Un cadre de conception pour la Visualisation d'Information Interactive

Romain Vuillemot

INSTITUT NATIONAL DES SCIENCES APPLIQUEES DE LYON

ECOLE DOCTORALE INFOMATHS

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

T H E S E D E D O C T O R A T

pour obtenir le grade de

Docteur en Informatique

de l'Université de Lyon

Specialité : Informatique

Défendue par

Romain VUILLEMOT

Un cadre de conception pour la Visualisation d'Information Interactive

Encadrants : Béatrice RUMPLER et Jean-Marie PINON

Préparée au laboratoire LIRIS, Lyon

N° d'ordre 2010-ISAL-0125

Soutenue publiquement le 7 décembre 2010 devant le jury composé de :

Jury :

Rapporteurs :	Jean-Daniel FEKETE	-	INRIA (AVIZ), Orsay
	Gaëlle CALVARY	-	LIG, Grenoble
Examinateurs :	Mountaz HASCOËT	-	LIRMM, Université Montpellier II
	Alain MILLE	-	LIRIS, Université Claude Bernard Lyon 1
Encadrants :	Béatrice RUMPLER	-	LIRIS, INSA Lyon
	Jean-Marie PINON	-	LIRIS, INSA Lyon
Invitée :	Catherine PLAISANT	-	HCIL, Université du Maryland, USA

Remerciements

"Brick walls are there for a reason.
They let us prove how badly we want things."

Randy Pausch

Je tiens à remercier les membres du jury d'avoir accepté de s'intéresser à ces travaux, ainsi que pour les retours critiques qu'ils ont effectués. Jean-Daniel Fekete dont j'ai beaucoup suivi les travaux dès le début de ma thèse, et dont j'ai contemplé les treemaps millionvis [FP02a] au HCIL. Gaëlle Calvary que j'ai rencontré lors des rencontres doctorales à IHM 2009 à Grenoble, ainsi que Mountaz Hascoët que j'ai croisé à cette même conférence. Catherine Plaisant qui a rendu mes six mois au HCIL aussi agréables dans le laboratoire qu'en dehors autour d'une bière ou d'un crabe. Ces six mois peuvent être bien résumés en *work hard, play hard*. Alain Mille avec qui j'ai pu avoir de nombreuses discussions tout au long de ma thèse.

I would like to thank all the people I got acquainted and worked with, during my stay at HCIL from February until August 2008. First of all Catherine Plaisant again for her invitation, availability and help with absolutely everything. Kiki and David T. Wang for their warm welcome and assistance during my first days. Tanya Clement and Amit Kumar for introducing me to the MONK project and their cooperation. Alex Aris for our runs on campus and the Google Pizza (that we actually never got..). Adam Perer as my cubicle neighbor and for our numerous discussions. Allison Druin for hosting me in her lab. Finally Ben Shneiderman and Ben Bederson for the discussions -as interesting as challenging !- I had the privilege of having with them.

Résumé de la thèse

Un cadre de conception pour la Visualisation d'Information Interactive

Résumé :

Une quantité de plus en plus grandissante d'information est produite à chaque instant sur Internet. Il s'agit aussi bien de pages web, que d'images, de vidéos, ou même de connexions sociales, et qui sont mises à la disposition du public, qui peut dès lors commenter et évaluer ces informations. Ce nouveau paradigme (dit web 2.0) produit donc des nouvelles données qui sont cependant difficiles à manipuler, analyser et comprendre, de part leur taille et le nombre de dimensions qu'elles comportent, mais permettent de nouvelles formes d'évaluation et de partage d'information.

Afin de mieux comprendre et manipuler ces données, de nombreuses applications visuelles interactives sont développées chaque année, aussi bien par la communauté de recherche en visualisation d'information, que par des entreprises sous forme de logiciels. Ces applications se basent sur un modèle de référence de traitement de données, qui encode les données abstraites propres à la visualisation d'information en variables graphiques afin de produire des images. Ces images peuvent ensuite être rendues interactives si elles sont inclues et régénérées au moyen de widgets interactifs et d'interfaces graphique. L'utilisateur peut alors au fil de requêtes et de visualisation de résultats résoudre des problèmes complexes, mais souvent connus à l'avance et très spécifiques à une tâche ou un jeu de données. Par exemple si une nouvelle tâche apparaît ou si le modèle de données change, une nouvelle itération de développement et déploiement logiciel sera nécessaire.

Récemment, de nombreux sites internet proposent des visualisations sous forme de service (ex : IBM ManyEyes, Google Visualization, etc.), permettant aux utilisateurs de s'affranchir de tout déploiement logiciel, car elles sont consultables à partir d'un simple navigateur web. Ces sites permettent aux utilisateurs de créer, annoter et partager des visualisations d'information et déclencher un processus d'analyse sociale de données (autrement dit une découverte collaborative de connaissances). Ce phénomène de visualisation façon web 2.0 prend une place grandissante, et la plupart des logiciels classiques de visualisation sont désormais disponibles sous forme de "services" (à savoir des sites internet). Mais au delà de ces opportunités sociales offertes par la mise en réseau, de nombreux facteurs techniques (coupure de connexion, variation du débit, etc.) et conceptuels (traçage de l'activité utilisateur, statistiques d'usages, etc.) sont méconnus ou peu exploités.

Dans cette thèse nous nous sommes intéressés aux conséquences de la mise en réseau des visualisations d'information. La première étape a été d'explorer l'espace de design des visualisations d'information, ainsi que les interactions possibles et les stratégies d'exploration visuelles également, afin de proposer un cadre techniques de mise en réseau de ces visualisations. Ces alternatives de design ont été inclues dans une architecture globale, VizOD (*VIsualiZation-On-Demand*), permettant de faciliter leur développement et leur évaluation. VizOD implémente le modèle de référence de visualisation d'information sous forme de services web qui, une fois composés au moyen d'une interface de programmation visuelle que nous avons développée, génère des visualisations de données accessible via une simple URL. Ces services sont ensuite coordonnés par un graphe de scène et rendues visibles et interactives chez le client grâce à son profil applicatif local, tel qu'un navigateur web ou une application Java. Des métriques d'analyse technique ont été identifiées, afin de permettre un traçage et une évaluation systématique de ces visualisations côté serveur et côté client.

VizOD a été validé techniquement par le développement complet et déployé de quatre applications visuelles interactives : navigation multi-échelle dans de grands graphes type noeuds/liens, exploration d'une base d'images par similarité, analyse et caractérisation de texte par nuages de mots clés, et exploration multi-facette de bases de données. La diversité de ces interfaces a permis de montrer, au de là de la faisabilité technique, la flexibilité et l'adaptabilité de VizOD en cas de nouvelles itérations logicielles, ainsi que de variation de la qualité de la connexion réseau. Nous avons conclu nos travaux par une discussion des principaux axes d'amélioration de VizOD, mais aussi par les différentes possibilités d'optimisations et de prototypage qu'offre cette architecture dans le cadre du développement de nouvelles interfaces. Enfin parmi les principales pistes futures de recherche que nous avons identifié, un des verrous scientifiques que nous considérons comme majeur est celui de la visualisation de données continues.

Mots-clés : Visualisation d'information, Interactions Homme/Machine, Services Web.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Les travaux présentés dans ce manuscrit se positionnent dans le domaine de la *visualisation d'information interactive*. Notre objectif est de favoriser la création et le partage de représentations visuelles d'information. Pour cela, nous proposons un cadre de développement spécifique aux interfaces visuelles interactives dont une partie des traitements est exécutée à distance, accessible au moyen d'un réseau type Internet.

Ces travaux ont été réalisés au laboratoire LIRIS [1] à Lyon de 2006 à 2010, financés par une bourse ministérielle, et initiés dans le cadre du projet d'ACI APMD [2] (2004 - 2007). Une partie de ces travaux a été réalisée au HCIL [3] à l'Université du Maryland, financée par une bourse de la région Rhône-Alpes. Enfin, ces travaux se sont terminés grâce à un support d'ATER au département Informatique de l'INSA de Lyon. D'un point de vue scientifique, ces travaux constituent la suite à un Master Recherche [Vui06] soutenu en juin 2006 à l'INSA de Lyon.

1.1 Contexte de nos travaux

Une quantité croissante d'information produite et disponible

Le monde ouvert et mis en réseau dans lequel nous vivons produit à chaque instant un nombre croissant de données. D'un point de vue quantitatif, 988 exabytes de données seront générées en 2010 [4], soit plus de 18 millions de fois la quantité d'information jamais écrite dans les livres. Nous-mêmes, en tant qu'utilisateurs de systèmes informatiques, en produisons un grand nombre. A la fois de manière *explicite* via la production d'information telle que la prise ou annotation d'une photo, la communication par email, l'échange via des sites participatifs (forums, réseaux sociaux) ou la communication en ligne (Twitter [Twi10] génère 8 Terabits de données par jour [5], soit 80Mo par seconde, chaque message étant de 140 caractères au maximum). Mais un grand nombre de données sont générées de manière *implicite*, sans s'en rendre compte via les données enregistrées automatiquement par des systèmes (position GPS dans un téléphone portable, identification dans un système,

1. http://liris.cnrs.fr/
2. http://apmd.prism.uvsq.fr/
3. http://www.cs.umd.edu/hcil/
4. John Gantz (March, 2008). *"An Updated Forecast of Worldwide Information Growth Through 2011"* http://www.emc.com/digital_universe/
5. http://techcrunch.com/2010/09/17/twitter-seeing-6-billion-api-calls-per-day-70k-per-second/

consultation d'une page web, etc.), car chaque accès à une ressource connectée à un réseau génère des journaux (*logs*) qui sont enregistrés et stockés côté serveur.

Par exemple, le code 1 regroupe une partie des informations qui sont envoyées à tout serveur web à chaque fois qu'une page web est consultée. Le serveur web en question est le site `http://slashdot.org/` (`Host : slashdot.org`) dont l'utilisateur souhaite obtenir la page située à la racine du site (`GET /`). Ce serveur web connaitra l'adresse IP de l'utilisateur (et qui peut ainsi être géolocalisé), l'heure de consultation, le type de navigateur web utilisé (`User-Agent :`), ainsi que le type de fichier accepté en retour (`Accept :`). Enfin le champs de référant (`Referer :`) indique quel était le site précédent (par exemple l'adresse du site à partir duquel l'utilisateur est passé pour arriver sur `slashdot.org`) et un champ de session (`Cookie :`) indique un pointeur vers une éventuelle session en cours.

Code 1 Exemple d'en-tête HTTP communiqué à un serveur pour obtenir une page web.

```
GET / HTTP/1.1
Host: slashdot.org
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-GB; rv:1.9.2.10)
Gecko/20100914 Firefox/3.6.10 GTB7.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: http://www.google.fr/url?sa=t&source=web&q=slashdot
Cookie: __utma=9273847.958059206.1283456015.1287082366.1287086035.7; [..]
```

Ces données d'accès à une ressource distante sont systématiquement communiquées à un serveur[6]. Elles permettent à un site d'identifier la popularité de ses ressources (quelle page web est la plus visitée, la plus commentée, etc.), et de proposer des fonctionnalités de recommandation d'information à partir de navigations ou sessions précédentes. Ces données implicites collectées, ainsi que les données explicites, sont souvent rendues public et disponibles aux programmeurs sous forme d'APIs (*Application Programming Interfaces*). Google [Goo10a], Facebook [Fac10], Twitter [Twi10] ou Flickr [Fli10] possèdent tous une ou plusieurs APIs, dont le but est de permettre à des tiers de proposer de nouveaux services interopérables avec ces sites.

Créer des vues pour comprendre les données

Ces données, même produites par les utilisateurs eux-mêmes, sont difficilement compréhensibles à leur état brut. C'est à dire sans mise en forme graphique, précédée par une phase

6. Les IP peuvent cependant être détournées avec l'utilisation d'un proxy ou dans le cas d'un sous-réseau qui possède un routeur NAT (*Network Address Translation*) qui traduit des adresses locales à un sous-réseau, en une adresse unique visible de l'extérieure. Un serveur pourra cependant toujours identifier et suivre un utilisateur de manière précise au moyen du champ de session (`Cookie :`) de l'en-tête.

de traitements tels que le filtrage, la sélection ou l'agrégation, afin d'exploiter au mieux les capacités visuelles et cognitives de l'humain. La création de *représentations visuelles* ou *vues* sur les données est donc nécessaire. La démarche de conception n'est cependant pas automatique, car elle dépend de la taille et de la nature des données, de la tâche à réaliser et du niveau d'expérience de l'utilisateur auquel seront confrontées les vues.

La création de ces vues relève d'un domaine de recherche dit de la visualisation d'information, du fait que les données sont abstraites, à savoir sans support de représentation immédiat. Chaque année, aux conférences de ce domaines telles que IEEE Infovis ou ACM CHI, sont présentés de nouveaux systèmes pour créer ces vues et interagir avec. Ces systèmes sont souvent spécifiques aux trois critères évoqués précédemment (les données, la tâche et l'utilisateur) et dont la généralisation n'est pas immédiate. Les systèmes peuvent être des *logiciels* commerciaux tels que Tableau Software [Sof10], Spotfire [Spo10] ou IBM ILOG Visualization [Vis10], ou issus de projets de recherche comme GeoVISTA [TG02] ou Jigsaw [SGL08]. Il existe également des *bibliothèques* et *boîtes à outils*, produites par la communauté et qui regroupent les bonnes pratiques de design ou de structure de données, au fil d'années d'expériences, ainsi que de nombreux exemples : The Infovis Toolkit [Fek04], Prefuse [HCL05], VTK [VTK10], Piccolo2D [BGM04], Flare [Fla10b], Processing [Pro10a]. Ces derniers outils sont par contre destinés à des utilisateurs experts en programmation, et nécessite,t un cycle ou plusieurs cycles de développement pour produire une interface utilisable par un utilisateur.

Un exemple de vues qu'il est possible de créer, dans le cas de réseaux complexes, est disponible sur le site `visualcomplexity.com` (figure 1.1).

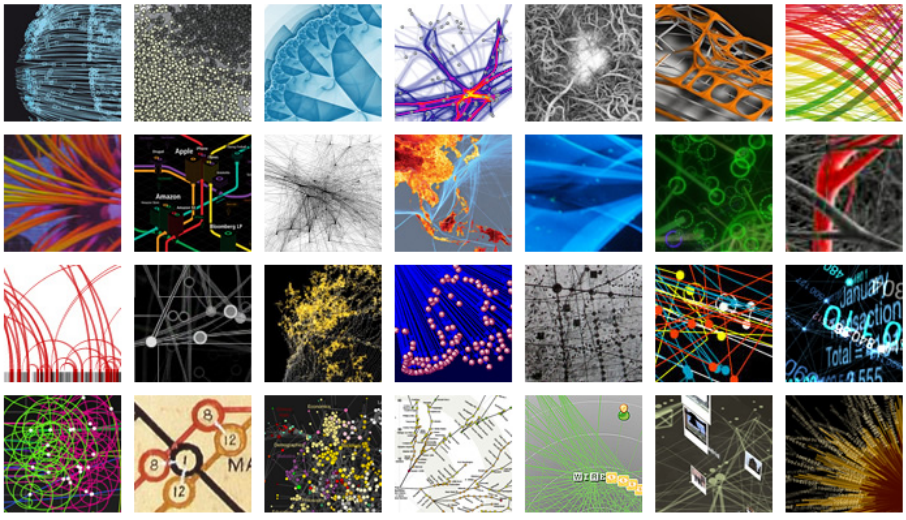


Figure 1.1 – Le site web `visualcomplexity.com` regroupe une collection classée de représentations visuelles de réseaux complexes.

Implication croissante de l'utilisateur dans la création et le partage des vues

L'utilisateur a récemment été mis au centre d'une nouvelle dimension collaborative de visualisation de données, au moyen de site web dits "2.0" où ils peuvent réaliser en quelques clics des visualisations, ainsi qu'en modifier ou en annoter d'autres. Les outils classiques logiciels (autrefois disponibles uniquement en version locale à une machine) sont devenus accessibles sous forme de sites web, application JAVA ou services web. TABLEAU PUBLIC [Pub10], SAP BUSINESS OBJECTS BI [SAP10], TIBCO SPOTFIRE ANALYTICS SERVER [TIB10] en sont les exemples, connectés à des bases de données distantes qui hébergent les données et les services qui réalisent des calculs tels que le filtrage ou la réduction de dimensions des données. Dès lors un simple navigateur web permet d'accéder à toutes ces fonctionnalités qui auparavant n'étaient disponibles que via l'installation et la mise à jour de logiciels.

Récemment sont apparus des sites web et services Web nouveaux qui permettent aussi la mise en réseau de ces vues tels que MANYEYES [Man09], OPENVIZ [Ope10c], SWIVEL [Swi09]. MANYEYES (figure 1.2) propose par exemple aux utilisateurs de transférer leur jeu de données sur le serveur d'IBM et d'ensuite le visualiser avec différentes représentations visuelles. La vue ainsi générée en quelques clics pourra être communiquée via une URL, inclue dans un blog et commentée. La facilité d'utilisation, les faibles coûts de conception, de transport et de consultation, font que tout individu a potentiellement à sa disposition de nombreux exemples de visualisations, ainsi qu'un grand pouvoir de dissémination.

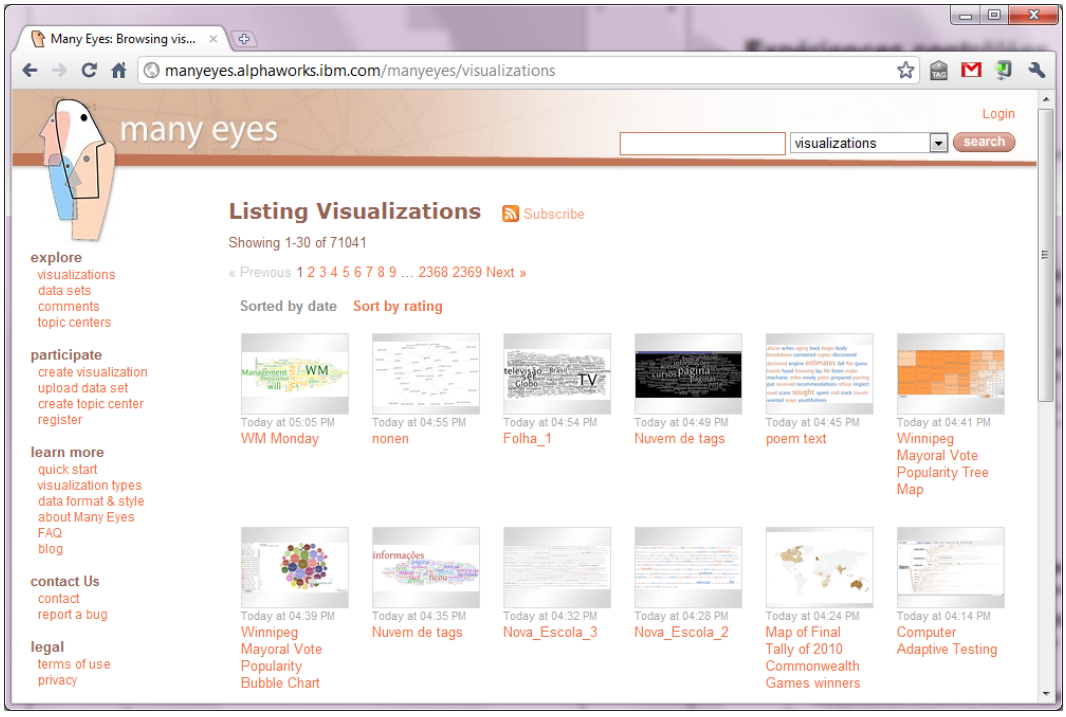


FIGURE 1.2 – Capture d'écran du site web MANY EYES et exemple de visualisations disponibles [Man09].

A noter qu'historiquement, la transmission d'informations graphiques sur un réseau a

déjà fait l'objet de nombreux travaux. Le système `X-Windows`[7] fût le premiers système de fenêtrage construit sur une architecture client/serveur. Le serveur gère le système de fenêtrage, et le client communique des requêtes (en fonction de l'interaction ou de l'état du système client) et reçoit un évènement. L'avantage de ce système est d'être multiplateforme et indépendant du logiciel exécuté, au détriment de performances. Des outils récents comme RealVNC [Rea10] permettent la connexion à des machines distances, de manière sécurisé avec des mécanismes de compression de données, mais restreint à des mécanismes client/serveurs propriétaires et limités en fonction de la qualité de la connexion réseau ou des pare-feux.

1.2 Problématique

Notre objectif est de favoriser la création et le partage de représentations visuelles d'information. Pour rappel, le contexte que nous venons de décrire et dans lequel s'inscrivent nos travaux est triple :

1. **De nombreuses représentations visuelles existent**, sous forme d'images, de logiciels, de bibliothèques ou de services web.
2. **Les communications réseau (type Internet)** permettent le suivi de l'activité d'utilisateur, et l'interconnexion de sources de données (au moyen d'APIs).
3. **Un cadre de travail asynchrone et d'évaluation sociale** est offert aux utilisateurs au moyen d'Internet, qui deviennent ainsi producteurs de contenu (visualisations, commentaires, annotations, etc.).

A partir de ce contexte, nous identifions trois problématiques.

La première problématique que nous identifions est la limite de l'*interopérabilité* des visualisations. L'*interopérabilité* permet à un utilisateur de travailler dans le contexte d'une application, et ensuite d'utiliser ses résultats dans le contexte d'une autre application tout en se concentrant au maximum sur sa tâche [SPCJ09]. Il s'agit donc de pouvoir connecter plusieurs applications, au moyen de formats de fichiers similaires ou convertibles par exemple. A titre d'illustration, le site ManyEyes permet de créer des visualisations en Java sans permettre une connexion directe avec Microsoft Excel [Mic10a] pour la source de données, ni Prefuse [HCL05] pour inclure la vue créée dans un environnement interactif. Le manque d'interropérabilité impose donc à l'utilisateur de réaliser des tâches répétitives qui peuvent soit nécessiter des compétences techniques (comme la création de scripts de conversion de formats de fichiers) soit tout simplement décourager le décourager ou lui faire perdre sa concentration.

Cette problématique est bien connue des communautés de programmeurs et de chercheurs : lors d'une discussion sur la liste de développement de Graphviz au sujet de créer un service web interropérable pour la génération de graphe, la question[8] *"Would somebody be able to set up a Graphviz server with the following features : [..] Users save a dot-file in the public_html part of their own server : [..] they call the Graphviz server including their dot-file as parameter [..]"* a eu pour réponse[9] : *"Countless hours have been spent there on developing theory*

7. http ://en.wikipedia.org/wiki/X_Window_System
8. https://mailman.research.att.com/pipermail/graphviz-interest/2010q1/006817.html
9. https://mailman.research.att.com/pipermail/graphviz-interest/2010q1/006821.html

and research tools, most of them unfortunately still only used in the labs. The bottleneck also there is "the last mile".". De nombreux systèmes sont développés sans malheureusement être rendus interopérables avec d'autres.

La deuxième problématique que nous identifions est la limite de *flexibilité*. La *flexibilité* est la capacité d'un programme à fonctionner même si une nouvelle tâche, type d'interface, d'interaction ou de visualisation apparait (ou disparait). La plupart des bibliothèques que nous avons énuméré dans la section précédente sont flexibles car elles ont une architecture modulaire qui permet une expressivité importante. Un programmeur pourra rapidement inclure un nouvel algorithme de traitement ou de visualisation de données, via la réécriture ou surcharge d'une fonction. De même les logiciels possèdent souvent un système de gestion de modules qui permettent leur extension dans redéploiement logiciel complet.

Dans le contexte de la mise en réseau, la problématique de flexibilité s'étant à la prise en compte des nouvelles incertitudes que rajoute la connexion réseau au cycle d'exécution d'un programme. L'accès à des ressources partagées rend l'usage d'applications connectées imprévisibles, à cause de coupures de l'accès réseau, variation du délai ou du débit, non-synchronisation de données. Par exemple concernant le *délai* du réseau, un simple *ping* sur un serveur Google indique qu'il varie de 7*ms* en étant connecté à Renater (*Réseau national de télécommunications pour la technologie, l'enseignement et la recherche*), en passant par 32*ms* avec un abonnement ADSL grand public et jusqu'à 122*ms* en connexion Wifi vers un hot spot distant. Même ces délais connus, d'autres paramètres "aléatoires" sont à prendre en compte (illustrés figure 1.3) tels que le *temps d'exécution* du programme qui va générer la ressource (script PHP, requête SQL, etc.) et le paramètre de *gigue* (*jitter*) de fluctuation du délai, qui rendent ainsi l'accès à une ressource distante difficilement prévisible.

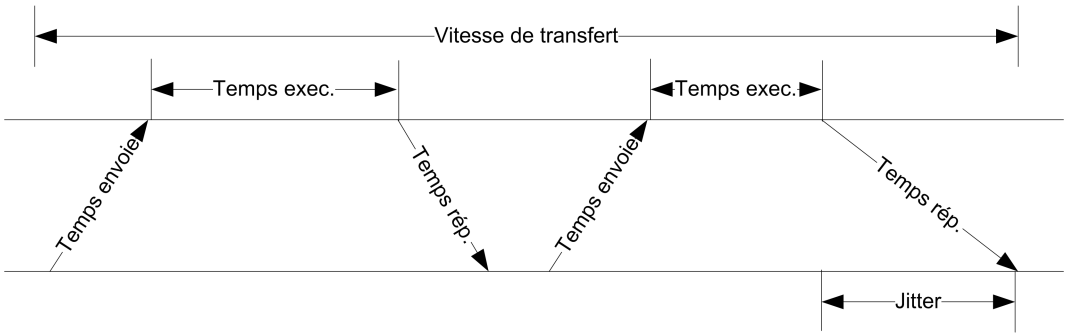


Figure 1.3 – Illustration de différentes métriques réseaux.

Il existe actuellement des approches hybrides permettant aux interfaces visuelles qui accèdent à des ressources distantes de combler le manque de données et garantir la réactivité de l'interface. Google Map [Goo10d] propose par exemple d'afficher uniquement un damier représentant le sol et un ciel bleu afin de représenter une rue virtuelle manquante (image de gauche figure 1.4). Le système Microsoft Photosynth [Mic10b] propose un index des points d'intérêt d'une scène (image de droite figure 1.4) même si certaines données sont soit tout simplement non disponibles (comme certains angles de vues de photos de la scène), soit pas encore téléchargées dans l'application client. Ces deux approches reposent sur les

capacités humaines à compléter une information manquante.

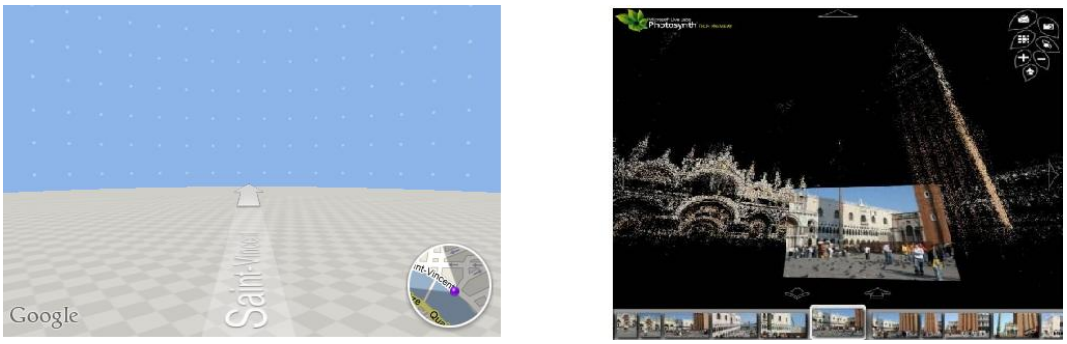


FIGURE 1.4 – Stratégies des applications GOOGLE MAP [Goo10d] et MICROSOFT PHOTOSYNTH [Mic10b] pour afficher des données manquantes.

Enfin une troisième problématique que nous identifions est le manque d'*adaptabilité*. L'adaptabilité d'une interface est la possibilité de personnalisation d'un système de manière explicite par un utilisateur, par exemple au moyen d'un menu de configuration. Cette personnalisation peut être aussi bien en fonction de préférences de l'utilisateur, qu'en fonction de contraintes techniques. Les préférences utilisateurs peuvent être liées à certaines déficiences visuelles de celui-ci (daltonisme, âge avancé, etc.) et donc nécessiter l'usage d'un fort contraste de couleurs ou de certaines couleurs seulement. MANYEYES propose une visualisation de textes en nuage de mots clés, mais ne permet pas de changer la couleur (autre que bleu et rouge), ni de faire varier la taille des caractères, ni de changer la couleur de fond. L'adaptabilité permet aussi l'évaluation d'interfaces, en laissant la possibilité aux évaluateurs de faire varier les alternatives de design et tester l'impact sur les performances de l'utilisateur (telles que la rapidité pour réaliser une tâche ou le taux d'erreur). Même si la plupart des logiciels actuels intègrent une visualisation [Pla04] (par exemple une ligne temporelle), très peu d'entre elles sont adaptables et leur évaluation est donc complexe.

1.3 Objectifs

Même si le contexte actuel permet déjà la création et le partage de représentations visuelles d'information, leur manque d'*interopérabilité*, de *flexibilité* et d'*adaptabilité* limite leur conception et leur évaluation. Le but de nos travaux est d'y remédier, pour cela nous avons identifié trois objectifs :

- **Identifier les dimensions de l'espace de design des représentations visuelles interactives, ainsi que leurs dépendances.** Cela permettra de mieux comprendre les étapes qui permettent la création de visualisations, et pour chacune d'elles les alternatives possibles.
- **Proposer un cadre complet de conception** afin d'appréhender le manque d'interopérabilité, de flexibilité et d'adaptabilité des visualisations actuelles. **Ainsi que définir une architecture opérationnelle** permettant leur conception et leur évaluation.
- **Développer des applications à partir de cette architecture** dans le but de la tester et de

la valider dans des situations de développement proches de la réalité.

1.4 Structure de ce mémoire

Ce manuscrit se structure en trois grandes parties :

La partie I – "État de l'art" : dans cette partie nous allons tenter d'identifier les dimensions et les dépendances de l'espace de design de la visualisation d'information interactive. Pour cela nous nous intéressons à l'espace général de design la visualisation d'information (chapitre 2), ainsi qu'aux interactions pour la visualisation (chapitre 3) et enfin aux principales techniques d'exploration visuelle (chapitre 4).

La partie II – "Architecture VizOD et applications" : dans cette partie nous proposons notre cadre de conception et architecture VizOD (chapitre 5), à partir de l'espace de design identifié dans la partie I. Nous décrivons ensuite quatre applications développées à partir de cette architecture (chapitre 6), et discutons de cette expérience de développement et des limites de notre architecture (chapitre 7).

La partie III – "Conclusion et perspectives générales" nos travaux se terminent par une conclusion ainsi que par les perspectives de recherche que nous souhaitons donner à ces travaux (chapitre 8).

Première partie

État de l'art

Chapitre 2

Espace de design de la visualisation d'information

Sommaire

2.1 Introduction à la visualisation d'information

La visualisation d'information est un processus permettant de générer des mises en forme graphiques de données, afin d'aider l'humain à mieux exploiter son système visuel et cognitif. Le résultat de la mise en forme peut être accessible sur divers supports, aussi bien papier que sur écran, et peut être interactif (exemple de visualisations figure 1.1). L'objectif de ce processus est très souvent de permettre la compréhension de grandes quantités de données qu'il n'aurait pas été possible de réaliser uniquement muni d'un simple papier et d'un crayon, sans mise en forme avancée des données brutes.

Les images résultant de ce processus sont variées, belles et inspirantes, mais la visualisation n'est cependant qu'un moyen, et non une finalité. Ben Shneiderman le résume bien en ces termes : *"The purpose of visualization is insight, not pictures"* [1] pour mettre en avant l'appui à l'humain qu'elle constitue, lors d'un processus de réflexion qui dépasse la simple perception des données.

Dans ce chapitre nous nous intéressons aux dimensions de l'espace de design des représentations visuelles d'information. Il s'agit donc d'identifier les différentes étapes permettant la création de ces visualisations, et pour chaque étape énumérer les différentes alternatives possibles. Le but de ce chapitre n'est pas d'être exhaustif pour chacune des dimensions de l'espace de design, mais d'identifier les étapes principales et dépendances entre ces dimensions.

Le domaine de la visualisation héritant d'autres domaines, nous allons tout d'abord étudier les liens communs avec d'autres domaines historiquement fondateurs (section 2.2). Ensuite nous verrons les deux principales dimensions de conception des représentations visuelles : les étapes successives de traitement de données qui mènent à une visualisation (section 2.3), et l'association entre données et variables graphiques (section 2.4). Dans le chapitre suivant (Chapitre 3) nous verrons comment rendre interactives ces visualisations en les associant à des composants graphiques interactifs, et ensuite (Chapitre 4) comment réaliser un parcours cohérent entre les différentes visualisations pour réaliser des tâches exploratoires.

Afin de lever toute ambiguïté sur le terme de "visualisation d'information" nous allons tout d'abord énumérer différentes définitions (section 2.1.1) et donner la notre (section 2.1.2).

2.1.1 Définitions existantes de la visualisation d'information

De nombreuses définitions de la visualisation ont été données ces dernières années. A l'heure où est rédigé ce manuscrit, le WIKI INFOVIS [2] n'en recense pas moins de dix-sept. Ces définitions sont rarement exprimées de manière absolue et universelle, car contextualisées dans des articles ou des ouvrages, avec un périmètre d'application précis. Nous allons essayer dans cette section d'en détailler une liste représentant la diversité.

1. Qui paraphrase Hamming *"The purpose of computing is insight, not numbers"*
2. Site collaboratif modéré mis en place par la communauté infovis http://www.infovis-wiki.net/index.php?title=Information_Visualization

La visualisation, considérée dans un cadre général, a déjà fait elle-même l'objet de multiples définitions. [Ber83], propose dans son ouvrage une première démarche d'énumération des différents codes et symboles qu'il est possible d'utiliser pour la communication visuelle. Selon [Ber83] *"la représentation graphique est la transcription, dans le système graphique de signes, d'une pensée, d'une information connue par l'intermédiaire d'un système de signes quelconques"*.

[McC88] définie la visualisation d'information comme *"transformation of the symbolic into the geometric"* et donc un périmètre de données plus précis, à savoir des données symboliques qui n'ont donc pas de codes immédiats de représentations (à l'opposé par exemple d'une carte géographique qui elle, possède une représentation immédiate). De même cette dualité apparaît également avec [GYG] *"Information visualizations attempt to efficiently map data variables onto visual dimensions in order to create graphic representations"* et chez [Che05] *"In contrast to scientific visualization, information visualization typically deals with nonnumeric, nonspatial, and high-dimensional data"*.

[CMS99] définissent la visualisation d'information comme *"The use of computer-supported, interactive, visual representations of abstract data to amplify cognition"* qui étend le positionnement sur des données abstraites à l'interaction et à des mécanismes de pensée et de réflexion, assistés par l'ordinateur. Tout comme [JS03] qui conclue *"Information visualization is a set of technologies that use visual computing to amplify human cognition with abstract information"*.

D'autres points de vue cherchent au contraire à élargir la définition comme [FD05] *"Information visualization is the broadest term that could be taken to subsume all the developments described here. At this level, almost anything, if sufficiently organized, is information of a sort. Tables, graphs, maps and even text, whether static or dynamic, provide some means to see what lies within, determine the answer to a question, find relations, and perhaps apprehend things which could not be seen so readily in other forms."*. De même [TM04b] proposent une taxonomie de la visualisation dans son intégralité, afin d'éviter justement les divisions historiques entre le domaine de la visualisation d'information et celui de la visualisation scientifique.

2.1.2 Synthèse et notre définition

La très grande variété et richesse des définitions précédentes est signe d'une relative jeunesse du domaine, mais aussi le reflet d'une grande diversité d'acteurs pluridisciplinaires. Dans le cadre des travaux réalisés dans ce manuscrit, nous considérons que les caractéristiques de la visualisation d'information comme les suivantes :

Définition 1 (Visualisation d'information) *La visualisation d'information est la représentation visuelle de données organisées et principalement abstraites, afin d'optimiser les capacités visuelles et cognitives de l'humain.*

Nous nous réfèrerons explicitement au terme de visualisation d'information *interactive* pour indiquer si la visualisation est couplée à un système interactif qui permettra à un utilisateur d'interagir avec.

2.2 Domaines connexes à la visualisation d'information et comparaison

La visualisation d'information est un domaine de Recherche à part entière[3]. C'est un domaine relativement récent au regard de l'informatique, et en particulier de la communauté Interactions Hommes/Machines (IHM). En effet, les dispositifs de présentation graphique (comme les écrans par exemple) n'ont fait de progrès spectaculaires que depuis la fin des années 80, alors que les premiers systèmes interactifs remontent eux au milieu du 20ème siècle. Ceux-ci possédaient certes des instruments de visualisations (boutons lumineux, etc.) mais ils étaient plutôt destinés à représenter l'état du système, que le résultat d'une opération complexe (le résultat pouvait par contre être imprimé sur des cartes perforées).

Cependant, comme de nombreux domaines dits "récents", la visualisation d'information n'est pas un domaine apparu *ex nihilo* et peut être positionné dans un contexte scientifique déjà mature et développé. Ainsi en préambule de l'exploration de l'espace de design propre à la visualisation d'information, nous allons nous intéresser aux principaux domaines qui y ont contribué d'un point de vue technique et conceptuel : l'*informatique graphique* (section 2.2.1), les *systèmes d'information géographique* (section 2.2.3) et la *visualisation scientifique* (section 2.2.2).

2.2.1 Informatique graphique

Le domaine de l'informatique graphique est très vaste, il regroupe aussi bien les composants matériels que logiciels qui sont destinés à produire des images au moyen d'un système informatique.

Couches matérielles et logicielles

Dans le domaine de l'informatique graphique, le matériel et le logiciel possèdent entre eux des interactions et des dépendances très étroites. Les techniques de traçage de dessins sur écran matriciel (domaine de la géométrie discrète) et leur programmation dépendent par exemple de la capacité technique de la carte graphique, du taux de rafraichissement et de la résolution de l'écran. Ces registres sont à prendre en compte en cas de programmation "bas niveau" et de recherche d'optimisations éventuelles, mais peuvent être pris en charge par des bibliothèques type OpenGL [Ope10b] d'abstraction du matériel, qui permettent dès lors une programmation plus "haut niveau".

Du côté de l'architecture matérielle, le processeur graphique GPU (*Graphics Processing Unit*) situé sur les cartes graphique peut se voir sous-traiter des tâches graphiques telles que le rendu 3D, la décompression d'images, le décodage de vidéos, etc. Ce processeur a fait l'objet de beaucoup d'attention et d'amélioration ces dernières années grâce aux acteurs de l'industrie du jeu vidéo et du divertissement, tels que NVDIA [NVD10] ou AMD Graphics (anciennement ATI) [AMD10]. Il n'est pas vraiment possible de comparer le GPU au CPU en termes d'architecture. Par exemple le GPU est optimisé pour les calculs graphiques, et est très performant pour les calculs parallèles, quitte à dépasser le CPU pour ce type de calculs. Ainsi une tendance récente est de permettre les calculs du CPU dits "general purpose", tels

3. Etant donné qu'il produit de connaissances nouvelles, qu'il influence d'autres sciences et qu'une communauté existe et des individus s'en proclament membres.

que les calculs à virgule flottante, sur le GPU. On parle alors de GPGPU (General Purpose GPU). L'usage du GPU peut être étendu au rendu d'une visualisation d'information [ME09], et les calculs de types GPGPU pourront être appliqués par exemple à des tâches classiques de filtrage, de sélection de données et de calcul de la disposition de données d'un graphe.

Calibration du système

L'objectif de la calibration est de permettre à un système informatique de restituer visuellement de manière fidèle les données en fonction des dispositifs d'affichage graphique. En effet, ces dispositifs peuvent varier en termes de luminosité, contraste ou résolution et ainsi modifier les propriétés graphiques des données à communiquer. Ces variations peuvent s'expliquer certes par une mauvaise configuration initiale, mais aussi par l'évolution de la qualité du matériel au fil du temps. La calibration permet également au système de prendre en compte les capacités visuelles propres à chaque humain.

Des méthodologies de calibration très simples existent, comme les mires d'écrans qui consistent à afficher une image sur un système et à demander à l'utilisateur d'ajuster le système pour obtenir un résultat attendu (qui peut être un exemplaire papier de la mire ou une suite de chiffre ou d'éléments visuels à identifier). Par exemple il existe des tests de résolution d'écran (figure 2.1 à gauche) et des tests de colorimétrie [4] (figure 2.1 à droite). Les paramètres de calibration peuvent être enregistrés dans un profil, comme le profil colorimétrique dont le stockage est standardisé [5]. Ces opérations de calibration demandent une intervention humaine, ainsi aucune garantie universelle et constante n'est donnée à un système que les données seront perçues telles que souhaitées.

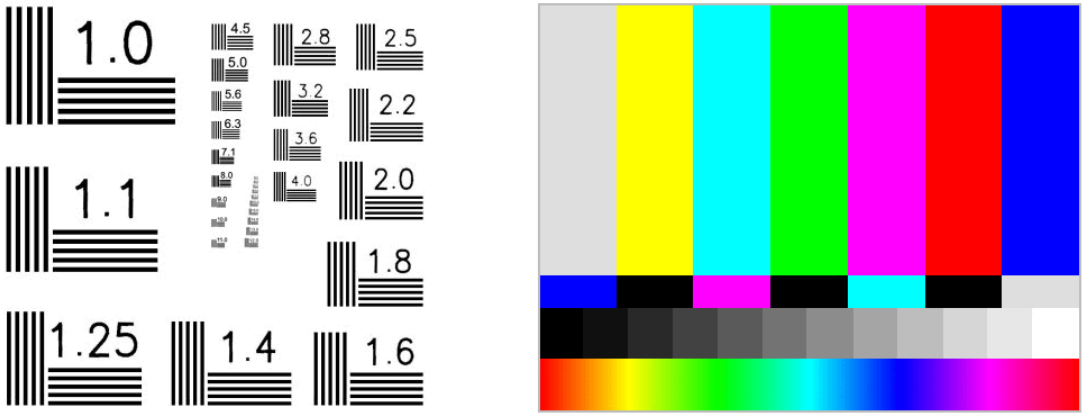


FIGURE 2.1 – Exemple de mires de calibration : à gauche la mire de calibration *"1951 USAF"* pour déterminer la résolution spatiale du système d'affichage, à droite une mire de calibration de couleurs.

Graphe de scène

Un graphe de scène est une structure de données qui permet de stocker de manière logique la représentation d'une scène virtuelle (exemples sur la figure 2.2). Cette scène

4. http://www.displaycalibration.com/
5. http://www.color.org/iccprofile.xalter

peut être spatiale (comme un univers dans un jeu vidéo) ou relationnelle pour indiquer des liens entre des éléments qui n'ont pas de position physique immédiate. Il n'existe pas de norme universelle concernant la structure de données, mais il s'agit très souvent d'un graphe orienté acyclique. L'intérêt d'une telle structure de données est de pouvoir facilement propager des transformations à un groupe d'éléments, tels que des translations, rotations ou mises à l'échelle. Ou par exemple détecter facilement les éléments invisibles (car trop éloigné du point de vue) ou hors du champ de vision.

Le standard X3D [X3D10] (*Extensible 3D*, successeur du VRML *Virtual Reality Modeling Language*) permet d'encoder des scènes selon un graphe de scène avec une syntaxe de type XML. Les systèmes de réalité virtuelle utilisent un graphe de scène afin de positionner les éléments dans l'espace, tel que le système VirChor [Jac] (*Virtual Choreographer*). L'API Java 3D [SRS97] spécifie une structure de données de graphes de scène sous forme de graphe orienté acyclique, où chaque noeud est un objet. Les bibliothèques Piccolo2D [BGM04] et Protovis [BH09] possèdent un graphe de scène interne pour permettre (entre autre) l'interaction et l'animation entre les différentes vues sur les données.

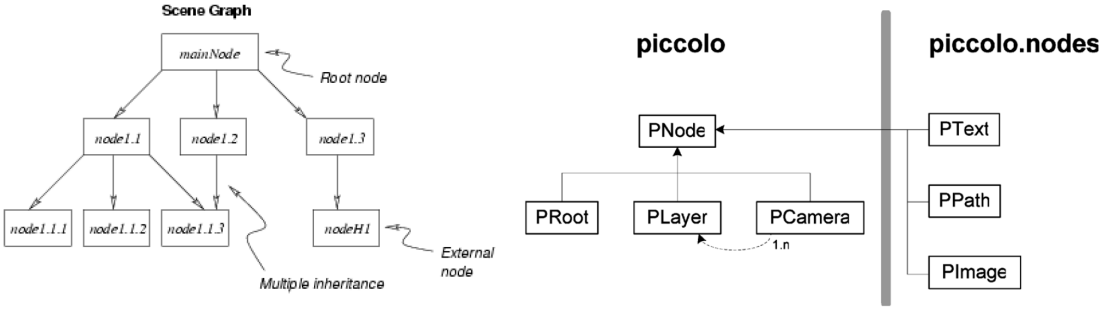


Figure 2.2 – Extraits de graphes de scène issus VirChor [Jac] (à gauche) et Piccolo2D [BGM04] (à droite).

Techniques d'animation

L'animation consiste à assembler une série d'image en une séquence temporelle, afin de représenter un mouvement continue (figure 2.3). [Las87] identifie onze principes pour définir le cadre de l'animation appliquée à l'ordinateur. Ces principes indiquent qu'il faut aussi bien prendre en compte les propriétés physiques des objets en mouvement (par exemple la déformation) tout en exagérant certains trains pour renforcer le mouvement, qu'il faut considérer les limites du matériel de diffusion (nombre de rafraichissement de l'écran limité), et enfin rendre le résultat attrayant pour que l'utilisateur prenne du plaisir à regarder et y fasse attention. Ces techniques d'animations peuvent tout aussi bien être employées pour des mouvements dans un univers similaire au monde réel (figure 2.3), mais également pour animer un monde abstrait tel que la transition entre différentes représentations de graphiques [HR07] ou la navigation dans des dimensions de nuages de points [EDF08] (*scatterplot*).

