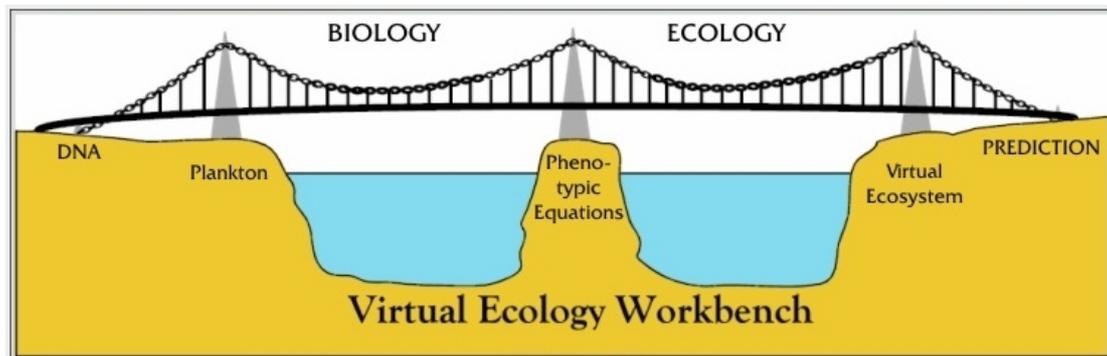


VIRTUAL ECOLOGY WORKBENCH

USERS HANDBOOK

Volume 1 Building Virtual Ecosystems



VERSION 3.3

WES HINSLEY
Imperial College, London

SECOND EDITION
3rd DECEMBER 2007

www.virtualecology.org

Foreword

The Virtual Ecology Workbench (VEW) is a software tool that makes it easy for biological oceanographers to create one-dimensional mathematical simulations of the plankton ecosystem in a mid-ocean mesocosm. It uses the Lagrangian Ensemble metamodel (Woods 2005), which computes emergent demography and biofeedback from individual-based models comprising phenotypic equations for the biological functions of each species in the modelled plankton community. The mathematical simulations created by the VEW are globally stable; each adjusts to an attractor that is independent of initial conditions (Woods et al 2005). That makes such simulations useful for What-if? Prediction.

The Virtual Ecology Workbench is designed to be used by scientists who have no skills in computer programming; the VEW automatically writes Java code from phenotypic equations written in familiar form. Nor does the user need to be skilled in ecosystem modelling: the VEW automatically manages such tasks as chemical budgeting.

The Virtual Ecology Workbench empowers:

- the marine biologist to explore the ecological consequences of phenotypic equations derived from laboratory experiments
- the biological oceanographer to test hypotheses framed to explain observations at sea.
- the university teacher to prepare demonstrations and student exercises on the paradigms of biological oceanography.

The Lagrangian Ensemble metamodel used in the VEW provides a mature method for applying individual-based modelling to plankton ecology. It computes as emergent properties the two quintessential properties of ecosystems: demography of each species, and biofeedback to the environment. To do so the simulations describe the life history of every individual plankter in the virtual ecosystem. That is done by agent-based modelling, in which each computer agent behaves like an individual plankter, while carrying information about a dynamic sub-population of identical plankters. Every plankter in the ecosystem is contained in one of these sub-populations. The accuracy of the emergent demography and biofeedback depends on the number of agents used in the computation. The number is controlled automatically by splitting and merging sub-populations according to the user's specification.

A virtual ecosystem created by the LE metamodel features all the emergent properties simulated by classical population-based modelling, including profiles of environment variables and the demography of each population. It also features properties that classical modelling cannot produce, notably the life history of every plankton in the virtual ecosystem, extending through the generations for each lineage. These life histories provide the ultimate diagnostic information for explaining features in the environment and populations in terms of the basic laws of physics, chemistry and biology contained in the model equations.

The Lagrangian Ensemble metamodel has been refined over thirty years. The R&D programme has resolved all the challenges posed in attempting to simulate a plankton ecosystem in which the actions of individual plankton combine to modify the demography of each population and the physics and chemistry of the environment as the simulation proceeds. The capability of the LE metamodel has been demonstrated across a wide range of applications, including:

- re-analysis of the key paradigms of biological oceanography, for example revising Sverdrup's (1953) description of the onset of the spring bloom to take account of the diurnal thermocline;
- geographically-Lagrangian simulation showed how the geographical zone of permanent oligotrophy is created by the ecosystem drifting into the region where annual heating by the sun exceeds cooling to the atmosphere.
- natural selection among competing species to reveal, as emergent properties, the ranking of competitive advantage, best-fitted species, and extinction times;
- demographic consequences of infection by viruses, with explicit simulation of the viruses as they pass between the water and host plankton;
- mutation of parameters in the phenotypic equations describing the biological functions of plankton with transmission through lineages spreads biomass through multi-dimensional parameter space, where natural selection reduces the abundance of alleles that are poorly fitted to the ecosystem.

John Woods, April 2007

Information about this version: VEW 3.3

Listed below are the significant changes from the last published version of the VEW, which was version 3.1. In short, two major updates have occurred since then; the first (VEW 3.2) allowed jobs to be launched on remote computers using secure shell. This was created for, and tested on Imperial College's CX1 cluster, and should be customisable for any cluster that supports Java, a suitable job queue, and SSH access. VEW 3.3 includes a substantially improved version of the LiveSim model visualiser. In more detail, the difference between VEW 3.3 and VEW 3.1 are as follows.

- Added mixed-layer average number of agents as an option in particle manager.
- Added facility to initialise taller (more than 500m) mesocosms in initial conditions panel. Turbocline will therefore be able to go deeper without causing exceptions.
- Added facility to optionally reset chemical recycling statistics after each recycling event.
- Added a method for adding temperature events in a way that affects the model rules, but does not cause other consequences to the physics of the column (eg, the turbocline compensating for the temperature change).
- Also events that change over depth can now include an interpolated change, allowing for example shallower water to be affected by a single event more significantly than deeper water.
- Output options now allow logging of ingested chemistry – for mesocosm, field plots and ambient audit trail data. You can log which chemical was ingested *by a species/state*, and optionally from which *species/state* the chemical was ingested.
- New job launching interface supports SSH, with user-customisable scripts. Progress monitoring and job control is supporting both locally, and remotely.
- As part of the above interface, the maximum memory (heap size) can now be customised for both local and remote jobs.
- LiveSim completely rewritten with tab-panels to organise the different variables, which for some models were numbering thousands of variables. Additionally a new vertical depth graph, including scale (automatic or customisable), and customisable depth range is now available, offering many more graphs than previously available. These graphs may be compound like analyser (eg, summing the concentration of a number of stages).
- Various improvements to the format of Analyser graphs exported in CSV format.
- New library used for Analyser exports in PostScript format, which now works reliably on MAC.
- Analyser progress windows are now fixed so they don't persist after 100% on MAC and Linux.
- A number of other minor interface glitches have been resolved, and many more performance tweaks for the compiled simulation. Additionally, the code is now entirely generated by the system, rather than by using a default kernel and a linker. This gives room for more advanced optimisation in the future.

1. INTRODUCTION	9
1.1 INSTALLATION NOTES FOR WINDOWS	9
1.1.1 <i>Downloading and Installing Java (JDK)</i>	9
1.1.2 <i>Adding Java to the System Path</i>	9
1.1.3 <i>Enabling Java Server mode</i>	10
1.1.4 <i>Copy Tools.jar library to VEW</i>	10
1.1.5 <i>Compiling and running the VEW</i>	10
1.1.6 <i>Upgrading Java at a later date</i>	10
1.2 THE FRONT PAGE	11
2. THE MODEL SCREEN	12
2.1 CREATING A NEW, EMPTY, MODEL.....	12
2.2 ADDING OR EDITING AUTHOR INFORMATION AND COMMENTS.....	12
2.3 IMPORTING A MODEL.....	13
2.4 RENAMING A MODEL.....	13
2.5 DELETING A MODEL.....	13
2.6 OPENING A MODEL.....	13
3. THE VERSION SCREEN	14
3.1 CREATING A NEW VERSION.....	14
3.2 ADDING OR EDITING AUTHOR INFORMATION AND COMMENTS.....	14
3.3 IMPORTING A VERSION.....	15
3.4 RENAMING A VERSION.....	15
3.5 DELETING A VERSION.....	15
3.6 OPENING A VERSION.....	15
4. THE VEW CONTROLLER DESIGN SCREEN	16
4.1 SAVING YOUR WORK.....	17
4.2 GREY LIGHT.....	17
4.3 YELLOW LIGHT.....	17
4.4 RED LIGHT.....	17
4.5 GREEN LIGHT.....	17
5. THE PLANKTON COMMUNITY – FUNCTIONAL GROUPS	18
5.1 CREATING A NEW FUNCTIONAL GROUP.....	19
5.2 ADDING A COPY OF A FUNCTIONAL GROUP.....	19
5.3 RENAMING A FUNCTIONAL GROUP.....	20
5.4 CHANGING THE ORDERING OF FUNCTIONAL GROUPS.....	20
5.5 DELETING A FUNCTIONAL GROUP.....	20
5.6 SETTING A FUNCTIONAL GROUP TO BE A TOP PREDATOR.....	20
6. CHEMICALS AND PIGMENTS	21
6.1 CREATING A NEW CHEMICAL.....	21
6.2 GIVING A CHEMICAL PIGMENT PROPERTIES.....	22
6.3 IMPORTING CHLOROPHYLL FROM THE ARCHIVE.....	23
6.4 MAKING A COPY OF A CHEMICAL.....	23
6.5 RENAMING A CHEMICAL.....	24
6.6 CHANGING THE ORDERING OF CHEMICALS.....	24
6.7 DELETING A CHEMICAL.....	24
7. AN INTRODUCTION TO FUNCTIONS	25
7.1 CREATING A NEW FUNCTION.....	25
7.2 ADDING A COPY OF A FUNCTION.....	26
7.3 RENAMING A FUNCTION.....	26
7.4 CHANGING THE ORDERING OF FUNCTIONS.....	26
7.5 DELETING A FUNCTION.....	27
7.6 EDITING THE RULES OF A FUNCTION.....	27

8. STAGES (OR STATES)	28
8.1 THE STAGE INTERFACE	28
8.2 SWITCHING FUNCTIONS ON AND OFF FOR EACH STAGE.....	29
8.3 ADDING A NEW STAGE	29
8.4 REMOVING A STAGE	29
8.5 RENAMING A STAGE	29
9. WRITING RULES	30
9.1 THREE GOLDEN RULES	30
9.1.1 <i>Rules are timestep-Based</i>	30
9.1.2 <i>Some variables are timestep-based</i>	30
9.1.3 <i>Rules are individual-based</i>	31
9.2 INTRODUCTION TO THE RULE EDITOR	31
9.3 EXAMPLE 1: A SIMPLE MOTION RULE.....	32
9.4 EXAMPLE 2: FIXING THE ERROR IN EXAMPLE 1.....	37
9.5 EXAMPLE 3: NUTRIENT UPTAKE	38
9.5.1 <i>Local Variables and Ordering</i>	38
9.5.2 <i>The Uptake Statement</i>	39
9.5.3 <i>Building the Rules</i>	39
9.5.4 <i>A note about the units of chemicals</i>	40
9.5.5 <i>Updating the Chemical Pool</i>	41
9.5.6 <i>Chemical concentration in solution</i>	41
9.6 EXAMPLE 4: RELEASING CHEMICALS TO SOLUTION.....	41
9.6.1 <i>Using one rule to set a state variable</i>	42
9.6.2 <i>The release statement</i>	42
9.6.3 <i>Building the rules</i>	42
9.7 EXAMPLE 5: GROWTH BY DIVISION.....	43
9.7.1 <i>The divide statement</i>	43
9.7.2 <i>The ifthen statement</i>	43
9.7.3 <i>Building the rules</i>	44
9.7.4 <i>Choice of functions</i>	44
9.7.5 <i>Important note for releasing chemicals</i>	44
9.8 EXAMPLE 6: CREATING NEW INDIVIDUALS	45
9.8.1 <i>The Create Statement</i>	45
9.8.2 <i>Building the Rules</i>	45
9.8.3 <i>More about the Set statement</i>	46
9.8.4 <i>Chemical Budget reminder</i>	46
9.9 EXAMPLE 7: CHANGES OF STAGE.....	46
9.9.1 <i>The Change Statement</i>	47
9.9.2 <i>Ordering of Rules with Stage Changes</i>	47
9.9.3 <i>Building the Rules</i>	47
9.10 EXAMPLE 8: PROBABILISTIC CHANGES OF STAGE	48
9.10.1 <i>The PChange Statement</i>	48
9.10.2 <i>Particle Management Issues</i>	48
9.10.3 <i>Building the Rules</i>	49
9.11 EXAMPLE 9: INTEGRATING OVER DEPTH.....	49
9.11.1 <i>The integrate Function</i>	49
9.11.2 <i>Variables can have a history</i>	50
9.11.3 <i>Building the rule</i>	50
9.12 EXAMPLE 10: INGESTION.....	51
9.12.1 <i>The Food-Set Variable Type – “The Diet”</i>	51
9.12.2 <i>Food-Based Locals, State Variables and Parameters</i>	52
9.12.3 <i>Reducing food-based variables down to scalars</i>	52
9.12.4 <i>Food-based conditional statements</i>	52
9.12.5 <i>Mixing food-based and non-food-based terms</i>	53
9.12.6 <i>Use of multiple food-sets</i>	53
9.12.7 <i>The ingest statement</i>	53
9.12.8 <i>Predator and Prey Interaction</i>	54
9.12.9 <i>How much did I eat?</i>	54

9.12.10	<i>Building the rules</i>	55
9.13	THE SYNTAX CHECKING WINDOW.....	56
9.13.1	<i>Duplicate Writes</i>	56
9.13.2	<i>Variable Written and not used</i>	56
9.13.3	<i>Variable Read Before Write</i>	56
10.	MODELLING LANGUAGE QUICK REFERENCE.....	57
10.1	STATEMENTS.....	57
10.3	BOOLEAN FUNCTIONS.....	60
11.	SPECIES BUILDER.....	61
11.1	CREATING A NEW SPECIES, FROM A FUNCTIONAL GROUP.....	62
11.2	CREATING A NEW SPECIES, BASED ON AN EXISTING SPECIES.....	62
11.3	CREATING A NEW SPECIES USING ALLOMETRY.....	62
11.4	CREATING AN ALLOMETRIC SET OF SPECIES.....	63
11.5	EDITING AN EXISTING SPECIES.....	63
11.6	RENAMING A SPECIES.....	63
11.7	REMOVING A SPECIES.....	63
12.	FOOD-SETS.....	64
12.1	CHOOSING THE FOOD THAT A PREDATOR WILL EAT.....	65
12.2	SETTING THE FOOD-SPECIFIC PARAMETERS.....	65
12.3	SETTING ALL THE FOOD-SPECIFIC PARAMETERS TO THE SAME VALUE.....	65
13.	MESOCOSM TRACK.....	66
13.1	SETTING THE TIMES OF SIMULATION.....	67
13.2	NAVIGATING AND CHOOSING A STARTING POINT ON THE MAP.....	67
13.3	CHOOSING THE TYPE OF INTEGRATION: FIXED, FORWARDS, OR BACKWARDS.....	68
13.4	CHOOSING THE DATA SET FOR THE BOUNDARY CONDITIONS.....	69
14.	VISUALISING THE DATASETS.....	70
14.1	BASIC VISUALISATION.....	70
14.2	OVERLAYING LANDMASSES.....	70
14.3	VISUALISING LAYERED DATA.....	71
14.4	VISUALISING VECTOR DATA.....	71
15.	PARTICLE MANAGER.....	73
15.1	ADDING A NEW SPLIT RULE.....	74
15.2	ADDING A NEW MERGE RULE.....	74
15.3	ADDING A NEW MAINTAIN RULE.....	75
15.4	EDITING EXISTING RULES.....	75
15.5	ADDING A RULE BASED ANOTHER ONE.....	75
15.6	REMOVING RULES.....	75
15.7	A NOTE OF CAUTION REGARDING PARTICLE MANAGEMENT.....	76
16.	INITIALISING THE MESOCOSM.....	77
16.2	EDITING THE DATA MANUALLY.....	78
16.3	INTERPOLATING BETWEEN VALUES.....	78
16.4	SETTING THE MAXIMUM DEPTH OF THE MESOCOSM.....	78
17.	DISTRIBUTIONS OF PLANKTON.....	79
17.1	ADDING A NEW DISTRIBUTION OF PLANKTON.....	80
17.2	EDITING AN EXISTING DISTRIBUTION (STANDARD METHOD).....	80
17.3	EDITING AN EXISTING DISTRIBUTION (QUICKER METHOD).....	80
17.4	USING AN EXISTING DISTRIBUTION TO CREATE A NEW ONE.....	81
17.5	REMOVING A DISTRIBUTION.....	81
17.6	A NOTE ABOUT TOP PREDATORS.....	81
18.	TROPHIC CLOSURE.....	82
18.1	DEFINING THE TROPHIC CLOSURE, STEP-BY-STEP.....	83

18.2	THE TOTAL CONCENTRATION RULE, N_T	83
18.3	THE VERTICAL DISTRIBUTION RULE, D_T	83
18.4	THE SIZE RULE, S_T	83
19.	CHEMICAL RECYCLING.....	84
19.1	DEFINING THE CHEMICAL RECYCLING, STEP-BY-STEP	85
20.	EVENTS	86
20.1	ADDING A NEW EVENT (E.G., CHEMICAL EVENT)	87
20.2	EDITING AN EVENT	87
20.3	CREATING A NEW EVENT BASED ON AN EXISTING EVENT	88
20.4	REMOVING AN EVENT	88
20.5	PHYSICAL EVENTS	88
21.	LOGGING	89
21.1	VARIABLES THAT CAN BE LOGGED	90
21.2	ADDING VARIABLES TO BE LOGGED	91
21.3	PREVENTING PREVIOUSLY SELECTED VARIABLES FROM BEING LOGGED	91
21.4	WINDOWING THE LOGGING BETWEEN TWO TIMES, AT A GIVEN FREQUENCY	91
21.5	WINDOWING THE LOGGING BETWEEN TWO DEPTHS.....	92
21.6	SETTING THE STATES FOR AUDIT TRAILS	92
21.7	LINEAGE AND LIFESPAN LOGGING.....	92
21.8	ACTIVATING THE RIGHT LOGGING OUTPUT FILES	92
22.	BUILDING JOBS	93
22.1	SETTING ONE OR MORE RANDOM SEEDS.....	94
22.2	ADDING A BATCH OF JOBS.....	94
22.3	REMOVING A BATCH.....	95
22.4	SPECIFYING WHETHER CHECKPOINTS SHOULD BE WRITTEN	95
22.5	SPECIFYING WHETHER THE SIMULATION SHOULD START AT A CHECKPOINT	95
22.6	SUBMITTING THE JOB AND CHOOSING THE RESULTS LOCATION.....	96
23.	THE JOB CONTROL WINDOW.....	97
23.1	JOB STATUS.....	98
23.2	LAUNCHING A JOB ON A REMOTE COMPUTER – CX1 EXAMPLE.	99
23.3	NOTES FOR SETTING UP OTHER REMOTE COMPUTERS.	100
23.4	LAUNCHING A JOB ON YOUR OWN COMPUTER.	100
23.5	JOB CONTROL.....	101
23.6	LAUNCHING AN INTERACTIVE JOB	101
24.	LIVESIM – THE INTERACTIVE SIMULATION.....	102
24.1	CHOOSING AN AGENT TO WATCH.....	102
24.2	THE DATA WINDOW	103
24.3	DEPTH PROFILES	104
24.3.1	<i>Choosing the depth profile.....</i>	<i>104</i>
24.3.2	<i>Scaling the depth profile.....</i>	<i>105</i>
24.3.3	<i>Choosing the depth range.....</i>	<i>105</i>
24.4	ADVANCING THE SIMULATION	105

1. Introduction

This handbook describes each section of the main interface of the Virtual Ecology Workbench, version 3.3, December 2007. It covers the processes required starting from species builder, and finishing with launching the job. Each section consists of an overview to that component of the VEW – experienced users may find everything they need in this summary. Following that are instructions detailing how to perform particular common tasks for that section. But first, some instructions regarding installation of the VEW.

1.1 Installation Notes for Windows

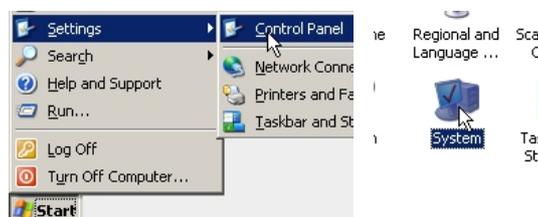
For this section, it is assumed that you have already obtained the latest version of the VEW, and unzipped it into a directory. Please contact us to get the latest version, and the large accompanying data files, which cannot be downloaded at this stage.

1.1.1 Downloading and Installing Java (JDK)

VEW will run on any Java platform from version 1.4 onwards, but we recommend always using the latest version of the Java Development Kit, (JDK), from Sun. At December 1st 2007, the latest version is 1.6.0, update 3, available from <http://java.sun.com>, then click on “Java SE”. Versions are available for all versions of Windows since Windows 2000, and special 64-bit versions are available if you have 64-bit hardware, and either Windows XP-64, or Vista. Java is supplied for Windows with a simple automatic installer, which will not require any settings changed for a successful install. After installing Java, there are a couple of extra steps required to enable the VEW to run.

1.1.2 Adding Java to the System Path

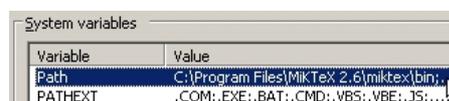
VEW needs your system to know where to find some critical Java files – unfortunately, this is not done automatically by the Sun installer. To update the path, click on Start, Settings, and then Control Panel. Then click on the System icon.



Next, click on the Advanced tab, and then on the environment variables button.



On the next screen, look in the System variables table for a variable called “Path”, and double-click on it to edit. If you installed Java version 1.6.0 update 3, then you need to ensure that *C:\Program Files\Java\jdk1.6.0_03\bin* is part of this line of text. Note that different directories are separated by semi-colons (;). This directory is the default installation directory that the Java installer suggests.



If there are any other entries in the path referring to older Java version, these should be deleted, to ensure that VEW uses the latest Java version. See the notes later regarding updating your Java version.

After that, this part is completed, and you should click OK until all the windows go away!

1.1.3 Enabling Java Server mode

VEW Simulations can be made to perform considerably faster by enabling Java server mode. This, also unfortunately, is not possible with the default installation from Sun, but it can be very easily enabled. Use Windows explorer to browse to the location where JDK was installed – usually C:\Program Files\Java\jdk1.6.0_03. Then move into the directory called *jre*, then *bin*, and then right-click on the *Server* directory, and choose “Copy”.



Now move back three levels, and into the directory where the *Java Runtime Environment (JRE)* was installed – usually C:\Program Files\Java\jre1.6.0_03. Again, click in *bin*, and then right click and choose “Paste”. Java will then recognise Server mode, and the VEW will then perform well when you choose Server mode in the Job Control section described in section 23.



1.1.4 Copy Tools.jar library to VEW.

Tools.jar is a file needed for the VEW to use Java’s own compiler, for building VEW jobs. It needs to be copied from the directory C:\Program Files\Java\jdk1.6.0_03\lib into the VEW install directory, the same directory where you will find various other ‘jar’ files, such as *trove.jar*. Use the process for copying files described in 1.1.3.

1.1.5 Compiling and running the VEW.

The VEW needs compiling with the Java version installed on your machine. To do this on Windows, open Windows Explorer, and browse to the directory where you installed the VEW. Double click on the file *c.bat*, and the VEW will be compiled. You will only need to repeat this step if you have updated to a new version of the VEW, or to a new version of Java.

To run the VEW, double click on *vev.bat*.

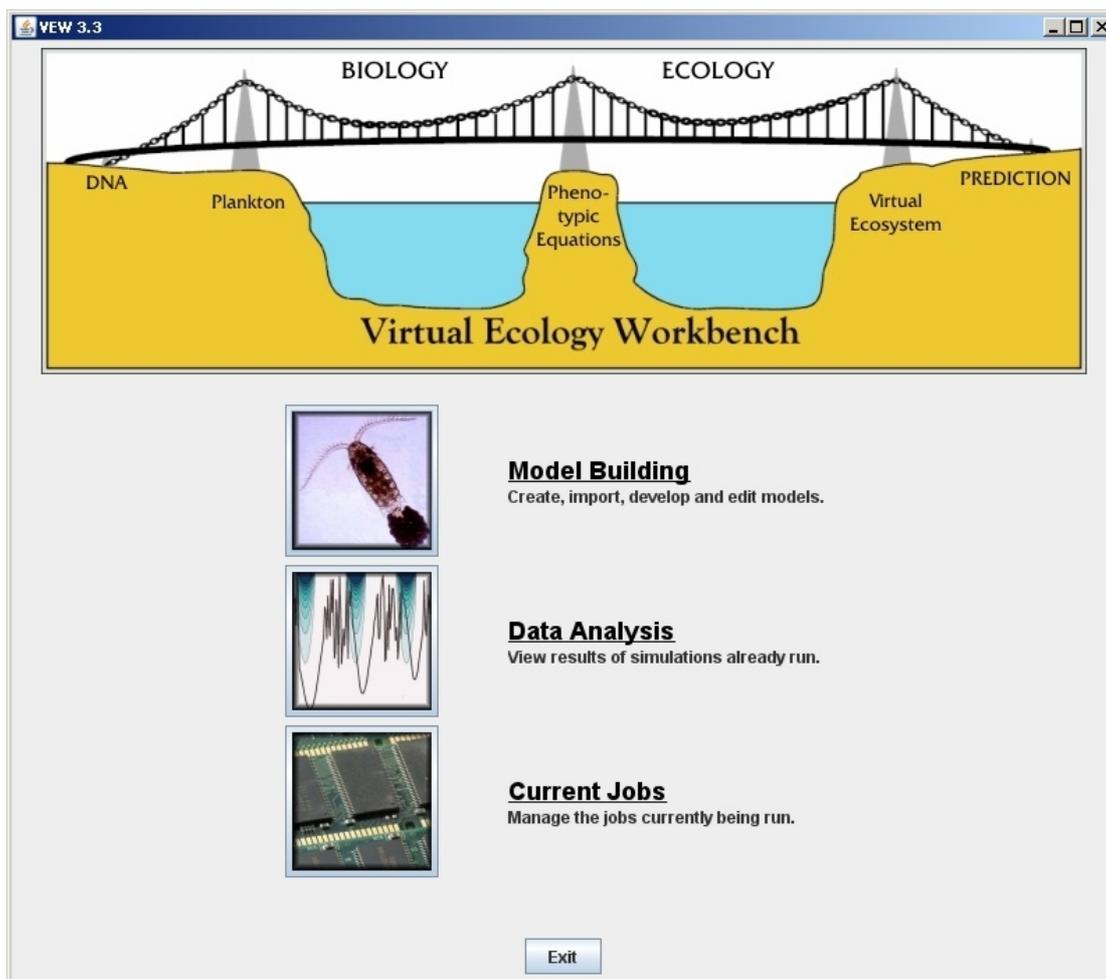
1.1.6 Upgrading Java at a later date.

Java updates are released periodically – usually around 2 or 3 months apart. To ensure that the VEW is running with that version, firstly go to Control Panel, and choose Add/Remove programs. Locate any versions of the Java Runtime Environment (JRE), and the Java Development Kit, and *Remove* them.



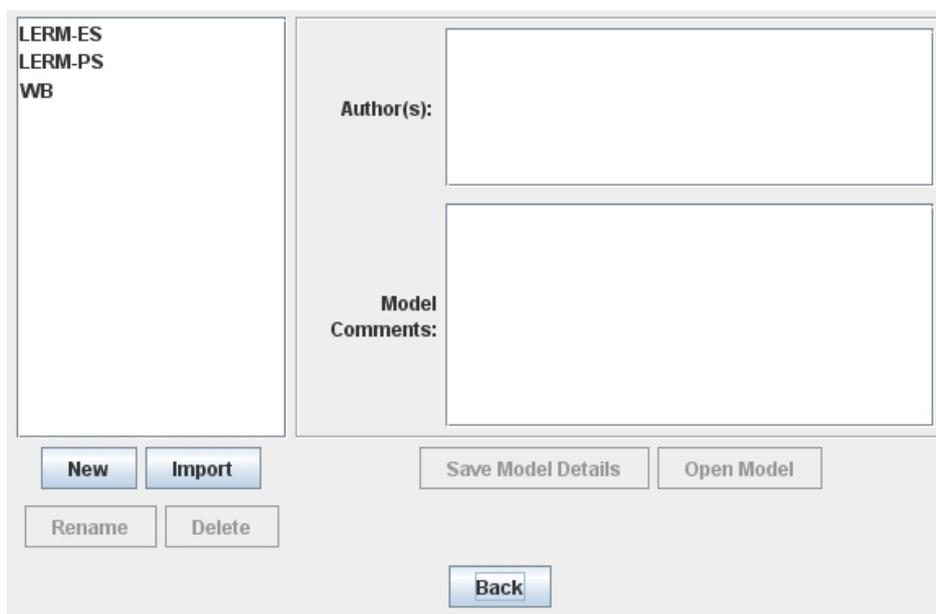
Then follow the steps from 1.1.1 to install the new version of Java, making sure that you remove any old versions from the path in section 1.1.2.

1.2 The Front Page



When you first run VEW 3.3, you are given the menu shown above. The **Model Building** section is the focus of this volume, in which we will describe the entire process of building an executable virtual ecosystem. Having done so, we will be able to launch a job, and watch its progress using the **Current Jobs** page. Having executed a job, we can analyse the results, from the **Data Analysis** page. For now, we concern ourselves with building models, and clicking on the icon, or the text for **Model Building** allows us to create new virtual ecosystems from scratch, or by adapting existing virtual ecosystems.

2. The Model Screen



The model chooser shows the list of models currently available to you. If you click on a model, any provided information about the author(s), and the content of the model will appear in the two text boxes to the right of the page. Also, having clicked a model, the buttons to open, rename or delete the model become available. You will also be able to update the author and comment information, which causes the Save Model Details button to become available. The Back button returns you to the title page.

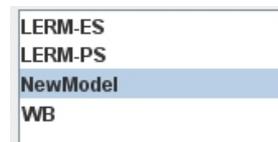
A model, for the purpose of this screen, should be thought of as a significant project. After creating a model, you can create different versions of that model, which can be considered as versions in development or different experiments based on the same model.

2.1 Creating a new, empty, model.

1. Click on **New**. A dialog appears asking you to name the new model. Type the name for your new model, and click on **OK**.

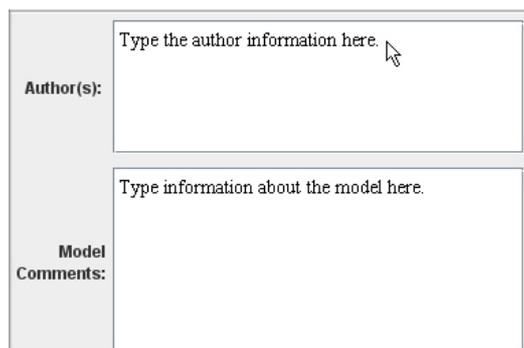


2. The new model will appear in the list of models. It is good practice at this stage to also type any information about the author, and comments about the model later, into the two text boxes provided. See the next section.



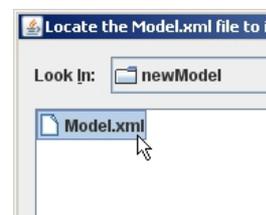
2.2 Adding or editing author information and comments.

1. After creating, importing (see later), or clicking on a model name, the author and comments information appear in the text boxes on the right. Click in these boxes and edit the text.
2. You can save your changes by clicking on **Save Model Details**.



2.3 Importing a model

1. If you have been sent a model, by email for example, that was created with the VEW, (since 3.0), you can import it into the VEW running on your computer. In the case of models taken from 3.0, several items will require your attention over the next few chapters, as VEW will automatically update various aspects of the model.
2. To import a model, click on . A file browser window will open, and you must locate the model to be imported; it will be called **Model.xml**.
3. You will then be asked to name the imported model, and it will be added to the list, just like creating a new model.



