How to use Maya2Renderer

- Development environment

see ./readme.txt

- Build

see ./readme.txt

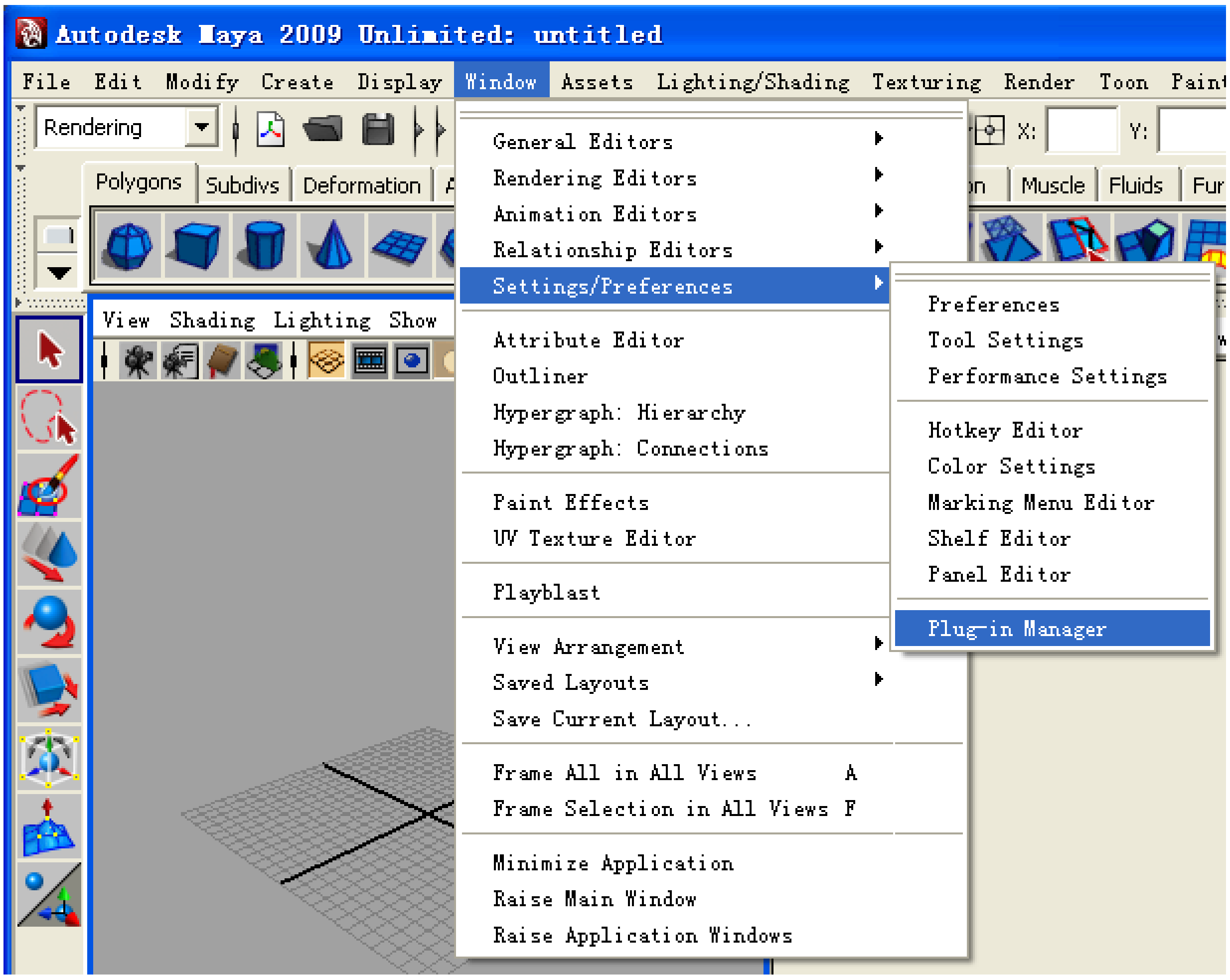
- Install

see ./readme.txt

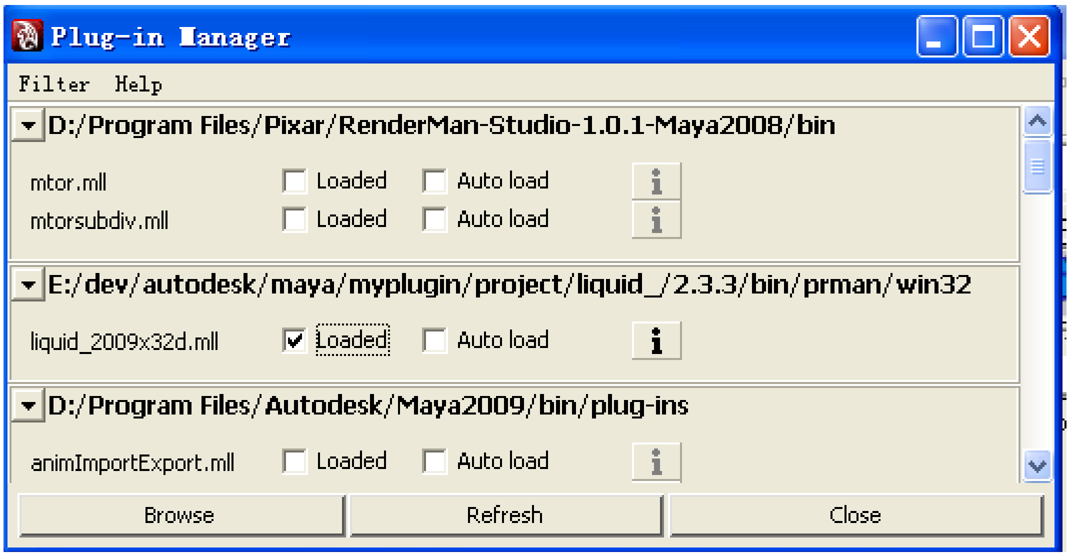
-load liquid plugin

Demo scene is ($liquid_root)/2.3.3/doc/HowToUseMaya2Renderer.ma

1)



2) select liquid_2009x32d.mll



Plug-in Manager

Filter Help

D:/Program Files/Pixar/RenderMan-Studio-1.0.1-Maya2008/bin

mtor.mll	☐ Loaded	☐ Auto load
mtorsubdiv.mll	☐ Loaded	☐ Auto load

E:/dev/autodesk/maya/myplugin/project/liquid_/2.3.3/bin/prman/win32

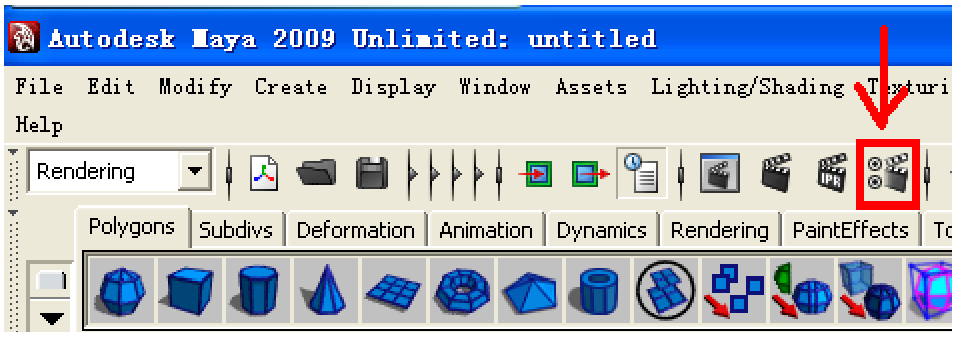
liquid_2009x32d.mll	☑ Loaded	☐ Auto load

