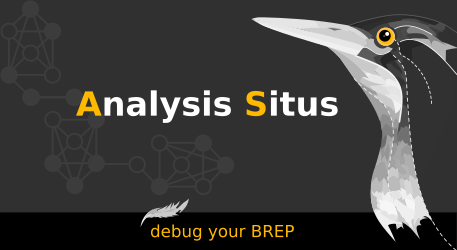
Analysis Situs compilation on Linux

## Note

This guide describes the building process as it is now. The procedure itself requires some polishing and even reconsidering a few stages to become more convenient. Anyway, this document serves as a reference guide, so if you encounter some problem or have ideas how to improve it, feel free to post a comment here.



This instruction assumes that you have a “clean” installation of Linux. Therefore, it contains the steps to install 3-rd party dependencies that your system can already have installed. If so, feel free to skip the corresponding steps.

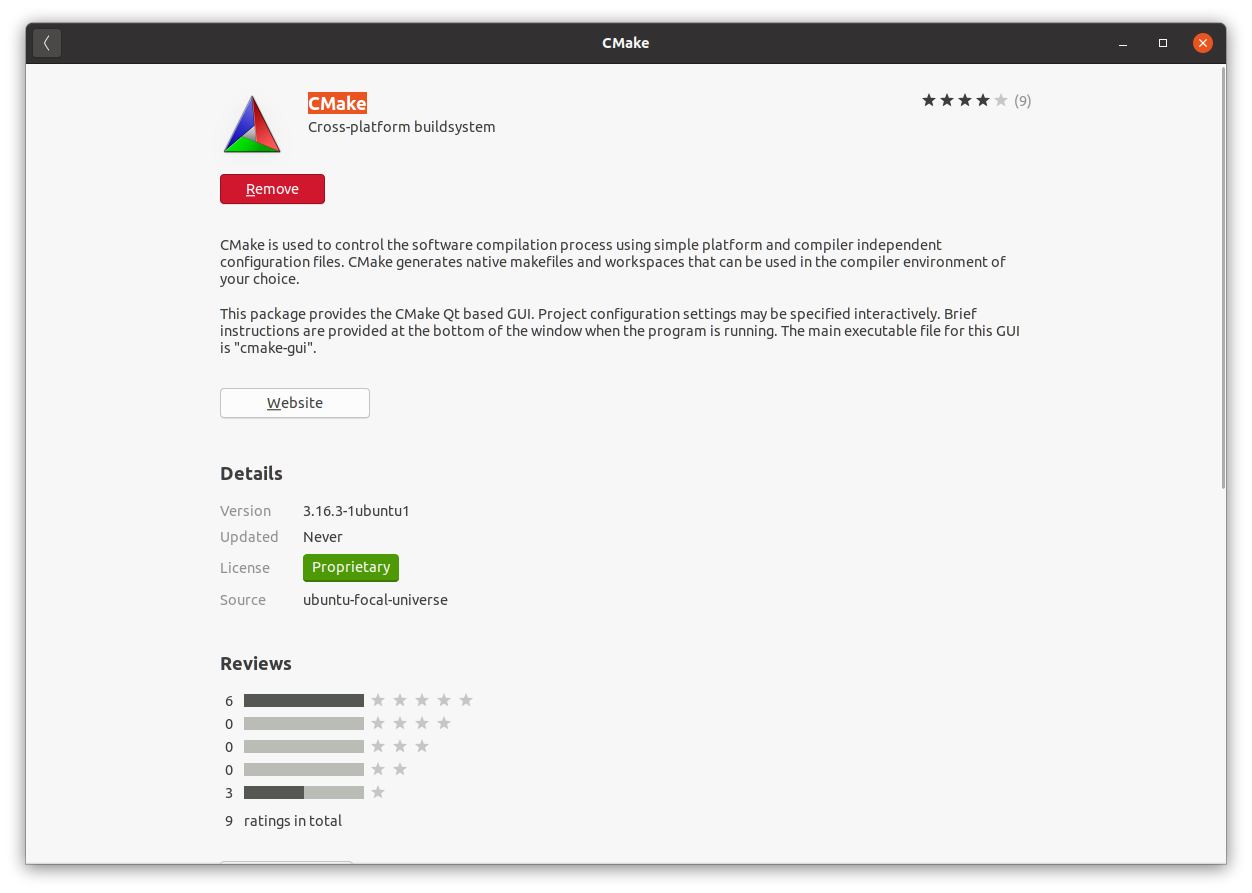
## Step-by-step instruction

The following guide describes the building process for Analysis Situs on Linux OS. In what follows, Linux Ubuntu 18.04.3 LTS and 20.04.1 LTS were used.

1. Install cmake-gui.

**sudo apt install cmake-qt-gui**

Alternatively, CMake can be installed from the Ubuntu Software application.



1. Install Qt asset.

**sudo apt install build-essential**

sudo apt install qtcreator

