

Port of the OGRE 3D Engine to the Pocket PC Platform

João Paulo Silva do Monte Lima¹, Thiago Souto Maior Cordeiro de Farias¹, Veronica Teichrieb², Judith Kelner¹

¹Universidade Federal de Pernambuco, Centro de Informática
CP 7851, 50732-970, Recife, Pernambuco, Brasil
Telefone +55 81 21268954, Fax +55 81 21268955, {jpsml;mouse;jk}@cin.ufpe.br

²Universidade de Pernambuco, Escola Politécnica de Pernambuco,
Depto. de Sistemas Computacionais

Rua Benfica n° 455, Bairro Madalena, 50720-001, Recife, Pernambuco, Brasil
Telefone +55 81 21268954, Fax +55 81 21268955, vt@dsc.upe.br

Abstract. *This paper presents a port of the OGRE 3D rendering engine to the Pocket PC platform, largely used by mobile devices such as handhelds and smartphones. OGRE is a full-featured open source library for developing applications that make use of real time 3D graphics in any possible way. The original source code has been modified to make OGRE work on the Pocket PC platform, and further optimizations have been done to increase the runtime performance as well. The engine has been tested on a real mobile device with sample applications, using two different graphics APIs available to the Pocket PC. The obtained results were then compared with the ones collected from the desktop version of the library.*

1. Introduction

The development of 3D applications usually becomes a complex project that demands a set of reliable and efficient tools for being completed. One of these tools is a graphics engine that is responsible for properly rendering the scenes needed in the application. Graphics engines should have plenty of features that make the developer job easier, and at the same time they should still present a good overall performance. Nowadays, there are diverse application areas that require a 3D interface, such as simulations, games and virtual and augmented reality projects. One of the latest trends for this kind of project is targeted to mobile and embedded devices, which represent a challenge, due to the limited resources present on those systems. The existence of a high-level graphics engine for mobile devices is extremely useful for a better and faster development experience.

This paper details the process of porting OGRE (Object-oriented Graphics Rendering Engine), described in [1] and [2], to the Pocket PC platform, which is currently used by many mobile devices. OGRE is an open source solution for the creation of 3D real time applications and its official release supports many operating systems, like Windows, Linux and Mac OS. Moreover, there are ports for other different platforms, such as FreeBSD. OGRE was also

modified to work with several gaming consoles, like Xbox, PlayStation 1, PlayStation 2 and Dreamcast.

The motivation for using OGRE on Pocket PC came from the desire to develop platform independent augmented reality applications in a high-level environment. The same OGRE application can be used on several systems, eliminating the need to build completely different versions for each platform.

Section 2 specifies some libraries used on the authoring of 3D applications for mobile devices. Section 3 talks about the OGRE engine, detailing its features and its architecture. Section 4 explains what was needed to port OGRE to the Pocket PC platform. The tests made on the mobile device and the results obtained are described in section 5. Finally, section 6 shows the conclusions of this work and presents some future work to be done on the port.

2. Related Work

Recently, the growth of the handheld, PDA (Personal Digital Assistant) and smartphone markets has been creating a demand for libraries that make possible the development of applications for mobile devices that use 3D graphics. The most popular library for this purpose is OpenGL ES, a limited version of OpenGL targeted to embedded systems [3]. It has implementations for almost every embedded platform, such as Pocket PC, Symbian OS and BREW. There are basically two versions intended to be developed in parallel: version 1.x, designed for fixed function hardware, and version 2.x, specified for programmable hardware. However, most operating systems still do not have an OpenGL ES API (Application Programming Interface) that exploits 3D hardware acceleration. This situation is expected to change with the release of many mobile devices that have an integrated GPU (Graphics Processing Unit). Both OpenGL and OpenGL ES provide a low-level programming API, which decreases the developer productivity.

Another 3D graphics solution heavily used by mobile device developers is Klimt, which intends to provide an interface like OpenGL, addressing some of the functionalities not supported by OpenGL ES [6]. Klimt also contains some optimizations for Pocket PC systems. The available version is open source and uses a proprietary software-based engine. Due to the similarity with OpenGL, the programming level is also very low.

For the Pocket PC platform, the Windows Mobile 5.0 operating system supports Direct3D Mobile, which is based on the Direct3D 8 Desktop release [7]. When combined with the .NET Compact Framework, it can be used in managed mode, providing a secure and easy development environment. On the other hand, its abstraction level is almost the same of OpenGL and OpenGL ES.

