

Net-WMS

SPECIFIC TARGETED RESEARCH OR INNOVATION PROJECT

Networked Businesses

D5.5 - Prototype of the collaborative system between user and optimisation module

Due date of deliverable: 30 October 2008

Actual submission date: 30 October 2008

Start date of project: 1 September 2006

Duration: 36 months

Organisation name of lead contractor for this deliverable: INRIA

Version 1

**Project co-funded by the European Commission within the Sixth Framework Programme
(2002-2006)**

Dissemination Level : PU

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	Net-WMS
Project Full Name:	Towards integrating Virtual Reality and optimisation techniques in a new generation of Net worked businesses in Warehouse Management Systems under constraints
Document id:	D 5.5
Document name:	Prototype of the collaborative system between user and optimisation module
Document type (PU, INT, RE, CO)	PU
Version:	1
Submission date:	30 October 2008
Authors: Organisation: Email:	François FAGES (INRIA) - Francois.Fages@inria.fr Camille CHIGOT (CEA) – camille.chigot@cea.fr Philippe GRAVEZ (CEA) – philippe.gravez@cea.fr

Document type PU = public, INT = internal, RE = restricted, CO = confidential

ABSTRACT:

This document describes the software prototype (deliverable D5.5) that integrates the Virtual Warehouse previously specified in WP5 with the optimisation module developed in WP4. It thus supports the interactions between human users and solvers. At the moment, the D5.5 prototype has achieved a functional level that makes it suitable for tests with Net-WMS end-users. We expect that the feedback from these tests will allow us to significantly refine the initial specifications written in deliverables D5.2 and D5.3.

KEYWORD LIST: Virtual Warehouse, Virtual Reality, 3D Visualisation, Specification

MODIFICATION CONTROL			
Version	Date	Status	Author
0	DD-MM-YYYY	Template	ROHOU Philippe
03	24-10-2008	Draft	CHIGOT Camille
1	29-10-2008	Version 1	FAGES François

Deliverable manager

- François FAGES (INRIA)

List of Contributors

- Camille CHIGOT (CEA)
- Philippe GRAVEZ (CEA)
- Abder AGGOUN (KLS Optim)

List of Evaluators

- Nicolas BELDICEANU (EMN)
- Filipe CARVALHO (WideScope)

TABLE of CONTENTS

1	INTRODUCTION	6
2	ARCHITECTURE.....	8
2.1	XML MANAGER	9
2.2	WEB SERVICES	10
3	DESCRIPTION OF THE VIRTUAL WAREHOUSE PROTOTYPE.....	11
3.1	ARCHITECTURE	11
3.1.1	<i>XDE: the Physical Engine.....</i>	<i>12</i>
3.1.2	<i>DSimI: the Communication framework.....</i>	<i>13</i>
3.1.3	<i>XDE Physic Pack</i>	<i>14</i>
3.1.4	<i>Virtual Warehouse.....</i>	<i>15</i>
3.1.4.1	New classes.....	15
3.1.4.2	Modifications on the Scene object	15
3.1.4.3	Simulation Manipulator added specifications	16
3.1.4.4	Visual Manipulator added specifications	16
3.1.4.5	Solver Interface Modification	17
3.1.5	<i>Virtual Warehouse GUI</i>	<i>18</i>
3.2	MAN-MACHINE INTERFACE.....	19
3.2.1	<i>Here is the description of the functionalities of the GUI</i>	<i>19</i>
3.2.2	<i>Main Windows.....</i>	<i>19</i>
3.2.3	<i>Shortcuts and Selection</i>	<i>21</i>
3.2.4	<i>Menus</i>	<i>22</i>
3.2.4.1	File Menu.....	22
3.2.4.2	Windows Menu.....	22
3.2.4.3	History Menu	23
3.2.4.4	Test Menu	23
3.2.4.5	Solver Menu	24
3.2.5	<i>Object Placement Window</i>	<i>25</i>
3.2.6	<i>Colour Management.....</i>	<i>27</i>
3.2.7	<i>Camera Management</i>	<i>28</i>
3.2.8	<i>New Colour Window</i>	<i>29</i>
3.2.9	<i>Warning and Colour Management.....</i>	<i>30</i>
3.2.10	<i>Object Properties</i>	<i>32</i>
3.2.11	<i>Physics Window.....</i>	<i>33</i>
3.2.12	<i>Option Window.....</i>	<i>34</i>
3.2.13	<i>Test Windows.....</i>	<i>36</i>
3.2.13.1	Console Window.....	36
3.2.13.2	Data Window	36
3.2.13.3	Wrong Rules Window.....	37
3.2.13.4	Object Maker	38

3.3	VIRTUAL WAREHOUSE-SOLVER INTERFACE	39
3.3.1	<i>Starting a Scene.....</i>	39
3.3.1.1	Pure Graphical	39
3.3.1.2	Physicalisation	39
3.3.1.3	NWMS Simulation	40
3.3.2	<i>Configuration and compatibility</i>	40
3.3.3	<i>Loading a problem from database</i>	42
3.3.4	<i>Examples of use.....</i>	42
3.3.4.1	Best Positions.....	43
3.3.4.2	Iterative Placement	43
4	DESCRIPTION OF THE SOLVER PROTOTYPE	46
4.1	D5.2 SPECIFICATION VALIDATION	47
4.1.1	<i>Communication from the Virtual Warehouse to the Optimizer.....</i>	47
4.1.1.1	Uploading a Packing Knowledge Model	47
4.1.1.2	Uploading a Packing Problem Instance	47
4.1.1.3	Solving Request	47
4.1.1.4	Request for other solution	47
4.1.1.5	Request for Getting a Solution Close to Current Configuration.....	48
4.1.1.6	Placement Requests for one Object.....	48
4.1.1.7	Placement Requests for one Region.....	48
4.1.1.8	Information Request on one Object or one Region.....	48
4.1.1.9	Control Requests.....	48
4.1.2	<i>Communication from the Optimizer to the Virtual Warehouse.....</i>	48
4.1.2.1	Solutions	48
4.1.2.2	Failure and Explanations.....	48
4.1.2.3	Regions	49
4.1.3	<i>Incremental Execution.....</i>	49
4.1.4	<i>Synchronous Communication.....</i>	49
5	FIRST EXPERIMENTS	50
5.1	“WHAT IF” ANALYSIS	50
5.1.1	<i>I want to place a single object on a bin.....</i>	50
5.1.1.1	Solver Placement	50
5.1.1.2	Automatic Placement.....	50
5.1.1.3	3D Manual Placement.....	51
5.1.1.4	Arrow Manual Placement	51
5.1.2	<i>Testing the stability of the structure.....</i>	51
5.2	MISSING FEATURES	52
5.2.1	<i>Weld/Unweld a single item to a bin</i>	52
5.2.2	<i>Reset.</i>	52
5.2.3	<i>Freeze/Unfreeze everything by level</i>	52
5.2.4	<i>Activate/Deactivate Centre of Gravity</i>	52
5.2.5	<i>Rule Colour feedback and texture.</i>	52
5.2.6	<i>Internationalization.....</i>	53
5.2.7	<i>Adding Objects Dynamically.....</i>	53
5.3	BUGS AND MALFUNCTIONS	53
5.3.1	<i>Interpenetration of objects</i>	53
5.3.2	<i>Texture Application</i>	54
5.3.3	<i>Automatic Place outside of the bin.....</i>	54
6	CONCLUSION	55

1 Introduction

Net-WMS aims at simplifying and making more efficient the packing processes (planning, packing and unpacking) in warehouses so as to save shipping space and perform faster and better. In packing activities, several levels of containment must be considered from the boxes that are typically provided by suppliers to the containers that are actually shipped as single units. In the automotive industry, storing a simple part in a container involves two or three packing levels as described in Net-WMS deliverable D2.2 "Case Studies". This multi-level packing process is difficult to manage and Net-WMS intends to demonstrate the benefits that optimization and virtual reality (VR) techniques may bring to Warehouse Management Systems (WMS). These two approaches are fully complementary; the former mainly focuses on automated decision-making assistance while the latter emphasizes interactivity with the humans in charge of planning the packing processes.

Net-WMS Work Package WP5 "Virtual Reality applied to packing problems in a WMS" addresses three main issues:

1. The 3D visualisation of the packed and packing objects allowing human operators to handle them "virtually".
2. The interactive (i.e. real-time with respect to human perception) simulation of the physics laws ruling these objects in order to "virtually rehearse" packing and unpacking operations, as well as to dynamically study the behaviour of container contents during shipping.
3. Combining the capabilities of optimisation and virtual reality to provide the most efficient assistance to the human packing planners.

For WP5, the second year objective was the definition and specification of the Virtual Warehouse module. This module will be the main user interface of the whole application. The specification of the Virtual Warehouse was divided between:

D5.2: Specification of the collaborative system between user and optimisation module (M18) specify the integration between the Warehouse API and the solver (3rd point above).

D5.3: Specification of the Virtual Warehouse API (M18) specify the visualisation and interactive simulation functions.

The end of the second year / beginning of the third year is dedicated to the development of the Virtual Warehouse module according to the specifications given in the previous deliverables. Due to the complexity of the task, the work was divided into two deliverables.

D5.4: Implementation of the interactive physical simulation embedded in the virtual warehouse.

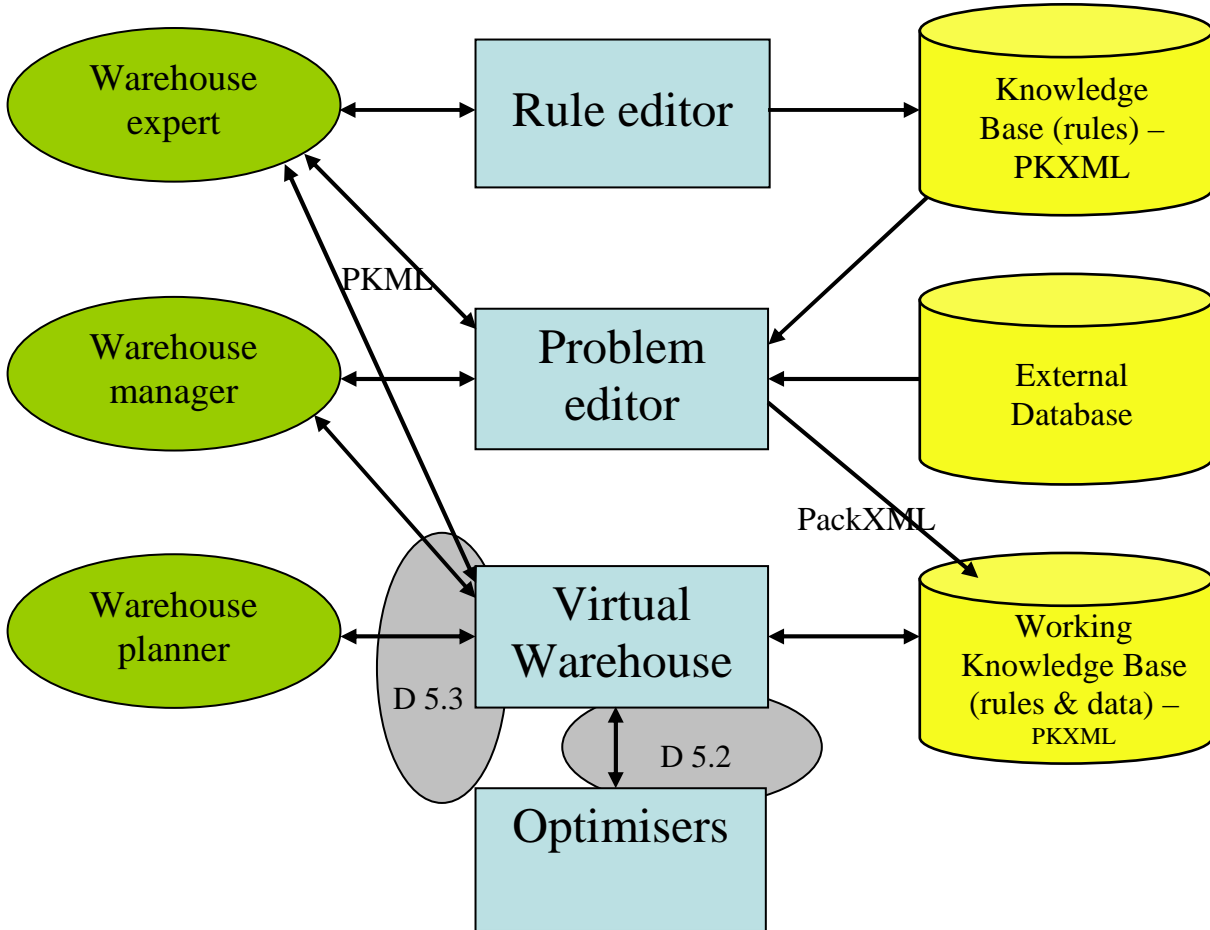
D5.5: Associated with the current document and formerly scheduled on M28, it prototypes the collaboration between user and optimisation module. However, because of its importance with respect to end-user requirements, the decision was made to switch D5.4 and D5.5 in order to give the end-users a working prototype at an earlier date.

The current document is thus linked with deliverable D5.5 "Prototype of the collaborative system between user and optimisation module". It describes the architecture and the collaboration between solvers and the Virtual Warehouse. The implementation of D5.5 is an important milestone in the project: it shows the collaboration between the developer partners of the consortium to provide a system usable by the end-users. The D5.5 prototype was implemented according to the D5.2 (and D5.3) specifications.

The goal of this document is to describe the implementation of the final architecture of the solver/Virtual Warehouse interaction. At the same time, it will validate (or modify) the specifications defined in the D5.2 and initially prototyped in CLPGUI without physical simulations. This document will also ensure the definitive stability of architecture and protocols of the application.

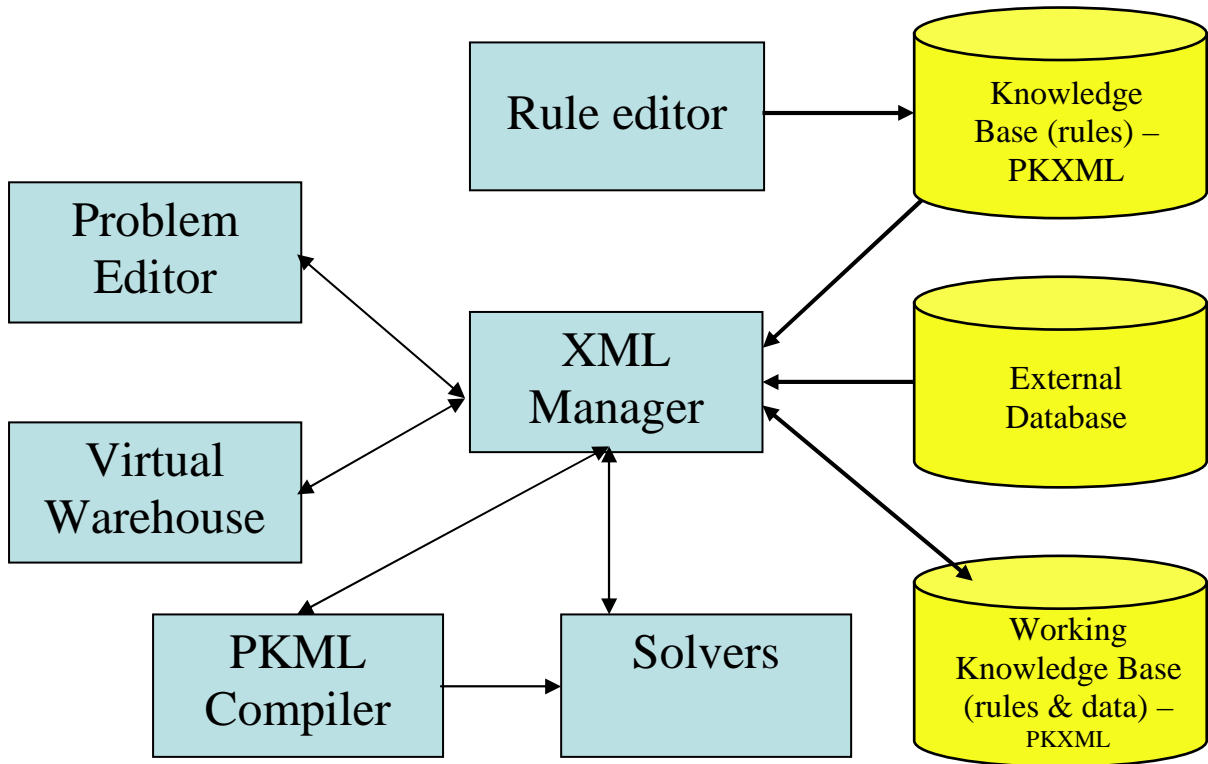
2 Architecture

The architecture of the application has slightly evolved with a better understanding of the problems to deal with a generic multi-language distributed application. The architecture designed in D5.2 and D5.3 was:



Since then, the amount of data sent on the network has led to centralize all the conversion of data from PKML to PackXML and Java in a single unit called XML Manager. This unit loads and translates all exterior objects into Java objects that can feed the solvers made in Java-Choco. To assure the compatibility with the Virtual Warehouse module that is written in C#, the solution of web services was taken. It allows an easy way for the C# application to access to methods of an application running on a server – J2EE application server (Apache Tomcat).

