MBS COMPANION

Automated Control

Perth, April 2020

Table of Contents

0.1 Introduction 0

0.2 The 3DMBS Interface 1

0.3 Design Philosophy 1

0.4 Basic Principles 2

0.5 Requirements 4

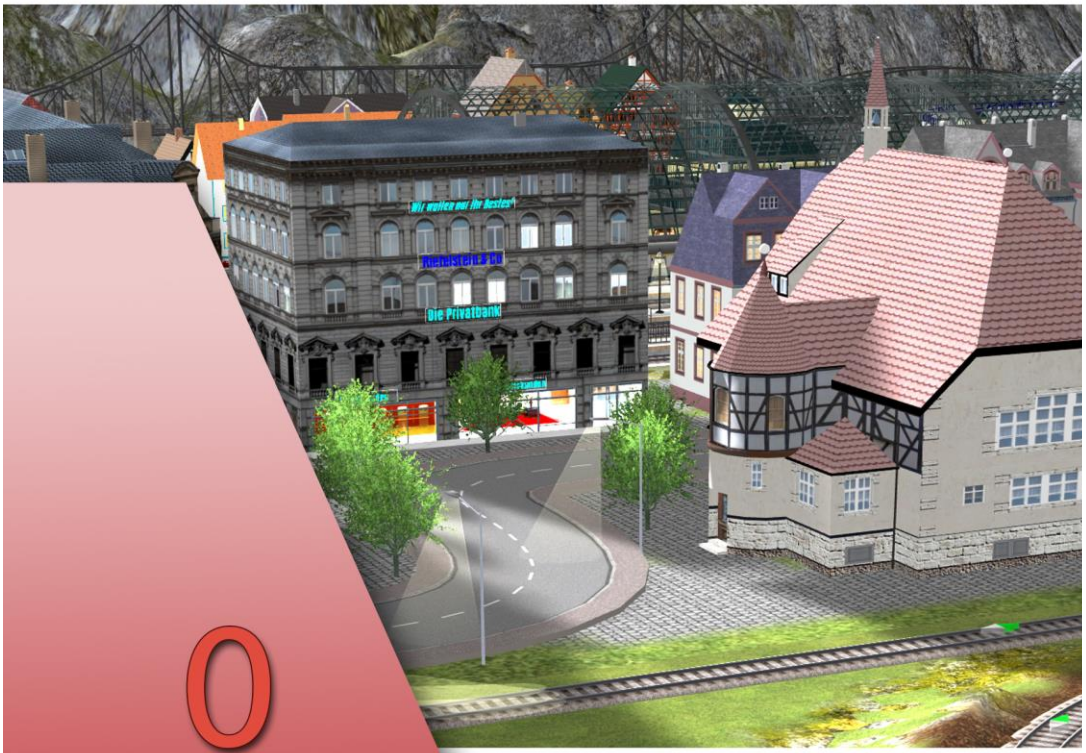
1.0 User Interface 0

2.0 Function Reference 0

2.1 MBS-Function Panel (MFP) 0

2.2 Companion Definition Section (CDS) 0

2.3 Detail View Section (DVS) 0



0 CHAPTER

Introduction

0.1 Introduction

The creation of MBS-Companion was triggered by experiencing the latest version of 3D Modellbahn Studio[2] and the scripting language LUA in particular. Building a reasonably large set using the 3DMBS is very simple, however if automated train and street traffic is desired, setting up the control mechanisms using the Event concept, with or without LUA, can be a very tedious task. Using Rocrail[1] for such automation does not really help either, because a huge amount of definition work needs to be done, in order to set up a reasonably functioning automation for larger train sets.

Firstly, it is the definition effort, which MBS-Companion attempts to address and reduce as much as possible, and secondly I wanted to build a control panel, which can be placed outside the 3DMBS window. Many tools allow building a control panel with graphical tools, however, again the definition effort is massive because all these tools allow only to position basic elements rather than more complex components like blocks, which would greatly reduce the definition effort. The main pain point of all these control tools is that they are built to connect to different environments, including real life train sets. This leads to the philosophy that the definition and identification of all components need to be done in that control tool. However, with a “smart” train set like 3DMBS, where the elements can already have names and control algorithms can be implemented, the definition effort is redundant in the external tool. The better way is to use either, the train set or the control program to take charge of the definition of elements, or better, both share the workload of defining aggregate components which can be used in a higher-level control definition.

