

OCEAN

ENGINEERING

GROUP

Visualization of Multi-Component
Waves and Wave-Body Interactions

Aravind Hariramasamy

May 2002

Report No. 02-2

ENVIRONMENTAL AND WATER RESOURCES ENGINEERING

DEPARTMENT OF CIVIL ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

Austin, TX 78712

Acknowledgements

This thesis would not have been possible without the guidance and continuous support of my adviser Dr. Spyros Kinnas of the civil engineering department. His ideas and suggestions made this research a success. I would like to thank Dr. Robert Flake and Dr. Fred Hudson of the Electrical & Computer Engineering department for reviewing my thesis and providing me with valuable comments. The National Science Foundation (Grant No. EEC-9711733) funded this project and without them this project would not have been realized. I greatly appreciate the help provided by my colleague Shreenath Natarajan in processing geometries and also in solving hydrodynamic problems. Finally I would like to thank The University of Texas at Austin for giving this opportunity for graduate studies and the excellent education and facilities given to me.

May 2002

Abstract

Visualization of multi-component waves and wave-body interactions

Aravind Kumar Hariramasamy, M.S.E.

The University of Texas at Austin, 2002

Supervisors: Spyros Kinnas

Robert Flake

The primary objective of this work is to develop a simulation tool to visualize wave environments and wave-body interactions, using 3D graphics modeling and animation techniques. The simulation tool is designed such that it is web based and can be accessed through a web browser. This thesis describes the design and implementation of the visualization tool, explaining how hydrodynamic analysis is integrated with 3D graphics. This tool is available in the electronic class room on wave theory website (<http://otrc93.ce.utexas.edu/~waveroom>) under the interactive tools section.

Table of Contents

List of Figures.....	viii
Nomenclature	x
Chapter 1 Introduction	1
Chapter 2 3D Graphics Technology available.....	4
Java 3D	5
Transform Group.....	7
Behavior node	7
Hardware acceleration.....	8
Chapter 3 Surface Generation	9
Surface generation using line strips	11
Animation Technique	15
Flipbook animation.....	15
Key frame animation	15
Animation and Transforms	16
Event Driven Animation.....	16
Chapter 4 Wave-Body Interactions.....	18
Three Dimensional waves	18
Multi – Component waves	19
Hydrodynamics	21
Computation of the Response Amplitude Operator	22
Hull Geometry.....	25
3D model for rendering.....	28
Paneled Geometry For WAMIT.....	31
Database of Coefficients.....	31

Validation	33
Chapter 5 Architecture of the Simulation and rendering program	37
Database Creation.....	38
Wave environment generation	38
Wave Animation.....	39
Solver Module	39
Interpolation	42
Hull Motion Animation	44
Applet and user interface	46
Chapter 6 Conclusion.....	49
Future work.....	49
Bibliography	51
Vita	52

List of Figures

Figure 2.1 Scene graph tree.....	6
Figure 3.1 Surface using triangular faces.....	9
Figure 3.2 Grid	11
Figure 3.3 Line strips parallel to x axis.....	12
Figure 3.4 Line strips parallel to z axis.....	13
Figure 3.5 Grid with Equation.....	14
Figure 3.6 Animation Technique.....	16
Figure 4.1 Coordinate system.....	18
Figure 4.2 Hull Stations in Autocad	26
Figure 4.3 Displaced Stations.....	27
Figure 4.4 Triangular Faces.....	29
Figure 4.5 Hull bottom view	30
Figure 4.6 Close up view of the 3D model of the Hull.....	30
Figure 4.7 Organization of Coefficients & Forces	32
Figure 4.8 Convergence of values	33
Figure 4.9 Comparison of Heave amplitude for different panel density	34
Figure 4.11 Roll Vs CG	36
Figure 5.1 Overall Architecture.....	37
Figure 5.2 Comparison of RAO(Roll) computed by WAMIT and Solver module	41
Figure 5.3 Comparison of RAO(Heave) computed by WAMIT and Solver module	41
Figure 5.4 Comparison of Actual added mass coefficient(a_{33}) Vs Interpolated value	43
Figure 5.5 Animation Implementation.....	45

Figure 5.6 Screen shot showing monochromatic waves.....	46
Figure 5.7 Screen shot showing multi-component waves.....	47

Nomenclature

A	Wave amplitude.
a_{ij}	Added-Mass Coefficients; $i = 1,6; j = 1,6$
b_{ij}	Damping Coefficients; $i = 1,6; j = 1,6$
c_{ij}	Restoring forces; $i = 1,6; j = 1,6$
CG	Vertical location of hull's center of gravity with respect to the free surface level.
d	Depth of the water body.
g	Gravity acceleration.
k	Wave number.
l	Wave length.
M_{ij}	Inertial Mass Matrix; $i = 1,6 ; j = 1,6$
T	Time period.
$U_j(t)$	Velocity of motion at time t for each degree of freedom; $j = 1,6$
Value _{inp}	Interpolated value.
X_i	Exciting Force. ; $i = 1,6$
y	Wave elevation.
θ	Wave Heading angle.
ω	Angular frequency of the wave.
ϕ	Phase of the wave.
ξ_j	Response amplitude Operator. ; $j = 1,6$

Chapter 1

Introduction

Visualization in the engineering world has been a key in deriving a wide picture of numerical results and better understanding and acuity of what the results really mean. This thesis describes a project that attempts to develop a comprehensive tool which enables the user to visualize multi-component waves and wave-body interactions.

“The purpose of computing is insight, not numbers” - Richard Hamming 1962

The advances in computer technology and specifically three-dimensional graphics computation have widened the horizon in the field of 3D visualization. Since the computing power of an average computer has risen so drastically over the years, what was considered to be advanced and complicated technologies have become common place in today's world. The accelerated growth of 3D- graphics processing power of a desktop PC, though fuelled by the gaming industry, also opens a window to a realm where this technology can be taken advantage of for engineering purposes.

It has been increasingly difficult for modern sciences to be communicated in print. DNA sequences, molecular models, medical imaging, simulated flights etc have extended the necessity for a visual medium to be taught and understood. The field of wave and hydrodynamics are no exceptions; the sheer amount of numerical data involved immediately brings in the need for visualization.

Visualization is a form of communication that transcends application and technological boundaries.

The advent of technologies like Java has made code development for different platforms easier, thus reducing problems of scalability and portability to a large extent. The code that is developed in Java can be easily scaled to handle a much larger problem by simply switching to a platform that supports high end computers; an example is moving from a desktop to a silicon graphics workstation and vice versa. The simulation tool is completely developed in Java, using the OpenGL graphics package (described later) provided by Sun Micro Systems Java3D, thus making it theoretically possible to switch platforms with ease. It is capable of high user interactivity and manipulation and provides for straightforward visualization. Since this tool is designed to be an online educational aid, it becomes critical that the simulation can run on an average desktop PC. This lays down multiple constraints on design and development.

3D graphics on a low-end machine may unduly overload the system and make the simulation tool unworthy for use. This calls for innovative methods to achieve rendering of surfaces. Traditionally surfaces are created by polygon faces and normals, this needs extensive processing power to simulate wave surfaces continuously over time. Instead of surfaces created by polygons, an alternative method would be to establish surface features by primitive line strips, which would reduce the complexity of the simulation to a great extent. Though a compromise is made by eliminating textures, reflective-ness etc. this innovative

method can give valuable functionality to thin clients and low end computers. Such a methodology is described in this thesis.

The field of Hydrodynamics is directly associated with numerical results. These numerical results are highly complicated and difficult to make sense of without proper visualization tools. A good visualization tool would aid teaching/research immensely and it is almost essential to understand the meaning and significance of numerical results. 3D graphics is an excellent medium for representing numerical results when compared to representing data in the form of 2D graphs which in essence lacks the depth and detail a 3D simulation can provide.

This project does not attempt to arrive at hydrodynamic results but rather borrows the result from a software package which deals with hydrodynamics and uses those results to simulate the motions of a floating body in a multi-component and multi-directional wave environment.

In the subsequent chapters, a detailed account of the features, architecture, implementation and theory behind this simulation tool is elucidated.

Chapter 2

3D Graphics Technology available

A simulation tool which has to be compliant with multiple platforms and provides smooth 3D graphics calls for the latest technology in the 3D graphics field. OpenGL [3] and DirectX are application programming interfaces (API) that are widely used hardware interfaces which provide enhanced functionality and performance.

OpenGL was developed by SGI and the name OpenGL is a registered trademark owned by the company. For many years, the only real access to the OpenGL API was by way of high-end UNIX workstations, which were used for engineering, scientific and visualization work. SGI workstations were advanced and had extended hardware for 3D acceleration. OpenGL is an API to assist in delivering 3D drawing services in real-time environments. The API is carefully designed to allow large portions of the work to be carried out with the assistance of 3D hardware. A full implementation will fall back from hardware acceleration to a software version for any features requested that are not available from the hardware. This is referred to as falling back (software acceleration) from the fast path to a slow path and can dramatically lower the frame rate delivered. OpenGL is the only truly cross-platform 3D API available that is designed for hardware acceleration. An application using OpenGL to run on one system, should port to another platform with little more than running a build in the new environment.

OpenGL continues to be the only choice for most high-end, real-time applications.

JAVA 3D

The Java 3D [1] API provides a set of object-oriented interfaces for applications that require high-performance, interactive 3D graphics. With the Java 3D API, developers can take advantage of a simple, high-level programming model that enables them to build, render, and control the behavior of 3D objects and visual environments. The Java 3D API takes advantage of existing hardware accelerators via its use of low-level APIs such as OpenGL and Direct3D. This layering allows applications written using the Java 3D API to run on any platform with a Java virtual machine and an OpenGL or a Direct3D implementation. Java 3D API was designed to offer a high-level, object-oriented, programming paradigm which enables integration with other modules that are not related to the 3D rendering. Support for run-time loaders is included to allow Java3D to accommodate a wide variety of file formats. The Java 3D API benefits developers in a diverse set of application areas including scientific visualization, animation, web site design, games and design automation. Java3D implements scene graph architecture and is unique in that aspect. A scene graph environment allows an application developer to form a scene of objects as an interrelated hierarchy.

