A beginner’s guide to using the HumMod Model Editor and XML in conjunction with HumMod:

HumMod functions by using documents written in XML code. By using the Model Editor, you can edit the existing documents used by HumMod (note that this is not recommended unless you are proficient in both XML and physiology), or you can write your own XML documents that use the HumMod solver to run custom simulations.

In order to make your own models, it is necessary to know how to write XML coding. XML code allows for free use of whitespace, and any characters not within tags are ignored, meaning that formatting and comments in an XML document can be done almost any way that you wish. Observe:

```
Distance

Created : 10-Aug-96
Last Modified : 30-Nov-08
Author : Tom Coleman
Copyright : 2008-2009
By : University of Mississippi Medical Center
Solver : Digital Human V0.4
Schema : DES V1.0

We're shooting the shotput.  The idea here
is to find the angle (from horizontal) that
makes the shotput go the greatest distance
-- at a constant initial velocity.

Distance is the integral of velocity
over time while velocity is the integral
of acceleration over time.

This model is two-dimensional.  Acceleration,
velocity and distance are described using
vertical and horizontal components.
```

This is a “header” similar to the one that adorns most of the XML documents used by HumMod. This block of text does not code for anything at all due to a lack of tags—it merely gives information to whoever is reading the code.

All of the actual code must be contained within an “element.” An element consists of two “tags” and the information between them. The tags are keywords contained in less than and greater than signs, the second tag always being the same word as the first tag with a foreslash in front of it, a command which closes the element.

An example element: <var><name> ExampleVar </name></var>

The <var> tag creates a variable, and the <name> tag assigns a name to that variable, so the overall effect of this element is to create a variable named “ExampleVar.”

Notice that this element consists of multiple, layered tags. Almost every element in XML will, and, in fact, the document is really one giant element contained in the <model> tag.

Now we move on to the specific document used as an example in this tutorial, **Distance.DES**, a file which can be found in the model library.

Distance.DES is a very basic simulation that does two-dimensional kinematics based on the values for initial angle and velocity of a projectile that the user chooses. This program contains the basic physics equations needed to solve such a model and the code that determines how the results and options will be displayed to the user.

We will now list the code and its effects:

```
<?xml version = '1.0' ?> -- All documents must begin with this declaration
```

```
<model>  This opens the model’s code
```

```
<title><basic> Distance </basic></title> This is the name of the model
```

```
<structure>
<name> Model </name> -- Now the programming structure begins
```

```
<variables> ======================================
```

All of the variables used by the program must be declared in this variables section

We define the initial angle and velocity here
using fixed parameters. This information is
used at the start of the solution to start the
integrals.

```
<parm>
  <name> InitAngle </name>   This parameter is used to give an initial value for the projectile to
  <val> 70 </val>            be launched at.
</parm>
```

```
<parm>
  <name> InitVelocity </name>  Same as above, but with velocity
  <val> 10 </val>
```

```
</parm>
```

The horizontal acceleration is zero. The
vertical acceleration is gravity, or -9.8
m/sec^2 at the earth's surface.

```
<constant>
  <name> HorzAcc </name>  Horizontal acceleration is zero, as this model does not factor in air
  <val> 0 </val>           resistance.
</constant>

<constant>
  <name> VertAcc </name>   Acceleration due to gravity is constant
  <val> -9.8 </val>
</constant>

<var>
  <name> Radians </name>   This variable will be used to turn the angle given in degrees into
</var>                      radians for use in trigonometric calculations.

<var>
  <name> MaxAltitude </name>  This variable will be used to store the maximum altitude of the
  <val> 0 </val>               projectile once the calculations have determined it.
</var>

</variables>

<equations> ========================================
```

Rather than determine the distance traveled by the projectile through traditional algebraic methods, calculus can be used easily in a two-step calculation. However, both horizontal and vertical distances must be calculated independently of each other.

The horizontal velocity is the integral over
time of the horizontal acceleration. We'll
use some context code below to set the initial
velocity.

