An Experimental Environment for Construction of Virtual Terrains

Ioan Mircea Popovici, Mircea Popovici Jr.
Civile Marine Institute, OVIDIUS University
Constanta, Romania

Abstract

This paper presents an experimental environment used in automatic generation of terrain models. The instrument supports multiple user access (navigators), allows comunication between them, dynamic modelling activities, and supports a user configurable interface.

The initial implementation of this environment runs on a network of Windows stations, and we prepare a Unix version too.

Keywords: *Digital elevation model, virtual environment, virtual world, automatic terrain generation.*

1. INTRODUCTION

With the rapid development of numerical techniques for the solution of engineering problems, there has been an increasing demand for the automatic generation of computational meshes in the past few years. Nowadays, automatic mesh generation is widely used in a variety of scientific applications, such as contour plotting, finite-element analysis, computer-aided design systems, and not least in synthetic environments.

A synthetic environment is a set of data elements that describe a scene, in particular a geographical region, simulation entities, such as vehicles, airplanes, and humans; and events expected to take place during the interactions with that environment.

2. VIRTUAL TERRAINS CONSTRUCTION

Virtual terrains can be obtained by applying a tesselation algorithm with a digital map data, aerial image, or other source, as input data.

Digital terrain data consists of earth surface elevations associated with different geographic positions and collected in a variety of formats. This data can be stored as isolated spot elevations from a field survey (giving elevation only at points), cartographic source material such as terrain contour lines (giving elevation along linear features), or a dense grid of elevations measured from a stereo photography (giving elevation at every point in that grid). However, most **D**igital **E**levation **M**odels (DEMs) use regular grid, where the geographic position is implicit from the point location in the array. DEM data can be collected directly from aerial photographs, or interpolated from point or contour data.

Therefore, we consider the regular DEM to be the basic form of terrain data, and focus on methods to create terrain models from it.

2.1 Terrain models

The primary limiting factor in the complexity of terrain models used in a virtual world simulation is the real time scene-rendering requirement. Achieving the necessary speed, and maintaining a realistically detailed terrain model typically requires specialized computer graphics hardware. Given computer graphics architectures, this implies use of a polygonal mesh. There are three common types of polygonal models: a regular mesh (the simplest polygonal model with the terrain points aligned on a grid), a hierarchical mesh (which displays a fine regular mesh in areas near the viewer, and successively coarser meshes in more distant areas) and a **t**riangulated **i**rregular **n**etwork (TIN, whose points are placed and connected freely into a triangular mesh).

A TIN can represent terrain at least as well as, and usually, better than DEM, because DEMs are basically restricted TINs. This fidelity improvement is not made at the cost of interoperability. Any polygonal-mesh terrain model is essentially a TIN with additional edges. A data construction team can integrate features such as roads, rivers and lakes into a TIN with their boundaries intact. The TIN construction software can select points directly from the DEM regardless of its resolution, and add point or linear terrain data to supplement this DEM. When no DEM is available, the team can still create a TIN by simply triangulating point, or linear terrain data.

2.2 Construction techniques

Several methods for TIN construction have been proposed. The basic issues are point selection from the DEM and point triangulation. The desired product is a mesh that closely approximates the original DEM.

The class of used algorithms can be organised into three types; static, semidynamic, and dynamic, based on the method implemented in the algorithm. For the static one, the entire set of DEM points is needed before the mesh process starts. This is the case of the algorithm discussed in [1] and [5]. In the case of semydinamic algorithm, the DEM points are processed at input. The only difference between the second and the third type of algorithms is that in the case of the last one, we can extract the points as easy as we insert them.

Each of these methods can be used in terrain generation.

Our DEM generation procedure begin with an initial data set, with a defined contour, and iteratively refine it. The initial DEM can be as simple as a triangulation of the DEM corners, but can also include user-selected points. The new vertex is connected to the vertices of the triangle containing it, based on empty circumcircle criterion. The process of adding points can be repeated until some stopping criterion is met (point or polygon count, or maximum error).

If the density of DEM points is too high to affect the quality of the resulted terrain, we can modify the point-selection process to use a terrain-importance mask, which assigns a weight to each DEM point. Correction points are selected based on the weighted error. This concentrates points in regions assigned a high relative importance. We can exclude individual points, or entire regions from point selection by setting a null importance. This affect situations where particular terrain features require additional fidelity, but do not generate a significant error because of fine spatial details, and the rendering time also. One example of this is the terrain at Nice, which shows complex terrain details (Figure 1).

2.3 Examples

Here are some examples of generated terrains using an irregular set of points (Figure 1), a regular grid (Figure 2) and hand-made data points or VRML file (Figure 3).