D:/Program Files/Autodesk/Maya2009/bin/plug-ins

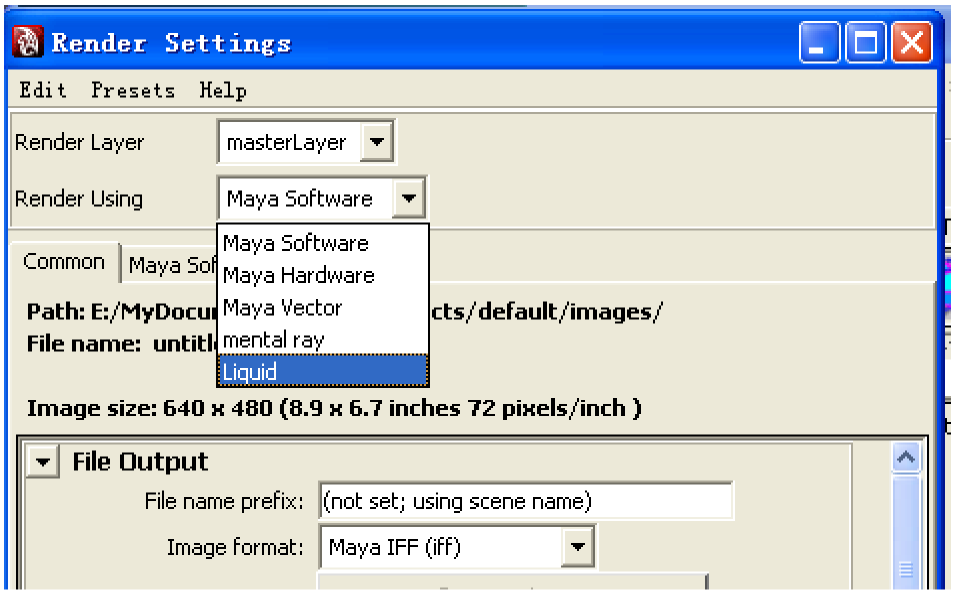
animImportExport.mll	☐ Loaded	☐ Auto load

-select liquid renderer

1)


Autodesk Maya 2009 Unlimited: untitled
File Edit Modify Create Display Window Assets Lighting/Shading Texturi
Help
Rendering
Polygons Subdivs Deformation Animation Dynamics Rendering PaintEffects To

2)



Render Layer masterLayer

Render Using Maya Software

Path: E:/MyDocu cts/default/images/

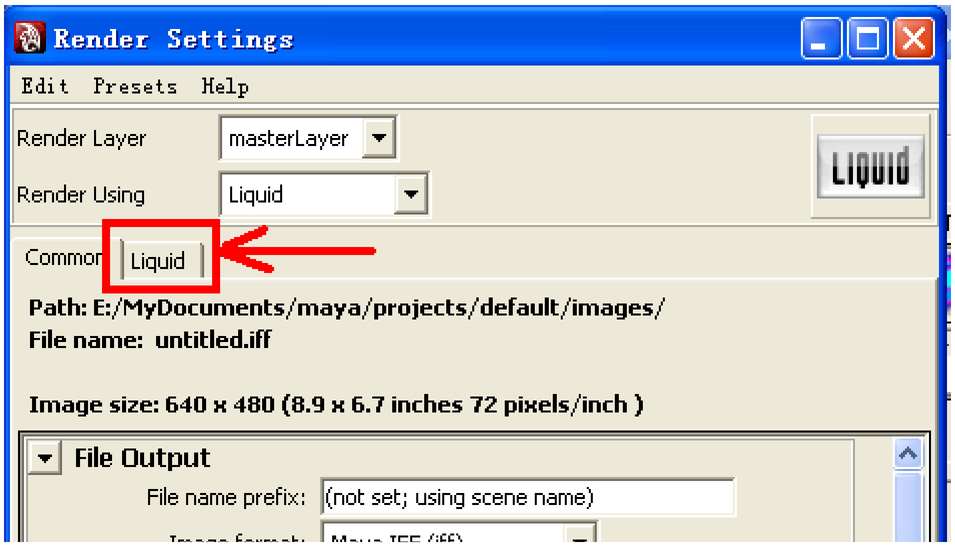
File name: untitl

Image size: 640 x 480 (8.9 x 6.7 inches 72 pixels/inch)

File name prefix: (not set; using scene name)

Image format: Maya IFF (iff)

3)



Render Layer masterLayer

Render Using Liquid

Path: E:/MyDocuments/maya/projects/default/images/

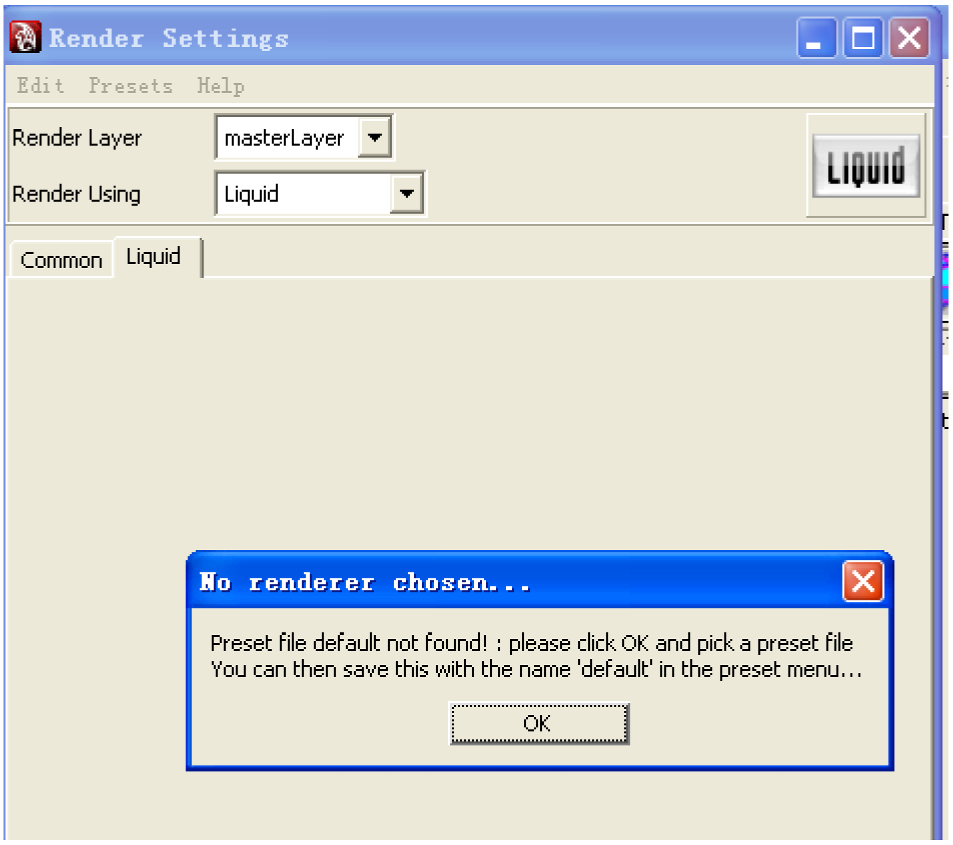
File name: untitled.iff

Image size: 640 x 480 (8.9 x 6.7 inches 72 pixels/inch)

File name prefix: (not set; using scene name)

Image format: Maya IFF (iff)

4)

