The usage of Vaa3D plugins

Jingzhou Yuan

2021-12-15

Introduction

- **A Swiss Army knife for**

exploring big big image data

·

$\longrightarrow$

FAST
Vaa3D visualizes and explores big 3D/4D/5D images with giga-voxels and even tera-voxels, within seconds or sub-seconds!

COOL
Vaa3D extracts complex surface objects from images, and performs comprehensive analyses such as brain connectome mapping.

EXTENSIBLE
100+ plugins for image acquisition, microsurgery, data management and analysis, and massive-scale pipelining

Extensibility: Plugins

Plugins are independent with main program

1.Interact at dynamic library level (*.so/*.dll/*.dylib)

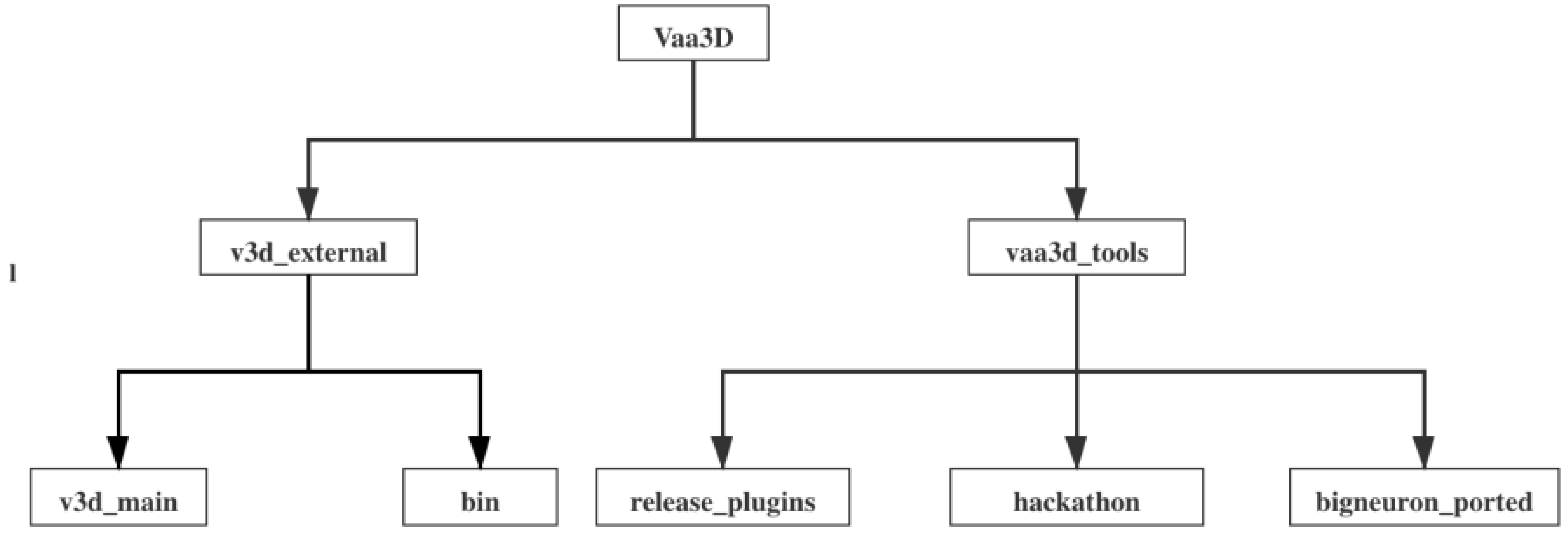
2.Addition/update of plugin does not require re-compiling of main program

Vaa3D provide facilities that

1.automatically detect, load, and call plugins

2.reserves extensible interface for new plugins

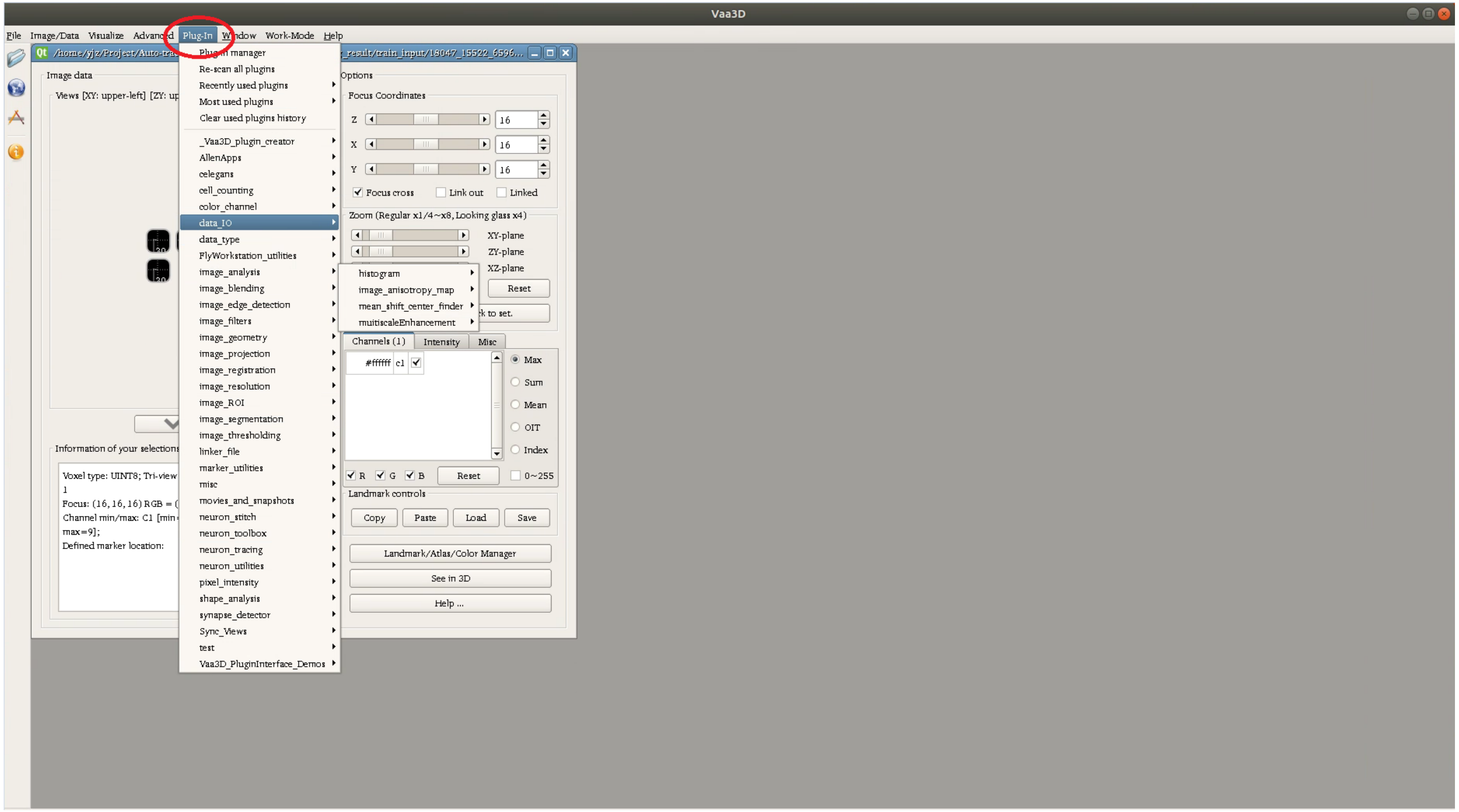
Vaa3D architecture



released_plugin:
1.Generally and usually use
2.pre-built with binary, automatic built while compiling

