Projecto “CROSS – Uma infra-estrutura para certificação e re-engenharia de Software Livre”

T2: Combining OSS: Certification of Open Source Software Stacks

Relatório Técnico

Refª BI3-2013_PTDC/EIA-CCO/108995/2008_UMINHO

Aproveitando a disponibilidade de web services que disponibilizam informação 3D (desenvolvidos numa bolsa anterior), pretende-se ver até que ponto se conseguem construir interfaces eficientes para a exploração da rede 2D/3D em dispositivos móveis. No entanto, a visualização nestes dispositivos de dados potencialmente volumosos e complexos levanta algumas questões.

As maiores dificuldades prendem-se com limitações técnicas, relacionadas com o volume e complexidade dos dados. Mas também existem questões de usabilidade, já que não é fácil perceber e interagir com informação geográfica complexa (em 3D) num pequeno dispositivo móvel.

Por estas razões, este projeto começou com um estudo preliminar que visou indentificar as formas mais adequadas de apresentação de informação geográfica em 3D e as funcionalidades requeridas por um utilizador em movimento para alguns cenários. Destas etapas iniciais resultou a escolha da metáfora do globo virtual para a apresentação de informação geográfica em 3D. Após esta decisão, todo o projeto foi orientado para o desenvolvimento de um globo virtual em Android.

Para o desenvolvimento deste globo, era importante ter desde logo em consideração alguns standards para garantir que o globo poderia consumir serviços de uma variedade de fontes e em diferentes formatos.

Índice de conteúdo

Introdução

Neste projeto pretende-se desenvolver um cliente que permita aceder a informação geográfica 2d ou 3D em plataformas móveis. A renderização é feita no dispositivo móvel e não no servidor.

Após as fases anteriores de levantamento de requisitos, de estudo do estado da arte e da identificação de use cases, reportam-se agora os trabalhos relacionados com a visualização de 3D e com o porting do osgEarth para Android.

Visualização de modelos 3D no cliente

Para proceder ao *port* da framework para globos virtuais *osgEarth* foi necessário proceder também ao *porting* das *libraries* auxiliares às quais esta recorre. De referir as *libraries* geos, proj, e o motor de renderização *OpenSceneGraph*. Este processo de *porting* é caracterizado pela definição de um conjunto de ficheiros *.mk* relativos à compilação para sistemas Android que vão ser introduzidos no ambiente de compilação *cmake.* Através do recurso a um processo de cross-compiling possível através da ferramenta *ndk-build* são compiladas então *libraries* compatíveis com o sistema Android.

Este processo é então caracterizado pela definição dos ficheiros *.mk*, encontrando-se o principal desafio da operação de *cross-compiling* nas diferenças e limitações do ambiente de compilação Android. Estas limitações são mais intensas e comuns neste caso particular, uma vez que em Android não se encontra presente a biblioteca OpenGL, estando apenas disponível um seu subconjunto. Em relação aos ficheiros *.mk*, surgem com maior importância os ficheiro *Application.mk e Android.mk.* No primeiro define-se qual a plataforma Android para a qual se vai proceder à compilação e quais os módulos que irão ser compilados. Os ficheiros *Android.mk* contêm a listagem e relacionamento dos ficheiros de código fonte que definem cada módulo.

Um dos requisitos deste projecto identificado nas fases anteriores corresponde ao consumo de modelos gráficos 3D a partir de um servidor remoto de uma forma dinâmica. Este requisito tem como um dos principais desafios a escolha de um formato de representação de modelos 3D a utilizar.

Tomando em consideração o facto de termos controlo sobre o comportamento do serviço W3DS, a escolha do modelo 3D vai ser orientada principalmente pela facilidade e capacidade de suporte do tipo escolhido em relação com a tecnologia de render OpenGL ES 2.0. Esta escolha tem também de preservar a garantia de possibilidade de conversão entre o formato escolhido e os formatos 3D mais comuns.

Neste sentido chegou-se à escolha do modelo de dados *osg*. Este tipo de dados, uma vez que é o formato de representação de modelos nativo do motor de renderização utilizado, introduz uma maior facilidade no *parsing* e representação gráfica dos modelos. Uma outra vantagem inerente à utilização deste formato está relacionada com a relativa simplicidade e autonomia do *plugin* responsável pelo seu *parsing*, este revelou-se um factor muito importante no processo de *porting* para Android.