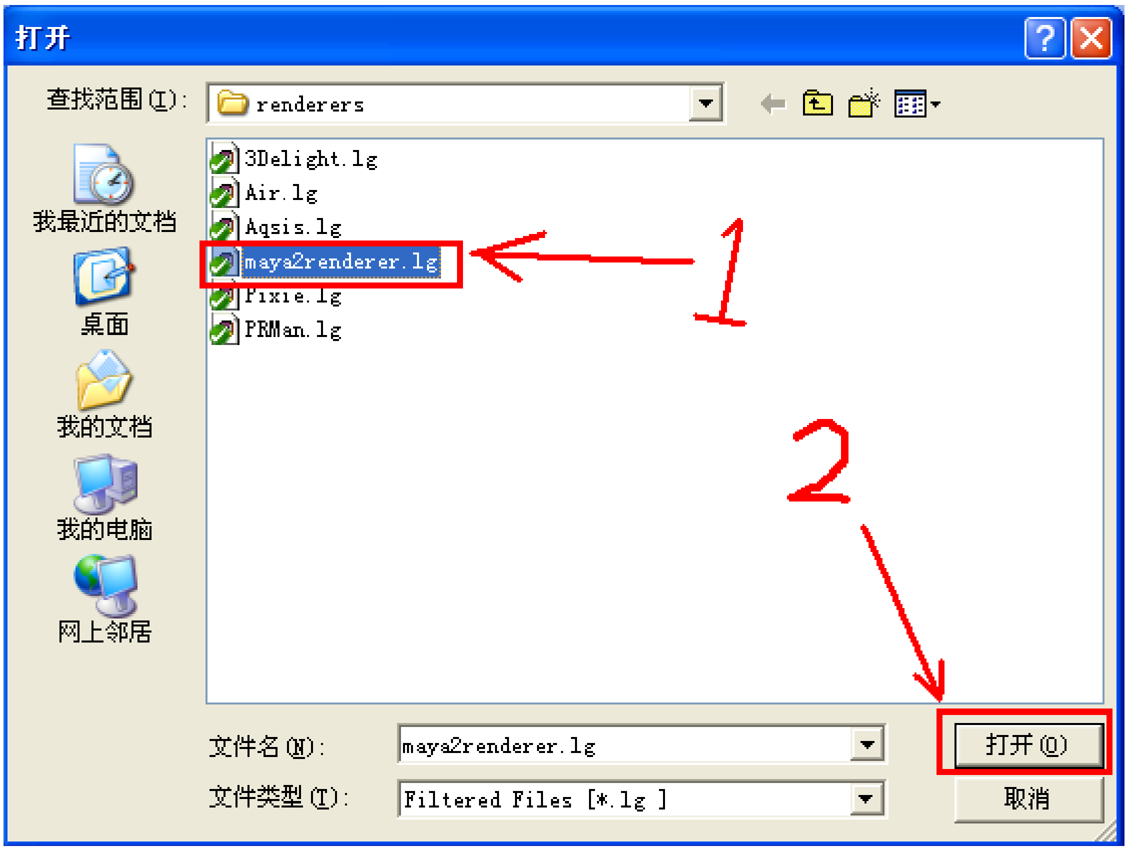


Render Layer: masterLayer

Render Using: Liquid

Preset file default not found! : please click OK and pick a preset file
You can then save this with the name 'default' in the preset menu...

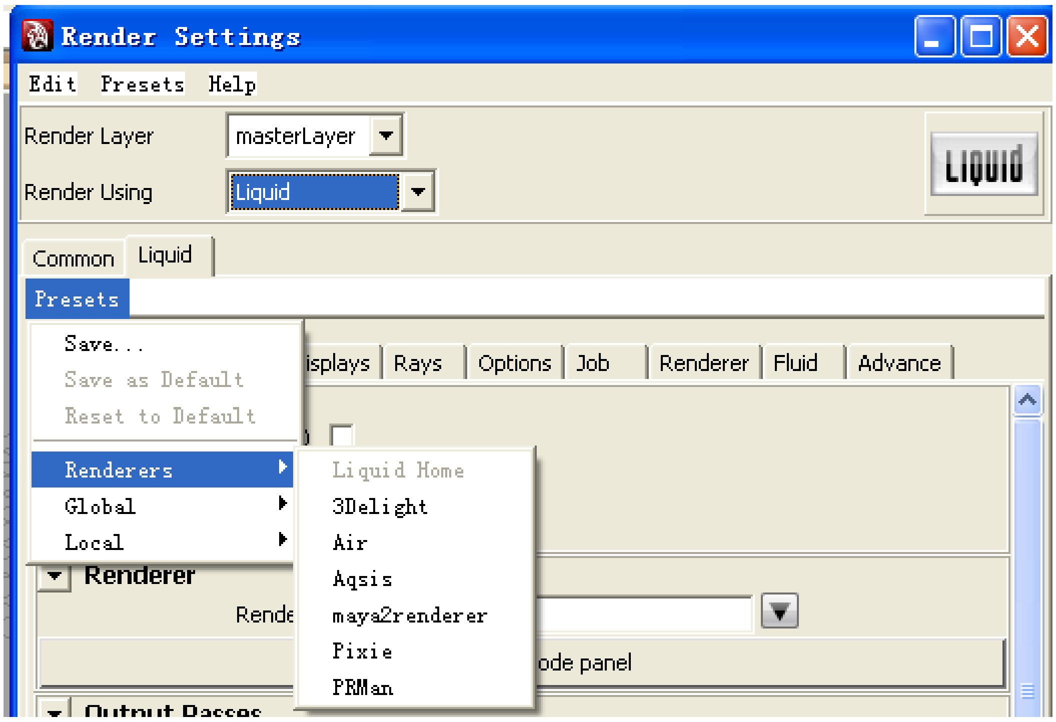
5)



文件名(N): maya2renderer.lg

文件类型(T): Filtered Files [*.lg]

(you can select *.lg in this way:

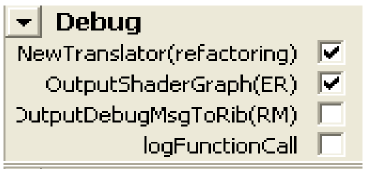


)

-RenderSettings of maya2renderer

Demo scene is ($liquid_root)/2.3.3/doc/HowToUseMaya2Renderer.ma

-Debug



Debug
NewTranslator(refactoring) ✓
OutputShaderGraph(ER) ✓
OutputDebugMsgToRib(RM)
logFunctionCall

1. NewTranslator(refactoring)
   This is a new feature of maya2renderer.
   I'm refactoring liquidMaya, and put the new export process into liqRibTranslator::_doItNew();
   The original export process is put into liqRibTranslator::_doIt();
   If NewTranslator(refactoring) is checked, liqRibTranslator::_doItNew() is executed, otherwise liqRibTranslator::_doIt() is executed.
   If you set NewTranslator(refactoring) to true, you are using maya2renderer,
   If you set NewTranslator(refactoring) to false, you are using liquidMaya,

2. OutputShaderGraph(ER)
   This is a new feature of maya2renderer.

In liquidMaya you MUST export the material of the mesh before you render that mesh.
In maya2Renderer, if you set OutputShaderGraph(ER) to true, you don't have to convert the material manually. The materials will be converted automatically.

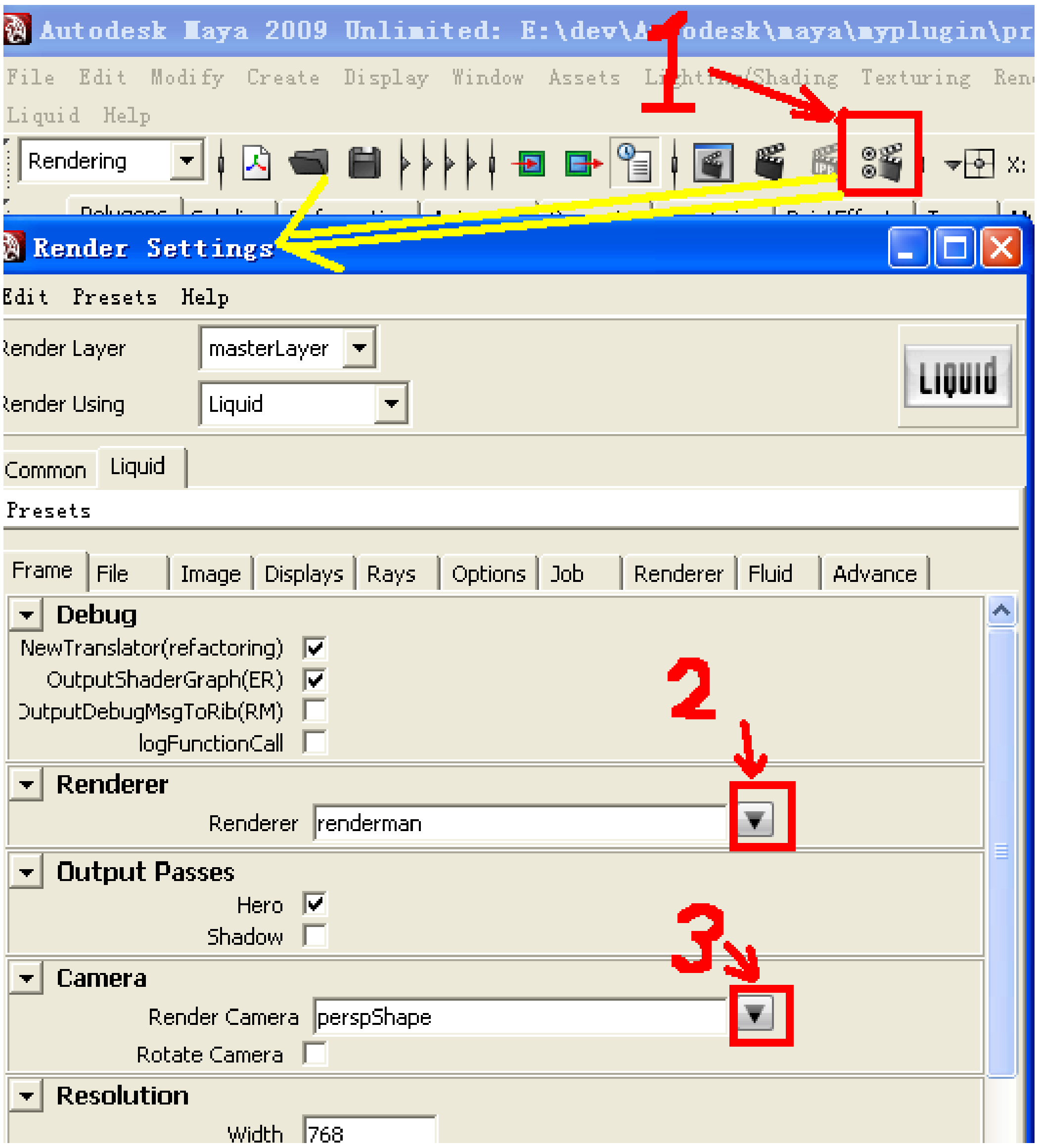
3. OutputDebugMsgToRib(RM)