sudo apt install qt5-default

**sudo apt-get install libqt5x11extras5-dev**

**sudo apt-get install qttools5-dev**

1. Go to your working directory where you’d like to have Analysis Situs sources.

**cd /home/quaoar**

**mkdir work**

**cd work**

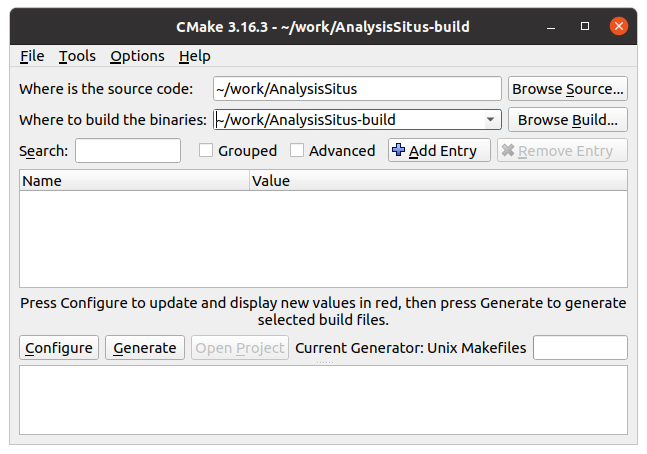
1. Install git.

**sudo apt install git**

1. Clone the repo.

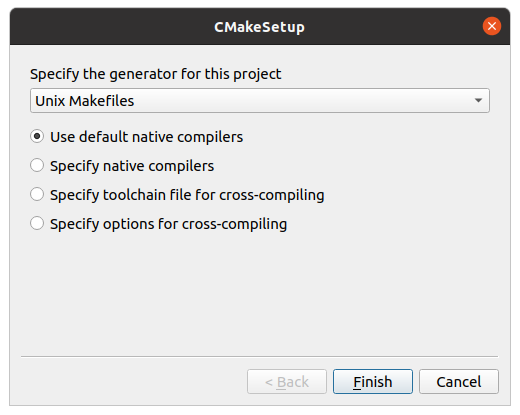
**git clone https://gitlab.com/ssv/AnalysisSitus.git**

1. Run cmake-gui and specify the source directory for Analysis Situs. You also have to specify the build directory.



Source: /home/quaoar/work/AnalysisSitus

Build: /home/quaoar/work/AnalysisSitus-build



1. Try to configure. The process will end up with errors, saying that 3-rd parties are not found. You can specify the following values for the cmake variables at this stage:

**3RDPARTY\_DIR** = /usr/lib

**INSTALL\_DIR** = /home/quaoar/work/AnalysisSitus-install  
**CMAKE\_BUILD\_TYPE** = Release

At this stage, the configuration process cannot be completed as the essential 3-rd parties are missing.