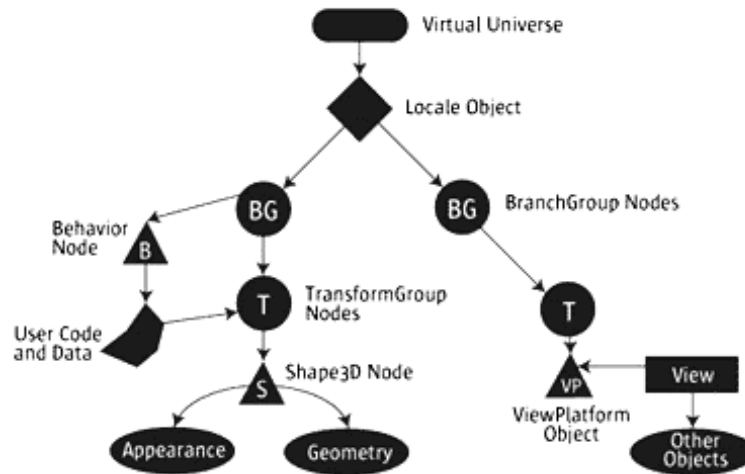


Figure 2.1 Scene graph tree [1]

Figure 2.1 shows the basic architecture of a scene graph based rendering system. It can be seen that the geometry is lowest in hierarchy in the graph structure taking the form of a leaf node. The next higher level of shape-3D, which consolidates the vertex and face information provided by the geometry and combines it with texture mapping and appearances (opacity, material, etc.) for the faces to create an independent entity within the graph. The transform group node describes how the geometry moves and transforms within the scene graph, the transform group can be integrated with an universal scale to make the geometry fit into the virtual universe's scale and also interacts with user code and data. One level up is the behavior node which is closely tied to the user code, which tells the transform group how to transform the geometry depending on what the main user code wants to do with the geometry (rotate, translate, scale etc.). All these entities are collected together by the branch group and this is added to a locale object; there can be more than one locale object within a scene graph. A parallel

locale object can also be created with a different set of branch groups. The locale objects are finally added to the virtual universe which is rendered on to the screen. The virtual universe has a universal co-ordinate system and all other nodes follow that co-ordinate system. The length scale of the system can be adjusted to the programmer's and the application's needs.

Transform Group

The Transform Group [1] node contains a transform; it specifies a single spatial transformation, via a Transform3D object, that can position, orient and scale all of its children.

Behavior node

Behavior nodes [1] provide the means for animating objects, processing keyboard and mouse inputs, reacting to movement, and enabling and processing pick events. Behavior nodes contain Java code and state variables. A Behavior's Java code can interact with Java objects, change node values within a Java 3D scene graph, change the behavior's internal state: in general perform any computation it wishes. Simple behaviors can add surprisingly interesting effects to a scene graph. For example, we can animate a rigid object by using a behavior node to repetitively modify the TransformGroup node that points to the object we wish to animate. Alternatively, a behavior node can track the current position of a mouse and modify portions of the scene graph in response. Behavior is an abstract Java class that defines properties common to all behavior objects in Java 3D. There are several predefined behaviors that are subclasses of Behavior. Additionally, a Behavior leaf node may be sub-classed by the user. Unlike many

other 3D graphics APIs and even most 2D APIs, behaviors form a fundamental part of the Java3D system. As a rough estimate, 80% of all non-static scene graph interactions at the code level will have originated with a behavior [2]. Typically a behavior is what triggers the code to start running, which in turn makes modifications to the scene graph. A cascade effect may then start where one behavior triggers a series of follow-on actions.

HARDWARE ACCELERATION

In modern day computers, hardware acceleration for 3D graphics is an inevitable part. Hardware acceleration for 3D graphics provides fast and efficient transformation and rendering, which the main CPU would not be able to provide. The graphics card takes care of the complicated transforms and does it rapidly by utilizing its graphics processing unit (GPU). Since the GPU is dedicated only for 3D transforms it makes on screen rendering faster. The OpenGL API provides an abstraction layer over the hardware and gives easy access to the hardware features. Java3D is added as a layer above OpenGL, makes any 3D application built using Java3D immediately take advantage of any hardware acceleration present in the system. Since Java3D is the top level abstraction layer, porting platforms is done with ease, which makes Java3D truly portable.

Chapter 3

Surface Generation

Conventionally surfaces are represented by polygons (triangles, quadrilaterals etc.) along the surface. These primitives, when combined together in a definite pattern, form the surfaces.

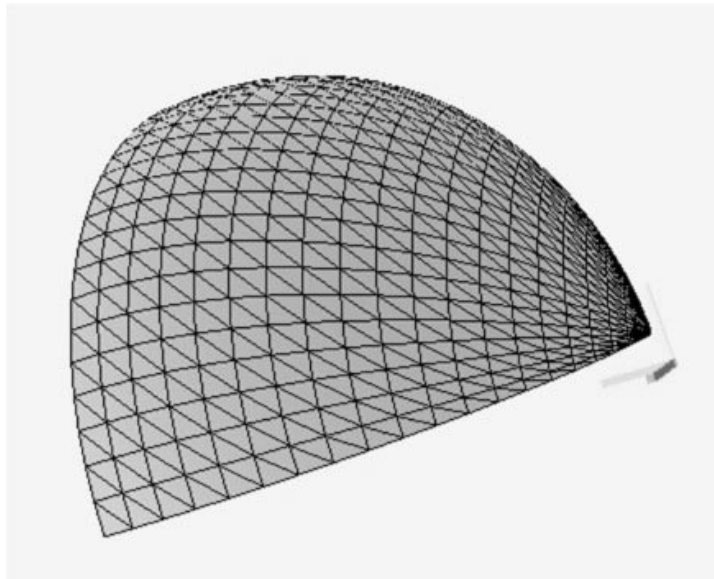


Figure 3.1 Surface using triangular faces

In Figure 3.1 it can be seen how triangle primitives are put together to describe a smooth curved surface. The smoothness of this surface is determined by the number of triangles that are packed in a unit of 3D surface. The smoothness of the surface is directly proportional to the density of triangles. The normal vector for each triangular plane can be calculated and stored along with the vertex information. The normal would assist with the appearance and viewing

of the triangular faces and hence the surface. Along with these triangular faces, texture maps can be created to give the surface a texture.

This kind of surface generation needs plenty of memory and the calculation of vertex co-ordinates are computationally complex. 3D hardware acceleration in the form of graphics cards cannot make the surface generation faster because the hardware can only work on a base geometry and manipulate it and not generate it. The vertex generation has to be done by the CPU and generating thousands of vertices takes a considerable amount of time.

The simulation tool under consideration needs a surface structure generated every fraction of a second. If we were to use the conventional means to generate the surface, it is almost impossible to do so on a desktop computer. It may be possible if the vertex values are pre-computed and readily available, but that is not the case. Every vertex co-ordinate has to be calculated using the three-dimensional wave equation and iterated for different wave components for a given wave environment. This would drastically bring down the number of frames per second the computer can render on screen.

To achieve smooth animation of the wave surface, extreme primitives like line strips can be used to mimic a solid surface. Though it may not be as detailed as having triangular or polygon faces for the surface, to realize smooth animation, appearance has to be compromised. The purpose of the visualization is to convey information, not provide pleasing pictures.

SURFACE GENERATION USING LINE STRIPS

Line strips are the simplest means to accomplish a reasonably accurate surface geometry, the surface would resemble a wire frame structure and it is acceptable for this application. When perpendicular line strips are superposed, a grid structure can be obtained; the grid shape can be altered by generating the line strips by using a governing set of equations. Each vertex of the line segment can be generated by an equation which has x,y and z variables. Values along one axis can be found by keeping values on the other axis constant. Example: for each value of z, x can be varied and the corresponding y values, this would yield a set of line strips which are parallel to the z axis and for each value of x, z can be varied and the corresponding y value can be obtained, this would yield another set of line strips parallel to the x axis. When these two perpendicular sets of line strips are superposed one over another, the resulting grid would reflect on the characteristics of the equation that generated the line strips.

Example: Figure 3.2 shows the simplest possible case where

$$y = 0 : x = -\infty \rightarrow \infty \text{ \& } z = -\infty \rightarrow \infty \quad (3-1)$$

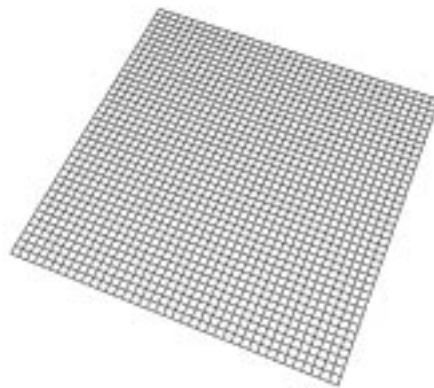


Figure 3.2 Grid

To generate a surface for the 3D equation $y = a \cos(kx + \omega t) + a \sin(kz + \omega t)$ (3-2), the x and z component line strips have to be generated separately and superposed to create the equation surface. For a time instant Let $t = 1$

```

for(x = minX; x = maxX ; x = x + stepX)
{
  for(z = minZ; z = maxZ ; z = z + stepZ)
  {
     $y = a \cos(kx + \omega t) + a \sin(kz + \omega t)$ 
  }
}

```

This nested iteration would result in a set of line strips which are parallel to the x-axis, such a set is shown in Figure 3.3.

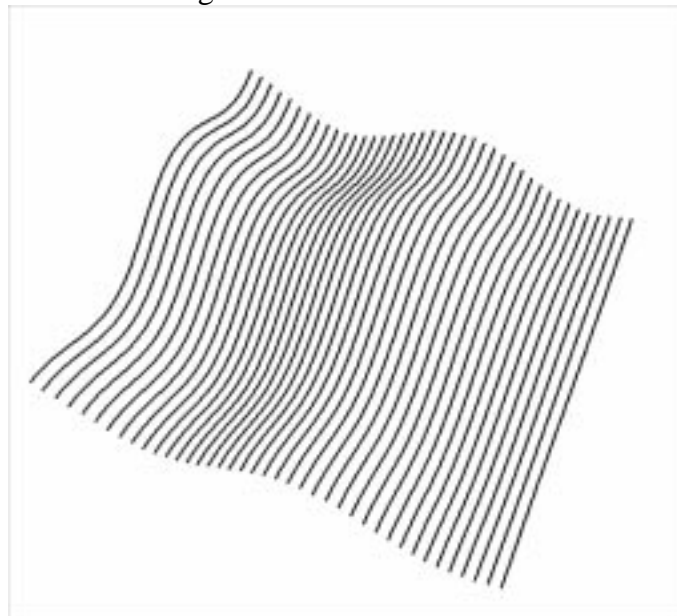


Figure 3.3 Line strips parallel to x axis

```

For a time instant Let t = 1
for(z = minZ; z= maxZ ; z = z + stepZ)
{
    for(x = minX; x = maxX ; x = x + stepX)
    {
        y = a cos(kx + ωt) + a sin(kz + ωt)
    }
}

```

Note that the nesting order is reversed. This nested iteration would result in a set of line strips which are parallel to the y-axis, such a set is shown in Figure3.4

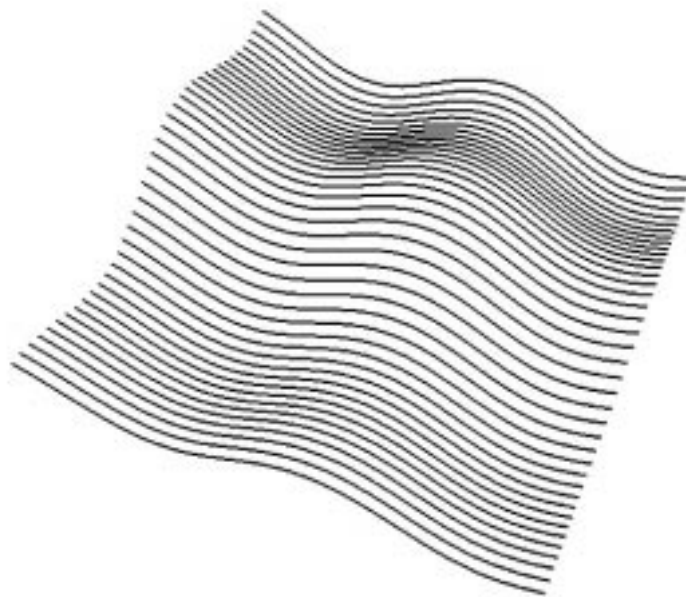


Figure3.4 Line strips parallel to z axis

The surface required is easily obtained by placing the previous two strips one over the other.