   This is a new feature of maya2renderer. _liqRIBMsg(const char* msg) will output the msg to rib file.

4. logFunctionCall

This is a new feature of maya2renderer. _logFunctionCall(const char* msg) will output the msg. It is useful to trace the call stack in batchrender mode.

7) select "Renderer" and "Camera" in "Render Settings"
"Renderer" MUST be setted to "renderman" in liquidMaya,(in maya2renderer, you can set the "Renderer" to another renderer, e.g. elvishray)


Autodesk Maya 2009 Unlimited: E:\dev\Autodesk\maya\myplugin\pr
File Edit Modify Create Display Window Assets Lighting/Shading Texturing Ren
Liquid Help
Rendering
1
Render Settings
Edit Presets Help
LIQUID
Common Liquid
Presets
Frame File Image Displays Rays Options Job Renderer Fluid Advance
2
3

Render Layer masterLayer
Render Using Liquid

Debug
NewTranslator(refactoring) ☑
OutputShaderGraph(ER) ☑
OutputDebugMsgToRib(RM) ☐
logFunctionCall ☐

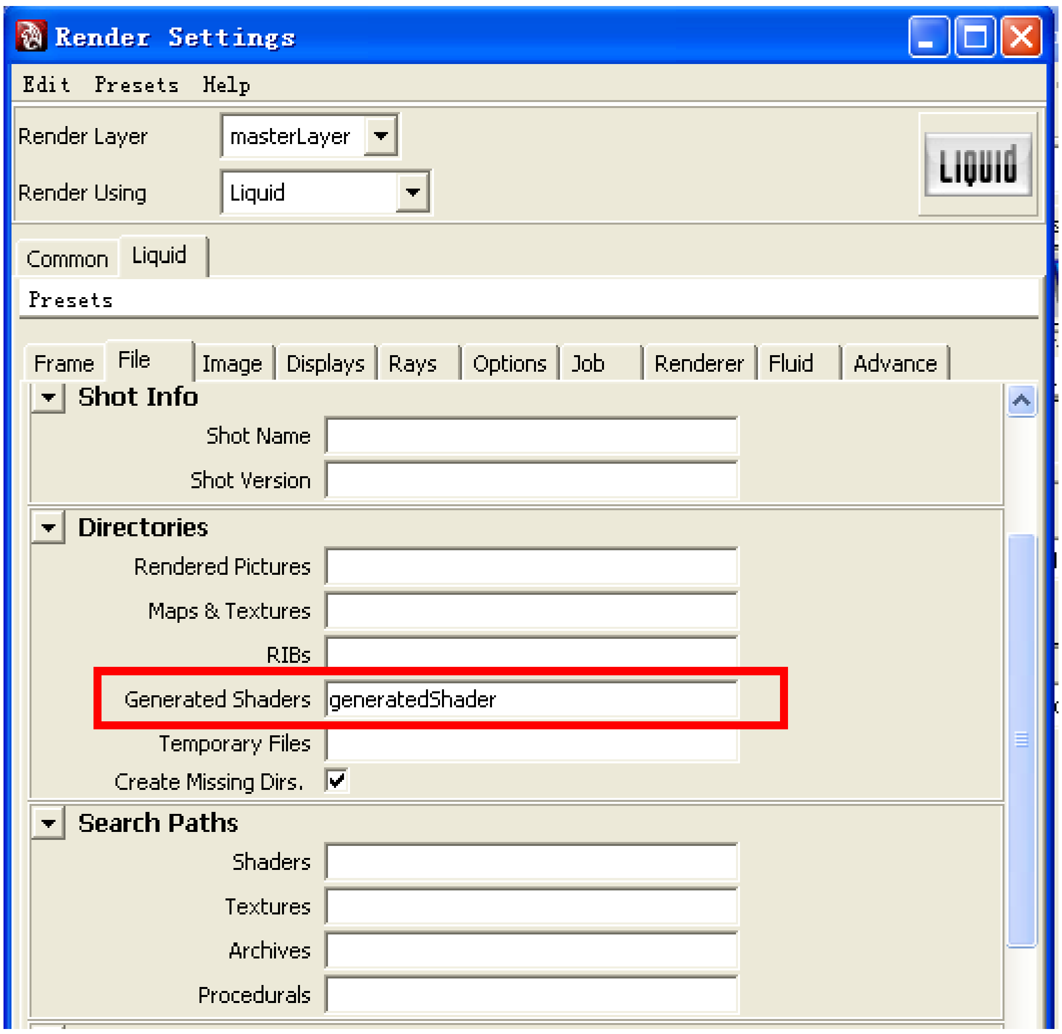
Renderer
Renderer renderman

Output Passes
Hero ☑
Shadow ☐

Camera
Render Camera perspShape
Rotate Camera ☐

Resolution
Width 768

8) set GeneratedShaders directory



Render Layer: masterLayer
Render Using: Liquid

Shot Info

Shot Name:
Shot Version:

Directories

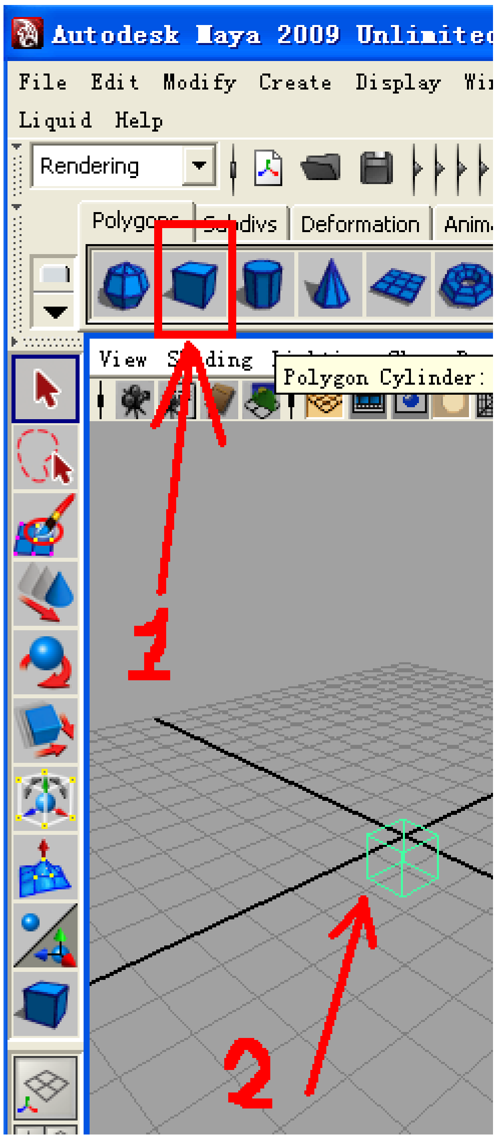
Rendered Pictures:
Maps & Textures:
RIBs:
Generated Shaders: generatedShader
Temporary Files:
Create Missing Dirs. ✓