De maneira a garantir a interoperabilidade entre a aplicação desenvolvida e os mais variados sistemas SIG, incluindo o serviço W3DS já disponível nos nossos servidores, disponibiliza-se um algoritmo de conversão que garante a transformação entre modelos 3D representados em formato COLLADA e modelos definidos no formato escolhido. Adicionalmente foi desenvolvido um método de correcção dos problemas de escala que por vezes se encontram presentes em modelos 3D definidos com auxílio do SketchUp. Para a resolução deste problema, que também pode ser verificado no GoogleEarth em ambiente Linux, foi então implementado um método com base na aplicação de uma matriz de transformação.

Decorrente do consumo de dados provenientes de um serviço OGC estão associados diversos desafios relacionados com a elevada quantidade de informação que estes normalmente comportam. Torna-se assim fundamental a definição e exploração de algoritmos de *caching* de maneira a que este consumo e representação seja comportável. Uma vez que nos dados presentes no nosso servidor existe um elevado número de geometria repetida em diferentes posições estes algoritmos revelam-se ainda mais importantes. Foram assim definidos dois momentos de caching, um para os pedidos efectuados e outro para a renderização dos modelos gráficos. Desta forma os recursos recolhidos nos pedidos aos serviços OGC são colocados numa cache, não se efectuando o download de recursos com caminhos idênticos. Em relação aos modelos gráficos, estes são construídos apenas uma vez, sendo todas as representações presentes em variadas posições geográficas baseadas na mesma estrutura de dados, partilhando também entre elas o mesmo programa de *shading* utilizado no *render* do modelo.

Um globo virtual em Android

Como foi anteriormente referido, o principal desafio no desenvolvimento de um globo virtual para Android encontra-se relacionado com a ausência da API para renderização de gráficos 3D OpenGL. No Android apenas se encontra disponível a sua vertente para sistemas embutidos OpenGL ES, uma vez que esta caracteriza um subconjunto do OpenGL normal, vem introduzir limitações e alterações na lógica de render.

Um dos principais passos para a implementação de um globo virtual para Android foi então a introdução de programas de *shader* compatíveis com o OpenGL ES 2.0 no pipeline de renderização. A definição destes *shaders* tem então de ter atenção às limitações deste ambiente, podendo-se referir como limitações mais importantes a ausência de diversas *fixed functions* como a ftransform() e glMultiTexCoird entre outras. Também se encontram ausentes alguns modos de render e tipos de dados, o que implicou, por exemplo, à alteração dos métodos de construção de modelos dado à ausência do tipo unsigned int.

No processo de *porting* do *plugin* de *kml* foi necessário proceder à substituição da classe responsável por conter o nodo do modelo gráfico. Originalmente esta responsabilidade pertencia à classe da *framework osgEarth, ModelNode.* No entanto a sua incompatibilidade com o OpenGL ES 2.0 levou ao recurso da classe *Node* do motor de renderização *OpenSceneGraph.* A geolocalização do modelo gráfico passa então a ser obtida pela construção de uma matriz de transformação e posterior aplicação da mesma ao modelo.

Um dos principais pontos de destaque das aplicações de globo virtual encontra-se relacionado com a sua iteração simples e intuitiva. As versões deste tipo de aplicações em ambiente móvel também já têm definidos uma série de movimentos *standard*, como o *two-finger drag* para efectuar o *tilt* dos modelos. De maneira a aumentar a familiarização dos utilizadores do nosso globo virtual estes eventos encontram-se também disponíveis tendo sido efectuado o mapeamento dos eventos em Android para os eventos já definidos na *framework osgEarth*.

De maneira a permitir uma maior iteração com os modelos representados foi também implementado um método de *picking.* Este método capta o evento de *double tap* em Android, projectando um segmento de recta com origem no ponto pressionado para o sistema de coordenadas do ambiente de render. É posteriormente mostrada uma *activity* com informações sobre a entidade que este segmento de recta intersectou.

A construção da aplicação Android e das diversas actividades que a comportam foi efectuada com recurso à *framework* JNI (Java Native Interface). Esta providência a interligação entre o código Java e o código nativo, sendo necessário respeitar uma série de normas na estrutura e definição dos diversos métodos, bem como na passagem de estruturas de dados entre os diferentes ambientes.

Compilação

O globo virtual desenvolvido é uma aplicação para Android (.apk). A mesma está escrita maioritariamente em C++. Do código C++ resulta uma library libosgNativeLib.so que é utilizada na geração do osgViewer.apk final.

