an open source geometry processing and rendering framework

Jan Möbius, Leif Kobbelt

Overview

- Motivation
- Design Goals
- User Interface
- Plugins
- Writing a Plugin
- Demo
- System Requirements

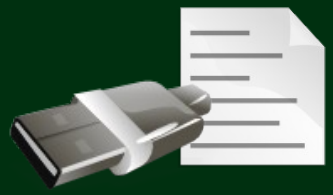
Motivation

- Reduce coding redundancy
  (viewers, datastructures, ...)
- Useable by everybody
  (research code vs. application)
- Exchange functionality without
  exchanging source code

Design Goals

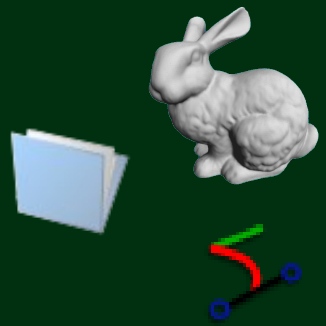
Platform independent

Extensible (→ Plugin based system, Scripting)



Flexibel Interface

Different Data Types (Surfaces / Volumes / ...)

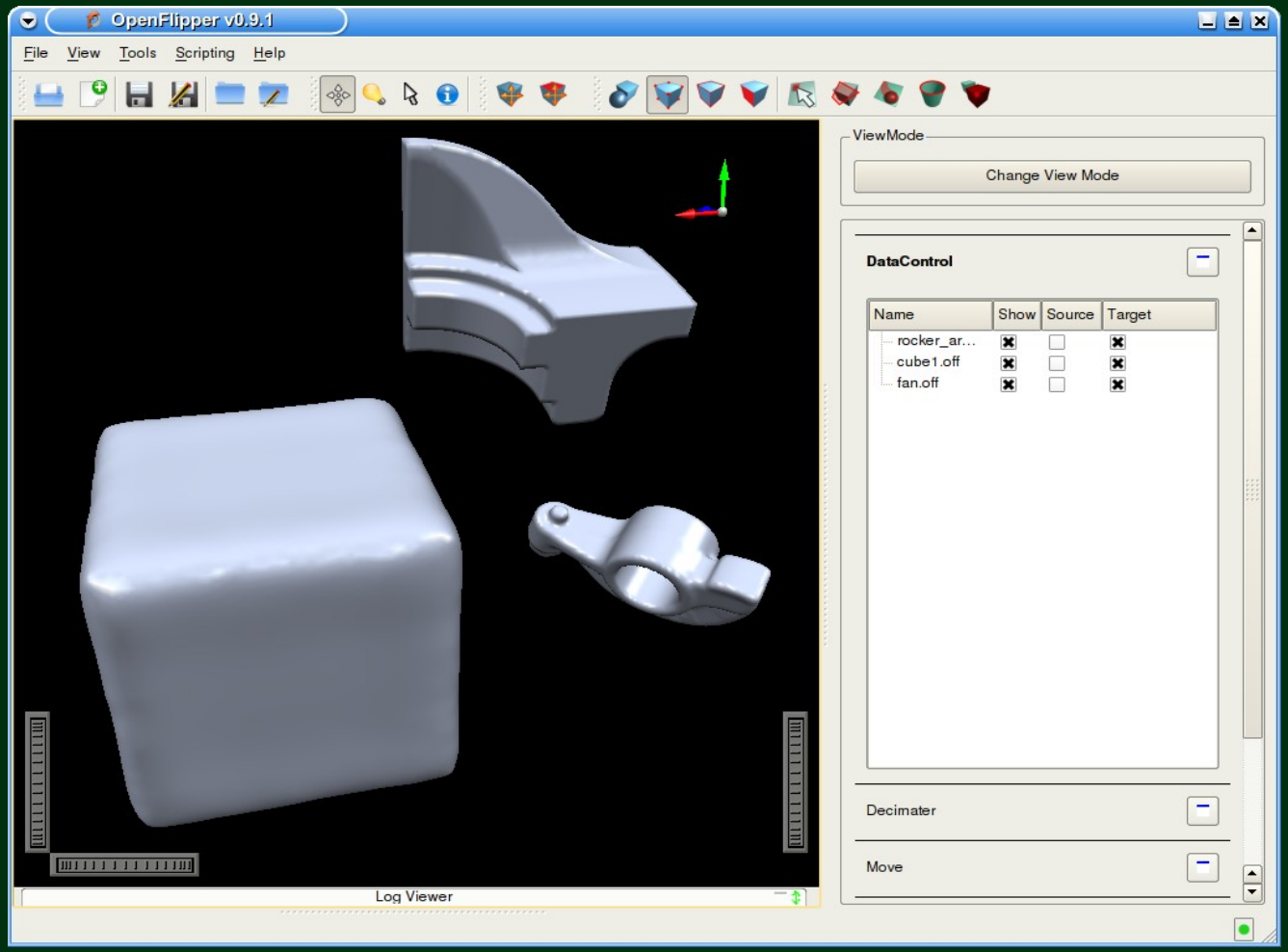


Major Features

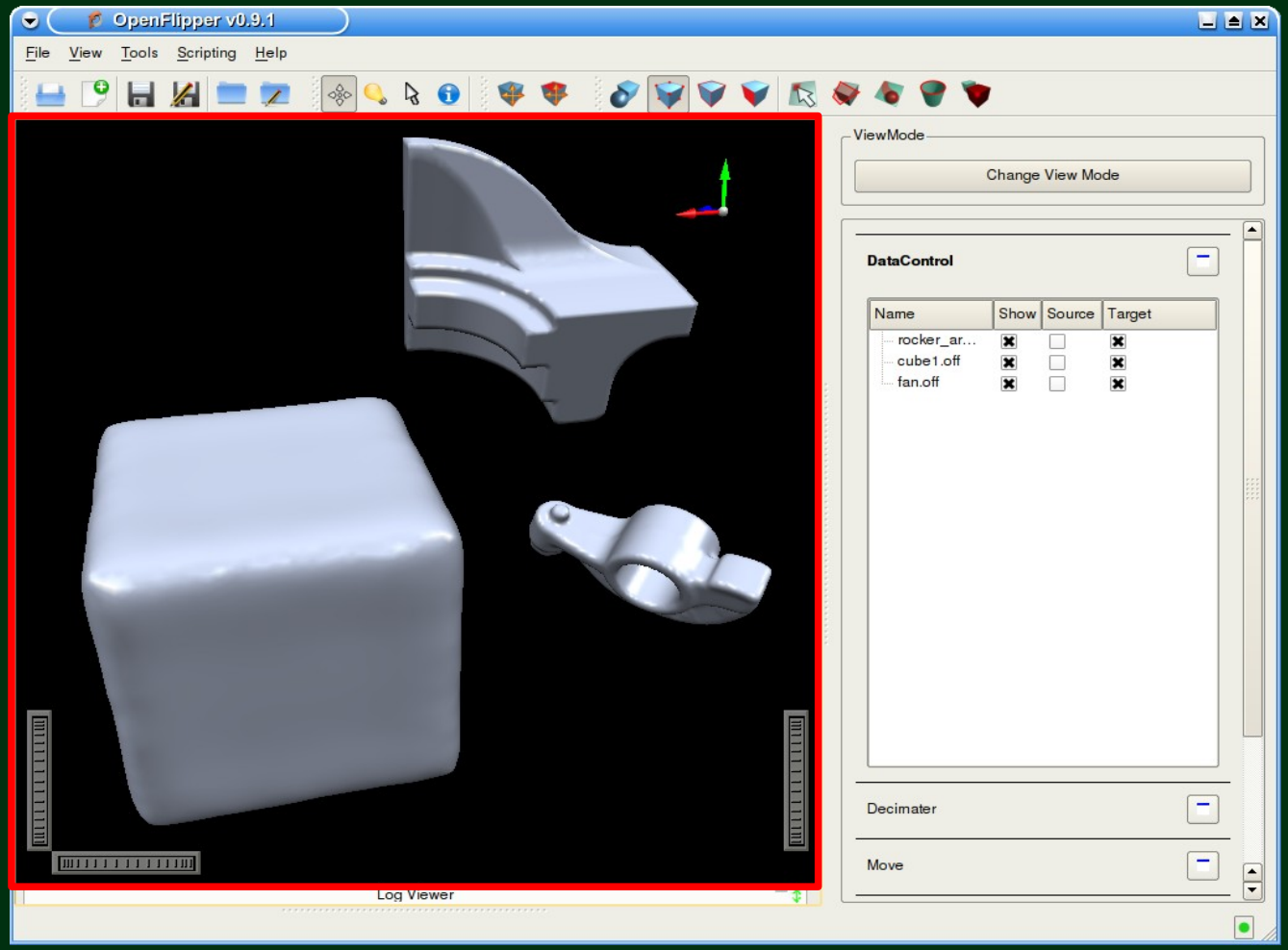
- Graphical User Interface
  - 3D Viewer
  - Selection Metaphors
- Flexible Scenegraph Structure
- Orthogonal Interaction Concepts
  - What (to do)
  - How
  - Where (to apply)
- Scripting

