A Visual System for a Traffic Simulator

R. Möller

Fachbereich Elektrotechnik, Lehr- und Forschungsgebiet Automatisierungstechnik und graphische Datenverarbeitung, Institute of Automation Techniques, University of Wuppertal, Fuhlrottstraße 10, D-5600 Wuppertal 1, F. R. Germany

INTRODUCTION

The prototype of a modular CGI-system for real time simulation in a traffic simulator will be presented. It will be shown, that with the proposed configuration of a large asymmetric multiprocessor system, organized in different layers of homogenous partial systems, and an overlayed pipeline - architecture the usage of common hardware components is generally possible for the realization of low cost solutions.

On the other hand the technology of the used processors (mostly MOS) requires intelligent programming to guarantee a real time reponse of the visual system. The well known algorithms of computer graphics are to be analyzed for their applicability, that means, that parallelisms in graphics algorithms are to be found and the required precision and expenditure are to be analyzed before the solutions in special designed hardware components are implemented.

The proposed kind of simulator is usable as a training simulator for driving schools or as a base for research (i.e. in ergonomic design of car cockpits).

Two types of special display generators are described: a Curve-/ Surface Generator, which is usable to render continuous curved objects and the Object Processor, which transforms complete textures in real-time.

TASKS AND REQUIREMENTS FOR A TRAFFIC SIMULATOR

The main task of a traffic simulator is the visualisation of highly realistic complex traffic situations, so that it can be used for the development of traffic guiding systems, research on traffic accidents, the development of ergonomic optimal car-manipulation equipment and finally for training-simulators.

Specially for the realization of training-simulators solutions at low cost are to be found, which make their use in driving schools possible.

The main difference between this CGI-system and a visual system for car driving simulators is the observer's freedom of movement in the model world and the scene complexity: The driver himself decides, whether or not to change his direction of movement (e.g. at road crossings). Independent controllable traffic guiding signals and detailed "faces" of buildings animate the scene.

The following requirements for the visual system of a traffic simulator exist:

- no restriction on the observer's movement in the modelled world,
- the possibility of visualisation of road crossings, traffic signs, signals and other objects,
- a minimum horizontal viewing angle of 150 degrees,
- visualisation of city traffic with other road users (especially other cars),
- simulation of special effects like fog,
- visualisation of car dynamics.

THE VISUAL SYSTEM

As seen in Fig. 1 the proposed CGI- system for the traffic simulator is divided into four main parts:

- model computer,
- control computer,
- display processor,
- simulation room.

The number of elements of the same kind in Fig. 1 (e.g. four screens) is no principal restriction. The concept was realized and tested with some representations of all shown components.

The four parts build a pipeline structure, where an action at the steering elements of the simulator-car causes a visual reaction on the screen. In parts of the visual system a layer structure was created, where computation on a great amount of data is to be done. Each layer contains a number of parallel working components so that the amount of data, which is to be processed, is cut into small usable pieces.