Requisitos

É necessário, para além do Android SDK, o framework NDK para Android da Google. Usou-se a versão r8b. Para obter e instalar o Andriod NDK, é necessário:

```
cd Android
wget https://dl.google.com/android/ndk/android-ndk-r8b-linux-x86.tar.bz2
```

Após a instalação, é necessário definir duas variáveis de ambiente:

```
export ANDROID_NDK=/home/jgr/Android/android-ndk-r8b
export ANDROID_SDK=/home/jgr/Android/android-sdk-linux
```

A pasta Android utilizada serve apenas para instalar os frameworks SDK e NDK.

A compilação é suportada pelo utilitário cmake, que será utilizado para gerar as necessárias Makefiles. É necessária a versão 2.6.4 ou superior do cmake.

Obtenção do código fonte

O código fonte necessário pode ser obtido a partir do repositório:

https://bitbucket.org/jgrocha/osgearthandroid

Para descarregar o código é necessário:

```
cd
git clone git@bitbucket.org:jgrocha/osgearthandroid.git
cd osgearthandroid/
```

Desta forma, todo o código irá ficar numa pasta osgearthandroid.

Compilação do OpenSceneGraph

A compilação do OpenSceneGraph depende de outras libraries. Antes de se compilar, devem-se descarregar as mesmas na pasta adequada.

```
cd OpenSceneGraph
wget http://www2.ai2.upv.es/difusion/osgAndroid/3rdpartyAndroid.zip
unzip 3rdpartyAndroid.zip

cmake . -DOSG_BUILD_PLATFORM_ANDROID=ON -DDYNAMIC_OPENTHREADS=OFF -DDYNAMIC_OPENSCENEGRAPH=OFF
-DOSG_GL1_AVAILABLE=OFF -DOSG_GL2_AVAILABLE=OFF -DOSG_GL3_AVAILABLE=OFF -DOSG_GLES1_AVAILABLE=OFF
-DOSG_GLES2_AVAILABLE=ON -DOSG_GL_LIBRARY_STATIC=OFF -DOSG_GL_DISPLAYLISTS_AVAILABLE=OFF
-DOSG_GL_MATRICES_AVAILABLE=OFF -DOSG_GL_VERTEX_FUNCS_AVAILABLE=OFF
-DOSG_GL_VERTEX_ARRAY_FUNCS_AVAILABLE=OFF -DOSG_GL_FIXED_FUNCTION_AVAILABLE=OFF
-DANDROID_ABI="armeabi armeabi-v7a" -DANDROID_PLATFORM=8 -DANDROID_STL="gnustl_static" -DJ=4

make
```

Compilação do osgEarth

Esta compilação faz-se em dois passos.

Compilação das dependências para Android

```
# Muda-se para osgearthandroid/3rdparty/jni
cd ../3rdparty/jni
~/Android/android-ndk-r8b/ndk-build
```

Compilação do osgEarth para Android

```
# Muda-se para osgearthandroid
cd ../..
cmake . -DOSG_BUILD_PLATFORM_ANDROID=ON -DJ=4 -DOSG_DIR="./OpenSceneGraph" -DDYNAMIC_OSGEARTH=OFF
-DOPENTHREADS_LIBRARY="./OpenSceneGraph/obj/local/armeabi/libOpenThreads.a"
-DCURL_LIBRARY="./OpenSceneGraph/3rdparty/build/curl/obj/local/armeabi/libcurl.a"
-DGDAL_LIBRARY="./osgearthandroid/OpenSceneGraph/3rdparty/build/gdal/obj/local/armeabi/libgdal.a"
-DGEOS_LIBRARY="./3rdparty/obj/local/armeabi/libgeos.a"
-DSQLITE3_INCLUDE_DIR="./3rdparty/jni/sqlite-autoconf-3071401"
-DSQLITE3_LIBRARY="./3rdparty/obj/local/armeabi/libsqlite3.a"

make
```

Geração do osgViewer

A geração da aplicação para Android faz-se em dois passos. Primeiro gera-se a library libosgNativeLib.so e depois o osgViewer.apk.

