Dynamically Changing Road Networks - Modelling and Visualization in Real Time

Christian Mark[1], Armin Kaußner[1], Martin Grein[1], and Hartmut Noltemeier[2]

[1] Center for Traffic Sciences (IZVW), Germany
WWW home page: http://www.izvw.de
[2] University of Würzburg, Department of Computer Science D-91405 Würzburg, Germany
WWW home page: http://www1.informatik.uni-wuerzburg.de/en/

Abstract. In this paper we present a new concept for a driving simulator database. The database consists of a three-dimensional road network, which is not fixed like in conventional databases. The parts lying outside the viewing range of the driver can be changed during simulation.
Changes in the road network are either initiated interactively by the researcher or automatically by the system based on predefined conditions.
Conventional visualization methods are not usable for such a road network. Therefore we present new visualization algorithms for real time graphics. The visualization is based on a layered coarsening of the geometrical representation. The algorithms presented in this paper guarantee a framerate of at least 60fps on standard PCs.
The concept is implemented in a high-fidelity simulator and currently used for psychological research on traffic and driver behaviour.

1 Introduction

It is commonly understood that driving simulation is an efficient tool in driver training as well as in traffic research. The more simulation is used in both areas the higher performance the user asks for. Therefore, new flexible concepts are needed. One of them is dynamic modelling of road networks.

Conventional Driving Simulators
Conventional driving simulators use a part of a virtual or real world as a database for the scenery (see [1]). They define a road network based on the geographical information about the world. Therefore, the road network is fixed and cannot be changed during simulation. A road network of this kind is called globally geometrically consistent and can be represented by a map.

Requirements of the Simulator at the IZVW
The restriction to drive within a fixed database often conflicts with the requirements of experimental research.
The following example - coming from research carried out at the Center for Traffic Sciences - may demonstrate the practical requirements:

To examine the effects of workload on driving, the driver has to handle a specific sequence of situations.

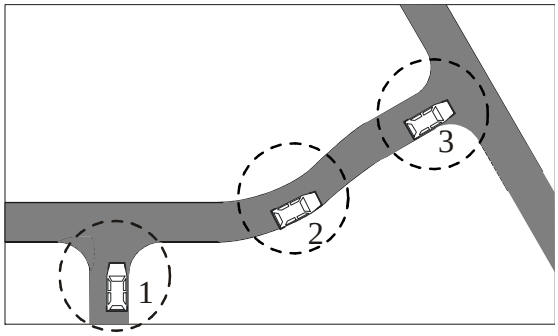

Fig. 1: Locally geom. consistent road network

Figure 1 shows an example. The circles around the driver illustrate his region of visibility. The radius of the circles in the simulation is about two kilometres. The driver is in position 1 and should turn left. Ignoring the instructions of the experimenter, he turns right. If the driver is in position 2 the crossroad of position 1 is moved to position 3 and the requested situation is repeated until he turns left.

The basic principle of the example in Figure 1 is that the road network is adapted to the behaviour of the driver. To make this adaptation possible, the experimenter has to define a topological representation of a road network. The resulting network is no longer globally geometrically consistent but only locally geometrically consistent. "Local" means that only the driver's region of visibility is geometrically consistent. Outside this region the road network can be changed without considering geometric limitations (see [2], [5]).

As a consequence the polygons for visualization have to be computed for every frame and cannot be built offline as for fixed databases. The framerate of this visualization must not drop below 60 fps to meet the requirements of modern driving simulators. In addition it should be possible to run the simulation completely on standard hardware (PCs) instead of specialized graphics computers.

This paper shows an alternative method for design and visualization that is built up at runtime. The main purpose is to find algorithms for visualization that satisfy our needs with as few resources as possible. The saved resources will be needed for changing the scenery at runtime as well as for traffic simulation.

2 Road Network

The road network is the basis of the scenery. It is a graph $RN = (V, E)$. The set of nodes V is divided into two parts: $V = COURSES \cup AREAS$. A $v \in COURSES$ is a continuous road segment of arbitrary length without turn-offs. The profile of the road's cross section (number of lanes, their width, type, ...) does not change.

A $v \in AREAS$ is a rectangular area containing lanes that model junctions as well as transitions between $COURSES$. An $AREA$ can have a different cross section at every port in order to connect $COURSES$ with different cross sections e.g. drive-ups to highways.

