Using Constraint Solver for 3D layout Assistance in Human-Scale Virtual Environment

Anonymous

Keywords: Virtual reality, human-scale interaction, constraint programming, 3D-spatial configuration, human performance

Abstract: This paper proposes a combination of virtual reality (VR) and constraint programming (CP) to develop an efficient system that will assist users with performing 3D layout tasks in a virtual environment. An experimental study has been carried out to investigate to which extent the solver's assistance affects user performance in 3D-layout tasks. Volunteer participants were instructed to create a generic 3D scene which was displayed on a large-scale rear-projected screen. Results showed that the solver provided a significant amount of assistance in both the simple and the complex tasks. The participants performed the layout tasks more accurately and in significantly less time when the constraint solver was used. Most of the subjects reported that the solver was useful in completing the layout tasks and they were generally satisfied by the proposed solution. However, it was observed that some subjects had difficulty in interacting with the solver, especially during the complex tasks.

1 INTRODUCTION

Solving three-dimensional (3D) placement or layout problems consists of placing components (i.e., a set of 3D objects) inside of a virtual container while satisfying a set of given constraints. These constraints allow researchers to specify how components, such as furniture and equipment, may be combined together to make a single layout. In such problems, the components and the container are generally functionally and geometrically linked. Depending on the designer's experience level, the development of semi-automatic methods for the resolution of spatial problems is often an interesting challenge as systems become more and more complex. This challenge relies on the difficulty of modeling and formulating these problems, as well as the difficulty of identifying resolution strategies.

Constraints are naturally present in several areas such as resources allocation, planning and industrial productions. A constraint expresses a property or condition that should be satisfied and is generally defined as a relationship between variables (positions of objects in our case). The spatial problems can be modeled using an efficient framework such as the Constraint Satisfaction Problem (CSP) formalism. Solving a CSP consists of assigning values to the variables while satisfying all the constraints (Fruhwirth and Abdennadher, 2003). Algorithms used to solve a CSP are called constraints solvers.

Virtual Reality (VR) is a human-centered technology based on software and hardware elements, simulating intuitive human interaction with virtual objects which could be synthetic, imaginary or symbolic. 3D interaction is the main component of VR, allowing the user to control the virtual environment (VE) and to interact with virtual entities (Bowman, 1999; Heim et al., 1995).

In recent years, VEs have become increasingly popular due to advances in graphics technology and user interfaces (Messinger et al., 2009). VEs are recognized as powerful design tools in most industrial sectors such as manufacturing, process engineering, and aerospace (Zorriassatine et al., 2003). One of the possible uses of VEs can be to monitor decision making process such as the 3D-object layout problems (or spatial problems) involving complex tasks

restricted by several constraints. However, in many cases, VEs are only being used as pure visualization tools for assessing final designs (Drieux et al., 2005) and do not generally provide assistance to the user and more specifically in 3D-layout tasks where an accurate placement of objects is required (Essabbah et al., 2014). Thusly, the integration of intelligent modules such as constraints solvers could make VEs more efficient to solve spatial configuration problems (Sanchez et al., 2002; Calderon et al., 2003; Kefi et al., 2011). In this context, specific interaction techniques and assistance involving both 3D objects and constraint solvers need to be developed and evaluated. In other words, our objective is to *facilitate the interaction between the user and the solver to efficiently lay out a 3D scene.*

This paper is organized as follows. In Section 2, we provide an overview of relevant research works and discusses our system. The interaction model and communication process between the VE and the solver is then explained in Section 3. An experimental study with description of research methodologies and measurements is described in Section 4. The results are described in Section 5 with discussion in Section 6. Section 7 concludes the study with recommendations for future work.

2 RELATED WORK

Previous works have shown the relevance of CP techniques in spatial configuration problems. As demonstrated by Honda and Mizoguchi (Honda and Mizoguchi, 1995) and Pfefferkorn (Pfefferkorn, 1975), CP techniques are particularly appropriate for the resolution and the description of spatial configurations and its efficiency at avoiding combinatorial explosion. Among the previous works that combined constraint programming (CP) or constraint logic programming (CLP) and VEs, a notable example is Codognet's work that included a concurrent constraint programming system into VRML to specify the behavior of artificial actors in dynamic environments (Codognet, 1999). Axling et al. (Axling et al., 1996), incorporated "OZ" (Smolka et al., 1993), a high-level programming language, supporting constraints in the DIVE-distributed VR environment (Andersson et al., 1993). These works have essentially been dedicated to the study of behavior of individual objects or autonomous agents within VE. However, both Axling and Codognet research groups have not addressed user interaction or assistance for problem solving.

Fernando et al. provided the design and implementation details of a constraint-based VE (Fernando et al., 1999), presenting a software framework to support constraint-based assembly and maintenance operations. Xu et al. studied the combination of physics, semantics, and placement constraints to investigate how such combination permits a user to quickly and easily layout a scene (Xu et al., 2002). The 3D-layout was substantially accelerated with a simple pseudo-physics engine and a small amount of semantic information. They generalized this approach and developed a richer set of semantic information leading to a new modeling technique. Sanchez et al. presented a general-purpose constraint-based system for non-isothetic 3D-object layouts, built on a genetic algorithm (Sanchez et al., 2002). This system is able to process a complex set of constraints, including geometric and pseudo-physics designs. To create an easy-to-use object-layout software, Sanchez and his colleagues described the 3D-scene by using semantic and functional features associated with the objects in regards to the layout. Smelik et al. used semantic constraint, a control mechanism imposed on the procedural generation of VEs, in order to satisfy explicit designers intent over a specific area. It is composed of sub-constraints feature mapped to low-level operations (Smelik et al., 2011).

Several approaches approximate the relationships between a large number of components with simplified "dimensional constraints" aiming at formulating the design problem as a system of (in)equalities. In this context, Theodosiou et al. proposed informational-complete models for design-constraints based on the analysis of geometric and non-geometric properties of the related space volumes (Theodosiou and Sapidis, 2004). An extended product model was proposed describing the system's structure and components as well as related procedures and constraints to be used as a system life-cycle model.

Sutherland proposed a man-machine graphical communication system called "Sketchpad". The geometric constraints-based system makes it possible for a user and a computer to converse rapidly through the medium of line drawings (Sutherland, 1964). On the other hand, Marriott et al. proposed a generic algorithm for linear arithmetic constraints that makes use of the Cassowary constraint solving algorithm (Marriott et al., 2001). Marriott and her colleagues described an algorithm for rapidly resolving disjunctions of constraints. The algorithm is