The latter is the concept MBS-Companion is based on. However, it is not meant as a universal, all magical, problem solving super tool, but an “automation of the automation”, with predefined sets of functionalities. The tool requires certain scripts to run in 3DMBS, which implement the control mechanism, and the tool itself helps defining the parameters used to drive these control mechanisms.

As much as an attempt was made to keep this as flexible as possible, certain limitations and restrictions had to be introduced to make this task manageable.

[1] Rocrail: TM and Copyright 2002-2019 Robert Jan Versluis. Dudenhofen, Germany

[2] 3D Modellbahn Studion Copyright Stefan Werner

0.2 The 3DMBS Interface

At this point in time the only connection to the 3DMBS is the external interface provided by the author of 3DMBS, but considered “old”. However, as it stands today certain limitations can be overcome by implementing an additional functionality of the interface using variables and times.

The interface allows to write values to a variable in the 3DMBS and within the program one can use timers to execute functions based on these variables. This way a “higher level” protocol can be defined, using the existing interface simply as a transport mechanism, which can be replaced as the interface functionality may evolve. For the purpose of the MBS-companion the interface functionality was assessed as sufficient. One drawback however may be the performance limitations of the solution, which is highly depending on the processing power of the machine the solution is running on, because the timer solution takes away processing power, which could be saved with a better event-based protocol. Nevertheless, for the purpose of the development of the basic functionality of MBS-companion the performance on a typical system seems sufficient.

The interface also provides a list of commands, which can be used to place objects and other features, which are being used in the MBS-companion.

0.3 Design Philosophy

MBS-companion was created out of frustration of trying to use other programs advertised as “the best possible thing since sliced bread”. Steep learning curves, old fashioned boring interfaces, overly complicated concepts, too many details and too expensive are just a few attributes, which could be assigned to the solutions I tried.

The 3DMBS stands out from the other construction and simulation programs through a more intuitive way of building a set, although the track catalogue still supports all possible manufacturers of tracks. I believe it is not really a dedicated planning program for a real world set, others try this with more CAD like features, it also has not a huge set of materials like other systems, however it combines intuitive construction features with a reasonable size library to construct a wide range of different sets. I would call it a more “modern” concept, although it still could be better in certain areas. It would be wrong to require everything to be implemented inside 3DMBS and there is

plenty of possibilities with plugins and external programs like MBS-companion to add value to the solution, especially with a more efficient interface. The program cab be the underlying platform of more aggregate “solution wizards”. In this spirit the MBS-companion is being created.

It is also not a concept to implement a control program and panel as close to the real-world train systems as possible. There are such programs out there, but they are not for me. While in real life you may only build one large set in a life time, with 3DMBS you are actually motivated to build new ones all the time and “replant” some of the aspects of a previous set, which are nice enough to reuse. Going through the definition process of an automated control over and over again seems daunting, and by accepting that a virtual set has it’s own laws and fascinations, so has a control mechanism. We need something far more efficient to create automated, schedule driven traffic on a virtual set.

I am writing this document not so much as a user manual, but as my own implementation guide as I move along and implement different sections of the program. It is a mixture of a white paper, project plan and user documentation. A bit like “How should I know what I think before I read what I write”.

0.4 Basic Principles

This could be considered more a list of frustrations with other solutions, rather than a list of new ideas, although I believe there are plenty of those as well. And of course, there is a personal note, because I may have a different opinion about things than most other people, at least I was told that many times.

I have my own personal opinion about structure, as others may have too. Unfortunately, my personal opinion about structure clash often with the way others design their programs. The main reason probably is that most program designs are orientated towards the more inexperienced user and not necessarily towards the very impatient, which I am a perfect example.

Especially when I use an application not very often it needs to be intuitive and well structured, with each component having a clear and easy to identify purpose. Deep menus are not really useful, because they hide the functionality. This is one reason that navigation menus have been replaced by ribbon concepts, which is obvious with Microsoft Office over the years.

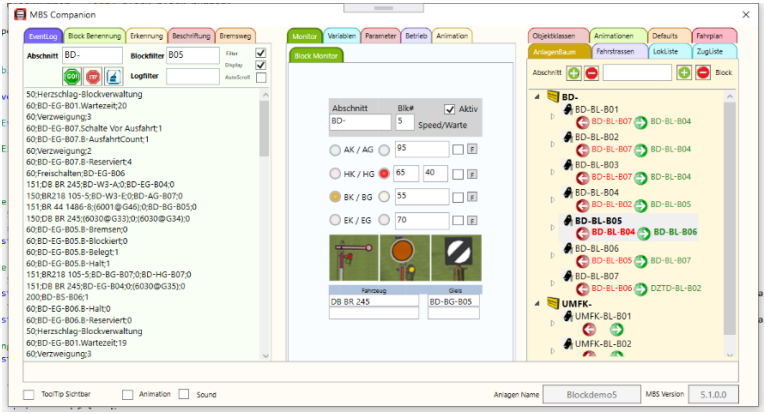
Many functions mean lots of menu entries or other concepts to activate each individual feature. I have decided to mainly use TabControls to organise different sections of functionality. They provide both a quick overview of what is available and a quick access to each section by just clicking on the tab, rather than selecting a pull-down menu entry, which inadvertently closes all the time if one does not position the mouse precisely enough.

