

Realtime Driving Simulation Using A Modular Modeling Methodology

Richard Romano

Realtime Technologies, Inc.

Copyright © 2000 Society of Automotive Engineers, Inc.

ABSTRACT

The use of driving simulation in vehicle design and development is growing. For maximal benefit, vehicle models used in the driving simulator must be rapidly reconfigurable and easy to develop. To evaluate potential modeling concepts, vehicle dynamics and vehicle subsystems are developed using modular model components. These are integrated with simulator cueing subsystems, using the same modeling concepts, to build a complete driving simulator. It was found that the vehicle dynamics and the simulator could be reconfigured easily to meet user needs.

INTRODUCTION

Over the past twenty years offline vehicle dynamics simulations have been used extensively in the automotive industry. As computer performance increased vehicle dynamics applications were introduced into hardware in the loop simulation and operator in the loop simulation. To meet the needs of vehicle designers the complexity and fidelity of the vehicle dynamics used in operator in the loop simulation has been increasing. Currently, driving simulators are being used to assess the impact of new systems such as active safety devices, active suspension systems, and advanced powertrain designs [1][2]. To evaluate these systems in the simulator, a model of the new system must be introduced into the vehicle dynamics. The purpose of this paper is to develop a modular modeling methodology that supports an easily reconfigurable realtime vehicle dynamics model into which new vehicle subsystems can be rapidly introduced. This modeling methodology will be applied to both the vehicle dynamics and the complete driving simulator to maximize the reconfiguration speed and model flexibility. The modular components will be developed and assembled in SimCreator a graphical simulation tool from Realtime Technologies, Inc. The vehicle dynamics model will be introduced first, followed by the rest of the simulator.

VEHICLE DYNAMICS

The vehicle dynamics model can be broken down into several modules: powertrain, brake system, steering

system, tire model, and suspension [3][4]. Each of these systems can in turn be broken down into additional submodules. The vehicle powertrain for example has been modeled previously in the both EASY5 a graphical simulation package from Boeing Computer Services [5] and Simulink a graphical simulation package from the Mathworks, Inc. [6] using a modular approach. Ciesla [5] developed powertrain modules for the engine, torque converter, differential, gears, and tires. The selection of how the vehicle model is broken down into modules directly effects the reconfigurability of the simulation. For the current analysis, the top level of the vehicle dynamics was broken down into the components shown in Figure 1.

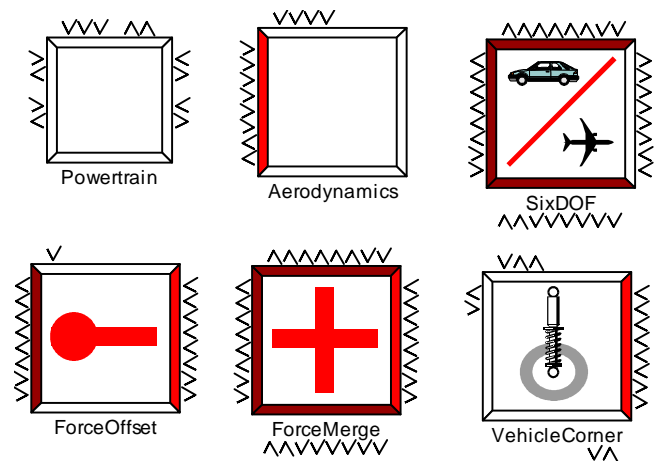


Figure 1: Vehicle Dynamics Components

The components are as follows: Powertrain – calculates all powertrain and brake system effects and outputs the four wheel speeds of the vehicle, Aerodynamics – calculates the aerodynamic forces on the vehicle, SixDOF – calculates the Newton Euler equations of motion for the given forces, ForceOffset – calculates the position, velocity and acceleration at an offset and translates the forces at the offset back to the center of gravity, ForceMerge – sums up forces, VehicleCorner – calculates the independent suspension and the tire model at a vehicle corner, and outputs the forces acting

