A Program to Generate 3D Animated Graphics of Car Accidents from an XML Template

David Williams
d98dw@efd.lth.se

2004-07-23

Summary

Carsim is a text-to-scene converter. A text-to-scene converter is a program, which reads a text and, by analyzing it, generates an image or 3D scene representing the objects and/or actions described in the text. Carsim is designed to do just this, and more specifically to do it for traffic accident reports. The program itself can be divided into two separate modules, one linguistic module, which analyzes the text and one graphical module, which generates the 3D scene.

Carsim has been expanded several times since the project started to incorporate more languages and a wider range of reports. Therefore, the intermediate representation of data has changed and thus a new graphical module was needed. The goal of this master thesis was to produce this new graphical module.

Contents

Chapter 1

Introduction

1.1 Carsim

Carsim is a text-to-scene converter. A text-to-scene converter is a program, which reads a text and, by analyzing it, generates an image or 3D scene representing the objects and/or actions described in the text. Carsim is designed to do just this, and more specifically to do it for traffic accident reports.

The program itself can be divided into two separate modules, one linguistic module, which analyzes the text and one graphical module, which generates the 3D scene. First, the language processing module extracts information from the text that describe the roads, the vehicles, and the actions. It produces an intermediate representation that serves as input to the visualizer. In a second step, the visualizer synthesizes a 3D symbolic world where it recreates the accident.

1.2 My Task

Carsim has been expanded several times since the project started to incorporate more languages and a wider range of reports. Therefore, the intermediate representation of data has changed and thus a new graphical module was needed.

My task and the goal of this master thesis were to develop a graphical module to generate a 3D scene from the new XML template. It was also to take into account the suggestions made for the old graphical module and try to synthesize as much of the new template as possible.

1.3 Text-to-Scene Conversion

Text-to-Scene conversion is a process in which you take a written text and compose an image or animation from the information in the text. The text is analyzed and any important information is extracted. Important information are words answering questions such as what objects are in the scene, where the objects are, what movements these objects do, etc.

Apart from Carsim, another promising Text-to-Scene conversion program is WordsEye.

1.3.1 WordsEye

WordsEye is a text-to-scene converter, which creates scenes from invented texts. It was first developed by Bob Coyne and Richard Sproat at AT&T Labs, and is currently under development by Semantic Light LLC, a company started by Coyne and Sproat to try to apply the concept commercially. The program has a very fitting slogan: "Humans visualize language, why can't computers?".

On their web page, they say that "It [WordsEye] is not intended to completely replace more traditional 3D software tools, but rather to augment them by, for example, allowing one to quickly set up a scene to be later refined by other methods". WordsEye has a database of several thousand objects and more objects are added regularly. [Coyne and Sproat, 2001].

Chapter 2

Carsim

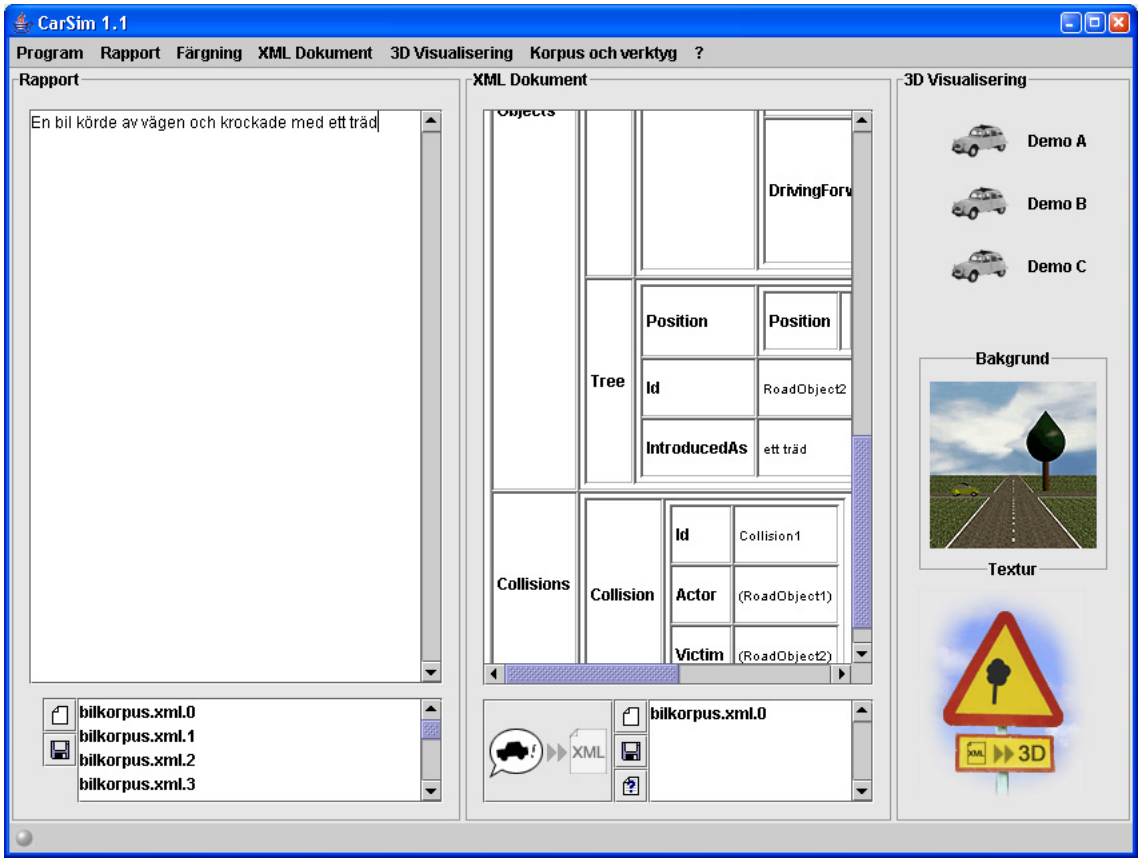


Figure 2.1: The Carsim program.

Carsim is a text-to-scene converter. It analyzes texts describing car accidents and composes 3D-animated scenes visualizing the accident. The program itself consists of two parts, an information extraction module and a vizualiza-

tion module. The information extraction module analyzes the text and stores the important information in a template, which the visualizer then can use to generate a 3D-animated scene of the accident.

The first version of Carsim was first developed in 2000 for car accidents written in French [Dupuy et al., 2001, Egges et al., 2001], then later extended to car accidents in English as well [Åkerberg et al., 2003]. A Swedish linguistic module was started in 2003. Upon getting suggestions for improvements and adjustments, it was clear that the old template was not extensive enough, so a new template was designed. With this change, a new visualization module was needed.

2.1 Knowledge Representation

2.1.1 The Old Template

The old version of Carsim was designed when the collection of texts was very limited. Thus, only a small number of objects was included in the template. Furthermore, when the old version was presented to a reference group a number of errors/flaws were pointed out.

- More objects on the scene should be visualized, like traffic lights, signs, etc.
- When a tree is the impacted object, it should not be shown alone in the scene. It is more likely that there are more trees close, which may have had some impact on the cause of the accident.
- Edge lines on the road are not desirable unless mentioned in the text. They can provide false meaning. This also extends to the color of the ground and sky. These should be neutral.
- There ought to be some way of showing the conditions at the scene, like rain, snow, fog, if the road was slippery or wet, etc.
- An easy way to modify the viewpoint, like setting it at the driver's seat or following the narrator's car.
- Information about speed, when mentioned. Both speed limit and current speed of the vehicles are interesting.
- etc...