Colours help to structure these tabs further, I like colours, easy to separate entries and easier to find a specific entry. And one more feature can be implemented, and this is something for the future, a tab element can be removed from it's control and placed in a docking area, even multiple times. We will see applications of this further down the track. Others implement different dialogues with multiple pop-up windows, which I greatly despise especially if those windows are modal.

An inspector concept, where one clicks on an object and a "property panel" shows detail is a more appropriate approach to organise a large variety of object types and stying consistent in the object specific dialogues.

So the principle of my design is:

- Quick overview of what is available,
- A fixed layout of panels to organise the flow of activity as structured as possible,
- Short ways when selecting different functions,
- Efficient ways to fill in required information.



Picture 1: Main User Interface

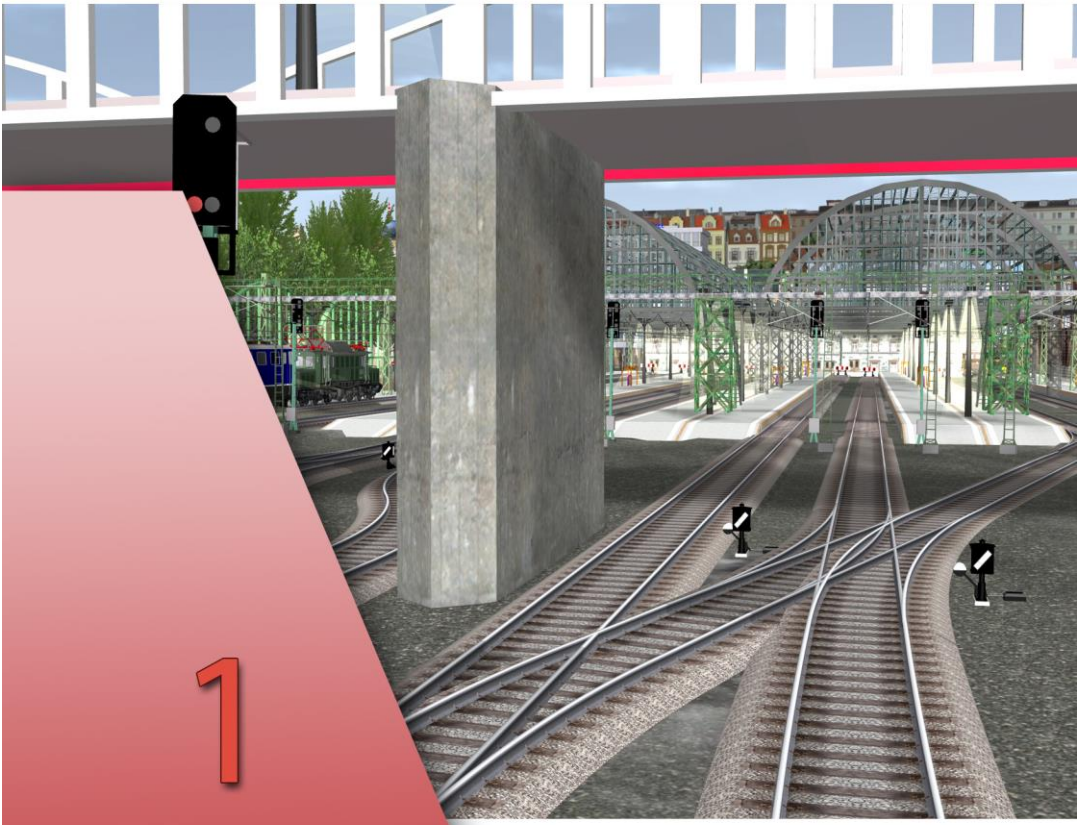
0.5 Requirements

The upmost important, most generic requirement for me is, to be able to fully automate a large, if not huge, trainset build with all possible features the 3DMBS can offer. While this sounds gigantic, it might be possible to do with the right approach of abstraction of each aspect of the control. I first looked at train traffic, but also looked at street traffic and it's specific requirements. And there are all these animations, which one may want to schedule and coordinate. And on top we have the cameras, which allow an even more personalised experience with a set.

