ACC stop & go simulation and visualization

ACC Stop & Go
Simulation and Visualization

2007.042

E. Reichardt
0554188

March 27, 2007

Eindhoven University of Technology
Department of Mechanical Engineering
Dynamics and Control Technology
Professor: Prof.dr.ir. M. Steinbuch
Supervisor: Dr.ir. M.J.G. van de Molengraft
Ir. G.J.L. Naus

Contents

1.Introduction

Nowadays the cruise control (CC) is a commonly confessed attribute in cars. Lots of people use the cruise control to make their trip more comfortable and relaxed. With the CC it is possible to choose a certain set speed which the vehicle has to maintain. A further development on the regular cruise control is the Adaptive Cruise Control (ACC). In an ACC, the CC is extended with a radar which makes it possible to overview the situation around the vehicle. The gathered information is used to decide if the speed of the host vehicle has to be adapted to the speed of a certain relevant predecessor. This ACC is already available in some luxury cars. An extension on this ACC is the ACC stop&go. As the name already implies this ACC is even capable of slowing the vehicle down to a full stop.

TNO Automotive in Helmond, the Netherlands, is developing ACC stop&go. Because there is a car available with the ACC stop&go on board it is possible to do real life experiments. In order to get useful results however, possibly dangerous situations have to be tried out which can lead to crashes when the controller malfunctions. Because this is not a feasible option, another way has to be found to get a good overview over the situation and the behavior of the controller. From simulations with computers, graphs of the behavior of the vehicles can be made. These graphs however are difficult to interpret and do not provide a clear overview over the situation when a significant number of cars is involved.

To get better insight into the behavior of the vehicles involved, a three-dimensional visualization of the simulation is desired. This leads to the objective of this project:

"Having a program which can execute a simulation to increase insight in how a truck, driven by an ACC, behaves in relevant situations."

The objective consists of a few different parts:

- Learning about ACC
- Building the simulation
- Creating a 3D visualization from the built simulation

This report consists of the different steps which are necessary to understand the basic working principle of ACC, to make a simulation and to build a 3D visualization. Chapter 2 is about the ACC; how it works, what it exists of, what kinds of ACC's exist and why they are used. In Chapter 3 is explained what parts the visualization consists of, which programs are used for it and why those programs are chosen. In Chapter 4, the link to a three-dimensional overview over the relevant situations is explained. Finally, conclusions and recommendations are drawn.

2.Adaptive Cruise Control

2.1 What is Adaptive Cruise Control?

Adaptive Cruise Control (ACC) is an extension to the better known standard cruise controls. With the standard cruise controls it is possible to choose a desired speed which the vehicle has to maintain. With an ACC the vehicle also has the possibility to adapt its speed to the speed of the predecessor.

ACC is not a completely new term; there already exist some modern vehicles with an adaptive cruise control built in, for example Lexus, Audi, BMW and Mercedes already have cars driving around with an ACC in it (1). Because the ACC still is a rather expensive option only the more luxury cars are available with this feature.

A distinction has to be made between different kinds of ACC. The first generation ACC's only had the possibility to adapt the speed of the vehicle by using a limited deceleration. Those are the ACC's which are already used. The second generation ACC's is capable of using the full braking power of the vehicle. This enables higher decelerations for the ACC-equipped vehicle. Until now an alarm is used which warns the driver if a higher then reachable deceleration is needed for the ACC to prevent a collision. With the second generation ACC this becomes unnecessary because the ACC can avoid the potential collision by itself, which can improve safety on the road. The next step is an ACC stop&go. As the name already implies, this ACC has to be able to slow the vehicle down till stoppage and then accelerate again. This can be useful in situations like a traffic light or a traffic jam.

2.2 Physical implementation

The ACC consists of different aspects ***(figure 2.1)***, which will be explained part by part to understand the working of it.

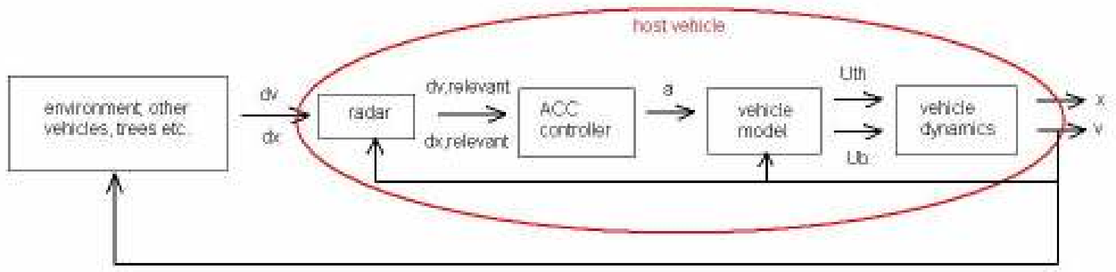


Figure 2.1: block schedule of the relevant parts of an ACC

2.2.1 Environment

The block environment consists of all the vehicles and objects surrounding the ACC host vehicle. All these objects have a certain relative distance and speed with respect to the host vehicle; dv and dx.

2.2.2 Radar

The radar is assembled to the host vehicle to observe the environment. The radar detects the objects around the vehicle and measures the relative distances. From the changes in these relative distances the relative speed is calculated. The data collected by the radar are transmitted to the Adaptive Cruise Controller.

2.2.3 ACC Controller

The ACC controller part has the collected data from the radar as input. With these data the ACC decides which objects are relevant to the host vehicle and whether or not it is necessary to adapt the speed of the host vehicle to the speed of the target vehicle. The output of the ACC controller block is the desired acceleration of the vehicle.

2.2.4 Vehicle model

The calculated acceleration by the ACC is not usable for the vehicle, so it has to be transformed to a certain throttle or brake signal. This is what the vehicle model does, it tells the vehicle to increase or decrease the throttle or to hit the breaks.

2.2.5 Vehicle Dynamics

The vehicle has the desired throttle and brake signal as input, and with these signals the speed of the vehicle is adapted. The actual speed and position are used as outputs and the speed is used to link back to the vehicle model and to the environment. In the vehicle model the change in speed is compared to the desired change and adapted if necessary. Of course the change in speed of the host vehicle also means a change in the relative distances and speeds to the surrounding vehicles and objects. This is why in the figure the speed signal is also linked back to the environment block, even though this is not a real return of the data. The radar detects the change in relative speed and distance as a result of the change in speed.

