TP 13790E

ICE NAVIGATION SIMULATION – PHASE III

Prepared for
Transportation Development Centre
Transport Canada

by
PhiloSoft Inc.
3608 Boulevard St. Charles, Suite #27c
Kirkland, Quebec
H9H 3C3

May 2001

TP 13790E

ICE NAVIGATION SIMULATION – PHASE III

by
Charles G. Marton
PhiloSoft Inc.

May 2001

This report reflects the views of PhiloSoft Inc. and not necessarily those of the Transportation Development Centre of Transport Canada or the sponsoring organizations.

The Transportation Development Centre does not endorse products or manufacturers. Trade or manufacturers' names appear in this report only because they are essential to its objectives.

Un sommaire français se trouve avant la table des matières.

Transport Canada / Transports Canada

PUBLICATION DATA FORM

1. Transport Canada Publication No.	2. Project No.	3. Recipient's Catalogue No.
TP 13790E	5036	

4. Title and Subtitle	5. Publication Date
Ice Navigation Simulation – Phase III	May 2001
	6. Performing Organization Document No.

7. Author(s)	8. Transport Canada File No.
Charles G. Marton	ZCD2450-C-368

9. Performing Organization Name and Address	10. PWGSC File No.
Philosoft Inc. 3608 St. Charles Blvd., Suite 27C Kirkland, Quebec Canada H9H 3C3	MTB-0-02210
	11. PWGSC or Transport Canada Contract No. T8200-0-0558/001/MTB

12. Sponsoring Agency Name and Address	13. Type of Publication and Period Covered
Transportation Development Centre (TDC) 800 René Lévesque Blvd. West Suite 600 Montreal, Quebec H3B 1X9	Final
	14. Project Officer C. Gautier

15. Supplementary Notes (Funding programs, titles of related publications, etc.)

Co-sponsored by the Program of Energy Research and Development (PERD)

16. Abstract

Navigating ships in ice-infested waters is a risky and dangerous task. The requirement to develop an ice navigation simulator stems from a study conducted for the Transportation Development Centre in which the requirements for a training course were identified.

In Phase III the feasibility of enhancing system performance by dividing the simulation world into smaller, more manageable ones was demonstrated. This was done by restructuring and modifying the simulator's software and databases.

In addition, the out-of-window bridge view simulation was reworked to support a more realistic visual representation of various ice types on the horizon. This activity consisted primarily of applying different texture-mapping algorithms to texture maps of varying size, scale and content.

The main simulator platform was developed in conjunction with other modules such as the training aids module and the shipboard radar simulation module.

The main simulation software provides several key elements:

- Synthetic Aperture Radar (SAR) data conversion tools.
- SAR data analysis tools.
- Virtual out-of-window bridge view.
- Database management system.
- GPS and GYRO simulation.
- Instructor and student functionality.
- Bulletin Board Service data downlink.
- Scaleable network operation.

To date all of the changes in the modules that make up the simulator have been successfully integrated.

17. Key Words	18. Distribution Statement
Marine, radar, simulation, ice navigation	Limited number of copies available from the Transportation Development Centre

19. Security Classification (of this publication)	20. Security Classification (of this page)	21. Declassification (date)	22. No. of Pages	23. Price
Unclassified	Unclassified	—	xii, 15	Shipping/ Handling

Transports Canada Transport Canada

FORMULE DE DONNÉES POUR PUBLICATION

1. N° de la publication de Transports Canada	2. N° de l'étude	3. N° de catalogue du destinataire
TP 13790E	5036	

4. Titre et sous-titre	5. Date de la publication
Ice Navigation Simulation – Phase III	Mai 2001
	6. N° de document de l'organisme exécutant

7. Auteur(s)	8. N° de dossier - Transports Canada
Charles G. Marton	ZCD2450-C-368

9. Nom et adresse de l'organisme exécutant	10. N° de dossier - TPSGC
Philosoft Inc. 3608, boul. St. Charles, Bureau 27C Kirkland, Québec Canada H9H 3C3	MTB-0-02210
	11. N° de contrat - TPSGC ou Transports Canada T8200-0-0558/001/MTB

12. Nom et adresse de l'organisme parrain	13. Genre de publication et période visée
Centre de développement des transports (CDT) 800, boul. René-Lévesque Ouest Bureau 600 Montréal (Québec) H3B 1X9	Final
	14. Agent de projet C. Gautier

15. Remarques additionnelles (programmes de financement, titres de publications connexes, etc.)

Projet coparrainé par le Programme de recherche et développement énergétiques (PRDE)

16. Résumé

Naviguer en eaux couvertes de glaces est une tâche risquée et dangereuse. La nécessité de développer un simulateur de navigation dans les glaces découle d'une étude menée par le Centre de développement des transports, qui a déterminé les critères que devrait respecter un cours de formation à la navigation dans les glaces.

Dans la phase III du projet, on a démontré qu'il était possible d'améliorer la performance du système en divisant le monde virtuel en parties plus petites, plus faciles à gérer. Les chercheurs y sont parvenus en restructurant et en modifiant le logiciel et les bases de données du simulateur.

De plus, la simulation du pont du navire vu de la passerelle a été réaménagée pour répondre au besoin d'une visualisation à plus grand réalisme des divers types de glaces à l'horizon. Il s'agissait principalement d'appliquer des algorithmes de mappage différents à des cartes de textures, dimensions, échelles et contenus variés.

