Nautilus – The Environment for Training and Testing

Jiří Chludil Jiří Žára
Department of Computer Science and Engineering,
Czech Technical University in Prague
{xchludil,zara}@fel.cvut.cz

Abstract

The paper describes an experimental web-based environment for teaching and testing. The application named Nautilus has been developed using Virtual Reality Modeling Language (VRML) and Java language with two special libraries: DILEWA and Vrmlworld. Although the resulting system is intended for low-level virtual reality systems without any specialized hardware support it is powerful enough to accurately simulate various situations and scenarios. Currently implemented multi-user environment for training and testing of yacht captains serves as an experimental workbench for creation of general applications where a developer combines a simulated environment from simpler simulation elements. Design and implementation of the system are presented together with several practical observations concerning efficiency of the real-time rendering and the level of implementation difficulty.

1. Introduction

Three years ago, the Czech Offshore Yachting Association asked the Department of Computer Science and Engineering at the Czech Technical University in Prague for a study about possible utilization of virtual reality for training and testing of candidates for yacht captains. Although some testing applications exist [8], none of them has enabled web-based interface so far; moreover, they do not support more then one user and their methods of displaying are not true 3D.

Recently we have designed and developed an application allowing multi-user simulation of ships on sea in various scripted situations [3]. Nautilus is based on cooperation of VRML [5] and Java applet interconnected through EAI interface [6] with Vrmlworld library extension. For multi-user purposes we use DILEWA library [7]. Both Vrmlworld and DILEWA library have been developed at our department and will be described later in the text. The application is going to be practically used as an environment for yacht captain tests.

Short introduction to yacht simulator can be found in Section 2. Architecture of the application is described in Section 3. Section 4 introduces methology of new models development (virtual objects and effects). Section 5 discusses issues to be solved in the future and concludes the paper.

2. Motivation

Yacht training and testing application is based on the developed environment described in this paper. We use this application as a typical example for the description of the architecture and the principles of the development environment.

Tests for ship recognition comprise an important part of captain tests. The ability to quickly determine a behavior of a moving ship approaching to a candidate's ship is essential from the perspective of the sea transport safety. Due to the various time and weather conditions, day, night and fog signs provide integrated information about type and activity of a ship.

During the day ships show day signals according to valid regulations. The day signs are simple shapes (spheres, cones and cylinders) with normalized sizes, proportions, colors and their combinations. They are to be displayed on specific places like top of a pole or a mast.

At night and in the evening ships use night signaling system. It is defined in such a way that it enables to unambiguously specify the position of a ship, its type and its current activity. The night signaling system consists of three position lights (red on the left, green on the right and white on back) with optional white pole lights and additional special lights informing about activities of a ship. Although the lights are to be visible from wide spatial angle, the angles and key light positions are mutually turned so that the observer can see just one of them.

In a fog and bad view conditions ships send acoustic signals repeatedly. Signals are generated either by a horn or a bell with defined frequency, magnitude and direction. Sounds are always combined with visible signs described above.

3. Implementation

The crucial requirement was that the application should support web environment. Therefore, we have chosen VRML for displaying and Java language for user interface and simulation. Another improvement was an extension of the application to a multi-user one. Since the displayed scene had been very complex, we designed a special plug-in architecture (see part 4).

Dividing the developed application into several functional parts with different purposes is apparently a very advantageous strategy [2] (see Fig. 1) in terms of complete and highly interactive virtual environment.