designed to support direct manipulation in interactive graphical applications which contain non-overlap constraints between graphical objects. They also demonstrated that the solver can support non-overlap of complex non-convex polygons, and complex diagrams such as State Charts that contain non-overlap as well as containment constraints. Calderon et al. presented a novel framework for the use of VEs in interactive problem solving (Calderon et al., 2003). This framework extends visualization to serve as a natural interface for the exploration of configurations space, and enables the implementation of reactive VEs. Their implementation was based on a fully-interactive solution where both visualization and the generation of a new solution are under the user control. To visualize and control logic programs, Fages et al. developed a generic graphic user-interface (CLPGUI). The proposed architecture involves a CLP process and a Graphical User Interface (GUI) which communicate through sockets (Fages et al., 2004). This approach has been evaluated using a simple layout problem involving a basic layout scene. Jacquenot developed a hybrid generic method to solve multi-objective placement problems for free form components (Jacquenot, 2009). The proposed method is a hybrid algorithm based on both a genetic and separation algorithms. Tim et al. introduced a novel rule-based layout solving approach to speed-up manual design methods and create parts of a game world automatically (Tim et al., 2009). They showed that their solving approach may be used for procedural generation by providing the solver with a user defined plan. In this plan, users can specify objects to be placed as instances of classes, which in turn contain rules about how instances should be placed. They validated their approach in different procedural generation scenarios. Medjdoub presented a system for ceiling-mounted fan coil system in a building ceiling. The system is simple to use with interactive modification of the 3D parametric model (Medjdoub, 2004). He showed that this approach significantly reduced design costs, improved the quality of the solution, and produced additional benefits in the supply chain. More recently, Merrell et al. identified a set of interior design guidelines for furniture layout and developed an interactive system that suggests possible furniture arrangements based on these guidelines. The system incorporates the layout guidelines as terms in a density function and generates layout suggestions by rapidly sampling the density function using a hardware accelerated Monte Carlo sampler (Merrell et al., 2011). The user is able to interact with the system to iteratively evolve the design of the interior.

Although interesting, most of the studies described above have some limitations and can be extended in several directions. One way to extend the research is to offer a deeper integration of the user to increase the interaction with the solver. For instance, during a 3D layout, the user should be able to: (1) add and manipulate objects from a menu at any time in the environment and (2) manually place some objects in the environment, the placement of the other objects being automatic. It should be noted that most of these previous works are based on CLP or genetic algorithms. However, in the last few years, powerful CP-based solvers such as Gecode (Schulte et al., 2012), CHOCO (Jussien et al., 2009) and, ILOG CP (IBM, 2012) have been developed. Unlike CLP and Prolog, these newer solvers are designed as a library. In addition, these solvers provide an API to developers to embed constraint programming in programs written in an object language (C++ or Java). Moreover, these solvers are not dependent on logic programming and make it easy to use constraints in an independent and efficient solving-engine.

3 SYSTEM DESCRIPTION

Before describing the architecture and the interaction model of the proposed system, we first present the solver process used to find a solution of a modeled problem using a CSP formalism.

3.1 3D layout problem formulation

To model a 3D layout problem, a CSP formalism was chosen, which offers a simple way to model and solve such problems. A set of variables (i.e., unknowns of the problem) and constraints have been identified. Each of these variables takes its values from a given domain. As such, the resolution of a CSP consists of assigning a value to each variable from its domain in order to satisfy all the constraints. The basic resolution process involves two stages: (1) constraint propagation and (2) enumerations (Fruhwirth and Abdennadher, 2003).

1. *Constraint propagation*: Constraint propagation consists of reducing variables domains by removing values to eliminate portions of the search space that cannot be part of a solution.

$$
\begin{aligned}
&bool\ Propagate(C, Vc, D)\{\\
&\quad for\ all\ var\ in\ Vc\{\\
&\quad\quad for\ all\ value\ in\ D(var)\{
\end{aligned}
$$

$find\ a\ solution\ to\ C\ with\ var = value$
$if\ no\ solution\ exists,\ remove\ value\ from\ D(var)$
$if\ D(var)\ is\ empty\ return\ false$
}
}
$return\ true$ }

2. *Enumeration steps*: Enumeration involves a succession of decisions (to approach a solution) consisting of choosing a value for each variable from the remaining domain.

A 3D layout problem can be modeled through the following parameters:

- *Container*: the space to be laid out
- *Objects*: objects to be placed in the container such as furniture or materials.
- *Layout requirements*: constraints or restrictions linking objects between them and the container.

To formulate the problem, it was supposed that the VE of dimensions (w,h,d), is composed of n objects related by m constraints. Let X be the set of the unknowns of the problem (3D positions of objects), $\mathcal{D}$ be a function that associates a domain (authorized values) to each variable, and C be the set of layout constraints. Thus, the problem can be defined by the triplet (X,$\mathcal{D}$,C):

- $X = \{x_1, y_1, z_1, ..., x_n, y_n, z_n\}$, $(x_i, y_i, z_i / i \in [1,n])$ is the position center of $object_i$
- $\mathcal{D}(x_i) = [w_i, w - w_i]$, $w_i / i \in [1,n]$ is the width of $object_i$

  $\mathcal{D}(y_i) = [h_i, h - h_i]$, h_i/i$\in$ $[1,n]$ is the height of $object_i$

  $\mathcal{D}(z_i) = [d_i, d - d_i]$, d_i/i$\in$ $[1,n]$ is the depth of $object_i$
- $C = \{c_1^{i,j}, c_2^{i,j}, ..., c_m^{i,j}\}$, $c_k^{i,j} / (k \in [1,m]$ and $i, j \in [1,n])$ is a constraint between $object_i$ and $object_j$.

3.2 Proposed system

The proposed system is based on a human-scale, real-time VE which supports the resolution of interactive 3D-object layouts using ongoing communication with an efficient constraint-solver (Gecode) writing in C++ language. The selection of objects and constraints as well as a user's 3D manipulation is converted into queries sent to the solver. The solver's outputs (positions of objects) allow real-time automatic reconfiguration of the 3D scene.