2.3 Why is Adaptive Cruise Control interesting?

The reason why ACC overall is interesting is that it can increase the safety on the road. Over 90% of the road accidents are caused by drivers' mistakes (2). Automation of automotive processes is seen as a possible solution which can decrease this number. When together with the driver of the vehicle there is a system which actively pays attention to the environment of the vehicle, the possibility to avoid accidents increases. Moreover, a computer has a far shorter reaction time than a human being. This means that the controller is able to intervene faster and the chance on a collision decreases.

Also because the ACC practically controls the vehicle the driver only has to steer and has more time to pay attention to his environment. This increases the chance on detecting a possible dangerous situation and reacting correct on it.

However a controller like discussed above also can have some disadvantages. The extra time the driver has because the ACC controls the speed of the vehicle increases the chance the driver gets bored. When this happens the driver will be easily distracted. This means that the advantage ACC has by increasing the chance for the driver to detect possible dangerous situations, also can be turned into a disadvantage because the driver has his mind on something else than driving the vehicle.

3. Set-up of the simulation and visualization

To make a visualization of the behavior of an ACC stop&go controlled vehicle, the different parts of the block schedule discussed in the previous chapter have to be modeled. These parts together form the simulation program which describes the behavior of the different vehicles. From this simulation the visualization can be made. In this chapter the modeling of the different parts of the simulation is discussed.

3.1 Programs used for the simulation and the visualization

The program consists of five relevant parts; the environment, the radar, the ACC, the vehicle model and the vehicle dynamics. TNO has provided two of these five parts, namely the ACC and the vehicle model. These parts are designed in MATLAB S*imulink*, and have certain in- and outputs which cannot be changed. Consequently, the rest of the program has been built around these parts. Hence, the environment, the vehicle and the radar are designed using MATLAB *Simulink* as well. Moreover, in this program it is fairly easy to connect the different parts of the program to each other and to let them interact, which is ideal for the desired simulation.

Another toolbox which can be used for the building of the visualization is *SimMechanics*, which is a part of *Simulink*. This program has the same way of use as *Simulink*, it also looks the same, only it is specially directed to the modeling, simulating and analyzing of mechanical systems (3). This makes is suitable for the building of the necessary vehicles, as these models can be implemented in the main *Simulink* model.

For the design of the 3D world in which the simulation will be viewed MATLAB has a toolbox called *V-realm Builder 2.0*. With this toolbox it is possible to create a world with all kinds of different objects in it, which can be linked to a simulation built in *Simulink*. The toolbox has its own library with objects which can be added to the world, however it is also possible to export files from other programs. One of these programs is *Unigraphics*. This means that it is possible to create every needed object in *Unigraphics* and use these objects in *V-realm*. How this works exactly will be explained in the next chapter.

The last toolbox which is necessary for the visualization is the *Simulink Virtual Reality Toolbox*. This toolbox is the link between the simulation made in *Simulink* and *SimMechanics* and the virtual world in *V-realm*. With this toolbox it is possible to add a block to the *Simulink* model which converts the movement signal to a signal which actuates the objects used in *V-realm*. In *figure 3.1* an overview is given over the different used programs and how they are linked to each other.

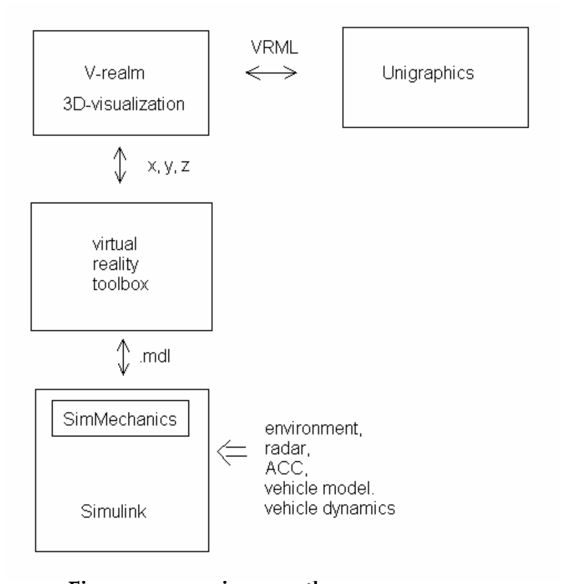


Figure 3.1: overview over the links between the programs

3.2 Vehicle, Environment and Radar

The parts which still have to be designed are the vehicle, the environment and the radar. Each of these parts will be discussed in the following.

3.2.1 The vehicle

For the design of the vehicle a truck is chosen. The objective of a model for the vehicle is that the vehicle dynamics may be limiting for the behavior of the vehicle. This means that it is possible that the ACC tells the vehicle to slow down with a certain deceleration while this is impossible for the vehicle because the brakes are relatively weak.

The main object of this project is creating a basic simulation which provides more insight in the behavior of the ACC stop&go. To get this insight for now it is not important how realistic the vehicle reacts on changes in speed, but if the ACC changes the speed when it is necessary.

In formula the model of the truck chosen for this simulation looks like

$$F = m \cdot a \tag{3.1}$$

The formula represents a single moving mass. This model is chosen because it is sufficient for this program. Moreover this model is easy to implement in the *Simulink* program, how it is modeled exactly can be seen in ***appendix B***. When the simulation is finished and it has to be developed further for a more realistic behavior of the truck, it is easy to replace because every part of the program has its own block in the simulation overview. This means that it is not necessary to change the complete program, only a single block has to be replaced. For the design of the trucks *SimMechanics* is used, and the final vehicle model is summarized in one *Simulink* block which makes it easy to connect the desired in- and outputs.

3.2.2 The environment

With the term ‘environment’, all objects surrounding the vehicle are meant. These objects vary from other trucks to trees and traffic signs along the road. In the simulation program only one other vehicle is included which the radar reacts on. For this vehicle the same model is used as for the vehicle with the ACC, because this is the simplest model. The only important thing of a target vehicle is that it can be seen by the radar, how it is modeled does not matter for the behavior of the ACC. Normally the radar detects all objects in its range and chooses the objects which may be dangerous for the host vehicle. In this case however it is not important if the radar functions or not, it can be assumed that the radar makes the correct choice so it is sufficient to include the model of only one other vehicle. It is possible to expand the environment block in the *Simulink* model and add more vehicles or objects.

