Trident Development Framework

Tom MacAdam
Jim Covill
Kathleen Svendsen
Martec Limited

Prepared By:
Martec Limited
1800 Brunswick Street, Suite 400
Halifax, Nova Scotia B3J 3J8 Canada

Contractor's Document Number: TR-14-85 (Control Number: 14.28008.1110)
Contract Project Manager: David Whitehouse, 902-425-5101
PWGSC Contract Number: W7707-145679/001/HAL
CSA: Malcolm Smith, Warship Performance, 902-426-3100 x383

Contract Report
DRDC-RDDC-2014-C328
December 2014

Lloyd's Register Marine

Working together for a safer world

Trident Development Framework

Martec Technical Report # TR-14-85
Control Number: 14.28008.1110

December 2014

Prepared for:

DRDC Atlantic
9 Grove Street
Dartmouth, Nova Scotia
B2Y 3Z7

Martec Limited
1888 Brunswick Street, Suite 400
Halifax, Nova Scotia B3J 3J8 Canada

tel. 902.425.5101
fax. 902.421.1923
email. martec-info@lr.org
www.martec.com

REVISION CONTROL

REVISION	REVISION DATE
Draft Release 0.1	**10 Nov 2014**
Draft Release 0.2	**2 Dec 2014**
Final Release	**10 Dec 2014**

PROPRIETARY NOTICE

This report was prepared under Contract **W7707-145679/001/HAL, Defence R&D Canada (DRDC) Atlantic** and contains information proprietary to Martec Limited.

The information contained herein may be used and/or further developed by **DRDC Atlantic** for their purposes only.

Complete use and disclosure limitations are contained in Contract **W7707-145679/001/HAL, DRDC Atlantic**.

SIGNATURE PAGE

TRIDENT DEVELOPMENT FRAMEWORK

Technical Report # TR-14-85 Rev 00
04 December 2014

Prepared by: _____ **Date:** 4 Dec 2014
Tom MacAdam
Team Leader, Software Development

Prepared by: _____ **Date:** Dec 9/14
Jim Covill
Team Leader, Field Services and Trident

Prepared by: _____ **Date:** 8 Dec 2014
Kathleen Svendsen
Research Engineer, Field Services and Trident

Reviewed by: _____ **Date:** 8 Dec 2014
David Whitehouse
Technical Manager

Approved by: _____ **Date:** 8 Dec, 2014
Claude DesRochers
Senior Project Manager

EXECUTIVE SUMMARY

This report details work undertaken to develop the Next Generation Trident (NGT) skeleton application. The work included an evaluation of alternatives for the graphics and user interface components of the application, from which the Visualization Toolkit (VTK) and Microsoft's PRISM were chosen, respectively. The technologies were then integrated to form the skeleton NGT application, which demonstrated the layout of the application as well as its performance in loading and visualizing an AVAST model, as well as other stress-test models. Finally, the complete list of features for the NGT was determined and presented as a series of related feature sets which can be used to carry out the remaining phases of the NGT implementation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

Previous work, undertaken under Task 5 – "Next Generation Trident Roadmap" of contract W7707-145679/001/HAL, documented the need to improve user interaction and/or usability shortcomings with the Trident software suite of applications [1]. The work identified, through user surveys, two main areas of focus – Trident's graphical rendering capability, and its user interface (UI) framework. The Trident derived code PVAST, contained within the main Trident application, exhibits the same issues and is addressed as part of the Next Generation Trident (NGT) upgrade. In addition, the AVAST UI/front end application, currently independent of Trident, can also significantly benefit via incorporation of the new Trident and PVAST architecture, should it undergo future enhancements to address usability concerns.

This task details the design and development of the skeleton framework for the NGT V1.0 application based on informed technology choices from a thorough evaluation of alternatives. This skeleton framework serves as the basis for all subsequent development tasks that will constitute the final NGT V1.0 application. This report also details the design considerations and decisions that underpinned the development of this framework as well as gives an overview of the framework deliverable.

Finally, this task defined the scope and extent of the feature set that will be carried from the Current Generation Trident (CGT) to the NGT. These features were chosen based on priority towards yielding a version 1.0 release that provides the most functionality to an end user within a reasonable development timeframe.

2.0 THE NGT GRAPHICS ENGINE

The *graphics engine* of the NGT application refers to the subsystem responsible for rendering the finite element model graphically on the screen and allowing the user to interact with it for purposes of pre-processing and post-processing their analysis. The graphics engine chosen for the NGT application is the Visualization Toolkit (VTK, see [2]), an open source, freely available library developed and commercially supported by Kitware Inc. [3]. The various design considerations leading to this choice over the alternatives are given in the following sections.

2.1 DESIGN CONSIDERATIONS

Results of the Trident user survey undertaken in Task 5 of this contract indicated that model rendering and interactivity were the leading concerns for users of the current-generation application. Specifically, the graphics subsystem was slow to render large models, and the lack of dynamic interaction with the model viewport made it more difficult to establish context when navigating the model and isolating features of interest.

