Supervision d'informations dynamiques et distribuées à l'aide de mondes 3D interactifs

Application à la gestion de réseaux

Mémoire présenté à la section de système de communication de l'

ÉCOLE POLYTECHNIQUE DE LAUSANNE

pour l'obtention du grade de docteur ès science

par

Pierre Abel

Ingénieur de l'école nationale Supérieure des Télécommunications, Paris, France

Jury :

Prof. D. Thalmann, directeur de thèse
Dr P. Gros, rapporteur
Maître de conférence E. Lecolinet, rapporteur
Dr J.P. Paris, rapporteur
Prof C. Wellekens, rapporteur

Table des matières

1 Introduction

Une des révolutions du XXème siècle est l'utilisation massive de l'informatique. Une définition exhaustive de l'informatique en quelques lignes est impossible, mais on peut dire grossièrement que l'informatique permet de créer, stocker, et traiter des informations sous forme numérique, ainsi que de les restituer par la suite. Les informations sous forme numérique ont de nombreux avantages : on peut les traiter automatiquement, les dupliquer sans perte, les faire voyager très vite, leur stockage nécessite très peu d'espace, etc. Ces avantages sont tels qu'actuellement le support numérique est la solution privilégiée pour de nombreux types d'informations.

Afin de partager des informations numériques stockées physiquement à différents endroits, la technologie des réseaux a été mise en place. De telle sorte qu'aujourd'hui, lorsqu'on s'assoit devant un ordinateur, c'est, par le biais d'Internet, devant des centaines de milliers d'ordinateurs que l'on est virtuellement assis, et donc devant d'autant plus d'informations. On parle alors d'informations *distribuées*.

La nature de ces informations est souvent *dynamique,* c'est à dire qu'elles évoluent au cours du temps. Des informations peuvent se créer (un nouveau document est mis sur le réseau, un nouvel utilisateur se loge sur le réseau), se modifier (mise à jour d'un document, l'utilisateur change d'activité) ou disparaître (destruction d'un document, un utilisateur se déloge).

L'utilisateur a donc accès à des montagnes d'informations distribuées et dynamiques. Il n'est bien sûr pas intéressé par toutes les informations disponibles mais seulement par certaines d'entre elles, qui sont en relation avec la tâche qu'il mène. Qu'il s'agisse, par exemple, d'analyser, de superviser ou de chercher parmi des informations, l'utilisateur a besoin d'*outils* lui permettant de mener à bien sa tâche.

Parmi ces outils, il existe une catégorie portant le nom de « visualisation d'informations », et qui s'attache à exploiter les capacités graphiques et interactives des ordinateurs afin de communiquer visuellement des informations à l'utilisateur.

1 Visualisation d'informations

La visualisation d'informations est définie par [Card99] en ces termes :

« L'utilisation de l'ordinateur pour créer des représentations visuelles interactives de données abstraites afin de développer le processus de connaissance ».

L'utilisation de l'informatique est donc la base de la visualisation d'informations selon cette définition, mais l'utilisation de représentations pour visualiser des informations est bien antérieure à la naissance de l'informatique, comme le souligne [Tufte83] en citant [Playfair1786], un des inventeurs de la représentation graphique de statistiques comme les graphes à barre.

La visualisation d'informations partage de nombreux points avec la visualisation scientifique. Elles exploitent toutes deux les capacités de perception visuelle propres à l'être humain, mais la visualisation scientifique s'occupe des informations ayant une représentation graphique inhérente. Par exemple, des informations sur le climat de la Terre se visualiseront logiquement sur une représentation graphique de la Terre.

En visualisation d'informations, le choix de la représentation graphique est plus délicat car les informations sont abstraites, par exemple les processus d'un ordinateur. Il n'existe donc pas de support privilégié pour leur représentation comme c'est le cas en visualisation scientifique. Pour cette raison, la sémiologie - l'étude des symboles et des signes -, les théories de la perception ainsi que les sciences cognitives peuvent aider lors du choix de la représentation, et certains principes de bases ont déjà été dégagés [Card99, p24-26].

Une attention particulière a été accordée à l'utilisation de métaphores visuelles car elles permettent, comme dans le langage, d'expérimenter un concept grâce à un autre. Ainsi, en choisissant une représentation graphique connue de l'utilisateur (par exemple un bureau, une ville ou un paysage), des informations n'ayant pas de réalité physique seront visualisées sous une forme que l'utilisateur pourra interpréter facilement.

L'informatique a pour avantage de pouvoir automatiser la création des visualisations et de les rendre dynamiques par le biais d'animations, mais également de pouvoir les rendre *interactives* par l'intermédiaire d'interfaces homme-machine. De récents travaux ont montré qu'un outil de visualisation d'informations n'était pas simplement constitué de bonnes représentations mais également de bonnes interactions accompagnant ces représentations [Ahlberg94] [Card99, p. 231-235]. Les opérations autorisées dans ces interfaces, comme le filtrage d'informations, permettent à l'utilisateur de changer la représentation de manière à faciliter la tâche qu'il mène.

Une partie des recherches effectuées dans le domaine de la visualisation d'informations se déroule en amont de la visualisation elle-même. Comme illustré à la Figure 1, il existe tout un travail de préparation des informations, travail consistant à les organiser et les structurer en vue d'une certaine tâche à effectuer. Il existe aussi un travail de mappage entre les informations structurées et les

structures visuelles utilisées, ainsi que des transformations de la visualisation permettant d'exploiter les structures visuelles et de créer la visualisation finale.

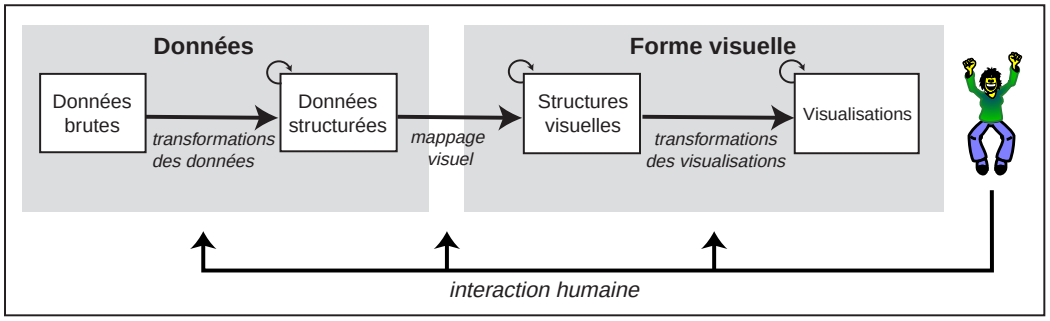


Figure 1 Modèle de référence pour la visualisation (adapté de [Card99])

Le domaine de la visualisation d'information est en pleine expansion et des ouvrages récents résumant ce domaine sont actuellement disponibles. Le premier ouvrage à citer est [Card99], qui reprend des articles importants de l'histoire de la visualisation d'informations tout en présentant clairement une vue d'ensemble du domaine. D'autres ouvrages intéressants sont ceux de [Ware00], [Chen99] et [Spence00]

Dans ces ouvrages, on peut trouver de nombreux exemples d'outils de visualisation qui ont inspiré le travail entrepris dans le cadre du projet CyberNet, au sein duquel s'est déroulé cette thèse.

2 Le projet CyberNet

Le projet CyberNet, financé par France Telecom et l'Institut Eurecom, a été initié devant le constat du manque d'efficacité des outils mis à la disposition des administrateurs de réseaux. En effet, avec le développement exponentiel de l'informatique, ces personnes doivent superviser et analyser de gigantesque quantités d'informations, et l'utilisation des outils traditionnels d'administration de réseaux se révèle parfois peu efficace.

Le but du projet CyberNet est de fournir de nouveaux outils permettant aux administrateurs de réseaux de mieux analyser et superviser les informations dynamiques et distribuées issues de leurs réseaux. Pour cela, il a été choisi de visualiser ces informations à l'aide de représentations tridimensionnelles (3D).

Grâce aux progrès accomplis par l'informatique, notamment en matière de puissance de calcul, il est maintenant possible d'obtenir des représentations 3D en temps réel sur un ordinateur de bureau. La richesse d'une représentation peut être très complexe et l'on a couramment recours à l'expression *mondes 3D* pour décrire de telles représentations, en particulier lorsqu'elles utilisent des métaphores de la réalité. Le projet CyberNet a pour but d'étudier les bénéfices que ces types de visualisation sont susceptibles d'apporter, ainsi que tous les types d'interactions

nécessaires à l'exploitation de telles visualisations afin de créer des *mondes 3D interactifs*.

Parallèlement à ces expériences de visualisations, le projet CyberNet vise aussi à concevoir le réseau non pas comme une somme de composants distincts mais plutôt comme un ensemble de *services* (activités des utilisateurs sur le réseau, état de la connectivité, du partage de fichiers,…). Pour cela, il est nécessaire d'utiliser un modèle de données permettant d'organiser et de structurer, selon le type du service, les informations disponibles sur le réseau.

Afin de concrétiser ces différents objectifs, une plate-forme logicielle distribuée a été mise en place afin de collecter les informations distribuées, de les structurer, et enfin de les visualiser sous forme de mondes 3D interactifs. L'un des points clés de cette plate-forme est sa gestion du caractère dynamique des informations collectées : le système devant être capable d'identifier les changements intervenus au sein des informations et, en temps réel, de modifier en conséquence les mondes 3D. Un autre point clé est sa puissance en matière de réutilisation, permettant de créer très rapidement de nouveaux types de service et de visualisation.

Bien que ce projet vise principalement à fournir un outil permettant à un administrateur de réseaux d'exécuter ses tâches de supervision, ses principes de base ne sont aucunement liés à ce seul domaine. Les principes se veulent génériques et applicables à n'importe quel type d'informations dynamiques et distribuées, telles que les données financières.

3 Contenu de la thèse

Le but de cette thèse était d'étudier tous les aspects du projet CyberNet que nous venons de décrire. Pour cela, nous avons travaillé sur trois grands thèmes :

- Un modèle de données permettant de créer des services, concept regroupant et organisant toutes les informations nécessaires à une tâche en vue de les visualiser. Ce modèle devait prendre en compte le caractère dynamique des informations.

- Un modèle permettant la construction de mondes 3D interactifs à partir des informations contenues dans un service, les éléments de ce modèle devant être conçus afin de faciliter cette construction et la définition de nouveaux types de visualisations interactives, en particulier par leur réutilisation.

- Une plate-forme logicielle orientée objet et distribuée permettant de valider et de tester les précédents modèles sur des exemples liés à la gestion de réseaux

Le deuxième thème a été le plus approfondi car il s'est avéré complexe. En travaillant sur ce modèle de construction de mondes 3D interactifs, nous avons

été amené à séparer l'aspect graphique de l'aspect interactif, ce qui a engendré deux sous-modèles servant de fondation au modèle global :

- Un modèle de visualisation permettant de construire des mondes 3D destinés à la visualisation des informations d'un service, et prenant en compte les contraintes liées au caractère dynamique de ces informations.

- Un modèle de navigation dans les mondes 3D permettant à l'utilisateur de naviguer et d'observer facilement ceux-ci sans se perdre, et ayant la particularité d'être adaptable à l'aspect visuel de la représentation utilisée.

Ce mémoire s'efforce donc de détailler ces différents travaux, et de présenter la manière dont ceux-ci s'interfacent et sont utilisés de manière à construire un environnement complet pour la visualisation d'informations dynamiques et distribuées à l'aide de mondes 3D interactifs.

Dans ce document, on trouvera également de nombreux exemples de visualisations réalisées grâce à cette plate-forme, mais on ne trouvera pas (malheureusement) de nouveaux types de visualisation révolutionnaire. Par contre, on pourra voir comment des types de visualisation déjà connus peuvent être appliqués à l'administration de réseaux, et dans certains cas améliorés.

4 Organisation du mémoire

Dans le chapitre 2, nous présentons le modèle de données permettant d'organiser des informations dynamiques, le chapitre 3 est consacré au modèle que nous avons utilisé pour construire l'aspect graphique des mondes 3D, le chapitre 4 s'intéresse aux interactions avec les mondes 3D en proposant notamment un modèle de navigation, le chapitre 5 introduit le modèle de mondes 3D interactifs au sein du projet CyberNet et détaille le mappage des informations sur ces mondes, le chapitre 6 est consacré à l'étude de l'implémentation de la plate-forme logicielle distribuée et le chapitre 7 présente des visualisations expérimentales ayant été réalisées durant cette thèse. Le chapitre 8 présente les conclusions de cette thèse ainsi que ses perpectives. En annexe, on trouvera une liste d'acronymes, un lexique résumant les principaux concepts utilisés dans ce mémoire, ainsi que des figures de référence permettant de résumer certains de ces concepts visuellement.

2 Modèle de données

1 Introduction

Ce chapitre décrit le modèle de données utilisé dans le projet CyberNet, c'est–à-dire les concepts utilisés de façon à structurer les informations brutes sous une forme qui puisse, à terme, être directement exploitée pour être visualisée. Le résultat de l'utilisation de ce modèle de donnée est donc destiné à être mappé sur le modèle de monde 3D interactif que l'on décrira dans la suite de ce mémoire.

Ce modèle de données peut donc être considéré comme le cœur de notre système dans le sens où c'est lui qui va déclencher la collecte d'informations pour ensuite les structurer et enfin engendrer les visualisations.

La structuration des informations, condition indispensable à leur analyse, est donc cruciale et le modèle de données doit offrir suffisamment de souplesse pour permettre de définir les informations nécessaires et de les organiser comme on l'entend. Nous présentons donc un modèle offrant des outils génériques permettant de remplir ces tâches.

Notre modèle est également conçu pour permettre de construire plusieurs structures de données en parallèle de manière à générer plusieurs visualisations simultanées. De plus, un mécanisme de partage des informations est intégré au modèle de façon à pouvoir synchroniser deux visualisations partageant les mêmes informations mais structurées différemment.

Les informations traitées dans ce projet sont distribuées et dynamiques. L'aspect distribué n'est pas pris en compte par le modèle en lui-même, l'implémentation de la plate-forme logicielle (cf. chapitre 6) se chargeant de rassembler localement les informations distribuées. Par contre, le caractère dynamique des informations est intrinsèque au modèle. Ainsi, les structures de données peuvent évoluer automatiquement au cours du temps de manière à décrire l'état actuel des informations que l'on désire visualiser.

Dans un premier temps, nous présentons des modèles de données utilisés dans des outils de visualisation déjà existants. Ensuite, nous présentons le modèle utilisé dans CyberNet pour finir par des exemples illustrant celui-ci.

2 Modèle de données en visualisation d'information

Contrairement à la visualisation scientifique où les données tendent à être physiques (le corps humain, la Terre, les molécules, etc.), en visualisation d'informations les données ont tendance a être abstraites (données financières par exemple). Pour cette raison, il n'est pas possible d'utiliser une représentation géométrique dérivée du type des données comme c'est le cas en visualisation scientifique. C'est pourquoi il est nécessaire d'utiliser un modèle de données abstrait permettant d'organiser n'importe quel type d'informations.

Les articles concernant les outils de visualisation présentent en détails leurs techniques de visualisation mais, malheureusement, exposent plus sommairement le modèle de données sous-jacent à leur système. La plupart du temps, ils présentent uniquement le type de leur modèle de données qui est, dans la grande majorité des cas, le modèle entités-relations. Nous présentons tout d'abord ce modèle, puis nous exposons comment ce modèle est exploiter pour créer un graphe ou un arbre qui doit, à terme, être visualisé.

2.1 Modèle entités-relations

2.1.1 Entités

La plupart des outils de visualisation évolués [He98] [Rish97] [Kazman96] [Mukherjea95a] [Roth97] [Fairchild88] exploitent un modèle d'information basé sur la notion d'entités (appelés aussi nœuds dans certains cas). Une entité est typée et représente un concept, comme par exemple en gestion de réseau un serveur ou un switch [He98] ou encore un processus. Les entités peuvent être classées selon qu'elles représentent un concept réel (une machine) ou abstrait (un processus).

Afin de décrire le concept représenté par une entité, celle-ci possède des attributs permettant de caractériser l'une de ses instances particulières : par exemple, les attributs d'une entité de type ordinateur peuvent comprendre son adresse IP, le modèle d'ordinateur, la charge CPU de la machine ou la mémoire utilisée. Le type d'attribut dépend du domaine d'application et peut aller du simple booléen à la vidéo en passant par le texte. Dans certains systèmes [Rish97] [Kazman96], les attributs sont conservés sous la forme d'une paire (nom, valeur) afin de faciliter la recherche des requêtes entre les entités. Au lieu de typer explicitement l'entité, on peut également définir un attribut de nom « type » dont la valeur fixera le type de l'entité.

2.1.2 Relations

Afin de structurer les entités, les différents systèmes permettent d'obtenir les relations existant entre celles-ci. Par exemple toutes les entités de type ordinateur ont une relation entre elles et toutes les entités de type « ordinateur » du même modèle possèdent une autre relation commune ; de même, toutes les entités ayant un attribut « NomUtilisateur » de même valeur ont une relation entre elles.

Les relations peuvent être classées en fonction de leur type. Deux types de relation sont généralement utilisés : les relations ensemblistes qui mettent toutes les entités sur le même plan, et les relations structurelles qui établissent une hiérarchie entre les entités. Par exemple toutes les entités de type « utilisateur » possèdent une relation ensembliste commune, et une entité de type « ordinateur » possède une relation structurelle commune avec chaque entité représentant les utilisateurs de cet ordinateur, la hiérarchie entre ces entités étant déterminée selon le point de vue qu'on adopte (du point de vue de l'utilisateur c'est l'ordinateur qui est le fils, alors que c'est le contraire du point de vue de l'ordinateur).

Selon les systèmes, ces relations sont fixées [Fairchild88] [Mukherjea95a] ou alors déterminées dynamiquement à la suite d'une ou plusieurs requêtes sur l'ensemble des entités existantes [He98] [Risch97] [Kazman96] [Roth97]. L'utilisation de requêtes permet beaucoup plus de flexibilité au sein du système car le choix des relations est ainsi limité seulement par la puissance des requêtes. Une requête est généralement exprimée sous la forme de critères sur le type et les attributs des entités désirées. Quand les attributs sont définis sous la forme de paires (nom, valeur), il est également possible d'exprimer des requêtes dont les critères portent seulement sur les attributs.

2.2 Construction du graphe à visualiser

Afin de préparer les informations pour la visualisation, les outils de visualisation construisent un graphe (qui peut prendre la forme d'un arbre) permettant de structurer les informations, les nœuds de ce graphe étant les entités et les liens les relations entre ces entités. La construction de ce graphe est différente selon que les relations sont fixées ou déterminées à partir de requêtes.

Dans les systèmes où les relations sont fixées, le graphe à visualiser peut être constitué de toutes les relations existantes spécialement si le volume de ces relations n'est pas trop important [Fairchild88]. [Mukherjea95a] propose de visualiser également le graphe contenant toutes les relations, mais offre également des outils permettant à l'utilisateur d'extraire des arbres de ce graphe en sélectionnant des relations structurelles.

Les outils de visualisation utilisant des requêtes sélectionnent les relations en fonction de la tâche qui leur est assignée afin d'obtenir un graphe contenant les entités et les relations qui les intéressent. Les systèmes se différencient par leur capacité à créer seulement une structure d'arbre [Rish97] [Kazman96] ou bien un graphe pouvant éventuellement prendre l'allure d'un arbre [Roth97] [He98]. Il faut

noter que pour construire une hiérarchie, les systèmes construisant des arbres doivent utiliser des relations structurelles.

Concernant l'utilisation de graphe ou d'arbre, [Mukherjea95b] soutient que l'analyse visuelle de graphes complexes est très difficile car ils génèrent des visualisations qui ne sont pas très efficaces. [Eick96] renforce cette idée en écrivant que la visualisation de graphes complexes peut apporter plus de confusion que d'éclaircissement. [Mukherjea95b] souligne qu'en revanche il existe de nombreux types de visualisation d'arbre efficaces, et qu'il est donc préférable de construire plusieurs hiérarchies extraites d'un graphe, présentant chacune les mêmes informations mais sous un angle différent.

Méthode de construction du graphe

Pour les systèmes à base de requêtes, il est intéressant d'étudier comment le graphe ou l'arbre est construit. [Risch97] s'en remet à l'algorithme intégré à la base de données (BDD) contenant les entités pour construire son arbre. La BDD utilisée par [Risch97], de type Contiguous Connection Model [Wurden97], est particulièrement intéressante car elle calcule en interne toutes les relations possibles entre les entités grâce à un format spécial imposé en entrée ; ceci permet à cette BDD de générer, à partir de critères spécifiés par l'utilisateur, un arbre grâce auquel il est possible de naviguer parmi les entités. Les critères spécifiés par l'utilisateur permettent à la BDD de sélectionner les relations structurelles qui vont être utilisées pour construire la structure de l'arbre.

[Roth97] ne construit pas directement un graphe mais propose plutôt à l'utilisateur d'interagir d'une façon transparente avec la BDD contenant les entités de manière à naviguer parmi celles-ci. A partir d'une entité, l'utilisateur peut sélectionner un ensemble de relations ensemblistes ou structurelles disponibles et ainsi accéder à d'autres entités. L'exploration des entités se fait donc au gré des choix que l'utilisateur se voit proposer.

[Kazman96] et [He98] construisent quant à eux directement l'arbre ou le graphe à visualiser. [Kazman96] demande que l'on spécifie des règles, définies à base de relations structurelles, permettant de construire itérativement un arbre, et pour chaque nouveau domaine d'application, il est nécessaire de spécifier ces règles. [Kazman96] propose donc une solution qui définit un squelette utilisant des relations structurelles pour spécifier comment les informations doivent être organisées.

Comme nous allons le voir, le projet CyberNet utilise également la notion de squelette et fournit des outils puissants, indépendants du domaine d'application, permettant de spécifier comment les informations doivent être organisées. De plus, le modèle de CyberNet intègre des mécanismes permettant de gérer le caractère éventuellement dynamique des informations.

3 Le modèle de données de CyberNet

Le projet CyberNet vise à proposer de nouveaux types de visualisation permettant de faciliter la supervision et l'analyse d'informations dynamiques. Nos études de cas se concentrent sur les réseaux informatiques, mais CyberNet a pour but d'être capable de fonctionner dans d'autres domaines d'applications comme par exemple l'analyse boursière. Le modèle de données se veut donc générique et doit permettre de modéliser et structurer n'importe quel type d'informations dynamiques.

Le but de notre modèle de données est de permettre la définition de squelettes spécifiant la façon dont un ensemble d'informations doit être structuré afin de faciliter la réalisation d'une tâche. Notre modèle est également conçu afin de permettre la construction simultanée de plusieurs structures de données de sorte que le système puisse générer plusieurs visualisations en même temps.

Comme c'est le cas dans la plupart des systèmes de visualisation d'informations, notre modèle de données s'appuie sur des entités et des requêtes ainsi que sur une structure d'arbre. Un concept supplémentaire, utilisant les entités et les requêtes, apporte des mécanismes permettant de construire et tenir à jour la structure organisant les informations. Cet aspect est crucial car il ne faut pas oublier que les informations traitées peuvent être dynamiques ; la structure doit donc pouvoir évoluer en même temps que celles-ci.

Dans un premier temps, nous introduisons le concept de service, pour ensuite exposer comment est réalisée la gestion des informations dans notre modèle. Puis nous présentons les bases de la construction de nos services, pour finir par décrire les squelettes de service.

3.1 Le concept de service

3.1.1 Définition

Le but de notre modèle de données est de pouvoir construire des *services*, concept permettant d'identifier et d'organiser toutes les informations nécessaires à une tâche, et dont le but est de préparer les informations en vue de leurs visualisations. En gestion de réseau, [Lewis99] définit un service comme « an abstraction over and above the entreprise network ». Par exemple, un administrateur réseau peut avoir accès à des visualisations permettant de superviser des services que son réseau lui offre : routage, courrier électronique, application client/serveur tel que NFS, ou plus typiquement topologie du réseau. La définition de [Lewis99] permet de souligner l'abstraction que réalisent les services : ceux-ci permettent de considérer à part certaines informations, en portant spécialement l'attention sur elles et en négligeant les autres, de manière à faciliter la réalisation d'une tâche.

3.1.2 Identifier les informations

Un service doit tout d'abord identifier les informations pertinentes à l'analyse de la tâche qui lui est assignée. Par exemple, un service de supervision d'un parc d'ordinateurs doit pouvoir regrouper toutes les informations relatives aux ordinateurs de ce parc, mais aussi celles concernant les utilisateurs de ces ordinateurs ainsi que leurs processus, ces informations pouvant être utiles pour analyser l'activité des ordinateurs.

3.1.3 Structurer les informations

Le travail d'un service n'est pas seulement d'identifier des informations, mais aussi de les structurer de manière à faciliter leur analyse dans l'optique de la tâche qui lui est assignée. Dans l'exemple précédent, si les informations concernant les ordinateurs, leurs utilisateurs et leurs processus ne sont pas structurées, la simple tâche consistant à déterminer quels sont les utilisateurs d'un ordinateur spécifique risque d'être fastidieuse. La structuration des informations permet de faciliter la tâche à effectuer : en structurant les utilisateurs par ordinateur, il sera pratiquement immédiat de déterminer quels sont les utilisateurs d'un ordinateur spécifique.

3.2 Gestion des informations

Nous avons présenté le concept de service, mais nous n'avons pas abordé la gestion de l'information qui doit être réalisée en amont afin que ces informations soient disponibles. Cette gestion nécessite de collecter les informations brutes pour ensuite les structurer sous forme d'entités, mais aussi de prévoir des mécanismes supportant le caractère dynamique des informations ainsi que le partage des entités entre plusieurs services.

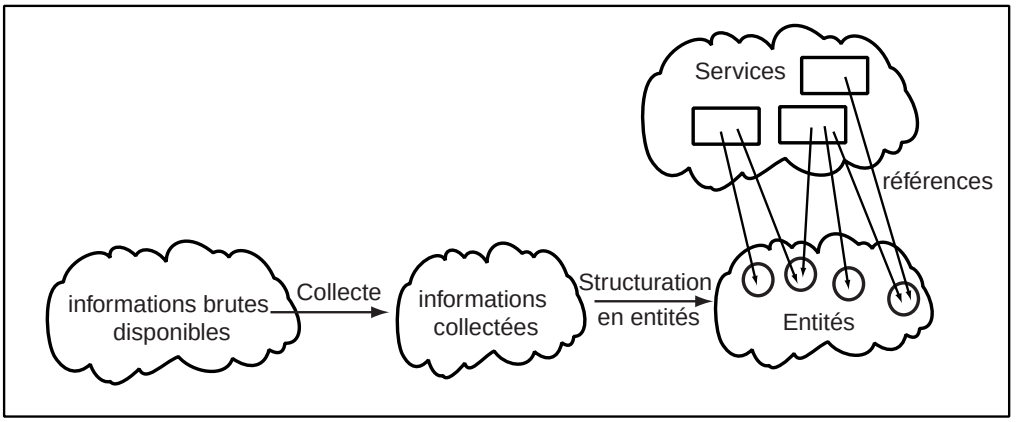


Figure 2 *Gestion de l'information*

3.2.1 Collecte des informations

La première étape est de collecter les informations brutes qui sont nécessaires au service. Selon la nature des informations, différents mécanismes ou technologies doivent être utilisés, comme par exemple des commandes système pour obtenir les informations sur l'état d'un ordinateur ou alors un protocole spécifique, comme SNMP en gestion de réseaux. Dans ce dernier exemple, les informations sont distribuées et doivent être collectées sur les différents éléments du réseau. Notre modèle ne prend pas en compte cet aspect distribué, l'implémentation de notre système se chargeant de centraliser toutes les informations (cf. chapitre 6). Une fois ces informations disponibles, il est possible de construire des entités.

3.2.2 Structuration des informations sous forme d'entités

Nous avons vu dans la partie précédente qu'il était pratique de réunir des informations sous la forme d'entités représentant le premier niveau de structuration des informations. Une entité représente un concept qui définit son type et contient une liste d'attribut qui représente les informations et permet de caractériser une instance particulière. Le choix des entités est entièrement dépendant du service que l'on veut créer. Par exemple, dans notre service de supervision d'ordinateurs il existe des entités de type « ordinateur » dont les attributs sont le nom de l'ordinateur, sa charge CPU, sa mémoire, son adresse IP, etc.

Chaque attribut est une paire (nom de l'attribut, valeur) dans laquelle le nom de l'attribut est commun à toutes les entités ayant ce type d'attribut. Par exemple, une entité de type « utilisateur d'ordinateur » et une entité de type « ordinateur » auront toutes les deux un attribut ayant pour nom « nom de l'ordinateur ». Nous verrons par la suite que cette spécification des attributs permet plus de souplesse dans la définition des services.

Dans notre système, un service manipule les informations exclusivement sous la forme d'entités. Dans ce sens, on peut donc dire que l'entité est l'unité d'information des services.

3.2.3 Caractère dynamique des entités

Le caractère dynamique des informations entraîne trois conséquences sur les entités :

- Premièrement, de nouvelles entités peuvent être créées (un nouvel utilisateur se loge sur un ordinateur).
- Deuxièmement, des entités peuvent disparaître (un utilisateur se déloge d'un ordinateur).
- Troisièmement, certains attributs de l'entité peuvent évoluer (la consommation de mémoire d'un utilisateur augmente).

Comme nous allons le voir, ce caractère dynamique des entités entraîne des conséquences sur le reste des éléments du modèle de données, par exemple le fait que les entités impliquées dans un service évoluent au cours du temps.

3.2.4 Partage des entités

Une possibilité intéressante offerte par certains outils de visualisation est de pouvoir générer plusieurs représentations des mêmes entités mais avec des organisations différentes. Nous avons vu un exemple de supervision d'ordinateurs et de leurs utilisateurs qui permet de faciliter la tâche consistant à analyser un ordinateur et ses utilisateurs. Mais ce service n'est pas très pratique si on veut analyser l'activité d'un utilisateur sur l'ensemble des ordinateurs car il faut analyser chaque ordinateur pour voir si l'utilisateur en question est logé dessus. Pour faciliter cette tâche, il faut définir un autre service qui va structurer les entités non pas par ordinateur, mais par utilisateur. Dans ces deux services, ce sont les mêmes informations (les entités utilisateurs) qui sont utilisées, mais elle sont structurées de manière différente.

Afin de maintenir une cohérence entre deux services différents mais contenant les mêmes entités, celles-ci doivent être partagées et non dupliquées. C'est pourquoi, dans notre modèle, les entités n'appartiennent pas à un service particulier et peuvent être partagées entre plusieurs services. Du point de vue de l'implémentation, plusieurs services peuvent donc posséder des références sur les mêmes entités. Nous verrons également plus tard dans ce mémoire que ce mécanisme permet également de faciliter la synchronisation de plusieurs visualisations de services partageant les mêmes entités.

Il faut également noter que les mêmes entités peuvent être utilisées plusieurs fois à l'intérieur d'un même service. Par exemple, dans un service permettant de visualiser l'activité des utilisateurs présents sur le réseau, il peut être utile de visualiser une entité de type « ordinateur » pour chacun des utilisateurs logés sur cet ordinateur. D'autres part, nous allons voir que les entités peuvent être exploitées de deux manières par un service, et qu'il est courant qu'un service ait besoin d'une entité plusieurs fois afin de l'exploiter de deux manières différentes.

3.3 Construction de services

3.3.1 Obtenir les entités : les requêtes

Une fois identifiées les différentes entités nécessaires à un service, il s'agit de les obtenir. Pour cela, on utilise des requêtes qui permettent d'obtenir la ou les références des entités qui satisfont certains critères. Les possibilités offertes par les requêtes sont un facteur crucial dans la souplesse de notre modèle de données. En effet, les services s'appuient sur les requêtes pour obtenir leurs entités et sont donc dépendants des possibilités offertes par celles-ci.

Les requêtes sont utilisées à deux fins différentes. Dans un premier cas, elles permettent d'obtenir une entité particulière (par exemple l'entité représentant un utilisateur particulier sur un ordinateur spécifique), et dans un second cas, elles permettent d'obtenir un ensemble d'entités ayant une relation entre elles (par exemple toutes les entités représentant un utilisateur sur un ordinateur spécifique).

Une requête est définie par une expression booléenne d'un ensemble de critères (par exemple : critère1 ET (critère2 OU critère3)). Chaque critère est une expression logique, arithmétique ou régulière sur le type de l'entité ou sur un attribut. Cette spécification permet, par exemple, d'obtenir :

- L'entité de type utilisateur ayant un attribut « nom » de valeur « toto » et un attribut « ordinateur » de valeur « www ».
- L'ensemble des entités de type utilisateur ayant un attribut « ordinateur » de valeur « www ».
- Toutes les entités de type « processus » ayant un attribut « cpu » de valeur supérieur à 50.

Le caractère dynamique des entités (création, disparition, mise à jour des attributs) a pour conséquence de faire évoluer dans le temps les entités référencées par une requête. On parle alors du caractère dynamique d'une requête.

3.3.2 Exploitation des entités

Les services utilisent les requêtes pour obtenir des entités qui sont ensuite exploitées soit pour engendrer d'autres requêtes et ainsi, comme nous allons le voir, structurer les informations, soit qui sont simplement conservées afin d'être visualisées. Ainsi, le service sépare les informations destinées à construire la structure du service et celles destinées à être visualisées.

3.3.2.1 Structuration par requêtes cascadées

La première utilisation des entités par un service est de permettre d'engendrer une ou plusieurs autres requêtes. Par exemple, dans notre service de supervision d'ordinateurs, une requête est émise pour obtenir la liste des noms des ordinateurs, et pour chaque nom le service va créer une requête pour obtenir l'entité de type « ordinateur » correspondant à ce nom. D'autres requêtes vont ensuite être émise de manière à obtenir les entités représentant les utilisateurs logés sur ces ordinateurs, puis encore d'autres pour obtenir des informations sur les processus de ces utilisateurs (cf. Figure 3).

Une information peut donc engendrer une cascade de requêtes, et c'est cette cascade qui va définir la structuration des informations. En effet, cette cascade de requêtes correspond à la structuration nécessaire des informations pour remplir la tâche assignée à ce service. Le résultat de cette cascade produit une structure d'arbre, une requête pouvant en entraîner plusieurs et donc créer plusieurs branches. L'arbre résultant de cette cascade de requête constitue la structure du service.

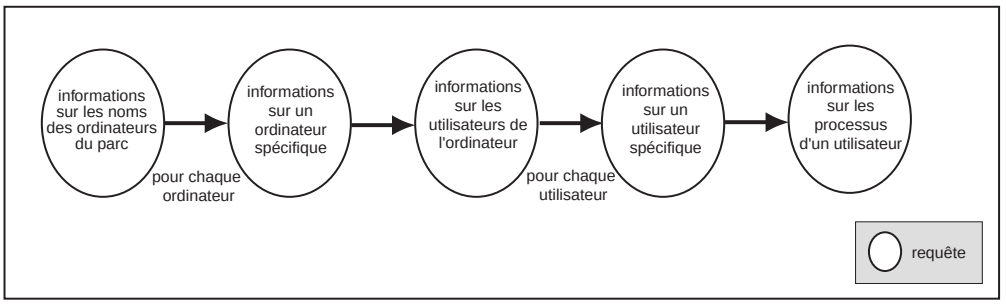


Figure 3 *Cascade des requêtes nécessaires pour un service de supervision d'ordinateurs*

3.3.2.2 Entités destinées à être visualisées

Les requêtes émises par un service lui permettent d'avoir accès à des informations sous la forme d'entités. L'organisateur exploite ces entités de deux façons. La première exploitation est celle dont on vient de parler : les informations peuvent permettre d'engendrer de nouvelles requêtes et ainsi de construire l'arbre du service et donc structurer les informations. Mais les entités nécessaires à la structuration ne sont pas forcement celles qui sont utiles à la visualisation.

En effet, il ne faut pas perdre de vue qu'un service est destiné à être visualisé. Par conséquent, il est important de pouvoir spécifier quelles vont être les entités visualisées de manière à les différencier de celles permettant uniquement la construction de la structure du service. Par exemple, dans la cascade de requêtes illustrée à la Figure 3, la liste des noms des ordinateurs obtenue par la première requête est indispensable pour créer la structure du service, mais il n'est pas utile de visualiser cette liste de noms. En revanche, les entités de type « ordinateur » obtenues par les requêtes suivantes sont, elles, utiles à la visualisation et sont explicitement déclarées par le service comme entités à visualiser.

3.3.3 Comportement vis-à-vis du caractère dynamique des informations

Un aspect particulier des informations traitées dans ce projet est leur nature dynamique. Un service doit donc prendre en compte cette caractéristique de manière à refléter l'état actuel des informations. Ainsi, il doit constamment vérifier si de nouvelles entités sont impliquées dans une de ses requêtes ou si, au contraire, certaines ont disparu, et réagir en conséquence.

Dans le cas où une nouvelle entité apparaîtrait, il doit vérifier si cette entité ne doit pas engendrer de nouvelles requêtes nécessaires à la compréhension du service. Par exemple, si un nouvel utilisateur se loge sur un ordinateur,il faudra créer de nouvelles requêtes pour obtenir des informations sur celui-ci et ses processus devront être créées.

Au contraire, si une entité ayant engendré des requêtes disparaît, il faut alors que toutes les requêtes engendrées par cette entité soient supprimées, car elles n'ont plus de raison d'être au niveau de la logique du service. Par exemple, si un

utilisateur se déloge d'un ordinateur, les requêtes s'occupant d'obtenir des informations sur celui-ci et ses processus n'ont plus de raison d'être et doivent donc être supprimées.

3.4 Squelette de service

Le but de notre modèle est de pouvoir définir un squelette de service qui va permettre de générer automatiquement une instance de ce service en fonction des informations existantes. Nous venons de décrire les principes de construction d'un service, et maintenant nous allons présenter comment ces principes (cascades de requêtes, entités destinées à être visualisées, etc.) sont appliqués pour la définition d'un squelette, laquelle s'appuie sur le concept d'organisateur.

3.4.1 Les organisateurs

3.4.1.1 Construction de la structure du service

La construction de la structure d'un service se résume à définir quelles sont les requêtes nécessaires et comment elles s'enchaînent. Cet enchaînement est la base de la structuration des informations du service dont le résultat est un arbre de nœuds contenant chacun au moins une requête. La Figure 3 illustre clairement cet enchaînement, mais elle ne montre pas comment il est réalisé. Afin de décrire cet enchaînement, le modèle de données offre un outil permettant de créer des requêtes, mais aussi de réagir en fonction des informations obtenues de manière à créer d'autres requêtes.

Cet outil, que nous avons appelé *organisateur,* a donc la possibilité de créer des requêtes et intègre un comportement permettant de décrire comment il doit exploiter les informations. Il est le seul type de nœud de l'arbre décrivant le service, et il a la possibilité de créer d'autres organisateurs qui seront vus comme ses fils au niveau de la structuration. Dans notre exemple de service de supervision d'ordinateurs, c'est un organisateur qui crée la requête permettant d'obtenir les noms des ordinateurs, et qui, pour chaque nom, crée un nouvel organisateur qui va s'occuper de d'obtenir les informations concernant un ordinateur spécifique. Ces derniers organisateurs créent eux aussi d'autres organisateurs et ainsi de suite comme illustré à la Figure 4. Le service est donc construit par une cascade de requêtes orchestrée par les organisateurs.

Organisateur fils par défaut

Une information peut entraîner la création d'un organisateur qui mène à la création d'une requête. Mais dans certains cas, une information peut nécessiter la création de plus d'une requête. Par exemple, chaque utilisateur existant sur un ordinateur entraîne la création de deux requêtes : une permettant d'obtenir des informations sur l'utilisateur en question, et une autre pour obtenir des informations sur ses processus (cf. Figure 4). De manière à ce que ces

informations soient structurées, il est préférable que les requêtes soient effectuées dans deux organisateurs différents. Le premier organisateur doit donc créer d'office un autre organisateur qui va se charger de réaliser la deuxième requête. On parle alors de création d'organisateur fils par défaut.

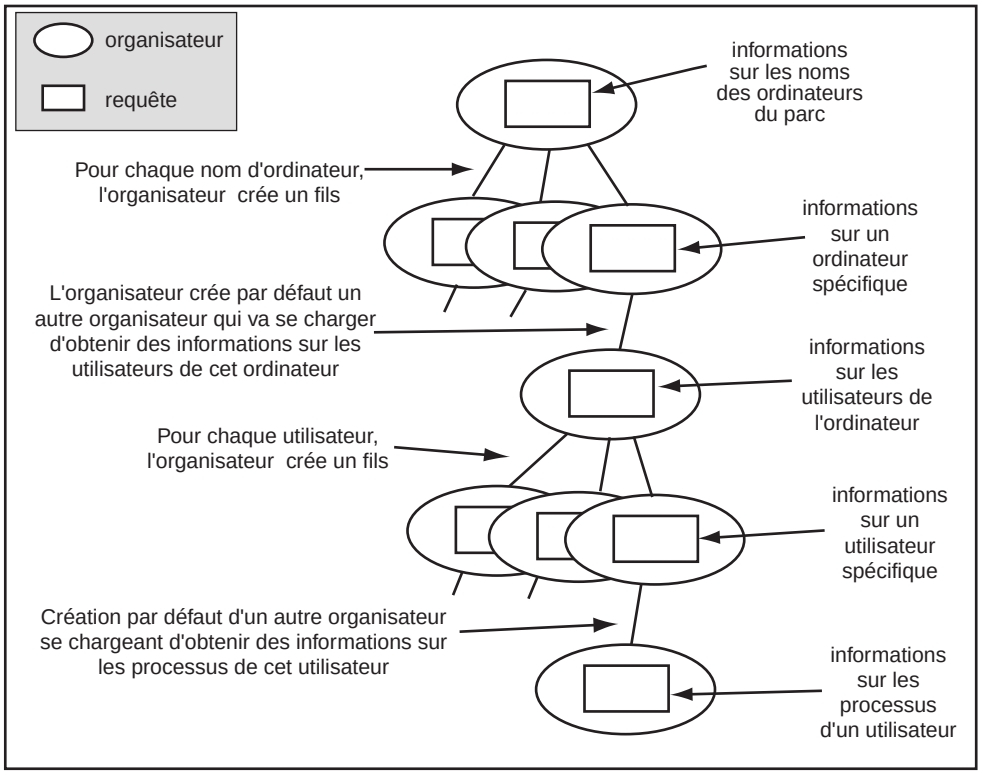


Figure 4 *Squelette d'un service de supervision d'ordinateurs*

3.4.1.2 Spécification des entités à visualiser

Les organisateurs permettent également de créer des requêtes dont les entités ne sont pas destinées à la construction de la structure du service, mais à être visualisées, ce qui permet de définir précisément quelles sont les entités utiles pour la visualisation sans tenir compte des entités nécessaire à la structuration du service.

3.4.1.3 Héritage de paramètres

Pour pouvoir créer une requête, il est nécessaire de pouvoir spécifier ses paramètres. Par exemple la requête permettant d'obtenir des informations sur un utilisateur d'un ordinateur nécessite de connaître le nom de l'utilisateur ainsi que celui de l'ordinateur. Les organisateurs ont donc besoin de connaître les paramètres des requêtes qu'ils doivent créer.

Ces paramètres leurs sont fournis par leur père, qui a lui-même hérité de certains paramètres de son propre père. On parle alors d'héritage de paramètres. Dans l'exemple de la Figure 4, l'organisateur se chargeant d'obtenir des

informations sur un ordinateur a reçu le nom de celui-ci en paramètre, et quand il crée un fils qui doit obtenir des informations sur les utilisateurs il lui donne en paramètre ce même nom, puis quand ce dernier organisateur crée un fils, il lui donne en paramètre le nom de l'ordinateur et le nom de l'utilisateur afin que celui-ci puisse obtenir des informations sur cet utilisateur.

3.4.2 Comportement d'un organisateur

Le comportement d'un organisateur peut être résumé par les actions qu'il peut effectuer, les événements qu'il peut recevoir au cours du temps et la manière dont il réagit à ces événements.

Un organisateur peut effectuer 2 types d'actions :

- Créer et s'ajouter des fils organisateurs, ainsi que les détruire par la suite
- Créer des requêtes dont les entités sont soit destinées à engendrer des organisateurs fils soit destinée à être visualisées

Un organisateur réagit aux événements qu'il reçoit ; ces derniers peuvent appartenir à l'un des 4 types d'événement présentés dans le Tableau 1.

Evénement	**Résultat de l'événement**
Création	Lorsqu'un organisateur est créé, il peut créer des requêtes et des organisateurs qu'il s'ajoute comme fils.
Une des requêtes de l'organisateur possède une nouvelle entité	• Soit l'entité engendre un nouvel organisateur fils, qui est alors lié à cette entité, • Soit l'entité est destinée afin être visualisée, et elle est uniquement conservée.
Une entité appartenant à une des requêtes de l'organisateur meurt	• L'entité est supprimée de l'organisateur, • Si l'entité avait engendré un organisateur fils, celui-ci est détruit (et donc retiré de l'arbre) ainsi que ses descendants.
Destruction	Toutes les requêtes et tous les fils organisateurs sont détruits.

Tableau 1 *Résultat des événements pouvant être reçus par un organisateur*

4 Exemple de services

4.1 Service de supervision d'un système de fichiers

Ce service nous montre comment notre modèle de données peut être utilisé pour modéliser les informations contenues dans un système hiérarchique de fichiers. Cet exemple illustre simplement les propriétés de notre modèle de données.

La tâche de ce service est de préparer les informations en vue de la création d'une visualisation qui puisse offrir une vue globale d'un système de fichiers, mais aussi l'analyse en détail de certaines branche de la hiérarchie. Pour cette tâche, le service doit donc rendre compte de l'organisation hiérarchique des différents répertoires existants, ainsi que des fichiers contenus dans ces répertoires. La structuration des informations reproduit donc celle qui existe dans le système de fichiers. Afin de permettre une analyse en détails des éléments du système de fichiers (fichiers et répertoires), nous allons voir que les entités créées pour les représenter contiennent de nombreuses informations.

4.1.1 Entités

Les entités nécessaires à ce service sont :

- L'entité de type *répertoire,* dont les attributs sont son nom, son chemin d'accès (par exemple /usr/local/bin/), le nombre de fichiers et de répertoire, qu'il contient, la somme des tailles des fichiers qu'il contient, le type de fichiers prépondérants (archive, postscript, image, etc.), son propriétaire, les droits associés (au sens Unix), sa date de dernière modification et sa date de création.

- L'entité de type *fichier,* dont les attributs sont son nom, son chemin d'accès (par exemple /usr/local/bin/), sa taille, son type, son propriétaire, les droits associés (au sens Unix), sa date de dernière modification et sa date de création.

4.1.2 Requêtes

Trois types de requêtes sont utilisés par les organisateurs décrits dans la partie suivante.

- La requête « répertoire spécifique », qui permet d'obtenir une entité de type « répertoire » dont on possède le nom et le chemin.
- La requête « répertoires », qui permet d'obtenir toutes les entités de type « répertoire » qui existent dans un répertoire dont on connaît le nom et le chemin.

- La requête « fichiers », qui permet d'obtenir toutes les entités de type « fichier » qui existent dans un répertoire dont on connaît le nom et le chemin

Ces requêtes sont résumées dans le Tableau 2.

Requête	**Critères de la requête**	
	Type d'entité désirée	**Attribut(s) de l'entité sur lequel il existe un critère**
Répertoire spécifique	Répertoire	Nom & Chemin
Répertoires	Répertoire	Chemin
Fichiers	Fichiers	Chemin

Tableau 2 *Requêtes utilisées dans un service de supervision de système de fichiers*

Organi-sateur	**Arguments**	**Créations par défaut**	**Utilisation des entités d'une requête**
A	Nom et chemin d'un répertoire	Requête répertoire spécifique	L'entité de type répertoire est destinée à être visualisée
		Organisateur B	
B	Nom et chemin d'un répertoire	Requête Répertoires	Pour chaque entité répertoire, on crée un organisateur A associé
		Requête Fichiers	Les entités de type fichier sont destinées à être visualisées

Tableau 3 *Organisateurs utilisés dans un service de supervision d'ordinateurs*

4.1.3 Organisateurs

Deux types d'organisateurs, décrits dans le Tableau 3, sont nécessaires à la construction du service :

- L'organisateur de type A nécessite un nom de répertoire ainsi que son chemin comme arguments. Il utilise la requête de type « répertoire spécifique » afin d'obtenir une référence à une entité « répertoire » qui doit être visualisée, et créer, également par défaut, un organisateur de type B qui va s'occuper d'obtenir des informations sur les répertoires et fichiers existant dans l'entité répertoire obtenue par la requête.
- L'organisateur de type B nécessite un nom de répertoire ainsi que son chemin comme arguments. Il utilise la requête de type «répertoires » afin d'obtenir les références à des entités «répertoires » contenues dans le

répertoire donné en argument, ces entités servant à créer des organisateurs de type A. D'autre part, la requête de type «fichiers » permet d'obtenir les références des entités «fichier » contenues dans le répertoire donné en argument, ces dernières étant destinées à être visualisées.

La Figure 5 illustre visuellement le service de supervision de fichiers

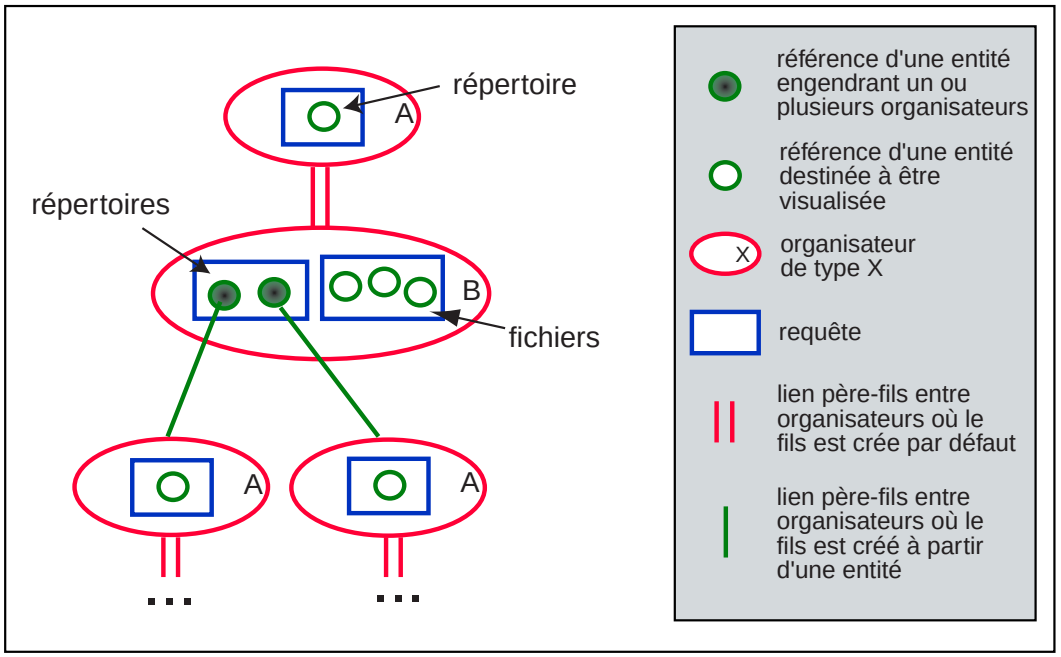


Figure 5 *Service de supervision d'un système de fichiers*

4.1.4 Conclusion

Cet exemple a présenté un exemple trivial de notre modèle données. En particulier, on peut remarquer comment on peut réutiliser des organisateurs et le caractère récursif du squelette dû au type d'information traitée. Cet exemple illustre également comment un service peut utiliser les mêmes entités dans différents organisateurs soit pour engendrer une nouvelle branche de l'arbre (les entités « répertoire » dans l'organisateur B), soit pour les visualiser (entité « répertoire » dans l'organisateur A).

4.2 Services de supervision d'un parc d'ordinateurs, de leurs utilisateurs et de leurs processus

Une tâche courante d'un administrateur système est de superviser les ordinateurs qu'il a dans son réseau, ainsi que les utilisateurs logés sur ceux-ci et les processus qui s'y exécutent. Nous présentons deux services permettant d'organiser ces informations selon différentes stratégies. La tâche du premier

service est de proposer une organisation des informations qui permette d'analyser rapidement celles-ci par ordinateur, tandis que la tâche du deuxième service est d' offrir la possibilité de superviser les mêmes informations mais organisées par utilisateur présent sur le réseau. Ces deux services sont complémentaires et partagent un certain nombre d'entités que nous présentons tout d'abord.

4.2.1 Entités

Les entités nécessaires à la construction de ces services sont :

- L'entité de type *ordinateur*, dont les attributs sont le nom de l'ordinateur, son adresse IP, sa charge CPU et l'utilisation de sa mémoire
- L'entité de type *utilisateur,* qui représente un utilisateur sur une machine spécifique. Ses attributs sont le nom de l'utilisateur, le nom de la machine sur lequel cet utilisateur est logé, le pourcentage de CPU qu'il utilise et la mémoire qu'il utilise.
- L'entité de type *processus,* qui représente un processus s'exécutant sur un ordinateur. Ses attributs sont le nom de la commande qui l'a créé, le nom de son propriétaire, le nom de la machine où il s'exécute, le pourcentage de CPU et la mémoire qu'il utilise.

Trois types d'entités particulières sont utilisées dans certains services.

- L'entité de type *netgroup* dont les attributs sont un nom et une liste d'ordinateurs du domaine définis par l'administrateur.
- L'entité de type *personnel* dont les attributs sont un nom et une liste de noms d'utilisateurs potentiels définis par l'administrateur.
- L'entité de type *UtilisateurGlobal,* qui représente un utilisateur potentiel dans le réseau. Les attributs de cet utilisateur sont son nom et le groupe Unix auquel il appartient.

4.2.2 Service de supervision orienté ordinateur

Ce service a pour but d'organiser les entités de type ordinateur, utilisateur et processus par ordinateur. Cette vue est pratique pour étudier l'activité des ordinateurs, mais elle ne permet pas d'étudier facilement l'activité d'un utilisateur particulier sur tout le réseau. Par la suite, nous présentons les mêmes informations mais organisées par utilisateur de manière à faciliter cette tâche.