3.2.3 The radar

To design a radar, the x- and y-position of both the host and the target have to be known. In this case the x-direction is the direction in front of the vehicle and the y-direction is the sideways direction *(figure 3.2)*.

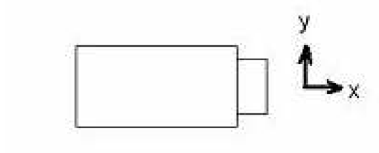


Figure 3.2: x- and y-directions (top view)

The positions of the target and the host can easily be measured in *Simulink* and then be subtracted from each other. This provides the relative distance between the vehicles in x- and y-direction.

The beam of the real radar is shaped like a triangle with a half round front end, and has a certain range. For the program the radar has to have the same shape and range. To make the modeling of the radar a bit easier the radar beam is simplified to a real triangle, without the half round front end. The radar used in this program has a range of 120 meters and an angle of 30 degrees[1] *(figure 3.3)*. This means that for the x-measurement the relative distance can be used and this number has to be compared to the range.

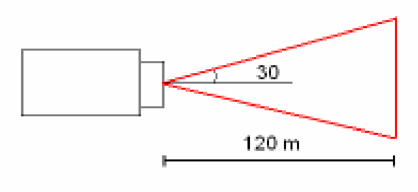


Figure 3.3: shape radar beam (top view)

For the y-measurement the relative y-position of the target is compared to the relative x-position of the target, multiplied by a certain gain. By multiplying the relative x-position with this gain the triangle shape of the radar beam can be made. The gain value is calculated on the following way:

$$y = 2 * [\tan^{-1}(30) \cdot 120] \qquad (3.2)$$

in which y is the range in y-direction of the radar beam at the maximum x-range. Now the gain is calculated by

$$g = \frac{\frac{1}{2}y}{120} \qquad (3.3)$$

[1] All relevant values in the model are defined in the MATLAB m-file called *params_compleet.m* ***(appendix B)*** so they are easy to change.

When both the x- and y-position of the target are in range of the radar it sends out a signal which activates the ACC. At that moment the ACC intervenes and controls the speed of the vehicle. All necessary parts of the radar are available in *Simulink*. How the radar looks like in *Simulink* can be seen in ***appendix A.***

3.3 Simulated situations

To learn more about the behavior of the ACC stop&go it is important that the controller is tested in different situations. To make a small start in this program five different situations are programmed and visualized:

1. A target with a constant speed which is lower then the speed of the host vehicle
2. A target which is slowing down till stoppage
3. A target which is slowing down till stoppage and then accelerates again
4. A target which is slowing down till a certain constant speed and then accelerates again
5. The host vehicle is being passed by another truck which slows down after the passing
6.

These situations are defined in the MATLAB M-file *params_compleet.m* ***(appendix B)***. This M-file can be extended with lots of other different situations.

3.4 Simulation results

It is possible to get results from the simulation built in *Simulink*. These results can be plotted in graphs, which look like *figure 3.4*. The two lines represent the speed of both the target and the host vehicle.

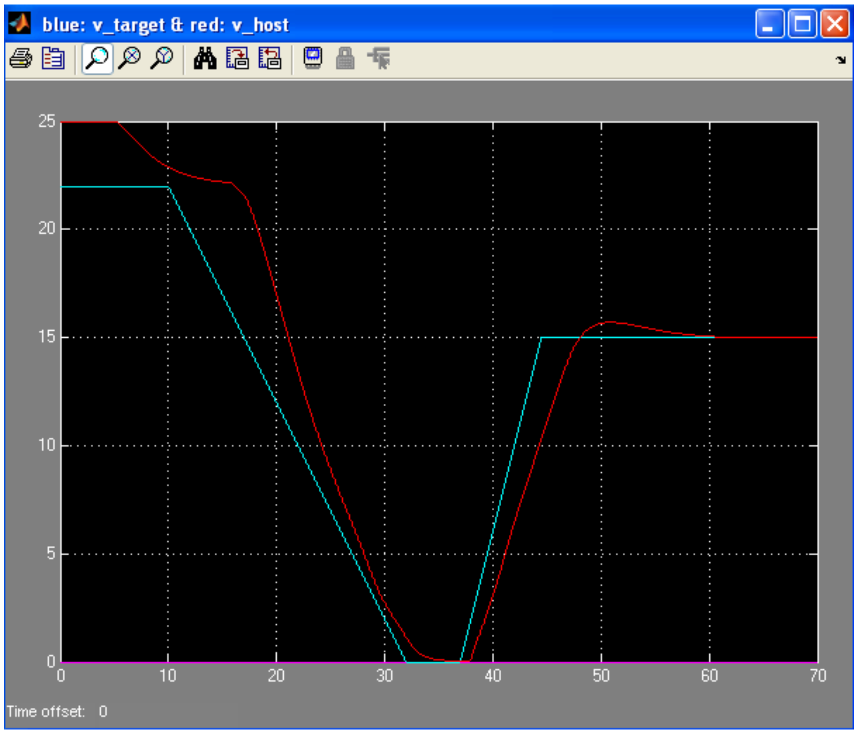


Figure 3.4: simulation results in a graph.
Red = host speed, Blue = target speed

The results in this graph are quite clear, but it is possible to imagine that when lots of different vehicles are involved the graph will quickly become a mess of different lines. Furthermore, the difference between two lines in this graph can be seen, but it is hard to imagine how big this difference in speed is in real life. This is why a 3D-visualization will give a better insight into the behavior of the vehicles.

4.3D Visualization in *V-realm*

In this chapter the necessary steps to build the desired three-dimensional world in *V-realm* are explained, as well as the results. The main goal of this visualization is to get more insight into the behavior of the vehicle.

4.1 Objectives of the visualization
Of course the most important is that when the visualization is opened the vehicles are showed, so these have to be designed. As mentioned before, these vehicles are designed in *Unigraphics*. From *Unigraphics* they are exported as VRML files to *V-realm*. When such a part is opened in this program and added to the object library, it can be added to every desired environment.

When a new world is created in *V-realm*, it starts with a completely black screen. This means that every single part has to be added, varying from the background to the trees along the road. It is important to create a world with not only vehicles but also objects surrounding the vehicles because else the movement of the vehicles cannot be seen.