Search Paths

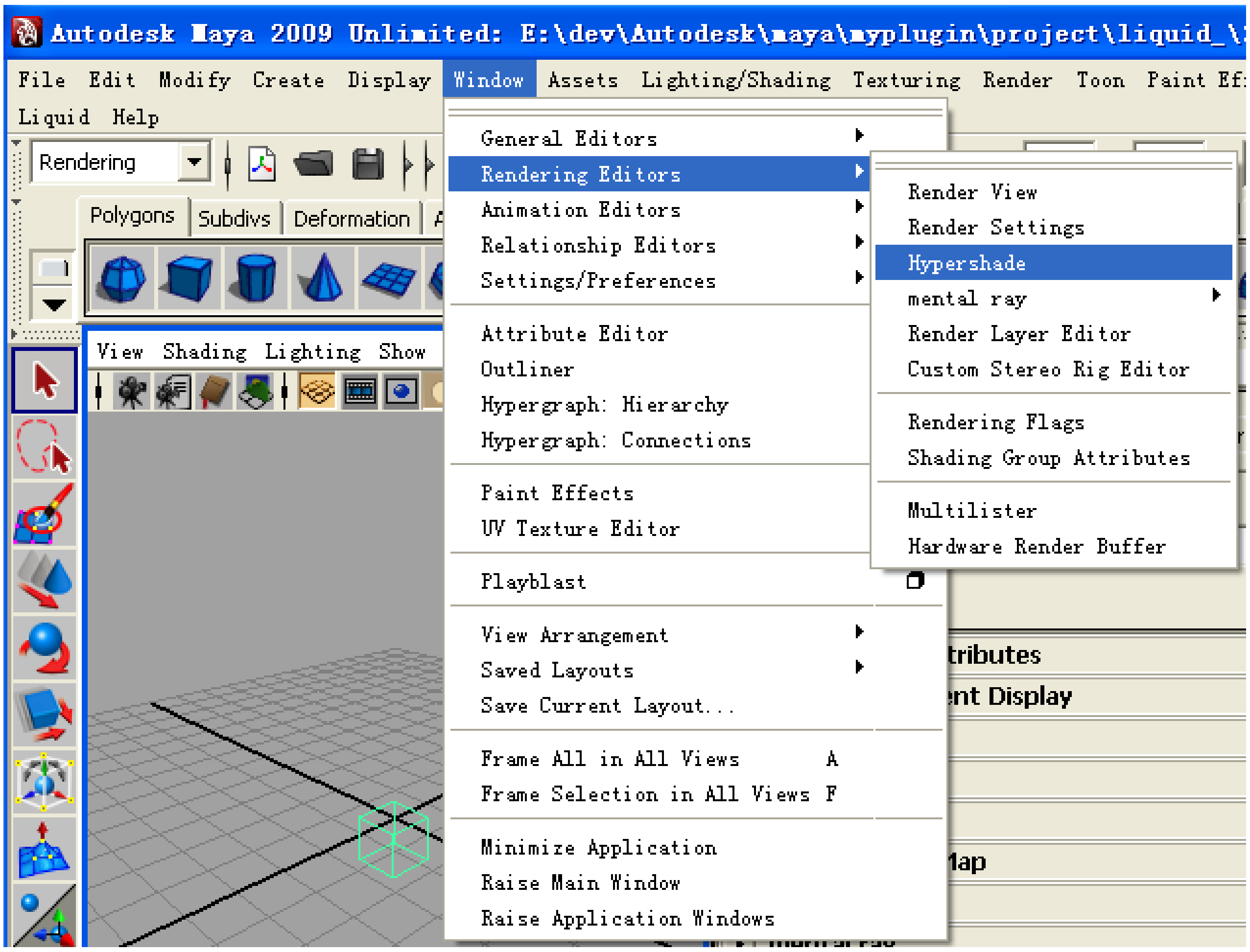
Shaders:
Textures:
Archives:
Procedurals:

-Create the scene

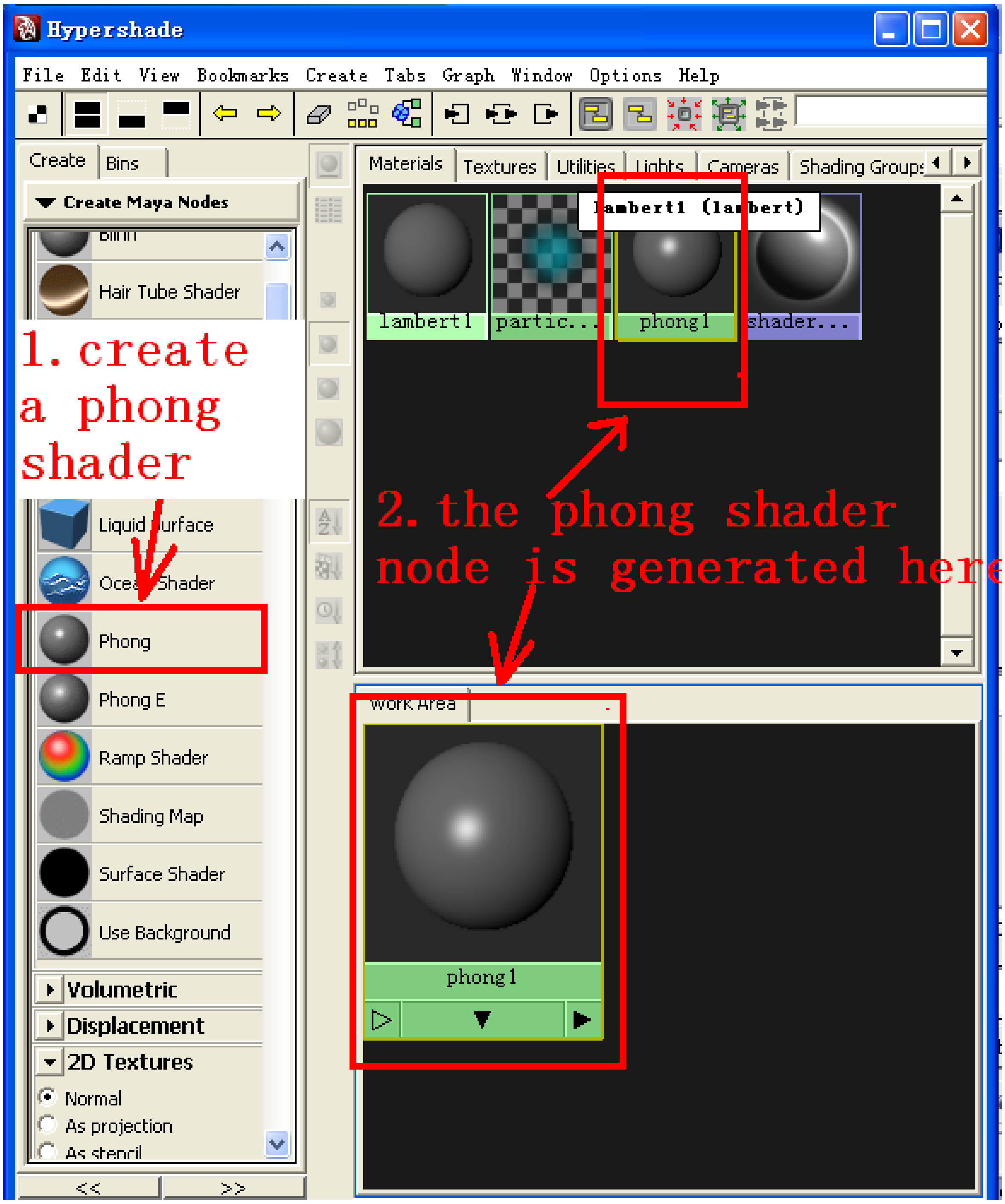
1.create a box


Autodesk Maya 2009 Unlimited
File Edit Modify Create Display Wi
Liquid Help
Rendering
Polygons Subdivs Deformation Anim
View Shading
Polygon Cylinder:
1
2

