The introduction of FreeCAD compilation (Linux version)

1. Requirement of Linux systems

The Ubutu (version =16) or Debian (Version > 8 Jessie) are recommended.

2. Install the dependence libraries of FreeCAD

The detail introduction could be referred to:

http://www.freecadweb.org/wiki/?title=CompileOnUnix

The following libraries have to be installed for compilation of FreeCAD and Opencadcade. On Debian-based systems (Debian, Ubuntu, Mint, etc...) it is quite easy to get all needed dependencies installed. Most of the libraries are available via apt-get or synaptic package manager. And a "*.sh"* file could be created and include these commands to install the libraries in batches.

- g++
- git
- build-essential
- cmake
- python
- python-matplotlib
- libtool
- libcoin80-dev (Debian unstable(Jesse), testing, Ubuntu 13.10 and forward)
- libsoqt4-dev
- libxerces-c-dev
- libboost-dev
- libboost-filesystem-dev
- libboost-regex-dev
- libboost-program-options-dev
- libboost-signals-dev
- libboost-thread-dev
- libboost-python-dev
- libqt4-dev
- libqt4-opengl-dev
- qt4-dev-tools
- python-dev
- python-pyside

- pyside-tools
- ftgl-dev
- libxmu-dev
- tk8.5-dev
- libeigen3-dev
- libqtwebkit-dev
- libshiboken-dev
- libpyside-dev
- libode-dev
- swig
- libzipios++-dev
- libfreetype6
- libfreetype6-dev

Optionally you can also install these extra packages:

- libsimage-dev (to make Coin to support additional image file formats)
- checkinstall (to register your installed files into your system's package manager, so yo can easily uninstall later)
- python-pivy (needed for the 2D Drafting module)
- python-qt4 (needed for the 2D Drafting module)
- doxygen and libcoin60-doc (if you intend to generate source code documentation)
- libspnav-dev (for 3Dconnexion devices support like the Space Navigator or Space Pilot)

3. Install OpenCascade libraries

The community verison of OpenCascade is OCE which is easy to install and it can be downloaded directly from Github. And with apt-get or synaptic package manager, the 0.15.0 could be installed. The latest version is 0.17.2. Here the latest version is recommended for its improved Boolean operation capability.

1) Download the package "*tar.gz"* file.

   https://github.com/tpaviot/oce/releases

2) Uncompress the oce package and create a "build" file folder.

```
$ mkdir build
```

3) Go into the "build" file folder and compile the source code and install the libraries.

```
$ cmake ..
$ make
$ sudo make install
```

4. **Download and FreeCAD source code and compile it.**

Currently, the latest release version of FreeCAD is 17.0 but it is developing version 16.0 is stable version. So the ver 16.0 is recommended to be employed.

1) Download the source code package and uncompress the “tar.gz” file.

https://github.com/FreeCAD/FreeCAD/releases/tag/0.16

2) Uncompress the “build” file folder in the source file folder.

```
$ mkdir build
```

3) Go into the “build” file folder and compile the FreeCAD source code.

```
$ cmake -DFREECAD_USE_EXTERNAL_PIVY=1 -DCMAKE_BUILD_TYPE=Debug ..
$ make
```

4) Run FreeCAD execute file in “bin” file folder

```
$ ./bin/FreeCAD
```

The option of release version compilation could be selected with command:

```
$ cmake -DFREECAD_USE_EXTERNAL_PIVY=1 -DCMAKE_BUILD_TYPE=Debug ..
$ make
```

The CMakefile could be modified to turn on or turn off the compilations of different modules.

