Otra guía paso a paso para instalación de OpenGL + ITK + VTK

Alejandro J. León Salas

Índice

Índice de figuras

1. Introducción

Este documento pretende ser una guía que facilite la instalación de OpenGL [Ope], ITK [ITK] y VTK [VTK] en una plataforma Linux, concretamente en Ubuntu. Antes de todo, la plataforma que he usado para la instalación es:

```
$> cat /proc/version
Linux version 4.13.0-32-generic (buildd@lgw01-amd64-004) (gcc version 5.4.0 20160609
 (Ubuntu 5.4.0-6ubuntu1~16.04.5)) #35~16.04.1-Ubuntu SMP Thu Jan 25 10:13:43 UTC 2018
```

```
$> sudo lshw
*-cpu
          description: CPU
          product: Intel(R) Core(TM) i7-4710MQ CPU @ 2.50GHz
          vendor: Intel Corp.
...
          capacity: 3500MHz
          width: 64 bits
...
*-memory
          description: System Memory
          physical id: 5
          slot: System board or motherboard
          size: 12GiB
...

$> sudo lshw -C display
  *-display
       description: VGA compatible controller
       product: GK107GLM [Quadro K1100M]
       vendor: NVIDIA Corporation
       physical id: 0
       bus info: pci@0000:01:00.0
       version: a1
       width: 64 bits
       clock: 33MHz
       capabilities: pm msi pciexpress vga_controller bus_master cap_list rom
       configuration: driver=nvidia latency=0
       resources: irq:32 memory:b0000000-b0ffffff memory:80000000-87ffffff
        memory:90000000-91ffffff ioport:4000(size=128) memory:b1000000-b107ffff
  *-display
       description: VGA compatible controller
       product: 4th Gen Core Processor Integrated Graphics Controller
       vendor: Intel Corporation
       physical id: 2
       bus info: pci@0000:00:02.0
       version: 06
       width: 64 bits
       clock: 33MHz
       capabilities: msi pm vga_controller bus_master cap_list rom
       configuration: driver=i915 latency=0
       resources: irq:27 memory:b1400000-b17fffff memory:a0000000-afffffff
        ioport:5000(size=64) memory:c0000-dffff
```

Listing 1: Cómo saber el HW que tengo instalado en mi sistema.

Voy a organizar este documento mediante las siguientes secciones principales:

- Instalación de OpenGL en Linux.
- Instalación de `cmake`, ya que se utiliza para configurar ITK y VTK, además de que la utilizaremos para la compilación y enlazado de nuestros programas.
- Instalación del *framework* VTK para Visualización (*Visualization*).
- Instalación del *framework* ITK para DIP (*Digital Image Processing*).
- Primeros programas de ejemplo para VTK, ITK y VTK+ITK.
- Ejemplos de código y *datasets* de VTK e ITK.

Pues nada más... ¡Empezamos!

2. Instalación de OpenGL en Linux

1. Instalación de OpenGL en Linux.

Lo primero que tienes que saber es la GPU que tienes en la máquina... ¡Por si has tirado la caja y no hay pegatina!

```
$> sudo lshw -C display
  *-display
       description: VGA compatible controller
       product: GK107GLM [Quadro K1100M]
       vendor: NVIDIA Corporation
       physical id: 0
       bus info: pci@0000:01:00.0
       version: a1
       width: 64 bits
       clock: 33MHz
       capabilities: pm msi pciexpress vga_controller bus_master cap_list rom
       configuration: driver=nvidia latency=0
       resources: irq:32 memory:b0000000-b0ffffff memory:80000000-8fffffff
        memory:90000000-91ffffff ioport:4000(size=128) memory:b1000000-b107ffff
  *-display
       description: VGA compatible controller
       product: 4th Gen Core Processor Integrated Graphics Controller
       vendor: Intel Corporation
       physical id: 2
       bus info: pci@0000:00:02.0
       version: 06
       width: 64 bits
       clock: 33MHz
       capabilities: msi pm vga_controller bus_master cap_list rom
       configuration: driver=i915 latency=0
       resources: irq:27 memory:b1400000-b17fffff memory:a0000000-afffffff
        ioport:5000(size=64) memory:c0000-dffff
```

Listing 2: Cómo saber la GPU que tengo instalada en mi sistema.

