Contents:

User's Guide

Introduction

What is HOM?

HOM is a numerical simulation of human homeostasis and physiology. It encompasses all major organ systems: cardiovascular, respiratory, renal, gastrointestinal, liver, muscle, skin, and brain. It works in real time or compressed time. It aims to be useful in the teaching of undergraduate physiology.

How does it work?

There are approximately 500 numerical values, each of which represents a physiological parameter, and governs or is governed by other values in the system. Some belong to the patient's environment, and the others belong to various organs in the patient's body.
The rules relating the variables to each other are either physical (hard-wired) rules, or 'controllers' – the homeostatic mechanisms of the body – which can be viewed and altered.

How do you interact with it?

The simulation can be started, paused and reset using the control bar at the top of the screen. The time compression of the simulation can be controlled with the slider at the top-right of the screen.
Several of the numerical variables can be altered by the user. When selected, a slider control appears which can be dragged to alter the value of the variable.
In addition to numerical variables, there are true/false values which can be viewed and altered as check-boxes, and action events which can be invoked by clicking buttons.
While the simulation is paused, you can draw on the graphs with the left button, and scroll backwards in time by dragging with the middle mouse button (or `CTRL`+left mouse button).

How do you view the effects?

The values of selected variables are displayed on screen in the form of graphs, numbers, or bars. The conscious level is always displayed above, and a text-based log at the bottom of the screen shows changes in the patient's state.

Can I alter the way it works?

The components that are shown on the screen can be fully customised. The display can be configured for very basic situations or for advanced students.
In addition, advanced users can alter the homeostatic controllers, use the scripting facility to add new physiological behaviour, and use the text files to create new scenarios and problems.

How can so many variables be controlled at once?

HOM can be started in one of many setups. Each setup is centred on a particular physiological scenario. A single setup comprises:

1. Several variables whose values are visible in graphs or bar charts

2. Other variables whose values can be altered by the student as sliders, check-boxes or buttons
3. Initial settings of certain variables, e.g. to simulate a problem or disease
4. A help-file (HTML-file) which displays information about the scenario
5. Settings to enable or disable specific menu options
6. Default values for time compression, graph scrolling speed etc.

If the tree is visible in your current setup, you will be able to add variables to the display by either

1. Selecting the variable from the tree on the left, right clicking to get a menu, go to `Display`, then select an option for displaying the item.
2. Selecting `Quick add` from the `Graphs` menu, or pressing `CTRL+A`, then typing the first few letters of the variable's name in the box. Press `ENTER` to display the variable.

Variables can be removed from the display using the red 'X' symbol above the display.

To find the value of a variable that is not visible, you can either

1. Double click the variable's name in the tree, or right-click and select `Info`. Then select the `Values` tab in the info box.
2. If the option `Show editor for selected variable` is selected in the `Options` menu, simply clicking on the variable's name will display the value below the tree in the bottom-left corner of the screen.
3. At any time, you may type the name of a variable (full name, abbreviated name, or the path name of the variable in the tree) into the console at the bottom of the screen. Pressing `ENTER` will give the full decimal value of the variable.

Specific quirks

1. To cause bleeding, first set the value of `BleedingRate` to the desired value, then turn on the check-box `Hmrg` to start or stop the bleeding.
2. To give the patient a fright, first set the value of `Fright` to the desired amount, and then press the button `Frighten` to give the patient a fright.
3. An `Uprightness` value of 50% represents lying flat. 0% is upside down, 100% is erect.

Using HOM: A Guided Tour

1. Getting Started

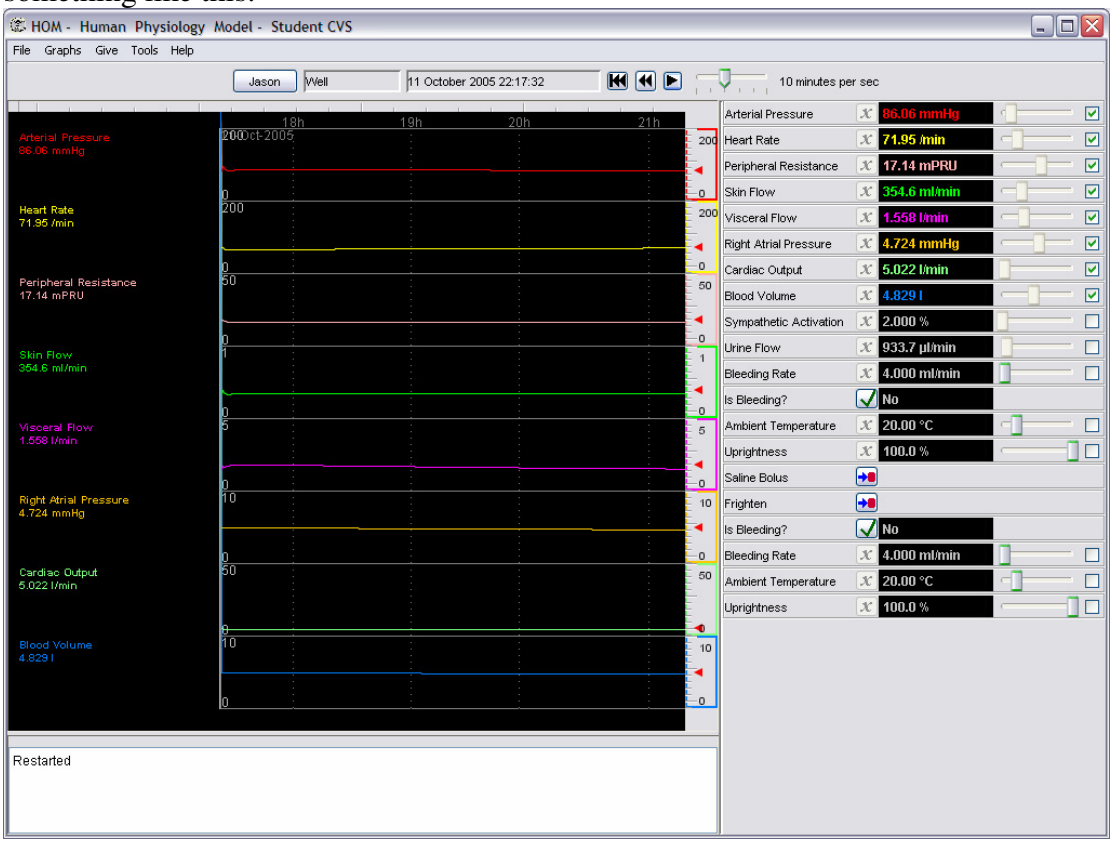
You need two things to run HOM:

1. Java 1.4.0 or higher (J2SE Runtime). This is incorporated into many operating systems, including Windows XP and MacOS X. If your version does not run HOM, please download the latest version from http://www.java.com/en/download/download_the_latest.jsp
2. The HOM program – "HOM.jar". If you double-click this file, the simulation should run. If different program pops up, e.g. WinZip, you may need to re-install Java.

