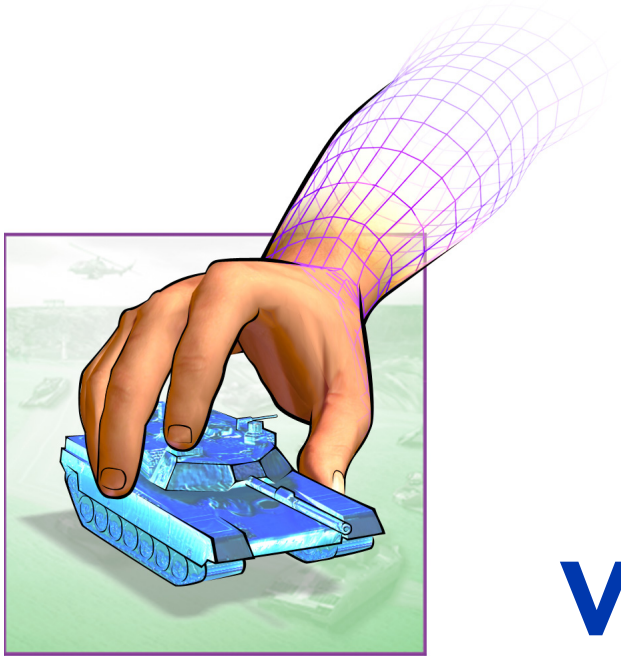
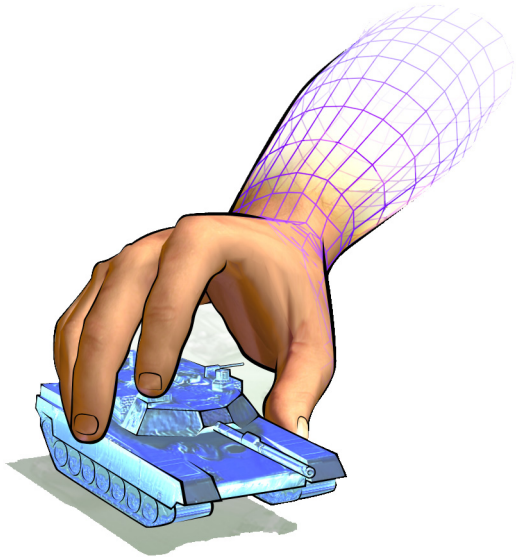


VR-Forces

Getting Started Guide

VT MÄK
A company of VT Systems



VR-Forces

Getting Started Guide

VT MÄK
A company of VT Systems

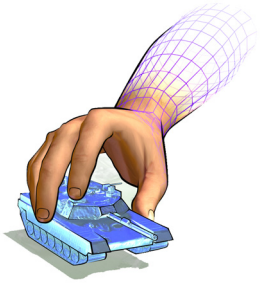
VT MÄK
68 Moulton St.
Cambridge, MA 02138 USA

Voice: 617-876-8085
Fax: 617-876-9208

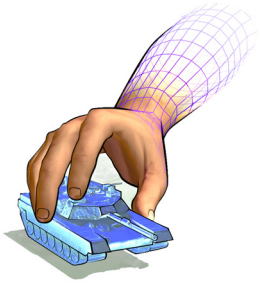
info@mak.com

www.mak.com

Revision VRF-3.12-14-090629



Contents



1. Introduction

VR-Forces is a robust application with many features. We think it is pretty easy to use – once you get to know it a bit. However, because it has so many features, which are documented in great detail in several manuals, you may be unsure where to start. And like any application, VR-Forces has its quirks, which can trip up new users. So in this Getting Started Guide we provide a simplified introduction to VR-Forces. We get you up and running quickly and we point out some of the key things to do and to avoid. Once you have gotten started, you can delve more deeply into the rest of the documentation to learn more – or just explore the application on your own.

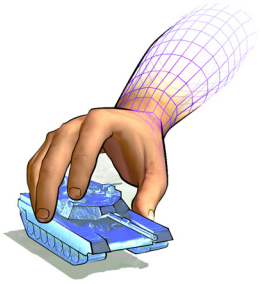
The VR-Forces documentation set is provided to all customers in print and PDF format. PDF versions of the manuals are in the *./doc* directory and, in Windows, are accessible from the VR-Forces shortcuts on the Start menu. Table 1 tells you which manual to use to find various kinds of information about VR-Forces.

Table 1: Guide to documentation

If you want to know:	Read this:
How the 2D GUI works, including navigation, scenario creation and management, and all menu commands (except performance related)	*VR-Forces Users Guide*, online help
How to use the 3D Front End and differences between the 2D GUI and the 3D Front End	*VR-Forces 3D Front End Users Guide*, online help

Table 1: Guide to documentation

If you want to know:	Read this:
How to create scenarios, assign tasks and sets, and write plans	*VR-Forces Users Guide* and video tutorials at www.mak.com/products/vrforces_tutorials.php
How to use the development kit (APIs)	*VR-Forces Developers Guide*, class documentation, header files
How to add, change, delete, or configure entity and object models and behaviors without writing code	*VR-Forces Configuration Guide*
How to optimize performance	*VR-Forces Configuration Guide*
How to use the Entity Editor, OPD Editor, or Scenario Merge utilities	*VR-Forces Configuration Guide*, online help in each utility
How to create GDB terrains from other formats, associate image data with terrains, and create MTD files	*MÄK Terrain Database Tool Users Guide*, TDB Tool online help
Add feature data to a GDB file	*MÄK Terrain Database Tool Users Guide*, TDB Tool online help
How to upgrade from older versions of VR-Forces	*VR-Forces Migration Guide, VR-Forces Release Notes*
High level information about component models	Class documentation
What the API examples do and how to build them	Class documentation
Answers to frequently asked questions	www.mak.com/support/faq.php
Which manual to look in to find specific information	*VR-Forces Master Index*



2. Installing VR-Forces

The Windows version of VR-Forces and its plug-in applications install from a typical installation wizard. The Linux version installs from an install script (DVD distribution) or by uncompressing and untarring a downloaded file.

As you proceed through the installation process, be aware of the following issues:

- Make sure you install the correct versions of the VR-Forces applications
- Install the applications in the correct order
- Uninstall and reinstall applications in the correct order
- Set up licensing correctly
- If you are a developer, install the correct version of VR-Link and other required libraries.

For details, please see Chapter 2 of *VR-Forces Users Guide* and the MÄK web site at: http://www.mak.com/products/faq_install.php. You can download an installation guide from: http://www.mak.com/doc/installguide.pdf.

Make Sure You Have the Correct Application Versions

VR-Forces has separate installers for VR-Forces, the VR-Forces 3D Front End, and the B-HAVE Module for VR-Forces. (The B-HAVE Module is an optional, extra-cost plug-in for VR-Forces.) Developers must also install VR-Link. HLA users must install an RTI. You might also have other MÄK products such as VR-Vantage and MÄK Logger. Mismatched applications will not run correctly, or at all.

- Make sure that all installers were built with the same compiler.
- Make sure that the version of VR-Link you have installed is the same version used to build VR-Forces.
- Make sure that the VR-Forces 3D Front End and the B-HAVE Module were built for your version of VR-Forces. (For example, B-HAVE Module for VR-Forces 1.4 for VR-Forces 3.12.)

If you are not sure which versions of the various MÄK products go together, please review the release notes for your version of VR-Forces and the product version and build compatibility pages on the MÄK web site: http://www.mak.com/support/productversions.php and http://www.mak.com/products/bctables.php

Install Applications in the Correct Order

The 3D Front End and B-HAVE overwrite some VR-Forces configuration files. If you install them in the wrong order, the applications will not work correctly.

To install the VR-Forces Developers Kit, you must enter an installation key during the installation process. Your MÄK salesperson will provide the key when you are given the download links or it will be included with your installation DVD. Be sure that you have it before you start installing VR-Forces. (If you do not have an installation key, you can install the VR-Forces runtime software and install the Developers Kit when you get the key.)