Each $v \in V$ has ports $v.p_i$ that connect to other nodes of V. A $v \in COURSES$ has the ports $v.BEGIN$ and $v.END$. A $v \in AREAS$ has an arbitrary number of ports $v.p_1, \ldots, v.p_n$. Each port is placed at one side (top, bottom, left, right) of the underlying rectangle. The ports on one side are labeled as TOP_1, $TOP_2, \ldots$, $BOTTOM_1, \ldots$. An edge out of E always connects ports of two nodes: $E = \{(v.p_i, w.p_j) : v, w \in V\}$.

From a topological view these connections can be ambiguous, as shown in Figure 2. A node can be connected to more than one other node as for $COURSE$ C_1 in the example. If the region "behind" or on the right side of $COURSE$ C_1 gets visible for the driver, one of these connections must be chosen to solve the ambiguity and create a geometrically consistent road network. They are resolved by an algorithm called geometric instantiation (see Algorithm 1).

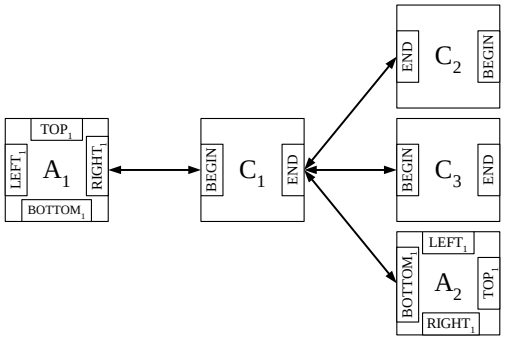

Fig. 2: A simple road network:$\mathbf{C_1}$, $\mathbf{C_2}$, $\mathbf{C_3} \in$ **COURSES**; $\mathbf{A_1}$, $\mathbf{A_2} \in$ **AREAS**

Remark on Figure 2:
The three connections from $C_1.END$ to $C_2.END$, $C_3.BEGIN$ and $A_2.BOTTOM_1$ represent alternatives in the road network. They are selected at runtime, as described in section 3.5.
In this paper we will focus on modelling and visualizing courses.

3 Modelling of *COURSES*

The model of a course consists of the following three parts:

1. Surface of the road
2. Height profile of the surrounding landscape
3. Objects on the surface

3.1 Modelling of the Road

The path a *COURSE* takes through the environment is described by a unit speed curve. Let c be a *COURSE* of length S. The unit speed curve $c.p : [0, S] \rightarrow \mathbb{R}^3$ specifies the path of the road centre. A position on this curve is specified by a number $s \in [0; S]$, which indicates the distance in meters from Port *BEGIN*. For the sake of convenience this number will be called "roadposition" from now on. The paths can be restricted to conform to german guidelines for road design (see [6]).

3.2 Height Profile of the Surrounding Landscape

It has to be guaranteed that the simulation will not reach undefined states in the case the driver leaves the road. Therefore, the representation of the surrounding landscape must provide detailed information. For example, the physical vehicle model needs the angles of inclination in each point of the landscape.

Let $c \in COURSES$ be a node of a road network. Each *COURSE* has a landscape on the left and on the right side of the road. On each side of c, a set of so-called height functions $\{F_1(d), \ldots, F_n(d)\}$ running perpendicularly to the unit speed curve is defined. $F_i(d)$ starts at a certain position $s_i \in [0, S]$ on the road. It defines the elevation of a point (relative to the road surface) which has distance d to the road. An example of a height curve can be:

$$F_i(d) = G(d, d_g^{(i)}) \cdot \left(A_1^{(i)} \cdot \sin\left(f_1^{(i)} \cdot d \right) + A_2^{(i)} \cdot \sin\left(f_2^{(i)} \cdot d \right) + o^{(i)} \right) \qquad \text{(HC)}$$

Two superposed sine waves and a constant offset $o^{(i)}$ are multiplied by a smoothing function G. The function F_i would then be specified by so-called shape parameters: $A_1^{(i)}$, $A_2^{(i)}$, $f_1^{(i)}$, $f_2^{(i)}$, $o^{(i)}$.
A smoothing function G has the following properties:

1. $G(0, d_g^{(i)}) = 0$
2. $G(d, d_g^{(i)}) = 1$ if $d \geq d_g^{(i)}$
3. $G'(0, d_g^{(i)}) = G'(d_g^{(i)}, d_g^{(i)}) = 0$

Independently from the slopes of the sine waves and the offset, this function ensures a smooth increase of elevation up to a distance $d_g^{(i)}$ from the road.

