

## SrrTrains v0.01

### Notes about the BIMPF (Browser Independent Multi Player Framework) – IV

#### Concepts for Modularity, Extensibility

Christoph Valentin, July 2011

Updated in September 2011

## 1 Table of Contents, Drawings, Tables

### Table of Contents

1 Table of Contents, Drawings, Tables.....	1
2 References.....	2
3 Summary.....	2
4 Basic Architecture of the SRR Framework.....	2
5 Modularity of the SRR Framework.....	3
6 Extensibility of the SRR Framework.....	4

### Illustration Index

Drawing 1: Basic Idea of the Modularity of the SRR Framework.....	4
---	---

## 2 References

- [1] X3D Specifications  
<http://web3d.org/x3d/specifications/>
- [2] The Network Sensor Proposal  
<http://web3d.org/x3d/workgroups/x3d-networking/>
- [3] Description of BS Collaborate (an example network sensor implementation)  
[http://bitmanagement.de/download/BS\\_Collaborate/BS\\_Collaborate\\_documentation.pdf](http://bitmanagement.de/download/BS_Collaborate/BS_Collaborate_documentation.pdf)
- [4] Notes about the BIMPF – I to III  
available at <http://simulrr.wordpress.com/berichte/> (page is in German language)

## 3 Summary

In year 2009 I started a hobby project - *SrrTrains v0.01* - that aims at the implementation of an X3D-based and network sensor-based framework for virtual multiplayer railroads (see the blog <http://simulrr.wordpress.com>).

By the end of the year 2010 it was clear that this *SRR Framework* (Simulated Railroad Framework) should own a reasonable degree

- of **modularity**
- and of **extensibility**

A **base module** should be able to support a basic class of "Simple Multiuser Online Scenes" (see "Notes about the BIMPF – II" at [4] and see the blog <http://smuos.wordpress.com>), where **extension modules** should be able to support more specific and more sophisticated kinds of models (railways, offroad vehicles, road vehicles, airplanes and so on).

An example extension module – the train manager module TMM – had already been implemented and tested at the 1<sup>st</sup> LAN Party (in March 2010). However, the LAN Party revealed several weaknesses of the SRR Framework, and I started a comprehensive re-design of the software.

**Update:** Now, in September 2011, the base module has been re-designed and it has been split into a "new" base module and two extension modules (one "Beamer Manager" and one "Key Manager"). So the interface between base module and extension modules is now more or less stable, at least for the SRR Controller.

SrrControl (old) = SrrControl (new) + SrrControlBm + SrrControlKm

## 4 Basic Architecture of the SRR Framework

The concept SrrTrains supports a specific architecture, which I call the "MMF Architecture" (**Model/Module/Frame Architecture**).

Hence any SrrTrains layout consists of three kinds of parts

- modules (at least one)
- interactive/animated/simulated models (that inhabit the modules)
- and a frame (some software that handles the central aspects of the layout – avatars, chat, the GUI and so on).

Now the **SRR Framework provides X3D prototypes for all three kinds of parts of the layout.**

The **SRR Controller** is the first of those prototypes. It is used to control the whole SRR Framework and it is instantiated as a **part of each SrrTrains frame.**

The **Module Coordinator** is the second of those prototypes. It is used to coordinate the SRR Objects (see below) of a module and to connect them with the SRR Controller. It is instantiated as a **part of each SrrTrains module.**

The **SRR Objects** are not strictly spoken a part of the SRR Framework (because it is possible for any programmers to develop their own SRR Objects), but they are delivered together with the SRR Framework. They are X3D prototypes, which are used to instrument the multiplayer capable animation, interactivity and simulation capabilities of SrrTrains models. SRR Objects are instantiated as **parts of SrrTrains models.**

Now, the SRR Controller depends on X3D and on the network sensor, only.

The module coordinator depends on all this, plus on the SRR Controller.

The SRR Objects depend on all this, plus on the module coordinator.