La plate-forme principale de simulation a été mise au point concurremment avec d'autres modules, y compris les modules d'aide à la formation et de simulation du radar embarqué.

Le logiciel principal de simulation comporte plusieurs éléments clés :

- outils de conversion des données recueillies par le radar à synthèse d'ouverture (RSO),
- outils d'analyse des données RSO,
- image virtuelle du pont du navire vu de la passerelle,
- système de gestion de la base de données,
- simulation du GPS et du gyrocompas,
- fonctionnalité instructeur et stagiaire,
- liaison avec un babillard électronique,
- adaptabilité à l'exploitation en réseau.

Jusqu'à maintenant, les chercheurs ont intégré avec succès tous les changements des différents modules constitutifs du simulateur.

17. Mots clés	18. Diffusion
Marine, radar, simulation, navigation dans les glaces	Le Centre de développement des transports dispose d'un nombre limité d'exemplaires.

19. Classification de sécurité (de cette publication)	20. Classification de sécurité (de cette page)	21. Déclassification (date)	22. Nombre de pages	23. Prix
Non classifiée	Non classifiée	—	xii, 15	Port et manutention

ACKNOWLEDGEMENTS

A number of individuals contributed to the completion of this phase of the project. I would like to extend extra special thanks to Charles Gautier of the Transportation Development Centre for his many valuable suggestions and thorough review of this and related documentation.

EXECUTIVE SUMMARY

This project was a result of a study conducted on behalf of the Canadian Coast Guard (TP 12496) that outlined the essential elements required to make up an International Ice Navigator Course. A significant component of this course involved training ice navigators with the aid of a simulator. The need for a simulator has been recognized by such countries as Finland, Russia, Sweden, Germany and Norway.

Different approaches have been considered for the implementation of this capability, hereafter called the Ice Navigation Simulator, and a number of facilities exist where large modifiable simulation engines could be used to perform this task. However, these systems use proprietary technologies and are not considered “open” systems. Besides being very expensive, they do not allow for widespread distribution and availability. Given state-of-the-art PC technology, complete with near workstation performance at a fraction of the cost, together with the explosion of multimedia and available virtual reality equipment, it is now possible to consider alternatives to the fixed, high-cost and proprietary systems currently in use for shipboard simulation. Therefore, a new approach to a realistic, low-cost implementation of the simulator was adopted.

The simulator encompasses the following elements:

- Simulation of ice on shipboard radar
- Simulation and management of remotely sensed data
- Simulation of visual aspect of ice, in daylight and night transit conditions
- Ship transit simulation (basic at this stage)
- Ice recognition and ice climatology training aids
- Ice regime entry rules training aid
- Ice Navigation systems and Electronic Charting and Display Information System (ECDIS) support

The Main Simulator Platform (MSP) ties these elements together to create an environment that visually and operationally resembles the ice navigation environment. Following the successful integration of the above items, Phase III of the project included:

- The acquisition of up-to-date PC hardware and development tools.
- Additional ice analysis.
- Segmentation of the larger worlds into smaller worlds with area of interest management.
- Improved display resolution of out-of-window display.

Although Phase III was a success, there still are improvements to the system that can and should be done to raise the simulator to a world-class tool. These include:

- Providing better feedback to the operator.
- Incorporating more training features such as the ice regime and numerology system.
- Ensuring an integrated course syllabus.

SOMMAIRE

Le projet découle d'une étude menée pour la Garde côtière canadienne (projet TP 12496), et qui établissait les principaux éléments d'un Cours international de navigation dans les glaces. Ce cours comprenait une composante importante : former les navigateurs dans les glaces à l'aide d'un simulateur. L'utilisation du simulateur pour assurer la formation a été reconnue par des pays comme la Finlande, la Russie, la Suède, l'Allemagne et la Norvège.

Différentes approches ont été étudiées pour la réalisation d'un simulateur de navigation dans les glaces. Il existe un bon nombre de systèmes qui pourraient satisfaire à ce besoin avec de puissants moteurs de simulation. Or, ces systèmes utilisent des technologies brevetées et ils ne sont pas considérés «ouverts». En plus d'être très coûteux, ils ne bénéficient ni d'une vaste distribution ni d'une grande disponibilité. Avec la technologie avancée des PC, qui sont entièrement équipés et présentent à une fraction du coût une performance proche de celle d'un poste de travail spécialisé, et compte tenu de l'explosion du multimédia et de l'équipement de réalité virtuelle offert sur le marché, on dispose maintenant de solutions de remplacement efficaces des systèmes fixes brevetés, très chers, actuellement employés pour la simulation des navires. Aussi, une nouvelle approche a-t-elle été adoptée en vue de satisfaire au besoin d'une installation de simulation à faible coût, produisant le degré de réalisme requis.

Le simulateur présente les caractéristiques suivantes :

- simulation des glaces sur le radar embarqué,
- simulation et gestion des données acquises par télédétection,
- visualisation de l'aspect de la glace, en conditions de navigation de jour et de nuit,
- simulation (élémentaire à ce stade-ci) du déplacement du navire,
- aides à la formation en reconnaissance et en climatologie des glaces,
- aide à la formation aux règles d'entrée en régime de glaces,
- systèmes de navigation dans les glaces et système de visualisation de cartes électroniques et d'information (SVCEI).