En mi caso tengo una Quadro K1100M de NVIDIA. En lugar de descargar el último driver proporcionado por el fabricante e instalarlo manualmente, voy a optar por lo fácil... ¡Usar los repositorios de mi distribución y rezar para que los drivers no estén demasiado desfasados!

En primer lugar hago click sobre el 'Dash' de Ubuntu (¡vamos! El botón que lanza el menú inicio de windows de toda la vida) y en el cuadro de texto que sirve para buscar programas introduzco: '`additional drivers`'. Ahora aparece la ventana de la aplicación 'Software & Updates' y la pestaña 'Additional Drivers'. Ahora basta con seleccionar la opción del driver propietario, en mi caso:

Using NVIDIA binary driver - version... (proprietary,tested)

en lugar de:

Using X.Org X server – Nouveau display driver...

y aplicar la nueva selección (si no tenías ya el driver de la GPU seleccionado), reiniciando la máquina para que los cambios del kernel se lleven a cabo. En la imagen 1 se muestra la pestaña Additional Drivers con el driver de X org seleccionado y en la imagen 2 se muestra el driver propietario de NVIDIA que ha reconocido automáticamente el programa.

Para comprobar que todo ha ido bien puedes cargar un programa que siempre suministran los fabricantes de GPUs para "toquetear" cosillas de gráficos. En mi caso uso de nuevo la búsqueda del Dash de Ubuntu: 'NVIDIA X Server Settings' y aquí comprueba que está trabajando con la versión del driver que me ha reconocido automáticamente Ubuntu. La imagen 3 muestra la información.

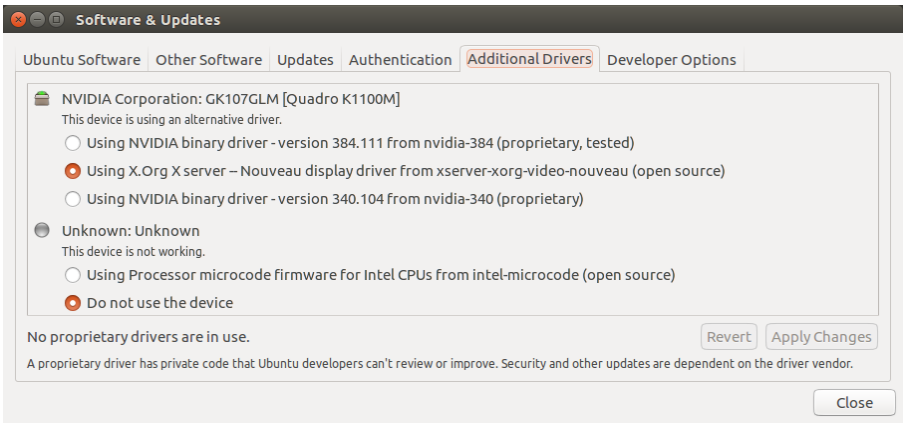


Figura 1: La pestaña Additional Drivers usando el driver de X org.

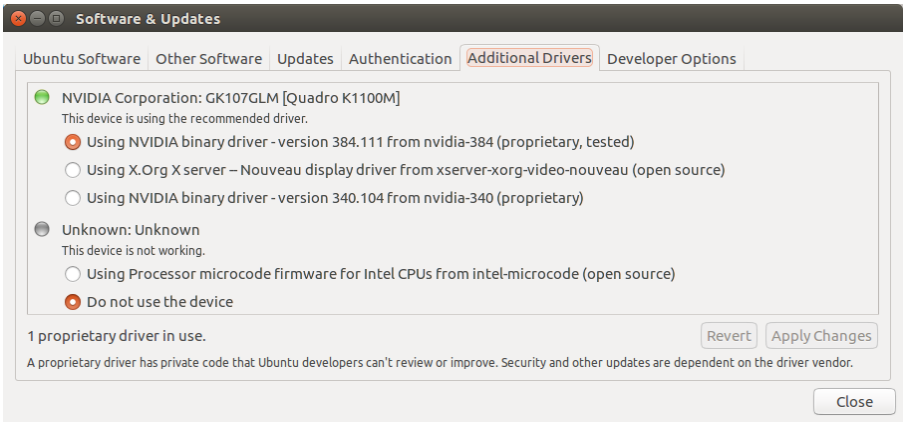


Figura 2: La pestaña Additional Drivers usando el driver de NVIDIA versión 384.111.