From these concerns, the leading design considerations for the graphics subsystem of the NGT were identified as:

1. True dynamic 3D interaction – a graphics viewport that supports the basic 3D manipulations (rotate, pan, zoom) in a dynamic, real-time manner driven by input from a mouse or similar tracking device.
2. Scalability – the graphics system must accommodate models at least one order of magnitude larger than the current norm (i.e. support models containing multiple millions of elements vs. hundreds of thousands) while maintaining acceptable performance.
3. Support for modern graphics features – the graphics system must support features such as lighting and transparency, which improve model fidelity and provide users with more visual cues to help interpret characteristics of their model such as curvature and context.

The VTK library meets all of these needs. Regarding true dynamic interaction, the default model renderer already provides the capability out of the box, and various pluggable algorithms exist for advanced interaction such as entity picking. Regarding advanced features, VTK provides access to lighting and material models to support feature shading and surface transparency, as well as algorithms to visualize vector, volumetric and flow fields.

To assess scalability, tests were done to demonstrate performance with increasingly large models. The tests were run on two different computers:

- HP EliteBook 8570w, featuring:
    - Intel Core i7-3720QM (2.6 GHz)
    - 16 GB RAM

  - NVIDIA Quadro K1000M graphics card (2048 MB)
  - Windows 7, 64 bit

- Dell XPS 8500, featuring:
  - Intel Core i7-3770 (3.4 GHz)
  - 16 GB RAM
  - AMD Radeon HD 7870 (2048 MB)
  - Windows 7, 64 bit

The testing application was implemented in Windows Forms using the ActiViz .NET wrapper for VTK 5.8 [4]. Frame rates were measured using FRAPS by Beepa Pty. Ltd. [5]. To determine the frame rate measurement, the model was manually rotated with the mouse until the frame rate reported by FRAPS reached a peak steady state. The application displayed a quad mesh with the elements filled using a solid color and their edges drawn using OpenGL's polygon offset feature. The results are shown in Figure 2-1. The "professional" NVIDIA Quadro card performed better than the "consumer" AMD Radeon card, which was likely due to the mesh model containing many wireframe edges that professional cards are better optimized to handle. In both cases, however, interactive performance was still reasonable and could be further improved using other capabilities offered by VTK (for instance auto-clipping of elements outside the view area or inserting a polygon decimation filter).

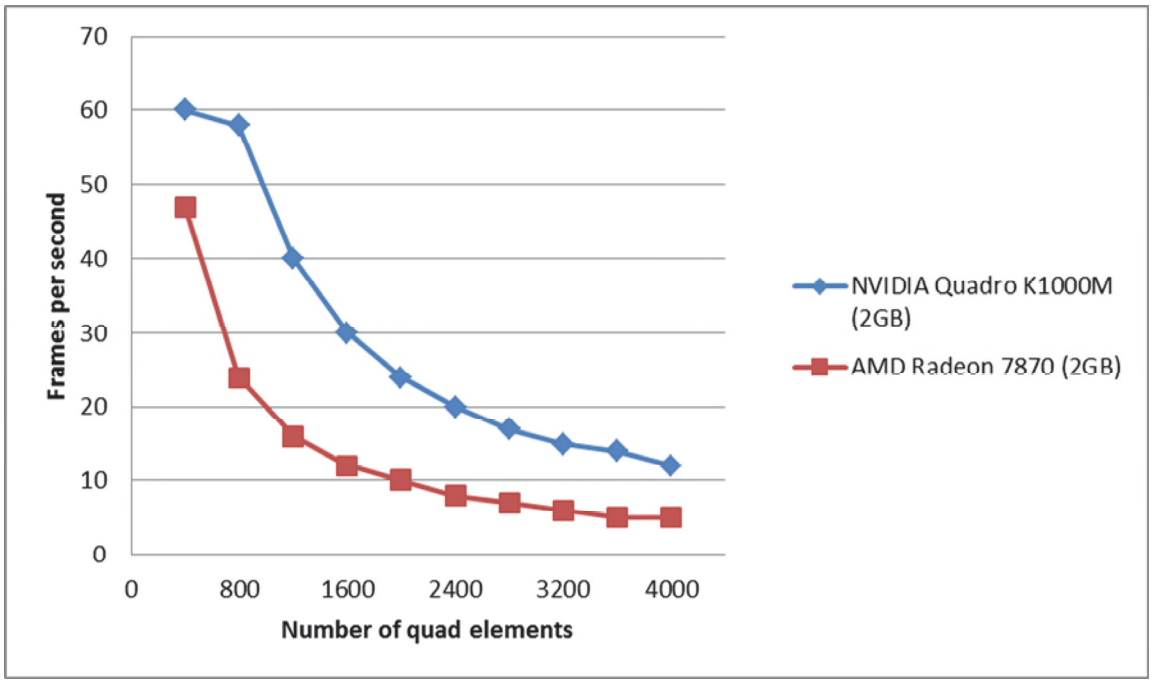


Figure 2-1: Results of VTK performance tests.

In addition, beyond just meeting these requirements for the NGT, VTK offered various additional benefits over other alternatives. VTK is compared to other alternatives below.

