Component developer guide / Mepp - 3D MEsh Processing Platform

About **Authors** **Download** **Documentation** **Component developer guide** **For developers** **Gallery** **Videos**

Component developer guide

Table of contents

1. Creating a new component (example : from component CGAL_Example)

Note : the MEPP "core" shall not normally be changed.

1.1. Your component must have a unique name and must be related to a category from the list below:

- analysis,
- compression,
- distance,
- remeshing,
- segmentation,
- tools,
- watermarking,

or optionally,

- examples.

a) This name must also be the name of the subdirectory (containing the source code of your component) in this category (see CGAL_Example component).

b) The name of the C++ class for this component must be of type **YourComponent**_Component (see

CGAL_Example component).

c) The name of the C++ plugin class (which goes along with the above class) of your component must be of type mepp_component_**YourComponent**_plugin (see CGAL_Example component).

Important : beware of the "case sensitive" for these three points.

1.2. Follow CGAL_Example component in \src\components\Examples\CGAL_Example:

A "clean" and accurate "case sensitive" renaming in ".cpp", ".h" and ".hxx" files should allow you to easily obtain the skeleton of your component.

So, we will take below as an example the creation of the new component Various_Tools who belong to the category Tools.

To do this, you must:

a) copy the \src\components\Examples\CGAL_Example folder in \src\components\Tools folder,

b) rename this "new" CGAL_Example folder to Various_Tools,

c) **VERY IMPORTANT** : recursively delete all ".svn" files from the Various_Tools folder (not necessary if you use git instead),

d) optionally uncomment line 10 (deletion of #) in \src\components\Tools\Various_Tools\cmake \use_components.txt file if you want to use another component within your own component, see 1.4. below (here we leave the comment because we do not want to use Curvature in Various_Tools),

e) in \src\components\Tools\Various_Tools\src folder, rename all files by changing CGAL_Example with Various_Tools (beware of the "case sensitive"),

f) finally, with a text editor and a "find and replace" function, replace (respecting the "case") in all ".cpp", ".h" and ".hxx" files in the \src\components\Tools\Various_Tools\src folder:

CGAL_Example by Various_Tools
and
CGAL_EXAMPLE by VARIOUS_TOOLS

Your menu is then declared in the mepp_component_Various_Tools_plugin.hxx file by linking it to the category mentioned above, and declaring the menu actions as shown below (see CGAL_Example component).

```
public:
    mepp_component_CGAL_Example_plugin() : mepp_component_plugin_interface() {}
    ~mepp_component_CGAL_Example_plugin()
    {
        delete actionStep_1; delete actionStep_2; delete actionStep_3;
        delete actionStep_4; delete actionStep_5; delete actionStep_6;
        delete actionStep_7; delete actionStep_8; delete actionStep_9;
    }

    void init(mainwindow* mainWindow, QList<QMdiSubWindow *> lw)
    {
        this->mw = mainWindow;
        this->lwindow = lw;
        this->mPluginName = this->metaObject()->className();

        // choice: menuTools, menuDistance_Quality_measure, menuAnalysis_Filtering, menuSegmentation, menuRemeshing_Su
        mParentMenu = mainWindow->menuExamples;

        // begin --- actions ---
        actionStep_1 = new QAction(tr("Triangulate And Random Color Facets"), this);
        if (actionStep_1)
            connect(actionStep_1, SIGNAL(triggered()), this, SLOT(step1()));

        actionStep_2 = new QAction(tr("Create Center Vertex"), this);
        if (actionStep_2)
            connect(actionStep_2, SIGNAL(triggered()), this, SLOT(step2()));

        actionStep_3 = new QAction(tr("Show Black And White Facets"), this);
        if (actionStep_3)
            connect(actionStep_3, SIGNAL(triggered()), this, SLOT(step3()));

        actionStep_4 = new QAction(tr("Draw Connections"), this);
        if (actionStep_4)
            connect(actionStep_4, SIGNAL(triggered()), this, SLOT(step4()));

        actionStep_5 = new QAction(tr("Set Position And Orientation"), this);
        if (actionStep_5)
            connect(actionStep_5, SIGNAL(triggered()), this, SLOT(step5()));

        actionStep_6 = new QAction(tr("New/Add Polyhedron"), this);
        if (actionStep_6)
            connect(actionStep_6, SIGNAL(triggered()), this, SLOT(step6()));

        actionStep_7 = new QAction(tr("Load File From Component"), this);
        if (actionStep_7)
            connect(actionStep_7, SIGNAL(triggered()), this, SLOT(step7()));

        actionStep_8 = new QAction(tr("Save File From Component"), this);
        if (actionStep_8)
            connect(actionStep_8, SIGNAL(triggered()), this, SLOT(step8()));

        actionStep_9 = new QAction(tr("Sample to use Curvature component from this component"), this);
        if (actionStep_9)
            connect(actionStep_9, SIGNAL(triggered()), this, SLOT(step9()));
        // end --- actions ---
    }

    QList<QAction*> actions() const
    {
        return QList<QAction*>()    << actionStep_1
                                    << actionStep_2
                                    << actionStep_3
                                    << NULL         // menu separator
                                    << actionStep_4
                                    << actionStep_5
                                    << NULL         // menu separator
                                    << actionStep_6
                                    << NULL         // menu separator
                                    << actionStep_7
                                    << actionStep_8
                                    << NULL         // menu separator
                                    << actionStep_9;
    }
```

From there, the "CMake script" takes care of everything, there is no need to touch a single line of code in the MEPP kernel (nor the mepp_config.h.in file, nor the polyhedron_enriched_polyhedron.h file, nor the mainwindow.ui file for the menu).
Just simply invoke CMake (see readme_EN_Windows_VSxxxx.txt, readme_EN_Linux.txt or readme_EN_Mac_OS_X.txt).

1.3. Comment your new component with Doxygen

See tutorial for Doxygen: Franck Hecht - http://franckh.developpez.com/tutoriels/outils/doxygen/ (not exhaustive but generally sufficient).

1.4. Using the code of a component X in your component (without dialogs management of component X at this time):

Note : in this example, we use the Curvature component in the CGAL_Example component.

a) In \src\components\Examples\CGAL_Example\cmake\use_components.txt set the ".c"/".cpp" files you need with the CMake "set" function (see last line):
set(use_components ../../Analysis/Curvature/src/Curvature_Component.cpp ../../Analysis/Curvature/src/extract_Vpropres.cpp)

b) In \src\components\Examples\CGAL_Example\mepp_component_CGAL_Example_plugin.cpp set the following lines at the top of the file:

```
// we want to use Curvature component
#include "../../../Analysis/Curvature/src/Curvature_Component.h"
typedef boost::shared_ptr<Curvature_Component> Curvature_ComponentPtr;
// we want to use Curvature component
```

?

c) Then the Curvature call and use in \src\components\Examples\CGAL_Example\mepp_component_CGAL_Example_plugin.cpp happens like this, see **step9** function() below:

```
void mepp_component_CGAL_Example_plugin::step9()
{
    // active viewer
    if (mw->activeMdiChild() != 0)
    {
        Viewer* viewer = (Viewer *)mw->activeMdiChild();
        PolyhedronPtr polyhedron_ptr = viewer->getScenePtr()->get_polyhedron();

        QApplication::setOverrideCursor(Qt::WaitCursor);

        // we use CGAL_Example component here (as usual)
        CGAL_Example_ComponentPtr component_ptr = findOrCreateComponentForViewer<CGAL_Example_ComponentPtr, CGAL_Exampl
        component_ptr->TriangulateAndRandomColorFacets(polyhedron_ptr);

        component_ptr->set_init(2);
        // we use CGAL_Example component here (as usual)

        // we use Curvature component here
        bool IsGeo;
        double radius;
        Curvature_ComponentPtr component_ptr_curvature = findOrCreateComponentForViewer<Curvature_ComponentPtr, Curvatu

        // params
        radius = 0.001;
        IsGeo=true;

        mw->statusBar()->showMessage(tr("Curvature..."));
        component_ptr_curvature->principal_curvature(polyhedron_ptr,IsGeo,radius);
        mw->statusBar()->showMessage(tr("Curvature is done"));

        component_ptr_curvature->set_init(2);

        component_ptr_curvature->ConstructColorMap(polyhedron_ptr,1);
```

?

```
            viewer->recreateListsAndUpdateGL();
            // we use Curvature component here
        }

        QApplication::restoreOverrideCursor();
    }
```

2. Description of component files (example : from component CGAL_Example)

CGAL_Example_Polyhedron.h

Includes all statements and all related typedefs to CGAL and to the polyhedron.
As it includes the polyhedron, simply include this file in your component to access the polyhedron and declared typedefs.
It is therefore advisable to include this file and not directly the MEPP polyhedron.h file.