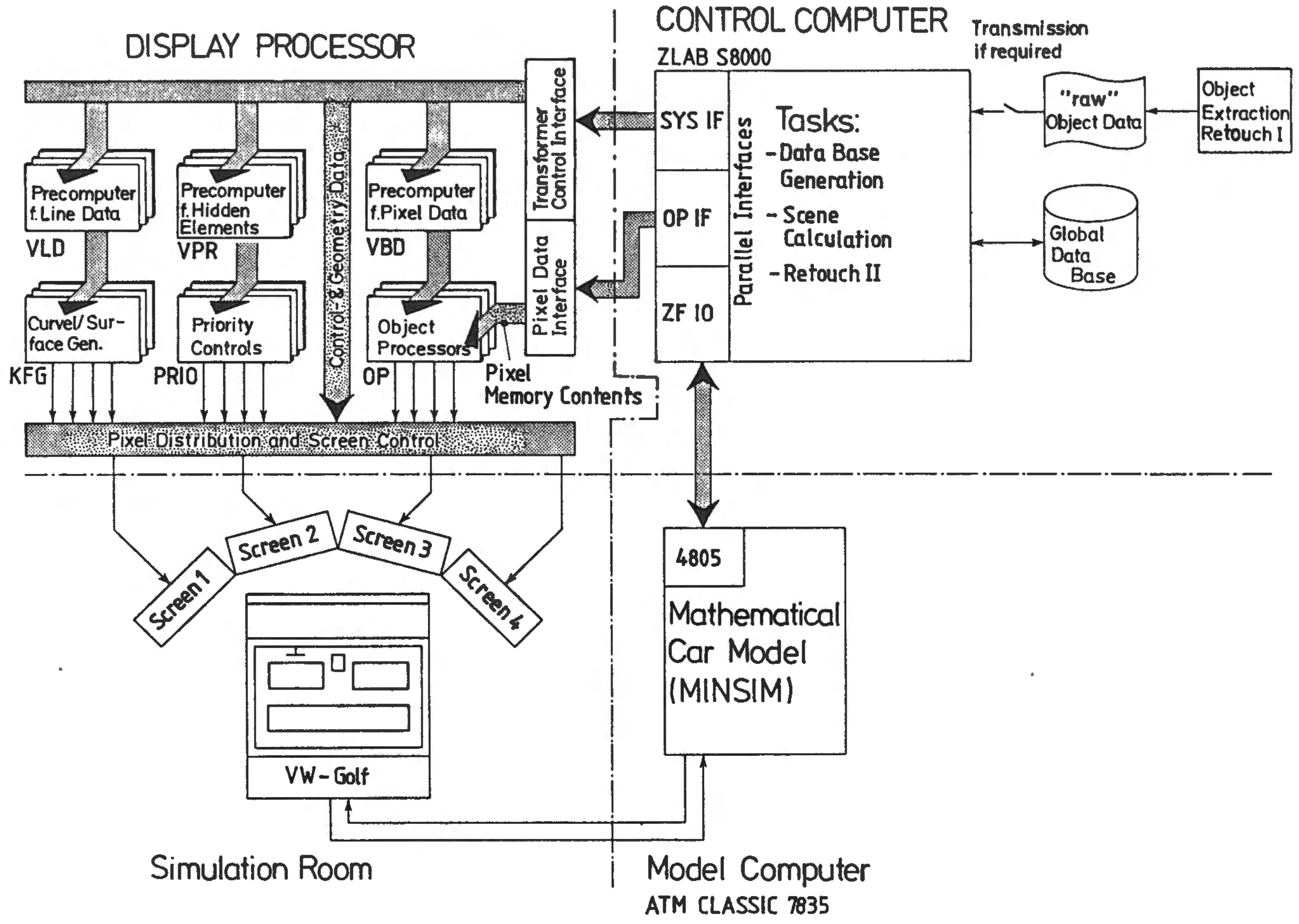

Figure 1: Visual System for a Traffic Simulator

Model Computer / Car Model

The model computer connects the control computer with the simulation car. Its main task is the calculation of the car model. The mathematical model MINSIM was worked out in a undergraduate study (Ganter 1982). It receives the output values of the physical car model in the simulation room (fourth part of Fig. 1) and calculates coordinates of the driver's position and angles of the drivers orientation in the simulator data base (Fig. 2).

The model, which made the verification of the realized visual system possible, consists of seven coupled second order differential equations. A detailed theoretical description of the model is given in Mitschke (1972).