OpenFlipper

OpenFlipper

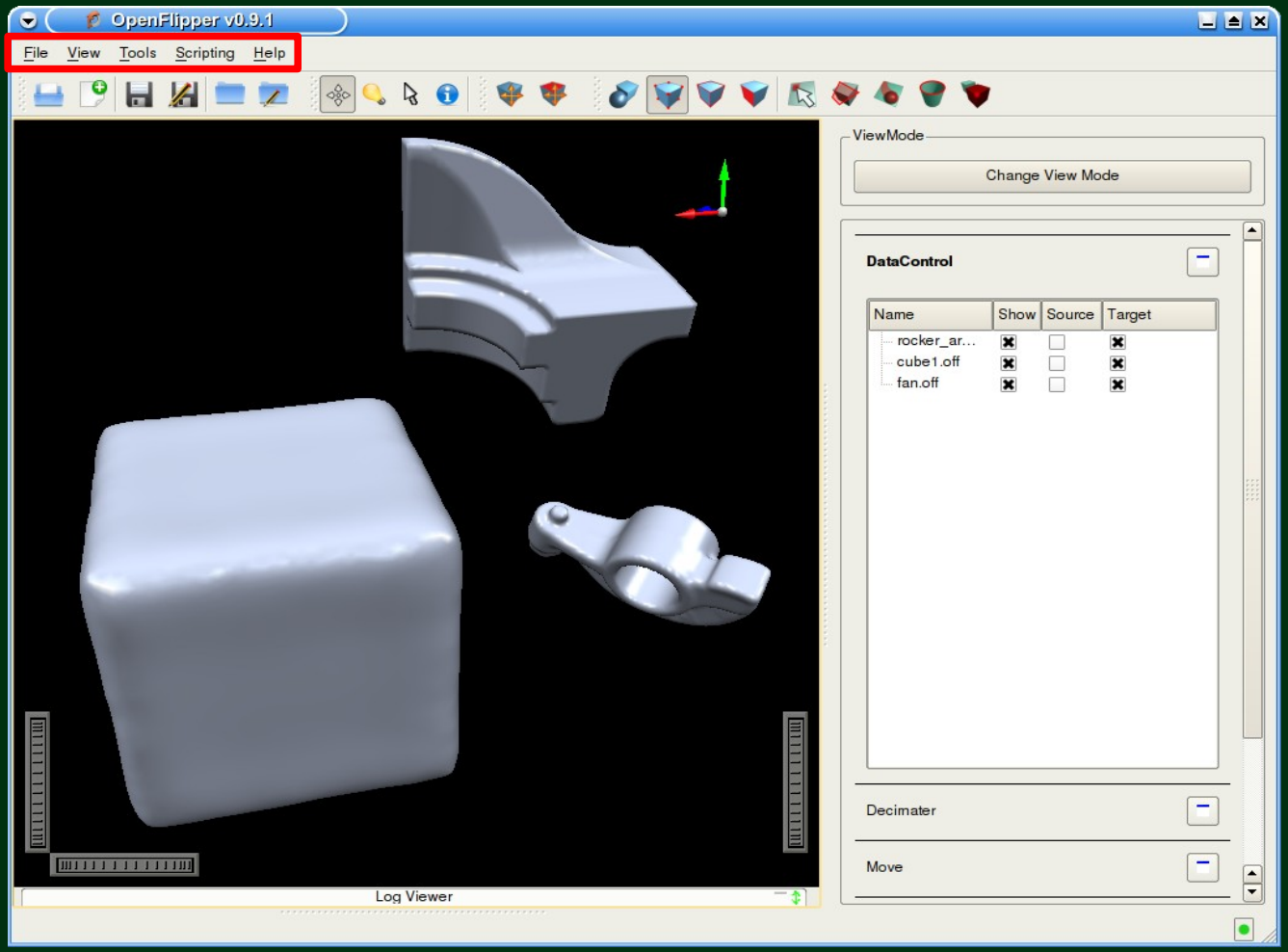


User Interface



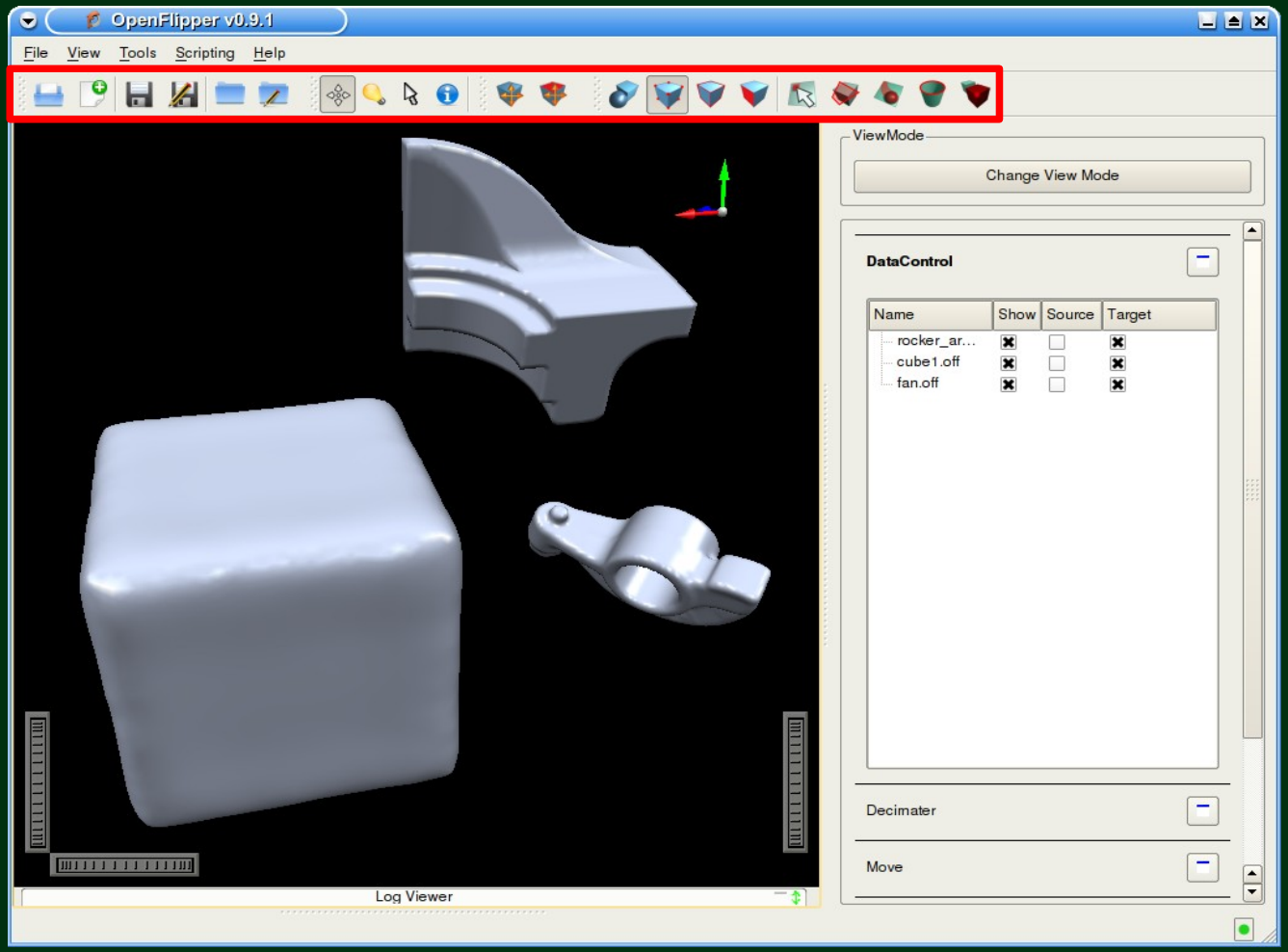
Viewer

User Interface



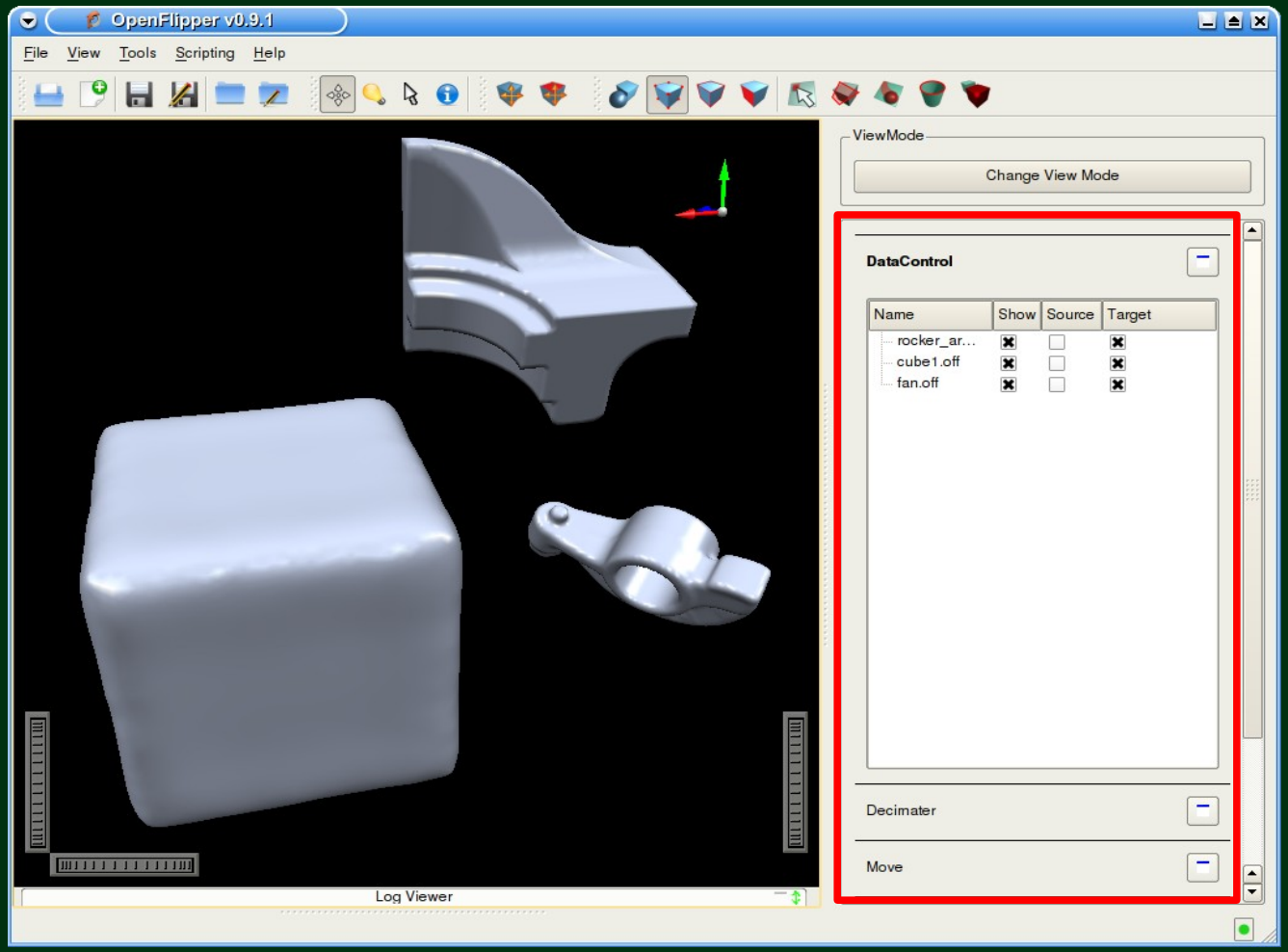
Menubar contains global functions

User Interface



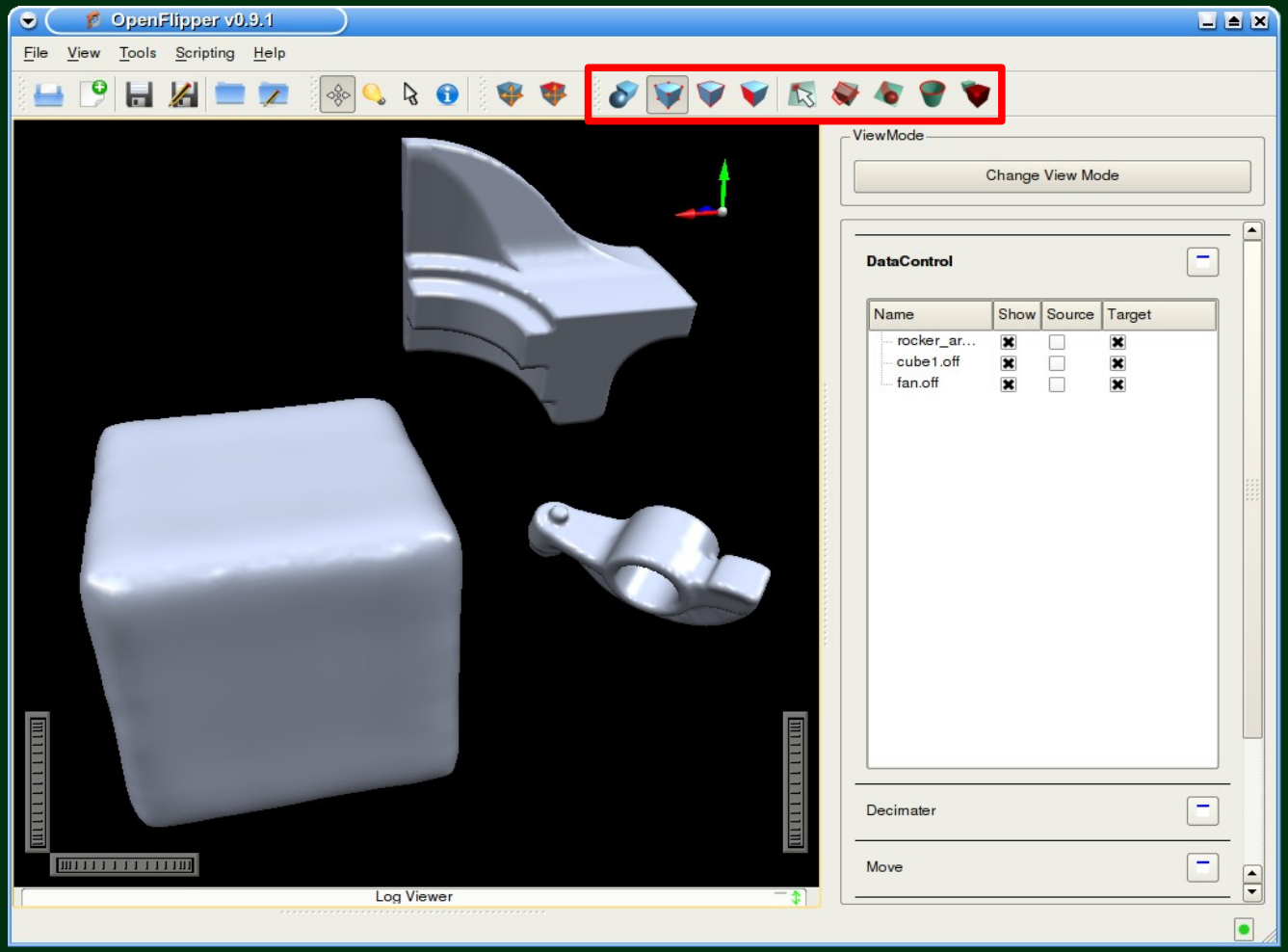
Toolbar for functions requiring user interaction

