The use of digital elevation models in generating real-time 3D environments: a case study

Maricel PALAMARIU, Prof. PhD.,"1 Decembrie 1918" University, Romania, mpalamariu@uab.ro
Vlad BUDA, Student, "1 Decembrie 1918" University, Romania, vlad.buda@hotmail.com
Adrian Robert IANCU, Student, Babeş-Bolyai University*, Romania, iaro2006@gmail.com*

Abstract: *In this paper we illustrate the basic tasks which have been undertaken, with reference to a case study represented by a hill in the vicinity of Alba Iulia, Romania, in order to generate a 3D real-time environment using a digital elevation model as support. A 3D application has been created in order to visualize and explore the virtual environment.*

Keywords: *digital elevation models, virtual environment, computer graphics, 3D application, panorama*

1. Introduction

Capabilities for visualizing topography in GIS have evolved from 2D raster maps, contours, and simple 3D meshes in the 1980s to interactive 3D rendered surfaces with shading and color maps draped over the surface that became standard in the 1990s [1]. Recently, several new technologies that combine the flexibility of digital landscape representation with intuitive 3D physical models have emerged [2]. These technologies [3] open new possibilities for user interaction with geospatial data. A prototype tangible geospatial modelling environment lets users interact with landscape analysis and simulations using a tangible physical model [4].

New mapping technologies, especially laser scanning (for example, Lidar and Ladar), have dramatically improved terrain-mapping efficiency and high-resolution, digital elevation models are now increasingly available. These data provide a new and more dynamic view of landscapes and stimulate the need for viewing, analyzing, and modeling terrain change and its impacts within GIS. Interacting with terrain data has become important for different types of spatial analysis and design and planning tasks. It can provide insights into the relation between terrain surface shape changes and spatial patterns of the terrain's derived parameters [1]

Virtual environments are highly interactive, allowing users to explore and visualize large landscapes. Real time modeling of the physical world has many uses, such as disaster impact prediction, disaster recovery planning and training, urban planning and virtual tourism [5].

We present a method for creating a real time virtual environment by using a digital elevation model as our base for the terrain. While this method is quite complex, it has the advantage of allowing developers to implement additional data, such as illumination in real time, buildings, trees, vegetation, water etc., thus enabling the construction of a complete 3D world, not just a 3D terrain. A very important auxiliary data that has been used in this application, in order to exemplify the usefulness of this method, is a full 360° panorama.

This virtual environment has been achieved by using software such as: Autodesk Maya 2008, Irrlicht Engine, Adobe Photoshop CS 3, Global Mapper 9, Terragen, and Earth Sculptor.

2. Digital elevation models

Modern cartography in digital form, or, equivalently, numeric cartography embodies, among others, digital models of the terrain, denominated with various acronyms such as Digital Terrain Model (DTM), Digital Elevation Model (DEM), Digital Height Model (DHM) or Digital Ground Model (DGM); further, ortho-images clad or superimposed on the DTM as textured photo, become the metrically correct and highly descriptive base on which information of any other nature can be integrated [6]. Such three-dimensional models inserted in Territorial Information System (SIT) or a dynamic Geographical Information System (GIS) can supply a useful instrument to open new opportunities in the management of environmental and territorial data, in the monitoring of environmental resources, etc [7].

A digital elevation model (DEM) is a set of points defined in a three-dimensional Cartesian space (X, Y, Z) that approximates a real surface. The X and Y axes may be expressed as geographic coordinates (i.e. longitude and latitude), whereas the Z axis represents the altitude above sea level [8]. It is a digital file consisting of the terrain elevations for ground positions at regularly spaced horizontal intervals [9]. Digital elevation models can be generated directly through photogrammetric processing [10] of stereo-photos [11] or satellite imagery such as stereoscopic SPOT images [12], or indirectly from the interpolation of scattered point elevation data, of contour lines, or of Triangular Irregular Networks [13].

For this case study we have chosen a digital elevation model of the Gorgan Hill region. The digital elevation model covers approximately 500 square meters of this specific area. The Gorgan hill (Gorganu, The Gorgan Peak), belongs to the West rim of the Secaşe Plateau, in the contact sector with the Mureş Valley on its middle sector, near Alba Iulia, which is situated in the West. For a clearer localization within the micro area, we mention the fact that the hill is situated approximately at a distance of 4-5 km South-East from the fortified hallstattian location in Teleac, at approximately 2.5 km West from the Măgura Străji Hill Peak and at about 6 km East from Alba Iulia (latitude: 46° 3' 53''; longitude: 23° 39' 2'') [14].