La plate-forme principale de simulation relie entre eux tous ces éléments pour créer un environnement qui reproduit visuellement, et du point de vue opérationnel, la réalité de la navigation dans les glaces. Après l'intégration réussie des caractéristiques ci-haut, le projet est entré dans la phase III, qui comprenait les activités ci-après :

- acquisition de matériel PC et d'outils de développement de pointe;
- analyses additionnelles des glaces;
- fractionnement des univers en sous-univers, pouvant être gérés en fonction des secteurs d'intérêt;
- plus grande résolution des vues de la passerelle.

Si la phase III s'est révélée un succès, d'autres améliorations sont à prévoir pour faire du simulateur un outil de classe mondiale; entre autres :

- assurer une meilleure rétroaction pour l'opérateur,
- incorporer d'autres outils de formation, par exemple le système de numérologie des glaces,
- créer un programme de cours intégré.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. Introduction

1.1. Scope

This document is the final report for e Phase III of the Ice Navigation Simulator Project. The information described in this report defines the work performed during this phase of the project.

1.2. Overview

The Main Simulator Platform (MSP) was developed in order that a full and complete training platform could be implemented for ice navigation. The overall program objective was to develop a low-cost PC-based Ice Navigation Simulation platform to train entry-level ice navigators. The MSP included the development of a scenario generation module (SGM); the base data management module (BDM), for Synthetic Aperture Radar (SAR) and remotely sensed imagery; the visual simulator module (VSM); and the user interface to the system. The MSP project included the development of interfaces to auxiliary simulator modules such as the shipboard radar simulation module (SRSM), the transit simulator module (TSM) as well as interfaces to the Ice Navigation system and the electronic charting and display information system (ECDIS). In addition, the interfaces to various other training aid modules (TAMs) were implemented during the course of this project. These included the Ice Recognition and the Ice Regime Entry Rules Training Aids.

This Phase

Up until Phase III, the Ice Navigation Simulation used the simulation worlds in their entirety. Because of this the CPU and graphics processing limits were reached and surpassed. Three factors affect performance:

1) When the world has a large number of features, the MSP server needs a longer time to go through all the features to draw and find the right ones for the simulation.
2) The VSM needs a long time and a huge memory to be able to load the entire world to the video memory and move the ship on the world.
3) The geographic extent keeps the density of ice features low.

Phase III addresses these problems. The Ice Navigation Simulation was modified to be able to divide large worlds into smaller, more manageable ones. The simulator now has the ability to divide the big world into a series of smaller worlds via the SGM. A user simply locates the ship on any place in the big world overview and the MSP will load the corresponding smaller world automatically. At runtime the MSP also automatically switches to the adjacent smaller world when the ship crosses the boundary defined by two adjoining smaller worlds.

As the MSP loads another smaller world, it simultaneously updates the world information. The VSM checks the world information to determine whether the world is changed. When the VSM finds that the MSP has loaded a new world, it pauses the current visual simulation, releases and frees the current world's objects from the scene, requests new world information from the MSP, reloads and then adds the new world to the scene, ands resume the visual simulation. Another benefit to utilizing smaller worlds is that it is easier for the Visual Generation Module (VGM) to create smaller OpenFlight databases than larger ones. Each feature is composed of several triangles. A world that has a large number of features needs a huge number of triangles. The OpenFlight database will therefore include a huge number of polygon nodes. The OpenGVS API used in the VSM for 3D programming has to work on these huge nodes at a slow speed with many resources.

1.3. Document Overview

This document contains a brief description of the changes and additions to the main simulator software elements since the beginning of Phase III. The document is divided into several parts based on subsystem, with each part representing a series of design and implementation activities, and ends with a section containing the conclusion and recommendations for the system.

Additional SAR Analysis (Section 3) describes:

- In tabular form, the SAR analysis work performed by Enfotec Technical Services, as well as the performance gains achieved by dividing the large worlds into smaller ones.

SGM Modifications (Section 4) describes:

- Addition of new world parameters.
- Division of big world into smaller worlds.
- Creation of a new file type.

MSP Modifications (Section 5) describes:

- Modification of overview and detail display.
- Regeneration of the height file required by SRSM server.
- Addition of a C++ class to deal with small worlds.
- Modification of design and addition of code to change world dynamically.
- Signaling of VSM that world has changed.

VSM Modifications (Section 6) describes:

- Mechanism to get new world information from MSP.
- Design changes to support changing worlds dynamically (freeing and loading worlds).

Other Modifications (Section 7) describes:

- New file type header structure.
- Changes to the world information structure.

2. MSP General Architecture

The MSP provides the data to and interfaces with various subsystems. The MSP software is composed of several components and dynamic link libraries. Figure 2-1 illustrates the relationship of the various software components that make up the Ice Navigation Simulator.

