Using the Graphics Hardware to Compute Wind Forces

Sérgio Alvares R. de S. Maffra, *Tecgraf/PUC-Rio*

Luiz Cristovão Gomes Coelho, *Tecgraf/PUC-Rio*

Guilherme Tavares Malizia Alves, *Tecgraf/PUC-Rio*

Mauro Costa de Oliveira, *Petrobras/Cenpes*

Carlos Gomes Jordani, *Petrobras/Cenpes*

ABSTRACT

We propose an alternative procedure to compute the forces and moments resulting from the action of winds on a vessel, by using a computer generated image to determine the projected area of its exposed surfaces. The results obtained by the new method are then compared with real wind tunnel tests.

Keywords: *wind force, center of pressure, rasterization,*

1. INTRODUCTION

According to IMO (2001), the forces and moments resulting from the action of winds on a vessel can be computed as a function of the projected area of the objects, of its shape coefficients, of its height coefficients, of the velocity of the wind and of the density of the air:

$$F = \frac{1}{2} C_S C_H \rho V^2 A \qquad (1)$$

F	Wind force
C_S	Shape coefficient
C_H	Height coefficient
ρ	Density of the air
V	Velocity of the wind
A	Projected area of exposed surfaces

Although much simpler than a computational fluid dynamics simulation, the methodology suggested by IMO can be challenging when one deals with objects that present a complex geometry. The challenge, in these cases, is concerned with the determination of the projected area of all exposed surfaces of the vessel. If one decides to work directly with the geometry of the object, Boolean operations must be used on polygons (intersection, subtraction, addition, etc) which are hard to implement and are also error prone due to imprecision in the representation of floating point numbers.

As an alternative, IMO allows the use of wind tunnel tests on a representative model of the unit instead of the formula above. Wind tunnel tests however are not the most practical approach, especially when the vessel being tested is in an initial design phase. Changes in the design will probably occur and invalidate the tests. Therefore, a less expensive computational procedure can be of great value.

Since IMO requires that wind forces should be considered from any direction relative to the unit and with different values of wind velocity, the analysis of wind heeling forces using computational fluid dynamic (CFD) simulations can also be a costly procedure, given the large number of different simulations and the large number of meshes that must be constructed for these simulations.

We propose an alternative procedure to the direct use of geometric primitives when evaluating Equation 1. This alternative, as implemented in all modern graphics hardware, is to convert every geometric primitive into a set of *pixels*, what turns the Boolean operations trivial. Once the set of visible pixels is determined, which correspond to determining the visible surface of the object, one must only compute the summation of their area to obtain the projected area needed to compute the wind force. As most of the operations necessary in this algorithm are already implemented in the hardware of all commercial graphics card, an efficient implementation of the algorithm is not hard to achieve.

When computed according to our proposal, the projected area is actually, an approximation of the real area. This is a consequence of the transformation of the geometric primitives into a set of pixels. We also present in the paper how an upper bound of this error can be computed.

2. COMPUTING THE WIND FORCE

The determination of visible surface (Foley et al., 1997) is a well-known problem in the computer graphics field. Several algorithms have been proposed to this date, but they can be divided in two main categories: image space and object space algorithms (Rij, 1994).

Object space algorithms solve the visibility problem by working directly with the geometric primitives, trying to sort them from the farthest to the closest one with respect to the camera (viewer). When the primitives cannot be sorted, Boolean operations must be used to settle the problem.

Digital images are usually represented as a rectangular matrix of numerical values used to represent different colors. Image space algorithms explore this representation directly, avoiding most of the complex geometric operations that object space algorithms must deal with. Image space algorithms are also naturally parallel, what makes them easier to implement efficiently in hardware. In this work, we show how the Z-Buffer algorithm (Foley et al., 1997) can be used to define the wind exposed areas of a complex model.

2.1 The Model

The model of a floating system may be constructed as a set of individual compartments that define the complete stability model. With this approach, we may classify these compartments as *external* (Hull body, Deck elements, Hellideck) or *internal* (Void spaces, Elevators, Access trunks, Chain lockers, etc), according to its *wind exposure*. Figure 1 shows the external compartments of a semi-submersible oil platform. Each individual compartment is composed of a set of planar faces, each face is defined by a set 3D positions, which are related to a global Cartesian coordinate system.