User Interface

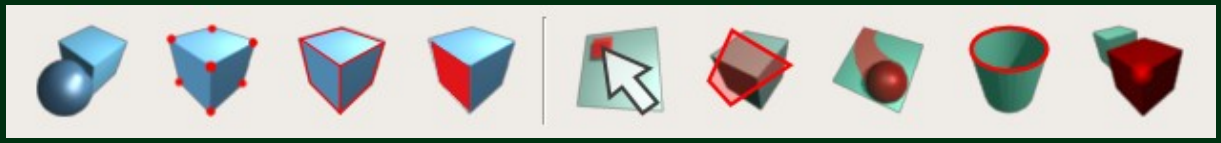


Toolbox for algorithm settings

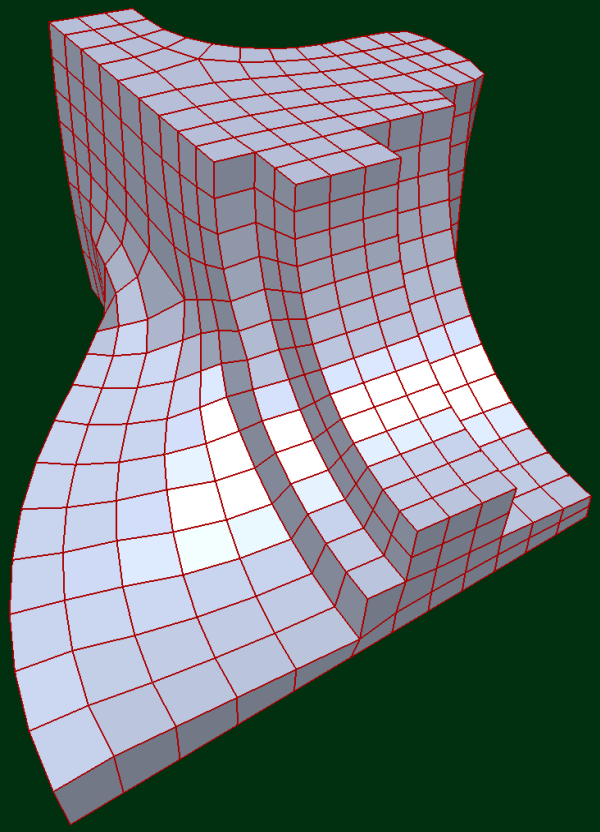
Selection Plugin

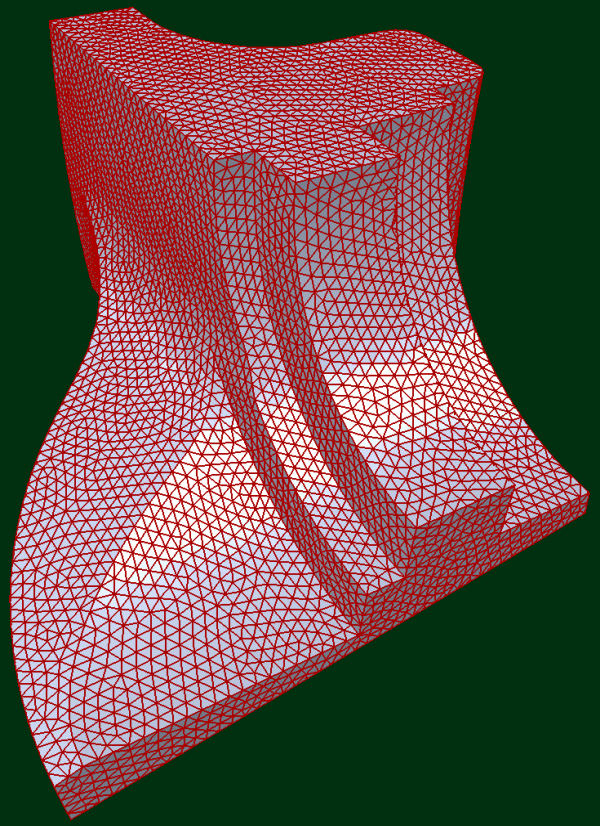


Selection Plugin

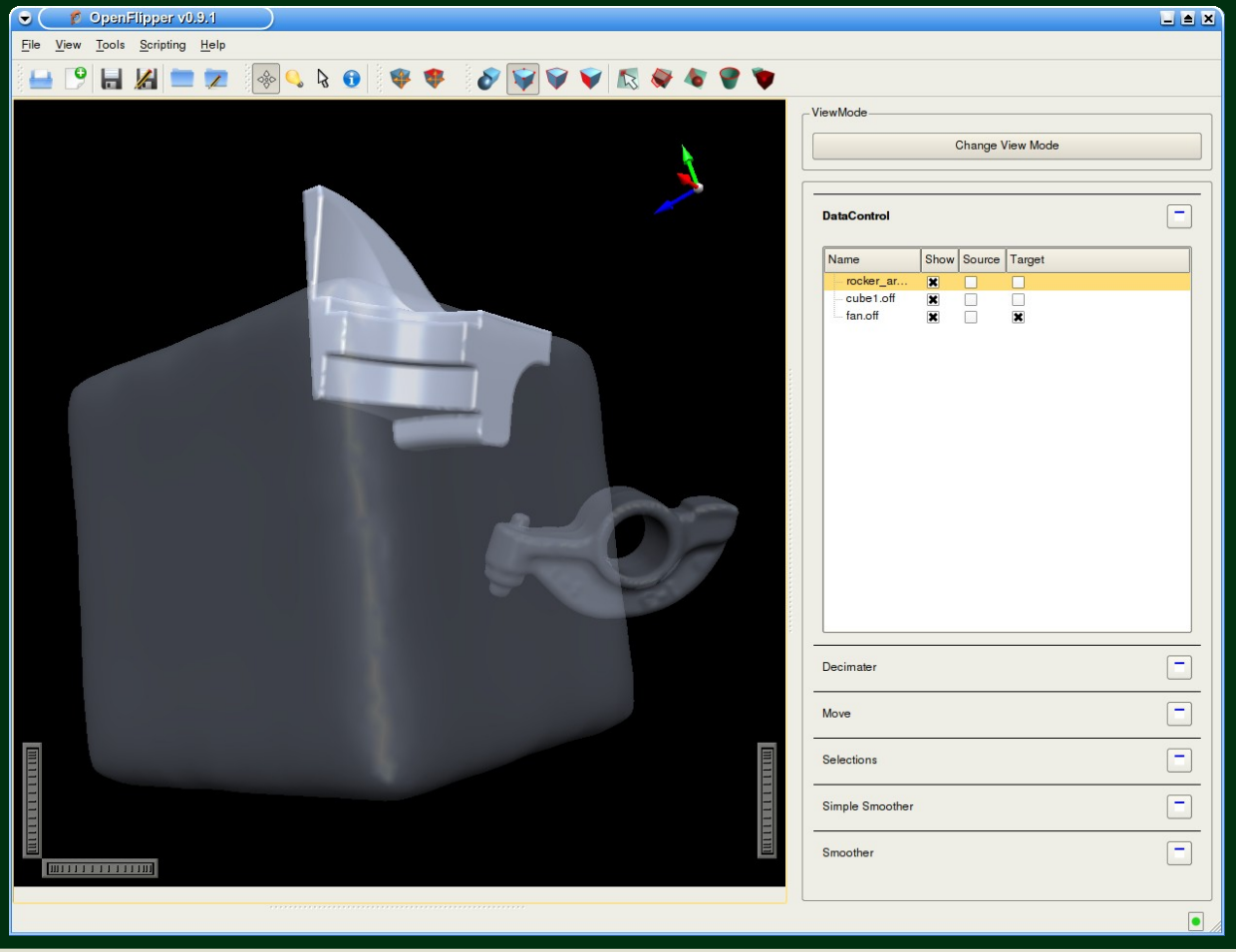


→ Selection on arbitrary Polygonal meshes

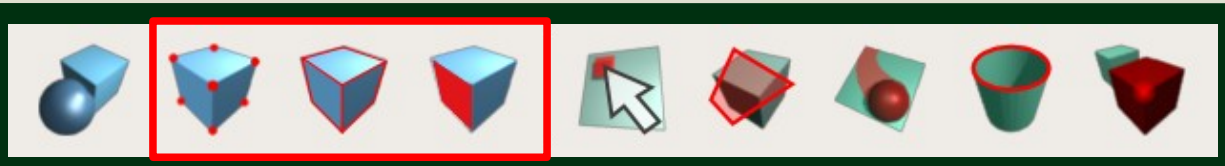




Selection Plugin



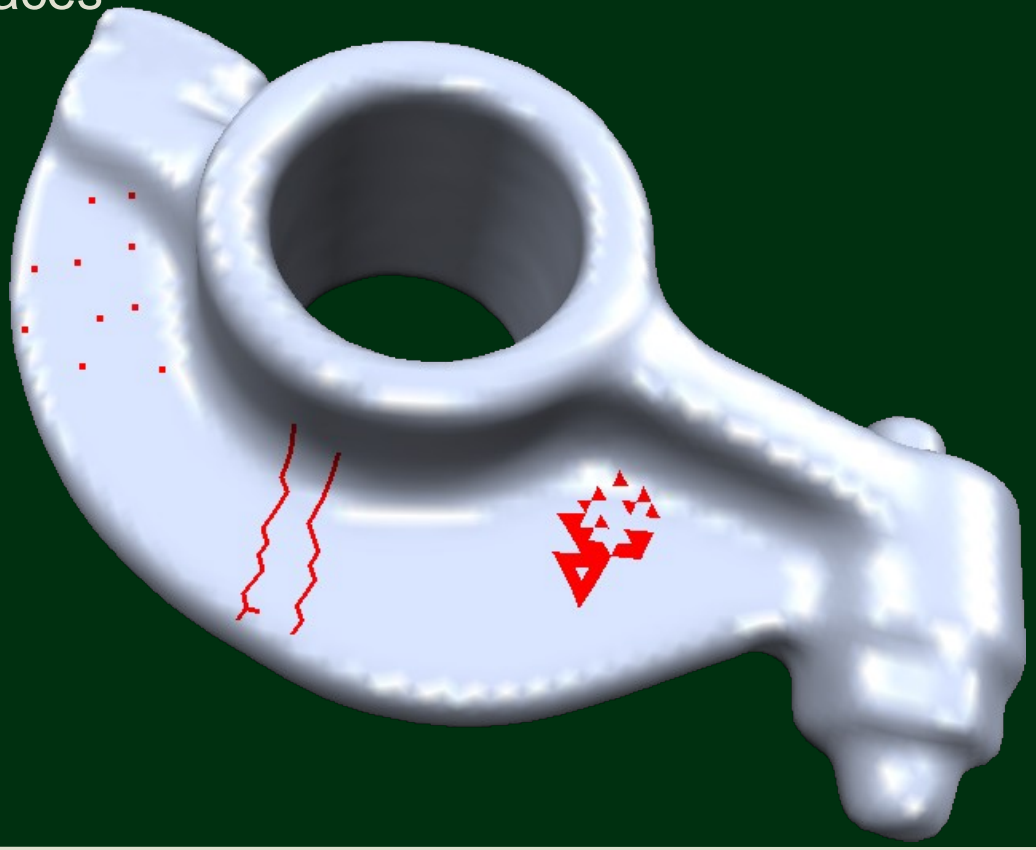