2.2 VTK OVER OPENGL/DIRECT3D

While OpenGL [6] or Direct3D [7] can be used from the ground up to achieve all of the design considerations listed above, the VTK system was chosen in large part because it manipulates graphics at a higher level than either of these libraries. The OpenGL/Direct3D libraries provide routines for loading data into graphics memory and then writing custom code to inject functionality into the graphics pipeline for customizing how the data is displayed (i.e. *shader* codes). The user must configure their graphics context, write and load their shaders, write the code to introduce dynamic interaction with the render window, etc. VTK, on the other hand, abstracts much of the direct interaction with graphics hardware by introducing its own higher-level notion of the graphics pipeline. In VTK, the pipeline consists of data sources, mappers, filters and algorithms that wire together and ultimately funnel into a renderer for display. The renderer already covers configuring the graphics context as well as basic 3D manipulations (rotate, pan, zoom), thereby saving significant effort over straight OpenGL/Direct3D.

While direct interaction with the graphics hardware through OpenGL/Direct3D can yield extremely performant solutions, it requires much more work to implement a complete solution. VTK's simplified pipeline architecture allows a solution to be created by simply snapping together a series of logical building blocks. This comes at the cost of losing some fine-grained control over the low-level rendering calls, and that introduces the potential for some performance loss. But these concerns can be mitigated by the fact that VTK is itself built on OpenGL, and provides access to OpenGL at various points (e.g. the ability to build and hook custom shaders). Furthermore, since VTK is open-source, its implementation can be studied and tweaked if particular performance bottlenecks are identified.

2.3 VTK OVER TRIDENT VISUALIZER

Martec developed the Trident Visualizer application starting in 2006 specifically for providing interactive post-processing of VAST analyses for the SubSAS software. The Visualizer was implemented for the Microsoft .NET platform using the Windows Forms UI framework with an embedded OpenGL graphics window. To access OpenGL from the managed (C#) code, the open source wrapper, OpenTK [8] was used. In the end, Trident Visualizer was able to load a Trident model stored in the Trident Database format, visualize the model, perform basic element quality checks and visualize stress, mode shape and displacement results.

It was considered to use the graphics capability built into the Visualizer as a basis for the NGT. This approach was ultimately decided against, however, due to the effort that would be required to meet design considerations 2 and 3 above. Most notably, the Visualizer was based on the deprecated fixed-function OpenGL pipeline, meaning scalability would suffer from not being able to leverage the raw power of modern programmable pipeline based hardware. While some research and prototyping work had been done to explore refactoring to the programmable pipeline [9], significant effort would be necessary to implement the recommendations. Furthermore, though the Visualizer supported simple lighting, it had no support for various other advanced features (e.g. transparency). To add these features would, again, incur upfront work that would not be needed with VTK.

2.4 VTK VS. COMMERCIAL GRAPHICS ENGINES

Various commercial graphics engines were known to the team designing the NGT. These tools required licensing fees to use and in some cases royalty payments to embed into commercial software. While a few compelling alternatives were identified, including Hoops3D (Tech Soft 3D) and Open Inventor (Visualization Sciences Group), they were set aside outright, owing to commercial concerns rather than any sort of technical shortcomings.

2.5 LIMITATIONS

While VTK was chosen over the alternatives for the strong benefits mentioned above, there are a small number of drawbacks to consider. Firstly, though the source code is available, it is a large and somewhat complicated library to build. The CMake tool must first be used to configure files for building with Visual Studio, and success of this step depends on proper support for the Visual Studio version being included in the CMake files. The CMake files are not always updated promptly with the release of new versions of Visual Studio.

VTK is natively C++ and in order to use VTK from .NET managed languages such as C#, one must use a managed wrapper. Kitware provides the ActiViz .NET wrapper for this purpose. While the source code is available, known as ActiViz .NET OpenSource Edition, it is very difficult to build successfully. The company supplies pre-built binaries for older versions of the library, but to get binaries for the latest versions, they must be purchased through a commercial support contract with the company. For the skeleton NGT application, ActiViz .NET 5.8 was used, which was a couple versions behind the latest (6.1). This was not observed to have a significant negative impact, though some of the latest features of VTK were not available. It is the recommendation of the authors to purchase the ActiViz commercial support to obtain the latest ActiViz build for integrating into the NGT.

3.0 THE NGT USER INTERFACE FRAMEWORK

The UI framework of the NGT application refers to the subsystem responsible for displaying interactive interfaces to the user for purposes of presenting and/or inputting data. The UI framework includes the windows, dialogs and controls that the user interacts with, but also the infrastructure that manages them and the messages they send to each other. The UI framework chosen for the NGT application is Microsoft's Windows Presentation Foundation (WPF) coupled with their PRISM [10] framework for composite applications. The design considerations leading to this choice are detailed below.

3.1 DESIGN CONSIDERATIONS

The Trident user survey indicated that the non-standard nature of the Trident UI detracted from the user experience interacting with the program. The UI controls not only looked dissimilar to common Windows controls, they behaved differently and in some cases did not support standard usability patterns such as cut/copy/paste or text box undo/redo. Furthermore, the dialogs were mostly fundamentally modal by nature, meaning the user could only interact with one dialog at a time. This had the effect of limiting the user to singular workflows, i.e. they could not perform multiple operations at once.

