de différentes manières. Cette exploration des technologies avancées d'interaction humain-machine a permis d'acquérir des connaissances sur la façon d'utiliser avantageusement ces innovations dans un contexte de surveillance maritime ainsi que pour d'autres activités dans le domaine du renseignement.

This page intentionally left blank.

Executive summary

Exploration of collaborative environment technologies for maritime analysis

Étienne Vachon; Valérie Lavigne; DRDC Valcartier TM 2011-233; Defence R&D Canada – Valcartier; April 2012.

Introduction or background: In 2009, Defence Research and Development Canada (DRDC) Valcartier initiated the 11hm applied research project entitled "Maritime Domain Analysis through Collaboration and Interactive Visualization". The identification and tracking of Vessels of Interest (VOI) require the collaboration of various analysts from the federal departments forming the Maritime Security Operations Centers, to collect and analyze information about this vessel, to understand its intentions and assess if it may represent a threat.

In order to support collaborative work, the Intelligence Laboratory (ILAB) of the Intelligence and Information (I2) section was equipped with new displays and interactions devices. For a period of eight months, a team constituted of an intern student and a defence scientist, worked to explore how these advanced technologies could enhance collaboration and team awareness in the maritime domain.

Results: The technologies explored within this short-term project included:

- Two different multi-touch tables (Evoluce One and a Displax Overlay Multitouch surface over a Liquid Crystal Display (LCD) screen);
- A large group display (knowledge wall);
- A gesture input interaction device (Microsoft Kinect sensor);
- SitScape's web-based software for User Defined Operating Pictures (UDOP).

Multi-touch interaction is making its way into numerous every day devices and offers much more complexity and flexibility than single touch devices. Although it is spreading and evolving rapidly, it should be noted that there is not yet a standard way of handling multi-touch gestures. We explored various frameworks and implemented a multi-touch version of Google Earth that can be used on a multi-touch table as well as with a wall display using gestures that are being captured by a Microsoft Kinect sensor. A maritime situation awareness portal including various VOI analysis tools mock-ups was developed using the SitScape software and displayed on the knowledge wall. A simple application was also developed to display vessel information windows with various orientations.

Significance: The expected outcome of the project was to assess a number of advanced Human-Computer Interaction (HCI) technologies with a maritime domain application and gain insight on how to take advantage of those innovations for maritime analysis as well as for general intelligence domain activities. It is believed that these advanced interface and interaction devices will be helpful for collaborative analysis and for sharing situation awareness among a team of analysts. Many possible ways to enable teamwork were explored and the insight gained will be relevant to many of the I2 section research projects.

Future plans: The use of collaborative technologies will be developed further at DRDC Valcartier within the 11hm applied research project. Future exploration could include tablet and smart phone devices as well a speech interaction. For future work in multi-touch, the developments of MT4j and GestureWorks will be monitored as they are regularly updated. Sparsh-UI, on the other hand, is a project that is currently stalled. Use of gestures to control applications with the Microsoft Kinect will require more efforts, especially regarding gestures definition.

Sommaire

Exploration of collaborative environment technologies for maritime analysis

Étienne Vachon; Valérie Lavigne; DRDC Valcartier TM 2011-233; R & D pour la défense Canada – Valcartier; avril 2012.

Introduction ou contexte: En 2009, Recherche et développement pour la défense Canada (RDDC) Valcartier a lancé le projet de recherche appliquée 11hm « Analyse du domaine maritime à l'aide de visualisation interactive et de technologies de collaboration ». L'identification et le suivi de navires d'intérêt demande la collaboration des nombreux analystes des différent départements formant les Centres des opérations de la sûreté maritime afin de rassembler, interpréter, et présenter autant d'information que possible au sujet du navire, de comprendre ses intentions et d'évaluer s'il représente une menace.

Afin de permettre le travail en collaboration, le laboratoire du renseignement (ILAB) de la section Renseignement et information (RI) a fait l'acquisition de nouveaux équipements d'affichage et d'interaction. Pendant une période de huit mois, une équipe constituée d'un stagiaire et d'une scientifique de la défense a été formé afin d'explorer comment ces technologies poussées pourraient améliorer le travail collaboratif et la connaissance de la situation dans le domaine maritime.

Résultats: Les technologies explorées dans le cadre de ce projet à court terme incluaient:

- Deux tables multi-tactiles différentes (Evoluce One et une surface Displax Overlay Multitouch placée au-dessus d'un écran LCD);
- Un affichage mural;
- Un équipement d'interaction gestuelle (senseur Microsoft Kinect);
- Le logiciel web SitScape's pour la définition de vues opérationnelles par l'usager.

Les interactions multi-tactiles se multiplient sur des équipements de tous les jours et offrent beaucoup plus de complexité et de flexibilité que les interactions impliquant un seul contact. Même si elles se répandent et évoluent rapidement, il est important de noter qu'une manière standard de gérer les interactions multi-tactiles n'est pas encore définie. Nous avons exploré différentes architecture et implémenté une version multi-tactile de Google Earth qui peut être utilisée aussi bien sur une table multi-tactile qu'avec un affichage mural à l'aide de gestes captés par le senseur Kinect de Microsoft. Un portail d'éveil situationnel maritime incluant différentes maquettes d'outils d'analyse de navire d'intérêt a été développé à l'aide du logiciel SitScape pour être affiché sur le mur d'écrans. Une application simple a également été développée pour permettre l'affichage de fenêtres d'information au sujet des navires qui peuvent être orientées de différentes manières.

Importance: Le résultat attendu du projet consistait en l'évaluation de plusieurs technologies avancées d'interaction humain-machine avec une application maritime afin d'acquérir des connaissances sur la façon d'utiliser avantageusement ces innovations pour l'analyse maritime ainsi que pour des activités dans le domaine du renseignement en général. Nous croyons que ces interfaces et moyens d'interaction avancés seront utiles à l'analyse collaborative en équipe et au

partage de l'état de la situation parmi une équipe d'analystes. Plusieurs moyens de permettre le travail collaboratif en équipe ont été explorés et les connaissances acquises seront pertinentes pour plusieurs projets de recherche de la section RI.

Perspectives: L'utilisation de technologies collaboratives continuera d'être étudiée à RDDC Valcartier dans le cadre du projet de recherche appliquée 11hm. Des travaux d'exploration futurs pourraient inclure des tablettes et des téléphones intelligents ainsi que de la reconnaissance vocale. Pour de futurs travaux sur des appareils multi-tactiles, le développement de MT4j et de GestureWorks sera suivi car ces logiciels sont fréquemment mis à jour. Par contre, Sparsh-UI est un projet actuellement arrêté. L'utilisation de gestes pour le contrôle d'applications logicielles à l'aide de la Kinect de Microsoft nécessitera du travail supplémentaire, surtout en ce qui a trait à la définition de gestes.

Table of contents

List of figures

Acknowledgements

The authors wish to thank Denis Gouin from DRDC Valcartier for his invaluable help in characterizing the collaborative aspect of Vessel of Interest (VOI) analysis. We also are grateful to Vincent Bergeron for his participation to the brainstorming sessions, his technical help during the project and his work on the Microsoft Kinect integration. We also want to acknowledge the work performed by Roger Fortin in setting up the iLab infrastructure. Finally, we thank Frédéric Côté for his dedicated technical assistance.

1 Introduction

1.1 Context

In 2009, DRDC Valcartier initiated the 11hm applied research project entitled "Maritime Domain Analysis through Collaboration and Interactive Visualization". This research project aimed at exploring and demonstrating how visual analytics and collaborative technologies can enable a client to:

- rapidly grasp and identify the key information elements;
- gain insight into the information / situation;
- improve the visualization of the recognized maritime picture;
- better comprehend a situation and how it could develop;
- identify what is not known (data gaps and uncertainty);
- support anomaly detection, alerting and visualization;
- support teamwork.

The identification and tracking of Vessels of Interest (VOIs) is a key activity of the Regional Joint Operations Centres (RJOCs) and of the Maritime Security Operations Centers (MSOCs). VOIs are designated by the Department of National Defence (DND), MSOC partners or international partners such as the US Department of Defense. In Canada, the analysis and management of a VOI require the collaboration of various analysts from federal departments forming the MSOCs, to collect and analyze information about this vessel, to understand its intentions and assess if it may represent a threat. In order to support collaborative work, a collaboration infrastructure has also been defined and is being implemented.

In order to explore collaborative and advanced human-computer interactions, the Intelligence Laboratory (ILAB) of the Intelligence and Information (I2) section was equipped with new displays and interactions devices. For a period of eight months (from January 2011 to August 2011), a team constituted of an intern student from École des Technologies Supérieures (ÉTS) and a defence scientist, worked to explore how these advanced technologies could enhance collaboration and team awareness in the maritime domain. Another defence scientist and a computer scientist also contributed to brainstorming sessions.

1.2 Scope and objectives

The scope of the project was defined in such way that it could be achieved by the team within the period allowed (8 months). The expected outcome of the project was to assess a number of advanced Human-Computer Interaction (HCI) technologies with a maritime domain application

and gain insight on how to best take advantage of these innovations for maritime analysis as well as for general intelligence domain activities.

The technologies explored included:

- Two different multi-touch tables;
- A large group display (knowledge wall);
- A gesture input interaction device (Microsoft Kinect sensor);
- SitScape's web-based User Defined Operating Picture (UDOP).

This document presents the challenges encountered while exploring new display and interaction technologies as well as the insights gained by experimenting with them. Software considerations for integrating these devices are explained in this document. Recommendations relative to their use in the 11hm project and possibly other projects within the I2 Section are also provided.

2 Multi-touch applications development

"The use of a multi-touch table environment has much potential for use with visual analytics. The most exciting prospect is the ability for multiple analysts to simultaneously explore the same data/information space" [1]. A unique advantage of multi-touch tables is that at all times, all collaborators have equal opportunity to participate in the analysis – control is neither centralized nor passed from person to person. Users can gather around the display surface; eye-to-eye interaction is provided; information is at the users' fingertips; multi-touch interaction can improve user efficiency for several tasks; and globally, this can lead to improved shared awareness.

2.1 Maturity of multi-touch gestures, frameworks and protocols

Multi-touch interaction is fascinating and making its way into numerous every day devices. Single touch selection has been there for a while and is easy to work with. On the other hand, multi-touch gestures offer much more complexity and flexibility. Although they are spreading and evolving rapidly, it should be noted that there is not yet a standard way of handling multi-touch gestures.

Moreover, there is no recognized standard for the gestures themselves. Some gestures such as those for selection, zooming and rotation have become well accepted, although some variants exist and other less recognized gestures are emerging. Even when dealing with common gestures, there are many proposed ways to handle them and different protocols are supported by different devices making their integration complex.

