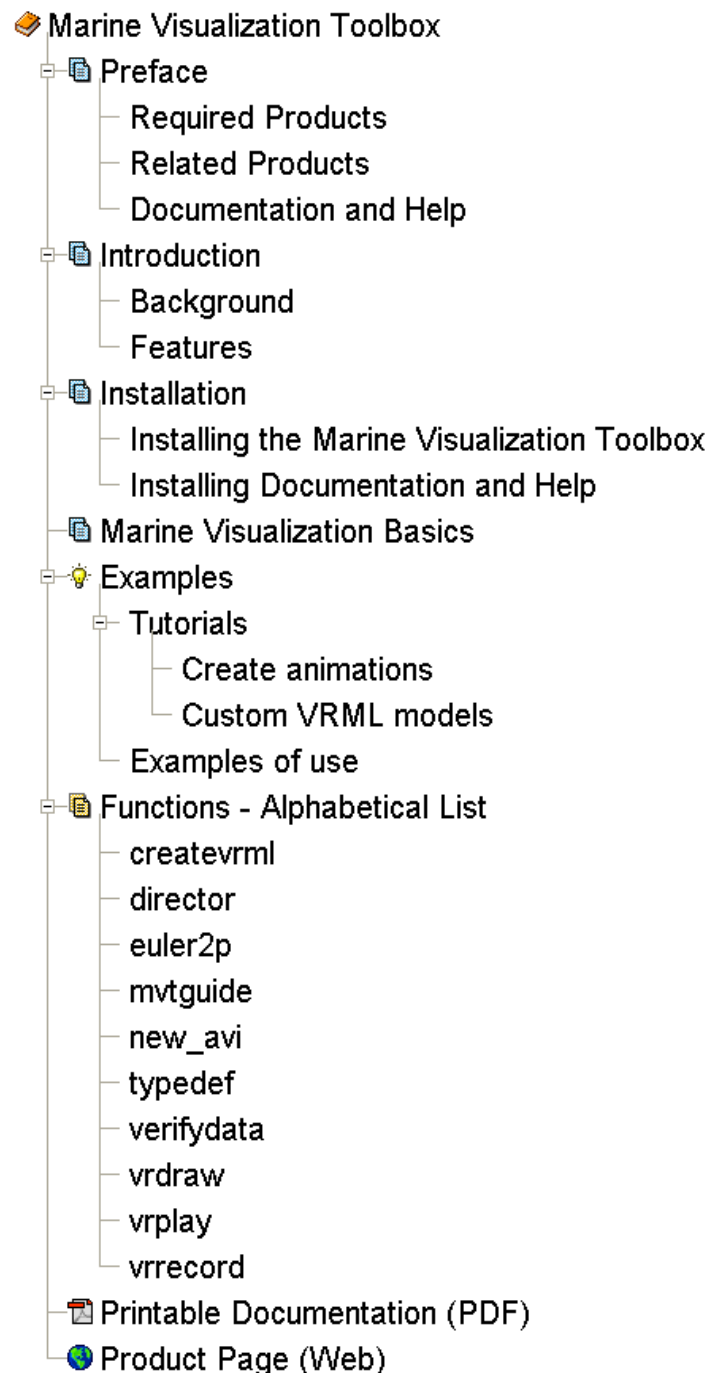


This document is a PDF-version of the help files for the Marine Visualization Toolbox, table of contents is seen in the figure below (screen shot from the Matlab® help browser).



- Marine Visualization Toolbox
  - Preface
    - Required Products
    - Related Products
    - Documentation and Help
  - Introduction
    - Background
    - Features
  - Installation
    - Installing the Marine Visualization Toolbox
    - Installing Documentation and Help
  - Marine Visualization Basics
  - Examples
    - Tutorials
      - Create animations
      - Custom VRML models
    - Examples of use
  - Functions - Alphabetical List
    - createvrml
    - director
    - euler2p
    - mvtguide
    - new\_avi
    - typedef
    - verifydata
    - vrdraw
    - vrplay
    - vrrecord
  - Printable Documentation (PDF)
  - Product Page (Web)

## ● **Learning About the Marine Visualization Toolbox.**

- [What is it?](#) - Features for the Marine Visualization Toolbox.
- [Introduction](#) - Procedures for using the Marine Visualization Toolbox with MATLAB.

## ● **Web Pages**

[Product Page](#) - Key features and product requirements.

## ● **Finding Functions**

[Functions - Alphabetical List](#)

## ● **Printing the Documentation**

See the [Virtual Reality Toolbox User's Guide](#) and [Marine Visualization Toolbox Release Article](#) in PDF format.

## ● **Required and Related Products**

- [Required products](#) - Required products to run the Marine Visualization Toolbox.
- [Related products](#) - Relevant products to the Marine Visualization Toolbox

## Preface

See the following pages for information about the Marine Visualization Toolbox

- [Required Products](#) Products that are required to run the toolbox
- [Related Products](#) Products that are especially relevant when using the toolbox
- [Documentation and Help](#) Overview of documentation and help functions

The following apply to all text and functions in the Marine Visualization Toolbox:  
All occurrences of MATLAB and Matlab<sup>®</sup> refer to Matlab<sup>®</sup> by MathWorks, Inc.



## Required Products

The Marine Visualization Toolbox requires the following products to be installed:

- [Virtual Reality Toolbox](#) Provides a MATLAB interface for interacting with virtual worlds. See the MathWorks web site, at: <http://www.mathworks.com> for more information.



## Related Products

For more information about any of these products, see either

- The online documentation for that product if it is installed
- The MathWorks Web site, at <http://www.mathworks.com>; see the "products" section

Product	Description
<a href="#">Virtual Reality Toolbox</a>	Create and manipulate virtual reality worlds from within MATLAB and SIMULINK
<a href="#">GNC Toolbox</a>	Guidance, Navigation and Control Toolbox for Marine Applications
<a href="#">Simulink</a>	Design and simulate continuous- and discrete-time systems



## Documentation and Help

Links to tutorials, examples, documentation and help on specific functions:

- [Tutorials](#) to get you started
- [Examples](#) show different ways of using the toolbox
- [Documentation](#) in printable PDF format is found on the product page
- [Help on functions](#)



## Introduction to the Marine Visualization Toolbox

Description of what the Marine Visualization Toolbox can, and cannot, do.

- [What is](#) the Marine Visualization Toolbox? Description and introduction.
- Specific [features](#) of the toolbox

## What is the Marine Visualization Toolbox?

Several computer tools for development and analysis of marine control systems have been introduced the last few years (Fossen, 2002; Perez, 2003). These concern mathematical modelling, control and analysis of multiple variable systems. Results from simulations, experiments and measurements of such systems are usually presented as functions of time in one or more figures.

The presented toolbox helps displaying data from multivariable systems in MATLAB, utilizing the Virtual Reality Toolbox 3.0. The Virtual Reality Toolbox provides a MATLAB interface for viewing 3D models (Mathworks, 2002). The Marine Visualization Toolbox (MVT) uses this interface to animate 3D vessel models, moving and rotating the vessels according to the time varying input data.

The input data may represent a maximum of six degrees of freedom (6 DOF), xyz-positions and Euler angles. Any number of vessels may be animated in the same scene.

The vessels are animated in a virtual world representing the actual surroundings of the simulation, experiment or full scale measurement (eg., a coast line, a specific basin, etc.).

## Preferences

Fossen, T. I. (2002). *Marine Control Systems, Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles*. Marine Cybernetics.

Perez, T. and Blanke M. (2003). DCMV a MATLAB/-Simulink® Toolbox for Dynamics and Control of Marine Vehicles. In: ???.

Mathworks Inc. (2002). Virtual Reality Toolbox; using version 3.0.



