l'université nantes angers le mans
PÔLE DE RECHERCHE ET D'ENSEIGNEMENT SUPÉRIEUR

université angers

Thèse de Doctorat

Marouene KEFI

Mémoire présenté en vue de l'obtention du
grade de Docteur de l'Université d'Angers
sous le label de l'Université de Nantes Angers Le Mans

Discipline : Informatique
Spécialité :
Laboratoire : Laboratoire d'Ingénierie des Systèmes Automatisés, 62 Avenue ND du Lac, Angers, France
Laboratoire d'Étude et de Recherche en Informatique d'Angers, 2 Bd Lavoisier, Angers, France

Soutenue le 29 novembre 2012

École doctorale : 503 (STIM)
Thèse n° : 1270

Assistance à l'agencement d'environnements virtuels: apport de la programmation par contraintes

JURY

Rapporteurs :	**M. Marc CAVAZZA**, Professor, Assistant Dean for Research, Teesside University, Middlesbrough, England **M. Samir OTMANE**, Professeur des universités, Université d'Evry
Examinateurs :	**M. Jacques TISSEAU**, Professeur des universités, École Nationale d'Ingénieurs de Brest **M. Luc JAULIN**, Professeur des universités, École Nationale Supérieure de Techniques Avancées
Directeur de thèse :	**M. Paul RICHARD**, Maître de Conférences-HDR, Université d'Angers
Co-encadrant de thèse :	**M. Vincent BARICHARD**, Maître de Conférences, Université d'Angers

Remerciements

Je tiens en premier lieu à remercier les Professeurs Jean-Louis BOIMOND, directeur du Laboratoire LISA et Jin-Kao HAO, directeur du Laboratoire LERIA pour m'avoir accueilli dans leur structure respective et permis de réaliser mes travaux de recherches dans de bonnes conditions.

Je remercie infiniment mon directeur de thèse, Paul RICHARD, pour sa générosité, sa disponibilité ; et pour m'avoir guidé et inspiré tout au long de ce doctorat ; et surtout pour la confiance qu'il m'a accordée.

Je tiens très sincèrement à remercier mon co-encadrant, Vincent BARICHARD, pour ses grandes qualités humaines, son professionnalisme, le partage de ses connaissances et ses conseils avisés tout au long de cette thèse, qui ne serait pas ce qu'elle est aujourd'hui sans son implication.

Mes remerciements vont aussi à Messieurs Marc CAVAZZA, Professeur à l'Université de Teeside (Royaume Uni), et Samir OTMANE, Professeur à l'Université d'Evry, pour avoir accepter d'examiner ce travail en qualité de rapporteurs. Je tiens également à exprimer ma gratitude à Messieurs Jacques TISSEAU, Professeur à l'école Nationale d'Ingénieurs de Brest, et Luc JAULIN, Professeur à l'école Nationale Supérieure de Techniques Avancées, pour avoir accepté de faire partie du jury en qualité d'examinateurs.

Je remercie chaleureusement tous mes collègues du LISA et du LERIA que j'ai côtoyé au cours de ces trois années : Hamza, Sami, Také, Déborah, Damien, Aymeric, Ludovic et Taner.

Cette thèse n'aurait pas vu le jour sans le soutien permanent de ma famille. C'est pour cela que je tiens à remercier chaleureusement mes parents Béchir et Faouzia qui ont toujours cru en moi et qui sont fiers du chemin parcouru. Cette étape de ma vie n'aurait pas été la même sans la présence et l'amour infini de ma bien aimée Rawia qui m'a apporté son enthousiasme et sa joie de vivre. Rawia, je te remercie de tout mon coeur d'être patiente et souriante, ton indéfectible soutien m'a toujours encouragé à aller plus loin. Je dédie tout ce travail à qui est de loin le plus chère à mon coeur, Mohamed-Béchir, mon fils...

1

Introduction

Contexte de travail

L'aménagement ou l'agencement d'espaces restreints est généralement une tâche complexe impliquant des exigences de conception diverses. La diversité des problèmes d'agencement a donné naissance à une multitude de recherches et d'applications industrielles liées à des domaines différents (agencement de locaux, chargement de véhicules, disposition de composants électroniques, etc.). La résolution manuelle, encore très largement utilisée, est basée sur le savoir faire et l'expérience des concepteurs. Cette approche repose sur un procédé coûteux et long qui ne répond, la plupart du temps, pas à toutes les exigences de conception et aux préférences des utilisateurs.

Pour pallier à ces problèmes, la programmation par contraintes (PPC) a été proposée dans ce contexte à travers l'utilisation de solveurs de contraintes. En effet, la diversité et l'efficacité des solveurs développés durant les dernières années, permettent la formulation et la résolution efficace de divers problèmes d'agencement. Ces solveurs utilisent souvent des méthodes basées sur un formalisme appelé CSP (*Constraint Satisfaction Problem*). Bien que ces méthodes exactes ou complètes aient fait leurs preuves dans la résolution des problèmes d'agencement sous contraintes, des méthodes stochastiques ou incomplètes ont été également utilisées [22, 87, 148]. La plupart de ces méthodes stochastiques sont des méthodes évolutionnistes basées sur des algorithmes génétiques.

De nombreuses méthodes de résolution ont été développées en vue de résoudre automatiquement une grande diversité de problèmes d'agencement [16, 144, 54, 131, 34]. Ces méthodes sont souvent adaptées à des cas spécifiques et ne peuvent pas être généralisées. Le principal objectif de ces méthodes est de proposer une ou plusieurs solutions au problème en un temps raisonnable. Cependant, les méthodes de résolution automatique peuvent s'avérer insuffisantes pour résoudre des problèmes où les concepteurs doivent être impliqués dans le processus de résolution, ce qui nécessite une complémentarité entre ceux-ci et l'outil de résolution.

En effet, l'objectif d'un outil de résolution ne doit pas se limiter à la recherche de solutions ; il doit s'étendre à la présentation d'informations permettant d'aider les concepteurs dans leur prise de décision. Un tel outil doit également permettre la prise en compte des préférences des concepteurs, qui sont difficilement formalisables.

La PPC a largement fait ses preuves en pratique mais n'est pas toujours accessible et facile à comprendre pour un non-spécialiste. Il est donc indispensable d'utiliser des outils graphiques permettant non seulement la visualisation des données issues du solveur, mais aussi l'interaction avec ces données. L'interactivité entre le solveur et le concepteur doit également permettre à celui-ci de formuler dynamiquement un problème d'agencement en sélectionnant les composants et les contraintes à appliquer.

C'est dans ce contexte que certaines approches de résolution ont fait appel aux techniques de la réalité virtuelle (RV). En effet, la RV permet une réduction considérable du temps et des coûts relatifs à la conception d'un produit ou d'un système. En ce qui concerne les problèmes d'agencement, la RV peut servir à la réalisation de prototypes virtuels offrant la possibilité de visualiser à l'échelle 1 :1 et de valider les différentes étapes liées au processus de conception.

Cadre de la thèse

Une problématique liée à l'aménagement de véhicules a été soulevée par l'entreprise ***Thales Communications & Security*** située à Cholet. Cette problématique a motivé nos travaux de recherche et a permis d'initier une collaboration entre deux laboratoires de l'Université d'Angers : le ***Laboratoire d'Ingénierie des Systèmes Automatisés*** (LISA) et le ***Laboratoire d'Etude et de Recherche en Informatique d'Angers*** (LERIA).

Problématique et verrous

La résolution d'un problème d'agencement repose sur plusieurs verrous qu'il faut adresser. En effet, des verrous d'ordre méthodologique, technologique et interactionnel co-existent. Les ***verrous méthodologiques*** reposent sur la formulation du problème d'agencement et la traduction des exigences de conception en contraintes. Quant au ***verrou technologique***, il concerne la mise en place d'un cadre de communication fiable et structuré entre un système de résolution de contraintes (solveur) et un système de visualisation et d'interaction 3D. Finalement, les ***verrous interactionnels/communicationnels*** incluent les deux points suivants : (1) comment un utilisateur peut-il intuitivement commander le solveur de contraintes ? et (2) comment le solveur peut-il assister l'utilisateur durant la tâche d'agencement ?

Objectifs et contributions

L'objectif principal de cette thèse est de développer une approche interactive supportant la résolution de problèmes d'agencement et l'assistance des utilisateurs. Cette approche est basée sur un couplage fort entre les outils de formulation et de résolution issus de la PPC, et les techniques de visualisation et d'interaction 3D issues de la RV. Dans ce contexte, la PPC a été utilisée à travers une interface plus intuitive et interactive qu'un éditeur de texte pour entrer un ensemble de contraintes.

Notre approche satisfait simultanément des critères liés à l'efficacité, l'interactivité et la généricité.

-***Efficacité*** : l'approche propose une ou plusieurs solutions en un temps de calcul raisonnable. En cas d'insolubilité, le système renvoi des informations pertinentes pour expliquer la ou les raisons d'échecs.

-***Interactivité*** : l'approche permet aux concepteurs de participer à la formulation du problème, d'explorer et d'interagir avec les solutions proposées. A travers une association entre les données issues du solveur et l'interface graphique, le système développé propose également différentes formes d'assistance permettant de guider les concepteurs vers les choix possibles de placement. Les mécanismes de résolution sont complètement transparents vis à vis du concepteur puisque celui-ci interagit seulement avec l'environnement virtuel simulant l'espace à agencer.

-***Généricité*** : l'approche peut être appliquée à une grande variété de problèmes d'agencement pour lesquels le concepteur choisit lui même les composants et les contraintes, et doit être assisté dans ses choix.

Organisation du mémoire

Ce manuscrit est divisé en plusieurs chapitres répartis de la manière suivante. Le chapitre 2 est consacré à la définition des problèmes de placement et en particulier les problèmes tridimensionnels. Les différents aspects liés à ces problèmes sont également décrits, à savoir la classification, la formulation et les méthodes de résolution. Des exemples de travaux portant sur cette problématique sont donnés.

Le chapitre 3 propose un état de l'art de la PPC, concept issu de l'intelligence artificielle et utilisé pour formuler et résoudre les problèmes d'agencement complexes. Les mécanismes de base de la PPC sont détaillés dans ce chapitre, à savoir le filtrage, la propagation et la recherche de solutions. Une attention particulière est portée à la présentation des solveurs de contraintes existants ainsi qu'à des exemples d'applications illustrant l'apport de la PPC dans divers domaines.

Le quatrième chapitre constitue un état de l'art des outils et techniques de réalité virtuelle (RV). Il est dédié à la présentation des caractéristiques d'un système de RV en mettant l'accent sur les notions d'interaction et d'immersion. Un aperçu des différentes techniques d'évaluation des systèmes de RV, est donné à la fin du chapitre.

Le cinquième chapitre porte sur la description de la problématique industrielle sur laquelle nous nous sommes appuyés pour proposer une formulation et une approche de résolution innovante. En effet, après avoir identifié les besoins et les objectifs nous proposons une approche interactive basée sur un couplage rassemblant les deux concepts présentés dans les chapitres 2 et 3. L'approche proposée est générique et peut être adaptée pour résoudre d'autres problèmes d'agencement.

Après avoir formulé et modélisé le problème d'agencement 3D, le chapitre 6 constitue une implémentation de l'approche proposée dans le chapitre 5. L'architecture générale du système développé, l'implémentation des contraintes ainsi que le protocole de communication entre l'utilisateur et le solveur sont décrits et illustrés. Les diverses fonctionnalités du système ainsi que les algorithmes utilisés sont également détaillés. A la fin du chapitre, un aperçu sur les possibilités d'assistance que peut fournir le système aux utilisateurs durant les agencement manuels est donné.

Le chapitre 7 constitue une présentation détaillée des possibilités d'assistance supportées par le système développé. Nous proposons des assistances basées sur le mécanisme d'anticipation (***look ahead***) très utilisé dans la recherche combinatoire. Ces assistances prédictives fournissent des informations visuelles pertinentes afin d'orienter l'utilisateur vers les possibilités du placement tout en respectant ses préférences.

Le chapitre 8 décrit la première des trois expérimentations présentées dans la troisième partie de ce chapitre. L'étude expérimentale réalisée permet de vérifier la résolution du verrou technologique (communication solveur-environnement virtuel) et d'analyser l'influence de l'intégration d'un solveur de contraintes dans un environnement virtuel, sur la performance de l'utilisateur au cours d'une tâche d'agencement 3D. Des données subjectives ont été également recueillies et analysées.

La deuxième étude expérimentale présentée dans le chapitre 9 a pour objectif d'identifier et d'évaluer différentes techniques d'interaction permettant à l'utilisateur de communiquer de manière efficace avec le système proposé. Dans ce contexte, différentes techniques d'interaction sont proposées et testées. Une analyse objective et subjective est réalisée sur les données recueillies durant des tâches d'agencement 3D.

Afin de montrer l'apport de l'assistance proposée par le système, le dernier chapitre est dédié à la présentation d'une étude expérimentale visant à étudier l'impact de l'une des assistances proposées dans le chapitre 7 sur la performance durant une tâche d'agencent 3D. Cette expérimentation, comme celle présentée dans le chapitre 9, permet de vérifier dans quelle mesure les verrous interactionnels sont résolus.

I

État de l’art

2

Problème de placement : Définition, formulation et résolution

2.1 Introduction

Plusieurs termes ou expressions sont utilisés dans la littérature pour définir les problèmes de placement [87]. Le terme "placement" a été utilisé par Wäscher et al. [63] pour définir les problèmes impliquant des composants devant être affectés à un ensemble fini de contenants. Dans les problèmes de *Bin packing* [10] par exemple, les composants sont référencés par les termes : objets, items ou pièces.

Les problèmes de placement désignent généralement les problèmes de découpe et de conditionnement ainsi que les problèmes d'agencement. L'étude des problèmes d'agencement 3D constitue le centre d'intérêt de notre travail. La formulation et la résolution de ces problèmes ont fait l'objet de plusieurs études impliquant des enjeux industriels importants. La diversité de ces problèmes conduit à différentes classifications [44] et conditionne la formulation et le développement des méthodes de résolution.

Dans certaines variantes du problème d'agencement (agencement de locaux, bâtiments, industrie maritime, etc), où la prise de décision est primordiale, la résolution automatique ne suffit pas au concepteur à prendre les meilleures décisions. En effet, l'interaction avec l'outil de résolution est de plus en plus recherchée, car les exigences de conception ne cessent d'évoluer et sont parfois difficiles à modéliser.

Le présent chapitre est consacré à la définition, la formulation et la classification des problèmes de placement. Un aperçu des méthodes de résolution est proposé et analysé. Puis, nous détaillons une problématique industrielle liée à ce type de problème, et qui constitue le point initial de la présente thèse.

2.2 Définition et exemples d'applications

Dans la littérature, on trouve différentes définitions du problème de placement. Cette diversité est due au fait que les chercheurs adaptent la définition de ce problème à leurs domaines [83][148]. Dans ce qui suit, nous retiendrons la définition la plus générique. En effet, un problème de placement peut être défini par un ou plusieurs contenants, un ensemble de composants (ou objets) soumis à des contraintes. L'objectif est de trouver un positionnement de ces composants dans le(s) contenant(s), tout en respectant les contraintes. Chacun des termes employés dans la définition nécessite plus de détails :

- Un ***contenant*** est un élément physique susceptible de contenir l'ensemble des composants. Un contenant peut être une salle, un véhicule, un bateau, une usine, une carte électronique, etc. Dans le cas où le problème est défini par plusieurs contenants, l'objectif est soit d'identifier le contenant dans lequel un composant donné sera placé, ou soit de trouver le nombre minimal de contenants nécessaires pour contenir l'ensemble des composants.

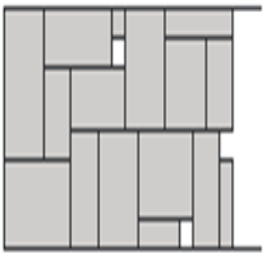
- Un ***composant*** est un objet physique ou abstrait (virtuel) caractérisé par sa forme géométrique et une liste d'attributs. Sa géométrie est simplement une longueur pour les problèmes 1D, et un modèle 3D pour les problèmes tridimensionnels. Les attributs sont les propriétés des composants, comme par exemple le poids, l'émission de la chaleur, etc. Ces attributs peuvent être impliqués dans des contraintes appliquées sur les composants.

- Les ***contraintes*** d'un problème de placement illustrent les restrictions ou les relations reliant les composants entre eux et avec le contenant. Elles permettent, quand cela est possible, de modéliser les exigences de conception et donc de filtrer les solutions admissibles. Certains critères de conception sont difficilement modélisables sous forme mathématique, et nécessitent le point de vue subjectif du concepteur ou de l'expert. Les contraintes les plus basiques sont les contraintes de non-chevauchement et d'appartenance au contenant.

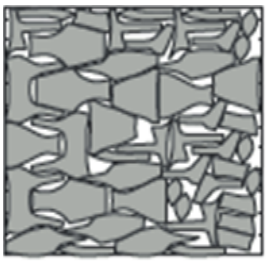
- Les ***inconnues*** d'un problème de placement sont généralement les positions et les orientations des composants dans le contenant. Également appelées variables, elles peuvent être de différents types : discrètes, continues ou permutations (liste ordonnée d'entiers faisant référence aux identifiants des composants). Nous présentons dans la suite les deux premiers types. Le choix du type de variables dépend de plusieurs facteurs comme la forme géométrique des composants, les contraintes, les objectifs, etc.

L'agencement d'espaces 2D et 3D a fait l'objet de nombreuses études et applications industrielles (Fig. 2.1) impliquant divers domaines. Par exemple, des problèmes de découpe de matières premières (tissus, tôles, etc.) dont l'objectif est de limiter le coût engendré par la découpe [87]. Des problèmes liés à la maximisation du chargement de véhicules (camions, avions, etc) ont été également traités. Dans le domaine de l'électronique, la problématique de l'agencement a été étudiée afin d'optimiser la répartition de composants sur les cartes et ainsi limiter l'échauffe-

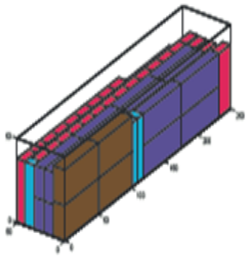
ment et la consommation du courant électrique [50]. A la fin de ce chapitre, nous mettrons l'accent sur les applications liées à l'agencement 3D.



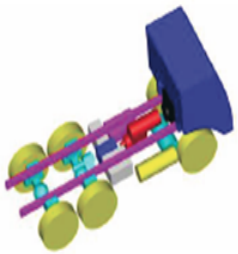
Problème de découpe



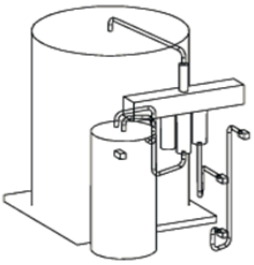
Problème de découpe irrégulière



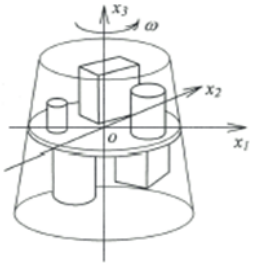
Chargement de containers



Agencement de composants d'un camion



Agencement d'une pompe à chaleur



Équipement d'un satellite

FIGURE 2.1 – *Exemples d'applications impliquant la problématique d'agencement (Images tirées de Jacquenot* [87]).

2.3 Formulation

La formulation d'un problème d'agencement consiste principalement à définir un modèle mathématique du problème. Ce modèle doit permettre : (1) d'identifier les inconnues ou variables du problème, (2) de tenir compte des exigences de conception en les traduisant sous forme de contraintes, et (3) de modéliser les objectifs visés. La description de ce type de problème diffère d'un domaine de recherche à un autre, ce qui conduit à une multitude de formulations. Le problème peut être vu comme un *problème de satisfaction de contraintes* (CSP) ou un *problème d'optimisation sous contraintes*. Un aperçu de ces deux types de modélisations est donné dans le chapitre suivant.

La partie qui suit présente une description des paramètres généralement impliqués dans la formulation d'un problème d'agencement.

2.3.1 Les variables du problème

Comme dit précédemment, les variables d'un problème d'agencement définissent généralement la position et l'orientation des composants dans le contenant. Elles peuvent être discrètes, continues ou permutations. Cependant, rien n'empêche d'intégrer les trois types de variables au sein d'un même problème.

Variables discrètes ou binaires

Ce type de variables est très utilisé lorsque les composants ont une forme parallélépipèdique/rectangulaire, ou sont définis comme une conjonction de cubes/carrés. Une variable discrète peut être utilisée pour représenter l'orientation d'un composant. L'orientation est choisie parmi un ensemble de valeurs définies au préalable. Quant aux variables binaires, elles permettent de vérifier si certaines contraintes sont respectées ou non.

Variables réelles

L'utilisation des variables réelles permet de placer les composants de manière continue. Ce type de variables est peu adapté pour des problèmes impliquant des exigences particulières (orientations discrètes, etc). Le placement des composants à travers les variables réelles est absolu ou relatif. Le placement d'un composant est dit absolu lorsque celui-ci n'est lié à aucun autre composant. En revanche, il est relatif lorsque sa position dépend d'un autre composant.

2.3.2 Contraintes

En plus des contraintes de non-chevauchement et d'appartenance [97][46][71][95], d'autres contraintes peuvent être formulées afin de traduire les exigences de conception et les relations inter-composants. Nous pouvons citer par exemple les contraintes de position/orientation, et les contraintes de proximité (distance minimale/maximale, alignement, etc.). D'autres contraintes, peu considérées dans les problèmes d'agencement, peuvent également être formulées. Il s'agit des contraintes *physiques* qui permettent la prise en compte de l'effet de certains composants sur d'autres, comme par exemple l'émission d'un champ électrique ou thermique. Dans le contexte de notre travail, nous avons implémenté trois contraintes physiques dont la description est donnée dans le chapitre 6. D'une façon générale, les contraintes d'un problème d'agencement dépendent fortement du cas d'application traité.

2.3.3 Le cas de l'accessibilité

Une exigence de conception particulière, souvent exprimée dans les problèmes d'agencement de locaux et de véhicules, concerne l'accessibilité aux composants. Elle peut traduire le besoin d'avoir un accès permanent à un composant donné pour une utilisation continue ou pour assurer sa maintenance. La représentation de cette exigence peut se faire simplement en ajoutant des objets abstraits (espaces libres) connectés au composant en question, ou en implémentant explicitement une

contrainte dédiée à cet effet. Récemment, Bénabès [22] a proposé une méthode permettant de mesurer l'accès à un composant depuis l'entrée du contenant. La méthode proposée se base sur le calcul du polygone d'appartenance (*Inner-Fit-Polygon en anglais*) [97].

Étant donné que la définition et la formulation des problèmes de placement dépendent fortement du cas étudié, et plus précisément de la forme des composants et des contraintes, il n'est pas évident de donner une classification ou une typologie complète à ces problèmes. Cependant, nous pouvons trier les problèmes de placement en problèmes de découpe et de conditionnement et en problèmes d'agencement.

2.4 Les problèmes de découpe et de conditionnement

Dans les problèmes de découpe et de conditionnement (*Cutting and Packing Problems en Anglais*) [45][41] [69], les composants ne sont pas reliés entre eux, ce qui fait que le positionnement d'un composant ne dépend pas d'un autre composant. La formulation mathématique de tels problèmes est généralement simple mais leur résolution est difficile [65]. La première classification de ces problèmes a été donnée par Dyckhoff [45], permettant d'identifier 96 problèmes. Celle-ci est basée sur quatre critères :

1. La dimension du problème (1D, 2D, 3D ou multi-dimensionnel) ;
2. Le type du placement : tous les composants et une sélection de contenants, ou bien une sélection de composants et tous les contenants ;
3. Les caractéristiques de contenants : 1 seul contenant, plusieurs contenants de tailles identiques ou différentes ;
4. Les caractéristiques des objets : composants identiques ou composants plus ou moins différents.

La typologie proposée a fait l'objet de quelques réserves au niveau international à cause de la redondance de certains problèmes (un problème pouvait être classé dans plusieurs catégories). C'est pour pallier à ce problème, que Washer et al. [63] ont proposé une nouvelle typologie en reprenant celle de Dyckhoff et en ajoutant un nouveau critère de classification portant sur la forme des composants (Fig. 2.2).

Plusieurs variétés de tels problèmes ont été étudiées. Comme par exemple, les problèmes de découpe [67][48], les problèmes de chargement de palettes [47], les problèmes de sac à dos et les problèmes de *Bin Packing* [53][17][57]. Pour les problèmes de *Bin Packing* (ou problèmes d'empaquetage), on note en particulier les algorithmes proposés par Richard Korf [92][93] [80]. En se basant sur une discrétisation de l'espace, l'auteur a testé son approche sur un problème consistant à trouver le plus petit conteneur pouvant contenir un ensemble de carrés ou cubes.

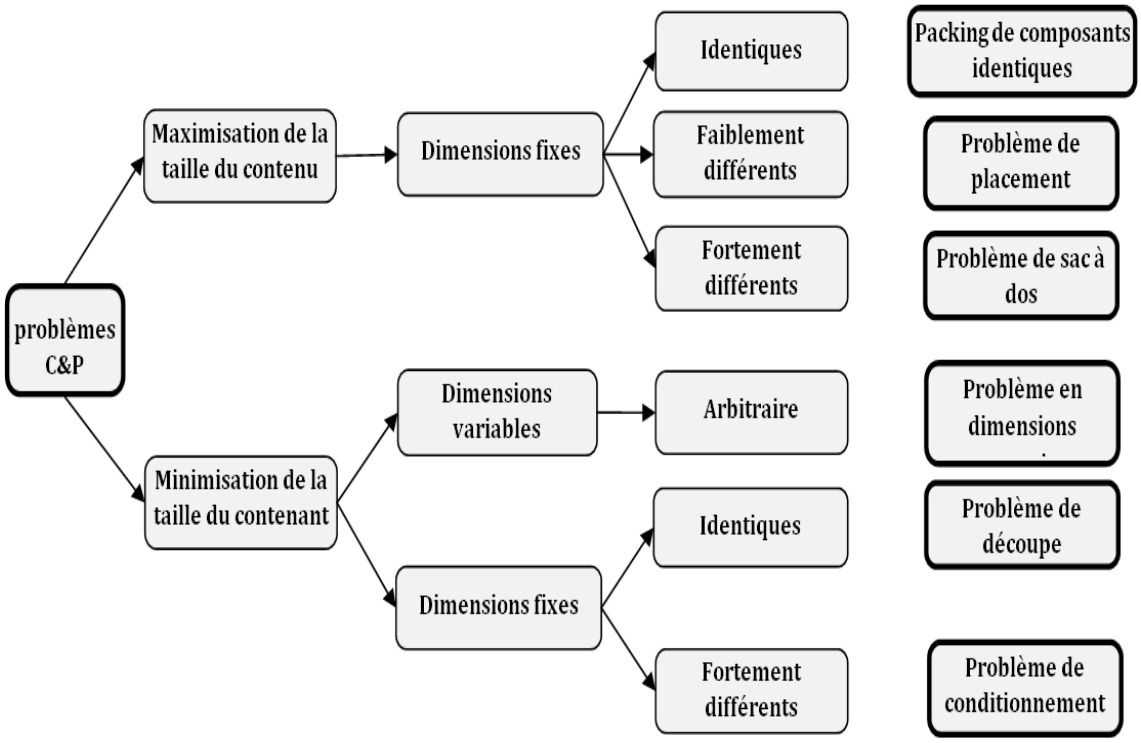


FIGURE 2.2 – *Classification des problèmes de découpe et de conditionnement proposée par Washer et al.* [63].

2.5 Les problèmes d'agencement

Contrairement aux problèmes de découpe et de conditionnement, dans les problèmes d'agencement, les composants et le contenant sont reliés entre eux. En d'autres termes, il existe une interaction entre les composants, conditionnant le positionnement de chacun. Cette catégorie de problèmes regroupe par exemple les problèmes d'agencement de locaux (ateliers, bureaux, etc.), les problèmes d'agencement de composants électroniques, ainsi que les problèmes d'agencement de véhicules (camions, bateaux, etc.).

L'agencement de locaux (*Facility layout problems*) consiste à agencer un ensemble de composants (meubles, machines, etc.) en tenant compte des interactions fonctionnelles inter-composants. L'agencement de composants électroniques vise à prendre en compte des spécificités liées au bon fonctionnement et à la durée de vie de ces composants (échauffement, consommation, etc.). L'agencement de véhicules vise à maximiser le chargement de ceux-ci (véhicules de transport) et à respecter des spécificités liées au confort et à l'accessibilité.

Les problèmes d'agencement d'espaces nécessitent généralement l'implication des concepteurs dans le processus de résolution. En effet, l'évolution des besoins et la complexité des systèmes nécessite de plus en plus une interaction entre le savoir faire des concepteurs et l'outil utilisé pour la résolution. Les exigences et les remarques des concepteurs servent à la définition des contraintes et à l'identification des objectifs. Après résolution, la décision finale appartient toujours aux concepteurs, qui valident ou déclinent la solution proposée. C'est dans ce contexte que le

présent travail a été mené. En effet, nous avons privilégié l'étude de l'interaction que peut avoir un concepteur avec un outil de résolution et surtout l'assistance que peut fournir celui-ci durant la phase de prise de décision.

2.5.1 Agencement 3D

A travers sa pluridisciplinarité et les moyens qu'elle offre pour synthétiser et interagir avec des scènes tant réelles qu'imaginaires, la réalité virtuelle (RV) simplifie considérablement la manière avec laquelle nous percevons et analysons le monde qui nous entoure. Ce concept a été introduit dans la problématique de l'agencement d'espaces, offrant une meilleure visualisation des données et une meilleure prise de décision. La géométrie complexe des composants est modélisée sous forme de modèles CAO.

L'agencement 3D a fait l'objet de plusieurs travaux dans la littérature. Par exemple, Fernando et al, ont exposé les détails de conception d'un environnement virtuel (EV) supportant l'intégration des techniques de programmation par contraintes [55]. Le travail réalisé porte sur la conception d'EVs pouvant supporter des contraintes sans pour autant traiter un problème spécifique.

Des travaux plus ciblés ont traité le problème d'agencement dans les EVs. En effet, Xu et .al, ont combiné des contraintes physiques et géométriques dans le but de faciliter la disposition ou l'aménagement d'une scène 3D [147]. L'agencement de la scène peut être sensiblement accéléré avec un moteur pseudo-physique simple de et quelques informations sémantiques. Dans le même contexte, Sanchez et al., ont présenté un nouveau concept pour concevoir un système dédié à la disposition des scènes 3D. Ce travail est basé sur l'utilisation d'un algorithme génétique [131]. Le système peut traiter un ensemble complexe de contraintes, y compris des contraintes géométriques et pseudo-physiques. Calderon et al., ont élaboré un travail original pour l'usage des EVs dans la résolution des problèmes d'agencement interactifs [34]. Ce travail étend les environnements 3D afin qu'ils servent d'interface réactive permettant l'exploration de l'espace de solutions. Le système proposé est basé sur une communication entièrement interactive où la visualisation et la génération d'une nouvelle solution sont sous le contrôle de l'utilisateur. Les auteurs étaient parmi les premiers à intégrer et implémenter des contraintes physiques et plus particulièrement la contrainte thermique.

Fages et al., ont implémenté une interface graphique générique (CLPGUI) pour visualiser et commander des programmes contraints [54]. L'architecture proposée se base sur une communication entre deux parties : un processus de résolution et une interface graphique. Le CLPGUI a été testé sur un problème d'agencement 3D impliquant quelques objets à placer dans un espace restreint. Plus récemment, Benabés [22] a proposé une démarche globale d'optimisation d'agencement d'espaces offrant les outils pour décrire, formuler et résoudre un problème d'agencement. L'approche de résolution est basée sur algorithme hybride, préalablement proposé par Jacquenot [87]. Cette approche permet au concepteur d'interagir, d'une façon très basique, avec les solutions proposées par l'algorithme d'optimisation. La démarche proposée a été testée sur un cas industriel consistant à placer divers équipements dans un "shelter" (véhicule de forme parallélépipèdique).

Dans le contexte de la conception des jeux vidéo, Tim et al. [144], ont introduit une nouvelle approche d'agencement afin d'accélérer les méthodes de conception manuelles et créer automatiquement des pièces d'un jeu 3D. Leur approche de résolution peut être utilisée pour la génération procédurale en fournissant au solveur utilisé un plan défini par l'utilisateur. Dans ce plan, les utilisateurs peuvent spécifier les objets à placer comme des instances de classes, qui à leur tour contiennent des règles sur la façon dont les objets doivent être placés.

Medjdoub [101] a proposé une approche basée sur la programmation par contraintes afin de réduire la charge de la conception schématique, le dimensionnement et la disposition des systèmes de ventilation sur les plafonds d'un bâtiment. Afin de faire face à la géométrie complexe et la taille du bâtiment réel, l'auteur a opté pour un raisonnement modulaire (cas par cas).

La problématique de l'agencement dans sa version tridimensionnelle, a fait également l'objet d'autres travaux issus des cas industriels spécifiques. Voici quelques exemples cités dans la thèse de Jacquenot [87] :

- L'agencement des différents composants d'une pompe à chaleur [84] ;
- La disposition des différents composants d'un moteur de voiture [149] ;
- L'agencement des différents composants d'une transmission mécanique [130] ;
- La disposition des différents composants d'un satellite [14][70].

Problème d'agencement d'engins militaires

Ce problème, détaillé dans le chapitre 5, consiste à disposer un ensemble d'équipements informatiques et électroniques à l'intérieur d'un shelter (espace technique mobile) ou d'un VAB (Véhicule de l'Avant Blindé). La problématique a été traitée dans le cadre d'une collaboration entre le laboratoire LISA et *Thales Communications & Security*. Des propositions ont été alors faites mais restent focalisées sur l'approche manuelle (agencement manuel). Différents logiciels ont été développés (*DIM 3D* 1.0 et 1.1) :

Une première version du logiciel *DIM 3D* (Fig. 2.3) a été développée pour aménager un prototype virtuel de VAB, avec lequel le client peut interagir et choisir la disposition des objets prédéfinis (extincteur, armoire, etc). Différentes interfaces d'interaction ont été également proposées (souris, clavier, Wiimote™, Patriot™). Cette version ne supporte pas l'automatisation de l'aménagement et le chargement dynamique des objets, et atteint rapidement ses limites en terme d'interaction puisque un utilisateur non expérimenté se retrouvait rapidement perdu dans l'orientation et au déplacement des objets.

Une deuxième version (Fig. 2.4) a été ensuite développée pour améliorer l'interaction avec le prototype virtuel et faciliter la disposition des objets, sans résoudre

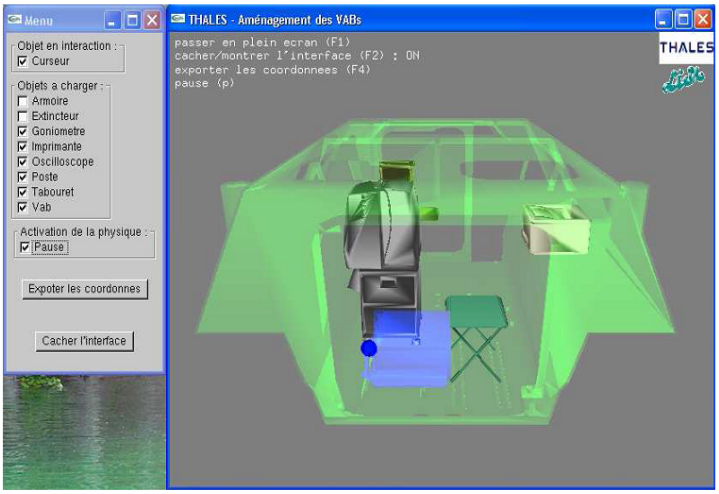


FIGURE 2.3 – *Interface du logiciel DIM 3D* 1.0.

pour autant la problématique liée à l'aménagement automatique. Un moteur physique a été intégré pour détecter la collision entre objets et véhicule.

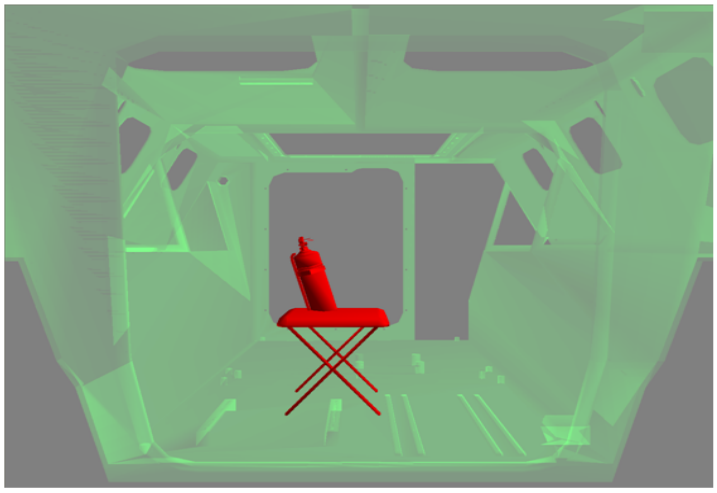


FIGURE 2.4 – *Détection de collisions entre deux objets lors d'un agencement manuel du VAB.*

Le problème de l'aménagement des shelters a fait l'objet d'une étude plus approfondie dans le cadre de la thèse de Bénabès [22]. Cette étude constituait à matérialiser et valider la méthode que l'auteur a proposé sur un cas industriel proposé par *Thales Communications & Security*. L'auteur a proposé une formulation du problème en 2D (Fig. 2.5) et en 3D en ne prenant en considération que des contraintes géométriques simples (non-chevauchement).

La méthode de résolution proposée, et basée sur l'utilisation d'un algorithme génétique couplé avec des modules d'optimisation locale, a certes proposé une solution au problème (Fig. 2.6) mais n'a pas considéré de contraintes physiques liées aux caractéristiques de chaque équipement (champ thermique par exemple). De plus, l'interaction avec les solutions calculées est très basique et ne permet pas de traiter toutes les demandes de l'utilisateur (ajout/suppression de contrainte par

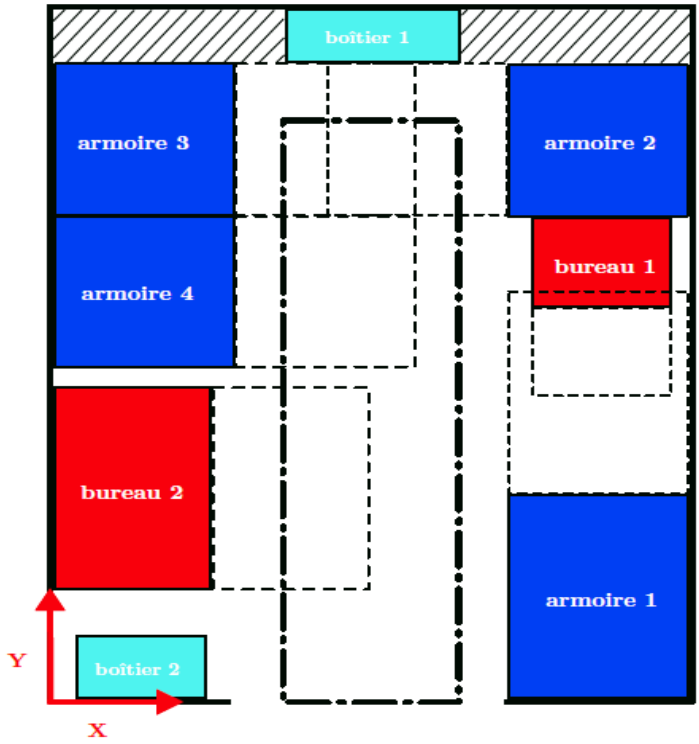


FIGURE 2.5 – *Modèle géométrique 2D du problème d'agencement du shelter proposé par Bénabès* [22].

exemple). Dans le cas où le client agence manuellement certains équipements, aucune information ou indication n'est délivrée pour le guider et l'aider à prendre les bonnes décisions.

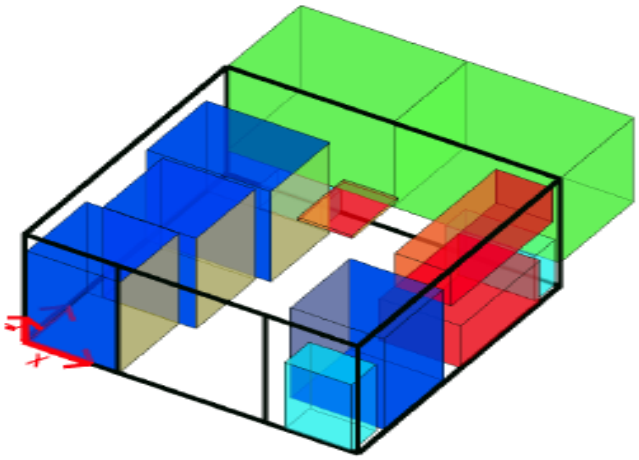


FIGURE 2.6 – *Agencement 3D proposé par la méthode de résolution (Image tirée de Bénabès* [22]).

Indépendamment des méthodes de résolutions utilisées dans les approches antérieures, la plupart de ces approches ([131],[147],[55],[22],[101],[144]) n'ont pas privilégié l'assistance de l'utilisateur lors de son interaction avec le système de résolution. Par exemple, l'interaction proposée par Bénabès [22], Calderon [34] et Fages [54] est purement statique. En effet, l'utilisateur modifie manuellement les

solutions proposées et c'est après cela que le système agit pour vérifier la faisabilité ou non des solutions modifiées. Ainsi, et dans le but d'améliorer l'interaction entre l'utilisateur et l'outil de résolution, nous proposons dans le chapitre 7 divers mécanises d'assistance quasi temps-réel basés sur le principe de ***look ahead*** [39].

2.6 Méthodes de résolution

Les stratégies de résolution élaborées sont souvent associées à un problème d'agencement particulier. En effet, ces stratégies sont adaptées aux exigences et aux spécificités du problème. Ainsi, leur généralisation pour résoudre d'autres problèmes est difficile. Il est cependant possible de classifier ces méthodes selon l'approche de résolution utilisée :

1. ***Les méthodes exactes***, qui garantissent le respect de toutes les contraintes de conception. Ces méthodes sont principalement issues de la programmation par contraintes.

2. ***Les méthodes approchées***, qui tolèrent un non-respect de certaines contraintes et qui peuvent trouver des solutions plus rapidement que les méthodes exactes. Nous pouvons citer par exemple, les méthodes basées sur l'algorithme du recuit simulé ([134],[142]) et sur les algorithmes génétiques ([148],[129]).

Malgré les différences entre ces deux types de méthodes, qui seront détaillées dans le chapitre 3, rien n'empêche de les combiner pour tirer profit de chacune d'entre-elle. L'élaboration des stratégies hybrides a été déjà abordée dans la littérature [148, 87].

2.6.1 Exemple de stratégie de résolution

Miao et al. [148] ont utilisé un algorithme génétique pour placer un ensemble de composants (moteur, système de transmissions, réservoir, etc) sur un camion militaire. Des aspects liés à la sécurité et l'accessibilité ont été considérés. Les auteurs ont élaboré un modèle paramétrique, sur lequel une optimisation multi-objectif a été réalisée. Deux types de variables ont été utilisés : des variables réelles pour les positions et des variables discrètes pour les orientations.

Dans le même contexte, Jacquenot a développé une méthode hybride pour résoudre des problèmes de placement multi-objectifs [87]. La méthode utilisée repose sur un couplage entre un algorithme génétique et un algorithme de séparation. La méthode est illustrée à la Figure 2.7. Un algorithme de séparation est utilisé pour modifier les variables de positionnement des composants afin que l'agencement respecte toutes les contraintes de conception. L'algorithme hybride teste si un "individu" respecte les contraintes de conception. Si c'est le cas, les objectifs sont évalués et l'algorithme essaie de faire évoluer cet individu vers un individu meilleur, via les différents opérateurs génétiques. Dans le cas contraire, l'algorithme de séparation est appliqué et l'algorithme global poursuit son processus de résolution.

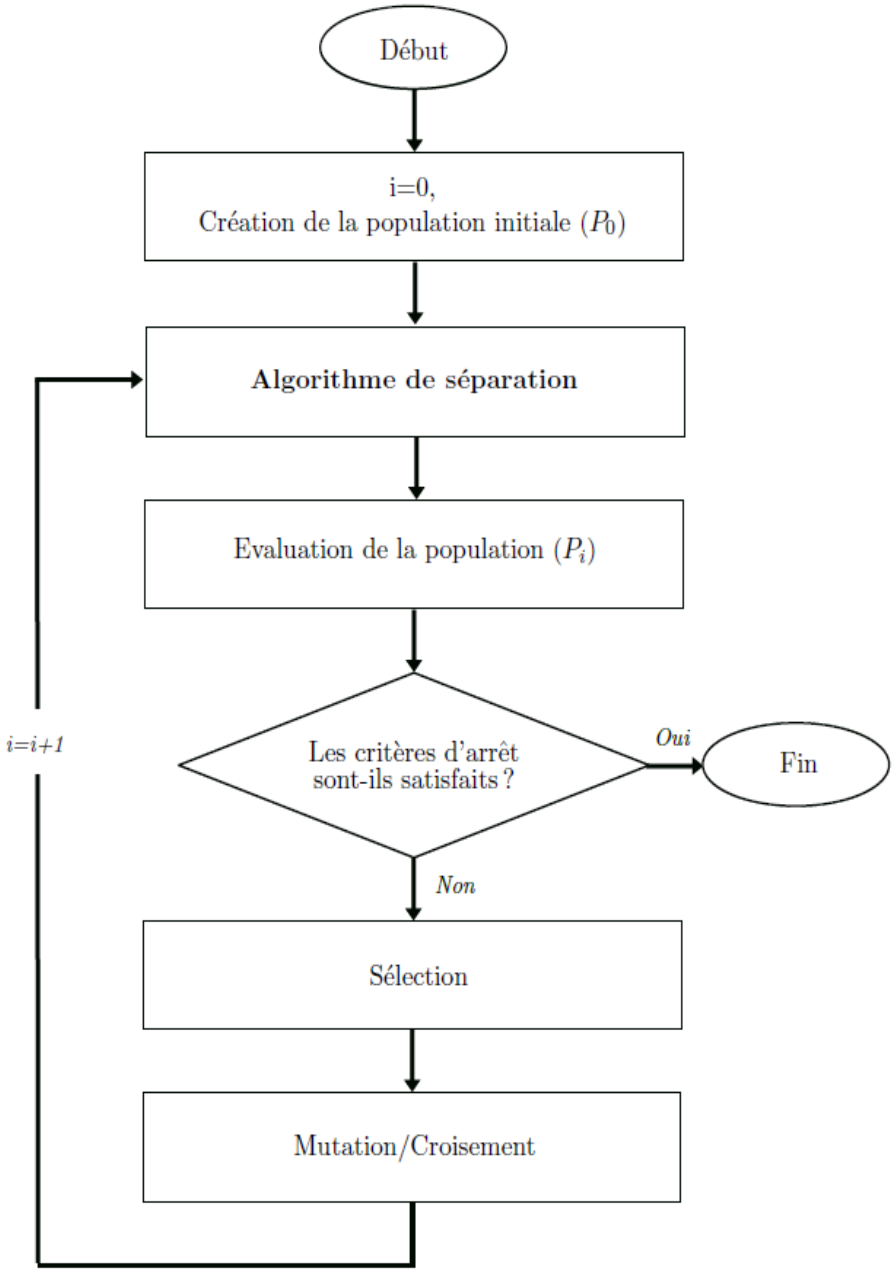


FIGURE 2.7 – *La méthode hybride proposée par Jacquenot [87].*

Aider les concepteurs a faire les bons choix de conception nécessite des outils interactifs permettant de visualiser les données numériques issues du processus de résolution. La visualisation de ces données peut aller de la présentation de plusieurs solutions jusqu'à la justification de l'infaisabilité d'un problème donné. En effet, une fois confronté au message ***pas de solution*** renvoyé par le solveur, il est peu évident de comprendre réellement la cause de ce message sans entrer dans le détail du fonctionnement du système de résolution. Dans de telles situations, et bien que l'utilisation des méthodes approchées ait fait ses preuves dans la résolution de problèmes d'agencement, celles-ci peuvent s'avérer insuffisantes et peu adaptées. En effet, ces méthodes ne couvrent pas la totalité de l'espace de recherche et ne fournissent pas d'information sur les causes d'un possible échec lors de la recherche de solutions. Ainsi, pour apporter une assistance pertinente aux concepteurs, nous avons proposé d'utiliser la programmation par contrainte pour la formulation et la résolution interactive de problèmes d'agencement 3D. Ce choix est justifié avec plus de détails dans le chapitre 5.

2.7 Conclusion

Ce chapitre a présenté l'ensemble des concepts permettant la description, la formulation et la résolution de problèmes de placement d'une façon générale et les problèmes d'agencement 3D en particulier. Un système de résolution doit, non seulement proposer des calculs automatiques de solutions, mais aussi des moyens permettant aux concepteurs de prendre les meilleurs décisions quant aux choix de conception. La diversité de ces problèmes a donné naissance à plusieurs applications et études dont un aperçu a été donné au cours de ce chapitre. Notre choix s'est porté sur l'utilisation de la programmation par contraintes (PPC) pour la formulation et la résolution interactive. Une description détaillée des concepts de celle-ci est fournie dans le chapitre suivant.

L'utilisation de la PPC à travers un simple éditeur de texte permettant d'entrer un ensemble de contraintes est bien évidemment inappropriée dans le contexte des problèmes d'agencement 3D. Pour cette raison, nous avons décidé de faire appel aux techniques de la réalité virtuelle (RV) pour : (1) utiliser la PPC d'une manière intuitive et attrayante et (2) offrir à l'utilisateur des moyens d'interaction pertinents pour une meilleure implication dans le processus de résolution et une prise en compte de ses préférences. Les concepts de la RV ainsi que des exemples d'application sont donnés dans le chapitre 4.

3

Programmation par contraintes

3.1 Introduction

Issue de de la programmation logique et de l'intelligence artificielle, la Programmation Par Contraintes (PPC) est apparue dans les années 80. Elle rassemble les méthodes et les techniques utilisées pour la résolution de problèmes définis sur des domaines discrets et continus, et s'applique plus particulièrement aux problèmes de satisfaction de contraintes (CSP). En PPC, un problème est défini par un ensemble de relations ou restrictions (contraintes) reliant un ensemble de variables. Trouver une solution à un problème en PPC consiste à instancier toutes les variables (leur donner des valeurs) du problème en respectant la totalité des contraintes.

La programmation par contraintes est basée sur trois principaux mécanismes : filtrage, propagation de contraintes et recherche de solutions. La PPC se base également sur un aspect énumératif dû au fait que toutes les pistes d'éventuelles solutions sont explorées. Ainsi, aucune solution ne peut être omise durant l'exploration de l'espace de recherche. Les domaines d'applications de la PPC sont très variés, elle est utilisée pour résoudre des problèmes dans de nombreux domaines de l'activité humaine. Des exemples sont donnés dans la première partie de ce chapitre.

Dans ce qui suit, nous présentons quelques exemples d'applications ainsi que le formalisme CSP. Par la suite, nous décrivons chacun des trois mécanismes de la PPC (filtrage, propagation et recherche de solutions). Finalement, nous donnons un aperçu de solveurs de contraintes existants qui mettent à disposition des utilisateurs des moyens pour la modélisation et la résolution de problèmes variés.

3.2 Définition et exemples d'applications

3.2.1 Définition

Certains des algorithmes utilisés pour résoudre des problèmes définis par des contraintes, ont été intégrés dans des langages de programmation, définissant ainsi

un nouveau paradigme de programmation appelé "programmation par contraintes" [138]. La PPC est un mode de programmation déclaratif basé sur la séparation entre la modélisation d'un problème et sa résolution. En PPC, un problème est modélisé via une conjonction de contraintes (propriétés qui doivent être vérifiées) postées sur des variables (inconnues du problème).

