# A Web Based Real-Time 3D Simulator for Ship Design Virtual Prototype and Motion Prediction

**Olivia Chaves**, UNIFEI, Itabira/Brazil, oliviachaves@poli.ufrj.br
**Henrique Gaspar**, NTNU, Ålesund/Norway, henrique.gaspar@ntnu.no

**Abstract**

*This paper proposes an open-source application capable to run real-time ship motion simulations in a web browser, in any device of any operational system with HTML5 compatibility. This becomes possible by implementing closed-form expressions for wave-induced motion in JavaScript code, assisted by THREE.js and WebGL libraries to handle 3D graphics. Furthermore, this approach offers support for parametric 3D models and fast collaborative virtual prototype development. The breakthrough advantage consists in adapting the simulation tool requirements to user's common platform (modern web browser), instead of forcing the user to comply with the tool requirements (operational system, installs, updates, commercial software or file format).*

## 1. Introduction – Virtual Simulation for the Web Environment

Virtual simulators have been used in maritime applications for decades, where experience-based design has progressively integrated simulation and virtual tools to assist the design process. Nowadays, industry relies in such tools to predict system response, assist operation, and assess design options. Analysis regarding stability, hydrodynamics and structure, together with 3D modelling, are examples of the most common virtual applications in ship design – Compit proceedings are full of such examples.

Currently, mainly commercial software programs are used to perform simulations. The benefits they provide are undoubted. However, most of them are restricted to a specific operating system (e.g. Windows), and codes are usually black boxes, preventing collaborative development and constraining its applicability and usability. Moreover, when it comes to 3D modeling software, each of them typically has its own proprietary file extension format, often causing compatibility issues, and imposing obstacles for sharing.

With respect to ship motion, several non-commercial programs have been developed to simulate vessel's response in waves, using diverse approaches. *Bertram (2014)* provides a compilation on this subject. Two factors could have contributed to the wide application of self-projects: the ship motion theory is accessible, and if kept to a basic level, the ruling equations do not require sophisticated solvers. Such projects would be recommended for the conceptual design, where fast and low-cost solutions are desired, and precise results are not essential. However, such projects may become outdated, due to the programming language they are written in, and possibly discontinued.

As the internet is essential for everyday life, it is very unlikely that a language specially developed for web will fade away soon, especially with big players in the market (e.g. Google) supporting common and open standards such as HTML5, www.w3.org/TR/html5/, and JavaScript. Actually, the tendency points to a fast and constant development based on current technology, without discontinuation. Recently, open-source web applications have been growing and computational capacity has been expanding towards web efficiency. Whereas hardware suppliers constantly and rapidly release new processors optimized for web performance, software suppliers often release new versions with improvements on features rather than on programming, not necessarily updating their code to meet new processor's routines.

Given these circumstances, software installed in the client machine will normally be more costly for computer capacity than any application running inside a web browser, doing the same as the software. In a recent benchmark, for instance, it was analyzed that MatLab can take up to 800x more time to a simple parse integer operation when compared to modern script languages, http://julialang.org/. More-

over, even commercial software developers are adapting to web trends, creating online versions of their products (e.g. Microsoft Office, AutoCAD and Adobe Photoshop).

In this context, we propose a web-based open-source ship motion simulator, focusing in collaborative development. The idea is to give continuity to the work, instead of rebuilding new versions of old achievements, because they are now unassessed due to the disuse of software programs, programming languages, or emulators. Besides, developing for web means developing for any device or operating system (OS), increasing the application reach and eliminating the need of client-side software.

The present simulator is based on *Jensen et al. (2004)*, who proposed closed-form expressions to estimate wave-induced motion for mono-hull vessels. These expressions require only vessel main dimensions and basic hull form coefficients, being especially relevant for conceptual design, where little information about the hull form is available. The approach allows the designer to vary those parameters, and quickly assess their influence on the wave-induced motion. Jensen's expressions were implemented in JavaScript code, and transposed to time domain. In this way, parametric real-time simulations are provided by the application, together with a 3D visualization of vessel's motion in regular waves.

## 2. Open-source tools overview

### 2.1. Open-source technology

Open source is a development methodology (or philosophy) that stands up for a transparent and collaborative platform, where monetary profit is secondary. It is based on free access to the software code, allowing users to not only see, but also modify it. In short, it can be summarized by the following facts:

- Continuous improvement; anyone can contribute or fix the code without having to wait for the next version release.
- Room for customization; issues can be better addressed due to customized solutions.
- Company independence; the software improvement or continuation may depend on the users, rather than exclusively on the owner company.
- Free support online; there are many user's communities online ready to assist each other.
- Total cost; comparing to commercial software, it is cheaper due to collaborative development and lack of investments in marketing, security, testing, etc.
- Flexibility; the code can be modified to best adapt to user's needs.