Horizontal distance is the integral over time of
the horizontal velocity.

```
<diffeq>
  <name> HorzVel </name>
  <integralname> HorzVel </integralname>
  <initialval> 0.0 </initialval>
  <dervname> dHorzVel/dt </dervname>
  <errorlim> 0.1 </errorlim>
</diffeq>

<diffeq>
  <name> HorzDis </name>
```

The first element here creates a calculation of the velocity based on the acceleration, and the second element creates a calculation of the distance based on the velocity.

```
  <integralname> HorzDis </integralname>
  <initialval> 0.0 </initialval>
  <dervname> dHorzDis/dt </dervname>
  <errorlim> 0.1 </errorlim>
</diffeq>
```

The vertical velocity is the integral over time of the vertical acceleration. We'll use some context code below to set the initial velocity.

Vertical distance is the integral over time of the vertical velocity.

```
<diffeq>
  <name> VertVel </name>
  <integralname> VertVel </integralname>
  <initialval> 0.0 </initialval>
  <dervname> dVertVel/dt </dervname>
  <errorlim> 0.1 </errorlim>
</diffeq>

<diffeq>
  <name> VertDis </name>
  <integralname> VertDis </integralname>
  <initialval> 0.0 </initialval>
  <dervname> dVertDis/dt </dervname>
  <errorlim> 0.1 </errorlim>
</diffeq>
```

These two elements function exactly like their horizontal cousins.

```
</equations>

<definitions> ======================================
```

In the definitions section, the equations that control your variables are created.

```
<block><name> Parms </name> =======================
```

The following code is executed when a parameter value is changed.

```
<def>
  <name> Radians </name>
  <val> InitAngle * System.Pi / 180 </val>
</def>

<def>
  <name> VertVel </name>
  <val> InitVelocity * SIN Radians </val>
</def>
```

First, the angle given in degrees is converted into radians, then that angle is used to convert the initial velocity into its component vectors.

```
<def>
  <name> HorzVel </name>
  <val> InitVelocity * COS Radians </val>
</def>

</block>

<block><name> Dervs </name> ========================

<def>
  <name> dHorzVel/dt </name>
  <val> HorzAcc </val>
</def>

<def>
  <name> dHorzDis/dt </name>
  <val> HorzVel </val>
</def>

<def>
  <name> dVertVel/dt </name>
  <val> VertAcc </val>
</def>

<def>
  <name> dVertDis/dt </name>
  <val> VertVel </val>
</def>
```

This block of definitions gives the variables from the differential equation section actual values so that they will be usable in the program

```
<if>
  <test> VertDis GT MaxAltitude </test>
  <true>
    <def>
      <name> MaxAltitude </name>
      <val> VertDis </val>
    </def>
  </true>
</if>
```

This element caps the vertical distance of the projectile at its max altitude to prevent strange errors.

```
</block>

<block><name> Wrapup </name> ======================
```

This stops the solution when height falls to below 0.

```
<if>
  <test> VertDis LT 0 </test>
  <true>
    <def>
     <name> VertDis </name>
      <val> 0 </val>
    </def>
    <stop/>
  </true>
</if>

</block>

</definitions>
</structure>

<math>

  <parms> Model.Parms </parms>
  <dervs> Model.Dervs </dervs>
  <wrapup> Model.Wrapup </wrapup>

</math>

<control> =========================================

<gofor>
  <solutionint> 5 </solutionint>
  <displayint> 0.05 </displayint>
</gofor>

</control>
```

This element creates a menu option that allows you to advance time, thus moving the projectile at the values selected.

Now most of the underlying math of the model has been coded for and finished, but there is still work to do. The following display section wraps up the program by creating the visual component of the XML code that will present the results of the equations and the graphs.

```
<display> =========================================
<panel>

<name> Distance (Shotput) </name>

<structurename> Model </structurename>

<showpanelname>
  <row> 0.5 </row><col> 1.0 </col>
</showpanelname>
```

This names the display and displays the name chosen at the top of the window. The <row> and <column> tags determine the position of every part of the display

```
<showgraph>
 <row> 2 </row><col> 32 </col>
 <high> 14 </high><wide> 30 </wide>
 <leftmargin> 7 </leftmargin>
 <xaxis>
  <name> HorzDis </name>
  <label> Distance </label>
  <scale><min> 0 </min><max> 10 </max></scale>
 </xaxis>
 <yaxis>
  <yvar>
   <name> VertDis </name>
   <label> Height </label>
  </yvar>
  <scale><min> 0 </min><max> 10 </max></scale>
 </yaxis>
</showgraph>
```