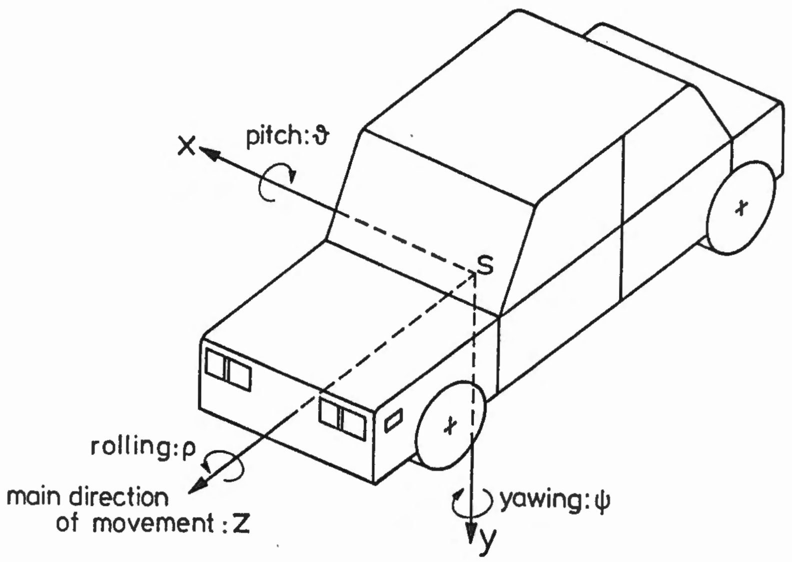

Figure 2: Degrees of Freedom of the Simulated Car

There is no tactile and acoustic, but only the visual reaction of the model implemented, and only the speed of the simulated car is shown on the speedometer. So the visual system must give all the necessary information to the driver to create the impression of driving a real car.

Control Computer

In the traffic simulator the control computer which generates and manipulates the scene database of the CGI-system is a main part. The scene database contains all the elements for the visual simulation, e.g.

- geometric data describing a cartographically reproduction of a road network including the primitive representation of road elements,
- 2D picture objects,
- 3D solid objects,
- all parameters and constants for the necessary transformations.

The control computer contains interactive programs for the scene generation (construction of the world model and application of the scene elements). The reproduction of a cartographically given landscape is possible, also the definition of traffic junctions, traffic guiding signals and road signs (Kuckel 1985).

The result of these operations is the data base, which will be partially distributed into the special modules of the display processor in the initial phase of the simulation. The driver of the simulated car may move to any point of the modelled world with six degrees of freedom but he can also be directed by road signs which are controlled by the scene calculation program during simulation.

Display Processor

The proposed concept requires parallel processing in different layers because of the amount of data to be administrated. Therefore precomputers (VR) have been developed based on a 16-bit-microprocessor. Their hardware is universal, but every VR has to solve one special problem like clipping, sorting or transformation.

Most of the tasks require high precision calculations and a wide range of numbers. So the precomputers were designed as multiprocessor-systems with one CPU and some special auxiliary processors(Skiba 1986). Because of the parallelism of many algorithms (e.g. 4x4 transformation) three floating point processors are implemented as standard auxiliary processors.

The precomputers use autonomous parallel interfaces for all of their data communications.

For the movement of 2D picture objects containing extremely detailed picture information <u>Object Processors</u> were developed (Köster 1985), which obtain their transformation data from a precomputer (here called VBD) while pixel data (picture memory contents) are loaded via a special direct interface from the control computer.

The picture objects defined by line data are to be divided into two categories:

- polygonal (vector-) defined objects and
- continuous curved objects.

Rendering a building requires only a few straight vectors (edges), but a continuous stretch of road with variable curvature is only renderable by a normally nonconvex polygon with many edges.

So for the special task of continuous curved objects Curve-/ Surface Generators (KFG) have been designed which are also able to generate polygonally defined objects on the screen (Niechciol 1986).

The object processors and the curve-/surface generators work in real time, that means, they generate all the necessary pixel information while the electron beam of the CRT is running (calculations must be finished before the beam has reached the actual display coordinate).

The hidden elements calculations are an important part of the visual system. Here some restrictions are very helpful:

- all 3D scene objects are opaque,
- all scene objects are solid (no penetration),
- all objects are joined in the order of principal visibility.

Because the objects are opaque and no penetration is allowed, the hidden object calculations become very simple (e.g. object 1 hides object 2). A visibility chain accelerates the speed of the hidden object calculations.

The object-dependent hidden lines calculations are to be presolved by procedural methods whenever it is possible. That means, priority chains of the parts of every object are defined for all viewpoints of the observer relative to the viewed objects (an example is realized in Rood (1986).

The pixel distribution- and screen control combines all pixel data in real time. This main element of the display processor is able to support a maximum of 7 display channels with the same number of screens. The viewing angle is adjustable by special parameters in the visual system. In the realized prototype version a horizontal field of view of about 150 degree is chosen. Angles less than 150^{o} would complicate the task of turning into another road at crossings.

The geometrical problem to connect screens as parts of a large display is solved: the proposed system does all transformations individually for any display channel, with immediate connections to their neighbour channels. Another problem, the loss of intensity and colour differences near the edges between screens of a multichannel display, is to be solved by analog methods.

For special effects like fog or driving at night experimental electronic circuits have been developed, an implementation into the visual system later on is possible.

