3D INTERACTIVE OBJECTS LAYOUT USING VIRTUAL REALITY TECHNIQUES AND CONSTRAINT PROGRAMMING

Marouene Kefi, Paul Richard
Laboratoire d'Ingénierie des Systèmes Automatisés (LISA), Université d'Angers, 62 Avenue ND du Lac, Angers, France

Vincent Barichard
Laboratoire d'Etudes et de Recherche en Informatique d'Angers (LERIA)
Université d'Angers, 2 Bd Lavoisier, Angers, France

Keywords: Virtual environment, 3D Interaction, Decision-making, 3D Objects layout, Constraint programming.

Abstract: Virtual Reality (VR) is a promising tool for effectively visualizing and interacting with objects and 3D environments. However, Virtual Environments (VEs) should provide some assistance to help the users in complex solving tasks. We aim to combine VR and Constraint Programming (CP) techniques in order to assist the users in realizing 3D objects layout in restricted spaces. The proposed approaches are based on a tight communication between a 3D rendering module and a highly efficient constraint solver. Layout modification are translated in incoming queries to the solver which generates the solutions that satisfy predefined constraints. In order to achieve users' immersion in the VE and intuitive manipulation of the objects, a human-scale VE platform with haptic feedback is used. In this paper, we focus on the system architecture and we describe the implementation of simple constraints. Finally, some results based on geometric constraints are presented.

1 INTRODUCTION

A spatial problem may be defined as a placement problem for which a positioning of components inside a container is seeked. The development of methods for automatic solving of such problems is challenging while the systems become more complex. This is mainly explained by the difficulty related to the modeling and the formalization of these problems.

Virtual Reality (VR) aims to immerse users in synthetic worlds in which they will experience multimodal interactions with virtual objects (Bowman, 1999). VR is therefore a promising tool for effectively visualizing and interacting with 3D environments (Drieux et al., 2005). However, in order to be really effective, virtual environments (VEs) should provide some assistance to help the users in complex solving tasks.

We apropose to combine VR and Constraint Programming (CP) techniques in order to assist the users in 3D objects layout in restricted spaces. The context of this work is the design and the 3D layout of military vehicles or shelters. The proposed approaches involve a tight communication between a 3D rendering module and a highly efficient constraint solver. Layout modification are translated in incoming queries to the solver which generates the solutions that satisfy predefined constraints. A constraint expresses a property or a condition that must be satisfied. It can be defined as a relation-ship between one or more variables. The notion of constraint is naturally present in several areas such as resources allocation, planning and industrial production.

In the next section, we survey the previous work. Then, we present our system including a human-scale VR platform. We focus on the interaction model and the communication process between the 3D environment and the solver. In section four, we describe interactive approaches based on CP techniques. In section five, we present the results associated with some basic geometric constraints. Then we examine the time required by the solver to compute the existing solutions of an under-constrained problem (two constraints only). The paper ends by a conclusion that provides some tracks for future work.

Kefi M., Richard P. and Barichard V..
3D INTERACTIVE OBJECTS LAYOUT USING VIRTUAL REALITY TECHNIQUES AND CONSTRAINT PROGRAMMING.
DOI: 10.5220/0003377303080313
In *Proceedings of the International Conference on Computer Graphics Theory and Applications* (GRAPP-2011), pages 308-313
ISBN: 978-989-8425-45-4

2 RELATED WORK

Some works have proved the relevance of CP techniques for the solving of configuration problems (Honda and Mizoguchi, 1995) and Pfefferkorn (Pfefferkorn, 1975). Honda and Mizoguchi, and Pfefferkorn have demonstrated the suitability of CP as an approach which facilitates the description of constrained problems and its efficiency for avoiding combinatorial explosion. CP have been also successfully used for defining and solving physical constraints (Glencross and Murta, 1998).

Among interesting approaches based CP or Constraint Logic Programming (CLP) in 3D environments, we can cite Codognet's work in which he included a concurrent CP system into the VRML language (Codognet, 1999). Axling et al., (Axling et al., 1996) have incorporated OZ (Smolka et al., 1993), a high-level programming language supporting constraints into a distributed VE (Andersson et al., 1993). These works have been essentially dedicated to the behavior of individual objects or autonomous agents within the environment and did not address user interaction or interactive problem solving.