hackathon and bigneuron_ported:
1.they are third-party plugins
2.we could write our own plugins there
3.need to compile manually

Usage of plugins: through main menu



Voxel type: UINT8; Tri-view
1
Focus: (16, 16, 16) RGB = (
Channel min/max: C1 [min
max=9];
Defined marker location:

Usage of plugins: through command line

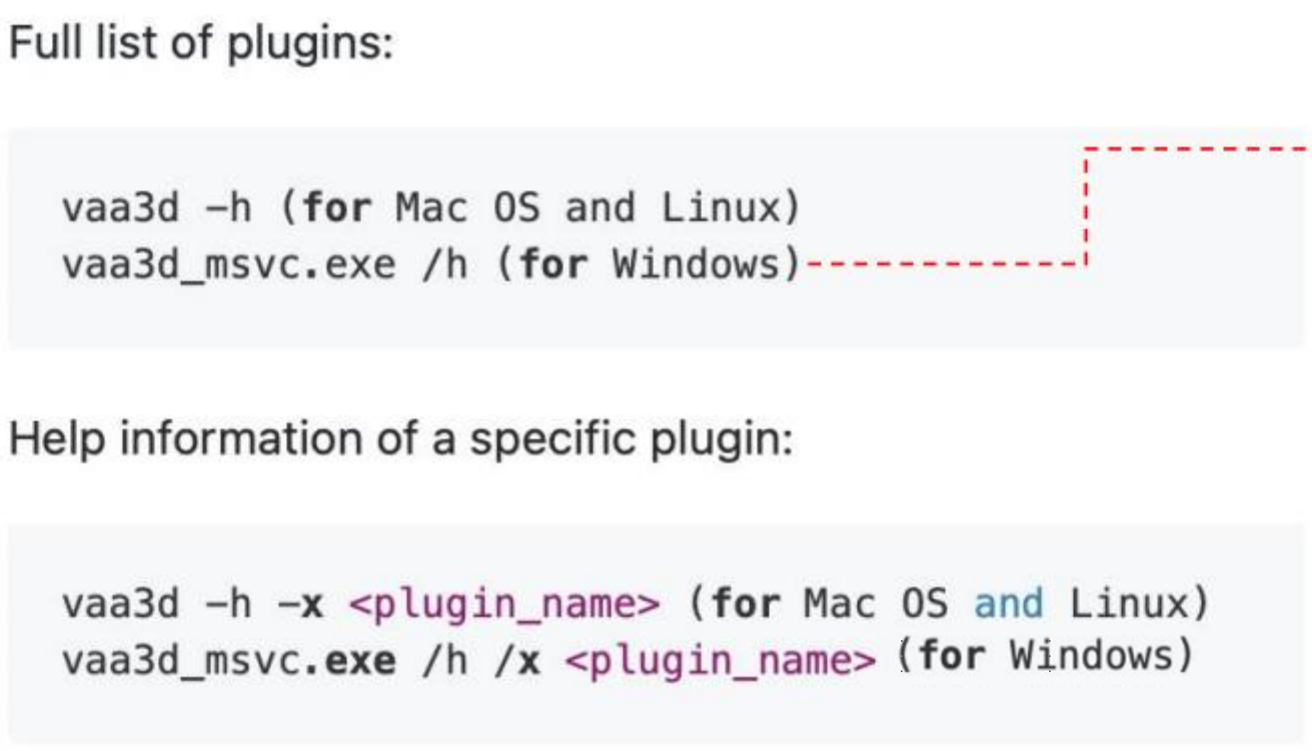
large scale data

Additional configurable parameters for some plugins

Better exception control

Speed up via parallelization

Usage of plugins: through command line



Full list of plugins:

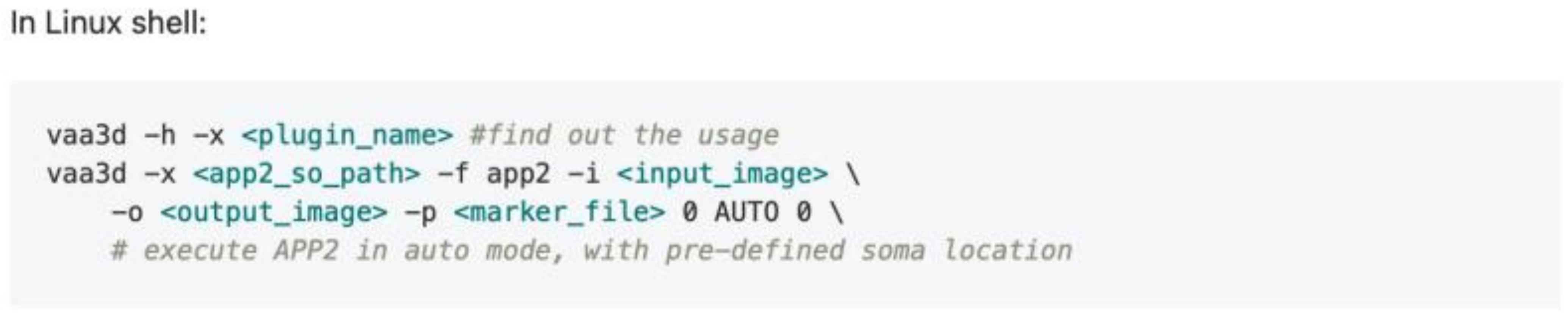
```
vaa3d -h (for Mac OS and Linux)
vaa3d_msvc.exe /h (for Windows)
```

Arguments form is different for Windows

Help information of a specific plugin:

```
vaa3d -h -x <plugin_name> (for Mac OS and Linux)
vaa3d_msvc.exe /h /x <plugin_name> (for Windows)
```