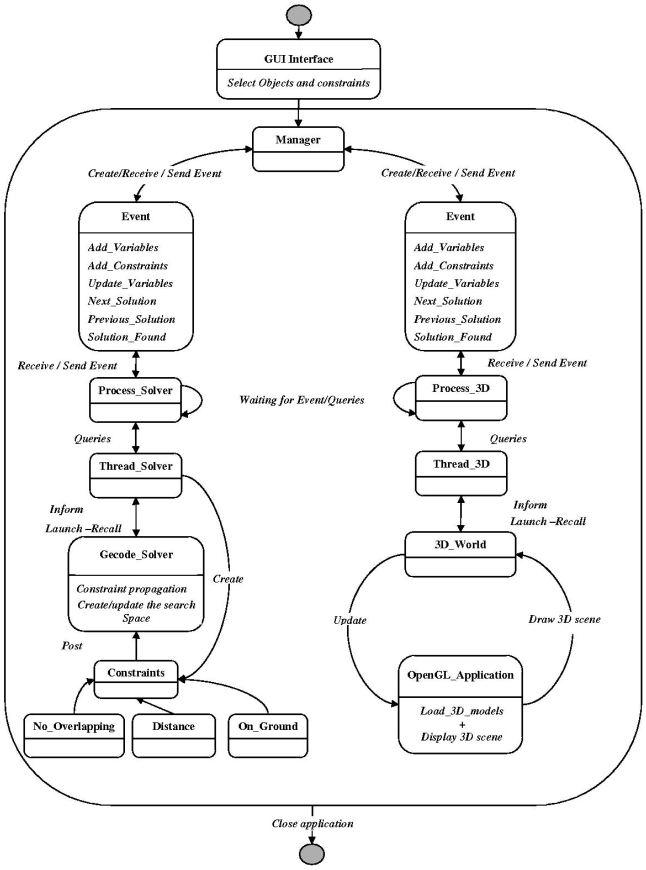


Figure 1: *System architecture (event exchange between VE and Solver).*

Figure 1 illustrates the general resolution process and the link between the 3D environment and the solver. The events exchange is managed by a *Manager* class which creates and sends the corresponding event to the visualization module (*Process_3D*) and the resolution module (*Process_Solver*). Each module sends the necessary requests to the corresponding thread. To ensure real-time interaction, each module runs on a different thread. The request sent to the *Thread 3D* is to display the selected objects. The *Thread_Solver* class creates a CSP in which the variables are the positions of the center of selected objects, and the constraints are those initially specified. The Gecode solver is called to find a solution to the proposed problem. The *Thread_Solver* class will be informed about the results and will transmit the corresponding request to the upper level (*Process_Solver*). Thus the *Process_Solver* class informs the *Manager* class that a solution has been found (through the event "*Solution_found*"). The resulting 3D positions of each object are then forwarded level by level until the VE reconfigures itself by updating the 3D scene.

3.3 Computation time

A decision-making system is intended to facilitate choices and decisions in a given context. In the 3D layout context, a well-designed interactive decision-making system should provide real-time assistance to ensure effective interaction. Therefore, the solver computation time is an important criterion to evaluate the performance of the proposed system. The solver computation time depends on three parameters: (1) number of objects, (2) constraints number and complexity and (3) the heuristics used for search (Fruhwirth and Abdennadher, 2003).

Figure 2 shows an example of the required solver computing time to layout different number of objects. For this example, a no-overlapping constraint was used between each pair of objects which are placed in a 3D space with size $(e.g., 20, 20, 20)$. It should be noted that the computed time was obtained in the worst case because only one constraint was considered. Indeed, the fact to consider is that many constraints reduces the search space and therefore the computation time. These results were obtained on a machine equipped with an $Intel^{TM}$ $i7-2630$ processor running at $2.00Ghz$.

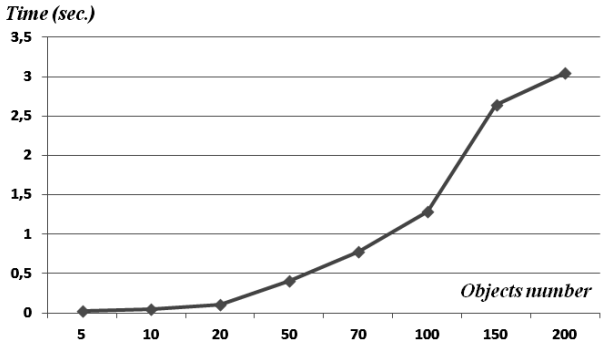


Figure 2: *The evolution of computing time vs. objects number.*

3.4 Interaction framework

In the proposed system, the possible user manipulations inside the VE can be summarized in the following actions or groups of actions: (1) add/remove/displace objects, (2) add/remove constraints, and (3) call the solver to search a solution for a defined layout problem. A Nintendo Wiimote^{TM} was used as an input device for selecting/deselecting and moving objects in the 3D space. To track the user's hand position, the Wiimote^{TM} was equipped with two reflective markers. An OptiTrack^{TM} camera system (OptiTrack, 2011) was used to track the position of these markers. A Nunchuk^{TM} was used to interact with the solver and to select constraints from a menu. This interaction technique was chosen because Wiimote devices offer multiple interaction methods, such as buttons, vibro-tactile feedback, that can be easily integrated in any VR systems at a relatively low price.

To add objects in the VE, the user selects and introduces a given object (e.g., cube, sphere or cone) using the 3D menu. To do this, he/she must press and hold the B button on the Wiimote^{TM} and gesture to move a given object to a desired position. The selected object is released when the user releases the B button. In the same way, the user can apply constraints on the selected objects. After selecting one or several objects, the Nunchuk^{TM} is used to view (using the joystick) and select (Z button) a desired constraint from a constraint menu and trigger the solver. To facilitate constraint identification, a description (i.e., information sheets) of each one is displayed on the top of the screen as 2D text.

It is also important to make note of how the solver responds to the user's requests and how the solver's results are interpreted in the VE. It seems obvious that the main goal of the interaction model is to make the link between the user's interactions and the inputs/outputs of the solver. In other words, the interaction model was developed to facilitate user interaction since modeling the problem and finding solutions are completely transparent for the user since he/she interacts with the VE regardless of the resolution mechanisms. The user's manipulation, such as the selection of objects and constraints as well as objects transformations (rotation and translation), are converted to queries sent to the solver. Based on these queries, the solver updates the existing CSP or creates a new one if new objects are added to the scene. Next, the solver instructs the updated or the newly-created CSP to find a new solution. Consider the example in which the user defines a layout problem by adding some objects in the VE and selecting a set of constraints. According to the user's choices, the system will automatically create a CSP that represents the problem. The formulation and the resolution of this problem is left to the solver and this process is completely transparent to the user (Fig. 3).

Algorithms for solving constraint-based problems are generally triggered by changing the values of variables and/or constraints which define the problem. In our case, user interaction will be translated

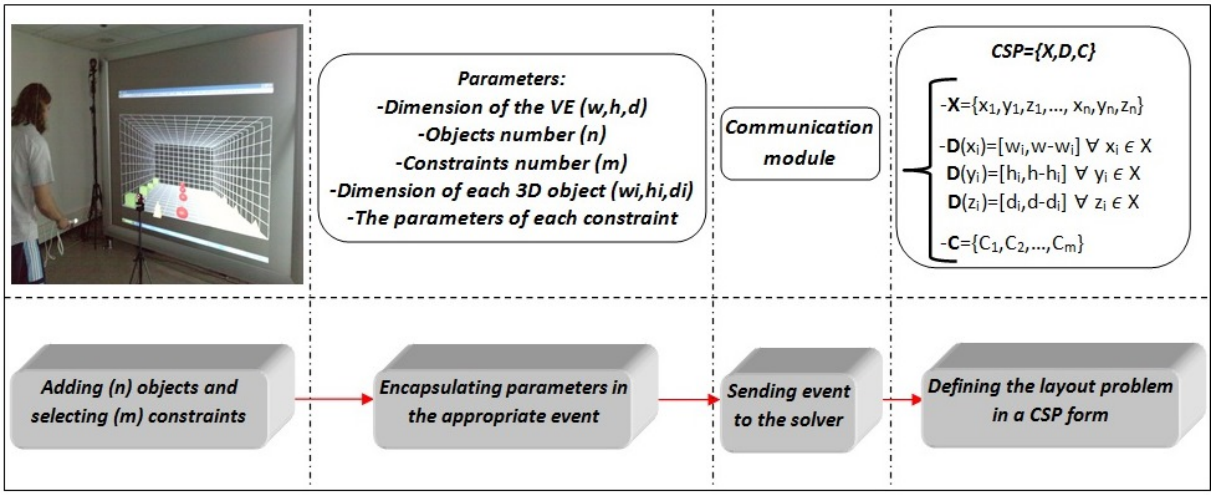


Figure 3: *Architecture of the system.*

into inputs to the solver which will deal with the variables whose values were changed by the interaction. For instance, if the user displaces a given object, the associated constraints are modified. This displacement triggers the solver which propagates the constraints and computes new results.

