2010 12th International Conference on Computer Modelling and Simulation

Modular Technology in the modelling of large virtual environments in driving simulators.

Carlota Tovar; Ginés Jesús Jimena ; José Mª Cabanellas; Carlos Zoido.
Departamento de Ingeniería Mecánica y Fabricación.
CITEF Research Centre on Railway Technologies.
Universidad Politécnica de Madrid.
Madrid, Spain.
ctovar@etsii.upm.es; citef-gjimena@etsii.upm.es; jmcabanellas@etsii.upm.es; czoido@etsii.upm.es.

Abstract— **This paper presents the latest research and developments in Modular Technology. That is, the optimized repetition of the same geometry or module, for the generation of large virtual environments for the simulators that are designed by CITEF. The current trend is on redirecting the maximum possible share of graphical calculation to the GPU to lighten the load of the CPU as far as possible, as this is slower and less suitable for graphic calculation. Modular Technology, which has been successfully used for some time in driving training simulators, is particularly suitable for the implementation of operations in the GPU through programming shaders. In this way a substantial reduction in the number of geographical entities in the environment can be attained and an increase in the diversity and flexibility of the environment. This is achieved by discretizing the environment into a series of instantiated modules and using shaders in an appropriate manner based only on a few parameters. There are many advantages to be had from this form of generation: savings in scene size, loading times and resource requirements, greater flexibility and clarity of the scene graph and a more substantial upgrade capacity of the environment.**

Keywords-component; geometric modelling; virtual environments; driving simulators; shaders.

I. INTRODUCTION

The use of virtual driving simulators as a learning and training tool is a strongly established procedure that is becoming more and more widespread with a whole range of possibilities going from high and medium to low-cost. All this is a result of the rapid progress in hardware which has led to like increases in the demands for realism in virtual display scenes.

The high processing capacity of present-day GPUs means that the CPU can be freed of tasks, which is vitally important in terrain driving simulation. The geometric load to be rendered can be increased making the CPU-GPU communication bus the new bottle-neck.

In response to this change of priorities in scene optimisation, a new generation of algorithms has begun to appear to adapt the level of detail (lods) whose work primitive has ceased to be the triangle [1]. Blocks of triangles (batches), are used instead. These are optimised in pre-load time to ensure optimal spatial organisation (stripping) thereby speeding up the CPU-GPU transfer.

Following this latest line of research, Modular Technology, a methodology used for building virtual environments in CITEF (Research Centre on Railway Technologies), has converted the module into its base primitive. After searching for some repetitive patterns in the environment, geometric as well as functional, Modular Technology discretizes the scene into a finite number of modules or portions of the environment, which after being instantiated and subjected to a series of geometric transformations using shaders, are meticulously assembled to reproduce the virtual scene.

II. BUILDING LARGE ENVIRONMENTS IN PRESENT-DAY TERRAIN DRIVING SIMULATORS

By constructive methodology of a virtual environment we mean a set of procedures that allow interpreting, processing, analysing, modelling, optimising and virtually displaying certain initial information.

When it comes to choosing the most ideal methodology for generating large virtual environments in terrain driving simulators, it is imperative to choose the data structures that will respond to the constraints imposed by the environment to be displayed and minimise CPU consumption and maximise the resources offered by current GPUs.

In the case of overland driving simulations, the kind of paths to be driven over will be a decisive factor in the choice of the most ideal types of data structures. Depending on this, two environment types can be differentiated:

- **Undetermined route environments**.

This is the case with combat simulations in military training. Since the paths do not impose any geometric constraint, the data structures are more flexible (regular or irregular) and the variety of algorithms for creating multiresolution models is very wide. In this field, the latest trends, as stated in the introduction, point to lightening the CPU load and speeding up CPU-GPU transfer using the new work component: the batch [2][3]. The batches or tiles are constructed in the pre-load time optimising to the maximum the organisation of the triangles they contain (strips).Thanks to the batch the selection metrics are no longer evaluated at a vertex level, but are performed at a batch level, enormously reducing the load of the CPU. Although this means the number of polygons to be displayed is less optimised, the high processing capacity of current GPUs means this is not a problem.