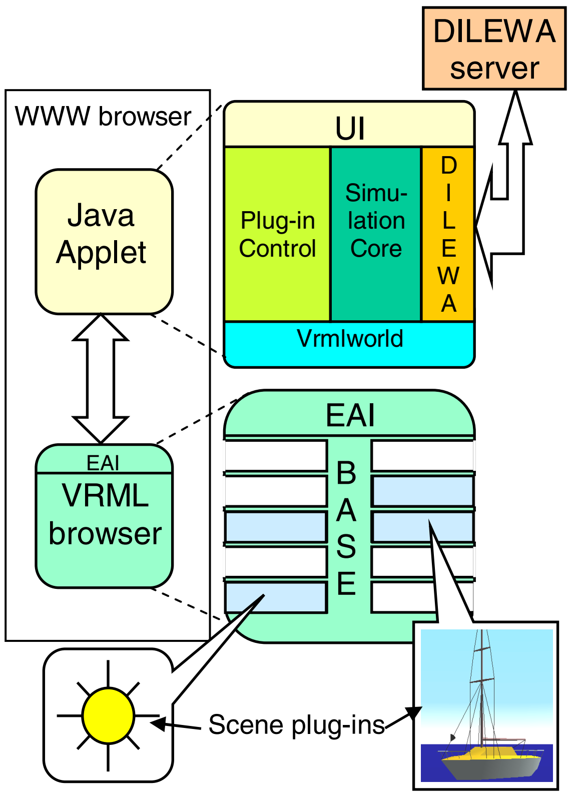


Figure 1. Modular structure of Nautilus client

3.1. Vrmlworld Library

This library serves for communication between a VRML scene and a Java code. It encapsulates a standardized programming interface called EAI (Interface between VRML and Java). Thus, the virtual scene may be controlled via Java application code in case of web browser hosting the VRML viewer. A typical web application consists of VRML browser window and additional controls in Java applet.

Unfortunately the EAI interface is too loose in terms of checking the type of VRML nodes and their fields. A frequent and hard to find programmer's mistake is sending a value to non-existing exposedField that has been referenced by misspelled identifier. Similar mistake is sending a value with wrong data type.

Therefore we have developed an object-oriented library with declarations of VRML nodes and variables. Each type of node in a VRML scene has a mirror (prototype of a Java class) in Vrmlworld library. Class initialization corresponds with a new VRML node creation or with a connection to existing named Node. Change of class attribute is represented by relevant response in a VRML node. It can also watch all changes in scene and update relevant attributes in Java classes. Vrmlworld has protection against above mentioned run-time errors. This solution enables simple access to the tree structure of VRML. In addition, Vrmlworld allows obtaining of hidden information about the scene e.g. non-named child nodes. This library dramatically reduces a size of the source code in Java (to 25%).

The library contains several special functions intended especially for debugging of communication between Java application code and a VRML scene. Typical examples: a function that allows extracting tree of the VRML scene with current values of node variables, finding routes between nodes, scene structure management.

3.2. DILEWA Library

The name DILEWA stands for the **DI**stributed **L**earning **E**nvironment **W**ithout **A**vatars. It is a Java library developed at our Department allowing simple and straightforward extension from a single-user VRML application to a multi-user distributed one. Library is currently used in the Nautilus application for two purposes: firstly, synchronization of the state of the scene (positions, trajectories and configurations of properties of ships, changes of environments, e.g. fog, sun, sky, clouds, etc.), and secondly, communication among users (text-based chat). This use of DILEWA increases the interactivity of the teaching process either by immediate interventions of the teacher/examiner or by introducing multiple candidates in the same sea space concurrently (see Fig 2).

3.2.1. DILEWA Architecture. The DILEWA library serves for multiple users connected via Internet to share one virtual world described e.g. by VRML.

Users can interact and manipulate with the VRML world and these events are distributed to others. The virtual worlds displayed on individual screens of the users are thus immediately updated. To minimize network bandwidth requirements, DILEWA is used to distribute only events in the scene while simulation computation and scene rendering are executed on individual workplaces.

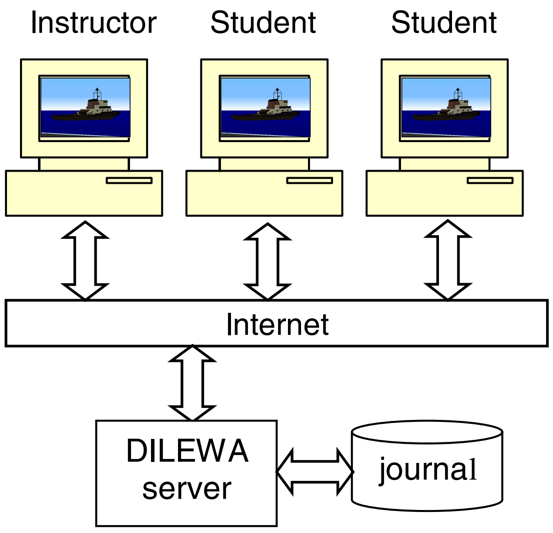


Figure 2. DILEWA communication scheme

The communication among distributed users is based on server-client strategy. The DILEWA server, implemented in Java, is responsible for event broadcasting and basic user management. The server can hold a history of selected events performed during the distributed session so that later joined users receive fully updated virtual scenes. To further reduce the amount of distributed information; dead reckoning is used to transmit status data less frequently.