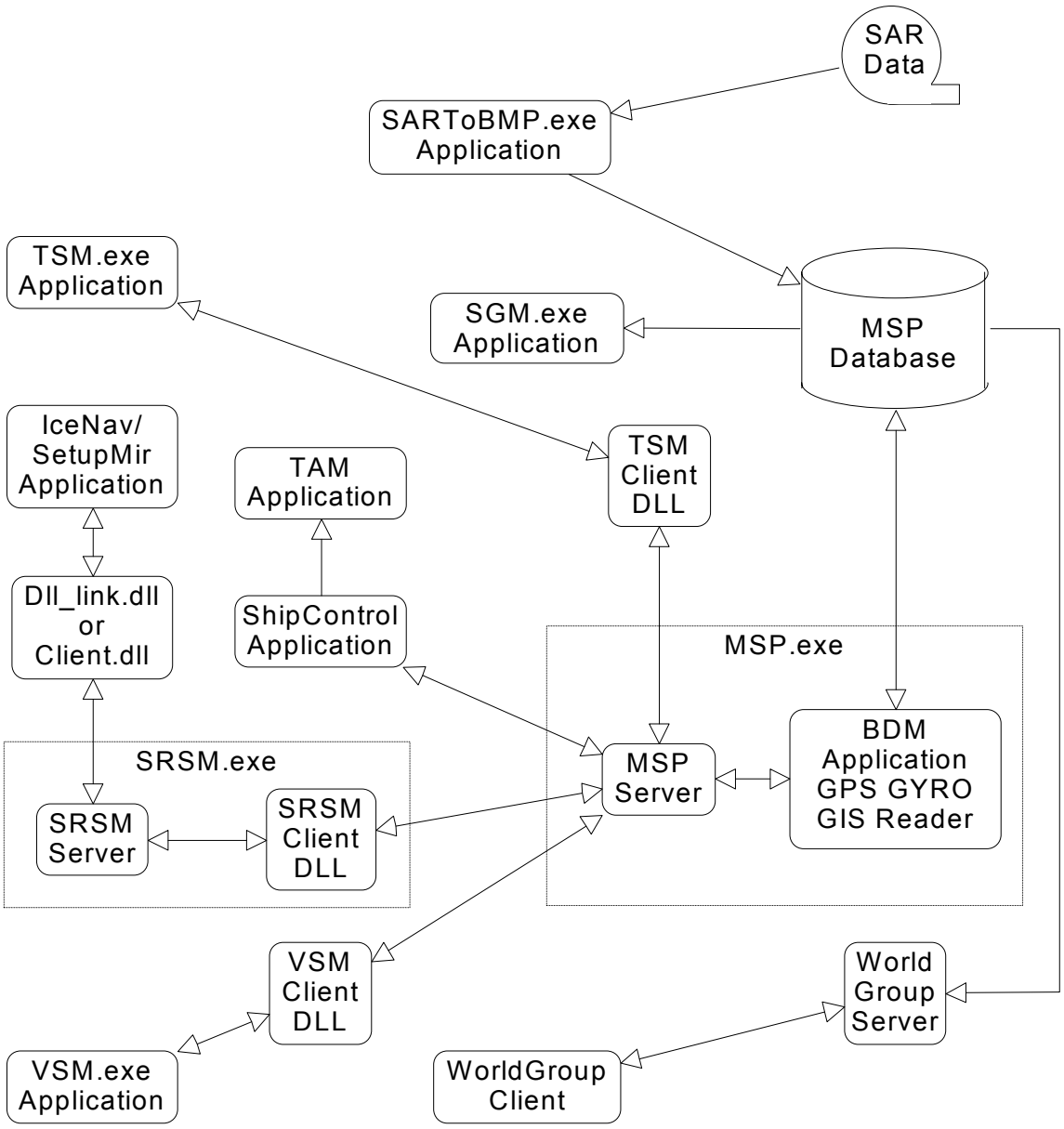

Figure 2-1 Simulator Applications and DLLs

3. Additional SAR Analysis

An ice expert from Enfotec Technical Services Inc. was contracted to provide additional ice feature information. The addition of features was required to improve the variety of ice imagery used in a scene.

3.1. Description

For example, the “Gulf” world now has 4058 features. This large world was divided into five smaller worlds with 200 scan lines of overlap between adjacent small worlds. Tables 3.1-1 and 3.1-2 show the differences in work that need to be performed by both the VGM and the VSM for the big Gulf World verses the smaller Gulf Worlds.

Table 3.1- 1 VGM Performance – Big versus Smaller Worlds (Gulf)

	Gulf Small World						Gulf World (Big World)
	1	2	3	4	5	Total	
Number of scan lines	4,200	4,200	4,200	4,200	4,000	28,000	20,000
Number of features	321	775	1,100	1,200	893	4,289	4,058
VGM Memory Usage (Kilobytes)	14,324	26,564	36,640	45,886	30,478	45,886 (max)	50,620
System Memory Peak (Kilobytes)	55,328	65,820	85,876	94,538	78,056	94,538 (max)	223,824
Number of triangles	8,363	16,831	26,140	34,511	24,455	110,300	106,123
VGM Process Time (minutes)	2.5	4	5	5.5	4.5	21.5	> 47
OpenFlight Database Size (Kilobytes)	1,444	2,836	4,222	5,521	3,968	17,991	17,155

Table 3.1- 2 VSM Performance – Big versus Small World (Gulf)

	Gulf Big World	Gulf Small World
World Load Time (h:m:s)	0:02:22	0:01:23
World Change Time (h:m:s)	None	< 0:00:10
VSM Memory Usage (Kilobytes)	31,420	13,196
System Memory Peak (Kilobytes)	97,128	87,672
Frame Update Time (millisecond)	46 – 70	46 – 47

Another example is the “Parry” world, which has a total of 7488 features. It was also divided into five smaller worlds, each with 300 scan lines of overlap between any two adjacent small worlds. Tables 3.1-3 and 3.1-4 show the differences in work that need to be performed by both the VGM and the VSM for the big Parry World verses the smaller Parry Worlds.