2. Choose a scenario

A *scenario* is a set of initial values for the simulation, and a configuration of graphs and controls, that allow you to investigate one aspect of physiology.
Once HOM starts, you will be presented with a menu of scenarios. Click on one of the items to start HOM with a given scenario. If you hover over an item in HOM, a description will pop up.

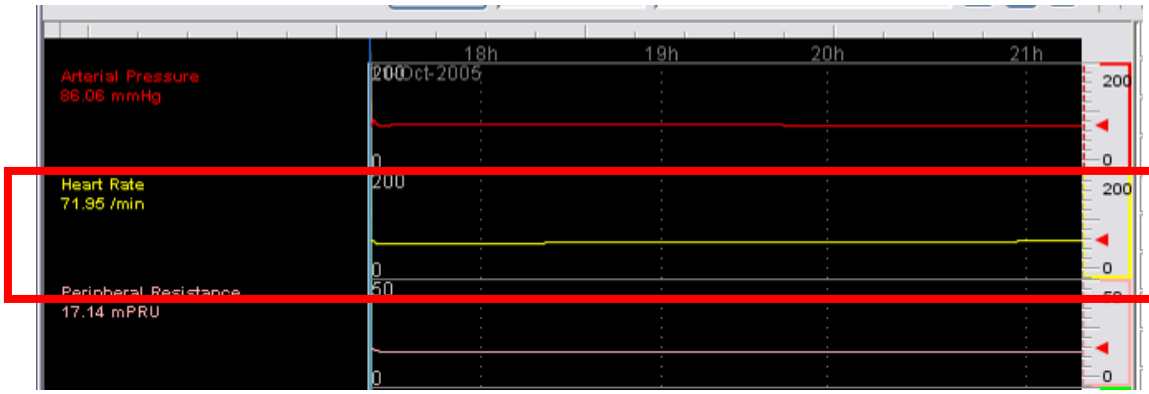
The simulation will then load, and after about 15 seconds you will then see the main simulation window. Depending on the scenario you have selected, it may look something like this:



The main simulation window

The five main areas of this window will now be described: the Graphs Panel, Control Bar, Variable Controls, Scrolling Log and the Menu Bar.

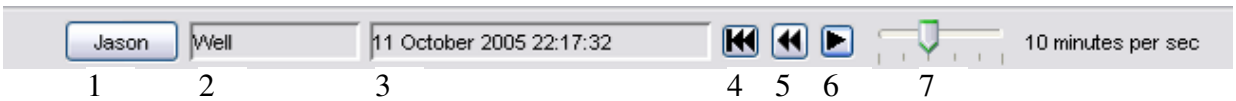
Graphs panel



Time runs horizontally, with the past on the left, and present on the right. Along the top is the time scale.
Each variable occupies one horizontal band across the screen, and has a unique colour. On the left is the variable's name and current value. To the right is the scale for the variable, which shows the values corresponding to the top and bottom of this horizontal band. Also in the scale is a red arrowhead, which represents the normal value of the variable.
In the shown example, it is now 21:15. The heart rate is in yellow, and it is currently about 72 beats per minute, with the graph scaled between 0 and 200 /min. The red triangle shows that this is also the normal value.

Rescaling:
The graph can be rescaled by right-clicking in the scale, and selecting one of the scale options. This will only rescale graphs that are drawn in the future, and not graphs that are already on the screen.
Scrolling: You can look back at older traces by either:
Using the mouse wheel while over the black area
Dragging left and right with the middle mouse button
If you do not have a middle button, you can drag with the [CTRL] key held down, with the left mouse button.
Marking:
You can draw marks on the graph by holding down the left mouse button. If you move the mouse over the horizontal scale at the top of the area, a vertical line is displayed. Click here to make a vertical mark.

Control bar

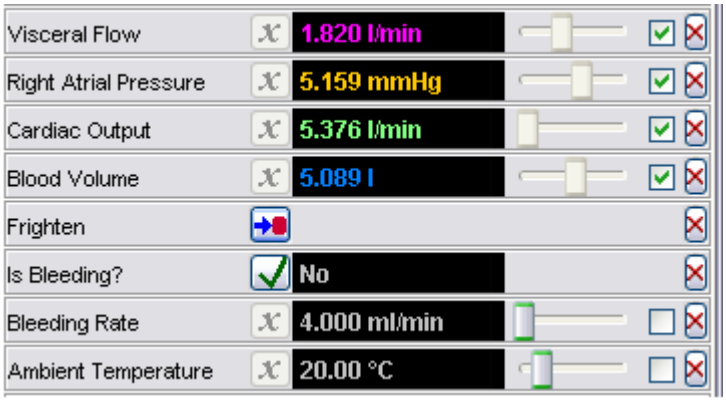


This bar along the top of the screen allows you to control the running of the simulation. You will see:

1. Patient's name

2. Patient's state: one of 'Well', 'Unwell', 'Unconscious' or 'Dead'.
3. Current time (in the simulation)
4. Restart – clears all current settings, physiology modifications, and returns all body and environment variables to normal values.
5. Reset variables – returns all body variables to normal values.
6. Stop / go – starts and stops the simulation
7. Simulation speed – this is the 'time compression', or the rate at which simulation time goes by, relative to real time. The simulation can work at any speed, limited by the power of your computer. On a 1 GHz Pentium, the maximum speed is 1 day per second. That is, for every second you run the HOM, the 24 hours elapse for the patient.

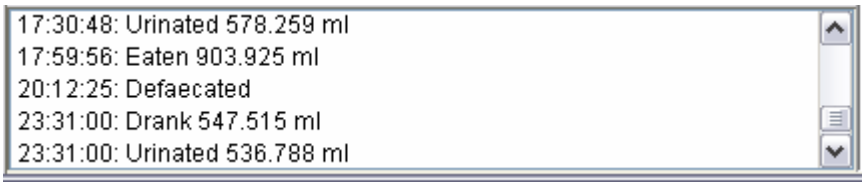
Variable Controls



These controls vary according to the variable displayed. Generally, you will have

1. Variable name – Clicking on this brings up a box with information on this variable.
2. Action button – for numerical variables, this does nothing. For Yes/No variables (e.g. Is Bleeding?), the button toggles the variable's value. For events (e.g. Frighten), the button performs the event.
3. Value display – this shows the numeric value of the variable. Hover here to see the normal value, and click here to show the variable information box. If the variable is displayed as a graph in the black graphs panel, the colour is reflected here.
4. Slider – this is present for numerical variables only. It reflects the value of the variable, and in certain cases, can be changed (e.g. Bleeding Rate). To change the value of a variable, drag the handle left or right. Hover here to see the range of the slider.
5. Check-box to draw graph – numerical variables can be displayed as a graph by checking this box. The variable is allocated a colour, and appears at the bottom of the graphs panel.
6. Remove – (in advanced setup only) allows you to remove the variable from this display.