MODULAR CONCEPT

For the display processor a concept of parallel working modules was developed. In the following the terms Module and Transformer have to be differentiated:

Transformer for Line Data

The transformer for line data consists of a precomputer for line data (VLD) and n curve-/surface generators (KFGs). For the realized visual system n_max = 16 is chosen.

Object Transformer

The object transformer consists of a precomputer for pixel data (VBD) and m object processors (OPs). The realized precomputer VBD supplies m_max = 5 OPs with transformation data in realtime (20ms).

Module

The Module is the leading representation of a displayable object (e.g. house or an other than the viewer's car) or a function control (e.g. priority module = VPR and priority control). A module consists of one or more transformers. It is possible to combine object transformers and transformers for line data in one module.

Different methods for realizing Transformers have been developed:

- problem oriented methods (special algorithms for a special problem),
- sequential methods (e.g. use of recursive algorithms for rendering smooth curves),
- procedural methods (every perspective of an object is rendered by a special procedure).

The modular concept allows an object bound, parallel transformation and the rendering of all scenic elements in a given time range of 20ms. For the prototype version of the traffic simulator the modules "horizon", "road", "mean line", "border line", "crossing" and "other car" have been realized (Fig. 3). With adaptation of a module "2D-Object" all required scenic elements can be realized: Faces of buildings (2D -Objects), moving or parking vehicles with lights and signals, the road network and traffic signals.

The described visual system for a traffic simulator is modular. Relevant objects can be made obtainable for simulation with a scene generation program. They are realized with the implementation of a new module in the display processor.

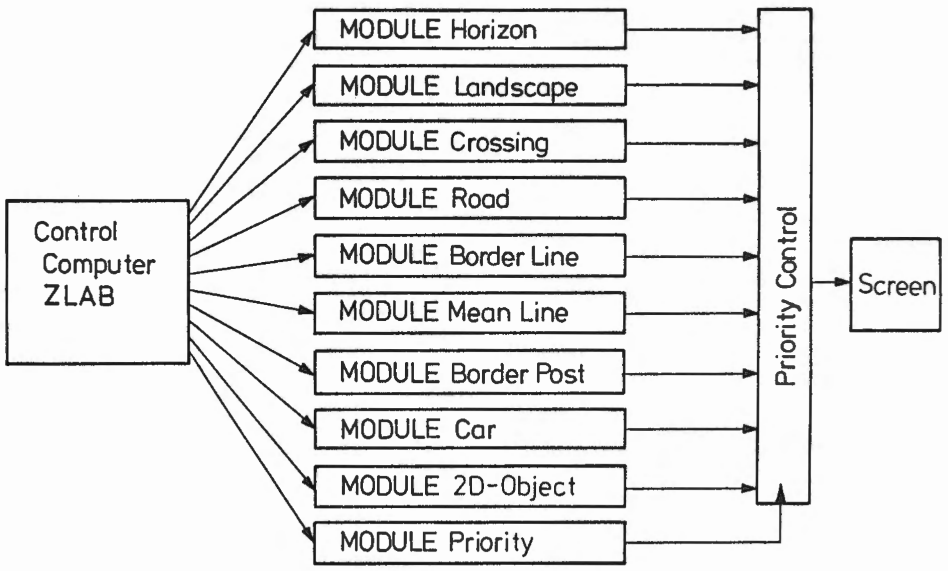

Figure 3: Modular Concept of a Visual System for Traffic Simulation (Example)

SYNCHRONIZATION

The whole visual system is synchronized by signals of the display processor. So the time lag between action at the steering elements of the car model and the reaction with a synthetic image amounts to 60ms. This is shown in Fig. 4.

The calculation of the car model MINSIM is done in a frame time of 10ms, that means, that a changing input value of the car model is interpreted after a maximum of 10ms. The necessary time for the solution of the model equations in MINSIM is about 7ms, then the output of the calculated data to the control computer follows (DÜ1).

The control computer is interrupted periodically by the described synchronization signals (t_p =20ms). The data from the car model are in the control computer at this time and will now be interpreted. The "dynamic" data (data changing every simulation cycle t_p, e.g. position of the observer and the orientation transformation matrix T_{WB}) are transmitted directly to the transformers in the display processor (DÜ2).