3.2.2 Exemples d'applications

Comme dit précédemment, la PPC couvre un large éventail d'applications. Parmi les plus connues, on pourra citer la planification, l'ordonnancement, la gestion du temps et l'affectation de ressources, etc [122]. Il est important de noter que la PPC est utilisée par des acteurs mondialement connus tels que Oracle, Daimler-Chrysler, Nissan, Peugeot-Citron (Production Industrielle), Airbus, etc. Nous reproduisons dans ce qui suit des exemples tirés du rapport de HDR de Régin [122].

Gestion et Planification La PPC est utilisée dans la planification de la logistique, de la distribution, du personnel, de l'affectation d'équipes, de missions, des techniciens d'installation et de maintenance. Elle a été également utilisée pour la gestion de la chaîne logistique et d'entrepôts, la configuration et diagnostic d'équipements, l'ordonnancement de lignes de production, etc.

Production industrielle L'entreprise *Chrysler* a mis en place un système de planification de la production de ses véhicules via ILOG Solver [4]. L'application gère la séquence des opérations de peinture des véhicules et améliore la productivité de 15 usines du groupe en Amérique du nord, au Mexique et en Europe. Ce système a permis au producteur automobile de réduire ses coûts de production de 500000 dollars par an et par usine, soit une économie totale de 7 à 9 millions de dollars par an.

Transport Dans le domaine des transports, la PPC a fait ses preuves pour des applications telles que l'affectation d'équipages, de comptoirs, de portes d'embarquement et de tapis à bagages, l'ordonnancement d'équipements, la gestion de flottes et la planification du trafic.

Commerce en ligne La PPC est utilisée pour résoudre des problèmes liés à la gestion des commandes et des approvisionnements, le service de voyages, la gestion des crédits ou de conseils financiers.

Défense La PPC a été utilisée pour planifier la formation des 85000 militaires de la British Army.

Agencement ou aménagement La PPC a fait ses preuves dans de nombreuses variétés de problèmes de placement d'une façon générale. De l'aménagement d'intérieur, en passant par l'assemblage, jusqu'à l'agencement des véhicules et bateaux, la PPC a permis de résoudre tous ces problèmes. Une description détaillée sur l'utilisation de la PPC dans ce genre de problème est donnée au chapitre 4.

3.3 Modélisation

Nous décrivons dans cette section les différents éléments permettant de modéliser un problème défini par des contraintes. Tout d'abord, nous définissons les problèmes de satisfaction de contraintes puis nous présentons les problèmes d'optimisation sous contraintes dont les solutions maximisent ou minimisent la valeur d'une fonction de coût.

3.3.1 Problème de Satisfaction de Contraintes (CSP)

Un Problème de Satisfaction de Contraintes ou (CSP) est une modélisation d'un problème sous forme de variables (les inconnues du problème) et de contraintes. Chacune de ces variables dispose d'un ensemble de valeurs "autorisées" appelé domaine. Les contraintes présentent des restrictions sur les valeurs que peuvent prendre simultanément les variables.

Plus formellement, un CSP est défini par un triplet (X, D, C) tel que :

- $X = \{X_1, X_2, ..., X_n\}$ est l'ensemble des variables du problème,
- $D(X_i)/X_i \in X$ est la fonction qui retourne le domaine de X_i,
- $C = \{C_1, C_2, ..., C_k\}$ est l'ensemble des contraintes du problème.

Un grand nombre de problèmes peuvent être formalisées sous forme de CSP. A titre d'exemple, nous citons le problème des reines sur l'échiquier (Sec. 3.3.1), les cryptogrammes ($SEND + MORE = MONEY$), les problèmes d'allocation de ressources, la gestion d'emploi du temps, etc. Généralement, les CSP font partie de la classe des problèmes NP-complets et les méthodes de résolution sont souvent basées sur des algorithmes de complexité très élevée. Nous nous limiterons dans ce chapitre à la présentation des méthodes de résolution des CSP à domaines finis.

Notion de contrainte et d'affectation

Une ***contrainte*** conditionne l'instanciation des variables sur lesquelles elle porte. Elle est considérée comme une propriété déclarative et relationnelle parce qu'elle impose une relation entre les variables sans spécifier d'algorithme pour assurer cette relation. Le nombre de variables sur lequel porte une contrainte est appelé "arité", une contrainte peut être donc unaire, binaire,..., ou n-aire. Les contraintes peuvent être exprimées sous forme mathématique ou sous forme de valeurs compatibles, etc.

On appelle ***affectation***, l'instanciation d'une ou plusieurs variables par des valeurs appartenant aux domaines de celles-ci. Une affectation est dite ***totale*** si toutes les variables du problème sont instanciées. En revanche, elle est ***partielle*** s'il existe au moins une variable qui n'est pas encore instanciée. Une affectation (totale ou partielle) est ***consistante*** si elle ne viole aucune contrainte, et ***inconsistante*** dans le cas contraire. Une solution à un CSP est une affectation totale consistante.

Exemple de formulation : le problème des N-reines

Un des problèmes les plus connus en PPC est le problème des N-reines. Il consiste à placer les N reines sur un échiquier défini par N lignes et N colonnes tout en garantissant qu'aucune reine ne soit en prise. Deux reines sont en prise si elles sont placées sur une même ligne, sur une même colonne ou sur une même diagonale de l'échiquier.

Partant du fait qu'une reine et une seule sera placée par colonne, le problème se résume au choix de la ligne. Une modélisation possible du problème des N-reines est définie par un ensemble de variables X de telle sorte que X_i désigne le numéro de ligne où se place la reine située sur la colonne i.

- $X = \{X_i / i \in [1..n]\}$
- $D(X_i) = \{j / X_i \in X, j \in [1..n]\}$
- $C = \{C_1 \bigcup C_2 \bigcup C_3\}$ avec :
  - $C_1 = \{X_i \neq X_j / i \neq j$ et $i, j \in [1..n]\}$
    les reines doivent être placées sur des lignes différentes
  - $C_2 = \{X_i + i \neq X_j + j / i \neq j$ et $i, j \in [1..n]\}$
    les reines doivent être placées sur des diagonales montantes différentes
  - $C_3 = \{X_i - i \neq X_j - j / i \neq j$ et $i, j \in [1..n]\}$
    les reines doivent être placées sur des diagonales descendantes différentes

Exemple de formulation : Aménagement d'une salle de spectacle

Nous présentons dans ce paragraphe un deuxième exemple de formulation de problème sous forme de CSP. Le problème consiste à aménager k chaises dans une salle de spectacle tout en respectant un ensemble de contraintes géométriques ou spatiales. En effet, les chaises doivent être séparés des murs gauche et droit par une distance de $1m$, $2m$ du mur arrière et $3m$ du mur avant (C_1). Une distance de $50cm$ doit séparée les chaises "voisines" (C_2 et C_3). Horizontalement, les chaises doivent être placés par groupe de trois, séparés par une distance de $1.5m$ (C_4). Les chaises doivent être également alignées horizontalement et verticalement(C_5). On suppose que les chaises ont la même largeur et la même longueur ($50cm$).

Une formulation possible de ce problème peut se baser sur une discrétisation de l'espace à aménager (Fig 3.1). Des variables booléennes peuvent alors être utilisées pour désigner chaque position dans l'espace discrétisé. Par exemple, $P_{i,j}$ vaut 1 si une chaise peut être placée dans la position (i, j), 0 sinon. Ainsi, le CSP représentant le problème est défini comme suit :

- $X = \{P_{i,j} / i \in [0..n]$,$j \in [0..m]\}$
- $D(P_{i,j}) = [0, 1] / i \in [0..n]$,$j \in [0..m]$
- $C = \{C_1 \bigcup C_2 \bigcup C_3 \bigcup C_4 \bigcup C_5 \bigcup C_6\}$ avec :

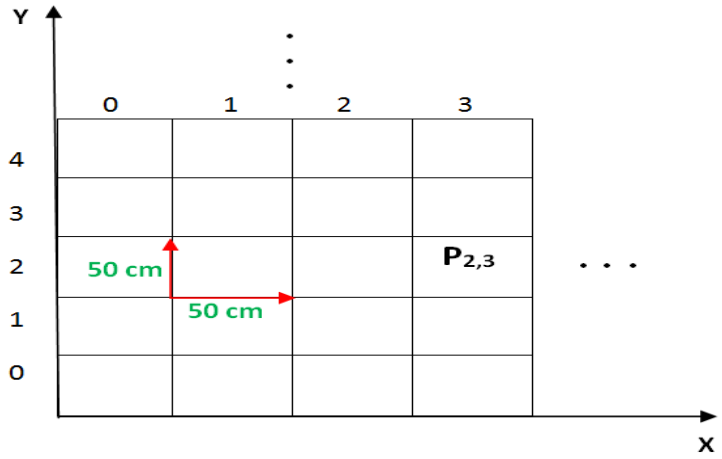


FIGURE 3.1 – *Discrétisation de l'espace à agencer.*

- C_1 : Pour $j \in [0..m], \sum P_{0,j} + \sum P_{1,j} = 0$
  Pour $i \in [0..n], \sum P_{i,0} + \sum P_{i,1} + \sum P_{i,2} + \sum P_{i,3} = 0$
  Pour $i \in [0..n], \sum P_{i,0} + \sum P_{i,1} + \sum P_{i,2} + \sum P_{i,3} + \sum P_{i,4} + \sum P_{i,5} = 0$

- C_2 : si $P_{i,j} = 1$ alors
  (1) $P_{i+1,j} = 0$
  (2) Si $\sum P_{i,j} < k$ alors $P_{i+2,j} = 1$

- C_3 : si $P_{i,j} = 1$ alors
  (1) $P_{i,j+1} = 0$
  (2) Si $\sum P_{i,j} < k$ alors $P_{i,j+2} = 1$

- C_4 : si $P_{i,j} + P_{i+2,j} + P_{i+4,j} = 3$ alors
  (1) $P_{i+5,j} + P_{i+6,j} + P_{i+7,j} = 0$
  (2) Si $\sum P_{i,j} < k$ alors $P_{i+8,j} = 1$

- C_5 : si $P_{i,j} = 1$ alors $P_{i+1,j+1} + P_{i+1,j-1} = 0$

- C_6 : $\sum P_{i,j} = k$

Notion de CSP sur-contraint/sous-contraint

Quand un CSP n'admet pas de solution, il est dit sur-contraint. En effet, il y a trop de contraintes et ce n'est pas possible de toutes les satisfaire. Une solution approchée à ce problème consiste à trouver une affectation totale avec un minimum de contraintes violées.

Inversement, quand un CSP admet un grand nombre de solutions, il est dit sous-contraint. Dans ce cas, des préférences ou des critères d'optimalité peuvent être fixés pour "filtrer" les bonnes solutions. Pour ce faire, on ajoute une fonction objectif à minimiser ou maximiser. Ce qui amène à définir les Problèmes d'optimisation sous contraintes.

3.3.2 Problème d'optimisation sous contraintes

Un problème d'optimisation sous contraintes est un CSP auquel est ajoutée une fonction objectif F. La définition de cette fonction engendre une double recherche : trouver les solutions réalisables et identifier la meilleure. La fonction F est souvent associée à une variable dont le domaine coincide avec les limites supérieures et inférieures de F. Dans de nombreuses situations, l'éventuelle solution à un problème doit tenir compte des critères ou des préférences fixés par l'utilisateur ou le décideur. Ces critères d'optimalité permettent de différencier les solutions et donc de choisir la solution optimale vis à vis de ces critères. Un critère d'optimalité est souvent défini comme une maximisation ou minimisation de la fonction F.

3.4 Filtrage et consistance locale

3.4.1 Filtrage

Appelé également algorithme de contraction de domaines, un algorithme de filtrage représente l'effet d'une contrainte sur le domaine des variables pour lesquelles elle a été définie. Il consiste à supprimer des valeurs des domaines des variables pour lesquelles il n'est pas possible de satisfaire la contrainte (valeurs inconsistantes). Par exemple, pour la contrainte $(X = Y)$ avec $D(X) = [0..15]$, $D(Y) = [10..30]$, un algorithme de filtrage associé à cette contrainte élimine les valeurs de 0 à 9 de $D(X)$ et les valeurs de 16 à 30 de $D(Y)$.

Le mécanisme de filtrage constitue une des approches les plus classiques en intelligence artificielle pour pallier au problème de complexité combinatoire rencontré lors de la résolution des CSPs. Il a pour effet la simplification du problème par réduction de l'espace de recherche et dans certains cas, il peut servir à détecter une inconsistance.

3.4.2 Niveaux de consistance locale

La consistance est une propriété de satisfiabilité entre valeurs de domaines des variables et contraintes. Prouver la consistance globale d'un CSP est un problème NP-complet, ce qui a conduit à définir des propriétés "moins fortes" que la consistance globale définissant ainsi la consistance locale. En général, les algorithmes de consistance sont incomplets, c'est-à-dire qu'ils ne retirent pas toutes les valeurs inconsistantes des domaines.

L'utilisation des techniques de renforcement de la consistance est devenue indispensable pour réduire efficacement l'espace de recherche. Ces techniques peuvent être utilisées durant une phase de prétraitement ou tout au long de la recherche d'une solution. Différents niveaux de consistance locale existent comme la consistance de noeud, la consistance d'arc ou la k-consistance.

Consistance de noeud

La consistance de noeud consiste à éliminer les valeurs de chacune des variables qui ne satisfont pas toutes les contraintes unaires (impliquant qu'une seule variable)

du problème. Ainsi, Un CSP, défini par le triplet (X, D, C), est *consistant de noeud* si pour toute variable $X_i \in X$, et pour toute valeur v de $\in D(X_i)$, l'affectation partielle $\{X_i \leftarrow v\}$ satisfait toutes les contraintes unaires de C.

Consistance d'arc (AC)

Les travaux de Mackworth [99] ont constitué les premières recherches sur la consistance d'arc. L'auteur a proposé trois algorithmes de renforcement de AC ($AC1$, $AC2$ et $AC3$). Une amélioration de ces algorithmes a été proposée en 1986 par Mohr et Henderson [105] en développant un nouvel algorithme ($AC4$). En 1994, Bessiére [18] a développé la méthode $AC6$ qui permet de pallier aux problèmes de complexité concernant l'espace de l'$AC4$. Bessiére, Freuder et al. [58] ont développé l'$AC7$, un algorithme efficace pour diminuer le nombre de vérifications de consistance inutiles pour les graphes de contraintes bidirectionnelles. Enfin, Bessiére [19] ont proposé le $AC2000$ et le $AC2001$ en 2001.

La consistance d'arc associée à une contrainte est réalisée en supprimant toutes les valeurs des variables impliquées dans la contrainte qui ne sont pas consistantes avec celle-ci. Dans un CSP (X, D, C), une contrainte $c \in C$ est consistante d'arc si pour toute variable de $X_i \in X$ impliquée dans la contrainte, et pour toute valeur $v \in D(X_i)$, il existe au moins une valeur dans le domaine de chaque autre variable impliquée de telle façon que c soit satisfaite.

Consistance de chemin (PC)

Les premiers algorithmes de la consistance de chemin ont été introduits par Mackworth [99] en 1977. Mohr et Henderson [105] ont développé l'algorithme $PC3$ en 1986, tandis que l'algorithme $PC4$ a été proposé en 1988.

Une paire des variables est chemin consistante avec une troisième variable si chaque paire des valeurs consistantes peut être étendue à l'autre variable de telle façon que toutes les contraintes entre les trois variables soient satisfaites. Ainsi, un CSP (X, D, C) vérifie la consistance de chemin, si et seulement si, pour chaque pair de variables $(X_i, X_j) \in X$ et pour toute troisième variable $X_k \in X$, si $X_i \leftarrow v_1, X_j \leftarrow v_2$ est une affectation partielle consistante, alors il existe une valeur $v_3 \in D(X_k)$ tel que $(X_i \leftarrow v_1, X_k \leftarrow v_3)$ et $(X_k \leftarrow v_3, X_j \leftarrow v_2)$ soient consistantes.

K-consistance

La consistance d'arc, comme la consistance de chemin, permet de réduire l'espace de recherche d'un CSP mais ne suffit pas à résoudre celui-ci. Dans le but de réduire davantage l'espace de recherche d'un CSP, des niveaux de consistances plus forts ont été proposés par Freuder [49], ce sont les k-consistances et les k-consistances fortes.

- ***K-consistance*** un CSP (X, D, C) est dit k-consistant si, pour chaque affectation partielle et consistante $\boldsymbol{A}$ d'un ensemble de $(k-1)$ variables $(X_1, ..., X_{k-1})$, il existe au moins une valeur $v \in D(X_k)$ tel que $\{X_k \leftarrow v\} \bigcup \boldsymbol{A}$ constitue une affectation consistante.

- ***K-consistance forte*** un CSP est dit fortement k-consistant s'il est i-consistant pour tout $i \in [1, .., k]$. Par exemple, pour qu'un CSP soit fortement 2-consistant, il faut qu'il soit 1-consistant et 2-consistant.

3.5 Propagation de contraintes

Dès qu'une contrainte conduit à une réduction du domaine d'une variable (filtrage), les conséquences de cette modification sont étudiées pour toutes les autres contraintes impliquant cette variable. Des algorithmes de filtrage sont alors lancés par les autres contraintes afin de déduire éventuellement d'autres réductions des domaines d'autres variables pour rétablir la consistance. On dit alors qu'une modification a été propagée. Ce mécanisme de propagation est réitéré jusqu'à ce que les techniques de consistance locale ne puissent plus réaliser aucune inférence, on parle dans ce cas d'un point fixe qui est atteint.

A l'issue d'une propagation de contraintes, trois situations sont possibles :

- Une contradiction a été identifiée (une variable a un domaine vide), il faut alors remonter d'un niveau dans l'arbre de recherche pour explorer d'autres branches,

- Le domaine d'au moins une variable n'est pas un singleton (comporte plus d'une valeur), il faut poursuivre l'exploration en profondeur de l'arbre de recherche,

- Le domaine de chaque variable est un singleton, une solution a été trouvée.

La propagation est rarement suffisante pour réduire chaque domaine de chaque variable à un singleton et donc trouver une solution. Une fois le point fixe atteint, il faut appliquer une énumération (choix de valeur pour chaque variable) à partir de l'espace de recherche résultant.

3.6 Recherche de solution

Rechercher une solution à un problème de PPC consiste à trouver une instanciation des variables de telle sorte que toute les contraintes du problème soient respectées. Le type de la solution recherchée pour un problème dépend fortement de la modélisation de celui-ci. Pour un problème satisfaction (sous forme de CSP), les solutions recherchées sont simplement des solutions qui satisfont les contraintes sans tenir compte du coût. Tandis que les solutions d'un problème d'optimisation sont celles qui minimisent (ou maximisent) une fonction de coût.

Lors de la phase de recherche, plusieurs méthodes ou techniques sont mises en oeuvre pour trouver d'éventuelles solutions :

- ***Méthodes de résolution*** : Ce sont les méthodes génériques de résolution qui peuvent être appliquées indépendamment des caractéristiques de l'instance

à résoudre. Elles peuvent être classifiées en trois classes : *méthodes complètes*, qui recherchent toutes les solutions, *méthodes incomplètes* (comme la recherche locale) qui se basent sur une recherche non systématique (aléatoire), et les *méthodes hybrides* qui combinent les deux autres méthodes.

- ***Heuristiques de sélection*** : Elles définissent les critères qui vont permettre de déterminer la prochaine variable à instancier ainsi que la valeur qui lui sera affectée.

3.6.1 Méthodes de résolution

Trouver une solution à un CSP est généralement un problème NP-complet. Cependant, diverses approches ont été développées pour tenter de résoudre des CSPs, notamment en utilisant les notions de parcours d'arbres pour explorer l'espace de recherche, et le renforcement de consistance pour réduire cet espace. Au cours de ce chapitre, nous mettons le point sur les principales méthodes de résolution complètes.

Méthodes complètes ou exactes

Les méthodes complètes sont des méthodes qui explorent totalement l'espace de recherche en parcourant exhaustivement l'ensemble des combinaisons possibles. Elles se basent sur une recherche arborescente et instancient séquentiellement les variables et effectuent des retours en arrière en cas de contradiction (échec). Ces méthodes sont caractérisées par aspect *constructif* puisqu'elles construisent une solution (affectation totale consistante) en partant d'une affectation partielle dont la consistance est vérifiée tout au long de l'exploration de l'arbre de recherche. Une deuxième caractéristique concerne la *complétude* de ces méthodes, dans le sens où l'on est certain de trouver une solution si le CSP est consistant. Cette propriété de complétude est très intéressante dans la mesure où elle garantie de trouver toutes les solutions du CSP ou prouver qu'il n'en a aucune. Le principal défaut de ces méthodes est que pour un espace de recherche de grande taille, elles passent trop de temps dans des branches qui ne conduisent pas forcément à une solution. Pour pallier à ce problème, des heuristiques peuvent être utilisées pour explorer les branches les plus prometteuses. Cependant, même en utilisant des heuristiques, il existe certains problèmes pour lesquels ce genre de méthodes ne se terminent pas en un temps raisonnable.

La méthode la plus triviale pour résoudre un CSP sur les domaines finis est appelée ***generate-and-test***. Cette méthode consiste à énumérer puis vérifier la consistance de toutes les affectations totales possibles jusqu'à en trouver une qui satisfasse la totalité des contraintes. La vérification de la consistance n'est effectuée que lorsque toutes les variables sont instanciées. Pour certains CSPs, le nombre des affectations totales peut être exponentiel, rendant cette méthode très peu efficace surtout si le problème est sur-contraint. L'algorithme récursif (Algo. 1) illustre cette méthode. Pour un CSP(X, D, C), l'algorithme tente d'étendre l'affectation partielle **A** (initialement vide) vers une affectation totale consistante.

Algorithm 1 Algorithme de la méthode *generate-and-test*

Generate_and_test(A,(X,D,C))

if pour toute variable $X_i \in X$, X_i est instanciée **then**
  %A est une affectation totale%
  if Est_consistante(A) **then**
    A est une solution
  else
    A n'est pas une solution
  end if
else
  Choisir une variable $X_i \in X$ / X_i est non instanciée
  for $V_j \in D(X_i)$ **do**
    Generate_and_test(A$\bigcup${$X_i \leftarrow V_j$},(X,D,C))
  end for
end if

Algorithm 2 Algorithme de la méthode *simple retour arrière*

Backtrack(A,(X,D,C))

if pour toute variable $X_i \in X$, X_i est instanciée **then**
  A est une solution
else
  Choisir une variable $X_i \in X$ / X_i est non instanciée
  for $V_j \in D(X_i)$ **do**
    A$\leftarrow$ A$\bigcup(X_i \leftarrow V_j)$
    if Est_consistante(A) **then**
      Backtrack(A,(X,D,C))
    else
      Annuler l'instanciation de X_i
    end if
  end for
end if

La seconde méthode, le ***simple retour arrière*** (*backtrack* en anglais), tente d'étendre progressivement une affectation partielle en instanciant à chaque étape une nouvelle variable et en vérifiant au fur et à mesure sa consistance. Si la nouvelle affectation partielle ainsi obtenue est inconsistante, la dernière instanciation de variable (X_i) est remise en cause. Un mécanisme de retour arrière est alors établi pour revenir dans un état antérieur du problème et instancier X_i avec une autre valeur et donc explorer une autre branche. Cette méthode, illustrée dans l'Algorithme 2, est à la base de tous les algorithmes complets de résolution des CSP. Malgré que le *simple retour arrière* est nettement plus efficace que le *generate-and-test*, il reste très coûteux et peu adapté aux problèmes de grande taille. En effet, la découverte redondante d'inconsistances locales dégrade les performances de cette méthode. Des algorithmes d'anticipation (algorithmes prospectifs) peuvent alors être utilisés pour tenter d'identifier les branches prometteuses, c'est à dire celles qui ne conduisent pas à l'inconsistance.

Le ***Forward Checking*** [73], est la technique d'anticipation d'échecs la plus simple à utiliser pendant la recherche de solutions. Après chaque instanciation de

Algorithm 3 Algorithme *Forward Checking*

```
1: FC(A,(X,D,C))
2: if pour toute variable X_i ∈ X, X_i est instanciée then
3:     A est une solution
4: else
5:     Choisir une variable X_i ∈ X / X_i est non instanciée
6:     for V_j ∈ D(X_i) do
7:         A←A⋃(X_i ← V_j)
8:         if Est_consistante(A) = false then
9:             % la consistance est vérifiée en réveillant que les contraintes impliquant X_i %
10:            D(X_i) ← D(X_i)/V_j
11:        end if
12:    end for
13:    if D(X_i) = ∅ then
14:        A ne peut pas être étendue par X_i
15:    else
16:        A peut être étendue par X_i
17:        Retour en (2)
18:    end if
19: end if
```

variable X_k, l'algorithme tente de renforcer la consistance d'arc entre X_k et toute autre variable non encore instanciée, et liée à X_k par une contrainte. L'objectif est de détecter une éventuelle inconsistance et donc d'anticiper de futures contradictions (échecs). Si une inconsistance est détectée, X_k est désinstanciée. En pratique le domaine de X_k est restauré à son état précédent. Si aucune inconsistance n'est détectée, l'algorithme continue d'étendre l'affectation partielle courante. L'Algorithme 3 illustre le mécanisme de *Forward Checking* ; (X, D, C) est le triplet définissant le CSP et **A** l'affectation courante.

Un autre algorithme, appelé ***Real Full Look Ahead*** [66], est également utilisé pour prévenir au plus tôt les infaisabilités futures et donc accélérer l'algorithme de *simple retour arrière*. Après chaque instanciation de variable, l'algorithme tente d'établir une consistance d'arc globale (contrairement au *Forward Checking*- voir Algo. 3) en ne se limitant pas à une seule propagation mais en renforçant la consistance pour l'ensemble des variables. Par rapport au *Forward Checking*, cet algorithme est plus coûteux en terme de renforcement de la consistance d'arc (beaucoup de calculs), mais est plus efficace en terme de réduction de domaines.

Plusieurs études et travaux ont été réalisés en se basant sur le mécanisme d'anticipation, montrant ainsi son intérêt et sa multidisciplinarité. Un aperçu de ces travaux est donné ci-après :

Song *et al.* [139] ont proposé une méthode de planification prédictive et un suivi temps réel des opérations de construction réalisées sur un terrain. Dans la méthode proposée, les données relatives aux opérations de construction sont constamment capturées à l'aide de capteurs spécifiques. Ces données sont ensuite introduites dans un simulateur pour mettre à jour automatiquement le modèle final. Cette mise à jour

permanente du modèle a permis de changer ou d'adapter avec précision la planification des tâches de construction. Des études de cas ont été effectuées pour démontrer la faisabilité du concept proposé.

Le problème de l'Acheteur Itinérant (Traveling Purchaser Problem ou TPP) est une généralisation du problème du Voyageur de Commerce, introduit pour la première fois par Ramesh [120]. Il consiste à trouver un tour qui part d'un dépôt, connectant un sous-ensemble de marchés, de telle sorte que l'ensemble des produits soit acheté, et que la somme des coûts du transport et ceux des achats des produits soit minimisée. Angelelli *et al.* [12] ont analysé une variante dynamique du problème, où les quantités des produits peuvent diminuer au cours du temps. Les quantités disponibles pour chaque produit sur chaque marché, ainsi que tout événement de consommation sont supposés connus à tout moment. Néanmoins, aucune information sur les événements futurs n'est disponible. Les auteurs ont proposé des heuristiques basées sur le ***look ahead***. Un premier groupe d'heuristiques comprend des approches simplifiées afin de choisir la prochaine visite sur la base de certains critères ne tenant compte que de l'un des deux coûts objectifs. Le second groupe comprend des heuristiques basées sur les coûts de transport, les coûts d'achats et une prévision de l'avenir.

Hellström *et al.* [78] ont proposé un algorithme basé sur le mécanisme de ***look ahead*** pour minimiser le temps de voyage et la consommation de carburant d'un poids lourd lors d'un trajet spécifique. Des informations relatives à la géométrie de la route sont extraites d'une base de données routière et un GPS. Ces informations anticipées sont utilisées pour optimiser la trajectoire de la vitesse en tenant compte des mesures du temps de voyage et de la consommation de carburant. L'algorithme proposé a été conçu et utilisé dans un schéma de commande prédictive pour commander constamment le régulateur de vitesse avec de nouvelles consignes. L'algorithme a été évalué dans un cas réel, et les résultats expérimentaux ont montré que la consommation de carburant a considérablement baissé.

Méthodes incomplètes ou méta-heuristiques

Les méthodes incomplètes explorent en partie l'espace de recherche (ne considèrent pas toutes les combinaisons) dans le but de trouver le plus vite possible une affectation totale "acceptable". Ces méthodes permettent de trouver rapidement de "bonnes" affectations de variables (qui violent peu ou pas de contraintes) mais ne peuvent prouver ni l'optimalité de l'affectation trouvée ni l'inexistance de solution d'un CSP sur-constraint (non consistant). Les méthodes incomplètes les plus utilisées sont celles basées sur des techniques de recherche locale pouvant être combinées avec les algorithmes génétiques ou stochastiques (méthodes tabou, recuit simulé, etc).

Méthodes hybrides

Bien que les méthodes complètes garantissent de trouver (si cela est possible) une solution, le processus de recherche peut être très long. Pour pallier à ce problème, plusieurs études et travaux ont été réalisés et ont montré que l'hybridation ou la fusion des techniques issues de la programmation par contraintes (méthodes

complètes) avec la recherche locale peut être plus efficace que leur utilisation séparée, et permet dans une certaine mesure de bénéficier des atouts respectifs de chacunes d'entre elles [89, 56, 117, 119].

3.6.2 Heuristiques de branchement

Les algorithmes de résolution décrits précédemment, procèdent à des choix de variables et de valeurs à chaque étape de la recherche. Bien qu'aucune précision n'a été donnée sur les critères de ces choix, la sélection de la prochaine variable à instancier ainsi que la sélection de la valeur qui lui sera affectée peuvent affecter considérablement l'efficacité de ces algorithmes. Plusieurs heuristiques de sélection ont été intégrées dans ces algorithmes pour déterminer l'ordre dans lequel les variables et les valeurs doivent être sélectionnées. Une heuristique est une règle qui fixe les critères de sélection des variables et leurs valeurs, donnant ainsi des indications sur les branches à explorer dans l'arbre de recherche. Une heuristique est statique si l'ordre de sélection est préalablement fixé lors d'une phase de pré-traitement ; elle est dynamique si cet ordre change au cours de la recherche. Généralement, les algorithmes de résolution combinés avec une heuristique de sélection fournissent plus rapidement une solution que ceux sans heuristique.

Sélection de variables

Différentes heuristiques d'ordonnancement de variables ont été développées. On distingue deux types d'heuristiques : les heuristiques statiques (*Statique Variables Ordering*) et les heuristiques dynamiques (*Dynamic Variables Ordering*). Pour les heuristiques statiques, qui ne dépendent pas de l'état courant de la recherche, nous pouvons citer *Max-Degree* [40] qui prend la valeur la plus connectée dans le graphe de contraintes original du problème. Un exemple d'heuristiques dynamiques, qui prennent en compte l'état courant de la recherche, la *Min-Domain* [74]. Dans cette heuristique, la prochaine variable à instancier est celle qui a le plus petit domaine parmi les variables non encore instanciées.

Smith [137] a également proposé d'autres heuristiques de sélection de variables comme *Fail-First* et *Succeed-First*. Le *Fail-First* consiste à sélectionner la variable réduisant le plus possible les domaines des autres variables non encore instanciées. Quant au *Succeed-First*, il sélectionne la variable réduisant le moins possible les domaines des autres variables non encore instanciées. Selon les expérimentations menées sur une chaîne de production automobile, le *Fail-First* obtient de meilleurs résultats pour le problème de *Car-Sequencing* (ordonnancement de véhicules) que le *Succeed-First*.

Sélection de valeurs

Une heuristique de sélection de valeurs détermine la prochaine valeur à donner à une variable en attente d'instanciation. Par exemple, l'heuristique *Min-Val* sélectionne la plus petite valeur du domaine, *Max-Val* sélectionne la plus grande, tandis que *Rand-Val* sélectionne aléatoirement une valeur. Ces heuristiques ont un impact considérable sur l'efficacité de la résolution puisqu'elles orientent la recherche vers des branches spécifiques.

3.7 Solveurs de contraintes

Ce paragraphe est dédié à la présentation de quelques outils de PPC utilisés pour résoudre des problèmes de satisfaction et d'optimisation. Ces outils peuvent être classés selon les algorithmes de recherche implémentés, le type de données qu'ils sont capables de traiter (entiers, réels), le langage de programmation dans lequel ils sont intégrés et enfin le type de leur license d'utilisation (libre ou commercial). Pour résoudre un problème à l'aide d'un solveur, on doit définir/spécifier l'ensemble des contraintes à considérer, les méthodes de résolutions et les stratégies de branchement (choix de variables et de valeurs), la résolution étant prise en charge par le solveur. Nous synthétisons dans ce qui suit des exemples cités par Malapert [100].

Le langage déclaratif de programmation logique *Prolog* [140] est à la base des premiers solveurs développés. Les extensions de *Prolog* les plus connus sont *Eclipse* [2] et *SICStus Prolog* [6] qui offrent des interfaces de développement basées sur les langages C/C++ et Java, et qui peuvent facilement communiquer avec les langages d'autres solveurs. Les langages C et C++, répandus pour leur rapidité de calculs et leur gestion optimisée de la mémoire, ont été utilisés pour développer des outils libres comme *Gecode* [133] et *Mistral* [9]. Des outils commerciaux ont été aussi entièrement écrits en C/C++ comme *CHIP* [1] et *Ilog CP Optimizer* [4]. Le langage Java a été utilisé pour développer les noyaux des solveurs libres comme *Choco* [7] et *Koalog* [5].

Au début de la thèse, notre choix s'est porté sur le solveur Gecode pour trois raisons :

- Il propose des mécanismes classiques et avancés pour la résolution des problèmes en PPC, et une grande facilité d'implémentation de contraintes,
- Il permet la résolution de contraintes réelles et discrètes. En effet, un travail consistant a été réalisé pour intégrer les variables réelles dans Gecode qui était originairement dédié aux contraintes discrètes,
- Son code source ouvert, ajouté à la portabilité du langage C++ sont très utiles pour intégrer *Gecode* dans d'autres systèmes (systèmes de visualisation et d'interaction 3D dans notre cas),
- Nous disposons dans notre équipe d'une très bonne expérience dans l'utilisation de *Gecode*.

3.8 Conclusion

Nous avons décrit dans ce chapitre les principes de la Programmation Par Contraintes (PPC). Cette approche déclarative permet de résoudre des problèmes combinatoires variés. Les utilisateurs décrivent leur problème en posant des contraintes sur les valeurs que peuvent prendre les différentes variables composant le problème. Un solveur de contraintes est alors chargé de calculer les solutions du problème. Le processus de résolution exacte de problèmes de satisfaction de contraintes permet de générer efficacement les solutions d'un problème grâce à la combinaison des

techniques de filtrage et d'algorithmes de recherche. Dans le chapitre suivant, nous décrivons le concept de la réalité virtuelle (RV) d'une façon générale et mettons l'accent sur les aspects relatifs à l'interaction et à l'évaluation des systèmes de RV.

4

Réalité virtuelle

4.1 Introduction

Il y a plus de 50 ans, le professionnel du cinéma Morton Heilig invente le ***Sensorama Simulator*** (machine cinématographique) qui fut le premier système à proposer une immersion multisensorielle. Le système développé était en effet capable de restituer du mouvement, de la couleur, du son stéréo, des odeurs, et même des vibrations. En 1960, Morton Heilig déposa un autre brevet pour un casque de visualisation, repris par Ivan Sutherland qui l'a enrichi par des images synthétiques fabriquées par ordinateur. Cette technique intéressait les militaires pour leurs simulateurs de vols recréant les conditions d'environnement de l'espace.

En 1985, Scott Fisher intégra au système un gant sensitif inventé par Thomas Zimmermann et Jaron Lanier, il a été commercialisé en 1987. En 1988, Scott Fisher créa, avec Elizabeth Wenzel, un système capable de gérer quatre sources sonores qui restent localisés quand l'utilisateur tourne la tête.

Dans ce chapitre, nous présentons les principaux concepts de la Réalité Virtuelle (RV), à savoir les caractéristiques et l'architecture générale d'un système de RV. Nous présentons également différentes applications ainsi qu'une typologie des périphériques et techniques d'interaction utilisés pour communiquer et agir dans les univers 3D. Finalement, nous décrivons les méthodes d'évaluation utilisées pour évaluer les systèmes de RV.

4.2 Définitions

Dans la littérature, différentes définitions de la **Réalité Virtuelle (RV)** ont été proposées. Nous retiendrons la définition suivante, qui fait l'objet d'un consensus dans la communauté :

La ***RV*** est un domaine scientifique et technique rassemblant l'ensemble des techniques et systèmes qui procurent à l'utilisateur le sentiment de pénétrer dans

des ***Environnements Virtuels (EVs)*** (univers synthétiques créés sur ordinateur), avec lesquels il pourra communiquer et interagir via des techniques et protocoles basés sur ses facultés naturelles d'expression, ses capacités d'action et de perception [32]. Ces techniques permettent à l'utilisateur d'éprouver physiquement un certain nombre de sensations (visuelles, auditives, haptiques, olfactives, etc.) et de pouvoir agir dans les EVs par des moyens d'action intuitifs définis par un ou plusieurs programmes informatiques (mouvements du corps, gestes, voix,etc.)[13].

4.3 Caractéristiques de systèmes de réalité virtuelle

La RV est caractérisée par le **concept des 3I** introduit par en 1993 par Burdea et Coiffet [32]. Appelé également *triangle de la réalité virtuelle*, ce concept regroupe les trois composantes de base de la RV : l'*Immersion*, l'*Interaction* et l'*Imagination*. Un deuxième triangle appelé *triangle de la réalité augmentée* a été introduit par Dubois [43] afin d'enrichir le concept des 3I. En effet, la **Réalité Augmentée (RA)** désigne les différentes techniques permettant d'intégrer, de façon réaliste, des entités numériques (objets 3D, images, sons, etc.) dans le monde réel [103], en offrant à l'utilisateur la possibilité d'être immergé dans cet environnement mixte. Comme le montre la Figure 4.1, les deux triangles se rejoignent en ce qui concerne la notion d'***interaction***. Les quatre composantes fondamentales d'un système de RV sont décrits ci-après.

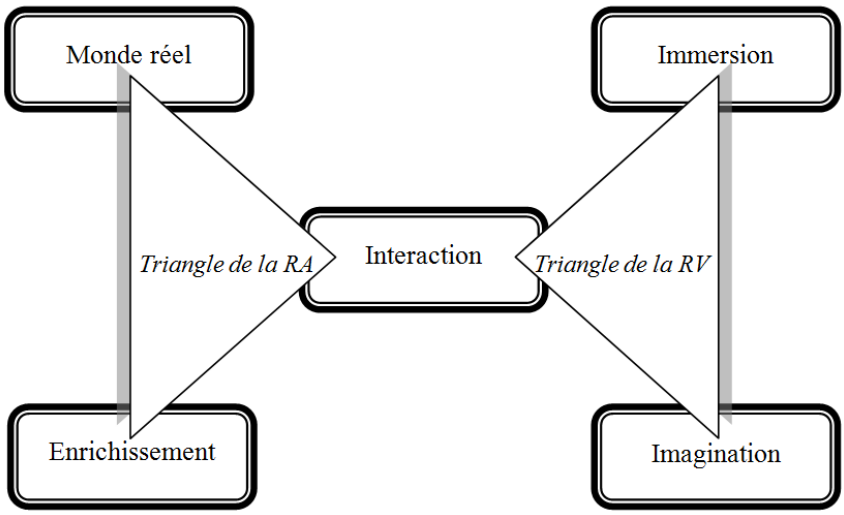


FIGURE 4.1 – *Triangles de la Réalité Virtuelle (RV) (D'après [32]) et de la Réalité Augmentée (RA) (D'après [43])*

4.3.1 Environnement virtuel

Le terme "environnement virtuel" (EV) a été introduit par les chercheurs du MIT (Massachusetts Institute of Technology) au début des années 90 comme synonyme de RV [76]. Il est considéré comme le lieu suggéré par la RV pour accueillir un ou plusieurs utilisateurs et leur permettre d'accomplir certaines tâches. Un EV est représenté par un modèle 3D de données réelles ou imaginaires que l'on peut visualiser et avec lesquelles on peut interagir en temps réel. Les EVs ne se contentent pas

de simuler l'aspect visuel d'objets mais ils possèdent également des propriétés physiques (masse, élasticité, fluidité, etc.) utilisées pour les faire interagir de manière réaliste.

4.3.2 Interaction temps réel

La notion de temps-réel est le fait que l'utilisateur ne perçoive pas de décalage temporel entre ses actions sur l'EV et la réponse sensorielle fournie par le système. Cette interaction dépend du degré d'accessibilité et de modification des paramètres de l'EV.

4.3.3 Immersion pseudo-naturelle

C'est la faculté d'un système de RV à immerger l'utilisateur dans l'univers synthétique (EV), en délivrant une information multi-sensorielle spatio-temporelle complète et cohérente [136]. La sensation de l'immersion est renforcée par la capacité de l'utilisateur à effectuer des actions dans l'EV et à percevoir sa position.

4.3.4 Interfaçage comportemental

L'interfaçage comportemental repose sur l'utilisation d'interfaces matérielles et de techniques d'interaction qui permettent à l'utilisateur d'interagir de manière intuitive avec l'EV par l'intermédiaire des canaux sensoriels [28]. L'objectif est de réduire au maximum les ressources cognitives liées à l'interfaçage pour permettre à l'utilisateur de se concentrer sur les tâches sensori-motrices (sélection, manipulation, etc.) et cognitives (recherche d'informations, etc.).

4.4 Notions d'immersion et de présence

L'objectif majeur de la RV est de procurer à l'utilisateur la sensation qu'il fait partie de l'EV avec lequel il communique. Concrètement, il s'agit d'occulter partiellement ou totalement la perception de l'humain pour qu'elle soit principalement exposée aux stimulis de l'EV. Cette ***immersion***, qui peut être totale ou partielle, repose sur une stimulation pertinente et cohérente de l'ensemble des canaux sensoriels (visuel, auditif, haptique et olfactif) [60].

Une immersion complète est obtenue au niveau visuel avec des visiocasques. Celle-ci doit être limitée dans le temps car l'être humain ne peut pas supporter les visiocasques plus de quelques minutes par heure. En effet, l'utilisation d'un visiocasque réduit considérablement le champ visuel de l'utilisateur et provoque des conflits vestibulo-occulaires liés à la présence de décalages temporels.

D'après Slater et al. [135], la notion d'*immersion* se rapporte à la description quantifiable de la technologie et est obtenue en remplaçant le plus grand nombre de sensations naturelles par leurs correspondances dans l'univers virtuel. L'*immersion* est alors une description objective de ce que n'importe quel système particulier fournit.

La ***présence*** est un état de conscience donnant la sensation à l'utilisateur d'être réellement présent dans un monde virtuel [77]. Witmer et Singer [146] définissent la *présence* comme l'expérience subjective d'être dans une place ou dans un environnement, même lorsqu'on est physiquement situé ailleurs. Elle se rapporte alors à remarquer l'environnement généré par ordinateur plutôt que le local physique réel. Selon les auteurs, les facteurs qui influent sur l'*immersion* sont donc l'isolement de l'environnement physique, la perception de soi dans l'EV, les modes naturels d'interaction et de contrôle, et la perception du mouvement propre. Bien que la sensation de présence soit subjective et qu'il soit difficile de mesurer le degré de présence d'un sujet dans un EV, une évaluation fiable, valide et objective doit être recherchée [102].

Pour donner le sentiment de présence aux utilisateurs dans un EV, les utilisateurs sont fréquemment représentés par des objets virtuels appelés *avatars*. Kadri et al. [90] évoquent l'importance de cette représentation par des *avatars* dont l'apparence influe sur la manipulation d'objets dans les EVs.

4.5 Notion d'interaction

L'***interaction*** peut être définie comme étant un langage de communication entre l'homme et la machine. Ce langage correspond à l'ensemble des actions/réactions réciproques entre l'homme et l'ordinateur par l'intermédiaire d'interfaces sensorielles, d'interfaces motrices et de techniques d'interactions. L'*interaction* est un aspect majeur de la RV. Elle doit être pseudo-naturelle, c'est-à-dire proche de l'interaction de l'homme avec son environnement réel. En effet, dans le monde réel, l'être humain a acquis, assimilé un certain nombre de gestes, de comportements sensori-moteurs voire cognitifs. Dans un EV, il va être spontanément amené à répéter ces gestes ou à les adapter à l'EV. Les travaux sur l'*interaction* en RV ont pour objectif de faciliter cette transposition, pour que celle-ci se fasse inconsciemment et sans grand effort mental [36].

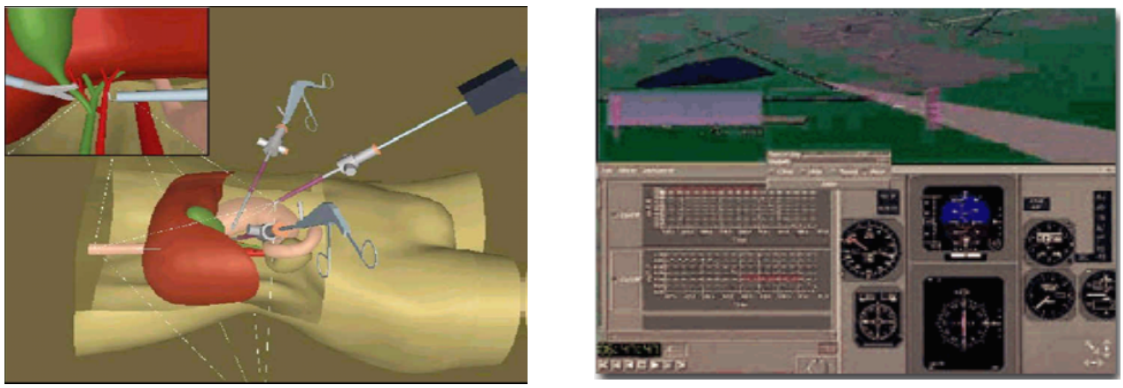
4.6 Exemples d'applications

Le champ d'application de la RV est extrêmement large du fait de sa pluridisciplinarité et des moyens qu'elle offre pour représenter et synthétiser des situations complexes tant réelles qu'imaginaires. voici une brève description des principaux domaines d'application :

– La *visualisation scientifique* : c'est sans doute le premier domaine d'application de la **RV**. De part l'immersion qu'elle apporte, elle permet une meilleure perception et une facilité d'interprétation de résultats de simulation scientifiques. Il est ainsi possible d'explorer visuellement des modèles mathématiques complexes ce qui permet de mieux les comprendre. Cela apporte également une grande souplesse dans la manipulation des données très utile par exemple dans l'étude de phénomènes physiques. Un élément important de la réalité virtuelle est en effet la possibilité d'intervenir en temps réel sur l'environnement.

– L'*aide à la décision* : la représentation tridimensionnelle est celle à laquelle on est habitué dans notre vie de tous les jours, c'est comme cela que l'on perçoit le monde qui nous entoure. Dans certains domaines de pointe, la complexité des données à prendre en compte dans la prise de décisions peut nécessiter des compétences de spécialistes que ne possèdent pas forcément les décideurs. C'est dans ce contexte qu'un environnement de réalité virtuelle peut apporter une vision synthétique plus facilement assimilable.

– La *formation* : le fait de pouvoir représenter de manière synthétique des notions complexes ou abstraites ainsi que la mise en jeu de tous les sens dans la perception des données présentées, apporte une très grande facilité d'assimilation. Cela peut être très utile dans la formation de spécialistes comme des pilotes ou des chirurgiens qui peuvent par exemple acquérir de l'expérience dans la réalisation de certaines opérations complexes. Cela peut donc être très utile pour éviter les risques réels liés à certaines formations.



(a) (b)

FIGURE 4.2 – Simulation d'opérations chirurgicales (a) et interface de simulateur de vol (b).

– L'*éducation* : la **RV** peut également jouer un rôle très important dans l'explication de la science aux plus jeunes en la rendant ainsi plus attrayante et moins abstraite. Elle peut également être un puissant outil culturel en permettant par exemple la visite virtuelle de lieux historiques, de musées, etc. Le coté ludique de la réalité virtuelle peut ainsi susciter plus d'intérêt chez les enfants.

– Les *loisirs, le divertissement, la vente* : c'est sans doute dans les domaines grands publics que la **RV** trouvera ses applications les plus importantes. Imaginez un jeu si réaliste que vous vous sentiriez comme dans un rêve où vous pourriez agir sur tout ce qui vous entoure avec la sensation d'en être le héro. Également l'imagination des voyages virtuels dans des pays exotiques un peu à la manière de ce que l'on voit dans le film "Avatar". L'apport de la **RV** s'est rapidement développé pour toucher par exemple le domaine de la vente. La possibilité de pouvoir montrer à un client le produit qu'il va acquérir sans que celui ci n'existe encore ou ne soit disponible sur place. Par exemple, proposer la visite de pavillons virtuels, la conduite d'une voiture.

FIGURE 4.3 – Visualisation 3D à l'échelle 1 :1 de prototype virtuel d'une voiture.

4.7 Architecture générale d'un système de réalité virtuelle

D'une façon générale, un *système de RV* se compose d'un moteur de RV et de périphériques d'entrée et de sortie (Fig. 4.4). Le rôle principal du moteur de RV est d'adapter correctement la restitution multisensorielle du monde virtuel à travers les périphériques de sortie en fonction des informations provenant des périphériques d'entrée. L'utilisateur est divisé en : (1) un *homme décisionnel* qui active, (2) l'*homme physique* qui commande le système, et (3) un *homme sensoriel* qui reçoit les retours sensoriels. Les interfaces utilisées dans un *système de RV* peuvent être classées en trois catégories selon le type d'informations qu'elles véhiculent.

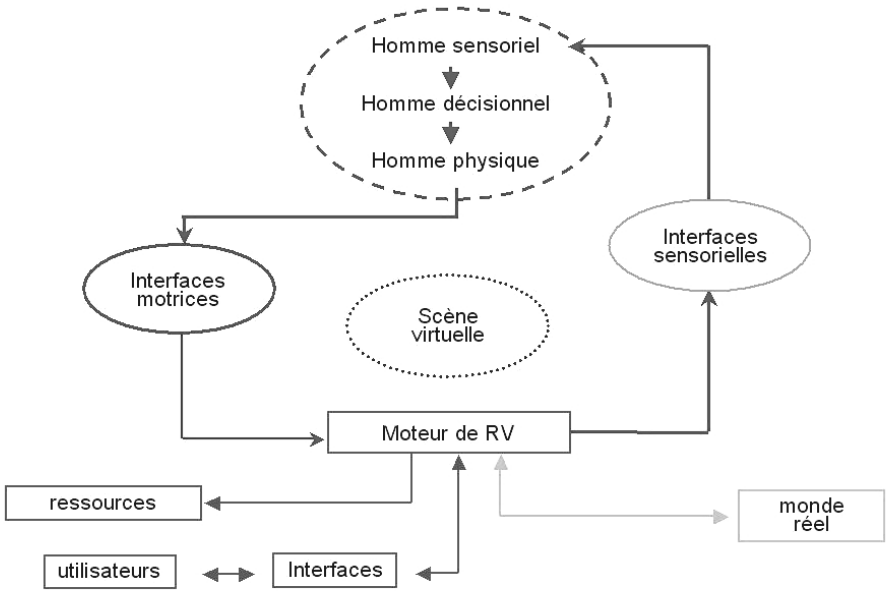


FIGURE 4.4 – *Architecture d'un système de RV (D'après [123]).*

4.7.1 Interfaces d'entrée

Les interfaces d'entrée ou motrices sont utilisées pour transmettre les intentions et les actions de l'utilisateur à l'EV et sont souvent classées suivant leur nombre de degrés de liberté [33], leur mode de résistance [150], ou les possibilités de séparation de leurs degrés de liberté [86].