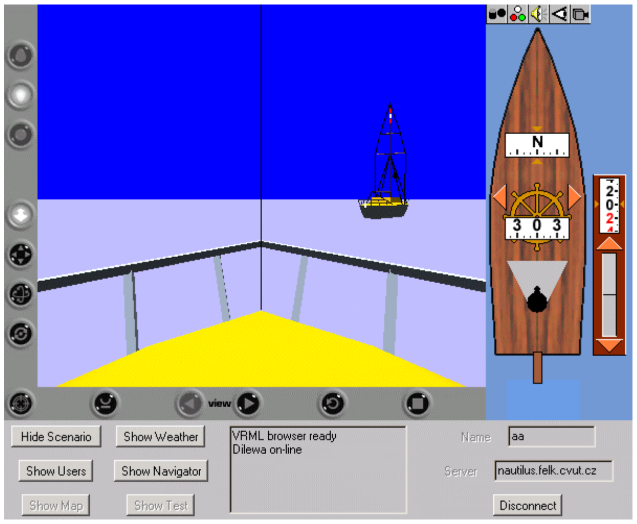
2.2.2. Users and their Roles. In the DILEWA, users are identified by their names/nicknames and the server maintains a list of active users. This identification serves as a key to access additional information about users (their roles, modes, scene area, events in scene, etc.). A user with a special permission obtains the instructor/examiner role and the others become students/candidates. Users can change their scene area that contains various scenarios with different difficulty, weather and traffic conditions, etc.

User in student role can see the list of users connected to visited scene area, while the user in instructor role can see the list of all users connected to Nautilus.

Instructor works in three modes – the first mode is a standard navigation mode allowing changing user position without interacting with the scene (ghost). The second mode allows to become a captain of a ship and to interact with the scene and other users. The third mode is curious but highly beneficial – it allows to select an arbitrary user and to watch the virtual world through his/her eyes. In this mode, all interactions and viewpoint changes performed by the selected user are sent to observer. This is extremely useful for the instructor.

3.3. User Interface

The user interface (UI) Nautilus comprises several dialogues for application setup, simulation control and evaluations (example is shown on Figure 3). The UI provides two user roles – student and instructor. The student role is limited to training and testing in predefined scenarios, while the role for instructor allows editing of scenarios (starting position, courses and speeds of all ships, weather setup, accident events etc.), students monitoring (see also part 3.2.2) and classification of a student being tested



VRML browser ready
Dilewa on-line

Name: aa
Server: nautilus.felk.cvut.cz

Figure 3. Nautilus User Interface

3.4. Simulation Core

Computation of simulation is usually very demanding to computer time and performance. Simulation of the entire scene in real-time is very difficulted [1]. Therefore we have split its computation into two phases. Firstly, full simulation is preprocessed using special software Dynast [4]. The resulting relations and values are stored in form of simple samples. Final behavior of objects is simulated in simulation core, based on the samples obtained real-time presentation.

3.5. Plug-in Control

Several strategies exist for managing a scene in VRML: firstly, whole scene is loaded into a memory of browser and the whole scene is displayed; secondly, whole scene is loaded into the memory but only a part of scene is displayed; thirdly, parts of scene are successively loaded into browser and displayed (scene set up like a puzzle). Obviously, the third scene strategy is the most effective from the point of memory, speed of rendering and variability of scene. It puts, however, higher demands on the entire scene management.

Architecture of plug-ins has been chosen because of easier management. Management of plug-ins is implemented in "Plug-in control" layer. Via Plug-in control it is possible to add plug-ins to scene, communicate with them and remove them. Example of communication sequence shows Figure 4. Universal interface was designed for individual types of plug-ins since the communication is relatively simple.