Scrolling Log (Console)



The scrolling log continuously reports events in the patient's life. You can also type expressions into the box, to use it as a calculator.

4. Navigating the cardiovascular scenario

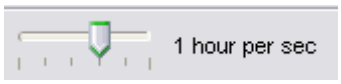
The cardiovascular scenario should be running. You will see horizontal lines as the graph scrolls – this indicates that Jason is fairly stable
Select from the menu, Graphs – Zoom in graph scales. This will allow you to see the smaller fluctuations in the variables. You will see that Jason's heart rate actually varies between 60 and 75 beats per minute. If you choose Graphs – Zoom out graph scales, you can see larger changes.

At the bottom of the main graph panel, note the letters 'E' and 'D' – which signify the times when Jason eats and drinks.

Press the pause button on the control bar to stop the calculations.

Use the middle mouse button to look back at the first graphs, noting that the point you changed the scale is marked on.

5. Speeding up and slowing down



On the control bar, use the simulation speed slider to speed up the simulation to '1 hour per second', by dragging it one notch to the right.

Press the play button to unpause the simulation. You will notice that the values appear to change much faster. Each vertical dotted line represents one hour.

Menu Bar

Menus vary according to the setup. For basic investigations there is little need to use the menus; however there are some useful utilities for advanced users.

File Menu:

Save	save a data file representing the state of all numerical variables. The interface, i.e. which graphs and controls are displayed, is not saved.
Open	open a data file representing the state of all variables.
Print Screen	prints out the whole HOM window
Print Log	prints out all the text in the HOM log
Print Graphs	prints out the graphs currently on screen
Exit	leave HOM, discarding any changes
Restart	Does the same as the restart button, clearing all settings and controllers, resetting all environment and body variable values

Graphs Menu:

Rate	changes the scrolling rate of the graphs. A high scrolling rate visualises fast changes better, and a slow rate is better to see long-term trends
New Orbit	allows you to plot the value of one variable against the value of another. At the origin of the plot are the normal values, allowing you to easily see disturbances in the values
Setup	this re-arranges the graphs and variable controls to view a different aspect of physiology.
Close all	(advanced setup only) this clears the screen, removing all graphs and variable controls
Rescale	this adjusts the scale of each graph so that the current values fit into the scale range.
Zoom in / Zoom out	graph scales – switches between the zoomed-in and zoomed-out view for all the graphs simultaneously.
Drug Levels	displays a box allowing you to show graphs of drug concentrations in various body compartments. Since drugs are dynamically added and removed, they do not appear in the main variable hierarchy of HOM.
Quick add	add a variable to the display using a quick search-by-name. Use this if you know the name of the variable you want to add. You can use the full name, abbreviated name, or the pathname of the variable.

Give Menu:

Fluids	a list of fluids that can be given as intravenous infusions
IV Drug bolus	brings up a list from which you can select a drug to be given as a bolus into the bloodstream

Oral drug	brings up a list from which you can select a drug to be given orally.

Tool Menu:

Medical Reports	shows the current values of certain variables in predefined lists, concerning Circulation, Fluids, Respiration, Energy, Hormones, Urine, Blood flows and Biochemistry.
Pathology analysis	shows the variables which are currently out of range and causing problems.
Fluid balance sheet	shows the amount of fluid that has been drunk, eaten, urinated, defaecated, sweated and exhaled in the course of a day, or a week.
Examine Patient	this contains several tools, allowing you to look at the patient's eyes, skin, ECG, blood film and symptoms
Diseases	select an item from this list to invoke a disease. Each item causes certain physiological parameters to change, to model a disease.
Script editor	allows you to create and edit scripts (small programs that give HOM instructions)
Options	allows you to change some configurable properties of HOM

Help Menu:

Scenario information	– shows a help file about the current scenario, and what the setup you have chosen is about
Physiology help	shows a help file with information on HOM's physiology model
Interpreter help	shows information on how to type interpreter commands into the log console, and how to write scripts
About	tells you about this version of HOM
Look and feel	allows you to change the style of the graphical interface of HOM

Graphical Interface Summary

1) Control bar
   a) Patient name: Double click to edit the patient's name and description
   b) State: Displays whether the patient is ill, unconscious or dead
   c) Time: The present time and date, in terms of the body clock
   d) Play: Stop or start the clock
   e) Time compression: The slider's values indicate
      i) Real time
      ii) 1 minute per second = x 60
      iii) 10 minutes per second = x 600
      iv) 1 hour per second = x 3600
      v) 6 hours per second = x 21600
      vi) 1 week per second = x 604800 (only works on fast computers!)
   f) Reset: Reverts all variables' values to the initial state
2) Menu bar
   a) `File`: Allows saving and loading of current variables' values.
   b) `Graph`: Commands to manipulate the scrolling graphs displayed in the main section of the display.
      i) `Rate`: Control the timebase of the scrolling graph, in pixels per 50ms.
      ii) `New Orbit`: Create graphs that plot one variable's value against another
      iii) `Setup`: Display a predefined set of graphs
      iv) `Close all`: Close all graphs
   c) `Fluids`: Select fluids and drugs to administer, which appear as intravenous infusions on the top-left. You can also administer boluses of drugs intravenously or orally
   d) `Tools`: Various dialog boxes to help with diagnosis of problems
   e) `Help`: Display help boxes on various aspects of the program
      i) `Scenario information`: Information on the current setup, including your learning objectives
      ii) `Physiology help`: Details of the physiological model, with explanatory diagrams
      iii) `Interpreter help`: Summary of commands for the interpreter in the console or in scripts
      iv) `About`: Information about the program version, and the current system
3) Tree
   a) Values of variables are displayed to the right of their names
   b) Expand or contract branches with the + or – buttons
   c) Right click on a variable to get a menu, or double-click to see info
   d) Use the `Display` button/menu to show a graph of the variable on the right
4) Graphs
   a) On a scrolling graph, click the name of the variable to see info
   b) Click on the scale above the graph to zoom in or out
   c) Drag in the black graph area with the left mouse button to make markings
   d) When the simulation is paused, drag upwards and downwards in the black graph area with the middle mouse button, or with `CTRL` + left mouse button, to scroll back and see previous graphs
   e) Right click in the graph to alter the timebase
5) Message pane

a) Messages about events in the simulator are displayed in this box.
b) On a blank line, type expressions and press return to evaluate them
   i) Variable paths, from the `body` or `environment` level of the tree (e.g. `air.O2`, `kidney.urine.Osm`)
   ii) Shortened variable names or long names also allowed
   iii) Mathematical symbols e.g. `+ - * / pow(X,Y), sin(X), max(X,Y) –` you can use the area as a calculator or scratchpad
   iv) Equations to set variables, e.g. blood.K=0.003