The correct order for installing VR-Forces is:

1. Install VR-Forces.
2. Install VR-Forces 3D Front End.
3. Install the B-HAVE Module for VR-Forces (if purchased).

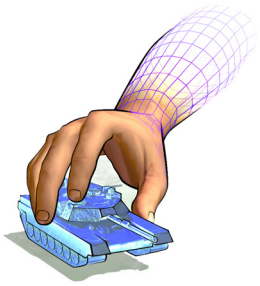
Uninstall and Reinstall Applications in the Correct Order

If you must reinstall any of the VR-Forces applications (for example, because you have received a patched version), then you must uninstall the applications in reverse order before reinstalling the updated version.

Set Up Licensing

All MÄK products require that you run a license server and have the appropriate product licenses. Complete instructions are in chapter 2 of *VR-Forces Users Guide*. License instructions are also on the MÄK web site, at: http://www.mak.com/products/install_license.php

Plug-ins may be licensed separately from VR-Forces. Read their documentation for licensing details. If you are installing the B-HAVE Module, you must read section 1.2, “Installing the B-HAVE Module”, in *B-HAVE Module for* VR-Forces *Users Guide* for licensing information.



3. Running a Scenario

This chapter explains the basic procedures for loading a scenario, explores what is going on in VR-Forces, and points you to the sections in *VR-Forces Users Guide* that provide all the details. Chapter 4, *Creating a Scenario* shows you how to create a scenario.

To simulate entities in VR-Forces, you have to:

1. Start VR-Forces.
2. Load a scenario or create a scenario.
3. Run the scenario.

! Some new users start VR-Forces and load a terrain database. Then they wonder why most of the menu options are unavailable. To create or manipulate entities, you must have a scenario loaded. Although there are some cases in which you would just load a terrain database, they are the exception for typical use of VR-Forces.

---

i This guide uses the VR-Forces 2D front-end for its examples. Most procedures are the same in the 3D Front End. Please see *VR-Forces 3D Front End Users Guide* for the exact procedures to use in that version.

Start VR-Forces

To use VR-Forces, you start two executables: a front-end (the graphical user interface, or GUI) and a back-end (the simulation engine). You can start them separately (independent mode) or with one command or menu option (combined mode). In this guide, we just use combined mode.

If you want to run VR-Forces using HLA, you may need to start your RTI before you start VR-Forces. If you are using the MÄK RTI, it will prompt you if you need to start the rtiexec.

To start VR-Forces on Windows:

1. On the Start menu, choose **Programs** → **MAK Technologies** → **VR-Forces** *x.x* → **VR-Forces 2D GUI + Simulation Engine**.

   If this is the first time that you are running VR-Forces, or if you have not previously selected a default startup connection, the Simulation Connections Configuration dialog box opens (Figure 1).

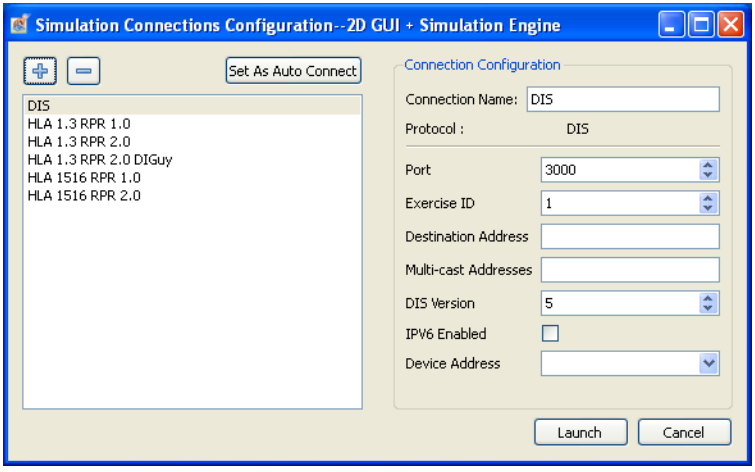


Figure 1. Simulation Connections Configuration dialog box

2. In the Simulation Connections Configuration dialog box, select a connection configuration. (You can create your own configurations, but for now use DIS or HLA 1.3 RPR 1.0.)
3. Click Launch. VR-Forces starts up (Figure 2).

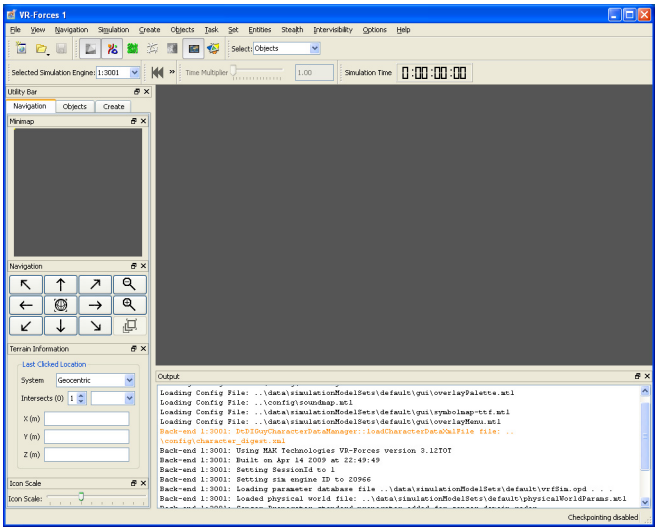


Figure 2. VR-Forces window at startup

To start VR-Forces in combined mode on Windows or Linux from the command line:

1. Open a console window.
2. Change to the *./bin* directory for the protocol you are using.
3. Enter the following command:

```
vrfLauncher --app vrfGui -z vrfSim
```

For details, please see the following sections in *VR-Forces Users Guide*:

- Section 3.1, "The VR-Forces Program"
- Section 3.2, "Working with Multiple Front-ends and Back-ends"
- Section 4.1, "Starting VR-Forces".

Load a Scenario

To get a quick feel for VR-Forces, load one of the example scenarios and run it.

To load a scenario:

1. Choose **File** → **Load Scenario**. The Load Scenario dialog box opens. It defaults to the *./data/scenarios* directory. Each example scenario is in its own directory.
2. Select the *makland* directory.
3. Select *maklanddemo.scn*.
4. Click Open. A terrain and entity icons are displayed in the VR-Forces window (Figure 3). (In the illustration, the console window has been closed.)

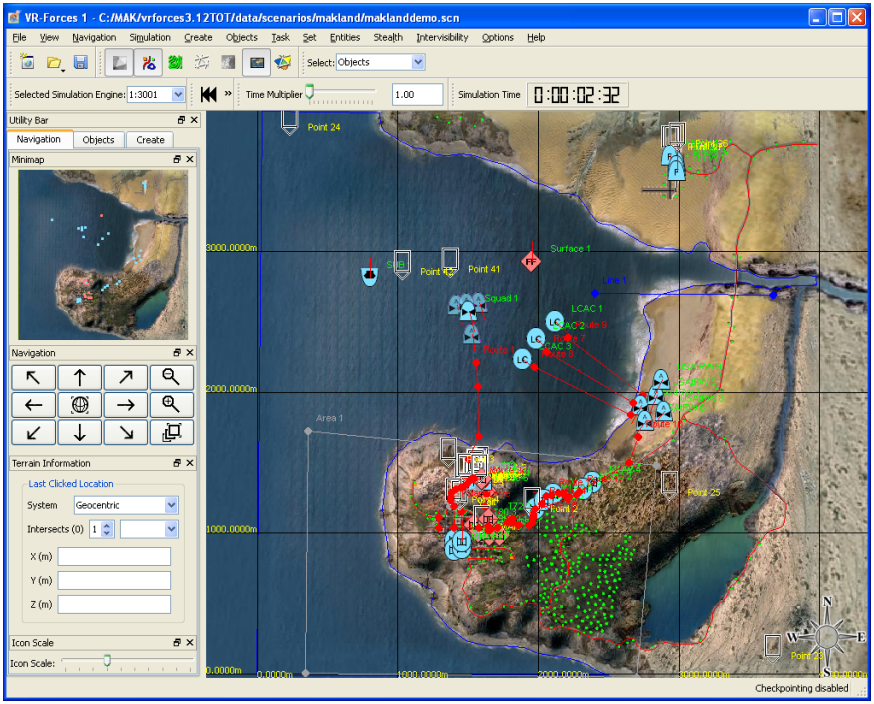


Figure 3. maklandDemo scenario

Run the Scenario

When the scenario is finished loading, you can run it.