2.2 Determination of Visible Surface Using Vector Algebra

In a traditional approach, the visible portions can be computed using classical vector algebra in the 3D coordinate space. If we assume that the compartment faces do not overlap, a BSP Tree (Foley et al., 1997) technique can be used to sort the faces and to define the visible ones, although an algorithm to construct such representations can become slow when the number of faces increases significantly. Those algorithms should be coupled to a Boolean operator tool (Coelho et al., 2000) to obtain a consistent set of polygonal areas, necessary to represent the remaining visible faces.

Figure 1 shows the external model of a semi-submersible oil platform. If we imagine the wind acting in the same view direction (the direction of projection, or DOP) adopted to plot the drawing, the computation of the visible

areas in the detail shown by Figure 2 is an example of the use of the Boolean operations necessary to determine the visible areas near the helideck where there are intersections between several possible visible areas, that must be trimmed.

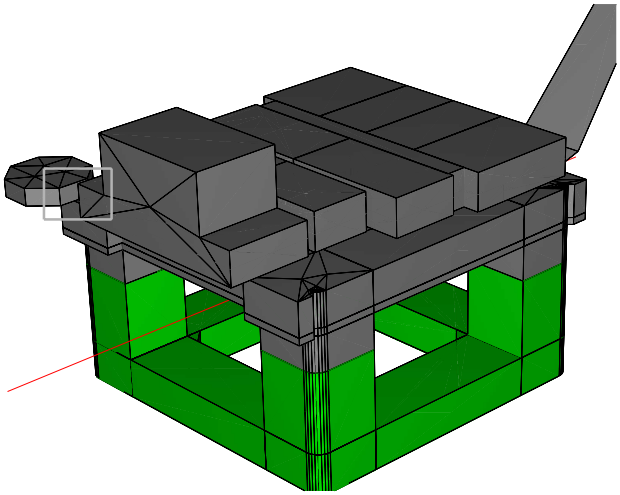

Figure 1 – Semi-submersible oil platform.

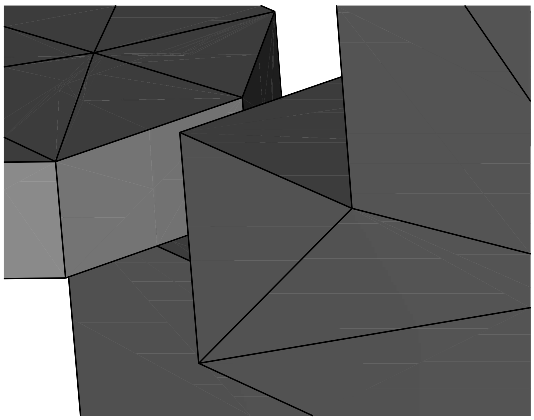

Figure 2 – Detail near the helideck.

2.3 Determination of Visible Surface Using the Graphics Card

To produce the visible surfaces of Figure 1, a video card that implements the OpenGL library primitives was employed. The same procedure used in the generation of the image can be used to determine the projected area of a vessel that is exposed to the wind.

The graphics library used in this work (OpenGL) contains an implementation of the Z-Buffer algorithm. As mentioned before, this algorithm avoids the sorting and the Boolean operations by transforming each of the geometric primitives into a set of pixels that are tested against each other for visibility. The visible pixels are then stored in the final image. The next sections explain how to configure the Z-Buffer algorithm in order to generate an image appropriate for the computation of wind forces and how to compute these forces using such an image.

2.4 Configuring the Z-Buffer Algorithm

Generating images using OpenGL can be a complex task due to the large quantity of options and different rendering techniques that can be applied. Fortunately, in order to generate images suitable for the computation of wind forces, only a few steps are required.

The first step consists in configuring the camera. The camera must be positioned in a way that allows the entire object to be seen. The viewing direction of the camera (its direction of projection or DOP) must be the same as the direction of incidence of the wind. As Equation 1 requires the areas projected to the vertical plane, the projection plane of the camera must be a vertical plane (its up vector is equal to (0,0,1)). Also, an orthographic projection must be used, in order to allow a correct measurement of areas. A perspective projection cannot be used as it would deform the model to create the perspective effect on the generated image.

The second step consists in choosing appropriate colors for the components of the geometric models that represent the different compartments of the floating system being studied. After the generation of the final image has been completed, there is no relationship between the pixels in the image and the compartment it represents.