Beyond the UI's effect on end user experience, however, there were other needs identified for the NGT that were influenced by its UI. Firstly, it was desired to have access to Trident functionality from other applications. While this had been previously partially possible using the Trident Database API (used, for instance, by SubSAS and STRUC), access to the full breadth of Trident features was not possible because many were fundamentally tied to its own UI. Secondly, it was desired to have an easier means to treat Trident as a platform into which additional functionality could be plugged. Finally, in order to support increased test coverage, it was useful to have a means to test Trident's logic without having to automate UI interaction.

From these points, the leading design considerations for the UI subsystem of the NGT were identified as:

1. Controls that match those in peer applications and support the standard UI interactions such as keyboard navigation, cut/copy/paste, local undo/redo, drag and drop, etc. Achieve this through a framework providing native controls that are common amongst peer applications on the given operating system.
2. Overcome the limitation that only one dialog can be shown at once, thereby limiting the user to performing only one task at a time. Achieve this through use of modeless dialogs or other modeless dockable property editors.
3. Improve the utility of Trident by enabling re-use of program logic, even by other applications. Enable this through rigorous separation of logic from presentation (UI) using a modern, highly data-bound, UI framework.
4. Future-proof the program by making aspects of the UI more easily replaceable. Achieve this through rigorous separation of individual aspects of the presentation layer from each other as well as the logic layer.

5. Improve quality by improving testability of the program. Achieve this through rigorous separation of the presentation and logic layers, supporting the ability to directly test the logic layer without complicated UI automation.

One important thing to note was that cross-platform support for NGT was not deemed a high priority. While it was recognized that it might be desirable to run Trident's standalone VAST solver on, for example, Linux clusters (to leverage concurrency or larger memory pools), this was not stressed for the Trident pre and post-processor. As such, it was concluded that NGT would remain fundamentally within the Microsoft Windows ecosystem.

3.2 WPF OVER WINDOWS FORMS OR MFC

Microsoft's current recommended framework for desktop application development is WPF. WPF supersedes the older Microsoft Foundation Classes (MFC) and Windows Forms (WinForms) UI frameworks by providing a rich toolkit based fundamentally on modern UI development patterns. Its advantages over the older technologies are many and include:

- **Rich composability** – all UI widgets and controls in the WPF framework can be composed freely. Whereas to customize existing controls in MFC and WinForms was quite difficult, in WPF it is often a matter of simply composing new controls out of existing ones. For example, if one wishes to add images and buttons to entries in a list box, instead of being required to override low-level routines as in MFC/WinForms, one simply defines the list box entries to consist of images and buttons along with any text that would normally appear. WPF handles sizing the control, propagating events, drawing, etc. automatically.
- **Data binding** – data binding is the mechanism that automatically ties data in a UI control to data in an object used by the logic layer of an application. While MFC and WinForms had limited data binding support, WPF makes data binding a fundamental mechanism for synchronizing between the UI and logic layers of the application. Data binding is much easier to set up and supports a great deal more flexibility (for instance one way as well as bidirectional).
- **Fully stylable** – all WPF UI controls can have their look entirely customized through the definition of custom styles. Styles, much like the Cascading Style Sheets (CSS) ubiquitous in web development, can apply at various levels of an application, from the top (i.e. applying to a whole application), to a specific dialog, to a specific control.

Perhaps most importantly, the features of WPF allow developing applications according to an architectural pattern derived from Model-View-Controller (MVC) known as Model-View-ViewModel (MVVM). This pattern enforces strict separation between the UI layer (View) and the storage/logic layers (Model and ViewModel). Used in the development of the NGT, this will force clean separation between the Trident logic and its UI, thereby meeting the goals of making the logic more broadly accessible and directly testable.

Though there are multiple third-party frameworks to assist in developing WPF applications using the MVVM pattern, Microsoft provides its own through the PRISM composite application guidance from its Patterns and Practices team. Now in its fifth major version,

PRISM provides not only MVVM infrastructure classes, but also a means to develop applications comprised of loosely-coupled modules. This has the added benefit that the modules can be reused or replaced more readily without affecting other modules, and the modules can be dynamically discovered at run time to allow, for instance, a plug-in loading mechanism for the application. WinForms had a similar, though not identical, guidance termed the Smart Client Software Factory (SCSF) as recently as 2010, but this has now been retired by Microsoft in favor of PRISM.

In addition to these benefits, since WPF is also Microsoft's current recommended technology for Windows desktop development, it is expected to see continued support and compatibility through future versions of Windows. Microsoft has traditionally worked hard to maintain compatibility for its development technologies through updates to the operating system, and it is not expected this will change. Furthermore, with the NGT application being fundamentally 64 bit from the outset, it does not require emulation on modern PCs, which typically run 64 bit natively.