Another limitation of the old template is that there only are a limited number of different scenarios available.

Scenario I	Zero or more DrivingForward or Stop TurnLeft possibly LeaveRoadLeft or LeaveRoadRight possibly OverturnLeft or OverturnRight Zero or more DrivingForward or Stop
Scenario II	Zero or more DrivingForward or Stop TurnRight possibly LeaveRoadLeft or LeaveRoadRight possibly OverturnLeft or OverturnRight Zero or more DrivingForward or Stop
Scenario III	Zero or more DrivingForward or Stop Depass possibly LeaveRoadLeft or LeaveRoadRight possibly OverturnLeft or OverturnRight Zero or more DrivingForward or Stop
Scenario IV	Zero or more DrivingForward or Stop ChangeLaneLeft possibly LeaveRoadLeft or LeaveRoadRight possibly OverturnLeft or OverturnRight Zero or more DrivingForward or Stop
Scenario V	Zero or more DrivingForward or Stop ChangeLaneRight possibly LeaveRoadLeft or LeaveRoadRight possibly OverturnLeft or OverturnRight Zero or more DrivingForward or Stop
Scenario VI	One or more DrivingForward or Stop possibly LeaveRoadLeft or LeaveRoadRight possibly OverturnLeft or OverturnRight

2.1.2 The New Template

A new template was developed, which took into account many of the suggestions offered by the reference group. The new template now gives more information about the scene and there are no limits to the event chain.

Here is a comparison of the old and the new template:

	Old template	New template
Static objects		
	RoadSign StopSign TrafficLight Tree	Building Ditch Embankment FallenTree Fence Hedge PedestrianCrossing Pit Pole Pool RoadBump Rock StopSign StreetLight TrafficIsland TrafficLight Tree Wall
Dynamic objects		
	FrenchDuck SimpleCar	Bicycle Bus Car Caravan Deer Elk LevelCrossing LiftableBridge Pedestrian Minibus Motorcycle Tractor TrailerVehicle Train Truck Van

	Old template	New template
Events		
	ChangeLaneLeft	Accelerate
	ChangeLaneRight	BackOff
	Depass	Bounce
	DrivingForward	CatchFire
	LeaveRoadLeft	ChangeLane
	LeaveRoadRight	Decelerate
	OverturnLeft	DrivingForward
	OverturnRight	DropLoad
	Stop	LeaveRoad
	TurnLeft	Override
	TurnRight	Overtake
		Overturn
		Rotate
		Separate
		Slide
		Stop
		Turn
		UTurn

Chapter 3

Three-dimensional Graphics

3.1 Drawing in Three Dimensions

Let's say you want to draw a triangle in three dimensions. A triangle is represented by three points, one for each corner. In 3D graphics, these points are called vertices. A vertex holds information about where it is located in the world (scene), it's world space coordinates. It can also hold more information, but more on that subject later. Position is usually expressed in a left-handed Cartesian coordinate system, where the X-axis is horizontal on the screen, the Y-axis is vertical on the screen and the Z-axis is into the screen.

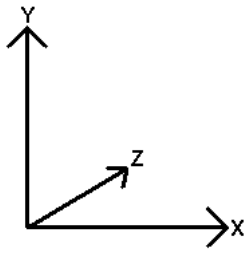


Figure 3.1: Left-handed Cartesian coordinate system.

When you draw the triangle on the screen, each coordinate is transformed into screen coordinates by applying any number of transformation matrices to its world space coordinates. These matrices can perform operations such as translation[1], rotation and scaling.

Vertices can also hold information about color. The color is then interpolated to give a smooth crossover if the other vertices on the triangle do not have the same color.

When you want to have more complex objects in your scene you must build these from triangles, as triangles are the simplest non-line objects. A cube, for instance, is made up of six squares, each square being composed of two triangles. The other common primitives, as they are called, are sphere, cylinder and torus.

[1] Moving an object

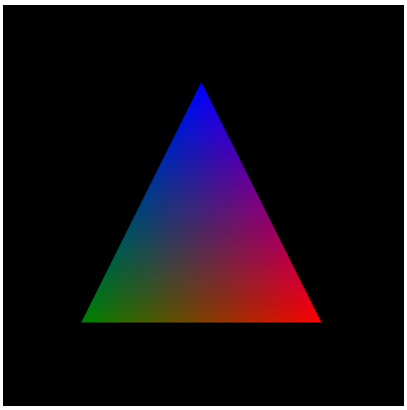


Figure 3.2: Colored triangle.

3.1.1 Z-buffer

If you are rendering[2] a scene with two objects, one in front of the other, you often only want the object closest to the camera to be visible (and, of course, the parts of the distant object that aren't covered by the nearest object). Now, if you draw the distant object first and then the front object second, you will get the desired result. However, if you draw the front object first and then the distant object you will draw over the front object unless you do some sort of check to see which object is closest to the camera. For this, we have the z-buffer, also called depth buffer.

The z-buffer is usually represented by a matrix, which has the size of the screen or the current window. As each pixel of an object is drawn, the rendering process checks the z-buffer to see what depth the last pixel drawn at this position had. If the current pixel has a z-value closer to the camera, it puts this value in the z-buffer and draws the pixel. If the value in the buffer is closer to the camera, the pixel is considered to be behind another object and is not drawn.

3.1.2 Lighting

To give the scene the appearance of three dimensions, you need to provide some sense of depth to the scene. This is done by lighting the objects, thus creating shadows, which will allow us to interpret the relative position of objects. There are many different types of lighting. Here are the ones used in Java3D:

- Point light – This light shines in all direction from its position. It has an attenuation value to determine how far it shines. It can be used to model a light bulb.

- Spot light – This light has a position in the scene. It emits a cone of light in a specific direction. The cone is controlled by an attenuation value and an angle (which controls the width of the cone). This light can be used for flashlights.

[2]The process of drawing to the screen

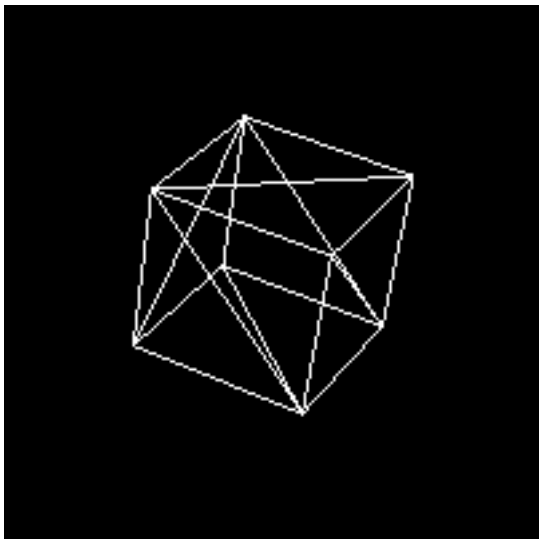


Figure 3.3: A cube is made up of 12 triangles.