1. Get VTK by cloning its repo and checkout the corresponding version.

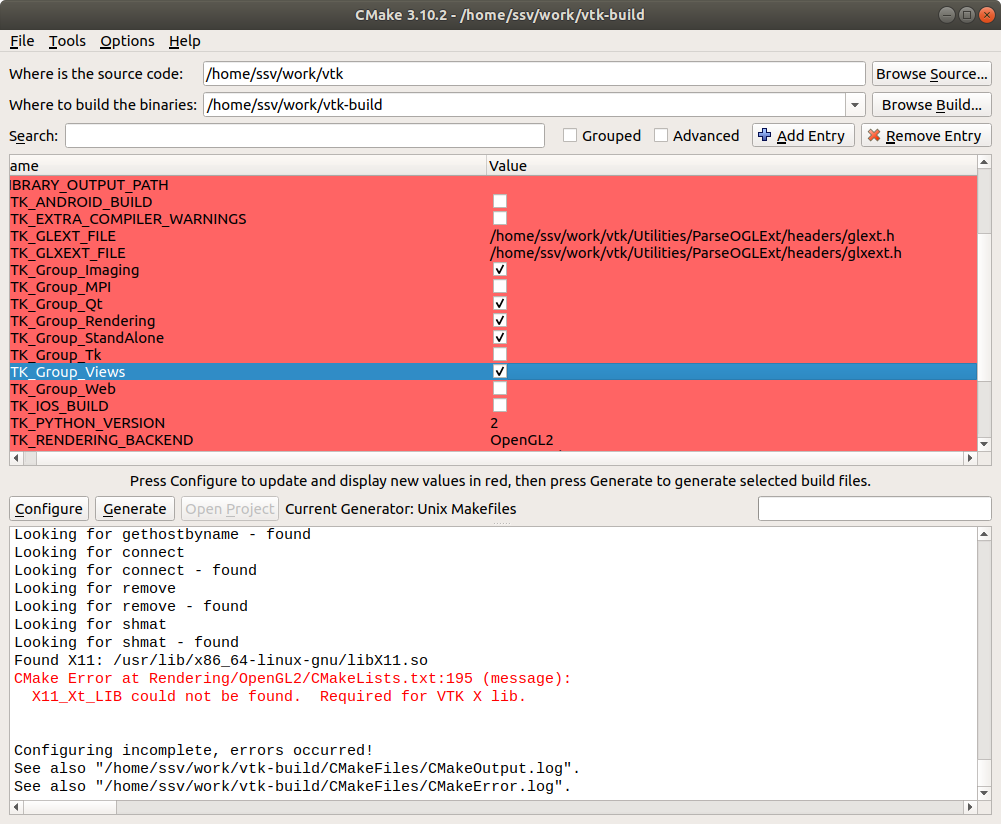
**git clone** [**https://gitlab.kitware.com/vtk/vtk.git**](https://gitlab.kitware.com/vtk/vtk.git)

**git checkout v8.2.0**

1. Install libxt-dev (X Toolkit).

**sudo apt-get install libxt-dev**

1. Run cmake-gui and configure it to build VTK.



Pay attention to setting up some specific VTK props:

**VTK\_Group\_Imaging** = YES

**VTK\_Group\_Qt** = YES

**VTK\_Group\_Views** = YES

1. Set CMAKE\_INSTALL\_PREFIX to the directory where VTK will be installed.

**CMAKE\_INSTALL\_PREFIX** = /home/quaoar/work/vtk-install

1. Configure and Generate in cmake. Then make and install while in the VTK build directory.

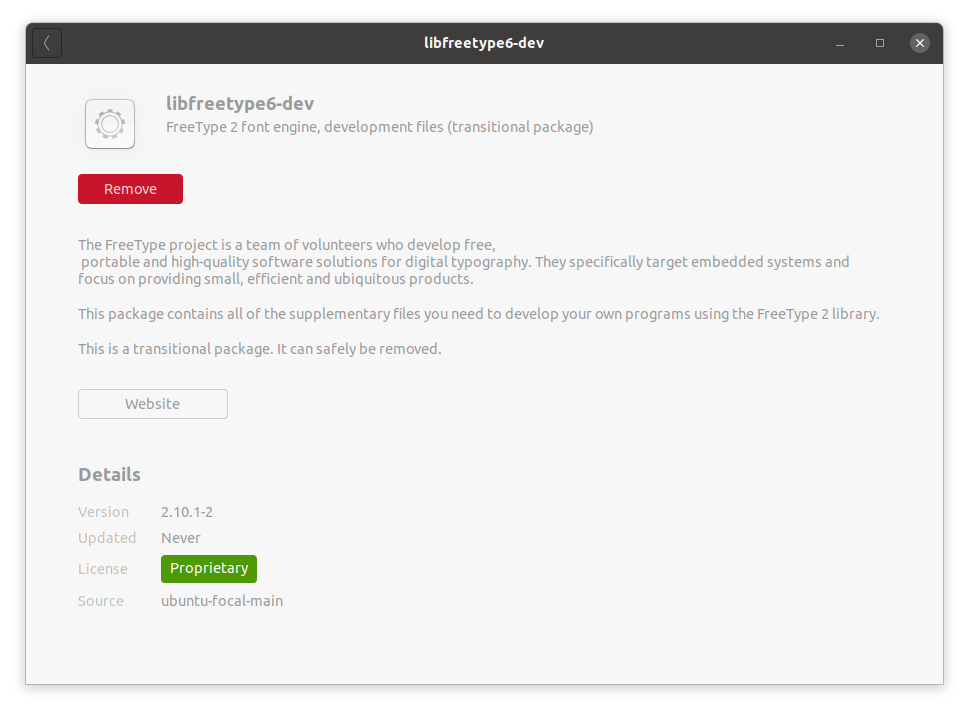
**make**

**make install**

1. Install the development package of freetype.

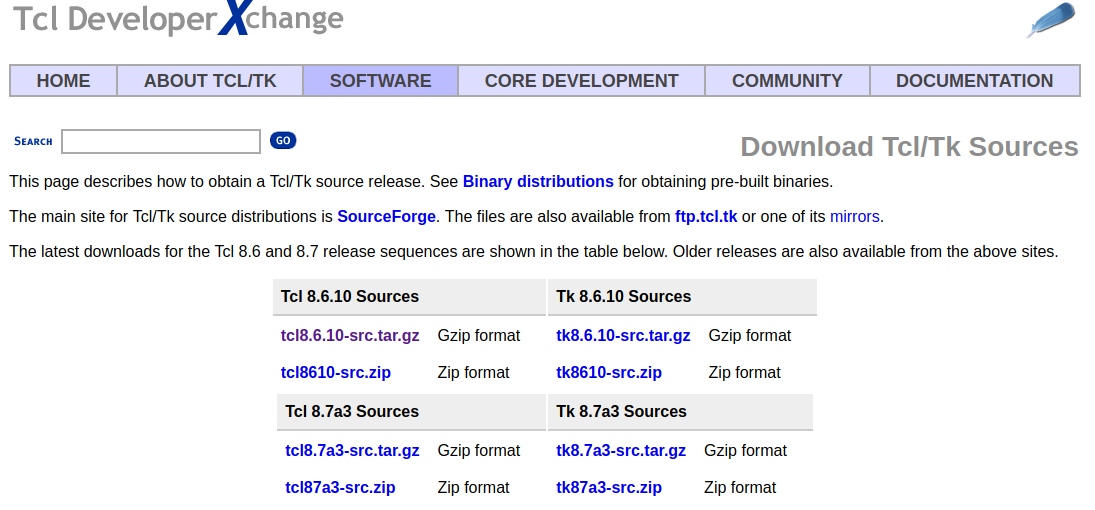
Ubuntu 18\_04: <https://packages.ubuntu.com/bionic/libfreetype6-dev>  
Ubuntu 20\_04: <https://www.ubuntuupdates.org/package/core/focal/main/base/libfreetype6-dev>

Or use Ubuntu Software to install this package.