More recent approaches involving under-constrained problems in VEs have been developed. For example, Fernando et al., have presented the design and the implementation details of a constraint-based VE (Fernando et al., 1999). Xu et al., have investigated the combination of physics, semantics, and placement constraints and how it permits to quickly and easily layout a scene (Xu et al., 2002). Xu also generalized a richer set of semantic information leading to a new modeling technique where users can create scenes by specifying the number and distribution of classes of object to be included in the scene. Calderon et al., have presented a novel framework for interactive problem solving applied to VEs (Calderon et al., 2003). The proposed implementation was based on a fully interactive solution where both visualization and the generation of a new solution are under the control of the user. Sanchez et al. have presented a general-purpose constraint-based system for 3D object layout built on a genetic algorithm (Sanchez et al., 2003). They have described the 3D-scene by using semantic and functional features associated with the objects. The system was able to process a complex set of constraints, including geometric and pseudo-physics ones. Fages et al., have developed a generic graphical user interface (CLPGUI) for visualizing and controlling logic programs (Fages et al., 2004). The proposed architecture involves a CLP process.

More recently, Jacquenot has developed a hybrid generic method to solve multi-objective placement problems of free-form components (Jacquenot, 2009). The proposed method involves a hybrid approach based on both an evolutionary algorithm and a separation algorithm. A main drawback of these approaches is that all the possible solutions are not provided. In addition the user has no explanation about the non-feasibility of a given solution.

In the last few years, powerful CP-based solvers such as Gecode (Schulte, 1997), CHOCO (Jussien et al., 2009), or ILOG CP (IBM, 2010) have been developed. These CP-based solvers provide an API and could therefore be embedded in any C/C++ or java applications. However, none of them has been used in the context of interactive layout of 3D environments.

3 SYSTEM OVERVIEW

The proposed system supports interactive 3D objects layout through a thight communication between the constraint solver and the 3D layout. Objects manipulation are transcribed to queries sent to the constraints solver. Then automatic reconfiguration of the 3D layout will be achieved. In addition, this system allows the user to explore the different solution (Kefi et al., 2010).

In order to allow users' immersion in the virtual world and assist him/her in the 3D layout task, a human-scale VR platform with haptic feedback is used (Fig.1). This platform provides haptic cues using a string-based force feedback interface (Richard et al., 2006). Stereoscopic images are displayed on a rear-projected large screen and are viewed using polarized glasses.

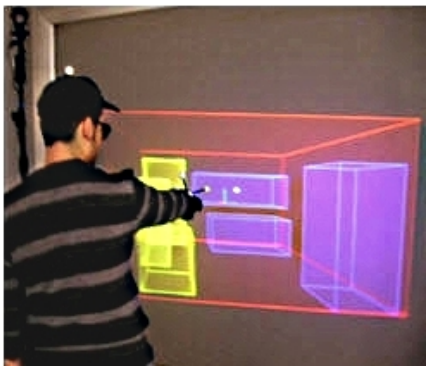

Figure 1: User manipulating the 3D objects using the human-scale VR platform.

Generally, mechanisms for solving under-constrained problems are triggered by a modification of the variables values and/or constraints.

In our case, 3D layout modifications will be translated into inputs to the solver which will act on the variables whose values were changed by the user's interaction. For example, for a given spatial configuration, the user can change the position of certain objects which modifies the underlaying constraints. This triggers the solver which, by propagation of constraints, provides new possible configurations of the 3D layout.

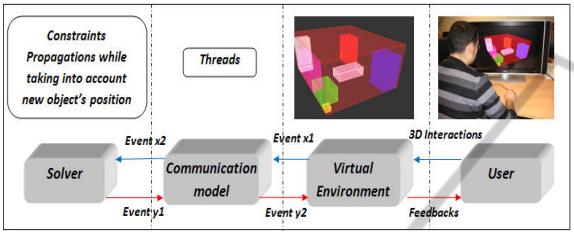

Figure 2: Illustration of the system architecture.

The general architecture of the system is illustrated in figure 2. Exploration of the solutions begins by a first solution from which the user can explore other possible configurations. The user is thus able to interact with the selected configuration by moving any objects.

