Copy No. _____

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

DEFENCE R&D DÉFENSE

Exploring Operator Interface Design Concepts Using Adobe Flash

Design Concepts and Prototype Documentation

Carrie McMullin
Don Coady

Defence R&D Canada – Atlantic

Technical Memorandum
DRDC Atlantic TM 2009-007
May 2009

Canada

This page intentionally left blank.

Exploring Operator Interface Design Concepts Using Adobe Flash

Design Concepts and Prototype Documentation

Carrie McMullin
Don Coady

Defence R&D Canada – Atlantic

Technical Memorandum
DRDC Atlantic TM 2009-007
May 2009

Principal Author

Original signed by Carrie McMullin

Carrie McMullin

Approved by

Original signed by Francine Desharnais

Francine Desharnais

Head, Maritime Information Combat Systems Section

Approved for release by

Original signed by Ron Kuwahara for

Calvin Hyatt

Chair, DRDC Atlantic Document Review Panel

Abstract

The Shipboard Command and Control Group at DRDC Atlantic has previously investigated several decision support tools aimed at improving the contact recognition and identification processes performed by Naval Task Group operators. This document describes the results of using Adobe Flash CS3 Professional as an interface prototyping tool to examine more closely a subset of the proposed interface concepts.

Flash was used to develop two interface demonstrations which provided a tangible means for exploring and evaluating the 'look and feel' of interface design options. The "Track Data Block and Display Concepts" demonstration explored real-time design concepts such as a track data block in the form of a mouse hover activated 'details on demand' feature (tooltips), tailored right-click context menus, variable-hour track history (breadcrumbs) for contacts, interactive breadcrumb track information, and track future information (dead reckoning). The Bearing Only (BO) Overlay demonstration showed the usefulness of temporal review capabilities, especially when using an electronic warfare overlay highlighting scheme to assist in associating BO tracks with radar tracks.

The extensive graphic, animation, and interactive design capabilities provided by Flash made it an excellent rapid prototyping tool. Flash symbols, components, and ActionScript code further enhanced its flexibility and reusability as a design and development platform.

Résumé

Le Groupe des systèmes de commandement et de contrôle embarqués de RDDC Atlantique a déjà examiné plusieurs outils de soutien à la décision afin d'améliorer les processus de reconnaissance et d'identification des contacts effectués par les opérateurs appartenant au Groupe opérationnel naval. Ce document présente les résultats de l'utilisation d'Adobe Flash CS3 Professional comme outil de prototypage d'interface afin d'examiner plus en détails un sous-ensemble des principes proposés pour les interfaces.

Flash a été utilisé afin d'élaborer deux démonstrations d'interface qui constituent des moyens tangibles pour explorer et évaluer la présentation et le comportement des options de conception d'interface. La démonstration « Principes de blocs et d'affichage de données de piste » a exploré des principes de conception en temps réel, comme le principe de bloc de données de piste » au moyen de « l'affichage de détails sur demande » activé par le passage de la souris (info-bulles), menus contextuels adaptés, historique de piste à variation horaire (pistes de navigation) pour les

contacts, information sur les pistes par piste de navigation interactive et information sur le futur des pistes (valeur estimée). La démonstration Calque de relèvements seuls (BO) a montré l'utilité des capacités d'examen chronologique, en particulier lorsqu'on applique une technique de mise en surbrillance de calque de guerre électronique pour aider à effectuer l'association de pistes en relèvements avec des pistes radar.

Les capacités étendues de Flash dans les domaines du graphisme, de l'animation et de la conception interactive en fond un excellent outil pour effectuer du prototypage rapide. Les symboles et composants de Flash ainsi que son code ActionScript ajoutent à sa souplesse et à la réutilisabilité de ses produits comme plateforme de conception et de développement.

Executive summary

Exploring Operator Interface Design Concepts Using Adobe Flash: Design Concepts and Prototype Documentation

Carrie McMullin; Don Coady; DRDC Atlantic TM 2009-007; Defence R&D Canada – Atlantic; May 2009.

Background

The Shipboard Command and Control Group of DRDC-Atlantic previously investigated several decision support tools aimed at improving the contact recognition and identification processes performed by Naval Task Group operators. Based on various cognitive analyses of the work operators engage in while performing these processes and on subject matter experts' (SMEs) feedback, a number of interface design concepts were proposed that could ease the cognitive workload on operators and subsequently aid in compiling and maintaining the maritime tactical picture.

Results

Two interactive software demonstrations were developed, using Adobe Flash, to explore several proposed interface design concepts. The Track Data Block (TDB) and Display Concepts demonstration implements 'Track Data Blocks' (a mouse hover activated details-on-demand feature), variable-hour track history (breadcrumbs), interactive breadcrumb track information, and track future information (dead reckoning). The Bearing Only (BO) Overlay demonstration adds a BO overlay highlighting scheme to assist in associating BO tracks with radar tracks. Both demonstrations incorporate interactive temporal review capabilities using both conventional playback controls and temporal scrubbing via direct contact manipulation.

Significance

The interface demos provide a tangible means for exploring and evaluating the 'look and feel' of interface design options. Additionally, these interactive display prototypes serve as a mechanism for soliciting end user feedback as the design further evolves. The extensive graphic, animation, and interactive design capabilities provided by Flash make it an excellent rapid prototyping tool for any type of user interface, regardless of the application domain.

Future plans

Creating prototypes of interface design concepts is only one of the many steps towards their actual implementation. The next logical step would be to involve SMEs in evaluating the concepts to determine which elements are the most useful and worthwhile. The demonstrations developed in this work would be the starting point for that further research.

Sommaire

Exploration de principes de conception d'interfaces opérateur au moyen d'Adobe Flash : Principes de conception et documentation du prototype

Carrie McMullin; Don Coady; DRDC Atlantic TM 2009-007; R & D pour la défense Canada – Atlantique; Mai 2009.

Introduction ou contexte: Le Groupe des systèmes de commandement et de contrôle embarqués de RDDC - Atlantique a déjà examiné plusieurs outils de soutien à la décision afin d'améliorer les processus de reconnaissance et d'identification des contacts effectués par les opérateurs appartenant au Groupe opérationnel naval. En se fondant sur diverses analyses cognitives portant sur le travail effectué par les opérateurs effectuant ces processus et sur des commentaires d'experts, divers principes de conception d'interface ont été proposés afin d'alléger la charge de travail cognitif imposée aux opérateurs et éventuellement d'aider à compiler et à tenir à jour le tableau tactique de la situation maritime.

Résultats: Deux démonstrations logicielles interactives ont été élaborées au moyen d'Adobe Flash afin d'explorer plusieurs principes de conception d'interface qui ont été proposés. La démonstration « Principes de blocs et d'affichage de données de piste » met en oeuvre le principe de « bloc de données de piste » (affichage de détails sur demande activé par le passage de la souris), l'historique de piste à variation horaire (pistes de navigation), l'information sur les pistes par piste de navigation interactive et l'information sur le futur des pistes (valeur estimée). La démonstration Calque de relèvements seuls (BO) appliquait une technique de mise en surbrillance de calque composé seulement de relèvements pour aider à effectuer l'association de pistes en relèvements seulement avec des pistes radar. Les deux démonstrations intégraient des capacités d'examen temporel rétroactif interactif offrant des commandes de lecture classique ainsi que le « brossage » temporel par manipulation par contact direct.

Importance: Les deux démonstrations d'interface constituent des moyens tangibles pour explorer et évaluer la présentation et le comportement des options de conception d'interface. De plus, ces prototypes d'affichage interactif servent de mécanisme utilisé pour obtenir la réaction des utilisateurs finals à mesure que la conception évolue. Les capacités étendues de Flash dans les domaines du graphisme, de l'animation et de la conception interactive en fond un excellent outil pour effectuer le prototypage rapide de tout type d'interface utilisateur, quel que soit son domaine d'application.

Perspectives: La création de prototypes fondés sur des concepts d'interface n'est qu'une seule de nombreuses étapes en vue de leur mise en œuvre réelle. L'étape logique suivante consisterait à faire appel à des experts en la matière afin de déterminer les éléments les plus utiles et les plus valables. Les démonstrations élaborées au cours de ces travaux constitueraient le point de départ de recherches ultérieures.

Table of contents

List of figures

List of tables

Acknowledgements

Many thanks to Bruce Chalmers of DRDC Atlantic, co-supervisor of this project, for providing advice and feedback as the work was planned and performed.

This page intentionally left blank.

1 Introduction and Background

The operator machine interface (OMI) design concepts discussed in this document are derived from prior DRDC research work [1,2] aimed at improving decision support tools in the HALIFAX Class operations room, with particular emphasis on enhancing the contact recognition and identification processes operators engage in to build and maintain a tactical picture. Many of the proposed cognitive support concepts subsequently manifest as design modifications to the operator's tactical interface. More specifically, these modifications focus on the operator's Tactical Situation Area (TSA) display; a 2D overhead (map) view of all air, surface, and sub-surface contacts being tracked by ownship.

Utilizing custom scenarios, demonstrations of the interface design concepts were developed that provided a tangible means for exploring and evaluating the 'look and feel' of the various interface design options. The interface design concepts are presented in two demonstrations: the Track Data Block and Display Concepts demonstration, and the Bearing Only (BO) Overlay with personal video recorder (PVR) Capabilities demonstration.

The design concepts implemented in the first demo include details-on-demand (Tooltips: Section 2.1.3), temporal data visualization (Track History: Section 2.1.1), and temporal data interaction (Scrubber: Section 2.1.6). These interface enhancements are intended to address the lack of explicit temporal visualization tools available in current operator displays. More specifically, the proposed modifications enable operators to view and compare data for multiple contacts at both current and prior points in time. Ultimately, the additional display capabilities are designed to assist operators in determining the identity and intent of ownship's contacts.

The second demonstration implements a design option for BO track association (Bearing Only Overlay Demonstration: Section 2.2), as well as the temporal data interaction seen in the first demonstration. The prime motivation for these display features was to aid operators in associating radar tracks (position based) with BO tracks (direction based) by providing improved visualization options for the latter. Having more fused tracks results in an overall simpler and better quality tactical picture for the operator to manage.

The tool used to create these interactive demonstrations is Adobe Flash CS3 Professional, which runs the demonstrations using Flash Player v9.0 and utilizes ActionScript 3.0, the Flash object-oriented programming language [3]. The development model used to complete this project is described in detail in Annex A, while the code and code documentation is located in Annex B.

Flash allows the author to combine timeline animation and programmable actions to create an interactive visual movie. Most often, Flash is used in the creation of websites and online games. Animations can be drawn wholly on the timeline, programmed using ActionScript, or a combination of the two can be used which is the case in the presented demonstrations. The extensive graphic, animation, and interactive design capabilities provided by Flash make it an excellent rapid prototyping tool. Flash symbols, components, and ActionScript code further enhance the flexibility and reusability of the Flash platform. All these features combined make Flash a highly versatile tool well suited for the design and development of any type of user interface, regardless of the information domain.

2 Two Interactive Demonstrations

2.1 Track Data Block and Display Concepts Demonstration

The first of the two demonstrations, the Track Data Block and Display Concepts Demonstration, explores many concepts, including details-on-demand, temporal data visualization, and temporal data interaction. The scenario data in this demonstration is completely fictional, and was contrived to merely illustrate the interface design concepts. Figure 1 shows the layout of the demonstration interface.