## Features

The Marine Visualization Toolbox present the following features:

- View 3D animations of vessel in 6 DOF
- Save animations as MATLAB independent movie files
- 3D model library with some standard off-shore vessels
- 3D model library of virtual worlds, typical surroundings for marine operations or experiments
- Help on how to use the toolbox, compatible with the MATLAB help browser
- Tutorials to get you started



## Installation

Follow the step by step instructions to install:

- the [toolbox](#)
- the toolbox [documentation and help](#)

## Installing the toolbox

To use the Marine Visualization Toolbox with MATLAB make sure to perform the following steps:

- **Create a new folder for the toolbox.**  
The MVT files must be located in a folder named **mvt**, this folder must be located in the MATLAB toolbox folder, eg. *C:\MATLAB6p5\toolbox\mvt*. Use your preferred file browser or the MATLAB file browser to create this folder.
- **Copy toolbox files to folder.**  
Copy all MVT files to the new folder.
- **Add MVT folder to MATLAB search path.**  
Use MATLAB command *addpath* to update the MATLAB search path, or use *Set Path* in the *File* menu. More information about MATLAB [search path](#).

If extracting MVT from a compressed file, select the MATLAB toolbox path as destination folder. Files are then copied into the correct folder.

MATLAB command *rehash toolbox* forces MATLAB to refresh it's search path, run this command after copying or editing files in a toolbox folder.

## Installing Documentation and Help

MVT help are written as HTML pages and may be viewed in any web browser, but intergration with the [MATLAB help browser](#) is highly recommended. Perform the following steps to include help on MVT functions in your MATLAB help browser:

- **Create a new folder for the MVT help files.**  
Create a new folder for help files in the MATLAB help folder, eg.  
*C:\MATLAB6p5\help\mvt.*
- **Copy help files to folder.**  
Copy the MVT help files to the new folder.
- **Copy the index file to MATLAB help folder.**  
Copy the file named *mwdoctoc\_visualtbx.xml* to the MATLAB help folder, typically *C:\MATLAB6p5\help* . This file is read by the MATLAB help browser, and will display the MVT help table-of-contents in your [help navigator](#). The XML file is located with your MVT help files, or may be downloaded from the [product web page](#)

Note that links to other MATLAB functions will fail if the MVT help files are *not* located in the MATLAB help folder.

## Marine Visualization Basics

Basic knowledge needed to use the Marine Visualization Toolbox:

- The [coordinate system](#), understanding directions and rotations
- Use of [structures and arrays](#) with MATLAB and MVT
- [6 DOF data](#), how movement of marine vessels are described
- [3D models](#) and VRML files in MVT

### The coordinate system

The coordinate systems used in MVT follows the convention from (SNAME, 1950). Figure 1 illustrates the positive directions of the inertial frame and the rotation axis of the body fixed vessel frame.

The vessels are translated according to the inertial frame's origo, and the vessels' body fixed frame are rotated about their local origo. Translation inputs to MVT are assumed to be the time varying  $x$ ,  $y$  and  $z$ -positions relative to the inertial frame origo, and rotations are assumed to be Euler rotations:  $\phi$ ,  $\theta$  and  $\psi$ , see example in figure 2. Note that the axis directions of a *non rotated* body fixed frame is parallel to the inertial frame.

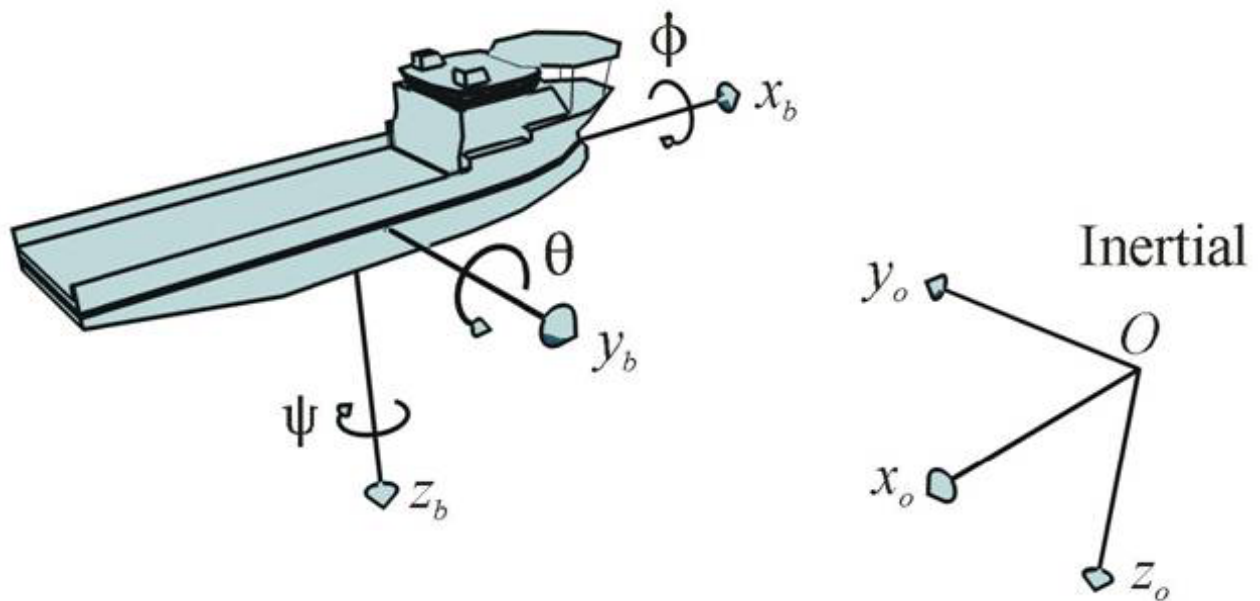
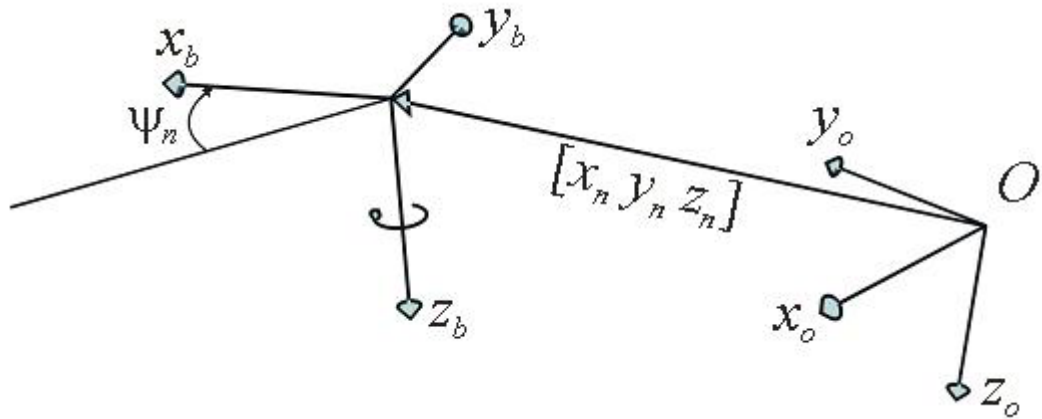


Figure 1. Inertial and body fixed frames.

### 6 DOF data

The six input variables described above are often referred to as the six degrees of freedom (Fossen, 2002). In MVT such a set of time varying positions and rotations will be referred to as 6 DOF data. The 6 DOF data in MATLAB are implemented as double precision [arrays](#), with dimensions  $N$  rows by 6 columns, where  $N$  is the number of samples. Each column contains the values of  $x(n)$ ,  $y(n)$ ,  $z(n)$ ,  $\phi(n)$ ,  $\theta(n)$  and  $\psi(n)$ , where sample number  $n = 1$ ,



2, ..., N.

Figure 2. Body fixed frame translated  $[x_n \ y_n \ z_n]$  and rotated  $\psi_n$  about the  $z$  axis.