- **Pre-determined route environments**.

This is the case with the railway and urban driving simulation dealt with in this paper. The enormous realism required for the path surroundings can only be achieved through the geometric insertion of the paths into the landscape. This has large repercussions on the choice of data structures to be used. Inserting this geometry involves a readjustment and a retriangulation of the terrain mesh in the geometry surroundings [4], so that it can be adapted to the highest path resolution. This supposes a notable increase in the number of calculations to be performed, but also an ensuing increase in the main drawback posed by the continuous levels of detail adaptation algorithms (continuous lods) and the heavy use of the CPU. Discrete levels of detail [5] a [7] therefore, become the preferred alternative, their main drawbacks being:

- The high memory consumption required to store the geometry associated with the different levels of detail during the pre-load time.
- Little constructive flexibility.
- Failure to make full use of the functionalities offered by present-day GPUs.

Modular Technology has arisen to meet the latest needs. Modular Technology uses route tracking as the starting point of its work from which it generates the surrounding environment using the assembly and repetition of a finite number of modules. It thus exploits one of the basic tools for resource optimisation: instantiation. Calculating the number and optimum geometric design of the modules as well as the variety of their families will be decisive in obtaining the realism and refresh speed required in these simulations. It also takes advantage of the priori knowledge of these paths to generate discrete pseudo-variant levels of detail with the view point. This is achieved by discretizing the environment transversally to the path.

The outcome is a scene with all the realism required by driving simulations with the desired refresh speeds, with a comfortable communication interface with the simulation module and graphic engine. All this is done automatically and with the possibility of easily inserting any environmental restructuring, something that is essential in this kind of virtual display. Moreover, the main problems associated up to now with the discrete level of detail adaptation algorithms (discrete lods) are eliminated:

- Thanks to instantiation and the shaders a straight module of each family need only be stored during the pre-load time, which means that memory ceases to be a problem.
- The scene is synthesised from a set of repetitive parameters, which means modelling is simplified.
- The modules are subjected to geometric transformations using shaders during execution time which lets the realism of these scenes be enhanced. These transformations will be defined during the pre-load time by means of a set of parameters integrated next to the scene graph.
- The modularity of the environment endows it with high constructive flexibility with a high range of virtual scenes being able to be generated with a minimum of constructive parameters.

III. Modular Technology

Modular Technology synthesises the construction of the environment to calculate a set of transversal profiles and a set of guidelines that will act as extrusion axes for these profiles. To define these profiles repetitive patterns will be sought so that the environment can be defined by taking a set of lofts, which from now on we shall call modules, which correctly instantiated and positioned reproduce the scene.

Therefore, the pillars on which Modular Technology is built are: the module, which is its work primitive, and the guidelines, both of which will be taken for calculating the geometric site for inserting the modules.

However, finding a guideline for the environment is not always possible. This has led Modular Technology to classifying the environment into two types of zones:

- **Modular zones:**

These are generated by the assembly and repetition of a finite number of modules. As already stated, the modules are portions of environment which, correctly positioned and hierarchised in a scene graph, let this environment be partially or totally virtually displayed.

The Figure 1. shows an example of how these modules are used for constructing virtual environments.

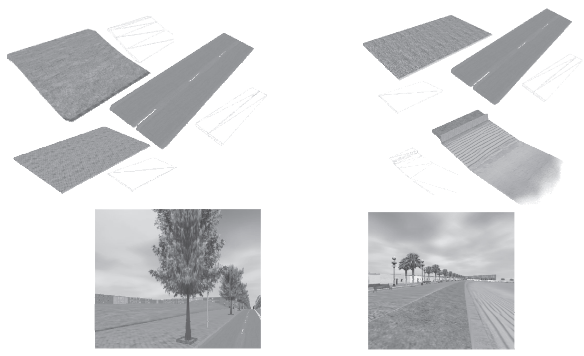

Figure 1. Example of module usage.

