Kerkythea Messaging System

Introduction – XML File Short Description

The XML file used by Kerkythea contains the complete hierarchically organized data structure of the complete Kerkythea parameterization. The file describes the complete hierarchy of object/subobjects including the parameters for each object.

When an object is given in the XML file, the object is described with the following four strings:
`Identifier` – This string is closely related to the name of the object, it's the actual string used to identify the object within the complete hierarchy. This means that it may contain the character '/' to show that the object is located deeply in the tree structure or as in most cases will simply match the object name.
`Label` – This is the label of the object; this string will be used to search in the K plugins and find the appropriate one. If the right plugin is found, an object of this type is created, and is initialized according to the parameterization followed in the file.
`Name` – The actual name of the created object – can be any user name.
`Type` – The generic category where the objects falls in.

On the other hand, when a parameter is given this is described by the following three strings:
`Name` – The name to access the parameter; this must be a string that can be identified by the developed object.
`Type` – The type of the parameter; this is used to make the appropriate conversion for the Value string. The most used categories for this are: "Boolean", "Integer", "Real", "String". The correct type MUST always be provided for the correct interpretation of the value.
`Value` – The actual value of the object in a string format.

XML Description of Triangular Meshes

The triangular meshes in Kerkythea have a very compact description inside the XML file. They are indexed meshes, meaning that the vertex list is given separately from the triangle list, where in the latter, triangles are described by just providing the integer indices of their vertices in the list.
A triangular mesh object is initially described with the *Object* keyword, followed by the object parameters, i.e. identifier, label, name and type, all of them in a string format with the label required to be *`Triangular Mesh`* and the type equal to *`Surface`*. The first thing after that, is a vertex list which is described as parameter with the attributes, name equal to *`Vertex List`*, type equal to *`Point3D List`* and then the value of total number of vertices in the mesh. The list of vertices must follow with the position of described in a single string containing 3 float numbers.
After the vertex list description, an optional normal list may be provided with the attributes, name equal to *`Normal List`*, type equal to *`Point3D List`* and then the number of normal vectors. Kerkythea can work both with normal description per vertex and per

triangle (but not mixed). Thus, the number of given normal vectors can be either equal to the number of vertices in the mesh (normal per vertex) or equal to the number of triangles multiplied by 3 (normal per triangle); the application will automatically make the correct handling of the normal list.
Right afterwards, the triangle list is given with the attributes, name equal to *`Index List`*, type equal to *`Triangle Index List`* and then in value the total number of triangles in the list. The list of indices followed described in the form of strings containing 3 integer numbers. These numbers are the 0-based indices of the positions in the first list (vertex list), thus they cannot be greater-equal to the number of vertices.
An optional Boolean parameter at the end with the name *`Smooth`* defines whether the mesh appears smooth (i.e. makes use or re-computes the normal vectors) or flat.

```
<Object Identifier="Triangular Mesh" Label="Triangular Mesh" Name="" Type="Surface">
        <Parameter Name="Vertex List" Type="Point3D List" Value="8">
                <P xyz="5 -3.54243 0.0164385"/>
                …
        </Parameter>
        <Parameter Name="Normal List" Type="Point3D List" Value="36">
                <P xyz="0 0 -1"/>
                …
        </Parameter>
        <Parameter Name="Index List" Type="Triangle Index List" Value="12">
                <F ijk="0 1 2"/>
                …
        </Parameter>
        <Parameter Name="Smooth" Type="Boolean" Value="1"/>
</Object>
```

XML Code Snapshot from Triangular Mesh Description

Advanced Settings and XML File

In the "Advanced Settings" dialog, you can see the complete hierarchy of the objects and parameters of Kerkythea. This means that the "Advanced Settings" is another complete description of the parameterization and it thus closely related to the XML file description. In fact, one can find all the parameters taken place in the XML file except for certain data that can not be easily viewed (like mesh data).
The "Advanced Settings" can also be used for testing some of the messages of Kerkythea. Thus, left-clicking on an object will bring up its parameters (on the right pane), while right clicking on an object will pop up a menu of the supported messages (or a part of the messages – some of them needing additional information will not appear in that menu).

Description of the Supported Messages

The messaging system can be invoked directly from the script files used by Kerkythea. The script files give the ability not only to address messages to Kerkythea kernel but also to set the parameters in the hierarchy.
One important thing about the messages and parameters is that they can only be addressed if they are given in the appropriate "path" in the hierarchy. This can be done by

using the special strings “./” (to begin a path) and “/” (to branch to another path), similarly to a folder/files structure.

Here is the supported message list. Note, that in order for the message to work, it should be included in “” marks and placed after a message command, for example Load myscene.xml -> message “Load myscene.xml”.

Load myscene.3ds

Causes the kernel to load a scene. Simple ☺

Merge scene2.xml #model #light #camera #rendersettings #globalsettings

Causes the kernel to open given scene and perform merging according to the integers included in the message. The integers have the following meaning:
#any = 0 : Keep old settings
#any = 1 : Replace old settings
#any = 2 : Merge (add) settings
#any = 3 : Merge settings but in case of name collision keep the old ones
#any = 4 : Merge settings but in case of name collision replace with the new ones
#any = 5 : Merge settings but in case of name collision keep the old ones and replace with new coordinates (mesh/surface and coordinate frame position).
#any = 6 : Merge settings but in case of name collision keep the old ones and replace with material and general appearance settings.

All the values make sense for the model, light, camera but only the values 0, 1 make sense for the render and global settings.
For example we could issue the following merging, if we want to replace with new materials, merge cameras & lights, but keep old render and global settings.
message “Merge scene2.xml 6 3 3 0 0”

Save newscene.xml

Causes the kernel to save the complete current parameterization (note that at the current moment, only xml extension is supported).

Render

Causes the kernel to begin rendering, applying also filters at the end.

SaveImage output.png

Probably after a render job, this message will save the final image on disk.

GenerateSun #lon #lat #zone #date #time

This message is handled by a scene sub-object, meaning that the complete scene path should precede the GenerateSky keyword. The #lon and #lat are real parameters indicating the longitude and latitude in decimal degrees of the place of interest, while #zone corresponds to Greenwich time zone of the place in the form GT followed by the integer zone offset. The date should be given in dd-mm-yyyy format, while the time should be given in hh:mm:ss format.

Assuming that we have loaded a scene called “Paris”, here is an example to generate the sun in one hour past noon, mid August:
message “./Scenes/Paris/GenerateSun 48.82 2.48 GT+1 15-08-2006 13:00:00”
Note, during the generation of sun, a point light source is created with the name “sun”. If a light already exists with that name then it is replaced with the new parameters.

Lock Cache
Reset Cache

These two messages are often used in various Kerkythea modules to signify that various cached data should be reused instead of reinitialized (which may involve heavy computations). This is especially true for generated photon maps and irradiance caches where reusing these data may decrease considerably the rendering times without affecting the quality.
Here are some example messages:
message “./Irradiance Estimators/Density Estimation/Lock Cache”
message “./Irradiance Estimators/Diffuse Interreflection/Lock Cache”
message “./Environments/Octree Environment/Lock Cache”
Note that the user becomes responsible after locking these data, to unlock them (by using the “Reset Cache” message) when they become invalidated. This will happen when certain changes are made to the scene (for example moving an object or changing the light position).

About

Application	Kerkythea© Rendering System
Version	1.02 (Copyright 2004-2006)
Author	Ioannis Pantazopoulos
Web	http://www.softlab.ntua.gr/~jpanta/Graphics/Kerkythea
Contact	jpanta@softlab.ntua.gr