1. Download tcl 8.6 from the official website.

<https://www.tcl.tk/software/tcltk/download.html>



1. Build tcl.

**cd <TCL\_SRC\_DIR>/unix**

**./configure --enable-gcc --enable-shared --enable-threads --prefix=/home/quaoar/work/tcl-8.6-install --enable-64bit**

**make**

**make install**

1. Do the same with tk 8.6 (not necessary for Analysis Situs, do this only if you want to have Tk to run DRAW application of OpenCascade kernel).

**cd <TK\_SRC\_DIR>/unix**

**./configure --enable-gcc --enable-shared --enable-threads --with-tcl=/home/quaoar/work/tcl-8.6-install/lib --prefix=/home/quaoar/work/tk-8.6-install**

**make**

**make install**

1. Get the sources of OpenCascade library.

**git clone** [**https://git.dev.opencascade.org/repos/occt.git**](https://git.dev.opencascade.org/repos/occt.git)

Make sure the appropriate version of OpenCascade is used. Generally, the master branch should be just fine. If not, stick to the release ver7.4.

**OPTIONAL:** git checkout V7\_4\_0p1

1. Run cmake-gui and set 3RDPARTY\_DIR to the directory where your tcl and tk libs are installed.

**NOTE:** if some errors regarding Tcl/Tk appear on configuration, switch to the advanced mode of cmake and specify the corresponding Tcl/Tk libs manually.

1. Make sure to specify the INSTALL\_DIR for OpenCascade.

**INSTALL\_DIR** = ~/work/occt-install

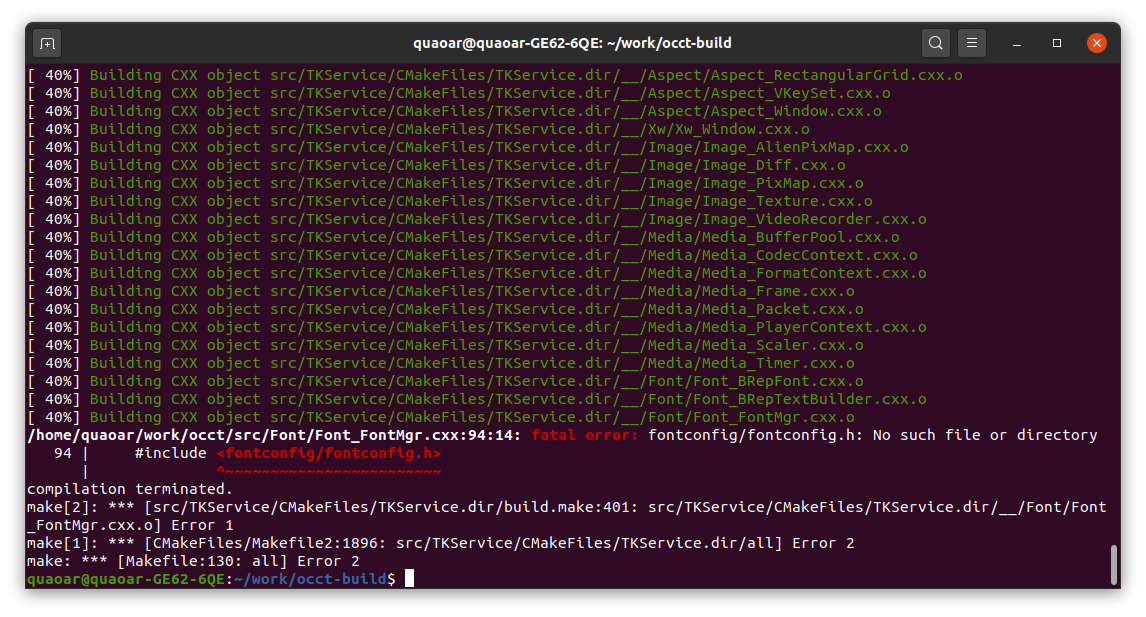
1. Run make and make install in the build directory of OpenCascade.