D'un point de vue technique, l'animation entretient un lien étroit avec le matériel qui doit être de qualité suffisante afin de réaliser une animation fluide. Un indicateur de fluidité d'ani-

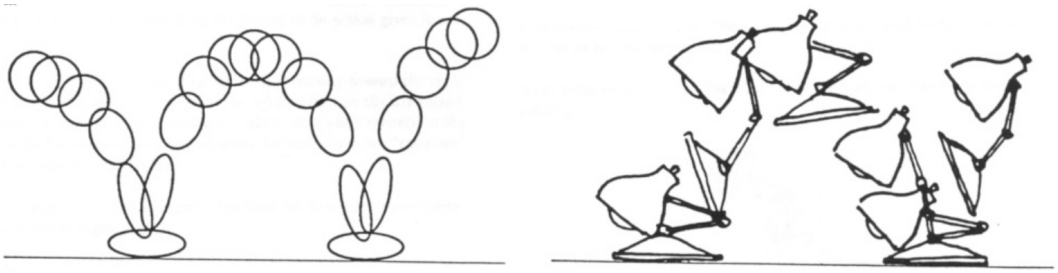


FIGURE 2.3 – Illustration de principes d'animation [Las87].

mation est le nombre d'images par secondes (*framerate*) : au delà de 16 images par seconde, un mouvement sera perçu comme fluide grâce aux propriétés de persistance rétinienne de l'humain. L'affichage d'un grand nombre d'éléments sur un écran peut être la cause d'une baisse de framerate [FP02b]. Afin de garantir celui-ci, il peut être nécessaire de baisser la qualité du rendu (pendant l'animation). Par exemple lors d'un zoom avec la bibliothèque PICCOLO2D [BGM04] (figure 2.4), un phénomène de crénelage apparaît durant l'animation (figure 2.4, cercle du milieu), car les techniques d'anticrénelage (*anti-aliasing*) (comme le lissage des couleurs par voisinage) sont coûteuses en temps de calculs et peuvent faire baisser le *framerate* pendant l'animation qui est elle-même coûteuse en temps de calculs.

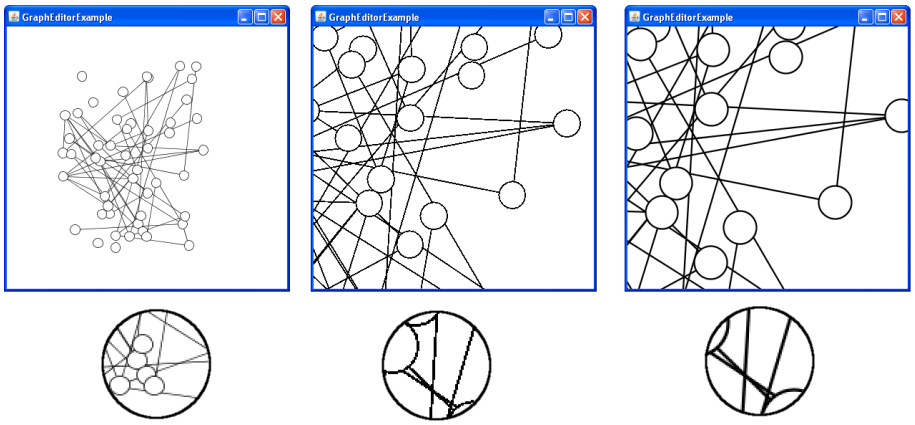


FIGURE 2.4 – Exemple d'animation (zoom de l'image de gauche, à l'image de droite) avec baisse de la qualité d'aspect final du rendu (image du milieu).

2.2.2 Visualisation scientifique

La visualisation scientifique concerne la représentation graphique, l'interaction et la collaboration avec des données scientifiques. Ces données peuvent être aussi bien des observations, que les résultats d'une modélisation, et ont pour propriété de posséder un support de représentation explicite, souvent exclusif (contrairement à la visualisation d'information [FdOL03]). Par exemple la radiographie du poumon d'un patient devra toujours être représenté avec comme support le corps humain. La visualisation scientifique permet la prise de décision, la pédagogie ou l'exploration de données dans des environnements souvent en 3D et distribués (car les quantités de données et leur résolution de capture sont importantes).

Réalité augmentée/diminuée

Les données issues de mesures peuvent être multidimensionnelles et posséder plusieurs représentations qu'il est possible de superposer (figure 2.5)[6]. Ces représentations peuvent être rajoutées en temps réel à des flux de données telles que des caméras et la réalité filmée par celles-ci sera dite *augmentée*. La réalité sera dite *diminuée* permet de réduire la surcharge visuelle de données, et afficher les plans ou une abstraction de structure en effaçant des informations non pertinentes. Ces opérations étant des tâches critiques à réaliser, il est nécessaire d'avoir des résultats fiables de calculs, des précisions et réponses optimaux des systèmes.

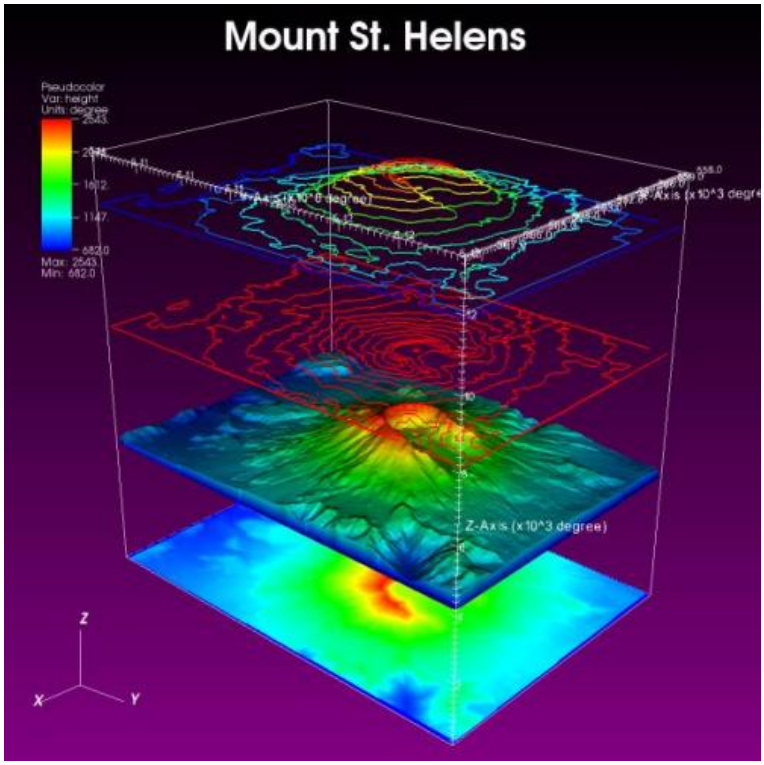


FIGURE 2.5 – Visualisation en couches superposées de visualisation de l'éruption volcanique du Mont St Helens en 1980.

Espaces de travail collaboratif

Les tâches liées à la visualisation scientifique font très souvent appel à la collaboration, et demandent donc de réunir plusieurs experts autour d'un même objet conceptuel de travail. Le mur HYPERWALL-2 (figure 2.6)[7] de la NASA permet de supporter cela. Il mesure 7×3 mètres, il est composé de 128 GPUs et 1 024 CPUs ce qui permet aux chercheurs de collaborer autour de résultats de simulation ou de grandes images. D'autres formes de collaboration (à distance, asynchrones) sont également supportées dans la conception et la publication de traitement de données comme avec la plateforme IRIS EXPLORER [Wal04].

6. Issue de la galerie en ligne https://wci.llnl.gov/codes/visit/gallery.html
7. http://www.nas.nasa.gov/Resources/Visualization/visualization.html

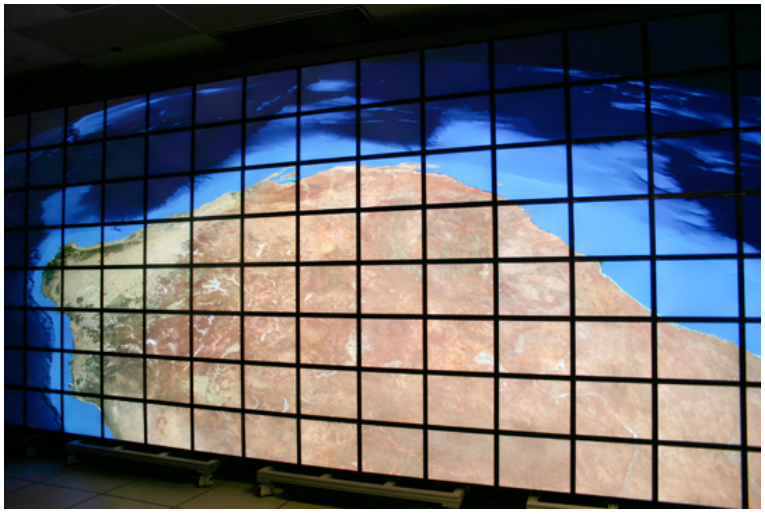


FIGURE 2.6 – Photo du mur *Hyperwall-2* de la NASA.

2.2.3 Systèmes d'information géographique

Les systèmes d'information géographiques (SIG) permettent le stockage, la manipulation et la présentation de données possédant une composante spatiale. Ce très riche domaine puise ses origines bien avant l'arrivée de technologies modernes [FD05]. En effet, l'homme a toujours eu le besoin de se localiser, et donc de se représenter un espace à la fois pour lui-même, mais aussi pour le communiquer aux autres. La carte est le résultat de cette représentation et peut jouer de très nombreux rôles : communication visuelle, support de raisonnement, support de collecte de données, confrontation de données. Le lien avec les techniques de *cartographie* est donc très étroit.

Encodage multiple des données

Les données encodées sur les cartes peuvent posséder plusieurs dimensions au delà de leur composante spatiale, comme la température ou la densité de population (figure 2.7). Le globe terrestre devient ainsi un immense index de données multidimensionnelles dites "géo-codées". A partir de ces données, il sera possible de réaliser des "géo-tâches" de sélection, filtrage ou fouille de données qui doivent parfois être mises à mise à jour en temps réel, par exemple si le résultat de ces données est issu de capteurs, ce qui implique donc un défi de visualisation de données spatio-temporels [WDSC07].

Cartographique abstraite et sociale

La cartographie étant très efficace pour donner une vue d'ensemble d'un grand nombre d'informations, l'extension à des cartes virtuelles de données (telles que générées par auto-organisation [Koh82]) a fait l'objet de nombreuses études et a permis de nouvelles formes d'affichage de résultats de moteurs de recherche. KARTOO [8] [Kar09] est souvent cité comme le premier meta-moteur de recherche cartographique (figure 2.8) par sa présentation de résultats d'une recherche par mot clé sous forme de cartes. Cette représentation s'oppose aux listes qui permettent d'ordonner des résultats mais ne permettent pas de les regrouper

8. Le service a cessé d'exister depuis 2009.

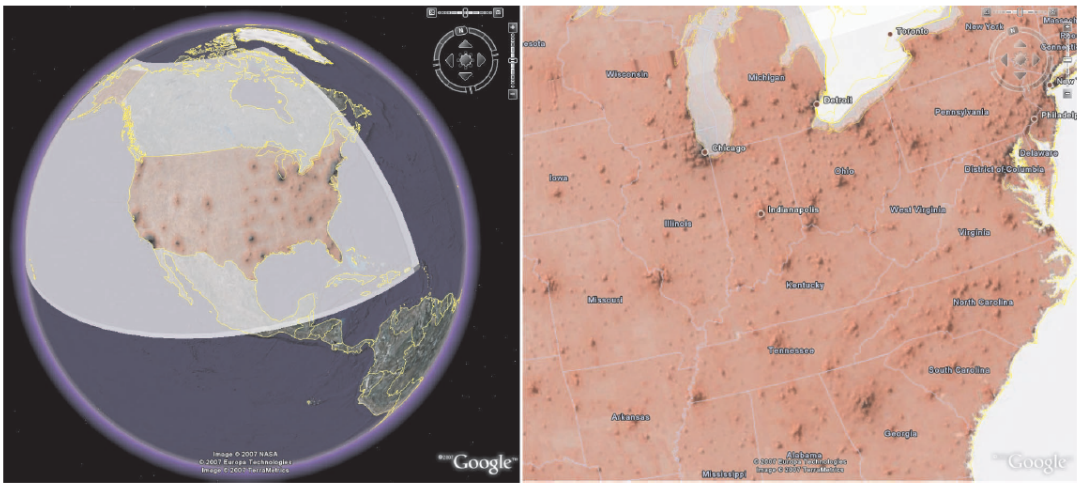


FIGURE 2.7 – Visualisation de la densité de la population aux Etats-Unis, sur un globe terrestre via l'application GOOGLE EARTH [Goo10b].

ou de montrer les liens entre eux.

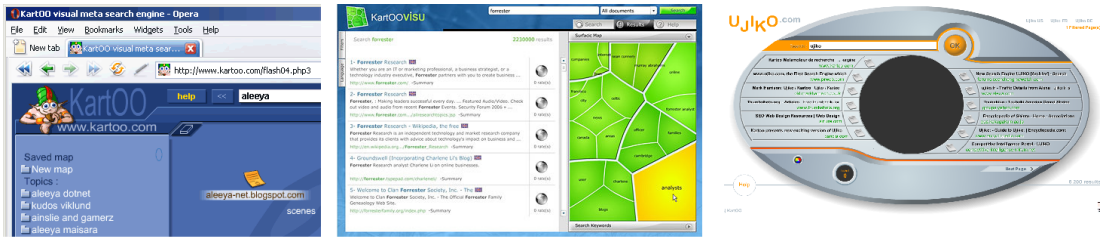


FIGURE 2.8 – Exemple de visualisations cartographique de résultats de meta-moteurs de recherche (de gauche à droite : KARTOO, KARTOOVIS et UJIKO).

La cartographie tend à être de plus en plus un support social, de part la pratique du "geo-tagging". Il s'agit de l'association par l'utilisateur (via un capteur GPS, un hot spot wifi ou une triangulation de bornes 3G) d'évènements sociaux (arrivée, position actuelle, rencontre) ou d'objets (image, texte, vidéo) à un lieu (autrement dit à un couple de données bidimensionnel (x, y)). La carte géographique reste donc un support d'indexation croissant [9], mais également de recherche (figure 2.9) qui permet d'explorer les données autour de soit (si l'on est géo-localisé).

2.2.4 Autres domaines connexes

Les trois domaines que nous venons d'aborder succinctement ont déjà permis d'identifier des similarités étroites en termes des concepts et structures de données fondamentaux avec la visualisation d'information, tels que les techniques d'animation ou la structure de données de graphe de scène. D'autres domaines comme que les jeux vidéos, l'analyse d'image,

9. Selon `http://www.programmableweb.com/` les services GOOGLE MAPS [Goo10d] et FLICKR [htt09] sont les plus utilisés pour composer des sources d'information

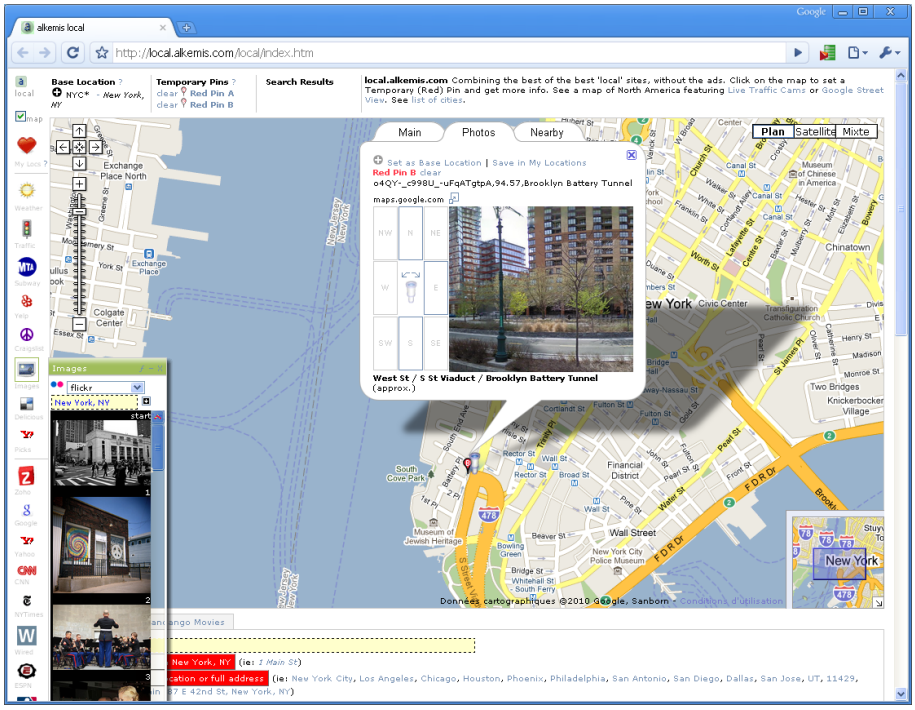


FIGURE 2.9 – Images géo-localisées sur une Google Map et accessibles dans un navigateur web.

le design et même les arts graphiques auraient pu être abordés, et de manière très pertinente, pour expliquer les techniques et méthodes actuelles utilisées pour la visualisation d'information. Ces domaines peuvent également fournir de nouvelles alternatives ou perspectives de dimensions de l'espace de design.

2.3 Modèles de référence de traitement des données

Tout programme informatique, quel que soit son domaine applicatif, réalise une chaîne de traitement de données, plus ou moins complexe. Cette chaîne démarre généralement à partir de la lecture d'un fichier de données ou l'interrogation d'une base de données, suivi par une transformation en structure de données interne (tableau, liste, graphe, etc.) afin d'optimiser leur accès et parcours par les programmes suivant. Cette chaîne aura un nom différent selon le domaine applicatif : en base de données, on parlera de QVT (*Query View Transform*), en intergiciels, de ETL (*Extraction Transformation Load*).

La visualisation d'information ne déroge pas à la règle et possède sa propre chaîne de traitement de données, étendue jusqu'à la visualisation. Cette chaîne (appelée *"modèle de référence"* ou *"pipeline infovis"*, exemple sur la figure 2.10) possède des étapes (section 2.3.1) et opérations (section 2.3.2) qui permettront d'identifier les principales dimensions de l'espace de conception.

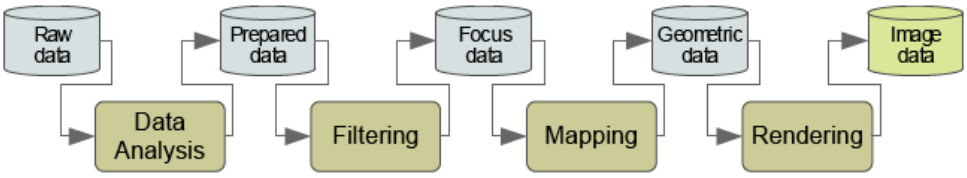


FIGURE 2.10 – Un exemple d'étapes et d'opérations de traitement de données en visualisation d'information [DSB04].

2.3.1 Etapes de traitement des données

Les modèles de traitement de données [HM90, Chi00, HCL05, DSB04] ont une *source* et une *finalité* communes. Concernant la *source*, il s'agit de données "brutes" [10], et la *finalité* un résultat visuel perceptible par l'utilisateur. Ces modèles sont très similaires car ils reposent sur la séparation des données de leur représentation graphique, ce qui est un motif classique de conception de type MVC [HA06, KP88] (*Modèle-Vue-Contrôleur*). Les variantes entre les modèles apparaissent au niveau du rendu des données, car il peut s'agir aussi bien d'une image [DSB04] que d'un rendu interactif [HCL05].

Ces modèles peuvent être décomposés en trois parties complémentaires (figure 2.11) dont chacune possède une terminologie différente :

1. ***Abstraction de données :*** les données dérivées [HM90], l'abstraction analytique [Chi00], les données abstraites [HCL05] données préparées et focus [DSB04].
2. ***Abstraction visuelle :*** un objet visuel abstrait [HM90], une abstraction visuelle [Chi00], un analogue visuel [HCL05], données géométriques [DSB04].
3. ***Vue sur le rendu :*** une image affichable [HM90], une vue [Chi00], un affichage [HCL05], données d'image [DSB04].

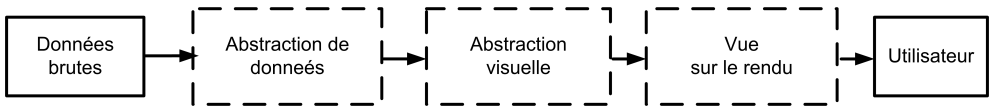


FIGURE 2.11 – Principales étapes de traitement des données.

Ces étapes sont assez génériques et peu formelles. Elles permettent cependant un cadre d'analyse et de comparaison varié selon différents types d'applications : classification des techniques d'interaction visuelle [Chi00], positionnement d'un espace d'optimisation technique avec l'utilisation du GPU pour le rendu graphique [ME09], ainsi que la2 répartition et équilibrage de traitements de données côté client ou côté serveur [WBW96].

2.3.2 Opérations sur les données

Les trois étapes précédemment énumérées constituent des états (intermédiaires) pour les données, au fil d'*opérations*. Ces opérations comportent des opérateurs de deux types

10. En pratique il est difficile d'identifier des données brutes, car toute donnée résulte d'une vue partielle d'un dispositif de collecte (capteur, observation, etc.), qui possède une limite telle que la fréquence d'échantillonnage ou la catégorisation par l'humain.

[CR98, STH03] : opérateurs de *valeurs* (*value operator*) qui modifient les données, et les opérateurs de *vues* (*view operator*) qui modifient la projection des données (sans modifier les données sources). Un troisième type d'opérateur de *transformation de vue* est identifié [Chi00] qui permet la manipulation des rendus par changement de point de vue (rotation, zoom, etc.).

Nous identifions trois catégories d'opérateurs communes (figure 2.12), associés respectivement aux étapes de traitement précédentes :

1. ***Transformation de données :*** enrichissement de données [HM90], transformation des données [Chi00], filtrage [HCL05].
2. ***Transformation d'association visuelle :*** cartographie visuel [HM90], transformation visuelle [Chi00], filtrage [HCL05].
3. ***Transformation de vue :*** rendu [HM90], transformation de cartographie visuelle [Chi00], rendu (interactif) [HCL05].

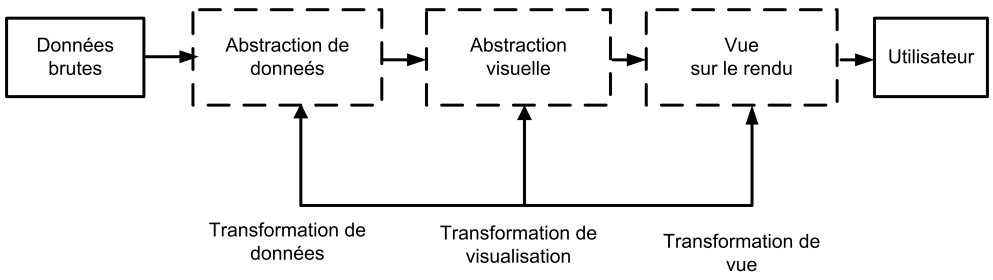


FIGURE 2.12 – Opérateurs de traitement des données.

Ces trois types d'opérateurs possèdent une forme de dépendance [CR98] ce qui augmente la difficulté d'identification de leur périmètre d'action. Par exemple un opérateur de vue, qui filtre des éléments dans l'espace visuel, sans pour autant changer les données à la source, rend ces éléments invisibles alors qu'ils sont présents dans le jeu de données. De même une transformation de point de vue telle que le zoom peut faire varier la taille d'un objet, sans changer ni sa représentation visuelle, ni ses valeurs.

2.4 Association entre attributs des données et variables graphiques

Parmi les étapes du modèle de référence évoqué dans la section précédente, l'étape de *"transformation visuelle"*, autrement dit d'association entre les données et les variables graphiques est essentielle car elle permet aux données abstraites d'être visualisées. De nombreux termes sont employés pour y faire référence : *"Mapping"* [DSB04], *"Visual Mapping"* [STH02], *"Visual Mapping Transformation"* [CR98], *"Visual Form"* [ME09], *"Visualization Mapping"* [HM90].

Cette étape est complexe car elle a pour but d'optimiser les capacités du système visuel et cognitif de l'humain tout en combinant deux mondes hétérogènes que sont d'un côté les *données* (section 2.4.1) et de l'autre les *variables graphiques* (section 2.4.2). Cette combinaison

devant respecter des règles précises et cohérentes souvent regroupées sous le terme de *grammaires visuelles* (section 2.4.3).

2.4.1 Valeurs et types des données

Les données [11] sont le point de départ de toute forme de représentation visuelle. Celles-ci sont dans la réalité stockées et partagées de façons très hétérogènes telles que des tableaux, documents Microsoft Excel [Mic10a], bases de données, etc. Elles peuvent cependant être décrites de deux façons, indépendamment de leur format de stockage : 1) selon leurs valeurs, et 2) selon leur type.

Valeurs des données

Les valeurs des données peuvent être distinguées sous trois formes [CM97] :

- **Nominales :** *qui ne peuvent être qu'égales ou différentes d'autres valeurs.* Par exemple des catégories d'animaux, le sexe ou la couleur des yeux d'un individu, un code postal, etc.
- **Ordinales :** *qui peuvent être égales ou différentes, et comparées.* Par exemple les lettres d'un alphabet, les mois d'une année, les notes des élèves, etc.
- **Quantitatives :** *qui peuvent être égales ou différentes, comparés, et qui peuvent être manipulées par des opérateurs arithmétiques (addition/soustraction, multiplication/division, etc.).* Par exemple toute valeur numérique telles que la température, l'âge d'une personne, etc.

D'autres types particuliers de valeurs existent : sous forme de ratio (rapports de valeurs), d'intervalle de valeur (dates, températures, ..) ou de valeurs discrètes (binaire, etc.).

L'identification de ces valeurs permet une première structure de l'espace de conception de la visualisation [CM97]. Cet espace permet la *création* de visualisations, afin de guider ou d'automatiser [Mac86] le choix de variables graphiques pertinentes à associer aux données (nous le verrons section 2.4.3), mais permet aussi la *description* des interfaces existantes [CM97]. Par exemple l'interface visuelle FilmFinder [AS94b] d'exploration de films (figure 2.13 à droite) peut être décrite au moyen d'une matrice (figure 2.13 à gauche) où les colonnes représentent progressivement l'état de transformation des attributs de données affichés [CM97] :

1. La colonne D indique le type de données que nous venons d'évoquer (Nominal, Ordinal et Quantitatif).
2. La colonne F indique la réduction de données (>), en particulier par sliders (sl>) ou sliders de rang (br>).
3. La colonne D' pour le type de résultat de la transformation.
4. Les colonnes X, Y indiquent que les attributs sont encodés dans l'espace et R indique l'encodage en variables rétiniennes (ici C, la couleur).

11. Nous entendons par *données* une collection d'objets et d'attributs, ces derniers étant des variables qui définissent les propriétés des objets.

Name	D	F	D'	X	Y	Z	T	R	—	[]	CP
Year	Q	>	Q	P							
Quality	Q	>	Q		P						
Type	N	>	N					C			
Title	O	sl>									
Actor	O	sl>									
Actress	O	sl>									
Director	O	sl>									
Length	Q	br>									
Rating	N	br>									

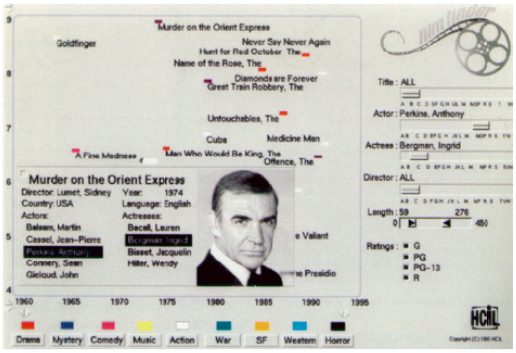


FIGURE 2.13 – Caractérisation de l'interface de FILMFINDER [AS94b] (à droite) selon la grille de transformation de données et de transformation visuelle [CM97] (à gauche).

Types de données

Les *types* de données permettent également d'identifier une structure de l'espace de conception de la visualisation d'information [Shn96, Kei02b] (figure 2.14).

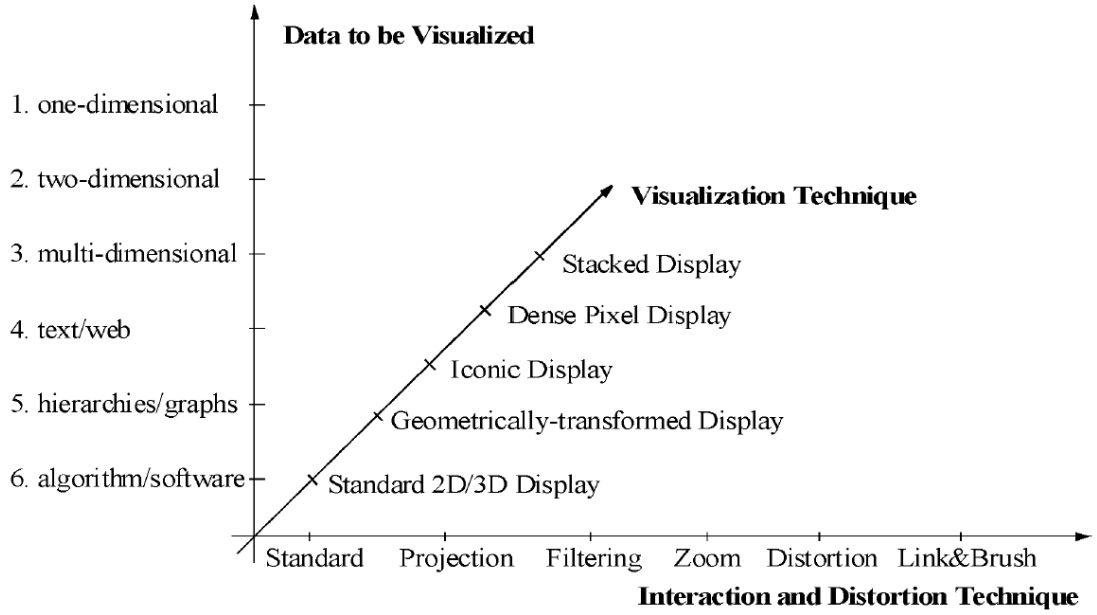


FIGURE 2.14 – Espace de conception de la visualisation interactive en fonction du type de données [Kei02b].

Données à 1-dimension. Il s'agit des données qui peuvent être positionnées sur un axe unique de valeurs. Les données temporelles en sont un exemple, telles qu'une liste d'emails, un flux RSS, une liste de musique ou les événements d'un dossier médical. La représentation la plus classique de ces données est en séries temporelles qui est l'un des types de graphiques le plus utilisés [Tuf01].

Données à 2-dimensions. Il s'agit de données qui peuvent être positionnées dans un plan. Par exemple les données géographiques, une page web, etc. Le plan de représentation peut être soit immédiat (carte géographique, mise en page), soit abstrait tel qu'un graphique où chaque axe représente une dimension.

Données à N-dimensions. Il s'agit de données possédant au moins trois dimensions (on parlera alors de données multidimensionnelles). Par exemple les tuples issus d'une requête de base de données, la description d'un individu, etc. Leur représentation est en général très difficile. Les coordonnées parallèles [ID90] (*parallel coordinates*) permettent d'afficher toutes les dimensions sur un plan au moyen de N axes équidistants représentant chacun l'échelle de valeurs minimum et maximum des attributs. Chaque donnée est ensuite représentée par une polyligne reliant les valeurs des attributs sur chacun de ces axes. Un autre type de visualisation, plus compact, est la visualisation orientée pixel (*pixel-oriented visualization*) où une dimension est encodée dans chaque pixel [Kei02a].

Graphes, Réseaux et Arbres. Les graphes sont des structures de données quasiment omniprésentes dans le monde réel, et un grand nombre d'information peut être représenté sous cette forme. Par exemple des liens hypertextes, les réseaux sociaux, les réseaux de co-citation de publication scientifique, etc. Il existe une très grande diversité de représentation [HMM00] qui se base généralement sur les propriétés topologiques offertes par la théorie des graphes : degré d'un sommet, regroupement, chemin dans le graphe ou connexité du graphe. La disposition du graphe dans le plan est la problématique la plus complexe car dépend de la topologie du réseau et de la tâche à réaliser. La visualisation de graphe est possible par noeuds/liens auto-organisé [Ead84] comme en matrice d'adjacence qui évite les recoupements d'arêtes. Les liens des noeuds d'un réseau possèdent des valeurs ou attributs à représenter, et qui peuvent être de plusieurs dimensions. Concernant les arbres, ils ont la caractéristique de ne pas posséder de circuit et leur structure hiérarchique permet la représentation planaire comme une *treemap* [JS02] (figure 2.15 issu du site web *Map of Markets* [12]) sans superposition de liens.

Texte et document. La représentation de textes est complexe car chaque mot représente une dimension, qu'il n'est pas immédiat de classer, filtrer ou agréger. Les principales approches de représentation de texte se basent sur une quantification de celui-ci, au travers de méthodes statistiques ou d'algorithmes. Les données textuelles peuvent être quantifiées (fréquence, co-occurrences, etc.) et les dimensions d'un document réduites par analyse en composante principale [KO07]. Un texte peut aussi être vu à différent niveaux d'abstraction, en fonction de sa structure ou comme un document à part entière. Il peut alors être classé en fonction de ses meta-données, et visualisé sous forme d'une carte, en fonction de sa proximité et de sa différence avec d'autres documents [RCL+98] (figure 2.16).

Les objets du monde réel sont fortement multidimensionnels, et mélangent plusieurs types de données qui sont autant de points de vue possible pour représenter ou classer ces objets. Par exemple une image peut être vue selon différentes modélisations [GS00] (figure 2.17) : une matrice de pixels, une collection de meta-données, une collection d'objets, etc. qui ont chacun leur technique de représentation.

12. http://www.smartmoney.com/map-of-the-market/

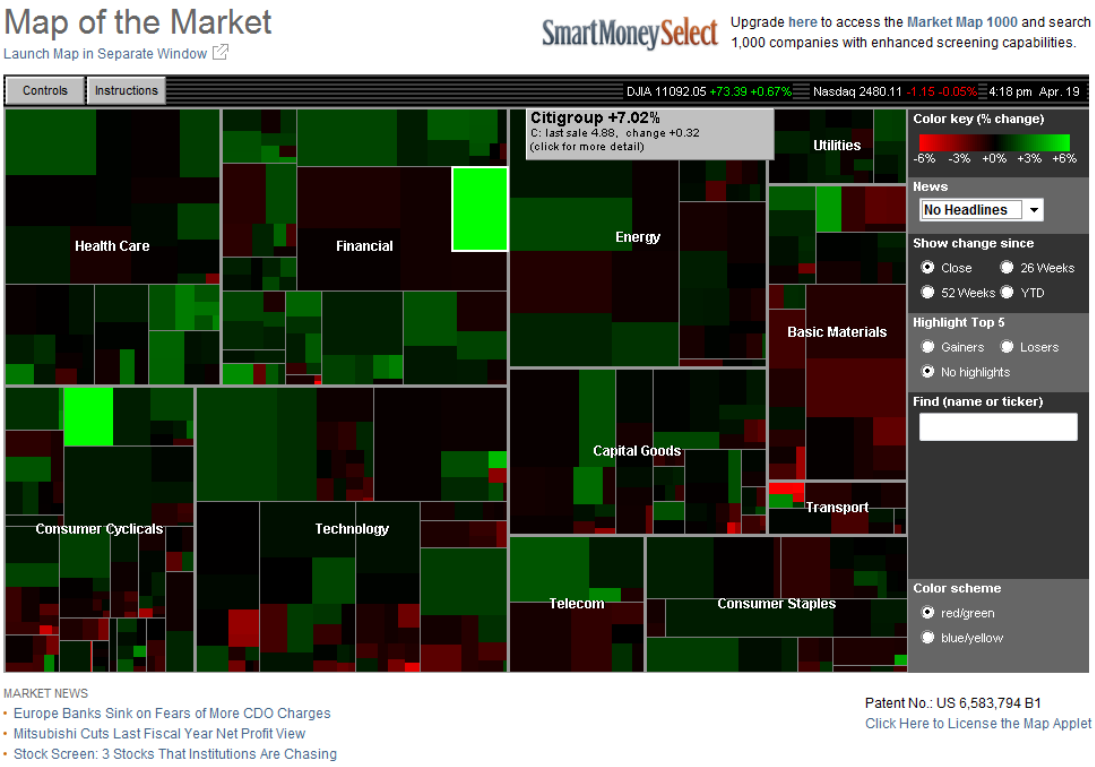


FIGURE 2.15 – Un exemple de Treemap [JS02] : *Map of Markets*.

FIGURE 2.16 – Agencement spatial d'une centaine de documents textuels dans DATA MOUNTAIN [RCL+98]

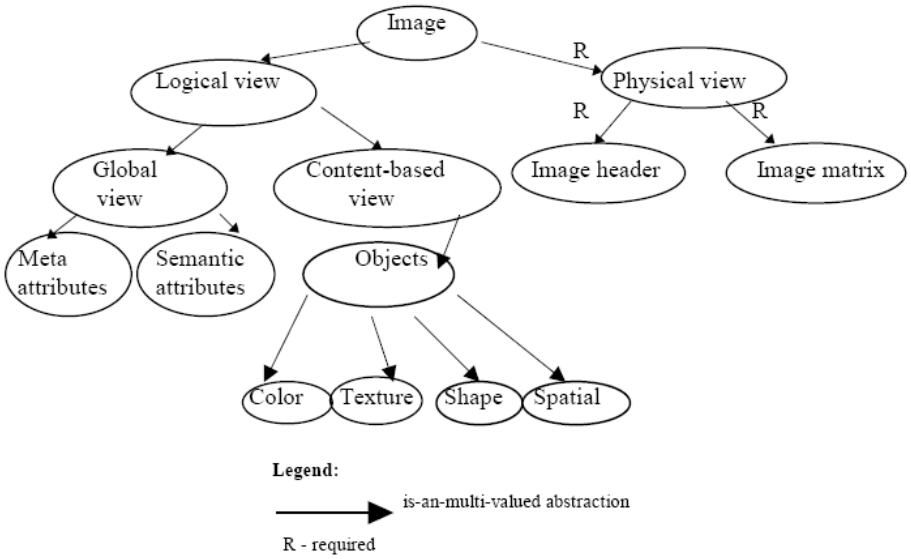


FIGURE 2.17 – Différentes modélisations et types de données d'une image [GS00].

2.4.2 Variables graphiques

Les variables graphiques permettent de représenter visuellement les données. Les principales variables ont notamment été énumérées par [Ber83], dans le cadre de règles de conception cartographique (figure 2.18). Ces règles prennent en compte les capacités de perception de l'humain et fournissent un *langage visuel* adapté. Ces règles ont été reprises par [CM97] qui identifie trois types de représentations élémentaires : les *marques* (*Marks*, que nous nommerons *glyphes* par la suite) tels que les points, lignes, surfaces ; la *position* dans l'espace et le temps ; et enfin un ensemble de *propriétés visuelles* dites rétiniennes *"retinal properties"* telles que la taille ou la couleur.

Position

La position désigne la localisation d'un élément visuel dans un espace. Cette propriété est de loin la plus immédiate pour l'humain, quel que soit la valeur des données [Mac86]. De manière générale la position permet de représenter des relations d'ordre, et d'induire une similarité fonctionnelle (ou non) entre deux objets si ils sont proches (ou non) [Nei93]. La position peut provoquer des phénomènes d'occlusion si deux objets visuels occupent la même localisation dans un plan. Enfin l'évolution progressive de la position au fil du temps permet, via une animation donc, d'encoder l'évolution d'un objet ou d'attirer l'attention sur celui-ci.

Propriétés visuelles

Les principales propriétés visuelles sont la taille (longueur, aire, volume), la couleur (teinte, saturation, luminance), l'orientation, la forme, la densité, la transparence, la texture, la densité, etc. [Mac86] propose une hiérarchie de pertinence de ces propriétés en fonction de la valeur de la donnée à représenter, ainsi qu'un ordre de précision (figure 2.19). De manière similaire à la position, les propriétés visuelles peuvent par exemple encoder des

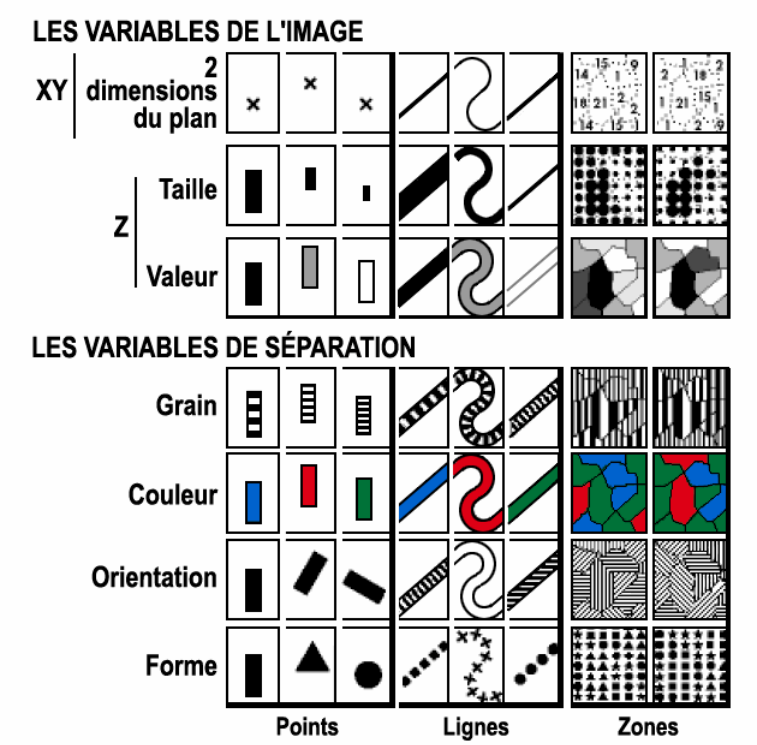


FIGURE 2.18 – Variables graphiques énumérées par [Ber83]

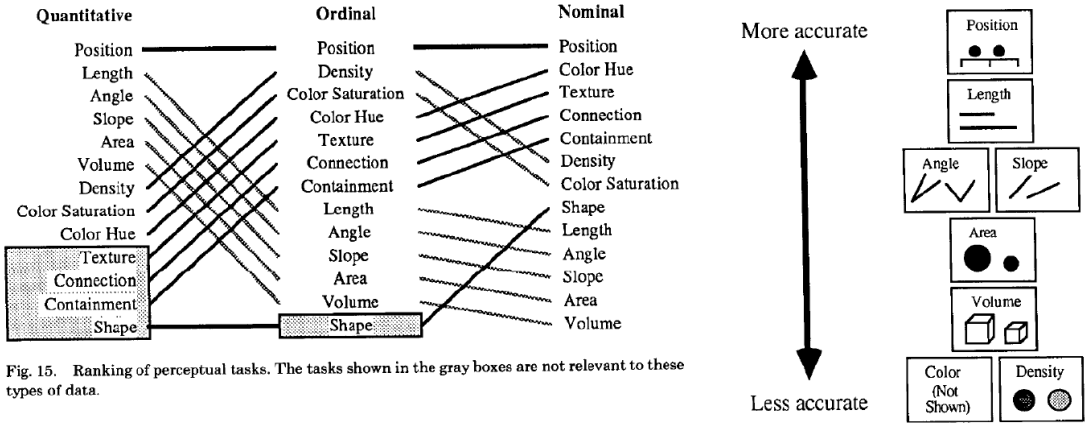


Fig. 15. Ranking of perceptual tasks. The tasks shown in the gray boxes are not relevant to these types of data.

FIGURE 2.19 – Hiérarchie des propriétés visuelles en fonction des valeurs des données (à gauche) et de l'ordre de précision des variables graphiques (à droite) [Mac86].

relations d'ordre (figure 2.20) et de similarité.

Parmi ces propriétés, la couleur possède en particulier de nombreuses contraintes de design. La principale contrainte étant que la teinte de la couleur qui est perçue de manière subjective en fonction du contexte culturel de l'individu (figure 2.21) [13]. Les couleurs peuvent

13. http://www.informationisbeautiful.net/visualizations/colours-in-cultures/

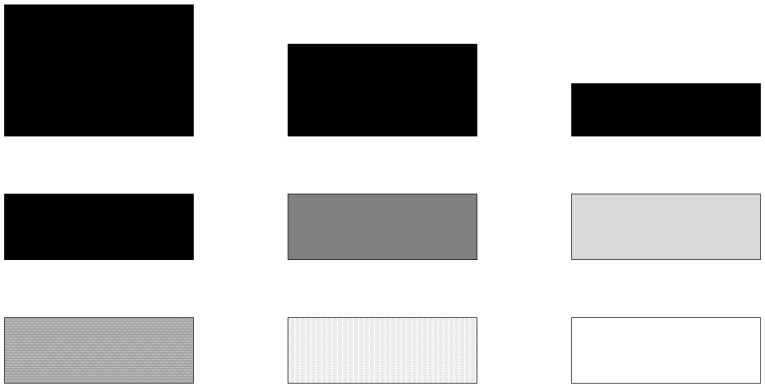


FIGURE 2.20 – Relations d'ordre induites (respectivement de haut en bas,) par la taille, luminosité et motifs.

identifier un nombre maximal de sept éléments sans entrainer une confusion, et certaines couleurs ne peuvent pas êtres combinées entre elles [Wil05]. Enfin les couleurs à forte intensité doivent être réservées pour les éléments importants [Shn96].

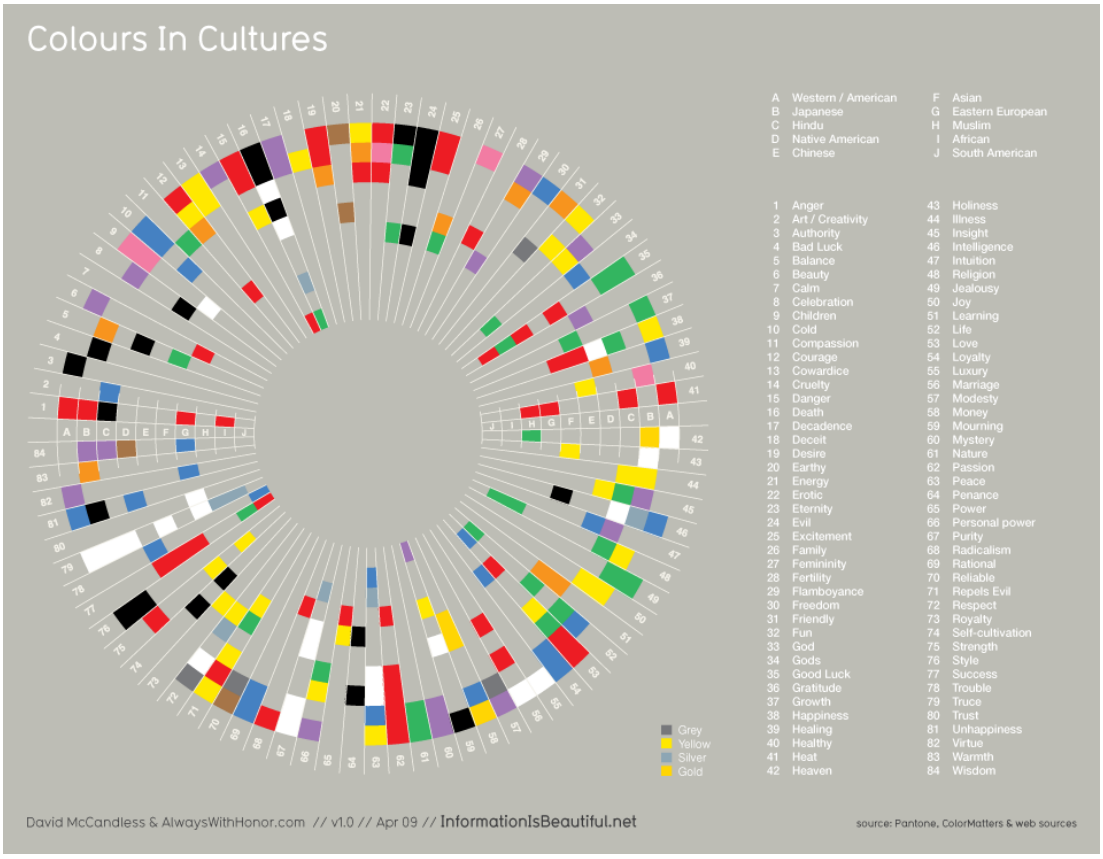


FIGURE 2.21 – Interprétation de la couleur en fonction de la culture.

