ECP-2008-EDU-428037

D2.3
Feasibility study of Wonderland usage

Deliverable	*D2.3 – Feasibility study of Wonderland usage*
Dissemination level	*Public*
Delivery date	*1 December 2011*
Status	*Final*
Author(s)	*Otto Tronarp, Thomas Richter, Luis Bellido*

e*Content*plus

This project is funded under the *e*Content*plus* programme[1],
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

[1] OJ L 79, 24.3.2005, p. 1.

Contents

1. Introduction

The objective of the Lila project is to combine virtual laboratories and remote experiments (i.e. simulated experiments and experiments which are controlled remotely by computers) spread out over Europe, making them reachable in an environment with central retrieval and access facilitating synchronous collaboration and user generated production. The goal of this project is not only to integrate experiments and laboratories into a software infrastructure, but also to build a virtual portal in which experiments and laboratories are provided.

One of the technical challenges in LiLa has been the connection of laboratories and the 3D-engine Wonderland in which students, teachers and researchers can collaborate while interacting with the 3D model of an experiment. This document describes the work carried out in the LiLa project to provide a proof of concept prototype of a 3D collaboration environment for laboratories.

In this document, section 2 will provide a technical overview of Open Wonderland focusing on how it is possible to extend it to integrate interactive content (virtual experiments and/or remote laboratories). Sections 3 and 4 describe the work carried out to integrate MathModelica simulations in OpenWonderland, in which a coupled pendula simulation has been used to show the feasibility of this approach. Section 5 describes the experience of setting up an Open Wonderland server at the University of Stuttgart, showing that the current state of its development is not stable enough to be used in a production environment. Finally, section 6 explains the main links between this work and other work packages.

2. Technology overview

2.1. Project Wonderland – Open Wonderland

Project Wonderland is a Java open source toolkit for creating collaborative 3D virtual worlds developed by Sun Microsystems. The overall vision of the project was to build a highly scalable and flexible platform for creating collaborative 3D virtual worlds that could be used by organizations for building custom virtual worlds. The envisioned usage was for example:

- Virtual workplaces for companies with substantial amount of employees working remotely.
- Virtual meeting place for meeting customers, partners and employees.
- Virtual class rooms for distance learning and e-learning.

When Oracle acquired Sun Microsystems in the beginning of 2010 they decided to cease all funding for the project. Since then the project has continued as a community driven open source project named Open Wonderland with the non-profit corporation Open Wonderland Foundation as maintainer. The Open Wonderland software can be downloaded from their home page [1].

OpenWonderland itself builds on top of the DarkStar multiplayer game server, which has also been discontinued by Oracle. The project itself was then later on released as an open source project named "Red Dwarf", but the last development activity recorded in the version management system is from 2010.

2.2.Integrating interactive content in Open Wonderland

Open Wonderland has rich extension capabilities through a number of extension points, the most prominent are:

- Add new types of 3D objects (cells in Open Wonderland terms) with behaviour.
- Add new capabilities that can be used to augment the functionality of existing cells.
- Add custom connections, i.e. create custom data channels for client to client communication.

All extensions are package into a module that is deployed on the server and they are then automatically downloaded by the clients.

SAS/VNC

The module system with its extension points is the obvious way of integrating interactive content, but before one ventures on to develop a custom module one should be aware of that it exists at least two more noteworthy options for getting existing GUI applications into Open Wonderland.

- SAS – Shared Application Server
- VNC Viewer module

Open Wonderland features a server component called SAS – Shared Application Server, through that mechanism it is possible to show an arbitrary X11 application within the world. That of course only works for applications that runs under Linux or any other system that uses X11. However, there also exists a VNC Viewer module that makes it possible to show a VNC session within the world that could be used to integrate Windows applications. The VNC Viewer module is available in the Open Wonderland module warehose [2]. An example of a VNC View in Open Wonderland can be seen in Figure 1 where a LabView experiment is shown within the Open Wonderland world.

