Simulated Railroad Framework, http://simulrr.sourceforge.net
Synopsis: 100_SrrFramework

This file valid for step 0033.10.5
Issue Date: 2019-06-09

The SRR Controller (Train Manager) (*not finished* - might change completely)

1 Synopsis

The topics of "Trains" and of "Rail Vehicles" were considered to be too specific to handle them in the very general SMUOS Framework directly.

Furthermore, we decided to do a simplification in step 0033 of the project: we will only support "one-vehicle-trains" that means: we will not implement the UOCs "Trains" nor "RailVehicles", but only the UOC "VehicleTrains".

It was decided to implement an SSC Extension with following characteristics:

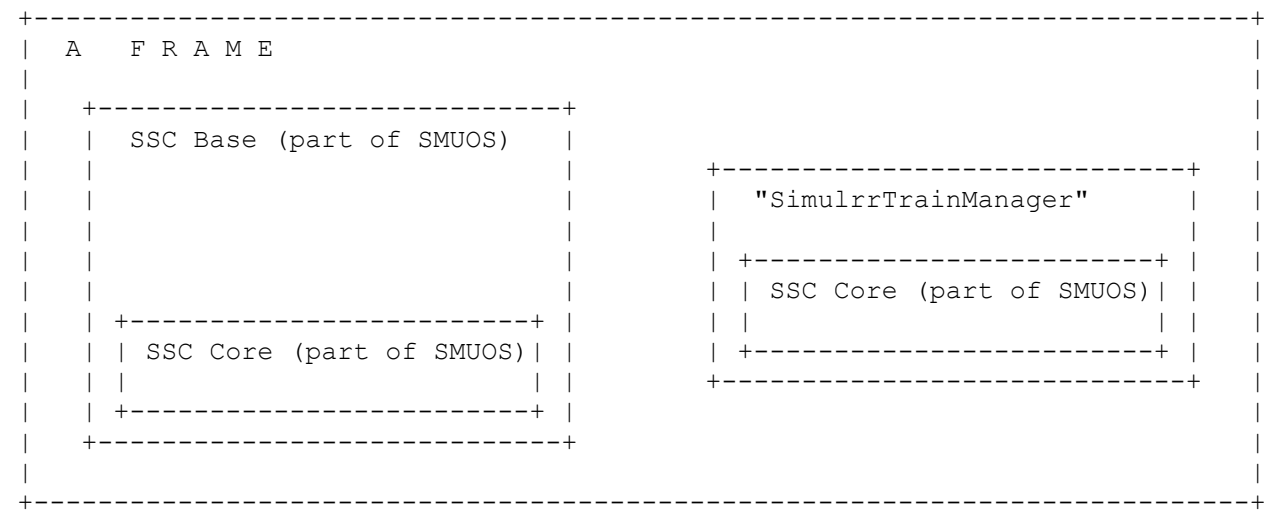
```
+-----------------+-------------------------------------------------------+
|   Name          |   Train Manager Extension                             |
+-----------------+-------------------------------------------------------+
|      refinedClassPaths = '"Ssc.Base"'                                   |
|      wellKnownId       =  "SimulrrTrainManager"                         |
|      noState           =  TRUE                                          |
|      networkSensors    =  none                                          |
|                         +------------+-------------------------------+  |
|      sscDispatchers    = |   Name     |   VehicleTrains               |  |
|                         +------------+-------------------------------+  |
|                         |      wellKnownUoc     = "VehicleTrains"    |  |
|                         |      sscParameters    =  NO                |  |
|                         |      uboLoader        =  YES               |  |
|                         +-----------------------------------------+  |
+-------------------------------------------------------------------------+
```

This SSC Extension should be accompanied by one Module Coordinator Extension (described at 221_ModuleCoordinatorTm), which should be the "Train Manager" Extension with WKI=SimulrrTrainManager, which supported the tracks and turnouts of the SrrTrains layout,

and by

some MIDAS Objects, which are called SRR Objects, because they are specific for Simulated Railroads (described at 401_TracksAndTurnouts and 402_Trains).

The SimulrrTrainManager SSC Extension can be used together with SSC Base in a frame, as follows, to enable the usage of the ModCoordTrainManager MC Extension and of SRR Objects:



The "Train Manager" Extension maintains a global state. It is implemented in the file BaseControlTm.x3d.

Info about SSC Base and SSC Core can be found at 121_SimpleSceneController, some general considerations about SMUOS Extensions at 051_Extensibility.

2 The Purpose of the SRR Controller (Train Manager)

The train manager extension enables the SRR/SMUOS Framework to support bound models of tracks and turnouts and to support unbound models of rail vehicles. Rail vehicles can be combined to trains (*rail vehicles and trains not yet implemented*).

The example SRR Objects that accompany the train manager extension, provide the support of the bound models for tracks and turnouts and the support of the models of rail vehicles (*rail vehicles and trains not yet implemented*).

The support for the bound models of tracks and turnouts is mainly located in the module coordinator (see 221_ModuleCoordinatorTm) and in the associated SRR Objects.

3 External View (uiControl, commParamExt and eiControl)

When the <ProtoInstance> "SscBase" and all <ProtoInstances> of the SSC Extensions have been loaded, then the Simple Scene Controller is in mode of operation "LOADED" (MOO "LOADED").