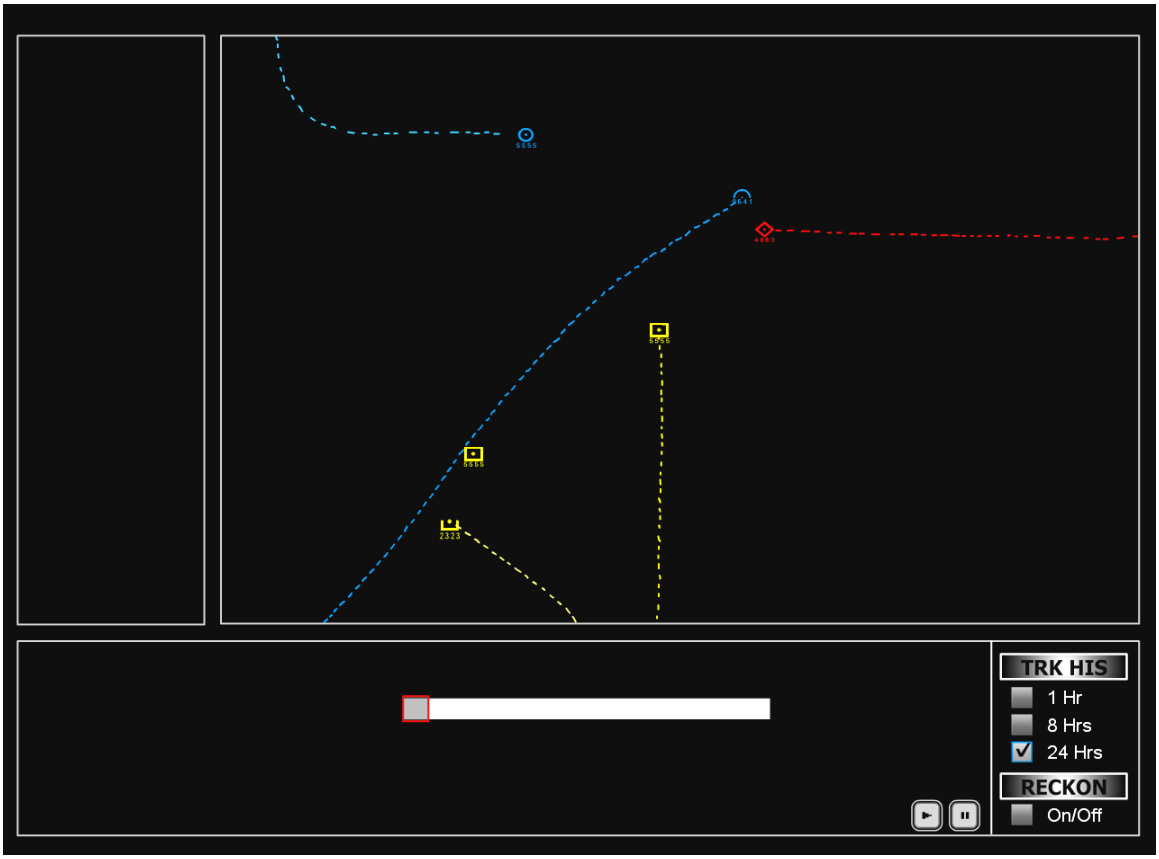

Figure 1: Track Data Block and Display Concepts demonstration interface. Track histories for all contacts are visible for time period: 24 Hours

2.1.1 Track History

One of the first interface features to be implemented was the idea of a “track history” (Figure 1), i.e., a trail of time stamps visually displaying the path of the contact. These track histories are displayed as a broken line, or a path of “breadcrumbs” in the same colour range as that of the contact, i.e., if a contact is “friendly” it is coloured blue, so the track history issued from the contact would also be blue, thusly a red track history would be for hostile contacts, and a yellow track history for unknown contacts.

Track histories can be displayed in one of two ways; all track histories for all contacts become visible if controlled by the universal toggle boxes located in the lower right-hand corner of the demonstration, or, each track history for each contact can be toggled individually by activating/deactivating the track history through the right-click context menu of the contact. This method is further discussed in section 3.1.2. Figure 1 shows the toggle boxes for three options to display the track histories, 1 hour of time, 8 hours, or 24 hours. These time periods were chosen merely for demonstration purposes, and specific time designations for use in real-world applications need to be further researched. Further consideration must also be taken if a feature was created that allowed the operator to change a contact’s status, i.e. from unknown to hostile, if the track history would reflect the colour change of the contact, or if would remain the same. This was discussed informally, and was not introduced into the demonstration as further research must be conducted to determine the value of such an option.

2.1.2 Context Menus

Most users are familiar with context menus as the box of options that appears when a user right-clicks on a digital file, image, etc. These context menus are customizable in Flash, and are activated when a user right-clicks on any part of the flash movie. In this demonstration, custom context menus (Figure 2) become available when the user right-clicks on a contact on the screen. The context menu displays options to: turn on/off track histories, and turn on/off dead reckoning on those contact symbols where these features are available. One difficulty discovered in activating the context menu for a contact exists when the contact is moving quickly. In this case, it is challenging to maintain the mouse over top of the contact for a long enough period to right-click and activate the context menu.

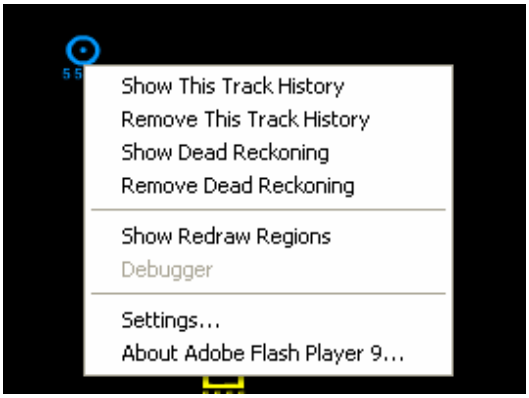

Figure 2: A custom context menu providing options to the user for controlling the track history and dead reckoning for a specific contact.

2.1.3 Tooltips

The concept of tooltips for use in a tactical setting was derived from the CONOPS Report [1]. In the CONOPS report, the idea of a Track Data Block (TDB) (Figure 3) was examined as a way for operators to gain more information about a contact without their eyes leaving the Tactical Situation Area (TSA) within their tactical display. The track data block would be launched after the operator hovers the cursor over a contact for a short time, e.g., 100 msec. The TDB also allows direct simultaneous comparison of contact data between the selected contact ("hooked contact") and any other desired contact. In this demonstration, five possible ways of accessing and displaying the track data block are examined. These different ways are illustrated below with various examples of contacts. At the end of Section 2.1.3.5, Table 1 offers a comparison of the pros and cons of five ways of accessing track data blocks (Sections 2.1.3.1 – 2.1.3.5).

2.1.3.1 Contact 1: Friendly Surface Contact

In this case, when the operator hovers the cursor over the contact symbol, the tooltip appears above, and slightly to the right of the cursor. As long as the cursor is held over the contact, the tooltip will continue to appear, and move, with the cursor. If the cursor is removed from the contact, the tooltip is no longer visible. The style of the tooltip is similar to that of the track data block described in [1], a rectangle grid containing critical data on a contact (shown in Figure 3).

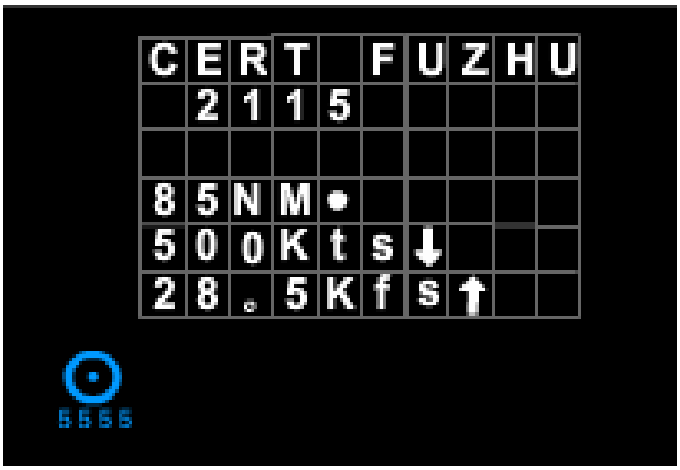

Figure 3: The tooltip that corresponds to the friendly surface contact, the tooltip appears above and to the right of the active contact.

2.1.3.2 Contact 2: Friendly Air Contact

This approach is similar the previous one in that it is also activated by hovering the cursor over the contact, and the tooltip moves relative to the position of the cursor (Figure 4). However, this tooltip differs in two respects. First, there was concern that if the operator's control screen is very busy, or, multiple contacts are close together, it will be difficult to determine to which contact the tooltip belongs. To resolve this, this tooltip displays a line which extends from the lower-left corner of the tooltip to the centre of the contact. This line is white in colour, as white is not often used on the TSA. Second, the tooltip itself uses the colour of the contact as the colour of the text within the tooltip. This is an additional means to attempt to associate the tooltip with its parent contact.

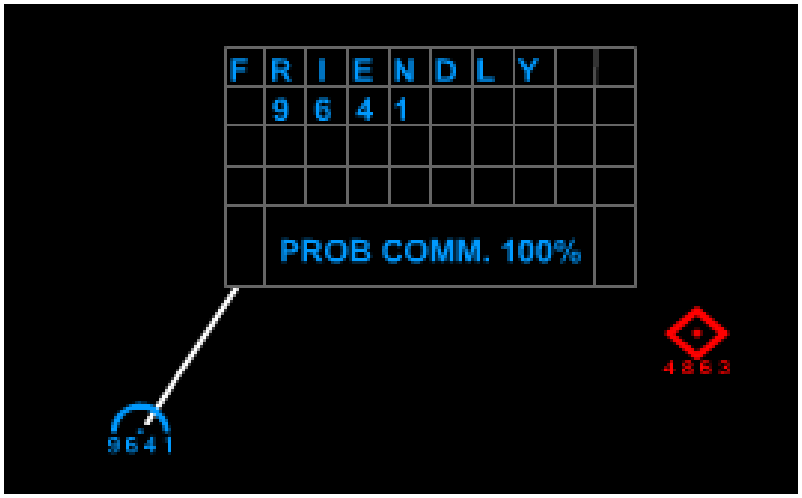

Figure 4: The tooltip that corresponds to the friendly air contact. Like the tooltip in Figure 3, the tooltip appears above and to the right of the active contact, only it has a line that connects the contact to the tooltip.

2.1.3.3 Contact 3: Hostile Surface Contact

The tooltip for this contact is also made visible by the operator hovering over the contact (Figure 5). However, if the operator is to remove the mouse from the contact, the tooltip will remain visible for a preset period of time, and will no longer be visible once the amount of time programmed has passed. This tooltip is also different in the respect that it appears in the same position each time, regardless of the location of the contact. In order to determine which contact the tooltip belongs to, a white line also runs from the lower left of the tooltip to the centre of the contact. This tooltip also employs an additional feature, a scroll-bar that is located in the bottom of the tooltip. This scroll bar can be programmed to display any message, but for the purposes of the demonstration a warning is issued. This scrolling text is also a hyperlink, meaning that upon clicking the scrolling text a new window is opened up to display any additional information pertinent to the contact.

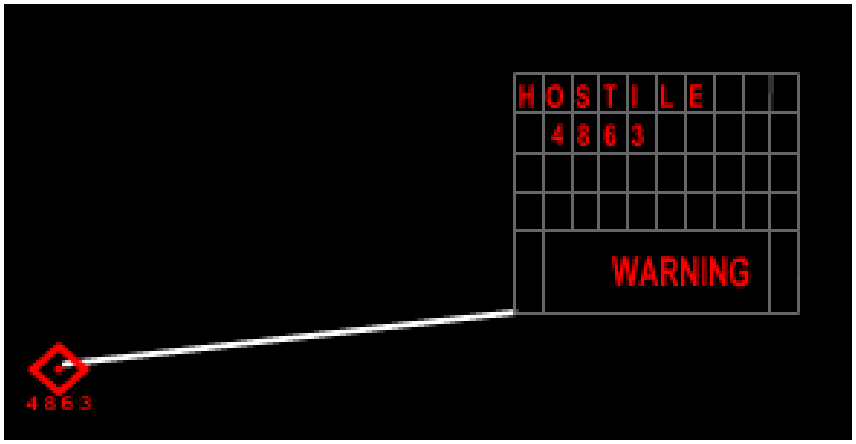

Figure 5: The tooltip that corresponds to the hostile surface contact. It remains in one location for a pre-determined length of time with a line connecting the tooltip to the active contact.

2.1.3.4 Contact 4: Unknown Subsurface Contact

This tooltip is made visible in the same manner as the first and second tooltips, by the operator hovering the cursor over the contact symbol. In this case, the tooltip appears above and to the right of the contact, and retains its position relative to the position of the contact. In order to signify which contact this tooltip corresponds to, when the cursor is over the contact, a glow filter in the same colour as the contact appears both on the contact and around the perimeter of the tooltip (Figure 6). This technique also illustrates the importance of keeping the tooltip translucent as the situation where other contacts may be travelling behind the tooltip is important to consider. The glow around the tooltip is also made semi-transparent, to allow for the situation where an additional contact is located behind the glow.

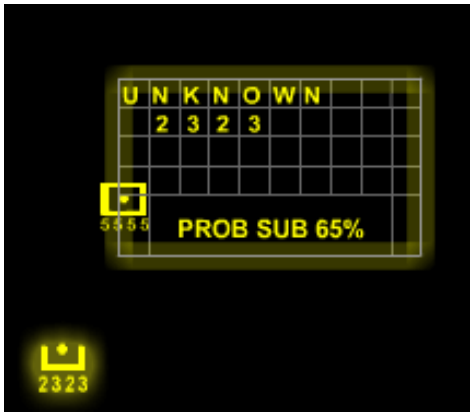

Figure 6: The tooltip that corresponds to the unknown subsurface contact. The tooltip appears above and to the right of the active contact with a glow that matched the tooltip to its' relative contact.

2.1.3.5 Contact 5: Unknown Surface Contact

This tooltip is also made visible by moving the mouse over the contact which begins a timer, keeping the tooltip visible for a programmed amount of time. The tooltip contains an additional feature; the ability to “pin” the tooltip to the screen (Figure 7). When the tooltip is visible, the operator clicks on the pin symbol located in the upper right corner of the tooltip, the tooltip becomes permanently visible. The tooltip appears in a position relative to the contact, though when the operator activates the “pinning” feature, the tooltip stays in a fixed location as the contact moves on. If the tooltip were to be reactivated at a later time, the tooltip would again appear in a location relative to the contact, and stay pinned in that location if that feature is activated. This feature is useful in the case where the operator needs to compare the data in a set TDBs on different contacts by keeping them simultaneously visible. When the operator clicks on the pin symbol again, the tooltip is no longer visible.