In respect of the Java development, there is an API for 3D rendering called M3G (Mobile 3D Graphics API for J2ME), which is detailed in [8]. It is an object-oriented managed library that has two different modes: the immediate mode, using low-level access similar to OpenGL and OpenGL ES, and the retained mode, that utilizes a scene graph, a much higher-level structure for 3D rendering, which is also applied to OGRE, making the coding process more intuitive. The disadvantage of Java applications is the low runtime speed, which decreases the frame rate.

3. OGRE

There are some features present in OGRE that can be considered the main reasons to its success. One of them is the object-oriented approach, which contributes to a more structured and dynamic development.

The fact that OGRE is made up by classes with well-defined interfaces between each other favors the implementation abstraction of different parts of the engine. There is not a dependence on how the modules responsible for performing native operations and 3D rendering are coded. Due to this, OGRE becomes a platform independent engine, being possible to port it to a wide range of environments.

OGRE is concerned to be a flexible engine, capable of working together with other external tools, like physics and artificial intelligence libraries. Another advantage of OGRE is the usage of scripts written in text format for loading the application settings. The developer is then able to change configurations like objects' textures and materials without having to recompile the program.

The architecture of OGRE is shown in Figure 1. The main class of OGRE is the `Root` class, responsible for the instantiation and access to all other classes of the engine. OGRE has management classes for each element of the system. A `ResourceManager` supervises all the resources available for the application. There are many types of `ResourceManager`, like `TextureManager`, `FontManager` and `MeshManager`. The class responsible for the organization of the scene elements is called `SceneManager`. The `SceneManager` has a reference to a `RenderSystem` object, which is in charge of rendering the scene using a graphics library. Other important class is the `PlatformManager`, which encapsulates all the functionalities that are native to the operating system aimed by the application, such as input handling, timing and GUI (Graphical User Interface) management.

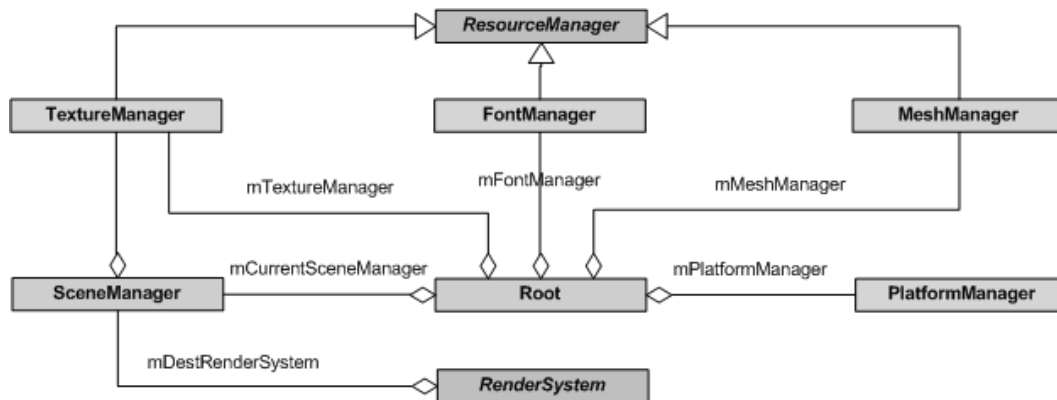


Figure 1. OGRE architecture

4. OGRE Port for Pocket PC

The OGRE version for Pocket PC ported by the authors was based on version 1.1.0 alpha (Dagon) of the OGRE official source code. The first task performed to make OGRE available for the Pocket PC platform was to port all the dependencies of OGRE that do not have a

working version for the Pocket PC platform, and extend the libraries that supported it to supply the needed functionalities, such as a more complete file handling. These dependencies are libraries used by the engine for specific tasks, like texture loading and file management. The second step consisted in implementing a `RenderSystem` using a graphics API available for Pocket PC, in this case OpenGL ES. After that, a `PlatformManager` for the operating system had to be written. Then, targeting better performance results, OGRE was adapted to work with fixed-point math routines, since mobile devices do not have an FPU (Floating Point Unit) and all hardware-based math operations are done using fixed point. Finally, it was implemented another `RenderSystem` using Klimg, trying to get higher frame rates.

4.1. Dependencies

The libraries that did not have a working version for the Pocket PC platform were `STLPort`, `DevIL`, `FreeType2`, and `ZzipLib`. Along with their purpose, some details on how the port process was done are described next.

