MODULAR TECHNOLOGY: A METHODOLOGY FOR THE CREATION OF VIRTUAL ENVIRONMENTS IN TERRAIN DRIVING SIMULATORS

Carlota Tovar
Ginés Jesús Jimena
Jose María Cabanellas
CITEF Research Centre on Railway Technologies
Universidad Politécnica de Madrid.
José Gutierrez Abascal 2. 28006 Madrid,
Spain
E-mail: ctovar@etsii.upm.es;citef-gjimena@etsii.upm.es ;jmcabanellas@etsii.upm.es

KEYWORDS
training driving simulators; geometric modelling; shaders; modularity; virtual environments.

ABSTRACT

This paper presents the latest research and developments in Modular Technology, a constructive methodology designed to create the environments of the virtual driving simulators of CITEF. 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. This tiling system has a set of particularities that make it different from the rest of tiling systems used up to now. These particularities are designed to get the maximum benefit of these type of virtual representations and can be summarized in four points: the shape of the module; the way in which the family is defined; the positioning system of these modules; and the deformation of these modules using 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.
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.

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.
When creating virtual environments for training driving simulators, it is indispensable to take into account the final aim of these virtual representations: to train a driver. Sometimes drivers are not used to work with computers, so that, they need intuitive, handy and useful tools. The functionality of the environment must be their main aim and the aesthetic appearance has a secondary relevance. Trying to fulfill these requirements, the different entities that develop training driving simulators create their own applications.
CITEF (Research Centre on Railway Technologies) is one of these entities and it has developed Modular Technology, a constructive methodology designed to create the environments of 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.
In the development of its tools, Modular Technology has considered the change of priorities in scene optimization. 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. At the same time, the programming possibilities of current GPUs (*shaders*) (Wolfgang,E 2004) offer great flexibility to build stunning environments.
In response to this change of priorities, 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 (Gobbetti et al, 2006). 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 has converted the module into its base primitive and has created the environments using thanks to the use of instantiation and shaders.
Instantiation is an useful technique to save memory (Schultz, Schumann, 2000). For scenes with many similar objects it is desirable to save just the properties that are different between those objects and not to save the full geometric representation of each object. The *Master* is the object which holds full geometric information and the *Instance*, is a transformed variant of the *Master* object (Sutherland, 1963). In this way, as the *Instance* is a pointer to the *Master*, the geometry is saved just once and it can be reuse (instantiated) as many times as desired.
A tile-based (or instantiation-based) methodology, is characterised by its seeking the use of repetitive patterns as the basis of its designs to generate an environment by assembling and repeating a finite number of modules generated in pre-load time. This methodology usually goes

hand-in-hand with the use of a system of discrete levels of detail (Luebke 2003). The most usual geometric shapes for generating these tiles are the rectangle, the circle, the hexagon and the diamond shape. The purpose of the two latter is to provide a greater feeling of depth. These tiles are placed adjacently to one another as the following figures show, completely filling the 2D space.

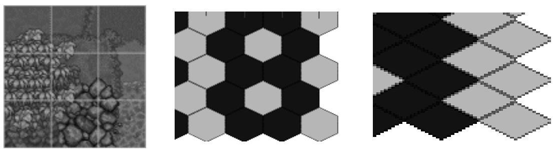
Figure 1. Examples of tiles

This methodology is used frequently in video games. Two of the best-known 3D games using this methodology are *Simcity*, and *Neverwinter Nights*. Their main feature is that they define the scene by using tiles that form part of what is known as a tileset. A tileset is nothing more than a set of tiles that follow a similar theme. For example, the City tileset can include tiles of pubs, inns…and other elements to be found in a city. The main aim of this methodology is to cut memory consumption and to reduce modelling tasks. The level of detail of these scenes, however, plays a secondary role in fulfilling the objectives of the game, which means that realism is not a decisive factor when generating the scenes.
Modular Technology is tile-based methodology that searches for repetitive patterns in the environment, geometric as well as functional. Using these patterns (*Masters*) 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.