Figure 3.5 shows the final surface obtained from the intermediate line strips. The wire frame surface generated will not exhibit the properties surfaces made of polygons, but it is much faster to generate than polygon surfaces.

The spacing between the strips (stepZ and stepX) can be decreased to improve the accuracy of the surface. This kind of surface can also be used to depict 3 dimensional data by using each data point for a vertex and interpolating the data for intermediate vertices. Any equation with not more than three dependant variables can be used to create this surface plot. If the independent variable (in this case time) can varied by incrementing time by a definite time step and generating the surface for each time step, the series of surfaces obtained can be animated to observe the patter and the effect of the independent variable on the system.

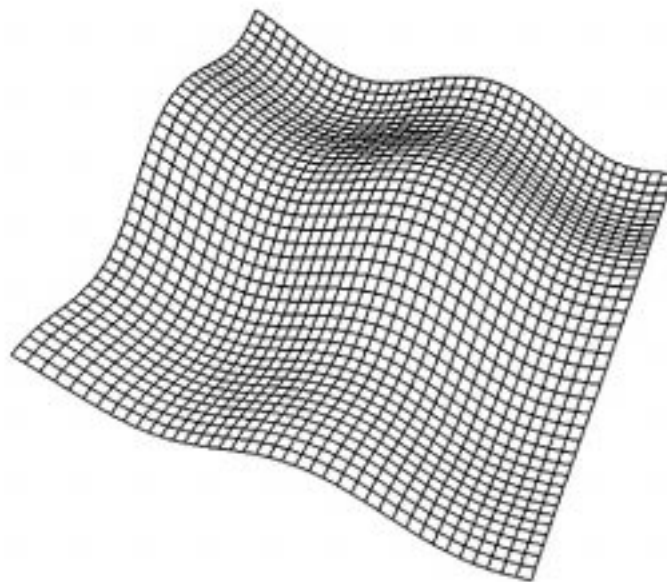


Figure 3.5 Grid with Equation

ANIMATION TECHNIQUE

Animation is the process of generating and replacing one screen frame with another in a continuous time sequence, with equal time interval between two frames. Animation facilitates effective simulation of time dependent systems and it can get quite complicated depending on the complexity of the system under consideration. There are different techniques to implement animation and it is never a fixed implementation.

Flipbook animation

This is a well known technique. The generated images are displayed one after the other. Its name is attached to the thumbing or flipping through a series of images [8].

Key frame animation

For this technique one only has to generate so-called key frames. Key frames mark changes in the characteristics of the motion, for example the sudden change in the direction of motion of an electron due to a collision with an ion. Interpolation techniques are used to generate a set of images between two key frames. The larger the interpolated set of images the smoother the conversion from one key frame to the other will appear to the viewer [8].

The above mentioned animation schemes does not deal with the generation of content associated with the frame, it rather assumes the presence of content. But in case of real time simulation of physical phenomenon the content has to be generated to be placed in the frame, this poses a significant challenge in

frame generation and sequencing of the frames in the right order and at the right instant.

ANIMATION AND TRANSFORMS

Animation can be brought about in certain cases by continuously transforming (rotation, translation, scale etc.) the geometry; this kind technique can be applied only to cases where the geometry itself does not change. The transitions are very smooth since most of the transforms can be taken care by a graphics processing unit which is independent of the CPU.

EVENT DRIVEN ANIMATION

In this simulation tool a unique animation method is employed and the way it is done is described below.

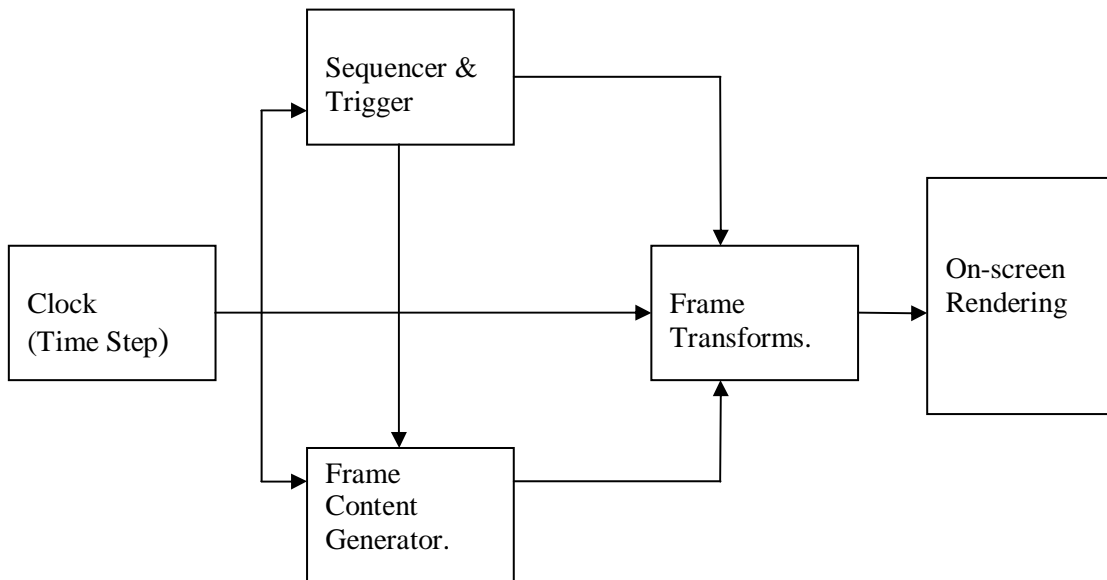


Figure 3.6 Animation Technique

Figure 3.6 describes the architecture of the animation scheme used. The core of the animation is the time step generator, which is nothing more than an event generated at appropriate time intervals. This makes the animation event driven and completely parallel to other process that may be taking place along with the animation. With each time step the sequencer trigger is triggered which in turn triggers the module that generates the content for the frame. When the content is ready the sequencer puts the frame on screen at the proper instant at which the frame is supposed to be rendered. Before the frame is rendered on screen, it is subjected to the transforms it has to undergo ahead of rendering; this ensures that the content is not out of place in the virtual universe.

There is a bottle neck introduced in to this system by the frame content generator module. Depending on the complexity of content generation the whole animation process is slowed down or expedited. This unreliability can be resolved by buffering frames/content ahead of on-screen rendering. Buffering takes away the bottle neck of content generation, but only a fixed time period of rendering can be done if the buffer runs out at a faster rate than frame creation.

Chapter 4

Wave-Body Interactions

THREE DIMENSIONAL WAVES

An ocean wave environment is usually composed of waves of different, wavelength, amplitude, phase and heading angle. This composition of waves is termed as multi-component and multi-directional wave environment. The wave displacement at any given location is a function of the wave heading angle and the instant of time with wavelength and time period being constant. A three dimensional wave has a heading direction and can have a velocity vector directed anywhere parallel to the plane of wave propagation.

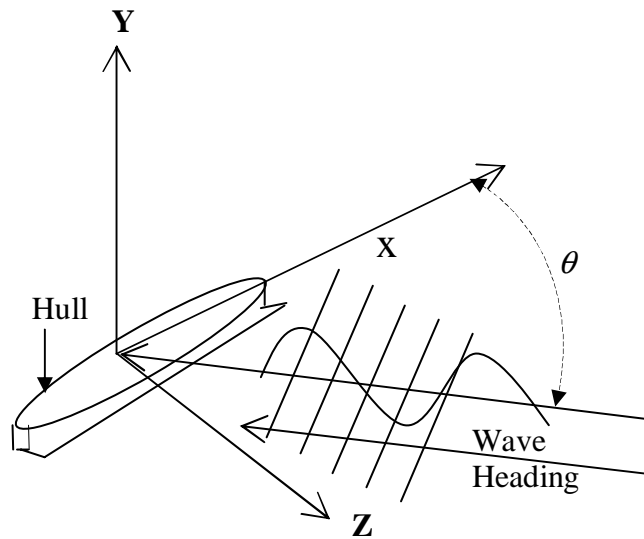


Figure 4.1 Coordinate system

Figure 4.1 shows how the coordinate system is defined and also defines the heading angle θ with respect to the coordinate system.

The following equation describes a 3D wave in a 3D co-ordinate space:

$$y = A \cos(kx \cos \theta + kz \cos \theta + \omega t - \phi) \quad (4-1)$$

where:

y = displacement

A = wave amplitude

k = wave number.

θ = Wave Heading angle.

ω = Angular frequency.

ϕ = Phase.

Multi – Component waves

A typical ocean wave environment is composed of not just one wave, but of numerous waves with different wave parameters. The final wave form is the interference due to the component waves. The superposed waves generate unique wave patterns which can be analyzed using Fourier transforms and determine the energy distribution between frequencies and also determine the fundamental frequency. Ocean wave environments can be subjected to such analysis and have its characteristics studied and it is been found that each ocean has its own wave frequency distribution.

Waves behave differently in water bodies that are shallow and those which are deep. The water bodies are classified by the following set of definitions:

d – depth ; λ - wavelength

If $\frac{d}{\lambda} < 0.40$ then the water body is termed as shallow water.

If $0.4 < \frac{d}{\lambda} < 0.5$ this is transitional water

If $\frac{d}{\lambda} > 0.5$ this makes the water body deep water.

The way the waves behave and the wave parameters are modified depends on the above classification. This simulation tool only deals with deep water characteristics, and the time period is calculated from wavelength and vice versa using deep water formula.

$$\lambda = \frac{T^2 g}{2\pi} \quad (4-2)$$

Where

λ = wave length

T = Time period.

g = Gravity acceleration.