Here is the new architecture including the XML Manager translating all the data:



2.1 XML Manager

The XML Manager is a Business Packing Controller enhanced with capabilities to support various packing data models ranging from Pallet Loading Problems to packing unit loads into containers and vehicles. The management of data is separated from solving instances of packing. Its architecture design offers capabilities to support several interfaces:

- Reading data model from a Data Base
- Reading data model from flat ASCII, Excel and XML files
- Creating packing models through foreign applications using API modules
- Feeding packing models through Web services using SOA technologies.

2.2 Web Services

The web services are a state of the art technology that allows any application to run on high level platform (J2EE for Java and .net for C# for example) to communicate without having to marshal or translate data from a language to another.

The main goal of web services is to allow an application to provide a service on the web (meaning running somewhere on a server) to other applications via a common interface independent of the language the developers used to code the applications. In a practical way, it allows us to make a link easily between the Java part (KLS solvers) and the C# part (CEA Virtual Warehouse) of the application. The web services have a common description language called WSDL (Web Services Description Language) and use the SOAP (Service-Oriented Architecture Protocol) to communicate.

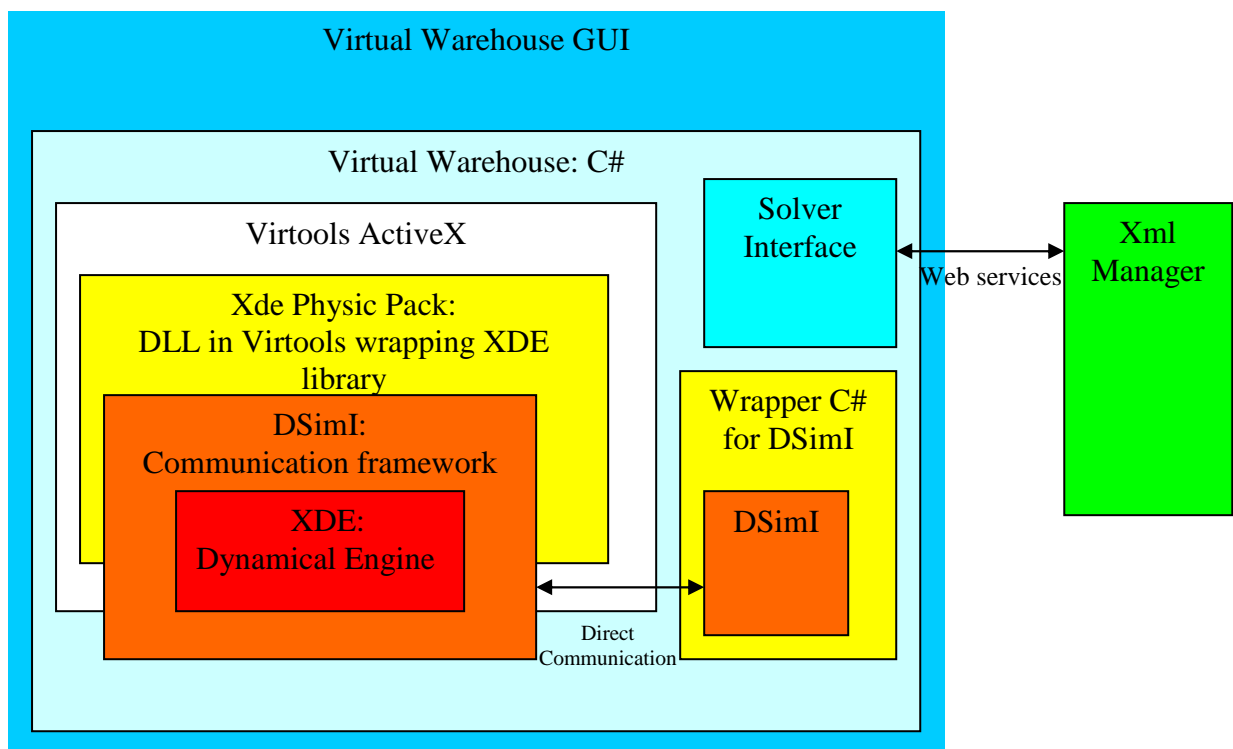
In the current architecture of our application, the XML manager is a component that turns on an Apache server (using Tomcat). It provides a WSDL file describing its interface. The file is transformed into a C# class inside the Virtual Warehouse API. In this way, all the methods to modify the data inside the XML manager (that represent the state of the solver) and to command the solver are embedded inside the Virtual Warehouse API. It allows to create a fully working GUI for end-users accessing Java data completely in C#.

3 Description of the Virtual Warehouse Prototype

Here comes the description of the prototype. This prototype is the core of the future product that will be given to the end-users. It will be completed to better take into account physical aspects with the help of virtual reality through deliverable D5.4. Those VR functions will be improved further with the feedback of end users.

3.1 Architecture

The global architecture of the Virtual Warehouse is quite complex.



Architecture of the Prototype: blue components are in C#, red and yellow components are in C++ and the XML Manager is in Java.

The main component of the Virtual Warehouse is the Virtools ActiveX that allows visualizing the scene. Inside, we have a library made with the SDK Virtools that allows the update of visual components given their position in the physical engine. The communication between the physical engine and these graphical units is made by the DSImI library: a communication framework. To address directly the physical engine to have quicker answers and more modularity, a part of DSImI is directly wrapped in C#. Finally, the Solver interface uses the implementation of the wsdl to communicate with the solvers and the XML manager.

In this architecture, the only component not developed by companies of the consortium is the Virtools ActiveX. Furthermore, the XDE and DSImI component are developed in the LSI laboratory in the scope of Net-WMS project as well as in other internal and industrial projects.

We will now develop each component to explain what they do.

3.1.1 XDE: the Physical Engine

XDE alias eXtended Dynamical Engine is the physical engine developed by CEA-LIST at the LSI laboratory.

It is a real time physical engine used for simulation according to the rules of mechanics. Its main goal is to allow the manipulation of physical objects with various devices of manipulation. The real-time part allows such devices to render haptic feedback.

One of the main difficulties of performing effective physical simulation is to have efficient collision detection. For this purpose, XDE has integrated two libraries of collision detection:

- VPS++ is an enhancement of the VPS™ library from Boeing and is based on a voxel representation of the objects. With VPS++, the collision computation is very fast but the contact model is very crude. See <http://www.boeing.com/phantom/vps/> for more information about VPS.
- LMD++ is developed at CEA/LIST and implements an efficient computation of Local Minimum Distances between simplicial complexes. It is based on a polygonal representation of objects and uses a BVH (Bounding Volume Hierarchy) of pairs composed of spheres and cones. LMD++ gives a natural definition of gaps for point contacts. Optional dilatation of objects is available, allowing the implementation of safety margins for insertion tasks. With LMD++ the contact model is accurate but the computation is slower than VPS++ (it is however faster than game engines like Novodex™ or HAVOK™ for concave objects).

Both collision detection libraries are associated with the same mechanical engine called GVM 2.0 and developed at CEA/LIST. This software uses constraint based dynamics and has the main following features:

- Management of bilateral constraints (kinematic joints)
- Scleronomous binary constraints: usual joints (fixed, prismatic, hinge, helical, ball-and-socket, free) and specific exotic joints for 3D curve and 6D path constraints
- Tree-like kinematic graphs
- Kinematic loop closure.

Management of unilateral constraints (non smooth dynamics):

- LCP-based
- Contacts between rigid bodies
- Joint limits
- Cartesian and joint velocity limits
- Exact Coulomb friction.

Implementation of Proportional-Derivative control:

- In operational manifold $SE(3)^n$ (Cartesian space control, 6D springs and dampers)
- In configuration manifold (joint space control)
- Implementation of a geometric implicit integrator on Lie group.

Using GVM, VPS and LMD, XDE provides an easy to use API and a description language based on XML to construct scenes easily. This language called XDEXML allows the user to describe the scene including simple objects, multi-meshes objects, constraints, robots and manikins. This engine is widely used for the developments of prototype in Virtual Reality inside the LSI laboratory.

3.1.2 **DSimI: the Communication framework**

DSimI is a communication framework developed by the LSI. The goal was to be able to communicate between different virtual reality engines (visual rendering, physical rendering ...) with a unified protocol and without having to know where these engines were located. Since the considered modules are real time engines, there was the additional constraint of excellent performance in term of communication delays. No existing software fulfilling these constraints was identified and LSI developed its own communication framework in C++.

DSimI relies on several concepts:

- The first one is the loop: a loop is an autonomous unit (i.e. a thread) that is capable to communicate with other loops both synchronously (i.e. data is sent regularly, at a constant frequency) and asynchronously (data is sent only once often in answer to a user action: kind of event mechanism). It is the basic concept to define an engine: an engine is often embedded in a loop and run autonomously reacting to other engine information and request.
- The second concept is a concept of message/callback: this mechanism allows an engine to ask information to another engine or to make another engine modify its internal state. These communications are asynchronous and one well-known implementation of this concept is RPC (remote procedure control). This concept is divided in two parts: the message that is like typing a command in a shell (you make a command and gives it its parameters). The second part is the execution of the request: it is a single piece of code that analyses the parameters and do what the sender asked.
- The third concept tackles data sending: the engines must send to each other data with a high frequency (24 data/s for visual engine to have a good quality, and even more – around 100 data/s – for a sound engine). These data must be able to reach their destination wherever they are (on the same process or on another physical computer). DSimI implements that and have no coupling between the data, the engine and the way it is handled by the engine. This means a data can be used by different engine to convey different values and be interpreted differently (for example, a vector of position can represent the position of objects or of the centre of gravity of these objects ...).

From these basic communication concepts, DSimI allows the complete desynchronization between the engine location and the sending of messages and data. Mechanisms such as a message server, allows the designer of an engine to completely ignore where the location of other engines are.

Further during the development, a wrapper for C# was developed. In fact, it is an engine that does nothing but exchanges with others engines in order to allow addressing the different engines directly from C#. As other engines, it can receive and send data and messages. However, those action can be made in C#. This feature is widely used in the Virtual Warehouse where 90% of the functions send a message to either the physical or the graphical engine.

3.1.3 XDE Physic Pack

Alias XPP, it is a DLL designed to integrate the XDE engine inside Virtools. It is a DLL made inside the SDK Virtools to be able to make a physical simulation of a Virtools scene.

A Virtools DLL can contain two kinds of components:

- Managers are used to overrun the basic action of Virtools ; for example, to save/load your own data when saving a scene, you override the save function in a manager and it is done. It is not pure object overriding: all managers that implement a function are called when this function is called in Virtools. The classical examples are to override save, load, play ...
- Building Blocks: those are Virtools script blocks. A lot of them are already in Virtools and allow you to interact with the Virtools Scene. An example of Virtools made Building Blocks is a block to change the colour of the objects.

The XPP DLL contains a manager that contains all the DSimI protocol to create and communicate with a XDE engine and building blocks that allow the interaction of the user with the XDE engine. The manager also contains a description of the scene that will feed the XDE engine to create the physical scene. The building blocks are of three types:

- Modification of the physical simulation: allows modifying the simulation or taking instant information from the physical engine. Examples: freeze, attach, get position ...
- Modification of the scene before the creation of the simulation: allows configuring the physical engine. Examples: Disable Collisions ...
- Mass information getters: allow getting the data that come from the physical engine and render them in the Virtools scene. Examples: Render Contact Points, Get Mesh Position ...

All these building blocks can seem hard to use but all the scripts already exists and are fully operational.

3.1.4 Virtual Warehouse

This is the main component of the architecture. Its specifications are given in deliverable D5.3. Its goal is to manage the physical simulation, the visual effects and the interactions with solvers through a single interface.

The specifications have evolved since the deliverable D5.3 to allow the designer to have better way to script and to help the final user to understand the simulation.

Here will be found a description of the new classes and of the new methods.

3.1.4.1 New classes

Some new classes were implemented to simplify the life of users/GUI designer.

A class called WindowsEssential is the main component for constructing all the windows made. It allows the windows to be portable from an application to another without constructors taking the main windows as argument. WindowsEssential contains a Scene, the list of selected objects and an event that trigger when an object is selected.

A math namespace contains classes describing Vector, Quaternion and Frame (a frame being a Vector3 as a position and a Quaternion as orientation). This allows the scripters to make quick and fully comprehensible mathematical operations.

The NWMSObject class is created to reflect the objects in the simulation. Its members are its position, name and level. The equal method is based on the name of the object. Any object is dependant of the scene that created it (allowing the call of a SetPosition method when the position member is set).

3.1.4.2 Modifications on the Scene object.

The main philosophy of this class remains the same: managing all others manipulators as well as gathering/updating information from the NWMS Objects. All NWMSObjects are now created on the scene object. Furthermore, comparatively to D5.3, the history management is now located on the scene object. Regarding the new methods of this object, mainly, getters were added for an easier life:

`NWMSObject[]` GetObjects(): Get all the NWMS Objects of the scene.

`NWMSObject` GetObjectByName(`string` name): Get the object according to its name, return null if none is found.

`List<NWMSObject>` GetObjectsByLevel(`int` lvl): Get all the NWMS Objects whose level equals lvl

`string[]` GetAllPhysicalObjects(): get a list of the physical objects on the scene, not only the NWMS Objects (useful for building new scenes: see

Object Properties

`wGenericValue` GetObjectAttribute(`NWMSObject` obj, `string` attribute): Retrieve an attribute from an object (the list of attribute could be found on Data Window)

`bool` IsPhysical(`string` name): check if the given name is a physical object of the scene.

`NWMSObject` GetParent(`NWMSObject` obj): Specialisation of GetObjectAttribute, return the parent of an object (meaning the bin associated to an item) if it has one, null otherwise

A few others methods were added to ease the global management of the application:

`void AddPiece(string name, string file, Vector3 place):` add a piece to the scene from a file and place it on at the creation.

`void StartScene():` start the physicalization of the scene. Before all is purely graphical objects, any objects added after that will be purely graphical too.

Events also were added to signal the state of the simulation: started signals the beginning of the simulation, and configurationRead signals that a configuration has been read, starting the real NWMS simulation.