➤ To run a scenario, choose **Simulation** → **Run Simulation**, or click the Run arrow on the Simulation toolbar (Figure 4).

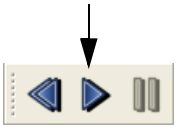


Figure 4. Simulation toolbar

The entity icons begin moving. You will see transient icons that represent missiles, animated explosions, lines representing munition fire and detonation, and other graphics.

Figure 5. Simulation running

Understand the Scenario

In addition to displaying entities, objects, and animated graphics, VR-Forces provides a lot of information about what is going on as a scenario unfolds. In this section we will cover some of the ways to learn about a scenario.

Getting Information about Individual Entities

On the left side of the VR-Forces window is the Utility toolbar.

1. Select the Objects tab. It has five sub-tabs that show different views of the objects in a scenario.
2. Select the leftmost tab. It lists all entities.
3. Select (click) the entity named M1A2 4. In Figure 6, it is highlighted. (It is just to the east of the town, advancing along a road towards the town.)

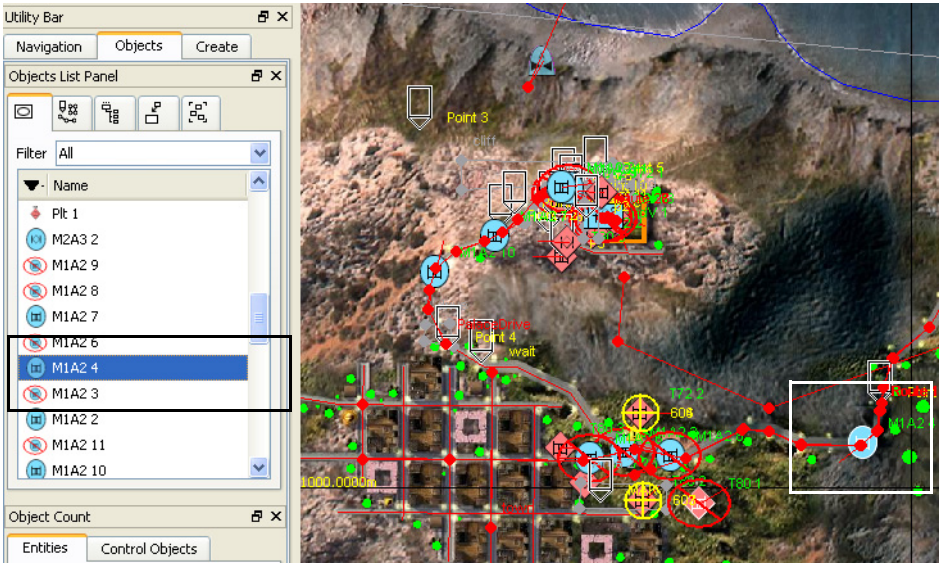


Figure 6. M1A2 4 advancing towards the town

4. Double-click the icon. A dialog box opens (Figure 7). It has tabs that describe the entity's state and what it is doing. The M1A2 is an individual entity. If it were an aggregate entity, the dialog box would list its subordinates. At the bottom of the dialog box is a console window. It displays messages that the entity has received.

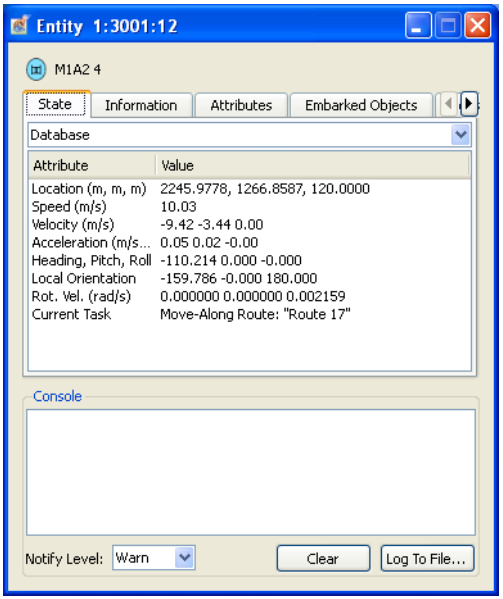


Figure 7. Entity Information dialog box

You can display information dialog boxes like this for most objects.

For more information about displaying information about entities and objects, please see Chapter 10, *Displaying Entity Information*, in *VR-Forces Users Guide*.

Viewing Plans

Most of the entities in the scenario are moving. Entities move in response to independent tasks or tasks that are part of a plan. A plan is a list of tasks and set data requests that an entity executes in sequence. A plan can also contain tasks that are executed only under certain conditions.

To view the plan for M1A2 4:

1. Select M1A2 4.
2. Choose **Entities** → **Plan**. The Plan window opens (Figure 8).

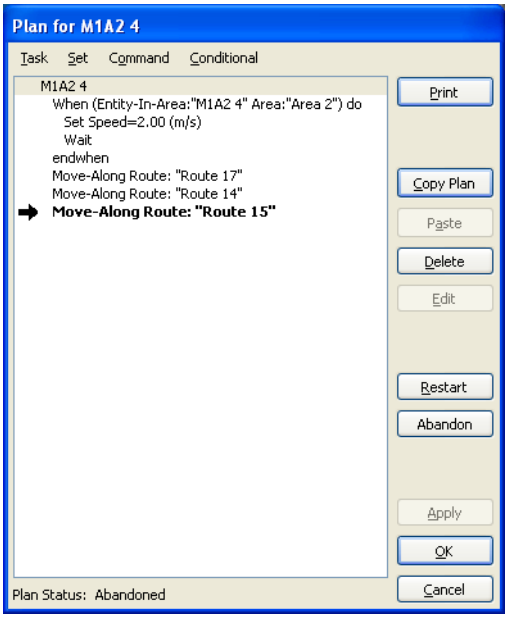


Figure 8. Entity plan

This plan is fairly simple. It tells the entity to move along three routes in succession. It also includes a condition block. The condition is that when the entity enters Area 2, it should change its speed and stop (Wait task).

For more information about tasks and plans, please see Chapter 5, *Assigning Tasks* and Chapter 6, *Writing Plans*.

Viewing Force Hierarchies and Relationships

All entities exist within the context of a force, usually just called Friendly and Opposing. Some entities may be Neutral. (You can create up to 255 different forces.) The forces emulate military hierarchies.

➤ On the Objects tab of the Utility toolbar, select the Echelon View tab (Figure 9). This tab shows the hierarchical arrangement of the entities by force.

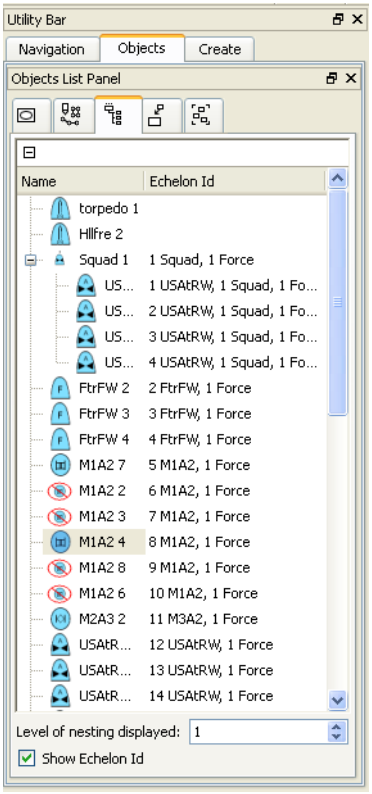


Figure 9. Echelon View

Scenario Objects

In addition to entities, a scenario can contain control objects and overlay objects. Control objects are points, lines, and areal objects that entities can interact with. For example, an entity can move to a point, or follow a route. Overlay objects are point, lines, areas, text, and various types of symbols that you can draw on the screen as though they were on transparent plastic overlays. You can hide and display them by overlay.

Sometimes the distinction between control objects and overlay objects is a bit blurred, but in general, control objects are paths and places on the terrain and overlay objects are tactical graphics that you create to help you understand the scenario.

Figure 10 shows the list of objects on the Objects toolbar and identifies some objects on the map.