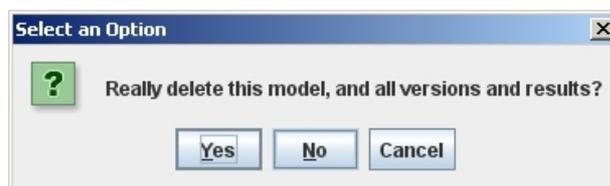
2.4 Renaming a model

1. Select the model in the list, and click on .
2. Type the new name for the model in the text box to rename the model.



2.5 Deleting a model

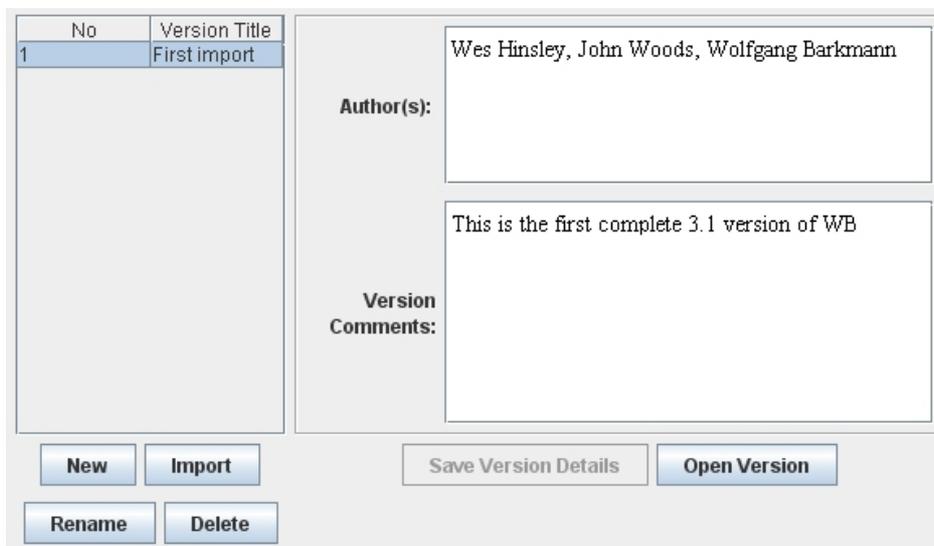
1. Deleting a model will permanently remove all versions of the model, and any results that were created within the folder for that model. Therefore, use this button with care.
2. Select the model to delete from the list, and then click on . The warning dialog appears, and you must confirm that you really want to delete the model.



2.6 Opening a model.

1. To open a model to view it, edit it or run jobs with it, select the model in the list and click on . You will be taken to the version screen, described next.

3. The Version Screen



The version chooser is very similar to the model chooser. In practice, you may find it takes several iterations of design before your model is complete, and having completed it, you may want to create more adaptations of the model. The version page is for this purpose. As in the model page, each version has author information, and space for comments. Versions can be renamed and deleted, and furthermore, a version can be created based on another version.

3.1 Creating a new version.

1. If you have just created or imported a model, then a first version of that model will already have been created. Otherwise, you always create new versions based on a previous version. Click on the version you would like to use as a template.

No	Version Title
1	First import

2. Click on **New**. A dialog appears asking you to name the new version. Type the name for the version and click on **OK**.

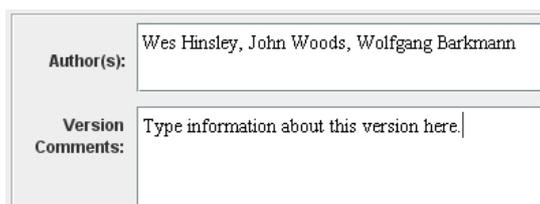


3. The new version will appear in the list of models.

No	Version Title
1	First import
2	New Version

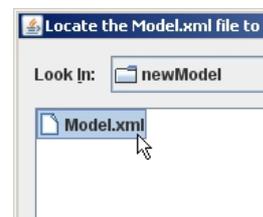
3.2 Adding or editing author information and comments.

1. As before, it is good practice to add author and comment information for your new version. Click in the boxes and edit the text.
2. You can save your changes by clicking on **Save Version Details**.



3.3 Importing a version

1. If you have been sent a VEW model, by email for example, you can also import it into the VEW as a new version, rather than as a new model, which was described in the previous chapter.
2. Click on . A file browser window will open, and you must locate the model file to be imported; it will be called **Model.xml**.
3. The import will create a new version, which you must name.



3.4 Renaming a version.

1. Select the version in the list, and click on .
2. Type the new name for the version in the text box.



3.5 Deleting a version

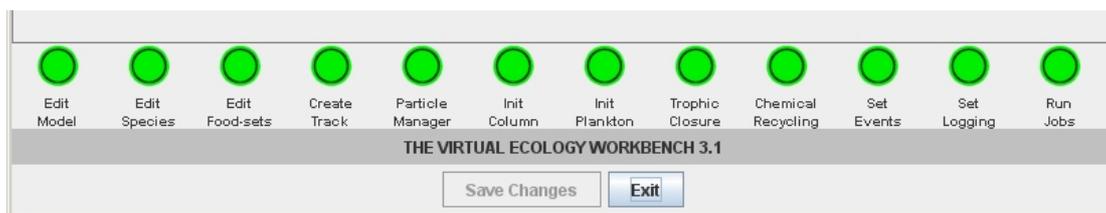
1. Click on the version to be deleted, and then click on .
2. You will be asked to confirm deletion of the version, and any results that were stored in the default locations for that version.



3.6 Opening a version.

1. To open the version currently selected, click on . This will open the main VEW Controller design screen, where we design the virtual ecosystem.

4. The VEW Controller Design Screen



A virtual ecosystem is built in twelve stages, shown by the twelve lights on the interface. Clicking on any of these opens that part of the VEW in the main portion of the screen. The twelve tasks required are shown below.

No	Title	Description
1	Edit Model	Define the functional groups and states, and the behaviour of plankton using phenotypic rules and 8 special VEW functions. This includes the behaviour of top predators. Also define chemicals and pigments.
2	Edit Species	Define the species, separately or allometric sets.
3	Edit Food-sets	For all ingestion rules, define predator-prey relationships, and parameter values that are food-specific.
4	Create Track	Set the start and end date of the simulation and the track for the mesocosm. Also choose the data-set to use for the climate, and preview the data-set on a world-map.
5	Particle Manager	Control demographic noise by adding conditional rules for splitting and merging agents, or maintaining the number of agents when handling state changes.
6	Initialise Column	Initialise the physical and chemical profiles for the column at the start of the simulation, using Levitus and NOAA data as a template, where available.
7	Initialise Plankton	Specify the abundance of plankton of different species, and set their initial biological properties, at the beginning of the simulation, or define the introduction of plankton at a later point (called a biological event)
8	Trophic Closure	Specify the exogenous rules for top predators – total concentration, size and depth distribution.
9	Chemical Recycling	For experiments that require a stationary annual cycle, identify chemicals that will be lost from the mixed layer by sedimentation, and make a mathematical adjustment to return them.
10	Set Events	For “What-if?” predictions, force changes at a given time concerning the chemistry throughout the mesocosm and the climate conditions.
11	Set Logging	Specify the emergent properties to log, including column totals, field data and audit trails, windowed where applicable by space and time. Also select whether parish register (family tree) is required.
12	Run Jobs	Specify random seed(s) for jobs. You can also create a batch of runs with some parameter value varied for each. Checkpoint files can be written which allow restarting interrupted jobs (eg, after a power-cut), or creating a stable starting point for future runs. Finally submit the job(s) for execution and choose where output data should be stored.

4.1 Saving your work

VEW has a single save button, which will save all of the changes you have made. It is good practise to do this regularly.



4.2 Grey Light

VEW automatically checks the consistency of your model. You can break the consistency in a large number of ways. For example, if having created some species, specify that you would like the simulation to create audit trails for one of the species, and then later you delete that species, an inconsistency will have occurred. This will happen frequently, and VEW 3.1 will automatically detect these. The consistency status of your model is indicated by the lights along the bottom of the page.

Grey Lights occur in the early stages of a model, where a page is not yet available, because there is some pre-requisite that has not yet been defined yet. In the example below, the dates of the simulation must be specified, before various date-dependent other features can be configured.



4.3 Yellow Light

The **Yellow Light** indicates the part of the VEW you are currently working on.



4.4 Red Light

A red light indicates a problem that requires your attention. In this example, the inconsistency was caused by adding a new species, and VEW required a definition on the logging page, to specify whether audit trails for that species are required. Visiting the logging page in this case would cause VEW to fix the problem using default settings, and a warning message appears telling you what VEW has done automatically, to repair the consistency error.



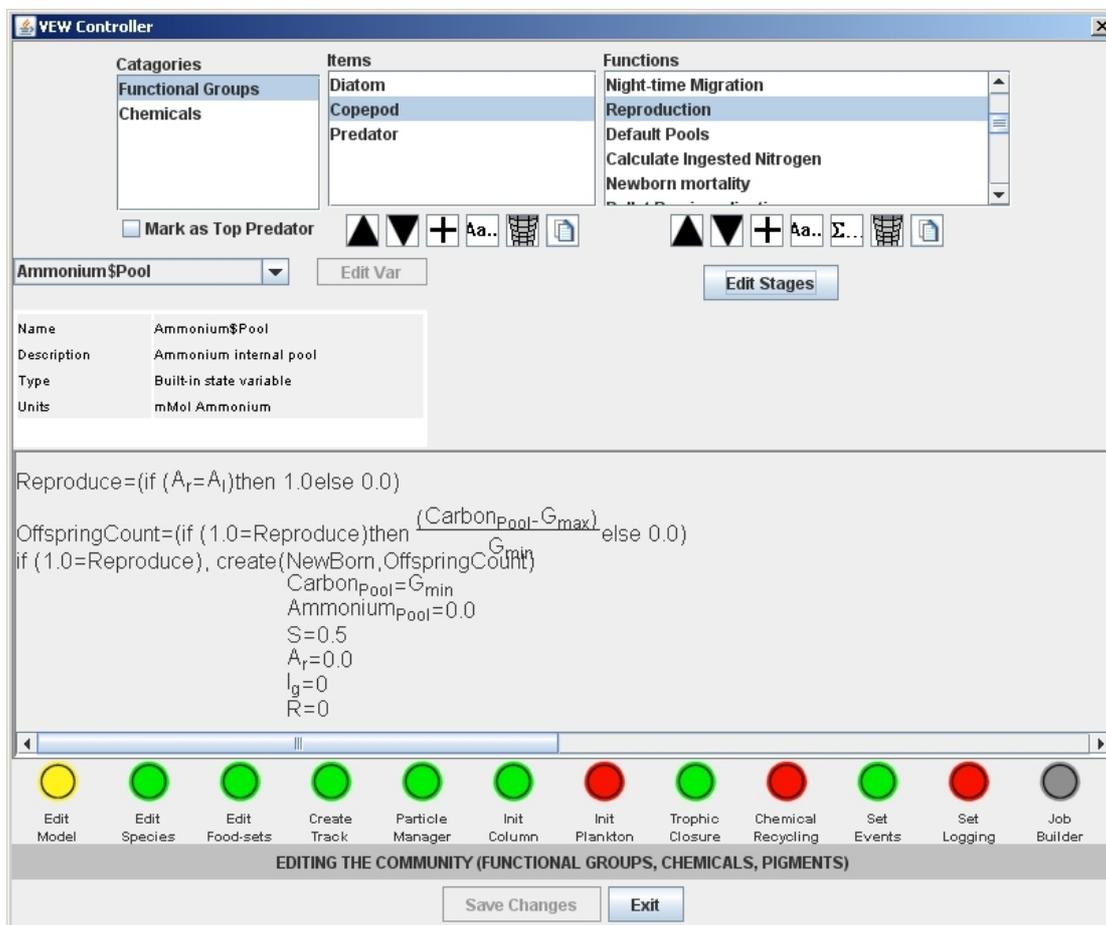
4.5 Green Light

If all the lights are all green, then your model is ready to run.



5. The Plankton Community – Functional Groups

The model builder is the place where the members of the plankton community – the functional groups – are defined. The behaviour is specified function-by-function, and within functions, rule-by-rule. Some functional groups may be defined as “top predators”, which has a special exogenous purpose we will see later. Additionally, the chemicals to be considered in the virtual ecosystem are defined here, along with their pigment properties where applicable.

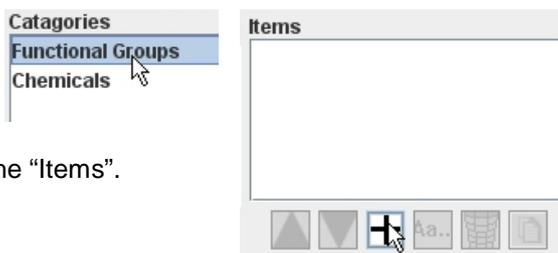


The screenshot above shows the main model builder interface. At the top, you select whether you want to edit a functional group or a chemical from the “Categories” menu. Depending on what you choose, the existing functional groups or chemicals are shown in the “Items” menu. Again, depending on your choice, the existing functions are shown. For functional groups, you will also see a list of stages in the middle right, which can be used to represent growth stages, or a range of other categorisations of a functional group. In the bottom portion of the screen, you see a representation of the function currently highlighted.

The model builder is a large and complicated interface. Never-the-less, users can become fluent through practise, and its current form yields certain benefits in terms of building valid models. However, it is the oldest part of the VEW, and we have plans to improve its ease of use and appearance while also offering further tools to assist in efficiently creating models that do what you intend them to do.

5.1 Creating a new functional group

1. Click on the "Functional Group" category.



2. Click on the plus symbol below the "Items".

3. The import/create screen appears. Importing functional groups is not yet fully supported in VEW, since a master repository of functional groups has not been created. This is planned for the future. To create a new empty functional group, click on 

4. You will be asked to give a name for the new functional group, and the VEW will ensure that the name has not been previously used.



5. The functional group will appear in the list. Note that a stage called "Default" will automatically have been created for you.

5.2 Adding a copy of a functional group

1. Click on the "Functional Group" category.



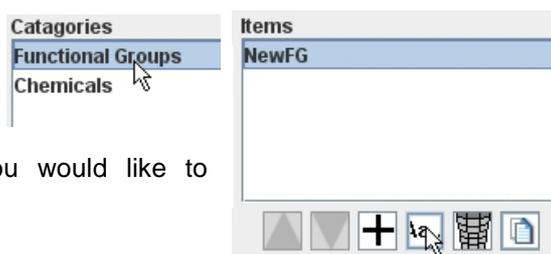
2. Select the functional group you would like to copy, and then click on the copy button: .

3. You will be asked to name the copy of the functional group. VEW will make sure this name is unique. Clicking OK will add an identical copy of the functional group, with the new name.



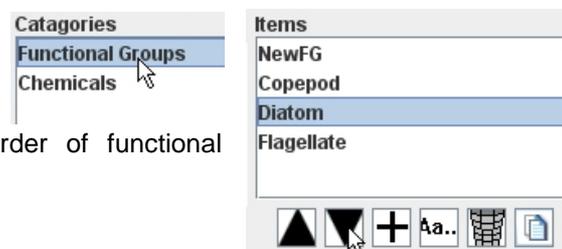
5.3 Renaming a functional group

1. Click on the “Functional Group” category.
2. Select the functional group you would like to rename, and then click on .
3. You will be asked to give a new name for the functional group. VEW will ensure your new name is unique in the system.



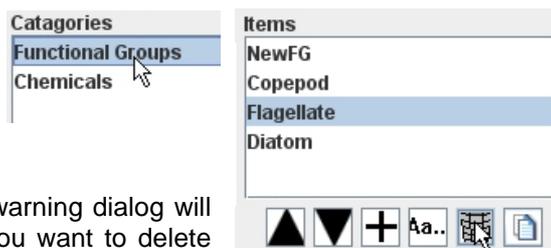
5.4 Changing the ordering of functional groups

1. Changing the order of functional groups in the list makes no difference to the model in any respect, so this section is purely to allow you to arrange the order of functional groups on screen.
2. Select the “Functional Group” category, and then choose the functional group to move from the item list.
3. Click on the arrow icons to move the functional group up or down in the list.



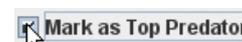
5.5 Deleting a functional group

1. Choose “Functional Groups” from the categories menu, and then select the functional group from the items list.
2. Click on the delete icon: . A warning dialog will appear, asking you to confirm you want to delete this functional group.

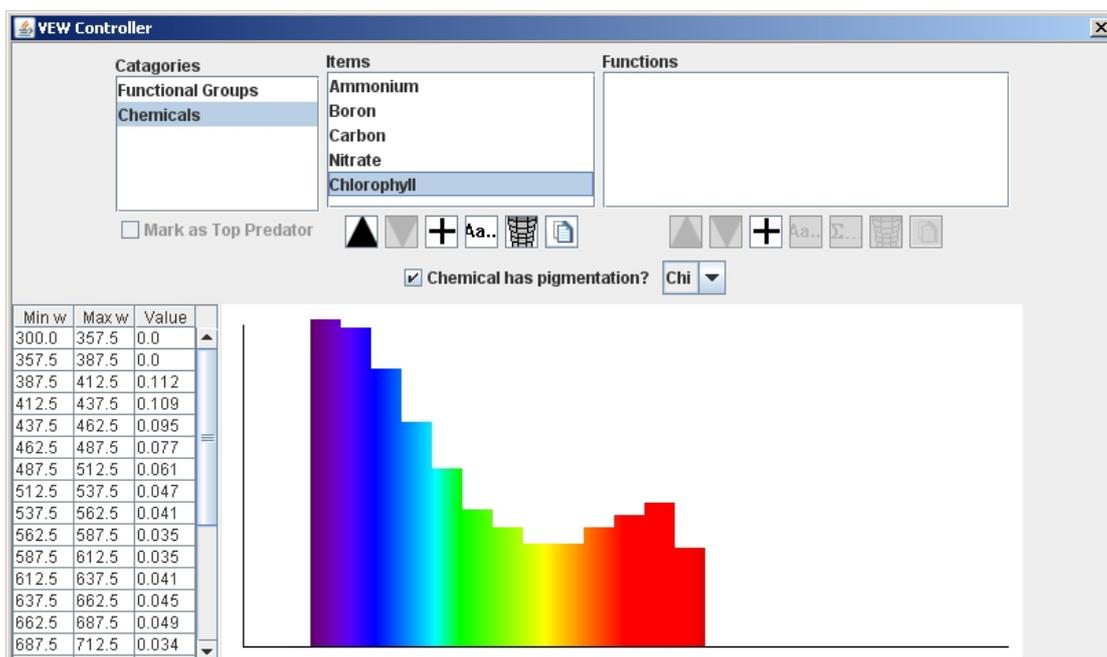


5.6 Setting a functional group to be a top predator

To set the currently selected functional group to be a top predator, click the box next to “Mark as Top Predator”. Later on in the process, VEW will expect you to define the concentration, size and distribution of that functional group, as exogenous properties.



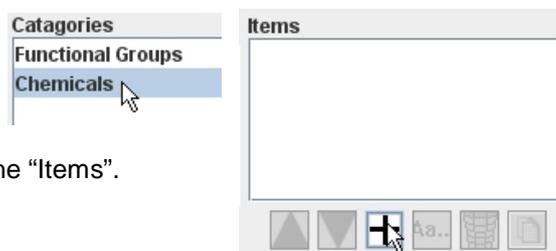
6. Chemicals and Pigments



Chemicals can be added, removed, copied and renamed just like functional groups can. You can create simple functions for chemicals too, but not much of the functionality of VEW’s rule language is applicable to chemicals – see the later chapters on building rules. Additionally, chemicals can be classed as pigments, in which case they have a biofeedback effect on the physics of the mesocosm.

6.1 Creating a new chemical

1. Click on the “Chemicals” category.



2. Click on the plus symbol below the “Items”.

3. The import/create screen appears. For convenience, the chlorophyll chemical, which has defined pigment spectra, can be imported here – details of how to do this follow. For this example, to create a new chemical, click on **Create**.

4. You will be asked to give a name for the new chemical, and the VEW will ensure that the name has not been previously used.



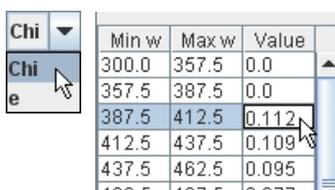
5. The new chemical will appear in the list, and every plankter in your simulation inherits a variable called “NewChemical Pool”, to define how much of each chemical the plankter contains.

6.2 Giving a chemical pigment properties.

- To give a chemical properties for use in biofeedback, firstly tick the “Chemical has pigmentation” tickbox. The pigment display window will appear. 
- The biofeedback is defined by two parameters, where each varies for different wavelengths of light. The two parameters are χ (Chi) (m^2/mg of Pigment), and e. The equation in the physics code that uses these two parameters is as follows:-

$$k^*(\lambda) = \chi_P(\lambda) P^{e_P(\lambda)}$$

This equation is computed at every depth in the water column, for each of 25 wavebands, for each pigment P. The value of P in the equation is the total of a pigment, calculated by summing the pool of that pigment in every individual plankter at that depth.

- Select the parameter to define using the drop down list. Notice that the values in the table will update to show that parameter. The table shows the minimum and maximum wavelength for each band. 
- Edit the values of χ and e, for each wave length by clicking in the “Value” column and typing values. It is not possible to change the values for the wave length boundaries. Notice that the graph will automatically update as you type values.

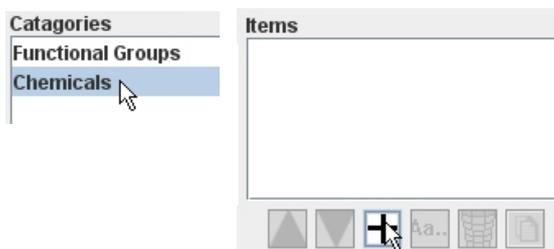
For convenience, below are the sample values for Chlorophyll, obtained by Morel (1988) through regression analysis.

Min λ	Max λ	χ	e
300	357.5	0	1
357.5	387.5	0	1
387.5	412.5	0.112	0.677
412.5	437.5	0.109	0.702
437.5	462.5	0.095	0.702
462.5	487.5	0.077	0.703
487.5	512.5	0.061	0.695
512.5	537.5	0.047	0.673
537.5	562.5	0.041	0.65
562.5	587.5	0.035	0.618
587.5	612.5	0.035	0.628
612.5	637.5	0.041	0.65
637.5	662.5	0.045	0.672

Min λ	Max λ	χ	e
662.5	687.5	0.049	0.687
687.5	712.5	0.034	0.62
712.5	737.5	0	1
737.5	787.5	0	1
787.5	900	0	1
900	1100	0	1
1100	1300	0	1
1300	1500	0	1
1500	1700	0	1
1700	1900	0	1
1900	2100	0	1
2100	2300	0	1

6.3 Importing Chlorophyll from the archive.

1. You may find it useful to import chlorophyll complete with the values from the table above. To do this, click on the "Chemicals" category.



2. Click on the plus symbol below the "Items".

3. The import/create screen appears. The only chemical available in the list is Chlorophyll; click on it.



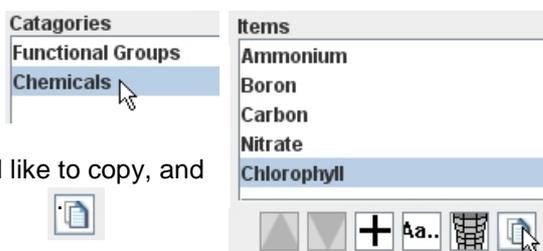
4. Click on **Analyse** - VEW will check the import causes no other problems.

5. Finally, click on **Confirm** to add the default chlorophyll to your model.

Note that the import process here is a new development; its functionality and appearance will be improved in due course.

6.4 Making a copy of a chemical

1. Click on the "Chemicals" category.



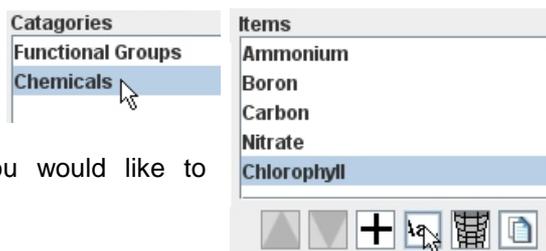
2. Select the chemical you would like to copy, and then click on the copy button:

3. You will be asked to name the copy of the chemical. VEW will make sure this name is unique. Clicking OK will add an identical copy of the chemical, with the new name.



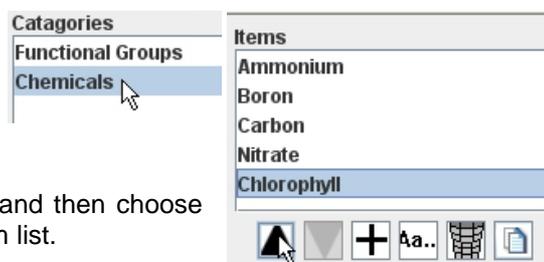
6.5 Renaming a chemical

1. Click on the “Chemicals” category.
2. Select the functional group you would like to rename, and then click on .
3. You will be asked to give a new name for the functional group. VEW will ensure your new name is unique in the system.



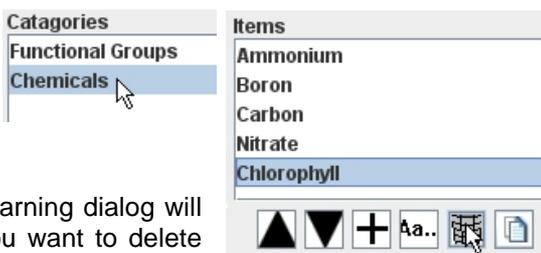
6.6 Changing the ordering of chemicals

1. Changing the order of chemicals makes no difference to the model in any respect, so again, this is purely for your preference.
2. Select the “Chemicals” category, and then choose the chemical to move from the item list.
3. Click on the arrow icons to move the chemical up or down in the list.



6.7 Deleting a chemical

1. Choose “Chemicals” from the categories menu, and then select the chemical from the items list.
2. Click on the delete icon: . A warning dialog will appear, asking you to confirm you want to delete this chemical.



7. An Introduction to Functions

In nature, we observe that plankton tend to be arranged in sets that have the same basic behaviour; these sets we have called functional groups. Now, we are considering the behaviour of each functional group, and trying to neatly separate the behaviour into a range of *functions*, each of which will ideally represent a discrete physiological behaviour.

In the VEW, a function may consist of a number of rules, each of which can be compared to a mathematical equation, or in some cases a simple program statement. Creating the rules is a detailed process which will have several chapters devoted to it later. This section will briefly describe the processes of creating an empty function, renaming functions, deleting functions, making copies of functions, and some notes on re-ordering functions which, differently to the order of functional groups and chemicals **can make a difference in certain circumstances** to the meaning of your biological model.

While these functions are focussed almost entirely on plankton behaviour, you can also create functions for chemicals, which differ substantially from plankton functions in their nature, but the steps described in this chapter apply equally to both. The steps assume that you have already chosen a functional group, (or a chemical), in the "Items" box, as only then do the buttons for working with functions become available.

7.1 Creating a new function

1. Having selected the Category and Item, the function buttons become available. Click on the plus symbol below the Functions menu, to add a new function.



2. The import/create screen appears. Importing functions is not yet fully supported in VEW, since a master repository of functions has not been created. This is planned for the future. To create a new empty function, click on  .

3. You will be asked to give a name for the new function, and the VEW will ensure that the name has not been previously used.



4. The function will appear in the list.

7.2 Adding a copy of a function

1. Select the function you would like to duplicate in the functions list.

2. Click on the copy button:  .

3. You will be asked to name the copy of the function. VEW will make sure this name is unique. Clicking OK will add an identical copy of the function, with the new name.

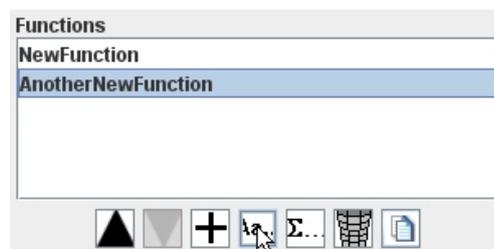


7.3 Renaming a function

1. Click on the function you would like to rename.

2. Click on the rename button:  .

3. You will be asked to give a new name for the function. VEW will ensure your new name is unique. Click OK to rename the function.

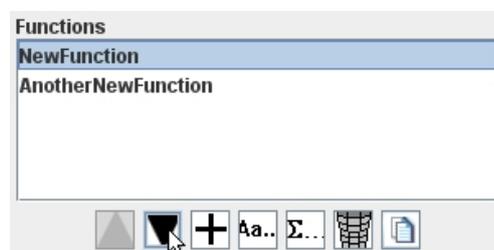


7.4 Changing the ordering of functions

1. Click on the function you would like to move. The arrow buttons will become available.

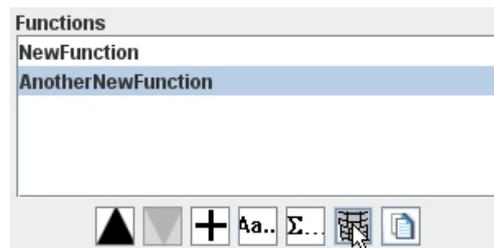
2. Click on the up or down arrow as appropriate.

3. In the section about rules, we will discuss the implications (and sometimes the necessity) of re-ordering functions, which, in summary, can make a difference to the behaviour of your model if the functions being moved contain **local variables**, or **stage changes**.



7.5 Deleting a function

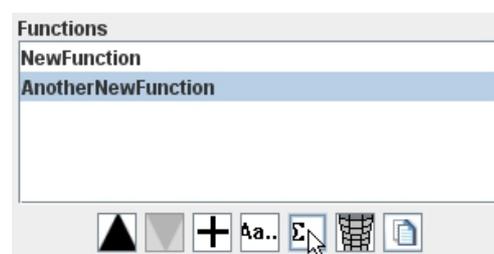
1. Click on the function to be deleted.
2. Click on the delete icon: . A warning dialog will appear, asking you to confirm you want to delete this function.



7.6 Editing the rules of a function

f

1. Click on the function to be edited.
2. Click on the rule editor icon: .
3. The rule editor appears. The process of building rules from here will be described in great detail later.



8. Stages (or States)

Before we discuss making rules, there is one more aspect which needs to be described. Stages serve a number of purposes. Firstly, they allow you to define a functional group that has behaviour switched on or off at particular times, when the plankter is in a certain stage, or state. This encapsulates behaviour that changes throughout a plankter's life, such as different stages of growth, or periods of hibernation.

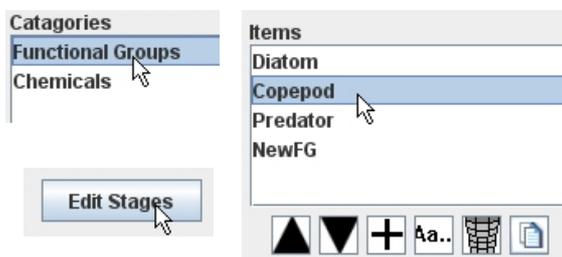
Furthermore, when we define ingestion later, we will be able to distinguish between different stages, and preferentially feed on one stage of a functional group over another. This allows the possibility for a "small" predator to only targeting prey that are small enough for it to eat.

When we discuss reproduction later, we will see that the offspring can be of a different stage to the parent, hence newborns may be created with different behaviour to their parents. This mechanism was found to also represent production of pellets very adequately, and so while strictly speaking we wouldn't call a pellet a "stage" of a copepod, treating it as such is convenient, since the same procedure for creating them is required.

The same concept of stages has since been used in conceptual epidemiology studies, where stages were added for diseased, infectious, or immune. So in general, while stages were designed with stages of growth in mind, the same mechanism has offered a variety of different possibilities, at the cost of slightly inaccurate terminology from time to time.

8.1 The Stage Interface

1. Select the functional group category, and a functional group in the items menu.
2. The "edit stages" button becomes available. Click it to show the stage selection window.



	NewBorn	Juvenile	Adult	Senile	Dead	Pellet	Imnr
Satiation	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Weighting Function	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Sinking	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Turbulence	<input checked="" type="checkbox"/>						
Day-time Migration	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Night-time Migration	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Reproduction	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Default Pools	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Calculate Ingested Nitrogen	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Newborn mortality	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Pellet Remineralisation	<input type="checkbox"/>	<input checked="" type="checkbox"/>					
Become Adult	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Increase Age	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

On the left side of the table are all the functions in your model, and along the top are all the stages of the functional group you selected – copepods in this example. The scrollbars will move the table contents around without altering the stage and function headings. The tick boxes show the current setting, in which certain functions are switched on or off in certain stages.