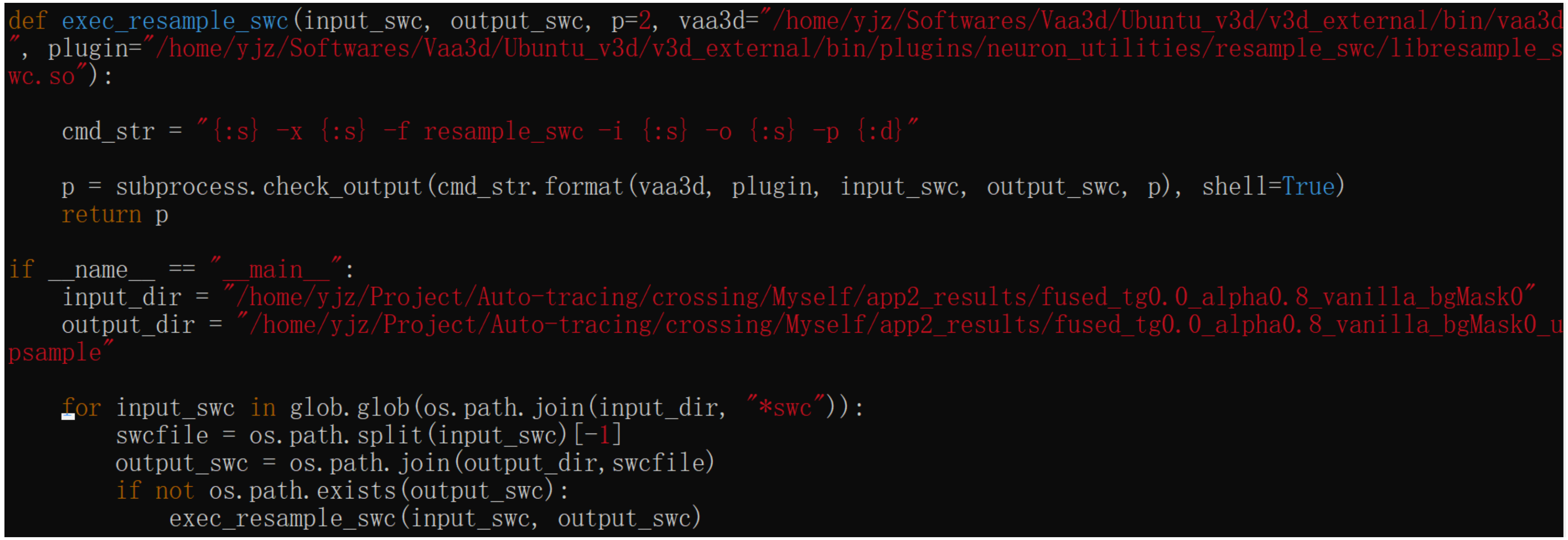
Usage of plugins: through command line



In Linux shell:

```
vaa3d -h -x <plugin_name> #find out the usage
vaa3d -x <app2_so_path> -f app2 -i <input_image> \
    -o <output_image> -p <marker_file> 0 AUTO 0 \
    # execute APP2 in auto mode, with pre-defined soma location
```

an example of the usage of plugin through python

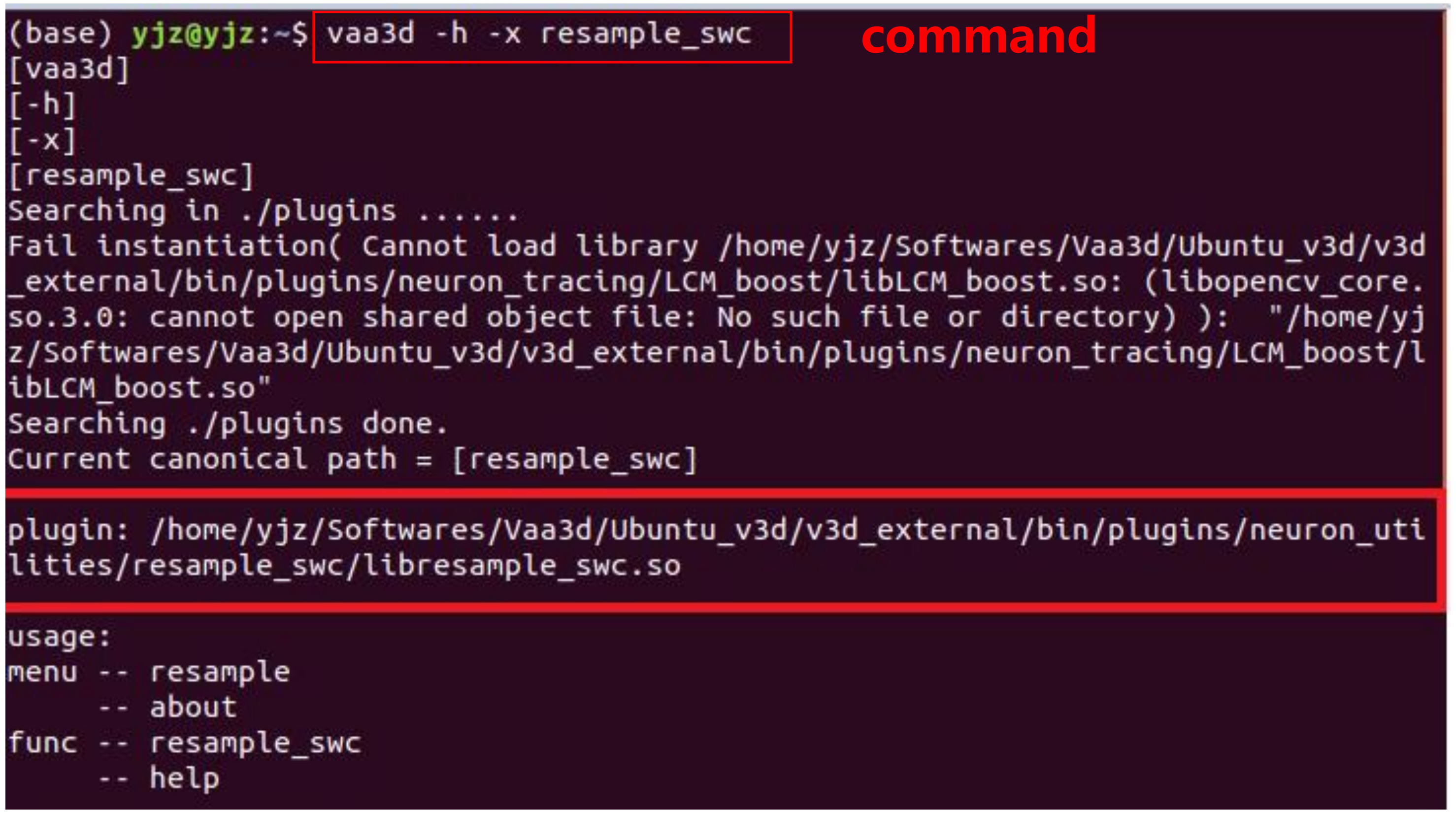


```
def exec_resample_swc(input_swc, output_swc, p=2, vaa3d="/home/yjz/Softwares/Vaa3d/Ubuntu_v3d/v3d_external/bin/vaa3d
", plugin="/home/yjz/Softwares/Vaa3d/Ubuntu_v3d/v3d_external/bin/plugins/neuron_utilities/resample_swc/libresample_s
wc.so"):

    cmd_str = "{:s} -x {:s} -f resample_swc -i {:s} -o {:s} -p {:d}"

    p = subprocess.check_output(cmd_str.format(vaa3d, plugin, input_swc, output_swc, p), shell=True)
    return p

if __name__ == "__main__":
    input_dir = "/home/yjz/Project/Auto-tracing/crossing/Myself/app2_results/fused_tg0.0_alpha0.8_vanilla_bgMask0"
    output_dir = "/home/yjz/Project/Auto-tracing/crossing/Myself/app2_results/fused_tg0.0_alpha0.8_vanilla_bgMask0_u
psample"

    for input_swc in glob.glob(os.path.join(input_dir, "*swc")):
        swcfile = os.path.split(input_swc)[-1]
        output_swc = os.path.join(output_dir,swcfile)
        if not os.path.exists(output_swc):
            exec_resample_swc(input_swc, output_swc)
```