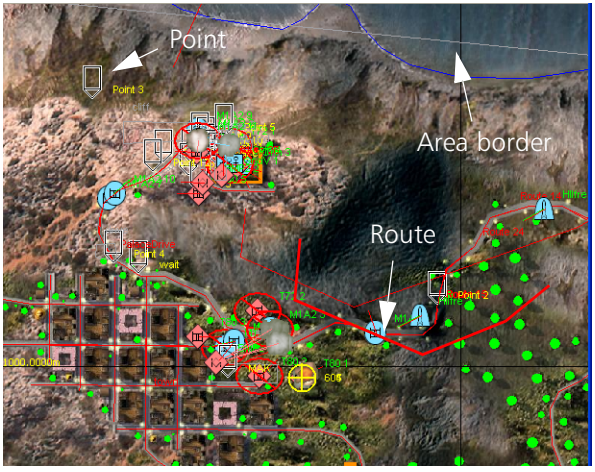
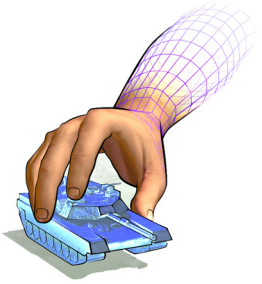


Figure 10. Scenario objects

For more information about objects, please see the following sections in *VR-Forces Users Guide*:

- Chapter 7, *Managing Objects*
- Chapter 16, *Overlays and Graphical Objects*.



4. Creating a Scenario

This chapter and the next two are essentially an extended tutorial for creating a simple scenario. The finished scenario, GettingStartedExample (*./data/scenarios/GettingStarted/GettingStartedExample*), is distributed with VR-Forces.

To create a scenario, you must:

1. Choose a terrain.
2. Create entities.
3. Optionally, create control objects and overlay objects.
4. Tell the entities what to do.
5. Save the scenario.

Create a Scenario

To create a scenario:

1. Choose **File** → **New Scenario**, or click the New Scenario button on the File toolbar. The Select Simulation Database dialog box opens (Figure 11). By default, it opens in *./data/terrain*, the directory that contains the terrain databases shipped with VR-Forces.

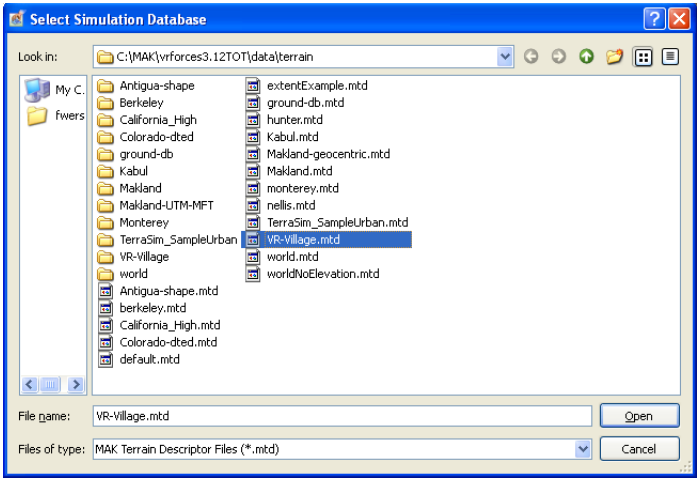


Figure 11. Select Simulation Database dialog box

2. In the file list, select *VR-Village.mtd*. (A MÄK terrain descriptor (MTD) file packages together information about a terrain and related imagery for convenient loading in VR-Forces.)
3. Click Open. The New Scenario dialog box opens (Figure 12).

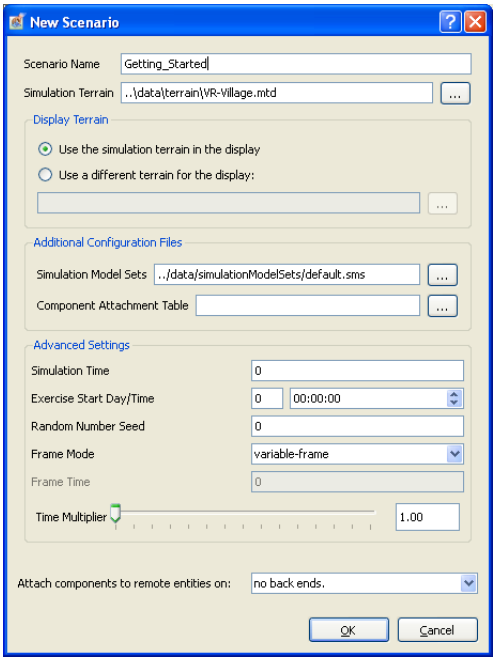


Figure 12. New Scenario dialog box

4. Give the scenario a name.
5. Accept all the default scenario parameters. Click OK.

   The map displays the terrain database that you chose (Figure 13).

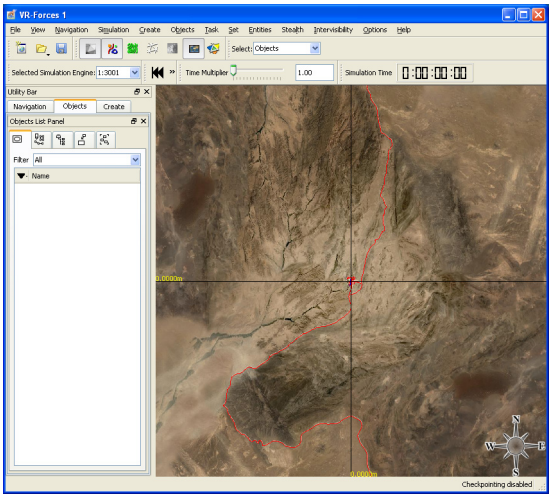


Figure 13. VR-Village terrain

6. Save the scenario. (It is a good idea to save your scenario frequently as you develop it.)

    a. Choose **File** → **Save Scenario**. A Save dialog box opens.

    b. Click the create folder icon.

    c. Type a name for the folder, for example, myGettingStarted.

    d. Select the folder.

    e. Click Open.

    f. Type a name for the scenario, for example myGettingStartedExample.

    g. Click Save.

For details about creating scenarios, please see the following sections in *VR-Forces Users Guide*:

- Section 3.3, "Scenarios"
- Chapter 5, *Creating and Running Scenarios*.

For details about MTD files, please see *MÄK Terrain Database Tool Users Guide*.

Create an Entity

After you create a new scenario, you populate it with the entities you need to achieve the goals of your simulation project.

To create a new entity:

1. In the VR-Village terrain, the highly detailed portion of the terrain is the village in the very center. Hold down the **Shift** key and drag a box around the center of the terrain to zoom in on it (Figure 14 and Figure 15).

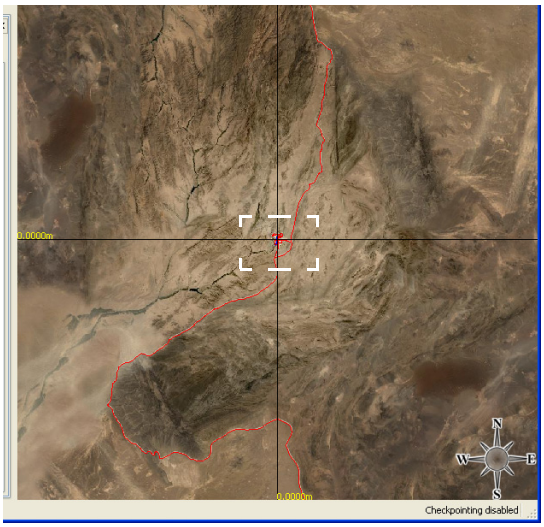


Figure 14. Zooming in on the terrain

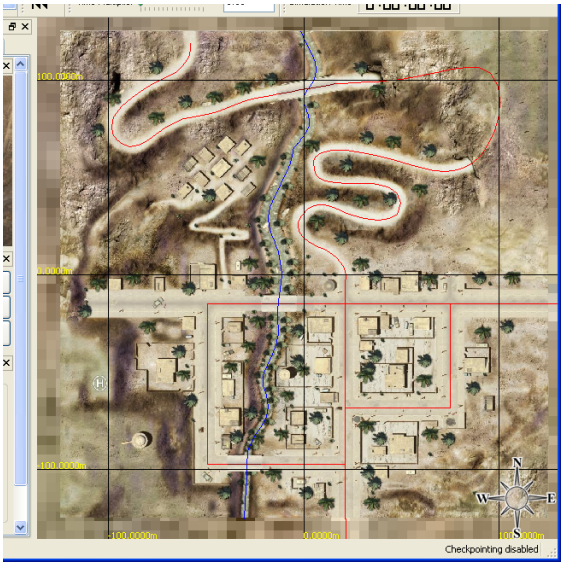


Figure 15. VR-Village

2. On the Utility toolbar, select the Create tab.
3. In the Categories list, select Ground.
4. In the Force list, select Friendly.
5. In the list of entities, select M2A2 Bradley IFV. The cursor changes to show the icon for an M2A2 Bradley.
6. Click on the map where the road enters the village from the south. An icon is placed at that location. (When you are in create entity mode, you can continue clicking to create multiple instances of the selected entity type.)
7. Right-click to exit create entity mode. The scenario now looks like (Figure 16).