After calculating the potential visible scene elements for the next simulation cycle the results (position transformation matrices and control data) are transmitted as "static" data to the display processor. The time limit for these scene calculations is set to $t_{scene} <= 0.9\ t_p$ (DÜ2).

The transformers in the display processor begin their calculations immediately after receiving the "dynamic" data (in Fig. 4: VBD and VLD) und finish before the second raster is drawn on the screen (interlaced mode assumed). The time boundary is: $t_{Trans} <= 0.9*t_p$. The remaining time is used for transmissions (DÜ3) and for organization tasks following "static" data.

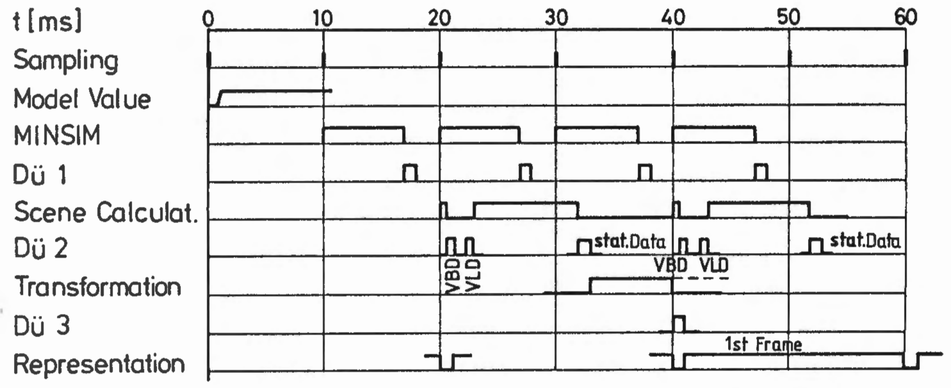

Figure 4: Simulation Timing

It is shown in Fig. 4, that the first raster following an action in the model car is drawn after 60ms, if the time boundary is not hurted.

A mean simulation cycle of $4*t_c$ is possible if the control computer and the model computer are combined.

OBJECT PROCESSOR

The transformation of extremely detailed objects, e.g. textures with 512 by 512 different coloured pixels, is difficult to do in real time. It requires a large amount of special and expensive hardware if the common way of transforming object coordinates into display coordinates is chosen.

In the proposed Object Processor a reverse way of transformation was realized: The given display coordinates are back - projected into a 3D - viewer´s coordinate system and from there back - transformed and projected into a pixel memory coordinate system, which describes a picture object stored in the memory of the Object Processor.

This leads to the Projection Equation:

$$\boxed{\begin{aligned} x_S &= X / Z + d_x \\ y_S &= Y / Z + d_y. \end{aligned}}$$

In these equations X, Y and Z are some functions of the display coordinates x_D and y_D, d_x and d_y translation coefficients and x_S and y_S the pixel memory coordinates.

The Projection Equation consists of six multiplications, two divisions and eight additions (all of them single precision floating point operations), together to be solved in less than 70 ns.

Because of the counter characteristics of the display coordinates the multiplications could be replaced by sequential additions. The parallelisms found in the Projection Equation led to the design of the Object Processor as shown in Fig. 5.

The controlling element of the Object Processor is a common signal processor. With it the non-timecritical calculations and operations are done, while a scan line is being drawn. It controls two interfaces: the transformation coefficients interface to the precomputer VBD and the pixel data interface, with which pixel data (picture memory contents) are loaded directly from the control computer.

The calculation of $X(x_D,y_D)$, $Y(x_D,y_D)$ and $Z(x_D,y_D)$ is done with three parallel working arithmetic-units, which deliver their solutions to two floating point dividers.

Today one-step dividers are not commonly available, so that a solution with special exponent-ALU, barrel shifters and a parallel multiplier unit was designed.

The dividers deliver the required pixel memory coordinates which are usable as addresses for the pixel memory. A following memory resolution control is used for the reduction of pixel flickering, which is a special aliasing effect of moving picture - objects.

The pixel memory of one Object Processor amounts to 512 by 512 pixels, coded with 12 bit colour information. This is a useful resolution, where unscaled picture objects cover about 2/3 of the screen.