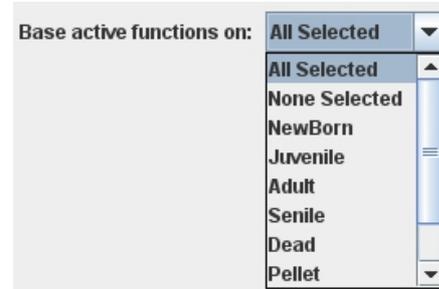
8.2 Switching functions on and off for each stage

1. To switch a function on in a given stage, tick the box for that function and stage!
2. To switch a function off in a given stage, untick the box.

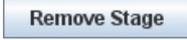
	NewBorn	Juvenile
Initiation	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Transition	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Linking	<input type="checkbox"/>	<input type="checkbox"/>
Presence	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Migration	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Migration	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

8.3 Adding a new stage

1. To add a stage, firstly click  .
2. A dialog appears, where firstly you type the name of the new stage in the text box. VEW will in due course ensure this is a unique name.
3. You can specify that you want the tick-boxes for that stage to start either all selected, none selected, or to copy the configuration of an existing box. Select this from the drop-down box.
4. Finally, click  to add the new stage.



8.4 Removing a stage

1. To remove a stage, click  .
2. A dialog appears, where you must select the stage to remove from the drop down list.
3. Note that the stage cannot appear in any reproduction or state-change rules in your model; VEW will refuse to delete the stage until it is not in use anywhere.
4. Click on  to finish the process.



8.5 Renaming a stage

1. To remove a stage, click  .
2. A dialog appears, where you must select the stage to rename from the drop down list, and type the new name in the textbox.
3. Click on  to finish the process; any rules in the model that used this stage will automatically be corrected.



9. Writing Rules

This is the unique core of the Virtual Ecology Workbench: writing rules for the behaviour of plankton. Most of the time spent creating a new model will be here. The language we will use is known as *Planktonica*. It consists mostly of straightforward mathematical assignments, but where special behaviour only applicable to plankton ecosystems is required, a special function is provided. There are 8 such functions, described throughout this chapter. See Appendix 1 for information on how to build all of these examples into test models – it requires knowledge of the later chapters of the handbook to complete the models.

9.1 Three Golden Rules

A clear understanding of the following two points may save a lot of time debugging later! They are necessary for the time-based Lagrangian-Ensemble simulation method we are using.

9.1.1 Rules are timestep-Based

Every rule that you write will be executed once per plankton agent, per timestep. VEW will not do anything subtle to your equation like multiplying it by the timestep size without you knowing; a design decision has been made to let the user do this, without anything implicit happening. A variable called "TimeStep" is available to tell you the size of the timestep in hours.

For example, if you want to write an rule that contains a rate in per-second units, then you should convert that rate to per-timestep by multiplying it by 3600 (seconds per hour), and by Timestep (hours per timestep), to reach units of per-timestep.

9.1.2 Some variables are timestep-based

We will discuss the different variable types Planktonica offers shortly. One of the types is a biological property, (known in some fields as a state variable) – this is an instantaneous property of an individual at a moment in time, such as its depth, its size or its satiation. Typically, the purpose of a rule is to define the *new* value of such a variable, as a function of its *previous* value, and the values of other variables, current environmental conditions etc.

The important point to note is that the changes to these variables happen on the timestep boundary. Consider the following two rules:-

$$\begin{aligned} X &= P + 2 \\ Y &= X + 5 \end{aligned}$$

Intuitively, you may think that after these rules, Y is equivalent to P+7, but this is not the case, because the value of X is not updated until the end of the timestep. What we are really saying is this:-

$$\begin{aligned} X_{t+1} &= P + 2 \\ Y &= X_t + 5 \end{aligned}$$

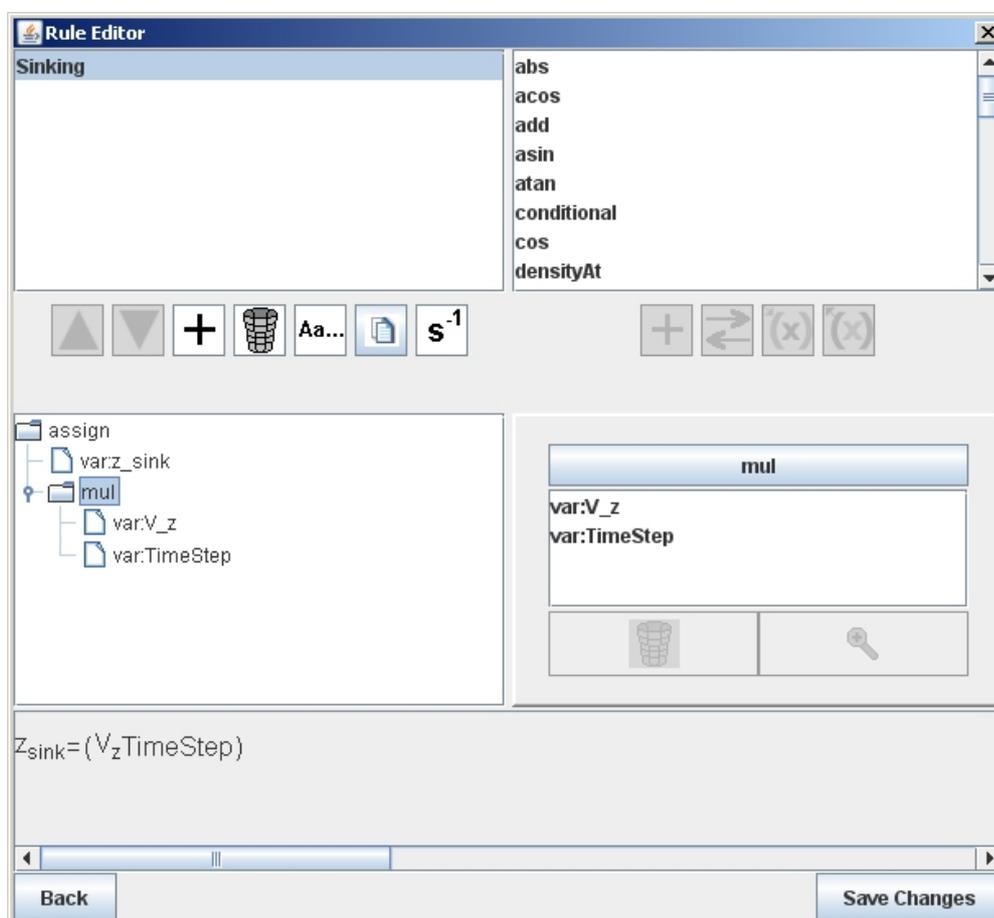
This shows more clearly that the X in the second equation is not the same as the X in the first equation; these types of variables are called buffered – their latest value is not 'published' until the next timestep. The left hand side of such equations refers to the new value for the next timestep, whereas the right hand side refers to variables in the current timestep.

9.1.3 Rules are individual-based

When you write a rule, you are defining the behaviour of an individual plankter. Internally, the system treats this as an agent representing a number of individuals, but when you are writing rules, this does not concern you. Therefore, the rules you write should only consider what an individual plankter “experiences”. A rule should not, for example, require knowledge about the total amount of a chemical in the mesocosm – a plankter would not do that, and indeed, the VEW will not permit you to write such a rule.

There are occasional exceptions to this: the depth of the turbocline is made available for rules, as without that it is difficult to model turbulent motion. Why not automatically force all plankton to be moved by turbulence? Forcing this would prevent you from writing equations that involve buoyancy, so we decided to allow use of the turbocline in rules, as a useful exception.

9.2 Introduction to the Rule Editor



The rule editor is shown above; in the top left is the list of rules that are part of the function currently being edited. The highlighted rule is previewed at the bottom, and it is shown in a tree format on the left-middle part of the screen. The currently highlighted node of the tree is shown in the detail viewer on the middle-right of the screen, and depending on the type of node highlighted, different options may be available. Generally, you can click on anything in those two middle sections, to navigate around the rule you are editing.

The menu at the top-right of the screen is a context-sensitive menu that changes depending on what is selected in the panel just below it; if you click on the “mul” (multiply), the list will update to show all the possible mathematical operations that you could replace the “mul” with, or that you could surround it with. Selecting something in the context-sensitive menu activates the buttons below it, which can add, replace, or surround an item selected in the middle-right panel.

The rule editor becomes simple to use with practise, so the following sections will show how to build a range of example rules, covering all the functionality that the rule editor has.

9.3 Example 1: A simple motion rule.

In this example, we will write a rule that updates the depth of a plankter. If it is above the turbocline, it will take a random depth between the surface and the turbocline, whereas if it is below the turbocline, it will sink at a constant rate. We will create a parameter to represent the sinking rate. Assuming that you have added a function as described in 7.1, and begin to edit it as in 7.6. It is a good idea to write the rule roughly on paper before trying to create it in the rule editor. What we are trying to make is this:-

$$z = \begin{matrix} \text{if } z \leq \text{turbocline,} & \text{rnd}(\text{turbocline}) \\ \text{otherwise} & z + \text{sink_rate} \end{matrix}$$

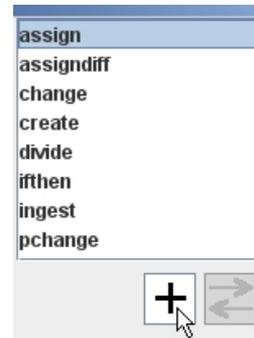
1. Click on the plus sign towards the top-left of the screen to add a new rule.



2. You will be asked to give a name for the rule.



3. After adding the rule, the menu in the top right of the screen gives the possible top-level types of rule you can create.



4. Click on the “assign” from the list, which is for simple X=Y kinds of rule, and then click on the plus symbol, to use add the assign to the currently empty rule.

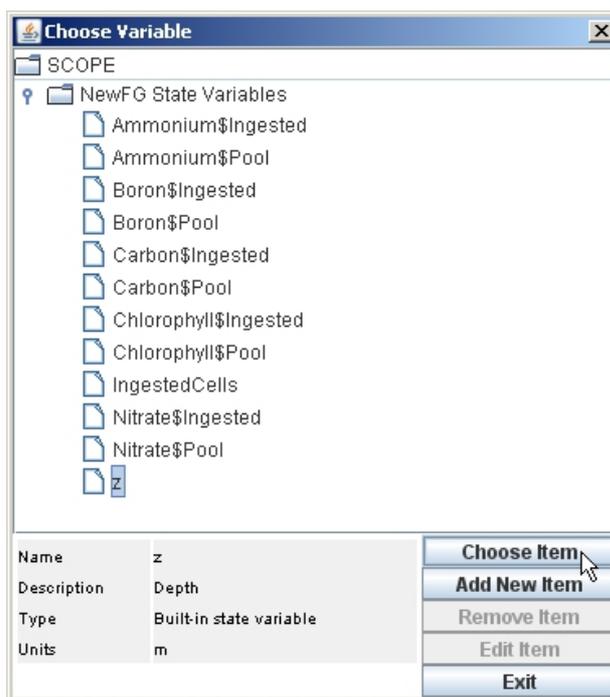
5. Notice the tree now shows three nodes, the parent is the “assign”, and below that the left and right hand side of the equation appear. “? Var” means that a thus-far unspecified variable is represented, whereas “? Num” implies a numerical term that is so far undefined.



6. This is mirrored in the detail viewer, which shows the “assign” title, and the “? Var” and “? Num” for the left and right hand sides. To set the left hand side, click on . Then click on .

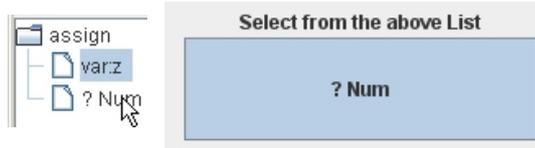


7. A menu of the available variables appears. As we are setting the variable for an assignment, this menu shows just the variables we can assign values to.



8. Depth, "z", is a built-in state variable for all functional groups. If necessary, click on the folder next to the State Variables menu for the functional group you have made. Then click on "z". To select the variable, click on **Choose Item**.

9. Notice that now the tree on the left has updated, replacing "? Var" with "var:z" – the variable you chose. Now let's define the right hand side. To navigate to the right-hand side, click on "? Num", in the tree – the second member of the assign statement.

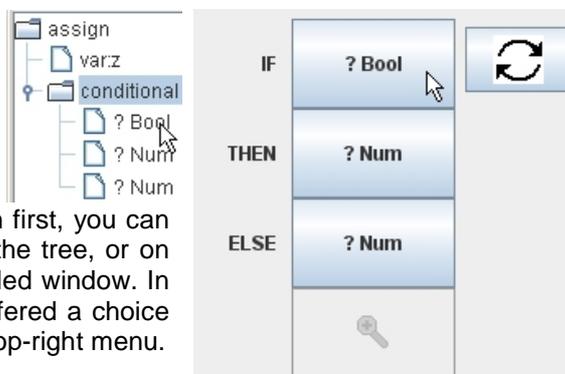


10. The detail viewer on the middle right of the screen now displays simply "? Num" - a numerical expression not yet defined.

11. To represent "if $z \leq \text{turbocline}$ ", Choose "conditional" from the menu in the top right, and then click on the replace icon. The tree, and the detail window will update to show the conditional: this is function takes a Boolean expression, and depending on whether it is true or false, it returns one of two possible answers.

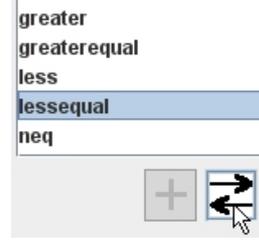


12. For our rule, the Boolean expression is $z \leq \text{Turbocline}$, where if true, the answer is $\text{rnd}(\text{Turbocline})$, but if not, the answer given is $z + \text{sink_rate}$.

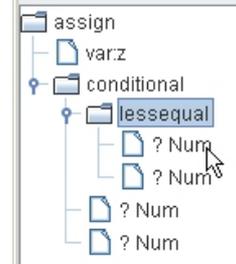
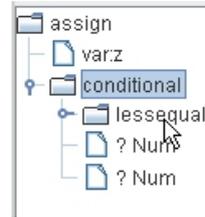


To edit the boolean expression first, you can either click on the "? Bool" in the tree, or on the "? Bool" button in the detailed window. In either case, we will then be offered a choice of boolean expressions in the top-right menu.

- The Boolean expression we want is "lessequal". Select this in the list, and click on the replace button.



- You will see that the "lessequal" has been added on the detail window, and on the tree. If you click on the newly created "lessequal" node on the tree, it will expand, so you can now see 4 undefined numerical terms (? Num) in the tree :- two are for the "lessequal" part, and the other two are the true/false values for the conditional.



- We want to construct "z <= Turbocline". Click on the topmost of the "? Num" items, which is the left-hand term of the Boolean. This time, the numerical expression we want is a variable, so select "var" from the list in the top right corner, and click the replace button.

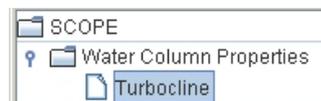


- The detail window now shows two buttons the left one, "var" is the type – you could click on this, and replace it with some other numerical function from the top-right list. The second button is the specific variable selected and as we have not selected one yet, it reads "? Var". Click on this, and select "z" from the variable chooser, under "state variables" - just as in step 8.



with some other

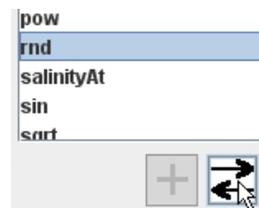
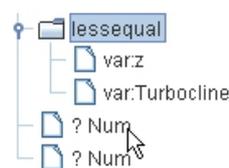
- Notice that the top "? Num" has now become "z" on the tree. Click on the next "? Num", and repeat the previous couple of steps, choosing "var" from the top right menu, clicking the replace icon. The "turbocline" variable we want is found under "Water Column Properties".



- The Boolean part is now finished. You'll notice as we build the rule that the equation panel in the bottom shows the preview of what we've done so far. There are gaps, as we haven't finished the conditional yet.

```
z=(if (z<=Turbocline)then else )
```

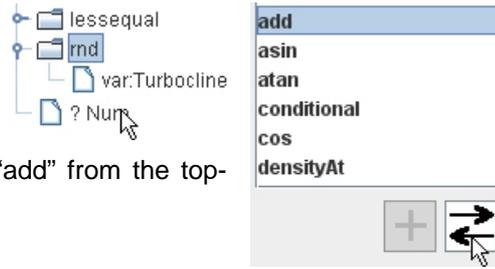
- Click on the next "? Num" in the tree window. This is the "then" part of the conditional, which we want to replace with a random depth between 0, and the turbocline depth. Select "rnd" from the list in the top right of the screen, and choose replace.



- The detail window shows a "rnd" button, and a "? Num" button, along with the expand button. Click on the "? Num", replace it with a "var", and choose the turbocline as the var, as in step 17.



21. Now click on the final “? Num” in the tree. This is the “else” part of the conditional, which we want to replace with “z+sink_rate”. This expression is an addition of two variables. Replace the “? Num” with “add” from the top-right menu.



22. The add function (and also multiply, and the Boolean functions *or* and *and*) take as many arguments as you like, since the order of them doesn't matter. If we wanted to change from “add” to something else, you could click on the title bar, and the top-right menu would give you options to with which to replace the “add”.



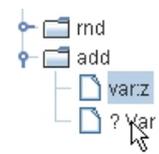
23. We need to add two variables. Click on “var” in the top-right table, and click the add button twice.



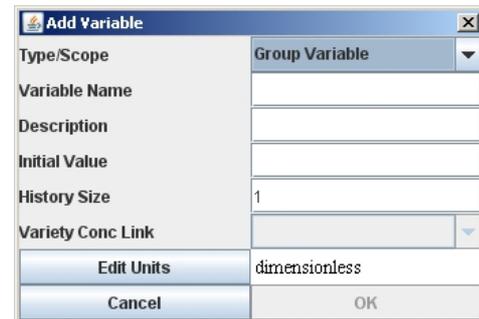
24. Then click on the first “? Var” – notice that the two icons become active. There is a delete icon, that allows you to remove items from this addition, and the expand button allows further editing. Click on the expand button to edit the first “? Var”, and as described in step 8, choose “z” as the first variable.



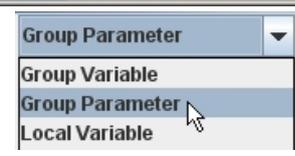
25. The second variable in the addition is the *sink_rate*. We will use this as an example of how to add a new variable – in this case, a parameter. Click on the remaining “? Var” in the tree, and then on the subsequent “? Var” that appears.



26. This time, choose **Add New Item**. A screen appears asking you for details for the new item.

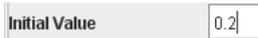


27. The Type/Scope list gives the available variable types. We want to create a Group Parameter – a constant sink rate for the functional group.



28. Type a name and description for the new parameter.

Variable Name	sink_rate
Description	Sinking Rate

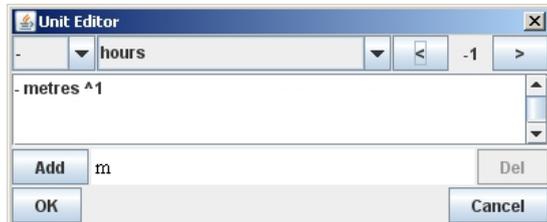
29. Next type a value for the parameter. Let's use 0.2 metres per hour as a sinking rate. 

30. Next, we'll set the units for the parameter. Break the units into their separate atoms – in this case, there are two components, metres, and per-hour. Click on  to show the unit editor.

31. The unit editor has three menus at the top, which allow you to choose the prefix (nano, micro, etc), the unit, and the exponent of the unit. Having chosen one of these, you can add and remove atoms. To select our first atom, "metres", choose "-" from the prefix list, "metres" from the middle list, and leave the exponent as "1". Then click .

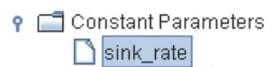


32. The atom is added to the list, and the preview of the units is shown at the bottom. Now, to add per hour, ensure the prefix list shows "-", select "hours" from the middle list, and then push the left arrow, which decreases the exponent from 1, to -1. Then click .



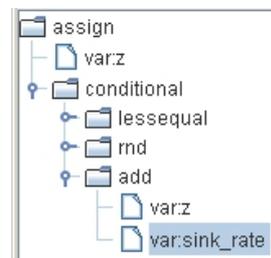
33. The units are now done, so press . The window now shows the units you added. 

34. Click on  to finish adding the parameter. You must now choose the parameter from the list, under "Constant Parameters". Then click on .



35. You have now finished this example; the complete equation is shown in the preview window at the bottom, and in the tree on the left. Click on .

```
z=(if (z ≤ Turbocline) then rnd(Turbocline) else (z+sinkrate))
```



36. However, examining the rule closely, we now observe with shock and horror that we have broken the golden rule described in 9.1.1. Rules must be timestep based; *rnd(turbocline)* is fine, however, if the agent is below the turbocline, the difference will be *sink_{rate}*, which is in metres per hour, not metres per timestep. In the next example, we will fix this with some simple editing.

37. Don't forget, when a rule is done to make sure the right stages of your functional group will use that rule – return to chapter 8 for instructions on stages.

9.4 Example 2: Fixing the error in Example 1.

We want to replace this:-

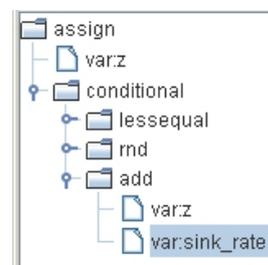
$$z = \begin{cases} \text{if } z \leq \text{turbocline,} & \text{rnd (turbocline)} \\ \text{otherwise} & z + \text{sink_rate} \end{cases}$$

with this:-

$$z = \begin{cases} \text{if } z \leq \text{turbocline,} & \text{rnd (turbocline)} \\ \text{otherwise} & z + (\text{sink_rate} * \text{TimeStep}) \end{cases}$$

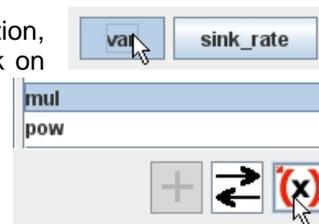
Then the rule becomes timestep-based. Consider, for example, if TimeStep is 0.5 hours, as is commonly the case with these simulations (due to assumptions of the physics code, regarding turbulence). As *sink_rate* is 0.2 metres per hour, multiplying by 0.5 hours per timestep, gives 0.1 metre per timestep, as the amount that z should change by, in a timestep. To fix the error, we carry out the following steps.

1. Open the equation as before. The part we want to edit is the “else” condition, which is currently an addition of two variables. Expand the tree by clicking on “conditional”, and then on “add”. Then click on the “sink_rate”.



2. We will use the Rule Editor’s “surround” button, to surround the variable “sink_rate”, with a multiply function, and then we will include the Timestep variable in the multiplication. Having clicked on “sink_rate”, you should see the variable in the detail window.

3. Click on “var”, and then choose the multiply function, “mul”, from the list in the top right corner. Now click on the “surround item” button, just next to the replace button. This surrounds the item selected in the detail window, with the function chosen in the top-right hand menu.



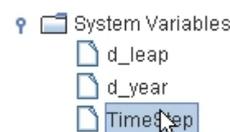
4. Notice how the tree has now been edited, and the “mul” is the child of the “add”, but the parent of the “sink_rate” variable. Also the detail window, which used to be a “var”, has now been replaced with a “mul” window, and the sink_rate is one of the items being multiplied together.



5. The next task is to add the timestep to the multiplication. Click on “var” in the top right menu, and click on the plus button, to add a var to the list of items to be multiplied. Then click on “? Var”, and the expand button. Click “? Var” again on the next screen:-

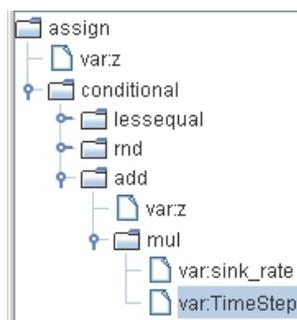


- In the Variable Chooser window, click on the “System Variables” heading, and then click on the “TimeStep” variable – this is the size of the timestep in hours. Then click on .



- The edit is now complete, and it is now a correct time-step based rule.

```
z=(if (z ≤ Turbocline) then rnd(Turbocline) else (z + (sinkrate TimeStep)))
```



- So click on , and once again, ensure that this function is called in the right stage(s) – see chapter 8.

9.5 Example 3: Nutrient uptake

In this example, we will consider a plankter that is performing uptake of ammonium. Our rules look as follows:-

$$rate = \left(1 - \left(\frac{Ammonium_{pool}}{Ammonium_{max}} \right) \right) rate_{max}$$

$$uptake(rate * Timestep, Ammonium)$$

In the first rule, we define the rate, which is a maximum uptake rate ($rate_{max}$), scaled inversely by the “satiation” of the plankter, measured by the current content of ammonium ($Ammonium_{pool}$), divided by its capacity $Ammonium_{max}$. The second rule performs the uptake, using a special function. Two important points to note here:-

9.5.1 Local Variables and Ordering

We want the result of the *rate* variable immediately for use in the uptake equation. This is different to the previous example, in which we used the variable *z*, which is a state variable; the new value of *z* does not become available until the following timestep, as described in 9.1.2. Here, we want the latest value of *rate*, without waiting a timestep.

Of course, we could instead write the equation for *rate* directly into the *uptake* rule, but this would lead to lengthy and confusing rules. Additionally, if we were to re-use the value of *rate* in a number of places, it would be less efficient to repeat the calculation many times. The modelling language therefore provides a local variable type, which can be used to calculate a value and have it ready for use immediately.

The important point to note is that the use of local variables is one example where **the order of functions is important**. A local variable is visible throughout all of your functions, and you must ensure that you only read that variable after it has been set. In this example, if you take all your functions from top to bottom, and for each one take all of its rules from top to bottom, the definition of *rate* must occur before (that is, higher in the list), then the use of *rate*, in the uptake rule.

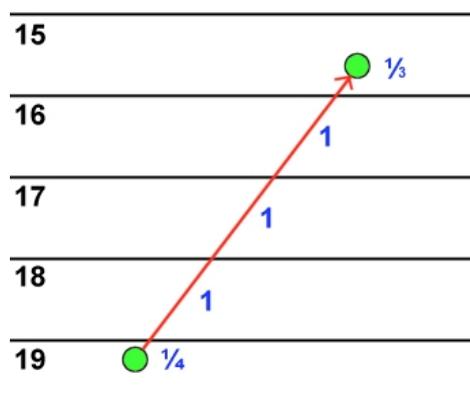
9.5.2 The Uptake Statement

The syntax of the uptake statement is:-

Uptake(amount to uptake per timestep, Chemical).

If our rate is in units per hour, as assumed in the rules above, we will have to multiply the final rate by the timestep. This statement issues a request for the given chemical, because there may be many other plankters issuing similar requests, and there may not be enough nutrient to satisfy all requests.

At the end of the timestep, by which time all the requests will have been made, VEW takes each plankter that made a request, and considers the plankter's trajectory. It may have visited several layers, in which case a number of requests in separate layers will be made, each one scaled by the length of time the plankter spent in that layer, as a fraction of its complete journey in that timestep.



For each layer, for each chemical, the requests are summed. If supply cannot meet demand, then every request is scaled down proportionally, so that the total requests will equal the amount available. The actual amount of chemical a plankter acquired is then placed in a special variable for incoming chemical.

9.5.3 Building the Rules

The previous examples have gone into button-by-button detail and the examples from here on are entered in a very similar way. So from here on, we'll summarise how to create the rules, and only mention in detail where new interfaces are encountered.

$$rate = \left(1 - \left(\frac{Ammonium_{pool}}{Ammonium_{max}} \right) \right) rate_{max}$$

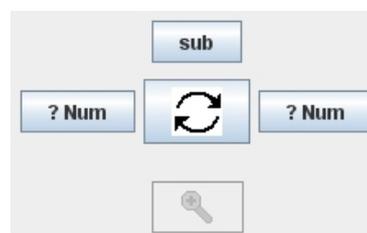
$$uptake(rate * Timestep, Ammonium)$$

Firstly, the example assumes that you have created a chemical called Ammonium – see chapter 6 for instructions. On doing so, the variable Ammonium_{pool} is automatically created in the state variables menu – the same place as “z” in the previous examples.

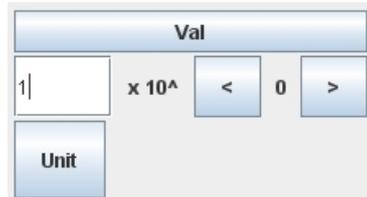
- For the first rule, we must create parameters $rate_{max}$ and $Ammonium_{max}$. This was described in 9.3, step 26. Values that would produce reasonably obvious results could be 0.05 mMol Ammonium per hour, and 1 mMol Ammonium, respectively. We also need to create the local variable $rate$, which is done similarly - just choose "Local Variable" from the top menu.

- Having created the variables, the first rule we are creating is an assignment, and the left hand side is the new "rate" local variable. The right hand side is a multiple of a subtraction, (sub, in the menu in the top right), and the parameter (var) $rate_{max}$.

- All subtractions have just two parts, to avoid any bracketing ambiguity. The subtract here takes an absolute value (val) and a division (also a binary function); the division is of the state variable $Ammonium_{pool}$, and our maximum capacity parameter, $Ammonium_{max}$.



- The absolute value interface is a little different. The top title, like all such titles, allows you to replace (or surround), the value with another function from the top-right menu. The next line allows you to type a value and to select an exponent of the value, using the increase/decrease arrows. The Unit button takes you to the unit editor as in 9.3, part 30, to identify the units and dimensions of the value you are giving.



- The second rule is not an assignment; instead, choose $uptake$, from the list in the top right. The uptake interface appears. Choose the chemical from the drop-down list, and then click on the "? Num" to set the amount – the size of the request to be made.



- The rest of the rule is simple; a multiple of two variables, the timestep, found under "System Variables", and the rate we created, found under "Local Variables".



9.5.4 A note about the units of chemicals.

The VEW does not force you to use a particular standard of units for chemicals. The Levitus data set provides chemical concentrations in units of μg for each chemical, and the VEW can import from that dataset to use as initial conditions. However, we will see later that you can also specify your own values for the initial concentration of chemicals, which could be in any unit you like. Therefore, in the unit editor, you can choose units of either micrograms, or millimols of chemicals.

9.5.5 Updating the Chemical Pool

Note that uptake does not automatically put the chemical obtained into the pool of the plankter that requested the chemical. The reason is that you may want to do something particular with the chemical, without having things automatically done to it. For example, some models may update carbon from solution, and then convert it into different types of carbon internally – carbon lipids and proteins for example.

To allow for this kind of flexibility, the responsibility of budgeting the nutrients on the site of the plankter, is the modeller's, not the systems. Hence, to balance the nutrients from this uptake rule, you need to add one further rule:-

$$Ammonium_{pool} = Ammonium_{pool} + Ammonium_{ingest}$$

The “incoming chemical” from uptake, (along with any chemicals gained by ingesting other plankton that themselves contained chemical), is put in the state variable $Ammonium_{ingest}$. This is created automatically for every chemical you introduce.

If you do not include this rule, then the chemical budget will be broken; chemical will be removed from solution, but not added to the particulate chemical.

9.5.6 Chemical concentration in solution

The simple rule described in this section did not need to know the ambient concentration of chemical; it requested chemical at a rate independent to the available concentration. Other rules for nutrient uptake may not be so simple and may require the ambient (local) concentration of a chemical in order to calculate the requested uptake rate. A variable is therefore provided under “Chemical Properties” for each chemical in your model.



This variable is read only – because of the risk of full depletion of chemicals, the only way you can change the concentration in solution is by asking the VEW to do it for you, in which case it makes the changes ensuring that depletion errors do not occur.

9.6 Example 4: Releasing chemicals to solution.

Remineralisation is treated very similarly to uptake. Since it adds chemical to the water rather than attempts to remove it, releasing chemicals does not require the request/repair mechanism that uptake requires. However, because for the purposes of uptake it is forbidden to write to the chemical concentration directly, the VEW kernel provides a *release* statement to make this change to the ambient environment.

We still need to remember that the amount released to solution must be subtracted from the chemical pool from which you release it. Our rules could therefore look like this.

$$Ammonium_{release} = Ammonium_{pool} * release_{rate}$$

$$release(Ammonium_{release}, Ammonium)$$

$$Ammonium_{pool} = (Ammonium_{pool} + Ammonium_{ingest}) - Ammonium_{release}$$

9.6.1 Using one rule to set a state variable

Remember from section 9.1.2 that some variables are buffered, or timestep-based. Chemical pools are an example of such a variable-type. Therefore, each plankter should only execute a rule that beings “ $Ammonium_{pool} =$ ” once per timestep. This means that the update for the Ammonium Pool must include the changes from both uptake (and ingestion), and any release of chemicals. We will see in 9.7 that growth by division is also relevant to this.