3.1.4.3 Simulation Manipulator added specifications

The additions to this manipulator are very light, it will be widely expanded with the D5.4 and the creation of all physical interactions associated to the VR possibilities. Until now, two getters and two modifiers:

`math.Frame GetPosition(NWMSObject source):` get the position of the object

`double[] GetBorningBox(NWMSObject source):` get a double[6] corresponding to the borning box (non oriented) of the object

`void WeldToParent(NWMSObject object1):` weld an item to its associated bin

`void Freeze(int lvl, bool freeze):` freeze or unfreeze all the objects of a level.

3.1.4.4 Visual Manipulator added specifications

This manipulator has been quite widely expanded because D5.5 focuses on increasing the understanding by the user of the information given by the solver and of the real configuration of the scene. It is quite complicated too because of the way it is used internally. The Visual manipulator often sends messages directly to the Virtools viewer and so uses an interface not dedicated ... making it harder to use than the internally developed ways.

A particular effort was made on the colour management for the future scripters to be able to render information easily as colours.

`void AddColor(string colorName, int R, int G, int B, int alpha):` add a colour in the list of colours available in the Virtools player.

`void AddTexture(string colorName, string textureFileName):` add a new texture to the Virtools player (for details, see

New Colour Window)

`List<String> GetColors():` retrieve the list of colours available on the Virtools player. (Be careful, you must have used the Refresh Colour list before.)

`void RefreshColorList():` refresh the list of colours from the Virtools Player: launch colorsListUpdated event when complete.

`void SetColor(NWMSObject obj, string colorName):` force the colour of an object.

`void WrongRuleSet ...:` there are various format for the parameters. The methods allows to set broken rules for objects. (see details in Warning and Colour Management and Wrong Rules Window). Some of these methods have a wait option, meaning all these rules will be applied when WrongRuleApply() is called.

`void WrongRulesReset():` delete the not yet applied.

`void WrongRuleApply():` apply the rules that were waiting.

`void SetCameraPosition(math.Frame frame):` Set the position of the camera.

`math.Frame GetCameraPosition():` Get the position of the camera.

`List<NWMSObject> GetSelectedObjects():` get a list of objects selected visually in the Virtools player.

`void PlaceAbsoluteGraphical(string objectName, Vector3 position):` place a graphical object visually on the scene. Trying to move a physical object with

this method will do nothing. (the object will go back immediately at its physical position)

```
event colorsListUpdated (object sender, List<String> colorsName): event
giving the list of colours available. Allow the user to know the retrieving from
the Virtools player is complete.
```

```
event select (object sender, List<NWMSObject> pickedObjects): event sent
each time an object is picked on the view.
```

Finally, creating a fully graphical object embedding the Virtools XE player and available as a graphical component in Visual Studio being a lot of hard work for a very little improvement, it was decided that the visual manipulators would be created taking as parameter the XE player. Doing so, the designer of a new GUI or window, could simply drag and drop the XE player and create the Visual Manipulator with a code line.

3.1.4.5 Solver Interface Modification

The solver modification is still going into the maturation development as the KLS front end is still evolving and the understanding of the interactions between the solvers and the Virtual Reality increases. A few configuration aspects were added to hone the utilisation of the solver (securities, URL, see Configuration and compatibility for details). A part of generation was added to be able to create a scene from a database problem, and waiting for the real geometries to be added as a parameter of the objects, a generator of file for parallelepiped objects was added. Some other developments were done on the solver but will be detailed further (see First Experiments).

3.1.5 Virtual Warehouse GUI

This is the interface of the prototype. It allows the user to understand how the scene is. This component is fully developed in the scope of Net-WMS with the exception of the Virtools ActiveX that are imported.

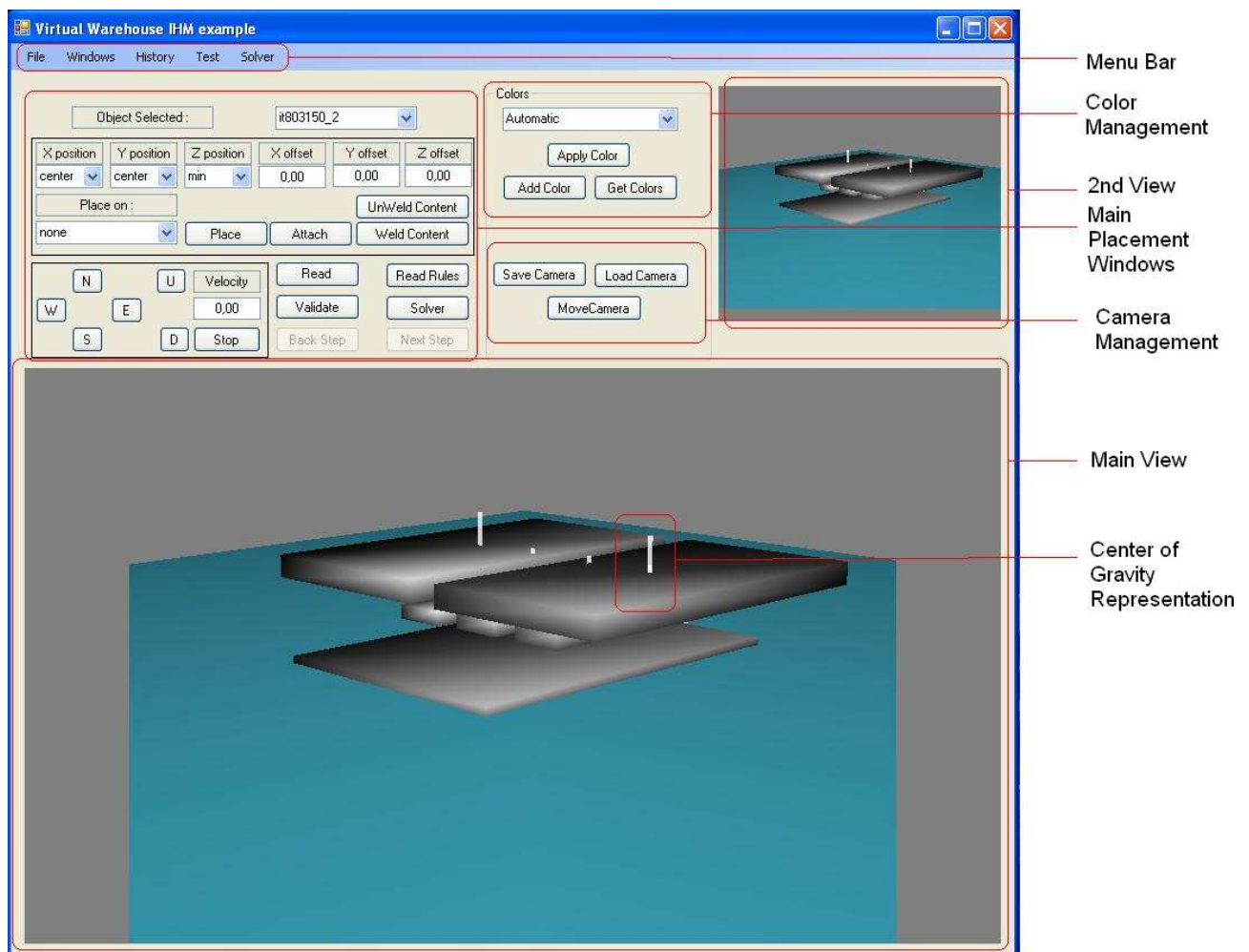
This component will be described in the next part of the report.

3.2 Man-Machine Interface

3.2.1 Here is the description of the functionalities of the GUI

The functions presented here will be very close to those of the final prototype given to the end-users. This part describes the main windows, its menu and the others options windows. One last window will be detailed in another section. It is the configuration window for the solver interface (Cf. Configuration and compatibility in Solver Interface part).

3.2.2 Main Windows



The main windows are the interface the user gets when he launches the application.

There are two Views that can show different aspects of the same scene. These views represent the objects as they are in the physical engine. Their colours indicate the different rules that they are violating. (Here the colours are not activated; this will be detailed in Warning and Colour

Management). The views also show the centres of gravity of the objects. These can be hidden with an option (cf. Physics Window). All the commands that deal with the graphical display (colours ...) are basically applied to the main views. There is an option (Cf. Option Window) that allows changing the views where the user will do the modification.

The global interface allows the user to easily move objects with the placement part of the interface. (Cf. Object Placement Window for details of the functionalities).

The Colour Management allows the user to force the colours of object in order to have a clearer view of the scene. (Cf. Camera Management window for the functionalities.)

Furthermore, the main window has buttons to place the camera of the views (allowing the user to make cuts for example) and save or load this placement (Cf. Camera Management Window).

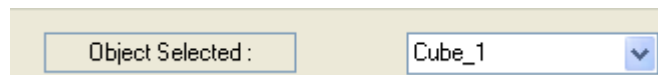
Most of the commands of the main interface will be describe inside the description of the windows because the same functionalities are accessible there.

Anyway, a few buttons are unique to the main windows:



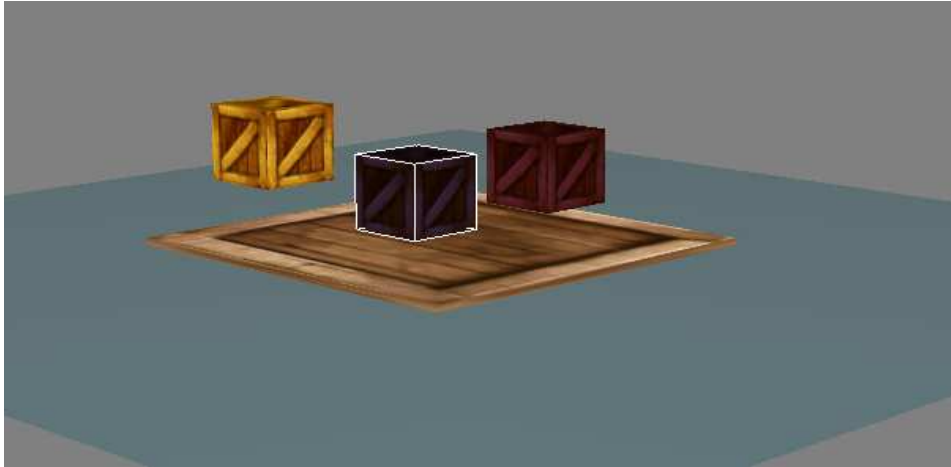
- **Read:** This button allows the user to configure the scene. It reads a file that configures objects with their level, their weight ... such files can be edited by hand or with the editor in Object Properties Window.
- **Read rules:** This allows the user to configure which rule is associated to which colour. It loads a file of description that is editable in the Warning and Colour Management Window.
- **Solver:** This button calls the solver to have the best placement solution. Further details will be given in the Virtual Warehouse-Solver Interface part of the report.
- **Validate:** This is the main button that allows history management. The history consists in fact on steps that can be loaded or saved. Validate saves a step and goes to the next step without loading it.
- **Back Step:** Allows the user to go backward into the history. Each use takes the previous step and loads the configuration.
- **Next Step:** Allows the user to go forward into the history. Each use takes the next step and loads the configuration.

The following part allows you to see what the selected item is and to select an item without needing to have a sight of it. Most of the commands or manipulations used this selected item, even if the commands are in other windows. So be careful to the selected object before applying your modifications.

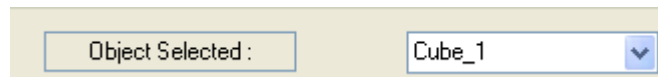


3.2.3 Shortcuts and Selection

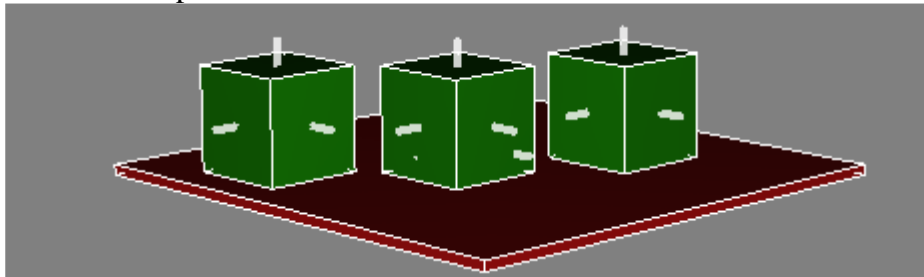
The first thing to understand is how to select an object.



Clicking on an object will select it. If the object is a Net-WMS object (i.e. bin or item, not decorum), its name will appear in the object selected combo box:



You can choose to select more than one object. In this case, only the first selected will appear in the box. Anyway, very few functions deal with more than one object, the first selected is the one chosen in case of multiple selection.



There are also several shortcuts that can be useful, all these shortcuts are similar to the placement functionalities that will be seen in Object Placement Window.

Ctrl+F: freeze all objects selected.

Ctrl+Shift+F: Unfreeze all selected objects.

Ctrl+A: Attach the object to the first device of manipulation.

4,5,6,8,page Up and Page down: move the selected objects in a given direction. Each of those moves is relatively to the camera of the main window. 8 is away from the point of view, 5 is backward, 4 to the left, 6 to the right, Page up, is up and Page Down is down.

When using those keys, trace will be shown to help you reminded of what happens:

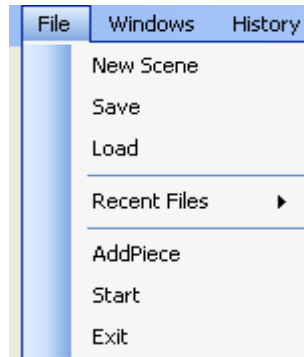
```
N Z S Z W Z W
```

N means north and is the 8 key; S, south and 5; W, west and 4 ; E, east and 6. U means up and D down. The Z means zero and means the object stopped moving.

3.2.4 Menus

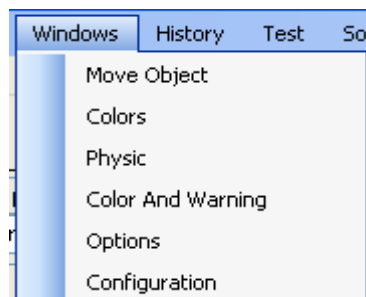
Here are the different menus and what functionalities they contain.

3.2.4.1 File Menu



- **New Scene:** Reset the interface and load an empty scene. Such an empty scene can be filled dynamically using an object Maker window (see Virtual Warehouse-Solver Interface) or from a problem taken from database.
- **Save:** Save the scene in a vmo file. It will be reloadable with the load button. The configuration files (cf. Colour Management Window) will not be saved with the vmo.
- **Load:** Load a file saved from the Virtual Warehouse GUI or saved from the Virtools Dev.
- **Recent Files:** List of recent files loaded: click on one to load it.
- **Start:** Begins the physicalisation of the scene. All the object visible will from now have a physical behaviour. Any further addition of object will be purely graphical. Start also unlocks various buttons and functionalities.
- **Add Piece:** Add a piece to the scene. Open a file dialog that will be loaded into the scene. If you have started the physicalisation, the object loaded will be purely graphical.
- **Exit:** Leave the application.

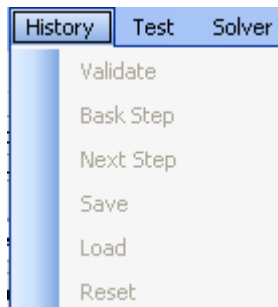
3.2.4.2 Windows Menu



- **Move Object:** Open the Object Placement Window
- **Colours:** Open the Colour Management Window
- **Physic:** Open the Physics Window
- **Colour and Warning:** Open the Warning and Colour Management Window

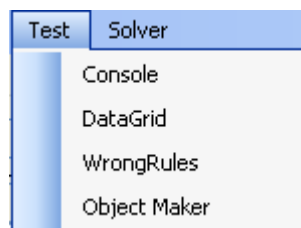
- **Options:** Open the Option Window
- **Configuration:** Open the Object Properties Window.