## Structures and arrays in MATLAB and MVT

In MVT arrays are used as input to and output from functions. MATLAB arrays are analog to mathematical matrices and may be of any dimension, MVT only uses one and two dimensional arrays, a one dimensional array equals a vector.

The [6 DOF data](#) array has dimensions  $N \times 6$ , the following command examples show how to manipulate and retrieve information from certain columns, rows or the whole array:

Create an  $10 \times 6$  array of zeros:

```
my_array = zeros(10,6)
```

Using the colon operator to address specific columns and rows. First command sets each value in the first column equal to their row number. Second command retrieves the fourth row from the newly created array:

```
my_array(:,1) = [1:10]';
third_row = my_array(4,:);
third_row = my_array(4,:);
```

The 6 DOF data array now contain zeros for all vessel variables, except for  $x$ -position. From this data we know that the vessel moves straight forward (since  $\psi_n=0$  for all  $n$ ) from  $x=1$  to  $x=10$ . We now have to define a time vector ( $N \times 1$  array, a column vector) which identifies the time of each sample in *my\_array*. The following command creates a linear time vector of the same length as the 6 DOF data, with time steps of 0.1 seconds:

```
time_vector = 0.1 * [1:10]'
```

The `'` operator transposes the row vector to a column vector (see help on `:`). The *time\_vector* now contain sample time information for the 6 DOF data in *my\_array*. Before you pass these two arrays to a MVT function assign them to a structure (MATLAB *struct*) by typing:

```
vessel_struct.x = my_array
vessel_struct.t = time_vector
```

The structure `vessel_struct` can now be used as input argument to MVT functions that require 6 DOF data input. And the fields of `vessel_struct` `x` and `t` may be used as ordinary arrays, for example:

```
vessel_struct.x(:,2) = ones(10,1)
```

To create a array of structures (in case you have more than one vessel) simply more append entries to the existing structure. Imagine there's a vessel moving from `x=10` to `x=28` simultaneously as the first vessel moves from `x=1` to `x=10`:

```
my_second_array = zeros(10,6)
my_second_array(:,1) = [10:2:28]
my_second_array = zeros(10,6)
vessel_struct(2).t = time_vector
```

Once 6 DOF data for all your vessels are assigned to one structure, it might be a good idea to save (see MATLAB *save*) the structure for future use, retrieve saved structures with *load*.

## VRML 3D models in MVT

Two components are needed to display a 3D model in MVT. A VRML file describing the properties of the 3D model and a VRML viewer to draw the 3D model onto the screen. Short description of terms in this chapter:

- **VRML** (Virtual Reality Modelling Language) is combination of a 3D modelling language and a network communication language (Carey and Bell, 1997). VRML features are simple geometric shapes and extensive user multi user functionality. MVT combines several geomtric shapes to model complex 3D models. Figure 3 shows a 3D VRML model displayed by MVT.
- A **3D model** refer to any three dimensional model, such as an off-shore supply vessel, a coast line etc. A VRML 3D model refer to a 3D model written in this specific modelling language.
- The **VRML viewer** displays the 3D model, MVT uses the viewer supplied with the [Virtual Reality Toolbox](#) to do this. A VRML editor is also supplied with the toolbox, use this editor to create your own 3D models (see *Examples*).
- All **interactions** with the displayed VRML model, such as moving and rotating a vessel, utilize the functions of the [Virtual Reality Toolbox](#).

## References

Carey, R. and Bell, G. (1997). *The Annotated VRML 2.0 Reference Manual*. Addison-Wesley Developers Press.

Fossen, T. I. (2002). *Marine Control Systems, Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles*. Marine Cybernetics.

SNAME (1950). Nomenclature for treating the motion of a submerged body through a fluid. Technical Report Bulletin 1-5, Society of Naval Architects and Marine Engineers, New York, USA.







## Tutorials

- [Tutorial#1](#) - Learn how to create animations.
- [Tutorial#2](#) - Learn how to use custom VRML models with the toolbox.

## Tutorial#1

This tutorial is a step-by-step instruction to create an animation with the Marine Visualization Toolbox. Before starting make sure the MVT toolbox is correctly installed: While in your *work* folder, type *help mvt*. This should display the contents of the MVT, if not see the [installation instructions](#).

---

### Step 1 - Setting up the 6 DOF data

It is assumed that you are already familiar with 6 DOF data, if not see [6 DOF data](#) in [Marine Visualization basics](#). Operating on structures and arrays are described [here](#).

In MVT the 6 DOF data is represented by a two dimensional matrix, with N row and 6 columns (Nx6), where N is the total number of data samples. Each row holds the values of the 6 vessel states in the following order: [x y z phi theta psi] (Equal to: [x y z roll pitch yaw]). All angles are given in radians.

#### Step 1:

Load the saved data from *tut1-01.mat* (Hint: Highlight the text below, right click and select *Evaluate Selection*):

```
filename = fullfile(matlabroot, 'toolbox', 'mvt', 'tutorials', 'tut1-01.mat');  
load(filename);
```

This command loads the 6 DOF data saved in *tut1-01.mat* into your MATLAB workspace as a variable named *sixDOF*. To see the contents of the array, type:

```
sixDOF
```

To plot the 6 DOF data, for instance the *xy*-positions, type something like:

```
plot(sixDOF(:,1), sixDOF(:,2), 'r-')
```

This will plot a red line in the *xy*-plane, recall that *x* and *y* is found in the first and second column.

---

### Step 2 - Setting up the time vector

Since animations describe movement over time, the relative time of each sample (each row in the 6 DOF data) must be identified. This is done in a column vector, with dimensions  $N \times 1$ . Each entry,  $n = 1 \dots N$ , contain the time value for it's respective samples (rows in sixDOF). The time vector may have varying time steps (the time between subsequent time values), but must be strictly increasing.

## Step 2:

```
filename = fullfile(matlabroot, 'toolbox', 'mvt', 'tutorials', 'tut1-02.mat');  
load(filename);
```

This loads the time vector *time\_vec* to your MATLAB workspace, to see it's contents, type:

```
time_vec
```

The column vector *time\_vec* now contains the time values (in seconds) for the samples in sixDOF. Note that the length of *time\_vec* equals the length of *sixDOF*.

To see how any of the variables in the 6 DOF data vary over time, for instance the *x*-position (*yaw* or *psi*), type:

```
plot(time_vec, sixDOF(:,1), 'r-');
```

This will plot the *x*-position versus time. This line is drawn from one sample point to another, if you want to see the sample points, type:

```
hold on;  
plot(time_vec, sixDOF(:,1), 'bo');
```

These commands first tell MATLAB not to erase the old plot, then to plot the sample points as blue circles.

---

## Step 3 - Setting up the vessel structure

Most function arguments in MVT are based on [structures](#), a structure can hold several values of different classes (e.g. double and char). Structures are created using the *.*-operator (the *dot*-operator), structure arrays are created simply by indexing the structure name, see example below. Variables contained in a structure is called a *field*, see *struct* for more information on structures.

Structures may be named anything for use with MVT, but the *fields* must apply to the following convention:

- The 6 DOF data must be in a field named *x*.
- The time vector must be in a field named *t*.

### Step 3:

The 6 DOF data and time vector representing one particular vessel must be organized in a structure. In this example we name this structure *vesselData*. To insert the 6 DOF data *sixDOF* and the time vector *time\_vec* into *vesselData*, type:

```
vesselData.x = sixDOF;  
vesselData.t = time_vec;
```

To view the contents of *vesselData*, type:

```
vesselData
```

This command will describe the contents of *vesselData*.

To retrieve, or edit, a specific field in the structure, use the dot-operator, for instance:

```
vesselData.x
```

These commands will print the contents of *vesselData.x*.