Given a pixel of the image, it is possible to determine all the parameters needed for Equation 1 with the exception of its associated

shape coefficient, which can only be determined by knowing the compartment associated with a pixel of the image. The technique to solve this problem is to use as the color of each compartment an integer identifier (an integer number that represents a class of objects or a single object). As we are interested in the exposed area of each compartment, each compartment is assigned a unique color. In this way, it is possible to determine the set of visible pixels associated with each individual compartment, by examining the colors of the pixels.

As the lighting computations performed by the graphics library can alter the colors it receives as input, lighting must be disabled when generating images for the computation of wind forces.

After configuring the graphics library with the procedure described in this section, one must only render the final image to obtain the data necessary for the computation of the wind force.

Figure 3 shows an example of an image of an oil platform generated using the techniques described in this section.

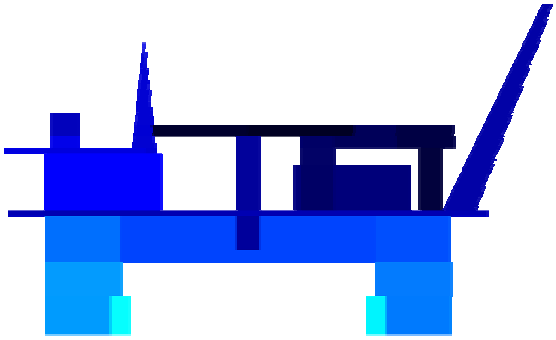

Figure 3- Image for wind force computation.

2.5 Computing the Wind Force

The computation of the wind force using the image generated according to the previous section is similar to a numerical integration procedure, since each pixel of the image can be seen as an integration element. Equations 2 and 3 show how to compute the value of the wind force and its center of pressure, respectively.

$$F = \sum_{i=1}^{w}\sum_{j=1}^{h}\frac{1}{2}C_s(P_{ij})C_h(P_{ij})\rho V^2 A_p \qquad (1)$$

F	Wind force
P_{ij}	Pixel located at position (i,j) of the image
ρ	Density of the air
V	Velocity of the air
A_p	Area of a pixel
$C_s(p)$	Shape coefficient of pixel p
$C_h(p)$	Height coefficient of pixel p
w	Width of the image
h	Height of the image

$$C_p = \frac{1}{F}\sum_{i=1}^{w}\sum_{j=1}^{h}\left[\frac{1}{2}C_s(P_{ij})C_h(P_{ij})\rho V^2 A_p\right]h(P_{ij}) \qquad (3)$$

C_p	Center of pressure
$h(p)$	Height of pixel p

Equation 2 depends on two functions that have a pixel as a parameter: the shape coefficient function (Cs) and the height coefficient function (Ch). Both functions can be implemented as tables to indicate their respective results.

The shape coefficient table has colors as indices. The same colors used to represent each individual compartment. In this way, with the color of a pixel, one can determine the corresponding compartment of the floating system and, consequently, its shape coefficient.

The table that implements the height coefficient function is indexed by the distance of a pixel to the surface of the sea. One of the parameters of the camera configuration is the region of space that is visible, which is completely covered by the pixels of the image. Therefore, the position of a pixel in space can be easily determined from its position inside the image. Then, by looking up the corresponding height range in the height

coefficient table, the coefficient desired is determined.

Equation 3 depends on another function that has a pixel as parameter. This function computes the distance of the pixel to the surface of the sea, using the same procedure to determine the index of the height coefficient table.

Table 1: Height Coefficients Table

From (m)	Until(m)	Height Coeff.
0,00	15,30	1,00
15,30	30,50	1,10
30,50	46,00	1,20
46,00	61,00	1,30
61,00	76,00	1,37
76,00	91,50	1,43
91,50	106,50	1,48
106,50	122,00	1,52
122,00	137,00	1,56
137,00	152,50	1,60
152,50	167,50	1,63
167,50	183,00	1,67
183,00	198,00	1,70
198,00	213,50	1,72
213,50	228,50	1,75
228,50	244,00	1,77
244,00	256,00	1,79
256,00	---	1,80

Table 2: Shape Coefficients

Shape	C_s
Spherical	0.4
Cylindrical	0.5
Large flat surfaces (hull, deckhouse, etc)	1.0
Drilling derrick	1.25
Wires	1.2
Exposed beams and girders under deck	1.3
Small Parts	1.4
Isolated Shapes (crane, beam, etc)	1.5
Clustered deckhouses and similar	1.1

The previous tables present some values of shape coefficients and an example of a height coefficient table.