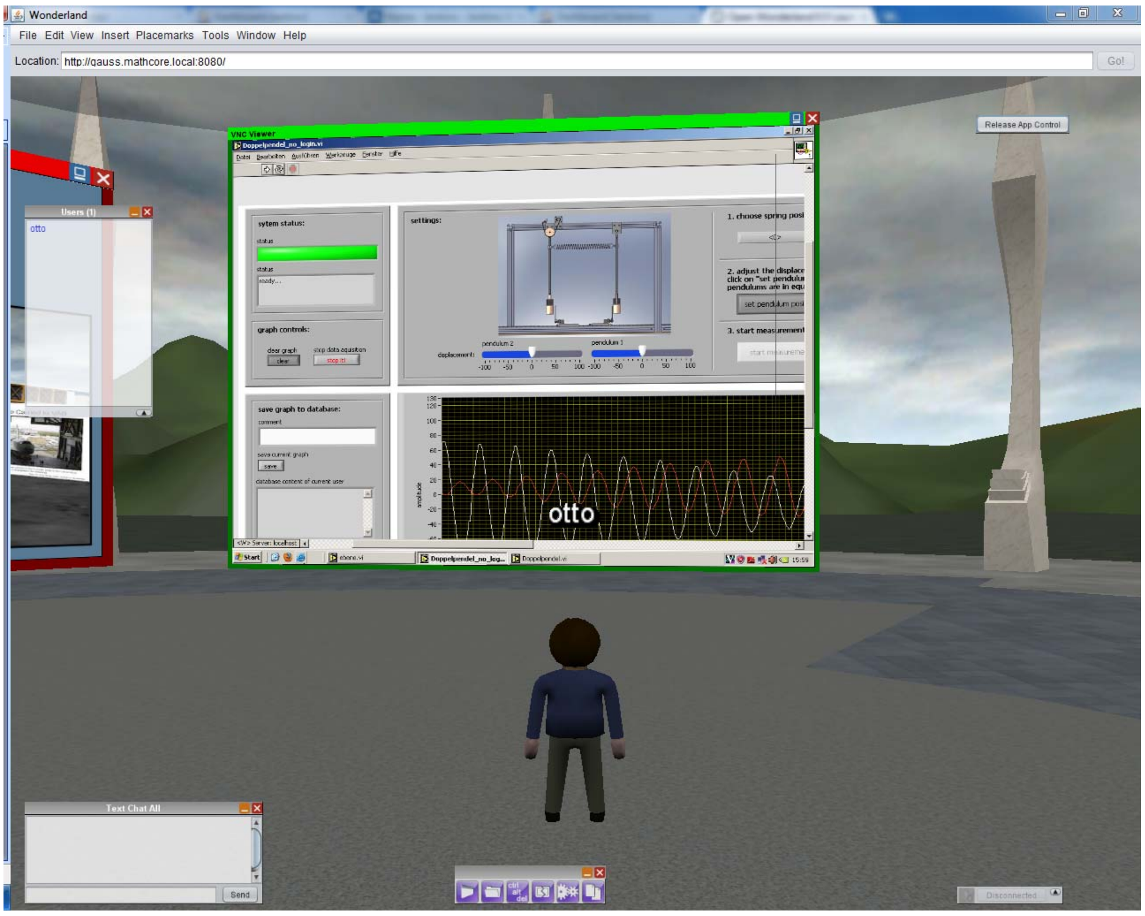

Wonderland
File Edit View Insert Placemarks Tools Window Help
Location: http://gauss.mathcore.local:8080/
Go!
Release App Control
VNC Viewer
Users (1)
otto
otto
Text Chat All
Send
Disconnected

Figure 1: Screenshot of VNC session with a LabView application shown within Open Wonderland.

2.3. Custom cell module

To truly integrate content with a 3D experience it is necessary write a custom cell module. At the bare minimum one needs to implement the following classes:

- Cell – Client side representation of the cell.
- CellMO – Server side representation of the cell.
- CellFactory – Client side factory that makes it possible create cells dynamically.
- CellRenderer – Renders a 3D view of the Cell in the client.
- CellClientState – Contains necessary information to configure a client side representation of a cell (Cell).
- CellServerState – Contains necessary information to configure a server side representation of a cell (CellMO).

A large part of implementing this is just boilerplate code and most of the actual behaviour of the cell is defined in the Cell and CellRenderer classes.