Keyboard shortcuts

When the cursor is not in a text-typing item, you can use the following shortcut keys:

1. `P` toggles pause and play
2. `R` reset
3. `Ctrl+A` quick-add a variable to the graph
4. `Ctrl+W` Close all graphs
5. `Ctrl+E` Edit controllers
6. `D` Oral drug
7. `Ctrl+D` Intravenous drug
8. `M` Medical reports
9. `Ctrl+S` Go to script console (and back again)
10. `F` Fluid balance chart
11. `F1` Help

Physiology Summary

Systems and controllers modelled in HOM

1) CVS
    a) Baroreflexes, Starling's law, Vasoconstriction
    b) Systolic and diastolic BP, individual organ perfusion
    c) 3 compartment model: ICF, ECF, Blood
    d) Capillary pressures and oedema
    e) Intracellular osmosis and electrolytes
2) Respiratory
    a) Intrathoracic pressure/volume
    b) Alveolar ventilation and water vapour
    c) A-a gradient and shunting
    d) Henderson-Hesselbach equilibrium
3) Metabolic
    a) Basal metabolism, Thyroxine, heat generation
    b) Respiratory quotient, expired gases
    c) Body mass, water, surface area
4) Muscle
    a) Exercise and efficiency
    b) Oxygen consumption, delivery and debt
    c) Vascular autoregulation
5) Kidney
    a) Autoregulation, glomerular filtration and leakage
    b) PCT, loop, DCT and duct reabsorption
    c) Na+, K+, H+, Glucose, Urea, Bicarb-
    d) Angiotensin, Aldosterone, ADH and erythropoietin
6) GI tract
    a) Reabsorption and transit time
    b) Acidification of stomach
    c) Protein synthesis and breakdown, Urea/creatinine production
    d) Insulin production, Glycogen and fat conversion
    e) Ketone formation, glucose/potassium channels
7) Brain
    a) Hunger, thirst, defaecation and urination
    b) Sedation, pain, nausea
    c) Symptoms
    d) Respiratory drive, Autonomic control
8) Skin
    a) Sweating, vasodilatation, cooling
    b) Skin colour

Advanced Introduction

Comments

HOM is a complex and rich system of equations, and produces new output every time it is run. Most of the possible combinations of parameters have never been tested! The program is under continuous development (as of August 2008).
If you try something which has an unexpected result, please let the author know.

APPENDIX

A. Setting Up

For advanced use of Human Physiology, the program can be unzipped. To do this in on a PC running Microsoft Windows 95 or better:

1. Run WinZip and open the file `'Phic.jar'`
2. Click on 'Extract'
3. Select a folder to expand to, preferably a new empty folder called `'Classes'`.
   From now on, the full pathname to this folder will be referred to as `'Classes'`.
4. Ensure that 'User folder names' is ticked
5. Press OK

Next you can create a shortcut to the program by following these steps:

1. Right click in an explorer window (or on the desktop), and select New/Shortcut.
2. Press 'Browse…', select the folder `'Classes'`, and press OK.
3. Edit the text in the box so that it reads:
   `java –jar phic.gui.PhicApplication –cp "Classes"`
   replacing `'Classes'` with the full pathname of the classes directory, that should be already in the box. Note that if the pathname to `'Classes'` contains spaces or other special characters, it must be **enclosed in quotation marks**. Note that this line is case sensitive.
4. Press OK
5. Select a name for the shortcut. (If all is well, the name `'java.exe'` should appear as a suggestion.)

You can also run the program from the command prompt:

1. Set the current directory to `'Classes'` using `'cd Classes'`. Ensure that the command prompt displays the path to the classes folder.
2. Type the command `'java –jar phic.gui.PhicApplication'`

When adding options to the command line, it is this line (commencing with 'java') that should be altered; either in the shortcut properties, or in the command line.

B. Locating Resource Files

Resource files are stored in the folder `'Classes/phic/resources'`. There are three types of resource file.

1) Text files (*.txt)
   These are simple ASCII text files that can be edited using 'notepad' or equivalent. They are often case-sensitive, and sometimes the correct spacing and tabbing can be important.
2) HTML files (*.html)
   These are standard web 'hypertext markup language' documents, which are displayed using the cascading-stylesheet model. They can be edited in notepad (for this you will need to know how to write HTML), or in another HTML editor, or even in Microsoft Word.
3) GIF images (*.gif)
   These images are in the compressed Graphics Interchange Format (CompuServe

bitmap format). Most have a 256-colour palette (though some have only two colours), and many have a transparency (mask) colour. They can be edited with any commercial image editing program, or with Microsoft Paint.

4) Patient files (*.patient)
These are files used to store patients' properties, created by the program itself. They should only be opened from within the program using the 'File/Open' command.

The important Text Resources

1. `FrameSetup.txt`
2. `GraphSetups.txt`
3. `OrbitSetups.txt`
4. `SimpleVariables.txt`
5. `Pathology.txt`
6. `Pharmacy.txt`
7. `BloodTests.txt`
8. `Screens.txt`
9. `ReferenceValues.txt`
10. `VariableInfo.txt`
11. `Fluids.txt`
12. `UnitOverrides.txt`
13. `Variables.txt`
14. `pO2Table.txt`
15. `pCO2Table.txt`
16. `SVP.txt`
17. `Controllers.txt`
18. `scripts/PhicScripts.txt`

C-style conventions

1. Text resources may contain comment lines, which begin with '`//`'. These lines are ignored by the program, and are just used for annotation.
2. In text resource files in a table format, there is a fixed number of items that determines how the rows and columns are organised when the file is read, and the 'white space' (tabs, spaces, and line breaks) are ignored. Therefore, having this **exact** number of items on every line is important.
3. Text resource files with a properties-definition format have a 'section name' in square brackets on one line, followed by a set of properties in that section, one per line. The properties often have the format of `'propertyname = propertyvalue'`, where everything after the equals sign is the value assigned to the property on the left.