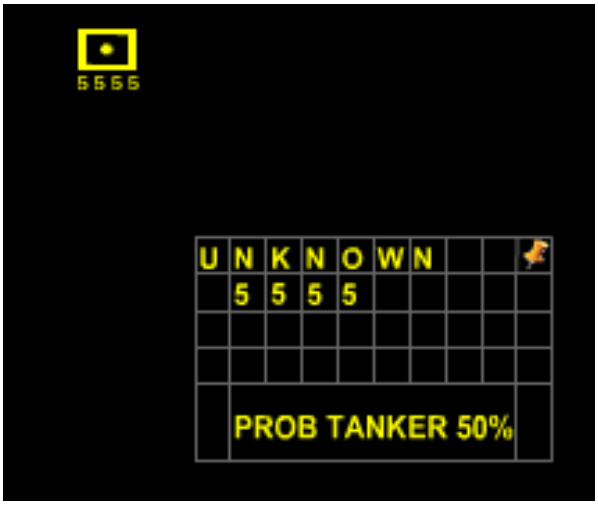

Figure 7: The tooltip that corresponds to the unknown surface contact. The tooltip appears in a relative location to the active contact, though remains stationary once ‘pinned’ by the operator.

Table 1: A comparison between the Pros and Cons of the designed Track Data Blocks styles

Pros and Cons of the Five Discussed TDB Styles		
	Pros	**Cons**
Contact 1 (Figure 3)	- Display duration is controlled by the operator	- Difficult to maintain for small, buried, or fast moving contacts. - Association between the TDB and the contact can be lost
Contact 2 (Figure 4)	- The line connecting the TDB and the contact guarantees a known association between the two - The colouring of the contact's vital information shown in the TDB provides more of a connection with the contact than the white text	- Line connecting the TDB to the contact can clutter the screen if contact is far away from the fixed location - Difficult to maintain for small, buried, or fast moving contacts.
Contact 3 (Figure 5)	- Easier to remain visible for fast moving or congested contacts - Fixed location means the operator always knows where to look for invoked details	- Fixed location means the TDB can lose proximity and association with the contact - Line connecting the TDB to the contact can clutter the screen if contact is far away from the fixed location
Contact 4 (Figure 6)	- Proximity between contact and TDB is maintained automatically - No unnecessary lines to clutter the screen - Glow intuitively connects the contact to the TDB	- Though the TDB and glow are translucent, in a very busy scenario other contacts could become lost in the busyness
Contact 5 (Figure 7)	- Allows for the operator to compare multiple TDBs at one time	- If all TDBs are pinnable, too many active TDBs could clutter the screen - As TDB is in a fixed location, losing the association to the contact is a possibility

2.1.4 Breadcrumb Time Recognition

As discussed earlier in the track history section, one additional desired feature of a contact's track history would be to recall information about the contact from any past time stamp. Within the demonstration, this feature becomes available at time = 300 (Figure 8), denoted by the darker coloured breadcrumb within the track history of the friendly surface contact. Upon clicking this breadcrumb, the breadcrumbs at time 300 of all other contacts become more visible by use of a contact-relevant glow filter applied to each of the breadcrumbs. Also, upon clicking the aforementioned breadcrumb, a tooltip relaying the information related to the contact at that time is displayed. This tooltip will stay visible until the breadcrumb is clicked once more. Both the friendly air and unknown subsurface contacts' track histories contain tooltips. The friendly air tooltip is accessed by hovering the cursor overtop the breadcrumb, while the unknown surface tooltip becomes visible when the cursor is moved over the breadcrumb, and will stay visible for an arbitrarily programmed amount of time. These three tooltips have been put in place to show the different ways tooltips can be accessed and read. To determine which tooltip format is best, further work outside the scope of this project would be required.

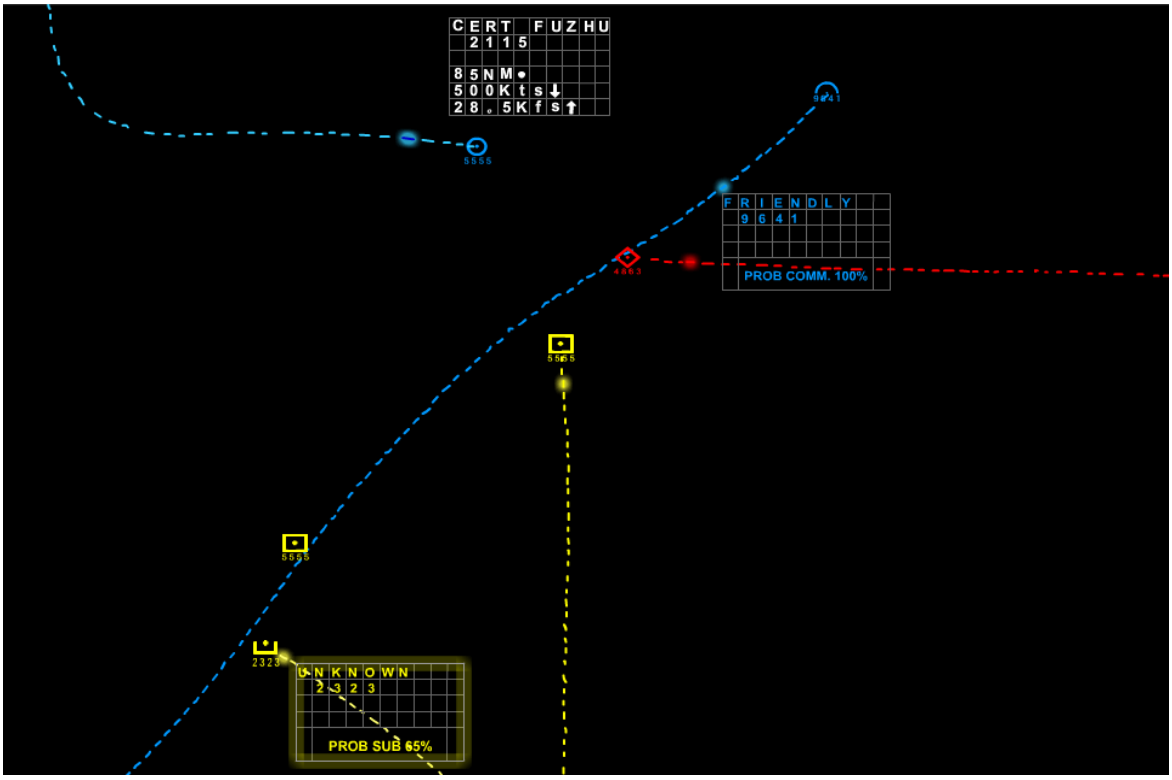

Figure 8: Breadcrumbs are highlighted at Time = 300, possible tooltip methods for breadcrumb information visible.

2.1.5 Dead Reckoning

The term “Dead Reckoning” refers to the process of estimating one’s current position based on previously determined position, and advancing that position based upon known or estimated speed over elapsed time and course [4]. By no means is dead reckoning exact, however, within this demonstration two styles are presented to help the operator visualize this idea. The dead reckoning can be made visible/invisible by two means: globally controlling all dead reckoning through the toggle checkbox located in the right lower corner of the demonstration, or, by right-clicking on either the friendly surface contact or the hostile surface contact and activating/deactivating the dead reckoning through the context menus of those contacts.

2.1.5.1 Style 1

The first style explored in the demonstration is simply a straight line projecting from the centre of the friendly surface contact to the edge of the screen in the direction the contact is headed (Figure 9). In order to differentiate the dead reckon from the track history trail, a fully connected, semi-transparent line is used instead of using a broken line, the colour of the dead reckon line reflects the colour of the contact. The path of the contact is determined by simply extending the line ahead of the contact along its current heading.

Figure 9: Style 1 of dead reckoning issuing from the friendly surface contact: a single, semi-transparent line, the colour of the active contact.

2.1.5.2 Style 2

The second style explored in the demonstration is a cone shape filled by a fading gradient (Figure 10). The apex of the cone extends from the centre of the hostile surface contact and extends to roughly half the screen. Similar to the first style, it extends in the direction the contact is heading. The gradient colouring within the cone represents the probability of what course the contact will be travelling, based on its current trajectory. The denser the gradient fill, the more likely the contact will enter that point in its course. Although this style is interesting, further work would be needed to develop a more concrete and workable design proposal.

Figure 10: Style 2 of dead reckoning issuing from the hostile surface contact showing a probabilistic gradient of possible future tracks.

2.1.6 Scrubber Personal Video Recorder (PVR)

The playback control feature in the demonstration is the moveable playback head (referred to here as the “thumb”) located at the bottom of the screen (Figure 11). This feature is intrinsically different from the previous features described. The main difference is that all of the previous features are meant to be used while the operator is looking at the main display screen, and no other screen is necessary. However, the scrubber PVR feature is meant as a review feature, similar to a rewind button for a movie. This feature would allow the operator to review the positions of contacts in the desired time frame. In this demonstration, as soon as the operator moves the contacts back in time, the current positions of the contacts are no longer on the screen. To develop this concept further, additional design steps would have to be taken to allow for contacts to continue forward as planned in demonstration time while the playback was used to illustrate the review feature of the scrubber PVR.

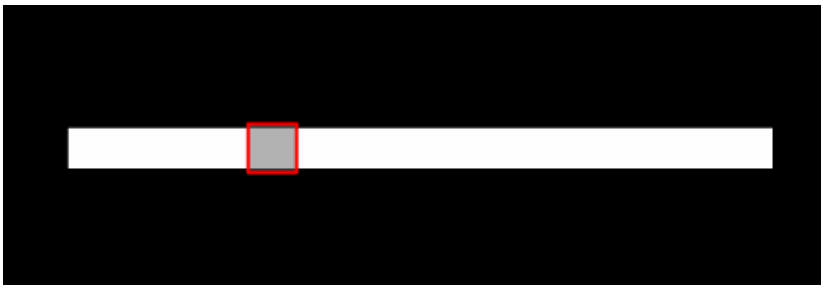

Figure 11: The scrubber bar with moveable thumb

2.1.7 Direct Contact Manipulation

This feature is similar to the scrubber PVR in that this feature is also meant for reviewing the past positions of contacts. However, instead of controlling the playback through the scrubber, playback is controlled by clicking on the desired contact and simply dragging it back in time (direct object manipulation). When the operator is reviewing the contact's course, all of the other contacts on the screen move back in time as well. This provides the operator with the information needed to analyze the locations of all contacts at a given point in time. Within the demonstration, this can be controlled by clicking and dragging on the friendly surface contact and dragging. Only this contact is enabled with this feature for demonstration purposes.

2.2 Bearing Only Overlay Demonstration

This second demonstration addresses the confusion an Electronic Warfare (EW) overlay can cause if there is no feature that allows for the contacts to be animated from their previous positions. EW bearing lines have an inherent ambiguity as it can be difficult to tell which radar contact belongs to the overlay if multiple contacts are within the overlay area. By providing the operator with a feature that allows for past events to be viewed, potential critical contact recognition information can be presented. The EW overlay uses a filled cone shape with its apex anchored to ownship to represent bearing only contacts. The animation begins with the three contacts moving across the screen, with their final positions clustered together at the top of the screen (Figure 12). The EW overlay covers two of the contacts, so at the end of the animation it is impossible to tell which contact the EW applies to. By using either the scrubber or grabbing the unknown surface contact on the right side of the screen, the operator can go back in time to determine to which contact the EW relates. The point of the demonstration is to show how by giving the operator the ability to review past track history information, in some cases it may be possible to resolve problems in ambiguous associations between EW bearing lines and radar contacts.

