Valentin Popov
University of Poitiers
popov@giotto.univ-poitiers.fr

Lozka Popova
University of Poitiers
popova@giotto.univ-poitiers.fr

Giovanni De Paoli
University of Montreal
depaolig@ere.umontreal.ca

Towards an Object-Oriented Language for the Declarative Design of Scenes

We propose a prototype "kernel" of an object-oriented language, SOML (Scene Objects Modeling Language), intended to assist in the declarative design of scenes in image synthesis. This language is an attempt to provide the designer with a tool to facilitate the rapid prototyping of 3D scenes. It can also serve as a tool for knowledge acquisition and representation , and for communication and exchange of data with other tools in a design environment.

Advantages offered by the implementation of SOML are: (a) from user's viewpoint: the possibility of declarative description of the initial concept associated with the target scene in terms of properties and constraint vocabulary, the possibility of quantitative and qualitative reasoning on these properties, the modification of the intermediate solutions to different levels of detail, the utilisation of previous solutions; and (b) from the implementation viewpoint: the structuring of the properties and methods in the form of domain knowledge, the optimal solution generation according to heuristic causal-probabilistic criteria, the transformation of the semantic concept description of the scene in generic entry code for a geometrical CSG modeler or for rendering and visualization software, the integration of functionality for parameter generation and modification, the compilation of a scene from components of other final scenes and operations of geometrical transformations acting on groups of scenes.

We present the architecture of the object-based implantation of the language and its interpreter, in the unified notation formalism UML. The utilization of the SOML language is illustrated by some examples.

Vers un langage à objets pour la conception déclarative de scènes

Nous proposons un prototype de «noyau» à l'aide d'un langage à objets SOML (Scenes Objects Modeling Language) destiné à aider à la conception déclarative de scènes par la synthèse d'images. Ce langage est une tentative de fournir au concepteur des outils de création rapide de prototypes de scènes 3D. Il peut servir aussi comme moyen d'acquisition et représentation de connaissances, de communication et échange de données avec d'autres outils dans un environnement de conception.

Les avantages offerts par la mise en oeuvre de SOML sont (a) du point de vue utilisateur: la possibilité de description déclarative du concept initial associé à la scène en termes de vocabulaire de propriétés et contraintes, la possibilité de raisonner de manière qualitative et quantitative sur ces propriétés, la modification de solutions intermédiaires à différents niveaux de détail, la réutilisation de solutions anciennes; (b) du point de vue mise en oeuvre: la structuration des propriétés et des méthodes sous forme de connaissances du domaine, la génération de solutions optimales selon des critères heuristiques causal-probabilistes, la transformation de la description des concepts sémantiques de scènes en code générique d'entrée d'un modeleur géométrique CSG (Constructive Solid Geometry) et du logiciel de visualisation, l'intégration des fonctionnalités de génération et modification de paramètres, la compilation d'une scène à partir de composants d'autres scènes finales et d'opérations de transformations géométriques de groupes de scènes.

Nous présentons ensuite l'architecture de l'implantation à objets du langage et son interpréteur, dans le formalisme de notation unifiée UML (Unified Modeling Language). L'utilisation du langage SOML est illustrée par quelques exemples.

introduction

This study is in the scope of research on the creation of tools to assist in the design of scenes in image synthesis. Independent of its particular interpretation, the concept of scene is used in order to represent a finite subset of objects within an application domain, defined formally in the *universe* paradigm (Gomes 1995; Gomes 1996). The notion of design is exploited in different domains of man's activities. It is a concept, associated with the set of creative activities in society (Trousse 1989). Its underlying methodologies face a significant evolution in the domain of mechanical and electrical products manufacturing, and in the field of architecture. In general terms, design is a cyclic process of composition (assembly) stages of complex structures from completely specified components, and the decomposition of structures, the information of which lacks precision. The duration of the design process is usually determined by how long it takes to attain stable solution states.

In image synthesis the objects considered are the components of the scene structure, the construction and representation methods in 3D space, and its attributes. An object is characterized by a geometrical shape of any complexity, together with physical features, which could be represented in a realistic manner on the computer screen as graphical image. In terms of computer representation, the design in image synthesis encompasses various methodologies, integrating geometric models, photometric models, cognitive models and their exploration for the construction and instantiation of various solution classes.

From a cognitive viewpoint, the scenes represent a way of encoding of complex information and its perception, and of reasoning about its unknown aspects belonging to numerous underlying problems. With the growing need for operational models of this type, one notes during their creation and exploration, the existence of one major problem. It is difficult to control the geometric appearance, usual, functional, technological etc. aspects of complex 3D scenes at all the stages of their life cycle. This has led to the development of methodologies based on the creation of CAD tools for design assistance.

The current CAD market is dominated by systems designed for specific applications, having only a limited access to databases (Kehrer 1996). In these design assistance systems, the objects considered are explicit in the case where knowledge of all their parameters is complete, characteristic of a "from-the-ground-up" design. This fact presents a limitation for the designer because he doesn't have the option of exploring partial solutions. Often, in the case of over-constrained problems, it is useful to be able to relax the constraints in the intermediate solutions. These systems are articulated essentially around the geometrical aspect, missing high level semantics better adapted to the thought of a designer. As Charman (1995) underlines it, the geometry is only an artifice of manipulation for a universe with complex semantics.

One step toward the improvement of the disign-assistance tools is presented by the works dedicated to the creation of declarative CAD (Miaoulis 1996; Colin 1997). The design of scenes by the declarative approach consists in offering ways to express the design needs in linguistic terms at every stage of the design cycle. The creation of a final scene necessitates many repetitive attempts of decomposition of the found solutions (intermediate scenes) into their constituent elements in order to refine them in accord with the goal requirements and new constructions of the scene. It is a process of manipulation of initially unknown semantically rich properties in the cycle description- generation-knowledge understanding (Lucas 1990), the process of declarative design.

Among the numerous problems to be dealt with in this approach, it is necessary to underline the difficulty of optimal management of the number of solutions coherent with the imposed constraints, of providing for the reusability of the experience of other designers, and of developing a user-friendly description and control language, close to the thought of the user and covering all the stages of the design process. A promising approach in order to address these difficulties resides in the development of knowledge based "intelligent" CAD systems. These systems represent a new way to deal with the problem of limited assistance in CAD by integrating design knowledge into the geometri-

introduction

Cette étude s'inscrit dans un cadre de recherche visant la création d'outils d'aide à la conception de scènes en synthèse d'images. Indépendamment de son interprétation particulière, le concept de scène s'utilise toujours pour représenter un sous-ensemble fini d'un de ses domaines d'application, définis formellement dans le paradigme univers (Gomes 1995, 1996).

Ces dernières années, la notion de *conception* est exploitée dans différents domaines d'activités de l'homme. C'est un concept, associé à l'ensemble d'activités créatives dans la société (Trousse 1989). Les méthodologies créées à sa base ont subi une grande évolution dans le domaine de fabrication de produits mécaniques, électroniques et dans le domaine de l'architecture. En termes généraux, la conception est un processus d'enchaînement cyclique d'étapes de composition de structures complexes à partir de composants complètement spécifiés et la décomposition de structures, dont les informations manquent de précision. La durée d'un processus de conception est habituellement limitée par l'aboutissement d'états de stabilité des solutions obtenues.

En synthèse d'images, les objets à concevoir sont les composants de la structure de la scène, des méthodes de construction et représentation dans l'espace 3D et ses attributs. Un objet est caractérisé par une forme géométrique de complexité quelconque, munie de caractéristiques d'apparence physiques, qui peut être représentée de manière réaliste sur l'écran de l'ordinateur sous forme d'une image graphique. En termes de représentation informatique, la conception en synthèse d'images comprend des méthodologies, intégrantes des modèles géométriques, modèles photométriques, modèles cognitifs et leurs explorations pour la construction et l'explicitation des classes de solutions.

Du point de vue cognitif, les scènes représentent un moyen de coder l'information complexe, de la percevoir et de raisonner sur ses aspects inconnus faisant partie de nombreux problèmes sous-jacents. Avec le besoin de plus en plus nécessaire de tels modèles performants, on constate durant leur création et exploration, l'existence d'un problème majeur. Il s'avère difficile de maîtriser l'aspect géométrique, d'apparence, fonctionnel, technologique etc. de scènes complexes 3D à toutes les étapes du cycle de leurs vies. Tous ceci a provoqué l'apparition des méthodologies d'assistance, basées sur la création d'outils CAO d'aide à la conception.

Le marché actuel de la CAO est dominé par des systèmes destinés aux applications spécifiques, n'ayant qu'un accès limité aux bases de données (Kehrer 1996). Dans ces systèmes d'aide à la conception, les objets à concevoir ne sont explicités que dans le cas où les connaissances de tous ses paramètres sont complètes, donnant une conception ascendante. Ce fait présente une limitation pour le concepteur car il n'a pas de possibilités d'expliciter des solutions partielles. Souvent, dans le cas de problèmes sur-contraintes, il est utile de pouvoir contrôler la relaxation des contraintes dans les solutions intermédiaires. Ces systèmes sont articulés essentiellement autour de l'aspect géométrique, manquant de sémantique de haut niveau et aussi l'adaptation à la pensée du concepteur. Comme le souligne Charman (1995), la géométrie n'est qu'un artifice de manipulation d'un univers dont la sémantique est complexe.

Un pas vers l'amélioration de l'assistance est présenté par les travaux dédiés à la création de *CAO* déclaratives (Miaoulis 1996; Colin 1997). La conception de scènes par l'approche déclarative consiste à offrir des moyens pour exprimer en termes de langage les besoins de chaque étape du cycle de l'évolution du processus de conception.

La création d'une scène finale nécessite de nombreux essais répétitifs de décomposition des solutions trouvées (scènes intermédiaires) aux éléments à raffiner en accord avec les exigences des buts et des nouvelles reconstitutions de la scène. C'est un processus de manipulation des propriétés sémantiquement riches, jamais connues auparavant, dans la boucle description-génération-prise de connaissances (Lucas 1990), ce qui signifie une tâche de conception déclarative. A cause du nombre de problèmes à traiter dans cette approche, il faut souligner la difficulté de gestion optimale du nombre des solutions cohérentes avec

cal models (Kehrer 1996; Vargas 1995; Trousse 1989). The modeling and the exploitation of this knowledge is possible thanks to the couplings of the CAD systems with other advanced computing tools, in particular the coupling of traditional modelers based on object-oriented languages, inference engines, systems of constraint resolution, and methods of qualitative and quantitative reasoning.

With the exception of the work on declarative modeling in Geode (1997), which is dedicated to the problem of scene design within multimedia computer systems, one notes a lack of scientific communications about the use of such systems. The reasons for this lack despite the importance of the topic, are the diversity, complexity and multidisciplinary character of the field. Nevertheless one finds some descriptions of systems and utilities partially covering the needs of the different design stages. These include tools that collect various abilities to facilitate geometric modeling and realistic scene visualization.

Despite the obvious progress of systems for design assistance, one notes some limitations, summarized by Kehrer and Vatterrott: an insufficiently open evolution, a lack of configuration possibility, poor integration capacity, lack of possibility for operational exchange between the systems, lack of capacities of cooperative design, obstacles to the migration of the application domains, absence of modularity, and awkward user interfaces.

The interest of our work lies in the proposition of one methodological support for the creation of a "kernel" of a language integrating tools for synthetic scene design, SOML (Scene Object Modeling Language), based on the "declarative" principle. The goal of the language is to offer functionality to guide the designer in the activities defined by the Model of the Process of Scene Design (MPCS) throughout the different stages of the design cycle. For the definition of the MPCS model we are inspired a great deal by the works of C. Vargas presenting the mechanical design as a tree-like structure made up of tasks / methods (Vargas 1995). In their TROPES System, Genzel and Girard represent the tasks as cognitive units within a system for knowledge representation by multiple-viewpoint objects (Genzel 1997). In the case of synthetic scenes, the MPCS refers to the conceptual and logical models of the scene components and gives the answer to the question of how to design the scene. It allows one to consider the design from a new angle, like being a problem based on hybrid knowledge resolution.

The work we present here is a contribution to current CAD applications' requirements of flexibility, ease of use, possibility of integration with other tools, and functionality. In the sections that follow, we explain the context and essential features of the SOML kernel. Before concluding on the prospects for future research, we present the result of one experiment.

problems of design languages

Language in general, thanks to its power of expression, is a good existing way in man's possession to describe, model and exploit the world. Computer technology today has provided a panoply of languages, varied according to the different needs and working techniques: imperative, functional, logical and lately object-based (Oussalah 1997). In our study we are particularly interested in the implementation of languages capable of offering functionality that could have as its objective the design of scenes, that is, activities of description and knowledge acquisition, generation of parametric solutions and their processing, gaining information by means of visualization, making modifications as necessary, and communication with the environment.

Within the methodology of declarative modeling, for the description of the properties and the constraints according to different points of view, one uses a languages of "scripts." Given this, one obtains descriptions that constitute the external declarative model (Plemenos 1991; Desmontil 1995). These are languages of reduced syntax, structured on some restricted vocabularies, oriented toward modeling. They are restricted to represent only the semantics of selected properties. The internal geometric model, created by a declarative modeler and the process of its exploration (Plemenos 1991; Martin 1989) is coded in terms of imperative universal languages like Pascal, C, or of logical languages (Prolog, Lisp). All these languages present the inconvenience that they can not bring

les contraintes imposées ainsi que la réutilisation de l'expérience d'autres concepteurs et le besoin d'un langage convivial de description et de contrôle, proche de la pensée de l'utilisateur et couvrant toutes les étapes du processus de conception. Une approche prometteuse pour répondre à ces difficultés réside dans le développement de systèmes CAO «intelligents» basés sur les connaissances.

Ces systèmes représentent une voie nouvelle pour aborder le problème d'assistance réduite en *CAO* en intégrant aux modèles géométriques des connaissances de conception (Kehrer 1996; Vargas 1995; Trousse 1989). La modélisation et l'exploitation de ces connaissances est possible grâce aux couplages des systèmes CAO avec d'autres outils informatiques avancés, parmi lesquels on distingue les cas de couplage des modeleurs tarditionnels basés sur les langages orienté objets, des moteurs d'inférence, des systèmes de résolution de contraintes, des méthodes de raisonnement qualitatif et quantitatif.

Mis à part les travaux de modélisation déclarative Geode *(*1997*)*, dédiés aux problèmes de la conception de scènes dans le cadre de systèmes informatiques multimédia et la méthodologie de conception à base déclarative, on constate un manque de communications scientifiques pour la mise en œuvre de tels systèmes. Les causes de cet état sont à chercher, malgré l'importance de sujets, dans la diversité, la complexité et le caractère multidisciplinaire du domaine. Néanmoins, on trouve des descriptions des systèmes et des utilitaires couvrant partiellement les besoins d'assistance aux différentes étapes de la conception. Il s'agit de boites d'outils regroupant des utilitaires facilitant la modélisation géométrique et la visualisation réaliste des scènes.

Malgré le progrès évident des systèmes d'aide à la conception, on constate ses limites, résumées par Kehrer et Vatterrott comme: l'ouverture insuffisante d'évolution, le manque de possibilité de configuration, la pauvre capacité d'intégration, le manque de possibilité d'échange opérationnel entre les systèmes, le manque de capacités de conception coopérative, l'existence des obstacles pour la migration des domaines d'application, l'absence de modularité, le manque des interfaces utilisateurs suffisamment conviviales.

L'intérêt de notre travail porte sur la proposition d'un support méthodologique pour la création d'un «noyau» avec un langage intégrant des outils d'aide à la conception de scènes synthétiques SOML (Scene Objects Modeling Language) basé sur le principe «déclaratif». L'objectif du langage est de proposer des fonctionnalités permettant de guider le concepteur à effectuer des activités prévues par le Modèle du Processus de Conception de Scènes MPCS aux différentes étapes du cycle de vie. Pour la définition du modèle MPCS, nous nous sommes inspirés des grandes lignes des travaux de C. Vargas présentant la conception en mécanique comme une structure arborescente de tâches / méthodes (Vargas 1995), Genzel et Girard représentant les tâches comme des unités cognitives au sein du système de représentation de connaissances par objets multi-points de vue TROPES (Genzel 1997). Dans le cas de scènes synthétiques, le MPCS se réfère aux modèles conceptuels et logiques des composants de la scène et donne la réponse à la question comment concevoir la scène. Il permet d'aborder la conception sous un nouvel angle, comme un problème de résolution à base de connaissances hybrides.

Le travail présenté est un effort de contribution aux exigences actuelles des applications CAO de flexibilité, facilité d'utilisation, possibilité d'intégration d'autres outils et fonctionnalités. Dans les sectins suivante, nous exposons le contexte et les caractéristiques essentielles du noyau SOML. Avant de conclure sur les ouvertures à des futures recherches, nous présentons le résultat d'une expérimentation.

problèmes des langages dans la conception

Le langage en général, grâce à son pouvoir d'expression, est le meilleur moyen existant en possession de l'homme pour décrire, modéliser et exploiter le monde. La technologie informatique actuelle a fourni une panoplie de langages, diversifiés selon les différents besoins et techniques de fonctionnement: impératifs, fonctionnels, logiques et dernièrement par objets (Oussalah 1997). Dans notre étude nous nous sommes intéressé particulièrement à la mise en œuvre de

out the modularity and the semantics of concrete concepts presented by the scene models. They remain far from the thought of the designer and the linguistic schemes supported by natural language. They don't permit a comfortable structuring of the domain knowledge, nor the intervening of the designer in the cycle of design evolution.

In the field of product design, there are languages better adapted to the particular needs. For example in the design of mechanical parts we have the use of languages like "Maily" (Trousse 1988), "DDL" (Vargas 1995), "FDL" (Brunetti 1996), "EREP" (Chen 1995), as well as others. There also exists a group of languages designed to model the activities of computer systems in general, independently of the application domain, collecting the methods relevant to their application. The best-known are Common KADS, Merise, Graffcet etc. (Vargas 1995). They all have the constraint to be not directly usable for a particular domain.