- Directional light – This light is assumed to have originated so far away that all rays of light are parallel. It has no position in the scene, but only a direction. It does not have an attenuation value either. It can be used to emulate sun light.

- Ambient light – This light is a sort of background light. It is meant to provide the light that bounces off other objects. It lights everything equally.

Each light source is composed of two different components: diffuse and specular. The diffuse component can be described as the "normal" type of light. The specular component is used to create the highlighting effect on smooth objects, like light shining on an apple.

In order to be able to calculate the angle of the light shining on a surface, you need the normal vector for the surface. The angle then determines how much light is actually shining on the surface. If the angle is small, most of the light will shine on the surface. If the angle approaches 90 degrees, the surface will be more dimly lit. If the angle is above 90 degrees, no light will reach the surface and it will be black.

Calculating the normal vector takes several operations each time, and when the scene has thousands of surfaces the time taken is somewhat expensive in time units. Therefore, the easiest way is to assign a normal value to each vertex in the scene and use the three normals for each vertex in a triangle to calculate an approximate normal for that triangle, which can then be used to calculate the angle of the light. This method, however, gives a very blocky image, as the light can change abruptly from surface to surface even if the normal doesn't differ very much.

A slightly more complex shading method is *vertex shading*. This method still uses the three normal vector of the vertices, but for each point on the surface,

it interpolates a new normal. This gives a much smoother lightning and is often sufficient for a reasonably realistic scene.

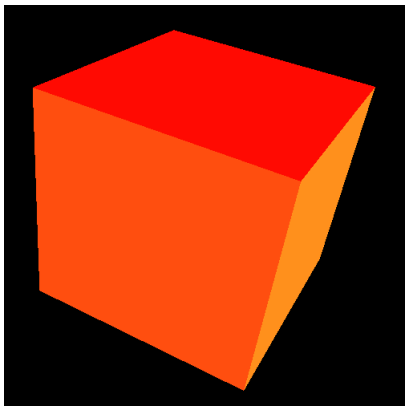


Figure 3.4: Red cube lighted by a yellow directional light coming from the right of the camera.

3.1.3 Camera

The camera is not an actual object in the rendered scene, but a representation of the eye of the observer. The camera has several parameters:

- Position – The position of the center of the camera lens in the scene.
- Orientation – Which direction is up for the camera.
- Direction – Where the camera is aimed at.
- Near Plane – The distance from the camera at which objects become visible.
- Far Plane – The distance from the camera at which objects no longer are visible.

3.1.4 Meshes

When you want to create a 3D-scene, you usually want to have fairly complex objects in it, like people, animals, trees, furniture, etc. Thus, all these objects need to be defined by a collection of vertices. This collection is called a *mesh*. The primitive objects mentioned before are also meshes.

3.1.5 Textures

Usually when creating objects, you don't want them to be just colored, but have the appearance of, say, a brick wall. This can of course be done by modeling each brick independently, but that would be very time consuming, both to produce each object manually and in the rendering process. Instead, you can just create

a large square (consisting of two or more triangles) and "glue" an image of a brick wall onto it. This is called *texturing* and the image is called a *texture*.

What you do is for each vertex on the square you specify a coordinate on the texture image. The renderer then interpolates the texture coordinates for the entire triangle and fetches the corresponding texture pixel, called *texel*. Of course, the texture may not be exactly the correct size so one texel may end up on several pixels on the triangle.

3.2 VRML

VRML, or Virtual Reality Markup Language as it stands for, is a markup language used to describe 3D scenes. It was developed as a web standard language but is used in off-line application as well. Recently, the successor to VRML, X3D has been developed.

3.3 3D Graphics In Java

Java does not have standard support for three-dimensional graphics, but an additional API[3] called Java3D[4] has been developed by Sun, which adds 3D graphics capabilities to Java.

Java3D does not directly communicate with the hardware, it is more of a scene creator. This means that Java3D has to work through another, more hardware-near 3D-API. There are currently two different implementations of Java3D, one which uses OpenGL[5] and another one which uses DirectX[6]. The DirectX version is currently limited to Windows-based computers, while OpenGL is available on many different platforms. The DirectX version is slightly faster than OpenGL on Windows computers.

[3]Application Programming Interface, a library of classes/interfaces
[4]http://java.sun.com/products/java-media/3D/
[5]http://www.opengl.org
[6]http://www.microsoft.com/windows/directx

Chapter 4

Graphics

4.1 Objects

The new template includes many new objects, and these needed graphical representations. The old version of Carsim used VRML description for some of its graphical objects, so the new objects could reuse some of the old code if these were also described in VRML. The program used for creating these objects was Caligari trueSpace 3.2, an older, freeware version of a professional 3D tool.

4.1.1 Dynamic objects

Figure 4.1: Bicycle.

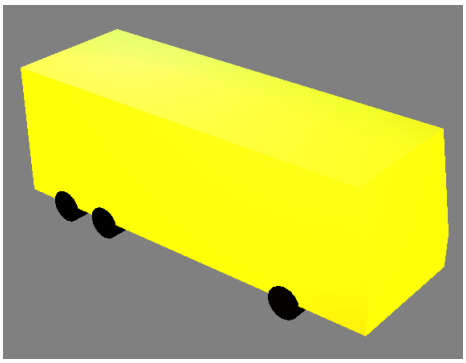


Figure 4.2: Bus.

4.1.2 Static objects

4.2 Environment

The environment includes things such as the surroundings, the weather and the ground surface. These are not animated, so they are simple to place in the scene. Currently, the environments graphical appearance shows as a blue sky texture displayed on the inside of a large sphere, giving the appearance of a homogeneous sky all around the scene.

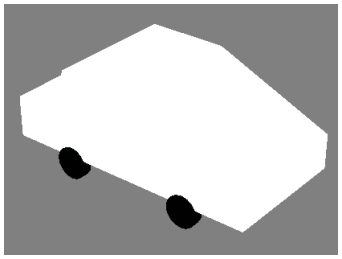


Figure 4.3: Car.

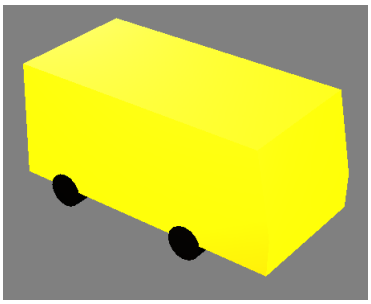


Figure 4.4: Minibus.

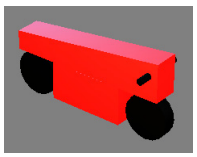


Figure 4.5: Motorcycle.

Figure 4.6: Pedestrian.

The ground is a thin box placed parallel to the x and z axis. It is textured with a grass-like texture.

Currently, no weather effects are implemented in the visualizer.

4.3 Road

There are several different layouts available for the road. These are

- Straight Road – A straight road going either north-south or east-west. The road is represented by a thin box just above ground level (to make sure it is visible).
- Bend – A 90 degree turn, either left or right. The road consists of three graphical elements, two straight roads with the same width and then a turn section. This section is constructed by computing a polygon with two curved sides and two straight sides (where the straight roads will connect).