Our digital elevation model is in Generic ASCII XYZ format. This is the raw, unaltered version of the DEM, containing only the elevation data for the three axes without any projection information or datum. The digital elevation model has a resolution of 10 meters (Fig. 1).

```
"X","Y","Z"
395405.00000,506495.00000,432.413
395415.00000,506495.00000,432.654
395425.00000,506495.00000,432.894
395435.00000,506495.00000,433.135
395445.00000,506495.00000,433.350
395455.00000,506495.00000,433.565
395465.00000,506495.00000,433.780
395475.00000,506495.00000,434.070
395485.00000,506495.00000,434.361
395495.00000,506495.00000,434.651
395505.00000,506495.00000,434.879
395515.00000,506495.00000,435.108
395525.00000,506495.00000,435.336
395535.00000,506495.00000,434.618
395545.00000,506495.00000,433.900
395555.00000,506495.00000,433.182
```

Fig 1. Digital elevation model in Generic ASCII XYZ format

3. Converting the DEM into a 3D terrain

Since the digital elevation model is just a raw, unaltered .txt file we had to convert it into a format that could be imported in a 3D editing program. Below are the steps in the process of transforming a .txt DEM into an .obj file format which is the standard format for any 3D editing program. We present the entire process which is quite complex.

Global Mapper is a program that accurately performs distance and area calculations, raster blending and contrast adjustments, elevation querying, and line-of-sight calculations to maximize precision, digitize new vector features, edit existing features, and easily save them to supported export formats, automatically triangulate and grid 3D point datasets to convert a set of elevation samples into a fully gridded dataset.

We have used the Global Mapper program to import the ASCII file as an elevation grid from 3D point data, set additional information and export it as a Terragen File (Fig. 2). The simple Generic ASCII file does not contain any projection or datum information. We have added the information manually using Global Mapper. The standard romanian projection is stereo 70 and the standard datum is "Dealul Piscului", found in this application as S-42. After entering these new settings in Global Mapper the program will automatically generate a height map model displayed in this case by shades of red, green and blue (Fig. 3).