4.7.2 Interfaces de sortie

Les interfaces de sortie ou sensorielles ont pour rôle de communiquer à l'utilisateur des informations provenant de l'environnement virtuel, souvent en réponse à ses propres actions. On distingue quatre grandes familles d'*interfaces sensorielles* : les interfaces visuelles, les interfaces tactiles, les interfaces à simulation de mouvement, les interfaces olfactives et les interfaces auditives [116].

4.7.3 Interfaces sensori-motrices

Les interfaces sensori-motrices à retour d'effort sont à la fois des interfaces d'entrée et des interfaces de sortie. Les *interfaces à retour d'effort* sont les principales interfaces sensori-motrices : elles transmettent à l'utilisateur des informations kinesthésiques provenant de l'EV.

4.8 Interfaces et périphériques d'interaction

4.8.1 Les interfaces visuelles

Les systèmes de RV sont basés sur différentes interfaces et périphériques matériels hautement expressifs, principalement liés aux interfaces visuelles. Au cours des dernières années, plusieurs dispositifs visuels ont été développés pour les applications de RV. Ces dispositifs visuels peuvent être classés selon leurs degrés d'immersion :

- Les *configurations non immersives* permettent la visualisation des entités synthétiques dans le monde réel ou dans un monde virtuel. En utilisant un simple dispositif d'affichage (par exemple un écran), l'utilisateur visualise alors le monde à travers une "fenêtre" représentée par l'écran [82] ;

- Les *configurations semi-immersives* dans ce cas, l'EV de grande dimension ne recouvre pas la totalité du champ visuel de l'utilisateur. Notons que certaines configurations immersives n'impliquent pas un recouvrement total du champ visuel de l'utilisateur (mur, workbench) ;

- Les *configurations immersives* l'objectif est d'immerger l'utilisateur dans l'EV par le biais d'un affichage stéréoscopique, ou d'immerger des entités (images, objets 3D, etc.) virtuelles dans le monde réel, via l'utilisation de visio-casques (ou Head-Mounted Displays : HMDs).

Une deuxième classification distingue les dispositifs visuels de RV selon leurs caractéristiques matérielles : (1) à base de moniteurs et d'écrans (configuration *desktop*), (2) à base de projection sur grand écran et (3) les visio-casques.

Les configurations non immersives

Les *configurations non-immersives* sont basées sur des écrans de plus ou moins grande dimension. Leurs coûts et leurs facilités de mise en place les rendent plus accessibles que les configurations immersives.

Sur un écran classique, la scène virtuelle est affichée avec les mêmes caractéristiques de profondeur, de couleur, de texture, d'éclairage et d'ombrage que dans un visio-casque. Actuellement, la mise sur le marché d'écrans plats (LCD, plasma ou OLED) de grandes tailles, à des coûts raisonnables, permet d'envisager des configurations non-immersives pour des applications diverses de RV.

Plusieurs types de configurations non-immersives existent pour lesquelles les systèmes d'affichage utilisent :

- Des *écrans CRT classiques* [11] ou plus récemment des *écrans de grandes dimensions*(LCD ou plasma) ;
- Des *écrans portables* comme les écrans de téléphones mobiles [121, 104] ;
- Des *systèmes de vidéoprojection* sur des surfaces variées ;
- Des *solutions mixtes* [21].

L'utilisation des écrans de grandes dimensions ou des vidéoprojecteurs dans des applications de RV, présente plusieurs avantages :

- L'affichage est partageable par plusieurs utilisateurs ce qui est bien utile pour des applications multi-utilisateurs.
- L'image affichée sur l'écran est indépendante de la position de l'utilisateur.
- En comparaison avec les visiocasques, le coût des vidéoprojecteurs ou des écrans plats est plus faible.

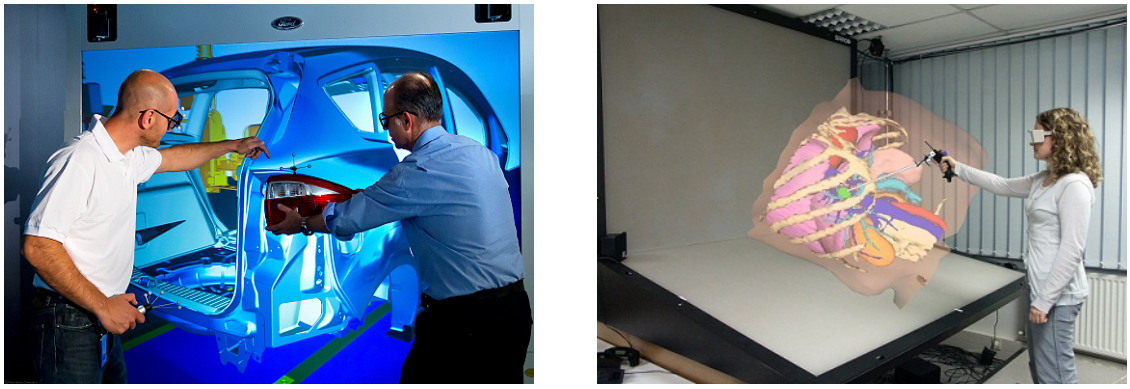
Les configurations semi-immersives

Des plate-formes de RV plus ou moins immersives avec ou sans dispositif multi-sensoriels sont installées dans les organismes de recherche et les entreprises françaises [61],

Les workbenchs ou plans de travail Ils représentent un plan de travail réel et permettent une manipulation naturelle et simple d'objets virtuels [115, 94, 143, 96]. Ils sont équipés d'un ou deux écrans stéréoscopiques à affichage rétroprojeté et nécessitent l'utilisation de lunettes spéciales (Fig. 4.5.b).

Les murs Dans ce type de configurations, les images sont rétro-projetées sur un écran de grande dimension. Une visualisation stéréoscopique est généralement utilisée afin de décaler les images pour l'œil gauche de celles pour l'œil droit. La visualisation stéréoscopique peut être obtenue par exemple via l'utilisation de lunettes passives.

Les écrans stéréoscopiques Une tendance vise actuellement à utiliser des écrans LCD ou plasma de grandes dimensions, pour des applications impliquant la manipulation d'objets. Ces configurations visuelles présentent une meilleure résolution et un contraste plus élevé que les configurations stéréoscopiques rétro-projetées. En outre, elles sont facilement transportables, et sont moins coûteuses.



(a) (b)

FIGURE 4.5 – Exemples de configurations visuelles semi-immersives : (a) mur et (b) plan de travail ou workbench.

Les configurations immersives

Les *configurations immersives* sont basées sur différentes types d'interfaces visuelles à base de dispositifs de rétro-projection ou de visio-casques.

Les dômes immersifs Il s'agit des écrans de surface hémi-sphérique sur lesquels est projeté l'environnement virtuel (**EV**). Cette surface de projection intensifie l'immersion visuelle grâce à un recouvrement important du champ visuel de l'utilisateur [3].

Les salles immersives Elles sont constituées de plusieurs murs de projection et sont généralement utilisées pour permettre l'observation et l'exploration d'objets de grandes dimensions tels que des bâtiments. Elles permettent à un nombre important d'utilisateurs de les visualiser simultanément une scène virtuelle

Les visio-cubes Ils se présentent sous la forme d'une enceinte cubique possédant quatre à six écrans orthogonaux de grande dimension. L'affichage stéréoscopique est rétroprojeté. Dans la configuration à 6 écrans, le champ visuel de l'utilisateur est totalement recouvert et celui-ci n'a plus aucun repère issu du monde réel. Le $CAVE^{TM}$[38] et le $SAScube^{TM}$[132, 106], représentent probablement les plus connues des solutions de type visiocube. Ce sont aussi les plus chères et les plus lourdes à mettre en œuvre et à maintenir.
En 1999, un Reality Center^{TM} et par la suite en 2001, le SASCube^{TM}, visiocube bas-coût et transportable, sont installés à l'Ingéniérium de Laval. Puis, en 2002 le MoVE^{TM}, visiocube reconfigurable est installé dans les locaux de PSA Peugeot Citroën à Paris. Ce système est maintenant installé à l'Institut Image de Châlon-sur-Saône.

Les visio-casques Également appelés ***casques immersifs*** ou ***HMDs***, permettent, via un affichage stéréoscopique des images et un suivi en temps réel de la position et de l'orientation de la tête de l'utilisateur, une immersion (théorique) visuelle totale de celui-ci (Fig. 4.6.a). Son usage est strictement individuel, ce qui nécessite plusieurs casques pour des dispositifs multi-utilisateurs et/ou collaboratifs.

(a) (b)

FIGURE 4.6 – *Exemples de visio-casques : (a) visio-casque Sony de dernière génération et (b) visio-casque semi-transparent (D'après [37]).*

4.8.2 Les interfaces haptiques

La *perception haptique* est basée sur un grand nombre d'informations provenant d'un ensemble de récepteurs sensoriels situés au niveau des articulations, des tendons, des muscles et de la peau [31]. Elle peut être décomposée en : (1) perception des contours et de la forme, (2) perception de la compliance (forces internes), (3) perception du poids (forces externes), (4) perception de la température et (5) perception de l'état de surface. Différents critères de classification des interfaces haptiques ont été proposés, la Figure 4.7 présente une classification basée sur le type d'architecture mécanique du dispositif utilisé pour appliquer les forces (interfaces à réaction externe ou interne).

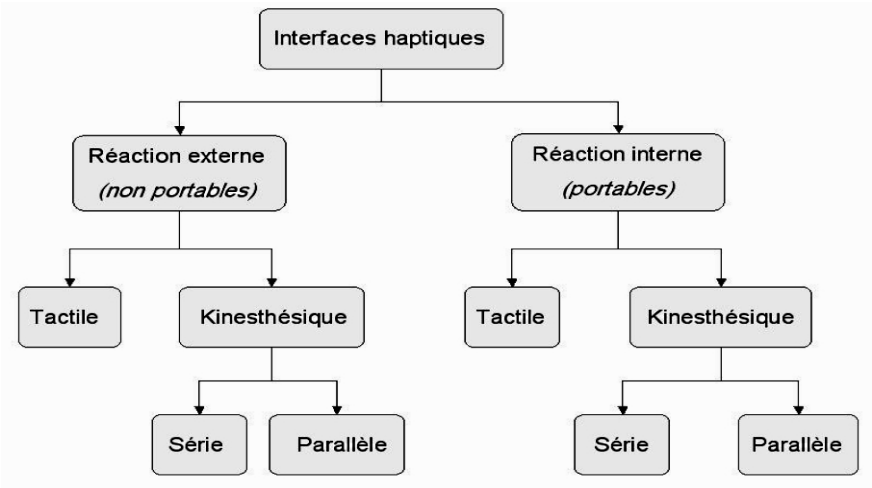


FIGURE 4.7 – Classification des interfaces haptiques.

La recherche et le développement de nouvelles *interfaces à retour d'effort* pour la RV est un domaine toujours très actif [20, 62, 28, 27]. Outre, les secteurs de la téléopération, de la robotique, de la simulation chirurgicale, qui utilisent depuis plus de 30 ans, le retour haptique, d'autres disciplines comme la bio-mécanique, la conception, l'architecture et le design tendent à l'intégrer de plus en plus dans des EVs.

Des études expérimentales ont montré que la sensation haptique prodiguait une information essentielle pendant des tâches de manipulation, en améliorant la perception des objets (taille, forme). En effet, une augmentation importante de la performance humaine a été démontrée en terme de vitesse et de précision avec l'utilisation du retour haptique [111, 124, 125].

4.9 Techniques d'interaction 3D

4.9.1 Définitions

Considérée comme un élément majeur dans toute application de RV, l'*interaction 3D* est définie comme une action menée par l'utilisateur afin de modifier l'état d'un objet ou le cours des événements dans EV. Une *technique d'interaction* est une méthode, un scénario d'utilisation d'une interface matérielle, permettant à l'utilisateur d'accomplir une tâche précise. Comme démontré par Dragicevic [42], une même technique d'interaction peut être utilisée avec des interfaces matérielles différentes afin d'accomplir des tâche différentes. L'*interaction 3D* nécessite l'utilisation (1) d'une *interface motrice*, (2) d'une *interface sensorielle* et la définition (3) d'un *scénario* d'utilisation de ces interfaces (Fig. 4.8).

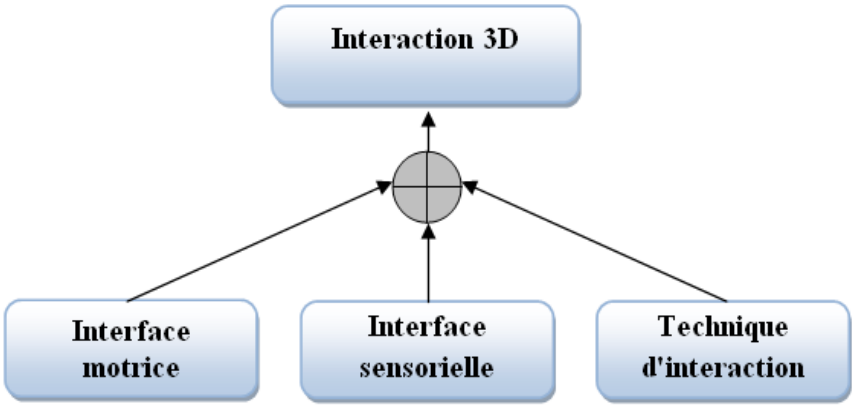


FIGURE 4.8 – Contexte de l'interaction 3D.

D'après Dragicevic [42], un *paradigme d'interaction* est défini par un ensemble cohérent de techniques d'interaction cumulables qui reposent sur les mêmes principes. Par exemple, le paradigme d'interaction WIMP (Windows, Icons, Menus, Pointers) regroupe des techniques liées à la manipulation de fenêtres, d'icônes et de menus comme le pointage et le glisser-déposer etc.

Fuchs et BurkhardtEnfin [59] définissent une ***métaphore d'interaction*** comme l'image symbolique d'une action ou d'une perception utilisée pour réaliser une

tâche. Il s'agit de la transposition d'un objet ou d'un concept réel dans le monde virtuel. Ainsi, il est préférable d'avoir recours à une métaphore d'interaction uniquement lorsqu'il n'est pas possible d'exploiter directement une interaction naturelle. Par exemple, la métaphore de la main virtuelle fait référence à la transposition de la main de l'utilisateur dans l'environnement virtuel.

4.9.2 Classification des techniques d'interaction 3D

D'après Bowman [28], les *techniques d'interaction 3D* peuvent être classées en quatre catégories : (1) la sélection, (2) la manipulation, (3) la navigation et (4) le contrôle d'application. Différentes taxinomies ont été proposées dans le cadre de la conception et de l'analyse de techniques d'interaction (Fig. 4.9).

La sélection

La sélection d'une ou plusieurs entités virtuelles, en vue d'une interaction spécifique, a deux composantes : (1) l'indication ou la désignation du ou des entités virtuelles et (2) la confirmation de la sélection via une interface motrice dédiée. Nous observons sur la figure les différentes possibilités offertes pour le développement de techniques de sélection (pareil pour manipulation et selection). La Figure 4.9.a présente la taxonomie proposée par Bowman [24].

La manipulation

La manipulation désigne généralement la transformation géométrique (*translation* et/ou la *la rotation*) d'un objet virtuel. Cependant, la manipulation d'un objet virtuel est plus étendue que celle d'un objet réel (modification de la taille, de la couleur, etc.). Comme pour la sélection, Bowman [24] a proposé une taxonomie pour la manipulation d'objets virtuels (Fig. 4.9.b).

La navigation

La navigation correspond au changement du point de vue (caméra virtuelle) de l'utilisateur dans l'EV sans déplacement physique ou déplacement limité de celui-ci. Une tâche de navigation est divisée en deux composantes complémentaires : (1) le déplacement (*travel*) et (2) la recherche d'un itinéraire (*way-finding*). Notons que la *locomotion* est considérée comme une technique de déplacement dans laquelle l'utilisateur fait de l'effort via l'utilisation d'une interface motrice.

Le contrôle d'application

Le contrôle d'application consiste à la modification d'un ou plusieurs paramètres de l'application, au cours d'une tâche ou d'une activité en EV (généralement en situation immersive ou pseudo-immersive). Il fait souvent appel à l'utilisation de menus (2D ou 3D) ou des interfaces motrices. Une taxonomie, illustrée sur la Figure 4.10, a été proposé par Grosjean [68].

- ***La commande vocale***
  La commande vocale utilise un moteur de *reconnaissance vocale* pour commander un système. Cette approche est attrayante car elle est naturelle, elle

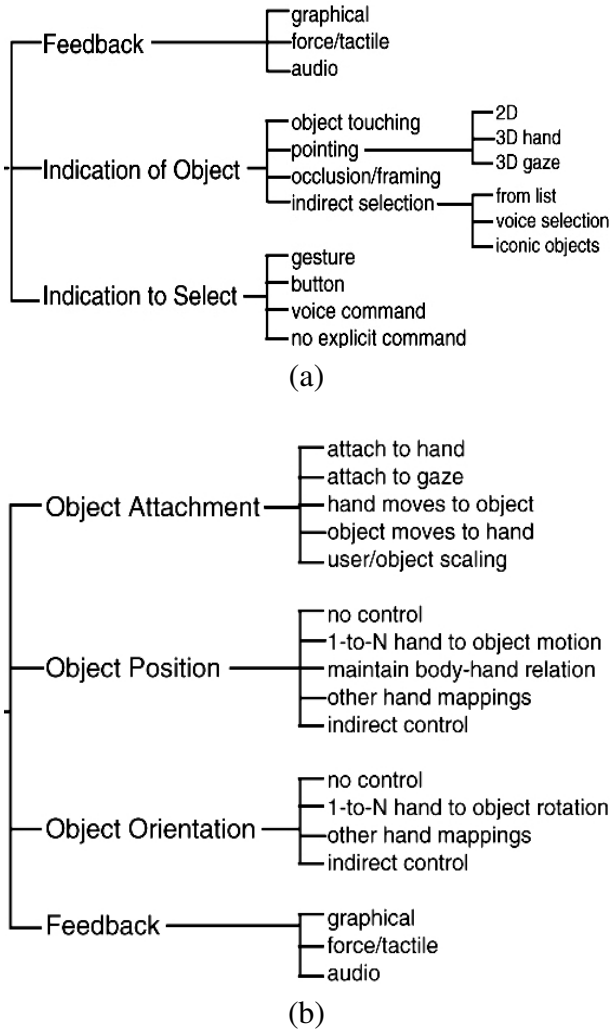


FIGURE 4.9 – Taxonomies proposées par Bowman pour : (a) la sélection et (b) la manipulation d'objets virtuels (D'après [24]).

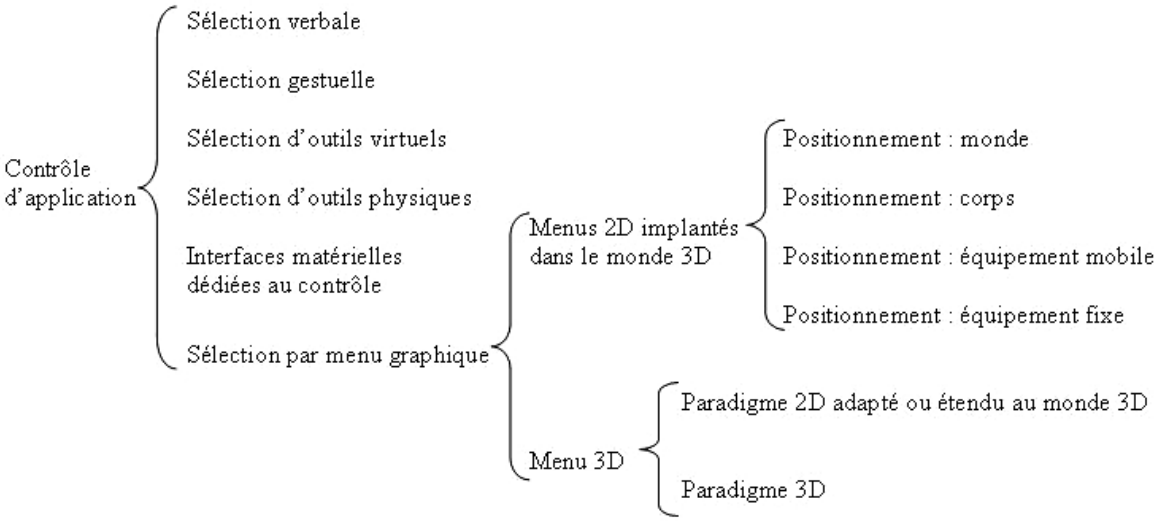


FIGURE 4.10 – Taxonomie proposée pour le *contrôle d'application* (D'après [68]).

laisse les mains libres et ne requiert pas l'affichage de menus, ni l'utilisation d'interfaces matérielles qui peuvent être incompatibles avec la réalisation de la tâche. Cependant, elle est la moins répandue car elle nécessite un apprentissage qui peut devenir long si le nombre de commandes est important. Le moteur de reconnaissance est l'élément le plus critique car de nombreux paramètres influencent le taux de réussite et la vitesse d'acquisition des commandes.

- ***La commande gestuelle***
  La commande gestuelle est associée à un geste (mouvement dynamique) ou à une posture (configuration statique) de la main. Deux technologies sont utilisées pour analyser les gestes : les systèmes optiques à l'aide de caméras et l'utilisation de gants de données. Cette méthode n'est pas très naturelle, et l'utilisateur peut avoir des difficultés à mémoriser un grand nombre de commandes.

4.9.3 Interaction multimodale

Dans un dialogue homme-homme, la communication est naturellement basée sur la multimodalité. En effet, le geste est un deuxième mode de communication complémentaire au premier mode qu'est la communication orale. La multimodalité dans l'interaction homme-machine se base sur l'utilisation complémentaire de plusieurs modes de communication en entrée ou en sortie. Lors de son interaction avec une machine, l'homme utilise différents moyens ou modalités d'interaction (langage écrit, la voix, le geste, etc.) selon la tâche à accomplir [113][85][112].

Selon Nigay et al. [107], la multimodalité permet de faciliter l'interaction avec des systèmes interactifs et améliorer leur utilisabilité. Les auteurs proposent trois critères pour évaluer une interaction multimodale :

- le caractère naturel de l'interaction,
- la robustesse de l'interaction,
- la flexibilité de l'interaction.

L'interaction multimodale a bien sur été largement utilisée dans le domaine de RV dans différents contexte théoriques et applicatifs [126][23][35][72]. Dans la plupart des cas, la multimodalité a été utilisée en sortie et dans le contexte de l'interaction haptique. En effet, des retours visuels, auditifs ou vitro-tactiles ont été proposés pour compléter (redondance informative) ou remplacer (substitution sensorielle) les informations kinesthésiques (retour d'effort) lors de manipulation d'objets virtuels.

4.9.4 Les guides virtuels

Le concept de guides virtuels (*virtual fixtures*) a été introduit par Rosenberg dans le contexte de la téléprésence [127][128]. L'objectif était d'améliorer la performance d'un opérateur humain, utilisant un robot réel pour effectuer des tâches d'insertion, via des informations sensorielles (haptiques et/ou auditives). Les résultats ont montré que l'utilisation des *virtual fixtures* a permis d'améliorer jusqu'à 70% les performances de l'opérateur. D'une façon générale, les guides virtuels peuvent

être définis comme des informations abstraites pour aider les utilisateurs à accomplir des tâches spécifiques. Ces informations permettent par exemple de guider les mouvements des utilisateurs dans l'EV ou leur interdir l'accès à certaines zones, etc.

En 2000, Otmane a proposé plusieurs méthodes de modélisation et de construction des guides virtuels pouvant être utilisées pour assister les utilisateurs durant une grande variété de tâches [109]. Les méthodes de création de guides virtuels se basent sur (1) différentes types de surfaces (plan, disque, etc.), (2) des volumes fermés (cônes, sphères, etc.), (3) des volumes ouverts (sous forme de tubes) et (4) des courbes. La Figure 4.11 montre un exemple d'utilisation des guides visuels lors de la manipulation d'un robot.

Veyret et al. [98] ont utilisé des techniques de réalité augmentée pour transmettre aux utilisateurs les informations issues du guide virtuel utilisé dans le cadre de la visite d'un aquarium marin. Ce guide est capable de s'adapter dynamiquement aux différentes contraintes de façon autonome grâce à la mise en place d'une boucle perception/décision/action. En effet, et en se basant sur une représentation de l'environnement et de connaissances structurées, le guide sélectionne et présente en temps réel des informations adaptées au visiteur.

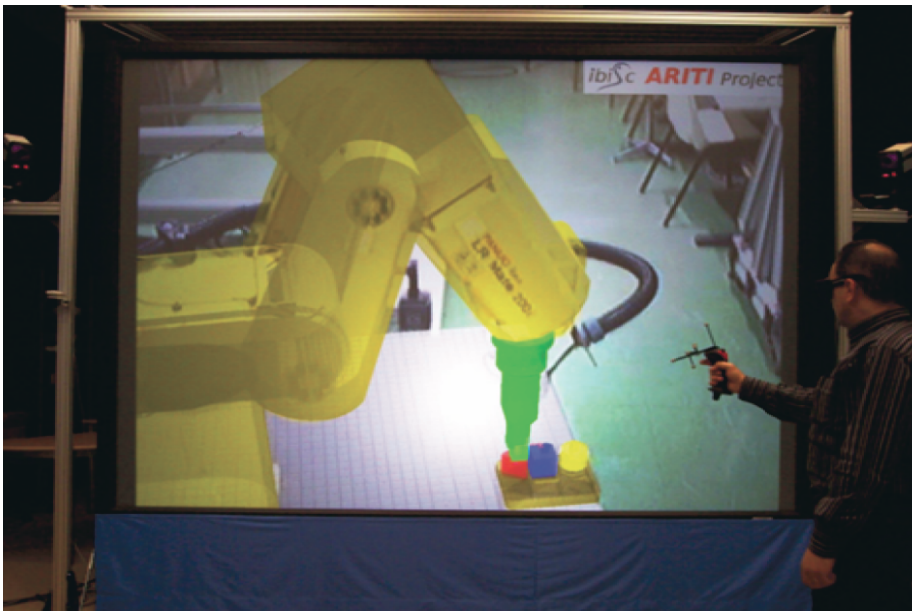


FIGURE 4.11 – Utilisation des guides visuels pour la manipulation du robot *FANUC LR-MATE 200i* (D'après [110]).

4.10 Méthodes d'évaluation

De nos jours, l'évaluation devient un concept important permettant de perfectionner le processus de conception via l'analyse de la performance humaine. D'après Bowman et al. [28], l'objectif principal de l'évaluation d'une technique d'interaction est l'identification d'éventuels problèmes ou contraintes fonctionnelles et ergonomiques. Ils ont mis en évidence que les processus de conception et d'évaluation

d'une technique d'interaction sont itératifs et ne s'achèvent que lorsque la technique répond aux objectifs d'utilisation.

4.10.1 Techniques d'évaluation

Différentes méthodes pour l'évaluation des tâches dans les EV et des techniques d'interaction ont été formalisées et développées. La Figure 4.12 propose une classification de certaines d'entre elles [28].

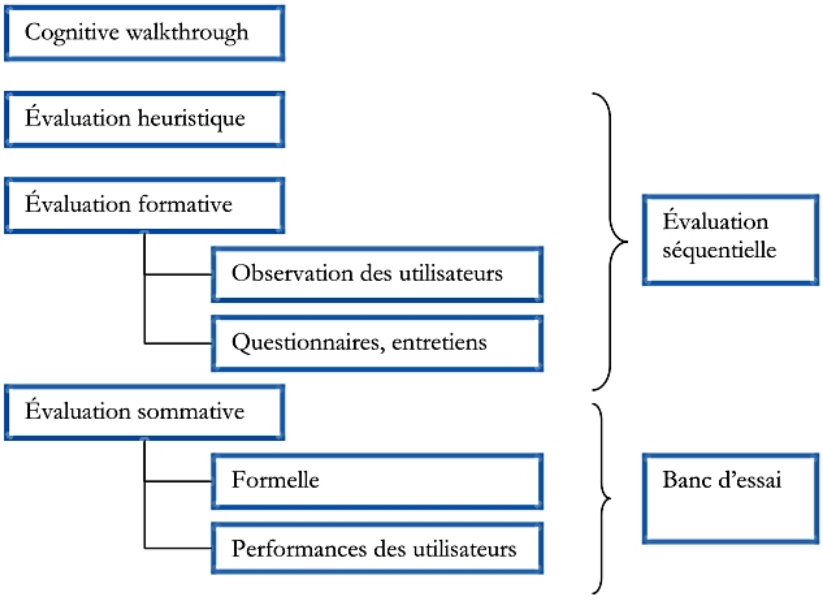


FIGURE 4.12 – *Classification des techniques d'évaluation (D'après [28]).*

Cognitive walkthrough

Il s'agit d'une technique d'évaluation où chaque tâche principale du système à évaluer est analysée par des experts en utilisant des questionnaires détaillés.

Évaluation heuristique

L'*évaluation heuristique* est basée sur le savoir faire et l'expérience acquise par des experts en conception d'interfaces ou de techniques d'interaction 3D. Plusieurs experts du domaine évaluent une interface correspondant à leur expertise en la comparant aux règles de conception prédéfinies [52]. Une liste d'heuristiques concerne, entre autres, la limitation de la charge cognitive de l'utilisateur, la visibilité de l'état du système, la prévention d'erreurs, etc. En conséquence, l'utilisateur final n'est pas impliqué dans ce type d'évaluation.

Évaluation formative

L'*évaluation formative* est une méthode empirique consistant à placer l'utilisateur final dans des conditions de réalisation de tâches avec la technique à évaluer [79]. L'objectif est d'identifier des problèmes fonctionnels, ergonomiques ou de performance liés à la technique employée. Une évaluation formative peut être

informelle, aboutissant généralement à des résultats qualitatifs (réactions générales de l'utilisateur). Elle peut être également formelle en conduisant des résultats qualitatifs et quantitatifs (mesures de précision, temps d'accomplissement de la tâche, etc.). Deux modes d'évaluation formative sont généralement utilisés : (1) un mode en ligne relatif à l'observation des utilisateurs (la présence simultanée de l'utilisateur et de l'expert est requise) et (2) un mode hors ligne relatif à des questionnaires, entretiens, enregistrements audio et/ou vidéo.

Évaluation sommative

L'*évaluation sommative* consiste à comparer les performances de plusieurs techniques dans le même contexte. Elle permet de mesurer et de comparer la productivité de l'utilisateur réalisant une tâche donnée avec différentes techniques et implique des utilisateurs finaux qui doivent effectuer une tâche scénarisée pendant laquelle des informations à la fois quantitatives et qualitatives sont enregistrées. Ces études comparatives sont généralement menées dans les phases finales de développement d'une technique. On peut donner comme exemple l'étude de la technique la plus adéquate pour l'accomplissement d'une tâche spécifique. Dans ce cas, le contexte d'évaluation sommative est formel (protocole expérimental strict, sélection des utilisateurs etc.). Les résultats des expériences formelles sont habituellement quantitatifs et sont analysés statistiquement [29, 81].

Approche par bancs d'essais

Proposée par Bowman et Hodges [26], l'objectif d'un *banc d'essais* est d'évaluer indépendamment chaque aspect d'une technique selon plusieurs indicateurs de performance afin de fournir des résultats génériques et réutilisables. Il est important qu'un grand nombre de sujets prenne part aux évaluations. Si les techniques d'interaction à évaluer sont nombreuses, la quantité d'essais par personne peut rapidement devenir conséquente. De ce fait, les différentes techniques d'interaction évaluées sont souvent considérées comme des variables inter-sujets (between-subjects), c'est-à-dire que chaque sujet n'utilise qu'une seule technique d'interaction, alors que les autres paramètres sont des variables intra-sujets (within-subjects).

Des exemples concrets de mise en application d'évaluations basées sur des bancs d'essais sont décrits pour des tâches de navigation [25] et pour des tâches de sélection et manipulation [26].

Approche séquentielle

Gabbard et al. [64] ont proposé cette approche basée sur des itérations successives des principales méthodes décrites précédemment. Le principe de cette approche est d'évaluer puis d'améliorer de manière itérative la technique étudiée. La progression séquentielle est conçue de manière à guider le concepteur dans l'élaboration d'une technique d'interaction optimale. Le processus inclut à la fois des méthodes d'évaluation réalisées par des experts de l'interaction 3D et par des utilisateurs finaux.

L'*approche séquentielle* se base sur des tâches précises, et doit par conséquent intervenir une fois que les spécificités de l'application sont implémentées. Les éva-

luations heuristiques et formatrices d'un prototype gênèrent des améliorations à prendre en compte pour la suite du processus. Quand à la l'évaluation comparative, elle peut être conduite lorsque le choix d'une solution n'est pas évident a priori (par exemple le choix de la technique d'interaction la plus appropriée pour une tâche donnée).

4.10.2 Mesures de performance

L'*efficacité* d'une technique d'interaction ou d'un paradigme expérimental est évaluée par des *indices ou mesures* permettant de quantifier les performances de l'utilisateur lors d'une tâche expérimentale. On distingue (1) les indicateurs *objectifs* et (2) les indicateurs *subjectifs*.

Mesures objectives

Les *mesures objectives* sont généralement des grandeurs physiques mesurées lors de l'évaluation, telles que le temps de réalisation, la précision du résultat, le nombre d'échecs avant réussite, etc. Le temps de réalisation et la précision sont les indicateurs les plus couramment utilisés. Cependant, Bowman et al. mettent en évidence que la stratégie utilisée par l'utilisateur pour accomplir la tâche, peut entraîner une variabilité importante des résultats d'un utilisateur à un autre [28]. Par conséquent il est souvent important de contraindre l'utilisateur à utiliser une stratégie particulière afin d'obtenir un contrôle optimal sur les résultats de l'évaluation. Néanmoins, il peut être intéressant parfois de donner les deux consignes à l'utilisateur afin d'étudier la stratégie qu'il mettra spontanément en oeuvre dans un cas de figure précis.

Les *mesures physiologiques* permettent également de recueillir des mesures objectives. Des capteurs sont placés sur le sujet pour enregistrer ses réactions dans l'EV et son état émotionnel. Le plus souvent, les données enregistrées sont la fréquence cardiaque, la conductivité et la température de la peau [102]. D'autres données peuvent être également enregistrées comme par exemple, la respiration, le suivi des yeux [15], etc.

Mesures subjectives

Les *mesures subjectives* pour l'évaluation en EV, peuvent porter sur différents aspects. Une grande partie des travaux ont mis l'accent sur la mesure de la *sensation de présence*. Certaines *mesures subjectives* concernent en particulier, les environnements virtuels collaboratifs (EVCs), telles la *co-présence*, la *présence sociale*, l'*effort collaboratif* et l'*awareness*. Ces indices sont souvent révélés et mesurés via des questionnaires ou des entretiens post-expérimentation avec les utilisateurs.

Les questionnaires permettent de collecter des données démographiques (âge, sexe, expérience, etc.) et subjectives (charge de travail, commentaires, préférences, impressions, etc.). Un *entretien* permet de collecter des informations en discutant directement avec les utilisateurs après une évaluation formatrice ou comparative. L'*entretien* est moins rigoureux mais plus flexible qu'un questionnaire car il permet d'approfondir certains points soulevés lors de le discussion.

4.11 Conclusion

Après avoir présenté les aspects liés à la programmation par contraintes (PPC) dans le chapitre précédent, ce chapitre a été consacré à la description du deuxième concept utilisé pour la résolution des problèmes d'agencements interactifs à savoir la réalité virtuelle. Nous avons mis l'accent sur les périphériques et techniques d'interaction 3D ainsi que sur les méthodes d'évaluation des systèmes de RV. En effet, le système de résolution proposé dans le chapitre 6 se base sur une interaction continue avec l'utilisateur qui sélectionne les objets et contraintes et commande le solveur à travers des scénarios d'interaction spécifiques. Une série d'expérimentation est décrite dans la troisième partie de cette thèse visant à évaluer le système proposé et l'interaction avec celui-ci. Après avoir présenté la problématique de l'agencement (chapitre 2), la PPC (chapitre 3) et les concepts, outils et techniques de la RV (dans ce chapitre), nous décrivons en détail dans le chapitre suivant le contexte et la problématique d'agencement étudiée. Une spécification et une formulation du problème seront également données.

II

Description du problème et approche proposée

5

Identification et formulation du problème

5.1 Introduction

Le développement des méthodes de résolution des problèmes d'agencement est actuellement en plein essor et suscite l'intérêt de nombreux chercheurs et industriels. Plusieurs variétés de ces problèmes font l'objet de diverses applications, allant de l'agencement d'intérieurs à l'assemblage des composants mécaniques ou électroniques, en passant par les problèmes d'aménagement de bateaux et de véhicules militaires. Les problèmes d'agencement sont généralement liés à des cas industriels spécifiques (impliquant des exigences conceptuelles spécifiques) ce qui ne permet pas la standardisation de leur modélisation.

La problématique traitée dans le cadre cette thèse s'est principalement inspirée d'un problème concret proposé par *Thales Communications & Security*. En effet, l'entreprise souhaite s'investir dans le développement d'un système informatique permettant d'aménager des Véhicules de l'Avant Blindés (VAB) et/ou des shelters (Espaces techniques mobiles) et ainsi réduire les coûts liés à leurs prototypages.

Comme c'est le cas pour n'importe quel problème d'agencement, les concepteurs ont leurs exigences de conception, dont certaines sont relativement difficiles à formuler et modéliser. Ceci rend indispensable l'implication des concepteurs pour la validation des solutions proposées par le système de résolution. Celui-ci doit être donc interactif et suffisamment "souple" pour traiter les demandes des concepteurs.

Nous nous concentrons dans notre travail sur les problèmes d'agencement 3D avec deux principaux objectifs : (1) résoudre automatiquement les problèmes d'agencement en proposant diverses solutions successivement explorables, et (2) assister l'utilisateur (ou concepteur) lors des agencements manuels.

Bien que le problème soulevé par *Thales Communications & Security* soit à la base de ce travail, nous proposerons une formulation générique des problèmes d'agencement pouvant être utilisée pour formuler plusieurs variétés de ces problèmes. Une modélisation informatique spécifique sera également proposée dans ce chapitre, permettant l'intégration d'un module de résolution de contraintes dans un système de visualisation 3D. Le système de résolution de problèmes d'agencement développé sera présenté dans le chapitre 6.

5.2 Identification du problème

5.2.1 Contexte

Aménager un shelter (Fig. 5.1) implique l'embarquement de nombreux équipements informatiques et électroniques dans un espace réduit. Un shelter peut être défini comme un espace technique mobile ayant une forme parallélépipèdique. L'objectif est d'aménager l'espace tout en permettant aux opérateurs d'avoir un poste de travail le plus pratique possible. Des aspects liés à l'accessibilité (facilité d'access) et au confort (éclairage) des postes de travail sont également considérés. Actuellement, lorsqu'un client désire aménager un shelter ou tout autre véhicule, *Thales Communications & Security* lui propose des solutions via des logiciels de conceptions classiques (Catia, AutoCAD, etc). Le client choisit le modèle qui lui parait le plus adapté, l'entreprise lance alors la création d'un prototype (shelter aménagé) sous forme de maquette en bois. Ce procédé est assez complexe, coûteux, long et, parfois même, peut ne pas satisfaire le client, ce qui implique une nouvelle réflexion et la réalisation d'une nouvelle maquette. En effet, le résultat perçu sur un logiciel de conception peut être différent de la réalité.

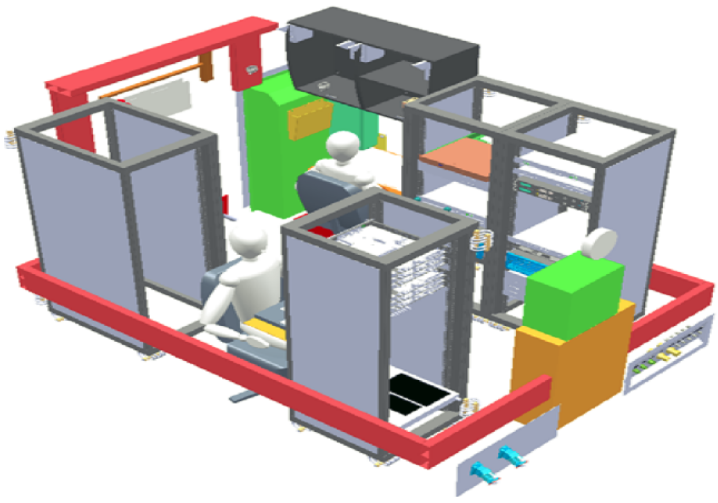


FIGURE 5.1 – *Modéle CAO d'un shelter (Image tirée de Bénabès* [22]).

5.2.2 Objectifs

Pour pallier à ce problème, un système permettant à la fois, l'automatisation de l'agencement des espaces restreints, et l'assistance de l'opérateur doivent être mis en place. Dans cet objectif, nous avons proposé de combiner un module de

résolution de contraintes (solveur) et un système de visualisation/interaction 3D visant principalement à réduire les coûts engendrés par la méthode de conception utilisée actuellement. A travers le formalisme CSP, les exigences des concepteurs peuvent être formulées et implémentées pour leur prise en compte dans la recherche de solutions. De plus, l'utilisation des méthodes de résolutions exactes (chapitre 3) permettra de trouver plus d'une solution et ainsi faciliter la prise de décisions des concepteurs.

5.2.3 Spécification

Le système basé sur le couplage solveur-environnement virtuel permettra de :

- Formuler et implémenter des contraintes géométriques de placement (contraintes de distance, de proximité, etc) ;
- Formuler et implémenter des contraintes physiques pour tenir compte de l'effet des champs électriques, magnétiques et thermiques que peuvent avoir certains objets sur d'autres (voir chapitre 6) ;
- Formuler et implémenter des contraintes d'accessibilité et d'éclairages. Ce type de contraintes permet à l'utilisateur d'exprimer le besoin d'éclairer certains postes de travail et garanti l'accès à ceux-ci ;
- Interagir avec la scène simplement et intuitivement afin de faciliter la manipulation des objets ;
- Proposer plusieurs points de vues afin d'améliorer la perception pour une meilleure prise de décisions ;
- Proposer, quand cela est possible, plus d'une solution réalisable afin de donner un éventail de choix à l'utilisateur ;
- Ajouter dynamiquement des objets et/ou contraintes ;
- Supprimer dynamiquement des objets et/ou contraintes ;
- Permettre à l'utilisateur de changer une configuration d'objets proposée par le système en plaçant/orientant manuellement certains objets à l'intérieur du shelter. Le système doit annuler l'action de l'utilisateur en cas d'infaisabilité ;
- Assister l'utilisateur lors d'un agencement manuel en l'orientant vers les "bonnes" positions des objets ;
- Fournir des informations relatives à l'infaisabilité d'un problème d'agencement donné. Par exemple, afficher à l'utilisateur les contraintes violées, ce qui lui permettra de les modifier/supprimer pour lancer une nouvelle recherche.

5.3 Approche proposée

Afin de résoudre la problématique liée à l'agencement automatique d'un espace donné et celle liée à l'assistance à fournir durant le placement manuel de certains objets, nous proposons de combiner des techniques de programmation par contraintes (PPC) et de réalité virtuelle (RV) pour concevoir et développer un outil d'aide à la décision interactif.

La PPC sera utilisée pour la formalisation et la résolution du problème décrit ci-dessus. Une communication étroite entre le solveur et l'environnement virtuel (EV) qui représente l'espace à agencer, sera établie afin de permettre à l'utilisateur d'exprimer ses exigences (contraintes), de choisir la meilleure configuration des objets et de demander une assistance. Une description plus détaillée de cette approche sera donnée dans le chapitre 6.

5.4 Formulation du problème

Formuler un problème d'agencement 3D nécessite un formalisme efficace et bien structuré, permettant de représenter tous les paramètres (inconnues et contraintes) du problème. Nous avons choisi d'utiliser la programmation par contraintes, à travers le formalisme CSP, pour formuler et résoudre ce problème pour plusieurs raisons :

- Comme montré par Mizoguchi [145] et Pfefferkorn [118], la programmation par contraintes est particulièrement appropriée pour résoudre des problèmes liés à l'agencement spatial,

- Bien que la plupart des contraintes géométriques de placement (non chevauchement, objet-sur-objet, etc) peuvent être satisfaites manuellement sans faire appel à des systèmes de résolution, la satisfaction des contraintes physiques (voir chapitre 6) est très difficile, voir impossible sans l'aide d'un solveur de contraintes,

- Aider l'utilisateur dans sa prise de décision au cours de l'agencement, inclut la recherche et la présentation de plusieurs solutions réalisables. L'utilisation des méthodes exactes de résolution (voir chapitre 3) est ainsi justifiée puisqu'elles permettent de trouver toutes les solutions. Contrairement aux méthodes incomplètes qui ne garantissent pas de trouver une solution et ne peuvent pas expliquer l'infaisabilité, les méthodes exactes peuvent fournir des informations quand aux causes de l'infaisabilité d'un problème d'agencement donné. Ces informations permettent à l'utilisateur de revenir sur ses choix (objets et contraintes).

D'une façon générale un problème d'agencement 3D est défini par les paramètres suivants :

- *Le conteneur* : représente l'espace à agencer (bureau, shelter, etc) ;
- *Les objets* : les équipements, meubles ou matériels, dispositif d'éclairage à placer dans le conteneur ;

- *Les exigences de l'agencement* : les contraintes conditionnant la disposition de l'ensemble des objets.

Dans notre contexte, les inconnues du problème sont les positions et les orientations des objets à agencer, les contraintes sont les restrictions ou les relations reliant les objets entre eux et avec le conteneur. Étant donné que les objets et les équipements considérés dans notre cas, ont une forme parallélépipèdique, ils peuvent être orientés selon quatre angles (0^0, 90^0, 180^0, 270^0). Pour formuler le problème d'agencement 3D sous forme de CSP, nous supposons que l'espace 3D de dimensions (w, h, d) contient n objets sur lesquels m contraintes sont postées. X représente l'ensemble des variables (inconnues) du problème, et $\mathcal{D}$, une fonction qui renvoi le domaine (valeurs possibles) de chaque variable, et C l'ensemble des contraintes. Ainsi, le problème est défini par le triplet (X,$\mathcal{D}$,C) :

- $X = \{x_1, y_1, z_1, \theta_1 ..., x_n, y_n, z_n, \theta_n\}/(x_i, y_i, z_i, \theta_i/i \in [1, n])$ représente respectivement la position et l'orientation de l'$objet_i$

- $\mathcal{D}(x_i) = [w_i, w - w_i]$, $w_i/i \in [1, n]$ est la largeur de l'$objet_i$
  $\mathcal{D}(y_i) = [h_i, h - h_i]$, h_i/i$\in [1, n]$ est la hauteur de l'$objet_i$
  $\mathcal{D}(z_i) = [d_i, d - d_i]$, d_i/i$\in [1, n]$ est la profondeur de l'$objet_i$
  $\mathcal{D}(\theta_i) = \{0, 90, 180, 270\}$, d_i/i$\in [1, n]$ est l'orientation de l'$objet_i$

- $C = \{c_1^{i,j}, c_2^{i,j}, ..., c_m^{i,j}\}$, $c_k^{i,j}/(k \in [1, m]$ and $i, j \in [1, n])$ est une contrainte reliant l'$objet_i$ et $objet_j$.

Afin de mettre en place un système ou un outil qui illustre l'intégralité de l'approche proposée et qui répond aux spécifications recensées, nous donnons dans ce qui suit une architecture logicielle décrivant d'une manière symbolique et schématique les différentes fonctionnalités du système à développer. Cette architecture permet également d'expliciter les interactions entre les éléments qui composent le système.

5.5 Architecture logicielle

La mise en place d'une architecture logicielle est une phase très importante puisqu'elle permettra d'implémenter l'approche proposée (résolution automatique + assistance) en définissant les structures et les flux d'informations entre les entités. En outre, la réalisation d'un modèle structuré permettra une évolution plus aisée du système. Le langage de modélisation UML (*Unified Modeling langage*) a été utilisé pour modéliser le problème.

5.5.1 Diagramme de cas d'utilisation

Afin de structurer les besoins et les objectifs visés sans chercher l'exhaustivité, nous proposons un diagramme de cas d'utilisations (Fig. 5.2). En effet, ce modèle conceptuel permet une meilleure compréhension des fonctionnalités du système.

Le seul acteur dans ce diagramme est l'utilisateur qui interagit avec le système en ajoutant des objets dans l'environnement 3D et en sélectionnant les contraintes qui conditionnent la disposition de ces objets dans l'espace à agencer. Ainsi, le système doit créer le CSP relatif au problème d'agencement proposé. L'utilisateur doit pouvoir modifier dynamiquement le problème en ajoutant ou supprimant des objets et des contraintes au cours de l'agencement. Selon sa volonté, il peut demander soit un placement automatique des objets soit demander une assistance lors du placement manuel de certains d'entre eux, comme il peut combiner ces deux fonctionnalités. L'assistance fournie est une aide "prédictive", visuelle et pseudo temps-réel permettant de guider l'utilisateur pour placer correctement un objet donné. Plusieurs formes d'assistance sont possibles et se basent sur le mécanisme d'anticipation (***look ahead***). Quant à la résolution automatique, elle fait appel aux mécanismes de base de la programmation par contraintes (filtrage, propagation, etc).

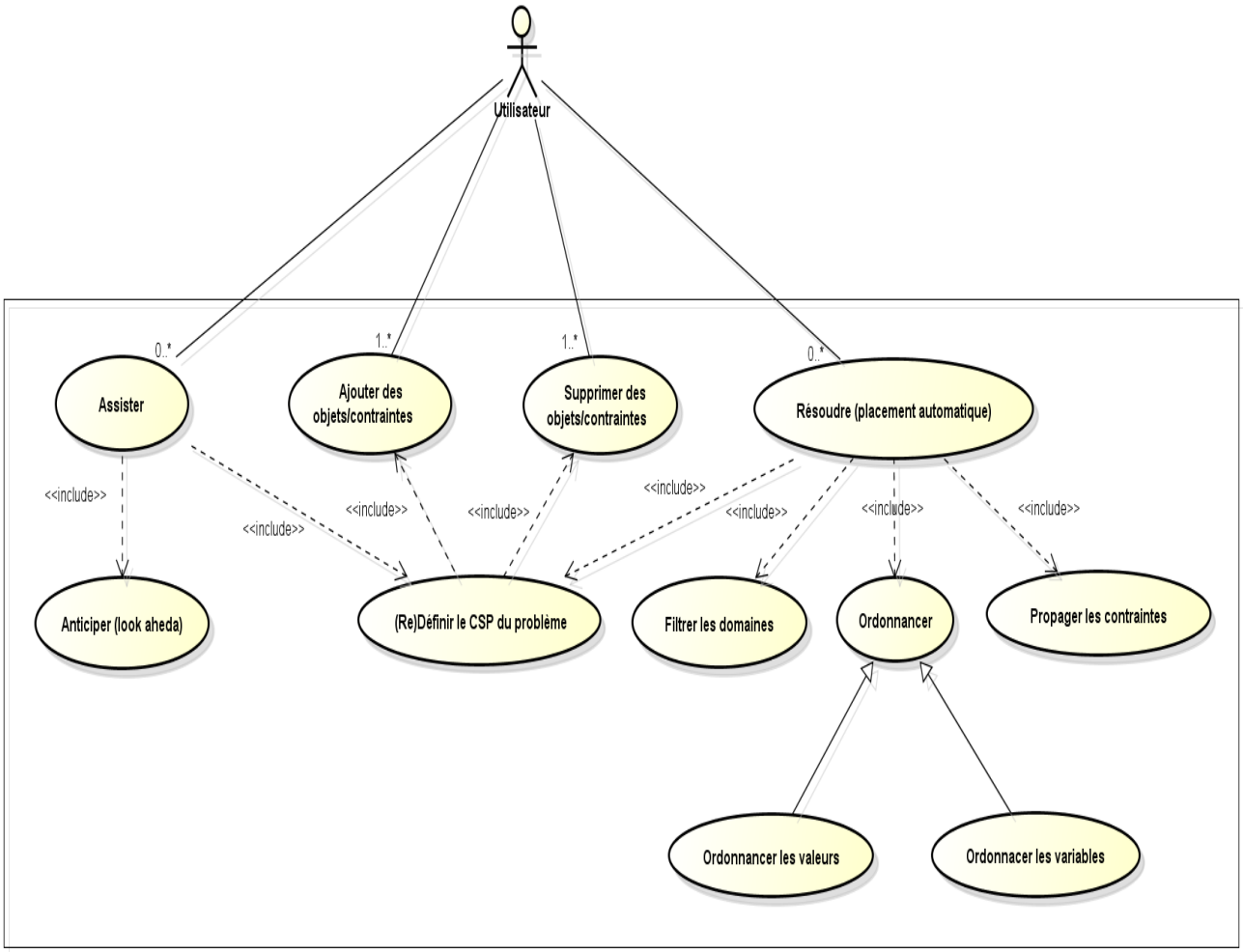


FIGURE 5.2 – *Diagramme des cas d'utilisations.*

5.5.2 Diagramme de classes

Ce diagramme est généralement utilisé pour décrire la structure et la hiérarchie des entités manipulées par le programme. En conception, il représente la structure d'un code orienté objet.