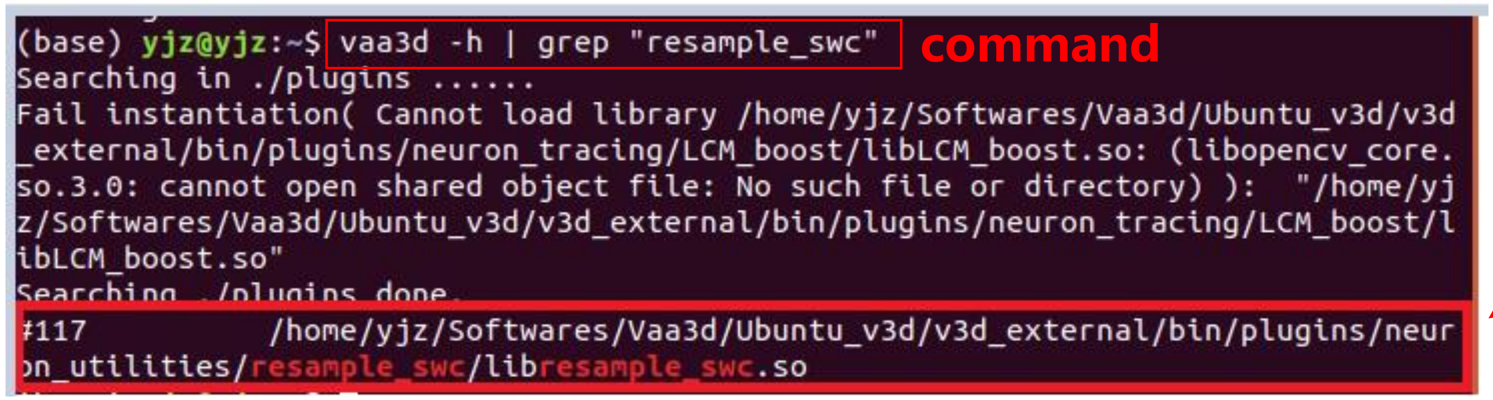
Finding out the specific plugin you want

- For built-in plugins:
  - GUI: click Plug-ins, drop down and find the specific plugins
  - Command line: vaa3d –h for information, or find plugin with specific name, try:
    - vaa3d –h | grep “keyword”
    - vaa3d –x “plugin_path” –f help
- For third-party plugins:
  - Go to the directory:
    - vaa3d_tools/hackathon & vaa3d_tools/bigneuron_ported
  - Search by keyword:
    - Find vaa3d_tools/ -name “*keyword*” –type f