D'autres propriétés visuelles peuvent être employées (tel que le grisé qui peut indiquer de manière binaire si des boutons sont disponibles ou indisponibles) en jouant sur les contrastes

ou les oppositions (de taille, de couleur) [Pet07], ainsi qu'en utilisant la propriété de perception humaine dite "Gestalt" [14]. Ces propriétés peuvent représenter des états (complexes) d'un système de manière intuitive, tout en minimisant la surcharge visuelle et permettre la concentration de l'utilisateur. Cette concentration peut en effet diminuer si l'utilisateur doit mémoriser la légende des couleurs et des symboles, ou se référer en permanence à une vue externe, pour comprendre les données.

Glyphes

Les glyphes sont le regroupement de plusieurs propriétés graphiques afin d'encoder des données multidimensionnelles. Les propriétés graphiques sont des éléments tels que les points, lignes, régions, icônes, etc. qui sont associés aux valeurs des mesures des dimensions des données. Les glyphes regroupent sous un même terme plusieurs formes de graphiques tels que des histogrammes, des diagrammes en étoile, des icônes, etc.

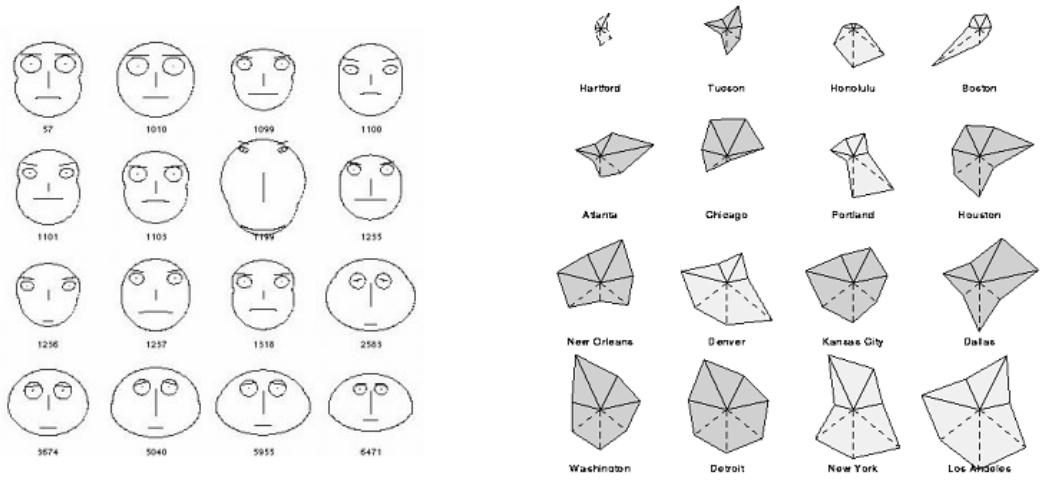


FIGURE 2.22 – Exemple de glyphes pour la visualisation de données multidimensionnelles : visages de Chernoff [Che73] (à gauche) et diagrammes en étoile [FD05] (à droite).

Les glyphes peuvent être *abstraits* ou *métaphoriques* (ce qui facilite leur interprétation). Un exemple de glyphe *abstrait* est le diagramme en étoile [FD05] (*star plots*) (figure 2.22 à droite) dont chacune des branches est une dimension, et une donnée une polyligne (comme pour les coordonnées parallèles). Un exemple de glyphe *métaphorique* est le visage de Chernoff [Che73] (figure 2.22 à gauche) où chaque visage encode jusqu'à 18 dimensions (ou 36 dans le cas de visages asymétriques) dans des attributs du visage : la taille des yeux, forme du visage, longueur de la bouche etc. Cette méthode est efficace car l'humain est habitué à détecter et comparer des nuances fines entre les visages, ainsi l'usage de cette technique de visualisation ne demande pas un apprentissage très long.

2.4.3 Règles d'association

Les règles d'association ont pour objectif de définir les conditions d'association entre les valeurs et types de données d'un côté, et les variables graphiques de l'autre.

14. http://fr.wikipedia.org/wiki/Psychologie_de_la_forme

Le système APT [Mac86] (*A Presentation Tool*, désormais inclue dans TABLEAU SOFTWARE [Sof10], capture d'écran figure 2.23), permet de réduire cet espace de combinaisons en limitant les variables graphiques pertinentes à associer aux valeurs de données. Par exemple, si une dimension a des valeurs numériques, alors la luminance d'une couleur sera proposée pour la représenter (comme pour représenter la valeur du budget figure 2.23), si il s'agit de valeurs cardinales (comme les états américains figure 2.23), la teinte d'une couleur sera proposée[15]. Pour des données multidimensionnelles et les types de données complexe, l'association restera le rôle d'un designer ou de l'utilisateur, qui doit prendre en compte les caractéristiques des valeurs et des données (section 2.4.1), ainsi que les contraintes liées aux variables graphiques (section 2.4.2), mais aussi les propriétés visuelles et les recommandations pour la tâche à réaliser [Mac86, Wil05, Shn96].

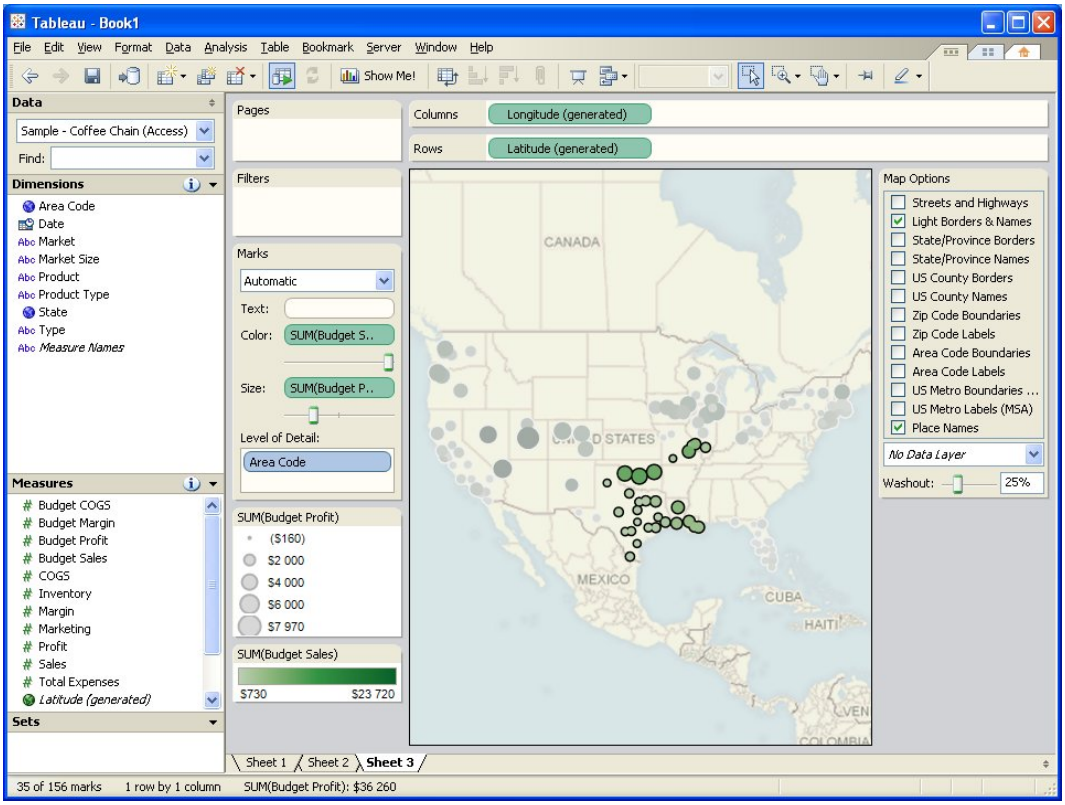


FIGURE 2.23 – Capture d'écran de l'interface de TABLEAU SOFTWARE [Sof10].

De manière formelle, les règles d'association peuvent être vues comme une fonction F [Nor05], qui a les propriétés suivantes d'être :

1. *Calculable* par un algorithme.
2. *Inversible* (fonction inverse F^{-1}), afin de pouvoir reconstruire les données de manière précise, et éviter les ambigüités et erreurs d'interprétation.

15. mais ce choix sera contestable car le nombre d'états américains est de cinquante, et il n'est pas recommandé d'utiliser plus de sept couleurs.

3. *Transmissible* à savoir doit être connue par l'utilisateur pour pouvoir être décodée (par exemple grâce à des légendes).
4. *Connaissable* afin de minimiser la charge cognitive pour la décoder, comme l'utilisation de propriétés visuelles consistantes ou l'utilisation de conventions connues d'un domaine.

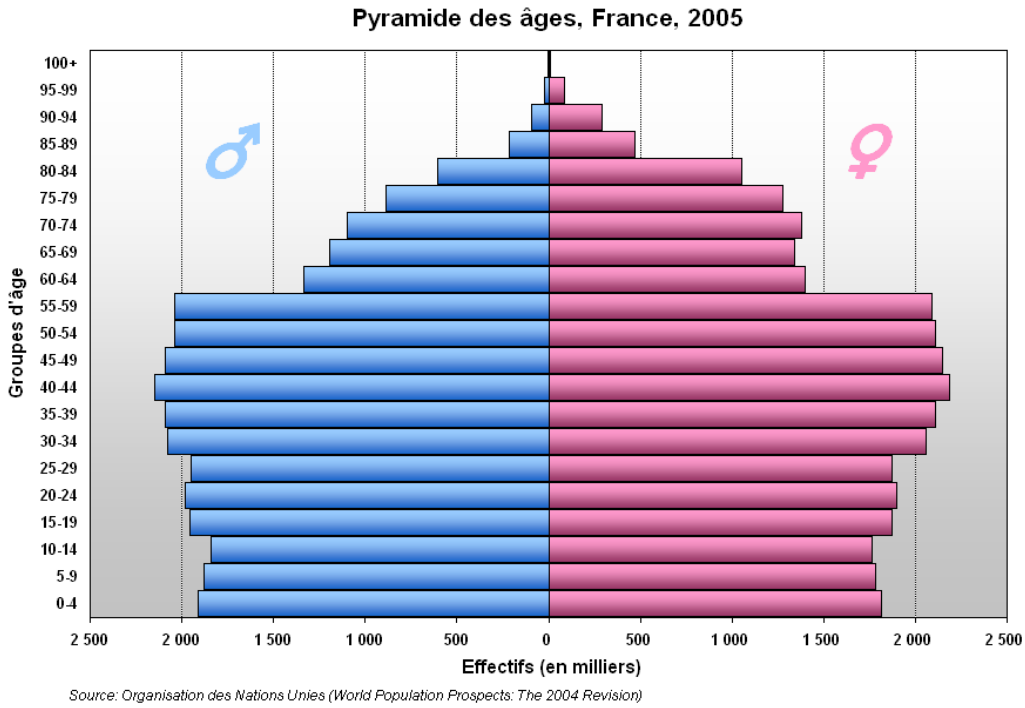


FIGURE 2.24 – Exemple de pyramide des âges (2005, en France).

Une propriété de l'association visuelle qui est peut abordée est celle la *redondance visuelle*. Cette propriété consiste à encoder une même dimension en deux variables graphiques, ou plus. L'intérêt est de pouvoir renforcer la communication d'une dimension, d'éviter les ambiguïtés, ainsi que les erreurs d'interprétation. La pyramide des âges est un exemple typique de redondance (figure 2.24) [16]. Elle permet de communiquer la distribution des individus au sein d'une population, en fonction de leur âge et de leur nombre. La dimension *sexe de la population* y est triplement encodée : position (gauche ou droite), couleur (bleu ou rose) et symboles (colorés eux aussi, donc par déduction une légende n'est plus indispensable). Un autre exemple de redondance est, sur la figure 2.23, la valeur du budget encodé en taille et en luminance. La redondance est également parfois nécessaire. Par exemple les visages de Chernoff sont redondants en terme de position (car ils sont symétriques) mais cela facilite leur interprétation et comparaison. [Tuf01], qui suggère pourtant une approche minimaliste de la conception graphique, reconnait même que la redondance est utile afin de permettre une comparaison ou d'aider l'utilisateur à mieux explorer les données en incluant une continuité de l'information (comme par exemple à un très haut niveau d'altitude sur une GOOGLE MAPS [Goo10d], les pays sont certes présents en double, mais cela permet la continuité de la carte).

16. http://upload.wikimedia.org/wikipedia/commons/e/e0/Pyramide_France.PNG

2.5 Synthèse du chapitre

Dans ce chapitre, nous avons tenté d'énumérer les principaux axes de l'espace de design de la visualisation d'information. Nous avons commencé par étudier certains domaines dont hérite la visualisation d'information, comme l'informatique graphique, la visualisation scientifique et les systèmes d'information géographique. Ces domaines ont permis de réaliser une comparaison et un lien avec les structures de données ou techniques de représentation actuellement utilisées en visualisation d'information (graphe de scène, animation, etc.). Nous avons ensuite étudié le modèle de référence de la visualisation d'information, afin de connaitre les étapes complémentaires qui permettent de transformer les données en images. En particulier, nous nous sommes intéressés à l'étape d'abstraction visuelle (à savoir l'association entre données et variables graphiques) qui est propre à la visualisation d'information et intègre de nombreuses contraintes de design.

Nous pensons que la synthèse suivante de l'espace de design de la visualisation d'information (figure 4.12) permet d'identifier les principaux états des données, et pour chacun les différentes alternatives possibles :

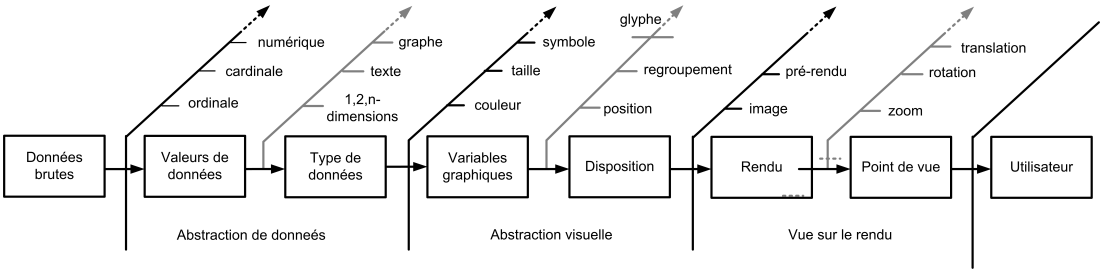


FIGURE 2.25 – Synthèse de l'espace de design de la visualisation d'information.

Nous allons dans le chapitre suivant (chapitre 3) nous intéresser aux différentes interactions qu'il est possible d'associer aux étapes de traitement des données, afin de permettre à un utilisateur d'explorer cet espace de design.

Chapitre 3

Interactions pour la visualisation

Sommaire

3.1 Introduction

Dans le chapitre précédent (Chapitre 2), nous avons identifié un espace de design de la visualisation d'information composé de trois types d'état des données :

1. ***Abstraction de données.***
2. ***Abstraction visuelle.***
3. ***Vue sur le rendu.***

A ces états, sont associés trois types d'opérations de transformation :

1. ***Transformation de données.***
2. ***Transformation d'association visuelle.***
3. ***Transformation de vue.***

Dans ce chapitre nous allons étudier les techniques d'interaction permettant à l'utilisateur de réaliser ces opérations de changement d'état des données. Selon l'approche écologique de [Gib86], il est même nécessaire d'agir pour percevoir, les deux étant indissociables. Nous allons donc intégrer dynamiquement l'utilisateur dans le cycle d'exécution des programmes, pour les rendres *interactifs*. Cependant les programmes interactifs ne permettent pas *automatiquement* une navigation satisfaisante dans les visualisations. Par exemple (figure 3.1), le rendu de la visualisation sous forme d'image (qui est le format très souvent utilisé pour le partage de visualisations) permet certes une opération de transformation de point de vue telle que le zoom (type bitmap). Mais une pixellisation sera engendrée (car le nombre de pixels de l'image reste le même, mais le nombre de pixels pour l'afficher augmente proportionnellement au facteur de zoom) et le programme d'affichage ne permet pas d'obtenir de détails sur les données, ni même de régénérer l'image pour l'afficher avec une résolution adaptée. Pour résoudre ce problème, il est nécessaire de régénérer une image à un meilleur niveau de résolution, et d'inclure de nouveaux détails d'abstraction visuelle, ce qui n'est pas toujours possible.

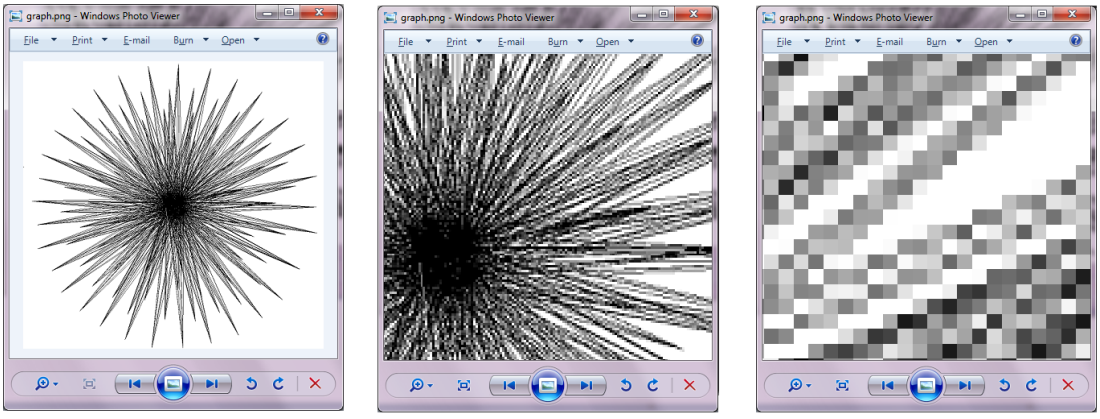


FIGURE 3.1 – Zoom progressif de type bitmap dans le rendu d'une en image d'une visualisation.

Nous distinguons dans ce chapitre deux méthodes d'interaction permettant le changement d'état des données : dans un premier temps les *techniques d'interaction* (section 3.2), qui

permettent principalement de composer une requête et changer le paramétrage des états, et ensuite les *interfaces graphiques* (section 3.3) qui permettent principalement d'afficher et d'interagir avec le rendu des données.

3.2 Techniques d'interaction

Derrière le terme *techniques d'interaction* peut être regroupé un grand nombre d'interactions dont le design, l'implémentation et l'évaluation sont l'objet de la discipline de l'Interaction Homme/Machine (IHM)[1]. Dans le cadre de nos travaux, nous n'étudierons que le sous-ensemble de ces techniques applicables aux représentations visuelles d'information, à savoir qui permettent de faire varier l'espace de design et d'afficher le rendu.

Nous allons étudier les widgets interactifs (section 3.2.1) et le type de requêtes qu'ils permettent d'exprimer (section 3.2.2). Dans une dernière partie (section 3.2.3) nous discuterons d'autres types de techniques existantes et émergentes.

3.2.1 Widgets graphiques interactifs

Les widgets (*window gadget*) [SA88] sont des composants graphiques qui offrent à l'utilisateur un langage visuel facile à utiliser (comme au moyen de simples clics de souris par exemple), pour réaliser une tâche élémentaire (telle que le filtrage d'information ou la saisie d'un mot clé). Une tâche plus complexe pourra est réalisée par combinaison des tâches élémentaires, à la condition que le nombre de widgets et leur capacité à explorer un rang de valeurs soit suffisante.

Types de widgets et gestion des événements

Les widgets doivent aussi posséder une affordance, à savoir offrir une manière naturelle d'être utilisés sans phase d'apprentissage préalable. Par exemple figure 3.1, il est naturel que le zoom sur le rendu est réalisable en cliquant sur l'icône de la loupe. D'autres widgets peuvent être utilisés (exemples sur la figure 3.2) et sont fournis par les bibliothèques standards[2] telles que SUN SWING[3] Standard Widget Toolkit[4] (SWT), Microsoft MFC (*Microsoft Foundation Classes*), et aussi celles orientées Web apparues ces dernières années, comme Google Web Toolkit (GWT) [htt10a] ou Yahoo User Interfaces (YUI) [htt10b].

D'un point de vue fonctionnel, les tâches élémentaires qu'ils permettent de réaliser dépendent de la valeur des données (section 2.4.1) et ont un rôle fonctionnel très varié :

– **Sélection, filtrage :** il s'agit d'opérations naturelles qui visent respectivement à sélectionner une donnée et à restreindre les données à un sous-ensemble. Ces fonctionnalités permettent ainsi à l'utilisateur d'éviter d'écrire une requête SQL (*Structured Query Language*, dont le format est bien défini, strict et du domaine du programmeur avec

1. ainsi que la conception des menus, des boîtes de dialogues, l'optimisation du positionnement des éléments interactifs dans un plan ou des techniques générales telles que la manipulation directe [Shn87].
2. http://java.sun.com/docs/books/tutorial/ui/features/components.html
3. http://java.sun.com/javase/6/docs/technotes/guides/swing/
4. http://www.eclipse.org/swt/

FIGURE 3.2 – Exemple de widgets disponibles dans la bibliothèque JAVA SWING.

un temps d'apprentissage long pour les novices [AWS92]). Exemples : bouton, slider, menu, liste de sélection, liens hypertextes, etc.

- **Navigation de la vue** : permet d'explorer et de comprendre l'image rendue, car l'image peut être partiellement visible ou être superposée. Exemples : barre de défilement (*scrollbar*), tabulations, zoom, etc.
- **Saisie de données :** permet à l'utilisateur de saisir dans le système des données de manière libre ou semi-assistée (comme avec un système de recommandation, illustration figure 3.5). Exemples : boîte de saisie (*text field*), champs de saisie (*box field*), etc.
- **Affichage de détails et légendes :** permet d'inclure des données dans le widget et d'afficher des détails sur le rendu [FP99]. Ils permettent aussi de rendre les légendes interactives [RLP10] afin d'explorer l'espace d'abstraction visuelle. Exemples : labels, barres d'informations, info bulles etc.

Les boîtes à outils qui fournissent les widgets intègrent également un mécanisme de gestion de l'état du widget, qui peut être évènementielle afin de le relier à un programme. Cette étape de couplage (*binding*) permet par exemple d'associer une fonction de rappel (*callback*) afin de mettre à jour l'état du widget, de soumettre la requête de l'utilisateur, mais aussi de déclencher des mécanismes de vérification d'erreur ou de recommandation de requêtes (figure 3.5). Ces mécanismes peuvent être déclenchés aussi bien en fonction des données qui sont saisies dans le système, qu'en fonction de la position du dispositif de pointage sur l'écran (la souris par exemple) et de l'état de l'action qui y est associée (survol par la souris, focus, clic engagé, clique relâché, etc.). Certains widgets ne permettent pas de réaliser d'erreur "syntaxique" en terme de requête, car le vocabulaire est contrôlé (comme dans les boutons radios, checkboxes ou listes) alors que d'autre widgets (comme la saisie de texte libre) permettent d'explorer un espace de données imprévisible.

Intégration de visualisations dans les widgets

Les widgets occupent un espace sur l'écran qui peut servir à afficher soit leur état actuel (comme un bouton radio coché ou non), soit l'état d'un autre widget, soit d'afficher l'espace de données à explorer sous forme graphique : intervalle de valeurs, distribution de données, etc. Les *alpha sliders* [AS94a] permettent d'énumérer le champ des valeurs qu'il est possible de sélectionner. Les *"scented widgets"* [WHA07] reposent sur le même principe où des éléments visuels "scented" (qui sont souvent représentés sous forme de graphiques) permettent à l'utilisateur d'anticiper les résultat de sa requête, tout en gardant son focus sur le widget. A noter que les widgets ne permettent pas d'afficher en général de grandes quantités de données

(une dizaine d'éléments sont visibles à chaque instant).

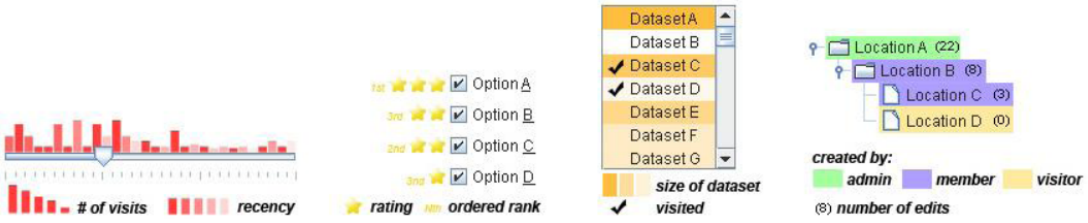


FIGURE 3.3 – Exemple de *"scented widgets"* [WHA07].

3.2.2 Réactivité des interactions et construction de requête

L'expression des besoins de l'utilisateur au moyen de widgets n'est cependant pas un processus linéaire, mais composé d'allers/retours fréquents. Ils doivent permettre à l'utilisateur de rentrer dans un processus incrémental et réversible d'expression de ses besoins. Les widgets doivent donc être très réactifs et permettre immédiatement d'avoir une rétroaction continue engendrée par la manipulation directe [Shn87] de l'utilisateur. Nous allons également nous intéresser aux différentes requêtes qui peuvent être construites et qui peuvent demander une réactivité différente.

Réactivité et rétroaction des interactions

Les widgets permettent à l'utilisateur d'effectuer un processus d'actions et de réactions immédiates avec le système, deux propriétés importantes doivent être garanties : la *rétroaction* et la *réactivité*.

La rétroaction (*feedback*). Il s'agit de permettre à l'utilisateur de connaître l'état du système suite à une interaction [NB02]. L'utilisateur pourra par exemple savoir si sa requête a été prise en compte, si elle est en cours d'exécution, ou si elle est déjà exécutée. L'utilisateur peut ainsi éviter de répéter une action et peut savoir ce qu'il lui est possible de faire ensuite.

La réactivité. Il s'agit du temps de réponse du système suite à une action de l'utilisateur. [Nei93] indique que même si ce temps doit être le plus rapide possible, dans la réalité, cela est rarement le cas. Rendre les systèmes réactifs est même un défi majeur en visualisation [Hib04] en général. Différentes stratégies sont donc à envisager, et permettent d'adapter le type de rétroactivité en fonction du temps de réponse.

[Nei93] identifie trois types de réactivité :

- **< 0.1s** : donne l'impression à l'utilisateur que le système réagit instantanément et aucune rétroactivité n'est nécessaire (hors mis celles liées à l'affordance, comme indiquer qu'un bouton se presse et se relâche), et le résultat s'affiche.
- **< 1.0s** : n'interrompt pas le fil de pensées de l'utilisateur, mais celui-ci a tout de même remarqué le retard du système. Il est possible d'indiquer une attente temporaire.

– **< 10s** : est la limite pour garder l'attention de l'utilisateur sur la vue ou boîte de dialogue en cours de présentation. Une rétroaction est indispensable afin de communiquer à l'utilisateur ce qu'il doit attendre du système, et quelles sont ses actions possibles.

Le dispositif de pointage peut être le lieu de la rétroaction, avec une icône variable en fonction des différents états du système (figure 3.4) indiquant respectivement les différents temps de réponse de [Nei93]. D'autres formes de disponibilités peuvent être intégrées dans le widget ou dans le rendu associé, au moyen des variables graphiques énumérées précédemment (section 2.4.2).

FIGURE 3.4 – Etats possibles du dispositif de pointage (un curseur de souris) afin d'indiquer l'état de disponibilité du système ou de la vue en cours.

La suggestion de requêtes ou l'aide à la composition de requête est une autre forme de rétroactivité. C'est une façon d'assister l'utilisateur et de guider sa requête (exemple figure 3.5 de recommandation de requêtes les plus fréquentes à partir des mots déjà saisis) sans passer par un vocabulaire contrôlé dans une liste fermée de choix.

FIGURE 3.5 – Suggestion dynamique de mots clés dans une requête GOOGLE [Goo10a].

Les auteurs [SPCJ09] généralisent la réactivité sous le terme général de *"qualité de service"* d'un système. Ils ajoutent que même si un temps de réponse rapide (à savoir $< 0.1s$) permet d'augmenter la productivité, il peut augmenter le taux d'erreur dans le cas de tâches complexes. Certes un système doit être réactif quand il s'agit de situations critiques à gérer en temps réel. Mais il est parfois nécessaire d'attendre pour ne pas interrompre une tâche en cours qui est prioritaire [SPCJ09]. Ainsi un design judicieux doit accompagner les temps de réponse, et s'adapter aux cas où celui-ci varie, notamment dans le cas de la recommandation de requête GOOGLE figure 3.5 dont le temps de réponse (et par conséquent la réactivité du widget) dépend de la latence de la connexion réseau.

Construction et affichage de résultats de requête

La succession des interactions avec les widgets par l'utilisateur constitue une *requête*. Trois types de requêtes peuvent être identifiés en fonction du temps de calcul du résultat : *requête dynamique* où le résultat est affiché à chaque interaction avec le système, *requête progressive* où une prévisualisation du résultat de la requête est affichée à chaque interaction et enfin la *requête par composition* ne déclenche le calcul du résultat qu'une fois la soumission de la requête au système effectuée.

La *requête dynamique* (*dynamic query*) [AWS92] permet de visualiser immédiatement le résultat de la requête en cours de construction, ce qui permet par exemple de réaliser efficacement une requête multicritères par ajustements successifs. Ce précédé est efficace quand il y a peu de données, et donc permet de former des hypothèses et de les tester rapidement comme dans le cas du système FilmFinder [AS94b] (figure 2.13 à droite) ou une table périodique d'éléments [AWS92] (figure 3.6).

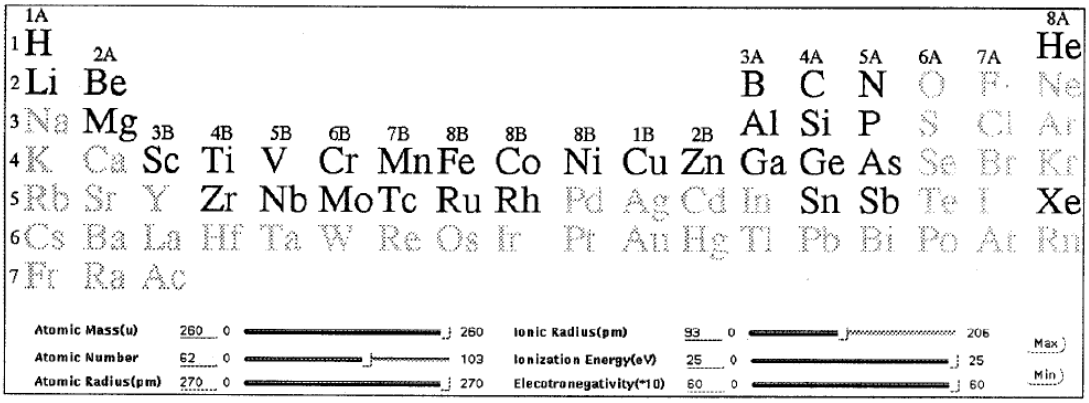


FIGURE 3.6 – Filtrage dynamique appliqué à la table périodique des éléments [AWS92].

La *requête progressive* [PSDB99] est transmise au fur et à mesure de sa composition, afin d'obtenir un aperçu (comme la suggestion de requête figure 3.5), mais le résultat complet sera affiché une fois l'interaction avec le widget finie ou la requête soumise au système.

La *requête par composition* ne produit aucun résultat avant d'être soumise par un widget de soumission. Comme par exemple un formulaire sur une page web ou une recherche par mot clé classique dans un moteur de recherche (sans recommandation). Pour soumettre la requête au système, un widget particulier (souvent un bouton) sera associé à une fonction de rappel qui communiquera la requête au noyau fonctionnel du système.

Il existe des schémas hybrides de requêtes (en termes de temps de réponse) en fonction du parcours de l'utilisateur dans l'espace de données. A titre d'exemple, le site web Kayak [Kay10] (figure 3.7, à gauche) permet de choisir un itinéraire d'avion, en fonction d'un lieu de départ, d'une destination et d'une date d'aller/retour. Le trajet une fois saisi (au moyen du formulaire sur la capture d'écran à gauche figure 3.7), et soumis par un clic sur un bouton "Rechercher" mettra le système dans un état dit occupé avant d'afficher le résultat au bout de quelques secondes sur une nouvelle page web (figure 3.7, à droite). Cette

attente est acceptable par l'utilisateur, car elle est prédictible, des variables visuelles (symboliques) indiquent le temps d'attente, et aussi car il s'agit d'une requête qui ne sera effectuée qu'une seule fois car elle ne varie pas (dû moins pendant cette session de recherche). Une nouvelle page web (figure 3.7, à droite) affiche les dates et tarifs possibles, qui pourront être explorés au moyen de requêtes dynamiques. Ces requêtes ont un temps de réponse court ce qui permet à l'utilisateur de réaliser une tâche de recherche du coût minimal du trajet par balayage d'un intervalle valeurs (comme les jours de départ et d'arrivée) et par visualisation des résultats.

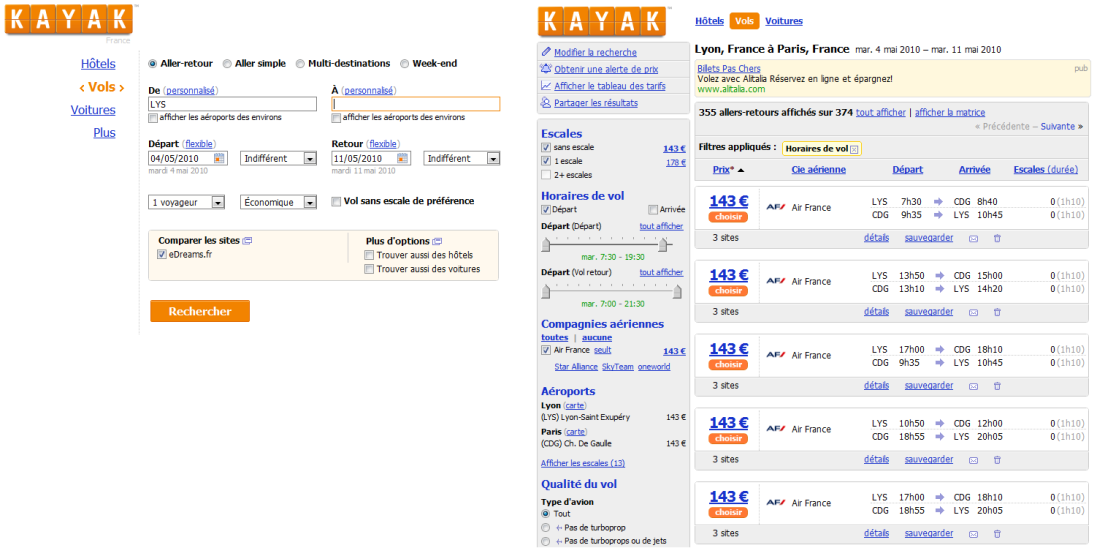


FIGURE 3.7 – Capture d'écran du site KAYAK : composition de requête (à gauche) et exploration dynamique des résultats (à droite).

3.2.3 Autres types de techniques d'interaction

Nous nous sommes intéressés aux interfaces de type *"WIMP"* (*Window*, *Icon*, *Menu*, *Pointing device*), très utilisées depuis leur création au XEROX PARC dans les années 80. Ces interfaces ont l'avantage d'être simples et maitrisés par les utilisateurs sans besoin de phase préalable d'apprentissage. Seulement, les widgets peuvent devenir assez nombreux et occuper beaucoup d'espace sur l'écran (*screen real estate*) au détriment des données. D'autres alternatives peuvent libérer de la place et posséder de nouvelles formes d'interaction : *"post-wimp"* [vD97] (exemple de BRYCE 3D [Bry10] sur la figure 3.13), gestuelles, vocales, etc. De même en sortie, la notification de l'état d'un widget (et par extension de l'état du système) peut utiliser d'autres canaux (dits "modalités") tels que sonores ou haptiques (qui utilise le sens du toucher).

3.3 Types d'interfaces graphiques

Une interface graphique est constituée de l'agencement d'une ou plusieurs vues sur le rendu des données, et permet (entre autres) une transformation du point de vue sur le rendu. Nous allons nous intéresser dans cette section aux principaux types d'interfaces graphiques [5]

5. Pour une revue plus exhaustive des types d'interfaces graphiques, il existe une très bonne référence [CKB08].

à savoir les *vues multiples* (section 3.3.1), qui offrent différents points de vue sur un même objet d'étude, ainsi que les *interfaces multi-échelles* (section 3.3.2) qui offrent plusieurs niveaux progressifs d'exploration de l'espace de rendu.

3.3.1 Vues multiples

On parle de "vues multiples" dès que le nombre de vues sur un même objet est égal ou supérieur à deux [WBWK00]. La motivation de telles interfaces part du constat que parfois, il n'est pas possible sous peine de superposition visuelle (*visual cluttering*) de combiner plusieurs représentations dans un même espace. L'approche repose sur la séparation en différentes vues, et sur la capacité de l'utilisateur à les relier entre elles par similarités de données ou de variables graphiques, ainsi qu'au fil d'interactions. Ces types de vues sont très utilisées et implémentées dans les principaux logiciels d'exploration visuelle actuels [Rob07].

Vues multiples coordonnées

Les vues multiples coordonnées (*Multiple Coordinated Views, MCV*) possèdent un mécanisme de synchronisation, qui permet de répercuter une transformation effectuée sur une vue, vers les autres. Ces autres vues pouvant avoir différentes tailles et résolutions de rendu, ainsi qu'un type d'abstraction visuelle différent. Par exemple le système IMPROVISE [Wea04] illustré figure 3.8, connecte entre elles des listes, des vues statistiques et des vues géographiques sur les données.

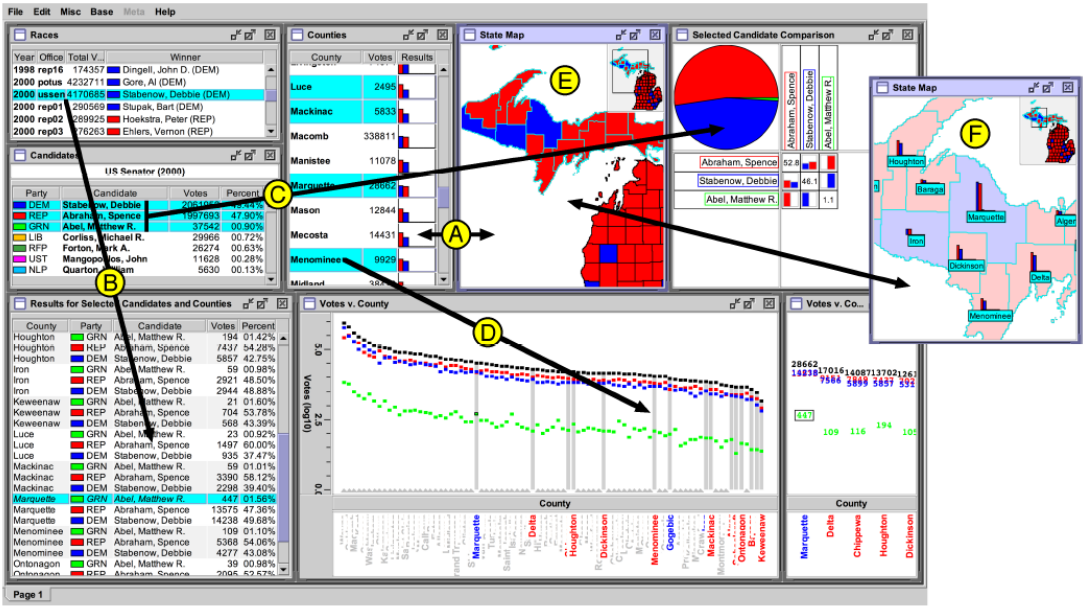


FIGURE 3.8 – Improvise [Wea04] permet la coordination entre différents points de vue sur les données.

Un cas particulier de vues multiples est celui des vues dites *globales+détails* sont des vues coordonnées qui possèdent le même point de vue sur les données, mais à un niveau de résolution différent (comme l'aperçu de la carte globale couplée à la vue détaillée dans

Google Maps [Goo10d] figure 3.9).

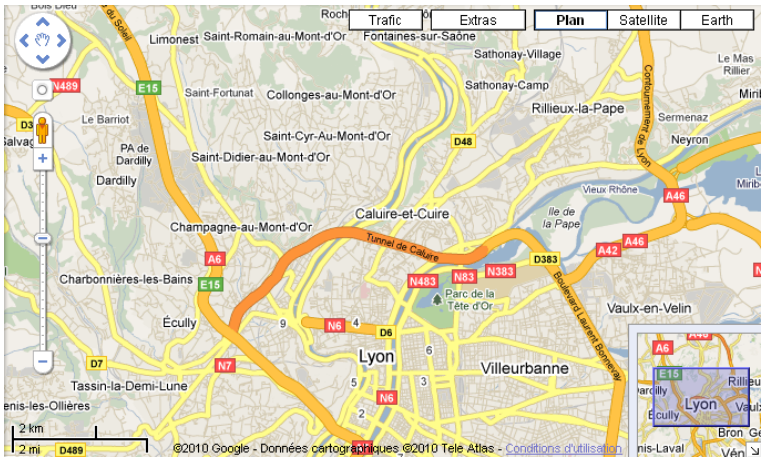


Figure 3.9 – Vue globale et détail dans Google Maps [Goo10d].

Des interactions spécifiques à ces vues permettent de découvrir les relations causales entre différentes dimensions des données, et de réaliser des tâches complexes comme déduire l'apparition ou le déroulement d'un phénomène complexe. Le *"brushing and linking"* [Kei02b] est une technique spécifique aux MCV qui permet d'effectuer une requête (telle qu'une sélection) sur une vue, et qui sera répercutée sur les autres vues. La synchronisation entre les vues peut être définie dynamiquement, comme le système Snap-Together [NS00] qui permet de définir les attributs qui sont synchronisés et offre ainsi une adaptabilité dans le mécanisme de synchronisation entre les vues.

Vues par facettes

Les vues par facettes offrent elles aussi différents points de vue sur les données, où chaque vue représente une catégorie ou *facette* prédéterminée [EHS+02]. Chaque vue propose un mécanisme de sélection selon la facette des données, et peut être appliqué au filtrage de requête de recherche d'image [YSLH03, Fla10a] (figure 3.10, filtrage par date, annotations, taille de l'image, etc.).

Vues focus+contexte

Les vues *focus+contexte* sont des vues multiples avec continuité spatiale entre un "focus" (information d'intérêt qui est détaillée) et son "contexte" [LA94] (information générale et de taille réduite). Ce type de vues permet de repositionner un objet dans l'espace des données qui l'entoure, sans changement de contexte. L'identification et l'interprétation de l'information en est donc facilité, mais une déformation (géométrique) est souvent nécessaire. [Fur86] a proposé un cadre général de ces techniques de déformations. La déformation peut être linéaire (exemple du mur fuyant [MRC91] *The Perspective Wall* figure 3.11), par palier ou hyperbolique (la technique la plus connue est celle de *fish-eye* [Fur86]). A noter que l'impact de la déformation du rendu peut cependant modifier l'abstraction visuelle (par déformation non-linéaire d'icônes et de symboles), et le dispositif de pointage (qui indique le focus) n'est pas forcément la zone d'attention de l'utilisateur (l'utilisateur doit donc déplacer le dispositif

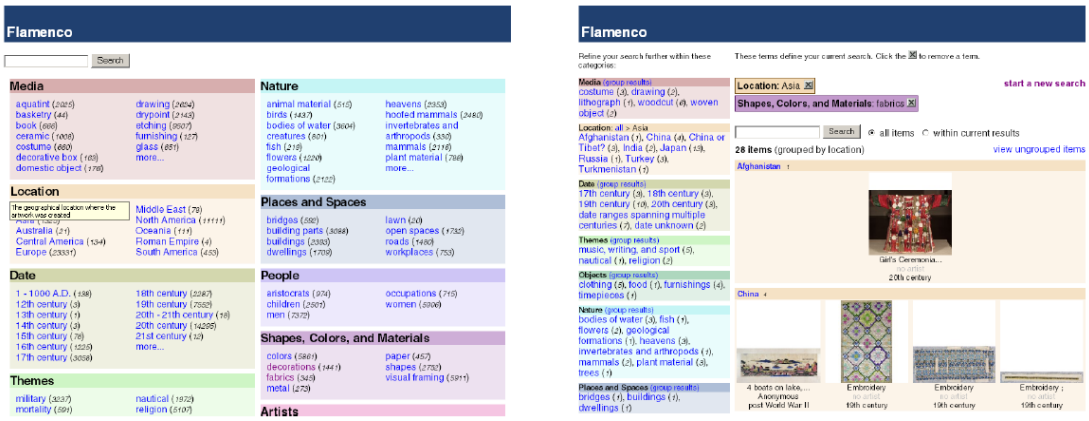


FIGURE 3.10 – Vues par facettes de recherche d'image [Fla10a] : interface de recherche dans une base d'image (à gauche) et présentation des résultats et filtrage par facettes (à droite).

de pointage pour obtenir les focus désirés).

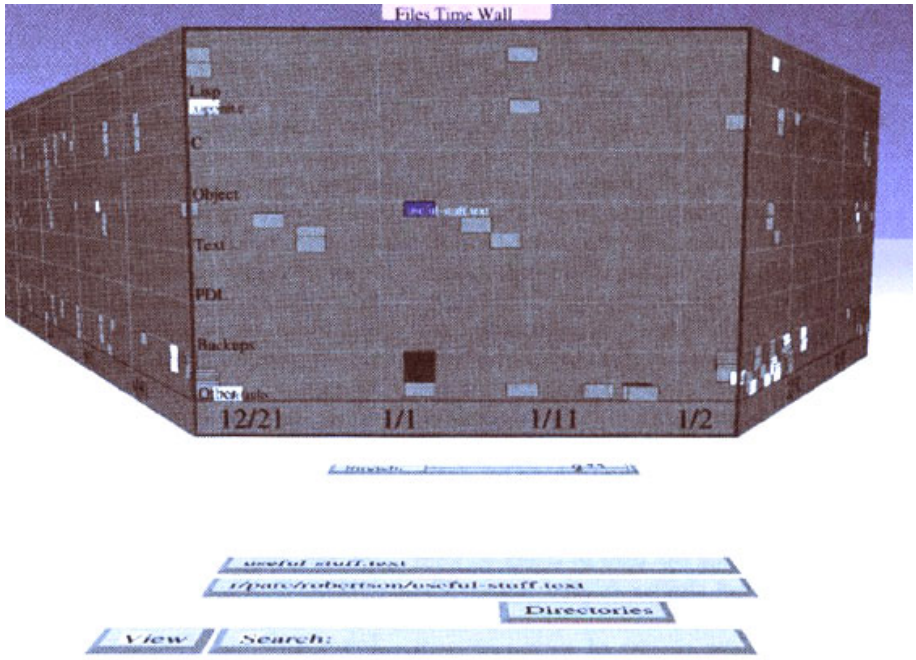


FIGURE 3.11 – Le mur fuyant [MRC91] (*The Perspective Wall*) permet une vue focus+contexte.