5. **Compile FreeCAD with QTCreator**

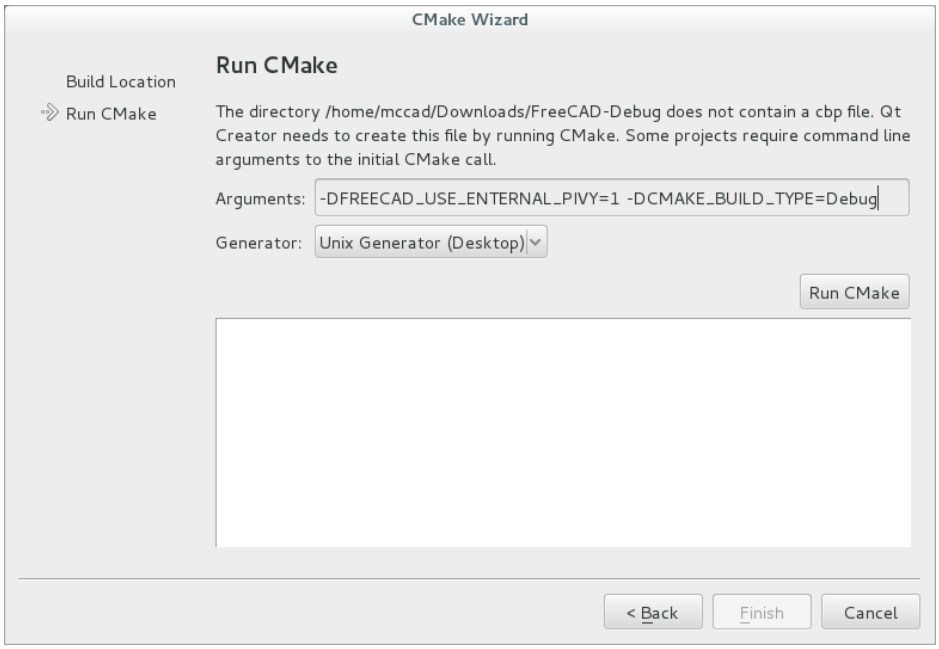
1) Install the QTcreator.

Qt Creator is the cross-platform IDE for QT projects, with apt-get or synaptic manager, the compliable QTCreator version could be installed, normally, it is not the latest version. The latest version should be downloaded from the QT website and installed.

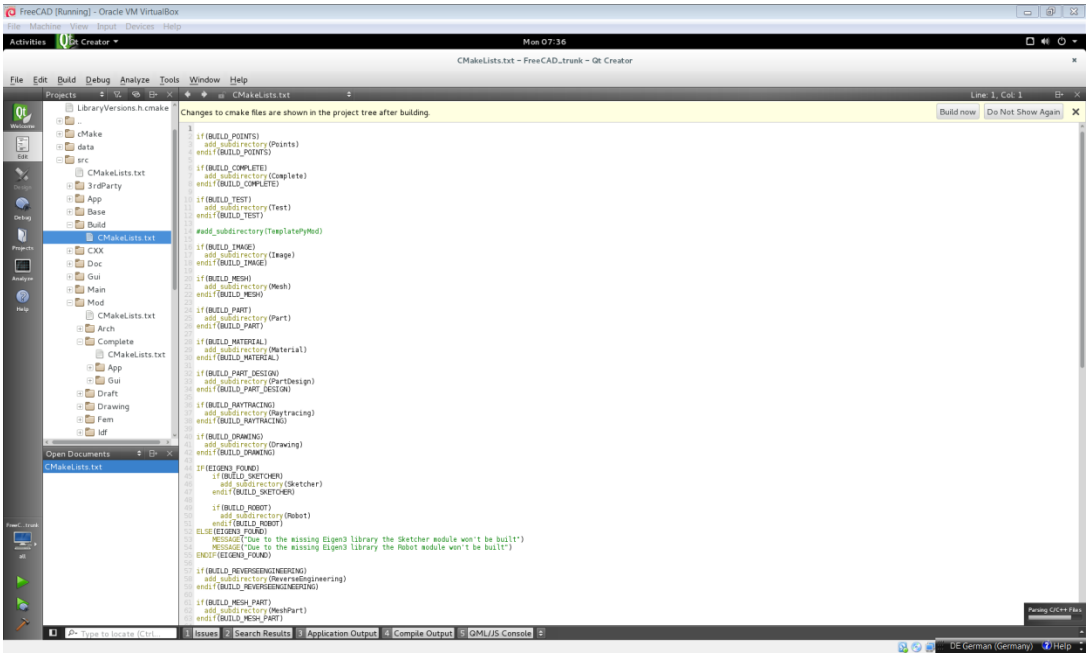
https://www.qt.io/ide/

2) Open QTCreator and load the FreeCAD project.

Open the source code file folder of FreeCAD and load the “CMakeLists.txt”, set the configurations and run the CMake, the source code and resource files are loaded in QTCreator IDE.



3) With the build function of QTCreator, the source code could be compiled and executed.