3.2.4.3 History Menu



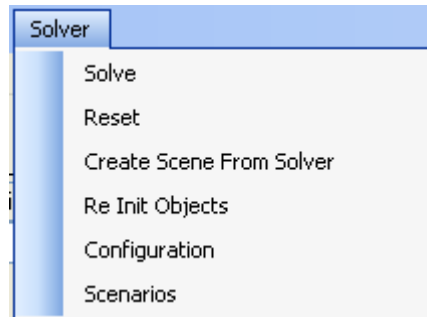
- **Validate:** Same as the Validate Allows history management. The history consists in fact on steps that can be loaded or saved. Validate save a step and go to the next step without loading it.
- **Back Step:** Allows the user to go backward into the history. Each use takes the previous step and loads the configuration.
- **Next Step:** Allows the user to go forward into the history. Each use takes the next step and loads the configuration.
- **Save:** Save the history
- **Load:** Load a history previously saved
- **Reset:** Delete the history in memory. Allowing the user to build a new history.

3.2.4.4 Test Menu



- **Console:** Show a Console Window. Allows every command to the physical engine. It is mainly a debug feature.
- **DataGrid:** Show a window containing all the available information from the items of the scene. It contains their position, their level, their weight (supported, tare weight...), their bounding box, their parent (on which bin an item is), their frozen state...
- **WrongRules:** Open a window that allows the user to set rules broken by objects. Allows tests on colours and ultimately emulate a check on which rules is broken by a solver.
- **Object Maker:** Open a window that allows the user to create parallelepiped objects and add them to scene. Main way to create a scene from scratch. Emulate the automated building from a database problem.

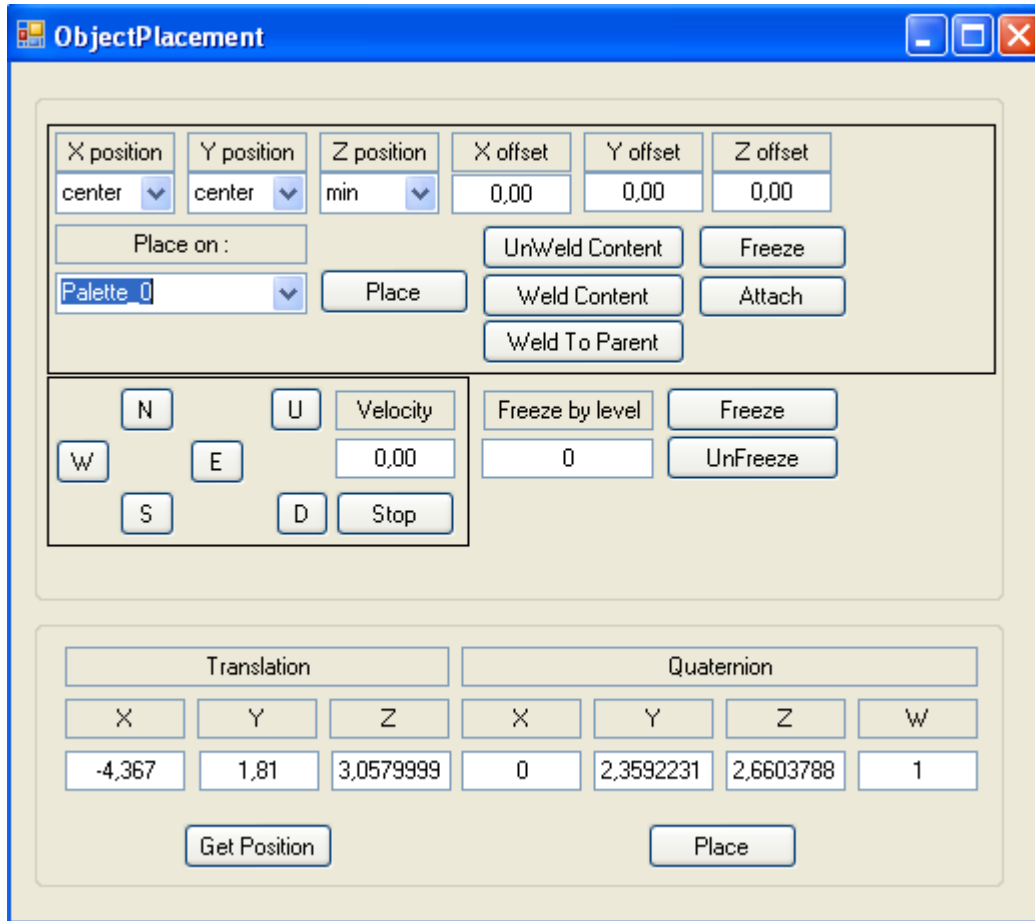
3.2.4.5 Solver Menu



- **Solve:** Call the solver to give the objects the calculated optimized position
- **Reset:** Reset the Server
- **Create Scene From Solver:** Create the scene corresponding to the solver problem
- **Re Init Objects:** Allow the user to reset the states of the objects inside the server. This is particularly important when working with static bounding boxes (Cf. Virtual Warehouse-Solver Interface).
- **Configuration:** Open a Configuration Window for the Solver, allowing the user to change: the URL of the solvers, the securities, others options (bin level ...)
- **Scenarios:** Open a Window that allows the user to manage the scenarios in the database (load them, modify them, delete them).

3.2.5 Object Placement Window

This Window is accessible in the menu Windows by the tile "Move Object". It represents every way of moving an object in the scene, as well as manipulation on parents or levels. Some of these functions are also implemented in the main window.



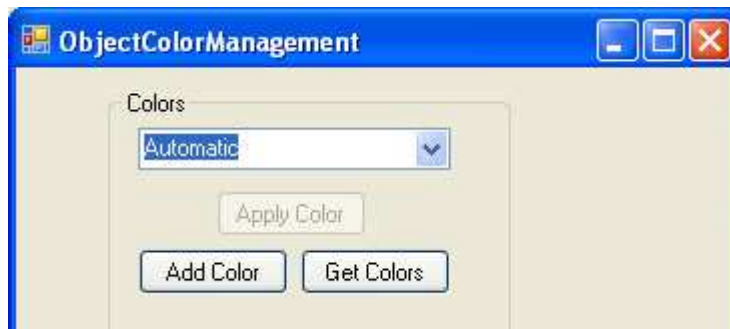
Most of the functionalities of this window deals with the selected object so be careful on the main window when using the Object Placement window.

- **Place:** This is one of the most complicated and useful feature of this window. It allows the placement of an object relatively to a bin or another object. The target object is defined by the combo box "Place on" that contains a list of the possible targets. The parameter on the position allows you to place the object where you want: min on the Y position targeting a bin means the min Y corner of the bin will be next to the min Y corner of the selected object. An algorithm prevents the selected object to be put on already existing objects, in case of such a problem, the item will be placed next to the already placed objects. The placement relatively to a target of the same level is quite different as the selected item will be placed directly next to the target (for example: centre, centre, max will place the selected item directly upper the target; the parameters centre, centre, centre has no meaning in this case (that would mean the target and the selected items are on the same place), so it is not allowed). There is also the possibility to put an offset that will move the objects once the place is determined. For more information on this, see D5.3.
- **Weld Content:** This option freezes every object contained on a bin. If the selected object is not a bin or contains nothing, nothing happens.

- **Unweld Content:** reflexive function of the previous one. It unfreezes all the objects contained in a bin.
- **Weld to Parent:** "Attach" the selected item to the parent on which it is. It freezes the object on its bin.
- **Freeze:** Freeze an object. It will be impossible to move. If it is an item associated with a bin, it will stay associated with the bin.
- **Attach:** Attach a 3D device to the selected item. It allows moving the selected item with a space mouse for example.
- **N, S, E,W, U, D, STOP:** these buttons have all the same purpose. They move the selected object in a direction at a constant speed. The directions are absolute (i.e. independent of the camera position). The speed is the velocity value. The stop button stops the displacement. N means north, U is up and D is down.
- **Freeze/Unfreeze By Level:** Freeze or unfreeze all the object of a given level (in Level text box).
- **Get Position:** Get the absolute position of the selected item.
- **Set Position:** Set the absolute position of the selected item. Be careful, there will be no check of interpenetration in this case.

3.2.6 Colour Management

This window allows the user to define and change the colour of the selected object.



- **Get Colours:** get the list of the colours that exists in the scene.
- **Add Colour:** Open the New Colour Window: the goal is to create a colour that does not exist yet.
- **Apply Colour:** Apply the colour to the selected object.

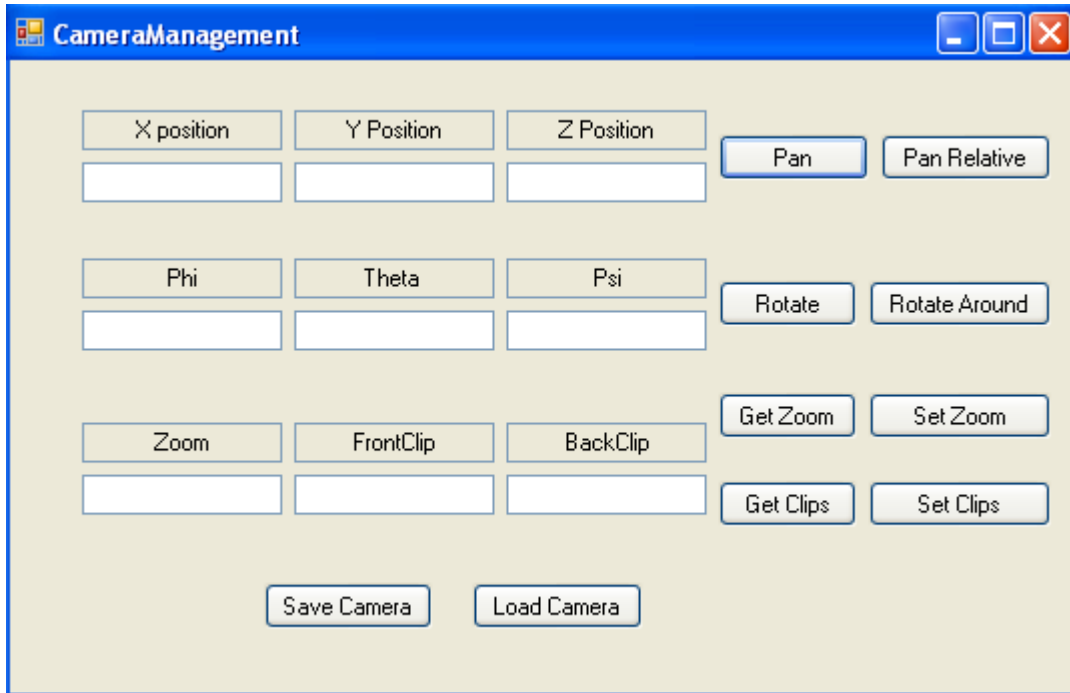
The way of using the colour management is the following: Get the Colour List present in your scene first. Add the colours you want (you can both create colour from RGB values or from a texture). Finally apply the colour to the objects you want to see.

There are two special colours:

- **Automatic:** the object gets the colour according to the warning level it has. (Cf. Warning and Colour Management).
- **Invisible:** The object becomes invisible. Still, it can be selected by the combo box or by clicking on its position. The selection will make its bounding box visible in white.

3.2.7 Camera Management

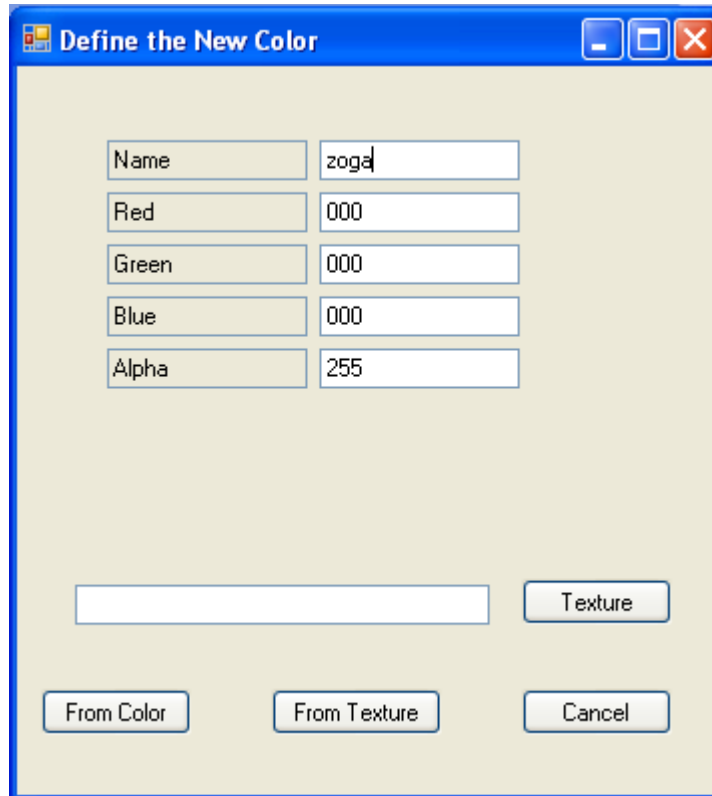
This window allows the user to have finer control over the camera.



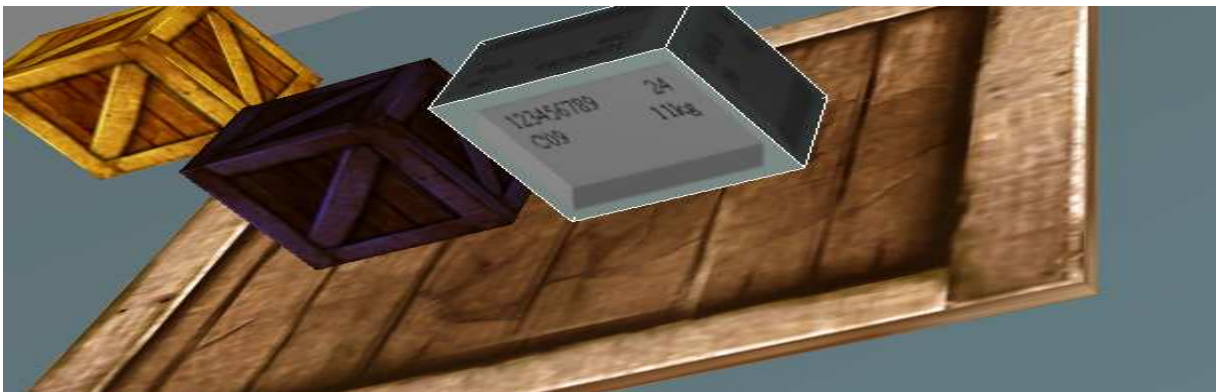
- **Pan:** Move the camera of a vector (x,y,z).
- **Pan relative:** Move the camera of a vector (x,y,z) relatively to the current camera.
- **Rotate:** Rotate the camera around its own axe of a theta angle (theta is in radian).
- **Rotate Around:** Rotate the camera of angles (theta,phy,psi) around a point situated in (X,Y,Z).
- **Get Zoom:** Get the Zoom Value of the Camera.
- **Set Zoom:** Set the Zoom Value of the Camera.
- **Get Clips:** Get the front clip and Back Clip values of the camera.
- **Set Clips:** Set the front clip and Back Clip values of the camera.
- **Save Camera:** Save the position of the Camera. Can be restored any time with Load Camera.
- **Load Camera:** Load back the previously saved camera position.