La classe singleton (*Manager*) gère la création et l'envoi des événements (les requêtes) aux différents *Process* (Fig. 5.3). Chaque *process* exécute chaque événement dans sa liste *All_Events* sur un *Thread* indépendant. Ces threads se chargent de lancer/réveiller les instances du solveur et de l'environnement 3D. Les événements portent sur des variables ou des contraintes, représentées par une seule classe (*Objet*). Différentes contraintes, dont la description est donnée dans le chapitre 6, sont traitées par le solveur.

5.5.3 Diagramme d'activités

Un diagramme d'activité représente généralement les étapes séquentielles illustrant l'exécution d'un mécanisme ou d'une méthode. Dans notre cas, nous utilisons ce diagramme pour représenter graphiquement le comportement et le déroulement des mécanismes d'assistance représentés par le cas d'utilisation *"Assister"*. Les différents types d'assistances possibles seront détaillés dans le chapitre 7.

Modélisation des mécanismes d'assistance

L'assistance la plus basique qui peut orienter l'utilisateur lors d'un placement manuel d'un objet est l'affichage des surfaces ou des zones 3D qui ne peuvent pas contenir l'objet en question. En effet, placer l'objet dans ces zones conduira à une violation d'une ou plusieurs contraintes. Dès la demande de ce type d'assistance pour un objet obj_i, le système procède à une décomposition de l'espace à agencer en plusieurs zones 3D. Pour chaque zone obtenue (z_j), le solveur vérifie si le placement de obj_i dans z_j provoque ou non une inconsistance (Fig. 5.4). Ainsi, un mécanisme d'anticipation est utilisé en supposant à chaque fois que l'utilisateur veut placer obj_i dans une des zones obtenues, et en étudiant la conséquence de cette supposition sur la validité du CSP courant. Ce processus est répété pour toutes les zones obtenues. Les zones identifiées comme impossibles sont marquées dans l'EV par une couleur spécifique.

Le deuxième type d'assistance permet de vérifier la faisabilité du placement d'un objet dans une position donnée. Dès lors que l'utilisateur demande ce type assistance pour placer un objet à un endroit donné, le système crée et transmet la requête appropriée au solveur. Celui-ci va procéder à un test dans lequel il suppose que l'utilisateur veut placer l'objet à sa position courante et anticipe l'effet de cette supposition sur la validité d'une copie du CSP courant (Fig. 5.5). Ce test illustre le mécanisme du ***look ahead***. Selon le résultat de ce test, le solveur crée et envoie une requête contenant les informations nécessaires à la caractérisation (possible ou non) de la position courante de l'objet. Pour chaque déplacement de l'objet, le CSP original est restauré pour répéter le processus jusqu'à annulation de la demande d'assistance.

Le troisième type d'assistance constitue une extension de la précédente assistance, en effet le système cherche des alternatives de placement quand la position courante est invalide. Suite à la demande de ce type d'assistance, le système teste la validité des n positions les plus proches de la position courante. Dès qu'une alternative de placement est identifiée, le solveur stope le processus et transmets à l'EV les informations nécessaires à la caractérisation de la zone trouvée (Fig. 5.6).

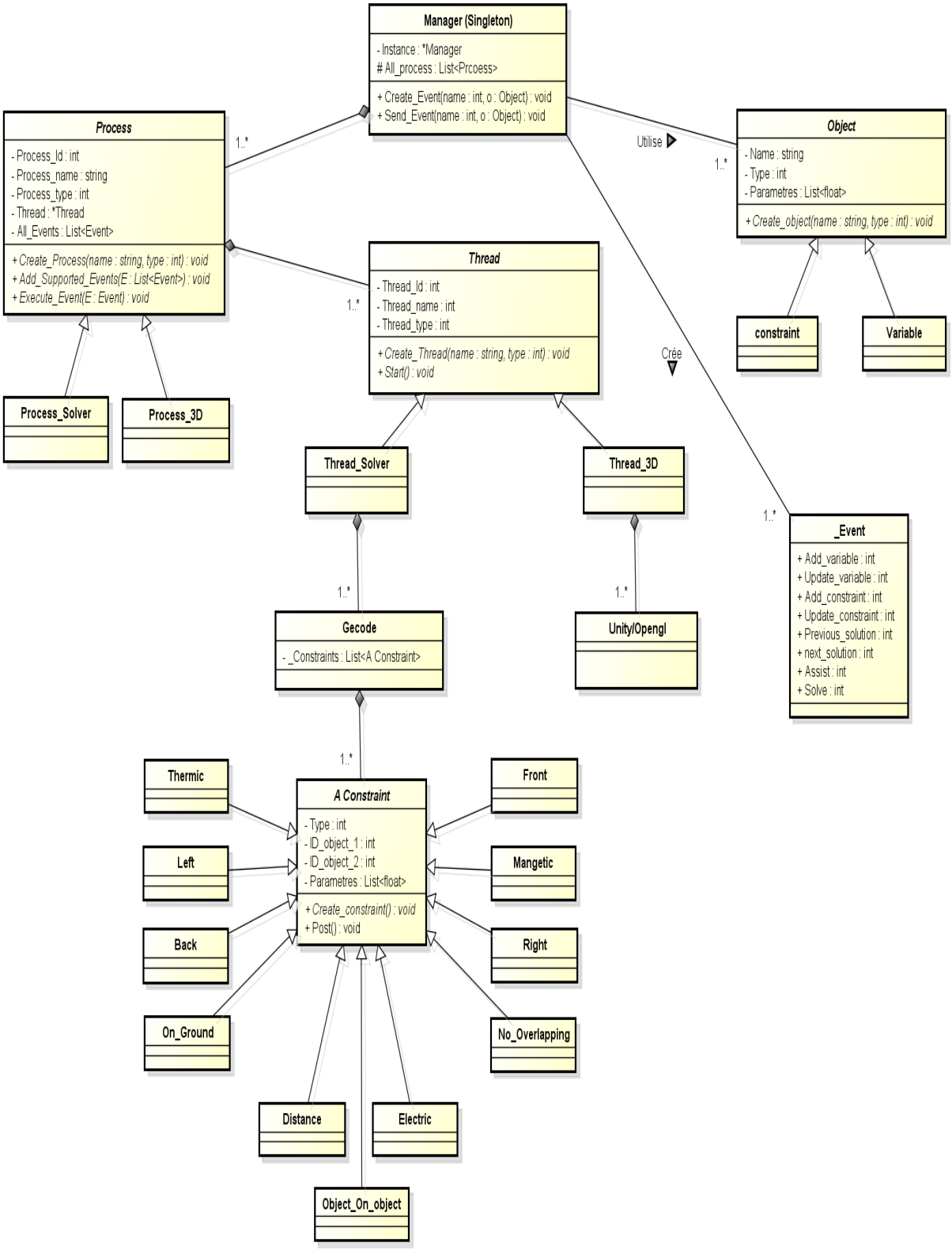


FIGURE 5.3 – *Diagrammes de classe.*

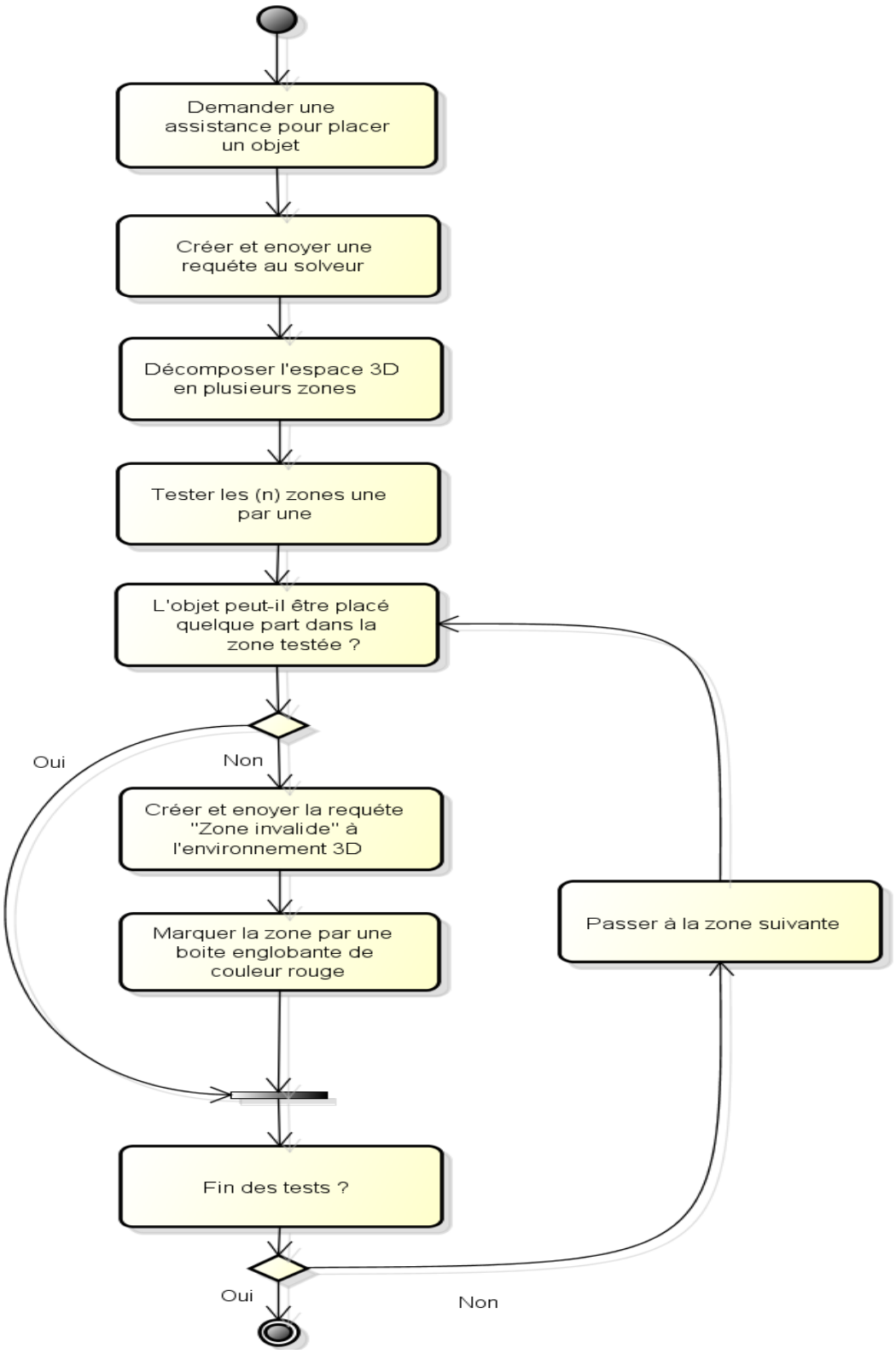


FIGURE 5.4 – *Diagrammes d'activité "Zones impossibles".*

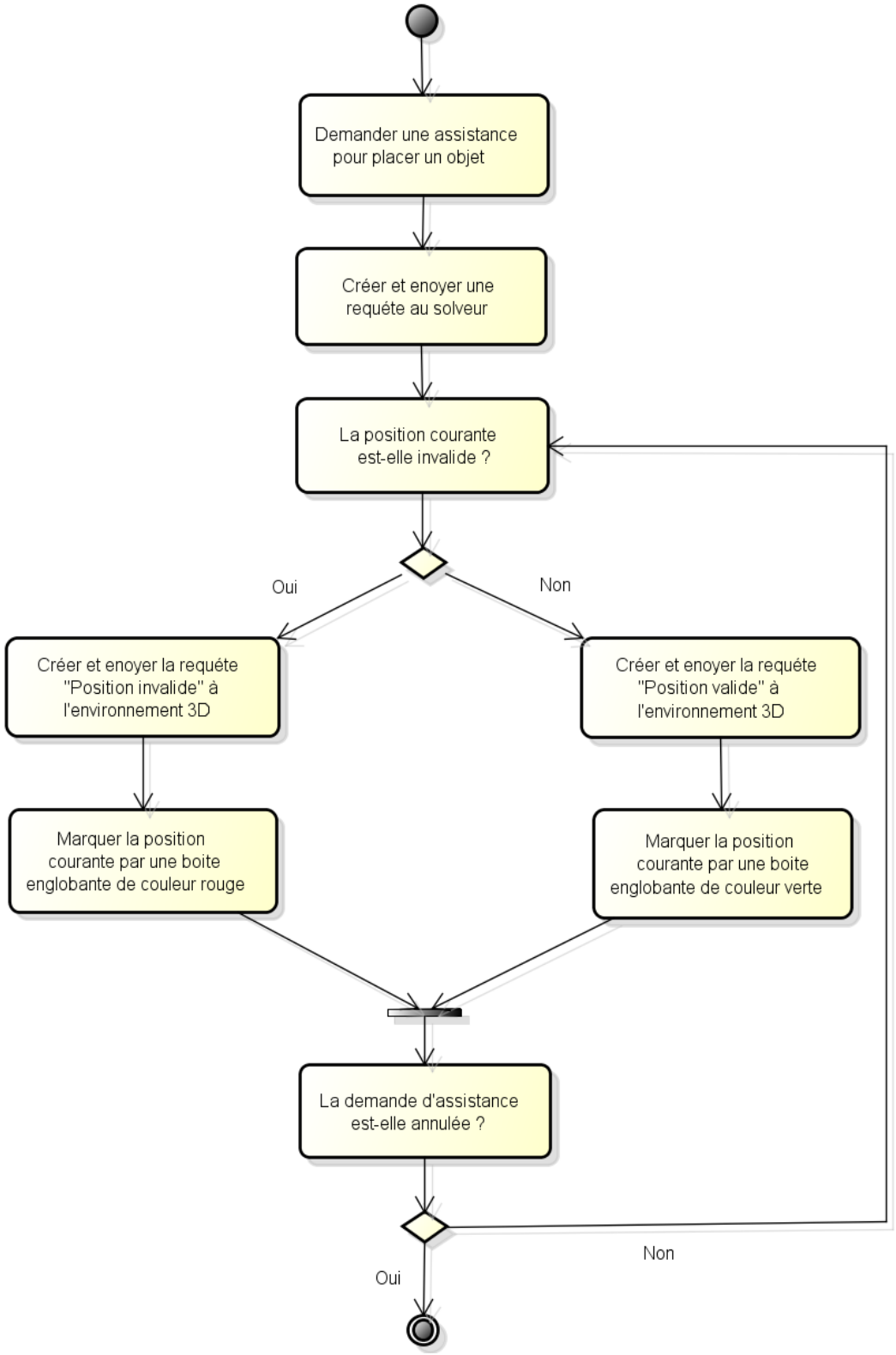


FIGURE 5.5 – *Diagrammes d'activité "État de la position courante".*

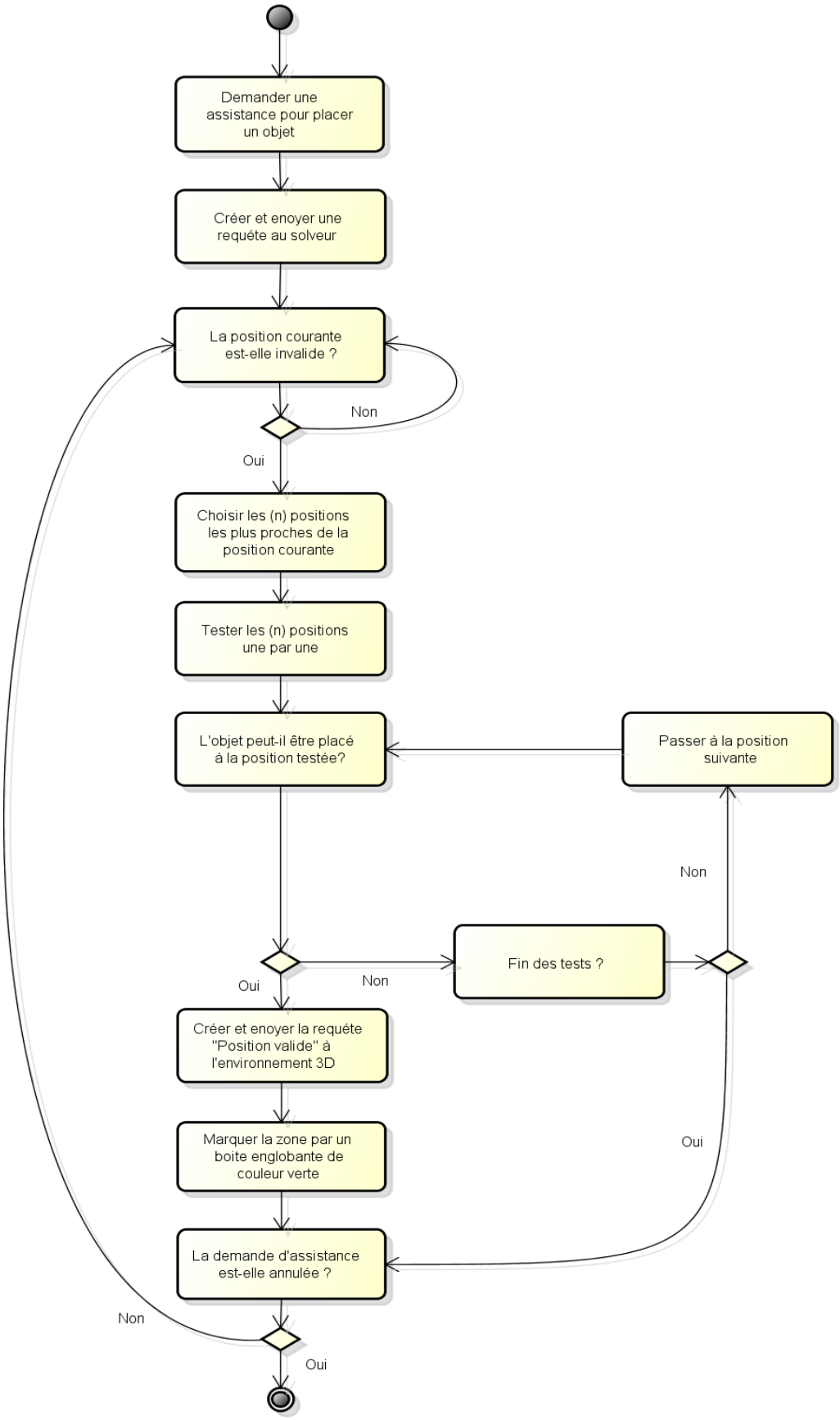


FIGURE 5.6 – *Diagrammes d'activité "Alternatives de placement".*

5.6 Conclusion

Ce chapitre a été consacré à l'identification et la formulation du problème en commençant par la description du contexte. Les besoins et les objectifs visés ont été clairement identifiés, une formulation sous forme de CSP a été également donnée après avoir justifié notre choix d'utiliser la programmation par contraintes (PPC) comme outil de formulation et de résolution du problème. Afin de mener au mieux le développement de l'approche proposée (PPC-RV), nous avons réalisé une modélisation *UML*. La méthodologie proposée, et même si elle est inspirée d'un cas industriel, peut être généralisée et appliquée à d'autres problèmes d'agencement. La mise en oeuvre de notre approche ainsi qu'une description détaillée des fonctionnalités, feront l'objet du chapitre suivant.

6

Système de résolution proposé

6.1 Introduction

Trouver une solution à un problème d'agencement défini par un ensemble de contraintes consiste à chercher un placement des composants (objets, matériels, meubles) à l'intérieur d'un contenant (bureaux, véhicules, etc). L'agencement d'un espace est généralement une tâche fastidieuse et longue quand il s'agit d'espaces contenant un grand nombre de composants soumis à différents types de contraintes. A l'heure où les systèmes deviennent de plus en plus complexes, le développement de méthodes automatiques de résolution devient un enjeu considérable. Cette complexité implique généralement une difficulté en termes de formulation et d'identification des stratégies de résolution appropriées.

C'est dans ce contexte que nous proposons de combiner des techniques de programmation par contraintes (PPC) et de réalité virtuelle (RV) pour concevoir un outil interactif d'aide à la décision afin de résoudre ce type de problèmes et en particulier ceux en 3D. Via un solveur de contraintes (Gecode), nous avons pu modéliser et résoudre différents types de contraintes d'agencement. Un module de communication solveur-environnement virtuel (EV) transmet la solution trouvée à ce dernier qui se reconfigure automatiquement. L'EV qui présente la scène 3D à agencer est étroitement lié avec le solveur, l'utilisateur est en mesure de piloter ce dernier par des manipulations 3D dans l'EV. La formulation du problème ainsi que la recherche des solutions sont complètement transparentes vis à vis de l'utilisateur puisqu'il interagit seulement avec l'EV sans se soucier des mécanismes de résolution.

6.2 Version préliminaire du système

Le développement du système proposé a été précédé par une version préliminaire dont le but était de valider l'approche théorique conçue et basée sur le couplage entre le solveur et le système de visualisation 3D. En effet, un EV basé sur OpenGl a été mis en place permettant un placement automatique (via le solveur) des objets 3D simples. Une étude expérimentale a été menée pour étudier l'influence de

l'intégration d'un module de résolution de contraintes (solveur) dans un EV sur des facteurs objectifs et subjectifs. Les résultats de cette étude sont détaillés dans le chapitre 9.

6.3 Description générale

Le système proposé est un environnement 3D temps-réel développé avec l'outil professionnel Unity3D (version $3.2.0f4$), couramment utilisé pour la création d'EVs et jeux 3D. Unity3D propose un éditeur universel de scripts permettant de coder des routines en C#, JavaScript et Boo [51]. Par ce biais, le système développé intègre des fonctionnalités permettant un échange d'informations et de données entre le solveur Gecode et l'EV à l'aide de scripts C#.

A travers l'interface développée sous Unity 3D (figure 6.1), l'utilisateur utilise un menu pour sélectionner les objets 3D (meubles, matériels, etc) qu'il veut agencer. Un autre menu lui permet de poster des contraintes géométriques et/ou physiques à appliquer sur les objets sélectionnés. Afin d'aider l'utilisateur à comprendre l'intérêt de chaque contrainte, une description schématique simplifiée est affichée sur le haut du menu de contraintes (figure 6.1). Chaque sélection effectuée par l'utilisateur est associée à un retour visuel pour confirmer la validation de l'action. A tout moment, l'utilisateur peut déclencher la résolution, ce qui conduit à la création d'un CSP dans lequel les variables sont les positions 3D des centres des objets. La résolution de ce CSP est laissée à la charge du solveur qui cherche des configurations ou placements possibles des objets dans un temps raisonnable (celui-ci dépend du nombre d'objets et de la complexité des contraintes). Les données relatives aux solutions trouvées sont transmises à l'EV qui se met automatiquement à jour selon les nouvelles positions calculées par le solveur. L'utilisateur est également en mesure de naviguer dans l'espace des solutions afin de choisir celle qui lui convient le plus (via le menu de gauche).

6.4 Architecture du système

Le système proposé est basé sur l'intégration d'un module de résolution de contraintes (Gecode) dans un EV temps-réel. L'objectif est de résoudre des problèmes d'agencements 3D de manière interactive. Le choix des objets et contraintes ainsi que l'interaction de l'utilisateur (manipulation d'objets) sont convertis en requêtes transmises au solveur. La rapidité des calculs (sous forme de positions d'objets) effectués par le solveur permettent une reconfiguration automatique et rapide de la scène.

La Figure 6.2 illustre le processus général de résolution et la communication entre l'environnement 3D et le solveur. L'échange des événements est géré par une classe principale (*Manager*) qui crée et envoie l'événement approprié aux modules de visualisation (*Process_3D*) et de résolution (*Process_Solver*). Chaque module transmet les requêtes à traiter au processus (*thread*) correspondant. Afin d'accélérer les calculs et assurer une interaction 3D temps-réel, chaque module est exécuté sur un *thread* différent. La classe *Thread_3D* assure l'affichage et la mise à jour de l'EV créé par l'utilisateur et informe la classe *Process_3D* des choix d'objets et de

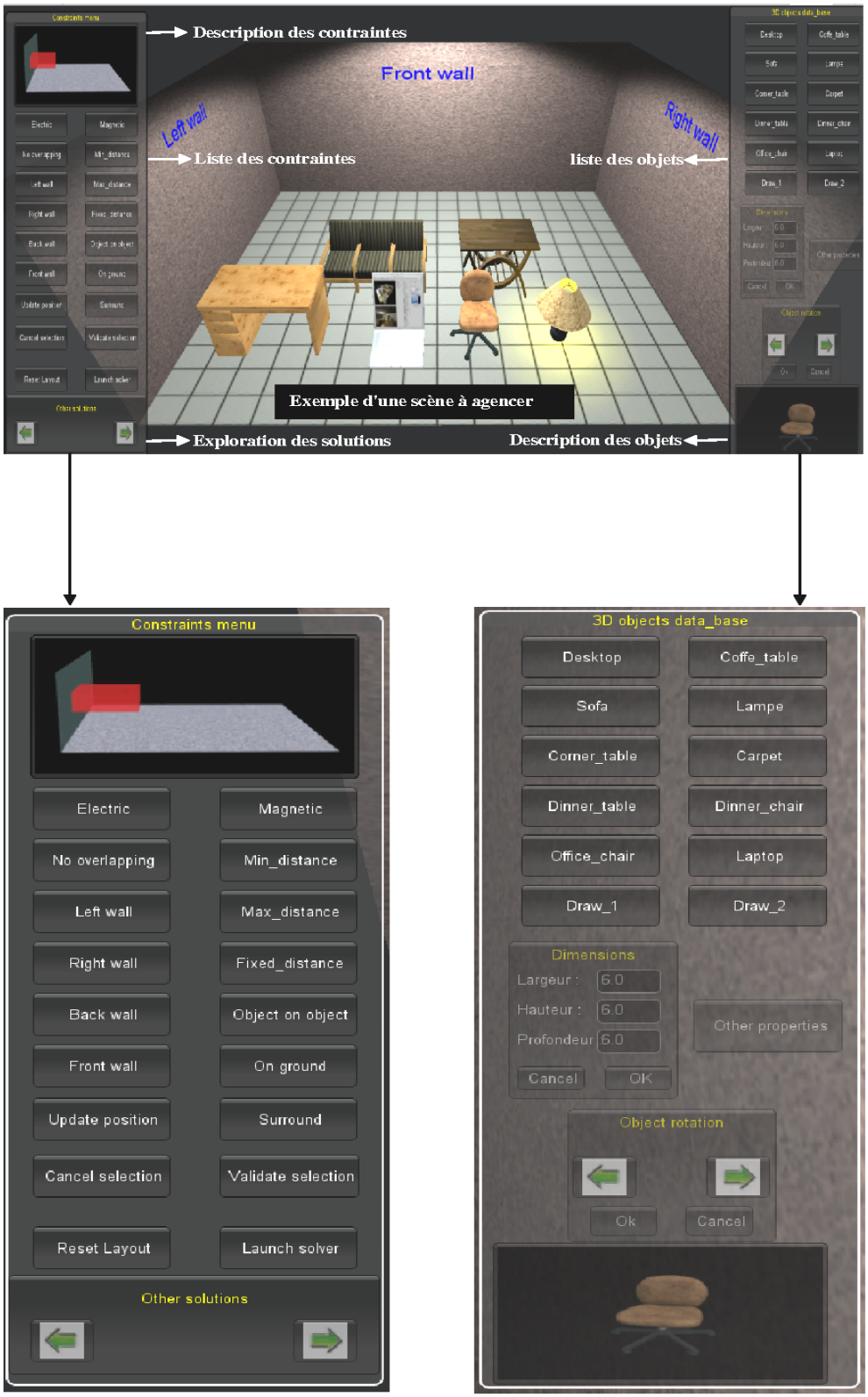


FIGURE 6.1 – *L'interface graphique du système proposé.*

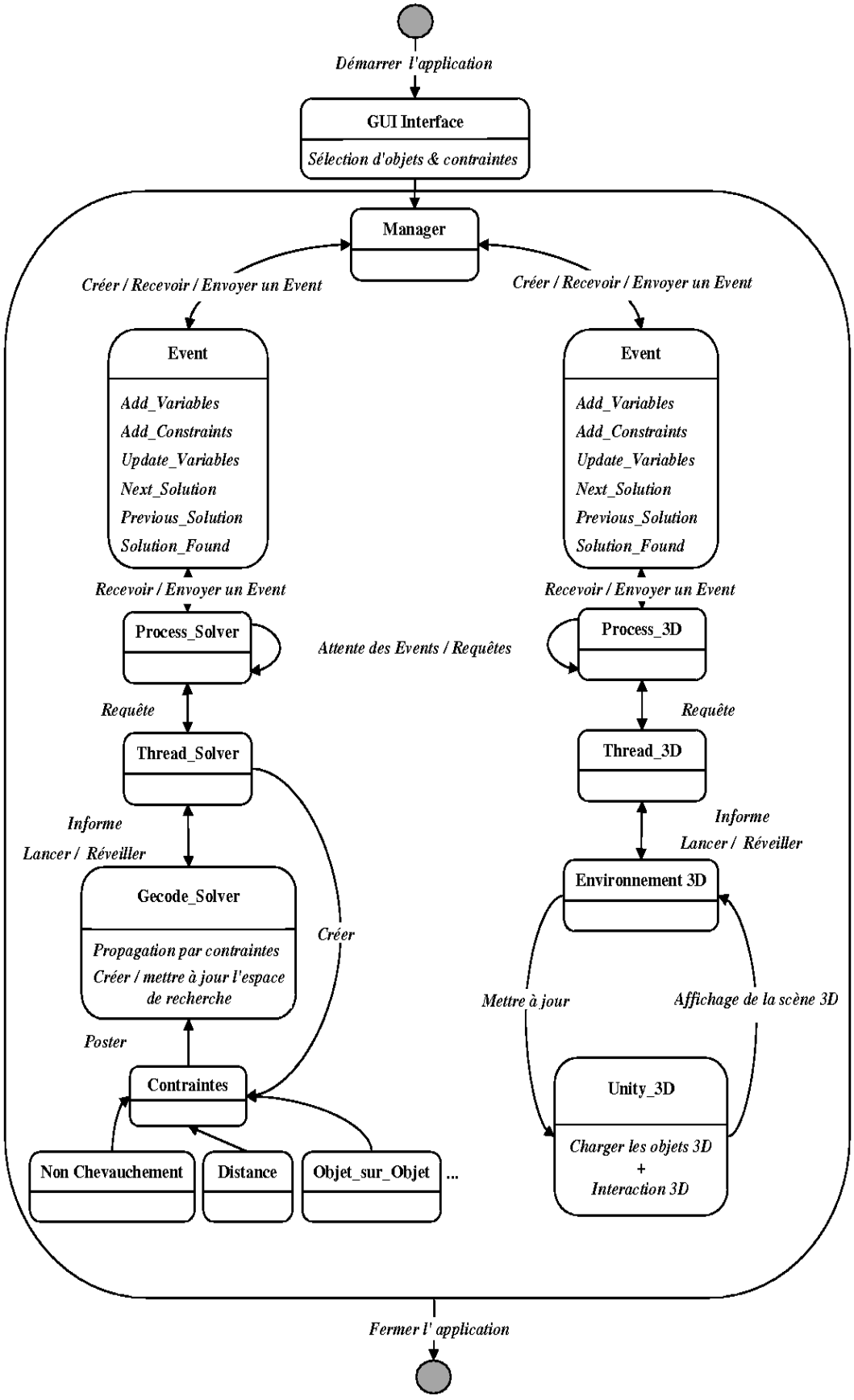


FIGURE 6.2 – *Architecture générale du Système : communication solveur-EV.*

contraintes effectués. Après l'acheminement des requêtes, la classe *Process_Solver* crée ou met à jour le CSP représentant le problème. Après un certain temps de calcul, la classe *Thread_Solver* est informée de la (ou des) solution(s) trouvée(s) et transmet l'événement approprié à la classe du niveau supérieur. Ainsi, la classe *Process_Solver* informe la classe *Manager* qu'une solution a été trouvée (à travers l'événement "*Solution_found*"). Selon les directives de la classe *Manager*, les nouvelles positions 3D calculées sont transmises jusqu'à ce que l'EV se reconfigure en mettant à jour les positions des objets 3D.

6.5 Module d'interaction avec le solveur

En général, le processus de résolution d'un CSP est déclenché par la modification des valeurs des variables ou des contraintes. Dans notre cas, l'interaction de l'utilisateur avec les objets virtuels se traduit par des commandes (ou requêtes) envoyées au solveur, celui-ci agit alors sur les variables et modifie leurs valeurs (figure 6.3). Par exemple, l'utilisateur interagit avec une configuration proposée par le solveur en déplaçant manuellement certains objets. Cette interaction modifie les domaines des variables et déclenche le processus de propagation des contraintes (étape 1-traits pleins). Le solveur est donc "réveillé" pour (i) mettre à jour le CSP, (ii) vérifier la validité de celui-ci et finalement (iii) accepter ou décliner la solution proposée par l'utilisateur (étape 2-traits pointillés).

Afin de se rapprocher le plus possible de la réalité et apporter une aide à l'utilisateur, les objets 3D sont physicalisés (via le moteur physique "Physx" de NVIDIA™) permettant l'application de forces de collision entre les objets quand l'utilisateur les manipule.

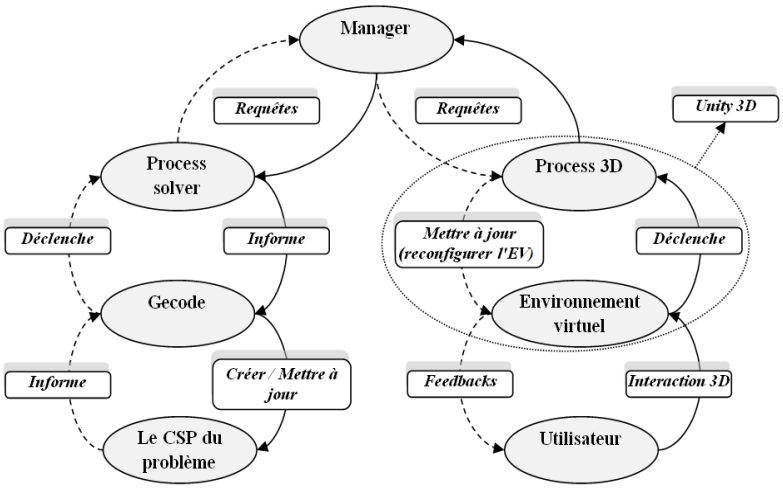


FIGURE 6.3 – *Module d'interaction avec le solveur.*

6.6 Implémentation des contraintes

Afin de résoudre le problème exposé dans le chapitre 5, nous avons choisi de développer des contraintes spécifiques, liées à ce type de problème. Les *contraintes géométriques* reflètent les exigences naturelles qui peuvent être définies pour un

agencement concret. Quant aux *contraintes physiques*, elles ont été développées pour tenir compte de l'effet possiblement néfaste que peuvent avoir certains objets sur d'autres. Afin de garantir un aspect générique des contraintes développées, le système considère les quatre murs de la scène et le sol comme des objets, dont les positions sont fixes et ne peuvent pas être modifiées ni par l'utilisateur ni par le solveur. Les contraintes implémentées via le solveur sont classifiées et décrites ci-après :

1. **Contraintes géométriques**
   - *Contrainte "Distance_minimale" (DMin)*
     Cette contrainte peut être appliquée sur au moins deux objets pré-sélectionnés. Le solveur fera en sorte que les objets sélectionnés soient séparés par une distance minimale (*dmin*) qui peut être fixée par l'utilisateur. Le même principe est utilisé pour les contraintes *"Distance_maximale" (DMax)* et *"Distance_fixe" (DF)*.

   - *Contrainte "Non_chevauchement" (NC)*
     Considérée comme un cas particulier de la contrainte *"Distance_minimale"*, cette contrainte assure la non superposition de deux objets. Dans tous les cas, la contrainte *"Non_chevauchement"* est postée par défaut.

   - *Contrainte "A_droite" (AD)*
     La contrainte "A_droite" place un premier objet sélectionné à droite d'un deuxième objet. Les deux objets ne seront pas forcément collés. Cette contrainte peut être utilisée pour placer un objet contre le mur droit puisque ce dernier est considéré comme un objet. Le même principe est utilisé pour les contraintes *"A_gauche" (AG)*, *"Devant" (DV)* et *"Derrière" (DR)*.

   - *Contrainte "Objet_sur_objet" (OSO)*
     Elle peut être appliquée sur deux objets ; celui sélectionné en deuxième sera placé sur le premier. Placer un objet sur le sol constitue un cas particulier de cette contrainte.

   Les contraintes géométriques peuvent être mixées pour répondre à d'autres exigences de conception souvent exprimées dans les problèmes d'agencement. Par exemple, la contrainte *"Autour" (A)* pour laquelle le solveur cherche un placement de $(n-1)$ objets autour d'un objet central (obj_0 : le premier sélectionné). Cette contrainte peut être définie via une association entre les contraintes de type *"Distance_minimale"* et *"Distance_fixe"* (Les distances *_dfixe* et *_dmin* sont choisies par défaut par le système) :

   – $Distance\_fixe(obj_0, obj_i, \_dfixe)/i \in [1, n-1]$
   – $Distance\_minimale(obj_i, obj_j, \_dmin)/i, j \in [1, n-1]$

2. **Les contraintes physiques**
   Comme leur nom l'indique, ces contraintes font appel aux lois de la physique et prennent en entrée les valeurs des paramètres définissant chaque loi. Dans certains cas la durée de vie d'un objet ou matériel peut être diminuée en présence de fortes valeurs des champs électriques, magnétiques ou thermiques.

Ces contraintes ont été développées afin de "minimiser" l'effet de ces champs sur les objets qui leurs sont sensibles. Pour ce faire, et lors de l'ajout d'un objet dans la scène à agencer, l'utilisateur doit spécifier si ce dernier est susceptible d'émettre un champ particulier en saisissant les valeurs définissant le champ en question. Par exemple pour le champ magnétique, il faut saisir la valeur de la charge électrique $\boldsymbol{q}$ ainsi que sa vitesse $\boldsymbol{v}$.

- *Contrainte "Électrique" (E)* :
  L'utilisateur doit sélectionner l'objet qu'il veut protéger. Il doit ensuite saisir la valeur maximale du champ électrique que l'objet peut supporter. Étant donné que le solveur est déjà informé de l'existence d'objets émetteurs de ce type de champ et en se basant sur la formule permettant de calculer le module du champ électrique en fonction de la distance par rapport à une charge $\boldsymbol{q}$ (équation 6.1), il place l'objet en question à une distance minimale afin de le protéger. En effet, la tâche du solveur consiste à trouver la distance minimale devant séparer chaque objet émetteur du champ électrique de l'objet à protéger (objet$_p$) de telle façon que le champ électrique résultant émis par l'ensemble des objets émetteurs soit inférieur ou égal au champ maximum supporté par l'objet$_p$.

$$\| E \| = \frac{q}{4.\Pi.\epsilon_0.d_E^2} \tag{6.1}$$

- *Contrainte "Magnétique" (M)* :
  Dans ce cas les calculs du solveur sont basés sur la formule générale permettant de calculer le module du champ magnétique en fonction de la distance par rapport à une charge $\boldsymbol{q}$ (équation 6.2). Comme pour la contrainte *Électrique*, le solveur positionne l'objet suffisamment loin afin de le protéger.

$$\| B \| = \frac{10^{-7}.q.v}{d_M^2} \tag{6.2}$$

- *Contrainte "Thermique" (T)* :
  Comme pour les champs électriques et magnétiques, certains objets sont sensibles à la chaleur et leur fonctionnement est prévu pour une température nominale déterminée. En outre, un excès de chaleur réduit considérablement leur durée de vie. La température d'un point appartenant à l'espace à agencer dépend fortement de sa proximité par rapport l'environnement extérieur. En effet, plus il est proche des murs (ou parois ou fenêtres), plus sa température est élevée. Ainsi, deux cas sont possibles :

  (a) La température externe n'est pas prise en compte ou n'influence pas la température interne (celle de la scène). En se basant sur le principe de l'équilibre thermodynamique, la température d'équilibre atteinte au bout d'un certain temps est égale à celle de la source de la chaleur (généralement un radiateur). Ainsi pour tout point M : $T(M) = T(s)$.

  (b) La température externe est prise en compte ce qui modifie la température des murs (T_m). Pour tout point M de l'espace à agencer, la température est dépendante de celle du mur (le plus proche de M) et de la distance entre celui ci et la source de chaleur. Le calcul est donc relativement compliqué. Nous proposons un modèle simplifié et approximé

permettant de répondre à notre besoin (équation 6.3). Ce modèle a été validé par un spécialiste du domaine.

$$\| T \| = \frac{R_m.R_s.(T_s - T_m)}{R_m - R_s}.\frac{1}{d_T} + \frac{R_m.T_m - R_s - T_s}{R_m - R_s} \tag{6.3}$$

R_m : La distance entre la source de chaleur et le mur
R_s : Le rayon de la source de la chaleur (peut être approximée par la plus petite dimension de l'objet)
T_m : La température du mur
T_s : La température de la source de chaleur (pour un radiateur, c'est la température réglée par l'utilisateur)
d_T : La distance (qui sera calculée par le solveur) entre le point M et le centre de la source de chaleur

Pour mieux comprendre l'intérêt des contraintes physiques, prenons le cas où deux objets obj_E et obj_M émettent respectivement un champ électrique et magnétique et un obj_S sensible à ces deux champs et doit donc être protéger. Une fois que tous les paramètres (***q***, ***v***, etc) sont connus, le système poste deux contraintes *"Distance_minimale"* comme suit :

- $Distance\_minimale(obj_S, obj_E, max(d_E, d_B))$
- $Distance\_minimale(obj_S, obj_M, max(d_E, d_B))$

où d_E (respectivement d_B) est la distance *d* calculée à partir de l'équation 6.1 (respectivement l'équation 6.2). Les distances d_E et d_B sont dynamiques puisqu'elles dépendent des positions des objets obj_E et obj_M par rapport à obj_S.

Le tableau 6.1 présente un résumé de l'ensemble des contraintes implémentées sous Gecode. Pour chacune d'entre elles, on spécifie les paramètres d'entrée, le résultat, la compatibilité par rapport aux autres contraintes et un exemple d'utilisation. La compatibilité permet d'identifier les contraintes qui peuvent être postées de manière conjointe, et celles qui présentent des contradictions avec d'autres.

6.7 Temps de résolution

Un système d'aide à la décision est destiné à faciliter les choix et les prises de décision dans un contexte donné. Dans le contexte des problèmes de placement 3D, un système d'aide à la décision bien conçu est un logiciel interactif qui facilite la tâche d'un utilisateur et qui dégage le plus rapidement possible des informations utiles afin d'orienter l'utilisateur vers les bons choix de placement. De ce fait, le temps de résolution est un critère important pour l'évaluation du système proposé. Le temps de calcul nécessaire pour que le solveur trouve une solution à un problème ou réponde à une action de l'utilisateur dépend de quatre paramètres :

- *Le nombre des objets à agencer*
  Pour chaque objet de la scène 3D, le solveur définit trois variables de position (x, y, z) et une variable d'orientation dans le CSP du problème. Un grand nombre d'objets implique donc un grand nombre de variables à instancier ce qui nécessite plus de temps de calcul.

- *Le nombre des contraintes du problème*
  Une contrainte présente une restriction sur une ou plusieurs variables. Plus il y a de contraintes, plus il y a des conditions à vérifier sur les variables et plus il est difficile de trouver rapidement une solution.

- *L'ordre du choix des variables à instancier*
  Il existe de nombreuses heuristiques d'ordre d'instanciation des variables qui permettent bien souvent d'accélérer la recherche. Dans notre contexte, le solveur instancie en premier les variables les plus "critiques", c'est-à-dire celles qui interviennent dans plusieurs contraintes. Ce type d'heuristique réduit considérablement le nombre de retours en arrière (*Backtrack*).

- *L'ordre du choix des valeurs des variables*
  Les heuristiques concernant l'ordre d'instanciation des valeurs sont généralement dépendantes de l'application considérée et difficilement généralisables. Dans notre contexte, le solveur affecte aléatoirement des valeurs aux variables afin de présenter à l'utilisateur des solutions diversifiées. L'utilisation de cette heuristique, relativement simple, ne garantit pas une "nette" diversité entre chaque paire de solutions présentées à l'utilisateur. Au cours de ce travail nous nous sommes limités à cette heuristique sans négliger l'importance d'un tel point dans nos futurs travaux.

La Figure 6.4 montre un exemple d'évolution de temps de calcul du solveur en fonction du nombre des objets et du nombre des contraintes. Dans cet exemple, nous avons défini des contraintes de non-chevauchement entre tous les objets de dimension unitaire (2 à 2) placés dans un espace de dimension $(20, 20, 20)$. Les valeurs des temps correspondent à la durée que met le solveur pour trouver une solution. Ces tests ont été obtenus sur un ordinateur ayant un processeur *Intel*^{TM} $i7 - 2630$ de fréquence $2.00Ghz$.

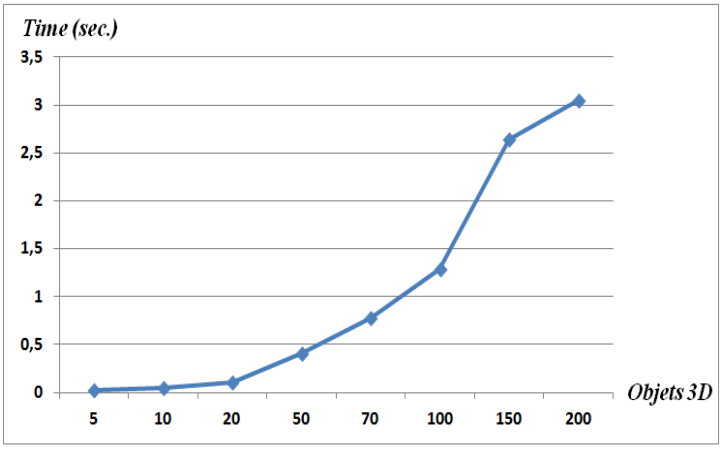


FIGURE 6.4 – *Evolution du temps de calcul en fonction du nombre des objets et du nombre des contraintes.*

Contrainte	Entrée	Sortie	Compatibilité	Exemple
NC	n objets	Tous les objets 2 à 2 sont non superposés	Compatible avec toutes les autres contraintes	Applicable à tous les objets (meubles, matériels,...)
DMin	n objets et *dmin*	Les objets sont 2 à 2 séparés par au minimum *dmin*	Non compatible avec : DMax (si *dmin*>*dmax*), DF (si *dmin*>*dfixe*), OSO	Son utilisation permet le passage de l'utilisateur entre deux objets
DMax	n objets et *dmax*	Les objets sont 2 à 2 séparés par au maximum *dmax*	Non compatible avec : DMin (si *dmax*<*dmin*), DF (si *dmax*<*dfixe*), E (si *dmax*<d_E), M (si *dmax*<d_M), T (si *dmax*<d_T)	Assure le placement à proximité d'un bureau et de sa chaise
DF	n objets et *dfixe*	Les objets sont équidistants de *dfixe*	Non compatible avec : DMin (si *dfixe*<*dmin*), DMax (si *dfixe*>*dmax*), E (si *dfixe*<d_E), M (si *dfixe*<d_M), T (si *dfixe*<d_T), OSO	Peut être utilisée pour placer deux objets dépendants (par exemple l'ordinateur et l'imprimante)
A	n objets	$(n-1)$ objets autour d'un objet central	Non compatible avec : DMin (si *_dmin*<*dmin*) DF (si *_dfixe*≠*dfixe*)	Utile pour placer des chaises autour d'une table
AD	2 objets	Le deuxième objet est placé à droite du premier ($x_2 > x_1$)	Compatible avec toutes les autres contraintes	Place un objet contre le mur droit
AG	2 objets	Le deuxième objet est placé à gauche du premier ($x_2 < x_1$)	Compatible avec toutes les autres contraintes	Place un objet contre le mur gauche
DV	2 objets	Le deuxième objet est placé devant du premier ($z_2 < z_1$)	Compatible avec toutes les autres contraintes	Place un objet contre le mur de devant
DR	2 objets	Le deuxième objet est placé derrière du premier ($z_2 > z_1$)	Compatible avec toutes les autres contraintes	Place un objet contre le mur arriére
OSO	2 objets	Le deuxième objet est placé sur le premier	Non compatible avec : DMin, DF, A, E, M, T	Place un objet sur le sol
E	1 objet et ***q***	L'objet est placé suffisamment loin pour éviter l'effet du champ résultant	Non compatible avec : DMax (si d_E>*dmax*), DF (si d_E>*dfixe*), OSO	Protéger un ordinateur contre des champs électriques
M	1 objet, ***q*** et ***v***	L'objet est placé suffisamment loin pour éviter l'effet du champ résultant	Non compatible avec : DMax (si d_M>*dmax*), DF (si d_M>*dfixe*), OSO	Protéger un ordinateur contre des ondes magnétiques
T	1 objet	L'objet est placé suffisamment loin pour éviter l'effet du champ résultant	Non compatible avec : DMax (si d_T>*dmax*), DF (si d_T>*dfixe*), OSO	Protéger un ordinateur de la chaleur émise par un radiateur

TABLE 6.1 – *Tableau récapitulatif des contraintes développées.*

6.8 Fonctionnalités du système

Notre approche, basée sur l'intégration d'un solveur de contraintes (Gecode) dans un EV, étend les travaux antérieurs dans différentes directions. Par exemple, en termes de mécanismes d'interaction, nous offrons une plus grande intéractivité pour rendre la manipulation d'objets plus efficace et plus intuitive [30, 91, 141]. En effet, au cours d'une tâche d'agencement 3D, l'utilisateur est en mesure de :

- Changer différentes vues de la scène 3D,
- Ajouter des objets, préciser leurs dimensions et sélectionner les contraintes à appliquer à tout moment,
- Exprimer ses préférences en plaçant manuellement des objets, le placement des autres objets est laissé à la charge du solveur.

Il est à noter que le solveur transmet, dans la mesure du possible, les dix premières solutions d'un problème donné. Le changement de l'ordre de sélection des valeurs des variables permet dans certains cas de diversifier les solutions calculées.