The parts the final world consists of are:

- A background
- The road the vehicles drive on
- The vehicles
- A traffic light
- Buildings along the road
- Trees along the road
- Street lights along the road

Except from the traffic light and the vehicles all these objects are available in the object library, they only have to be scaled to the desired size. The top of the traffic light is also available in the library, only the pole which it stands on has to be inserted. Since a cylinder is one of the basic design forms in the program the pole is also easy to design.

Furthermore it is important that the visualized situation can be viewed clear. In *V-realm* it is necessary to define certain viewpoints from which the visualization will be sighted, so it is important to define these viewpoints precisely. The relevant viewpoints for this visualization are a top view to have an overview over the complete simulation, a view from the host truck so it is possible to see what the driver experiences, and a three-dimensional view over the whole situation.

4.2 Difficulties during the building of the visualization
The largest part of the designing of the three-dimensional environment is pretty easy to find out by reading the manual of the program. However, the users manual is not very clear on some aspects which makes them interesting to discuss here. The aspects which will be dealt with are the importing of objects from *Unigraphics* and the creating of a viewpoint.

4.2.1 Importing objects from Unigraphics
What is important to keep in mind during the designing process of a desired object in *Unigraphics* is that the x-, y- and z-directions are different from the used directions in *V-realm*. The coordinate system in *V-realm* is rotated 90 degrees around its x-axis in comparison to *Unigraphics*. This can be solved by rotating the coordinate system in *Unigraphics* 90 degrees around its x-axis before the modeling is started. When this is done the y-axis directs upwards and the z-axis directs in the negative standard y-direction. In *figure 4.1* it can be seen how the coordinate system should be orientated. This is also a picture of the used truck in the visualization. The complete manual to import a figure from *V-realm* into *Unigraphics* can be found in ***appendix C***.

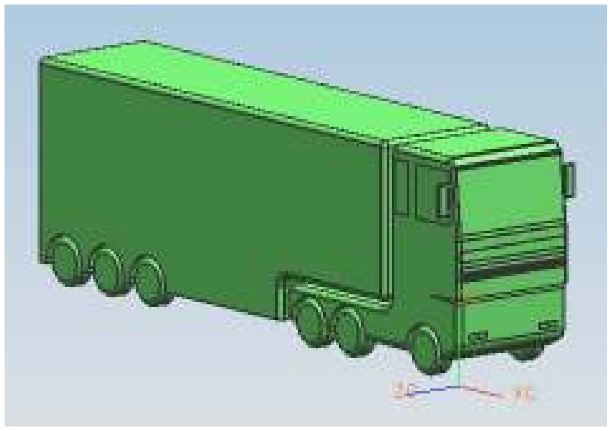
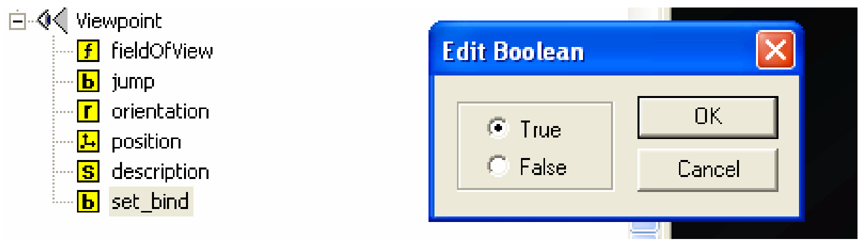


Figure 4.1: 3D model of the truck with directions of coordinate axes

4.2.2 Creating a viewpoint
The hardest part of the whole visualization probably is the defining of the desired viewpoints. It is not hard to insert a viewpoint, but it is very difficult to connect the viewpoint to the right part and let it face into the right direction. A manual to add a viewpoint is enclosed in ***appendix D***.

The best way to approach the directing of the viewpoint is by first setting the set_bind to *True (figure 4.2)*. By doing this the view in the *V-realm builder* becomes the viewpoint which is being created. Moreover by doing this the viewpoint is attached to the vehicle which means that it will follow the movement of the vehicle during the visualization.



Viewpoint
- fieldOfView
- jump
- orientation
- position
- description
- set_bind

Edit Boolean

[✓ True] False

Figure 4.2: setting set_bind on True

Now both the position and the orientation parameters should all be set to zero. The overview of the world created so far changes automatically when the parameters are changed. By paying attention to the changes in view during the adapting of the orientation and position parameters it is possible to find out roughly how the viewpoint is orientated. Now by adapting the parameters one by one, the desired viewpoint can be created, mostly by trial and error. When the viewpoint meets the requirements it is possible to give it a certain name. During the simulation, in the visualization screen the different created viewpoints are listed. It is possible to choose the desired viewpoint for the visualization and to change between the viewpoints during the visualization.

4.3 Visualization results

For the visualization in this case the world consists of the objects listed in the first paragraph of this chapter. For the vehicles two different designs are made; one truck with a trailer and a truck without one. This creates the possibility to visualize a loaded or an unloaded truck, depending on the desired situation. Furthermore four different viewpoints are created; a viewpoint from behind the truck, a top view and two three-dimensional overviews over the situation. The viewpoint from behind the truck is chosen because this viewpoint gives the overview over the situation close to how the driver would experience the situations. The top view provides an overview which makes it possible to get the best insight into the behavior of the vehicles. Even the smallest movements of the vehicles relative to each other can be seen. Finally the three dimensional overviews are created because these give a nice overview over the world and the vehicles, which can be useful for presentations. In the figures below an example of every viewpoint is depicted for the truck without trailer. The situation with the truck with trailer looks exactly the same only the vehicle looks like *figure 4.1*.