Although anyone can adapt the source code, its ownership/license may remain with the original developer (if desired), regardless anyone's contribution to the software. Famous examples of open-source software are Linux operating systems (e.g. Ubuntu, Gentoo) and the Mozilla Firefox browser. An overview of the license terms for maritime engineering is discussed by *Bertram et al. (2006)*. All tools and resources used to develop the Ship Motion Simulator are either free or open source, and a brief overview about them will be given in the following topics.

### 2.2. JavaScript

According to Mozilla Developers Network (MDN), https://developer.mozilla.org/, JavaScript (JS) is a lightweight, interpreted, programming language with first-class functions. It is best known as the scripting language for the Web. It is a prototype-based, multi-paradigm, dynamic scripting language, supporting object-oriented, imperative, and functional programming styles. Practically, JavaScript is responsible for adding interactivity to webpages. As the most used programing language nowadays, it will stay relevant as long as people use the Internet. JavaScript is compatible to diverse OS and devices. If there is a web browser the code can be executed. It has the analytical capacity of any other programming language, plus it can run real-time simulations. There is no need to compile or wait for the results, and a text editor (e.g. Notepad) suffices to write code. Compared to the old times, where each

mobile OS had its own programming language (e.g. Objective-C for iOS; Java for Android; C# for Microsoft), JavaScript presents a great evolution in standardizing. Additionally, it runs in the client machine, which performance may be either enhanced or constrained by the client's hardware; it does not require constant connection to the net.

### 2.3. WebGL

WebGL (Web Graphics Library) is a cross-browser JavaScript library/API, which is used by THREE.js as a rendering complement. It allows interactive advanced graphics to be rendered within a web browser and optimizes the hardware use. WebGL demands less from hardware because it "shares" the rendering work. Some features that needed to be handled by the hardware can now be handled by the language, resulting in a lower hardware requests. In addition, WebGL does not use plug-ins and there is no need for installs or updates, which is another significant advantage compared to the decaying Adobe Flash. WebGL has been used in applications from gaming to science. For example, NASA released their interactive educative tool 'Experience Curiosity', Fig. 1, which simulates the Curiosity Rover's adventure in Mars. Neuroscientists have developed a real-time visual exploration of the connectome data including FreeSurfer structural reconstruction, tractography, and network data all within the browser, Fig. 2, *Ginsburg et al. (2011)*, all WebGL based.

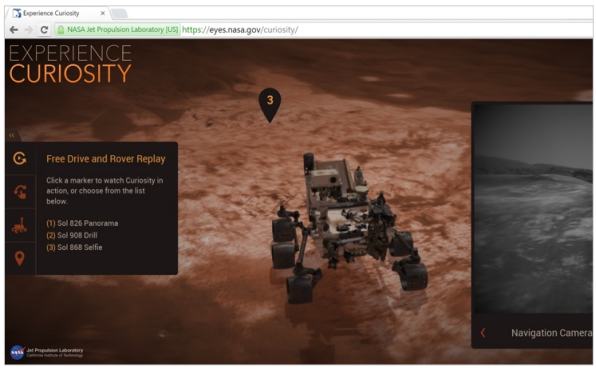


Fig. 1: Experience Curiosity

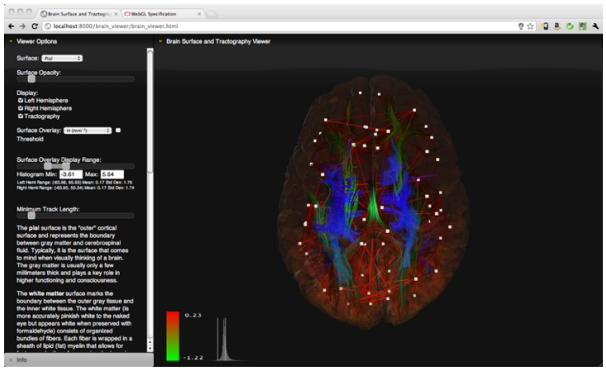


Fig. 2: Connectome visualization

### 2.4. THREE.js

THREE.js is another cross-browser JavaScript library/API, threejs.org. It is used to create and display animated 3D computer graphics on a browser. THREE.js allows GPU-accelerated 3D animations to be created as part of a website using JavaScript language. Significant advantages of THREE.js are the possibility to add and remove objects from a scene at run-time, communication with OpenGL (through WebGL), several compatibility solutions, collaborative development, and free support online. Moreover, the THREE.js library has innumerous features such as (currently) three camera options, six control types, thirteen material variations, four texture types, nineteen loader options, debugging, etc. This set of features allows building 3D animations just like in any dedicated software.

### 2.5. STL (STereoLithography)

STL is a three-dimensional representation of a surface geometry. Although lacking some modern features and criticized for its heavy file size, it is an open-standard file format accepted by almost every CAD program in the market today. Especially used for 3D rapid prototyping and 3D printing, this format carries information exclusively about the object's geometry, describing its surfaces through triangles, without any other object property such as color or texture. Thus, despite its simplicity, this format can be used to transport the hull geometry into the Simulator, not requiring any specific software to generate it.