Selection Plugin



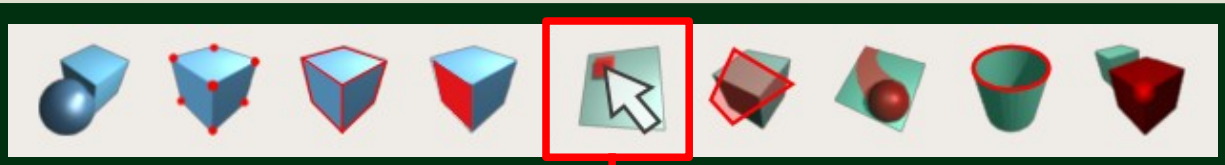
Vertices

Edges

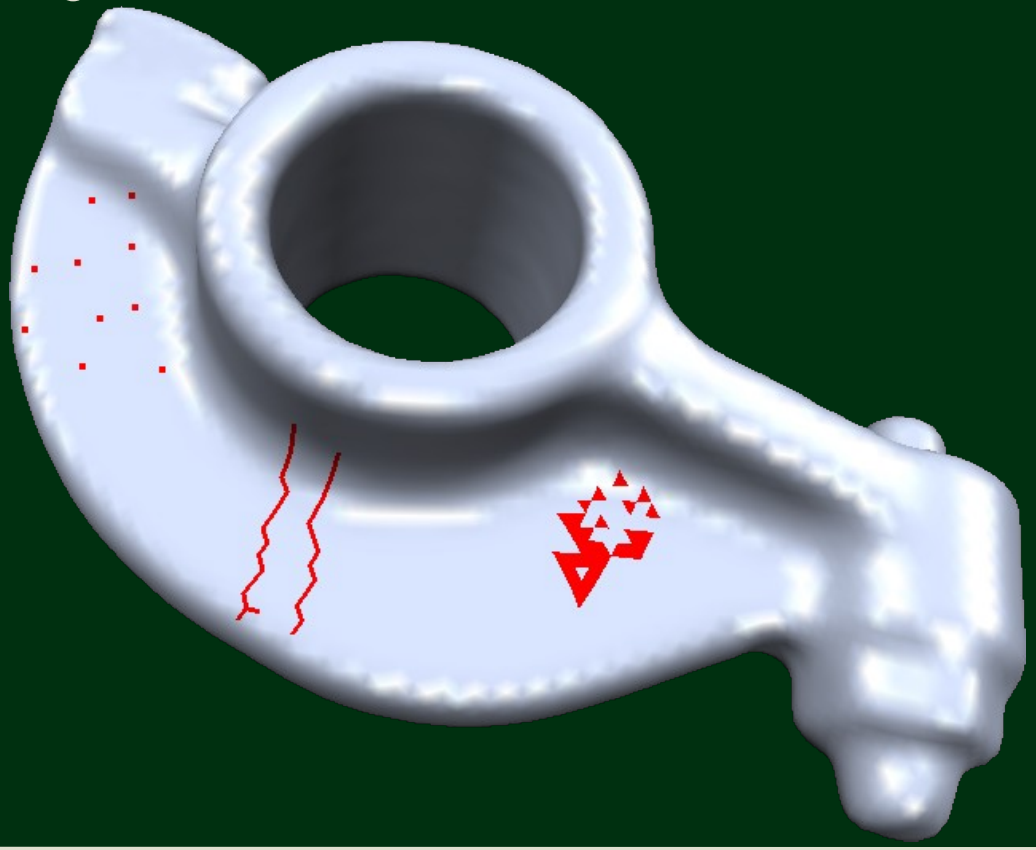
Faces



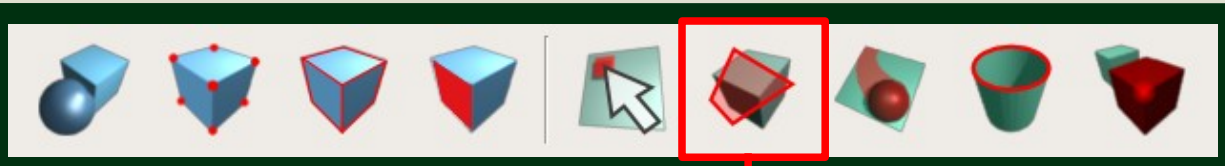
Selection Plugin



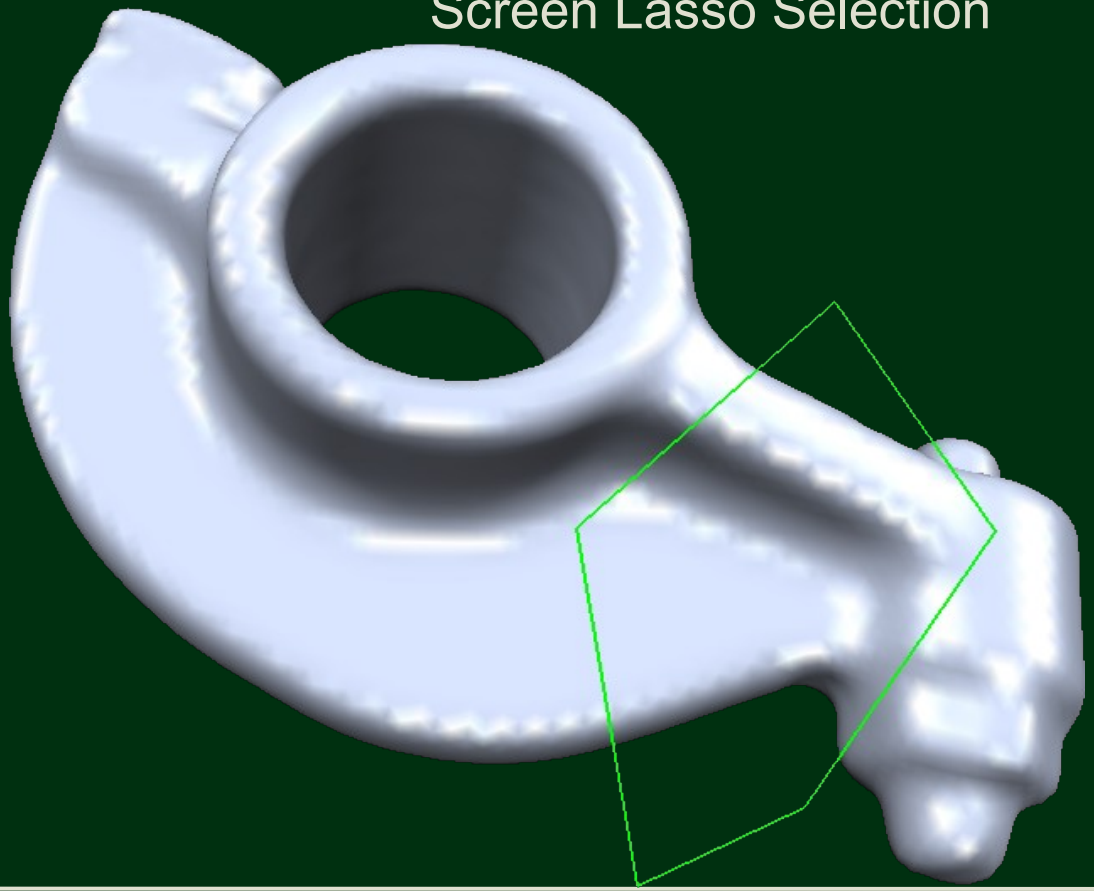
Single Selection



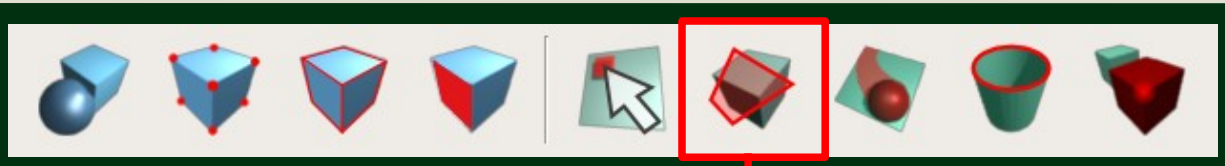
Selection Plugin



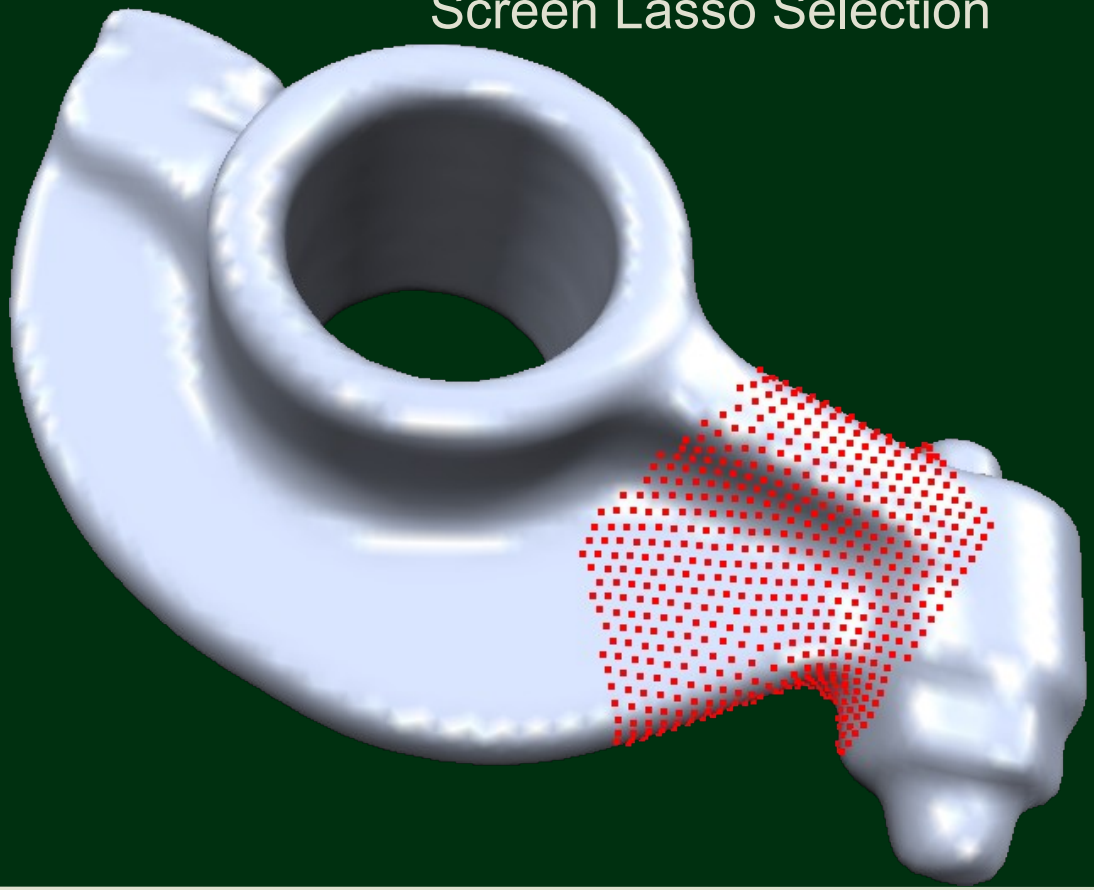
Screen Lasso Selection