The maximum value of d is computed at runtime in a such way that no intersections of two height functions exist on the inner side of a bend. The height of points beyond such an intersection is ambiguous and therefore useless. The surface in such areas is just plane. This convention is sufficient because offroad driving is not intended.
Let s^* be an arbitrary point lying between two pre-defined height curves e.g. $s_i < s^* < s_{i+1}$. Let p^* be a point on the virtual height curve in s^* with a distance d^* to the road (see Figure 3). To determine the exact height of p^*, we must interpolate the height between the two neighbouring height curves. Therefore we create a new height curve by a linear interpolation of the neighbouring parameters. With increasing distance between s_i and s_{i+1} we have to build more than one additional height curve.

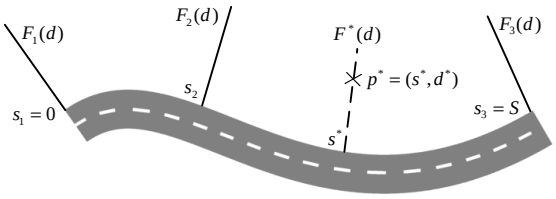

Fig. 3: A **COURSE** with height functions on its left side

3.3 Objects on the Surface

To design a realistic landscape, various objects *obj* must be placed onto the surface. Therefore, each *COURSE* c has a list $c.objList$ of objects *obj*. Each object consists of a roadposition $obj.s$, a distance $obj.d$ and a 3D graphical representation. The graphical representation of an object is placed at the position specified by $obj.s$ and $obj.d$.

3.4 Automatic Generation of Landscapes

The presented way of modelling a course is very time consuming and in part difficult to handle. One reason is that the experimenter has to specify a lot of height curves. In most cases he is not interested in details of the landscape, he only wants to control the basic appearance by a few parameters. Therefore, the experimenter has the option to specify the following parameters:

1. altitude difference
2. ripple
3. offset

For each parameter the experimenter may specify an interval. During simulation a random number is drawn from the interval and assigned to the appropriate parameter. Altitude difference controls the amplitudes and ripple controls the frequencies of the sine waves defining height curves. Height curves are created every 500m at each side of a course automatically.

Designing a realistic surface also means to distribute many objects within the landscape. To shorten this lengthy operation, the experimenter has the option to select a landscape type. At the moment there are three types: farmed, forested and inhabited. The objects which are characteristic for a specific landscape type are placed on the surface in a reasonable way by using R-Trees (see [3] as a method for spatial indexing)

Regarding the example from the beginning, it is now possible to vary the height profile as well as the objects on the landscape every time a node gets visible for the driver.

3.5 Geometric Instantiation

In this section the subgraph of the road network representing the drivers' region of visibility will be constructed. The first *COURSE* in this subgraph is the one nearest to the driver. Starting with this *COURSE*, the geometric instantiation algorithm traverses the road network along the ports. If there is more than one course connected to a port the algorithm selects one of them according to criteria defined by the experimenter. Furthermore, any *COURSE* in a distance of more than two kilometres to the driver is defined as being outside the region of visibility.

During simulation a global *timer* is counting the number of displayed frames. Every *COURSE* has a variable *tInstantiation* for saving the highest frame number at which the associated *COURSE* was visible. This variable is used for avoiding infinite loops (particularly if areas are also used for creating a road network). In addition, every port has a pointer called *edge* where a reference to the next visible *COURSE* is saved.

Algorithm 1 *COURSE*::Geom_Inst(*COURSE* caller)

```
    if tInstantiation == time then {abortion of recursion}
      return;
    end if
    bool outside = (COURSE is completely invisible ?)
 5: if outside then
      Reset edges of port BEGIN and port END
      return
    end if
    if tInstantiation < time-1 then {COURSE not treated in last inst. step ?}
10:   generate_height_profile();
      transform(caller); {transform COURSE smoothly to caller}
      distribute_objects();
      Port p = Port BEGIN;
      while p ∈ {BEGIN, END} do {loop over all ports connected to COURSE}
15:     p.edge = SelectEdge(); {select edge out of the alternatives of port p}
        p = NextPort();
      end while
    end if
    tInstantiation = time {set instantiation timestamp}
20: (Course connected to BEGIN)→Geom_Inst(this)
    (Course connected to END)→Geom_Inst(this)
```

Remarks on the geometric instantiation algorithm:

Line 10:
Due to the definition of the experimenter a new height profile is generated for *COURSES* getting visible.