  The curved sides are computed by first determining an inner radius for the turn and then, for the outer section, stepping through a 90 degree circle arc with a radius of (inner radius + road width). The inner section is then computed backwards the same way. The straight roads are created as above, and then translated to fit with the straight sides of the curve polygon.
- Crossing – A crossing of two straight roads, one going north-south and the other going east-west. The graphical representation is two straight roads.
- T-Crossing – A three-way crossing. The main road goes either north-south or east-west while the incoming road comes in at 90 degrees to the main road. The graphical representation is two straight roads. The incoming

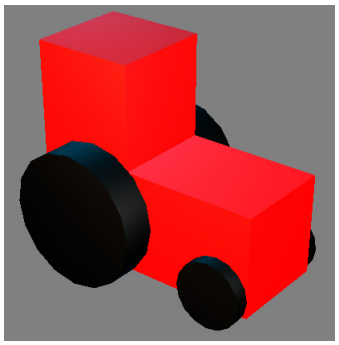


Figure 4.7: Tractor.

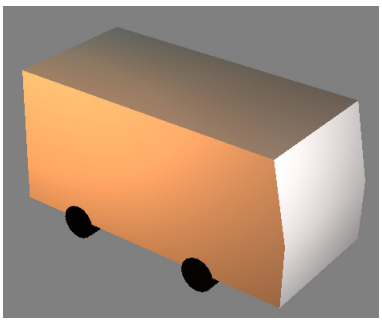


Figure 4.8: Van.

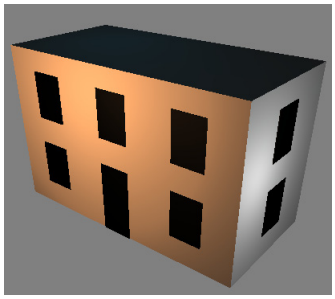


Figure 4.9: Building.

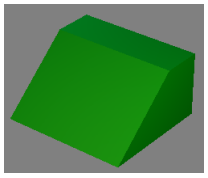


Figure 4.10: Embankment.

road is translated (half its length + half the main road width) away from the crossing to make it end where the main main road meets it.

- Roundabout – A circulation road with up to four roads entering it. The construction of the graphical representation is similar to the bend layout, but the stepping algorithm does a full circle twice this time.

  There can be one to four incoming roads, where each road is translated away from the roundabout so that its end segment is just inside the roundabout. The template actually supports any number of incoming roads, but since the rest of the graphical representation only deals with four directions, the roundabout was also limited to this number.

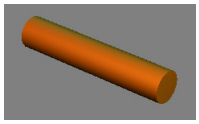


Figure 4.11: Fallen tree.

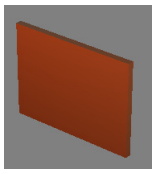


Figure 4.12: Fence.

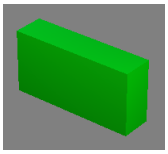


Figure 4.13: Hedge.

Figure 4.14: Pole.

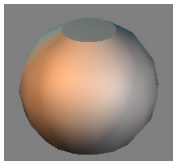


Figure 4.15: Rock.

Figure 4.16: Stop sign.

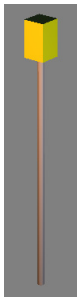


Figure 4.17: Streetlight.

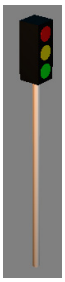


Figure 4.18: Traffic light.

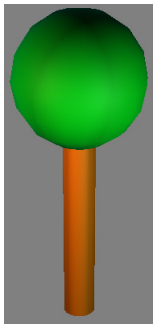


Figure 4.19: Tree.

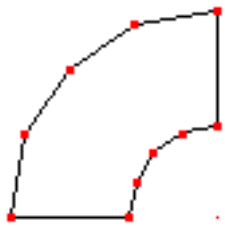


Figure 4.20: A curve polygon.

Chapter 5

Planning

5.1 Overview

A large part of my work involved planning the scene, that is placing objects, creating paths for the dynamic objects, etc. The planning is done in two steps, first placing all objects and then creating paths for the dynamic objects.

Let's use two example texts to illustrate the planning procedure, one with a collision between a dynamic object and a static object and the other between two dynamic objects.

5.1.1 First example

Using the sentence

En bil körde av vägen och krockade med ett träd

as input, the language processing module produces a template with the interesting parts being a static object of the type *Tree*, a dynamic object of the type *Car*, with the events *Driving Forward* and *Leave Road* and also a *collision* involving *Car* and *Tree*. Here is the XML template generated by Carsim:

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<!DOCTYPE Template SYSTEM "file:template.dtd">
<Template>
    <Template_object>
        <RoadObject id="RoadObject1" introducedAs="En bil">
            <DynamicObject>
                <Vehicle>
                    <Car/>
                </Vehicle>
                <DynamicObject_event>
                    <Event>
                        <DrivingForward/>
                        <Position/>
                    </Event>
                    <Event>
```

```
                    <LeaveRoad/>
                    <Position/>
                </Event>
                <Event collision="Collision1">
                    <DrivingForward/>
                    <Position/>
                </Event>
            </DynamicObject_event>
        </DynamicObject>
        <Position/>
    </RoadObject>
    <RoadObject id="RoadObject2" introducedAs="ett träd">
        <StaticObject>
            <Obstacle>
                <VerticalObstacle>
                    <Tree/>
                </VerticalObstacle>
            </Obstacle>
        </StaticObject>
        <Position/>
    </RoadObject>
</Template_object>
<Scene environment="rural">
    <RoadConfig>
        <StraightRoad>
            <SideAttrs/>
            <SideAttrs/>
            <RoadAttrs/>
        </StraightRoad>
    </RoadConfig>
</Scene>
<Template_collision>
    <Collision actor="RoadObject1" id="Collision1"
               victim="RoadObject2"/>
</Template_collision>
</Template>
```

5.1.2 Second example

Using this text

> *En bil körde in i sidan på en annan bil i en korsning.*

Carsim's language module creates a template with two dynamic objects, *Car1* and *Car2*, both with the events *Driving Forward* and a *collision* between the two cars. Here is the XML template generated by Carsim for the second example:

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<!DOCTYPE Template SYSTEM "file:template.dtd">
<Template>
```

```
<Template_object>
    <RoadObject id="RoadObject3" introducedAs="En bil">
        <DynamicObject>
            <Vehicle>
                <Car/>
            </Vehicle>
            <DynamicObject_event>
                <Event collision="Collision2">
                    <DrivingForward/>
                    <Position/>
                </Event>
            </DynamicObject_event>
        </DynamicObject>
        <Position/>
    </RoadObject>
    <RoadObject id="RoadObject4" introducedAs="en annan bil">
        <DynamicObject>
            <Vehicle>
                <Car/>
            </Vehicle>
            <DynamicObject_event>
                <Event collision="Collision2">
                    <DrivingForward/>
                    <Position/>
                </Event>
            </DynamicObject_event>
        </DynamicObject>
        <Position/>
    </RoadObject>
</Template_object>
<Scene environment="rural">
    <RoadConfig>
        <Crossroad>
            <Crossroad_road>
                <RoadConfig>
                    <StraightRoad>
                        <SideAttrs/>
                        <SideAttrs/>
                        <RoadAttrs/>
                    </StraightRoad>
                </RoadConfig>
                <RoadConfig>
                    <StraightRoad>
                        <SideAttrs/>
                        <SideAttrs/>
                        <RoadAttrs/>
                    </StraightRoad>
                </RoadConfig>
            </Crossroad_road>
        </Crossroad>
```

```
        </RoadConfig>
    </Scene>
    <Template_collision>
        <Collision actor="RoadObject3" actorSide="front"
           id="Collision2" victim="RoadObject4"
           victimSide="leftside"/>
    </Template_collision>
</Template>
```

5.2 Position Planning

Often in the texts, precise locations of objects are not mentioned. This means that the position planner has to calculate approximately where in the scene the object should be placed. From Example 1, we know that the tree is most likely placed outside the road and, by default, to the right side of the road. The text doesn't mention anything about the car driving backwards, so we can also assume that the tree is somewhere in front of the car when the animation starts. Thus, we can place the tree a default distance from the road, on the right side of the road.