3.1 Architecture of Interaction Model

The aim of the interaction model is to link user's interaction with the VE and inputs / outputs of the solver. The work of the solver is based on a specific logic depending on which it is triggered by the addition of new constraints and produces results in the form of new positions of objects. Thus, two aspects are concerned: (1) how the solver responds to user's actions, and (2) how the solutions proposed by the solver will automatically modify the VE.

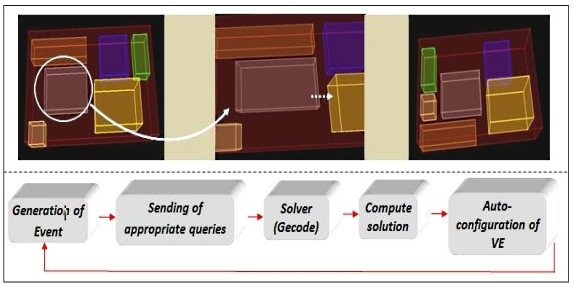

Figure 3: Interaction mechanism.

The modification of the environment will generate an event that will be sent to the solver through the communication module. According to these queries, the solver will propose new solutions that will update the current configuration of the VE. Let us consider the simple example where the user moves the gray object (circled) to the right from an initial solution computed by the solver (Fig. 3). An event will be automatically generated and the communication module will post new constraints to the solver. These constraints will be applied on the object whose index is encapsulated in the event sent to the solver. The solver will thus be re-called and the new position of the concerned objects, and possibly those of other objects, will be encapsulated in another event sent to the VE via the communication module. The new positions of objects will be extracted and the VE will be updated.

4 INTERACTIVE APPROACHES

As mentioned before, our objective is to propose and implement interactive approaches to allow interactive 3D layout. Most approaches are based on the architecture and interaction mechanisms described above. The application starts by allowing the user to select 3D objects and constraints to be satisfied. Then the system launches a dialogue with the constraint solver to check the feasibility of 3D arrangements. Then, the user will have three possibilities for interaction with the objects. These are described in following paragraphs.

4.1 First Approach

This approach is the most straightforward. Taking into account the predefined constraints and the objects selection, all feasible configurations will be computed by the solver and displayed. Then, the user will be able to visualize successively the solutions. The only possible interactions are the control of the viewpoint. If the case of too many solutions, the system only displays the first ten solutions. In order to reduce the number of proposed solutions, closer ones will be automatically eliminated.

4.2 Second Approach

The second approach is illustrated in Figure 4. In this case, the user interacts with the solutions by moving any objects in space. After each displacement, the solver is re-called to compute the new solutions. The last displacement could also be canceled if a constraint is not respected. Once the new solutions are computed, the 3D environment is automatically reconfigured. In order to assist the user during object manipulation, the system provides multi-modal (visual, auditory and kinesthetic) feedbacks.

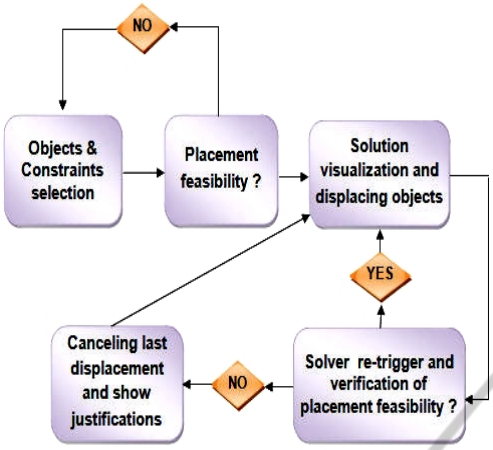

Figure 4: Illustration of the second approach.

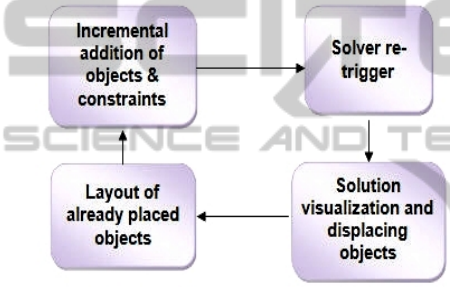

Figure 5: Illustration of the third approach.