Il existe d'autres techniques qui permettent de recontextualiser une information sans déformation. [FP99] proposent de détailler l'information d'intérêt au moyen de labels "excentrés" (*excentric labels*) sans recouvrir ou déformer le focus. D'autres variables graphiques peuvent être utilisées comme [KMH02] qui propose de laisser l'élément sélectionné (focus) identique, mais d'appliquer un flou à l'arrière plan afin de mettre le focus en avant. Le flou permettant de garder un contexte sans déformation spatiale (méthode nommée *Semantic Depth of Field*).

Conclusion

Les vues multiples permettent une exploration de l'espace de vue sur le rendu, mais qui nécessite de générer des rendus supplémentaires pouvant conduire à des calculs et des contrôles de synchronisation couteux [WBWK00]. Une surcharge cognitive pour l'utilisateur peut être engendrée, car il devra par exemple comparer des éléments avec un changement de contexte, à cause de la discontinuité spatiale et des différentes abstractions visuelles utilisées. Enfin si le nombre de vues multiples est important, leur taille en est inversement proportionnelle et donc diminue.

3.3.2 Interfaces multi-échelles

Les interfaces multi-échelles sont des interfaces qui n'offrent qu'une vue sur les données à la fois. Elles sont particulièrement adaptées à un dispositif unique de visualisation (un seul écran, souvent de taille réduite) et permettent de ne pas surcharger visuellement l'utilisateur.

Interfaces zoomables

Les interfaces zoomables (*Zooming/Zoomable User Interface (ZUI)*) permettent de naviguer dans un espace de rendu infini. Il s'agit donc d'une stratégie d'exploration du point de vue sur les données, afin d'obtenir une vue plus détaillée sur un objet dans cet espace. L'objet ou la partie zoomée, deviendra la nouvelle interface de navigation, elle aussi zoomable. L'article de [FB95] donne les fondations de ce type d'interface, qui repose sur la navigation dans un diagramme espace/échelle (*space/scale diagram*). Un exemple d'application de telles interfaces pour la gestion de photos est le logiciel PHOTOMESA [Bed01] (figure 3.12), qui permet de zoomer sur des groupes de photos, dans une interface combinée à d'autres vues de filtrage. Un autre exemple est la navigation dans un graphe (figure 2.2) avec PICCOLO2D [BGM04].

La navigation dans une interface zoomable est composée de la succession de transformation de vues, au moyen d'animations de transitions. Ces animations permettent à l'utilisateur d'identifier le parcours qu'il réalise et de mémoriser son contexte. La position de l'utilisateur dans cet espace constitue une requête et permettra l'apparition ou la disparition d'élément en fonction du point de vue. Cette technique de filtrage s'appelle le *zoom sémantique* et permet aussi de faire changer l'abstraction visuelle d'un objet. Une difficulté de conception des interfaces zoomable est d'éviter le *desert fog*, à savoir de créer une situation où l'utilisateur n'a pas d'indice pour s'orienter.

Interfaces multi-Résolution

La résolution peut avoir différentes signification en fonction de l'opérateur de transformation sur lequel elle s'applique. La variation de résolution de l'abstraction de données permet d'afficher différent niveaux de détails en fonction d'un zoom de l'utilisateur [STH03] (tels que le filtrage ou agrégation). L'abstraction visuelle permet de générer des données avec différents types de représentation et de précision, comme sur une carte géographique différente en fonction de l'altitude du point de vue. Enfin la résolution du rendu indique le niveau de détail, de crénelage, ou de rafraichissement de l'image qui est générée. La coordination de ces types de résolutions permettent différentes stratégies d'affichage permettant d'adapter

FIGURE 3.12 – Le système PHOTOMESA [Bed01] propose une interface de navigation zoomable dans des images disposées dans un plan.

la vue en fonction de contraintes techniques (telles que le framerate) ou cognitives (comme diminuer la surcharge visuelle).

3.3.3 Autres types d'interfaces

Les types d'interfaces que nous avons identifié peuvent également être combinées entre elles. D'autres types d'interfaces existent telles que les interfaces *textuelles*, et d'*interaction en 3D* (mondes virtuels). Les interfaces *textuelles* offrent une ligne de commandes qui permet de dialoguer avec le système, mais nécessite la mémorisation des commandes et son plutôt destinées aux experts [DFA04]. Les *interfaces d'interaction 3D* permettent de représenter des métaphores de navigation connues (maison, salle de réunion) et nécessitent des dispositifs d'interactions spécifiques afin de pouvoir contrôler tous les degrés de liberté efficacement.

Les interfaces de type *"post-wimp"* [vD97], apparues plus récemment, proposent de s'affranchir des widgets classiques comme nous l'avons vu précédemment, et d'offrir de nouvelles formes de navigation. Par exemple le logiciel BRYCE 3D [Bry10] (figure 3.13) est un logiciel grand publique qui ne comporte ni menu, ni icône, et permet une navigation intuitive au moyen de la souris. Il existe d'autres types d'interfaces : interfaces tangibles, la réalité augmentée, les environnements pervasifs, l'informatique ubiquitaire, ambiante (capteurs), et les mondes virtuels.

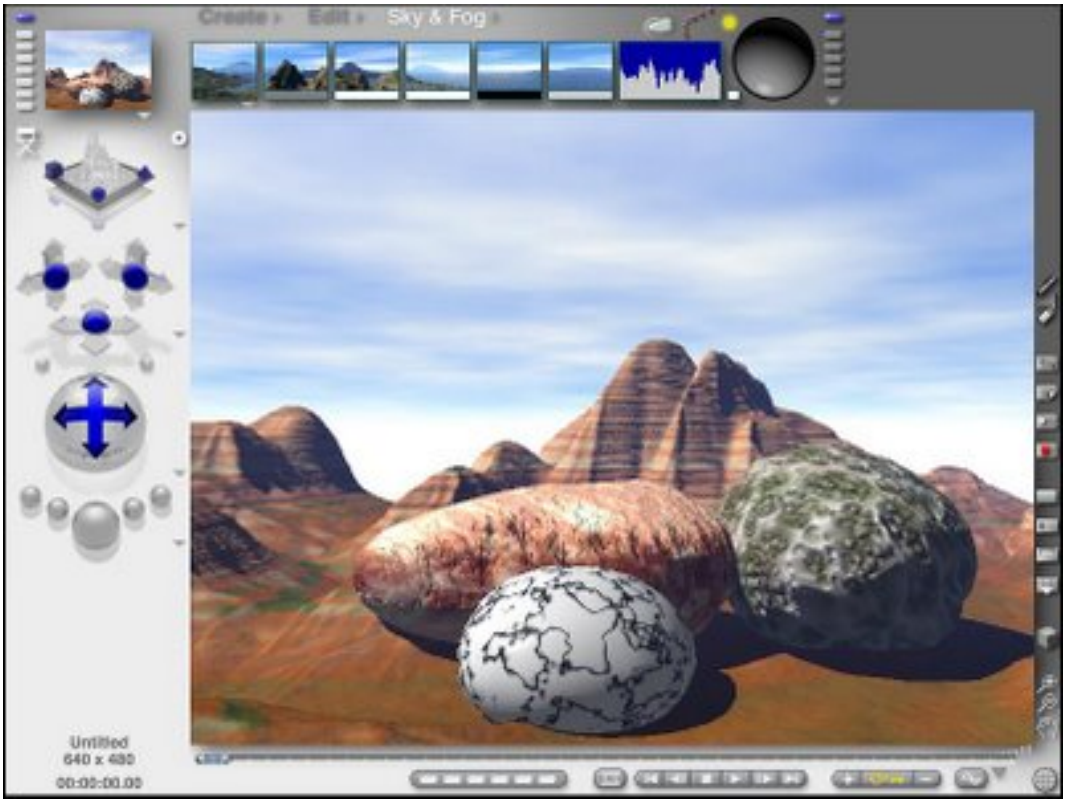


FIGURE 3.13 – Capture d'écran de l'interface de BRYCE 3D [Bry10] un logiciel de création 3D de type Post-Wimp.

3.4 Synthèse du chapitre

Dans ce chapitre nous nous sommes intéressés aux opérateurs de transformation de l'espace de design de la visualisation d'information. Le domaine de l'interaction étant vaste, nous avons limité notre étude aux techniques classiques d'interaction, qui permettent des requêtes dynamiques, progressives ou composées, au moyen de widgets. Les interfaces graphiques permettent d'explorer et de coordonner les rendus des données, et permettent d'exprimer des requêtes en fonction du point de vue ou d'éléments sélectionnés dans les vues. L'utilisateur perçoit les résultats issus d'une requête ou de la mise à jour du système, ainsi que la rétroactivité du système qui indique la prise en compte de ses actions.

Dans le chapitre suivant (Chapitre 4) nous allons étudier les techniques d'exploration d'information afin de permettre à l'utilisateur de réaliser un parcours *cohérent* dans l'espace de transformation des données et des vues, pour réaliser une tâche complexe.

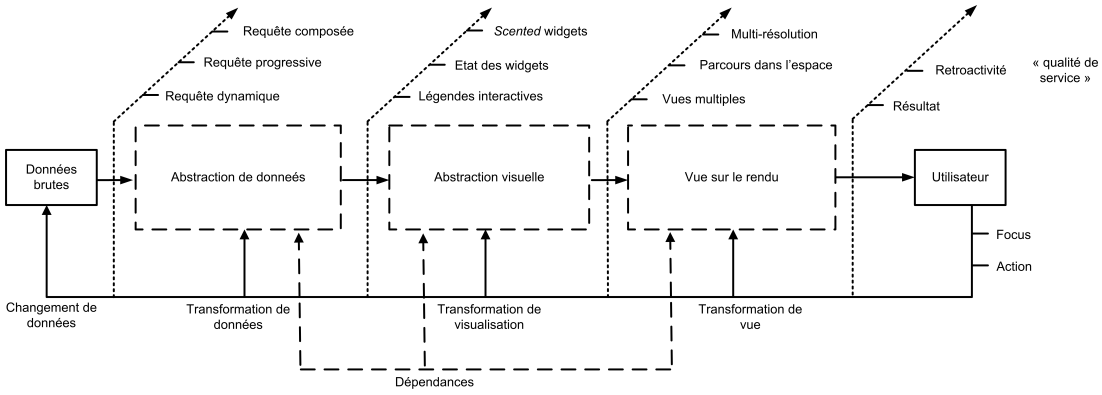


FIGURE 3.14 – Synthèse de l'espace d'interaction visuelle.

Chapitre 4

Stratégies d'exploration visuelle d'information

Sommaire

4.1 Introduction

L'espace de design de la visualisation (Chapitre 2) et les interactions de l'utilisateur qui se positionnent dedans (Chapitre 3), permettent une combinaison très importante de vues possibles. Un utilisateur qui souhaite explorer un jeu de données, afin de réaliser ou simplifier une tâche (telle que la recherche d'une information, la recherche de corrélation entre des informations, voir une tâche inconnue à l'avance[1]), possède de nombreuses alternatives d'exploration visuelles avant de converger vers la vue souhaitée. Les vues se construisent au fil d'actions/perceptions, et les questions suivantes sont à se poser : quel est le point de départ comme la première donnée ou dimension à choisir ? Comment parcourir efficacement les valeurs des données ? Comment évaluer la qualité du résultat ou de la démarche ? Comment rejouer cette démarche ? Comment la partager de manière collaborative avec un groupe de recherche plus ou moins expert et pluridisciplinaire ?

Par exemple GeoVISTA [TG02] est un système d'exploration visuelle qui offre des vues multiples coordonnées de données multidimensionnelles (capture d'écran figure 4.1). L'utilisateur peut sélectionner un ou plusieurs éléments (valeurs, dimension) et cette sélection sera propagée sur toutes les vues, tout en gardant le contexte, à savoir les données déjà visibles (mais rendues floues par exemple pour montrer leur non-sélection). L'utilisateur est certes guidé dans l'exploration des différentes transformations visuelles et de données, mais ne l'est pas dans sa démarche intellectuelle.

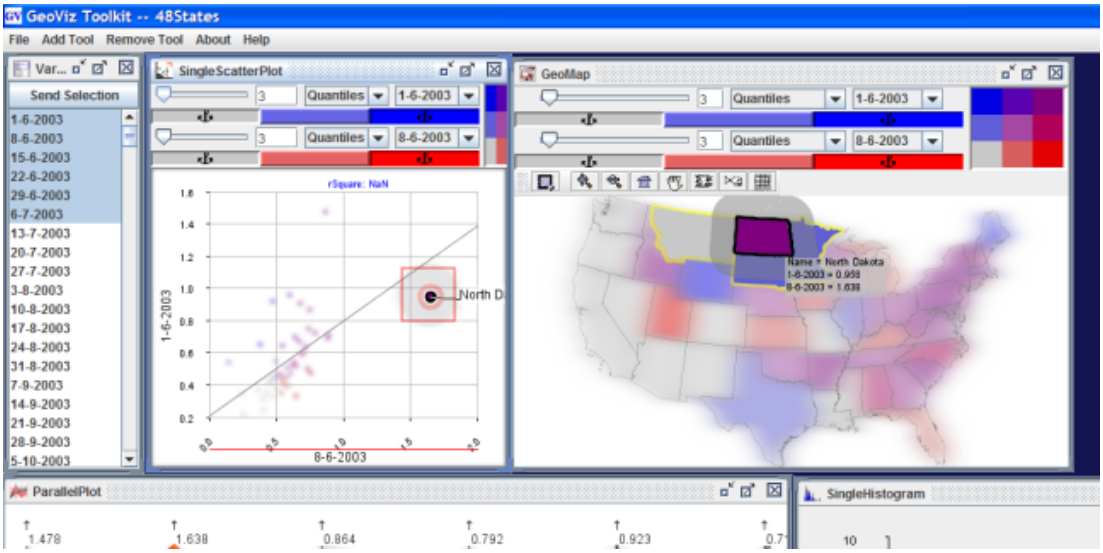


FIGURE 4.1 – Capture d'écran de l'interface du système GeoVISTA [TG02].

Dans ce chapitre nous allons tenter d'identifier les différentes tâches et stratégies d'exploration visuelle par l'utilisateur (section 4.2), et ensuite nous allons nous intéresser aux méthodes automatiques et sociales d'exploration visuelle (section 4.3).

1. Tels les scénarii hypothétiques "et si .. ?" qui permettent de poser et tester des hypothèses, mais ne permettent pas d'identifier à l'avance un modèle utilisateur et de tâche.

4.2 Tâches et stratégies utilisateur de navigation visuelle

La technique la plus simple -voire simpliste- pour résoudre un problème serait de parcourir systématiquement toutes les abstractions de données (valeurs possibles à sélectionner ou filtrer), ainsi que les abstractions visuelles (encodages, position, etc.) et enfin les vues sur le rendu (zoom, rotation, etc.), afin d'obtenir le résultat recherché. Cette démarche n'est pas satisfaisante car elle offre une combinatoire importante et ne pourrait être réalisable en un temps humain raisonnable. Une démarche structurée et méthodique doit donc être mise en place, afin d'éviter (ou du moins de limiter) toute forme d'opportunisme et de hasard dans la phase d'exploration visuelle. Nous allons nous intéresser aux tâches (section 4.2.1) et aux séquences de tâches (section 4.2.2) qui permettent d'assister la navigation visuelle.

4.2.1 Tâches de navigation visuelle

Un problème ou une tâche de "haut niveau" (comme l'identification de tendances, la recherche d'une corrélation, etc.) peut être difficile à appréhender par l'utilisateur. Il est donc nécessaire d'effectuer une décomposition en sous-problèmes [DFA04], réalisables facilement au moyen d'interactions (section 3.2) ou de tâches élémentaires [AES05]. L'utilisateur devra ensuite rassembler ou coordonner les résultats afin de résoudre la tâche initiale.

Parmi les techniques d'interactions de bas niveau, les BVI (*Basic Visualization Interaction*) [CR96] sont une catégorisation hiérarchique en trois classes : les opérations graphiques (*graphical operations*), afin de changer l'apparence de la visualisation, les opérations sur les données (*data operations*), afin de manipuler les données, et enfin les opérations ensemblistes (*set operations*), pour créer et manipuler des objets. Les deux derniers types d'interactions agissent sur l'abstraction de données [Chi00] et peuvent donc nécessiter la mise à jour de celle-ci.

[Shn96] propose une collection plus précise de sept types de tâches de visualisation :

1. **Vue globale (*Overview*) :** *obtenir une vue globale de la collection complète.* La vue globale peut être une abstraction sur les données, qui ne modifie pas les données mais qui extrait une composante (structure, relation, dimension, etc.) et offre un cadre de réflexion propice à la prise de décision.
2. **Zoom (*Zoom*) :** *zoomer sur un objet intéressant afin d'obtenir plus de détails.*
3. **Filtrer (*Filter*) :** *filtrer les éléments non-intéressants.* Notamment au moyen de widgets interactifs, tels que les requêtes dynamiques [AS94b].
4. **Détails à la demande (*Details-on-demand*) :** *sélectionner un élément ou un groupe d'éléments et obtenir les détails quand il y en a besoin.*
5. **Mise en relation (*Relate*) :** *visualiser les relations entre les éléments.* Par exemple au moyen de juxtaposition de vues différentes sur les données, et une coordination entre les vues.
6. **Historique (*History*) :** *garder un historique des actions afin de permettre les refaire (*undo*), rejouer et raffinement progressif.* Par exemple le système GRASPARC [BPW+93] qui construit un arbre d'historique au fil des différentes interactions.
7. **Extraire (*Extract*) :** *permettre l'extraction de sous-collections et des paramètres des requêtes.*

Il existe de nombreux autres types de tâches, mais qui possèdent une terminologie propre, ce qui limite leur comparaison [YKSJ07]. Des tâches plus spécifiques aux structures de données peuvent être énumérées comme pour les graphes [LPP+06]. Ces tâches spécifiques ne sont par contre pas généralisables, et donc pas extensibles aux autres structures de données (mais les tâches génériques [AES05, Shn96] peuvent elles s'appliquer aux graphes).

4.2.2 Stratégies de navigation visuelle

Les stratégies de navigation permettent d'agencer les tâches de manière cohérente, soit parce qu'elles sont dépendantes entre elles, soit parce qu'il faut fournir à l'utilisateur une suite logique d'un point de vue cognitif. L'utilisateur doit pour sa part être activement attentif et essayer d'extraire des motifs au travers de variables graphiques telles que les regroupements, la taille ou les couleurs. A noter que les utilisateurs experts, qui possèdent déjà une bonne maîtrise des abstractions de données et visuelles, peuvent explorer plus rapidement l'espace et éluder des étapes.

La stratégie de navigation la plus souvent citée est le *visualization-information-seeking mantra* de [Shn96], issu d'années d'expériences en développement d'interfaces. Ce mantra propose une stratégie en trois étapes : *"Overview first, zoom and filter, then details-on-demand"*, qui reprend une partie des tâches énumérées dans la partie précédente. Démarrer par une vue globale n'est cependant pas le point de départ universelle. Par exemple dans le cas d'un graphe [LPP+06] il peut être pertinent de réaliser une démarche liée à la topologie du graphe, comme démarrer d'un noeud et ensuite d'explorer ses voisins. La connaissance sera ainsi construite par associativité.

Les stratégies de navigation visuelle peuvent être également locales à une vue. Par exemple l'intégration de *"scented widgets"* (section 3.2.1) ou la prévisualisation de résultats (requêtes dynamiques et progressives), permettent de donner un aperçu de l'étape suivante, telle que la quantité de résultats attendus. L'animation (section 2.2.1) est également une possibilité d'exploration afin d'éviter une série d'interactions répétitives (au détriment de l'interactivité, mais qui permet à l'utilisateur de se concentrer sur la visualisation). GapMinder [Gap10] (figure 4.2) permet de visualiser automatiquement l'évolution des données graphiques au fil des années (le temps est donc encodé en une variable *physique*, et non visuelle). Une approche mixte est l'interface Google Insights for Search [Goo10c] (en bas à droite sur la figure 4.3)[2] qui permet la lecture d'une vue seulement (la carte choroplèthe) ce qui laisse le reste de l'interface interactif et permet à l'utilisateur d'obtenir des détails à la demande.

Conclusion

Les stratégies de navigation peuvent soit être inclues dans les systèmes dès leur design, au moyen de motifs de conception [Che04], soit un système reste ouvert à tout type de stratégie [STH02] *"Often the path of exploration is unpredictable, and thus analysts need to be able to rapidly change both what data they are viewing and how they are viewing that data."*. A noter que la résolution des systèmes et leur taille influencent grandement la perception et l'efficacité qu'en

2. http://www.google.com/insights/search/#q=election%20time%2Cterror&geo=US&cmpt=q

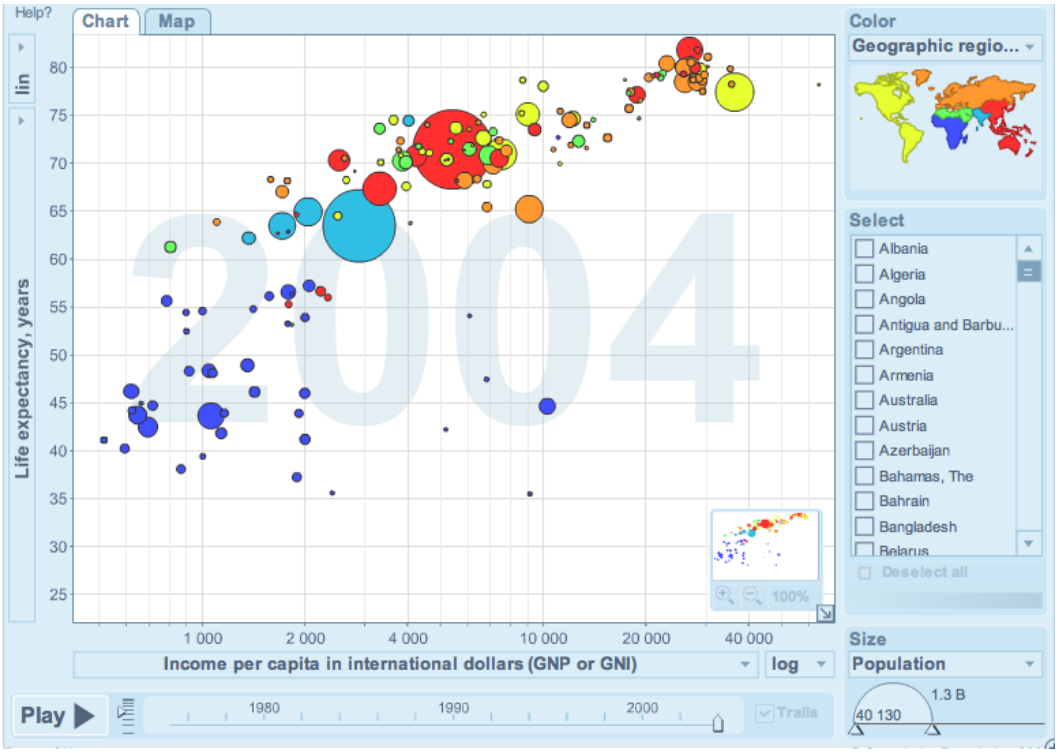


FIGURE 4.2 – Capture d'écran de l'interface GAPMINDER [Gap10].

a l'utilisateur [YHN07]. Par exemple, même si en théorie tout jeu de donnée est visible avec un nombre infini de pixel, la réalité des écrans matriciels actuels et leur résolution variable limitent les possibilités d'affichage. Les stratégies de navigation doivent donc tenir compte de ces paramètres externes, ainsi que de paramètres liés à l'utilisateur [TM04a] : préférences, environnement physique, contexte social et culturel, etc.

4.3 Méthodes automatiques et sociales d'explorations visuelles

Nous allons étudier la complémentarité des méthodes automatiques (analyse statistique, analyse exploratoire de données) qui peuvent assister l'utilisateur dans la réduction de l'espace de données (section 4.3.1). Ces techniques permettent également d'assister l'utilisateur dans une démarche de fouille visuelle interactive de données (section 4.3.2). Nous allons pour finir nous intéresser aux collaborations asynchrones et à l'analyse sociale de données (section 4.3.3) qui permettent aussi la réduction de l'espace de données et de point de vues sur celles-ci.

4.3.1 Couplage avec les méthodes statistiques et de fouille de données

L'utilisation de méthodes *automatiques* d'exploration des données, à partir de méthodes statistiques et d'analyse exploratoire de données [Tuk77] pourrait être une solution afin de remplacer l'utilisateur dans la prise de décision. Un contre-exemple classique d'utilisation

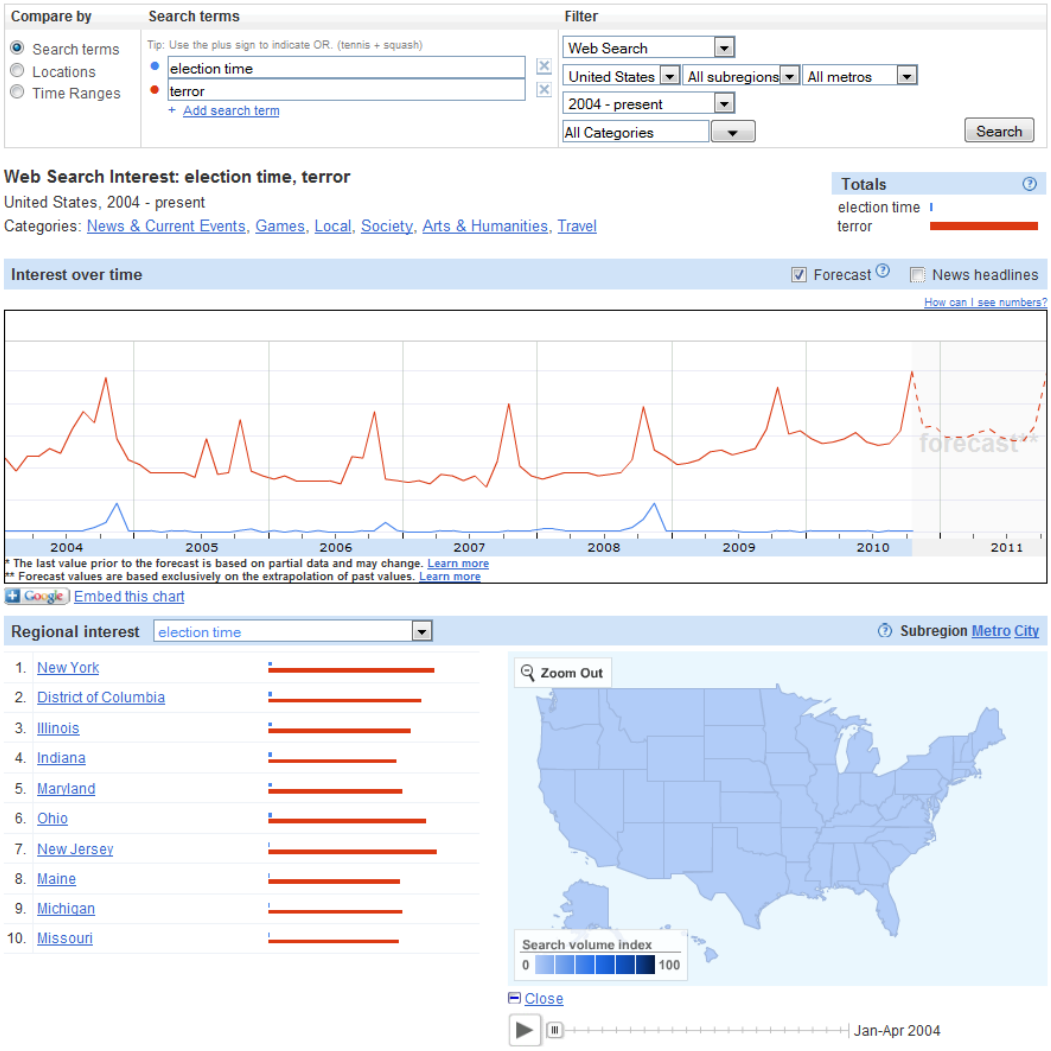


FIGURE 4.3 – Capture d'écran de l'interface de GOOGLE INSIGHTS FOR SEARCH [Goo10c].

exclusive de ces méthodes est le *Anscombe's Quartet* qui est un jeu de données de douze entrées [3] composées de quatre attributs chacune, et qui possèdent des propriétés statistiques *identiques* : la valeur moyenne en $x = 9.0$, la valeur moyenne en $y = 7.5$, l'équation de régression linéaire $y = 3 + 0.5x$ et la somme d'erreur quadratique moyenne $= 110.0$. Il serait logique d'en déduire que les données sont identiques, or la visualisation des données est différente (figure 4.4). Cette visualisation permet de détecter en un coup d'oeil des valeurs de données aberrantes (*outliers*), qui n'ont pas été mises en avant par les méthodes statistiques.

Les outils d'analyse statistique ont une longue histoire de visualisation sous forme de graphiques, comme les histogrammes ou les séries temporelles. Des outils comme R [R10] (avec le module GGPLOT [4] notamment) et MATLAB [MAT10] permettent de réduire, à partir de modèles, l'espace de données ou de trouver des corrélations qu'il n'aurait pas été possible de trouver en un temps humain raisonnable sans l'expertise nécessaire [KMSZ06]. Ces

3. les données sont disponibles sur http://en.wikipedia.org/wiki/Anscombe%27s_quartet
4. http://had.co.nz/ggplot2/

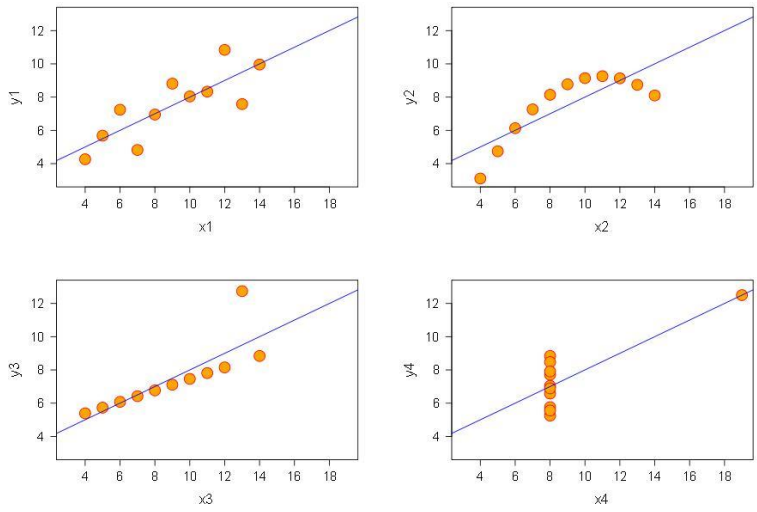


FIGURE 4.4 – Représentation graphique du *Anscombe's Quartet*.

modèles permettent donc d'une certaine façon d'automatiser des opérations d'abstraction de données, qui auraient normalement été réalisées par un statisticien.

Ces outils intègrent des méthodes classiques de réduction de dimensions, telles que la méthode d'Analyse en Composantes Principales[5] (ACP) basée sur le regroupement de variables corrélée, afin de faciliter leur visualisation[6].

Le système KNIME [Kni10] (figure 4.5) est une plateforme d'analyse de données, à partir de laquelle l'utilisateur peut construire un flot de traitement de données et facilement comparer et modifier des algorithmes de fouille de données ou de classification. Le système WEKA [HFH+09] permet de réaliser des opérations d'apprentissage à partir des données, et permet aussi la classification d'éléments. Les résultats de ces outils peuvent être présentés sous forme de graphiques et de valeurs, mais aussi sous forme de rapports statistiques qui combinent plusieurs vues. Ces outils ne permettent cependant pas de réaliser un processus d'analyse visuelle à partir des visualisations, qui sont souvent statiques et non interactives.

Ces méthodes et outils reposent sur le principe d'associer d'un côté ce que l'homme sait bien faire (comme poser une hypothèse ou reconnaître des formes), et de l'autre ce que la machine sait bien faire (réaliser des vérifications ou effectuer rapidement un grand nombre de calculs répétitifs) [BL09].

4.3.2 Fouille visuelle interactive de données

Face à la capacité croissante de collecte et de stockage de données, et face à la limite des algorithmes de fouille de données (si le nombre de faux positifs est trop grand, ou si la tâche n'est pas connue à l'avance et si le problème est mal formulé (*ill-formulated*)) la prise en

5. http://fr.wikipedia.org/wiki/Analyse_en_composantes_principales
6. En effet, les données avec un grand nombre de dimensions sont difficiles à représenter dans leur intégralité, car il peut ne pas y avoir assez de variables graphiques disponibles (phénomène appelé *Curse of Dimensionality*, appliqué à la visualisation).

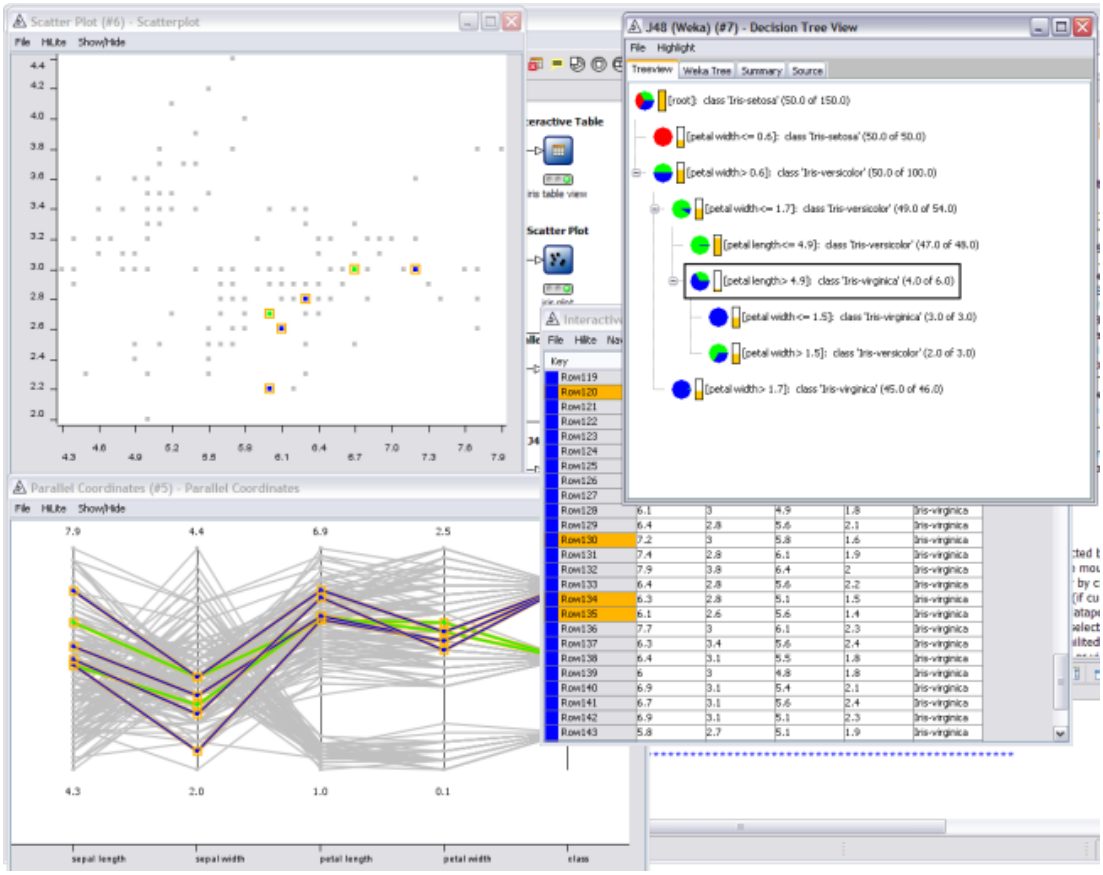


FIGURE 4.5 – Capture d'écran de l'interface de KNIME [Kni10] d'analyse de données.

compte de l'utilisateur dans le processus de décision est nécessaire, comme par exemple afin de guider l'exécution des algorithmes.

La fouille visuelle interactive de données (*Visual Analytics*[7]) est un domaine apparu récemment (2005) et qui a pour défi de combiner (entre autres, illustration sur la figure 4.6 d'autres champs impliqués) les méthodes de visualisation d'information, avec celles d'analyse automatique, d'apprentissage et de fouille de données, indispensables à la création de sens à partir de grandes quantités de données [KMSZ06] . Cette combinaison permet à l'utilisateur de réaliser toute forme d'hypothèse et de visualiser et interagir avec le résultat. Il peut ainsi réaliser un processus global de découverte de connaissances, qui se construit de manière itérative et par boucles de rétroaction dites de *"sens making loop"* [PC05] (figure 4.7).

Par exemple JIGSAW [SGL08] est un système permettant de supporter le raisonnement analytique au moyen de vues multiples sur les données (figure 4.8). Une hypothèse peut être réalisée sur une vue, et le résultat propagé sur les autres. Afin d'assister l'utilisateur dans ces étapes d'exploration, [SvW08] suggère de distinguer trois types de vues :

– **Vue sur les données (*data view*) :** représentation visuelle des données.

7. défini comme *"Visual analytics is the science of analytical reasoning facilitated by interactive visual interfaces"* [TC05]

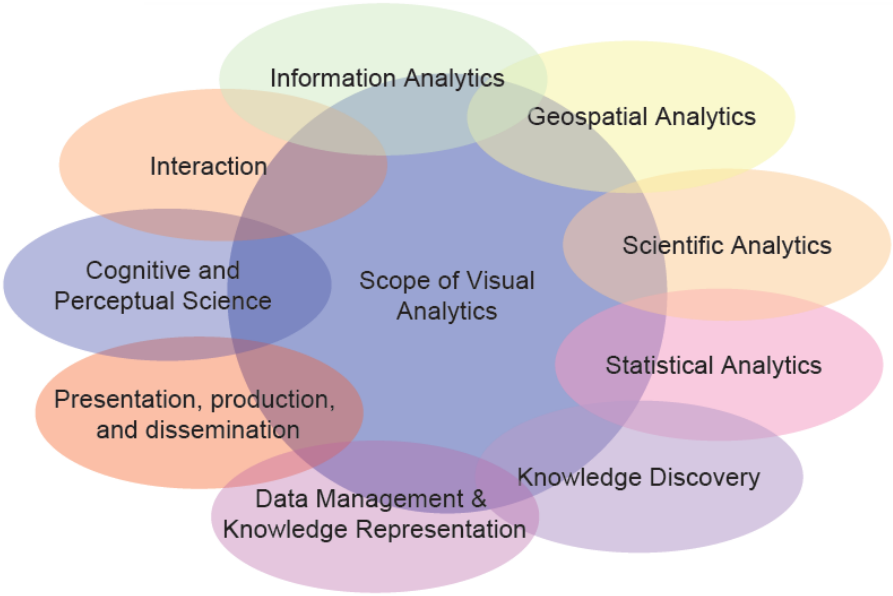


FIGURE 4.6 – Champs connexes aux *visual analytics* [KMSZ06]

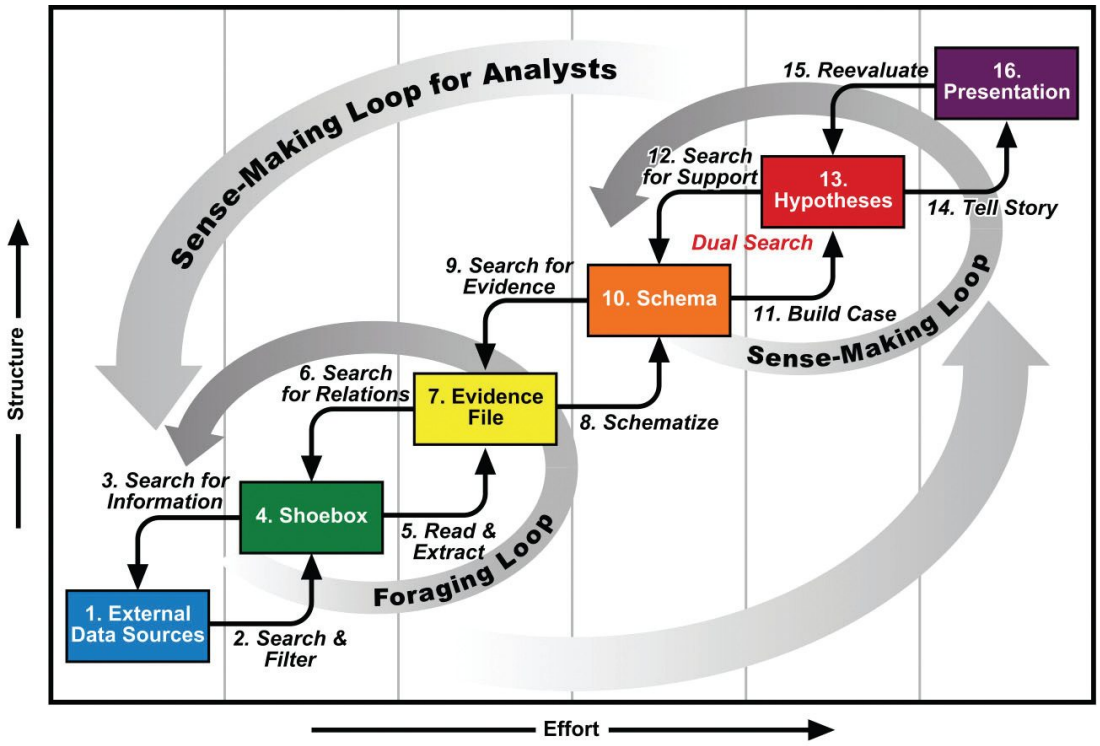


FIGURE 4.7 – Boucles itératives de création de sens (*"sens making loop"*) [PC05].

– **Vue de navigation (*navigation view*) :** représentation visuelle du processus d'exploration.

– **Vue sur les connaissances (*knowledge view*) :** représentation visuelle des artéfacts et

des liens causals.

Les techniques classiques d'exploration visuelles (comme celles mentionnées précédemment section 4.2.2) sont à adapter au raisonnement analytique. [KMSZ06] propose une stratégie d'exploration, basée sur le mantra de [Shn96] *"Analyse First - Show the Important - Zoom, Filter and Analyse Further - Details on Demand"*.

FIGURE 4.8 – Capture d'écran du système d'analyse visuel JIGSAW [SGL08].

4.3.3 Collaboration asynchrone et analyse sociale de données

La plus grande partie des découvertes scientifiques modernes a été réalisée de manière collaborative [WWB95]. La collaboration est une activité humaine de partage, d'annotation ou de discussion et qui peut donc avoir différentes temporalités [BR10] (figure 4.9). La collaboration est dite asynchrone si les acteurs ne sont pas présents en même temps, et peut être réalisée sur plusieurs lieux géographiques. La collaboration doit aussi être expliquée et motivée aux acteurs, afin qu'ils perçoivent un intérêt à y participer.

La collaboration permet une forme de navigation *sociale* [DC94] dans les données, à savoir une navigation guidée en fonction de celle d'autres personnes. En complément de la navigation *spatiale* (la navigation de données qui ont une composante géographique) et de la navigation *sémantique* (la navigation guidée à partir de visualisations).

Le design de la navigation *sociale* n'est selon [HA08] pas immédiat, car les recommandations actuelles de design ne le sont que pour un seul individu seul face à son écran. Visualiser le parcours d'autres individus implique d'afficher les opérations qu'ils sont réalisé, aussi bien dans l'espace d'abstraction de données, que dans celui d'abstraction visuelle ou dans l'espace de vues sur les données [HA08]. De manière générale, le design d'environnements visuels collaboratifs implique la séparation les tâches de travail, une modélisation avec une

TIME SCALE OF HUMAN ACTION

SCALE (sec)	SYSTEM	STRATUM
10^{7} 10^{6} 10^{5}		SOCIAL
10^{4} 10^{3} 10^{2}	Task Task Task	RATIONAL
10^{1} 10^{0} 10^{-1}	Unit Task Operations Deliberate Act	COGNITIVE
10^{-2} 10^{-3} 10^{-4}	Neural Circuit Neuron Organelle	BIOLOGICAL

FIGURE 4.9 – Différentes échelles de temps en fonction de l'activité humaine [BR10].

granularité plus fine, ainsi que la gestion de conflits [8] et le contrôle de versions [MB07].

Concevoir un système capable de se déployer et de toucher un maximum de personnes est un défi rendu possible au travers du WORLD WIDE WEB [WBW96]. La collaboration à grande échelle est donc aujourd'hui possible et à moindre coût. Le site internet SENSE.US développé par [HVW07] a mis en avant des collaborations asynchrones sur des représentations graphiques, servant de bases de discussion sur lesquelles ont été pointés des éléments intéressants (données, *outliers*, tendances, etc..) par les utilisateurs. La collaboration a été possible au moyen d'outils faciles à utiliser et ludiques, tels qu'une palette d'outils de dessin et de saisie de texte (figure 4.10). Ces annotations constituent une vue supplémentaire de connaissances, et peuvent être reliées à d'autres visualisations au moyen de liens hypertextes.

Le site collaboratif MANYEYES [WKM07] d'IBM est ouvert à tout utilisateur connecté à internet, qui peut y transférer un jeu de données [9], et ensuite le visualiser avec différentes représentations visuelles (figure 4.11). Des widgets en Java permettent de changer l'encodage visuel des vues et de changer de point de vue (zoomer, déplacement dans le plan). Les interactions possibles sont cependant limitées, et l'export des vues n'est pas possible (hors

8. Ce qui est classique dans un contexte de gestion de travail collaboratif.
9. Qui doit respecter un format générique (de type tabulaire classique) mais ce qui demande toute de même à l'utilisateur de prétraiter ses données et de réfléchir aux dimensions qu'il souhaite visualiser..

