

Writing a K-3D plugin

Bart Janssens

24 Feb 2008

Overview

About K-3D

Using a standard plugin

Writing a plugin

- Anatomy of a plugin

- Pick an interface and add properties

- Implement constructor and methods

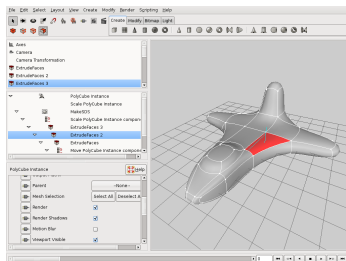
- Register the plugin

Using the new plugin

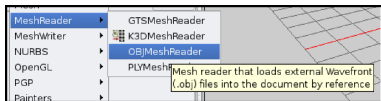
Conclusion

About K-3D

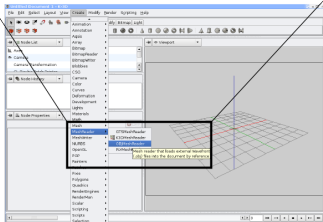
- ▶ Free 3D modeling and animation package, written by Timothy M. Shead
- ▶ Supports multiple external rendering engines, i.e. Aqsis
- ▶ Very modular “everything is a plugin” design
- ▶ Connect plugins using a visual programming system



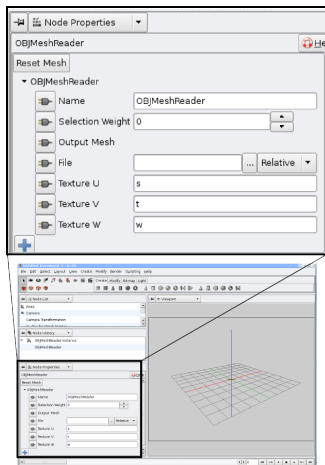
Using a standard plugin



- ▶ Create a new plugin node from the Create menu

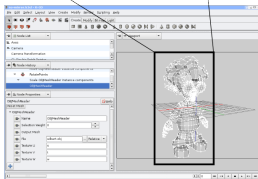
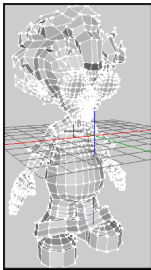


Using a standard plugin



- ▶ Create a new plugin node from the Create menu
- ▶ The node shows up in the node list, and its properties are shown

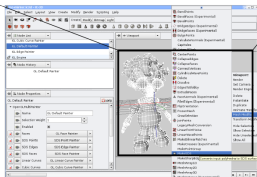
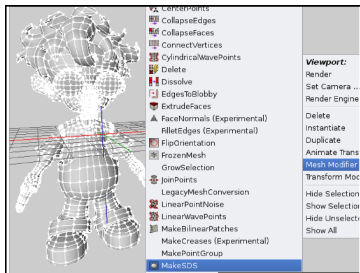
Using a standard plugin



Wilbert model courtesy of Joe Crawford, <http://celestinestudios.com>

- ▶ Create a new plugin node from the Create menu
- ▶ The node shows up in the node list, and its properties are shown
- ▶ **Set the parameter properties**

Using a standard plugin



- ▶ Create a new plugin node from the Create menu
- ▶ The node shows up in the node list, and its properties are shown
- ▶ Set the parameter properties
- ▶ Apply the SDS modifier

Anatomy of a plugin

- ▶ Reference: http://www.k-3d.org/wiki/Plugin_Tutorial
- ▶ CMake file:

```
K3D_BUILD_EXTERNAL_MODULE(sample-plugin ${sample_plugin_EXTERNAL_SOURCE_DIR})
K3D_CREATE_MODULE_PROXY(sample-plugin)
```

- ▶ A `module.cpp` file, with one or more node classes to do the real work, containing
 - ▶ A class implementing an interface or base class that inherits from `k3d::inode`
 - ▶ ... with member Properties to set the node inputs, outputs and parameters
 - ▶ and a plugin factory with a unique id
- ▶ Example: mirror tool



Pick an interface and add properties

- ▶ Base class to change mesh geometry:

k3d::mesh_deformation_modifier

- ▶ Declare the class:

```
class mirror :  
    public k3d::mesh_deformation_modifier  
{  
    typedef k3d::mesh_deformation_modifier base;
```

- ▶ Declare the parameter properties:

```
private :  
    k3d_data(bool, immutable_name, change_signal,  
            with_undo, local_storage, no_constraint,  
            writable_property, with_serialization)  
    m_mirror_x;
```