After some preparation, the frame sends the "init"=true event to the SSC Base and triggers the initialization of the whole Simple Scene Controller.

After successful or unsuccessful initialization the SSC Base outputs the SFNode event "commParam", that points to the common parameters. The common parameters are hold as fields of a <Script> node, which is contained in the SSC Base.

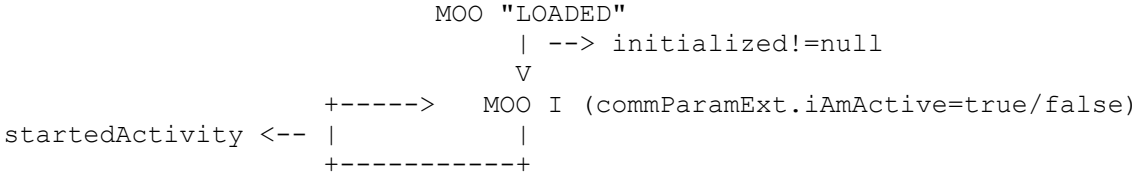
The "commParam" node contains the field "commParam.extensions", which can be used to access all successfully initialized SSC Extensions by any part of the scene.

3.1 MOO Diagram (MOOD)

SSC Extensions are initialized and activated together with the SSC Base.

Two phases of startup are relevant, the "initialization" and the "start activity" phases.

All this is coordinated by the SSC Core, when the correct fields of the external interface miControl are connected to the correct fields of the SscCore prototype (please refer to 121_SimpleSceneController, ch. 4.2 for details).



3.2 Use Case Matrix (uiControl)

Following use cases are available in following MOOs:

MOO "LOADED"	MOO I (inactive)	MOO I (activated)
Preparations	-	-
Initialization	-	-
-	StartActivity	-
SetTraceLevels	SetTraceLevels	SetTraceLevels

The State Event Matrix displays, which use case is available via the uiControl interface in which mode of operation (MOO) of the SSC Extension. An additional differentiation is done based on "commParamExt.iAmActive" (activated/inactive).

The following sections explain each use case in detail

3.2.1 Preparations

Before the initialization has started, the user (frame author) can modify the value of some "uocName"s, which default to "Trains", "RailVehicles" and "VehicleTrains".

3.2.2 Initialization

If the user (frame author) has registered the SSC Extension at the SSC Base (see an example in chapter 3 of 051_Extensibility), and when he sends the "init"=true event to the SSC Base, then the SSC Base will try initialize the extension together with all registered mandatory and optional SSC Extensions.

When all mandatory SSC extensions are successfully initialized and all optional SSC extensions are successfully or unsuccessfully initialized, then the SSC Base will continue its own initialization. This will eventually lead to "start activity".

3.2.3 Start Activity

Quite at the end of its own initialization, the SSC Base will trigger "start activity" of all successfully initialized SSC extensions.

Two options exist:

Either (1) the scene instance is the first scene instance in the game and all extensions must initialize their global states – in this case the SSC Base will send "initializeStateAndStartActivity" – or (2) the global state is already valid – in this case the SSC Base will send "startActivity".

The fields "initializeStateAndStartActivity" and "startActivity" are actually handled by the contained "SSC Core" prototype, however the specific actions are implemented by the SSC Extension:

The "Train Manager" does not maintain a global state, hence he leaves the "noState" Flag of SscCore at the default value "TRUE" and has no further duties.

3.2.4 SetTraceLevels

The frame can set the trace levels of the train manager (SrrControl, TrainControl and Modules). The semantics of the SrrControl and TrainControl trace levels are very similar to those of the trace levels SscBase and CommControl of the SSC Base, as the semantics of the Modules trace levels.

Please refer to 101_SmsBase for a more detailed description of the trace levels of the SSC Base.

3.3 Interworking with the MC Extension and with the SRR Objects (eiControl)

3.3.1 ModCoorAnnouncement

When a module is initialized, and when the module has got a train manager extension of the module coordinator, then the module coordinator extension announces itself at the SRR Controller (Train Manager).

When a module is being disabled, and when the module has got a train manager extension of the module coordinator, then the module coordinator extension deannounces itself at the SRR Controller (Train Manager).

???Hence the SRR Controller (Train Manager) is always informed about the being ???loaded/unloaded of all relevant modules and can decide whether to load/unload ???unbound models, which are assigned to this or that module.

3.3.2 Replicators

When an unbound model (vehicle model) is being created from its vehicle type (* not yet implemented *), then it needs to be assigned an initial state (position and velocity).

In the case of vehicle models, the initial state is derived from a so-called "Replicator".

That means: a given position in the track layout is referenced by an extended object ID (i.e. the extObjId of the replicator). The replicator itself describes the position within the track layout (name of the parent edge, position relative to the parent edge, direction) and the initial velocity.

Furthermore the replicator provides a graphical interface to the user, where he can select one from the (filtered) list of registered vehicle type ids, which shall be created at this replicator.

Everytime, when a replicator announces/deannounces itself at the SRR Controller (Train Manager), then SRR Controller (Train Manager) updates the list of "registeredReplicatorIds".

Everytime, a new vehicle is registered, the SRR Controller (Train Manager) updates the list of "registeredVehicleIds" in all replicators (filtering by "categories").

4 Internal View

tbd

5 Additional Info

tbd