---

## Step 4 - Starting *mvtguide*

The function *mvtguide* calls other functions of MVT in the following order:

- *verifydata* - Verify input user data
- *typedef* - Link the verified user data to vessel 3D models
- *createvrml* - Assemble the linked 3D models into a single file
- *director* - Graphical user interface where animations finally are created

If you are an experienced MATLAB user these functions may be called manually (see their help pages for input-output arguments), as *mvtguide* is only a help function to keep track of input and output arguments.

### Step 4:

Call *mvtguide* with the newly created structure as input argument:

```
mvtguide(vesselData)
```

This command calls the *mvtguide* function, which in turn prompts the user for input of various kinds, see following steps.

---

## Step 5 - Define type of vessel

By executing the steps above you should now see a dialog box named *Settings*, this is where you define the type of vessels in your animation, and what scene the vessels will be animated in. In this tutorial you should select the *Cybership* vessel model and *basin* scene model, but any selections are valid (the input 6 DOF data will probably not match any other vessel model).

### Step 5:

Select the vessel model named *Cybership* in the list: *Selected vessel is linked to...*

Select the scene model named *basin* in the popupmenu: *Select scene:*

Confirm your choices by pressing the *OK* button. The vessel is now linked to a 3D model, and a 3D scene is selected as the surroundings for the animation. These are passed to the *createvrmf* function.

---

## Step 6 - Saving the VRML file

You are now prompted for a filename by *createvrmf*, the requested filename is where the vessel model and scene model are assembled into one single file. You should specify a filename that makes it easy to identify later, in this tutorial you should name it *my\_tutorial.vrmf*, otherwise the tutorial might not complete.

### Step 6:

Enter the filename *my\_tutorial.vrmf* and press *OK*.

---

## Step 7 - Setting up a sequence

Two new windows should now appear on your screen, and look like the two figures below. If not, check the error message in your MATLAB workspace. See help on [director](#) for detailed information on fields and settings.

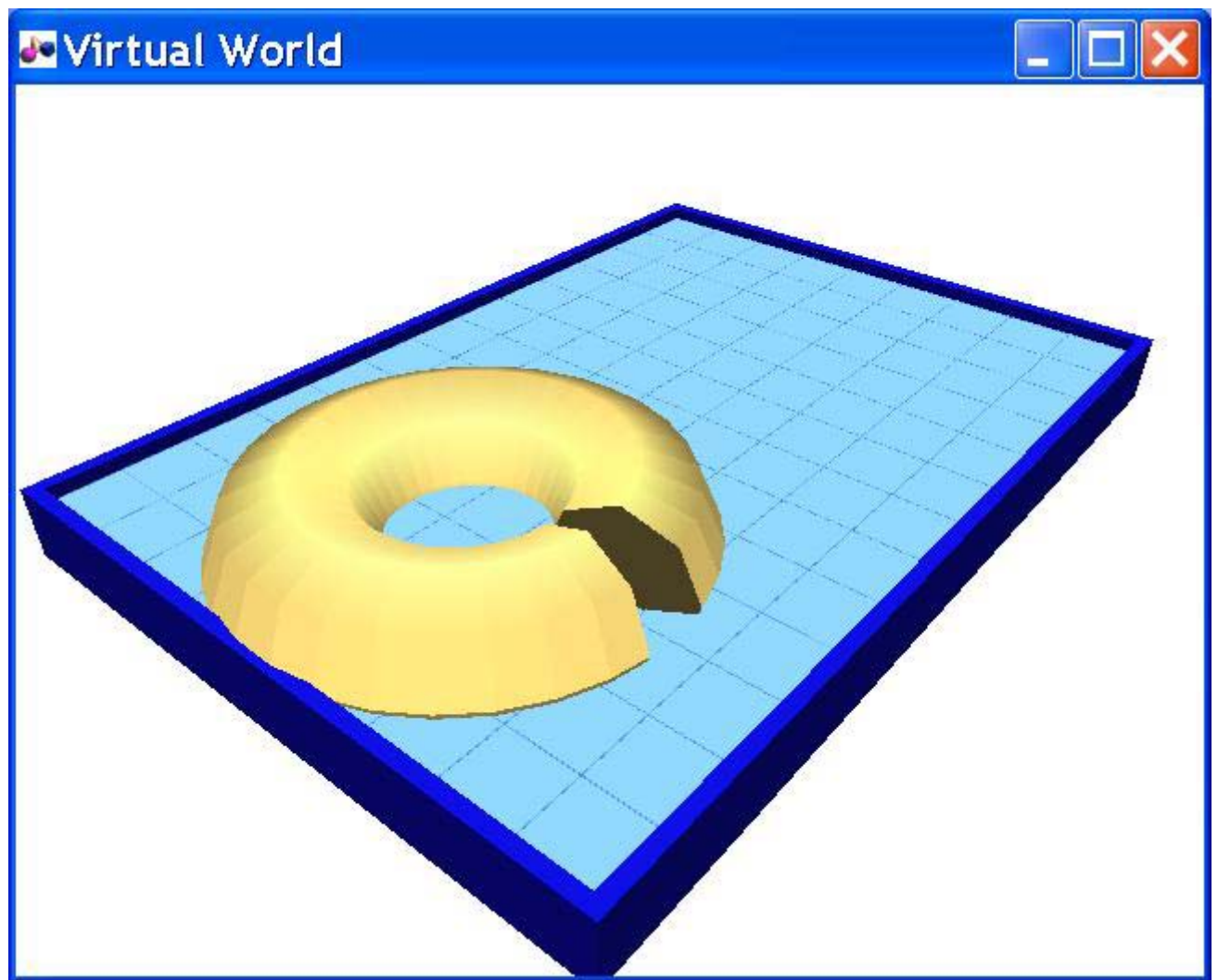


Figure of the virtual reality figure window.