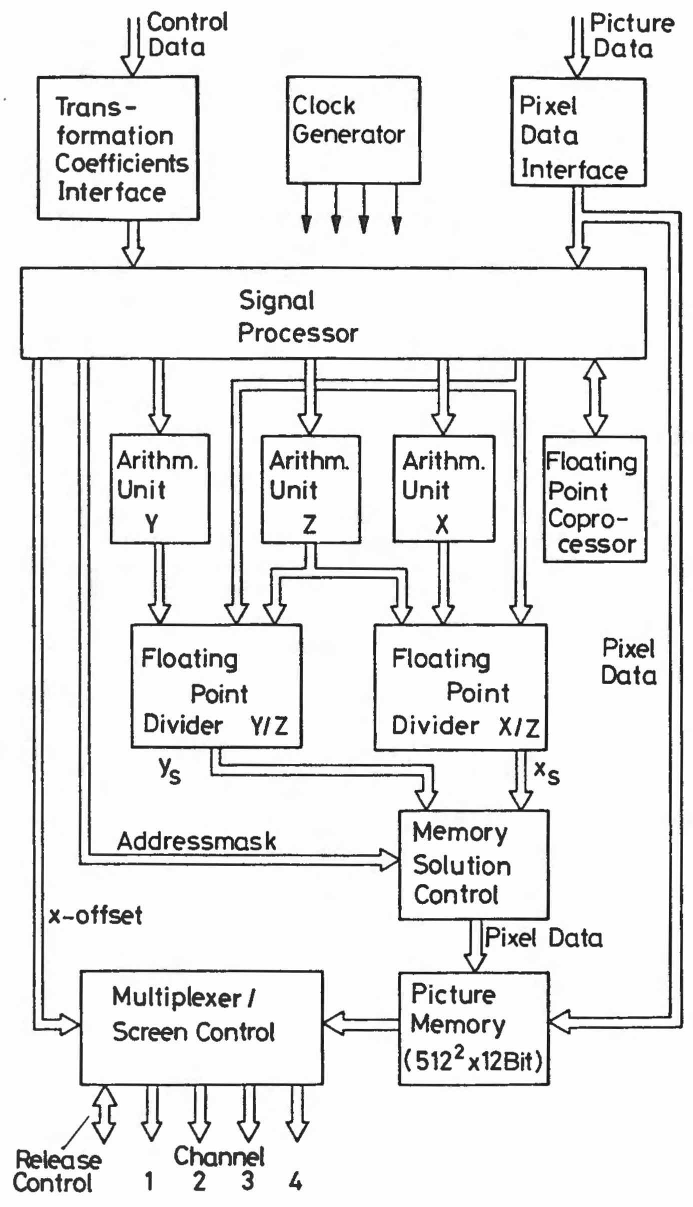

Figure 5: Block Diagram of the Object Processor

CURVE-/SURFACE GENERATOR

The Curve-/Surface Generator was designed for rendering 3D-Objects. These objects in the traffic simulator have rather monochrome surfaces and are not extremely detailed, but have a continuous curvature.

For the line- data modules special software solutions exist, which generate points, lines (vectors), polygons or slopes. The Curve-/ Surface Generators are programmable to manage all these input data because they contain a common signal processor as a controlling unit like the Object Processor does. The structure of the whole generator is shown in Fig. 6.

The signalprocessor receives its information via a ringbuffer from the allocated precomputer. This ringbuffer is read out every frame cycle and if there is no change in the stored information an idle picture appears on the screen.

For the real time generation of picture elements all input data are converted for use in a digital differential algorithm (DDA), which is multiply implemented in the arithmetic units. The output values of the DDAs are compared with the display coordinates of the used screen and a following priority logic performs a display request to the screen control of the display processor.
If the request is acknowledged, the colour control will be freed and sends the colour information for one pixel to the screen control.

The operation time for the above described scan conversion procedure may be about 60 µs maximum. Relevant pixels are generated in real time, that means about 60 ns.