The simulation program automatically computes the wavelength or the time period and vice versa if either one of the parameter is specified by the user. The computed deep water parameters are then used for the simulation. The

incoming wave elevations are shown without the effects of diffraction and radiation due to the hull included. A future improvement should include these modifications.

HYDRODYNAMICS

Webster definition for hydrodynamics:

“Branch of physics that deals with the motion of fluids and the forces acting on solid bodies immersed in fluids and in motion relative to them “

Modern day ocean engineering deals with structures floating on a fluid medium. This is of great importance to industries like oil & gas and shipping, as these industries heavily rely upon ships and off shore structures for oil production and transportation. Ensuring stability of those structures is of vital importance since life and property is at stake. Engineering those structures needs special consideration unlike terrestrial structures, as influence of the water medium and waves affect the stability to a large extent.

The capability to predict the response of the structures to the wave environment surrounding the structure helps the designers to come up with better designs. Predicting the response is not straightforward, as the structures are not simple in their geometry. Marine hydrodynamic principles help to solve such problems, by applying basic physics and mathematical analysis to arrive at a solution for the problem in hand.

Any given body has six degrees of freedom in 3D space and the motions for each degree of freedom are influenced by various forces acting along different directions [4]. The motions for each degree of freedom are termed as follows:

1. Surge
2. Sway
3. Heave
4. Roll
5. Pitch
6. Yaw

The motion amplitudes along and about each axis are the responses of a body to its wave environment on which it is floating. The motion is harmonic and repeats itself as long as the wave environment doesn't get modified. For each component wave with a time period and heading angle the response of the floating body is unique. In a multi component wave environment the final motion of the floating body is the resultant motion caused by the component waves.

Computation of the Response Amplitude Operator

Any floating body has certain variables associated with them depending on the nature of the waves in which they are floating on. Those variables are:

- Added mass coefficients
- Damping Coefficients
- Restoring forces
- Exciting forces

These variables have to be computed before the response amplitude operator can be arrived at. The computation of these variables requires a panel method analysis of the geometry under consideration and applying boundary

conditions on these panels and finally calculating the coefficients for the whole geometry.

Definitions of these variables are listed below:

Added Mass Coefficients

For a body undergoing small oscillations in an unbounded fluid, they represent the force component proportional to the acceleration [4].

Damping coefficients

They represent the forces proportional to the velocity of the body, for this reason it is known as the damping coefficients. The presence of such a force results from the generation of waves on the free surface, due to the motions of the body. These waves radiate outward, with a corresponding energy flux that can be computed [4].

Restoring Forces

The restoring force is the static component which remains constant with change in wavelength and heading direction of the wave and it is proportional to the buoyancy of the floating body which provides an upward thrust when the body submerges from the free surface [4].

Exciting forces

This force is proportional to the incident-wave amplitude and it is due to the energy the waves transfer to the body under consideration. The exciting force is usually calculated for a unit amplitude wave [4].

Mass matrix

It is the matrix which has the inertial force for the six components, associated with the body mass [4].

Using these variables, a system of six simultaneous equations [4] can be formulated for the six motions involved with the body.

$$\sum_{j=1}^6 \xi_i l \left[\frac{-\omega^2}{g} (M_{ij} + a_{ij}) + i \frac{\omega^2}{g} b_{ij} + c_{ij} \right] = AX_i \quad (4-3)$$

Where

a_{ij} = Added-Mass Coefficients.

b_{ij} = Damping Coefficients.

M_{ij} = Inertial Mass Matrix.

c_{ij} = Restoring forces.

X_i = Exciting Force.

A = Wave Amplitude.

ξ_i = Response amplitude Operator.

ω = Angular frequency.

By solving the system of equations the response operator ξ_i can be calculated. The response amplitude operator is also known as the transfer function. This can be used to predict the position of the body at each time instant and the velocities are sinusoidal in time with the same frequency as the incident waves. The six velocities are given by

$$U_j(t) = \text{Re}(i\omega \xi_j e^{i\omega t}) \quad (4-4)$$

Where

$U_j(t)$ Is velocity at time t for each degree of motion (j = 1 to 6)

“Waves at MIT” (WAMIT) [5] is a radiation/ diffraction program for the analysis of the interaction of surface waves with offshore structures. It is hydrodynamic analysis package which can compute the added-mass, damping coefficients and the restoring and exciting forces for a structure whose geometry information is known and can be paneled. WAMIT is based on three dimensional panel method and assumes linearized free surface conditions, and it can handle any paneled geometry. It also solves the system of six simultaneous equations and computes the response amplitude operator (RAO). WAMIT takes inputs like the body geometry, the wavelength and direction of the incident wave to compute the coefficients and RAO, the geometry of the body has to be paneled in a way WAMIT can handle.

Hull Geometry

In this project, given the station information of a hull, a procedure was developed to panel the hull geometry and make it suitable for analysis using WAMIT. It cannot handle geometries described in conventional formats like AutoCAD (which is the most common file format to describe hulls), as it needs the vertex to be ordered in specific manner and also uses quadrilateral panels.

A method was developed for this project to convert an AutoCAD file with station information to a paneled geometry suitable for WAMIT, with minimal manual interference. The hull geometry under consideration was available in AutoCAD DXF (data exchange file) file format. The following was the method adopted to create panel based geometry:

Defining station position

Hull geometries are usually represented in 2D in terms of consecutive station curves. They do not have the information as to where the stations are positioned along the hull. It becomes critical to place them in the right position before they can be paneled.

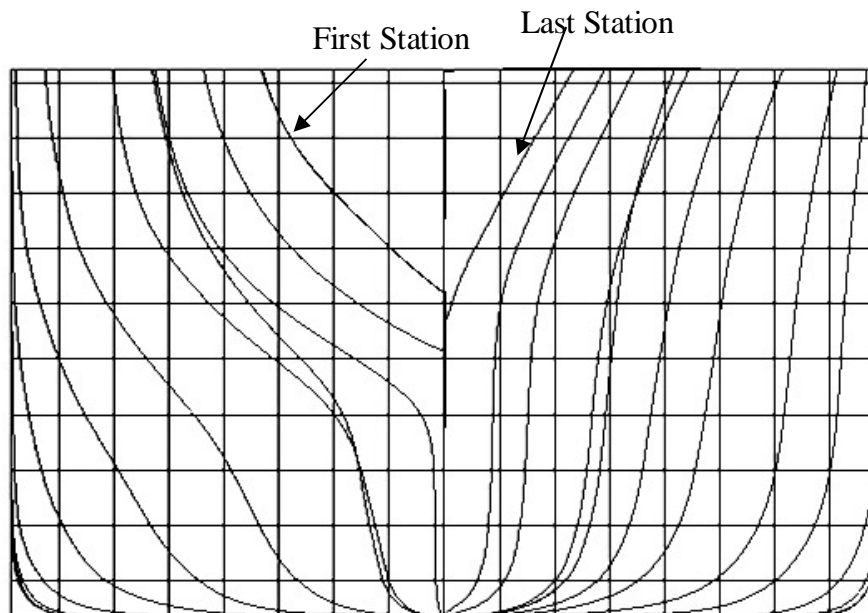


Figure 4.2 Hull Stations in AutoCAD

Figure 4.2 shows how each station is arranged in 2D, the first station starts on the left hand side and continues till to the middle and the left hand sides has the rest of the stations arranged in ascending order. As it can be seen that all the stations are defined using only 2 co-ordinates, in this case X and Z, the Y coordinates have to be introduced manually. By only introducing the Y co-ordinate, it would be possible for any algorithm to distinguish overlapping stations and also establish the order of arrangement. Although it is theoretically possible to develop an algorithm to distinguish the station curves by analyzing the curvature of curves that is beyond the scope of this project. The stations were put in their designated locations by using AutoCAD2000 by choosing each curve manually and assigning them new Y co-ordinates.

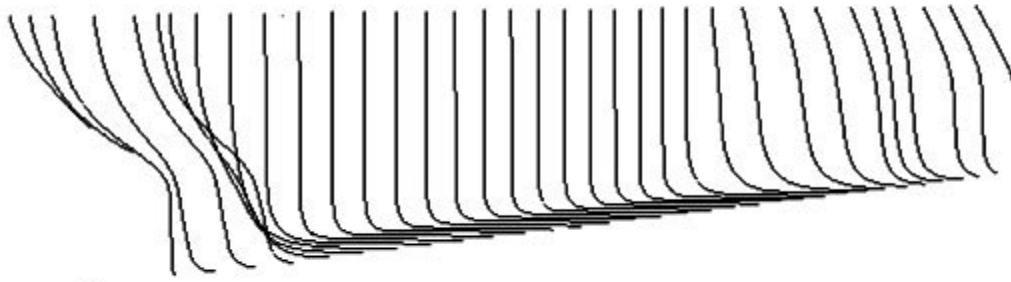


Figure 4.3 Displaced Stations

Figure 4.3 shows a perspective view of the curves at the right Y location. Now the AutoCAD DXF file is ready for processing.

DXF

Data Exchange format is a unique format used by AutoCAD whose documentation is readily available in the public domain; these files can be parsed through to retrieve station curve information as each of the stations is represented as a poly-line entity within the DXF file. Poly-lines can be easily tracked inside the file since the vertices of identical poly-lines are grouped together under a common header [6]. Once the station information is retrieved they are stored in separate files for easy processing.

3D model for rendering

Once the station vertices are available they are equally spaced using a dedicated interpolation algorithm¹. When the station vertices are equally spaced it is only a matter of arranging the equally spaced vertices between two stations and forming triangular panels.

Figure 4.4 shows how triangular face primitives are created from two stations with equally spaced vertices. Vertices of station 1 (V_{1i}) are connected with their corresponding counterpart vertices of station 2 (V_{2i}) to form triangular geometry primitives. A triangular face has the following sequence of vertices.

Triangular panel 1 >> V_{11}, V_{22}, V_{21}

Triangular panel 2 >> V_{12}, V_{22}, V_{11}

¹ This algorithm was developed by Shreenath Natarajan, Masters Student, Ocean Engineering Group, Civil Engineering, University of Texas at Austin.

