1 ECTo-VATR

1.1 Summary

ECTo-VATR is a system design tool that enables editing of the master model primarily through the hierarchical assembly and manipulation of components from the CML. It is focused primarily on empowering a designer in the early design phase to be able to incorporate and manipulate major design drivers and rapidly assess the qualities of system concepts. Resultant concepts can be used as the basis for more detailed design.

ECTo-VATR includes a 3D viewer called Zulu to represent the vehicle design concept and to aid in initial spatial layout and rudimentary packaging without the burden of a commercial CAD tool.

1.2 Launching ECTo-VATR

ECTo-VATR and Zulu are best run as client-side tools and running these on a remote machine will likely result in slow or degraded performance, especially the 3D visualization. ECTo-VATR and Zulu have currently only been built and tested on the Windows OS, however ECTo-VATR is based on the Qt framework and should be portable to linux or Mac without significant effort. Zulu is built using Unity which itself uses Mono and is supported only on Windows or Mac platforms.

ECTo-VATR normally runs in the following working directory: <installdir>\exec\ECTo-VATR\.

To run ECTo-VATR and Zulu (the 3D visualization) run the following bat files:

- <installdir>\exec\ECTo-3D\Zulu.exe (select the desired resolution and press 'Play' when the popup window comes up)
- <installdir>\exec\ECTo-VATR\startEcto.bat

1.3 Tool Overview

1.3.1 ECTo-VATR Main Window

ECTo-VATR is made up of several configurable panels. All but the system design panel are dockable widgets which can be undocked or moved around by dragging the title-bar of the panel. ECTo-VATR saves the layout configuration and window position when the program is closed and restores this when opened. This is saved in the layout.ini file which can be deleted or modified if desired. Initially there is no layout.ini file so the system comes up in a typical layout which looks something like the following:

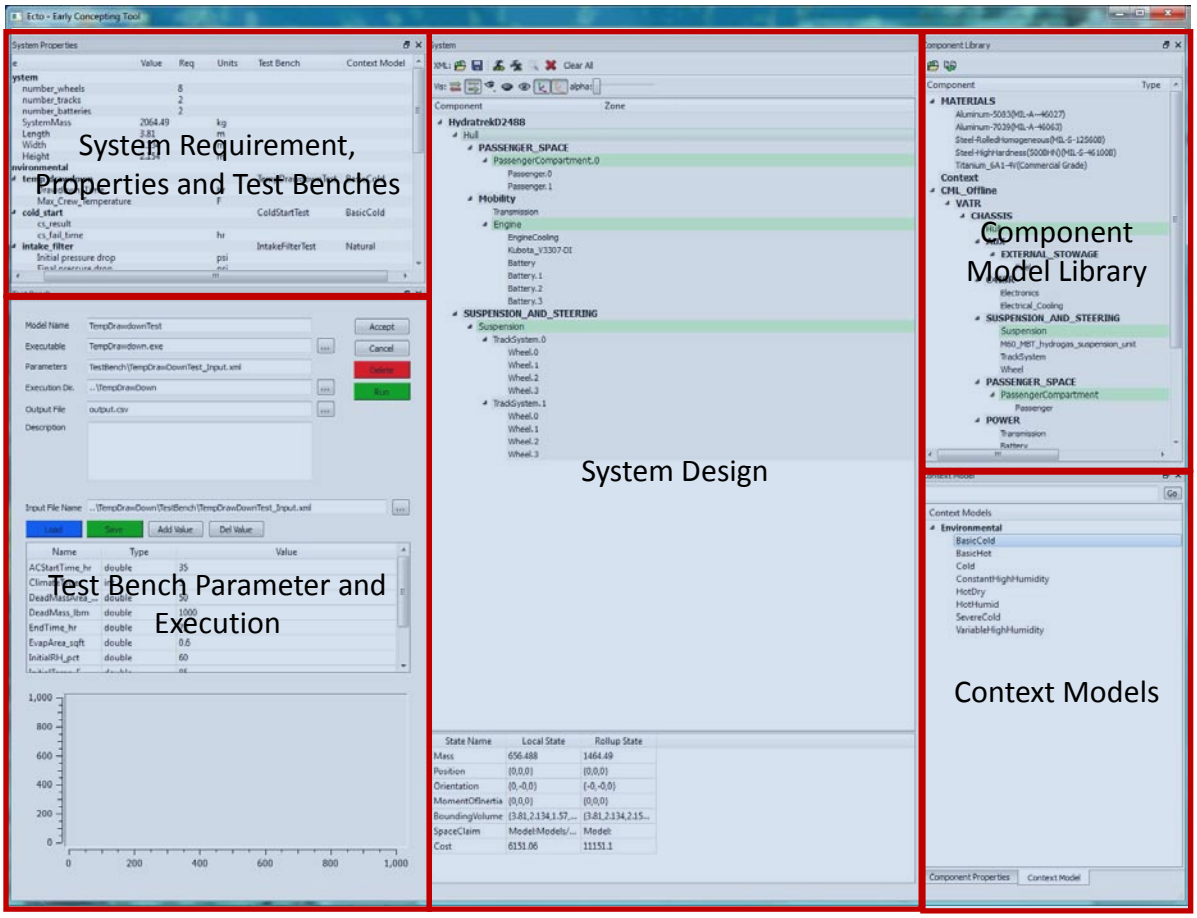


Figure 1 - ECTo-VATR Overview

By default when ECTo-VATR comes up the file system.xml is loaded. If you want to start from nothing you can press the clear all button.

1.3.2 Zulu

The 3D visualization component to ECTo-VATR, called Zulu runs as a separate process. No state is stored in the visualization and it is not required to be run for ECTo-VATR to function, though this provides the easiest way to move components around and get a spatial view of the system. If Zulu is started after ECTo-VATR or gets out of synch for any reason you can always press the 'Synch Vis with the system' button  located in the System Visualization Toolbar.