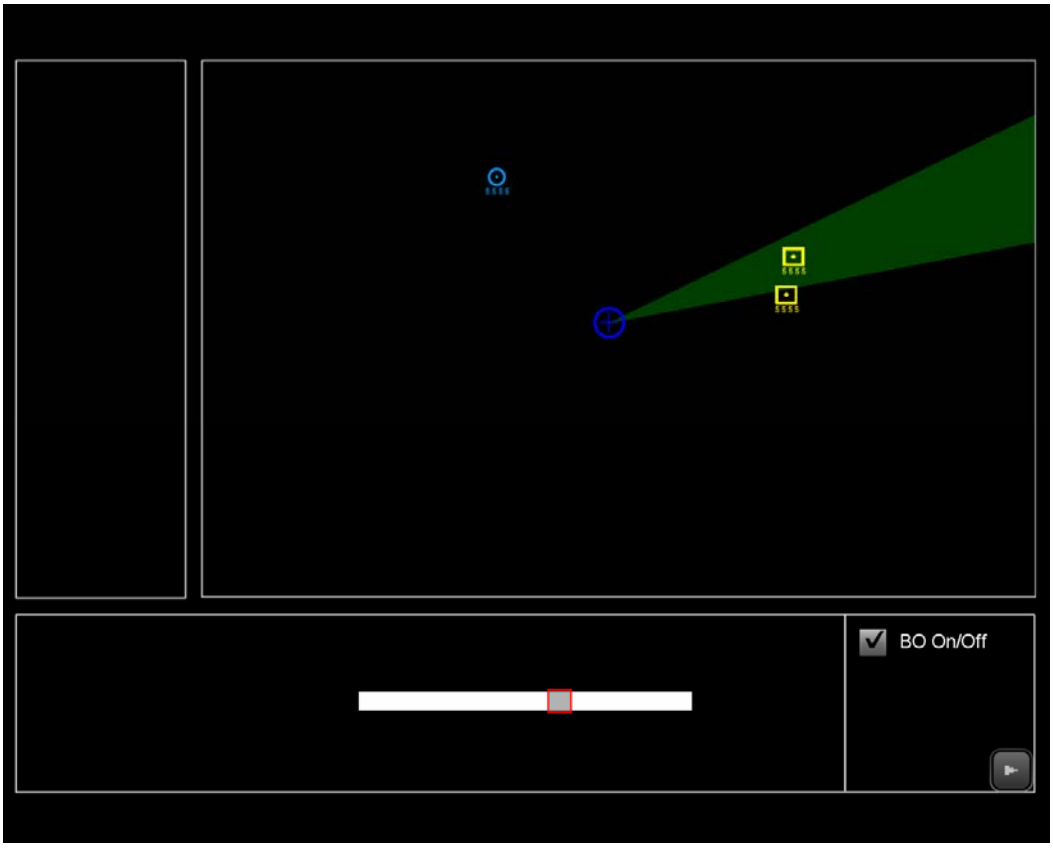

Figure 12: Bearing Only Overlay demonstration with the BO turned on

2.2.1 On/Off Checkbox

A toggle box is located in the lower right of the screen which enables the operator to turn on and off the EW overlay in the demonstration.

2.2.2 Scrubber PVR

The scrubber PVR in this demonstration is used in the same way as the scrubber PVR in the previously discussed demonstration.

2.2.3 Direct Contact Manipulation

The moveable contact feature in this demonstration is also the same as the contact grabbing in the previously discussed demonstration. However, in this demonstration it is the unknown surface contact, located on the right side of the screen that is the active contact.

3 Conclusions and Future Work

The interface design concepts presented in this document are only a few of the candidate features that could be considered for implementation in the HALIFAX Class frigates' shipboard Control and Command system [2].

Creating a visual representation of interface design concepts is an important step towards the actual implementation of the designs. The next logical step in the process would be to have actual operators use the demonstration to determine from their expertise which elements and (styles of elements) are the most useful and worthwhile. For example, human-in-the-loop experiments which incorporate realistic data and scenarios should be conducted to determine which style of tooltip is the most readable, informative, and the least distracting from the remainder of the screen. Once the style of the tooltip is determined, one could attempt to determine how it should be activated to appear on the tactical display. These are only the first of many questions to be asked and answered. While providing answers to such critical design questions is beyond the scope of this project, it is hoped that was has been completed in these demonstrations will be a jumping off point for research to come.

In regards to the design program, the Flash authoring environment works well as an exploratory interface design and development tool. It allows for easy creation of complicated images through an extensive set of drawing options, but involves a more complicated process when using code to draw. Animation can be pre-designed using the timeline and/or programmed at runtime using ActionScript code. Events can be triggered programmatically or by user-interaction such as mouse events. Flash symbols, components, and ActionScript code further enhance the flexibility and reusability of the Flash platform. Overall, Flash's graphic, animation, and interactive design capabilities make it an excellent rapid prototyping tool for any style of user interface, regardless of the application domain.

References

[1] Keeble, A.R., Matthews, M.L., Lamoureux, T.M., Berger, N., and Zobarich, R. (2007). Concept of Operations (CONOPS) for Recognition, Standard Identification and Track Management and Recommendations for Interface Design for an MSDF Enhanced System. (DRDC Toronto CR 2007-056). Humansystems Incorporated. (Limited Distribution - Controlled Goods).

[2] Moore, T., Torenvliet, G., Coates, C., Chalmers, B. (2008). Collaborative Workspace Requirements for Tactical Picture Compilation in a Maritime Task Group: SME Evaluation. Unpublished presentation, presented 7-8 May 2008 at DRDC Atlantic.

[3] Adobe, (2009). “Adobe Flash CS4 Professional”. Accessed Online on: April 16, 2009 from: http://www.adobe.com/products/flash/

[4] Wikipedia, (2009). “Dead Reckoning”. Accessed Online on: April 16, 2009 from: http://en.wikipedia.org/wiki/Dead_reckoning

This page intentionally left blank.

Annex A Development Model

A.1 Requirements

The Flash source files[1] (.fla) used to create the TDB and EW overlay demonstrations are named whole_demo3.fla and EW Demo.fla respectively. The source files were created (authored) and compiled (published) using Adobe Flash CS3 Professional[2]. Adobe Flash CS3 professional (or newer) is required to view and/or edit the Flash source files (.fla).

The Flash runtime files (.swf) files were executed and tested using the Adobe Flash Stand-alone Player (version 9.0) included with Adobe Flash CS3 Professional. The compiled runtime files are also available in HTML format (for the Flash Player web browser plug-in), Windows .exe, and Macintosh (.APP). The latter two formats simply bundle the Flash stand-alone player with the .swf runtime files (in the same manner as self executing zip files), hence there is no pre-requisite for the end user to have the Flash player. Regardless of output format, all the runtime files retain the same filename as their corresponding source file, save the filename extension (Figure A-1).

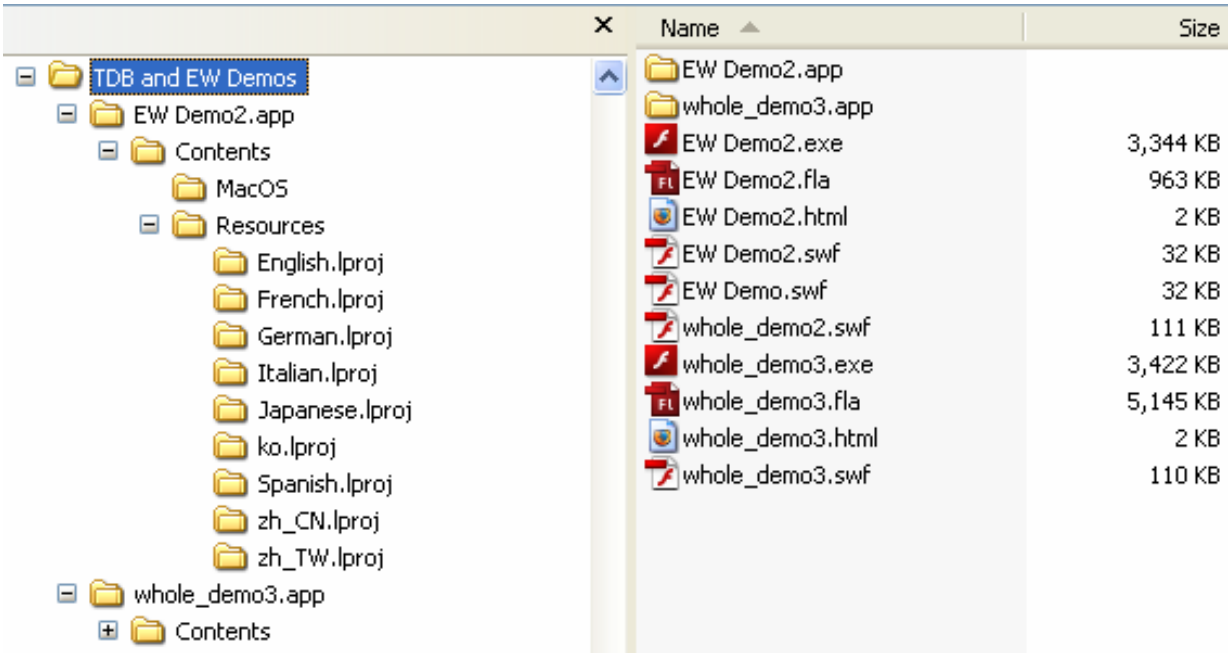

Figure A-13: Flash source and runtime file listing.

[1] The Flash runtime and/or source files associated with the TDB and EW Overlay demos described in this document are available via email request to Don.Coady@drdc-rddc.gc.ca.
[2] Adobe Flash CS3 Professional is available for purchase and/or free trial at https://www.adobe.com.

A.2 Operation

To launch either format of the files, the user can either double-click with the left mouse button on the runtime file, or single-click with the right mouse button to open the context menu for the file. The user simple moves the cursor to the option labelled “Open”.

In the Track Data Block and Display Concepts Demonstration, ‘Play’ and ‘Pause’ buttons are located in the lower right hand of the screen. These buttons can be used to pause and restart the animation. In the Bearing Only (BO) Overlay Demonstration, a ‘Play’ button is located in the lower right of the screen. By pushing this button, the animation will play of its own accord, as the animation is originally set to be manipulated by the user, and not to automatically play.

The animations were designed to be played individually on a single display screen. The boxed background for the animation is meant to simulate the TSA, though is not realistically accurate.

A.3 Design

Adobe Flash CS3 Professional was the software used to compile the interactive demonstrations; the Track Data Block and Display Concepts Demonstration, and the Bearing Only (BO) Overlay Demonstration. All source objects and code are contained within the respective files. All ActionScript 3.0 coding is embedded within the timeline of the animations; meaning, there are no external ActionScript files required to execute the demonstrations. A full documentation of the code, including layer and frame number can be found in Annex B.

The table below (Table A-2) illustrates what features of the Track Data Block and Display Concepts Demonstration become available at what frame number. The BO Overlay demonstration is not discussed because all features within that demonstration are available from the first frame.

Table A-2: A guide for what features occur at what frame for the Track Data Block and Display Concepts Demonstration

Frame Number	Feature
1	- Track history – 1 Hr/ 8 Hrs/ 24 Hrs - Tooltips 1, 2, 3, 4, 5 - Dead Reckoning – Style 1, Style 2 - Context menus - Scrubber - Direct Contact Manipulation
300	- Breadcrumb Time Recognition - Breadcrumb Tooltips – 1, 2
400	- Unknown Air contact is launched from the Unknown surface contact - Trail 6

Within the Track Data Block and Display Concepts Demonstration, there are two main locations where the majority of the code and layers on the timeline are located. The outer location, which is the immediate images on the stage that appear when the file is open, and the inner main movie clip, which can be accessed by double clicking almost anywhere on the stage.

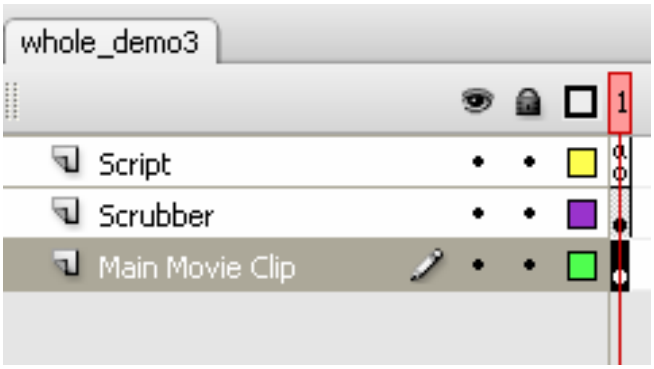

Figure A-14: The layers on the main stage of the Track data Block and Display Concepts Demonstration

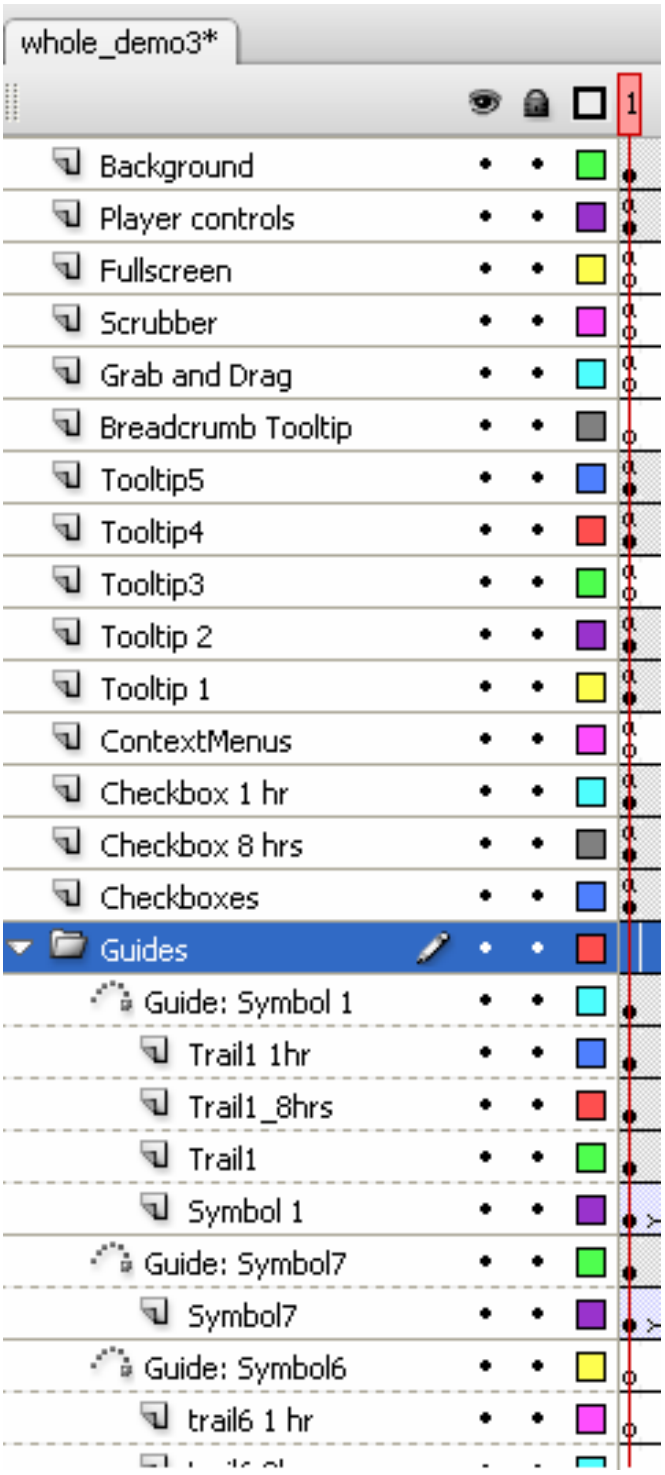

Figure A-15: The layers within the Main Movie Clip layer of the Track data Block and Display Concepts Demonstration. Additional layers exist in the 'Guides' folder which are not visible in this image