This creates the graph of the projectile's motion by giving its vertical position as a function of its horizontal position. Pay attention to the position commands. The <row> and <column> functions determine its starting position. The <high> and <wide> and <leftmargin> tags determine its size, and the <scale> tag determines the range of numbers on the axes. The <name> tags pull the variable used for the axis, and the <label> tag determines the name that is actually displayed to the user

```
<groupbox>
  <row> 2 </row><col> 1 </col>
  <high> 6.4 </high><wide> 30 </wide>
  <title> Initial Values </title>

<repeatlist>
 <name> InitAngle </name>
 <repeat>
  <reps> 50 </reps>
  <stepsize> 2 </stepsize>
 </repeat>
</repeatlist>

<slidebar>
 <row> 1.5 </row><col> 1 </col>
 <name> InitAngle </name>
 <listname> InitAngle </listname>
 <label> Angle </label>
</slidebar>

<showvalue>
  <row> 3.0 </row><col> 1 </col>
  <name> Radians </name>
  <format><decimal> 2 </decimal></format>
  <label> Radians </label>
</showvalue>

<repeatlist>
 <name> InitVelocity </name>
 <repeat>
  <reps> 20 </reps>
  <stepsize> 1 </stepsize>
```

The <groupbox> tag allows a space with several display elements to be created. Usually, elements are displayed one-by-one, but this command allows the coder to group the displays of various parameters, variables, and calculations by category so that visual messiness does not occur. The <showvalue> tag displays a value in the space you set aside for it, and the <slidebar> tag allows these parameters to be changed directly by the user.

```
 </repeat>
</repeatlist>

<slidebar>
 <row> 4.5 </row><col> 1 </col>
 <name> InitVelocity </name>
 <listname> InitVelocity </listname>
 <label> Velocity </label>
</slidebar>

</groupbox>

<showvalue>
  <row> 10.4 </row><col> 1 </col>
  <name> System.X </name>
  <format><decimal> 1 </decimal></format>
  <label> Time </label>
</showvalue>

<showvalue>
  <row> 11.8 </row><col> 1 </col>
  <name> VertDis </name>
  <format><decimal> 1 </decimal></format>
  <label> Height </label>
</showvalue>

<showvalue>
  <row> 12.8 </row><col> 1 </col>
  <name> VertVel </name>
  <format><decimal> 1 </decimal></format>
  <label> Vertical Velocity </label>
</showvalue>

<showvalue>
  <row> 13.8 </row><col> 1 </col>
  <name> VertAcc </name>
  <format><decimal> 1 </decimal></format>
  <label> Vertical Acceleration </label>
</showvalue>

<showvalue>
  <row> 15.2 </row><col> 1 </col>
  <name> HorzDis </name>
  <format><decimal> 1 </decimal></format>
  <label> Distance </label>
</showvalue>

<showvalue>
  <row> 16.2 </row><col> 1 </col>
  <name> HorzVel </name>
  <format><decimal> 1 </decimal></format>
```