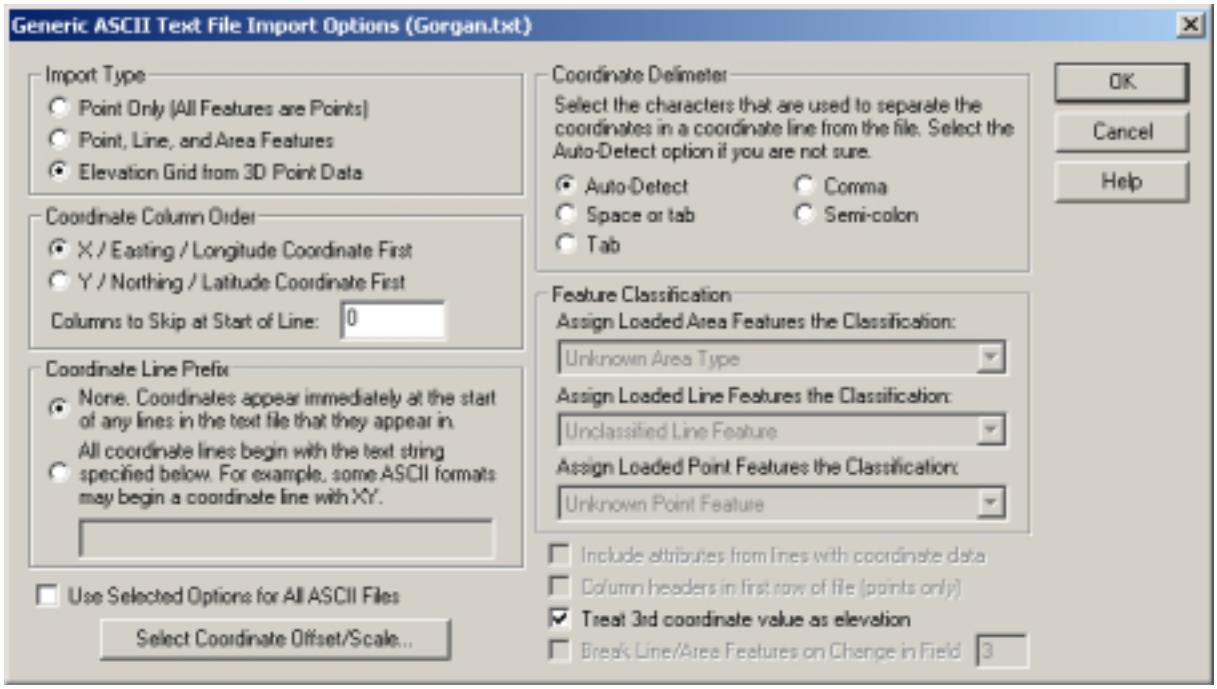

Fig 2. Generic ASCII XYZ file import options

We needed to create a height map image that could be read by a 3D application. There are a couple of methods to generate a height map image in Global Mapper directly but the downside of those methods is the fact that they don't support 16 bit files. Only 16 bit files can retain enough terrain data so that they don't alter the information in the digital elevation model. If we would have used an 8 bit image file the interpolation algorithm would have caused considerable damage to the data. Our method is bit more complex and it requires a few more steps and a few more programs to use but in the end we end up with an accurate heightmap. Therefore, we exported the

DEM as a Terragen File, setting the terrain size to a resolution of 1025x1025. Terragen files can store 16 bit information size without any loss of data.