Table 3.1- 3 VGM Performance – Big versus Smaller Worlds (Parry)

	Parry Small World						Parry World (Big World)
	1	2	3	4	5	Total	
Number of scan lines	7,215	7,215	7,215	7,215	6,915	35,775	34,576
Number of features	2,058	2,050	1,888	1,004	944	7,944	7,488
VGM Memory Usage (Kilobytes)	181,218	148,218	127,584	90,6321	90,4171	181,218 (max)	235,972
System Memory Peak (Kilobytes)	247,696	223,900	198,460	142,044	129,532	247,696 (max)	761,700
Number of triangles	53,605	47,928	42,024	25,641	25,534	194,734	186,397
VGM Process Time (minutes)	5.5	4.8	4.5	1.6	1.7	18.1	> 130
OpenFlight Database Size (Kilobytes)	17,296	15,668	13,853	8,369	8,304	63,490	60,518

Table 3.1- 4 VSM Performance – Big versus Small World (Parry)

	Parry Big World	Parry Small World
World Load Time	0:05:12	0:01:50
World Change Time	None	< 0:00:20
VSM Memory Usage (k)	91,716	34,044
System Memory Peak (k)	136,144	69,848
Frame Update Time (millisecond)	46 - 67	46 – 47

4. SGM Modifications

Users can add, modify and delete features in a scenario file or world by running the SGM application. As users add more and more features into the world, they may find that the world becomes too big for the simulation engine to handle. To overcome this, they can use the newly added “Utilities” command called “Divide Into Small Worlds” to partition a large feature-laden world into a set of smaller worlds. These are all interconnected via a newly generated index file whose extension is “map”. The functional modifications, software design and implementation updates of the SGM are described in sections 4.1 through 4.3.

4.1. Divide World Dialog Box

When the user clicks the “Divide Into Small World” command, the “Divide the Current World” dialog will pop up (see Figure 4.1). Three sections make up this dialog.

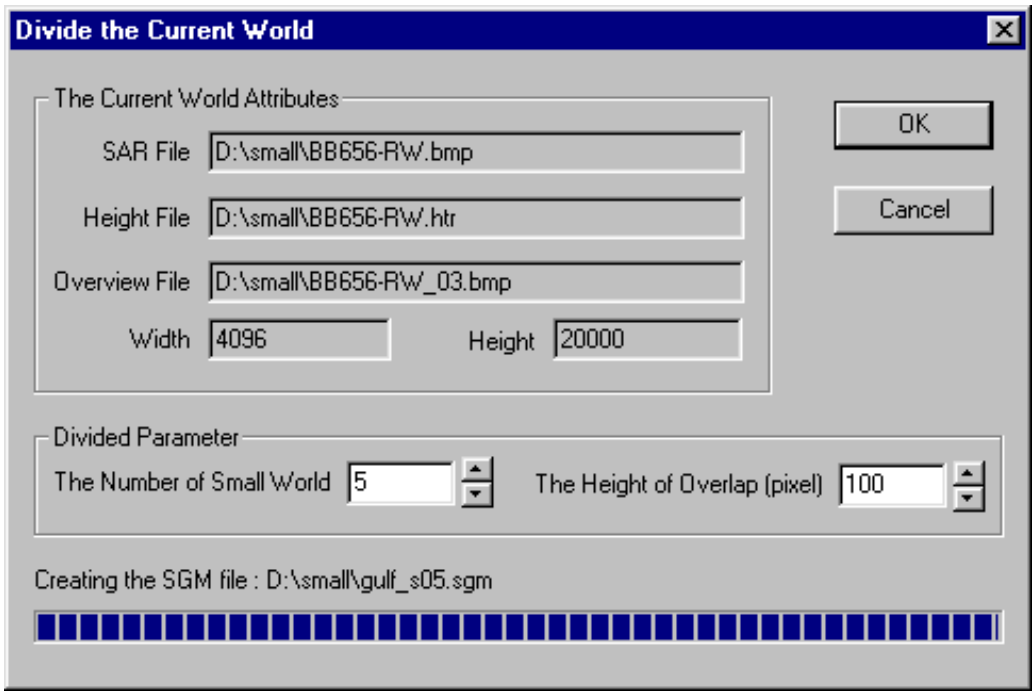

Figure 2.1 “Divide the Current World” Dialog Box

- The Current World Attributes

  This part shows some properties of the current world.

  “SAR File” is the current world’s SAR bitmap file.
  “Height File” is the current world’s height raster file.
  “Overview file” displays in the overview window in the MSP.
  “Width” and “Height” show the current world size in pixels.

- Divided Parameter(s)

  These parameters are used to decide how to divide the current world.

  "Number of Small Worlds" is used to determine into how many parts the world is to be divided.

  "Height of Overlap" is used to determine how many scan lines are to be in the overlap section of two small worlds.

- The status or progress bar

  This part shows the progress or status while the SGM is partitioning the big world. It is displayed only after the user clicks the "OK" button.

4.2. New File Types

When the SGM has finished dividing the current world, it creates the following new files:

- A new type file with "map" extension. This file is used to record the number of small worlds, the scan line number of overlap, the big world size, the big world height file and overview file, and each small world's SGM file name. When the MSP opens a "map" file, the user can work on a big world composed of multiple small worlds. The MSP manages all of the details for interchanging between small worlds. The MSP still uses the big world's overview bitmap file and existing height files for better performance.
- Small world SAR files. Each title name is the big world's SAR file title name followed by the small world order number. For example, if the big world's SAR file name is "D:\Small\BB656-RW.bmp", the SAR file name for the small world with the order number 2 is "D:\Small\BB656-RW_s02.bmp"
- Small world feature files. These files are created from the big world's feature file. All the features in the big world are cut by the small world's border rectangle. All the features that are entirely or partly inside of the border rectangle make up the new small world feature file.
- Small world "sgm" files. These "sgm" files all have the same structure as that of the big world, so that the SGM and MSP can open them and work on them like the other worlds.