As technology is moving fast in this domain, it is reasonable to expect that standard Application Programming Interfaces (APIs) and frameworks will emerge in the next few years, as formalization efforts are already underway [2]. In the meantime, creating multi-touch applications requires dealing with various protocols and the best strategy is to try as much as possible to isolate the applications code from the gesture detection and recognition APIs.

2.2 Multi-touch hardware and protocols

During this project, two multi-touch tables were used. First, we used the Evoluce One table manufactured by a German company (Evoluce) that can detect more than 20 touches simultaneously. The Evoluce One table can handle various touch protocol such as TUIO (Tangible User Interface Objects) and Window 7 Touch. The second device is not directly a multi-touch table, but a surface that is place over an LCD to give it multi-touch capabilities. This surface is a Displax Overlay Multitouch and can handle up to four touches simultaneously. This surface works using the Window 7 Touch protocol. In addition, an experiment was done using the Microsoft Kinect as detailed in Section 5.2.

The TUIO and Windows 7 Touch protocols. These two protocols are among the most used so far by multi-touch developers. TUIO is an open framework that defines a common protocol and an API for tangible surfaces. The TUIO protocol allows the transmission of an abstract description of interactive surfaces, including touch events and tangible object states [3]. Windows 7 Touch shares

the same goal as TUIO, however, it is a proprietary protocol and is an integral part of the Windows environment.

2.3 Choosing a multi-touch framework

The multi-touch framework sought must be highly customizable and easy to use. It should allow defining custom gestures. It should also be possible, for each component or group of components, to receive information when gestures are performed and thus, enable them to react accordingly. In addition, the multi-touch framework must be light and not restrictive. Being multi-platform and supporting multiple multi-touch protocols (e.g. TUIO, Windows7 Touch) is a plus.

2.3.1 MT4j – Multi-touch for Java™

MT4j is an open source, cross platform Java framework for developing visually rich multi-touch applications. This framework supports among others, the TUIO protocol and the new Window 7 Touch features. MT4j is built on top of the Processing framework which allows the use of its features and libraries. The most common multi-touch gestures are already included and can be registered modularly with any component. MT4j offers software or hardware accelerated graphics rendering and uses the OpenGL graphics standard and a component-based scene graph, which allows MT4j to achieve very good performance on supported platforms.

2.3.2 GestureWorks

GestureWorks is a multi-touch framework developed by Ideum, an Adobe Solution Partner. This framework can be used with either Adobe Flash™ or Adobe Flex™. GestureWorks uses an open source gesture library which offers over 200 predefined gestures and allows customization. As the framework is based on Flash technology, it makes it possible to integrate an application into a web browser; some demos are available online to demonstrate the concepts.

2.3.3 Sparsh-UI

Sparsh-UI is an open source multi-touch framework developed by the Iowa State University's Virtual Reality Applications Center. This framework, available in Java and C++, has the distinction of having a client-server architecture where the server is responsible for receiving and processing touch points and subsequently turning them into gestures and sending them to the client application. The modular nature of Sparsh-UI architecture allows it to be used with almost any type of device. It supports all the basic gestures and allows easy implementations of new gestures.

2.3.4 Tests and discussion

Each of these frameworks allows the construction of multi-touch applications. However, they all have their own characteristics, which make them more or less suited to our requirements. Although GestureWorks seems to be an interesting framework, it has been quickly dismissed because the trial version entitles only one hour of testing, which is too brief to give a good idea of its abilities. In fact, one hour barely provided the time to run the demos. This leaves two potential frameworks, which are open-source and thus have no limitations for testing.

MT4j comes with a dozen small applications both for demonstrations and examples. It is quite easy to make some minor changes to these examples. The processing of gestures is consistent with what is typically found in Java for event handling; that is to say, we register a component so that it receives one or many gesture type events when they occur, which it processes accordingly. As previously stated, MT4j is built using the Processing framework and OpenGL. This characteristic makes it possible to create complex and rich interfaces. However, although this feature is essential for creating games and interactive applications, it is not necessary for managing applications. The difficulty created by this architecture greatly increases the complexity of creating an application: it requires knowledge not only of MT4j, but also about Processing and OpenGL. Furthermore, MT4j currently does not have any pre-built graphical components. That means it would require the programming of all the basic interface components of our interface, and therefore require a considerable amount of effort and time, even for a simple application.

Sparsh-UI was very easy to learn. It differs from other proposed solutions by offering only the management of the multi-touch and nothing else which gives us the freedom to choose any GUI framework. The first step to run Sparsh-UI is to build a bridge, an adapter between the multi-touch device and the "GestureServer". By doing so, it is possible to support many, if not all devices with a fairly minimal effort. Subsequently, the "GestureServer" transforms the received touch-points into gestures and communicated them to the appropriate graphical components previously registered as multi-touch listeners. Because of its lighter nature, Sparsh-UI is easy to modify. Its source code is simple to understand and can be adjusted according to ours needs.

2.3.5 Conclusion

We focused our evaluation on open source multi-touch frameworks. MT4j framework is the most powerful and complete but because of its architecture complexity, it was set aside. Sparsh-UI is therefore, the selected framework. The main justification behind this choice is the freedom that this framework provides for the conception of the user interface since it only deals with multi-touch points and their transformation into gestures.

2.4 Using Sparsh-UI

2.4.1 Sparsh-UI architecture

This section is a summary of the "Sparsh Developer's Guide" available in one of the Google code wiki's page of the project: http://code.google.com/p/sparsh-ui/wiki/SparshUI_Developer_Guide. See Annex A.1 for a code extract of TouchSender.

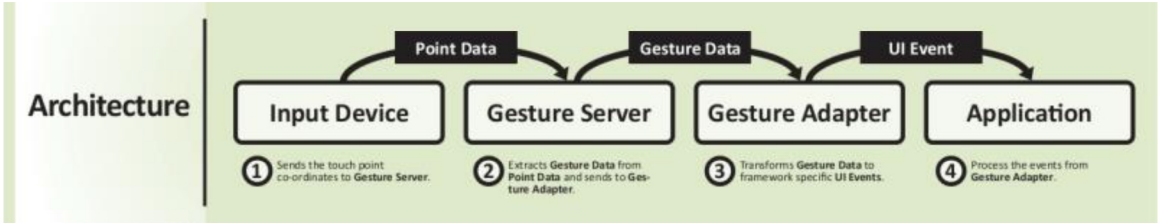


Figure 1: Sparsh-UI Architecture [4].

1. Input Device represents any touch device used to obtain input from the user. Sparsh-UI is compatible with all devices through the implementation of an adapter that transforms the devices touch data to Sparsh-UI personal touch protocol. Device adapter is discussed in section 2.4.2.

2. Gesture Server extracts the touch data received from the input device and makes this information available to the Gesture adapter.

3. Gesture Adapter transforms the touch points into gestures events and sends them to the application.

4. The application receives the gesture events from Gesture Adapter and processes them.

2.4.2 Device Adapter

The implementation of a device adapter is not a difficult task. The creator of Sparsh-UI provides an easy to follow tutorial with an example written in C++ (http://code.google.com/p/sparsh-ui/wiki/SparshUI_Device_Adapter). The following subsection will briefly describe the particularity of the implementation, in the Java programming language, of two adapters for different multi-touch devices, one using TUIO and the other using Window7 touch.

2.4.2.1 Evoluce One – TUIO

The Evoluce One multi-touch table sends on port 3333 several packets per second compliant to the TUIO protocol for each contact with the table surface. We can capture those packets using the libraries “libTUIO” easily found on the Internet. Once a packet is captured, it only remains to transform it to the Sparsh-UI format and send it to the Gesture Server on port 5945. See annexe A.1 for an example of code sending a touch point.

2.4.2.2 Displax Overlay Multitouch – Window 7 Touch

The Displax Overlay Multitouch meanwhile offers a bit more of a challenge. Compared to the TUIO protocol, Window7 Touch requires registering each frame individually with the operating system. After the application has registered its window, touch messages are forwarded to the application when an input occurs on the window.

The window registration and messages listening are not yet directly possible in Java. Since the technology is new, there is nothing in the Java standard library to handle those actions. So, we need to use native code. However, MT4j, studied previously, has a library that is able to perform those operations. As the project is open source, it is thus possible to use the library and associated source code to get the touch points. Unlike TUIO, the coordinates of the touch point are not the coordinates of the touch on the screen, but the coordinates of the touch within the window. At this stage, it is necessary that all the protocols provide the same data. The coordinates of the window on the screen are added, which gives the global position of the touch point. Once done, we can send them to the server the same way we did with the Evoluce One and the TUIO protocol.

2.4.3 Sparsh-UI and Java Swing/AWT

The Abstract Window Toolkit (AWT) is Java's original platform-independent GUI toolkit. Swing is based on AWT and was developed to answer some issues encountered with its predecessor. In order to facilitate the integration of multi-touch and to mimic as much as possible the behaviour of AWT and Swing, an overlay was built on Swing and AWT.

Figure 2 provides an overview of the Swing and AWT architectures. Figure 3 also represents an overview of this architecture, but with the addition of a multi-touch overlay. These two figures allow us to see that the multi-touch layer is inserted into what already exists without messing up the existing structure.

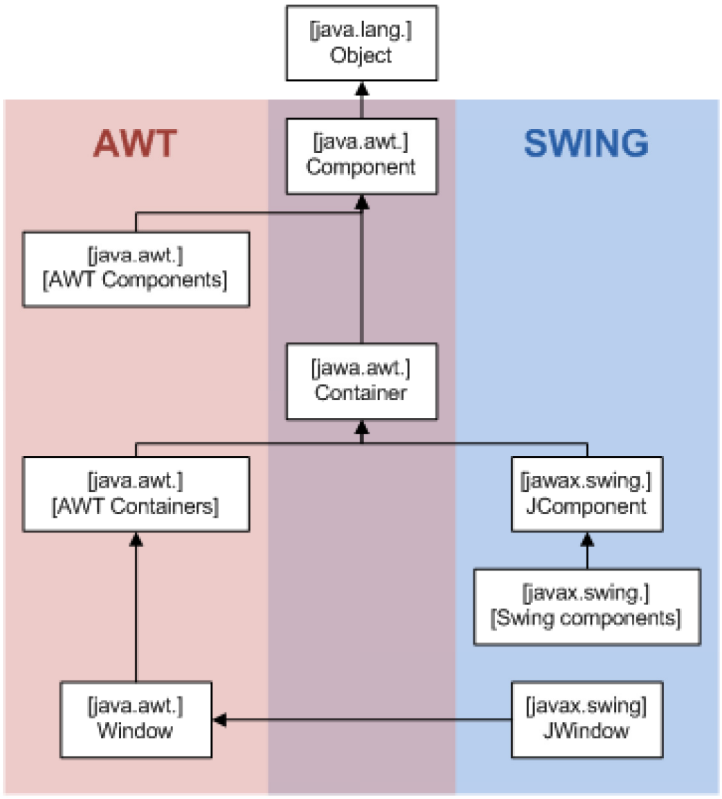


Figure 2: AWT and Swing class hierarchy.