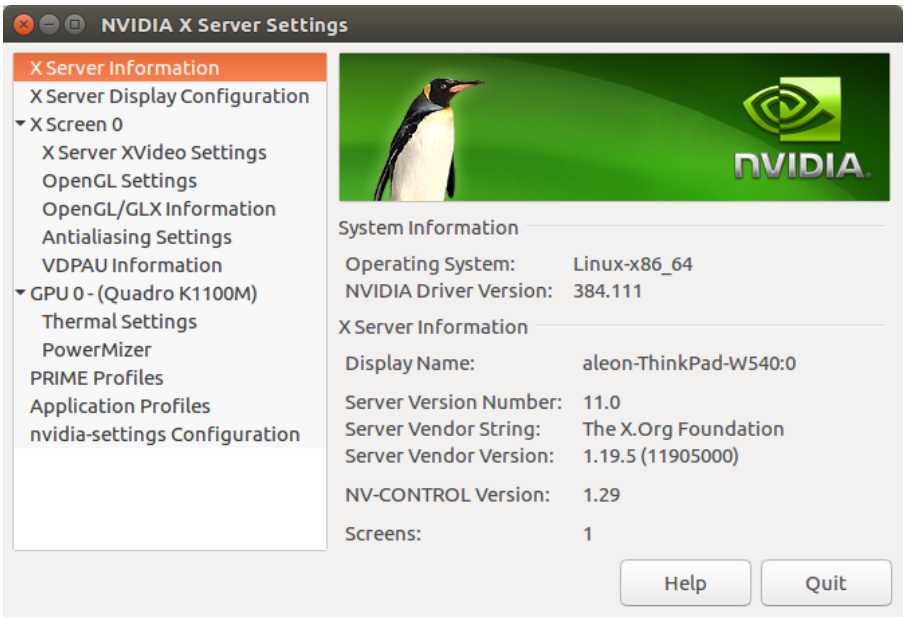


Figura 3: La aplicación NVIDIA X Server Settings con el driver correcto (384.111).

Ahora me gustaría saber que versión de OpenGL soporta mi driver. Para eso hago lo siguiente:

```
$> glxinfo | grep -i opengl
OpenGL vendor string: NVIDIA Corporation
OpenGL renderer string: Quadro K1100M/PCIe/SSE2
OpenGL core profile version string: 4.5.0 NVIDIA 384.111
OpenGL core profile shading language version string: 4.50 NVIDIA
OpenGL core profile context flags: (none)
OpenGL core profile profile mask: core profile
OpenGL core profile extensions:
OpenGL version string: 4.5.0 NVIDIA 384.111
OpenGL shading language version string: 4.50 NVIDIA
OpenGL context flags: (none)
OpenGL profile mask: (none)
OpenGL extensions:
OpenGL ES profile version string: OpenGL ES 3.2 NVIDIA 384.111
OpenGL ES profile shading language version string: OpenGL ES GLSL ES 3.20
OpenGL ES profile extensions:
```

Listing 3: Cómo saber la versión de OpenGL y "amigos" que soporta el driver que acabo de instalar.

En el listado3 puedo ver que soporta OpenGL 4.5.0, y GLSL 4.5.0, además de OpenGL ES 3.2. Una vez que tengo el driver bien instalado y sé la especificación de OpenGL y GLSL que implementa solamente me queda poder ejecutar un programilla simple en OpenGL (el `helloWorld` de OpenGL). Esto es conveniente para tener claro que todo está en su sitio y que que el software de más alto nivel, en nuestro caso ITK y VTK, pueda a su vez permitirnos construir programas gráficos. Para esto tengo que instalar algunos paquetes, para lo que puedes usar `dpkg` o `apt-get` o `synaptic`, ¡a gusto del consumidor!

```
$> sudo synaptic
Mark for installation:

[] libgl1-mesa-dev
[] libglu1-mesa-dev
[] freeglut3-dev
```

Listing 4: Paquetes a instalar para desarrollar programas con gráficos.

Nota: Uso `glut` como biblioteca de gestión de ventanas y eventos de teclado y ratón para el `helloWorld` de OpenGL por facilidad de uso y..., ¡por costrumbre!