En plus de l'intéractivité, le système fournit une assistance à l'utilisateur en :

- Offrant plus d'une solution (quand cela est possible) et lui permette de naviguer dans l'espace des solutions pour sélectionner celle qu'il préfère ;
- Fournissant un retour visuel illustrant les "mauvaises" zones de placement pour aider l'utilisateur à choisir correctement les positions d'objets (voir la Sous-section 6.8.5). D'autres formes d'assistance sont fournies quand l'utilisateur décide de placer lui même certains objets ;
- Annulant toute manipulation d'objets 3D conduisant à une violation des contraintes.

Notre contribution est non seulement liée à l'assistance fournie par le solveur, mais aussi à la façon dont l'utilisateur interagit avec ce dernier. En effet, l'interface graphique proposée est directement liée au solveur permettant des échanges interactifs avec l'utilisateur. Excepté les approches proposées par Calderon *et al.* [34] et Fages *et al.* [54] qui permettent à l'utilisateur un certain degré d'interactivité avec l'EV, les travaux antérieurs ([131, 147, 55, 87]) n'ont pas privilégié cet aspect et l'assistance que peut fournir un système de résolution de contraintes.

A travers le système proposé, l'utilisateur peut intervenir dans l'EV et changer la configuration proposée. Trois situations peuvent se présenter :

- L'utilisateur peut augmenter la taille du problème en insérant de nouveaux objets et/ou choisir de nouvelles contraintes. Dans ce cas, le système définit un nouveau CSP en ajoutant les nouvelles variables (positions d'objets) ou contraintes. Le solveur se chargera alors de résoudre le nouveau CSP et de transmettre les solutions.

- L'utilisateur peut se contenter de la configuration courante et modifier, selon ses préférences, l'emplacement de certains objets. Contrairement au premier cas, le solveur ajoute alors des contraintes aux objets déplacés et met à jour le CSP courant.

- L'utilisateur peut revenir sur ses choix d'objets et/ou de contraintes en les "enlevant" (via un menu) de la configuration courante. Dans ce cas, le solveur revient en arrière pour trouver un CSP déjà créé qui correspond à la nouvelle situation. Si aucun CSP n'est trouvé, le solveur défini alors un nouveau CSP avec les nouveaux paramètres et se charge de le résoudre.

6.8.1 Algorithmes

Nous proposons quatre algorithmes illustrant la réponse du solveur suite à une action de l'utilisateur. Afin de conserver les CSPs déjà créés et une partie du travail effectué par le solveur, une pile (*Stack_spaces*) est utilisée pour stocker chaque CSP créé et valide (tous les domaines des variables sont non vides). Mais avant de commencer la description des algorithmes, il est important de comprendre pourquoi nous avons utilisé le mécanisme de *CSP_backtrack*. L'exemple simple donné dans la Figure 6.5 montre qu'une partie du travail du solveur a été conservée et utilisée pour résoudre le problème et réduire le temps des calculs. Nous supposons que la forme CSP du problème est donnée par :

- $X = \{x_1, x_2\}$,
- $\mathcal{D}(x_1) = \mathcal{D}(x_2) = [0, 5]$
- $C = \{c_1, c_2\}; c_1 = \{x_1 > 2\}, c_2 = \{x_1 = x_2\}$

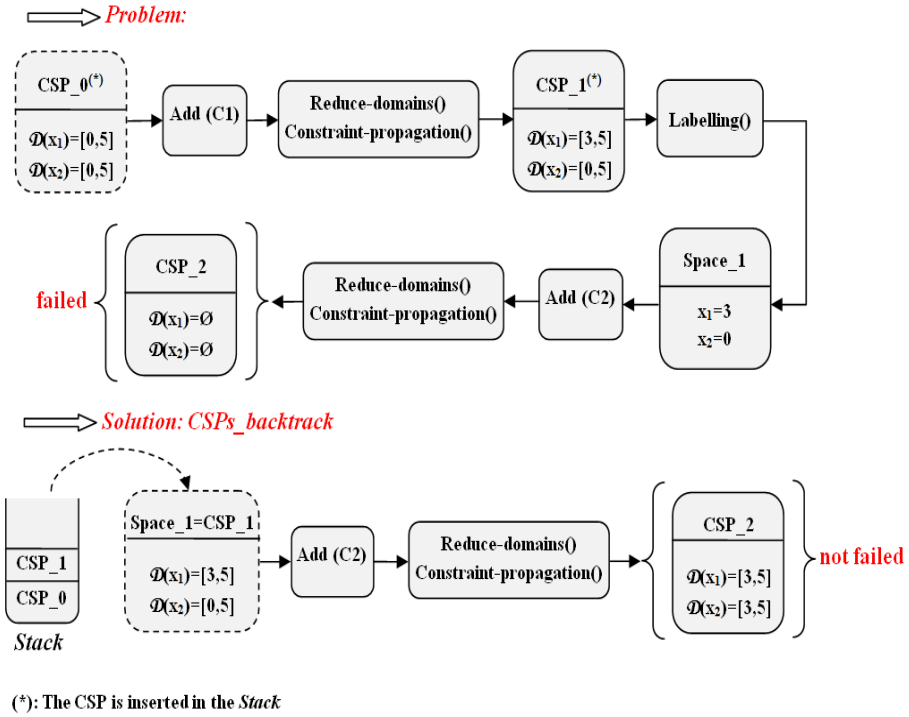


FIGURE 6.5 – *Exemple illustrant le mécanisme du CSP_backtrack.*

Comme décrit précédemment, l'utilisateur peut appliquer des contraintes sur un objet obj_i et le positionner manuellement s'il le souhaite. Dans les deux cas, le solveur met à jour le CSP courant en modifiant les domaines des variables x_i, y_i et

Algorithm 4 Ajout d'une nouvelle contrainte pour l'objet_i

```
1: Ajout_nouvelle_contrainte(i)
2: CurrentSpace.Contraction_des_domaines(x_i, y_i, z_i)
3: CurrentSpace.Propagation_par_contraintes()

4: Solution ← true
5: X'← CurrentSpace.X
6: C'← CurrentSpace.C
7: S'← Stack_Spaces
8: for all var ∈ X' do
9:    if D(var)= ∅ then
10:      if S'≠ ∅ then
11:         CurrentSpace ← S'.pop()
12:         CurrentSpace.C←C'
13:         CurrentSpace.Contraction_des_domaines()
14:         CurrentSpace.Propagation_par_contraintes()
15:         X'← CurrentSpace.X
16:      else
17:         X'← ∅
18:         Solution ← false
19:      end if
20:   else
21:      X'← X'/{var}
22:   end if
23: end for

24: if Solution = true then
25:    Envoi_Event(Solution_Found)
26:    Stack_Spaces.push_back(CurrentSpace)
27: else
28:    CurrentSpace ← Stack_Spaces.pop()
29:    Envoi_Event(No_Solution_Found)
30: end if
```

z_i (l'algorithme 4). Par propagation de contraintes, le solveur vérifie la validité du CSP modifié. Dans le cas où le CSP est non valide (failed), il explore la pile des CSPs *"Stack_spaces"* (en utilisant le mécanisme *CSP_backtrack*) pour en trouver un qui "accepte" la nouvelle contrainte. Sinon (CSP not failed), il retourne la solution trouvée et insère le CSP dans la pile. La Figure 6.5 illustre un exemple dans lequel l'ajout de la contrainte C2 rend invalide le CSP courant, ce qui amène à l'exploration de la pile *"Stack_spaces"* afin de revenir en arrière et tenter de résoudre le problème.

Contrairement à l'ajout de contraintes, le solveur crée un nouveau CSP lorsque l'utilisateur insère un nouveau objet dans la scène 3D (algorithme 5). Les dimensions de l'objet inséré, spécifiées par l'utilisateur, serviront à créer de nouvelles variables dans le nouveau CSP. Par conséquent, le système définit un nouveau problème d'agencement à résoudre.

Pour plus de souplesse dans l'utilisation du système, l'utilisateur est en mesure de supprimer une contrainte c_j déjà sélectionnée (algorithme 6) et d'enlever

Algorithm 5 Ajout d'un nouveau objet à la scène 3D

Ajout_nouveau_objet(*dim_x*,*dim_y*,*dim_z*)
Créer_variables(x_{n+1},y_{n+1},z_{n+1})

CSP1 ← **new_CSP**(CurrentCSP)
CSP1.X=CSP1.X$\bigcup\{x_{n+1},y_{n+1},z_{n+1}\}$
%(w,h,d) *are the dimensions of the container*%
CSP1.$\mathcal{D}(x_{n+1})$ ← [*dim_x*,w-*dim_x*]
CSP1.$\mathcal{D}(y_{n+1})$ ← [*dim_y*,h-*dim_y*]
CSP1.$\mathcal{D}(z_{n+1})$ ← [*dim_z*,d-*dim_z*]
X'← CurrentSpace.X
S'← Stack_Spaces
CurrentSpace.**Contraction_des_domaines**()
CurrentSpace.**Propagation_par_contraintes**()

if Est_Failed(CurrentSpace) **then**
    CurrentSpace ← Stack_Spaces.**pop**()
    Envoi_Event(*No_Solution_Found*)
else
    Envoi_Event(*Solution_Found*)
    Stack_Spaces.**push**(CurrentSpace)
end if

un objet obj_i de la scène (algorithme 7). Dans les deux cas, le solveur explore alors la pile pour trouver le CSP_k qui satisfait les nouvelles conditions ($CSP_k.X = CurrentSpace.X/\{x_i, y_i, z_i\}$ ou $CSP_k.C = CurrentSpace.C/\{c_j\}$). Si aucun CSP satisfaisant les nouvelles conditions n'est trouvé, le solveur défini un nouveau CSP et se charge de le résoudre.

Dans tous les algorithmes présentés, une notation telle que $CSP_k.X$ retourne l'ensemble des variables X définies dans le CSP_k, $CSP_k.\mathcal{D}$ appelle la fonction $\mathcal{D}$ (qui retourne le domaine d'une variable passée en paramètre) de CSP_k et $CSP_k.C$ renvoie l'ensemble des contraintes C définies dans le CSP_k.

6.8.2 Accessibilité

Afin de permettre l'accès aux composants ou aux postes de travail dans l'EV, nous avons décidé d'illustrer les espaces libres par des objets abstraits (virtuels) dont les dimensions tiennent compte par exemple de la mobilité des opérateurs devant leurs postes de travail. Le solveur considère ces objets abstraits comme des composants à placer tout en ajoutant des contraintes de proximité entre ceux-ci et les postes pour lesquels un accès est recherché. Quant au système de visualisation, il désactive l'affichage de ces objets dans la configuration présentée à l'utilisateur. Le choix d'une telle stratégie est adapté à la forme parallélépipèdique des postes de travail considérés dans la problématique de l'agencement du shelter (voir chapitre 5). la Figure 6.6 illustre l'accès aux postes de travail dans un shelter.

6.8.3 Orientation des objets

Comme le montre la Figure 6.1 (menu de droite), le système permet à l'utilisateur de changer l'orientation d'un objet 3D. Comme les objets traités dans notre cas,

Algorithm 6 Suppression d'une contrainte c_j

```
1: Supprimer_contrainte(c_i)

2: Solution ← false
3: C'← CurrentSpace.C /{c_i}
4: X'← CurrentSpace.X
5: S'← Stack_Spaces

6: while S'≠ ∅ do
7:    S_temp ← S'.pop()
8:    if S_temp.C=C' AND S_temp.X=X' then
9:       CurrentSpace← S_temp
10:      S'← ∅
11:      Solution ← true
12:   end if
13: end while

14: if Solution = true then
15:    Envoi_Event(Solution_Found)
16: else
17:    CSP1 ← new_CSP(CurrentCSP)
18:    CSP1.C ← CSP1.C/{c_i}
19:    CSP1.Propagation_par_contraintes()
20:    if Est_Failed(CSP1) then
21:       Envoi_Event(No_Solution_Found)
22:    else
23:       CurrentSpace← CSP1
24:       Envoi_Event(Solution_Found)
25:    end if
26: end if

27: Stack_Spaces.push_back(CurrentSpace)
```

ont une forme parallélépipèdique, quatre rotations sont proposées : 0, $+90$, 180, et -90. Par défaut, chaque objet est inséré avec une rotation nulle. Pour les rotations de $+90$ et -90, le solveur crée un nouveau CSP (à partir du CSP courant) dans lequel les domaines des variables x_i et z_i sont permutés. Le solveur valide la rotation effectuée lorsque tous les domaines des variables sont non-vides, sinon la rotation est annulée. Pour des cas particuliers comme le placement contre les murs, le solveur oriente automatiquement les objets vers l'intérieur de la scène. Par exemple, pour un placement d'objet contre le mur droit, le solveur effectue une rotation de $+90$ sur l'objet en question. D'autres orientations basiques sont également proposées par le système comme par exemple le placement des objets dépendants (un bureau et une chaise).

6.8.4 Éclairage des objets

Comme dit dans le chapitre précédent, des aspects liés au confort doivent être pris en compte, comme par exemple l'éclairage des postes de travail. Le système développé permet à l'utilisateur de spécifier si un poste de travail particulier doit être éclairé. Le système de visualisation 3D créé alors un objet virtuel de type "lampe"

Algorithm 7 Suppression d'un objet obj_i

```
1: Supprimer_objet(obj_i)
2: Solution ← false
3: X'← CurrentSpace.X /{x_i,y_i,z_i}
4: C'← CurrentSpace.C
5: S'← Stack_Spaces
6: while S'≠ ∅ do
7:    S_temp ← S'.pop()
8:    if S_temp.C=C' AND S_temp.X=X' then
9:       CurrentSpace← S_temp
10:      S'← ∅
11:      Solution ← true
12:   end if
13: end while
14: if Solution = true then
15:   Envoi_Event(Solution_Found)
16: else
17:   CSP1 ← new_CSP(CurrentCSP)
18:   CSP1.X ← CSP1.X/{x_i,y_i,z_i}
19:   CSP1.Propagation_par_contraintes()
20:   if Est_Failed(CSP1) then
21:      Envoi_Event(No_Solution_Found)
22:   else
23:      CurrentSpace← CSP1
24:      Envoi_Event(Solution_Found)
25:   end if
26: end if
27: Stack_Spaces.push_back(CurrentSpace)
```

qui sera placé par le solveur au dessus du poste concerné. Le solveur considère l'objet "lampe" comme un objet à placer et définit une relation de dépendance entre celui-ci et le poste à éclairer.

6.8.5 Assistances visuelles statiques et dynamiques

Pour rendre le système plus efficace et intuitif, nous avons décidé de fournir une assistance temps-réel et "à priori" avant même le placement manuel d'un objet donné. Le solveur "interviendra" non pas après le placement effectif d'un objet mais au moment où l'utilisateur décide de placer lui même cet objet. Deux types d'assistance sont proposés :

- *Assistance visuelle statique*
  L'utilisateur est en mesure d'identifier les zones 3D où le positionnement de l'objet provoque une violation de contraintes. Le solveur identifie ces zones et transmet les informations nécessaires à l'EV afin de les visualiser sous forme de parallélépipèdes 3D marqués par une couleur spéciale. Une description détaillée de l'implémentation de cette approche est donnée dans le chapitre

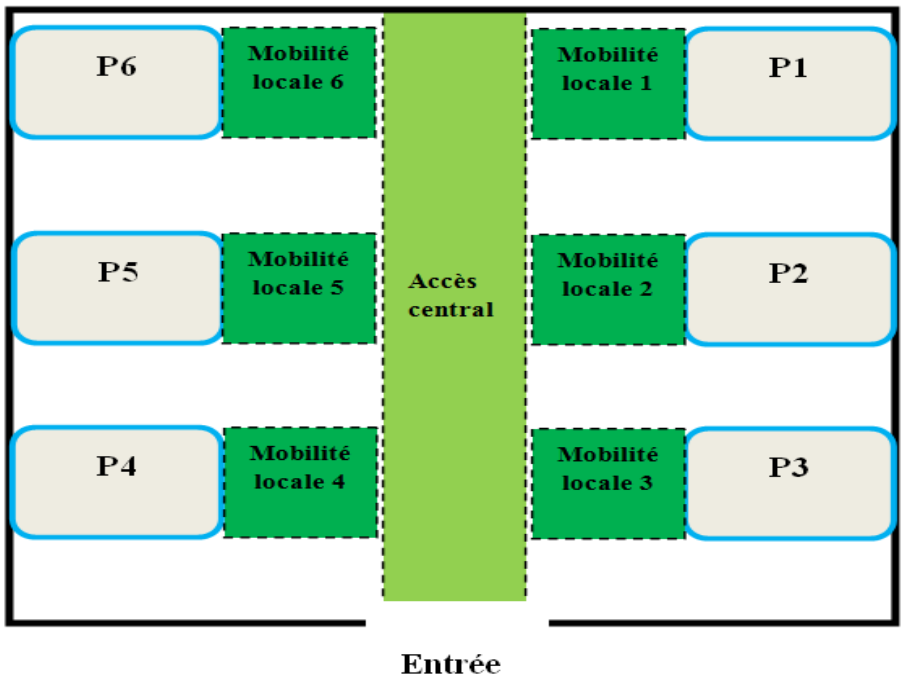


FIGURE 6.6 – *Exemple d'utilisation des objets abstraits (objets en pointillés) pour accéder aux postes de travail (P1 à P6) d'un shelter.*

7. Une étude expérimentale a été également réalisée afin d'étudier l'effet de la visualisation des zones impossibles sur les performances des utilisateurs durant une tâche d'agencement 3D. Les résultats de cette étude sont donnés dans le chapitre 10.

- *Assistance visuelle dynamique*
  Dans notre contexte, une assistance visuelle est dite dynamique quand elle survient en tenant compte de la position courante d'un objet donné (pour qui une assistance est demandée). Trois formes d'assistance dynamique sont proposées. La première consiste à vérifier en temps-réel si la position courante d'un objet est valide ou non. La deuxième vérifie, en cas de position courante invalide, les postions "voisines" pour en trouver une qui satisfasse toutes les contraintes. Ainsi, l'utilisateur place manuellement l'objet dans une des positions proposées. Enfin, la troisième forme présente une amélioration de la deuxième puisque elle permet de déplacer automatiquement un objet vers une position proposée par le solveur. L'ensemble des assistances dynamique est détaillé dans le chapitre 7.

6.9 Conclusion

Ce chapitre a constitué une mise en oeuvre de l'approche de résolution, basée sur un couplage entre programmation par contraintes et réalité virtuelle et proposée dans le chapitre précédent. Le système développé, qui illustre l'intégralité de l'approche, peut être utilisé pour résoudre différentes applications industrielles. Une version spécifique et adaptée au problème d'agencement 3D d'un shelter est en cours de développement. Les composants impliqués dans ce problème sont de forme parallélépipédique. L'interaction proposée avec le système développé ainsi

que les temps relativement faibles des calculs, permettent une certaine souplesse du système. En effet, l'utilisateur est capable de (1) changer la dimension du problème en ajoutant/supprimant des composants/contraintes et (2) choisir le mode de résolution (agencement automatique ou assistance). Lors d'un placement manuel de composants, l'utilisateur est en mesure de déclencher différents moyens d'assistance en vue de l'orienter vers les possibilités de placement. Le chapitre suivant sera consacré à la présentation et l'illustration de ces moyens. Le système ainsi que certaines assistances seront testés et évalués dans la troisième partie de cette thèse.

7

Assistances visuelles anticipées

7.1 Introduction

Malgré l'intérêt des approches antérieures proposées pour la résolution des problèmes d'agencement, l'assistance de l'utilisateur lors de son interaction avec la scène 3D à agencer n'a pas été privilégiée ([22, 54, 131, 147, 55]). En effet, dans la plupart de ces approches, l'assistance proposée survient après le placement effectif des objets. Comme décrit précédemment, notre système permet la résolution d'un problème d'agencement en proposant des configurations avec lesquelles l'utilisateur peut interagir. Le système proposé est capable d'annuler ou rejeter un placement manuel d'objets s'il conduit à une violation de contraintes.

Dans l'objectif de concevoir et présenter un système plus efficace et générique, nous avons décidé de fournir une assistance "prédictive", visuelle et temps-réel avant même de placer un objet donné. Différentes formes d'assistance sont présentées dans ce chapitre et sont toutes basées sur le mécanisme de ***look ahead*** [39], très utilisé dans la recherche combinatoire.

7.2 Le mécanisme d'anticipation (*look ahead*)

7.2.1 Définition

Considéré comme une amélioration de l'algorithme *simple retour-arrière*, le ***look ahead*** est le terme générique de toute méthode consistant à anticiper l'effet de l'affectation d'une variable (affectation partielle) sur les domaines des variables qui ne sont pas encore affectées. Après propagation, si le domaine d'une variable non affectée devient vide et donc il ne contient plus de valeur pouvant être localement consistante avec l'affectation partielle, alors il est inutile de continuer à explorer cette branche, et l'on peut retourner en arrière pour voir d'autres possibilités.

Parmi les techniques de ***look ahead*** nous pouvons citer le ***Forward Checking*** [73] et le ***Full Look Ahead*** [66]. La mise en oeuvre de ces techniques repose sur un fil-

trage des domaines des variables non instanciées. Le but de ce filtrage est de réduire les domaines (suppression de valeurs) pour conserver que les valeurs qui sont localement consistantes, c'est-à-dire celles qui sont susceptibles d'appartenir à une solution. Différents niveaux de filtrages peuvent être effectués pour réduire plus ou moins les domaines des variables mais peuvent aussi prendre du temps à s'exécuter. Un compromis judicieux doit donc être établi entre le coût d'une exploration basée sur le ***look ahead*** et les effets bénéfiques de celle-ci.

7.2.2 La technique du "Forward Checking"

Étant donnée une affectation partielle A et une variable x_i à instancier, cette technique vérifie s'il existe une valeur v_i de x_i pour laquelle l'affectation $A \bigcup x_i \leftarrow v_i$ est consistante. La vérification de la consistance s'établie en considérant que les variables non instanciées et liées à x_i par des contraintes. En d'autres termes, cette technique tente de renforcer la consistance d'arc *locale* en supprimant les valeurs de variables liées à x_i pour lesquelles A ne peut pas être étendue.

Afin d'illustrer le principe du ***Forward Checking***, considérons l'exemple défini par le triplet (X, D, C) :

- $X = \{x_1, x_2, x_3, x_4\}$,
- $\mathcal{D}(x_1) = \mathcal{D}(x_2) = \mathcal{D}(x_3) = \mathcal{D}(x_4) = [1, 5]$
- $C = \{c_1, c_2, c_3, c_4\}; c_1 = \{x_1 < x_2\}, c_2 = \{x_2 < x_3\}, c_3 = \{x_2 < x_4\} c_4 = \{x_4 < x_3\}$

Dans cet exemple, $x_1 \leftarrow 1$ et x_2 constituent respectivement l'affectation partielle et la variable à instancier. Comme le montre la Figure 7.1.a, la technique de ***Forward Checking*** tente d'affecter la valuer 2 à x_2 tout en renforçant la consistance d'arc locale, c'est-à-dire en considérant que les variables directement liées à x_2.

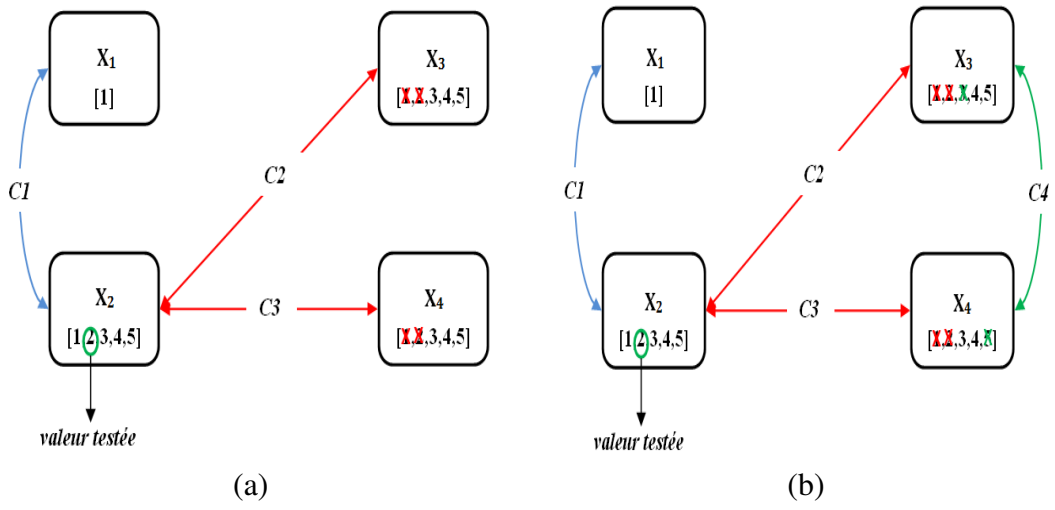


FIGURE 7.1 – *Exemples de techniques de look ahead : Forward Checking (a) et Full Look ahead (b).*

7.2.3 La technique "Full Look Ahead"

Pour une affectation partielle A et une variable x_i à instancier et contrairement au ***Forward Checking***, la technique de ***Full Look Ahead*** tente d'établir la consistance d'arc *globale* en considérant non seulement les variables non instanciées et liées à x_i, mais l'ensemble de variables non encore instanciées. Les valeurs de variables ne conduisant pas à une consistance d'arc globale sont supprimées. Bien que cette méthode soit plus efficace en termes de réduction de domaine, sa mise en ouvre est plus coûteuse en temps de calcul.

En se basant sur l'exemple décrit dans la section précédente, la Figure 7.1.b illustre les réductions des domaines des variables quand la technique de ***Full Look Ahead*** est utilisée. Dans ce cas, la technique teste la valeur 2 de x_2 et établie la consistance d'arc en prenant en compte toutes les variables non encore instanciées.

7.2.4 Objectif

Les résultats obtenus par le mécanisme d'anticipation sont généralement utilisés pour décider de la prochaine variable à évaluer ainsi que l'ordre des valeurs à donner à cette variable [39]. Pour toutes les valeurs d'une variable v_i non affectée, le ***look ahead*** prédit l'effet de l'affectation de v_i à chacune de ses valeurs. Le choix de la variable à évaluer et le choix de sa valeur sont complémentaires : la valeur est typiquement choisie pour trouver une solution (si elle existe) le plus rapidement possible, tandis que la variable suivante est choisie pour détecter une éventuelle inconsistance aussi rapidement que possible.

Dans notre contexte, nous avons décidé d'utiliser la technique de ***Forward Checking*** pour proposer différentes formes d'assistance durant des tâches d'agencement. Quand un utilisateur demande une assistance, celle-ci doit être fourni d'une manière quasi temps réel afin de suivre dynamiquement les actions de l'utilisateur dans l'EV. C'est pour cette raison que nous avons privilégie l'utilisation de la technique de ***Forward Checking***.

7.3 Assistance visuelle statique : les zones impossibles

Comme dit précédemment, un ***look ahead*** permet réduire les domaines de variables et donc de détecter ou de prévoir une possible inconsistance. C'est sur ce principe que se base l'utilisation du ***look ahead*** dans la recherche des zones impossibles. Nous considérons les zones impossibles d'un objet obj_i comme les surfaces 3D non susceptibles de contenir obj_i, à cause d'une violation de contrainte(s). Le solveur identifie ces zones et transmet les informations nécessaires à l'EV afin de les visualiser sous forme de parallélépipèdes 3D, physicalisés et identifiés par une couleur rouge.

Pour rechercher les zones impossibles pour un objet obj_i, une décomposition de l'espace de travail (la scène à agencer) en plusieurs zones 3D doit être effectuée. Chaque zone obtenue est définie par six paramètres :

- $[Z_{xmin}, Z_{xmax}] \Longrightarrow$ *délimitation de la zone sur l'axe X*,

Algorithm 8 Zones impossibles pour l'objet obj_i

Affichage_zones_impossible(obj_i)
Décompostion_Scene_3D()*%décomposer la scène en n zones%*
Zi_{xmin}, Zi_{xmax}, Zi_{ymin}, Zi_{ymax}, Zi_{zmin},Zi_{zmax} / i in $\in [1..n]$ *sont les paramètres de la zone* Z_i

for all $j \in [1..n]$ **do**
    CSP_temp $\leftarrow$ CurrentCSP
    CSP_temp.$\mathcal{D}(x_i) \leftarrow [Zj_{xmin}, Zj_{xmax}]$
    CSP_temp.$\mathcal{D}(y_i) \leftarrow [Zj_{ymin}, Zj_{ymax}]$
    CSP_temp.$\mathcal{D}(z_i) \leftarrow [Zj_{zmin}, Zj_{zmax}]$
    CSP_temp.**Contraction_domaines**(x_i, y_i, z_i)
    CSP_temp.**Propagation_de_contraintes**()

    if Est_Failed(CSP_temp) **then**

        L'objet obj_i ne peut être placé quelque part dans la zone Zj
        Affichage_zone_impossible_dans_EV(Zj)
    else

        L'objet obj_i peut être placé quelque part dans la zone Zj
    end if

end for

- $[Z_{ymin}, Z_{ymax}] \Longrightarrow$ *délimitation de la zone sur l'axe Y,*
- $[Z_{zmin}, Z_{zmax}] \Longrightarrow$ *délimitation de la zone sur l'axe Z.*

Pour chacune des zones créées, le solveur crée un nouveau CSP (à partir du CSP courant) dans lequel les variables relatives à obj_i (x_i, y_i et z_i) ont chacune un nouveau domaine :

- $\mathcal{D}(x_i)$=[Z_{xmin},Z_{xmax}],
- $\mathcal{D}(y_i)$=[Z_{ymin},Z_{ymax}],
- $\mathcal{D}(z_i)$=[Z_{zmin},Z_{zmax}].

Ensuite, un ***look ahead*** agissant sur trois variables non instanciées (x_i, y_i et z_i) ainsi sur celles qui leurs sont liées, est utilisé pour "forcer" la consistance locale en supprimant les valeurs dont le choix conduit à une inconsistence. Le solveur vérifie alors si le CSP créé est localement consistant ou non (zone impossible). Le choix de trois variables candidates à l'instanciation est hérité du fait que dans notre contexte, la position de chaque objet 3D est défini par trois paramètres.

A travers ce type d'assistance, l'utilisateur est guidé par une information visuelle afin de mieux placer (manuellement) un objet donné. L'approche proposée est illustrée par l'Algorithme 8.

Il est à noter que le temps requis par le solveur pour la recherche des zones impossibles dépend fortement de la décomposition effectuée sur la scène 3D. En effet, plus le nombre de zones à tester est grand, plus le temps de calcul est long. A titre d'exemple, le solveur Gecode nécessite moins d'une seconde pour tester 1000 zones ($10 \times 10 \times 10$). Le même principe est valable pour la précision ; une décom-

position précise de la scène 3D (un grand nombre de zones) assure une meilleure identification des zones impossibles.

L'identification des zones impossibles pour un objet obj_i ne signifie pas que toutes les autres zones sont des placements valides pour obj_i ; ceci dépend principalement de la précision de la décomposition effectuée.

Pour accélérer l'identification des zones impossibles, une recherche dichotomique est utilisée pour ne pas tester certaines zones et donc alléger la tâche du solveur et accélérer les temps de calculs. En effet, l'espace à agencer est réparti en deux grandes zones suivant un des axes x, y, z. Chaque zone est testée afin de vérifier si elle est susceptible de contenir l'objet à placer. Si l'objet ne peut pas être placé dans la zone testée, celle-ci est ignorée par le solveur, sinon elle sera décomposé en deux autres zones. Ce processus itératif s'arrête après un nombre prédéfini de décomposition. Des heuristiques spécifiques sont également utilisées, par exemple si un objet doit être placé sur le sol, le solveur élimine toutes les zones dont le paramètre Z_{ymin} dépasse la hauteur de l'objet en question.

Voici un exemple simple où la chaise du bureau doit être placée sur le sol et à droite du bureau. Le système propose les zones rouges comme celles impossibles pour le placement de la chaise (Fig 7.2).

7.4 Assistance visuelle dynamique

L'affichage des zones impossibles pour un objet donné constitue une assistance considérable et utile permettant à l'utilisateur de prendre une décision plus au moins correcte quant au placement de cet objet. Néanmoins, et comme décrit précédemment, les zones qui ne sont pas considérées comme impossibles ne constituent pas forcément un placement correct pour un objet donné et ne tiennent pas compte de la position courante de cet objet. A moins que la scène à agencer soit décomposée en des très petites "zones" (ce qui augmente le temps de calcul), placer un objet en dehors des zones rouges n'exclut pas la possibilité de violer certaines contraintes. C'est pour cette raison qu'une assistance visuelle dynamique, tenant compte de la position courante des objets, s'est avérée indispensable.

Basée sur le mécanisme de ***look ahead***, notre approche consiste à prévoir l'effet d'un choix de placement sur le CSP courant. Une communication permanente entre le solveur et l'EV a été établie afin de vérifier si la position courante d'un objet pouvait constituer un placement correct de ce dernier. L'utilisateur peut déplacer un objet donné dans l'espace 3D, le solveur vérifie dynamiquement la validité de chaque position de cet objet. Une position correcte ou valide pour un objet obj_i est une position pour laquelle la consistance d'une affection partielle (instantiation de x_i, y_i et z_i) peut être étendue pour toutes les variables non encore affectées.

Afin d'anticiper l'effet du placement d'un objet obj_i dans une position donnée (px_i, py_i, pz_i) tout en conservant l'état du CSP courant, le solveur crée une copie de ce dernier où les variables relatives à obj_i (x_i, y_i et z_i) sont affectées et ont chacune un nouveau domaine :

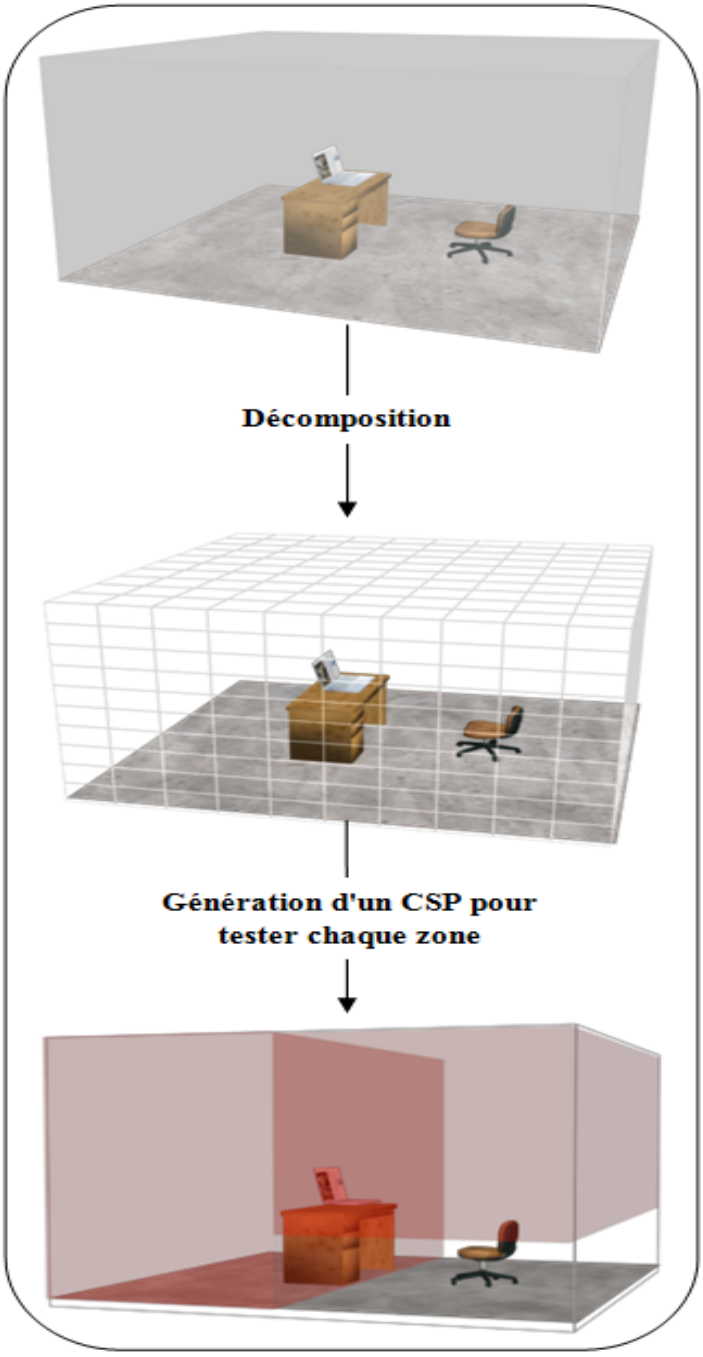


FIGURE 7.2 – *Visualisation des zones impossibles pour la chaise du bureau.*

Algorithm 9 Assistance visuelle dynamique

1: **Affichage_zones(**obj_i**)**
2: (px_i,py_i,pz_i) *la position courante de* obj_i *dans l'espace 3D*
3: **while** L'objet obj_i est déplacé **do**
4: CSP_temp ← CurrentCSP
5: CSP_temp.$\mathcal{D}(x_i) \leftarrow \{px_i\}$
6: CSP_temp.$\mathcal{D}(y_i) \leftarrow \{py_i\}$
7: CSP_temp.$\mathcal{D}(z_i) \leftarrow \{pz_i\}$
8: CSP_temp.**Contraction_domaines**(x_i, y_i, z_i)
9: CSP_temp.**Propagation_de_contraintes**()
10: **if Est_Failed**(CSP_temp) **then**
11: L'objet obj_i ne peut pas être placé dans la position courante
12: **Affichage_zone_rouge(**px_i,py_i,pz_i**)**
13: **else**
14: L'objet obj_i peut être placé dans la position courante
15: **Affichage_zone_verte(**px_i,py_i,pz_i**)**
16: **end if**
17: **end while**

- $\mathcal{D}(x_i)$=$\{px_i\}$,
- $\mathcal{D}(y_i)$=$\{py_i\}$,
- $\mathcal{D}(z_i)$=$\{pz_i\}$.

Suite à cette affectation partielle, le mécanisme du ***look ahead*** est utilisé pour supprimer les valeurs des variables qui ne conduisent pas à une consistance. Si le domaine d'une variable non affectée devient vide, le CSP créé devient invalide et par conséquent la position définie par px_i, py_i, pz_i est incorrecte. L'objet obj_i est alors coloré en rouge pour illustrer l'invalidité de sa position. En revanche, il sera coloré en vert si le CSP créé reste valide.

L'approche proposée (algorithme 9) fournit une aide visuelle en temps-réel permettant de caractériser n'importe quelle position dans l'espace 3D à agencer. Dans ce cas, et lors d'un placement manuel, le risque de violation de contraintes est nul. Ce type d'assistance est plus efficace (pas de risque de violation de contraintes) et plus rapide que le premier type. En effet, le solveur vérifie la validité d'une seule zone (position) à un instant donné. Tester si une position est correcte ou non ne prend que quelques micros secondes, permettant ainsi un affichage temps-réel de l'état de chaque position dans l'espace 3D.

La Figure 7.3 présente un exemple dans lequel un utilisateur veut placer manuellement un ordinateur portable. Supposant qu'une contrainte de type *Objet_sur_objet* relie l'ordinateur avec le bureau. Le système effectue alors un suivi temps-réel des déplacements effectués par l'utilisateur, en affichant l'état de chaque position (valide ou non).

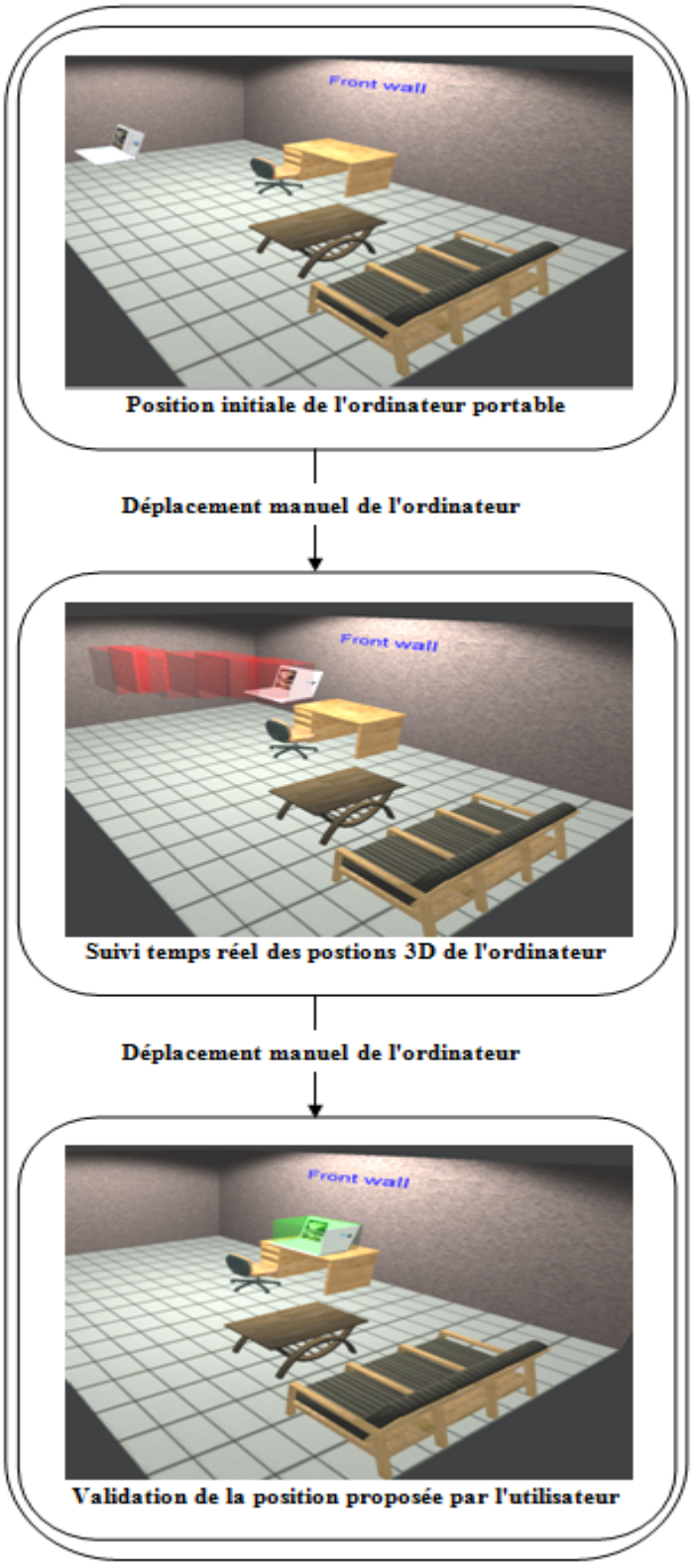


FIGURE 7.3 – *Caractérisation temps réel de chaque position 3D proposée par l'utilisateur.*

7.5 Assistance visuelle dynamique améliorée

Partant du fait que l'assistance dynamique présentée précédemment n'est activée qu'à la suite d'un déplacement d'un objet donné, cette assistance peut être insuffisante dans certains cas où l'utilisateur ne déplace pas l'objet dans des zones susceptibles de contenir ce dernier, le système affichera alors que des zones rouges. Souvent, il se trouve qu'une position valide est très proche de la position courante de l'objet. Afin d'améliorer l'assistance proposée, le système anticipe le placement d'un objet en regardant les positions se trouvant autour de la position courante. De cette façon, même si la position courante d'un objet est invalide, le système peut proposer des alternatives de placement. L'identification des positions voisines est basée sur un échantillonnage réalisé sur l'ensemble des positions les plus proches de la position courante. Pour le moment, la manière d'échantillonnage est basique puisqu'elle consiste à choisir arbitrairement les n positions les plus proches. Dans l'avenir, nous utiliserons un échantillonnage basé sur une méthode plus avancée. Pour se faire, il faut transformer le problème en un problème d'optimisation où la fonction objectif, à minimiser, porte sur le nombre de contraintes violées.

Le fait de partir d'une position courante et de chercher une solution voisine laisse penser que nous utilisons la recherche locale plutôt qu'une méthode d'anticipation ce qui n'est pas vraiment le cas. En effet, dans la recherche locale on part d'une solution courante à une autre dans l'espace de recherche jusqu'à ce qu'une solution qualifiée comme optimale est trouvée. Tandis que dans l'assistance proposée dans cette section, la recherche part d'une position courante invalide (donc ce n'est pas une solution) et anticipe l'effet du placement d'un objet dans une position voisine sans chercher une optimalité quelconque.

Toujours en utilisant le mécanisme du ***look ahead***, le principe de cette approche consiste à prévoir l'effet de plusieurs choix de placements (un par un) sur le CSP courant. Cette approche est donc basée sur plusieurs appels de la méthode présentée ci-dessus (*Assistance visuelle dynamique*). Le solveur commence par vérifier la validité de la position courante d'un objet. Si celle-ci est invalide, il teste séquentiellement les postions les plus proches de la position courante, et arrête la recherche dés qu'il en trouve une valide. Le nombre de positions "voisines" à tester a peu d'influence sur les temps des calculs. Pour chacune des positions testées, le solveur crée une copie du CSP courant.

L'assistance visuelle dynamique améliorée (algorithme 10) fournit une meilleure aide permettant non seulement de caractériser n'importe quelle position d'un objet mais aussi de "pousser" ce dernier vers les zones où il peut être placé sans violation de contraintes.

Afin d'illustrer ce type d'assistance, nous supposons dans l'exemple présenté dans la Figure 7.4 que la table de salon doit être placée à une distance fixe du canapé (contrainte *Distance_fixe*). Dés que l'utilisateur déplace la table à proximité du canapé, le système lui propose une position possible afin de lui éviter de se soucier de la distance qui sépare les deux objets. Dans cet exemple, le système a parcouru 18 positions voisines pour chercher la position valide la plus proche de la position courante.

Algorithm 10 Assistance visuelle dynamique améliorée

Affichage_zones(obj_i)
POS=$\{P_j / j \in [0..n]\}$ *pour* $j > 1$, P_j *est la* j^{me} *position la plus proche de la position courante*
($P_0.px$,$P_0.py$,$P_0.pz$) *la position courante de* obj_i *dans l'espace 3D*

while L'objet obj_i est déplacé **do**
  for all $k \in [0..n]$ **do**
    CSP_temp $\leftarrow$ CurrentCSP
    CSP_temp.$\mathcal{D}(x_i) \leftarrow \{P_k.px\}$
    CSP_temp.$\mathcal{D}(y_i) \leftarrow \{P_k.py\}$
    CSP_temp.$\mathcal{D}(z_i) \leftarrow \{P_k.pz\}$
    CSP_temp.**Contraction_domaines**(x_i, y_i, z_i)
    CSP_temp.**Propagation_de_contraintes**()
    if Est_Failed(CSP_temp) **then**
      L'objet obj_i ne peut pas être placé dans cette position
      Affichage_zone_rouge($P_k.px$,$P_k.py$,$P_k.pz$)
    else
      L'objet obj_i peut être placé dans cette position
      Affichage_zone_verte($P_k.px$,$P_k.py$,$P_k.pz$)
      k=-1
    end if
  end for
end while

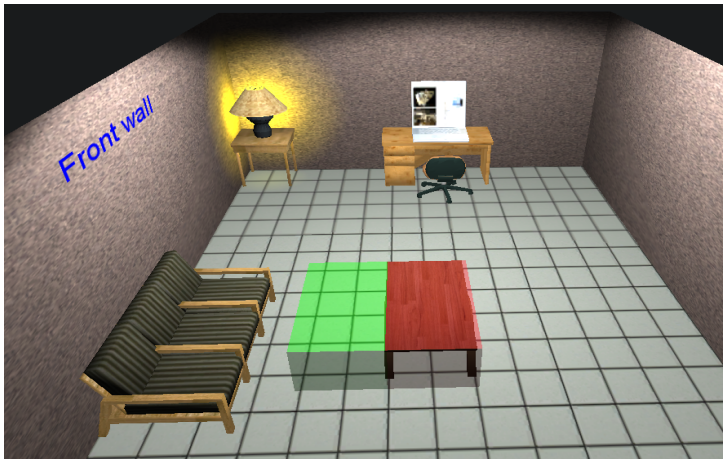


FIGURE 7.4 – *Proposition d'une alternative de placement de la table dont la position courante est invalide.*

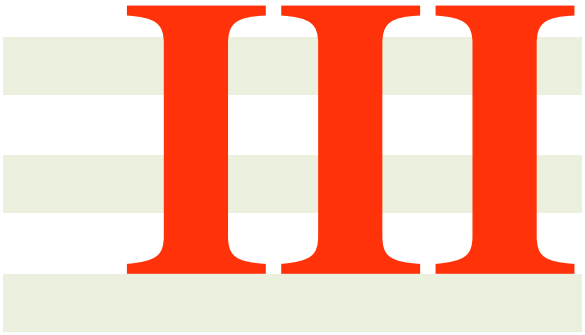
7.6 Assistance visuelle dynamique avec placement automatique

Pour solliciter l'utilisateur le moins possible durant une tâche d'agencement, l'assistance visuelle améliorée peut être combinée avec le placement automatique que peut réaliser le solveur comme décrit dans le chapitre 6. En effet, l'utilisateur peut activer l'assistance visuelle pour avoir des alternatives de placement pour un objet obj_i, et par la suite il peut faire appel aux autres fonctionnalités du sol-

veur pour déplacer automatiquement obj_i vers la position proposée. Cette combinaison peut être appliquée pour les autres assistances visuelles présentées dans ce chapitre. L'utilisateur peut, à tout moment, demander un placement automatique d'un ensemble d'objets et activer l'assistance visuelle s'il veut placer manuellement d'autres objets. Dans les deux cas, le module de résolution de contraintes "accompagne" et assiste un utilisateur durant un agencement 3D.

7.7 Conclusion

Ce chapitre a été consacré à la présentation de différentes formes d'assistances dynamiques proposées pour assister l'utilisateur lors d'un agencement spatial 3D. Après une revue des approches antérieures, nous avons proposé quatre méthodes basées sur le mécanisme du ***look ahead*** afin d'anticiper l'effet des décisions de l'utilisateur (choix de placement) sur la satisfaction des contraintes requises pour l'agencement. Une étude expérimentale a été réalisée pour étudier l'effet de l'une des assistances proposées sur la performance d'un utilisateur durant une tâche d'agencent 3D. Les résultats sont détaillés dans le chapitre 10. Dans le chapitre suivant, nous entamerons la partie expérimentale dédiée à l'évaluation du système proposé. Nous évaluons la capacité du système à apporter une aide considérable et diversifiée (placement automatique + affichage des zones) à l'utilisateur durant des tâches d'agencement 3D.



Expérimentations

8

Évaluation du système préliminaire

8.1 Introduction

L'agencement manuel des environnements virtuels (EVs) restreints est généralement fastidieux, long, frustrant et peu précis quand il s'agit de satisfaire un ensemble de contraintes. En effet, différents types d'erreurs peuvent survenir durant ce type de tâche telles que les erreurs de placement et les erreurs d'interaction 3D.Pour surmonter ces difficultés et pour assister l'utilisateur, nous avons proposé l'utilisation d'un solveur de contraintes que nous avons intégré dans une application 3D temps réel simulant un espace restreint composé d'objets virtuels.

Bien que les résultats de certains travaux antérieurs [16, 34, 54, 147] ont montré que l'utilisation de la programmation par contraintes (PPC) permettait la résolution automatique d'un problème d'agencement, très peu d'informations ont été données sur le temps de calcul et aucune comparaison entre l'agencement manuel (sans solveur) et l'agencement semi-automatique (avec solveur) n'a été effectuée.

L'étude expérimentale présentée dans le chapitre courant, a été réalisée certes pour évaluer le système proposé mais surtout pour savoir dans quelle mesure les problématiques suivantes ont été résolues :

- **Problématique logicielle** : comment intégrer un système de résolution de contraintes dans un système de visualisation et d'interaction 3D ? et comment gérer les échanges entre les deux systèmes ?
- **Problématique interactionnelle/communicationnelle** : comment l'utilisateur peut-il commander le solveur ?