6. Add New Module

1) Create new module “McCad”

The detail process could be referred to:

http://www.freecadweb.org/wiki/index.php?title=Module_Creation

Creating a new application module in FreeCAD is rather simple. In the FreeCAD development tree exists the *FreeCAD Build Tool* (fcbt) that does the most important things for you. It is a Python script located under

```
$ FreeCAD-trunk/src/Tools/fcbt.py
```

When your python interpreter is correctly installed you can execute the script from a command line with

```
python fcbt.py
```

It will display the following menu:

```
FreeCAD Build Tool
Usage:
   fcbt <command name> [command parameter]
possible commands are:
 - DistSrc         (DS)   Build a source Distr. of the current source
tree
 - DistBin         (DB)   Build a binary Distr. of the current source
tree
 - DistSetup       (DI)   Build a Setup Distr. of the current source
tree
 - DistSetup       (DUI)  Build a User Setup Distr. of the current
source tree
 - DistAll         (DA)   Run all three above modules
 - BuildDoc        (BD)   Create the documentation (source docs)
 - NextBuildNumber (NBN)  Increase the Build Number of this Version
 - CreateModule    (CM)   Insert a new FreeCAD Module in the module
directory

For help on the modules type:
  fcbt <command name> ?
```

At the command prompt enter CM to start the creation of a module:

```
Insert command: CM
Please enter a name for your application: McCad
```

After pressing enter fcbt starts copying all necessary files for your module in a new folder at "*FreeCAD-trunk/src/Mod/McCad/*"

2) Add new module into project

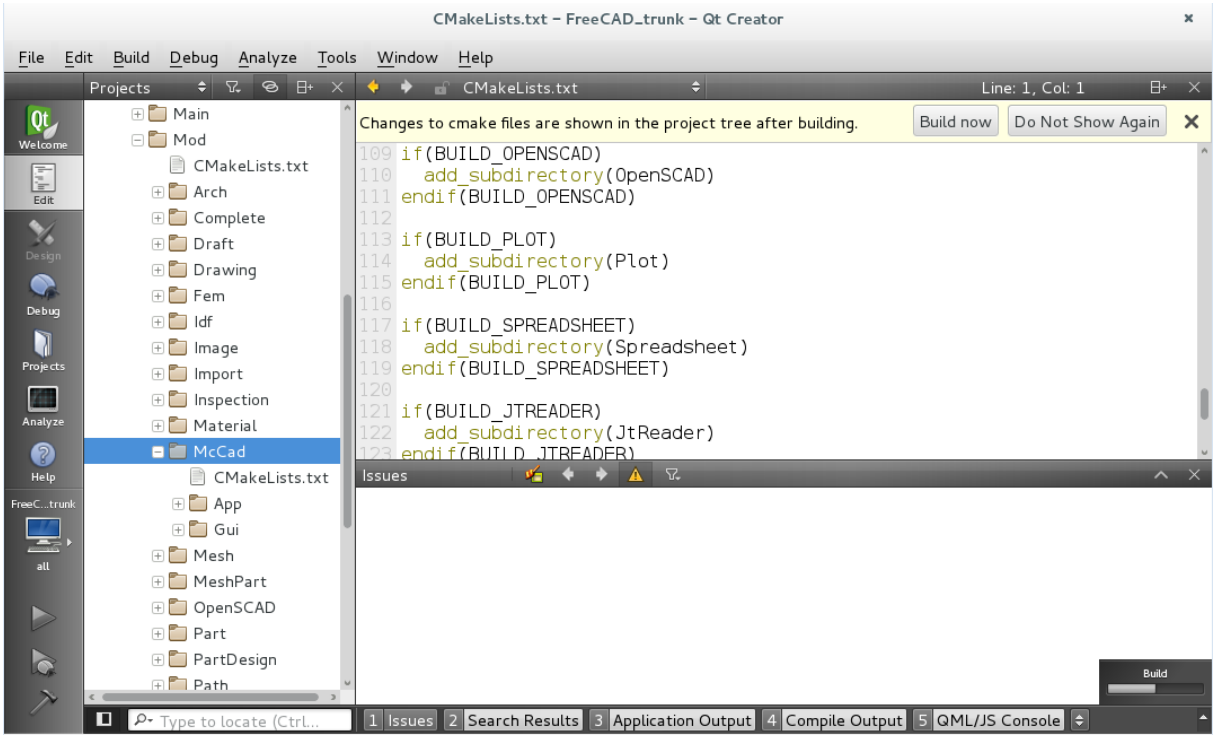
Modify the CMakeList.txt file at the folder "*FreeCAD-trunk/CMakeList.txt*", Define the module name and add a compilation switch "McCad" and add it:

```
OPTION(BUILD_MCCAD "Build the McCad Module" ON)
```

Modify the CMakeList.txt file at the folder "*FreeCAD-trunk/Mod/CMakeList.txt*":

```
if(BUILD_MCCAD)
  add_subdirectory(McCad)
endif(BUILD_MCCAD)
```

Re-Open the project, the McCad module will be added into the project and a folder could be found at "*FreeCAD-trunk/Mod/McCad*".