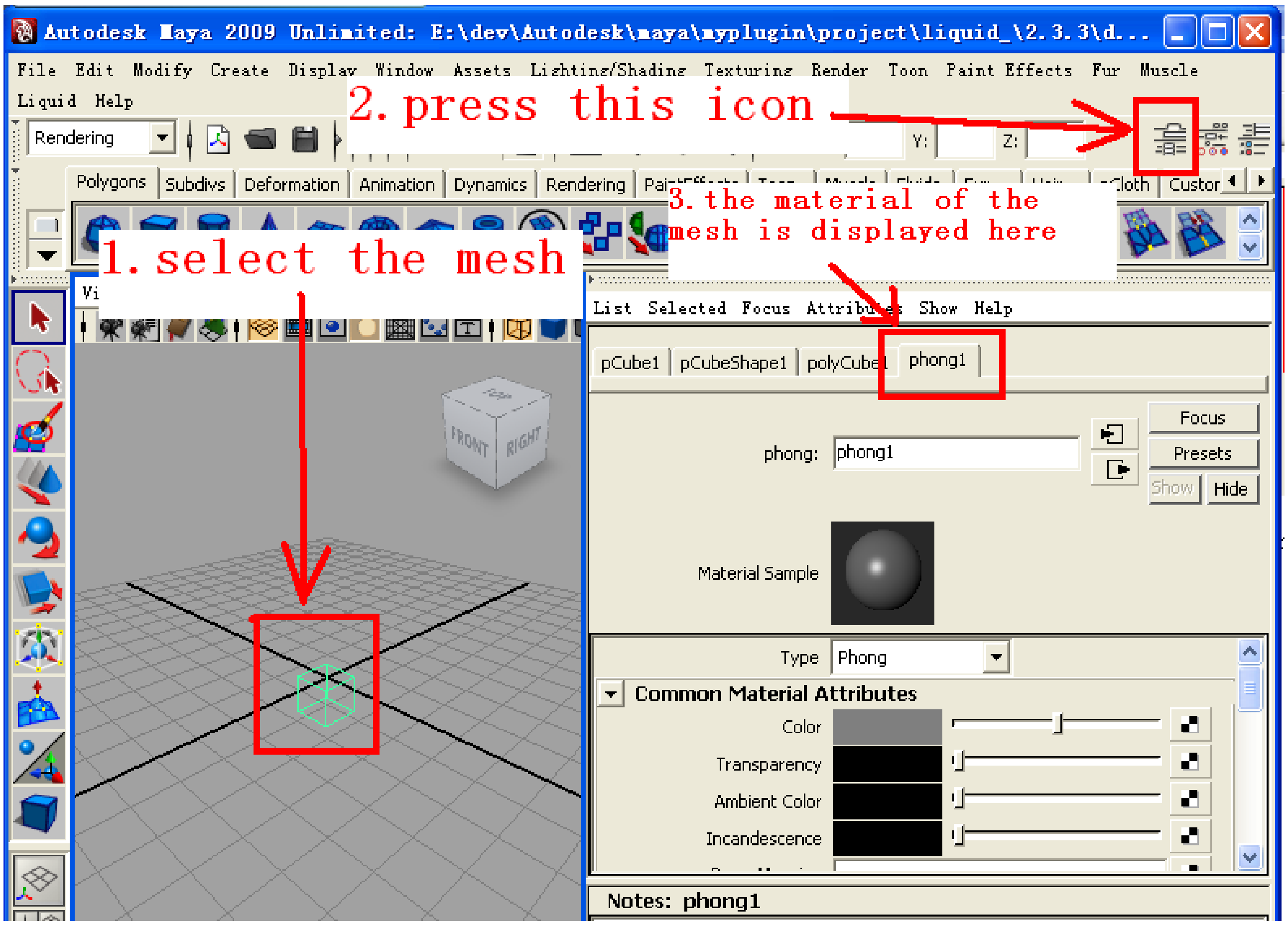
3. assigned a material to this box



4. let's create a phong shader

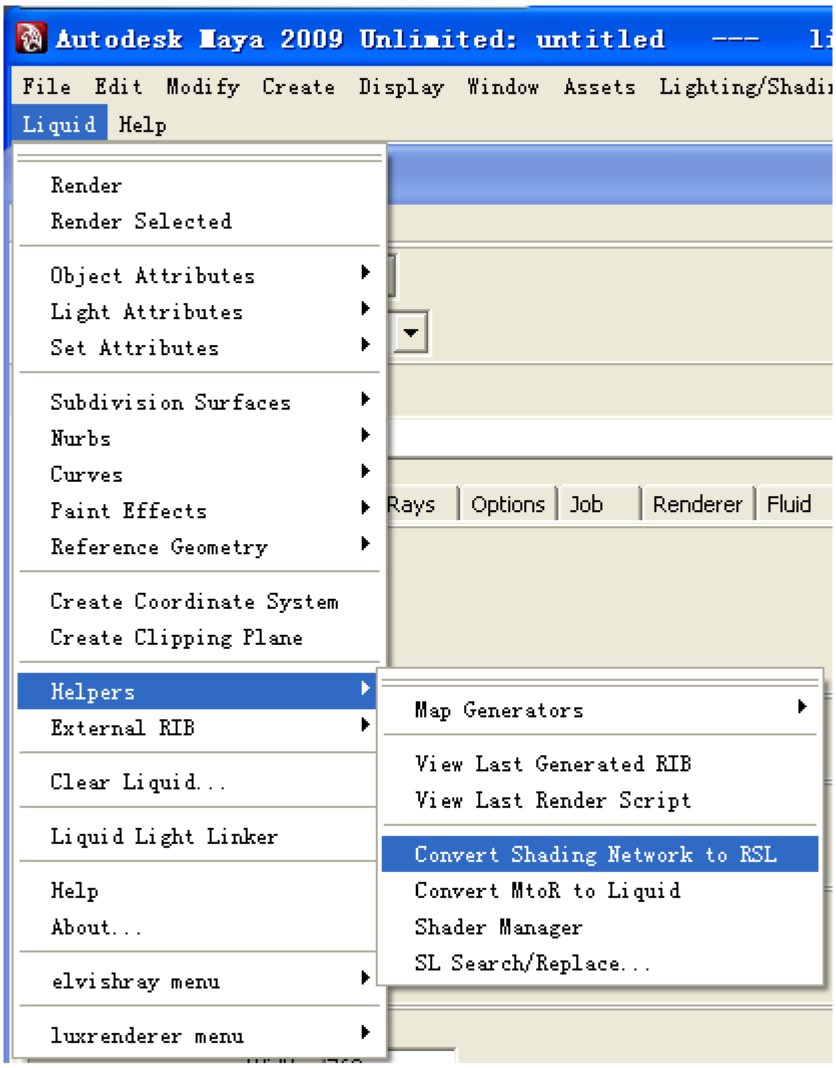


5.select the mesh, then press(not click) the right mouse button on the phong1 node, and select "Assign Material To Selectedion."
6.you can see the phong1 is assigned to the mesh.