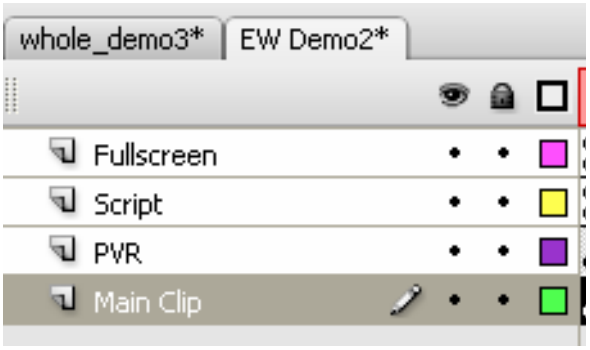

Figure A-16: The layers on the main stage of the Bearing Only Overlay Demonstration

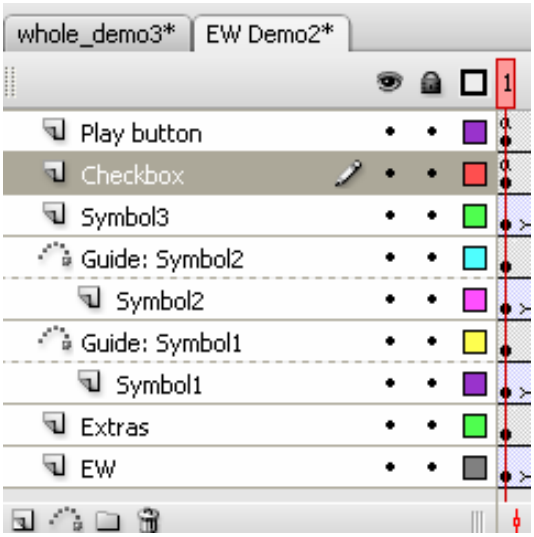

Figure A-17: The layers within the Main Clip layer of the Bearing Only Overlay Demonstration

The libraries within the two demonstrations contain symbols and components that were created to be used as elements of the demonstrations. Some symbols are common to both demonstrations, specifically the NTDS symbols used as contacts in the two demonstrations. The libraries are organized in a logical hierarchy, with related symbols grouped together under folders.

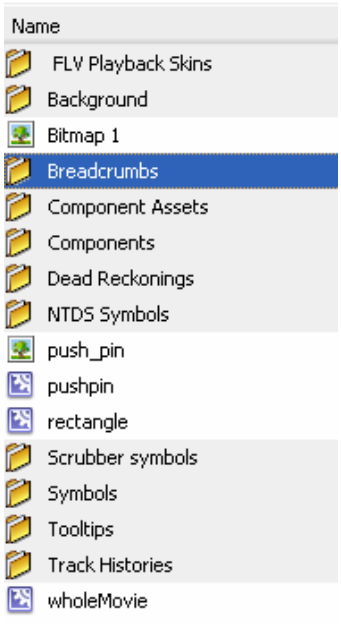

Figure A-18: The library of symbols in the Track Data Block and Display Concepts Demonstration

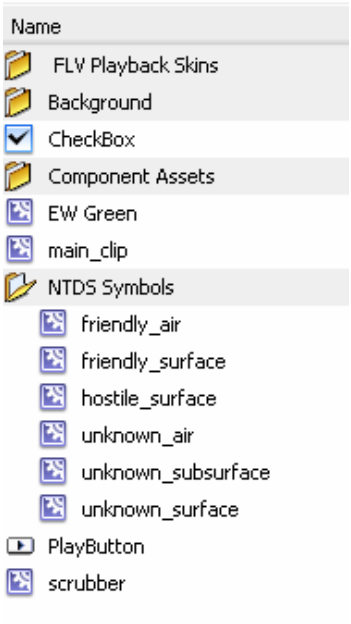

Figure A-19: The library of symbols in the Bearing Only Overlay Demonstration

Annex B Code Documentation

File:	Whole_demo3
Layer:	Script
Frame:	1

```
mcScrubber.mcTarget = mcClip;
mcScrubber.targetTrail1 = mcClip.trail1;
mcScrubber.targetTrail2 = mcClip.trail2;
mcScrubber.targetTrail3 = mcClip.trail3;
mcScrubber.targetTrail4 = mcClip.trail4;
mcScrubber.targetTrail5 = mcClip.trail5;
mcScrubber.targetTrail6 = mcClip.trail6;
mcScrubber.targetTrail1_8hr = mcClip.trail1_8hrs;
mcScrubber.targetTrail2_8hr = mcClip.trail2_8hrs;
mcScrubber.targetTrail3_8hr = mcClip.trail3_8hrs;
mcScrubber.targetTrail4_8hr = mcClip.trail4_8hrs;
mcScrubber.targetTrail5_8hr = mcClip.trail5_8hrs;
mcScrubber.targetTrail6_8hr = mcClip.trail6_8hrs;
mcScrubber.targetTrail1_1hr = mcClip.trail1_1hr;
mcScrubber.targetTrail2_1hr = mcClip.trail2_1hr;
mcScrubber.targetTrail3_1hr = mcClip.trail3_1hr;
mcScrubber.targetTrail4_1hr = mcClip.trail4_1hr;
mcScrubber.targetTrail5_1hr = mcClip.trail5_1hr;
mcScrubber.targetTrail6_1hr = mcClip.trail6_1hr;

/*************************************************************
*  ActionScript3 drag along a path by D Coady based on:
*
*          Peter Hall 2001
*
*          www.peterjoel.com
*          info@peterjoel.co.uk
*
*************************************************************/

mcClip.symbol1.addEventListener(MouseEvent.MOUSE_DOWN, myMouseDown);
mcClip.symbol1.addEventListener(MouseEvent.MOUSE_OVER, myMouseOver);
stage.addEventListener(MouseEvent.MOUSE_UP, myMouseUp);
stage.addEventListener(Event.ENTER_FRAME, enterFrameHandler);

function enterFrameHandler(event:Event):void {
          mcScrubber.mcThumb.x = mcClip.currentFrame/L*mcScrubber.mcSlide.width;
          removeEventListener("enterFrame", enterFrameHandler);
}
var mousePos:Point = new Point();

// calculate all positions of MC before we start
// create an array of all MC positions along the tween
```

```
var L:int = mcClip.totalFrames;
var pos:Array = new Array(L+1);

for(var i:int=1; i<L+1; i++){
	mcClip.gotoAndStop(i);
	pos[i] = new Point(mcClip.symbol1.x + mcClip.symbol1.width/2, mcClip.symbol1.y +
	mcClip.symbol1.height/2);
}

mcClip.gotoAndPlay(1);

//parameter to determine how far along the tween (in frames)
//the function will check to see if it is closer to the mouse
var speed:int = 20;

function myMouseDown(e:MouseEvent):void {
	stage.addEventListener(MouseEvent.MOUSE_MOVE, myMouseMove);
}

function myMouseOver(e:MouseEvent):void {
	mcClip.symbol1.buttonMode = true;
}

function myMouseMove(e:MouseEvent):void{
	setNewPos(e);
	setThumbPos(e);
	e.updateAfterEvent();
}

function myMouseUp(e:MouseEvent):void {
	stage.removeEventListener(MouseEvent.MOUSE_MOVE, myMouseMove);
}


// set position of MC to nearest to mouse
function setNewPos(e:MouseEvent){
	if(e.target != null){
		mousePos.x = stage.mouseX - 275;
		mousePos.y = stage.mouseY - 203;
		mcClip.gotoAndPlay(nearestFrame(mousePos)); // continue auto playback mode
	}
}

// calculate frame in the given range in which
// the MC is closest to the mouse
var currFrame:int;
var dir:int;
var lastFrame:int;
var frameCtr:int;

function nearestFrame(mousePos:Point):int {
	// get local references for extra speed
	currFrame = mcClip.currentFrame;
```

```
	//var pos = this.pos;
	dir = 0;
	// find out which direction along the curve takes the MC closer to the mouse position
	if(currFrame == L){
		dir = -1;
	}else if(currFrame == 1){
		dir = 1;
	}else if(dist(pos[currFrame+1], mousePos) < dist(pos[currFrame], mousePos)){
		dir = 1; // forwards
	}else if(dist(pos[currFrame-1], mousePos)< dist(pos[currFrame], mousePos)){
		dir = -1; // backwards
	}else{
		return currFrame; // stay in the same position
	}

	lastFrame = currFrame + speed*dir;
	if (lastFrame + dir > L)lastFrame = L;
	if (lastFrame + dir < 1)lastFrame = 1;
	for(frameCtr = currFrame; frameCtr != lastFrame; frameCtr += dir){
		if(dist(pos[frameCtr], mousePos) < dist(pos[frameCtr+dir], mousePos)){
			return frameCtr; // return minimum
		}
	}
	// didn't find a minimum, so return closest within the speed limit
	return frameCtr;
}

function dist(position:Point, mousePos:Point):Number{
	return Point.distance(position, mousePos);
}

function setThumbPos(e:MouseEvent):void{
	mcScrubber.mcThumb.x = mcClip.currentFrame/L*mcScrubber.mcSlide.width;
}
```

File:	Whole_demo3
Layer:	Scrubber -> actions
Frame:	1

```
//AS3 MovieClip scrubber control by D Coady

var minX:Number = mcSlide.x;
var maxX:Number = mcSlide.x + mcSlide.width -mcThumb.width;
var percent:Number;

var mcTarget:MovieClip;  //the target movieclip to be controlled
var targetTrail1:MovieClip;
var targetTrail2:MovieClip;
var targetTrail3:MovieClip;
var targetTrail4:MovieClip;
var targetTrail5:MovieClip;
```

```
var targetTrail6:MovieClip;
var targetTrail1_8hr:MovieClip;
var targetTrail2_8hr:MovieClip;
var targetTrail3_8hr:MovieClip;
var targetTrail4_8hr:MovieClip;
var targetTrail5_8hr:MovieClip;
var targetTrail6_8hr:MovieClip;
var targetTrail1_1hr:MovieClip;
var targetTrail2_1hr:MovieClip;
var targetTrail3_1hr:MovieClip;
var targetTrail4_1hr:MovieClip;
var targetTrail5_1hr:MovieClip;
var targetTrail6_1hr:MovieClip;
var targetBC_1:MovieClip;

mcThumb.buttonMode = true;

mcThumb.addEventListener(MouseEvent.MOUSE_DOWN, myMouseDown);
stage.addEventListener(MouseEvent.MOUSE_UP, myMouseUp);

function myMouseDown(e:MouseEvent):void {
	this.addEventListener(MouseEvent.MOUSE_MOVE, myMouseMove);
}

function myMouseMove(e:MouseEvent):void {
	if (mouseX - mcThumb.width/2 < minX){    //Left lower scrub limit
		mcThumb.x = minX;
	}else if(mouseX - mcThumb.width/2 > maxX){ //Right maximum scrub limit
		mcThumb.x = maxX;
	}else{
		mcThumb.x = this.mouseX - mcThumb.width/2;
	}
	percent = mcThumb.x/(maxX-minX);
	this.mcTarget.gotoAndStop(Math.floor(percent*this.mcTarget.totalFrames)+1);
	this.targetTrail1.gotoAndStop(Math.floor(percent*this.targetTrail1.totalFrames)+1);
	this.targetTrail2.gotoAndStop(Math.floor(percent*this.targetTrail2.totalFrames)+1);
	this.targetTrail3.gotoAndStop(Math.floor(percent*this.targetTrail3.totalFrames)+1);
	this.targetTrail4.gotoAndStop(Math.floor(percent*this.targetTrail4.totalFrames)+1);
	this.targetTrail5.gotoAndStop(Math.floor(percent*this.targetTrail5.totalFrames)+1);
	this.targetTrail1_8hr.gotoAndStop(Math.floor(percent*this.targetTrail1_8hr.totalFrames)+1);
	this.targetTrail2_8hr.gotoAndStop(Math.floor(percent*this.targetTrail2_8hr.totalFrames)+1);
	this.targetTrail3_8hr.gotoAndStop(Math.floor(percent*this.targetTrail3_8hr.totalFrames)+1);
	this.targetTrail4_8hr.gotoAndStop(Math.floor(percent*this.targetTrail4_8hr.totalFrames)+1);
	this.targetTrail5_8hr.gotoAndStop(Math.floor(percent*this.targetTrail5_8hr.totalFrames)+1);
	this.targetTrail1_1hr.gotoAndStop(Math.floor(percent*this.targetTrail1_1hr.totalFrames)+1);
	this.targetTrail2_1hr.gotoAndStop(Math.floor(percent*this.targetTrail2_1hr.totalFrames)+1);
	this.targetTrail3_1hr.gotoAndStop(Math.floor(percent*this.targetTrail3_1hr.totalFrames)+1);
	this.targetTrail4_1hr.gotoAndStop(Math.floor(percent*this.targetTrail4_1hr.totalFrames)+1);
	this.targetTrail5_1hr.gotoAndStop(Math.floor(percent*this.targetTrail5_1hr.totalFrames)+1);

	if (currentFrame >= 400){
		this.targetTrail6.gotoAndStop(Math.floor(percent*this.targetTrail6.totalFrames)+1);
	this.targetTrail6_8hr.gotoAndStop(Math.floor(percent*this.targetTrail6_8hr.totalFrames)+1);
```

```
        this.targetTrail6_1hr.gotoAndStop(Math.floor(percent*this.targetTrail6_1hr.totalFrames)+1);
        }

        e.updateAfterEvent();
}

function myMouseUp(e:MouseEvent):void {
        this.removeEventListener(MouseEvent.MOUSE_MOVE, myMouseMove);
}
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Trail 2
Frame:	300