### 2.6. Collada (COLLAborative Design Activity)

Collada (COLLAborative Design Activity) is a file format used to transport 3D assets. It is capable to carry more information about the 3D environment (including geometry, materials, shaders, effects, lights, or even physics and animations). Compared to STL, Collada is preferred as it can confer material properties to the model and offers support for exchanging advanced 3D graphics assets between applications. However, more sophisticated software is required to generate such file, and in this case, there are much fewer free programs supporting Collada than STL. Yet, Blender is an example of free, open-source software that supports Collada, www.blender.org/, Fig. 3.

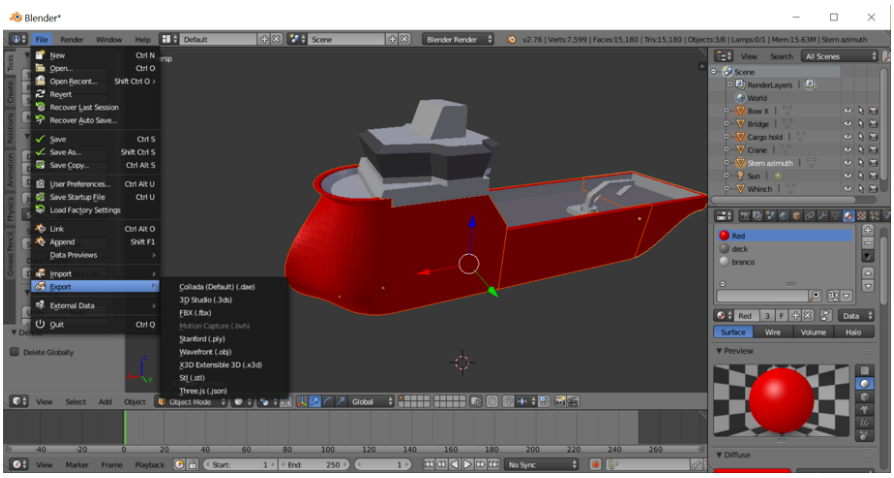

Fig. 3: Blender interface and exporting formats, such as Collada and .STL files

## 3. Virtual Prototype and the Web

Virtual prototyping, *Chaves et al. (2015)*, is extensively used by industry, but has not crossed the internet's barrier regarding the ship design sector. Currently available JavaScript APIs (e.g. THREE.js and WebGL) generate favorable conditions to bring prototyping to the web or even to a web-based virtual reality environment. Although there are not many examples yet, the internet is evolving to embrace productivity tools (e.g. Office and Photoshop), and engineering applications (e.g. Hull Lines Generator, www.shiplab.hials.org/app/shiplines/, 3D Configurator, www.shiplab.hials.org/app/3dconfigurator/, and Ship Motion Application, www.shiplab.hials.org/app/shipmotion/).

Software engineering is completely focused on the end user, which fundamentally should provide solutions that people wants to see. *Pressman (2005)* establishes a comparison between software and web applications, demystifying their difference and distance: Web pages are user interfaces, HTML programming is programming, and browser-deployed applications are software systems that can benefit from basic software engineering principles.

Web is today the most optimized and used GUI (Graphical User Interface) worldwide. Based on Event-driven Programming, a Web GUI provides interactivity by "watching" for events. Events include user's actions such as mouse click, scrolling, hover over, key press, etc. This type of GUI is ready to accept user's input at any time, rather than accepting inputs only when asked, Fig. 4.

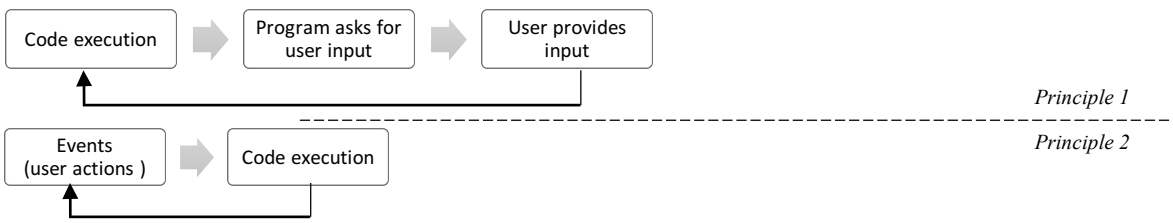


Fig. 4: GUI action principles

Both the amount of interactivity and the focus on usability make the web appealing and contribute to attracting more and more adepts. User-friendly characteristics are even more enhanced on mobiles, increasing the potential to use it as a tool. According to *Hoos et al. (2015)*, mobile apps provide a huge potential for increased flexibility and efficiency due to their 'anywhere and anytime' characteristics.