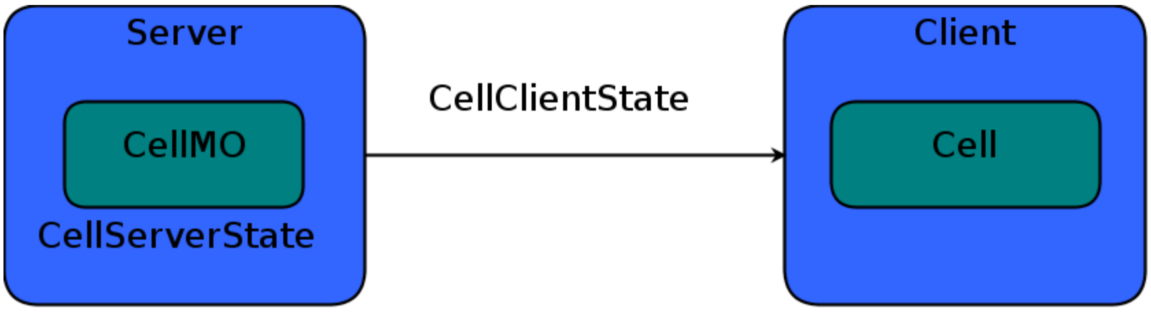
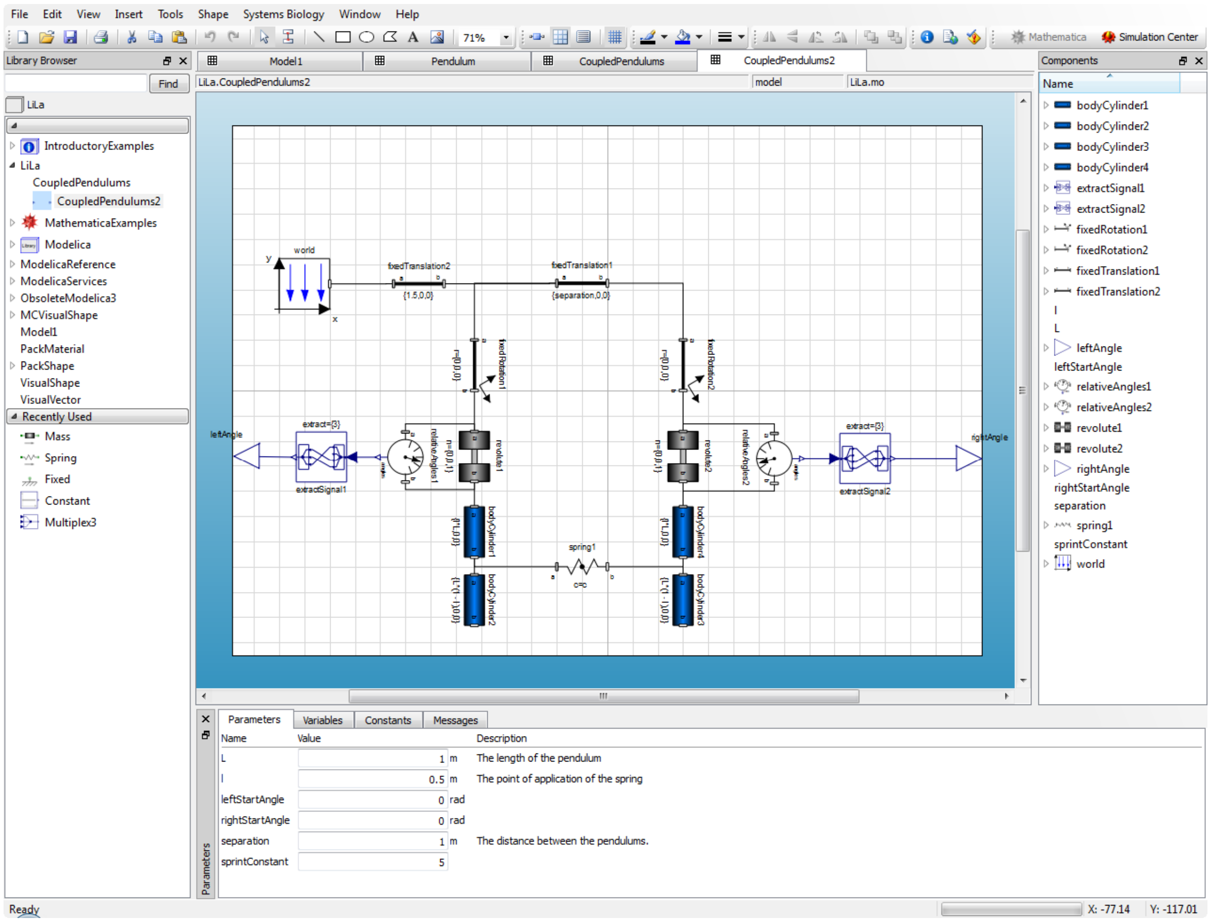


Figure 2: Relation between the different cell classes.