3.2.8 New Colour Window

This window permits the user to create new colours. These colours are in fact Virtools materials. There are two kind of materials. First is material of pure colours, the second is material made from texture. The user is able to create both materials with the following window:



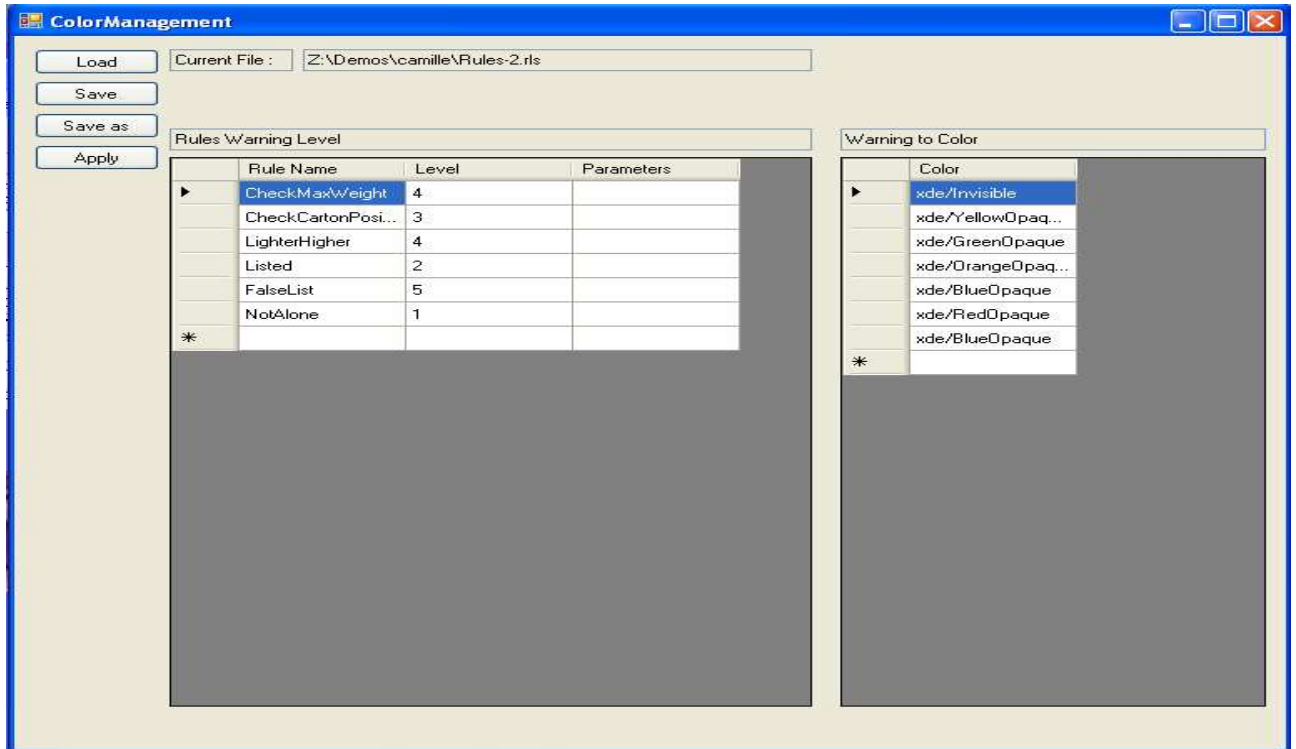
- **Texture:** find a picture file that will be the texture.
- **From Colour:** Create a colour from RGB and alpha values. Its name is Name.
- **From Texture:** Create a material from the texture.
- **Cancel:** Close the window.



Textured Objects.

3.2.9 Warning and Colour Management

This window allows the user to manage the colour of the objects according to the rules broken. Each visual object received a list of rules that they do not satisfied, either from C++ part inside XPP or from the solvers via the “WrongRuleSet” method in the Virtual Warehouse API. This window associates a level of warning to each rules and a colour to each level of warning.



Buttons:

- **Load:** Load a file of configuration. Usually it is a .rls file. The name is printed in the current file text box.
- **Save:** Save the Current file or open a dialog to save a file if the current file is not set.
- **Save as:** Open a file dialog to save the file under a new name. (Change the current file).
- **Apply:** Apply the Current file to the main View (unless you choose the other view in the Option Window).

In the window, the Rule warning Level box associates the level of warning with the rules. A certain set of rules are already implemented in the C++ part of the Virtual Warehouse API:

- **CheckMaxWeight:** verify if the weight supported by a bin is not more than the supported weight.
- **CheckCartonPosition:** verify that an item is not outside of the bin it is associated with (Cf. Deliverable D5.1)
- **Lighter Higher:** Verify that, if two items of the same level on top of each other, the lighter object is at the top
- **Listed:** Check if an item is associated with a bin (see Connexity in Deliverable D5.1)

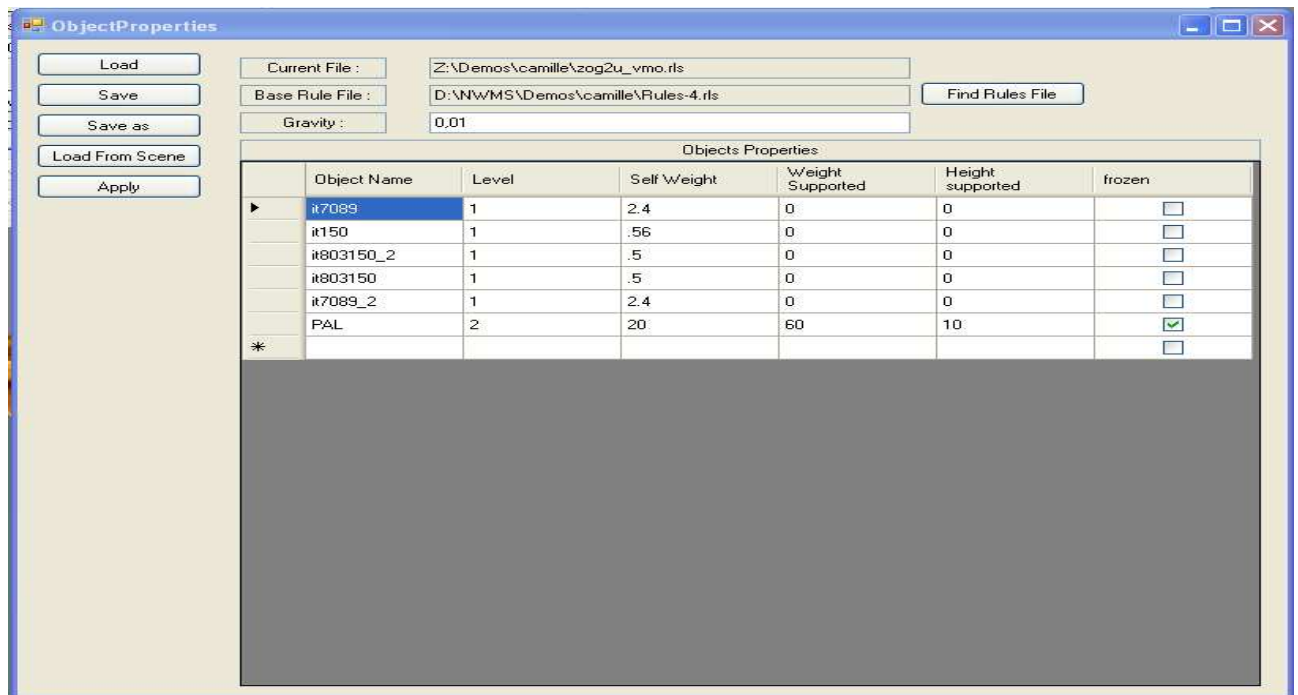
- **FalseList:** Check if the item is not in the invalid list (contain object in equilibrium between two bins, or object of more than a level of difference in contact)
- **Not Alone:** Check if the object is in contact with another. If it is not alone, the rule is broken.

The user will be able to define other rules as the PKML rules that are processed by the solver. Once the rules Warning are set, it is time to associate a colour to each level of warning.

The second part of the windows (Warning to Colour) is the association between a colour and a level of warning. This is a list of colour, the first colour has number 0, meaning it is the colour of objects that break no rules. The second colour is associated with level 1 ... etc

3.2.10 Object Properties

This window allows the user to give the objects their weight, level ...



Such a file contains a base rules file, a gravity value and a list of objects with their parameters.

The parameters of an object are its level, its weight, the weight it can support (often 0 for the items), the height that the object can support (idem, it is for bins) and if the object is frozen at the beginning of the simulation. Any object that you want its name saved but not taken as a NWMS object (physical decorum), may have its level put to -1.

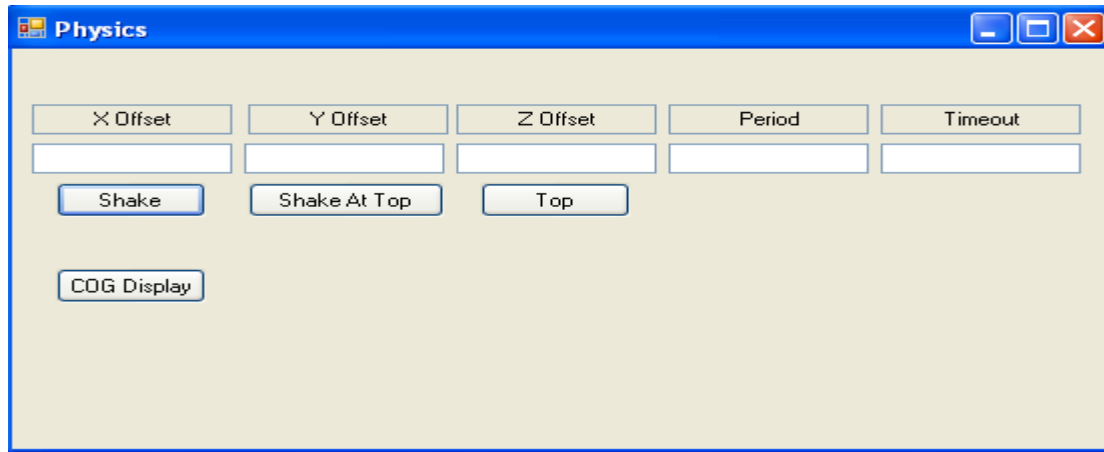
Buttons:

- **Load:** Load a configuration file. Usually it is a .rls file. The name is printed in the current file text box.
- **Save:** Save the Current file or open a dialog to save a file if the current file is not set.
- **Save as:** Open a file dialog to save the file under a new name. (Change the current file).
- **Apply:** Apply the Current file to the current scene.
- **Load From Scene:** gather information from the scene to make the editor match the current configuration. When constructing a scene from scratch, it also allowed initializing the name of objects...
- **Find Rules Files:** Open a dialog to find a rule file (Cf. Warning and Colour Management) to add it as a basic rules file.

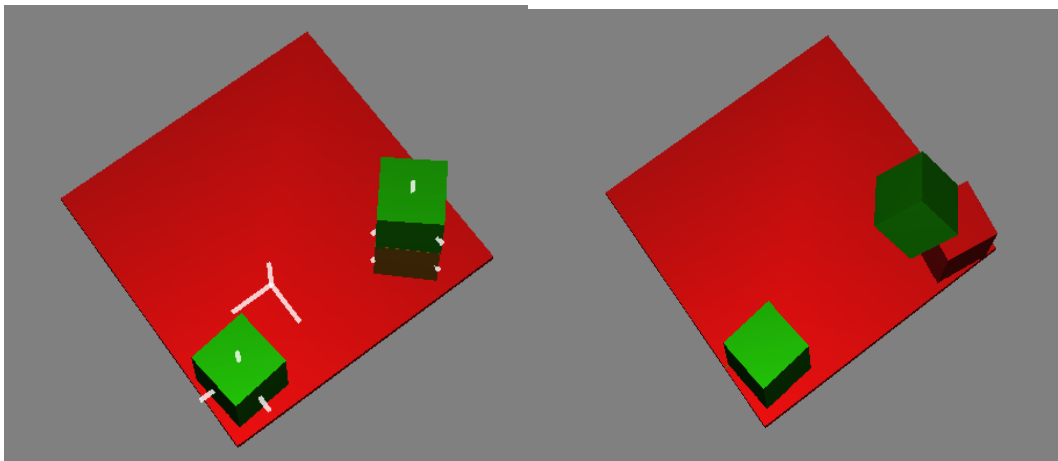
Remarks: the Load From Scene button gathers both information from existing NWMS objects but also lists the others physical objects (with level -1), it allows the user to retrieve all objects in the scene and give them a role. It is especially useful when devising a new scene: the objects created have no role yet so you retrieve all their names to give them their weight, level

3.2.11 Physics Window

This window is the reflection of the first implementation of the physical functionalities needed by the end users. It will be completed with the elaboration of D5.4, and a better understanding of what might help the end-users to test the validity, stability, efficiency of a structure.



- **Shake:** make an object move around its current position at a high speed. Emulate the shaking of an object or the movements made by an object in a boat or in a moving truck. The X,Y,Z values are the distance the object will do at every step. The period in millisecond is the time between two changes of direction of the object. Timeout is the time in seconds during which the object moves.
- **Shake At Top:** allow the user to prepare the shaking for an object without activating it. This allows beginning the shaking of several objects at the same time.
- **Top:** begin the shaking of all waiting objects.
- **COG Display:** Activate or deactivate the display of the centre of gravity of objects.

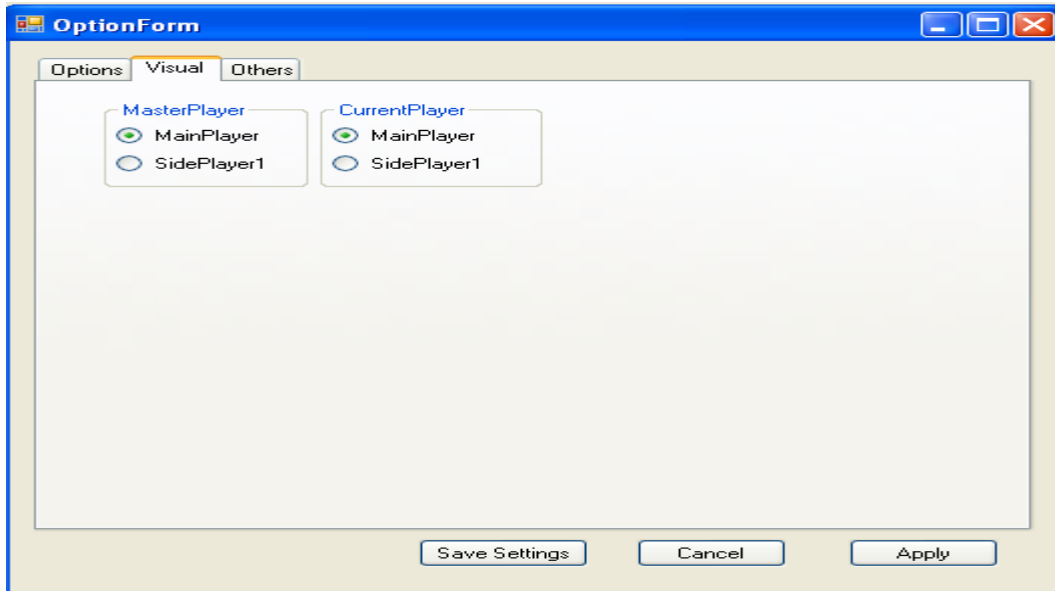


Main view with or without the visualisation of the centre of gravities.
On the first view, we see that the centre of gravity of the palette is "moved" by the position of the items on the lower part.