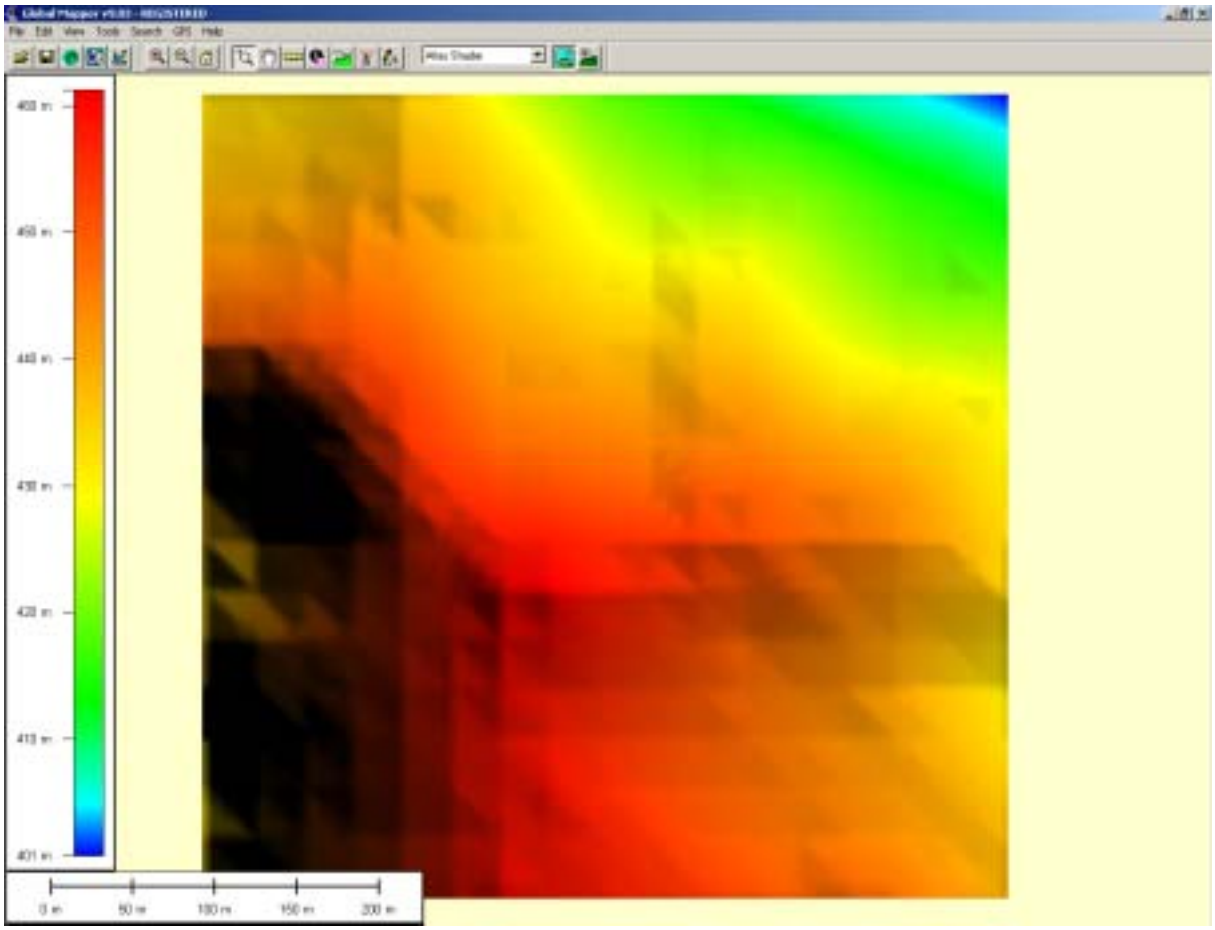

Fig 3. Digital elevation model in Global Mapper

In Terragen we loaded the new terragen file that we created in Global Mapper and we checked for any inconsistencies that might have occurred during the exporting process. Terragen has a very formal 3D preview. We have used that to inspect our terrain. We exported the terrain as a RAW 16 bit Intel Byte-Order. The reason for exporting it as a RAW 16 bit file is so that the file can retain all its data. If we save it out as an 8 bit file it would lose the finer detail and the interpolation algorithm can cause damage to the data (Fig. 4).

Next, we took the RAW 16 bit file in Adobe Photoshop CS 3 which can open and manipulate RAW files. Photoshop has many features that enable work on 16 bit RAW files. We have set the options for Photoshop to read the file and we exported the RAW file as a 16 bit PNG file format. We are using the 16 bit PNG file format because this is the standard file format in Earth Sculptor.