C. Individual Resource Files

1. Variables.txt

This contains all the 'Visible' variables – i.e. all those with numerical ranges and common names, that can be displayed. The normal and initial values in this file may be edited, as may the common names.
It should **not** be necessary to alter this file to change the initial conditions; simply alter the values directly from the interface, and save the state. This can then be loaded on startup. (Note however that some initial values are actually used by the program as set-points for variables. To change these set-points, you will need to change this file.)
This file is an 8-column table, each row representing one numerical variable. Each numerical variable has the following columns in the table:

1) Initial value
2) Minimum of normal range of value
3) Maximum of normal range of value
4) Unit number. This represents the unit that the variable will be displayed in. For a list of these, see the programmers' documentation.
5) Type of variable (currently set to 1 for all variables, no meaning at present)
6) Short nane of variable
7) Long (common) name of value. Must not contain spaces.
8) Pathname (**full name**) of variable, giving its full location in the variable tree.

Currently, you will be able to alter items 1, 2, 3, 6 and 7 with meaningful results.

2. GraphSetups.txt

Each section creates an entry on the 'Graphs/Setup' menu in the program. When the entry is selected, each property under this section creates a variable display. The property name is the **full name** (pathname) of the variable to be displayed. The value of the property indicates how the variable is to be displayed; this should be one of the standard types (e.g. Graph, Value, Checkbox, Slider etc.).
To display a two-variable orbit plot, there should be two separate lines, first the *X-variable*=Orbit, then the *Y-variable*=Orbit.

3. FrameSetup.txt

Each section in this file represents a way of starting the program. To start the program using a given setup, use the section name of the setup (the name in square brackets) in the command line, e.g. for [Temperature],

```
Java –jar phic.gui.PhicApplication –cp C:/Classes setup Temperature
```

Under the section heading, you will find properties relating to how the program appears when started.

1. `GraphSetup`: this is the name of the graph setup to display on startup. It is the name as seen on the 'graph/setup' menu in the program. These names are taken from the section headings in the resource file GraphSetups.txt.
2. `ControlBar`: If this is 'Yes' then the bar of controls at the top of the window is displayed. The control bar includes the patient name, status, current time, the play/rewind controls, and the time compression.
3. `FluidsMenu`: if this is 'Yes' then fluids can be selected from the menu, and a drip can be put up. If this is 'No' then fluids via drip are disabled.
4. `ToolsMenu`: if 'Yes', then the tools menu can be viewed. This menu includes advanced dialog boxes for analysis.
5. `Console`: if 'Yes', then the text-box at the bottom of the screen is shown.

6. `LimitChecker`: if 'Yes', then near the bottom of the window, a list is shown of the short names of variables that are out-of-range.
7. `Tree`: if 'Yes', then the user will be able to select any of the variables from a tree on the left of the window.
8. `Started`: if 'Yes', then the clock starts when the program is loaded. Otherwise, it will wait for the play button to be pressed.
9. `Timebase`: This value is the timebase for the scrolling graphs. A value of 1 is roughly 1 pixel every 100ms (this varies from computer to computer). The normally selectable values are 0.1, 0.25, 1, 4, 10.
10. `TimeCompression`: This is the number of seconds in 'body-time' that elapse in one second of real time. A value of 60 will cause the simulation to run at one minute per second.
11. `PatientFile`: This is the name of the patient file to load at startup. This contains all the person's bodily and environmental parameters. The file name is relative to the 'Classes/resources' folder, is usually in the 'patient' subfolder, and is usually of the form 'patient/***.patient'.
12. `FrameWidth` and `FrameHeight`: these values determine the size of the frame at startup, in pixels.
13. `Resizeable`: if `'Yes'`, then the frame is resizeable by dragging the appropriate parts of the window.
14. `ScenarioHtml`: This is the name of the document file to display at startup. The file can also be viewed from the 'Help/Scenario help' menu in the program. The file name is relative to the `'Classes/resources'` folder, is usually in the `'patient'` subfolder, and is usually of the form `'patient/***.html'`. If this parameter is not specified, then invoking the 'help/scenario help' command shows the file `'patient/DefaultScenario.html'`

4. OrbitSetups.txt

The sections in this file create entries on the menu 'Graph/Orbits'. Under each section are two variables given by their **full names** (pathnames), one per line, indicating the X and Y axes of the orbit.

5. Screens.txt

This file is used by 'Tools/Medical Reports' menu item. Each section heading represents one tab (information screen) in the dialog. Under each section is a list of the variables, given by their **full names**, one per line, indicating which values are to be displayed on that screen.

6. Pharmacy.txt

This is the drugs definition file. Each section header represents one drug. Under the heading is a set of properties for that drug. The recognised property names are:

1. `Description`: a description of the drug's action. The description must not contain any new-line characters (but can wrap onto a second line, as long as you do not press return).
2. `UNIT`: This describes the normal unit in which the drug is measured. It is may be moles or grams.

3. `SINGLE_DOSE`: This is a value, in the above unit, indicating a normal single dose of the drug.
4. Physiological parameters; for a list of these, see the programmers' documentation. They include `RENAL_REABSORPTION, LIVER_METABOLISM, LIPID_SOLUBILITY, OSMOTIC_EFFECT`, and many specific receptor activities.

You may add drugs to this list, with your own combinations of properties. They will then appear in the list of drugs that may be administered (add to infusion, oral drug, intravenous bolus).

7. Pathology.txt

The information in this resource is used by the 'Tools/Pathology' menu item. Each property name in this file is the name of a pathological status, and the value of the property represents the variable out-of-range status that corresponds to that pathology. The value is of the form "`High`" or "`Low`" followed by the **full name** of the variable; e.g.

```
Acidosis=Low body.blood.pH
```

The upper and lower limits of normal are taken from Variables.txt.

8. Fluids.txt

A table with 9 columns, representing the following. Concentrations are multiplied by 1E-6 M

1. Bicarbonate concentration
2. Hydrogen ion concentration
3. Glucose concentration
4. Urea concentration
5. Protein concentration
6. Potassium concentration
7. Sodium concentration
8. Solid fraction (1,000,000 gives 100% solid)
9. Name of fluid (must not contain spaces)

Each fluid is used in various ways;

1. `Saline, Hartmanns, Dextrose, PackedCells, Colloid, Plasma` are used for infusions.
2. `Seawater, Hypertonic, Acid, Alkali` can be used experimentally with 'eat' or 'infuse' to create a perturbation.
3. `Blood`: not used, but represents normal blood concentrations.
4. `Water`: used by many models of physiological processes, e.g. osmosis; also drinking.
5. `Large`: used by ultrafiltration to remove protein.
6. `Ideal`: the fluid reabsorbed by the proximal convoluted tubule in the kidney.
7. `Food`: this is added to the stomach when the patient eats.