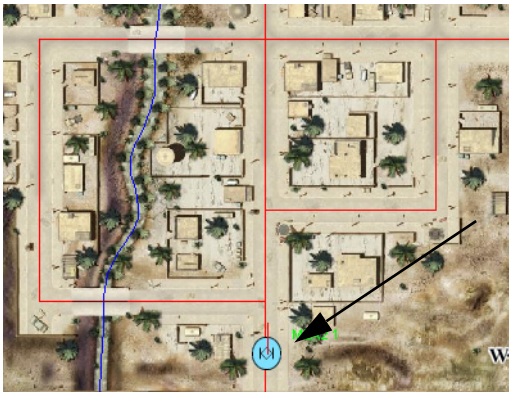


Figure 16. New entity

For details, please see Section 8.2, "Creating Entities", in *VR-Forces Users Guide*.

Create a Control Object

Control objects are points, lines, and areas that identify locations on the terrain. An entity can interact with these objects in various ways. One common use for control objects is to have an entity move to a point or along a route.

To create the route:

1. Choose **Create** → **Route**. The Create Route dialog box opens and the cursor changes to create object mode. If the Create Route dialog box obscures the terrain, move it out of the way.
2. Click on the map in front of the Bradley (1 in Figure 17).

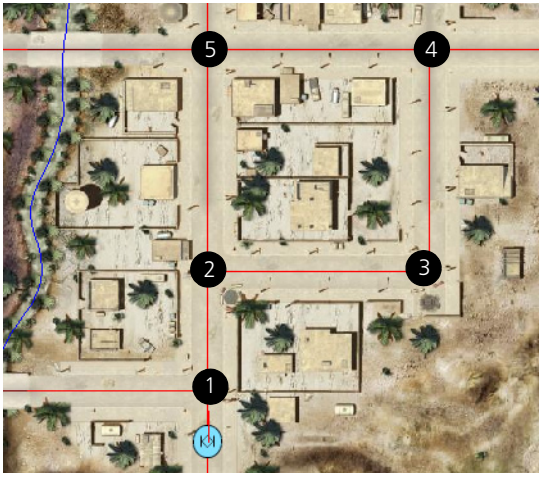


Figure 17. Vertex locations

3. Add additional route vertices by clicking in the locations of 2, 3, and 4 in Figure 17.
4. For the end point, right-click at location 5.
5. Accept the default name for the route (Route 1).
6. Click OK. The route is created.
7. Save the scenario.

VR-Village has road features, which are displayed as red lines. Routes are also displayed as red lines. To make the location of the route clear, turn off the display of features, as follows:

1. Choose **Options** → **Terrain View**. The Terrain View dialog box opens.
2. In the Terrain Drawing group box, clear the Features check box.
3. Click OK. It should look like (Figure 18).

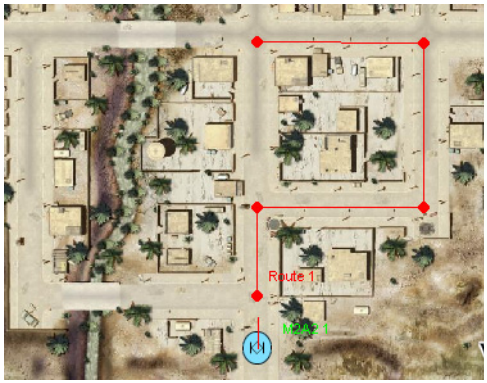


Figure 18. Route 1

For details about creating control objects, please see Section 16.3, "Creating Graphical Objects", in *VR-Forces Users Guide*.

Save the Scenario

The biggest mistake that new VR-Forces users make is to run a simulation before they save it. VR-Forces prompts you to save new or changed scenarios before you run them, but it does not automatically save them.

If you have not saved the scenario yet, save it now.

To save a scenario:

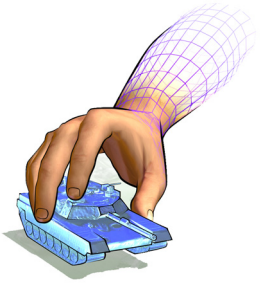
1. Choose **File** → **Save Scenario** or click the Save button on the file toolbar. The Save Scenario dialog box opens.
2. Optionally, create a directory in which to save the scenario.
3. Select the directory in which you want to save the scenario.
4. Give the scenario a name.
5. Click Save.

For details, please see Section 5.3, "Saving a Scenario", in *VR-Forces Users Guide*.

Tell the Entity What To Do

Entities know how to do some things without being told. If an entity detects an enemy entity, it fires automatically (unless you have told it not to). If an entity is moving and it detects another entity in its way, or an obstacle of some sort, it tries to avoid it. However, to do anything really useful, you have to give an entity a task. You can give an entity tasks by writing plans, or by assigning independent tasks.

This is an important enough subject that we will devote the next two chapters to it. Please read on.



5. Assigning Tasks

This chapter is an introduction to tasks and how to assign them. You can give an entity a task by including it in a plan or by assigning it to the entity from the Task menu. We call tasks that are assigned from the Task menu “independent tasks”.

The process of assigning a task is largely the same however you do it. We provide examples in this guide, but we really want to focus on the strategy for assigning tasks and plans and alert you to common problems that new users have.

Independent Tasks

You can give an entity an independent task at any time. You can do it during scenario creation and you can do it while a scenario is running. The most important things to remember about independent tasks are that when you give an entity an independent task:

- It stops whatever it is doing and immediately begins to execute the new task (unless the new task is a concurrent task).
- If the entity is executing a plan, the plan is abandoned and the entity executes the independent task. When it is finished with the task, it does not return to the plan.

Concurrent Tasks

Most independent tasks immediately override whatever an entity is doing. Concurrent tasks are the exceptions to this rule. A concurrent task does not cancel an entity's current task. An entity executes the concurrent task and then continues the task it was working on. The primary purpose of concurrent tasks is to let an entity send a radio message while it is executing a movement task. (All movement tasks are mutually exclusive (that is, not concurrent)).

Assigning an Independent Task

In our example scenario, the M2A2 Bradley vehicle is sitting at the edge of town, doing nothing. Now we will tell it to move along the route using an independent task.

1. Select the Bradley.
2. Choose **Task** → **Move Along Route**. The Move Along Route dialog box opens (Figure 19). It lists the routes in the scenario. In this example, there is just one route.

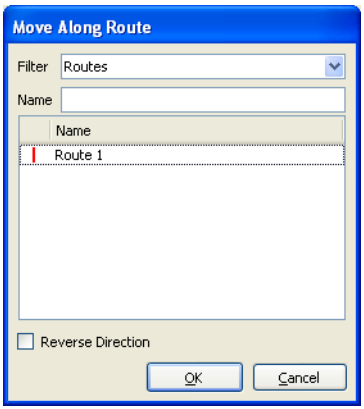


Figure 19. Move Along Route dialog box

3. Select Route 1.
4. Click OK.
5. Choose **File** → **Save**.

At this point nothing is happening, because the scenario is not running. However, you can check the Entity Information dialog box for the Bradley (double-click its icon) to see that the task has been assigned (Figure 20).