In terms of knowledge representation and utilization within a system of scenes design assistance (Trousse 1989), the object-oriented paradigm enables the class based structuring of static knowledge (fields of attributes), and of dynamic knowledge (methods). In this context a class determines a generic model that permits the "instantiation" of similar objects. The cognitive aspects of the design assistance systems is partially covered by the languages for knowledge modeling, based on the notion of production rules, semantic networks and frames (SHIRKA, KRL, KI-ONE).

In image synthesis, there are have been developed script- based interpreted languages used for modeling, and the animation of scenes like "Put" (Clay 1996), "Smile" (Argues 91), "Sphigs" (Foley 1995), "Mira" (Thalman 1988), "Fabule" (Gascuel 1996), and of languages giving priority to the visualization like "Pov-ray²" (Dif 1996), "MSDL²" (Gatenby 1993), "RayShade," "Radiance," and "RenderMan." They all were conceived in order to answer to the needs for tools, especially in the research laboratories of computer graphics and mechanical CAD. In all these cases particular formats for graphical data were created(more than 40 different types). The type of the languages is procedural, using as arguments the parameters of the geometrical primitives. These languages limit to support only some stages of the evolution cycle of the designed objects. In the case of complex scenes, the control on the design parameters, in terms of structuring and of reusability of codes, becomes difficult. In the case of coupling of systems based on knowledge resolution, it is necessary that the design language permit a global support on all the levels of the design process.

In the following section, we show details of the philosophy of integration, in one specialized scene design language, of a functionality permitting the designer to create from solid primitives a CSG-constituted synthetic 3D scene, to represent the methods of its assembly and modification in terms of methodical and causal probabilistic knowledge, and to interact with the environment or other scenes. The causal-probabilistic solution allows one to introduce the concept of hypothetical covering, reflecting the non-determinist nature of the possible semantic interpretation associated with the properties related to the concept of the target scene. In this context, the result of the constraint resolution is the choice of one subset of coverings (set of instance parameters) relevant to the constraints for direct causes, sorted according to the criterion of maximal likelihood (Peng 1990).

For the conceptualization of a system we adopted the UML formalism (Lai 1997), becoming a standard for the biggest publisher of industrial object-oriented software. The formalism of object UML modeling was created by the OMG Association with the goal of unifying the best existing methods of analysis and object-oriented design. This formalism is the most advanced one at present, and it makes possible the elaboration of conceptual and logical models of complex systems.

structure and main aspects of SOML

The approach adopted in the choice of SOML structure is a hierarchical layering of the functionality, implemented in the context of a design-assistance environment (Figure 1. UML-diagram of the object-oriented conceptual model of SOML).

The components of the environment are accessible by means of UNIX shell processes. One distinguishes the following layers: knowledge, mod-

langages capables de proposer des fonctionnalités qui puissent avoir comme objectif la conception de scènes, c'est-à-dire, des activités de description et l'acquisition de connaissances, la génération de solutions parametrées et leurs instances, la prise de connaissance par visualisation, la modification en cas de besoin, la communication avec l'environnement.

Au sein de la méthodologie de modélisation déclarative, pour la description des propriétés et des contraintes selon différents points de vue, on utilise des langages à «scripts», grâce auxquels on obtient des descriptions qui constituent le modèle déclaratif externe (Plemenos 1991; Desmontil 1995). Ce sont des langages à syntaxe réduite, bâti sur des vocabulaires restreints, orientés vers la modélisation. Ils se limitent à représenter la sémantique des propriétés visées. Le modèle géométrique interne, créé par un modeleur déclaratif et le processus de son exploration (Plemenos 1991; Martin 1989) sont programmés en termes de langages impératifs universels comme *Pascal*, *C*, ou de langages logiques (*Prolog, Lisp*). Tous ces langages présentent l'inconvénient de ne pas pouvoir dégager la modularité et la sémantique des concepts concrets présentés par les modèles des scènes. Ils restent loin de la pensée du concepteur et des schémas linguistiques supportés par le langage naturel. Ils ne permettent pas une structuration aisée des connaissances du domaine, ni l'intervention du concepteur dans le cycle d'évolution de la conception.

Dans le domaine de conception de produits, nous trouvons des langages mieux adaptés aux besoins particuliers. Par exemple dans la conception en génie mécanique nous avons la mise en œuvre de langages comme «Maily» (Trousse 1988), «DDL» (Vargas 1995), «FDL» (Brunetti1996), «EREP» (Chen 1995), et bien d'autres.

Il existe également un groupe de langages conçus pour modéliser les activités d'un système informatique en général, indépendamment du domaine, regroupés selon les méthodes de leurs application. Les plus connus sont Common KADS, Merise, Graffcet, etc. (Vargas 1995). Ils ont tous la contrainte de ne pas être utilisables directement pour un domaine particulier.

En termes de représentation et utilisation de connaissances au sein d'un système d'aide à la conception de scènes (Trousse 1989), le paradigme orienté objets permet la structuration, sous forme de classes d'objets, des connaissances statiques (champs d'attributs de propriétés) et de connaissances dynamiques (méthodes). Dans ce contexte une classe détermine un modèle générique qui permet «l'instanciation» d'objets similaires. Les aspects cognitifs des systèmes d'aide à la conception sont partiellement couverts par les langages de modélisation des connaissances, basés sur la notion de règle de production, réseaux sémantiques et frames (SHIRKA, KRL, KI-ONE).

En synthèse d'images, se développent des langages interprétés par des «scripts» utilisés pour la modélisation et l'animation de scènes comme «Put» (Clay 1996), «Smile» (Argues 1991), «Sphigs » (Foley 1995), «Mira» (Thalman 1988), «Fabule» (Gascuel 1996), et des langages donnant priorité à la visualisation comme «POV-Ray» (Dif 1996), «MSDL » (Gatenby 1993), «RayShade» , «Radiance», «RenderMan». Ils ont été conçus afin de répondre aux besoins d'outils, surtout dans les laboratoires de recherche, d'infographie et de CAO en mécanique. Dans tous ces cas, ils ont aussi été crées avec des formats de données graphiques particulières (plus de 40 types différents). Ces langages sont de type procédural, utilisant comme arguments les paramètres des objets–primitives géométriques. Ces langages se limitent à ne supporter qu'à certaines étapes du cycle d'évolution des objets en conception. Dans le cas de scènes complexes, le contrôle sur les paramètres de la conception, en termes de structuration et de réutilisation du code, devient difficile. Dans le cas de couplage de systèmes de résolution à base de connaissances, il est nécessaire que le langage de conception permette un support global sur tous les niveaux de la conception.

Dans la section suivante nous montrons des détails sur l'idéologie d'intégration, dans un langage spécialisé pour la conception de scènes 3D, des fonctionnalités permettant au concepteur de créer des primitives de solides CSG constituant une scène synthétique, de représenter les méthodes de son assemblage et modification en termes de

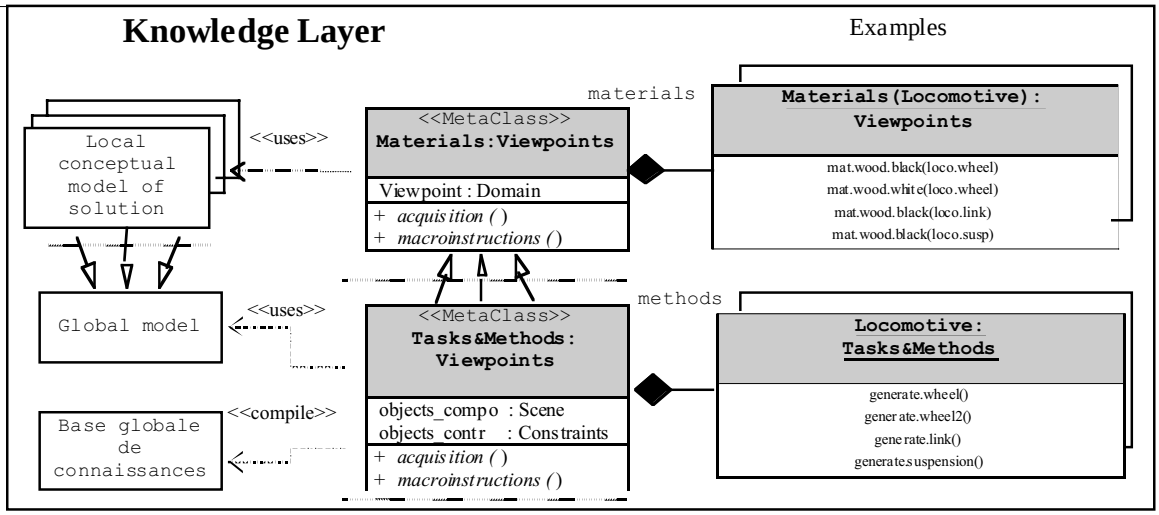


Figure 1a. Classes and instances of knowledge representation (multiple-viewpoint classification).

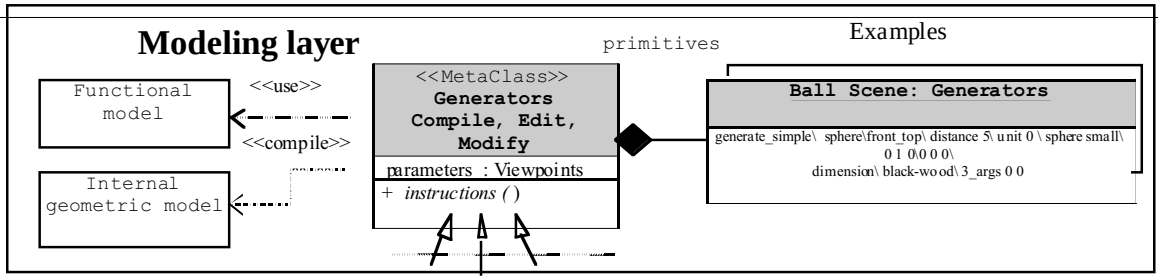


Figure 1b. Classes and instances of geometric modeling and of the photo-realistic aspect (Methodological knowledge).

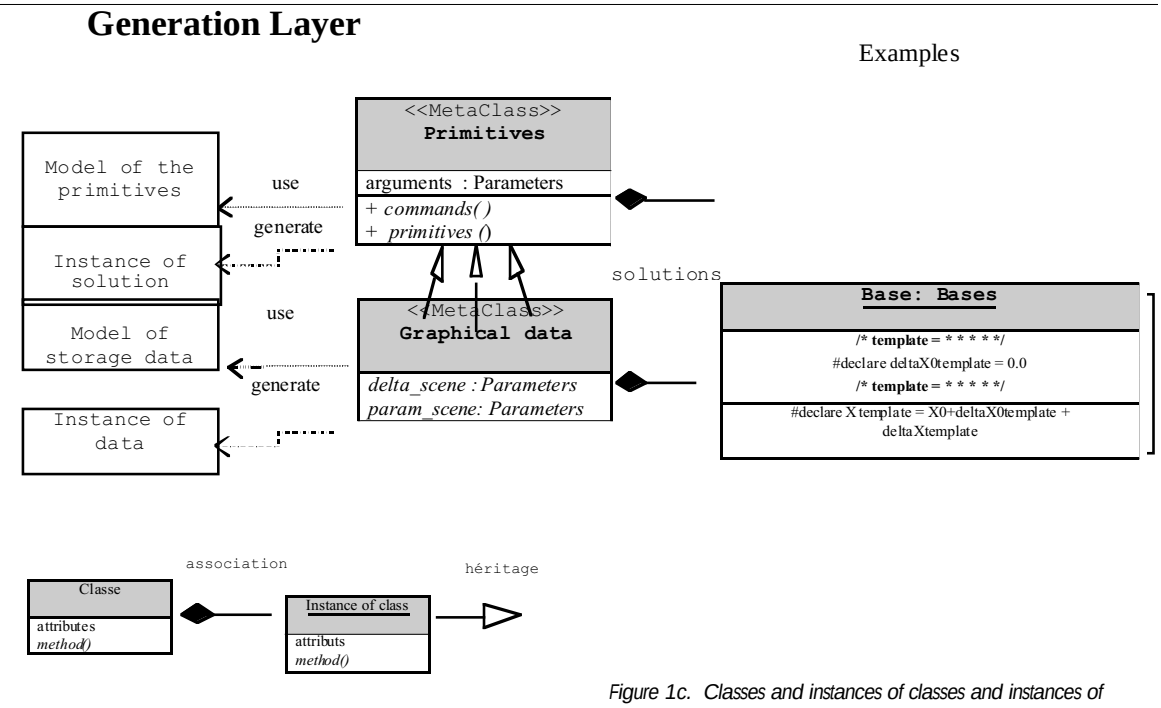


Figure 1c. Classes and instances of classes and instances of solution prototypes (satisfaction of constraints).

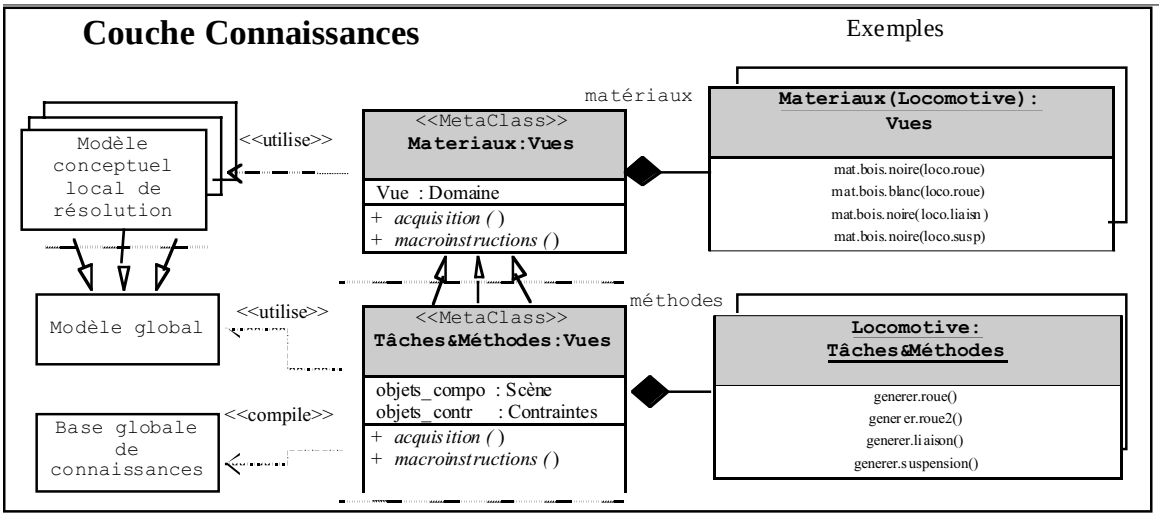


Figure 1a. Classes et instances de représentati des connaissances (classification multi-vues).

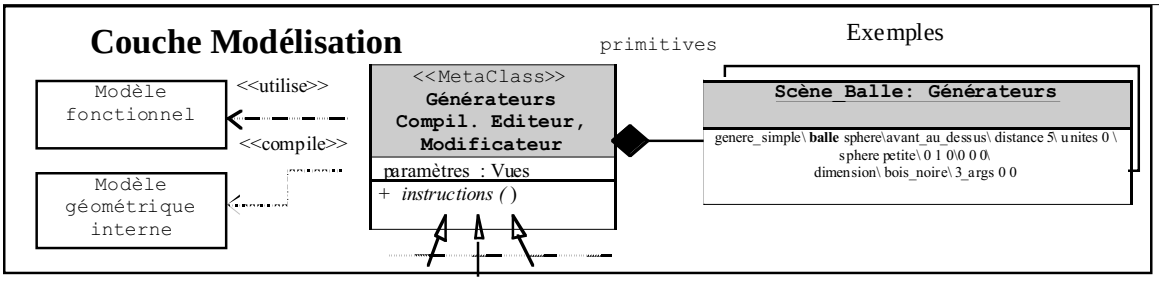


Figure 1b. Classes et instances de modélisation géométrique et de l'aspect photo réaliste (connaissances méthodologiques).

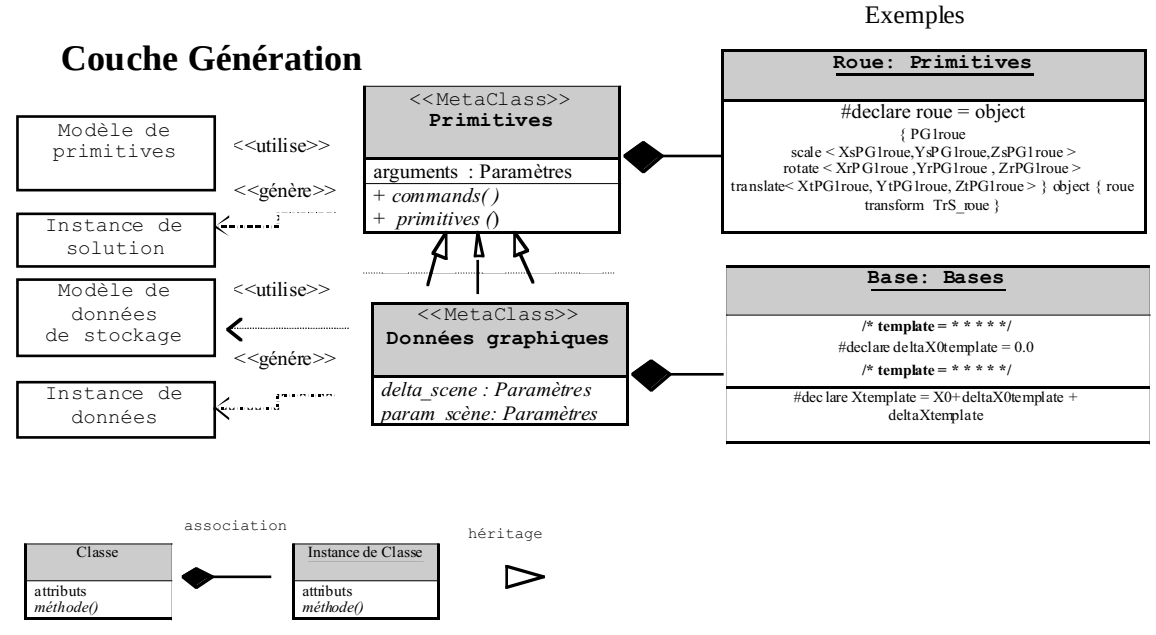


Figure 1c. Classes et instances de solutions - prototypes (résolutin de contraintes).

eling, and generation. These will be presented in this chapter. The functionality supported by SOML revolves around the three principal phases of the iterative design process; description, model generation and information gathering. This functionality is intended for the creation of cognitive geometric models with a realistic appearance, and the generation of solutions (instances). It includes actions performed on the objects constituting the design assistance environment. The principal actions are: representation, compilation, modification, duplication, classification, and inference. Each action is described in terms of macro-instructions, instructions, SOML commands and arguments; structures according to a multi-layer hierarchy, to be described below.