Dynamic objects are a little harder to place, since they can perform various maneuvers before the collision takes place. The first step is to calculate where the collision takes place, since we know all the events leading up to the collision. When there is a static object involved, this is easy, since the collision has to take place where the static object is placed. When two dynamic objects collide, the placing of the collision is often difficult to calculate, because of their maneuvers leading to their accident. Therefore, it was decided that it was easiest and most efficient to place every collision with to dynamic objects involved in the center of the scene, since that's where the camera is pointing.

Now that we know where the collision takes place, we could try to calculate the starting position of each dynamic object using the event chain in backwards order. This, however, is not easy to do since most events are not set to any specific length or time. An easier approach is to give each dynamic object a starting position a certain distance from the center. This will also mean that we don't risk having any events taking place outside the starting viewing area.

Using Example 1 above, we can now place the car on the right side of the road at a default distance from the center of the scene, with the car facing down the road toward the tree. Looking at Example 2, we would place one car (the impacting vehicle in this case) driving left in the scene and the other car diving south in the scene.

5.3 Path Planning

Now that each dynamic object is placed and we know where the collision takes place, we can create paths for each of the dynamic objects. This is done by first determining where each event should have its starting and ending point. For each event, ten subpoints on the path between its starting and ending point are calculated and then a spline interpolator calculates all necessary path points between each subpoint, thus forming a complete path.

Figure 5.1: Example 1 result.

Figure 5.2: Example 2 result.

Chapter 6

Evaluation

In this section, we present an informal evaluation of the visualization module and we show the results Carsim produces. We selected ten texts randomly from a Swedish corpus of newspaper clippings and we ran through the text analyzer and the visualizer in order.

The results are shown by two screen shots and rated depending on how well they correspond to the template. A result cannot be compared to the actual text, since there are no guarantees that the template produced is correct. The different ratings are:

++	**Very Good**. There may be small glitches, but nothing major.
+	**Good**. Some visuals may not be entirely correct, but the overall visualization is correct.
=	**Acceptable**. You can understand what happened, though some elements are clearly wrong.
−	**Erroneous**. This visualization is not accurate.

6.1 Test1

Text	En krock mellan två bilar strax norr om Laxå utlöste stort larm på söndagens eftermiddag. När räddningspersonalen kom fram kunde de dock konstatera att skadorna inte var så allvarliga och flera ambulanser kunde återkallas, enligt SOS Alarm. Strax efteråt kom larm om en olycka vid Laxsjöarna, men där var det inga personskador alls, uppger Orebropolisen.
Template	<Template> <Template_object> <RoadObject id="RoadObject1" introducedAs="två bilar"> <DynamicObject> <Vehicle> <Car/> </Vehicle> <DynamicObject_event> <Event collision="Collision1"> <DrivingForward/> <Position/> </Event> </DynamicObject_event> </DynamicObject> <Position/> </RoadObject> <RoadObject id="RoadObject2" introducedAs="två bilar"> <DynamicObject> <Vehicle> <Car/> </Vehicle> <DynamicObject_event> <Event collision="Collision1"> <DrivingForward/> <Position/> </Event> </DynamicObject_event> </DynamicObject> <Position/> </RoadObject> </Template_object>

Template (cont.)	<Scene environment="rural" location="strax norr om Laxå"> <RoadConfig> <StraightRoad> <SideAttrs/> <SideAttrs/> <RoadAttrs/> </StraightRoad> </RoadConfig> </Scene> <Template_collision> <Collision actor="RoadObject1" id="Collision1" victim="RoadObject2"/> </Template_collision> </Template>
Rating	+
Comment	The two cars merge in the middle (screen shot 2), a minor glitch. Apart from that, it is a good representation.

Screenshots:

6.2 Test 2

Text	Tre personer skadades allvarligt vid en biloycka i Sjöbo vid niotiden på torsdagskvällen. Den 19-årige föraren, som saknade körkort, körde in i ett träd på Sandåkravägen i Sjöbo. Han och två unga kvinnor, som också färdades i bilen, fick föras till sjukhus. En tredje passagerare skadades lindrigt. Polisen misstänker rattonykterhet.
Template	<Template> <Template_object> <RoadObject id="RoadObject3" introducedAs="Den 19-årige föraren"> <DynamicObject> <Vehicle> <Car/> </Vehicle> <DynamicObject_event> <Event collision="Collision2"> <DrivingForward/> <Position/> </Event> </DynamicObject_event> </DynamicObject> <Position/> </RoadObject> <RoadObject id="RoadObject4" introducedAs="ett träd"> <StaticObject> <Obstacle> <VerticalObstacle> <Tree/> </VerticalObstacle> </Obstacle> </StaticObject> <Position/> </RoadObject> </Template_object>

Template (cont.)	<Scene environment="urban" location="i Sjöbo"> <RoadConfig> <StraightRoad> <SideAttrs/> <SideAttrs/> <RoadAttrs roadName="Sandåkravägen"/> </StraightRoad> </RoadConfig> </Scene> <Template_collision> <Collision actor="RoadObject3" id="Collision2" victim="RoadObject4"/> </Template_collision> </Template>
Rating	+
Comment	The curve motion of the car is not very likely and the tree ends up in the middle of the car. Otherwise a good visualization.

Screenshots:

6.3 Test 3

Text	Tre personer i 60-årsåldern omkom när två personbilar på torsdagskvällen frontalkrockade på E4 utanför i Töre i Norrbotten. Två personer fördes också till sjukhus med svårare skador. - En bil kom av hittills oklar anledning över på fel sida vägen. Den frontalkrockade med en annan bil. En av de skadade fördes till Kalix lasarett och den andra till Sunderbyns sjukhus. Vi har inte kunnat höra dem om olyckan, säger polisinspektör Christer Österstedt till TT. De tre omkomna färdades alla i samma bil, tillsammans med en av de skadade. Olyckan inträffade strax efter klockan åtta. E4 spärrades efter olyckan av helt under räddnings- och bärgningsarbetet, delvis för att en ambulanshelikopter skulle kunna landa. Enligt Österstedt hade vägen åtminstone delvis åter öppnats för trafik sent på torsdagskvällen.
Template	<Template> <Template_object> <RoadObject id="RoadObject15" introducedAs="En bil"> <DynamicObject> <Vehicle> <Car/> </Vehicle> <DynamicObject_event> <Event> <DrivingForward/> <Position/> </Event> <Event> <ChangeLane/> <Position/> </Event> <Event collision="Collision8"> <DrivingForward/> <Position/> </Event> </DynamicObject_event> </DynamicObject> <Position/> </RoadObject> <RoadObject id="RoadObject16" introducedAs="en annan bil"> <DynamicObject> <Vehicle> <Car/> </Vehicle>

Template (cont.)	<DynamicObject_event> <Event collision="Collision8"> <DrivingForward/> <Position/> </Event> </DynamicObject_event> </DynamicObject> <Position/> </RoadObject> </Template_object> <Scene environment="urban" location="i Töre, i Norrbotten"> <RoadConfig> <StraightRoad> <SideAttrs/> <SideAttrs/> <RoadAttrs roadName="E4"/> </StraightRoad> </RoadConfig> </Scene> <Template_collision> <Collision actor="RoadObject15" actorSide="front" id="Collision8" victim="RoadObject16" victimSide="front"/> </Template_collision> </Template>
Rating	=
Comment	You get a decent understanding of what happened, although the car coming from the left should not be backing.