CGAL_Example_Items.h

Includes all definitions of component items.
Allows you to add variables and methods to the vertices, edges and facets of the polyhedron and to the polyhedron itself.

CGAL_Example_Component.h/.cpp

Defines the component API (programming interface).
This is the set of functions introduced to platform developers. The component is used exclusively through this API.
No GUI element should be present inside of these files, or their dependent files.

mepp_component_CGAL_Example_plugin.h/.cpp

This is the link between the GUI and the component.
All GUI elements should be located within these files (and their dependent files if needed). Communicates with the component through the API defined in CGAL_Example_Component.h.

3. Detailed component explanation (example : from component CGAL_Example)

Component files (plugin, ie dynamic library): mepp_component_CGAL_Example_plugin.h/.cpp
Component files (CGAL pure part): CGAL_Example_Component.h/.cpp

3.1. Common methods to all components (see mepp_component_CGAL_Example_plugin.cpp):

void mepp_component_CGAL_Example_plugin::pre_draw(), method automatically called before the polyhedron rendering.
It is possible to add rendering of component specific elements. All pre_draw() methods are called before rendering.
It is advisable to allow enable/disable pre_draw rendering (when it is not empty) through one or more variables related to the GUI (pressing a button, click on a menu item).

void mepp_component_CGAL_Example_plugin::post_draw(), method automatically called after the polyhedron rendering.
It is possible to add rendering of component specific elements. All post_draw() methods are called after rendering.
It is advisable to allow enable/disable post_draw rendering (when it is not empty) through one or more variables related to the GUI (pressing a button, click on a menu item).

---

void mepp_component_CGAL_Example_plugin::pre_draw_all_scene(), same as pre_draw(), but called automatically before rendering ALL polyhedrons (only in "space" mode).

void mepp_component_CGAL_Example_plugin::post_draw_all_scene(), same as post_draw(), but called automatically after rendering ALL polyhedrons (only in "space" mode).

---

void mepp_component_CGAL_Example_plugin::OnMouseLeftDown(QMouseEvent *event), method called after a mouse left click pressed when the "alt" key (on Windows and Mac OS X) or the "meta" key ie "the Windows key" (on Linux) is maintained.

void mepp_component_CGAL_Example_plugin::OnMouseLeftUp(QMouseEvent *event), method called after a mouse left click released when the "alt" key (on Windows and Mac OS X) or the "meta" key ie "the Windows key" (on Linux) is maintained.

---

void mepp_component_CGAL_Example_plugin::OnMouseRightDown(QMouseEvent *event), method called after a mouse right click pressed when the "alt" key (on Windows and Mac OS X) or the "meta" key ie "the Windows key" (on Linux) is maintained.

void mepp_component_CGAL_Example_plugin::OnMouseRightUp(QMouseEvent *event), method called after a mouse right click released when the "alt" key (on Windows and Mac OS X) or the "meta" key ie "the Windows key" (on Linux) is maintained.

---

void mepp_component_CGAL_Example_plugin::OnMouseMotion(QMouseEvent *event), method called after a mouse movement when the "alt" key (on Windows and Mac OS X) or the "meta" key ie "the Windows key" (on Linux) is maintained.

---

void mepp_component_CGAL_Example_plugin::OnMouseWheel(QWheelEvent *event), method called after a mouse wheel movement when the "alt" key (on Windows and Mac OS X) or the "meta" key ie "the Windows key" (on Linux) is maintained.

---

void mepp_component_CGAL_Example_plugin::OnKeyPress(QKeyEvent *event), method called after pressing a key when the "alt" key (on Windows and Mac OS X) or the "meta" key ie "the Windows key" (on Linux) is maintained.

void mepp_component_CGAL_Example_plugin::OnKeyRelease(QKeyEvent *event), method called after releasing a key when the "alt" key (on Windows and Mac OS X) or the "meta" key ie "the Windows key" (on Linux) is maintained.

3.2. Common methods to all components (see CGAL_Example_Component.cpp):

CGAL_Example_Component::CGAL_Example_Component(Viewer* v, PolyhedronPtr p), component constructor.

You can initialize your variables here but it is important to at least inform correctly those below (that is to say the name of your component and the initial state of the latter at its creation, see 3.3.1. below):

```
componentName = "CGAL_Example_Component";
init = 1;
```
?

3.3. CGAL_Example component specific methods

3.3.1. Step1: mesh triangulation and random color facets

In CGAL_Example_Component.h:

We will first declare our first step function. This function will be "public".
void TriangulateAndRandomColorFacets(PolyhedronPtr pMesh);

The body of this function will be present in CGAL_Example_Component.cpp:

```
void CGAL_Example_Component::TriangulateAndRandomColorFacets(PolyhedronPtr pMesh)
{
    pMesh->triangulate(); //(1)

    srand((unsigned)time(NULL));

    Facet_iterator pFacet = NULL; //(2)
    for (pFacet = pMesh->facets_begin(); pFacet != pMesh->facets_end(); pFacet++)
    {
        float rand255r = (float) rand() / (float) RAND_MAX;
        float rand255g = (float) rand() / (float) RAND_MAX;
        float rand255b = (float) rand() / (float) RAND_MAX;

        pFacet->color(rand255r, rand255g, rand255b);
    }
}
```
?

Explanations:

(1) The polyhedron is first triangulated in all cases. This operation has only an effect if the polyhedron is not already triangulated.
(2) We go through all facets with a facets iterator and each facet is assigned with random R, G, B components.

In mepp_component_CGAL_Example_plugin.cpp:

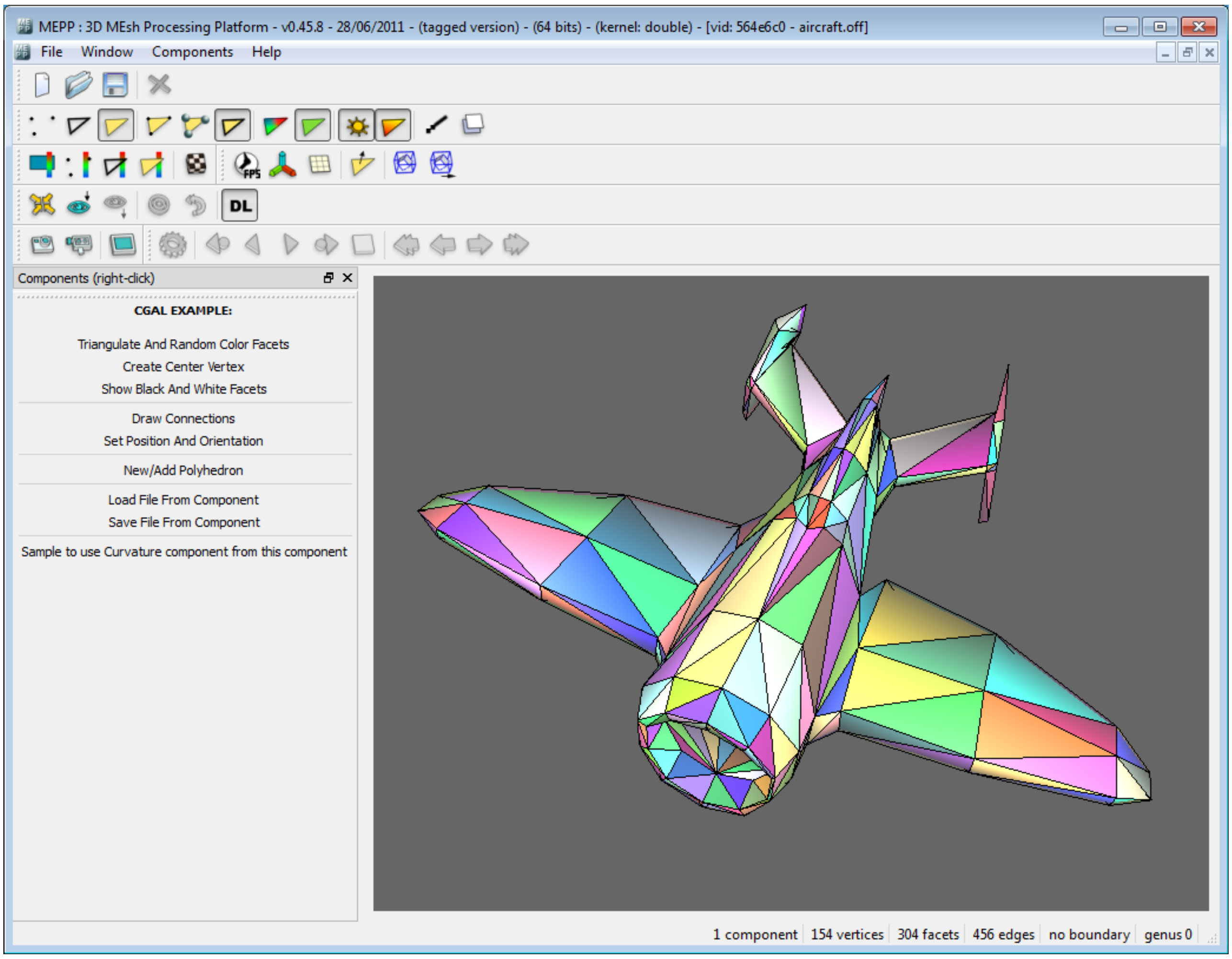
We called our first function above through **step1** function() which can be invoked by the component menu or toolbar.

```
void mepp_component_CGAL_Example_plugin::step1()
{
    QApplication::setOverrideCursor(Qt::WaitCursor); //(3)

    // active viewer
```
?

```
    if (mw->activeMdiChild() != 0)
    {
        Viewer* viewer = (Viewer *)mw->activeMdiChild();
        PolyhedronPtr polyhedron_ptr = viewer->getScenePtr()->get_polyhedron(); //(4)

        CGAL_Example_ComponentPtr component_ptr = findOrCreateComponentForViewer<CGAL_Example_ComponentPtr, CGAL_Examp
        component_ptr->TriangulateAndRandomColorFacets(polyhedron_ptr); //(6)

        component_ptr->set_init(2); //(7)
        viewer->recreateListsAndUpdateGL();
    }

    QApplication::restoreOverrideCursor(); //(8)
}
```