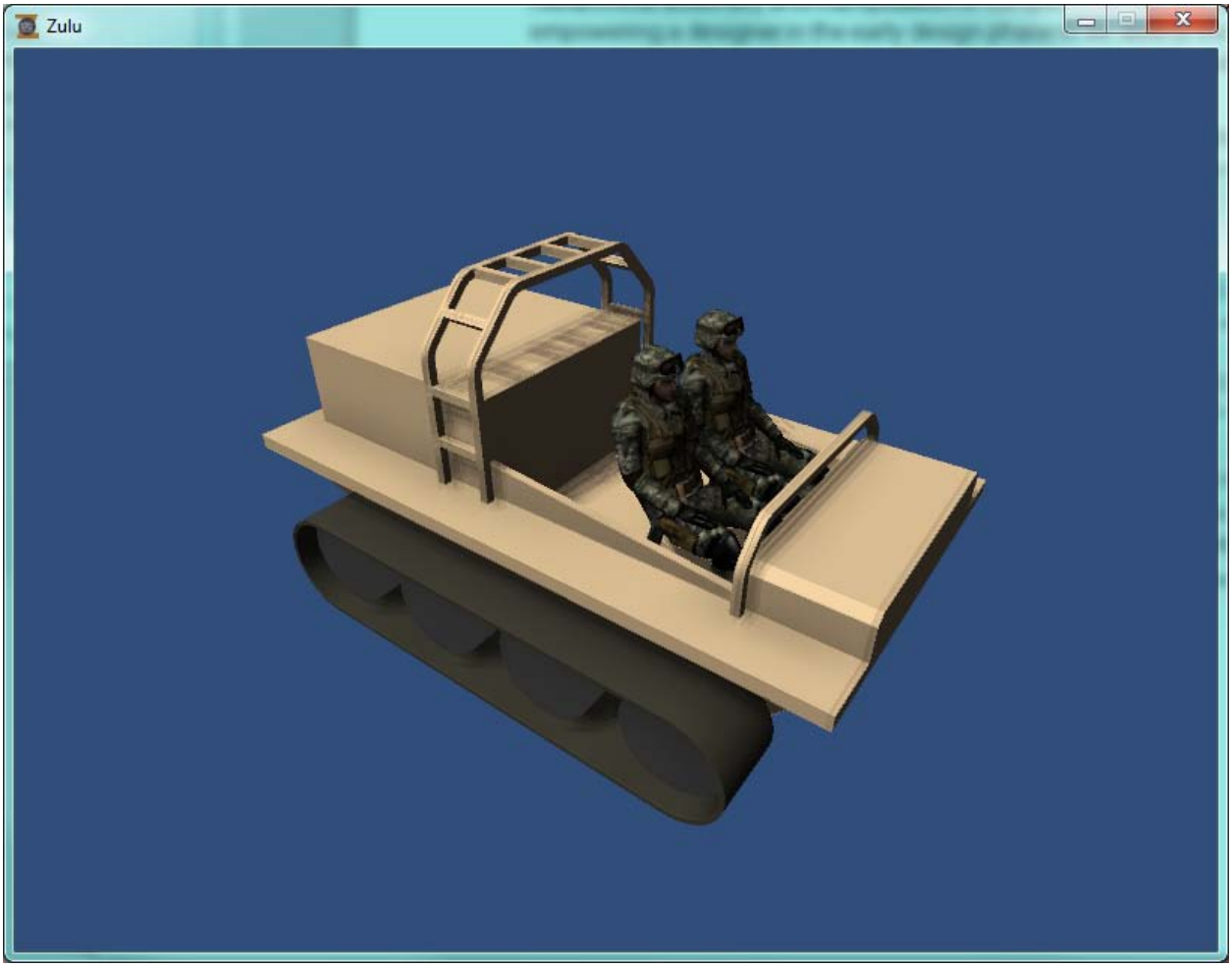


Figure 2 - Zulu (ECTo-VATR's 3D Visualization)

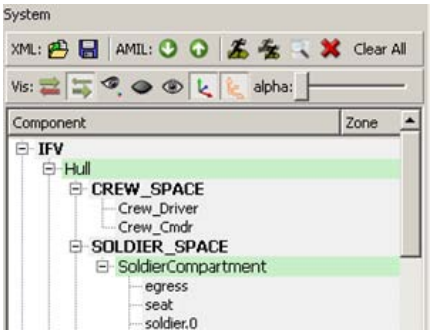
1.3.2.1 Zulu Camera Control

Zulu starts out in a free fly camera mode where by pressing the left mouse button and moving the mouse changes the direction of the camera. When the left mouse button is down the WASD keys translate the camera around the scene.

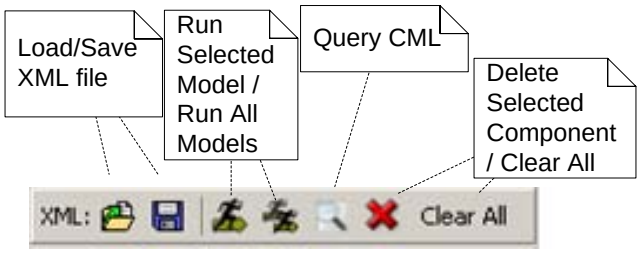
You can tether the camera to look at any node in your system by selecting a node and pressing the 'Tether Camera to Selected Node' button  . Once tethered the camera spins around the selected

node by holding the left mouse button down and dragging it. Rolling the middle mouse button will zoom the view in/out. To go back to free fly mode press the 'g' key or 'tab' to get the Buttons Window and press the 'Free Camera' button under the Controls tab.

1.3.3 System Design panel



1.3.3.1 System Design Toolbar



XML load and Save buttons store/load the state of the system design using local XML files. Loading a new system files clears the existing system.

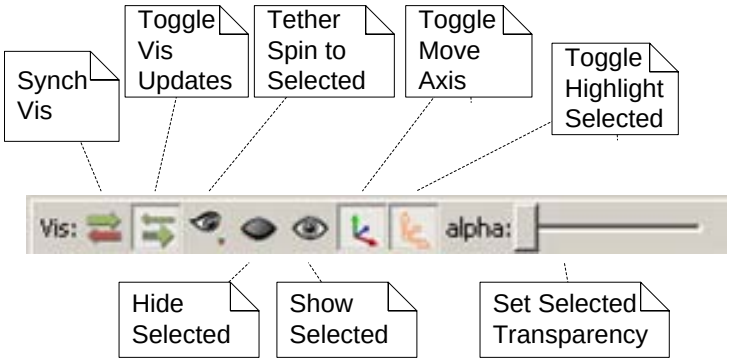
Run Selected Model/Run all Models: ECTo-VATR models can be run explicitly by selecting a model or more commonly all at once by pressing the 'Run All Models' button in the System Design Toolbar. Run all models execute all of the models in the System Hierarchy. See ECTo-VATR Models section below for more information.