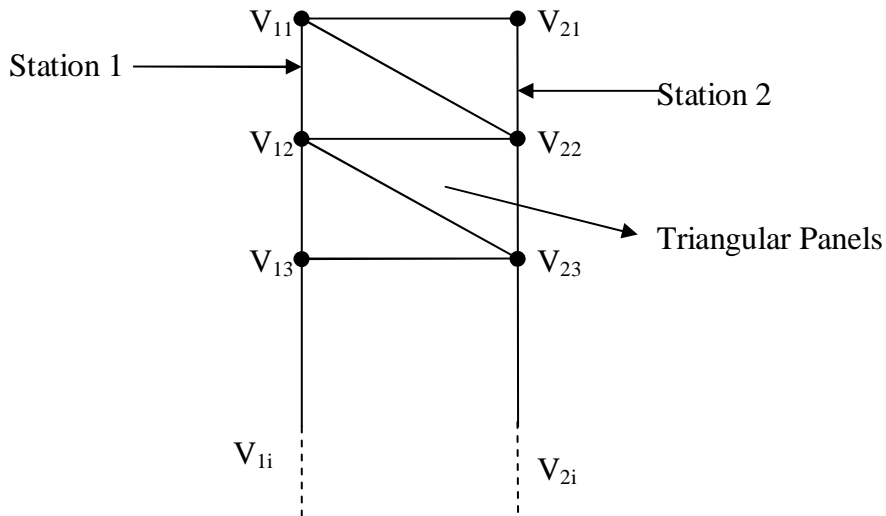


Figure 4.4 Triangular Faces

This procedure is repeated for all stations till the complete 3D model is created. This 3D model can be converted to a file format for which Java3D loaders are available. In this case it is the Wavefront OBJ file format. This conversion is done using a third party software. For this project a software package called 3D exploration (Now known as Deep exploration) from a company called Right Hemisphere [9] was used. This software can do geometry transforms and file format conversions and it is quite effective for such simple operations. To the 3D model an additional super structure can be added on top of the hull manually using AutoCAD or any other 3D editing software. This would make the hull look more realistic on the final simulation screen.

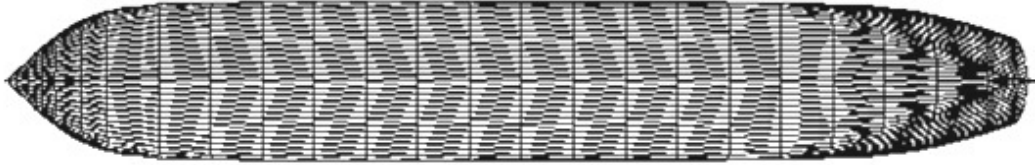


Figure 4.5 Hull bottom view

Figure 4.5 shows the bottom side of the Hull 3d Model, It can be observed how the shape of the hull is distinct and completely derived from the station information available from the original DXF file.



Figure 4.6 Close up view of the 3D model of the Hull

Figure 4.6 shows a zoomed close up of the bottom of the hull model, and it reveals how the panels are organized as mentioned before.

Paneled Geometry for WAMIT

The stations are truncated at the draft line using an interpolation algorithm² and they are paneled to form quadrilateral panels in a method similar to forming triangular faces. The vertices of the quadrilateral panels are ordered in a clockwise direction. Clockwise ordering indicates that the panel faces an outward direction and adds up to a positive volume and counter-clockwise signifies that the panel faces an inward direction and adds to a negative volume [5].

WAMIT can accept panels in any order and panels from any part of the geometry can be randomly jumbled and it would not affect the final computations. The units and scale of the geometry is has to be uniformly maintained through out the hull to ensure accuracy.

Database of Coefficients

In order to make available the coefficients that are necessary to solve the six equations of motions for the solver module, WAMIT is run for a range of wave time periods with equal intervals and for heading angles from 0 to 180 degrees at equal intervals. The values from WAMIT are stored in a hierarchical way for fast access, this is critical because the simulation tool has to find these coefficients as fast possible in the most efficient way.

For a given hull geometry, the restoring forces are constant, and added mass coefficients are constant with heading angle, but varies with the time period

² This algorithm was developed by Shreenath Natarajan , Masters Student, Ocean Engineering Group, Civil Engineering, University of Texas at Austin.

of the incident wave. The exciting forces are the only ones which change with both time periods and heading direction. The values are stored in a directory structure so that accessing the coefficients is as easy as forming a path string to the values, by this method access is instantaneous. The interval between one time period and the next is adjusted so that interpolation for values in between is close to the actual value.

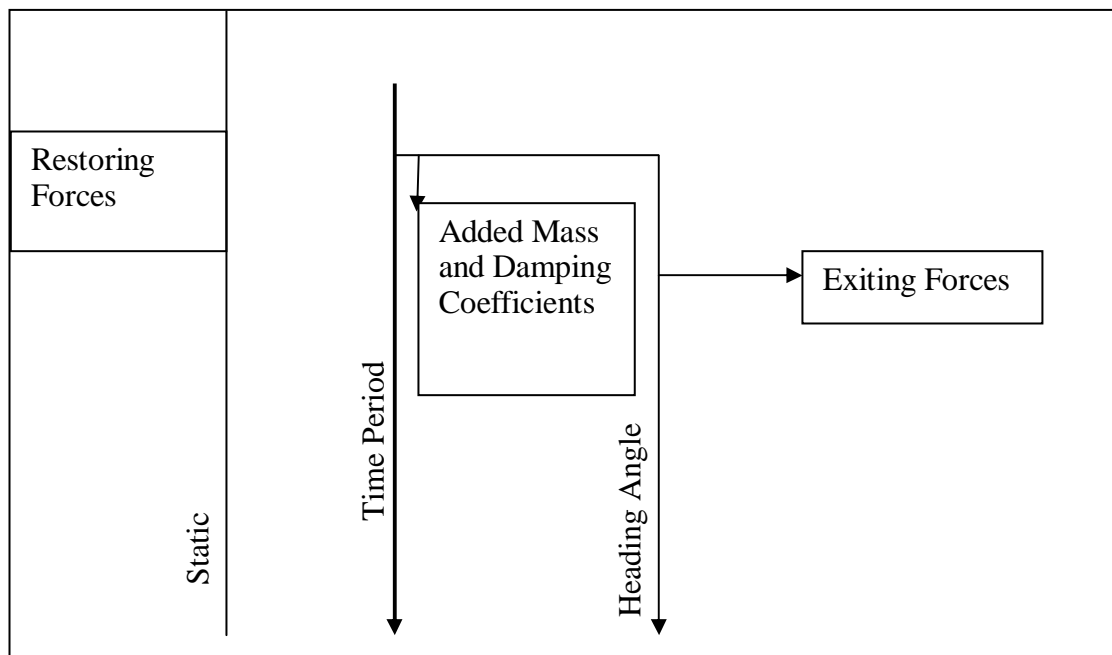


Figure 4.7 Organization of Coefficients & Forces

Figure 4.7 shows how values are stored in the database. This way of storing data in a directory structure makes the code highly portable and reduces the hassle of setting up databases whenever the code is moved around between computers.

VALIDATION

The whole process of evaluating the hydrodynamic data for a hull has to be validated before this procedure can be applied to a real hull. For this purpose a geometry whose hydrodynamic data is known and established is used to test the procedure. In this case a fully submerged half sphere is used to analyze its motion characteristics and compare it with established results. The sphere was fully submerged and has a considerable size of ten meter radius. The added-mass coefficient values converged with an increase in the number of panels and they matched with analytical values. The hull geometry was paneled with different panel densities and the added-mass and damping coefficients were observed and it was found to converge, thus validating the procedure and eliminating any possibility of random values being generated. Figure 4.8 shows the convergence study on the hull geometry.

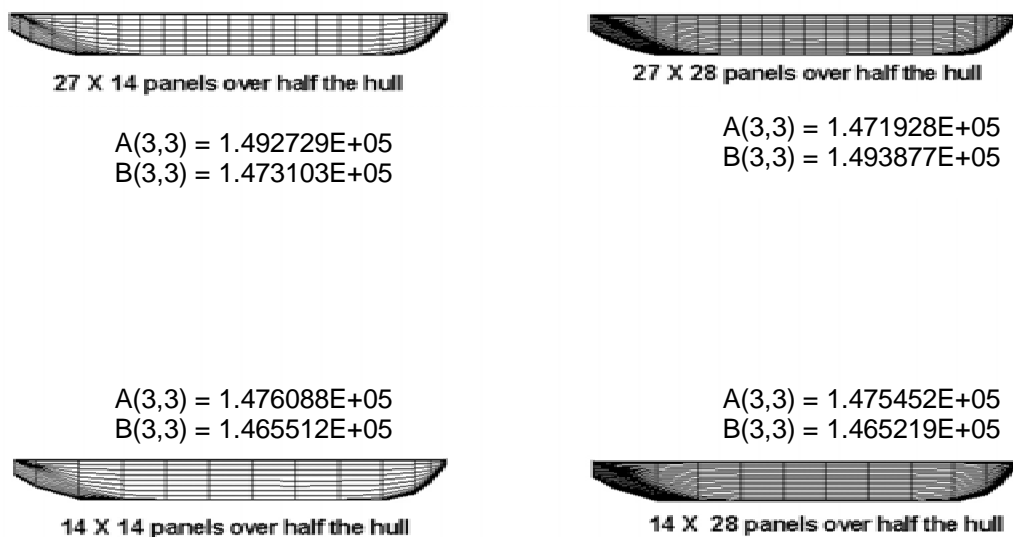


Figure 4.8 Convergence of values

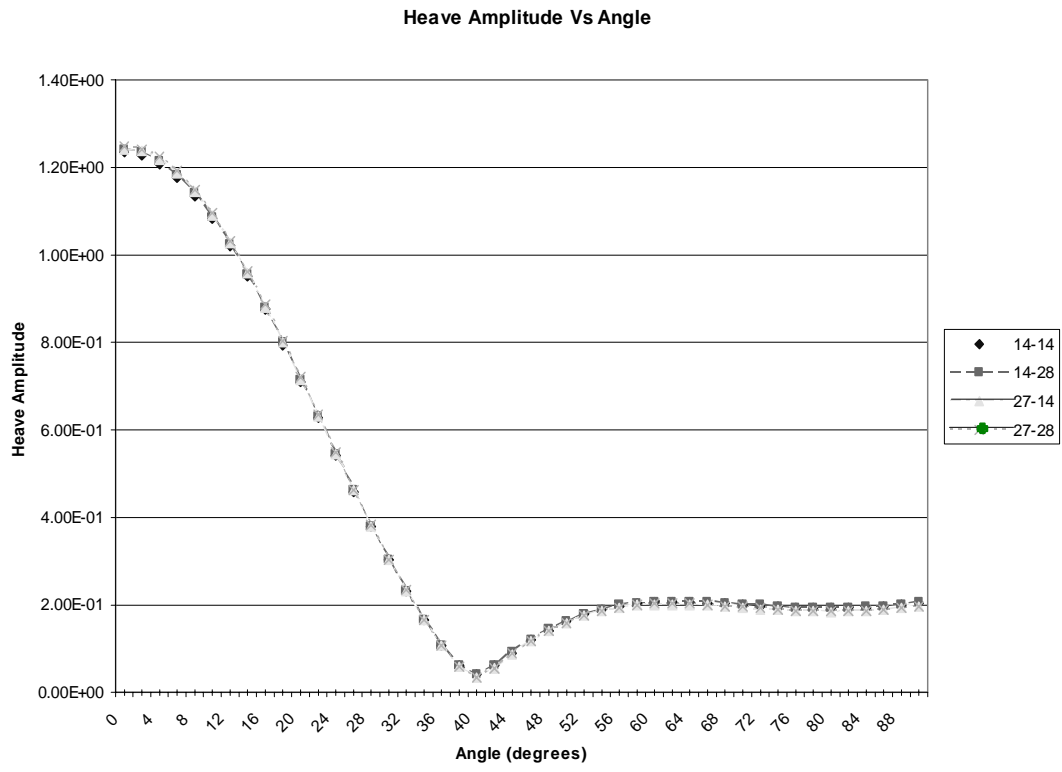


Figure 4.9 Comparison of Heave amplitude for different panel density

The non dimensional heave amplitude is also studied for different panel densities and it can be noticed from Figure 4.9 that they converge.