Explanations:

(3) Through a mouse cursor "hourglass", we tell the user that MEPP is currently occupied.
(4) We get a pointer to the polyhedron.
(5) We get a pointer to the component.
(6) We call our function, passing it our polyhedron pointer.
(7) We set the component to "state 2" (it was in "state 1" at its creation, see 3.2. above) and we refresh the display.
(8) We inform the user that processing is now done by repositioning the "classic" mouse cursor.


MEPP : 3D MEsh Processing Platform - v0.45.8 - 28/06/2011 - (tagged version) - (64 bits) - (kernel: double) - [vid: 564e6c0 - aircraft.off]

1 component | 154 vertices | 304 facets | 456 edges | no boundary | genus 0

3.3.2. Step2: creating a point in the central facet (and thus 3 sub-facets) whose color is near to a color chosen by the user

In CGAL_Example_Component.h:

We will first declare our second step function. This function will be "public".
void CreateCenterVertex(PolyhedronPtr pMesh, bool save);

The body of this function will be present in CGAL_Example_Component.cpp:

```
void CGAL_Example_Component::CreateCenterVertex(PolyhedronPtr pMesh)
{
    CSubdivider_sqrt3<Polyhedron,Enriched_kernel> subdivider;

    long dist, maxdist=0, mindist=ColourDistance(0, 0, 0, 255, 255, 255), mindist_pct;

    Facet_iterator pFacet = NULL; //(1)
    for (pFacet = pMesh->facets_begin(); pFacet != pMesh->facets_end(); pFacet++)
    {
        pFacet->tag(0);

        dist = ColourDistance((unsigned char)round(color(0)*255.), (unsigned char)round(color(1)*255.), (unsigned char)

        if (dist < mindist)
            mindist = dist;
        if (dist > maxdist)
            maxdist = dist;
    }

    int pct=3;
    mindist_pct=((maxdist-mindist)*pct/100)+mindist; //(2)

    int cpt = 1;
    for (pFacet = pMesh->facets_begin(); pFacet != pMesh->facets_end(); pFacet++)
    {
        if (pFacet->tag()==0)
        {
            dist = ColourDistance((unsigned char)round(color(0)*255.), (unsigned char)round(color(1)*255.), (unsigned c

            if ((dist >= mindist) && (dist <= mindist_pct))
            {
                pFacet->tag(1);
                Vertex_handle hVertex = subdivider.create_center_vertex(*pMesh, pFacet)->vertex(); //(3)

                float rand255r = (float) rand() / (float) RAND_MAX;
                float rand255g = (float) rand() / (float) RAND_MAX;
                float rand255b = (float) rand() / (float) RAND_MAX;
                hVertex->color(rand255r, rand255g, rand255b);
            }
        }
    }
}
```

Explanations:

(1) We go through all facets with a facets iterator to calculate "min" and "max" color distances from the color chosen by the user.
(2) We go through all facets with a facets iterator. For all facets not previously cut into 3 sub-facets (ie not tagged) and close in terms of color distance about 3% of the color chosen by the user, we create a central point and three sub-facets.
(3) The central point and the three sub-facets are created through an object of class CSubdivider_sqrt3.

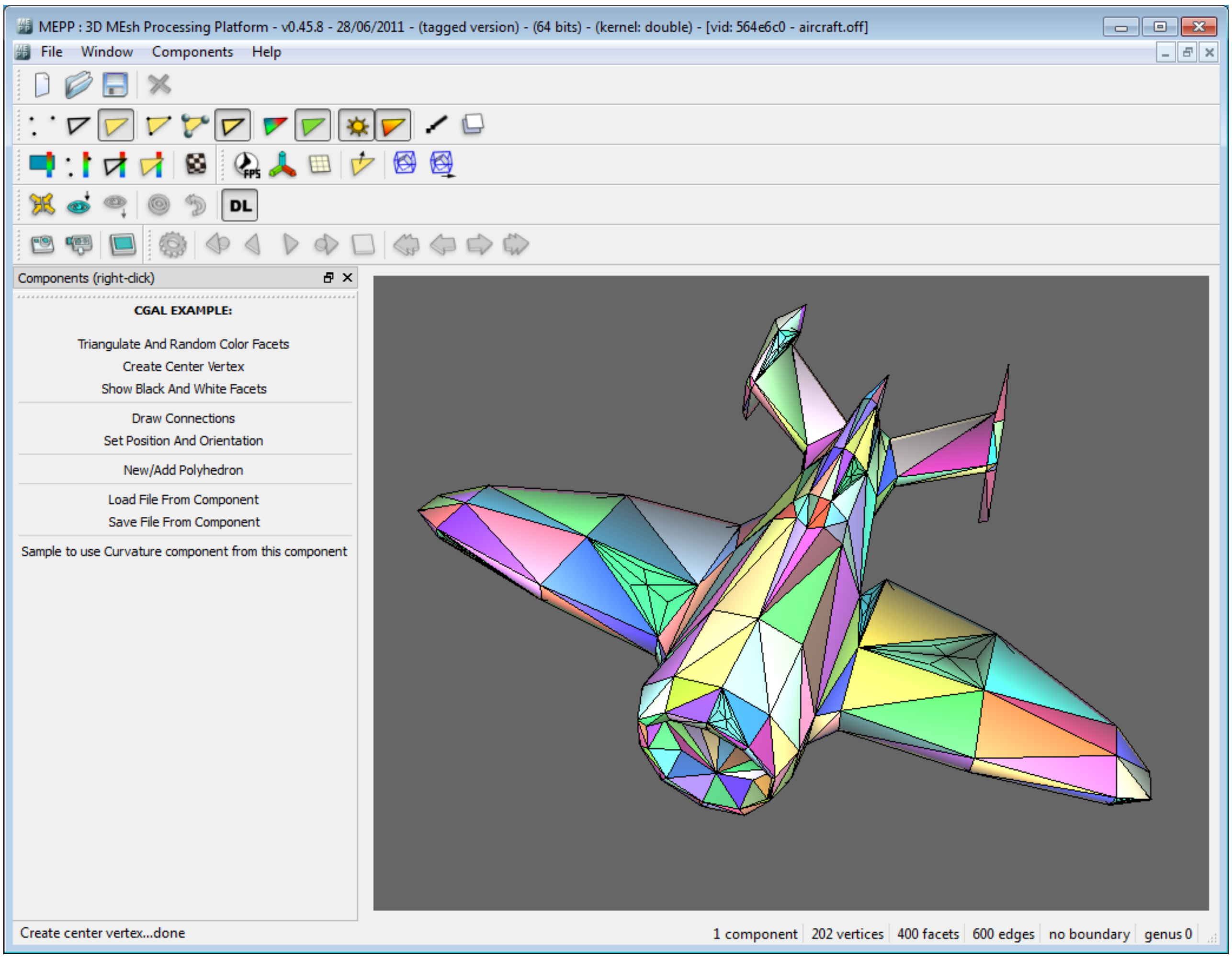
In mepp_component_CGAL_Example_plugin.cpp:

We called our second function above through **step2** function() which can be invoked by the component menu or toolbar.

```
void mepp_component_CGAL_Example_plugin::step2()
{
    // active viewer
    if (mw->activeMdiChild() != 0)
    {
        Viewer* viewer = (Viewer *)mw->activeMdiChild();
        PolyhedronPtr polyhedron_ptr = viewer->getScenePtr()->get_polyhedron(); //(4)

        CGAL_Example_ComponentPtr component_ptr = findOrCreateComponentForViewer<CGAL_Example_ComponentPtr, CGAL_Exampl

        QColor current_color(int(component_ptr->color(0)*255.), int(component_ptr->color(1)*255.), int(component_ptr->c
        QColor new_color = QColorDialog::getColor(current_color, viewer); //(6)

        if (new_color.isValid())
        {
            component_ptr->color(float(new_color.red())/255., float(new_color.green())/255., float(new_color.blue())/25

            SettingsDialog_CGAL_Example dial;
            if (dial.exec() == QDialog::Accepted) //(7)
            {
                QApplication::setOverrideCursor(Qt::WaitCursor);

                int iteration = dial.Iteration->value();

                mw->statusBar()->showMessage(tr("Create center vertex...")); //(8)

                for (int p=0; p<viewer->getScenePtr()->get_nb_polyhedrons(); p++) //(9)
                    for (int i=0; i<iteration; i++)
                        component_ptr->CreateCenterVertex(viewer->getScenePtr()->get_polyhedron(p), false);

                mw->statusBar()->showMessage(tr("Create center vertex...done"));

                viewer->recreateListsAndUpdateGL();

                QApplication::restoreOverrideCursor();
                return;
            }
        }

        mw->statusBar()->showMessage(tr("Create center vertex...canceled"));
    }

    QApplication::restoreOverrideCursor();
}
```

Explanations:

(4) We get a pointer to the polyhedron.
(5) We get a pointer to the component.
(6) We call the Qt color selector dialog and get the chosen color.
(7) We call our dialog box created using Qt Designer (see 4. below) in order to know how many times the user wants to create central points (and thus sub-facets): iteration.
(8) We inform the MEPP status bar to mean that a treatment is being.
(9) For all polyhedrons, we call our function (iteration times), passing it our polyhedron pointer.



MEPP : 3D MEsh Processing Platform - v0.45.8 - 28/06/2011 - (tagged version) - (64 bits) - (kernel: double) - [vid: 564e6c0 - aircraft.off]

Create center vertex...done | 1 component | 202 vertices | 400 facets | 600 edges | no boundary | genus 0

3.3.3. Step3: for all facets cut in step2, look into all neighboring facets having a common half-edge and allocation of a white or black color to these according to a given criterion

In CGAL_Example_Component.h:

We will first declare our third step function. This function will be "public".
void ShowBlackAndWhiteFacets(PolyhedronPtr pMesh);

The body of this function will be present in CGAL_Example_Component.cpp:

```cpp
void CGAL_Example_Component::ShowBlackAndWhiteFacets(PolyhedronPtr pMesh)
{
    Facet_iterator pFacet = NULL; //(1)
    for (pFacet = pMesh->facets_begin(); pFacet != pMesh->facets_end(); pFacet++)
    {
        int all_tags_are_one;

        if (pFacet->tag()==1)
        {
            // circulate around facet
            Halfedge_around_facet_circulator he, end;
            he = end = pFacet->facet_begin();
            all_tags_are_one=1;

            CGAL_For_all(he, end) //(2)
            {
                if (he->opposite()->face() != NULL)
                {
                    if (he->opposite()->face()->tag()==0) //(3)
                    {
                        he->opposite()->face()->color(1., 1., 1.);
                        all_tags_are_one=0;
                    }
                }
            }
            if (all_tags_are_one) //(4)
                pFacet->color(0., 0., 0.);
        }
    }
}
```

Explanations:

(1) We go through all facets with a facets iterator.
(2) For all facets cut in step2 (and hence tagged), we use a half-edges circulator and for all these half-edges,
(3) we color the incident facet to the opposite half-edge (if it exists!) in white if it's not a tagged facet, otherwise,
(4) we color the incident facet to the opposite half-edge (if it exists!) in black if all neighboring facets are tagged.

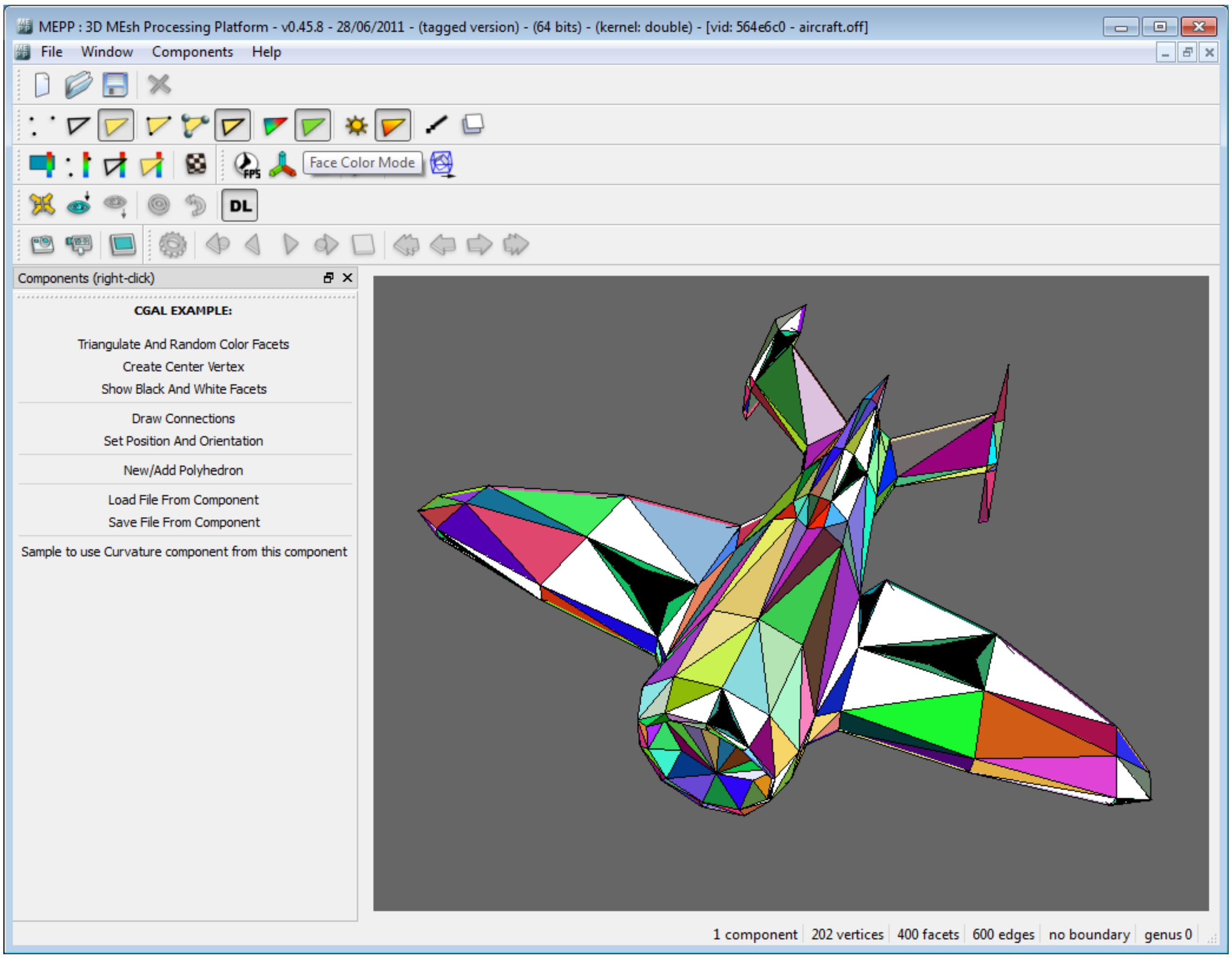
In mepp_component_CGAL_Example_plugin.cpp:

We called our third function above through **step3** function() which can be invoked by the component menu or toolbar.

```
void mepp_component_CGAL_Example_plugin::step3()
{
    QApplication::setOverrideCursor(Qt::WaitCursor);

    // active viewer
    if (mw->activeMdiChild() != 0)
    {
        Viewer* viewer = (Viewer *)mw->activeMdiChild();
        PolyhedronPtr polyhedron_ptr = viewer->getScenePtr()->get_polyhedron();

        CGAL_Example_ComponentPtr component_ptr = findOrCreateComponentForViewer<CGAL_Example_ComponentPtr, CGAL_Examp
        component_ptr->ShowBlackAndWhiteFacets(polyhedron_ptr); //(5)

        viewer->recreateListsAndUpdateGL();
    }

    QApplication::restoreOverrideCursor();
}
```

Explanations:

(5) We call our function, passing it our polyhedron pointer.