3. MathModelica simulations in OpenWonderland

Based on the Modelica language [3], MathModelica is a platform suitable for modeling and simulation of dynamic multi-engineering and life science systems. MathModelica comes with a customizable set of Modelica component libraries and provides an environment for modeling, simulation, analysis, and documentation.



Name	Value		Description
L	1	m	The length of the pendulum
l	0.5	m	The point of application of the spring
leftStartAngle	0	rad	
rightStartAngle	0	rad	
separation	1	m	The distance between the pendulums.
sprintConstant	5		

Figure 3: System Designer, the modeling environment in MathModelica, with a multi body model of a coupled pendula.

One domain that the Modelica standard library has good support for is multi body mechanical systems. Figure 3 shows System Designer, the modelling environment in MathModelica, with a multi body model of a coupled pendula. One of the strong points of the multi body library is that from the system model it is possible to automatically generate a 3D visualisation of the system. In the MathModelica-module for Open Wonderland we leverage that possibility to create a generic module that is able to visualize and control an arbitrary MathModelica multi body model.

4. MathModelica-module

The overall structure and work flow of the MathModelica-module is depicted in Figure 4. First a multi body model is created in MathModelica and a simulation executable is exported. That executable contains the optimized equations for the system as well as a numeric solver. To be able to communicate with the simulation a protocol named MMCom has been developed and an MMCom server are embedded in the simulation executable.

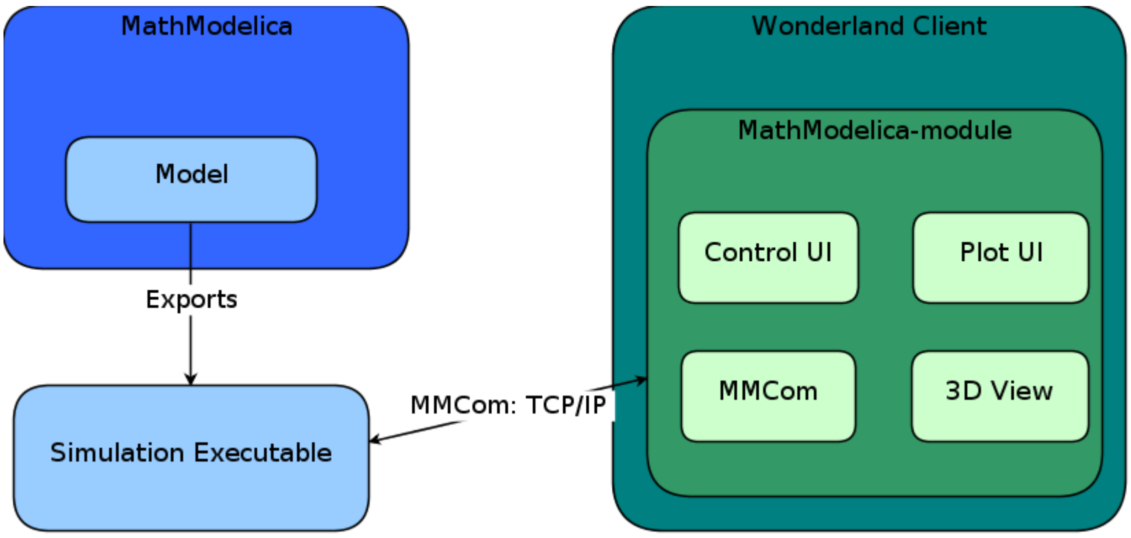


Figure 4: Structure and work flow of the MathModelica-module.

The MathModelica-module consists of 4 fundamental parts the MMCom, Control UI, Plot UI and the 3D View.

4.1. MMCom

MMCom is the underlying communication protocol that is used for communicating with a running simulation. The protocol consists of two session types MathModelica Simulation Control Sessions (MM-SCS) and MathModelica Simulation Data Session (MM-SDS).