Usage of plugins: through command line-detailed

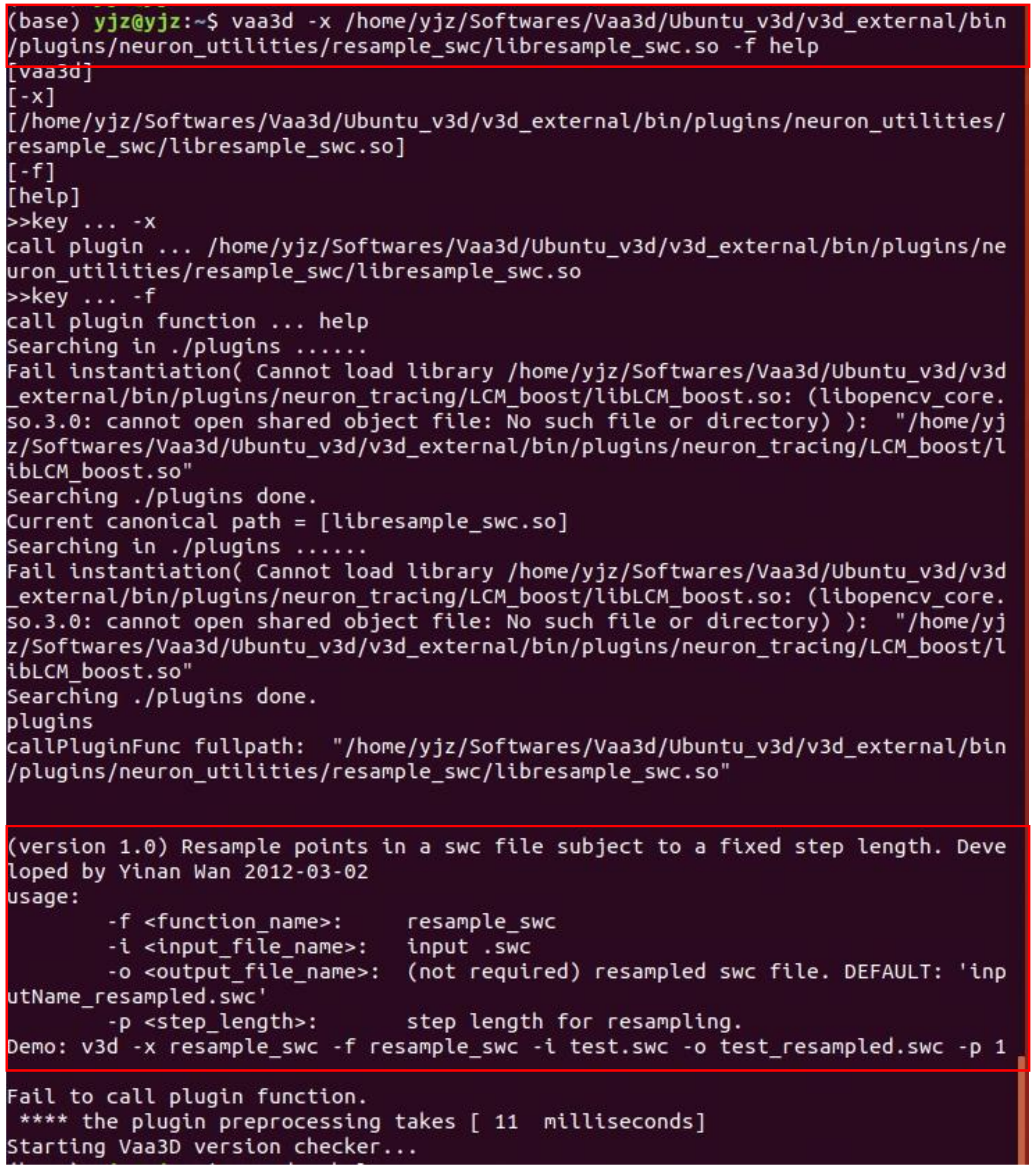


```
(base) yjz@yjz:~$ vaa3d -h -x resample_swc
[vaa3d]
[-h]
[-x]
[resample_swc]
Searching in ./plugins ......
Fail instantiation( Cannot load library /home/yjz/Softwares/Vaa3d/Ubuntu_v3d/v3d
_external/bin/plugins/neuron_tracing/LCM_boost/libLCM_boost.so: (libopencv_core.
so.3.0: cannot open shared object file: No such file or directory) ):  "/home/yj
z/Softwares/Vaa3d/Ubuntu_v3d/v3d_external/bin/plugins/neuron_tracing/LCM_boost/l
ibLCM_boost.so"
Searching ./plugins done.
Current canonical path = [resample_swc]

plugin: /home/yjz/Softwares/Vaa3d/Ubuntu_v3d/v3d_external/bin/plugins/neuron_uti
lities/resample_swc/libresample_swc.so

usage:
menu -- resample
     -- about
func -- resample_swc
     -- help
```



```
(base) yjz@yjz:~$ vaa3d -h | grep "resample_swc"
Searching in ./plugins ......
Fail instantiation( Cannot load library /home/yjz/Softwares/Vaa3d/Ubuntu_v3d/v3d
_external/bin/plugins/neuron_tracing/LCM_boost/libLCM_boost.so: (libopencv_core.
so.3.0: cannot open shared object file: No such file or directory) ):  "/home/yj
z/Softwares/Vaa3d/Ubuntu_v3d/v3d_external/bin/plugins/neuron_tracing/LCM_boost/l
ibLCM_boost.so"
Searching ./plugins done.
#117          /home/yjz/Softwares/Vaa3d/Ubuntu_v3d/v3d_external/bin/plugins/neur
on_utilities/resample_swc/libresample_swc.so
```

Usage of plugins: through command line-detailed

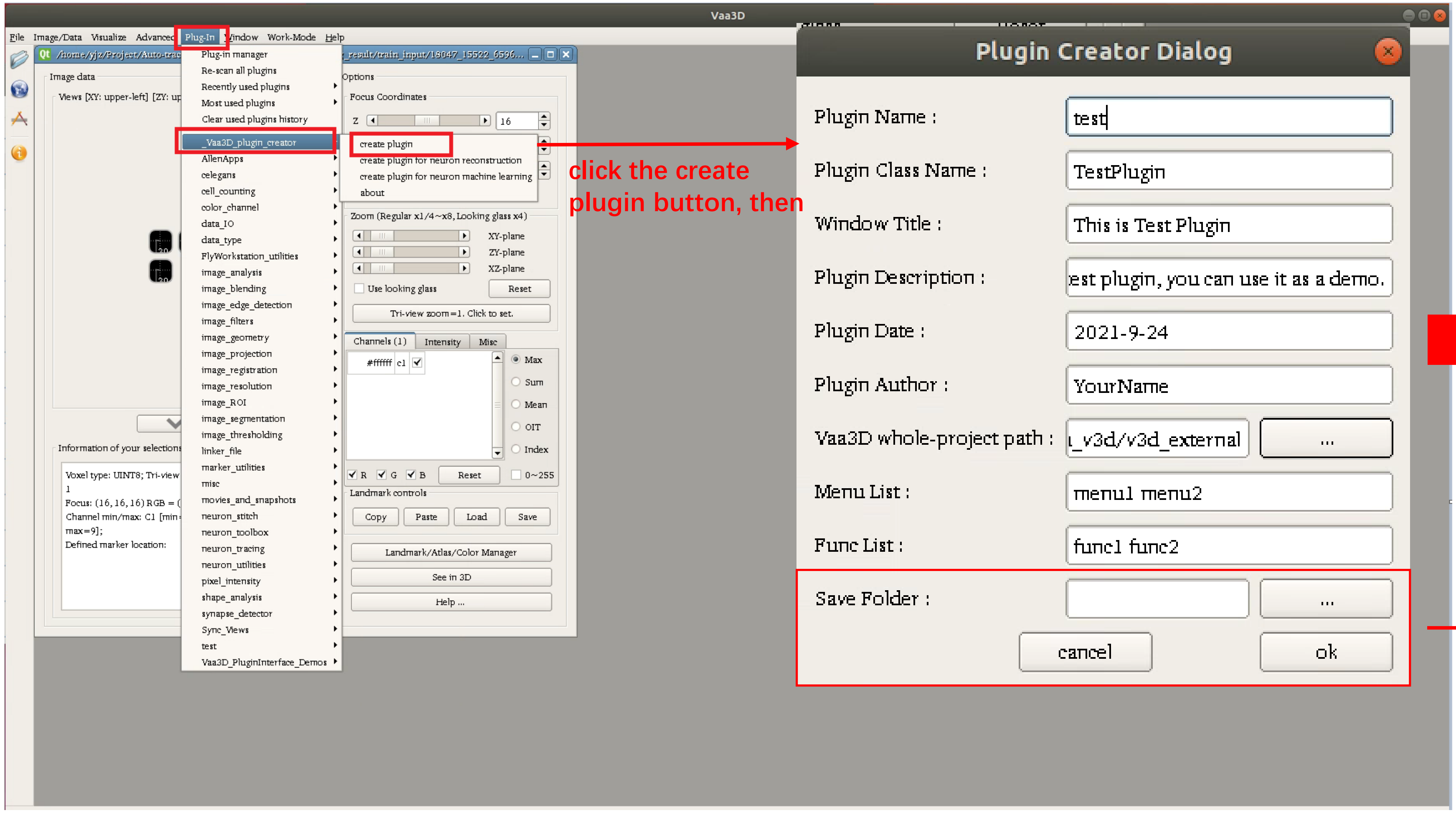


```
(base) yjz@yjz:~$ vaa3d -x /home/yjz/Softwares/Vaa3d/Ubuntu_v3d/v3d_external/bin
/plugins/neuron_utilities/resample_swc/libresample_swc.so -f help
[vaa3d]
[-x]
[/home/yjz/Softwares/Vaa3d/Ubuntu_v3d/v3d_external/bin/plugins/neuron_utilities/
resample_swc/libresample_swc.so]
[-f]
[help]
>>key ... -x
call plugin ... /home/yjz/Softwares/Vaa3d/Ubuntu_v3d/v3d_external/bin/plugins/ne
uron_utilities/resample_swc/libresample_swc.so
>>key ... -f
call plugin function ... help
Searching in ./plugins ......
Fail instantiation( Cannot load library /home/yjz/Softwares/Vaa3d/Ubuntu_v3d/v3d
_external/bin/plugins/neuron_tracing/LCM_boost/libLCM_boost.so: (libopencv_core.
so.3.0: cannot open shared object file: No such file or directory) ):  "/home/yj
z/Softwares/Vaa3d/Ubuntu_v3d/v3d_external/bin/plugins/neuron_tracing/LCM_boost/l
ibLCM_boost.so"
Searching ./plugins done.
Current canonical path = [libresample_swc.so]
Searching in ./plugins ......
Fail instantiation( Cannot load library /home/yjz/Softwares/Vaa3d/Ubuntu_v3d/v3d
_external/bin/plugins/neuron_tracing/LCM_boost/libLCM_boost.so: (libopencv_core.
so.3.0: cannot open shared object file: No such file or directory) ):  "/home/yj
z/Softwares/Vaa3d/Ubuntu_v3d/v3d_external/bin/plugins/neuron_tracing/LCM_boost/l
ibLCM_boost.so"
Searching ./plugins done.
plugins
callPluginFunc fullpath:  "/home/yjz/Softwares/Vaa3d/Ubuntu_v3d/v3d_external/bin
/plugins/neuron_utilities/resample_swc/libresample_swc.so"


(version 1.0) Resample points in a swc file subject to a fixed step length. Deve
loped by Yinan Wan 2012-03-02
usage:
        -f <function_name>:     resample_swc
        -i <input_file_name>:   input .swc
        -o <output_file_name>:  (not required) resampled swc file. DEFAULT: 'inp
utName_resampled.swc'
        -p <step_length>:       step length for resampling.
Demo: v3d -x resample_swc -f resample_swc -i test.swc -o test_resampled.swc -p 1

Fail to call plugin function.
 **** the plugin preprocessing takes [ 11  milliseconds]
Starting Vaa3D version checker...
```