3.3 WPF OVER QT

The QT framework by Digia [11] is an extensive framework for developing cross-platform applications. While it is mature (currently in version 5.3) and well regarded, its UI capabilities are in transition. While its older UI approach, QWidget-derived controls, was similar to WinForms, its newer UI technology, QT Meta Language (QML), is in many ways similar to WPF. Like WPF's Extensible Markup Language (XAML), it provides a declarative language for defining rich, stylable controls. QML also supports data binding controls to their logic classes, which is referred to as *property binding*.

Despite these options, however, WPF was still the preferred choice. Using the QWidget-derived controls would forgo the benefits of the newer UI frameworks, namely improved composability and better separation of concerns. While QML could have been a viable option, it was not as mature as the rest of QT, and its control suite, QT Quick Controls, contained only simple controls and lacked more complicated controls like a tree view, grid or list box. As such, it was behind WPF's capabilities for desktop application development. Finally, since NGT was aimed at the Windows platform, any cross-platform features offered by QT were viewed as low-priority benefits.

3.4 LIMITATIONS

Even though WPF was recognized as most suitable for the NGT platform, there were still some limitations identified. Firstly, since the VTK graphics engine is wrapped in a WinForms-based wrapper (ActiViz .NET), this has to be hosted in a special interop host control in WPF. While the WindowsFormsHost control was provided by Microsoft for this purpose, it suffers from the inability to render WPF content that is drawn on top of its region (this is commonly referred to as "airspace" issues). While this is indeed an issue, it is not expected to impact on NGT since no WPF content will be required to extend within the VTK viewport region of the screen.

Another limitation of using WPF and PRISM to base the NGT architecture on is that it makes various modules of the NGT reliant on Microsoft's .NET Framework. While this is somewhat of a risk in that the .NET Framework may be phased out by its owner (Microsoft), it is suggested that any commercial third-party toolkit/framework chosen suffers the same category of risk. It is mitigated as much as possible by the fact that WPF is presently the technology recommended by Microsoft for desktop application development and there are no suggestions that it is going to be phased out any time soon. As such, it is expected to be supported well into the future. Furthermore, architecting the NGT to adhere to modularity and strict separation of concerns will mean the majority of the logic code will be self-contained in native libraries. These should be maximally portable well into the future, even if the entire presentation layer has to be replaced.

