Tree on Mars, an Immersive Virtual Reality Experience

Paulo Ricardo Duarte
MEI-M ISCTE-IUL – Lisboa, Portugal
Biodroid Productions, SA
pauloduartewalac@gmail.com

Miguel Sales Dias
ISCTE-IUL – Lisboa, Portugal
MLDC, Microsoft Language Development Center – Porto Salvo
miguel.dias@microsoft.com

1. INTRODUCTION

Tree on Mars is an Immersive Reality project, with the following story: the viewer is an astronaut who travels to Mars aiming to examine a tree found in the center of a crater. The project is to be viewed in the CaveH platform located at Centro de Ciência Viva at Lousal, and was developed in the context of the 3D programming curriculum of the Masters of Informatics Engineering - Multimedia of ISCTE-IUL in Lisbon. By definition, a Cave is an Immersive Virtual Reality (VR) system, which projects stereoscopic images with real scale dimensions [1]. In the CaveH case, the set-up is of a parallelepiped form and each face is a planar projection image, where the 3D effect comes from its stereoscopic projection.

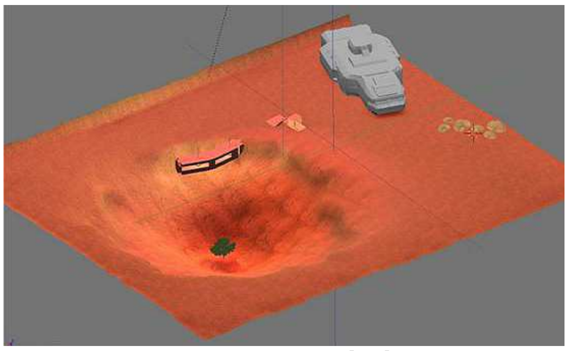
Figure 1: Tree on Mars landscape

The scenario of Mars has one base facing the tree, a spaceship, a rock and an Explore Mars Rover, as shown in Figure 1. In this poster we will highlight the tools and 3D authoring techniques appropriate for real-time 3D rendering, in large scale immersive virtual reality settings. We will illustrate the various stages of development and the achieved results.

2. AUTHORING CONTENT FOR CAVEH

The CaveH content authoring and development process goes through several phases. The process starts by modeling the 3D objects in an authoring tool such as Blender [2]. Then, the content is exported to the Open Scene Graph (OSG) [3] format, to be inserted in the respective folders of GTKRadiant [4], the free game engine used in the CaveH content authoring environment. GtkRadiant is able to author and run the 3D scenario, where the OSG files are included, with many authoring possibilities, like setting Newton physics simulation, including collision detection and response between 3D objects, key-frame forward kinematic animation of characters and virtual camera and visual effects proprieties, such as sky-dome and particle systems and others. After the authoring is completed in GtkRadiant, CaveHSpawner compiles the map, 3D models and their proprieties in an OSG scene graph and, upon error free compilation, generates an octree-based volumetric organization of the scene-graph, with Level of Detail nodes for the more complex 3D models, ready for real-time rendering. Finally, the user is able to run the CaveH runtime viewer application (in a single projection system such as a PC or a multi-projection setting, such as a CAVE), with the compiled map. OpenSceneGraph is the adopted open source 3D graphics scene graph and toolkit, specially developed for real-time rendering or large and complex 3D scene graphs. The OSG models can be generated in two different ways:

1. By producing OSG objects from scratch with the C++ programming language using, for example, Microsoft Visual Studio C++ development environment.
2. By exporting 3D models from Blender with OSG Exporter Script [2], based on the Python language. This script transforms Blender models into OSG objects.

The second method was the chosen option. All models were created in Blender and then exported to OSG objects. The content was developed on a computer with the following specs: 1.60GHZ 2x CPU, 1 GB Ram, and NVIDIA GeForce 7900 GT/GTO 512 MB graphic card. As tools, we have used the following one: Blender, Photoshop, GtkRadiant 1.5 and CaveHSpawner, under MS Windows.

2.1 SETTING-UP BLENDER FOR REAL-TIME

Blender is an open source 3D content creation tool that uses Python as its scripting language. Blender was primarily designed for making non real-time scenes, rendered with a ray tracer. It has a different way of processing material properties and lights, which doesn't quite fit with real-time formats, like Collada and OSG own format. For real-time rendering we need to understand how to set resources such as: materials, normals, transforms and reference frames:

- Materials: applying materials for real-time use is different than for non real-time applications. It is better to use one material per model; this stands for textures as well. Additionally, the texture has to be mapped onto the object using UV coordinates.
- Normals: all normals should face outwards. If not, the polygons will not be shown in the 3D view.
- Transforms and reference frames: without care, an object will spin around its local reference frame. Values can be set for Location, Rotation and Scale. If Rotation gets non-zero values, this object is going to

rotate the way it is expected at runtime. All of this can be adjust and fixed by using Apply Scale and Rotation in Blender.