`STLPort` provides the developer with all the classes present in the C++ STL (Standard Template Library) [9]. STL is heavily used by the OGRE dependencies and OGRE itself, but the current development environments for Pocket PC do not offer a complete implementation of it. The IDE (Integrated Development Environment) that implements the greatest number of STL features is Microsoft Visual Studio 2005, so the ported `STLPort` version was designed to work under that environment. The Pocket PC version of `STLPort` was based on the release 5.0 RC6, which had some small changes. Basically, it was needed just to add an additional header file to the library source code to set up the IDE configurations.

`DevIL` consists in a library responsible for texture loading [10]. It supports several image formats, like bmp, gif, ico, jpeg, png, psd and tga. `DevIL` for Pocket PC was built based on version 1.6.7 of the library source code. The modifications made on the original library were almost all related to differences on function signatures. On the Pocket PC platform, the strings are encoded using wide char standard (Unicode), in which every character is represented by 2 (two) bytes. The format used in most of the desktop applications is ASCII, that uses 1 (one) byte to encode a character. Some functions only receive strings in the Unicode format and some others have a version for each encoding. As a result, the string encoding has to be checked to ensure that the correct function is being called.

`FreeType2` is used to load fonts in many formats, such as TrueType and OpenType [11]. All the text elements in OGRE scenes rely on this library for defining its layout. After configuring `STLPort` on the Pocket PC environment, the version 2.1.10 of `FreeType2` library source could be built without any changes.

`ZzipLib` is a library that makes it possible to access files that are compressed in zip packages [12]. The application resources can then be packed in one or more zip files, being easier to distribute it later. The library also creates an abstraction about how the resource is stored on the disk. Files that belong to the filesystem and files that are zipped are treated the same way. For example, when the application tries to open a file, the library searches for it in the path supplied. If it is not found, then it searches in all zip files stored in the path. Version

0.10.82 of `ZzipLib` was used on the Pocket PC platform, which was built properly after having `STLPort` configured in the development environment.

The libraries that already had a working version for the Pocket PC platform and were extended to supply the needed functionalities are `LibCE`, `Zlib` and `OpenGL ES`. Along with their purpose, some details on how the porting process was done are described next.

`LibCE` is a library created to fill in some gaps that are present in the `libc` library for Pocket PC, which does not implement many functions that are present in the Win32 version, like system time access [13]. The version 1.0 of `LibCE` was the starting point to the library used by OGRE. Some procedures were added for file handling with Unicode parameters and other extra features that were not available in the library, like search, move, delete and rename files.

`Zlib` implements the compression algorithm used in the `.zip` file format [14]. OGRE uses `Zlib` to uncompress zipped resources and make them available for use. There was a version of `Zlib` for Pocket PC based on build 1.2.1 of the library. Some small corrections were made and it was then successfully compiled in the IDE.

`OpenGL ES` is the graphics API used in the OGRE version for Pocket PC. The implementation applied to the port is called Vincent, which does not make use of hardware processing [4]. The version 0.84 of the library served as a base for the project. The changes made in the library source code were related to pixel buffer functionalities, which were not working properly for the rendering process used in OGRE.

4.2. RenderSystem using OpenGL ES

In the OGRE source code distribution for Win32, there are three implementations of the `RenderSystem`, each one using three different graphics libraries: `OpenGL` and `Direct3D` versions 7 and 9. Since none of these libraries supports Pocket PC, there was a need for implementing a `RenderSystem` using a graphics library available for this platform.

The library chosen was `OpenGL ES 1.1`, because of its similarity with `OpenGL`, allowing the former `RenderSystem` to be adapted without major changes. The Vincent library, an open source implementation of `OpenGL ES`, was used in the project. The current version is complete software-based, using no hardware acceleration, which results in some performance issues. The Hybrid `OpenGL ES` implementation for Win32, described in [5], has also been used, because the implemented `RenderSystem` was first tested in the desktop environment and then used in the Pocket PC platform.

The attempt to make `OpenGL ES` as small as possible to fit the constraints of embedded devices lead to the lack of many functionalities present in `OpenGL`. Due to this, actions of the OGRE `RenderSystem` that require, for example, 1D or 3D textures or cube maps will not be performed and will raise a “feature not supported” exception. `OpenGL ES` also have other limitations, such as the inability to read the contents of a texture and to use 32-bit indices for rendering polygons.