8.2 Objectif

Une étude comparative a été réalisée sur les deux types d'approches (agencement manuel et semi-automatique) avec deux niveaux de complexité (simple et

complexe). Différents facteurs tels que le temps de réalisation des tâches et la précision des placements ont été mesurés. L'étude vise à analyser l'influence de l'intégration d'un module intelligent (solveur de contraintes) dans un EV sur la performance de l'utilisateur au cours d'une tâche d'agencement 3D. L'expérimentation consistait à placer différents objets dans un espace 3D restreint tout en respectant un certain nombre de contraintes.

L'objectif de l'étude présentée dans ce chapitre est de vérifier les trois hypothèses suivantes :

1. *H1 :* Les sujets agenceront plus rapidement la scène quand le solveur est utilisé.
2. *H2 :* Les sujets feront moins d'erreurs de placement et d'interaction 3D quand le solveur est utilisé.
3. *H3 :* La méthode proposée pour l'interaction avec le solveur est facile et intuitive.

Pour vérifier l'hypothèse *H1*, nous avons mesuré et comparé le temps de réalisation dans chaque tâche d'agencement (manuelle et semi-automatique). La comparaison des erreurs survenues dans chacune d'elles a permis d'étudier l'hypothèse *H2*. Une mesure subjective de la charge de travail a été également réalisée en utilisant un questionnaire basé sur la NASA-TLX (Task Load Index, développée à la NASA) [75]. Une partie des réponses à ce questionnaire reflète le niveau de la difficulté ressenti lors de l'interaction avec le solveur et permet donc d'étudier l'hypothèse *H3*.

8.3 Protocole expérimental

L'expérimentation consistait à placer un ensemble de douze objets génériques (quatre cubes, quatre sphères et quatre cônes) dans un espace présenté sous forme d'un grand cube 3D (Fig 8.1). Les contraintes d'agencement sont affichées sur le haut de l'écran sous forme textuelle. Étant donné que deux niveaux de complexité sont considérés, chaque sujet été amené à tester les deux types d'agencement (sans et avec utilisation du solveur) pour chaque niveau. Les quatre sous-tâches à effectuer sont détaillées dans la Section 8.4.

Vingt deux sujets, âgés de 22 à 47 ans, ont accepté de participer à l'expérimentation. Nous leurs avons demandé d'évaluer leurs niveaux d'expérience avec les jeux vidéo et la réalité virtuelle (RV). Douze sujets ont indiqué qu'ils avaient joué régulièrement aux jeux vidéo. Concernant leurs expériences en RV, sept sujets ont prétendu qu'ils avaient une expérience modérée, tandis que huit sujets ont évalué leur expérience comme très faible et sept comme très haute.

L'expérience a été réalisée sur la plate-forme de réalité virtuelle de l'ISTIA (Université d'Angers) [114]. Un menu 3D a été implémenté (Fig 8.1) pour permettre à l'utilisateur une introduction facile des objets (présentés en groupes) dans l'EV. Une Nintendo Wiimote™ a été utilisée comme un dispositif d'entrée pour la sélection/désélection et le déplacement des objets dans l'espace 3D. Pour suivre la position de la main d'un sujet dans l'espace (Fig 8.2), la Wiimote™ a été équipée

de deux marqueurs infrarouge et d'une caméra OptiTrack™ [108]. Un Nunchuk™ a été employé (Fig 8.3) pour l'interaction avec le solveur : sélection, validation des contraintes et déclenchement de résolution. L'expérimentation a été réalisée avec un ordinateur ayant un processeur *Intel*™ $i7 - 2630$ de fréquence $2.00Ghz$.

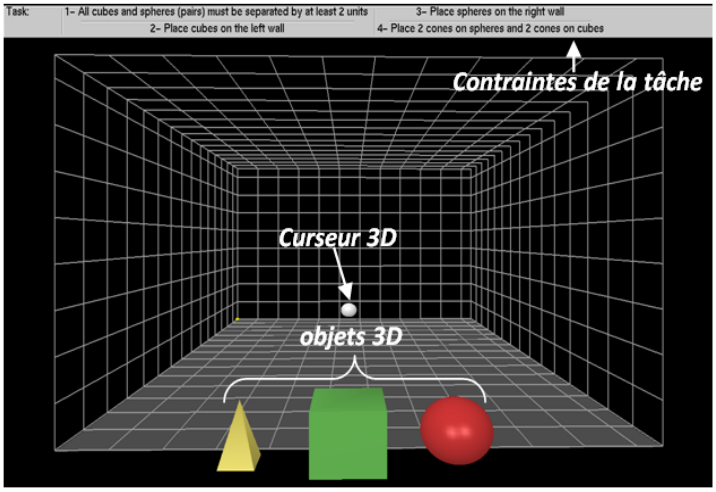


FIGURE 8.1 – *Copie d'écran de l'environnement virtuel expérimental.*

Avant le début de chaque sous-tâche, les sujets ont été informés de l'objectif et des étapes à suivre pour accomplir la tâche en question. Des sessions d'essais ont été proposées aux sujets, elles consistaient à accomplir les quatre sous-tâches de l'expérimentation. Dans les deux conditions (sans et avec solveur), ces essais ont permis aux sujets de se familiariser avec l'EV proposé et les dispositifs matériels utilisés. L'ordre de passage des tâches a été contrebalancé afin d'éviter un biais dû au transfert d'apprentissage.

Après chaque essai expérimental, les sujets devaient répondre à un questionnaire permettant d'évaluer la charge de travail (workload) associée à chaque essai. Des questions supplémentaires ont été ajoutées afin d'évaluer le niveau de satisfaction et de l'assistance fournie par le solveur.

8.4 Description des tâches d'agencement

8.4.1 Agencement manuel

Il consiste à placer manuellement les objets dans la scène, dans n'importe quel ordre, en utilisant la Wiimote™. Le sujet devait pour cela sélectionner et déplacer les objets à l'aide d'un curseur 3D dont la position était mesurée à l'aide des données fournies par la caméra OptiTrack™. Pour ce faire, il devait appuyer sur le bouton B de la Wiimote™ et déplacer l'objet vers la position souhaitée. L'objet est relâché lorsque le bouton est relâché (Fig 8.2). Chaque sujet devait réaliser deux agencements manuels avec des niveaux de complexités différents.

Agencement manuel simple (SS)

Chaque sujet était appelé à agencer manuellement l'EV en respectant les contraintes suivantes :

1. *Sur le sol* : Le sujet devait placer tous les objets (4 cubes, 4 spheres and 4 cones) sur le sol de la scène. Aucun ordre de sélection d'objets n'est requis pour cette contrainte. Comme dit dans le chapitre 6, le placement sur le sol constitue un cas particulier de la contrainte "*Objet_sur_objet*".
2. *Distance_minimale* : Le sujet devait placer tous les objets de telle façon qu'ils soient séparés 2 à 2 par une distance ***d_min*** (2 unités). La présentation de l'EV sous forme de grille, permet d'estimer la distance entre les objets.

Agencement manuel complexe (CS)

Comme pour ***SS***, les sujets devaient placer manuellement tous les objets de la scène. Dans ce cas la difficulté de l'agencement est augmentée par l'ajout de trois nouvelles contraintes. La contrainte *Sur le sol* n'est pas considérée dans ce type d'agencement. L'augmentation du nombre des contraintes engendre une augmentation de la difficulté d'interaction 3D avec l'EV. En effet, plus il y a des contraintes, plus les sujets devaient sélectionner les objets et aussi leurs contraintes.

1. *Mur_gauche* : Tous les cubes doivent être placés contre le mur gauche de l'EV.
2. *Mur_droit* : Toutes les sphères doivent être placées contre le mur droit de l'EV.
3. *Objet_sur_objet* : Cette contrainte requière le placement de deux cones chacun sur un cube. Chacun des deux cones restants doit être placé sur une sphère. Il est important de noter que cette contrainte exige un ordre de sélection d'objets : pour placer *l'objet x* on *l'objet y*, il faut sélectionner *l'objet y* en premier.

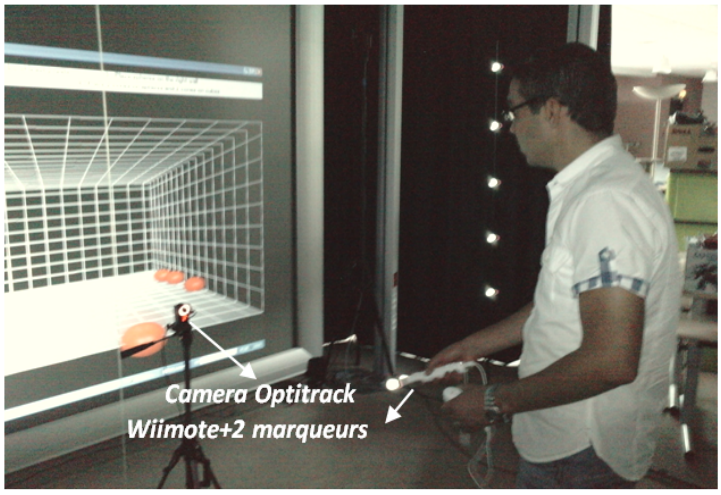


FIGURE 8.2 – *Sujet effectuant un agencement manuel. Une caméra OptiTrack™ et deux réflecteurs positionnés sur l'avant de la Wiimote™, permettent la capture des mouvements du sujet.*

8.4.2 Agencement semi-automatique

Il consiste à interagir avec le solveur pour agencer automatiquement l'EV. Celui-ci doit être informé des contraintes associées à chaque objet. Pour cela, le sujet sélectionne un ou plusieurs objets en appuyant sur le bouton A de la Wiimote™, navigue dans le menu des contraintes en utilisant le joystick du Nunchuk™, puis sélectionne les contraintes souhaitées (bouton Z du Nunchuk™), et finalement appelle le solveur (Fig 8.3). Un agencement simple et un autre complexe ont été proposés aux sujets.

Agencement semi-automatique simple (SA)

Chaque sujet devait respecter les mêmes contraintes décrites dans ***SS***, mais en faisant appel au solveur. Pour appliquer la contrainte de *Distance_minimale*, il faut sélectionner au moins deux objets de l'EV avant de choisir la contrainte en question.

Agencement semi-automatique complexe (CA)

Les mêmes contraintes dans *CS* sont maintenues mais les sujets devaient interagir avec le solveur pour réaliser l'agencement.

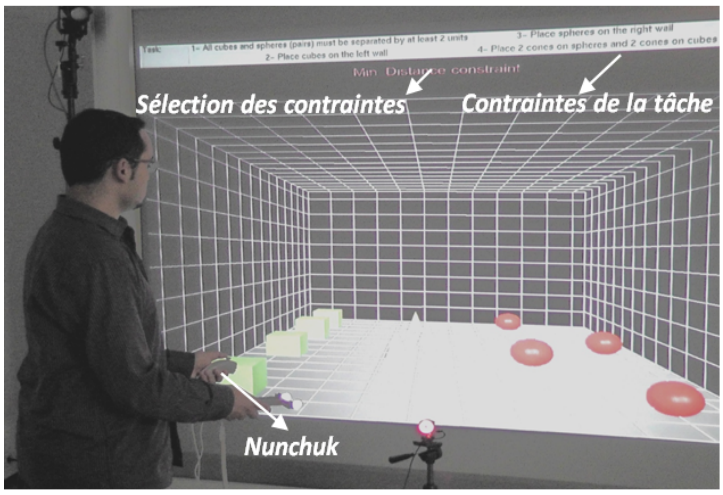


FIGURE 8.3 – *Sujet sélectionnant une contrainte (via le Nunchuk™) à appliquer sur des objets de la scène 3D.*

8.4.3 Données recueillies

Pour étudier et analyser l'effet de l'utilisation d'un solveur de contraintes sur les performances des utilisateurs durant différentes types d'agencements 3D, nous avons mesuré et comparé différents types de données classifiées comme suit.

Données objectives

1. *Le temps d'agencement*
   Le temps d'agencement est le temps entre le premier appui sur le bouton A (Wiimote™) et le moment où le sujet estime que la tâche est accomplie.

2. *Les erreurs d'agencement*
   Ces erreurs sont évaluées à travers trois paramètres :

   Erreur de placement : Ce type d'erreur n'est lié à aucune des contraintes décrites dans la Section 8.4. Pour chaque objet, le système vérifie si oui ou non il est inclus dans la scène en effectuant des vérifications sur les axes X, Y et Z. Nous avons calculé le nombre d'occurrences de cette erreur pour chaque sous-tâche.

   Erreur de distance : Partant du fait que la contrainte *Distance_minimale* est considérée dans les quatre conditions de l'expérimentation, nous avons décidé de calculer l'erreur générée par la violation de cette contrainte. Après la fin de chaque tâche, le système calcule la distance entre chaque paire d'objets. Si la distance calculée est supérieure au seuil (*dmin*), le système incrémente le nombre d'occurrences de ce type d'erreurs.

   Erreur d'interaction 3D : Une erreur d'interaction 3D se produit lorsque le sujet essaie de sélectionner un objet, mais échoue à cause d'une mauvaise perception 3D. Nous avons décidé de mesurer ces erreurs dans chaque condition expérimentale pour étudier dans quelle mesure l'utilisation du solveur permet leurs réduction.

Données subjectives

1. *Charge du travail (Workload)*
   Les sujets devaient répondre à un questionnaire basé sur la NASA-TLX afin d'évaluer le niveau de la charge de travail causé par la tâche. L'évaluation est basée sur six indices supposés indépendantes et envisagés comme sources de charge de travail.

   – *Charge mentale (MD)* : Activité mentale et perceptive nécessaire (par exemple la pensée, le calcul mental, le rappel, la recherche, la décision, etc.).
   – *Charge physique (PD)* : Activité physique nécessaire (par exemple tirer, pousser, déplacer, etc).
   – *Charge temporelle (TD)* : Pression du temps perçu en raison de la vitesse ou le rythme de la tâche.
   – *Performance personnelle (OP)* : Estimation personnelle quant à l'atteinte des objectifs visés.
   – *Effort (EF)* : Niveau de l'effort physique et mental nécessaire pour réaliser un niveau de performance.
   – *Frustration (FR)* : Niveau de découragement, d'irritation et de gêne par rapport à la satisfaction et la réalisation de la tâche.

   Les charges mentale, physique et temporelle concernent les exigences imposées sur le sujet, tandis que les trois autres indices reflètent l'interaction entre les sujets et les tâches.

2. *Satisfaction*
   En utilisant une échelle de Likert à sept points (1 pour "très faible" à 7 pour

"très élevé"), les sujets devaient évaluer leur satisfaction par rapport à l'agencement réalisé.

3. *Assistance*
   En utilisant la même échelle de Likert, les sujets devaient exprimer leur ressentis par rapport à l'assistance fournie par le solveur. Le but était de savoir dans quelle mesure l'utilisation du solveur a apporté une assistance aux sujets et a facilité l'accomplissement des sous-tâches.

   Il est important de noter que les sujets évaluaient leur satisfaction et le niveau de l'assistance fournie que pour les sous-tâches où le solveur était activé (***SA*** et ***CA***). En effet, une évaluation du niveau de l'assistance n'a pas de sens puisque les autres tâches (***SS*** et ***CS***) sont totalement manuelles et ne fournissent aucune forme d'aide. De même, pour les tâches manuelles, les sujets agençaient l'EV selon leurs préférences contrairement aux tâches semi-automatiques, où les décisions du solveur (solutions calculées) ne prennent pas forcément en compte ces préférences. C'est pour cette raison que la question sur la satisfaction est plus pertinente dans le cas où le solveur est utilisé.

8.5 Résultats et discussion

Étant donné que les sujets avaient des niveaux d'expériences différents avec les jeux vidéo et les applications de RV, nous avons effectué un ***t-test***, test généralement utilisé pour vérifier si deux moyennes sont statistiquement égales, afin de vérifier si cette différence affectait leur performance. Les sujets étaient répartis en deux groupes :

– Un groupe G1, regroupe les sujets qui ont une expérience avec des jeux vidéo et des applications de RV (12 sujets)
– Un groupe G2, regroupe les sujets qui n'ont jamais testé les jeux vidéo et les applications de RV (10 sujets)

Afin d'utiliser le ***t-test*** approprié (pour des variances égales ou différentes), un ***F-test*** a été réalisé pour chacune des variables dépendantes (temps d'agencement et erreur d'interaction 3D). En effet, un ***F-test*** permet de vérifier si deux variances sont statistiquement égales. Le Tableau 8.1 résume les valeurs de ***p*** calculées pour le ***F-test*** et le ***t-test***. En se basant sur les valeurs de ***p*** obtenues pour le ***t-test***, il n'y

	SS	**SA**	**CS**	**CA**
F-test	$p = 0.245$	$p = 0.384$	$p = 0.466$	$p = 0.185$
t-test	$p = 0.332$	$p = 0.357$	$p = 0.343$	$p = 0.230$

TABLE 8.1 – *Les valeurs de p obtenues pour chaque tâche de l'expérimentation.*

a pas de différence statistique significative entre les groupes testés (G1 et G2), et par conséquent les performances ne sont pas affectées par la différence des niveaux d'expériences des sujets.

8.5.1 Résultats objectifs

Une analyse de variance (ANOVA) intra-groupes (*within subjects*) à deux facteurs de variabilité a été également réalisée. La complexité de la tâche (***CT***) et l'assistance fournie (***A***) constituent les variables indépendantes, tandis que le temps d'agencement et l'erreur d'interaction 3D représentent les variables dépendantes. Chacun des facteurs de variabilité est défini par deux niveaux : simple et complexe pour ***CT***, sans et avec assistance pour ***A***.

Le Tableau 8.2 illustre les effets des facteurs de variabilité sur les variables dépendantes, ainsi que l'effet d'interaction entre facteurs. Les valeurs de ***p*** et ***F*** montrent que ***CT*** et ***A*** affectent significativement le temps d'agencement et l'erreur d'interaction 3D et qu'il n'y a pas un effet d'interaction significatif entre les variables indépendantes.

Effet de la complexité des tâches (CT) et de l'assistance (A) sur le temps d'agencement :
Contrairement aux tâches simples, dans les tâches complexes, les sujets devaient prendre en compte et satisfaire quatre contraintes impliquant différents objets 3D. Par conséquent, ils ont pris plus de temps pour achever les tâches complexes. De même, l'assistance a significativement affectée le temps d'agencement. Au cours des tâches non assistées, les sujets devaient placer les objets manuellement tout en pensant à la précision des placements et aux contraintes, ils ont donc pris plus de temps pour accomplir les tâches manuelles.

Effet de la complexité des tâches (CT) et de l'assistance (A) sur l'erreur d'interaction 3D :
Le facteur **CT** a produit un effet significatif sur l'erreur d'interaction 3D parce que les sujets devaient sélectionner et appliquer des contraintes/objets 3D dans un large espace. Le nombre d'interactions et donc celui des erreurs d'interaction 3D, dépend du nombre de contraintes requises (complexité de la tâche). L'effet significatif de **A** peut s'expliquer par le fait que pour le premier niveau de **A** (sans le solveur), les sujets devaient sélectionner et placer des objets et affiner fréquemment leurs positions 3D par peur de violer des contraintes. Pour le deuxième niveau de **A** (avec le solveur), les sujets devaient seulement sélectionner des objets et appliquer des contraintes, l'agencement étant laissé à la charge du solveur.

Afin de vérifier si l'hypothèse H1 (Section 8.2) est confirmée ou non, nous avons calculé et comparé le temps moyen d'agencement (Tab 8.3). Les sujets avaient accompli l'agencement sous la tâche ***SA*** après en moyenne $79.26\ sec.$ contre $128.50\ sec.$ sous la tâche ***SS***. Des résultats similaires ont été obtenus pour les tâches complexes ***CS*** et ***CA*** : le délai moyen de réalisation de la tâche ***CA*** était significativement plus court ($139.46\ sec.$) que celui de la tâche ***CS*** ($165.99\ sec.$). Il est évident que le niveau de complexité des tâches affecte le temps nécessaire d'agencement. En effet, tous les sujets ont pris moins de temps pour achever les tâches simples (***SS*** et ***SA***) que les tâches complexes (***CS*** et ***CA***).

Pour les tâches assistées (***SA*** et ***CA***), l'EV est correctement agencé puisque la configuration spatiale proposée par le solveur ne peut évidemment pas contenir d'erreurs (pas d'erreurs de placement et de distance). Pour cette raison, il n'y a pas

Facteur de variabilité	Temps d'agencement	Erreur d'interaction 3D
CT	$p < 0.05$ $F_{1,21} = 18.60$	$p < 0.05$ $F_{1,21} = 101.09$
A	$p < 0.05$ $F_{1,21} = 50.98$	$p < 0.05$ $F_{1,21} = 4.00$
CTxA	$p = 0.4198$ $F_{1,21} = 0.66$	$p = 0.0626$ $F_{1,21} = 3.56$

TABLE 8.2 – *Effet d'interaction et effet principal de CT et A sur le temps d'agencement et l'erreur d'interaction 3D.*

de sens à comparer les agencements manuels (sans solveur) et semi-automatiques (avec solveur) en se basant sur ces deux types d'erreurs. Cependant, il est intéressant d'examiner le nombre d'occurrences de ces erreurs. Le Tableau 8.4 indique le nombre total d'erreurs survenues dans chaque type de tâche.

Complexité	Assistance	N	Moyenne	Écart-type
Simple	*Sans solveur*	22	128.50	31.51
	Avec solveur	22	79.26	25.46
Complexe	*Sans solveur*	22	165.99	42.62
	Avec solveur	22	139.46	39.85

(a)

Complexité	Assistance	N	Moyenne	Écart-type
Simple	*Sans solveur*	22	102.27	24.23
	Avec solveur	22	51.77	41.14
Complexe	*Sans solveur*	22	126.62	34.02
	Avec solveur	22	56.95	32.36

(b)

TABLE 8.3 – *Temps (sec.) d'agencement (a) et nombre d'erreurs (b).*

Contrairement aux erreurs de placement et de distance, l'erreur d'interaction 3D (erreur de sélection dans notre cas) n'est pas nulle pour les agencements semi-automatiques. Ainsi, une comparaison entre les agencements manuels et semi-automatiques peut être basée sur ce type d'erreur. Le Tableau 8.3 montrent que les sujets ont commis beaucoup moins d'erreurs d'interaction 3D en moyenne au cours de ***SA*** (51.77 erreurs) qu'au cours de ***SS*** (102.27 erreurs). Des résultats semblables ont été ob-

servés pour les tâches complexes. Les sujets ont commis en moyenne 56.95 erreurs d'interaction 3D au cours de ***CA*** contre 126.62 erreurs au cours de ***CS***. Ces résultats montrent que malgré la complexité de la tâche, les participants faisaient moins d'erreurs d'interaction 3D lorsque le solveur est utilisé. Ceci peut être expliqué par le fait que dans les agencements manuels, les sujets devaient placer avec précision chaque objet dans l'EV et pouvaient donc être amené à sélectionner le même objet à plusieurs reprises. Dans les agencements semi-automatiques, les sujets sélectionnaient les objets et appliquaient les contraintes sans se soucier de la précision des placements.

	SS	**SA**	**CS**	**CA**
Erreur de placement	116	0	129	0
Erreur de distance	55	0	69	0

TABLE 8.4 – *Le nombre total des erreurs de placement et de distance.*

Bien que les sujets aient choisi différentes stratégies pour finaliser les agencements, les profils d'erreurs d'interaction 3D (des séquences de sélections réussies/échouées au cours du temps) sont assez semblables d'un sujet à l'autre. Par exemple, la Figure 8.4 illustre l'interaction 3D d'un sujet avec les objets au cours des quatre tâches. Il est évident de remarquer que le sujet faisait plus d'erreurs de sélection au cours des tâches manuelles. Celui-ci a commis 97 erreurs de sélection dans ***SS*** et 26 erreurs dans ***SA***. Pour les tâches complexes qui exigent plus de sélection d'objets, le sujet a commis 106 erreurs dans ***CS*** contre 31 erreurs dans ***CA***. Ces résultats montrent que l'utilisation du solveur a permis de réduire le nombre de sélection des objets 3D et par conséquence le nombre d'erreurs de sélection. L'étude des erreurs d'interaction 3D peut également fournir des informations sur l'effort requis pendant les tâches. La Figure 8.5 montre que le nombre moyen de sélections de chaque objet dépend de l'activation ou non de l'assistance.

8.5.2 Aspects subjectifs

Comme dit précédemment, les sujets devaient répondre à un questionnaire NASA-TLX pour évaluer leur perception du niveau de la charge de travail. Les sujets évaluaient chaque indice en utilisant une échelle de Likert à sept points. La Figure 8.6 illustre les scores moyens pondérés de chaque indice pour chacune des tâches. Les résultats révèlent, que les sujets ont trouvé que ***SS*** a provoqué un niveau plus élevé de charge mentale, de charge physique, de charge temporelle, de frustration, et un faible sentiment de performance que dans ***SA***. Des résultats semblables ont été observés pour ***CA*** et ***CS***. Pour confirmer ces résultats, les scores moyens des charges de travail pondérés (Weighted Workload en anglais (WWL)) des sujets étaient de $4,84$ pour ***SS*** contre $4,15$ pour ***SA***, et $4,93$ pour ***CS*** contre $4,21$ pour ***CA***.

Les sujets répondaient également à deux questions supplémentaires sur leurs satisfactions et sur l'assistance fournie lorsque le solveur est utilisé (Fig 8.7). Pour le niveau de satisfaction approuvé, les sujets évaluaient les configurations proposées par le solveur. Pour ***SA*** (agencement simple avec solveur), 18 sujets ont in-

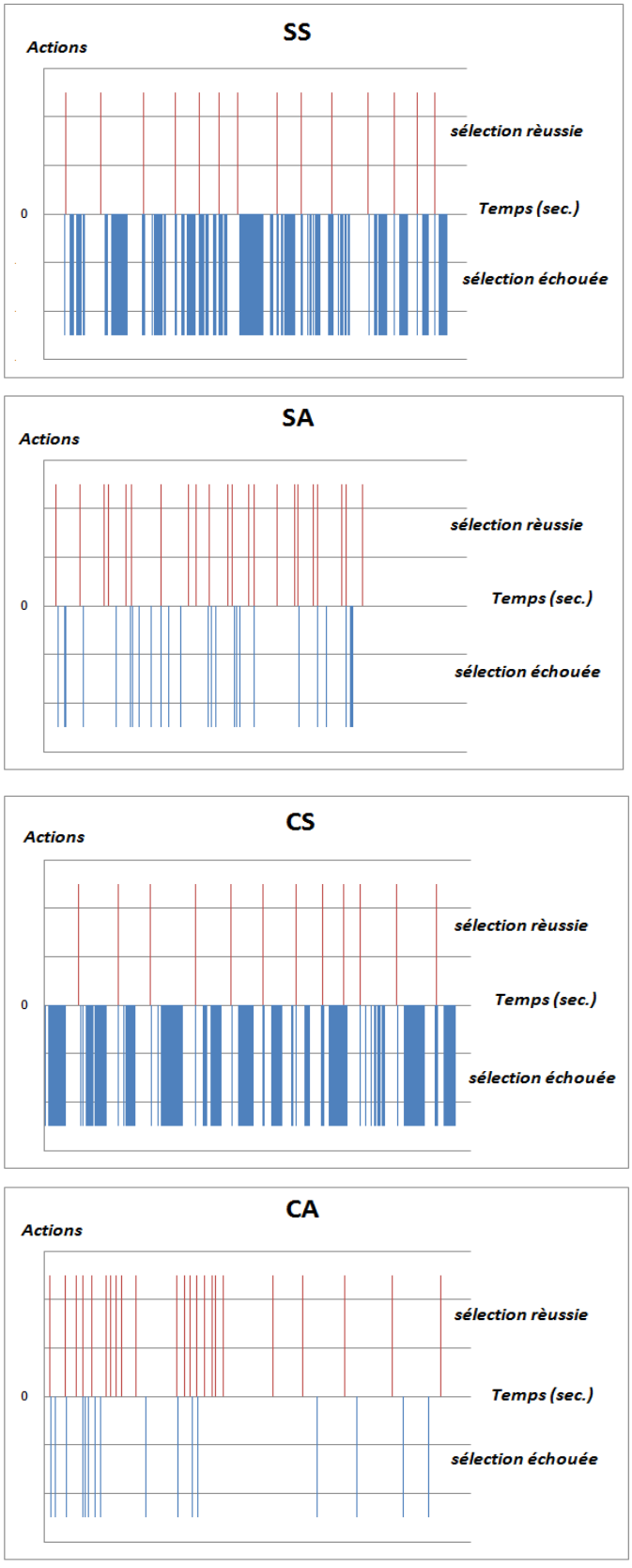


FIGURE 8.4 – *Exemple de séquences de sélections rèussies/échouées observées dans chaque condition.*

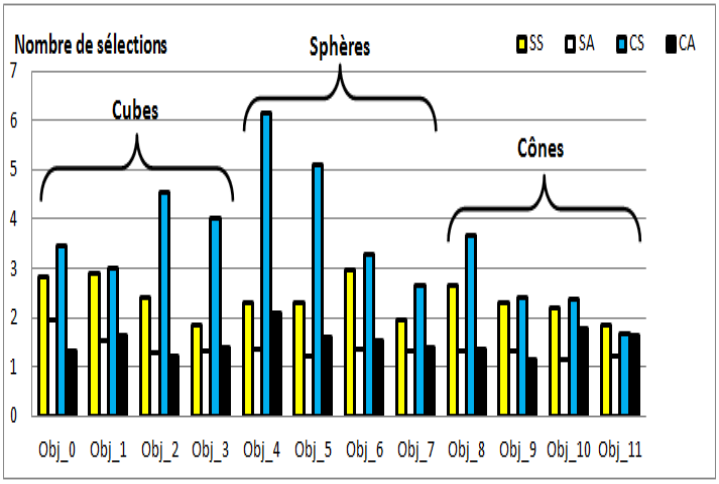


FIGURE 8.5 – *Le nombre moyen de selection pour chaque objet de l'EV.*

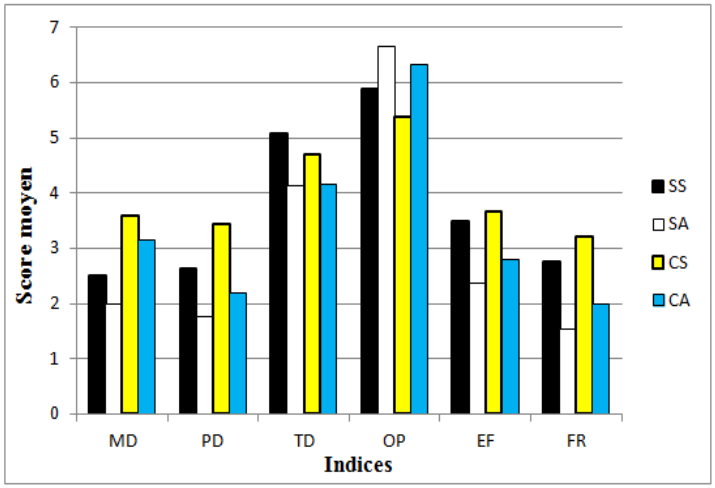
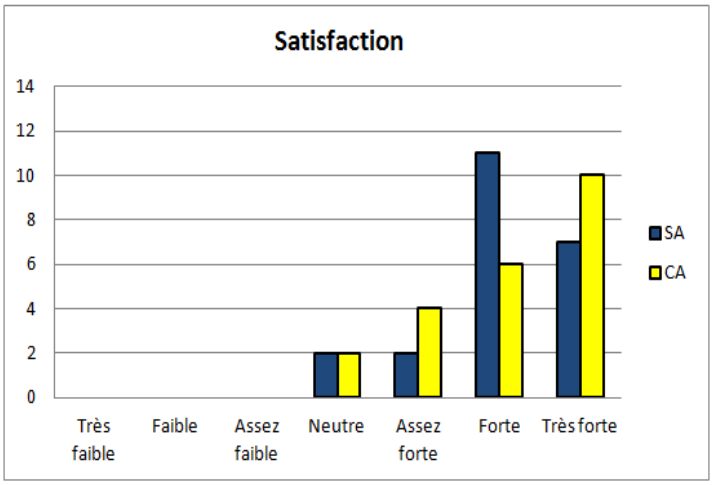


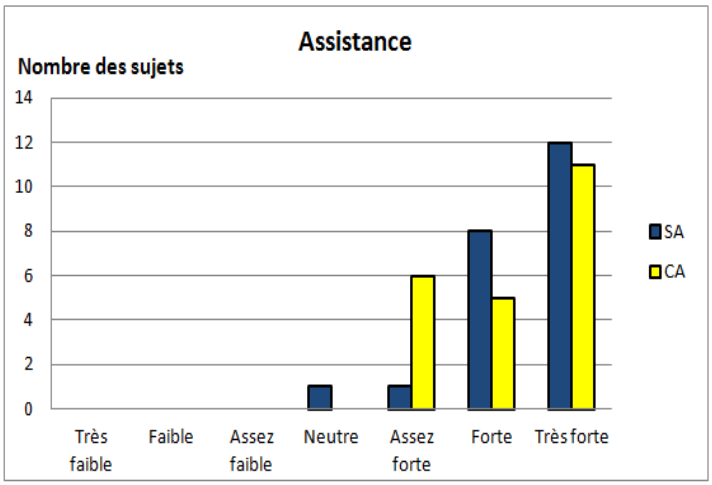
FIGURE 8.6 – *Les scores moyens des indices de la charge de travail.*

diqué qu'ils étaient satisfaits de l'agencement proposé. Deux sujets ont donné une réponse neutre et ont rapporté que la configuration spatiale proposée n'est pas optimisée même si elle respecte toutes les contraintes. Pour ***CA*** (agencement complexe sans solveur), 16 sujets ont été satisfaits ou très satisfaits de la configuration proposée. Seuls deux sujets ont rapporté que le solveur n'a pas proposé la meilleure solution. Il est important de noter que dans la mesure du possible, le solveur peut proposer plusieurs solutions mais pour l'étude actuelle il propose que la première solution trouvée qui ne correspond pas forcément aux attentes des sujets.

En ce qui concerne l'assistance, les sujets devaient évaluer le niveau de l'aide fournie au cours des tâches semi-automatiques. Pour ***SA***, 20 sujets ont constaté que l'agencement est devenu plus facile en utilisant le solveur. Un seul sujet n'a pas senti de différence entre les tâches manuelles et les tâches semi-automatiques. Quant à la tâche ***CA***, où la difficulté est plus élevée, tous les sujets ont trouvé que le solveur a rendu l'agencement nettement plus facile.



(a)



(b)

FIGURE 8.7 – *Données subjectives concernant la satisfaction des sujets (a) et l'assistance fournie par le solveur (b).*

En résumé, les résultats montrent que :

– Les sujets ont pris moins de temps pour placer l'ensemble des objets lorsque le solveur est utilisé. La durée moyenne de l'agencement simple assisté (***SA***) est d'environ $79.25\ sec.$ contre environ $128.50\ sec.$ pour l'agencement simple non assisté (***SS***). De même pour les agencements complexes, les sujets ont passé en moyenne $139.47\ sec.$ pour finaliser l'agencements complexe assisté (***CA***) contre environ $165.99\ sec.$ pour achever l'agencement complexe non assisté (***CS***). Ainsi, la première hypothèse *H1* a été confirmée. La validation de cette hypothèse montre qu'un agencement 3D assisté par un solveur de contraintes, est beaucoup plus rapide qu'un agencement non assisté. Il convient de noter que le délai de réalisation des tâches inclut le temps requis par le solveur pour chercher une solution possible. Ceci dépend fortement des techniques de propagation de contraintes et des heuristiques de recherche utilisées.

– Contrairement aux tâches assistées, où toutes les contraintes ont été respectées, les sujets ont commis beaucoup d'erreurs de placement. En effet, environ 27.27% des objets manipulés dans ***SS*** et 33.91% dans ***CS*** ont été placés en partie en dehors de l'EV. En plus les sujets ont souvent échoué à respecter la contrainte de *Distance minimale*, 21.7% des objets dans ***SS*** et $16,2\%$ dans ***CS*** ont été placés trop près l'un de l'autre (***<d_min***). Ce nombre relativement élevé d'erreurs est principalement causé par la difficulté de manipuler et déplacer des entités en 3D, et pourrait être réduit par l'introduction des indices de profondeur supplémentaires. L'utilisation du solveur a complètement évité ce type d'erreurs. En outre, nous avons observé que le nombre moyen d'erreurs d'interaction 3D lors de la sélection et la manipulation des objets au sein de l'EV était beaucoup plus élevé dans les tâches manuelles (102.27 dans ***SS*** et 126.62 dans ***CS***) que dans les tâches semi-automatiques (51.77 dans ***SA*** et 56.95 dans ***CA***). Ainsi, la seconde hypothèse *H2* concernant l'effet du solveur sur la précision des placements a été également validée. Sa validation montre que l'utilisation du solveur améliore la précision et réduit indirectement la difficulté globale (réduction du nombre de sélections) des agencements 3D.

– La plupart des sujets ont rapporté (via le questionnaire de NASA-TLX) que l'interaction avec le solveur via la technique d'interaction 3D proposé (Nunchuk^{TM}) était simple. Cependant, certains sujets ont eu quelques difficultés à identifier et appliquer les contraintes. En outre, certains participants ont indiqué que la sélection des objets ainsi que les contraintes, été une opération légèrement fastidieuse. Cette difficulté est due à l'utilisation du solveur (sélection des objets et des contraintes), mais pourrait être réduite par d'autres techniques d'interaction 3D intuitives basées sur la multimodalité (par exemple l'introduction d'une commande vocale pour la sélection des contraintes et l'introduction des objets dans la scène). En conséquence, la dernière hypothèse H3 a été dans une certaine confirmée. La satisfaction relative de cette hypothèse montre que la commande du solveur n'est pas encore optimisée et nécessite quelques améliorations.

Au cours des essais, nous avons observé que les sujets avaient des stratégies

différentes lors de l'interaction 3D avec l'EV. Par exemple, certains d'entre eux appliquaient la contrainte *"Sur le sol"* d'une façon séquentielle (objet par objet). D'autres sujets commençaient par sélectionner un ensemble d'objets, puis appliquer sur cet ensemble la dite contrainte. Il est clair que la deuxième stratégie est plus efficace car elle réduit le nombre total de mouvements d'aller retour entre le menu des contraintes et les objets 3D. En outre, dans la deuxième stratégie, le solveur est appelé une seule fois, réduisant ainsi le temps des calculs.

8.6 Conclusion

L'étude expérimentale présentée dans ce chapitre a été menée pour étudier l'effet de l'integration d'un solveur dans un EV générique à la fois sur des données objectives et subjectives mesurées lors d'une tâche d'agencement 3D. L'étude s'est basée sur une comparaison entre des tâches d'agencements manuelles et semi-automatiques, en considérant deux niveaux de difficulté (simple et complexe). Ainsi, quatre conditions expérimentales ont été définies : ***SS*** (tâche simple et sans solveur), ***SA*** (tâche simple avec solveur), ***CS*** (tâche complexe sans solveur) et ***CA*** (tâche complexe avec solveur). Les résultats indiquent que l'utilisation du solveur a totalement éliminé les erreurs de placement et a considérablement réduit le nombre d'erreurs d'interaction 3D quelle que soit le niveau de complexité des tâches. Les résultats ont également révélé, que les sujets réalisaient les agencements plus rapidement quand le solveur était utilisé. La plupart des sujets ont estimé que le solveur a fourni une assistance utile à l'accomplissement des tâches. Ils étaient généralement satisfaits par la solution proposée. Les données objectives et subjectives ont montré que la technique d'interaction 3D proposée présente quelques lacunes. Même dans les tâches semi-automatiques, les participants ont fait des erreurs de sélection en raison d'une mauvaise perception 3D, et ont senti un degré assez élevé de la charge de travail. Afin de pallier à ce problème, différentes techniques d'interaction 3D seront proposées et évaluées dans le chapitre suivant. De plus, l'utilisation de la stéréoscopie et du *head tracking* pourra faciliter l'interaction 3D avec l'EV et donc d'améliorer les performances des sujets [88].

La réalisation de cette expérimentation a quelque part montré les limites de l'outil OpenGl pour ce genre d'application. En effet, tout ce qui est gestion des lumières, chargements des modèles 3D, sélection et déplacements des objets a nécessité un travail supplémentaire pour un résultat très basique. L'utilisation d'un logiciel qui permet facilement de faire tout ce travail s'est avéré indispensable. C'est pour cela qu'une autre version du système a été développée sous Unity 3D (chapitre 6).

9

Évaluation de techniques d'interaction vocale et de type WIMP

9.1 Introduction

L'intégration de modèles 3D dans un environnement virtuel (EV) permet de simuler et de valider différentes tâches d'agencement. Dans cet objectif, une configuration matérielle adéquate, composée d'interfaces et périphériques d'interaction doit être utilisée. A travers ces interfaces et de périphériques, l'utilisateur doit être capable d'interagir de manière naturelle, rapide et efficace avec les entités virtuelles.

Une technique d'interaction influence considérablement la perception de l'environnement dans lequel une tâche d'agencement est réalisée, et affecte évidemment la performance humaine (temps, précision, etc.). De plus, l'appréciation et la satisfaction des utilisateurs quant au système proposé est fortement dépendante de la simplicité et de l'efficacité des interfaces utilisées.

Dans le contexte des problèmes d'agencement 3D, certains travaux ont évoqué la possibilité d'interagir avec l'EV sans pour autant expliciter les protocoles d'interaction utilisés ([101],[144],[54],[34]). De plus, les résultats subjectifs obtenus suite à l'étude expérimentale présentée dans le chapitre précédent, ont révélé que certains sujets avaient des difficultés à suivre le scénario proposé. En effet, les sujets ont trouvé que la sélection des objets (via la Wiimote^{TM}) était difficile et fastidieuse surtout pour les objets se trouvant en profondeur dans l'EV. La sélection des contraintes via les menus était également assez complexe du fait que les sujets devaient identifier la bonne contrainte à partir d'une liste.

Dans ce cadre, nous sommes confrontés à une problématique liée à l'identification des besoins relatifs à la perception des actions que l'utilisateur effectue dans l'EV. Pour pallier ce problème, nous proposons dans ce chapitre une étude expérimentale visant à comparer la pertinence et l'influence sur la performance de l'utilisateur de trois techniques d'interaction : la première est basée sur une interface de

type WIMP (souris), la deuxième sur une commande vocale, et la troisième sur une association de ces deux techniques. L'utilisabilité et l'efficacité de ces techniques d'interaction sont étudiées à travers l'agencement de cinq postes de travail dans une salle. Les techniques proposées, dont une est basée sur la multimodalité [123], seront utilisées pour commander le solveur (validation des contraintes et lancement de la résolution), et pour sélectionner et déselectionner des objets 3D et/ou des contraintes.

9.2 Objectif

Comme montré par Bowman et al. [28], l'évaluation d'une technique d'interaction vise l'identification de problèmes ou de contraintes fonctionnelles et ergonomiques causés par l'emploi de la technique en question.

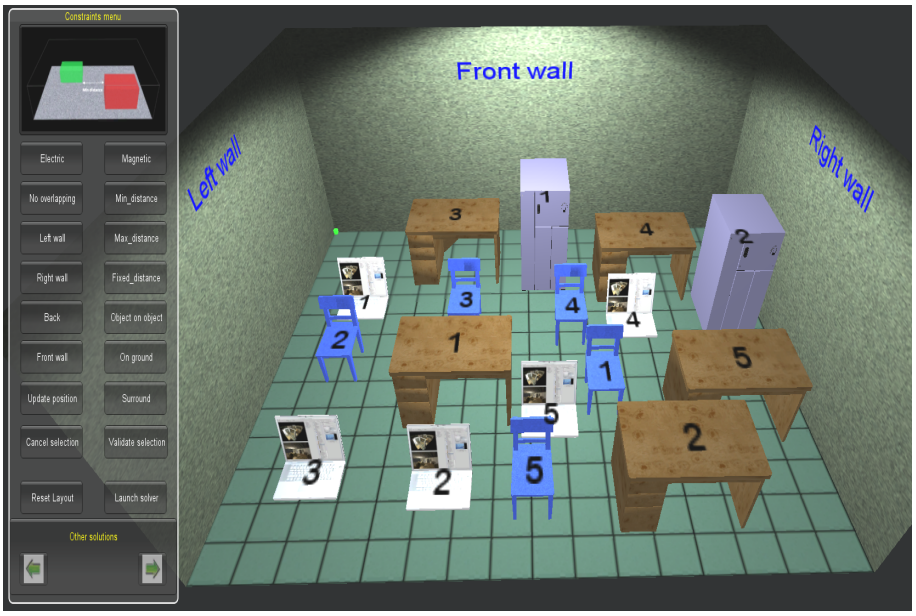
Dans notre contexte, l'objectif de l'étude expérimentale décrite dans ce chapitre est d'identifier la technique d'interaction la plus intuitive et la plus efficace, en s'appuyant sur la comparaison de techniques mono- et multi-modales. L'identification de la technique la plus appropriée dans le contexte d'agencement 3D se base essentiellement sur l'étude de son effet sur la performance humaine (temps de réalisation des tâches), sur l'évaluation de la charge de travail causée par cette technique et sur son influence sur le processus d'apprentissage. Des données subjectives seront également recueillies afin d'évaluer le niveau d'appréciation et le caractère intuitif de chaque technique d'interaction proposée.

9.3 Protocole expérimental

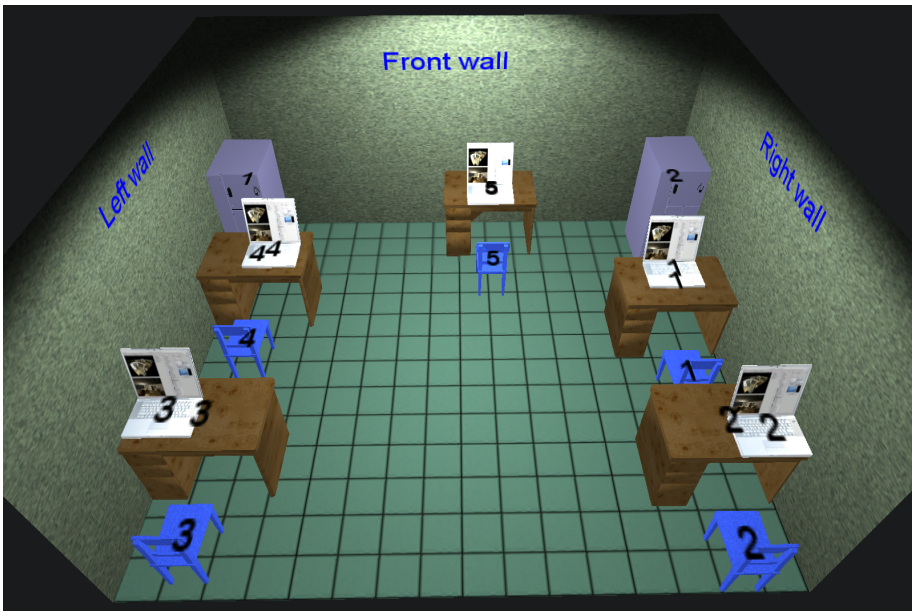
La tâche consistait à commander le solveur pour agencer un ensemble de cinq postes de travail et deux armoires dans une grande salle, selon une configuration précise (contraintes déterminées). Chaque poste de travail est composé d'un bureau, d'une chaise et d'un ordinateur (Fig. 9.1.b). La condition initiale (début de la tâche) est telle que l'ensemble des objets constituant les postes de travail ainsi que les deux armoires, sont répartis aléatoirement dans la scène 3D (Fig. 9.1.a). L'objectif étant de configurer la scène comme le montre la même figure en respectant la numérotation donnée. Par exemple, le bureau$_i$ doit être associé à la chaise$_i$ et à l'ordinateur$_i$. Les objets sont numérotés afin de faciliter leur identification quand la commande vocale est activée.

Les sujets ont été placés devant un mur sur lequel les images ont été projetées. Chaque sujet été amené à réaliser la tâche en utilisant à chaque fois une des techniques d'interaction proposées et détaillées dans la Section 9.4.

Douze sujets volontaires, âgés de 24 à 49 ans, ont accepté de participer à cette expérience. Chacun d'entre eux devait effectuer la tâche 3 fois successivement avec un temps de repos de 30 sec. entre chaque essai. Afin qu'ils puissent s'approprier le système, les sujets ont bénéficié d'une démonstration et ont réalisé une session d'essai pour chaque sous-tâche avant de commencer les passations. L'ordre de passage a été contrebalancé pour éviter un biais dû au transfert d'apprentissage. L'expérimentation a été réalisée avec un ordinateur doté d'un processeur *Intel*$^{\text{TM}}$ $i7-2630$



(a)



(b)

FIGURE 9.1 – *Configuration initiale de la scène (a) et de l'agencement à réaliser (b).*

de fréquence 2.00 Ghz. La fréquence de rafraîchissement des images était de 120Hz.

Le temps de réalisation des sous-tâches a été enregistré à la fin de chaque essai. La charge de travail causée par chacune d'entre elles, a été également évaluée en utilisant un questionnaire basé sur la NASA-TLX (Task Load Index) [75]. Une partie des réponses a permis de collecter les évaluations des sujets quant à la technique d'interaction utilisée et leurs préférences par rapport à l'ensemble des techniques testées.

9.4 Description des tâches

Chaque sujet devait effectuer la tâche décrite dans la section précédente en communiquant avec l'EV via trois techniques d'interaction différentes. Les sujets étaient appelés à reproduire l'agencement final (Fig. 9.1.b) en satisfaisant les contraintes suivantes :

Pour i$\in [1, 5]$:

- Mettre l'ordinateur$_i$ sur le bureau$_i$$\Longrightarrow$ *contrainte Objet_sur_objet* ;
- Placer la chaise$_i$ derrière le bureau$_i$, à une distance fixé ***d_fixe***$\Longrightarrow$ *contrainte Derrière* + *contrainte Distance_fixe* ;
- Placer l'armoire$_1$ dans le coin entre les murs avant et gauche $\Longrightarrow$ *contrainte Devant* + *contrainte Gauche* ;
- Placer l'armoire$_2$ dans le coin entre les murs avant et droit $\Longrightarrow$ *contrainte Devant* + *contrainte Droite* ;
- Placer le bureau$_1$ et le bureau$_2$ contre le mur droit$\Longrightarrow$ *contrainte Droite* ;
- Placer le bureau$_3$ et le bureau$_4$ contre le mur gauche$\Longrightarrow$ *contrainte Gauche* ;
- Placer le bureau$_5$ contre le mur avant$\Longrightarrow$ *contrainte Devant* ;
- Séparer tout les bureaux par une distance minimale ***d_min***$\Longrightarrow$ *contrainte Distance_minimale*.

Afin d'évaluer l'influence des techniques d'interaction proposées sur le processus d'apprentissage, chacune a été testée trois fois de suite, ainsi chaque sujet devait réaliser neuf sous-tâches.

9.4.1 Agencement via une interface de type WIMP

En utilisant uniquement la souris et en changeant la position de la caméra virtuelle, les sujets devaient sélectionner les objets dans la scène puis choisir les contraintes à partir du menu affiché sur la gauche de l'écran (Fig. 9.1.a). Ce processus a été répété pour chaque association objet(s)-contrainte(s). Le lancement de la résolution se fait également via la souris en appuyant sur le bouton adéquat dans le menu des contraintes. Pour ce type de tâche, le sujet manipule la souris tout en étant assis sur une chaise devant l'écran (Fig. 9.2).