Ce service est composé de cinq types d'organisateurs, qui sont présentés dans le Tableau 5. Chaque organisateur crée des requêtes spécifiques qui sont présentées dans le Tableau 4. La Figure 6 représente visuellement ce service.

Requête	Critères de la requête	
	Type entité désirée	**Attribut(s) de l'entité sur lequel il existe un critère**
Netgroup	Netgroup	Nom
Ordinateur	Ordinateur	Nom
UtilisateursOrdinateur	Utilisateur	Nom ordinateur
Utilisateur spécifique	Utilisateur	Nom ordinateur & Nom utilisateur
Processus	processus	Nom ordinateur & Nom utilisateur

Tableau 4 *Requêtes utilisées dans un service de supervision d'ordinateurs*

Organi-sateur	**Arguments**	**Créations par défaut**	**Utilisation des entités d'une requête**
A	Nom d'un Netgroup	Requête Netgroup	Chaque élément de la liste d'ordinateurs de l'entité netgroup engendre la création d'un organisateur B
B	Nom d'un ordinateur	Requête Ordinateur	L'entité de type ordinateur est destinée à être visualisée
		Organisateur C	
C	Nom d'un ordinateur	Requête UtilisateursOrdinateur	Pour chaque entité utilisateur, on crée un organisateur D
D	Nom d'un ordinateur & Nom d'un utilisateur	Requête Utilisateur spécifique	L'entité de type ordinateur est destinée à être visualisée
		Organisateur E	
E	Nom d'un ordinateur & Nom d'un utilisateur	Requête Processus	L'entité de type ordinateur est destinée à être visualisée

Tableau 5 *Organisateurs utilisés dans un service de supervision d'ordinateurs*

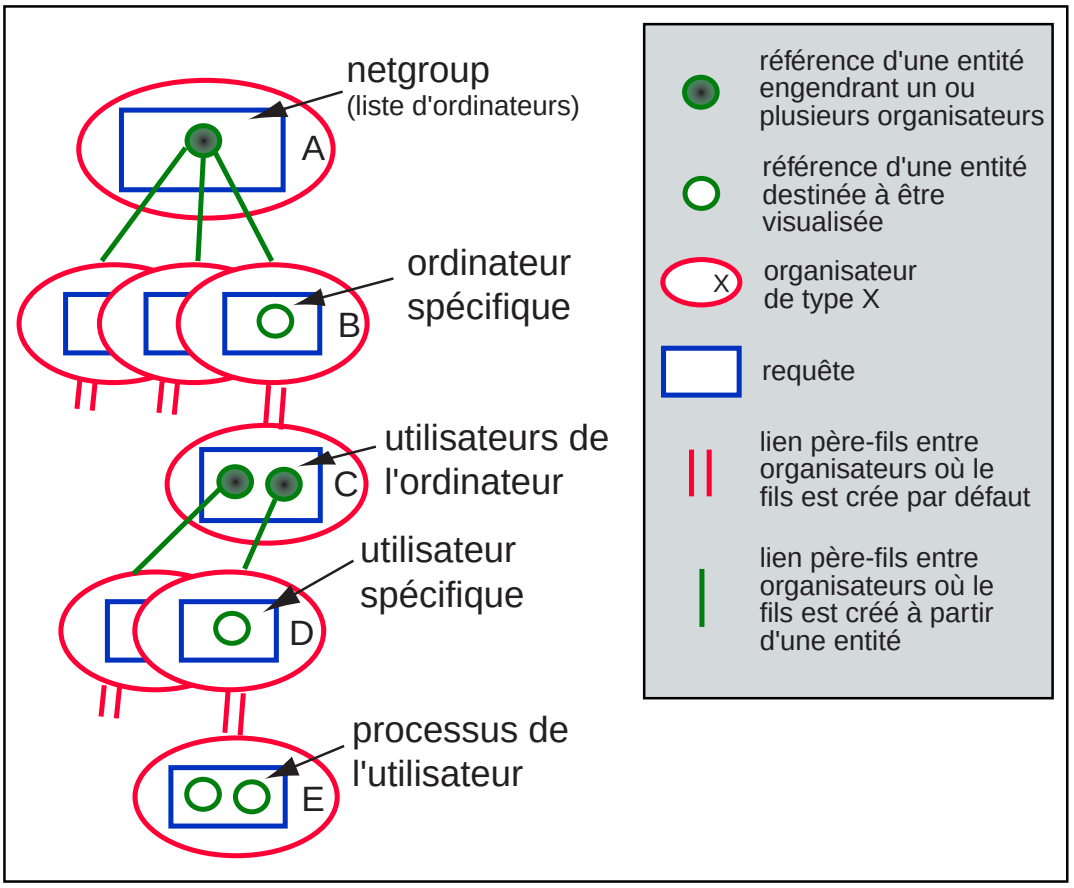


Figure 6 *Service de supervision d'ordinateurs*

4.2.3 Service de supervision orienté utilisateur

Ce service a pour but d'organiser les entités de type ordinateur, utilisateur et processus du réseau, mais contrairement au précédent service il est orienté utilisateur. Il permet donc d'organiser les informations de manière à faciliter la supervision de l'activité des utilisateurs sur le réseau pour, par exemple, détecter un utilisateur consommant beaucoup de CPU sur un grand nombre de machines.

Ce service est composé de cinq types d'organisateur, qui sont présentés dans le Tableau 7. Chaque organisateur crée des requêtes spécifiques présentées dans le Tableau 6. La Figure 7 illustre visuellement ce service.

Requête	Critères de la requête	
	Type d'entité désirée	**Attribut(s) de l'entité sur lequel il existe un critère**
Personnel	Personnel	Nom
UtilisateurGlobal	UtilisateurGlobal	Nom
Netgroup	Netgroup	Nom
Utilisateur spécifique	Utilisateur	Nom ordinateur & Nom utilisateur
Processus	Processus	Nom ordinateur & Nom utilisateur

Tableau 6 *Requêtes utilisées dans un service de supervision des utilisateurs d'un réseau*

Organi-sateur	**Arguments**	**Créations par défaut**	**Utilisation des entités d'une requête**
A	Nom d'un groupe de personnel	Requête Personnel	Chaque élément de la liste d'utilisateurs de l'entité Personnel engendre la création d'un organisateur B.
B	Nom d'un utilisateur	Requête UtilisateurGlobal	L'entité de type UtilisateurGlobal est destinée à être visualisée.
		Organisateur C	
C	Nom d'un utilisateur & nom d'un netgroup	Requête Netgroup	Chaque élément de la liste de l'entité netgroup engendre la création d'un organisateur D.
D	Nom d'un utilisateur & nom d'un ordinateur	Requête utilisateur spécifique	L'entité de type utilisateur, lorsqu'elle existe, engendre la création d'un organisateur.
E	Nom d'un utilisateur & nom d'un ordinateur	Requête utilisateur spécifique	L'entité de type utilisateur est destinée à être visualisée.
		Requête Processus	L'entité de type processus est destinée à être visualisée.

Tableau 7 *Organisateurs utilisés dans un service de supervision des utilisateurs d'un réseau*

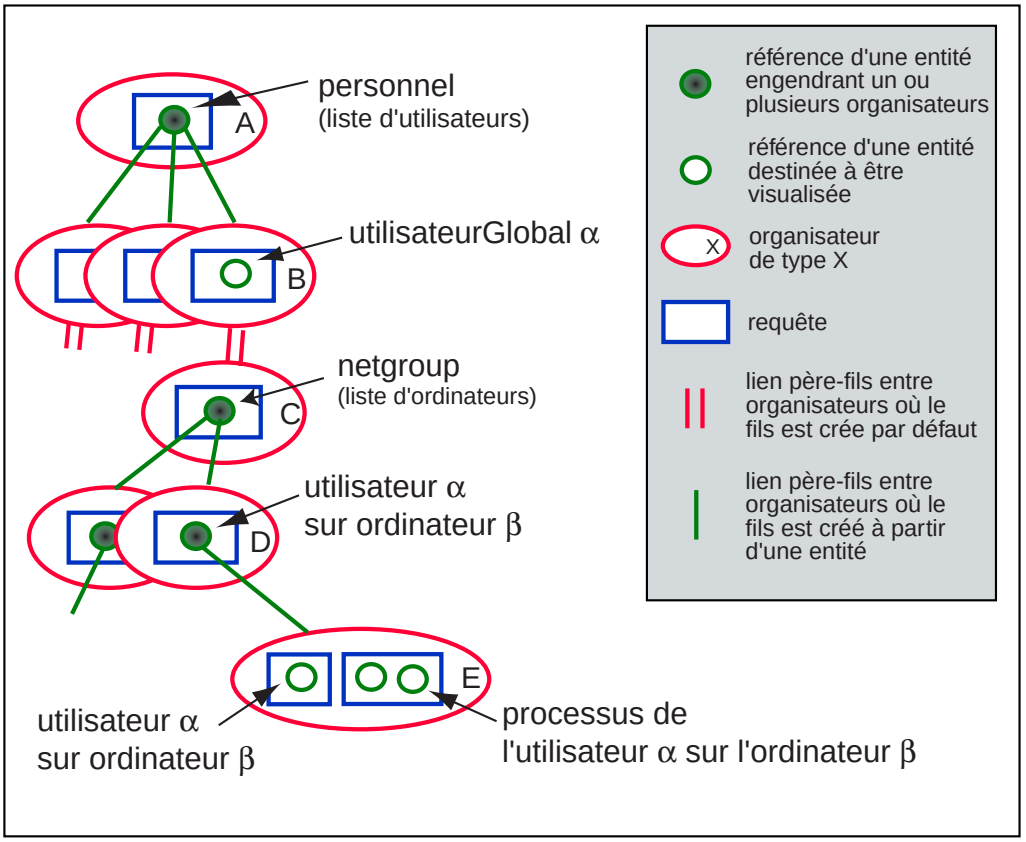


Figure 7 *un service de supervision des utilisateurs d'un réseau*

4.2.4 Conclusion

Ces deux exemples ont montré que le concept d'entité de notre modèle permet de partager des informations entre plusieurs services, tout en permettant à chacun de les structurer comme il l'entend de manière à remplir la tâche qui leur est assignée. Ce partage des informations est particulièrement important dans le cas où les deux services sont utilisés en même temps, car il permet de garder la cohérence entre eux, et comme nous le verrons plus tard, de faciliter la synchronisation des deux visualisations.

5 Perspectives

De nombreux paramètres supplémentaires pourraient être ajoutés aux organisateurs afin de leur donner plus de puissance dans le traitement de l'information. En particulier, il serait intéressant de proposer des fonctions de filtrage des informations, afin de permettre à l'utilisateur final de régler la densité d'informations de sa visualisation à travers un pilotage du service. Pour cela, les critères des requêtes devraient être accessibles en temps réel à l'utilisateur final

afin qu'il définisse la densité d'informations qu'il désire. Les travaux de [Ahlberg94] vont dans ce sens, mais sans prendre en compte le caractère hiérarchique et éventuellement dynamique des informations.

Une autre fonction utile, permettant à l'utilisateur de pouvoir changer la densité d'informations qu'il supervise, serait de lui proposer un résumé de n'importe quel sous-arbre du service. Ceci permettrait à l'utilisateur d'obtenir des visualisations très synthétiques, intéressantes pour avoir un aperçu global des informations, mais demanderait plus de travail au concepteur du service car il devrait définir comment cette synthèse doit être effectuée, à moins qu'elle ne soit automatisée.

6 Conclusion

Nous avons présenté un modèle de données permettant d'identifier et de structurer les informations nécessaires à une tâche, pour donner naissance à un service. Notre modèle repose sur des bases déjà bien connues. D'une part, des entités typées regroupant des informations brutes, et d'autre part, des requêtes permettant d'obtenir des références à des entités qui satisfont à certains critères de type et/ou d'attributs.

L'originalité de ce modèle réside dans la possibilité de définir des squelettes de service permettant de générer automatiquement des instances de service en fonction des informations existantes. Ce squelette est défini sous la forme d'un arbre dont les nœuds sont des organisateurs qui encapsulent des requêtes, et réagissent aux informations obtenues par celles-ci de manière à créer dynamiquement d'autres organisateurs, et ainsi construire l'arbre dont ils font partie. Les organisateurs permettent également de différencier les informations utile à la visualisation de celles permettant la construction de l'arbre, point important puisqu'un service est destiné à être exploité pour la construction de visualisations.

Le modèle spécifie quelques points plus orientés vers l'implémentation, mais nécessaires à son fonctionnement dans un environnement dynamique, où plusieurs services distincts sont susceptibles d'utiliser les mêmes informations. En particulier, nous avons mis en avant la nécessité de partager les informations et les conséquences du caractère dynamique de celles-ci (évolution des entités, des requêtes et de l'arbre d'organisateurs dans le temps).

3 Modèle de visualisation pour la construction de mondes 3D dynamiques

1 Introduction

Dans le chapitre précédent, nous avons présenté un modèle de données permettant de structurer les informations destinées à être visualisées. Dans le projet CyberNet, les informations sont visualisées sous forme de monde 3D interactifs. Ce chapitre présente un modèle de visualisation permettant de définir la partie visuelle de tels mondes sans tenir compte de ses aspects interactifs, ce modèle ayant pour but d'être intégré au sein d'un modèle plus général permettant la définition de mondes 3D interactifs.

Le modèle a pour but de permettre la définition et la construction rapide de mondes 3D en proposant des concepts génériques de haut niveau conçus spécialement pour le mappage d'informations. Ces concepts sont matérialisés sous la forme de composants réutilisables que l'on peut assembler afin de construire des mondes 3D.

Notre système utilise ce modèle pour automatiser la construction de mondes 3D à partir d'un ensemble d'informations dynamiques. Ce caractère dynamique se retrouve donc également dans les mondes 3D, où il peut engendrer une instabilité visuelle. Nous montrons comment notre modèle a été pensé pour prendre en compte ce type de problème.

Avant d'exposer en détail ce modèle, nous présentons différentes techniques de visualisation en 2D et en 3D que l'on peut trouver dans la littérature scientifique. En dernier lieu, nous présentons des exemples de différents types de mondes 3D crées en utilisant notre modèle.

2 Techniques de visualisation

Une partie des buts d'une visualisation d'informations est de pouvoir comprendre, explorer et analyser les informations par une méthode visuelle. De nombreux types de visualisations ont été inventés de manière à exploiter les capacités de perception visuelle humaines. Les premières visualisations d'informations ont été réalisées bien avant l'arrivée de l'outil informatique. [Tufte83] rapporte le travail de [Playfair1786] comme un des premiers utilisant des propriétés visuelles abstraites, comme des lignes ou des aires, pour représenter des informations.

Les premiers principes de la visualisation d'informations ont été développés par un cartographe qui a identifié les différents éléments graphiques de base et comment les utiliser [Bertin67][Bertin77]. Le travail d'un autre précurseur de la visualisation d'informations, [Tukey77], met l'accent sur la capacité d'une visualisation à donner un aperçu d'un ensemble d'informations. Par la suite, la visualisation d'informations a pris son essor, notamment grâce à l'utilisation de l'ordinateur qui a permis l'automatisation des constructions graphiques et l'interaction de l'utilisateur sur les visualisations.

Dans un premier temps, nous présentons les indicateurs visuels permettant de transmettre de l'information vers l'utilisateur sous forme visuelle. Ensuite, nous présentons différentes techniques utilisées pour visualiser des informations. Ces techniques sont présentées en fonction de la structuration des informations : en premier lieu nous abordons brièvement la visualisation d'informations non structurées pour ensuite parler plus en détail des informations structurées, notamment lorsque celles-ci sont représentées à l'aide de métaphores 3D. Comme décrit dans le précédent chapitre, le terme *entité* est utilisé dans cette partie pour désigner un groupe d'informations, par exemple une entité de type «employé » peut contenir les informations «nom de l'employé », « date de naissance », etc. ; chaque information est appelée un *attribut* de l'entité.

2.1 Indicateurs visuels

Dans le domaine de la visualisation d'informations, la transmission visuelle des informations est effectuée par l'intermédiaire d'indicateurs visuels. Ces indicateurs visuels ont été identifiés par [Bertin67][Bertin77] ainsi que par [Card97][Mackinlay86].

Dans nos visualisations, nous exploitons les indicateurs visuels suivants :

- Les formes géométriques qui peuvent être définies en 1D (un point), 2D (un carré) ou en 3D (un cube). En 2D et en 3D, ces formes géométriques peuvent être complexes (une maison par exemple), on les appelle alors souvent des icônes ou des glyphes.
- Les attributs visuels des formes géométriques, souvent appelés paramètres rétiniens, comme par exemple la couleur ou la taille. Ceux-ci peuvent être plus ou moins nombreux selon le type de visualisation. C'est en 3D que l'on

dispose du plus grand nombre d'attributs visuels, en partie grâce à la dimension supplémentaire et au modèle de rendu qui permet de gérer facilement des attributs visuels comme l'éclairement, la transparence ou la texture.

- la position et l'orientation dans l'espace des formes géométriques les unes par rapport aux autres. Encore une fois, la 3D apporte une dimension supplémentaire permettant d'exploiter au mieux cet indicateur visuel.
- L'animation dans le temps des précédents indicateurs visuels.

Les deux parties suivantes présentent des exemples de techniques de visualisation. Nous allons voir que la majorité des techniques se concentrent sur le positionnement des formes géométriques dans l'espace car c'est un indicateur visuel primordial pour réaliser une bonne visualisation.

2.2 Informations non structurées

Dans le cas d'informations non structurées, le but de la visualisation est de permettre à l'utilisateur de comprendre, de classifier, et de trouver des relations entre les entités visualisées. A titre d'exemple, nous présentons deux techniques parmi les plus connues.

2.2.1 Nuage de points

La plus connue et la plus ancienne des techniques de visualisation d'informations non structurées est le nuage de points. Il s'agit d'afficher chaque entité sous forme d'un point dans un repère à une, deux ou trois dimensions selon le nombre d'attributs de l'entité. Cette technique peut être améliorée grâce à l'utilisation de couleurs et d'éléments graphiques supplémentaires [Tukey77][Tufte83], d'interactions [Cleveland88] ou de matrice de nuage de points liés [Cleveland93] et peut même permettre de visualiser de grandes quantités d'informations comme des bases de données[Keim96]. Le point fort de cette technique est qu'elle procure une vision intuitive, sous forme de motif, d'un ensemble d'entités.

Ces points peuvent être remplacés par des icônes paramétrables permettant d'augmenter le nombre d'attributs affichables pour une entité, chaque paramètre de l'icône permettant d'afficher un attribut de l'entité. Le travail pionnier dans ce domaine est l'utilisation de métaphores de visages comme icônes [Chernoff73], dont les paramètres sont la forme et la taille du visage, des yeux, l'orientation des sourcils, etc. Comme les autres techniques métaphoriques, l'utilisation d'icônes de visage est intéressante car elle permet d'exploiter des concepts déjà connus par l'utilisateur et donc de faciliter leur compréhension.

2.2.2 Coordonnées parallèles

La technique de coordonnées parallèles [Inselberg85] [Inselberg90] (cf. Figure 8) consiste à d'utiliser autant d'axes parallèles qu'il y a d'attributs dans les entités

que l'on veut visualiser. En affichant les axes côte à côte, une entité est représentée sous la forme d'une ligne se brisant à chaque intersection avec un axe. L'intersection entre cette ligne et chaque axe correspond à la valeur de l'attribut associée à cet axe pour cette entité.

La force de cette technique est qu'elle permet de visualiser des entités de n'importe quelles dimensions et qu'il est facile de déceler dans la visualisation des motifs récurrents menant à une analyse globale de toutes les entités. L'ajout de couleurs et d'interactions permet d'améliorer l'efficacité de cette méthode.

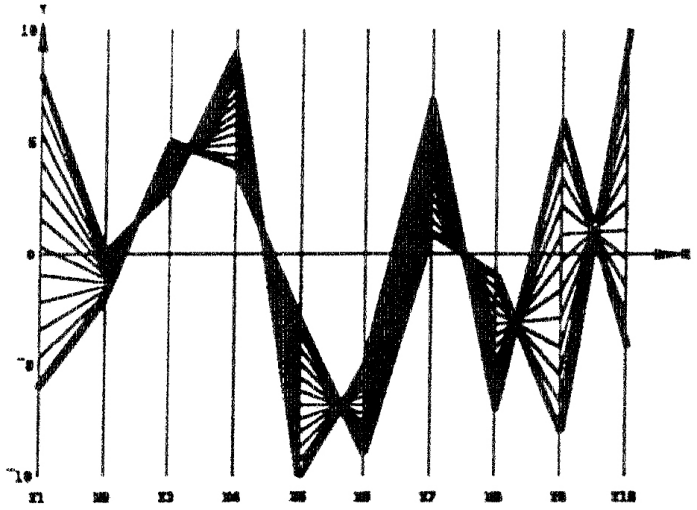


Figure 8 *Coordonnées parallèles [Inselberg85]*

2.3 Informations structurées

Les informations peuvent être considérées comme structurées lorsqu'il existe des relations entre les entités. Dans ce cas, les entités peuvent être considérées comme les nœuds d'un graphe et les liens de ce graphe comme les relations entre ces entités. La visualisation de ce type d'informations vise plus à superviser et analyser les informations qu'à les classifier comme c'est le cas pour les informations non structurées. D'excellents états de l'art ont été écrits sur ce sujet, certains plus orientés vers la visualisation d'informations [Herman00], d'autres plus généraux [Battista94][Battista99].

Un type de graphe a particulièrement retenu l'attention des chercheurs : la structure d'arbre. Nous présenterons des exemples de techniques de visualisation de ce type de graphe en détail mais, auparavant, nous présentons des exemples de techniques inventées pour un graphe quelconque.

2.3.1 Graphe

La forme la plus courante pour représenter un graphe est de représenter ses entités sous forme de points et les relations sous forme de lignes. Dans le cas où les entités ont plusieurs attributs, les points peuvent être remplacés par des icônes afin de pouvoir afficher ces attributs. De même, les relations peuvent être visualisées par un élément graphique plus complexe que la ligne. Ce type de représentation est relativement facile à réaliser dans le cas d'un graphe avec peu de nœuds et de relations. Les problèmes surviennent lorsque la taille du graphe augmente, et dans certains cas il n'est même pas possible de proposer une représentation acceptable du point de vue visuel, en partie à cause des lignes représentant les relations, qui font ressembler le graphe à un plat de spaghetti.

Le problème dans la visualisation d'un graphe est de définir une procédure permettant de positionner les nœuds de manière à obtenir un résultat visuel agréable. Un premier type de placement, appelé Sugiyama [Sugiyama89], permet de positionner les nœuds en définissant tout d'abord des couches contenant tous les nœuds, puis en plaçant tout à tour les nœuds d'une couche donnée à partir de critères esthétiques comme le non-croisement des lignes représentant les relations. Une autre possibilité de positionnement des nœuds est d'utiliser un système masse-ressort [Eades84] : on assigne un poids à chaque nœud et une force à chaque relation, puis on effectue une simulation du système par itération jusqu'à atteindre un équilibre. D'autres méthodes basées sur des simulations peuvent également être utilisées comme la méthode de recuit simulé (simulated annealing) [Cruz95].

La majorité des méthodes peuvent être mises en œuvre aussi bien en 2D qu'en 3D. Par exemple, [Sprenger98][Hendley95][Wood95] ont utilisé un système masse-ressort en 3D (cf. Figure 9). Un cas particulier est SemNet [Fairchild88], un des premiers système de visualisation de graphe en 3D, qui permet de choisir parmi plusieurs méthodes de placement basées sur des heuristiques, mais qui laisse aussi la possibilité à l'utilisateur de placer lui-même les nœuds dans l'espace. La 3D apporte plus d'espace pour placer les nœuds ainsi que des possibilités graphiques supplémentaires (éclairage, transparence,…) , mais génère aussi des problèmes d'occlusions inhérents à la visualisation 3D ainsi que la nécessité de fournir des moyens de navigation dans l'espace 3D.

Une technique pour visualiser les graphes en 3D est basée sur l'utilisation de la géométrie hyperbolique [Munzer97] (cf. Figure 10). Cette technique effectue les calculs de placement des entités et des relations dans un espace hyperbolique, avant de revenir dans l'espace euclidien pour donner un résultat intéressant.

Lorsque les données à visualiser ont une réalité physique, il est possible d'utiliser l'aspect visuel inhérent à cette réalité physique pour créer la visualisation. Par exemple [Cox92], [Cox96] et [Eick96] ont crée des visualisations géographiques 3D de réseaux informatiques (cf. Figure 11 et Figure 12).

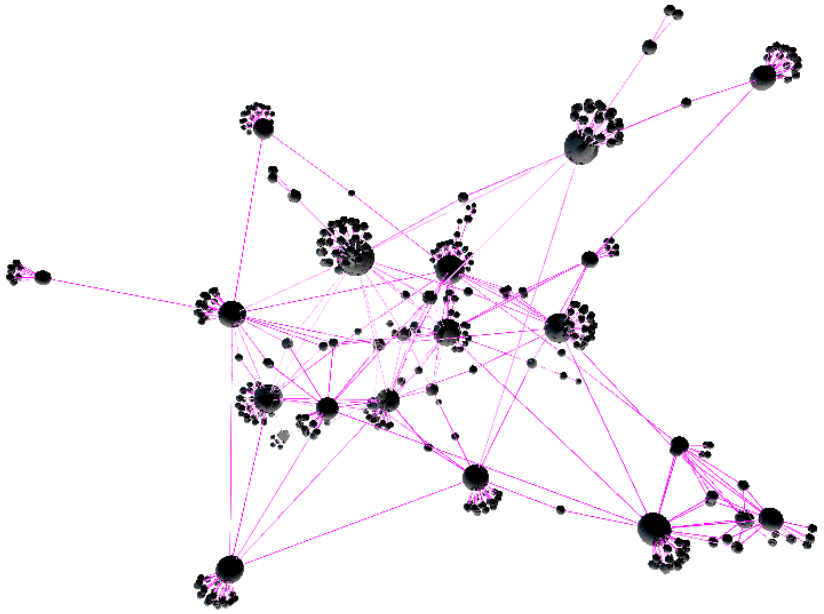


Figure 9 *Visualisation d'un graphe en 3D à l'aide d'un système masse-ressort (image extraite de [Wood95])*

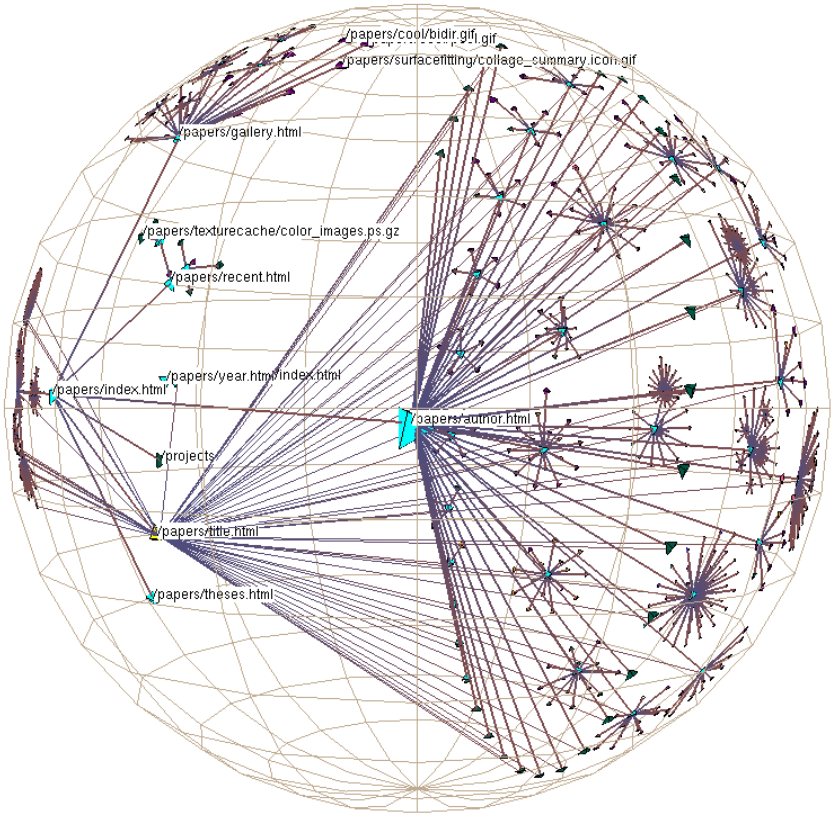


Figure 10 *Visualisation d'un graphe en 3D à l'aide de la géométrie hyperbolique (extrait de [Munzer97])*

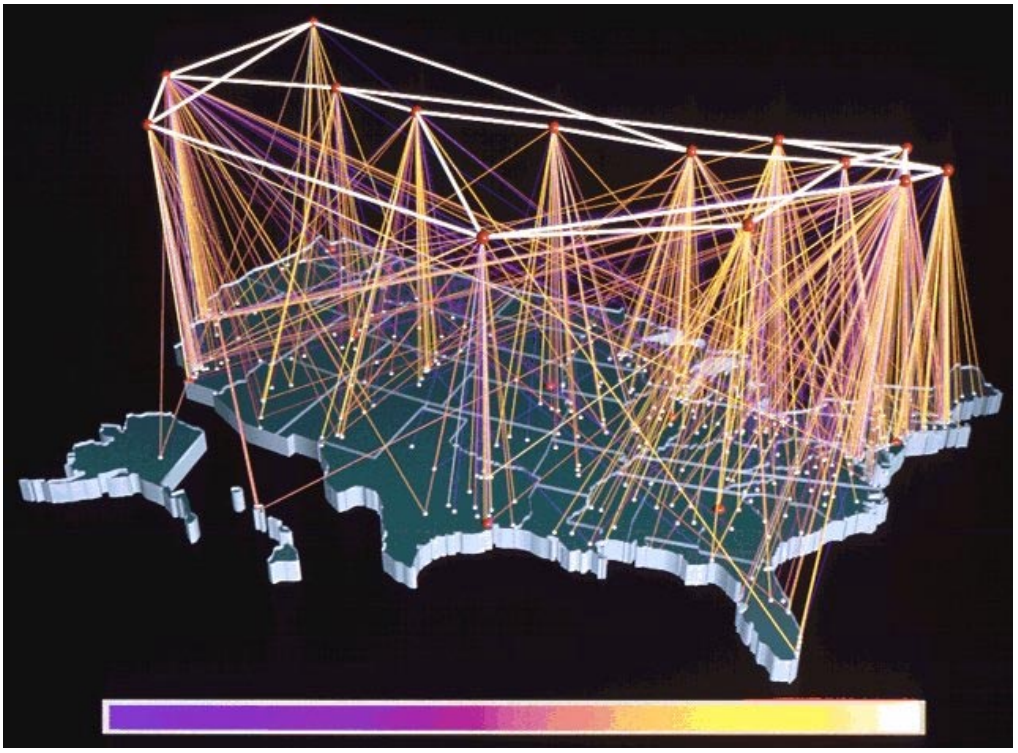


Figure 11 *Visualisation géographique du réseau NSFNET ([Cox92])*

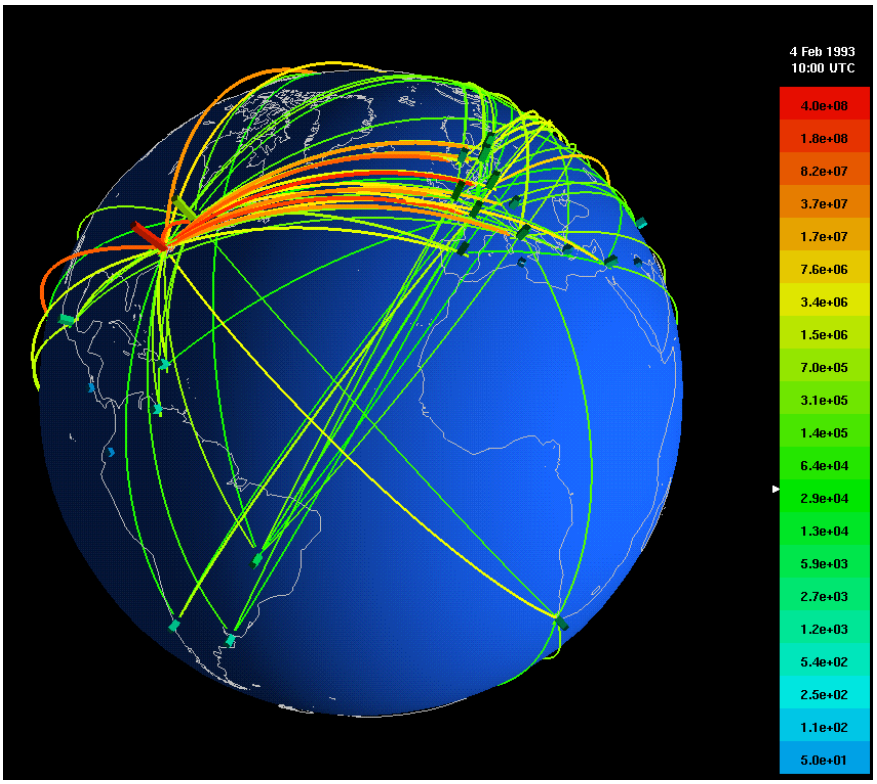


Figure 12 *Visualisation géographique de du traffic sur Internet ([Cox96])*

Une méthode couramment utilisée consiste à calculer, à l'aide d'algorithmes plus ou moins évolués, un arbre extrait du graphe que l'on désire visualiser, puis à utiliser une technique d'affichage d'arbre et enfin à rajouter les relations manquantes. Cette méthode est souvent utilisée car les algorithmes d'affichage d'arbres sont moins complexes que ceux de graphes, pour un résultat équivalent lorsque le graphe dépasse quelques centaines de nœuds.

2.3.2 Arbre

Les techniques présentées pour la visualisation de graphe peuvent bien sûr être appliquées au cas particulier du graphe orienté acyclique, c'est-à-dire à la structure d'arbre, mais des méthodes de positionnement des entités plus efficaces ont été créées spécialement pour les arbres. Ces méthodes sont moins lourdes et donnent des résultats graphiques plus satisfaisants car la structure d'arbre est bien moins contraignantes que celle d'un graphe générique. Nous présentons dans un premier temps des techniques permettant de créer une visualisation 2D puis d'autres exploitant les possibilités de la 3D.

2.3.2.1 Visualisation 2D

L'algorithme classique de positionnement des nœuds d'un arbre est celui de Reingold et Tilford [Reingold81] qui produit un résultat où chaque sous-arbre isomorphique est représenté de la même façon. De nombreuses autres techniques ont été développées. La technique dite radiale [Eades92] place les nœuds sur des cercles concentriques en fonction de leur profondeur dans l'arbre. La géométrie hyperbolique, dont l'utilisation a déjà été présentée pour les graphes, a été à l'origine utilisée pour les arbres [Lamping95][Lamping96] (cf. Figure 13).

Les méthodes que nous venons de voir reposent toutes sur l'utilisation de nœuds représentés sous forme de points ou d'icônes et de liens représentés sous forme de lignes. Des approches différentes ont été inventées. Une des plus intéressantes est celle des TreeMap [Johnson91], qui remplit la zone d'affichage de rectangles représentant les nœuds du graphe, le rectangle d'un nœud étant toujours contenu dans les rectangles de ses ancêtres et la taille d'un rectangle étant proportionnelle à un attribut de l'entité associée à ce nœud. [Andrews98] propose une méthode appelée « Information Slices », qui reprend des concepts des TreeMaps mais en utilisant un ou plusieurs demi-cercles contenant d'autres demi-cercles concentriques correspondant aux différents niveaux de la hiérarchie. [Sindre93] utilise une technique dite de «graphe d'oignon » qui permet de visualiser une hiérarchie à l'aide d'une série de boîtes imbriquées. Cheops [Beaudoin96] est un système utilisant des empilements de triangles pour représenter d'une façon très compacte une hiérarchie.

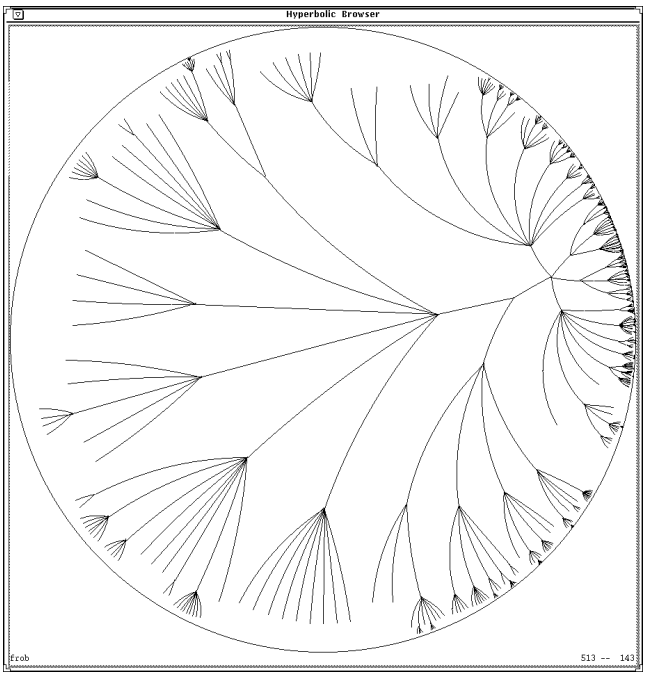


Figure 13 *Visualisation d'un arbre en 2D à l'aide de la géométrie hyperbolique (extrait de [Lamping95])*

2.3.2.2 Visualisation 3D

De la même manière que pour les graphes, certains algorithmes prévus initialement en 2D s'étendent très bien en 3D. Nous avons déjà présenté le travail en géométrie hyperbolique de [Munzer97] sur les graphes, dont les résultats graphiques sont encore meilleurs pour les arbres étant donné les plus faibles contraintes. [Rekimoto93] propose un équivalent 3D des «graphes d'oignon » avec son «cube d'informations » qui utilise des boîtes transparentes imbriquées pour représenter chaque nœud du graphe (cf. Figure 14).

Un type de visualisation de hiérarchie 3D ayant particulièrement marqué la communauté des chercheurs est l'arbre de cône (cone tree) [Roberston91] [Carrière95] (cf. Figure 15). L'algorithme de placement est relativement simple : chaque nœud interne du graphe est placé au sommet d'un cône et ses enfants sont répartis sur la base du même cône. La taille des cônes peut être équivalente pour un même niveau de hiérarchie ou alors s'adapter afin d'utiliser au mieux l'espace. [Jeong98] a pensé à remplacer les cônes par de simples disque de manière à minimiser l'occlusion que peuvent induire les cônes. Dans un arbre de cône, l'affichage de l'arbre se fait verticalement, une variante appelée arbre à cames (cam tree) affiche l'arbre horizontalement ce qui peut être utile si les nœuds du graphe doivent afficher du texte.

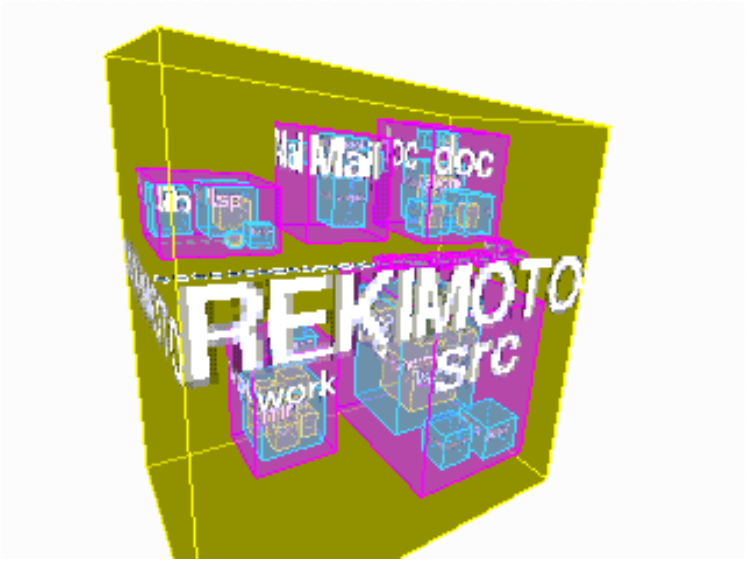


Figure 14 *Visualisation d'un arbre en 3D à l'aide de boîtes transparentes imbriquées (image extraite de [Rekimoto93])*

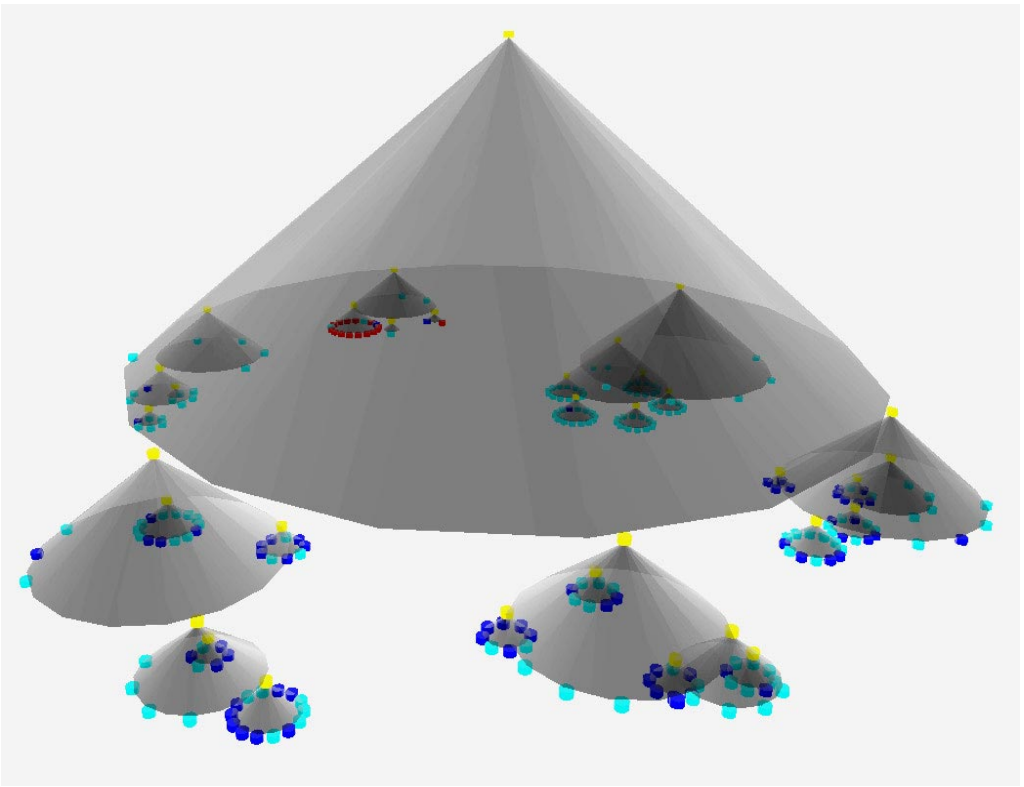


Figure 15 *Visualisation d'un arbre en 3D à l'aide d'une stratégie d'arbre de cônes (adapté de [Roberston91])*

2.4 Visualisation 3D métaphorique d'informations structurées sous forme d'arbre

L'utilisation de la 3D n'a pas pour seul avantage d'offrir plus d'espace pour positionner les nœuds d'un graphe. Elle permet aussi d'apporter plus de réalisme à la représentation graphique, d'exploiter au mieux les indicateurs visuels, et permet donc de créer des métaphores du monde réel qui peuvent aider à mieux appréhender des structures complexes ainsi que les informations qu'elles contiennent.

Les métaphores ont l'avantage d'utiliser des éléments visuels ayant un aspect sémantique que l'utilisateur connaît déjà de par son expérience personnelle. De plus, les éléments visuels de la métaphore possèdent des relations ensemblistes et structurelles qui peuvent être exploitées pour la visualisation des informations. Par exemple, dans une métaphore de ville, l'utilisateur connaît *a priori* la différence sémantique entre un immeuble et une maison (le premier est plus grand que la seconde), et il sait que les éléments de ce type appartiennent à un quartier (relation structurelle). Nous reviendrons plus en détail sur l'utilisation des métaphores dans le chapitre 5.

Il est parfois assez délicat de délimiter la frontière entre les visualisations métaphoriques et celles qui ne le sont pas. Par exemple, l'arbre de cônes que nous avons présenté possède un caractère métaphorique dans le sens où il utilise une structure visuelle faisant penser à un arbre. Dans cette partie, nous présentons des métaphores plus riches dans le sens où elles utilisent un ensemble d'éléments et de structures visuelles appartenant à une même métaphore.

2.4.1 Paysage d'informations

La métaphore de paysage d'informations consiste à disposer les différents éléments visuels sur un plan de manière à construire une espèce de paysage. Selon les éléments visuels utilisés et la stratégie utilisée pour les placer, cette métaphore peut être plus ou moins riche.

Un des premiers exemples de ce type de visualisation métaphorique est le programme FSN [Tesler92] permettant de visualiser un système de fichiers Unix (cf. Figure 16). Les répertoires sont représentés par des blocs placés sur un plan, leur hauteur représentant la somme des tailles des fichiers contenus dans le répertoire. Des arcs entre les différents répertoires permettent de visualiser leur hiérarchie. Les fichiers sont représentés par de plus petits blocs posés sur le bloc représentant le répertoire auquel ces fichiers appartiennent. La taille d'un fichier est visualisée par la hauteur du bloc et son type par une image projetée sur le haut du bloc. L'utilisateur a bien sûr la possibilité de naviguer entre les différents répertoires de manière à explorer la hiérarchie.

[Andrews95][Andrews96] utilise une métaphore similaire pour visualiser une hiérarchie de fichiers avec des améliorations permettant d'enrichir la métaphore, comme le choix de l'icône 3D associée à chaque fichier ou l'utilisation de la couleur. Une possibilité intéressante de ce système est de permettre de visualiser

également des liens traversant la structure hiérarchique (comme les liens symboliques), de sorte que la visualisation d'une hiérarchie se transforme en visualisation de graphe. Les liens traversants sont alors visualisés en dessous et au-dessus du plan servant de base à la visualisation des fichiers.

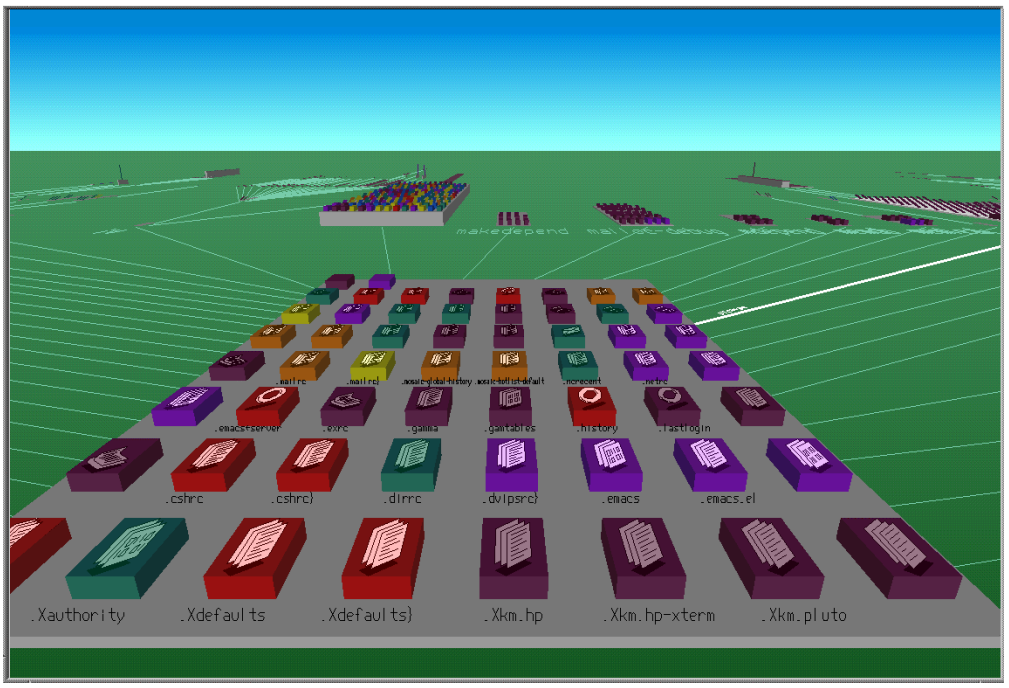


Figure 16 *Visualisation d'un système de fichiers à l'aide d'une métaphore de paysage (image produite par le logiciel fsn [Tesler92])*

2.4.2 Pyramides d'informations

Les pyramides d'informations [Andrews97] exploitent une métaphore de plateaux posés les uns sur les autres : un plateau représente la racine de l'arbre et d'autres plateaux plus petits posés dessus représentent les sous-arbres (cf. Figure 17). Chaque plateau représente donc un nœud interne de l'arbre, et les feuilles de l'arbre sont quant à elles représentées par des icônes 3D posés sur les plateaux. Le résultat graphique est un ensemble de pyramides avec des bases communes, d'autant plus hautes que les sous-arbres qu'elles représentent sont profonds. Il est intéressant de noter que l'utilisation de cette méthode génère automatiquement des plateaux dont les dimensions sont proportionnelles à la taille du sous-arbre que représentent les plateaux.

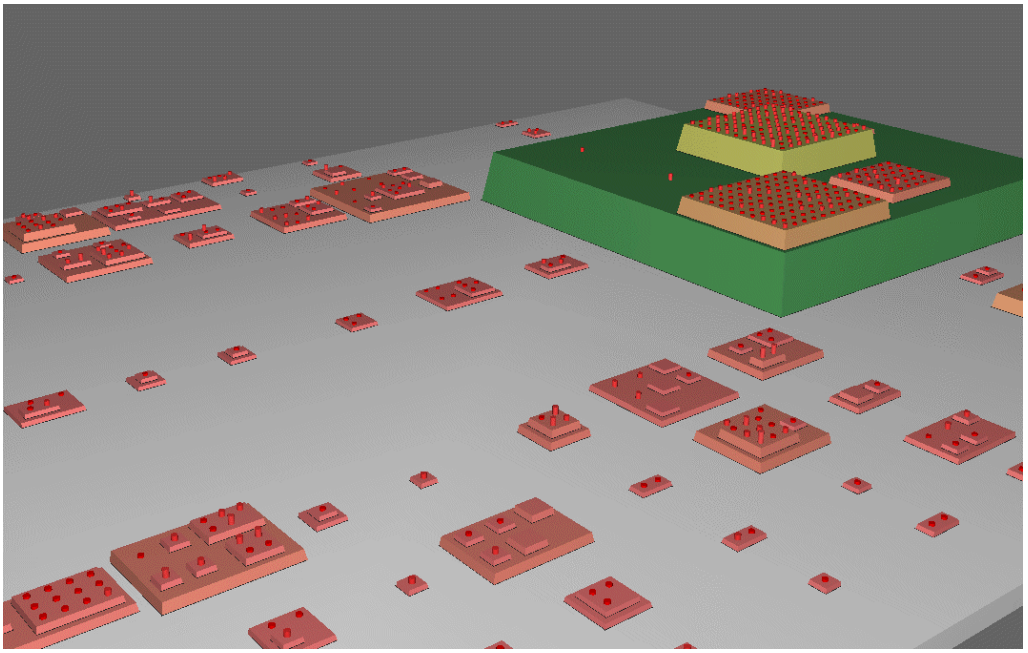


Figure 17 *Visualisation 3D d'un arbre à l'aide d'une métaphore de pyramide (adapté de [Andrews97])*

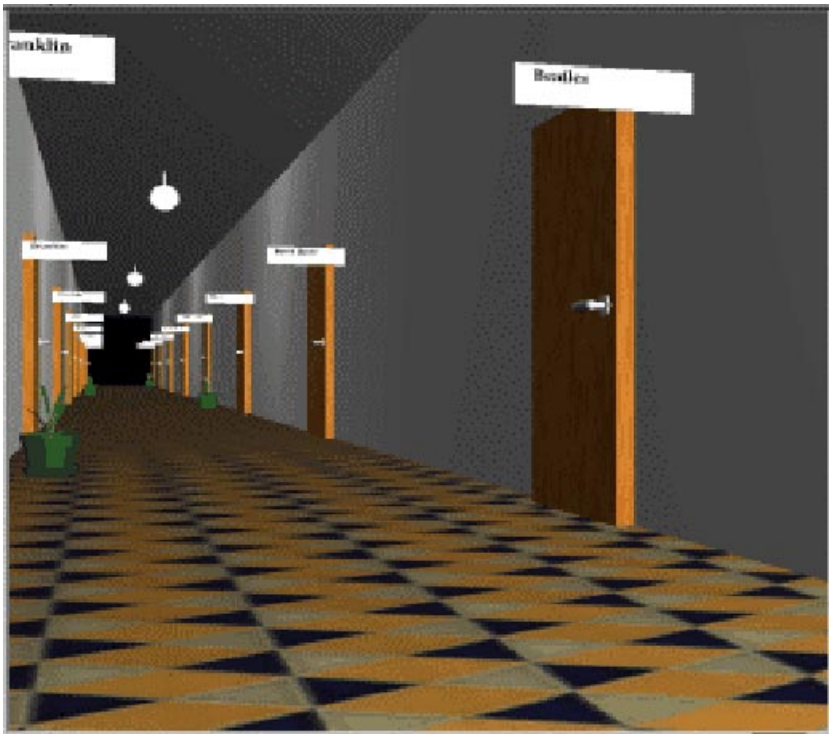


Figure 18 *visualisation utilisant une métaphore de bâtiments composée de couloirs et de pièces [Massari97]*

2.4.3 Métaphores de bâtiments

Le projet VIRGILIO [Massari97] permet d'explorer des bases de données multimédias, classées hiérarchiquement, grâce à l'utilisation de métaphores de bâtiments (cf. Figure 18). Cette métaphore est particulièrement riche car les éléments qui la constitue sont très structurés : un bâtiment contient un certain nombre d'étages contenant chacun au moins un couloir qui contient des pièces. Le problème de cette métaphore est qu'il est impossible d'obtenir une vue d'ensemble de la structuration des éléments visuels, à moins de recourir à des interactions permettant de faire disparaître ou de rendre transparentes les structures du bâtiment [Russo00b].

2.4.4 Métaphores de villes

La ville est également une métaphore présentant une hiérarchie (quartiers, rues, différents type de bâtiments, etc.) qui peut être exploitée de manière à présenter des informations hiérarchiques. [Map.net] exploite cette métaphore pour visualiser un répertoire hiérarchique de sites Web (cf. Figure 19). Ce type de métaphore particulièrement riche permet de tirer parti des nombreuses possibilités de la 3D, notamment des indicateurs visuels. De plus, elle permet d'obtenir facilement une vue d'ensemble des informations en survolant la ville.