The initial OpenGL Win32 `RenderSystem` project has been modified in turn to use OpenGL ES. First, all the code related to shaders has been removed, as this functionality requires GPU processing and is not available in Vincent.

The `RenderSystem` startup function was changed to initialize the capabilities correctly. Vincent does not support some features like hardware mipmapping, anisotropy, cubemapping, vertex and fragment shading.

Many of the OpenGL ES function signatures are equivalent to OpenGL, contributing to decrease the number of changes needed in the original code. Some function names have little differences, like `glClearDepth` (OpenGL) and `glClearDepthf` (OpenGL ES). The `RenderSystem` also used extensions to OpenGL, like GL ARB, which were substituted for the equivalent OpenGL ES functions. The code relative to window creation and management, which is platform specific, contains some calls to WGL procedures, the OpenGL API for Win32 native support. They were all replaced by functions from EGL, the OpenGL ES interface responsible for the window system.

The GLU (OpenGL Utilities) library is used in the `RenderSystem` for creating texture mipmaps. Since GLU for OpenGL ES is not available, the SGI GLU implementation for OpenGL was adapted to work with OpenGL ES.

4.3. PlatformManager for Pocket PC

After extending the `LibCE` library, all the missing Win32 functions that were needed on the Pocket PC API were available for use. For that reason, the `PlatformManager` for Pocket PC consisted in an adaptation of the Win32 version.

The `PlatformManager` consists in five classes: `ConfigDialog`, `ErrorDialog`, `Timer`, `Input` and `PlatformDll`.

`ConfigDialog` is responsible for showing the configuration options to the user when the application starts, allowing him to change options like color depth, display frequency and screen resolution.

`ErrorDialog` is in charge of warning the user about any error that occurs during application execution. The exceptions raised by OGRE are very clarifying, pointing out the file and the line where the error occurred, along with a detailed description of what happened.

`Timer` provides functions for accessing the operating system time. It is very useful for several actions taken by the engine, such as measuring the frame rate of the application.

`Input` captures the interaction made by the user using any input device, like the hardware buttons and the pen.

`PlatformDll` acts as an interface for all the functionalities implemented by the `PlatformManager`, accessing the respective methods for each one.

These classes remained almost the same in the `Pocket PCPlatformManager` as in the Win32 platform, with minor changes related to Unicode string conversion.

The `Input` class demanded a more careful work, since the user interaction on the desktop environment is completely different from a mobile device. When sitting in front of a

PC, the user commonly has a keyboard with 101 keys and a mouse with two or three buttons. In many mobile devices, most of the times the only way the user can interact with the system is using a few hardware buttons and the pen. Some PDAs and smartphones have a small keyboard with letters, digits and punctuation.

In this project implementation, the interaction done by the keyboard has been mapped to the hardware buttons of the device and the pen works as a mouse. The default interaction model of an OGRE application is detailed in Tables 1 and 2, alongside with the mapping between desktop and mobile device input. Some keyboard keys do not exist on embedded devices and could not be mapped. However, if the developer has any other available keys that are not being used on his application, he can assign the desired behaviour to them. About the mouse to pen mapping, the movement can be related to the pen being dragged over the touch sensitive display of the device. Because the mouse right button cannot be related directly to any input, the developer has to choose a key to perform the action.

Table 1. Keyboard input mapping of OGRE applications on mobile devices

Desktop Input	Mobile Device Input	Default Action
Up Arrow Key W Key	Up Hardware Button W Key (if present)	Move camera forward
Down Arrow Key S Key	Down Hardware Button S Key (if present)	Move camera backwards
Left Arrow Key A Key	Left Hardware Button A Key (if present)	Move camera to the left
Right Arrow Key D Key	Right Hardware Button D Key (if present)	Move camera to the right
Page Up Key Page Down Key	Not mapped Not mapped	Move camera up Move camera down
F Key	F Key (if present)	Hide/Show overlays
R Key	R Key (if present)	Change rendering mode
P Key	P Key (if present)	Change texturing filtering
T Key	T Key (if present)	Hide/Show camera coordinates

Table 2. Mouse input mapping of OGRE applications on mobile devices

Desktop Input	Mobile Device Input	Default Action
Mouse movement	Pen dragging	Move camera
Movement + right button pressed	Not mapped	Rotate camera

On the Pocket PC PlatformManager polling captures the keyboard and pen input. For every frame that is rendered, keyboard and mouse state are tracked and the results are returned to the application. The program can check if a key is pressed or not or if the pen is touching the screen, as well as verify how much pixels the pen moved over the screen.