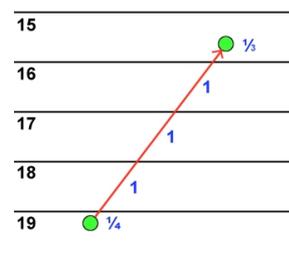
If we were to write two rules that defined the new value for the Ammonium Pool, then the first would be ignored, because both of them would be defining the value of the pool in the *next* timestep, as a function of the pool, and other variables from the *current* timestep.

9.6.2 The release statement.

The syntax of the release statement is:-

release(amount to release per timestep, Chemical).

Like uptake, the amount of chemical to be released is uniformly distributed across the layers the plankter visited while travelling from one timestep to the next, depending on what fraction of the timestep it spent in each layer.



9.6.3 Building the rules

$$Ammonium_{release} = Ammonium_{pool} * release_{rate} * Timestep$$

release(Ammonium_{release}, Ammonium)

$$Ammonium_{pool} = (Ammonium_{pool} + Ammonium_{ingest}) - Ammonium_{release}$$

The first rule is a simple assignment, where $Ammonium_{release}$ is a local variable, since we want to use it immediately after assigning it. $Ammonium_{pool}$ is found under state variables, and the $release_{rate}$ can be a parameter – for example 0.05 mMol of ammonium per hour. To convert from per-hour into per-timestep, we multiply by the *Timestep* size in hours, found under System Variables.

The second rule is a “*release*” rule. It is very similar to the uptake interface, so you must choose the chemical to release from the drop-down list, and then replace the “? Num” with the amount to release – in this case, the local variable $Ammonium_{release}$.



The third rule is an assignment. On the left hand side is the state variable $Ammonium_{pool}$, and on the right hand side is a subtraction of two parts; the leftmost part is the addition of $Ammonium_{pool}$ and $Amomnium_{ingest}$ – both state variables, and the right hand side of the subtract is the local variable $Ammonium_{release}$. This rule combines the changes for both the chemical the plankter gained in the previous timestep due to update or ingestion, and the chemical lost through this release rule, which will be added to the concentration of the layers the plankter swum through for the beginning of the next timestep.



9.7 Example 5: Growth by Division

Two methods of reproduction are available in the VEW: division and reproduction. The main difference is that with division, all the plankters after the division are identical, whereas for reproduction, the offspring can be different from the parent. The latter option is more computationally expensive, since it requires creating a new plankton agent. As division can happen very frequently (for example in a diatom bloom), for the sake of performance it is worth modelling division separately and more efficiently. In this example, the individual divides once it reached a target Ammonium content. The rules are as follows:-

$$C_{div} = \text{if} (Ammonium_{pool} > Ammonium_{cdiv}) : 2, \text{else} : 1$$

$$\text{if} (C_{div} = 2), \text{divide}(2)$$

Additionally, we will have to adjust the value of the $Ammonium_{pool}$, because after division, the available Ammonium should be spread between the two individuals. Therefore, we also write:-

$$Ammonium_{pool} = \frac{(Ammonium_{pool} + Ammonium_{ingest})}{C_{div}}$$

9.7.1 The divide statement

The second rule above introduces the divide statement. When you write this rule, considering an individual, you are stating that an individual should



divide into 'n' identical individuals. The system interprets this efficiently by multiplying the sub-population size for that agent by 'n', so division does not cause any new agents to be created. The syntax of the divide statement is simply:-

divide (number to divide into)

Although division is commonly splitting into two, the divide rule generically allows one individual to divide into any number. Curiously perhaps, because of the nature of the Lagrangian Ensemble model, this number can even be fractional, although we have had no cause to use this flexibility as yet.

9.7.2 The ifthen statement

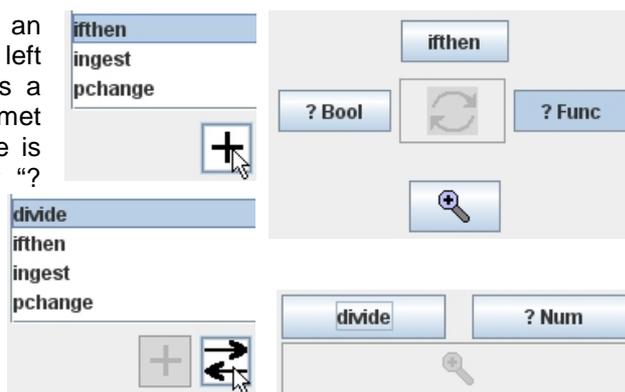
Notice also that in the second rule, we only divide if a certain condition is met. VEW provides an *ifthen* statement for this purpose. It differs from the conditional described in 9.3.11 (and used in the previous rule), because *ifthen* decides whether to *do* something, rather than *which value to return*. It can be used to decide whether to execute any of the statements (assign, uptake, release, divide, and those described in the next few sections), and they can even be nested (if a>b then, if c>d, then ...).

9.7.3 Building the rules

The first rule is a conditional similar to that in 9.3, part 11. We must create the C_{div} variable, which should be a local variable, since we require its value immediately. As your model may use this fact (ie, whether the individual has divided this timestep), make sure that your division function occurs higher in the list of functions than any other functions that use it – see section 7.4.

Additionally, we must create the parameter $Ammonium_{cdiv}$, the threshold at which division occurs. An interesting value here could be a little less than the maximum Ammonium content, so that division happens eventually but not immediately; try 0.95 mMol of Ammonium.

For the second rule, we create an *ifthen* statement. This has a left and right hand side; the left is a Boolean, which we have met before, and the right hand side is another function, indicated by “? Func”. Clicking on this lets you replace “? Func” with one of the statements, which in this case is *divide*.



The *divide* detail screen contains just the *divide* button, to let you replace it with something else if you choose, and the “? Num”, which will be replaced with the value 2, as described in 9.5.3, part 4.

The Boolean part of the *ifthen* statement, is a comparison between the local variable C_{div} , and the value 2, which you create by choosing “equal” from the list of Boolean functions.

9.7.4 Choice of functions

The third rule should be a fairly simple assignment now. However, we have mentioned rules for the $Ammonium_{pool}$ in uptake, release and cell division. As there must only be one statement that gives a value to $Ammonium_{pool}$, where should it be?

Perhaps the best answer is to create a new function called “Set Ammonium Pool”, arranging it later in the list of functions than your uptake, release and cell division functions, thus ensuring that all the local variables are assigned.

The rule is then an assignment, with the state variable $Ammonium_{pool}$ on the left. The right hand side is a divide with the local variable C_{div} on the bottom, and on the top, it is a subtraction. The left side of the subtraction is the addition of the two state variable, $Ammonium_{pool}$ (from the previous timestep), and $Ammonium_{ingest}$, and on the right side of the subtraction is the local variable $Ammonium_{release}$.

9.7.5 Important note for releasing chemicals

If you have a functional group that does both cell division, and releases some chemical, then you must divide the amount of chemical you release by the cell division variable (ie, if you have divided into 2, you should divide the amount of chemical remineralised by two. Or more generally in the above example, divide by C_{div} .)

9.8 Example 6: Creating New Individuals

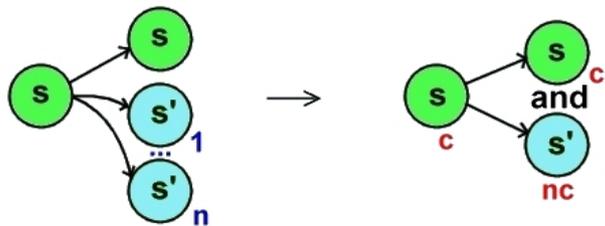
We will now consider the production of new individuals. In this example, we will replace remineralisation with the production of a pellet containing the Ammonium that would have been remineralised. The method is identical to how you would define the production of offspring. We'll use a combination of the previous parts: firstly, we'll create a pellet stage, and then use the chemical release example, in that stage only. We'll take a fraction of the ammonium gained by uptake in a timestep, subtract that fraction from the new value of the pool, and put that amount into a new pellet, with the following rules:-

$$\text{Ammonium}_{\text{pellet}} = \text{Ammonium}_{\text{ingest}} * \text{Ammonium}_{\text{excretionfraction}}$$

If ($\text{Ammonium}_{\text{pellet}} > 0$)
 $\text{create}(\text{Pellet}, 1$ where
 $\text{Ammonium}_{\text{pool}} = \text{Ammonium}_{\text{pellet}})$

9.8.1 The Create Statement

When writing rules, we consider individuals. The create rule therefore states that an individual should produce 'n' new individuals of the same functional group, but perhaps a different stage. These newly created individuals are identical to each other both in stage, and in the values for their state variables, which may also be different from their parent. Since all the newly created agents are the same, VEW represents this by creating one new agent for all of the "offspring", multiplying by the sub-population size of the parent agent. The syntax of the create rule to be written is:-

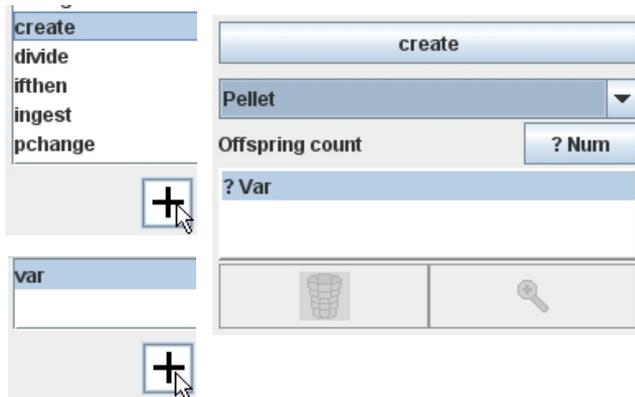


Create(stage, number of offspring, list of assignments)

A number of individuals are created by the parent, in a given stage, and any number of assignments may be added: these assignments define properties of the "offspring", as a function of the ambient environment, and the properties of the parent.

9.8.2 Building the Rules

In preparation for building this kind of rule, make sure that you have made the stage of the individuals that the rule will create. In this example, a stage called "Pellet" must exist: see section 8.3. Create a new rule, and choose "create" from the menu.



Firstly, select the stage "Pellet" from the drop down list. Notice also that in the menu in the top left part of the screen, a single entry "var" is shown. Clicking the "Add" button adds a "? Var" to the list on the create screen, as shown above. Each of these represents a variable of the new individual that you would like to particularly set. All other variables of the new individual will assume equal values to those of the parent.

We want to set $Ammonium_{pool}$ to a different value. Click on the “? Var”, and expand it with the magnifying glass icon. The detail window now shows a “Set” dialog, which is similar to an assign. The “? Var” button allows us to choose $Ammonium_{pool}$ under “State Variables”, and the “? Num” allows us to choose a Var, and then select $Ammonium_{release}$ under local variables.



Finally, back on the create page, the “? Num” sets the number of individuals to create, which should be set to 1, by choosing a “Val” from the menu, as in 9.5.3, part 4.

Also remember to update the function that set the pools – instead of adding $Ammonium_{ingest}$, you should add $(Ammonium_{ingest} - Ammonium_{pellet})$ – see 9.7.

9.8.3 More about the Set statement.

When using the set statement with create, the left hand side refers to a variable of the individual being created, and the right hand side refers to the scope of the parent. Hence, you can write a set rule that says:-

$$Ammonium_{pool} = \frac{Ammonium_{pool}}{2}$$

and in this context, it means that the pool of the new individual will be set to half the value of its parent’s pool.

9.8.4 Chemical Budget reminder

Remember when setting chemical pools for new individuals that the chemical must have come from somewhere – specifically, the parent’s pool. You must ensure that if you create individuals that have chemicals in them, you subtract the amount per individual, multiplied by the number of “offspring” from the pool of the parent, in order for the chemical budget to be maintained.

9.9 Example 7: Changes of Stage

This example shows how an individual can move from one stage to another. We will use two stages, one for “day-time mode”, and another for “night-time mode”. We will use visible irradiance with a threshold value to determine when a stage change should happen. Our rules are as follows:-

NightTime :

if (VisibleIrradiance > I_{ref}) change(DayTime)

DayTime :

if (VisibleIrradiance < I_{ref}) change(NightTime)

The two functions will be called in different stages: it will only be possibly to change to daytime mode if the plankter is in night-time mode, and vice versa.

9.9.1 The Change Statement

The change statement is very simple:-

change (Target Stage)

The change in stage happens at the end of the timestep, so the plankter will finish the set of rules it was performing according to its initial stage in that timestep. It will effectively change state immediately before the following timestep begins.

9.9.2 Ordering of Rules with Stage Changes

Note however that the change statement is another example that makes the order of your rules and functions important. If you write more than one *change* statement, and both get executed for one plankter, then the earlier change will be ignored, and the latter change will be taken as the stage for the following timestep.

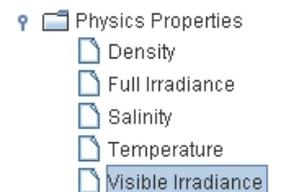
Ideally you should use a single state change command, which takes into account all the potential contributors and chooses the stage the plankter should assume, if a change is necessary. However, this can make building complex models even more complex, so VEW does not force you to use this suggestion.

9.9.3 Building the Rules

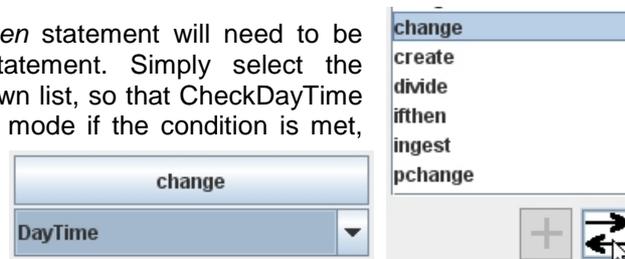
Firstly, create the two stages, DayTime and NightTime, as described in section 8.3. Next, you need to create two functions, CheckDayTime, and CheckNightTime. You need two separate functions so that you can have the rules executed in different stages. Use the stage editor to set that the CheckDayTime function is only called when the functional group is in the NightTime stage, and vice versa.

Stage Editor		
	DayTime	NightTime
CheckNightTime	<input checked="" type="checkbox"/>	<input type="checkbox"/>
CheckDayTime	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Now edit the functions one by one. Each function contains one rule, which is an *ifthen* type of rule. The condition is a greater than, or a less than, for CheckDayTime and CheckNightTime respectively. The variable “visible irradiance” in Wm^{-2} is found under Physical Properties. The variable I_{ref} we’ll create as a parameter, set at $4 Wm^{-2}$; this will be the threshold above which the plankter will assume day-time operation, and below which it will assume night-time operation.



The “? Func” part of the *ifthen* statement will need to be replaced by the *change* statement. Simply select the required stage on the drop-down list, so that CheckDayTime will change into the DayTime mode if the condition is met, and CheckNightTime will change into the NightTime mode.



9.10 Example 8: Probabilistic Changes of Stage

The previous example allowed plankters to oscillate between to stages. You may also want to have a probalistic stage change, with which you can say there is a probability of changing state. This has been used to model gradual senility, where the probability of dying by senility increases over time. It has also been used to set a probability that a plankter may choose to go through an overwintering phase.

In this example, we will say there is a certain chance of dying through some disease, and this chance increases if the sunlight is greater. The rules we will write are as follows:-

$$p_{death} = \left(0.0005 * \frac{VisibleIrradiance}{T_{ref}} \right)$$

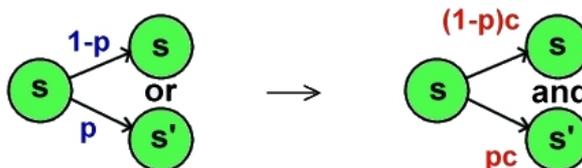
$$pchange(Dead, p_{death})$$

9.10.1 The PChange Statement

The probabilistic change statement has the syntax:-

$$pchange (Stage, probability)$$

When modelling an individual plankter, this rule is read as: there is a probability p , of the individual changing stage, whereas the remainder, (probability $1-p$) stay in the original stage. The VEW interprets this a little differently: it converts a *probability* for an individual into a *proportion* of an agent. It creates a new agent consisting of the proportion p of the original agent's sub-population, and reduces the size of the original agent's population to the proportion $1-p$.



9.10.2 Particle Management Issues

Note that the use of the pchange statement causes new agents to be generated, and you must control this addition to ensure that your simulation does not fill up with so many agents that it grinds to a halt.

Particle management rules are provided as we will see later. Also, when building your model, if a plankter has performed a pchange, consider whether the plankter should be forbidden from performing subsequent pchanges. If it should be forbidden, then you should arrange that having done the *pchange*, the plankter then changes stage to one that does not call the function that executes the *pchange*.

In general, the important point is to be aware that *pchange* can generate agents regularly, so you may need to address the point either with particle management rules, or to limit in some way how frequently it performs a *pchange*.

9.10.3 Building the Rules

The first rule is simple enough. The probability of death, p_{death} should be a local variable, since we require its result immediately. The rule is then a simple assignment with p_{death} on the left, and on the right the multiple, which is of a scaler, 0.0005, and a division of two vars. The *visible irradiance* variable is found under Physical Properties, as in 9.9.3, and I_{ref} is a parameter we must make to scale the temperature; a value of 4 Wm^{-2} may be a reasonable reference irradiance.

The second rule uses the pchange interface, which we have not yet seen. It has the usual title button, which you can use to replace the pchange with some other statement. It then has a numerical button, which when expanded lets you define the equation for the probability of changing state. For this rule, we define that as a "var", choosing the local variable p_{death} . Lastly it has a drop down menu to select the stage to change to, which in our case is *Dead*.



If these rules are combined with the motion rule in example 2, then the agent will produce dead agents when it is light, and the closer the agent is to the surface at any moment, the larger the dead agents it produces will be.

9.11 Example 9: Integrating over depth

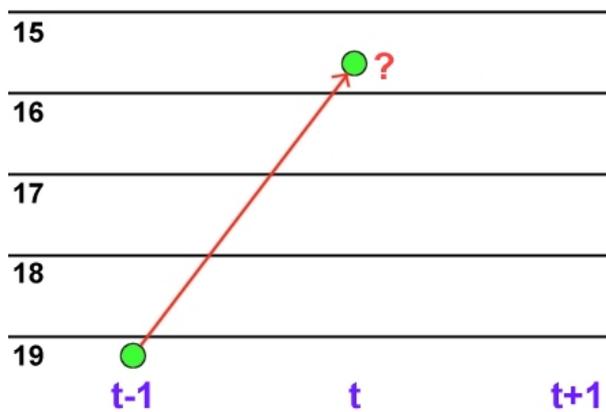
Sometimes in order to decide what behaviour to perform next, details about the conditions a plankter encountered across the distance it travelled in the last timestep may be required. For example, we may like to write rules that deal with the prey encountered throughout the distance the plankter has just swum, or perhaps the average temperature over a previous journey. The *integrate* function allows any property that varies over depth, to be integrated over the distance the plankter travelled in the previous timestep. In this chapter, we will consider a simple example, and then we will refer to integrate again when describing with ingestion. Consider the following rule:-

$$T_{avg} = \frac{\int Temperature}{|z - z[1]|}$$

The purpose of this rule is return the average temperature (a depth-dependent property), across the depth the plankter swam through in the previous timestep.

9.11.1 The integrate Function

The *integrate* function is used when a plankter must consider what to do in the next timestep, based on properties that varied over depth, as it carried out the changes specified in the previous timestep. It always integrates over depth between z , the depth at the beginning of the current timestep, and z from the previous timestep – referred to as $z[1]$.



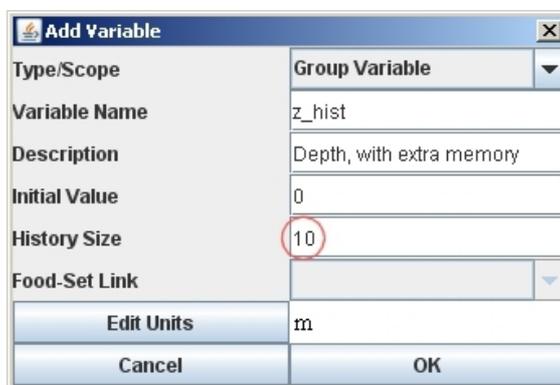
Any numerical function can be put inside an *integrate*. For example, if you calculate *integrate*(1), then you will calculate the integration of 1 between *z* and *z*[1], giving you, trivially, the distance travelled in the last timestep.

The VEW automatically detects which items inside the *integrate* are depth-dependent. Like the previous depth-dependent functions (uptake and release), the calculation is done layer-by-layer, proportionally including the fractions of a layer visited at the beginning and end of the journey.

9.11.2 Variables can have a history.

The next question is how to represent *z*[1]. VEW allows all state variables to have a history, which you can set when creating the variable. The depth, *z*, is a built-in variable, and this has been given a history size of 2, meaning you can see not only *z* from the beginning of the current timestep, but also *z* from the beginning of the previous timestep. This is likely to be all that is required for *z*.

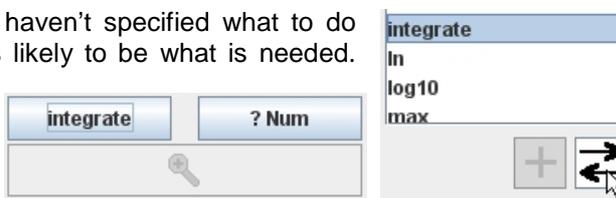
However, you can create state variables with a memory as large as you like, allowing you to look back any number of timesteps to see what the value of that variable was. This has been useful in the past for modelling gut passage, where a plankter's excretion was based on what was eaten a number of hours ago. Incidentally, if you did want to keep track of the depth for more than a timestep ago, you could simply create a variable with a larger history, and assign it the value of *z*.



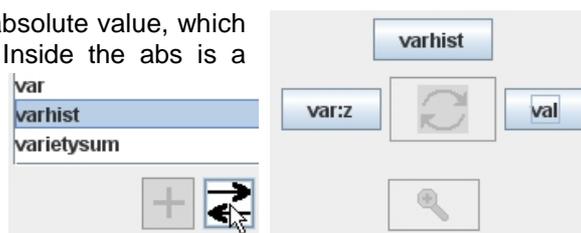
Having set the history size, you need to know about the *varhist* numerical function, which takes a variable, and a numerical value, which is the index in the memory (ie, the number of timesteps to look back) to recall. This numerical value can be calculated in any way – it will usually be an absolute value, but can be the result of some other calculation.

9.11.3 Building the rule

The rule is an assignment; we haven't specified what to do with T_{avg} , but a local variable is likely to be what is needed. The right hand side is a divide, the top part of which is an *integrate* numerical, which takes just a numerical expression. The expression it takes here is just a *var*, Temperature, under physical properties, which we have used before.



The bottom half of the divide is an absolute value, which we find as *abs* in the top menu. Inside the *abs* is a subtraction of the variable *z*, which we have used before, and then we need to choose the value of *z* at the beginning of the previous timestep. To do this, select *varhist* from the top-right list, and then select the variable *z* for the left hand side, and the value of 1, for the right hand side.



9.12 Example 10: Ingestion

The final, and most complex part of the VEW modelling language is ingestion. It involves choosing the target food to eat, ensuring that many individuals together do not attempt to ingest more than is available, then at some stage reducing the individuals that get ingested, and providing a mechanism by which the predator knows exactly what it managed to eat. It is further complicated by the fact that so far we have are modelling a functional group, but in the next part of the VEW, we will extend this to modelling species.

However, the solution to these challenges is elegant, and although we need to deal with some new variable types, the flexibility and simplicity of the final solution is appealing. Essentially, we specify that ingestion is going to happen, without at this stage specifying the prey that will be ingested. The aim of the next few steps is to calculate a rate of ingestion, taking into account how much food is present.

9.12.1 The Food-Set Variable Type – “The Diet”

We introduce here a new variable type, called a **food-set**. This represents a list of types of food, where each food is defined by its species and stage. You could think of it as a “diet” of acceptable foods. Clearly we haven’t defined the species yet, so we will be **declaring** that the food-set exists, but **not defining** the contents of the food-set until later on in the model building process; species will be described in chapter 11, and food-set relationships in chapter 12.

Type/Scope	Food Set/Concentration
Variable Name	Food
Description	Food-types I can eat
Initial Value	0
History Size	1
Food-Set Link	
Edit Units	dimensionless
Cancel	OK

You can add any number of food-sets; for example, a predator may have one diet in a certain stage of its life, and another when it has grown. Or, different food-sets might be used concurrently, if there are substantially different processes for ingesting different foods.

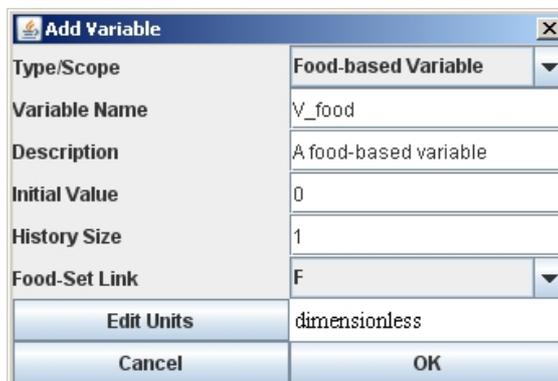
The food-set serves various purposes.

1. Firstly, it defines for a particular act of ingestion the diet for that ingestion. It is a required part of the ingestion statement we will look at shortly.
2. It is also used as an **index set**; other variable types later will essentially say, “for each member, m , of food-set F , I will have a parameter, P_m .” The linking between that type of parameter, and the food-set, must be specifically stated.
3. When used as a variable on its own, it gives the ambient concentration of food. However, the set represents a number of types of food. Hence, the rule will be executed a number of times, one for each food type in the diet. Therefore, we need ways of either dealing with a rule giving us multiple results, and ways of reducing the multiple results down to a scalar term.
4. When used as a concentration, it is a depth-dependent property and crucially, can be used in the *integrate* function, to integrate how many prey of each type were encountered in the journey during the previous timestep.

9.12.2 Food-Based Locals, State Variables and Parameters

So we now have a way of talking about the concentration of food-types that a predator may eat. However, as this food-set is a set, (i.e., an array, or vector) of values, we need other array-based variables types to use in calculations involving food-sets.

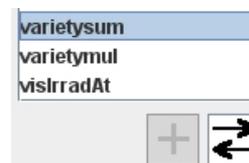
The VEW provides food-based versions of local variables, state variables and parameters – all of which we have met earlier in their scalar forms. Now we are creating, for example, a parameter for each member of a food-set – how ever many members there may be in due course. On the right, the creation of a food-based state variable is shown. Note that the Food-Set link identifies a food set F – that means for every food-type in the set F, there will be a value for V_{food} .



This means we can write rules like $V_{food} = F$, and both sides of the rule are equivalent in type – they are both arrays of equal length, and each index of V_{food} relates to the same index of F.

9.12.3 Reducing food-based variables down to scalars

For some rules, you will want to reduce food-based variables down to single values again. VEW provides two functions that enable you to do this, *varietysum* takes any of the food-based variable types, and sums all the elements. For example, *varietysum(F)* would add up the individuals of all acceptable food types and return a single value. *Varietymul* returns, the product of the elements of a food-based variable.



9.12.4 Food-based conditional statements

Additionally, there are some conditional functions that can take food-based variables and return a single true or false value. These are *allVariety*, *noVariety* and *someVariety*, which take a conditional statement involving some food-based terms, and return true if every, none, or some foods meet the required condition. For example, *someVariety(F>1000)* will return true only the ambient concentration of at least one food type in set F is above 1000.



You can nest functions too:- *someVariety(integrate(F>1000))* is also possible, and will return true if the number of prey of one or more food types encountered in the last timestep was greater than 1000.

9.12.5 Mixing food-based and non-food-based terms

The VEW executes statements that it detects are food-based a number of times – once for each food. You are free to include non-food-based terms in rules that are food-based, and they will be treated as you would expect. For example, if you have a standard non-food-based parameter P, you can write $V_{Food} = P * F$, and VEW will correctly execute the rule for each value of F, using the same scalar value for P.

9.12.6 Use of multiple food-sets

You can create as many food-sets as you like. However, if you write food-based rules, all the food-based terms in that rule must be linked to the same food-set. You cannot for example create foodsets F1 and F2, and in any way refer to them in the same rule. Even if F1 and F2 represent the same number of food entries, there is no guarantee that the first food in F1 is the same as the first food in F2, so the types are not compatible. There is one exception to this, described in 9.12.9

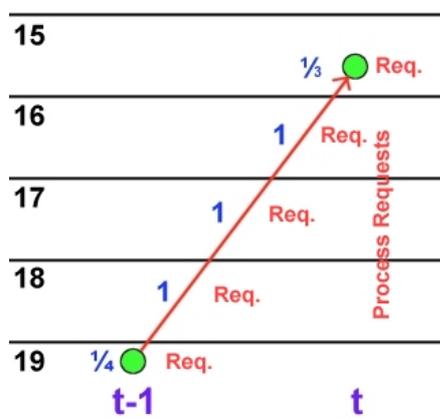
9.12.7 The ingest statement

The ingest statement is a food-based statement with the following form:-

ingest(Food-set, Threshold for each food, Rate for each food).

The statement takes one food-set variable, and two terms which it treats as food-based – if they are scalar, then the same threshold or rate is assumed for every acceptable food. The threshold is measured in **individuals** of each food type, and the rate is measured in **individuals per second** for each food. The threshold is the minimum concentration of prey per metre that can be seen – if the actual concentration is below that, then it is assumed the prey are too sparse to be seen, and no ingestion of them will take place.

In the diagram on the left, timestep *t* is the timestep we are considering, so the plankter is already at about depth 15. When the *ingest* statement is called, it examines the journey the plankter took between the last timestep and the current one. It then issues requests in each layer for ingestion, which the VEW proportionally assigns to each layer, depending on the fraction of time spent there, assuming a constant swimming speed.



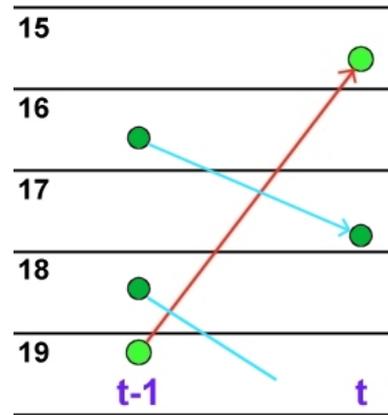
When all the plankters have been updated, (and all the ingestion requests have therefore been made), the requests are processed. Similarly to nutrient uptake (see 9.5.2), if the concentration of prey is insufficient, the requests will all be proportionally scaled down.

Here, we have been talking about predators ingesting prey at a individual-based level. The system translates this to an agent-based level, and we now consider the effects of the ingestion statement on the prey, and on the predator.

9.12.8 Predator and Prey Interaction

The previous section showed a predator swimming through a number of layers, making ingestion requests in each one. However, the prey may also be swimming between layers in a timestep. So when dealing with ingestion, we must firstly note that the concentrations of prey in each layer are computed dynamically, according to how much time the prey spent in each layer as it swam from place to place in the previous timestep.

So far, we have been discussing ingestion in terms of individuals. The VEW now has to translate this into a discussion about agents. When a prey-agent swims from place to place, we say that a proportion of the individuals that agent represents, are candidates for ingestion, in each layer, depending proportionally on what fraction on the journey was spent in that layer. The VEW firstly uses those calculations, comparing it to the sum of all requests, to determine whether there is sufficient prey.



If the concentration is above the threshold you specified, then the **sub-population size of the prey is reduced** by the *ingestion rate* (individuals per second), multiplied by the sub-population size, and the fraction of the prey's journey spent in that layer, and then it is scaled up to the timestep size.

Consider also that the prey being eaten contains chemicals, and since the predator has eaten some of the prey, the **chemical in the predator should be increased** by the amount of chemical in the prey it has eaten. The calculation for this is to multiply the number of individuals eaten (ie, the change in sub-population size for the prey), by the chemical in the prey's pool, and then divide it by the number of individuals the predator agent represents. It then appears in the incoming chemical pool – see 9.5.5.

9.12.9 How much did I eat?

Finally, the predator rules may need to know the number of individuals eaten in the previous timestep – this may differ from the requested amount since there may not have been sufficient food available to satisfy all the requests.

The VEW provides a food-based state variable called *IngestedCells*, which gives the number of individuals of *any* food that could have been eaten. Since this is a built-in variable, that represents all of the possible food at any time, it can be safely used in the same rule as other food-types, and the system will automatically match up each food-type in, say, food-set F, with the appropriate number of individuals in *IngestedCells*. This means you could write a rule as follows:-

$$Satisfaction = varietiesum(IngestedCells * F_preference)$$

Where *satisfaction* is a single term, *F_preference* is a food-based parameter for a number of different foods in some foodset F, and *IngestedCells* successfully matches with the foods mentioned in F.