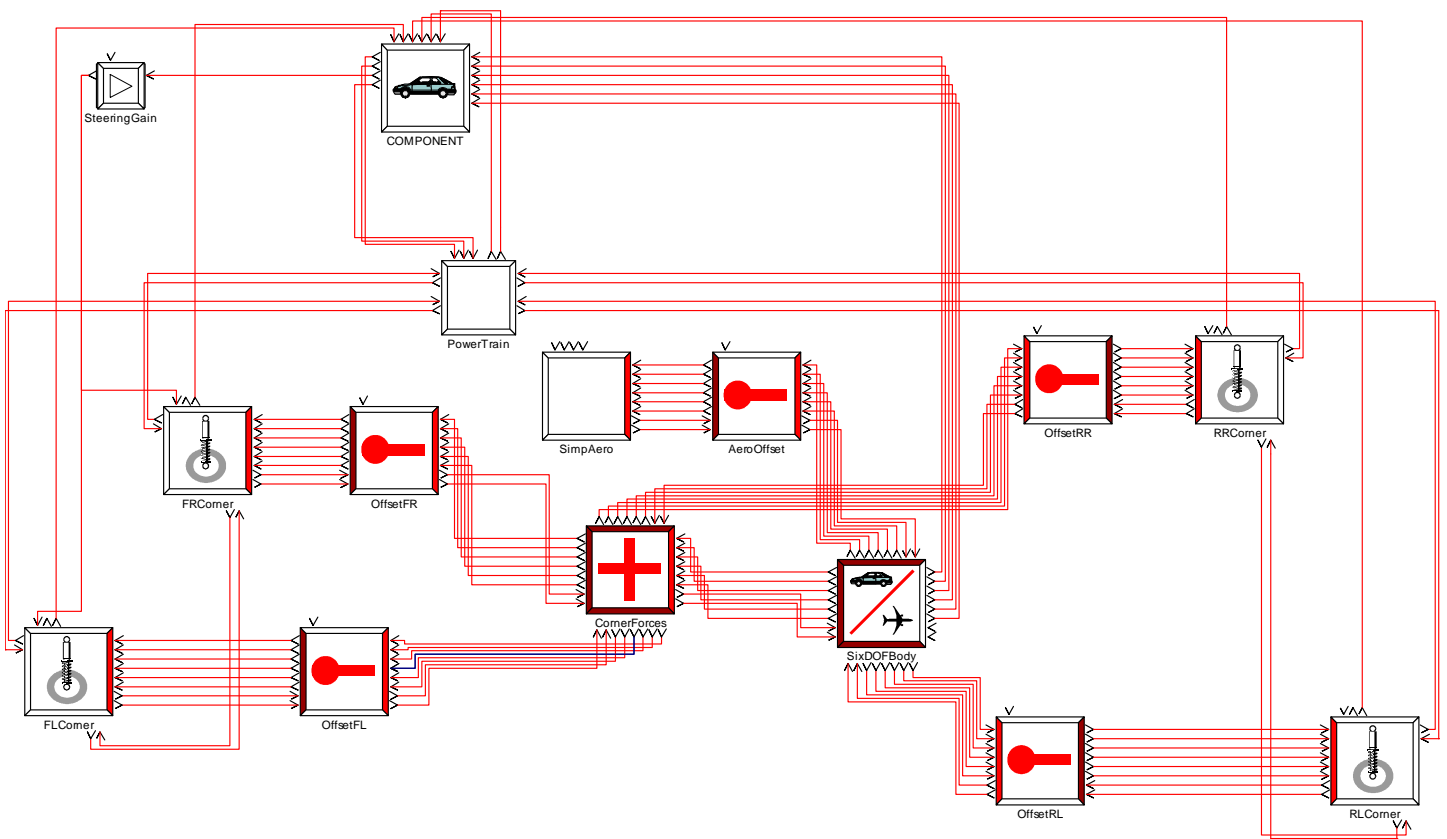


Figure 2: Complete Vehicle Dynamics

at the corner and the torque acting on the wheel. The components are designed for reuse. In addition using SimCreator's connection paradigm, the connections between components have been designed for reuse. For example, the force components can be used both with the vehicle corner and the aerodynamics module. The modules are assembled together to generate the complete vehicle shown in Figure 2. The VehicleCorner component has been replicated four times for each corner of the vehicle. The corners have been connected to the powertrain, aerodynamics and SixDOF modules using the offsets and force merge components. The block at the top of the diagram named COMPONENT is SimCreator's method of allowing components in the model to exchange data with models at the next higher level (which is our case is the complete driving simulator).

POWERTRAIN

For complex modules such as the powertrain, it is important to break out the component into its own model. This in turn allows for greater understanding of the module and for rapid changes to be made. For the powertrain a set of angular dynamics components are required. These are shown in Figure 3 and are similar to those developed by Ciesla [5]. As with the force components in the vehicle model, the connections of the angular dynamics modules have been designed so that a variety of components can be connected together. These connections consist of angular velocity and torque inputs and outputs. These modules are assembled