A real or virtual scene can be described as a Sy system with different states–initial, intermediate, or final. Such a system is capable of transforming a set of properties observed upon entry into a new set of properties observed upon exit. These properties resemble constraints when they are of a geometric, mechanical, topological, or other nature. There is often a fine line between the notions of properties and constraints. We distinguish two types of properties: those which are explicitly stated by the designer, and those which are implicit, i.e. related to a particular domain of application and are not stated during the action of description. The activity of description is a process of acquisition and external representation (external declarative model) of the properties and constraints on the objects constituting the scene (and on the scene itself). This activity is expressed by a set of conjunctions of natural statements. The linguistic schemes used are based on unary and binary relations. A statement, for example, is a disjunction of pairs (attribute_identifier, value), and triples (target_attribute_identifier, value, reference_attribute_identifier), where these attribute identifiers are described by means of a limited vocabulary derived from natural language (Figure 15) and adapted to each specific case (Plemenos 1991; Lucas 1996; Desmontils 1995). The statement expresses the semantics of the property under consideration.

In terms of cognitive units, the properties are represented (action of representation) by means of classes and instances, and structured in terms of viewpoints and causal relations. The knowledge is of a hybrid type due to its heterogeneous nature (symbolic, qualitative and quantitative, causal, probabilistic). After structuring the information about the scene domain, it becomes possible to infer (action of inference) non-explicit information from it. This information is available in the knowledge database and consists of solutions in terms of coverings, introduced by Peng (1990). These solutions (coverings) represent the semantic cognitive aspect of the final solutions, produced during the activities of compilation and generation.

The action of compilation consists in the transformation of information (views, classes of properties, and causal relations) into models (geometric, photometric,functional, of construction methods, and others) in keeping with the semantics of "multiple viewpoints." The action of generation produces instances of stable solutions. The solution of a scene is the set of instances of classes of the scene's properties, created by a concrete method of construction, and satisfying the imposed constraints. In terms of data, a solution is the set of qualitative and quantitative values attained by the parameters of the construction method. Each solution is named after its construction method, and numbered based on the order in which it was created. The number of solutions can be very big, and in that case it is suitable to classify them in terms of the acceptance interval for the values of the parameters. In this manner we generate classes of solution prototypes, to be used by default. This classification sheds light on the knowledge base of hybrid, multi-viewpoint information with respect to a new solution.

The solution of a problem in declarative design is based on the integration of an analytic learning method (supported by the causal-probabilistic model, CPM) and a hybrid method of spatial constraint resolution. The causal-probabilistic model, proposed by Peng (1990) and generalized by Dubois (1995) offers analytic and probabilistic learning formalisms. This learning consists in acquiring methodological knowledge by means of few examples, often alone, and a very rich theory of the domain. The resolution of spatial constrains is a necessary activity for the instantiation of topo-

connaissances méthodologiques et de résolution causale-probabiliste et d'interagir avec l'environnement ou d'autres scènes. La résolution causale-probabiliste permet d'introduire les concepts de couverture hypothétique reflétant la nature non déterministe de l'interprétation des sémantiques possibles associées aux propriétés couvrant le concept d'une scène cible. Dans ce contexte, le résultat de la résolution est le choix d'un ensemble de couvertures (ensemble de paramètres) pertinentes aux contraintes de causalité directe, triées selon un critère de vraisemblance maximale (Peng 1990).

Pour la conceptualisation du système nous avons adopté le formalisme Unified Modeling Language-UML (Lai 1997), devenu un standard pour les plus grands éditeurs de logiciels industriels dans la modélisation objet. Le formalisme de modélisation objet UML est crée par l'association Object Management Group-OMG avec le souci d'unifier les meilleures méthodes existantes dans l'analyse et la conception orienté objets. Ce formalisme est le plus évolué à l'heure actuelle et il permet l'élaboration des modèles conceptuels et logiques dans un environnement de systèmes complexes.

Structure et aspects principaux du SOML

La solution adoptée pour le choix d'architecture de SOML est la structuration hiérarchique multicouche des fonctionnalités, implantées dans le contexte d'un environnement d'aide à la conception (Figure 1. Diagramme *UML* du modèle conceptuel des dépendances classes / instances du langage SOML).

Les composants de l'environnement sont accessibles par l'intermédiaire de processus Unix et le shell de ce système d'exploitation. On distingue les couches Connaissances, Modélisation et Génération présentées dans ce chapitre. Les fonctionnalités supportées par SOML s'articulent autours des trois principales phases, constituant les étapes du processus itératif de conception : description, génération et prise de connaissances (Lucas 1990). Ces fonctionnalités visent la construction des modèles cognitifs, géométriques, d'apparence réaliste et la génération d'instances–solutions. Elles incluent des activités exécutées sur les objets constituant l'environnement d'aide à la conception. Les activités principales sont description, représentation, compilation, modification, clonage, classification, inférence. Chaque activité est représentée en termes de macroinstructions, instructions, commandes et arguments SOML, structurés dans une hiérarchie multicouche, detaillée plus loin.

Une scène réelle (ou virtuelle) peut être représentée comme un système *Sy* avec des états différents - initial, intermédiaire ou final. Un tel système est capable de transformer un ensemble de propriétés affectées à son entrée dans un nouvel ensemble de propriétés observées à sa sortie. Ces propriétés s'apparentent beaucoup à des contraintes lorsque celles-ci sont géométriques, topologiques, mécaniques ou autres. Souvent la frontière entre les notions de propriétés et contraintes est particulièrement floue. Nous distinguons deux types de propriétés: celles qui sont explicitement énoncées par le concepteur et celles qui sont implicites, c'est-à-dire liées au domaine d'application et ne sont pas déclarées pendant l'activité de description.

L'activité *description* est un processus d'acquisition et de représentation externe (modèle déclaratif externe) des propriétés et des contraintes sur les objets physiques qui composent la scène (et sur la scène elle-même). Cette activité s'exprime comme un ensemble de conjonctions de phrases naturelles. Les schémas linguistiques utilisés sont basés sur des relations unaires et binaires. Une phrase par exemple est une disjonction de couples (identificateur_d'attribut, valeur) et de triples (identificateur_d'attribut_cible, valeur, identificateur_d'attribut_référence) où les identificateur_d'attribut, identificateur_d'attribut_(référence ou cible) sont dénotés par un vocabulaire restreint spécialisé du langage naturel (Figure 16) adopté pour chaque cas concret (Plemenos 1991; Lucas 1996 ; Desmontils 1995). La phrase exprime la sémantique de la propriété visée.

En termes d'unités cognitives, les propriétés sont représentées (activité *représentation*) par classes et instances, structurées en «*vues*» et relations de causalité (Couche Connaissances). Les

logical models of spatial configuration, and is based on spatial-temporal logic (Allen 1983, Donikian 1992).

The paradigm for our approach to solving design problems can be summarized by the following rules:

R1: *If* the designer does not know how to construct the scene, *then* the new solution = *inference* from the most similar existing solution + *duplication* of the most similar existing solution + eventual *modification*.

R2: *If* the designer knows how to contruct the scene, *then* the new solution = *description* + *representation* + *compilation* + *generation* + *classification*.

The duplication (or "cloning") operation has the aim of deriving a "generic" copy of the method of construction of the scene or its parts referenced in terms of the most similar existing solution and its instantiation. The copy is reinstantiated as a new solution in a new context (the referenced welcome scene). The new solution is renamed and can be subsequently modified. For example, the intermediate scene "hub" of Figure 18 created by the method "generate_hub" and shown in Figure 18, is derived from the intermediate scene "wheel" and modified locally without affecting the rest of the scene in terms of the properties "dimensions" and "photometric aspects." The search for existing solutions and the test for similarity is effected with the help of decisional (causal-probabilistic) heuristics, and remains outside the scope of this work.

design environment

The environment of the assisted design constitutes the application context of the *SOML* language. Its architecture is founded on the coupling of an extended geometrical modeler (MoCSG), an interpreter (ISOML) of programs (PgSOML), a module of knowledge treatment (TrCo), a graphical user interface (IGU), the software of visualization POV-Ray (RePov), and the data bases: knowledge data (BaCo), graphic data (DoGra), picture data (Dolm) and the Unix shell.

The architecture of the environment is conceived in view of providing declarative modeling of scenes, using strategies and techniques of qualitative reasoning and causal probabilistic heuristics. The reasoning task concerns the obtaining of an answer to questions about the composition (e.g., what are the components of a scene?), the geometry (e.g., which is the scene with the given proportions?), the spatial configuration (e.g., what are the possible configurations in the available space?) and the realistic aspect of some scenes (e.g., which are the scenes with the given appearance?).

The coupling of the tools accomplished through data exchange actions. This provides the advantages of management and modular manipulation of data through the language in using knowledge based optimization strategies. In the following paragraphs we present the conceptual context and the architecture of the kernel *SOML* language. They are specified in terms of conceptual and logical models, expressed by classes and their associations, in the conceptualization formalism *UML*.

knowledge layer and MCR

At this level of the hierarchy, the functionality is defined according to MCR (Model of Conceptual Resolution). This model is build on the basis of concepts stemming from the OBR (Objects-based representation) formalism of object oriented modeling of domain knowledge by multiple views (Caponi 1995, 1997) and the MCP (Causal probabilistic model) model of reasoning based on the application of the causal probabilistic strategies of resolution.

According to MCR, a scene is viewed as a cognitive model of a designed world. The properties and the constraints of the scenes and also the methods of their treatment, are defined as cognitive objects–units of knowledge. They constitute the domain knowledge according to the different points of view of the scene. The knowledge is structured on the basis of the object oriented formalism OBR, defined by the SHIRKA-TROPES methodology (Caponi 1995; Gensel 1997; Schmeltzer 1995). The MCR formalism permits the enumeration and the attachment of properties to the objects of the scenes and the application of strategies for resolution defined by MCP.

Multiple points of view, classes, instances and

connaissances sont de type hybride due à sa nature hétérogène (symbolique qualitative et quantitative, causale et probabiliste). Après la structuration des connaissances du domaine des scènes, il devient possible d'en inférer (activité inférence) des informations non explicites, disponibles dans la base de connaissance, à savoir des solutions en termes de couvertures, introduit par Peng (Peng 1990). Ces solutions (couvertures) représentent l'aspect sémantique cognitif des solutions finales, produites pendant l'activité compilation et génération.

L'activité compilation consiste à la transformation des connaissances (vues, classes de propriétés et relations de causalité) sous formes de modèles: géométriques, d'apparence photométrique, méthodes de construction, fonctionnels, et d'autres conformes aux sémantiques des «*vues*». La génération produit des instances de solutions consistantes. La solution d'une sc*ène* est l'ensemble des instances des classes des propriétés de la scène, créées par une méthode concrète de construction, étant consistantes avec les contraintes imposées. En termes de données, une solution est l'ensemble des valeurs quantitatives et qualitatives affectées aux paramètres de la méthode de construction. Chaque solution est nommée par le nom de sa méthode, énumérée par le numéro d'ordre de la génération. Le nombre des solutions peut être très grand et il est convenable de les classifier par l'intervalle d'acceptation pour les résultats de l'évaluation des propriétés paramétriques. De cette manière nous allons générer des classes de prototypes solutions, utilisées par défaut. La classification met à jour la base de connaissances hybrides multi-vues par rapport à une nouvelle solution.

La résolution d'un problème de conception déclarative est basée sur l'intégration d'une méthode analytique d'apprentissage (supportée par Causal probabilistic Model CPM) et une méthode hybride de résolution des contraintes spatiales. Le modèle causal probabiliste CPM, proposé par (Peng 1990) et généralisé par (Dubois 1995) offre des formalismes d'apprentissage analytique et probabiliste. Cet apprentissage consiste à apprendre des connaissances méthodologiques à travers peu d'exemples, souvent un seul, et une très riche théorie du domaine. La résolution de contraintes spatiales est une activité nécessaire pour l'instanciation des modèles topologiques de configuration spatiale et s'appuie sur le raisonnement spatio-temporel issu de la logique (Allen 1983; Donikian 1992).

Le paradigme de notre approche de résolution du problème de conception s'exprime par les règles suivantes :

R1: *Si* le concepteur ne sai pas comment construire la scéne, *alors* la nouvelle solutin = *inférence* de la solution existante, la plus similaire + *clonage* de la soluti existante, la pls similaire + éventuelle *modification*.

R2: *Si* le concepteur sait comment construire la scéne, *alors* la nouvelle solution = *description* + *répresentation* + *compilation* + *génération* + *classification*.

L'opération clonage vise la dérivation d'une copie "générique" de la méthode de construction de la scène (ou une partie de la scène) référencée comme solution existante et son instance la plus similaire. La copie sera reinstanciée comme une nouvelle solution dans un nouveau contexte (la scène d'accueil référencée). La nouvelle solution sera renommée et peut subir des modifications éventuelles. Par exemple la scène intermédiaire "scène hublot" de la Figure 17 crée par la méthode "génére.hublo," montrée dans l'image sur la Figure 19, est dérivée da la scène intermédiaire "scene_roue" et modifiée (activité de modification locale sans affecter le reste de la scène) selon les points de vue "dimensions" et "aspect photométrique". La recherche de solutions existantes et le test de l'évaluation de la similarité s'effectue à l'aide d'heuristiques décisionnelles (causales probabilistes) reste au dehors de l'intention de ce travail.

Environnement

L'environnement d'aide à la conception constitue le contexte d'utilisation du langage SOML. Son architecture est fondée sur le couplage d'un modeleur géométrique CSG (MoCSG) régularisé et étendu, un interpréteur (IntpSOML) de programmes (PgSOML), un module de traitement des connaissances (TrCo), une interface graphique

Concept_of_scene	*::=*	*viewpoints*
viewpoints	*::=*	*type_of_composition | classes_of_attributes*
type_of_composition	*::=*	*simple | complex*
simple	*::=*	*component [context]*
complex	*::=*	*collection | primitive GM [context]*
context	*::=*	*reference point | observation | lighting | simple | complex*
collection	*::=*	*{ component }*
component	*::=*	***name*** *id primitive GM*
classes_of_attributes	*::=*	*viewpoint_properties viewpoint_methods viewpoint_constraints*
viewpoint_properties	*::=*	*classes_spatial_configuration classes_causal_probabilistic_knowledge ...*
viewpoint_methods	*::=*	*classes_geometric_constraints | classes_numeric_intervals |*
viewpoint_constraints	*::=*	*classes_spatial_intervals | classes_constraint_size*

Figure 2. BNF definition of the OBR scene concept.

OBR. The universe of the scene is represented by a collection of cognitive objects structured hierarchically by class, instances and multiple points of view. These objects are autonomous, characterized by the object-oriented knowledge representation OBR. A generic family of objects itself is modeled by a concept, subdivided according to different points of view (Figures 1-3). The objects communicate between themselves and the environment by the help of data exchange interfaces in a special unified format for all types of designed scenes.

In terms of data the classes of properties (including the constraints) can be likened to the fields (members) of a structure. The range of possible values for the fields is defined in the space of primitive types (integers, reals, character strings, or structured data types). The objects are considered from different "viewpoints." Each viewpoint determines a set of fields visible from it. For example, the scene can be regarded as a complex geometric shape (Geometric Class of Shapes), as an object located at a precise point of space (Spatial Configuration viewpoint), as a structure composed of objects from other scenes (Class of Composition), etc. This multiple-viewpoint approach has the advantage of a local grouping of the properties classes relevant to a particular vision of the designer, in conformity with his area of expertise. The classes of instances of concepts making up a "viewpoint" are hierarchically-structured.

This procedure presents the advantage of a local and distributed approach, allowing for parallel and cooperative design of scenes. The classes are hierarchical (interms of the relation of specialization) and can be instantiated at different levels of the hierarchy. This characteristic makes it possible to simplify the modeling phase of a design session, the price being a loss of details (Plemenos 1996b) A possible inference goal could be knowledge classification, i.e. finding the class of an object of which one knows an instance. The "Methods" point of view, for example, regroups the different methods used in constructing the components of a domain. These methods make up the hierarchies of resolution tasks in terms of methodological knowledge.

Causal probabilistic knowledge and MCP. The causal-probabilistic model describes the causal relations existing among units of knowledge, denoted by the terms "causes" and "effects" (manifestations). Causal knowledge is "deeper" than the heuristic knowledge used by a designer to solve a problem. The Sy system corresponding to a description has definite states defined by n-tuples of binary attributes (Dubois 1995). According to this system, If $a_i = 1$, there are manifestations m_i; and if $a_i = 0$, they are absent. If there are no manifestations present, we say that the system is in its normal (initial) state and can be described by n-tuples of the form (0,…,0).

Let M denote a set of n possible manifestations ($m_1 \ldots m_n$). Let D denote the set of direct causes of these manifestations ($d_1 \ldots d_n$). a cause can be present or absent. To each d_i is associated a set effects (d_i) of manifestations which are a consequence of the presence of the causes d_i. The relation R on DxM is defined as $(d_i,m_i) \in R \Leftrightarrow m_i \in$ effects (d_i). And this associates the manifestations

```
Concept_of_scène   .:=  vues
vuew               .:=  type_de_composition / classes_d'attributes
type_de_composition .:= simple / complexe
simple             .:=  composant / contexte/
complexe           .:=  collection / primitive GM /contexte /
contexte           .:=  repère / observation / illumination / simple / complexe
collection         .:=  { composant }
composant          .:=  name id primitive GM
classes_d'attributes .:= vue_propriétés vue_méthodes vue_contraintes
vue_propriétés     .:=  classes_configuration_spatiale classes_causales_probabilistes
                        classes_aspect réaliste_dimension_et_mesure classes_primitives_géométriques_et_déformations
vue_méthodes       .:=  classes_génération_de_modèle_paramétrique / classes_d'édition_et_modification /
                        classes_inférence_causale / classes_satisfaction_de_contraintes
vue_contraintes    .:=  classes_contrantes_géometrques / classes_intervalles_numériques /
                        classes_intervalles_spatiales / classes_contrantes taille
```

Figure 2. Définition BNF du concept de scène selon OBR.

utilisateur (IGU), le logiciel de visualisation–rendu d'images POV-Ray (RePov), et les bases de données: connaissances (BaCo), données graphiques (DoGra), données images (DoIm) et le shell Unix.