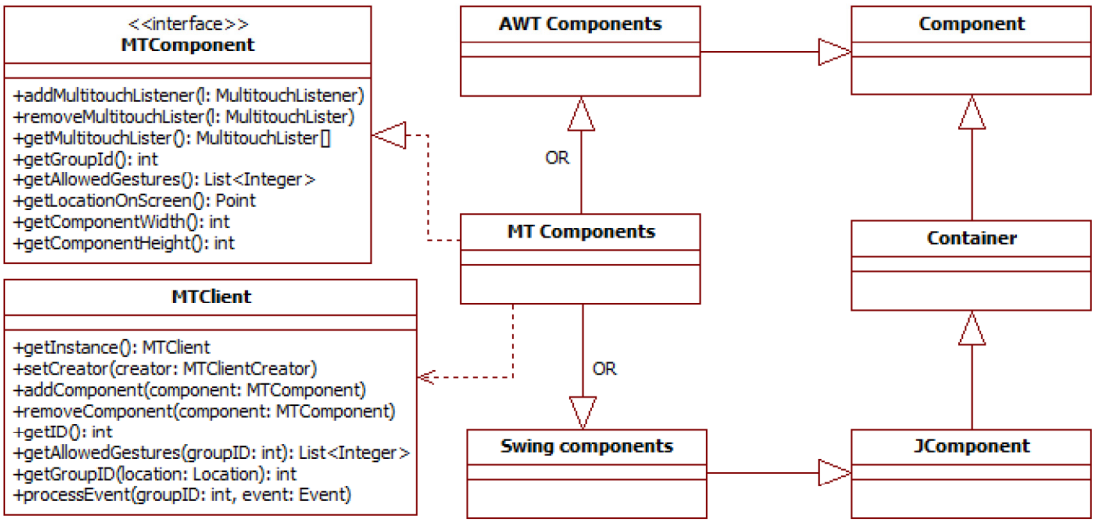


Figure 3: AWT and Swing class hierarchy with the addition of a multi-touch overlay.

The creation of a Swing/AWT multi-touch interface is almost the same as the creation of a regular interface. For each component needing multi-touch functionality, we need to create a class which implements the MTComponent interface and inherits the class that would normally be used to achieve the same interface in a normal situation without multi-touch. Once this is done, the component is usable in the same way as a regular Swing/AWT component with the exception that it can receive multi-touch events.

Some "MT Components" were made during this project and should be reusable independently of the context (see Annex A.2 for an example). In the future, if all the AWT and Swing components were made through different projects, we could have a great collection of easy-to-use multi-touch components.

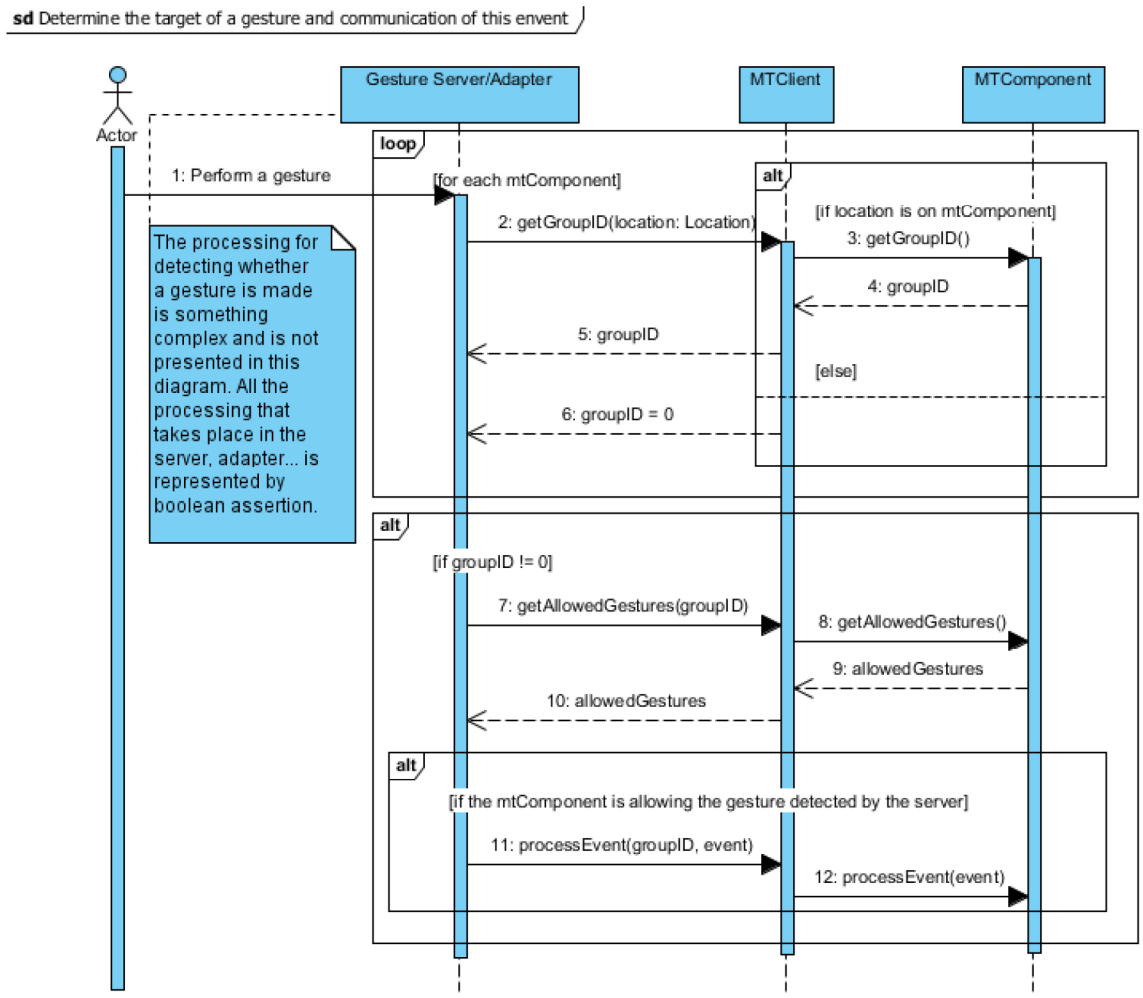


Figure 4: Sequence diagram which shows the process to determine the target component of a gesture and how the multi-touch event is communicated from the server to this component.

2.5 Embedding Swing/AWT applications into JavaFx for rotation windows

On a tabletop display, orienting display components "upwards" has no meaning and being able to display a window with an arbitrary orientation is a desirable feature since users will interact with the display from all around the table. Unfortunately, one of the problems faced with the development of our tabletop application is that no current UI toolkit provides adequate directionality support. Although arbitrary orientation would be ideal, we have found that cardinal points orientations were sufficient for our needs as users will necessarily be positioned on one of the four sides of the multi-touch table.

An application built with Swing and AWT is an easy way to start working with multi-touch applications. However, this graphics environment was not originally made to handle some of the

basic functions that one expects to find in a touch application, like the rotation of a window. To meet these needs, a solution based on JavaFx was explored.

The solution begins with a simple idea; we have an application, written using Swing/AWT and we would like this application to use some JavaFx functions. It is possible in JavaFx to use Swing/AWT components; however, they are not fully supported and can generate compatibility problems. Nevertheless, it should be doable to encapsulate an entire application made with these components.

The integration of the Swing/AWT interface was pretty straightforward, although, as expected, there were issues. Especially, several "freezes" occur because of the different ways to manage the event dispatching thread (EDT) which is responsible of the interface display. There was also a problem with the graphic theme which was not working properly with JavaFx. However, there were more important problems with the integration of multi-touch functionalities. The problems stem from the fact that the positioning methods (getLocationOnScreen(), getLocation()) do not work in JavaFx and that these methods are absolutely necessary for the multi-touch layer.

Work was performed to find a way to make these methods work and consequently enable the multi-touch. Using the position and the dimension of the window, it is possible to know the position of the top level component and enable some multi-touch gestures in our application. This solution is not perfect because we are still unable to know the position of each individual component, and therefore enable multi-touch gestures on each component. Further work is required; however, a new version of JavaFx is about to be released and according to the roadmap, the future Swing interoperability should be much better than the current version.

2.6 Multi-touch Google Earth

Some applications, such as Google Earth, could benefit greatly from being multi-touch capable. Fortunately, in this case, Google provides access to some basic functions through programming. It was therefore attempted to add a multi-touch functionality to Google Earth, as illustrated in Figure 5.

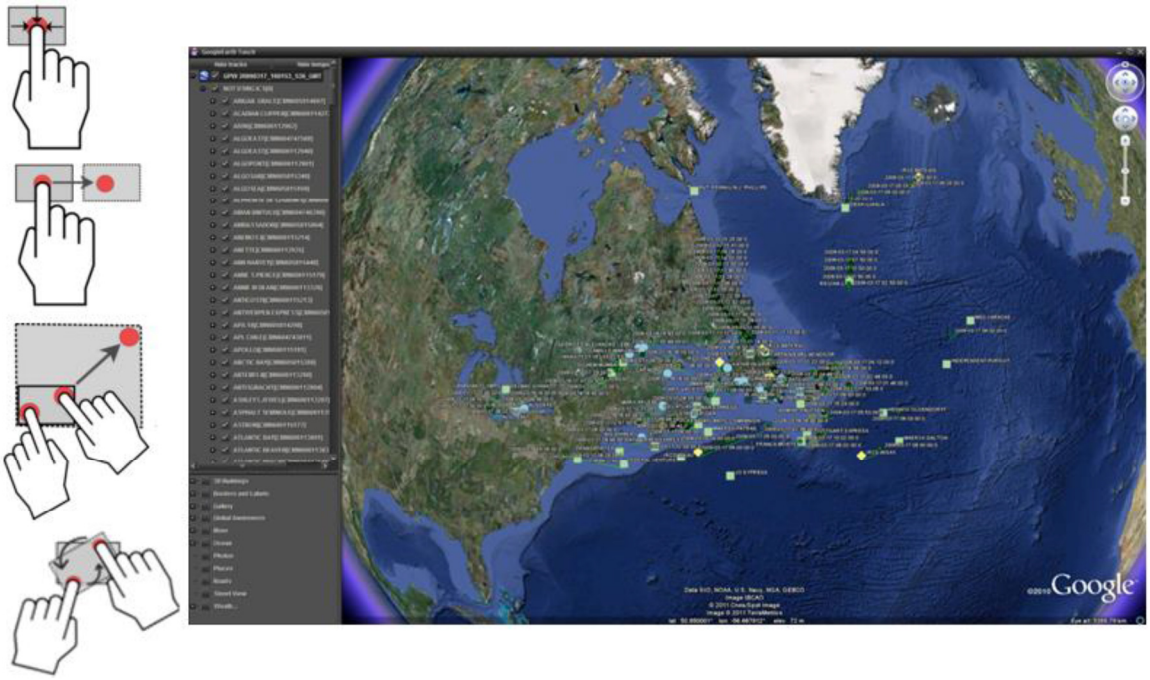


Figure 5: Multi-touch Google Earth application and gestures.