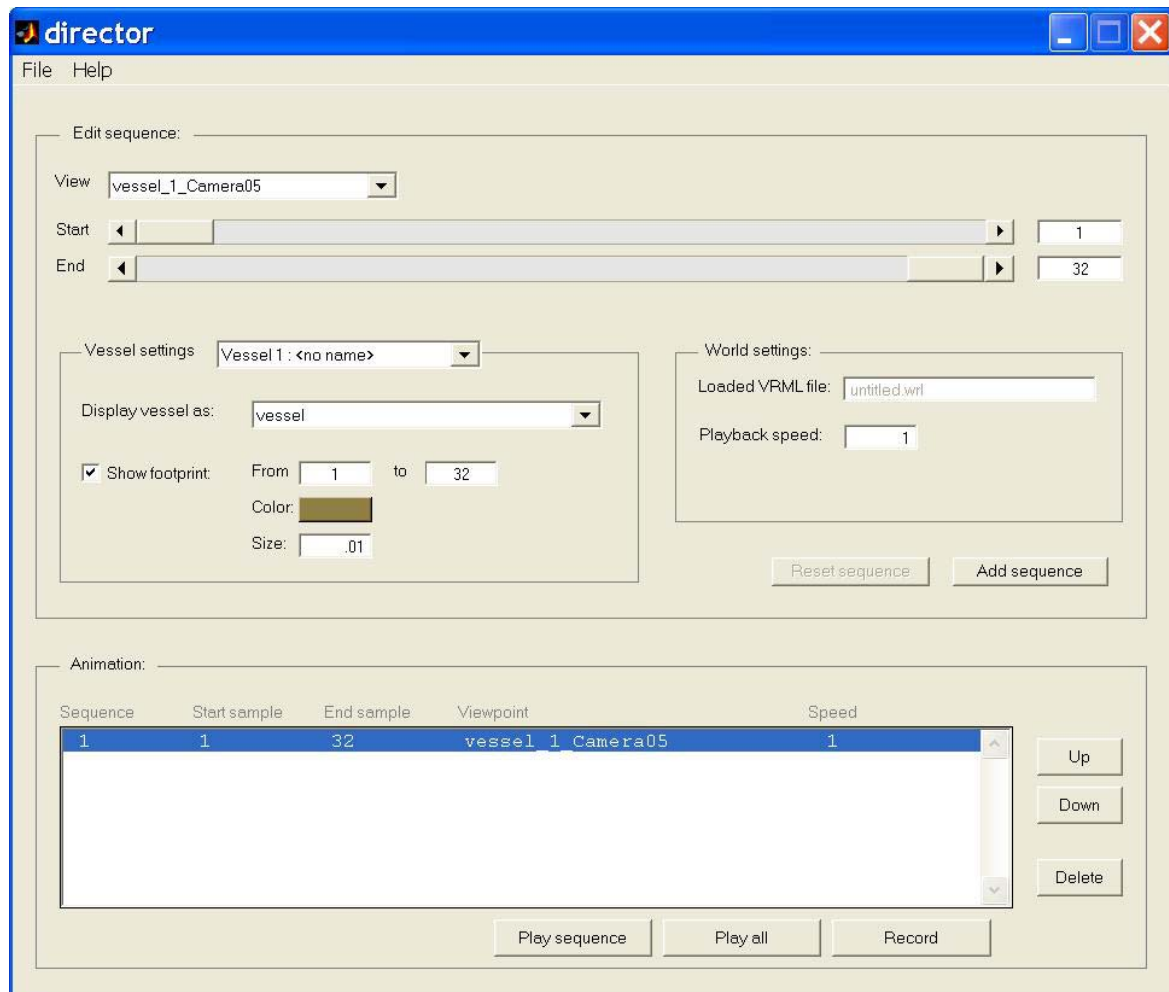


Figure of the *director* graphical user interface

The huge donut displayed in the virtual world is the path of the vessel, to decrease its size enter a smaller value in the *Path Size* field. The pool is the selected scene model *basin*.

### Step 7:

To decrease the size of the vessel path, enter a smaller value in the *Path Size* field, for instance 0.01. You should now be able to see the vessel *Cybership*, which is positioned according to the first sample in the 6 DOF data.

Select the desired viewpoint from the viewpoint list, viewpoints with prefix *vessel* move with the specified vessel, while all other viewpoints stay fixed to the scene model.

The selected viewpoint, start sample, end sample, Vessel settings and World settings define a sequence, press the *Add sequence* button to add the sequence to the total animation. All sequences are listed in the animation listbox, which now should look something like the above figure.

Press *Play sequence* to preview the sequence.

Press *Record* to save the animation to file, you are prompted for video settings. The default values should produce a satisfactory result in most cases, but remember to change the filename

to avoid overwriting old video-files. Select or enter the filename *my\_tutorial.avi*. Recording will start, and a bar shows the recording progress. Make sure that the figure window is not minimized or completely covered by other windows while recording, this will ruin the video-file!

---

## Step 8 - View the saved animation

When recording is finished, locate your newly created video-file (which was specified just before the recording started) in the file browser, and play the animation!

---

To get to know the functionality of *director*, experiment by changing the settings, change the viewpoint, record different sample intervals, change the appearance of your vessel, record several sequences into one animation etc.





## Tutorial#2

This tutorial is an instruction on how to use custom VRML models with the MVT, it is *not* an introduction to 3D modelling.

The VRML editor used in this tutorial is the editor included in the Virtual Reality Toolbox 3.0: V-Realm Builder, Version 2.0 by Integrated Data Systems, Inc. The program is located in the toolbox subfolder *vr*, under *vrealm*.

Files included for this tutorial are located in the MVT help folder, under *tutorials*, typically: *[your\_matlabroot]\help\toolbox\mvt\tutorials*. Always search this folder when asked to open a file during this tutorial.

---

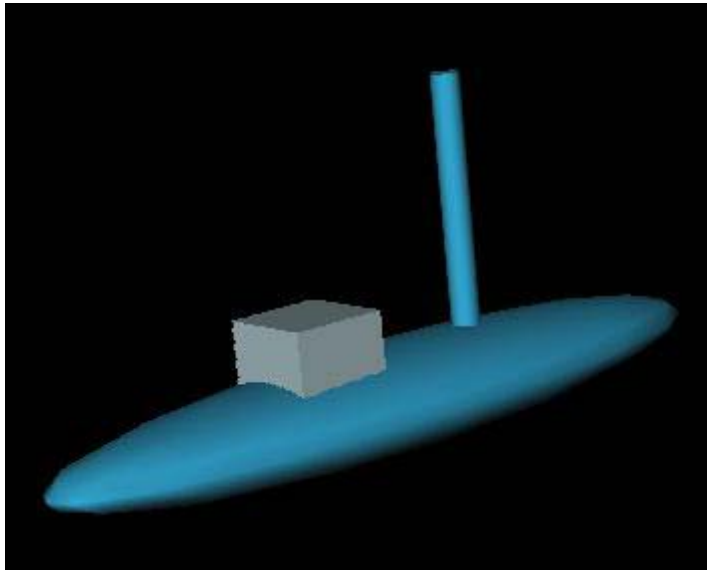
### Step 1 - Alternative vessel models

All vessels used with the MVT must have two levels of VRML files. The first level may be several VRML models, which represent the alternative ways of displaying a vessel (see the *Display vessel as...* field in the director GUI). Level two integrates all these files in one single file, using the VRML command *Inline*.

This tutorial includes two different VRML models, a vessel and the axis it's body-fixed frame. These have the file extension *inline*, this is to avoid confusion with the VRML files with extension *wrl* when selecting vessel models in the *typedef* GUI.

#### Step 1:

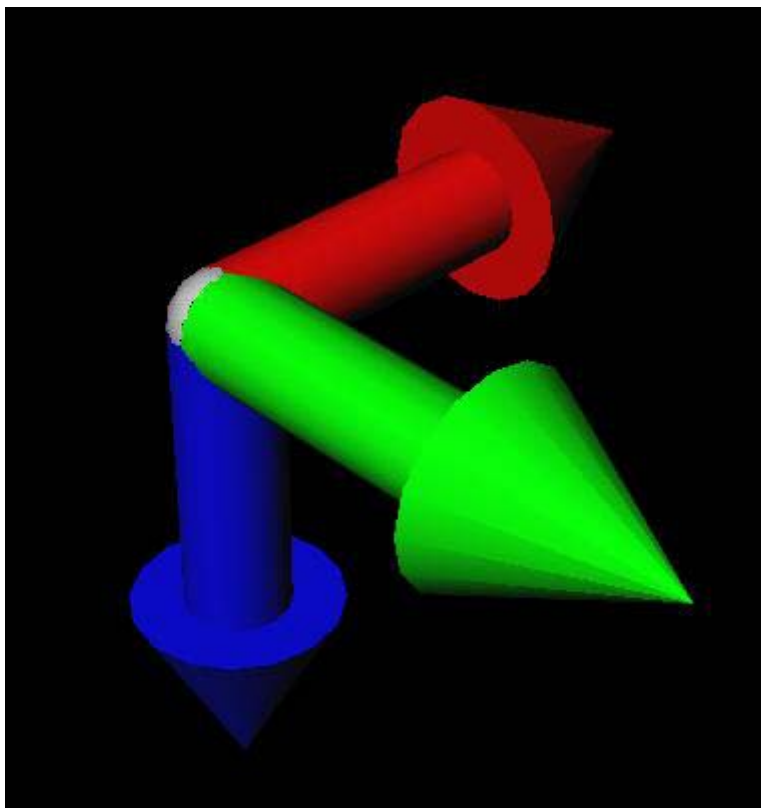
Open the file: *my\_inlined\_vessel.inline* in your VRML editor. By rotating the object a little bit, it should something like the figure below.



The VRML model in *my\_inlined\_vessel.inline*.