3.2.12 Option Window

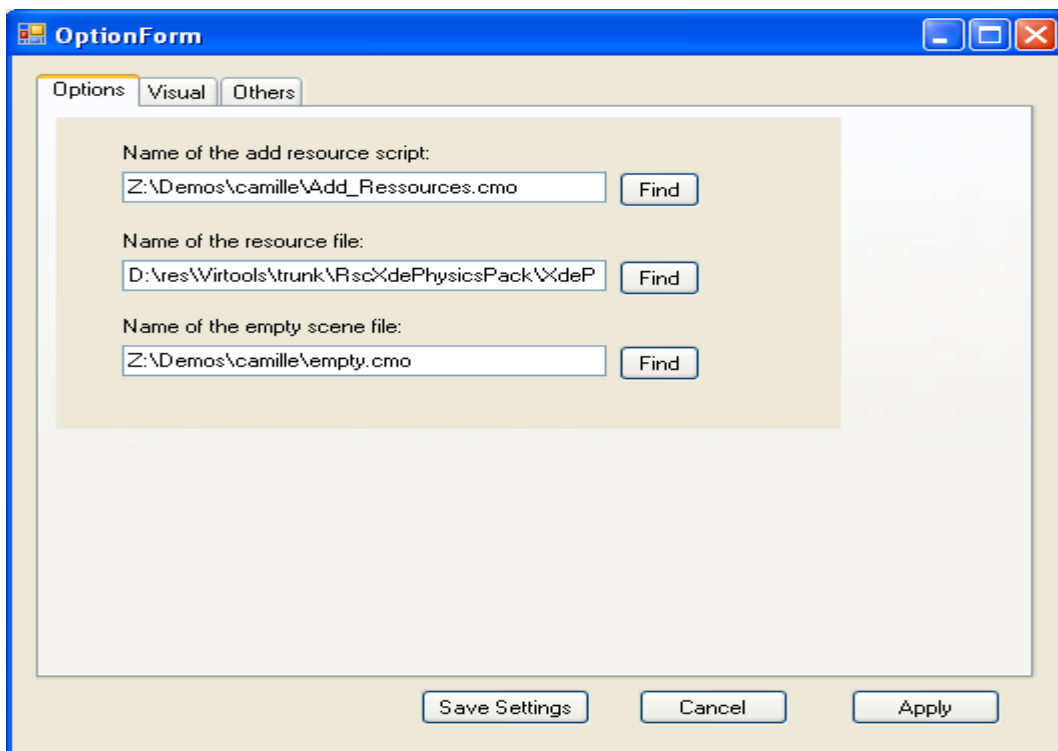
This window allows the users to set global variables of configuration that will permit the application to work (for example, the Virtools resources directory ...). This is a tabbed windows with currently three tabs with one empty.

- **Visual Tab:**



This tab configures the visual objects priority. The current player is the player to which the commands for the views will be applied (read rules, COG Display, Apply Colour ...). The Master Player is the player which camera is used for every action that needs a camera (objects movement by arrows ...).

- **Option Tab:**



This tab configures the scripts and the resources needed.

- The Add Resource script is the CMO that loads the resources into the XE players (it is often at the same place than the resources file of Virtools).
- The Resource File contains all the scripts that are needed to launch a physical simulation.
- The Empty Scene is the CMO that is loaded when you click on the new scene button or when you try to make a scene from scratch.

Buttons: Apply save the configuration (on each tab), Save Settings does the same and closes the window. Cancel simply closes the window discarding the modifications.

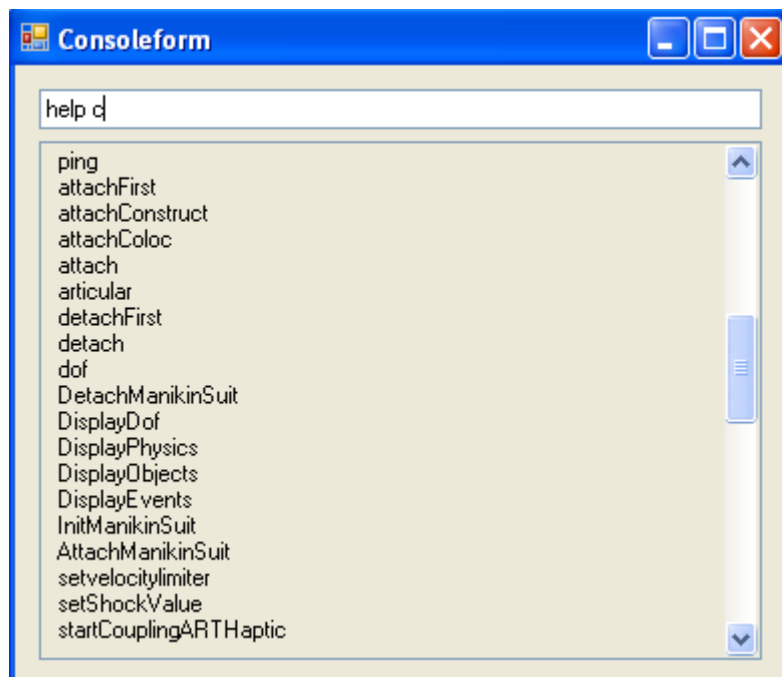
3.2.13 Test Windows

Test windows are mainly used for debug purposes. Some also emulate the behaviour of future elements. There are four test windows:

- a console,
- a window recording all the variables associated with the objects in the scene (weight supported, position, bounding boxes ...),
- a window for emulating the future solver function that will check the rules (it allows testing the Virtual Warehouse API to additional broken rules that do not come from C++),
- a window that creates parallelepiped objects and add/place them in the scene.

3.2.13.1 Console Window

This window is a simple text console that allows direct actions on the physical engine. The help command gives the list of commands and their parameters.



It supports more commands than those contained in the Virtual Warehouse API but it should not be used except for debug purposes.

3.2.13.2 Data Window

This window is used to visualise all data necessary for the physical simulation that are transmitted to the C# interface.

Object Name	pos_z	pos_x	pos_y	Max weight supported	Weight supported	rot_x	rot_z	box_max_y	rot_w
Cube_0	3,057999999552...	7,866653725486...	5,587283353761...	0	0	0,706678754643...	0,024599547466...	7,721514072422...	0,706678754643...
Cube_2	-17,5780765209...	53,566525905944	53,34810844420...	0	0	-0,52262433031...	-0,13792762264...	56,75096829480...	-0,44336656480...
Palette_0	0,838	0	-0,912	50	109	0	0	9,088	1
Cube_1	3,0378989868753	-8,15167891880...	7,223324490536...	0	0	-0,03004967510...	0,669607492111...	9,434737123691...	0,040321291009...

It is an Excel sheet that presents the object and their attributes: their position and orientation, their weight, max weight supported and max height supported, the present supported weight, their bounding box, their level, their parent if they have one, if they are frozen and if they are welded to their parent.

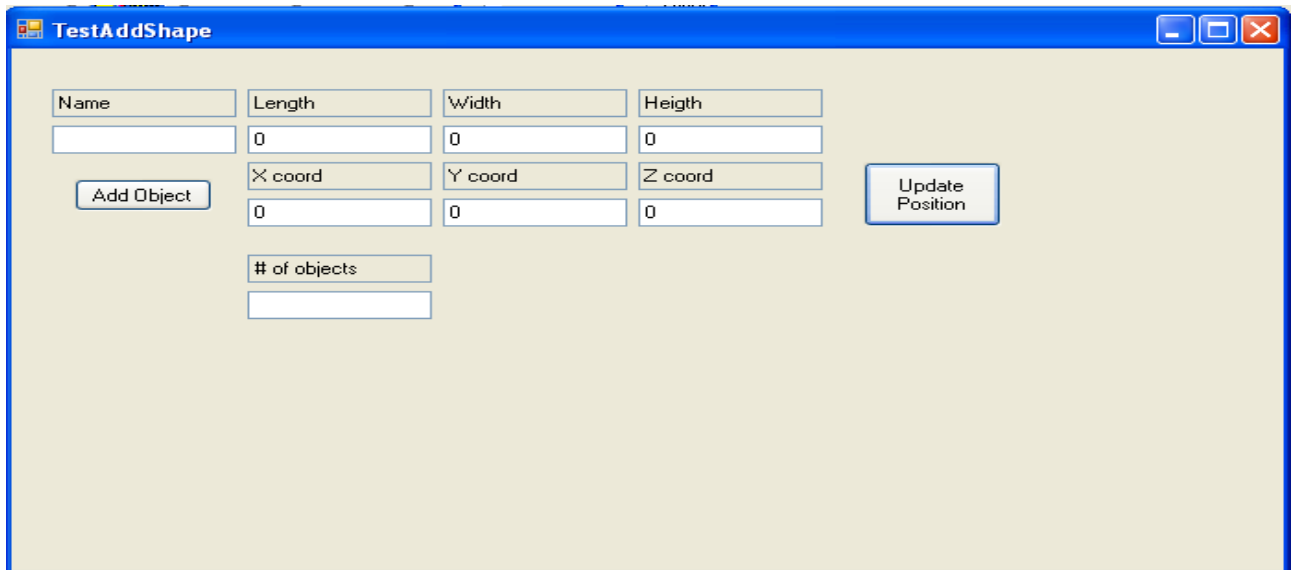
3.2.13.3 Wrong Rules Window

This window emulates the future comportment of the solver module that will check every rule to tell if an object breaks a rule or not (with custom rules from PKML and not only C++ hard coded rules). It is only here for debug/test purpose.

You choose one or two objects and add rules that they break. You can choose either to apply them at once or at a later point. The Validate button then applies all the rules that are stored, whereas Reset deletes them.

3.2.13.4 Object Maker

This window allows the user to create objects and add them to the scene. This is useful to create a scene from scratch. It also emulates the creation of objects from a database.



The screenshot shows a window titled "TestAddShape" with a blue title bar and standard window controls. The main area is light beige and contains several input fields and buttons. At the top, there are four input fields labeled "Name", "Length", "Width", and "Height", each containing the value "0". Below these are three more input fields labeled "X coord", "Y coord", and "Z coord", each containing the value "0". To the left of the "X coord" field is a button labeled "Add Object". To the right of the "Z coord" field is a button labeled "Update Position". At the bottom, there is an input field labeled "# of objects" which is currently empty.

A new object is defined by its name (if the name already exists, the creation fails), its length, width and height. You can either add coordinates to this object in order not to superpose every object, or update their position at a later point. Finally, you can add more than one object of the same type at the same time. They will be named: name, name_1, name_2 ... and will be aligned along the X-axis.

3.3 Virtual Warehouse-Solver Interface

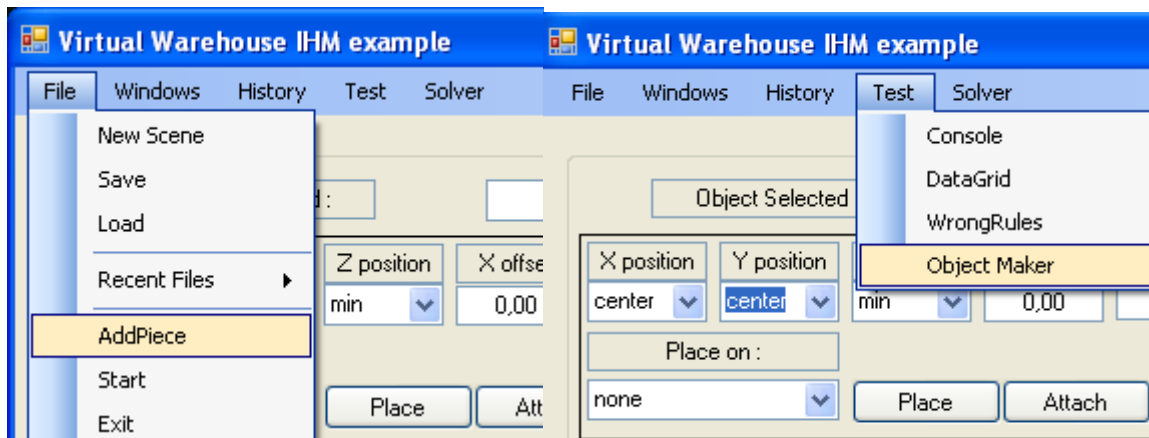
We describe here how the solver and the Virtual Warehouse work together. It is important to keep in mind that the solver is a Java object that contains a description of the scene and is called/filled via web services.

3.3.1 Starting a Scene

This is an explication about the main step to use the prototype.

3.3.1.1 Pure Graphical

When you load a scene or decide to create an empty scene. You begin with a purely graphical scene. There is no physics, no role (it is unknown whether the considered objects are items or bins). You can add objects with the menu or with the Object Maker.

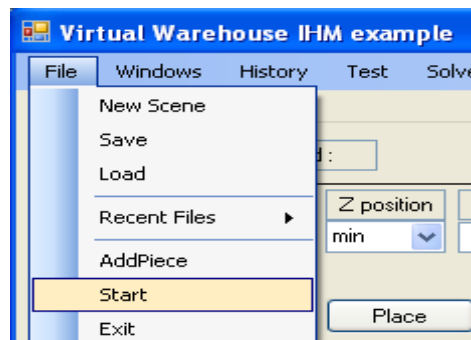


Menus to add objects in a graphical scene

Be careful, all objects added at this time will be associated with a physical behaviour during the next step. If you need to add a purely graphical object, it is better to wait after the "physicalisation" process.

3.3.1.2 Physicalisation


Once you are satisfied of your scene, it is time to physicalize it.

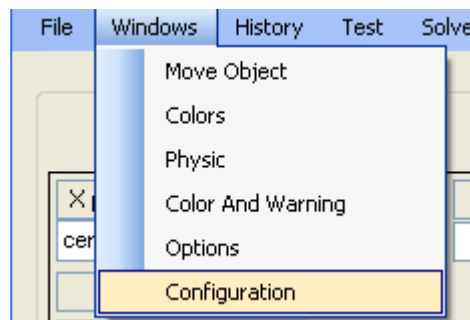


Start physicalize the scene

Now you can move your object with a navigation device like a SpaceMouse (click on an object and Ctrl+A to attach). All the objects are initially frozen and with no weight. That means you will need to unfreeze them before being able to move them (click and Ctrl+shift+F). You can still add objects to the scene, but from now on, they will be only graphical (if you want to create a nice decorum, this is the right time).

3.3.1.3 NWMS Simulation

Until now, all the objects have undefined roles (we know neither if they are bins or items, neither their weight...). The mechanism to define this is reading a configuration file. This file is also editable by hand and it is advised to use the editor given by the Object Properties Window. If an existing file has been loaded, you can read directly it with the  button on the main window. If you are building a new scene or want to edit it by hand, open the Object Properties Window:



The editor allow you to retrieve the name of all the objects of the scene to make your configuration file (just suppress the objects that have no special roles such as obstacles).

Once the roles are set, the weight is activated, making your (non frozen) objects fall. The Frozen objects will also be unmoved by the solver but taken into account nevertheless.

3.3.2 Configuration and compatibility

The solver is a very generic tool and presents a fair number of particularities that can be misunderstood and cause problems with the interface.

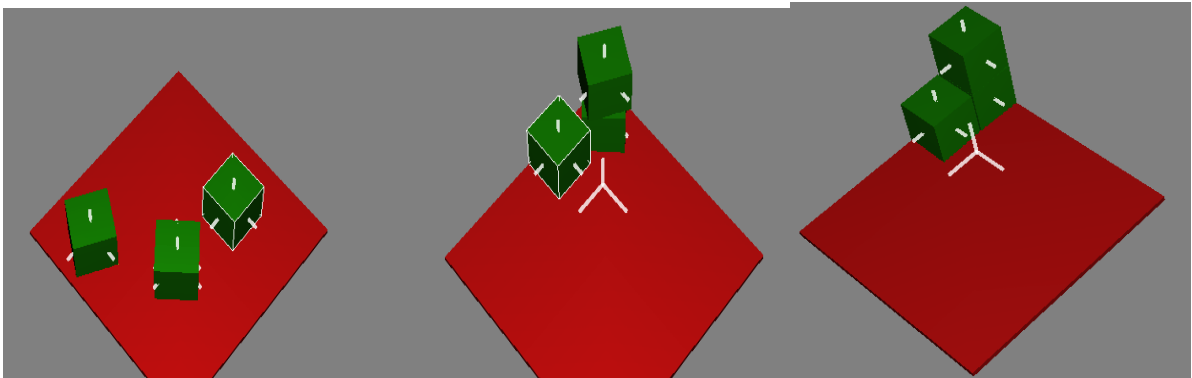
The first problem is that the solver deals only with integer and no floating numbers. That is a problem to give him data because the scale of the objects in the physical engine is the meter (to place an object with a possible error of a meter is an absurdity). We decided with KLS partner that a precision in millimetres would be sufficient. Therefore, all measures are expanded by a factor of a thousand.