```
bc_clickable2.addEventListener(MouseEvent.MOUSE_OVER, mouse_over);
bc_clickable2.addEventListener(MouseEvent.MOUSE_OUT, mouse_out);

function mouse_over(e:Event):void{
        tooltip2.visible = true;
        bc_clickable2.addEventListener(Event.ENTER_FRAME, frame);
}

function frame(e:Event):void{
        tooltip2.x = bc_clickable2.x + 30;
        tooltip2.y = bc_clickable2.y + 20;
}

function mouse_out(e:Event):void{
        tooltip2.visible = false;
        bc_clickable2.removeEventListener(Event.ENTER_FRAME, frame);
}
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Trail 4
Frame:	300

```
var time:Timer = new Timer(100, 15);

tooltip4.width = 60;
tooltip4.height = 35;
tooltip4.visible = false;

bc_clickable4.addEventListener(MouseEvent.MOUSE_OVER, start_timer);

function start_timer(e:Event):void{
        time.addEventListener(TimerEvent.TIMER_COMPLETE, end_timer);
        time.start();
        tooltip4.visible = true;
        tooltip4.x = bc_clickable4.x + 35;
        tooltip4.y = bc_clickable4.y + 20;
}
```

```
function end_timer(e:Event):void{
	tooltip4.visible = false;
	time.reset();
}
```

File	Whole_demo3
Layer:	Main Movie Clip -> Checkboxes
Frame:	1

```
//Sets the text style for the checkbox
var myTextFormat:TextFormat = new TextFormat();
myTextFormat.color = 0xFFFFFF;
myTextFormat.size = 9;
cb1.setStyle("textFormat", myTextFormat);
cb3.setStyle("textFormat", myTextFormat);
cb4.setStyle("textFormat", myTextFormat);

dummytrail.visible = true;
trail1.visible = false;
trail2.visible = false;
trail3.visible = false;
trail4.visible = false;
trail5.visible = false;

var trailArray:Array = new Array();
trailArray.push(trail1);
trailArray.push(trail2);
trailArray.push(trail3);
trailArray.push(trail4);
trailArray.push(trail5);

reckon1.visible = false;
reckon3.visible = false;

var reckonArray:Array = new Array();
reckonArray.push(reckon1);
reckonArray.push(reckon3);

//Initiating the Trail On/Off checkbox
cb1.addEventListener(MouseEvent.CLICK, changeTrailState);
cb4.addEventListener(MouseEvent.CLICK, changeReckonState);

function changeTrailState(event:MouseEvent):void{
	if (dummytrail.visible == true){
		var i:int;
		for(i = 0; i < trailArray.length; i++){
			trailArray[i].visible = true;
			dummytrail.visible = false;
		}
	}
	else {
```

```
                var m:int;
                for (m = 0; m < trailArray.length; m++){
                        trailArray[m].visible = false;
                        dummytrail.visible = true;
                }
        }
}


function changeReckonState(e:Event):void{
        if (reckon1.visible == false){
                var k:int;
                for(k = 0; k < reckonArray.length; k++){
                        reckonArray[k].visible = true;
                }
        }
        else{
                var n:int;
                for(n = 0; n < reckonArray.length; n++){
                        reckonArray[n].visible = false;
                }
        }
}
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Checkboxes
Frame:	300

```
bc_clickable.visible = false;
bc_clickable2.visible = false;
bc_clickable3.visible = false;
bc_clickable4.visible = false;
bc_clickable5.visible = false;

trailArray.push(bc_clickable);
trailArray.push(bc_clickable2);
trailArray.push(bc_clickable3);
trailArray.push(bc_clickable4);
trailArray.push(bc_clickable5);
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Checkboxes
Frame:	400

```
trail6.visible = false;
trailArray.push(trail6);
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Checkbox 8 hrs
Frame:	1

```
var myFormat:TextFormat = new TextFormat();
myFormat.color = 0xFFFFFF;
myFormat.size = 9;
cb2.setStyle("textFormat", myFormat);

trail1_8hrs.visible = false;
trail2_8hrs.visible = false;
trail3_8hrs.visible = false;
trail4_8hrs.visible = false;
trail5_8hrs.visible = false;

var eightHoursArray:Array = new Array();
eightHoursArray.push(trail1_8hrs);
eightHoursArray.push(trail2_8hrs);
eightHoursArray.push(trail3_8hrs);
eightHoursArray.push(trail4_8hrs);
eightHoursArray.push(trail5_8hrs);

cb2.addEventListener(MouseEvent.CLICK, changeEightState);

function changeEightState(e:MouseEvent):void{

        if (trail1_8hrs.visible == false){
                var i:int;
                for(i = 0; i < eightHoursArray.length; i++){
                        eightHoursArray[i].visible = true;
                }
        }
        else if (trail1_8hrs.visible == true){
                var m:int;
                for (m = 0; m < eightHoursArray.length; m++){
                        eightHoursArray[m].visible = false;
                }
        }
}
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Checkbox 8 hrs
Frame:	300

```
bc_clickable22.visible = false;

eightHoursArray.push(bc_clickable);
eightHoursArray.push(bc_clickable22);
eightHoursArray.push(bc_clickable3);
eightHoursArray.push(bc_clickable4);
eightHoursArray.push(bc_clickable5);
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Checkbox 8 hrs
Frame:	400

```
trail6_8hrs.visible = false;
eightHoursArray.push(trail6_8hrs);
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Checkbox 8 hrs
Frame:	525

```
eightHoursArray.splice(5, 5);
bc_clickable22.visible = false;
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Checkbox 1 hr
Frame:	1

```
trail1_1hr.visible = false;
trail2_1hr.visible = false;
trail3_1hr.visible = false;
trail4_1hr.visible = false;
trail5_1hr.visible = false;

var oneHourArray:Array = new Array();
oneHourArray.push(trail1_1hr);
oneHourArray.push(trail2_1hr);
oneHourArray.push(trail3_1hr);
oneHourArray.push(trail4_1hr);
oneHourArray.push(trail5_1hr);

cb3.addEventListener(MouseEvent.CLICK, changeOneState);

function changeOneState(event:MouseEvent):void{
	if (trail1_1hr.visible == false ){
		var g:int;
		for(g = 0; g < oneHourArray.length; g++){
			oneHourArray[g].visible = true;
		}
	}
	else if(trail1_1hr.visible == true){
		var h:int;
		for (h = 0; h < oneHourArray.length; h++){
			oneHourArray[h].visible = false;
		}
	}
}
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Checkbox 1 hr
Frame:	300

```
oneHourArray.push(bc_clickable);
oneHourArray.push(bc_clickable22);
oneHourArray.push(bc_clickable3);
```

```
oneHourArray.push(bc_clickable4);
oneHourArray.push(bc_clickable5);
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Checkbox 1 hr
Frame:	330

```
oneHourArray.splice(5,5);
bc_clickable22.visible = false;
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Checkbox 1 hr
Frame:	400

```
trail6_1hr.visible = false;
oneHourArray.push(trail6_1hr);
```

File:	Whole_demo3
Layer:	Main Movie Clip -> ContextMenus
Frame:	1

```
trail1.visible = false;
trail2.visible = false;
trail3.visible = false;
trail4.visible = false;
trail5.visible = false;

var tArray:Array = new Array();
tArray.push(trail1);
tArray.push(trail2);
tArray.push(trail3);
tArray.push(trail4);
tArray.push(trail5);

var menuItem1:ContextMenuItem = new ContextMenuItem("Show This Track History");
var menuItem11:ContextMenuItem = new ContextMenuItem("Remove This Track History");
var menuItem111:ContextMenuItem = new ContextMenuItem("Show Dead Reckoning");
var menuItem1111:ContextMenuItem = new ContextMenuItem("Remove Dead Reckoning");
var menuItem2:ContextMenuItem = new ContextMenuItem("Show This Track History");
var menuItem22:ContextMenuItem = new ContextMenuItem("Remove This Track History");
var menuItem3:ContextMenuItem = new ContextMenuItem("Show This Track History");
var menuItem33:ContextMenuItem = new ContextMenuItem("Remove This Track History");
var menuItem333:ContextMenuItem = new ContextMenuItem("Show Dead Reckoning");
var menuItem3333:ContextMenuItem = new ContextMenuItem("Remove Dead Reckoning");
var menuItem4:ContextMenuItem = new ContextMenuItem("Show This Track History");
var menuItem44:ContextMenuItem = new ContextMenuItem("Remove This Track History");
var menuItem5:ContextMenuItem = new ContextMenuItem("Show This Track History");
var menuItem55:ContextMenuItem = new ContextMenuItem("Remove This Track History");

var customMenu1:ContextMenu = new ContextMenu();
var customMenu2:ContextMenu = new ContextMenu();
```

```
var customMenu3:ContextMenu = new ContextMenu();
var customMenu4:ContextMenu = new ContextMenu();
var customMenu5:ContextMenu = new ContextMenu();

menuItem1.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, showTrail);
menuItem11.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, hideTrail);
menuItem111.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, showReckon);
menuItem1111.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, hideReckon);
menuItem2.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, showTrail2);
menuItem22.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, hideTrail2);
menuItem3.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, showTrail3);
menuItem33.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, hideTrail3);
menuItem333.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, showReckon3);
menuItem3333.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, hideReckon3);
menuItem4.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, showTrail4);
menuItem44.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, hideTrail4);
menuItem5.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, showTrail5);
menuItem55.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, hideTrail5);

customMenu1.hideBuiltInItems();
customMenu2.hideBuiltInItems();
customMenu3.hideBuiltInItems();
customMenu4.hideBuiltInItems();
customMenu5.hideBuiltInItems();

customMenu1.customItems.push(menuItem1, menuItem11, menuItem111, menuItem1111);
customMenu2.customItems.push(menuItem2, menuItem22);
customMenu3.customItems.push(menuItem3, menuItem33, menuItem333, menuItem3333);
customMenu4.customItems.push(menuItem4, menuItem44);
customMenu5.customItems.push(menuItem5, menuItem55);

symbol1.contextMenu = customMenu1;
symbol2.contextMenu = customMenu2;
symbol3.contextMenu = customMenu3;
symbol4.contextMenu = customMenu4;
symbol5.contextMenu = customMenu5;

function showTrail(event:ContextMenuEvent):void{
        if (currentFrame < 300){
                if (tArray[0].visible == false){
                        tArray[0].visible = true;
                }
        }
        else if (currentFrame >= 300){
                if (tArray[0].visible == false){
                        tArray[0].visible = true;
                        tArray[5].visible = true;
                }
        }
}

function hideTrail(event:ContextMenuEvent):void{
        if (currentFrame < 300){
```

```
                if (tArray[0].visible == true){
                        tArray[0].visible = false;
                }
        }
        else if (currentFrame >= 300){
                if (tArray[0].visible == true){
                        tArray[0].visible = false;
                        tArray[5].visible = false;
                }
        }
}

function hideReckon(e:Event):void{
        if (reckon1.visible == true){
                reckon1.visible = false;
        }
}

function showReckon(e:Event):void{
        if (reckon1.visible == false){
                reckon1.visible = true;
        }
}


function showTrail2(event:ContextMenuEvent):void{
        if (currentFrame < 300){
                if (tArray[1].visible == false){
                        tArray[1].visible = true;
                }
        }
        else if (currentFrame >= 300){
                if (tArray[1].visible == false){
                        tArray[1].visible = true;
                        tArray[6].visible = true;
                }
        }
}

function hideTrail2(event:ContextMenuEvent):void{
        if (currentFrame < 300){
                if (tArray[1].visible == true){
                        tArray[1].visible = false;
                }
        }
        else if (currentFrame >= 300){
                if (tArray[1].visible == true){
                        tArray[1].visible = false;
                        tArray[6].visible = false;
                }
        }
}
```

```
function showTrail3(event:ContextMenuEvent):void{
	if (currentFrame < 300){
		if (tArray[2].visible == false){
			tArray[2].visible = true;
		}
	}
	else if (currentFrame >= 300){
		if (tArray[2].visible == false){
			tArray[2].visible = true;
			tArray[7].visible = true;
		}
	}
}

function hideTrail3(event:ContextMenuEvent):void{
	if (currentFrame < 300){
		if (tArray[2].visible == true){
			tArray[2].visible = false;
		}
	}
	else if (currentFrame >= 300){
		if (tArray[2].visible == true){
			tArray[2].visible = false;
			tArray[7].visible = false;
		}
	}
}

function hideReckon3(e:Event):void{
	if (reckon3.visible == true)
	{
		reckon3.visible = false;
	}
}

function showReckon3(e:Event):void{
	if (reckon3.visible == false)
	{
		reckon3.visible = true;
	}
}

function showTrail4(event:ContextMenuEvent):void{
	if (currentFrame < 300){
		if (tArray[3].visible == false){
			tArray[3].visible = true;
		}
	}
	else if (currentFrame >= 300){
		if (tArray[3].visible == false){
			tArray[3].visible = true;
			tArray[8].visible = true;
```

```
                }
        }
}

function hideTrail4(event:ContextMenuEvent):void{
        if (currentFrame < 300){
                if (tArray[3].visible == true){
                        tArray[3].visible = false;
                }
        }
        else if (currentFrame >= 300){
                if (tArray[3].visible == true){
                        tArray[3].visible = false;
                        tArray[8].visible = false;
                }
        }
}

function showTrail5(event:ContextMenuEvent):void{
        if (currentFrame < 300){
                if (tArray[4].visible == false){
                        tArray[4].visible = true;
                }
        }
        else if (currentFrame >= 300){
                if (tArray[4].visible == false){
                        tArray[4].visible = true;
                        tArray[9].visible = true;
                }
        }
}

function hideTrail5(event:ContextMenuEvent):void{
        if (currentFrame < 300){
                if (tArray[4].visible == true){
                        tArray[4].visible = false;
                }
        }
        else if (currentFrame >= 300){
                if (tArray[4].visible == true){
                        tArray[4].visible = false;
                        tArray[9].visible = false;
                }
        }
}
```