Ahora podemos comprobar que tenemos los archivos de cabecera para incluir en nuestros programas como se muestra en el listado 5

```
$>ls /usr/include/GL
freeglut_ext.h  freeglut_std.h  glext.h  gl_mangle.h  glu_mangle.h  glxext.h
glxint.h      glxmd.h     glxtokens.h
freeglut.h      glcorearb.h     gl.h     glu.h        glut.h        glx.h
glx_mangle.h  glxproto.h  internal
```

Listing 5: Nuevo directorio, `/usr/include/GL`, y todos los archivos de cabecera y bibliotecas necesarios para desarrollar programas con gráficos.

Ya podemos compilar algunos programillas básicos que se encuentran en el directorio `/src/openGL`: `./src/openGL/firstOpenGL+glut.cpp`, `./src/openGL/openGL+glut01.cpp`, e incluso uno con algo de geometría y *rendering*, `./src/openGL/openGL+glut02.cpp`. Si todo ha ido bien tendrías que tener los tres ejecutables correspondientes tras compilarlos y enlazarlos. Deberías poder ver algo parecido a las capturas de pantalla que aparecen en la figura 4.

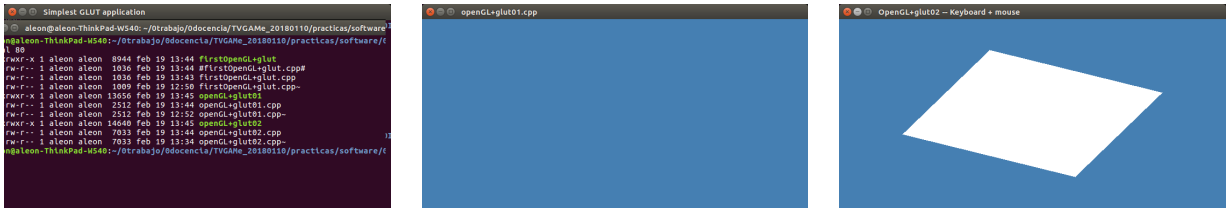


Figura 4: Capturas de pantalla con tres programas que utilizan OpenGL y Glut.

Para finalizar es conveniente instalar algunos paquetes adicionales que harán que OpenGL y sus “amigos” vayan finos en tu plataforma:

```
$> sudo synaptic
Mark for installation:

[] glew-utils
[] libglew-dev
[] driconf
[] libdrm-common
[] libdrm-dev
[] libdrm2
```

Listing 6: Paquetes adicionales a instalar para mejorar la experiencia de uso de OpenGL en tu plataforma.

3. Instalacion de `cmake`

3.1. Descargar `cmake`

Accede a la dirección https://cmake.org/ y pulsa en `Download` (https://cmake.org/download/), a continuación descarga el archivo "`cmake-3.10.2.tar.gz`".

3.2. Instalar `cmake` en `/usr/local`

A) Copia el archivo descargado a `/usr/local`.

```
$> sudo cp ~/Downloads/cmake-3.10.2.tar.gz /usr/local
```

B) Desempaqueta y descomprime (`uncompress and untar`) el archivo `cmake-3.10.2.tar.gz`.

```
/usr/local$> sudo tar xzvf cmake-3.10.2.tar.gz
```

C) Y finalmente ya puedes instalar `cmake`.

```
/usr/local$> cd cmake-3.10.2/
/usr/local/cmake-3.10.2$> sudo ./configure
/usr/local/cmake-3.10.2$> sudo make && sudo make install
```

El paso `make install` es opcional ya que podríamos ejecutar `cmake` y `ccmake` desde el directorio en donde se ha generado el ejecutable: `/usr/local/cmake-3.10.2/bin/`; pero es mejor que los ejecutables se encuentren en `/usr/local/bin` para “hacer las cosas bien hechas”. Ahora deberíamos tener instalados en nuestro sistema `cmake` y `ccmake` lo cual podemos comprobar con la orden:

```
$> ls /usr/local/bin/c*
cmake  cpack  ctest
```

¡Vaya, vaya!...pues `ccmake` no aparece en el listado. Claro, el problema es que la interfaz `ccmake` para `cmake` requiere el paquete de `ncurses-dev` que contine las bibliotecas estáticas y archivos de cabecera para *ncurses*. ncurses es una biblioteca para trabajar con texto en terminales que permite al programador abstraerse del tipo de terminal. La solución pasa por instalar los paquetes `libncurses5-dev` y `libncursesw5-dev` directamente desde los repositorios de la distribución; y repetir el proceso de instalación de `cmake`, es decir re-configurar/compilar/instalar `cmake`.