9.12.10 Building the rules

Consider the following simple example:-

$$S = \frac{Ammonium_{pool}}{Ammonium_{max}}$$

$$F_{rate} = \frac{F_{max}(1 - S)}{3600}$$

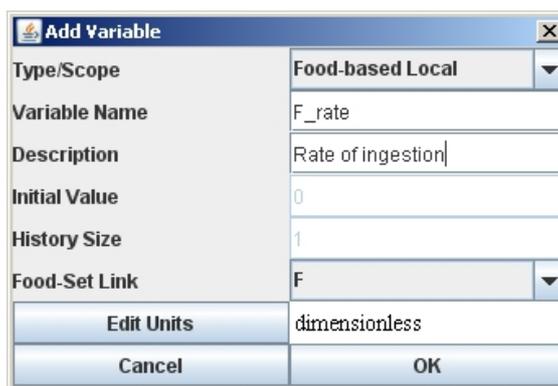
$$ingest(F, F_{min}, F_{rate})$$

$$Ammonium_{pool} = Ammonium_{pool} + Ammonium_{ingested}$$

These rules are designed to model hunger very simply, using Ammonium saturation as a satiation factor. If the predator is hungry, its ingestion rate will be closest to F_{max} , which is a theoretical maximum for each food-type, in individuals per hour.

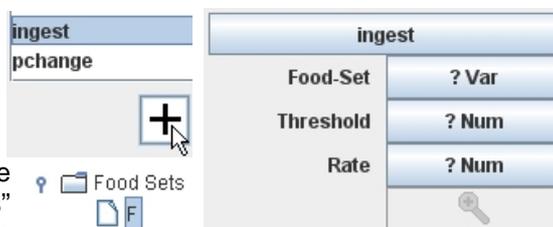
The first rule is simple. S stands for satiation, which is being measured as the fraction of the current ammonium content, over the maximum ammonium content, both used earlier in 9.5.3. If the Ammonium pool is full, S will be 1, otherwise it will vary down to zero. Since S is used immediately, it should be a local variable. This first rule is then a simple division of two vars.

The second rule is more complex. We start by defining an assignment, but before we create the variable F_{rate} , we must firstly create a food-set F , as in 9.12.1, for F_{rate} and F_{max} to be linked to. We do not care at this stage which foods F will contain – that is defined later in chapter 12. We can then create food-based parameters, F_{max} and F_{min} and a food-based local variable, F_{rate} which are both linked to our food-set, F .



Both food-based, and non-food-based parameters are both found under the heading “Constant Parameters” when choosing variables, and similarly, both types of local variables are found under the heading “Local Variables”. The rest of the rule is then simple – it is an assignment of a multiple with two parts: the first is the parameter F_{max} , and the other a subtraction of a value and the local variable S.

The third rule is the *ingest* rule. It takes three terms: the food-set (diet) to be ingested, the threshold for each food, and the rate for each food. These are identified on the detail panel in the usual way. The food-set is F , found under “Food Sets” when choosing the variable. The Threshold “? Num” in the example is set to zero, done in the same way as usual, and the rate is the local variable F_{rate} which we defined just now.

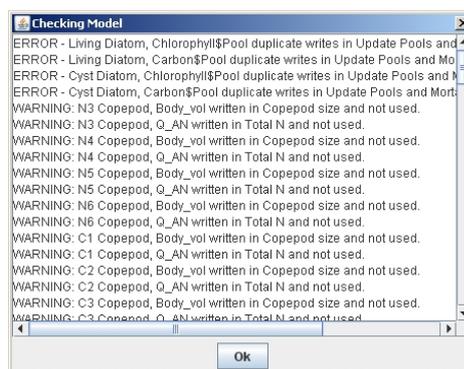


Finally, since ingestion, if the predators are successful, causes chemicals to arrive in the incoming pool, make sure you add a rule for dealing with that incoming chemical – see section 9.5.5.

9.13 The Syntax Checking Window

You leave the model page by clicking on one of the other traffic-light buttons at the bottom of the interface. On doing so, the VEW automatically performs some checks on your model, to reveal some potential mistakes. The errors it notices, and possible solutions are described here.

You do not have to fix any of these warnings, but you should check them to see if anything is highlighted that you should correct.



9.13.1 Duplicate Writes

This message tells you that the same variable has been assigned a value in two separate places, for the same stage and functional group. In this case, the second write will override the first, so you should try to combine the writes into one final statement.

There is one case where the error may be a false alarm, and this is when you use an *ifthen* statement, and if the condition is true, you perform the assignment. The VEW cannot tell at this stage whether the assignment is written – it depends on how the simulation runs. Even so, your model will be clearer if you can avoid duplicate writes.

9.13.2 Variable Written and not used

If you have assigned a value to a variable, but that variable never gets used anywhere else, then your model will run more slowly than it needs to. You should remove such assignments when you want to maximise the running speed of your model.

9.13.3 Variable Read Before Write

This is a common source of problems: if you have a local variable that you read in a function that occurs higher in the list of functions than the one that assigned a value to the local variable, then your model will certainly have unexpected behaviour. Pay particular attention to this message, which tells you the two functions that have this problem.

10. Modelling Language Quick Reference

This chapter gives some summary tables of all the statements, numerical and Boolean functions.

10.1 Statements

Name	Arguments	Description
Assign	Writeable Variable Numerical Expression	Set a variable to a value. Effects for local variable are immediate, effects for state variable are published in next timestep. See 9.3.3, part 3.
Change	State	Change state of the individual (and hence agent), at the end of the timestep. Remember to adjust chemical pools consequently. See 9.9.
Create	State of newborn, Number of creations, List of assignments	Create a number of new individuals, in a given state, and optionally assign any state variables for new individuals. See 9.8.
Divide	Number to divide into	Individual divides into a number of identical individuals at end of timestep. Remember to adjust chemical pools. See 9.7.
If Then	Boolean Condition Statement	If the Boolean conditions returns true, perform the specified statement (anything in this table). See 9.7.2.
Ingest	Food-Set Food-Based Threshold (indiv) Food-Based Rate (indiv/s)	Make requests for ingestion for a range of food types in each layer that was visited travelling from the position in the previous timestep, to the current timestep. In each layer, if the concentration of a food is above the specified threshold, then request food at the specified rate. Incoming chemical pools will be updated. See 9.12.
Pchange	Probability (Proportion) State	The individual has a probability of changing to the given state – in fact the agent is split in two parts at the end of the timestep, and the specified proportion are in their new state. See 9.10.
Release	Chemical Numerical amount (in timestep)	Release the specified amount of chemical into solution at the end of the timestep. Remember to adjust chemical pools. See 9.6.
Uptake	Chemical Numerical amount (in timestep)	Issue requests for the amount of chemical spread evenly over each layer visited while travelling from the position in the previous timestep, to the current timestep. Incoming chemical pools will be updated. See 9.5.

10.2 Numerical Functions

Name	Arguments	Description
abs	Numerical Expression	Absolute value.
acos	Numerical Expression	Arc-cosine
add	Any number of Numerical Expressions	Addition of any number of values
asin	Numerical Expression	Arc-sine
atan	Numerical Expression	Arc-tan
conditional	Boolean Expression Numerical Expression (t) Numerical Expression (f)	If the Boolean expression returns true, then conditional returns the first numerical expression, otherwise the second.
cos	Numerical Expression	Cosine
densityAt*	Numerical Expression	The density of the water at a specified depth.
depthForFI*	Numerical Expression	Return the closest depth at which the full irradiance (Wm^{-2}) is the specified value.
depthForVI*	Numerical Expression	Return the closest depth at which the visible irradiance (Wm^{-2}) is the specified value.
div	Numerical Expression (num) Numerical Expression (denom)	Division of numerator and denominator.
exp	Numerical Expression	Natural exponent.
fullIrradAt*	Numerical Expression	The full irradiance in the water at a given depth.
integrate	Numerical Expression	Integrates a depth-dependent expression across the layers the individual swam through between last timestep and the current timestep. See 9.12.
ln	Numerical Expression	Natural log.
log10	Numerical Expression	Log base-10.
max	Any number of Numerical Expressions	Maximum out of a list of expressions
min	Any number of Numerical Expressions	Minimum out of a list of expressions
minus	Numerical Expression	Reverse the sign of an expression
mul	Any number of Numerical Expressions	Product of any number of expressions

pow	Numerical Expression (x) Numerical Expression (y)	Raise x to the power of y.
rnd	Numerical Expression (x)	Return random number, $0 < r \leq x$ from the stream. Uses Marsenne-Twister algorithm.
salinityAt*	Numerical Expression	Salinity at a given depth
sin	Numerical Expression	Sine
sqrt	Numerical Expression	Square Root
sub	Numerical Expression (x) Numerical Expression (y)	Subtract y from x.
tan	Numerical Expression	Tangent
temperatureAt*	Numerical Expression	Temperature at a specified depth
UVIrradAt*	Numerical Expression	Ultra Violet irradiance (Wm^{-2}) at a specified depth
val	-	An absolute value
var	-	A variable
varhist	State Variable Numerical Expression	Return the value of a state variable from a number of timesteps ago.

10.3 Boolean Functions

Name	Arguments	Description
and	Any number of Boolean Expressions	Returns true only if all given expressions return true.
allVariety	Boolean Expression	For expressions that contain food-based terms, returns true only if the expression returns true for every food-types
equal	Numerical Expression (x) Numerical Expression (y)	Returns true only if $x=y$.
greater	Numerical Expression (x) Numerical Expression (y)	Returns true only if $x>y$
greaterEqual	Numerical Expression (x) Numerical Expression (y)	Returns true only if $x \geq y$
less	Numerical Expression (x) Numerical Expression (y)	Returns true only if $x<y$
lessEqual	Numerical Expression (x) Numerical Expression (y)	Returns true only if $x \leq y$
neq	Numerical Expression (x) Numerical Expression (y)	Returns true only if $x \neq y$
not	Boolean Expression	Calculates the Boolean expression and returns the converse.
noVariety	Boolean Expression	For expressions that contain food-based terms, returns true only if the expression returns false for every food-type.
or	Any Number of Boolean Expressions	Returns true if one ore more of the given expressions returns true.
somevariety	Boolean Expression	For expressions that contain food-based terms, returns true only if the expression is true for at least one food type.

11. Species Builder

The screen shot below shows the species builder. In the top left are the functional groups in your plankton community. Clicking on one of these will show the parameters for that functional group, and their default values, as specified when building the model, in the table below. On the top right are the species defined in your plankton community. Clicking on a species will cause its functional group to be highlighted in the top left of the screen, and the specific parameter values for that species to be displayed in the table.

Values in the table can be changed at any time. If you have a species selected in the top-right, then you can overwrite the parameter values of that species with those shown in the table by clicking the **Overwrite Species** button. Alternatively, you can add a new species by clicking the **Add Species** button.

Name	Description	A	B	Value
A_E	Slope of linear region of Arrhenius plot	-10000.0	0.0	-10000.0
Alpha_Ch1	Initial slope of photosynthesis light curve	7.9E-7	0.0	7.9E-7
C_minS	Min C pool for Si uptake to start	1.584E-8	0.0	1.584E-8
C_rep	Carbon threshold for cell division	1.76E-8	0.0	1.76E-8
C_starve	Carbon content threshold for starvation	6.24E-9	0.0	6.24E-9
C_struct	Structural carbon	8.5E-8	0.0	8.5E-8
k_AR	Half-saturation constant for uptake of nitrate a...	0.027	0.0	0.027
k_S	Half-saturation constant for Silicate uptake	1.0	0.0	1.0
Ndis	N spec rate of N dissolution	0.0042	0.0	0.0042
P_ref_c	Maximum carbon-specific rate of photosynthe...	0.14	0.0	0.14
Q_Nmax	Maximum Nitrogen to Carbon Ratio	0.17	0.0	0.17
Q_Nmin	Minimum Nitrogen to Carbon ratio	0.034	0.0	0.034
Q_remN	Increase in diss rate with T	2.95	0.0	2.95
Q_remS	Increase of Siremin with T	2.27	0.0	2.27
Q_S_max	maximum Si:C ratio	0.43	0.0	0.43
Q_S_min	Minimum Silicon to Carbon ratio	0.04	0.0	0.04

The table shows the name, description, and value of each parameter in the species. The value is represented by an A and B value, which is useful for allometric sets. The Value column shows the calculation Ax^B , where the X-Value is a number you can set by typing it into the **X-Value** box, or by choosing the default value of a parameter in the **Pick value of** box. Of course, if B is zero, then Ax^B is equal to A.

The **Add Many Species** button allows you to create many species with A and B value currently shown in the parameter table, but a different **x-value** for each. This allows convenient creation of any number of species, based on one varying parameter.

11.1 Creating a new species, from a functional group

1. Click on the functional group upon which the species will be based. The parameter table will change to show the default values in the 'A' column for each parameter in the functional group you selected.

Functional Groups	
Diatom	
Copepod	
Squid	
Predator	
Basal predator	

2. Make changes to the parameter table by double-clicking on the A values and typing. Notice that the 'Value' column is updated to the new values you give in the A column. When you have finished, click 

	A	
	-10000.0	0.0
e	7.9E-7	0.0
	1.584E-8	0.0
	1.76E-8	0.0
	6.24E-9	0.0
	8.5E-8	0.0

3. You will be asked to name the new species. Give a suitable name and click **OK** to create the species. The species builder now shows your new species in the list at the top right.

Input X

 Type name for species

11.2 Creating a new species, based on an existing species

1. Click on the species in the top right of the screen; your new species will be based on the species you select. Continue from step **A2** to edit the values, and add the new species.

Species
Default Diatom Variety
Default Copepod Variety
Default Squid Variety
Default Predator Variety
Default Basal Predator Variety

11.3 Creating a new species using allometry

1. Click on either a functional group, or a species, which your new allometric species will be based on. Recall that allometry allows you to set parameter values in the form Ax^B . To set the A and B values, double-click on values on the table as before.

A	B
30.0	0.0
1.225	0.0
33.3	0.0
1.62	2.0
3.0E-5	0.0
1.0042	0.0

2. Set the 'x' value. You can either type the value directly, or you can choose the default value of a parameter in the functional group. Notice that changing the value of 'x' causes the 'value' column of the table to change wherever a B value has been specified.

X-Value: Pick value of

A_F (Radius of Cell)
A_F (Radius of Cell)
Ammonium Semin (I)

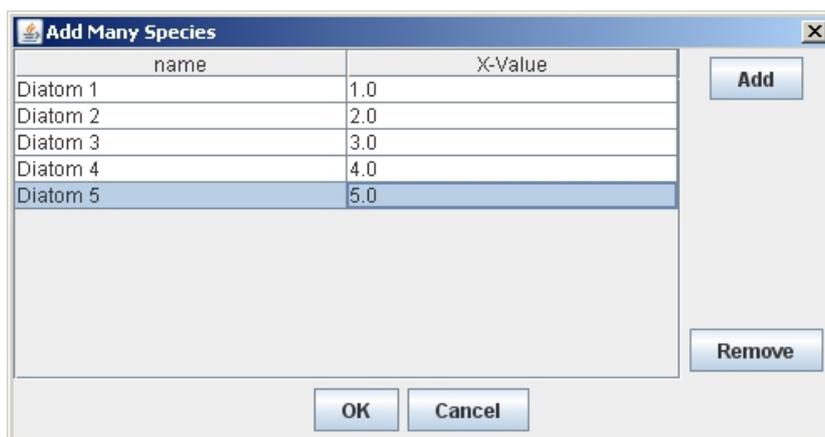
3. Click on  and continue from step **A3** to add the new species.

11.4 Creating an allometric set of species

1. Select the functional group, or species to use as a template, and set the A and B values as before. To create many species quickly, we will specify a number of different 'x' values, and one new species will be created for each. When the A and B values are set, click on **Add Many Species**.

A	B
30.0	0.0
1.225	0.0
33.3	0.0
1.62	2.0
3.0E-5	0.0
1.0042	0.0

2. A dialog appears, allowing you to add any number of species, each with a separate name and 'x' value. To add a new species, click on **Add**, and edit the name and x-values by double clicking the table cells.



3. You can remove species from the "Add Many Species" table by selecting the species to remove, (or a range of species, by holding Shift or Control while you select the species), and then click on **Remove**.
4. When you have finished creating the list of species to add, click **OK** to return to the main species builder window, and you will see the new species added in the list on the top right. Alternatively, pressing **Cancel** returns you to the main species builder without making any changes.

11.5 Editing an existing species

1. Select the species to be edited, and make the necessary changes in the parameter table. Click on **Overwrite Species** when you've finished.

	A	B
	-1.0000.0	0.0
e	7.9E-7	0.0
	1.584E-8	0.0
	1.76E-8	0.0
	6.24E-9	0.0
	8.5E-8	0.0

11.6 Renaming a species

1. Select the species in the top right of the screen, and click on **Rename Species**.



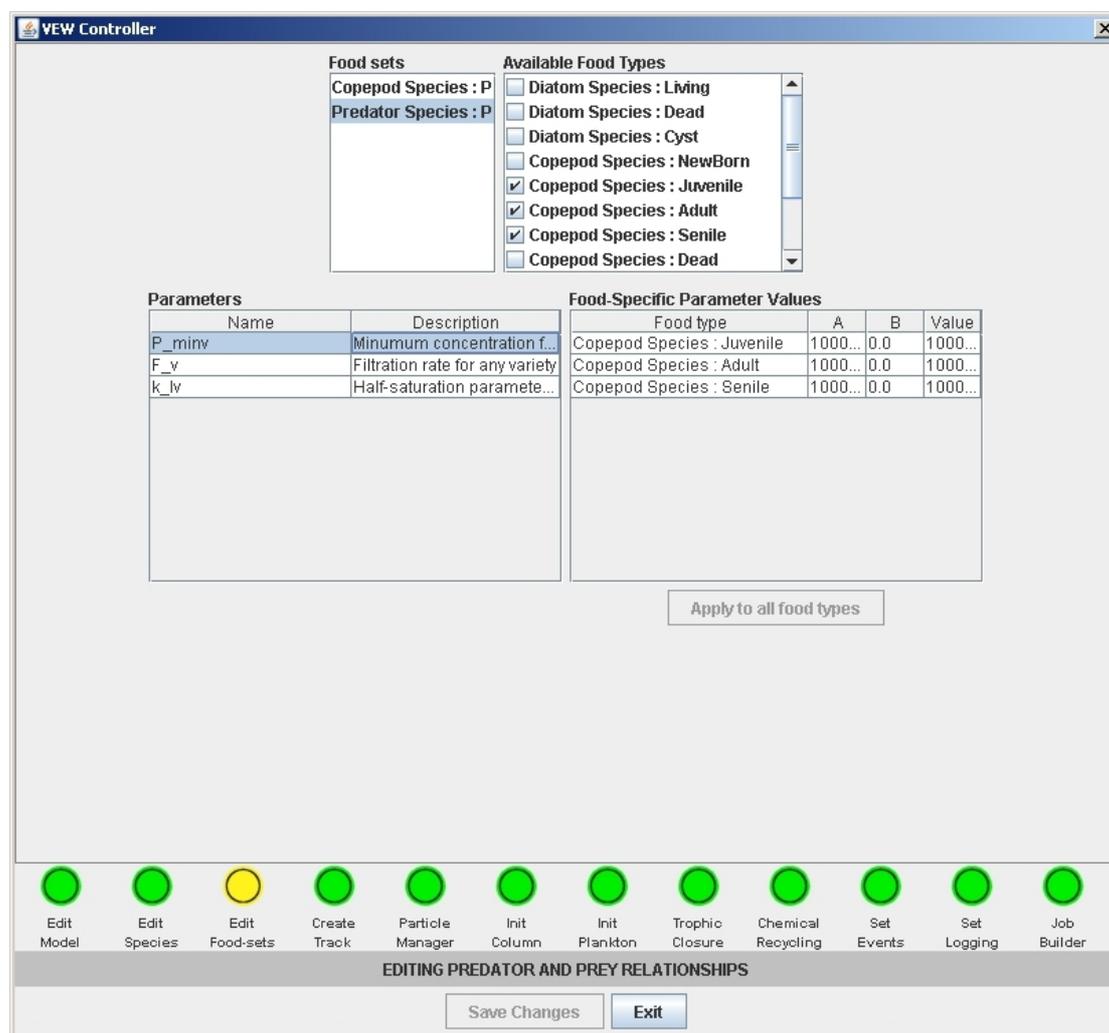
11.7 Removing a species

1. Select the species in the top right of the screen, and click on **Remove Species**.

12. Food-Sets

Having defined the species, we can now define the ingestion relationships between different species. Recall that in creating a model, you can create variables called “Ingestion Sets”. At the time of creation, this variable was defined to contain “any number of groups of plankton, specified by species and state”, however the specific content of the set was not specified, since species had not yet been specified. Now, however, we have all the necessary information to specify the predator-prey relationships.

The screenshot below shows the interface where these relationships are set. Specifically, this example shows a predator called “Predator Species”, which ingests juvenile, adult and senile copepods. The list at the top left shows all the ingestion sets and the species which will be performing the ingestion. Note that a species may have more than one ingestion set – it may ingest a different selection of food at a different time of year for example. The list on the top right shows “the menu” of all species-state combinations available for predators to eat.

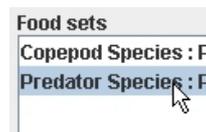


The list on the left shows the parameters that are relevant to the selected food-set. These were made when creating the model, where they were called “species-parameters”, and they were explicitly linked with an ingestion-set, so that for every member of the ingestion-set, (which we are now defining), a separate value of that parameter is required.

Having selected a parameter, a value is set for each type of food that this predator will eat, using the Ax^B form, where ‘x’ is the X-Value given when creating the species for the predator.

12.1 Choosing the food that a predator will eat

1. First, choose the food-set in the top left list; an ingestion set is a list of foods that has been used in one of the equations in the model, and having defined the species, we can now define what kinds of food should be included.
2. Now in the **Available Food Types** menu, tick the boxes for the foods that the predator should eat, for the food-set you have selected.



12.2 Setting the food-specific parameters

1. Having selected the ingestion set, click on the parameter to set. The table in the bottom right shows one row for each food that is ticked in the **Available Food Types** menu.
2. Noting that the **X-Value** is provided by the species of the predator, set the A and B values as desired – the resulting value is Ax^B .

Parameters	
Name	Description
P_minv	Minimum concentration f...
F_v	Filtration rate for any variety
k_lv	Half-saturation paramete...

Food-Specific Parameter Values			
Food type	A	B	Value
Copepod Species : Juvenile	1.0E-9	0.0	1.0E-9
Copepod Species : Adult	1.0E-9	0.0	1.0E-9
Copepod Species : Senile	1.0E-9	0.0	1.0E-9

12.3 Setting all the food-specific parameters to the same value

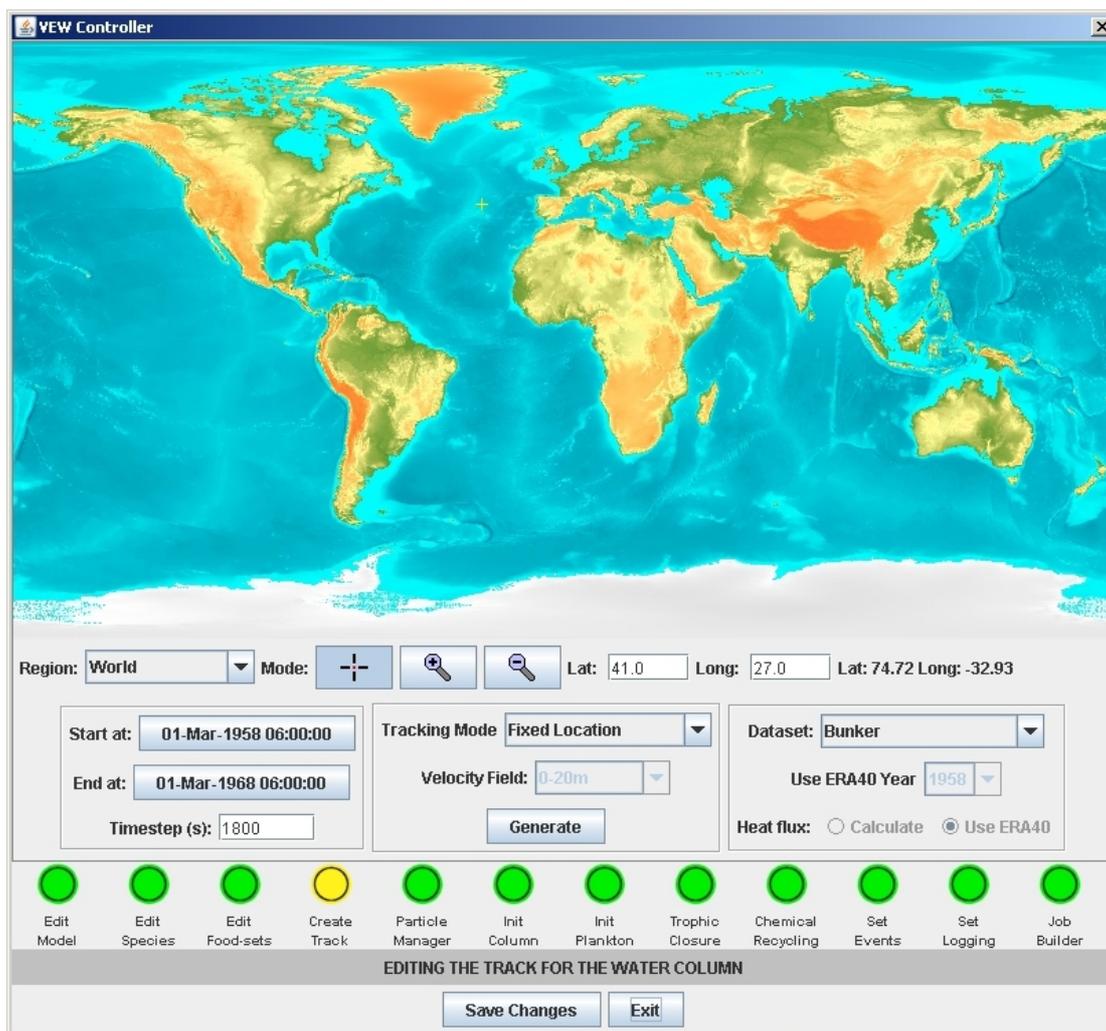
1. If you would like to specify the same value of a parameter for all different types of food (i.e., the parameter is not necessarily food-specific), then select the row in the value table, and click on **Apply to all food types**.

Food-Specific Parameter Values			
Food type	A	B	Value
Copepod Species : Juvenile	4.0E7	0.0	4.0E7
Copepod Species : Adult	4.1E7	0.0	4.1E7
Copepod Species : Senile	4.2E7	0.0	4.2E7

Food-Specific Parameter Values			
Food type	A	B	Value
Copepod Species : Juvenile	4.0E7	0.0	4.0E7
Copepod Species : Adult	4.0E7	0.0	4.1E7
Copepod Species : Senile	4.0E7	0.0	4.2E7

13. Mesocosm Track

Next, the starting time and duration of the simulation are set, along with the trajectory of the mesocosm and the selection of the climate forcing data to be used. The screenshot below shows the map, with a yellow cross marking the starting position of the mesocosm. The set of icons below the map allow you to zoom onto the map, or set the longitude and latitude for the starting point of the mesocosm manually. The panels at the bottom from left to right handle the starting and ending date of the simulation, the trajectory of the mesocosm, and the selection of forcing data respectively. Left clicking on the map also sets the initial longitude and latitude of the mesocosm. Right clicking on the map shows a pop up menu offering visualisation of the Bunker, ERA40, Levitus and OCCAM datasets used for the initial and boundary conditions for the mesocosm.



13.1 Setting the times of simulation

1. Click on either the “Start date” or the “End date” buttons in the bottom left panel. A calendar window appears where you can set the date:-

Start at: 01-Mar-1958 06:00:00

End at: 01-Mar-1968 06:00:00



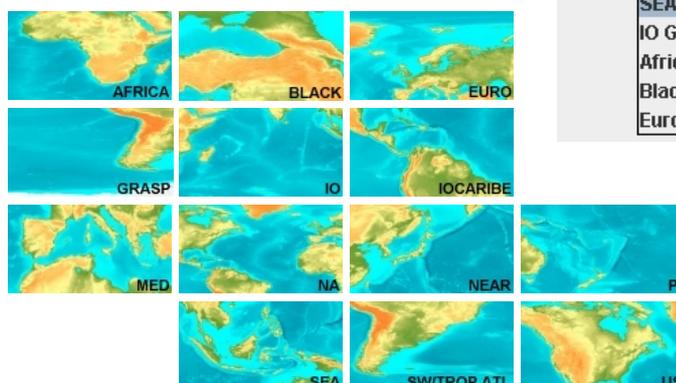
2. Select the time and date using the calendar. The drop-down year selection box contains the years 1958-2001, as these are the years contained in the ERA40 dataset. However, you can also click in the year box and type any year directly.
3. Click the “Accept” button to choose the date you have selected, or “Cancel” to return to the Track selection page without making any changes.

13.2 Navigating and choosing a starting point on the map

1. Firstly, note that hovering the mouse over the map causes the longitude and latitude label to update its value. It may be sufficient to simply point and click to choose the longitude and latitude.

Lat: 74.72 Long: -32.93

2. If however you need to see the map in more detail, there are a number of ways of doing so. One way is to select one of the pre-defined regions of the map, using the “Region” drop-down menu.



- Alternatively, you can manually zoom into, or out of the map, by clicking on one of the magnifying classes, and then clicking a point on the map. When you've zoomed to the right place, to select the point of longitude and latitude, click on the crosshair again, and then click on the point on the map.
 
- Finally, if you know the exact longitude and latitude where you'd like the mesocosm to start, you can type it directly by clicking in the text boxes.

Lat:	41.0	Long:	27.0
------	------	-------	------

13.3 Choosing the type of integration: fixed, forwards, or backwards

- Choose the method in the "Tracking mode" drop down menu. If you choose the forward or backward integration method, then the OCCAM circulation data is required to calculate the location of the mesocosm over time. A forward integration creates a track that starts at the location you select, whereas a backward integration creates a track that ends at the location you select.

Tracking Mode	Fixed Location
	Forward Integration
	Backward Integration
	Fixed Location

Velocity Field:	0-20m
	182-222m
	222-269m
	269-324m
	324-389m
	389-465m
	465-555m
	0-100m AVG
	0-555m AVG
- If this is required, OCCAM provides velocity fields at a range of different depths, and averages for 100m, and 555m. Select the desired depth from the Velocity field menu.
- If you'd like to preview the track the mesocosm will take, click on

Generate

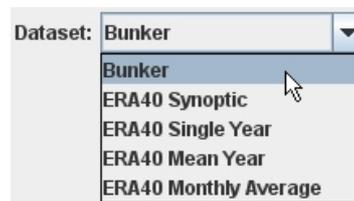
- The track will appear on the map; the large yellow cross is the starting point, each small yellow cross marks a year of time, and the large red cross marks the end of the simulation.



13.4 Choosing the data set for the boundary conditions

1. Choose the dataset from the drop-down list. These datasets provide the simulation with climate data – cloud cover, heat fluxes, wind speed, air temperature and relative humidity.

The datasets contain the following data, all at a geographical resolution of 1° by 1° longitude and latitude.



Name	Description	Geography	Time range	Frequency
Bunker	Climate Data	North Atlantic	Jan-Dec	Monthly
NOAA '01	Hydrological Data Physical	Global	Jan-Dec	Monthly
NOAA '01	Hydrological Data Nutrients	Global	Single Val	Single Val
ERA40 Synoptic	ECWMF Re-analysis	Global	1958-2001	6-hourly
ERA40 Single Year	Any year of ERA40 repeated.	Global	1958-2001	6-hourly
ERA40 Mean Year	Average of all ERA40 years	Global	Jan-Dec	6-hourly
ERA40 Monthly Avg	Average of all ERA40 months.	Global	Jan-Dec	Monthly

2. If you select the ERA40 Single Year, you will also need to select the year of the ERA40 data set you would like to repeat for the duration of the simulation.