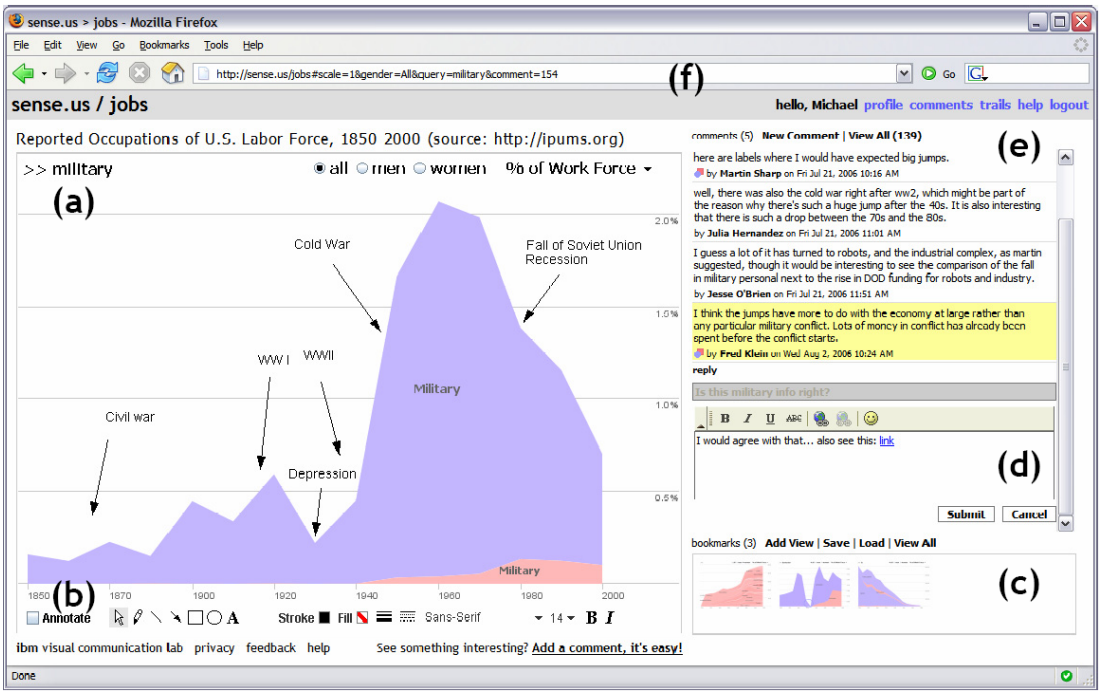


FIGURE 4.10 – Capture d'écran du site web SENSE.US [HVW07].

mis en réalisant une capture d'écran..). L'unique moyen de partage est au moyen d'une URL (*Uniform Resource Locator*), qui est un moyen courant d'échange de vues [HHC+08], et qui pourra ensuite être partagée ou inclus dans un blog.

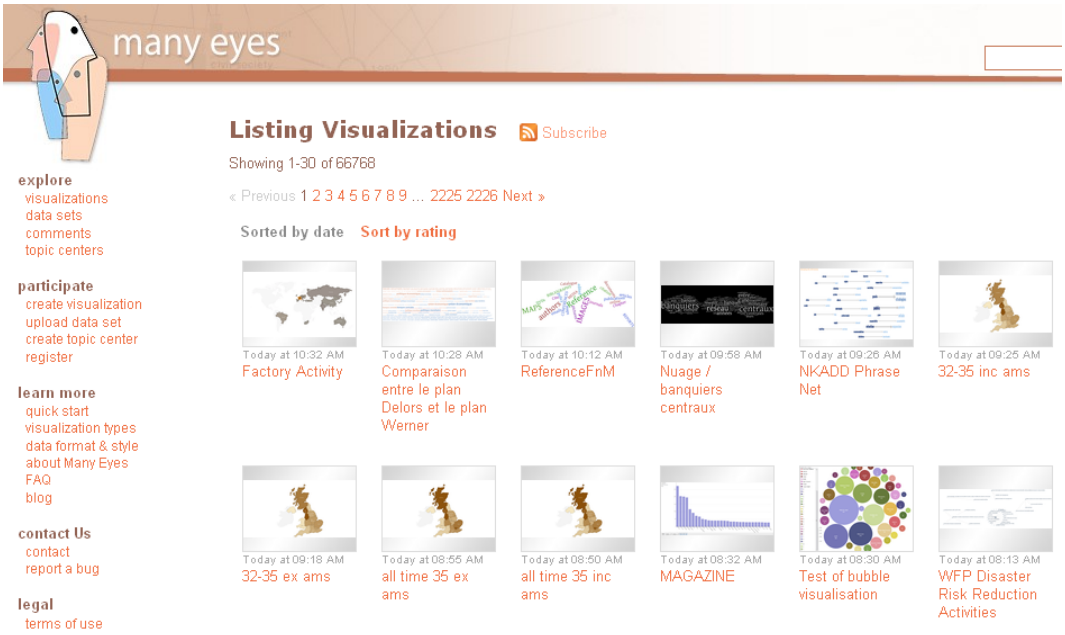


FIGURE 4.11 – Capture d'écran du site web MANYEYES [WKM07].

4.4 Synthèse du chapitre

Les stratégies d'exploration visuelle d'information dépendent de nombreux paramètres tels que la tâche et le contexte de l'utilisateur. L'équilibrage entre un système spécifique à chaque tâche (mais qui implique la conception de nombreux systèmes) et un système générique (mais qui doit alors assister l'utilisateur ou celui-ci doit avoir une démarche très structurée) est donc à trouver par le concepteur de l'application. L'exploration peut être guidée par des méthodes automatiques qui permettent une exploration (par réduction notamment) de l'espace de données, et l'utilisateur peut réaliser une démarche de fouille visuelle interactive de données par interactions successives. La collaboration et l'analyse sociale de données sont également des méthodes d'aide à l'exploration de l'espace de données et permettent également d'explorer l'espace d'encodage visuel et de point de vue par partage et annotation de visualisations.

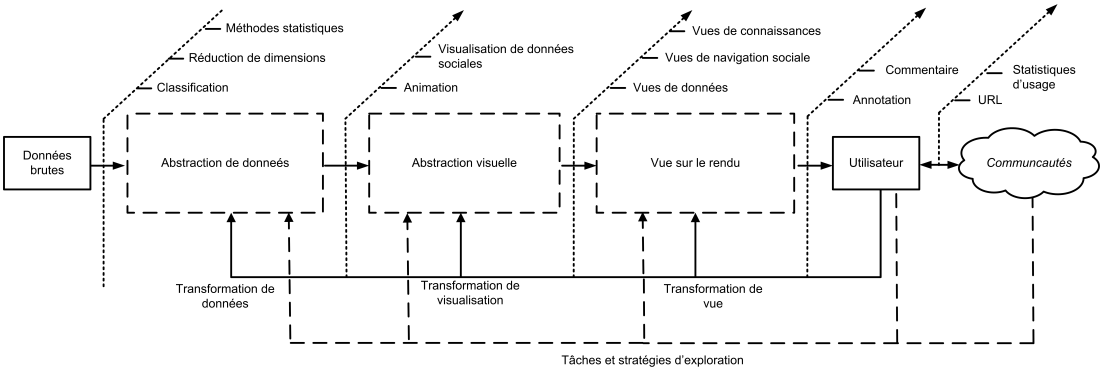


FIGURE 4.12 – Synthèse de l'espace de design de l'exploration visuelle d'information.

Deuxième partie

Architecture VizOD et applications

Chapitre 5

Architecture VizOD

Sommaire

5.1 Principes de l'architecture VizOD et définitions

5.1.1 Rappel des objectifs

Nous avions, dès l'introduction de nos travaux, cerné trois objectifs (section 1.3) :

- **Identifier les dimensions de l'espace de design des représentations visuelles interactives, ainsi que leurs dépendances.**
- **Proposer un cadre complet de conception, ainsi qu'une architecture opérationnelle**.
- **Développer des applications à partir de cette architecture**.

Ce chapitre est consacré au deuxième objectif et nous proposons donc un cadre général de conception et une architecture, basés sur l'espace de design que nous avons identifié dans la partie I. Dans le chapitre suivant (chapitre 6) nous détaillerons les applications que nous avons développé à partir de cette architecture.

5.1.2 Proposition et motivation

Proposition

Suite à notre identification de l'espace de design de la visualisation d'information interactive (partie I) nous introduisons VizOD (pour *VIsualiZation-On-Demand* [1]), un cadre de décomposition de cet espace en trois parties complémentaires (figure 5.1 et détail en annexe A.1) :

- **Des *services* et *flots de visualisation*** (section 5.1.3) afin de créer et partager les vues *statiques* sur les données, accessibles via un réseau. Les services sont les étapes de traitement de données du modèle de référence (telles que l'*abstraction de données*, l'*abstraction visuelle* ou le *point de vue sur le rendu*) et les flots sont la composition de ces services afin d'assembler un modèle de référence complet et produire une visualisation.
- **Un *profil applicatif*** (section 5.1.4) est l'application cliente avec laquelle l'utilisateur interagit. Cette application est composée de widgets et d'interfaces graphiques, qui permettent de construire une requête et d'afficher le résultat visuel de celle-ci (issu des flots de visualisation). Le profil applicatif est garant de la réactivité et de la rétroactivité suite aux interactions de l'utilisateur, ainsi que de réaliser les animations de transition entre les vues.
- **Un *graphe de scène*** (section 5.1.5) coordonne les flots de visualisation en fonction des interactions de l'utilisateur.

Motivation

Notre cadre de décomposition part du constat établi dans l'état de l'art qu'une interface visuelle interactive est complexe à plusieurs points de vue. D'un point de vue *conceptuel* car les capacités de l'utilisateur, les tâches à réaliser et son environnement peuvent varier. D'un point de vue *technique*, car le traitement des données doit être réalisé dans son ensemble, de la source jusqu'au rendu et fait appel à des compétences techniques et à des acteurs (utilisateurs,

1. Qui s'inspire de la VoD (*Video-on-Demand*) où la vidéo n'est plus liée à un support physique de stockage, ni même un fichier, mais devient un service.

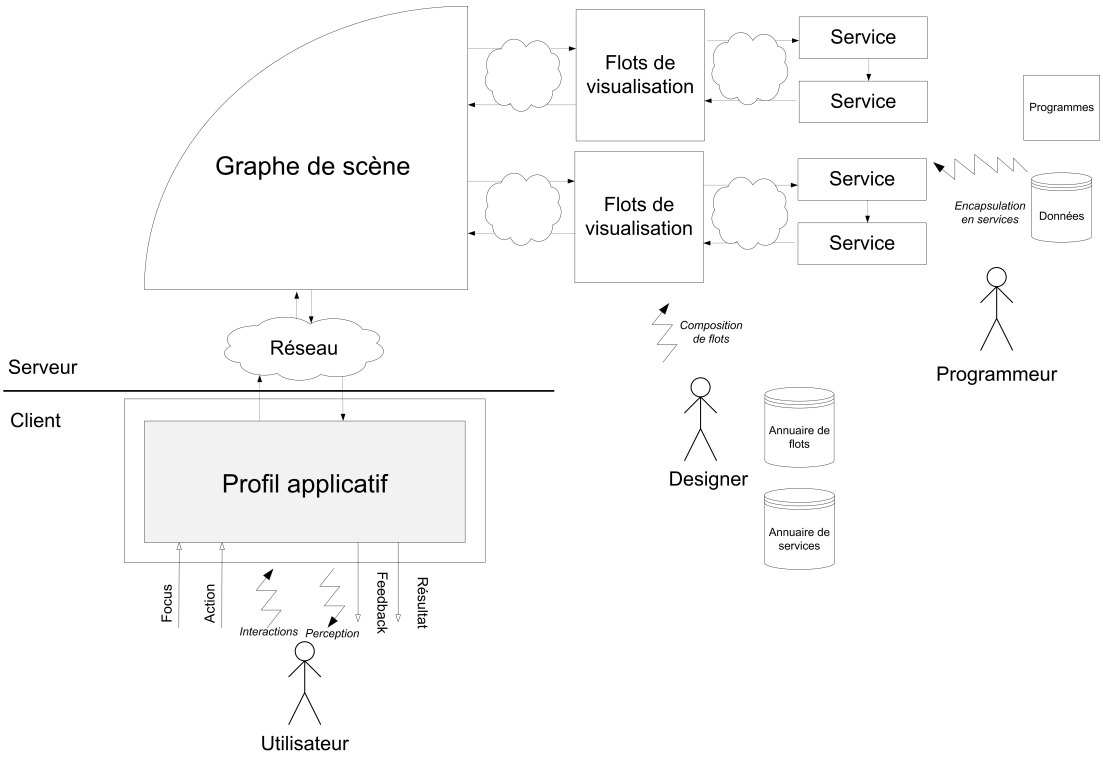


FIGURE 5.1 – Aperçu de l'architecture VIZOD.

designers et programmeurs) variés. Enfin du point de vue de l'*évaluation*, car celle-ci nécessite une implémentation fidèle (notamment dans le cas de la navigation sociale) et rapide, tout en permettant des variations de design, pour être validée.

Implémentation en architecture "3-tiers"

Le cadre de décomposition que nous avons proposé permet la mise en réseau de manière indépendante des services et des flots de visualisation. Ceux-ci sont des données ou des traitements sur les données indépendants (dans leur exécution), et qui doivent être dynamiquement interchangeables. Le profil applicatif qui interagit avec ces flots ne le fait plus dans le cadre d'une simple architecture client/serveur, mais dans le cadre du domaine général des architectures distribuées, en particulier des architectures "3-tiers"[2]. Ces architectures sont composées de trois couches successives qui communiquent entre elles de manière linéaire (figure 5.2). La couche de *présentation* contient les fonctions de l'interface et d'affichage les données. La couche d'*application* contient toutes les fonctionnalités propres au domaine d'une application (tel un noyau fonctionnel). Enfin la couche de *données* est constituée d'une base de données ou d'un système de fichier.

Ce type d'architecture permet de faire communiquer entre eux différents systèmes hétérogènes (comme un navigateur, des services web, des bases de données disponibles sous forme d'APIs, etc.). La séparation entre l'application et la présentation permet d'isoler les étapes de traitement des données et de présentation. La différence avec les architectures

2. http ://en.wikipedia.org/wiki/Multitier_architecture

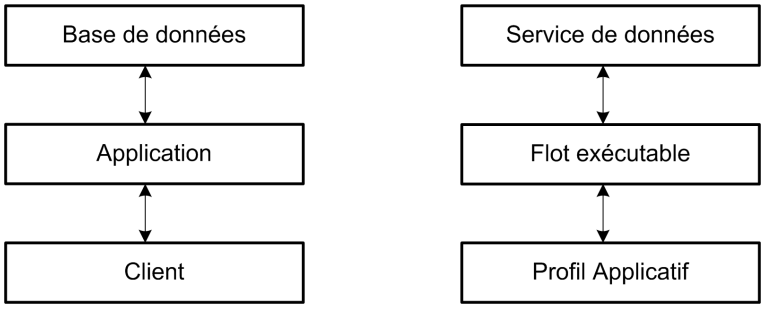


FIGURE 5.2 – Correspondance entre une architecture 3-tiers (à gauche) et VIZOD (à droite).

classiques telles que MVC [KP88] ou PAC [Cou87], est que celles-ci sont destinées au design d'une application résidant un seul système et implique un *couplage fort*. La séparation de la présentation et de l'application dans les architectures "3-tiers" est dit de *couplage faible* car elles échangent peu de données, et le font avec une interface uniforme. Une autre différence avec les architectures classiques (notamment avec MVC) est que la mise à jour des données n'est pas directe depuis les données vers la vue, mais doit passer par l'intermédiaire de la couche d'application (il s'agit d'un modèle linéaire et non pas triangulaire).

Nous allons maintenant détailler les trois parties de notre architecture.

5.1.3 Services et flots de visualisation

Définition 2 (Services de visualisation) *Un service de visualisation est un programme principalement défini par des entrées et des sorties, et qui possède des propriétés fonctionnelles et non-fonctionnelles.*

Un service peut être tout programme informatique exécutable, un script, une base de données ou un système de fichiers (figure 5.3). Un service peut être exécuté en local ou à distance et permet de réaliser une ou plusieurs étapes de traitement de données du modèle de référence. Il est indépendant d'autres services dans son exécution, mais peut communiquer avec d'autres services par messages. Les entrées et les sorties des services possèdent une description qui peut être de type MIME *(Multipurpose Internet Mail Extensions)* [BF93], ainsi que des extensions de formats de fichiers classiques (JPEG, SVG, etc.) ou des formats génériques de type tabulaire. Les services se caractérisent selon deux types de propriétés :

- **Propriétés *fonctionnelles* :** interfaces et fonction du service.
- **Propriétés *non-fonctionnelles* :** temps moyen d'exécution, temps d'accès, disponibilité du service, etc.

Définition 3 (Flots de visualisation) *Un flot de visualisation est la composition d'un nombre fini de services. La composition de services est l'association de la ou des sorties d'un service avec la ou les entrées d'un autre service.*

La composition de services permet d'assembler un modèle de référence complet et d'aboutir à une visualisation. Les entrées ou sorties de certains services peuvent être optionnelles et donc ne pas nécessiter de connexion à d'autres services (comme il existe des paramètres optionnels dans des programmes ou fonctions).

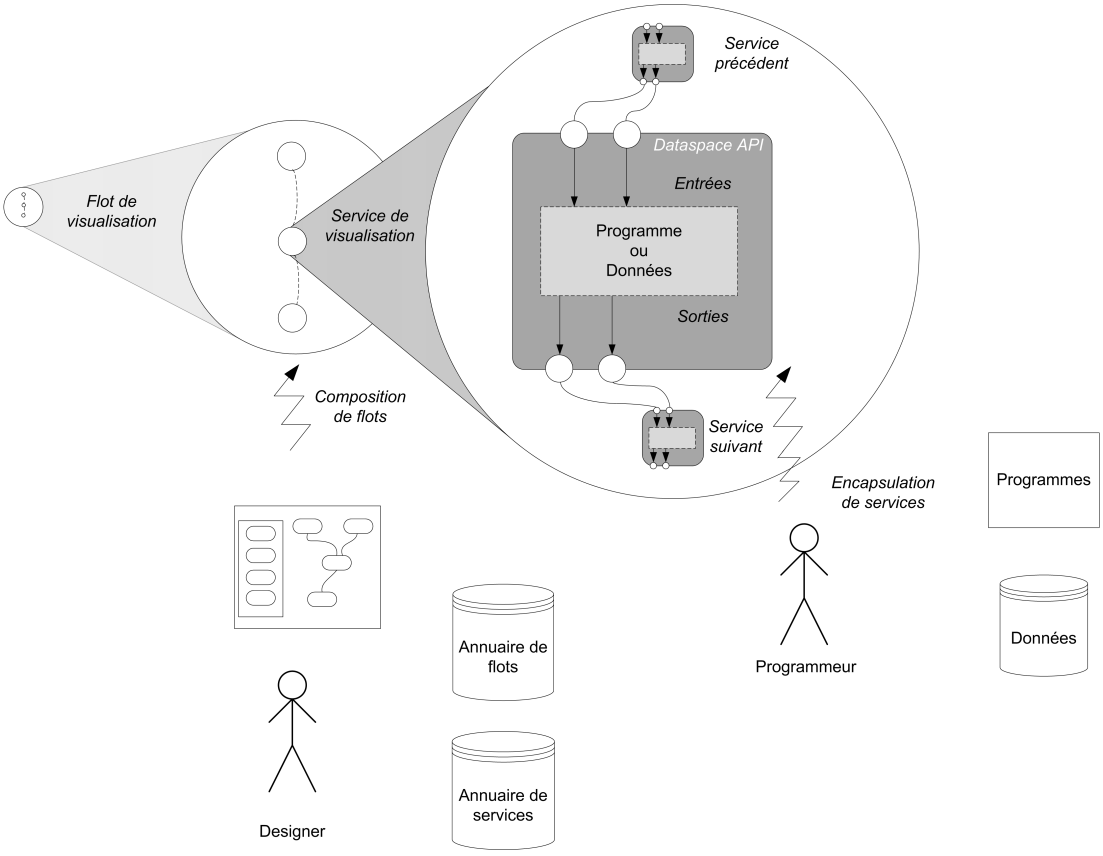


FIGURE 5.3 – Schéma détaillé de services et flots de visualisation dans l'architecture VIZOD.

Définition 4 (Services et flots exécutables) *Un service et un flot de visualisation sont dits exécutables s'ils peuvent être interprétés par un profil applicatif.*

Les flots exécutables permettent de créer une vue sur les données, qui pourra être manipulée et affichée avec un profil applicatif. Par exemple un flot de visualisation dont la sortie est au format JPEG ou SVG sera dit exécutable avec les profils applicatifs pouvant manipuler ces formats (comme le logiciel de traitement d'image GIMP [Pro10b]).

5.1.4 Profil applicatif

Définition 5 (Profil applicatif) *Un profil applicatif est une application exécutée côté client, à l'interface entre l'utilisateur et les flots exécutables.*

La philosophie du profil applicatif repose sur l'hypothèse qu'un programme peut intégrer une ou plusieurs vues générée par un ou plusieurs flots exécutables (figure 5.4). Un profil applicatif est par exemple tout type de logiciel qui peut interpréter un fichier dont le format est celui de la sortie d'un flot (GIMP [Pro10b] avec les formats JPEG ou SVG, un navigateur web avec le format HTML, MICROSOFT EXCEL [Mic10a] avec un format XLS). Le flot exécuté peut contenir aussi bien des données, qu'une abstraction visuelle ou un point de vue sur les données. Un profil applicatif peut inclure plusieurs flots exécutables (comme par exemple une page web HTML (qui est un flot) peut contenir la balise `<IMG>` (un flot supplémentaire,

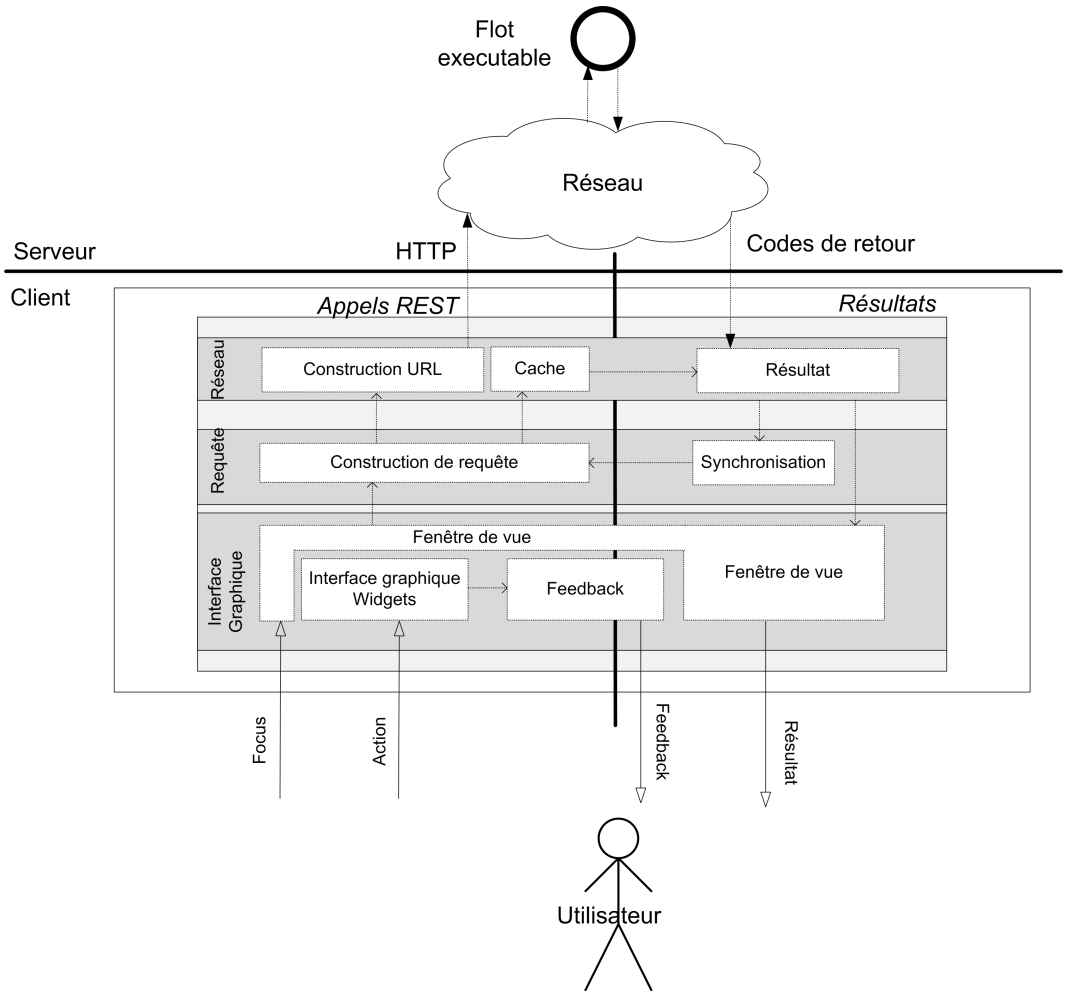


FIGURE 5.4 – Schéma détaillé d'un profil applicatif dans l'architecture VIZOD.

dans ce cas une image à exécuter par le navigateur)).

Un profil applicatif peut être décomposé en quatre modules :

- **Interface graphique :** regroupe les techniques d'interaction (section 3.2) telles que les widgets (section 3.2.1) et les interfaces graphiques (section 3.3) afin d'afficher les flots exécutables. Ce module gère les entrées (focus et actions utilisateurs), ainsi que les sorties (rétroactivité et résultat).
- **Composition de requête :** permet de composer la requête de l'utilisateur et de synchroniser les vues.
- **Réseau :** prise en charge des communications réseau et gestion du cache des données (en cas de requêtes similaire ou de coupure réseau).

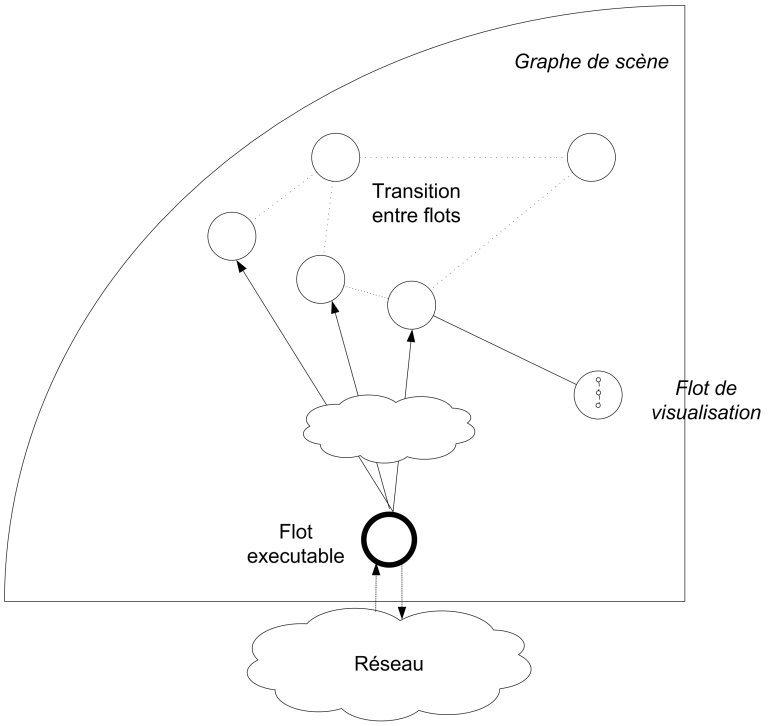


FIGURE 5.5 – Schéma d'un graphe de scène.

5.1.5 Graphe de scène

Définition 6 (Graphe de scène) *Un graphe de scène est la structure logique permettant de relier les flots exécutables entre eux, en fonction de la requête de l'utilisateur.*

Les *noeuds* du graphe de scène sont les flots exécutables, et les *arêtes* permettent la transition d'un noeud à l'autre (figure 5.5), en fonction d'une requête utilisateur. Ces requêtes peuvent être regroupées en fonction du type d'état de données sur lesquelles opèrent, et permettent de caractériser les liens du graphe :

- Liens d'*abstraction de données* : par exemple les liens hypertextes entre deux pages web, similarité entre deux images, connexion sociale entre deux profils sociaux, etc.
- Liens d'*abstraction visuelle* : déplacement de la position d'un objet, changement de propriété visuelle (forme, symbole, etc.).
- Liens de *vue sur le rendu* : zoom dans un espace infini, changement de niveau de résolution, sélection d'un élément dans une vue, etc.

Les liens sont parcourus au fil de la requête de l'utilisateur (construite au niveau du profil applicatif) et les liens peuvent aussi être automatiques si les vues doivent être mises à jour, synchronisées (si les vues sont coordonnées) ou animées.

Le nombre de sommets d'un graphe de scène est très variable, et dépend du type de la tâche à réaliser. Si la tâche consiste en la navigation dans une collection d'images, alors le nombre de vues sera le nombre d'image à quelques vues globales et latérales de détails près (illustration avec le système PHOTOMESA [Bed01] figure 3.12). Si par contre il s'agit de

la navigation web avec Google [Goo10a], alors le graphe de scène commencera avec la page principale du site, et chaque nouveau mot clé saisie et chaque page de résultat parcourue sera un noeud supplémentaire du graphe (le graphe aura donc potentiellement une taille très grande). L'exploration visuelle d'un même jeu de données peut engendrer des graphes de scènes différents (si par exemple une collection d'image est explorée par contenu, par un système de recherche d'image ou par une interface multi-résolution).

Le graphe de scène est orienté, et permet le support de l'exploration visuelle (section 4.2) et le support de l'historique (section 4.2.1) par annotation de ses noeuds et de ses arêtes.

5.2 Mise en oeuvre pratique de l'architecture

Nous allons détailler la mise en oeuvre opérationnelle de VizOD sous forme de services web (section 5.2.2). Nous allons également détailler la création (section 5.2.2) et la composition de ces services en flots de visualisation (section 5.2.3). Ces flots pourront être partagés (section 5.2.4) et couplés à un profil applicatif (section 5.2.5).

5.2.1 Services web de visualisation

Les services web permettent l'implémentation d'une architecture distribuée faiblement couplée. Ils sont définis par le W3C [S+02] comme *"a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via Internet-based protocols"* et un client *"a software that makes use of a Web Service, acting as its 'user' or 'customer"'*. Les services web sont donc des programmes accessibles que par leur interface rendue publique et standardisée, accessible via un réseau. Par conséquent, la localisation, l'environnement d'exécution, le langage d'implémentation et la méthodologie de développement ne sont pas contraignant. Ils permettent donc d'éliminer les problèmes d'interopérabilité au niveau matériel et plateforme d'exécution, et peuvent être utilisés par plusieurs applications. Ainsi les *services de visualisation* définis dans le cadre de VizOD (section 5.1.3) seront par la suite assimilables à des services web, et les *flots de visualisation* assimilables à des services web composés.

Nous choisissons une approche de services web de type REST[3] (*REepresentational State Transfer*), afin de rester en phase avec [HHC+08] qui indique que la collaboration sociale à l'échelle d'internet est facilitée par l'échange de simples URL. Cette approche permet à un simple navigateur web d'interroger tous les services et devient donc un profil applicatif commun de navigation et d'évaluation [Ols07].

Le type de services web type REST a été introduit en 2000 par [Fie00] pour qui *"REST-based architectures communicate primarily through the transfer of representations of resources"*. Il

3. Cette approche est simple et largement utilisée, car elle consiste à permettre la publication et l'interrogation des services au dessus de HTTP et au moyen d'une URL classique. Une alternative pourrait être SOAP (*Simple Object Access Protocol*), recommandée par le *Web Services Architecture* (WSA) qui est utilisée dans certaines systèmes de distribution de visualisation (ADVISE [WBS+08]) mais complexe à mettre en oeuvre. Le système XWeb propose un nouveau protocole pour communiquer les interactions [OJN+00] (autre que HTTP), mais qui ne permet pas d'être interopérable à l'échelle d'internet.

reprend le style de l'architecture *World Wide Web* et intègre également les caractéristiques suivantes :

- **Sans états :** c'est à dire que les paramètres passés au service doivent être suffisants pour exécuter le service sans que le serveur n'ait eu besoin au préalable de stocker des informations.
- **URI unique :** afin d'identifier les ressources (i.e. unités d'information) sans ambigüité. L'URL est une URI qui permet d'accéder à une ressource et composée d'un hôte, d'une chaine de requête (*query string*).
- **Opérations HTTP :** permettent d'interroger de manière uniforme les services : `GET`, `POST`, `PUT`, et `DELETE`.

A noter que l'opération (dit *verbe*) `GET` est souvent la plus utilisée et implémentée par les serveurs web, car elle permet la transmission de paramètre dans l'URL. `POST` est également utilisée mais impose la transmission de paramètre dans le *corps* du message HTTP, ce qui ne permet pas de communiquer l'état dans l'URL (`POST` est principalement utilisé pour les services de type SOAP). Enfin les opérations `PUT` et `DELETE` ne sont pas toujours implémentées.

5.2.2 Création de services web de visualisation

La phase *technique* de création des services web repose sur une approche de création d'*adaptateurs* (*wrappers*) afin d'uniformiser les entrées et sorties de programmes existants [4] au moyen d'une API [Scu09] [5] (*Application Programming Interface*, interface de programmation applicative). L'API [Scu09] rend donc les interfaces des programmes uniformes et disponibles sur un réseau avec une IP et un port réseau. Un programmeur pourra ainsi adapter un script, une fonction, une source de données, etc. en un service web, avec de simples notions élémentaires de Java ou de C++. Ce type d'approche (*wrapping*) existe déjà pour des bibliothèques et plateformes de visualisation VTK [GC07], IRIS Explorer [Wal04] et ADVISE [WBS+08]. Notre approche se différencie par sa simplicité et sa généricité à tout programme exécutable et tout type de source de données (l'API [Scu09] a une implémentation en Java qui fonctionne sous Windows, Linux et MacOS).

A titre d'illustration, le code 2 représente un script actuel de création de graphe avec la bibliothèque LGL [ADWM04] (utilisée pour générer de grands graphes types noeuds/liens) sous Linux. Ce programme est décomposé en deux parties : un exécutable `lglplacer` qui génère la disposition du graphe à partir d'un fichier de données formatées (générées par un script `gen_graph.sh`) et un programme Java `imageMaker.jar` qui effectue le rendu du graphe disposé dans le plan, sous forme d'image (avec un fichier de coloriage de graphe en option). L'utilisation de l'API [Scu09] permet en peu de temps (environ une heure lors du premier usage de l'API, une dizaine de minutes une fois celle-ci maîtrisée) de produire la séquence de services (code 3, sans les paramètres d'exécution des services, ni leur intercon-

4. A condition qu'il soit possible de transmettre au programme les paramètres d'entrées automatiquement, sans nécessiter l'usage d'une interface graphique. Certains logiciels (comme GIMP [Pro10b] ou Tulip [Aub03]) proposent une approche mixte avec une interface graphique utilisateur, et une interface de script. De même pour la sortie du programme.

5. Cette API a été développée dans le cadre du projet "dataspace" au LIRIS `http://ds.liris.cnrs.fr/` et vise à fournir une vue intégrée et dynamique sur un espace de données issues de capteurs, bases de données, fichiers, etc. sous forme de services web de type REST.

nexion) qui permet la création du même graphe. Ces services peuvent être interrogés dans un simple navigateur, qui effectuera automatiquement un `GET` qui déclenchera l'exécution des programmes avec les paramètres donnés en URL.

Code 2 Script permettant de générer un graphe avec LGL.

```
// Execution d'un script de génération de graphe et stockage dans un fichier
// au format LGL
./gen_graph.sh > graph.lgl

// Disposition (layout) du graphe
lglplacer -o graph.coords

// Rendu du graphe en image 1024x768 pixels
java -mx512M -jar cgi-bin/imageMaker.jar 1024 764 graph.lgl graph.coords

// Affichage de l'image générée
xv graph.coords.png
```

Code 3 Services permettant de générer un graphe avec LGL.

```
// Service de génération de données de type graphe LGL
http://127.0.0.1:8015/GenGraph/

// Service de disposition (layout) du graphe
http://127.0.0.1:8016/GraphLayout/

// Service de rendu du graphe
http://127.0.0.1:8017/GraphRender/

// Service d'affichage du graphe (image PNG)
http://127.0.0.1:8018/ViewImage/
```

5.2.3 Composition de services en flots de visualisation

Suite à la création des services, il est nécessaire de proposer un mécanisme *technique* de composition en flots de visualisation. Le principe que nous utilisons est similaire aux *pipes* Unix[6], qui permettent de faire communiquer plusieurs programmes au moyen d'une interface commune. Les détails techniques des communications sont décrits dans l'API [Scu09][7],

6. http://en.wikipedia.org/wiki/Pipeline_(Unix)

7. les communications entre services se réalisent au moyen d'URL. Par exemple l'URL suivante permet de connecter l'entrée d'un service de construction de graphes à une sortie source de données (nombres

nous allons nous intéresser à la méthodologie de composition par l'utilisateur, à laquelle nous avons contribué.

Composer les services peut être complexe car il s'agit d'intégrer de nombreux paramètres de correspondance entre les services (paramètres d'entrée et de sortie) tout en explorant l'espace de conception (choix des services). Notre approche a été de proposer une interface de programmation visuelle MASHVIZ [VR09c] (figure 5.7) qui permet aux utilisateurs de créer des flots en manipulant des éléments graphiques (symboles, agencement dans l'espace, etc.) qui représentent les services et leurs connexions. L'utilisation de ce type d'interface est courante dans des domaines très variés où les données sont sous forme de flux [JHM04]. C'est une approche mixte de programmation qui offre un niveau d'abstraction de composants qui permet à la fois d'impliquer des experts et des utilisateurs experts d'un domaine (*power user*). L'objectif d'une telle interface est de permettre à un utilisateur de comprendre et de manipuler du code à un niveau intuitif (comme par exemple éviter de manipuler directement le code 3), et lui fournir un ensemble d'interactions possibles, qui permettent également d'éviter des erreurs (de bas niveau, comme éviter le choix de paramètres incompatibles entre deux services).

MASHVIZ s'inscrit dans un écosystème plus large de création et de partage des services. Pour cela nous avons identifiés trois types d'acteurs [VR09b] (diagramme de cas d'utilisation figure 5.6) :

- **Les programmeurs :** qui programment une machine en fonction des besoins d'utilisateurs ou de designers.
- **Les designers :** ou architectes qui définissent des spécifications ou manipulent des briques déjà produites, et qui s'intéressent à la structure globale.
- **Les utilisateurs** qui utilisent un système afin de réaliser une tâche.

L'implémentation de MASHVIZ (capture d'écran figure 5.7) est orientée web (et ne dépend donc pas d'un système d'exploitation). Elle reprend le *look and feel* de Yahoo ! Pipes [Pip10] pour l'apparence générale, l'adaptation des services en boites, et la connexion élastique par *tuyaux* en courbes de Bézier. L'utilisateur possède donc un plan de travail dans lequel des services peuvent être glissés/déposés depuis la *base de service*, qui a été divisée en catégories d'étapes de traitement des données [VRP09]. Chaque service déposé peut être personnalisé dans la boite qui le représente. Cette boite est publiée par le service lui même et est affichée sous forme de sous-page web dans la boîte. Ensuite, l'utilisateur peut connecter/déconnecter les entrées et sorties des services afin de créer un flot. Il n'est pas possible pour l'utilisateur de relier des boites incompatibles.

L'interface MASHVIZ ne possède pas les ressources qu'elle exécute, car les services sont exécutés à distance. Le flot de visualisation ainsi créé peut être sauvegardé et restauré par la suite au moyen de l'interface de partage de flots (section 5.2.4).

aléatoires) : `http://127.0.0.1:1713/GraphBuilder/input/Data/connection/add?URL=http://127.0.0.1:8015/RandomNumber/output/number`

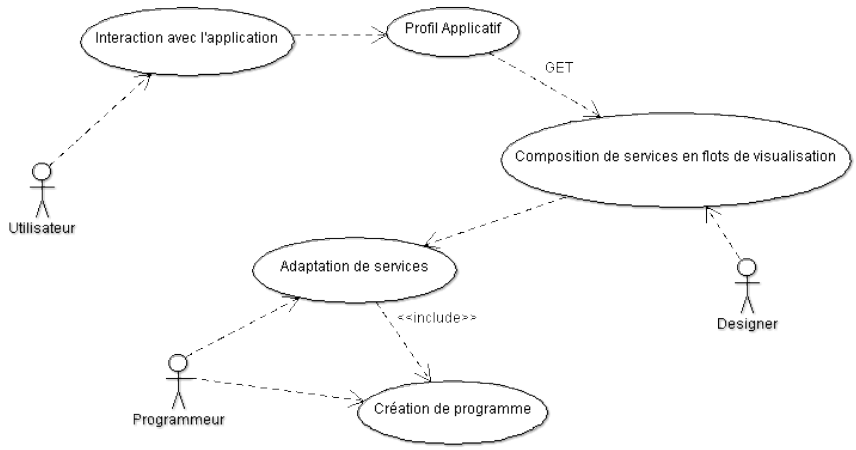


FIGURE 5.6 – Diagramme de cas d'utilisation VIZOD (version simplifiée).

5.2.4 Partage de flots de visualisations

Les flots et services de visualisation peuvent être partagés à l'échelle du web sous forme d'URL (et permettre la collaboration sociale [HHC+08]). Editer un flot existant permet d'éviter la répétition de tâches fréquentes en réutilisant des connaissances capitalisées dans les flots. Le partage permet aussi de réduire l'espace de conception et permet la collaboration asynchrone au moyen de commentaires, annotations ou statistiques d'usage. Afin d'assister ces opérations et de les centraliser, une deuxième interface web (figure 5.8) propose un répertoire de flots et de services, et permet de les visualiser et de les éditer (avec l'interface précédente de composition section 5.2.3). Les attributs de ce répertoire de flots sont les suivants :

- ❶ la *vue globale du flot*, qui est une image miniature qui donne une information quantitative sur la structure du flot et sa complexité.
- ❷ la *description* : nom, description.
- ❸ les *commandes* : exécuter, cloner, supprimer.
- ❹ les données d'*usage* : tags, propriétaire, noms des services qui composent le flot, dernières modifications, profils applicatifs compatibles avec ce flot.

Des fonctionnalités de filtrage et de recherche de flots sont fournies, ce qui permet par exemple de trouver les flots qui utilisent un certain service, qui ont été créé par un utilisateur ou qui sont compatibles avec un profil applicatif particulier.

Partager la composition de services qui constitue un flot permet une analyse d'usage plus fine que les approches classiques de partage de visualisation (telles que ManyEyes), car il

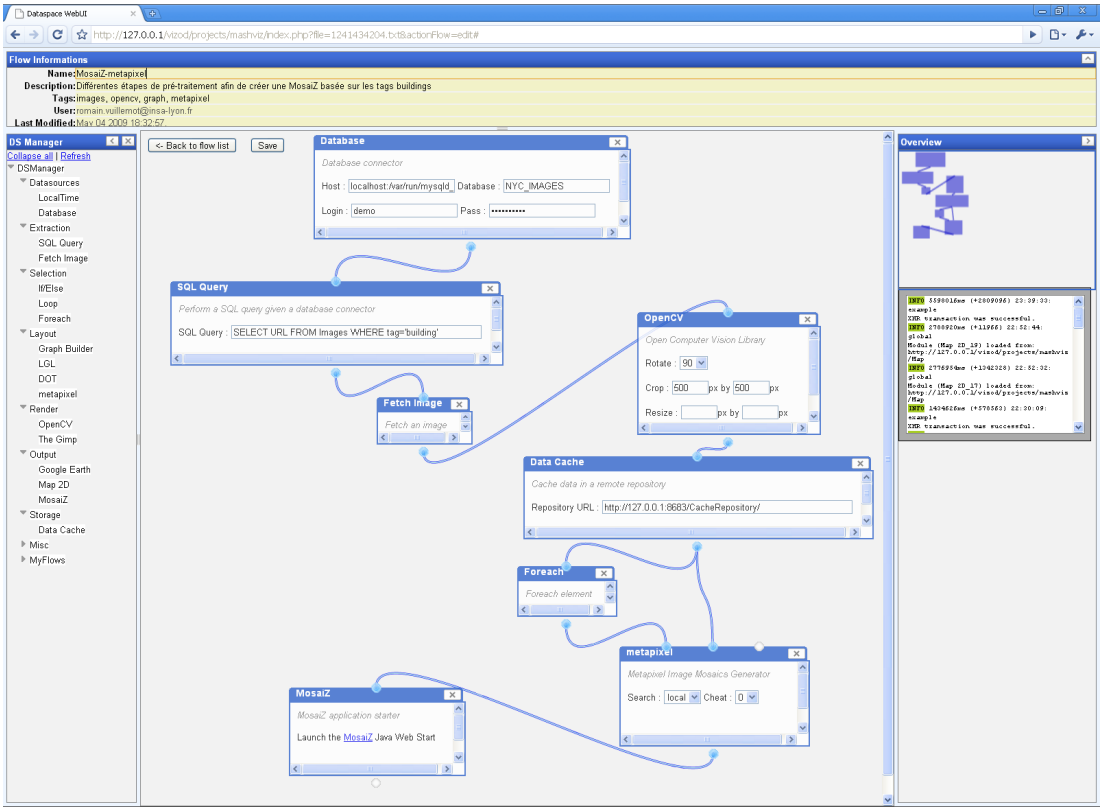


FIGURE 5.7 – Capture d'écran de l'interface MASHVIZ d'édition de flots.

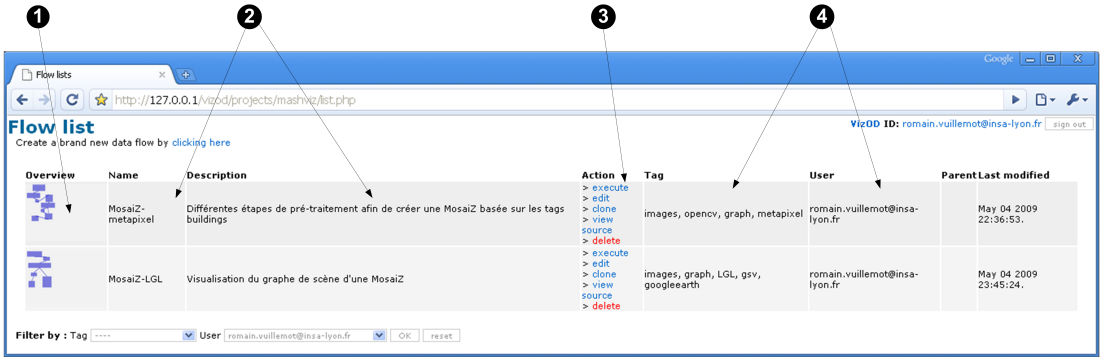


FIGURE 5.8 – Capture d'écran de l'interface MASHVIZ dédiée au partage de flots. Les flots déjà créés sont visibles sous forme de liste qu'il est possible de filtrer par tags ou par utilisateurs.

ne s'agit plus de partager le rendu (image) ou l'abstraction de données (comme un jeu de données) mais *toutes* les étapes qui constituent le résultat (dans la mesure où la granularité des services est assez fine). Les données d'usage sont visibles pour le fournisseur des services, et des tâches supplémentaires de surveillance et de débogage d'exécution peuvent être envisagées et affichées dans MASHVIZ [8].