L'architecture de l'environnement est conçue en vue de fournir des moyens nécessaires pour l'application de l'approche de modélisation déclarative dans la conception de scènes, en utilisant des stratégies et des techniques de raisonnement qualitatif et des heuristiques exploratoires causales-probabilistes. Le but du raisonnement concerne l'obtention d'une réponse aux questions sur la composition (par ex. Quelles sont les composants d'une scène ?), la géométrie (p. ex. Quelle est la scène avec les proportions données ?), la configuration spatiale (par ex. Quelles sont les configurations possibles dans l'espace disponible ?) et l'aspect réaliste des scènes (par ex. Quelles sont les scènes avec l'apparence donnée ?).

Le couplage des outils est fait à base d'actions d'échange de données. Celui-ci offre les avantages d'une gestion et manipulation modulaire des données par l'intermédiaire du langage en utilisant des stratégies d'optimisation basées sur les connaissances causal probabilistes. Dans les paragraphes suivants nous présentons le support théoriques du contexte conceptuel et l'architecture du langage SOML. Ils sont spécifiés en termes de modèles conceptuels et logiques, exprimés par classes à objets et leurs associations. Le formalisme de représentation choisi est l'utilisation des diagrammes conceptuels UML.

Couche connaissances et CRM

A ce niveau de la hiérarchie, les fonctionnalités sont définies conformément au modèle conceptuel de résolution CRM (Conceptual Resolution Model). Ce modèle est construit sur la base de concepts issus du formalisme objet orienté multi-vues OBR(Objects based representation) de modélisation de connaissances du domaine (Caponi 1995) et du modèle de raisonnement basé à l'application des stratégies de résolution causal probabiliste CPM.

Selon CRM, une scène est vue comme un modèle cognitif du monde de la conception. Les propriétés et les contraintes des scènes mais aussi les méthodes de leur traitement, sont définies comme des objets cognitifs–unités de connaissance. Ils constituent les connaissances du domaine selon les différents points de vue sur la scène. Les connaissances sont structurées à base du formalisme OBR définie par la méthodologie SHIRKA-TROPES (Caponi 1995; Schmeltzer 1995). Le CRM permet l'énumération et l'attachement de propriétés ou des contraintes aux objets des scènes et l'application de stratégies de résolution définies par CPM.

Le multi point de vue, classes, instances et OBR. L'univers de la scène est représenté par une collection d'objets cognitifs structurés hiérarchiquement par des classes, des instances et des multi-points de vue. Ces objets sont autonomes, caractérisés par une représentation de ses connaissances orienté objets. Une famille générique d'objets se modélise par un concept, partitionné en différents points de vue (Figures 1-3). Les objets communiquent entre eux et l'environnement par le

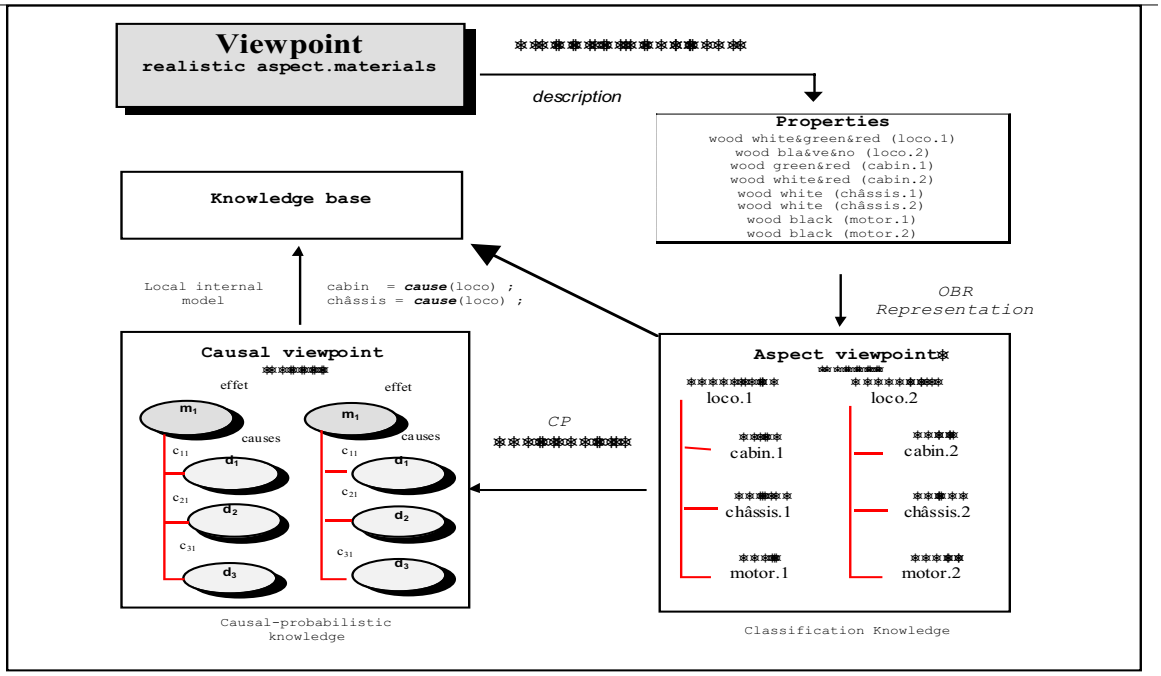


Figure 3. Example of creation of a local base of hybrid knowledge (view "materials", scene "locomotive").

with the causes.

For a Sy system there are various possible schemes of semantic associations, allowing one to obtain different types of solutions. For example, if one is interested in representing the taxonomy of the scenes based on their different properties, it is appropriate to associate to the manifestations M the concepts of properties (or property attributes), and to the causes D, the concepts of scenes possessing these properties. All this expresses the fact of having one class of properties due to the existence of scenes characterized by these properties.

Given a set M+ representing the set of properties that are present (described or required), the solution to a design problem consists in finding the scene having these properties. We define the set $M^- = M - M^+ = \neg M^+$, to be the set of properties that are not present, that is, all the manifestations present are observable. The mode of abductive reasoning used here allows us to look for possible causes (e.g. scenes) explaining or justifying the effects (eg. the properties) observed. In other words, we look for explanations in terms of possible existing scene solutions for the set of properties required for the design.

For each "view" of the scenes one creates a local knowledge base, represented by a bipartite graph with two types of nodes: the nodes di representing the direct causes of the effects m_i and the nodes of the effects m_j (Figures 3, 5). To the edges joining the two types of nodes are associated probabilistic weights c_{ij} (strength of causality) reflecting domain knowledge in terms of normalized numeric values in the interval [0,1]. For example they can be the frequency of occurrence of the cognitive concept of a scene property in an event (scene present). The basis is completed by the addition of a priori probabilities of the existence of the causes d_i in a domain.

Probabilistic knowledge about a domain is used to calculate a criterion named "measure of likelihood", which makes it possible to evaluate the plausibility of the different hypotheses about potential solutions in terms of coverings. As mentioned above, it is possible to have other semantic associations between the two types of nodes of Figure 3. The nodes d_i for the direct causes, instead of representing scenes, could represent prop-

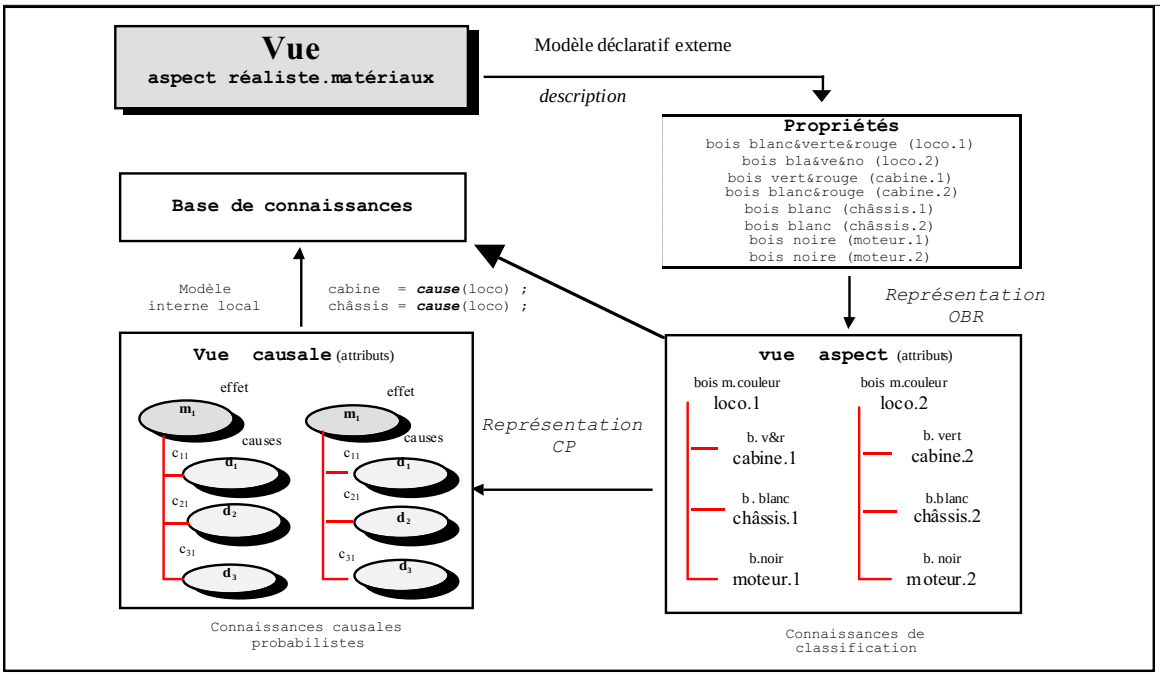


*Figure 3. Exemple de création d'une base locale des connaissances hybrides (Vue "*Matériaux*", scène "*locomotive*").*

biais d'interfaces d'échange de données dans un format particulier unifié pour tous les types de scènes conçues.

En termes de données, les classes des propriétés (les contraintes inclues) peuvent être assimilées aux champs d'une structure. Les domaines des valeurs des champs sont définis dans l'espace des types de primitives (entiers, réels, chaînes de caractères ou de types structurés). Les objets sont vus de différents «points de vue». Chaque point de vue détermine un ensemble de champs visibles par celui-ci. Par exemple, la scène peut être vue en tant qu'une forme géométrique complexe du point vue Classes géométriques de formes, un objet situé à un endroit précis dans l'espace selon le point de vue Configuration spatiale, une structure composée d'autres objets de scènes spécifiées dans Classes de composition, etc. Cette approche multi-vues présente les avantages d'un regroupement local des classes de propriétés pertinentes par rapport à une vision particulière du concepteur, conforme à sa compétence. Les classes des instances des concepts dans un «*vue*» sont structurées hiérarchiquement.

Cette démarche propose l'avantage d'un traitement local et distribué, permettant une conception parallèle et coopérative de scènes complexes. Les classes sont hiérarchiques (par relation de spécialisation) et peuvent être «instanciées» aux différents niveaux de la hiérarchie. Cette caractéristique permet la simplification de l'étape de modélisation dans une session de conception, au prix d'une perte de détails (Plemenos 1996b). Un possible but d'inférence peut être la classification de la connaissance, c'est-à-dire la recherche de la classe d'un objet auquel on connaît l'instance.

Le point de vue Méthodes par exemple regroupe les différents méthodes utilisées pour la construction des composants d'un domaine (Figure 4). Ces méthodes constituent des hiérarchies de tâches de résolution comme étant des connaissances méthodologiques.

Connaissances causales probabilistes et CPM. Le modèle CPM (Causal Probabiliste Model) décrit des relations causales existantes entre les unités de connaissances, dénotées par les termes «causes»

	Classes of generating tasks	Input	Output
1	task .motor	engine, chimney	motor
2	task .chassis	front suspension rear suspension	chassis
3	task .cabin	roof, win.l &win.r.	
4	task .engine		
5	task .chimney		
6	task .roof		
7	task .left window. &right window		
8	task .left wheel		
9	task .right wheel		
10.	task .front suspension		
11.	task .rear suspension		
12.	task .door		
13.	task .link		
14.	task .cylinder		
15.	cube		

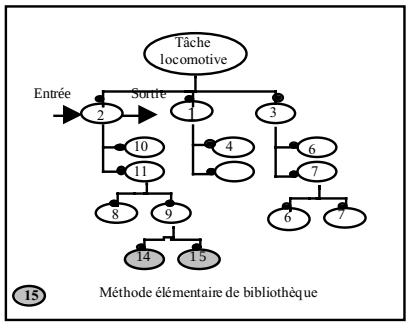


Figure 4. Example of tasks / methods of geometric parametric model generation for the "locomotive" scene.

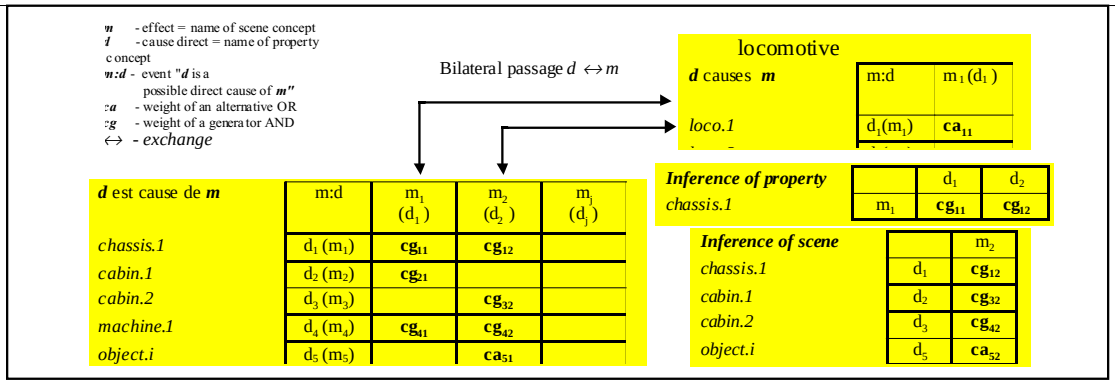


Figure 5. Logical representatiion of the causal dependencies by means of matrices.

erties, and the effects m_j, instead of representing properties, could represent scenes (Figure 5). The symbols cg_{ij} express the probabilistic weights in the heirarchical causal graph, where edges are joined by means of a logical operator ET. In the case where this connection is expressed by means of the logical "or" operator, we use the symbol cg_{ij}. For the example in Figure 5 each row of the causal dependency matrix allows one to infer a property, and each column of the matrix allows one to infer a scene.

We are also interested in knowing all the causes belonging to the alternative possible coverings which can lead to a given state of the system Sy, and furthermore to propose an arrangement of the coverings, considered as hypothetical solutions, in terms of their level of likelihood. The coverings can be classified as being redundant or non-redundant. The concept of a covering is useful for finding subsets of instances of property attributes that will lead to solutions in terms of scenes. We are interested in non-redundant coverings satisfying the specified optimization criteria.

We define a non-redundant covering M^+ as being the set of causes $d_i \in D$, linked with the manifestations $m_j \in M^+$, and not containing any subsets that are also coverings of M^+. The hypothetical coverings are calculated by exploring the graph of the CPM model. For this one needs the notion of "alternative generators" and the algebraic operation on sets: division, remainder, augmented remainder, that are derived from the classical set operations union, intersection and difference. (Peng 1990). By applying the operations union (joining) and projection to the local CPM graphs, one obtains the global basis for the scene concept. Figures 6-7show possible inference examples in terms of coverings (Peng 1990) for the causal knowledge base of Figure 5. The two coverings loco.1

	Classes des tâches de génération	Entrée	Sortie
1	tâche .moteur	engin, cheminée	moteur
2	tâche .châssis	suspens. avant, suspens. derrière	châssis
3	tâche .cabine	toit, fen.g&fen.d.	
4	tâche .engin		
5	tâche .cheminée		
6	tâche .toit		
7	tâche .fenêtre gauche. &fenêtre droite		
8	tâche .roue gauche		
9	tâche .roue droite		
10.	tâche .suspens. avant		
11.	tâche .suspens. derrière		
12.	tâche .portière		
13.	tâche .liaison		
14.	tâche .cylindre		
15.	cube		

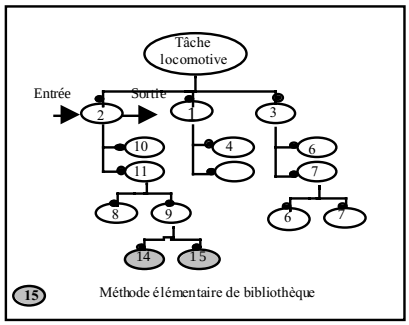


Figure 4. Exemple de tâches / méthodes de génération de modèle géométrique paramétrique de la scène "locomotive."

