Multi-way constraints for describing high-level object behaviours in VRML

Philippe Codognet (INRIA), Nadine Richard (INRIA/ENST)

{Philippe.Codognet,Nadine.Richard}@inria.fr

INRIA Rocquencourt – BP 105, 78153 Le Chesnay, FRANCE
ENST – 46 rue Barrault, 75013 Paris, FRANCE
Fax: (+33) 1 45 81 31 19

Abstract

The aim of this paper is to advocate for the development of high-level tools such as object-oriented features and constraint programming inside and outside VRML, in order to facilitate the description of complex object behaviours in virtual worlds.

1 Introduction

VRML[1] has been quite successful in simplifying the description of 3D animated objects because the author does not have to care about low-level computer graphics problems such as rendering. But it is not high enough a level to easily describe "active" objects[2]: VRML should evolve from a 3D animated scenes to a 3D virtual worlds description language with high-level features such as object-orientation and constraint processing.

Constraint programming has proved to be very successful for problem solving and combinatorial optimization. It combines the expressiveness of a high-level language with the efficiency of specialized algorithms for constraint solving, sometimes borrowing techniques from operations research and numerical analysis. Other types of applications might benefit from this duality; in this paper, we will focus on constraint programming applied to Virtual Reality.

Constraints have already been successfully used in 2D authoring systems for some time [Gleicher94, Gleicher94b] in order to maintain basic object geometric properties while drawing, and more recently in 3D animation [Gleicher97, Gobbetti95] for motion and kinematics control. These notions have been very recently introduced in VRML [Diehl97]: the proposed system extends VRML++[3] [Diehl97b] by adding "one-way" constraints. This system already enables the author to describe some interesting complex behaviours, but we propose to implement true multi-directional constraints as in classical constraint solvers.

The main idea is that constraints can be used to enforce hidden relationships between 3D objects and further describe the behaviour of "animated virtual agents" in

[1] In this paper, VRML (Virtual Reality Modeling Language) stands for VRML97, the current VRML ISO standard.

[2] "Active objects" designates autonomous agents living in a dynamic and unpredictable virtual universe.

[3] VRML++ introduces object-oriented features to VRML, such as classes and inheritance.

VRML worlds. Interesting examples could be positional constraints such as minimal and maximal distances or non-collision, or general integrity rules such as gravity. The simulated world should not depart too much from the one imagined by the VRML designer: the world must be as coherent and realistic as possible but the constraint solver must slightly interfere with real-time animation.

2 Constraint Processing

Constraints are a simple way to describe strong logical relationships between variables. Those constraints must be respected as much as possible, in order to keep the system in a coherent state. Each variable is bound to a "value domain" that contains all the possible values this variable can take. When a constrained variable is modified, its new value is propagated in the constraint network in order to ensure the system consistency.

In classical constraint solvers, the system is said to be "consistent" when all the constraints are satisfied. Some solvers may also relax selected constraints so that the most important ones keep satisfied, usually the ones with the biggest priority. When some variables cannot be modified anymore and the constraint network is still inconsistent, the system must explore another path to solve the problem using for example backtracking techniques.

The whole idea of constraint solving is to start reasoning and computing with partial information, ensuring the overall consistency and reducing the variable domains as much as possible. The most classical and successful techniques used to process and solve constraints are called *consistency* or *local propagation* techniques. Such techniques stem from constraint satisfaction problems in Artificial Intelligence [Montanari74, MacWorth77, Nadel88].

Constraints are powerful tools for describing high-level relationships between entities. When integrated in a programming language such as Prolog or Java, constraints can be manipulated as any other variable, and thus be used in lists or control structures; the constraint solver is in charge of the global system consistency during the whole program execution. Constraints are truly an interesting extension for any kind of programming or description language.

3 Constraints in VRML

The VRML designer may want to set specific constraints on some of the objects composing a virtual world and let the constraint solver keep the scene coherent. Constraints may also be used to describe planning methods, triggers and actions for autonomous agents in a 3D virtual world: the designer gives some rules to active objects that must cope with an unpredictable environment.

3.1 The basic ideas

In VRML, 3D objects are described as nodes that can be hierarchically combined. Each node is made of read-only fields and in/out events. Animations are obtained by routing eventOuts of some nodes to eventIns of the same type, in order to propagate the modified values. Script nodes have been introduced in VRML to perform more complex animations by calling external programs, which are usually written in Java or Javascript.