2.6 Rasterization Errors

The process of transforming a geometric primitive (usually a polygon) into a set of pixels is called rasterization. Whenever a geometric primitive is rasterized there are errors due to the discrete nature of the digital images. Continuous lines, for example, cannot be represented correctly on an image or on a computer monitor. The discretization errors, however, occur only on the border of the primitives. Figure 4 shows how the rasterization of some geometric primitives look like on an image. Notice how the rasterized primitives try to approximate the continuous ones (marked as red lines in the figure).

Since only the pixels on borders contain errors, it is possible to compute an upper bound for the error on the computation of the projected area. This upper bound is computed by dividing the total area of pixels located on borders by the total area occupied by pixels. The smaller the value on this upper bound, the better is the estimative of the wind force.

One way to reduce the rasterization error is to increase the resolution of the image generated. Larger images will approximate the geometric primitives better at the cost of more memory and more computational time for the evaluation of the wind force and of the center of pressure.

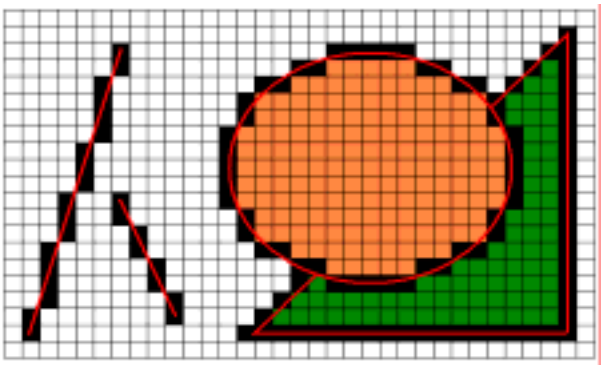

Figure 4- Rasterization errors.

3. RESULTS

The algorithm described in the previous sections has been implemented in the Sstab program (Coelho et al., 2003) and is currently being used to calculate the wind and stream forces of Petrobras-BR models, both in the design stage and for existing units. The algorithm was named WFE, whose initials stand for Wind Force Estimator. As an example of the use of WFE, we show some results for the semi-submersible platform of Figures 1 and 2. The wind computations were compared with the wind tunnel test (WTT) results.

The wind tunnel test report presents results for the operational draft and for transit draft. We selected the highest heeling levers that were obtained for an angle of wind incidence of 320 degrees with respect to the longitudinal axis of the hull. The tests were obtained for three different wind velocities: 25.7 m/s, 37.0 m/s, 51.4 m/s. We show the comparative results in the Table 3.

Table 3 – Heeling levers for operational draft

	25.7 m/s	37.0 m/s	51.4 m/s	Degree
WTT	0.188	0.389	0.750	0
WFE	0.127	0.252	0.519	0
WTT	0.192	0.397	0.767	5
WFE	0.127	0.278	0.563	5
WTT	0.183	0.379	0.730	10
WFE	0.140	0.268	0.560	10
WTT	0.187	0.388	0.750	15
WFE	0.156	0.335	0.661	15
WTT	0.179	0.372	0.718	20
WFE	0.173	0.368	0.708	20
WTT	0.135	0.281	0.541	25
WFE	0.184	0.348	0.625	25

Figures 5 and 6 show the models used in the tunnel tests and in the WFE algorithm, respectively. Both models are heeled by 5 degrees, but Sstab performs the buoyancy and trim equilibrium computations. The wind direction is assumed to be from the left to right side of the figures. The gray areas shown by Figure 6 are very similar to the visible portion of the picture shown by Figure 5. In these figures we can also observe the differences in the modeling of wind elements at the topside of the platform.

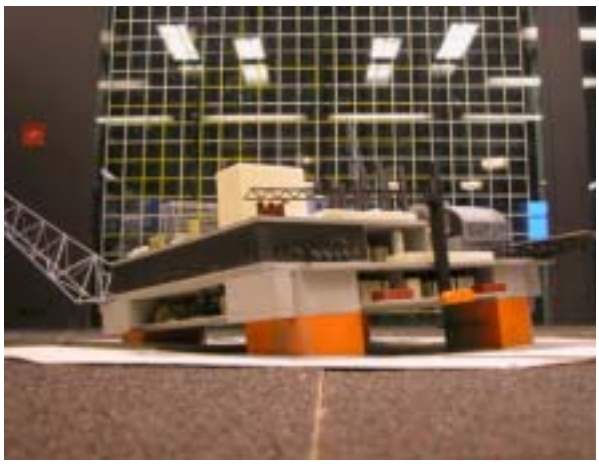

Figure 5 – Tunnel test model at 5 degrees.