Ahora si que aparece `ccmake` al ejecutar la orden:

```
$> ls /usr/local/bin/c*
ccmake  cmake  cpack  ctest
```

4. Instalación del `framework` VTK

4.1. Descargar VTK

Accede a la dirección https://www.vtk.org/download/. La última versión estable es la 8.1.0. Descarga los siguientes archivos: Latest Release (8.1.0) `VTK-*`, `VTKData-*`, `VTKLargeData-*` and `vtkDocHtml-*`

4.2. Instalar VTK en `/usr/local`

A) Crea el directorio /usr/local/VTK y copia en él los archivos descargados.

```
/usr/local$> sudo mkdir VTK && cd /usr/local/VTK
/usr/local/VTK$> sudo cp ~/Downloads/VTK* ~/Downloads/vtkDocHtml-8.1.0.tar.gz .
```

B) Desempaqueta y descomprime (*uncompress and untar*) los archivos `VTK-*`, `VTKData-*` y `VTKLargeData-*`

```
/usr/local/VTK$> sudo tar xzvf VTK-8.1.0.tar.gz
/usr/local/VTK$> sudo tar xzvf VTKData-8.1.0.tar.gz
/usr/local/VTK$> sudo tar xzvf VTKLargeData-8.1.0.tar.gz
```

C) Instala VTK usando `ccmake`.

```
/usr/local/VTK$> sudo mkdir VTK-build
/usr/local/VTK$> cd VTK-build/
/usr/local/VTK/VTK-build$> sudo ccmake /usr/local/VTK/VTK-8.1.0
```

Es muy importante tener en cuenta que hay que configurar VTK (`ccmake`) antes de configurar ITK. Una vez configurados los dos `frameworks` se puede proceder a construir (compilar y enlazar) ITK y después VTK.

El "GUI"(basado en ncurses) que proporciona CCMake para CMake se comporta como un proceso de configuración iterativo que termina cuando nos aparece la opción para generar. La idea es seleccionar valores de variables, los cuales pueden ser ON/OFF o cadenas o archivos, y una vez seleccionados pasamos a configurar (pulsar la tecla 'c'). Esto se repite hasta que aparece la opción de generar, momento en el que podemos pulsar la tecla 'g'.

El listado 7 muestra la configuración de VTK que aconsejo usar la primera vez que se utiliza `ccmake`.

```
Variable Valor
BUILD_SHARED_LIBS ON
BUILD_TESTING ON
CMAKE_BUILD_TYPE Debug
VTK_Group_Imaging ON
```

Listing 7: Configuración inicial de variables de `ccmake` para VTK.

Configuramos hasta poder generar y generamos. Durante el proceso de generación aparece el siguiente error en mi plataforma:

```
CMake Error at Rendering/OpenGL2/CMakeLists.txt:177 (message):
  X11_Xt_LIB could not be found.  Required for VTK X lib.
```

Este error se debe a la opción: VTK_RENDERING_BACKEND `OpenGL2`. Para resolver el problema simplemente tenemos que instalar el paquete `libxt-dev`. Puede que esto no os ocurra en vuestro caso pero viene bien para saber como solventar otras posibles dependencias que requiera VTK en vuestra plataforma.

```
$> sudo apt-get install libxt-dev
```

Salimos de `ccmake`, entramos de nuevo y a configurar. Como siempre continuamos hasta que aparezca la opción de generar. Generamos y salimos de `ccmake` (o puede que él salga automáticamente). Ahora tenemos a VTK preparado para ser construido con `make` en el directorio `/usr/local/VTK/VTK-build`.

Antes de construir VTK vamos a configurar y generar ITK para que aparezca la opción `Module_ITKVtkGlue`, la cual nos permite utilizar las capacidades de visualización de VTK para poder visualizar imágenes de ITK en nuestros propios programas, sin tener que hacer uso de aplicaciones de visualización de imágenes externas. **Por consiguiente, dentro de la sección de ITK contruiremos VTK**.