Command: view the plugin usage

Usage and Demo

Write our own plugin



Plugin Name : test

Plugin Class Name : TestPlugin

Window Title : This is Test Plugin

Plugin Description : est plugin, you can use it as a demo.

Plugin Date : 2021-9-24

Plugin Author : YourName

Vaa3D whole-project path : ı_v3d/v3d_external

Menu List : menu1 menu2

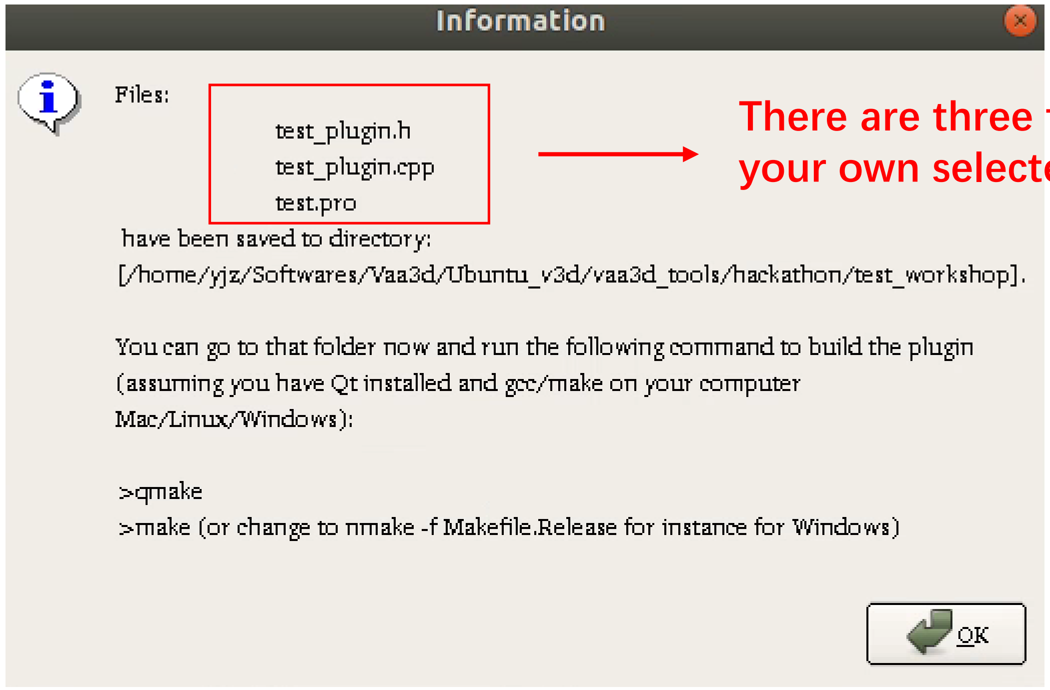
Func List : func1 func2

Save Folder :

Write some information

Select your own path(usually include in the hackathon file)

Write our own plugin



Information

Files:

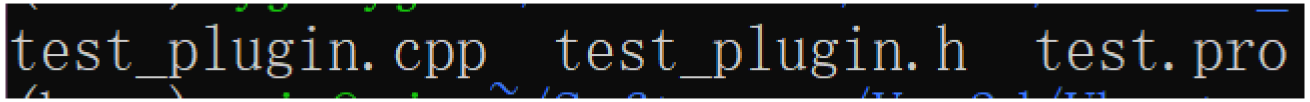
test_plugin.h
test_plugin.cpp
test.pro

have been saved to directory:
[/home/yjz/Softwares/Vaa3d/Ubuntu_v3d/vaa3d_tools/hackathon/test_workshop].