4.4. Fixed-Point Math Implementation

Aiming optimization purposes, the former OGRE source code has been changed to be able to use fixed-point math in its calculations. The extensible architecture of OGRE provided a clean way to do this, since the type that is going to be used to perform math operations can be chosen by the developer at compilation time, modifying just two or three lines of code.

Table 3 describes the math representations used by OGRE and the default types responsible for implementing them on desktop and mobile device platforms. The Real type represents a real number, the Real32 type represents a 32-bit number and the RealGPU type represents the number used in GPU math operations. On the mobile device, a FixedPoint math class represents the Real and Real32 types. Initially, the class used the Intel GPP (Graphics Performance Primitives) library for Pocket PC, which intends to provide the appropriate math functions [15]. On the other hand, it uses a 32-bit buffer to perform the operations, causing overflow when a large scene has to be rendered. Due to this, the FixedPoint class uses a 64-bit buffer to operate, supporting more complex scenes and more accurate math during computations. However, OpenGL ES fixed point size is 32-bit. Therefore, after finishing the results computation, the FixedPoint class has to reduce them to 32-bit. The RealGPU type is represented by the float type on both platforms, since OpenGL ES still uses floats for GPU related operations, like geometry buffers.

Table 3. OGRE math types mapping on mobile devices

Math Representation	Desktop Type	Mobile Device Type
Real	double	FixedPoint
Real32	float	FixedPoint
RealGPU	float	float

The next step performed was to correct some modules of OGRE that used the floating-point type directly, instead of the Real type. This was necessary to ensure that all parts of the code were using the correct math definition.

After that, it was created a version of the RenderSystem based on OpenGL ES using fixed-point math. Therefore, there are two different RenderSystem implementations available for the Pocket PC platform: a floating-point version and a fixed-point math. Both of them use OpenGL ES as the graphics API for rendering operations.

4.5. RenderSystem using Klimt

Klimt is an open source 3D library for mobile devices that is very similar to OpenGL and OpenGL ES. It does not use hardware acceleration, so the complete rendering process is performed by software. Nevertheless, it tries to be as efficient as possible on the Pocket PC platform.

The strategy used by Klimt for speeding up the applications is based mainly on using the Intel GPP library for fixed-point math and the PocketHAL library for linear frame buffer implementation, which is actually faster than the Pocket PC native API [16].

The Klimt RenderSystem was built out of the OpenGL ES RenderSystem, given that the function calls and the whole architecture of the both are almost the same. The project was first tested on the Win32 platform, supported by Klimt, and then validated on the Pocket PC.

5. Case Study and Results

Some applications have been used to validate the OGRE port on a Pocket PC. The mobile device used for testing the OGRE applications was the PDA HP iPAQ H5500. It has a 400

MHz Intel PXA255 XScale processor, 128 MB of RAM, 48 MB of ROM and a LCD display with 16 bit color depth and 320 x 240 pixels. The operating system is Microsoft Windows Mobile 2003 (version 4.20.1081 build 13100). The desktop computer used to test Win32 OGRE applications has a 2.80GHz Intel Pentium 4 processor, 512 MB of RAM, an Intel 82865G video adapter with 64 MB of memory and a screen resolution of 1024 x 768 pixels. The development tool utilized to implement the project was Microsoft Visual Studio .NET 2005 Professional Edition IDE (version 8.0.50727.42 RTM.050727-4200).

The applications used for verifying if OGRE was working properly were the demos SkyBox and SkeletalAnimation, which are part of the official distribution of OGRE. The code of the examples suffered minor modifications, related to the Pocket PC GUI API, whose function signatures are different from the ones for Win32.



Figure 2. Screenshots of OGRE demos on Pocket PC: a) SkeletalAnimation using OpenGL ES RenderSystem b) SkyBox using OpenGL ES RenderSystem c) SkeletalAnimation using Klimt RenderSystem

The evaluation metrics applied to the tests were the following: frame rate, image quality and feature availability. The illusion of movement is completely related to the frame rate, which is measured by the number of frames that are rendered in one second (frames per second, or fps). The higher the frame rate of the application, the better the resulting animation. The way the polygons and textures are shown on the screen is very important to the realism of the scene. The measurement of the image quality is heavily influenced by the resolution of the display and the algorithms used by the rendering API. The mobile application should have as much functionalities present in the desktop version of OGRE as possible. However, some features will not be accessible because of platform constraints, like the absence of GPU processing.