Let us consider a simple example where the user moves the gray object (circled object) to the right (Fig.4). An event will be automatically generated and a new constraint, which sets the position of the gray object, is defined in the already created CSP. These constraints will be applied on the object whose index is encapsulated in the event sent to the solver. Therefore the solver is re-called to detect a possible constraints dissatisfaction and to compute the objects' positions which respect all the constraints. The computed data are encapsulated in another event sent to the VE via the communication module, in order to reconfigure the 3D scene.

4 EXPERIMENTAL STUDY

Previous studies did not provide detailed analysis of the human performance, and particularly, nor the comparison between manual (without solver) and assisted (with solver) 3D-layout tasks. There are multiple issues to be considered when integrating a constraint solver in a VE: data exchange between the constraint solver and VE, user interaction, and effective and sufficient user assistance. An experimental study was conducted to evaluate the effectiveness of a combination of a solver and a VE, with special focus on the interaction technique. Quantitative and qualitative data were collected, including the effects on task completion time and placement accuracy, as well as subjective preference from users. The study compared two approaches, namely manual and assisted, with two different levels of layout difficulty, simple and complex. The following hypotheses were proposed:

1. *H1:* the participants are satisfied with the solver assistance.
2. *H2:* the interaction with the solver during the layout task is easy.
3. *H3:* the participants solve the layout task faster with the solver.
4. *H4:* the participants make less placement errors with the solver.

A NASA-TLX questionnaire was used to access user's satisfaction (*H1*) and the ease of interaction with the solver (*H2*). Task completion data was collected for *H3*. The satisfaction of the hypothesis *H4* is relatively obvious since a solver does not make errors. However, it is important to confirm it in the interest of the solver integration.

4.1 Experimental Design

Twenty two subjects were recruited from a local university with ages ranging from 22 to 47. The participants were asked about their previous experience with video games and VR and to rate their level of expertise. For prior video games experience, fifteen out of the twenty two participants answered yes to this question. For VR expertise, seven subjects

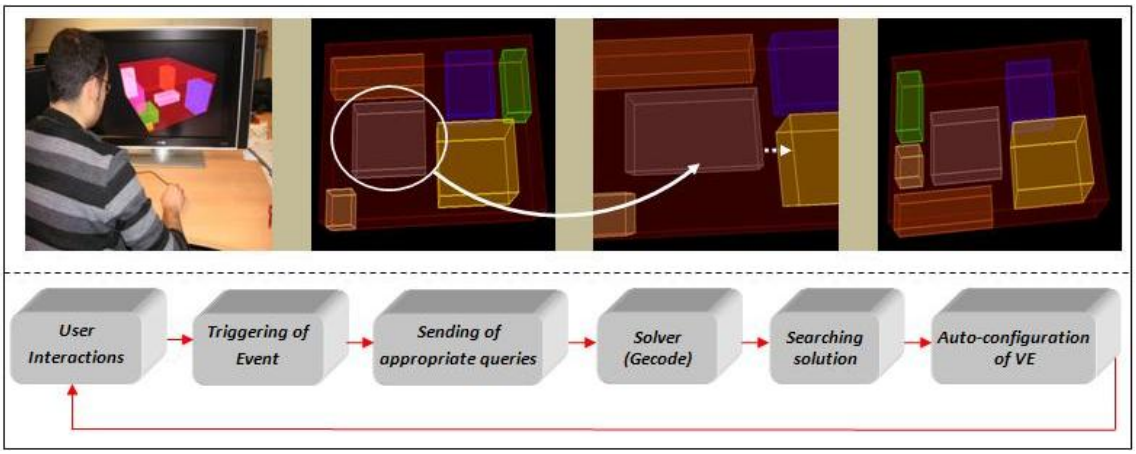


Figure 4: *Illustration of the rearrangement of the 3D scene according to an object displacing.*

claimed they had a moderate amount of experience with VR tasks while nine subjects rated their experience as low and eight as high.

A 2 x 2 balanced, within-subjects factorial design, was used where the independent variables were the user assistance (UA) and the task complexity level (TCL). The first factor (UA) has two levels: assisted (with solver) and manual (without solver). Through a rear-projected human-scale VE (Richard et al., 2006), participants were instructed to layout twelve 3D objects (four cubes, four spheres and four cones) in the space represented by a large 3D room while respecting the displayed constraints, which were diplayed as text on screen. In order to be able to generalize the results to different application contexts a generic task and simple 3D environment was chosen.The task involves some constraints such as object on object and object against wall.

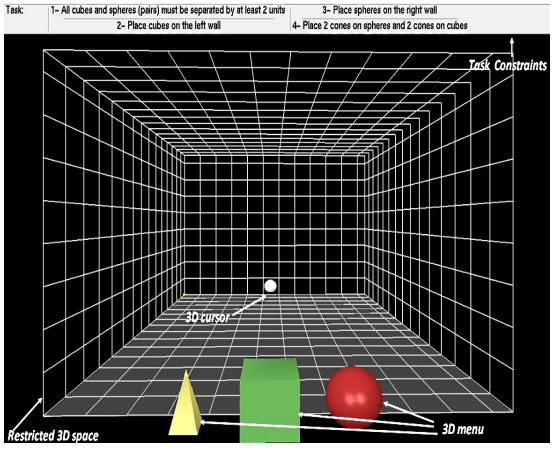


Figure 5: *Snapshot of the virtual environment at the beginning of the task.*

At the beginning of the task, the 3D space was empty. The user had to select a given object (cube, sphere or cone) using a 3D menu (Fig. 5). During the two layout tasks, a reminder (information sheets) of the constraints was displayed on the top of the screen as 2D text.

The *manual task* consists of laying out the objects in any order, while abiding by the proposed constraints. The participants were instructed to select the objects using a 3D cursor. Selection was performed by pressing and holding the B button of the Wiimote^{TM} and moving the selected object to a desired position. The object is released when the participant releases the B button on the controller (Fig. 6).

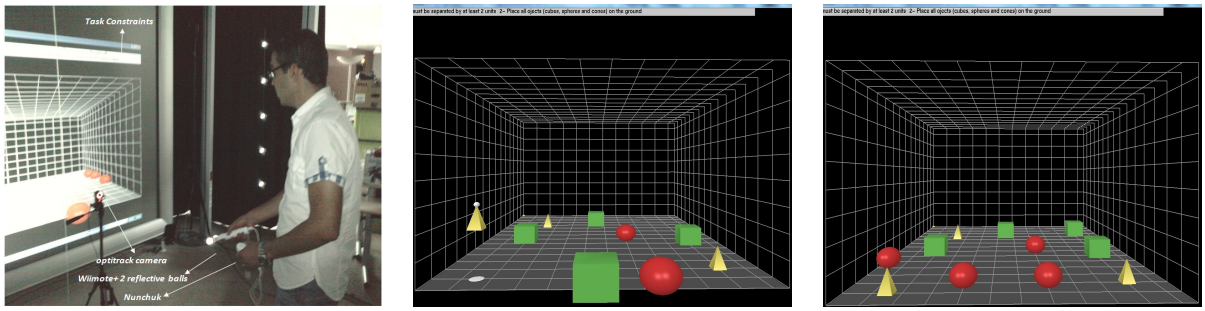
4.1.1 Manual simple task (*MS*)

The participant is required to manually lay out the 3D scene while abiding by the following constraints:

- *On_ground constraint*: The participant must place all the objects (i.e., 4 cubes, 4 spheres and 4 cones) on the ground. There is no requirement about the order of placement. The participant may start with any object.
- *Minimum_distance constraint*: The participant must place the objects with a distance ***d_min*** (2 unit squares) between them.