The second point is that the physical engine needs a certain gap of security between object. It is a matter of collision detection. If objects are too close, there is a risk that contacts are not detected (due to errors in floating values ...). On the opposite, the solver, to have the best solution, always puts the objects side by side. The two being incompatible, the decision was made to slightly inflate the objects to prevent such problems. In doing so, the solver will give a side by side result that will correspond in reality to a quite separated solution that satisfies the physical engine. Such a security value is modifiable in the configuration window (see later).

One point is the very use of the Web-services. Having a solver running on another computer (or the same) and taking the associated WDSL to make the application, what will happen if we move the service? This problem is bound to appear once in the deployment phase, when the solver could be installed on the customer's computer. Therefore, we had a configuration option in the window to be able to change the URL of the service.

The last compatibility problem is the orientation of the objects. At the present time, orientations are not taken into account by the solvers that can only deal with a limited numbers of shapes associated to one single object (typically, rotated by 30°, 60°, 90° ...). This is not very disturbing for cubic forms but for complex objects, the problem is serious (how to put a pipe of car at a 30° orientation). The decision was taken to send to the solver only the bounding boxes of objects. There, we have two compartments that are both valid:

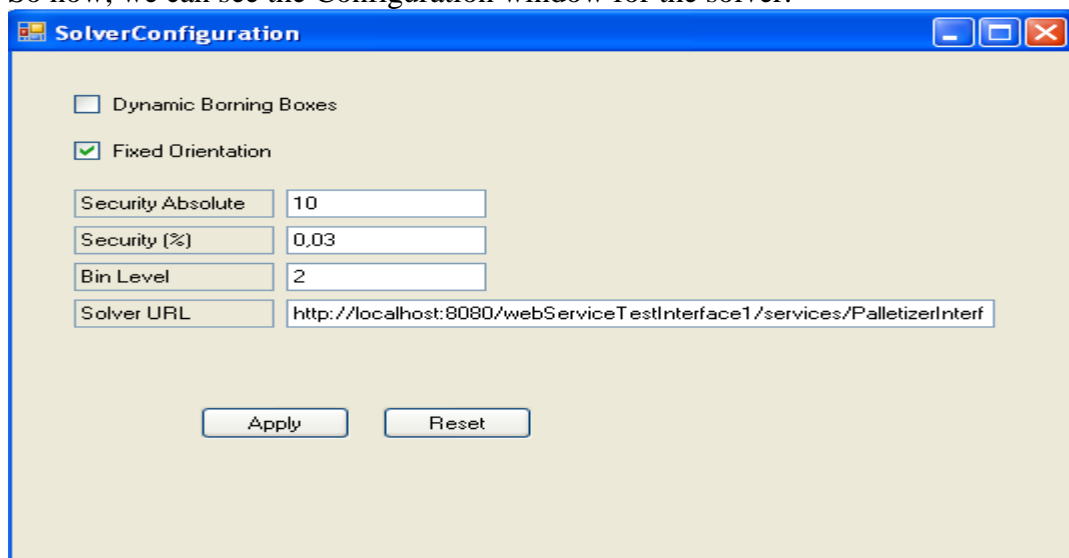
- Send each time the Boring box of the real position of the object. The solver can turn it (or not) but the object will not be laid flat (or very unlikely). We call this dynamic boring boxes.
- Memorize the boring boxes when the objects are flat so that we always have the perfect boring box. Then, when we call the solver the objects are always laid flat but the user would be able to put an object at 30° and keep this orientation and keep it on the solver.



A configuration and the results of the solver in case of dynamic boring boxes or not

As both solutions are valid, an option can switch from one mode to another. In the case of no dynamic boring boxes, a mechanism was also designed to be able to reinitialize the boring box of an object.

So now, we can see the Configuration window for the solver:



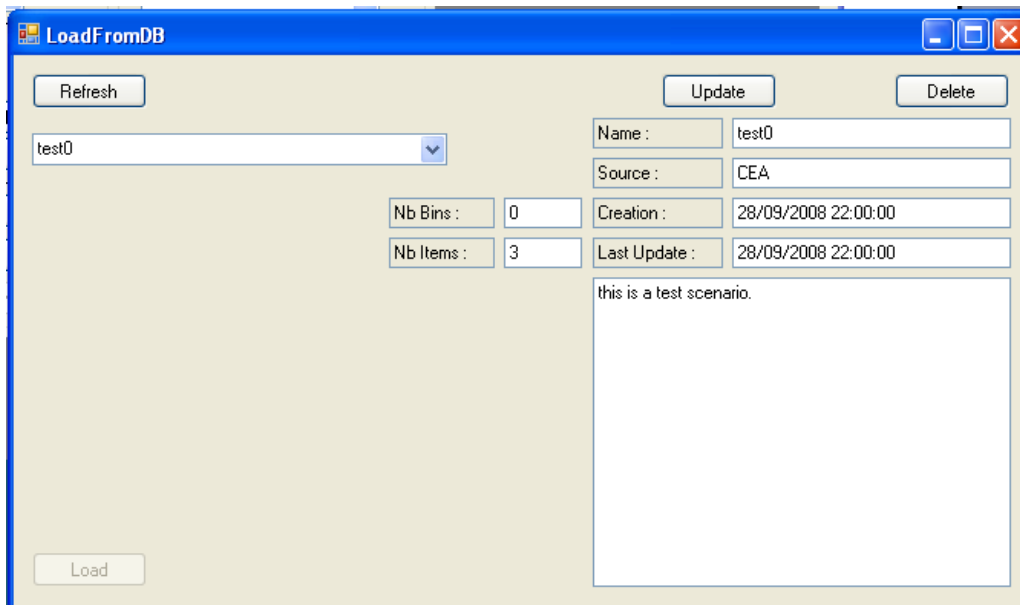
The two flags allow to change the mode to dynamic bounding boxes or not, and if the solver is allowed to change the orientations. There are two security measures: an absolute one (flat add to the size of every objects) and a relative one (each object is inflated of n%).

The bin level is the level we consider as bins (right now, we cannot have more than two levels of objects).

Finally, the URL of the service can be changed. When loading the window, the values are loaded automatically. Click on Apply to make the new ones effective and Reset otherwise.

3.3.3 Loading a problem from database

This section explains how to load a scenario present in the database. There is a window specialized for that. It can be open by clicking on Scenarios in the menu Solver. This window is temporary and the mechanisms used are very likely to evolve with the future integration of the whole architecture in J2EE in WP8?



This window allows the user to modify the information relative to a scenario, to load them or delete them.

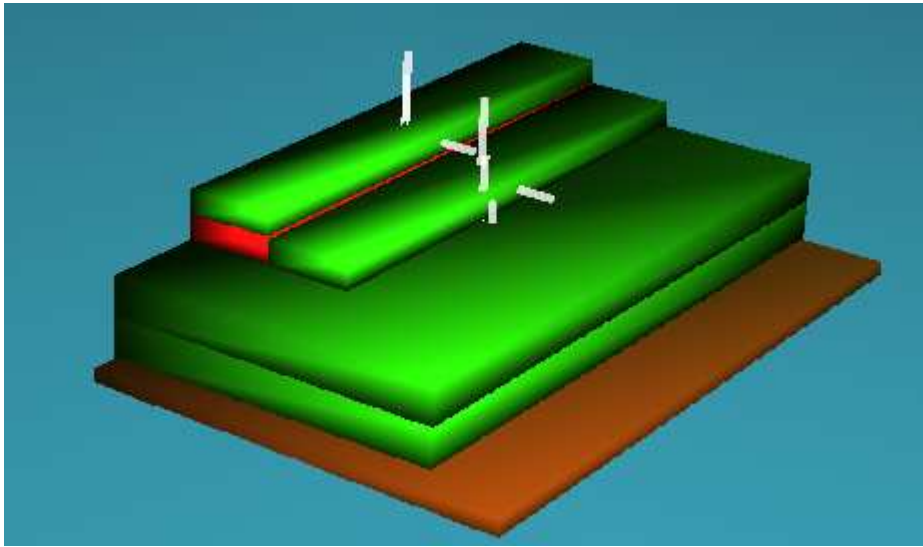
You can choose the scenario from the combo box and edit the fields relative to it. To save your modification, use the Update Button. To delete the scenario, use the Delete button. If your database has been modified, the Refresh button will refresh the scenario list.

Finally, the Load button allows you to take a scenario in the database to load it in the scene. The objects will be loaded in the views. An Object Properties window will pop up containing the information of the items and bins. Then you can start the physicalization and apply the parameters to the scene.

3.3.4 Examples of use

3.3.4.1 Best Positions

This is the simplest way to use the solver to interact with the scene. You get a bin and several items and call the solver via the solver button. All objects are instantaneously placed in the best position computed by the solver.

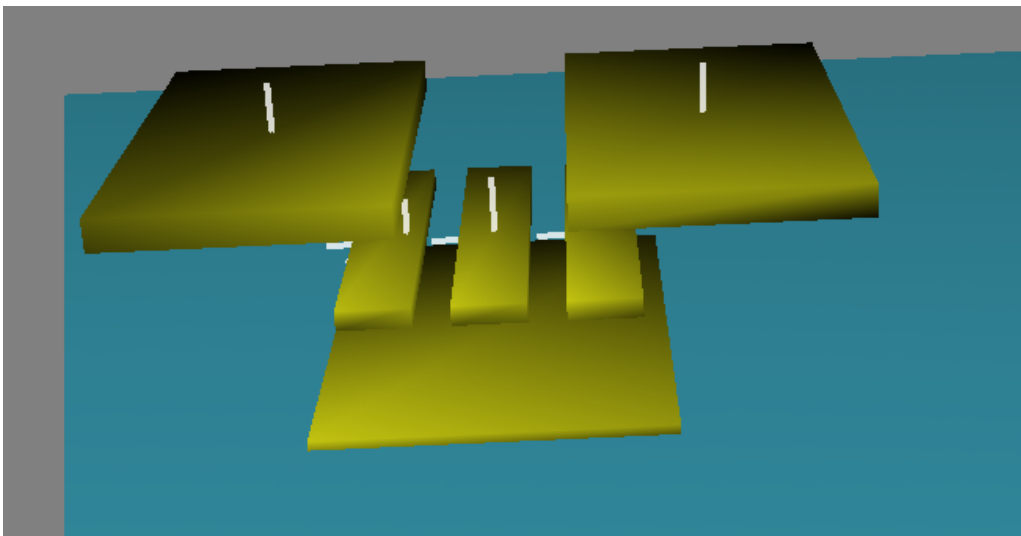


Automatic Placement of all the objects by the solver

In this example, a constraint is not validated. The red box is actually lighter than the green item above. This kind of constraint is not currently available on the solver whereas the C++ hard coded rules allow us to see this problem. An iterative placement process will then allows us to place the objects correctly.

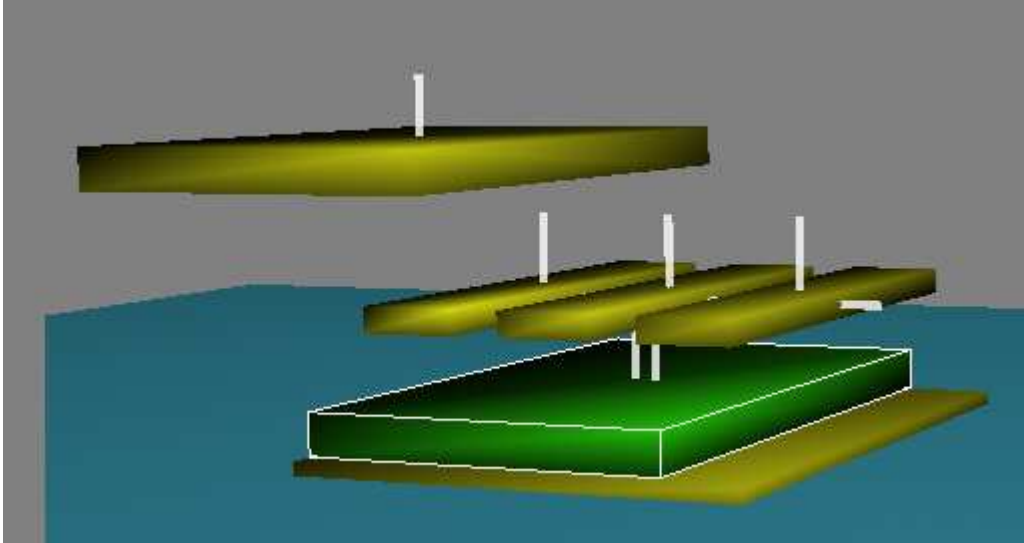
3.3.4.2 Iterative Placement

You want to place each carton in a given order. You can use the solver to place one after another in an iterative way. The first point to do this is to choose a default rule coloration that does not render your lone items invisible (typically, the first warning colour should not be xde/Invisible). Make or load your scene. The first thing to do is to freeze the item level (use reset if your items already fall down too much):



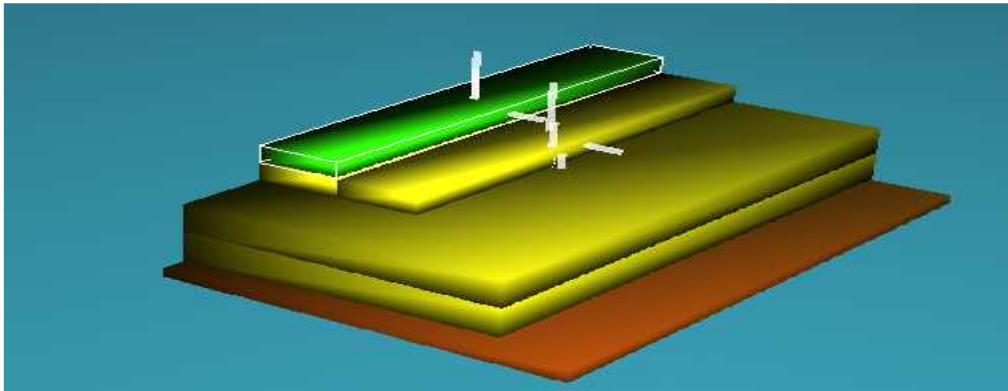
Scene with a bin and 5 items

Your items are frozen to prevent the solver to move them. Then place the first object you want to be placed. Select it, unfreeze it and call the solver.



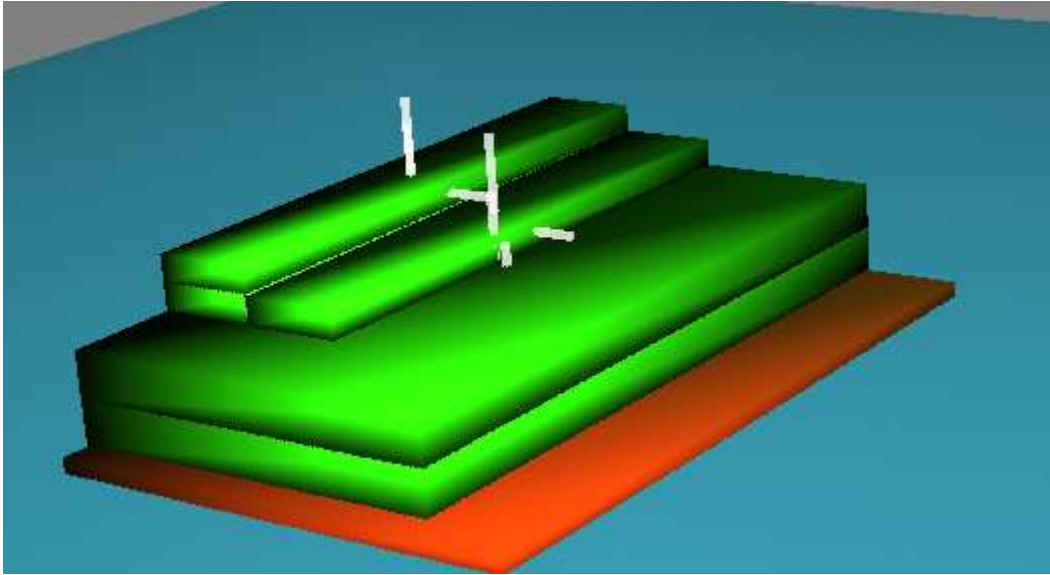
First item placed.