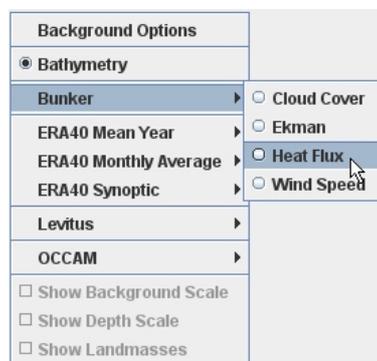
3. The ERA40 dataset comes with its own ocean heat flux data. However, the fluxes can also be calculated using the air temperature, relative humidity (also provided by ERA40), and the sea temperature that emerges from your simulation as you run it. If you would prefer this dynamically calculated heat flux to be used, select **Calculate** heat flux, or if you'd prefer to use the heat fluxes provided in the ERA40 data set, choose **Use ERA40**.



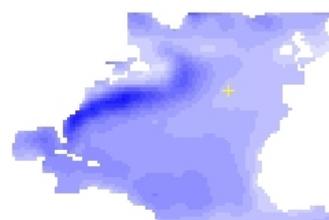
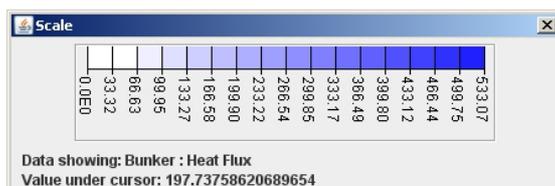
14. Visualising the datasets

14.1 Basic Visualisation

- Right click on the map. A pop-up menu appears, offering you all the available datasets. Hovering the mouse over a dataset will show the variables in that dataset, all of which can be visualised. This example is taken from Bunker – recall that Bunker only gives data for the North Atlantic.

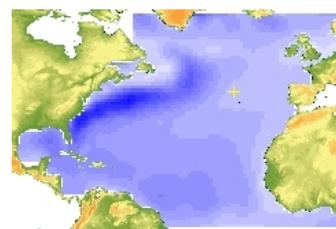
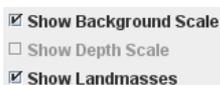


- Selecting a variable will, (after a short pause), update the map to show that data, along with a floating window that shows the scale. Notice that moving the mouse over the map will update the value shown on the scale. The data plotted is interpolated over time, to give values at the simulation start date that you specified earlier.



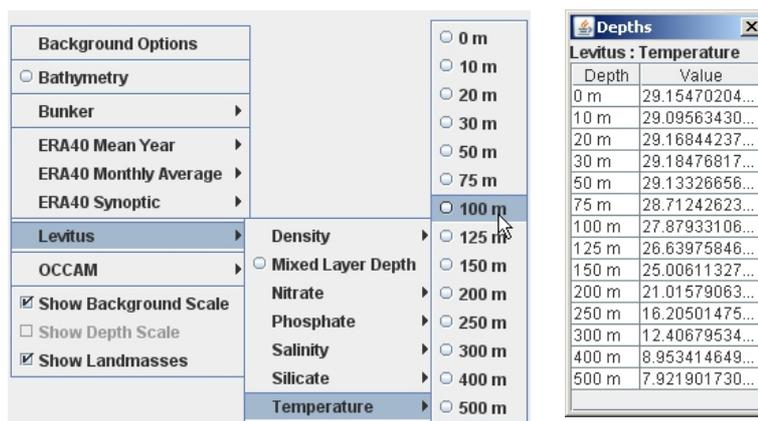
14.2 Overlaying landmasses

- Having visualised some data, some extra options become available on the pop-up menu. The “Show Landmasses” tickbox superimposes the land onto the data map, as shown on the right. The “Show Background Scale” tickbox ensures that the scale window is visible, if you have previously closed it.



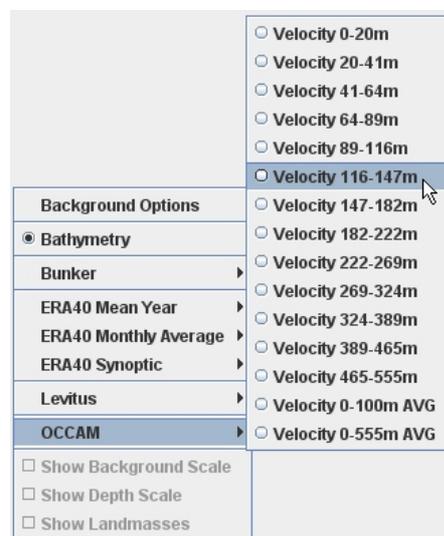
14.3 Visualising layered data

- Variables provided by levitus data are supplied in 14 layers (except for the mixed layer depth). To visualise these, you must select the layer to use for the background on the popup menu; the scale window appears as before, showing the value at the select layer, at the location under the mouse pointer. A second window will also be displayed, which shows the value of the variable for all of the layers, at the location your mouse pointer is currently over. The "Show Depth Scale" on the pop-up menu can be used to ensure this window is visible, should it get closed.

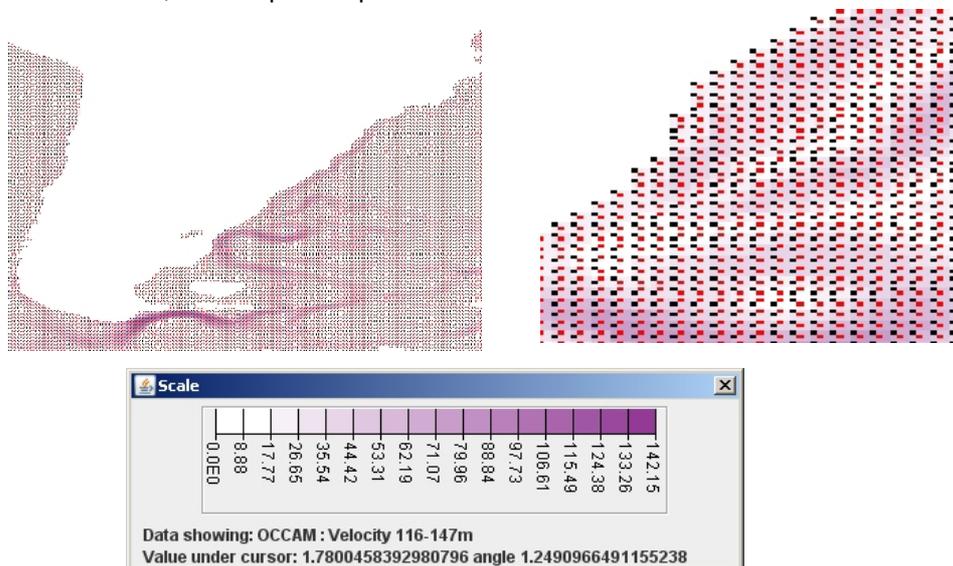


14.4 Visualising vector data

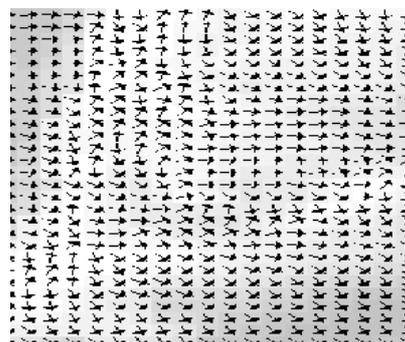
- Some of the variables are supplied in vector form. The ERA40 synoptic wind speed is one such variable, although since VEW only requires the magnitude of the windspeed, the mean year and monthly averages files have been reduced to scalar quantities. The other vector variable occurs in the ocean circulation data (OCCAM), which is supplied in a layers described earlier.
- Note also the ERA40 wind speed is, like all ERA40 data, at resolution of $1^{\circ} \times 1^{\circ}$ longitude and latitude. OCCAM however is supplied at a higher resolution of $\frac{1}{4}^{\circ} \times \frac{1}{4}^{\circ}$.



- When selecting a vector based variable, the background colour of the map shows the magnitude of the vector, and an arrow overlaid on the background shows the direction. The angle of the vector in radians is also shown in the scale window. For the OCCAM data, the high resolution presents a problem for displaying the vectors clearly; as a temporary measure, when zooming into OCCAM data, the red point represents the arrow head.



- The ERA40 wind speed data (right), since it is at a lower resolution, is easier to represent, however the presentation of vectors is an area that has been noted for future improvement.



15. Particle Manager

The Virtual Ecology Workbench creates agent-based simulations, where each agent is a Lagrangian-Ensemble agent. It follows its own trajectory, and has its own internal state. It represents a dynamic number of individuals that are all doing the same thing. To improve the quality of the results in terms of demographic noise, you should increase the number of agents in the species of interest, and have each agent represent a smaller number of individuals. However this comes with a cost in terms of computer memory, and the time taken to run simulations with more agents. Similarly, if you have types of agent whose individual properties are less interesting, (pellets, or detritus for example), then you can save memory cost, and increase performance by merging these into fewer agents, representing more individuals per agent.

The screenshot shows the 'VEW Controller' window with the 'Particle Manager' configuration interface. At the top, there are dropdown menus for 'Merge', 'Diatom Species', and 'Cyst'. Below these, there is a configuration for merging agents: 'Merge smallest agents if agent count > 40'. There are radio buttons for 'in Column', 'avg per metre in Mixing Layer', and 'per Metre' (which is selected). Below this, there are checkboxes for 'Continuous Management' and 'Whole of column or, between depths'. The 'Continuous Management' section shows a date range from '01-Jun-1961 06:00:00' to '01-Mar-1968 05:30:00'. The 'Whole of column' section shows depths from '0' to '499'. There are 'Add Rule' and 'Overwrite Rule' buttons. Below these is a table of rules:

Plankton Type	Rule	Criteria	Scope	From	To	Top	Bottom
Living Diatom Species	Split	20	Layer	01-Jun-1961 06:00:00	01-Mar-1968 05:30:00	0	499
Cyst Diatom Species	Split	20	Layer	01-Jun-1961 06:00:00	01-Mar-1968 05:30:00	0	499
Dead Diatom Species	Merge	1	Layer	01-Jun-1961 06:00:00	01-Mar-1968 05:30:00	0	499
Dead Copepod Species	Merge	1	Layer	01-Jun-1961 06:00:00	01-Mar-1968 05:30:00	0	499
Pellet Copepod Speci...	Merge	1	Layer	01-Jun-1961 06:00:00	01-Mar-1968 05:30:00	0	499
Living Diatom Species	Merge	40	Layer	01-Jun-1961 06:00:00	01-Mar-1968 05:30:00	0	499
Cyst Diatom Species	Merge	40	Layer	01-Jun-1961 06:00:00	01-Mar-1968 05:30:00	0	499

Below the table is a 'Remove Rule' button. At the bottom of the window is a toolbar with icons for: Edit Model, Edit Species, Edit Food-sets, Create Track, Particle Manager (highlighted in yellow), Init Column, Init Plankton, Trophic Closure, Chemical Recycling, Set Events, Set Logging, and Job Builder. Below the toolbar is the text 'EDITING AGENT MANAGEMENT RULES' and 'Save Changes' and 'Exit' buttons.

All of these possibilities are available through the particle management interface:-

The table shows all of the rules currently configured for this model; you can create as many rules as you like. Clicking on a rule shows its details in the top half of the screen, allowing you to change the details and overwrite the rule, or to add a new rule with the options you have selected. There are three different rule types, Merge, Split and Maintain, which we will discuss below. These rules can be applied separately to each state of each species, at certain times of year, or repeatedly for the same selection of the year, each year. For the merge and split rules, these can be applied to the whole mesocosm, as an average throughout the mixed layer, or metre-by-metre, between specified depths. Additionally, the dynamically varying turbocline can be used for the bottom depth of the particle management rule.

Obviously, where there are no particles to operate on for a particular rule, then the rule must be ignored; it is not possible to perform a split when there are no agents to split. Also note that if any agent represents a number of individuals that is less than 1×10^{-10} , it becomes too small to represent accurately, and is removed from the simulation.

15.1 Adding a new split rule

- The split rule is used to ensure that there are sufficient agents of a certain species and state, at a given time and location. Select "Split" from the top left drop-down menu. Also select the species and state that this rule concerns.



- You must now specify the number of agents you require, either in the mesocosm, the average number of agents in the mixed layer, or the number per metre. Click the appropriate button, type the number in the text box. The largest agents will be split to meet the criteria.



- Next select the times when this particle management rule should be active. If it is to be applied all of the time, then tick **Continuous Management**. Otherwise, untick that box, and click on the two date buttons to specify a start and end date for this rule. If you would like the rule to be applied between the dates you specify but regardless of the year, then tick **execute every year**.



- If you selected "per metre" in step 2, then you can additionally specify a range of the mesocosm in which to apply the particle management rule. To apply it to the whole column, tick **Whole of column**. Otherwise, untick that box, and select the top and bottom depths from the drop-down menus. If you would like to perform this particle management down to the turbocline depth, as it changes through the simulation, rather than specifying a fixed bottom depth, then tick **Use Turbocline**.

- Finally, to add the rule to the list, click on .

15.2 Adding a new merge rule

- The merge rule is used to reduce the number of agents in cases where there is insufficient value or interest in modelling them as individuals, rather than en masse. Adding such a rule is very similar to adding a split rule; select "Merge" from the top left menu, and select the species and state that you want to merge.



- The only difference is in the text description, which states that the smallest agents found in the mesocosm, the section of the mesocosm above the turbocline, or per metre, will be merged if the number of agents exceeds the value you specify. Continue from step A3 to add set the times and depths of merging, and to add the rule.

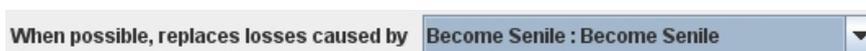


15.3 Adding a new maintain rule

- The maintain rule is used where your model has a specific way that agents can effectively leave the simulation; starvation is a common example. A typical situation is where you have decided to merge dead agents, and there is a way in which an agent can change state to "Dead", immediately get merged, and thus you have fewer "Active" agents than you originally did.



In these situations, you may like to specify that when an agent leaves the simulation due to a particular rule in your model, a split should be carried out so that another agent in the same species and state replaces the lost one. To make such a rule, select the "Maintain" option in the top left drop-down, and choose the species involved. The state is not chosen at this point because we will instead choose the rule that causes the agent to leave the simulation, regardless of the state it started in.



- Next, choose the rule that causes agents to leave the simulation. The function name, and the equation name are both shown. When an agents leaves its state using this equation, the largest agent of the originating state will be split to replace the lost agent. If no such agents are available, the system waits until there are available agents, so in that case, the loss will be made up next time another agent enters that state. Time and depth settings are not applicable to maintenance rules.

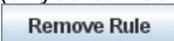
15.4 Editing existing rules

- Clicking on a rule in the table will show the details of that rule in the top half of the screen. You can then edit the rule, using the steps above, and when you have finished editing, click  .

15.5 Adding a rule based another one

- Click on the rule in the table, and make any changes to the rule as before. You can then click on  to add the edited rule to the list.

15.6 Removing rules

- Select the rule in the list, (or you can select more than one rule by holding control as you click). Clicking on  removes all selected rules.

15.7 A note of caution regarding particle management

It is very important to understand that a split rule becomes a source of agents, particularly when considering transitions of particles through states. If you specify that you require a minimum of 20 particles in a given species and state, then the moment a single agent enters that state, it will be split into 20 agents, each one a twentieth the size of the original agent, even if others are due to arrive in this state soon. Similarly, if later on an agent leaves a state, so that there are 19 left, then the largest particle will be split into two. This could happen a large number of times – in fact it continues until a set of agents all leave the state together, leaving no remainders, or when the agents start to represent such a small concentration of individuals, (less than 1×10^{-10}) that they are removed from the simulation.

If splits are occurring in this sort of way, the number of agents produced may force you to also merge agents. When agents are merged, the number of individuals represented is summed, and all the other internal properties are computed by a weighted averaged, depending on how many individuals were represented by each agent. This may be fine for dead plankton and pellets and even simple diatom, where a weighted average of their pools may not make a great difference. However, for some plankton it may be a problem; consider merging two particles where age is one of their internal properties. They may never reach an age threshold, and particle management would have interfered with the biology of the model.

These are all issues being addressed as we continue to develop the Lagrangian Ensemble method of integration. In the meantime, particle management rules should be used with great care and consideration of the potential consequences of splits and merges.

The maintenance rule was added as a safer option for splitting, since it only splits when you have specified that an agent has effectively been lost from the simulation, thus there is no great swell in the number of agents; it simply corrects the number of agents when it can, when a loss has already occurred.

16. Initialising the Mesocosm

This section deals with the initial chemical and physical state of the mesocosm. VEW comes with Levitus/NOAA initial profiles for nitrate, phosphate, silicate, density, salinity and temperature. It also comes with initial values of the turbocline, which is of minor significance to us, since VEW simulations compute the turbocline value each time step. The datasets can be visualised completely from the Track page described in section 3. On this page, we allow importing and editing of the data from Levitus, to set up the initial conditions of the mesocosm.

The screenshot shows the 'VEW Controller' window. On the left, under 'Items that require initialisation:', a list includes Ammonium, Boron, Carbon, Chlorophyll, Density, Nitrate, Salinity, Temperature, and Turbocline. Below this, 'Data sets available for import:' lists Levitus, Nitrate, Phosphate, and Silicate. A 'Preview' button is visible. In the center, a graph plots a variable against depth from 0 to 500 metres. On the right, a table shows depth and value data. At the bottom, a row of buttons includes Edit Model, Edit Species, Edit Food-sets, Create Track, Particle Manager, Init Column, Init Plankton, Trophic Closure, Chemical Recycling, Set Events, Set Logging, and Job Builder. The 'Init Column' button is highlighted in yellow. Below the buttons is the text 'EDITING INITIALISATION OF THE COLUMN PROFILE' and 'Save Changes' and 'Exit' buttons.

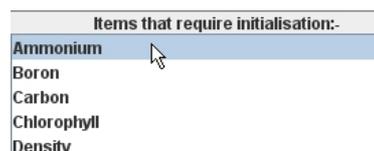
Depth /m	Value
0.0	9.010725334...
1.0	9.010725334...
2.0	9.010725334...
3.0	9.010725334...
4.0	9.010725334...
5.0	9.010725334...
6.0	9.010725334...
7.0	9.010725334...
8.0	9.010725334...
9.0	9.010725334...
10.0	9.010725334...
11.0	9.010725334...
12.0	9.010725334...
13.0	9.010725334...
14.0	9.010725334...
15.0	9.010725334...
16.0	9.010725334...
17.0	9.010725334...
18.0	9.010725334...
19.0	9.010725334...
20.0	9.010725334...
21.0	9.010725334...
22.0	9.010725334...
23.0	9.010725334...
24.0	9.010725334...
25.0	9.010725334...
26.0	9.010725334...
27.0	9.010725334...
28.0	9.010725334...
29.0	9.010725334...
30.0	9.010725334...
31.0	9.010725334...
32.0	9.010725334...

The top left list shows all the field variables (and the turbocline) in your model that need initialising. The bottom left menu shows the available data that can be imported for the purposes of initialising the column. When the “Preview” button is selected, the Levitus variable selected in the bottom left is shown in the graph and table on the right. Levitus data is available in the form of a monthly average, at a resolution of 1°x1°. The values shown in the table are interpolated both chronologically, to the starting time of your simulation, and geographically, to the starting longitude and latitude of the mesocosm, as set on the Track page, section 3.

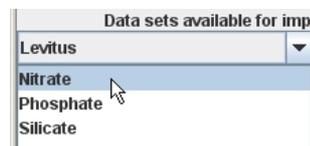
When “Preview” is not selected, then the graph and table instead show the current initialisation of the variable selected in the top left list. The “Use Data” button imports data from the source highlighted in the bottom left, and uses it as the initialisation for your variable highlighted in the top left. You can edit values in the table, and if you highlight two rows in the table, you can perform linear interpolation between them.

16.1 Initialising a field using the Levitus dataset

1. Select the field variable to be initialised in the list in the top left of the screen. The current profile will be shown in the graph and table on the right.



2. Select the field variable from the Levitus dataset that you wish to import. Note that the names do not have to match; for test purposes if you add a new chemical that has no matching field in the Levitus data, you can pick one that it does contain as a substitute. Also note that the drop-down list of sources contains just the one entry – Levitus. Future versions may add new data sources should they become available.

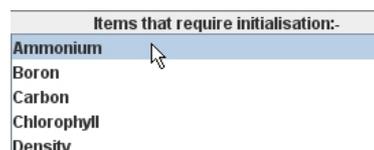


3. If you wish, you can preview the data you are about to import, before you import it. If so, click on  and examine the data in the graph and table on the right.

4. To use the data for initialising the variable you selected, click on .

16.2 Editing the data manually

1. Select the field variable to be edited in the list in the top left of the screen. The current profile will be shown in the graph and table on the right.



2. Click on a row in the table, and type a new value.

100.0	9.010725334...
101.0	9.010725334...
102.0	9.01072533498
103.0	9.010725334...
104.0	9.010725334...
105.0	9.010725334...

16.3 Interpolating between values

1. To interpolate between two values, firstly set the top and bottom values as described. Then select the top and bottom cells by holding **control** and clicking on them.

99.0	9.010725334...
100.0	10
101.0	9.010725334...
102.0	9.010725334...
103.0	9.010725334...
104.0	20
105.0	9.010725334...

2. Click on , which becomes available when you select the two rows, and a linear interpolation is carried out.

99.0	9.010725334...
100.0	10
101.0	12.5
102.0	15.0
103.0	17.5
104.0	20
105.0	9.010725334...

16.4 Setting the maximum depth of the mesocosm

1. Setting the value for "size of column" will allow modelling of deeper water. The table will be updated, so if you increase the size, make sure that you initialise all of it with values. The import data is only available down to 500m; when you extend the mesocosm further than that, the bottom-most values will be copied.



17. Distributions of Plankton

The next part of the VEW concerns initialising sets of plankton and adding them to the mesocosm. It involves defining their initial state, and the values of their internal variables, choosing how many agents should be placed in the column, where the agents are, how many individuals each agents represent, and when the agents should appear in the mesocosm.

Name	Description	Value (lo)	+random(x)
Ammonium\$Pool	Ammonium Pool	2E-9	0.0
Boron\$Pool	Boron Pool	0.0	0.0
Carbon\$Pool	Carbon Pool	0.0	0.0
Chlorophyll\$Pool	Chlorophyll Pool	0.0	0.0
E_p	Energy pool	1E-4	0.0
I_m	Photoadaptive Variable	10	0.0
Nitrate\$Pool	Nitrate Pool	2E-9	0.0

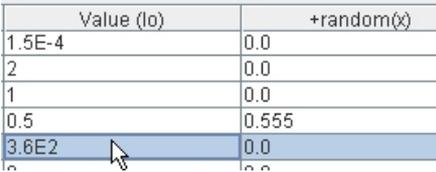
Name	Agents/metre	Individuals/Agent	Top depth (m)	Bottom depth (m)	Date/Time
Diatom Species : Living	20	25000.0	0	200	01-Mar-1958 06:00:00
Copepod Species : L...	3	17.0	0	200	01-Mar-1958 06:00:00
Predator Species : Exi...	1	0.0	0	200	01-Mar-1958 06:00:00

The lower table shows the distributions of plankton already in place – three are shown here, and for each one, how many agents per metre, how many individuals per agent, the upper and lower depth bounds where they should be distributed, and the time at which the distribution should occur – in this case the start of the simulation.

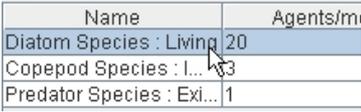
The upper half of the page is used to define the properties of plankton to be added; the species and state are at the top, and the table just below that shows all of the variables that need to be initialised – these should be thought of as referring to an individual plankter of the species and state chosen. The values shown here by default are those that were given when the variables were created. The variable values can either be set to a single value, or alternatively a sample from a random distribution can be added to the given value, meaning that the different agents in the distribution may have a spread of values for any of their variables.

The date at which a distribution should appear is useful for creating a “biological event”, where a new species may suddenly appear, to model the introduction of a foreign type of plankton for instance.

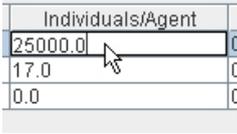
17.1 Adding a new distribution of plankton

- Choose the species and state for the plankton you would like to introduce. 
- The table will be populated with default values for all the variables you need to initialise – these defaults are copied from when the variables were created. If you would like the all agents in the distribution to have the same values (that is, every individual on every agent), then you only need to define the “value” column. If you would like a spread of values, then define the “+random(x)” column as well, and the final value will be $value+random(x)$. 
- Select the date at which the plankton should appear 
- Set the top and bottom depth boundaries. 
- Set the number of agents per metre, and the number of individuals per agent. 
- Click on  , and the distribution will be added to the table.

17.2 Editing an existing distribution (Standard Method)

- Click on the distribution you would like to edit. The details for that distribution will be shown in the top part of the screen. Edit the details as described in section A. 
- When you have finished editing, click on  . The table of distributions will be updated with the changes you have made.

17.3 Editing an existing distribution (Quicker Method)

- Alternatively, if you only need to edit the agents per metre, the depth range, or the date and time, you can double-click directly on the values in the distribution table and edit them. Using this method is fast, but it cannot be used for editing the species, state or initial variable values. 

17.4 Using an existing distribution to create a new one

1. Choose a distribution, and make any necessary changes as described in section B.

Name	Agents/m
Diatom Species : Living	20
Copepod Species : L...	3
Predator Species : Exi...	1

2. Click on  , and the new distribution will be added.

17.5 Removing a distribution

1. Select the distribution to be removed in the table. (You can also hold *control* or *shift* to select more than one distribution).

Name	Agents/m
Diatom Species : Living	20
Copepod Species : L...	3
Predator Species : Exi...	1

2. Click on  to remove the distribution(s).

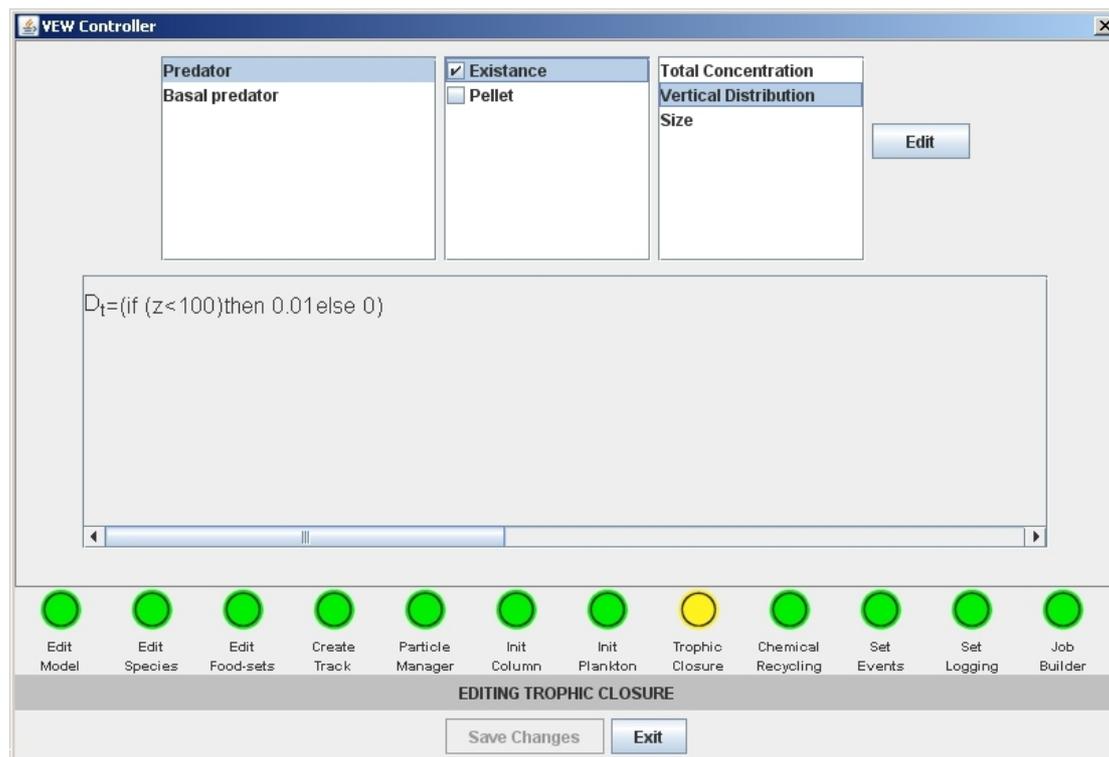
17.6 A note about top predators

Note that the number of individuals per agent is ignored for species that you have defined to be top predators. In this case, the number of individuals per agent is specified exogenously, with a value that can vary over time. See the later section on trophic closure.

18. Trophic Closure

When creating functional groups, you have the option to state that a functional group is a “top predator”. This means that the number and distribution of that functional group will be prescribed exogenously, rather than emerging from the simulation as it runs.

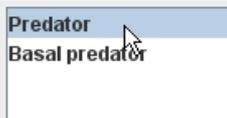
With all simulations, some sort of closure is needed, since no model will simulate all the species in the ocean. Traditionally, this closure has been defined implicitly as a loss term, which you can easily do in the VEW as well – for example, by using a pchange in your model. However in the VEW, you can also define the top predators *explicitly*, giving them rules for behaviour, but prescribing how many there should be, and where.

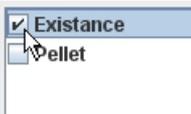


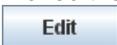
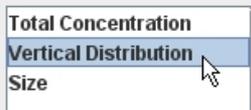
Above is the trophic closure interface. The top left list shows all functional groups marked as top predators. The next list shows the states of that predator. You may not want to prescribe the distribution for all states of a predator. For example, a common use of states is for representing pellets and if you use this method for the top predator to produce pellets, then the size and distribution of the pellets should emerge from the simulation, even if the distribution of the predators is prescribed.

The third list shows the fixed set of three functions which you must define to complete the trophic closure. The total concentration is the total number of individuals in the mesocosm. The vertical distribution is a function of depth; when given a depth “z”, it must return the fraction of the total concentration that should be found in that layer. Thirdly, the size function is any generic representation of size – it could represent volume, weight or diameter for instance. This is the only one of the three functions that can be used when defining the behaviour of the top predator – it is the variable S_i .

18.1 Defining the trophic closure, step-by-step

1. You may have more than one top predator in your model. For example, you may have one background (basal) predator that is present all year round, and another seasonal predator. Furthermore, you may have defined different species of a predator. For each species that effectively has been ticked as "top predator", you need to define the exogenous closure rules. So for each entry, click on it in the top left list.
 

2. Next, tick the states that require exogenous closure rules. If your predator produces pellets, the sizes and distributions of these should emerge from the simulation, rather than being prescribed here.
 

3. Now click on each rule type. The current rule is shown in the panel in the centre of the page. To edit each rule using the standard rule editor, click on  .
 

4. Repeat this process for each rule, and each predator species.

18.2 The Total Concentration Rule, N_t

1. This specifies the total number of individuals of the given predator in the mesocosm. For basal predators, this tends to be constant, whereas for seasonal predators, you may want this to vary over time. A useful variable here is d_{year} , which gives the day number of the current year, with 1st January being 1.

18.3 The Vertical Distribution Rule, D_t

1. This rule specifies where the individuals should be located. VEW assigns a number of individuals to each metre, and calls this rule for each. The rule shown in the screenshots is $D_t = \text{if } z < 100, 0.01, \text{ else } 0$. This specifies that for all depths, metre-by-metre, down to 100m, 1% of the total number of individuals (N_t) should be allocated. As the rule is called metre-by-metre, the sum of D_t for every later will be 1, as it should be.

18.4 The Size Rule, S_t

1. The size rule is the only one of the three rules which can be used in the behaviour definitions for the top predator – (the functional group rules). Commonly, top predators will have behavioural rules for ingestion, and this rule allows a size variable, defined here exogenously, to be used in ingestion equations, and others.

19. Chemical Recycling

When running simulations of 20 years or more, we have found that in our models plankters may absorb nutrients and eventually sink below the seasonal turbocline. They take the nutrients with them, and the nutrients are remineralised as the plankters sink. However, nutrients that are remineralised below the mixed layer may not ever return to it, depending on the sinking speed of the plankters, and the variation of the mixed layer depth. Thus, although the total nutrient budget for the mesocosm may be balanced, the distribution of that nutrient may change, resulting in there being less nutrient in the mixed layer, and eventually it may cause side-effects such as reduced diatom blooms due to nutrient limitation.