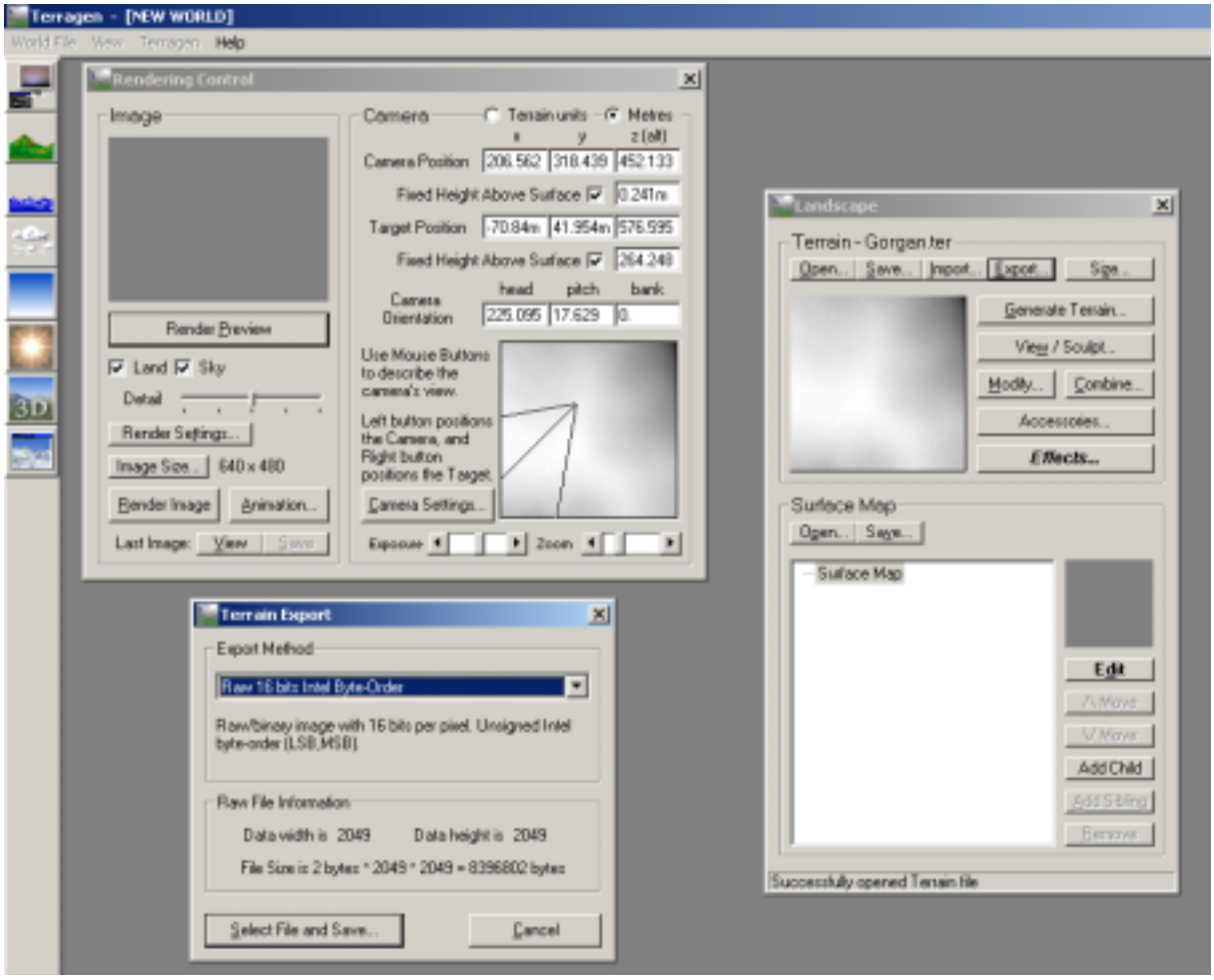

Fig 4. Terragen

Earth Sculptor is a real-time terrain editor program designed exclusively for rapid development of 3D landscapes for visualization, multimedia and game development. Earth Sculptor has the ability to load height maps and transform them into digital terrain. We've used Earth Sculptor for optimizing the terrain and exporting it as an obj file.

4. Rebuilding, texturing and illuminating the terrain in Autodesk Maya

Autodesk Maya is a prominent modeling tool used for animation, game and architectural development [15]. Maya offers the possibility to model additional 3D structures and objects for use in a virtual environment. It allows users to create textures and light maps.

We have imported the terrain as an obj into Autodesk Maya and we began the process of rebuilding the terrain. Maya offers the possibility to model additional 3D structures and objects for use in a virtual environment. After parts of the terrain have been edited we began to texture the ground using a texture map that we created in Adobe Photoshop. We added a simple material to the ground surface of the terrain for the texture that we created earlier.

Illumination is a very important part in refining a virtual environment. Using a static illumination system, also called lightmapping, we were able to recreate an ambient light atmosphere, simulating an overcast sky. Both the texture map and the light map were saved in a folder ready to be used for the next step.

5. Panorama

A panorama is an image having a wide field of view, up to a full 360 degrees [16]. We created a panorama of the surroundings of the Gorgan hill in order to provide users with an extended feeling of reality (Fig. 5).

Fig 6. Panoramic view from the top of the Gorgan Hill

6. Integrating the scene in the engine

The terrain is now ready for the Irrlicht Engine. We chose the Irrlicht Engine because it's an open source, high performance real-time 3D engine written and usable in C++ and also available for .NET languages. It is completely cross-platform, using D3D, OpenGL and its own software renderer, and has all of the state-of the-art features which can be found in commercial 3d engines. In the application the engine is used to render the reconstructed location.

For the purpose of this project we worked with the Visual Studio 2005 IDE. First we included the header for the Irrlicht Engine. *#include <irrlicht.h>*. In Irrlicht all classes reside in the "irr" namespace in one of the five sub-namespaces. For the executable to be able to use the Irrlicht engine, residing in the Irrlicht.dll file, we had to link to the Irrlicht library.

```
#pragma comment(lib,"Irrlicht.lib")
```

In the main program, the first thing we did was to declare pointers for the Irrlicht device, the video driver and the scene manager.

```
int main()
{
	IVideoDriver* driver;
	ISceneManager* smgr;
	IrrlichtDevice *device;
```

We initialized the device. The parameters in order are: the type of driver we use, in this case OpenGL, the resolution, the bits per pixel, and if we want to run the application in full

screen or window mode. There are a number of other drivers such as Direct3D, Software Driver, etc.

```
device = createDevice(EDT_OPENGL,dimension2d<s32>(800,600),32,0);
```