5. Instalación del `framework` ITK

5.1. Descargar ITK

Accede a la dirección https://itk.org/ITK/resources/software.html. La última versión estable es la 4.13.0. Descarga los siguientes archivos:

```
Current Release
The current release is version 4.13.0 from December 2017. The following downloads
 are available

Library Source
InsightToolkit-4.13.0.tar.gz

Testing Data
Unpack optional testing data in the same directory where the Library Source is unpacked.
InsightData-4.13.0.tar.gz

Documentation
InsightSoftwareGuide-Book1-4.13.0.pdf
InsightSoftwareGuide-Book2-4.13.0.pdf
InsightSphinxExamplesPdf-4.13.0.pdf
InsightSphinxExamples-4.13.0.tar.gz
InsightWikiExamples-4.13.0.tar.gz
```

5.2. Instalar ITK en `/usr/local`

A) Crea el directorio /usr/local/ITK y copia en él los archivos descargados.

```
/usr/local$> sudo mkdir ITK
/usr/local$> sudo cp /home/aleon/Downloads/Insight* /usr/local/ITK
```

B) Desempaqueta y descomprime (*uncompress and untar*) los archivos `Insight*tar.gz`

```
/usr/local/ITK$> sudo tar xvzf InsightToolkit-4.13.0.tar.gz
/usr/local/ITK$> sudo tar xvzf InsightData-4.13.0.tar.gz
/usr/local/ITK$> sudo tar xvzf InsightSphinxExamples-4.13.0.tar.gz
/usr/local/ITK$> sudo tar xvzf InsightWikiExamples-4.13.0.tar.gz
```

C) Instala ITK usando `ccmake`.

```
/usr/local/ITK$> sudo mkdir /usr/local/ITK/InsightToolkit-4.13.0/ITK-build
/usr/local/ITK$> cd /usr/local/ITK/InsightToolkit-4.13.0/ITK-build
/usr/local/ITK$> sudo ccmake /usr/local/ITK/InsightToolkit-4.13.0
```

En este caso tenemos que pulsar la tecla 't' para cambiar a modo avanzado de `ccmake` y buscar la variable `Module_ITKVtkGlue` para activarla. Queda así:

```
Module_ITKVtkGlue  ON  # Request building ITKVtkGlue for visualizing
                       # using a VTK visualizer (viewer).
```

A continuación pulsamos la 'c' para configurar hasta que aparezca la opción de generar y generamos.

D) Construcción de VTK y después de ITK. Los procesos de generación de VTK e ITK nos han dotado de todo lo necesario para poder construir (compilar y enlazar) ITK y VTK por lo que procedemos de la siguiente forma:

```
$> cd /usr/local/VTK/VTK-build
$> sudo make -j4 && sudo make install
```

Una vez finalizado el proceso (tarda un ratillo), podemos comprobar que todo está bien:

```
$> ls /usr/local/libvtk*
```

Finalmente construimos ITK:

```
$> cd /usr/local/ITK/InsightToolkit-4.13.0/ITK-build
$> sudo make -j4 && sudo make install
```

Una vez finalizado el proceso (tarda otro ratillo), podemos comprobar que todo está bien:

```
$> ls /usr/local/lib/libitk* /usr/local/lib/libITK*
```