4.1.2 Manual complex task (*MC*)

The difficulty level of the simple task was increased by adding three constraints (the On ground constraint was removed). The participant needs to satisfy specific constraints related to different objects. There-



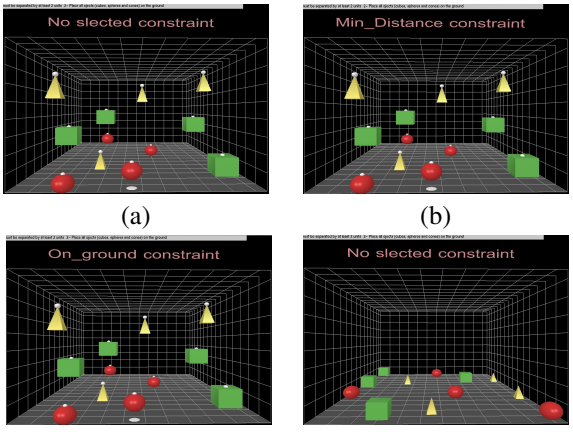
(a) (b) (c)

Figure 6: *A subject uses a Wiimote^{TM} (a) and shadows (b) to layout the 3D scene (c).*

fore, more effort was required from the user to complete each task.

- *Left_wall constraint*: participants must place all cubes against the left wall;
- *Right_wall constraint*: participants must place all spheres against the right wall;
- *Object_on_object constraint*: participants must sequentially place two cones on two separate cubes. Similarly, the remaining two cones must be placed on the two spheres. It is important to note that this constraint requires a selection order: thus, to put *object 1* on *object 2*, the participant needs to first select *object 2* then *object 1*.
- *Minimum_distance constraint*: participants must place the objects (except cones) with a distance ***d_min*** (2 units) between them.

The assisted task consists of using the solver to apply constraints on selected objects in order to automatically lay out the scene. The participants may select a given object by pressing the A button on the Wiimote^{TM}. This allows the user to view (using the joystick) and select (Z button) the desired constraint from a menu and trigger the solver (Fig. 7).



(a) (b) (c) (d)

Figure 7: *A subject selects the Minimum_distance (b) and the On_ground (c) constraints using the Nunchuk^{TM}, and finally call the solver for automatic layout (d).*

4.1.3 Assisted simple task (*AS*)

In this condition, the same constraints as in the MS condition were used, but the solver was used to assist the users. Since the constraints should be applied on all the objects, the participant may adopt different strategies:

- Strategy 1:
  (1) Select one object (cube or sphere or cone), (2) select the *On_ground* constraint, (3) select two objects, (4) select the *Minimum_distance* constraint and (5) validate the constraint and trigger the solver for resolution. The same steps should be repeated for all the remaining objects.
- Strategy 2:
  (1) Select all objects (cubes, spheres and cones), (2) Select the *On_ground* constraint and (3) Validate the constraint and trigger the solver. The same steps should be repeated for the *Minimum_distance* constraint.
- Strategy 3:
  (1) Select all objects (cubes, spheres and cones), (2) select *On_ground* and *Minimum_distance* constraints, (3) validate constraints and launch the solver.

4.1.4 Assisted complex task (*AC*)

In this condition, the same constraints as in MC condition were used, but the participants had to perform the layout task using the solver. Similarly to AS, the participant may adopt different strategies:

- Strategy 1:
  (1) Select a cube, (2) Select the *Left_wall constraint* constraint, (3) Repeat these steps for the other 3 cubes. (4) Select a sphere, (5) select the *Right_wall constraint* constraint, (6) repeat these steps for the other 3 spheres. (7) Select two cubes, (8) select the *Minimum_distance* constraint, (9) repeat these steps for the other 2 cubes. (10) Select two spheres, (11) select the *Minimum_distance* constraint, (12) repeat these steps for the other 2 spheres. (13) Select a cube and a cone, (14) select the *Object on object* constraint, (15) repeat these steps for two other cube and cone. (16) Select a sphere and a cone, (17) select the *Object_on_object constraint* constraint, (18) repeat these steps for two other sphere and cone. (19) Validate constraints and launch the solver for resolution.

- Strategy 2:
  (1) Select all cubes, (2) select the *Left_wall constraint* constraint, (3) Select all spheres, (4) select the *Right_wall constraint* constraint, (5) select all cubes, (6) select the *Minimum_distance* constraint, (7) select all spheres, (8) select the *Minimum_distance* constraint, (9) select a cube and a cone, (10) select the *Object_on_object constraint* constraint, (11) repeat these steps for two other cubes and cones. (12) Select a sphere and a cone, (13) select the *Object_on_object constraint* constraint, (14) repeat these steps for two other sphere and cone. (15) Validate constraints and launch the solver for resolution.

- Strategy 3:
  (1) Select all cubes and spheres, (2) select the *Minimum distance* constraint, (3) select all cubes, (4) select the *Left_wall constraint* constraint, (5) select all spheres, (6) select the *Right_wall constraint* constraint, (7) select a cube and a cone, (8) select the *Object_on_object constraint* constraint, (9) repeat these steps for two other cubes and cones, (10) select a sphere and a cone, (11) select the *Object_on_object constraint* constraint, (12) repeat these steps for two other sphere and cone. (13) Validate constraints and launch the solver for resolution.
  The interaction strategy to perform the tasks has not been rigorously defined in order to analyze and study subjects' preferences. The best strategies were observed and it was determined that strategy 3 in simple and complex assisted tasks was chosen by more than 85% of subjects. This information is very interesting and will help us improve user interaction.

4.2 Procedure

The overall goal of the study was explained to the participants, especially about the possibility of controlling the solver during the layout task. Afterwards, a brief overview of the four tasks was given. Before starting the study, participants performed a training session with each task in a randomized order without recording any data. Each subject was given a set of information sheets that described the constraints for each condition. After each experiment, the participants completed the NASA-TLX questionnaire. To avoid the learning effect, the tasks were counterbalanced. In conditions *AS* and *AC* two more questions were added to assess the participants' satisfaction and the subjective preference about the solver's assistance.

4.2.1 Collected data

To investigate the effect of the solver integration on task performance, several measures were utilized that can be categorized as subjective preference (via questionnaire) and task performance in terms of completion time and layout errors.