From the device we got pointers for the video driver and the scene manager.

```
driver = device->getVideoDriver();
smgr = device->getSceneManager();
```

Next we declared a mesh which will hold the scene data, and we loaded an object into this mesh. Irrlicht supports a number of formats besides “Obj”.

```
IAnimatedMesh *levelMesh;
levelMesh = smgr->getMesh("../data/white_tower.obj");
```

We then attached this data to a scene node. If the size of the node is too small we can modify it with the setScale method.

```
ISceneNode *node = smgr->addAnimatedMeshSceneNode(levelMesh);
node->setScale(vector3df(85,85,85));
```

We took a look at the material properties. Because we have transparent textures in the scene (namely the tree leaves), we set the material type of the node as transparent. This will not affect the other textures, because they do not contain an alpha channel.

```
node->setMaterialFlag(EMF_LIGHTING,false);
node->setMaterialType(EMT_TRANSPARENT_ALPHA_CHANNEL);
```

To be able to view the scene we added a camera. For this camera we modified its position in the scene as well as the maximum distance for which objects will be rendered.

```
ICameraSceneNode *camera = smgr->addCameraSceneNodeFPS();
camera->setPosition(vector3df(-341,110,-50));
camera->setFarValue(100000.0f);
```

The next thing we arranged was collision. The first thing we did is to create a triangle selector. This is a class which can take triangles from a scene node and do something with them. The triangle selector is attached to the node.

```
ITriangleSelector* sel;
sel = smgr->createOctTreeTriangleSelector(levelMesh->getMesh(0),node,128);
node->setTriangleSelector(sel);
```

To make the camera capable of collisions, we created a collision animator. The parameters are: the triangle selector created in the previous step, the camera node, a 3D vector which defines the dimensions of the collision ellipsoid, the gravity vector, and the offset of the camera from the collision ellipsoid.

```
ISceneNodeAnimator* anim = smgr->createCollisionResponseAnimator(
        sel, camera, vector3df(30,50,30),
        vector3df(0,-1,0),
        vector3df(0,30,0));
```

The animator is then attached to the camera node.

```
camera->addAnimator(anim);
```

We no longer need the animator and the selector so we told the engine to dispose of them automatically when no longer in use.

```
anim->drop();
sel->drop();
```

Instead of having a black sky for the scene we loaded a skybox representing a panorama of the actual location. A skybox is a cube that completely encloses the scene and gives the impression of a sky. We first disabled the creation of mipmaps, added the skybox node (the parameters are six textures), and re-enabled mipmap creation.

```
driver->setTextureCreationFlag(ETCF_CREATE_MIP_MAPS, false);

smgr->addSkyBoxSceneNode(
                                driver->getTexture("../data/panorama-6.jpg"),
                                driver->getTexture("../data/panorama-5.jpg"),
                                driver->getTexture("../data/panorama-1.jpg"),
                                driver->getTexture("../data/panorama-3.jpg"),
                                driver->getTexture("../data/panorama-4.jpg"),
                                driver->getTexture("../data/panorama-2.jpg"));

driver->setTextureCreationFlag(ETCF_CREATE_MIP_MAPS, true);
```

Next was the render loop. Inside the render loop the scene is continually redrawn. We also added a condition to only render the scene when the application window has focus.

```
while(device->run())
        //if window has focus
        if (device->isWindowActive())
        {
                //begin the scene
                driver->beginScene(true, true, 0);
                //tell the scene manager to draw everything
                smgr->drawAll();
                //end the scene
                driver->endScene();
        }
```

Lastly we closed the device and exited the program.

```
        device->drop();
```

```
    return 0;
}
```

We built the application by selecting "Build Solution" from the "Build" menu. Before we could ran the application we needed to copy the Irrlicht DLL file from the "Bin/Win32-VisualStudio" subdirectory of the SDK into the Debug Directory of our application. Also we copied the scene that we exported from Maya as well as its textures into the Data directory (Fig. 7).