4.0 THE NGT 1.0 SKELETON APPLICATION

4.1 CHARACTERISTICS

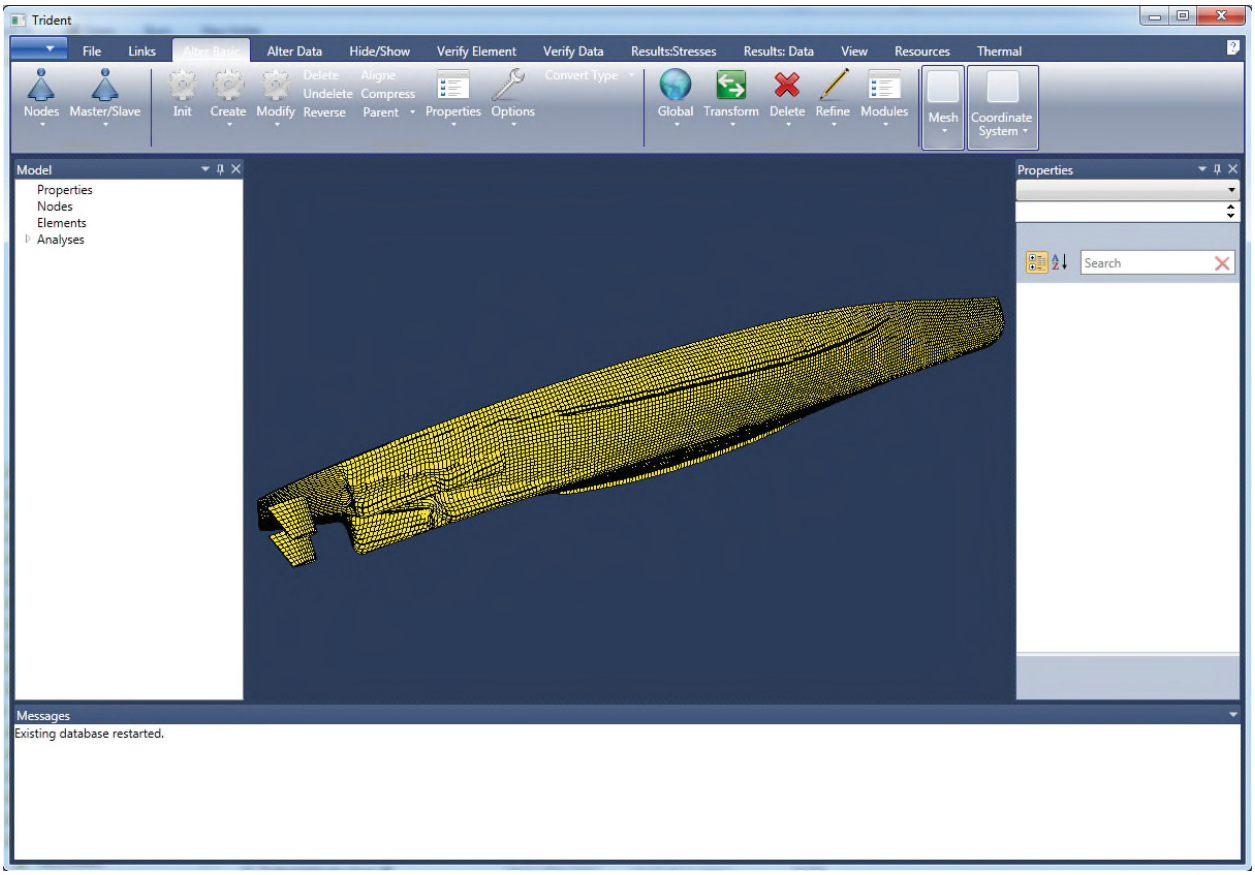
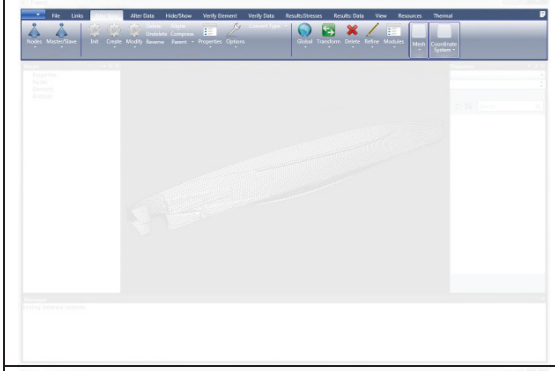
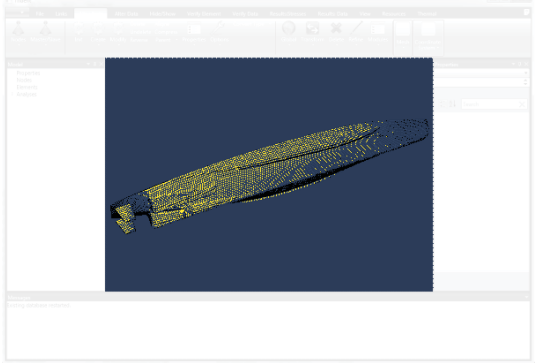
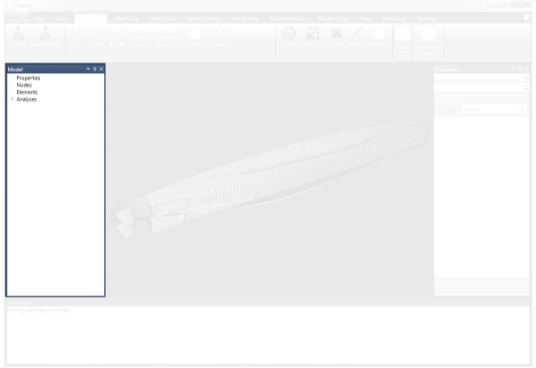
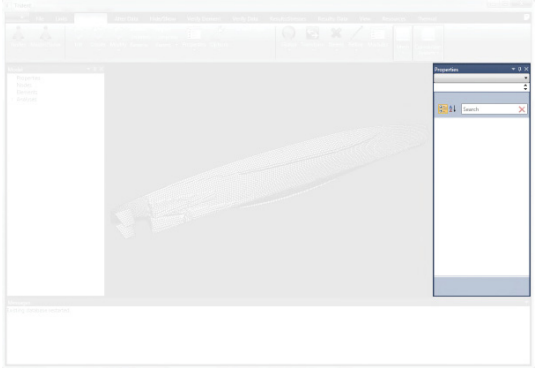


Figure 4-1: The NGT V1.0 Skeleton Application.

Building upon the design decisions described in the previous Sections, the skeleton NGT V1.0 application was constructed using WPF, PRISM and VTK (Figure 4-1). The application consisted of five loosely-coupled PRISM modules, as described below in Table 4-1. The AvalonDock component of the open source Extended WPF Toolkit Community Edition [12] was used to arrange the module regions within the PRISM shell. Microsoft's own WPF ribbon control (.NET Framework 4.5) was used for the menu. The "VS2010" theme (dark blue) was chosen for the AvalonDock controls and the default theme of the ribbon control was manually adjusted to bring it in line with the rest of the controls.

Table 4-1: The five modules of the NGT V1.0 skeleton application.

	**Main Menu Module** The main menu closely reflected the menu structure of the current generation Trident application, but was implemented in the form of a "ribbon" interface. It was felt that the ribbon interface offered improved usability and more closely reflected the modern UI style of mainstream application such as Microsoft Office or AutoCAD.
	**Viewport Module** The viewport was the module responsible for rendering the model using the VTK graphics engine. The default VTK interactor was used to allow the model to be rotated, zoomed and panned using the mouse. The viewport background was adjusted to bring it in line with the dark blue theme used by the other controls, and though lighting was enabled, the material reflection model was set to flat to diminish shading; this can be adjusted to preference in subsequent tasks.
	**Model Browser Module** The model browser was put in place to eventually provide a location where the model objects could be browsed in a hierarchical/tree view. An example arrangement of top-level entity collections was implemented, though it was expected this would ultimately change. The model is expected to be synchronized with the viewport and property browser to show the location and details of selected objects(s).
	**Property Browser Module** The property browser was created to allow properties of selected objects (for example nodes, elements, etc.) to be shown in a concise, tabular form. The property browser was designed to be used to quickly interrogate and/or modify properties of individual or groups of objects.