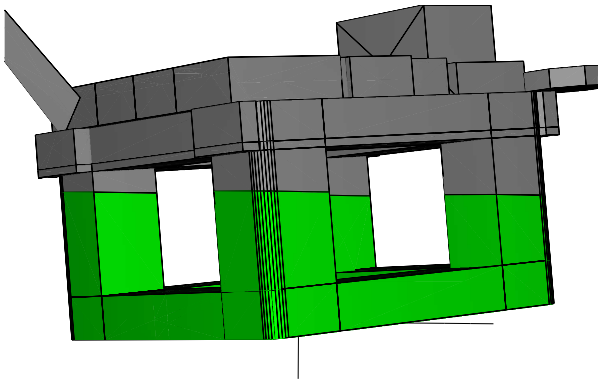

Figure 6 – Sstab model at 5 degrees

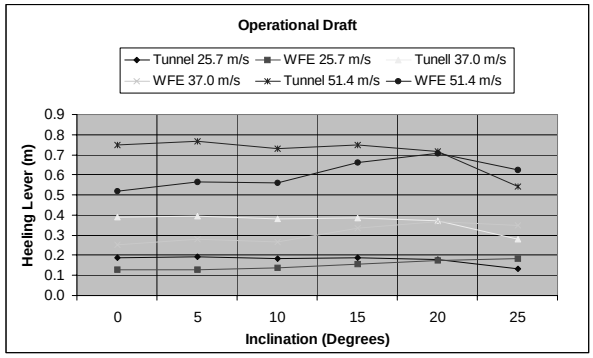

Figure 7 – Operational Draft Heeling Lever Curves.

Another pair of images, now showing the 10 degrees heeling angle are presented by Figures 8 and 9. In these figures the top compartment over the column above the flare is beginning to submerge.

Table 4 – Heeling levers for transit draft

	25.7 m/s	37.0 m/s	51.4 m/s	Degree
WTT	0.432	0.895	1.727	0
WFE	0.232	0.509	0.939	0
WTT	0.410	0.850	1.640	5
WFE	0.241	0.523	0.975	5
WTT	0.363	0.753	1.452	10
WFE	0.275	0.540	1.087	10
WTT	0.342	0.709	1.369	15
WFE	0.263	0.542	1.087	15
WTT	0.328	0.680	1.312	20
WFE	0.285	0.543	1.037	20
WTT	0.294	0.610	1.177	25
WFE	0.311	0.638	1.256	25

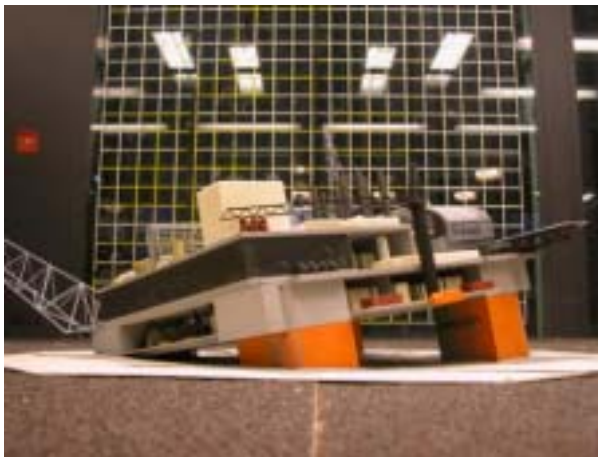

Figure 8 – Tunnel test model at 10 degrees.