MODELLING LARGE VIRTUAL ENVIRONMENTS IN TERRAIN DRIVING SIMULATORS

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 (Livny et al, 2007; Pajarola and Gobbetti, 2007). 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 (Chew, 1987), 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 (Govil and Mourant, 2005; Papelis et al, 2003; Suresh and Mourant, 2005) 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 designing a new tile-based system, with some particularities that let us to solve all these drawbacks as it will be explained in next sections.

PRINCIPLES OF MODULAR TECHNOLOGY

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.
Knowing the driving paths allows setting the distance at which the different elements will be observed, thereby enabling the optimum parameters of the levels of detail to be defined in pre-load time: the cut-off distances and polygon load for each level of detail.
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, graphic engine and graphic designers (Figure 2).

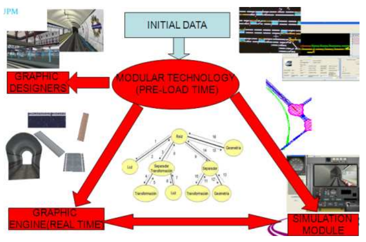

Figure 2. Areas involved in the creation of a virtual environment in CITEF

Modular Technology analyses the initial data, solving incoherences and mistakes. Once this information is ready it distributes it, sending:

- to the simulation module all data that the vehicle needs from the environment to function properly. Data related to the geometry of the paths, sections, signals, magnets, etc.
- to the graphic engine the geometry that it has generated automatically and a text file in which this geometry is positioned and structured in a scene graph. This scene graph contains information about the levels of details, switches, shaders..
- to the graphic designers the information of the geometry that can not be generated automatically because of the high level of detail required.

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 and load times are minimised. This last point is very important for the user to be able to launch different scenes easily and efficiently in a training session.

• 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.

MODULAR TECHNOLOGY PARTICULARITIES

The particularities of this tiling system will be explained throughout this section. These particularities, which make Modular technology different from all other tiling systems utilised up to now, are related to:

1. The shape of the module.
2. The way in which the family is defined.
3. The positioning system of these modules.
4. The deformation of these modules in execution time using shaders.

Particularity 1: the shape of the module

The shape of the module usually used by the different tiling systems does not take into account the shape of the driving paths. Squares, circles and rhombus are the shapes more frequently used by these tiling systems.

In Modular Technology, 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 to represent the shape of the driving paths.

This group of modules of different curvature forms 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. If the straight module is curved in execution time using shaders, this basic set of modules will be compound by infinite elements.

The module comprises three basic components: a longitudinal path and a set of transversal profiles with their corresponding textures (Figure 3.).

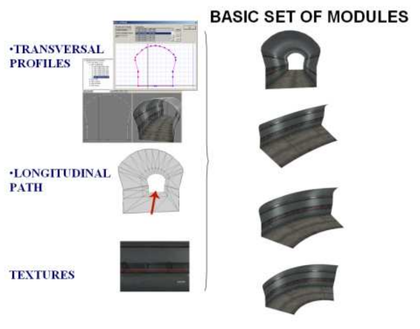

Figure 3. Components of a module and basic set of modules

Particularity 2: The definition of the family

Frequently, the design of these families is very rigid, so that, the environments created with the repetition of these families tend to be very unrealistic. These families include too much detail and the user can perceived the repetition. Next figure shows an example of one of these families.

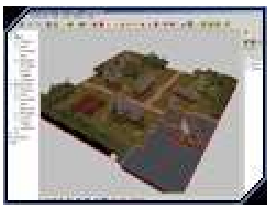

Figure 4. Example of family used by a tiling system

Modular Technology takes into account the driving paths to define these families. So, the concept of a family of modules now appears as the base set of modules that shares the same definition of transversal profiles and textures (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.