The Logger Module

The logger window provided a location where messages of various types could be shown to the user. The messages could include information, status updates, cues to indicate a process was underway and error messages. Beyond NGT 1.0, the logger window can be extended to also serve as an interactive command window to accept user-input commands to drive the program.

The current phase of work called for the NGT 1.0 skeleton application to be able to load and visualize a representative PVAST or AVAST geometric model. To achieve this, the current generation Trident core functionality was encapsulated into a module called TridentCore and integrated into the PRISM application. Modifications were made to the TridentCore code to disable some of the legacy UI and graphics calls, leaving a set of routines capable of processing model load operations as simply an update to the Trident database. The skeleton application was then free to implement its own means to reflect the change to the Trident database in its own UI, which for this phase of the work was to render the database in the VTK viewport.

The other modules communicated with the TridentCore through an interface layer referred to as the *core surface*. The core surface was responsible for marshalling calls from the .NET managed runtime to the native TridentCore library. Message calls arrived at the core surface from the remaining modules via PRISM's *event aggregator*, to which the core surface *subscribed* to message types *published* from the remaining modules.

Through the implementation of the skeleton NGT V1.0 application in the manner described above, the design choices were validated and found to perform very well. The VTK-based viewport rendered the test models, as well as stress test models consisting of 1.2 million elements, with very good interactive performance. Application startup time was relatively slow in comparison to the current generation Trident due to the requirement to load the .NET framework as well as the PRISM, VTK and other assemblies, but was felt to be in line with other large applications.

4.2 LIMITATIONS

As per the requirements of this phase of work, the NGT V1.0 skeleton application was only implemented to the point of being able to open a model and display the model in the VTK viewport. The remainder of the functions shown in the menu were not enabled. The logger window was hooked up to display standard information messages coming from the Trident core module, but the model browser and properties browsers were not enabled. Though the overall color theme of the main menu was adjusted to match the AvalonDock theme, the adjustments made to the text color rendered it invisible in some contexts. As such more work customizing the ribbon theme is recommended to fix this.

5.0 PRIORITIZED IMPLEMENTATION ORDER FOR NGT V1.0

This section documents the feature set considered in scope for the NGT version 1.0 application. These features were identified as critical to achieving an application that provides a useful, operational product for the 1.0 release.

5.1 FEATURE SET 1 - 'FILES' AND 'LINKS' MODULES

Combine the existing and proven Trident database functionality (specifically the native VAST database functionality, and a) FEMAP, b) NASTRAN, c) ANSYS and d) MAESTRO) to 'import' and 'export' into/from Trident into NGT 1.0. The bulk of this task will comprise of the porting/repackaging of the current 'Files' and 'Links' dropdown.

Functionality to include all import and export functions for

- Trident and VAST
- FEMAP
- NASTRAN
- ANSYS
- MAESTRO

The importing and display of the largest available VAST models will be used to demonstrate the scalability of this module with respect to time to load and render the FE model elements. Correct visualization of the model using the supported simple render modes (i.e. solid, wireframe, hidden line, with and without lighting enabled), will constitute completion of this capability.

Fundamental changes to the makeup of the Trident database will not be made as part of this work so that this work can be isolated mainly to the presentation layer. Work to modify the database storage will be undertaken in a later phase.

5.2 FEATURE SET 2 – GENERATE/MODIFY MODULE

Port/repackage the underlying Generate/Modify dropdown menu capability to include:

- All Property Data functionality
- All Node functionality (except for the Options functions, where only the Scale, Translate, Align, and Fluid to Structure functions will be included)
- All Elements functionality for Bar, Triangular Membrane, Quadrilateral Membrane, Beam, Triangular Plate, Quadrilateral Shell, Stiffened Shell, 8 Node Shell, 4 Node Fracture, Solid, Fluid, Spring, Gap and Rigid Link elements

- All Boundary Conditions functionality
- All Lumped Masses functionality
- All Load functionality (except for the Options functions, where only Translate and Static Balance will be included)
- All existing Fluid-Structure functionality
- All Crack Lines and Tips functionality (except for Crack Mesher)
- All Master and Slave Nodes functionality
- All Formulation functionality
- All Global Model functionality
- All Functions and Lists functionality (except Harmonic List)
- Mesh functions Nodes, Bar and Beam, 2D surface (Corner Points) and 2D surface (Boundary Points)
- All Refine functionality (except Corrugated Panel)
- All Modules functionality
- All Transform Model functionality
- All Delete functionality

5.3 FEATURE SET 3 – BASIC ANALYSIS GENERATION AND VERIFICATION

The ability to create and execute a basic VAST analysis will be added to the NGT framework. Initial analysis conditions such as loads, boundary conditions, material properties will be generated/obtained from the Trident 2014 system via the database. The analysis verification process will include integration of Trident analysis verification modules, which allow viewing of analysis parameters and plotting of initial conditions.