Selection Plugin

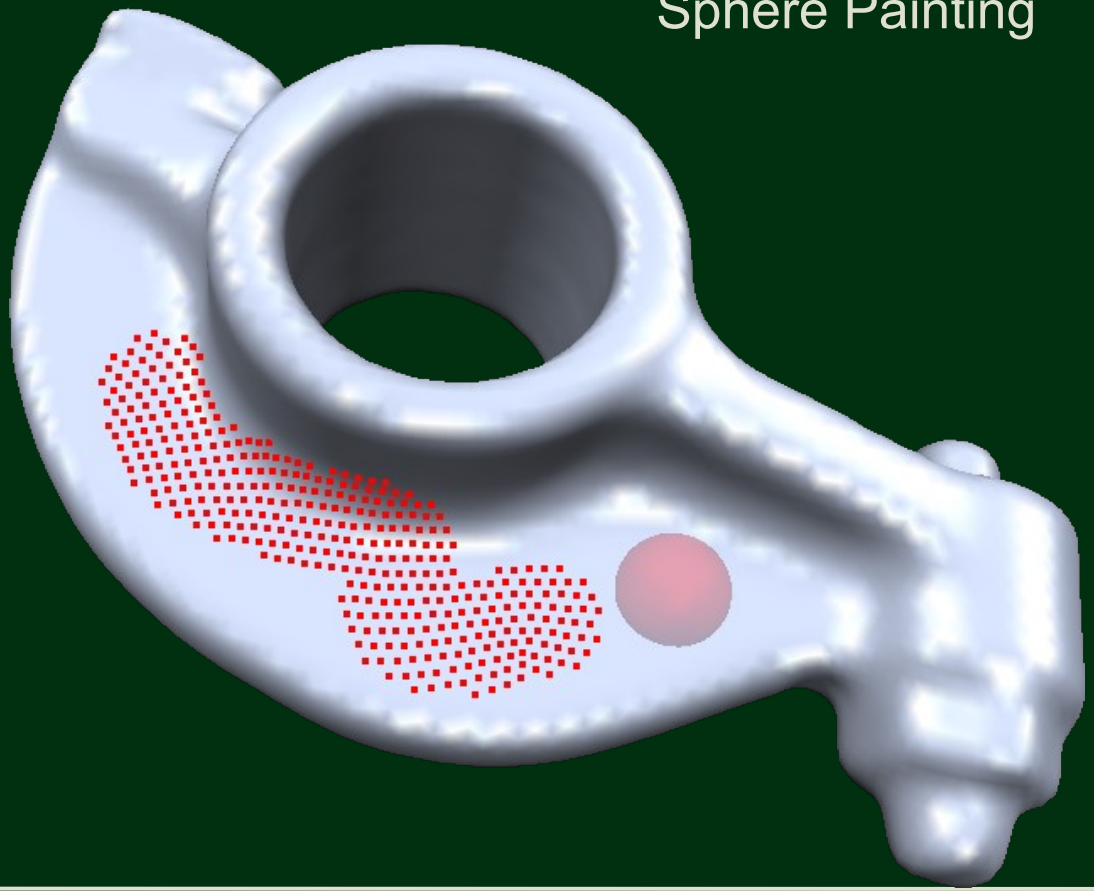


Screen Lasso Selection



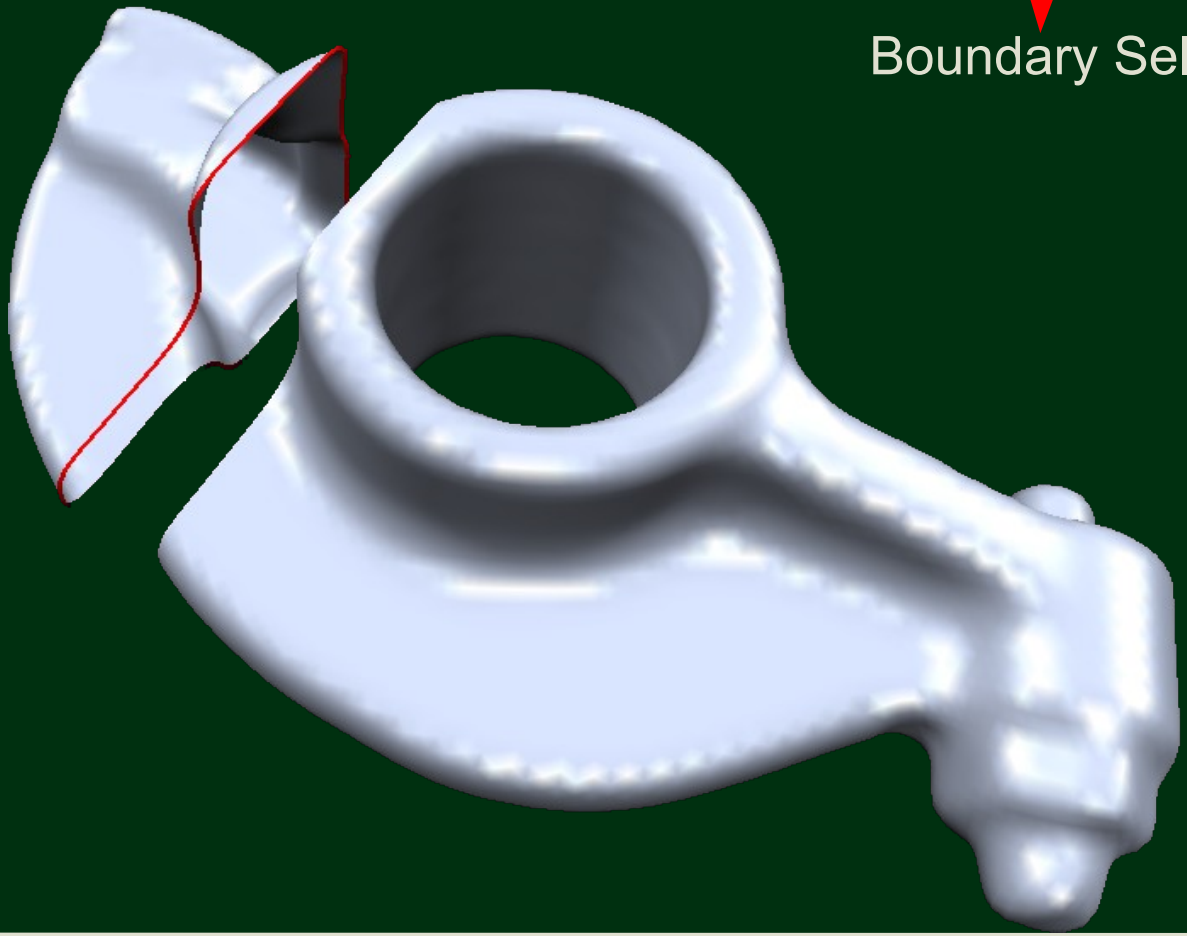
Selection Plugin

Sphere Painting

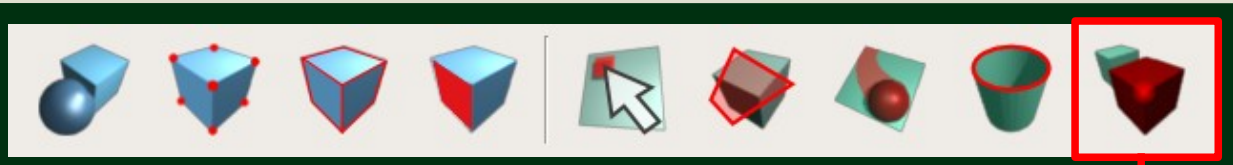


Selection Plugin

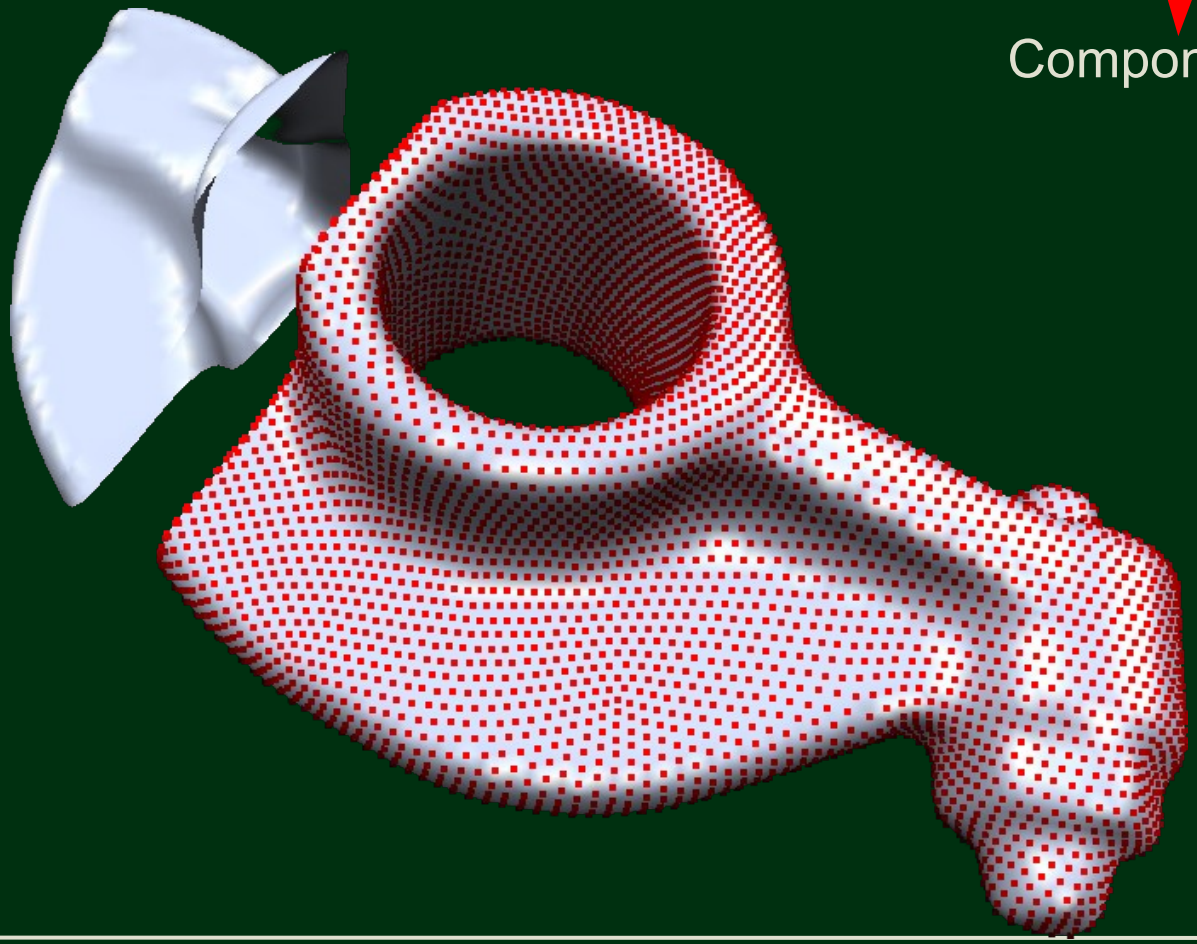
Boundary Selection



Selection Plugin

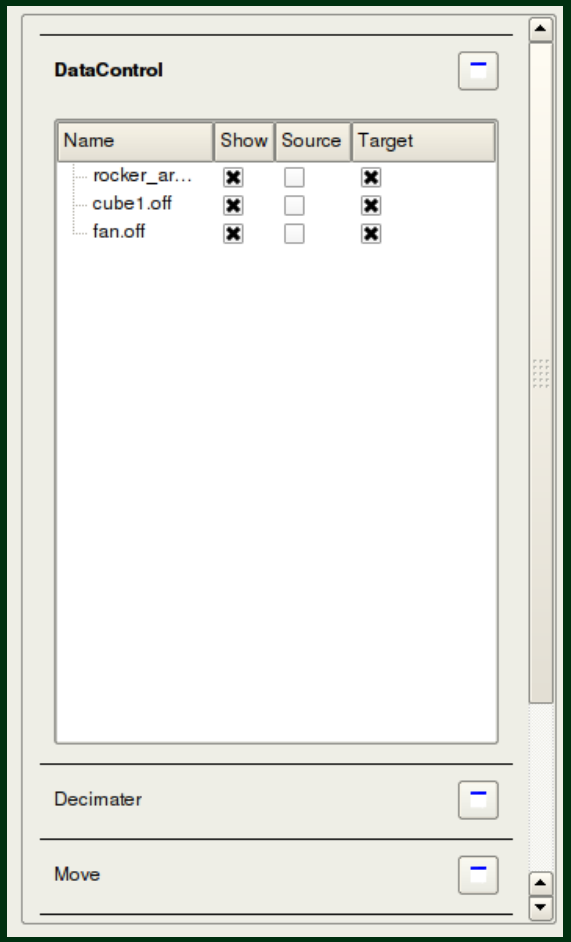


Component Selection