Line 11:
A *COURSE* getting visible has to be connected to the calling *COURSE*. Therefore the path of their roads has to be connected smoothly. In addition the first height curve of the new *COURSE* is set to the same values as the last height curve of the *COURSE* already visible.

Line 12:
A set of objects is chosen and distributed on the surface generated in the line before.

Line 15:
If a port is connected to more than one *COURSE*, one of them is chosen due to various measurements (already described in this paper).

Line 20, 21:
Every neighbouring *COURSE* to the current *COURSE* will be instantiated.

4 Visualization of *COURSES*

This section focuses on the visualization of the subgraph given by algorithm 1. First a visualization of the road surface is generated. Then, the surface of the landscape will be generated and the objects on the terrain will be visualized.

4.1 Surface of Roads

In order to visualize the road of a *COURSE* c, a sequence of quadrilaterals (quadstrips) parallel to the centre of the road has to be generated.

First of all a set of points is chosen on the unit speed curve, at which the vectors of the quadstrips are calculated. These points are distributed along the *COURSE* unevenly: Closer to the driver and in curves the point density is higher in order to ensure smoothness of representation; otherwise, the distances between points are increased to save resources in visualization.

All other elements of a road which are parallel to the unit speed curve are visualized in the same way. Vertical quadstrips can be used to create guardrails or barriers. Examples for these elements are guardrails or noise barriers.

Figure 4 shows the visualization of a *COURSE*. The larger the distance to the driver, the coarser is the subdivision; the higher the curvature, the finer is the subdivision.

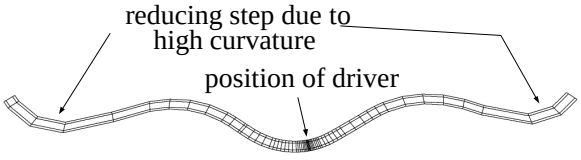

Fig. 4: Visualization of a road

4.2 Landscapes of Courses

Starting with the boundary points used for visualizing the road surface, the surface of a landscape is visualized by a series of quadstrips. These quadstrips are placed parallel to the road. The width (parallel to the path of a road) of each quad between two boundary points depends on the distance to the driver.

Figure 5 shows a complete subdivision of a landscape which was generated according to these conventions. In the figure the surface is restricted to the areas in which no height curves overlap each other.

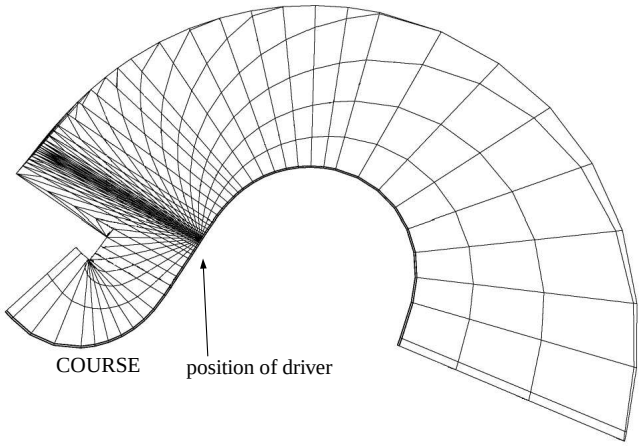

Fig. 5: Visualization of a landscape

The generated subdivision is aligned perpendicularly to the road. Along the road the subdivision of the landscape depends on the subdivision of the road and not on the distance to the driver.
Therefore the length of a quad must be greater than a minimal value. This minimal value depends on the distance to the driver. If the length of a quad is smaller than the value, a boundary point is dropped out. Figure 6 shows the same landscape as Figure 5 using this advanced method.