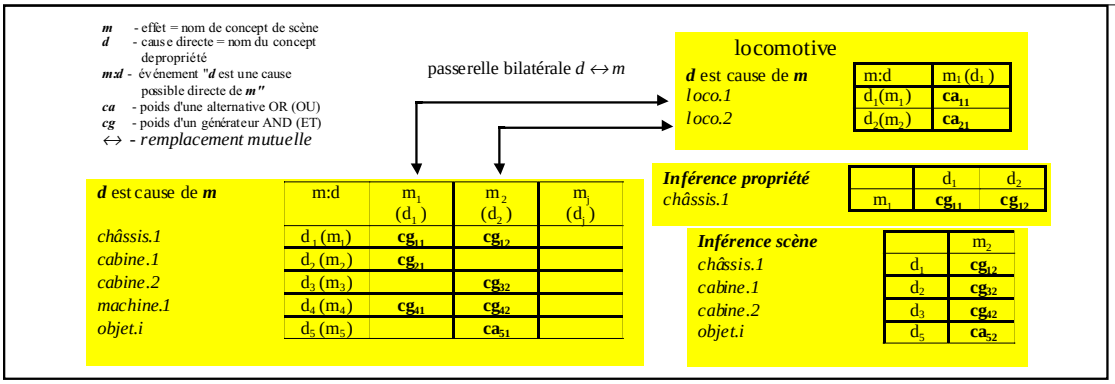


Figure 5. Types of possible inferences from a causal knowledge base. [Représentation logique des dépendances de causalité par matrice.]

	propriétés annoncés	*propriétés inférés (solutions couvertures)*
1	$\{d_i\} \subset D$	$\{m_j\} \in M, d_i : m_j \quad \{d_{ij}\} \in \{ d_i : m_j \mid d_i \}, \{d_i\} \in D$
2	$\{m_j^+\} \subset M$	$\{d_{ij}\} \in \{d_i : m_j \mid di\}, \{d_i\} \in D$
3	$\{d_i\}, \{m_j^+\}, \{d_i\} \subset D, \{m_j^+\} \subset M$	$\{d_{ij}\} \in \{d_i : m_j \mid d_i\}, \{d_i\} \in D, \{m_j\} \in M$
4	$\{m_j^+\}, \{d_i\}, \{d_i\} \subset D, \{m_j^+\} \subset M$	$\{d_{ij}\} \in \{d_i : m_j \mid d_i\}, \{d_i\} \in D, \{m_j\} \in M$

Figure 6. Types of possible inferences from a causal knowledge base. [Types d'inférences possibles à partir d'une base de connaissances causales.]

	propriétés annoncés	*propriétés inférés (solutions couvertures)* « , » = ET logique
1	{châssis.1} ⊂ {loco.1, loco.2}	{chassis.1, cabine.1, cabine.2, machine.1, objet$_i$} {châssis.1 : loco.1 , chaâssis.1: loco.2} loco.1 = { chassi.1, cabine.1, machine.1} loco.2 = { chassi.1, cabine.2, machine.1}
2	{loco.1} ⊂ {loco.1, loco.2} {loco.2} ⊂ {loco.1, loco.2}	{chassi.1, cabine.1, machine.1} {chassi.1, cabine.2, machine.1} OU{ objet.i}
3	{châssis.1 }, {loco.1},	loco.1 = {chassi.1, cabine.1, machine.1}
4	{loco.2}, {châssis.1 }	loco.2 = {chassi.1, cabine.2,machine.1} OU{obej.i}

Figure 7. Examples of inference of covering solutions. [Exemples d'inférence de soluti couvertures.]

and loco.2 offer alternative explanations for the existence of the "locomotive" scene. This proves useful in the situation described in R1 above. If information about the designer's intended scene is limited and insufficient, he can formulate a request for help, by describing his goals (scene concepts, represented by tasks and their methods (Figure 4). The request leads to a prediction of one or more scene concepts and their properties. These lead to the generation of hypothetical solutions nine sets of instances of classes of attributes in causal relation with the concepts stated as goals).

modeling layer

To this level of the hierarchy, the functionality is defined consistently to the functional model of SOML language. This model is build on the base of functionality concepts kept on this work phase, represented in terms of class in the diagram of the Figure 8.

Functional concepts and geometric internal model. The SOML language permits the creation of geometrical scene models which can be modified in later stages of the design process. The necessary mechanisms for these activities are grouped into the classes Edition, Generation- Integration, Spatial Configuration and Interrogation. These classes contain mechanisms for a) instantiation and integration of the primitive components in order to obtain a final solution (Generation- Integration); b) modification of the solution shapes, the photo-realistic aspect and the context (Edition); c) positioning and reconfiguration of scenes and their components in 3D space (Spatial configuration); d) obtaining answers to the questions about the states of scenes (Interrogation).

According to the conceptual model, presented briefly in the previous paragraphs the prototype of final solution of a scene is a set of instances of the data structure fields of the properties class. This solution is generated from the geometric parametric model of the scene in a process of constraint resolution and qualitative reasoning on the spatial configuration. The geometric parametric model of scene is a list of SOML commands stored in the graphical data base (DoGra) of the environment. The command structure is constituted from lexical elements of the geometrical canonical primitives of POV format. This presents the advantage of one facilitated coupling of geometrical model instances with the rendering software POV-Ray.

The list of commands contains implicitly the CSG tree of geometrical and appearance primitives, of not instanced variable of their parameters, a set of geometrical transformations and a boolean operations permitting one to make a solution explicit by means of a graphical photo-realistic shape. The code of the model is "compiled" by the interpreter of the environment from the methods of the scenes tasks.

Spatial configuration, qualitative resolution, and positioning scenario. A scene is constituted either of individual components located in 3D space, or of components to be used in a future assembly. In each case one is faced with the problem of spatial configuration, constituting a task of positioning. In a declarative way the user expresses his goal , usually in a qualitative manner, using the syntax and vocabulary of positioning provided by natural language (Donikian 1992). The statement is converted into coordinates in the local reference system of the scene. We now examine how to simulate the problem of declarative control of the 3D objects' spatial configuration.

The method used is based on the formalism of space discretization by means of trees of "octree" form and Allen spatial intervals (Mäntylä 1988; Foley 1995). The target space is considered to be like a box with dimensions width, height and depth, measured with respect to the frame of reference chosen for that specific context. The volume of this space is recursively subdivided into octants, up to the moment where primitive boxes will be completely bounded. The bounding boxes O of positioned objects have the dimensions O.width, O.eight and O.depth, defined with respect to the frame of reference.

generation layer

Once the parametric model of the scene is created, it must be instantiated by the values of the numeric constraints.

The system of coordinates. The system of coordinates (frame of reference of the scene) used is

et «effets» (manifestations). Les connaissances causales sont «plus profondes» que les connaissances heuristiques utilisées par le concepteur pour résoudre un problème.

Le système *Sy* du à une description possède des états définis par le *n-uple* d'attributs binaires (a_1,.., a_i, ., a_n) (Dubois 1995). Si a_i = 1, on accepte qu'il y a des manifestations m_i présentes, et si a_i = 0, les manifestations m_i sont absentes. S'il n'y a pas de manifestations présentes, nous disons que le système *Sy* se trouve dans son état normal (initial) et qu'il peut se décrire par des *n-uples* (*0, …, 0, …, 0*).

Soit *M* la dénotation d'un ensemble de *n* manifestations possibles {m_1, .., m_j, .., m_n}. Soit *D* un ensemble de *causes* directes des manifestations possibles {d_1,.., d_i, ., d_k}. Une *cause* peut être présente ou absente. Pour chaque d_i est associé un ensemble *effets*(d_i) de manifestations qui sont une conséquence de la présence des causes d_i. La relation *R* sur $D \times M$ est définie comme (d_i, m_j) Œ *R* ´ m_j Œ *effets* (d_i) et celle-ci associe des manifestations et des causes.

Pour un système *Sy* sont possibles différents schèmes d'associations sémantiques, permettant d'obtenir différents types de solutions. Par exemple si l'on s'intéresse de représenter les taxonomies des scènes par rapport aux différentes propriétés qu'elles possèdent, il est convenable d'associer aux manifestations *M* les concepts de propriétés (encore attributs de propriétés) et aux causes D les concepts de scènes possédant ces propriétés. Tous ceci exprime le fait d'en avoir une classe de propriétés due à l'existence des scènes, caractérisés par ces propriétés.

Etant donnée un ensemble M^+ de propriétés présentes (décrites ou demandées par une requête), le problème de résolution résulte d'en trouver la (es) scène(s) qu'elle puisse(ent) posséder les propriétés de l'ensemble M^+. Nous supposons que l'ensemble $M^- = M - M^+ = \neg(M^+)$ est l'ensemble de propriétés absentes, c'est-à-dire toutes les manifestations présentes sont observables. Le mode de raisonnement abductif que nous utilisons permet la recherche de causes (par ex. scènes) possibles justifiant (ou expliquant) les effets (par ex. les propriétés) observées. Autrement dit, nous cherchons des explications en termes de solutions de scènes existantes possibles pour un ensemble des propriétés demandés pendant une conception.

Pour chaque «*vue*» des scènes, on crée une base locale, représentée par un graphe bipartie possédant deux types de noeuds: les noeuds d_i des causes directes de l'effet m_j et les noeuds des effets m_j (Figure 3, Figure 5). Aux liens entre ces deux types de noeuds sont associés des poids probabilistes c_{ij}, (force de causalité) reflétant les connaissances du domaine en termes des valeurs numériques normalisées dans l'intervalle [*0,1*]. Par exemple elles peuvent être les fréquences d'occurrences de chaque concept cognitif de propriété de scène dans un événement «scène présente». La base est complétée aussi par des probabilités *à priori* de l'existences des causes d_i dans un domaine.

Les connaissances probabilistes du domaine servent à calculer un critère, appelé «mesure de vraisemblance» permettant l'évaluation de la crédibilité des différentes hypothèses de solutions potentielles en termes de couvertures. Comme il a été déjà noté plus haut il est possible d'en avoir d'autre type d'associations sémantiques sur les deux types des noeuds de la Figure 3. Les noeuds d_i des causes directes au lieu de représenter des scènes, ils peuvent représenter des propriétés, et des *effets* m_j au lieu de représenter des propriétés, ils peuvent représenter des scènes (voir la Figure 5). Les symbole cg_{ij} expriment les poids probabilistes dans le graphe causal hiérarchique, où les branches sont liées avec un opérateur *ET* logique. Dans le cas où cette liaison est exprimée par l'opération *OU* logique nous utilisons le symbole ca_{ij}. Pour l'exemple de la Figure 5, chaque ligne de la matrice de dépendances causales permet d'inférer une propriété et chaque colonne de la même matrice permet d'en inférer une scène.

Nous nous intéressons aussi de connaître toutes les causes appartenant aux couvertures concourantes (alternatives) possibles qui peuvent conduire vers un état donné du système *Sy*, et en plus de proposer un rangement des couvertures, considérées comme des hypothèses des solutions possibles selon leur niveau de vraisemblance. Les

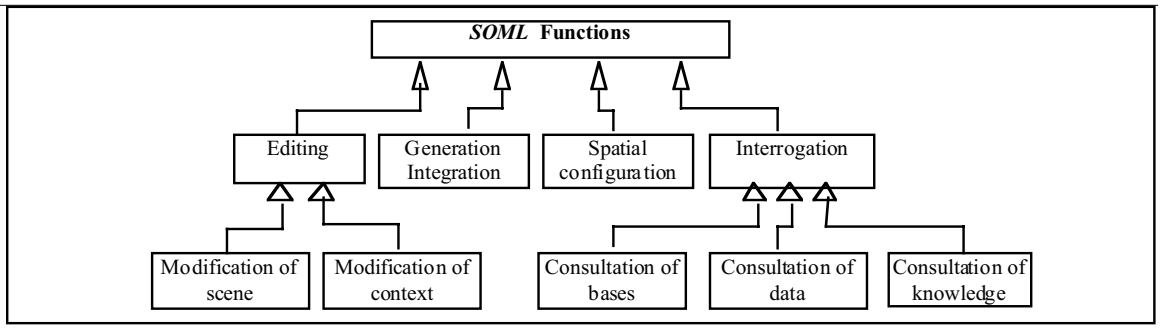


Figure 8. Diagram of functional classes for SOML.

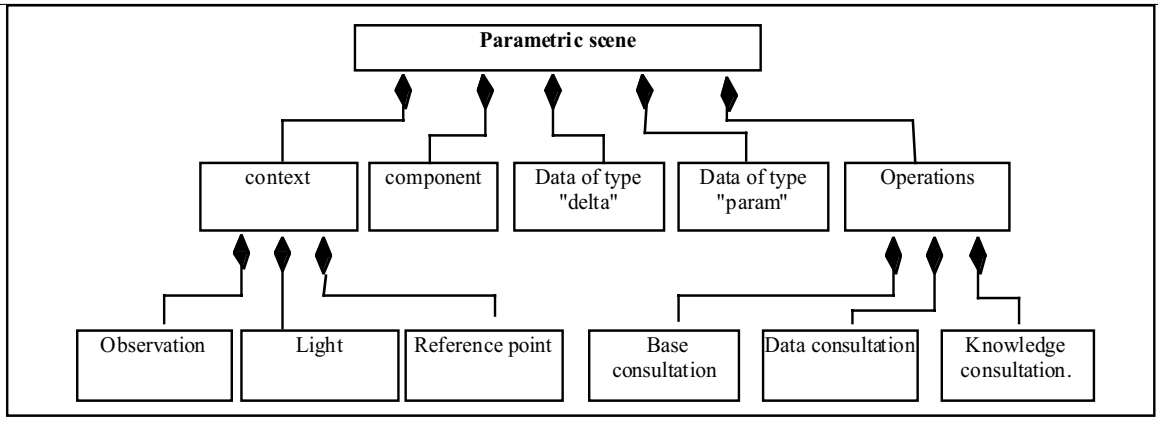


Figure 9. Class diagram for the internal parametric model.

of type" left-handed": the X axis is oriented to the right of the screen before the observer, the axis Y is oriented upward and the Z axis is oriented away from the observer. The dimensions on the three axes are expressed by some real number or by the percentage with regard to the 3D of the bounding volume where there exists the equivalence right ↔ "+ X," left ↔ " -X," high ↔ "+ Y,"low ↔ " -Y," before ↔ " Z" and behind ↔ "+ Z." The localization in the reference point O(X, Y, Z) ↔ O(right, high, behind) is specified by vectors of real of one or three arguments. For the position, the orientation, the dimensions and their modifications one uses geometric parametric transformations, in our case according to expression 3-1, 3-2, 3-3 (Foley1995).

Parametric model of the primitives. The generic parametric primitives are represented by some characteristic points according to the visualization format of POV. An example of such (canonical) primitives is the sphere algebraic class represented by the points {<CENTER>, RADIUS}.

In the following paragraph we present the basic theoretical concepts and the principal techniques used for the definition and the experimentation of the data storage definition model, explored by SOML during a design session

Data Model. In terms of data a concrete solution consists in a list of instances of the parameters of the geometric primitives of the model, satisfying the imposed constraints. These data have a particular format visible by means of the rendering software POV and they reside in the graphical data base. The data of one scene solution are defined in the object-oriented conceptualization formalism according to the diagram presented in the Figure 11. This model defines the concepts and the classes

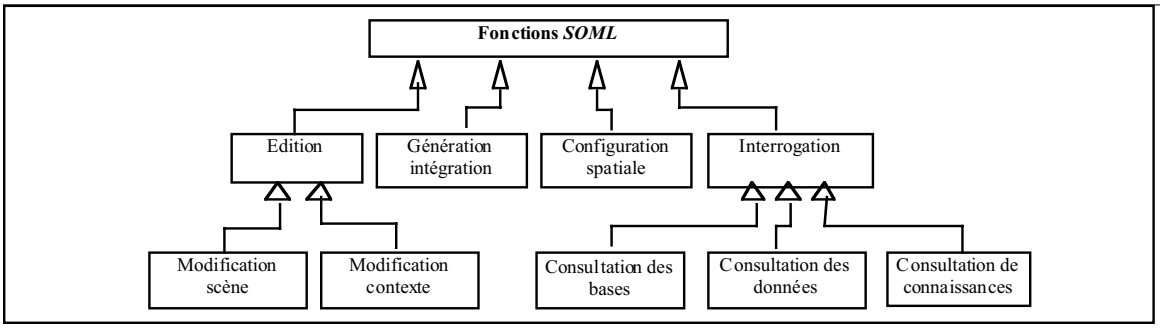


Figure 8. Diagramme des classes de fonctis mantenues par SOML.

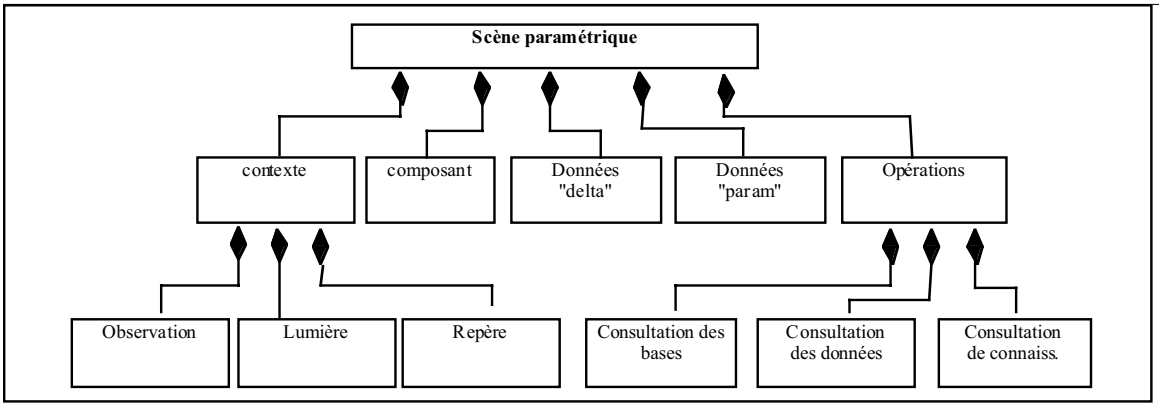


Figure 9. Diagramme des classes du modèle paramétrique interne.

op rations de transformation

$$\begin{bmatrix} X^* \\ Y^* \\ Z^* \end{bmatrix} = f\left(\begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix}, \begin{bmatrix} X^*_0 + deltaX^*_0 \\ Y^*_0 + deltaY^*_0 \\ Z^*_0 + deltaZ^*_0 \end{bmatrix}, \begin{bmatrix} deltaX^* \\ deltaY^* \\ deltaZ^* \end{bmatrix}\right) \quad (3\text{-}1)$$

* ∈ {*s-scale, t-translate, r-rotate, R-rayon, D-distance, A-sommet min, B-sommet max, V-vecteur* de 1 arg.}
f = fonction de *modification*

$\text{position}_{\text{scène}}(X,Y,Z) = f(univers(X_0, Y_0, Z_0), init(X_0 + \text{deltaX}_0, Y_0 + \text{delta}\,Y_0, Z_0 + \text{deltaZ}_0), modif(\text{deltaX}_0, \text{deltaY}_0, \text{deltaZ}_0))$ (3-2)

X, Y, Z	*coordonnées de positionnement de la scène*
X_0, Y_0, Z_0	*coordonnées initiales du repère de la scène englobante*
$deltaX_0, deltaY_0, deltaZ_0$	*incréments de modification des coordonnées de la scène englobante*
deltaX, deltaY, deltaZ	*incréments de modification de coordonnées de la scène*

$modif(\text{deltaX}_0,\text{deltaY}_0,\text{deltaZ}_0) = (X_0,Y_0,Z_0)^T+(\text{deltaX*}_0,\text{deltaY*}_0,\text{deltaZ*}_0)^T+ (\text{deltaX*}, \text{deltaY*}, \text{deltaZ*})^T$ (3-3)

Figure 10. Calculation of the parameters of an instance of the object "Instance_param$_{scene}$." [Calcule des paramètres géométriques d'une instance de l'objet "Instance_param$_{scene}$".]