3) Configure the McCad Module to create a new workbench

The *"initGui.py"* file at the folder *"Mod/McCad"* will be loaded when the GUI starts. It is generated automatically and could be edited for modifying the GUI.

- **Define the new functions:**

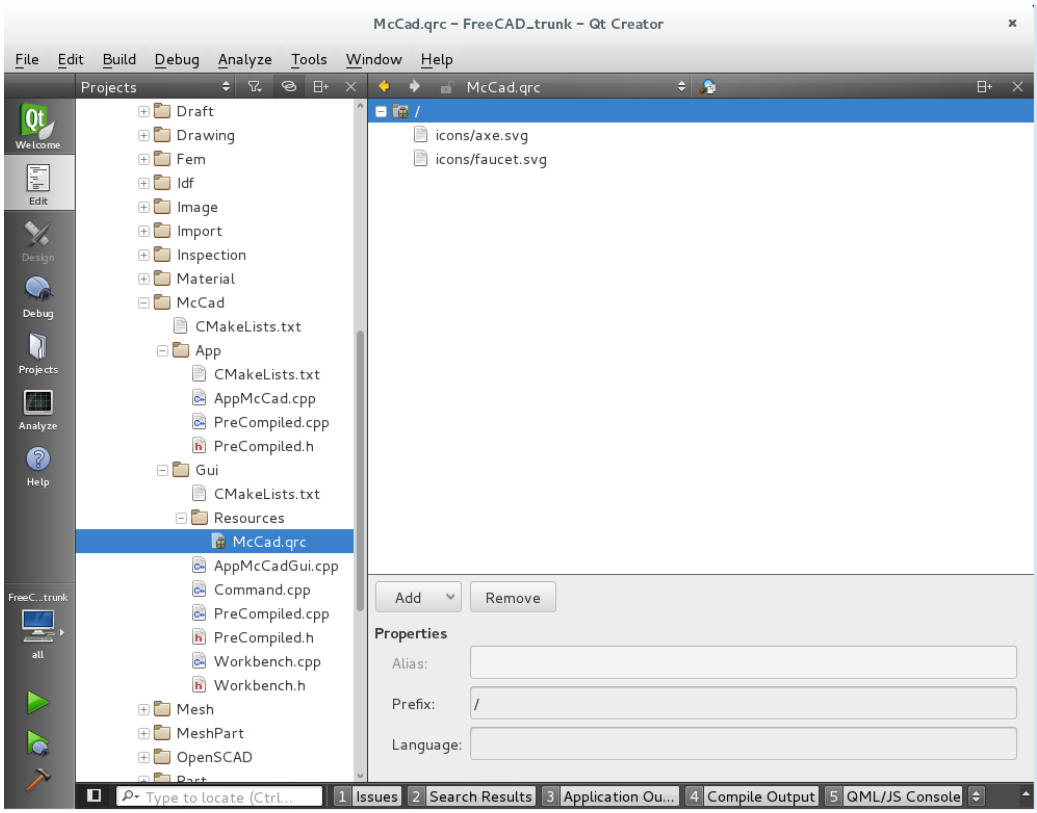
Modifying "*command.cpp"* at the folder *"Mod/McCad/Gui"*.

```
DEF_STD_CMD(CmdMcCadSplit);      //Define the class CmdMcCadSplit

CmdMcCadSplit::CmdMcCadSplit()
{
    sAppModule    = "McCad" ;
    sGroup        = QT_TR_NOOP("McCad");
    sMenuText     = QT_TR_NOOP("Split");
    sToolTipText  = QT_TR_NOOP("Split");
    sWhatsThis    = QT_TR_NOOP("Split");
    sStatusTip     = QT_TR_NOOP("Split");
    sPixmap       = axe.svg;       // The icon of this function
    sAccel        = "CTRL+S";      // Define the hot key
}
```

- **Add the icons**

At the Gui folder there are *"mccad.qrc"* resource file, click the *"mccad.qrc"* node and the resource dialogbox displays, add the .svg files as the icon files. The .svg files must be storage at the "Mod/McCad/Gui/icons" file folder.