- **Satisfaction** Using a seven-points Likert scale (i.e., 1 for "very low" to 7 for "very high"), subjects were asked to express their satisfaction about the layout process and the solution proposed by the solver. It should be noted that the satisfaction concerns the final proposed solution and not the user interaction.

- **Assistance** Using the same seven-points Likert scale, subjects were asked to express their opinions about the assistance provided by the solver. The aim was to know to what extent the solver's integration assisted subjects and facilitated the layout tasks.

  It should be noted that these two questions were not asked in manual tasks (*MS* and *MC*) because the participants lay out the scene themselves without any assistance. In addition, in manual tasks the subjects placed objects one by one according to their preferences, unlike the assisted tasks (*AS* and *AC*) where the solver's decision (computed solution) does not take their preferences into consideration. For this reason, the question about the satisfaction is only of interest when the solver is used (i.e., assisted tasks).

- **Workload** Subjects filled out the NASA-TLX questionnaire to assess their level of workload using six indices:

- *Mental Demands (MD)*: The mental and perceptual activity required, such as thinking, calculating, deciding, remembering, and looking;
- *Physical Demands (PD)*: The physical activity required, such as pulling, pushing, turning, and controlling;
- *Temporal Demands (TD)*: The time pressure perceived due to the rate or pace at which tasks or task elements occurred;
- *Own Performance (OP)*: The estimation of successfully accomplishing the goals;
- *Effort (EF)*: The effort required to achieve the task;
- *Frustration (FR)*: The level of discouragement, irritation and annoyance versus gratification, contentment, and complacency felt during the task.

- **Task Performance** Although the main objective of the study was to assess to what extent the subjects appreciated the assistance provided by the solver, objective data (completion time and number of errors) was also collected. Completion time is defined as the time between the first pressing on the B button (Wiimote^{TM}) and the moment when the participant considers the task achieved. Layout error can be evaluated by two parameters:

  1. ***Placement error***: This type of error is not related to any of the constraints previously described. The system checks whether or not each object was included in the 3D scene (large 3D cube) based on the x, y, and z coordinates. The number of occurrences of this error was calculated for each experimental condition.
  2. ***Distance error***: Since the *Minimum_Distance* constraint exists in all conditions, the error generated by the dissatisfaction of this constraint was calculated. At the end of each task, the system calculates the distance between each pair of objects. If the calculated distance is greater than the threshold (*dmin*), the system records the occurrence of the error.

  In addition to these objective data, we also measured the number of objects selection made by the participants during tasks. Indeed, the study of this data reflects the subject's effort during the tasks. The number of selection of each 3D object was calculated for each experimental condition.

5 RESULTS AND DISCUSSION

Since the subjects have different levels of expertise in video games and VR, a ***t-test*** was carried out to check whether this difference of experience affects task performance. The subjects were separated into two groups:

- G1, concerns subjects who have experience with video games/VR (12 subjects).
- G2, includes subjects who have never played video games (10 subjects).

In order to select the appropriate ***t-test*** (for equal or different variances), an ***F-test*** was performed for the dependent variable (completion time) with the task complexity level (TCL) and user assistance (UA) as the independent variables.Table 1 summarizes the computed ***p-values*** of the ***F-test*** and the ***t-test***.

	MS	AS	MC	AC
F-test	0.245	0.384	0.466	0.185
t-test	0.332	0.357	0.343	0.230

Table 1: *The P-values of the F-test and t-test (for all experiment conditions)*

According to the obtained p-values for the ***t-test***, there is no statistical difference between the tested groups (G1 and G2) and therefore task performance was not affected by the expertise level of the subjects.

5.1 Objective results

As stated before, a two-way analysis of variance (ANOVA) was performed for the dependent variable with TCL and UA as the independent variables. Table 2 summarizes the main effects of the independent variables as well as their interaction for the completion time. TCL and UA significantly affected the completion time and there were no significant interaction effects between them.

Task completion time The results for the TCL were as expected. Unlike the simple task, in the complex task, subjects had to perform many steps to satisfy the four constraints. Therefore, subjects took longer to complete the task. Similarly, the UA significantly affected task completion time. With the first level of UA (without solver), subjects had to manually pick and place objects with a required level of accuracy, which resulted in longer task time.

Effect	*TCL*	*UA*	*TCL* $\times$ *UA*
Time	$p < 0.05$	$p < 0.05$	$p = 0.4198$
	$F_{1,21} = 18.60$	$F_{1,21} = 50.98$	$F_{1,21} = 0.66$

Table 2: *The main and interaction effects of the independent variables on the completion time.*

To assess *H3*, we calculated and compared the mean task completion time for each task. Subjects completed the layout task significantly faster under condition *AS* (79.26 sec) than under condition *MS* (118.06 sec). Similar results were obtained for *MC* and *AC*: the completion time under condition *AC* was significantly shorter (139.46 sec) than for *MC* (165.99 sec).
The results show that *H3* was supported even for complex tasks where the participants need to apply more constraints on selected objects. The results obtained for the hypothesis *H3* can be explained by the fact that subjects only needed to select objects and constraints and trigger the solver. Additionally, subjects didn't need to refine the proposed solution since it satisfied all the constraints. The validation of the *H3* shows that the use of the solver makes it faster and easier for 3D layout tasks compared to the manual method. It should be noted that the task completion time includes the time required by the solver to find a solution and the time for interaction. For a layout problem involving few objects, such as the proposed scene, the solver provides a real-time response and has a little effect on task completion time.
The calculation time of the solver highly depends on the propagation techniques used and on the heuristics of these search. In this experiment standard propagation techniques were used (**?**). These techniques will have to be improved for future experiments with more objects and new kinds of constraints.

Placement and distance errors When subjects were assisted (*AS* and *AC*), the spatial configuration proposed by the solver did not contain errors (no placement and minimum_distance errors). Therefore the system will flawlessly reconfigure the VE. For this reason, it is not justifiable to compare manual and assisted tasks using these two types of errors. However, it is interesting to examine these errors in manual tasks (*MS* and *MC*).

Results showed that about 27.27% of manipulated objects were placed partially outside the VE in *MS* and about 33.91% in *MC*. The participants often failed to respect the minimum distance constraint. In fact, 21.7% of the objects were placed too close to each other ($<$ ***d_min***) in *MS* and 16.2% in *MC*. This relatively high number of errors was highly dependent on the visual condition (without stereoscopy) despite the use of shadows, but it could be reduced by incorporating stereo viewing and depth cues (Polys et al., 2011). The use of the solver completely avoided this type of error which validates hypothesis *H4* concerning the effect of the solver on object placement accuracy. These results can be explained by the fact that the solver satisfies all constraints and correctly places the objects .

Number of selection As previously mentioned, the number of object selection reflects the subject's effort during the tasks. Subjects often selected the same objects many times in the manual tasks, which required far more effort than in the assisted tasks. Figure 8 shows that the average number of selection on each object (*obj0 to obj11*) depends on the type of task (manual or assisted). The subjects made more object selections in the manual tasks and consequently took more time. Thus, the use of the solver indirectly reduced the overall difficulty by reducing the number of object selections.