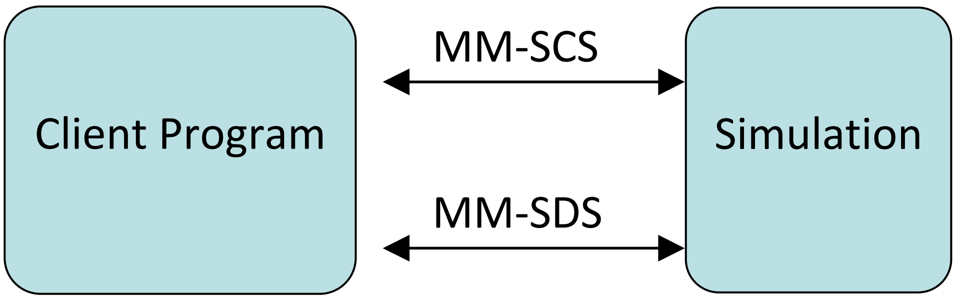


Figure 5: In MMCom two session types are used for communicating with the simulation.

The MathModelica-Simulation Control Session is used to control a running simulation server by issuing start/stop commands and setting input variable values, start values and parameter values. A MM-SCS can also be paired with a MM-SDS and then it can be used to setup which variables the MM-SDS wants to subscribe to. The protocol runs over TCP and a message consists of a binary header and a text payload.

The MathModelica-Simulation Data Session is used to receive simulation data from a running simulation server. An MM-SDS session must be paired with an MM-SCS session that is used to setup the variable subscriptions; the default subscription is to only receive time data. The protocol runs over TCP and a message consists of a binary header and a binary payload.

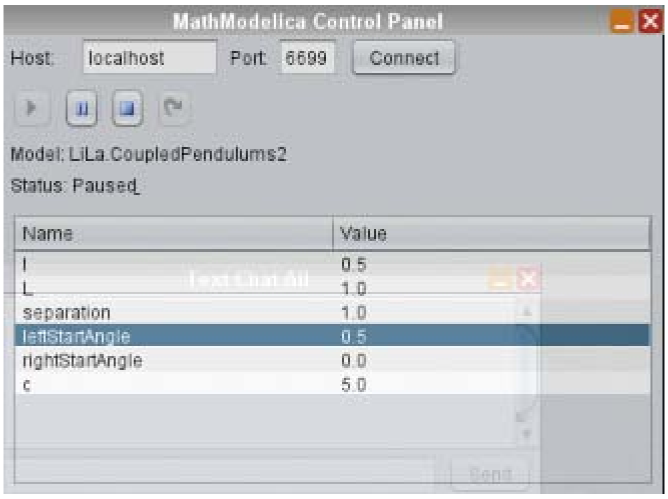
4.2. Control UI

The control UI shown in Figure 6, resides in the heads up display (HUD) and has static controls for

- Initiate a connection to a simulation.
- start/ pause/restart a running simulation.

When the MathModelica-module connects to a simulation it queries the simulation for all user settable parameters and populates the table in the UI with these. That table is editable by the user and if changes are made they are sent to the simulation which reinitializes the system with the new values.

To avoid cluttering interface with all parameters from the model only the top level parameters are shown in the GUI. This means that the model designer needs to propagate all parameters that the user should be able to change from the top level. This is in any case a good modelling practice.


MathModelica Control Panel

Host: localhost Port: 6699 Connect

Model: LiLa.CoupledPendulums2

Status: Paused

Name	Value
l	0.5
L	1.0
separation	1.0
leftStartAngle	0.5
rightStartAngle	0.0
c	5.0

Figure 6: The Control UI.

4.3. Plot UI

When a simulation is running the MathModelica-module detects if the model contains any top level output variables. If it does a plot ui is created in the HUD with a plot curve for each variable. By propagating interesting variables to top level outputs the model designer can control what is shown within the Open Wonderland world.