2.2 REAL-TIME TECHNIQUES

To map correctly the geometry we have used some techniques normally adopted in real-time productions. In the Mars landscape we have adopted a common technique which raises soil with a Height Map. A Height Map [5] is a grayscale image, where pure white represents the highest points and pure black the lowest points in the set. The Height Map is then applied onto a subdivided plane, where the ground is raised or lowered by controlling the grayscale vertices.

As for texturing, this is a relatively complex task. First comes the UV mapping of the 3D models. Each object produces shadows and reflections. These are calculated by baking everything – the texture, shadows, shading and ambient occlusion – in one simple UV texture. It is critical to set appropriately the light and the ambient occlusion in this phase.

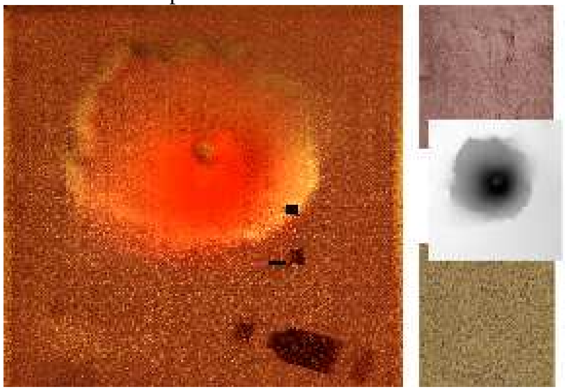

Figure 2: The left image is the result of the Stencil technique used in the right image.

For mapping the Mars soil we have used a Stencil technique [5]. This method uses 3 textures to blend 2 via the grayscale of another, as shown in Figure 2. The upper texture blends dark areas of grayscale Stencil and the lower texture blends white areas (the blend method used is overlay). The final texture (left in Figure 2) shows some after effects made on Photoshop. This grayscale image was also used for the Height Map. There are specific settings in the textures that must be considered to obtain the maximum quality of the OSG files [3]. To have alpha channel working well on OSG objects, the texture must have 32-Bit color and has to be saved in a format that holds the alpha channel, like .png or .dds. The .dds file was the chosen format for two reasons: first, this is the format for videogames proposed by Nvidia [6] and second, this format can save the same detail in less disk space than .png.

2.3 GTK RADIANT

After having exported and saved all 3DS and OSG files into their own GtkRadiant folders, the next step was the creation of the scene. First, all the collider parameters were interactively defined for each 3D object and, whenever necessary, Newton physics was enabled. Next, we have included a representation of a first-person view, where it is possible to modify a number of human scales, motion parameters and fog options. We have also included a light and a skybox, which simulates the environment with dynamic clouds. A particle system allowed us the insertion of flying leaves over the tree. Lastly, we have used a trigger on the first-person view location to load a movie in the beginning of the interaction and a sound listener to set on the environmental music. Finally, GtkRadiant can save a map file (with all the authoring changes made), that will be loaded in another tool, the CaveSpwaner, for compilation and run-time execution.

3. CONCLUSIONS

The development of 3D content for real-time rendering is far different than for traditional ray-tracing rendering. For real-time uses, there are a number of techniques that requires different modeling skills and some degree of originality. In real-time 3D rendering, we have to simulate shadows and reflections by using texture baking, we need to be aware of normals and polygons counts of our 3D models and, last but not least, we need to have in mind the level design related with the storyboard. In real time, there is not a single camera to guide us by the desired key-frame spots. Here the camera is the player that has the possibility to see what he/she wants, so the level needs a trustworthy environment and a world limit must be imposed. The CaveH multi-projection system is a brand new environment in Portugal for Immersive Virtual Reality. A field of view of 180º provides more credible immersions and makes the player believe that he or she is in a real scale landscape. In this poster, we have described an immersive virtual reality experience carried in the CaveH of Lousal, highlighting specially the content authoring techniques that are more appropriate for real-time rendering. The experience allowed the learning of some concepts about Scene Graph development, the correct parameterization of modeling techniques for real-time image synthesis and the integration of some classical animation techniques that, in spite of being largely used in the 90's for computer graphics, are still in use today, namely in gaming, like action triggering and key-frame animation. Perhaps very soon the entertainment will be projected on entire room walls with the same classic 90's video game techniques.

4. REFERENCES

[1] Bastos, P., Dias, M. S., "Experiência de Realidade Virtual Imersiva no Ambiente CaveHollowspace do Lousal", Proceedings of Interacção 2008 – 3ª Conferência Interacção Pessoa-Máquina, Universidade de Évora, 15 – 17 Oct 2008.

[2] Blender. <http://www.blender.org/> and OSG Script. <thttp://vterrain.org/Doc/Blender/>

[3] Martz, P., *OpenSceneGraph - A Quick Introduction to the Cross-Platform OSG API.* Computer Graphics Systems Development Corp., 2007.

[4] GtkRadian. <http://qeradiant.com/cgi-bin/trac.cgi>

[5] Luke A., *3D Game Textures*, 2ª Edition. *Focal Fress,* 2009, 296-319.

[6] NVIDIA DDS Plug-in for Adobe Photoshop. <http://developer.nvidia.com>