together to form a complete front wheel drive powertrain as shown in Figure 4.

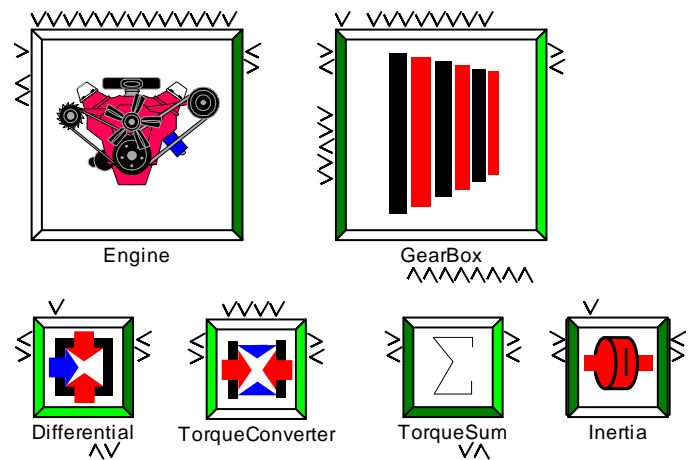


Figure 3: Powertrain Components

The Inertia components are used to represent the individual wheel inertias and the TorqueSum component is used to sum the torques coming from the tire model and the brake model. The wheel speeds are output from the powertrain model to the rest of the simulation through the COMPONENT block. Wheel torques calculated in the tire models in the vehicle corners form the input to the powertrain model. In addition to the angular dynamics modules, the brake system is also placed in the powertrain model.

Components such as the differential actually consist of two subcomponents: a torque calculation component

and a velocity calculation component. During powertrain simulation, velocity calculations start at the wheel inertias and the engine and travel to the torque converter, which calculates the torque applied both up and down stream. The torque calculations then flow back to the wheel inertias and engine. These torques provide the information to perform another integration step. The order of calculations is generated automatically by the simulation tool. Similar calculations are performed for the forces in the top level vehicle dynamics model but in that case there are position, velocity, force, and acceleration passes.