**cd /home/quaoar/work/occt-build**

**make**

**make install**

**NOTE:** When building OpenCascade, you might discover compilation errors, which might be resolved by addings extra packages to your Linux and remaking the project.



I had to add the following packages:

**sudo apt-get install libfontconfig1-dev**

**sudo apt-get install libxmu-dev libxi-dev**

1. Get the sources of Active Data.

**git clone https://gitlab.com/ssv/active-data.git**

1. Run cmake-gui and specify the directory with the 3-rd parties and the INSTALL\_DIR. Make sure that the ActiveData\_USE\_TBB flag remains unchecked. Then make and make install.

**NOTE:** you may want to keep TBB enabled. TBB allows for detaching of the task execution thread from the main thread (i.e., all Tree Functions could be executed in a detached thread). Download TBB from <https://github.com/oneapi-src/oneTBB/releases> (tbb-2020.2-lin.tgz). Then enable ActiveData\_USE\_TBB flag in cmake and make sure to specify 3RDPARTY\_tbb\_DIR, 3RDPARTY\_tbb\_INCLUDE\_DIR, 3RDPARTY\_tbb\_LIBRARY\_DIR cmake variables.

**3RDPARTY\_tbb\_DIR** = /home/quaoar/work/tbb-2020.2-lin/tbb

**3RDPARTY\_tbb\_INCLUDE\_DIR** = /home/quaoar/work/tbb-2020.2-lin/tbb/include

**3RDPARTY\_tbb\_LIBRARY\_DIR** = /home/quaoar/work/tbb-2020.2-lin/tbb/lib/intel64/gcc4.8

1. Get the sources of Eigen. There is no need to configure or build anything for Eigen as this library is implemented in C++ includes entirely.

<http://eigen.tuxfamily.org/>

1. Run cmake-gui for Analysis Situs. The list of primary dependencies appears at the first configuration attempt.

CMake Error at CMakeLists.txt:124 (message):

NOT FOUND: 3RDPARTY\_active\_data\_INCLUDE\_DIR

3RDPARTY\_active\_data\_LIBRARY\_DIR 3RDPARTY\_vtk\_LIBRARY\_DIR

3RDPARTY\_OCCT\_INCLUDE\_DIR 3RDPARTY\_OCCT\_LIBRARY\_DIR

3RDPARTY\_tcl\_LIBRARY\_DIR 3RDPARTY\_tk\_LIBRARY\_DIR

3RDPARTY\_freetype\_INCLUDE\_DIR 3RDPARTY\_freetype\_LIBRARY\_DIR

1. Make sure to disable USE\_MOBIUS, USE\_INSTANT\_MESHES, USE\_THREADING cmake flags.
2. From the build directory of Analysis Situs, run

**make  
make install**

## Configuring in CLI

git clone <https://gitlab.com/ssv/AnalysisSitus.git> asitus-src

cd asitus-src

mkdir build

cd ./build

cmake -D3RDPARTY\_DIR=/usr/lib/ ../

cmake -D3RDPARTY\_active\_data\_DIR=/home/ssv/work/active-data-install/ ../

cmake -D3RDPARTY\_active\_data\_LIBRARY\_DIR=/home/ssv/work/active-data-install/lib/ ../

cmake -D3RDPARTY\_EIGEN\_DIR=/home/ssv/work/eigen-3.3.7/ ../

cmake -D3RDPARTY\_vtk\_DIR=/home/ssv/work/vtk-install/ ../

cmake -D3RDPARTY\_vtk\_LIBRARY\_DIR=/home/ssv/work/vtk-install/lib/ ../

cmake -D3RDPARTY\_OCCT\_DIR=/home/ssv/work/occt-install/ ../

cmake -D3RDPARTY\_OCCT\_INCLUDE\_DIR=/home/ssv/work/occt-install/include/opencascade/ ../