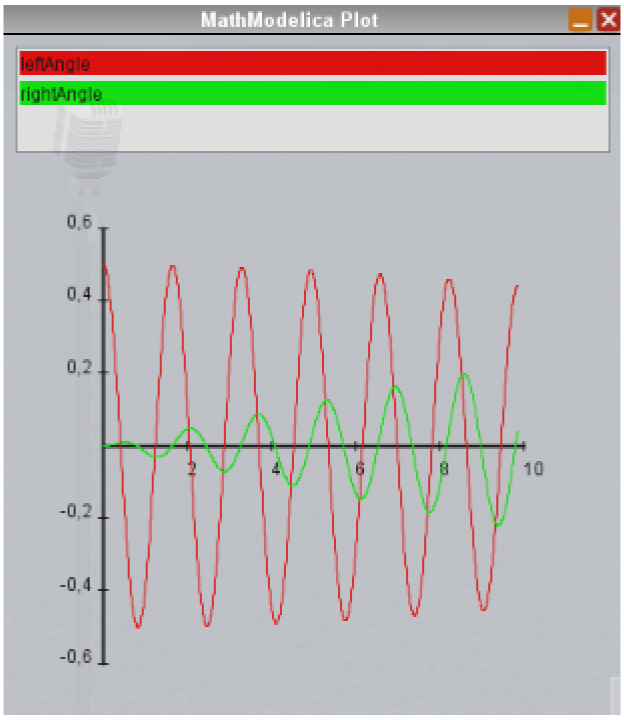
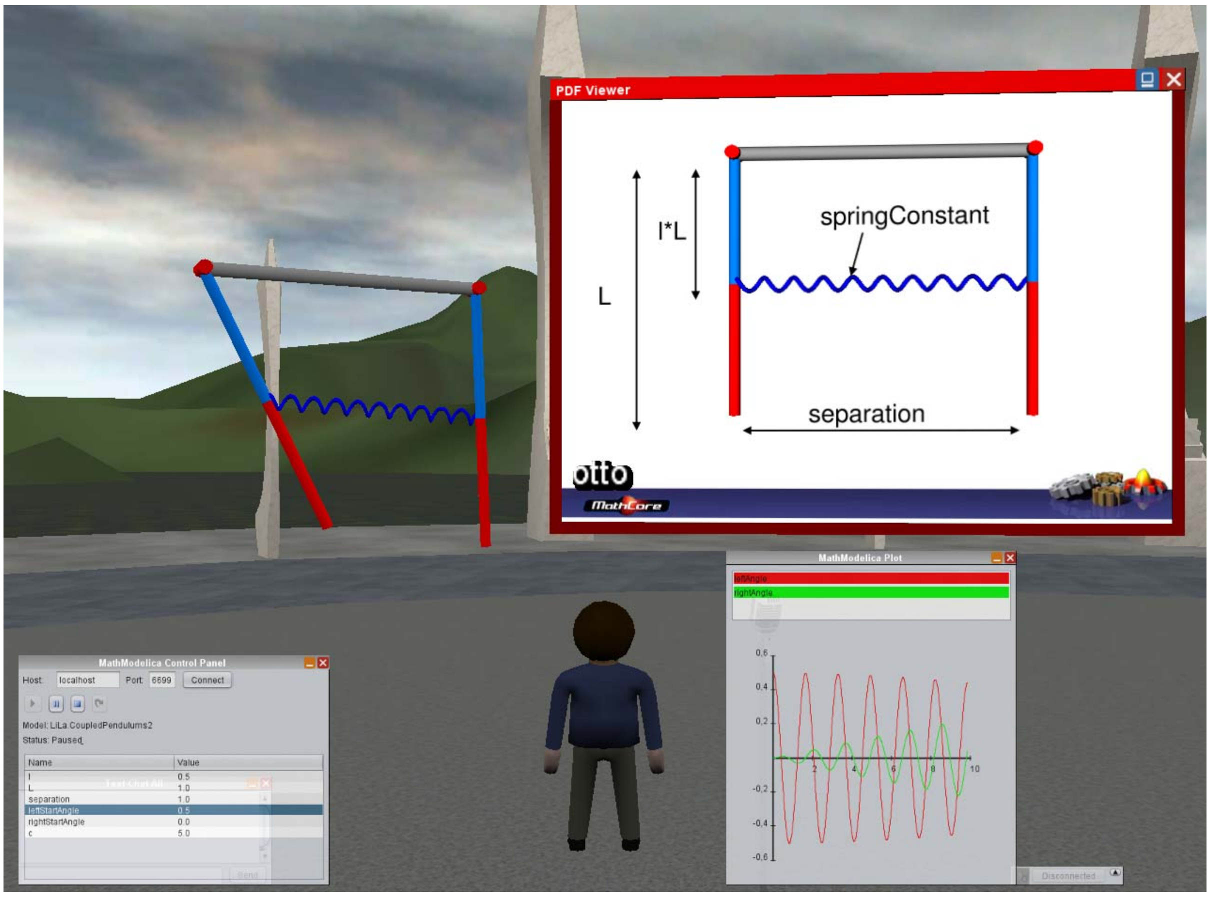