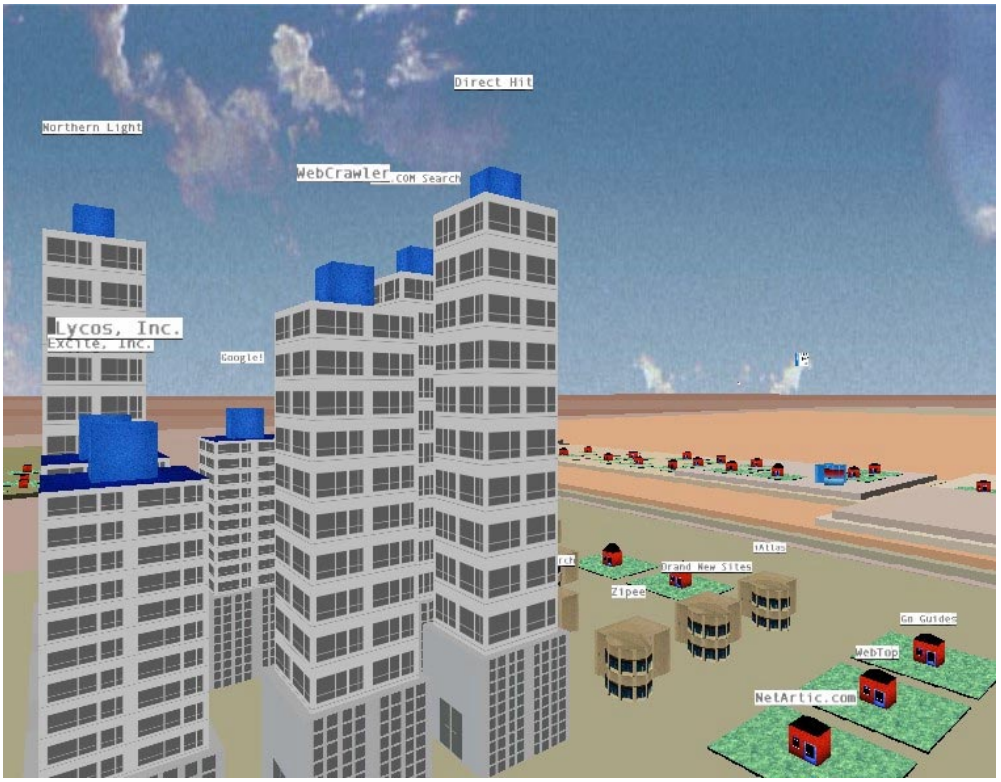


Figure 19 *Métaphore de ville pour visualiser un répertoire de sites Web ([Map.net])*

2.4.5 Autres métaphores

Bien d'autres métaphores ont été développées. Par exemple, on peut citer la métaphore du système solaire, qui permet de représenter les informations sous la forme de satellites, de planètes et d'étoiles gravitant les uns autour des autres [Abel99]. Un autre exemple est une métaphore basée sur la salle de marché de la Bourse de New-York (cf. Figure 20), actuellement utilisée pour la visualisation des données boursières [Delaney99].

Figure 20 *Visualisation de données boursières basée sur une salle de marché réelle [Delaney99]*

3 Modèle de visualisation de CyberNet

Cette partie présente le modèle permettant de créer la partie visuelle des mondes 3D interactifs utilisés pour visualiser les informations au sein du projet CyberNet. Ce modèle a pour but d'être intégré dans le modèle plus général de mondes 3D interactifs qui sera exposé au chapitre 5.

Dans un premier temps nous présentons les objectifs qui ont motivé les spécifications de ce modèle, puis nous abordons les concepts de base de celui-ci pour ensuite détailler comment la gestion de l'espace 3D est effectuée, en particulier dans des situations où les informations sont dynamiques.

3.1 Objectifs

3.1.1 Un modèle ouvert

Ce modèle de visualisation a pour objectif principal de faciliter la création de tout type de monde 3D servant de support à la visualisation d'informations structurées sous forme d'arbre. En particulier, ce modèle a pour but de pouvoir au moins créer n'importe quel type de visualisation d'arbre existante comme les arbres de cônes, les paysages d'informations, les pyramides d'informations ou les métaphores de ville.

3.1.2 Faciliter le mappage

Ce modèle est conçu pour construire des mondes 3D permettant de visualiser des informations contenues dans un service tel que nous l'avons décrit au chapitre précédent, c'est à dire un ensemble d'informations structuré en arbre. Le modèle propose donc des concepts offrant des indicateurs visuels sur lesquels on peut mapper les informations contenues dans un service.

3.1.3 Gérer le caractère dynamique

Le caractère dynamique des informations à visualiser implique que les mondes 3D sont voués à évoluer au cours du temps. Le modèle propose donc des concepts et des mécanismes qui prennent en compte ce caractère dynamique. Par exemple, la modèle offre des solutions pour limiter l'instabilité visuelle engendrée par l'apparition, la disparition ou la modification d'un élément dans le monde 3D.

3.1.4 Faciliter la définition et la construction des mondes 3D

Dans le but de faciliter la définition de nouveaux types de visualisations ainsi que la construction des mondes 3D, nous avons appliqué des concepts inspirés de l'analyse orientée objet [Meyler88] [Rumbaugh91] [Booch94]. Le résultat est la définition d'objets visuels autonomes et réutilisables que l'on peut assembler afin de construire un monde 3D. Ces objets, appelés composants graphiques, vont être maintenant présentés.

3.2 Concepts de base

3.2.1 Composant graphique (CG)

Notre modèle de visualisation est basé sur la notion de *composant graphique* (CG). Un CG est un objet intervenant dans la réalisation du monde 3D en exploitant certains indicateurs visuels. En assemblant des CG, il est possible de construire un monde 3D à la manière des célèbres legos.

L'utilisation de CG a deux grands avantages. Premièrement, de par leur capacité à s'assembler, ils facilitent l'automatisation de la construction de la visualisation, facteur important puisque notre système construit toutes les visualisations automatiquement à partir d'informations dynamiques. Deuxièmement, les CG ont la propriété d'être réutilisables et peuvent donc être utilisés dans différents contextes, et ainsi accélérer la création de nouveaux types de visualisation. En d'autres termes, les CG sont sémantiquement neutres, leur aspect sémantique existant seulement lorsqu'ils sont placés dans un contexte.

Il existe deux types de CG permettant de construire la visualisation 3D :

- Les *glyphes*
- Les *gestionnaires de placement* (GP)

Chacun d'entre eux offre des indicateurs visuels permettant de mapper des informations.

3.2.2 Les glyphes

Les glyphes [Chuah98] sont des objets visuels 3D prédéfinis permettant d'exploiter des indicateurs visuels de formes géométriques et d'apparence ainsi que leurs animations. Pour cela, un glyphe est constitué d'une ou plusieurs formes géométriques 3D ayant des attributs visuels (dimensions, couleur, texture, transparence). Une partie de ces indicateurs visuels sont contrôlables afin de pouvoir modifier la représentation graphique du glyphe. Ces indicateurs sont appelés *paramètres visuels* du glyphe. Chaque paramètre visuel permet d'afficher une information. Ainsi la complexité d'un glyphe peut être définie par le nombre de paramètres qu'il offre et donc par le nombre d'informations que ce glyphe peut afficher en même temps.

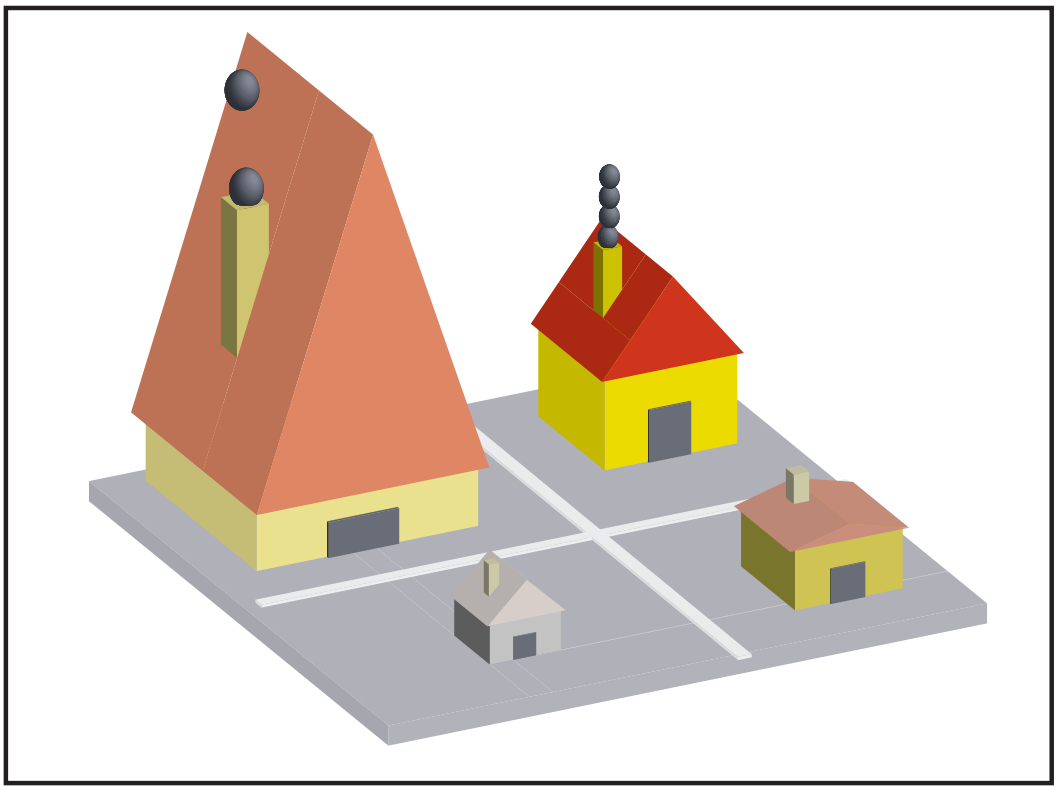
Un glyphe peut prendre de très nombreuses formes. Un exemple de glyphe très simple est un cube dont les paramètres visuels sont ses dimensions et sa couleur. Un exemple de glyphe plus complexe est présenté à la Figure 21 avec plusieurs représentations correspondant à différentes valeurs de ses paramètres visuels. Ce glyphe représente une maison dont on peut contrôler les dimensions et la couleur du toit et des murs, ainsi que spécifier le débit de la fumée sortant de la cheminé. Les informations visualisées par les glyphes étant dynamiques, il faut

imaginer qu'une seule instance d'un glyphe du type que nous venons de voir peut passer par tous les états illustrés à la Figure 21.

Le type d'un glyphe doit pouvoir être identifiable instantanément par l'utilisateur car un glyphe est utilisé pour matérialiser un concept dans le monde 3D. Bien que les paramètres visuels du glyphe servent à caractériser une instance de ce concept, il faut prendre soin que leur exploitation n'entraîne pas une déformation si importante du glyphe que l'utilisateur ne puisse plus identifier le type du glyphe (par exemple une maison avec un corps gigantesque et un toit minuscule).

Le glyphe est le composant permettant de définir graphiquement le monde 3D. Les glyphes savent se définir visuellement mais par contre, ils ne savent pas se placer dans le monde 3D. Ce travail de placement est réalisé par un autre composant : le gestionnaire de placement

.



3.2.3 Les gestionnaires de placements (GP)

Les GP sont responsables d'un des indicateurs visuels les plus important dans la construction d'une visualisation 3D efficace : l'exploitation de l'espace. Chaque GP possède des CG fils et son travail est de spécifier la position et l'orientation

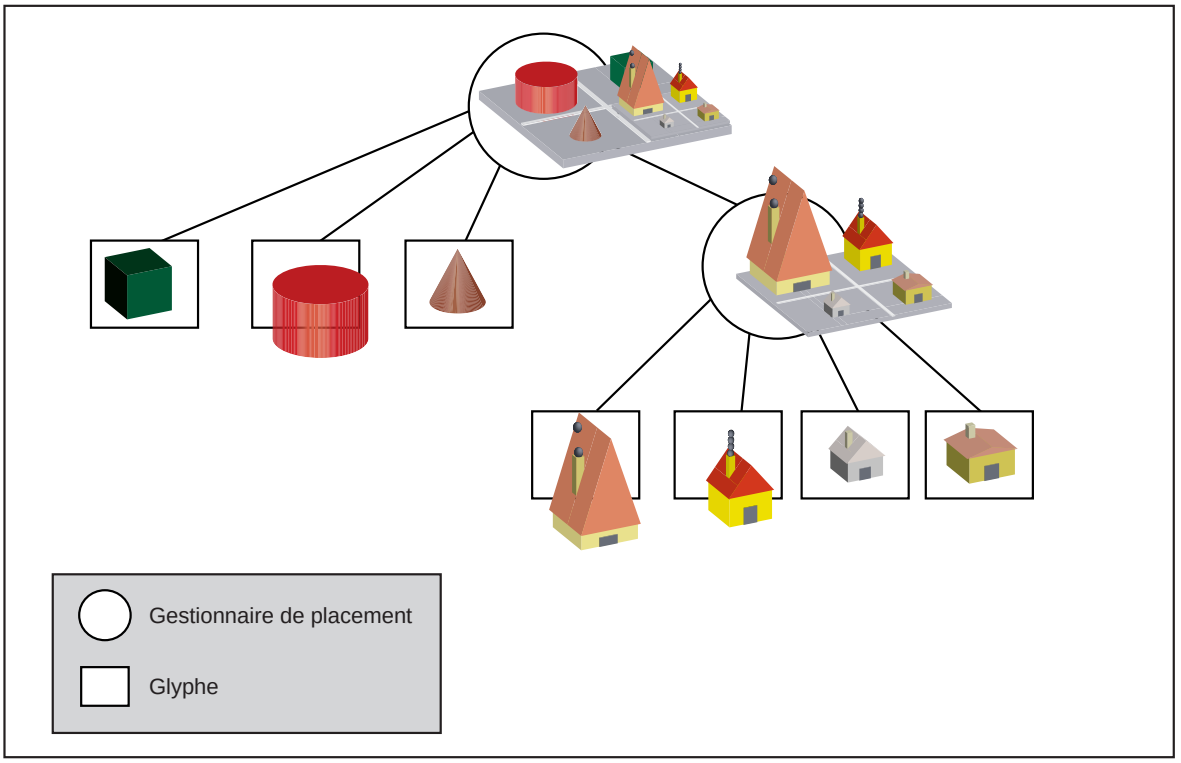
de ses fils. Ce travail est réalisé différemment par chaque GP selon une politique qui lui est propre. Par exemple, un GP peut placer ses fils sur un échiquier comme illustré sur la Figure 21, ou encore empiler ses fils les uns sur les autres. Comme pour les glyphes, il n'y pas de restriction dans la définition de la politique des GP. Ainsi il est possible créer n'importe quel type de visualisation.

Un GP peut également intervenir sur les dimensions de ses fils afin de les contraindre à rester à l'intérieur d'un espace donné. La gestion de l'espace par les GP est étudiée en détail dans la partie suivante.

Un GP contient également un glyphe interne lui permettant d'afficher sa propre représentation visuelle et à améliorer la visualisation. Par exemple, un GP de type échiquier peut utiliser son glyphe interne de façon à matérialiser l'échiquier et ses cases comme c'est le cas dans la Figure 21. Un autre exemple est l'ajout d'une boîte transparente englobant tous les fils du GP de façon à mieux les associer visuellement.

3.2.4 Un monde 3D : une hiérarchie de composants graphiques

L'utilisation de ces deux types de CG permet de construire une hiérarchie de CG définissant un monde 3D, les glyphes créant l'aspect graphique et les GP positionnant les CG dans l'espace. En d'autres termes, un monde 3D est défini par un arbre dont les nœuds internes sont des GP et les feuilles des glyphes (cf. Figure 22).



3.3 Gestion de l'espace 3D

3.3.1 Fonctionnement général

Dans notre modèle de visualisation, le placement de chaque CG (i.e. glyphe ou GP) est effectué par son père GP dans un repère propre à ce GP. Pour placer ses fils dans l'espace, un GP a besoin de connaître les dimensions de chacun de ses fils, c'est à dire l'espace qu'il occupe. Grâce à cette connaissance, il peut décider où placer chacun de ses fils de manière à ce qu'ils soient tous bien visibles dans le monde 3D. Chaque CG calcule lui-même ses dimensions, les glyphes à partir de leurs types et de leurs paramètres visuels et les GP à partir de leurs politiques et/ou des dimensions de leurs fils.

Le calcul des dimensions et des positions de chaque GC dans le monde 3D est initié par le calcul des positions des fils du GP racine. Ce calcul nécessite au préalable que le GP demande les dimensions désirées par chacun de ses fils, ce qui engendre chez les fils GP la même demande auprès de leurs propres fils. Le calcul des positions et des dimensions de tous les CG est donc réalisé itérativement comme illustré à la Figure 23.

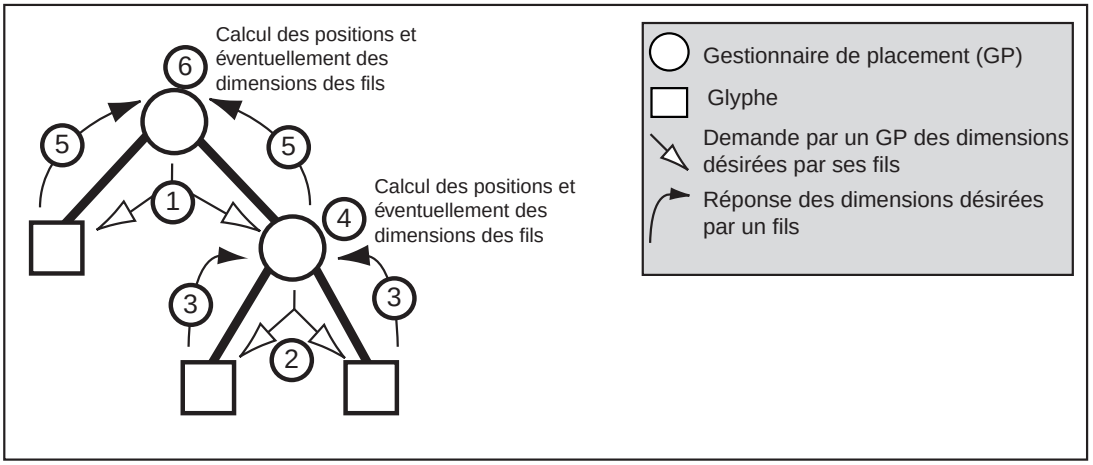
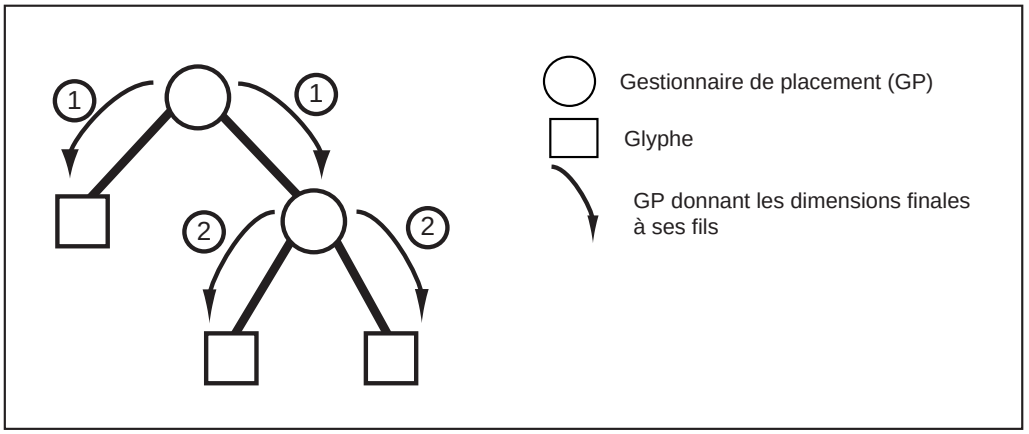


Figure 23 *Première étape du calcul des paramètres du monde 3D*

Une deuxième étape permet au système de communiquer à chaque CG ses dimensions réelles afin qu'il s'adapte à celle-ci. En effet, lors de la première étape, les GP sont tout puissants et peuvent décider de ne pas respecter les dimensions désirées par leurs fils et de leur imposer des dimensions pour des raisons que nous verrons par la suite. Les dimensions réelles de chaque CG ne peuvent donc être calculées qu'au terme de la première étape puisque les dimensions d'un GP peuvent également être modifiées par son propre père, ce qui entraîne un

changement des dimensions des fils de ce GP. Cette deuxième étape est illustrée à la Figure 24.



3.3.2 Gestion de l'espace dans un environnement dynamique

Comme nous l'avons déjà dit, notre modèle de visualisation est utilisé dans un environnement dynamique. La hiérarchie des CG ainsi que leurs paramètres évoluent donc avec le temps, engendrant des changements dans le rendu visuel du monde 3D. Ces changements peuvent intervenir sur tous les types d'indicateurs visuels et peuvent perturber plus ou moins l'utilisateur.

Instabilité du monde 3D

Un des changements visuels le plus perturbant pour l'utilisateur est le changement de position des représentations graphiques (i.e. les CG) du monde 3D. En effet, l'utilisateur peut se trouver désorienté et avoir des difficultés à exploiter les informations visualisées si les éléments graphiques du monde 3D changent constamment de position. Ce phénomène est appelé *instabilité du monde 3D*.

Trois types de modifications dans la hiérarchie décrivant le monde 3D peuvent mener à un changement de position des CG. Ce sont les modifications qui entraînent un changement dans les dimensions de CG déjà existants :

- Les dimensions d'un CG sont changées
- Un nouveau CG est ajouté (son père peut donc changer de taille)
- Un CG existant est détruit (idem)

Une seule modification peut avoir un effet boule de neige sur la position d'un grand nombre de CG. Par exemple, si un CG grandit, le système doit vérifier qu'il ne s'intersecte pas avec d'autres CG. Si une intersection potentielle est détectée, le système doit alors effectuer des modifications de manière à ce que la visualisation soit toujours efficace (il peut par exemple pousser ou changer les dimensions

d'autres CG afin de faire de la place pour le CG grandissant). Ainsi, une modification peut avoir un impact sur les dimensions et la position d'autres CG qui entraîne à son tour un impact sur d'autres CG, etc. Ce phénomène est appelé *propagation de modifications* et il est dans une large mesure responsable de l'instabilité du monde 3D.

Dans la suite de la présentation de la gestion de l'espace 3D au sein de notre modèle, nous indiquons comment les stratégies employés par les GP peuvent permettre de minimiser l'impact de la propagation de modifications sur la stabilité visuelle du monde.

3.3.3 Stratégie de placement d'un GP

Chaque GP possède une stratégie de placement spécifique. Celle-ci définit la façon dont les fils vont être placés et orientés dans le monde 3D. Le système ne pose aucune limite quant à la stratégie de placement d'un GP. Par exemple, un GP peut placer ses fils sur un échiquier (cf. Figure 25), un autre les faire tourner en orbite autour de son centre (cf. Figure 26) et un dernier les empiler les uns sur les autres (cf. Figure 27).

Au sein d'un GP ayant une stratégie donnée, le choix de la place des fils les uns par rapport aux autres n'est pas sans importance. Nous avons dénombré quatre tactiques possibles pour réaliser ce choix :

- Premièrement, on peut utiliser l'ordre dans lequel les fils sont créés, ce qui donne une indication de nature temporelle.
- Deuxièmement, un GP dont tous les fils sont du même type pourra les classer en fonction de la valeur de l'un de leurs paramètres visuels, c'est-à-dire en fonction d'une information qu'ils permettent de visualiser, et ainsi faciliter à l'utilisateur la comparaison entre les fils.
- Troisièmement, un GP peut choisir la place de chacun des fils de manière à optimiser la visualisation de tous les fils. Par exemple, si un GP doit placer des gratte-ciel et une maison, il évitera que la maison se retrouve au milieu des gratte-ciel de façon à ce qu'elle soit visible.
- Quatrièmement, le choix de la place des fils peut être déterminé de manière à limiter l'instabilité visuelle engendrée par la nature dynamique du monde 3D. Pour cela, le GP doit veiller, dans la mesure du possible, à ce que les positions des fils soient constantes dans le temps.

La tactique à choisir est dépendante du contexte dans lequel se trouve le GP et des buts de la visualisation, mais, sauf cas particulier, nous avons généralement choisi d'utiliser la quatrième solution afin de limiter les pertes de repères que peut engendrer le caractère dynamique des mondes 3D.

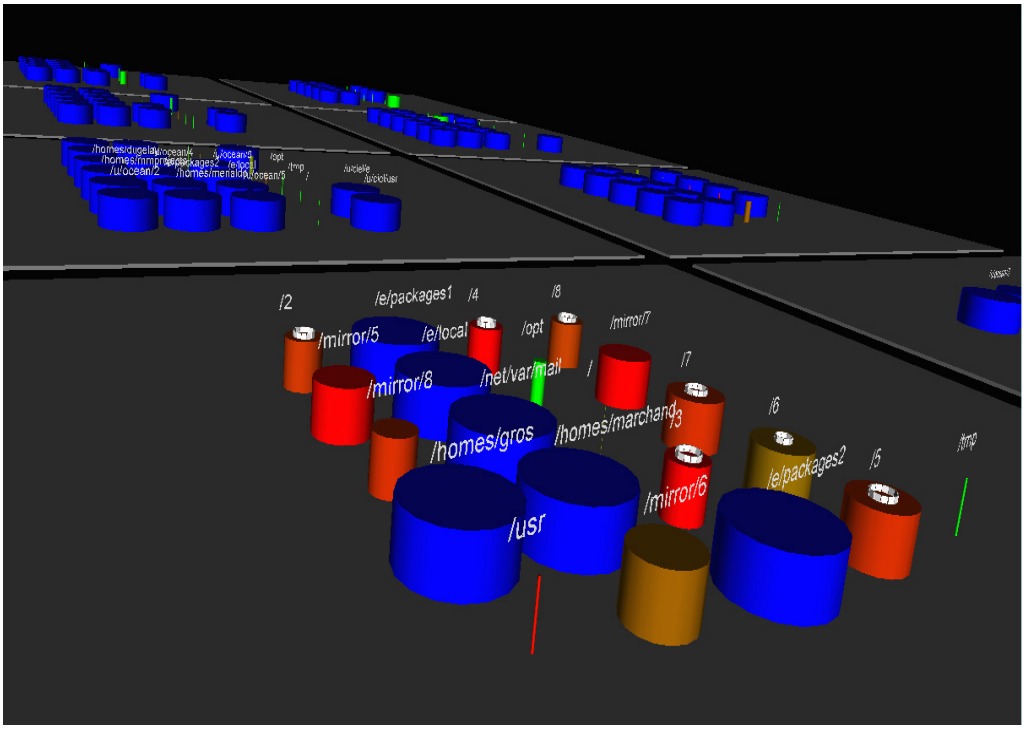


Figure 25 *Stratégie de placement d'un GP de type échiquier*

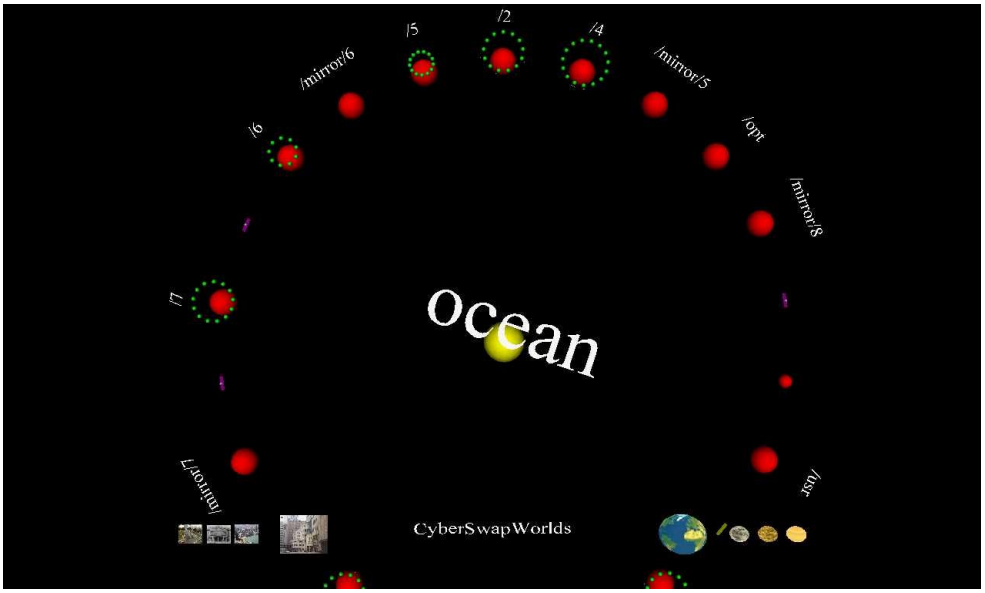


Figure 26 *Stratégie de placement d'un GP de type orbite*

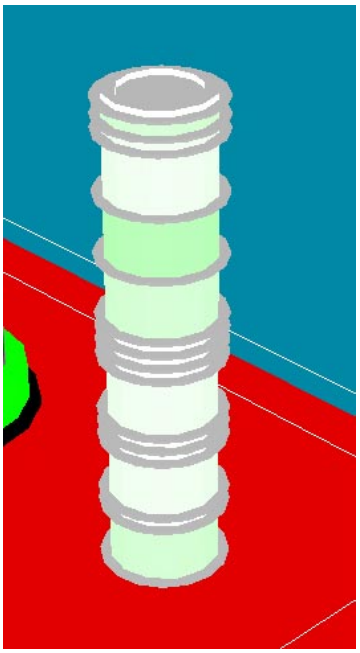


Figure 27 *Stratégie de placement d'un GP de type pile*

3.3.4 Politique des GP sur les dimensions de leurs fils

Outre la stratégie de placement propre à chaque GP, les GP peuvent être classés selon qu'ils respectent ou non les dimensions désirées par leur fils. Dans la majorité des cas, le GP respecte ces dimensions, mais il peut être intéressant que le GP impose des dimensions à ses fils de manière à limiter l'instabilité du monde 3D ou à respecter une autre logique visuelle, comme par exemple que ses propres dimensions ne dépassent un certain seuil.

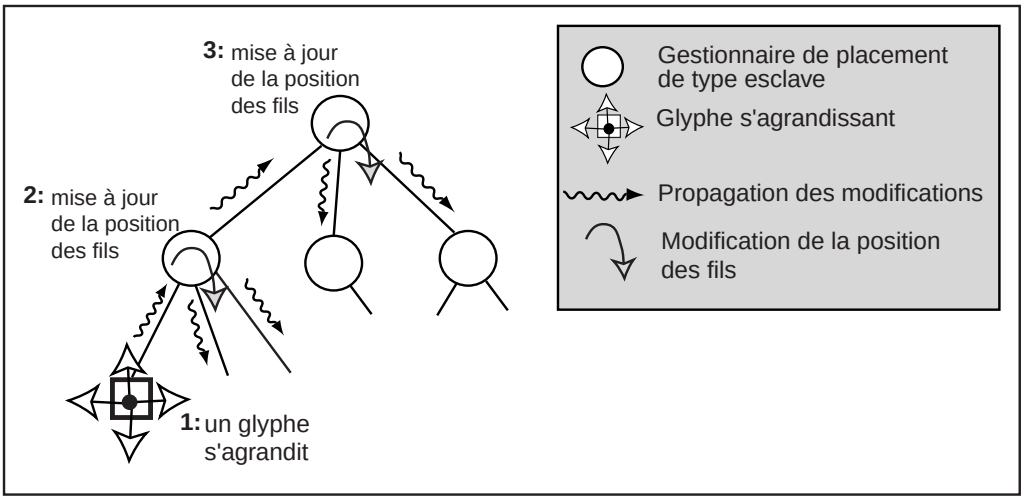
Un GP respectant les dimensions désirées par ses fils est appelé un *GP esclave*, un GP imposant des dimensions à ses fils est appelé *GP maître,* et un GP ne respectant qu'une partie des dimensions désirées par ses fils un *GP hybride*. Nous allons présenter ces trois types de GP en précisant le résultat de leur utilisation dans un environnement dynamique.

3.3.4.1 GP esclave

Un GP esclave respecte les dimensions désirées par ses fils. La conséquence de cette politique est que le changement des dimensions d'un fils peut influencer les positions de tous les autres CG. Par exemple, un changement des dimensions d'un fils d'un GP de type échiquier entraîne l'agrandissement de la taille de la case attribuée à ce fils et donc la translation de tous les autres cases de l'échiquier de façon à ce que celles-ci ne s'intersectent pas. L'espace occupé par ce GP va également augmenter de manière à pouvoir afficher convenablement tous ses fils.

Ce changement de dimensions va être transmis au père de ce GP qui va également déplacer ses fils et ajuster ses dimensions, et ainsi de suite (cf. Figure 28).

Un GP esclave peut donc entraîner une instabilité du monde 3D en propageant vers son père et ses fils les modifications qu'il a subies. Ce type de GP est à l'origine d'une grande partie de l'instabilité dans le monde 3D, mais il est incontournable car il conserve les dimensions désirées par les fils qui représentent souvent des informations primordiales.



3.3.4.2 GP maître

Une solution au problème d'instabilité engendré par l'utilisation de GP esclaves est d'utiliser un GP maître. En effet, le GP maître impose des dimensions à chacun de ses fils. Il peut donc, par exemple, faire en sorte que ses propres dimensions ne varient pas même si le nombre de ses fils ou leurs dimensions changent. Si les dimensions du GP ne changent pas, la propagation des modifications observées dans l'utilisation de GP esclaves se réduit aux fils de ce GP(cf. Figure 29).

L'avantage des GP maîtres est donc leur capacité à produire des mondes 3D plus stables au niveau de la position des CG puisque les pères (i.e. les GP) sont tout-puissants sur les dimensions de leurs fils et donc sur la position de ceux-ci. Par contre, l'utilisation de cette stratégie mène à des mondes où il est possible d'avoir de graves pertes de cohérence au niveau des dimensions relatives de chaque CG. En effet, puisque c'est le père qui alloue l'espace à ses fils, deux glyphes à l'origine identiques peuvent avoir une taille finale différente. Il en résulte une détérioration de la visualisation pour certaines tâches comme la comparaison visuelle des dimensions de deux objets. L'utilisation de cette technique est donc sujette à quelques précautions, mais elle peut se révéler indispensable en particulier associée à la stratégie de type esclave.

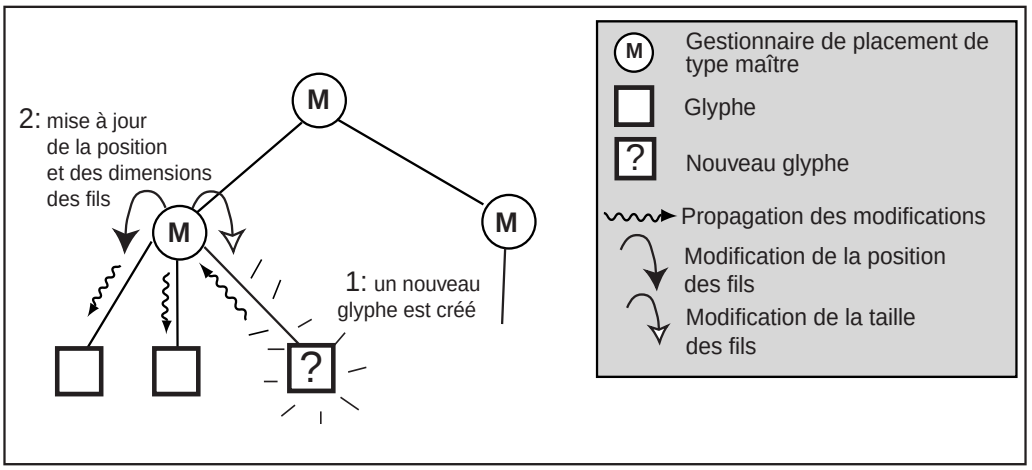


Figure 29 *Résultat de l'ajout d'un nouveau glyphe avec des GP maître*

3.3.4.3 GP hybride

Un GP hybride est un GP respectant en partie les dimensions désirées par ses fils. Prenons un exemple. Dans une métaphore de ville, un quartier peut être représenté par un GP qui place les différents bâtiments du quartier sur une surface plane. L'utilisation d'un GP maître peut permettre de fixer les dimensions du quartier dans le temps : une fois le quartier construit, celui-ci ne changera plus de dimensions, et l'ajout de nouveaux bâtiments nécessitera de réduire les dimensions de ceux déjà existants. Il peut donc être intéressant d'utiliser un GP maître de façon à garder fixe les dimensions d'un quartier, mais il peut être également intéressant de ne pas limiter les dimensions des bâtiments en hauteur. Pour obtenir ce résultat, le GP doit imposer la largeur et la longueur aux bâtiments mais respecter la hauteur demandée par chaque bâtiment. Ce GP est un exemple de GP hybride. Le GP hybride peut être vu comme un cas particulier de GP maître qui calcule les dimensions qu'il va imposer à ses fils en respectant en partie les dimensions désirées par ses fils.

Comme pour le GP maître, le fait d'imposer une taille à ses fils peut entraîner une perte de cohérence au niveau de la taille relative de chaque CG dans le monde 3D, et des opérations comme la comparaison visuelle de la taille de deux objets du même type ne peut plus être réalisée.

3.3.5 Utilisation des différents types de GP

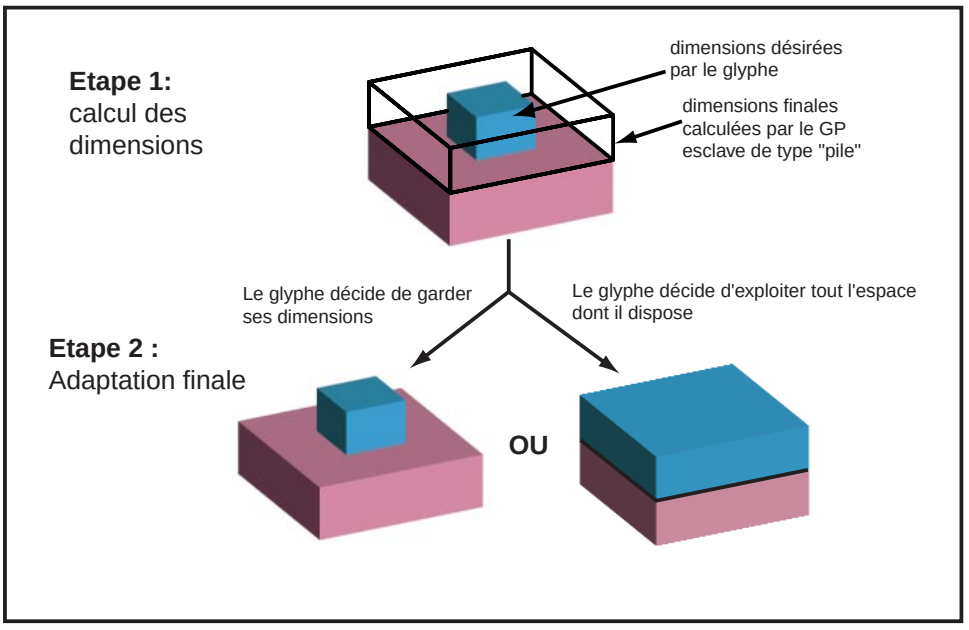
Comme nous l'avons vu, les GP maîtres, esclaves et hybrides ont leurs avantages et leurs inconvénients. La décision d'utiliser une stratégie plutôt qu'une autre dépend du résultat visuel voulu. Mais il est évident que ces trois types de GP peuvent être utiles à l'intérieur d'un même monde, les GP esclaves permettant aux CG d'obtenir leurs dimensions désirées mais engendrant de l'instabilité dans le monde 3D, le GP maître permettant de stabiliser certaines partie du monde, et le GP hybride permettant d'associer ces deux stratégies au sein d'un même GP.

3.3.6 Adaptation aux dimensions finales

Dans le fonctionnement général de la gestion de l'espace, nous avons vu que le système permet aux CG de s'adapter à leurs dimensions finales en obligeant les GP à envoyer celles-ci à tous leurs fils (cf. Figure 24). On pourrait penser que cette étape est inutile car les GP pourraient prendre en charge le changement des dimensions de leur fils sans les avertir. Mais comme nous allons le voir, cette deuxième étape permet aux CG de mieux s'adapter et de gérer leur apparence ou d'optimiser l'espace dont ils disposent dans le monde 3D.

La connaissance de ses dimensions finales permet à un CG d'optimiser son apparence. Un exemple est l'utilisation de représentations géométriques plus ou moins précises. En effet, il n'est pas utile d'utiliser une représentation géométrique très précise si celle-ci est vouée à être confinée à un espace très restreint. Les CG connaissant leurs dimensions finales peuvent exploiter cette information de manière à choisir leur représentation géométrique en fonction de l'espace dont il dispose.

Dans un autre contexte un GP peut, de part son fonctionnement, réserver pour un fils un espace supérieur aux dimensions désirées par celui-ci. Par exemple un GP esclave de type « pile » peut réserver des dimensions pour chacun de ses fils dont la largeur et la longueur correspondent à celles du plus grand fils, le GP considérant que sa propre géométrie est définie par un parallélépipède (cf. Figure 30). Dans ce cas, un fils du GP sachant qu'il dispose de plus d'espace que prévu peut décider de s'agrandir. Ce type de situation peut donc permettre à un CG de s'adapter à son environnement. Mais cette décision n'est pas automatique, et tous les CG informés ne changeront pas leurs dimensions, en particulier si celles-ci reflètent une information.



Un résumé de toutes les étapes, dont celle de l'adaptation finale, permettant la construction d'un monde 3D, est présenté dans le Tableau 8.

	GP Esclave	**GP Maître ou hybride**
Etape 1	**1-**Demande les dimensions que ses fils désirent.	
	2-Positionne chacun de ses fils en respectant les dimensions qu'ils désirent.	**2-**Calcule les dimensions de ses fils selon ses propres critères, puis les positionne.
	3-Calcule ses dimensions à partir de celles de ses fils et de sa politique.	
Etape 2	**4-** Si son père n'a pas respecté ses dimensions, le GP doit recalculer la taille réelle de ses fils. **5-** Donne les dimensions finales à ses fils afin qu'ils s'adaptent.	

Tableau 8 *Résumé du comportement des GP lors de la construction ou la mise à jour du monde 3D*

4 Exemples d'utilisation du modèle de visualisation

4.1 Arbre de cônes

La Figure 31 illustre la façon dont le modèle de visualisation permet d'implémenter très simplement un type de visualisation 3D d'informations hiérarchiques très connu : l'arbre de cônes [Robertson91].

Un seul type de GP est nécessaire pour réaliser cette visualisation. Ce GP, de type esclave et appelé tout simplement « ConeTree », possède un glyphe interne représentant un cône. Les dimensions du cône sont déterminées en fonction des dimensions de ses fils de façon à ce qu'il soit possible de les placer à la base du cône sans qu'ils se touchent. Ces fils peuvent être d'autres GP de type « ConeTree » ou bien alors des glyphes représentant des cylindres dont le seul paramètre visuel est la couleur.

Pour définir cette visualisation, il a donc seulement été nécessaire de spécifier le GP de type « ConeTree » et le glyphe de type cylindre. Bien sûr, cette

visualisation pourrait être bien plus riche. Par exemple, le glyphe cylindre pourrait posséder des paramètres visuels supplémentaires (e.g. la taille), et d'autres types de glyphe pourraient être présents. Il pourrait être également intéressant d'expérimenter l'utilisation d'un GP ayant la même stratégie (i.e. coneTree) mais appliquant une politique de type maître qui change la taille de ses fils de manière à remplir un espace donné. Ainsi, en construisant une hiérarchie de CG où le GP racine serait de type maître et tous les autres esclaves, on pourrait obtenir une visualisation ne changeant pas de taille quel que soit le nombre d'informations à visualiser.

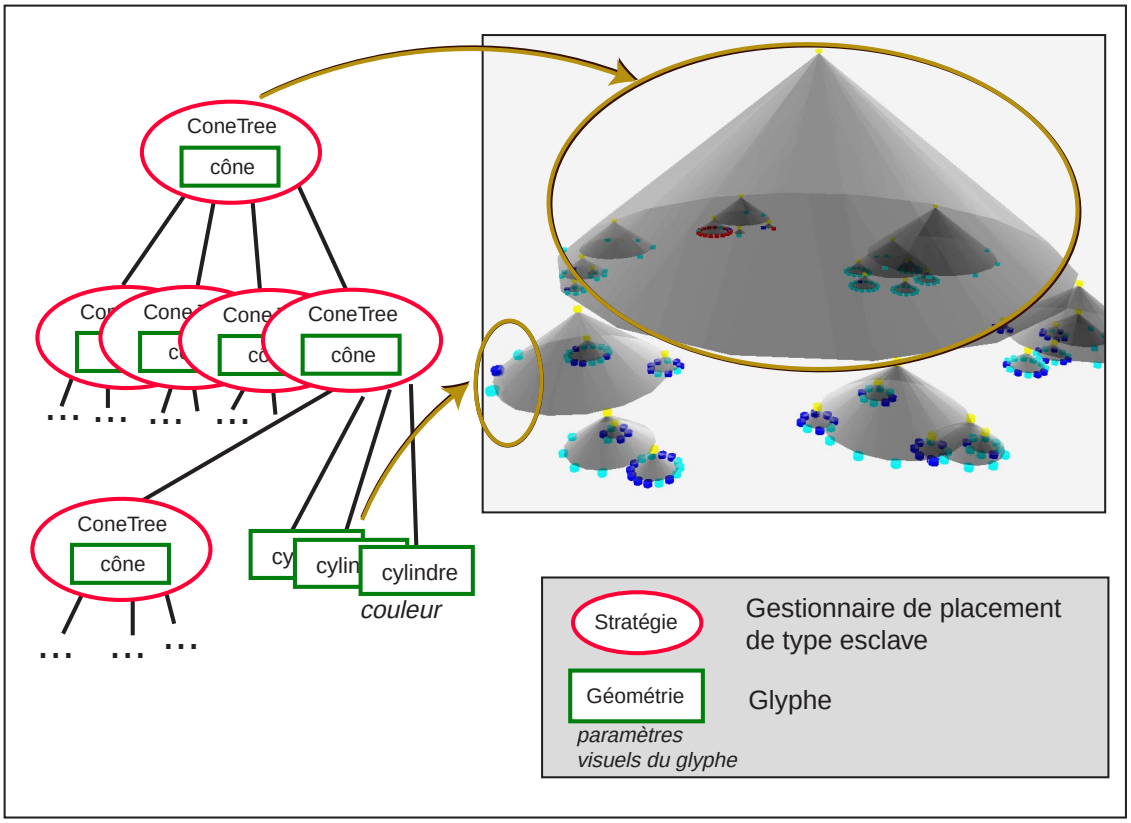


Figure 31 *Hiérarchie de composants graphiques utilisés pour une visualisation de type arbre de cônes*

4.2 La ville

La Figure 32 et la Figure 33 illustrent l'utilisation du modèle de visualisation pour la construction de mondes 3D basés sur des métaphores de ville. Ces métaphores sont toutes deux constituées de quartiers contenant des bâtiments plus ou moins sophistiqués, ainsi que des arbres pour la deuxième.

Les glyphes et les GP utilisés sont basiques, et c'est leur combinaison qui permet de produire une métaphore relativement riche. Les glyphes sont principalement des boîtes ou des cylindres avec quelquefois des éléments géométriques supplémentaires comme, par exemple, un drapeau sur le haut d'une

boîte ou un socle à la base. Les stratégies des GP sont de type pile ou échiquier, excepté le GP « Etage » de la Figure 33 qui place un ensemble de fenêtres sur les murs d'un étage.

Les GP appliquent tous une politique du type esclave, c'est à dire qu'ils respectent les dimensions que leurs fils ont demandées, excepté le GP « Fenêtres » du deuxième exemple (Figure 33). Celui-ci applique une politique du type maître de manière à ce que les fenêtres d'un étage couvrent juste celui-ci. Ainsi lorsque le nombre de fenêtres est grand, le GP réduit la taille des fenêtres de manière à ce qu'elles ne dépassent pas sur les autres étages. En gérant la taille des fenêtres, on peut donc avoir des étages ayant tous la même taille.

Les glyphes « trottoir » de la première métaphore exploitent la possibilité donnée par le système de s'adapter à leur environnement (cf. 3.2.6). Ainsi les dimensions finales du trottoir correspondent à la taille que lui a réservée le GP « quartier ». On peut donc voir sur la Figure 32 que les dimensions des trottoirs correspondent au nombre de bâtiments présents dans le quartier, contrairement à la Figure 33 où la taille d'un quartier est fixée.

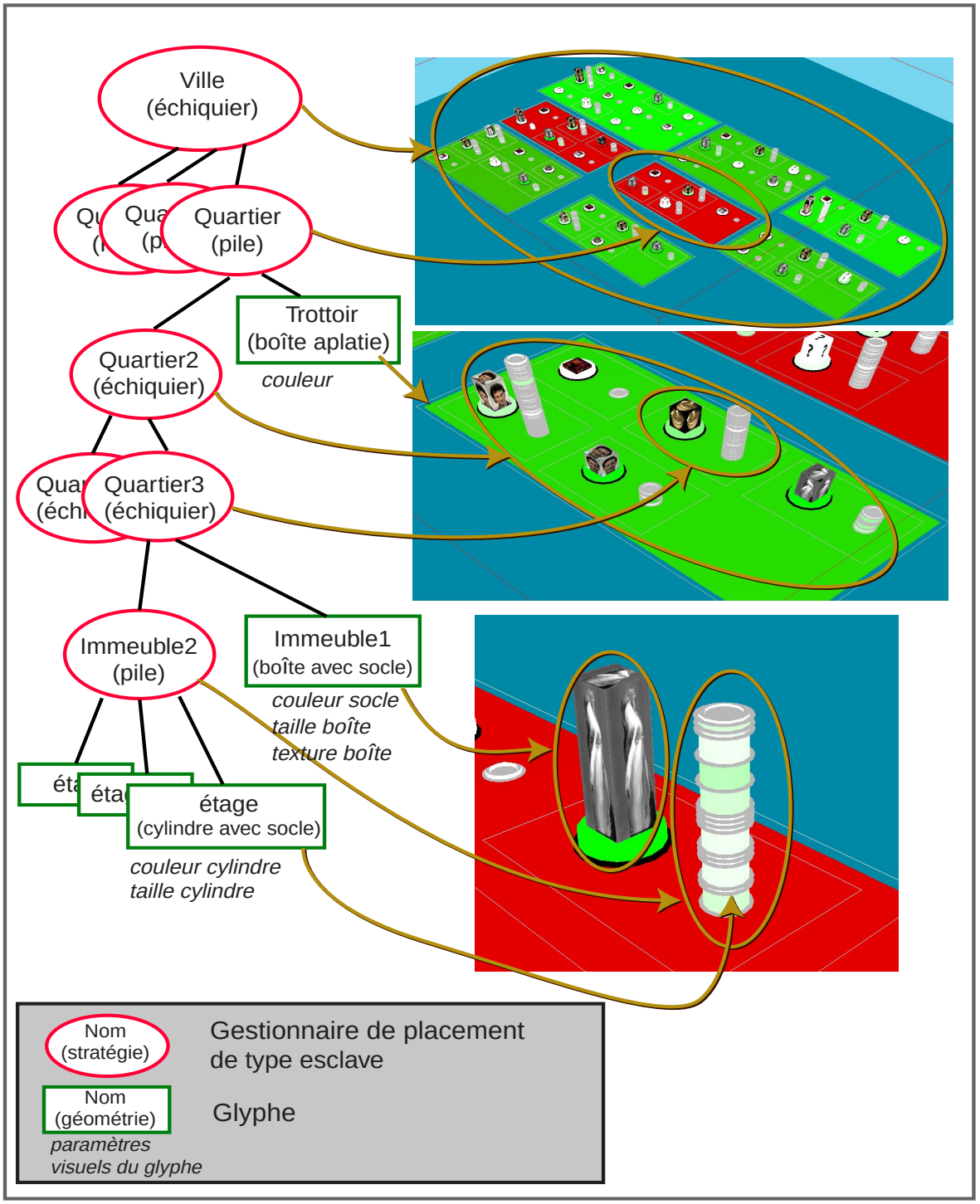


Figure 32 *Hiérarchie de composants graphiques décrivant une ville*

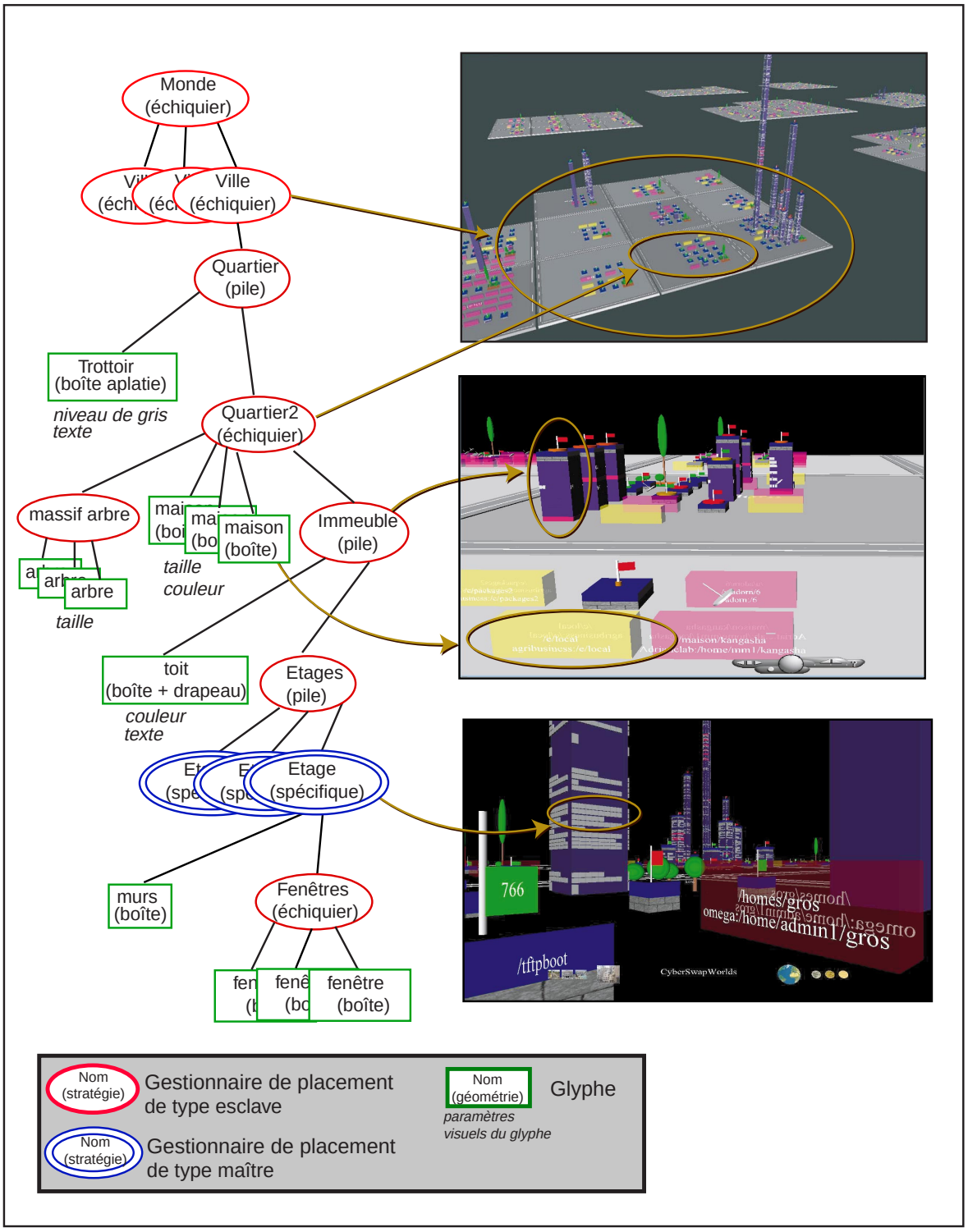


Figure 33 *Hiérarchie de composants graphiques décrivant une ville*

5 Conclusion

Le modèle de visualisation que nous avons présenté permet de définir des mondes 3D par une hiérarchie de composants graphiques (CG). Les CG se divisent en deux types : d'une part les gestionnaires de placements (GP) qui permettent de gérer l'espace 3D, et d'autres part les glyphes qui constituent la partie graphique d'un monde 3D. La conception orientée objet des CG permet leur réutilisation et de créer rapidement de nouveaux type de visualisation. Cette approche permet également de faciliter l'automatisation de la construction d'un monde 3D.