9. VariableInfo.txt

Each section heading is the full name of a variable. The rest of the section is displayed in the variable information box when a variable's name is double-clicked. The program 'intelligently' senses other variable names (full names or short names) and creates clickable hyperlinks to other variables.

10. ReferenceValues.txt

The reference values are taken from this file when they are displayed graphically in the variable information dialog box. It helps conceptualise the range of values a variable could take.
Each section heading is the full name of a variable. Each property in the section is the name of a reference point on the scale. The value after the '=' sign gives the value of this point on the scale.

11. UnitOverrides.txt

This file allows any variable's value to be displayed in customised units. This is useful in different settings (e.g. US hospitals) in which different conventional units are used.
Each section heading is the full name of a variable. The two properties that follow are:

1. `Unit`: the new unit abbreviation to display (excluding any prefix)
2. `Conversion`: a factor to multiply values by before displaying them.

For example, to display blood gas partial pressures in kilopascals, use

```
[body.blood.arterial.PO2]
Unit=Pa
Conversion=132894
```

D. Script Reference

The script facility is a powerful tool in the physiology model. Any mathematical function or operation can be used with any numerical variable, or any basic operation can be performed on any data object in the physiological model. A series of these operations can be combined into a script, which can be run once, or continuously over time.

1. Interpreter commands and syntax

The interpreter can be accessed from scripts (text documents with commands, executed using the "Tools/Scripts" dialog box) and also by typing directly in the console at the bottom of the window. (This window might be hidden in simplified versions)

The interpreter understands a simplified java syntax, including mathematical operations. You have access to all the viewable HOM variables, and in addition, it is possible to invoke many functions that are not normally accessible, e.g. those requiring one or more parameters.

1) `01234.567E89` is the format for numeric literal double-precision floating point values.

2) `true, false` represent boolean literal values

3) String literals can be entered with double quotation marks.

4) `+, -, /, *, ^` are the normal mathematical operators. Note that the exponent `x^y` can be replaced with `pow(x,y)`

5) `item.subitem` accesses an object found within another item, for example, `blood.arterial.O2` represents the arterial oxygen content.

6) `FullVariableName` accesses a variable by its full name, for example `ArterialO2Content`

7) Abbreviated names for variables may be used to represent the values, for example `APO2`

8) Functions can be called on objects, and objects or values can be passed as parameters, for example `blood.add( Pharmacy.dispenseDrug("Adrenaline",0.001) )`

9) Multiple commands can be placed in one line (e.g. for scripts) using a semicolon delimiter.

10) Temporary variables can be created to hold values in calculations, for example `extract = blood.ultraFilter(0.100)`

2. Useful functions from the library

These are a few examples of useful functions you can call:

1) `message("Message text")` sends a string to the console along with the current date and time.

2) `drink(volume)` adds a volume of water to the stomach

3) `eat(volume)` adds a volume of food to the stomach

4) `time` is a variable returning the current time and date as a string

5) `setRunning(true/false)` instructs the engine to stop or start calculating

6) `PhicApplication.markEvent("TEST")` marks the current time in the margin of the scrolling graph panel with the given text.

7) `actions.drinkAcid()`, or `actions.potassiumBolus()` etc. etc. do what you expect them to.

8) `HTMLMessagePane.showDialog("HTML_Resource_Filename")` shows a message dialog with the given html file displayed.

9) `Variables.forName("VariableName")` searches the variable list for a variable that matches the given name

10) `Container_Object.withdrawVol(volume)` returns a container with the specified volume of fluid removed from the given container. The container object could be any item which holds fluids, e.g. `ecf, icf, blood, kidney.urine` etc.

11) `Container_Object_1.add(Container_object_2)` transfers the whole contents of container_2 into container_1

12) `Container_Object.filterSolids()` returns a container with just the fluid compartment of the specified container. The solids remain in the original container.

13) `Drug_Container_Object.getDrugBinding( "DRUG_PROPERTY_NAME")` returns a value indicating the amount of binding of drugs to a particular receptor, as given in the file resouces/Pharmacy.txt

14) `PhicApplication.frame.doSetup("FrameSetup.txt", "Setup_Type_Name")` reinitialises the environment with a new setup section from the file resources/FrameSetup.txt

15) `DrugParser.createSubstance("Water 0.010 + K 0.005")` returns a container, whose contents reflect the string typed in the box. Permitted additions include the standard container variables (`Na`, `K`, `Glu`, `Prot` etc.), Fluid names as defined in the file `resources/Fluids.txt`, and drug names as defined in the file `resources/Pharmacy.txt`; each item being followed by a number indicating how much to add.

16) `Organ.completedCycles` returns the internal count of how many cycles of calculation have been performed in this instance of HOM.

For a full list of variables, subvariables and functions, refer to the JavaDoc source documentation (this is a large file weighing 7 MB!).

3. Root objects

The implicit root objects that are searched include

- The following packages, whose classes are directly visible:
  - `java.lang`
  - `java.awt`
  - `javax.swing`
  - `java.util`
  - `phic`
  - `phic.common`
  - `phic.gui`
  - `phic.drug`
  - `phic.doctor`
- The root objects, whose elements are directly apparent:
  - `Current.body`
  - `Current.environment`
- The root classes, whose static members are direcly visible:
  - `java.lang.Math`
  - `evaluator.MathExtra`

4. Mathematical functions

The mathematical functions (many of which are normally accessible in the `java.lang.Math`) class are accessible directly:

```
exp                     cosh
log                     tanh                    floor
log10                                           ceil
E = exp(1)              sech                    abs
PI                      cosech                  sgn
sqrt                    coth                    mod(x, base)
                                                frac
sin                     acos
cos                     asin                    ramp
tan                     atan                    truncatedRamp
                        atan2(x,y)              factorial
sec                                             eulerGamma =
cosec                   random                 0.57721...
cot                     min(x,y)                pow(x,y) = x^y
                        max(x,y)
sinh                    nCr(x,y)
```

Troubleshooting

Principles

Check the website

Visit the HOM website on http://www.homphysiology.com.
Often other's problems will be posted here, and you may find the solution to yours. There will be support available online using the blog or forums.

Email the author

The author can be contacted at: homphysiology@hotmail.com .
I am very happy to hear of

- any problems you have experienced with getting HOM running
- any errors that have occurred
- any unexpected physiological behaviour, ideally sent using 'Help/Send problem to author'
- any suggestions for improvements
- comparison with other programs you have used

Turning on the Java console

The console is a text window that shows internal information and errors that occur in HOM. It can be useful to switch it on if there are problems.
You can turn on the Java console by going to '`Start / Control panel / Java plug-in / Basic / Show console`'. This means that when HOM next starts, another window will appear. You can run HOM with a larger console by going to 'Start / Run' and typing the command `java -jar HOM.jar` (inserting the full path to the JAR file).