As described in the API documentation: "The Google Earth COM (Component Object Model) API allows third party applications to query information from and send commands to Google Earth" [5]. In fact, the API allows to:

- Know Google Earth state (is open, is initialized);
- Get Google Earth render window;
- Open KML (Keyhole Markup Language) files;
- Load KML from a string;
- Manage the camera view (position, tilt, rotate...);
- Partially manage map layers (weather, borders, streets...);
- Partially manage map features (loaded from KML).

2.6.1 Using Google Earth API with Java

As mentioned before, it is possible to interact with Google Earth through programming using the COM API. The earth.idl file provided by Google (http://services.google.com/earth/earth.idl) contains the description of the interfaces necessary to query and control Google Earth. Google also provides an easy to follow HTML (Hypertext Markup Language) summary of the earth.idl file: http://earth.google.com/comapi/index.html.

The main interface that represents the instance of Google Earth is IApplicationGE (Figure 6). This interface is the bridge between Google Earth and third party applications. In Java we can't call the

interface directly; we need a COM library to instantiate a Java object. There are several open source COM API libraries such as:

- COM4j : http://com4j.java.net/
- Jawin : http://jawinproject.sourceforge.net/
- JACOB : http://sourceforge.net/projects/jacob-project/

These three libraries appear to provide the desired functionality; however, JACOB is the latest of the three and the only active project. So, JACOB was chosen. Using this library, it is possible to implement all Google Earth COM API interfaces into concrete Java classes and thus, use their features (see Annex A.3 for examples).

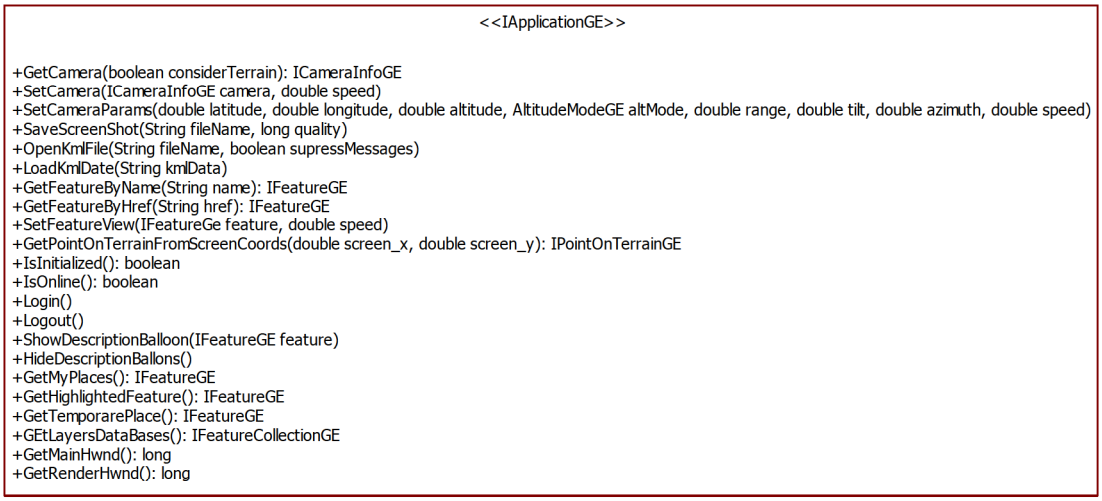


Figure 6: Class diagram of the interface "IApplicationGE", extract of the earth.idl file.

2.6.2 Embedding Google Earth in a Swing/AWT GUI

The Google Earth COM API allows to get the window handle of the render view of Google Earth (the panel containing the globe). With this handle, it is possible to move this view to another application.

To be able to perform this move, it is important to be familiar with the structure and the way Swing/AWT work and their differences. As mentioned in Section 2.4.3, Swing is built on AWT but, if we dig a little more, we find that AWT use native libraries specific to each operating system to build its components. This type of component is said to be heavyweight, as they are dependent on the underlying operating system (e.g. Window, Linux, OS X) to provide Graphical User Interface (GUI) controls like painting. On the other hand, Swing components are 100% pure Java code (except for the top-level ones: JWindow, JFrame, JDialog and JApplet) and do not have any native library dependencies, so are said to be lightweight [6]. This difference plays a fundamental role to the operation we seek to accomplish because a lightweight component does not have the capability to receive the window handle of Google Earth as opposed to a heavyweight component.

In addition, Java does not directly support the operation of re-parenting a window as we would like. To do this, we will have to uses the JNA (Java Native Access) libraries to establish a link to some Window specific functions that allow these operations. (See Annex A.4 for an example).

This page intentionally left blank.

3 Communication between applications

The workflow of VOI analysis will require easy movement between different interconnected "views" of the domain. In accordance with many "portal" models that use linked displays, the various views presenting vessel information and analysis tools may be stacked on a single screen, distributed across multiple screens, or distributed between devices.

In order to keep the same information selected on multiple views and to send relevant information from one tool to another, applications will be required to communicate with each other. To keep our solutions scalable across the multiple devices and screens, we also wish to develop applications or "widgets" that can operate independently of each other. This section proposes a design for enabling widgets to exchange information among them in order to present some information in different but linked displays and improve situational awareness according to the context.

3.1 Analysis

We have several independent applications. These applications may, however, need to exchange information. Each application has a defined role and can transmit and receive information from other widgets. These messages should be of any shape, such as text messages or complex structures containing more information.

All applications should remain fully independent from each other. In all cases, applications should avoid coupling and not be in direct relationship. The chosen solution does not have to support network communication (communication among widgets running on different computers) nor communication among applications programmed using different programming languages. However, the solution must be flexible enough to allow an easy evolution that might support these options.

3.2 Design choices

This section briefly describes two possible design choices that are each based on a design pattern. The two proposals briefly describe these patterns and the advantages and disadvantages of using them in the context of the problem presented. Then, these proposals are compared and a selection is made. The selected solution is fully described in Section 3.3.

3.2.1 First proposal - Facade

The first proposal is the use of the Facade design pattern. The intent of this pattern is to provide a unified interface to access a subsystem (in our case another widget). This would allow defining an interface, a facade, which other applications could use. This design offers the benefits of reducing the coupling between an application and its clients. However, this solution requires that a client calls the facade of all the applications that may need to receive information about an event or an action. This also implies that an application must know precisely which method of the facade should be used for each action. With few applications, this is not really a problem, but if the system evolves to contain a large number of applications, this could become more problematic.

3.2.2 Second proposal – Mediator

The second proposal is the use of the Mediator design pattern. The intent of this pattern is to avoid direct coupling by keeping objects from referring to each other explicitly. This pattern allows this by making the communication, or the mediation, an independent concept; it lets us focus on how objects interact apart from their individual behaviour. For the implementation of this pattern, each application must define an object which implements the Colleague interface; this colleague acts as a spokesman to the Mediator, who can communicate a message to other colleagues. Nevertheless, centralizing communication in this way can lead to a great complexity in the mediator because it usually encapsulates protocols, and it can become complex and make a class hard to maintain.

3.2.3 Conclusion

In a one to one communication Facade may be a good solution, but in this situation Mediator appears to be a better choice. Facade differs from Mediator in that it abstracts a subsystem of objects to provide a more convenient way to communicate. In the Facade design pattern, each application is aware of each other and communicates directly. However, in the Mediator design pattern, applications are only aware of the Mediator, which is aware of all the applications. In addition, Facadc is unidircctional; Facadc objccts makc rcqucsts to a subsystcm classcs but not vicc vcrsa, unlike Mediator which enables cooperative behaviour. In conclusion, even if the Facade is less complex and easier to implement, the Mediator seems a better choice because of its capability to evolve, to handle many applications and to provide bidirectional communication.

3.3 Solution description

3.3.1 Applicability context

The solution presented in the following sub-sections was originally designed for use with the Java programming language. However, this solution should be usable with any object-oriented language. Also, as mentioned previously in the analysis, this solution does not support network communication. For the applications to be able to communicate, they have to run in the same Java Virtual Machine (JVM). To do so, applications must be bundled together and started by a "parent application" or by any application already running in the JVM. However, if needed, this restriction can be bypassed by redefining the Mediator class to handle a different way of doing.

3.3.2 Solving the problem

To solve the problem of communication between applications, it was decided to establish a communication interface dedicated only to this problem. To do so, a solution using the Mediator design pattern (Figure 7) was developed. This pattern is a predefined solution to the problem we face. Concretely, to implement this design pattern, each application must define its own implementation of the Colleague interface. Then, when a colleague wants to send a message, it gives it to the mediator who sends it to the others colleagues who do whatever they want with it. This solution meets the requirements set out earlier by not creating any direct coupling between applications and provides sufficient flexibility to respond to other requests.

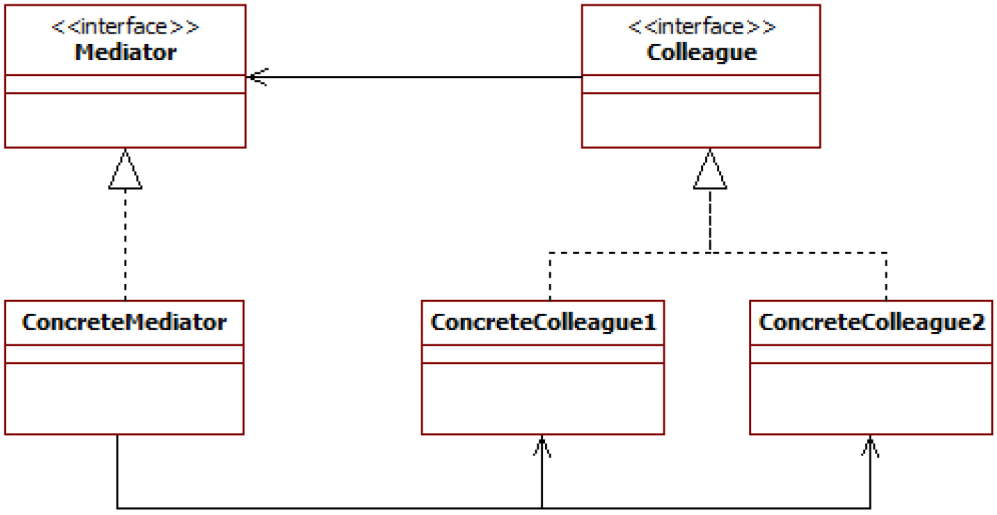


Figure 7: UML representation – Mediator design pattern.