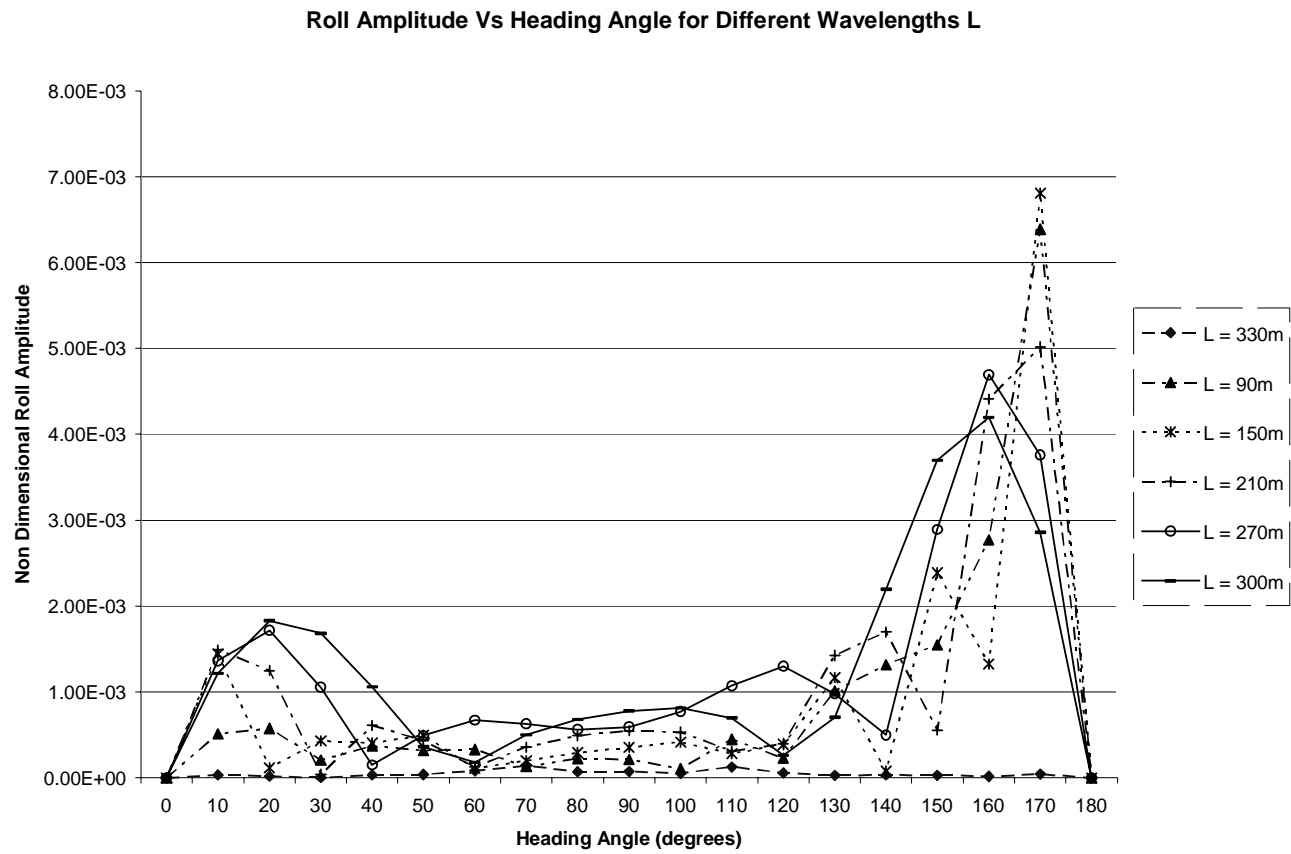


Figure 4.10 Change in Roll Amplitude with Wavelength

Figure 4.10 shows a graph depicting the change in roll amplitude with heading angles for different wavelengths. It can be inferred that the roll amplitude is not only affected by the heading angle but also by the wavelength of the incident wave.

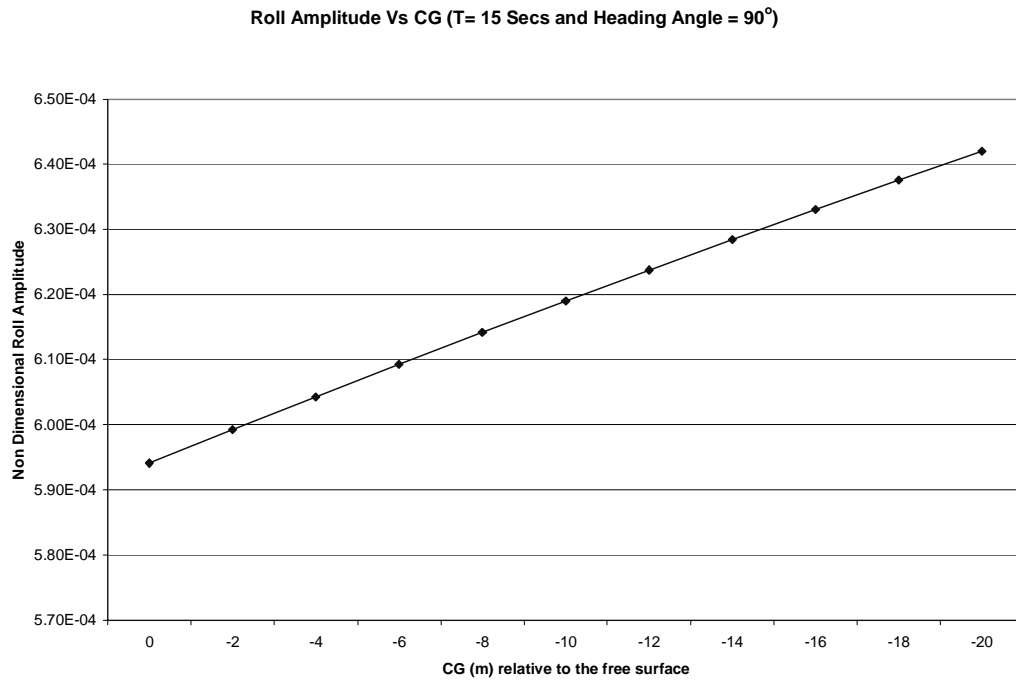


Figure 4.11 Roll Vs CG

The roll amplitude also varies with the center of gravity of the floating body. In the Figure 4.11 it can be observed from the graph, that the roll amplitude increases as the center of gravity drops from the free surface to the base of the hull.

Chapter 5

Architecture of the Simulation and rendering program

This simulation tool heavily relies upon the values generated by WAMIT for predicting the motions of the hull and the database of values is used in close context with the hull motions. The rendering of the wave environment is completely independent of the motions and can be generated in parallel to motion prediction.

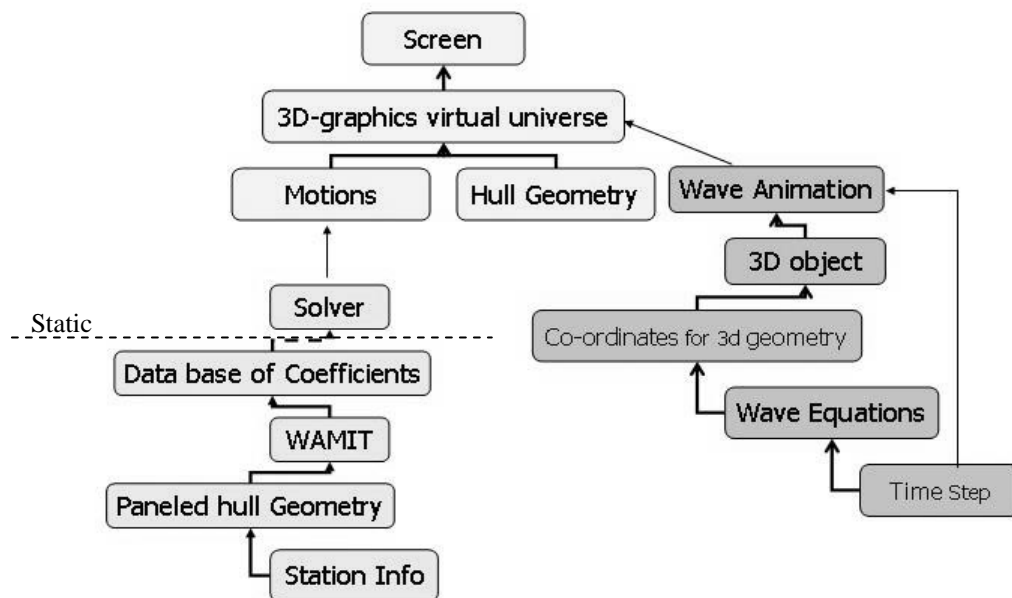


Figure 5.1 Overall Architecture

Figure 5.1 shows the architectural diagram of the simulation system, part of which is static and requires only a one time operation and the rest being dynamic.

Database Creation

Using the ship geometry information from the AutoCAD files, the station vertices are extracted and those are used to panel the geometry. WAMIT is run on the hull for a range of time periods and heading angles and output files were created. To create the directory structure from the output files, an automated set of codes is used to create the directories and also the files containing the coefficients in them. The database is created for values $T = 1$ to 20 seconds with an interval of 1 second and $\theta = 0^\circ$ to 180° with an interval of 2° . The whole process of creating this database for a given hull geometry takes around 30 minutes on a 1 GHz Microsoft windows based computer.