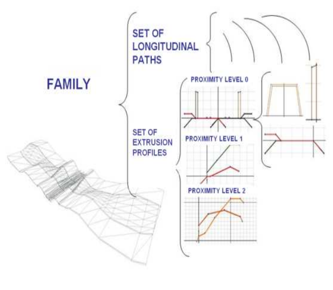

Figure 5. Components of a family

Modular Technology takes advantage of the priori knowledge of the paths to generate discrete pseudo-variant levels of detail with the view point. This is achieved by discretizing the environment transversally to the path in different levels of proximity (Figure 6).

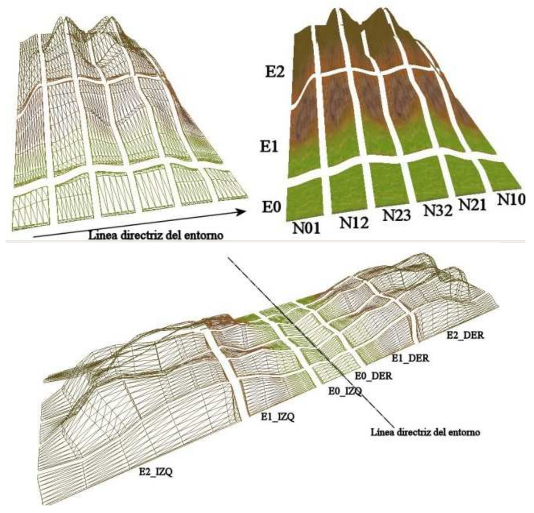

Figure 6. Example of a terrain family

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

Particularity 3: The positioning system

Up to now, the different tiling systems positioned the tiles forming a mesh. The distance to the driving paths was not taken into account, so that, the level of detail could not be calculated with precision. Modular Tecnology solves this problem positioning the tiles along the tracks, so that, the level of detail of each module can be optimized evaluating this distance.
The modular positioning algorithm has to solve the following issues (Tovar, 2003):

- The optimum number of modules, responsible for a higher or lower graphic load and a better or worse scene graph path optimisation. In order to decide the optimal number of modules to make up the environment, it is firstly necessary to define the families comprising it and secondly to define the basic length, LB, for each of these families. Although it is obvious that instantiating involves shorter scene loading times, it is advisable to proceed with caution when deciding on the number of elements and how to define the elements that are instantiated, since an excessive number lead to an opposite effect:
    - a large number of changes in state results in a GPU overload and a drop in frame rate.
    - a large number of elements involves more calculations in the scene graph in order to determine how many elements need to be displayed.

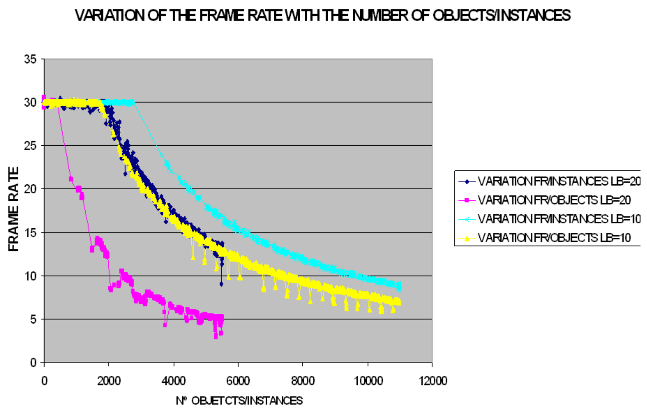

Figure 7. Evaluation of the the number of elements on the frame rate

In the above example, the frame rate has been limited to 30. As can be seen, there is a threshold in the number of instances to be used and if this is not exceeded the performance of the graphic engine is stable and supplies a maximum frame rate regardless of the basic length used. When creating each scene, this threshold must be calculated, and consequently, the basic length.
This threshold varies according to the number of changes of state linked to each module, which, in turn, depends on the number of drawables and textures defining the threshold (Figure 8).