Screenshots:

6.4 Test 4

Text	Bilföraren kan ha drabbats av en hjärtinfarkt, säger polisinsatschef Markus Matsson. Tio äldre har evakuerats och två lägenheter har helt bränts ut. En personbil körde vid femtiden på torsdagseftermiddagen in i ett radhus i ett äldreboende på Alvägen i Enebyberg norr om Stockholm. Bilföraren hittades död i den brinnande bilen som stod helt inne i en av lägenheterna. Trots att husen ligger vid en gata med begränsning till 30 km i timmen och väggupp har bilen kört cirka tio meter in på tomten och rakt in ett rum i lägenheten, där den fattade eld. - Bilföraren som är i femtioårsåldern kan ha drabbats av en hjärtinfarkt eller annan akut sjukdom, säger polisinsatschef Markus Matsson till TT. Elden spred sig från bilen och fyra av sju lägenheter blev helt utbrända eller rökskadade. Ingen av de tio gamla som bor i äldreboendet skadades i olyckan. De flesta av dem har tagits om hand av socialsekreterare som samlat dem för information i ett närbeläget hus. - Det är bra med dem, de får kaffe och prata om det som hänt. De flesta har redan ordnat boende, för de övriga kommer det att ordnas, säger socialnämnden ordförande Boris von Uexküll. Några av de boende har förts till sjukhus för vård av chock och för medicinering men har inga fysiska skador. Signe Lenstad, 83 år, bor i grannlängan och känner flera av de som evakuerats. - Jag kan inte förstå att det har hänt, det är fruktansvärt, säger hon. Ännu vid 20-tiden bekämpade brandkårer från Täby, Vallentuna och Sollentuna branden men hade den under kontroll. Ett daghem i närheten öppnades så att boende i området kunde komma för att få information om det som inträffat. Affärscentrumet Enebytorg en bit från olycksplatsen spärrades av eftersom röken kunde vara farlig och under fredagen ska polisens tekniker undersöka brandplatsen närmare.
Template	<Template> <Template_object> <RoadObject id="RoadObject17" introducedAs="En personbil"> <DynamicObject> <Vehicle> <Car/> </Vehicle> <DynamicObject_event> <Event collision="Collision9"> <DrivingForward/> <Position/> </Event>

Template (cont.)	<Event> <CatchFire/> <Position/> </Event> </DynamicObject_event> </DynamicObject> <Position/> </RoadObject> <RoadObject id="RoadObject18" introducedAs="ett radhus"> <StaticObject> <Obstacle> <HorizontalObstacle> <Building/> </HorizontalObstacle> </Obstacle> </StaticObject> <Position/> </RoadObject> </Template_object> <Scene environment="urban" location="i Enebyberg, norr om Stockholm"> <RoadConfig> <StraightRoad> <SideAttrs/> <SideAttrs/> <RoadAttrs roadName="Alvägen"/> </StraightRoad> </RoadConfig> </Scene> <Template_collision> <Collision actor="RoadObject17" id="Collision9" victim="RoadObject18"/> </Template_collision> </Template>
Rating	+
Comment	A fairly good visualization, though the problem with the curve mentioned in test 2 is evident here as well. The car ends up inside the house, a minor glitch.

Screenshots:

6.5 Test 5

Text	En ung man omkom och en ung kvinna fick livshotande skador i en svår trafikolycka på riksväg 45 norr om Kil i Värmland. Elva personer fick föras till sjukhus efter olyckan, som inträffade vid Frykerud tidigt på lördagseftermiddagen. Tre bilar var direkt inblandade i olyckan, som hände i halt väglag. Mannen som omkom, och kvinnan som fick mycket svåra skador, kom söderifrån i en bil som troligen först touchade en annan bil och sedan krockade med en tredje bil som kom norrifrån, uppger polisen i Värmland. Personerna i de övriga bilarna fick lindriga skador. Det fick också två personer i en dansk minibuss, som körde i diket för att undvika att köra på de andra fordonen. Dessutom blev fyra personer i ytterligare en dansk bil så svårt chockade att de fick föras till sjukhus.
Template	<Template> <Template_object> <RoadObject id="RoadObject28" introducedAs="en bil"> <DynamicObject> <Vehicle> <Car/> </Vehicle> <DynamicObject_event> <Event collision="Collision13"> <DrivingForward/> <Position/> </Event>

Template (cont.)

```
        <Event collision="Collision14">
          <DrivingForward/>
          <Position/>
        </Event>
        <Event collision="Collision15">
          <DrivingForward/>
          <Position/>
        </Event>
      </DynamicObject_event>
    </DynamicObject>
    <Position/>
  </RoadObject>
  <RoadObject id="RoadObject29"
   introducedAs="en annan bil">
    <DynamicObject>
      <Vehicle>
        <Car/>
      </Vehicle>
      <DynamicObject_event>
        <Event collision="Collision13">
          <DrivingForward/>
          <Position/>
        </Event>
      </DynamicObject_event>
    </DynamicObject>
    <Position/>
  </RoadObject>
  <RoadObject id="RoadObject30"
   introducedAs="en tredje bil">
    <DynamicObject>
      <Vehicle>
        <Car/>
      </Vehicle>
      <DynamicObject_event>
        <Event collision="Collision14">
          <DrivingForward/>
          <Position/>
        </Event>
        <Event collision="Collision16">
          <DrivingForward/>
          <Position/>
        </Event>
      </DynamicObject_event>
    </DynamicObject>
    <Position/>
  </RoadObject>
```

Template (cont.)	<RoadObject id="RoadObject31" introducedAs="en dansk minibuss"> <DynamicObject> <Vehicle> <Minibus/> </Vehicle> <DynamicObject_event> <Event collision="Collision15"> <DrivingForward/> <Position/> </Event> <Event collision="Collision16"> <DrivingForward/> <Position/> </Event> </DynamicObject_event> </DynamicObject> <Position/> </RoadObject> </Template_object> <Scene environment="rural" location="norr om Kil, i Värmland, vid Frykerud" roadCondition="icy"> <RoadConfig> <StraightRoad> <SideAttrs/> <SideAttrs/> <RoadAttrs roadName="riksväg 45"/> </StraightRoad> </RoadConfig> </Scene> <Template_collision> <Collision actor="RoadObject28" id="Collision13" victim="RoadObject29"/> <Collision actor="RoadObject28" id="Collision14" victim="RoadObject30"/> <Collision actor="RoadObject31" id="Collision15" victim="RoadObject28"/> <Collision actor="RoadObject31" id="Collision16" victim="RoadObject30"/> </Template_collision> </Template>
Rating	+
Comment	Apart from the merging of all the vehicles in the middle, this visualization works well.