Figure 7: The Plot UI

4.4. 3D View

The simulation runs synchronized with real time and streams simulation result data to the MathModelica-module. From the simulation data every multi body component is identified and a 3D representation of that component is created. With every new data sample that arrives from the simulation the position, orientation, size, and colour of the 3D objects are up dated. In that way a complete 3D representation of the system is constructed and animated for as long as the simulation is running. The complete user view is show in Figure 8. There the Control UI can be seen in the lower left, the Plot UI in the lower right and the 3D view in middle slightly to the left.



Host: localhost Port: 6699

Model: LiLa.CoupledPendulums2

Status: Paused

Name	Value
l	0.5
L	1.0
separation	1.0
leftStartAngle	0.5
rightStartAngle	0.0
c	5.0

Figure 8: Screenshot of the MathModelica-module in wonderland showing a coupled pendula simulation along with a PDF documentation for the simulation.

4.5. Summary and future possibilities

With the MathModelica-module for Open Wonderland we have created an almost zero-effort experience for publishing an existing MathModelica MultiBody model as an interactive experiment in the Wonderland world. There the user can se a live 3D view of the system and interact with it in real time by changing parameters in the system.

Currently it is limited to work with simulations that runs in real time, and that is a limitation since there is a limit to how complex the system can be to actually be solved in real time.

Interacting with the system by setting values in a regular 2D GUI somewhat weakens the 3D experience. To enhance the 3D experience it would be worthwhile to provide ways to interact directly with the 3D representation in the world, i.e. push with the mouse to apply a force.

5. Open Wonderland setup at the University of Stuttgart

The University of Stuttgart hosted an OpenWonderland and a DarkStar server for experimental usage until Spring 2011. In March 2011, an intruder was detected on the system and the compromised server had to be taken down from the net. Further investigation on the event took up several weeks, and a new virtual machine had to be setup from scratch. On this machine, an OpenWonderland was freshly installed, including a new version of the Darkstar server. It was found that this installation, unfortunately, was non-working. Inspecting the server logs showed that the Darkstar server, a functional basis on which Wonderland operates, tried to connect to an unknown port; permanently opening the firewall of the system was not considered an option given the former intrusion, but even upon temporarily opening all ports could not resolve this problem. Several older versions of the same OpenWonderland system were also tried without any success.

While the source code location of the apparent failure is approximately known due to the log files, the publically available DarkStar sources are no longer identical to the compiled DarkStar binaries in the Wonderland distribution, and due to this, a clear reason for the failure could not be found.

The overall feasibility of running or deploying Wonderland is thus quite low: OpenWonderland being no longer supported by Oracle, and the supporting DarkStar (now RedDwarf) server not having been updated for over a year and being out of sync with Wonderland shows that the whole OpenWonderland project is in a rather desolate, and probably unsupported state. Opening all network ports on a computer system is not an option, as the intrusion into our systems demonstrated quite clearly.

The current state of both projects does not allow a continuous and professional deployment at a computing center, probably requires additional manpower to be brought into a working state, and the development pace of both projects seemed to have slowed down, if not stopped completely. Taking Wonderland as a basis for a stable service can thus not be recommended.

6. Links to other workpackages

In this task, we attempted to realize a link between a real lab, namely the coupled pendulum in Berlin, and the simulation of the pendulum in OpenModelica. The outcome would be that students had the possibility to compare reality with simulations next to each other, in the 3D world provided by Wonderland. As such, this is both a project for WP4, where the contents was generated, and WP3 as a new didactical model would have been deployed. Unfortunately, the current state of the technical platform made a realization of this experiment impossible.

References

[1] http://openwonderland.org/. Last accessed: 2011-11-14

[2] http://openwonderland.org/module-warehouse/module-warehouse. Last accessed: 2011-11-14

[3] http://www.modelica.org/. Last accessed: 2011-11-14