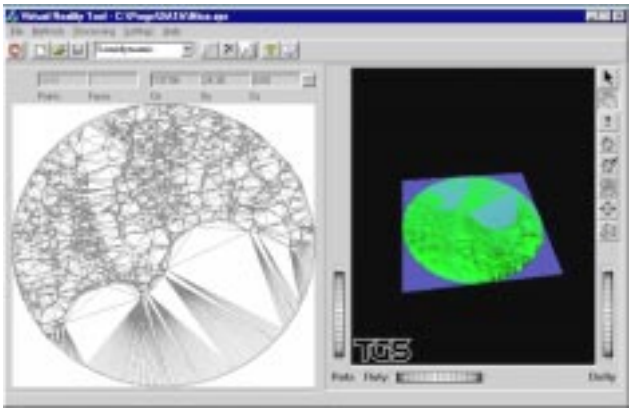

Figure 1: Irregular set of points (Nice)

3. THE VIRTUAL ENVIRONMENT

The **VRT**ool has been developed to satisfy a number of requirements that will be discussed in subsection 3.1.

3.1 This environment is based on a number of concepts that will be outlined in subsection 3.2. Requirements

The fundamental requirement is the ability to share information in real time. This shared information will then be the basis for the concomitant navigation sessions. A second fundamental requirement is the treatment of individual perspectives associated to the navigators. A perspective defines a context within which the shared information can be manipulated in a meaningful way by a navigator.

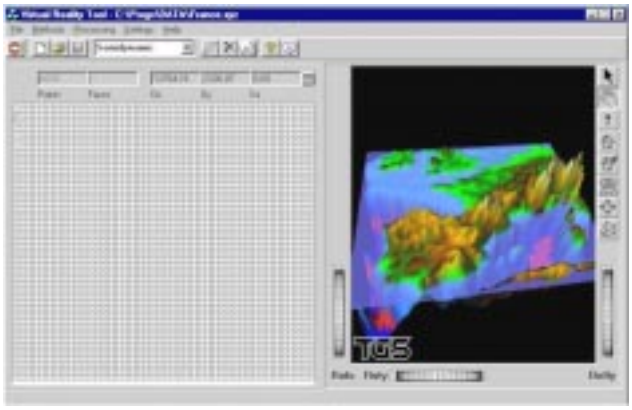

Figure 2: Regular grid (west of Europe)

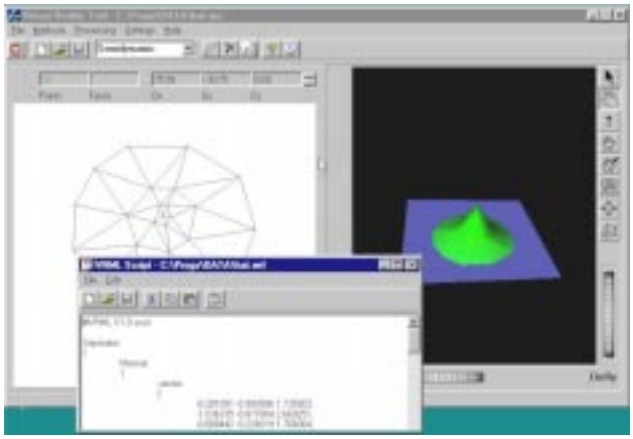

Figure 3: Hand-made or VRML (Hat)

In addition, several requirements have been considered during the implementation of the virtual environment. The architecture should:

- provide support to permit dynamic modeling activities,
- be open and extensible so that different perspectives can be integrated easily,
- have the potential to be scaleable for large meetings, with a dynamic number of navigators.

3.2 View of the architecture

In the following we introduce a number of fundamental concepts that are used in the **VRT** environment to satisfy some of the requirements outlined above. Here is a conceptual view of the **VRT** architecture.

If the user intends to add/modify information, then the model will be viewed as a sum of local models, or else, local models are copies of the main model.

The user's perspective defines what information is meaningful to them, how they will interact with that information, and how they will manipulate the information

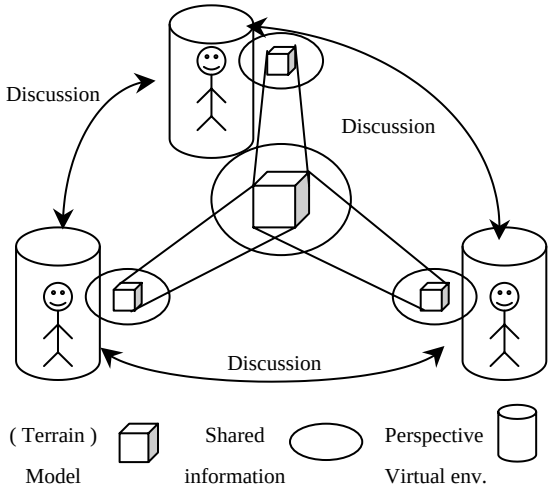

Figure 4: Conceptual view of architecture

they access. More than that, the users can add information to the model they study. To support this, each user of **VRT** environment has a user configurable interface, with all the operations they wish to perform and visualisation style they want to use. The users may modify their perspectives at any time during a sesion, by changing the configuration of their interface. These changes will also affect the other users subject of study.