Figure 2 shows some screenshots of the sample applications SkyBox and SkeletalAnimation running on the Pocket PC, using OpenGL ES and Klimt RenderSystem implementations.

The library benchmark was done using the SkeletalAnimation demo. The frame rates obtained on Pocket PC are compared with the desktop version performance, and presented in Figure 3. The values were obtained from scenes with varied triangle number and different RenderSystem versions were used (OpenGL and Klimt on desktop, OpenGL ES and Klimt on mobile device). The OpenGL ES RenderSystem for Win32 was not compared, because it uses Hybrid OpenGL ES implementation, while Vincent is used on Pocket PC.

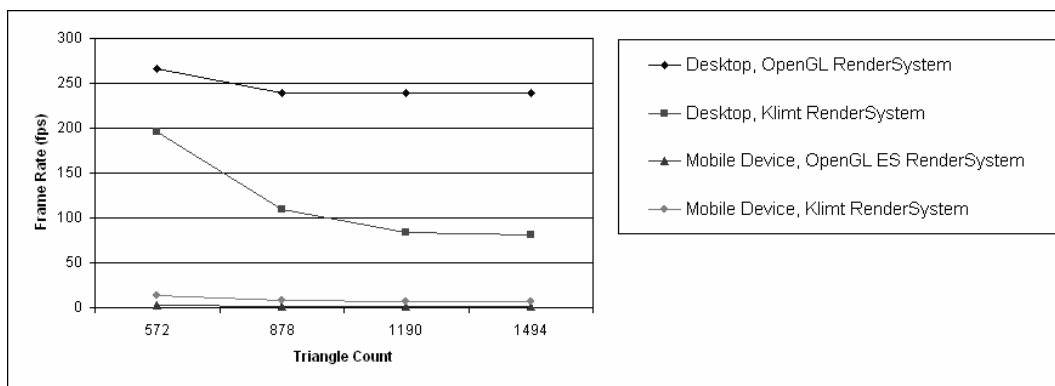


Figure 3. Frame rate comparison between desktop and mobile device platforms

In Figure 4, the Pocket PC results with different RenderSystem implementations are put side by side. The frame rate of the demo using OpenGL ES RenderSystem on the Pocket PC ranged between 1.057 and 2.814 fps, what is very low for a 3D real time graphics application. The main reason for this poor result is the fact that Vincent rendering is emulated by software, while for the desktop version the GPU is used. The result obtained using Klimt RenderSystem, which varied between 6.561 and 13.390 fps, is much better than the one from the OpenGL ES version, due to the optimizations present in the graphics library. However, it is far from the frame rate obtained with desktop configurations.

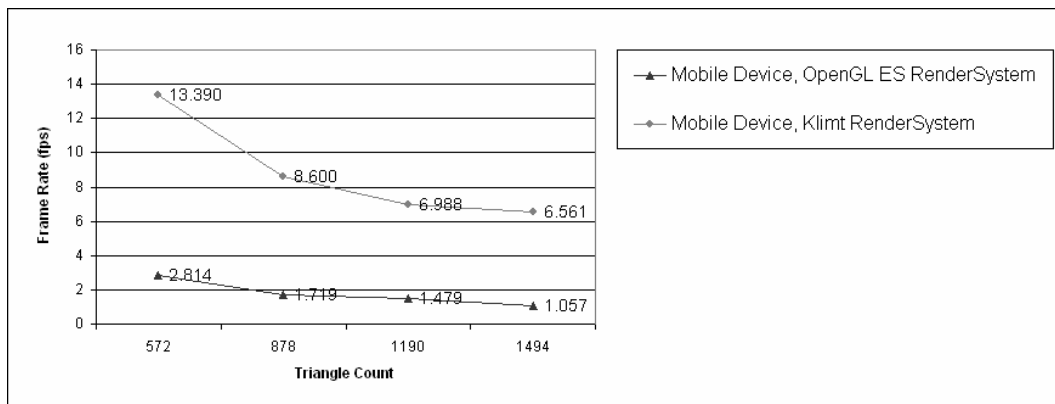


Figure 4. Frame rate on mobile device using different RenderSystems

The low resolution of the device display caused some aliasing on the rendered objects, even though their aspect is still very acceptable.