One of the specific needs is the ability to send messages of any kind. To do this, an interface "Message" has been created. To send a complex structure a colleague only needs to create an object which implements the Message interface. This will hide the complexity of the message to the mediator and to all the colleagues who do not need to read the message. It also provides a first way for a colleague to dispose of messages that do not concern it: if the colleague does not recognize the type of the message, it simply ignores it. If a message type is used by several colleagues and for various operations, the implementation of the Message interface should provide a way for a colleague to identify whether the message is intended for it. This identification could be done, for example, using an id specific to an operation; however, in this case, all involved colleagues must know the relevant identifiers.

The handling of incoming messages can become problematic at the conceptual level if a colleague receives a large number of different messages. In fact, a potential problem could arise if the operations are highly variable and if those operations need to access many parts of the application. This would result in a non-cohesive class with a lot of code only related to a specific operation. To resolve this problem, the default implementation uses the design pattern Chain of Responsibility (Figure 8). This pattern can prevent the implementation of a colleague from containing all the algorithms. These algorithms are rather encapsulated in the MessageHandler subclasses. The colleague gives the message to the first handler who tries to deal with it if he cannot, the message is passed to the next handler in the chain and so on until the message is read or the end of the chain is reached, which means that the message is not for this colleague.

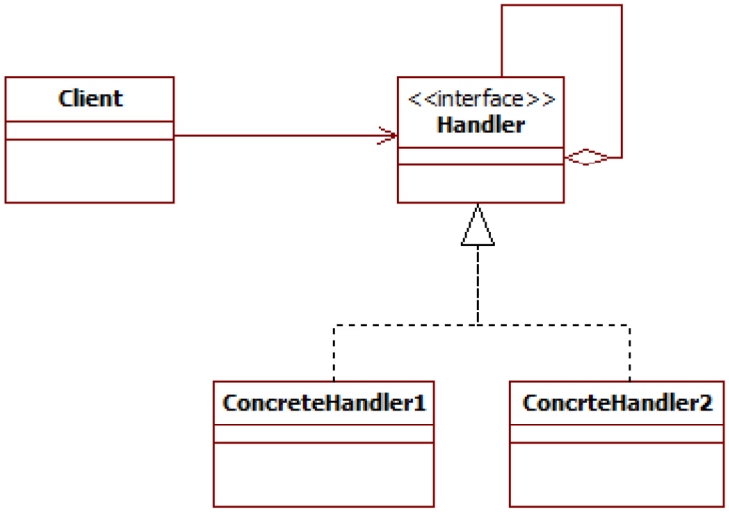


Figure 8: UML representation – Chain of responsibility design pattern.

Figure 9 presents an overview of the implemented solution. Here are the descriptions of the classes from the UML diagram:

- Client: sends commands to the first object in the chain that may handle the command. In our situation, the client is a Colleague.
- Colleague: defines the interface of a colleague. This is a family of objects, which are unaware of each other but who must exchange information. A colleague has access to application resources it represents.
- ConcreteColleague: implementation of the Colleague interface. This class represents an application, a widget. The "ConcreateColleague" may send messages through the Mediator and is responsible for processing incoming messages.
- ConcreteMediator: implementation of the Mediator interface. This class maintains a reference to the colleague objects and implements the communication between them.
- ConcreteHandler: handles the request it is responsible for; if it can handle the request it does so, otherwise it sends the request to its successor.
- DefaultColleague: Default implementation of a colleague. This implementation uses the Chain of Responsibility pattern to handle messages.
- DefaultMediator: Default implementation of a Mediator.
- Handler: defines the interface for handling requests.
- Mediator: defines an interface of a Mediator. A mediator is an object which allows communication between Colleague objects.
- Message: defines the interface of a Message. A Message is the representation of information exchanged between applications.
- MessageHandler: is a type of a handler appropriate to the situation.

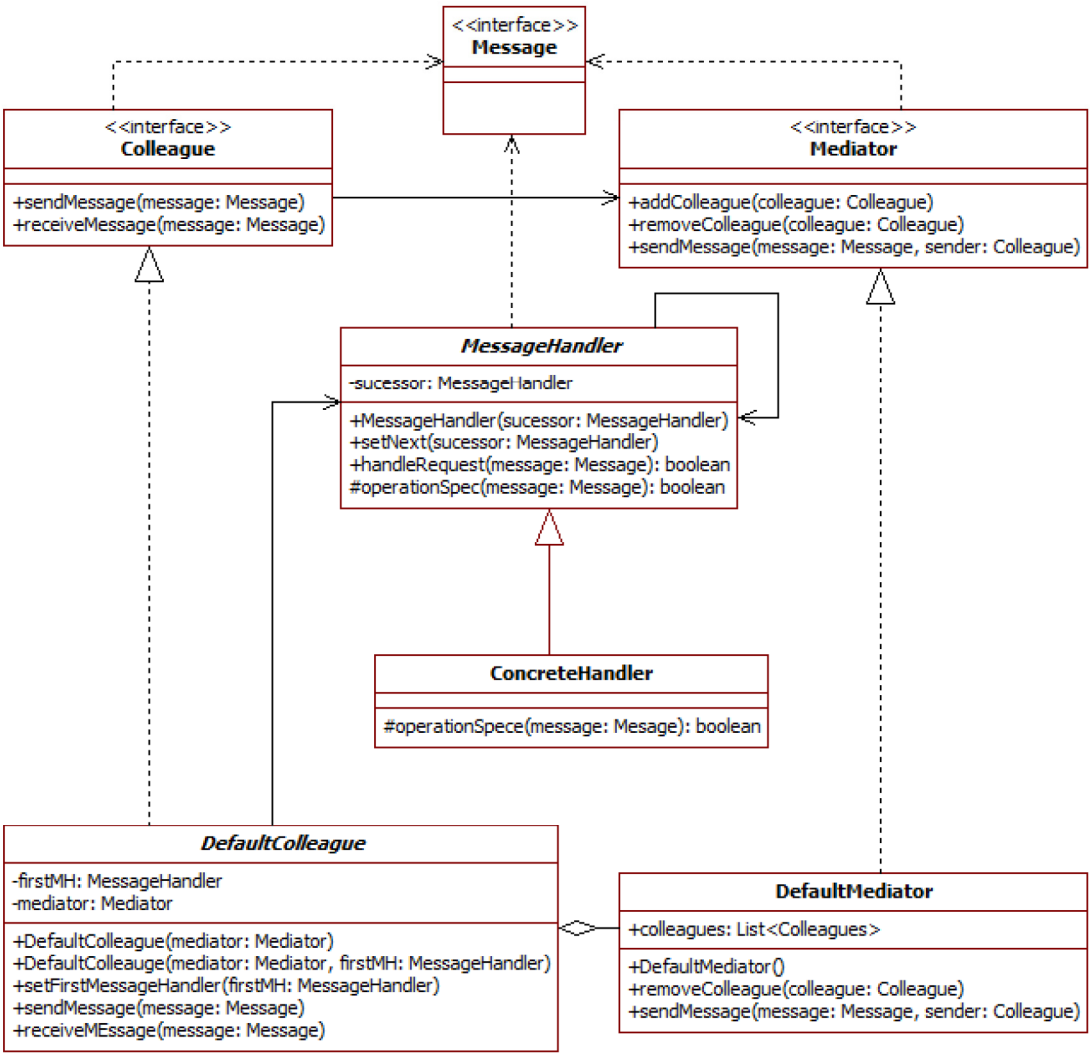


Figure 9: UML representation – Communication between applications; implemented solution.

This page intentionally left blank.

4 Extracting vessel information

VOI information is a mixture of geospatial and temporal shapes, commercial facts, photographs, schedules, self-reported information, and intelligence information, as sketched in Figure 10. Lloyd's Ship Register is one the available sources for vessel information. Photos of a large number of ships can also be obtained from http://shipspotting.com/.

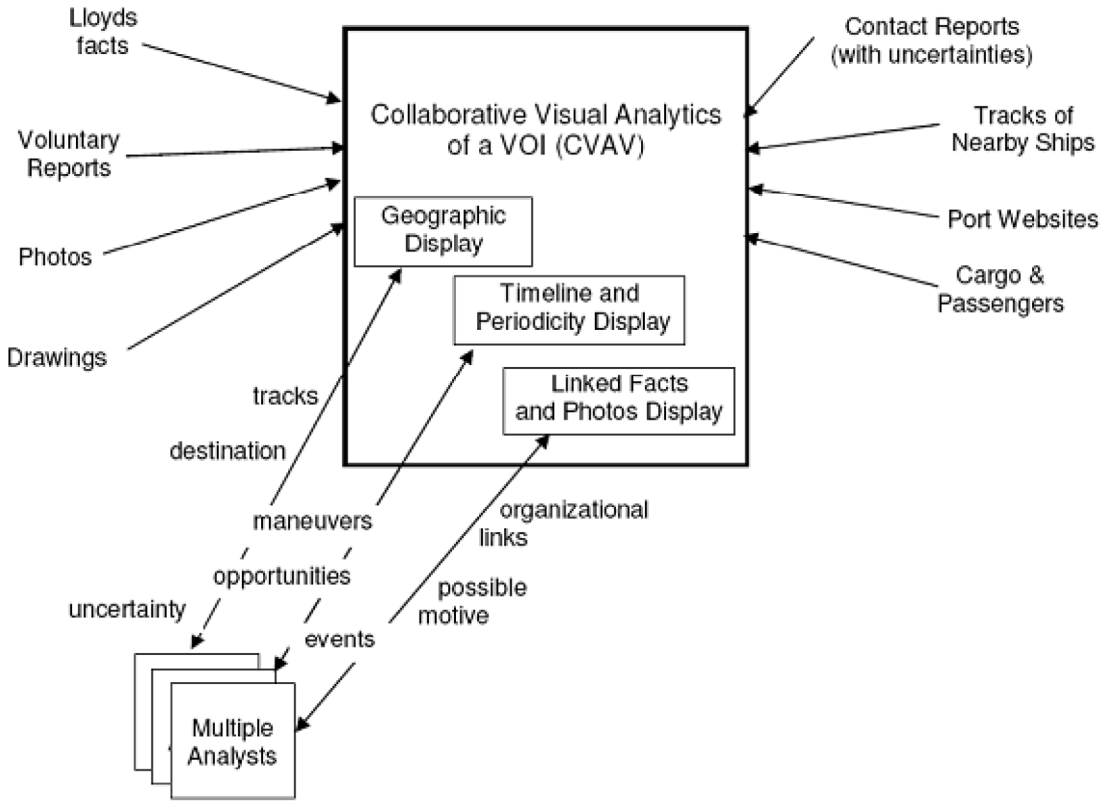


Figure 10: Context diagram for the collaborative analysis of a VOI, implying the use of linked geographical, temporal and abstract visual information, collected and interpreted by multiple analysts [7].