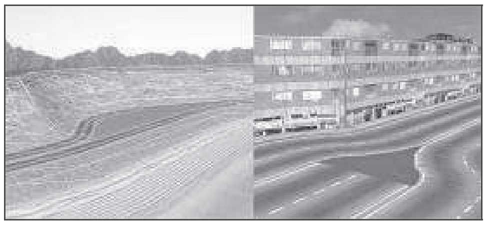

Figure 2. Example of the combined use of modules and meshes.

- **Meshable zones.**

It is not always possible to assimilate an environmental fragment to a set of repeated elements. The existence of irregular geometries where it is not possible to establish a guideline impedes the use of this technology, so these zones are modelled using meshes. Figure 2. shows examples of zones where modules are not applicable and can only be done by using meshes.

IV. The Module And The Module Family

The module originates from a basic design made up of a straight longitudinal portion of the environment. This basic design is curved into a set of degrees forming what is called a basic set of modular components or modules (Figure 3.). In this way the basic set is characterised by the existence of a finite number of modules of fixed curvature and discrete curvature values.

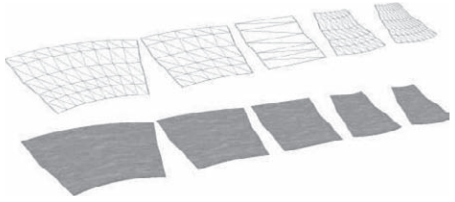

Figure 3. Basic set of modules.

The module comprises three basic components: a longitudinal path and a set of transversal profiles with their corresponding textures (Figure 4.).

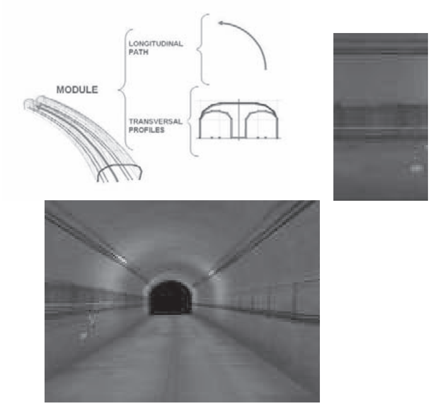

Figure 4. Components of a tunnel module.

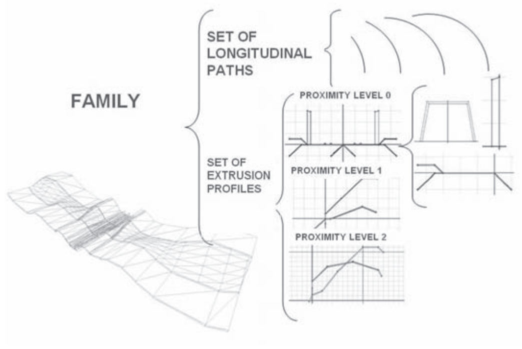

Figure 5. Components of a family.

The base set of modules together with a suitable positioning algorithm allows reproducing any geometry without holes or overlaps.

The next problem arises when attempting to calculate the transversal profiles of these modules so that they can reproduce any environment. So, the concept of a family of modules now appears as the base set of modules and shares the same definition of transversal profiles (Figure 5.). These families will verify relationships of compatibility, which will be what ensures there are no incoherencies or discontinuities after the modules have been correctly positioned.

In order to ensure these families are correctly positioned, the guidelines are divided into intervals. Each interval is associated with a different family type.