Un algorithme permettant de calculer la position et l'apparence de chaque CG d'un l'arbre décrivant un monde 3D a été également exposé. Cet algorithme s'applique en deux étapes de manière à permettre plus de souplesse dans la création des visualisations, notamment pour la gestion des dimensions de chaque CG.

Nous avons également présenté la prise en compte par notre modèle du problème d'instabilité visuelle lié au caractère dynamique de nos visualisations. Ce problème a été abordé en définissant des GP dont la politique est d'intervenir sur les dimensions des CG qu'ils doivent placer dans l'espace afin de limiter l'instabilité du monde 3D.

Différents types de visualisation ont été implémentés grâce à ce modèle pour prouver que celui-ci pouvait être applicable dans de nombreuses situations. En plus de ceux présentés dans ce chapitre, on pourra trouver ces exemples dans le chapitre 7 concernant les expérimentations qui ont été réalisées au cours de cette thèse.

Bien qu'il puisse l'être, ce modèle n'a pas pour but d'être utilisé directement. Dans le chapitre 5, nous verrons comment il est intégré dans un modèle plus général de mondes 3D interactifs servant de base à la construction des visualisations utilisées au sein du projet CyberNet.

4 Navigation & autres interactions avec des mondes 3D

1 Introduction

De même que l'homme possède différents moyens pour se déplacer dans le monde réel, il est nécessaire de fournir des outils permettant à un utilisateur confronté à un monde 3D de naviguer dans celui-ci. Mais contrairement à la réalité, il n'y a aucune contrainte physique dans la spécification de ces outils et, par exemple, la téléportation dont nous rêvons tous les jours est une réalité dans les mondes 3D. Les outils de navigation se doivent donc d'exploiter les possibilités offertes par les mondes virtuels sans se contenter d'imiter celles existant dans le monde réel.

Un des problèmes des techniques de navigation actuelles est le risque relativement élevé qu'elles comportent de perdre ses repères dans l'espace, ainsi qu'une certaine difficulté à réaliser les déplacements voulus. Dans ce chapitre, nous présentons un modèle de navigation permettant d'éviter ces problèmes et d'exploiter les libertés offertes par les mondes 3D.

Notre approche est d'essayer de décharger l'utilisateur de tout ce qui peut être fait automatiquement et ainsi d'optimiser ses tâches de navigation. Pour cela, notre modèle de navigation s'appuie sur la définition de *centres d'intérêt* dans les mondes 3D, d'outils pour les observer, et de mécanismes pour naviguer automatiquement entre eux. De plus, nous mettons en avant les avantages d'offrir des outils qui s'adaptent à la structure du monde 3D et des objets le composant, et non des outils génériques.

Nous présentons plus brièvement d'autres interactions, non liées à la navigation mais permettant également de faciliter l'exploitation d'un monde 3D dont le but est la visualisation d'informations.

Une partie de ce chapitre est réservée à l'utilisation du modèle de navigation et d'autres interactions conjointement au modèle de visualisation décrit au chapitre précédent, ceci en vue de la présentation au prochain chapitre de notre modèle plus général de mondes 3D interactifs.

Ce chapitre est organisé de la façon suivante : tout d'abord nous présentons des techniques de navigation déjà existantes, puis nous exposons notre modèle de navigation, pour ensuite étudier comment il peut s'intégrer dans le modèle de visualisation présenté au chapitre précédent. Pour finir, nous présentons quelques exemples d'autres types d'interactions.

2 Techniques de navigation 3D

Plusieurs types de recherches ont été réalisés afin de faciliter la navigation dans des mondes 3D. Une partie de ces travaux cherchent à définir des principes de structuration des mondes 3D permettant à l'utilisateur de ne pas se perdre et de mieux s'orienter. Par exemple, [Ingram95] a défini des principes de structuration issus des travaux d'un urbaniste sur la structure des villes[Lynch60].

Dans cette partie, nous ne nous intéressons pas à la construction d'un monde 3D facilitant la navigation, mais aux outils permettant à l'utilisateur de naviguer dans un monde 3D déjà construit. Nous présentons tout d'abord des outils permettant à l'utilisateur d'avoir une connaissance spatiale du monde 3D dans lequel il évolue, puis des outils permettant à l'utilisateur de se déplacer dans ce monde. Ensuite nous présentons des outils de navigation contraignant l'utilisateur dans ses déplacements, et le concept de point de vue.

2.1 Connaissance spatiale du monde 3D

[Thorndyke83] a classifié les connaissances spatiales en trois classes :

- *Les points de repère* (par exemple un monument), qui permettent à l'utilisateur de connaître sa position ; ce principe rejoint les travaux des chercheurs qui cherchent à définir des principes de structuration aidant l'utilisateur à s'orienter
- *La connaissance des routes*, c'est à dire la connaissance des relations spatiales entre les différents éléments graphiques constituant le monde 3D
- *La connaissance générale* du monde 3D qui s'obtient lorsque l'utilisateur a une bonne connaissance des routes

Afin d'aider l'utilisateur à obtenir ces connaissances spatiales, différents outils ont été étudiés. L'idée principale est de donner à l'utilisateur un retour visuel sur sa position globale dans le monde. La solution la plus simple est de donner à l'utilisateur sa position en termes de coordonnées, mais cette solution n'est pas d'une grande aide à moins que l'utilisateur soit un grand géographe du monde 3D. Des solutions plus élaborées se basent sur l'affichage d'une vue globale (ou locale si le monde est trop grand) et simplifiée du monde 3D. [Stoackley95] présente par exemple son concept de « monde en miniature » et [Satalich95] a étudié comment une vue 2D peut aider les utilisateurs à naviguer dans des bâtiments.

[Edwards97] a présenté un outil constitué d'une carte 2D du monde 3D dans laquelle la position de l'utilisateur est spécifiée, cette carte étant affichée en surimposition dans la vue 3D. Cette technique se révèle très efficace pour que l'utilisateur ne se perde pas. [Edwards97] a également proposer un outil permettant à l'utilisateur de créer lui-même des points de repère dans le monde 3D.

Dans [Darken93][Darken96], on trouve des études sur l'utilisation des cartes et des grilles de navigation dans les mondes virtuels, en parallèle avec des travaux sur des principes de structuration des mondes 3D aidant l'utilisateur à s'orienter.

2.2 Déplacement dans le monde 3D

On peut trouver une description complète des différents travaux réalisés sur les outils de navigation dans un monde 3D dans [Hand97]. Ces outils peuvent être classés selon qu'ils sont égocentriques (l'utilisateur se déplace dans le monde) ou exocentriques (le monde se déplace devant l'utilisateur).

La plupart des outils de déplacement sont égocentriques. On les retrouve dans la majorité des navigateurs 3D comme les navigateurs VRML. Les outils permettant d'effectuer des *mouvements généraux* [Mackinlay90] autorisent l'utilisateur à naviguer facilement dans le monde 3D. Une partie de ces outils permettent à l'utilisateur de se déplacer dans un plan spécifique. Par exemple, l'utilisateur peut se déplacer dans un plan parallèle ou perpendiculaire à son corps virtuel. Un autre outil permet d'effectuer une rotation sans changer sa position, comme si l'utilisateur tournait la tête. D'autres outils permettent d'effectuer des *mouvements cibles* [Mackinlay90] afin se rendre directement sur un objet spécifique du monde 3D, qui peut être déterminé par une sélection directe dans le monde 3D.

Un outil exocentrique est la sphère virtuelle [Chen88] permettant d'examiner en détail un objet en faisant tourner le monde devant l'utilisateur. Ce type d'outil fait également parte intégrante des navigateurs VRML mais, étant exocentrique, il est plus utile pour visualiser un monde 3D contenant un seul objet. Le concept de la sphère virtuelle peut également être utilisé d'une manière égocentrique (i.e. c'est l'utilisateur qui tourne autour de l'objet et non l'inverse).

2.3 Navigation contrainte

Permettre à l'utilisateur de se déplacer librement dans le monde 3D est important, mais la plupart des personnes ayant utilisé les outils permettant d'effectuer les mouvements généraux que nous avons vus précédemment peuvent confirmer qu'il est très facile de se retrouver perdu dans le monde 3D suite à l'utilisation de ces outils. Des recherches ont été effectuées de façon à fournir de nouveaux outils contraignant ou automatisant les mouvements de l'utilisateur en fonction de ses buts.

[Kaye97] présente un concept permettant de modifier automatiquement la direction de la caméra de manière à suivre un objet spécifique dans le monde. [Hanson97] présente un outil qui contraint les paramètres de la caméra en fonction de l'environnement : la position de la caméra est limitée à une surface déterminée, et son orientation est contrainte en fonction des objets qui l'entoure. Cet outil a été testé dans un contexte de navigation sur un terrain, mais il faut retenir que cet outil est dépendant du contexte avoisinant la caméra et non générique comme les outils que nous avons vus auparavant.

[Roberston00] propose également des outils de navigation contrainte réalisés dans le cadre d'un prototype permettant d'étendre à la 3D la célèbre métaphore du bureau. Dans ce prototype, le bureau devient une pièce et la navigation s'effectue à partir de simples actions comme avancer d'un cran, regarder à gauche ou aller à une vue globale, chacune de ces actions entraînant le déplacement automatique de l'utilisateur et/ou de son regard vers un endroit précalculé.

2.4 Les points de vue

Un point de vue détermine une position, une orientation et une focale spécifiques de la caméra représentant le regard de l'utilisateur sur la scène 3D. Ce concept est très pratique car en utilisant un outil adéquat, l'utilisateur peut se rendre directement à un point de vue. Le mouvement menant à un point de vue est appelé *mouvement aux coordonnées spécifiées* [Mackinlay90].

Dans le langage VRML, il est par exemple possible de spécifier des points de vue associés à un nom, qui sont ensuite exploités par les navigateurs VRML afin de proposer à l'utilisateur la liste de ces points de vue.

3 Modèle de navigation pour la visualisation d'informations en 3D

La navigation est un des problèmes principaux auquel est confronté l'utilisateur lorsqu'il interagit avec un monde 3D. La navigation libre, telle que nous venons de la voir, mène la plupart du temps à une perte de repères désignée par l'expression anglaise « lost in hyperspace » et ne facilite pas une navigation dirigés par des buts comme « aller à cet endroit ». Il est donc indispensable de fournir de nouveaux outils permettant à l'utilisateur de naviguer avec efficacité et sans se perdre dans le monde 3D, sous peine que celui-ci ne soit pas exploité à sa juste valeur.

Notre modèle de navigation a été pensé de manière à permettre l'exploitation de mondes 3D destinés à visualiser des informations, et non pour n'importe quel type de monde 3D, bien que nous pensions que ce modèle peut être utilisable dans d'autres situations. De plus, le modèle ne prend pas en compte les outils qui permettent à l'utilisateur d'améliorer ses connaissances spatiales du monde 3D, par exemple en utilisant une carte, mais se focalise sur les moyens de se déplacer

dans le monde 3D pour visualiser les informations, en particulier en utilisant des moyens de navigation contraints.

Notre modèle est basé sur les principes suivants qui seront détaillés dans cette partie :

- Définition automatique des centres d'intérêt d'un monde 3D
- Possibilité pour l'utilisateur de sélectionner un centre d'intérêt existant afin de l'étudier visuellement
- Définition de listes de centres d'intérêt spécifiques et de méthodes pour les consulter
- Mise à disposition de l'utilisateur d'un moyen d'observer facilement un centre d'intérêt sous toutes ses coutures
- Définition des méthodes de transition entre les centres d'intérêt

3.1 Les centres d'intérêt (CDI)

3.1.1 Définition

Le but de nos visualisations est de permettre à l'utilisateur de superviser et d'analyser des informations. Pour cela, l'utilisateur porte son attention sur la partie du monde 3D qui contient ces informations. Dans le cas extrême où l'utilisateur désire avoir une vue d'ensemble des informations visualisées, il porte son attention sur la totalité du monde 3D.

Les parties du monde sur lesquelles l'utilisateur est susceptible de porter son attention sont appelées *centres d'intérêt (CDI)*. Une partie du monde est constituée d'un ensemble d'éléments graphiques. Par exemple, dans une métaphore de ville, l'utilisateur peut avoir comme CDI un bâtiment, un quartier constitué de plusieurs bâtiments ou encore la ville tout entière constituée de tous les quartiers.

Les CDI sont au cœur de notre modèle de navigation, et pour que celui-ci soit efficace, il est nécessaire que le système connaisse tous les CDI susceptibles d'intéresser l'utilisateur, puis qu'il fournisse des outils permettant leur exploitation.

3.1.2 La navigation comme moyen de consultation des CDI

Durant l'utilisation d'une visualisation d'informations en 3D, l'utilisateur change couramment de CDI afin de visualiser différentes informations. Les mouvements effectués pour réaliser ces changements correspondent à une partie des tâches de navigation de l'utilisateur. Ces mouvements sont complexes à réaliser en navigation libre, et nous avons donc voulu les automatiser afin d'optimiser la consultation des CDI, et donc des informations visualisées dans le monde 3D.

Une autre tâche de navigation de l'utilisateur consiste à d'observer un CDI particulier (par exemple en tournant autour) . Encore une fois, cette tâche n'est pas facile à réaliser avec des outils de navigation standard, notre système de navigation propose donc des moyens spécifiques pour y parvenir.

3.1.3 Structuration des CDI

Pour bien fonctionner, le système de navigation doit pouvoir déterminer tous les CDI possibles et les organiser hiérarchiquement vis-à-vis de la logique de la représentation visuelle. Par exemple, dans une visualisation basée sur une métaphore de ville, les CDI associés aux bâtiments sont les fils du CDI du quartier les contenant, qui est lui-même fils du CDI « ville » (cf. Figure 34). Certains moyens de sélection d'un nouveau CDI s'appuient sur cette hiérarchie, et il est donc important que celle-ci soit bien définie sous peine de rendre les outils de sélection inopérants.

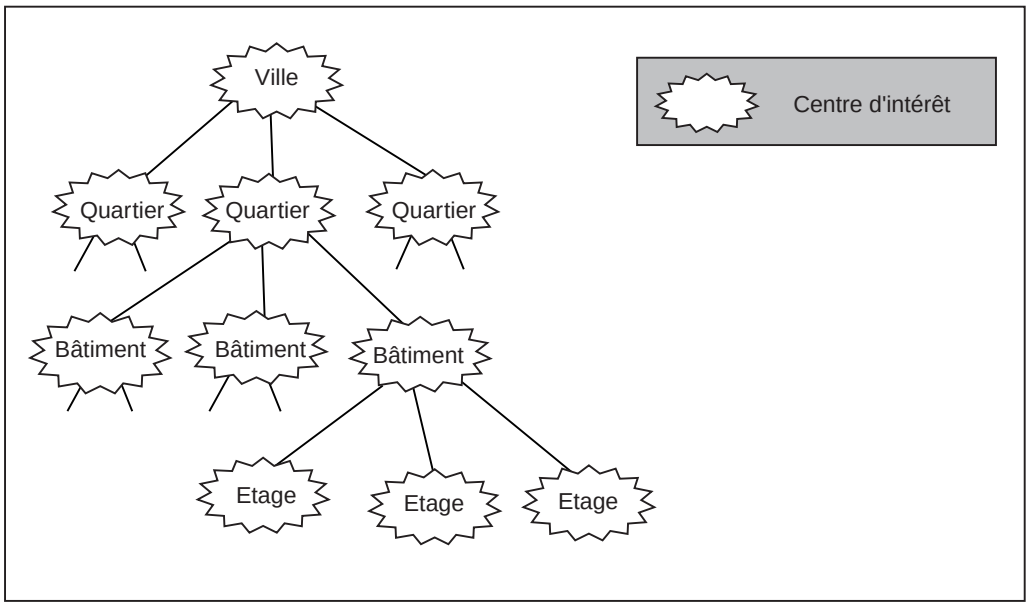


Figure 34 *Exemple d'une hiérarchie de CDI dans une visualisation basée sur une métaphore de ville*

3.1.4 Un point de vue pour chaque centre d'intérêt

Lors de l'utilisation d'un monde 3D, l'utilisateur regarde toujours depuis un certain emplacement dans une direction donnée. Un *point de vue* est l'association d'un emplacement et d'une orientation de la caméra représentant le regard de l'utilisateur. Afin de faciliter l'observation d'un CDI, notre système calcule dynamiquement un point de vue lié à ce CDI, de manière à permettre à l'utilisateur de visualiser la représentation graphique associée à ce CDI d'une façon satisfaisante.

Les représentations graphiques associées à chaque CDI pouvant être différentes en taille et en forme, les points de vue ne sont pas définis d'une façon générique mais en fonction de cette représentation graphique et du contexte dans lequel elle se trouve. Par exemple, un simple cube peut être utilisé pour symboliser une maison ou l'étage d'un immeuble. Dans le premier cas, le point de vue peut être situé au-dessus du cube, alors que dans le deuxième cas cela pourrait générer un point de vue inutilisable car situé à l'intérieur des étages supérieurs.

La position de l'utilisateur pour observer un CDI est primordiale. L'utilisation du point de vue associé à chaque CDI est un moyen sûr de lui permettre de visualiser correctement les CDI mais il serait dommage de limiter l'utilisateur à ce point de vue. Nous verrons par la suite qu'un mécanisme d'observation est disponible pour chaque CDI, de manière à ce que l'utilisateur puisse choisir le regard qu'il porte sur le CDI courant.

3.1.5 Un CDI d'attachement pour éviter à l'utilisateur de se perdre

Les utilisateurs des mondes 3D souffrent généralement du syndrome désigné par l'expression « perdu dans l'hyperespace ». Le modèle de navigation veille à ce que cette situation ne survienne pas. Mais parce qu'il propose également des moyens de navigation plus traditionnels, il peut arriver que l'utilisateur se perde. Dans ce cas, il est alors très utile de lui donner la possibilité de revenir au dernier endroit où l'on pense qu'il n'était pas perdu. Dans notre système, cet endroit est matérialisé par le point de vue du dernier CDI que l'utilisateur a sélectionné, ce CDI est appelé *CDI d'attachement*. Ainsi, si l'utilisateur se perd ou pour n'importe quelle autre raison, il peut revenir en un seul clic au point de vue associé au CDI d'attachement.

3.2 Sélection du centre d'intérêt

Dans notre système l'utilisateur navigue de CDI en CDI. La sélection d'un CDI est donc cruciale pour que l'utilisateur puisse bien explorer le monde 3D et donc bien visualiser les informations. Nous avons défini plusieurs méthodes de sélection d'un nouveau CDI. Ces méthodes peuvent être classés selon que la sélection est absolue ou bien relative au CDI courant.

3.2.1 Sélection absolue

Il existe deux types de sélection absolue. La première est réalisée en choisissant un CDI sous forme de nom décrivant la nature visuelle de celui-ci (e.g. Quartier Toto) ou décrivant les informations visualisées par ce CDI (Utilisateur Toto). Dans le second cas, on pourra donc choisir, dans une visualisation d'un parc d'ordinateurs et de leurs utilisateurs, parmi tous les noms des ordinateurs et des utilisateurs présents. La présentation des noms se fait par une interaction capable de visualiser une hiérarchie correspondant à celle des CDI, comme par

exemple un menu hiérarchique (cf. Figure 35) mais on peut aussi imaginer un arbre hyperbolique ou même une représentation 3D de cette hiérarchie.

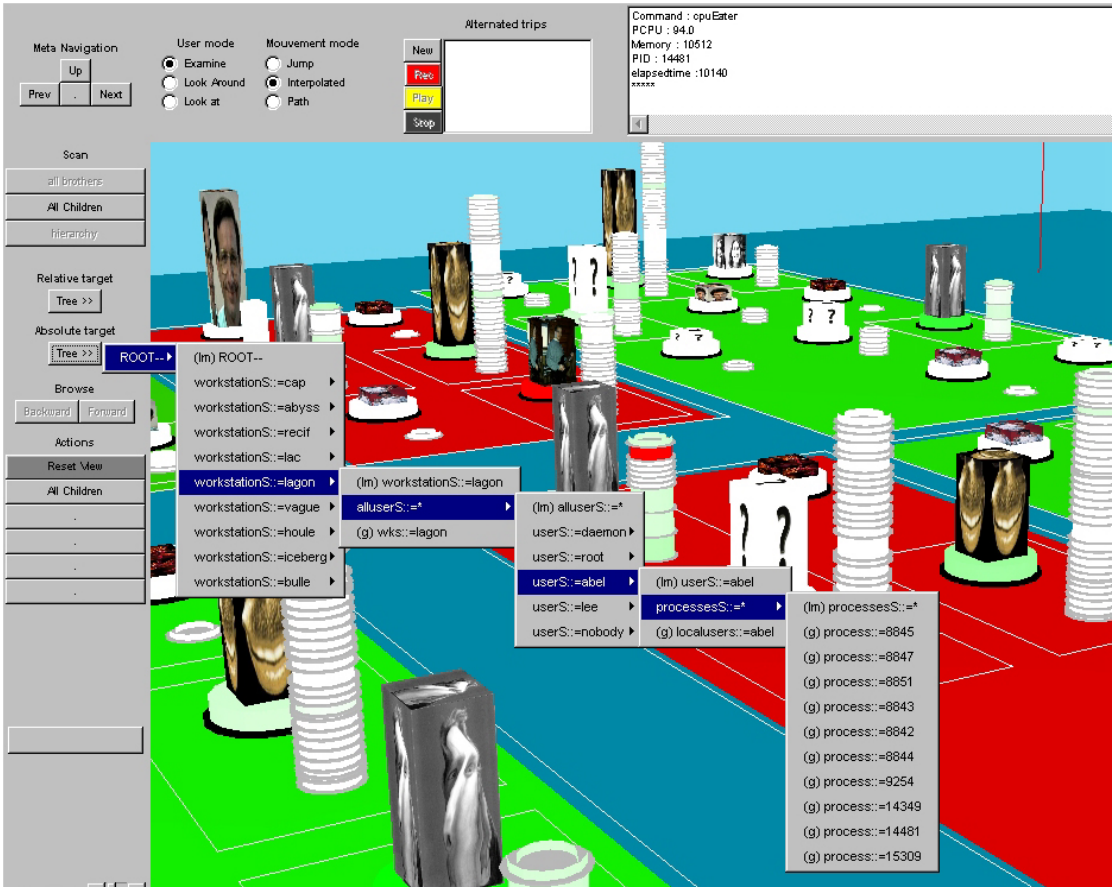


Figure 35 *Sélection d'un nouveau CDI dans un menu hiérarchique*

Le second type de sélection absolue est réalisée de manière visuelle en cliquant sur la représentation graphique d'un CDI visible par l'utilisateur. Cette sélection directe permet une navigation beaucoup plus rapide, puisque aucun mécanisme intermédiaire (comme le menu) n'est nécessaire, et plus intuitive, puisque l'utilisateur sélectionne visuellement son nouveau CDI. Par contre, la sélection d'un nouveau CDI est limitée aux CDI actuellement visibles alors que la première méthode permet d'atteindre tous les CDI existant dans le monde 3D.

3.2.2 Sélection relative

La sélection relative consiste à naviguer dans l'arbre des CDI relativement à la position du CDI courant. De la même façon qu'en sélection absolue, on peut proposer une interaction permettant de choisir un CDI sous forme de nom (par exemple dans un menu). Mais dans le cas de la sélection relative, le parcours de la hiérarchie des CDI est réalisé en partant du CDI courant et non à partir de la racine de la hiérarchie, comme c'est le cas en sélection absolue.

3.2.2.1 Des opérateurs pour la navigation relative

Un mécanisme beaucoup plus intéressant pour réaliser des sélections relatives rapides est de proposer des opérateurs traditionnels comme « remonter » (sélection du CDI père du CDI courant) ou « prochain/précédent » (sélection de CDI du même niveau hiérarchique). Cette technique permet de visiter les CDI assez facilement. Par exemple dans une visualisation constituée d'un immeuble, de couloirs et de bureaux, si un utilisateur se trouve dans un bureau et choisit l'opérateur « prochain », il sera déplacé vers le prochain bureau, tandis que le choix de « remonter » le déplacera vers le couloir lui offrant une vision globale de tous les bureaux.

Navigation à différents niveaux d'informations

Grâce aux opérateurs, on peut naviguer selon le niveau hiérarchique des CDI et donc explorer un monde 3D selon différentes perspectives, selon les différents niveaux d'information que représentent les CDI. Dans une métaphore de ville, par exemple, si le CDI courant de l'utilisateur se trouve sur un bâtiment, l'utilisation des opérateurs « prochain/précédent » lui permet de se déplacer au niveau de la rue, de bâtiment en bâtiment, et donc à un niveau d'information détaillé. En revanche, si son CDI courant se trouve sur un quartier, l'utilisation des mêmes opérateurs va lui permettre de se déplacer au niveau de la ville, de quartier en quartier, c'est à dire à un niveau d'information plus synthétique lui permettant d'obtenir une vision plus globale des informations visualisées par le monde 3D.

Les mécanismes nécessaires à l'utilisation des CDI peuvent être implémentés facilement grâce à un parcours de la hiérarchie des CDI, mais seulement si les CDI sont définis optimalement et classés logiquement au niveau visuel comme nous l'avons souligné précédemment.

Figure 36 *Opérateurs de navigation relative*

3.3 Définir des listes de centres d'intérêt

Certaines tâches de navigation ont pour but la visite d'un ensemble de CDI spécifiques. Par exemple, l'utilisateur peut vouloir revenir aux CDI qu'il vient de visiter ou alors aller surveiller des CDI qu'il considère comme potentiellement problématiques. Les listes de CDI ont pour but de faciliter ce type de tâche, en

fournissant à l'utilisateur des moyen de naviguer parmi un nombre plus restreint de CDI. Deux types de listes sont disponibles : les listes automatiques et les listes utilisateur.

3.3.1 Listes automatiques

Le premier type de liste est géré par le système et défini par rapport à des tâches utiles à l'utilisateur comme :

- *La liste de tous les CDI ayant déjà été visités.*
- *Des listes basées sur des critères liés à la hiérarchie des CDI et relatives au CDI courant.* Une telle liste peut être, par exemple, constituée de tous les fils ou tous les frères du CDI courant. Encore une fois, il est nécessaire que les CDI soient définis optimalement et classés logiquement au niveau visuel, si on veut utiliser directement l'arbre des CDI pour réaliser ces opérations.

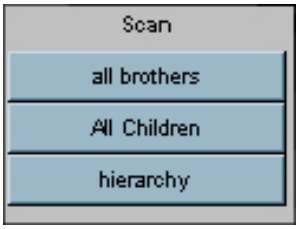


Figure 37 *Interface permettant de consulter automatiquement des listes de CDI générées par le système*

3.3.2 Listes utilisateur

Le deuxième type de liste est défini par l'utilisateur. L'utilisateur a la possibilité de créer des listes contenant un certain nombre de CDI qui l'intéressent particulièrement, comme un ensemble de CDI représentant des informations sensibles. Il peut créer autant de listes qu'il le veut, et ajouter à ces listes autant d'éléments qu'il le désire. Il peut ainsi créer des ensembles de CDI spécifiques parmi lesquels il peut naviguer rapidement.

3.3.3 Consultation des CDI des listes : manuelle ou automatique

Des opérateurs « précédent » et « suivant », semblables à ceux utilisés par la majorité des navigateurs hypertexte, ainsi que la sélection directe parmi tous les CDI de la liste, permettent la sélection manuelle des CDI d'une liste.

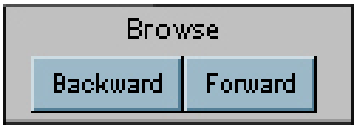


Figure 38 *Interface permettant de parcourir les éléments d'un liste de CDI*

La visite automatique des CDI d'une liste constitue un moyen agréable de visualiser plusieurs CDI. Elle permet d'effectuer une sorte de chemin de ronde. Le système sélectionne un à un les CDI de la liste et, une fois à la fin de la liste, revient au CDI de départ. Des fonctionnalités comme la pause, la reprise ou l'arrêt du processus de navigation automatique permettent d'ajouter de la valeur à ce mécanisme. La vitesse de la visite doit aussi pouvoir être paramétrable. La visite automatique peut se révéler très pratique dans le cas de la supervision d'un certain nombre de CDI.

3.4 Proposer à l'utilisateur un moyen d'observer facilement un centre d'intérêt sous toutes ses coutures

Une fois un CDI choisi, il est nécessaire de fournir à l'utilisateur des moyens d'observer la représentation graphique associée à ce CDI. L'utilisateur peut avoir besoin de se déplacer dans le monde 3D de manière à regarder le CDI sous toutes ses coutures, spécialement si celui-ci est d'une taille importante comme, par exemple, une ville. De plus, il serait dommage de ne pas permettre à l'utilisateur de choisir sa position pour observer ce CDI, et de ne lui offrir que le point de vue associé à ce CDI. Un mécanisme, propre à la logique visuelle du CDI observé, est donc proposé à l'utilisateur.

3.4.1 Point de vue utilisateur

L'utilisateur visualise le monde à partir d'un point de vue, appelé *point de vue utilisateur*. Lors de l'initialisation d'une visualisation, le CDI correspond à la racine de la hiérarchie des CDI et le point de vue utilisateur se confond avec le point de vue associé à ce CDI. En utilisant les mécanismes d'observation, le point de vue utilisateur ne correspond plus au point de vue associé à ce CDI, mais une interaction lui permet de revenir au point de vue du CDI d'attachement (c'est à dire le CDI courant, sauf dans un cas que nous verrons par la suite). Ce mécanisme peut être pratique lorsque l'utilisateur se perd ou qu'il veut revenir à une vue standard (cf. Figure 39).

Le point de vue utilisateur est utilisé par le système, lors de la sélection d'un nouveau CDI, afin de calculer un nouveau point de vue utilisateur dont le résultat est une vue du nouveau CDI, équivalente à la vue qu'avait l'utilisateur du précédent CDI (cf. Figure 39). Le calcul du nouveau point de vue utilisateur est une fonction des points de vue du nouveau et de l'ancien CDI et du point de vue utilisateur existant lors du changement de CDI.

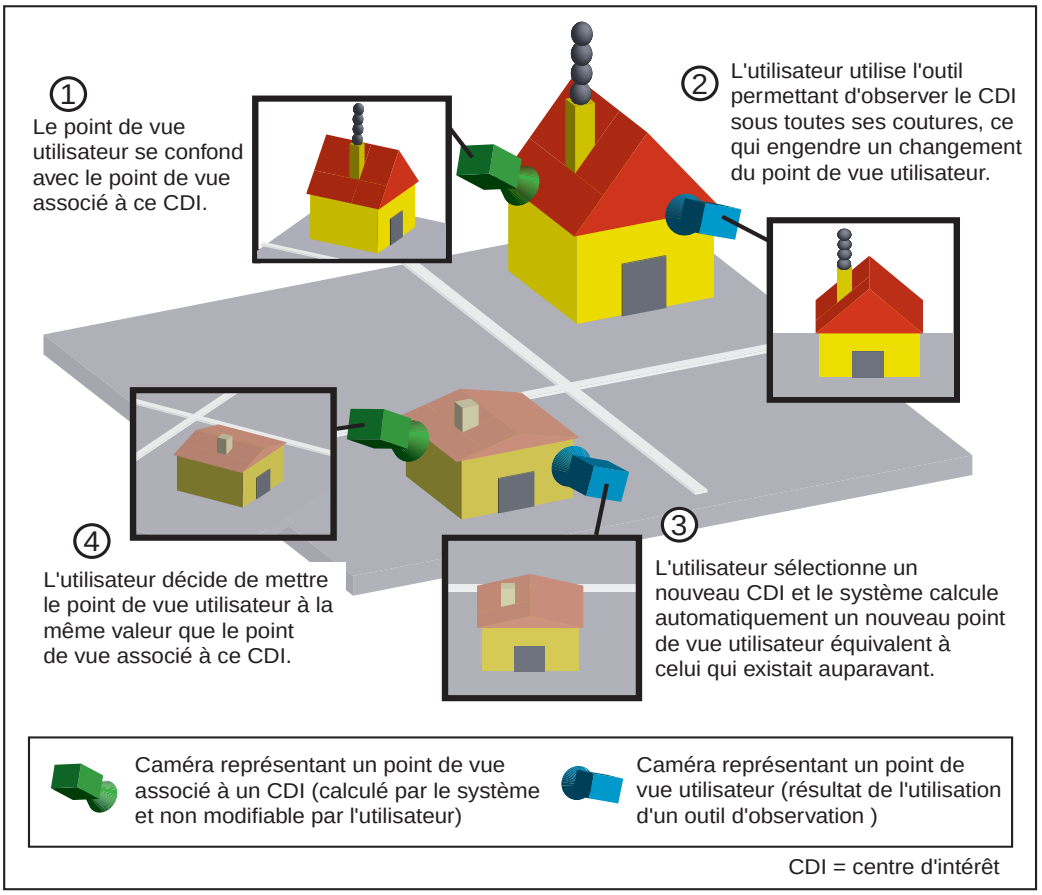


Figure 39 *Point de vue associé aux CDI et point de vue utilisateur*

3.4.2 Pour une observation spécifique aux caractéristiques visuelles du CDI

Le mécanisme d'observation n'est pas le même pour tous les CDI. C'est la connaissance de la représentation graphique et de l'environnement visuel autour du CDI qui mène à la spécification du mécanisme d'observation utilisé. En effet, l'utilisation d'une seule méthode ne serait pas optimale puisque les CDI peuvent varier considérablement en taille et en forme.

Par exemple, pour observer un cube ou n'importe quel objet ayant des proportions à peu près équivalentes dans toutes les dimensions, il est pratique d'utiliser une sphère virtuelle [Chen88] égocentrique, c'est à dire une caméra visant automatiquement le centre de l'objet et laissant l'utilisateur décider de la position de la caméra sur une sphère virtuelle dont il peut contrôler le rayon. Pour visualiser un CDI ayant plutôt l'aspect d'un plan de grandes dimensions (comme par exemple une ville), une caméra aérienne visant automatiquement le bas et laissant l'utilisateur décider de sa position au-dessus de la ville sera plus appropriée.

3.4.3 Effet de bord du mécanisme d'observation : un moyen de navigation

L'observation d'un CDI entraîne un déplacement dont le résultat est conservé dans le point de vue utilisateur. Ce déplacement permet indirectement à l'utilisateur de naviguer aux alentours du CDI. En particulier, l'utilisation du mécanisme d'observation permet de faire apparaître visuellement d'autres CDI qui peuvent donc être analysés voire sélectionnés en cliquant dessus (sélection absolue). La liberté avec laquelle cette navigation est possible dépend entièrement du mécanisme d'observation associé au CDI. Ceci est un effet de bord, les autres moyens de navigation sont les transitions automatiques engendrées par la sélection de nouveaux CDI.

3.5 Transitions entre les centres d'intérêts

Jusqu'à maintenant, nous avons vu que l'utilisateur pouvait sélectionner ou visiter automatiquement des CDI mais nous n'avons pas parlé de ce qu'il se passait entre le moment où l'utilisateur observe l'ancien CDI et celui où il peut observer le nouveau CDI qu'il a choisi, c'est à dire la transition entre deux CDI.

Plusieurs modes de transitions complémentaires sont possibles, ils ont été déterminés à partir des questions suivantes :

- L'utilisateur veut-il se déplacer dans le monde ou rester au même endroit et juste déplacer son regard ?
- Comment doit s'effectuer dans le temps le mouvement de la position et/ou du regard de l'utilisateur ?

La réponse à la première question est déterminée par les choix existant dans le *mode utilisateur*. La réponse à la deuxième question dépend quant à elle des possibilités existant dans le *mode de mouvement*. La transition est donc définie par la combinaison de ces deux modes et le résultat des combinaisons possibles est résumé, à la fin de cette partie, dans le Tableau 9.

3.5.1 Modes utilisateur : « aller » / « regarder »

Le mode utilisateur permet à l'utilisateur de spécifier s'il désire se déplacer vers le nouveau CDI, ou s'il veut juste tourner la tête afin de le regarder (cf. Figure 40). Le premier mode est appelé *« aller »* et le second *« regarder »*. Dans le mode « regarder », on peut noter que la direction dans laquelle regarde l'utilisateur doit être corrigée en temps réel au cas où la position du nouveau CDI n'est pas statique.

Chaque option entraîne un résultat différent dans la valeur du CDI d'attachement (cf. 3.1.5). Normalement, celui-ci correspond toujours au CDI courant, le but de celui-ci étant de constituer le dernier endroit connu où l'on

pense que l'utilisateur n'était pas perdu. Dans le cas du mode « regarder », en revanche, le CDI courant ne correspond pas à l'endroit où se trouve l'utilisateur mais à celui qu'il regarde ; il ne faut donc pas que le CDI d'attachement soit le même que le CDI courant. Par conséquent, lorsque un nouveau CDI est sélectionné, le CDI d'attachement est mis à la même valeur que le nouveau CDI en mode « aller » et reste inchangé en mode « regarder ».

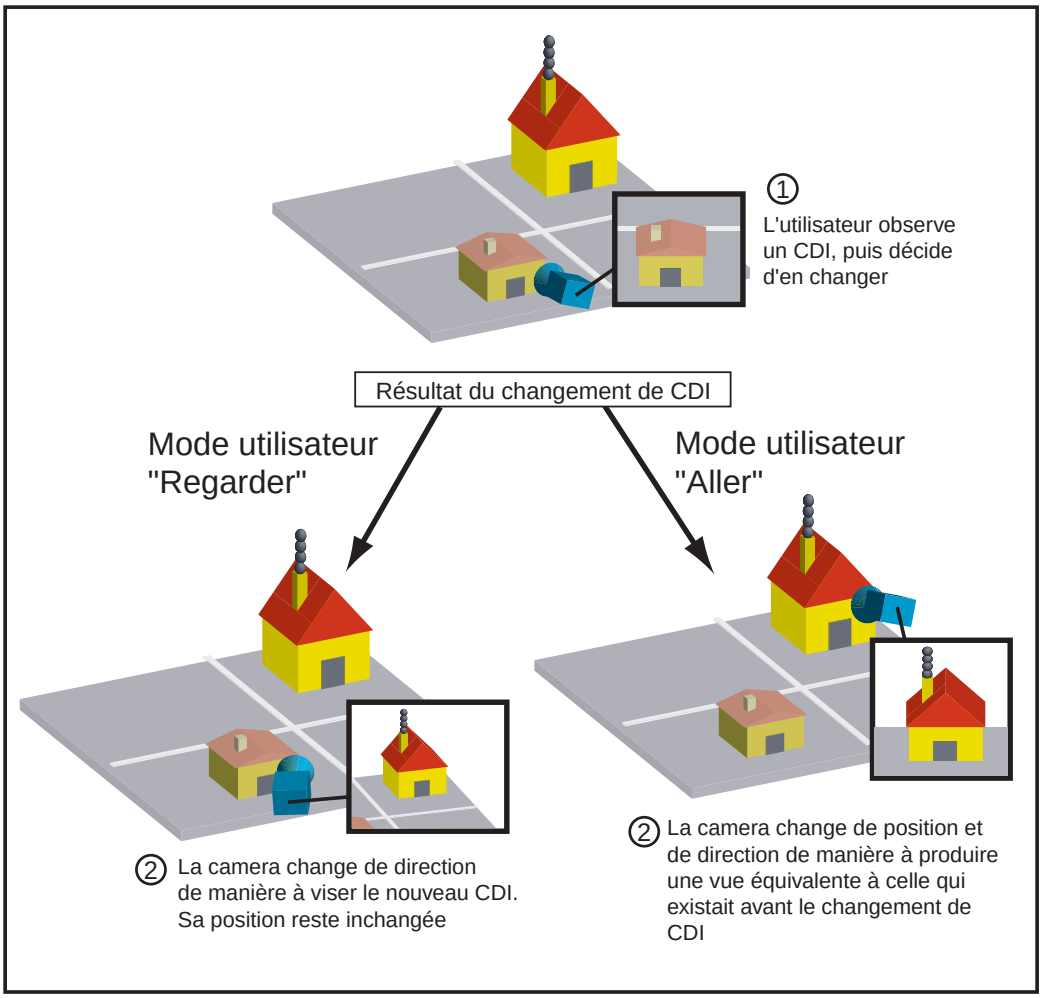


Figure 40 *Résultat d'un changement de centre d'intérêt (CDI) en mode utilisateur « aller » et « regarder »*

3.5.2 Modes de mouvement : « saut »/ « interpolation » / « chemin »

Sur les trois modes de mouvement possibles, les deux premiers (« saut » et « interpolation ») sont les modes standards que l'on peut trouver, par exemple, dans les navigateurs VRML. Le mode « chemin » est quant à lui un mode original que nous décrivons dans la partie suivante.

3.5.2.1 Mode « saut »

Le mode « saut » téléporte l'utilisateur (ou son regard). Bien qu'il soit fascinant car contraire aux lois de la réalité, il ne permet pas à l'utilisateur une bonne connaissance spatiale du monde 3D, c'est à dire sa structure générale ainsi que la position relative des objets les uns par rapport aux autres. En revanche, c'est le moyen le plus rapide pour se rendre à une nouvelle position, ce qui peut s'avérer très utile dans certains cas.

3.5.2.2 Mode « interpolation »

L'utilisation du mode « interpolation » entraîne une animation linéaire de la position et de la direction actuelles de la caméra vers celles de destination (mode utilisateur « aller») ou une simple animation de l'orientation (mode utilisateur « regarder »). Ce mode permet une certaine compréhension du monde 3D mais étant mis en œuvre de la même façon quel soit le type de la visualisation, il ne prend pas en compte l'apparence visuelle du monde 3D comme le fait le mode « chemin ».

Mode mouvement **Mode utilisateur**	Saut	Interpolation	Chemin
Aller	L'utilisateur se téléporte vers le nouveau CDI	L'utilisateur vole en ligne droite de sa position courante vers son nouveau CDI	L'utilisateur voyage de sa position courante vers son nouveau CDI selon un chemin respectant la logique de la visualisation en cours
Regarder	L'utilisateur reste à sa position courante et son regard est tourné en direction de son nouveau CDI.	L'utilisateur reste à sa position courante, son regard est animé linérairement de sa direction courante jusqu'en direction de son nouveau CDI.	L'utilisateur reste à sa position courante, son regard est animé de sa direction courante jusque dans la direction de son nouveau CDI, mais en suivant un chemin logique par rapport à la structure de la visualisation en cours.

Tableau 9 *Mouvements obtenus par la combinaison des différents modes de navigation*

3.6 Transition entre les centres d'intérêt à l'aide du mode de mouvement « chemin »

Le mode « chemin », aussi appelé « navigation métaphorique » [Russo00c], permet de prendre en compte l'apparence et la structure visuelle du monde 3D pour réaliser le mouvement de transition entre deux CDI. Les mouvements réalisés dans ce mode sont donc complètement différents d'un monde à l'autre. Contrairement aux deux autres modes de mouvement, l'élaboration des déplacements en mode « chemin » est la plupart du temps complexe. En effet, ceux-ci ne sont que très rarement linéaires puisqu'ils sont déterminés de façon à respecter la logique du type de visualisation utilisé.

3.6.1 Exemple

Une façon de comprendre le concept de « chemin » est d'étudier le cas d'un mouvement dans une métaphore d'un bâtiment constitué d'étages et de bureaux. Si l'utilisateur se trouve dans un bureau (son CDI) et désire aller dans un nouveau bureau, il va suivre un chemin empruntant le couloir et les escaliers (cf. Figure 41) correspondant à la logique de la structure visuelle. En suivant ce chemin, l'utilisateur va pouvoir observer les CDI se trouvant sur son chemin et prendre connaissance de la position des uns par rapport aux autres. Ce mouvement serait difficile à réaliser grâce à des mécanismes de navigation libre. Le mode « chemin » se charge d'effectuer automatiquement tous les mouvements nécessaires (tourner, avancer, descendre, etc.).

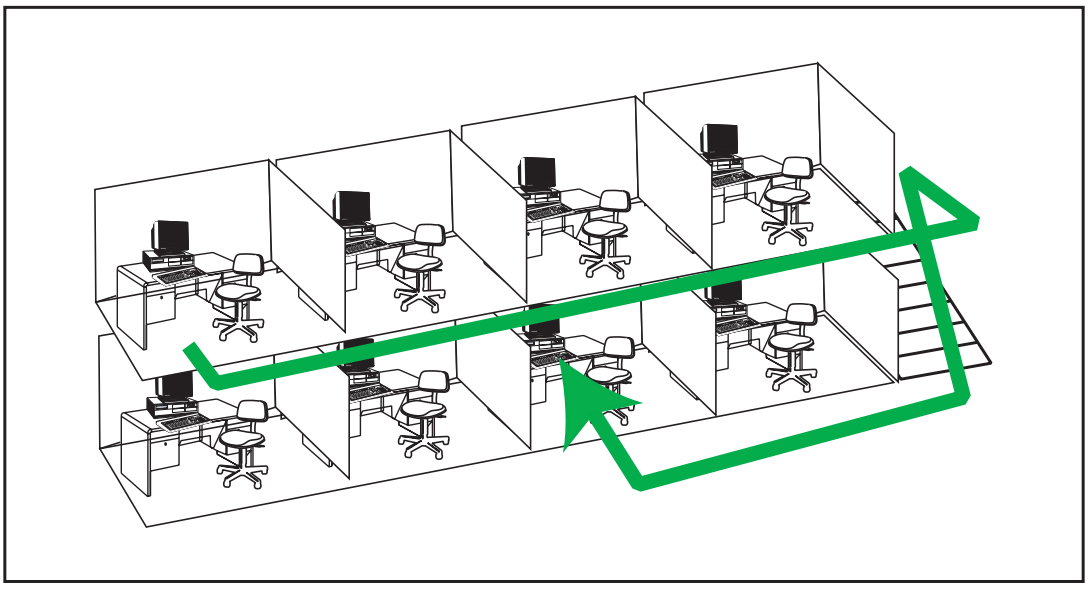


Figure 41 *Trajectoire d'un bureau à un autre dans un bâtiment*

3.6.2 Bénéfices

Dans le mode « chemin », la trajectoire (du regard et/ou de la position) de l'utilisateur pour se rendre vers le nouveau CDI n'est plus indépendante de la structure de la visualisation comme c'est le cas dans les autres modes de mouvement. De ce fait, nous pensons que le mode « chemin » permet à l'utilisateur de mieux appréhender, de mieux découvrir la structure du monde en la visitant logiquement et automatiquement, et ainsi de pouvoir mieux exploiter le monde 3D. Pour les mêmes raisons, ce mode de mouvement permet à l'utilisateur d'obtenir une meilleure connaissance de l'emplacement relatif des CDI, notion importante car elle facilite à l'utilisateur l'utilisation de la sélection absolue visuelle pour passer d'un CDI à l'autre (i.e. en cliquant sur le CDI de destination).

Le chemin généré par ce mode permet également à l'utilisateur de visualiser d'autres CDI en cours de route, et lui donne donc l'opportunité de les analyser, ceci pouvant être utile lorsque l'utilisateur ne cherche pas une information en particulier mais veut plutôt explorer les informations existantes.

Un autre avantage de ce type de mouvement est de permettre de limiter la désorientation de l'utilisateur. En effet, dans de nombreux cas (par exemple dans une métaphore de ville) les déplacements engendrés par ce mode simulent ceux que l'utilisateur réaliseraient s'il était confronté, dans la réalité, à une structure similaire à celles du monde 3D. Les déplacements lui semblent naturels et limitent donc sa possible désorientation suite aux mouvements automatiques.

3.6.3 Désavantages

Ce mode est cependant celui entraînant la transition la plus longue de par la nature complexe de la trajectoire, il n'est donc pas recommandé pour certaines tâches comme « se rendre le plus vite possible à ce CDI » dont le but n'est pas d'appréhender la structure du monde, ni de visualiser d'autres CDI se trouvant sur la trajectoire calculée.

4 Utilisation du modèle de navigation dans CyberNet

Dans cette partie, nous expliquons comment le modèle de navigation que nous venons de présenter peut être utilisé conjointement au modèle de visualisation que nous avons exposé au chapitre précédent. Cette partie prépare le terrain, en vue de la présentation de notre modèle plus général de monde 3D interactifs.

4.1 Un CDI pour chaque CG

Afin d'exploiter le modèle de navigation conjointement au modèle de visualisation, un CDI est défini pour chaque CG (glyphe ou gestionnaire de placement) existant dans le monde 3D. Ainsi tous les CDI possibles sont visuellement couverts, du plus petit glyphe (e.g. une maison) au plus grand gestionnaire de placement (e.g la ville entière). Le CG correspondant au centre intérêt courant est appelé *composant graphique d'intérêt (CGI)*. Lors de l'initialisation d'une visualisation, le CGI est le CG racine de la hiérarchie représentant le monde 3D, ce qui équivaut à placer le centre d'intérêt de l'utilisateur sur le monde entier.

4.2 Problème de non optimalité des CDI pour la navigation

En définissant un CDI pour chaque CG, la hiérarchie des CDI correspond à celle des CG, ce qui entraîne parfois la définition de CDI redondants pour la navigation relative. Prenons l'exemple d'une visualisation basée sur une métaphore de ville et composée de quartiers où des bâtiments sont posés sur un trottoir. Cette visualisation peut être basée sur la hiérarchie de CG de la Figure 42. Dans ce cas, le point de vue associée au CDI3 est dit redondant par rapport au CDI2. En effet, de par la structure visuelle du monde, le CDI3 permet de voir tous les bâtiments, alors que le CDI2 permet de les voir aussi correctement avec, en plus, le trottoir. La définition des CDI dans CyberNet n'est donc pas optimale pour la navigation relative car deux CG de la structure hiérarchique peuvent générer des CDI dont les points de vue sont redondants, ce qui entraîne une navigation relative plus fastidieuse pour l'utilisateur.

Le résultat de ces CDI redondants est que les fonctions reposant sur la sélection d'un nouveau CDI relativement au CDI courant ne doivent pas être définies d'une façon générique. Dans le cas précédent, par exemple, si l'utilisateur désire naviguer du CDI4 (un bâtiment) jusqu'au CDI1 (la ville entière) en utilisant plusieurs fois l'opérateur « remonter », une méthode générique imposerait d'utiliser cet opérateur 3 fois pour effectuer le parcours le menant au CDI3, puis au CDI2 et enfin au CDI1. Or nous avons vu que le CDI3 était redondant avec le CDI2, il n'est donc pas performant de faire passer l'utilisateur par le CDI3. L'action de l'opérateur « remonter » à partir du CDI4 sélectionne donc directement le CDI2 et non le CDI3 (cf. Figure 42).

Utilisation de critères visuels

Le résultat de l'utilisation d'un opérateur (et donc la sélection d'un nouveau CDI) est entièrement dépendant du type de la visualisation et du CDI courant. Les opérateurs répondent donc à partir de critères visuels et non à partir de considérations purement hiérarchiques. D'autres mécanismes, propres à chaque CDI, sont également définis de la sorte et regroupés dans un composant de navigation.