Query CML creates a SPARQL query for refinements of the selected component. It generated as query for things in the ontology that are valid refinements of abstract item selected. Note for the demo as delivered the only component that had the CML populated for it to illustrate how this works is the 'Engine'. See the Demo walkthrough below for an example.

Delete Selected Component: removes the selected component *and all its children* from the system hierarchy.

Clear All: deletes the entire system hierarchy.

1.3.3.2 System Visualization Toolbar



The synch vis button clears what is in the Vis and resends the entire system design. No state is ever persisted in the visualization so it is generally ok to do this any time. Once synched the vis typically should stay in synch as changes are made. Zulu can be closed and restarted anytime and the synch can be used to re-establish the 3D representation of the system in ECTo-VATR.

Toggle Vis Updates is enabled by default and generally does not need to be changed. When enabled tells the Zulu visualization to automatically send positional updates back to ECTo-VATR when things are moved around. A designer may want to disable this if they want to manipulate the 3D visualization without changing the system.

Tether Spin to Selected forces the camera to always look at the selected node in the system hierarchy.

Hide selected tells the visualization to not render the component in the visualization. This is only a visual preference and does not affect the system design and is not saved in XML or AMIL.

Show selected turns a previously hidden component visible again.

Toggle Move Axis turn on/off the Axis control in Zulu which allows you to move or rotate a component.

Toggle Highlight Selected turn on/off the highlighting of edges around the component which is currently selected.

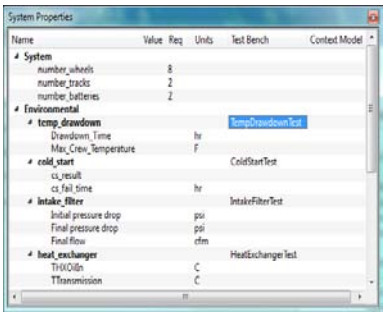
Set Selected Transparency makes the selected component's alpha value so it becomes transparent.

1.3.3.3 System Hierarchy

This is where the hierarchical representation of the system is displayed. You can drag and drop components around.

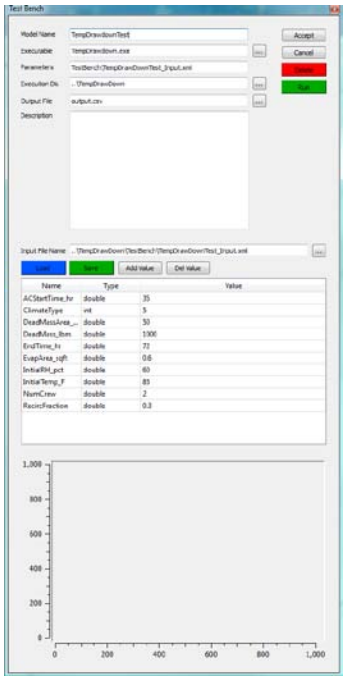
As you select components in the tree you will see the values of the major states displayed in a table under the system hierarchy tree. The 'Local State' column is the state values for *only the component selected*. The 'Rollup State' column displays a rollup of values for the selected state and all that states children. System level rollups for Mass, Cost, Length, Width and Height are always displayed in the System Properties panel.

1.3.4 System Requirements and Properties Panel



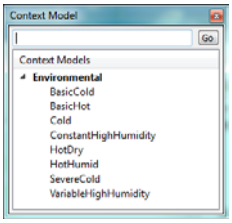
This panel stores all the system level properties or any properties that need to be exchanged between component models. The 'Value' column indicates the current systems estimated performance and the 'Req' column is used to maintain the system requirement or derived requirements for that property. Most key system inputs are considered requirements but they can be adjusted by a designer so they can assess how particular inputs drive a design. The test benches are seen in the 'Test Bench' column and they are selected by double clicking the text. When a context is selected for the current test bench it is seen in the 'Context Model' field.