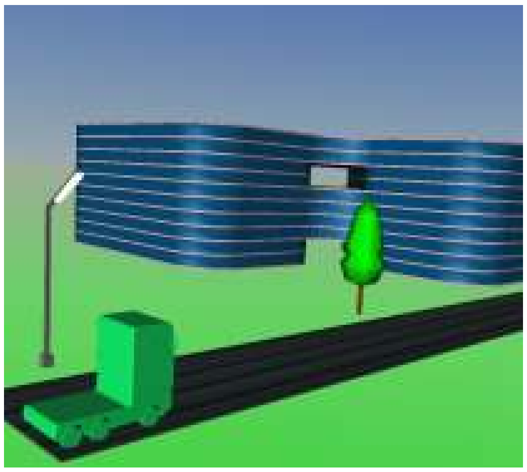


Figure 4.3: first of two 3D viewpoints

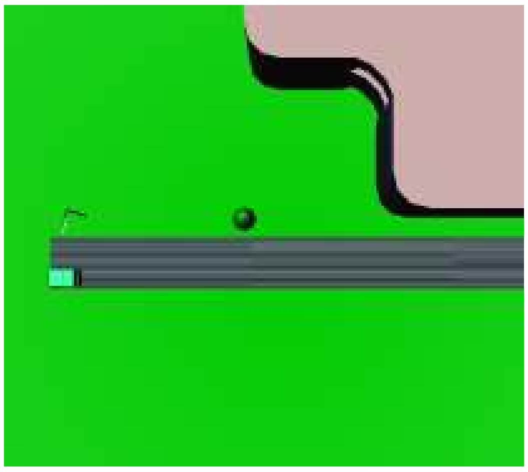


Figure 4.4: top view

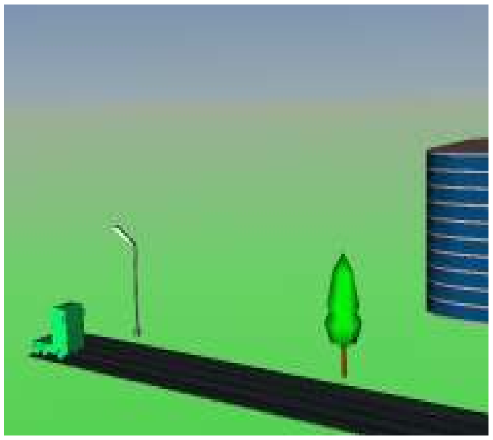


Figure 4.5: second of two 3D viewpoints

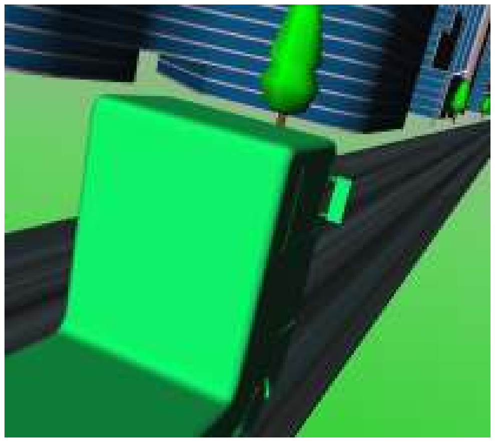


Figure 4.6: viewpoint from behind the truck

The necessary files to run the visualizations can be found on the CD which is included with this report. How to install these files is explained in the readme file on this CD, which can also be found in ***appendix E***.

5.Conclusion and Recommendations

To get better insight into the behavior of an ACC stop&go controlled vehicle a simulation is made. This simulation imitates situations which can occur on the road. By making a visualization of this simulation it is possible to view these situations in a three dimensional environment.

The simulation is built in MATLAB *Simulink*. For the model of the vehicle a single mass is used. This can be implemented easily into the simulation with *SimMechanics*. To make the link to a three dimensional visualization the *Simulink Virtual Reality Toolbox* is used. This toolbox is the link between *Simulink* and the three dimensional world, built in *V-realm Builder 2.0*, also a toolbox from MATLAB.

The simulation is built in such a way that it is easy to change the different parts of it. This is done because this program is a basic program, a foundation for further developments.

The visualization consists of the vehicles, an environment in which the vehicles are driving and different viewpoints. Different viewpoints are implemented and it is possible to change between them during the visualization for the best overview over the situation. Finally two worlds are created, one world for the situation when a truck is driving without a trailer, in the other world the vehicle consists of a truck with trailer.

Recommendations
Because this program is a first start for a 3D visualization, an overview of some first improvements that may be added to the program is given next:

- A more realistic vehicle model, to come closer to the real behavior of the vehicle
- A road with turns and slopes in it, to create more challenging situations for the ACC
- More vehicles surrounding the host vehicle, so the ACC has to choose the most threatening vehicle.
- Gauges in the visualization screen, for a better overview over the simulation parameters such as speed and distance
- An extended model of the radar, which takes more relevant parameters of the objects surrounding the host vehicle into account.

Literature List

1. Autonomous cruise control system. (2007, March 15). In *Wikipedia, The Free Encyclopedia*. Retrieved 11:58, March 20, 2007, from http://en.wikipedia.org/w/index.php?title=Autonomous_cruise_control_system&oldid=115273375
2. Integrated Automotive Control, ir. G.J.L. Naus, 25-10-2006
3. SimMechanics 2.6. In *The Mathworks, Accelerating the pace of engineering and science*. Retrieved 10:34, February 21, 2007, from http://www.mathworks.com/products/simmechanics/