User-friendly comes with simplicity. Therefore, making extensive trials or complex procedures to become user-friendly might be a challenge when it comes to adapting engineering tools to the web frame. However, precise analyses and optimized results are not necessary in the early design phase. Design concepts are full of considerations and assumptions, creating simplified models to be further developed. So, we believe that the potential to apply web-based engineering tools are in solving these early simple problems, in order to save time, money and avoid mistakes.

For ship design, web tools allow an intuitive, multi-platform application, which can be used in mobiles, computers, tablets, or any device supporting HTML5. The presented simulator allows assessing design options by varying vessel's main dimensions and verifying resulting ship motion. It also includes a vessel's 3D visualization (representation), making room for virtual prototyping, *Chaves et al. (2015)*.

## 4. Methodology

### 4.1 Code structure

Considering today's trends, recognizing JavaScript potential and WebGL capabilities to handle 3D advanced graphics, THREE.js was chosen to develop a web-based simulator, Fig. 5. The 3D environment is composed by four basic elements: camera, light, scene, and render. In order to allow the user to navigate the camera, orbit controls were implemented. Then two core objects, the vessel and the ocean, were included in the scene and their motion encoded. Finally, a console (GUI) was added so that the user can input both wave and hull form parameters.

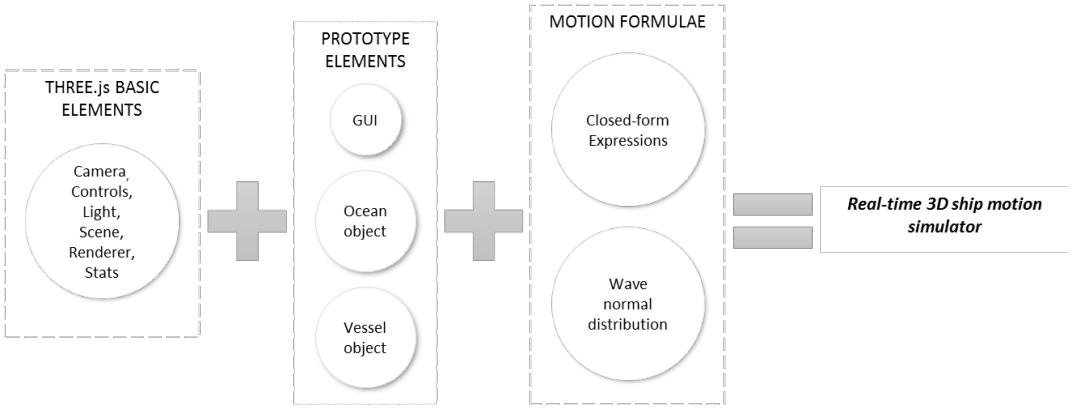


Fig. 5: Ship Motion Simulator main components

The vessel 3D model was designed outside the web application, and included (imported) into the scene exclusively for visualization purposes. As modeling a hull is not simple, and becomes even more difficult without using a dedicated CAD software, the following procedures are suggested to create a Collada file.

(1) Using a hull modeling dedicated CAD software to generate the hull 3D geometry;
(2) Exporting the 3D geometry in a standard format such as .stl, .obj, .igs;
(3) Importing to a software that supports 3D advanced assets, and is capable of exporting Collada (e.g. Blender, Rhinoceros);
(4) Adding material, texture, shaders, effects, ambient light, and/or any other asset to the hull geometry, in order to confer reality to the hull representation;
(5) Exporting the final scene as Collada file.

Collada was the file format chosen to transport advanced 3D assets from the modelling software to the web application. Simpler file formats such as STL and OBJ could also be used, and the procedure above would stop at number (2), but the visualization realism would be considerably decreased.

The vessel motion amplitude is calculated by closed-form expressions given in *Jensen et al. (2004)*, based on hull form parameters (L, B, T, $C_B$, $C_P$, $C_{WP}$ etc.). This method provides a rational and efficient procedure to predict the wave-induced motions with sufficient engineering accuracy in conceptual design. *Bertram (2014)* discusses that this method, together with many others for seakeeping analysis, works well if used within its designed scope. Apparently, both *Jensen et al. (2004)* and *Bertram (2014)* agree that this method provides acceptable results, but remind us that reality is more complex. For instance, roll motions are dominated by nonlinear viscous damping, which are much more difficult to calculate.

In *Jensen's et al. (2004)* method, the ship is represented by a homogeneously loaded box-shaped barge with the beam modified so that the total mass of the ship equals the buoyancy. The method considers the vessel in regular waves and deep waters. Pitch and heave are always considered with 90° phase angle between them, disregarding coupling between them. The closed-form expressions' results were verified against model tests and strip theory calculations.