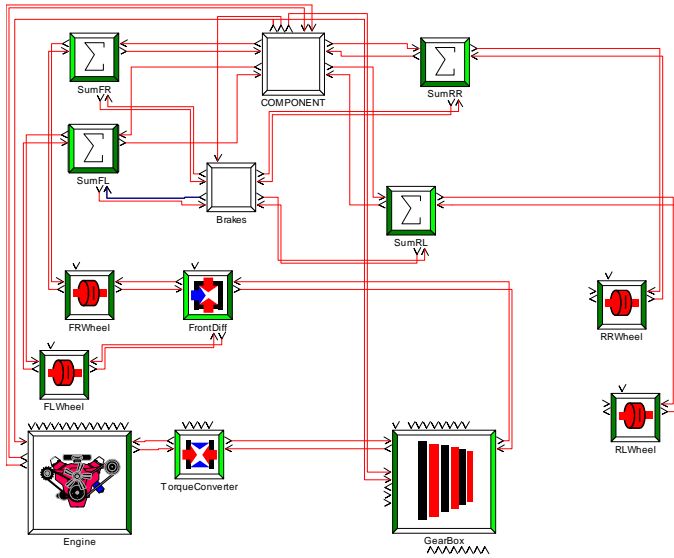


Figure 4: Front Wheel Drive Powertrain

OTHER VEHICLE SUBSYSTEMS

Vehicle subsystems with a level of sophistication comparable to the powertrain, such as the brake system, steering system, and vehicle corner modules were modeled similar to the powertrain.

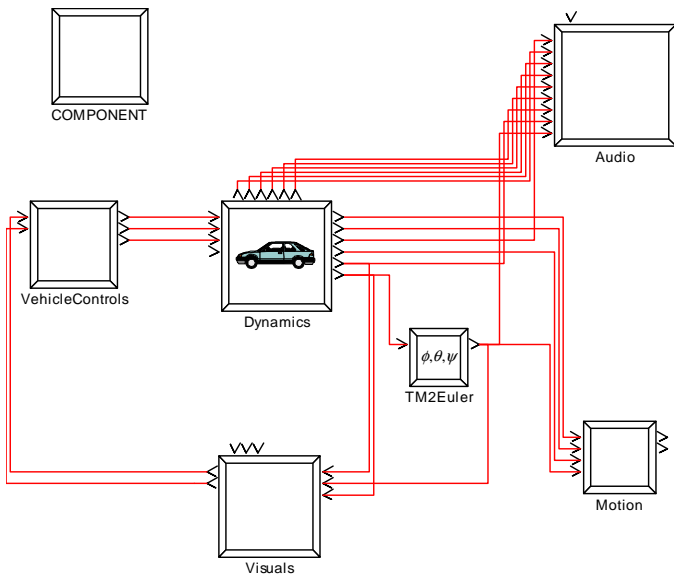


Figure 5: Driving Simulator

DRIVING SIMULATOR

The complete vehicle dynamics can now be inserted into a driving simulator for test and evaluation. A typical driving simulator has the following cueing systems: audio, out the window graphics, driving control interface, and motion. These can be encapsulated as modules and integrated with the vehicle dynamics to form the driving simulator shown in Figure 5. In this case the simulation tool would resolve the calculation order for the entire simulation. For example vehicle controls would be calculated first, follow by the vehicle dynamics and then the various cueing systems.

MOTION SOFTWARE

Each of the cueing systems (motion, audio, visuals, etc) can be quite complex and if required should be broken out into its own set of components. For the motion system a classical washout algorithm will be used. Washout algorithms have been designed in the past using modular control components in Simulink [7]. Using SimCreator's built in control system components the washout algorithm shown in Figure 6 was developed.

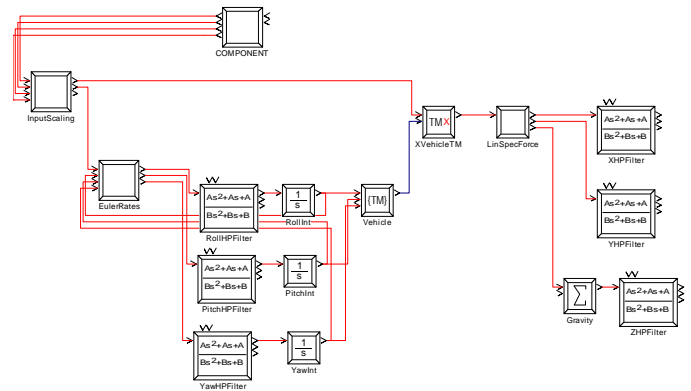


Figure 6: Classical Washout Algorithm

The simulator is now complete and can be used for operator in the loop analysis. SimCreator's graphical environment is used to compile the simulator, set up the vehicle dynamics' parameters and initial conditions, and run the simulation. In addition SimCreator allows for the saving and retrieval of multiple data sets. This allows for several similar vehicles to be parameterized for a single dynamics model. Any other simulation parameters such as filter settings for the washout algorithm can also be configured and saved using the SimCreator.