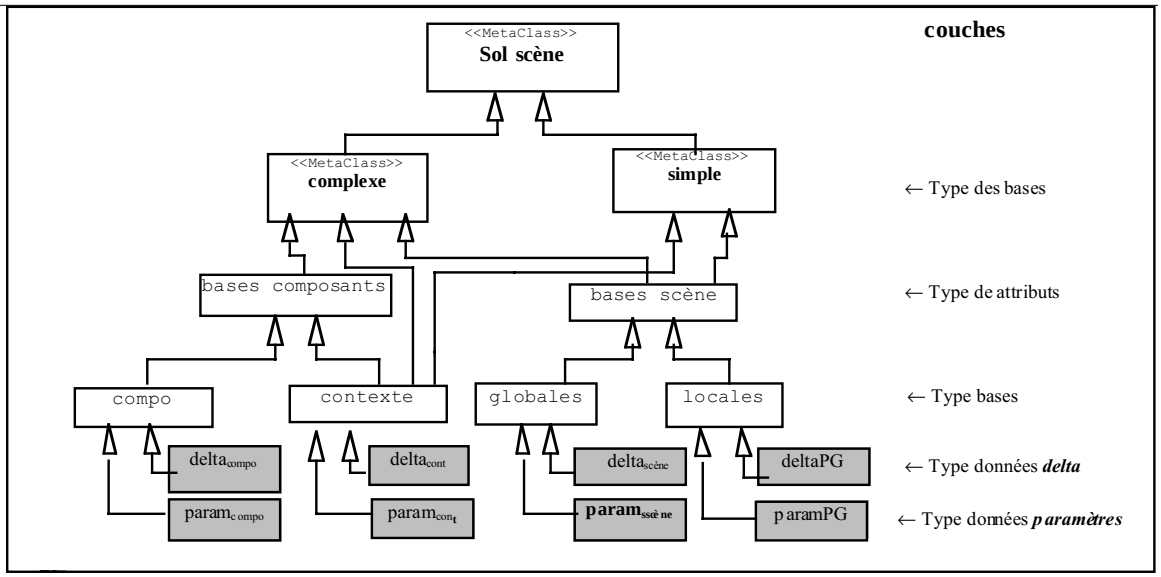


Figure 11. Conceptual diagram of the data storage model of a scene. [Schéma conceptuel du modèle de données de stockage d'une scène.

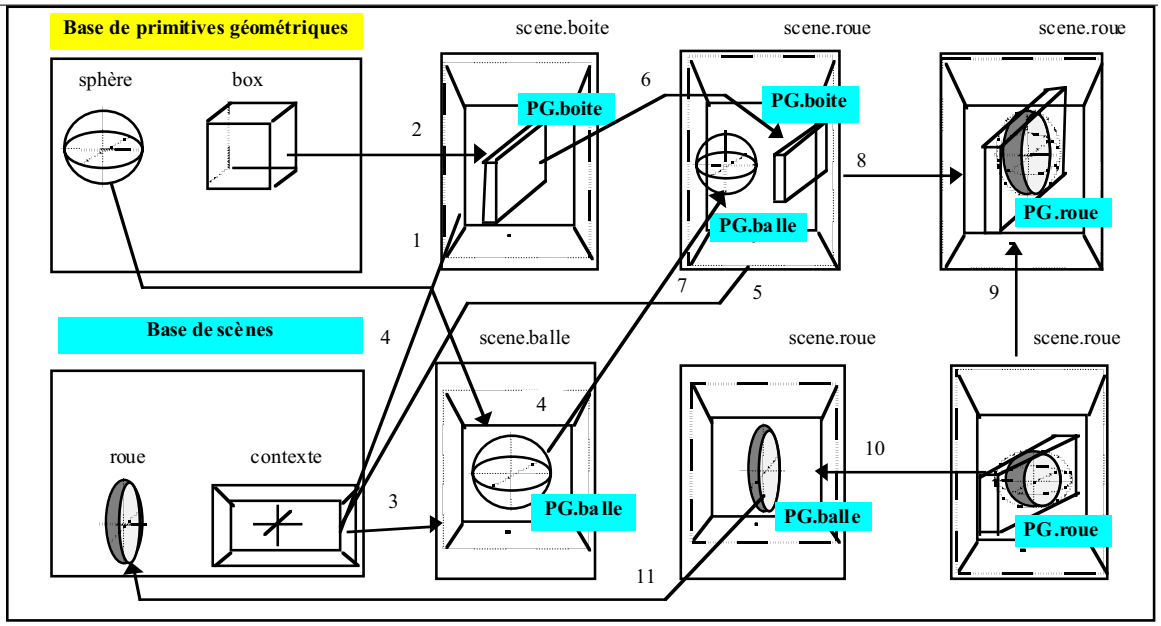


Figure 12. Sample scenario of actions involved in generating the "wheel" scene. [Exemple de scénario d'actions de génération de la scène "roue".]

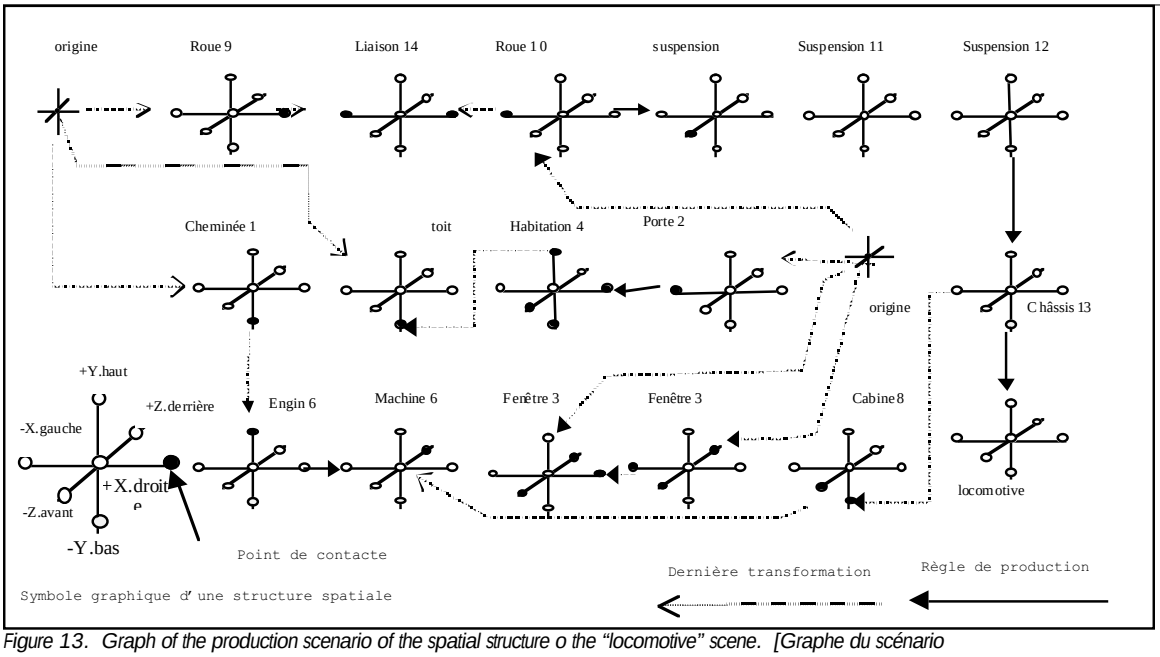


Figure 13. Graph of the production scenario of the spatial structure o the "locomotive" scene. [Graphe du scénario de production de la structure spatiale de la scène "locomotive."]

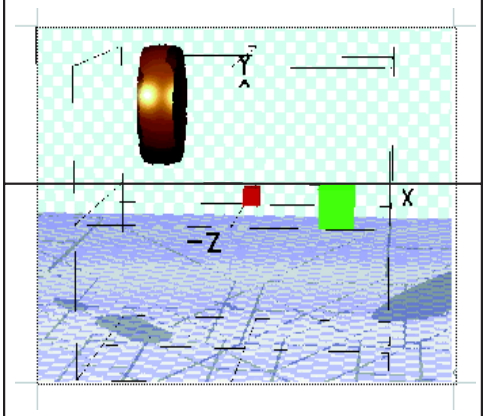
Steps	Steps
after executing the instructions: generate_simple(ball) \ place(scene_ball, ball, wheel, small_ball)	**after executing the instructions:** generate_simple(ball) \ place(scene_ball, ball, wheel, small_ball) generate_simple(box) place(scene_box, box, wheel, flat_box)
compilation of parametric model: #include "delta_wheel.inc" #include "param_wheel.inc" #include "delta_context_wheel.inc" #include "param_context_wheel.inc" #include "context_wheel.inc" #include "small_ball.inc" object { small_ball }	**compilation of parametric model:** #include "delta_wheel.inc" #include "param_wheel.inc" #include "delta_context_wheel.inc" #include "param_context_wheel.inc" #include "context_wheel.inc" #include "small_ball.inc" object { small_ball } #include "flat_box.inc" object {flat_box}
	**after executing the instructions:** generate_simple(ball) \ place(scene_ball, ball, wheel, small_ball) generate_simple(box) \ place(scene_box, box, wheel, flat_box)\ generate_complex(small_ball, flat_box, intersection
	compilation of parametric model: #include "base_wheel.inc" #include "context_wheel.inc" #include "small_ball.inc" #include "flat_box.inc" #declare PG1wheel = intersection { object { small_ball } object { flat_stone } } #declare wheel = object { PG1wheel scale < XsPG1wheel , YsPG1wheel , ZsPG1wheel > rotate < XrPG1wheel ,YrPG1wheel , ZrPG1wheel > translate< XtPG1wheel,YtPG1wheel,ZtPG1wheel > } object { wheel transform TrS_wheel }

Figure 14. Steps in creating the geometric parametric model of the "wheel" scene. [Les étapes de création du modèle géométrique paramétrique de la scène "roue".]

of objects necessary for the generation of one concrete solution.

A scene is considered to be of “complex” type if it must manage the data for its constituent components; otherwise it is of type “simple” and contains only a single geometric structure. The method of “simple” scene generation doesn’t use boolean operations in the final stage of the design. A scene of “complex” type, on the other hand, is obtained by means of methods containing instructions with regular boolean operations.

In the example of the Figure 14, the scene obtained by the method containing the last instruction “generate.simple” (ball, sphere,…) is of type "simple". In the same example the scene "wheel" is of type "complex". The two types of scenes refer to the two classes of objects representing the basic concepts in the data bases: "scenes database" and "components database". The class "scenes database" specializes to "global database" and "local database."

language structure

The language is structured on the basis of a multiple layer hierarchy of different functionality classes supported by the context of the design-assistance environment. Every class belongs to a model and to a particular view of the scene defined by the conceptual model (Figure 1). The classes of one layer contain methods constituting the type of the sub-language corresponding to the linguistic schemes at this level. One distinguishes three levels of functionality represented by the layers: *knowledge* (acquisition, representation), *modeling* (geometric, photometric), and *generation* (constraint resolution, instantiation). To every layer corresponds a sub-language: macro-instructions of description of the methodical and causal probabilistic knowledge; instructions for geometric and photometric modeling; and commands, for instantiation of primitives and of operations (geometric transformations, boolean operations). Such an approach appears to be consistent with the declarative methodology’s characteristic feature of allowing a certain inaccuracy and incompleteness in the scene description. By subdividing the means of description into classes one insures the possibility of conceptualizing the scenes from partial descriptions of properties belonging to the known views.

macro-instructions and tasks

The role of the macro-instructions of the upper layer is to provide a way of modeling the tasks of the design process as being methodological knowledge. We distinguish macro-instructions: (a) making possible, according to the concepts defined in the MCR model, a description of the tasks, made up of incremental methods of construction of scene solutions; and (b) serving to describe the properties and their causal dependencies according to the CPM model. The notion of a task permits a compact and structured representation of a scene design problem and information about its solution in terms of methods and scenarios of assembly / disassembly. This strategy is represented by a graph of specialization and decomposition into sub-problems of reduced complexity. In the lowest layer the decomposition of the problem becomes a single task, and its solution strategy consists in calling one method from the domain knowledge base.

The simplified syntax is as follows:

task	::= *macro* [*name_task*] {*knowledge*}[*status*]
macro	::= {*vocabulary*[argo] }
vocabulary	::= generate | modify |...
knowledge	::= *descending* | *task* | *cause* | *effect* | *relation* | *inference*
name_task	::= [*alpha*]
alpha	::= *letter*|-|0|...|9.
letter	::= a|b|...|z|A|...|Z
status	::= complex | final | entry | exit

The strategy for executing the tasks is a causal chaining between the levels in the hierarchy. The inference engine explores the space of the sub tasks based on the causality knowledge, valued to different levels of detail. The role of the reasoning is to identify the sequence of tasks to execute starting from data entry in a general and incomplete case. The set macro-instructions of causal links descriptions is intended to give causal probabilistic knowledge containing the identifiers of the nodes of the causes and the effects and the values of the causality strengths between those nodes.

instructions layer, taxonomy of methods

The execution of the methods of the tasks of

couvertures peuvent être classifiées comme des redondantes ou irredondantes. Le concept de couverture sert à trouver des sous-ensembles d'instances d'attributs de propriétés qui vont évoquer des instances de solutions en terme de scènes. Nous nous intéressons de couvertures irredondantes satisfaisant les critères d'optimisation choisis.

Définition : Une couverture irredondante de M^+ est l'ensemble de causes $d_i \in D$, liées avec les manifestations $m_i \in M^+$, mais qui ne contient pas des sous ensembles étant aussi des couvertures de M^+.

Les couvertures - hypothèses sont calculées par exploration du graphe du modèle CPM. Pour ceci sont utilisées les notions de générateurs d'alternatives et des opérations algébriques ensemblistes reste, division et reste augmenté dérivés des opérations classiques union, intersection et différence (pour des détails voir Peng 1990). En appliquant des opérations de jointure et de projection sur les graphes CPM locaux s'obtient la base de connaissances globale du concept de la scène. Les Figure 6-7 montrent des exemples d'inférence possibles en termes de couvertures, introduit par (Peng 1990) pour la base de connaissances causalles de la Figure 5. Les deux couvertures loco.1 et loco.2 sont en compétition pour expliquer l'existence de la scène "locomotive." Ceci s'avère utile dans le cas formulé antérieurement par la règle *R1*. Si les informations connues sur la scène visée par le concepteur sont réduites et insuffisantes, il peut formuler une requête d'assistance, exprimée par une description des buts (concepts de scènes, représentés par tâches et ses méthodes (Figure 4). La requête provoque la prédiction d'un ou plusieurs concepts de scènes et des propriétés, existantes. Ces derniers déclenchent la génération d´hypothèses de solutions (ensembles d'instances des classes d'attributs étant en relations de causalité avec les concepts énoncés comme des buts).

Couche modélisation

A ce niveau de la hiérarchie, les fonctionnalités sont définies conformément au modèle fonctionnel du langage SOML. Ce modèle est construit à base de concepts de fonctionnalité retenus à ce stade du travail, représentés en termes de classes dans la diagramme de la Figure 8.

Concepts fonctionnels et modèle géométrique paramétrique interne. Le langage SOML permet la création de modèles cognitifs et géométriques de scènes, spécifiés dans les différentes «vues» qui peuvent être postérieurement modifiées dans les étapes successives de l'évolution de la conception. Les mécanismes nécessaires pour l'exécution de ces activités sont regroupes dans les classes Edition, Génération-Intégration, Configuration spatiale et Interrogation. Ces classes regroupent des mécanismes pour (a) l'instanciation et l'intégration des primitives des composantes afin d'obtenir une solution finale (Génération-Intégration); (b) la modification des solutions de formes, l'aspect photo et le contexte(Edition); (c) l'effectuer le positionnement et la reconfiguration des scènes et ses composants dans l'espace 3D (Configuration spatiale); (d) l'obtention de réponses aux questions concernant les états de scènes (Interrogation).

Selon le modèle conceptuel, présenté brièvement dans les paragraphes précédents, le prototype de solution–finale d'une scène est un ensemble d'instances des champs de la structure de données des classes des ses propriétés. Cette solution est générée à partir du modèle géométrique paramétrique de la scène dans un processus de résolution des contraintes et de raisonnement qualitatif sur la configuration spatiale.

Le modèle géométrique paramétrique d'une scène (Figure 9) est une liste de commandes *SOML* stockées dans la base de données graphiques (DoGra) de l'environnement. La structure des commandes est constituée des éléments lexicaux des primitives géométriques canoniques du format POV. Ceci présente l'avantage d'un couplage facilité des instances du modèle géométrique avec le logiciel du rendu d'images réalistes *POV-ray*.

La liste de commandes contient implicitement l'arbre CSG, constitué de primitives géométriques et d'apparence, de variables non instanciées de leurs paramètres, d'un ensemble de transformations géométriques et d'opérations booléennes permettant d'expliciter une solution sous forme d'image photo réaliste. Le code du modèle est

the macro-instructions layer decomposes into chains of instructions associated to the different activities necessary to construct the geometric parametric model of the designed scene. The instructions are interpreted by the ISOML interpreter belonging to the environment. The principal types of instructions make up the sub-languages: Generation and Integration of Scenes, Configuration of Scenes, Editing (modification of scenes, modification of context- initial bounding scene), and Interrogation (accessing the graphical and knowledge data bases).