Only the first item was placed. To place the second, you can freeze the first one so that it will not move. Do that until you are satisfied to have placed all your items. Contrary to the automatic placement seen before, this time, you can choose the objects you place. So do not forget to place the heavier first.



All items are placed.

All the items are placed and we can now try to see the stability and the validity of the structure. All we have to do is to unfreeze the level of the items to have the valid configuration free.



All items have been placed, the frozen status are cancelled, all objects are free

Contrary to the automatic placement, we see that there is not conflict about an item lighter being below an item with more weight because we choose to take the heavier items first.

4 Description of the Solver Prototype

The prototype is powered by the solver of Optim Pallet, software developed by KLS OPTIM using Net-WLS technologies to optimise the packing of various products (multi-references) into pallets. Additional features were developed and added to the solver to provide interactive capabilities with the Virtual Warehouse.

The Web-services provided are detailed in the manual "Net-WMS SOA / Interface Designer / Packing solvers". The manual describes the Web-services facilitating the piloting of the packing solvers used by the application developed by CEA-List within the framework of the project Net-WMS. The services are shared in four categories:

- Reading of the data
- Updates by the current scenarios
- Getting the results
- Piloting the solvers.

References:

"Optim Pallet: User manual" version 1.2.5, 2008, KLS OPTIM.

"Optim Pallet: Installation manual" version 1.2.5, 2008, KLS OPTIM.

"Net-WMS SOA / Interface Designer / Packing solvers", version 1.2.3, July 2008, KLS OPTIM.

4.1 D5.2 Specification validation

The D5.2 was defining the specifications of the Solver interface to the Virtual Warehouse. This specifications were evolved and in a certain measure validated using the CLPGUI prototype described in the D5.2 document. However, this software was lacking in physical simulation. It is thus superseded now by the D5.5 prototype described in the current deliverable.

This new prototype that first implements collaboration between the solvers and the physical simulation, allows a better check on these specifications. The followings paragraphs will list point by point how to answer the specifications with the new architecture and interface. In case of failure, we will see how to solve the difficulty.

A part of the D5.2 & D5.3 specification has not been implemented yet, as it is just an early prototype. The full interface will not be available before the end of the corresponding work packages: WP6 for instance will deliver integration of all PKML functionalities in the solvers around M+30. Nevertheless, this interface is sufficient to be able to give the end-users a prototype they can begin to play with and give feedback on.

4.1.1 Communication from the Virtual Warehouse to the Optimizer

4.1.1.1 Uploading a Packing Knowledge Model

This is not possible now, the current configuration of the solver embodies a few rule to activate or to deactivate:

- The Weight Control controls that the solver does not put items on a bin that would mean it is overweighed.
- The Volume Control is the same for the volume: some objects can have a maximum volume as well as a maximum weight.

4.1.1.2 Uploading a Packing Problem Instance

The main purpose of the new interface is to be able to create and check a scene with ease, not having to rely on big text files (potentially long to transmit and that require development for parsers).

The current interface allows the Virtual Warehouse to create a problem instance as surely as in PKML. On details, the only shape available is the box (a parallelepiped shape) with full data for the solver (weight, maximum weight for bins, Is Stackable for items ...).

The Interface also displays the patterns interface, but the interactions with the Virtual Warehouse do not exist yet.

4.1.1.3 Solving Request

The method Solve (int [solve](#)(int idScenario)) satisfies this specification.

4.1.1.4 Request for other solution

This specification is not yet met since the solver always gives the same solution.

4.1.1.5 Request for Getting a Solution Close to Current Configuration

This specification is not yet met since the solver always gives the same solution.

4.1.1.6 Placement Requests for one Object

This specification, even if it is not formally implemented can be satisfied by freezing all the objects except one and asking the solver to solve this problem. This is the method explain in 3.3.4.2 Iterative Placement. That also required the definition of some of the new functionalities (cf. Simulation Manipulator added specifications).

4.1.1.7 Placement Requests for one Region

These specifications are not yet implemented. However, the interface for the holes management is already in place.

4.1.1.8 Information Request on one Object or one Region

Most of these requests will be done directly by asking the web service for information (such as the weight is given by GetWeight). It is also planned to add custom information for objects and bins that will be retrievable later. The PKML expressions, however, are not supported yet due to the fact that the KLS solver does not use WP6 results yet.

For the regions, this has not been implemented yet.

4.1.1.9 Control Requests

Some of these specifications are implemented but the only part done is the management of the objects (clear the solver is calling the init() method, removing, adding items and bins). The backtracks, history of the solver process are not accessible yet. However, a history of the scene is managed directly by the Virtual Warehouse.

4.1.2 Communication from the Optimizer to the Virtual Warehouse

4.1.2.1 Solutions

The need for the solver to send back a solution (meaning a file describing the solution) to the Virtual Warehouse has now disappeared because of the web service. Actually, the web service allows the Virtual Warehouse to directly get the position of the items after the solving part and update the scene according to the solver answer. In other words, the Virtual Warehouse does not wait anymore for the solver to provide its solution but ask the web service for it. The only thing currently not available is the range of object not placed. If an object is not placed, its position is not valid (-1, -1, -1).

4.1.2.2 Failure and Explanations

You can get the numbers of errors, messages and warnings and get them in the order you want and display them. If those messages were correctly implemented in the solvers, the interface would be sufficient to give the user the information he needs.

4.1.2.3 Regions

Nothing is implemented yet on this point.

4.1.3 Incremental Execution

The backtracks and the definition of partial goals are not implemented. However, the use of freeze on objects of the same level and the calls of the solvers allow the user to make incremental research.

4.1.4 Synchronous Communication

These specifications are some of the most important to be able to give back information to the end user. The great gift of the use of web-services is that they render every communication with server asynchronous. There is no more use of synchronous requests as every information is instantaneously available. All what is needed to do is to make a loop to get the information regularly.

Unfortunately, the information needed by the end-user is not available yet. We cannot know yet what rules are broken by which objects, of the forbidden regions when we select an object. We look forward to have them available as soon as possible but this requires nearly everything else to be finished. Therefore, this precious information are not available to the users yet, except for those hard-coded in C++.

5 First Experiments

The prototype was already presented twice at the consortium or part of it. That gives precious feedback and a lot of ameliorations (and bugs) to work on. The redaction of this document and the exchanges it generates also leads to more feedback.

The first presentation was at a working meeting between INRIA, KLS and CEA in the CEA facilities (21/09/2008). Due to the fact that KLS is a specialist of palletisation, some very interesting feedback was collected. Some conceptual problems were found that were not foreseen, such as the borning boxes being static or dynamic.

The second presentation was at the plenary meeting in Paris (08/10/2008). A lot of interesting feedback was gathered, especially on the overall stability.

Here are some of the feedback gathered and their advancement.

5.1 “What if” Analysis

This section presents a few of the basic actions that can be made by the virtual warehouse interface. It is quite light at the moment but will be greatly enhanced with the feedback and questions of the end users.

5.1.1 I want to place a single object on a bin

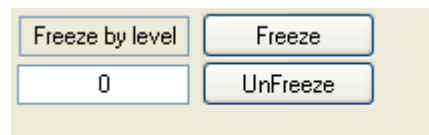
There are already a few items placed on a bin and you want to place one more object on it (only one).

To do so, you will need the Object Placement window.

There are then four possibilities. Some allow you to place exactly the object where you want, other are more automatic.

5.1.1.1 Solver Placement

This method consists in calling the solver to place the single object. The first step is to tell the solver that there is only one object to place. This is done by freezing every item in the scene.



Generally, the item level is 1. Then select the object you want to move and unfreeze it (via the freeze button or the shortcut ctrl+shift+Z). Then call the solver with the solve button. The object will be placed on a bin according to the solver best solution. The development of the interaction with the solver (particularly the “request for another solution”) will allow to test many place if you are not satisfied on the first one.

5.1.1.2 Automatic Placement

This method consists in placing the object next to another with an automatic method. To place this object, you do not need freezing the scene.

X position	Y position	Z position	X offset	Y offset	Z offset
center	center	min	0,00	0,00	0,00
Place on :					
none					
Place					

This part allows the user to place the item relatively to another. The complete description of this functionality is made in D5.5. To summarize, placing the object will ensure the scene is a correct state. If the place is not free, the algorithm will try to find another place for the object, so be careful to the result before validating this point. However, if the bin is quite empty, you can place the object with a perfect precision.

5.1.1.3 3D Manual Placement

This method consists in moving the object with a 3D device like a SpaceMouse. So select the object you want to move (and check that your object is not frozen) and attach it to the device (using CTRL+A or the attach button). Then move your device until you have the place you want. It is recommended to freeze the rest of the scene if you want to prevent other objects from moving.

5.1.1.4 Arrow Manual Placement

This method consists in moving the object slowly toward a direction to place it. First, select your object and check it is not frozen. Then you can move this object either using keys on the keyboard (4, 5, 6, 8, PageUp and PageDown) or the interface:

N	U	Velocity
W	E	0,00
S	D	Stop

The keys have moved the object relatively to the camera and the interface controls absolute moves. It is advised to freeze the other objects to prevent them from moving (due to collisions).

5.1.2 Testing the stability of the structure

Now that you have a fully made bin, you are able to test its stability. Creating a bin configuration is often easier with disabling the gravity on objects. The first step to test the stability is to put back the gravity. To do it, open the Object Property window and set a gravity value not equal to 0.

The second step, if the structure is stable with the gravity is to use the shaking methods. There is many way to use this. We will detail only two. Select the bin and shake it. This simulates the displacement of the bin using the Physics window.

X Offset	Y Offset	Z Offset	Period	Timeout
Shake	Shake At Top	Top		

The second way is to shake every object on the bin. This simulates the bin being on a truck. To do it, enter in the physics window the values you want for the shaking. Then, select one by one

the objects to which you want to apply the shaking and select the synchronize form of shaking (Shake at top button). When you have finished, start the whole (Top button).

5.2 Missing Features

5.2.1 Weld/Unweld a single item to a bin

The user can already freeze all the objects contained in a bin. However, what if he wants to add only one object to this bin when many are in contact with the bin? What about associating and fixing an object to a bin that is not in contact with it (I want to freeze an object on a bin so that the solver can put some others objects between the bin and this object ...)?

This feature was added soon after the plenary meeting and is still in test.

5.2.2 Reset.

How to reset the scene to the configuration when the scene started? The previous way to do this was to save it in history at the beginning of the scene. However, with the fall of object due to the weight, it was very hard to click on save before the scene have changed too much.

Easy to do, just save the scene at the occurrence of the event that signals the configuration file has been successfully read.

5.2.3 Freeze/Unfreeze everything by level

We have often a scene consisting in a bin and several items. To place those at the correct place or if the user wants to make an unstable structure, it is often easier to place them one by one freezing the rest of the scene. The test of stability or fall (you want to see how it will fall), can be spoiled if you have to unfreeze the items one by one.

This feature is implemented and fully working (rather simple to develop or test).

5.2.4 Activate/Deactivate Centre of Gravity

The centres of gravity are useful information to be displayed, but, in case of numerous little items, this information can become confusing. A means to stop the display of centres of gravity had thus to be devised.

This feature is complete. The next step that is considered is to be able to select which objects have there centres of gravity displayed and make the representation size variable from an object to another.

5.2.5 Rule Colour feedback and texture.

As said earlier, some partners have requested to put textures on objects to be able to recognize an item from another. A feature was implemented to do that. The problem is that the texturing of an object considered as a forcing of colour of this object prevents the feedback from the rules that is normally shown as a colour (automatic colour).

This feature deals deeply with the mechanism of Virtools. In addition, it raises a number of questions that will demand a lot of development time:

- When there is a colour forcing, is it a texture forcing only or a whole colour to suspend feedback?
- How to make the difference between a perfect colour and a texture in Virtools?
- How to blend the texture and the colour to make something worth using?

Quite a lot of development is needed to make this feature and it was decided to postpone its development until the finishing of the prototype.

5.2.6 Internationalization

Right now, the prototype is designed in English, but, as we have partners coming from different countries, it would be a good thing to be able to change the language of the interface easily.

The Internationalization is something very probably possible in .Net framework, we have yet to explore how to do it properly and to evaluate the needed development time.

5.2.7 Adding Objects Dynamically

Right now, adding objects is quite complex. Their function depends on when you add them to the scene. It would be better to be able to load objects whenever you want and tell at the adding if they are due to be purely graphical or physical too.

Unfortunately, due to the limitations of the physical engine, it will not be possible to add physical object after the starting of the physical simulation. On the contrary, adding purely graphical objects whenever you want might be possible quite soon. This feature should be ready soon.

5.3 Bugs and malfunctions

Unfortunately, the prototype is not bug free. Here is a list of bugs identified during the demonstrations. As it will appear, the two main bugs of this section are related to the auto-generation of valid scene and the coupling with database.

5.3.1 Interpenetration of objects

It is one of the most disturbing and difficult bugs encountered so far. When generating object (via Object Maker and loading from a database) and then placing them with solver help, it happens that two identical items tend to collapse one inside another. This is due to a failure of the collision detection or the integration of these contacts. It is a very serious bug on the core development of the laboratory.

We identified the bug as a default on the cube geometries that go through the Virtools process. When we load an object in Virtools, its geometry is modified by the Virtools engine to create a better representation for display purpose. These modifications (in instance edge splitting) is very bad for the collision detection module because it means a lot of extra-calculus on degenerate

cases (probabilistically, an edge/edge collision never happens, but with the help of solvers, it is often true because of the exact placement of objects).

The measure to overcome this bug is to save the geometry before sending the object in Virtools to be able to have a correct geometry in the physical engine. This bug is under debugging and expected to be fully debugged before the sending of this report.

5.3.2 Texture Application

The users have expressed to be able to put textures on objects. It seems the application to automatically generated object does not work. That bug is linked to the previous one. It is simply due to the fact that the geometries we make in Virtools have no texture coordinates. The resolution of this bug and the previous one imply a better parser from the generated file to Virtools geometries. It is under debugging.

5.3.3 Automatic Place outside of the bin

During the demos, we encounter some special case where the automatic placement of objects (Place button in the main window) ends with the item being out of the bounding box of the bin. This bug was solved by adding the proper test at the place where it could occur. This bug is now solved.

6 Conclusion

This document has described the prototype developed to validate the Net-WMS concept of "interactive optimisation" that allows a user to finely control the solving of a packing problem through a 3D virtual reality interface. The proposed software derived from specifications contained in previous deliverables D5.2 and D5.3. The work done has led to validating a large part of these specifications and to amending another smaller part. Although it does not implement all the D5.2 and D5.3 functions (mainly because some of the required components are not available in their definitive shape since the project is only entering its third year), the prototype is sufficiently complete to start its evaluation with end-users. Actually, the Virtual Warehouse framework is now implemented and linked to databases and solvers. As feedback from the end-users will reach the development team, further functionalities will be added inside this framework.

In parallel, work will proceed on the physical simulation functions addressing the behaviour the packages during shipping and the validation of packing / unpacking processes. This is the main goal of the next deliverable (D5.4 "Implementation of the interactive simulation functions") planned on M28. However, it is worth to notice that some of these simulation functions are already available on the current prototype (see section 3.2.11).