1.3.5 Test Bench Parameters and Execution Panel



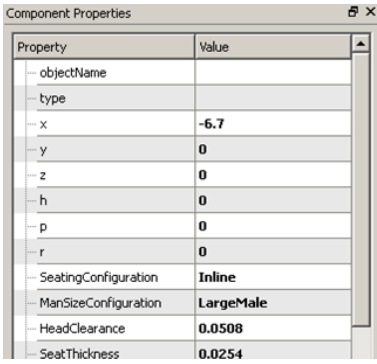
This panel shows and allows one to edit the parameters for the current test bench. The panel is broken down into three sections. The top section is the information needed to run the test bench executable. The middle section is the input parameters for the test bench. The bottom section is the output graph that was generated during the execution of the test bench. To run the test bench the user needs to click on the Run button. If the execution of the test bench takes a long time the panel will display a progress indicator.

1.3.6 Context Model Panel



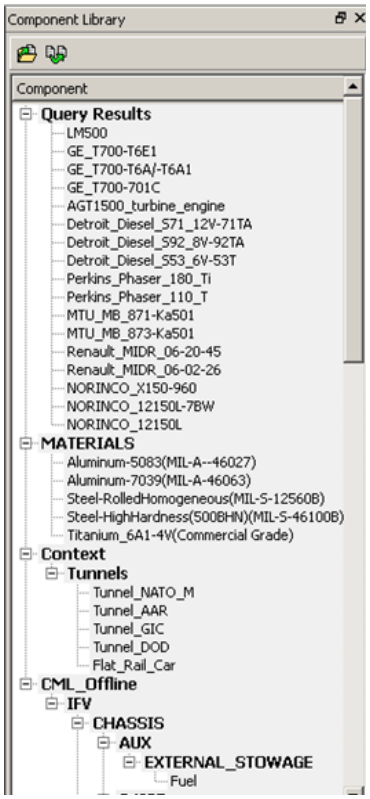
This panel shows the available context models for the current test bench. Double clicking the text will select which of the context that is used when the next run of the test bench. ECTo-VATR will automatically change the parameters in the current test bench that corresponds to that context.

1.3.7 Component Panel



This panel shows and allows one to edit the properties of a specific component. Components with associated models will frequently have additional properties that are specific to that model. For example, the PassengerCompartment model uses additional properties to construct the soldier compartment such as seating arrangement, squad size assumptions, clearances, etc.

1.3.8 Component Model Library Panel



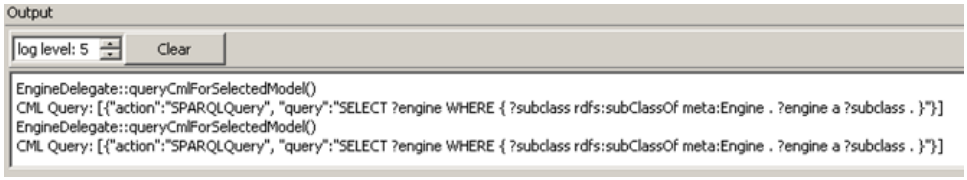
The Component Model Library panel shows components from the CML which can be included into the system hierarchy by dragging a component from this panel to the parent in the system you want to attach this component to. Alternatively, you can select a parent in the system hierarchy and select a component in the CML panel and press the Copy Selected Component to CML button  .

Components can be loaded into this panel either through XML files using the load file button 

As additional ways to interact with and search a CML are established this panel can evolve to support additional approaches. Additionally, CML repositories should be able to easily export indexes of components into a compatible XML file. The file CML.xml in ECTo-VATR's working directory provides an example of various CML components used to build up the IFV in the demo walkthrough below.

1.3.9 Tool Log

The tool log provides informational output to the user as the tool is used. The output level can be adjusted so more or less output is displayed in this panel. The clear button wipes the current contents away.