V. Modular Positioning Algorithm

There are several factors to be borne in mind if maximum benefit is to be had from this system of instantiation. The modular positioning algorithm is responsible for this, which means it has to solve the following issues [8]:

- The optimum number of modules, responsible for a higher or lower graphic load and a better or worse scene graph path optimisation.
- The type of module to be placed at each point of the scene: if the module curvature is wrong, spatial and tangential discontinuities are produced. Figure 6. shows the visual appearance resulting from inserting a rail module with a greater or lesser curvature than appropriate.

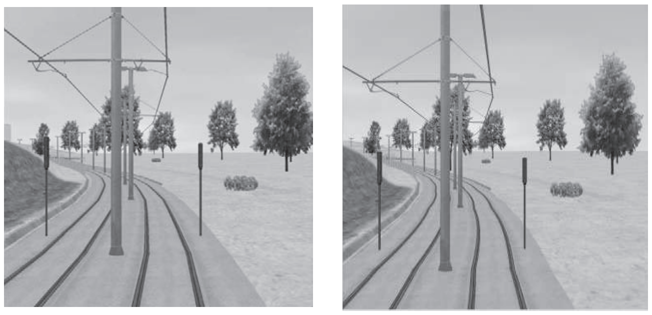

Figure 6. Undercurved and overcurved modules

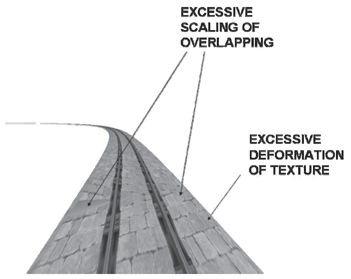
EXCESSIVE SCALING OF OVERLAPPING
EXCESSIVE DEFORMATION OF TEXTURE

Figure 7. Excessive scaling of modules.

- Scaling each module. The modules can be subjected to an X, Y and/or Z scale. The maximum scale admitted will vary depending on the length of the module (basic length, L_B). Exceeding this scale leads to an excessive deformation of the overlapping and texture. On the other hand, excessive scaling leads to a noticeable variation in module curvature that results in defective module coupling. These scaling tolerances can only be overcome by shaders to correct these distortions. In these circumstances Modular Technology will calculate these deformation parameters through the graphic engine, which will be responsible for managing the execution time. Figure 7. shows the visual appearance of a path made up of modules subjected to excessive scaling.
- The positioning coordinates of each module. The geometric positioning site for modules is called a polygon site and consists of a set of points obtained by interpolation on the said biarc. The modules are positioned at the mid-point of their left end and orient their chord in line with the polygon segment on which they are positioned, taking in the angle of arc included between both ends of its chord. Thus the modules circumscribe the guideline as can be seen in the following Figure 8. Figure 9. shows how a set of railway platform modules is put in place in accordance with the positioning algorithm. Each module appears on the left in a different colour so that the correct coupling between them can be more clearly seen since in the right-hand image the perfection of the coupling makes it impossible to discern this joint.

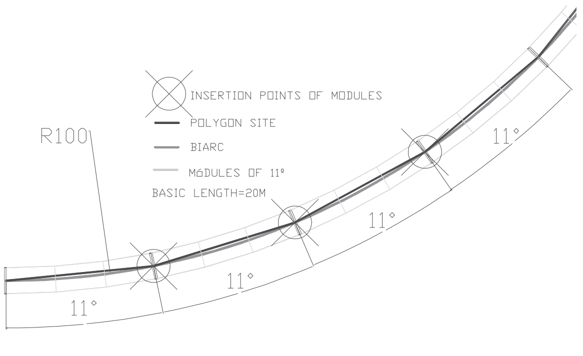

Figure 8. Positioning algorithm.

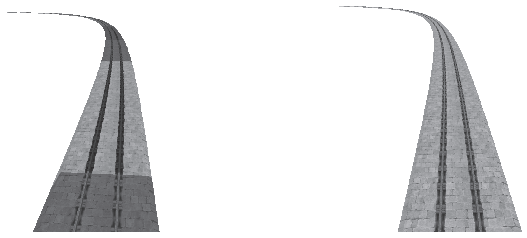

Figure 9. Railway platform and rail modules positioned according to the positioning algorithm.