You can go to that folder now and run the following command to build the plugin (assuming you have Qt installed and gcc/make on your computer Mac/Linux/Windows):

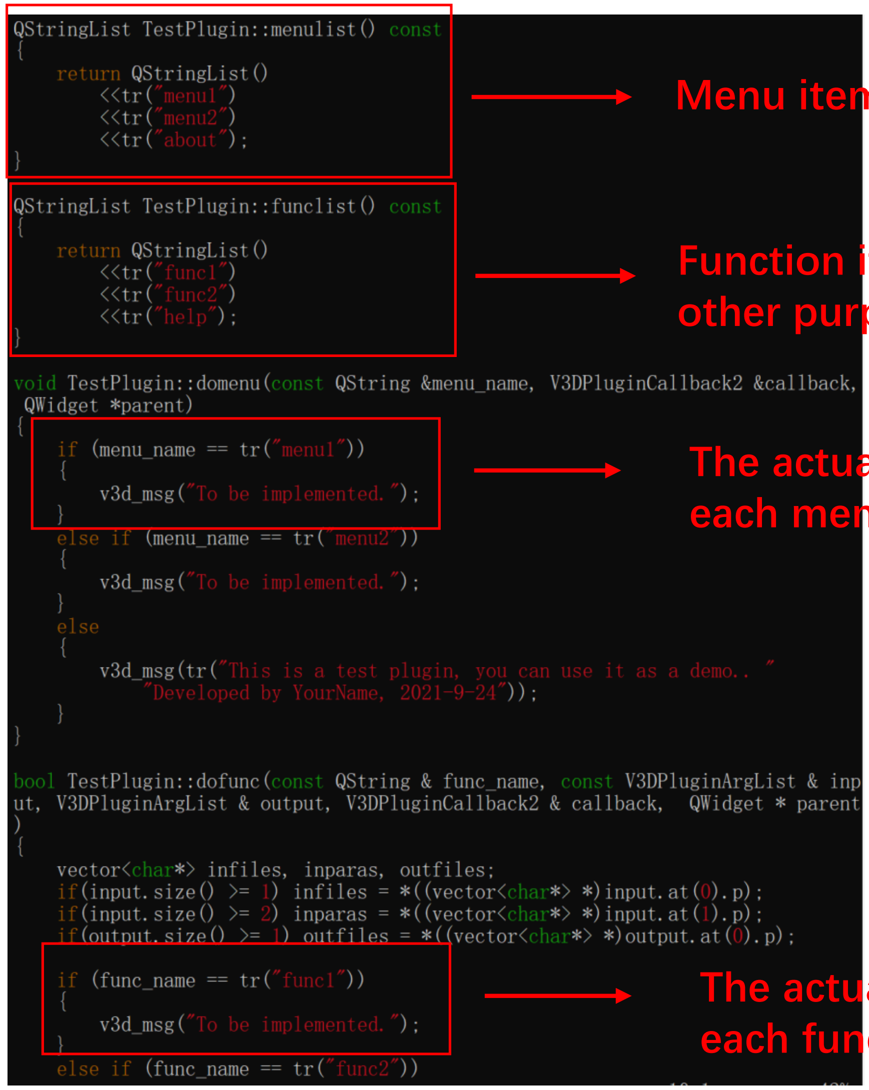
```
>qmake
>make (or change to nmake -f Makefile.Release for instance for Windows)
```

There are three files in your own selected path



```
test_plugin.cpp   test_plugin.h   test.pro
```

Write our own plugin



```
QStringList TestPlugin::menulist() const
{
    return QStringList()
        <<tr("menu1")
        <<tr("menu2")
        <<tr("about");
}
QStringList TestPlugin::funclist() const
{
    return QStringList()
        <<tr("func1")
        <<tr("func2")
        <<tr("help");
}

void TestPlugin::domenu(const QString &menu_name, V3DPluginCallback2 &callback,
 QWidget *parent)
{
    if (menu_name == tr("menu1"))
    {
        v3d_msg("To be implemented.");
    }
    else if (menu_name == tr("menu2"))
    {
        v3d_msg("To be implemented.");
    }
    else
    {
        v3d_msg(tr("This is a test plugin, you can use it as a demo.. "
            "Developed by YourName, 2021-9-24"));
    }
}

bool TestPlugin::dofunc(const QString & func_name, const V3DPluginArgList & inp
ut, V3DPluginArgList & output, V3DPluginCallback2 & callback,  QWidget * parent
)
{
    vector<char*> infiles, inparas, outfiles;
    if(input.size() >= 1) infiles = *((vector<char*> *)input.at(0).p);
    if(input.size() >= 2) inparas = *((vector<char*> *)input.at(1).p);
    if(output.size() >= 1) outfiles = *((vector<char*> *)output.at(0).p);

    if (func_name == tr("func1"))
    {
        v3d_msg("To be implemented.");
    }
    else if (func_name == tr("func2"))
```