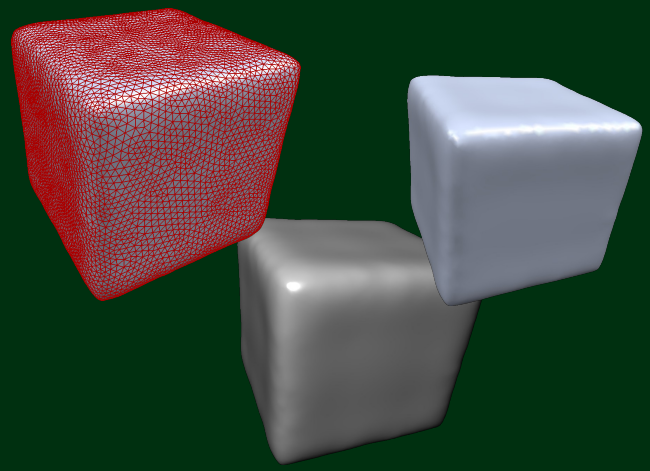


OpenFlipper Plugins

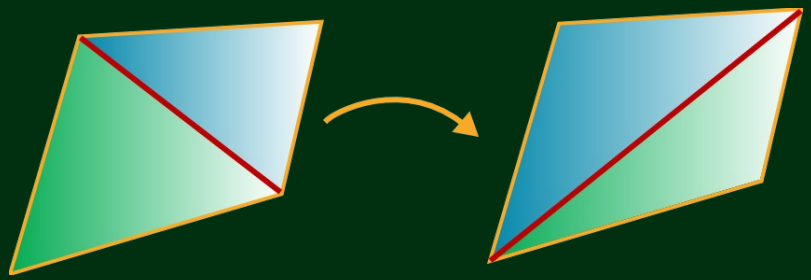
Data control



View Control



Topology

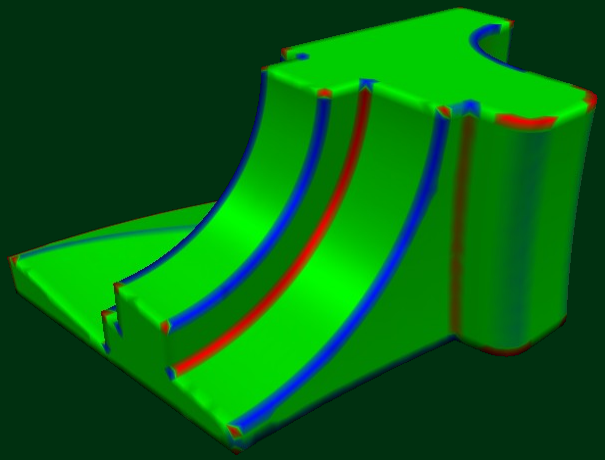


Flip Edges → OpenFlipper

Plugin System Other Interfaces

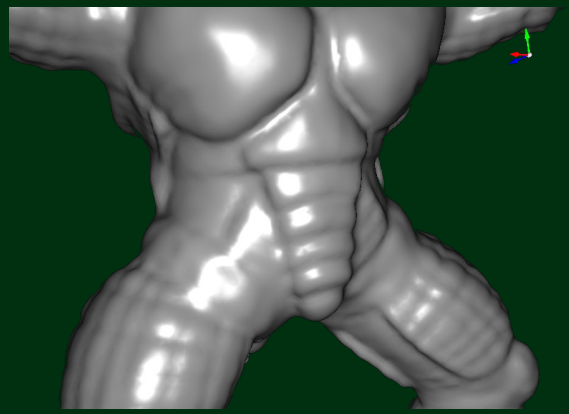
Texture control Plugin

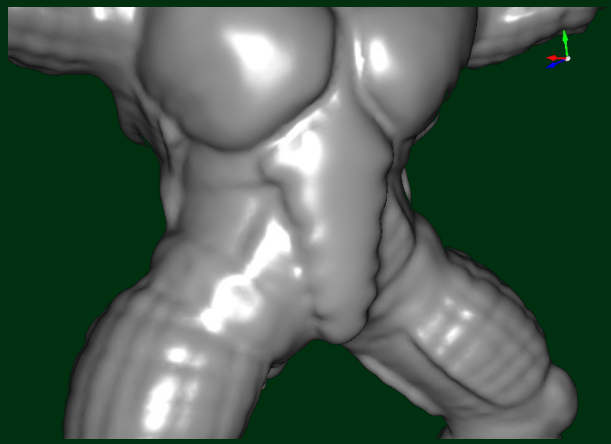
- Visualize arbitrary mesh properties
- Show textures