Appendix A: Relevant models in *Simulink*

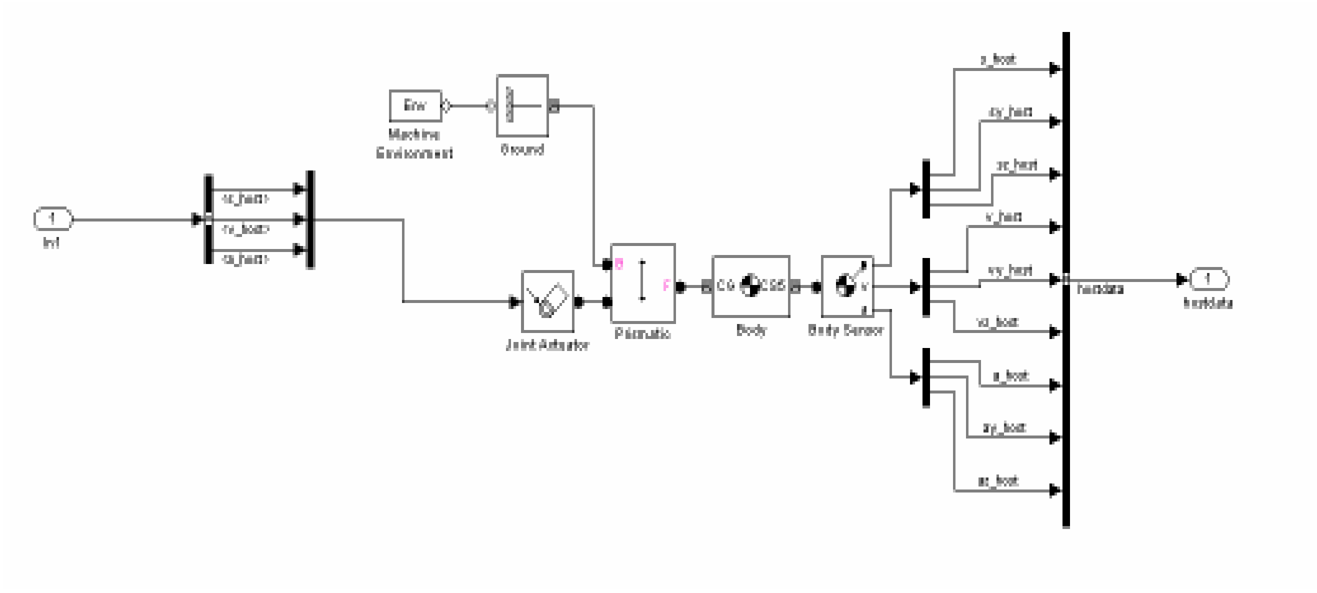


Figure B.1: model of a vehicle with actuators and sensors

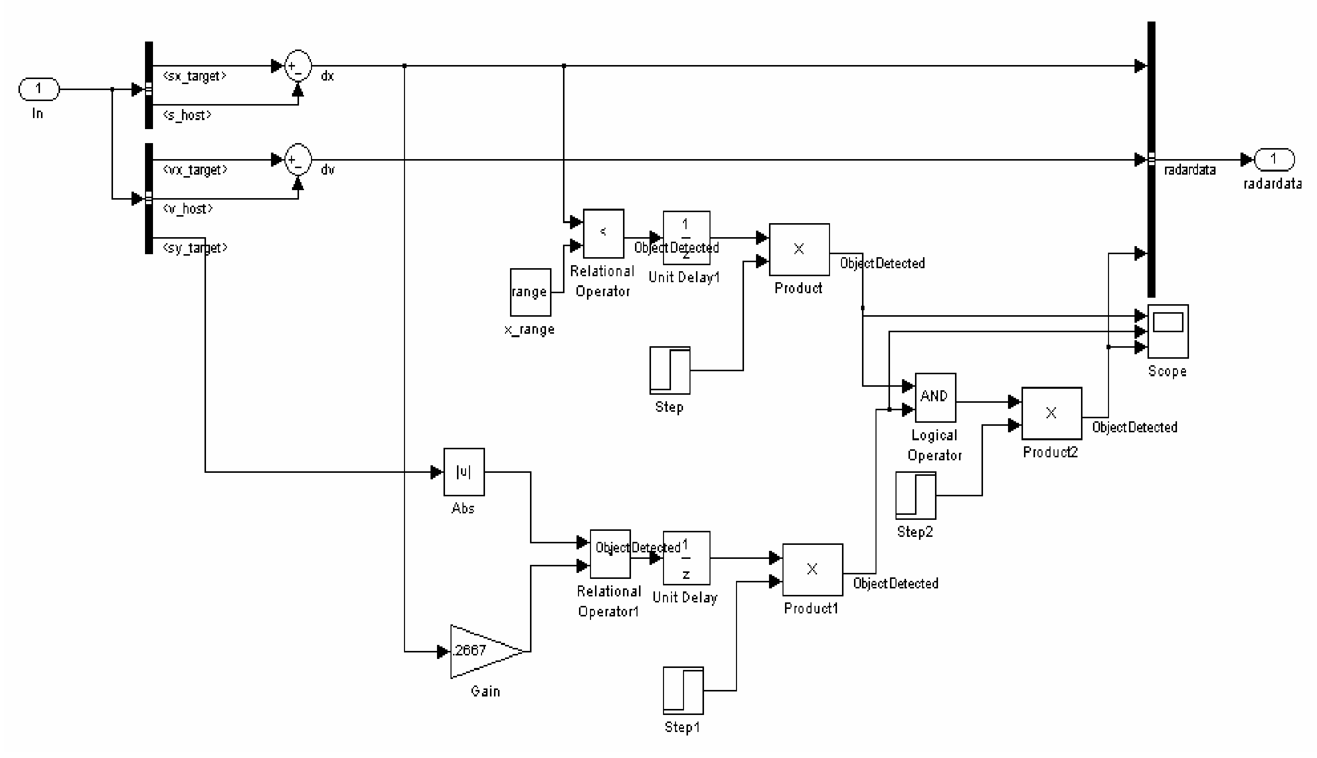


Figure B.2: model of the radar; x- and y-measurement

Appendix B: M-file Params_ compleet

```
simulation_time = 70;
%simulation time
ts = 1/1000;
%sample time

%radardata
range = 100;
%range radar [m]
angle = 30;
%angle of radar

%calculation y-range radar
width = tan((angle/2)*((2*pi)/360))*range;
y_gain = width/range;
%----------------------------------------------------------------------
------

%situation

situation = 3;

switch situation,
    case 1,
        %targetdata
        %target with constant speed
        x0_target = 200;
%initial x_position target [m]
        vx0_target = 15;
%initial speed target in x direction [m/s]

        acceleration = 0;
%acceleration [m/s2]
        start_brake = 0;
%starting time braking [s]
        stop_brake = 0;
%stop braking time [s]

        vx1_target = 0;
%speed target accelerates to [m/s]
        start_acceleration = 0
%starting time accelerating [s]
        acceleration_1 = 0;
%acceleration [m/s2]
        stop_acceleration = 0;
%stopping time acceleration [s]

        sy0_target = 0;
%y_movenment target

        start_pass = 0;
        stop_pass = 0;

        start_back = 0;
        stop_back = 0;

        %hostdata
        s0_host = 0;
%initial position host [m]
        v0_host = 25;
%initial speed host [m/s]
        MinLeverSpeed = 5;
%minimal lever speed [m/s]
        MaxLeverSpeed = 22;
%maximal lever speed [m/s]

    case 2,
        %targetdata
        %target slowing down till stoppage
        x0_target = 200;
%initial x_position target [m]
        vx0_target = 22;
%initial speed target in x direction [m/s]

        acceleration = -3;
%acceleration [m/s2]
        start_brake = 10;
%starting time braking [s]
        stop_brake = start_brake+(vx0_target/abs(acceleration));
%stop braking time [s]

        vx1_target = 0;
%speed target accelerates to [m/s]
        start_acceleration = 0;
%acceleration [m/s2]
        acceleration_1 = 0;
%acceleration [m/s2]
        stop_acceleration = 0;
%stopping time acceleration [s]

        sy0_target = 0;
%y_movement target

        start_pass = 0;
        stop_pass = 0;

        start_back = 0;
        stop_back = 0;

        %hostdata
        s0_host = 0;
%initial position host [m]
        v0_host = 25;
%initial speed host [m/s]
        MinLeverSpeed = 5;
%minimal lever speed [m/s]
        MaxLeverSpeed = 22;
%maximal lever speed [m/s]

    case 3,
        %targetdata
        %target slowing down till stoppage and accelerating again
        x0_target = 100;
%initial x_position target [m]
        vx0_target = 22;
%initial speed target in x direction [m/s]
```

```
        start_brake = 10;
%starting time braking [s]
        acceleration = -1;
%acceleration [m/s2]
        stop_brake = start_brake+(vx0_target/abs(acceleration));
%stop braking time [s]

        vx1_target = 15;
%speed target accelerates to [m/s]
        start_acceleration = stop_brake + 5;
%starting time accelerating [s]
        acceleration_1 = 2;
%acceleration [m/s2]
        stop_acceleration = start_acceleration +