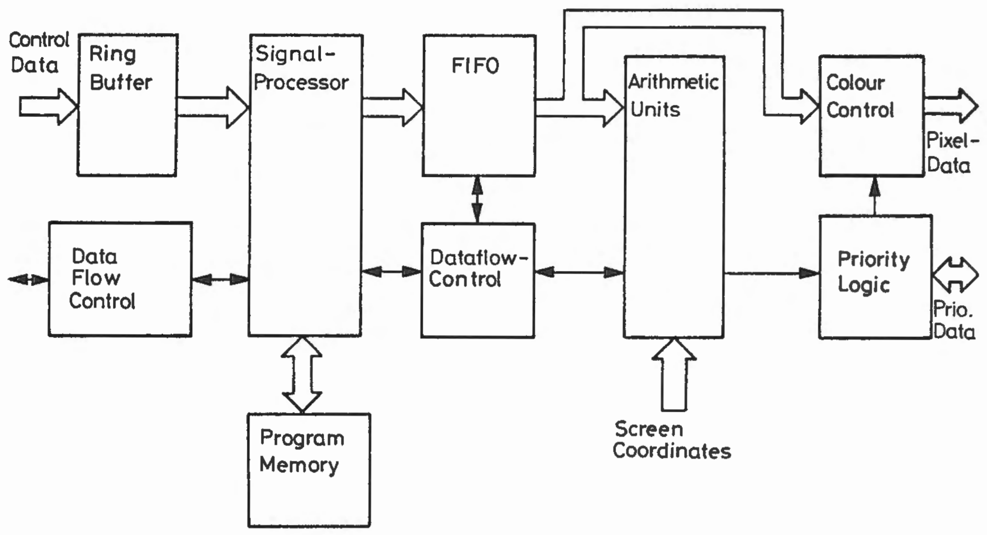

Figure 6: Block Diagram of a Curve-/ Surface Generator

CONCLUSION

A modular visual system for a traffic simulator has been proposed. It was shown, that not only special hardware has to be developed, if a solution for a special graphics problem without immense costs is required, but also the combination of intelligently constructed algorithms - with respect to their parallelisms - and the necessary hardware is advantageous.

While working on the simulator problems a lot of logic has been found, that can be replaced by gate arrays or custom VLSI-chips. This and the intelligent-programming concept makes the proposed modular system applicable for low cost training simulators.

A powerful tool for transformation of planar textures has been shown. For an additional advance in terms of reality it would be necessary, to solve the special problem of rendering curved textures. So that is one of the next problems the author intends to work on.

ACKNOWLEDGEMENTS

I would like to thank Prof. J. Heidepriem and all other members of the graphics group at the Institute of Automation Techniques for the many fruitful discussions and their active help in the successful realization of the concept of our visual system.

The work on the visual system for a traffic simulator was supported by Volkswagenwerk AG in Wolfsburg (FRG).

REFERENCES

Ganter M, (1982) Realzeit- Simulation eines Pkws. Studienarbeit am Lehrstuhl I für Automatisierungstechnik im Fachbereich Elektrotechnik der BUGH Wuppertal

Köster A, (1985) Prozessor zur 3D-Transformation zweidimensionaler Bildpunktfelder. Studienarbeit am Lehrstuhl I für Automatisierungstechnik im Fachbereich Elektrotechnik der BUGH Wuppertal

Kuckel K, (1985) Implementation neuer Datenstrukturen in die Szenenrechnung eines Verkehrssimulators. Diplomarbeit am Lehrstuhl I für Automatisierungstechnik im Fachbereich Elektrotechnik der BUGH Wuppertal

Mitschke M, (1972) Dynamik der Kraftfahrzeuge. Springer-Verlag, New York

Niechciol V, (1986) Generatoren für die Echtzeitdarstellung nichtkonvexer Flächen mit quasikontinuierlichen Berandungskurven auf Rasterbildschirmen. Diplomarbeit am Lehrstuhl I für Automatisierungstechnik im Fachbereich Elektrotechnik der BUGH Wuppertal

Rood HA, (1986) Entwicklung eines Bildmoduls auf der Basis prozeduraler Beschreibungen. Studienarbeit am Lehrstuhl I für Automatisierungstechnik im Fachb. Elektrotechnik der BUGH Wuppertal

Schachter BJ (ed.) (1983) Computer Image Generation. John Wiley & Sons, New York

Skiba T (1986) Implementation von Floating-Point-Prozessoren in das Mikrorechnerkonzept eines Sichtsimulators. Studienarbeit am Lehrstuhl I für Automatisierungstechnik im Fachbereich Elektrotechnik der BUGH Wuppertal

Yan JK (1985) Advances in Computer-Generated Imagery for Flight Simulation. IEEE Computer Graphics & Applications 5 (No.8): pp.37-51