The presented methods generate a surface consisting of polygons. Local extreme values will be visualized if they coincide with the corner of a polygon. This will not be the case if they lie inside or on the edge of a polygon. This would result in a surface which seems to "flutter". To avoid this effect, a simple heuristic algorithm was developed which detects those points or at least points nearby. In equation (HC) two superposed sine waves are used to build a height curve. Assuming that the first sine wave has a very low amplitude in comparison to the second one, it suffices to calculate the extreme values of the second and to disregard the influence of the first sine wave.

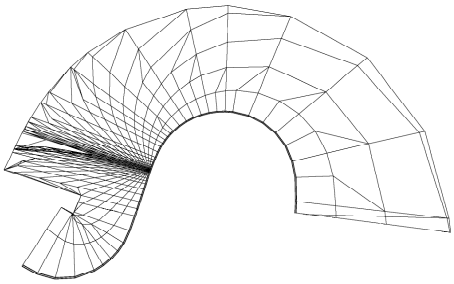

Fig. 6: Visualization of a landscape with subdivision tangential to the road

Figure 7 shows the number of vectors used for a surface. The three described methods are compared concerning the number of required vectors.

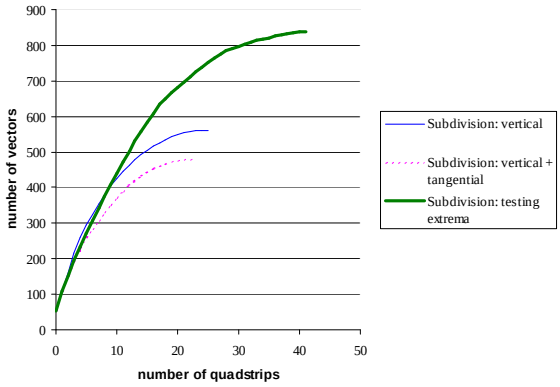

Fig. 7: Comparison of methods generating surfaces of landscapes

The x-axis gives the number of quadstrips parallel to the road which have been used to visualize the surface. We can save lots of vectors using these subdivision techniques.

4.3 Objects on the Surface

The next step after visualizing the surface is to place objects within the landscape. Let *obj* be an object on a landscape. The associated graphical representation must be transformed in such a way that the reference point has a distance of $obj.d$ to the road at roadposition $obj.s$.

Figure 8 shows an example of a *COURSE* with different height profiles and landscape types on the left vs. right side of the road and various examples of objects on the surface.

Fig. 8: Example of a visualization of a road network

The usage of predefined graphical objects which are transformed to fit to the surface makes it possible to change the appearance of the surface. This means that the driver cannot realize that he is driving on the same course.

5 Conclusion

The presented methods enhance the use of driving simulators in traffic sciences. Dynamic modelling of road networks and automatic generation of landscapes open up new experimental procedures for driving simulation. Simulation becomes fully adaptive to the behaviour of the driver, offering him or her the adequate driving situation throughout the whole simulation.

References

1. M. Desrochers, "Database Structures", Tagungsband 6. Workshop Sichtsysteme - Visualisierung in der Simulationstechnik [Proceedings of the 6.th Workshop for Visual Systems - Visualization for Simulation Technology], Shaker Verlag, Bremen, 1999, pp. 165-190.
2. M. Grein, A. Kaußner, H.-P. Krüger, H. Noltemeier, "A flexible application framework for distributed real time systems with applications in PC based driving simulators", Proceedings of the Driving Simulation Conference, DSC2001, Sophia-Antipolis France, 2001.
3. A. Guttman, "R-TREES: A dynamic index structure for spatial searching", Proc. ACM SIGMOD Int. Conference on Management of Data, Boston, 1984, pp. 47-57.
4. J. Hammes, "Modeling of ecosystems as a data source for real-time terrain rendering", LNCS 2181, Springer, 2001.
5. A. Kaußner, M. Grein, H.-P. Krüger, H. Noltemeier, "An architecture for driving simulator databases with generic and dynamically changing road networks", Proceedings of the Driving Simulation Conference, DSC2001, Sophia-Antipolis France, 2001.
6. Forschungsgesellschaft für Straßen- und Verkehrswesen (Arbeitsgruppe Strassenentwurf), "Richtlinien für die Anlage von Straßen RAS Teil: Linienführung RAS-L und Querschnitte RAS-Q" [German Guidelines for Road Construction], 1995.