This model will be one of the two possible models to display the custom vessel in an animation.

The other model is the vessel's body-fixed axis, open the file *my\_inlined\_axis.inline* to view this model. It should look something like the figure below, the red, green and blue arrows are the  $x$ ,  $y$  and  $z$ -directions respectively.



The vessel's body-fixed axis.

You now have two models that may represent the vessel in an animation, but you may have any number of alternative models for one vessel.

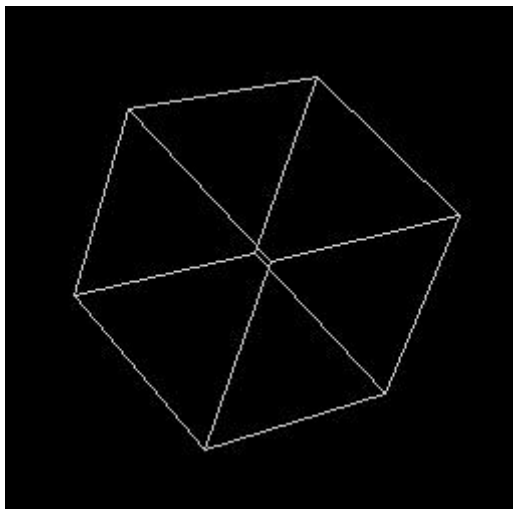
---

## Step 2 - The vessel VRML file

All vessel VRML files have file extensions *wrl* (e.g. *auv.wrl*, *supply.wrl*), and they all reference files with extension *inline* (e.g. *auv.inline*, *axis.inline*). You should follow this naming convention, to avoid confusion between inlined files and vessel files, when creating your own custom vessels.

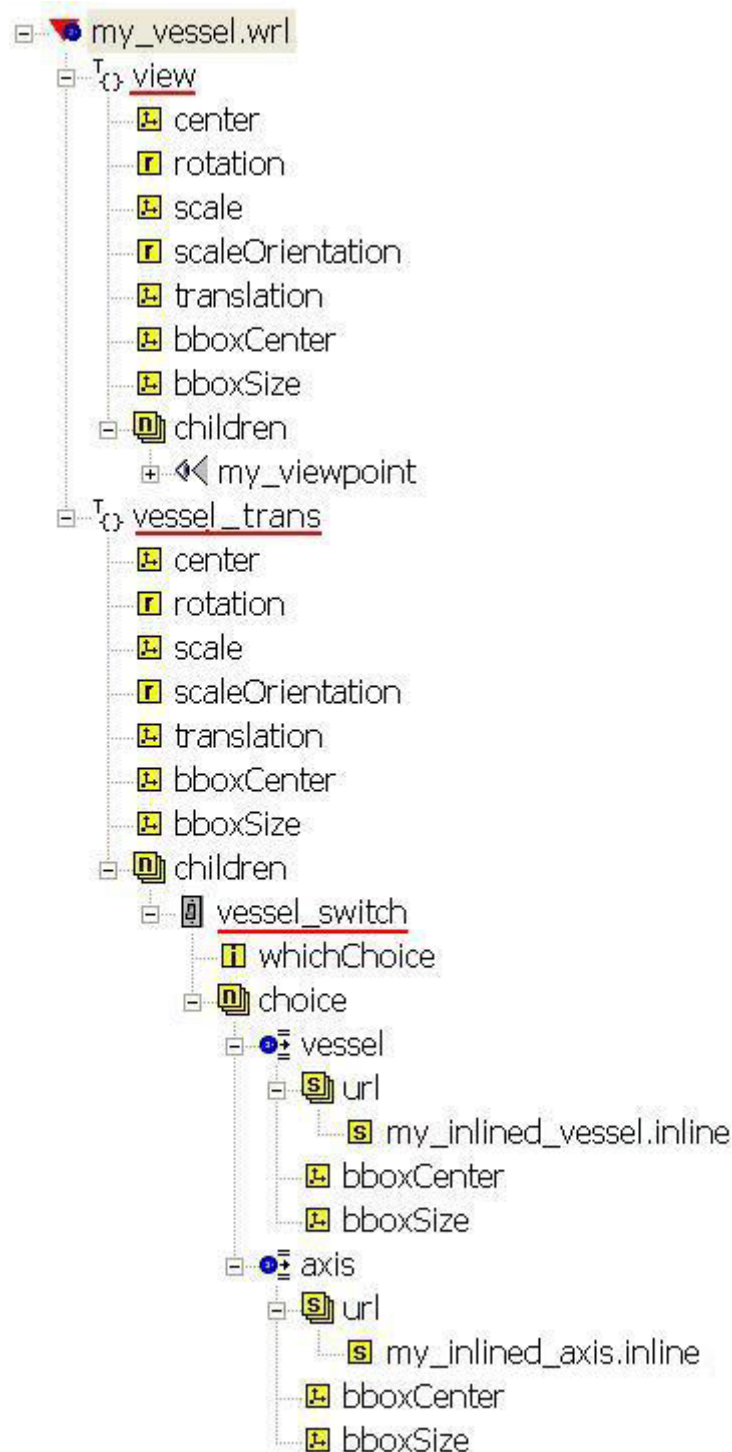
### Step 2:

Open the file *my\_vessel.wrl* in the VRML editor. This file might not appear to contain anything at all, but you should be able to see a wireframe box centered on the screen, like in the figure below.



A bounding box illustrating the position of the inlined VRML file.

Since the only models in this file are referenced by *inline*, the editor only displays a bounding box to give you a general idea of the model's position. If you're using V-Realm Builder, you should see a hierarchy tree to the left, like the one below except for the red underlining.



The hierarchy of the VRML file *my\_vessel.wrl*.

The two model files from previous step are referenced as *inlines*, these are both children of the *vessel\_switch* (underlined red). If you want to add other alternative models, add them under the *vessel\_switch* as inlined files.

The translation and rotation of *vessel\_trans* are manipulated directly by the MVT, these are the fields that move your vessel! Entering values in these fields have no effect, since MVT overwrites them when creating animations.

The given names with red underlines must be spelled in this exact manner to be recognized by the MVT. All others may be named anything, but note that the names of the *inline* fields (*vessel* and *axis*), will appear in the popupmenu *Display vessel as...* in the *director* GUI.

When animating the translation field of *view* will be set equal to the translation field of *vessel\_trans*. Thus the viewpoints that are children of *view* will follow the vessel as it moves, you may define any number of viewpoints.

---

## Step 3 - Copy the VRML files to the MVT folder

In order to use the above VRML files with the MVT you have to copy the files to the MVT subfolder *vessel*, this is where all vessel models must be located.

### Step 3:

Copy the three files from step 1 and 2 to your MVT subfolder *vessel*, typically:  
*[your\_matlabroot]\toolbox\mvt\vessel*.

The VRML files are now recognized by the *typedef* GUI and displayed as any other vessels the next time you run the toolbox.

---

## Step 4 - Custom scene models

To use custom scene models with the MVT, simply create your own scene model, and save it to the MVT subfolder *world*, typically: *[your\_matlabroot]\toolbox\mvt\world*. You may include any number of viewpoints in a scene model.

## Examples of use

The following examples describe how to visualize some typical scenarios in marine operations. None of the examples are worked through, but the short descriptions should give you an idea of how to complete the visualization.

Syntax, listed expressions occur on this page, recall their meaning by clicking them.

- [6 DOF data](#)
- Vessel structure [1](#) [2](#)

---

### A single vessel in 6 DOF

The vessel structure should now consist of a field  $x$  and a field  $t$ , the time-varying 6 DOF vessel states and time vector respectively.

Run `mvtguide` with this structure as input argument, select the desired vessel type, scene model and the desired VRML filename when prompted. The default settings will now display your selected vessel and it's [path](#). If all you see is the vessel's path, reduce it's size, if you don't see the path, increase it's size.