9.4.2 Agencement avec commande vocale

Dans ce cas, une deuxième technique d'interaction mono-modale a été proposée pour la sélection des objets/contraintes et le lancement de la résolution (Fig. 9.3).

FIGURE 9.2 – *Un sujet sélectionne les objets et les contraintes à l'aide de la souris.*

En effet, les sujets devaient se baser sur la numérotation donnée pour les objets afin de les sélectionner. La sélection des contraintes se fait simplement en prononçant le non de la contrainte souhaitée.

FIGURE 9.3 – *Un sujet sélectionne vocalement les objets et les contraintes.*

Afin de faciliter la sélection des objets, les sujets pouvaient désigner un objet en prononçant simplement la première lettre de son nom suivi de son numéro. Par exemple C1 pour la chaise$_1$, B2 pour le bureau$_2$, etc. Cette méthode rend plus efficace la reconnaissance vocale, puisqu'il est plus rapide de détecter deux lettres que de détecter un mot ou une phrase. Le Tableau 9.1 énumère les mots utilisés pour interagir avec l'EV. Il est à noter que le développement du programme permettant la reconnaissance vocale est basé sur *Microsoft Speech API* [8].

9.4.3 Agencement avec commande multimodale

La multimodalité désigne l'association de plusieurs modalités pour appréhender un phénomène ou pour interagir avec un objet. Plusieurs travaux ont souligné l'apport de la multimodalité lors de l'interaction avec les EVs [113][85][112].

Mots	Action associée
***"B** i"*	Sélectionner/déselectionner le bureau numéro i (i$\in [1,5]$)
***"C** i"*	Sélectionner/déselectionner la chaise numéro i (i$\in [1,5]$)
***"O** i"*	Sélectionner/déselectionner l'ordinateur numéro i (i$\in [1,5]$)
***"A** i"*	Sélectionner/déselectionner l'armoire numéro i (i$\in [1,2]$)
"Mur devant"	Appliquer la contrainte *Devant* entre l'objet sélectionné et le mur avant
"Mur Droit"	Appliquer la contrainte *Droite* entre l'objet sélectionné et le mur droit
"Mur Gauche"	Appliquer la contrainte *Gauche* entre l'objet sélectionné et le mur gauche
"Derrière"	Appliquer la contrainte *Derrière* entre les deux objets sélectionnés
"Dessus"	Appliquer la contrainte *Objet_sur_objet* entre les deux objets sélectionnés
"Distance fixe"	Appliquer la contrainte *Distance_fixe* entre les objets sélectionnés
"Distance min"	Appliquer la contrainte *Distance_min* entre les objets sélectionnés
"Valider"	Valider les contraintes sélectionnées
"Agencer"	Lancer la résolution

TABLE 9.1 – *Liste des mots utilisés pour la commande vocale relative à la sélection des objets et des contraintes.*

Il est clair que l'interaction via la souris peut devenir fastidieuse lorsque l'agencement implique un grand nombre d'objets et de contraintes, vu l'augmentation du nombre des allers-retours entre le menu des contraintes et les objets. En plus, l'efficacité de la reconnaissance vocale se dégrade en fonction du nombre de mots à détecter, et est très dépendante de l'environnement extérieur (bruit). Pour ces deux raisons, nous avons décidé de combiner les deux techniques (commandes WIMP et vocales) pour interagir avec l'EV. En effet, les sujets sélectionnaient les objets par la voix et choisissaient les contraintes via la souris. La sélection des objets se base sur les mots donnés dans le Tableau 9.1.

Il est à noter que nous avons décidé d'utiliser respectivement la commande vocale et la souris pour la sélection des objets et des contraintes en se basant sur l'hypothèse qu'il est était plus facile de mémoriser le nom d'un objet plutôt que le nom d'une contrainte. Durant l'étude actuelle, nous nous limité à tester cette hypothèse mais nous prévoyons dans le futur de tester l'hypothèse inverse dans laquelle les objets seront sélectionnés par la souris et les contraintes par la voix. Un tel mode de sélection facilitera surtout la sélection des contraintes physiques.

9.4.4 Stratégie de lancement de la résolution

Deux stratégies peuvent être adoptées pour lancer la résolution et réaliser l'agencement. La première consiste à lancer la résolution une fois que toutes les contraintes sont postées, le système proposera alors l'agencement final. Quant à la deuxième stratégie, elle consiste à lancer des résolutions partielles, au fur et à mesure de l'avancement de la tâche, c'est-à-dire après la spécification de certaines contraintes. L'agencement final est donc proposé suite à plusieurs appels au solveur.

La deuxième stratégie est plus appropriée puisqu'elle permet à un sujet de savoir ce qui est fait et ce qui reste à faire. Les multiples appels du solveur n'engendrent pas de longs calculs puisque le nombre d'objets est restreint (17 au total).

9.5 Données recueillies

Afin d'étudier l'effet des techniques d'interaction proposées sur la performance des utilisateurs, nous avons mesuré et comparé le temps de réalisation de la tâche dans chaque condition expérimentale. Le processus d'apprentissage a également été étudié. De plus, des données subjectives ont été recueillies pour évaluer la charge de travail et le niveau d'appréciation des techniques d'interaction testées.

1. **Données objectives**
   (a) *Temps d'agencement*
   Le temps d'agencement est le temps mis par l'utilisateur pour agencer la scène selon la configuration donnée à la Figure 9.1.

2. **Données subjectives**
   (a) *Charge de travail*
   L'évaluation de la charge de travail causée par la tâche s'obtient grâce à six indices : charge mentale (MD), charge physique (PD), charge temporelle (TD), performance personnelle (OP), effort (EF), frustration (FR).

   (b) *Appréciation de la technique d'interaction*
   Via une échelle de Likert à sept points (1 pour "très intuitive" à 7 pour "très complexe"), les sujets devaient exprimer leur appréciation concernant la technique testée.

   (c) *Préférence*
   A la fin de l'expérimentation, une question générale a été posée à chaque sujet afin d'identifier la technique d'interaction préférée. En effet, les sujets devaient choisir, selon leur ressenti personnel, la meilleure technique et justifier leur choix.

9.6 Résultats et discussion

9.6.1 Résultats objectifs

Une analyse de variance (ANOVA) intra-groupes (*within subjects*) à un facteur de variabilité, a été réalisée pour étudier l'effet des techniques d'interaction sur la performance des participants. La seule variable indépendante est le type d'interaction (***I***), tandis que le temps d'agencement (temps de réalisation de la tâche) constitue la seule variable dépendante. Le facteur de variabilité ***I*** est défini par trois niveaux : commande via la souris (***S***), commande vocale (***V***) et commande multimodale (souris-vocale : ***S&V***).

1. *Temps de réalisation*
   Les résultats, obtenus et illustrés à la Figure 9.4, indiquent un effet significatif de ***I*** sur le temps de réalisation de la tâche ($F_{2,11} = 21.88, p < 0.05$). En effet, en utilisant uniquement la souris, le temps moyen de réalisation de la tâche

était de 254.96 sec. contre 201.30 sec. quand l'interaction se fait exclusivement par la voix (Tab. 10.2). Un temps moyen de 207.03 sec. a été observé quand les deux techniques étaient utilisées de manière complémentaire. Ainsi nous constatons que l'utilisation de la commande vocale a permis d'augmenter la performance des sujets d'environ 21.1% pour une utilisation exclusive, et d'environ 18.8% pour une utilisation partielle.

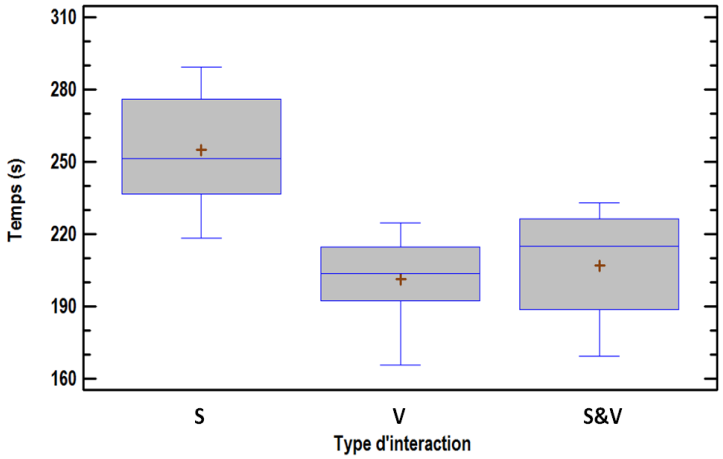


FIGURE 9.4 – *Boîte à moustaches (Box plot) concernant le temps de réalisation de la tâche pour chaque type d'interaction.*

	Interaction (I)	N	Moyenne	Écart-type
Temps (sec.)	***S***	12	254.96	24.17
	V	12	201.30	18.30
	S&V	12	207.03	22.57

TABLE 9.2 – *Moyennes et écart-types des temps de réalisation de la tâche pour chaque type d'interaction.*

La différence de performance entre l'interaction ***S*** et les deux autres, peut être expliquée par le fait que le processus de sélection des objets via la souris était relativement long et fastidieux. En effet, les sujets devaient : (1) repérer l'objet à sélectionner dans la scène 3D, (2) parfois changer le point de vue de la caméra pour la sélection, et (3) déplacer manuellement le curseur de la souris vers l'objet. De plus, sélectionner les contraintes par la souris (à partir du menu de contraintes) peut engendrer une perte de temps puisque les sujets devaient identifier la bonne contrainte dans le menu. L'utilisation de la commande vocale (***S*** et ***S&V***) réduit considérablement le temps de sélection des objets puisque les sujets devaient seulement prononcer le nom des objets sans changer le point de vue de la caméra et sans faire de mouvements de la main.

Bien que l'utilisation exclusive de la commande vocale semble la technique la plus facile et la plus intuitive, l'ANOVA n'a indiqué aucune différence significative entre les performances obtenues avec les techniques ***V*** et ***S&V*** ($F_{1,11} = 0.658$, $p = 0.49$). En effet, la sélection des contraintes, différente

selon les deux techniques, nécessite dans les deux cas presque le même temps de réflexion. Avec la commande vocale (***V***), les sujets devaient se souvenir du nom de la contrainte pour pouvoir l'appliquer, alors qu'ils devaient l'identifier dans le menu de contraintes avec la commande multimodale (***S&V***).

2. *Apprentissage*
   La notion d'apprentissage peut être définie comme l'évolution de la performance des sujets au cours des essais répétitifs d'une tâche donnée. L'étude du processus d'apprentissage d'une technique d'interaction s'avère intéressante puisqu'elle reflète la capacité des utilisateurs à se l'approprier. Une absence d'apprentissage ou une dégradation de la performance peuvent également renseigner sur les difficultés quant à l'utilisation de la technique d'interaction testée.

   Afin d'étudier plus en détail l'effet des techniques d'interaction sur le temps de réalisation, nous avons analysé l'évolution de celui-ci au cours des essais. Comme le montre la Figure 9.5, une diminution du temps de réalisation de la tâche est observée pour les trois conditions expérimentales (***S***, ***V*** et ***S&V***). En effet, les temps mesurés étaient respectivement de 292.31 sec., 248.16 sec. et 250.93 sec. lors du premier essai, et respectivement de 216.05 sec., 156.06 sec. et 168.36 sec. lors du troisième. Un gain de performance plus important a été observé pour l'interactions ***S***. En effet, le temps de réalisation a été amélioré de 73%, 63% et 67% pour ***S***, ***V*** et ***S&V*** respectivement.

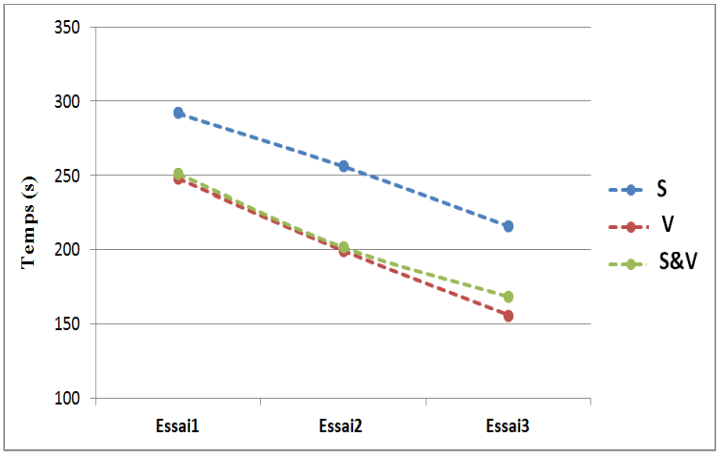


FIGURE 9.5 – *Processus d'apprentissage relatif au temps de réalisation de la tâche en fonction du type d'interaction*. S (commande via la souris), V (commande vocale) et S&V (commande multimodale).

9.6.2 Aspects subjectifs

Au cours de cette sous-section, nous traitons, via le questionnaire, les aspects subjectifs relatifs à la charge de travail. Nous faisons également état des informations notées pendant l'expérience (difficultés rencontrées, préférence des sujets, etc.).

1. *Charge de travail*
   La Figure 9.6 illustre le score moyen obtenu pour chaque indice impliqué dans l'évaluation de la charge de travail. Les résultats révèlent que la technique utilisant la souris seule a provoqué un niveau élevé de charge mentale, physique, temporelle. Elle a également engendré plus frustration par rapport aux autres techniques proposées. L'interaction exclusivement vocale, a causé également un niveau assez élevé de charge mentale (par rapport à la technique combinée) dû au fait que les sujets devaient se souvenir des noms des objets et des contraintes. Nous estimons que pour un nombre plus grand d'objets et de contraintes, cette technique peut rapidement atteindre ses limites. Quant à la technique combinée, elle a exigé plus d'effort physique et a provoqué une charge physique plus importante que l'interaction purement vocale. Ceci est dû à l'utilisation, même partielle, de la souris.

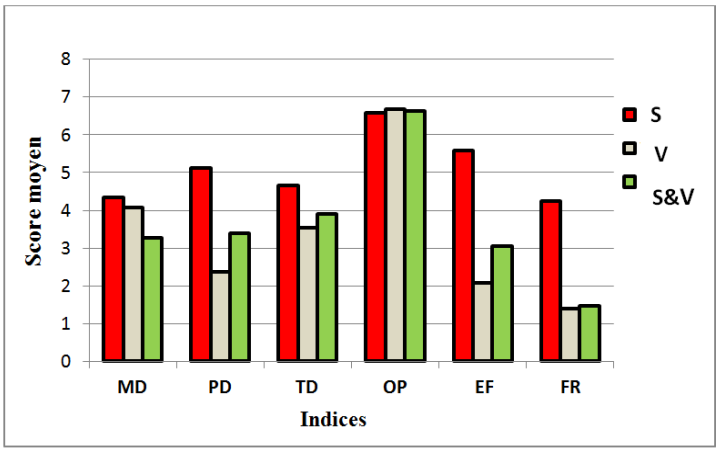


FIGURE 9.6 – *Scores moyens des indices de la charge de travail pour chaque type d'interaction.*

2. *Évaluation des techniques d'interaction*
   Les sujets devaient évaluer la technique d'interaction testée après la fin de chaque sous-tâche. L'évaluation est basée sur une échelle de Likert à sept points (1 pour "très facile" à 7 pour "très complexe"). Comme indiqué dans la Figure 9.7, plus de 66% des sujets ont trouvé que la technique ***S*** était complexe ou très complexe. Un seul sujet a rapporté qu'elle était facile. En revanche, prés de 92% des sujets ont trouvé la technique ***V*** était facile et même très facile. Un seul sujet a senti une certaine difficulté avec l'interaction vocale. Des avis semblables sont exprimés concernant la technique combinée. En effet, seulement deux sujets n'ont pas senti de facilité lors de l'interaction avec l'EV.

3. *Préférences des sujets*
   Comme dit précédemment, une question générale a été posée à la fin de l'expérimentation afin d'avoir un retour sur les préférences des sujets quant aux techniques d'interaction proposées. Les informations rapportées ne font que confirmer les résultats décrits dans la partie *Évaluation des techniques d'interaction*. En effet, les préférences des sujets étaient partagées entre les techniques d'interaction ***V*** et ***S&V***. Un seul sujet avait une préférence pour l'in-

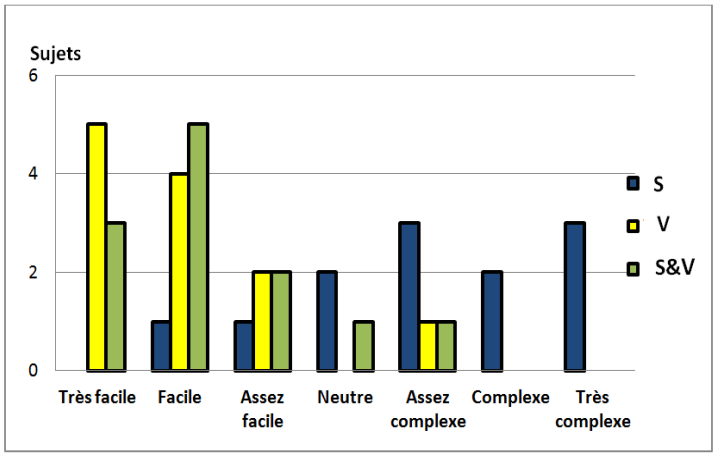


FIGURE 9.7 – *Évaluation subjective de chaque type d'interaction.*

teraction via la souris. La majorité des sujets ont justifié leur choix en soulignant que l'utilisation de la commande vocale pour ce type de tâche, était plus simple et plus naturelle que l'interaction de type WIMP (souris). Ils ont rapporté que contrairement à l'interaction via la souris, ils n'avaient pas besoin de connaître l'interface du système pour pouvoir interagir avec l'EV.

Les résultats décrits ci-dessous laissent penser que la commande vocale a considérablement aidé les sujets à accomplir la tâche. Même une utilisation partielle de cette technique (associée à l'utilisation de la souris) a fait ses preuves en terme de performance et a suscité la satisfaction des sujets. Cependant, nous pensons que l'utilisation exclusive de cette technique dans une scène plus complexe (beaucoup d'objets et de contraintes), n'est pas forcement efficace. En effet, l'utilisateur sera contraint d'apprendre par coeur tous les mots désignant les contraintes. Ce qui nous amène à penser qu'une technique basée sur la multimodalité est plus appropriée.

9.7 Conclusion

L'étude expérimentale présentée dans ce chapitre a été mise en place pour étudier l'influence de trois techniques d'interaction sur la performance, sur la charge de travail et sur la préférence des utilisateurs. L'expérimentation proposée est basée sur une étude comparative entre deux techniques d'interaction monomodales et une technique multimodale. Trois conditions expérimentales ont été définies : ***S*** (Agencement avec la souris), ***V*** (Agencement avec interaction vocale), ***S&V*** (Agencement avec interaction souris-vocale).

Les résultats obtenus indiquent que les sujets ont réalisé la tâche plus rapidement quand la commande vocale était utilisée ***S&V***. La technique ***V*** permettait d'obtenir les temps de réalisation les plus courts. Les résultats ont également révélé que les sujets favorisaient les techniques ***V*** et ***S&V*** plutôt que la technique ***S***. En effet, celle-ci a causé un niveau assez élevé de charge de travail.

La réalisation de cette expérimentation a montré l'intérêt de la multimodalité dans l'interaction avec la scène à agencer. Le choix de l'interaction la plus appropriée dépendra fortement de la taille du problème d'agencement 3D. En effet, une

interaction multimodale (souris-vocale) semble plus efficace quand il s'agit d'agencer un grand nombre d'objets. Après avoir : (1) confirmé l'utilité de l'intégration d'un solveur dans un EV pour résoudre un problème d'agencement (chapitre 8), (2) identifié une technique d'interaction adéquate, nous poursuivons nos études expérimentales dans le chapitre suivant avec pour objectif d'évaluer une des formes d'assistance proposées par le système et décrites dans le chapitre 7.

10

Évaluation de l'assistance visuelle anticipée

10.1 Introduction

Comme dit dans le chapitre 6, le système développé permet la résolution d'un problème d'agencement 3D en proposant automatiquement des configurations d'objets. Il permet également d'annuler ou rejeter un placement manuel d'objets si celui-ci conduit à une violation de contraintes. Ainsi, l'aide apportée par le solveur survient après le placement effectif des objets, ce qui est insuffisant dans la plupart des cas.

Une des problématiques liées à l'utilisation du solveur, concerne les formes d'assistance que peut fournir celui-ci. En effet, il est tout à fait légitime de poser la question : **de quelles manières le solveur peut-il assister l'utilisateur au cours de l'agencement** ?

Pour tenter de résoudre cette problématique, nous avons décidé de proposer une aide visuelle anticipée (basée sur le mécanisme de ***look ahead***) avant même de placer un objet donné. L'utilisateur est alors en mesure de visualiser les "mauvaises" zones de placement pour un objet sélectionné. Les zones impossibles d'un objet obj_i représentent les surfaces 3D où le placement de l'objet obj_i conduit à une violation de contrainte(s). Le solveur identifie ces zones et transmet les informations nécessaires à l'EV afin de les visualiser. Ainsi, ce chapitre sera dédié à la présentation d'une étude expérimentale menée pour étudier l'effet de la visualisation des zones impossibles sur la performance des utilisateurs durant une tâche d'agencement 3D.

10.2 Objectif

L'objectif principal consiste à évaluer l'effet de l'assistance fournie sur des facteurs de performances tel que le temps d'accomplissement d'une tâche d'agencement, les erreurs d'agencement et le comportement de l'utilisateur durant la tâche.

Une comparaison de deux approches différentes (avec/sans assistance) a été réalisée. Contrairement à la tâche assistée, l'affichage des zones impossibles est désactivé dans la tâche sans assistance. Afin d'obtenir des informations spécifiques sur les difficultés ressenties par les sujets durant les tâches, une mesure subjective de la charge de travail a été réalisée à travers un questionnaire basé sur la NASA-TLX (Task Load Index, développée à la NASA) [75]. D'autres données subjectives ont été également visées par ce questionnaire portant sur le ressenti des sujets par rapport à l'assistance fournie et leur satisfaction par rapport l'agencement réalisé.

10.3 Protocole expérimental

La tâche consistait à placer manuellement quatre objets 3D (un canapé, un ordinateur portable, une lampe et une table de salon) en tenant compte des objets déjà placés (un bureau, un canapé, 2 tables de coin, une chaise de bureau et un objet générique de couleur bleu sensible au champ électrique), tout en respectant un ensemble de contraintes prédéfinies.

Douze sujets volontaires, âgés de 22 à 34 ans, ont participé à l'expérimentation. Ils avaient presque le même niveau d'expertise dans les jeux vidéo et les applications de RV. Une souris a été utilisée comme périphérique d'entrée pour sélectionner, désélectionner et déplacer les objets dans l'espace 3D affiché sur un écran de moyenne dimension (Fig 10.1). Les sujets pouvaient changer leur point de vue dans la scène 3D en utilisant les touches de direction du clavier. L'objectif principal de

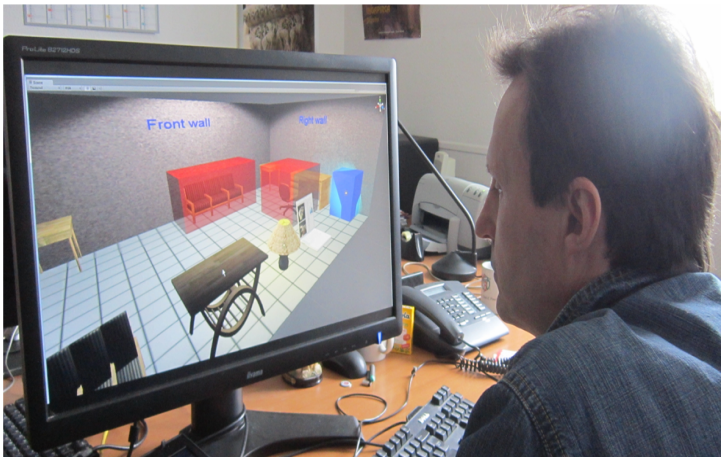


FIGURE 10.1 – *Un sujet agence la scène 3D à l'aide la visualization des zones impossibles.*

l'étude a été expliqué aux sujets, une brève description de deux conditions expérimentales (sans/avec assistance) a été également présentée. Avant le début de la tâche, les sujets étaient informés de l'ensemble des contraintes exigées pour l'agencement, de la valeur de la charge électrique $\boldsymbol{q}$ de l'objet bleu et de la valeur maximale du champ électrique supportée par l'ordinateur portable.

Une session d'essai a été proposée à chaque sujet afin de se familiariser avec l'interface du système, le contrôle de la caméra virtuelle, la manipulation des objets et la sélection des contraintes. Elle consistait à accomplir les deux conditions de

l'expérimentation. L'ordre de passage des conditions a été contrebalancé afin d'éviter un biais dû au transfert d'apprentissage.

Après chaque sous-tâche, les sujets devaient répondre à un questionnaire permettant d'évaluer leur niveau de frustration, l'effort fourni, la performance, la satisfaction, etc. Il est à noter que deux questions supplémentaires ont été ajoutées afin d'évaluer le niveau et l'effet de l'assistance fournie par le solveur durant la tâche.

Chaque sujet été appelé à réaliser la tâche sous deux conditions (sans/avec assistance). Les quatre objets à agencer et ceux déjà placés ont été disposés comme le montre la Figure 10.2. Au cours de la tâche, le sujet devait sélectionner puis déplacer un objet donné en utilisant le bouton gauche de la souris. Afin d'illustrer

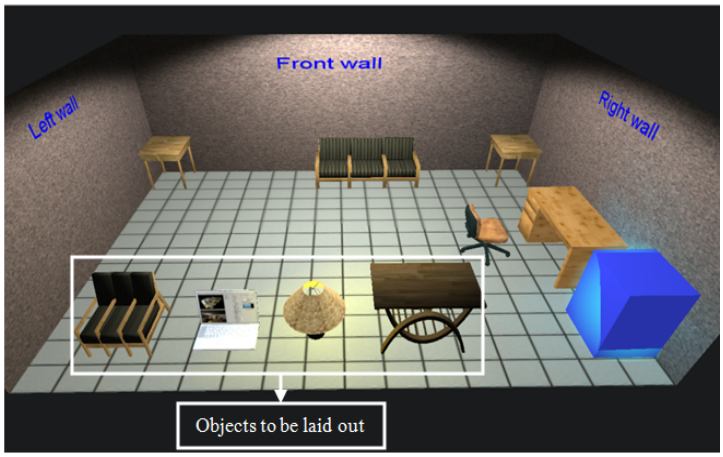


FIGURE 10.2 – *Copie d'écran de la condition initiale de la tâche.*

les contraintes physiques, nous avons supposé que l'objet bleu émettait un champ électrique et que l'ordinateur portable était sensible à un tel champ et ne pouvait supporter qu'un seuil bien déterminé du champ électrique.

L'expérimentation a été réalisée avec un ordinateur ayant un processeur *Intel*^{TM} $i7-2630$ de fréquence $2.00Ghz$. La scène 3D a été affichée sur un écran de grande taille ; la fréquence du rafraîchissement de l'écran est de 120Hz (60Hz par oeil).

10.4 Description des tâches

Comme décrit ci-dessus, chaque sujet devait agencer manuellement la scène 3D sans et avec assistance (visualisation des zones impossibles) tout en respectant les contraintes décrites ci-dessous :

- Placer l'ordinateur portable sur le bureau$\Longrightarrow$ *contrainte Objet_sur_objet*
- Protéger l'ordinateur portable du champ électrique émis par l'objet bleu$\Longrightarrow$ *contrainte Électrique*
- Mettre la lampe sur la table1 du coin (celle du gauche)$\Longrightarrow$ *contrainte Objet_sur_objet*
- Placer le canapé$_2$ contre le mur avant$\Longrightarrow$ *contrainte Devant*
- Séparer le canapé$_2$ et le canapé1 par une distance minimale ***d_min***$\Longrightarrow$ *contrainte Distance_min*

- Séparer le canapé$_2$ et la table1 du coin par une distance minimale ***d_min*** $\Longrightarrow$ *contrainte Distance_min*
- Séparer la table de salon et le canapé1 par une distance fixe ***dist*** $\Longrightarrow$ *contrainte Distance_fixe*
- Séparer la table de salon et la chaise du bureau par une distance minimale ***d_min*** $\Longrightarrow$ *contrainte Distance_min*

10.4.1 Agencement non assisté

En utilisant la souris et en changeant la position de la caméra virtuelle, les sujets devaient placer manuellement les quatre objets tout en respectant les contraintes présentées avant le début de la tâche. Aucune information ou indication n'a été donnée pour aider les sujets durant l'agencement.

10.4.2 Agencement assisté

Via le menu des contraintes, les sujets devaient sélectionner les contraintes définies pour les quatre objets à agencer. Cette opération est primordiale afin d'"informer" le solveur quant aux contraintes requises pour l'agencement. En utilisant le même menu, les sujets pouvaient saisir les valeurs correspondant à la charge électrique ***q*** de l'objet bleu et le champ électrique maximal supporté par l'ordinateur portable. Ces valeurs sont transmises au solveur afin d'identifier les zones dans lesquelles la valeur du champ électrique dépasse la valeur maximale supportée par l'ordinateur portable.

Une fois la spécification des contraintes effectuée, les sujets pouvaient activer/désactiver la visualisation des zones impossibles pour chaque objet de la scène avant de le positionner manuellement. Ainsi, les sujets étaient guidés par le solveur (à travers l'affichage des zones) pour effectuer leurs choix de placement. Afin d'identifier les zones impossibles de chaque objet, la scène 3D a été décomposée en 125 zones ($5 \times 5 \times 5$).

10.5 Données recueillies

Afin d'étudier l'effet de la visualisation des zones impossibles sur les performances des utilisateurs, nous avons recueilli et comparé des données objectives (le temps d'agencement, les erreurs d'agencement et le nombre des actions de l'utilisateur durant la tâche) et des données subjectives (charge de travail, la satisfaction et le besoin d'assistance) dans chaque condition expérimentale.

1. **Données objectives**

   (a) *Le temps d'agencement*
   Le temps d'agencement est la durée qui sépare l'instant où le sujet sélectionne/manipule un objet dans la scène et l'instant où il considère que la tâche est accomplie.

(b) *Les erreurs d'agencement*
En se basant sur les positions finales de chaque objet, une vérification est faite pour la satisfaction de chaque contrainte (Section 10.4). En effet, une erreur d'agencement survient quand une contrainte est violée. Nous avons compté le nombre totale des erreurs dans chaque condition.

(c) *Le nombre d'actions de l'utilisateur*
Pour obtenir des informations sur le taux d'interaction des sujets et l'effort fourni pour la réalisation de la tâche, nous avons décidé de compter chaque sélection/manipulation d'objet effectuée par les sujets. Dans notre contexte une action survient quand un sujet sélectionne et déplace un objet 3D.

2. **Données subjectives**

(a) *Charge de travail*
Après la fin de chaque condition expérimentale, les sujets répondaient à un questionnaire basé sur la NASA-TLX afin d'évaluer le niveau de la charge de travail causé par la tâche. L'évaluation de la charge de travail se calcul à travers six indices : Charge mentale (MD), Charge physique (PD), Charge temporelle (TD), Performance personnelle (OP), Effort (EF), Frustration (FR).

(b) *Satisfaction des sujets et assistance fournie*

– *Satisfaction*
En utilisant une échelle de Likert à sept points (1 pour "très faible" à 7 pour "très élevé"), les sujets devaient évaluer leur satisfaction concernant l'agencement réalisé. Leur satisfaction dépendait fortement de leur ressenti à propos du respect des contraintes.

– *Le besoin d'assistance*
En utilisant la même échelle de Likert, les sujets devaient exprimer s'ils avaient besoin d'une assistance durant les tâches. Chaque sujet a évalué le niveau de ce besoin.

– *Effets de l'assistance*
Deux questions supplémentaires ont été posées après la fin des tâches assistées. Les sujets devaient exprimer leur ressenti par rapport à l'assistance fournie par le solveur. Le but de la première question était de savoir jusqu'à quel niveau la visualisation des zones a facilité ou non l'agencement (*facilité de l'agencement*). Les réponses de la deuxième question ont permis de savoir si la visualisation des zones a aidé les sujets à se souvenir des contraintes au cours de la tâche (*le souvenir des contraintes*). Il est important de noter que ces deux questions n'étaient pas posées pour les tâches non assistées. En effet, une évaluation du niveau de l'assistance n'a pas de sens puisque ces tâches sont totalement manuelles et ne fournissent aucune forme d'aide.

10.6 Résultats et discussion

10.6.1 Résultats objectifs

Une analyse de variance (ANOVA) intra-groupes (*within subjects*) à un facteur de variabilité a été réalisée pour évaluer l'effet de la variable indépendante (assistance) sur le temps d'agencement, les erreurs d'agencement et le nombre des actions des sujets durant les tâches.

Le Tableau 10.1 illustre l'effet de ***A*** (assistance) sur chacun des facteurs mesurés. Comme indiqué dans le tableau, ***A*** affecte d'une manière significative les erreurs d'agencement et le nombre des actions des sujets durant les tâches. Quant au temps d'agencement, les résultats ont montré qu'il n'y pas de différence significative entre les temps d'accomplissement des tâches non assistées et celles assistées.

Facteur de variabilité	**Temps**	**Erreurs**	**Actions des sujets**
A	$p = 0.3259$ $F_{1,11} = 1.31$	$p < 0.05$ $F_{1,11} = 50$	$p < 0.05$ $F_{1,11} = 77.12$

TABLE 10.1 – *Effet de l'assistance (A) sur le temps, les erreurs d'agencement et le nombre des actions des sujets.*

Effet de l'assistance (A) sur le temps, les erreurs d'agencement et le nombre des actions des sujets

1. *Erreur d'agencement*

   L'effet significatif de **A** sur les erreurs d'agencement est expliqué par le fait que durant les tâches non assistées, aucune information ou indication n'est fournie aux sujets concernant les zones non susceptibles de contenir un objet donné. De ce fait, les sujets plaçaient aléatoirement les objets selon leurs propres estimations, ce qui génére souvent des erreurs de placement. Une difficulté particulière a été ressenti par la plupart des sujets quand ils voulaient placer l'ordinateur portable en le protégeant du champ électrique émis par l'objet bleu, bien qu'ils étaient informés des valeurs des paramètres requis (la charge ***q*** et le champ électrique maximum supporté).

   Comme décrit dans la Section 10.4, les contraintes étaient présentées et testées seulement avant le début de la tâche. Les sujets ont donc souvent oublié les contraintes requises pour l'agencement ce qui provoque la violation de ces dernières. Contrairement aux tâches non assistées, les sujets sont guidés au cours des tâches assistées à travers la visualisation des zones impossibles pour chacun des quatre objets à agencer. Cette assistance réduit considérablement la probabilité de violation des contraintes.

   En plus, la visualisation de ces zones a aidé les sujets à se souvenir des contraintes exigées et donc de mieux les satisfaire. Ceci est confirmé par le

nombre moyen d'erreurs d'agencement pour chaque type de tâche. En effet, les sujets ont commis moins d'une erreur en moyenne (0.77 erreurs) au cours des tâches assistées contre 4 erreurs au cours des tâches non assistées (Tab. 10.2).

2. *Nombre d'actions*

   Comme illustré dans le Tableau 10.2, les sujets ont interagi avec la scène 3D moins souvent quand la visualisation des zones impossibles était activée. Ils sélectionnaient et manipulaient les objets 39.44 fois en moyenne durant les tâches non assistées, contre 19.22 fois en moyenne durant les tâches assistées.

   Ces résultats peuvent être expliqués par le fait que les sujets ont beaucoup de doutes concernant les placements d'objets effectués puisque il n'y a pas d'indication sur la validité de leurs choix. Pour cette raison, la plupart d'entre eux changeaient fréquemment la position des objets dans l'espoir de satisfaire le maximum des contraintes. Par conséquence, ils ont produit des efforts supplémentaires pour achever la tâche. Au cours des tâches assistées, leur fréquence d'actions est plus faible parce qu'ils avaient une meilleure vision des impossibilités de placement de chaque objet.

3. *Le temps d'agencement*

   Pour le temps d'agencement, aucune différence significative n'a été observée pour les deux niveaux de **A** (sans/avec visualisation des zones). Cependant, l'étude du temps moyen pour chaque condition expérimentale montre que les sujets ont fini plus rapidement les tâches non assistées que celles assistées.

   En moyenne les sujets ont pris 126.76 $Sec.$ pour finir l'agencement non assisté contre 129.31 $Sec.$ pour l'agencement assisté. Ceci est causé par la nécessité d'informer le solveur des contraintes requises (via le menu des contraintes) afin de calculer les zones impossibles de chaque objet. Cette étape nécessite un temps supplémentaire qui s'ajoute au temps d'agencement réel. Comme mentionné dans le chapitre 7, le solveur peut tester 1000 zones en moins d'une seconde. Pour notre étude expérimentale, le solveur a testé au maximum 125 zones ($5 \times 5 \times 5$: la précision de la décomposition), le temps du calcul pour la vérification des zones est donc largement inférieur à une seconde, ce qui assure une interaction quasi temps réel entre l'utilisateur et le système .

10.6.2 Aspects subjectifs

1. *Charge de travail*

   A travers une échelle de Likert, les sujets devaient évaluer le niveau de la charge de travail ressenti durant les sous-tâches. La Figure 10.3, illustre le score moyen pour chaque indice nécessaire à l'évaluation de la charge de travail. Les résultats révèlent que l'agencement non assisté a causé un niveau élevé de charge mentale, charge physique, charge temporelle, frustration et a nécessité plus d'effort par rapport à l'agencement assisté.

	Assistance(A)	N	Moyenne	Écart-type
Temps(sec.)	*Sans assistance*	12	126.76	33.01
	Avec assistance	12	129.31	32.09
Erreurs	*Sans assistance*	12	4	1.22
	Avec assistance	12	0.77	0.83
Actions des sujets	*Sans assistance*	12	39.44	5.81
	Avec assistance	12	19.22	3.73

TABLE 10.2 – *Moyennes et écart-types dans chaque condition expérimentale.*

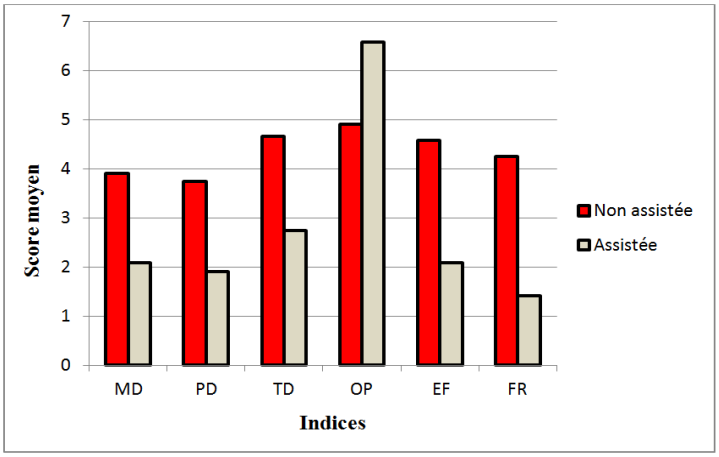


FIGURE 10.3 – *Scores moyens des indices de la charge de travail pour chaque condition.*

2. *Satisfaction*

   Comme présenté dans la Figure 10.4, 11 sujets ont exprimé leur grande satisfaction concernant l'agencement final réalisé durant les tâches assistées. Un seul sujet a donné une réponse neutre à cette question. Pour les tâches non assistées, 4 sujets ont exprimé une satisfaction moyenne, 3 ont donné une réponse neutre et 5 ont affirmé leurs insatisfaction à cause des doutes qu'ils ont eu concernant les placements effectués.

   Ces résultats confirme l'effet de la visualisation des zones sur la satisfaction ressentie par les sujets. En effet, quand l'assistance était activée, les sujets ont eu l'impression de respecter les contraintes en évitant simplement de placer les objets dans les zones impossibles. Les données objectives, en particulier celles des erreurs d'agencement, laissent penser que certains sujets ont exprimé leurs entière satisfaction après les tâches non assistées, parce qu'ils ont cru respecter toute ou la majorité des contraintes. Souvent, ce n'est pas le cas.

3. *Besoin d'assistance*

   Une grande majorité des sujets ont affirmé qu'il était primordiale pour eux

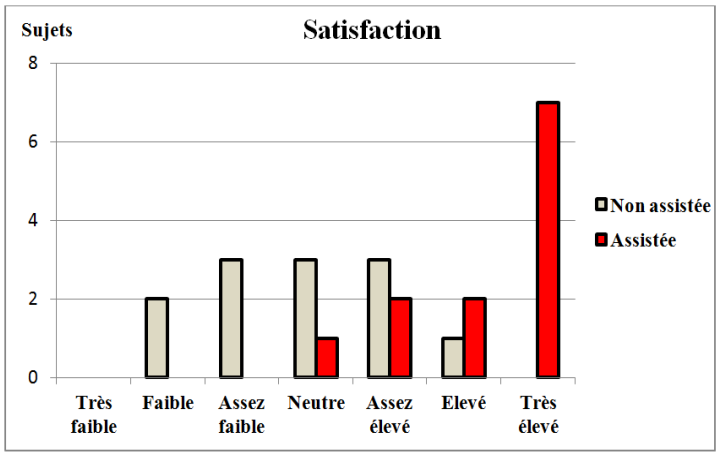


FIGURE 10.4 – *Satisfaction des sujets dans chaque condition.*

d'avoir une assistance afin de perfectionner l'agencement réalisé. Tout les sujets ont ressenti un besoin d'assistance élevé durant les tâches non assistées. Quant aux tâches assistées, 8 sujets ont assuré que la visualisation des zones était largement suffisante pour respecter les contraintes. Seuls 4 sujets ont exprimé un besoin d'avoir un autre type d'aide durant ces tâches (Fig 10.5).

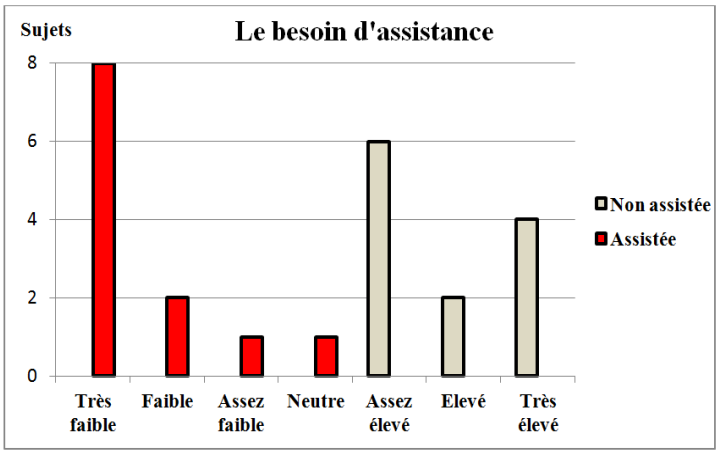


FIGURE 10.5 – *Besoin d'assistance ressenti par les sujets pour chaque condition.*

Ces résultats révèlent que la visualisation des zones impossible était, selon les sujets, une assistance suffisante et efficace et à travers laquelle les sujets sont parvenus à satisfaire le maximum de contraintes.

4. *Facilité d'agencement et le souvenir des contraintes*

   Comme dit précédemment, deux questions supplémentaires ont été posées pour évaluer l'assistance fournie durant les tâches assistées. La Figure 10.6, montre que 11 sujets ont trouvé que la tâche est devenue plus facile en étant assisté par le solveur. Un seul sujet a rapporté qu'il n'a pas senti de différence entre la difficulté rencontrée durant les deux types de tâches (avis neutre). Plus de 58% des sujets ont remarqué une grande facilité d'agencement quand le solveur est utilisé.

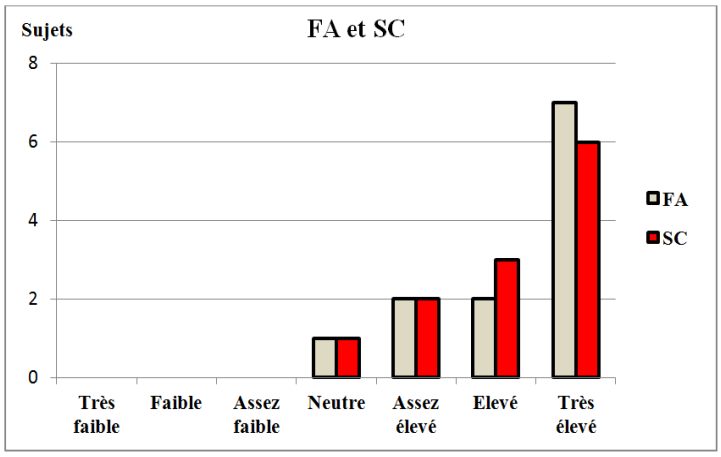


FIGURE 10.6 – *Données subjectives concernant la facilité de l'agencement (FA) et le souvenir des contraintes (SC) ressentis durant les tâches assistées.*

Pour ce qui est souvenir des contraintes, 50% des sujets ont affirmé que l'assistance les a beaucoup aidé pour se souvenir des contraintes durant la tâche et par conséquent satisfaire le maximum de celles-ci. Un seul sujet a donné une opinion neutre.

10.7 Conclusion

Les analyses objectives et subjectives des résultats ont montré que l'assistance apportée par le solveur de contraintes (visualisation des zones impossibles) a considérablement aidé les sujets durant les tâches. Même si les sujets ont passé plus de temps pour achever la tâche quand ils étaient assistés, l'utilisation du solveur a beaucoup minimisé le nombre d'erreurs d'agencement et le niveau de la charge de travail causée au cours de la tâche. L'intérêt de la visualisation des zones impossibles a été particulièrement ressenti lors de la satisfaction de la contrainte électrique (liée à l'ordinateur portable). En effet, les sujets ne pouvaient pas savoir avec précision l'étendu du champ électrique émis par l'objet bleu, ce qui rendait le placement de l'ordinateur très difficile et aléatoire. Sans assistance, une telle situation peut devenir très complexe quand un ensemble d'objets doit être protégé des champs électriques (ou magnétiques) émis par d'autres objets.

Bien que la finalité principale de l'utilisation des zones était d'assister les sujets durant des tâches d'agencement 3D, l'affichage de ces zones a indirectement permis la visualisation des contraintes définies pour l'agencement en question. La visualisation des contraintes peut être très utile dans de nombreuses situations. Par exemple quand l'utilisateur propose un placement incorrect de certains objets, la visualisation des contraintes proposée par le système, peut constituer un retour visuel expliquant pourquoi le solveur rejette un tel placement. L'utilisateur pourrait donc modifier/supprimer des contraintes et relancer la résolution.

Finalement, l'expérimentation présentée complète celle décrite dans le chapitre 8 qui a montré que le solveur peut être utilisé pour résoudre un problème d'agencement 3D en proposant automatiquement des solutions. Pour bénéficier au maximum

des calculs effectués par le solveur tout en privilégiant l'interaction et la prise de décision humaine, l'utilisateur peut interroger le solveur pour placer automatiquement un ensemble d'objets et demander une assistance (visualisation des zones par exemple) quand il préfère placer manuellement un autre ensemble d'objets.

11

Conclusion

Les travaux présentés dans ce manuscrit s'inscrivent dans le cadre de la conception et du développement d'un outil d'aide à la décision conçu pour résoudre d'une manière interactive des problèmes d'agencement d'espaces sous contraintes. Ainsi, à travers un couplage étroit entre la programmation par contraintes (PPC) et les techniques de réalité virtuelle (RV), nous avons pu proposer une approche de résolution à la fois efficace, interactive et générique. Deux principaux axes ont été explorés : (1) la formulation et la résolution automatique de problèmes d'agencement 3D, et (2) l'interaction avec l'outil de résolution et l'assistance de l'utilisateur durant les tâches d'agencement.

L'état de l'art a permis de mettre en évidence d'une part, les difficultés relatives à la formulation et à la résolution de problèmes d'agencement, et d'autre part le potentiel de la PPC et de la RV à offrir des moyens pour la formulation, la résolution, la visualisation et l'interaction. L'analyse des approches de résolution proposées dans la littérature, a permis de constater que la plupart d'entre elles sont liées à des problèmes spécifiques et ne considèrent généralement que des contraintes géométriques (contraintes de non-chevauchement, de proximité, etc.). En outre, peu d'approches ont étudié l'interaction entre l'utilisateur et la méthode de résolution proposée [22, 54, 34]. En effet, ces approches permettent à l'utilisateur de modifier manuellement les solutions proposées dont la faisabilité est vérifiée à posteriori. L'analyse a également montré qu'aucun des travaux antérieurs n'a privilégié l'assistance de l'utilisateur afin que celui-ci puisse exprimer ses préférences de placement en positionnant manuellement certains composants.

Un aspect original de nos travaux réside dans la formulation des problèmes d'agencement 3D. En Effet, le cinquième chapitre a été consacré à l'identification et à la formulation d'un cas industriel impliquant une problématique d'agencement 3D relativement générique. Les exigences de conception et les besoins ont été clairement identifiés et une formulation sous forme de CSP a été également donnée. La méthodologie proposée peut être généralisée et appliquée à d'autres problèmes d'agencement (agencement de locaux, agencement de bateaux,etc.). Les contraintes proposées dans ce chapitre ne se limitent aux contraintes géométriques illustrant

les exigences naturelles d'un agencement (contraintes de non-chevauchement, distance, etc.), mais s'est étendue aux contraintes physiques. Celles-ci, très peu considérées dans les travaux antérieurs, permettent la prise en compte de l'effet possiblement néfaste que peuvent avoir certains objets (émetteurs des champs électriques, magnétiques ou thermiques) sur d'autres.

La mise en oeuvre de l'approche de résolution proposée a fait l'objet du sixième chapitre dans lequel l'architecture de l'outil conçu ainsi que le module d'interaction entre celui-ci et l'utilisateur ont été clairement décrits. Ce chapitre met en évidence l'une de nos principales contributions, liée à l'interactivité. En effet, l'utilisateur peut participer à la formulation d'un problème d'agencement en ajoutant ***dynamiquement*** de nouveaux objets et/ou de nouvelles contraintes. Il est également capable d'exprimer ses préférences en : (1) sélectionnant sa configuration préférée parmi celles proposées par le système, et en (2) modifiant manuellement la position de certains composants. Dans ce dernier cas, le solveur est sollicité non seulement pour vérifier la faisabilité du changement effectué, mais aussi pour assister l'utilisateur durant l'agencement manuel.

Un autre aspect original de nos travaux concerne l'assistance que peut fournir le système développé à l'utilisateur. Deux types d'assistance ont été développés et illustrés tout au long du septième chapitre : une assistance ***à posteriori*** et une assistance ***à priori***. En effet, le système peut assister l'utilisateur en : (1) annulant toute manipulation d'objets conduisant à une violation des contraintes, ou en (2) fournissant des informations visuelles pertinentes pour expliquer les causes d'infaisabilité ou d'échec (marquage des objets provocant un conflit). Quant à l'assistance à priori, plusieurs formes basées sur le principe du ***look ahead*** ont été implémentées afin de fournir une aide *prédictive*, visuelle et temps-réel avant même de placer un objet. La première forme permet à l'utilisateur d'identifier les zones 3D où le positionnement d'un objet donné provoquera une violation de contraintes ; la deuxième consiste à vérifier en temps-réel si la position courante de l'objet est valide ou non. Dans le cas d'une position courante invalide, la troisième forme d'assistance peut être déclenchée pour étendre la vérification de validité vers les postions "voisines" afin d'en trouver une qui satisfasse toutes les contraintes. Enfin, la quatrième et dernière forme d'assistance constitue une amélioration de la troisième puisque elle permet de déplacer automatiquement un objet vers une position proche proposée par le solveur.

Les travaux présentés dans les trois derniers chapitres de ce manuscript constituent une contribution expérimentale impliquant l'étude de la performance humaine et des aspects subjectifs durant des tâches d'agencement 3D. En effet, malgré l'intérêt des travaux proposés dans la littérature, très peu d'entre eux ont validé leur approche à travers des études expérimentales complètes et justifiées. C'est dans ce contexte, que nous avons mis en place une série cohérente et progressive de trois expérimentations.