4.3. Detailed Design Modifications to SGM

4.3.1. MAP File Type Structure

The MAPSTRUCT structure was defined to save the header information of the MAP file when a big world is cut into a set of smaller worlds. It has the following data members:

int *filetype*	Type of file.
int *width*	Width of the big world.
int *height*	Height of the big world.
int *overlap*	The scan lines in the overlap section of the two smaller worlds.
int *divnum*	Number of smaller worlds.

Following the header information is each small world's SGM file path name. These are NULL terminated strings.

Following the SGM file path name(s) is the height file name and overview file name of the big world. The following table shows the structure of a MAP file.

MAP File Structure

Header Information (MAPSTRUCT)
Each smaller world's SGM file path name
Big world height file path name
Big world overview file path name

4.3.2. DivideWorld Class

This class is used to divide the big world into the smaller ones. The following functions are used to perform this work:

void **SetCurrentWorldData**(CString paraImageFile, CString paraGeoFile,
CString paraHeightFile, CString paraFeatureFile,
CString paraDescript, CString paraDate, CString paraTime,
CString paraASCFile, float paraRot, double paraRes,
int paraHeight, int paraWidth);

This function is used to set the parameters about the current world, where

paraImageFile	is the bitmap file path name.
paraGeoFile	is the geographic file path name.
paraHeightFile	is the height raster file path name.
paraFeatureFile	is the feature file path name.
paraDescript	is the description of the world.
paraDate	is the date of creating the world.
paraTime	is the time of creating the world.
paraASCFile	is the ASCII file path name.
paraRot	is the big world rotation.
paraRes	is the big world resolution.
paraHeight	is the big world height
paraWidth	is the big world width

BOOL **CutBmp**(CFile &sFile, CFile &dFile, int Height, int StartLine);

This function is used to "cut" a smaller bitmap from a big bitmap, where

sFile	is the source bitmap file.
dFile	is the small bitmap file that will be created.
Height	is the height of the small bitmap file in lines.
*StartLin*e	is the line number on the big bitmap.

int **DivideWorlds**();

This function is used to divide the current world into smaller ones. The following files are created by this function:

- small world raster files
- small world geographic files
- small world feature files
- small world SGM files
- the MAP file

4.3.3. PolygonCaculate Class

This class is used to cut the features by the small world border. All features entirely or partly inside of the border rectangle compose the new small world feature file. Functions in this class are:

static int **DividePolygon**(CPoint *pBox, CPoint *pPolygon, int PolygonPointNum,
CPoint **pDividedPolygon, int *pDividedPolygonStart,
int *pDividedPolygonNum);

This function divides a polygon by a rectangle. In this function,

pBox	is the 4 dividing box points.
pPolygon	is the polygon points.
PolygonPointNum	is the number of the polygon points.
pDividedPolygon	is all divided polygon points.
pDividedPolygonStart	is the start location for each divided polygon in *pDividedPolygon*.
PDividedPolygonNum	is the number of the divided polygon.

static int **TwoLineIntersectPoint**(CPoint Line1Head, CPoint Line1Tail,
CPoint Line2Head, CPoint Line2Tail,
CPoint &IntersctPoint, double &t);

This function calculates the intersection point of two lines.

Line1Head	is the start point of the first line.
Line1Tail	is the end point of the first line.
Line2Head	is the start point of the second line.
Line2Tail	is the end point of the second line.
IntersctPoint	is the intersect point.
t	is the distance between intersect and the start point on the second line.

static int **TestInsect**(CPoint Poly_LT, CPoint Poly_BR, CPoint Div_LT, CPoint Div_BR);

This function tests if two rectangles intersect.

Poly_LT	is the left top point of first rectangle.
Poly_BR	is the right bottom point of first rectangle.
DIV_LT	is the left top point of second rectangle.
DIV_BR	is the right bottom point of second rectangle.

5. MSP Modifications

In the MSP, when starting a new simulation session, users can select the new type file, which has the extension "*.map". This "map" file records the data about the big world and information on its divided small worlds. The details of the "map" file structure are described in section 4.2.

When a user selects a "map" file for the simulation, the MSP will display the world's overview in the left "overview" panel and load a small world into the "detail" panel on the right. All of the features of the world will be displayed on the "overview" panel when the user stops recording to a new ship position. Only the active small world features are displayed on the "overview" panel when the MSP is recording or playing back a simulation.

If, when positioning the ship in the world, a user clicks on the overlapping section, the MSP will pop up a window for the user to select a small world. As the recording or playing back simulation starts/resumes, the MSP will automatically change to the alternate small world when the ship crosses a trigger point in the overlap. When this happens, information about the new loaded world is added in a message that is sent to the VSM. The VSM uses this information to change the visual world.

5.1. Detailed Design Modifications to MSP

5.1.1. New Class

Two classes are added to the MSP module to allow the MSP to deal with a new "map" type file and switch among different worlds more efficiently.

- **CEntireWorld class**: This class is used to manage the relationship between the small worlds and the big world, and to set the parameters required for swapping the small worlds.