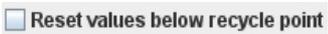
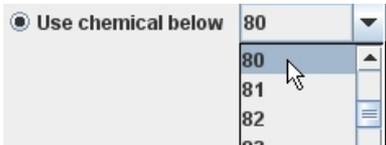
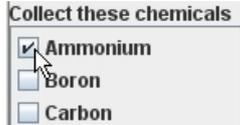
VEW allows a correction to be made to avoid such cases. Of course, this is an artificial fix – it does not represent any real biological, chemical or physical process. However, for the purposes of running numerical experiments, it may be a useful correction, and the VEW allows you to make it at an appropriate “quiet season”, where it is least likely to cause other unwanted implications.

The screenshot shows the 'VEW Controller' window with the 'Use Chemical Recycling' option checked. The start time is set to '01-Jan-1959 06:00:00'. Under 'Update yearly from start time', 'Update yearly from start time' is selected. Under 'Auto-detect start-depth', 'Deepest Each Year' is selected. The 'Use chemical below' dropdown is set to '120', and 'Reset values below recycle point' is unchecked. In the 'Collect these chemicals' section, 'Ammonium' and 'Nitrate' are checked, while 'Boron', 'Carbon', and 'Chlorophyll' are unchecked. A row of 12 circular icons is shown, with the 'Chemical Recycling' icon highlighted in yellow. The 'EDITING CHEMICAL RECYCLING OPTIONS' section contains 'Save Changes' and 'Exit' buttons.

The chemical recycling menu is shown above. At the top, we select whether chemical recycling is activated, and if so, when the first chemical recycling event should happen. Next, we select the time that chemical recycling events should happen. Then we select the depth, below which remineralised nutrient may potentially be recycled. Finally, we select the chemicals that are to be recycled.

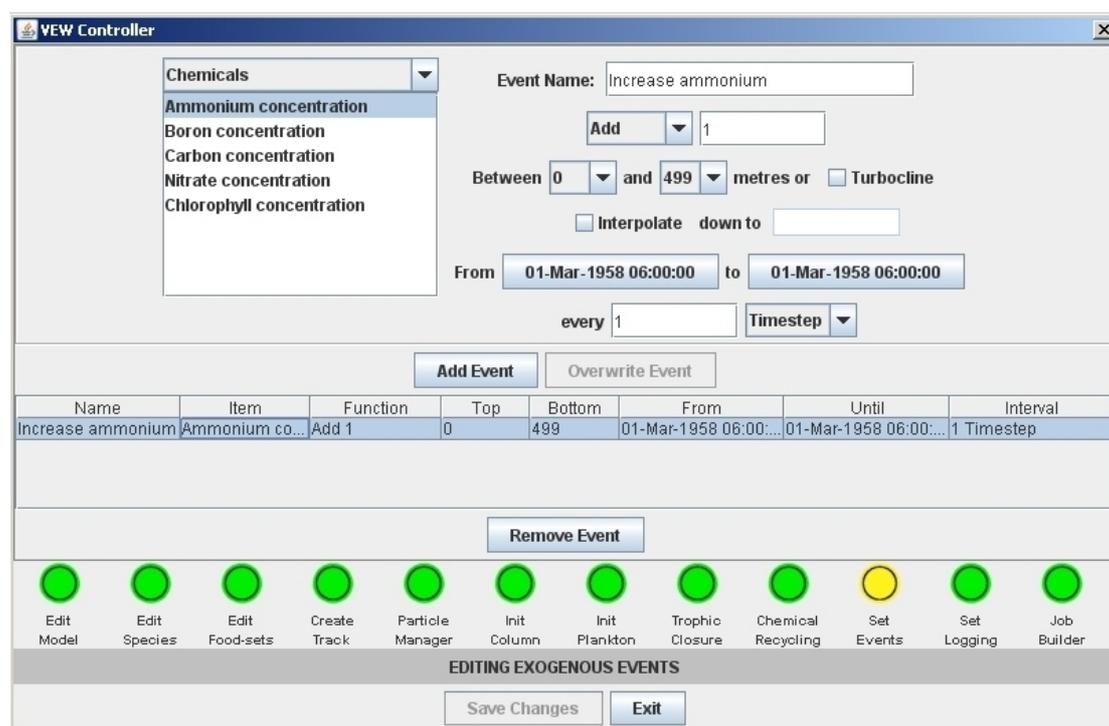
Having set all of these, the remineralised chemicals below the given depth will be summed, and added to the mixed layer at the given time.

19.1 Defining the chemical recycling, step-by-step

1. Firstly, tick the "Use Chemical Recycling" if you would like this switched on. Click on the date button, to select when chemical recycling should become active.
 
2. Secondly, select the method used for deciding when to perform the correction. You can either let the VEW automatically detect when a quiet season, which is found by detecting when the mixed layer depth has reached a maximum, and is starting to become shallower again. Alternatively, you can choose to perform recycling at yearly intervals from your start date. Thirdly, you can choose continuous adjustment, which recycles continually. This option causes the smallest numerical changes to the chemical concentration in the mixed layer, however it has the side effect of artificially placing traces of nutrient in the mixed layer all through the year – including times where the mixed layer might be fully depleted of some nutrients. For this reason, continuous adjustment should only be used with extreme care.
 
3. Next select the depth boundary, below which nutrients will be classed as "lost" when they are remineralised, and they will be added to the mixed layer at the next recycling event. You can either choose automatic detection, or you can specify a fixed depth.
 
4. If you would like chemical recycling to reset itself every year, tick the reset box. Ticking this box means that all chemical released below the maximum turbocline in one year will be cancelled after one attempt to recycle it; if you leave this box unticked, then the chemicals released below the maximum turbocline will be tracked continually onward, in case the maximum in a future year is deep enough to recycle that chemical. **Note:** if you require compatibility with versions *before* VEW 3.3, you should leave this box unticked.
 
5. For automatic detection, VEW will keep track of the deepest that the mixed layer has been in the past year, and also over the entire simulation. Select the option that you prefer.
 
6. For fixed-depth, choose the depth from the drop down menu. Remineralised chemicals below this depth will be recycled.
 
7. Finally, select the chemicals that you would like to be recycled, using the settings you have given.
 

20. Events

The events screen is used to force exogenous changes to the system at a given time. These changes could be biological, chemical, or physical. Biological events for convenience are handled previously in section 6 – Distributions of Plankton. A biological event might be the introduction of a new species into the mesocosm at a given time. Such an event could be used to model the introduction of a foreign type of plankton into an environment due to a tanker emptying ballast water that it picked up earlier at some other location. Chemical events might involve the addition of a new nutrient into the mesocosm at a given time, perhaps for modelling a spillage of chemical. Physical events may involve changes in the temperature or light levels in the water, perhaps modelling an eclipse, or mimicking global warming. These kind of “what-if” experiments can be modelled here.



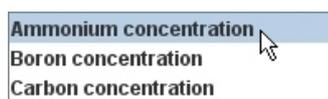
The events screen is shown above. The table in the middle shows the events currently added to the simulation. The type of the event and the variable name is shown in the top left. Three types are available here – chemistry, physics, and mesocosm properties, explained later. In the remaining top right portion of the screen is the event’s name, the numerical change that should be made, and if it is a field variable, the top and bottom depth boundaries for the change. Additionally, the starting and ending times where the event should be carried out, and the frequency are set.

20.1 Adding a new event (e.g., chemical event)

- Suppose we are trying to model an artificial fertilisation of the water, by adding a quantity of ammonium at a certain time. Choose “Chemicals” from the drop-down menu in the top left of the screen. The choice of variables updates as you select the event type.



- Select the variable, in this case Ammonium, from the variable list.



- Give a name for the new event.



- Define the numerical change that the event should cause. Select the function from the drop down menu, and type the number in the accompanying textbox, to define the change that should be made.



- As chemicals vary over depth, we can now choose the depth range for applying the event.

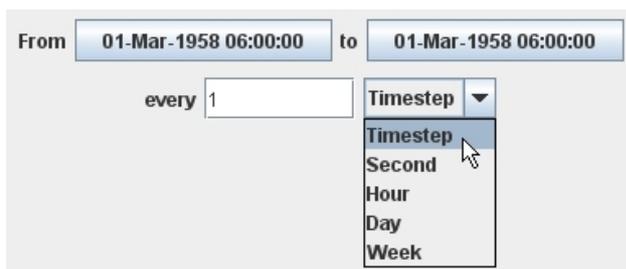


The range can either be set by a fixed top and bottom depth, or the bottom depth can be replaced by the instantaneous turbocline depth.

- If you would like the value you entered in part 4 to vary over depth, (whichever function you selected), to change over depth, then tick the interpolate box, and type the target value for linear interpolation into the box.



- Now select the starting time, ending time, and frequency, if the event is to be repeated. For a one-off event, the start and end dates should be equal.



- Finally, click on **Add Event** to complete the process.



20.2 Editing an event

- Select the event in the events table. Its details will appear in the top right part of the page.



- Make the necessary changes as described in section A. To overwrite the selected event with the edited event, click on **Overwrite Event**.



20.3 Creating a new event based on an existing event

1. Select the event in the events table. Its details will appear in the top right part of the page.

Name	Item	
Increase ammonium	Ammonium co...	Ac

2. Make any necessary changes and click on  .

20.4 Removing an event

1. Select the event in the events table. You can hold shift or control while clicking to select more than one event to remove.

Name	Item	
Increase ammonium	Ammonium co...	Ac

2. Finally, click on  .

20.5 Physical Events

1. Physical events can also be set, but their use is a little more complicated. The five variables here are part of a recursive relationship in the physics module:- their values from the previous time step are used to calculate their values in the following time step.

Physical Properties
Density
Full Irradiance
Salinity
Temperature
Visible Irradiance

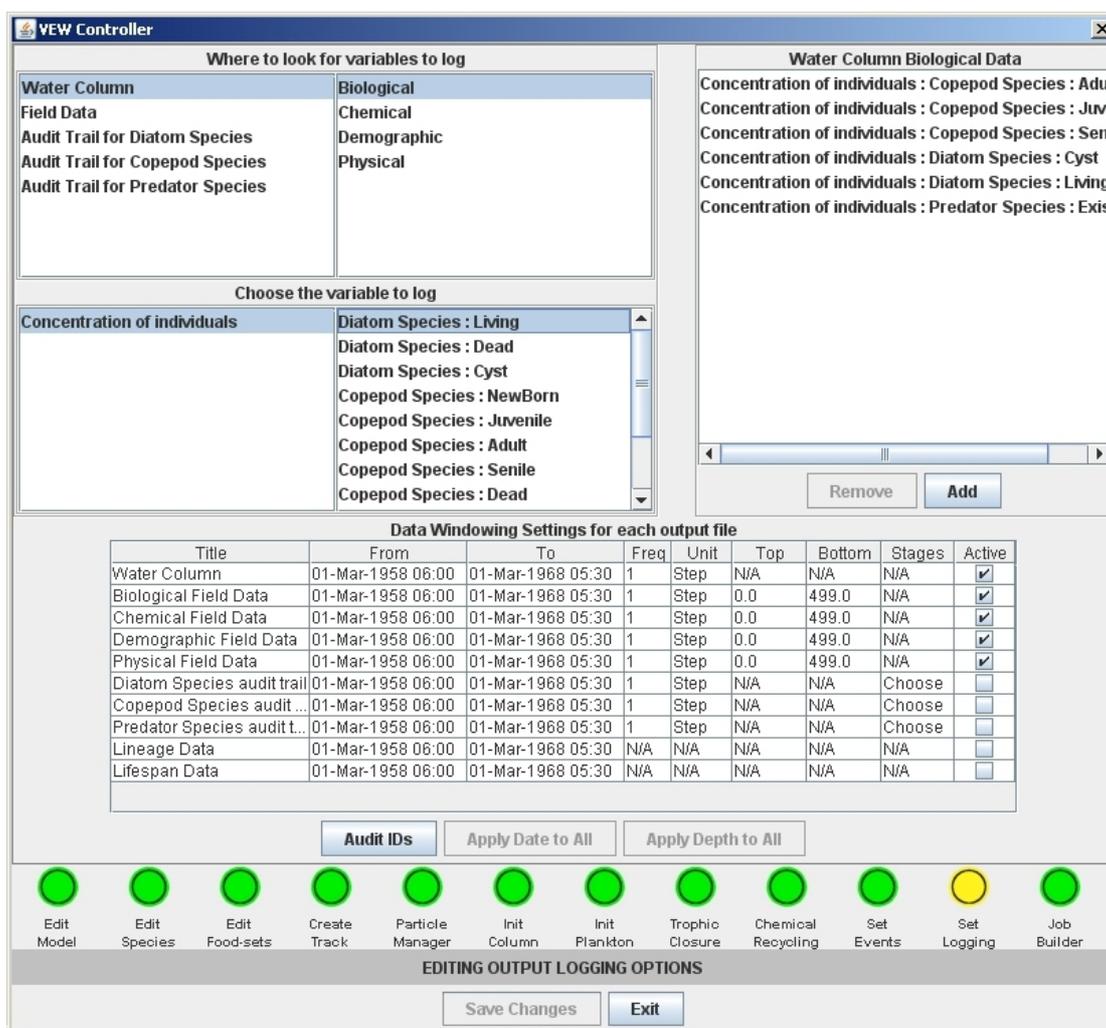
As a result, simply changing these values may have highly undesirable effects on the physics code, while also, because of feedback effects, not giving you the actual effect you would like for the biology.

This section is still under review. However, for the moment, we have enabled a number of useful experiments to be run by providing a special mechanism for **temperature**. Altering the temperature by using an event will cause the **biology** to notice a change, so model rules that use ambient temperature will be affected. However, this change will not be noticed by the physics code, so there will be no adverse reaction such as unexpected changes in turbocline as a consequence.

This will probably not be the final solution to this problem, but for the moment a number of useful experiments concerning temperature change can be carried out.

21. Logging

Logging is the fundamental part of the VEW which allows you to specify the data you wish to log. At the analysis stage, you will only be able to plot data that you have selected for logging on this page. This data is logged at cost – processing speed is affected in a minor way, but disk space requirement can be affected in a major way. Furthermore, requesting an extravagant number of variables may cause confusion at the analysis stage, making it harder to find the variables you are interested in. Therefore, care is required to ensure you have everything you need, without much redundancy.



The variables to be logged are selected in the top left quarter of the screen. They are categorised firstly by scope – whether the variable is a mesocosm property, a field-data property, or a plankton property. Secondly, variables are split into descriptive headings – for mesocosm data there are biological, chemical, demographic or physical headings, for other scopes, there are different headings, described later.

The next two lists pinpoint the variable you wish to log. The name (Concentration of individuals) appears in the left list, and if necessary, a qualifier (in this case, specifying which species and state) must be selected. The list in the top right of the screen shows the set of variables you have currently selected, for the scope and category that are currently highlighted.

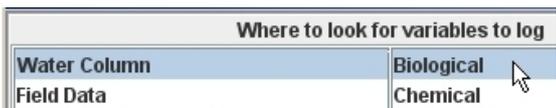
In the data windowing table, you can save disk space by choosing to window the data logged by time, choose a frequency of output data, and for field data, you can window by depth. For audit trails, you can select the states that you wish to be logged.

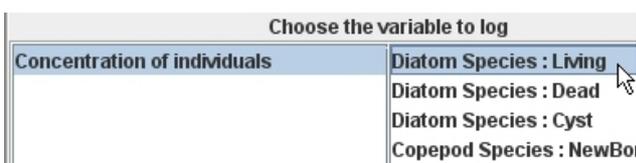
21.1 Variables that can be logged

The table below shows the scope and category of all variables that you can log from a simulation.

Scope	Category	Variables
Mesocosm (Column) One bulk value	Biological	Concentration of individuals (<i>species, state</i>)
	Chemical	Concentration in solution (<i>chemical</i>) Concentration in particulate form (<i>chemical, species, state</i>) Amount (<i>chemical</i>) ingested by species/state, and optionally which <i>species/state</i> it was ingested from.
	Demographic	Number of agents (<i>species, state</i>) Number ingested (<i>species, state</i>) All Model-specific state changes (<i>species, state</i>)
	Physical	Latitude, Longitude, Turbocline
Field Data Layer-by-layer values	Biological	Concentration of individuals (<i>species, state</i>)
	Chemical	Concentration in solution (<i>chemical</i>) Concentration in particulate form (<i>chemical, species, state</i>) Amount (<i>chemical</i>) ingested by species/state, and optionally which <i>species/state</i> it was ingested from.
	Demographic	Number of agents (<i>species, state</i>) Number ingested (<i>species, state</i>) All Model-specific state changes (<i>species, state</i>)
	Physical	Density, Full Irradiance Salinity, Temperature Visible Irradiance
Audit Trails (each species)	Agent/ Individual	Number of agents on the individual All user-defined biological properties of the individual
	Ambient Biological	Concentration of individuals (<i>species, state</i>)
	Ambient Chemical	Concentration in solution (<i>chemical</i>) Concentration in particulate form (<i>chemical, species, state</i>) Amount (<i>chemical</i>) ingested by species/state, and optionally which <i>species/state</i> it was ingested from.
	Ambient Demographic	Number of agents (<i>species, state</i>) Number ingested (<i>species, state</i>) All Model-specific state changes (<i>species, state</i>)
	Ambient Physical	Density, Full Irradiance Salinity, Temperature Visible Irradiance
	Internal Chemical	Chemical pool for each chemical Chemical gain (<i>ingestion+uptake</i>) for each chemical
	Local Variables	All user-defined local variables – used for intermediate calculations.
Species-based variables	Number of individuals ingested (<i>species, state</i>) All user-defined species-based variables.	

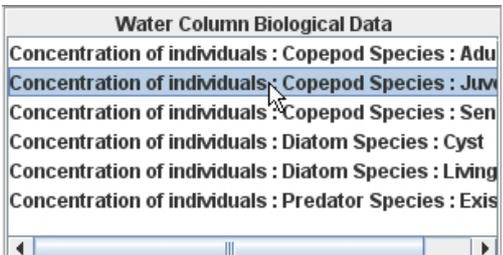
21.2 Adding variables to be logged

- Starting at the top left list, select the scope, and the category for the variable you want to log. Notice that the lower two lists of variable choices will change depending on the scope and category you have chosen, and the list on the top right will show the variables of that scope and category you have selected for logging so far.
 

- The lower two lists show you the variables in that scope and category that you can log. Select the variable you want to log from these lists. Note that sometimes (e.g. Species-based variables) you must firstly choose the variable type, and according to which variable you choose, different species and stages may become available in the bottom right hand list, whereas others (e.g. demographic variables) require you to choose the species and stage first, and different variables become available depending on your choice. Note you can select multiple entries by holding control or shift as you click.
 

- Click on  to select the variable for logging.

21.3 Preventing previously selected variables from being logged

- Having selected the appropriate scope and category, click on the variable to remove, in the top right list on the page.
 
- Click on  .

21.4 Windowing the logging between two times, at a given frequency

- The logged output is written to a number of separate files. Each one uses a row in the data windowing table.

Data Windowing Settings for each output file									
Title	From	To	Freq	Unit	Top	Bottom	Stages	Active	
Water Column	01-Mar-1958 06:00	01-Mar-1968 05:30	1	Step	N/A	N/A	N/A	<input checked="" type="checkbox"/>	
Biological Field Data	01-Mar-1958 06:00	01-Mar-1968 05:30	1	Step	0.0	499.0	N/A	<input checked="" type="checkbox"/>	

- To set the dates for each file, click on the *From* or *To* date, and use the pop-up calendar to set the date.

- To set the frequency of logging, click in the *Freq* column to change the frequency, and select the units from the *Unit* drop down menu.

Freq	Unit
1	Step
1	S... ▼
1	Step
1	Hour
1	Day
1	Week

- If you would like all the files to be windowed with the same date, select the row which has the date range you would like, and click on  .

21.5 Windowing the logging between two depths

1. For biological, chemical, demographic, and physical field data, you can also log data between two fixed depths. This is useful for saving disk space if you know you are not interested in values below a certain depth for example.
2. Click on the *Top* or *Bottom* column of the data windowing table, for the file that you wish to window by depth.
3. To specify that all field data should be windowed in the same way, you can copy the currently selected row to all the other field-data files by clicking on  .

Top	Bottom
N/A	N/A
0	49...
0.0	249.0
0	250.0
0.0	251.0
N/A	252.0
N/A	253.0
N/A	254.0
N/A	255.0
N/A	256.0

21.6 Setting the states for audit trails

1. You may be interested in some states of a species more than others; live diatoms more than dead diatoms for example. You can save disk space, and analysis time by restricting the audit trails to certain states. To do this, click on "Choose", in the *Stages* column, for the relevant audit trail file.

Species	Start	End	Step	Step	Stages	Active		
Copepod Species audit...	01-Mar-1958 06:00	01-Mar-1968 05:30	1	Step	N/A	N/A	Choose	<input type="checkbox"/>

2. The state selection window appears. Tick the boxes for the stages you would like audit trails for, and then click OK.



21.7 Lineage and Lifespan logging

1. The lineage and lifespan files allow for some additional post-processing operations to be performed.
2. The **lineage** file writes a parish register of all the particle management events, births and deaths. This is required if you want to use the *family tree* section of Analyser, which is used to determine the ancestors and descendents of a given agent. It also significantly speeds up the process of creating audit trail subsets in Analyser.
3. The **lifespan** file stores more detailed information about the individuals of an agent, including how many individuals change state (using a *pchange* operation), and how many are ingested. This is required if you want to do demographic studies using the *Lifespan* toolkit for Analyser, but since ingestion of individuals is so common, it is expensive in terms of disk space, with a very large number of entries.

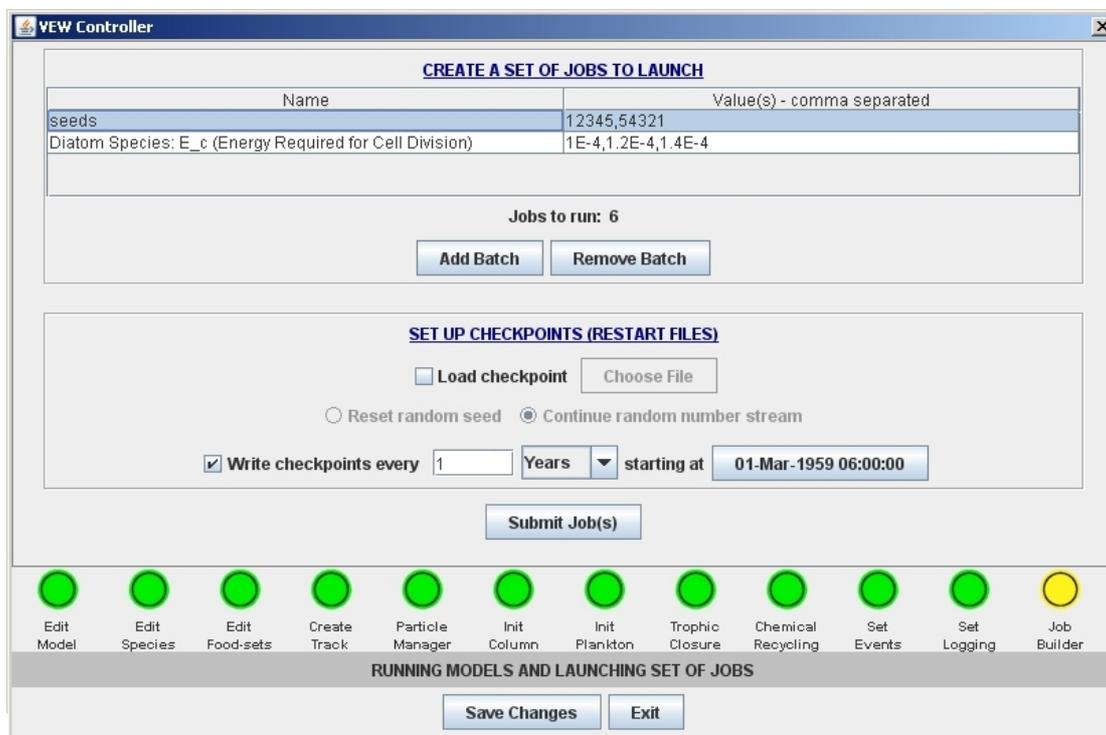
21.8 Activating the right logging output files

The final column of the data window table is the *Active* column. This switches each output file on and off, without affecting any of the other settings for that file. Make sure that you have all the logging files you like switched on or off appropriately.

Active
<input checked="" type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>

22. Building jobs

We now come to the final part of VEW Controller, in which the virtual ecosystem model is compiled and launched. Various options are available as shown below:-



The table at the top always shows in its first row the value for initialising the random number generator. A job can be run a number of times with different seeds, by supplying more than one value, separated with commas. The table may contain more rows, which represent *batches* of jobs; in the example above, for each seed, the job will be run three times, each with a different value set for the parameter E_c .

The second section allows checkpoints to be written regularly, or for the simulation to start from a specified checkpoint. A checkpoint is a snapshot of the entire simulation environment at a given instant, written to a file. These are useful in two ways; firstly, should a power-cut interrupt a long-running job, it can be restarted from a checkpoint file rather than from the beginning of the simulation time, thus potentially saving time. Secondly, if a simulation takes a while to stabilise, you may want to run it for a period to let it settle, and then launch future runs from a 'stable' checkpoint, thus saving execution time. For the first of these applications, you will want to continue the random number stream seamlessly, whereas for the second application, you may want to start the job from the checkpoint with a new random number seed.

Finally, the *submit* button gives you the choice of where results should be placed for each job, and starts the compilation process.

22.1 Setting one or more random seeds

- The top row of the table at the top of the screen shows the random seed. Double click in the value column to set the seed.

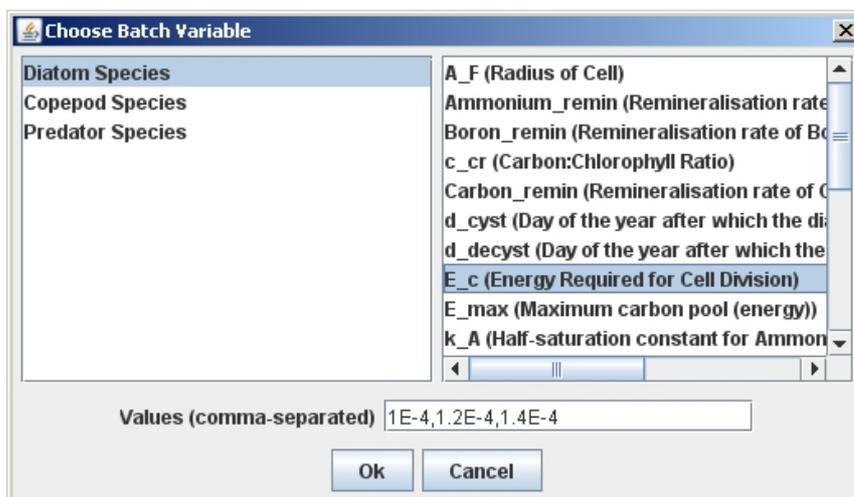
Name	Value(s) - comma separated
seeds	12345

- If you would like the job to be run more than once, with a different seed each time, type more than one value into the table, with a comma between each.

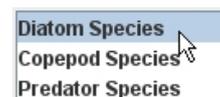
Name	Value(s) - comma separated
seeds	12345,54321

22.2 Adding a batch of jobs

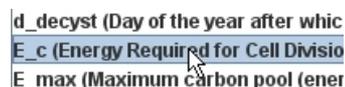
- A batch is a set of values for a selected parameter, where a separate job will be executed for each value; the job executed will be identical except for that parameter value. If you have specified multiple random seeds, then a job for each parameter value, for each seed will be created.
- Click on **Add Batch** to create a new batch. The parameter selection screen appears.



- Choose the species on the left hand side, which updates the list of parameters on the right.



- Then, choose the parameter that should be varied.



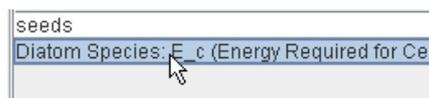
- Type the values, separated by commas, into the text field at the bottom. Note that you cannot add the same batch variable twice, as the values for one would overwrite those of the other.



- Click on **Ok** to add the batch. You'll see the new entry in the table.

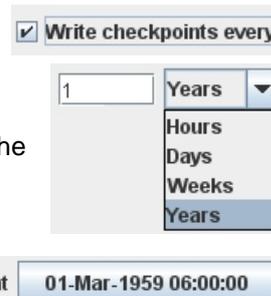
22.3 Removing a batch

1. Click on the batch in the batch table.
2. Click on 



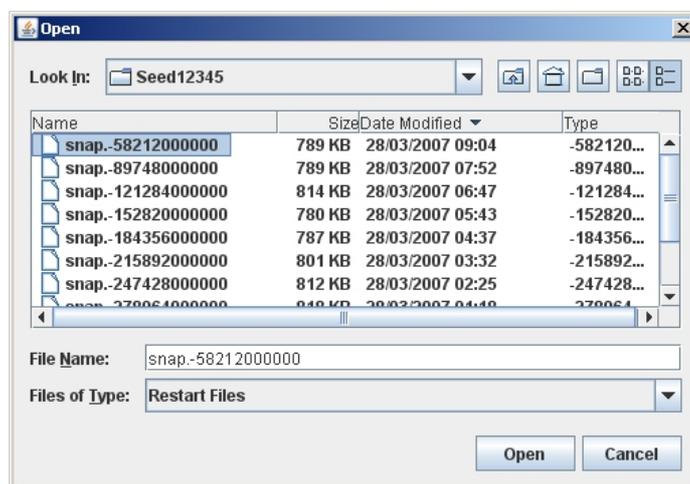
22.4 Specifying whether checkpoints should be written

1. If you would like checkpoints written, then ensure that the “Write checkpoints” tickbox is ticked.
2. Select the interval between writing checkpoints, using the textbox, and the drop-down unit selection box.
3. Select the time and date of the first checkpoint by clicking on the date selection button.



22.5 Specifying whether the simulation should start at a checkpoint

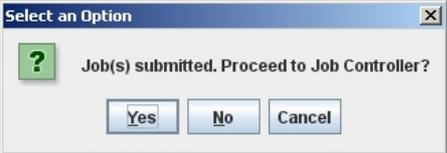
1. If you would like the simulation to start from a previously generated checkpoint, then ensure that “load checkpoint” is ticked.
2. You must then select the file for the checkpoint to load. Click on 



3. Checkpoint files start with “snap.”, and are followed by the timestamp from the simulation. [This part of the interface is tagged for improvement.] Select the file in the window.

- Note that you can only use a checkpoint from a “similar” model, where similar means the model has the same functional groups and species, with the same variables within, and the same chemicals as before. Parameter values, logging options and events may be different, but the basic structure of the model must be identical.
- When the simulation loads the checkpoint, if you would like the random number stream to continue seamlessly from the checkpoint, then ensure that “continue random number stream” is selected.  Continue random number stream
- Alternatively, if you would perhaps like to launch several jobs with different random number streams from this checkpoint, then selecting “reset random seed” will cause the random number stream to be reset, using the seed(s) you specified at the top of the page.  Reset random seed

22.6 Submitting the job and choosing the results location

- To submit the job(s) for execution, click on  .
- The job(s) will be submitted and compiled. Compilation involves two processes; firstly the input data for the jobs is created, since even if you are running a batch of jobs, they will require the same input data for the track. Secondly, the jobs you have submitted are compiled into Java code. This happens in the background, and meanwhile, you can either proceed to the job controller screen, or stay at the job builder page, and create more sets of jobs.  .

23. The Job Control Window

You can get to the job control window in two ways; one is by clicking on Current Jobs on the title page of the VEW. The other is from the menu offered immediately after submitting a job, where you can proceed to the job page immediately if you'd like to.



ID	Name	Status
101	BASE_Seed12345_Job0_101	Running
102	LERM-PS_Seed4637_Job0_102	Ready

Host: login.cx1.hpc.ic.ac.uk

Type: SSH Server

Username: wrh1

Password: ●●●●●●

Results Root Dir: /work/wrh1

Memory (Mb): 850

Use Java Server Mode

Max Time (h:m:s): 24:00:00

Queue command: /usr/pbs/bin/qsub

Run Script:

```
#!/bin/sh
#PBS -l walltime=%walltime%,mem=%mem%
module load java
java %server% -Xmx%mem% -jar %dir%/VEW.jar /setpath:%dir%/
```

Start Time:

Progress: Not Available

Predicted End Time:

Launched on:

Result Path:

The job control window shows the list of jobs currently being handled by the system. Two jobs are listed above – one is running already, and the other is ready to be run on the computer of your choice. Clicking on a job shows the information about that job in the bottom left. On the right of the screen are all the options concerned with the computer where a job is to be launched.

23.1 Job Status

Here are the different status messages you may see on this page.