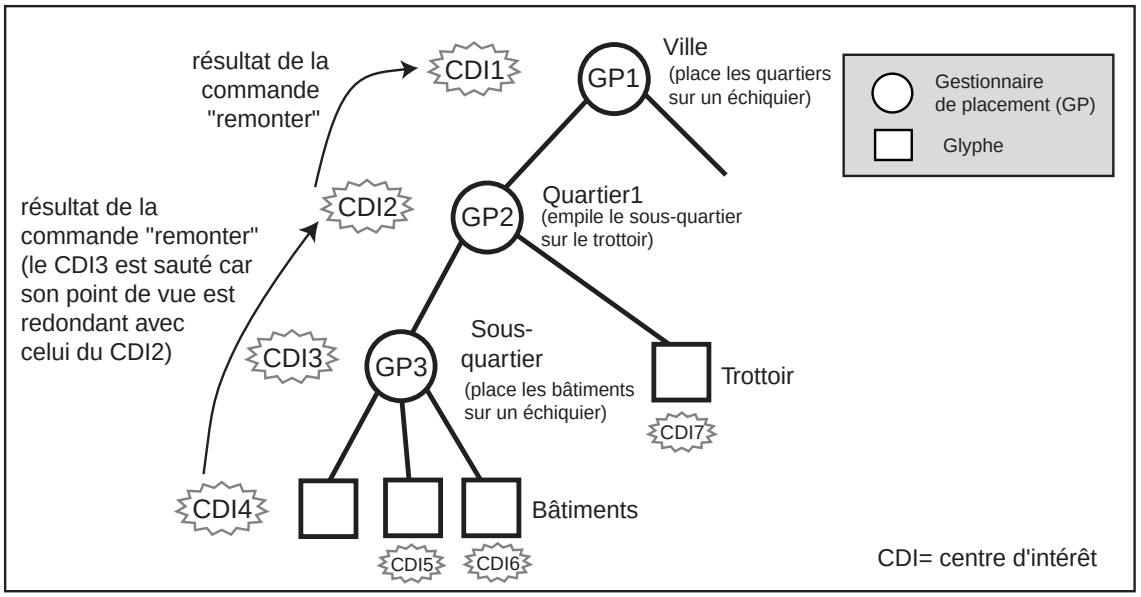


Figure 42 *Exemple d'une navigation relative sautant un CDI ayant un point de vue redondant*

4.3 Composant de navigation (CN)

4.3.1 Un composant de navigation pour tous

Afin de réaliser certaines opérations qui sont définies dans le modèle de navigation en fonction de critères visuels et non d'une façon générique (par exemple l'utilisation de la hiérarchie de CDI), un composant de navigation (CN) est associé à chaque CDI et donc à chaque CG. Chaque CN est défini en fonction du CG auquel il est associé, et le résultat des opérations qu'il réalise dépend du type et de la place de ce CG dans l'arbre décrivant le monde 3D.

Etant donné qu'il existe un CN pour chaque CG, il en résulte un arbre de CN correspondant à la hiérarchie des CG (cf. Figure 43).

Un CN comporte plusieurs fonctions afin de réaliser les opérations nécessitant une connaissance de la représentation graphique du CG associé et de l'environnement visuel autour de ce CG. Ces fonctions sont :

- Une fonction de sélection relative
- Une fonction permettant d'implémenter les mouvements nécessaires dans le mode de mouvement « chemin »
- Une fonction délivrant le mécanisme utilisé pour observer sous toutes ses coutures le CG associé

Les deux premières fonctions ont besoin de communiquer avec les autres composants de navigation comme nous allons le voir.

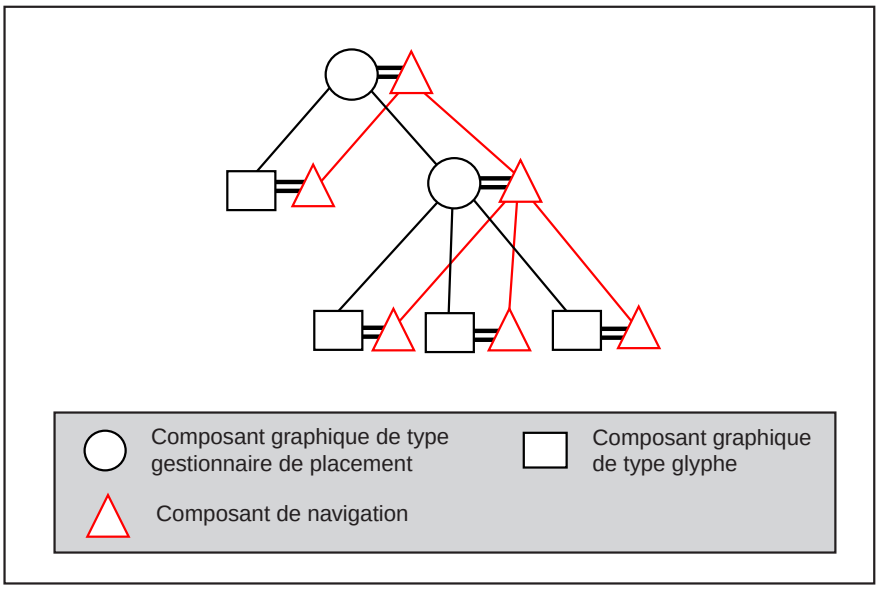


Figure 43 *Hiérarchie de composant de navigation*

4.3.2 Communication entre les composants de navigation

Pour réaliser leur travail, les fonctions de sélection implémentant le mode « chemin » ont besoin de communiquer avec les fonctions similaires de leurs CN père ou fils. Grâce à cette communication, le calcul des requêtes de l'utilisateur peut être réalisé d'une façon décentralisée.

Les fonctions de sélection coopèrent entre elles pour répondre à certaines questions. Nous avons vu dans l'exemple de la visualisation sous forme de ville (Figure 42), que l'opérateur de navigation « remonter » permettait d'omettre un CDI dont le point de vue était redondant au niveau visuel. Ceci est rendu possible grâce aux communications entre les fonctions de sélection des CN. Dans cet exemple, le CN associé au CDI4 va envoyer la requête « sélectionne-toi » à son CN père qui va faire suivre cette requête à son père car il sait qu'il est associé à un CDI redondant. C'est donc la coopération des CN qui va permettre de respecter la logique visuelle.

Les mécanismes des fonctions de sélection sont relativement simples, contrairement à ceux utilisés pour réaliser les mouvements nécessaires au mode « chemin »

4.3.3 Mode chemin : une navigation distribuée

Pour réaliser les mouvements nécessaires à l'utilisation du mode « chemin », deux mécanismes sont implémentés :

- Une table de routage dont le fonctionnement est assez proche de celui utilisé par le protocole IP afin de router les paquets sur Internet. Le but de cette table est de définir quel est le prochain CN qui doit être traversé pour aller de la position courante à la destination désirée. Dans l'exemple d'une visualisation de bâtiment, aller à un bureau situé à un étage différent nécessite de visiter des CN intermédiaires comme l'escalier.
- Un mécanisme permettant de contrôler les mouvements dans l'espace géré par le CG associé. Par exemple, le CN associé au CG représentant l'escalier doit gérer les déplacements dans celui-ci. Durant l'implémentation de ces mécanismes, on doit prendre soin que le chemin soit continu lorsqu'on se déplace d'un CN à un autre.

Le système de navigation utilise ces deux mécanismes pour créer le mouvement menant d'un CDI de départ à un CDI de destination. La génération du mouvement implique l'activation de toute une série de CN. Le CN associé au CDI de départ est le premier activé. Il utilise sa table de routage pour déterminer quel est le prochain CN à activer pour se rendre à la destination voulue, et avant de l'activer, déplace l'utilisateur dans son espace. Ce schéma est répété jusqu'à l'arrivée au CDI de destination (cf. Figure 44).

Le mouvement est donc réalisé de manière décentralisée par chaque CN activé. La complexité du chemin est le résultat d'une série de mouvements plus simples calculés par chaque CN.

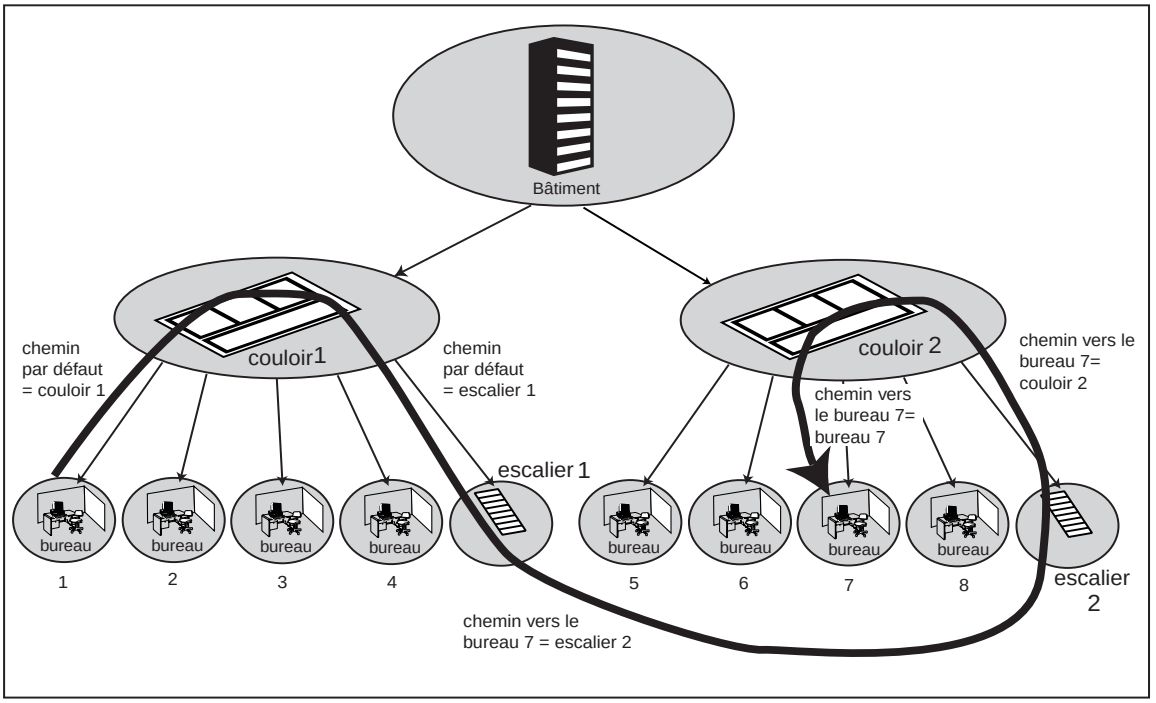


Figure 44*Mode navigation chemin à l'aide des tables de routage*

5 Autres interactions

Jusqu'à présent nous avons vu des interactions permettant la navigation dans le monde 3D. Nous présentons maintenant d'autres mécanismes d'interaction, mais sans les approfondir de la même façon que nous l'avons fait pour la navigation. Ces mécanismes ont toujours le même but : permettre à l'utilisateur d'exploiter pleinement les informations représentées visuellement dans le monde 3D.

Nous présentons brièvement quatre types d'interactions :

- Obtenir les informations d'un CDI sous forme de texte
- Effectuer des modifications visuelles pour optimiser un type de visualisation
- Sélectionner/Mettre en avant visuellement des CDI

Nous présentons également le composant d'interaction qui, à l'instar du composant de navigation, permet de spécifier des interactions spécifiques à un CG. En dernier lieu, nous discutons de la gestion de multiples visualisations en parallèle, et des conséquences que cela implique.

5.1 Informations sous forme de texte

La représentation visuelle permet à l'utilisateur d'interpréter plus vite les informations que si elles étaient sous forme de texte. Cependant, il est parfois nécessaire pour lui de connaître les informations associées à un certain CDI sous forme de texte afin d'en connaître les valeurs exactes. Pour cela, il est possible dans CyberNet d'interagir avec les CDI (et donc les CG) visibles dans la vue de l'utilisateur afin que s'affichent en surimposition les informations sous forme de texte (Figure 45).

Ce mécanisme permet de s'assurer de l'interprétation des informations représentées visuellement, et est particulièrement utile dans les cas où un CDI semble préoccupant (par exemple sur la Figure 45, on demande des informations sur un processus consommant beaucoup de CPU).

Figure 45 *Informations d'un CDI sous forme de texte*

5.2 Modifications visuelles

Il est intéressant de proposer des mécanismes permettant de modifier certains paramètres visuels pour optimiser certaines tâches. Par exemple dans une visualisation basée sur un bâtiment, il peut être utile à l'utilisateur de faire disparaître les murs (ou de les rendre semi-transparents) afin de l'aider à s'orienter plus facilement ou pour obtenir une vision d'ensemble du bâtiment. La Figure 46 contient plusieurs captures d'écran de la même vue, avec des paramètres de transparence différents pour divers éléments graphiques.

Ce type de modification ne peut intervenir que sur des éléments ne représentant pas des informations, mais qui structure le monde 3D, comme les murs d'un bâtiment. Pour les éléments représentant des informations, ce type de modifications équivaudrait à modifier le mode mappage des information sur les éléments graphiques, ce qui n'est pas le but de cette interaction visant à aider l'utilisateur dans son exploitation du monde 3D.

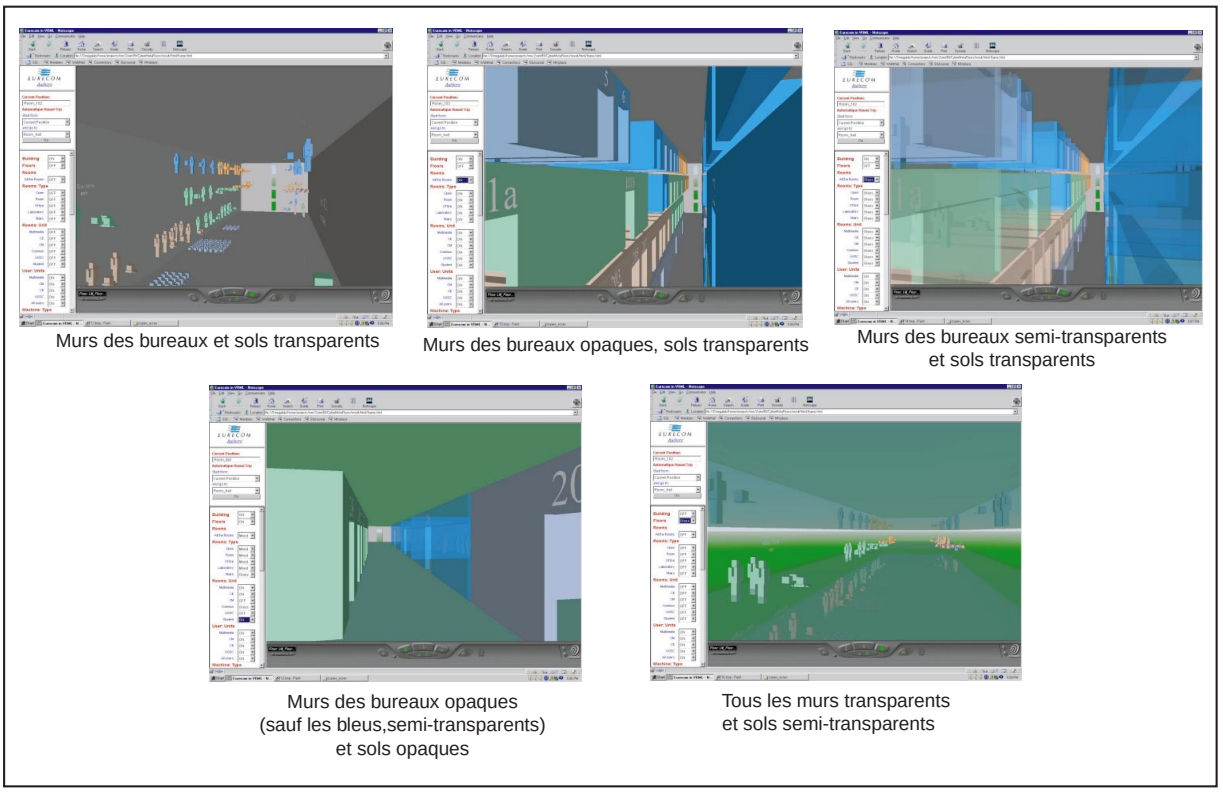


Figure 46 *Résultats de la modification de la transparence des divers éléments d'une métaphore de bâtiment*

.

5.3 Sélection/Mise en avant de CDI

Nous avons vu dans le modèle de navigation qu'il existait toujours un CDI courant ainsi que des méthodes permettant de le sélectionner. La sélection dont nous parlons ici n'a rien à voir avec la navigation mais avec la mise en avant, visuellement, d'un ou de plusieurs CDI pouvant éventuellement contenir le CDI courant. Ce type de sélection peut se révéler pratique pour visualiser certains CDI ayant des relations qui ne sont pas bien représentées dans la visualisation de par la structuration des informations.

Par exemple, dans une visualisation d'un parc d'ordinateurs où chaque ordinateur est visualisé séparément avec ses utilisateurs, il peut être intéressant de mettre en avant toutes les instances d'un utilisateur particulier de manière à voir son activité sur le réseau. Dans le cas de la visualisation d'un réseau, il peut être intéressant de mettre en avant les éléments actifs du réseau (hubs, routeur, liens) par lesquels doivent passer les données pour arriver à un ordinateur particulier.

Cette mise en avant visuelle peut s'effectuer en éclairant les CDI sélectionnés ou, au contraire, en cachant ou en rendant translucides les autres représentations graphiques non sélectionnées.

5.3.1 Utilisation avec plusieurs visualisations

Cette mise en avant peut se révéler très utile lorsqu'on utilise plusieurs vues des mêmes informations à l'aide de visualisations différentes apportant chacune ses particularités. Par exemple, on a vu au chapitre 2 deux exemples organisant les mêmes informations (ordinateurs, utilisateurs et processus) selon deux stratégies différentes (par ordinateur ou par utilisateur)

Dans ce cas, si la mise en avant des CDI dans une visualisation se répercute dans les autres, il est possible de passer d'une visualisation à l'autre sans perdre de vue les CDI mis en avant, et de profiter des particularités apportées par chaque visualisation.

5.4 Composant d'interaction (CI)

Dans le but d'incorporer les interactions au sein des mondes 3D créés grâce au modèle de visualisation défini au chapitre 2, on associe un composant d'interaction (CI) à chaque CG (cf. Figure 47), de la même manière que l'on a associé un composant de navigation (CN) à chaque CG. Comme dans le cas du CN, les fonctions d'un CI sont spécifiques au CG auquel il est associé. Ainsi, des interactions spécifiques à chaque CG peuvent être proposées à l'utilisateur. Par exemple, c'est le CI qui peut prendre en charge les fonctions de mise en avant ou de modification visuelle que nous venons de voir.

Un autre exemple d'interaction qui pourrait être pris en charge par le CI est la possibilité de changer les paramètres de positionnement d'un GP. Imaginons un GP plaçant ses fils sur un échiquier. Nous avons vu au chapitre précédent que le placement de ces fils sur l'échiquier peut permettre de les classer selon un critère. Une possibilité serait de permettre à l'utilisateur de spécifier ce critère à travers une interaction, celle-ci étant définie et gérée par le CI associé au GP.

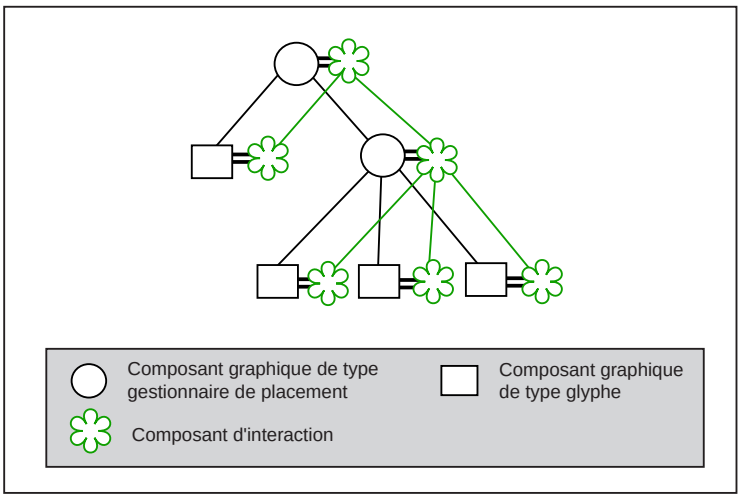


Figure 47 *Hiérarchie de composants d'interaction*

5.5 Perspectives

Nous n'avons pas étudié, par manque de temps, la définition d'un modèle cohérent d'interaction ainsi que son utilisation conjointe avec le modèle de visualisation, comme nous l'avons fait pour le modèle de navigation. Mais nous avons quand même quelques idées sur la question.

Par exemple, nous pensons que les interactions proposées par les CI existants dans une visualisation ne doivent pas être toutes proposées à l'utilisateur en même temps, sous peine qu'il se perde dans leur nombre. C'est donc logiquement les interactions associées au CG courant (i.e. le CG associé au CDI courant) qui devraient être proposées à l'utilisateur. Mais, à nos yeux, certaines interactions devraient être accessibles en permanence comme, par exemple, la possibilité de changer la transparence des murs comme dans l'exemple de la Figure 46. C'est pourquoi il nous semble que l'utilisateur devrait pouvoir accéder en permanence à, d'une part, des fonctions d'interaction utiles quel que soit le CDI courant, et, d'autres part, les fonctions spécifiques au CDI courant.

D'un autre coté, nous pensons que de nombreux autres types d'interaction pourraient être utiles dans l'exploitation de la visualisation, comme par exemple la possibilité de choisir la complexité de la représentation d'une partie du monde (i.e. d'un CDI), comme le fait [Bederson94] dans son travail sur les interfaces 2D zoomables.

5.6 Multiples visualisations

Nous avons déjà évoqué les avantages pour l'utilisateur de pouvoir utiliser plusieurs visualisations en même temps, en particulier lorsque ces visualisations représentent des services contenant les mêmes informations mais organisées différemment. Selon la tâche de l'utilisateur, il y a toujours un service préférable, et si cette tâche ne changeait jamais l'utilisation de multiples visualisations n'aurait pas de sens. Mais la tâche d'un utilisateur peut bien sur changer.

Imaginons le travail d'un superviseur d'un réseau d'ordinateur, en train d'utiliser un service présentant les ordinateurs avec leurs utilisateurs et qui s'aperçoit que tous les ordinateurs sont très chargés. Il identifie le problème sur un ordinateur et s'aperçoit que c'est un utilisateur qui consomme tout son CPU. Il se dit alors que c'est peut-être la même personne qui prend le CPU de tous les autres ordinateurs. Il décide alors de changer de service (et donc de visualisation) pour celui présentant les utilisateurs avec les ordinateurs sur lesquels ils sont connectés, de manière à analyser le comportement de l'utilisateur suspect sur tout le réseau. Il s'aperçoit alors que cet utilisateur est connecté sur tous les ordinateurs du réseau et consomme sur chacun d'eux tout le CPU.

Ce petit exemple nous montre qu'il peut être très productif de pouvoir changer de type de service, et donc de visualisation, afin d'analyser un problème. Nous avons donc réfléchi à des solutions pour permettre à un utilisateur de pouvoir changer de service tout en essayant de conserver le contexte dans lequel il se trouve lors du passage d'une visualisation à une autre.

5.6.1 Conserver le contexte

Définir l'interaction permettant de changer de visualisation de manière à pouvoir analyser des informations sous un autre angle ne pose pas trop de problèmes. On peut par exemple imaginer une simple liste où l'utilisateur peut choisir la visualisation voulue. En revanche, il importe de conserver le contexte lors du passage entre les deux visualisations.

Le contexte est constitué par les informations que l'utilisateur est en train de visualiser. Dans l'exemple que nous avons vu précédemment, lorsque le superviseur change de visualisation, le contexte est constitué par l'utilisateur suspect ou plus précisément l'activité de celui-ci sur une machine. Pour conserver ce contexte, le système le recherche dans la visualisation qui vient d'être sélectionnée. Puis il positionne l'utilisateur de façon à ce qu'il visualise les mêmes informations (i.e. le contexte) que dans la précédente situation. Ainsi, le superviseur se retrouve directement en train de visualiser l'activité de l'utilisateur sur tout le réseau.

Dans notre système, le contexte est matérialisé par le CDI courant de l'utilisateur. Ainsi lorsque l'utilisateur passe d'une première visualisation à une seconde, le système essaye de trouver si les informations représentées par le CDI dans la première ont un équivalent dans la seconde, auquel cas il sélectionne automatiquement le CDI correspondant dans la seconde visualisation.

Bien sûr, ceci est possible seulement si les mêmes informations sont partagées entre les deux visualisations. Dans le cas contraire, le CDI courant de la nouvelle visualisation est inchangé. Mais même si les mêmes informations sont présentes dans les deux visualisations il n'est pas évident de trouver l'équivalent du CDI (par exemple, quel est l'équivalent du CDI représentant un ordinateur dans le service orienté ordinateur de l'exemple précédent, dans le service orienté utilisateur où cet ordinateur est représenté autant de fois qu'il y a d'utilisateurs sur celui-ci ?). Ces questions restent à explorer.

5.6.2 Organisation des différentes visualisations

Un critère important dans la productivité des multiples visualisations est la façon dont celles-ci sont présentées à l'utilisateur : plusieurs fenêtres avec une interface propre à chacune ? Plusieurs sous-fenêtres mais avec une interface partagée ? Une seule fenêtre avec la visualisation en cours (cf. Figure 48) ?

Nous écartons la première solution car nous pensons qu'obliger l'utilisateur à modifier tout son environnement à chaque changement de visualisation n'est pas très pratique. La seconde solution (plusieurs sous-fenêtres avec une interface partagée) nous paraît très intéressante car elle permet de visualiser en parallèle les mêmes informations sous plusieurs angles. De plus, si les visualisations sont synchronisées et partagent les mêmes informations, la navigation dans une d'entre elles peut être répercutée dans les autres, le contexte étant alors le même dans toutes les visualisations. Le problème de cette solution est que l'écran est partagé entre plusieurs fenêtres rendant moins lisibles les visualisations par manque d'espace.

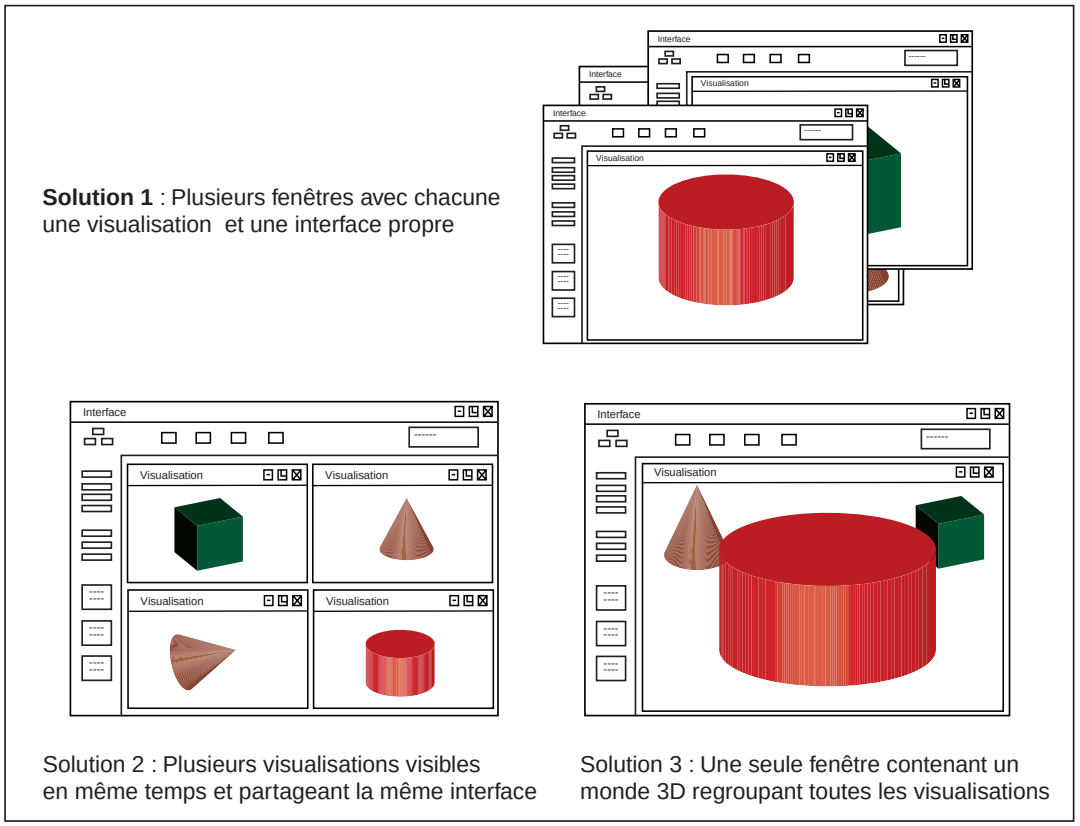


Figure 48 : *Possibilités d'organisation des différentes visualisations*

La dernière solution est d'utiliser une seule fenêtre de visualisation, c'est à dire un seul monde 3D incluant les visualisations de tous les services. Le passage d'une visualisation à l'autre se fait alors de la même manière que pour la navigation à l'intérieur d'une même visualisation. Cette solution a l'avantage d'offrir un espace maximal à l'utilisateur et de lui permettre de se concentrer sur une seule fenêtre, mais elle n'apporte pas l'avantage de la seconde solution qui permet de visualiser en parallèle les mêmes informations sous plusieurs angles.

Une solution intermédiaire serait de proposer un environnement alliant les deux dernières solutions. Un seul monde 3D contenant toutes les visualisations existerait, mais il serait possible de choisir le nombre de vues (i.e. de fenêtres) que l'on désire avoir sur ce monde 3D. Ainsi l'utilisateur pourrait choisir la solution qui lui convient le mieux. Cette solution est celle qui a été adoptée par de nombreux logiciels de modélisation 3D.

6 Conclusion

Dans ce chapitre, nous avons présenté un modèle permettant de faciliter la navigation à un utilisateur dans un monde 3D, et cela sans qu'il se perde. Ce modèle repose sur la définition de tous les centres d'intérêt (CDI), représentant

chacun une partie du monde 3D où l'utilisateur est susceptible de porter son attention. Grâce à une organisation hiérarchique des CDI et des outils de sélection, absolue ou relative au CDI courant, l'utilisateur peut naviguer automatiquement et logiquement entre les CDI, le chemin à parcourir entre les CDI étant calculé selon une stratégie choisie par l'utilisateur. Un outil permettant d'observer un CDI sous toutes ses coutures a été également introduit.

Nous avons mis en avant la nécessité de disposer d'outils capables de s'adapter à la structure du monde 3D et des objets le composant, et non seulement d'outils génériques. Par exemple l'outil permettant d'observer un CDI sous toutes ses coutures n'est pas le même quel que soit le CDI.

D'autres types d'interaction ont été plus brièvement présentés, comme la possibilité de modifier certaines caractéristiques visuelles du monde 3D de façon à améliorer certaines tâches de l'utilisateur. Nous avons également abordé la possibilité de créer un modèle complet d'interaction comme cela été fait pour la navigation.

Une présentation de l'utilisation du modèle de navigation et des autres interactions conjointement au modèle de visualisation décrit au chapitre précédent a été exposée, ceci en vue de la présentation au prochain chapitre de notre modèle plus général de mondes 3D interactifs. Notre solution consiste à définir un CDI pour chaque composant graphique de la visualisation, puis d'associer à chaque CDI un composant de navigation (CN) régissant les différents outils nécessaires au modèle de navigation en fonction des caractéristiques du composant graphique auquel il est rattaché par l'intermédiaire du CDI.

5 Mappage d'informations sur des mondes 3D interactifs

1 Introduction

Dans ce chapitre, nous présentons le processus menant au but du projet CyberNet : la construction automatique des mondes 3D interactifs représentant les informations dynamiques structurées hiérarchiquement par des services.

Pour cela, nous présentons un modèle de monde 3D interactif reposant sur les modèles de visualisation, de navigation ainsi que sur les autres interactions que nous avons étudiés dans les chapitres précédents. Ce modèle est conçu de manière à faciliter la construction automatique de mondes 3D interactifs, mais aussi le mappage des informations sur ceux-ci.

Dans ce processus de mappage d'informations, il est nécessaire de choisir sur quel type de monde 3D interactif on mappe ces informations. Au sein du projet CyberNet, un type de monde 3D interactif est appelé une métaphore, car la plupart des mondes que nous développons possèdent un caractère métaphorique.

Dans ce chapitre, nous n'abordons que très brièvement la question du bien-fondé d'un mappage. En d'autres termes, nous nous focalisons plus sur les mécanismes permettant le mappage d'un service sur une métaphore que sur la question de l'efficacité visuelle de ce mappage.

Dans un premier temps, nous présentons la notion de métaphore dans un cadre plus général et nous expliquons en quoi elle peut être utile aux visualisations. Puis, nous présentons notre modèle de mondes 3D interactifs et comment il permet de construire des métaphores, pour ensuite discuter de son utilisation pour la visualisation d'informations. Nous décrivons ensuite le processus de mappage d'un service sur une métaphore, pour finir par deux exemples de tels mappages.

2 Métaphores

2.1 Définition

Les métaphores sont des figures de style très largement utilisées dans le langage. [Lakoff80] en donne la définition suivante:

"Une métaphore est une figure de rhétorique, dont l'essence est de comprendre et d'expérimenter un concept grâce à un autre"

Ce procédé a souvent été utilisé dans les interfaces homme-machine et en particulier dans les interfaces graphiques destinées à l'utilisation d'ordinateurs. Ainsi, on peut trouver une grande quantité de travaux, résumés dans [Neale97], qui visent à améliorer et analyser l'utilisation des métaphores dans ce contexte. En effet, ce type d'interface présente l'avantage de permettre à l'utilisateur débutant de manipuler des concepts propres à l'ordinateur par l'intermédiaire de concepts dont il a déjà connaissance. Ce type d'interface facilite donc l'apprentissage de l'utilisateur. L'exemple le plus connu est bien sûr la métaphore du bureau, dont presque tous les ordinateurs sont aujourd'hui pourvus. Un autre avantage de la métaphore est de rendre l'utilisation de l'interface utilisateur plus facile et rapide. Il est par exemple très rapide de déplacer un fichier d'un répertoire à l'autre en le faisant glisser à l'aide de la souris.

La métaphore comporte la plupart du temps un aspect graphique et un aspect interactif. Par exemple, il existe une métaphore où les fichiers sont représentés sous forme d'icônes de feuilles de papier et les répertoires sous forme de dossiers. Ceci constitue une partie de l'aspect graphique de la métaphore. L'aspect interactif provient quant à lui des actions que l'utilisateur peut effectuer sur ces icônes : par exemple, pour déplacer un fichier d'un répertoire à un autre il peut se contenter de faire glisser l'icône du fichier sur l'icône du répertoire de destination.

Dans une interface utilisateur, il est rare de trouver une seule métaphore. La plupart du temps, l'interface est composée de plusieurs métaphores permettant de réaliser toutes les tâches nécessaires. On parle alors de métaphores composites. Par exemple, à côté de la métaphore du bureau on trouve les métaphores des menus, du presse-papiers, etc. D'autre part, il est très rare que les propriétés de la métaphore puissent remplir toutes les tâches nécessaires à l'interface utilisateur. Il est alors nécessaire d'ajouter à l'interface des fonctionnalités non métaphoriques, souvent appelées «fonctionnalités magiques» [Smith87], et qui permettent d'améliorer l'efficacité de l'interface. La majorité des interfaces utilisateur existantes sont des métaphores composites agrémentées de fonctionnalités magiques.

[Martin90] décrit les métaphores comme des mappages de connaissances d'un domaine *source,* familier à l'utilisateur, vers un domaine de *destination,* que l'utilisateur ne connaît pas. En d'autres termes, une métaphore est un ensemble d'associations entre les concepts du domaine source et ceux du domaine de

destination. Par exemple, la métaphore du bureau associe les fichiers du disque dur (concept de destination) à des feuilles de papier (concept source).

2.2 Les métaphores en visualisation

Dans le domaine de la visualisation, un substrat visuel est utilisé pour matérialiser l'ensemble des données que l'on désire observer. Lorsque ces données ont une réalité physique, typiquement en visualisation scientifique, il est possible d'utiliser l'aspect visuel inhérent à cette réalité physique pour créer la visualisation. Dans ce cas, le concept de métaphore n'est pas nécessaire pour générer la visualisation. Par exemple, la visualisation des continents se fera logiquement sur un globe terrestre. Par contre, il peut être tout à fait intéressant d'ajouter des éléments métaphoriques à ce type de visualisation afin d'en augmenter l'efficacité.

En visualisation d'informations, les données ont tendance à être abstraites (par exemple les processus d'une station de travail ou les fichiers d'un disque dur). Il n'y a donc pas de support privilégié pour construire la visualisation comme c'est le cas en visualisation scientifique. L'utilisation de métaphores est alors appropriée pour les raisons évoquées précédemment. Les métaphores utilisées en visualisation ont forcément pour source un domaine où les concepts ont une réalité visuelle que l'utilisateur connaît déjà. Ce domaine peut être par exemple une ville, un paysage, un supermarché ou bien le fameux bureau. Le domaine de destination est constitué par l'ensemble des concepts existant dans les informations que l'on désire visualiser.

A titre d'exemple, si on veut visualiser les processus d'une station de travail (domaine de destination sans réalité physique) sous la forme d'une ville (domaine source), on peut utiliser la représentation d'une maison pour visualiser un processus. Les paramètres visuels de cette maison (taille, couleur, nombre de fenêtres, etc.) serviront à visualiser les paramètres du processus. On a donc défini un mappage des connaissances visuelles du domaine source (une maison et ses attributs visuels) vers le domaine de destination (les processus).

Pour le bon fonctionnement de la métaphore, il est important de bien choisir les paramètres visuels qui vont être utilisés pour visualiser les informations du domaine de destination. Ceci fait l'étude de recherche approfondie, notamment pour automatiser cette tâche [Zhou96].

Selon la définition de la métaphore exposée ci-dessus, les visualisations géographiques ne sont pas métaphoriques. En effet, elles n'empruntent pas de connaissances d'un domaine source pour les utiliser dans le domaine de destination, les concepts de celui-ci contenant déjà leur position géographique (leur réalité physique). Cependant, en visualisation d'informations, les deux types d'informations sont souvent présentes et on obtient donc des visualisations mixtes associant éléments métaphoriques et non métaphoriques. Par exemple, dans les projets de [Eick96], les objets sont métaphoriques mais leur position ne l'est pas car elle reflète leur position dans la réalité.

3 Modèle de mondes 3D interactifs

Le modèle de mondes 3D interactifs permet, entre autres, la définition des aspects visuels et interactifs de métaphores 3D. Il repose sur l'utilisation des modèles que nous avons décrits dans les chapitres précédents : le modèle de visualisation permet de définir l'aspect visuel, tandis que le modèle de navigation ainsi que les composants d'interaction permettent de décrire l'aspect interactif.

3.1 Objectifs

Le premier objectif est de réunir au sein d'un même concept l'aspect visuel et interactif, de manière à faciliter la construction de mondes 3D interactifs en disposant de concepts de haut niveau.

Le modèle partage les mêmes buts que les modèles sur lesquels il s'appuie :

- Définir un modèle ouvert facilitant la création de tout type de monde 3D interactif représentant des informations structurées hiérarchiquement
- Faciliter le mappage visuel de ces informations
- Gérer le caractère dynamique inhérent à la visualisation d'informations dynamiques
- Fournir des interactions définies en fonction de la représentation visuelle, et non des interactions génériques
- Faciliter la définition et la construction des mondes 3D en utilisant des concepts inspirés de l'analyse orientée objet [Meyler88] [Rumbaugh91] [Booch94] permettant la définition d'objets visuels et interactifs que l'on peut assembler facilement afin de définir un monde 3D interactifs

Composant métaphorique (CM)

3.1.1 Définition

Composant métaphorique = composant graphique + composant de navigation + composant d'interaction

Dans le chapitre concernant le modèle de visualisation, nous avons vu comment la représentation visuelle d'un monde 3D était construite à partir de composants graphiques (CG) organisés en hiérarchie. Au chapitre précédent, nous avons vu qu'un composant de navigation (CN) ainsi qu'un composant d'interaction (CI) étaient associés à chaque CG de manière à offrir à l'utilisateur des moyens de navigation et d'interaction spécifiques à la représentation visuelle utilisée.

Le composant métaphorique (CM) permet de réunir un CG et ses CN et CI associés. Un CM se définit donc comme un composant qui a d'une part des propriétés spécifiées par le CG lui permettant de participer à la création visuelle d'un monde 3D, et d'autre part des propriétés spécifiées par le CN et le CI,

procurant des interactions en harmonie avec la type du CG et du contexte dans lequel il se trouve.

La différence entre un CM et un CG, outre le fait que le CM possède des interactions, est que le CM ne peut généralement pas être réutilisé dans une autre contexte contrairement aux CG. En effet, un CG simple représentant un cube peut être utilisé pour visualiser une maison schématique ou bien les étages d'un immeuble, mais un CM maison ne peut pas être utilisé comme un CM étage car ses interactions sont différentes. Le CG n'a pas la connaissance de l'environnement dans lequel il se trouve, alors que le CM, de par la définition de ses interactions, en a la connaissance. Les CM possèdent donc un aspect sémantique qui n'existe pas chez les autres composants.

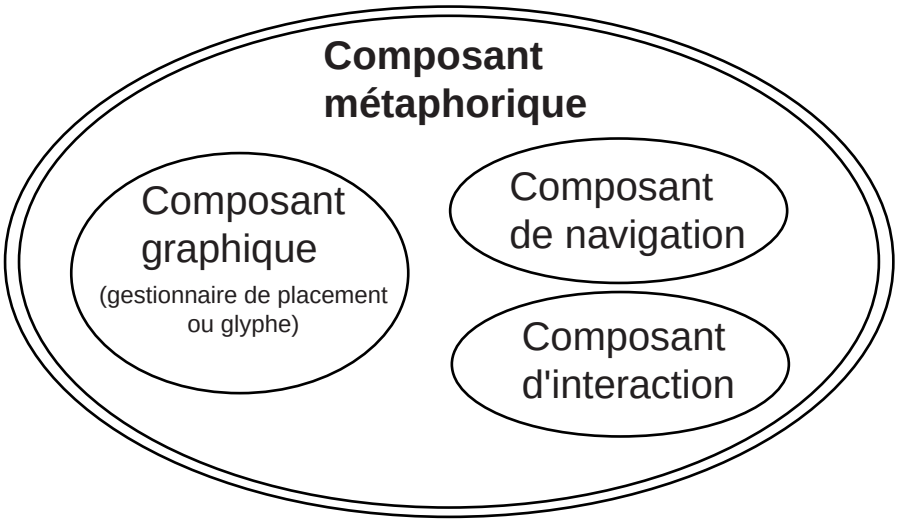


Figure 49 *Composant métaphorique*

3.1.2 Types de composant métaphorique

Le type d'un CM est défini par le type de CG qu'il contient. Il en existe donc deux :

- Le CM glyphe, qui contient un CG de type glyphe
- Le CM gestionnaire de placement (CM-GP), contenant un CG de type gestionnaire de placement

Les deux types de CM ont les mêmes propriétés de structuration en arbre que les CG qu'ils contiennent. En d'autres termes, l'utilisation des deux types de CM permet de construire des hiérarchies de CM dont les nœuds internes sont des CM-GP et les feuilles des CM-glyphes.

3.1.3 Une métaphore : une hiérarchie de composants métaphoriques

Dans le système CyberNet, une métaphore est un type de représentation visuelle 3D (une ville par exemple) associée à des interactions permettant, en particulier, de naviguer dans un monde 3D construit sur la base de cette représentation. Bien que le modèle permette de construire des représentations non métaphoriques, on appelle abusivement métaphore, tout type de monde 3D interactif.

Une métaphore est définie par une hiérarchie de CM. Cette hiérarchie en induit trois autres ayant la même structure :

- Une hiérarchie de CG permettant de construire l'aspect visuel du monde
- Une hiérarchie de CN permettant de réaliser les opérations de navigation
- Une hiérarchie de CI prenant en charge d'autres interactions que la navigation

Dans la pratique, pour construire une métaphore on décide d'abord quel aspect visuel on veut obtenir. Pour cela, on construit des CG et on définit leur organisation hiérarchique. Ensuite on associe un CN à chaque CG en fonction de la représentation visuelle et de l'environnement dans lequel se trouvera ce CG. Enfin, on associe à chaque CG un CI qui spécifie quel type d'interaction sera possible avec ce CG. Le résultat est une hiérarchie de CM. Un exemple de hiérarchie d'une métaphore est présenté à la Figure 50.

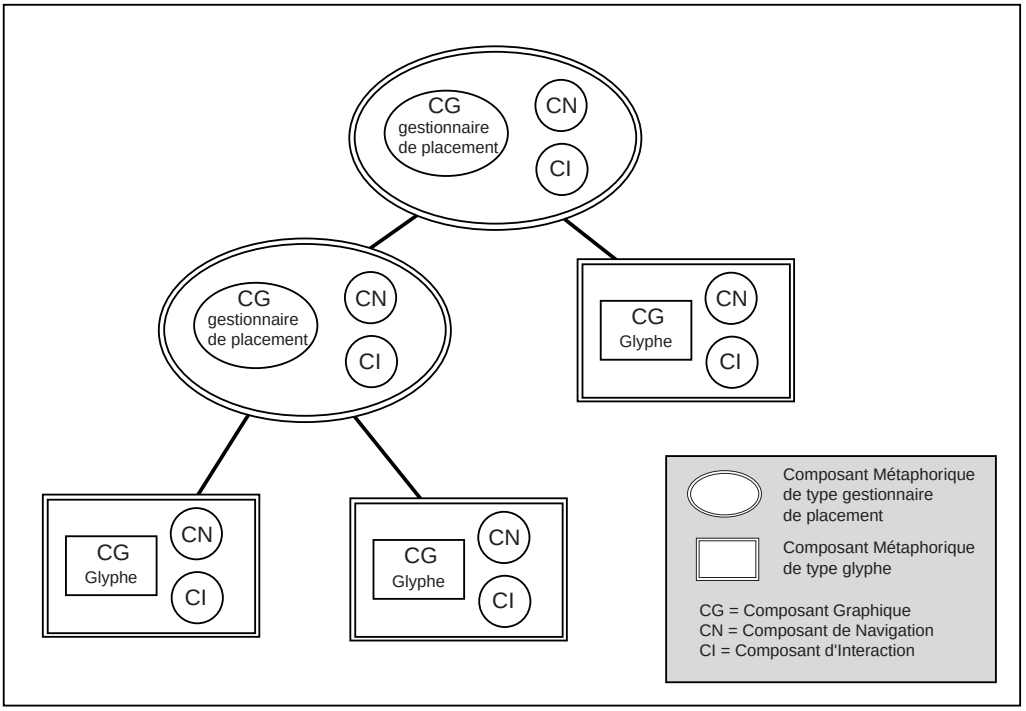


Figure 50 *Hiérarchie de composants métaphoriques*

4 Utilisation de métaphores pour la visualisation 3D d'informations

Comme expliqué précédemment, l'utilisation de métaphores permet d'expérimenter un concept grâce à un autre. L'utilisation de métaphores visuelles est donc tout indiquée afin de visualiser des informations, en particulier abstraites. Mais des questions subsistent : quelle métaphore choisir pour quelles informations et comment doit-on mapper les informations sur les paramètres visuels de la métaphore ? Ces questions sont simplement soulevées car hors du sujet de cette thèse mais quelques éléments de réflexions sont exposés. Des travaux plus approfondis sur le mappage automatique d'informations sur une métaphore sont actuellement réalisés par Cristina Russo Dos Santos dans le cadre du projet CyberNet, et j'invite le lecteur à consulter son futur mémoire de thèse ou les résultats préliminaires [Russo00].

4.1 Choix de la métaphore

Le choix d'une métaphore doit être effectué en fonction de la quantité d'informations à visualiser. En effet, les métaphores peuvent être plus ou moins complexes du point de vue visuel et peuvent donc représenter plus ou moins d'informations. La métaphore choisie doit donc au minimum pouvoir représenter le volume d'informations que l'on désire visualiser.

Un autre critère influant sur le choix d'une métaphore est sa capacité à représenter les informations sous une forme efficace. Le critère d'efficacité est subjectif mais il semble que certains types d'informations soient mieux visualisés à l'aide de certaines métaphores.

4.2 Utilisation des paramètres visuels d'une métaphore

Après avoir choisi une métaphore, il est nécessaire de mapper les différentes informations que l'on désire observer sur les paramètres visuels de la métaphore. Par paramètres visuels, on entend les différents éléments graphiques (e.g. maisons et quartiers dans une ville) et leur position dans l'espace, mais aussi les paramètres visuels propres à chaque élément graphique (e.g. couleur, taille). Dans un premier temps, nous présentons quelques critères possibles pouvant mener au choix d'un élément graphique pour visualiser un ensemble d'informations puis nous discutons brièvement de l'utilisation des paramètres visuels des éléments graphiques.

4.2.1 Choix d'un élément graphique

Le mappage des informations sur les éléments graphiques d'une métaphore doit permettre de transcrire visuellement les relations qui existent entre ces

informations. On a donc essayé d'analyser les relations visuelles au sein d'une métaphore.

Les éléments de réflexion suivants sont tous basés sur la connaissance qu'a l'utilisateur des relations qui existent entre les différents éléments graphiques qui constituent la métaphore. L'utilisateur connaît plusieurs types de relation entre éléments graphiques d'une métaphore.

Un premier type de relation est la relation hiérarchique. Par exemple, dans une métaphore de ville composée de quartiers de maisons et d'immeubles, il existe une relation hiérarchique entre ces éléments : la ville contient les quartiers qui contiennent les maisons et les immeubles.

Entre deux éléments du même type, il existe naturellement une relation de groupe, mais l'apparence visuelle ou la position dans l'espace des éléments peut leur conférer des différences permettant de les comparer. Par exemple, dans une ville, il peut exister des quartiers d'affaires et des quartiers résidentiels visuellement différents car ne comportant pas le même type de bâtiment (des gratte-ciel pour le premier, des maisons pour le second). Le quartier d'affaires suggère l'activité tandis qu'un quartier résidentiel est associé à plus de tranquillité. La position d'un quartier dans une ville peut également rentrer en compte dans sa comparaison avec un autre quartier ou dans l'interprétation de son importance : un quartier en centre ville semblera plus important que le même quartier en périphérie.

L'utilisateur a donc connaissance de certaines relations qui existent entre les différents éléments graphiques d'une métaphore. Ces connaissances peuvent donc être exploitées de manière à montrer visuellement les relations qui existent entre les différentes informations qui doivent être supervisées.

4.2.2 Utilisation des paramètres visuels des éléments graphiques

Chaque élément graphique permet de visualiser une ou plusieurs informations atomiques par l'intermédiaire de ses propres paramètres visuels (par exemple la couleur et la taille). Encore une fois, l'utilisation de tel type de paramètres visuels pour visualiser tel type d'informations dépasse le cadre de cette thèse, mais nous présentons brièvement deux règles d'utilisation des paramètres visuels.

Premièrement, il est préférable que l'utilisation des paramètres visuels des éléments d'une même métaphore soit cohérente, c'est à dire qu'un type de paramètre soit toujours utilisé pour visualiser le même type d'informations. Par exemple, si la couleur d'un élément permet de visualiser une quantité, il n'est pas cohérent que la couleur d'un autre type d'élément soit utilisée à des fins d'identification. De telles incohérences peuvent troubler l'utilisateur et donc réduire l'efficacité de la visualisation.

Deuxièmement, nous avons déjà dit que la valeur ajoutée des métaphores en visualisation est qu'elles permettent d'exploiter les connaissances qu'a déjà l'utilisateur de la métaphore. Ceci est valable si les éléments graphiques sont identifiables par l'utilisateur. Il faut donc prendre soin que l'utilisation des paramètres visuels des éléments graphiques laissent ces derniers toujours

identifiables. Par exemple, dans une ville les gratte-ciel sont généralement plus hauts que larges. Si on mappe une grande valeur sur la largeur et une petite valeur sur la hauteur d'un élément graphique censé représenter un gratte-ciel, celui-ci risque de ressembler à une plate-forme, ce qui perturbe l'identification du gratte-ciel par l'utilisateur et rend donc la métaphore inutile.

5 Mappage d'un service sur une métaphore

Dans cette partie, nous étudions comment est réalisé le processus menant à la construction automatique d'un monde 3D interactif à partir des informations contenues dans un service, ce monde étant basé sur une métaphore prédéfinie par une hiérarchie de CM.

5.1 Construction de la visualisation

Nous allons maintenant étudier comment sont mappées les informations contenues dans le service sur les paramètres visuels d'une métaphore. La base de ce processus est définie par des associations entre les différents éléments de la métaphore et du service.

La première étape consiste à choisir la métaphore qui va être utilisée pour visualiser les informations. Le reste du processus de mappage s'effectue en deux étapes :

- Le mappage hiérarchique : dans cette étape, on associe un CM de la métaphore à chaque organisateur et à chaque entité à visualiser du service
- Le mappage visuel : dans cette étape, on spécifie comment les informations d'un élément du service vont être visualisées grâce aux paramètres visuels du CM associé.

5.1.1 Mappage hiérarchique

Le mappage hiérarchique consiste à associer un CM à chaque organisateur et à chaque entité (destinée à être visualisée) du service. Les CM disponibles sont ceux qui composent la métaphore choisie. Pour réaliser cette étape, on définit des règles permettant d'associer chaque élément que l'on peut trouver dans le service à un CM. Il existe un type de règle pour les organisateurs et un autre pour les entités.

Les informations contenues dans les organisateurs sont ses organisateurs fils et ses entités. Ces informations sont mappées sur le gestionnaire de placement contenu dans un CM-GP. D'où une règle définissant le mappage d'un organisateur sur un CM-GP.

- (type de l'organisateur, position dans l'arbre du service) → type du CM-GP

On peut noter que le critère de position dans l'arbre est indispensable, car un élément peut être présent plusieurs fois dans un service et il faut donc se laisser la possibilité de le visualiser différemment. Pour la même raison, ce critère est également présent dans la règle définissant l'association d'une entité avec le glyphe d'un CM-glyphe permettant de visualiser l'entité et ses attributs :

- (type de l'entité, position dans l'arbre du service) → type du CM-glyphe

Pour construire une métaphore 3D, il faut donc définir une association pour chaque entité et organisateur du service. Le choix des associations est défini de façon à répercuter visuellement les relations qui existent entre les organisateurs et les entités du service. Ces relations sont définies par la structure d'arbre du service, et les relations entre les CM par la structure d'arbre de la métaphore. Les associations sont donc assez logiques : on essaye de réaliser une correspondance entre les relations dans le domaine des informations et celles existant dans le domaine visuel. Ainsi, les relations entre les informations sont conservées du point de vue visuel. Des exemples de mappage hiérarchique sont donnés dans la section suivante.

5.1.2 Mappage visuel

A la suite du mappage hiérarchique, chaque organisateur et chaque entité sont associés à un CM. Le CM choisi doit donc représenter visuellement les informations contenues dans les élément du service. Des exemples de tels mappages sont donnés dans la section suivante.