Determining the module to be inserted at each point of the polygon from those that the geometric discretization has deemed to be a basic set of modules, will be determined by the biarc angle of embrace between the two points of the polygon:

$$\alpha = \frac{L_{arc}}{R}$$

where L_{arc} is the length of the mid-line of the module and R the radius of the biarc.

The longitudinal scaling to which the module will be subjected is:

$$e = \frac{L_{chord}}{L_{basic}}$$

where L_{basic} is the basic length taken for constructing the module family and L_{chord} the polygon segment length or module chord length.

VI. Modular Technology And Shaders

The use of shaders lets each modular family be formed during the pre-load time by a single straight module. In execution time the shader will take charge of deforming this module until it attains the desired geometry. Thus, modelling effort is reduced as well as initial load times and the modular positioning algorithm is made more versatile since the shader is capable of deforming the start and end sections of each module so that they fit perfectly without any spatial or tangential discontinuities. As a result, each family goes on to be formed in execution time by infinite modular components.

Modular Technology starts out from a straight module of basic length L_B. This length is measured in the direction of the axis i_1, as Figure 10. shows.

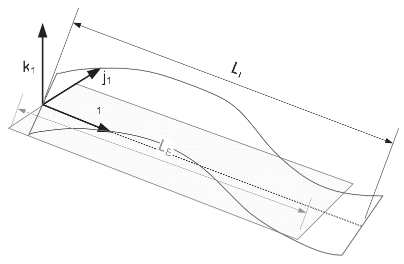

Figure 10. Straight module deformed by the shader.

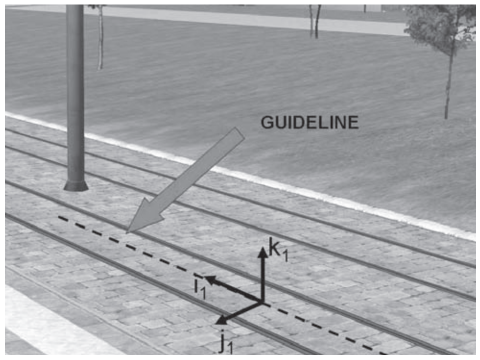

Figure 11. Positioning the module in the environment.

The axis i_1 in turn, coincides with the direction marked by the guideline that positions it, while its direction is marked by the increasing pks, as Figure 11. shows.

Using the composite transformation shader, Modular Technology seeks to subject the module to a deformation that will guarantee perfect coupling with the adjacent modules after being positioned by the modular positioning algorithms.

To do this, firstly a dimensionless parameter s must be defined as the relationship between the x_0 coordinate and the basic length L_B. The parameter s will vary between 0 and 1.

$$s = \frac{x_0}{L_B}$$
$$0 \le s \le 1$$

What is sought is to define a function that will ensure zero displacement at the ends of the module and whose tangent angles coincide with those required (those defined by modular positioning algorithm).

In this way, the function calculating the displacement at y to which the module points have to be subjected, should verify:

$$f_{\Delta y}(s) \equiv \begin{cases} f_{\Delta y}(0) = 0 \\ f_{\Delta y}(1) = 0 \\ \dfrac{df_{\Delta y}}{dx_R}(s=0) = \tan\gamma_{ini} \\ \dfrac{df_{\Delta y}}{dx_R}(s=1) = \tan\gamma_{fin} \end{cases} \tag{1}$$

In the above equation x_R represents the actual module coordinate after scaling. The relation with x_0, the module path coordinate before deformation and with s is:

$$\begin{gathered} x_R = x_0 \cdot E_x \\ \frac{dx_R}{dx_0} = E_x \\ \frac{dx_0}{ds} = L_B \\ \frac{dx_R}{ds} = L_B \cdot E_x = L_i \end{gathered} \tag{2}$$