cmake -D3RDPARTY\_OCCT\_LIBRARY\_DIR=/home/ssv/work/occt-install/lib/ ../

cmake -D3RDPARTY\_tcl\_DIR=/home/ssv/work/tcl-8.6-install/ ../

cmake -D3RDPARTY\_tcl\_LIBRARY\_DIR=/home/ssv/work/tcl-8.6-install/lib/ ../

cmake -DCMAKE\_BUILD\_TYPE=Release ../

cmake -DINSTALL\_DIR=/home/ssv/work/asitus-install/ ../

make

make install

# Headless rendering

**NOTE:** It is advised to use dedicated build/install directories for the offscreen versions of VTK and Analysis Situs. E.g.:

/home/quaoar/work/AnalysisSitus-build  
/home/quaoar/work/**AnalysisSitus-offscreen-build**  
/home/quaoar/work/vtk-build  
/home/quaoar/work/**vtk-offscreen-build**

Install OSMESA (will be required to build VTK): sudo apt-get install libosmesa6-dev

* Enable **VTK\_OPENGL\_HAS\_OSMESA** cmake flag when configuring VTK.
* Enable **VTK\_DEFAULT\_RENDER\_WINDOW\_HEADLESS**.

The following example demonstrating the offscreen rendering should work with VTK8.2.

#include <vtkAutoInit.h>

#include <vtkPolyDataMapper.h>

#include <vtkActor.h>

#include <vtkRenderWindow.h>

#include <vtkRenderer.h>

#include <vtkPolyData.h>

#include <vtkSmartPointer.h>

#include <vtkSphereSource.h>

#include <vtkWindowToImageFilter.h>

#include <vtkPNGWriter.h>

VTK\_MODULE\_INIT(vtkRenderingOpenGL2); // VTK was built with vtkRenderingOpenGL2

VTK\_MODULE\_INIT(vtkInteractionStyle);

int main(int, char \*[])

{

std::cout << "Hello, offscreen rendering!" << std::endl;

// Create a sphere

vtkSmartPointer<vtkSphereSource> sphereSource =

vtkSmartPointer<vtkSphereSource>::New();

// Create a mapper and actor

vtkSmartPointer<vtkPolyDataMapper> mapper =

vtkSmartPointer<vtkPolyDataMapper>::New();

mapper->SetInputConnection(sphereSource->GetOutputPort());

vtkSmartPointer<vtkActor> actor =

vtkSmartPointer<vtkActor>::New();

actor->SetMapper(mapper);

// A renderer and render window

vtkSmartPointer<vtkRenderer> renderer =

vtkSmartPointer<vtkRenderer>::New();

vtkSmartPointer<vtkRenderWindow> renderWindow =

vtkSmartPointer<vtkRenderWindow>::New();

renderWindow->SetOffScreenRendering( 1 );

renderWindow->AddRenderer(renderer);

// Add the actors to the scene

renderer->AddActor(actor);

renderer->SetBackground(1,1,1); // Background color white

renderWindow->Render();

vtkSmartPointer<vtkWindowToImageFilter> windowToImageFilter =

vtkSmartPointer<vtkWindowToImageFilter>::New();

windowToImageFilter->SetInput(renderWindow);

windowToImageFilter->Update();

vtkSmartPointer<vtkPNGWriter> writer =

vtkSmartPointer<vtkPNGWriter>::New();

writer->SetFileName("screenshot.png");

writer->SetInputConnection(windowToImageFilter->GetOutputPort());

writer->Write();

return 0;

}

# Discussion

See FreeCAD topic here: <https://forum.freecadweb.org/viewtopic.php?f=8&t=23620>