Generation and Integration of Scenes. The instructions of generation and integration of the primitive are the tools for constructing scenes from canonical geometric and photometric primitives, by means of CSG construction operations and constraint resolution. Reserved words are as follows:

Generation and integration of scenes: generate_simple, generate_complex, generate_context, generate_box.

Scene editing: place, displace, erase, constraint, reposition.

Modification of scenes: modifie taille, modifie orientation, initialise.

Modification of context: *modify observation, modify light, modify texture*.

Information gathering: *create scene, attach geometry, attach base of primitives, attach information, attach Spatial configuration,render image, visualize scene*.

command statements

The level of the commands depends on the type of geometric modeler used and specifies the format of the primitives expected by this modeler. The software validating this stage of the approach, SOML is based on a solids modeler that supports standardized expanded CSG operations, constructed from public domain utilities available through the Internet. The primitive commands keywords and syntax of commands is:

primitive *commands - reserved keywords:*
sphere, box, cylinder, cone, plane, etc.
scene := *language_directives camera*
{camera_items} object_statements
atmosphere_statements
global_statements{global_items}

command ::= *[#declare] [name alpha] [=*object*]*
[{PG1] name alpha {operation}
operation ::= transform "<" vector ">"
vector ::= {axis index [PG1 name]}
axis ::= *X* | *Y* | *Z*
index ::= s | r | t
transform ::= *scale* | *rotate* | *translate [object {] name*
transform TrS_[name]}

execution scenarios

The design scripts determine the possible sequence of tasks, predicted in the conceptual model of the scene. An example of such a script is shown in the Figure 12. The labels above the arrows reflect the order of execution of the different stages. In order to facilitate the representation of the scenario of one spatial configuration, we propose the diagrammatic formalism based on the use of graphic symbols, shown in Figure 13. The advantage of this approach consists in the possibility of creating a language and a grammar of structures for the production of other spatial configurations. It is possible to have more than one scenario for the construction of the target scene. These are all part of the methodological knowledge explained above. A scenario is made up of a sequence of targeted methods of executing the tasks of the scene. It can be compiled either automatically from the global model of scene knowledge, or assembled and modified with the SOML tools of the Instructions layer.

experimental results

To illustrate the above-mentioned ideas, in the Figures 15-18, we present one example of the design of the scene of the wooden toy "train." The descriptions of the known properties (Figure 15) will be converted into a set of SOML instructions (Figure 16), constituting the methods of construction (Figure 17).

conclusion

The creation of software to assist with scenes design is a difficult task due to the complexity of the underlying problems. Indeed the requirements of functionality necessitate a computer system regrouping several objects and methods of treatment. The potential advantages resulting from the use of

«compilé» par l'interpréteur de l'environnement à partir des méthodes des tâches de la scène.

Configuration spatiale, résolution qualitative et scénario de positionnement. Une scène est constitués soit de composants individuels localisés dans l'espace *3D,* soit de composants d'un assemblage futur. Dans les deux cas on se heurte avec le problème de configuration spatiale, constituant une tache de positionnement. Dans un cadre déclaratif l'utilisateur exprime son but surtout qualitativement par un vocabulaire appartenant aux syntagmes locatifs du langage naturel (Donikian 1992). L'énoncé est converti sous forme de coordonnées dans le repère local de la scène. Le sujet concerne la simulation du problème du contrôle déclaratif de l'aménagement spatial d'objets 3D.

La méthode qualitative quantitative utilisée est basée sur le formalisme de discrétisation de l'espace par intervalles spatiaux d'Allen sous forme d'arbre octree (Allen 1983; Mäntylä 1988; Foley 1995). L'espace cible est considéré comme étant une boite avec des dimensions largeur, hauteur et profondeur définies par rapport au repère du contexte choisi. Le volume de cet espace est partitionné recursivement en octants jusqu'au le moment où les boites des primitives à positionner pourront être englobées complètement. Les boites englobantes des objets à positionner O ont des dimensions O.largeur, O.hauteur et O.profondeur definies par rapport du centre du repère.

Couche génération

Une fois créé, le modèle paramétrique de la scène doit être «instancié» par les valeurs des données des contraintes numériques.

Le système de coordonnées. Le système de coordonnées (repère de la scène) utilisé est de type «main gauche»: l'axe *X* est orientée vers la droite de l'écran devant l'observateur, l'axe *Y* est orienté vers le haut et l'axe *Z* est orienté dans le sens opposé de l'observateur. Les mesures sur les trois axes sont exprimées par des réels ou par le pourcentage par rapport aux 3 dimensions du volume englobant où il existe l'équivalence *droite* ↔ « *+X* », *gauche* ↔ «*-X*», haut ↔ «+Y» , *bas* ↔ «*-Y* «, *avant* ↔ «*-Z*» et *arrière* ↔ «*+Z*».

Les allocations dans le repère $O(X,Y,Z) \leftrightarrow O(droite, haut, arrière)$ sont spécifiées par des vecteurs de réels de 1 ou 3 arguments. Pour la position, l'orientation, les dimensions et leurs modifications on utilise des transformations géométriques, parametrées dans notre cas selon expression 3-1, 3-2, 3-3 (Foley 1995).

Modèle paramétrique des primitives. Les primitives génériques paramétriques, sont représentés par des points caractéristiques selon le format de visualisation POV. Un exemple de telle primitive canonique est la classe algébrique *sphère* représentée par les points caractéristiques { <CENTER > , RADIUS }.

Dans le paragraphe suivant nous présentons les concepts théoriques de base et les techniques principales utilisées pour la définition et l'expérimentation du modèle de définition des données de stockage, explorés par SOML au cours d'une session de conception

Modèle des données. En termes de donnés une solution concrète se traduit dans l'obtention d'une liste d'instances de paramètres des primitives du modèle géométrique satisfaisant les contraintes imposées. Ces données possèdent un format particulier visible par le logiciel de rendu Pov est elles résident dans la base de données graphiques. Les données d'une solution de scène sont définies dans le *formalisme conception objet orienté* selon le schéma conceptuel présenté dans la Figure 11. Ce modèle définit les concepts et les classes d'objets nécessaires pour la génération d'une solution concrète.

Une scène est considérée comme un type «complexe» si elle doit gérer les données des composants qu'elle englobe par contenance sinon elle est du type «simple» et elle ne contient qu'une seule structure géométrique. Les méthodes de génération de scènes «simples» n'utilisent pas d'opérations booléennes à l'étape finale de la conception. La scène de type «complexe» au contraire, s'obtient par l'intermédiaire des méthodes contenant des instructions ayant des opérations booléennes régulières.

Dans l'exemple de la Figure 4-5, la scène

Type of Viewpoint	"Natural" Description	Deduced Relations
Description of viewpoint #1 *"Composition"*	The scene train is made up of 5 components The components locomotive are loco.1, loco.2, loco.3 The components wagon ar wagon.1, wagon.2	(components, 5) (compo.1, loco.1) (compo.5, wagon.2)
Description of viewpoint #2 *"Spatial Configuration"*	Component loco.1 is on the left. Component loco.2 is in the middle. Component loco.3 is on the right Component wagon.1 is to the right of component loco.1 Component wagon.2 is to the left of component loco.3	(loco.1, on_left) (loco.2, in_middle) (loco.3, on_right) (wagon.1, on_right, loco.1) (wagon.2, on_left, loco.3)
Other viewpoints . . .		. . . . .

Figure 15. Part of the declarative description of the "train" scene.

1	2	3	4	5	6
place\ scene_loco loco\ train loco.1 position\ above\ distance\ 6 units 0	move \ train loco.1\ higher 0 \ distance\ 7 units 0	place\ scene_loco loco\ train loco.2\ position\ to_right_of\ distance\ 10units 0	change_size\ train loco.2\ smaller 0\ distance 0.5 units 0	move \ train loco.2\ more_to_right \ distance 7 units\ higher 0\ distance\ 7 units 0	place\ scene_loco loco\ train loco.3\ positionl\ to_left_of\ distance\ 20 inites 0
7	8	9	10	11	12
change_size\trai- n loco.3\smaller 0\distance\0.5 unites 0	move \ train loco.2\ move_to_right 0\ distance\ 7 units\ higher 0\ distance\	rote \ train loco.2\ horizontal_ to_left 0\ angle\ 60 units\	rote \ train loco.3\ horizontal_ to_left 0\ angle\ 30 units\	generate_compl- ex train 0\ union 0 0\ 2 units\ white_wood\ 0 0 3\ loco1 loco2 loco3	modify_ observation\ train c_loc\ more_to_front 0\ distance60\ plus_a_right 0\ distance\ 100 units 0

Figure 16. Instructions for the construction method for the class "wooden trains:" generate.train.

#	#	macro-instruction (method)	(scene)	(scene)	(scene)	exit.3 (scene)	Comments
1.		generate.wheel	ball	box	-	wheel	generation of wheel
2.		generate.hub	ball.2	box.2		hub	generation of wheel2
3.		generate.link	side.left	side.right	axe	link	generation of link
4.		generate.suspensi	wheel	wheel	link	suspension	generation of suspension
5.		generate.chassis	suspension	suspension	-	chassis	generation of chassis
6.		generate.chassis.2	suspension.2	suspension.2	-	chassis.2	generation of chassis.2
7.		generate.roof	box	ball	-	roof	generation of roof
8.		generate.caboose	box	window.g	window.d	caboose	generation of caboose
9.		generate.caboose.2	box	window.g	window.d	caboose.2	generation of caboose.2
10.		generate.cabin	caboose	roof	-	cabin	generation of cabin
11.		genrate.cabin.2	caboose.2	roof	-	cabin.2	generation of cabin.2
12.		generate.chimney	hyperboloid	side.1	side.2	chimney	generation of chimney
13.		generate.engine	side 1	cylinder	side.2	engine	generation of engine
14.		generate.machine	engine	chimney	-	machine	generation of machine
15.		generate.loco	machine	cabin	chassis	loco	generation of locomotive
16.		generate.wagon	cabion.2	chassis.2		wagpm	generation of wagon
17.		generate.train	loco.1	wago.1	wago.2	train	generation of train

Figure 17. Incremental method of hierarchical generation of the "train" scene.

the language consists in its functionality enabling it to be of assistance during all stages of the design process of complex scenes. Beyond these initial encouraging results, there remains to improve the degree of application of these cognitive techniques of representation and knowledge exploitation; to integrate Bayesian learning; to perfect the terminology(i.e. to create a base of terminological knowledge, BCT, that will facilitate the use of the SOML language for different types of applications); to develop a linguistic scheme; and to create a convivial work environment.

To this end, as already emphasized, the SOML language makes possible the creation of models of scenes, including such features as geometry, topological configurations, realistic aspects, methods of construction, etc., that can be modified during subsequent stages of the design process. Indeed, a designer's most important task is to make precise the often vague and ambiguous objectives stated by the client or by specialists from other fields, to fulfill these objectives, and thereby identify the fundamental purpose of the system (Rivero 1977).

By analyzing how a designer works, we can envisage an application of the SOML language as design-assistance tool in Architecture. Indeed, early in the design process, when communicating with a client, an architect might use a sketch, which is a link between the design concept and the real world. By analogy with this evolving sketch, we can say that a language which takes into account the descriptive declaration of the initial concept, and the possibility of reasoning in a qualitative and quantitative manner about the properties of the model, is related to the architectural design process. We see in this language a tool based not only on the production of synthetic and realistic images of 3D scenes (i.e. drawings), but also on their development. In this process, the form is not the result of an analysis of solutions, nor of a simple list of functions, but rather the result of cybernetic activity. The functions (activities) can be considered as a process of verification of the architectural idea; the mental pre-conceptualization of a problem throughout the evolving representation of the form (Portoghesi 1981).

The use of computers in design leads to re-

obtenue par la méthode contenant comme dernière instruction genere.simple(balle, sphère, ….) est du type «simple». Sur la même Figure la scène «roue» est du type «complexe». Les deux types de scènes se réfèrent aux deux classes d'objets représentant les concepts de base des données de stockage: «bases scènes» et «bases composants». La classe «bases scène» est spécialisée en «bases globales» et «bases locales».

Structure

La structure du langage est définie à la base de l'architecture constituée de la hiérarchie de classes correspondantes aux différentes fonctionnalités supportées dans le contexte de l'environnement d'aide à la conception. Chaque classe appartient à une couche et à une vue particulière sur la scène définie par le modèle conceptuel (Figure 1). Les classes d'une couche possèdent de méthodes constituant le type du sous–langage correspondant au schémas linguistiques associés à ce niveau.

On distingue 3 niveaux de fonctionnalités représentés par les couches Connaissances (acquisition et représentation), Modélisation (géométrique photométrique) et Génération (résolution de contraintes et instantiation). A chaque couche correspond un sous-langage: Macroinstructions de description des connaissances méthodologiques et causal probabilistes, Instructions de modélisation géometrique et photometrique, Commandes d'instantiation de primitives et d'operations (transformations géométriques et opérations booleennes).

Une telle approche nous semble très cohérente avec la caractéristique de la méthodologie déclarative de permettre une certaine inexactitude et incomplétude de la description. En partageant les moyens de description par classes nous assurons la possibilité de concevoir des scènes à partir de description partielle des propriétés appartenant aux vues connues.

Niveau de macroinstructions et tâches

Le rôle des *macroinstructions* du niveau supérieur est de fournir un moyen pour la modélisation des tâches de la conception comme étant des connaissances méthodologiques. On

Type de Vue	Description "naturelle"	Relations déduites
Description de point de vue N° 1 "Composition"	La scène "train" est composée de 5 composants Les composants "locomotive" sont "loco.1", "loco.2", "loco.3" Les composants "wagon" sont wagon.1, wagon.2	(composants, 5) (compo.1, loco.1) (compo.5, wagon.2)
Description de point de vue N° 2 "Configuration spatiale"	Le composant "loco.1" est à gauche. Le composant "loco.2" est au milieu. Le composant "loco.3" est à droite Le composant "wagon.1" est à droite du composant "loco.1" Le composant "wagon.2" est à gauche du composant "loco.3"	(loco.1, à_gauche) (loco.2, àu_milieu) (loco.3, à_droite) (wagon.1,à_droite, loco.1) (wagon.2,à_gauche, loco.3)
d'autres "vues"		. . . . .

Figure 15. Une partie de la description déclarative de la scène "train."

1	2	3	4	5	6
place\ scene_loco loco\ train loco.1 position_spatiale\ au_dessus_de\ distance\ 6 inites 0	deplace \ train loco.1\ plus_haut 0 \ distance\ 7 unites 0	place\ scene_loco loco\ train loco.2\ position_spatiale\ au_droite_de\ distance\ 10 inites 0	change_taille\ train loco.2\ plus_petite 0\ distance 0.5 unites 0	deplace \ train loco.2\ plus_a_droite \ distance 7 unites\ plus_haut 0\ distance\ 7 unites 0	place\ scene_loco loco\ train loco.3\ position_spatiale\ a_gauche_de\ distance\ 20 inites 0
7	8	9	10	11	12
change_taille\ train loco.3\ plus_petite 0\ distance\ 0.5 unites 0	deplace \ train loco.2\ plus_a_droite 0\ distance\ 7 unites\ plus_haut 0\ distance\ 7 unites 0	rote \ train loco.2\ horizontale_ a_gauche 0\ angle\ 60 unites\	rote \ train loco.3\ horizontal_ to_left 0\ angle\ 30 units\	genere_complex train 0\ union 0 0\ 2 unites\ bois_blanc\ 0 0 3\ loco1 loco2 loco3	modifie_ observation\ train c_loc\ plus_avant 0\ distance60\ plus_a_droite 0\ distance\ 100 unites 0

Figure 16. Le jeu d'instructions de la méthode de construction d'une classe de trains de boi "generer.train."

#	#	macro-instruction (method)	entrée.1 (scène)	entrée.2 (scène)	entrée.3 (scene)	sortie.3 (scéne)	Commetaire
1.		generate.roue	balle	boite	-	roue	generation de wheel
2.		generate.hublot	balle.2	boite.2		hublot	generation de roue2
3.		generate.liaison	parois.gauch	parois.droite	axe	liaison	generation of liaison
4.		generate.suspensi	roue	roue	liaison	suspension	generation de suspension
5.		generate.chassis	suspension	suspension	-	châssis	generation de châssis
6.		generate.chassis.2	suspension.2	suspension.2	-	châssis.2	generation de châssis.2
7.		generate.toit	boite	bale	-	toit	generation de toit
8.		generate.habitat	boite	fenêtre.g	fenêtre.d	habitation	generation de habitation
9.		generate.habitat.2	boite	fenêtre.g	fenêtre.d	habitation.2	generation de habitation.2
10.		generate.cabine	habitation	toit	-	cabine	generation de cabine
11.		genrate.cabine.2	habitation.2	toit	-	cabine.2	generation de cabine.2
12.		generate.cheminee	hyperboloide	parois.1	parois.2	cheminee	generation de cheminee
13.		generate.engin	paroi 1	cylindre	parois.2	engin	generation de engin
14.		generate.machine	engine	cheminé	-	machine	generation de machine
15.		generate.loco	machine	cabin	châssis	loco	generation de locomotive
16.		generate.wagon	cabine.2	châssis.2		wagon	generation de wagon
17.		generate.train	loco.1	vago.1	vago.2	tren	generatin de train

Figure 17. Méthode incrémentale de générati hiérarchique de la scène "train".