4.1 Extracting information from Lloyd's and ShipSpotting

Using an application written in Java, it is possible to perform queries on the Lloyd's website. In summary, the application performs an http query which returns a web page. Then, using the library HTMLParser (http://htmlparser.sourceforge.net/), it is possible to parse this page, gather the information it contains, format it and show it in our application. The same process was used to extract vessels pictures from http://shipspotting.com/.

4.2 Display of vessel information

Our small Lloyd's application allows us to retrieve vessel information along with a vessel photo and display it as a small vessel card as shown in Figure 11. This application also support queries to find specific vessels (Figure 12). As explained in Section 2.5, working on a surface device may require displaying windows along different orientations as in Figure 13.

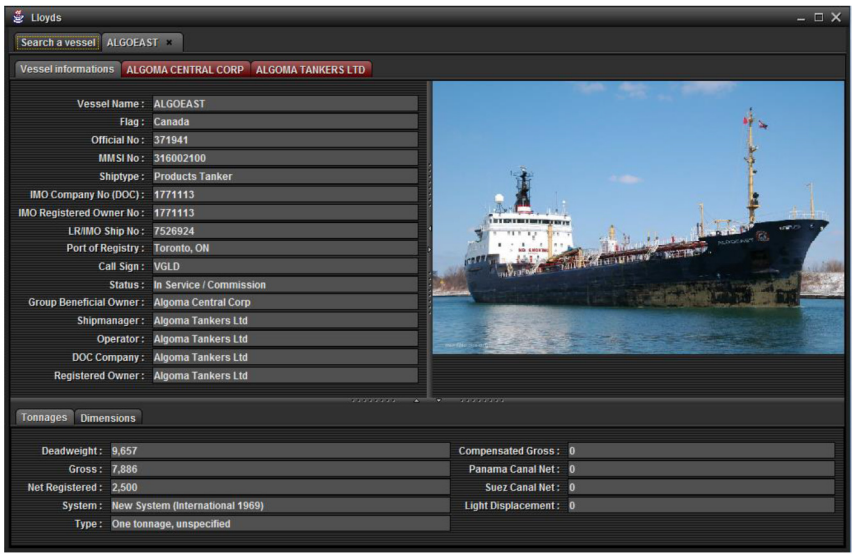
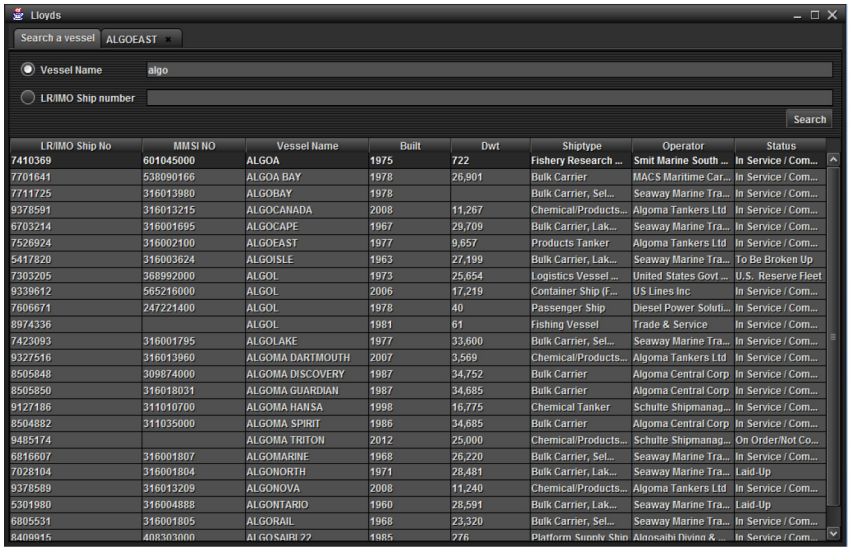


Figure 11: A vessel card in our Lloyd's application, showing retrieved information about a vessel and a photo from the ShipSpotting website.



LR/IMO Ship No	MMSI NO	Vessel Name	Built	Dwt	Shiptype	Operator	Status
7410369	601045000	ALGOA	1975	722	Fishery Research ...	Smit Marine South ...	In Service / Com...
7701641	538090166	ALGOA BAY	1978	26,901	Bulk Carrier	MACS Maritime Car...	In Service / Com...
7711725	316013980	ALGOBAY	1978		Bulk Carrier, Sel...	Seaway Marine Tra...	In Service / Com...
9378591	316013215	ALGOCANADA	2008	11,267	Chemical/Products...	Algoma Tankers Ltd	In Service / Com...
6703214	316001695	ALGOCAPE	1967	29,709	Bulk Carrier, Lak...	Seaway Marine Tra...	In Service / Com...
7526924	316002100	ALGOEAST	1977	9,657	Products Tanker	Algoma Tankers Ltd	In Service / Com...
5417820	316003624	ALGOISLE	1963	27,199	Bulk Carrier, Lak...	Seaway Marine Tra...	To Be Broken Up
7303205	368992000	ALGOL	1973	25,654	Logistics Vessel ...	United States Govt ...	U.S. Reserve Fleet
9339612	565216000	ALGOL	2006	17,219	Container Ship (F...	US Lines Inc	In Service / Com...
7606671	247221400	ALGOL	1978	40	Passenger Ship	Diesel Power Soluti...	In Service / Com...
8974336		ALGOL	1981	61	Fishing Vessel	Trade & Service	In Service / Com...
7423093	316001795	ALGOLAKE	1977	33,600	Bulk Carrier, Sel...	Seaway Marine Tra...	In Service / Com...
9327516	316013960	ALGOMA DARTMOUTH	2007	3,569	Chemical/Products...	Algoma Tankers Ltd	In Service / Com...
8505848	309874000	ALGOMA DISCOVERY	1987	34,752	Bulk Carrier	Algoma Central Corp	In Service / Com...
8505850	316018031	ALGOMA GUARDIAN	1987	34,685	Bulk Carrier	Algoma Central Corp	In Service / Com...
9127186	311010700	ALGOMA HANSA	1998	16,775	Chemical Tanker	Schulte Shipmanag...	In Service / Com...
8504882	311035000	ALGOMA SPIRIT	1986	34,685	Bulk Carrier	Algoma Central Corp	In Service / Com...
9485174		ALGOMA TRITON	2012	25,000	Chemical/Products...	Schulte Shipmanag...	On Order/Not Co...
6816607	316001807	ALGOMARINE	1968	26,220	Bulk Carrier, Sel...	Seaway Marine Tra...	In Service / Com...
7028104	316001804	ALGONORTH	1971	28,481	Bulk Carrier, Lak...	Seaway Marine Tra...	Laid-Up
9378589	316013209	ALGONOVA	2008	11,240	Chemical/Products...	Algoma Tankers Ltd	In Service / Com...
5301980	316004888	ALGONTARIO	1960	28,591	Bulk Carrier, Lak...	Seaway Marine Tra...	Laid-Up
6805531	316001805	ALGORAIL	1968	23,320	Bulk Carrier, Sel...	Seaway Marine Tra...	In Service / Com...
8409915	408303000	ALGOSAIBI 22	1985	276	Platform Supply Ship	Algosaibi Diving & ...	In Service / Com...

Figure 12: Another picture of our Lloyd's application, showing the results of a vessel search.

Figure 13: Lloyd's application embedded in JavaFx as explained in Section 2.5.

This page intentionally left blank.

5 Use of a large display for situation awareness

Perhaps the simplest model of collaboration is through the use of a large group display, also called a knowledge wall. Control of the displayed information related to various tasks can be shared among an assembled team of analysts, each of whom can view contributions from the others in real time. Knowledge walls are already common in operation centers.

In a maritime context, a large display can be used to maintain situation awareness among the agencies working together. For collaborative analysis, we believe that this type of display should be as interactive as possible. It could be used to analyse the situation in real time as well as present results to a larger audience.

This section explores the use of SitScape's software, a knowledge wall and a gesture interaction device in the context of a maritime portal.

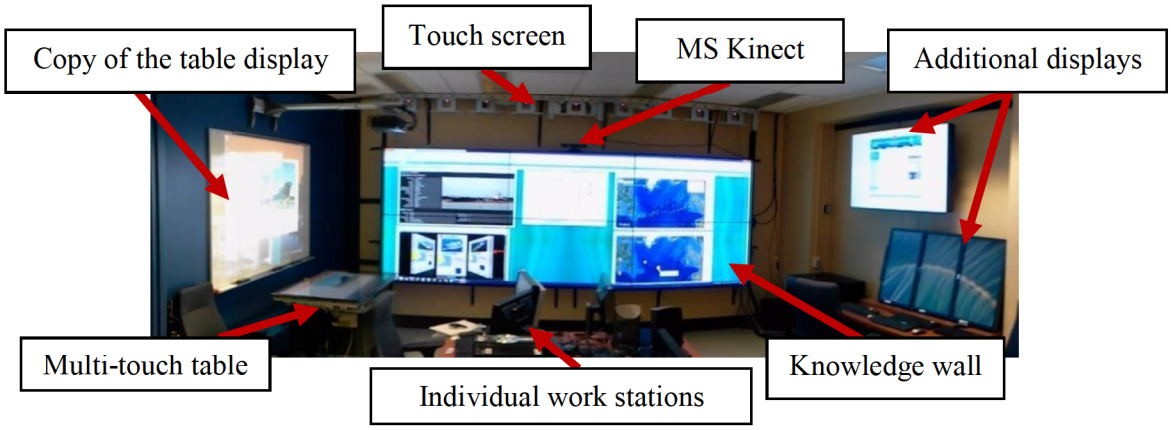


Figure 14: A picture of the large display and its immediate surroundings in the ILAB.

5.1 Mock-up with SitScape

In accordance with UDOP concepts, team members can assemble the operational picture using a variety of apps, running on many portlets, on multiple displays, and if necessary in different locations. SitScape's web-based software [8] allows users to easily aggregate and visualize disparate applications and information sources into a collaborative UDOP for live monitoring, situational awareness, information sharing, and visual contextual collaboration.

SitScape makes it possible to embed various elements in a web page, such as: desktop applications, PDF documents, PowerPoint presentations, web pages or parts of them, etc. This can be easily configured using the "Add Content" function which guides us in the process of adding almost anything (shown in Figure 15). SitScape allows sharing views with other users. It is even possible to annotate a view to facilitate the collaboration. It also includes many functions to help the users manage their content and provides many ways to display it.