4.3 Third Approach

This approach is the more interesting one because it allows to take into account the user's preferences from the very beginning. Thus, the system enables the user to add objects successively in the environment. Each addition of object calls the solver that applies the new constraints added on the object and propagates these to calculate new layout solutions (Fig. 5). Thus, the system automatically reconfigures itself as a consequence of the addition of a new object. It is important to note that the solver is not called to define a new constraint satisfaction problem but only to propagate new constraints.

5 IMPLEMENTATION OF CONSTRAINTS

In our context, constraints can be divided into two categories: geometric constraints and sematic constraints. The geometric constraints are related to the physical placement of 3D objects. For example, the *no_overlapping* constraint, illustrated in Figure 6, prevents the objects to overlap. The semantic constraints are related to variables such as temperature, vibration or electromagnetic radiation. This type of constraints are based of physical equation and are more difficult to implement.

This section presents the results associated with some basic geometric constraints that are illustrated by simple examples. In these examples, the same propagation techniques have been used. Similarly, we used the same heuristics to select the variables and their associated values (Solnon, 2003).

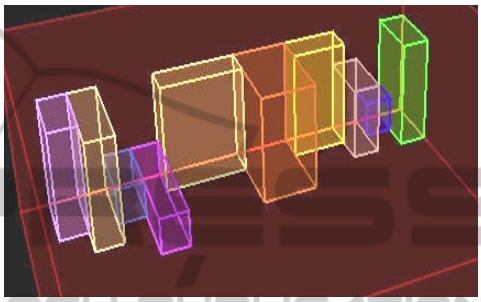

Figure 6: Illustration of the *no_overlapping* constraint.

5.1 No_overlapping Constraint

Whatever the context of the layout problem, the straightforward constraint to satisfy is the *No_overlapping* constraint. Indeed, this constraint avoids the intersection between the different objects. Let us consider the example of ten objects (represented by their bounding box) on which we apply the *No_overlapping* constraint. The figure shows the first solution computed by the solver.

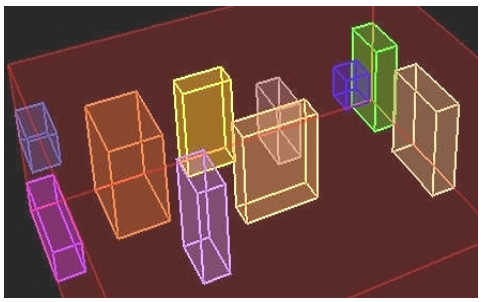

Figure 7: Illustration of the *minimum_distance* constraint.

5.2 Minimum_distance Constraint

This constraint forces the 3D objects to be separated by a distance greater than or equal to a minimum distance (*dmin*) specified by the user. This constraint can also be used for positioning heat sources away

from other objects. An illustration of this constraint is given in the Figure 7.

5.3 Object_on_object Constraint

In many situations, the application requires that some objects are placed on others. We have considered this situation by implementing a *Object_on_object* constraint. In the example illustrated in Figure 8, this constraint is applied on the green and yellow objects, and then on the yellow and the gray objects.

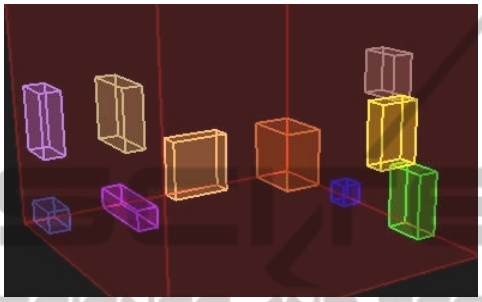

Figure 8: Illustration of the *object_on_object* constraint.

5.4 On_floor Constraint

In most applications, some objects have to be placed on the floor to reduce the center of gravity of the system. Therefore, we implement the *on_floor* constraint. In the example illustrated in Figure 9, this constraint was applied to ten objects.

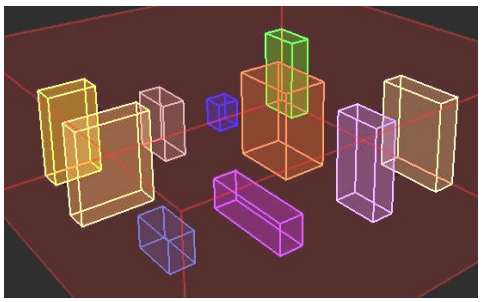

Figure 9: Illustration of the *on_floor* constraint.