- **CSelectSGMFile class**: This class handles the smaller world selection dialog when the user places a ship in overlapping sections of the world and will therefore allow the users to choose which world they want to load.

5.1.2. Data Member Added in ENVIRON_INFO Structure

A data member named *change_World* was added to the ENVIRON_INFO data structure for the purpose of notifying the VSM about a change in worlds from the MSP. When the MSP loads another world; this data member increases by one. "*change_World*" increases its value only when the MSP loads a world that is different from the previous world. By doing this the VSM doesn't need to change worlds if the MSP restarts with the same world.

5.1.3. EntireWorld Class

This class is used to load a "map" file and save the information about the entire world. It also includes some functions to get the entire world attribute. The following are functions in this class:

int **GetNewWorld**(CFile &paraWorldFile);
This function is used to read the MAP file and get the header information and SGM file path name. (The MAP file details refer to section 2.2.2).
paraWorldFile is the MAP file path name.

int **SetCurrentWorld**(int paraIndex, CString &paraSGMFile);
This function sets the current world by the *paraIndex* and returns the current world SGM file path name.

paraIndex	is the index of the world to be set.
paraSGMFile	is the world path name with the index number *paraIndex*.

int **GetCurrentPart**();
This function is used to get the current world index number.

int **GetCurrentRect**(CRect &paraRect);
This function is used to get the border rectangle for the current world.

paraRect	is the current world rectangle.

CPoint **GetCurrentTopLeft**();
The return value of this function is the top left coordinate of the current world.

int **GetWorldHeight**(),
int **GetWorldWidth**();
These two functions are used to get the height and the width of the entire world in pixels.

int **GetEntireCoord**(CPoint paraOrigin, CPoint &paraNew);
int **GetEntireCoord**(int paraOriginX, int paraOriginY, int &paraNewX, int &paraNewY);
These overloaded functions are used to change the small world coordinates to the entire or big world coordinate.

paraOrigin	is the small world coordinate.
paraNew	is the entire world coordinate.
paraOriginX	is the small world coordinate X value.
paraOriginY	is the small world coordinate Y value.
paraNewX	is the entire world coordinate X value.
paraNewY	is the entire world coordinate Y value.

void **ChangeToCurrentPoint**(CPoint &paraPoint, int paraPart);
This function is used to change the ship coordinates from the entire world to the current world.

paraPoint	input value is the entire world coordinate, output value is the coordinate in the current world.
paraPart	is the world index number for the current world.

BOOL **CheckReachEdge**(CPoint &shipPos);
This function is used if the ship reaches the edge beyond which the world must be changed. At present, the edge is set to the midway of the overlap (if one exists) or the edge of the current world if there is no overlap.

shipPos	is the current ship position.

int **CheckifChangeWorld**(CPoint &shipPos);
This function is used if the ship reaches the world's edge. It checks whether there is another world for the ship to change to.

shipPos	is the current ship position.

int **CheckPointInWorld**(CPoint paraPoint, CString &paraSGMFile);
This function is used to check which small world(s) contains the current point. If the point is located in the current world, the return value indicates that the world will not need to change. If it is located in another world, the return value indicates that the world needs to be changed. The *paraSGMFile* variable returns the changed world SGM file path name. If the point is located in

the overlapping section of the worlds, a dialog will be displayed to prompt the user to select from one of the two overlapping worlds.

paraPoint is the coordinate of the point.
paraSGMFile is the path name of the world to be changed to, if it is available.

5.1.4. Modification to MspDoc Class

As the ship moves, the MSP needs to check whether the ship has reached the world's edge and whether there is a new world for the ship to change to. Also, when changing the world, the MSP needs to change the coordinates of the ship and features for it to display correctly. The following C++ functions were added to the MspDoc class to implement this functionality.

void **CheckIfChangeWorld**();
This function is called from the OnTimer() function when the ship is moving. It will check the following conditions:

- Whether the world is a single or whether there are multiple worlds.
- Whether the ship has reached the edge of a world.
- Whether there is a world for the ship to change to.

If all the conditions are satisfied, the world is changed.

void **ChangeWorld**(int newWorldIndex);
This function is used when the CheckIfChangeWorld() function finds that the world needs to be changed. This function will perform the following tasks:

- Calculate the ship coordinates transformation when changing the world to keep the ship in the same place.
- Open the new world SGM file.
- Set changing world information for the VSM.

NewWorldIndex is the index number of the new world that the MSP loads.

void **CheckPointInWorld**(CPoint paraPoint);
This function is used to determine in which of the small worlds the specific point is located. If the point is not in the current world, then another SGM file is opened.

paraPoint is the coordinate of the point the user selected.

void **ChangeToCurrentPoint**(CPoint &paraPoint, int paraPart);
This function is used to change the old coordinate to the coordinate of the current world.

paraPoint is the coordinate of the point to be changed.
paraPart is the world index number.

5.1.5. SeleSGMfile Class

This class is used to display a dialog box for the user to select an SGM file when the user clicks on the overlap section of the worlds.

6. VSM Modifications

6.1. Task Description

The VSM module handles the 3D simulation. Its tasks include loading the world and the ship 3D model, positioning and moving the ship model within the world model, simulating the sunlight, environmental conditions, and so on. The VSM is a real-time application that requires an abundance of memory, CPU speed and graphics processing to maintain the simulation screen refresh rate.