Menu items in GUI

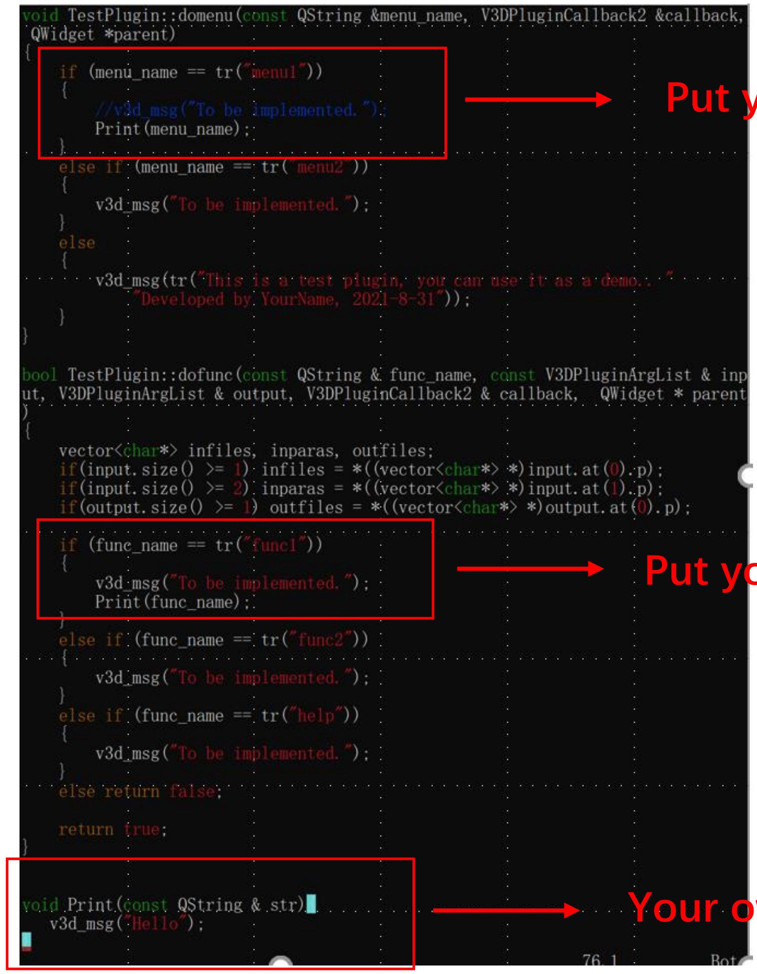
Function items for any other purposes

The actual action(s) of each menu item

The actual action(s) of each function

We could write needed function in test_plugin.cpp

Write our own plugin



```
void TestPlugin::domenu(const QString &menu_name, V3DPluginCallback2 &callback,
 QWidget *parent)
{
    if (menu_name == tr("menu1"))
    {
        //v3d_msg("To be implemented.");
        Print(menu_name);
    }
    else if (menu_name == tr("menu2"))
    {
        v3d_msg("To be implemented.");
    }
    else
    {
        v3d_msg(tr("This is a test plugin, you can use it as a demo.. "
            "Developed by YourName, 2021-8-31"));
    }
}

bool TestPlugin::dofunc(const QString & func_name, const V3DPluginArgList & inp
ut, V3DPluginArgList & output, V3DPluginCallback2 & callback,  QWidget * parent
)
{
    vector<char*> infiles, inparas, outfiles;
    if(input.size() >= 1) infiles = *((vector<char*> *)input.at(0).p);
    if(input.size() >= 2) inparas = *((vector<char*> *)input.at(1).p);
    if(output.size() >= 1) outfiles = *((vector<char*> *)output.at(0).p);

    if (func_name == tr("func1"))
    {
        v3d_msg("To be implemented.");
        Print(func_name);
    }
    else if (func_name == tr("func2"))
    {
        v3d_msg("To be implemented.");
    }
    else if (func_name == tr("help"))
    {
        v3d_msg("To be implemented.");
    }
    else return false;

    return true;
}


void Print(const QString & str)
  v3d_msg("Hello");

                                                          76,1          Bot
```

Put your own method in GUI

Put your own method in Function

Your own method

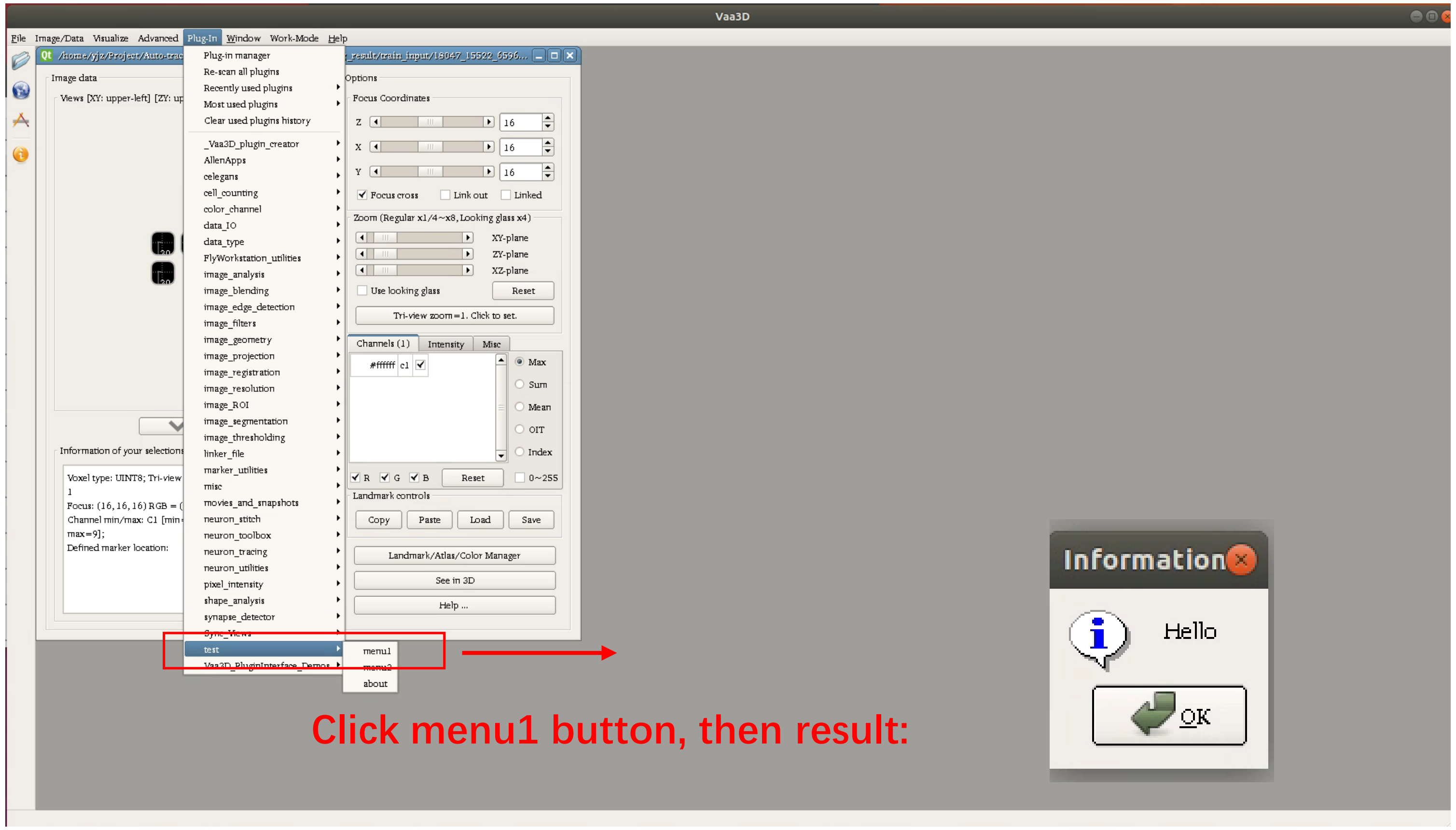
Writing example plugin: "Hello world"

And compile manually

After writing plugin: "Hello world"

Using qmake command in your path and then make

Our own plugin result



Voxel type: UINT8; Tri-view
1
Focus: (16, 16, 16) RGB = (
Channel min/max: C1 [min=
max=9];
Defined marker location:

Click menu1 button, then result:

Hello

Thank you !