MEPP : 3D MEsh Processing Platform - v0.45.8 - 28/06/2011 - (tagged version) - (64 bits) - (kernel: double) - [vid: 564e6c0 - aircraft.off]

1 component | 202 vertices | 400 facets | 600 edges | no boundary | genus 0

3.3.4. Step4: display "connections" between various polyhedrons (only in "space" mode)

In mepp_component_CGAL_Example_plugin.cpp:

We called our fourth function above through **step4** function() which can be invoked by the component menu or toolbar.

```cpp
void mepp_component_CGAL_Example_plugin::step4()
{
    QApplication::setOverrideCursor(Qt::WaitCursor);

    // active viewer
    if (mw->activeMdiChild() != 0)
    {
        Viewer* viewer = (Viewer *)mw->activeMdiChild();
        PolyhedronPtr polyhedron_ptr = viewer->getScenePtr()->get_polyhedron();

        CGAL_Example_ComponentPtr component_ptr = findOrCreateComponentForViewer<CGAL_Example_ComponentPtr, CGAL_Exampl

        component_ptr->set_init(3); //(1)
        viewer->updateGL();
    }

    QApplication::restoreOverrideCursor();
}
```

Explanations:

(1) We set the "init" flag to 3 (at the creation of the component, the "init" flag was initialized to 1).

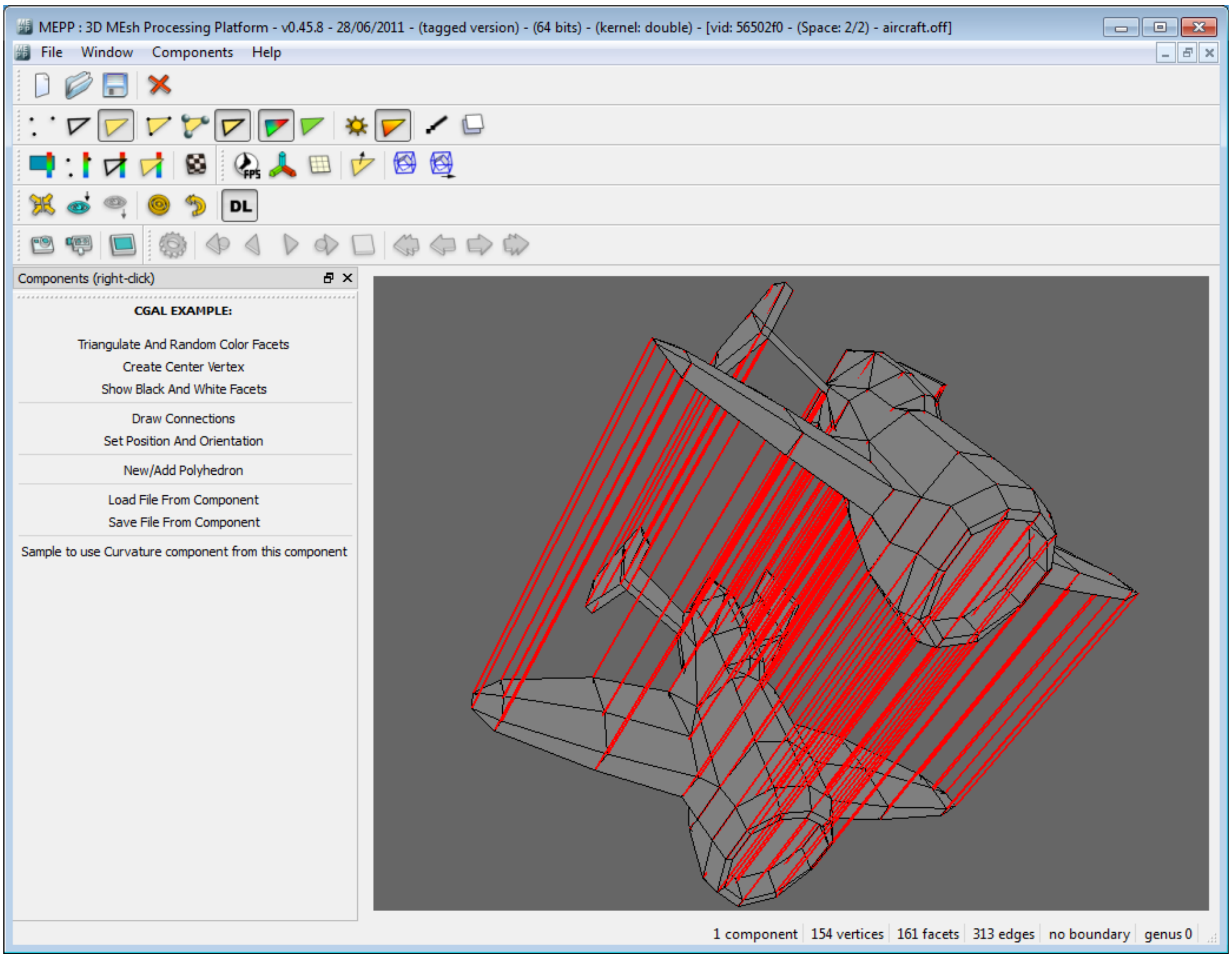
Note : the flag positioned to 2 serves for another function.

Then the **void mepp_component_CGAL_Example_plugin::post_draw_all_scene()** function (see 3.1.) automatically called after rendering ALL polyhedrons (only in "space" mode) tests the "init" flag and if its value is 3 performs then the display of "connections" (see below).

```cpp
void mepp_component_CGAL_Example_plugin::post_draw_all_scene()
{
    Viewer* viewer = (Viewer *)mw->activeMdiChild();
    PolyhedronPtr polyhedron_ptr = viewer->getScenePtr()->get_polyhedron();

    if (doesExistComponentForViewer<CGAL_Example_ComponentPtr, CGAL_Example_Component>(viewer, polyhedron_ptr)) // impo
    {
        CGAL_Example_ComponentPtr component_ptr = findOrCreateComponentForViewer<CGAL_Example_ComponentPtr, CGAL_Exampl
        if (component_ptr->get_init() == 3) //(2)
        {
            int nbMesh = qMin(viewer->getScenePtr()->get_nb_polyhedrons(), viewer->get_nb_frames());
            if (nbMesh == 2)
            {
                glPushMatrix();
                    draw_connections(viewer, 0, 1); // link between first and second mesh (first and second frame) //(3
                glPopMatrix();
            }
        }
    }
}
```

Explanations:

(2) We test the value of the "init" flag.
(3) We draw connections.


MEPP : 3D MEsh Processing Platform - v0.45.8 - 28/06/2011 - (tagged version) - (64 bits) - (kernel: double) - [vid: 56502f0 - (Space: 2/2) - aircraft.off]
File
Window
Components
Help
DL
Components (right-click)
CGAL EXAMPLE:
Triangulate And Random Color Facets
Create Center Vertex
Show Black And White Facets
Draw Connections
Set Position And Orientation
New/Add Polyhedron
Load File From Component
Save File From Component
Sample to use Curvature component from this component
1 component
154 vertices
161 facets
313 edges
no boundary
genus 0

3.3.5. Step5: fix a position and orientation for the frame of the current polyhedron (only in "space" mode)

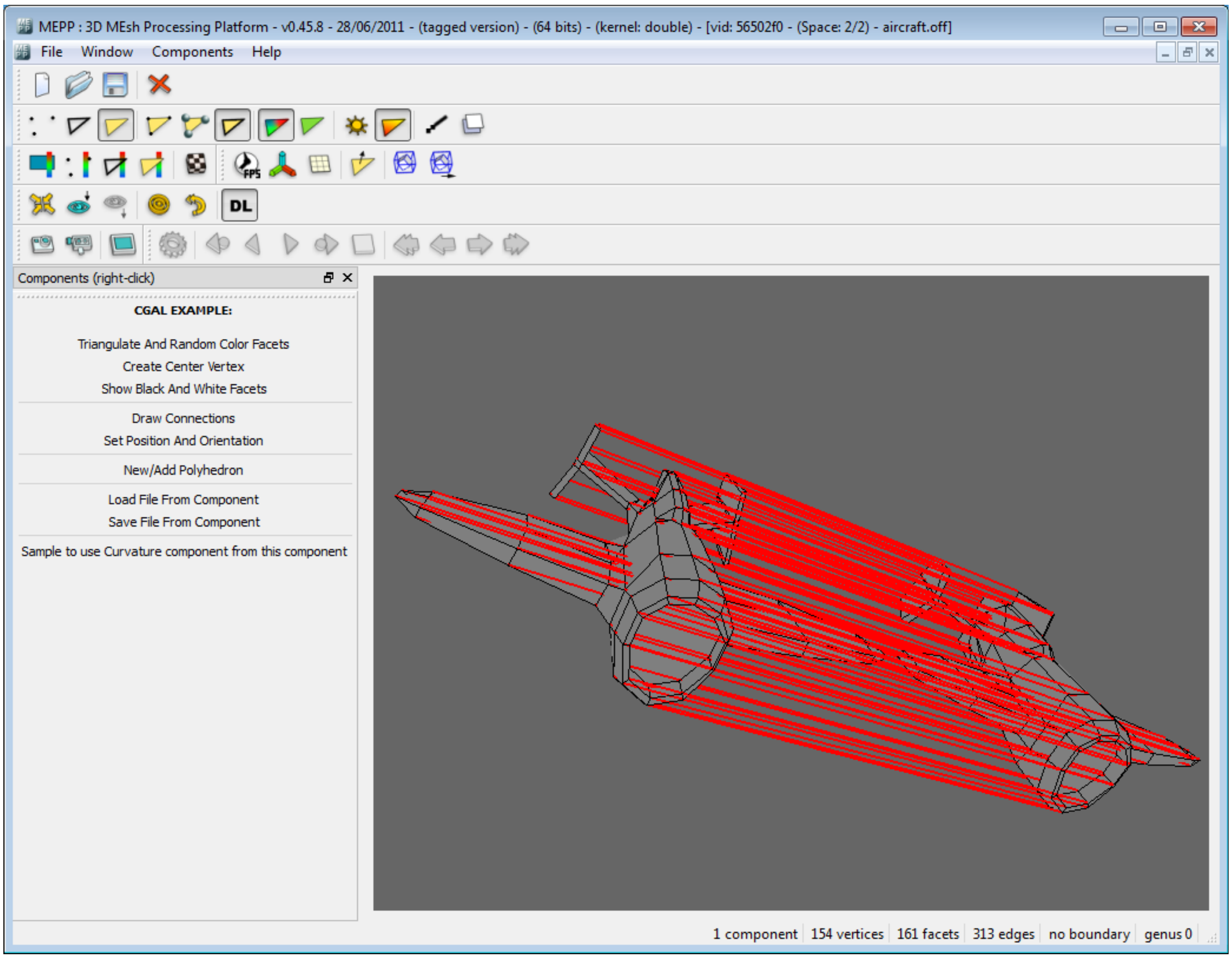
In mepp_component_CGAL_Example_plugin.cpp:

We called our fifth function above through **step5** function() which can be invoked by the component menu or toolbar.

```
void mepp_component_CGAL_Example_plugin::step5()
{
    QApplication::setOverrideCursor(Qt::WaitCursor);

    // active viewer
    if (mw->activeMdiChild() != 0)
    {
        Viewer* viewer = (Viewer *)mw->activeMdiChild();
        PolyhedronPtr polyhedron_ptr = viewer->getScenePtr()->get_polyhedron();

        CGAL_Example_ComponentPtr component_ptr = findOrCreateComponentForViewer<CGAL_Example_ComponentPtr, CGAL_Examp

        Vec pw(2, 0, 0); viewer->frame(viewer->getScenePtr()->get_current_polyhedron())->setPosition(pw); // position i

        Quaternion qw(0, 0, 0, 1); // identity quaternion (i.e., no rotation)
        viewer->frame(viewer->getScenePtr()->get_current_polyhedron())->setOrientation(qw); // rotation in world coord

        viewer->updateGL(); //(3)
    }

    QApplication::restoreOverrideCursor();
}
```

Explanations:

(1) We set the frame of the current polyhedron to (2, 0, 0) in the world coordinate system.
(2) We turn the frame of the current polyhedron without any rotation on all axes in the world coordinate system.
(3) We refresh the display.


MEPP : 3D MEsh Processing Platform - v0.45.8 - 28/06/2011 - (tagged version) - (64 bits) - (kernel: double) - [vid: 56502f0 - (Space: 2/2) - aircraft.off]
File
Window
Components
Help
DL
Components (right-click)
CGAL EXAMPLE:
Triangulate And Random Color Facets
Create Center Vertex
Show Black And White Facets
Draw Connections
Set Position And Orientation
New/Add Polyhedron
Load File From Component
Save File From Component
Sample to use Curvature component from this component
1 component
154 vertices
161 facets
313 edges
no boundary
genus 0

3.3.6. Step6: create a new child window with an example polyhedron or add an example polyhedron to the current child window

In mepp_component_CGAL_Example_plugin.cpp:

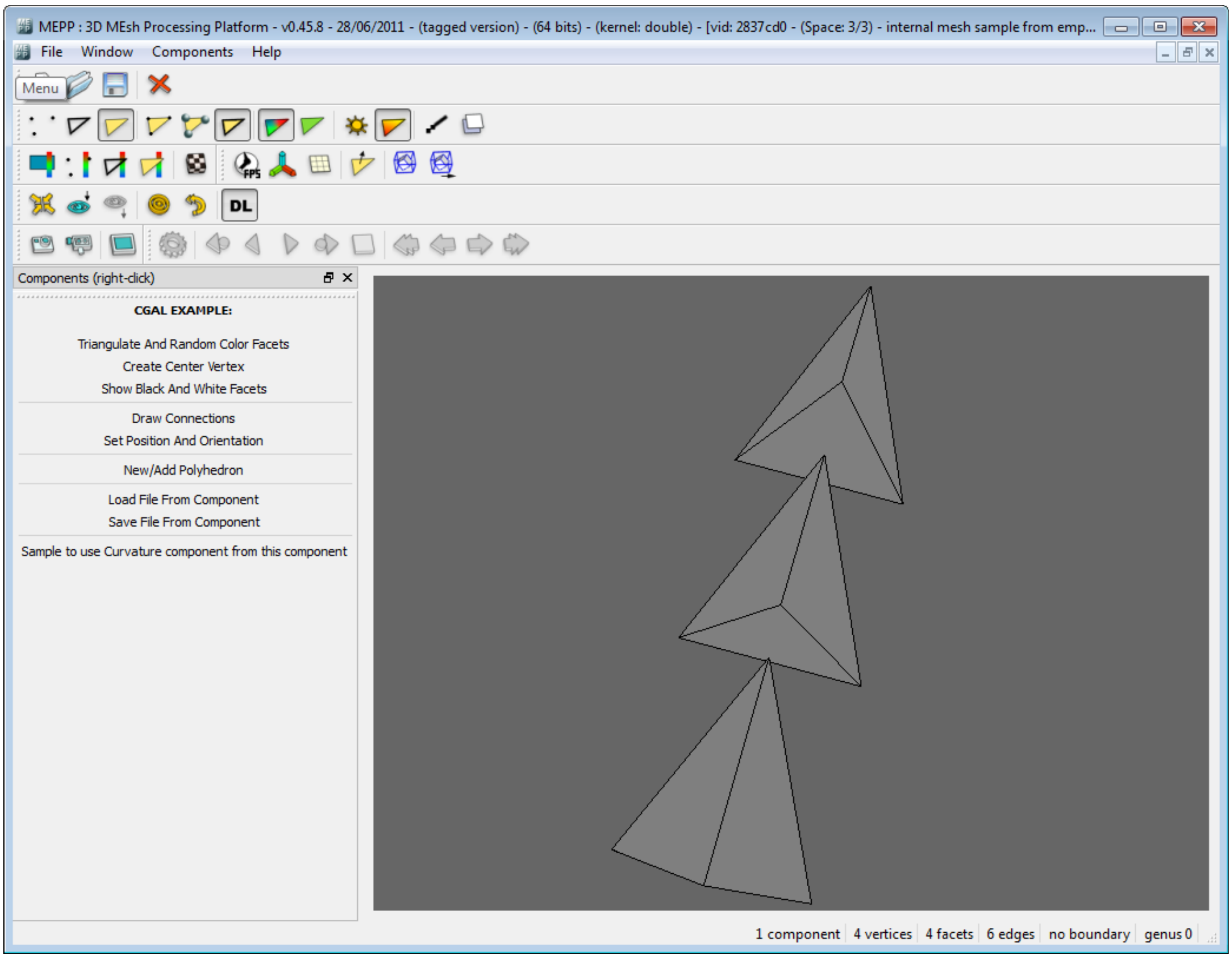
We called our sixth function above through **step6** function() which can be invoked by the component menu or toolbar.

```
void mepp_component_CGAL_Example_plugin::step6()
{
    QApplication::setOverrideCursor(Qt::WaitCursor);

    // active viewer
    if (mw->activeMdiChild() != 0) //(1)
    {
        Viewer* viewer = (Viewer *)mw->activeMdiChild();
        PolyhedronPtr polyhedron_ptr = viewer->getScenePtr()->get_polyhedron();

        CGAL_Example_ComponentPtr component_ptr = findOrCreateComponentForViewer<CGAL_Example_ComponentPtr, CGAL_Examp
        // step 6a : begin
        emit(mw->get_actionAddEmpty()->trigger()); //(2)

        int nb_polyhedrons = viewer->getScenePtr()->get_nb_polyhedrons(); //(3)
        polyhedron_ptr = viewer->getScenePtr()->get_polyhedron(nb_polyhedrons-1);
        component_ptr->CreateTetrahedron(polyhedron_ptr);

        viewer->getScenePtr()->setcurrentFile(tr("internal mesh sample from empty")); //(4)
        viewer->setDynTitle();

        viewer->recreateListsAndUpdateGL(); //(5)
        // step 6a : end
    }
    else
    {
        // step 6b : begin
        emit(mw->get_actionNewEmpty()->trigger()); //(6)

        for (int i=0; i<lwindow.size(); i++) // all viewers
        {
            Viewer* viewer = (Viewer *)qobject_cast<QWidget *>(lwindow[i]->widget());
            if (viewer->getScenePtr()->get_polyhedron()->empty()) //(7)
            {
                PolyhedronPtr polyhedron_ptr = viewer->getScenePtr()->get_polyhedron();

                CGAL_Example_ComponentPtr component_ptr = findOrCreateComponentForViewer<CGAL_Example_ComponentPtr, CG
                component_ptr->CreateTetrahedron(polyhedron_ptr); //(8)

                viewer->showAllScene(); //(9)

                viewer->getScenePtr()->setcurrentFile(tr("internal mesh sample from empty")); //(10)
                viewer->setDynTitle();
            }
        }
        // step 6b : end
    }

    QApplication::restoreOverrideCursor();
}
```