See also: [tutorial#1](#).

---

### Visualizing vessels with less than 6 DOF

Data from less than 6 DOF are easily expanded to be compatible with MVT. For instance, a vessel movement where only the  $xy$ -positions and heading are measured is a movement in 3 DOF. The 3 DOF data is converted to 6 DOF data by adding a row of zeros for each vessel state that was not measured. This equals that the  $z$ -position, roll and pitch are all measured to zero. See figure below: Assume that the  $x$ ,  $y$ , and  $yaw$  variables contain the measured states for 6 different samples. The `zeros(6,3)` command (underlined red) creates a zero-matrix with 6 rows and 3 columns, these represent the vessel states that are not measured ( $z$ , roll and pitch).

```
>> measured_3DOF = [x y yaw]
```

```
measured_3DOF =
```

0	1.0000	0
0.0998	0.9950	0.1000
0.1987	0.9801	0.2000
0.2955	0.9553	0.3000
0.3894	0.9211	0.4000
0.4794	0.8776	0.5000

```
>> constructed_6DOF = [x y zeros(6,3) yaw]
```

```
constructed_6DOF =
```

		<i>z</i>	<i>roll</i>	<i>pitch</i>	
0	1.0000	0	0	0	0
0.0998	0.9950	0	0	0	0.1000
0.1987	0.9801	0	0	0	0.2000
0.2955	0.9553	0	0	0	0.3000
0.3894	0.9211	0	0	0	0.4000
0.4794	0.8776	0	0	0	0.5000

The general formula is thus to insert a row of zeros (see *zeros*) for every vessel state that is not already defined. The time vector should not be chaged.

---

### A vessel tracking a reference vessel

To do this you need the 6 DOF data of both the actual vessel and the reference vessel.

Set up a structure with the fields *x* and *t*. And assign the 6 DOF data and time-vector for the vessel in these. When animating more than one vessel, the vessel structure must be a *structure array*. This is simply created by indexing the structure. To add your reference vessel to the animation simply index the structure as number 2, and assign it's 6 DOF data and time vector. Note that the time vectors must be of the same length, but not necessarily with the same time samples. The following example creates a structure array with two sets of vessel data:

```
% add vessel to structure 'data'
data.x = vessel_6DOF_data;      % vessel_6DOF_data is a Nx6 double array
data.t = vessel_time_vec;       % vessel_time_vec as a Nx1 column vector
% add reference vessel to structure 'data'
```

```
data(2).x = ref_6DOF_data;    % ref_6DOF_data is a Nx6 double array  
data(2).t = ref_time_vec;    % ref_time_vec as a Nx1 column vector
```

◀ Tutorial#2

Functions ▶





## Functions - Alphabetical list

Alphabetical list of functions in MVT:

- [createvrml](#)
- [director](#)
- [euler2p](#)
- [mvtguide](#)
- [new\\_avi](#)
- [typedef](#)
- [verifydata](#)
- [vrdraw](#)
- [vrplay](#)
- [vrrecord](#)



## vrplay

**VRPLAY** Play VRML animations for the Visualization Toolbox.

VRPLAY(SEQUENCE, VRFIG, VRWORLD, DATA) interpolates the desired sample range and time step (specified in SEQUENCE) of the data values given in DATA. The interpolated data modifies corresponding nodes in VRWORLD and VRFIG.

See also VRFIGURE, VRWORLD, VRNODE, [DIRECTOR](#), [CREATEVRML](#).

Author: Andreas Lund Danielsen

Date: 10th November 2003

Revisions:



## vrdraw

**VRDRAW** Updates the current VR-figure.

VRDRAW(VRWORLD, SEQUENCE, DATA, T) updates the VR-figure associated with VRWORLD. Vessel states are retrieved from sample T in DATA. Object properties are retrieved from the settings in SEQUENCE. Called by DIRECTOR in the Visualization Toolbox.

See also [DIRECTOR](#), [CREATEVRML](#).

Author: Andreas Lund Danielsen

Date: 10th November 2003

Revisions:



## verifydata

**VERIFYDATA** Verify input data for use with the Visualization Toolbox.

[FLAG] = VERIFYDATA(DATA) analyzes the structure of DATA. DATA must be of class struct and the following requirements applies to the fields:

**Field name:**

t  
x  
type

**Field class:**

[nx1] double vector  
[nx6] double array  
char array

Note: The fields 'type' and 'name' are optional, VERIFYDATA will return FLAG = TRUE if they don't exist. If they do exist they must be of class char array, otherwise FLAG = FALSE is returned.

See also STRUCT, [CREATEVRML](#), [TYPEDEF](#).

Author: Andreas Lund Danielsen

Date: 5th November 2003

Revisions:

## typedef

**TYPEDEF** Link user data to available vessel models.

[NEWDATA, WORLD] = TYPEDEF(DATA) displays an interface where vessels found in DATA may be linked to specific 3d vessel models. These links are returned in NEWDATA for use with CREATEVRML.

A search for all VRML files (\*.wrl) in the folder: 'toolbox\visual\vessel' returns a list of available vessel models. Each vessel in DATA then may be linked to any listed vessel model.

The required structure of DATA is described and verified by VERIFYDATA. Returned structure NEWDATA equals DATA, but contains the updated links in field 'type'. Returned char array WORLD is the filename of the scenario file used by CREATEVRML.

Note: If the field 'type' is already defined for all vessels, and the desired scenario file is known, it's not necessary to run TYPEDEF!

See toolbox documentation on how to use your own vessel models.

See also: [CREATEVRML](#), [VERIFYDATA](#), [DIRECTOR](#).

Author: Andreas LD

Date: 6th November 2003

Revisions:



## new\_avi

**NEW\_AVI** Open new movie AVI file for recording.

[FILENAME, X, Y, QUALITY, COMP, FPS] = NEW\_AVI returns the selected settings for the new AVI file FILENAME.

- X and Y, resolution X horizontal by Y vertical, in pixels
- QUALITY, compression quality, 65-100%
- COMP, method of video compression
- FPS, Frames per second

Default settings works well under Windows, Unix users should change compression method. High QUALITY, large resolution and more FPS implies larger files and takes longer time to record (An animation of 1 minute length at default settings, generates a movie file of about 20 Mbyte).

See also: [DIRECTOR](#).

Author: Andreas Lund Danielsen

Date: 10th November 2003

Revisions:



## mvtguide

**MVTGUIDE** Runs the Marine Visualization Toolbox step-by-step.

**MVTGUIDE(DATA)** Runs the Marine Visualization Toolbox by calling MVT functions in the following order:

- 1: `verifydata` - verifies the input DATA
- 2: `typedef` - displays a graphical user interface where the vessels identified in DATA are linked to 3D vessel models and a scene model
- 3: `createvrml` - assembles the linked 3D files to a single file
- 4: `director` - displays a graphical user interface where animations are created and viewed/saved

DATA: Structure of 6 DOF data arrays and type vectors, see `VERIFYDATA` for required fields and dimensions.

Users comfortable with MVT and MATLAB are encouraged to make custom m-file functions that automatically convert their data to the correct format and run the files listed above.

See also [VERIFYDATA](#), [TYPEDEF](#), [CREATEVRML](#), [DIRECTOR](#),

Author: Andreas Lund Danielsen

Date: 20th November 2003

Revisions:



## euler2p

**EULER2P** Converts an Euler rotation to a principal rotation.

[BETA, EPSILON] = EULER2P(PTP) converts an euler rotation PTP = [PHI THETA PSI] about the xyz-axis to a principal rotation BETA about the vector EPSILON.

Author: Andreas Lund Danielsen

Date: 27th October 2003

Revisions:



## director

**DIRECTOR** Tool for creating 3d animations.

DIRECTOR(DATA, VRMLFILE) displays the VR world specified by VRMLFILE and opens a graphical user interface. By using TYPEDEF and CREATEVRML the properties of all vessels in VRMLFILE are ensured to be compatible with DATA.