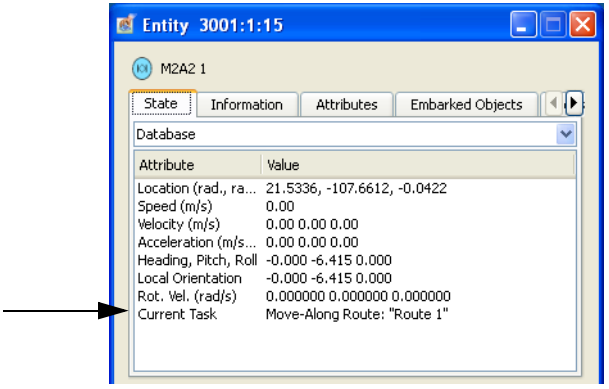


Figure 20. Entity Information dialog box

For complete details about task concepts and independent tasks, please see Chapter 11, *Assigning Tasks*, in *VR-Forces Users Guide*.

Running the Simulation

At this point, the example scenario is quite simple and does not do much. However run it now to verify that everything you did works as expected.

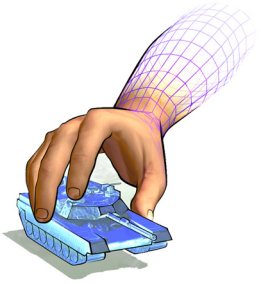
➤ To run the simulation, choose **Simulation** → **Run**. (If you have not saved the scenario, VR-Forces prompts you to save it.)

The Bradley should move along the route.

For details, please see Section 5.4, "Starting a Simulation", in *VR-Forces Users Guide*.

Set Data Requests

Set data requests (sets for short) change the state of an entity. For example, you could set an entity to the destroyed state, or restore an entity from a destroyed state to a healthy state. Like tasks, you can issue sets from the Set menu or in plans. The main distinction between a task and a set is that a set always takes place immediately and sets do not interrupt tasks, except where logically expected, such as Set Destroyed or if you set a needed resource to zero.



6. Writing Plans

In Chapter 5, *Assigning Tasks*, we gave the entity in our simple scenario a single, independent task. When we ran the simulation, it executed that task. What if you want the entity to do more than one thing? Could you assign it several independent tasks? No, because each task would override the previous one. You would have to watch the scenario and when the entity completed each task, give it the next task. What if there were many entities in the scenario? Could you keep track of them all and give the appropriate task at the appropriate time? Probably not.

If you want to control multiple entities in varying circumstances, you need to give them plans. In this chapter we talk about the types of plans, add a plan to our example scenario, and then discuss use of conditions in plans.

Choosing Between Independent Tasks and Plans

Here are some guidelines for choosing between independent tasks and plans:

- For simple scenarios with few entities or quick proof of concept actions: use plans or independent tasks
- For moderately to very complex scenarios: use plans
- To script assignment of independent tasks: use global plans
- While running a scenario: abandon an entity's planned actions by assigning independent tasks, by writing a new plan, or by the Abandon Plan command.

Entity Plans and Global Plans

VR-Forces has two kinds of plans – entity-specific plans (entity plans) and global plans. The two types of plans look similar. They both can include tasks, set data requests, and simulation commands. They both can include conditional blocks. However, the way that tasks are assigned and carried out is very different. It is very important that you understand the differences.

In an entity plan:

- The entity owns the plan and it stays with the entity regardless of name or force changes.
- All of the tasks and sets apply implicitly to the entity.
- Tasks are executed sequentially as they are completed. In other words, task 2 is not issued until task 1 is complete.

In a global plan:

- There is no ownership. Global plans automate entity-independent scenario management.
- Tasks and sets are explicitly assigned to the entity that will execute them.
- All tasks are essentially independent tasks. They are sent to the assigned entities sequentially, but without regard to what the entity is doing. If a global plan has successive tasks for the same entity, they are sent in order, but the second task immediately overrides the first task, just as if you assigned two successive tasks from the Task menu.

If you want an entity to execute a series of tasks, one after the other, write an entity plan.

For details about the ins and outs of using tasks, set data requests, and plans, please see the following chapters in *VR-Forces Users Guide*:

- Chapter 11, *Assigning Tasks*
- Chapter 12, *Issuing Set Data Requests*
- Chapter 15, *Writing Plans*.

Write an Entity Plan

Previously, we assigned the Bradley vehicle a Move Along Route independent task. Now we will do the same thing using a plan. And to highlight the difference between independent tasks and plans, we will give it another task in the plan.

To write the plan:

1. Select M2A2 1.
2. Choose **Entities** → **Plan**. The Plan dialog box opens (Figure 21).

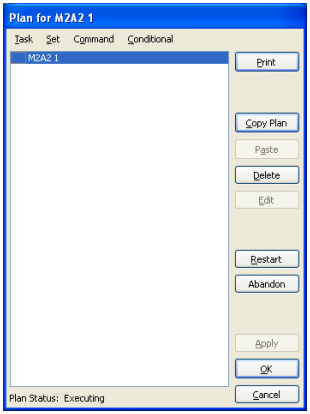


Figure 21. Plan dialog box

3. In the plan window, choose **Task** → **Move Along Route**. The Move Along Route dialog box opens (Figure 22).

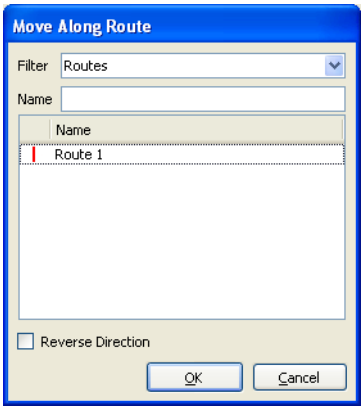


Figure 22. Move Along Route dialog box

4. In the window, select Route 1.
5. Click OK. The task is added to the plan (Figure 23).

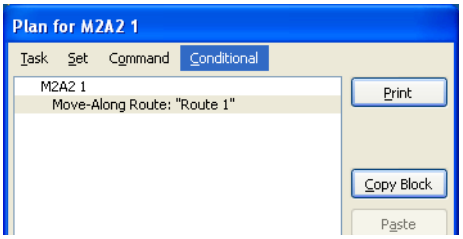


Figure 23. Plan with task

6. Choose **Set** → **Heading**. The Set Heading dialog box opens.
7. In the Heading text box, type 180.
8. Click OK. The set data request is added to the plan (Figure 24).

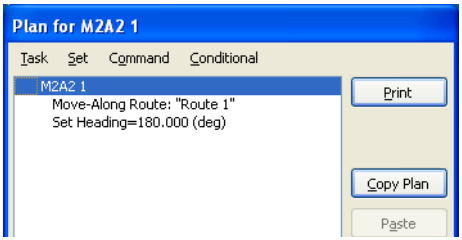


Figure 24. Plan with task and set data request

9. Click **Task** → **Move to Location**. The Move To Location dialog box opens.
10. Click along the road heading back towards where the Bradley started. The coordinates are entered in the Move To Location dialog box.
11. Click OK. The task is added to the plan. Figure 25 shows the final plan.

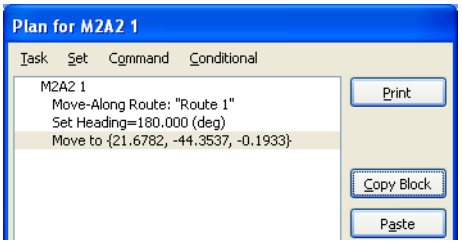


Figure 25. Plan with task and set data request

12. In the Plan window, click OK. The Bradley already has a task (the independent task we assigned earlier). VR-Forces displays a message to let you know that if you save the plan, it will override the current task.
13. Click Yes. The plan is saved.
14. Save the scenario.

Run the scenario. The Bradley follows the route. Then it changes its heading to face south and drives to the location you specified.

Using Conditions in Plans

If you have a simple scenario in which you know everything that will happen and when it will happen, you can write plans that script this process exactly. However, in many simulations you do not know exactly what will happen and when it will happen. You probably know that some things might happen and you know what you want entities to do if certain events occur. You can plan for these events by putting conditional blocks in plans.