As the discussion progresses, it may change focus, and more information may be added to the shared information space from/to the terrain model. The shared information space can potentially contain a huge amount of information that may be meaningful to a user, and therefore satisfy his information needs, but may not all be of interest. The environment allows a user to choose which subsets of the available information he is interested in; moreover, it also permits him to store/restore them within the environment. To achieve this aim, each navigator in have access to a consistent description of the environment. This description is accurate in its detail, its modeling of terrain, structure, and entity motion, and in its graphical modeling of the environment.

Users can also change their interest as the discussion progresses, by focusing on other sets of information, or discarding an interest in a particular set of information. Colaboration/competition can occur when users are interested in the same particular set of information.

4. IMPLEMENTATION AND RESULTS

The current prototype of the **VRT**ool architecture runs on computers from the PC family. The architecture has been implemented using Visual Basic, Visual C++ combined with ActiveX components provided by TGS. The files used in generating terrains were ASCII files (sets of triple (x,y,z) organized in a regular or irregular mesh). The output results are converted into OpenInventor or VRML formats.

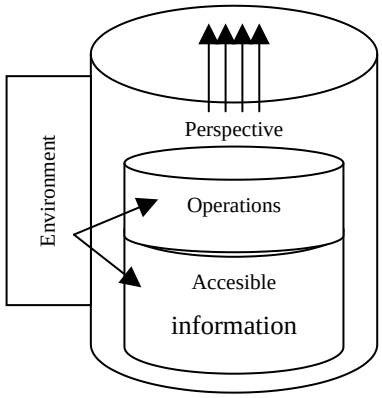

Figure 5: Detailed view

This prototype can be used as a modeling tool, and as a teaching tool also. For this, it provides the capability to use different algorithms in generating the triangular tesselation, from static, semidynamic, and dinamic classes. More, it lets the user to define his own methods of generating the terrains, or/and introducing constraints.

Last but not least, the users of the **VRT**ool can communicate to each other the results of the modeling phase via a meeting.

The current interface is illustrated in figures 1, 2, and 3.

5. CONCLUSIONS AND FUTURE WORK

In this paper we have discussed an architecture to support automatic terrain modeling over a multiuser configuration. The **VRT** environment allows geometric modeling and comunication between the users with a view to discussing and testing precise solutions in real time. The architecture is curently evaluated and refined using case studies.

6. ACKNOWLEDGEMENTS

The authors like to thank to professors Jaques Tisseau, Serge Morvan, Marc Parenthoen, Alexis Nedelec, Jean-Pierre Gerval, Patrick Reignier, Pierre Chevaillier, Pascal Redou, and Fabrice Harrouet, and all the team of Li2 from ENIB, France, for all the support and constant encouragement regarding our work. Thanks also go to professor Luca Dan Serbanati and Vladimir Boskoff from OVIDIUS University of Constanta, Romania, for useful discussions.

7. REFERENCES

[1] C.Du – *An algorithm for automatic Delaunay triangulation of arbitrary planar domains*, Advances in Engineering Software, **27** (1996), 21-26.

[2] A.Okabe, B.Boots, K.Sugihara – *Spatial Tessellations. Concepts and Applications of Voronoi Diagrams*, John Wiley & Sons, 1992.

[3] A.K.Chaturvedi, L.A.Piegl - *Procedural method for terrain surface interpolation*, Computer Graphics, **20** (1996), 541- 566.

[4] D.Hancock – *Prototyping the Hubbe Fix*, IEEE Spectrum, Special Issue on Virtual Reality: tools, trends and applications, October, 1993, 34-39.

[5] M. Popovici Jr. – *Some facts concerning the automatic Delaunay triangulation of planar domains*, Procceedings of the seventh symposium of mathematics and its applications, Timisoara, 6-9 November 1997.

[6] F.P.Preparata, M.L.Shamos – *Computational Geometry. An introduction.* Springer, New York.

[7] T.P.Fang, L.A. Piegl – *An algorithm for Delaunay triangulation and convex hull computation using a parse matrix*, Comput.-Aided Des. 1992, **24**, 425-436.

Authors:

Ioan Mircea Popovici, professor PhD at Mathematics Department, Marine Civile Institute of Constanta.
Address: 104 Mircea cel Batran, 8700 Constanta, Romania.
E-mail: impopovici@io.imc.ro

Mircea Popovici Jr., professor assistant at Department of Computer Science and Numerical Methods OVIDIUS University of Constanta.
Address: 124 Mamaia, 8700 Constanta, Romania.
E-mail: dmpopovici@ovidius.ct.ro