We have two basic types of plug-ins. The first one represents a model of a ship with its properties (see part 3.6.1), the second one represents nature phenomenon (see part 3.6.2). The second type is difficult for the management because of mutual dependence of natural phenomena (e.g. clouds decrease intensity of scene illumination). Therefore, plug-ins are capable of mutual communication (the communication is provided by the routes among plug-ins).

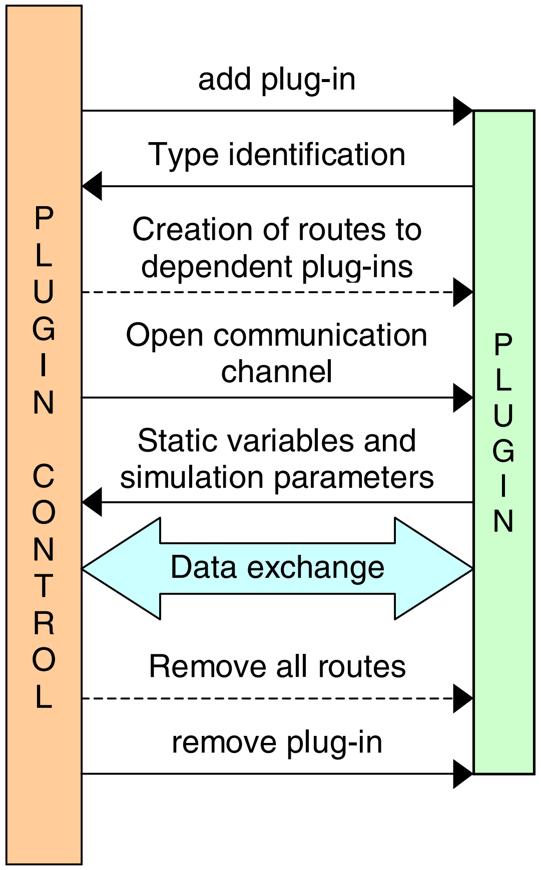

PLUGIN CONTROL
PLUGIN
add plug-in
Type identification
Creation of routes to dependent plug-ins
Open communication channel
Static variables and simulation parameters
Data exchange
Remove all routes
remove plug-in

Figure 4. Communication between Plug-in control and Plug-in

3.6. Scene Plug-ins

Architecture based on plug-ins is advantageous from the point of simple scene management, easy extendability and large adaptability. Plug-ins communicate with the base through an interface that is represented by a table of variables. These variables are divided into four categories: static variables (convex envelope, numbers of day signs etc.), variables for simulation (dependence of maneuverability on the speed of a ship), variables with low frequency of changes (index of active day sign), and finally variables with high frequency of changes (position and orientation of a ship).

3.6.1 Type "Object". This plug-in is determined for representation of a ship. It contains two types of data. The first one is the model of ship itself and its signs, lights, effects etc. All these models are described in VRML language. The second one is information describing the ship from the viewpoint of simulation (parameters concerning performance of engine, maneuverability, formulas for simulation of wind effect to ship, the number of day, night and sound signs, etc.) The data are in a form of constants, vectors and tensors. All of these parameters are obtained from the preprocessed and full he simulation of ship model in standalone application Dynast (see part 4.3). All of these parameters are used for the real-time, but also very precise simulation.