Screenshots:

6.6 Test 6

Text	Prins William av Luxemburg, son till landets regerande storhertig, skadades i går allvarligt i en trafikolycka i Frankrike. Den 37-årige prinsen fördes till Henri-Mondor-sjukhuset i Parisförorten Créteil där han rapporterades ligga i koma med svåra skallskador. Fem andra, bland dem hans fru Sibylla, skadades också i olyckan på motorvägen mellan Paris och Bordeaux. Prinsen körde en hyrbil och törnade emot ett annat fordon, voltade och kördes på av en annan bil.
Template	<Template> <Template_object> <RoadObject id="RoadObject32" introducedAs="Prinsen"> <DynamicObject> <Vehicle> <Car/> </Vehicle> <DynamicObject_event> <Event collision="Collision17"> <DrivingForward/> <Position/> </Event> </DynamicObject_event> </DynamicObject> <Position/> </RoadObject> <RoadObject id="RoadObject33" introducedAs="ett annat fordon"> <DynamicObject> <Vehicle> <Car/> </Vehicle> <DynamicObject_event> <Event collision="Collision17"> <DrivingForward/> <Position/> </Event> <Event> <Overturn/> <Position/> </Event> </DynamicObject_event> </DynamicObject> <Position/> </RoadObject> </Template_object>

Template (cont.)	<Scene environment="rural" location="i Frankrike"> <RoadConfig> <StraightRoad> <SideAttrs/> <SideAttrs/> <RoadAttrs/> </StraightRoad> </RoadConfig> </Scene> <Template_collision> <Collision actor="RoadObject32" id="Collision17" victim="RoadObject33"/> </Template_collision> </Template>
Rating	=
Comment	The overturn of the first car is not displayed correctly.

Screenshots:

6.7 Test 7

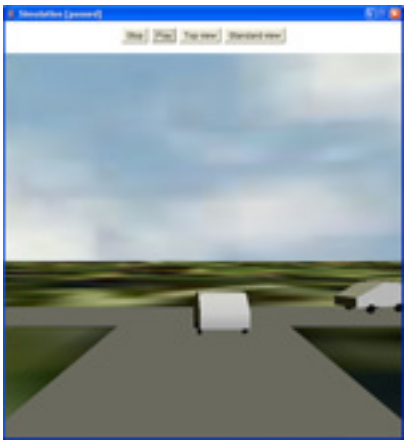
Text	Tre bilar var inblandade i en krock i Skottorp på lördagen. Men ingen av förarna skadades allvarligt. Flera Skottorpsbor vill ha sänkt hastighet på genomfartstrafiken. Det var vid middagstid på lördagen som tre bilar blev inblandade i en krock i höjd med Skummeslövsfrisörerna i Skottorp. En kvinna kom från Skottorps slott och skulle svänga ut på genomfartsvägen som går mellan Vallberga och Östra Karup. En yngre man kom samtidigt norrifrån och körde in i kvinnans bil. Det resulterade i att kvinnans bil for rakt in i en trädgård på andra sidan vägen och mannens bil kanade vidare. Flera inblandade Samtidigt hade en äldre kvinna precis satt sig i bilen på frisörsalongens parkering. Hon var i färd med att backa ut när den norrifrån kommande bilen kanade in i hennes fordon. Ingen av de inblandade uppges ha ådragit sig några allvarliga skador. Kvinnan som svängde ut från Skottorps slott var den första som fördes med ambulans till sjukhus. Den äldre kvinnan satt fastklämd i över en halvtimma innan räddningspersonalen kunde få loss henne. Hon var medvetande, men chockad. Den yngre mannan klarade sig utan några fysiska skador. Vänstra sidan av fronten på bilen blev skrot. Sänkt hastighet Vid utfarten från samhället har det inträffat olyckor tidigare. Flera boende längs vägen uppger att det förekommer en hel del höga hastigheter förbi samhället. Hastigheten är begränsad till 70 kilometer i timmen. Påtryckningar på Vägverket om en hastighetssänkning har inte vunnit gehör. Anledningen har sagts vara att det bara ligger hus på ena sidan av vägen.
Template	<Template> <Template_object> <RoadObject id="RoadObject44" introducedAs="Tre bilar"> <DynamicObject> <Vehicle> <Car/> </Vehicle> <DynamicObject_event> <Event> <DrivingForward/> <Position/> </Event>

Template (cont.)

```
      <Event>
        <Turn/>
        <Position/>
      </Event>
      <Event collision="Collision23">
        <DrivingForward/>
        <Position/>
      </Event>
    </DynamicObject_event>
  </DynamicObject>
  <Position/>
</RoadObject>
<RoadObject id="RoadObject45"
 introducedAs="kvinnans bil">
  <DynamicObject>
    <Vehicle>
      <Car/>
    </Vehicle>
    <DynamicObject_event>
      <Event>
        <DrivingForward/>
        <Position/>
      </Event>
      <Event>
        <Slide/>
        <Position/>
      </Event>
      <Event collision="Collision22">
        <DrivingForward/>
        <Position/>
      </Event>
      <Event collision="Collision23">
        <DrivingForward/>
        <Position/>
      </Event>
    </DynamicObject_event>
  </DynamicObject>
  <Position/>
</RoadObject>
<RoadObject id="RoadObject46"
 introducedAs="Tre bilar">
  <DynamicObject>
    <Vehicle>
      <Car/>
    </Vehicle>
```

Template (cont.)	<DynamicObject_event> <Event collision="Collision22"> <DrivingForward/> <Position/> </Event> </DynamicObject_event> </DynamicObject> <Position/> </RoadObject> </Template_object> <Scene environment="urban" location="i Skottorp, i höjd med Skummeslövsfrisörerna, i Skottorp, ut från Skottorps slott"> <RoadConfig> <TCrossing> <RoadConfig> <StraightRoad> <SideAttrs/> <SideAttrs/> <RoadAttrs/> </StraightRoad> </RoadConfig> <RoadConfig> <StraightRoad> <SideAttrs/> <SideAttrs/> <RoadAttrs/> </StraightRoad> </RoadConfig> </TCrossing> </RoadConfig> </Scene> <Template_collision> <Collision actor="RoadObject46" id="Collision22" victim="RoadObject45"/> <Collision actor="RoadObject44" id="Collision23" victim="RoadObject45"/> </Template_collision> </Template>
Rating	–
Comment	The three cars all display odd behavior in the middle of the crossing, so it is hard to tell what really happens.