Wave environment generation

The rendering of the wave environment has two major modules

- a. Geometry creation.
- b. Wave Animation.
- c. Hull motion animation

As described earlier, wave geometry is created using the wave equation (Equation 4.2). This equation is plugged into the dual line strip set superposition system discussed earlier. Using the xyz coordinates generated by the nested loops an indexed Line array primitive is created and grid pattern is generated. The whole algorithm is implemented in a single class. Using the Shape3D class of Java3D the Line Arrays are given a structure by the rendering pipeline. When more than one wave is present in the wave environment, the y value is computed for each of those waves and summed together to get the final wave geometry and

this is computed for every frame/time step. The simulation may slow down if too many waves are added to the wave environment and it depends on the speed of the computer on which it runs. When too many waves are added, the time required to compute displacement for the components increases and thereby slowing down the animation. The wave geometry has to be available before the next frame appears onscreen. This geometry array is passed on to the next level which is the wave animator.

Wave Animation

The wave animation is event driven and the events are time dependent and generated in definite time steps. The animator is implemented using the Behavior node of Java3D, and the animator class inherits the behavior class and implements the methods of the behavior class to customize it for this application. The animation system is triggered in equal time steps and rendered on screen. The behavior class implements a separate thread and it is completely parallel with other operations and hence the animation is made smooth irrespective of other back ground operations. Thus the animation technique described earlier is implemented.

Solver Module

This module solves the six complex simultaneous equations by retrieving the coefficients and forces from the database. The solver constructs the six by six left hand side which constitutes thirty six complex numbers in the matrix and the right hand side is the exciting force for each degree of freedom. This system of

complex equations are solved using a numeric package called Java Numeric Library which provides classes for solving complex matrices.

Following is the modified motion equation which incorporates the phase of the wave in it:

$$\sum_{j=1}^6 \zeta_i \left[\frac{-\omega^2}{g} (M_{ij} + a_{ij}) + i \frac{\omega^2}{g} b_{ij} + c_{ij} \right] = X_i e^{i\phi} \quad (5-1)$$

where

ϕ = phase of the incident wave

The phase of the wave is introduced to make sure the simulation is accurate in terms of the offset introduced in the hull's motion due to the phase involved with the wave itself, and the amplitude A is made unity to generalize for any amplitude. LU factorization, Gaussian elimination method is employed to solve the six equations. The resulting is the response amplitude operator.



Figure 5.2 Comparison of RAO(Roll) computed by WAMIT and Solver module

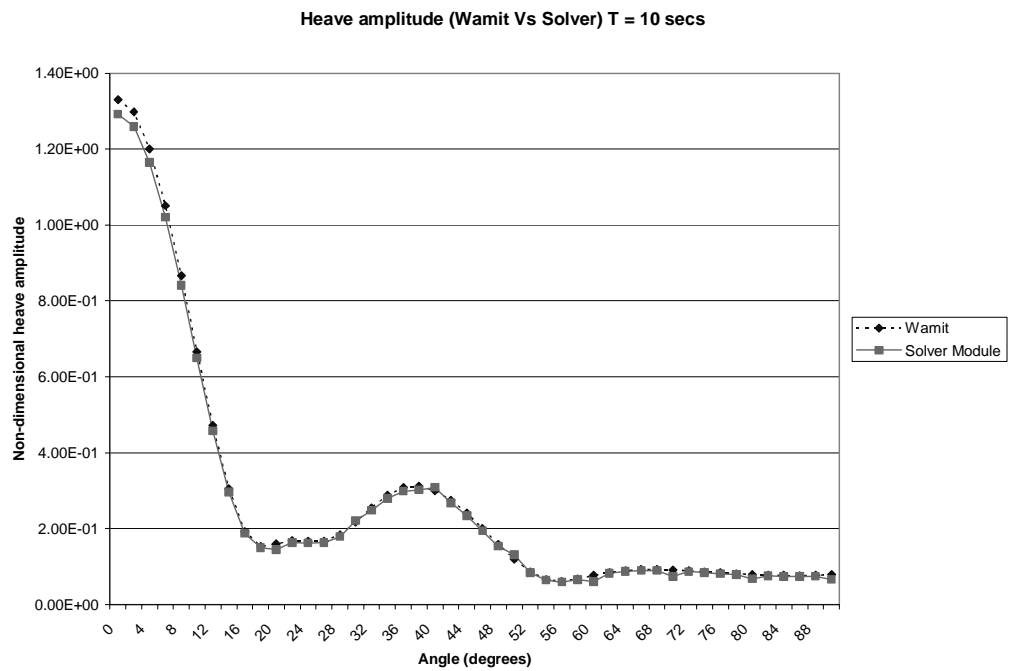


Figure 5.3 Comparison of RAO(Heave) computed by WAMIT and Solver module

The comparison between the RAO values generated by WAMIT and by the solver module is shown in figure 5.2 & 5.3. It can be observed that the solution obtained through the solver module closely matches with that of WAMIT's.

Interpolation

When the user tries to add a wave whose parameters are such that, it does not have an entry in the database, example: when the user enters a value in between two consecutive entries say $T = 12.34$ & $\theta = 43.2^\circ$ and the nearest possible values available are $T = 12$ and $T = 13$, $\theta = 42^\circ$ and 44° , since the change in values is small between the intervals, the intermediate coefficients are obtained by linearly interpolation.

$$\text{Value}_{\text{inp}} = \text{Value}_1 + (\text{Value}_2 - \text{Value}_1) \frac{(T - T_1)}{(T_2 - T_1)} \quad (5-2)$$

Where

$\text{Value}_{\text{inp}}$ = Interpolated value.

T = time period of Incident Wave

$T_1 < T < T_2$ (T_1 and T_2 are values present in the database)

Value_1 and value_2 are values corresponding to T_1 and T_2

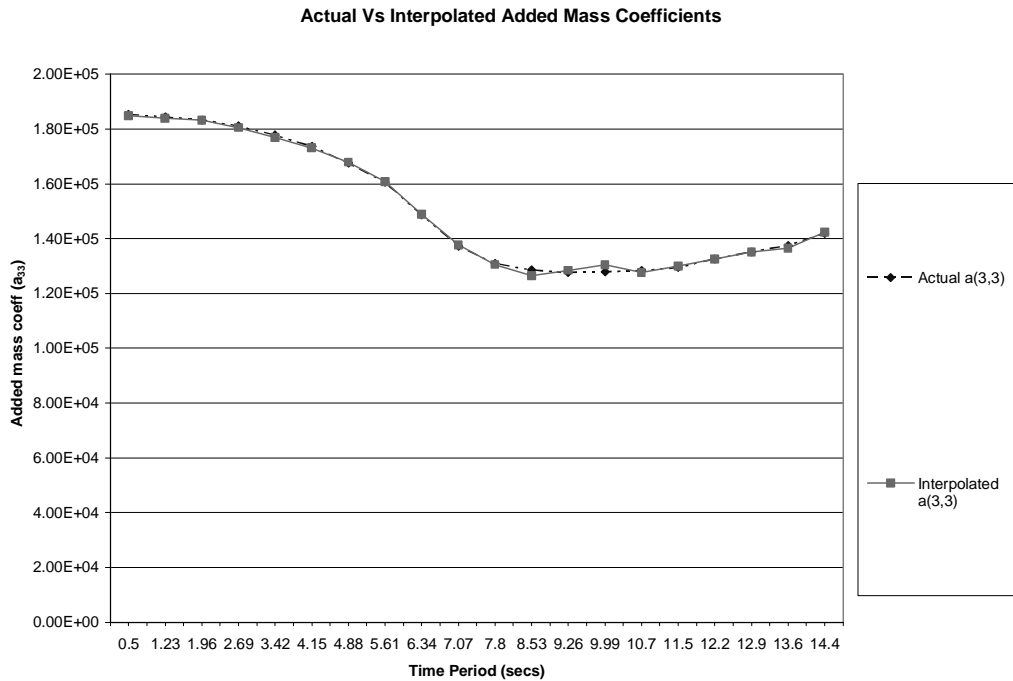


Figure 5.4 Comparison of Actual added mass coefficient(a_{33}) Vs Interpolated value

The interpolated values are used for the coefficients and the forces in the simultaneous equations; this makes it possible for the simulation tool to handle any wavelength and heading angle.

Figure 5.4 shows a graph drawn between the actual value of the added mass coefficient (a_{33}) computed by WAMIT and the interpolated value. The values appear to match each other very well.

The accuracy of the interpolation scheme can be increased by using smaller intervals between time periods and the angles for which WAMIT generates the values, though this would increase the size of the database but maybe it is necessary in cases where the numerical values are more important

than onscreen rendering. Furthermore the accuracy of interpolated values can be improved by introducing spline interpolation schemes rather than linear form, though that is not attempted in this project it may constitute future work.

Hull Motion Animation

The hull motion animation is implemented by using a behavior node and thus manipulating the hull 3D model. The motions computed using the solver module is used to rotate and translate the hull on screen. Like the wave animator node this node is also time triggered and the operations are carried out at equal intervals of time.

The velocity equation is:

$$U_j(t) = A \operatorname{Re}(i\omega\xi_j e^{i\omega t}) \quad (5-3)$$

The six response amplitude operators ξ_j are computed by the solver module and it has both magnitude and phase. The amplitude of the incident wave (A) makes the motion amplitudes dimensional. The equation 5.3 determines the position and orientation of the hull every time step and this is used by the animation node to render the hull on each frame. When more than one wave is added to the wave environment the motion of the hull is resultant motion caused by all the waves together. This is achieved by looping through all the motions caused by the waves and summing them up and this has to be done for every frame.

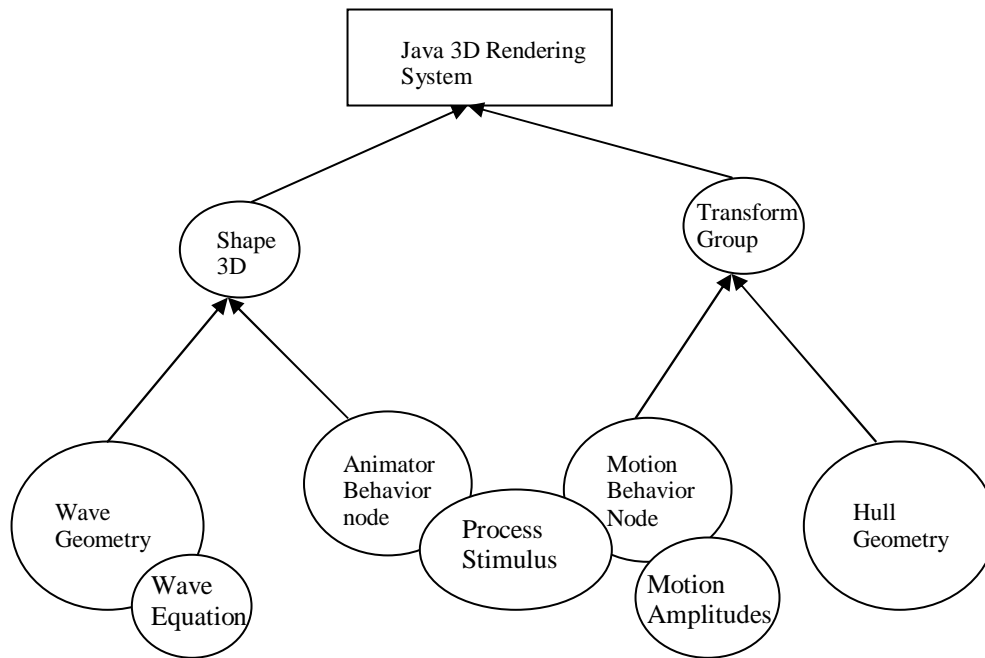


Figure 5.5 Animation Implementation

Figure 5.5 shows the animation implementation for both the waves and the hull motions and also shows how the behavior nodes relate to the geometries.