6. Primeros programas de ejemplo para VTK, ITK y VTK+ITK

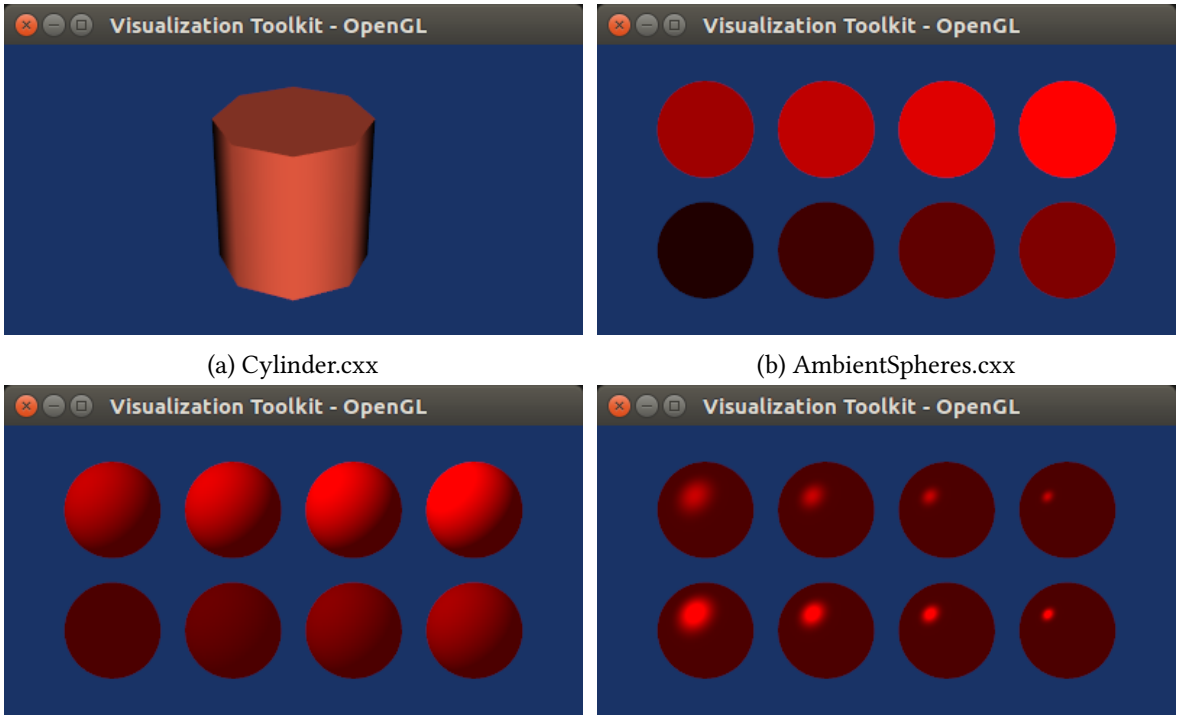
Ahora vamos a probar que todo va bien utilizando algunos ejemplos ya programados que nos permitiran comprobar que todo funciona en nuestro sistema, además de familiarizarnos con el proceso de configuración y construcción de programas mediante `cmake` y `ccmake` que será el que utilizaremos durante la asignatura. Todos los ejemplos están en el directorio `src`.

En primer lugar configuraremos un programa en VTK que viene en los ejemplos ubicados en el directorio de la instalación: `/usr/local/VTK/VTK-8.1.0/Examples/Rendering/Cxx` y que podéis encontrar también en el directorio `/src/vtk/rendering`.

En el directorio Rendering hay cuatro programas de ejemplo: `AmbientSpheres.cxx`, `Cylinder.cxx`, `DiffuseSpheres.cxx` y `SpecularSpheres.cxx`; junto con un archivo de texto plano `CMakeLists.txt`. Este archivo es el que instruye a `cmake` sobre qué cosas tiene que hacer para configurar lo necesario para, posteriormente, poder construir nuestro programa. Lo primero que hay que hacer es crearse un directorio en nuestro *directorio home* donde configuraremos (`ccmake` o `cmake`) y construiremos (`make`) nuestro proyecto. El listado 8 muestra los pasos a seguir.

```
$> mkdir ~/TVG/VTK/rendering && cd ~/TVG/VTK/rendering
$> cp /usr/local/VTK/VTK-8.1.0/Examples/Rendering/Cxx .
$> mkdir ./build && cd ./build
$> cmake ..
$> make
```

Listing 8: Procedimiento para configurar (`ccmake` o `cmake`) y construir (`make`) un proyecto con `cmake` para VTK.



(a) Cylinder.cxx (b) AmbientSpheres.cxx

(c) DiffuseSpheres.cxx (d) SpecularSpheres.cxx

Figura 5: Capturas de pantalla de los cuatro programas de ejemplo de `vtk`.

Si todo ha ido bien tendrías que tener en el directorio `build` los correspondientes programas y deberían funcionar mostrando algo parecido a las capturas de pantalla que aparecen en la figura 5.