The three types of conditions supported by plans are:

- `If`: An `If` condition tests whether something is true at precisely the point when the test is made. If the test is true, the tasks in the `If` block get executed. Use an `If` condition if you want to test a condition and respond to it only at a certain point in a plan's execution.
- `When`: A `When` condition is tested repeatedly during a scenario. If at any time the condition being tested is true, the tasks in the `When` block get executed. Use a `When` condition if you want to respond to an event, but you do not know when that event will occur. The first time that a `When` condition is encountered in a plan, VR-Forces registers it. Then VR-Forces begins to test it.
- `While`: A `While` is not really a condition. It is a loop. VR-Forces enters the loop if a condition is true and repeats the tasks in the `While` block over and over as long as the condition stays true.

The most important things to remember about using conditions are:

- `If` conditions get tested once – when the plan gets to the condition as it sequentially moves through the plan. `If` conditions often do not work as expected because of timing issues: they are tested too soon, or too late. If a condition is time-dependent, you need to control for that issue in your planning – perhaps by using a `When`.
- `When` conditions do not get tested until they are registered. You should almost always put `When` conditions at the beginning of a plan. This ensures that they get registered before the entity starts executing its tasks. The only time you would put a `When` condition in the middle of a plan is if you do not want VR-Forces to start testing until the entity has done some other things first. (And even then you could probably nest the `When` inside another `When` and put it at the beginning of the plan.)

Remember, the VR-Forces plan editor ensures that the syntax of plans is correct, but it cannot ensure that the logic of your plan is what you intend.

Global Plans

Earlier in this chapter, we explained the differences between entity plans and global plans ("Entity Plans and Global Plans," on page 32). Adding a task or set to a global plan is slightly different from adding a task to an entity plan, but overall the mechanics of creating global plans are similar to those for entity plans. In this section we focus on the behavioral differences between global plans and entity plans.

The GettingStartedExample scenario(*./data/scenarios/GettingStarted/GettingStartedExample*) has two global plans (Good Global Plan and Bad Global Plan) that demonstrate how to write a global plan that does what you want and how to write one that does not do what you want. (You can write as many global plans as you want to.)

In each plan, we create a second M2A2 entity and give it the same tasks as M2A2 1 – follow the route, set heading, and move to location.

The Bad Global Plan

In Bad Global Plan, we wait until 10 seconds of simulation time have elapsed (so we put everything in a `When` block), then create the entity and give it the three tasks, just like in the plan for M2A2 1. The plan looks like this:

```
1     When (SimTime(0:00:00:10) >) do
2       Create Object: Name: M2A2 2 Type: 1:1:1:225:2:1:1:0 Force: 1
          Location: {21.6724, -114.9208, 1.6544}
3       Task Object M2A2 2 Task: Move-Along Route: "Route 1"
4       Set Data Request; Object: M2A2 2 Request: Set Heading=
          180.000 (deg)
5       Task Object M2A2 2 Task: Move to {21.4123, -43.1433, -0.1933}
6     endwhen
```

Run the GettingStartedExample scenario. The following happens:

1. At the start of the scenario, the `When` block is registered.
2. After 10 seconds, the entity gets created.
3. It starts to move (line 3), but then immediately changes heading), then immediately starts moving to the location).

The entity never follows the route because a global plan sends tasks without regard to completion of prior tasks. The last task sent overrides all previous tasks for a given entity. In this case, the Move To Location task overrides the Move Along Route task. If you want an entity to complete each task before starting the next one, you must have the plan evaluate task completion before issuing a new task.

The Good Global Plan

The Good Global Plan is designed to wait until the entity finishes following the route before it sends the next task. We do this by drawing an area at the end point of the route. When the entity enters the area, we can assume that it has finished following the route. The plan is structured as follows:

1. Everything is in a `When` block that does not get triggered until one minute of simulation time has passed (line 1). (This is so the good global plan does not interfere with the bad global plan.)
2. After one minute, we delete the entity created by the Bad Global Plan (line 2) and then create a new entity for this plan (line 3).
3. We create the area at the end of the route (line 4).
4. We create a `When` block to test for when the entity enters the area (line 5). Inside the block we set the heading (line 6) and add the Move to Location task (line 7).
5. After the nested `When` block, but within the main `When` block, we put the Move Along Route task (line 9).

```
1    When (SimTime(0:00:01:00) >) do
2      Delete Object: Name: M2A2 2
3      Create Object: Name: M2A2 2 Type: 1:1:1:225:2:1:1:0 Force: 1
         Location: {21.6724, -114.9208, 1.6544}
4      Create Object: Name: Area 1 Type: 1:18:0:0:1:0:0:0 Force: 255
         First Point (of 4): {17.3571, -12.5476, -0.1933}
5      When (Entity-In-Area: ,, Entity:"M2A2 2" Area:"Area 1") do
6        Set Data Request; Object: M2A2 2 Request: Set
       Heading=180.000 (deg)
7        Task Object M2A2 2 Task: Move to {21.4123, -43.1433, -
       0.1933}
8      endwhen
9    Task Object M2A2 2 Task: Move-Along Route: "Route 1"
10   endwhen
```

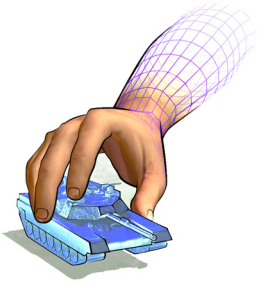
The execution sequence for the Good Global Plan is as follows:

1. When the scenario starts, the main `When` block is registered.
2. After one minute, M2A2 2 is deleted and a new M2A2 2 is created.
3. Area 1 is created.
4. The Entity In Area condition is registered.

5. The entity is assigned the Move Along Route task. It moves along the route.
6. When the entity enters Area 1, the Entity-In-Area `When` condition is satisfied. The entity's heading is set and it is given the Move To Location task.
7. The entity moves to the location.

This example shows how global plans are different from entity plans and how important it is to use the correct logic in a global plan.

For more information about global plans, please see Section 15.5, "Creating Global Plans", in *VR-Forces Users Guide*.



7. VR-Forces Performance Issues

This chapter is a brief introduction to the major factors that affect performance. Chapter 1, *Optimizing Performance*, in *VR-Forces Configuration Guide* discusses performance issues in detail.

Terrain Database Size

VR-Forces loads a copy of the terrain database for each front-end and back-end executable. Therefore, in a combined mode session it loads two copies of the database. Similarly, loading the 2D front-end and 3D front-end on the same computer would require loading two databases. The larger the database, as measured by the number of polygons and features, the more memory it will take, resulting in less memory available for simulation and display tasks. You can alleviate this problem somewhat by running the front-end and back-end on different computers.

If the terrain that you want to load is too large to fit into memory, you will have to reduce the number of polygons or features, or both to a manageable level. Depending on how the terrain was created, you may be able to optimize it to improve performance.

For information about optimizing terrain, see *MÄK Terrain Database Tool Users Guide*.

Scenario Size

The size of a scenario affects performance in several ways. The more entities and objects in the scenario, the harder the back-end has to work to simulate them and the longer it takes for the front-end to update the display as entities change their state.

The effect of large numbers of entities is highly dependent on what they are doing. If many entities are moving at the same time, it places a greater load on VR-Forces than if only some of them are. If many entities are moving in proximity to each other, the increased calculations required for obstacle avoidance affects performance.

You can improve performance for large scenarios by spreading the simulation load over several back-ends.

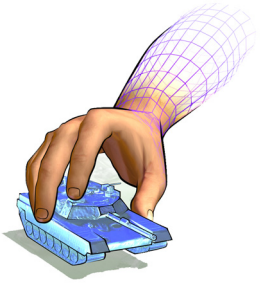
Simulation Protocol and Network Considerations

VR-Forces supports DIS and HLA. Each has its pros and cons.

DIS does not require any additional software, so it is easy to use out-of-the-box. However, DIS requires that entities send a complete state update (heartbeat) at regular intervals (typically every 5 seconds) even if their state has not changed. In large scenarios, this can flood the network with update messages, which can result in dropped packets.

HLA is more efficient than DIS. It does not require a heartbeat. Unless an application asks it to send a complete state update, HLA only sends data when an entity's state changes and it only sends the data that has changed. To use HLA, you must install an HLA RTI, which is available from various software vendors (including VT MÄK). This requirement introduces additional installation and configuration effort.