Les informations contenues dans les organisateurs qui doivent être mappés sont ses organisateurs fils et ses entités destinées à être visualisées. Ces informations doivent être mappées sur le placement dans l'espace effectué par le gestionnaire de placement associé. Dans certains cas, ce mappage peut être réalisé sans contraintes mais il est quelquefois important de pouvoir classer les entités et les fils de l'organisateur selon différents critères et de mapper ce classement sur la position dans l'espace de chacun, comme nous l'avons vu dans le chapitre concernant le modèle de visualisation.

Pour les entités, les informations à mapper sont les attributs de l'entité. Ces attributs doivent être mappés sur les paramètres visuels du glyphe associé. Généralement, on associe un paramètre visuel à chaque information contenue dans l'entité, mais il peut arriver qu'une information ne puisse pas être représentée par manque de paramètres visuels. Le travail consiste donc à spécifier comment sont calculées les valeurs des paramètres visuels du glyphe à partir des informations associées.

6 Exemples

Dans cette partie, nous présentons deux exemples de mappage de services sur des métaphores afin voir, *in situ*, comment ce processus est réalisé.

Notez que dans cette partie, les figures représentant des services sont simplifiées dans un soucis de clarté : seules les informations qui sont destinées à être mappées sont visualisées sur les figures, ainsi les requêtes ne sont pas présentes, de même que les entités permettant la construction du service.

6.1 Visualisation d'un service de système de fichiers

La visualisation de systèmes de fichiers, spécialement pour les grandes hiérarchies (plusieurs centaines, plusieurs milliers voire plusieurs dizaines de milliers d'objets à représenter), pose de nombreux problèmes de concision et d'exhaustivité. Nous présentons ici un exemple de mappage d'un service de système de fichiers sur une métaphore permettant d'apprécier la structure hiérarchique d'un tel système, ainsi que des informations liées à chaque répertoire le constituant.

Service

Le service visualisé est constitué de deux types d'organisateur (« répertoire » et « sous-répertoires ») et d'un type d'entité (« répertoire »), comme représenté schématiquement à gauche sur la Figure 51. Les fichiers ne sont pas pris en compte par le service dans un souci de concision; en revanche les entités de type « répertoire » contiennent des informations concernant le type de fichiers prépondérant dans le répertoire qu'elles conceptualisent (fichiers image, fichiers texte, fichiers exécutables, etc.). De plus, elles contiennent également des informations liées à leur date de dernière modification, leur nom, leur propriétaire, ainsi que leur taille.

Métaphore

La métaphore utilisée est inspirée de [Andrews97], elle représente des boîtes empilées les unes sur les autres donnant naissance à des sortes de pyramides (cf. images sur la Figure 52). Elle est constituée de trois types de composant métaphorique : le premier contient un glyphe représentant une boîte, le deuxième contient un gestionnaire de placement de type « échiquier », et le troisième un autre gestionnaire de placement mais de type « pile ». Ces composants graphiques sont organisés hiérarchiquement comme décrit à la Figure 51.

Mappage hiérarchique

Les règles de mappage hiérarchique permettent d'associer à chaque élément du service un CM de manière à créer une hiérarchie de CM décrivant le monde 3D interactif. La Figure 51 illustre les règles appliquées dans le cadre de cet exemple, qui sont au nombre de trois :

- (organisateur « répertoire », position indifférente) → CM « pile »
- (organisateur « sous-répertoire », position indifférente) → CM « échiquier »
- (entité « répertoire », position indifférente) → CM « boîte »

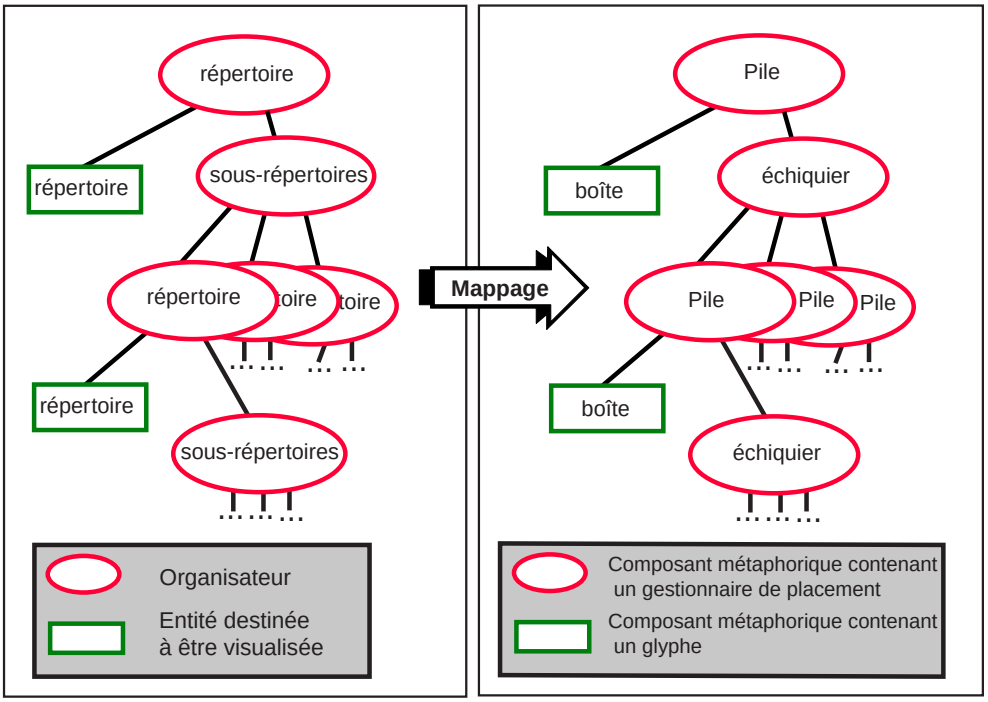


Figure 51 *Mappage hiérarchique d'un service de supervision d'une structure hiérarchique de répertoire sur une métaphore de pyramides*

Mappage visuel

Le CM « échiquier » ne possède aucun paramètre visuel, la position des différents éléments sur l'échiquier n'apporte donc aucune information. Néanmoins, étant donné que ce CM place les différents sous-répertoires d'un répertoire, il pourrait être intéressant d'utiliser un attribut de chaque sous-répertoire pour les classer sur l'échiquier.

Le CM « pile » n'accepte pas non plus de paramètres visuels, en revanche il est défini pour placer les CM contenant des glyphes en dessous des CM contenant des gestionnaires de placement. Cet ordre est particulièrement important car s'il n'est pas respecté, on pourrait obtenir des pyramides inversées qui ne constitueraient pas une visualisation réussie.

Le CM « boîte » acceptent plusieurs paramètres visuels :

- un texte (affiché lorsque l'utilisateur est suffisamment près du CM)
- une couleur
- une hauteur

Le texte permet d'afficher le nom du répertoire de l'entité associé, et la hauteur la taille de ce même répertoire. Comme illustré à la Figure 52, la couleur peut être utilisée de différentes manières de façon à générer des visualisations permettant d'analyser la structure hiérarchique de répertoire selon un critère de propriétaire, de dernière date de modification, ou selon le type de fichiers prépondérants dans chaque répertoire.

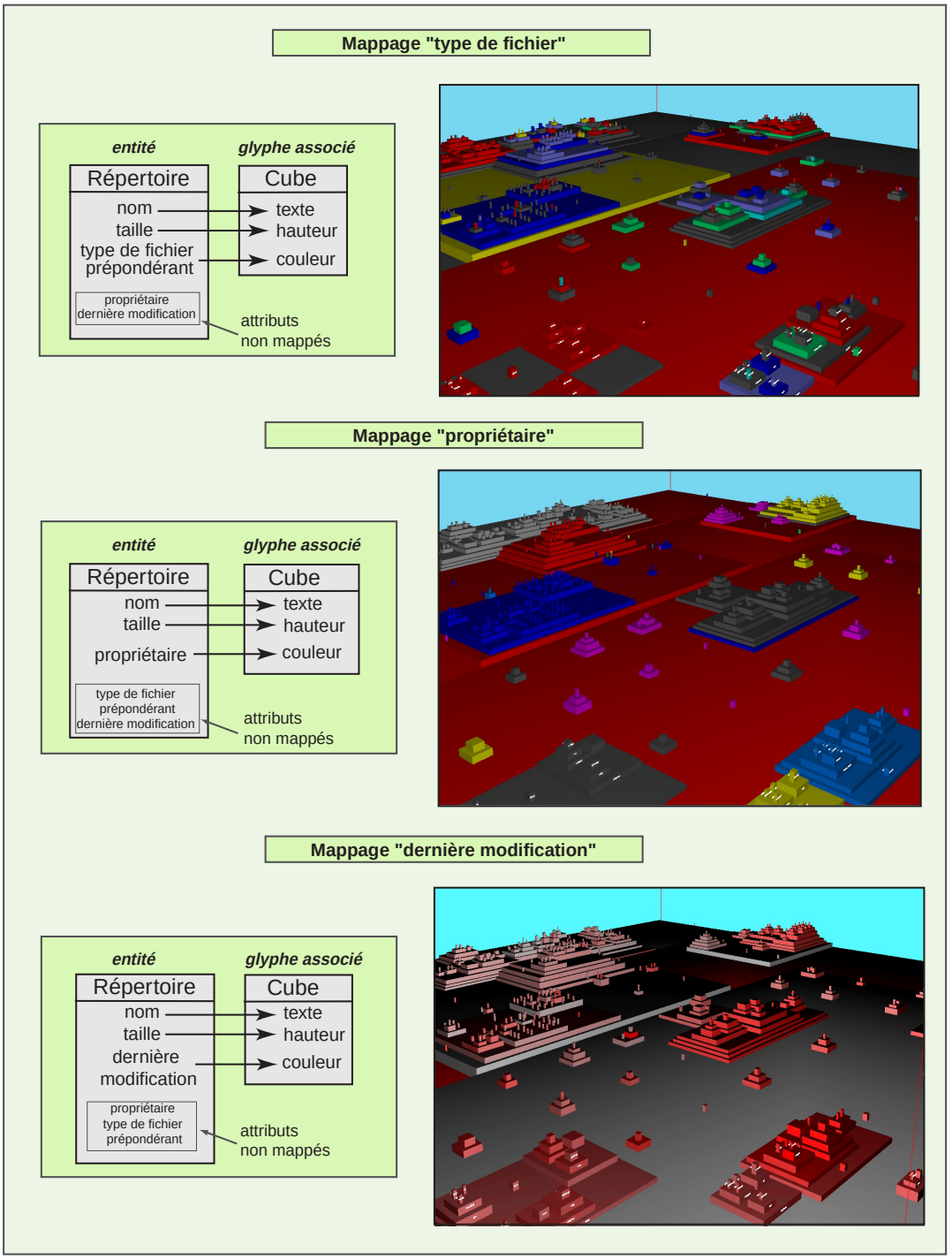


Figure 52 *Différentes possibilités de mappage visuel des informations d'entités de type « répertoire » sur des glyphes de type « boîtes » appartenant à une métaphore de pyramides*

6.2 Visualisation d'un parc d'ordinateurs

Le but de cette visualisation est d'exploiter la structure hiérarchique d'une métaphore de ville afin de représenter un parc d'ordinateurs, ainsi que leurs utilisateurs et leurs processus. Cet exemple est illustré par la Figure 53 et la Figure 54.

Service

Nous reprenons ici un service qui a été décrit dans les exemples du chapitre 2 concernant le modèle de données. Plusieurs types d'organisateurs existent qui sont représentés sur la gauche de la Figure 53. Pour plus de détails les concernant nous laissons le lecteur consulter leur description complète au chapitre 2. Les types d'entités visualisées dans ce service sont les ordinateurs, les utilisateurs et les processus.

Métaphore

La métaphore utilisée pour visualiser ce service correspond visuellement au modèle de ville que nous avons décrit dans les exemples du chapitre 3 concernant le modèle de visualisation. Il existe huit types de composants métaphoriques permettant de construire cette métaphore, comme on peut le voir à la Figure 53. Les CM « ville », « quartier2 » et « quartier3 » intègrent tous le même gestionnaire de placement de type échiquier, et c'est dans leurs composants de navigation et d'interaction qu'ils diffèrent. Il en est de même pour les CM « quartier » et « immeuble2 », qui intègrent tous deux un gestionnaire de placement de type pile. Les trois CM restants intègrent des glyphes qui sont visibles à la Figure 54.

Mappage hiérarchique

La Figure 53 illustre le mappage hiérarchique du service sur la métaphore. Grâce à ce mappage, les relations existantes dans le service sont visuellement représentées dans le monde 3D. Par exemple, l'organisateur « netgroup », qui a comme fils des organisateurs « ordinateur », est mappé sur un CM qui va placer ses fils, des CM « quartier » issus du mappage des « ordinateurs », sur un échiquier.

Les différentes associations (représentées visuellement à la Figure 53) sont représentés dans le Tableau 10

Mappage visuel

Comme dans l'exemple précédent, les gestionnaires de placement de type « échiquier » et « pile » ne possèdent aucun paramètre visuel permettant de spécifier quelle va être la position de chaque fils (dans le but de les classer par exemple). Une exception est celle du gestionnaire de placement de type « pile » intégré dans le CM « quartier », qui positionne le glyphe « trottoir » sous le CM « quartier » de façon à ne pas créer une ville à l'envers.

Elément du service	**Position dans le service**	**CM utilisé**
Organisateur « netgroup »,	indifférente	CM « ville »
Organisateur « ordinateur »,	indifférente	CM « quartier »
Organisateur « utilisateurs »,	indifférente	CM « quartier2 »
Organisateur « utilisateur spécifique »	indifférente	CM « quartier3 »
Organisateur « processus de l'utilisateur»,	indifférente	CM « immeuble2 »
Entité « ordinateur »,	indifférente	CM « trottoir »
Entité « utilisateur »,	indifférente	CM « immeuble1 »
Entité « processus »,	indifférente	CM « étage »

Tableau 10 *Association entre éléments du service et composant métaphorique*

Le mappage des attributs des entités sur les paramètres visuels des glyphes est résumé à la Figure 54. Comme on peut le voir, la couleur est utilisée dans tous les glyphes pour représenter la consommation CPU de l'entité associée. Quant à la taille, elle est utilisée pour visualiser la quantité de mémoire utilisée.

Une exception est le mappage des attributs des entités « ordinateurs » sur les glyphes « trottoir » : dans les premiers tests de cette visualisation, la mémoire utilisée par l'ordinateur était mappée sur la taille du trottoir, mais la visualisation résultante n'était pas très efficace, la hauteur de certains trottoirs masquant d'autres quartiers. Nous avons donc décidé de fixer la hauteur des trottoirs. Le résultat est que l'on ne visualise pas la quantité de mémoire utilisée ou encore disponible sur un ordinateur. Or, cette information est fort utile dans le cadre de ce service et nous pensons que cette information peut et doit être visualisée (par exemple en utilisant un glyphe d'un autre type de bâtiments positionné sur l'échiquier où sont représentés les utilisateurs et leurs processus).

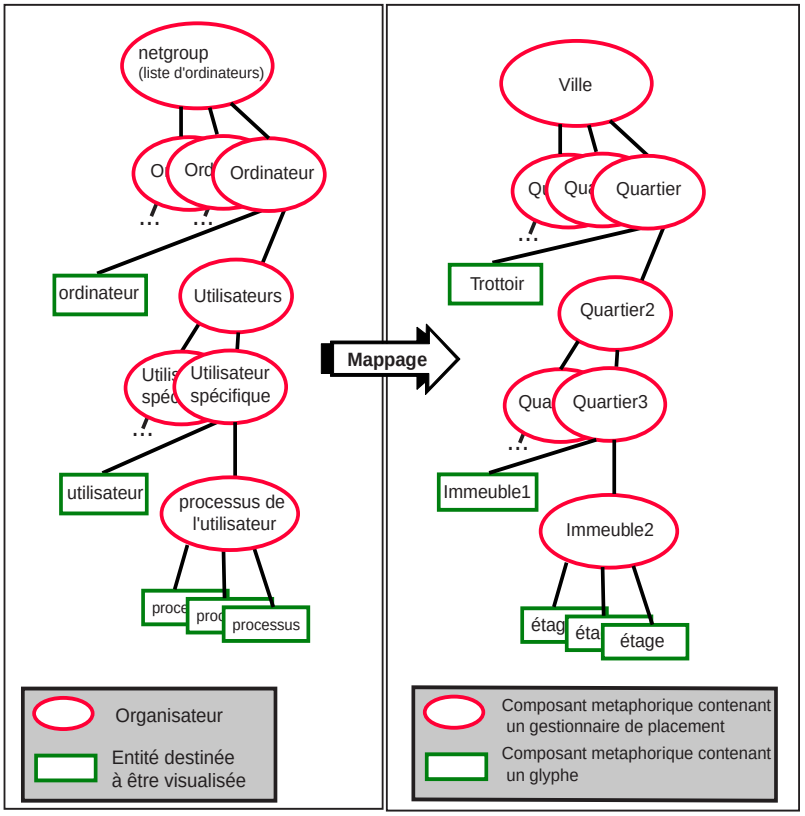


Figure 53 *Mappage hiérarchique d'un service de supervision d'ordinateurs sur une métaphore de ville*

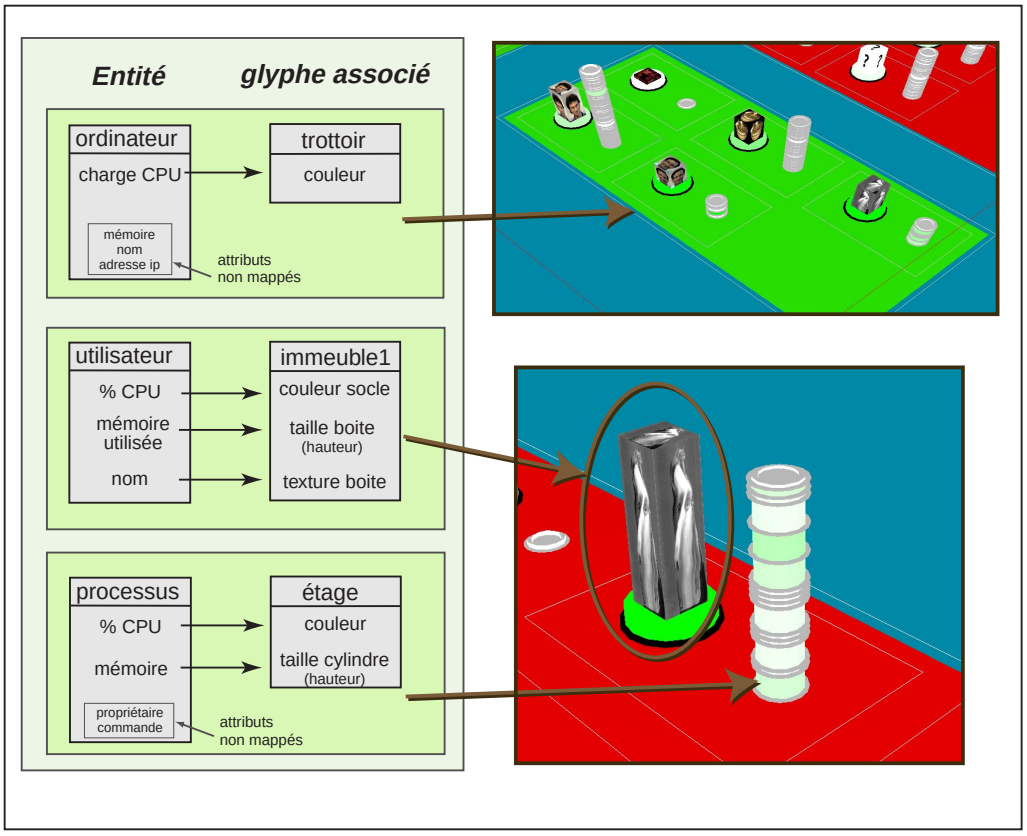


Figure 54 *Mappage visuel des informations de diverses entités d'un service de supervision d'ordinateurs sur des glyphes d'une métaphore de ville*

7 Conclusion

Dans ce chapitre, nous avons décrit un modèle de mondes 3D interactifs basé sur la notion de composants métaphoriques organisés hiérarchiquement, ces composants exploitant les modèles de visualisation et de navigation présentés aux chapitres précédents. L'exploitation de ce modèle permet de décrire des métaphores, c'est à dire des types de monde 3D interactif, où les interactions sont définies en fonction de la représentation visuelle, et non d'une façon générique.

Nous avons également expliqué comment ce modèle était utilisé de manière à automatiser la construction de mondes 3D interactifs permettant de visualiser des informations structurées hiérarchiquement au sein de services. La méthode employée consiste à décrire des associations entre les éléments d'un service et ceux d'une métaphore, et à définir comment les paramètres visuels d'un élément de la métaphore sont exploités afin de visualiser les informations contenues dans l'élément du service associé.

Une perspective importante dans ce procédé de mappage est de pouvoir, à terme, automatiser également la création des associations qui sont pour le moment définies à la main, cette automatisation nécessitant une caractérisation des informations ainsi que des paramètres visuels utilisés dans les mondes 3D interactifs.

6 Implémentation de la plate-forme logicielle CyberNet

1 Introduction

Dans ce chapitre, nous présentons la plate-forme logicielle qui a été réalisée pendant la durée de cette thèse et qui permet de mettre en œuvre les différents modèles que nous avons vus dans les précédents chapitres. Cette plate-forme permet de collecter des informations distribuées, de structurer ces informations sous forme de services, puis de construire, à partir de ces derniers, des mondes 3D interactifs visualisés par l'intermédiaire de navigateurs Web [Abel00][Abel00b].

Le développement de la plate-forme logicielle a été motivé par les buts suivants :

- **Validation des modèles** : le but principal était de valider les différents modèles présentés dans cette thèse
- **Réutilisation**: Un autre but était de développer un framework orienté objet permettant, grâce à la réutilisation de différents composants, de tester rapidement de nouveaux types de services et de visualisations

Dans un premier temps, nous présentons l'implémentation des différentes couches nécessaires à la réalisation de la plate-forme. Elles sont au nombre de trois :

- *La couche de collecte,* qui permet de collecter des informations distribuées sur un réseau
- *La couche de structuration,* qui permet de structurer les informations sous forme de service
- *La couche de visualisation et d'interaction* qui construit un monde 3D interactif à partir d'un service et gère les interactions avec l'utilisateur.

Au cours de la description de ces couches, nous présentons les contraintes imposées par le caractère dynamique des informations et quelles ont été les stratégies adoptées pour y faire face.

Nous présentons ensuite la manière dont ces couches sont distribuées sur le réseau, avant d'exposer les technologies utilisées par cette plate-forme.

2 Collecte des informations distribuées

Dans notre système, les informations nécessaires à la construction d'un service peuvent être distribuées sur le réseau. Il est donc nécessaire de disposer d'un mécanisme permettant au module chargé de construire un service de collecter ces informations. Deux approches sont possibles :

- Le mécanisme dit de « pull » qui consiste à aller chercher explicitement les informations
- Le mécanisme dit de « push » qui consiste à déléguer ce travail à des agents auxquels il est possible de s'abonner

Etant donné la nature dynamique des informations utilisées dans notre système et notre volonté de construire une visualisation quasi-temps réel de ces informations, il est nécessaire de collecter les informations à intervalles assez courts de manière à savoir si les informations ont changé. L'utilisation d'un mécanisme de type « pull » entraîne une solution où toutes les requêtes permettant la collecte des informations sont centralisées en un même endroit. Le volume d'information à collecter pouvant être important, cette solution serait très coûteuse pour un seul programme. De plus, cette approche surchargerait également le réseau puisqu'il serait nécessaires d'interroger régulièrement, par l'intermédiaire du réseau, toutes les sources d'informations de manière à savoir si les informations ont changé.

2.1 Les informateurs

Nous avons donc opté pour une collecte des informations utilisant un mécanisme de type « push » à base d'agents permettant de résoudre les deux problèmes. Nous avons choisi d'utiliser des agents spécialisés dans la collecte d' un ou de plusieurs types d'informations, que nous avons appelés *informateurs*. De cette façon, la charge de travail liée à la collecte des informations est répartie sur différents informateurs, chacun s'occupant de collecter tel ou tel type d'informations.

Afin de ne pas surcharger le réseau, ces informateurs s'exécutent sur l'ordinateur où se trouvent ces informations ou, si ce n'est pas possible, sur l'ordinateur engendrant le moins d'utilisation de ressources réseau pour accéder aux informations. L'utilisation du réseau est ainsi optimisée car les opérations les plus coûteuses, c'est à dire la collecte de toutes les informations pour connaître leur changement, sont réalisées en local ou très près de la source.

Le rôle des informateurs est de collecter des informations dynamiques. Un client intéressé par les informations d'un informateur doit s'abonner à celui-ci comme illustré à la Figure 55. Par la suite, l'informateur enverra à ce client des informations ainsi que leur mise à jour selon la politique de collecte que lui a fournie le client. Cette politique de collecte contient les types d'informations désirés par le client ainsi que des critères relatifs à la mise à jour de ces informations. Par exemple, dans un contexte de supervision de réseau, un client peut être intéressé par les informations du type « utilisateur » sur la machine « ocean.eurecom.fr » et souhaiter que la mise à jour de ces informations lui soit envoyée toutes les 3 secondes. L'envoi des mises à jour est bien sûr effectué seulement si les informations ont changé depuis la dernière fois, toujours afin de minimiser l'utilisation du réseau.

Créateurs d'entités

Dans le modèle d'information décrit au chapitre 2, nous avons vu que les informations étaient regroupées au sein d'entités représentant des concepts, par exemple un utilisateur. Ce sont les informateurs qui créent les entités à partir des informations atomiques collectées et selon la définition des entités définie dans le service. Ainsi, la couche de structuration peut s'abonner aux informateurs en leur demandant de leur envoyer tel ou tel type d'entités qui vont servir à la construction de service.

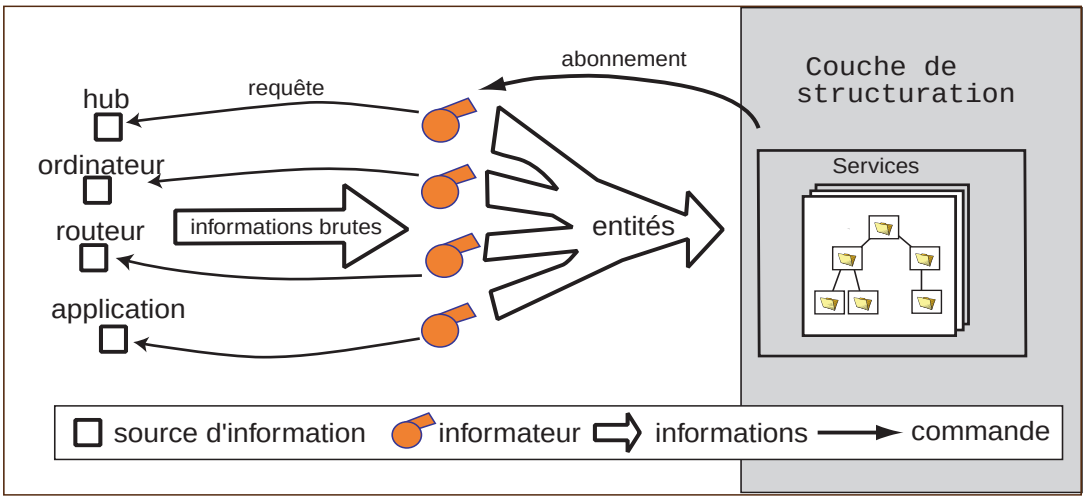


Figure 55 *Couche de collecte des informations*

3 Structuration des informations sous forme de services

Le but de cette couche est de structurer les informations, c'est à dire les entités, sous la forme d'un service. Ce module implémente donc le modèle décrit au chapitre 2 afin de créer les services. Nous allons voir que les contraintes

principales dans l'implémentation de cette couche sont dues, encore une fois, à la nature dynamique des informations ainsi qu'à la nécessité de partager des entités entre plusieurs services.

3.1 Partager les entités

Une des contraintes majeures dans l'implémentation de la couche de structuration est que plusieurs services peuvent exister en même temps, et qu'ils peuvent avoir besoin des mêmes entités. Par exemple, deux services de supervision des ordinateurs et des utilisateurs d'un réseau peuvent contenir les mêmes entités mais organisées différemment, le premier service présentant une vue centrée sur les ordinateurs du réseau, c'est à dire les ordinateurs avec leurs utilisateurs, et le deuxième service une vue centrée sur les utilisateurs, c'est à dire les utilisateurs avec les ordinateurs auxquels ils sont connectés. Un superviseur pouvant visualiser ces deux services en même temps, il est nécessaire de garantir une cohérence entre les informations contenues dans les deux services.

Pour garantir cette cohérence et optimiser le système en ne dupliquant pas une même entité, les entités n'appartiennent pas à un service en particulier. Ainsi, les services peuvent partager les entités qu'ils exploitent en commun. Les entités sont donc des objets autonomes et un service possède, par l'intermédiaire de ses relations, seulement des références à des entités.

La nature dynamique des informations, et donc des entités, entraîne un problème similaire à celui que nous avons vu dans la partie précédente sur la collecte des informations. En effet, un service doit constamment savoir si les attributs des entités, dont qu'il connaît déjà les références, ont changé, voire même si des entités sont mortes. Si cette tâche est effectuée par le service, cela signifie que le service doit constamment vérifier ses entités, et si deux services utilisent les mêmes entités cette tâche sera réalisée deux fois. Afin d'optimiser cette tâche, la communication entre une entité et un service utilise une méthode basée sur la « design pattern » observateur [Gamma96] : le service connaissant la référence d'une entité s'y abonne, et par la suite l'entité lui signale dès qu'un changement survient. Ainsi, c'est l'entité qui se charge de la tâche consistant à vérifier si ses attributs ont évoluées, et si un changement survient,c'est l'entité qui en informe les services qui se sont abonnés à cette entité.

3.2 Un référentiel pour les entités

Le service, par l'intermédiaire de ses relations, doit pouvoir obtenir les références des entités voulues. Pour cela, il est nécessaire d'utiliser un référentiel où sont stockées les références de toutes les entités actuellement dans le système. Ce référentiel est utilisé par les relations pour connaître les références des entités qui satisfont leurs paramètres. L'utilisation du référentiel est illustrée à la Figure 56 et la Figure 57.

Dans le chapitre 2, nous avons vu que les paramètres d'une relation permettaient de spécifier les entités impliquées dans celle-ci. Ces paramètres permettent de créer une relation contenant :

- toutes les entités d'un même type
- toutes les entités d'un même type ayant un ou plusieurs champs correspondant à un critère
- toutes les entités ayant un ou plusieurs champs correspondant à un critère.

Le référentiel doit pouvoir gérer des requêtes, venant des relations, avec ces paramètres. Une entrée dans le référentiel doit donc contenir la référence, le type et les informations de l'entité.

De par la nature dynamique du système, des entités se créent en continu. Afin que le service soit à jour , les relations doivent contenir les références de toutes les entités actuellement dans le système. Cette contrainte soulève un problème similaire à celui rencontré lors de la collecte des informations : il s'agit de savoir si les attributs d'une entité ont changé. En effet, pour savoir si de nouvelles entités sont impliquées dans une relation, celle-ci pourrait interroger le référentiel à de très courts intervalles, mais cette tâche serait très lourde et souvent inutile car de nouvelles entités n'auraient pas été créées depuis la dernière interrogation. Encore une fois, un système d'abonnement doit donc être mis en place de manière à optimiser le système. Ainsi, lorsqu'une relation se crée, elle interroge le référentiel pour connaître les références des entités lui correspondant et, en plus, s'abonne au référentiel afin de recevoir automatiquement les futures entités qui correspondront aux critères de cette relation. Un exemple de la création d'une entité correspondant aux critères d'une relation est illustré à la Figure 57.

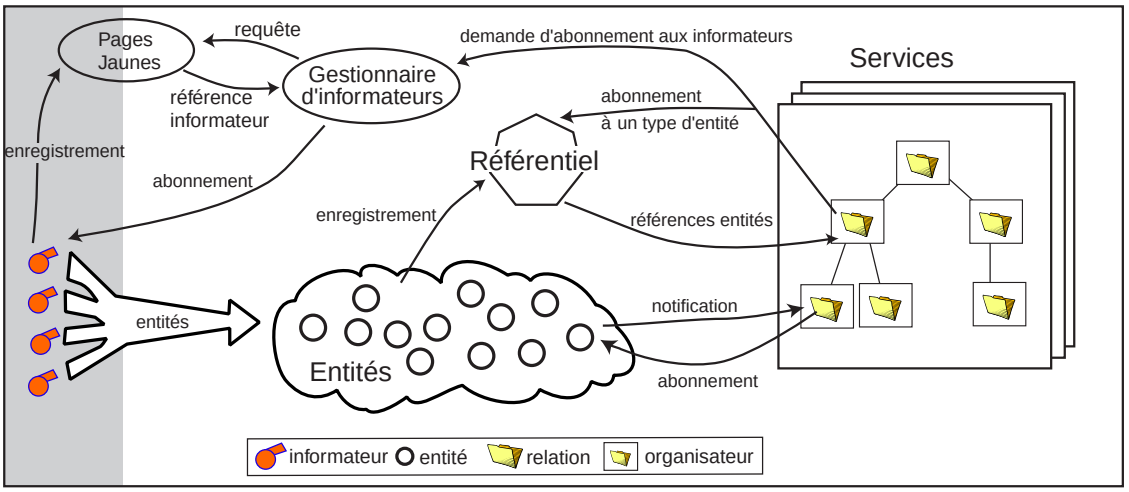


Figure 56 *Couche de structuration sous forme de service*

3.3 Gestion des informateurs

Jusqu'à présent, nous avons présenté comment les entités pouvaient être partagées grâce à l'utilisation d'un référentiel, mais nous n'avons pas abordé le problème lié à l'abonnement aux informateurs. En d'autres termes, comment le système sait-il à quel informateur s'abonner et d'autre part comment le système évite-t-il un abonnement redondant à un informateur.

Le problème de la localisation des informateurs est résolu grâce à un mécanisme de pages jaunes, appelé aussi « trading service ». Ce mécanisme permet à un objet localisé sur un réseau de s'enregistrer dans les pages jaunes avec des paramètres permettant de l'identifier. Les informateurs exploitent cette possibilité en s'enregistrant dans les pages jaunes avec des paramètres spécifiant quelles entités ils peuvent fournir aux abonnés. Pour trouver l'informateur pouvant lui fournir les entités voulues et s'y abonner, le système n'a plus qu'à interroger les pages jaunes afin de recevoir la référence de l'informateur comme illustré à la Figure 56.

Un deuxième problème lié à l'abonnement aux informateurs survient lorsque deux relations existantes sont intéressées par des mêmes entités. En effet, les relations sont à l'origine des entités existantes dans le système puisque ce sont elles qui spécifient quelles entités sont nécessaires à la bonne marche des services. Lorsqu'une relation est créée, le système doit prendre soin de s'abonner aux informateurs fournissant les entités qui correspondent aux paramètres de la nouvelle relation. Dans le cas où deux relations nécessitent des entités similaires, si le système ne prend pas de précautions, il risque de s'abonner deux fois aux informateurs fournissant les entités partagées par les deux relations, et donc que les entités existent en double dans le système. Pour éviter ce problème, nous avons utilisé un *gestionnaire d'informateurs* au sein du module de structuration. Toutes les requêtes des relations sont non seulement envoyées au référentiel, mais aussi à ce gestionnaire d'informateurs qui juge s'il est nécessaire de s'abonner à de nouveaux informateurs ou si les entités réclamées par la relation sont déjà présentes dans le système (cf. Figure 56).

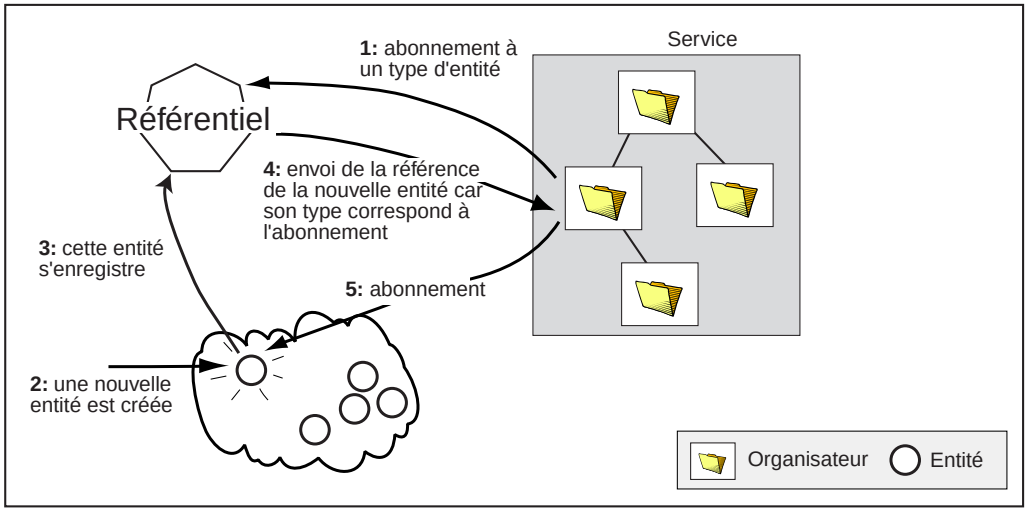


Figure 57 *Réception de la référence d'une entité par une relation*

4 Visualisation & interaction

Le rôle de la couche de visualisation, illustrée à la Figure 58, est de construire un monde 3D interactif à partir d'un service. Les différentes tâches nécessaires pour atteindre cet objectif sont :

- La construction d'une hiérarchie de composants métaphoriques (CM) représentant le modèle du monde 3D interactif, comme défini au chapitre 5
- Le calcul de la position et de l'apparence de tous les composants graphiques (CG) associés à chaque CM existant dans la hiérarchie susnommée
- La gestion de l'interactivité permettant à l'utilisateur d'exploiter la visualisation

4.1 Visualisation dans un navigateur Web

Une contrainte dans l'implémentation de cette couche est que la visualisation du monde 3D doit se faire à l'intérieur d'un navigateur Web (cf. Figure 59) , et étant donné que le client Web n'est pas forcément très puissant, nous avons décidé de réduire au minimum la charge du travail réalisé par le client.

Le travail du client se réduit donc au minimum nécessaire, c'est à dire créer la visualisation en elle-même à l'aide d'une technologie 3D et gérer une partie de l'interactivité, ce qui nécessite déjà beaucoup de puissance de calcul. Le reste des opérations, c'est à dire la construction de la hiérarchie des CM, le calcul des attributs visuels de ceux-ci et une partie de l'interaction est effectué sur un serveur. Nous allons voir que ce choix entraîne d'autres décisions, notamment dans le calcul des paramètres du monde 3D.

4.2 Construction de la hiérarchie de CM

La hiérarchie des CM constitue la description du monde 3D. Comme décrit au chapitre 5, cette hiérarchie est construite à partir d'un service grâce à des lois associant un CM à chaque élément du service destiné à être visualisé. La construction de cette hiérarchie se fait au même rythme que celle du service. En d'autres termes, la création d'u nouvel élément du service entraîne aussitôt celle de son CM associé.

4.3 Calcul des positions et apparences des CG

Lors de la construction de la hiérarchie des CM aucun calcul des positions et des apparences des CG associés n'est réalisé, seule la structure est construite, tâche ne consommant pas beaucoup de puissance de calcul contrairement à la première. En effet, le calcul de tous les indicateurs visuels d'un monde 3D peut être assez coûteux, spécialement si la hiérarchie des CM est importante.

Ces calculs ne sont utiles que si leurs résultats sont envoyés au client Web créant la visualisation en elle-même. Or il faut prendre en compte le fait que les communications entre le serveur et le client Web sont relativement coûteuses et que le client n'a pas forcément la puissance de calcul nécessaire pour lui permettre de gérer la réception des données, de mettre à jour le monde 3D à partir de ces données et également gérer l'interactivité.

Le calcul des positions et apparences des CG n'est donc pas réalisé à chaque fois qu'une modification survient dans la hiérarchie des CM ou dans les informations représentées par les CG. Au contraire, cette opération est réalisée à intervalles réguliers, ces intervalles dépendant de la puissance du client. Plus le client est puissant et plus il est possible de réduire cet intervalle, mais il faut également tenir compte qu'il est inutile que cet intervalle soit inférieur à la fréquence de mise à jour du service et donc de la hiérarchie des CM.

Le calcul des positions et des apparences des CG est effectué par l'algorithme en deux étapes décrit au chapitre 3 avec quelques optimisations de manière à n'envoyer au client Web que des mises à jour et non la description complète du monde 3D à chaque fois. Ces optimisations sont réalisées en comparant ce qui a été envoyé au client Web lors des calculs précédents. Ces comparaisons sont réalisées sur des critères visuels, c'est-à-dire qu'elles n'ont lieu que si les paramètres visuels ont changé et non si les informations ont changé. Ceci est important car le changement d'une information sans que le paramètre visuel correspondant ne soit modifié. Un exemple simple est le mappage d'une valeur numérique continue sur une échelle de couleur discrète, dans ce cas le changement de la valeur n'entraîne pas forcement un changement de couleur.

4.4 Interaction

L'interaction se fait par l'intermédiaire du client Web grâce auquel l'utilisateur peut interagir avec le monde 3D ainsi que par l'interface utilisateur 2D associée. Le résultat de ces interactions, la navigation par exemple, est naturellement visualisé au sein du client Web. Mais entre l'interaction que l'utilisateur effectue et le résultat visuel, il est nécessaire de recourir à la hiérarchie des CM contenant les composants de navigation et d'interaction qui, eux seuls, savent quel doit être le résultat de tel ou tel événement d'interaction comme nous l'avons défini dans le chapitre 4.

Une communication doit donc être établie entre le client Web et le serveur, où se trouvent les CM, de manière à pouvoir réaliser les opérations d'interaction. Pour communiquer avec le serveur, le client doit savoir à quel CM correspondent les représentations graphiques existant dans la visualisation. Par exemple, si l'utilisateur clique sur un objet graphique de la visualisation, le client doit pouvoir déterminer à quel CM correspond cet objet afin de lui envoyer l'événement concerné afin que le composant de navigation ou d'interaction détermine le résultat de cette interaction. Les données envoyées vers le client Web afin de construire le monde 3D doivent donc comprendre, outre les représentations graphiques à créer, à quel CM correspond chacune des représentations graphiques.

Le problème survenant dans cette stratégie est que le temps de réponse du système face à un événement d'interaction n'est pas négligeable. En effet, les communications nécessaires entre le client Web et le serveur, où se trouvent les CM, ralentissent la réaction du système. Dans certains types d'interaction, ce temps de réponse non négligeable est inacceptable. Ces types d'interaction sont ceux qui peuvent être souvent sollicités à de très courts intervalles. Par exemple, l'interaction permettant à l'utilisateur d'observer un centre d'intérêt nécessite des calculs en continu afin d'évaluer la nouvelle position de l'utilisateur en fonction des mouvements de la souris (ou de n'importe quel périphérique d'interaction). Par contre, des interactions qui ne sont pas sollicitées à de très courts intervalles, comme le calcul automatique d'un chemin pour aller d'un centre d'intérêt à un autre, peuvent être calculées sans problème sur le serveur.

Certains calculs nécessaires à la réalisation d'interactions doivent donc forcément être réalisés sur le client, sous peine être inexploitables. Un mécanisme, au sein du client, doit donc permettre de réaliser localement certaines tâches définies dans les composants de navigation et d'interaction. Actuellement, l'interaction permettant d'observer en détail un centre d'intérêt ainsi que les tâches de navigation non effectués en mode « chemin » sont réalisées sur le client.

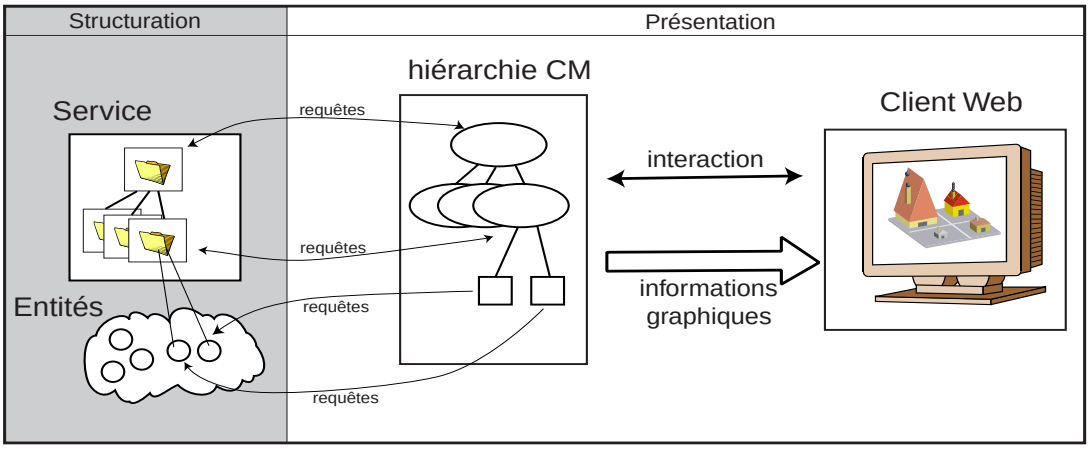


Figure 58 *Couche de visualisation & interaction*

5 Distribution des couches sur le réseau

Nous venons de voir les différentes couches de la plate-forme CyberNet et nous avons vu déjà quelques éléments de la distribution de ces couches sur le réseau :

- Les informateurs s'exécutent au plus près de l'information qu'ils collectent
- La couche de structuration n'est pas distribuée
- La couche de visualisation et d'interaction est distribuée entre un serveur et un client Web

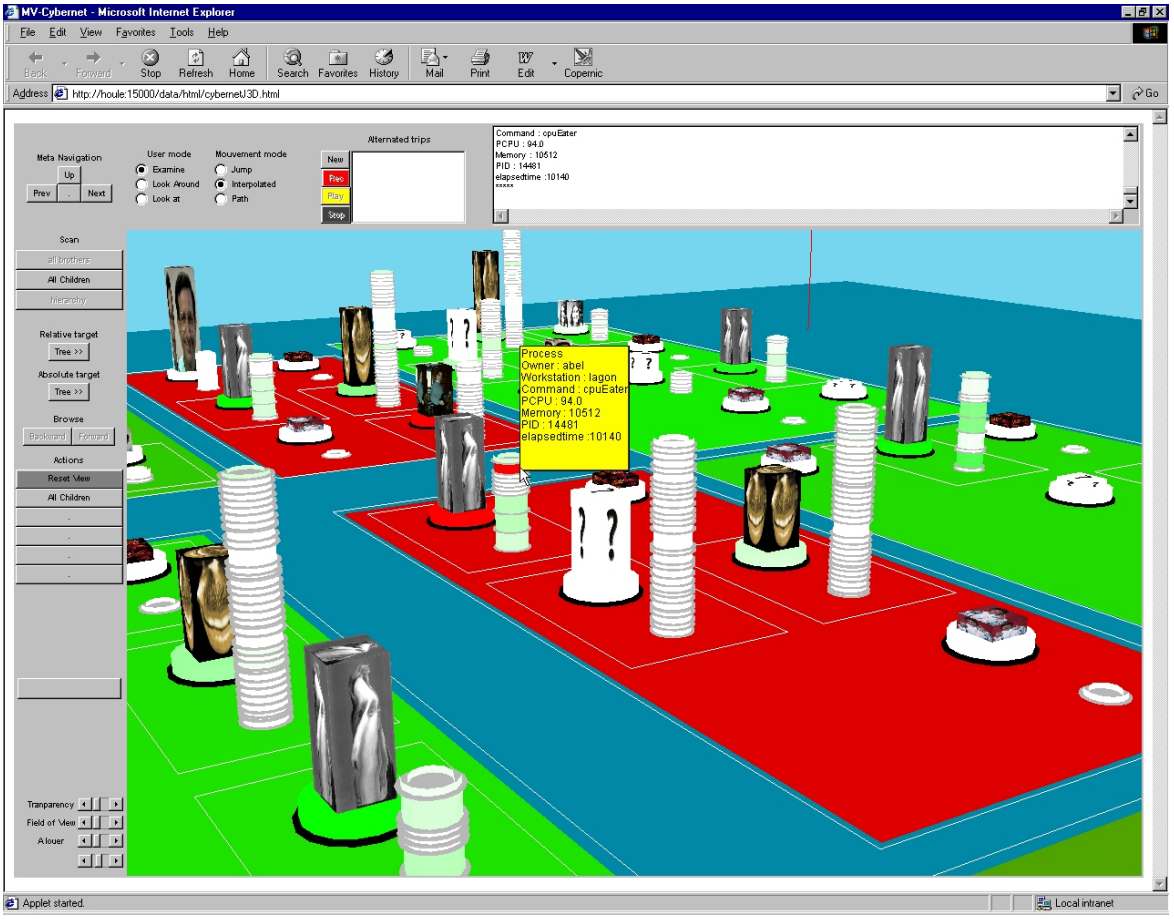


Figure 59 *Interface utilisateur et monde 3D interactif au sein d'un browser Web*

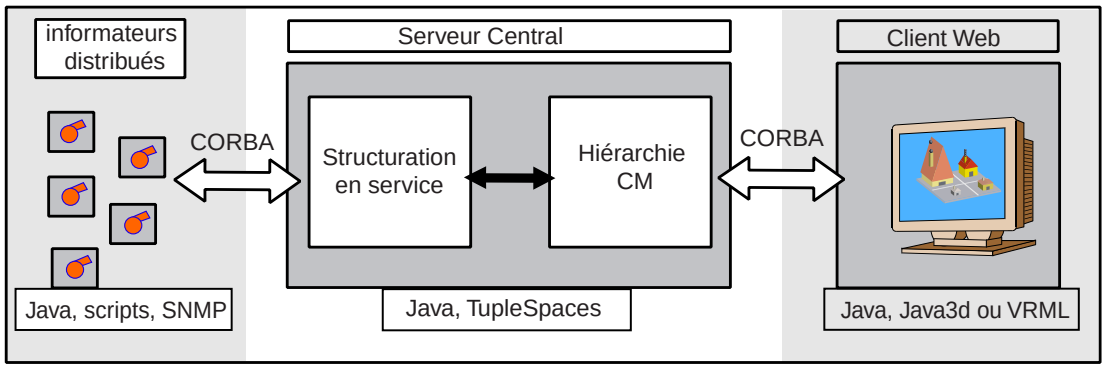


Figure 60 *Distribution des couches sur le réseau et technologies utilisées*

Dans notre implémentation de la plate-forme, le serveur abritant la hiérarchie des CM exécute également la couche de structuration comme illustré à la figure 5. Ainsi, on a une architecture distribuée avec des informateurs s'exécutant lorsque cela est nécessaire pour collecter les informations, et un serveur où s'exécutent la

couche de structuration ainsi que la totalité des opérations d'interaction et de construction du monde 3D, excepté la visualisation en elle-même qui est affichée sur le client Web.

6 Technologies utilisées

6.1 Langage de programmation

Toutes les couches ont été programmées à l'aide du langage Java [Java]. Celui-ci a été choisi pour plusieurs raisons :

- **Orienté objet** : Tout d'abord, java est un langage orienté objet permettant de simplifier la programmation des modèles ainsi que le processus de réutilisation
- **Portabilité** : Le code source Java est compilé en « byte code » qui est ensuite interprété par une machine virtuelle Java (MVJ). Le code compilé peut ainsi être exécuté dans n'importe quel environnement possédant une MVJ.
- **Librairies** : Le langage Java possède des librairies en tout genre permettant de simplifier le travail de programmation. En particulier, il existe une librairie permettant de créer et de synchroniser facilement des threads, qui nous a été particulièrement utile afin de gérer le caractère dynamique des informations.
- **Exécution dans un navigateur Web :** Java est le seul langage supporté en standard par les navigateurs Web.

Les informateurs sont également développés en Java, mais peuvent utiliser des scripts en Perl ou Awk, ou encore le protocole SNMP [Stallings96] afin d'obtenir des informations sur des éléments du réseau.

6.2 Communication entre les objets distribués

Afin de simplifier les communications et les échanges de données entre les différents objets distribués sur le réseau, il est utile d'utiliser une librairie masquant les opérations d'encodage et de recherche des objets distribués sur le réseau. Plusieurs libraires sont disponibles pour Java, les plus importantes étant RMI [RMI] et CORBA [CORBA].

Nous avons choisi CORBA car à l'époque de notre choix, c'était la seule qui fournissait un service de « pages jaunes » nécessaire pour que le serveur principal puisse localiser les informateurs en fonction des entités que ceux-ci peuvent fournir. De plus, à cette époque RMI (1997) n'était pas aussi stable que maintenant. Si le choix était à refaire, nous ne pourrions plus utiliser ces

arguments car RMI est très stable et il est possible de l'utiliser conjointement avec un service de pages jaunes.

6.3 Référentiel

Le référentiel est utilisé dans la couche de structuration afin de référencer toutes les entités existantes dans le système. Les références des entités doivent pouvoir être obtenues à partir d'une requête contenant des critères relatifs au type et/ou aux champs des entités. De plus, à partir d'une requête semblable, il doit être possible de s'abonner au référentiel afin de recevoir les références des futures entités qui satisferont cette requête.

Le référentiel peut donc être considéré comme une base de données à laquelle on peut s'abonner. Nous avons choisi d'utiliser le framework Tspaces [Wyckoff98] qui est une extension du concept de TupleSpaces[Gelernter82]. Les Tuplespaces fournissent des mécanismes de communication, de synchronisation et de transaction permettant de partager efficacement des tuples. Tspaces ajoute de nombreuses fonctionnalités, dont celle permettant de s'abonner afin de recevoir les futurs tuples. Une entité est donc enregistrée dans le TupleSpaces sous la forme d'un tuple dont les champs sont les informations, le type ainsi que la référence de l'entité associée. Un autre TupleSpaces avancé, appelé JavaSpaces [Waldo98],. est également disponible. JavaSpaces propose des fonctions similaires à Tspaces mais se base entièrement sur RMI, ce qui posait quelques problèmes avec l'utilisation de CORBA.

6.4 Librairie 3D

Em matière de librairies 3D, le choix était assez restreint car un des buts de cette plate-forme était de permettre à l'utilisateur de visualiser le monde 3D à l'intérieur d'un navigateur Web.