For the real-time Simulator, the motion amplitude is transposed to the time domain:

$$\eta_j(t) = |\eta_j| \times sin(\omega \times t + \theta_j) \qquad j = 3, 4, 5. \qquad (1)$$

Where $\eta_j$ is the closed-form expressions output for a given frequency;
$\omega$ is the wave period set by the user;
$t$ is the time;
$\theta_j$ is the phase angle;
$j$ = 3, 4 and 5 indexes refer to heave, roll and pitch, respectively.

The ocean 3D object is constructed entirely by THREE.js features. It is basically a plan geometry, Fig. 6, in which a texture is applied, Fig. 7.

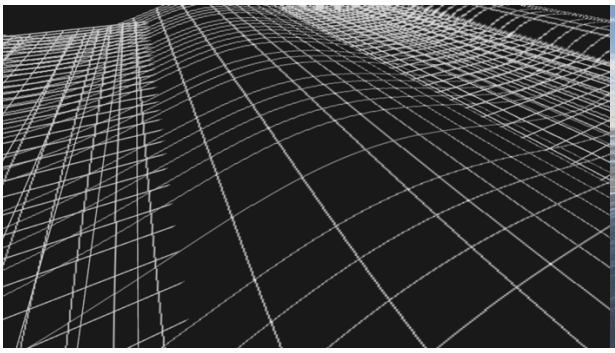

Fig. 6: Ocean dynamic geometry

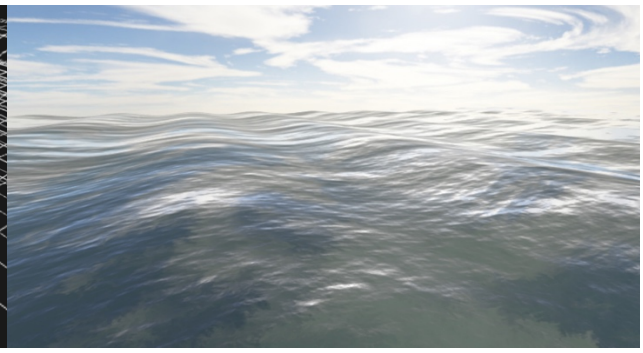

Fig. 7: Ocean geometry + texture + sky

The geometry is divided in several sub-segments, and set to dynamic. The waves are generated by a normal distribution function that updates the height of each vertex, forming waves in the plan. The normal distribution complies with the wave elevation equation for regular waves.

$$\zeta(x,t) = \zeta_a \times sin(\omega \times t - k \times \cancel{x}^{\,0}) \qquad (2)$$

Where $\zeta_a$ is the wave amplitude set by the user;
$x$ is assumed zero.

The console, Fig. 10, allows the user to define wave characteristics, scale vessel's main dimensions, and input hull form coefficients. At the moment these are changed, the application updates all the calculation and visualization instantaneously. Although the backside calculation is based on wave fre-

quency, this parameter is inverted in the GUI. For practical reasons and usability concerns, wave period is the parameter that the user specifies.

### 4.2 Simulator workflow

The Simulator workflow can be explained as any other input-process-output system, Fig. 8. It runs in two synchronized parallel routines that use the same input parameters. Each step is explained as follows.

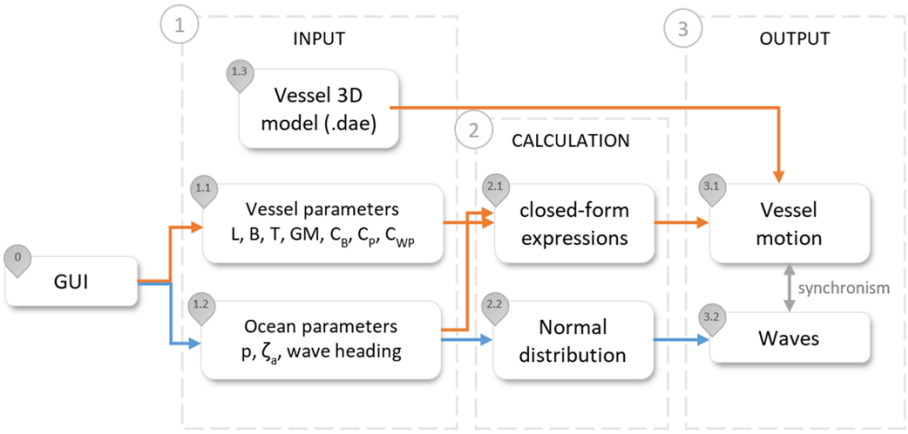


Fig. 8: Ship Motion Simulator workflow

The GUI centralizes all inputs the user can manage. It provides interactivity and instantaneously response. As a matter of organization, the GUI inputs are divided into vessel and ocean parameters:

- Input 1.1 - Parameters related to the hull form, exclusively used for the vessel motion calculation. They are length, beam, draught, metacentric height, block coefficient, prismatic length ratio, and waterplane area coefficient.
- Input 1.2 - Parameters related to the sea state, used for calculating vessel motions and wave generation. They are wave period, amplitude and heading.
- Calculation 2.1 - The closed-form expressions calculate the wave-induced motion for the parameters given in the GUI and return the motion amplitudes (heave, pitch and roll) separately. This function is called every rendered time step as the application runs; so every change in the GUI will automatically update the amplitudes.
- Calculation 2.2 - The normal distribution generates regular waves as a sinusoid. The same para-meters used in Calculation 2.1 are used here. There is no collision or physics engines calculating the interaction between the ocean and the vessel; they are rather ruled by two distinct synchronized set of equations. The ocean is simply a representation of the waves coming in the vessel, which are considered inside the closed-form expressions.
- Output 3.1 - The vessel motion is given by Eq.(1) and updated every render loop. This formula returns motion amplitudes for each time step, and assigns it to the vessel's 3D model (Input 1.3), making the model to move.
- Output 3.2 - The waves' formation is given by Eq.(2), which is updated in the same time step as Eq.(1). This formula applied to the ocean geometry generates the waves.

## 5. Web Based Real-Time 3D Simulator - Virtual Prototype and Motion Prediction

The application interface is very clean and focuses on the 3D environment. All features are meant to be straightforward and user-friendly. There is no need to "learn the software", as it will be intuitively unveiled within few minutes of use.

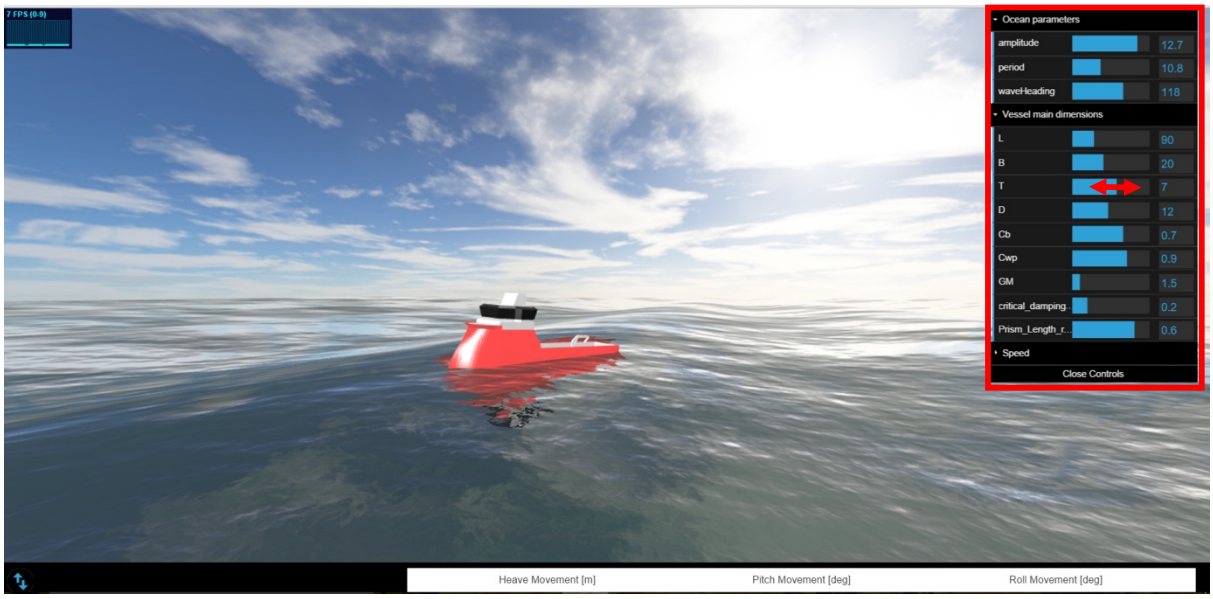


Fig. 9: Ship Motion Simulator: GUI

The GUI stays at the right-hand side, Fig. 9, and allows the user to vary each parameter value either by dragging and dropping the blue bars or typing a number. For better precision, it is advised to type the desired value. There is a retractable tab at the bottom side of the application, which contains the closed-form expressions output, the movements' amplitude, and the wave length corresponding to the given wave period, Fig. 10. The wave length λ is calculated under the assumption of deep water as:

$$\lambda = \frac{2\pi g}{\omega^2} \tag{3}$$

$g$ is the gravity acceleration, $\omega$ the wave circular frequency.

Also, down the page, there are graphs of the three uncoupled motions of the vessel, Fig. 11. The same formulas responsible for updating vessel's position and rotation also feed the graphs. In this way, they display the vessel motion in real-time, along with the 3D model.