Name	Description
Compiling	Compiling can take a while. It happens in two stages – firstly the climate data is created, this is the slow part, and the percentage updates regularly to keep you informed. Secondly, the Java code for your computer model is generated and compiled. It is not possible to predict how long this takes, but usually it is only a few seconds.
Ready	After compilation, if everything was successful, the job becomes Ready. It can then be launched either on a remote computer, or on your computer you are using to run the VEW. In the latter case, you can also launch it interactively using Live Sim.
Running	When you are running your simulation non-interactively, the Running message, and the percentage completed is displayed in the table below. The expected completion time is also shown, which will become more accurate as time goes by.
Finished	When your job has stopped running, either by completing normally, or by being aborted, the message changes to Finished.
Error...	Hopefully this will never happen. However, if there are errors in your model that your current VEW version has not been able to spot, then the simulation can fail to compile. If you get this message, clicking on it will bring up a window showing the details of the error. If you forward this to VEW Support, they will be able to help you, and perhaps upgrade the VEW to better deal with this problem in the future.

When a job is submitted, it will firstly show as *Compiling*. When compilation finishes, it will be *Ready*, and at this point, we can run a job.

23.2 Launching a job on a remote computer – CX1 example.

- We will describe the set up of launching jobs on CX1 as an example, which should show you how to set up jobs for any remote computer. Select CX1 from the menu on the top right of the screen to load the default details. At present, only two host-types are supported – *SSH Server* is for remote hosts, and *Local Computer* is for your own desktop.
- The host name is the computer that you log into in order to launch the job. You will have been provided with a username and password for that machine.
- The “results rot directory” is the place where you would like results to be put. A directory will be created within that directory, for each job you execute. You should use an absolute path here, because not all servers set up all of their drive mappings when logging in by ssh. (If you are experienced with ssh, you can test this by logging into the machine using a standard ssh client such as Putty, and type “ssh set” to see which environment variables will be set up.
- Below that is the memory usage – this is provided both for the job queuing system on CX1, and for Java. The memory usage can be difficult to predict; it depends very largely on the type of model you have built. Generally, we have found 850Mb to work well in most cases.
- Next, you will generally want to tick the “Use Java Server Mode” option, as the jobs you run will generally be lengthy, and your job will usually run in about 70% of the time taken in standard mode. See section 1.1.3 for information about how to enable server mode, although we have found that on Unix-based systems (Linux and Mac for instance), server mode is available by default.
- Queuing systems often require a maximum time to be specified. If there are limits on how much processing time you are allowed, consider using checkpoint files – see section 22.4. After this,
- VEW needs to know what command it should run to queue a job. This again should be an absolute path, since not all servers will have the default paths set up when logging in by ssh.
- Finally, the run script is used to create a file called *run.sh* on the remote computer. Running this script should run your job. The script is constructed by a combination of commands that your computer provider should tell you, followed by the Java command to run the job – the last line of the above example.
- If all of these details are correct, click on to queue the job. There will be a delay while the file is uploaded to the server.

23.3 Notes for setting up other remote computers.

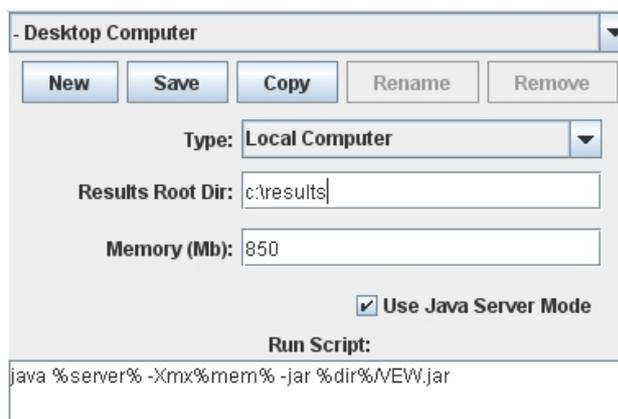
1. You can add another computer (like CX1) to the list of available computers by clicking on .
2. Alternatively, you can make a new copy of an existing computer setup (such as the CX1 example), by clicking on .
3. Make changes as required, and save them by clicking on .
4. The script will require some careful writing, and collaboration with your computer provider. The following bits of text can be used in your script, and the VEW launcher will automatically replace them when writing the script for a particular job.

%dir%	The directory stated in the “results root dir” box.
%mem%	The memory usage stated in the “memory” box.
%server%	Either “-server” if server mode is ticked, or nothing.
%walltime%	The maximum execution time stated in the “Max Time” box.

5. You can also rename a computer by clicking on , or remove one by clicking on .

23.4 Launching a job on your own computer.

1. Launching jobs on your own computer is simpler. Firstly choose “Desktop Computer” from the top menu. Fewer options are available, but you still need to specify the root directory where results will be stored, the memory Java will allocate to the job, and whether to use java server mode.



The screenshot shows a dialog box titled "- Desktop Computer". At the top, there are buttons for "New", "Save", "Copy", "Rename", and "Remove". Below these, the "Type" is set to "Local Computer". The "Results Root Dir" is "c:\results" and "Memory (Mb)" is "850". There is a checked checkbox for "Use Java Server Mode". The "Run Script" field contains the command: `java %server% -Xmx%mem% -jar %dir%\VEW.jar`.

2. The script for running the job on your local machine is much simpler. But you can still create your own scripts should you need something more specific. You can also create different profiles for different desktop computers – see section 23.2, but for local computers, many of the options will not be necessary.

23.5 Job Control

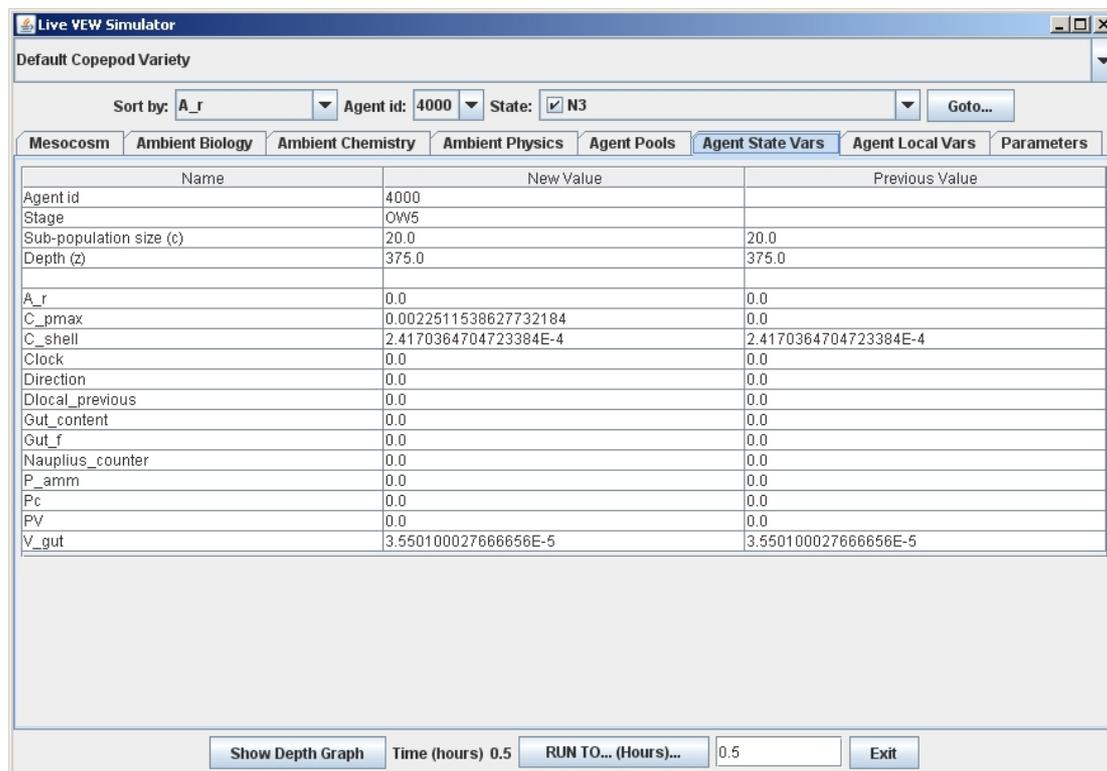
1. To abort a job that is currently running, click on . If the job is queued on a remote computer, then it will not be removed from the queue at this stage, but as soon as execution on the job begins, it will abort.
2. To analyse a job either when it is running, or has finished, click on . Note that this is only possible (at present) on jobs running locally. To analyse the results of jobs that have been run on remote computers, we suggest transferring the data files to your local machine using an FTP package such as FileZilla (<http://filezilla.sourceforge.net>), which is fast, convenient and free, and then locating the files as shown in Section 25.
3. To remove a job that has finished from the list of jobs, click on . This will not erase any results, but will just remove the entry from the list.

23.6 Launching an interactive job

LiveSim is an interactive visualisation, and model debugging tool. It was originally designed to test the VEW during its development, but it became obvious that it was far more useful than that. With LiveSim, you can investigate in detail every aspect of your model as it executes, timestep by timestep. Furthermore, in version 3.3, we hope that its presentation is become more clear and concise, allowing it be used very effectively as a teaching tool.

LiveSim will be described in detail in the next chapter. You can only run it on your local computer, and you do so by selecting a job that is ready to be launched, and click on .

24. LiveSim – The Interactive Simulation



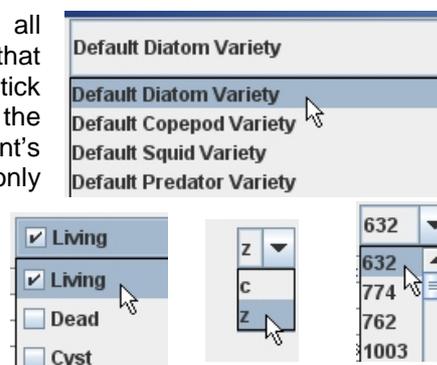
LiveSim is an interactive test tool that can help you visualise your model and discover what is happening at an individual-based level. This can be useful for demonstration purposes, or for checking or debugging your model. While the results it produces are the same as running the model in the standard way, it is considerably slower, so should not be used for running models in general.

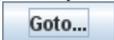
The top part of the screen allows you to select an agent to monitor. You select the agent by species first, and filter the options by the state or states you are interested in. The variables you can monitor are organised into tab screens, and those beginning with “ambient” or “agent” will then apply to the agent you have selected. The bottom part of the screen allows you to step through the simulation timestep by timestep, or to run until a certain time is reached.

24.1 Choosing an agent to watch

There are number of ways to choose an agent. In all cases, you should firstly select the species of agent that you are interested in, from the menu at the top. Next, tick the states of that species you are interested in from the box just below the species. You can now select the agent’s ID number from the drop down list, which will display only IDs of the species and stage you have selected.

If you have selected a variable from the drop-down list, then the ID numbers shown will have been sorted in increasing order according to that variable. Note that the variable called “c” is the sub-population size for that agent (historically called the cell-count).



Alternatively, if you know the ID of the agent you are looking for, you can select that agent directly by click on  and typing the number.



24.2 The Data Window

The main window in LiveSim shows information about the mesocosm as a whole, and also about the agent you have selected to monitor. They are organised into tabs, which have the following purposes.

Title	Description
Mesocosm	Shows the climate and astronomy data to the column (Ekman, oceanic heat loss, sunlight, wind speed, zenith height, and the resultant turbocline). Also shows the mesocosm totals of all different plankton (species and state), total chemicals in solution and particulate form, and a breakdown of in which plankton (species and state) the particulate chemical is found. That breakdown gives the total of the internal pools of the plankton, and also the amount gained by ingestion and uptake in the last timestep.
Ambient Biology	The concentration of all types of plankton in the ambient environment of the currently selected agent. It also gives the number of agents that represent each concentration.
Ambient Chemistry	The concentration of each chemical in the ambient environment of the currently selected agent. It is given in both solution and particulate form, and following that is the breakdown of in which plankton (species and state) the particulate chemical is found – both the pool, and the amount gained by ingestion or uptake in the last timestep.
Ambient Physics	The density, salinity, temperature and irradiance (in visible spectra, and the total irradiance), in the ambient environment of the currently selected agent.
Agent Pools	For the agent currently selected, this tab shows the new, and previous value of the pool for each chemical. It also shows the amount gained by ingestion and uptake in the last timestep. Finally, the numbers of plankton of different species and state ingested in the previous timestep is shown.
Agent State Vars (or biological properties)	This tab shows firstly the built in state variables for the agent currently being watched – its id, current state (i.e., growth stage), its depth, and the number of individuals the agent represents. After that come all the state variables in your model, for that species, including any are food-based (the special type used in ingestion equations), in which case there may be a value for each kind of food being ingested.

Title	Description
Agent Local Vars	This tab shows any local variables you have added in your model equations, for the agent being monitored. <i>Note that if you change the agent you are looking at, you will need to run another timestep before values are available – for memory saving reasons, local variables, which by their nature do not need to be preserved between timesteps, are accordingly not stored.</i>
Parameters	This tab contains for quick reference, all the parameters used, for all the species in your model.

24.3 Depth Profiles

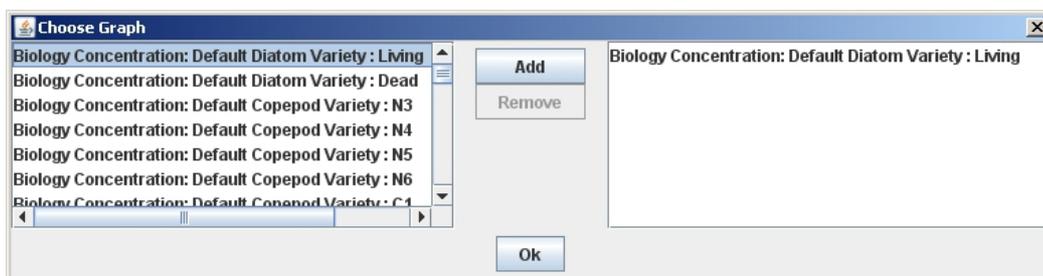
LiveSim can also show profiles of any simulation properties that vary over depth. To make this window, appear, click on



The first time you click on this button, no graph will have been selected, so the graph will be empty. To choose a graph to plot, click on



24.3.1 Choosing the depth profile



The dialog shows on the left the available variables, and on the right those you have selected. The depth profile plotted will be the sum of all the variables on the right – this is useful in case you would like to sum all the growth stages of a plankton species.

Select the variables as you like (remember that you can select more than one by holding control, or shift as you click), and then click on . Similarly, remove variables selected in the right window with .

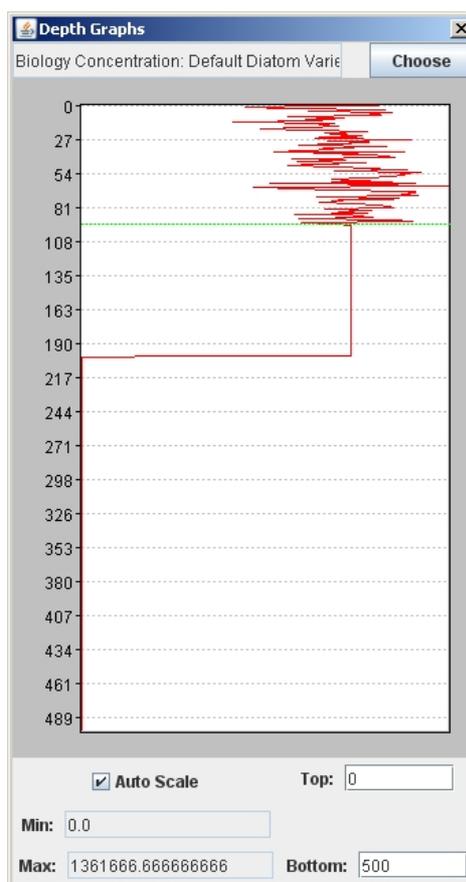
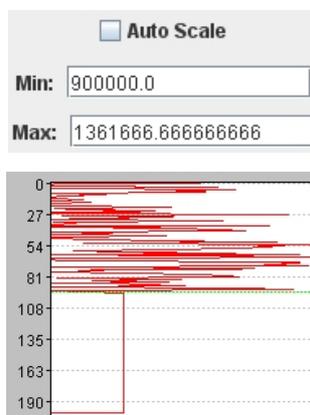
The variables available in the list are:-

- Biological concentrations of every species and state
- Chemical concentrations in solution
- Chemical concentrations in total particulate form
- Chemical concentrations in particulate form, indexed by species and state
- Physical properties - density, salinity, temperature, visible and full irradiance.

24.3.2 Scaling the depth profile

Having chosen the graph, it will then be shown in the depth window. The depth in metres is shown on the left axis, and for convenience, the green dotted line shows the current turbocline value. In this graph, we are looking at diatoms in the first few timesteps of a simulation, so those near the surface are being moved by the turbulence

The bottom left of the screen shows the minimum and maximum values – the minimum is the left edge of the graph, the maximum is the right. This is by default calculated automatically, but if you would like to change the scale, you can adjust this by unticking the auto scale tick box, and setting your own scale.



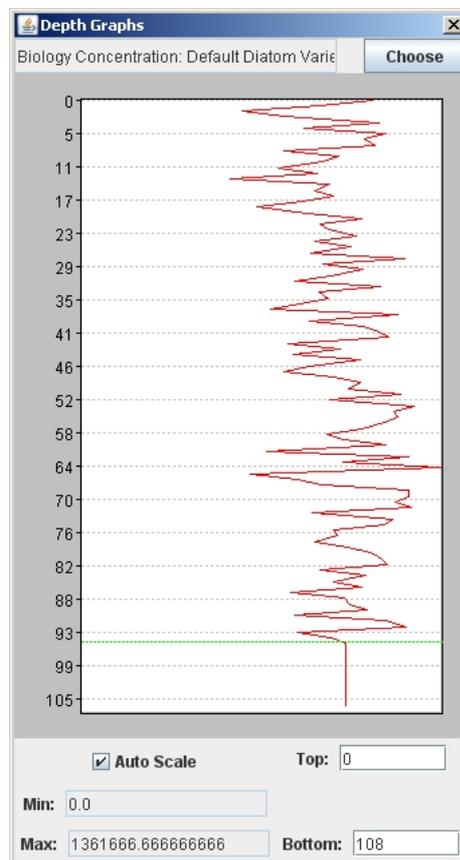
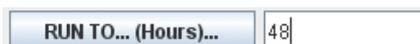
24.3.3 Choosing the depth range

Similarly, you can set the top and bottom depths of the graph to set the depth range of interest. By default, this will be set to the top and bottom extremities of the column.

24.4 Advancing the simulation

To perform a single timestep of the simulation, click on **RUN TO... (Hours)...**

To let LiveSim run a number of timesteps, type the hour which you would like LiveSim to run to, and click the Run button.



25. Launching Analyser

Having run a model, you will now likely spend a considerable amount of time Analysing the results. There are several ways to launch Analyser. One, as we have seen in 23.2, is to click on  from the job control window, while the job is still running.

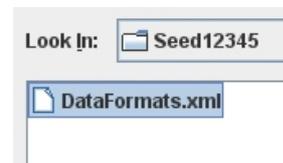
Alternatively, from the title page click on the Data Analysis button, to analyse data from jobs previously run.



Name	Path
First import	C:\westwork\VEWModelTree\BASE-ES\1\Results\Seed12345
<div style="display: flex; justify-content: space-around; margin-top: 10px;">    </div>	

The data analysis page shows a list of all the jobs you have recently run, and the path where the results for that job were stored. Click on a job, and then click  to analyse it.

If however you want to analyse data that is not in this list – perhaps it was run on a different machine – then you can click on . You will then be asked to locate the directory of the results, or more specifically, the file called DataFormats.xml, which Analyser is able to load.



As the list of available results will continue to fill, you can remove entries from the list when you have finished analysis by pressing . You will be asked whether you would like to save disk space by deleting the results as well. If you answer Yes to this, it will ask you once more to confirm that the results will be deleted forever. Otherwise, you can remove them from the list, but you will still be able to find them later by clicking .

Appendix 1 – Building the jobs in Chapter 9

Chapter 9 contains 10 example fragments of model, presented in order to demonstrate the facilities, and the use of the functional group builder and equation editor. The later chapters showed how to set up all the other parts of a model, which are required for a simulation to work. In this appendix, we will revisit the examples in chapter 9, and list in the most concise way how to make each one into a model, and how best to observe the behaviour of that model in action.

Unless specific details are mentioned, click on any red buttons, from left to right, in the VEW interface that come up, and allow the VEW to configure itself automatically. The notation of equations below is designed to mimic the way you should construct them.

Example 1 and 2 – Corrected Motion

- Create functional group “Plankton”, a function “Motion”, and edit the function.
- When writing the rule, you will need to create a Group Parameter “sink_rate”, with value 0.2 m h^{-1} .
- Write a rule: assign, (z = conditional (lessequal (z <= Turbocline) -> rnd (Turbocline) otherwise add(z + mul (sink_rate * TimeStep))))
- Set the Motion rule to execute in Default stage
- Set the track at Azores (Lat 41, Long 27), fixed location, and allow 1 month of data. (This is the same for all of these exercises). See chapter 13.
- Click Particle Manager – no options needed. See chapter 15
- In “Init Column”, Initialise Density, Salinity, Temperature, and Turbocline to the provided values by clicking each one and “Use Data”. See Chapter 16
- In “Init Plankton”, add a single distribution of plankton in the default stage, 1 agent per metre, 1 individual per agent, depth range 10-11m, at the default (start) time of the simulation. See chapter 17.
- Submit the job with no other options, and launch it in LiveSim. Run a single timestep to let the plankton initialise.
- Look at a depth graph of the biological concentration of Default Plankton.
- Also select agent 0 in the list of ids, and examine the Agent State Variables to see z, and the mesocosm details to see the turbocline value.

Example 3 – Nutrient Uptake

- Start a new model. Create a chemical “Ammonium” and functional group “Plankton”.
- Create a function for plankton called “Uptake”, and edit it.
- As you edit the rules, you’ll need to create a local variable “rate”, in mMol Ammonium h^{-1} , a parameter “Ammonium_max” with value 1 mMol Ammonium, another parameter “Ammonium_maxrate”, with value $0.05 \text{ mMol Ammonium h}^{-1}$.
- Create a rule, “Set rate”. assign (rate = mul (sub (1 - div (Ammonium_pool / Ammonium_max)) * Ammonium_maxrate))
- Create a rule, “Perform uptake”. Uptake (mul (rate * Timestep), Ammonium Concentration))
- Create a rule “Update pool”, assign(Ammonium_pool = add (Ammonium_pool + Ammonium_Ingsted))
- Set these three rules to execute in the Default stage.
- Track and initial conditons are as before, but since Levitus does not have Ammonium data, click on Ammonium in the top window, Nitrate in the bottom window, and use data.
- Initialise the plankton 1 default diatom agent per metre, 1 individual per agent, at 300-301m. You will be given the option to set the starting value of the Ammonium pool, but leave it as 0.
- Submit the job and run it in LiveSim.
- Look at Ammonium concentrations on the depth graph, and the Ammonium Pool in the agent pools tab for agent id 0.

Example 4 – Nutrient Release

- Create a new model, create the chemical Ammonium, and the functional group Plankton, with one function, "Release".
- When editing the rules you will need to create a local variable "Ammonium_release" in mMol Ammonium, and a parameter "Ammonium_releaserate" with value 0.05 mMol Ammonium h⁻¹.
- Create a rule "Calculate amount": assign (Ammonium_release = mul (Ammonium_Pool * Ammonium_releaserate * TimeStep))
- Create a rule "Perform release" : release (Ammonium_release , Ammonium Concentration).
- Create a rule "Update pools" : assign (Ammonium_Pool = sub (add (Ammonium_Pool + Ammonium_Ingest) – Ammonium_Release)).
- Ensure the function is run in the Default stage.
- Note that in this model, the Ammonium_Ingest isn't really necessary since uptake is not performed. But it's good practise to remember this.
- Track and initial conditions as before.
- Initialise plankton at 1 agent per metre, 1 individual per agent, between 300 and 301m, but this time ensure the starting Ammonium_Pool is 5.0.
- Submit job, run in LiveSim.
- Look at ammonium concentration around 300m, and also at agent 0's Ammonium pool in the agent pools tab.

Example 5 – Cell Division

- Create chemical Ammonium, functional group Plankton.
- Create a function "Cell Division".
- When you need to, create a local variable C_div, dimensionless, and a parameter Ammonium_cdiv with a value of 0.95 mMol Ammonium.
- First rule: "Decide Cell Division", assign (C_div = conditional (greater (Ammonium_Pool > Ammonium_cdiv) -> 2, otherwise 1))
- Second rule: "Perform cell Division" : ifthen (equals (C_div = 2) -> divide (2))
- Create another function, "Uptake", which contains two rules that are identical to "Calculate Amount" and "Perform Uptake" in example 3.
- Create a third function, "Update Pools", which contains one rule: assign (Ammonium_Pool = div (add (Ammonium_Pool + Ammonium_Ingested) / C_div))
- Ensure that all three functions are called in the Default stage.
- Track and initial conditions as before.
- For the initial plankton, have 1 agent per metre, 1 individual per agent from 300-301m, with the Ammonium Pool initialised at zero.
- Submit the job and run it. Look at the ammonium pool rising towards 0.95 on the agent pools tab for agent 0. The first time it reaches 0.95, you can either look at the agent local variables for C_div (which will be 2 in the following timestep, 1 in all other timesteps), or the agent state variables for the sub-population size c (which will double at the crucial moment). Also, when you return to the ammonium pool, it will have halved.
- On depth graphs you'll see the ammonium in solution decrease as before due to the uptake.

Example 6 – Creating new agents

- Create the ammonium chemical and the plankton functional group as before. You'll need to add an extra stage called "Pellet" in the Edit Stages window.
- Create a "Decide Cell Division" function identical to example 5.
- Create an "Uptake" function identical to example 5.
- Make a "Create Pellet" function, which contains two rules. It will need a local variable Ammonium_Pellet, in mMol ammonium, and a parameter Ammonium_excretefraction with value 0.1, dimensionless.
- It contains two rules: the first is assign (Ammonium_Pellet = mul (Ammonium_Ingested * Ammonium_excretefraction))
- The second rule: ifthen (greater (Ammonium_Pellet > 0) create 1 agent in stage Pellet). Add a single "set" rule to the create: Ammonium_Pool = Ammonium_Pellet.
- Create an Update Pools function, which is different to previous ones as it now subtracts the ammonium lost to the pellet: assign (Ammonium_Pool = div (add (Ammonium_Pool + sub (Ammonium_Ingested – Ammonium_Pellet)) / C_div))
- Create a function called "Release", which contains three rules. It requires a local variable, Ammonium_release, in mMol Ammonium, and a parameter Ammonium_releaserate, which is 0.05 mMol Ammonium h⁻¹.
- First rule: "Calculate release" : assign (Ammonium_release = mul (Ammonium_Pool * Ammonium_releaserate * TimeStep))
- Second rule: "Perform release" : release (Ammonium_release to Ammonium Concentration)
- Third rule: "Update pool" : assign (Ammonium_Pool = sub (add (Ammonium_Pool + Ammonium_Ingested) – Ammonium_release))
- Set the first four functions to run only in the Default stage, and set just this final function to run in the Pellet stage alone.
- In the Particle Manager screen, add a rule that merges Plankton in the pellet stage down to 1 per metre, throughout the whole of the column, continually.
- Track and initial conditions are the same as earlier examples, and a distribution of plankton in the default stage should be initialised with 1 agent per metre, 1 individual per agent between 300-301m, with an initial Ammonium_pool of 0.
- Submit the job and run it in LiveSim.
- Look at the depth graph of ammonium in solution, and you'll see it rise due to the pellet remineralisation.

Example 7 – Stage Changes

- Create a functional group "Plankton". No need for chemicals in this example.
- Rename the default stage to "DayTime" and create a second stage, "NightTime".
- Create a function "ChangeToDay". You'll need to create one parameter, I_ref, with a value of 4 W^{m⁻²}.
- Create a single rule : "Change to day" : ifthen (Visible Irradiance > I_ref) change(DayTime).
- Create a second function "ChangeToNight", which has another single rule: "Change to night" : ifthen (Visible Irradiance < I_ref) change(NightTime).
- Make sure that Change to Day is enabled in the NightTime stage, and Change to Night is enabled in the DayTime stage.
- No particle management rules needed, initial conditions as before, and a single plankton distribution, 1 agent per metre, 1 individual per agent, between 10-11m, initialised in NightTime stage.
- Submit the job and run it in LiveSim.
- Watch the agent state variables, and if you like open a depth graph of Physics: Visible Irradiance, so you can see the change in light, and see the stage of agent 0 oscillate depending on the light level.

Example 8 – Probabilistic Stage Changes

- Create a functional group “Plankton”, and an extra stage “Dead”.
- Create the motion function identically to example 2, and enable it in both stages.
- Create a second function called “Mortality”, and enable it in just the default stage.
- You’ll need a local dimensionless variable, P_death, and a parameter I_ref, set to 4Wm^{-2} .
- Create a rule, “Calculate probability”: assign (P_death = mul (0.0005 * div (Visible Irradiance / I_ref)))
- Create a second rule, “Probabilistic State Change” : pchange (P_death into state Dead).
- In Particle Management, merge the dead plankton down to 1 per metre for all depths, for all time.
- Initialise plankton in the Default stage, 1 agent per metre, 1 individual per agent, 10-11m.
- Submit the job, and run in LiveSim.
- Look at a depth graph of the Dead plankton. You’ll see a new flurry of dead plankton created when it is light, and there will be larger dead agents nearer the surface where the irradiance will be higher.

Example 9 – Integration over depth

- Start with example 2, so the agent has some motion, then add a new function “Average Temperature”.
- Make one rule, which will require a local variable T_avg, measured in degrees celcius.
- The rule is: assign (T_avg = div (integrate (Temperature) / abs (sub (z – varhist (z,1))))) – where the varhist represents z a timestep ago.
- Initialise the plankton with 1 agent per metre, 1 individual per agent, between 10-11m.
- Submit the job, run it in LiveSim.
- Look at the agent local variables and you’ll see the average temperature change, depending on how deep the agent is, and whether it is sinking, or being moved by turbulence.

Example 10 – Ingestion

- Make two functional groups called “Plankton” and “PlanktonFood”, and a chemical “Ammonium”.
- In the “Plankton” functional group, make a function “Motion”, and copy the motion function from Example 2.
- Make another function called “Ingestion”. It will contain four rules.
- The first rule sets the “satiation”, and requires a local variable S, and a parameter Ammonium_max, which should be set to 1 mMol Ammonium. The rule is assign ($S = \text{div}(\text{Ammonium_pool} / \text{Ammonium_max})$)
- The second rule sets the “feeding rate”. First create a food-set called F. Then create a food-based local variable F_rate, and two food-based parameter, F_min, and F_max, which are the threshold for feeding, and the maximum feeding rate respectively. The values are configured later. The rule is assign ($F_rate = \text{div}(\text{mul}(F_max * \text{sub}(1 - S)) / 3600)$). The 3600 is to convert to individuals **per second** for ingestion.
- The third rule is the “ingestion” rule: ingest (F, F_min, F_rate)
- Finally, write an “update pools” rule: assign ($\text{Ammonium_pool} = \text{add}(\text{Ammonium_Pool} + \text{Ammonium_Ingested})$).
- Ensure that both functions are called in the default stage.
- In Food-sets, select the one food set (Plankton Species : F), and tick the PlanktonFood:Default food on the menu. You can then set F_min for PlanktonFood:Default to 0, and F_max to 0.1.
- Use the same track and initial conditions, and no particle management rules.
- For the plankton initialisation, you need two distributions. The first is of Default Plankton. 1 Agent per metre, 1 individual per agent from 10-11m, and initial Ammonium_Pool 0 mMol ammonium.
- The second distribution is of the Plankton food: 1 Agent per metre, 1 individual per agent, but 0-499m, and with initial Ammonium_Pool 10 mMol Ammonium.
- Submit the job and run it in LiveSim.
- The simulation contains one “predator” agent. eating from a whole range of food agents. Use a depth graph and look at the PlanktonFood concentration. The change is quite subtle – untick auto-scale, and choose 0.95 to 1.00 as the scale to see the changes more vividly.
- You can browse through the (many) PlanktonFood agents – try clicking on the agent state variables tab, select PlanktonFood on the top menu, and hold the down arrow key – watch the sub-population size change as LiveSim cycles through the agents. Some of the individuals of some of the agents will have been eaten.
- Also look at the single Plankton agent, particularly the ammonium pool (Agent pools), which will rise asymptotically towards Ammonium_max, and in the Agent state variables table, satiation will increase asymptotically towards 1.