En 1997, une seule solution était possible : VRML [VRML]. Grâce à l'utilisation d'un plug-in VRML, on peut visualiser au sein d'un navigateur Web un monde 3D définit au format VRML dans un fichier texte. Encore plus intéressant pour nous, VRML définit également l'EAI [Marrin96] une interface permettant, à partir d'un programme java s'exécutant dans le navigateur, de gérer dynamiquement le monde 3D. Les premières versions de notre plate-forme utilisaient l'EAI afin de créer dynamiquement les mondes 3D, mais malheureusement cette interface n'était pas très efficace pour gérer beaucoup d'objets dynamiques et les mises à jour des mondes 3D, ainsi que pour définir de nouveaux modes d'interaction.

A partir de 1999, nous avons commencé à nous intéresser à Java3d [Java3d], une librairie permettant de créer des mondes 3D en Java, et de plus capable s'exécuter dans un navigateur Web. Cette librairie, de plus bas niveau que VRML, s'est montrée bien plus efficace dans la gestion de la dynamique de nos mondes 3D. De plus, elle nous a apporté plus de souplesse dans la création de nos mondes

,en particulier pour le côté interactif . Nous avons donc migré vers Java3d à partir du début de l'année 2000.

7 Conclusion

La plate-forme que nous avons présentée dans ce chapitre s'est révélée adaptée à la visualisation d'informations dynamiques et distribuées. Elle a permis de valider les modèles qui ont été décrit dans les chapitres précédents en les testant dans différentes situations comme nous pourrons le voir dans le chapitre suivant. Son approche orientée objet a permis de développer un framework permettant de réaliser rapidement des tests, tant au niveau de la définition de services qu'au niveau de nouveaux types de visualisation.

Les problèmes rencontrés dans le développement de cette plate-forme sont dus majoritairement au caractère dynamique des informations traitées. Ce caractère a entraîné de nombreuses stratégies qui n'auraient pas été nécessaires dans le cas d'informations statiques, comme l'utilisation d'agents distribués (les informateurs) ou les mécanismes d'abonnement aux entités et au référentiel.

D'autres problèmes ont été également rencontrés. Le partage des entités par plusieurs services a nécessité l'utilisation d'une base de données (concrétisé par le tupleSpace). L'utilisation d'un navigateur Web comme client a restreint les possibilités dans le choix des libraires graphiques 3D et a entraîné des contraintes concernant la stratégie de mise à jour et la gestion de l'interactivité des mondes 3D.

7 Expérimentations

1 Introduction

Dans ce chapitre nous présentons différentes expérimentations réalisées dans le cadre du projet CyberNet. La majorité de ces visualisations ont été réalisées de 1997 à 2000 par des étudiants de l'Institut Eurecom lors de leur projet de dernière année d'ingénieur. Le but des étudiants était d'utiliser la plate-forme logicielle CyberNet pour définir de nouveaux services et créer de nouveaux types de visualisation. Ces expérimentations nous ont été précieuses car elles ont permis de dégager les problèmes et les limitations existants dans nos modèles, ainsi que dans les versions successives de la plate-forme. Nous remercions donc particulièrement ces étudiants.

Il est à noter que certaines de ces expérimentions sont également exposées sur les pages Web du projet CyberNet (www.eurecom.fr/~abel/cybernet/), où on pourra trouver des mondes VRML et des vidéos complétant les captures d'écran présentées dans ce chapitre. Certaines de ces expériences ont été publiées notamment dans [Abel00][Abel00b][Gros00].

Les expérimentations sont présentées en fonction du type de service qu'elles permettent de visualiser, certaines ont déjà été exposées à titre d'exemple dans les chapitres précédents. Nous présentons des services d'administration de réseau, de stations de travail, d'applications distribuées, d'analyse de trafic réseau et enfin d'analyse de serveur Web. Avant cela, nous consacrons une partie à certains travaux réalisés par la communauté scientifique dans le domaine de la visualisation 3D d'informations liées aux réseaux.

2 Travaux similaires

A notre connaissance, les premières utilisations de la technologie 3D pour l'analyse de réseaux remonte à moins de dix ans. Ainsi, une des premières expériences a commencé en 1992 au sein du groupe COMET de l'université Columbia où la 3D a été utilisée afin de visualiser la topologie d'un réseau ATM à

l'aide d'un graphe représenté par des sphères et des cylindres [Feiner93][Crutcher95]. Leur système permettait également de superviser un élément particulier du réseau ou d'analyser un chemin virtuel à l'intérieur d'un réseau ATM.

Les Bells Labs ont également étudié les apports de la 3D pour la supervision de réseaux. Partant de leurs expérimentations en 2D [Becker95], ils ont développé plusieurs outils permettant d'analyser le trafic de réseaux, comme Internet [Cox95][Cox96][He98]. Certains de ces outils permettent de réaliser l'analyse géographiquement à l'aide d'un modèle 3D de la Terre ou des Etats-Unis, et leurs derniers travaux ajoutent de nombreuses possibilités au niveau interactif (e.g. regroupement ou repositionnement d'éléments)

Le projet VENoM [VENoM] propose de visualiser en temps réel des réseaux ATM à l'aide d'un monde 3D multi-utilisateurs. Plusieurs visualisations sont disponibles : reconstruction 3D de salles des machines, visualisation d'un élément particulier du réseau, topologie à l'aide d'arbre de cônes, analyse des ports d'un switch ATM. Un autre projet [Kahani96] s'est également intéressé aux mondes multi-utilisateurs.

Plus récemment, [Eleftherios99] présente un outil, Swift-3D, permettant la visualisation de larges volumes d'informations dynamiques provenant de réseaux de télécommunications. Les visualisations sont principalement à caractère géographique, et les résultats semblent assez prometteurs car les auteurs explique que cet outil a permis d'améliorer la gestion des réseaux ainsi que les offres commerciales de services réseau chez AT&T. Un autre outil plus générique, Virtue [Schaffer99], présente un environnement collaboratif permettant d'analyser les performances d'applications distribuées grâce à toutes sortes de métaphores visuelles. Flodar [Swing98], présente de nouvelles métaphores destinées à visualiser le flux d'informations sur un réseau. [Cichlid] est un programme libre permettant de visualiser en 3D la topologie et l'activité d'un réseau

Des outils à vocation commerciale existent également. Parmi eux, on peut noter Performance Co-Pilot de SGI [Performance], qui permet d'analyser les performances d'éléments ou d'applications critiques sur un réseau à l'aide de visualisations 3D bien connues (e.g. graphes à barres 3D). Nscope de Fourthplanet [Fourthplanet] permet de superviser un réseau à l'aide de mondes 3D représentant la topologie du réseau ainsi que les caractéristiques des éléments le composant. Unicenter TNG de Computer Associates [Unicenter] comporte également une interface graphique 3D avancée permettant de visualiser, en particulier géographiquement, les éléments à superviser.

Concernant la gestion d'informations liées au WWW, [Mukherjea95c] a présenté la visualisation de hiérarchies extraites du Web à l'aide d'arbres de cônes. [Andrews95] utilise de son côté des paysages d'informations pour visualiser des sites Hyper-G (une extension du WWW). [Frecon98] présente WebPath, un outil original permettant à un utilisateur du WWW de visualiser son parcours sur la toile sous la forme d'un graphe 3D où les nœuds représentent les pages visitées.

De nombreuses autres expériences permettant la visualisation d'informations liées aux réseaux ont été menées. Je recommande au lecteur intéressé par celles-ci

de consulter l'excellent site [CyberAtlas] qui présente, illustrations à l'appui, un nombre important de ces expériences.

3 Administration de réseaux

3.1 Administration géographique

Nous avons développé un outil de supervision géographique de réseau LAN dans le but d'étudier les apports du 3D dans le cadre de la gestion du réseau à l'intérieur d'un bâtiment. Le système peut aisément être étendu à la notion de WAN en multipliant les bâtiments. L'idée de départ est de développer une métaphore de bâtiment, composée d'étages, de couloirs, d'escaliers et de bureaux et de pouvoir placer des informations à l'intérieur de cette métaphore.

L'application principale de cette étude est de représenter un bâtiment (de façon symbolique) en respectant les localisations géographiques des bureaux, salles machines, locaux techniques, couloirs d'accès etc. Les différents éléments constituants le système informatique (ordinateurs, hubs, switchs, routeurs) et les utilisateurs sont ensuite placés à l'intérieur de ce bâtiment en conservant leur localisation réelle (ou approximative) ainsi que leur dépendance logique (quelle machine est connectée sur quel switch, quel switch est connecté sur quel routeur, etc.). Cette utilisation n'est pas métaphorique, puisque les informations représentées sont directement issues des informations réelles. Le modèle est enrichi par l'ajout d'un certain nombre d'informations structurelles de l'entreprise, comme le découpage du bâtiment en termes de départements (chaque local, chaque utilisateur et chaque ordinateur est attribué à un département). L'objectif est d'aider l'opérateur pour les opérations:

- de gestion de parc (informatique et réseau)
- de télémaintenance.

Le système peut être exploité de façon détournée en utilisant la métaphore de bâtiment afin de regrouper les informations sur des critères autres que leur proximité géographique. A titre d'exemple, il est possible de regrouper tous les bureaux d'un même service sur un même étage d'un bâtiment virtuel sans tenir compte de leur localisation réelle. Ces bureaux peuvent en fait être localisés dans des villes voire des pays différents. L'administrateur peut alors avoir une vue claire des ressources utilisées par les membres de ce département et de leur activité indépendamment de leur localisation géographique. Il peut identifier rapidement les éléments du réseau dont dépend le département. La métaphore de bâtiment est alors utilisée comme outil de regroupement (classification) visuelle de ressources possédant des caractéristiques communes.

Afin de tirer parti de cette représentation 3D, nous avons travaillé sur les apports de la réalité virtuelle. Il est rapidement devenu clair que l'opérateur devait pouvoir aisément exploiter la "virtualité" du bâtiment et qu'il est possible, dans le

monde virtuel, d'effectuer des actions impossibles dans le monde réel. Le deux principales opérations qui nous sont apparues intéressantes à étudier sont:

- La possibilité de contrôler l'apparence visuelle des objets (couleur et transparence) en fonction de critères. Pour cela l'utilisateur dispose d'une interface lui permettant d'interagir avec le système de façon à pouvoir modifier l'apparence du modèle. Il peut par exemple modifier la couleur ou la transparence des bureaux en fonction de leur appartenance à un département, il peut également ne visualiser que les ressources informatiques appartenant à ce département etc.
- La possibilité de naviguer de façon "intelligente" dans le bâtiment. La métaphore de bâtiment a été pour nous un terrain d'expérimentation pour le mode de navigation de type « chemin » que nous avons présenté au chapitre 4. L'outil intègre donc un système de déplacement contraint permettant à l'utilisateur de se déplacer de façon automatisée dans le bâtiment.

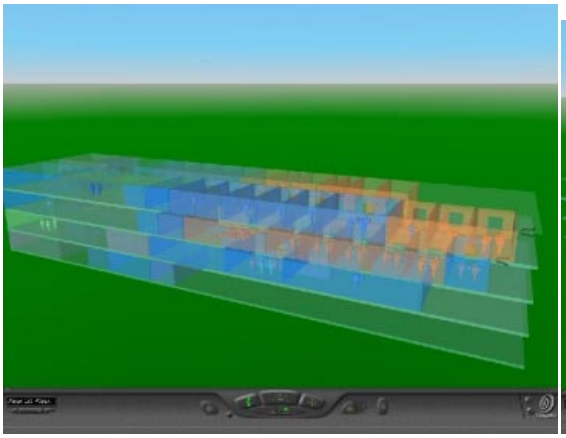


Figure 61 Exemple de contrôle de la transparence et de la couleur des cloisons des bureaux en fonction de leur appartenance à un département**.**

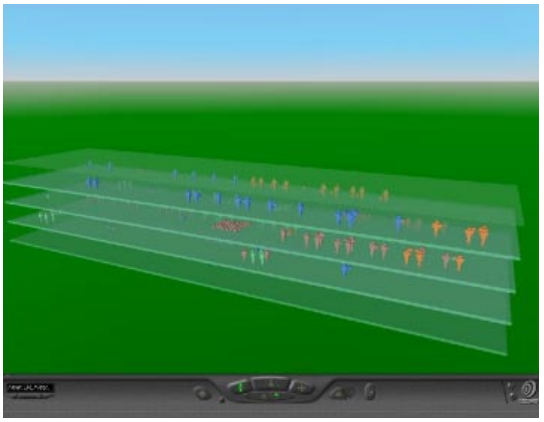


Figure 62 Le même bâtiment sans cloison avec positionnement des utilisateurs.

Figure 63 Vue symbolique de l'intérieur d'un bureau. L'opérateur peut interagir avec les éléments (ordinateurs et utilisateurs) de façon à obtenir des informations supplémentaire (il est possible par exemple de connaître les éléments actifs réseau dont dépend un ordinateur en interagissant avec la boite noire situé au-dessus de chaque ordinateur)

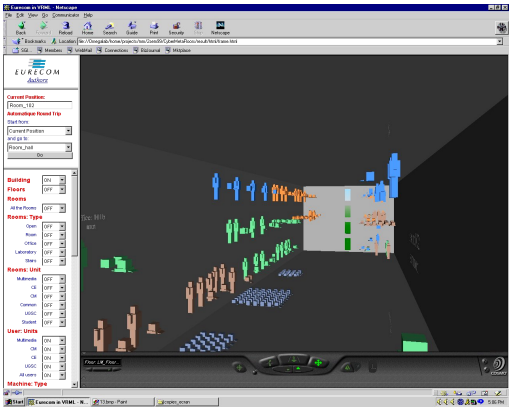


Figure 64 Bâtiment vu de l'intérieur sans cloisons ni sols, mais avec la représentation des ressources informatiques et des utilisateurs.

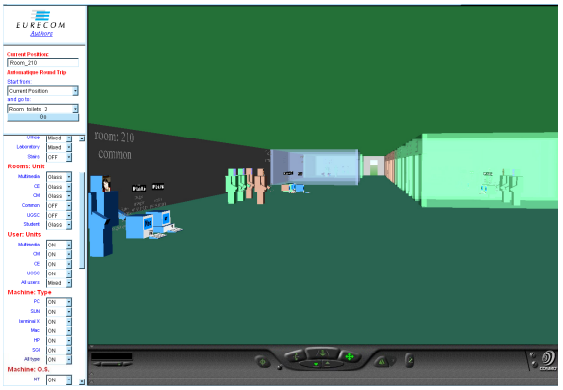


Figure 65 Le même bâtiment avec cloisons et sols.

Ce type de système de supervision de réseau peut être particulièrement intéressant pour identifier des éléments à partir de leur localisation géographique ou de tout autre critère de regroupement de ressources, comme l'appartenance à un département de l'entreprise. Lorsqu'un problème survient sur le réseau, l'opérateur peut identifier les relations entre éléments défectueux et l'outil permet de rapidement évaluer si le problème est de nature géographique (lié au câblage par exemple). L'outil offre des potentialités (que nous n'avons pas exploitées) quant à la gestion de parc, il est en particulier facile d'identifier des erreurs d'attribution de ressources ou de connexion. Il peut s'avérer également utile dans le cadre d'opérations de télémaintenance afin de guider une opération à distance. Le développement et l'expérimentation de cet outil nous a permis de valider nos idées en ce qui concerne l'interaction et la navigation.

3.2 Administration topologique

Les expériences effectuées avec l'outil précédent ont montré que ce type de visualisation s'avérait inadaptée pour représenter efficacement les informations de topologie du réseau et en particulier les dépendances entre éléments. L'appréhension de la topologie du réseau est un point capital pour ce qui concerne les outils de gestion de réseau. Un grand nombre de problèmes peuvent être aisément détectés à l'aide d'une représentation de la topologie qui mette en évidence les connexions et leur activité (bande passante, taux d'erreur) entre les différents éléments du réseau. Nous nous sommes intéressés à ce problème dans le cadre de l'administration d'un réseau local (LAN). Nous avons pour cela développé un outil se fondant sur le modèle de représentation de l'arbre de cônes. L'arbre de cônes permet de représenter de façon claire des hiérarchies. Bien que la structure physique d'un LAN ne soit pas nécessairement hiérarchique, sa structure logique l'est en général par application d'un algorithme de spanning tree. L'arbre de cônes n'est pas à proprement parler une métaphore du monde réel. Il nous est

cependant apparu particulièrement adapté à la représentation logique de la topologie d'un réseau local car :

- il peut supporter la visualisation de hiérarchies ayant un nombre illimité de niveaux (ce qui est le cas des arbres de spanning tree). C'est également le cas des métaphores de pyramides (utilisées plus loin pour représenter les hiérarchies de fichiers).

- il permet de représenter les liens entre les éléments de l'arbre. Ceci est particulièrement intéressant dans le cadre de la visualisation de la topologie d'un réseau, pour pouvoir représenter l'activité des connexions (ce qui n'est pas le cas des pyramides).

Le modèle de représentation adopté consiste à représenter le LAN sous la forme d'un (ou plusieurs) arbre, dont les nœuds feuilles sont les ordinateurs et les nœuds intermédiaires les éléments actifs du réseau (routeurs, concentrateurs et commutateurs). Le domaine de collision d'un concentrateur est représenté par un cone à la circonférence duquel apparaissent les ports et les éléments connectés à chaque port. C'est ce cône qui sera utilisé pour représenter l'activité présente sur ce domaine de collision. Pour les commutateurs et les routeurs, chaque port est représenté par une branche indépendante qui pourra représenter l'activité de ce port indépendamment des autres ports. Dans le système que nous avons développé, les informations sont collectées périodiquement (la période est de l'ordre de la minute) sur tous les éléments actifs du réseau en utilisant SNMP. Les valeurs de la bande passante et du taux d'erreur sont directement visualisées en modifiant la taille et la couleur des liens représentant les connexions.

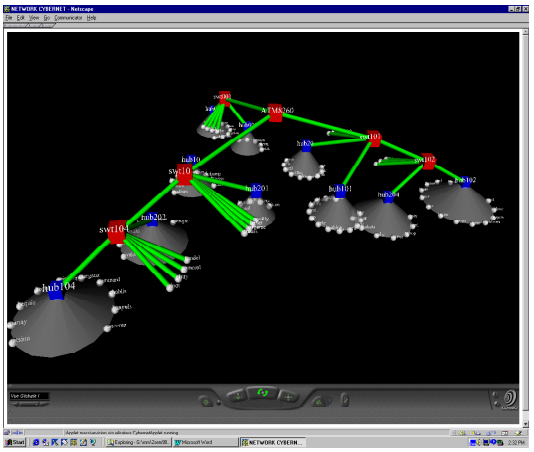


Figure 66 Arbre de cônes représentant la topologie d'un réseau (ici sans informations sur l'état des connexions)

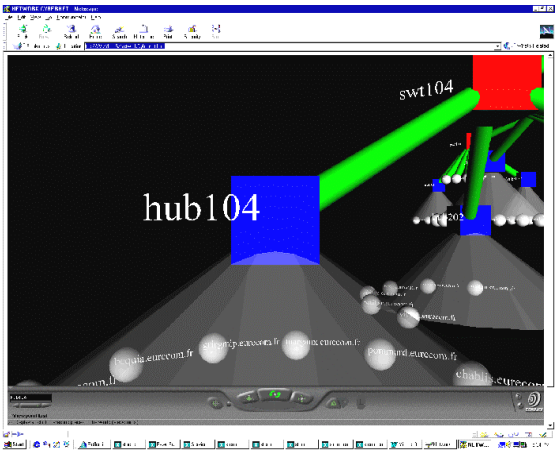


Figure 67 vue d'un concentrateur à l'intérieur de l'arbre de cônes.

L'utilisation de cet outil d'administration topologique s'avère particulièrement intéressante dans le cadre de la détection de problèmes de congestion ou pour la détection de dysfonctionnement de certains éléments. Elle est particulièrement intéressante pour avoir une vision synthétique de la topologie du réseau local, permettant, par exemple, de détecter rapidement les problèmes structurels issus d'une mauvaise configuration du spanning tree. Un autre intérêt important de cet outil est la possibilité de comprendre aisément les problèmes de trafic (détection et suivie de la montée en charge d'un lien ou d'un ensemble de liens). Il est bien sûr envisageable de représenter d'autres informations que la bande passante et le taux d'erreurs. On peut par exemple penser à représenter des informations plus structurelles comme les classes d'adresses transportées ou certains paramètres de configuration par exemple liés à la sécurisation du réseau. Nous n'avons pas exploré ces pistes pour le moment.

3.3 Conclusion

Pour l'administration purement réseau (en terme de géographie et de topologie), la 3D semble offrir la possibilité de présenter les informations d'une façon particulièrement synthétique qu'elles soient dynamiques ou non. Elle facilite la compréhension de la structure du réseau et peut accélérer la détection et l'analyse des problème. Il nous est également apparu que l'utilisation conjointe de ces deux outils en permettant, par exemple, de sélectionner un élément dans l'outil de gestion de la topologie et en observant la localisation de toutes les machines qui y sont connectées dans l'outil de gestion géographique, pourrait décupler la puissance du système. Dans l'état actuel des choses, le système est conçu de façon à pouvoir supporter l'interaction entre plusieurs outils mais pour des raisons de temps, nous n'avons pas encore effectué d'expérimentations sur ce sujet.

4 Administration de stations de travail

4.1 Administration des processus

Les outils présentés précédemment sont des extensions 3D d'interfaces usuelles de systèmes de gestion de réseau. La représentation topologique ou géographique est indispensable à tout système de gestion de réseau et la réalité virtuelle peut améliorer l'accès à l'information. Toutefois, il s'agit de modèles de représentation permettant une analyse « bas-niveau » du réseau. Le but de CyberNet est d'aller plus loin que la simple gestion du réseau au sens d'un ensemble d'équipements permettant de connecter des appareils. Notre objectif est de fournir à un opérateur un moyen d'administrer des services qui s'exécutent sur une partie ou l'ensemble des éléments du réseau. Les outils précédents peuvent

être utilisés pour identifier un problème au niveau du réseau, mais d'autres outils doivent être développés pour pouvoir analyser des problèmes d'autre nature.

Dans ce cadre, nous avons étudié un outil d'administration de l'activité des machines sur un réseau. L'idée de départ à été de développé un outil dérivé du "top" de Unix ou du "task manager" de NT. Contrairement à ces outils, qui se limitent à la visualisation des caractéristiques d'une machine, nous avons développé un système complet adapté à l'administration de processus répartis sur un ensemble d'ordinateurs et assurant l'affichage d'informations sur les processus s'exécutant sur les machines à administrer. Les informations visualisées sont classiques : le pid, le nom de l'exécutable, son âge, l'occupation CPU, l'occupation mémoire, la station sur laquelle il s'exécute et son propriétaire. A partir de ces informations, il nous est apparu que l'outil d'administration des processus devait pouvoir être capable de regrouper visuellement les informations concernant les processus selon d'autres critères que celui de la machine sur lequel le processus est exécuté. A titre d'exemple, nous avons implanté les représentations suivantes:

- les processus sont classés par machines et pour chaque machine par utilisateur (topbywks).
- les processus par utilisateur et pour chaque utilisateur par machine (topbyuser).
- les processus sont classés en fonction de l'exécutable et pour chaque exécutable en fonction de l'utilisateur (topbyexe)

Nous avons expérimenté plusieurs types de métaphore pour représenter les informations des processus. Les plus intéressantes sont la métaphore de ville et la métaphore de système solaire.

Dans la métaphore de ville appliquée au système topbywks, les données sont structurées de la manière suivante : la ville représente le réseau à observer. Elle est séparée en quartiers, qui symbolisent les sous réseaux. Chaque quartier contient des immeubles qui symbolisent les stations, le socle de chaque immeuble donne le nom de la station et sa charge CPU globale, ce qui constitue des informations propres à la station. Ces immeubles sont formés d'étages symbolisant les utilisateurs, et une fenêtre d'un étage représente un processus appartenant à l'utilisateur représenté par l'étage, la couleur de la fenêtre étant proportionnelle à l'activité du processus.

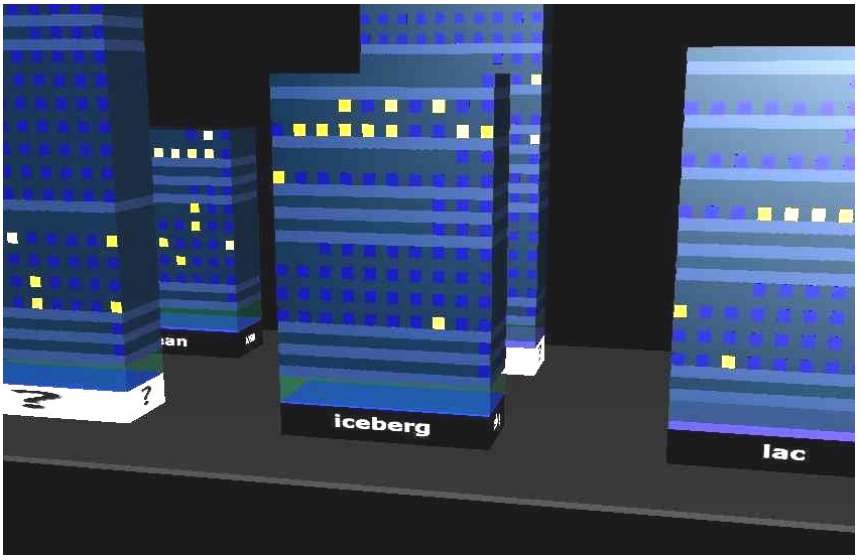


Figure 68 Une vue de l'activité d'une station (représentée par un immeuble de la ville)

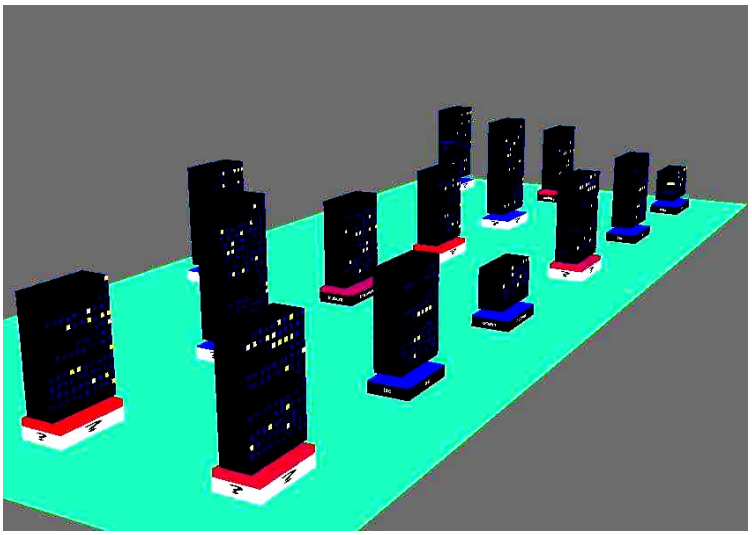


Figure 69 topbywks représenté à l'aide de la métaphore de ville**.**

Dans la métaphore de système solaire, les stations sont des planètes autour desquelles gravitent des satellites représentant les utilisateurs. Entre la planète et chaque satellite flotte un ensemble d'objets représentant les processus, leur diamètre et leur longueur indiquant la mémoire utilisée et le CPU de chaque processus. Des informations résumant l'activité globale d'un utilisateur sur une station sont également affichées, la taille et l'intensité de la couleur des satellites représentant la consommation de mémoire et de CPU.

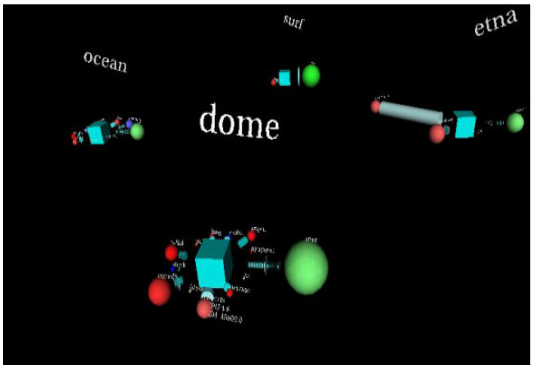


Figure 70 Topbywks représenté à l'aide de la métaphore de système solaire

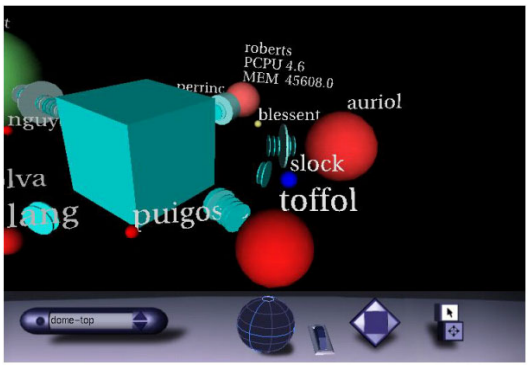


Figure 71 Vue rapprochée d'une station de travail

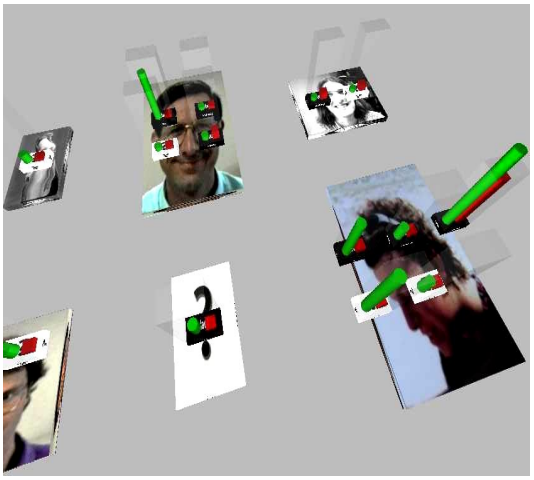


Figure 72 Topbyuser. Chaque plate-forme réunit les activités (en terme de mémoire et CPU) d'un utilisateur sur les ordinateurs auquel il est connecté

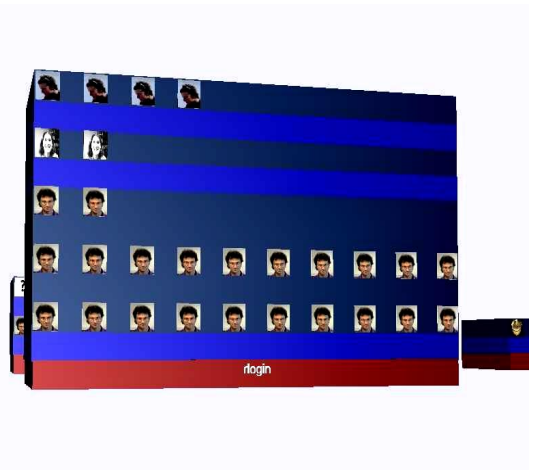


Figure 73 Topbyexe représenté à l'aide d'une métaphore de ville (ici l'immeuble des processus rlogin existants sur le réseau et rangés par utilisateur)

4.2 Administration de systèmes de fichiers

La gestion du système d'information d'une entreprise passe par une bonne gestion des données stockées sur les serveurs. La tache d'administration de ces données consiste à s'assurer de la sécurité des accès, à valider la structure et à détecter les problèmes potentiels. La visualisation de systèmes de fichiers, spécialement pour les serveurs de données, pose de nombreux problèmes de concision et d'exhaustivité. En effet, comment donner des informations concernant tout un système de fichiers d'une manière qui permette la

compréhension immédiate de ce système (Où sont localisés les fichiers d'images ? Qui a accès à ce répertoire ? Où je puis-je gagner de la place en compressant ?). De nombreuses solutions pour représenter ce type de hiérarchies ont été proposées, et celle qui a le plus retenu notre attention est la représentation pyramidale [Andrews97]. Le but de cet outil est de pouvoir remonter les informations pertinentes suivant les utilisateurs, et suivant les utilisations. Nous avons donc déterminé un certain nombre de scénarios où l'utilisation de cet outil, appelé FS3D, apportera des avantages en termes d'ergonomie, comparé à l'utilisation d'outils classiques, comme le parcours des systèmes de fichiers à l'aide de shells ou même d'explorateur, comme Xfm ou l'explorateur Windows.

Dans les scénarios que nous avons étudiés, nous nous sommes consacrés sur l'étude des répertoires en omettant les fichiers afin de réduire le nombre d'élément à visualiser et de créer une visualisation plus claire. Les scénarios ont été établis à partir des questions suivantes :

- Quelles sont les types de fichiers (images, texte, exécutables,…) présents sur chaque répertoire ?
- Qui sont les propriétaires des répertoires ?
- Quel est l'âge des répertoires ?
- Quels sont les niveaux de sécurité des répertoires ?

La métaphore, inspirée de [Andrews97], représente des boîtes empilées les unes sur les autres donnant naissance à des sortes de pyramides. Chaque boîte représente un répertoire et la hiérarchie du disque peut donc être étudiée en analysant visuellement les pyramides. La hauteur de chaque boîte représente la somme de tous les fichiers existants dans le répertoire correspondant. La couleur est utilisée de différentes manières en fonction du scénario choisi, comme on peut le voir dans les figures ci-dessous.

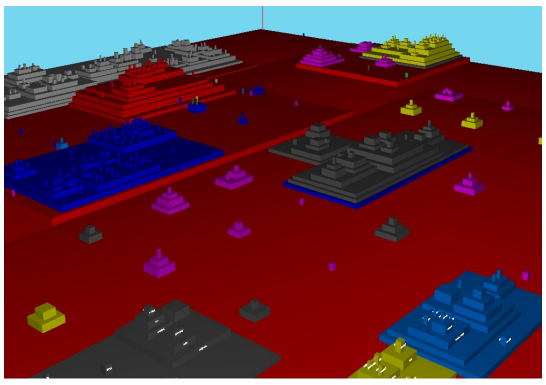


Figure 74 Visualisation d'une hiérarchie de répertoire à l'aide d'une métaphore de pyramide. La couleur de chaque répertoire correspond à son propriétaire

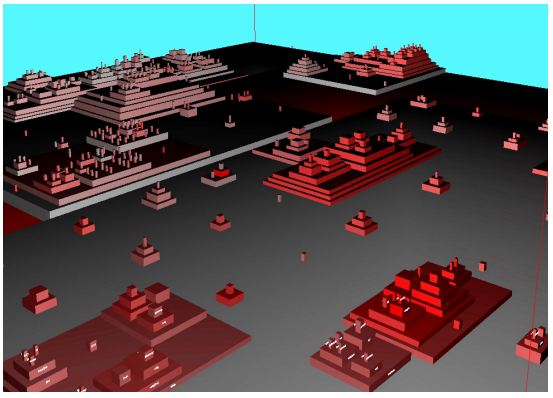


Figure 75 Même vue que la figure précédente, mais avec la couleur codant l'âge des répertoires , du gris (ancien) au rouge (récent)

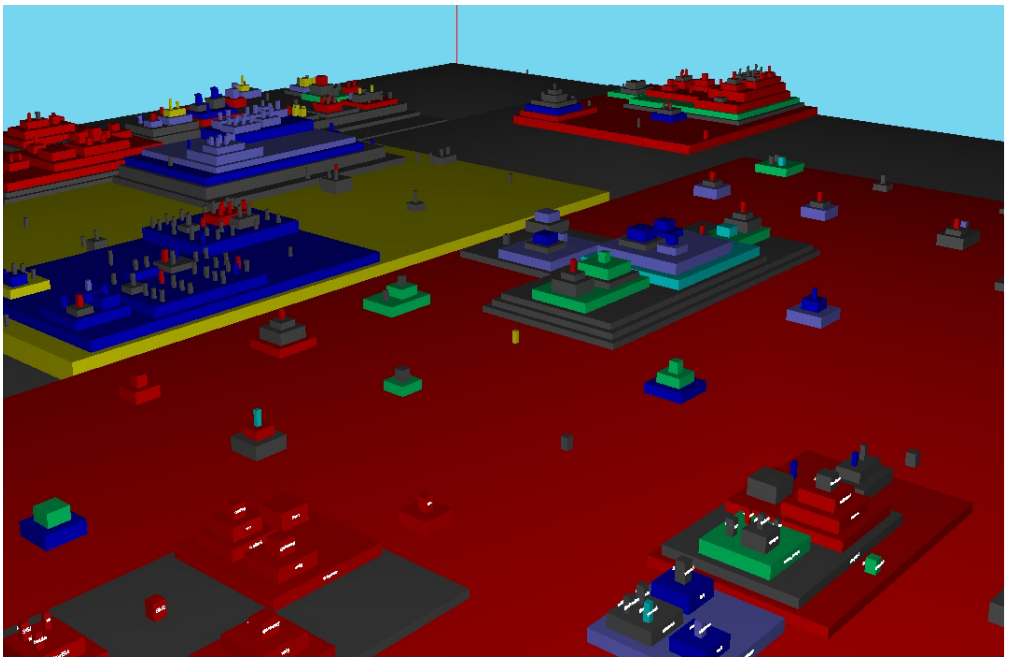


Figure 76 Même vue que les précédentes figures, mais avec la couleur codant le type de fichier prépondérant dans le répertoire**.**

Ce type de visualisation nous semble réellement efficace dans l'analyse d'un système de fichiers complexes, et nous nous sommes surpris à redécouvrir nos disques dur en les visualisant avec cet outil.

5 Administration de services

A côté des exemples traditionnels de visualisation que nous venons de voir, les administrateurs de réseaux ont également besoin de superviser des services plus abstraits et distribués sur plusieurs éléments du réseau. Les applications client/serveur, comme le courrier électronique ou les bases de données, en sont des exemples typiques. Pour notre expérimenter ce type de services, nous avons choisi NFS.

5.1 Supervision du service NFS

Le service NFS (système de fichiers distribués) est assez complexe et il est difficile de se faire une idée de son activité de celui-ci grâce aux outils actuels. Typiquement, tous les ordinateurs d'un réseau ont des liens sur plusieurs disques partagés grâce à NFS et les serveurs ont, en plus, des liens vers leurs clients. Le

résultat est qu'il existe de très nombreuses relations client-serveur. En utilisant des métaphores, notre but était de ne pas tomber dans le piège de la visualisation de graphes ressemblant à un plat de spaghetti. Nous avons expérimenter deux types de métaphores : les villes et l'arbre de cônes.

Dans la métaphore de ville, nous avons associé chaque ville à un sous réseau, chaque quartier à un ordinateur, chaque maison à un disque local ou NFS (i.e. montée à distance), et chaque immeuble à un disque local mais partagé via NFS. Chaque étage d'un immeuble représente un client NFS et le nombre de fenêtres à cet étage correspond au nombre de fichiers ouverts (filehandle) sur le serveur. La couleur du sol d'un quartier permet de distinguer si l'ordinateur associé est un serveur ou seulement un client. D'autres informations (par exemple les erreurs de transfert qui sont projetées sur des arbres) sont également affichées. Cette visualisation s'est avérée performante pour obtenir une vision globale du service NFS (charge et localisation des serveur, nombre de client sur un ordinateur, etc.).

Nous avons également testé la métaphore d'arbre de cônes pour visualiser les mêmes informations relatives au service NFS. Cette métpahore a pour avantage de présenter plus clairement le caractère hiérarchique des informations, mais elle pêche par manque de richesse par rapport à la métaphore de ville. La visualisation résultante est donc moins concise car il n'existe pas d'éléments visuels complexes, comme un immeuble contenant des étages qui contiennent des fenêtres

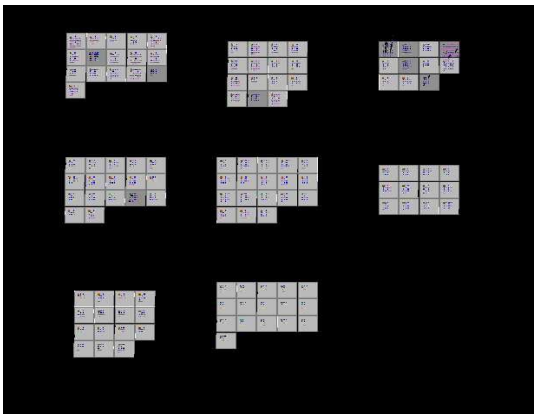


Figure 77 Vue globale du service NFS représentée par une métaphore de villes. Chaque ville représente un sous réseau et chaque quartier un ordinateur. La couleur du sol de chaque quartier permet de déterminer si l'ordinateur correspondant est un serveur ou un simple client.

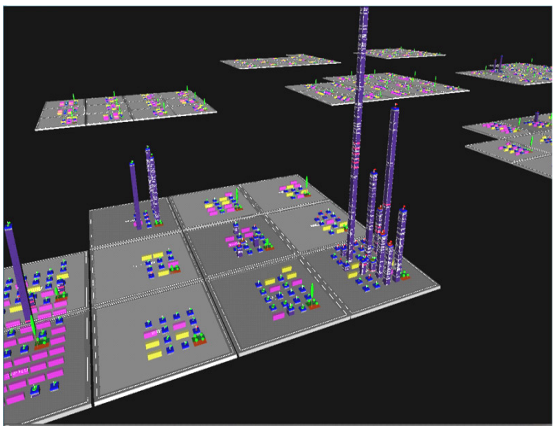


Figure 78 Vue d'une ville représentant un sous réseau. On peut constater qu'un quartier comporte de nombreux serveur (les gratte-ciel).

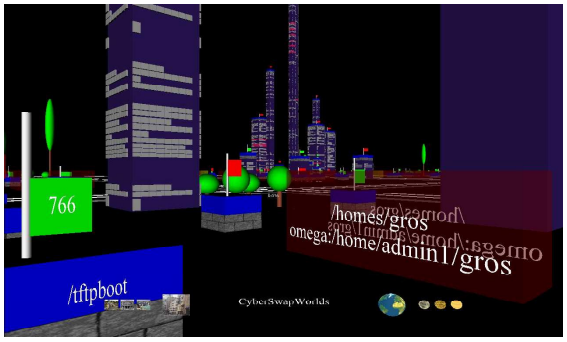


Figure 79 Vue à l'intérieur de la ville. Au premier plan on peut voir un monticule (surmonté d'un drapeau) qui symbolise une maison représentant un disque local de l'ordinateur associé à ce quartier.

Figure 80 Vue à l'intérieur de la ville, les boîtes roses et jaunes symbolisent des maisons qui représentent des disques montés à distance via NFS

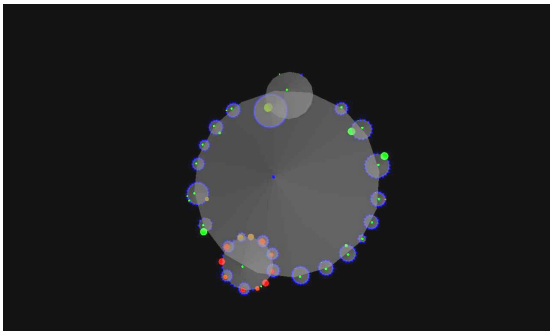


Figure 81 Vue du service NFS à l'aide d'un arbre de cônes. Chaque cône du premier niveau représente un ordinateur, ses fils repésentant les disques locaux et NFS de l'ordinateur.

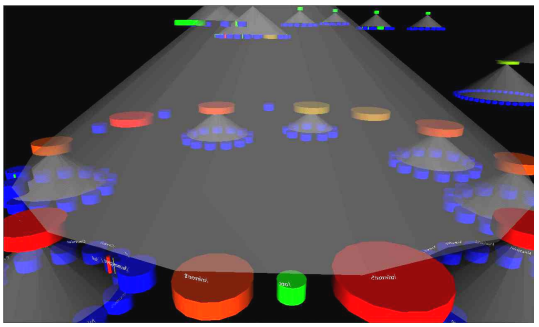


Figure 82 Vue rapproché d'une partie du cône visualisant les disques locaux et NFS (cylindres orange, rouge et vert) d'un ordinateur, ainsi que les clients des disques partagés (cylindres bleus).

6 Analyse de trafic réseau

Le but de cette expérimentation était de créer, à l'aide de la plate-forme CyberNet, des outils permettant d'analyser les grands volumes d'informations capturées par un analyseur de réseau. Cette analyse est cruciale pour les personnes ayant chargées de créer des modèles permettant de simuler les réseaux et de caractériser les types de connexions. Les informations collectées se composent des caractéristiques temporelles des connexions passant par l'analyseur, comme par exemple la taille de la fenêtre de congestion TCP ou le débit à un instant t, ainsi que la source et la destination de chaque connexion.

Pour effectuer cette analyse, nous avons mis au point plusieurs outils pouvant s'utiliser en parallèle. Premièrement, nous avons voulu créer un outil permettant d'analyser toutes les connections d'une manière synthétique, afin de pouvoir détecter les connections intéressantes et les étudier plus en détail par la suite. Il faut dire que le volume des informations récoltées par les analyseurs de réseaux est souvent gigantesque, mais qu'un nombre limité de ces informations sont intéressantes. Il est donc nécessaire de proposer un outil séparant le bon grain de l'ivraie pour ensuite l'analyser en détail.

Nous avons finalement défini deux outils permettant d'analyser l'ensemble des connexions et ayant chacun leur avantage. Le premier permet d'analyser l'ensemble des données selon des critères de source et de destination, ceci permettant d'avoir une vision globale de la répartition du trafic entre les différentes adresses IP. Le deuxième outil permet d'analyser le comportement des connections dans le temps, et ainsi de détecter quels sont les connexions les plus intéressantes de ce point de vue.

Une fois une connexion intéressante déterminée, il est nécessaire de l'analyser en profondeur. Pour cela, nous avons développé un troisième outil permettant de visualiser les différentes caractéristiques de cette connexion d'un point de vue temporel. Mais après avoir discuté avec des utilisateurs potentiels de cet outil, il nous est apparu intéressant de ne pas se contenter de représenter une seule connexion, mais toutes les connexions ayant la même source afin de pouvoir comparer la connexion intéressante avec celles-ci.

Ces trois outils utilisent des visualisations relativement rudimentaires mais efficaces pour ce genre d'analyse. De plus, quelques interactions ont été développées afin de rendre ces visualisations plus productive. Nous présentons maintenant, en image, ces visualisations.

6.1 Analyse par source/destination

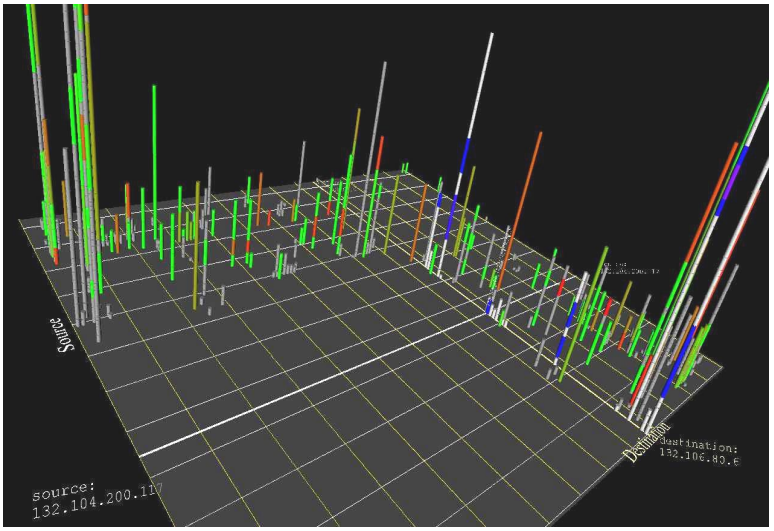


Figure 83 Cette visualisation permet d'analyser le trafic en fonction des sources et des destinations des connexions. Elle est organisée sous forme d'une matrice, avec les adresses sources sur l'axe des X et les destinations sur l'axe Y, de façon à analyser les connexions entre une source et une destination spécifiques. Ces connexions sont représentées par des cylindres, empilés les uns sur les autres s'il existe plusieurs connexions pour la même paire source/destination. La hauteur de ces cylindres représente la quantité de données de la connexion, et la couleur la variance de la fenêtre TCP. Afin de faciliter l'analyse d'une source ou d'une destination, lorsque l'utilisateur clique sur un cylindre, l'ensemble des connections ayant la même source ou la même destination sont mis en avant visuellement.

6.2 Analyse temporelle

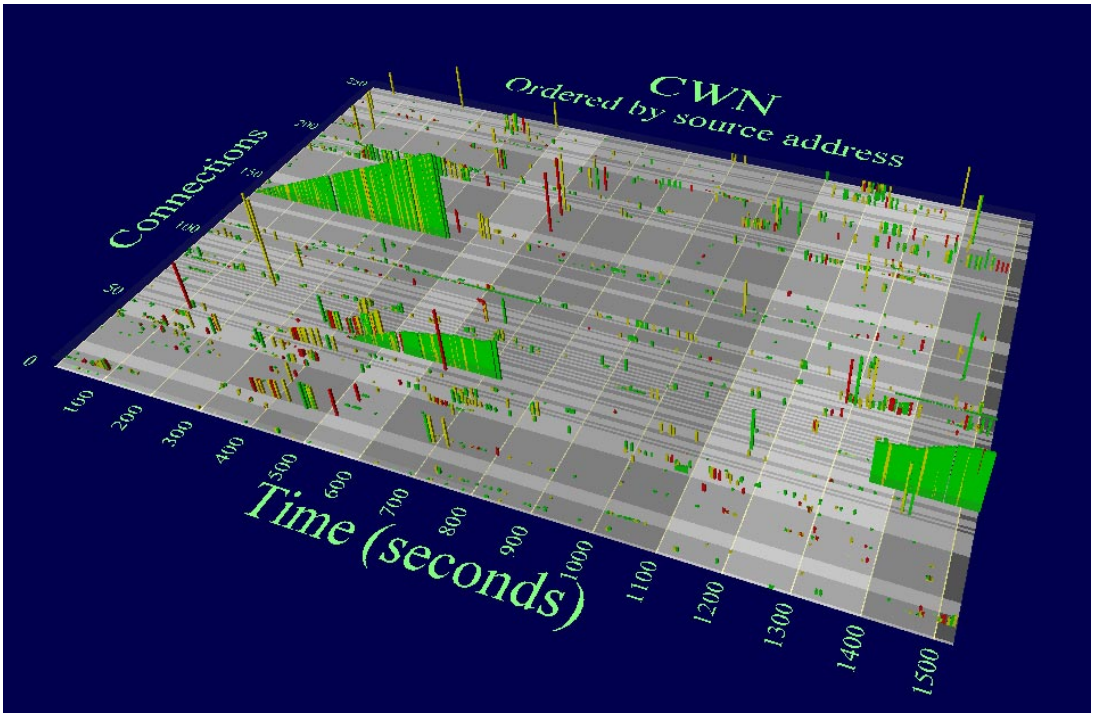


Figure 84 Cette visualisation présente l'évolution temporelle de la taille de la fenêtre TCP pour chaque connexion capturée. En plus de cette évolution, on peut également visualiser la qualité de la connexion qui est mappée sur la couleur. On peut également noter que la couleur du sol change de niveaux de gris selon la tranche de temps de façon à indiquer le nombre de mesures effectuées dans celle-ci.

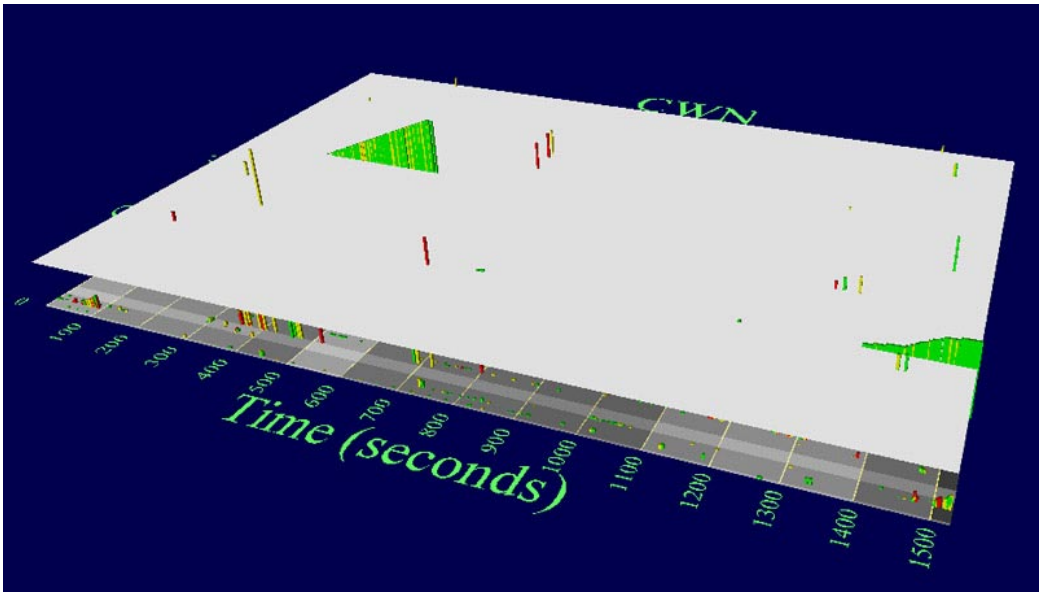


Figure 85 Même vue que la précédente figure mais avec un outil permettant de filtrer les connexions dont la taille de la fenêtre TCP est en dessous d'un seuil.

6.3 Analyse d'une source

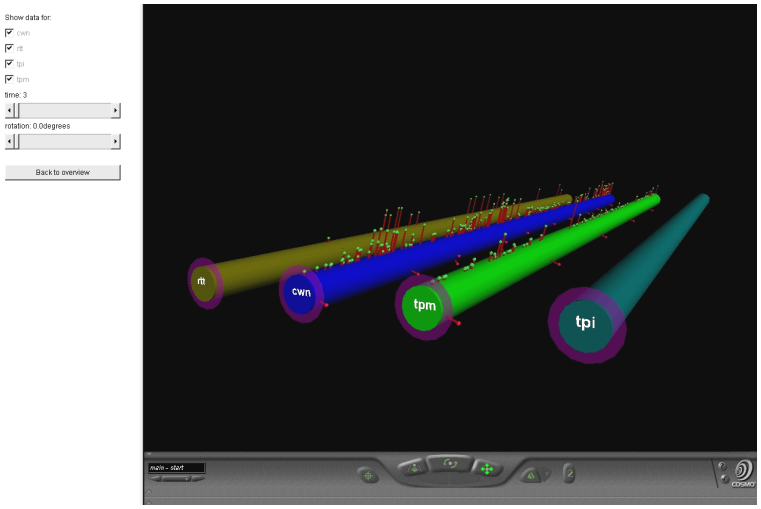


Figure 86 Cette vue permet d'analyser en détail tous les paramètres des connexions ayant la même source. Les quatre paramètres disponibles pour les connexions sont disposés sur quatre cylindres, permettant ainsi à l'utilisateur de comparer l'évolution temporelle de chaque paramètre. Afin de différencier les différentes connexions, celles-ci sont placés autour du cylindre. Une interaction (visible sur la gauche) permet de déplacer des anneaux synchronisés et placés autour de chaque cylindre, de manière à faciliter la comparaison temporelle entre chaque cylindre. Une autre interaction permet de faire tourner synchroniquement les cylindres sur eux-mêmes de manière à observer toutes les connexions.