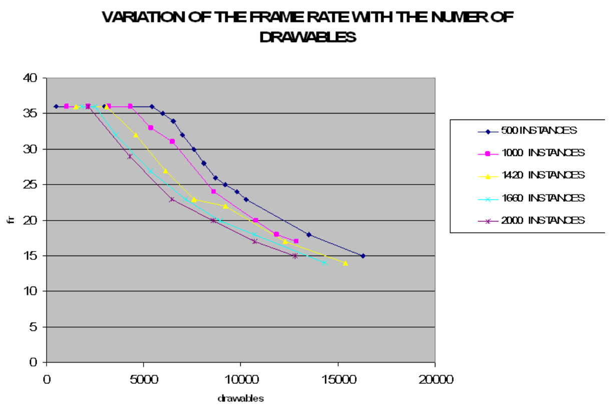

Figure 8. Evaluation of the number of drawables on the frame rate

Therefore, the procedure to be followed to calculate the basic length, LB, will consist in firstly determining the families that will set the environment and the number of drawables that is to comprise the module. Taking this information and using the graph depicted in Figure 7, after an iterative process, the appropriate number of instances and therefore the basic length is determined.

- 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.
- 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, LB). 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.

- 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 9. Figure 10. 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.

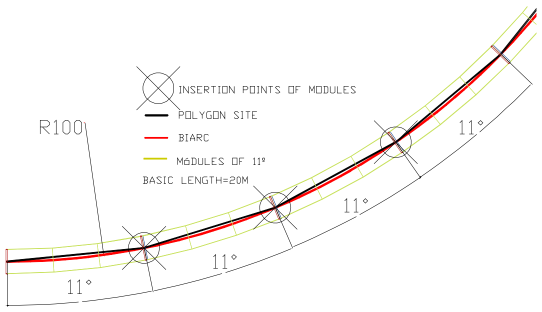

Figure 9. Positioning algorithm

Figure 10. 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:

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

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

Particularity 4 : The use of shaders

Up to now, the different tiling systems did not deform the geometry of the tile in execution time to create more realistic environments. 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 LB. This length is measured in the direction of the axis i_1, as Figure 11. shows.

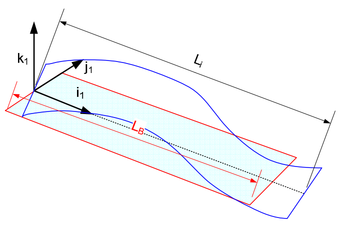

Figure 11. Straight module deformed by the shader

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.

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 LB. The parameter s will vary between 0 and 1.

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

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:

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

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:

$$\begin{aligned} x_R &= x_0 \cdot E_x \\ \frac{dx_R}{dx_0} &= E_x \\ \frac{dx_0}{ds} &= L_B \\ \frac{dx_R}{ds} &= L_B \cdot E_x = L_i \end{aligned} \qquad (2)$$

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

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

The functions f_1 and f_2 have the following properties:

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

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.

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

For the angle β(s):

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

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

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

The results of this deformation with shaders are shown in next Figure.

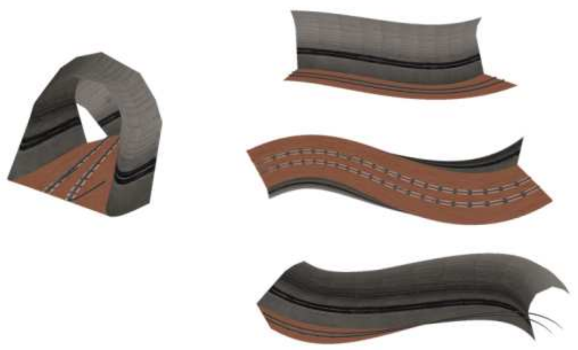

Figure 12. Straight module deformed by the shader

APPLICATIONS

All virtual environment projects developed in CITEF have a modular component to a greater or lesser degree, which helps develop automation.
The development of the simulator for Metro of Madrid is an example of the advantages of this tiling system. Per example, the 80% of Line 7 has been created modularly. Stations and deposits are the only elements created ad hoc because of their high number of specifics details. This implies a great reduction in modelling tasks: the 23 km of tunnel has been reduced to the creation of 5 families of tunnel, i.e, 5 modules of 20 meters. So that, the resources envolved in the creation of this geometry are drastically reduced.
Next table shows the savings in memory and loading times thanks to the use of Modular Technology in the representation of the tunnels of this line.