6 RESPONSE TIME

In order to be effective the system should have a response time as small as possible. We measured the time required by the solver (Gecode) to find all the solutions, for different number of objects. Two geometric constraints have been used :the *minimum_constraint* and the *on_floor constraint*. The computing time vs. the number of objects is shown

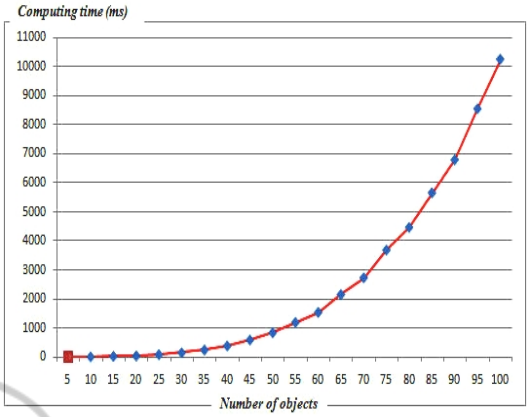

Figure 10: Computing time vs. number of objects.

in Figure 10. We observed that the computing time is less than 1 sec. for a number of objects less that 50. This is acceptable since our application (3D layout of military vehicles or shelters) involves less that 50 objects. In addition, the computing time of a constraint solver highly decreases with the number of constraints. In our example, we considered two types of constraints, and we therefore expect a decrease of the computing time and a reduced number of solutions with the addition of new constraints.

7 CONCLUSIONS

We proposed to combine Virtual Reality (VR) and Constraint Programming (CP) techniques in order to assist the users in realizing 3D objects layout in restricted spaces. The proposed approaches are based on a tight communication between a 3D rendering module and a highly efficient constraint solver (GeCode). Layout modification are translated in incoming queries to the solver which generates the solutions that satisfy some predefined constraints. Some basic geometric constraints have been implemented and tested. Results showed that the response time of the system is less than 1 sec. for a number of objects less that 50. In the near future we will define and implement some semantic constraints and other variables such as objects orientation. In addition we will investigate the evolution of the computing time for different geometric and semantic constraints.

REFERENCES

Andersson, M., Carlsoon, C., Hagsand, O., and Stahl, O. (1993). Dive - the distributed interactive virtual envi-

ronment - tutorials and installation guide. In *Swedish Institute of Computer Science*.

Axling, T., Haridi, S., and Fahlen, L. (February 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 Press.

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*.

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.

Glencross, M. and Murta, A. (Manchester, 1998). Multibody simulation in virtual environments. In *Proceedings of 12th European Simulation Multiconference*.

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

IBM (2010). Ilog products and solutions, http://ftp.ilog.fr/products/cp.

Jacquenot, G. (2009). Méthode générique pour l'optimisation d'agencement géometrique et fonctionnel. *Thése 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. (2010). Interactive configuration of restricted spaces using virtula reality and constraints programming techniques. In *International Conference on Computer Graphics Theory and Applications (GRAPP 09), May 17-21, Angers, France*, pages 384–389.

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

Richard, P., Chamaret, D., Inglese, F., Lucidarme, P., and Ferrier, J. (2006). Human-scale virtual environment for product design: Effect of sensory substitution. *The International Journal of Virtual Reality*, 5 (2) (2006):p. 37–44.

Sanchez, S., Roux, O. L., Inglese, F., Luga, H., and Gaildart, V. (2003). Constraint-based 3d-object layout using a genetic algorithm. In *International Conference on Computer Graphics and Artificial Intelligence (3IA 2003), May 14-15, Limoges, France*.

Schulte, C. (1997). Oz explorer: A visual constraint programming tool. *Proceedings of the 14th International Conference on Logic Programming, July 8-11, Leuven, Belgium*, pages 286–300.

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

Solnon, C. (2003). Programmation par contraintes. *http://www710.univ-lyon1.fr/ csolnon/Site-PPC/e-miage-ppc-som.htm*.

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