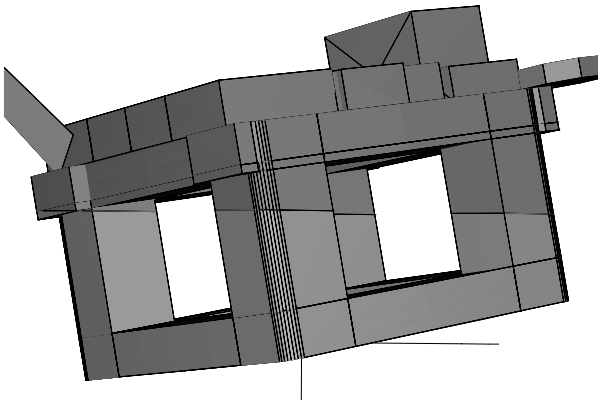

Figure 9 – Sstab model at 10 degrees

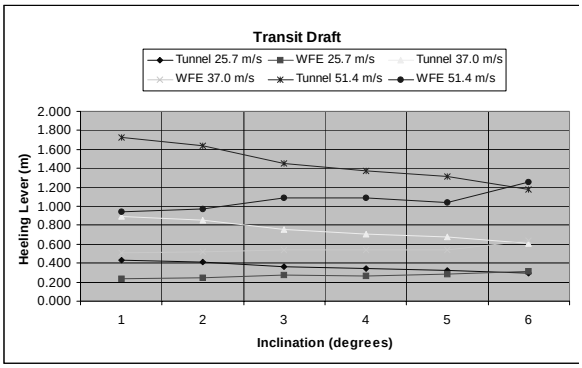

Figure 10 – Transit Draft Heeling Lever Curves.

Figures 7 and 10 show the heeling lever curves for the transit and operational draft configurations used in our tests.

The superstructure elements of the models used in the wind tunnel tests and in the WFE algorithm were considerably different. Another source of differences between the tunnel test results and WFE resides on the platform position. In Sstab the wind computations are performed with a consistent equilibrium (trim and displacement) position, what cannot be guaranteed for the wind tunnel tests. Considering these factors, the results obtained using WFE were very good for high angles of inclination, where the differences between the wind tunnel tests and WFE are small.

For the low angles of inclination we believe the cause of the different results is the wind force on the columns of the platform that are occluded. CFD simulations on this model have shown that, in reality, these columns do contribute for the final wind force.

Figure 11 shows a cross section of the model along with the wind flow lines that explain the differences in the result. Notice how the wind embraces column A and then hits column B. The resulting force on column B has not been taken into account by the WFE algorithm.

This problem of the WFE algorithm is not expected to happen when computing wind forces for ships.

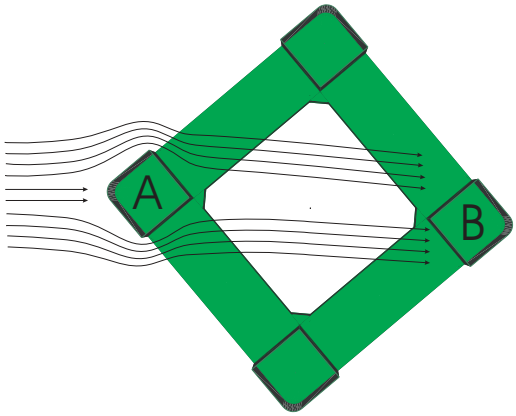

Figure 11 – Wind forces in occluded surfaces

4. CONCLUSIONS

Despite the differences observed in the test for the low angles of inclination, we believe the results obtained with the WFE algorithm are promising.

We are working to improve the algorithm formulation to consider the areas that are significantly distant from the areas that shadowed them. We believe a multi-pass algorithm, similar to the one used in the order independent transparency technique (Everitt, 2001), can be used to solve this problem. Equation 1 will probably need to be adapted to evaluate the forces in occluded surfaces. We plan to define coefficients for this situation by using CFD simulations.

Also, an implementation of this algorithm using graphics processor units (GPUs) is being considered.

5. ACKNOWLEDMENTS

The authors would like to thank Petrobras, CENPES and Tecgraf/PUC-Rio for their support and for the opportunity of working in this project.

6. REFERENCES

Coelho, L.C.G., Figueiredo, L.H. and Gattass, M., 2000, "Intersecting and trimming parametric meshes on finite-element shells". International Journal for Numerical Methods in Engineering, Vol. 47, No. 4, pp. 777-800.

Coelho, L.C.G., Jordani, C.G., Oliveira, M.C. and Masetti, I.Q., 2003, “Equilibrium, Ballast Control and Free-Surface Effect Computations Using the Sstab System”, Proceedings of the 8th International Conference on the Stability of Ships and Ocean Vehicles, pp. 377-389.

Everitt, C., 2001, “Interactive Order-Independent Transparency”. Nvidia SDK NVIDIA Corp., (http://www.nvidia.com/).

Foley, J.D., Van Dam, A., Feiner, S.K., and Hughes, J.F., 1997, Computer Graphics: Principles and Practice, Second Edition, C. Addison Wesley.

IMO, 2001, Code for the Construction and Equipment of Mobile Offshore Drilling Units.

Rij, T. van, 1994, “Object Space versus Image Space: A Comparison of Image Synthesis Algorithms”, Technical Report: CS-R9426. CWI, Department of Interactive Systems.