For a CRT-based system, the VSM needs to maintain a 20-hertz screen refresh rate to display smooth motion. This rate increases to 60 hertz for a helmet mounted display (HMD) because the possible occurrence of motion sickness needs to be minimized. Motion sickness in an HMD occurs when the refresh rate is < 60 hertz and is mainly a result of the sensory difference between what the eye sees and the balance mechanisms within the ear.

When the world model is large and includes many polygon nodes, the VSM needs a lot of CPU resources and time to calculate the display elements. This makes it difficult to maintain a high screen refresh rate. But when the world is small, loading and calculating it uses fewer resources and less time, and the screen refresh rate is good because there are fewer polygons to deal with. Also, changing a world takes less time; therefore, using smaller worlds allows the user to switch to a different world more quickly and with fewer CPU resource requirements.

6.2. Parameter for Changing World

Additional information, including the following, was added to the data that the VSM gets from the MSP.

World number. The MSP gives the VSM the current world a number. When the MSP loads a different world, it will give the VSM a new world number. The VSM checks this number to see whether it needs to change the visual world.

6.3. Changing World Process

When the VSM needs to change the world, it pauses the simulation and asks the MSP for the current world name and the current ship name. After freeing the old model, the VSM reloads the current world model and resumes the simulation.

The changing mechanism is not only used for small worlds. The MSP saves the world number in a file. When the MSP closes and restarts, it uses a new world number to inform the VSM changing the world. The VSM no longer needs to restart itself to respond to the MSP changing the world.

6.4. Description for Modification

The VSM communicates regularly with the MSP to get the new data about the world and the ship before it recalculates the display. There is an item called “change_World” in the data. The VSM checks this item each time to detremine whether the MSP has changed the world. If it finds that the MSP has changed the world, the VSM pauses the simulation, asks the MSP for the current world name and the current ship name, frees the old model, reloads the current world and ship models, and then resumes the simulation.

6.4.1. Modification to CWinVSMDlg Class

void **ChangeWorld**()
This function is used when the VSM is changing the world. It pauses the simulation and calls the initialization function to reload the new world model.

int **m_ChangeWorld**;
This new data member is used to pause the simulation. When the VSM is changing the world, it sets this data member to stop the simulation. After finishing the change, the VSM resets this data member to continue the simulation.

6.4.2. Modification to CVSM Class

int **ChangeWorldModel**()
This function is used to change the world and ship model. At first, it releases the current world and ship models from the scene. Next, it frees the current world and ship models. The function then reloads the new world and ship models. Finally, it adds the new models into the scene.

6.4.3. Modification to CMSPClient Class

BOOL **PreTranslateMessage**(MSG* pMsg)
Code was added in this function to check whether the MSP has changed the current world. If the MSP detects a change to the current world, this function pauses the simulation and sends a message to reload the new world model.
pMsg is a pointer that contains the message to process.

7. Other Modifications

7.1. Open New World

Modifications to the function **OnNewDocument**() of the MspDoc class and **OnBrowser**() of the MSPStartUp class allow users to select different world.

7.2. Processing of Two File Types

There are now two kinds of files – SGM files and MAP files. These file types need to be processed differently; therefore, one function was added to the MspDoc class to deal with this difference. The added function is:

BOOL **OpenSGMWorldFile**(CString &sgmWorldName);
This function is called when a MAP-*type file is opened. It gets initialization information concerning the opening of multiple worlds before opening individual SGM files.
sgmWorldName is the path name of the MAP file.

7.3. Locate the Ship on Different World

The function **OnLButtonDown** (UINT nFlags, CPonit point) in the MspView class was modified to allow a user to locate a ship in different small worlds. The MSP loads the world that the user selects when the user clicks on the different world.

8. Conclusion and Recommendations

The project was an overall success. Improvements, however, can still be made to the quality of the visual worlds, including the realism of the scenes (e.g., Ship's Track and Environmental Conditions). The basic framework is functional and could be made field-operative and useful today. The following improvements would elevate the simulator from strictly an R&D effort to a world-class simulation and training tool.

- Addition of a Ship's Console to display the operating parameters and instrumentation while navigating through the ice.
- Display of Ship's track on out-of-window view AND radar display to allow users to see where they have been.
- Bridge Sound to provide engine control feedback.
- Addition of image resolution synthesis of radar data for lower radar range settings.
- Addition of more ship types to train operators in the same ice conditions but with different hull and propulsion strengths.
- Assessment and/or warning of potential damage to a ship if it hits ice (e.g., a bergy bit) at too high a speed.
- Additional visual and scenario files. There are currently only two worlds (Parry and Gulf). More variety is required.
- An integrated course syllabus that would lead the student through a set of standardized exercises.
- Investigation into how multiple threads could be used to load the world model in the VSM so as to be able to load the next world model in background thread without affecting the main screen refresh thread. When the VSM needs to change the world, it could get the model that is already in memory. At present the OpenGL APIs used in the VSM do not support multiple threads. New 3D APIs may be required to use multiple threads.
- Extension of the small world handling mechanism to connect multiple big worlds together. This way, a ship could go through all the worlds without loading a new scenario file. This function would also be convenient to build an entire world map, the details of which the MSP would handle for the users.
- Incorporation of other ship types and classes. There is currently only one: the M.V. Arctic.
- Incorporation of more training features such as the ice numerology system, or a Red, Green and Yellow indicator system indicating dangerous conditions, hazards, warnings or to proceed with caution. Currently there is no indication that the operator is entering a dangerous area.
- Motion to add the physical realism of being on the ship's deck.