(vx1_target/abs(acceleration_1));     %stopping time acceleration [s]

        sy0_target = 0;
%y_movement target

        start_pass = 0;
        stop_pass = 0;

        start_back = 0;
        stop_back = 0;

        %hostdata
        s0_host = 0;
%initial position host [m]
        v0_host = 25;
%initial speed host [m/s]
        MinLeverSpeed = 5;
%minimal lever speed [m/s]
        MaxLeverSpeed = 22;
%maximal lever speed [m/s]

    case 4,
        %targetdata
        %target slowing down and accelerating till speed higher then
lever
        %speed
        x0_target = 100;
%initial x_position target [m]
        vx0_target = 22;
%initial speed target in x direction [m/s]
        vx1_target = 5;
%speed target slows down to [m/s]

        start_brake = 10;
%starting time braking [s]
        acceleration = -2;
%acceleration [m/s2]
        stop_brake = start_brake+((vx0_target-
vx1_target)/abs(acceleration));   %stop braking time [s]

        vx1_target = 5;
%speed target slows down to [m/s]
        vx2_target = 30;
%speed target accelerates to [m/s]
```

```
        start_acceleration = stop_brake + 15;
%starting time accelerating [s]
        acceleration_1 = 2;
%acceleration [m/s2]
        stop_acceleration = start_acceleration + ((vx2_target-
vx1_target)/abs(acceleration_1));     %stopping time acceleration [s]

        sy0_target = 0;
%y_movement target

        start_pass = 0;
        stop_pass = 0;

        start_back = 0;
        stop_back = 0;

        %hostdata
        s0_host = 0;
%initial position host [m]
        v0_host = 25;
%initial speed host [m/s]
        MinLeverSpeed = 5;
%minimal lever speed [m/s]
        MaxLeverSpeed = 22;
%maximal lever speed [m/s]

    case 5
        %targetdata
        %host being passed by other truck
        x0_target = 0;
%initial x_position target [m]
        vx0_target = 22;
%initial speed target in x direction [m/s]

        start_brake = 30;
%starting time braking [s]
        acceleration = -1;
%acceleration [m/s2]
        stop_brake = 40;
%stop braking time [s]

        vx1_target = 0;
%speed target accelerates to [m/s]
        vx2_target = 0;
        start_acceleration = 0;
%starting time accelerating [s]
        acceleration_1 = 0;
%acceleration [m/s2]
        stop_acceleration = 0;
%stopping time acceleration [s]

        sy0_target = -1;
%y_movement target

        start_pass = 14;
        stop_pass = 18;

        start_back = 27;
        stop_back = 31;
        %hostdata
        s0_host = 50;
%initial position host [m]
        v0_host = 20;
%initial speed host [m/s]
        MinLeverSpeed = 5;
%minimal lever speed [m/s]
        MaxLeverSpeed = 22;
%maximal lever speed [m/s]
end
```

Appendix C: Exporting objects from *Unigraphics* to *V-realm*

The exporting of a model from Unigraphics to a VRML is quite simple. In the menu line under the head file there is a head export. Under this head the different ways to export the model are shown, in this case VRML is chosen. After clicking on this head, a menu opens in which some parameters can be established. The only relevant parameter in this menu is the location where the VRML will be saved to. In *figure C.1* the described menu in *Unigraphics* is showed.