All pertinent VAST input files (USE, GOM, LOD, SMD) will be created at this point. NGT will be capable of performing basic FE analyses where a multistep restart procedure is not required. However, the restarts in the existing Trident will be retained in the following cases: restarts allowing additional eigenvalues/eigenvectors; restarts using previously determined eigenvalues/eigenvectors for modal based analyses (modal superposition, response spectrum and frequency response); and stress/strain recovery using previously determined linear displacements.

Analysis Types to be supported include:

- Static
- Eigenvalue-Natural Frequency
- Eigenvalue-Buckling Modes
- Dynamic-Modal Superposition
- Dynamic Direct Integration
- Dynamic Response Spectrum

- Dynamic-Modal Frequency Response
- Dynamic-Direct Frequency Response

5.4 FEATURE SET 4 – RESULTS MODULE

The VAST solver will be executable within the NGT environment as a result of Feature Set 2 (above). This Feature Set addresses the post processing and display of the VAST result files.

Display capabilities to include:

- Deformations (vibration, buckling, modal, velocities, displacements, acceleration),
- Stresses/Strains (including all components for all load cases),
- Support reactions,
- Summary of Results

5.5 FEATURE SET 5 – VERIFY MODULE

The verify module is used to assist the user to confirm the 'correctness' and/or integrity of the FEA model prior to VAST execution.

New GUI and 3D Graphics functionality to include:

- Property Data
- Elements Properties (Angles, Distances)
- Normals, Surfaces, Edges
- Integrity Test (including Angles, Distances)
- Boundary Conditions
- Mass
- Loads
- Crack Tips
- Functions and Lists
- Modules
- Summary

5.6 FEATURE SET 6 – OPTIONS MODULE

The options module permits the user to change such things as the measurement system, display modes and other features of Trident.

New GUI and 3D Graphics functionality to include:

- Initialize Functions
- Settings
- Animation
- Other Formats
- Unit Conversion
- Volumes and Areas
- Cross Section
- Centre of Gravity
- Prescribed Displacements
- X-Y Plots (using existing XY plotting program; future work will then seek to integrate the plotting capability natively into the application using Microsoft .NET plotting libraries, but this work will be in a subsequent phase)
- Contour

5.7 FEATURE SET 7 – INQUIRE MODULE

The inquire module permits tabulated summaries of various model entities.

New GUI and 3D Graphics functionality to include:

- Nodes
- Master Nodes
- Slave Nodes
- Elements
- Properties
- Elements Loads
- Concentrated Loads
- Deformations
- Stresses

5.8 FEATURE SET 8 – ERASE/RESTORE MODULE

This module permits the rapid masking of the model to assist in manipulating and viewing of models.

New GUI and 3D Graphics functionality to include:

- Restore All
- Erase All
- Erase by volume
- Erase by interactive picking
- Module Number
- Property Number
- Element Type
- Global
- Local
- External
- Internal
- Structure
- Fluid

5.9 FEATURE SET 9– DISPLAY MODULE

This module permits the enhanced viewing display options of FEA model entities (for example, element colours, node glyph visibility on/off).

New GUI and 3D Graphics functionality to include:

- Nodes
- Elements
- Erased Model Only
- Deleted Model Only
- Stiffened Panel Stiffeners
- Crack Lines and Tips
- Beam Cross Section Plotting

6.0 CONCLUSIONS AND RECOMMENDATIONS

The work undertaken for this task has decided on the key technologies that will form the Next Generation Trident application. The technologies were each researched and evaluated against alternatives and chosen based on their strong benefits. The technologies were then integrated to form the skeleton NGT application, which demonstrated the layout of the application as well as its performance loading and visualizing an AVAST model as well as other stress-test models. Finally, the complete list of features for the NGT was determined and presented as a series of related feature sets which can be used to carry out the remaining phases of the NGT implementation.

7.0 REFERENCES

[1] Svendsen, K., Covill, J. 2014. *Trident Development Roadmap*. TR-14-59. Martec Limited. Halifax, N.S.

[2] Online: www.vtk.org

[3] Online: www.kitware.com

[4] Online: www.kitware.com/opensource/avdownload.php

[5] Online: www.fraps.com

[6] Online: www.khronos.org/opengl/

[7] Online: msdn.microsoft.com/en-us/library/windows/desktop/hh309466(v=vs.85).aspx

[8] Online: www.opentk.com/

[9] MacCormick, D. 2013. *Research into Modern Graphics Technologies*. TR-13-16. Martec Limited. Halifax, N.S.

[10] Wastell, B., et. al. 2014. Developer's Guide to Microsoft Prism Library 5.0 for WPF. Patterns and Practices, Microsoft Corporation, Redmond, WA.

[11] Online: qt-project.org

[12] Online: wpftoolkit.codeplex.com/