8. Mais cela est encore considéré comme un problème complexe pour les interfaces de programmation visuelle [JHM04].

5.2.5 Couplage au profil applicatif

Le protocole de communication entre les flots de visualisation exécutables et un profil applicatif est réalisé au niveau du protocole applicatif HTTP [FGM+99]. L'appel d'un flot ou d'un service par un profil applicatif est identique à l'appel d'une page web par un navigateur (code 1). Il s'agit d'une séquence de caractères qui comporte l'IP ou l'hôte du destinataire, les paramètres de la requête, ainsi que les informations relatives au navigateur (profil applicatif). En retour, le service répond une chaine de caractère également composée d'un code [FGM+99] qui permet (entre autres) d'indiquer si la ressource (programme adapté par le service) est disponible et le résultat transféré (`200 OK`), si les droits d'accès ne sont pas suffisants (`403 Forbidden`) ou si la ressource n'est pas trouvée (`404 Not Found`). Les paramètres (transmis dans l'URL de la requête) et le format de retour des données (transmis dans le corps du message de retour) sont définis par l'API [Scu09]. Un exemple (Code 4) de données produites par une service en cas de code de retour positif (il s'agit d'un service de source de données de nombres aléatoires) [9] :

Code 4 Extraits de résultats de l'interrogation d'un service de source de données.

```
Name:TnVtYmVy
ID:0
Number:12

Name:TnVtYmVy
ID:1
Number:53
```

Le diagramme de séquence de VizOD (figure 5.9) présente une vue simplifiée du couplage entre le profil applicatif et les services et flots de visualisation. Le code de retour du résultat est communiqué immédiatement au profil applicatif, alors que les résultats peuvent eux n'arriver que progressivement (si le temps d'exécution est long, si seulement certaines résolutions de données sont générées d'abord ou si les données sont continues).

5.3 Métriques de mesure d'usage à partir de VizOD

Les services web, le profil applicatif et le graphe de scène offrent différentes métriques *quantitatives* de suivi et d'analyse de l'usage de l'utilisateur (exemple de métriques tableau 5.3), ainsi que de caractérisation de l'environnement d'exécution. Toute application interactive basée sur l'implémentation de VizOD peut donc être systématiquement instrumentalisée [10] et évaluée selon ces métriques. Les données issues de ces métriques sont générées sur les machines hébergeant les services web et les profils applicatifs, ou par un proxy intermédiaire [HHWL01]) et sont stockées dans des logs. Cette automatisation permet notamment de faciliter l'évaluation à long terme d'interfaces visuelles [SP06] [11] mais ne permet pas

9. L'URL d'interrogation de ce service est de la forme `http://127.0.0.1:8815/RandomNumber/output/number/stream`

10. Dans le cadre d'un respect de la vie privée, ce qui implique un accord explicite de l'utilisateur.

11. Toutes les métriques ne sont pas à utiliser, le choix de leur pertinence est laissée aux évaluateurs en fonction de la tâche, et de la quantité de données d'usage générées.

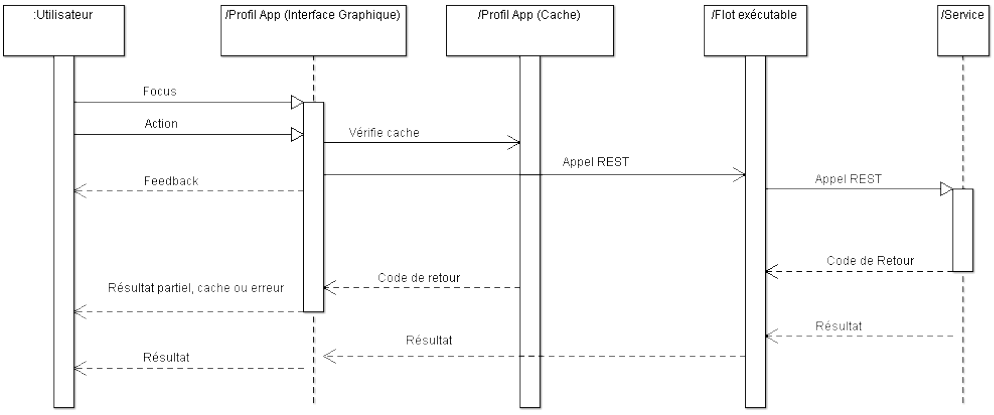


FIGURE 5.9 – Diagramme de séquence de VIZOD (version simplifiée).

d'expliquer la cause d'une action ou de détecter tout facteur externe (discussion, annotation sur un morceau de papier, etc.) à l'environnement informatique.

Module	Indicateur	Unité
Services et flots	Temps moyens d'exécution	*ms*
	Latence du réseau	*ms*
	Débit du réseau	*Kb/s*
	Jitter (variabilité de la latence)	*ms*
Graphe de scène	Nombre de sommets	*entier*
	Nombre d'arêtes	*entier*
	Connectivité des noeuds	*binaire*
Profil applicatif	Réactivité	*ms*
	Rétroactivité	*ms*
	Framerate	*images/s*

TABLE 5.1 – Liste de métriques quantitatives d'analyse VIZOD.

D'autres types de métriques peuvent compléter cette liste, comme des métriques qualitatives telles que le type de codes de retours, le type de requêtes ou le type de liens du graphe de scène. Il existe des métriques d'étude de performances propres aux architectures découplées et distribuées [12], pour notamment identifier les goulots d'étranglement (*bottlenecks*). Côté profil applicatif, le suivi du dispositif de pointage ou les techniques d'oculométrie (*eye-tracking*) permettraient d'identifier les points de fixation de l'utilisateur. Enfin l'analyse de la topologie du graphe de scène permettrait de mettre en évidence des parcours récurrents (en fonction du chemin, nombre de visites, liens entrants à un noeud, etc.).

12. http ://en.wikipedia.org/wiki/Application_Response_Measurement

5.4 Exemple de conception

Dans cette section nous allons décrire un exemple d'implémentation d'un programme visuel interactif simple basé sur VizOD. Ce programme consiste en un widget de type *slider* (figure 5.10) associé à l'affichage dynamique d'un graphe. La création du graphe reprend les scripts (code 2) adaptés en services (code 3) et sera générée dynamiquement avec un nombre de sommet égal à la valeur numérique du slider. Le défi de conception réside donc dans le choix du type de requête (section 3.2.2) adapté en fonction du temps de réponse du système [Nei93] qui dépend de la génération visuelle du graphe (qui évolue en fonction du nombre de sommets, et donc du déplacement du slider).

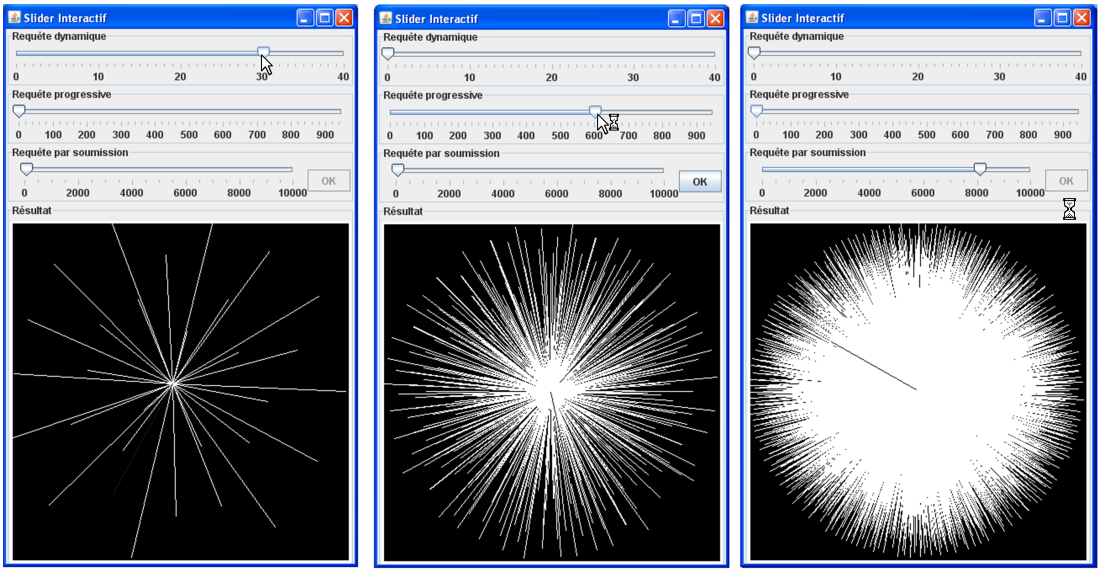


FIGURE 5.10 – Résultat de l'implémentation d'un slider interactif couplé à une visualisation de graphe.

1. La première étape consiste à créer à partir des scripts (code 2) les services `GenGraph_VIZOD`, `GraphLayout_LGL`, `GraphRender_LGL` au moyen de l'API [Scu09] (code 3). Les services seront ensuite composés avec MASHVIZ afin produire un flot dont le graphe de sortie est au format PNG (figure 5.11).

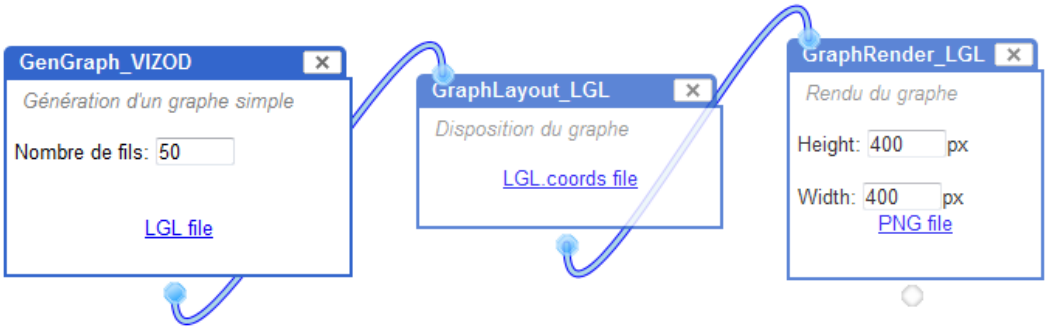


FIGURE 5.11 – Flot de visualisation du graphe simple composé avec MASHVIZ.

2. La seconde étape consiste à analyser au moyen de métriques (tableau 5.3) le temps d'exécution de ce flot. Etant donné que le service vient d'être créé, il n'existe pas encore de logs ou de valeurs d'usage sur lesquelles se baser. Il est donc nécessaire d'en générer en fonction de différentes valeurs $|N|$ (nombre de sommets du graphe). Suite à différents tests de variation de $|N|$, nous pouvons en déduire trois paliers de requêtes utilisateur[13] : la requête sera *dynamique* pour $|N| < 40$ (car au delà le temps de génération dépasse $0.5s$), *progressive* pour $|N| < 950$ (car au delà le temps de génération dépasse $1s$) et par *composition* pour tout $|N|$.
3. La troisième étape consiste à créer un profil applicatif intégrant les vues en fonction des requêtes de l'utilisateur. Par soucis de consistance et de lisibilité, nous avons utilisé trois sliders (mais un seul aurait pu être utilisé et s'adapter au temps de réponse en fonction $|N|$) (figure 5.10). Un slider peut être représenté comme une succession d'état (figure 5.12) reliés entre eux par des événements. L'événement `mouseDragged` déclenchera un appel à un flot en cas de requête dynamique, `mouseReleased` en cas de requête progressive, et enfin (non visible) `ActionListener` une requête par composition après l'appui sur le bouton `OK`. L'icône du dispositif de pointage sera variable en fonction du type de requête. A noter que chaque appel de flot nécessite de composer la requête, de former une URL, de vérifier si une donnée identique n'existe pas en cache, et enfin d'envoyer la requête et d'afficher la réponse (si le code de retour est valide). Chaque appel à un flot engendre la création d'un noeud dans le graphe de scène.

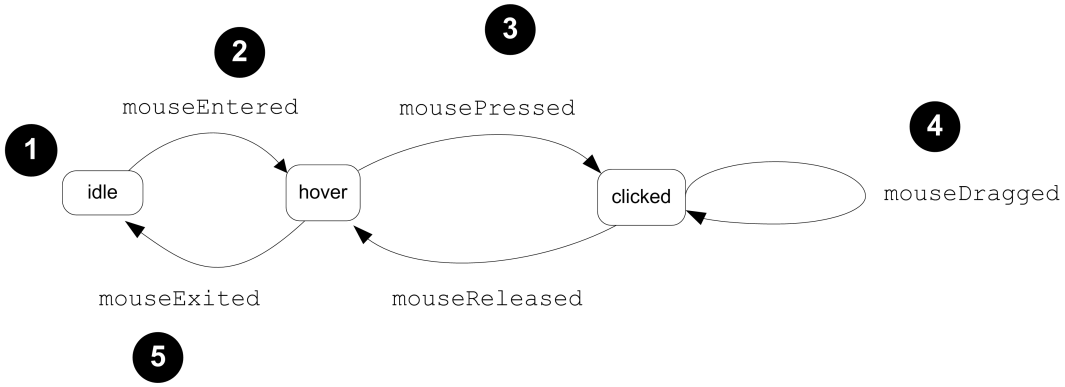


FIGURE 5.12 – Différents états et types d'événements d'un slider.

Des optimisations sont possibles comme la prise en compte des évènements `mouseEntered` et `mouseExited` pour capter respectivement le focus et le non focus du dispositif de pointage de l'utilisateur sur le profil applicatif (afin de vérifier à l'avance si les flots sont disponibles ou non, par exemple). Un affichage dans l'application de l'image des n derniers résultats permettrait à l'utilisateur de visualiser son historique. A noter que chaque noeud du graphe possède une URL unique qu'il est possible d'utiliser dans d'autres profils applicatifs, tels qu'un navigateur web. L'URL pourra aussi être inclue dans un blog ou dans une page web afin de permettre un processus d'analyse sociale [HHC+08].

13. Des tests utilisateurs seraient à réaliser pour affiner ces paliers.

5.5 Synthèse de l'architecture

Dans ce chapitre nous avons introduit VizOD, un cadre de décomposition de l'espace de design de la visualisation d'information interactive en trois parties : des *services* et *flots de visualisation*, un *profil applicatif* et un *graphe de scène*. Nous avons proposé une implémentation possible sous forme de services web, et implémenté une interface de programmation visuelle (mashviz) qui permet d'assister la composition et le partage de flots de visualisations. Nous avons indiqué une série de métriques quantitatives qui peuvent être utilisées afin d'évaluer les différents composants de VizOD. Enfin nous avons donné un exemple simple d'implémentation d'un slider interactif couplé à la visualisation de graphe généré par un flot de visualisation.

Dans le chapitre suivant (chapitre 6), nous allons détailler des applications plus complexes développées à partir de VizOD.

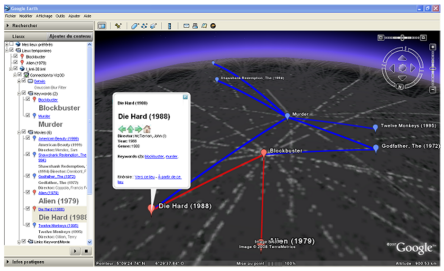
Chapitre 6

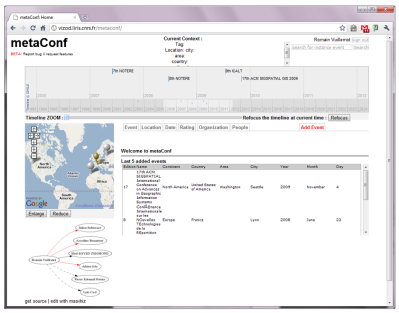
Applications développées avec VizOD

Sommaire

6.1 Aperçu des applications développées

Nous avons développé quatre applications visuelles interactives (liste tableau 6.1) à partir de l'architecture VizOD. Ces applications possèdent chacune un jeu de données, une tâche et un type de visualisation distinct. Le choix de ces applications a été opportuniste, au sens qu'ils ont répondu aux besoins ponctuels de projets sans planification préalable, mais ont permis de faire mûrir et converger l'architecture VizOD. Pour obtenir plus de détails sur ces applications, une liste de publications qui y font référence est donnée tableau 6.1.





Application	Description	Section
VizOD+GE	*Exploration multi-échelle de graphes* [VP07, VR08]	6.2
metaConf	*Exploration multi-facette de réseaux sociaux* [Vui08]	6.3
MosaiZ	*Exploration de collection d'images organisées par similarité* [VR09a]	6.4
POSvis	*Caractérisation d'entités textuelles par nuages de mots* [VCPK09, VCPV09]	6.5

Table 6.1 – Liste et capture d'écran des quatre applications développées avec VizOD.

Dans ce chapitre, nous décrivons pour chaque application les services utilisés (regroupés dans le tableau B.1), leurs flots ou parties de flots respectifs (annexes B.2, B.3, B.4, B.5) et donnons des statistiques sur leur exécution[1]. (issus de métriques énumérées section 5.3) afin d'expliquer certains choix et alternatives de design. Dans le chapitre suivant (chapitre 7) nous discuterons de l'architecture VizOD de manière plus générale, suite à ces expériences de développement.

1. Tous les programmes exécutés côté serveur l'ont été sur une machine dont le système d'exploitation est une distribution GNU/Linux Fedora Core 6, 512 Mo de mémoire RAM, processeur AMD AthlonTMXP 2500+ (1.8 GHz) avec 512 Ko de mémoire cache.

6.2 VizOD+GE : exploration multi-échelle de grands graphes

La première application développée s'appelle VizOD+GE (VizOD associé à Google Earth [Goo10b]). Cette application permet à un utilisateur d'explorer un graphe à différentes échelles ou niveaux de résolution (section 3.3.2), en utilisant l'application Google Earth existante (figure 6.1). Ce type d'application été montré particulièrement adaptée à l'exploration de données [SHR+08], nous avons donc décidé de l'utiliser mais pour un usage autre que géographique l'exploration d'un graphe. Le graphe à explorer est de type noeud/lien classique, la position des noeuds est auto-organisée [Ead84] en utilisant le programme LGL [ADWM04] (*Large Graph Layout*). Le graphe représente les connexions entre films ayant des mots-clés similaires [VP07] issus de films américains de 2000 to 2006 (à peu près 20 000 films) preovenant des bases IMDB [htt08] (Internet Movie Database) et de netflix [net10] [2].

Le *profil applicatif* utilisé est donc Google Earth, dont les différentes couches de données et annotations géographiques sont remplacées respectivement par des images et annotations abstraites (figure 6.2) issues de rendus du graphe. VizOD+GE reprend donc les principes classiques et bien établis de la cartographie d'information (section 2.2.3), mais cette fois appliqués à l'exploration de graphes qui peuvent êtres vus comme des cartes, certes, mais abstraites. L'approche multi-résolution de l'interface vient du fait que la résolution du graphe (à savoir le niveau de détails de l'abstraction visuelle et du rendu de celui-ci) varie progressivement en fonction de l'altitude de l'utilisateur par rapport au globe terrestre (comme c'est déjà le cas avec les données géographiques). Enfin l'utilisateur peut cliquer sur une localisation de la carte et obtenir des détails sur l'information géo-encodée (que sont désormais les films comme par exemple "*Die Hard*" sur la figure 6.1). Des widgets sont disponibles sur le panneau latéral gauche (sous forme de listes arborescentes et de checkboxes), qui permettent d'accéder aux données du graphe (sommets et arrêtes) sous forme de liste [VP07, VR08].

Nous allons nous intéresser aux phases *techniques* de conception de l'interface, qui nécessitent de considérer à la fois la création, mais aussi le dessin (*layout*) et la navigation du graphe :

1. Création d'un flot de visualisation de graphe et différentes résolutions de rendu (section 6.2.1).
2. Couplage avec le profil applicatif et navigation multi-résolution (section 6.2.2).
3. Exploration du graphe par l'utilisateur et analyse de son parcours (section 6.2.3).

6.2.1 Flot de visualisation du graphe et rendus multiples

La création du graphe est réalisée par un flot de visualisation (annexe B.2) dont la source est une base de données Oracle (publiée via le service `DatabaseHost_ORACLE`) et la sortie un fichier `kml` exécutable avec Google Earth (publié par le service `KML_VIZOD`). Le service `GraphBuilder_VIZOD` (issu d'une série de services développés pour VizOD [SVR08]) crée un graphe où chaque noeud est l'identifiant de film, et il existe une arrête entre deux films si ils

2. nous avons à la fois utilisé les données du projet ACI APMD et celles du concours Infovis 2007 [VP07] qui étaient du même type

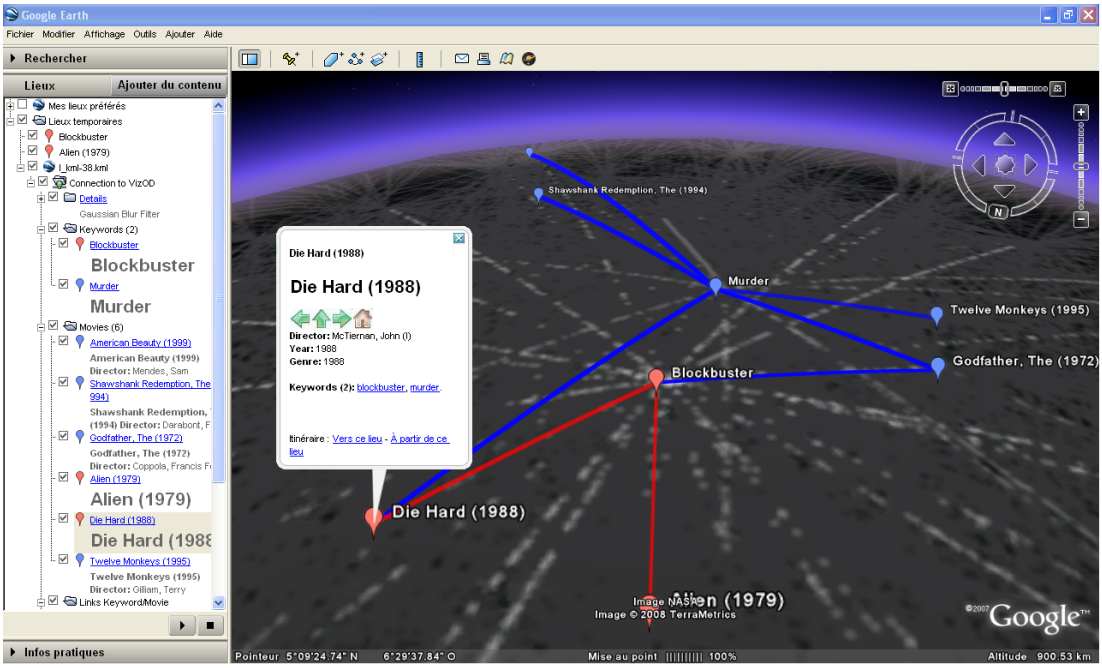


FIGURE 6.1 – Capture d'écran de VIZOD+GE et détail du sommet d'un graphe au niveau de résolution le plus bas.

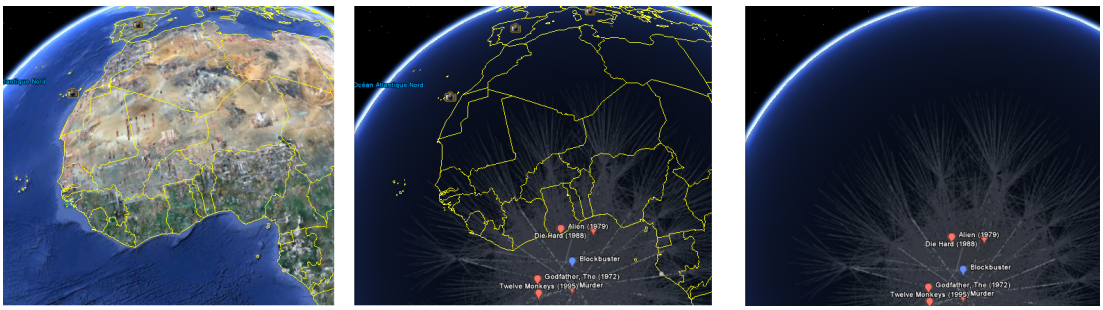


FIGURE 6.2 – Illustration de la suppression des couches géographiques de GOOGLE EARTH (à gauche) vers des couches de abstraites graphes (à droite).

ont des valeurs d'attributs en communs (dans ce cas il s'agit d'au moins un mot-clé similaire).

L'observation du temps d'exécution du flot de visualisation du graphe (figure 6.3) montre (et c'est courant) que la phase de disposition du graphe `GraphLayout_LGL` est très longue et ne peut être générée en temps réel pour chaque niveau de résolution du graphe. Certes les phases de requête `Query_ORACLE` et de construction du graphe `GraphBuilder_VIZOD` sont relativement rapides, mais engendrent ensuite un nouvel appel à `GraphLayout_LGL`. Il n'est donc pas possible de réaliser une requête différente pour chaque niveau d'échelle. La solution est donc de jouer sur l'abstraction visuelle (autre que la position) et le rendu pour donner un sentiment de changement de résolution à l'utilisateur. Nous avions évoqué (section 3.3.1) que le filtrage flou peut être utilisé afin de mettre en avant un objet sélectionné (focus) par l'effacement de son contexte. Ce mécanisme permet de réduire la surcharge visuelle, tout en

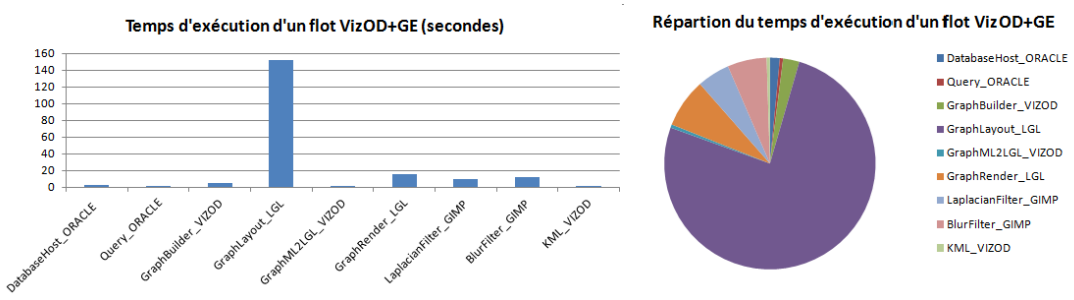


FIGURE 6.3 – Temps d'exécution du flot de visualisation VIZOD+GE et répartition entre services.

attirant l'attention de l'utilisateur sur le focus qui lui reste net. Pour donner cet effet de profondeur, nous avons utilisé le logiciel GIMP (*The GNU Image Manipulation Program*) [Pro10b] transformé en service (`BlurFilter_GIMP`) qui va flouter l'image du graphe (contexte), et le focus sera généré par l'application GOOGLE EARTH en affichant les données du graphe sous forme de lignes vectorielles (disponibles à partir des données issues de `GraphBuilder_VIZOD` et transmises à `KML_VIZOD`). Enfin la vue globale du graphe n'étant pas satisfaisante de cette manière là, nous avons aussi effectué une analyse de traitement de contour afin de mettre en avant les regroupements du graphe, toujours avec GIMP (service `LaplacianFilter_GIMP`).

Pour résumer, trois rendus de graphe sont donc générés en fonction du point de vue de l'utilisateur (altitude et champs de vision) :

- Vue *globale* (*overview*) : le filtre `LaplacianFilter_GIMP` qui permet la détection de contour est appliqué à l'image, afin de mettre en avant les regroupements.
- Vue de *zoom* (*zoom*) : aucun filtre n'est appliqué.
- Vue de *détails* (*details*) : le filtre `BlurFilter_GIMP` est appliqué mais laisse le graphe suffisamment visible pour restituer les noeuds/arrêtes connexes sous forme de contexte. Les données issues du graphe `GraphBuilder_VIZOD` sont affichées de façon vectorielle, ce qui permet un rendu des sommets et des arêtes avec une haute résolution de rendu.

Il n'est donc nécessaire de ne générer qu'une seule image du graphe (ce qui peut être réalisé au démarrage de l'application), et ensuite les différents niveaux de résolution peuvent être générés à la volée.

6.2.2 Couplage du rendu au profil applicatif et navigation multi-résolution

Le *profil applicatif* GOOGLE EARTH permet un degré de liberté de translation selon l'axe d'altitude, et deux degrés de navigation dans le plan, entre autres. Ces degrés de liberté permettent donc de naviguer dans les rendus de graphes préalablement créés, et le changement d'altitude permettra le changement de résolution du graphe. Pour connaître les données à afficher, GOOGLE EARTH effectue un appel périodique au service de visualisation `KML_VIZOD` avec pour paramètres le champ de vision de l'utilisateur en cours, et son altitude. Ainsi `KML_VIZOD` joue un rôle de contrôleur d'altitude et retourne le graphe adapté en fonction de l'altitude. A noter que l'application GOOGLE EARTH reste toujours très réactive même si l'image à afficher n'est pas encore transférée. Si par exemple à ce moment là l'interaction de

l'utilisateur est un zoom, et l'image n'est pas chargée, alors Google Earth effectuera un un zoom bitmap sur l'image en cours qui sera pixellisée (mais qui ne bloquera pas l'interface).

6.2.3 Exploration visuelle de l'utilisateur

Pour assister l'utilisateur dans l'exploration des graphes, nous avons rajouté deux vues de navigation [VR08] : des liens hypertextes entre les sommets, et la gestion de l'historique de la navigation. Le résultat des liens hypertextes (figure 6.4) est intuitif car il reprend l'encodage visuel et les interactions de la navigation web : les liens cliquables dans les bulles d'information sur les sommets du graphe (comme les mots clés du film "*Die Hard*" sur la figure 6.1), les arêtes non visitées sont bleu et celles visitées sont rouge (figure 6.1). De même, l'historique du parcours du graphe (noeuds précédents, noeud initial, etc.) est accessible dans chaque bulle sous forme d'icônes de navigation que l'on retrouve dans les navigateurs web. Enfin nous avons rajouté une flèche verticale (entre les flèches retour en arrière et retour en avant) pour permettre à l'utilisateur de revenir à l'état initial de sa navigation.

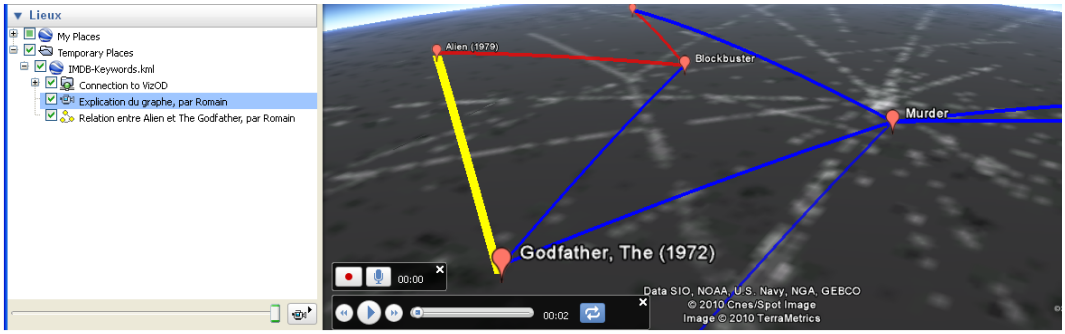


Figure 6.4 – Exemple d'ajout d'annotations et de relecture du parcours dans VizOD+GE.

Google Earth permet également d'annoter et de communiquer les navigations (figure 6.4). L'annotation consiste en l'ajout de nouvelles arêtes ou sommets dans le graphe, avec des commentaires possibles sur ceux-ci. Le partage d'animations d'exploration est possible, via l'enregistrement de navigations qui peuvent être associées à un enregistrement vocal.

Concernant l'analyse du parcours de l'utilisateur, il est possible de le construire et de le représenter à partir du graphe de scène de l'application sous forme de diagramme d'état (figure 6.5). Le diagramme est construit à partir des logs de l'activité d'utilisateurs, au fil de sessions [Vui10] (illustration figure 6.5 à gauche). Les états de ce diagramme sont les trois niveaux de résolution que l'utilisateur peut explorer (vue globale, zoom et détails). Les transitions entre états sont le passage d'un niveau de résolution à un autre. Cette représentation permet de vérifier si par exemple le mantra de [Shn96] est vérifié (ce qui n'est pas le cas car l'application démarre au niveau de zoom [Vui10]).

6.2.4 Bilan et discussion

Le premier résultat a été de type *qualitatif*, et a permis de rapidement pouvoir développer une application pour le concours IEEE Infovis 2007 [VP07]. La séparation entre le code de

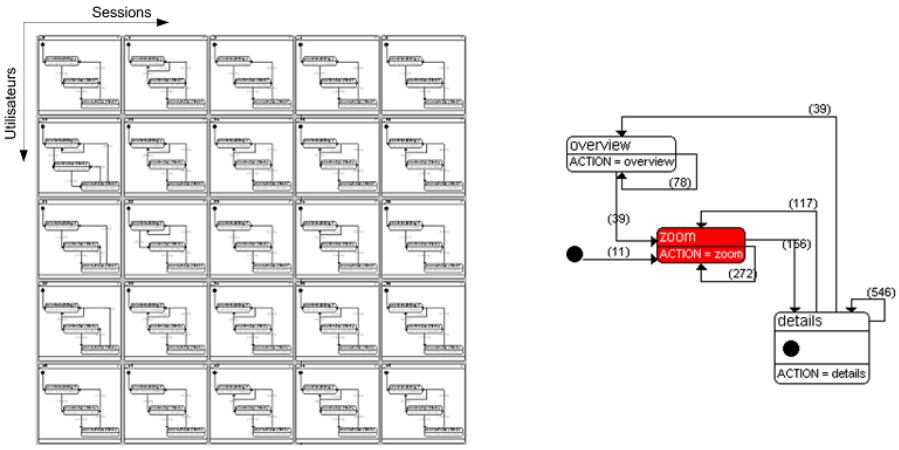


FIGURE 6.5 – Diagramme d'état de parcours utilisateur dans l'interface VIZOD+GE.

sélection des données (à savoir les requêtes SQL) et le code de présentation et de filtrage a permis à deux personnes de travailler en parallèle, l'une sur les requêtes, l'autre sur l'interface et la coordination des vues. Les optimisations de requêtes ont pu être systématiquement relevées via l'analyse d'exécution du service `Query_ORACLE`. Le prototype qui en résulte est fiable (basé sur GOOGLE EARTH qui est une application *stable* en terme d'état de développement) et facilement être déployé sur plusieurs plateforme et sans en gérer le déploiement ni la maintenance. A noter que même si la première version du prototype a été réalisée en mars 2006 en Master Recherche [Vui06] avec la version 4 de GOOGLE EARTH, il fonctionne toujours aujourd'hui (août 2010) avec la version 5.

Réutiliser le *profil applicatif* que constitue GOOGLE EARTH a cependant des limites. Il n'est pas possible d'obtenir les métriques côté profil applicatif comme le framerate, la réactivité, ou connaître les interactions avec les widgets locaux de filtrage.

FIGURE 6.6 – Couplage du flot de visualisation à l'application MICROSOFT ZOOM.IT.

Cependant le flot de visualisation créé peut être exploré avec d'autres profils applicatifs tels que ceux de type "Google Maps" [Goo10d] (à savoir l'interaction multi-échelle dans le plan d'une page web munie de scripts Javascript d'interaction, comme le permet la bibliothèque Javascript GSV [3]). Dès lors le niveau de résolution change par paliers (il n'y a plus de zoom continu). Un profil applicatif de ce type qui permet un zoom continue est Microsoft Zoom.It [4]. Il s'agit d'une application Microsoft Silverlight à laquelle on peut transférer (via un formulaire sur leur site web) une image à très haute résolution. Le site se charge d'indexer l'image et de permettre ensuite une navigation fluide avec chargement progressif en fonction des interactions utilisateur. Pour obtenir une telle image il faut donc paramétrer le service de rendu `GraphRender_LGL` avec par exemple pour sortie une image de dimension 10000×10000 pixels. Ce site offre un cadre applicatif nouveau (figure 6.6) mais une limite importante est le manque d'interopérabilité, car 'il n'est pas possible de connecter directement le flot de visualisation issu de VizOD avec le site Zoom.It (il est nécessaire de remplir le formulaire d'envoie d'image à chaque fois qu'un nouveau flot est généré).

6.3 metaConf : exploration multi-facette de réseaux sociaux

metaConf [5] est un site web destiné à l'exploration de conférences scientifiques (capture d'écran figure 6.7). Ce site est similaire à DBLP [6] en termes de jeux de données : il inclue les auteurs et co-auteurs de papiers scientifiques, ainsi que les noms et éditions de conférences scientifiques. Ces données sont complétées par le géo-encodage des lieux et des informations liées aux auteurs (qu'ils peuvent eux-même rajouter) tels que leur organisation, préférences pour une conférence et connexions professionnelles avec d'autres auteurs. Il s'agit donc également d'un réseau social autour des conférences scientifiques, comme le propose par exemple biomedexperts [7]. L'objectif est de permettre à un utilisateur de composer une requête par lieu ou par date, et ainsi de répondre à une question habituelle du type *"Quelles sont les conférences A+ organisées à Hawaï dont la date de soumission des papiers longs est dans deux mois ?"* [Vui08].

Nous souhaitons représenter les données de manière synthétique en intégrant plusieurs facettes (à savoir les catégories des données). Ces facettes sont disponibles sous forme de listes, mais aussi de visualisations telles que des cartes géographiques, lignes temporelles et graphes. Le *profil applicatif* utilisé est le navigateur web, dans le cadre des standards du web [8]. Nous allons détailler les points suivants :

1. La création d'un flot de visualisation d'un graphe à partir de données sociales (section 6.3.1).
2. Le couplage du flot de visualisation à une page web et l'analyse du temps de rendu de la page (section 6.3.2).

3. http://mike.teczno.com/giant/pan/
4. http://zoom.it/
5. Version en ligne http://vizod.liris.cnrs.fr/metaconf/
6. http://www.informatik.uni-trier.de/~ley/db/
7. http://www.biomedexperts.com/
8. à savoir HTML, Javascript, CSS, etc. sans plugins externe requis (comme par exemple Adobe Flash ou Microsoft Silverlight

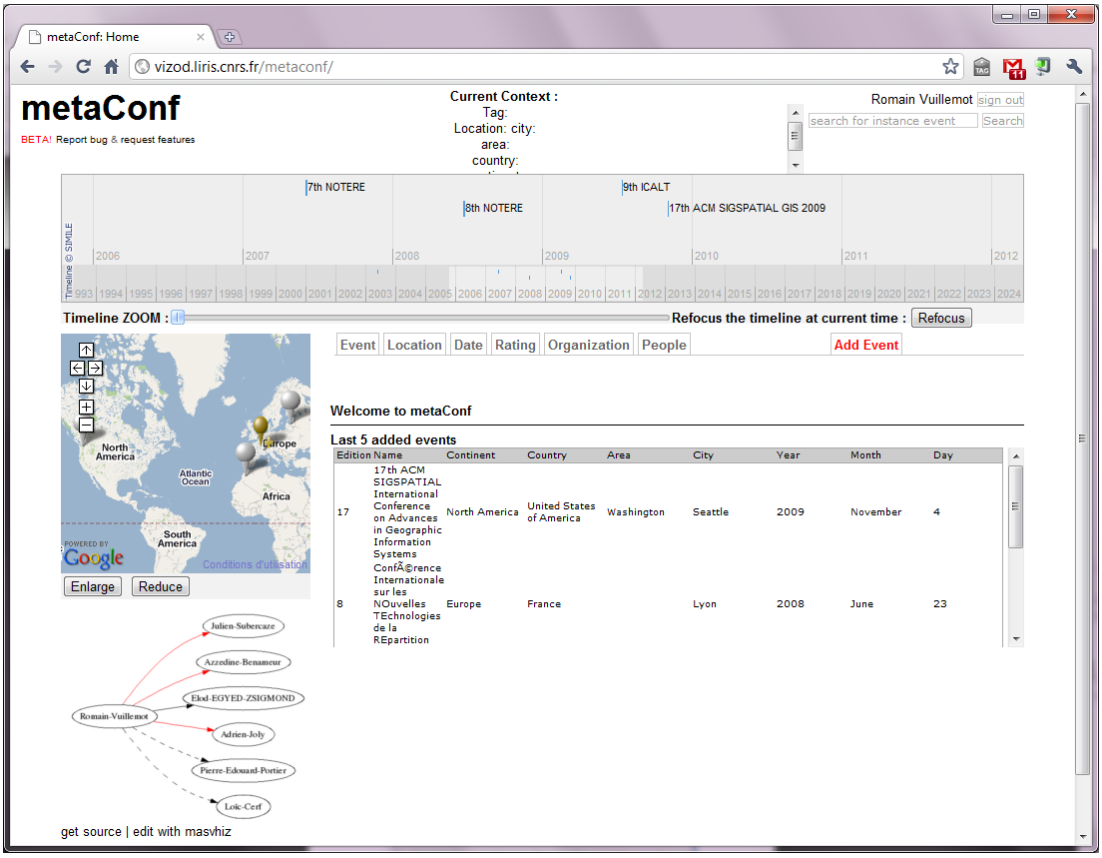


FIGURE 6.7 – Capture d'écran de la page principale de METACONF.

6.3.1 Flot de visualisation du graphe social

Plusieurs flots de visualisation sont inclus dans une page web : listes, cartes géographique, ligne temporelle, graphe social. Nous nous intéresserons en particulier au flot de visualisation du graphe social (illustration en bas à gauche sur la capture d'écran figure 6.7). Le flot de création du graphe social (annexe B.3) commence par l'extraction des données issues de la base de données `DatabaseHost_MYSQL`. La requête d'interrogation est composée d'informations communiquées dans l'en-tête HTTP d'appel au flot par le profil applicatif (`GraphRender_GRAPHVIZ` est interrogé par le profil applicatif, mais propage les en-têtes HTTP aux autre services). Le service `ParseHTTPHeader_VIZOD` extrait les en-têtes et extrait (service `urlParseMetaconf_VIZOD`) le nom de l'utilisateur. Ces informations contiennent l'URL de la page et le nom de la personne à partir de laquelle centrer le graphe social (qui est dans ce cas l'utilisateur connecté sur le site). Les étapes suivantes sont similaires au graphe généré avec VIZOD+GE pour la construction du graphe `GraphBuilder_VIZOD` et l'encodage visuel `ColorMapping_VIZOD`. Par contre la disposition du graphe est réalisée avec GRAPHVIZ [Gra10] `GraphLayout_GRAPHVIZ` et son rendu sous forme d'image `GraphRender_GRAPHVIZ`.

Afin d'améliorer la consistance visuelle de l'interface, nous avons introduit un service supplémentaire `GraphML2CSS_VIZOD` d'export de règle d'encodage visuel du graphe en un fichier CSS (*Cascading Style Sheets*, feuilles de style en cascade). Le résultat (figure 6.8) montre la similarité d'encodage visuel ainsi réalisée, entre les relations entre individus présents sur

le graphe, et sur la page HTML. Cela est permis par la présence de microformats[9] dans la page web. Ces microformats utilisent la balise "rel" invisible dans le rendu normal, mais qui permet d'encoder un ou plusieurs types de relation entre deux liens (par exemple la relation de type *"Friend"* code 5). Un rendu visuel est applicable à ces relations, en le spécifiant dans la feuille de style de rendu (code 6).

Code 5 Extrait de code HTML incluant l'attribut `rel` pour décrire le type de relation sociale.

```
<tr>
<td><a href="/metaconf/people/2/" rel="Friend ">Julien</a></td>
<td><a href="/metaconf/people/2/" rel="Friend ">Subercaze</a></td>
<td><a href="/metaconf/people/2/" rel="Friend ">Friend</a></td>
</tr>

<tr>
<td><a href="/metaconf/people/3/" rel="Friend ">Azzedine</a></td>
<td><a href="/metaconf/people/3/" rel="Friend ">Benameur</a></td>
<td><a href="/metaconf/people/3/" rel="Friend ">Friend</a></td>
</tr>

<tr>
<td><a href="/metaconf/people/8/" rel="Colleague ">Elod</a></td>
<td><a href="/metaconf/people/8/" rel="Colleague ">EGYED-ZSIGMOND</a></td>
<td><a href="/metaconf/people/8/" rel="Colleague ">Colleague</a></td>
</tr>
```

6.3.2 Couplage au navigateur et optimisation des transferts

L'image générée par le flot de visualisation du graphe social (image de type PNG) est inclue dans le navigateur web via une simple balise `<img>` standard qui contient l'URL du flot. Le graphe social étant très limité en taille, le temps de génération du flot est négligeable comparé au temps total de chargement de la page (en moyenne supérieur à 2s). A l'avenir, il sera nécessaire de charger l'image au moyen d'un mécanisme asynchrone afin de ne pas bloquer le temps de chargement de la page avec la création et le transfert de ce graphe. A noter que la construction de la page web est réalisée par un script PHP côté serveur qui génère la mise en page globale de la page et l'appel aux services. La page web n'est donc pas un service issu de VizOD (pour des raisons historiques de conception).

Concernant le temps de réactivité du navigateur web, celui-ci dépend du temps de chargement de la page. Pour mesurer cela, il est possible d'utiliser des applications tierces telles que le module de Mozilla Firefox, Firebug[10]. Ce module permet d'énumérer chronologiquement les appels HTTP du navigateur afin de charger une page complète. La figure 6.9 montre un exemple de transferts d'éléments pour un seul chargement de page. Chaque ligne

9. http://microformats.org/
10. http://getfirebug.com/

Code 6 Code CSS pour la visualisation sémantique.

```
a[rel~="Friend"] {
text-decoration:none;
border-bottom:1px solid red;
}

a[rel~="Colleague"] {
text-decoration:none;
border-bottom:1px solid black;
}

a[rel~="Met"] {
text-decoration:none;
border-bottom:1px dashed black;
}
```

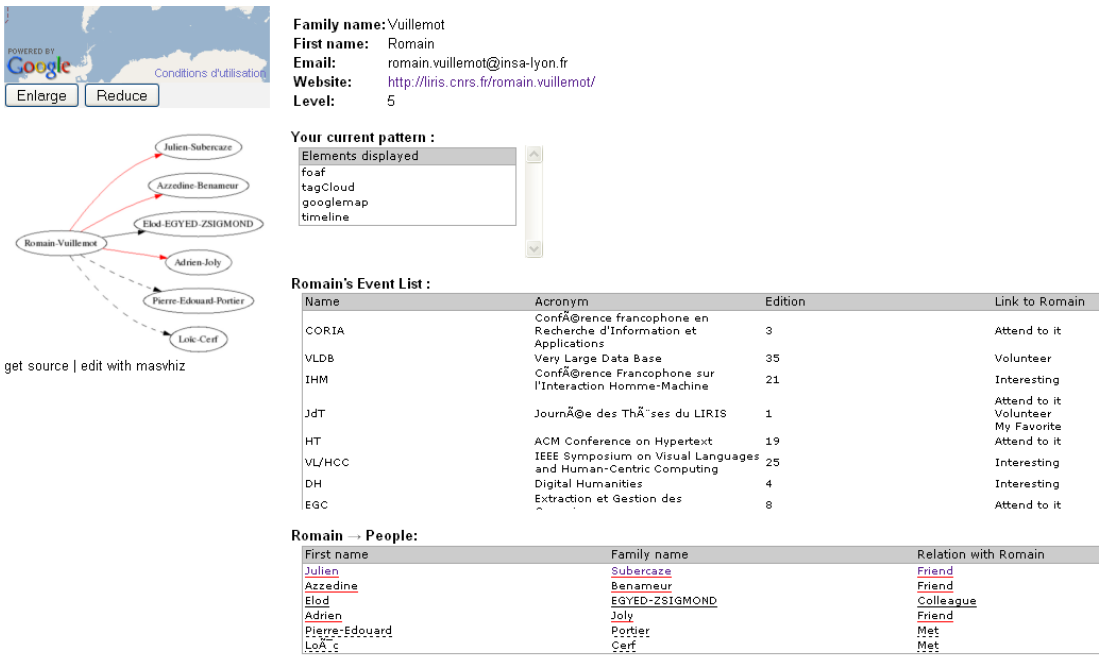


FIGURE 6.8 – Consistance de coloriage entre une liste et un graphe.

représente un appel GET soit à un élément de code (Javascript), soit de mise en page (CSS), ou soit à un service qui va générer des données. Nous voyons par exemple [11] que le téléchargement du fond de la carte GOOGLE MAP a nécessité 16*ms* et a une taille de 9*KB*. Mais pour la rendre interactive il est nécessaire de charger chaque élément qui constitue les annotations de cette carte, ainsi que le *profil applicatif* associé à ce service (autrement dit des fonctions javascript) qui permettront de zoomer et de se déplacer dans le plan de la carte. Au final, le temps de chargement complet de la page est de 2.11*s* pour un transfert total de 2.2*MB* de

11. A noter que la fonction du navigateur web de cache de données déjà transférées a été préalablement désactivée.

données. FIREBUG peut également afficher les en-têtes HTTP détaillés et les codes de retour des services.