Animation sequences are created by specifying time intervals, viewpoint and time varying vessel properties (translation, rotation and appearance). Animations may be viewed in the installed VRML viewer, or recorded as AVI files for Matlab independent playback.

See also: [CREATEVRML](#), [VERIFYDATA](#), [TYPEDEF](#).

Author: Andreas Lund Danielsen

Date: 10th November 2003

Revisions:

---

**Detailed information** of fields and buttons in the *director* GUI:

- [Animations and sequences](#)
- [Viewpoints](#)
- [Timeline](#)
- [Selecting vessels](#)
- [Display vessel as...](#)
- [Paths](#)
- [World settings](#)
- [Play sequence](#)
- [Play all](#)
- [Record](#)

---

## Animations and sequences

Animations consist of sequences, and each sequence consists of a time limited set of properties. Thus, all properties within the *Edit sequence* frame may be changed from one sequence to another.

To add a sequence to your animation, press the pushbutton *Add*. The sequence is then added to the list of sequences at the bottom of the screen. Change the sequences individual orders with the *Up* and *Down* buttons, or remove a sequence with *Delete*.

---

## Viewpoints

A UI element for selecting a viewpoint. It consists of a label 'View' followed by a text box containing 'Camera03' and a small downward-pointing arrow icon on the right side.

All viewpoints identified with the loaded VRML file are listed in this popupmenu. By selecting a viewpoint the view changes in the displayed virtual world.

Note that viewpoints beginning with *vessel* follow the corresponding vessel's movements, other viewpoints are fixed to the earth-fixed frame.

---

## Timeline

A UI element for a timeline. It features two horizontal sliders. The top slider is labeled 'Start' and has a value of '1' displayed in a box at its right end. The bottom slider is labeled 'End' and has a value of '485' displayed in a box at its right end. Both sliders have small arrowheads at their ends for adjustment.

The timeline sliders enables you to scroll through the input 6 DOF data. Note that the displayed value are *sample* numbers, not *time* values! These values determine for which time interval the sequence is animated.

You may enter the desired sample number in the edit fields at the right end of the sliders.

---

## Selecting vessels

A UI element for selecting a vessel. It consists of a label 'Vessel settings' followed by a text box containing 'Vessel 1 : <no name>' and a small downward-pointing arrow icon on the right side.

To change the sequence properties for a vessel, select a vessel in the popupmenu then set the properties you would like. These properties will take affect immidiatley in the displayed world. After adding the sequence to the animation, the properties will be saved with that specific sequence, and take affect whenever it is played back or recorded.

Changes in vessel properties are saved automatically, even if you select another vessel.

---

## Display vessels as...

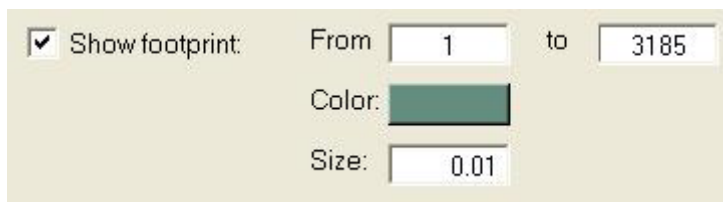
A UI element for selecting how vessels are displayed. It consists of a label 'Display vessel as:' followed by a text box containing 'vessel' and a small downward-pointing arrow icon on the right side.

Vessels may be displayed as other 3D models than it's vessel type, for instance as it's body fixed axis. Or the vessel may not displayed at all, not displaying all vessels can direct the focus to other vessels of particular interest.

If two vessels are moving close to each other, for instance a reference vessel and a simulated vessel, displaying them as body-fixed axis would give an indication of tracking quality.

---

## Paths



☒ Show footprint: From  to   
Color:   
Size:

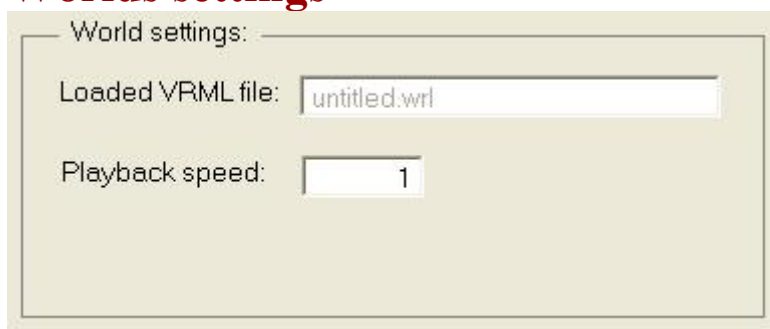
Paths are lines stretching from the selected vessel's first position, through all samples, to the vessel's last sampled position. Line thickness is set by entering a number in *Size* (given in meters, one centimeter in the above example).

The fields *From* and *to* determine for what sample range the path is visible. In the above example, the path is visible from first sample to sample number 3185. In other words: while the vessel is animated within this range, it will move along a line.

Instead of entering constant samples, you may enter  $t$ . Entering  $t$  in the *From* field will produce a path in front of the vessel, *from* the vessel *to* some constant sample. Entering  $t$  in the *to* field, will produce a path behind the vessel, *from* some constant sample *to* the vessel.

---

## Worlds settings



World settings:  
Loaded VRML file:   
Playback speed:

*Playback speed* is the ratio of animation time versus sample time. Recall that sample time is given by the values in the time vector.

Imagine a time vector containing time values reaching from 0 seconds to 10 seconds (Note that the length of the time vector is irrelevant!), and a corresponding set of 6 DOF data.

- A playback speed value of 1, means the data will produce a 10 seconds long animation
- A playback speed value of 0.5, will produce a 20 seconds long animation
- A playback speed value of 2, will produce a 5 seconds long animation



## **Play sequence**

*Play sequence* plays back the selected sequence from the animation list.

## **Play all**

*Play all* plays back all the sequences from the animation list.

## **Record**

*Record* starts recording all the sequences to file. You are asked to specify some video options before the recording starts.



## createvrml

**CREATEVRML** Creates a VRML-file for use with the Visualization toolbox.

FLAG = CREATEVRML(DATA, WORLD, TARGETFILE) creates a VRML-file TARGETFILE based on the information in DATA (see VERIFYDATA for help on DATA). The VR- world is created from the VRML file WORLD, which must be located in the Visualization Toolbox subfolder 'world'.

If the field 'type' is not specified in DATA, 'type' is set equal to the first entry in type list.

See also [TYPEDEF](#), [VERIFYDATA](#), [DIRECTOR](#).

Author: Andreas Lund Danielsen

Date: 7th November 2003

Revisions:



## vrrecord

**VRRECORD** Records animations for the Marine Visualization Toolbox.

AVIOUT = VRRECORD(AVIIN, SEQUENCE, VRFIG, VRWORLD, DATA) interpolates the desired range of samples and time step (specified in SEQUENCE) of data values given in DATA. This new set of data is sent to corresponding nodes in VRWORLD and VRFIG. Each frame is added to the AVI movie object AVIIN. AVIOUT is returned when all frames are added to movie.

Note: Due to the functionality of vrfigure/capture, the VR-figure must not be closed, minimized or completely covered by another window while the recording is in progress! This will cause the animation to freeze at the last valid frame.

See also VRFIGURE, VRWORLD, VRNODE, [DIRECTOR](#), [CREATEVRML](#).

Author: Andreas Lund Danielsen

Date: 10th November 2003

Revisions:



## Product Page - Web

For updates, bug fixes, feedback and questions see the product page on the [MATLAB Central](#) and the [File Exchange](#).

Click [here](#) to automatically search the MATLAB Central File Exchange for *Marine Visualization Toolbox*.