And all of the above needs to be defined in a way, that one does not have boring loops where everything is happening repeatedly with little or no variation over and over again. What is the point of that, it just shows limitations of the control system because you can only spend so much time to write individual scripts for a specific situation. Scheduling and random events of an ever-changing experience is impossible to handle in hand crafted scripts.

The high level requirements can be summarised as following:

- Different tool components addressing specific functionality of different set components, namely train tracks, streets, etc.
- Interactive functions to help identifying aggregate components and naming the elements, namely blocks, routes etc.
- Provide LUA scripts, which implement specific functionality on top of aggregate components, e.g. Signal control, switch control etc.
- Parametrising the scripts in accordance to user requirements.
- Possibility of detailed definitions of vehicle behaviour in specific use cases and conflict management between vehicle movements.
- Helper functions to ease management of larger sets, description generation, layer management etc.
- Schedule definition and execution for each type of traffic on the set, with coordination between the types, e.g. rail crossings, traffic lights, stop signs.
- Lok an train management for different applications, with easy check in, check out mechanisms on the set.
- Efficient way to create a control panel, based on aggregate components , eliminating the need of individually setting basic elements and individually naming and linking these elements to the actual tracks, switches and signals.

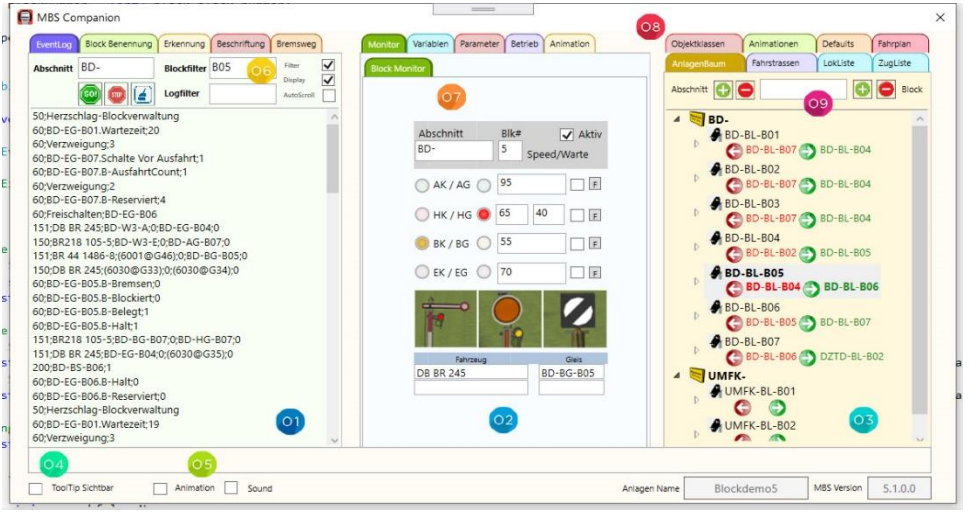


1 CHAPTER

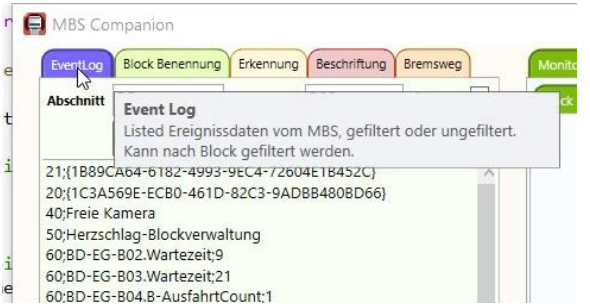
User Interface

1.0 User Interface

This chapter introduces the main elements of the user interface and the general concepts of behaviour.



The main interface is divided into three segments. The left is the MBS Function Panel (see 1) , the right one is the Companion Definition Section (see 2), and the centre position is the Detail View Section (see 3).



Most fields, tabs, buttons and frames have ToolTips attached, displaying a short hint about the purpose of the element the cursor is pointing to. I like the tip to pop up quickly, especially when you are exploring things, and not wait for it to activate, remember I am very impatient. But this has a drawback because

once you know what you are doing and if you don't ToolTips can be annoying. This is why I added (see 4) a CheckBox to turn tips off and on when needed.

The animation of the 3DMBS can be started using the CheckBox Animation (see 5) and next to it the sound can be switched on and off.