1.4 ECTo-VATR Models

Certain nodes, indicated by having a green background in the System Hierarchy, have Models associated with them. These are typically parametrically driven components or systems that change their properties or are 'built' based on inputs to those models. For example the 'PassengerCompartment' Model, when executed takes the requirement from the System Properties panel for 'number_passengers' and various other inputs in the Component Properties panel to construct the space claim and mass for a the Passenger Compartment. If you change the number_passengerss requirement and run this model you will see the number of soldiers and the various properties of the compartment, bench, egress volumes, etc. change to reflect this.

ECTo-VATR models are run from the System Design Toolbar, either individually on a selected model or by executing all models in the system. Running all models traverses the tree and executes models from children models up, models under the sub-tree of another model node are executed before the ancestor node's model. Executing Models can retrieve and set data in the System Properties widget as they run, so one models output can be used as input or modified by another model. This data flow can also be used to control the execution order of models or to establish an implicit causal network of models.

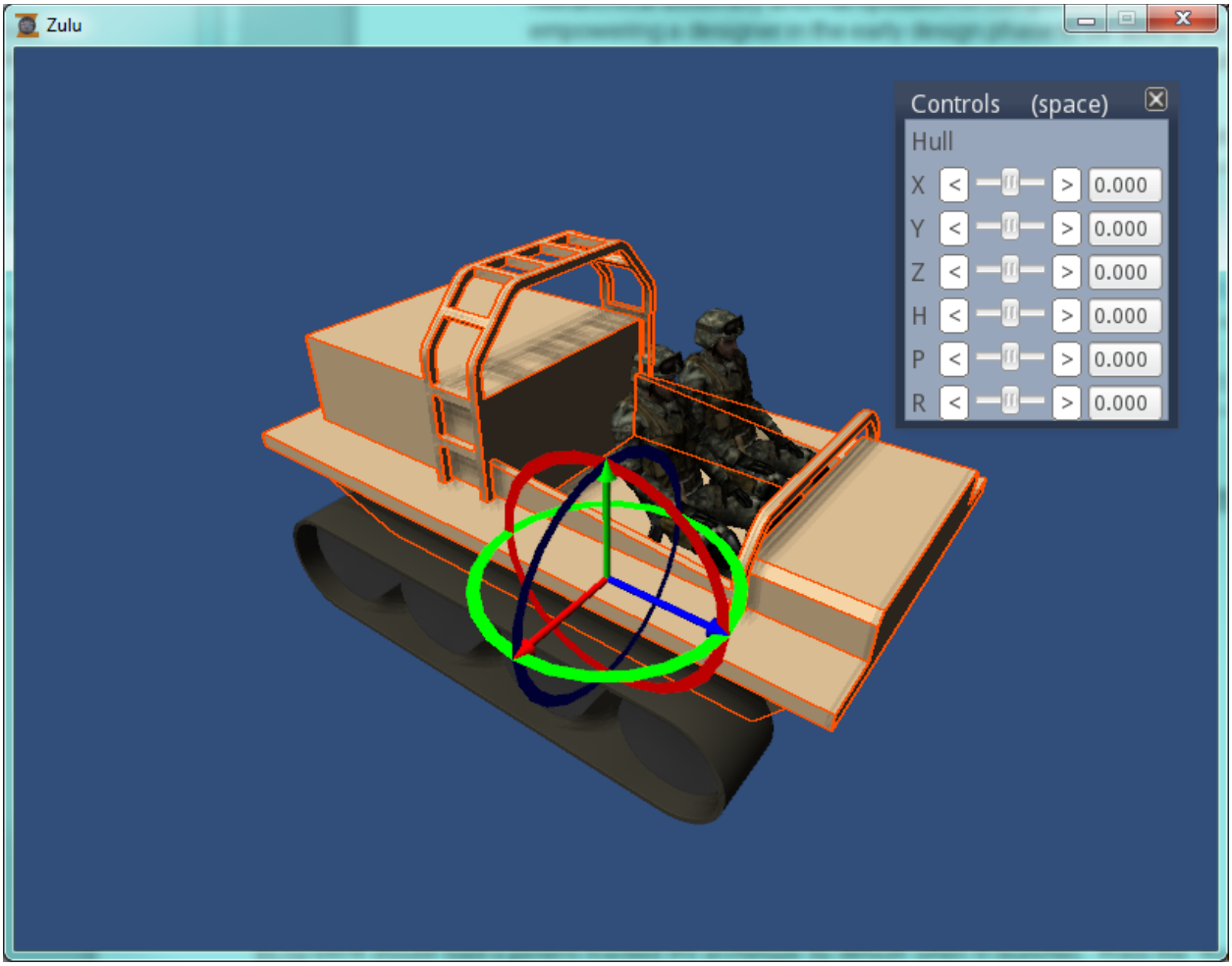


Figure 3 - Hull Positioning

1.5 Example Walkthrough

The following demo walks through an example of how ECTo-VATR could be used to quickly assess different system concepts and how that system operates in against a set of test benches.

1.5.1 Load Tracked VATR archetype

ECTo-VATR should load a generic tracked VATR archetype by default when it launches. Press the 'Run All Models' button in the System Design Toolbar.

1.5.2 Synch with 3D Vis (Zulu) and system properties

If Zulu does not show current vehicle press 'Synch Vis with System' button

1.5.3 Modify system requirements (increase number squad from 6 to 9) and execute all models again (rebuild system)

Change number_passengerss from 4 to 2 in the Systems Properties panel.

Press the 'Run All Models' button in the System Design Toolbar.

1.5.4 Observe quick look mobility model and Engine Power derived requirement

In the System Properties panel, as you modify the vehicle a quick look calculation is made that generates a minimum Engine Power that would be required for a vehicle of this weight class to achieve the required maximum speed (as show in the System Properties as top_speed). If you modify the system to make it heavier the required Engine Power will go up. Additionally, if you modify the top_speed requirement in the System Property panel this will also change the minimum required Engine Power. This calculated, or derived, requirement for minimum Engine Power is used when you query for viable refinements of Engine.