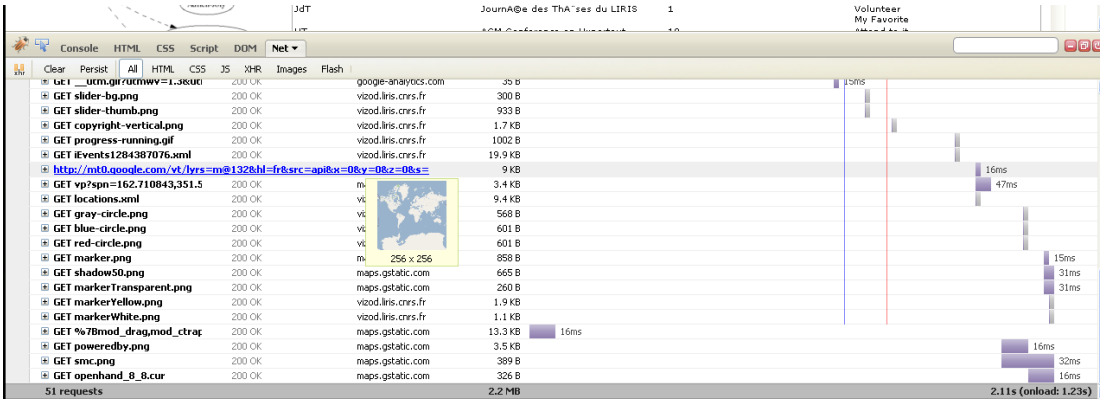


FIGURE 6.9 – Temps de chargement d'une page web avec FIREBUG.

6.3.3 Bilan et discussion

METACONF a permis de nous intéresser au profil applicatif largement répandu qu'est le navigateur web (restreint aux standards du web). Il est possible d'y inclure aussi bien des visualisations classiques telles que GOOGLE MAP, mais aussi des graphes issus de VIZOD. Nous avons aussi montré que des outils actuellement disponibles pour le développement et l'analyse du chargement de page web peuvent se révéler utiles afin de mieux comprendre le chargement des services utilisés. L'utilisation de librairies Javascript permet d'étendre le cadre applicatif basique du navigateur, avec de nouveaux widgets et interfaces graphiques. Une limite de ces *profils applicatifs* "additionnels" est qu'ils sont difficiles à évaluer en terme de performances et indépendamment d'un navigateur. En effet ils dépendent de l'implémentation du navigateur et des moteurs de rendu tels que WebKit [12] (utilisé dans Apple Safari et Google Chrome), et Gecko [13] (utilisé dans Mozilla Firefox), ainsi que de la configuration du navigateur chez le client (qui peut par exemple désactiver Javascript).

6.4 MosaiZ : exploration d'une collection d'images par similarité

MOSAIZ est une application développée dans le but d'explorer des collections d'images non-structurées à partir de la métaphore de la mosaïque d'image (figure 6.10). Une mosaïque d'image est une image globale constituée d'autres images qui lui sont localement similaires. Ainsi, une mosaïque d'images remplit de manière optimale et cohérente un plan avec d'autres images, sans superposition. Une mosaïque interactive permet à l'utilisateur de sélectionner une sous-image d'une image globale, qui deviendra le focus de zoom et ensuite une image zoomable elle-même [VR09a]. Cette succession d'interactions est permise par zoom sémantique, combiné à des transitions animées de déplacement. Cette application permet donc une

12. http://webkit.org/
13. https://developer.mozilla.org/en/Gecko

navigation par contenu des images, sans contrainte sur les images (comme des métadonnées, etc.). Le *profil applicatif* est une interface graphique zoomable, développée à partir de la bibliothèque PICCOLO2D [BGM04], et qui a nécessité un développement de profil applicatif spécifique (contrairement aux applications précédentes VIZOD+GE et METACONF). Nous allons nous intéresser aux points suivants liés au développement de l'application :

1. Flot de traitement d'image et création de mosaïques (section 6.4.1).
2. Graphe de scène d'une interface zoomable et performances (section 6.4.2).

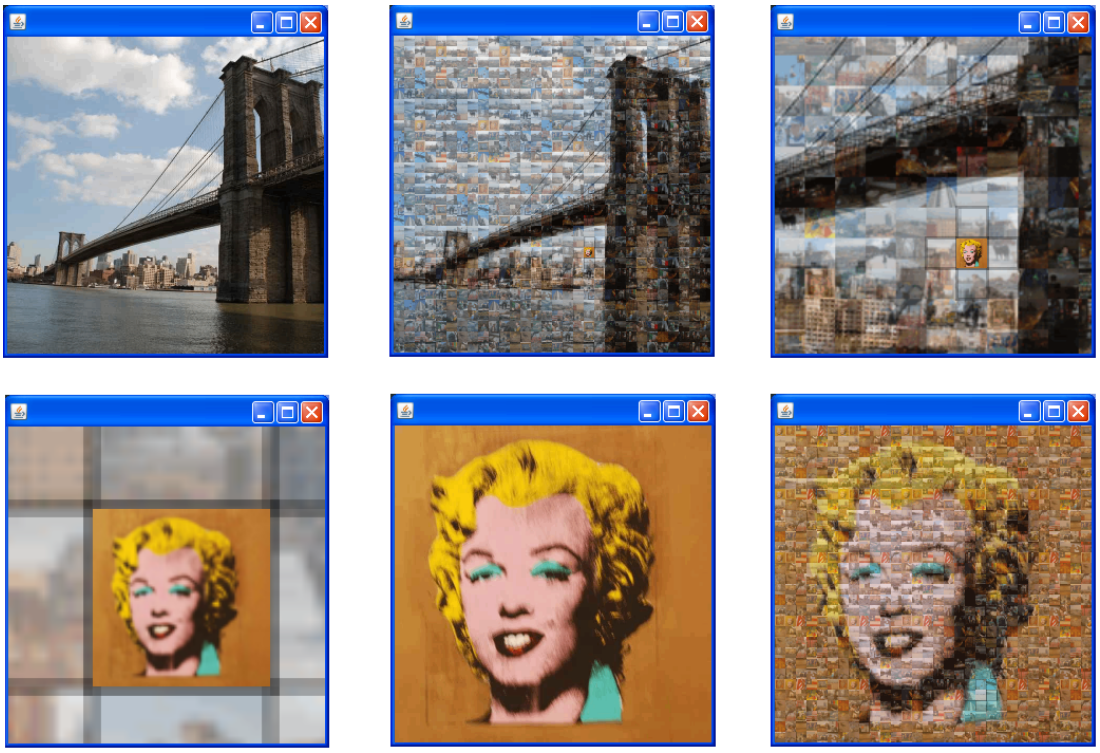


FIGURE 6.10 – Captures d'écrans de zoom progressif dans l'interface de MOSAIZ.

6.4.1 Flot de traitement d'images et création de mosaïques d'images

La préparation du flot de visualisation (annexe B.4) consiste à construire les différentes mosaïques d'images et à les connecter entre elles. Les images sont physiquement stockées sur un serveur externe, mais dont l'URL et certaines métadonnées (afin de faciliter les expérimentations) sont stockées dans une base de données `DatabaseHost_MYSQL`. La requête `StringBuilder_VIZOD` permet d'obtenir une série de 250 images. Chaque image est récupérée par `FetchImagePNG_VIZOD` et ensuite transformée avec `ProcessImage_OPENCV` [Ope10a] afin de pouvoir être inclues dans la grille d'une mosaïque. Cette phase s'assimile à une phase d'indexation et ces opérations n'ont besoin d'être réalisées qu'une seule fois, les images sont donc ensuite stockées sur un serveur intermédiaire `DataCache_VIZOD`.

Nous avons introduit un opérateur logique `ForEach_VIZOD` qui pour chaque image stockée en cache va générer une mosaïque avec le service `Mosaic_METAPIXEL` (qui utilise la

bibliothèque METAPIXEL [Met10] de création de mosaïques). Ce service possède un paramètre de taille de grille qui constitue la finesse (en nombre d'images) des grilles d'une mosaïque. Le graphique (figure 6.11) indique les différents temps de génération de cinq mosaïques en fonction de la taille de cette grille. Les variations sont importantes, et montrent qu'à partir d'une taille inférieure à 25×25 images il n'est pas possible de générer dynamiquement ces mosaïques (temps d'attente supérieur à $5s$) [14]. Cependant plus le maillage de la grille est fin, plus les mosaïques sont visuellement fidèles à l'image globale. Un compromis entre vitesse d'exécution et temps de chargement est à trouver.

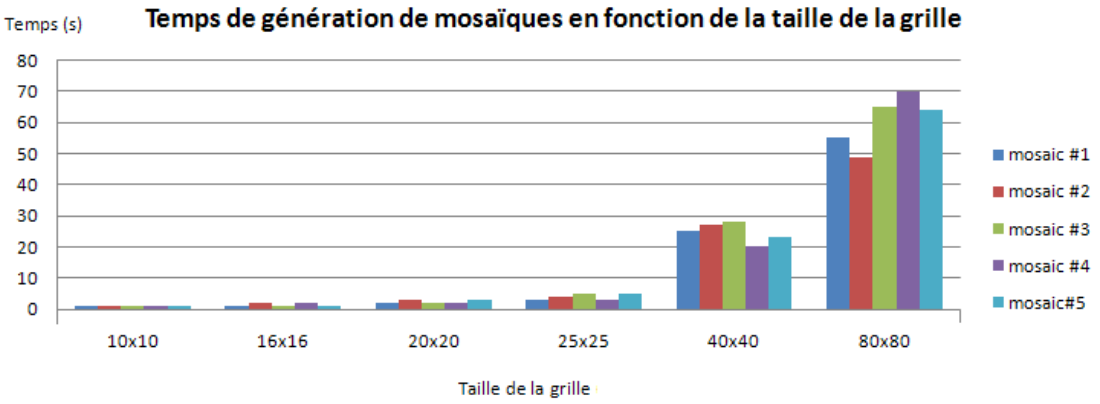


FIGURE 6.11 – Temps de génération de mosaïques d'images en fonction de la taille de la grille.

6.4.2 Graphe de Scène et performances

Le *graphe de scène* est composé de l'interconnexion des mosaïques entre elles (les noeuds étant les mosaïques, les arêtes les transitions entre les mosaïques). Ce graphe varie donc en fonction de la taille de la grille de la mosaïque, car un noeud sera connecté à autant de mosaïques que d'images qui composent sa grille. L'analyse de ce graphe permet d'assister la navigation [VR09a], comme par exemple pouvoir reconstruire l'*historique* (parcourir les images déjà visualisées), permettre le *rendez-vous* (revenir à une position précédente dans la MOSAIZ définie par l'utilisateur) ou le *parcours complet* (afin de vérifier si l'utilisateur a parcouru la totalité du jeu de données, et donc la totalité du graphe).

L'étude des performances côté profil applicatif a été nécessaire dans la phase de design afin de concevoir une interface sans baisse de framerate. La qualité de cet indicateur est à une autre échelle bien plus fine, au niveau de la milliseconde. Nous avons remarqué [VR09a] que des chutes ponctuelles de framerate (mais tout de même perceptibles par l'utilisateur) étaient enregistrées (figure 6.12) au fil de l'usage de l'application. Ces chutes sont liées au chargement d'images qui apparaissent au fil du zoom sémantique. Il s'agit d'un problème technique, qui peut être résolu en fonction d'alternatives de design techniques elles aussi (charger toutes les images au début, chargement progressif en arrière plan même si les images ne sont pas encore visitées) mais également en influant sur la qualité de la représentation visuelle des images (baisser la taille des images, leur résolution, nombre de couleurs,

14. Nous avons constaté qu'un temps d'attente supérieur à $1s$ est acceptable pour l'utilisateur étant donné qu'il s'agit d'une interface zoomable et donc non-bloquante si une donnée est manquante.

compression avec perte, etc.) afin de diminuer les transferts réseaux.

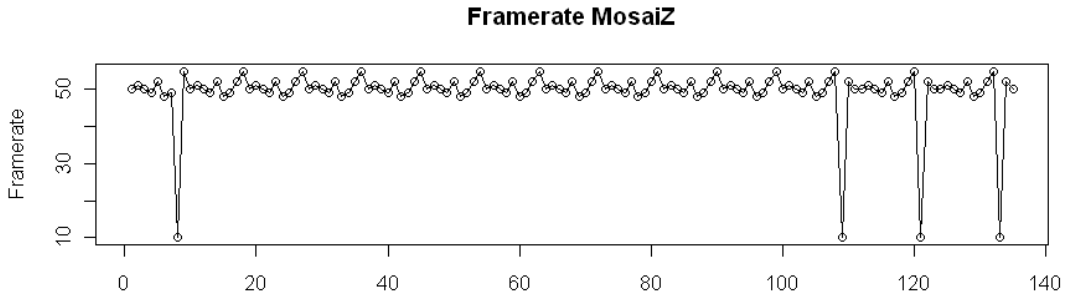


FIGURE 6.12 – Variation de framerate lors de l'usage d'une MOSAIZ.

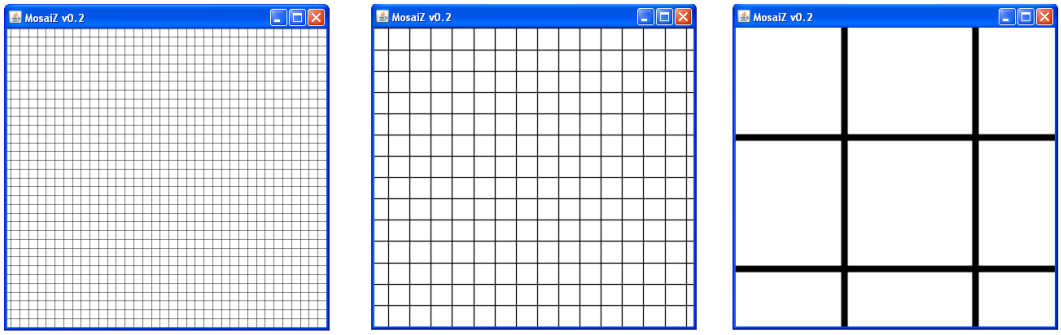


FIGURE 6.13 – Test à vide de l'application MOSAIZ.

Le test en "maquette fil de fer" (*wireframe*) (figure 6.13) de l'application MOSAIZ est possible grâce au découplage entre les vues et le profil applicatif, et permet de réaliser des tests de performance avec le chargement progressif de données (en nombre et en résolution).

6.4.3 Bilan et discussion

Nous avons développé avec MOSAIZ un profil applicatif d'interface zoomable pour naviguer dans des mosaïques d'images. Le rôle de l'architecture VIZOD a été de permettre de comparer les temps de traitement en fonction de différentes qualités du rendu visuel (à savoir la grille des mosaïques), et d'étudier le framerate d'un profil applicatif. Cet indicateur a permis d'envisager des alternatives possibles de réduction de qualité visuelle qu'il serait possible de rapidement tester en ne modifiant les paramètres que du service de rendu (`ProcessImage_OPENCV`) ou en incluant d'autres fonctions (par exemple la compression) de la bibliothèque OPENCV [Ope10a]. Les flots de visualisations sont des supports qui pourront à l'avenir permettre de tester rapidement d'autres alternatives de design, comme différentes bibliothèques de mosaïques d'images (ce qui n'impliquerait a priori que le changement du service `Mosaic_METAPIXEL`).

6.5 POSvis : caractérisation d'entités par nuages de mots clés

La dernière application que nous avons développé s'appelle POSvis (*Part-Of-Speech VISualization*) et a été réalisée au laboratoire HCIL [15] de l'Université de Maryland, en coopération avec une chercheuse en littérature anglaise, dans le cadre du projet MONK [16]. Notre chercheuse était intéressée par l'analyse du style littéraire utilisé pour caractériser des entités nommées (par exemple des noms, prénoms, etc.) dans des ouvrages littéraires du début du 20ème siècle [VCPV09]. Ces ouvrages sont issus de collections d'archives de livres numérisés, et nous avons travaillé sur la version du texte annotée avec les entités nommées et l'étiquetage grammatical (*Part-of-Speech*) (extrait de texte annoté code 7) [17].

FIGURE 6.14 – Capture d'écran de l'interface principale de POSvis.

L'interface POSvis (capture d'écran figure 6.14) permet d'explorer ce texte annoté, au moyen de requêtes visuelles itératives (grâce aux widgets et aux listes). A chaque interaction, il est possible de visualiser les résultats intermédiaires sous forme de nuage de mots et de réseau de co-occurrence [VCPK09]. Ces différentes visualisations permettent à l'utilisateur de réaliser une *lecture distante* (*distant reading*) et ainsi être en mesure d'analyser un texte littéraires dans sa globalité ou de manière quantitative.

15. http://www.cs.umd.edu/hcil/
16. MONK vise à améliorer la compréhension des textes littéraires, en regroupant des compétences pluridisciplinaires de méthodes automatiques de traitement de texte et de représentation http://monkproject.org/
17. Une liste détaillée de la signification des annotations est disponible sur le site http://nlp.stanford.edu/software/tagger.shtml

6.5.1 Détails du jeu de données

Le jeu de données est constitué du texte *"The Making of Americans"*, une nouvelle de Gertrude Stein, composée de 9 chapitres et de 517 027 mots. Ce texte possède une structure en chapitres, paragraphes et phrases que nous avons conservé, afin de permettre des requêtes de co-occurrences d'entités (à savoir la recherche d'entités apparaissant dans un même paragraphe ou chapitre). La structure du texte permet également de réaliser une vue globale (bandeau supérieur figure 6.14) qui indique la distribution des mots en fonction des paragraphes et des chapitres.

Le texte a été préalablement annoté au moyen de la plateforme SEASR [LAC+08]. Deux types d'annotation sont disponibles : en fonction de la catégorie grammaticale (extrait de texte annoté code 7 : `/VB` désigne un verbe `/VBD` un verbe au temps passé, etc.), et en fonction du type d'entité (un signe `$` est présent s'il s'agit d'une entité). A noter que contrairement aux phases d'indexation textuelles habituelles, la racine des mots n'a pas été extraite, et les mots muets (et, ou, ..) ont été conservés (notre experte estime que ces mots sont utiles pour sa tâche).

Code 7 Extrait de texte annoté en fonction de la catégorie grammaticale et des entités.

```
The/DT Making/NN of/IN Americans/NNPS


Once/RB an/DT angry/JJ man/NN dragged/VBD his/PRP$ father/NN along/IN
the/DT ground/NN through/IN his/PRP$ own/JJ orchard./NN "Stop!"/NN
cried/VBD the/DT groaning/VBG old/JJ man/NN at/IN last,/JJ "Stop!/NN
I/PRP did/VBD not/RB drag/VB my/PRP$ father/NN beyond/IN this/DT tree."/NN
```

6.5.2 Transformation d'un script de génération de nuage de mots en flots

POSvis a été développé sous la forme d'architecture classique de type client/serveur[18] (détails de l'architecture dans [VCPK09]). Le nuage de mots est généré par un script PHP `tagcloud.php`, et a pour paramètres la requête itérative composée au moyen des widgets de l'application Java. La limite de cette approche est qu'en cas de requête complexe et de temps de calcul de réponse longs, l'application ne sera pas réactive.

Notre approche est de transformer cette application dans le cadre de VizOD, et donc d'utiliser l'API [Scu09] d'adaptation de programme pour le script `tagcloud.php` et ainsi être en mesure d'analyser son usage, mais sans modifier le profil applicatif POSvis : le format d'URL actuel (code 8) doit être conservé. Nous avons créé un flot (annexe B.5) dont le service `URLBuilder_VIZOD` simulera le comportement de `tagcloud.php`, à savoir communiquera les paramètres de l'URL (code 8) à `tagcloud.php` (au moyen d'une chaine de paramètres (*query string*) classique dans l'URL) et affichera le retour de ce script.

Nous avons également rajouté une chaine de traitement permettant de "surcharger" les paramètres de l'URL créée par `URLBuilder_VIZOD`. Cela permettra de forcer la valeur

18. Car l'application n'a pas été développée à l'origine dans le cadre de VizOD.

de certains paramètres de filtrage et d'encodage visuel, si par exemple l'utilisateur a des préférences. Ces préférences sont définies par le service `ParamBuilder_VIZOD` qui permet le filtrage par catégories grammaticales (qui sont extraites depuis la base de données `DatabaseHost_MYSQL`). Et par le service `TagCloudParams_VIZOD` qui défini l'encodage visuel : `Order By` défini l'ordre des mots dans le nuage, `Size By` la taille et `Color By` la couleur selon une légende par défaut définie.

La sortie du service `URLBuilder_VIZOD` est à connecter directement dans POSvis.

Code 8 URL d'un nuage de mot clé généré par POSvis.

```
// Informations sur la source des données
http://HOST/posvis/tagcloud.php?chapter=1&collection=1
// Informations sur la sélection des données
 &startSelectionA=0&endSelectionA=4892&startSelectionB=4893&endSelectionB=6112
// Correspondance données/attributs visuels
 &orderBy=appearence&sizeBy=frequencyInB&colorBy=frequencyInA
// Seuillage pour permettre une mise en avant des mots importants
 &minSize=50&maxSize=500&limitMinSize=50
```

L'utilisateur n'a plus qu'à paramétrer POSvis pour se connecter au service `URLBuilder_VIZOD`, et ainsi ses préférences définies via les services `ParamBuilder_VIZOD` et `TagCloudParams_VIZOD` seront automatiquement prises en compte dans l'application.

6.5.3 Bilan et discussion

Nous avons présenté POSvis, une application permettant d'explorer un texte annoté d'entités et de catégories grammaticales. Nous avons montré comment une vue de type nuage de mot, déjà inclue dans cette application peut en être isolée et paramétrée à nouveau dans un flot de visualisation. L'usage de ce flot pourra être par la suite analysé au moyen de métriques de VizOD (temps de génération, etc.) et être exportable dans toute autre application (comme par exemple un navigateur web), ce qu'il n'était pas initialement possible de faire initialement avec POSvis.

6.6 Conclusion des applications

Nous avons montré la mise en oeuvre de VizOD dans le cadre de développements d'applications complètes. De manière générale les applications développées offrent une relative diversité en termes de jeux de données, interfaces graphiques et widgets interactifs. Le tableau 6.2 regroupe les différents langages utilisés pour développer le profil applicatif (ou créer des services exécutables avec un profil existant).

Le nombre de lignes de code et de services utilisés (tableau 6.3) donnent un aperçu quantitatif des programmes. Mais ils ne représentent pas la complexité (ou non) des développements, ni-même des services qui ont été adaptés.

Application	Langage (Bibliothèque)	Type d'interface graphique
VizOD+GE	PHP, KML, XML	Interface multi-résolutions
metaConf	Javascript (mootools [Moo10]), PHP, HTML	Vues par facettes
MosaiZ	Java (Piccolo2D [BGM04])	Interface zoomable
POSvis	Java (Lucene, Prefuse)	Vues multiples coordonnées

Table 6.2 – Tableau récapitulatif des applications et des langages utilisés.

Application	Lignes de code	Nombre de services
VizOD+GE	82	9
metaConf	5399	10
MosaiZ	2352	9
POSvis	8848	6

Table 6.3 – Tableau récapitulatif des lignes de code et nombre de services développés.

Dans la partie suivante (section 7) nous allons discuter des axes d'optimisation et des limites de l'architecture VizOD que nous avons pu identifier à partir de notre expérience de développement.

Chapitre 7

Discussions

Sommaire

7.1 Rappel des objectifs de nos travaux

Pour rappel, les trois objectifs de nos travaux étaient (section 1.3) :

- **Identifier les dimensions de l'espace de design des représentations visuelles interactives, ainsi que leurs dépendances (partie I).**
- **Proposer un cadre complet de conception, ainsi qu'une architecture opérationnelle (chapitre 5).**
- **Développer des applications à partir de cette architecture (chapitre 6).**

Dans ce chapitre nous revenons sur notre problématique initiale, à savoir : "favoriser la création et le partage de représentations visuelles d'information" (section 1.2). Nous discutons dans quelle mesure VizOD a permis d'appréhender le manque d'*interopérabilité* (section 7.2), de *flexibilité* (section 7.2.2) et d'*adaptabilité* (section 7.2.3) la création et le partage de représentations visuelles interactives, et quelles sont les limites de cette architecture (section 7.3).

7.2 Discussions

7.2.1 Interopérabilité

L'*interopérabilité* permet à un utilisateur de travailler dans le contexte d'une application, et ensuite d'utiliser ses résultats dans le contexte d'une autre application tout en se concentrant au maximum sur sa tâche [SPCJ09].

L'API DATASPACE [Scu09] permet de standardiser les interfaces de communication entre les services eux-mêmes, et entre les flots et les profils applicatifs. En cas de non-compatibilité entre eux, des services de conversion de format (comme les services `GraphML2Dot_VIZOD` ou `GraphML2CSS_VIZOD`) peuvent être développés et partagés pour y remédier. Un utilisateur peut ainsi utiliser MASHVIZ afin de rapidement recomposer un flot et le rendre compatible pour une nouvelle application (en faisant l'hypothèse que la base de service contienne le ou les services de conversion). La forme d'interropérabilité la plus courante et qui est systématique dans VizOD est l'export des données issues de services au moyen d'une URL, ce qui permet au flot d'être partagé à l'échelle d'internet et permettre une forme de navigation sociale dans les données [HHC+08].

L'interopérabilité de VizOD permet la création rapide de nouvelles applications. Par exemple le graphe social de l'application METACONF (section 6.3) qui était généré avec la bibliothèque GRAPHVIZ [Gra10] peut être visualisé avec une autre bibliothèque de graphe (LGL [ADWM04]) et inclus dans VizOD+GE (figure 7.1).

Nous avons également étendu l'usage de VizOD aux téléphones mobiles. En effet, un téléphone mobile possède un *profil applicatif* à part entière. Nous avons pour cela utilisé l'application IPHONE LAYAR [Lay10] de réalité augmentée (section 2.2.2) qui permet d'enrichir visuellement le flux de la caméra de l'IPHONE avec des données (image, texte) géolocalisées. Ces données apparaîtront si leur géo-coding est situé dans le champ de vue du flux de la caméra (elle-même est géolocalisé grâce au GPS de l'appareil). Nous avons pu (en moins

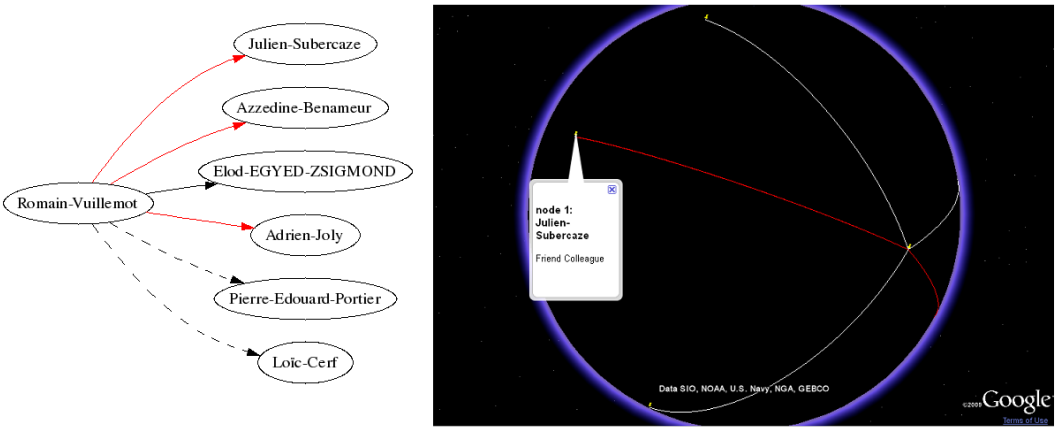


FIGURE 7.1 – Graphe social de METACONF visualisé dans VIZOD+GE.

d'une heure, l'API de LAYAR étant relativement simple) inclure un flot de visualisation exécutable avec l'application LAYAR, à savoir une image géo-codée. La figure 7.2 illustre les vues possibles à partir de l' IPHONE de cette visualisation géocodée (de gauche à droite) : flux augmenté de la caméra avec un disque noir qui indique la position de la visualisation, géolocalisation de la visualisation sur une carte et liste des éléments géo-codés dans un voisinage proche. A noter que le flot de visualisation que nous avons utilisé un graphe introduit dans l'application d'exemple (section 5.4), et cette application peut être utilisée pour régénérer le graphe et le mettre à jour dans l'IPHONE. L'usage du flot de visualisation par l'application IPHONE est analysable avec les métriques quantitatives de VIZOD et il est donc possible de manière non-intrusive de connaitre par exemple le nombre d'accès à la visualisation.

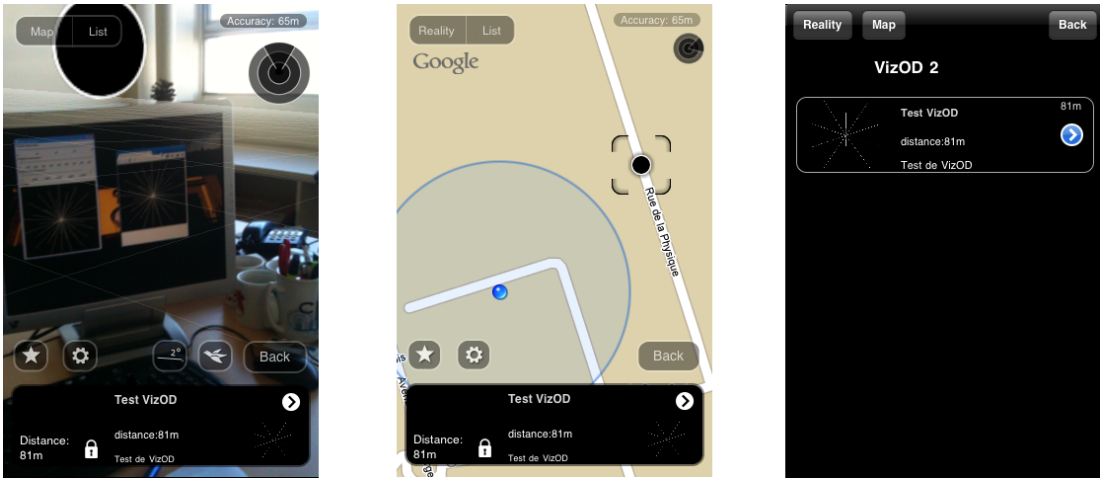


FIGURE 7.2 – Intégration de VIZOD dans l'application IPHONE LAYAR.

Explorer de nouveaux profils applicatifs qui permettent la navigation dans un espace (réel ou virtuel) est une perspective de nos travaux (figure 7.3).

La limite de l'interopérabilité réside dans l'adaptation des programmes (*wrapping*) qui

FIGURE 7.3 – Profils applicatifs potentiels d'interaction avec les visualisations.

ne permet pas (à ce jour) d'utiliser les programmes paramétrables sous forme d'interface graphique uniquement. Par exemple les visualisations de MANYEYES sont certes disponibles sous forme d'URL, mais il n'est possible de changer les paramètres de visualisation autrement qu'avec les widgets de l'interface Java. Il n'est pas possible non plus d'utiliser un logiciel tel que MICROSOFT EXCEL [Mic10a] : l'utilisateur doit alors réaliser des opérations manuelles et graphiques d'ouverture/fermeture de programmes pour convertir ses données.

7.2.2 Flexibilité

La *flexibilité* est la capacité d'un programme à fonctionner même si une nouvelle tâche, type d'interface, d'interaction ou de visualisation apparait (ou disparait).

Le principal point lié à la flexibilité que nous allons aborder est celui de la variation de la disponibilité des services et de leur accessibilité (via le réseau). Un service peut ne plus être accessible si par exemple le programme qu'il adapte est dans un état d'erreur ou si les ressources de la machine qui l'héberge sont insuffisantes. Un service peut aussi ne plus être accessible si la connexion réseau est coupée. L'analyse du temps de réponse d'un service (exemple de l'enregistrement du temps de réponse d'un service figure 7.4) [1] est un moyen de connaître sa disponibilité ou pas, et sa variation au fil du temps.

Les codes de retour des services (définis dans la RFC 2616 [FGM+99], tel que le `404 Not Found` ressource non trouvée) sont un moyen pour les services de communiquer au profil applicatif leur état. Le profil applicatif peut ainsi notifier l'utilisateur qu'une visualisation n'est pas disponible ou une erreur éventuelle (exemple de MANYEYES [Man09] (figure 7.5). Certes le message d'erreur n'est pas explicite et n'indique pas la prochaine étape à réaliser, mais une page web s'affiche et l'utilisateur a un aperçu de l'état du système distant.

Les systèmes MICROSOFT PHOTOSYNTH [Mic10b] et GOOGLE MAPS [Goo10d] (figure 1.4) proposent un découplage fort entre les vues et le profil applicatif (figure 1.4). Ainsi même si les vues ne sont pas disponibles, un aperçu de celles-ci est affiché : les points d'intérêt de la scène avec MICROSOFT PHOTOSYNTH ou le tracé d'une rue avec GOOGLE MAP. L'utilisateur

1. Réalisé avec l'outil RRDTOOLS d'enregistrement et de visualisation de journaux http://www.mrtg.org/rrdtool/

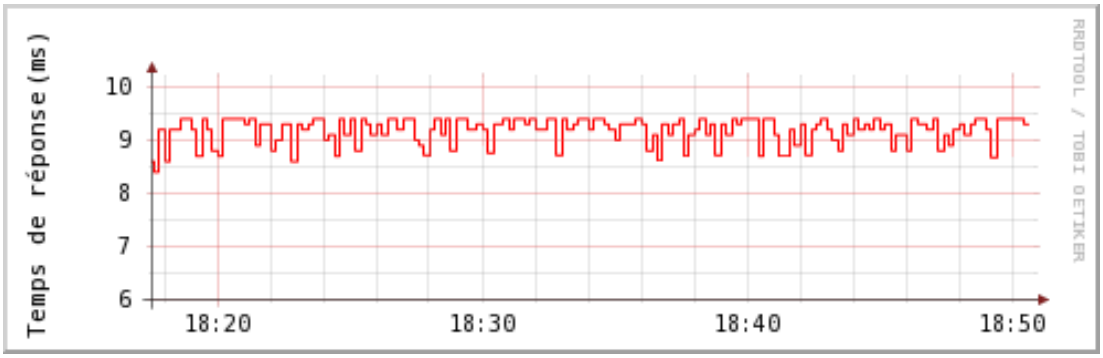


FIGURE 7.4 – Enregistrement du temps de réponse d'un service.

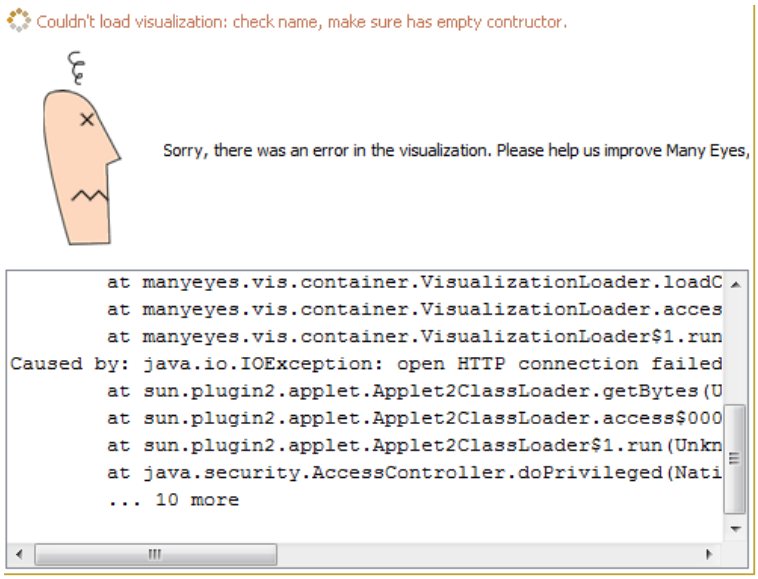


FIGURE 7.5 – Exemple d'erreur de visualisation sur le site de ManyEyes [Man09].

peut rester concentré sur sa tâche, et continuer à changer de point de vue même si les données ne sont que partielles.

Le temps de chargement des données doit aussi être communiqué à l'utilisateur, comme par exemple le chargement d'images gigapixel [2] qui est long et qui se réalise en fonction du champ de vision de l'utilisateur. La capture d'écran (figure 7.6) [3] illustre une image en cours de chargement et l'utilisation de variables visuelles (flou) et de glyphes animés (barre de chargement, icône d'attente) afin d'indiquer à l'utilisateur l'état du système.

A l'inverse, une page de résultats du moteur de recherche GOOGLE (figure 7.7) qui contient une GOOGLE MAPS [Goo10d] est très rapide, mais n'affiche pas toutes les données relatives à la carte. En effet, la page de résultats ne contiendra qu'un aperçu (image) de celle-ci et non le *profil applicatif* (scripts Javascript), ni même les annotations détaillées sur

2. http://en.wikipedia.org/wiki/Gigapixel_image
3. http://www.leprogres.fr/fr/balade_virtuelle/fourviere/index.html

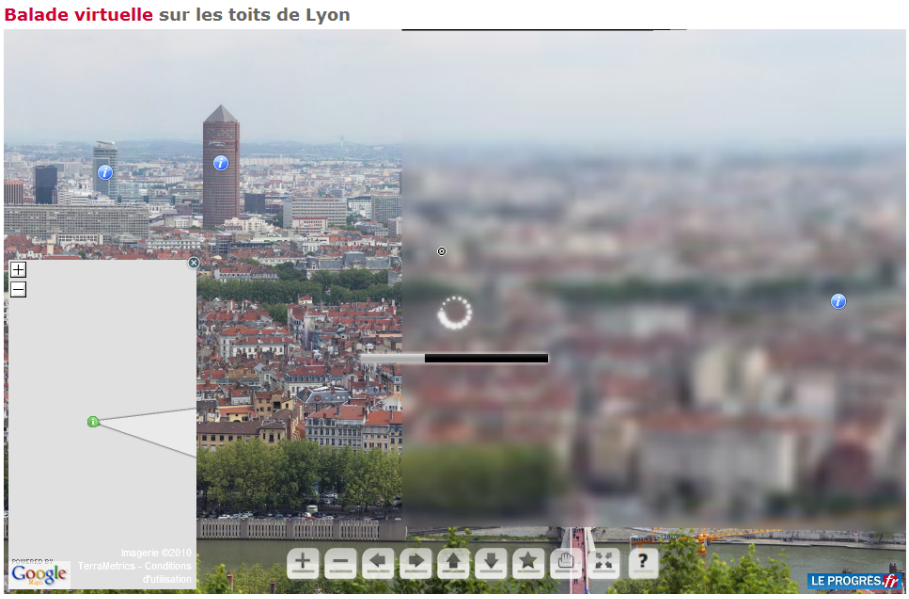


FIGURE 7.6 – Capture d'écran de chargement d'une image de type gigapixel.

celle-ci. Pour que la carte devienne interactive et que toutes les données soient chargées, l'utilisateur devra cliquer sur la carte dans la page de résultats et une nouvelle page, plus longue à charger cette fois, chargera les scripts d'interaction avec la carte. Dans cet exemple le temps de chargement des données dépend de la tâche à réaliser : dans le premier cas il s'agit de l'exploration rapide de l'espace de données, dans le deuxième cas de la navigation dans l'espace visuel.

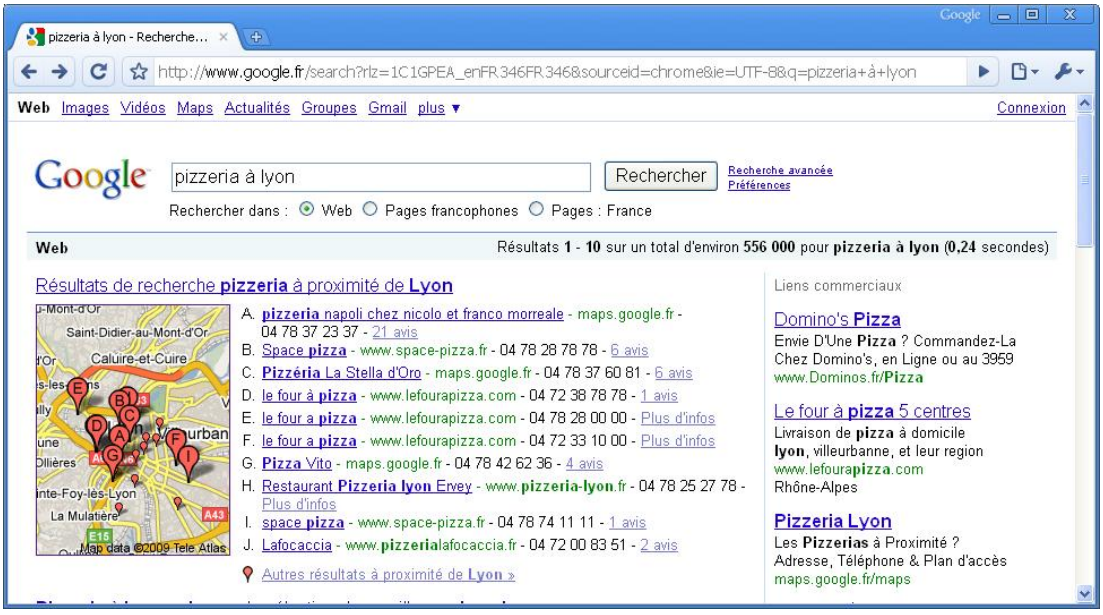


FIGURE 7.7 – Résultats d'une requête Google [Goo10a] qui inclue une GOOGLE MAP [Goo10d].

Nous proposons trois stratégies de design pour communiquer la non disponibilité des données d'un flot de visualisation [4] :

1. Afficher une image ou une chaine de caractères générique indiquant l'indisponibilité (figure 7.8).
2. Afficher une image ou une chaine de caractères indiquant le temps estimé d'indisponibilité où le code de retour du service.
3. Afficher une donnée issue du cache similaire à la dernière requête reçu (si le contexte est le même).

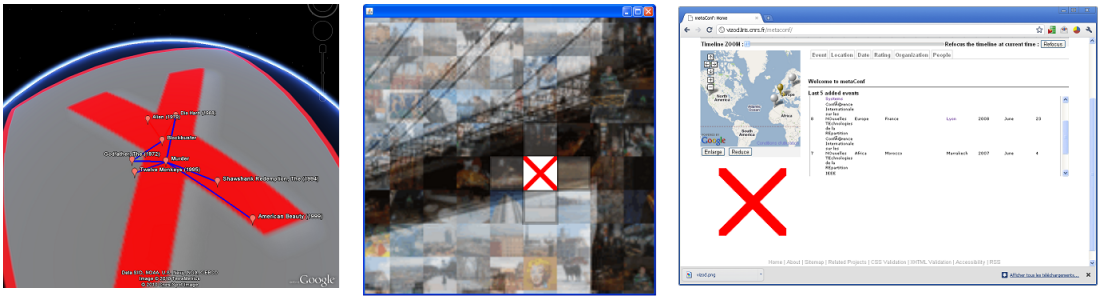


FIGURE 7.8 – Une stratégie de notification de vues non-disponibles avec VIZOD.

La dernière proposition doit être accompagné d'un design qui permet d'encoder visuellement la fraicheur des données, afin d'indiquer qu'elles ne sont pas à jour : rendre l'image floue, la transformer en noir et blanc, voire ajouter un symbole de date de dernière génération de l'image.

7.2.3 Adaptabilité

L'adaptabilité d'une interface est la possibilité de personnalisation d'un système de manière explicite par un utilisateur, par exemple au moyen d'un menu de configuration. Cette personnalisation peut être aussi bien en fonction de préférences de l'utilisateur, qu'en fonction de contraintes techniques.

L'interface MASHVIZ permet la configuration et la personnalisation des visualisations de manière accessible et ne requiert pas de compétence technique pointue [5]. Les profils applicatifs n'étant pas souvent adaptables, ils peuvent le devenir par variation des paramètres des flots de visualisation. Par exemple les codes de couleur de visualisation du graphe social dans METACONF (section 6.2) ont été composés dans MASHVIZ, et ensuite inclus dans le navigateur (sans avoir eu à saisir ces paramètres dans le site web). De même dans l'application POSVIS (section 6.5), le flot de visualisation a permis de surcharger la représentation du nuage de mots au moyen de MASHVIZ. Et ces nouveaux paramètres sont sauvegardables dans MASVHIZ, alors qu'ils ne le sont pas dans POSVIS.

4. A noter que ces stratégies peuvent être aussi utilisée pour la calibration visuelle du système en incluant une mire (section 2.2.1).
5. Une étude récente [ZR08] indique que les utilisateurs sont de plus en plus enclins à réaliser leur propre personnalisation au moyen notamment d'interfaces de programmation visuelle.

Même si certains paramétrages de flots ne sont pas disponibles (car le flot ou une fonctionalité précise n'existent pas), nous avons introduit [VR09b] la propriété de créer des flots "prototypes" qui permettent de simuler des services et des flots (figure 7.9). L'utilisateur peut ainsi communiquer au designer ses besoins de manière simple et tout en restant dans le cadre de MASHVIZ.

Une approche générative pourrait être réalisée à partir de MASHVIZ, afin de générer toutes les abstractions visuelles possibles d'un flot (selon une variable graphique comme la couleur) et laisser l'utilisateur choisir. Nous avions proposé dans [VCPK09] une navigation par nuage de mots-clés en "mosaïque" qui affiche 9 nuages pour autant de catégories grammaticales et de couleurs (figure 7.10). Il est donc possible de comparer différentes alternatives de design et choisir celle la plus pertinente (pour cela il faudrait inclure les mêmes données dans les nuages). Il est tout à fait envisageable de construire des matrices où chaque ligne et chaque colonne est une variation d'une variable graphique, de la taille du jeu de données ou du point de vue, afin d'avoir un aperçu des différentes alternatives de conception.