The group of equations (1) bearing in mind (2) may be re-written as:

$$f_{\Delta y}(s) \equiv \begin{cases} f_{\Delta y}(0) = 0 \\ f_{\Delta y}(1) = 0 \\ \dfrac{df_{\Delta y}}{dx_R}(s=0) = \dfrac{df_{\Delta y}}{dx_R}\dfrac{ds}{ds}(s=0) = \dfrac{df_{\Delta y}}{ds}\dfrac{ds}{dx_R}(s=0) = \tan\gamma_{ini} \\ \dfrac{df_{\Delta y}}{ds}(s=0) = L_i \cdot \tan\gamma_{ini} \\ \dfrac{df_{\Delta y}}{ds}(s=1) = L_i \cdot \tan\gamma_{fin} \end{cases} \tag{3}$$

The functions f_1 and f_2 have the following properties:

$$\begin{gathered} f_1(s) = s \cdot (1-s)^2 \quad f_2(s) = s^2 \cdot (1-s) \\ f_1(0) = 0 \quad f_2(0) = 0 \\ f_1(1) = 0 \quad f_2(1) = 0 \\ \frac{df_1}{ds}(s) = (1-s)(1-3s) \quad \frac{df_2}{ds}(s) = s \cdot (2-3s) \\ \frac{df_1}{ds}(0) = 1 \quad \frac{df_2}{ds}(0) = 0 \\ \frac{df_1}{ds}(1) = 0 \quad \frac{df_2}{ds}(0) = -1 \end{gathered} \tag{4}$$

By correctly putting together the requirements of (3) and the results of (4) a suitable function for $\mathbf{f_{\Delta y}}$ is obtained:

$$\begin{aligned} & f_{\Delta y}(s) = f_1(s) \cdot L_i \cdot \tan\gamma_{ini} - f_2(s) \cdot L_i \cdot \tan\gamma_{fin} \\ & f_{\Delta y}(0) = 0 \\ & f_{\Delta y}(1) = 0 \\ & \frac{df_{\Delta y}}{ds}(0) = L_i \cdot \tan\gamma_{ini} \\ & \frac{df_{\Delta y}}{ds}(1) = L_i \cdot \tan\gamma_{fin} \end{aligned}$$

For the increases in z, $\mathbf{f_{\Delta z}}$ the conditions for (3) will need to be appropriately adapted:

$$f_{\Delta z}(s) \equiv \begin{cases} f_{\Delta z}(0) = 0 \\ f_{\Delta z}(1) = 0 \\ \dfrac{df_{\Delta z}}{ds}(s=0) = -\dfrac{L_i \tan \beta_{ini}}{\cos \gamma_{ini}} \\ \dfrac{df_{\Delta z}}{ds}(s=1) = -\dfrac{L_i \tan \beta_{fin}}{\cos \gamma_{fin}} \end{cases}$$

Where the sign is negative it indicates that rotation β is in the opposite direction to the derivative. The displacement function is finally given as:

$$f_{\Delta z}(s) = \frac{-f_1(s) \cdot L_i \tan \beta_{ini} + f_2(s) \cdot L_i \tan \beta_{fin}}{\cos(\gamma(s))}$$

There are no deformations through deformation in the direction of x, except the scaling.