One way to quickly change the weight of the vehicle is to modify the 'AvgThickness' component property for the 'Hull' Model component. You can change the average hull thickness from 0.02 m to 0.03 and run the Hull model (or Run All Models) to represent a heavier armored vehicle.

1.5.5 Replace abstract engine with specific engine

Delete 'AbstractEngine' by selecting it in system hierarchy and pressing 'Delete Selected Component' button.

Drag an engine from the CML to the 'Engine' model component.

Press the 'Export to AMIL' button to save change to AMIL graph (Master Model).

1.6 Building ECTo-VATR

ECTo-VATR is built on top of the Qt framework and should be able to be compiled and run under windows/linux/Mac though only project files are supplied for Visual Studio.

Building using Visual Studio for Windows (XP or newer)

Requires:

- Visual Studio 2010
- Qt 4.5 or newer built for the version of Visual Studio you are using
- Visual Studio Qt addin (tested using version 1.1.9)

Open <svnroot>\VATR\VATR_VS2010.sln in Visual Studio 2010 and build release configuration.

1.7 Developers notes

ECTo-VATR is written in C++ and built on top of the Qt application framework. Zulu is built using the Unity Engine and communicates with ECTo-VATR using UDP messages.

1.8 Path Forward

While useful in its current state, ECTo-VATR is a prototype of an early system conception, analysis and test tool and has the potential to evolve into a more powerful system engineering asset.

Architectural modifications

- Migrate VATR specific delegates and models into dynamically loadable libraries, these libraries can then be stored and brought into ECTo-VATR from the CML.
- Include Ability to do multidimensional sweeps across input values or enumerated sets. Add output tables and data export utilities to facilitate analysis of this data.
- As statistical models are incorporated allow for execution of Monte Carlo runs.
- Fully incorporate a Design Set node that can be used to represent alternatives and any level in the hierarchy and tools.
- Enhance the System Properties panel to allow for a designer to adjust values and still maintain original requirements perspective. Could add a third column for actual requirement, a column for value to use as system input and a value for actual estimated output.
- Expand and make it easier to use server based context models such as those provided by OSCAR, the Dynamic Context Server and the Terrain Server.

ECTo-VATR then becomes more useful and powerful as more models are added and integrated together and the fidelity and accuracy of the models are improved. While the example models and analysis that currently exists in ECTo-VATR are related to the sample VATR with some relatively minor changes the underlying infrastructure can support the design of many types of systems.