Autodesk Maya 2009 Unlimited: E:\dev\Autodesk\maya\myplugin\project\liquid_\2.3.3\d...
File Edit Modify Create Display Window Assets Lighting/Shading Texturing Render Toon Paint Effects Fur Muscle
Liquid Help
Rendering
2. press this icon
Polygons Subdivs Deformation Animation Dynamics Rendering
1. select the mesh
3. the material of the mesh is displayed here
List Selected Focus Attributes Show Help
pCube1 pCubeShape1 polyCube1 phong1
phong: phong1
Focus
Presets
Show Hide
Material Sample
Type Phong
Common Material Attributes
Color
Transparency
Ambient Color
Incandescence
Notes: phong1

-export the material of the mesh

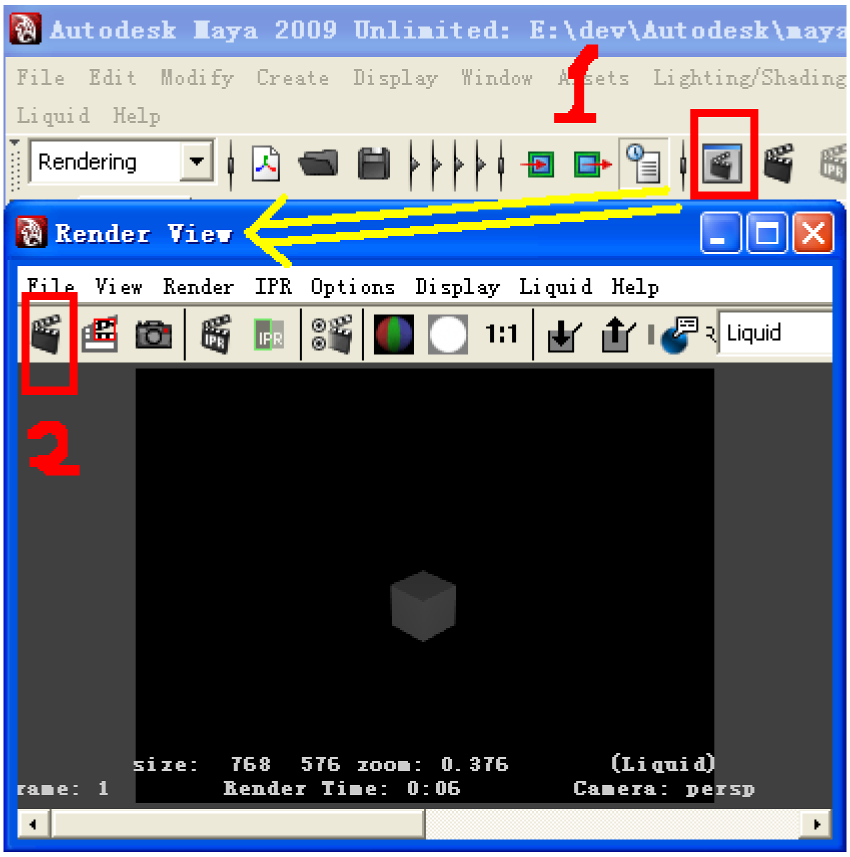
In liquidMaya you MUST export the material of the mesh before you render that mesh.This is the steps:
- make sure "GeneratedShaders" directory is set.
- select the mesh which you want to render
- Menu --> liquid -->Helpers --> Convert shading network to RSL:



phong1.sl and phong1.slo will be generated at E:\MyDocuments\maya\projects\default\generatedShader\

-Render

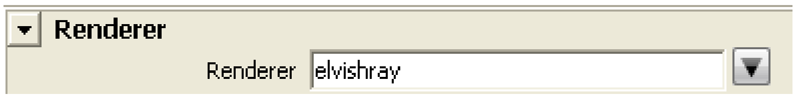
Press the "RenderView" button, and press "Render" button:


Autodesk Maya 2009 Unlimited: E:\dev\Autodesk\maya
File Edit Modify Create Display Window Assets Lighting/Shading
Liquid Help
Rendering
1
Render View
File View Render IPR Options Display Liquid Help
1:1
Liquid
2
size: 768 576 zoom: 0.376 (Liquid)
rame: 1 Render Time: 0:06 Camera: persp

The rib file will be generated at E:\MyDocuments\maya\projects\default\rib, (e.g. E:\MyDocuments\maya\projects\default\rib_perspShape.0001.rib)

-Renderer

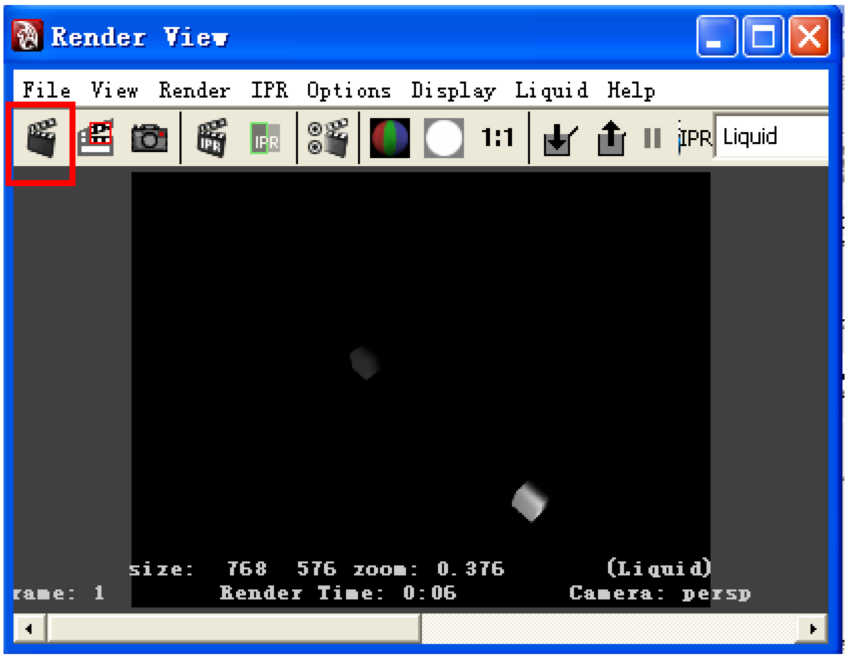
You can select other renderers in maya2renderer, e.g. elvisyray.



Renderer

Renderer elvishray

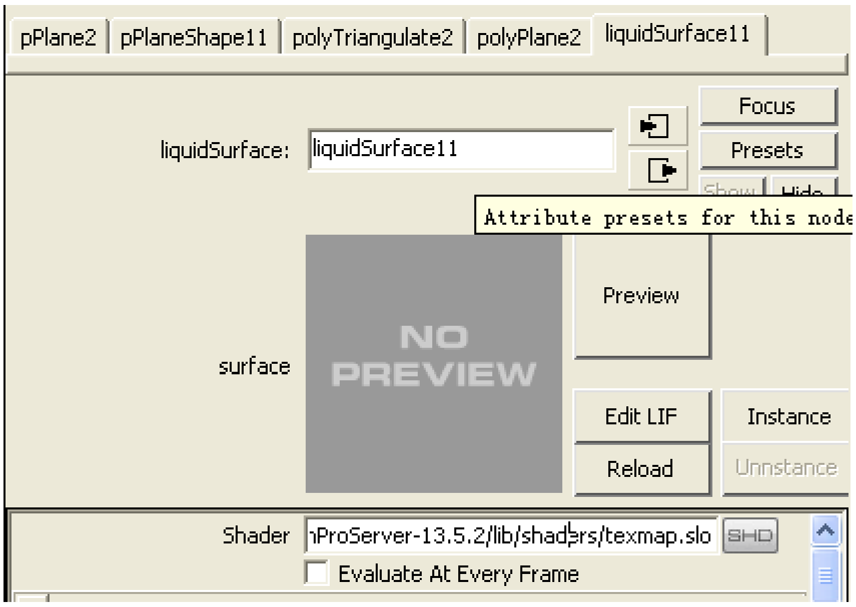
Render


Render View
File View Render IPR Options Display Liquid Help
IPR Liquid
size: 768 576 zoom: 0.376 (Liquid)
rame: 1 Render Time: 0:06 Camera: persp

It will generate file:///<MayaProjectDirectory>\rib\<SceneName>.er which trace the invoking of ERAPI

- liquidSurfaceShader

– In liquidMaya, liquidSurfaceShader is used for user-defined surface shader(it is *.slo for prman)