Ahora vamos a probar un proyecto en ITK. Para ello descomprimimos el archivo `WidthHeight.tar` y repetimos el proceso que previamente hemos utilizado con VTK adaptando algunas cosillas para ITK. El listado 9 muestra los distintos pasos.

```
$> mkdir ~/TVG/ITK+VTK/ && cp WidthHeight.tar ~/TVG/ITK+VTK/
$> cd ~/TVG/ITK+VTK/
$> tar xzvf WidthHeight.tar
$> mkdir ./WidthHeight/build && cd ./WidthHeight/build
$> cmake ..
$> make
$> ./WidthHeight /usr/local/ITK/InsightToolkit-4.13.0/Examples/
Data/BrainProtonDensitySlice256x256.png
```

Listing 9: Procedimiento para configurar (`ccmake` o `cmake`) y construir (`make`) un proyecto con `cmake` para ITK.

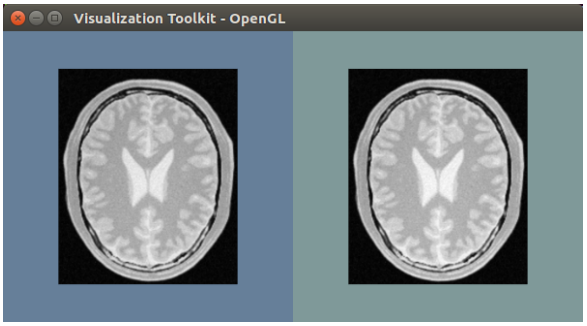
Ahora vamos a probar un programa que usa el modulo `itkVtkGlue`, el cual permite utilizar la clase `QuickView` para que nuestros programas puedan visualizar imágenes directamente sin tener que utilizar un programa externo para ello.

Descomprimimos el archivo `QuickViewDemo.tar` y repetimos el proceso de construcción de programas que ya hemos utilizado para `WidthHeight.tar`, probando al final el programa de la siguiente forma:

```
$> ./QuickViewDemo /usr/local/ITK/InsightToolkit-4.13.0/Examples/
Data/BrainProtonDensitySlice256x256.png
```

NOTA: Lo más importante de `QuickViewDemo` es que incorpora el archivo `CMakeLists.txt` cuya estructura utilizaremos para los programas que utilicen ITK y VTK, por lo que lo podrás usar como plantilla.

Finalmente vamos a configurar y construir un programa que utiliza ITK y VTK: `miTest`.



(a) QuickViewDemo.cxx

(b) miTest.cpp

Figura 6: Capturas de pantalla de dos programas de ejemplo de `itk`.

```
./miTest /usr/local/ITK/InsightToolkit-4.13.0/Examples/Data/BrainT1Slice.png
```

La figura6 muestra dos capturas de pantalla con la salida proporcionada por los programas `QuickViewDemo` y `miTest`.

7. Ejemplos de código y *datasets* de VTK e ITK

7.1. Ejemplos de VTK

Una forma de aprender VTK es leer el código de los ejemplos y explorar en la documentación las clases y métodos utilizados. Hay dos fuentes principales de ejemplos:

- Los ejemplos que vienen con la instalación de VTK (/usr/local/VTK/VTK-8.1.0/Examples/)
- El sitio oficial para los ejemplos de VTK (https://lorensen.github.io/VTKExamples/site/)

7.2. Ejemplos de ITK

Aunque ITK proporciona una buena guia de software con el *framework*, es bueno mirar ejemplos para comprender el modelo de programación genérica de ITK y aprender como se utilizan las clases y métodos que proporciona. Hay tres fuentes principales de ejemplos:

- Los ejemplos que vienen con la instalación de ITK (/usr/local/ITK/InsightToolkit-4.13.0/Examples)
- Los ejemplos incluidos en la página de la Wiki de ITK (https://itk.org/Wiki/ITK/Examples)
- Los ejemplos del Insight Toolkit (https://itk.org/ITKExamples/)

References

[ITK] ITK ; ITK (Hrsg.): *Insight Segmentation and Registration Toolkit (ITK)*. https://www.itk.org/, Abruf: 15.02.2018

[Ope] OpenGL(R) ; OpenGL(R) (Hrsg.): *OpenGL(R) The Industry's Foundation for High Performance Graphics*. https://www.opengl.org/, Abruf: 15.02.2018

[VTK] VTK ; VTK (Hrsg.): *Visualization Toolkit (VTK)*. https://www.vtk.org/, Abruf: 15.02.2018