Simulated Railroad Framework, http://simulrr.sourceforge.net
Synopsis: 000_Synopsis

This file valid for step 0033.10
Issue Date: 2017-03-17

Naming Rules, MMF Paradigm

1 Synopsis

The Naming Rules and the MMF Paradigm are *the* concepts of SMS. Everybody, who develops Simple Multiuser Scenes, e.g. using the SMUOS Framework or using the SRR Framework, should be aware of the Naming Rules and of the MMF Paradigm.

2 Purpose of Naming Rules and MMF Paradigm

Parts of Simple Multiuser Scenes are provided and used in a very "distributed" manner.

On the one hand, SMS deals with composite scenes. The parts of an SMS (we call them "facilities") may derive from several distributed resources. Many authors may contribute to Simple Multiuser Scenes, e.g. model authors or module authors, where they may provide their resources via different channels, e.g. local storage, FTP servers, web servers or even data bases and other means of static or dynamic content provisioning.

On the other hand, SMS deals with multiuser sessions, where each instance of the scene must be synchronized with all other instances of the scene.

So it is necessary that each part of the scene shall get

- a name to identify the network traffic (together with transport addresses)
- which is different from the names of the other parts of the scene.

Having names to identify the network traffic could easily be achieved by using hardcoded names that are provided by the authors, e.g. by model authors.

However, this would not preclude using e.g. two models from two different authors and those authors using the same names for their models unintentionally.

So the requirement to have composite scenes and the requirement of re-usability (which are the only reasons to have standards like VRML and X3D), force us to require some architectural pre-conditions:

- an SMS shall consist of one frame*),
- of at least one module*) and
- of models that inhabit the modules.

So the MMF Paradigm can be concisely depicted as follows:

frame – 1:n – module – 1:n – model

The frame is responsible that each module gets a unique module name and each module is responsible that it's models get unique object IDs.

*) The terms "frame" and "module" are derived from model railroading, where modules from different model railroaders can be used to compile complete layouts, which can then be used to really play with. The frame is the collection of common facilities that are used by all modules (e.g. the power supply).

3 External View

3.1 Module Names and UOC Names

A scene consists of one or more modules that can be loaded/unloaded independently. The modules of a scene are identified by unique string identifiers, the so-called module names.

Basically, objects are children of modules. However, some objects can change the module during their lifetime(*not yet implemented*) or they can exist outside of any module. Such objects need an additional identifier instead of the module name, we call this identifier the "universal object class name" (UOC name).

Module names and UOC names are strings that are built by the characters 'A'-'Z', 'a'-'z', '0'-'9' and '_'.

The UOC name "Base" is strictly reserved for use by the SMUOS Framework, the example SSC extension "Key Manager" uses the UOC name "Keys" (this can be superseded by the frame author) and the example SSC extension "Train Manager" (*not yet implemented*) will probably use the UOC names "RailVehicles" and "Trains".

3.2 Object IDs

Within a module (or within a UOC), each object must be distinguished from each other. Object identifiers are strings that are unique within a module (or within a UOC).

Object IDs are strings, that are built by the characters 'A'-'Z', 'a'-'z', '0'-'9', '_' and '.' *).

*) An object that is contained in a parent object, inherits a part of its <objId> from the parent object, i.e. <objId>=<parentObjId>.<objLocalId>.

3.3 Extended Object IDs

The extended object ID is used in the tracer (to identify the tracer instances) and as a part of the identifiers within network traffic (which are called stream names in case of the SMUOS Framework).

The extended object ID does not change during the lifetime of an object.

Bound objects (cannot change module):
<extObjId>=<moduleName>-<objId>
Unbound objects/Astral objects (can change modules or exist outside of modules):
<extObjId>=<uocName>-<objId>

The SSC Base (with the UOC name "Base") contains currently two objects with reserved object Ids, so following extended object IDs are reserved:
- "Base-SimpleSceneController" (which is not an object, strictly spoken, but the stream name is reserved, see next chapter)
- "Base-DefAvaCon"....the default avatar container (an astral object)

The "Key Manager" extension reserves the extended object Id "Keys-SimpleSceneController", where "Keys" could be superseded by the frame author with another string. Also this is not an object, but the stream name is reserved.

It's the same story with the "Train Manager" extension, which will probably reserve the extended object Id "Trains-SrrController".

3.4 Stream Names

The Simple Scene Controller (Base) uses the stream name "Sms-Base-SimpleSceneController-Master".

The SSC extension "Key Manager" uses the stream name "Sms-Keys-SimpleSceneController-Master" (where "Keys" could be superseded by the frame author with another string).

The SSC extension "Train Manager" uses the stream name "Sms-Trains-SrrController-Master" (where "Trains" could be superseded by the frame author with another string).

The SSC Dispatchers for the modules and for the UOCs use stream names "Sms-<moduleName>" and "Sms-<uocName>", respectively.

The example MIDAS Objects use stream names "Sms-<extObjId>-<user>-<suffix>", where <user> equals "Mib" for network sensors that are part of the MIDAS Base, and <user> equals "Obj" for network sensors that are part of the example MIDAS Objects. <suffix> is used by the MIDAS Base or by the MIDAS Objects to make a distinction, if more than one network sensor is used for the same <extObjId> and <user>.

It is the intention of the prefix "Sms-" to keep the SRR/SMUOS Framework compatible with modules and models that use their own network sensors or that even use another network sensor dependent framework.

3.5 Key IDs

Keys are identified by unstructured character strings. Key IDs need not be unique (same key may exist twice or more often).

3.6 Parameter Names

Parameter names are used at the console interface, together with module names, UOC names, and object IDs.

Parameter names are strings that are built by the characters 'A'-'Z', 'a'-'z', '0'-'9' and '_'.

3.7 Well-Known Extension IDs

The SSC may be extended by more than one SSC Extension

Hence the users of the SSC Extensions (i.e. the Module Coordinator Extensions and the Extension MIDAS Objects) must somehow get the possibility to address the specific extensions, which they need to access.

On the other hand, the Module Coordinators of a layout/SMS may be extended by more than one Module Coordinator Extension each.

Hence the users of the Module Coordinator Extensions (i.e. the Extension MIDAS Objects) must be able to address their extensions explicitely.

These two problems are solved by introducing
- Well-known SSC Extension IDs
- Well-known Module Coordinator Extension IDs

One layout/SMS (precisely spoken, one frame) cannot host two different SSC Extensions that use the same well-known ID.

One module cannot host two different Module Coordinator Extensions that use the same well-known ID.

The example SMUOS extensions that accompany the SRR/SMUOS Framework, use following well-known IDs:

   Beamer Manager Extension:
      WKI = "SscBeamerManager" (SSC Extension)

   Key Manager Extension:
      WKI = "SscKeyManager" (SSC Extension)

   Train Manager Extension:
      WKI = "SscTrainManager" (SSC Extension)
      WKI = "ModCoordTrainManager" (Module Coordinator Extension)

4 Internal View

N/A

5 Additional Info

The MMF Paradigm and most of the naming rules will hold for the overall concept of SMS, however some of the naming rules will not survive the experimental SMUOS Framework. This is ffs.