MODIFYING THE VEHICLE DYNAMICS

The ability to modify and add new subsystems to the vehicle dynamics is dependent on the original design of the components and their integration with each other. In this paper, the powertrain was developed using a set of components and therefore it is straightforward to modify.

For this example, the front wheel drive powertrain is converted into a four-wheel drive powertrain. First a TransferCase component must be designed. This will take the torque from the gearbox and distribute it between the front and rear axles. The TransferCase component also returns the average speed of the front and rear differential to the gearbox. It can be inserted into the powertrain model along with a rear differential as shown in Figure 7. At this point the simulator can be recompiled within the SimCreator environment. Since the complete simulator is captured in the environment, it is straightforward for the vehicle designer to modify the dynamics and compile and execute the complete simulation.

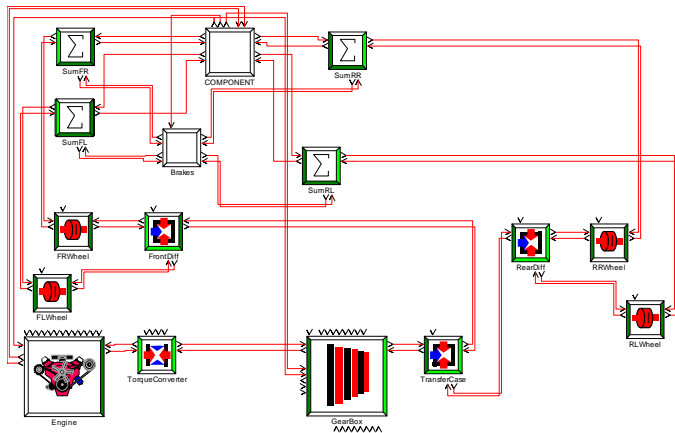


Figure 7: Four Wheel Drive Powertrain

DRIVING SIMULATOR PERFORMANCE

The driving simulator as shown in Figure 5 contains 88 states. They were integrated using a Runge Kutta second order method. Running the vehicle dynamics and motion cueing at 1000 Hz and the audio and visual systems at 60 Hz it was found that the entire simulation ran faster than realtime on a single processor Pentium III 450 MHz computer.

ADVANTAGES OF THE GRAPHICAL ENVIRONMENT

There are several advantages to using a graphical simulation tool. The simulation tool provides a structured framework to build C Code components in. An example of the SimCreator framework is shown in the Appendix for the Inertia component. The simulation tool executes the components in the model such that all data required for the current component has been calculated in previous components. An example would be that gearbox torque has been calculated before the TransferCase torque subcomponent is called. Graphical simulation tools provide a unified integration algorithm with the ability to globally set integration methods and integration time steps. It is typically easy to add states to the integrator from any component (this is also shown in the Appendix for the angular velocity of the inertia). Any component output can be easily collected as data during the simulation and plotted. Inputs to the model

and initial conditions of the states can be accessed from a single user interface. The graphical environment allows users greater insight into the model being developed. It also allows easy connections between components and provides for a hierarchical view of the model. In addition components and models are fully encapsulated and easy to share between users. This encourages component and model reuse. For example, the vehicle dynamics model could be compiled separately from the rest of the simulation and used in hardware in the loop testing. Some simulation tools, such as SimCreator, document the interface between components with descriptions and units (as shown in the Appendix). These descriptions and units can be used when connecting components and are also available when plotting. Typically simulation tools can provide realtime execution of the models using internal timers or external interrupts. Some tools also provide multiprocessor and distributed compilation and execution support.