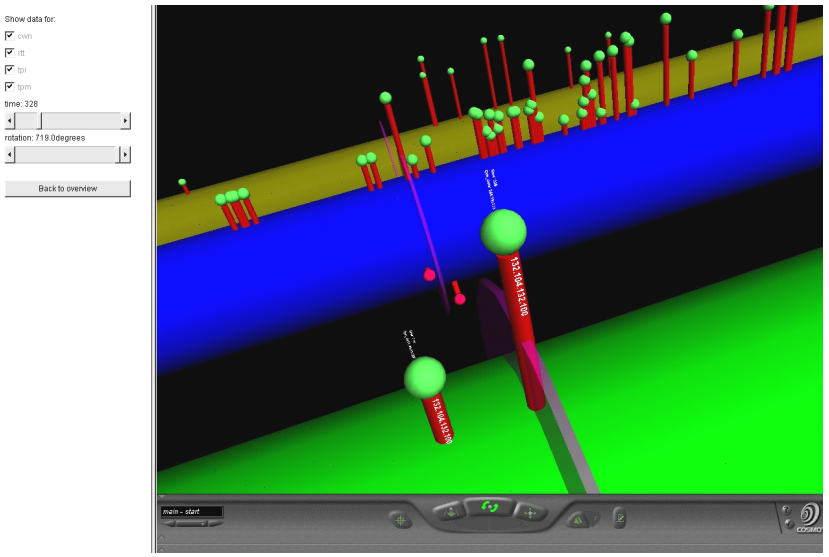


Figure 87 Zoom de la visualisation précédente. Chaque « allumette » (le cylindre avec une sphère au bout) représente une information sur une connexion à un instant t, la taille de l'allumette représentant la valeur de cette information et la couleur codant la connexion en elle-même (puisqu'une source peut avoir plusieurs connexions en même temps)

7 Analyse de serveur Web

L'analyse de serveur Web est un sujet à la mode tout particulièrement à cause l'explosion du commerce électronique. A des fins marketing (ou de curiosité), les administrateurs de sites doivent pouvoir fournir des réponses à des questions comme : d'où viennent les visiteurs ? Quand est-ce que ces visiteurs viennent ? Comment sont réparties les visites sur l'ensemble du site Web ?

La quantité d'informations à analyser peut être très importante dans le cas d'un serveur Web très fréquenté et la visualisation d'informations en 3D peut permettre de proposer des solutions à ce problème.

Il est à noter que ce type d'analyse pourrait être également réalisé pour des serveurs d'autre nature, comme un serveur d'application ou une base de données accessible par un autre protocole que http.

7.1 Analyse géographique des clients

L'analyse géographique permet à l'administrateur de situer et comparer par pays d'où proviennent les connexions vers le serveur Web. La visualisation présentée est relativement simple. Une carte du monde est affichée et des boîtes sont posées sur chaque pays. Le nombre de connexions par pays est représenté par la hauteur et la couleur de cette boîte.

Ce type de visualisation permet d'obtenir une vue d'ensemble synthétique, et de situer rapidement les pays d'où proviennent la majorité des clients.

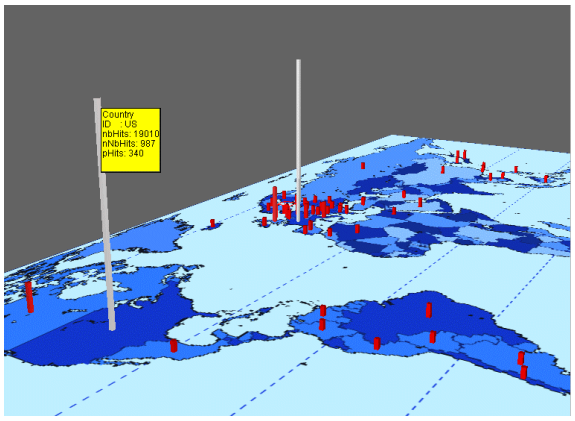


Figure 88 Vue globale de la répartition géographique des connexions sur le serveur Web d'Eurecom

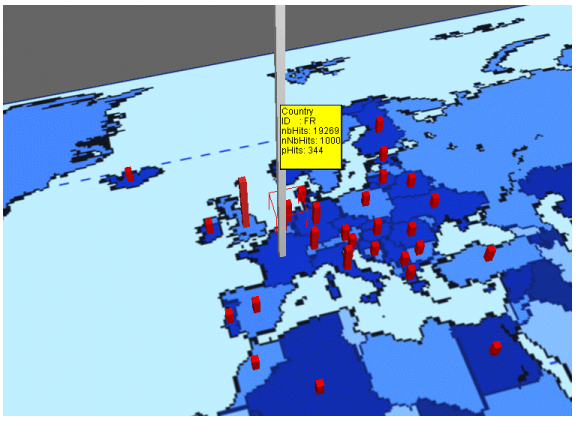


Figure 89 Une deuxième vue du même monde avec un zoom sur l'Europe

7.2 Analyse temporelle

L'administrateur d'un système de serveurs peut avoir besoin d'étudier l'évolution des connexions d'un point de vue temporel, par exemple pour étudier l'évolution de la fréquentation de son site ou pour savoir à quelles heures son site est le plus visité. Cette analyse peut devoir se faire sur une page en particulier ou sur la totalité des pages du site.

Pour cette analyse, nous avons développé une métaphore de bibliothèque divisée en plusieurs étagères dont les étages contiennent des livres. Dans cette métaphore, il existe donc trois niveaux de hiérarchie (étagère, étage, livre) qui nous permettent de présenter trois niveaux de segmentations temporelles (par mois, par jour et par heure). Chaque élément (étagère, étage, livre) possède des paramètres visuels permettant d'afficher des informations. Ces paramètres sont décrit aux figures suivantes.

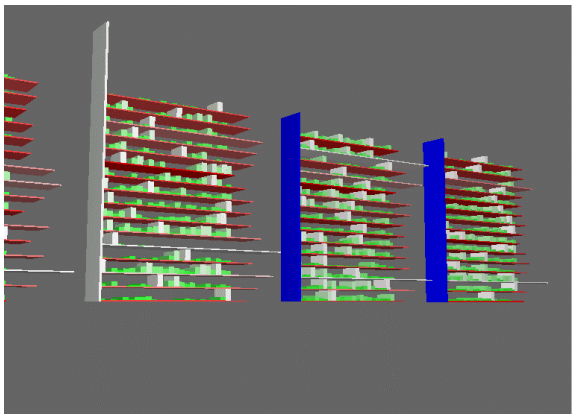


Figure 90 Métaphore de bibliothèque présentant des informations temporelles quant à la fréquentation d'un site Web. Chaque étagère représente un mois de fréquentation et on peut voir sa gauche, une bordure dont la taille et la couleur correspondent à la moyenne de la fréquentation sur le mois considéré

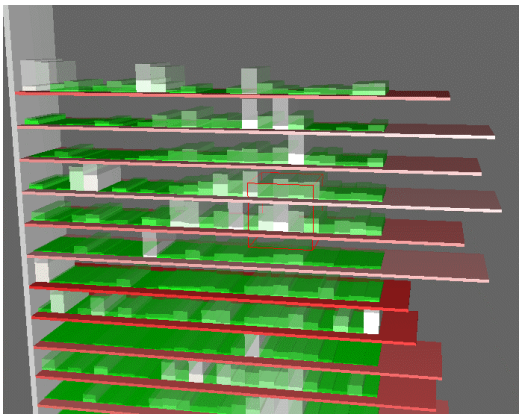


Figure 91 Zoom sur une étagère. Chaque étage représente un jour et la couleur et la taille représentent la fréquentation du jour associé. Chaque étage comporte 24 livres correspondant aux 24 heures d'un jour. Encore une fois la hauteur et la couleur du livre correspond à la fréquentation de l'heure associée.

7.3 Analyse des pages d'un site Web

L'analyse des pages d'un site Web est comparable à l'analyse de système de fichiers que nous avons vue précédemment : la structure hiérarchique du site doit être clairement visualisée tout en permettant une analyse de chaque élément de cette hiérarchie. Dans le cas de sites Web, cette analyse se résume à celle de la fréquentation de la page ou des pages contenues dans un répertoire.

Nous avons utilisé la même métaphore que dans le cas du système de fichier : la métaphore pyramidale. Les répertoires sont donc représentés par des boîtes empilées les unes sur les autres de manière à visualiser leur organisation hiérarchique. Les fichiers existant dans chaque répertoire sont représentés par des cylindres posés sur les boîtes. Chaque élément (boîte et cylindre) possède des paramètres visuels (hauteur et couleur) permettant d'afficher les informations relatives à la popularité de la page ou du répertoire associé. Deux types de visualisations ont été réalisés : le premier comportant seulement les répertoires de façon à obtenir une visualisation concise, et le secondcomportant en plus les fichiers de manière à obtenir une visualisation complète.

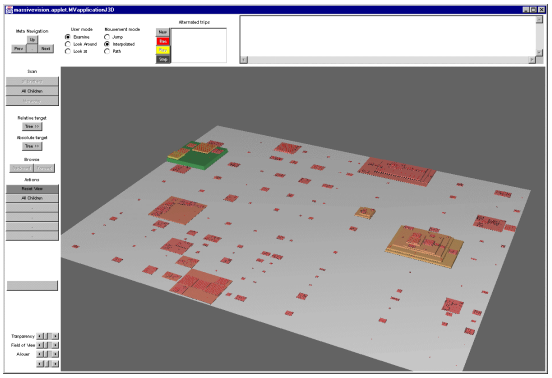


Figure 92 Vue globale des répertoires du site Web d'Eurecom, on peut tout de suite déterminer visuellement les répertoires les plus fréquentées (couleur verte).

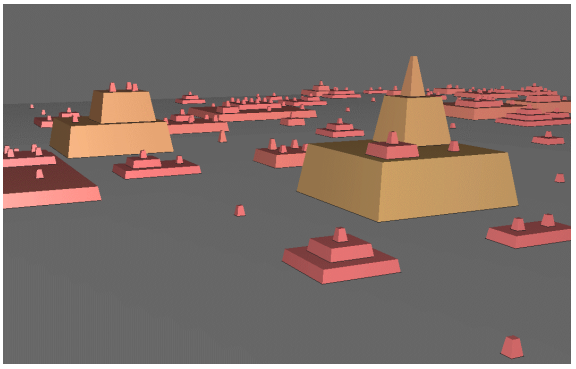


Figure 93 : Zoom sur une partie du monde 3D, chaque boîte correspond à un répertoire, sa couleur et sa taille permet d'analyser la fréquentation des pages contenues dans ce répertoire.

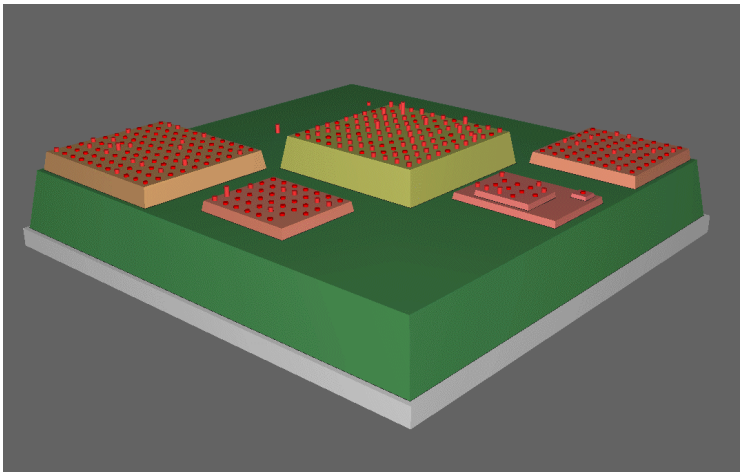


Figure 94 Une partie du site Web d'Eurecom particulièrement populaire. Dans cette visualisation, les fichiers sont visualisés, en plus de répertoires, sous la forme de cylindres posés sur les boîtes.

8 Conclusion

A travers les différentes expérimentations présentées dans ce chapitre, nous avons voulu montrer le potentiel de la plate-forme CyberNet en termes de création de visualisations diverses et variées. Nous avons aussi voulu montrer comment les techniques de visualisation 3D pouvaient constituer un atout majeur dans l'analyse d'informations, qu'elles soient dynamiques ou non, et quelle que soit leur origine. Quelques exemples ont aussi montré comment certaines interactions pouvaient ajouter de la valeur à ces visualisations.

Notre souhait serait à présent d'essayer de créer de nouveaux types de visualisations, particulièrement à base de métaphores, en utilisant les possibilités proposées par la plate-forme CyberNet. Nous voudrions aussi concrétiser nos idées concernant l'utilisation parallèle de multiples visualisations synchronisées des mêmes informations, car nous pensons que c'est une solution permettant d'appréhender certains problèmes difficiles à régler avec une seule visualisation. Enfin, nous aimerions également tester notre système pour des informations n'ayant aucun lien avec l'administration de réseau.

8 Conclusion & perspectives

Selon le rapport intitulé « How much information ? » [Lyman2000], la quantité d'informations produites chaque année dans le monde se situe entre 1 et 2 exabytes (10^{18} octets). Les auteurs estiment que 93% de ces informations sont numériques et que ce taux va continuer à augmenter. Une de leurs conclusions est qu' « il est clair que nous sommes tous en train de nous noyer dans un océan d'information. Le challenge est donc d'apprendre à nager dans cet océan plutôt que de s'y noyer. Une meilleure compréhension et de meilleurs *outils* sont nécessaires si nous voulons profiter pleinement du nombre d'informations toujours croissant décrit dans ce rapport ».

Les outils dont parlent ces auteurs permettent entre autres de chercher, d'analyser, de filtrer et de superviser des informations. Certains de ces outils sont classifiés sous l'expression « visualisation d'informations », car ils s'attachent à exploiter les remarquables capacités de perception visuelle propres à l'être humain.

Dans ce mémoire, nous avons proposé des solutions permettant la construction d'outils de visualisation d'informations dont les spécificités sont de traiter des informations dynamiques et de les visualiser à l'aide de mondes 3D interactifs. Pour cela, nous avons défini plusieurs modèles allant de la structuration des informations à visualiser à la construction des mondes 3D interactifs, en passant par la navigation assistée dans ces mondes. Ces modèles ainsi que les autres travaux réalisés dans cette thèse sont maintenant résumés.

Modèle de données

Le modèle de données que nous avons présenté a pour objectif la définition de *services*, concept permettant de regrouper et d'organiser toutes les informations nécessaires à une tâche. Les services étant destinés à être visualisés, le modèle de données fait la distinction entre les informations permettant la construction du service, et celles destinées à être visualisés. Un des points clés de ce modèle est sa capacité à gérer le caractère dynamique des informations grâce à un mécanisme permettant aux services de se construire dynamiquement, en fonction des informations existantes.

Modèle de visualisation 3D

Nous avons défini un modèle de visualisation 3D conçu spécialement dans le but de créer des mondes 3D destinés à visualiser des informations. Pour cela, le modèle permet de définir des composants graphiques offrant des paramètres

visuels sur lesquels les informations sont mappées. Ces composants peuvent être assemblés de manière à construire un monde 3D complet, et, grâce à leur conception orientée objet, simplifient l'expérimentation de nouvelles visualisations. En parallèle, nous avons également abordé le problème d'instabilité visuelle lié au caractère dynamique des mondes 3D et proposé quelques solutions possibles.

Modèle de navigation dans les mondes 3D

Il nous est apparu très rapidement que les visualisations 3D perdaient beaucoup de leur intérêt si l'utilisateur ne pouvait pas se déplacer correctement en leur sein, tâche difficile à réaliser avec les outils traditionnels de navigation. Notre modèle de navigation repose sur la définition de *centres d'intérêt* (CDI), représentant chacun une partie du monde 3D où l'utilisateur est susceptible de porter son attention, et sur des outils de navigation contrainte permettant à l'utilisateur de se déplacer automatiquement entre les CDI ou d'en observer un en particulier. Nous avons également mis en avant la nécessité d'offrir des outils pouvant s'adapter à la structure du monde 3D et des objets le composant, et non d'offrir seulement des outils génériques.

Modèle de mondes 3D interactifs

Nous avons également expliqué comment nous construisons des mondes 3D interactifs à base de composants réunissant les concepts proposés dans les modèles de visualisation et de navigation. En effet, il nous a paru nécessaire de ne pas séparer la partie graphique de la partie interactive afin de pouvoir adapter les interactions (dont la navigation) en fonction des éléments visuels composant les mondes 3D.

Mappage des services

Enfin, nous avons exposé une méthode permettant de mapper les informations contenues dans un service sur une métaphore, c'est-à-dire un type de monde 3D interactif, et ainsi relier entre eux tous les concepts présentés dans cette thèse.

Plate-forme logicielle

Un chapitre a été consacré à un aspect plus technique de cette thèse, c'est à dire la réalisation de la plate-forme logicielle distribuée intégrant tous les concepts vus ci-dessus. Cette plate-forme a soulevé de nombreux problèmes techniques mais s'est révélée un formidable outil d'expérimentation qui nous a permis de faire évoluer nos différents modèles.

Perspectives

Les travaux réalisés dans cette thèse ont pour résultat un environnement complet pour la visualisation d'informations dynamiques et distribuées à l'aide de mondes 3D interactifs. A l'aide de cet environnement, de nombreuses

expérimentations ont été menées en vue d'évaluer d'une part la faisabilité de ce type d'outils, et d'autre part leur utilité et leur pertinence. Les différentes expériences réalisées ont montré que ces outils s'avéraient très prometteurs dans des applications de supervision de vastes quantités d'informations dynamiques comme en gestion de réseaux. Nous avons attaché un soin important à la prise en compte de la principale critique en provenance des utilisateurs : la difficulté de naviguer dans un environnement 3D en développant un concept global de monde 3D interactif pour lesquels, visualisation, navigation et interactions sont intimement liées.

Mais de nombreuses améliorations pourraient être apportées à cet environnement. En premier lieu, l'environnement permet de ne visualiser que des informations structurées sous la forme d'arbres, et non de graphes. Cette caractéristique limite donc les types de services qu'il est possible de créer, même si, comme nous l'avons déjà dit, les visualisations de grands graphes manque de clarté, et qu'il est souvent préférable de visualiser à la place plusieurs arbres extraits du graphe. Une possibilité pour lever cette limitation serait de modifier le modèle de donnée de manière à ce qu'il accepte des liens spéciaux entre les différents nœuds de l'arbre, ces liens étant pris ensuite en compte par un module particulier du modèle de visualisation.

Un deuxième point méritant des recherches approfondies est l'interaction. Outre la navigation, que nous avons étudiée particulièrement mais qui pourrait être étendu afin de prendre en compte l'aspect dynamique du monde 3D, il nous semble qu'il serait intéressant d'établir un modèle complet d'interaction permettant d'exploiter pleinement la richesse et les possibilités offertes par les mondes 3D. Nous avons déjà effectué quelques expérimentations comme dans la métaphore du bâtiment où l'utilisateur a la possibilité d'adapter le monde à ses besoins en cachant ou rendant transparents certains éléments du monde. Cet exemple ainsi que les autres que nous avons évoqués (synchronisation de plusieurs visualisations, sélection, possibilités de résumer visuellement une partie complexe du monde), peuvent aider à la définition d'un modèle d'interaction qui pourrait s'intégrer dans l'environnement déjà existant.

Une limitation de notre environnement est qu'il n'existe pas de mécanisme permettant de cantonner le volume d'informations à visualiser. Ce type de mécanisme est nécessaire lorsque l'on veut visualiser des volumes d'informations gigantesque, comme par exemple une représentation visuelle du Web, car il n'est pas possible de créer un service contenant toutes les informations, ni un monde 3D permettant de les visualiser. Une solution possible est de choisir les informations contenues dans le service en fonction de la position de l'utilisateur dans le monde 3D, le service éliminant certaines informations pour en collecter d'autres lorsque l'utilisateur se déplace dans le monde. Ainsi l'utilisateur verrait le monde se construire devant ses yeux au fur et à mesure de sa navigation.

Un point dont nous n'avons pas discuté dans ce mémoire est la possibilité de mapper automatiquement un service sur un monde 3D interactif. Actuellement, il est nécessaire de définir manuellement des associations entre les éléments du service et ceux du monde 3D, mais ce travail peut être réalisé automatiquement Pour cela, il est nécessaire d'une part de caractériser les informations, et d'autre part de caractériser les paramètres visuels en fonction de leur efficacité à représenter un certain type d'information, puis d'établir un algorithme permettant

de réaliser les associations. Ce sujet est en cours de conception par une doctorante travaillant sur le projet CyberNet.

D'autres axes de recherches sont également à explorer. Nous n'avons pas abordé le problème de la rétroaction des actions de l'utilisateur sur le monde réel, nous limitant en cela à la notion de supervision. Par exemple, dans le cas de la gestion de réseaux, il pourrait être intéressant pour le superviseur de pouvoir agir sur les éléments qu'il visualise de manière à ce qu'il dispose d'un outil global. Nous ne nous sommes pas non plus intéressés aux problèmes d'historique (visualiser/rejouer des événements passés), cette possibilité pouvant être cruciale dans l'analyse de certains problèmes récurrents.

Une dernière perspective qui nous tient à cœur serait de créer de nouveaux types de mondes 3D interactifs exploitant pleinement le potentiel existant dans notre environnement. Comme on peut le lire dans [Card99,p.640], nous pensons que « les visualisations ne sont pas facile à inventer et que leur nombre actuel n'est pas encore très grand. Mais nous sentons qu'il y a encore un nombre important de visualisations qui attendent d'être découvertes ».

Annexe 1 : Acronymes

- **CG** : Composant graphique
- **CI** : Composant d'interaction
- **CM** : Composant métaphorique
- **CN** : Composant de navigation
- **GP** : Gestionnaire de placement
- **CDI** : Centre d'intérêt

Annexe 2 : Lexique

Termes relatifs au modèle de données

- **Entité :** Unité d'information au sein du projet CyberNet. Une entité est typée et représente un concept comme, en gestion de réseau, un serveur ou un switch ou encore un processus. Chaque type d'entité possède une liste d'attribut permettant de caractériser une instance particulière (par exemple les attributs d'une entité de type « ordinateur » sont : le nom de l'ordinateur, son adresse IP, sa charge CPU et l'utilisation de sa mémoire).

- **Requête :** Ensemble de références à des entités. Ces références sont définies en fonction de critères relatifs au type et/ou aux attributs des entités. Par exemple on peut créer une requête référençant toutes les entités de type « ordinateur » ou toutes les entités du même type et ayant une charge CPU supérieure à un certain seuil.

- **Organisateur :** Nœud de l'arbre permettant de décrire un service. Les organisateurs ont la possibilité de créer d'autres organisateurs qu'ils s'ajoutent comme fils, et ainsi de construire dynamiquement le service dont ils font partie. Les organisateurs utilisent les requêtes pour obtenir des références à des entités qui peuvent soit être destinées à être visualisées, soit engendrer la création de nouveaux organisateurs.

- **Service :** Concept identifiant et regroupant toutes les informations nécessaires à une tâche. Un exemple typique en gestion de réseaux est le service permettant d'analyser la topologie d'un réseau, un autre exemple est le service permettant de superviser un ensemble d'ordinateurs. Dans CyberNet, un service est représenté par un arbre dont les nœuds sont des organisateurs.

Termes relatifs au modèle de visualisation

- **Composant Graphique (CG)** : Objet prédéfini et réutilisable intervenant dans la réalisation graphique d'un monde 3D. Chaque type de CG a une fonction particulière et en assemblant des CG, à la manière des célèbres legos, il est possible de construire une hiérarchie de CG décrivant un monde 3D complet. Il existe deux types de CG : les gestionnaires de placement (GP) et les glyphes.

- **Glyphe :** Composant graphique permettant de définir un objet 3D, éventuellement animé, comme une maison ou plus simplement une sphère. Les glyphes fournissent une liste de paramètres visuels permettant de contrôler leurs représentations graphiques ; par exemple les paramètres visuels d'un glyphe de type sphère peuvent être ses dimensions, sa couleur et sa transparence.

- **Gestionnaire de placement (GP) :** Composant graphique permettant de spécifier une politique de placement dans l'espace d'un ensemble de CG, cet ensemble étant constitué par les fils CG que les GP peuvent posséder. Par exemple un GP peut placer ses fils sur un échiquier ou alors les empiler les uns sur les autres. Les GP peuvent également agir sur les dimensions de leurs fils de manière à respecter des contraintes visuelles (comme par exemple que ses propres dimensions ne changent pas)

Termes relatifs à la navigation et à l'interaction

- **Centre d'intérêt (CDI) :** Partie du monde sur laquelle l'utilisateur est susceptible de porter son attention. Un CDI est constitué d'un ensemble d'éléments graphiques. Par exemple, dans une métaphore de ville, l'utilisateur peut avoir comme CDI un bâtiment, un quartier constitué de plusieurs bâtiments ou encore la ville tout entière constituée de tous les quartiers.

- **Composant de navigation (CN) :** Composant permettant de spécifier certaines opérations de navigation associées à un CDI. Ces opérations sont définies en fonction de la représentation visuelle et du contexte dans lequel se trouve le CDI associé. Les différents CN existants dans un monde 3D peuvent communiquer entre eux de façon à réaliser certaines opérations (comme la navigation d'un CDI à un autre).

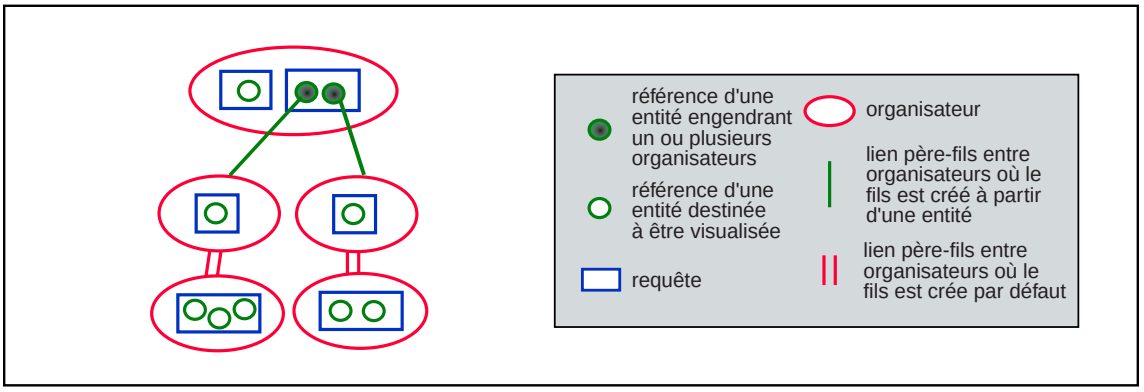
- **Composant d'interaction (CI) :** Composant permettant de spécifier certaines opérations d'interactions associées à un CDI, comme par exemple le changement de certaines propriétés visuelles du CDI.

Termes relatifs aux mondes 3D interactifs

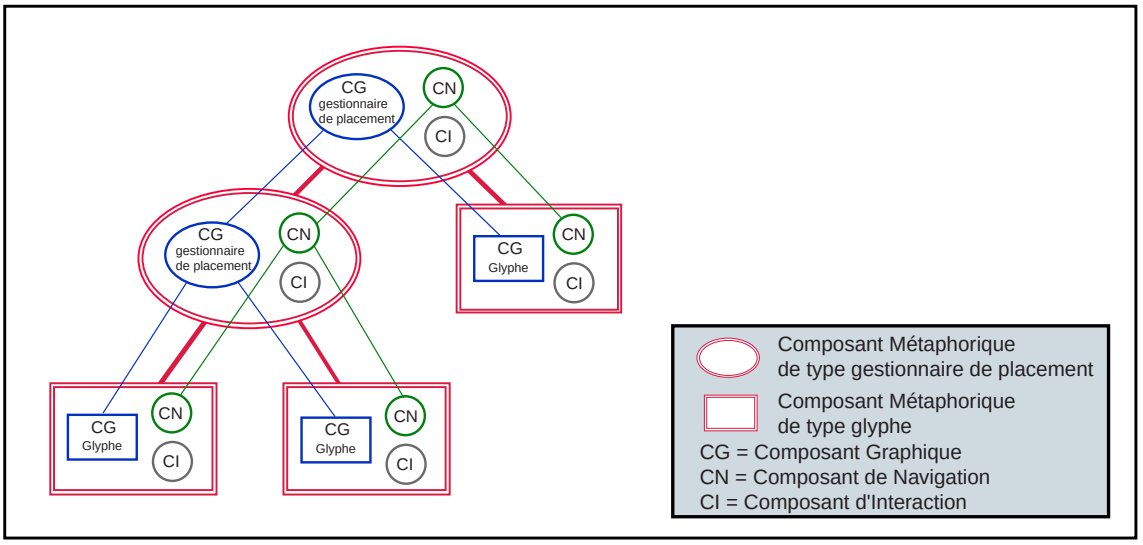
- **Composant métaphorique (CM) :** Un composant métaphorique contient un CG, un CN et un CI. Il permet de participer à la création d'un monde 3D interactif. A la manière des CGs il est possible de les assembler afin de construire une hiérarchie décrivant un monde 3D interactif complet. Il existe deux types de CM : les CM contenant un GP et les CM contenant un glyphe.

- **Métaphore :** Dans le système CyberNet, une métaphore est un type de monde 3D interactif utilisé pour visualiser des informations. Elle associe un type de représentation visuelle 3D (une ville par exemple) à des interactions permettant, en particulier, de naviguer dans un monde 3D construit sur la base de cette représentation.

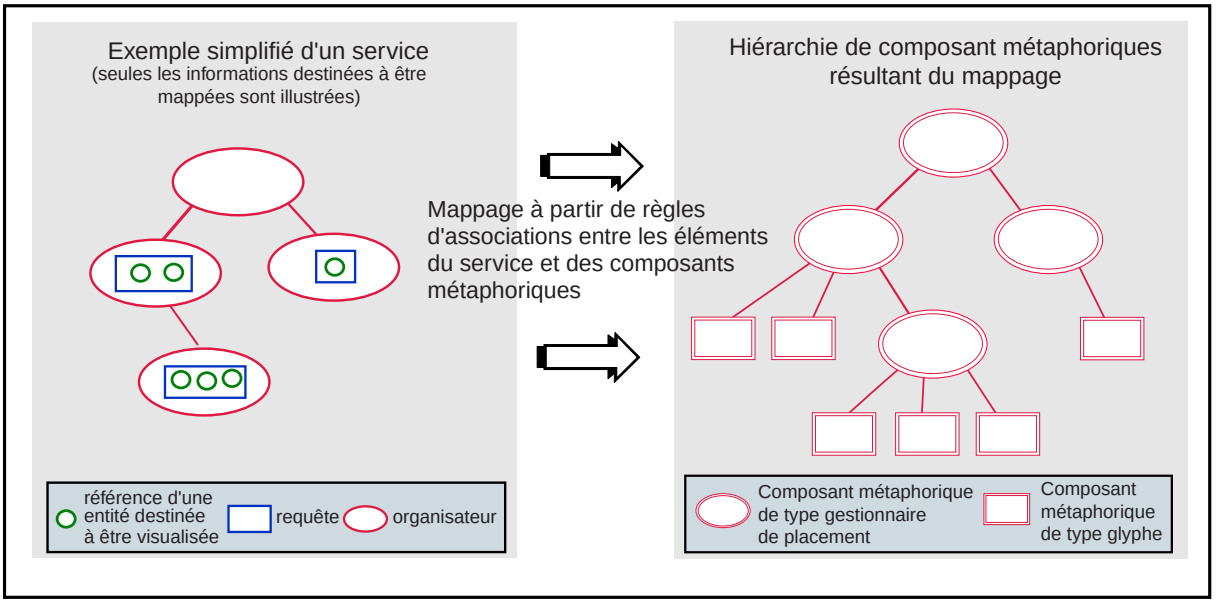
Annexe 3 : Figures de référence



Exemple de service



Exemple d'une hiérarchie de composants métaphoriques permettant de décrire un monde 3D interactif. En surimposition, on peut voir la hiérarchie de composants graphiques décrivant la partie visuelle du monde, et la hiérarchie de composants de navigation permettant de réaliser des opérations de navigation spécifiques, ces deux hiérarchies étant induites par celle des composants métaphoriques.



Exemple du mappage d'un service sur une hiérarchie de composants métaphoriques

Références bibliographiques

[Abel00] Abel P., Gros P., Loisel D., Russo Dos Santos C. & Paris J.P., *Automatic Construction of Dynamic 3D Metaphoric Worlds : An Application to network management*, SPIE Electronic Imaging, Visual Data Exploration and Analysis session, San Jose, January 2000

[Abel00b] Abel P., Gros P., Loisel D., Russo Dos Santos C. & Paris J.P., *CyberNet : A framework for Managing Networks Using 3D metaphoric worlds*, Annales des telecommunications, tome 55, N 3/4, April 2000.

[Abel99] Abel P., Gros P., Loisel D., Russo Dos Santos C. & Paris J.P. *Construction automatique de mondes virtuels représentant le comportement dynamique d'un réseau*. Proceeding of CORESA99, Sophia-antipolis, June 1999

[Andrews95] Andrews K., *Visualizing Cyberspace: Information Visualisation in the Harmony Internet Browser*, Proceedings of the IEEE Symposium on Information Visualization (InfoVis'95), IEEE CS Press, pp. 97-105

[Andrews96] Andrews K., Pichler M. & Wolf P.. *Towards rich information landscapes for visualising structured web spaces*. In Proc.2nd IEEE Symposium on Information Visualization (InfoVis'96), pages 62-63, San Francisco, CA, October 1996

[Andrews97] Andrews K., Wolte J. & Pichler M., *Information Pyramids: A New Approach to Visualising Large Hierarchies*. In IEEE Visualization'97, Late Breaking Hot Topics Proc., pages 49-52, Phoenix, Arizona (1997).

[Andrews98] Andrews K. & Heidegger H., *Information Slices: Visualising and Exploring Large Hierarchies using Cascading, Semi-Circular Discs*, In Late Breaking Hot Topic Paper, IEEE Symposium on Information Visualization (InfoVis'98), pages 9-12, Research Triangle Park, North Carolina

[Battista94] di Battista G., Eades P., Tamassia R. & Tollis I.G., *Algorithms for drawing graphs: an annotated bibliography*, Computational Geometry: Theory and Applications , 4 (5), pp. 235-282, 1994.

[Battista99] di Battista G., Eades P., Tamassia R., & Tollis I.G., *Graph Drawing: Algorithms for the Visualisation of Graphs* , Prentice Hall, 1999.

[Beaudoin96] Beaudoin L., Parent M.A. & Vroomen L. C., *Cheops: A Compact Explorer for Complex Hierarchies*, In Proc. IEEE Visualization'96, pages 87-92, San Francisco, California

[Becker95] Becker R. A., Eick S. G., and. Wilks A. R., *Visualizing network data*, IEEE Transactions on Visualization and Graphics, 1(1):16-28, 1995.

[Bederson94] Bederson B. & Holland J.D., *Pad++ : A Zooming Graphical Interface for Exploring Alternate Interface Physics*, Proceedings of UIST'94, ACM Symposium on User Interface Software and Technology, Marina Del Rey, California, USA

[Bertin67] Bertin J., *Sémiologie graphique - Les diagrammes, les réseaux, les cartes*, Mouton / Gauthier-Villars, Paris, 1967

[Bertin77] Bertin J., *La Graphique et le Traitement Graphique de l'Information,* Flammarion, Paris,1977

[Card97] Card S. K. & Mackinlay J., *The Structure of the Information Visualization Design Space*, IEEE Symposium on Information Visualization'97, Phoenix, Arizona, 1997

[Card99] Card S.K., Mackinlay J.D., & Shneiderman B., *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufmann Publishers, San Francisco, 1999

[Carrière95] J. Carrière & R. Kazman, *Research Report: Interacting with Huge Hierarchies: Beyond Cone Trees*, Proceedings of the IEEE Conference on Information Visualisation `95, IEEE CS Press, pp. 74-81,

[Chen88] Chen, M., Mountford, S.J., Sellen, A., *A Study in Interactive 3-D Rotation Using 2-D Control Devices*, Computer Graphics, Vol.22 #4, 1988.

[Chen99] Chen C., *Information Visualisation and Virtual Environments,* Springer Verlag, 1999

[Chernoff73] Chernoff H., *The Use of Faces to Represent Points in k-Dimensional Space Graphically,* Journal of American Statistical Association, Vol. 68, pp.361-368.

[Chuah95] Chuah M.C., Roth S.F., Mattis J. & Kolojejchick J., *SDM: Selective Dynamic Manipulation of Visualizations,* Proceedings of the ACM Symposium on User Interface Software and Technology, Pittsburgh, PA, November 1995, pp. 61-70

[Chuah98] Chuah M. & Eick S., *Information rich glyphs for software management data*, IEEE Computer Graphics and Applications, Vol. 18, No. 4, pp. 24-29, 1998.

[Cichlid] http://moat.nlanr.net/Software/Cichlid/

[Cleveland88] Cleveland W. S. & McGill M. E, Editors. *Dynamic Graphics for Statistics*. Belmont, CA: Wadsworth and Brooks/Cole Advanced Books and Software,1988.

[Cleveland93] Cleveland W. S., *Visualizing Data*, AT&T Bell Laboratories, Murray Hill, NJ, Hobart Press, Summit NJ, 1993.

[CORBA] http://www.corba.org

[Cox92] Cox D. & Patterson R, *Visualization Study of the NSFNET*, online videos, http://www.ncsa.uiuc.edu/SCMS/DigLib/text/technology/Visualization-Study-NSFNET-Cox.html

[Cox95] Cox K.C. & Eick S.G., *3D Displays of Internet Traffic*, IEEE Information Visualization Symposium, October 30-31st 1995, Atlanta, USA.

[Cox96] Cox K., Eick S., and He T., *3D Geographic Network Displays,* SIGMOD Record, 25(4), December 1996, pp. 50-54

[Crutcher95] Crutcher L., Lazar A.A., Feiner S. & Zhou M., *Managing Networks Through a Virtual World***,** IEEE Parallel & Distributed Technology, Vol. 3, No. 2, Summer 1995, pp. 4-13.

[Cruz95] I. F. Cruz & J. P. Twarog*, 3D Graph Drawing with Simulated Annealing*, Proceedings of the Symposium on Graph Drawing GD `95, Springer-Verlag, pp. 162-165

[CyberAtlas] http://www.cybergeography.org/atlas/atlas.html

[Darken93] Darken Rudolph P. & Silbert John L., *A Toolset for Navigation in Virtual Environments*, In Proceedings of ACM User Interface Software and Technology (UIST'93), New York, 1993.

[Darken96] Darken R.P., & Sibert J.L., *Navigating in Large Virtual Worlds*, The International Journal of Human-Computer Interaction, **8**(1), pp. 49-72. (1996).

[Delaney99] Delaney B., *The NYSE's 3D Trading Floor*, IEEE Computer Graphics & Applications*,* Vol. 19, No. 6, November/December 1999

[Eades84] P. Eades, *A Heuristic for Graph Drawing*, Congressus Numerantium, 42, pp. 149-160, 1984.

[Eades92] P. Eades, *Drawing Free Trees*, Bulletin of the Institute for Combinatorics and its Applications, pp. 10-36,1992

[Edwards97] Edwards J., & Hand C., *MaPS: Movement and Planning Support for Navigation in an Immersive VRML Browser*. Proceedings of The Second Symposium on the Virtual Reality Modeling Language (VRML'97), Monterey, California, USA, February 1997, pp65-73

[Eick96] Eick, S. G., *Aspects of network visualization*, IEEE Computer Graphics and Applications, 16(2), pp. 69-72, March 1996

[Eleftherios 99] Eleftherios E., Koutsofios E., North S. & Keim, D. *Visualizing Large Telecommunication Data Sets*, IEEE Computer Graphics and Applications, Mai/June 1999 pp 16-19.

[Fairchild88] Fairchild K. M., Poltrock S. E. & Furna, G. W., *SemNet: Three-Dimensional Representations of Large Knowledge Bases*, In Guindon, R., editor, Cognitive Science and its Applications for Human-Computer Interaction, pages 201-233. Lawrence Erlbaum, Hillsdale, New Jersey.

[Feiner93] Feiner S., Zhou M., Crutcher L. & Lazar A.A., *A Virtual World for Network Management***,** Proceedings of the IEEE Virtual Reality Annual Symposium, Seattle, WA, October 18-22, 1993.

[Fourthplanet] http://www.fourthplanet.com/

[Frecon98] Frécon E. & Smith G., *WebPath - A three-dimensional Web History*, IEEE Symposium on Information Visualization'98, Chapel Hill. NC, USA.

[Gabbard97] Gabbard J., Hix D., *A taxonomy of Usability Characteristics in Virtual Environments*, Deliverable to Office of Naval Research, Department of Computer Science, Virginia Polytechnic Institute and State University. November 1997.

[Gamma96] E. Gamma et al., *Design patterns : elements of reusable object-oriented software*, Addison Wesley, 1996.

[Gelernter82] D. Gelernter & A. Bernstein, *Distributed Communication via a Global Buffer*, Proc. ACM Symp. on Principles of Distributed Computing, pages 10–18, 1982.

[Gros00] Gros P., Abel P., Loisel D., Russo Dos Santos C. & Paris J.P., *Experimenting Service-Oriented 3D Metaphors For Managing Networks Using Virtual Reality*, VRIC2000, Laval 2000.

[Hand97] Hand C., *A Survey of 3D Interaction Techniques*, Computer Graphics Forum*,* 16(5), pp 269-281. (Dec 1997)

[Hanson97] Hanson A. J. & Wernert E., *Constrained 3D navigation with 2D controllers*, Proceedings of IEEE Visualization '97*,* pages 175-182.

[He98] T. He & S. Eick, *Constructing Interactive Network Visual Interface*, Bell Labs Technical Journal, 3(2), Spring 1998.

[Hendley95] R. J. Hendley, N. S. Drew, A. M. Wood, & R. Beale, *Narcissus: Visualising Information", Proceedings of the IEEE Symposium on Information Visualization*, IEEE CS Press, pp. 90-96, 1995.

[Herman00] I. Herman, G. Melancon, & M. S. Marshall, *Graph Visualisation and Navigation in Information Visualisation: a Survey*, IEEE Transactions on Visualization and Computer Graphics, vol. 6, pp. 24-43, 2000

[Ingram95] Ingram R., Benford S., *Legibility Enhancement for Information Visualisation,* in Proceedings of Visualization'95, Atlanta, Georgia, November 1995

[Inselberg85] Inselberg A., *The Plane with Parallel Coordinates,* The Visual Computer, Vol. 1, 1985, pp.69-97.

[Inselberg90] Inselberg A., Dimsdale B., *Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry*, Proceedings of the IEEE Visualization '90, San Francisco, CA, 1990, pp.361-370.

[JAVA] http://java.sun.com

[Java3D] http://java.sun.com/products/java-media/3D/

[JavaSpaces] http://java.sun.com/products/javaspaces/

[Jeong98] C.-S Jeong & A. Pang, *Reconfigurable Disc Trees for Visualizing Large Hierarchical Information Space*, Proceedings of the IEEE Symposium on Information Visualisation '98 (InfoVis'98), IEEE CS Press

[Johnson91] Johnson B. & Shneiderman, B., *Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures.* In Proc. IEEE Visualization '91, pages 284-291, San Diego, California (1991). IEEE Computer Society.

[Kahani96] Kahani M. & Beadle H. *WWW-based 3D Distributed, Collaborative Virtual Environment for Telecommunication Network Management*, Proc. Australian Telecommunication Networks and Applications Conference (ATNAC'96), December 1996, pp. 483-488.

[Kaye97] Kaye T., *The tracking viewpoint,* online paper, http://reality.sgi.com/tomk/demos/vrml2/TrackingViewpoint/readme.html

[Kazman96] R. Kazman & J. Carrière, *An Adaptable Software Architecture for Rapidly Creating Information Visualizations*, Proceedings of Graphics Interface '96. Toronto, ON, May 22-24, 1996.

[Keim96] Keim D. A., *Pixel-oriented Visualization Techniques for Exploring Very Large Databases*, Journal of Computational and Graphical Statistics, Vol. 5, No. 1, 1996, pp. 58-77.

[Lakoff80] Lakoff, G. & Johnson, M., *Metaphors We Live By*, University of Chicago Press, Chicago,1980

[Lamping95] J. Lamping, R. Rao, & P. Pirolli, *A Focus+context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies,* Human Factors in Computing Systems, CHI '95 Conference Proceedings, ACM Press, 1995

[Lamping96] J. Lamping & R. Rao, *The Hyperbolic Browser: A Focus+context Technique for Visualizing Large Hierarchies*, Journal of Visual Languages and Computing, 7(1), pp. 33-55, 1996

[Lewis99] Lewis L., *Service level management for enterprise networks*, Artech House, Boston,1999.

[Lyman2000] Lyman P., Varian H.R, Dunn J., Strygin A. & Swearingen K. , *How much Information ?,* Home page project :http://www.sims.berkeley.edu/how-much-info/

[Lynch60] Lynch K., *The Image of the City,* MIT Press, Cambridge, MA.

[Mackinlay86] Mackinlay J. D., *Automating the Design of Graphical Presentations of Relational Information.* ACM Transactions on Graphics, 5(2), 110–141.(1986)

[Mackinlay90] Mackinlay J.D., Card S.K., Robertson G.G., *Rapid Controlled Movement Through a Virtual 3D Workspace*, ACM Computer Graphics, 24(4) pp 171-176, August 90.

[Map.net] Visualisation 2D et 3D du répertoire de sites Web dmoz.org http://map.net

[Marrin96] C. Marrin., *External Authoring Interface Reference*, http://www.vrml.org/WorkingGroups/vrml-eai/ ExternalInterface.html

[Martin90] Martin J.H., *A computational Model of Metaphor Interpretation,* Academic Press, 1990

[Massari97] Massari A., Saladini L., Hemmje M. & Sisinni F., *Virgilio: A Non-Immersive VR System To Browse Multimedia Databases* , Proc. of the IEEE International Conference on Multimedia Computing and Systems 1997, IEEE Computer Society Press, pp 573-580

[Mukherjea95a] Mukherjea S. & Foley J.D., *Requirements and Architecture of an Information Visualization Tool*, In Database Issues for Data Visualization: IEEE Visualization '95 Workshop, October 1995, Atlanta, Lecture Notes in Computer Science, Vol. 1183, Springer-Verlag 1996.

[Mukherjea95b] Mukherjea S., Foley J.D. & Hudson S., *Visualizing Complex Hypermedia Networks through Multiple Hierarchical Views*, Proceedings of ACM CHI 1995, May 1995, Denver, Colorado.

[Mukherjea95c] Mukherjea S. & Foley J.D., *Visualizing the World-Wide Web with the Navigational View Builder*, Computer Networks and ISDN Systems 27:1075-1087, 1995

[Munzer97] T. Munzner, *H3: Laying out Large Directed Graphs in 3D Hyperbolic Space*, Proceedings of the 1997 IEEE Symposium on Information Visualization (InfoVis'97), IEEE CS Press, pp. 2-10, 1997.

[Neale97] Neale D.C. & Carroll J.M**.,** *The role of metaphors in user interface design*, In M. Helander & T.K. Landauer (Eds.) Handbook of Human-Computer Interaction, Second Edition, North Holland, Amsterdam,1997, pp. 441-462.

[Perfomance] http://www.sgi.com/software/co-pilot/

[Playfair1786] Playfair W., *The Commerical and Political Atlas*, Londres,1786

[Reingold81] E.M. Reingold & J.S. Tilford, *Tidier Drawing of Trees*, IEEE Transactions on Software Engineering, 7(2), pp. 223-228, (1981).

[Rekimoto93] J. Rekimoto & M. Green, *The Information Cube: Using Transparency in 3D Information Visualisation*, Proceedings of the Third Annual

Workshop on Information Technologies & Systems (WITS'93), 1993.

[Risch97] Risch J., Rex D., Dowson S. et al., *The STARLIGHT Information Visualization System*, Proceedings of IEEE International Conference on Information Visualization, pp. 42- 49, London, 1997

[RMI] http://java.sun.com/products/jdk/rmi/

[Roberston91] G.G. Robertson, J.D. Mackinlay, & S.K. Card, *Cone Trees: Animated 3D Visualizations of Hierarchical Information*, Human Factors in Computing Systems, CHI '91 Conference Proceedings, ACM Press, pp. 189-194.

[Robertson00] Robertson, G., van Dantzich, M., Robbins, D., Czerwinski, M., Hinckley, K., Risden, K., Thiel, D. and Gorokhovsky, V., *The Task Gallery: a 3D window manager*, in Proceedings of CHI 2000, ACM Press, The Hague, Netherland,2000.

[Roth97] Roth S. F., Chuah M. C., Kerpedjiev S., Kolojejchick J. A., & Lucas P. *Towards an Information Visualization Workspace: Combining Multiple Means of Expression*, Human-Computer Interaction Journal, Volume 12, Numbers 1 & 2, 1997,pp. 131-185.

[Russo00] Russo Dos Santos C., Gros P., Abel P., Loisel D. & Paris. J.P. .*Mapping Information Onto 3D virtual worlds*, IEEE Conference on Information Visualization 2000, Londres, July 2000.

[Russo00b] Russo Dos Santos C., Abel P., Gros P., Loisel D., Trichaud N. & Paris J.P. *Experiments In Information Visualisation using 3D Metaphoric worlds*, KMN2000 NIST Knowledge media networking, Washington, June 2000.

[Russo00c] Russo Dos Santos C., Gros P., Abel P., Loisel D., Trichaud N. & Paris. J.P. *Metaphor Aware 3D Navigation*. IEEE symposium on Information Visualization 2000, Salt Lake City, October 2000.

[Satalich95] Satalich G.A., *Navigation and Wayfinding in Virtual Reality: Finding the proper tools and cues to enhance navigation awareness.* Master Thesis of Science Engineering, University of Washington, 1995.

[Schaffer99] Schaffer E.; Reed D. A.; Whitmore S. & Schaeffer B., *Virtue: Performance Visualization of Parallel and Distributed Applications.* Computer, December 1999, pp. 44-51

[Sindre93] G. Sindre, B. Gulla, H. G. Jokstad, *Onion Graphs: Aesthetics and Layout*, Proceedings of IEEE/CS Symposium on Visual Languages (VL'93), IEEE CS Press, pp. 287-291, 1993.

[Smith87] Smith R.B. *Experiences With the Alternate Reality Kit--An Example of the Tension Between Literalism and Magic*. Proc. ACM SIGCHI 1987, April 1987.

[Spence00] *Information Visualization*, ACM Press, 2000.

[Sprenger98] T. C. Sprenger, M. Gross, D. Bielser, & T. Strasser, *IVORY -- An Object-Oriented Framework for Physics-Based Information*

Visualization in Java, Proceedings of the IEEE Symposium on Information Visualisation (InfoVis'98), IEEE CS Press, 1998.

[Stallings96] Stallings W., *Snmp, Snmpv2, and Rmon : Practical Network Management*, Addison-Wesley ,1996

[Stoakley95] Stoakley R., Conway M., Pausch R., *Virtual reality on a wim: Interactive Worlds in miniature*, CHI '95 Conference Proceedings, ACM Press, 1995

[Sugiyama89] K. Sugiyama, S. Tagawa & M. Toda, *Methods for Visual Understanding of Hierarchical Systems Structures*, IEEE Transactions on Systems, Man and Cybernetics SMC-11(2), pp. 109-125, 1989.

[Swing98] Swing E., *Flodar : Flow Visualisation of Network Traffic.* IEEE Computer Graphics and Applications, Vol.18, No. 5, September/October 1998

[Tesler92] Tesler J. D. & Strasnick S. L. *FSN: The 3D File System Navigato*r. Silicon Graphics Inc., online program, ftp://sgi.sgi.com/sgi/fsn, 1992

[Thorndyke83] Thorndyke, P.W. & Goldin S.E., *Spatial Learning and Reasoning Skill*, in Spatial Orientation: Theory, Research, and Application, H.L. Pick and L.P. Acredolo (Editors), 1983, Plenum Press, New York, pp. 195-217.

[Tufte83] Tufte E. R., *The Visual Display of Quantitative Information*, Graphics Press, 1983

[Tukey77] Tukey John W., *Exploratory Data Analysis*, Addison-Wesley Publishing Company,1977

[Unicenter] http://www.cai.com/unicenter/

[VENoM] VENoM - Virtual Environment for Network Monitoring http://www.nrl.navy.mil/CCS/people/cubeta/venom/

[VRML] http://www.web3d.org/vrml/vrml.htm

[Waldo98] J. Waldo et al*., Javaspace specification.* Technical report, Sun Microsystems, March 1998.

[Ware00] Ware C., *Information visualization : perception for design*, Morgan Kaufmann Publishers, San Francisco, 2000

[Wood95] Wood A.M., Drew N.S., Beale R., & Hendley R.J., *HyperSpace: Web Browsing with Visualisation*, Third International World-Wide Web Conference Poster Proceedings , 1995, Darmstadt, Germany

[Wurden97] Wurden F.L., *Content is King (If You Can Find It): A New Model for Knowledge Storage and Retrieval,* Proceedings of 13th International IEEE Conference on Engineering, Birmingham, UK, April 7-11, 1997

[Wyckoff98] P. Wyckoff, S. McLaughry, T. Lehman, D. Ford, *T spaces*, IBM Systems Journal, 37(3), pp. 454–474, 1998, http://www.almaden.ibm.com/cs/TSpaces/

[Zhou96] M. X. Zhou & S. K. Feiner, *Data characterization for automatically visualizing heterogeneous information*, IEEE Symposium on Information Visualization 1996, pages 13–20