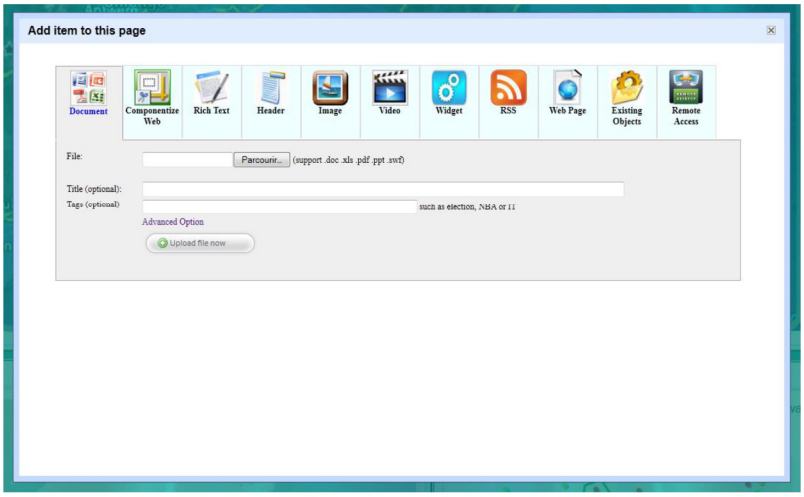


Figure 15: This figure show the Sitscape "Add Content" popup.

In order to explore SitScape's capabilities, we developed a small portal with various tabs related to maritime situation awareness and analysis topics such as surveillance, VOI analysis, search and rescue operations and anomaly detection (see Figures 16 and 17). The UDOP can be customized into a number of tab folders and it is accessed through a simple browser. Adding content to the SitScape portal was quite simple and really fast. This tool has good potential for rapid portal creation, although it may not support all types of applications and displaying applications running on a remote computer can lag noticeably.

The result on the knowledge wall is shown in Figure 18. This infrastructure has been built using a 3 x 2 configuration of 60 Inch LCD panels. It provides a seamless display of 1366 x 768 pixels. The total size of the configuration is 13' x 5'. A touch screen and MS Kinect interactions are enabled.

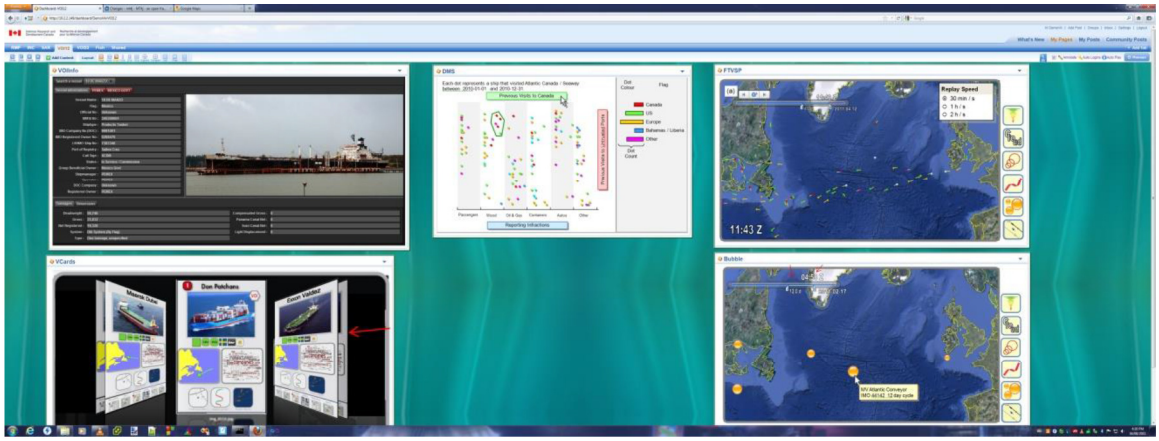


Figure 16: Example of a maritime portal mock-up built with SitScape involving many maritime application mock-ups.

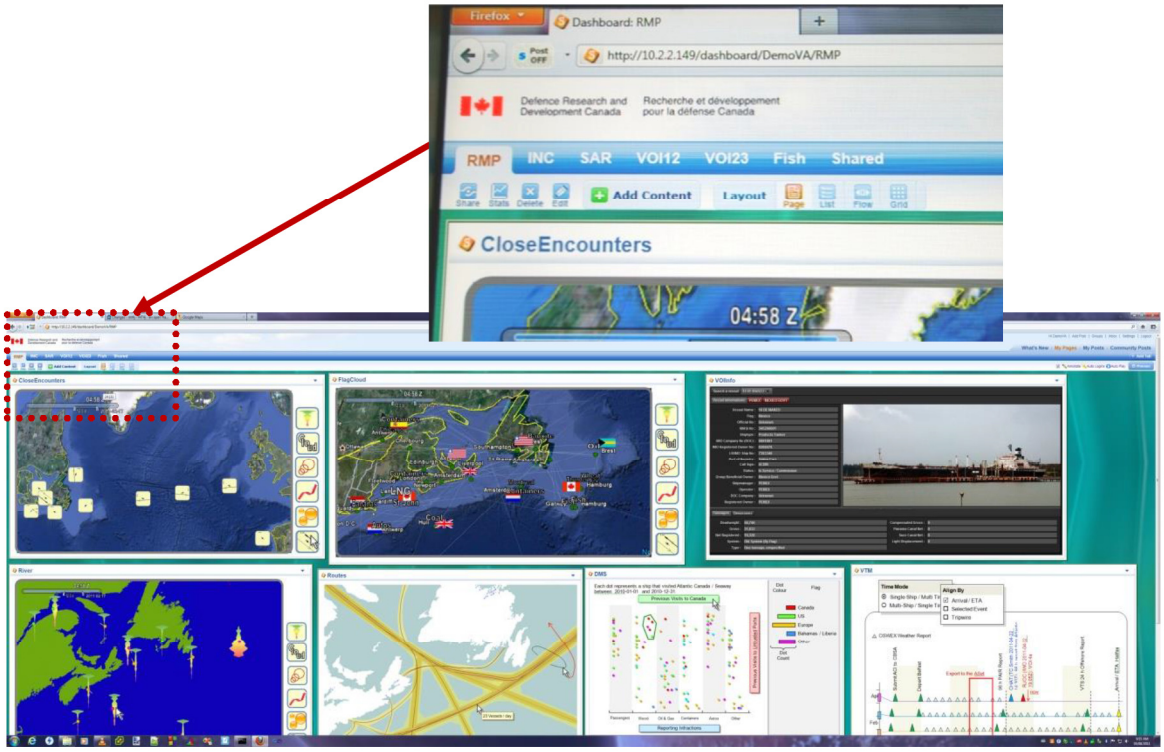


Figure 17: Another example of a maritime portal mock-up built with SitScape involving many maritime application mock-ups.

Figure 18: Maritime portal on the knowledge wall.

5.2 MS Kinect gesture interface with Google Earth on the large display

The knowledge wall of the ILAB can be controlled by standard keyboards and mice, as well as with a touch screen device that was installed on it. However, as the display is indeed quite large, users may want to interact with it from a few steps away in order to have a full view of the information presented. In that case, gesture interaction may prove handy. In this section we report on the use of a Microsoft Kinect sensor to interact with our multi-touch version of Google Earth (presented in Section 2.6).

Microsoft recently released a Software Development Kit (SDK) to ease software development with the Kinect. This SDK allows among other things to build wireframes of people in front on the Kinect and thus determine the position of certain body parts like hands. This feature allowed our computer scientist Vincent Bergeron to create, using hand position, a feed of TUIO events that are sent directly to our multi-touch Google Earth application without requiring modifications to it. All available gestures can theoretically be done. However, because the gestures are done with hands in the air, some are hard to do, like a click. A “touch” is detected when a hand is at a certain distance of the body. This means that to do a click, it is necessary to cross this invisible line and comeback rapidly and while doing this the cursor needs to stay at the same place which is not easy because even a little movement can change its position. Rotate and zoom gestures, on the other hand, are quite easy to perform.

This experience with the Kinect shows that it is possible to create a functional interface with this technology but, it will require a little more effort (especially related to gesture definitions) to make it usable in everyday life situations.

6 Dicussion and recommendations

For future work in multi-touch applications, it would be interesting to watch the developments of MT4j and GestureWorks. MT4j is a project regularly updated with a potentially bright future. The same goes for GestureWorks which, although it has not been tested enough in this project, could be an interesting framework to work with in the future. Sparsh-UI, on the other hand, is a project that is currently stalled; it can be used for proofs of concept or small applications, but its future is uncertain. Windows already has its own multi-touch protocol and it is possible to develop applications in this environment with many Microsoft technologies like Silverlight. However even if no work was done on these technologies in this project, the limited information available on this subject on internet suggests that improvements are needed and we can assume that this should come out soon.

Java Swing/AWT are nice technologies, but are not build to handle multi-touch operations like the rotation of an entire window. They can be used to do simple applications, but this technology should not be used for the GUI of a major application. An all new version of JavaFx is about to be released with a lot of new features which can be very interesting. This could be a great solution for a Java multi-touch GUI but a native way of handling touch is not announced yet and would be required.

Multi-touch interaction is also beginning to make its way into web pages, partly because of smart phones, which often offer a multi-touch capability. The HTML5 standard is not yet fully defined, but it will certainly contribute to advance this technology [9].

This page intentionally left blank.

7 Conclusion

This document presented the outcome of a short-term project which aimed at exploring new HCI technologies applied to maritime situation awareness and analysis. More specifically, we experimented with two different multi-touch tables, a large display wall, a gesture interaction device and a rapid portal designing software. Maritime applications and mock-ups were built to showcase the capabilities of these new technologies for their use in a maritime surveillance context.

We implemented a multi-touch version of Google Earth that can be used on a multi-touch table as well as with the wall display using gestures that are being captured by a Microsoft Kinect sensor. A maritime situation awareness portal including various VOI analysis tool mock-ups was developed using the SitScape software and displayed on the knowledge wall. A simple application was also developed to display vessel information windows with various orientations.

It is believed that these advanced interface and interaction devices will be helpful for collaborative analysis and for sharing situation awareness among a team of analysts. Many possible ways to enable collaborative teamwork were explored and the insight gained will be relevant to many of the I2 section research projects. They will be developed further at DRDC Valcartier within the 11hm applied research project “Maritime Domain Analysis through Collaboration and Interactive Visualization”. Future exploration could include tablet and smart phone devices as well a speech interaction.

This page intentionally left blank.

References

[1] T. Butkiewicz, D.H. Jeong, W. Ribarsky and R. Chang (2009), Hierarchical multi-touch selection techniques for collaborative geospatial analysis. In Proc SPIE Defense, Security and Sensing 2009.

[2] D. Kammer, J. Wojdziak, M. Keck, R. Groh and S. Taranko (2010), Towards a Formalization of Multi-touch Gestures, ACM International Conference on Interactive Tabletops and Surfaces 2010, 7-10 November, 2010, Germany, http://www.dfki.de/its2010/papers/pdf/fp198.pdf, retrieved August 2011.

[3] TUIO (2011), http://www.tuio.org/, retrieved August 2011.

[4] S. Gilbert (2011), Sparsh-UI: Project Goal, http://code.google.com/p/sparsh-ui/, retrieved August 2011.