Finally, there are some features from the desktop version of OGRE missing in the Pocket PC environment, like overlay elements, that were supposed to be drawn over the

image bringing information about the rendered scene, such as number of polygons rendered, frame rate and others.

6. Conclusions and Future Work

The OGRE port for the Pocket PC platform brings to the embedded devices developer the opportunity to take advantage of the high-level features of 3D rendering engines. Additionally, the implementation of a `RenderSystem` based on OpenGL ES opens up the possibility of having OGRE ported to other environments, like the mobile gaming console PlayStation Portable (PSP).

As future work, there are plans to implement a `RenderSystem` based on Direct3D Mobile, which is already supported by a few handhelds that support 3D hardware acceleration, leading to a better frame rate. Another practice for optimizing the runtime speed is to use the Intel C++ Compiler for Pocket PC, which is specific for the Intel XScale processors and provides a great performance improvement, according to Wagner et al. [17]. Some bug fixing is still needed, since some features are not working on the current version of the engine, like overlays. Furthermore, as soon as a stable and documented version is developed, this project is going to be available for the general public, as an OGRE add-on.

Finally, it is going to be applied some usability tests to evaluate how good is the user interaction with the system and how it can be improved.

7. Acknowledgement

The authors want to thank Matthew Scott from Manifest Games, for architecting the OGRE port and leading the development of the project.

References

- [1] T.S. Farias, S.A. Pessoa, V. Teichrieb, and J. Kelner, "Tutorial Engine Gráfico Ogre", *IV Brazilian Games and Digital Entertainment Workshop*, São Paulo, November 23-25, 2005, p. 1-13.
- [2] OGRE 3D: Open Source Graphics Engine. Available: The OGRE Team site. URL: <http://www.ogre3d.org>, visited on January 2006.
- [3] OpenGL ES Overview. Available: Khronos Group site. URL: <http://www.khronos.org/opengles/index.html>, visited on January 2006.
- [4] Vincent, the 3-D Rendering Library for Pocket PCs and Smartphones based on the Published OpenGL. Available: SourceForge site. URL: <http://ogl-es.sourceforge.net>, visited on January 2006.
- [5] Hybrid Graphics Ltd – Developer Tools. Available: Hybrid Graphics site. URL: <http://www.hybrid.fi/main/products/devtools.php>, visited on January 2006.
- [6] Klimt – the Open Source 3D Graphics Library for Mobile Devices. Available: Graz University of Technology site. URL: <http://studierstube.icg.tu-graz.ac.at/klimt>, visited on January 2006.
- [7] Direct3D Mobile. Available: Microsoft Developer Network site, URL: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wcemultimedia5/html/wce50oriDirect3DMobile.asp>, visited on January 2006.
- [8] Q.H. Mahmoud. Getting Started with the Mobile 3D Graphics API for J2ME. Available: Sun Microsystems, Inc. site. URL: <http://developers.sun.com/techtopics/mobility/apis/articles/3dgraphics/index.html>, visited on January 2006.
- [9] STLport. Available: STLport site. URL: <http://www.stlport.com>, visited on January 2006.

- [10] DevIL – a Full Featured Cross-Platform Image Library. Available: SourceForge.net site. URL: <http://openil.sourceforge.net>, visited on January 2006.
- [11] FreeType2 Overview. Available: SourceForge site. URL: <http://freetype.sourceforge.net/freetype2/index.html>, visited on January 2006.
- [12] Zziplib – the Library. Available: SourceForge site. URL: <http://zziplib.sourceforge.net>, visited on January 2006.
- [13] Handheld Augmented Reality. Available: Graz University of Technology site. URL: http://studierstube.icg.tu-graz.ac.at/handheld_ar, visited on January 2006.
- [14] zlib Home Site. Available: Zlib site. URL: <http://www.zlib.net>, visited on January 2006.
- [15] Intel Graphics Performance Primitives Version 4.0 for Microsoft Windows Mobile 2003 Software. Available: Intel Corporation site. URL: http://www.intel.com/design/pca/applicationsprocessors/swsup/gppv40_windows.htm, visited on January 2006.
- [16] PocketHAL – the Hardware Access Library. Available: PocketHAL site. URL: <http://pockethal.droneship.com>, visited on January 2006.
- [17] D. Wagner, and D. Schmalstieg, “ARToolKit on the PocketPC Platform”, *Technical Report*, Vienna University of Technology, Austria, 2003, pp. 1-2.