To the bottom right the current set name and 3DMBS version is displayed as general information. Above this "status line" is a display area for error- and activity messages. This will be obvious when working with the software.

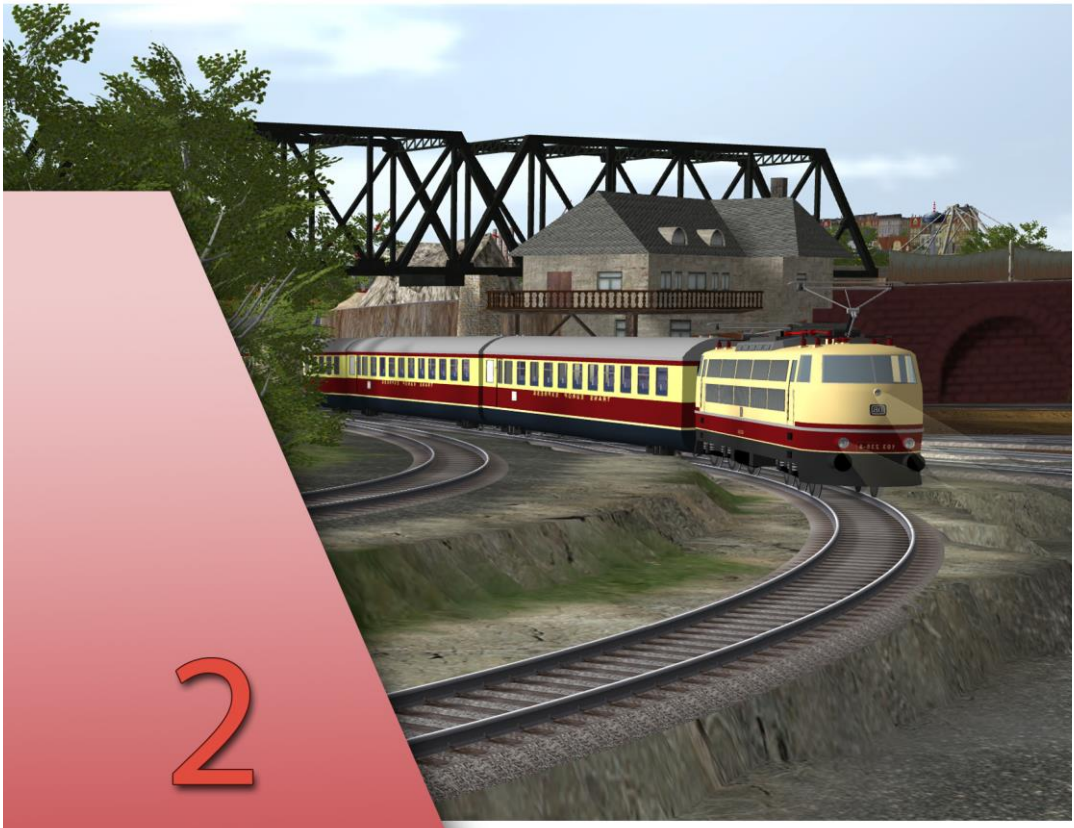
At the left in the Function Panel one can see the Eventlog being selected and Event records being displayed. Above one can see some control elements (see 6) , which belong to the Eventlog, in this case a preselected block number and section name. It was pre-loaded from the selection of a block in the Set Tree to the right. This shows the principal of all, functions displayed in this section. They have specific control elements necessary for that function, and using the Definition Section can pre-load some parameters to avoid typing. The actual Eventlog function is explained below in further detail.

In the Detail View Section, the Block Display is currently visible. (see 7) This is an example of a context specific content of this section, whenever detail information is available for a specific context.

Each section is organised in tabs and at the top of the window (see 8) one can access these tabs directly without pull down menus.

Similar to the MBS Function Panel items the Definition Section items have their own individual controls (see 9) providing control of that item.

This is all one needs to know about the general use of the interface. Anything else is specific to activities supported by the MBS-companion and is described in the Features Reference section following.



2

CHAPTER

Function Reference

2.0 Function Reference

This chapter contains descriptions of all available functions in MBS-companion. It is meant as a reference for individual features, but not as instructions how to apply them to achieve certain outcomes or solve certain problems. This is subject of the “Tasks” chapter. It does also not contain any references to the graphical Panel Tool, which is a separate unit of work, and will commence being developed once the MBS-companion is further developed. The definitions in the MBS-companion will drive the Panel Tool and the Layout Functions.

2.1 MBS-Function Panel (MFP)

2.2 Companion Definition Section (CDS)

2.3 Detail View Section (DVS)