[5] Google (2011), Google Earth COM API Documentation, http://earth.google.com/comapi/, retrieved August 2011.

[6] Deepun (2007), What are lightweight and heavyweight components?, http://www.interviewjava.com/2007/05/what-are-lightweight-and-heavyweight.html, May 2007, retrieved August 2011.

[7] M. Davenport (2009), Opportunities for Applying Visual Analytics for Maritime Awareness, MacDonald Dettwiler and Associates Ltd. & Salience Analytics, DRDC Scientific Authority: Valérie Lavigne, DRDC Valcartier CR 2009-227, October 2009.

[8] SitScape (2011), http://www.sitscape.com, retrieved August 2011.

[9] B. Smus (2011), Developing for Multi-Touch Web Browsers, http://www.html5rocks.com/en/mobile/touch.html , April 2008, retrieved August 2011.

This page intentionally left blank.

Annex A Implementation example

A.1 Code extract from TouchSender.java

ca.rddc.va.mt.sparshui.TouchSender.java – sendPoint(...)

```
1   public void sendPoint(int sessionId, int state, float  xCoord, float yCoord) {
2     if (connected) {
3       try {
4         // We send only one touch point by buffer
5         dos.writeInt(new Integer(1));
6         dos.writeInt(sessionId);
7         dos.writeFloat(xCoord);
8         dos.writeFloat(yCoord);
9         dos.writeByte(state);
10        dos.flush();
11      } catch (IOException ioe) {
12        System.err.println("[TouchSender] - Touch event
13                          has not been sent.");
14      }
15    }
16  }
```

A.2 Example of implementations of a multi-touch component

ca.rddc.va.mt.component.MTCanvas

```
1   package ca.rddc.va.mt.component;
2
3   import java.awt.Canvas;
4   import java.util.ArrayList;
5   import java.util.List;
6   import ca.rddc.va.mt.MTClient;
7   import ca.rddc.va.mt.MultitouchListener;
8
9   public abstract class MTCanvas extends Canvas implements MTComponent {
10     private final int groupId;
11     public transient List<MultitouchListener> multitouchListeners;
12
13     public MTCanvas() {
14        super();
15        groupId = MTClient.getInstance().getID();
16        multitouchListeners = new ArrayList<MultitouchListener>();
17     }
```

```
18  public int getGroupId() {
19      return groupId;
20  }
21
22  @Override
23  public void addMultitouchListener(MultitouchListener l) {
24      if (l != null) {
25          multitouchListeners.add(l);
26      }
27  }
28
29  @Override
30  public void removeMultitouchListener(MultitouchListener l) {
31      if (l != null) {
32          multitouchListeners.remove(l);
33      }
34  }
35
36  @Override
37  public MultitouchListener[] getMultitouchListener() {
38      MultitouchListener[] mts = new MultitouchListener[multitouchListeners.size()];
39      multitouchListeners.toArray(mts);
40      return mts;
41  }
42
43  @Override
44  public abstract List<Integer> getAllowedGestures();
45
46  @Override
47  public void setVisible(boolean b) {
48      if (b) {
49          MTClient.getInstance().addComponent(this);
50      } else {
51          MTClient.getInstance().removeComponent(this);
52      }
53      super.setVisible(b);
54  }
55
56  @Override
57  public int getComponentWidth() {
58      return super.getWidth();
59  }
60
61  @Override
62  public int getComponentHeight() {
63      return super.getHeight();
64  }
65  }
```

A.3 Example of method implementation using JACOB

A.3.1 Starting Google Earth and getting the IApplicationGE interface

This is an example demonstrating how to start Google Earth. The string "GoogleEarth.ApplicationGE" specifies to Window which interface we want to interact with so that it loads it into memory.

Starting Google Earth – startGE()

```
1  private ActiveXComponent applicationGE = null;
2
3  public boolean startGE() {
4     try {
5        applicationGE = new  ActiveXComponent("GoogleEarth.ApplicationGE");
6     } catch (final ComException e) {
7        e.printStackTrace();
8        return false;
9     }
10
11    // We block the execution other action while Google Earth is opening
12    while (!isInitialized() && !isOnline()) {
13       try {
14          synchronized (this) {
15             wait(100);
16          }
17       } catch (final InterruptedException e) {
18          e.printStackTrace();
19          return false;
20       }
21    }
22    return true;
23 }
```

A.3.2 Getting another interface

Unlike the IApplication interface, all the other interfaces can't be obtained directly from the operating system. They must be returned by an already existing interface, in this example, IApplicationGE.

Getting another interface – getTemporaryPlaces()

```
1   public IFeatureGE getTemporaryPlaces() {
2      final ActiveXComponent axc;
3      try {
4         axc = applicationGE.invokeGetComponent("GetTemporaryPlaces");
5      } catch (final ComException e) {
6         e.printStackTrace();
7         return null;
8      }
9      return new IFeatureGE(axc);
10  }
```

A.3.3 Calling a simple “get” method

Calling a simple “get” method – isInitialized()

```
1   public boolean isInitialized() {
2      int initialized = 0;
3      try {
4         initialized = (Integer)
5         applicationGE.invoke("IsInitialized").getInt();
6         return (initialized != 0);
7      } catch (final ComException e) {
8         return false;
9      }
10  }
```

A.4 Using JNA

Creating a link with User32 functions

```
1   public interface MyUser32 extends User32   {
2      public static MyUser32 INSTANCE = (User32) Native.loadLibrary("user32",
3                                                 User32.class);
4      public boolean EndTask(int hWnd);
5      public boolean ShowWindow(int hWnd, int nCmdShow);
6     public int SetParent(int hWndChild, int hWndNewParent);
7     public boolean MoveWindow(int hWnd, int X, int Y, int nWidth, int nHeight, boolean
8      bRepaint);
9   }
```

Using a Window function

```
1  // attach GE render window to the canvas
2  try {
3     MyUser32.INSTANCE.SetParent(iage.getRenderHwnd(), (int) getGUIHwnd());
4  } catch (ComException e) {
5     e.printStackTrace();
6  }
```

This page intentionally left blank.

List of symbols/abbreviations/acronyms/initialisms

API	Application Programming Interface
AWT	Abstract Window Toolkit
COM	Component Object Model
DND	Department of National Defence
DRDC	Defence Research and Development Canada
EDT	Event Dispatching Thread
ÉTS	École des Technologies Supérieures
GUI	Graphical User Interface
HCI	Human-Computer Interaction
HTML	Hypertext Markup Language
IDL	Interface Description Language
I2	Intelligence and Information
ILAB	Intelligence Laboratory
JVM	Java Virtual Machine
JNA	Java Native Access
KML	Keyhole Markup Language
LCD	Liquid Crystal Display
MSOC	Maritime Security Operations Center
MT4j	Multi-touch 4 Java
PDF	Portable Document Format
R&D	Research & Development
RDDC	Recherche et développement pour la défense Canada
RI	Renseignement et information
RJOC	Regional Joint Operations Centre
SDK	Software Development Kit
TUIO	Tangible User Interface Objects
UDOP	User Defined Operating Picture
UI	User Interface
UML	Unified Modeling Language
VOI	Vessel of Interest

This page intentionally left blank.

DOCUMENT CONTROL DATA

(Security markings for the title, abstract and indexing annotation must be entered when the document is Classified or Designated)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence Research and Development Canada – Valcartier
2459 Pie-XI Blvd North
Quebec (Quebec)
G3J 1X5 Canada

2a. SECURITY MARKING (Overall security marking of the document including special supplemental markings if applicable.)

UNCLASSIFIED

2b. CONTROLLED GOODS

(NON-CONTROLLED GOODS)
DMC A
REVIEW: GCEC JUNE 2010

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

Exploration of collaborative environment technologies for maritime analysis

4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)

Vachon. É.; Lavigne, V.

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)	6b. NO. OF REFS (Total cited in document.)
April 2012	60	9

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Memorandum

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence Research and Development Canada – Valcartier
2459 Pie-XI Blvd North
Quebec (Quebec)
G3J 1X5 Canada

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
11hm	

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
DRDC Valcartier TM 2011-233	

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

Unlimited

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

The identification and tracking of Vessels of Interest (VOIs) require the collaboration of various analysts from the federal departments forming the Maritime Security Operations Centers, to collect and analyze information about this vessel, to understand its intentions and assess if it may represent a threat. In order to support collaborative work, the Intelligence Laboratory (ILAB) of the Intelligence and Information (I2) section was equipped with new advanced displays and interactions devices. For a period of eight months, a team worked to explore how these technologies could enhance collaborative analysis and shared situation awareness among analysts.

Existing multi-touch frameworks were explored and a multi-touch version of Google Earth was implemented. It can be used on a multi-touch table and with a wall display using gestures that are being captured by a Microsoft Kinect sensor. A maritime situation awareness portal including various VOI analysis tools mock-ups was developed and displayed on the knowledge wall. A simple application was also developed to display vessel information windows with various orientations. This exploration of advanced Human-Computer Interaction technologies produced insights on how to take advantage of these innovations in a maritime surveillance context as well as in other intelligence domain activities.

L'identification et le suivi de navires d'intérêt demande la collaboration d'analystes variés provenant des départements fédéraux formant les Centre des opérations de la sûreté maritime afin de rassembler, interpréter, et présenter autant d'information que possible au sujet du navire, de comprendre ses intentions et d'évaluer s'il représente une menace. Afin de permettre le travail en collaboration, le laboratoire du renseignement (ILAB) de la section Renseignement et information (RI) a fait l'acquisition de nouveaux équipements d'affichage et d'interaction avancés. Pendant une période de huit mois, une équipe a exploré comment ces technologies pourraient améliorer le travail d'analyse collaboratif et le partage de la connaissance de la situation entre les analystes.

Les architectures logicielles existantes pour les applications multi-tactiles ont été explorées et une version multi-tactile de Google Earth a été implémentée. Elle peut être utilisée aussi bien sur une table multi-tactile qu'avec un affichage mural à l'aide de gestes captés par le senseur Kinect de Microsoft. Un portail d'éveil situationnel maritime incluant différentes maquettes d'outils d'analyse de navire d'intérêt a été développé afin d'être affiché sur le mur d'écrans. Une application simple a également été développée pour permettre l'affichage de fenêtres d'information sur les navires qui peuvent être orientées de différentes manières. Cette exploration des technologies avancées d'interaction humain-machine a permis d'acquérir des connaissances sur la façon d'utiliser avantageusement ces innovations dans un contexte de surveillance maritime ainsi que pour d'autres activités dans le domaine du renseignement.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

advanced technologies; human-computer interaction; large group display; knowledge wall; gesture interaction; multi-touch; surface computing; vessel of interest; maritime analysis; collaboration; maritime domain awareness; situation analysis

Defence R&D Canada

Canada's Leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

www.drdc-rddc.gc.ca