CONCLUSION

It was found that a vehicle dynamics model and a driving simulator could be designed using modular components. The complete simulator met the requirements of realtime performance using a Pentium III 450 MHz computer. The simulation tool's graphical interface provided insight into the modular design, allowed for component reuse and a unified compilation environment. The design of the individual components and submodels are important to maximize component and model reuse. Modification of the vehicle dynamics to support a four-wheel drive vehicle was found to be straightforward. Using SimCreator's graphical environment, input data could be easily modified and the various simulation outputs could be plotted.

CONTACT

Richard Romano has been working in driving simulation for ten years focusing on motion cueing, vehicle dynamics, and human factors research. Mr. Romano was the manager of simulator research and development at the Iowa Driving Simulator and supervised the brake system simulation group at ITT Automotive. He is now president of Realtime Technologies, Inc. He may be contacted at raromano@ix.netcom.com.

REFERENCES

1. Romano, R.A., Stoner, J.W., and Evans, D.F., "Real Time Vehicle Dynamics Simulation: Enabling Tool for Fundamental Human Factors Research," SAE Technical Paper Series No. 910237, March 1 1991.
2. Artz, B.E., Cathey, L.W., Greenberg, J., Romano, R., and Cheney, S., "A Hardware in the Loop Real Time Driving Simulator for Advanced Brake System Development," Proceedings of the 1997 Driving Simulator Conference, Paris, France, September 1997.

3. Allen, R.W., and Rosenthal, T.J. "Requirements for Vehicle Dynamics Simulation Models," SAE Technical Paper Series No. 940175, February 28, 1994.
4. Connair, K.M., Bodie, M.O., and Chaumette, P., "Development of a Common Vehicle Model for Chassis Control Design," SAE Technical Paper Series No. 1999-01-0732, March 1, 1999.
5. Ciesla, C.R., Jennings, M.J., "A Modular Approach to Powertrain Modeling and Shift Quality Analysis," SAE Technical Paper Series No. 950419, February 27, 1995.
6. Zachary, J.R., Munns, S.A., and Moskwa, J.J., "The Development of Vehicular Powertrain System Modeling Methodologies: Philosophy and Implementation," SAE Technical Paper Series No. 971089, February, 1997.
7. Romano, R.A. "A New Approach to Motion Control Logic for Ground Vehicle Simulators," Proceedings of the IMAGE VII Conference, The IMAGE Society, Tucson, AZ, 1996.

APPENDIX

```

TYPE(CONTINUOUS)
NAME(Inertia)
SIDES(25,25, GREEN, NONE, GREEN, NONE)

BEGIN_DEFINITIONS
  /* States */
  DEFINE_STATE(Vel, "Angular Velocity", "rad/s", SIDE1, LOCATION1)

  /* SIDE 1 */
  DEFINE_INPUT(Torque1, "Torque", "Nm", SIDE1, LOCATION1)
  DEFINE_OUTPUT(Vel1, "Angular Velocity", "rad/s", SIDE1, LOCATION2)

  /* SIDE 2 */
  DEFINE_INPUT(Inertia, "Moment of Inertia", "kg-m-m", SIDE2, LOCATION1)

  /* SIDE 3 */
  DEFINE_INPUT(Torque3, "Torque", "Nm", SIDE3, LOCATION1)
  DEFINE_OUTPUT(Vel3, "Angular Velocity", "rad/s", SIDE3, LOCATION2)
END_DEFINITIONS

BEGIN_INIT
END_INIT

BEGIN_STOP
END_STOP

BEGIN_RATES
  RATE(Vel)=(INPUT(Torque1)+(INPUT(Torque3)))/INPUT(Inertia);
END_RATES

BEGIN_OUTPUTS
  OUTPUT(Vel1)=OUTPUT(Vel3)=STATE(Vel);
END_OUTPUTS

BEGIN_STATE_OVERRIDE
END_STATE_OVERRIDE

```