Turn on Debugging modes

- Use the 'Programmer' mode from the startup menu.
- Switch on the 'verbose' variable for an individual organ to see details of its calculations.
- Turn off individual organs by using the 'active' variable.
- Use `Tools/Clamp` variable values to fix the value of a variable

Errors

Jason has suffered a mysterious fatality

This is a symptom of one or more variables going out of range in an unorthodox fashion. It is most common when the time compression is very large, and results from a bad long-term approximation of a linear (rate-determined first-order) process. Please report the circumstances if you can reproduce this.

All values go to 'Error'

This is usually because of a division by zero. It shouldn't happen, but it sometimes does. Please do report the context of this if you can reproduce the situation.

Simplified Equations of HOM (in pseudocode)

Note that these are being continually improved, so the actual behaviour of the current version may differ.

Body

```
MetabolicRate = 1440 J/Cal * MuscleOxygenRequirement +
    BasalMetabolicRate
TotalHeatGenerated = BasalMetabolicRate / 1440 J/cal +
    MuscleOxygenRequirement
OxygenUsageRate = NonMuscleO2Use + MuscleOxygen
CO2ProductionRate = OxygenUsageRate * RQ
NonMuscleO2Use = BasalMetabolicRate / 8x10^6
BodyWater = BloodVolume + ExtracellularWater + IntracellularWater
BasalMetabolicRate = 2 MCal/day + 10^17 * Thyroxine error + 10^5 *
    BodyTemperature error
BodyMass = StomachVolume + ColonVolume + BladderVolume +
    BodyWater + FatMass + GlycogenMass + 15 kg
VisceralResistance.regulate(VasoconstrictorTone +
    ALPHA_ADRENOCEPTOR)
```

Lung

```
Ventilation                  = RespiratoryRate * TidalVolume
AlveolarVentilation rate     = RespiratoryRate * (TidalVolume -
    DeadSpace)
Water loss rate = Ventilation * (SVP(BodyTemperature) / 760 mmHg)
    * 0.804 ml/mmHg * 273/(273 + AmbientTemperature) * (1 -
    Humidity);
PAO2 = (BarometricPressure  - SVP(BodyTemperature) ) *
    AtmosphericO2
PACO2 = (BarometricPressure - SVP(BodyTemperature) ) *
    AtmosphericCO2

Equlibration of    PaO2  → PAO2
                   PaCO2 → PACO2

ArterialO2Concentration = O2Conc(PaO2) * (1 - PulmonaryShunt) +
    VenousO2Concentration * PulmonaryShunt
ArterialCO2Concentration = CO2Conc(PaO2) * (1 - PulmonaryShunt) +
    VenousCO2Concentration * PulmonaryShunt
```

CVS

```
PeripheralResistance = BloodViscosity / ( 1/HeartResistance +
    1/BrainResistance + 1/MuscleResistance + 1/RenalResistance +
    1/SkinResistance + 1/VisceralResistance)
CardiacOutput = StrokeVolume * HeartRate
ArterialPressure = PeripheralResistance * CardiacOutput
```

```
PulsePressure = ArterialPressure * (StrokeVolume + 75ml) *
    ArteryHardening * 10
CapillaryPressure = 0.14 * ArterialPressure
PlasmaColloidalOsmoticPressure         = 17.02 mmHg/mmol *
    PlasmaProtein
InterstitialColloidalOsmoticPressure   = 17.02 mmHg/mmol *
    ExtracellularProtein
Oedema = (IntracellularColloidOsmoticPressure -
    PlasmaColloidOsmoticPressure + CapillaryPressure)
```

Transfer ultrafiltered plasma to Extracellular fluid at rate of Oedema

```
DifferenceIEOsmolarity = IntracellularOsmolarity -
    ExtracellularOsmolarity
```

Transfer water from Extracellular fluid to Intracellular fluid at rate

```
    DifferenceIEOsmolarity * 10 L/min/Molar

EquilibrateConcentration( IntracellularUrea, ExtracellularUrea ,
    30 %/min)
EquilibrateConcentration( ExtracellularGlucose, BloodGlucose, )
IntracellularPotassium regulated to ExtracellularPotassium + 8mmol *
    (Insulin + BETA_ADRENOCEPTOR)
Erythropoietin.regulate(Arterial[O2]Concentration, -100, 0.03
    %/min)
RedCellMass.regulate(Erythropoietin, 10, 0.01 %/min)
BloodViscosity = 1+ 1.3 * (Haematocrit - 0.5)
```

Heart

```
RightAtrialPressure = 1.67 mmHg * (BloodVolume - 2 L) * (1.5 -
    Uprightness / 2)
Stroke Volume = (RightAtrialPressure - 1.43mmHg) /
    (StarlingCurvePeakRAP - 1.43mmHg) * peakContractility + 30ml
    * SympatheticTone + BETA_ADRENOCEPTOR - 150ml * (pH less than
    7.0)
HeartRate.regulate ( AP, 3000, 30 %/min)
Heart[O2]Use = 0.027 * CardiacOutput * (ArterialPressure -
    RightAtrialPressure)
```

Brain

```
Brain[O2]Supply = BrainFlow * Arterial[O2]Concentration
Nausea = StomachVolume + StomachSodium + 100 * OPIATE_RECEPTOR
Sedation = 2.7 * OPIATE_RECEPTOR + 1 * GABA_RECEPTOR
Hunger increases by Greed * ( (StomachVolume < 10 ml) + 1000 *
    (BloodGlucose < 4)  )
Thirst increases by Thirstiness * ( (0.05 * PlasmaOsmolarity error)
    + (0.001 * BloodVolume error) )

BodyTemperature    < 22 °C   Die of hypothermia
    < 32 °C   Unconscious
    < 40 °C   OK
```

```
    < 44 °C   Delirious
    >=44 °C   Die of hyperthermia
IntracellularVolume     > 40 L    Confusion
    > 45 L    Die of cerebral oedema
ArterialPressure   > 150 mmHg     Headache
    > 200 mmHg     Die of cerebral haemorrhage
    < 70 mmHg Feel faint
BrainO2Supply < 90 ml/min    Unconscious
    < 60 ml/min    Die of cerebral hypoxia
BloodGlucose  > 20 mmol/L    Feel bloated
    < 3 mmol/L     Feel faint
    < 2 mmol/L     Unconscious
    < 0.2 mmol/L   Die of cerebral hypoglycaemia
Nausea   > 2  Feel nauseous
    > 4  Vomit
Haematocrit   < 35%     Feel tired
    < 30%     Has an anaemic complexion
PlasmaPotassium    > 9 mmol/L     Fatal arrhythmia
    < 2 mmol/L     Feel weak
    < 1.5 mmol/L   Fatal arrhythmia

O2Drive = ArterialO2Concentration + ArterialPO2
CO2Drive = (ArterialPCO2 - 37.5 mmHg) / 2.5 mmHg
pHDrive = 1 - 10 * (BloodPH - 7.4)
RespiratoryDrive → (O2Drive^2 > 1) * (pHDrive > 1) * (CO2Drive >
    0.55) * (1 + Pain) + (1 - 1500 * OPIATE_RECEPTOR)
RespiratoryRate = 13 /min * RespiratoryDrive
TidalVolume = 510 ml * RespiratoryDrive

SympatheticDrive → 0.01 * (BrainO2Supply - 40 ml/min) - 0.4 *
    (BrainO2Supply - 140 ml/min) + 0.1 * Pain
ADH.regulate(PlasmaOsmolarity, 3E-9, 1 %/min)
ADH.regulate(BloodVolume, -0.1E-12)
Thyroxine.regulate(BodyTemperature, -0.03, 0.0001 %/min)
```