File:	Whole_demo3
Layer:	Main Movie Clip -> ContextMenus
Frame:	300

```
tArray.push(bc_clickable);
tArray.push(bc_clickable2);
```

```
tArray.push(bc_clickable3);
tArray.push(bc_clickable4);
tArray.push(bc_clickable5);

if (currentFrame < 300){
	bc_clickable.visible = false;
	bc_clickable2.visible = false;
	bc_clickable3.visible = false;
	bc_clickable4.visible = false;
	bc_clickable5.visible = false;
}
```

File:	Whole_demo3
Layer:	Main Movie Clip -> ContextMenus
Frame:	400

```
trail6.visible = false;
tArray.push(trail6);

var menuItem6:ContextMenuItem = new ContextMenuItem("Show This Track History");
var menuItem66:ContextMenuItem = new ContextMenuItem("Remove This Track History");
var customMenu6:ContextMenu = new ContextMenu();

menuItem6.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, showTrail6);
menuItem66.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, hideTrail6);

customMenu6.hideBuiltInItems();
customMenu6.customItems.push(menuItem6, menuItem66);

symbol6.contextMenu = customMenu6;

function showTrail6(event:ContextMenuEvent):void{
	if (currentFrame < 300){
		if (tArray[10].visible == false){
			tArray[10].visible = true;
		}
	}
	else if (currentFrame >= 300){
		if (tArray[10].visible == false){
			tArray[10].visible = true;
		}
	}
}

function hideTrail6(event:ContextMenuEvent):void{
	if (currentFrame < 300){
		if (tArray[10].visible == true){
			tArray[10].visible = false;
		}
	}
	else if (currentFrame >= 300){
		if (tArray[10].visible == true){
```

```
                        tArray[10].visible = false;
                }
        }
}
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Tooltip 1
Frame:	1

```
//*****************************************************************************
// Tooltip 1 - TDB
//
// This tooltip is activated by a MOUSE_OVER and MOUSE_OUT event. It has no
// link to symbol1 other than the relation of its location to the mouse
// x/y positions.
//*****************************************************************************

tooltip.visible = false;
tooltip.width = 60;
tooltip.height = 35;

symbol1.addEventListener(MouseEvent.MOUSE_OVER, up);
symbol1.addEventListener(MouseEvent.MOUSE_OUT, down);

function up(evt:Event):void {
        tooltip.visible = true;
        symbol1.addEventListener(Event.ENTER_FRAME, ref);
}

function ref(evt:Event):void {
        tooltip.y = mouseY - 25;
        tooltip.x = mouseX + 35;
}

function down(evt:Event):void {
        symbol1.removeEventListener(Event.ENTER_FRAME, ref);
        tooltip.visible = false;
}
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Tooltip 2
Frame:	1

```
//*****************************************************************************
// Tooltip 2 - TDB
//
// This tooltip hovers at a position related to the mouse. It also has a line
// that connect the lower (x,y) of the tooltip to the + in the symbol (centre,
// bottom, or top, depending on the symbol).
//*****************************************************************************
```

```
//Sets the tooltip off the stage to invisible
tooltip2.visible = false;
tooltip2.width = 60;
tooltip2.height = 35;

//Controls the call to the functions dependant on the mouse state
symbol2.addEventListener(MouseEvent.MOUSE_OVER, up2);
symbol2.addEventListener(MouseEvent.MOUSE_OUT, down2);

//When the mouse is over the symbol, sets the tooltip to visible and calls the
//positioning function of the event listener
function up2(evt:Event):void {
	tooltip2.visible = true;
	line.visible = true;
	symbol2.addEventListener(Event.ENTER_FRAME, ref2);
	symbol2.addEventListener(Event.ENTER_FRAME, lineMove);
}

//Sets the tooltip's size and puts it at a location dependant on the mouse
function ref2(evt:Event):void {
	tooltip2.x = mouseX + 40;
	tooltip2.y = mouseY - 40;
	graphics.clear();
}

//Function for the mouse_Down event, makes the tooltip invisible again
function down2(evt:Event):void {
	tooltip2.visible = false;
	line.visible = false;
	graphics.clear();
	symbol2.removeEventListener(Event.ENTER_FRAME, lineMove);
	symbol2.removeEventListener(Event.ENTER_FRAME, ref2);
}

function lineMove(e:Event):void{
	graphics.lineStyle(1, 0xFFFFFF);
	graphics.moveTo((tooltip2.x - (tooltip2.width/2) + 2), tooltip2.y + (tooltip2.height/2));
	graphics.lineTo(symbol2.x + (symbol2.width/2), (symbol2.y + symbol2.height)-3);
}
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Tooltip 3
Frame:	1

```
//*****************************************************************************
// Tooltip 3
//*****************************************************************************

var timer:Timer = new Timer(100, 15);
var linetimer:Timer = new Timer(100, 15);
```

```
tooltip3.visible = false;
tooltip3.width = 60;
tooltip3.height = 35;

//Controls the call to the functions dependant on the mouse state
symbol3.addEventListener(MouseEvent.MOUSE_OVER, up3);

//When the mouse is over the symbol, sets the tooltip to visible and calls the
//positioning function of the event listener
function up3(evt:Event):void {
	timer.addEventListener(TimerEvent.TIMER, timedTip);
	linetimer.addEventListener(TimerEvent.TIMER, drawLine);
	timer.addEventListener(TimerEvent.TIMER_COMPLETE, done);
	timer.start();
	linetimer.start();
}

//Sets the tooltip's size and puts it at a location dependant on the mouse
function timedTip(evt:TimerEvent):void {
	tooltip3.visible = true;
	tooltip3.x = 200;
	tooltip3.y = -125;
	graphics.clear();
}

function drawLine(e:Event):void{
	graphics.lineStyle(1, 0xFFFFFF);
	graphics.moveTo((tooltip3.x - 21), tooltip3.y + (tooltip3.height/2));
	graphics.lineTo(symbol3.x + (symbol3.width/2), symbol3.y + (symbol3.height/2)-2);
}

function done(evt:TimerEvent):void{
	tooltip3.visible = false;
	timer.reset();
	linetimer.reset();
	graphics.clear();
	timer.removeEventListener(TimerEvent.TIMER, timedTip);
	linetimer.removeEventListener(TimerEvent.TIMER, drawLine);

}
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Tooltip 4
Frame:	1

```
var symglow:GlowFilter = new GlowFilter(0xF5F41B, .40, 6, 6, 8, 2, false, false);

tooltip4.width = 60;
tooltip4.height = 35;
tooltip4.visible = false;
```

```
symbol4_bg.mouseChildren = false;
symbol4_bg.mouseEnabled = false;

symbol4_bg.width = symbol4.width;
symbol4_bg.height = symbol4.height;

symbol4.addEventListener(MouseEvent.MOUSE_OVER, hoverUp);
symbol4.addEventListener(MouseEvent.MOUSE_OUT, hoverDown);

function hoverUp(e:Event):void{
        tooltip4.visible = true;
        symbol4.addEventListener(Event.ENTER_FRAME, tipMover);
}

//Sets the tooltip position relative to the mouse and follows the mouse during Mouse_over
function tipMover(e:Event):void{
        tooltip4.y = mouseY -40;
        tooltip4.x = mouseX + 40;

        symbol4_bg.visible = true;
        symbol4_bg.filters = [symglow];
        symbol4_bg.x = symbol4.x;
        symbol4_bg.y = symbol4.y;
}

function hoverDown(e:Event):void{
        tooltip4.visible = false;
        symbol4_bg.filters = [];
        symbol4.removeEventListener(Event.ENTER_FRAME, tipMover);
        symbol4_bg.visible = false;
}
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Tooltip 5
Frame:	1

```
//Sets the tooltip off the stage to invisible
tooltip6.visible = false;
tooltip6.width = 60;
tooltip6.height = 35;

var timed:Timer = new Timer(100, 15);

//Controls the call to the functions dependant on the mouse state
symbol5.addEventListener(MouseEvent.MOUSE_OVER, up5);

//When the mouse is over the symbol, sets the tooltip to visible and calls the
//positioning function of the event listener
function up5(evt:Event):void {
        tooltip6.pushpin.addEventListener(MouseEvent.CLICK, pinned);
        timed.addEventListener(TimerEvent.TIMER, ref5);
        timed.addEventListener(TimerEvent.TIMER_COMPLETE, down5);
```

```
		timed.start();
}

//Sets the tooltip's size and puts it at a location dependant on the mouse
function ref5(evt:Event):void {
		tooltip6.visible = true;
		tooltip6.x = 80;
		tooltip6.y = 10;
		graphics.clear();
}

//Function for the mouse_Down event, makes the tooltip invisible again
function down5(evt:Event):void {
		tooltip6.visible = false;
		timed.reset();
		graphics.clear();
		timed.removeEventListener(TimerEvent.TIMER, ref5);

}

function pinned(e:Event):void{
		timed.stop();
		tooltip6.visible = true;
		tooltip6.x = 80;
		tooltip6.y = 10;
		tooltip6.pushpin.addEventListener(MouseEvent.CLICK, unpin);
}

function unpin(e:Event):void{
		tooltip6.visible = false;
		tooltip6.pushpin.removeEventListener(MouseEvent.CLICK, pinned);
}
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Breadcrumb Tooltip
Frame:	300

```
var blue_glow:GlowFilter = new GlowFilter(0x33CCFF, .75, 4, 4, 10, 2, false, false);
var yellow_glow:GlowFilter = new GlowFilter(0xF7F53C, .75, 4, 4, 10, 2, false, false);
var red_glow:GlowFilter = new GlowFilter(0xDD0000, .75, 4, 4, 10, 2, false, false);

tooltip_bc.visible = false;
tooltip_bc.width = 60;
tooltip_bc.height = 35;

bc_clickable.addEventListener(MouseEvent.CLICK, up_bc);

function up_bc(evt:Event):void {
		if (tooltip_bc.visible == false){
			tooltip_bc.visible = true;
			tooltip_bc.x = 25;
			tooltip_bc.y = -170;
```

```
                bc_clickable.filters = [blue_glow];
                bc_clickable2.filters = [blue_glow];
                bc_clickable22.filters = [blue_glow];
                bc_clickable3.filters = [red_glow];
                bc_clickable4.filters = [yellow_glow];
                bc_clickable5.filters = [yellow_glow];
        }
        else{
                tooltip_bc.visible = false;
                bc_clickable.filters = [];
                bc_clickable2.filters = [];
                bc_clickable22.filters = [];
                bc_clickable3.filters = [];
                bc_clickable4.filters = [];
                bc_clickable5.filters = [];
        }
}
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Fullscreen
Frame:	1