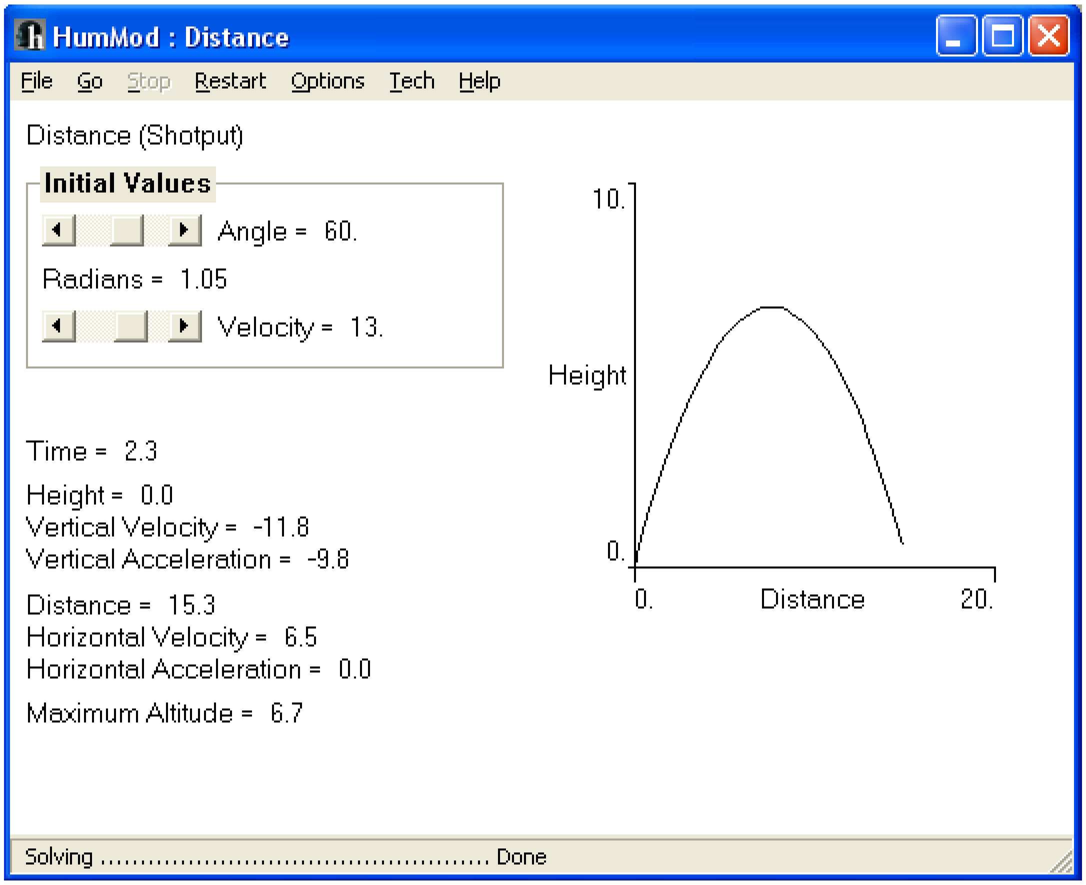
The display of various variables continues. Displaying all the program's variables, even the ones that do not mean much to the user, allows for greater transparency and clarity of your model's code.

```
  <label> Horizontal Velocity </label>
</showvalue>

<showvalue>
  <row> 17.2 </row><col> 1 </col>
  <name> HorzAcc </name>
  <format><decimal> 1 </decimal></format>
  <label> Horizontal Acceleration </label>
</showvalue>

<showvalue>
  <row> 18.6 </row><col> 1 </col>
  <name> MaxAltitude </name>
  <format><decimal> 1 </decimal></format>
  <label> Maximum Altitude </label>
</showvalue>

</panel>
</display>

</model>

End
```

</panel> and </display> end the display section, wrapping up all required elements of the model. </model> closes the code.

That marks the end of the code. Now let's focus on running it. First, open up the model editor. Then, use the folders to navigate the files on your computer until you find Distance.DES or whatever file you are attempting to run. Right-click the file and click "set as root XML document." Then you should be able to click the run option on the top toolbar to run the model. Additionally, you may need to set HumMod.exe as the solver under the settings tab if you have not already.

This is what the executed model should look like:



Distance (Shotput)

Initial Values

Angle = 60.

Radians = 1.05

Velocity = 13.

Time = 2.3

Height = 0.0
Vertical Velocity = -11.8
Vertical Acceleration = -9.8

Distance = 15.3
Horizontal Velocity = 6.5
Horizontal Acceleration = 0.0

Maximum Altitude = 6.7

Solving .... Done

Hopefully, this has served as a suitable introduction to XML modeling as used by HumMod by allowing you to learn what the different elements in the code do.

This information is merely intended as a quickstart guide for the HumMod Model Editor and a basic introduction to XML. For more in-depth and technical information regarding the XML Schema used by HumMod, please consult the HumMod Schema Guide.

Final reminder: Make sure to follow the formatting of all the elements contained in this program when you are editing it or creating your own model, because if you do not, there will be errors in the very rigid order the XML code follows and your document will not parse or run.