In particular, such programs may check the consistency of the constraints applied to VRML objects. There are two different approaches to introduce constraint solving in VRML, both needing the use of Java Script nodes:

- A VRML-based constraint solver: each constraint is implemented as a Script node, which is in charge of checking the constraint consistency. Value propagation is based on the VRML event routing mechanism. The backtrack manager is also a separated Script node. The solver is composed of all those Script nodes.
- A Java-based constraint solver: the VRML scene communicates with an unique Script node representing the constraint solver. Backtracking, constraints and constrained variables are managed by a single Java program that embodies the solver.

3.2 A general architecture for the two approaches

Constraint programming systems are usually designed to find an ideal solution, which is a consistent state for an entire constraint set. Even with highly optimized solvers, it generally takes seconds or minutes to get a result for a complicated problem: in a Virtual Reality system, we need to keep an approximative coherence while preserving the quality of the real-time animation.

Therefore we chose a classical but simplified architecture for the constraint solver. There are different types of constraints (applied to floats, integers, strings, etc.) but we focused on binary positional constraints, that is constraints between two 3D vectors: then we can solve typical 3D animation problems like collision and proximity detection between any two objects in the scene[4], but also classical constraint solving problems such as the well-known n-Queens challenge.

Our system consists of 3D objects, binary constraints and a backtrack manager. An object is compound of fields that may be constrained; it must define a "normal" and a "special" behaviour: the normal behaviour describes the default animation planned in the general case, while the special behaviour is activated only when one of the constraints becomes unsatisfied. Constrained variables have also two value domains (a normal and a special one) to handle conflictual situations.

There are two different kinds of priorities in this system: *variable priorities* and *constraint priorities*. The priorities applied to the constraints are managed by the backtrack manager and indicate which constraints are the most important for the world coherence. The priorities given to the constrained variables allow the constraint to choose which one of the two fields must be modified first when there is a conflict: it can be a value (the field with the lowest priority will be chosen) or a rule (random choice, "the last modified variable must change" or "the other variable must change").

3.3 A classical example : the Queens as Agents

Here is a well-known example: the n-Queens problem consists of placing n Queens on a $n \times n$ chessboard while never having two Queens on the same line, column or diagonal. Considering that there should be only one Queen on each line, we will note Q_i the Queen on the i^{th} line. Lines are represented on the Y axis of the VRML scene. We shall thus have for each couple of Queens Q_i and Q_j :

$$Q_j.X \neq Q_i.X$$
$$Q_j.X \neq Q_i.X + j - i$$
$$Q_j.X \neq Q_i.X - j + i$$

[4]This is possible in Java3D (according to the specification [Java3D]) but not in VRML97.

We chose to process the 3 disequations at the same time using a unique binary constraint called **NoAttack**. This constraint guaranties for each couple of Queens that they are not on the same column or diagonal. Each Queen has a small domain, that is the n possible positions on the X axis.

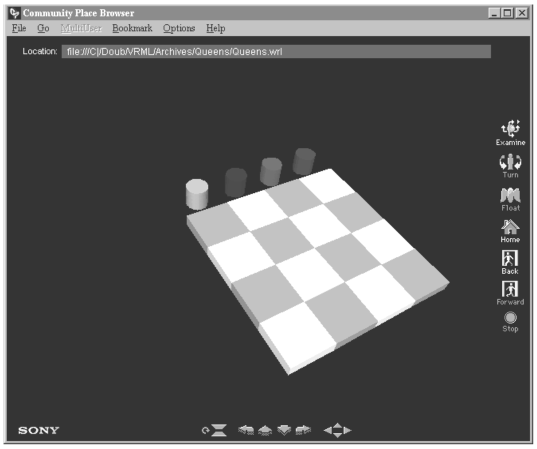

Fig. 1: *The 4 Queens before launching the constraint solving.*

When the user clicks on Q_1 (the first cylinder from the right), the Queen will try to find a place to go on the chessboard: it will stop at the first place because there are no other Queen on the board and therefore no conflict to solve. Then the user clicks on Q_2, which cannot choose the first or the second position on the board, because there is already a Queen on the first column and the first diagonal: the **NoAttack** constraint between Q_1 and Q_2, in association with the appropriate priority ("the last who has been modified must change"), makes Q_2 choose the third position. After several clicks, the solver finds a first solution:

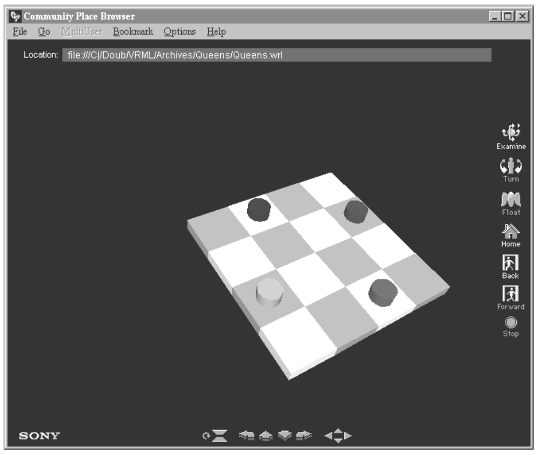

Fig. 2: *The 4 Queens are now correctly on the board.*

We consider that those Queens are similar to autonomous agents as far as they can modify their position depending on the situation, and they may possibly ask the backtrack manager to rearrange the problem when there is an unsolvable conflict.

This problem has been implemented with the VRML-oriented solution for only four Queens, because of the problems encountered with the VRML browsers available at this time which were unable to load larger files or a Java-based constraint solver[5].

4 Conclusion

We had many problems with the VRML 2.0 browsers that were available in summer 97. Therefore, we could not validate the complete results of our project but we can already say that, from a virtual world designer point of view, an authoring system with high-level features is more efficient and comfortable than a system based only on VRML current functionnalities.

VRML would certainly benefit from some extensions in order to evolve towards a high-level description language including object-oriented features such as inheritance and encapsulation, and optimized specific both one-way and multi-way constraint solvers. But even then, we would need a programming language to describe more complex object behaviours. Thus, it could be interesting to discuss about what is needed at the VRML level and what has to be programmed separately.

References

[Beeson97] Curtis Beeson – *An Object-Oriented approach to VRML Development* – 1997

[Diehl97] Stephan Diehl – *Extending VRML by One-Way Equational Constraints* – In: proc. CP'97 post-Workshop on Constraints and the Internet, 3rd International Conference on principles and practice of Constraint Programming, Linz, Austria, October 1997.

[Diehl97b] Stephan Diehl – *VRML++: adding classes to VRML* – `http://www.cs.uni-sb.de/RW/users/diehl/vrml++/content.html`, 1997.

[Gleicher94] Michael Gleicher – *Practical issues in Graphical Constraints* – In: proc. PPCP'94, 2nd Workshop on Principle and Practice of Constraint Programming, 1994.

[Gleicher94b] Michael Gleicher and Andrew Witkin – *Drawing with constraints* – The Visual Computer, vol. 7, 1994.

[Gleicher97] Michael Gleicher – *Motion Editing with Spacetime Constraints* – In: proc. 1997 Symposium on Interactive 3D Graphics, ACM Press, 1997.

[Gobbetti95] E. Gobbetti and J-F. Balaguer – *An Integrated Environment to Visually Construct 3D animations* – In: proceedings of SIGGRAPH 95, ACM Press, 1995.

[Java3D] *JavaTM 3D API Specification* – `http://www.javasoft.com/products/java-media/3D/forDevelopers/3Dguide/j3dTOC.doc.html`, August 1997.

[Montanari74] U. Montanari – *Networks of constraints: Fundamental properties and application to picture processing* – Information Science, vol. 7, 1974.

[MacWorth77] A.K. MacWorth – *Consistency in Networks of Relations* – Artificial Intelligence, vol. 8, 1977.

[Nadel88] B.A. Nadel – *Constraint Satisfaction Algorithms* – Computational Intelligence, vol. 5: pp 188-224, 1989.

[Park97] Sungwoo Park and Taisook Han – *Object-Oriented VRML for Multi-User Environments* – 1997

[VRML] *The Virtual Reality Modeling Language, International Standard ISO/IEC 14772-1:1997* – `http://www.vrml.org/Specifications/VRML97/`, 1997.

[5] The implemented Java solver works well when used in stand-alone, that is when an external Java program simulates input events. We hope that future VRML browsers will be able to load bigger Java scripts in order to finalize this proposal.