```
stage.displayState = StageDisplayState.FULL_SCREEN;
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Player Controls
Frame:	1

```
pause_but.addEventListener(MouseEvent.CLICK, pause_event);
play_but.addEventListener(MouseEvent.CLICK, play_event);

function pause_event(e:Event):void{
        var i:uint;
        var j:uint;
        var k:uint;
        var l:uint;

        stop();
        for (i = 0; i < tArray.length; i++){
                tArray[i].stop();
        }
        for (j = 0; j < reckonArray.length; j++){
                reckonArray[j].stop();
        }
        for (k = 0; k < eightHoursArray.length; k++){
                eightHoursArray[k].stop();
        }
        for (l = 0; l < oneHourArray.length;l++){
                oneHourArray[l].stop();
        }
}
```

```
function play_event(e:Event):void{
	var m:uint;
	var n:uint;
	var p:uint;
	var q:uint;

	play();
	for (m = 0; m < tArray.length; m++){
		tArray[m].play();
	}
	for (n = 0; n < reckonArray.length; n++){
		reckonArray[n].play();
	}
	for (p = 0; p < eightHoursArray.length; p++){
		eightHoursArray[p].play();
	}
	for (q = 0; q < oneHourArray.length; q++){
		oneHourArray[q].play();
	}
}
```

File:	Whole_demo3
Layer:	Main Movie Clip -> Player Controls
Frame:	400

```
tArray.push(trail6);
eightHoursArray.push(trail6_8hrs);
oneHourArray.push(trail6_1hr);
```

File:	EW Demo2
Layer:	Script
Frame:	1

```
mcScrubber.mcTarget = mcClip;

/*****************************************************************
*  ActionScript3 drag along a path by D Coady based on:
*
*		Peter Hall 2001
*
*		www.peterjoel.com
*		info@peterjoel.co.uk
*
*****************************************************************/

mcClip.unknown1.addEventListener(MouseEvent.MOUSE_DOWN, myMouseDown);
mcClip.unknown1.addEventListener(MouseEvent.MOUSE_OVER, myMouseOver);
stage.addEventListener(MouseEvent.MOUSE_UP, myMouseUp);
var mousePos:Point = new Point();

// calculate all positions of MC before we start
```

```
// create an array of all MC positions along the tween
var L:int = mcClip.totalFrames;
var pos:Array = new Array(L+1);

for(var i:int=1; i<L+1; i++){
	mcClip.gotoAndStop(i);
	pos[i] = new Point(mcClip.unknown1.x + mcClip.unknown1.width/2,mcClip.unknown1.y +
mcClip.unknown1.height/2);
}
//mcClip.gotoAndStop(1);
mcClip.gotoAndStop(L);

//parameter to determine how far along the tween (in frames)
//the function will check to see if it is closer to the mouse
var speed:int = 20;

function myMouseDown(e:MouseEvent):void {
	stage.addEventListener(MouseEvent.MOUSE_MOVE, myMouseMove);
}

function myMouseOver(e:MouseEvent):void {
	mcClip.unknown1.buttonMode = true;
}

function myMouseMove(e:MouseEvent):void{
	setNewPos(e);
	setThumbPos(e);
	e.updateAfterEvent();
}

function myMouseUp(e:MouseEvent):void {
	stage.removeEventListener(MouseEvent.MOUSE_MOVE, myMouseMove);
}


// set position of MC to nearest to mouse
function setNewPos(e:MouseEvent){
	if(e.target != null){
		mousePos.x = stage.mouseX - 275;
		mousePos.y = stage.mouseY - 203;
		//mcClip.gotoAndStop(nearestFrame(mousePos)); //stop auto playback mode
		mcClip.gotoAndStop(nearestFrame(mousePos)); // continue auto playback mode
		//mcClip.gotoAndPlay(nearestFrame(mousePos));
	}
}


// calculate frame in the given range in which
// the MC is closest to the mouse
var currFrame:int;
var dir:int;
var lastFrame:int;
var frameCtr:int;
```

```
function nearestFrame(mousePos:Point):int {
	// get local references for extra speed
	currFrame = mcClip.currentFrame;
	//var pos = this.pos;
	dir = 0;
	// find out which direction along the curve takes the MC closer to the mouse position
	if(currFrame == L){
		dir = -1;
	}else if(currFrame == 1){
		dir = 1;
	}else if(dist(pos[currFrame+1], mousePos) < dist(pos[currFrame], mousePos)){
		dir = 1; // forwards
	}else if(dist(pos[currFrame-1], mousePos)< dist(pos[currFrame], mousePos)){
		dir = -1; // backwards
	}else{
		return currFrame; // stay in the same position
	}

	lastFrame = currFrame + speed*dir;
	if (lastFrame + dir > L)lastFrame = L;
	if (lastFrame + dir < 1)lastFrame = 1;
	for(frameCtr=currFrame; frameCtr!=lastFrame; frameCtr+=dir){
		if(dist(pos[frameCtr], mousePos) < dist(pos[frameCtr+dir], mousePos)){
			return frameCtr; // return minimum
		}
	}
	// didn't find a minimum, so return closest within the speed limit
	return frameCtr;
}

function dist(position:Point, mousePos:Point):Number{
	return Point.distance(position,mousePos);
}

mcScrubber.mcTarget = mcClip;

function setThumbPos(e:MouseEvent):void{
	mcScrubber.mcThumb.x = mcClip.currentFrame/L*mcScrubber.mcSlide.width;
}
```

File:	EW Demo2
Layer:	Fullscreen
Frame:	1

```
stage.displayState = StageDisplayState.FULL_SCREEN;
```

File:	EW Demo2
Layer:	PVR -> actions
Frame:	1

```
//AS3 MovieClip scrubber control by D Coady
```

```
var minX:Number = mcSlide.x;
var maxX:Number = mcSlide.x + mcSlide.width -mcThumb.width;
var percent:Number;

var mcTarget:MovieClip;  //the target movieclip to be controlled

mcThumb.buttonMode = true;
mcThumb.x = maxX;
mcThumb.addEventListener(MouseEvent.MOUSE_DOWN, myMouseDown);
stage.addEventListener(MouseEvent.MOUSE_UP, myMouseUp);

function myMouseDown(e:MouseEvent):void {
	this.addEventListener(MouseEvent.MOUSE_MOVE, myMouseMove);
}

function myMouseMove(e:MouseEvent):void {
	if (mouseX - mcThumb.width/2 < minX){    //Left lower scrub limit
		mcThumb.x = minX;
	}else if(mouseX - mcThumb.width/2 > maxX){ //Right maximum scrub limit
		mcThumb.x = maxX;
	}else{
		mcThumb.x = this.mouseX - mcThumb.width/2;
	}
	percent = mcThumb.x/(maxX-minX);
	this.mcTarget.gotoAndStop(Math.floor(percent*this.mcTarget.totalFrames)+1);
	e.updateAfterEvent();
}

function myMouseUp(e:MouseEvent):void {
	this.removeEventListener(MouseEvent.MOUSE_MOVE, myMouseMove);
}
```

File:	EW Demo2
Layer:	Main clip -> Checkbox
Frame:	400

```
var myTextFormat2:TextFormat = new TextFormat();
myTextFormat2.color = 0xFFFFFF;
myTextFormat2.size = 10;
checkbox.setStyle("textFormat", myTextFormat2);

checkbox.selected = true;
checkbox.addEventListener(MouseEvent.CLICK, changeEW2);

function changeEW2(e:Event):void{
	if (EW.visible == false){
		EW.visible = true;
	}
	else{
		EW.visible = false;
	}
```

```
}
```

File:	EW Demo2
Layer:	Main clip -> Play button
Frame:	1

```
playBut.addEventListener(MouseEvent.CLICK, playEvent);

function playEvent(e:Event):void{
        play();
}
```

List of symbols/abbreviations/acronyms/initialisms

BO	Bearing Only
DND	Department of National Defence
DRDC	Defence Research & Development Canada
EW	Electronic Warfare
FLA	Flash
NTDS	Naval Tactical Display Symbol
OMI	Operator Machine Interface
PVR	Personal Video Recorder
R&D	Research & Development
SME	Subject Matter Expert
SWF	Shockwave Flash
TDB	Track Data Block
TSA	Tactical Situation Area

This page intentionally left blank.

Distribution list

Document No.: DRDC Atlantic TM 2009-007

LIST PART 1: Internal Distribution by Centre

1 Francine Desharnais
1 Carrie McMullin
1 Mark Hazen
1 Don Coady
1 Bruce Chalmers
5 DRDC Atlantic Library

10 TOTAL LIST PART 1

LIST PART 2: External Distribution by DRDKIM

1 Library and Archives Canada, Attn: Military Archivist, Government Records Branch

1 Jeff Hardy
PMO HCM/FELEX
MGen George R. Pearkes Building
National defence Headquarters
Ottawa, ON K1A 0K2

1 DRDKIM

3 TOTAL LIST PART 2

13 TOTAL COPIES REQUIRED

This page intentionally left blank.

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

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 R&D Canada – Atlantic
9 Grove Street
P.O. Box 1012
Dartmouth, Nova Scotia B2Y 3Z7

2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)

UNCLASSIFIED

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.)

Exploring Operator Interface Design Concepts Using Adobe Flash: Design Concepts and Prototype Documentation

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

McMullin, C.; Coady, D.

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.)
May 2009	66	4

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 R&D Canada – Atlantic
9 Grove Street
P.O. Box 1012
Dartmouth, Nova Scotia B2Y 3Z7

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.)
11BS	

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 Atlantic TM 2009-007	

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 Shipboard Command and Control Group at DRDC Atlantic has previously investigated several decision support tools aimed at improving the contact recognition and identification processes performed by Naval Task Group operators. This document describes the results of using Adobe Flash CS3 Professional as an interface prototyping tool to examine more closely a subset of the proposed interface concepts.

Flash was used to develop two interface demonstrations which provided a tangible means for exploring and evaluating the 'look and feel' of interface design options. The "Track Data Block and Display Concepts" demonstration explored real-time design concepts such as a track data block in the form of a mouse hover activated 'details on demand' feature (tooltips), tailored right-click context menus, variable-hour track history (breadcrumbs) for contacts, interactive breadcrumb track information, and track future information (dead reckoning). The Bearing Only (BO) Overlay demonstration showed the usefulness of temporal review capabilities, especially when using an electronic warfare overlay highlighting scheme to assist in associating BO tracks with radar tracks.

The extensive graphic, animation, and interactive design capabilities provided by Flash made it an excellent rapid prototyping tool. Flash symbols, components, and ActionScript code further enhanced its flexibility and reusability as a design and development platform.

Le Groupe des systèmes de commandement et de contrôle embarqués de RDDC Atlantique a déjà examiné plusieurs outils de soutien à la décision afin d'améliorer les processus de reconnaissance et d'identification des contacts effectués par les opérateurs appartenant au Groupe opérationnel naval. Ce document présente les résultats de l'utilisation d'Adobe Flash CS3 Professional comme outil de prototypage d'interface afin d'examiner plus en détails un sous-ensemble des principes proposés pour les interfaces.

Flash a été utilisé afin d'élaborer deux démonstrations d'interface qui constituent des moyens tangibles pour explorer et évaluer la présentation et le comportement des options de conception d'interface. La démonstration « Principes de blocs et d'affichage de données de piste » a exploré des principes de conception en temps réel, comme le principe de bloc de données de piste » au moyen de « l'affichage de détails sur demande » activé par le passage de la souris (info-bulles), menus contextuels adaptés, historique de piste à variation horaire (pistes de navigation) pour les contacts, information sur les pistes par piste de navigation interactive et information sur le futur des pistes (valeur estimée). La démonstration Calque de relèvements seuls (BO) a montré l'utilité des capacités d'examen chronologique, en particulier lorsqu'on applique une technique de mise en surbrillance de calque de guerre électronique pour aider à effectuer l'association de pistes en relèvements avec des pistes radar.

Les capacités étendues de Flash dans les domaines du graphisme, de l'animation et de la conception interactive en fond un excellent outil pour effectuer du prototypage rapide. Les symboles et composants de Flash ainsi que son code ActionScript ajoutent à sa souplesse et à la réutilisabilité de ses produits comme plateforme de conception et de développement.

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.)

Flash; Naval Tactical Displays; Temporal Visualization; Interface Design; Interface Prototype

This page intentionally left blank.

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