Geração da library libosgNativeLib.so

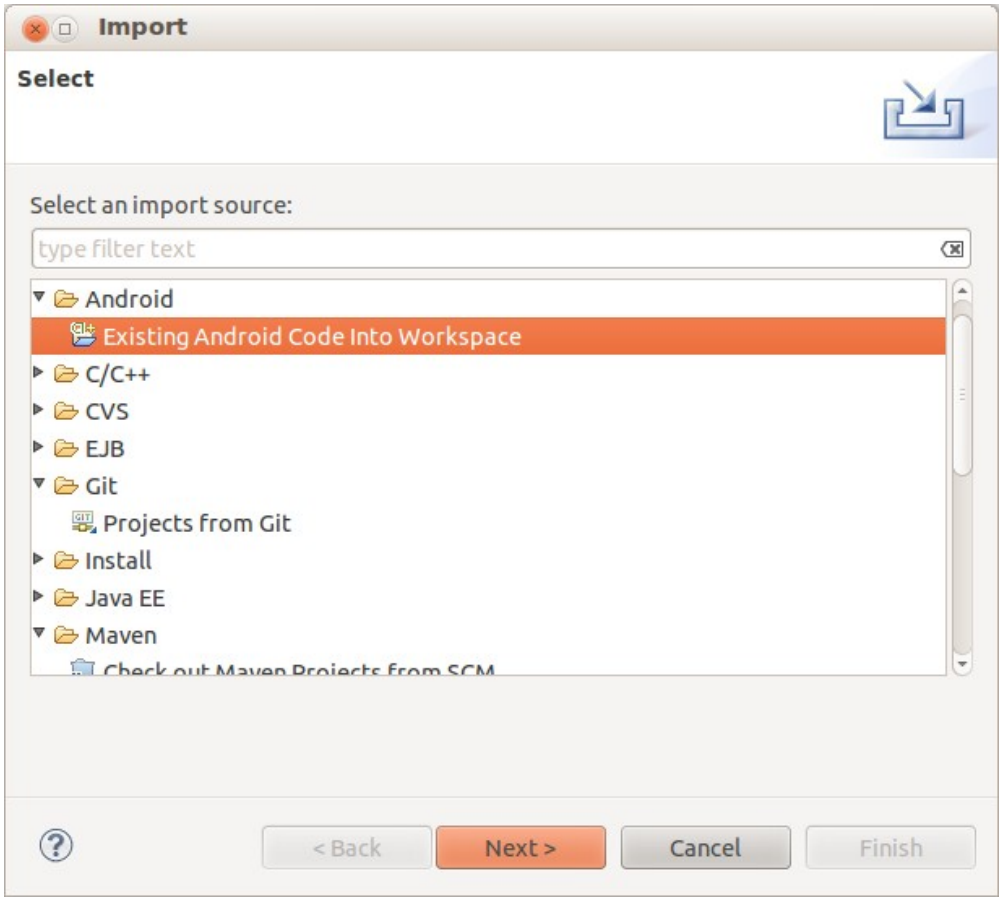
```
cd osgViewer/jni
~/Android/android-ndk-r8b/ndk-build
```

A makefile inclui duas instruções de debug que apresenta os PATH calculados. O output da execução desta makefile (via ndk-build) deverá ser algo do género:

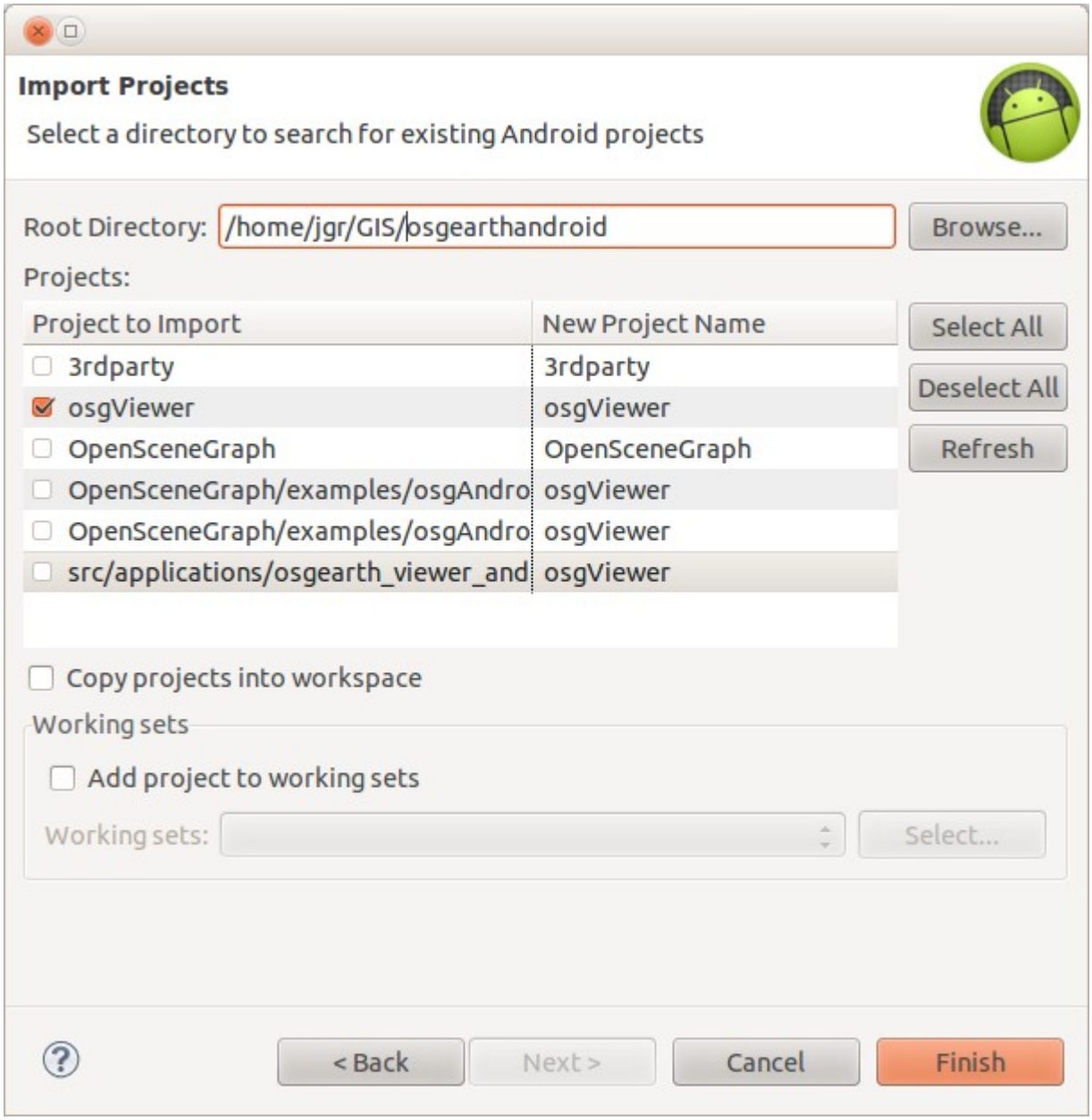
```
value of LOCAL_PATH is: /home/jgr/GIS/osgearthandroid/osgViewer/jni

value of OSGEARTH_ANDROID_DIR is: /home/jgr/GIS/osgearthandroid/osgViewer/jni/../..

Gdbserver      : [arm-linux-androideabi-4.6] libs/armeabi/gdbserver

Gdbsetup       : libs/armeabi/gdb.setup

Install        : libosgNativeLib.so => libs/armeabi/libosgNativeLib.so
```

Geração do osgViewer.apk

Depois de iniciar o Eclipse, em File → Import..., escolhe-se “Existing Android Code Into Workspace”, como ilustrado na imagem seguinte.



No diálogo seguinte, é necessário indicar a pasta com as sources (osgearthandroid) e selecionar apenas o projeto osgViewer, como ilustrado.



Depois de importado o projeto, o mesmo fica pronto a correr. O apk é gerado em osgViewer/bin/osgViewer.apk.

Conclusão

Os objetivos do projeto foram alcançados, tendo-se conseguido disponibilizar um visualizador de informação geográfica 3D em Android. Este visualizador usa a metáfora do globo virtual, já que os utilizadores de SIG estão invariavelmente familiarizados com o Google Earth.

O visualizador tem as funcionalidades típicas que se esperam de um globo, sendo que a interação é feita com os gestos habituais em Android.

O globo permite que dinamicamente se alterem todas as camadas de visualização, incluindo as camadas relacionadas com as infraestruturas 3D. Estes dados 3D são retornados por um serviço W3DS.

É, tanto quando sabemos, o primeiro globo open source a correr num dispositivo móvel.

Como era intenção deste projeto, criou-se uma aplicação genérica de consumo e visualização de dados 3D num dispositivo movel. Sobre esta aplicação, podem-se desenvolver soluções específicas para múltiplos casos de uso, que podem variar entre cenários apenas de visualização, até contextos em que interessa recolher informação.

O desenvolvimento para soluções móveis é recente, e ainda mais recente é a utilização do GPU para tirar o melhor proveito das capacidades gráficas do dispositivo móvel. Nesse contexto, interessa estar muito atento à evolução do suporte ao OpenGL em Android. Com a evolução do hardware dos dispositivos, é expectável que a API OpenGL ES evolua mais rapidamente. A aplicação está muito dependente desta camada para usufruir das capacidades gráficas nativas.