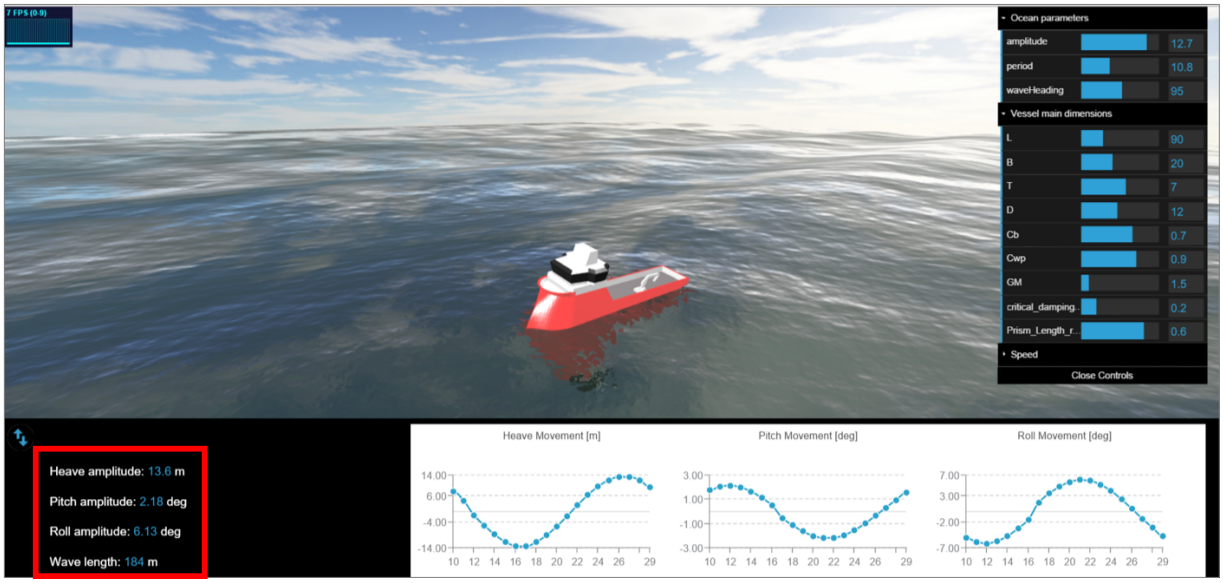


Fig. 10: Ship Motion Simulator: Amplitudes

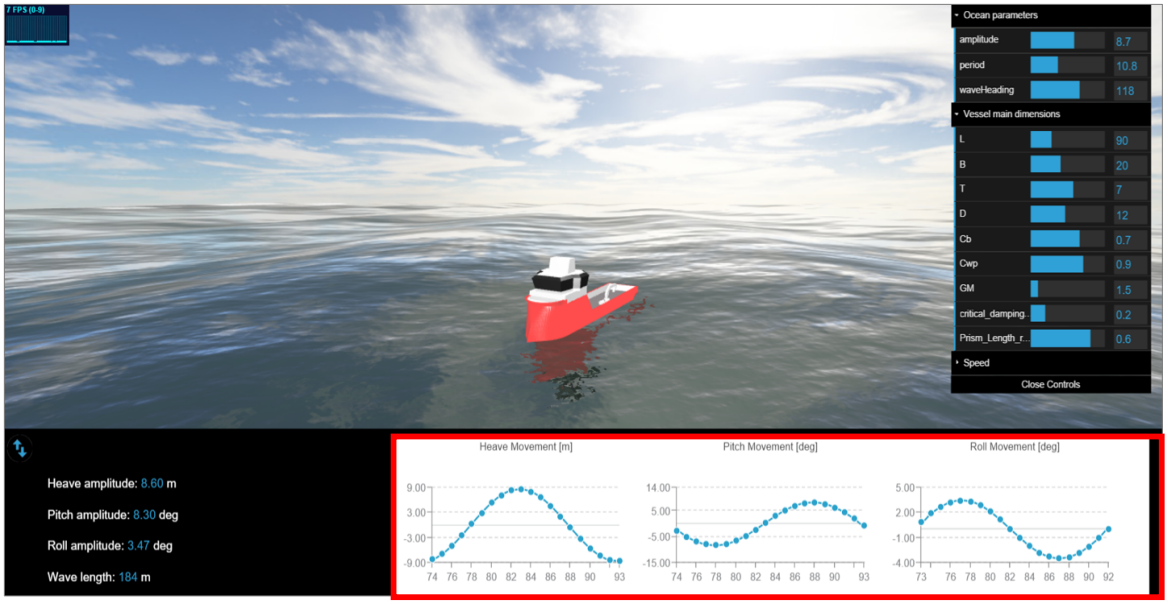


Fig. 11: Ship Motion Simulator: Real-time graphs

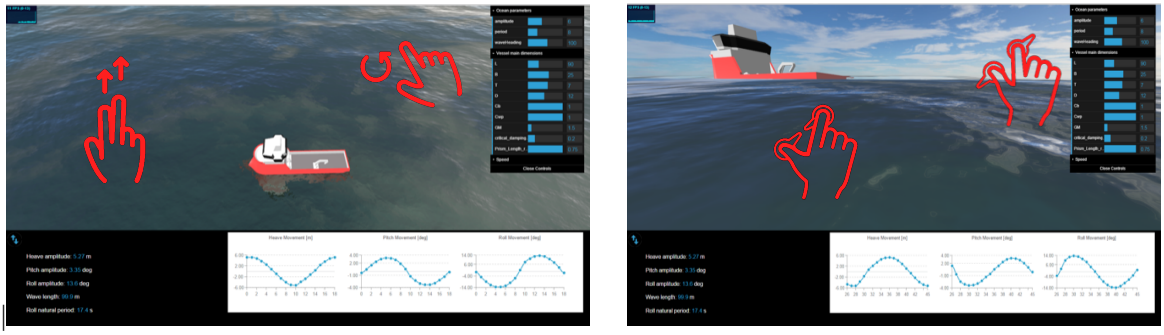

Fig.12: Ship Motion Simulator: Controls

It is also possible to navigate through the 3D environment. On mouse-controlled devices, the right button drags and drop the scene, the left button rotates it, and the scrolling button zooms in and out. On touchscreen devices, the controls are even more intuitive, since it follows the same scheme as in any other application: pinch to zoom in and out, twist fingers to rotate, and slide your finger to drag, Fig. 12.

## 6. Concluding remarks and extension of the work

There is no doubt that the current level of web-based technology allows the creation of simple and customized web-based tools that facilitate the engineer's and the designer's work. Mainly in early design stages, those tools provide enough support to go on with the project, saving time and money. Using open and free resources, it is possible to create applications to improve working practices.

If these applications are made open it favors fast and collaborative development. Open-source codes allow any user to add contributions, thus accelerating improvements. However, cooperation has its limits. While we can find many cooperative examples among academic developers, industry seems to make much less use of such boost. Most companies (or even scholars) are not willing to share their proprietary knowledge and assets. In doing so, they renounce all the benefits a cooperative undertaking could bring, balancing between profitability and obsolescence.

Our experience is that JavaScript has proved its versatility and potential when applied to maritime engineering toolbox. Diverse free libraries are available to implement brand new capabilities, or improve the existing ones without the need of hard coding. With the development of advanced solvers, more complex formulations can be implemented, and the computational capacity hardly will be a constraint.

Visualizing the system's response in a 3D web environment is certainly an enhancement when compared to the traditional 2D graphs, which often are not obvious to interpret even for experts. For the client side, companies can explain or document their results much more intuitively. 3D is appealing and understandable by anyone.