Implement the constructor

```
mirror(k3d::iplugin_factory& Factory ,
       k3d::idocument& Document) :
    base(Factory , Document) ,
    m_mirror_x(init_owner(*this) +
               init_name("mirror_x") +
               init_label("Mirror_X") +
               init_description("Mirror_the_X_component") +
               init_value(true))
{
    m_mirror_x.changed_signal().connect(
        make_update_mesh_slot());
}
```

Implement modifier method

```
void on_deform_mesh(const k3d::mesh& Input ,
    const k3d::mesh::points_t& InputPoints ,
    const k3d::mesh::selection_t& PointSelection ,
    k3d::mesh::points_t& OutputPoints)
{
    for (k3d::uint_t i=0;i!=OutputPoints.size();++ i)
    {
        k3d::point3 p = InputPoints[i];
        if (m_mirror_x.pipeline_value()) p[0] = -p[0];
        if (m_mirror_y.pipeline_value()) p[1] = -p[1];
        if (m_mirror_z.pipeline_value()) p[2] = -p[2];
        OutputPoints[i] = p;
    }
}
```

Factory method

Factory method as static member of `mirror`:

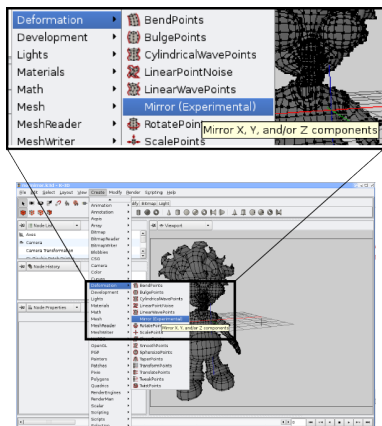
```
static k3d::iplugin_factory& get_factory ()
{
    static k3d::document_plugin_factory<mirror ,
        k3d::interface_list<k3d::imesh_source ,
        k3d::interface_list<k3d::imesh_sink > > >
        factory (
            k3d::uuid(0x8ead46a5, ... 0xa965e7ec),
            "Mirror",
            "Mirror_X,_Y,_and/or_Z_components",
            "Deformation",
            k3d::iplugin_factory::EXPERIMENTAL);
    return factory;
}
```

Factory registration

Registration macro at end of `module.cpp`:

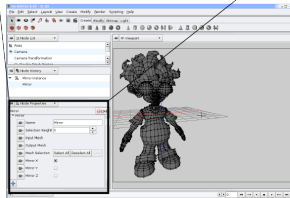
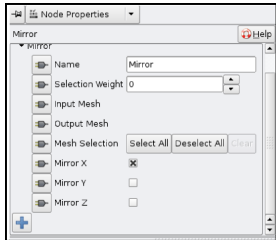
```
K3D_MODULE_START( Registry )
    Registry . register_factory (
        mirror :: mirror :: get_factory ( ) );
K3D_MODULE_END
```

Using the new plugin



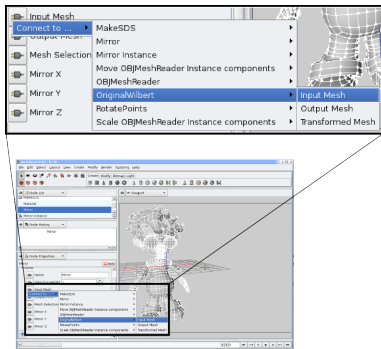
- ▶ The new plugin shows up in the Create menu

Using the new plugin



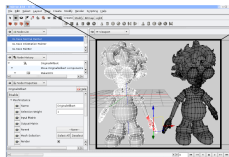
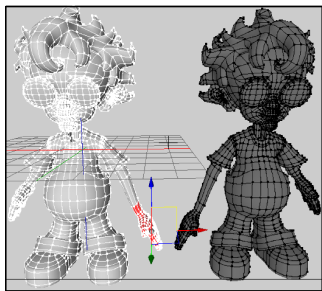
- ▶ The new plugin shows up in the Create menu
- ▶ The properties automatically show up in the GUI

Using the new plugin



- ▶ The new plugin shows up in the Create menu
- ▶ The properties automatically show up in the GUI
- ▶ We can now connect the mirror to the original mesh

Using the new plugin



- ▶ The new plugin shows up in the Create menu
- ▶ The properties automatically show up in the GUI
- ▶ We can now connect the mirror to the original mesh
- ▶ Changes to the original mesh get mirrored

Conclusion

- ▶ K-3D's modular design makes it very easy for people interested in CG to start writing new modules
- ▶ More information:
 - ▶ <http://www.k-3d.org>
 - ▶ k3d-development@lists.sourceforge.net