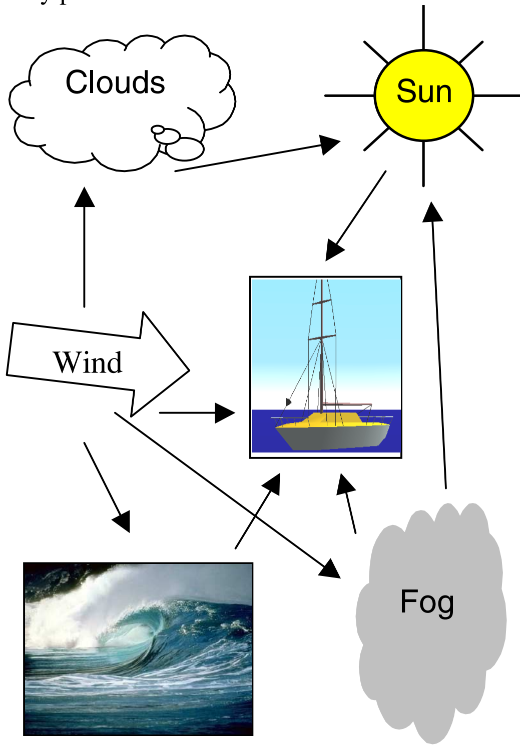


Figure 5. Diagram of effect and object dependence in real world

3.6.2 Type "Effect". This plug-in is determined for representation of nature phenomena. It contains three types of data. The first are models of the effects (model of cloud etc.). Such models are again written in VRML language. The second one is information that describes phenomena from the viewpoint of simulation (specialization of effect behavior). These data are in the form of constants, vectors, tensors.

The third type of data is expression of dependencies among the phenomena (see Figure 5) (e.g. how the wind influences waves).

4. Plug-in Development

Although the VRML file format has been published five years ago, no universal and widely available 3D editor can be found and used for our purposes. The following text introduces several implementation issues, methods and practical advices.

4.1. VRML Model

There are two different ways how we created models of ships in VRML format. In the first way we use a VRML text editor (e.g. VrmlPad). This text editor supports VRML language and makes it possible to automate function excerption. Final model can be saved in compressed *wrl* format. This solution demands high level of imagination, but model's code is small and effective.

Second way of creating ship models consists in using any 3D graphics program (Cinema4D, Rhino3D, trueSpace, 3DStudioMAX, etc.). This solution results in well-developed model of ship. Unfortunately, code of model is not effective and 3D editors do not support VRML scripts. Taking into account the advantages of both approaches we have decided to use 3D graphics program for design of ships and then we optimize it in VrmlPad. For writing scripts and new interface we used solely the VrmlPad.

4.2. Simulation Parameters

Interpolation of relation (see Figure 6) was used for simplicity; this method is based on interpolation of dependence of the third-order polynomial equation. Parameters of relation (polynomial attributes and interpolation intervals) are stored in a plug-in containing the model of a ship. This solution allows unification of simulation, e.g. behavior of a ship on sea.

The simulation system Dynast was used for finding the most precise interpolation. Some of these simplicities are executed by model modification (it is possible to eliminate the parts of models that influence the progress of the simulation only a little). Output of model simulation is formed by samples of value of observing variables that are later interpolated. They are used during simulation of the entire scene for calculation of model reactions to environments effects.

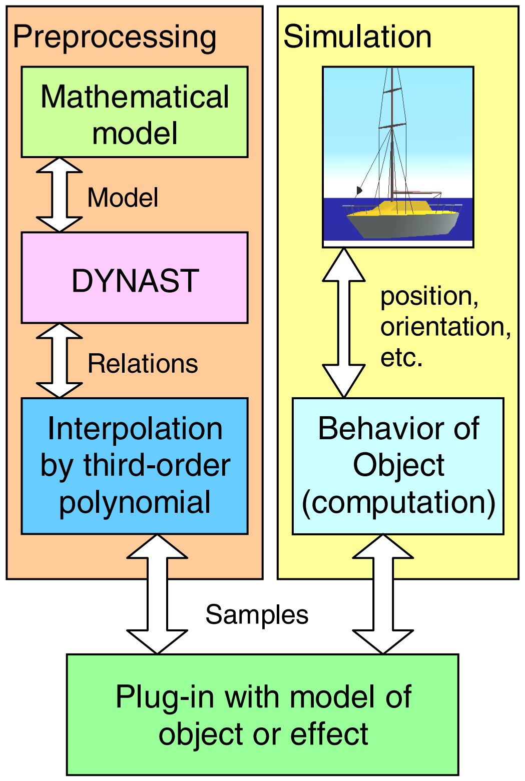

Preprocessing
Simulation
Mathematical model
Model
DYNAST
Relations
Interpolation by third-order polynomial
position, orientation, etc.
Behavior of Object (computation)
Samples
Plug-in with model of object or effect

Figure 6. Phases of simulation

4.3. Dynast

To make physical modeling easier, Herman Mann and his team have developed, many years ago, a simulation package Dynast. Now it can be easily used as a physical modeling toolbox for MATLAB, even thought the Internet. Dynast allows to set up models of engineering systems from system parts in a kit-like fashion based on a mere inspection of the real systems in the same way in which the systems are assembled from real components without forming any equations or graphs. Dynast formulates all the equations respecting the physical laws governing mutual energetic interactions between the components automatically.