La première étude présentée dans le huitième chapitre, a été menée pour étudier l'effet de l'intégration du solveur dans un environnement virtuel (EV) générique, à la fois sur des données objectives et subjectives mesurées lors d'une tâche d'agencement 3D. L'étude s'est basée sur une comparaison entre des tâches d'agencements

manuelles et semi-automatiques (assistées) plus ou moins complexes. Les résultats indiquent que l'utilisation du solveur a totalement éliminé les erreurs de placement et a considérablement réduit le nombre d'erreurs d'interaction, quelle que soit le niveau de complexité de la tâche. De point de vue subjectif, les résultats ont révélé que les sujets ont réalisé les agencements plus rapidement quand le solveur était utilisé et que celui-ci fournit une assistance utile à l'accomplissement des tâches.

Certaines données objectives et subjectives ont montré que la technique d'interaction proposée présente quelques lacunes liées à la perception 3D et au niveau relativement élevé de la charge de travail. Afin de pallier à ces défauts, différentes techniques d'interaction ont été proposées et évaluées dans le neuvième chapitre. Une étude expérimentale a été donc menée pour étudier l'influence de trois techniques d'interaction mono et multimodales (via la souris, vocale et souris-vocale) sur la performance, la charge de travail et la préférence des utilisateurs. Les résultats obtenus ont indiqué que les sujets ont réalisé la tâche plus rapidement quand la commande vocale était utilisée. Subjectivement, les sujets ont favorisé les techniques vocale et souris-vocale plutôt que la technique purement WIMP (via la souris) qui a induit un niveau assez élevé de charge de travail. La réalisation de cette expérimentation a mis en évidence l'intérêt de la multimodalité dans l'interaction avec la scène à agencer. Le choix de l'interaction la plus appropriée dépend fortement de la taille du problème d'agencement. En effet, une interaction souris-vocale semble plus efficace quand il s'agit d'agencer un grand nombre d'objets.

Enfin, nous avons abordé expérimentalement la problématique de l'assistance dans le dixième chapitre via la réalisation d'une étude expérimentale. L'objectif était d'étudier l'effet de la visualisation des zones impossibles sur la performance des utilisateurs durant une tâche d'agencement 3D. Les zones impossibles d'un objet représentent les surfaces 3D où le placement de cet objet conduit à une violation de contrainte(s). Les résultats objectifs et subjectifs ont montré que l'assistance apportée par le solveur a considérablement aidé les sujets durant les tâches. Même si ceux-ci ont passé plus de temps pour achever la tâche quand ils étaient assistés, l'utilisation du solveur a grandement minimisé le nombre d'erreurs d'agencement et le niveau de la charge de travail.

Même si l'approche proposée dans ce manuscrit présente des réponses à l'ensemble des verrous liés à la problématique de l'agencement 3D, il reste quelques pistes d'ordre scientifique et applicatif à explorer. Une perspective envisagée dans un futur proche concerne la formulation générique et l'implementation de la contrainte d'accessibilité aux composants. Une telle contrainte peut traduire la nécessité d'accéder à un composant donné pour une utilisation permanente ou pour des raisons de maintenance. La formulation donnée dans ce manuscript, qui consiste à définir des objets abstraits pour illustrer les espaces libres, est relativement simple et peu s'avérer inadaptée dans certaines situations. C'est pour cela que nous envisageons de formuler une contrainte d'accessibilité applicable aux problèmes d'agencement 2D et 3D. Une première idée consiste à décomposer l'espace à agencer en un nombre fini de zones 3D. Chaque zone représente un sommet, l'ensemble des sommets forme un graphe dont le sommet d'origine est l'entrée du contenant. Les sommets du graphe représentant les zones occupées par d'autres composants seront évalués par une valeur spécifique. Pour un composant donné, l'objectif consistera à

trouver un chemin entre le sommet d'origine et le sommet représentant la zone 3D incluant le composant en question. Pour se faire, un parcours en profondeur sera utilisé tout en "évitant" les sommets représentant les zones occupées.

Une deuxième perspective consiste à étendre notre approche pour supporter l'agencement des composants de forme quelconque. Le système utilisé actuellement approxime les composants à l'aide des volumes englobants de forme parallélépipèdique. Bien que cette approximation simplifie considérablement les calculs, la formulation et l'implémentation des contraintes, il est intéressant de prendre en compte la forme géométrique réelle de chaque composant pour une meilleure précision lors des agencements. L'idée est de représenter chaque composant par une hiérarchie de boîtes englobantes interconnectées par des contraintes de proximité. Cette modélisation modulaire des composants permettra une satisfaction plus naturelle des contraintes, comme par exemple placer un objet sur une chaise.

En terme d'interactivité, nous prévoyons d'accroître le réalisme des tâches d'agencement et d'intensifier la communication entre l'outil de résolution et l'utilisateur en offrant à celui-ci des moyens intuitifs pour exprimer ses préférences et commander facilement le solveur. Dans cet objectif, des canaux sensoriels tels que des retours haptiques seront utilisés. En effet, dans des espaces 3D surchargés (beaucoup d'objets) les retours visuels peuvent être insuffisants pour guider ou informer l'utilisateur, tandis que les modalités haptique permettent de restituer des informations relatives au contact entre l'avatar de l'utilisateur et les composants permettant ainsi une meilleure perception 3D.

Concernant la communication utilisateur-solveur, nous prévoyons de permettre à l'utilisateur d'exprimer des préférences particulières et ainsi participer à la phase de recherche de solutions. Par exemple, il pourra spécifier une zone 3D dans l'EV (en la limitant manuellement) et imposer au solveur de ne pas y placer des composants préalablement sélectionnés.

Dans certains contextes, et plus particulièrement dans la conception, la présentation de solutions (configurations d'objets) diversifiées peut faciliter la prise de décision notamment pour la validation d'un prototype final. Actuellement, nous utilisons une heuristique de sélection aléatoire de valeurs de variables pour diversifier dans une certaine mesure les solutions présentées à l'utilisateur. Dans le but de garantir cette diversité, nous prévoyons de définir un critère de sélection afin d'explorer les branches de l'arbre de recherche pouvant conduire à des solutions diversifiées. En d'autres termes, ce critère sera utilisé pour développer notre propre heuristique de sélection de valeurs de variables.

Au niveau applicatif, nos travaux vont être adaptés pour traiter d'autres problèmes d'agencement impliquant différentes formes de contenants ainsi que des contraintes spécifiques. En effet, l'entreprise *Thales Communications & Security* entend nous fournir les données techniques précises et nécessaires à la formulation et la modélisation de quelques cas industriels.

Nos travaux seront également intégrés à un projet médical dédié à des personnes présentant des atteintes de la mémoire de travail, de planification de l'action, ou des

troubles de l'attention. Dans ce contexte, la RV est utilisée comme support de rééducation grâce aux avantages qu'elle offre : interactivité, niveau de difficulté contrôlé, absence de danger. Le projet, nommé "cuisine virtuelle", propose différentes activités quotidiennes assistées (préparer le café par exemple) et se base sur la répétition de celles-ci pour que le patient puisse se détacher des assistances et devienne autonome. Notre système sera utilisé pour disposer les objets de la cuisine sur la table constituant l'espace de travail des patients. Pour chaque session de test, le système disposera les objets en tenant compte de l'ordre dans lequel ils seront manipulés, ou d'une façon totalement aléatoire afin d'habituer le patient à faire fasse au changement. La disposition manuelles des objets dans chaque session est bien évidemment une tâche longue et fastidieuse pour les médecins, ce qui nécessite son automatisation.

Table des matières

1 Introduction **1**

I État de l'art **5**

2 Problème de placement : Définition, formulation et résolution **7**
2.1 Introduction 7
2.2 Définition et exemples d'applications 8
2.3 Formulation 9
2.3.1 Les variables du problème 10
2.3.2 Contraintes 10
2.3.3 Le cas de l'accessibilité 10
2.4 Les problèmes de découpe et de conditionnement 11
2.5 Les problèmes d'agencement 12
2.5.1 Agencement 3D 13
2.6 Méthodes de résolution 17
2.6.1 Exemple de stratégie de résolution 17
2.7 Conclusion 19

3 Programmation par contraintes **21**
3.1 Introduction 21
3.2 Définition et exemples d'applications 21
3.2.1 Définition 21
3.2.2 Exemples d'applications 22
3.3 Modélisation 23
3.3.1 Problème de Satisfaction de Contraintes (CSP) 23
3.3.2 Problème d'optimisation sous contraintes 26
3.4 Filtrage et consistance locale 26
3.4.1 Filtrage 26
3.4.2 Niveaux de consistance locale 26
3.5 Propagation de contraintes 28
3.6 Recherche de solution 28
3.6.1 Méthodes de résolution 29
3.6.2 Heuristiques de branchement 33
3.7 Solveurs de contraintes 34
3.8 Conclusion 34

4 Réalité virtuelle **37**
4.1 Introduction . . . 37
4.2 Définitions . . . 37
4.3 Caractéristiques de systèmes de réalité virtuelle . . . 38
4.3.1 Environnement virtuel . . . 38
4.3.2 Interaction temps réel . . . 39
4.3.3 Immersion pseudo-naturelle . . . 39
4.3.4 Interfaçage comportemental . . . 39
4.4 Notions d'immersion et de présence . . . 39
4.5 Notion d'interaction . . . 40
4.6 Exemples d'applications . . . 40
4.7 Architecture générale d'un système de réalité virtuelle . . . 42
4.7.1 Interfaces d'entrée . . . 43
4.7.2 Interfaces de sortie . . . 43
4.7.3 Interfaces sensori-motrices . . . 43
4.8 Interfaces et périphériques d'interaction . . . 43
4.8.1 Les interfaces visuelles . . . 43
4.8.2 Les interfaces haptiques . . . 46
4.9 Techniques d'interaction 3D . . . 47
4.9.1 Définitions . . . 47
4.9.2 Classification des techniques d'interaction 3D . . . 48
4.9.3 Interaction multimodale . . . 50
4.9.4 Les guides virtuels . . . 50
4.10 Méthodes d'évaluation . . . 51
4.10.1 Techniques d'évaluation . . . 52
4.10.2 Mesures de performance . . . 54
4.11 Conclusion . . . 55

II Description du problème et approche proposée **57**

5 Identification et formulation du problème **59**
5.1 Introduction . . . 59
5.2 Identification du problème . . . 60
5.2.1 Contexte . . . 60
5.2.2 Objectifs . . . 60
5.2.3 Spécification . . . 61
5.3 Approche proposée . . . 62
5.4 Formulation du problème . . . 62
5.5 Architecture logicielle . . . 63
5.5.1 Diagramme de cas d'utilisation . . . 63
5.5.2 Diagramme de classes . . . 64
5.5.3 Diagramme d'activités . . . 65
5.6 Conclusion . . . 70

6 Système de résolution proposé **71**
6.1 Introduction . . . 71
6.2 Version préliminaire du système . . . 71
6.3 Description générale . . . 72
6.4 Architecture du système . . . 72

6.5 Module d'interaction avec le solveur . . . 75
6.6 Implémentation des contraintes . . . 75
6.7 Temps de résolution . . . 78
6.8 Fonctionnalités du système . . . 81
6.8.1 Algorithmes . . . 82
6.8.2 Accessibilité . . . 84
6.8.3 Orientation des objets . . . 84
6.8.4 Éclairage des objets . . . 85
6.8.5 Assistances visuelles statiques et dynamiques . . . 86
6.9 Conclusion . . . 87

7 Assistances visuelles anticipées 89
7.1 Introduction . . . 89
7.2 Le mécanisme d'anticipation (*look ahead*) . . . 89
7.2.1 Définition . . . 89
7.2.2 La technique du "Forward Checking" . . . 90
7.2.3 La technique "Full Look Ahead" . . . 91
7.2.4 Objectif . . . 91
7.3 Assistance visuelle statique : les zones impossibles . . . 91
7.4 Assistance visuelle dynamique . . . 93
7.5 Assistance visuelle dynamique améliorée . . . 97
7.6 Assistance visuelle dynamique avec placement automatique . . . 98
7.7 Conclusion . . . 99

III Expérimentations 101

8 Évaluation du système préliminaire 103
8.1 Introduction . . . 103
8.2 Objectif . . . 103
8.3 Protocole expérimental . . . 104
8.4 Description des tâches d'agencement . . . 105
8.4.1 Agencement manuel . . . 105
8.4.2 Agencement semi-automatique . . . 107
8.4.3 Données recueillies . . . 107
8.5 Résultats et discussion . . . 109
8.5.1 Résultats objectifs . . . 110
8.5.2 Aspects subjectifs . . . 112
8.6 Conclusion . . . 117

9 Évaluation de techniques d'interaction vocale et de type WIMP 119
9.1 Introduction . . . 119
9.2 Objectif . . . 120
9.3 Protocole expérimental . . . 120
9.4 Description des tâches . . . 122
9.4.1 Agencement via une interface de type WIMP . . . 122
9.4.2 Agencement avec commande vocale . . . 122
9.4.3 Agencement avec commande multimodale . . . 123
9.4.4 Stratégie de lancement de la résolution . . . 124
9.5 Données recueillies . . . 125

9.6 Résultats et discussion . . . 125
9.6.1 Résultats objectifs . . . 125
9.6.2 Aspects subjectifs . . . 127
9.7 Conclusion . . . 129

10 Évaluation de l'assistance visuelle anticipée 131
10.1 Introduction . . . 131
10.2 Objectif . . . 131
10.3 Protocole expérimental . . . 132
10.4 Description des tâches . . . 133
10.4.1 Agencement non assisté . . . 134
10.4.2 Agencement assisté . . . 134
10.5 Données recueillies . . . 134
10.6 Résultats et discussion . . . 136
10.6.1 Résultats objectifs . . . 136
10.6.2 Aspects subjectifs . . . 137
10.7 Conclusion . . . 140

11 Conclusion 143

Liste des publications 161

Références 161

Liste des tableaux

6.1 *Tableau récapitulatif des contraintes développées.* 80

8.1 *Les valeurs de p obtenues pour chaque tâche de l'expérimentation.* . 109
8.2 *Effet d'interaction et effet principal de CT et A sur le temps d'agencement et l'erreur d'interaction 3D.* 111
8.3 *Temps (sec.) d'agencement (a) et nombre d'erreurs (b).* 111
8.4 *Le nombre total des erreurs de placement et de distance.* 112

9.1 *Liste des mots utilisés pour la commande vocale relative à la sélection des objets et des contraintes.* 124
9.2 *Moyennes et écart-types des temps de réalisation de la tâche pour chaque type d'interaction.* 126

10.1 *Effet de l'assistance (A) sur le temps, les erreurs d'agencement et le nombre des actions des sujets.* 136
10.2 *Moyennes et écart-types dans chaque condition expérimentale.* . . . 138

Table des figures

2.1 *Exemples d'applications impliquant la problématique d'agencement (Images tirées de Jacquenot* [87]). 9
2.2 *Classification des problèmes de découpe et de conditionnement proposée par Washer et al.* [63]. 12
2.3 *Interface du logiciel DIM 3D* 1.0. 15
2.4 *Détection de collisions entre deux objets lors d'un agencement manuel du VAB.* 15
2.5 *Modèle géométrique 2D du problème d'agencement du shelter proposé par Bénabès* [22]. 16
2.6 *Agencement 3D proposé par la méthode de résolution (Image tirée de Bénabès* [22]). 16
2.7 *La méthode hybride proposée par Jacquenot [87].* 18

3.1 *Discrétisation de l'espace à agencer.* 25

4.1 *Triangles de la Réalité Virtuelle (RV) (D'après [32]) et de la Réalité Augmentée (RA) (D'après [43])* 38
4.2 Simulation d'opérations chirurgicales (a) et interface de simulateur de vol (b). 41
4.3 Visualisation 3D à l'échelle 1 :1 de prototype virtuel d'une voiture. . 42
4.4 *Architecture d'un système de RV (D'après [123]).* 42
4.5 Exemples de configurations visuelles semi-immersives : (a) mur et (b) plan de travail ou workbench. 45
4.6 *Exemples de visio-casques : (a) visio-casque Sony de dernière génération et (b) visio-casque semi-transparent (D'après [37]).* 46
4.7 Classification des interfaces haptiques. 46
4.8 Contexte de l'interaction 3D. 47
4.9 Taxonomies proposées par Bowman pour : (a) la sélection et (b) la manipulation d'objets virtuels (D'après [24]). 49
4.10 Taxonomie proposée pour le *contrôle d'application* (D'après [68]). . 49
4.11 Utilisation des guides visuels pour la manipulation du robot *FANUC LR-MATE 200i* (D'après [110]). 51
4.12 *Classification des techniques d'évaluation (D'après [28]).* 52

5.1 *Modéle CAO d'un shelter (Image tirée de Bénabès* [22]). 60
5.2 *Diagramme des cas d'utilisations.* 64
5.3 *Diagrammes de classe.* 66
5.4 *Diagrammes d'activité "Zones impossibles".* 67
5.5 *Diagrammes d'activité "État de la position courante".* 68
5.6 *Diagrammes d'activité "Alternatives de placement".* 69

6.1 *L'interface graphique du système proposé.* 73
6.2 *Architecture générale du Système : communication solveur-EV.* . . . 74
6.3 *Module d'interaction avec le solveur.* 75
6.4 *Evolution du temps de calcul en fonction du nombre des objets et du nombre des contraintes.* 79
6.5 *Exemple illustrant le mécanisme du CSP_backtrack.* 82
6.6 *Exemple d'utilisation des objets abstraits (objets en pointillés) pour accéder aux postes de travail (P1 à P6) d'un shelter.* 87

7.1 *Exemples de techniques de look ahead : Forward Checking (a) et Full Look ahead (b).* 90
7.2 *Visualisation des zones impossibles pour la chaise du bureau.* 94
7.3 *Caractérisation temps réel de chaque position 3D proposée par l'utilisateur.* 96
7.4 *Proposition d'une alternative de placement de la table dont la position courante est invalide.* 98

8.1 *Copie d'écran de l'environnement virtuel expérimental.* 105
8.2 *Sujet effectuant un agencement manuel. Une caméra OptiTrack™ et deux réflecteurs positionnés sur l'avant de la Wiimote™, permettent la capture des mouvements du sujet.* 106
8.3 *Sujet sélectionnant une contrainte (via le Nunchuk™) à appliquer sur des objets de la scène 3D.* 107
8.4 *Exemple de séquences de sélections rèussies/échouées observées dans chaque condition.* 113
8.5 *Le nombre moyen de selection pour chaque objet de l'EV.* 114
8.6 *Les scores moyens des indices de la charge de travail.* 114
8.7 *Données subjectives concernant la satisfaction des sujets (a) et l'assistance fournie par le solveur (b).* 115

9.1 *Configuration initiale de la scène (a) et de l'agencement à réaliser (b).* 121
9.2 *Un sujet sélectionne les objets et les contraintes à l'aide de la souris.* 123
9.3 *Un sujet sélectionne vocalement les objets et les contraintes.* 123
9.4 *Boîte à moustaches (Box plot) concernant le temps de réalisation de la tâche pour chaque type d'interaction.* 126
9.5 *Processus d'apprentissage relatif au temps de réalisation de la tâche en fonction du type d'interaction.* S (commande via la souris), V (commande vocale) et S&V (commande multimodale). 127
9.6 *Scores moyens des indices de la charge de travail pour chaque type d'interaction.* 128
9.7 *Évaluation subjective de chaque type d'interaction.* 129

10.1 *Un sujet agence la scène 3D à l'aide la visualization des zones impossibles.* 132
10.2 *Copie d'écran de la condition initiale de la tâche.* 133
10.3 *Scores moyens des indices de la charge de travail pour chaque condition.* 138
10.4 *Satisfaction des sujets dans chaque condition.* 139
10.5 *Besoin d'assistance ressenti par les sujets pour chaque condition.* . 139

10.6 *Données subjectives concernant la facilité de l'agencement (FA) et le souvenir des contraintes (SC) ressentis durant les tâches assistées.* 140

Listes des algorithmes

1 Algorithme de la méthode *generate-and-test* 30
2 Algorithme de la méthode *simple retour arrière* 30
3 Algorithme *Forward Checking* 31
4 Ajout d'une nouvelle contrainte pour l'objet_i 83
5 Ajout d'un nouveau objet à la scène 3D 84
6 Suppression d'une contrainte c_j 85
7 Suppression d'un objet obj_i 86
8 Zones impossibles pour l'objet obj_i 92
9 Assistance visuelle dynamique 95
10 Assistance visuelle dynamique améliorée 98

Liste des publications

1. Kefi, M., Barichard, V., and Richard, P. (2012). **A Constraint-Solver Based Tool for User-Assisted Interactive 3D Layout**. *In 24th IEEE International Conference on Tools with Artificial Intelligence*. Athens, Greece

2. Kefi, M., Richard, P., and Barichard, V. (2011). **3D Interactive objects Layout Using Virtual Reality Techniques and Constraint Programming**. *In International Conference on Computer Graphics Theory And Applications (GRAPP)*, pages 308-313. Vilamoura-Portugal

3. Kefi, M., Richard, P., and Barichard, V. (2011). **Using Virtual Reality and Constraint Programming Techniques for 3D Layout**. *In International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 51-54. Plzen-République Tchèque

4. Kefi, M., Richard, P., and Barichard, V. (2010). **Interactive Configuration of Restricted Spaces using Virtual Reality and Constraint Programming Techniques**. I*n International Conference on Computer Graphics Theory And Applications (GRAPP)*, pages (384-389). Angers-France

5. Kefi, M., Barichard, V., and Richard, P. (2011). **Aide à la Décision via les Techniques de Réalité Virtuelle et de la Programmation par Contraintes**. **Congrès Annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF)**, pages (883-884). Saint-Etienne-France

6. Kefi, M., Richard, P., and Barichard, V. (2010). **Aide à la Décision via les techniques de Réalité Virtuelle : Application à l'aménagement sous contraintes**. *3 ème journée des Démonstrateurs en Automatique*. Angers-France

Bibliographie

[1] Chip. *http ://www.cosytec.com*. 34

[2] The eclipse constraint programming system. *http ://eclipseclp.org*. 34

[3] Elumens : http ://www.elumens.com/. 45

[4] Ibm : Ibm ilog cplex cp optimizer. *http ://www-01.ibm.com/software/integration/ optimization/cplex-cp-optimizer/*. 22, 34

[5] Koalog constraint solver. 34

[6] Sicstus prolog. *http ://www.sics.se/isl/sicstuswww/site/index.html*. 34

[7] Choco team : Choco : an open source java constraint programming library. *http ://choco. mines-nantes.fr*, 2012. 34

[8] Microsoft speech api. *http ://msdn.microsoft.com/en-us/library/ms723627(v=vs.85).aspx*, 2012. 123

[9] Mistral. *http ://www.4c.ucc.ie/ ehebrard/mistral/doxygen/html/main.html*, 2012. 34

[10] S. Martello A. Lodi and D. Vigo. Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics*, 123(1-3) :379–396, 2002. 7

[11] K. Ahlers, D. Breen, and D. Greer. augmented vision system for industrial applications. *Telemanipulator and Telepresence Techonolgies, volume 2351, pages 345-359*, 1994. 44

[12] E. Angelelli, R. Mansini, and M. Vindigni. Look-ahead heuristics for the dynamic traveling purchaser problem. *Computers & Operations Research*, December, p. 1867-1876 2011. 32

[13] S. Aukstakalnis, D. Blatner, and F. Roth Stephen. *Silicon Mirage - The Art and Science of Virtual Reality*. Peachpit Press, 1992. 38

[14] H.F. Teng B. Zhang and Y.J. Shi. Layout optimization of satellite module using soft computing techniques. *Applied Soft Computing*, 8(1) :507–521, 2008. 14

[15] J. Van Baren and W. Ijsselsteijn. Measuring presence : A guide to current measurement approaches. *IST-FET OmniPres project IST-2001-39237*, Deliverable N^o 5 2004. 54

[16] A. Baykan. Automated space planning with global constraints and variable number of use-areas. In *Advances in Building Informatics*. Reza Behesti, 2001. 1, 103

[17] J. O. Berkey and P. Y. Wang. Two dimensional finite bin packing algorithms. *Journal of Operational Research Society*, 38 :423–429, 1987. 11

[18] C. Bessière. Arc-consistency and arc-consistency again. *Artificial Intelligence*, pages p. 179–190, 1994. 27

[19] C. Bessière, A. Chmeiss, and L. Sais. Neighborhood-based variable ordering heuristics for the constraint satisfaction problem. *Lecture Notes in Computer Science*, pages p. 565–569, Springer, 2001. 27

[20] S. Biggs and M. D. Srinivasan. chapter 5 : Haptic Interfaces. Lawrence Erlbaum Associates Pub., K. Stanney (Ed.), p. 93-116, 2002. 47

[21] M. Billinghurst, H. Kato, and I. Poupyrev. The magicbook : Moving seamlessly between reality and virtuality. *IEEE Computer Graphics and Applications*, 21, May/June, p. 6-8 2001. 44

[22] J. BÉNABÈS. Optimisation integrée et interactive pour l'agencement d'espace. *Thése de Doctorat, Ecole Centrale de Nantes*, 2011. 1, 11, 13, 15, 16, 60, 89, 143, 155

[23] G. Bouyer. Rendu multimodal en réalité virtuelle : Supervision des interactions au service de la tâche. *Thése de Doctorat, Université Paris XI*, 2007. 50

[24] D. Bowman. Interaction techniques for common tasks in immersive virtual environments. April, PhD. Thesis, Georgia Institute of Technology 1998. 48, 49, 155

[25] D. Bowman, E. Davis, A. Badre, and L. Hodges. Maintaining spatial orientation during travel in an immersive virtual environment. *Presence : Teleoperators & Virtual Environments*, 18 :619–31, 1999. 53

[26] D. Bowman and L. Hodges. Formalizing the design, evaluation, and application of interaction techniques for immersive virtual environments. *J. Visual Languages and Computing*, 10 :37–53, 1999. 53

[27] D. Bowman, E. Kruijff, J. LaViola, and I. Poupyrev. An introduction to 3-d user interface design. *Presence : Teleoperators & Virtual Environments*, 10(1) :96–108, 2001. 47

[28] D. Bowman, E. Kruijff, J. LaViola, and I. Poupyrev. *Evaluation of 3D User Interfaces*. Addison-Wesley Pub. ISBN : 0-201-75867-9, 2004. 39, 47, 48, 51, 52, 54, 120, 155

[29] D. Bowman, D. Ohnson, and L. Hodges. Testbed evaluation of virtual environment interaction techniques. *Proc. ACM Symp. on Virtual Reality Software and Technology (VRST'99)*, 18, London, UK, p. 26-33, 1999. 53

[30] W.-R. Bukowski and H.-C. Séquin. Object associations. In *ACM Symp. On Interactive 3D Graphics, Monterey, CA, USA*, 1995. 81

[31] G. Burdea. Force and Touch Feedback for Virtual Reality, John Wiley & Sons, Inc. (Pub.) , NY, USA. 339 p. ISBN : 0-471-02141-5, 1996. 46

[32] G. Burdea and P. Coiffet. La Réalité Virtuelle, Hermes Sciences Pub., 1993. 38, 155

[33] W. Buxton. Lexical and pragmatic considerations of input structures. *Computer Graphics*, 17 :31–7, 1983. 43

[34] C. Calderon, M. Cavazza, and D. Diaz. A new approach to the interactive resolution of configuration problems in virtual environments. *Lecture notes in computer science*, 2733 :112 – 122, 2003. 1, 13, 16, 81, 103, 119, 143

[35] D. Chamaret. Plate-forme de réalité virtuelle pour l'étude de l'accessibilité et de l'extraction de lampes sur prototype virtuel automobile. *Thése de Doctorat, Université d'Angers*, 2010. 50

[36] S. Coquillart, P. Fuchs, J. Grosjean, and A. Paljic. Chap. : Techniques d'immersion et d'interaction - Les primitives comportementales virtuelles, Vol. 1, Le Traité de la Réalité Virtuelle, Presses de l'Ecole des Mines, 2^{nd} Ed., 2004. 40

[37] Physical Optics Corporation. Optical see-through hmd. http ://www.poc.com/. 46, 155

[38] C. Cruz-Neira, D. Sandin, and T. F. DeFanti. Surround-screen projection-based virtual reality : The design and implementation of the cave. *Proc. 20^{th} Ann. Conf. on Computer Graphics and Interactive Techniques (ACM SIGGRAPH'93)*, Anaheim, CA, USA, 27 (2), p. 135-42, 1993. 45

[39] F. Daniel and D.Rina. Look-ahead value ordering for constraint satisfaction problems. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, (IJCAI)*, pages 572–578, 1995. 17, 89, 91

[40] R. Dechter and I. Meiri. Experimental evaluation of preprocessing algorithms for constraint satisfaction problems. *Artificial Intelligence*, pages p. 211–241, 1994. 33

[41] K. A. Dowsland and W. B. Dowsland. Packing problems. *European Journal of Operational Research*, 56(1) :2–14, 1992. 11

[42] P. Dragicevic. Un modèle d'interaction en entrée pour des systèmes interactifs multi-dispositifs hautement configurables. Thèse de Doctorat de l'Université de Nantes, Spécialité Informatique Mars 2004. 47

[43] E. Dubois. Chirurgie augmentée, un cas de réalité augmentée : Conception et réalisation centrées sur l'utilisateur. Thèse de Doctorat, Université Joseph Fourier, Grenoble 1 Juillet 2001. 38, 155

[44] H. Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational research*, 44 :145–159, 1990. 7

[45] H. Dyckhoff and U. Finke. Cutting and packing in production and distribution : Typology and bibliography. *Springer-Verlag, New York*, 1992. 11

[46] G. Kendall E. Burke, R. Helier and G. Whitwell. Complete and robust no-fit polygon generation for the irregular srock cutting problem. *European Journal of Operational Research*, 179 :27–49, 2007. 10

[47] R. Morabito E. G. Birgin and F. H. Nishihara. A note on an l-approach for solving the manufacturer's pallet loading problem. *Journal of the Operational Research Society*, 56 :1448–1451, 2005. 11

[48] G. Kendall E. K. Burke and G. Whitwell. A new placement heuristic for the orthogonal stock-cutting problem. *Operations Resarch*, 52(4) :655–671, 2004. 11

[49] Freuder E.C. A sufficient condition for backtrack-free search. *Journal of the Association for Computing Machinery*, pages p. 24–32, 1982. 27

[50] J. Egeblad. Mémoire de master, department of computer science, university of copenhagen. 2003. 9

[51] Unity 3D Game Engine. http ://unity3d.com. 2012. 72

[52] D. Hix et H. R. Hartson. Chap. : Heuristic Evaluation, in Usability Inspection Methods, J. Nielsen and R. L. Mack (Eds), John Wiley & Sons, Inc. (Pub.) , NY, p. 25-62, 1994. 52

[53] M. R. Garey F. R. K. Chung and D. S. Johnson. On packing two-dimensional bins. *SIAM Journal on Algebraic and Discrete Methods*, 3 :66–76, 1983. 11

[54] F. Fages, S. Soliman, and R. Coolen. Clpgui : A generic graphical user interface for constraint logic programming. *Constraints*, 9 :241 – 262, 2004. 1, 13, 16, 81, 89, 103, 119, 143

[55] T. Fernando, N. Murray, K. Tan, and P. Wimalaratne. Software architecture for a constraint-based virtual environment. *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 147 – 154, 1999. 13, 16, 81, 89

[56] F. Focacci, F Laburthe, and A. Lodi. Local search and constraint programming. *In Fred Glover and Gary Kochenberger, editors, Handbook of Metaheuristics*, pages p. 3–44, 2002. 33

[57] J. B. G. Frenk and G. Galambos. Hybrid next-fit algorithm for the twodimensional rectangle bin-packing problem. *Computing*, 39(3) :201–217, 1987. 11

[58] E.C. Freuder. Using inference to reduce arc consistency computation. *In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages p. 592–598, 1995. 27

[59] P. Fuchs and J.-M. Burkhardt. volume 1, chapter Approche théorique et pragmatique de la réalité virtuelle. Presses de l'Ecole des Mines, Le Traité de la Réalité Virtuelle, 2^{nd} Ed., 2004. 47

[60] P. Fuchs, G. Moreau, A. Berthoz, and J.-L. Vercher. volume 1 : L'homme et l'environnement virtuel. Presses de l'Ecole des Mines, Le Traité de la Réalité Virtuelle, P. Fuchs and G. Moreau (Eds), ISBN : 2-911762-63-0 2006. 39

[61] P. Fuchs, G. Moreau, and J. P. Papin. Presses de l'Ecole des Mines de Paris, Le Traité de la Réalité Virtuelle, 2001. 44

[62] P. Fuchs and P. Stergiopoulos. volume 1, chapter Les interfaces manuelles sensori-motrices, interfaces à retour d'effort. Presses de l'Ecole des Mines, Le Traité de la Réalité Virtuelle, 2^{nd} Ed. 47

[63] H. Haußner G. Wäscher and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183 :1109–1130, 2007. 7, 11, 12, 155

[64] J. Gabbard, D. Hix, and J. E. Swan. User-centered design and evaluation of virtual environments. *IEEE Computer Graphics and Applications*, 19 :51–9, 1999. 53

[65] M. R. Garey and D. S. Johnson. Computers and intractability : A guide to the theory of np-completeness. *W. H. Freeman & Co., New York (USA)*, 1979. 11

[66] J. Gaschnig. A general backtrack algorithm that eliminates most redundant tests. *International Joint Conference on Artificial Intelligence (IJCAI'77)*, 1977. 31, 89

[67] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock problem-part ii. *Operations Resarch*, 11(6) :863–888, 1963. 11

[68] J. Grosjean. Environnements virtuels : contrôle d'application et exploration de scènes 3d. Thèse de Doctorat de l'Université de Versailles - Saint-Quentin, Spécialité Informatique 2003. 48, 49, 155

[69] G. Scheithauer H. Dyckhoff and J. Terno. Annotated bibliographies in combinatorial optimization, chichester. *Chapitre Cutting and Packing (C&P)*, pages 393–413, 1997. 11

[70] D. Liu H. F. Teng, S. L. Sun and Y. Z. Li. Layout optimization for the objects located within a rotating vessel - a three-dimensional packing problem with behavioral constraints. *Computers & Operations Research*, 28 :521–535, 2001. 14

[71] Y. Tu H. T. Dean and J. F. Raffensperger. An improved method for calculating the no-fit polygon. *Computer and Operations Research*, 33(6) :1521–1539, 2006. 10

[72] L. Hamon. Modélisation et interaction temps réel avec des structures dynamiques de type l-système : Application aux plantes virtuelles. *Thése de Doctorat, Université d'Angers*, 2011. 50

[73] R. Haralick and G. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, pages p. 263–313, 1980. 30, 89

[74] R. M. Haralick and G. L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *International Joint Conference on Artificial Intelligence*, pages p. 356–364, 1979. 33

[75] S. Hart and L. Staveland. Development of nasa-tlx (task load index) : Results of empirical and theoretical research. In *Human mental workload.*, pages 139–183, 1988. 104, 122, 132

[76] M. Heim, M. Featherstone, and R. Burrows. The design of virtual reality. *Cyberspace/ Cyberbodies/Cyberpunk Cultures of Technological Embodiment*, pages 65–78, 1995. 38

[77] R. M. Held and N. I. Durlach. Telepresence. *Presence : Teleoperators & Virtual Environments*, 1992. 40

[78] E. Hellström, M. Ivarsson, J. Aslund, and L. Nielsen. Look-ahead control for heavy trucks to minimize trip time and fuel consumption. *Control Engineering Practice*, August, p. 245-254 2008. 32

[79] D. Hix and H. R. Hartson. Developing User Interfaces : Ensuring Usability Through Product & Process, Wiley Professional Computing, John Wiley & Sons, Inc. (Pub.) , NY, USA., 1993. 52

[80] E. Huang and R. E. Korf. New improvements in optimal rectangle packing. *IJCAI*, pages p. 511–516, 2009. 11

[81] M. Billinghurst et T. Ichikawa I. Poupyrev, S. Weghorst. A framework and testbed for studying manipulation techniques for immersive vr. *Proc. ACM Int. Symp on Virtual Reality Software and Technology (VRST'97)*, 1997. 53

[82] J. Isdale. What is virtual reality ? In *In a homebrew introduction and information resource list*, 1993. 43

[83] K. Shimada J. Cagan and S. Yin. A survey of computational approaches to the threedimensional layout problems. *Computer-Aided Design*, 34 :597–611, 2002. 8

[84] P. Dastidar S. Szykman J. Cagan, R. Clark and P. Weisser. Hvac cad layout tools : A case study of university-industry collaboration. 1996. 14

[85] C. R. Johnson J. D. Brederson, M. Ikits and C. D. Hansen. The visual haptic workbench. In *5th Phantom Users Group Workshop*, 2000. 50, 123

[86] R. Jacob, L. Silbert, C. McFarlane, and M. Mullen. Integrability and separability of input devices. *Transactions on Human-Computer Interaction*, 1 :3–26, 1994. 43

[87] G. JACQUENOT. Méthode générique pour l'optimisation d'agencement géometrique et fonctionnel. *Thése de Doctorat, Ecole Centrale de Nantes*, 2009. 1, 7, 8, 9, 13, 14, 17, 18, 81, 155

[88] J. Joseph, Jr. LaViola, J. Huffman, and A. Bragdon. The influence of head tracking and stereo on user performance with non-isomorphic 3d rotation. In *Eurographics Association*, pages 111–118, 2008. 117

[89] N. Jussien and O. Lhomme. Local search with constraint propagation and conflict-based heuristics. *Artificial Intelligence*, pages p. 21–45, 2002. 33

[90] A. Kadri, A. Lecuyer, J. Burkhardt, and S. Richir. The influence of visual appearance of user's avatar on the manipulation of objects in virtual environments. *Proc. IEEE Int. Conf. on Virtual Reality (IEEE VR'07)*, Sept. 95, 3, p. 291-2, 2007. 40

[91] M. Kallman and D. Thalman. Direct 3d interaction with smart objects. In *ACM International Symposium on Virtual Reality Software and Technology, London. UK*, 1999. 81

[92] R. E. Korf. Optimal rectangle packing : Initial results. *ICAPS*, pages p. 287–295, 2003. 11

[93] R. E. Korf. Optimal rectangle packing : New results. *ICAPS*, pages p. 142–149, 2004. 11

[94] W. Krüger and B. Fröhlich. The responsive workbench. *IEEE Computer Graphics and Applications*, 2001. 44

[95] H. Yuanjun L. Huyao and J. A. Bennell. The irregular nesting problem : a new approach for nofit polygon calculation. *Journal of the Operational Research Society*, 58(9) :1235–1245, 2007. 10

[96] A. Lécuyer, C. Megard, J. Burkhardt, T. Lim, S. Coquillart, and P. Coiffet. The effect of haptic, visual and auditory additional information on an insertion task on the holobench. 44

[97] A. Albano M. Adamowicz. Nesting two-dimensional shapes in rectangular modules. *Computer-Aided Design*, 8(1) :27–33, 1976. 10, 11

[98] E. Maisel M. Veyret and J. Tisseau. Guide virtuel autonome immergé dans un environnement réel dynamique. architecture générale et application à la visite guidée d'un aquarium marin. *Technique et Science Informatiques*, 28(6-7) :831–855, 2009. 51

[99] A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, pages p. 99–118, 1977. 27

[100] A. MALAPERT. Techniques d'ordonnancement d'atelier et de fournées basées sur la programmation par contraintes. *Thése de Doctorat- Université de Nantes, Faculté des Sciences et des Techniques*, 2011. 34

[101] B. Medjdoub. Constraint-based adaptation for complew space configuration in building services. *Journal of Information Technology in Construction*, 243(2007) :627–636, 2004. 14, 16, 119

[102] M. Meehan. *Physiological reaction as an objective measure of presence in virtual environments*. PhD thesis, University of North Carolina at Chapel Hill, 2001. 40, 54

[103] P. Milgram, D. Drascic, J. J. Grodski, A. Restogi, S. Zhai, and C. Zhou. Merging real and virtual worlds. *Proc. IMAGINA'95*, Monte-Carlo, Monaco, Feb. 1-3, p. 218-30, 1995. 38

[104] D. Mogilev, K. Kiyokawa, M. Billinghurst, and J. Pair. Ar pad : An interface for face-to-face ar collaboration. *Proc. ACM Int. Symp. Transactions on Computer Human Interaction (ACM CHI'02)*, pages p. 654–5, 2002. 44

[105] R. Mohr and T.C. Henderson. Arc and path consistency revisited. *Artificial Intelligence*, pages p. 225–233, 1986. 27

[106] D. Nahon. Salles immersives et cubes de réalité virtuelle, une première mondiale sur pc : le sas cube. *Imagina*, 2002. 45

[107] L. Nigay and F.Vernier. Design method of interaction techniques for large information spaces. In *AVI*, pages 37–46, 1998. 50

[108] OptiTrack. Optical motion capture systems and tracking software, http ://www.naturalpoint.com/optitrack/. 2011. 105

[109] S. Otmane. Télétravail robotisé et réalité augmentée : application à la téléopération via internet. *Thése de Doctorat, Université d'Evry-Val d'Essonne*, 2000. 51

[110] S. Otmane and F. Davesne. Utilization of human scale spidar-g in the framework of assistance to 3d interaction in a semi-immersive ar/vr platform. *In Joint Virtual Reality Conference EGVE-ICAT-EURO VR (JCVR)*, pages p. 129–134, 2009. 51, 155

[111] M. Ouh-Young, D. Beard, and F. W. Brooks. Force display performs better than visual display in a simple 6-d docking task. *Proc. IEEE Int. Conf. on Robotics and Automation*, 1989. 47

[112] G. Calvet P. De Loor, L. Le Bodic and J. Tisseau. Using simulation for automating usability evaluation of multimodal systems. *Journal of human-machine interaction*, 2006. 50, 123

[113] D. Gomez P. Richard, G. Burdea and P. Coiffet. A comparison of haptic, visual and auditive force feedback for deformable virtual objects. In *International conference on artificial reality and tele-existence*, pages p. 49–62, 1994. 50, 123

[114] F. Inglese P. Lucidarme P. Richard, D. Chamaret and J. Ferrier. Human-scale virtual environment for product design : Effect of sensory substitution. *The International Journal of Virtual Reality*, 2006. 104

[115] A. Paljic, S. Coquillart, J. M. Burkhardt, and P. Richard. A study on distance of manipulation in virtual environments : the closer the better ? *Proc.* 7^{th} *Ann. Symp. on Immersive Projection Technology (IPT'02)*, Orlando, USA, March 24-25 2002. 44

[116] J. P. Papin and P. Fuchs. volume 1, chapter Les sens et les réponses motrices de l'homme. Presses de l'Ecole des Mines, Le Traité de la Réalité Virtuelle, 2^{nd} Ed., 2004. 43

[117] G. Pesant and M. Gendreau. A view of local search in constraint programming. *Proceedings of Principles and Practice of Constraint Programming (CP 2000)*, pages p. 353–366, Springer, 1996. 33

[118] C. Pfefferkorn. A heuristic problem solving design system for equipment or furniture layouts. *Communications of the ACM*, 18(5), 1975. 62

[119] S. Prestwich. A hybrid search architecture applied to hard random 3-sat and low-autocorrelation binary sequences. *Proceedings of Principles and Practice of Constraint Programming (CP 2000)*, pages p. 337–352, Springer, 2000. 33

[120] T. Ramesh. Traveling purchaser problem. *Opsearch*, pages p. 78–91, 1981. 32

[121] J. Rekimoto and K. Nagao. The world through the computer : Computer augmented interaction with real world environments. *Proc. Symp. on User Interface Software and Technology (UIST'95)*, pages p. 29–36,, 1995. 44

[122] J. C. Régin. Modélisation et contraintes globales en programmation par contraintes. Habilitation à diriger des recherches- Université de Nice Nov. 2004. 22

[123] P. Richard. Analyse de l'interaction homme-monde virtuel lors de tâches de manipulation d'objets déformables. Thèse de doctorat de l'université Pierre et Marie Curie 1996. 42, 120, 155

[124] P. Richard and P. Coiffet. Dextrous haptic interaction in virtual environments : human performance evaluation. *Proc. 8^{th} IEEE Int. Work. on Robot and Human Interactive Communication (RO-MAN'99)*, Pisa, Italy, Oct. 27-29, p. 315-20, 1999. 47

[125] P. Richard, P. Coiffet, A. Kheddar, and R. England. Human performance evaluation of two handle haptic devices in a dextrous virtual telemanipulation task. *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'99)*, Taejon, Korea, Oct. 17-21, 3, p. 1543-48, 1999. 47

[126] R. Richard. Contribution à l'étude de l'interaction 3d multimodale et de la performance humaine en environnement virtuel. *Rapport de HDR, Université d'Angers*, 2011. 50

[127] L. B. Rosenberg. The use of virtual fixtures as perceptual overlays to enhance operator performance in remote environments. *Technical Report AL/CF-TR-1994-0089*, 1992. 50

[128] L. B. Rosenberg. Virtual fixtures : Perceptual tools for telerobotic manipulation. In *VR*, pages 76–82, 1993. 50

[129] V. Gaildrat S. Sanchez, O. Leroux and H. Luga. Résolution d'un problème d'aménagement spatial à l'aide d'un algorithme génétique. *Association Française d'Informatique Graphique, Grenoble*, 2000. 17

[130] J. Cagan S. Yin and P. Hodges. Layout optimization of shapeable components with extended pattern search applied to transmission design. *Journal of Mechanical Design*, 126 :188–190, 2004. 14

[131] S. Sanchez, O. Le Roux, F. Inglese, H. Luga, and V. Gaildart. Constraint-based 3d-object layout using a genetic algorithm. 2002. 1, 13, 16, 81, 89

[132] SASCube. http ://www.sascube.com. 45

[133] C. Schulte, G. Tack, and M. Lagerkvist. *Modeling with Gecode*. 2012. 34

[134] C. Sechen. Vlsi placement and global routing using simulated annealing. *Kluwer Academic Publishers*, 1998. 17

[135] M. Slater, C. Alberto, and M. Usoh. *In the building or through the window ? An experimental comparison of immersive and non-immersive walkthroughs*. Leeds UK 1995. 39

[136] M. Slater and A. Steed. A virtual presence counter. *Presence : Teleoperators & Virtual Environments*, 9(5) :413–34, 2000. 39

[137] B. M. Smith. Succeed-first or fail-first : A case study in variable and value ordering. 33

[138] C. Solnon. http ://liris.cnrs.fr/csolnon/site-ppc/e-miage-ppc-som.htm. 2012. 22

[139] L. Song and N. Eldin. Adaptive real-time tracking and simulation of heavy construction operations for look-ahead scheduling. *Automation in Construction*, November, p. 32-39 2012. 31

[140] L. Sterling and Shapiro E. The art of prolog. *The MIT Press*. 34

[141] W. Stuerzlinger and G. Smith. Efficient manipulation of object groups in virtual environments. 2000. 81

[142] S. Szykman and J. Cagan. Constrained three-dimensional component layout using simulated annealing. *Journal of Mechanical Design*, 119 :28–35, 1997. 17

[143] N. Tarin, S. Coquillard, S. Hasegawa, L. Bougila, and M. Sato. The stringed haptic workbench : a new haptic workbench solution. *Proc. Eur. Int. Computer Graphics Forum (EUROGRAPHICS'03)*, 22, p. 583-9 2003. 44

[144] T. Tim, B. Rafael, M. S. Ruben, and J. K. Klaas. Rule-based layout solving and its application to procedural interior generation. In *Proceedings of the CASA workshop on 3D advanced media in gaming and simulation (3AMIGAS)*, pages 212–227, 2009. 1, 14, 16, 119

[145] H. R. Trainer. http ://www.motomag.com/le-honda-riding-trainer-4600.html. 62

[146] B. Witmer and M. Singer. Measuring presence in virtual environments : A presence questionnaire. *Presence : Teleoperators & Virtual Environments*, 7(3) :225–40, 1998. 40

[147] K. Xu, J. Stewart, and E. Fiume. Constraint-based automatic placement for scene composition. In *Graphics Interface Proceedings, University of Calgary*, 2002. 13, 16, 81, 89, 103

[148] G. M. Fadel Y. Miao and V. B. Gantovnik. Vehicle configuration design with a packing genetic algorithm. *International Journal of Heavy Vehicle Systems*, 15 :433–448, 2008. 1, 8, 17

[149] S. Yin. A computational framework for automated product layout synthesis based on extended pattern search algorithm. *Thèse de doctorat, Carnegie Mellon University, Pittsburgh*, 2000. 14

[150] S. Zhai. Human performance in six degree of freedom input control. 1995. 43

l'université nantes angers le mans
PÔLE DE RECHERCHE ET D'ENSEIGNEMENT SUPÉRIEUR

université angers

Thèse de Doctorat

Marouene KEFI

Assistance à l'agencement d'environnements virtuels: apport de la programmation par contraintes

Résumé

La diversité des problèmes d'agencement a donné naissance à une multitude de travaux et d'applications liés à des domaines différents tels que la conception mécanique, l'urbanisme, l'aménagement d'intérieur, etc.. La résolution des problèmes d'agencement est généralement basée sur le savoir faire et l'expérience des concepteurs et repose sur un processus coûteux et long qui ne répond pas toujours à l'ensemble des contraintes du problème. En outre, dans la plupart des cas, l'agencement est effectué de manière manuelle à l'aide de plans ou des maquettes éventuellement à l'échelle. Dans les cas où des maquettes numériques sont utilisées, peu voir aucune assistance n'est proposée pour aider les concepteurs dans la prise de décision et la réalisation des tâches. Ce travail de thèse repose sur une démarche innovante impliquant une approche interactive générique basée sur l'utilisation conjointe d'outils de formulation et de résolution issus de la programmation par contraintes, et de techniques de réalité virtuelle (visualisation, immersion, et interaction 3D). L'approche proposée offre différents niveaux d'assistance et supporte la prise compte de diverses contraintes. Elle permet également de définir et de résoudre un problème d'agencement complexe de manière automatique. Les assistances offertes permettent de guider le concepteur vers des solutions viables et de prendre en compte ses préférences. Notre approche a été validée à travers différentes expérimentations impliquant la réalisation de tâches d'agencement en environnements virtuels. L'objectif était d'analyser la pertinence de l'approche proposée à travers l'étude de la performance humaine, et d'illustrer les possibilités d'interaction entre l'utilisateur et le système développé. Notre approche est actuellement étendue à différents cas applicatifs impliquant une problématique d'agencement.

Mots clés

Agencement d'espaces, aide à la décision, interactivité, programmation par contraintes, réalité virtuelle.

Abstract

The diversity of layout problems gave rise to a several applications and related works in different areas such as mechanical design, urban planning, interior design, etc.. Solving layout problems is generally based on the designers expertise and involves a costly and tedious process that do not always satisfies the all problem's constraints. In most cases, the layout is manually carried out using maps or a mock-ups. In cases where numerical mock-ups are used, no assistance is provided to help users in decision making and during the layout tasks. This manuscript proposes an interactive and generic approach based on the use of formulation and resolution tools from constraint programming, and virtual reality techniques (visualization, immersion and 3D interaction). The proposed approach provides different levels of assistance and supports several kinds of constraints. It also allows to define and automatically solve complex layouts. The various provided assistances can guide the user to possible solutions and take into account his/her preferences. Our approach has been validated through several experiments involving various layout tasks in virtual environments. The objectives were to analyze the relevance of the proposed approach through the study of human performance, and to illustrate the possibilities of interaction between the user and the developed system.

Key Words

Spaces layout, decision making, 3D interaction, constraint programming, virtual reality.