- **new buttons and items on the MenuBar and ToolBar on the interfaces:**

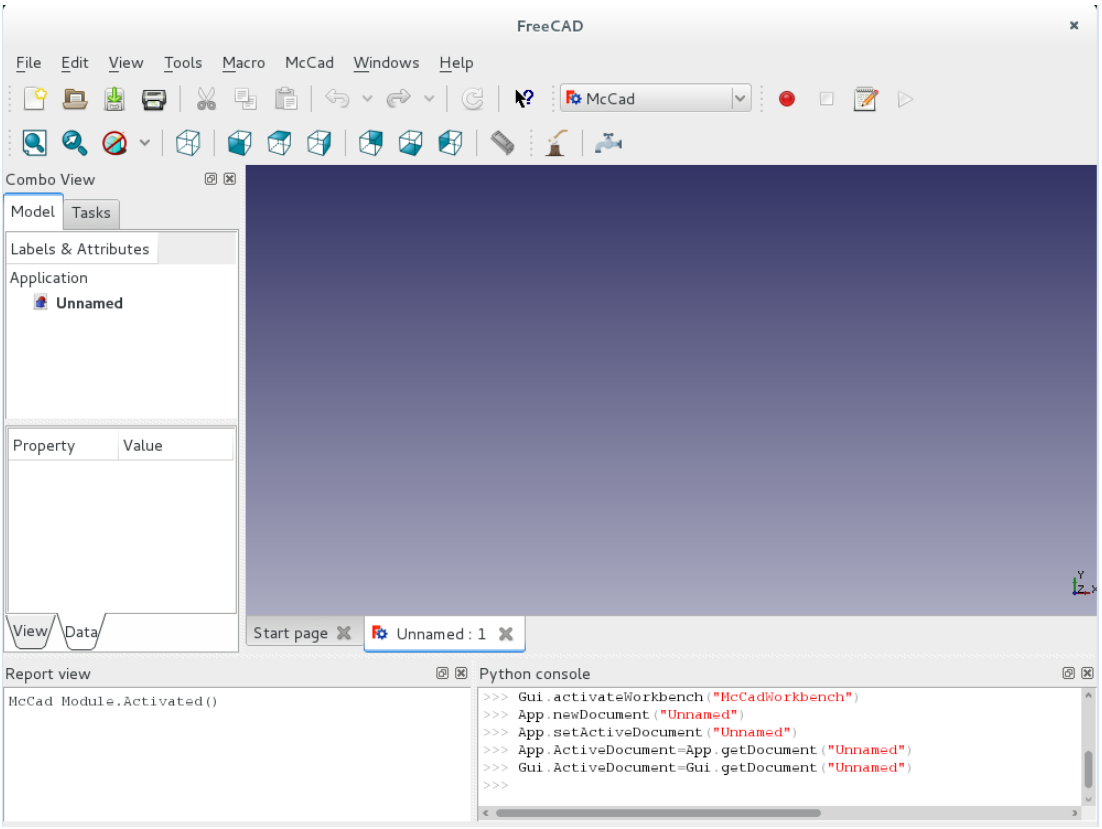
Modifying functions *setupMenuBar* ***and*** *setupToolBar* in the *"workbench.cpp"* at the folder *"Mod/McCad/Gui"*.

```
Gui::MenuItem* Workbench::setupMenuBar() const
{
    Gui::MenuItem * root = StdWorkbench::setupMenuBar();
    Gui::MenuItem * item = root->findItem("&Windows");
    Gui::MenuItem * mccad = new Gui::MenuItem;

    root->insertItem(item, mccad);
    mccad->setCommand("McCad");
    *mccad<<"McCad_Split"<<"McCad_Void";
    return root;
}

Gui::MenuItem* Workbench::setupToolBar() const
{
    Gui::MenuItem * root = StdWorkbench::setupToolBar();
    Gui::MenuItem * mccad = new Gui::ToolBarItem(root);
    mccad->setCommand("McCad Tools");
    *mccad<<"McCad_Split"<<"Separator"<<""McCad_Void";
    return root;
}
```

Finally, after the compilation, the FreeCAD with new module and new functions are added.


FreeCAD
File Edit View Tools Macro McCad Windows Help
McCad
Combo View
Model Tasks
Labels & Attributes
Application
Unnamed
Property Value
View Data
Start page
Unnamed : 1
Report view
McCad Module.Activated()
Python console
>>> Gui.activateWorkbench("McCadWorkbench")
>>> App.newDocument("Unnamed")
>>> App.setActiveDocument("Unnamed")
>>> App.ActiveDocument=App.getDocument("Unnamed")
>>> Gui.ActiveDocument=Gui.getDocument("Unnamed")
>>>