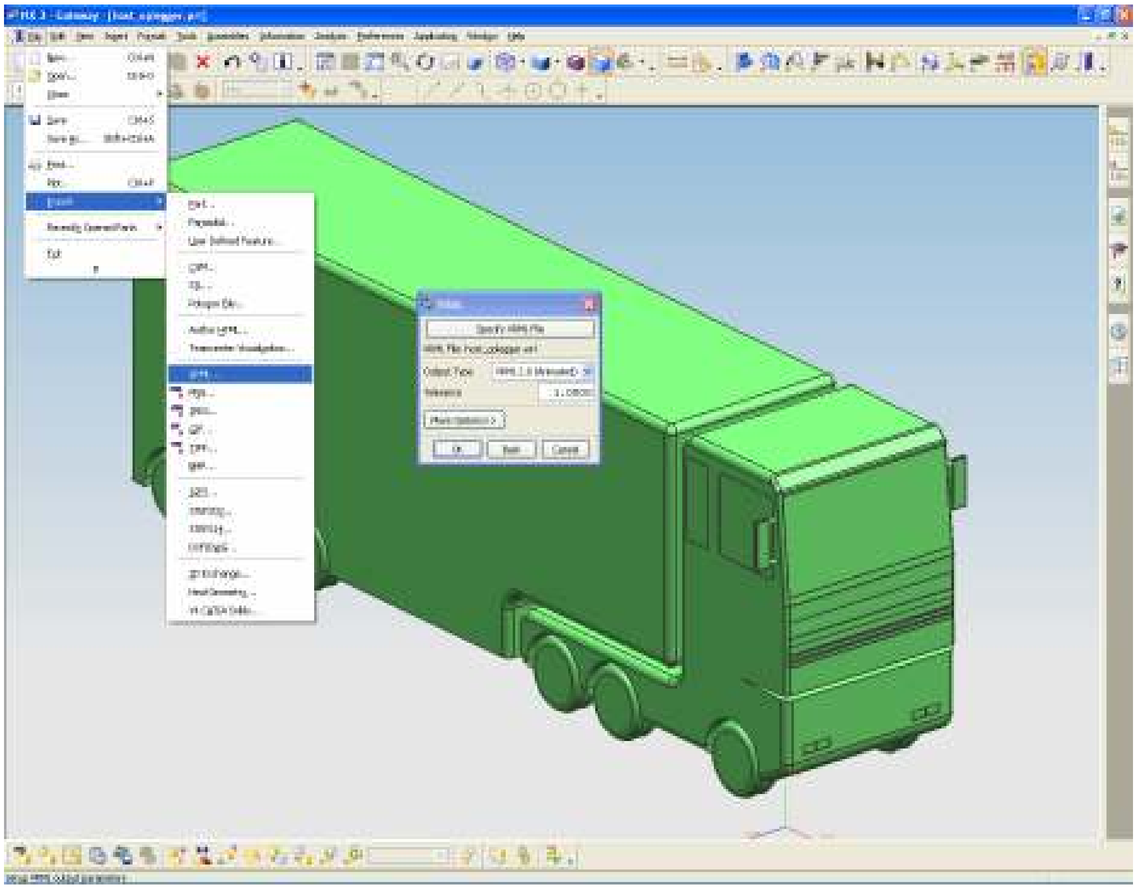


Figure C.1: VRML parameters menu

As mentioned in chapter 4, it is important to keep in mind that the coordinate system in *V-realm* is rotated 90 degrees about its x-axis in comparison to *Unigraphics*. This can be solved by rotating the coordinate system in *Unigraphics* 90 degreed around this axis, so the Y-axis directs upwards and the z-axis directs into the negative original y-axis. Now the vehicle will be orientated in *Vrealm* in the same direction as it is orientated in *Unigraphics*.

To import the object into *V-realm* the saved VRML only has to be opened in *V-realm*. Now the rest of the world van be created around the object, but it is better to add the object to the object library. When this is done the object can be added more times into the same world or other worlds. Adding the vehicle to the object library is done by right clicking on the description of the object in the overview of the world *(figure C.2)*, clicking on the ‘add to library’ menu and then by choosing the ‘object’ menu.

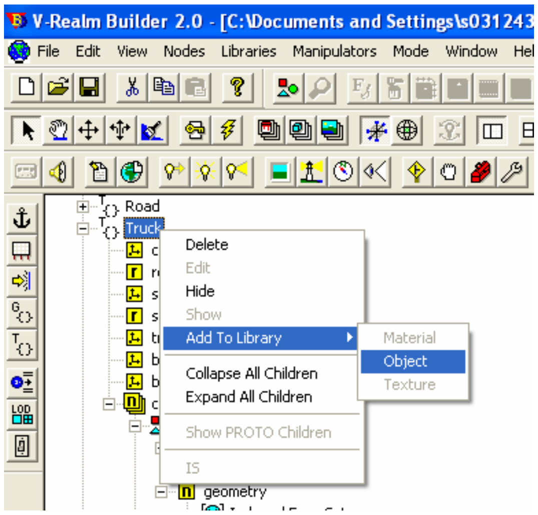


Figure C.2: Adding an object to the object library

Now the description, the name of the object and the category which the object has to be placed in can be chosen. When this is done, the object can be used in every desired world as many times as necessary.

Appendix D: Adding a viewpoint

To get the best overview over the visualized situations it is important that the viewpoints are well defined. Viewpoints can be added to every part of every object present in the designed world. A viewpoint is added to an object with the Access/Edit Viewpoint button in the menu *(figure D.1)*.

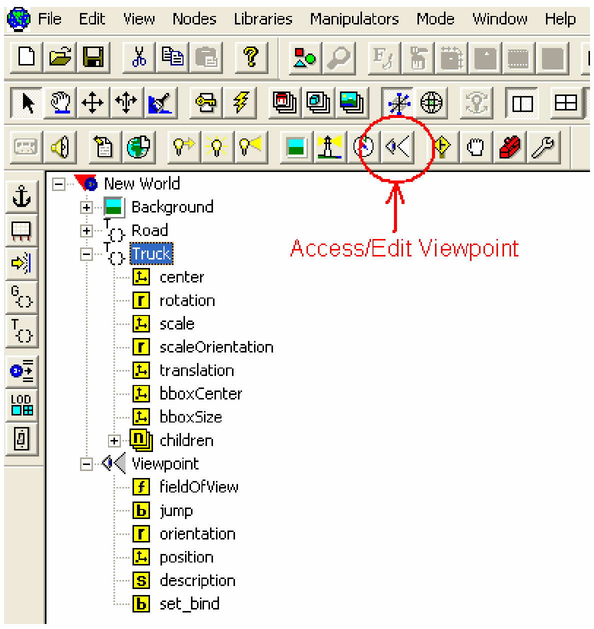


Figure D.1: inserting a viewpoint

The viewpoint can be added to the main object, but also to all the different parts the object consists of. These parts are collected in the 'children' head. With this option it is possible to install the viewpoints everywhere around the object.

When the viewpoint is added it still has to be orientated on the desired way, because *V-realm* itself orientates the viewpoint in a random way. The best way to handle the orientation of the viewpoint is by first setting the 'set_bind' to true and then setting all the position and orientation parameters to zero, as explained in chapter 4, under the head *Creating a viewpoint*. Now by adapting the values one by one the place of the viewpoint as well as the desired direction of it can be set.

Appendix E: Readme file

Readme for the installation of the ACC Stop&Go simulation:

To be able to run the simulation and visualization MATLAB R2006a is needed with the following toolboxes installed:

- SimMechanics
- Virtual Reality Toolbox

Next to these toolboxes the following files have to be installed in the MATLAB workspace, all in the same folder:

- compleet.mdl
- params_compleet.m
- TNO_ACC_SGmodule_sf.dll
- final_world.wrl
- final_world_oplegger.WRL

When this is done, the visualization can be started by clicking on the play button in the menu bar.
In the block model_outputs two blocks can be found named final_world and final_world_with_trailer. It is possible to choose beween these worlds depending on if it is desired to see a truck with or without a trialer.

In the M-file 5 different situations are programmed. A choice between these situations can be made by defining the situation in line 15.