Kidney

```
RenalFlow = ArterialPressure / RenalResistance
GlomerularPressure = ArterialPressure^2 * 10 * (RenalResistance -
    20 mPRU) / RenalResistance / (1 + 2 * (VasoconstrictorTone^3 -
    1)
GlomerularFiltrationRate = (GlomerularPressure -
    PlasmaColloidOsmoticPressure) * RenalFlow * 1.8
UltraFilter Plasma to give volume GlomerularFiltrationRate
Add PlasmaProtein fraction GlomerularLeak

Extract Angiotensin * 10^10 of fluid of composition IDEAL_FLUID
Extract Glucose at rate GLUCOSE_TRANSPORT_MAXIMUM
Extract 80% of Sodium and 96% of Potassium
```

Extract all but 20 ml/min of water

Aldosterone *controls rate of* Sodium:Potassium exchange
Extract all but 13 ml/min of water
Excrete PlasmaHydrogen *at rate* 8 µL/min * PlasmaHydrogen
Extract PlasmaBicarbonate *fraction* (1 – 2 * bloodPH error)
Extract up to 82% of Sodium proportional to Angiotensin
Extract Potassium *fraction* (1 – Aldosterone + 170 nM) / 160 nM

Extract Water *to leave osmolarity of* MaximumMedullaOsmolarity * LOOP_REABSORPTION * (1 – ADH)

Aldosterone.regulate(PlasmaPotassium, $2x10^{-6}$, 10 %/min)
AngiotensinII.regulate(ExtracellularVolume, $-2x10^{-9}$, 5 %/min)

Muscle

MuscleEnergyRequirement = ExerciseJoules * 0.2389 Cal/J / EFFICIENCY
$Muscle O_2 Requirement$ = MuscleEnergyRequirement * 0.18 mL/Cal
$Muscle O_2 Need$ = $Muscle O_2 Requirement$ + OxygenDebt
MuscleOxygenUptake = minimum($Muscle O_2 Supply$, $Muscle O_2 Requirement$)
OxygenDebt increases by ($Muscle O_2 Requirement$ – MuscleOxygenUptake)

MuscleResistance = (1 + 15 * (VasoconstrictorTone – 1)) * (90 mPRU – $43x10^{-6}$ * MuscleEnergyRequirement)
$Venous O_2 Content$ = $Arterial O_2 Content$ – OxygenUsageRate / CardiacOutput
$Venous CO_2 Content$ = $Arterial CO_2 Content$ + ((CarbonicAcidProduction * BloodVolume) + $CO_2 ProductionRate$) / CardiacOutput

GItract

Extract 1 %/min *of* Stomach
Solids *to* Colon
99% *of fluids to* blood, *remaining fluids to* Colon
Transfer 1.7 ml/min *of* GastricSecretions *from* Blood *to* Stomach
Transfer HydrogenIon *from* Colon *to* Blood *at rate of* 40 %/min
If (PlasmaProtein error) > 0 *convert* PlasmaProtein *to* AminoAcid *at rate* 10 %/min, increasing PlasmaUrea *by* 1.4 mol/mol *and* PlasmaCreatinine *by* 10 mmol/mol *of conversion*
If (PlasmaProtein error) < 0 *convert* AminoAcid *to* PlasmaProtein *at rate* 5 %/min
Decrease ECFGlucose *at rate* 10^{-9} * MetabolicRate mol/min
Equilibrate ECFGlucose *and* BloodGlucose

```
If (ECFGlucose error) > 0 convert ECFGlucose to Glycogen at rate
    ECFGlucose error * 0.5 * Insulin - 0.018 * (0.01 - Insulin) +
    0.001 * Insulin
If (ECFGlucose error) > 0 convert ECFGlucose to Fat (rate as above)
Convert Glycogen to Fat at rate 0.00025 * Glycogen error
Ketones.regulate( Insulin, -0.2, 0.03 %/min )
RespiratoryQuotient → 1.00 - 1.0 * fatBreakdownRate at rate 0.01
    %/min
Insulin decays by 5 %/min
Insulin produced at rate (ECFGlucose error) * 10 +
    (StomachGlucose > 1 mol/L) * 0.01 * GlucoseTolerance^3
```

Skin

```
Remove Saline from Blood at rate SweatRate
BodyToSkinTransfer = 0.1 * (BodyTemperature - SkinTemperature) *
    SkinFlow Cal/min
EffectiveHumidity = ClothingIndex * (1 - Humidity) + Humidity
MaximumVaporRate = SVP(AmbientTemperature) / BarometricPRessure *
    (1-EffectiveHumidity) * 0.5
SweatHeatLossRate = 500 cal/g * Min(SweatRate, MaximumVaporRate)
RadiationHeatLossRate = (1 - ClothingIndex) * 0.5 *
    (SkinTemperature - AmbientTemperature)

BodyTemperature increases at rate TotalHeatGenerated / BodyMass - 0.05
    * BodyToSkinTransfer °C/min
SkinTemperature increases at rate 0.95 * BodyToSkinTransfer °C/min -
    (SweatHeatLossRate + RadiationHeatLossRate) / 5 kg
SkinResistance → 0.25 - 0.125 * (BodyTemperature error + 0.1 *
    SkinTemperature error) * (1 + 15 * (VasoconstrictorTone - 1))
    at rate 40 %/min
SweatRate = 0.0005 + 0.001 * (BodyTemperature error + 0.1 *
    SkinTemperature error)
```