5. Related work

There are many web-based applications that visualize 3D environment. [9][10][11][12]. To our knowledge, this is the first computer graphics paper that describes the main innovative characteristic of our application which is fusion of four features into one product: Web-based 3D visualization, multi-user environment, simulation process preprocessing, and possibility to extend or change the scene using plug-in system (our product is not just a yacht simulator; if we change models and effects we can get, for example, a train simulator. It is quite difficult to find a comparable web-based 3D system as various authors often focus on description of the visualized effect and the visualization itself (to make it as good as possible), but do not pay attention to other problems (management of a scene, computation etc.).

Similar system is e-Agora [9] that is specialized to multi-lingual communication and interactive entertainment – it uses the same library DILEWA, VRML browser for 3D visualization, and our department takes part in its implementation as well. E-agora, however, is determine for one use

Another project present by Humusoft [10]company uses Simulink for computation of simulation and VRML for visualization. Both parts are interconnected through their Virtual Reality Toolbox (sold worldwide by The Mathworks, Inc.). This solution, however, does not take the advantage of VRML scripts, does not support multi user scenes and requires special software.

6. Conclusion and Future Work

An advanced web-based training environment for yacht captains has been described in this paper.

The novelty of our approach is the architecture of scene plug-ins and their universal interface that allows to create models of objects or effects independently from the development of the Nautilus environment itself (teamwork design). Next asset is splitting the computation of simulation into two phases (preprocessing of samples and their use).

The developed application uses standards (VRML, Java), publicly available software tools (editors, compilers) and libraries developed at our department (some parts are used for pedagogical purposes, too). Thus, the resulting program is very cheap, portable to various platforms and easily extendable for new models and functionality.

Currently, our goal is to rapidly increase the number of plug-in modules (new ships, swinging on the waves, sun traveling in the sky, wind effects – especially to yachts) and make preprocessing phase more user-friendly.

7. Acknowledgment

This work has been supported by the Czech Technical University, Prague, Czech Republic under Research program No. 3066/2D/A in 2002 (Internal grant of CTU).

8. References

[1] H. Mayr, *Virtual Automation Environment* Marcel Dekker 2002, ISBN 0-8247-0736-2

[2] Rory Start, *The Design of Virtual Environments* Barricade Books Inc. 2002, ISBN 1-56980-207-6

[3] J. Chludil and J.Žara. *Yacht captain training system in VRML. NATO PfP/PWP Workshop - CATE 2001 Brno : Military Academy, 2001, p. 1-8.*

[4] H. Mann *DYNAST - multidisciplinary Web-based simulator*. Dresden 2001

[5] The Virtual Reality Modeling Language. International Standard ISO/IEC 14772-1:1997.
http://www.web3d.org/technicalinfo/specifications/vrml97/

[6] The Virtual Reality Modeling Language External Authoring Interface. *Committee Draft* ISO/IEC 14772-2,
(EAI is now under final vote for approval by ISO as VRML 97 Amendment 1, Part 2.)
http://www.vrml.org/WorkingGroups/vrml-eai/Specification/

[7] J. Žára and M. Máša. *DILEWA: The DIstributed Learning Environment Without Avatars*. In Proceedings of IV 2000 - Information Visualization 2000, London, Great Britain. IEEE Computer Society, ISBN 0-7695-0743-3, pages 563-567.

[8] Posey Yacht Simulator on www.poseysail.com

[9] J. Adamec, J. Cizek, M. Masa, P. Sidoni, P. Smetana, and J. Zara. *Virtual House of European Culture: e-AGORA*. In Proceedings of the 1st International Conference on Virtual Storytelling, 2001. e-agora

[10] Humusoft on http://www.humusoft.cz/vr/index.htm

[11] Omitron on http://www.onitron.com/onitron/ehome.htm

[12] Web3D Training Program on
http://www.parallelgraphics.com/products/isa/success/dreamscape/