Table 1. Loading times and memory used by tunnels in Line 7 of Metro of Madrid

USE OF MODULAR TECHONOLOGY	LOAD TIME (seconds)	MEMORY
YES	0.059	173KB
NO	31	133MB

These loading times and memory savings are crucial during a training session. Drivers need to launch several exercises during these sessions, so they can not waste time loading them. On the other hand, these trainings imply to visualize huge environments, per example, the Madrid Metropolitan Network, so the memory used has to be minimize. Moreover, drivers need to modify and create new environments during their training sessions. Thanks to the use of Modular Technology this tunnel families can be reused without an extra modelling effort.
Figure 13 shows different scenes that have been modularly generated. These scenes are taken from the virtual environments of the driving simulators designed by CITEF for companies such as Metro de Santiago de Chile, Metro de Madrid, CAF, Alstom and Invensys.

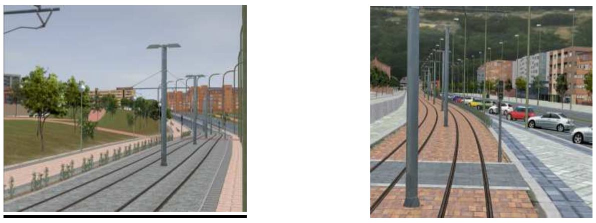

Figure 13. Examples of virtual environments created with the Modular Technology

CONCLUSIONS

From what we have seen from the applications used, we are convinced that modular technology has advantages when it comes to generating environments. Ten years of successful results in the generation of virtual environments for renowned driving simulators have proved it. Applying shaders increases the possibilities of modularity and 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

Chew,L.P.1987.“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.

Gobbetti, E.; F.Marton ;P. Cignoni. 2006. “C-BDAM - Compressed Batched Dynamic Adaptive Meshes for Terrain Rendering”. Computer Graphics Forum, 25(3), (Sept). Proc. Eurographics 2006.

Govil, V. and Ronald R. Mourant. 2005. “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).

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

Luebke, D., M. Reddy, J. Cohen, A. Varshney, B. Watson and R. Huebner. “Level of Detail for 3D Graphics”. Morgan Kauffman.2003.

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

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

Schultz,R., Schumann,H. 2000. “Efficient Scene Descriptions Using Advance Modelling Techniques in the RenderMan Context”. Proceedings of Spring Conference on Computer Graphics 2000.

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

Sutherland, I.E. 1963.“Sketchpad: A man-machine graphical communication system”. Proceedings of the Spring Joint Computer Conference 1963.

Tovar, C.; J.M. Cabanellas; J. Félez. 2003. “Procedimientos geométricos para el ajuste de trayectorias ferroviarias en simuladores 3d de diseño modular” Proceedings of ADM-INGEGRAF’2003 (June) Naples, Italy.

Wollfgang, E. 2004 “Programming vertex & pixel shaders”. Charles River Media Graphics.

CARLOTA TOVAR received her Mechanical Engineering Master degree from the Madrid Polithecnic University and her Master degree on Geographic Information Systems from the University Pontificia of Salamanca. She has been working for ten years in CITEF, Research Centre on Railway Technologies. Her main activities and research interests are mainly focused on the field of simulation, computer graphics, virtual reality and geographic information systems. Her main contribution is in the field of automatic generation of virtual environments. She has just finished her doctoral thesis on this subject. She has been actively involved in over 20 research and development projects.