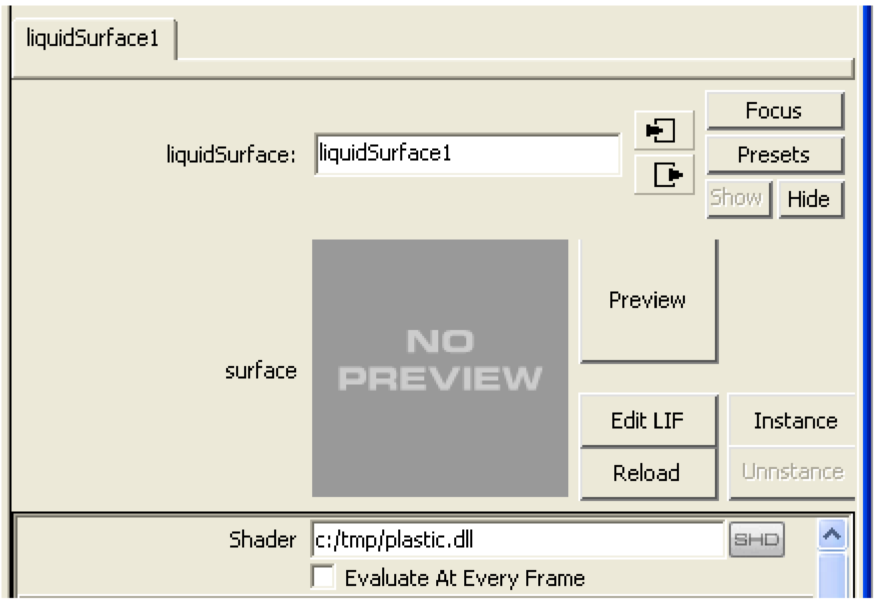
liquidSurface: liquidSurface11

surface

Shader ıProServer-13.5.2/lib/shaders/texmap.slo

Evaluate At Every Frame

– In maya2renderer, liquidSurfaceShader is extended to supported user-defined shader for other renderers.



liquidSurface: liquidSurface1
surface
Shader c:/tmp/plastic.dll
Evaluate At Every Frame

If you use prman, you set the Shader to a slo file, and don't have to do anyting else.

If you use elvishray, e.g. In liquidSurface1, if you set Shader to c:/tmp/plastic.dll, you **have to create** c:/tmp/plastic.pl. Because maya2renderer will search c:/tmp/plastic.pl, and read this pl file to generate shader parameters automatically.

- PL file

pl file contains shader type, shader parameter list.
pl file must locates at the same directory as the shader file. For elvishray, the shader file is a dll file.

pl file is used to generate the UI of parameters of user-defined shader automatically in Maya. The following image show this process in details

%liquidRoot%\2.3.3\doc\HowToGenerateShaderParameterUI\HowToGenerateShaderParameterUI.png

or
https://github.com/maya2renderer/maya2renderer/blob/master/2.3.3/doc/HowToGenerateShaderParameterUI/HowToGenerateShaderParameterUI.png

Here is an example of pl file:

```
#shaderType // shaderType: surface, volume, displacement, light. shaderType must be set before shadername
surface
#shaderName //shadername must be set after shaderType
liquidchecker
//shader parameter list. One line for each parameter,
#Name:Details:Type:IsOutput:Defaults:ArraySizes//Attributes of each parameter, and the attributes are seperated by colon.
frequency:uniform:float:0:8.5:-1
mode:uniform:float:0:0.0:-1
mode2:uniform:float:0:0.5:-1
```

- Refactoring

- Global variables

Put global variables to struct liqGlobalVariable(in liqGlobalVariable.h)
define struct liqGlobalVariable liqglo(in liqGlobalVariable.cpp)

- main atchitecture

See ./main_architecture.jpg (https://github.com/maya2renderer/maya2renderer/blob/master/2.3.3/doc/main_architecture.jpg). I list only a few classes in maya2renderer in this diagram. The main translation process is shown in ./main_translation_process.jpg(https://github.com/maya2renderer/maya2renderer/blob/master/2.3.3/doc/main_translation_process.jpg).

Class RendererInterface
This class provides basic interfaces for your renderer. The interfaces of this class are fat interface, and will be refactored in the future.
If you want to add your new renderer, e.g. appleseed, into maya2renderer, you should define namespace appleseed, and let class appleseed::Renderer inherite from liquid::RendererInterface

```
namespace appleseed{
      class  Renderer : public liquid::RendererInterface{
      };
}
```

ShaderOutputVisitor
I use the visitor pattern to translate the internal maya shader nodes for the new renderer. For example, ShaderOutputVisitor provides the following abstract interfaces: visitBlinn(), visitLambert(), visitPhong() and etc.

RSL::Visitor inherits from ShaderOutputVisitor, RSL::Visitor visits each shader node and output the renderman rsl shader to a sl file. I rename the extention of this sl file to sl_my. sl_my file can be compile by renderman shader compiler. RSL::Visitor use RSL::OutputHelper to add parameters of the shader.

ERCall::Visitor is responsible for constructing shader for elvishray from maya shader node. ER::Visitor uses **ERCall::OutputHelper** to add shader parameters.

ER:Visitor and **ER::OutputHelper** are designed to output the shader into a log file, so I can check whether there is something wrong.

See ./ExportShaderGraph.jpg(https://github.com/maya2renderer/maya2renderer/blob/master/2.3.3/doc/ExportShaderGraph.jpg) for more details.

Class tHeroRibWriterMgr is refactored from the liquidMaya::liqRibTranslator.

tHeroRibWriterMgr::write() contains the translation process of the hero pass. See ExportHeroPass.jpg (https://github.com/maya2renderer/maya2renderer/blob/master/2.3.3/doc/ExportHeroPass.jpg)for more details.

Class tShadowRibWriterMgr is refactored from the liquidMaya::liqRibTranslator. tShadowRibWriterMgr::write() contains the translation process of the shadow pass. See ExportShadowPass.jpg(https://github.com/maya2renderer/maya2renderer/blob/master/2.3.3/doc/ExportShadowPass.jpg) for more details.

- NOTE

Maya2renderer doesn't provide API for your renderer, because the general framework is not stable enough now. Many interfaces need to be refactored.

- export shader graph

In liquidMaya, you have to convert the shader graph manully before you render the scene.

In maya2renderer, you don't have to convert the shader graph to manully. The shader graph will be exported in each frame.

In order to translate maya shader graph for other renderers, I rewrite the following mel files with c++:
liquidConvertShadingNetworkToRSL.mel,
liquidWriteRSL_materials.mel,
liquidWriteRSL_textures.mel,
liquidWriteRSL_utilities.mel

shader graph must be exported before exporting mesh.

ShaderValidConnection::setValidConnection() set the connectable plugs in each shader. e.g.

```
//lambert
validConnection.clear();

validConnection.append("color");
validConnection.append("transparency");
validConnection.append("ambientColor");
validConnection.append("incandescence");
validConnection.append("diffuse");
validConnection.append("outColor");
validConnection.append("outTransparency");
validConnectionMap.insert(std::make_pair("lambert", validConnection));
```

The following images show more details:

./ExportShaderGraph.jpg
(https://github.com/maya2renderer/maya2renderer/blob/master/2.3.3/doc/ExportShaderGraph.jpg)
./How_to_output_a_ER_shader.jpg
(https://github.com/maya2renderer/maya2renderer/blob/master/2.3.3/doc/How_to_output_a_ER_shader.jpg)
./How_to_output_a_RSL_shader.jpg
(https://github.com/maya2renderer/maya2renderer/blob/master/2.3.3/doc/How_to_output_a_RSL_shader.jpg)
./convertShadingNetworkToRSL_er.jpg
(https://github.com/maya2renderer/maya2renderer/blob/master/2.3.3/doc/convertShadingNetworkToRSL_er.jpg)
./convertShadingNetworkToRSL_rsl.jpg
(https://github.com/maya2renderer/maya2renderer/blob/master/2.3.3/doc/convertShadingNetworkToRSL_rsl.jpg)

- export user-defined shader

The following images show more details:
./user-defined_shader_er.jpg
(https://github.com/maya2renderer/maya2renderer/blob/master/2.3.3/doc/user-defined_shader_er.jpg),
./user-defined_shader_rm.jpg
(https://github.com/maya2renderer/maya2renderer/blob/master/2.3.3/doc/user-defined_shader_rm.jpg)

- export mesh

todo

- export light

todo

- log system

Mel log

trace log

Shadow pass is refatored from liquidMaya, and is not extended for elvishray.

Each test case contains *.rib ,*.er, *.log, they are logs for that test case.

- TodoList

Some interfaces in class Renderer are fat interface, they will be refactored.

%liquidroot%/todolist.txt

- How to add a new renderer to maya2renderer

todo