VR-Forces supports the HLA 1.3 specification and the SISO DLC version of the IEEE 1516 specification. The HLA 1.3 version is more commonly used than HLA 1516.



8. VR-Forces Utility Applications

VR-Forces includes the following utility applications to help you with common tasks:

- Entity Editor
- OPD Editor
- Scenario Merge
- MÄK Terrain Database Tool.

The utilities are in the *./bin* directory and, on Windows, can be started from the VR-Forces\Tools submenu on the Start menu.

The Entity Editor

The Entity Editor helps you create new entity types and edit existing entities. Using the Entity Editor is the simplest way to add new systems to entities, such as weapons and sensors. It lets you:

- Add forces and specify force assignment
- Assign categories for the Create menu and Entity Creation Palette
- Specify the icon an entity uses
- Specify core entity attributes
- Edit DI-Guy character type, appearance, and animation
- Create and edit aggregates
- Edit aggregate formations
- Create new simulation model sets.

For complete instructions for using the Entity Editor, please see Chapter 5, *Editing Entity Models*, in *VR-Forces Configuration Guide*.

The OPD Editor

The OPD Editor lets you edit all entity parameters and systems. When you want to edit entity models, always start with the Entity Editor, it is much easier to use than the OPD Editor. But if you need to edit parameters that the Entity Editor does not cover, then use the OPD Editor.

For details about using the OPD Editor, please see Chapter 6, *Advanced Entity Model Editing*, in *VR-Forces Configuration Guide*.

Scenario Merge

By default, VR-Forces assumes that a scenario is being built by a single user, or several users at different front-ends that are sharing the same back-ends. However, in some use environments, creation of a scenario is distributed to different groups working asynchronously, with the intention of merging the resulting scenarios into one large scenario. Merging scenarios by hand is a very difficult process, which is prone to error. The Scenario Merge tool lets you easily merge scenarios together.

For details about using the Scenario Merge tool, please see Chapter 4, *Merging Scenarios*, in *VR-Forces Configuration Guide*.

MÄK Terrain Database Tool

The MÄK Terrain Database Tool (TDB Tool) lets you create GDB terrains from other terrain formats and package the terrain with image and feature data for quick loading in VR-Forces.

You can load terrain data in several different formats directly into VR-Forces. When you do this, VR-Forces uses the TDB Tool to convert the terrain data to VR-Forces's native format (GDB). The TDB Tool uses default values to convert the data. However, there may be cases in which you want to optimize a terrain for use with VR-Forces. If you convert a terrain in the TDB Tool, you can take advantage of its conversion parameters to create optimized GDB terrains.

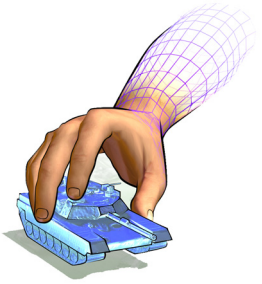
In addition to converting terrain data to GDB, the TDB Tool lets you specify images to display on top of the polygonal data. You can also specify 3D terrains for use with the VR-Forces 3D Front End.

All of this data is organized into MTD files for quick loading in VR-Forces.

The TDB Tool has other beneficial features, as follows:

- You can add point and area feature data to a terrain.
- You can split large images to create multi-resolution bitmaps for improved viewing in VR-Forces
- If you have installed the B-HAVE Module for VR-Forces, the TDB Tool can generate PathData for use by B-HAVE-enabled entities.

For complete details about the TDB Tool, please see *MÄK Terrain Database Tool Users Guide*.



9. Tutorials and Example Scenarios

VR-Forces Users Guide includes two tutorials that go into greater detail than the example in this guide. We also have some brief videos that illustrate the tutorials. You can view the video tutorials at www.mak.com/products/vrforces_tutorials.php.

VR-Forces includes many example scenarios that demonstrate its features. Here is an annotated list of the example scenarios provided with VR-Forces:

- apache_groundDB: Three helicopters attack a convoy. Demonstrates movement tasks, use of routes and waypoints, and a conditional task.
- berkeleydemo: Helicopters and tanks defend against an advancing convoy. Fixed-wing entities battle in the air. This scenario includes ground vehicles, rotary-wing, fixed-wing, surface, and lifeform. It includes an aggregate. it uses routes, waypoints, and phase lines. Plans use movement tasks and conditions. This scenario is the basis for the berkeleyDemo recording distributed with MÄK Data Logger.
- breaching: Three tank platoons engage opposing forces. Utility vehicles breach a mine field. This scenario contains aggregate entities. It also demonstrates use of tactical graphic overlays.

- embarkdemo: A helicopter picks up passengers and takes them to an aircraft carrier. An LCAC lands on a beach and discharges a HMMWV and truck. A firefight takes place near the airport. This scenario demonstrates the VR-Forces embarkation feature. It also shows a Patriot missile shooting down a helicopter. It includes fixed-wing, rotary-wing, ground, surface, and lifeform entities. This scenario is the basis for the embarkdemo recording distributed with MÄK Data Logger.
- embarkexample: This scenario is a subset of the embarkdemo. It only demonstrates the embarkation activities. The fire fight at the airport and the LCAC landing are not part of this scenario. This scenario is fully explained in the Embarkexample tutorial in *VR-Forces Users Guide*.
- makland: Makland is a complex scenario that includes ground, air, surface, subsurface, and aggregate entities. Two groups of helicopters and two tank platoons converge on the town and then to the palace. Some of them have fairly complex plans to help them navigate the narrow streets without running into each other. This scenario is the basis for the maklanddemo recording distributed with MÄK Data Logger.
- takeoffandlanding: Two jets take off and then land at an airport. This scenario demonstrates the fixed-wing take off and fixed-wing landing tasks.

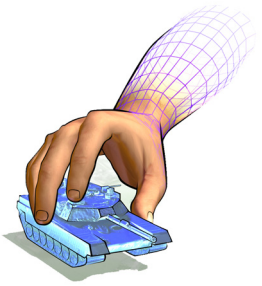
If you have the B-HAVE Module for VR-Forces installed, the following example scenarios are included:

The B-HAVE Module includes the following example scenarios:

- Crowd: 100 civilians wander randomly around the city in the TerraSim_SampleUrban terrain. They go in and out of buildings and to multiple floors of buildings.
- Interior Movement: Demonstrates the movement of lifeform entities within buildings. Some civilians wander randomly in and out of buildings, while some DI and other civilians follow pre-determined plans to reach points on different floors of some of the buildings in TerraSim_SampleUrban. DI 1 has no plan. You can task him to move to some of the waypoints to demonstrate dynamic path planning.
- Scatter: A hostile DI follows a plan to move to a set of patrol points while twenty civilians use Lua scripts to keep hidden from the hostile entity. The civilians run behind buildings whenever they see the hostile entity.

- Makland B-HAVE: Mixed vehicle and life form activity on the Makland terrain. Some civilian traffic moves along the roads. Some dismounted infantry (DI) walk in and out of the palace, changing guard duty, and some military vehicles leave the palace to drive along the roads to locations in the town.
- Rescue Convoy: All entity activity is controlled using Lua scripts that are assigned in plans. When a Blue civilian's vehicle is destroyed, friendly vehicles move in a convoy to rescue him. Hostile DIs flee the approaching convoy.
- Traffic: Civilians and civilian vehicles wander around the town in Makland. The scenario was created using multiple entity create, with a Wander task assigned at creation.
- CarBomb: Entity behavior is controlled through B-HAVE tasks in entity plans. Civilians wander around the business district of a small town. Enemy combatants move through tunnels and buildings to set up positions while a car drives to a location and explodes. Surviving civilians flee. Dismounted infantry search for the enemy combatants. Reinforcements arrive by helicopter, disembark, and help search for enemy combatants or take up positions to secure the town. Civilians return to the bomb site. This scenario is the basis for the VR-Village-CarBombDemo recording distributed with MÄK Data Logger

These scenarios are provided to demonstrate the features of VR-Forces. They do not pretend to demonstrate realistic military tactics or strategy.



Index

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

Link - Simulate - Visualize$_{VRF}$

VRF-3.12-14-090629

68 MOULTON STREET CAMBRIDGE, MA 02138 617.876.8085
www.mak.com