Fig 7. Gorgan – digital environment scene

7. Conclusions and proposals

The virtual terrain modeling tools are particularly effective in combination with geographic information systems (GIS) and numerical models of landscape processes. However, this type of visualization is aimed at a single user and the immersive environments can lead to a feeling of separation from the real world and nausea on the part of the user. For effective group discussion, collaboration, and collective decision making, virtual terrain representations require the development of appropriate interaction tools and new multiuser environments, such as the combination of computer-vision based hand and object tracking with augmented and virtual reality.

This project is still in progress but the work done up to now already allows us to draw methodological conclusions for the creation of a virtual reality system based on the use of digital elevation models with high accuracy and detailed data. The entire reconstruction of the virtual environment is next to be achieved. Further improvements to the system will be: simplifying the entire process of converting DEM files into 3D files, creating an application for procedurally generating additional virtual elements like trees, vegetation, panoramas and light.

8. References

1. *Mitasova, H.; Mitas, L.; Ratti, C.; Ishii, H.; Alonso, J.; Harmon, R.S. - Real-time landscape model interaction using a tangible geospatial modeling environment, Computer Graphics and Applications, IEEE Volume 26, Issue 4, July-Aug. 2006 Page(s):55 – 63;*
2. *Coucelo, C.; Duarte, P.; Crespo, R. - Combining 3D Solid Maps with GIS Data Video Projection, Proc. 2nd Int'l Conf. Geographic Information GIS Planet, 2005;*
3. *Hengl, T.; Gruber, S.; Shresta, D.P. - Digital Terrain Analysis in ILWIS, ITC, Enschede, The Netherlands, 2003;*
4. *Ratti, C. - Tangible User Interfaces (TUIs): A Novel Paradigm for GIS, Trans. GIS, vol. 8, no. 4, 2004, pp. 407-421;*
5. *Chen, S. C.; Zhang, K.; Chen, M. - A Real-Time 3D Animation Environment for Storm Surge, Proceedings of the IEEE International Conference on Multimedia & Expo (ICME), vol. I, pp. 705-708, Baltimore, MD, USA, July 6-9, 2003;*
6. *Marinis, G. - A Digital Terrain Model for the management of water distribution piping nets: a case study, Urban Remote Sensing, poster, 2007;*
7. *Sathyamoorthy, D. - Computation and Characterization of Surface Roughness of Catchments extracted from Simulated Digital Elevation Models, Journal of Applied Sciences Research, 3(12): 1969-1978, 2007;*
8. *Li, Z.; Zhu, Q.; Gold C. - Digital Terrain Modelling: Principles and Methodology, CRC Press, New York, 2005;*
9. *Tay, L.T.; Sagar, B.S.D.; Chuah, H.T. - Analysis of geophysical networks derived from multiscale digital elevation models: A morphological approach. IEEE Geosciences and Remote Sensing Letters, 2(4): 399-403, 2005;*
10. *Sathyamoorthy, D. - Computation of surface roughness of mountains extracted from digital elevation models, Journal of Applied Sciences, 8(2): 262-270, 2008;*
11. *Radhakrishnan, P. - Discrete Simulation, Spatial Modelling and Characterization of Certain Geophysical Phenomena, Ph.D. dissertation, Multimedia University: Malaysia, 2004;*
12. *Martynenko, A.I. - Digital Earth based on Metadata Electronic Maps Standard, Proceedings, 20th ICA International Cartographic Conference, vol. 4, pp. 2747-2752, Beijing, 2001;*
13. *Sharavin, A.A.; Martynenko, A.I.; Vorobiev, R.A. - The National Cartographic Corporation is open for collaboration on the Electronic Earth, Proceedings of Digital Earth, Fredericton, 2001;*
14. *Ciută, M. - An eneolithic cultic pit (bothroy) discovered at Şeuşa – Gorgan (Alba County), 2006*
15. *Buda, V.; Iancu, A. R.; Kadar, M. - Elemente de modelare cu UML în realitatea virtuală. Tehnici de vizualizare a obiectelor virtuale, "Sesiunea de comunicări ştiinţifice studenţeşti a Universităţii "1 Decembrie 1918" Alba Iulia, Buletinul ştiinţific nr. 7, ISSN 1583 – 6088, Ed. Aertenitas", 2008;*
16. *Huang, H.C.; Hung, P. - Panoramic stereo imaging system with automatic disparity warping and seaming, Graphical Models and Image Processing, 60(3):196–208, 1998;*