Screenshots:

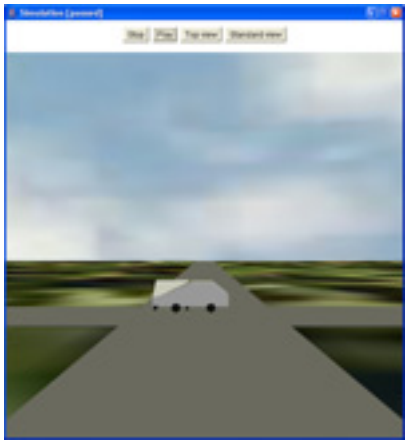


6.8 Test 8

Text	En 18-årig kvinnan har avlidit av de skador hon fick vid en krock mellan två personbilar i Vä utanför Kristianstad på fredagskvällen. Åtta personer skadades i olyckan. På söndagen vårdades två av dem fortfarande på sjukhus, men ingen av dem uppges ha livshotande skador, skriver Kristianstadsbladet. De två bilarna kolliderade i en vägkorsning, men hur olyckan gått till är oklart.
Template	<Template> <Template_object> <RoadObject id="RoadObject50" introducedAs="två personbilar"> <DynamicObject> <Vehicle> <Car/> </Vehicle> <DynamicObject_event> <Event collision="Collision25"> <DrivingForward/> <Position/> </Event> </DynamicObject_event> </DynamicObject> <Position/> </RoadObject> <RoadObject id="RoadObject51" introducedAs="två personbilar"> <DynamicObject> <Vehicle> <Car/> </Vehicle> <DynamicObject_event> <Event collision="Collision25"> <DrivingForward/> <Position/> </Event> </DynamicObject_event> </DynamicObject> <Position/> </RoadObject> </Template_object>

Template (cont.)	<Scene environment="urban" location="i Vä, utanför Kristianstad"> <RoadConfig> <Crossroad> <Crossroad_road> <RoadConfig> <StraightRoad> <SideAttrs/> <SideAttrs/> <RoadAttrs/> </StraightRoad> </RoadConfig> <RoadConfig> <StraightRoad> <SideAttrs/> <SideAttrs/> <RoadAttrs/> </StraightRoad> </RoadConfig> </Crossroad_road> </Crossroad> </RoadConfig> </Scene> <Template_collision> <Collision actor="RoadObject50" id="Collision25" victim="RoadObject51"/> </Template_collision> </Template>
Rating	+
Comment	A good visualization.

Screenshots:



6.9 Test 9

Text	Den kvinnliga förare som häromdagen körde på en 15-årig flicka på norr i Katrineholm blev än mer skärrad när flickans pappa kom till olycksplatsen. Han började sparka på bilen och slog även näven genom framrutan så att glaset krossades. Den kvinnliga föraren hade lånat bilen av sin far, som nu polisanmäler mannen för skadegörelse. Den påkörda flickan uppges ha stått bakom en parkerad bil och väntat på att en buss skulle köra förbi, sedan gick hon ut i gatan. Hon blev omhändertagen på Kullbergska för smärtor i ben och höfter, men kunde lämna sjukhuset samma dag.
Template	<Template> <Template_object> <RoadObject id="RoadObject58" introducedAs="Den kvinnliga förare"> <DynamicObject> <Vehicle> <Car/> </Vehicle> <DynamicObject_event> <Event collision="Collision29"> <DrivingForward/> <Position/> </Event> </DynamicObject_event> </DynamicObject> <Position/> </RoadObject> <RoadObject id="RoadObject59" introducedAs="en 15-årig flicka"> <DynamicObject> <Animate> <Pedestrian/> </Animate> <DynamicObject_event> <Event collision="Collision29"> <DrivingForward/> <Position/> </Event> </DynamicObject_event> </DynamicObject> <Position/> </RoadObject> </Template_object>

Template (cont.)	<Scene environment="urban" location="i Katrineholm"> <RoadConfig> <StraightRoad> <SideAttrs/> <SideAttrs/> <RoadAttrs/> </StraightRoad> </RoadConfig> </Scene> <Template_collision> <Collision actor="RoadObject58" id="Collision29" victim="RoadObject59"/> </Template_collision> </Template>
Rating	–
Comment	The major flaw is that the pedestrian moves faster than the car. Unfortunately, this gives the impression that the pedestrian runs into the back of the car, instead of the other way around.

Screenshots:

6.10 Test 10

Text	En personbil och en mejerilastbil krockade i Öjebyn strax för ett-tiden på tisdagen. Lastbilen kom körande efter Sandgatan och skulle korsa Hammarvägen. Personbilen missande högerregeln och träffades av mejeribilen på höger sida och blev ordentligt intryckt. I baksätet fanns ett barn i bilbarnsstol. Enligt polisen i Piteå fördes alla tre inblandade till lasarett för vård. Chauffören av lastbilen chockades och barnet fick skärsår. Ingen fick livshotande skador.
Template	<Template> <Template_object> <RoadObject id="RoadObject60" introducedAs="En personbil"> <DynamicObject> <Vehicle> <Car/> </Vehicle> <DynamicObject_event> <Event collision="Collision30"> <DrivingForward/> <Position/> </Event> </DynamicObject_event> </DynamicObject> <Position/> </RoadObject> <RoadObject id="RoadObject61" introducedAs="en mejerilastbil"> <DynamicObject> <Vehicle> <Truck/> </Vehicle> <DynamicObject_event> <Event collision="Collision30"> <DrivingForward/> <Position/> </Event> </DynamicObject_event> </DynamicObject> <Position/> </RoadObject> </Template_object>

Template (cont.)	<Scene environment="urban" location="i Öjebyn"> <RoadConfig> <StraightRoad> <SideAttrs/> <SideAttrs/> <RoadAttrs roadName="Sandgatan"/> </StraightRoad> </RoadConfig> </Scene> <Template_collision> <Collision actor="RoadObject60" id="Collision30" victim="RoadObject61"/> </Template_collision> </Template>
Rating	=
Comment	This accident is displayed correctly. However, the incorrect representation of the truck makes it a bit confusing.

Screenshots:

6.11 Test conclusions

A table of the test ratings shows that a majority of tested templates produced an acceptable visualization, though none were deemed to be very good. While ten tests may be too few to accurately judge the performance of the visualizer, these results are promising.

Rating	Number of tests
++	0
+	5
=	3
–	2

Bibliography

[Åkerberg et al., 2003] Åkerberg, O., Svensson, H., Schulz, B., and Nugues, P. (2003). Carsim: An automatic 3D text-to-scene conversion system applied to road accident reports. In *Proceedings of the Research Notes and Demonstrations of the 10th Conference of the European Chapter of the Association of Computational Linguistics*, pages 191–194, Budapest, Hungary. Association for Computational Linguistics.

[Coyne and Sproat, 2001] Coyne, B. and Sproat, R. (2001). Wordseye: An automatic text-to-scene conversion system. In *Proceedings of the Siggraph Conference*, Los Angeles.

[Dupuy et al., 2001] Dupuy, S., Egges, A., Legendre, V., and Nugues, P. (2001). Generating a 3D simulation of a car accident from a written description in natural language: The Carsim system. In *Proceedings of The Workshop on Temporal and Spatial Information Processing*, pages 1–8, Toulouse. Association for Computational Linguistics.

[Egges et al., 2001] Egges, A., Nijholt, A., and Nugues, P. (2001). Generating a 3D simulation of a car accident from a formal description. In Giagourta, V. and Strintzis, M. G., editors, *Proceedings of The International Conference on Augmented, Virtual Environments and Three-Dimensional Imaging (ICAV3D)*, pages 220–223, Mykonos, Greece.