Considering JavaScript's expected durability, no work is likely to be lost or outdated, favoring developments continuity. New features are in process to be implemented in the presented application, such as first person camera view and motion. Further work could point towards a platform multi-user, where the same virtual environment would support two simulations at the same time, commanded by two users in different machines, possibly interacting with each other.

**Acknowledgments**

The Simulator's source code benefited from Juliano Monte-Mor's (UNIFEI, Brazil) and Elias Hasle's (NTNU, Norway) expertise in programming. They significantly contributed to improve the code and fix bugs, providing fundamental assistance. We also benefited from Lars Ivar's (NTNU, Norway) work, experience in developing WebGL engineering tools, and assistance, helping on the project's kick off. This research is connected to the Ship Design and Operations Lab at NTNU in Ålesund, which is partly supported by the EMIS Project, in cooperation with Ulstein International AS (Norway) and the Research Council of Norway.

**References**

BERTRAM, V.; VEELO, B.; SÖDING, H.; GRAF, K. (2006), *Development of a freely available strip method for seakeeping*, 6th Conf. Computer and IT Applications to the Maritime Industries (COMPIT), Oestgeest, pp.356-368

BERTRAM, V. (2014), *Computational methods for seakeeping and added resistance in waves,* 13th Conf. Computer and IT Applications to the Maritime Industries (COMPIT), Redworth, pp.8-16

CHAVES, O.; NICKELSEN, M.; GASPAR, H.M. (2015), *Enhancing virtual prototype in ship design using modular techniques,* 29th Eur. Conf. Modelling and Simulation (ECMS), Varna

GINSBURG, D.; GERHARD, S.; CALLE, J.E.; PIENAAR, R. (2011), *Realtime visualization of the connectome in the browser using Webgl*, 4th INCF Congress of Neuroinformatics, Boston

HOOS, E.; GRÖGER, C.; MITSCHANG, B. (2015), *Mobile apps in engineering: a process-driven analysis of business potentials and technical challenges,* CIRP 33, pp.17-22

JENSEN, J.J.; MANSOUR, A.E.; OLSEN, A.S. (2004), *Estimation of ship motions using closed-form expressions*, Ocean Engineering 31/1, pp.61-85

PRESSMAN, R. (2005), *Software engineering: A Practitioner's Approach,* McGraw-Hill