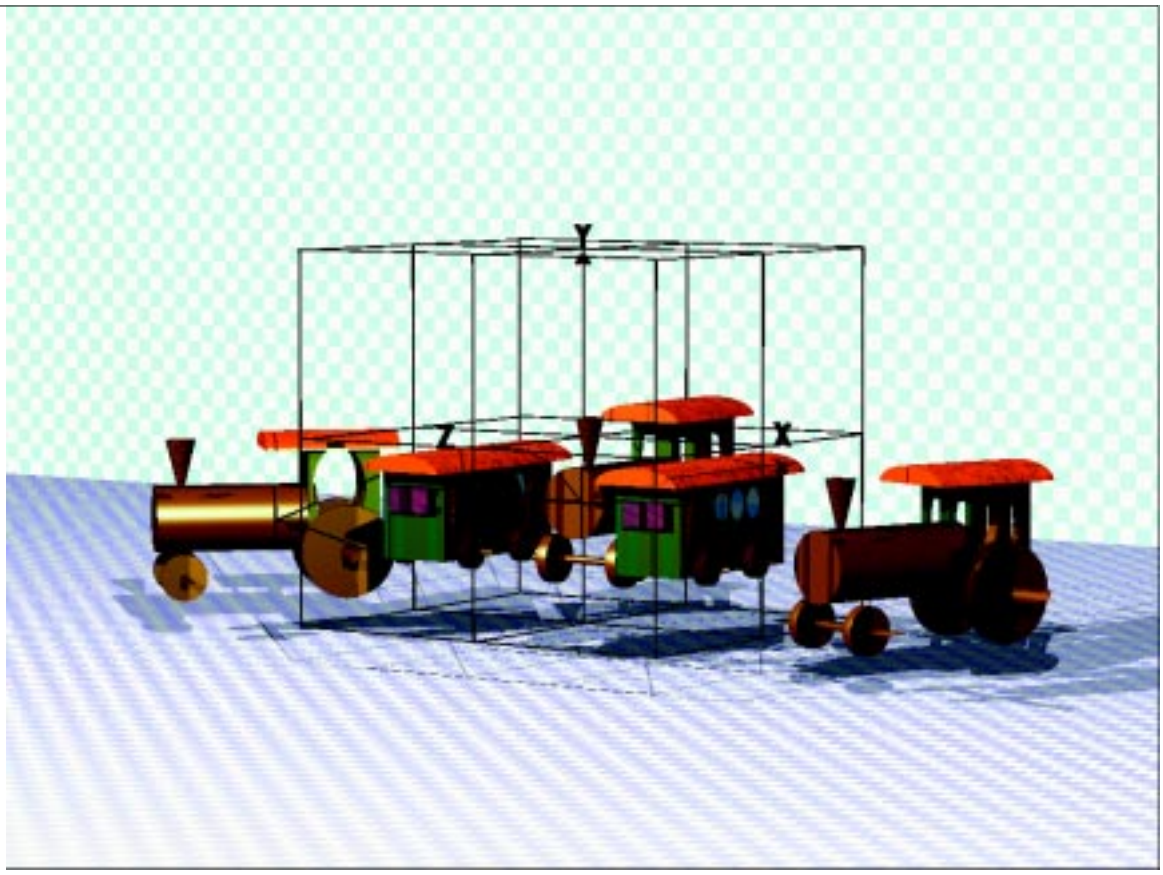


Figure 18. Image of the final "train" scene resulting from a design scenario. [Image de la scéne final "train" obtenue par une séquence d'activitiés exécutés selon un scénario de conception.]

search on systems that manipulate not only the components of a scene, but also the design process itself. These systems simplify the stages of analysis, synthesis, and design evaluation, right from the beginning of the design phase. A distinctive feature of this research is the form of reasoning used; based on inferences (deductions, abductions, or CSP) and the use of a symbolic approach instead of numeric calculations. This makes the design process more explicit and transparent. Other research in this direction, notably that of GRCAO and GE-ODE (research groups in computer-aided design), proposes using modeling methods based on the declarative description of the construction process and its translation into computer input, making possible a complete parameterization of the model (SGDLsoft 1995). In this way we can produce simulations and an interaction between the vision of the architect and the shape of the object. This re-

distingue des macroinstructions permettant selon (a) les concepts définis dans le modèle MCR, la description des tâches qui sont constituées des méthodes de construction incrementale des composants des scènes; (b) le modèle CPM, la description des propriétés et leurs dépendances de causalité.

Une tâche permet une représentation compacte et structurée d'un problème de conception des scènes et les connaissances de sa résolution en termes de méthodes et scénarios d'assemblage / désassemblage. Elle spécifie les données d'entrée, le résultat et la stratégie de la résolution. Cette stratégie est représentée sous forme d'un graphe de spécialisation et décomposition en sous problèmes de complexité réduite. Au niveau le plus bas la décomposition du problème devient une seule tâche, et sa stratégie de résolution consiste dans l'appel d'une méthode de la base des méthodes du domaine.

Le syntaxe simplifié est donné ci-dessous :

tache ::= *macro* [*nom_tache*]{*connaissance*}[*statut*]
macro ::= {*vocabulaire*[*argo*] }
vocabulaire ::= *génère* | *modifie* | ...
connaissance ::= *descendante* | *tache* | *cause* | *effet* | *relation* | *inférer*
nom_tache ::= [*alpha*]
alpha ::= *letter*|-|0|...|9.
letter ::= a|b|...|z|A|...|Z
statut ::= *complexe* | *final* | *entrée* | *sortie*

La stratégie de résolution des tâches est un chaînage causal entre les niveaux de la hiérarchie. Le moteur d'inférence explore l'espace des sous-tâches à base des connaissances de causalité, estimées à différents niveaux de détail entre les objets des différentes vues. Le rôle du raisonnement est d'identifier la séquence de tâches à exécuter à partir des donnée d'entrée dans un cas général et incomplet.

Le jeux des macroinstructions de descriptions de liens de causalité est destinées à construire une base de connaissances causales-probabilistes contenant les identificateurs des noeuds des causes et des effets et les valeurs des *forces de causalité* entre ces noeuds.

Niveau d'instructions et taxonomie des méthodes

L'exécution des méthodes des tâches du niveau *macroinstructions* se décompose en chaînes d'instructions associées aux différentes activités nécessaires pour construire le modèle géométrique paramétrique de la scène sous conception. Les instructions sont interprétées par l'interpréteur *IntpSOML* faisant partie de l'environnement.

Les types principaux des instructions constituent les sous - langages de *Génération et intégration de scènes*, de *Configuration de scènes*, *d'Edition* (*Modification de scènes*, *Modification de contexte* - scène initiale englobante), d'interrogation (*Consultation* des bases graphiques, de données et de connaissances). Les instructions de génération et d'intégration des primitives sont les outils de construction de scènes à partir de primitives géométriques et photométriques canoniques et opérations de construction *CSG* et assemblage par résolution de contraintes. Génération et intégration de scènes - mots réservés: génère simple, génère complexe, génère contexte, génére boite.

Langage d'édition de scènes

Edition *de scénes - mots réservés: place, déplace, efface, rote contrainte, repositionne;*
Modification *de scénes - mots réservés: modifie taille, modifie orientation, initialise; Modification de contexte - mots réservés: modifie observation, modifie lumière, modifie texture;*
Prise de *connaissances - mots réservés: crée scène, affiche géométrie, affiche base primitives, affiche connaissances, affiche config. Spatiale, rendu image, visualise scéne*

Commandes. Le niveau des commandes dépend du type du modeleur géométrique utilisé et spécifie le format des primitives prévues pour ce modeleur. La validation logiciel à ce niveau de l'approche, SOML est basée sur un modeleur de solides supportant d'opérations CSG régularisées étendues, construites à base d'utilitaires disponibles dans le domaine public d'Internet.

Les classes de primitives géométriques paramétriques et syntaxe d'une commande:

commandes de primitives - nots réservès sphere, box,

search complements the use of SOML in the declarative design of scenes.

Knowledge and mastery of these tools enable the architect to shed new light on design problems, improve classical design methods, and speed up the development of new methods. Through computer use, architecture will be able to evolve towards the goal of establishing a link between classical and modern methods (De Paoli 1996, Miaoulis 1996).

references

Allen, J. F., 1983. "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM*, vol. 26, n° 11, 832-843, 1983

Argués, Didier, and Stéphane Aubert, 1994. "Modélisation de scenes tridimensionnelles pour l'animation en synthèse d'images: notion d'arbres ECSG et computation," *AFIG'94, Toulouse.*

Brunetti, Gino, Ana Sofia Viera, and Jivka Ovtcharova, (). "Towards a Feature Description Language."

Caponi, Cecile, 1995. "Identification et explicitation des types dans un model de connaissances à objets," *These en Informatique LIFIA / IMAG.*

Charman, Ph., 1995. *Gestion des contraintes géométriques pour l'aide à l'aménagement spatial*. Thèse de doctorat de l'Ecole Nationale des Ponts et Chaussées.

Chen, Hiangpiung, 1995. *Representatin, Evaluation and Edition of Feature-based and Constraint-based Design*. Doctoral Dissertation, Purdue University.

Clay, Sharon, and Jane Wilhelms, 1996. "Put: Language-Based Interactive Manipulation of Objects," *IEEE Computer Graphics and Applications.*

Christian, Colin and Emmanuel Desmontils, Jean-Yves Martin, Jean-Philippe Mounier, 1997. "Modélisation déclarative: Vision de l'utilisateur," Rapport de recherche , Ecole des Mines de Nantes 97-8-INFO, IRIN-152.

De Paoli, Giovanni, and Alain Marty, 1996. "Architecture, simulation informatique et création industrielle," rapport de recherche, Ministère de l'enseignement supérieur, Québec.

Dif, Jean, 1996. "Images 3D," *Sybex.*

Desmontils, 1995. "Les modeleur déclaratifs," RR IRIN, Nantes.

Donikian, Stéphane, 1992. *Une approche déclarative pour la création de scenes tridimensionnelles: applications à la conception architecturale*. Thèse à l'Université de Rennes 1.

Dubois, Didier, and Henri Prade, 1995. "Fuzzy relation equations and causal reasoning," *Fuzzy sets and systems, 95:119-134.*

Foley, et al, 1995. *Introduction à l'Infographie*.

```
                cylindre, cône, plane, etc.
sol_scene   := language_directives camera
               {camera_items} object_statements
               atmosphere_statements
               global_statements{global_items}

commande    ::= [#declare] [name alpha] [=object]
               [{PG1] name alpha {operation}
opération   ::= transform "<" vecteur ">"
vecteur     ::= {axe index [PG1 name]}
axis        ::= X | Y | Z
index       ::= s | r | t
transform   ::= scale | rotate | translate [object {] name
               transform TrS_[name]}
```

scénarios d'exécution

Les scénarios de conception déterminent la séquence possible d'activités des tâches, prévues dans le modèle conceptuel de la scène. Un exemple de tel scénario est montré dans la Figure 12. Les étiquetés au dessus des flèches reflètent l'ordre d'exécution des différentes étapes.

Pour faciliter la représentation du scénario de configuration spatiale, nous proposons le formalisme diagrammatique basé sur l'utilisation de symboles graphiques, montrés sur la Figure 13. L'avantage de cette approche consiste dans la possibilité de créer un langage et une grammaire de structures permettant la production d'autres structures spatiales.

Il est possible d'en avoir plusieurs scénarios pour la construction de la scène ciblée. Ils font partie des connaissances méthodologiques, expliquées ci-dessus. Un scénario est constituée d'une séquence de méthodes ciblées d'exécution des tâches de la scène. Il peut être compilé soit automatiquement à partir du modèle global des connaissances de la scène, soit composé et modifié avec les outils SOML de la couche Instructions.

résultats d'expérimentation

A titre d'illustration des idées mentionnées et présentées ci-dessus, dans les Figure 15-18, nous présentons l'extrait d'un exemple de conception de la scène du jouet de bois «train». La description des propriétés connues (Figure 16) seront converties dans un ensemble d'instructions SOML (Figure 16), constituant les méthodes de construction (Figure 17).

conclusion

La création d'un logiciel d'aide à la conception de scènes est une tâche difficile vis à vis de la complexité des problèmes sous-jacents. En effet les exigences de fonctionnalité font de lui un système informatique regroupant plusieurs objets et méthodes de traitement. Les avantages potentiels provenant de l'utilisation du langage résident dans ses fonctionnalités permettant une assistance à toutes les étapes du processus de conception de scènes complexes.

Outre les résultats initiaux encourageants, il reste encore à améliorer le niveau d'application des techniques cognitives de représentation et exploitation des connaissances, l'intégration d'apprentissage Bayesien, le perfectionnement des vocabulaires (création d'une base de connaissances terminologiques - BCT, qui va faciliter l'utilisation du langage SOML pour différents types d'application envisagés), des schémas linguistiques et la convivialité de l'environnement de travail.

À ce propos, nous avons déjà souligné que le langage SOML permet la création de modèles de scènes (géométriques, configurations topologiques, d'aspect réaliste, de méthodes de construction etc), qui peuvent être postérieurement modifiées dans les étapes successives de l'évolution de la conception. Or, la tâche la plus importante du concepteur est de préciser les objectifs souvent flous et ambigus, définis par le client ou par les spécialistes d'autres domaines, satisfaire à ces objectifs et ainsi identifier le dessein fondamental d'un système (Rivero 1977).

Figure ?-1. Image de la scène final "train" obtenue par une séquence d'activités exécutés selon un scénario de conception.

Nous pouvons entrevoir, en analysant la tâche du concepteur, une utilisation du langage SOML comme outil d'aide à la conception en architecture. En effet, au début de l'idée architecturale, pour communiquer avec un client, l'architecte utilise le croquis qui est un rapport entre un concept et la réalité des choses. C'est par ce croquis qui évolue, que nous pouvons dire qu'un langage qui tient compte de la description déclarative du concept

New York, NY: Addisson-Wesley

Gascuel, 1996. "Fabule : Un environnement de recherche pour l'animation et la simulation," *iMAGIS / IMAG-INRIA.*

Gatenby, Neil, Martin Preston, and W. T. Hewitt, 1993. "The Manchester Scene Description Language," *CGU 88,* Manchester Computing Centre.

Geode, 1997. Web serveur Geode:http://www.emn.fr/dept.info/GEODE/welcome.html.

Gomes, Jonas, and Bruno Coste, Lucia Darsa, Luiz Velho, 1996. "Graphical objects," *Visual Computer,* 12:269-282.

Gomes, and Velho, 1995. "Velho Abstraction paradigms for computer graphics," *Vis Computer* 11:227-239.

Kehrer and Vaterrott, 1996. "Integration Aspects of STEP and their Expression in the CAO Reference Model," *Modellings and Graphics in Science and Technology*. Springer.

Lai, Michael, 1997. "UML La notation de modélisation objet. Application en JAVA," *Inter Edition.*

Lucas, Michel, 1990. "Emmanuel Desmontils, Les modeleurs déclaratifs," *Review de CFAO.*

Mäntilä, Martti, 1988. *Introduction to Solid Modeling*. Computer Science Press.

Martin, P., 1989. "Un système de génération déclarative de polyèdres." rapport N 89-02, LISIT, Université de Nantes, juin 1989.

Miaoulis, Georges and Dimitri Plemenos, 1996. "Le projet MultiCad," Rapport de recherche MSI 96-03.

Oussalah, Chabane , 1997. "Ingénierie Objet," *Interedition.*

Peng, Yun and James A. Reggia, 1990. *Abductive Inference Models for Diagnostic Problem-Solving*. Springer-Verlag.

Plemenos, Dimitri, 1991. *Contribution à l'étude et au développement des techniques de modélisation, génération et visualisation de scenes* . Thèse de doctorat d'état en informatique, Université de Nantes.

Paolo, Portoghesi, 1981. *Au-delà de l'architecture moderne*. Editions de l'équerre, Paris.

Rivero, Victor, 1977. *Une contribution à la conception architecturale assistée par ordinateur*. Thèse de doctorat Institut Polytechnique, Grénoble.

Schmeltzer, 1995. *Modélisation de cartes génomiques. Une formalisation et un algorithme de construction finale sur le raisonnement temporel*, Thèse, L'université J. Fourier, LIFIA / IMAG.

SGDLsoft, 1995. *Manuel de SGDLsoft*. S.G.D.L. Technology S.A., Lyon.

Thalman D., 1988. "Les systèmes de synthèse et d'animation des images de MIRALab," in T. Lieblins and H. Röthlisberger (eds), *Infographie et applications*. Masson.

Trousse, Brigitte, 1989. *Coopération entre systèmes à base de connaissances et outils de CAO : l'environnement multi-agent ANAXAGORE*. Thèse de Doctorat de l'Université de Nice Sophia Antipolis.

Vargas,Cataline, 1995. *Modelisation du processus de conception en Ingénierie des systèmes mécaniques*. Thèse en Informatique, Ecole Normale Supérieure de Cachan.

initial et de la possibilité de raisonner d'une façon qualitative et quantitative sur les propriétés du modèle s'apparente au processus de conception en architecture. Nous voyons dans ce langage un outil basé non seulement sur la production des images synthétiques réalistes de scènes 3D (dessins), mais aussi sur leur formation. Selon cette démarche, la forme n'est pas la résultante d'une analyse-solution, ni d'une simple liste de fonctions, mais elle est le résultat d'une activité cybernétique. Les fonctions (activités) sont considérées comme un procédé de vérification de l'idée architecturale, la préfiguration mentale d'un problème à travers la représentation évolutive de la forme (Portoghesi 1981).

L'adoption de l'ordinateur par les concepteurs pousse la recherche vers des systèmes manipulant à la fois des connaissances sur les composants de la scène et aussi sur le processus de conception lui-même. Ceci facilite les étapes d'analyse, de synthèse et d'évaluation du design dès le début de la phase de conception. Ces nouveaux systèmes se distinguent par le type de raisonnement basé sur des inférences (déductions, abductions, ou CSP) et l'utilisation d'une approche symbolique plutôt que d'un calcul numérique. Ce qui contribue à rendre le processus de design explicite et plus transparent. Plusieurs autres recherches se sont déjà orientées dans cette direction, notamment celles du GRCAO et du GEODE (Groupes de recherches en CAO), qui proposent l'utilisation des méthodes de modélisation basées sur la description déclarative du processus de construction et la traduction informatique de ce processus, ce qui permet de rendre le modèle totalement paramétrable (SGDLsoft 1995). Nous pouvons ainsi produire des simulations et une interaction entre l'oeil de l'architecte et la forme de l'objet. Ces démarches trouvent leur complémentarité avec celle du SOML pour la conception déclarative des scènes.

La connaissance et la maîtrise de ces outils pourront apporter à l'architecte un nouvel éclairage aux problèmes de la conception architecturale, l'aider à améliorer les méthodes classiques et accélérer la mise au point de méthodes nouvelles dans l'élaboration du projet architectural. Par le biais de l'informatique, l'architecture pourra évoluer vers un projet qui consiste à établir un pont entre les méthodes architecturales classiques et les nouvelles (De Paoli 1996; Miaoulis 1996).