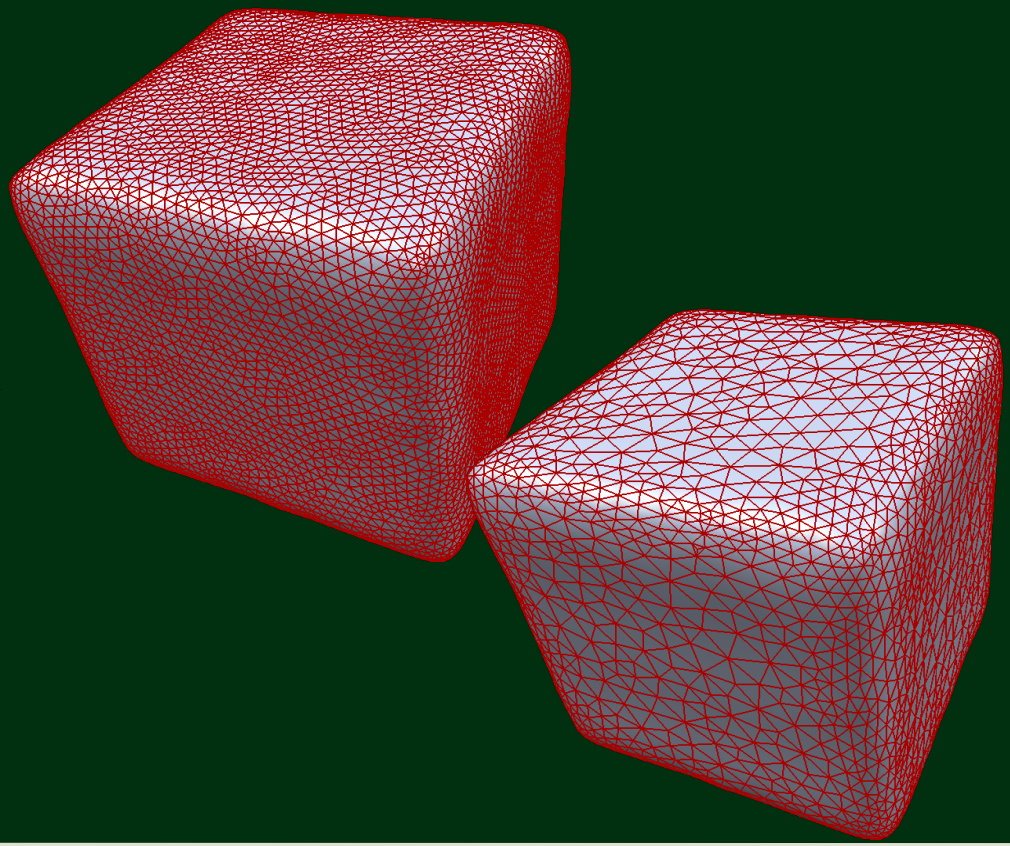
OpenFlipper Plugins

Smoother

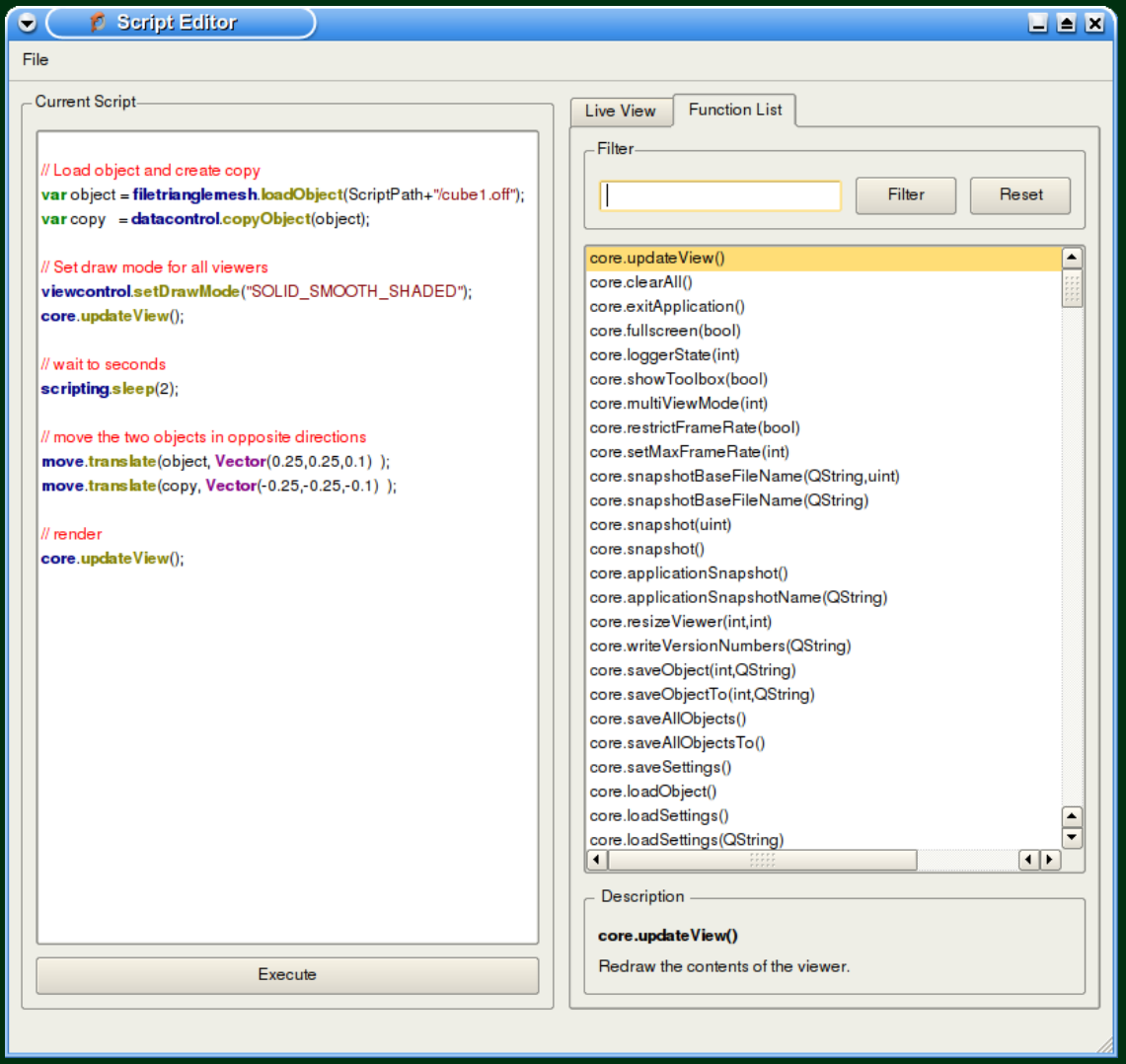




Decimater



OpenFlipper Scripting



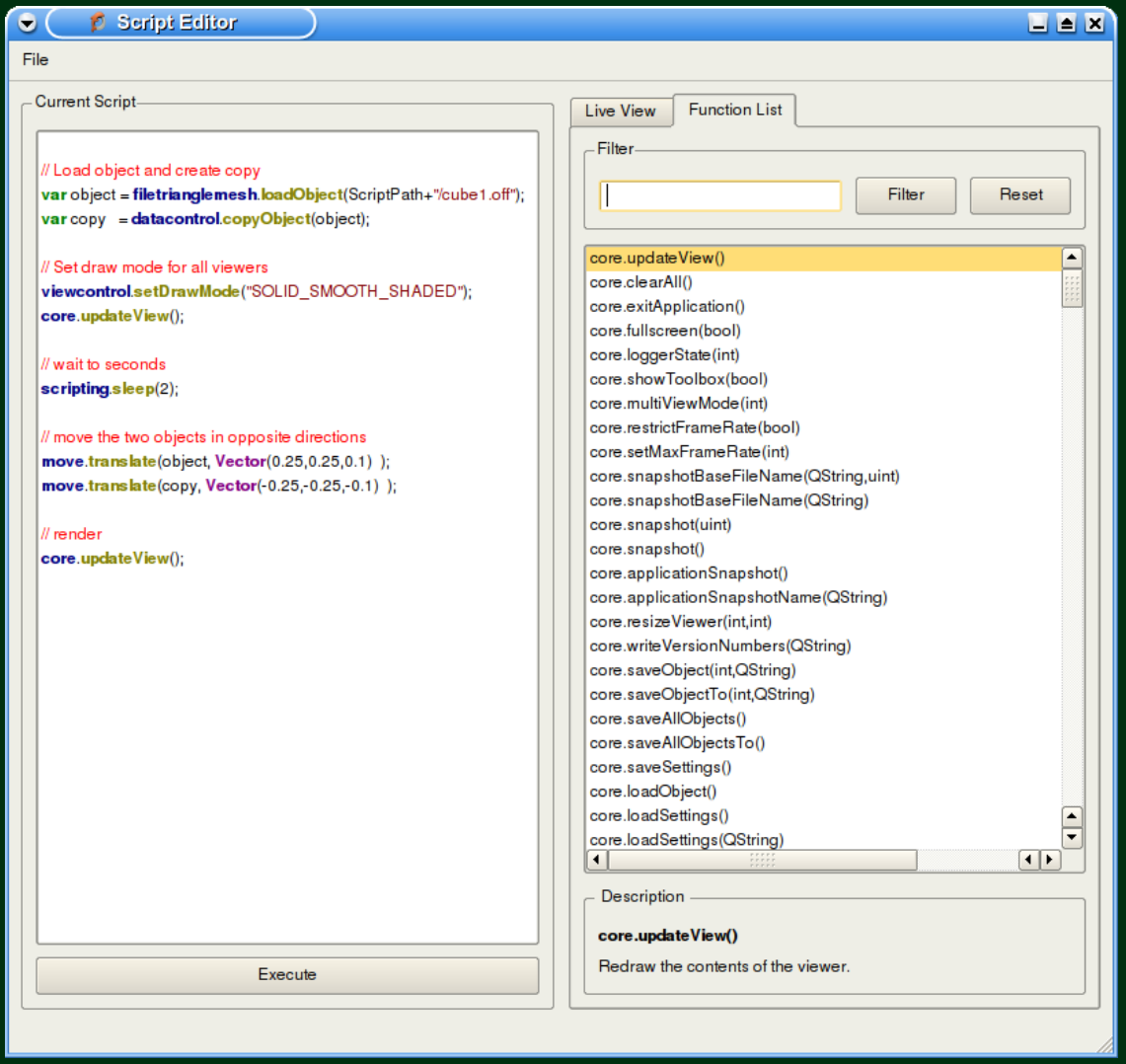
System

- Based on QtScript
- ECMAScript ECMA-262
- Plugins can easily provide functions for scripting
- Gui extension via script

Script Editor

- Collects available functions
- Collects descriptions
- Search for functions
- Editor with Syntax Highlighting
- Live View

OpenFlipper Scripting



System

- Based on QtScript
- ECMAScript ECMA-262
- Plugins can easily provide functions for scripting
- Gui extension via script

Script Editor

- Collects available functions
- Collects descriptions
- Search for functions
- Editor with Syntax Highlighting
- Live View

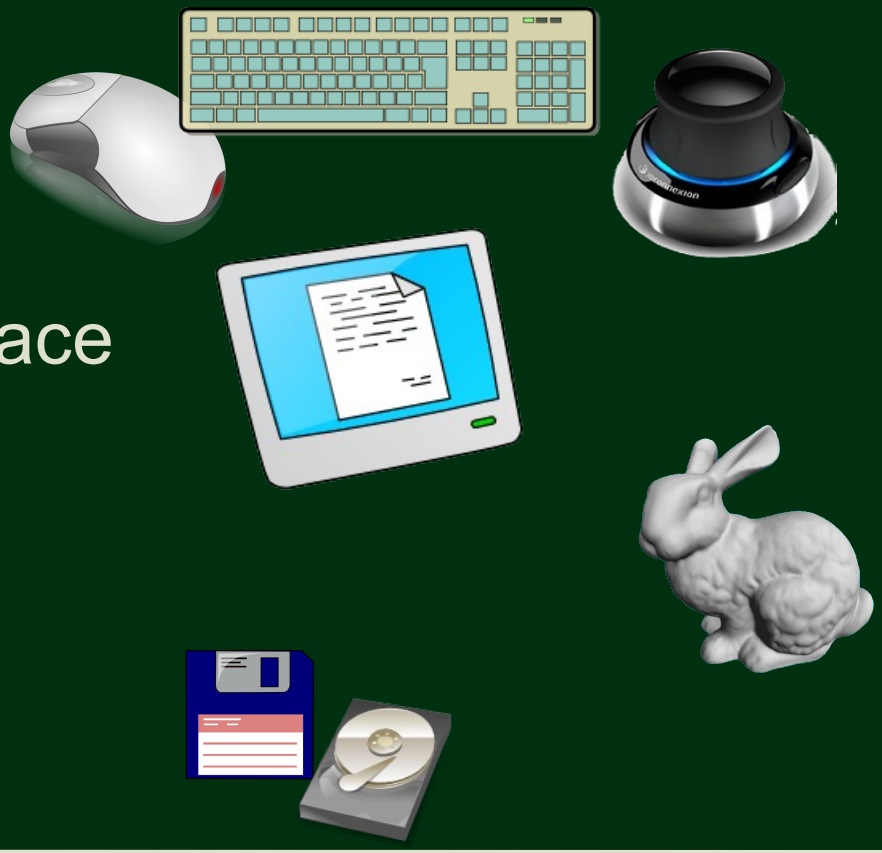
→ Writing a Plugin?

Plugin System

Loads plugins at startup / runtime

Uses Interfaces for communication with core

- User Interaction
- Graphical User Interface
- Visualization
- File/Object handling



Writing a Plugin

example.hh:

```
#include <OpenFlipper/common/Types.hh>
#include <OpenFlipper/BasePlugin/BaseInterface.hh>

class ExamplePlugin : public QObject, BaseInterface
{
  Q_OBJECT
  Q_INTERFACES(BaseInterface)

public :

  QString name() { return "ExamplePlugin"; };

  QString description() { return "Example Plugin Description"; };
};
```

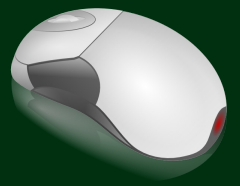
example.cc:

```
#include "example.hh"

Q_EXPORT_PLUGIN2( examplePlugin , ExamplePlugin );
```

Writing a Plugin

example.hh:

```
#include <OpenFlipper/common/Types.hh>
#include <OpenFlipper/BasePlugin/BaseInterface.hh>

class ExamplePlugin : public QObject, BaseInterface
{
  Q_OBJECT
  Q_INTERFACES(BaseInterface)

public :

  QString name() { return "ExamplePlugin"; };

  QString description() { return "Example Plugin Description"; };
};
```

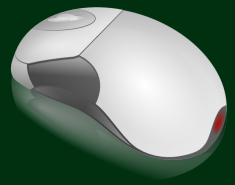


?

Writing a Plugin

example.hh:

```
#include <OpenFlipper/common/Types.hh>
#include <OpenFlipper/BasePlugin/BaseInterface.hh>
#include <OpenFlipper/BasePlugin/MouseInterface.hh>

class ExamplePlugin : public QObject, BaseInterface ,MouseInterface
{
  Q_OBJECT
  Q_INTERFACES(BaseInterface)
  Q_INTERFACES(MouseInterface)

private slots:
  void slotMouseEvent( QMouseEvent* _event ) {  …  };

public :

  QString name() { return "ExamplePlugin"; };

  QString description() { return "Example Plugin Description"; };
};
```



Writing a Plugin

Mesh Smoother :

```
private slots:
    void simpleLaplace() {

      for ( ObjectIterator o_it( TARGET_OBJECTS , DATA_TRIANGLE_MESH ) ;
                                  o_it != objectsEnd();
                              ++o_it)
      {
       Smooth ( triMesh(*o_it) ); // Get the mesh and smooth it

        emit updatedObject(o_it->id());
      }
    }
```

Writing a Plugin

Mesh Smoother :

```
private slots:
    void simpleLaplace() {

      for ( ObjectIterator o_it( TARGET_OBJECTS , DATA_TRIANGLE_MESH ) ;
                                  o_it != objectsEnd();
                              ++o_it)
      {
       Smooth ( triMesh(*o_it) ); // Get the mesh and smooth it

        emit updatedObject(o_it->id());
      }
    }
```

Scripting?

Writing a Plugin

Mesh Smoother :

```
public slots:
   void simpleLaplace() {

     for ( ObjectIterator o_it( TARGET_OBJECTS , DATA_TRIANGLE_MESH ) ;
                                o_it != objectsEnd();
                              ++o_it)
     {
      Smooth ( triMesh(*o_it) ); // Get the mesh and smooth it

       emit updatedObject(o_it->id());
     }
  }
```

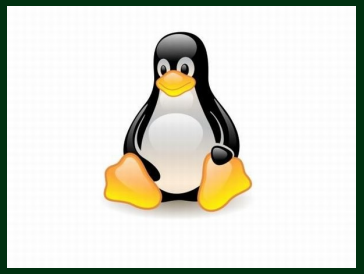
Scripting?

→ exampleplugin.simpleLaplace()

OpenFlipper

Demo

Supported Architectures



Linux

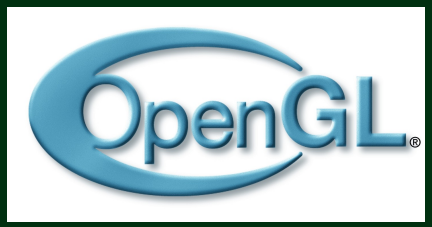
Windows(XP/ Vista)

Mac OS X

All Architectures in 32 / 64 -bit

Build Dependencies

Qt Framework (>= 4.5)



OpenGL

OpenGL Extension Wrangler Library (GLEW)

OpenGL Utility Toolkit (GLUT)

OpenMesh

OpenMesh (2.0 included)

OpenMesh 2.0

OpenMesh

Generic and efficient data structure for representing and manipulating polygonal meshes.

License:	LGPL 2.1
Latest Version:	OpenMesh 2.0
For more information see:	www.openmesh.org

Supported Compilers

Linux & MacOS X

GNU Compiler Collection (GCC >=4.0)

Windows

Visual Studio 2008

Windows / Linux (GCC >=4.3)

Conclusion

- Multi Platform Geometry processing Framework
- Open Source (License LGPL 3)
- Flexible User Interface
- Simple plugin api

Thank You!

www.OpenFlipper.org