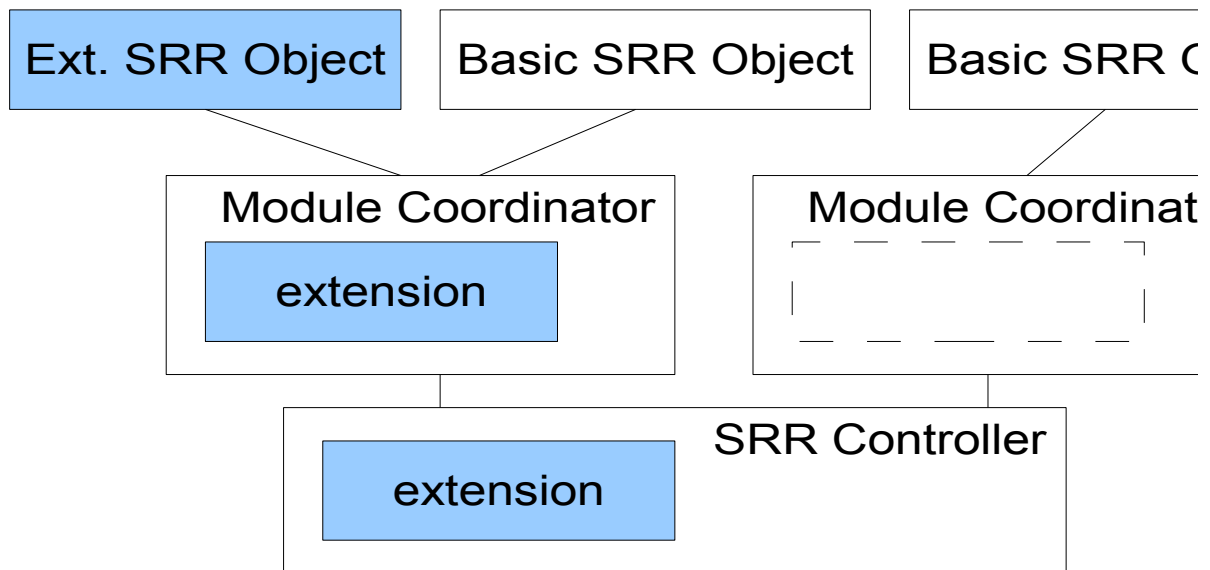
## 5 Modularity of the SRR Framework

When implementing new SRR Objects (or, let's say, new *classes* of SRR Objects), it will turn out often that the SRR Objects require modifications/enhancements of the module coordinator and/or of the SRR Controller.

This leads to the SRR Controller and the module coordinator getting more and more complex, when new functionalities are to be added.

Now, a **modular** approach was chosen for the Train Manager Module TMM (which is currently not available, because being re-designed).

The new functionality of the SRR Controller and of the Module Coordinator was outsourced to new X3D prototypes that were plugged in to the base SRR Controller and Module Coordinator. The new SRR Objects used the new functionality via the interface to the base module of the SRR Controller and of the module coordinator.



*Drawing 1: Basic Idea of the Modularity of the SRR Framework*

Drawing 1 illustrates how new ("extended") SRR Objects are supported by the extensions of the SRR Controller and of the Module Coordinator. We see two modules, where one module doesn't have the extension of the module coordinator and hence the extended SRR Objects are not supported in that module.

## 6 Extensibility of the SRR Framework

The idea of the SRR Objects is a torso, as long as it is not supported by the authoring tools.

Now, when the SRR Objects and the SRR Framework are extensible, also the authoring tools and the SRR Test Frame should be extensible by some handy plugin mechanisms.

Shortly spoken, at least following parts of the SRR biosphere should support the extensibility concept:

- SRR Test Frame
- SRR Framework (SRR Controller and Module Coordinator)
- Authoring Tools

When thinking about the SMUOS/C3P idea, following parts should support the extensibility, too.

- SMUOS
- C3P

So, this would be quite a comprehensive bunch of projects.

Let's see, what happens :-)