Behavior nodes

The behavior node is a fairly new concept in animation and it is implemented only in Java 3D though other APIs may have similar concepts. The Behavior leaf node allows an application to manipulate a scene graph at run time. Behavior is an abstract class that defines properties common to all Behavior objects in Java 3D. There are several predefined behaviors that are subclasses of Behavior. Additionally, a Behavior leaf node may be sub-classed by the user. The animation nodes are sub classes of the behavior class and provides customized even driven animation for the simulation tool.

Applet and user interface

The simulation tool is implemented as an applet; hence it can be integrated into web pages easily. One of the advantages of Java3D is that it provides clear-cut integration of 3D graphics and user interfaces.

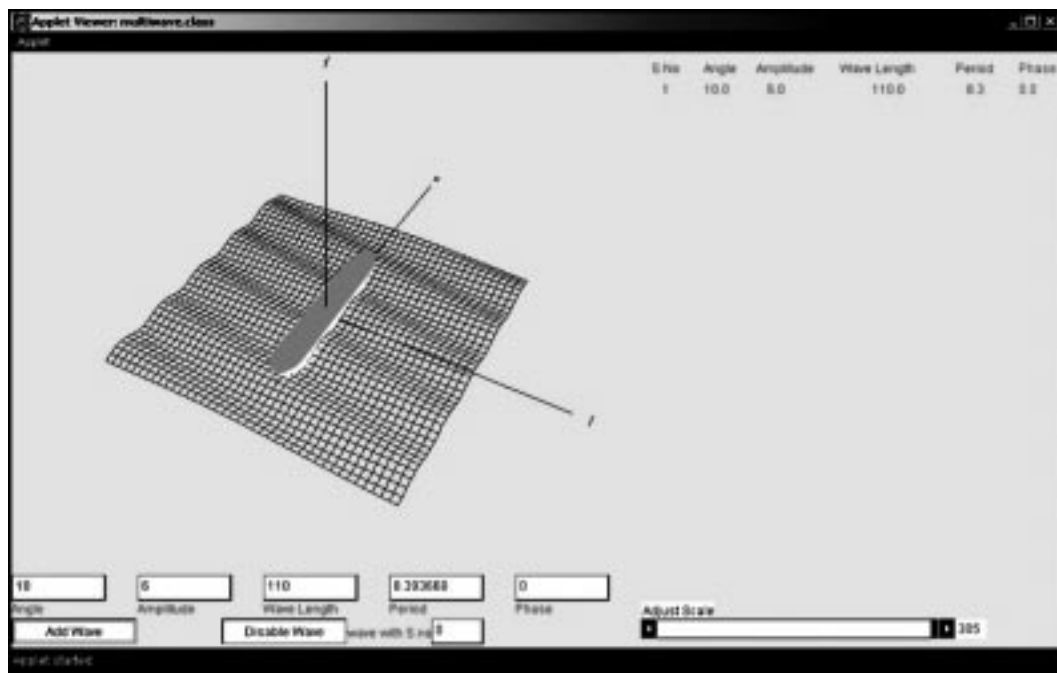


Figure 5.6 Screen shot showing monochromatic waves.

The wave parameters of the mono chromatic wave environment shown in Figure 5.6 are as follows:

$$\theta = 10^\circ, A = 6\text{m}, \lambda = 110\text{m}, T = 8.383\text{ secs}, \phi = 0^\circ$$

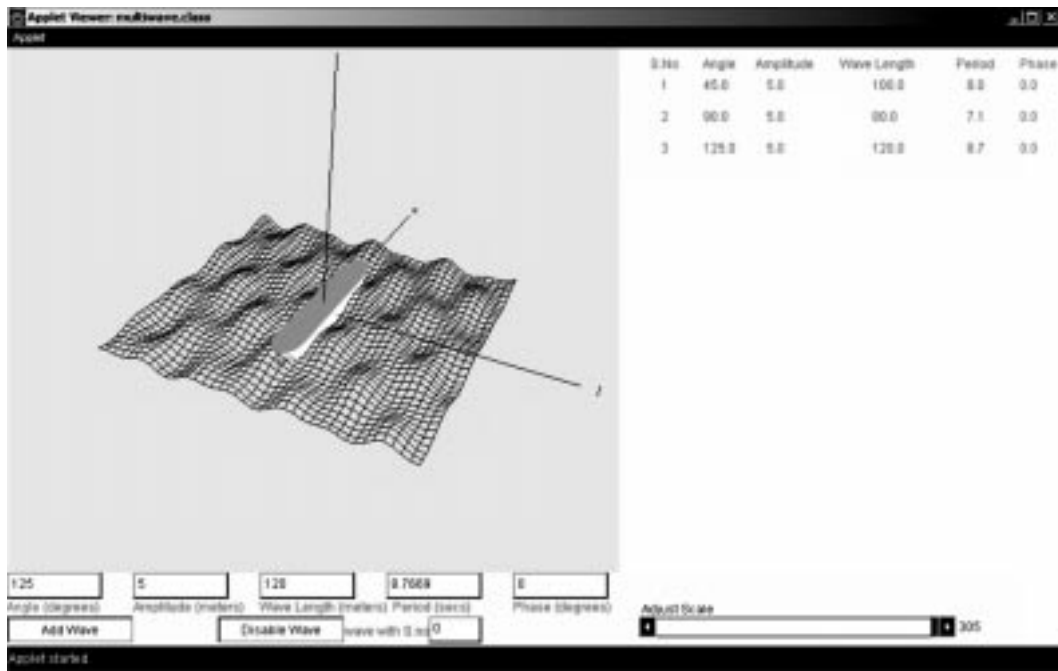


Figure 5.7 Screen shot showing multi-component waves

The wave parameters of the multi-component wave environment shown in Figure 5.7 are as follows:

$$\theta = 45^\circ, A = 5\text{m}, \lambda = 100\text{m}, T = 8.0\text{secs}, \phi = 0^\circ$$

$$\theta = 90^\circ, A = 5\text{m}, \lambda = 90\text{m}, T = 7.1\text{secs}, \phi = 0^\circ$$

$$\theta = 125^\circ, A = 5\text{m}, \lambda = 120\text{m}, T = 8.76\text{secs}, \phi = 0^\circ$$

Both these figures show rendering of the wave environments with different wave parameters and how distinctively different they are and it can also be observed how the graphic user interface (GUI) blends with the 3D rendering screen. The user interface provides functionality to add waves to the wave environment with the specified wave parameters and the scale of the hull geometry can also be adjusted using the slider control. The rendered wave environment and the hull can

be rotated along any axis using the mouse pointer; this provides for viewing from any angle and hence better visualization. The wave parameters of the component waves are displayed as a table on the right hand side of the screen. Any existing wave in the environment can be disabled using the “Disable wave” button.

Java3D is an excellent tool to create interactive 3D graphics applications. What has been implemented in this simulation tool is only basic. Java3D has many more capabilities and elaborate applications can be developed using it.

Chapter 6

Conclusion

This project has been successful in bringing complex simulations and visualization to an average desktop PC. The performance of the 3d graphics system surpassed expectations and smooth animation of wave environments was achieved. This tool can prove to be an effective educational instrument for teaching concepts in hydrodynamics and also help researchers to understand better the hydrodynamic responses of hulls and floating bodies in general. The surface generation technique described in this thesis has proven to be a very effective and complete way of animating complex geometries in real time. Though with a compromise on surface appearance, the complete information of floating bodies and waves is visualized. The event triggered animation methodology made feasible by Java 3D made the animation process simple and easy to implement.

This simulation tool is an example of how 3D graphics can be exploited for visual simulations and how such tools can be made prevalent by making it web based. This kind of technology is slowly making inroads in to all fields of science and engineering, thanks to innovative programming models like Java3D.

FUTURE WORK

The simulation tool can be modified to work with multiple numbers of hull geometries by creating separate databases of each one of them and letting the

user choose the hull of his/her preference. This can be accomplished by very little modification to the existing implementation.

A buffering scheme can be developed to provide smooth rendering in case the grid size is increased or if the simulation gets complex. Buffering of frames would greatly enhance the performances and much detailed rendering can be achieved.

A probability density distribution for wave heights can be used to simulate the wave environment instead of using individual components. This would need modifications in the wave geometry creation module.

The hydrodynamics backbone of the system, namely WAMIT may not be the only software package that can be used with this tool, with certain modifications WAMIT can be replaced by any package capable of estimating hydrodynamic characteristics of a floating body.

This simulation tool is made available to users in the Electronic Classroom on Ocean Wave Theory website. This website has detailed information on wave theory and other related topics and can be accessed online at this URL: <http://otrc93.ce.utexas.edu/~waveroom/>. This particular simulation tool can be found under the interactive tools section.

It is hoped that this simulation tool would serve its purpose and made use of to the fullest extent.

Bibliography

- [1] <http://java.sun.com/products/java-media/3D>
- [2] <http://www.j3d.org>
- [3] OpenGL reference manual: the official reference document to OpenGL, version 1.2 / OpenGL Architecture Review Board ; editor, Dave Shreiner; 2000.
- [4] J.N.Newman, 1997: Marine Hydrodynamics The MIT press Cambridge, Massachusetts and London, England
- [5] WAMIT user manual ver.6.0; WAMIT inc. MA USA:2000
- [6] <http://usa.autodesk.com/adsk/item/0,,752569-123112,00.html>
- [7] Atsumi Imamiya and Dong Zhang;1997:Geometric Model and Visualization of Breaking waves.
- [8] <http://www.webreference.com/3d>
- [9] <http://www.righthemisphere.com>

Vita

Aravind Kumar Hariramasamy was born in Chennai India on June 28th 1979, He is the son of Hariramasamy Sankarapandian and Ramalakshmi Solaiappan. He graduated with a bachelors degree in Instrumentation and control Engineering from University of Madras (India) in May 2000 and continued on to pursue a M.S in Electrical and Computer Engineering degree at The University of Texas at Austin and graduated in May 2002.

Permanent address: No 45 3rd Main Road Apt # 3A
Gandhi Nagar, Adyar
Chennai, 600020 India

This thesis was typed by the author.