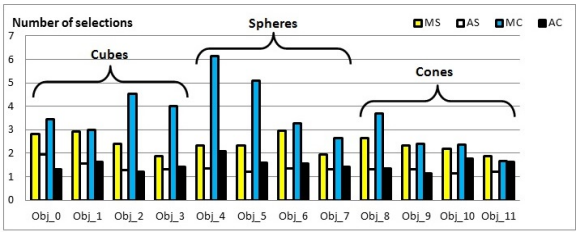


Figure 8: *Average number of selections for each object in each condition.*

5.2 Subjective results

Task workload Subjects filled out a NASA-TLX questionnaire to express their perceived level of workload. They were asked to rank each workload index using a seven-point Likert scale. An ANOVA analysis was performed to find the effect of the independent variables (user assistance and task complexity level) on the workload indexes (*MD, PD, TD, OP, EF, FR*).

As shown in Table 3, there were no significant interaction effects between the two independent variables and no significant effect of TCL on temporal demand, effort and frustration, whereas it significantly affects the mental, physical demands and their own performance. Indeed, the complex

Effect	*MD*	*PD*	*TD*	*OP*	*EF*	*FR*
TCL	$p = 0.0001$	$p = 0.0071$	$p = 0.703$	$p = 0.0166$	$p = 0.2663$	$p = 0.1025$
	$F_{1,21} = 17.64$	$F_{1,21} = 7.58$	$F_{1,21} = 0.15$	$F_{1,21} = 5.95$	$F_{1,21} = 1.25$	$F_{1,21} = 2.72$
UA	$p = 0.0286$	$p < 0.0001$	$p = 0.0174$	$p < 0.0001$	$p = 0.0018$	$p < 0.0001$
	$F_{1,21} = 4.95$	$F_{1,21} = 20.50$	$F_{1,21} = 5.87$	$F_{1,21} = 26.26$	$F_{1,21} = 10.34$	$F_{1,21} = 18.90$
TCL* × *UA	$p = 0.9349$	$p = 0.4272$	$p = 0.6127$	$p = 0.6281$	$p = 0.5788$	$p = 1.00$
	$F_{1,21} = 0.01$	$F_{1,21} = 0.64$	$F_{1,21} = 0.26$	$F_{1,21} = 0.24$	$F_{1,21} = 0.31$	$F_{1,21} = 0.001$

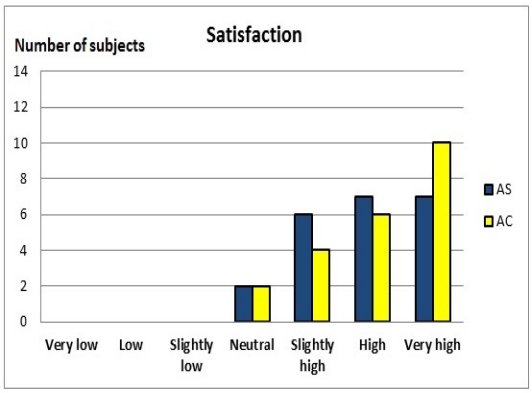
Table 3: *The main and interaction effects of the independent variables on the workload indexes (Mental demand (MD), Physical demand (PD), Temporal Demands (TD), Own Performance (OP), Effort (EF), Frustration (FR).*

tasks required a higher mental and physical demands ($Mean_{\{MD\}} = 3.29$ *and* $Mean_{\{PD\}} = 2.83$) than the simple tasks ($Mean_{\{MD\}} = 2.22$ *and* $Mean_{\{PD\}} = 2.18$). For their own performance, a lower score is observed in the complex tasks ($Mean_{\{OP\}} = 5.85$) than in the simple ones ($Mean_{\{OP\}} = 6.27$).

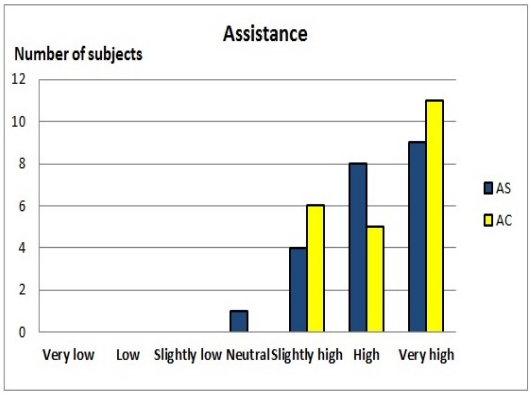
For the second independent variable (UA), the results showed a significant effect on all the workload indexes. The Table 3 illustrates how participants found the non-assisted tasks causing a higher mental, physical and temporal demands ($Mean_{\{MD\}} = 3.04$, $Mean_{\{PD\}} = 3.04$ *and* $Mean_{\{TD\}} = 4.93$), a higher level of effort and frustration ($Mean_{\{EF\}} = 3.58$ *and* $Mean_{\{FR\}} = 2.97$), and a lower feeling of performance ($Mean_{\{OP\}} = 5.62$) than in the assisted tasks ($Mean_{\{MD\}} = 2.47$, $Mean_{\{PD\}} = 1.97$, $Mean_{\{TD\}} = 4.14$, $Mean_{\{EF\}} = 2.62$, $Mean_{\{FR\}} = 1.77$ and $Mean_{\{OP\}} = 6.5$).

Satisfaction and solver assistance Subjects were asked to answer two additional questions concerning satisfaction and solver's assistance. For the satisfaction level, they assess solutions proposed by the solver (Fig. 9.a). In condition *AS*, 20 subjects reported that they were satisfied with the proposed solution. Two subjects gave a neutral response and said that the proposed spatial configuration is not optimized although all of the constraints were satisfied. In condition *AC*, 16 subjects were either satisfied or very satisfied with the proposed solution. Only two subjects reported that the solver had not given the best solution. It is important to note that the solver can find many solutions but only proposes the first one, which is not necessarily the best according to a subject's preferences.

Subjects were also asked to evaluate the assistance level provided by the solver over the layout tasks (Fig. 9.b). In condition *AS*, 21 subjects reported that the layout became easier to set up when using the solver; only one subject did not feel any difference between manual tasks and the assisted ones. In the *AC* condition, all subjects reported that the solver made the task much easier.
According to the results of workload, satisfaction and assistance, the hypothesis *H1* was supported. In addition to the benefits of utilizing the solver on task performance, subjects strongly appreciated the provided assistance.



(a)



(b)

Figure 9: *Satisfaction (a) and subjective feelings of assistance level provided by the solver (b) for conditions AS and AC*

Interaction technique evaluation Most of the participants reported that the interaction with the solver through the proposed technique was easy and intuitive. However, some subjects had difficulties identifying and applying the constraints. Some subjects reported that the selection of objects upon which constraints were to be applied was slightly tedious. This is due to the fact that some subjects have chosen one of the least effective strategies to interact with the system.According to these results, the hypothesis *H2* was moderately supported. The low satisfaction of this hypothesis shows that although the solver assistance satisfied the subjects and made the 3D layout easier, the interaction protocol is not yet optimized and requires some improvements. This difficulty is inherent to the proposed interaction technique. This lack of accuracy also affected task performance because subjects had to push the B button several times to select a given object. Similarly, the use of the Nunchuk™ to select and apply constraints was considered slightly difficult. Subjects needed to navigate in the constraint menu before finding the desired one. This problem could be reduced by more intuitive interaction techniques based on multi-modality (for example introducing voice commands to select objects and constraints).