Explanations:

(1) If there is an open and active child window in MEPP:
(2) we emit a signal to add an empty polyhedron in this child window,
(3) we get its pointer and create the example polyhedron: a tetrahedron,
(4) we give it a name and we refresh the child window title,
(5) we refresh the display.

If there is no open and active child window in MEPP:
(6) we emit a signal to create a new child window,
(7) then we go through all the MEPP child windows and we look for the empty child window newly created,
(8) we then retrieve the pointer of the empty polyhedron and create the example polyhedron: a tetrahedron,
(9) we center the scene to see the newly created polyhedron in its entirety,
(10) we give it a name and we refresh the child window title.


MEPP : 3D MEsh Processing Platform - v0.45.8 - 28/06/2011 - (tagged version) - (64 bits) - (kernel: double) - [vid: 2837cd0 - (Space: 3/3) - internal mesh sample from emp...

1 component | 4 vertices | 4 facets | 6 edges | no boundary | genus 0

3.3.7. Step7/8: load/save a polyhedron in its component (eg file format specific to the component)

In mepp_component_CGAL_Example_plugin.cpp:

We called our seventh function above through **step7** function() which can be invoked by the component menu or toolbar.

```
void mepp_component_CGAL_Example_plugin::step7()
{
    emit(mw->get_mainwindowActionOpen()->doSendParamsOpen(tr("Open Mesh File(s) - from CGAL_Example"), tr("OFF files (*.
}
```

Explanations:

(1) We send a signal to open a dialog with the parameters **tr("Open Mesh File(s) - from CGAL_Example"), tr("OFF files (*.off)"), Normal** and then invoke the **load_file_from_component** component-specific function (see below).

```
int mepp_component_plugin_interface::load_file_from_component(PolyhedronPtr polyhedron_ptr, QString filename, Viewer*
{
    mepp_component_CGAL_Example_plugin *mepp_component_plugin = NULL;
    for (int i=0; i<viewer->lplugin.size(); ++i)
    {
        if (dynamic_cast<mepp_component_CGAL_Example_plugin*>(viewer->lplugin[i]) != 0)
        {
            mepp_component_plugin = dynamic_cast<mepp_component_CGAL_Example_plugin*>(viewer->lplugin[i]); //(2)
        }
    }

    int res;

    if (mepp_component_plugin)
    {
        CGAL_Example_ComponentPtr component_ptr = mepp_component_plugin->findOrCreateComponentForViewer<CGAL_Example_Co

        res = polyhedron_ptr->load_mesh_off(filename.toStdString()); //(4)
    }
    else
        res=-1;

    return res;
}
```

Explanations:

(2) We get a pointer to the plugin component.
(3) We get a pointer to the component.
(4) We call a function to load the polyhedron. Here, in this example, it is the default ".off" files loading but it may be a specific format (see Compression_Valence component and P3D format).

Note : the principle is the same to save a polyhedron in its component (but instead we use the **save_file_from_component** component-specific function).

3.3.8. Step9: use another component within its component

In mepp_component_CGAL_Example_plugin.cpp:

We called our ninth function above through **step9** function() which can be invoked by the component menu or toolbar.

```
void mepp_component_CGAL_Example_plugin::step9()
{
    // active viewer
    if (mw->activeMdiChild() != 0)
    {
        Viewer* viewer = (Viewer *)mw->activeMdiChild();
        PolyhedronPtr polyhedron_ptr = viewer->getScenePtr()->get_polyhedron();

        QApplication::setOverrideCursor(Qt::WaitCursor);

        // we use CGAL_Example component here (as usual) //(1)
            CGAL_Example_ComponentPtr component_ptr = findOrCreateComponentForViewer<CGAL_Example_ComponentPtr, CGAL_E
            component_ptr->TriangulateAndRandomColorFacets(polyhedron_ptr);

            component_ptr->set_init(2);
        // we use CGAL_Example component here (as usual)

        // we use Curvature component here //(2)
            bool IsGeo;
            double radius;
            Curvature_ComponentPtr component_ptr_curvature = findOrCreateComponentForViewer<Curvature_ComponentPtr, Cur

            // params
            radius = 0.001;
            IsGeo=true;

            mw->statusBar()->showMessage(tr("Curvature..."));
            component_ptr_curvature->principal_curvature(polyhedron_ptr,IsGeo,radius);
            mw->statusBar()->showMessage(tr("Curvature is done"));

            component_ptr_curvature->set_init(2);

            component_ptr_curvature->ConstructColorMap(polyhedron_ptr,1);
            viewer->recreateListsAndUpdateGL();
        // we use Curvature component here
    }

    QApplication::restoreOverrideCursor();
}
```

Explanations:

(1) The component is used as usual.
(2) We use another component using another pointer to the component in question (but prior to, see 1.2.**d)** and see 1.4.).

3.3.9. Keyboard / mouse event handling, user interaction

In CGAL_Example_Component.h:

We will first declare our function. This function will be "public".
void GetClickedVertices(PolyhedronPtr pMesh, double x, double y, int tolerance);

The body of this function will be present in CGAL_Example_Component.cpp:

```
void CGAL_Example_Component::GetClickedVertices(PolyhedronPtr pMesh, double x, double y, int tolerance)
{
    GLdouble *model ;  GLdouble *proj ;  GLint *view;

    view=new int[4096];
    model=new double[4096];
    proj=new double[4096];

    glGetIntegerv (GL_VIEWPORT, view); //(1)
    glGetDoublev (GL_MODELVIEW_MATRIX, model);
    glGetDoublev (GL_PROJECTION_MATRIX, proj);

    y=view[3]-y;

    GLdouble wx ; GLdouble wy ; GLdouble wz;

    int vertexID=0;
    for (Vertex_iterator pVertex = pMesh->vertices_begin(); pVertex!= pMesh->vertices_end(); pVertex++)
    {
        gluProject (pVertex->point().x(),pVertex->point().y(),pVertex->point().z(), model, proj, view, &wx, &wy, &wz);
        if (wz>0. && wz<1)
        if (x>floor(wx)-tolerance && x<floor(wx)+tolerance)
        if (y>floor(wy)-tolerance && y<floor(wy)+tolerance)  // we set a small tolerance (2 * 5 pixels for example) for
        {
            pVertex->color(1., 0., 0.); //(4)
        }
        vertexID++;
    }

    delete[]view;
    delete[]model;
    delete[]proj;
}
```

Explanations:

(1) We get the three OpenGL matrices (GL_VIEWPORT, GL_MODELVIEW_MATRIX, GL_PROJECTION_MATRIX).
(2) For all vertices, we project its coordinates in the "window space",
(3) we fix a small tolerance to ease the "vertex clicking",
(4) we color the vertex (once identified) in red.