The rotations α(s), β(s) and γ(s) that are coherent with the displacement functions described are obtained by deriving. A simple reflection shows us that the derivative in respect of x_R of the displacement function at z corrected by the cosγ also gives the angle β(s). The slope angle is a linear extrapolation of the initial and final slopes.

$$f_{\Delta y}(s) = f_1(s) \cdot L_i \tan \gamma_{ini} - f_2(s) \cdot L_i \tan \gamma_{fin}$$
$$\frac{df_{\Delta y}}{ds}(s) = (1-s)(1-3s) L_i \tan \gamma_{ini} - s(2-3s) L_i \tan \gamma_{fin}$$
$$\frac{df_{\Delta y}}{dx_R} = \tan(\gamma(s)) = \frac{df_{\Delta y}}{ds}\frac{ds}{dx_R} = (1-s)(1-3s)\tan \gamma_{ini} - s(2-3s)\tan \gamma_{fin} \Rightarrow$$
$$\gamma(s) = \arctan((1-s)(1-3s)\tan \gamma_{ini} - s(2-3s)\tan \gamma_{fin})$$
$$\gamma(0) = \gamma_{ini} \quad \gamma(1) = \gamma_{fin}$$

For the angle β(s):

$$\frac{df_{\Delta z}}{dx_R} = -\frac{\tan(\beta(s))}{\cos(\gamma(s))} = -\frac{df_{\Delta z}}{ds}\frac{ds}{dx_R} = \frac{(1-s)(1-3s)\tan \beta_{ini} - s(2-3s)\tan \beta_{fin}}{\cos(\gamma(s))} \Rightarrow$$
$$\beta(s) = \arctan((1-s)(1-3s)\tan \beta_{ini} - s(2-3s)\tan \beta_{fin})$$
$$\beta(0) = \beta_{ini} \quad \beta(1) = \beta_{fin}$$

For the angle α(s) a linear interpolation is used:

$$\alpha(s) = (1-s) \cdot \alpha_{ini} + s \cdot \alpha_{fin}$$

VII. Conclusions

Ten years of successful results in the generation of virtual environments for renowned driving simulators have proved the validity of Modular Technology. Applying shaders increases the possibilities of modularity, reducing memory consumption, loading times and modelling efforts and increasing the flexibility in the construction of these environments. With ever more powerful hardware the quality of virtual environments will increase spectacularly in the near future.

Future developments will be to apply geometric shaders that create the full modular geometry in execution time so that the scene is inserted parametrically in line with the profiles and paths with only the most singular and least modular geometry being loaded. Real-time generation would therefore be adaptive. The greater the capacity for geometric calculation, the greater the complexity and quality of the environment.

References

[1] *E. Gobbetti.; F.Marton ;P. Cignon, C-BDAM - Compressed Batched* Dynamic Adaptive Meshes for Terrain Rendering. Computer Graphics Forum, 25(3), Proc. Eurographics 2006.

[2] Y. Livny; Z. Kogan; J. El-Sana, Seamless Patches for GPU-Based Terrain Rendering. WSCG 2007.

[3] R. Pajarola and E. Gobbetti., Survey on Semi-Regular Multiresolution Models for Interactive Terrain Rendering. The Visual Computer, 23(8) 2007, 583-605.

[4] L. P. Chew, Constrained Delaunay triangulations. In Proceedings of the Third Annual Symposium on Computational Geometry (Waterloo, Ontario, Canada, June 08 - 10, 1987). D. Soule, Ed. SCG '87. ACM Press, New York, NY, 215-222.

[5] Y. Papelis; O. Ahmad; G. Watson, Scenario Definition and Control for the National Advanced Driving Simulator. In Proceedings of the Driving Simulation Conference North America, Dearborn, Michigan 2003.

[6] P. Suresh and Ronald. R. Mourant, A Tile Manager for Deploying Scenarios in Virtual Driving Environments". Proceedings of the Driving Simulation Conference North America, December, 2005.

[7] V. Govil and Ronald R. Mourant, A Tile/Scenario Algorithm for Real-Time 3D Environments. Proceedings the 32nd International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2005), Los Angeles, California. August 2005.

[8] C. Tovar; J.M. Cabanellas; J. Félez, Procedimientos geométricos para el ajuste de trayectorias ferroviarias en simuladores 3d de diseño modula. Proceedings of ADM-INGEGRAF'2003.(June).Naples,Italy.