6 CONCLUSION AND FUTURE WORK

The main goal of the present study was to propose a combination of virtual reality and constraint programming to simulate layout tasks and assist users. Both subjective and objective effects of the solver integration were investigated by comparing manual versus assisted tasks for two levels of difficulty. Four conditions were proposed: *MS* (manual simple task), *AS* (assisted simple task), *MC* (manual complex task) and *AC* (assisted complex task). The results indicated that the use of the solver reduced both the number of object selections and errors, regardless of the level of the task's difficulty. The results also revealed that participants peroformed the layout task faster when the solver was used. Most of the subjects reported that the solver was useful and assisted them with completing layout tasks and they were generally satisfied by the proposed solution. Both objective and subjective data also showed that the proposed interaction technique has some shortcomings. In order to improve interaction with the solver, we propose that future designs use a multimodal approach based on gestures and voice commands which facilitate the selection and validation of constraints (solver control) and/or objects. We are currently developing a specific application jointly with industrial partners. This application involves real objects and specific constraints.

REFERENCES

Andersson, M., Carlson, C., Hagsand, O., and Stahl, O. (1993). Dive - the distributed interactive virtual environment - tutorials and installation guide. In *Swedish Institute of Computer Science*.

Axling, T., Haridi, S., and Fahlen, L. (1996). Virtual reality programming in oz. In *Proceedings of the 3rd EUROGRAPHICS Workshop on Virtual Environments*.

Bowman, D. (1999). *Interaction Techniques for Common Tasks in Immersive Virtual Environments : Design, Evaluation, and Application*. PhD thesis, Georgia Institute of Technology.

Calderon, C., Cavazza, M., and Diaz, D. (2003). A new approach to the interactive resolution of configuration problems in virtual environments. *Lecture notes in computer science*, 2733:112 – 122.

Codognet, P. (1999). Animating autonomous agents in shared virtual worlds. In *Proceedings DMS'99, IEEE International Conference on Distributed Multimedia Systems*. IEEE Pres.

Drieux, G., Guillaume, F., Leon, J., and Chevassus, N. (2005). Samira: A platform of virtual maintenance simulation with haptic feedback incorporating a model preparation process. In *Proceedings of Virtual Concepts*.

Essabbah, M., Bouyer, G., and Otmane, S. (2014). A framework to design 3d interaction assistance in constraints-based virtual environments. *Virtual Reality*, 18(3):219–234.

Fages, F., Soliman, S., and Coolen, R. (2004). Clpgui: A generic graphical user interface for constraint logic programming. *Constraints*, 9:241 – 262.

Fernando, T., Murray, N., Tan, K., and Wimalaratne, P. (1999). Software architecture for a constraint-based virtual environment. *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 147–154.

Fruhwirth, T. and Abdennadher, S. (2003). Essentials of constraint programming. *Springer Verlag*.

Heim, M., Featherstone, M., and Burrows, R. (1995). The design of virtual reality. *Cyberspace/ Cyberbodies/Cyberpunk Cultures of Technological Embodiment*, pages 65–78.

Honda, K. and Mizoguchi, F. (1995). Constraint-based approach for automatic spatial layout planning. In *Conference on Artificial Intelligence for Applications*. IEEE Press.

IBM (2012). Ilog products and solutions, http://ftp.ilog.fr/products/cp/.

Jacquenot, G. (2009). Mthode gnrique pour l'optimisation d'agencement gometrique et fonctionnel. *Thse de Doctorat, Ecole Centrale de Nantes*.

Jussien, N., homme, C. P., Cambazard, H., Rochart, G., and Laburthe, F. (2009). *choco: an Open Source Java Constraint Programming Library*.

Kefi, M., Richard, P., and Barichard, V. (2011). Use virtual reality and constraint programming techniques in interactive 3d objects layout. In *International Conference on Computer Graphics Theory and Applications*.

Marriott, K., Moulder, P., Stuckey, P. J., and Borning, A. (2001). Solving disjunctive constraints for interactive graphical applications. In *CP*, pages 361–376.

Medjdoub, B. (2004). Constraint-based adaptation for complew space configuration in building services. *Journal of Information Technology in Construction*, 243(2007):627–636.

Merrell, P., Schkufza, E., Li, Z., Agrawala, M., and Koltun, V. (2011). Interactive furniture layout using interior design guidelines. *ACM Trans. Graph.*, 30(4):87–95.

Messinger, P. R., Stroulia, E., Lyons, K., Bone, M., Niu, R. H., Smirnov, K., and Perelgut, S. (2009). Virtual worlds past, present, and future: New directions in social computing. *Decision Support Systems*, 47(3):204 – 228.

OptiTrack (2011). Optical motion capture systems and tracking software, http://www.naturalpoint.com/optitrack/.

Pfefferkorn, C. (1975). A heuristic problem solving design system for equipment or furniture layouts. *Communications of the ACM*, 18(5):286–297.

Polys, N. F., Bowman, D. A., and North, C. (2011). The role of depth and gestalt cues in information-rich virtual environments. *Int. J. Hum.-Comput. Stud.*, 69(1-2):30–51.

Richard, P., Chamaret, D., Inglese, F., Lucidarme, P., and Ferrier, J. (2006). Human-scale virtual environment for product design: Effect of sensory substitution. *IJVR*.

Sanchez, S., Roux, O. L., Inglese, F., Luga, H., and Gaildart, V. (2002). Constraint-based 3d-object layout using a genetic algorithm.

Schulte, C., Tack, G., and Lagerkvist, M. (2012). *Modeling with Gecode*.

Smelik, R., Galka, K., Kraker, K. J. D., Kuijper, F., and Bidarra, R. (2011). Semantic constraints for procedural generation of virtual worlds. volume 1, page 9.

Smolka, G., Henz, M., and Wurtz, J. (1993). Object-oriented concurrent constraint programming in oz. research report. In *Deutsches Forschungszentrum fur Kunstliche Intelligenz*.

Sutherland, I. E. (1964). Sketch pad a man-machine graphical communication system. In *Proceedings of the SHARE design automation workshop*, pages 329–346, New York, NY, USA. ACM.

Theodosiou, G. and Sapidis, N. S. (2004). Information models of layout constraints for product life-cycle management: a solid-modelling approach. *Computer-Aided Design*, 36(6):549–564.

Tim, T., Rafael, B., Ruben, M. S., and Klaas, J. K. (2009). Rule-based layout solving and its application to procedural interior generation. In *Proceedings of the CASA workshop on 3D advanced media in gaming and simulation (3AMIGAS)*, pages 212–227.

Xu, K., Stewart, J., and Fiume, E. (2002). Constraint-based automatic placement for scene composition. In *Graphics Interface Proceedings, University of Calgary*.

Zorriassatine, F., Wykses, C., Parkin, R., and Gindy, N. (2003). A survey of virtual prototyping techniques for mechanical product development. *Journal of Engineering Manufacture*, page 217.