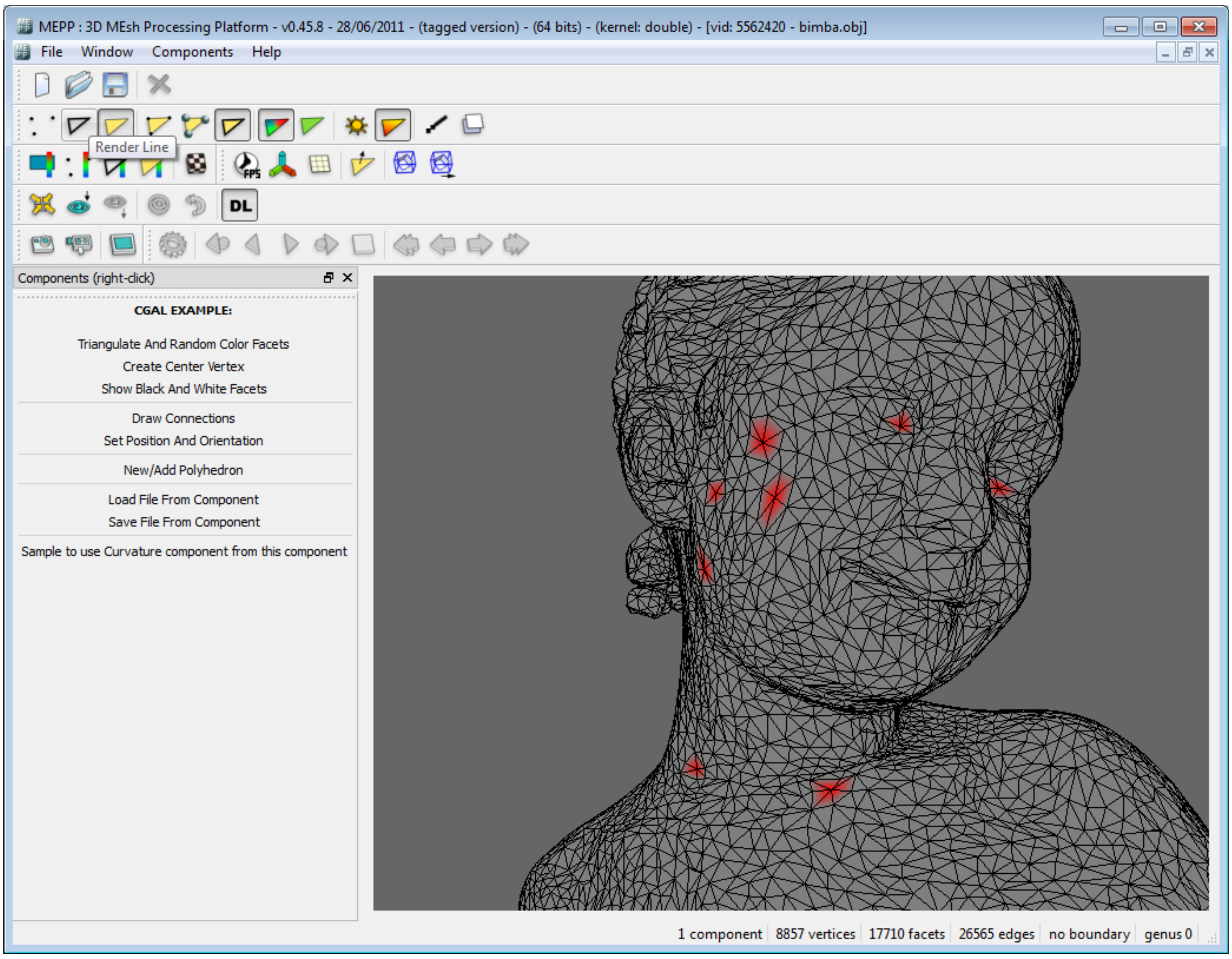
In mepp_component_CGAL_Example_plugin.cpp:

The function is called through the **OnMouseLeftDown(QMouseEvent *event)** event.

```
void mepp_component_CGAL_Example_plugin::OnMouseLeftDown(QMouseEvent *event)
{
    if (mw->activeMdiChild() != 0)
    {
        Viewer* viewer = (Viewer *)mw->activeMdiChild();
        PolyhedronPtr polyhedron_ptr = viewer->getScenePtr()->get_polyhedron();

        if (doesExistComponentForViewer<CGAL_Example_ComponentPtr, CGAL_Example_Component>(viewer, polyhedron_ptr)) //
        {
            mw->statusBar()->showMessage(tr("mepp_component_CGAL_Example_plugin: OnMouseLeftDown"), 1000); //(6)

            CGAL_Example_ComponentPtr component_ptr = findOrCreateComponentForViewer<CGAL_Example_ComponentPtr, CGAL_E

            if (viewer->getScenePtr()->get_loadType() == Normal)
            {
                component_ptr->GetClickedVertices(viewer->getScenePtr()->get_polyhedron(), event->x(), event->y(), 10)
                viewer->recreateListsAndUpdateGL();
            }
        }
    }
}
```

Explanations:

(5) We check first (through the **doesExistComponentForViewer** function) that this component (CGAL_Example) has already been instantiated within this child window, ie it has already been invoked and used.
Indeed, if this was not the case, the event would be to ignore.
This important test allows to send the event only to "active" components, otherwise all components (even "inactive") would receive the event.
(6) We inform the MEPP status bar to signify that a click has just happened and left this message displayed only for 1 second.
(7) We call our function, passing it our polyhedron pointer and the x, y "window space" coordinates. Here we fix a tolerance of 10 pixels in x and y for the accuracy of the click.


MEPP : 3D MEsh Processing Platform - v0.45.8 - 28/06/2011 - (tagged version) - (64 bits) - (kernel: double) - [vid: 5562420 - bimba.obj]
File
Window
Components
Help
Render Line
DL
Components (right-click)
CGAL EXAMPLE:
Triangulate And Random Color Facets
Create Center Vertex
Show Black And White Facets
Draw Connections
Set Position And Orientation
New/Add Polyhedron
Load File From Component
Save File From Component
Sample to use Curvature component from this component
1 component
8857 vertices
17710 facets
26565 edges
no boundary
genus 0

4. Addition of dialog boxes at the interface of a component

4.1. Creating a dialog (with Qt Designer)

a) Locate Qt Designer:
- on Windows, this tool is in **C:\dev\qt-x.x.x\bin** : designer.exe,
- on Linux, this tool is in the "**Applications > Programmation > Qt 4 Designer**" menu,
- on Mac, this tool is via "**Finder > Applications > MacPorts > Qt4 > Designer**".

b) Open the ".ui" file of your component to place your widgets as you want to be, you can read an interesting part of a Qt Designer tutorial here:
http://www.siteduzero.com/tutoriel-3-11360-modeliser-ses-fenetres-avec-qt-designer.html#ss_part_2

This one has indeed particularly important concepts like "layouts", "spacers" and also the window edition widgets properties.

You can also open ".ui" files of the following components to inspire you:
- Canonical component dialSettings.ui
- VSA component dialSettings.ui
- Compression_Valence component dialSettings*.ui

During this step it will be especially IMPORTANT TO NAME your widgets PROPERLY using the object inspector present at the top of the right column of Qt Designer.

c) Save your dialog.

4.2. Invoking your dialog and retrieve the values entered/indicated in the code of your component:

In \src\components\**Your_Category****Your_Component**\src\mepp_component_**Your_Component**_plugin.cpp use code like this:

```
SettingsDialog_Votre_Composant dial;
if (dial.exec() == QDialog::Accepted)
{
    int iteration = dial.Iteration->value(); // 'Iteration' is a widget name (see above step)
    ...
```

4.3. Note about the use of multiple dialog boxes in your component

a) Duplicate the 3 following files:
- dialSettings_**Your_Component**.ui
- dialSettings_**Your_Component**.cpp
- dialSettings_**Your_Component**.hxx

to

- dialSettings**Function_Your_Component**.ui
- dialSettings**Function_Your_Component**.cpp
- dialSettings**Function_Your_Component**.hxx

(**Function** is something here relating to this new dialog box.)

b) Open dialSettings**Function_Your_Component**.ui with Qt Designer to rename the new dialog box using the Object Inspector
(here the QDialog "objectName" property becomes "SettingsFonction" instead of "Settings".)

c) Save your dialog.

d) Find and replace respecting the "case":
- HEADER_MEPP_COMPONENT_**YOUR_COMPONENT**_PLUGIN_SETTINGS_H by HEADER_MEPP_COMPONENT_**YOUR_COMPONENT**_PLUGIN_SETTINGS_**FUNCTION**_H in dialSettings**Function_Your_Component**.hxx
- ui_dialSettings_**Your_Component**.h by ui_dialSettings**Function_Your_Component**.h in dialSettings**Function_Your_Component**.hxx
- Ui_Settings by Ui_Settings**Function** in dialSettings**Function_Your_Component**.hxx

- dialSettings_**Your_Component**.hxx by dialSettings**Function_Your_Component**.hxx in dialSettings**Function_Your_Component**.cpp

- SettingsDialog_**Your_Component** by SettingsDialog**Function_Your_Component** in dialSettings**Function_Your_Component**.cpp/(.hxx) files

e) In:
\src\components\**Your_Category****Your_Component**\src\mepp_component_**Your_Component**_plugin.cpp, in line 11 add the following line:

#include "dialSettings**Function_Your_Component**.hxx"

then use code similar to this one to retrieve the values entered/indicated in the code of your component:

```
SettingsDialogFunction_Your_Component dialFunction;
if (dialFunction.exec() == QDialog::Accepted)
{
    int iteration = dialFunction.Iteration->value();
    ...
```

Last update : April 13, 2012