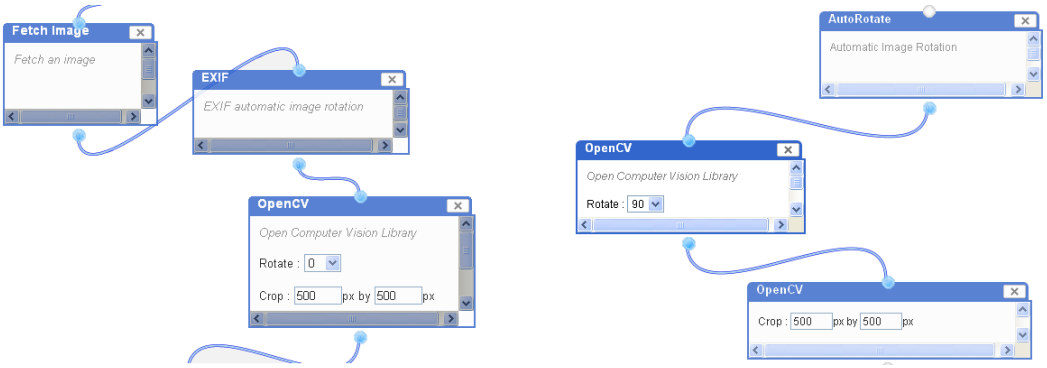


FIGURE 7.9 – Exemples de parties de flots "prototypes" d'un utilisateur.

7.3 Conclusion et limites de VizOD

7.3.1 Conclusion

Nous positionnons VIZOD comme un cadre structurant de développement, d'évaluation (au moyen de métriques et d'un annuaire de services et d'usages de services) et de comparaison de différentes alternatives de design d'application visuelles interactives.

Ce cadre nous a permis une réduction du temps de développement, par exemple dans la réutilisation de services identiques dans des applications différentes ou dans le choix de stratégies de visualisation en fonction du temps d'exécution des services. Cette réduction du temps de développement est un des principaux objectifs des systèmes [Ols07].

VIZOD permet aussi de capitaliser de manière *opérationnelle* des choix de design. Ces choix de design sont aujourd'hui souvent disponibles sous forme de retour d'expériences "qualitatifs" ou sous forme de benchmark[6], et il n'est pas toujours possible d'avoir accès

6. Un exemple de benchmark http://hcil.cs.umd.edu/localphp/hcil/vast/archive/index.php

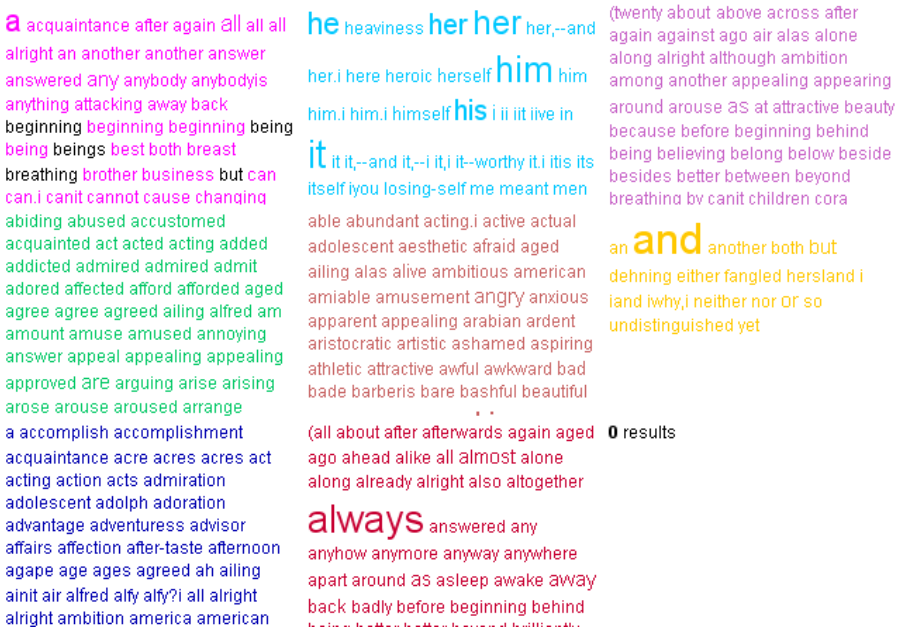


FIGURE 7.10 – Génération d'alternatives de design (en terme de couleur) d'un nuage de mot clé.

aux programmes de traitement de données et de représentation visuelle. L'interface MASHVIZ permet d'avoir accès à ces informations, et de rapidement tester les représentations visuelles correspondantes avec ses données ou ses préférences visuelles.

Une évaluation qualitative de VIZOD serait à réaliser, par exemple sous les trois angles [BH09] : d'*expressivité* (peut-on le construire ?) d'*efficience* (combien de temps cela va-t-il nécessiter ?) et d'*accessibilité* (est-ce que je sais comment ?).

7.3.2 Limites de VizOD

Une première limite est le manque de temps et de ressources pour ouvrir le cadre de VIZOD à d'autres développeurs. A noter que par contre l'API [Scu09] sur laquelle repose VIZOD est également utilisée dans d'autres contextes (intégration de données hétérogènes, requêtes continues) et fait l'objet d'un usage actif et d'une maintenance réactive. Certaines contraintes techniques pourront donc être résolues par le biais de ce projet et bénéficier à notre architecture VIZOD.

Les principales limites que nous avons pu identifier à partir de notre expérience de développement de VIZOD sont les suivantes :

– La mise en réseau implique une analyse de l'usage de l'utilisateur, et donc un traçage systématique de son activité. Un mécanisme de gestion de la privée doit être mise en place en accord avec les utilisateurs. A noter que le transfert des programmes sous forme de services fait perdre à l'utilisateur la propriété de l'application, ainsi que sa

liberté de modifier le code[7].

- Les temps de réponse du réseau et de génération de flots (au moins supérieur à 100*ms*) ne permettent pas de générer et transmettre des *rétroactions* visuelles rapides (inférieures à 100*ms*), et doivent donc être générées par le profil applicatif. De manière générale certains éléments de design sont déjà inclus dans les widgets et interfaces graphique, et il est difficile d'en dissocier la vue (le modèle MVC n'étant pas toujours respecté). Les requêtes *dynamiques* (section 3.2.1) restent limitées à l'abstraction de données (recommandation de résultats, comme pour compléter la saisie d'une requête (illustration figure 3.5)). Si une requête implique des traitements au niveau de l'abstraction visuelle et du rendu il semble plus pertinent de réaliser des requêtes de *prévisualisation* ou par *composition*. Ou alors de transférer les services de visualisation et de rendu sur le profil applicatif.
- La création libre de services avec l'API [Scu09] tend à produire des services qui n'ont pas le même niveau de granularité : une standardisation (via une taxonomie par exemple) du modèle de référence semble nécessaire en découpages fonctionnels.
- De même les entrées et sorties des services sont à formaliser : les services et flots développés (annexe B) n'ont pas de structure de données uniforme en entrée et en sortie. Il est nécessaire également de séparer les structures de données complexes (images, tuples, etc.) en structures plus simples (chaines de caractères, entiers, etc.). L'impact sur l'usage de MASHVIZ risque d'être non-négligeable et d'en limité la facilité d'usage.
- Les évaluations effectuées ont été réalisées en "laboratoire", sur des serveurs locaux ou peu distants, qui ne permettent pas de prendre en compte la réalité d'exécution et de transfert des données.
- La pause, redémarrage et la répétition de traitements sont difficiles dans un contexte de flots, de même que l'examen de la pile d'exécution des programmes (afin d'analyser la valeur des variables et des registres) n'est pas accessible (hors mis pour l'hébergeur du service).
- Le passage à l'échelle est identifié comme un des problèmes majeurs en visualisation d'information [Che05], ainsi qu'en visualisation en général [Hib04]. Nous pensons que la distribution des calculs avec VIZOD permet potentiellement de réduire les temps de chargement, mais un test de transfert de grands volumes de données à un rythme intense entre les services est à réaliser pour tester la robustesse de VIZOD dans ce contexte.

7. Une remarque de Richard Stallman à ce sujet http://www.gnu.org/philosophy/who-does-that-server-really-serve.html

Troisième partie

Conclusion et perspectives générales

Chapitre 8

Conclusion et perspectives générales

8.1 Bilan de nos travaux

Ces travaux nous ont permis d'identifier un cadre général de design de la visualisation d'information interactive.

A partir de ce cadre nous avons proposé VizOD, une architecture de création, de composition et de partage de services web de visualisation. VizOD nous a permis de développer de nouvelles applications visuelles, mais également de détourner l'usage d'applications existantes et d'y inclure de nouvelles visualisations qui n'étaient pas prévues à l'origine. VizOD nous a également permis d'identifier des alternatives de designs lors de la conception de l'interface, et d'analyser l'usage des applications lors de la phase d'évaluation à partir de métriques quantitatives (telles que les temps d'exécution des services et les capacités du réseau).

Nous avions cité [Pla04] en introduction de nos travaux *"La visualisation est aussi en train d'infiltrer des applications grand public de manière subtile"*. Nous pouvons paraphraser ce constat en fonction de notre approche et résumer ainsi la résumer *"Les services web sont en train d'infiltrer des applications grand public de manière subtile, ce qui permet à la visualisation de les infiltrer également"*.

8.2 Perspectives de recherche

Prise en compte de déficiences visuelles et profil visuel utilisateur

La prise en compte des déficiences visuelles est un défi majeur. Même si les humains partagent des caractéristiques communes, chacun possède un système biologique propre (qui peut par exemple être affecté par le daltonisme, un changement la perception des couleurs) et des connaissances propres (en fonction de son contexte culturel notamment). Les flots de visualisation sont un axe à étudier afin de systématiquement y intégrer des préférences de l'utilisateur : choix de contraste, taille de police ou d'attributs visuels, encodage visuel

consistant entre les applications (comme avec le service `GraphML2CSS_VIZOD` de l'application MosaiZ (section 6.4) et la surcharge d'encodage visuel dans POSvis (section 6.5)). Un axe de recherche est donc de généraliser ces abstractions visuelles (mais aussi de point de vue) et ainsi définir un profil visuel utilisateur inter-applicatif, qui permet à toute nouvelle application de systématiquement prendre en compte les préférences visuelles de l'utilisateur.

Caractérisation de profils applicatifs et prise en compte du contexte

Les profils applicatifs qui peuvent être *infiltrées* par la visualisation sont potentiellement nombreux. Au delà d'applications logicielles "classiques", les paradigmes technologiques évoluent vers l'intégration de phénomènes physiques dans l'interaction [DFA04]. Une description plus large du profil applicatif étendue au contexte environnant l'utilisateur est nécessaire, comme caractériser l'interaction multimodale en entrée [NC96] et en sortie. Les métriques d'évaluation de ces interactions seront à redéfinir, car certaines n'auront pas de rétroaction physique ou de dispositif de pointage au moyen desquels indiquer l'état du système

Visualisation de données continues

Nous avions indiqué en introduction que la quantité d'information produite par Twitter était de 8 Térabits par jour, soit 80Mo par seconde. Les défis techniques et conceptuels de visualisation de ces *donnés continues* sont importants. D'un point de vue technique il est nécessaire de garantir une qualité de service à l'utilisateur (qui souhaite par exemple avoir une synthèse en temps réel d'un flux), et donc de réaliser des opérations de traitement ou de mise à jour en temps réel de ces vues. D'un point de vue conceptuel, obtenir une vue globale de ces flux est difficile car elle évolue en permanence. Il est donc sûrement nécessaire de développer de nouvelles visualisations, ainsi que les flots permettant d'en supporter les traitements et les mises à jour. Il est également nécessaire de développer de nouveaux widgets, interfaces graphiques et types de requêtes permettant de manipuler ces flux car les données, la vue et donc les besoins de l'utilisateur peuvent désormais changer au fil même de la construction d'une requête ou de l'exploration des vues.

Interactive visual interfaces exploit human visual and cognitive skills, to assist the processing of large amounts of abstract data. The design and evaluation of these interfaces, however, remain a complex process because they cover the entire chain of information processing, from raw data, up to images. Current interfaces lack of flexibility and adaptability to new data, task or mining technique because they require new iteration of software development.

In this thesis we are interested in improving the software design process by enabling a service-oriented approach to facilitate any redeployment, sharing and collaboration around visualizations. Our approach has been first to extensive study of the information visualization design space, visual interactions and visual exploration techniques. We then proposed an architecture design consisting of an application profile, a scene graph and of visualization web services. The implementation of four interactive applications showed the operationalization of this architecture, in contexts as diverse -in terms of data and tasks- as movies database, pictures, social networks and collection of literary texts exploration.

Annexe A

Schéma détaillé de l'architecture VizOD

A.1 Schéma détaillé de l'architecture VizOD

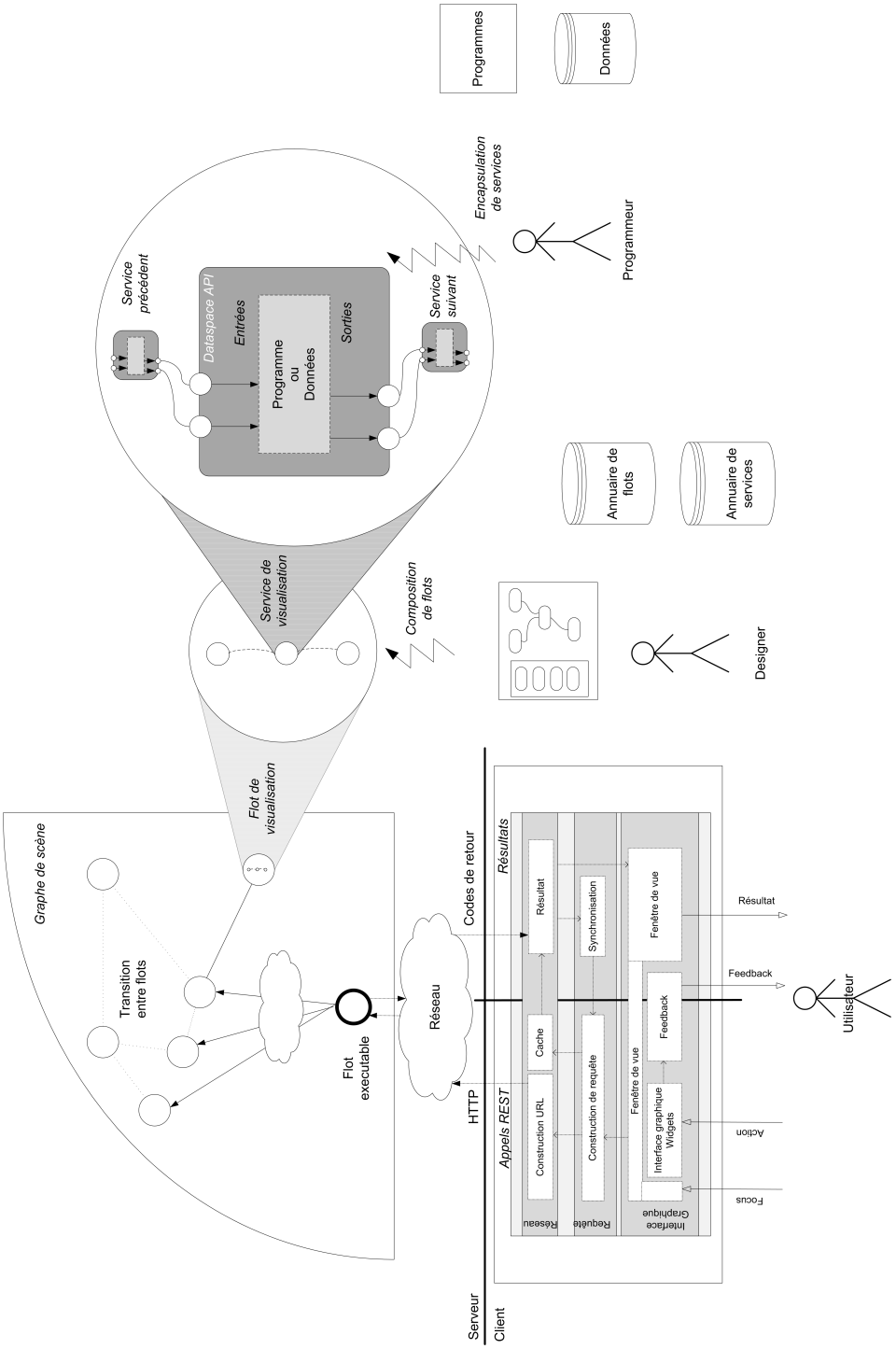


FIGURE A.1 – Vue globale de l'architecture VIZOD.

Annexe B

Services et flots de visualisation des applications

B.1 Liste des services développés

Application	Service	Description	Programme
VizOD+GE	DatabaseHost_ORACLE	*Connecteur à une base de données Oracle*	Oracle v10g XE [Ora10]
	Query_ORACLE	*Requête dans une base de données*	Oracle v10g XE [Ora10]
	GraphBuilder_VIZOD	*Construction de graphe*	Programme C
	GraphLayout_LGL	*Auto-organisation du graphe*	LGL v1.1 [ADWM04]
	GraphML2LGL_VIZOD	*Transformation du format GrapML en LGL*	LGL v1.1 [ADWM04]
	GraphRender_LGL	*Rendu de graphe LGL en image*	LGL v1.1 [ADWM04]
	LaplacianFilter_GIMP	*Filtrage laplacien d'une image*	GIMP [Pro10b]
	BlurFilter_GIMP	*Filtrage gaussien d'une image*	GIMP [Pro10b]
	KML_VIZOD	*Ecriture d'un fichier KML*	Script PHP
metaConf	DatabaseHost_MYSQL	*Connecteur à la base de données MySQL*	MySQL v5.0.37 [mys10]
	urlParseMetaconf_VIZOD	*Extraction de paramètres d'une URL*	Script PHP
	ParseHTTPHeader_VIZOD	*Extraction d'un en-tête HTTP*	Script PHP
	StringBuilder_VIZOD	*Construction d'une chaine avec paramètres*	Script PHP
	QueryString_MYSQL	*Requête dans une base MySQL*	Script PHP
	GraphBuilder_VIZOD	*Construction de graphe*	Programme C
	ColorMapping_VIZOD	*Association attribut et variable graphique*	Script PHP
	GraphML2Dot_VIZOD	*Conversion de formats de graphes*	Programme C
	GraphML2CSS_VIZOD	*Conversion en CSS de l'encodage visuel*	Programme C
	GraphRender_GRAPHVIZ	*Rendu de graphe avec GraphViz*	Graphvizv2.26.3 [Gra10]
MosaiZ	DatabaseHost_MYSQL	*Connecteur à la base de données MySQL*	MySQL v5.0.37 [mys10]
	StringBuilder_VIZOD	*Construction d'une chaine avec paramètres*	Script PHP
	QueryString_MYSQL	*Requête dans une base MySQL*	Script PHP
	FetchImagePNG_VIZOD	*Récupération d'images à partir d'une URL*	Script PHP
	ProcessImage_OPENCV	*Traitement d'image*	OpenCV v2.1 [Ope10a]
	DataCache_VIZOD	*Stockage temporaire d'images*	Programme C
	ForEach_VIZOD	*Opérateur logique*	Programme C
	Mosaic_METAPIXEL	*Création de mosaïques d'images*	metapixel v1.0.2 [Met10]
	MosaiZ_VIZOD	*Création de MosaiZ*	Java
POSvis	DatabaseHost_MYSQL	*Connecteur à la base de données MySQL*	MySQL v5.0.37 [mys10]
	StringBuilder_VIZOD	*Construction d'une chaine avec paramètres*	Script PHP
	QueryString_MYSQL	*Requête dans une base MySQL*	Script PHP
	ParamBuilder_VIZOD	*Construction de paramètre*	Script PHP
	TagCloudParams_VIZOD	*Paramètres d'un nuage de mots*	Script PHP
	URLBuilder_VIZOD	*Surcharge et communication de paramètres*	Script PHP

Table B.1 – Liste et description des services développés.

B.2 Flot de visualisation d'un graphe VizOD+GE

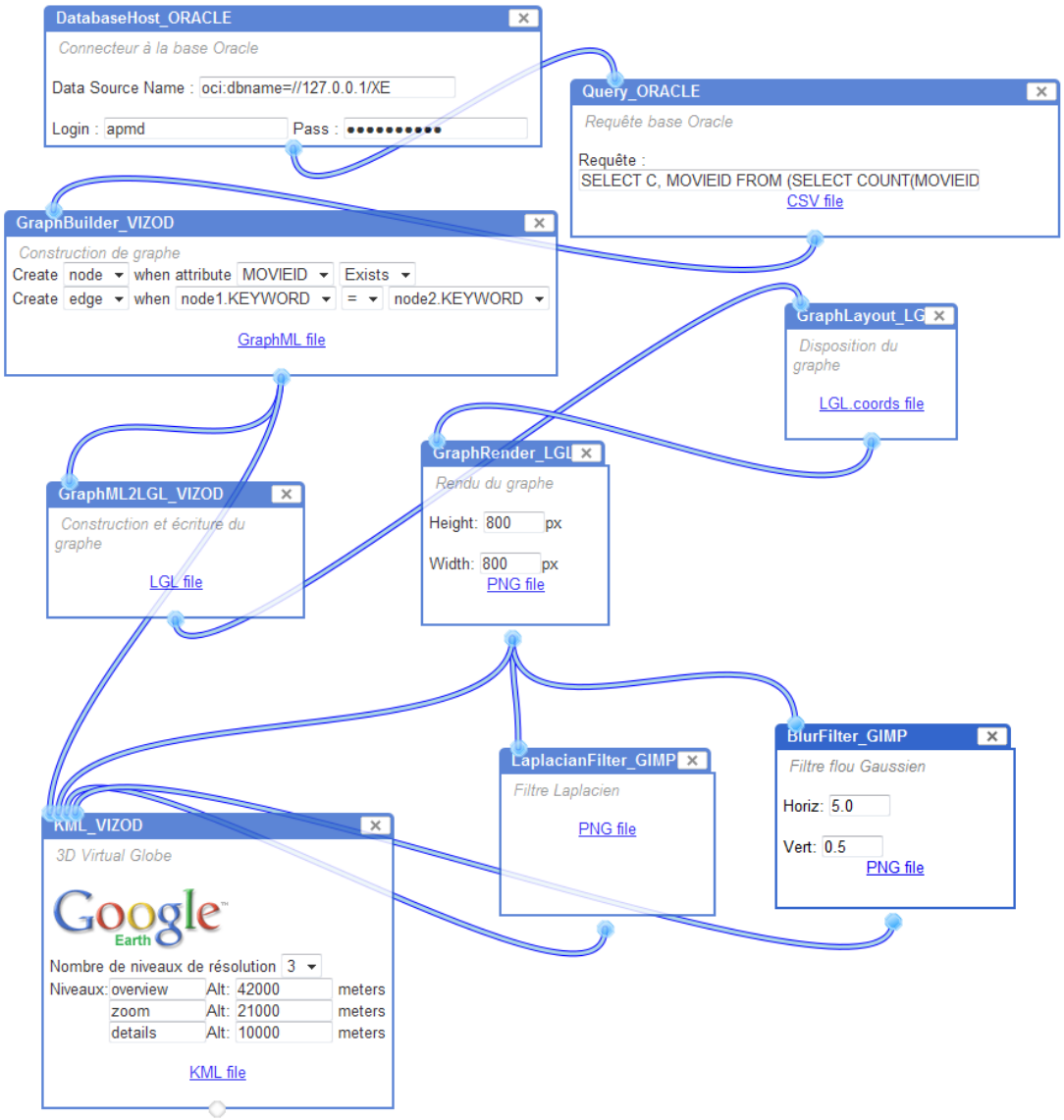


FIGURE B.1 – Flot de visualisation d'une graphe VIZOD+GE.

B.3 Flot de visualisation de metaConf

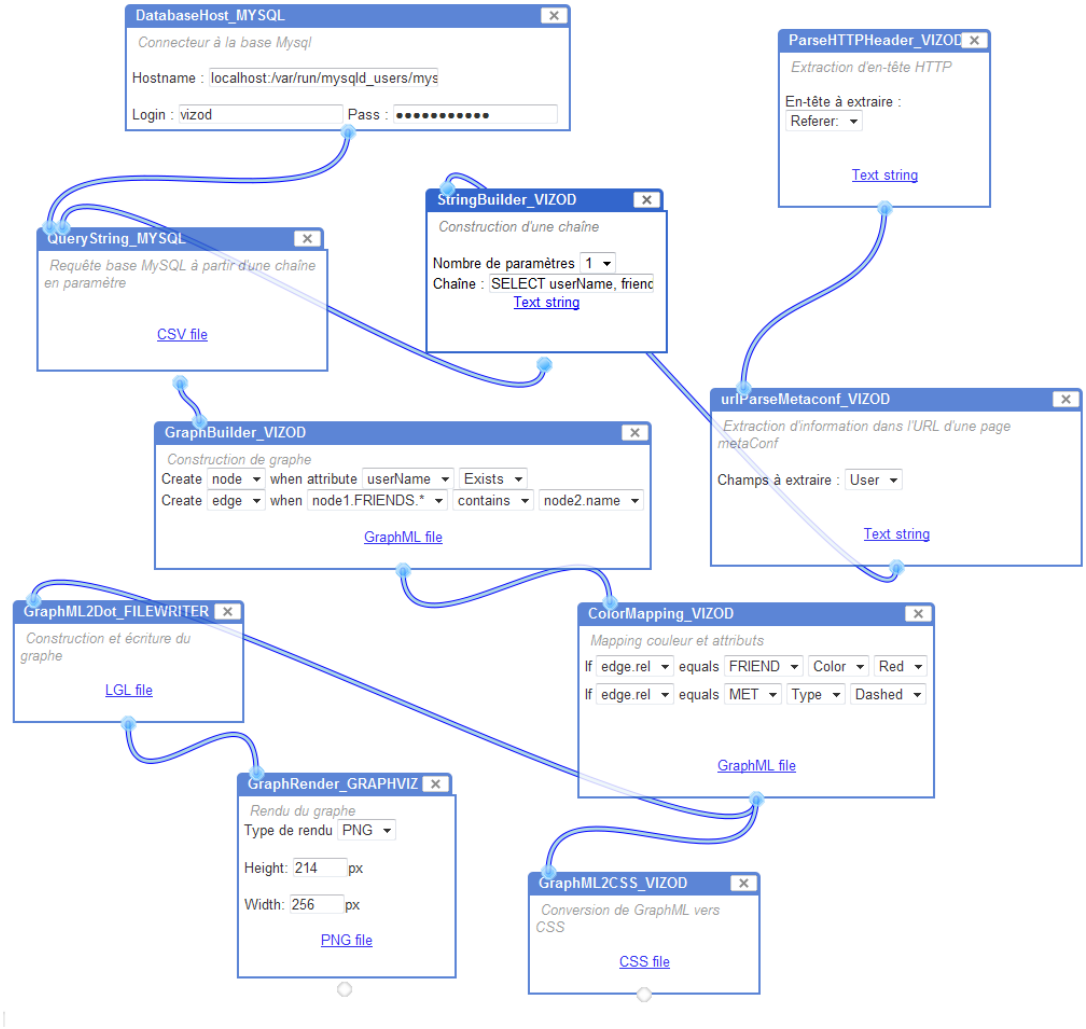


FIGURE B.2 – Flot de visualisation d'un graphe social dans METACONF.

B.4 Flot de visualisation de MosaiZ

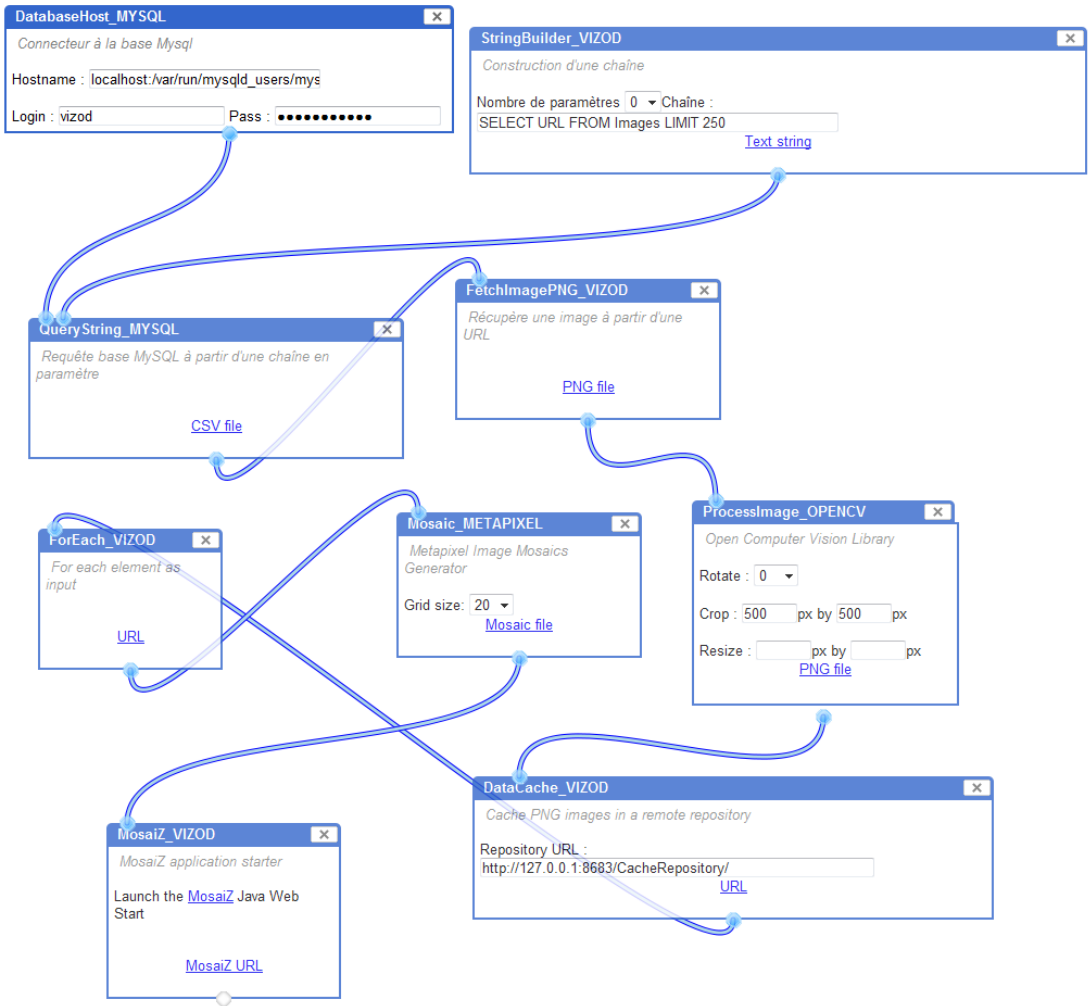


FIGURE B.3 – Flot de visualisation de MOSAIZ.

B.5 Flot de visualisation de POSvis

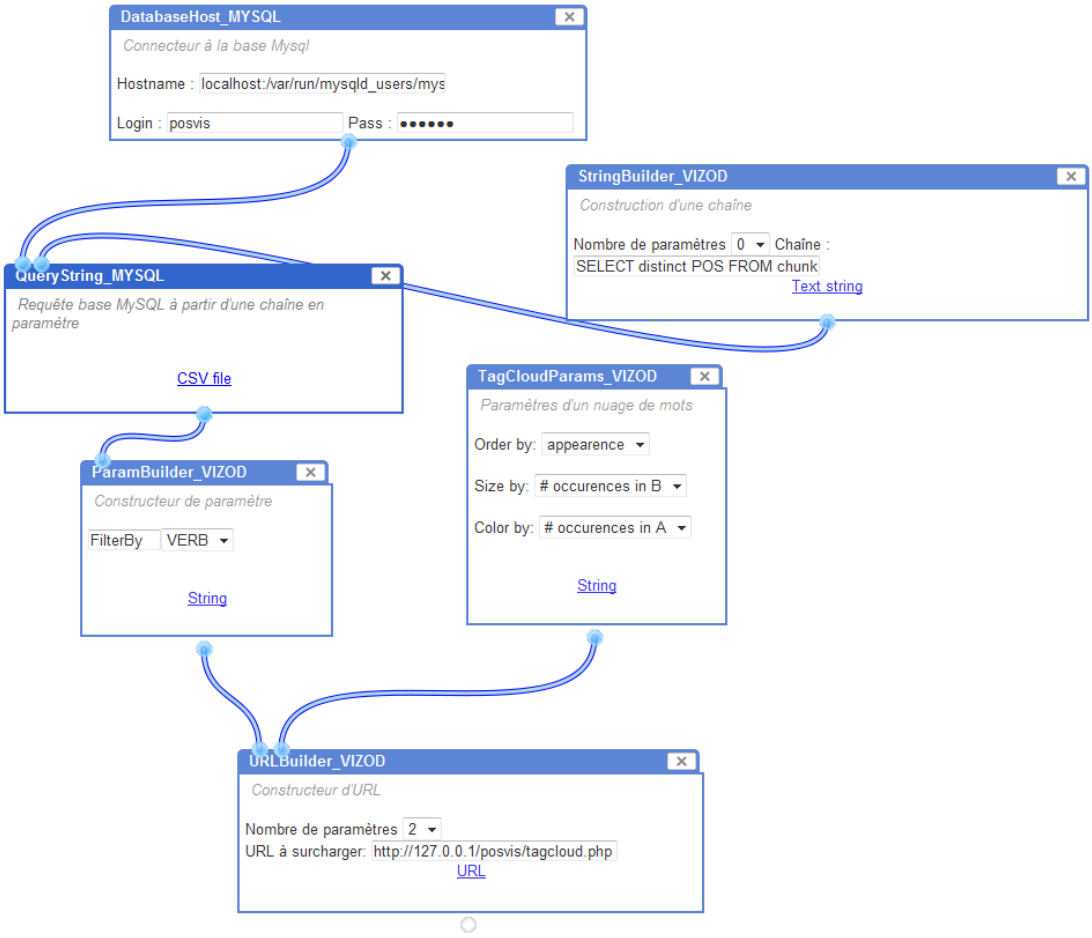


FIGURE B.4 – Flot d'adaptation de l'URL de nuage de mots de POSvis.

Bibliographie

[ADWM04] A. T. Adai, S. V. Date, S. Wieland, and E. M. Marcotte. Lgl : creating a map of protein function with an algorithm for visualizing very large biological networks. *J Mol Biol*, 340(1) :179–190, June 2004. 5.2.2, 6.2, 7.2.1, B.1

[AES05] R. Amar, J. Eagan, and J. Stasko. Low-level components of analytic activity in information visualization. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005*, pages 111–117, 2005. 4.2.1, 4.2.1

[AMD10] AMD Graphics. http ://ati.amd.com. 2010. 2.2.1

[AS94a] C. Ahlberg and B. Shneiderman. The alphaslider : a compact and rapid selector. In *Proceedings of the SIGCHI conference on Human factors in computing systems : celebrating interdependence*, pages 365–371. ACM, 1994. 3.2.1

[AS94b] Christopher Ahlberg and Ben Shneiderman. Visual information seeking using the filmfinder. In *CHI '94 : Conference companion on Human factors in computing systems*, pages 433–434, New York, NY, USA, 1994. ACM. (document), 2.4.1, 2.13, 3.2.2, 3

[Aub03] D. Auber. Tulip-a huge graph visualization framework. *Graph Drawing Software*, 2003. 4

[AWS92] Christopher Ahlberg, Christopher Williamson, and Ben Shneiderman. Dynamic queries for information exploration : An implementation and evaluation. In *Proc. ACM Conf. Human Factors in Computer Systems, CHI*, pages 619–626, 3–7 1992. (document), 3.2.1, 3.2.2, 3.6

[Bed01] Benjamin B. Bederson. Photomesa : a zoomable image browser using quantum treemaps and bubblemaps. In *UIST '01 : Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 71–80, New York, NY, USA, 2001. ACM. (document), 3.3.2, 3.12, 5.1.5

[Ber83] J. Bertin. *Semiology of graphics*. University of Wisconsin Press, 1983. (document), 2.1.1, 2.4.2, 2.18

[BF93] N. Borenstein and N. Freed. MIME (Multipurpose Internet Mail Extensions) Part One : Mechanisms for Specifying and Describing the Format... In *RFC 1521, BELLCORE, INNOSOFT*. Citeseer, 1993. 5.1.3

[BGM04] Benjamin B. Bederson, Jesse Grosjean, and Jon Meyer. Toolkit design for interactive structured graphics. *IEEE Trans. Softw. Eng.*, 30(8) :535–546, 2004. (document), 1.1, 2.2.1, 2.2, 2.2.1, 3.3.2, 6.4, 6.6

[BH09] Michael Bostock and Jeffrey Heer. Protovis : A graphical toolkit for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15 :1121–1128, 2009. 2.2.1, 7.3.1

[BL09] Enrico Bertini and Denis Lalanne. Surveying the complementary role of automatic data analysis and visualization in knowledge discovery. In *VAKD '09 : Proceedings of the ACM SIGKDD Workshop on Visual Analytics and Knowledge Discovery*, pages 12–20, New York, NY, USA, 2009. ACM. 4.3.1

[BPW+93] Ken Brodlie, Andrew Poon, Helen Wright, Lesley Brankin, Greg Banecki, and Alan Gay. Grasparc : a problem solving environment integrating computation and visualization. In *VIS '93 : Proceedings of the 4th conference on Visualization '93*, pages 102–109, Washington, DC, USA, 1993. IEEE Computer Society. 6

[BR10] P. Biswas and P. Robinson. A brief survey on user modelling in hci. In *International Conference on Intelligent Human Computer Interaction (IHCI)*, 2010. (document), 4.3.3, 4.9

[Bry10] Bryce 3D version 7. http ://www.daz3d.com/i/software/bryce7/. 2010. (document), 3.2.3, 3.3.3, 3.13

[Che73] H. Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, 68(342) :361–368, 1973. (document), 2.22, 2.4.2

[Che04] Hong Chen. Towards design patterns for dynamic analytical data visualization. In *Proceedings Of SPIE Visualization and Data Analysis*, pages 75–86, 2004. 4.2.2

[Che05] Chaomei Chen. Top 10 unsolved information visualization problems. *IEEE Comput. Graph. Appl.*, 25(4) :12–16, 2005. 2.1.1, 7.3.2

[Chi00] Ed H. Chi. A taxonomy of visualization techniques using the data state reference model. In *INFOVIS*, pages 69–76, 2000. 2.3.1, 1, 2, 3, 2.3.1, 2.3.2, 1, 2, 3, 4.2.1

[CKB08] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Comput. Surv.*, 41(1) :1–31, 2008. 5

[CM97] S.K. Card and J. Mackinlay. The structure of the information visualization design space. In *Proceedings of the 1997 IEEE Symposium on Information Visualization (InfoVis' 97)*, page 92. IEEE Computer Society Washington, DC, USA, 1997. (document), 2.4.1, 2.13, 2.4.2

[CMS99] S.K. Card, J.D. Mackinlay, and B. Shneiderman. *Readings in information visualization : using vision to think*. Morgan Kaufmann, 1999. 2.1.1

[Cou87] J. Coutaz. PAC : an object oriented model for implementing user interfaces. *ACM SIGCHI Bulletin*, 19(2) :41, 1987. 5.1.2

[CR96] M. C. Chuah and S. F. Roth. On the semantics of interactive visualizations. In *INFOVIS '96 : Proceedings of the 1996 IEEE Symposium on Information Visualization (INFOVIS '96)*, page 29, Washington, DC, USA, 1996. IEEE Computer Society. 4.2.1

[CR98] Ed Huai-hsin Chi and John Riedl. An operator interaction framework for visualization systems. In *INFOVIS '98 : Proceedings of the 1998 IEEE Symposium on Information Visualization*, pages 63–70, Washington, DC, USA, 1998. IEEE Computer Society. 2.3.2, 2.3.2, 2.4

[DC94] P. Dourish and M. Chalmers. Running out of space : models of information navigation. In *Short paper presented at HCI*, volume 94. Citeseer, 1994. 4.3.3

[DFA04] A. Dix, J. Finlay, and G.D. Abowd. *Human-computer interaction*. Prentice hall, 2004. 3.3.3, 4.2.1, 8.2

[DSB04] S. Dos Santos and K. Brodlie. Gaining understanding of multivariate and multidimensional data through visualization. *Computers & Graphics*, 28(3) :311–325, 2004. (document), 2.10, 2.3.1, 1, 2, 3, 2.4

[Ead84] P. Eades. A heuristic for graph drawing. *Congressus numerantium*, 42(149160) :194–202, 1984. 2.4.1, 6.2

[EDF08] N. Elmqvist, P. Dragicevic, and J.D. Fekete. Rolling the dice : Multidimensional visual exploration using scatterplot matrix navigation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6) :1539–1148, 2008. 2.2.1

[EHS+02] Jennifer English, Marti Hearst, Rashmi Sinha, Kirsten Swearingen, and Ka-Ping Yee. Flexible search and navigation using faceted metadata. Technical report, University of Berkeley, 2002. 3.3.1

[Fac10] Facebook API. http ://developers.facebook.com/docs/api. 2010. 1.1

[FB95] George W. Furnas and Benjamin B. Bederson. Space-scale diagrams : understanding multiscale interfaces. In *CHI '95 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 234–241, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. 3.3.2

[FD05] M. Friendly and D.J. Denis. Milestones in the history of thematic cartography, statistical graphics, and data visualization. *Retrieved August*, 7 :2005, 2005. (document), 2.1.1, 2.2.3, 2.22, 2.4.2

[FdOL03] MC Ferreira de Oliveira and H. Levkowitz. From visual data exploration to visual data mining : a survey. *IEEE Transactions on Visualization and Computer Graphics*, 9(3) :378–394, 2003. 2.2.2

[Fek04] J.D. Fekete. The infovis toolkit. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 167–174. IEEE Computer Society Washington, DC, USA, 2004. 1.1

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1. RFC 2616, Internet Engineering Task Force, June 1999. 5.2.5, 7.2.2

[Fie00] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, 2000. Chair-Taylor,, Richard N. 5.2.1

[Fla10a] Flamenco. http ://flamenco.berkeley.edu/. 2010. (document), 3.3.1, 3.10

[Fla10b] Flare. http ://flare.prefuse.org/. 2010. 1.1

[Fli10] Flickr API. http ://www.flickr.com/services/api/. 2010. 1.1

[FP99] J.D. Fekete and C. Plaisant. Excentric labeling : Dynamic neighborhood labeling for data visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems : the CHI is the limit*, page 519. ACM, 1999. 3.2.1, 3.3.1

[FP02a] J. Fekete and C. Plaisant. Interactive information visualization of a million items proceedings of ieee symposium on information visualization, 2002. (document)

[FP02b] Jean-Daniel Fekete and Catherine Plaisant. Interactive information visualization of a million items. In *INFOVIS '02 : Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, page 117, Washington, DC, USA, 2002. IEEE Computer Society. 2.2.1

[Fur86] G. W. Furnas. Generalized fisheye views. *SIGCHI Bull.*, 17(4) :16–23, 1986. 3.3.1

[Gap10] GapMinder. http ://www.gapminder.org/. 2010. (document), 4.2.2, 4.2

[GC07] S. Gao and D.F. Chen. Applying web service technology in Distributed Visualization. In *2007 International Conference on Machine Learning and Cybernetics*, volume 7, 2007. 5.2.2

[Gib86] J.J. Gibson. *The ecological approach to visual perception*. Lawrence Erlbaum, 1986. 3.1

[Goo10a] Google. http ://www.google.com/. 2010. (document), 1.1, 3.5, 5.1.5, 7.7

[Goo10b] Google Earth. http ://earth.google.com/. 2010. (document), 2.7, 6.2

[Goo10c] Google Insights for Search. http ://www.google.com/insights/search/. 2010. (document), 4.2.2, 4.3

[Goo10d] Google Maps. http ://maps.google.com/. 2010. (document), 1.2, 1.4, 9, 2.4.3, 3.3.1, 3.9, 6.2.4, 7.2.2, 7.2.2, 7.7

[Gra10] Graphviz - Graph Visualization Software. http ://www.graphviz.org/. 2010. 6.3.1, 7.2.1, B.1

[GS00] William I. Grosky and Peter L. Stanchev. An image data model. In *VISUAL '00 : Proceedings of the 4th International Conference on Advances in Visual Information Systems*, pages 14–25, London, UK, 2000. Springer-Verlag. (document), 2.4.1, 2.17

[GYG] A.G. Gee, M. Yu, and GG Grinstein. Dynamic and interactive dimensional anchors for spring-based visualizations. Technical report, Technical report, computer science, University of Massachussetts Lowell, 2005. 2.1.1

[HA06] J. Heer and M. Agrawala. Software design patterns for information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5) :853, 2006. 2.3.1

[HA08] J. Heer and M. Agrawala. Design considerations for collaborative visual analytics. *Information Visualization*, 7(1) :49–62, 2008. 4.3.3

[HCL05] Jeffrey Heer, Stuart K. Card, and James A. Landay. prefuse : a toolkit for interactive information visualization. In *CHI '05 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 421–430, New York, NY, USA, 2005. ACM. 1.1, 1.2, 2.3.1, 1, 2, 3, 1, 2, 3

[HFH+09] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The WEKA data mining software : An update. *ACM SIGKDD Explorations Newsletter*, 11(1) :10–18, 2009. 4.3.1

[HHC+08] Jeffrey Heer, Frank Ham, Sheelagh Carpendale, Chris Weaver, and Petra Isenberg. Creation and collaboration : Engaging new audiences for information visualization. pages 92–133, 2008. 4.3.3, 5.2.1, 5.2.4, 5.4, 7.2.1

[HHWL01] J.I. Hong, J. Heer, S. Waterson, and J.A. Landay. WebQuilt : A proxy-based approach to remote web usability testing. *ACM Transactions on Information Systems (TOIS)*, 19(3) :263–285, 2001. 5.3

[Hib04] Bill Hibbard. The top five problems that motivated my work. *IEEE Comput. Graph. Appl.*, 24(6) :9–13, 2004. 3.2.2, 7.3.2

[HM90] R.B. Haber and D.A. McNabb. Visualization idioms : A conceptual model for scientific visualization systems. *Visualization in scientific computing*, pages 74–93, 1990. 2.3.1, 1, 2, 3, 1, 2, 3, 2.4

[HMM00] Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph visualization and navigation in information visualization : A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1) :24–43, 2000. 2.4.1

[HR07] Jeffrey Heer and George Robertson. Animated transitions in statistical data graphics. *IEEE Transactions on Visualization and Computer Graphics*, 13(6) :1240–1247, 2007. 2.2.1

[htt08] http ://www.imdb.com/. The internet movie database. page http ://www.imdb.com/, 2008. 6.2

[htt09] http ://www.flickr.com/. Flickr. 2009. 9

[htt10a] http ://code.google.com/intl/en/webtoolkit/. Google Web Toolkit (GWT). 2010. 3.2.1

[htt10b] http ://developer.yahoo.com/yui/. The Yahoo ! User Interface Library (YUI). 2010. 3.2.1

[HVW07] Jeffrey Heer, Fernanda B. Viégas, and Martin Wattenberg. Voyagers and voyeurs : supporting asynchronous collaborative information visualization. In *CHI '07 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1029–1038, New York, NY, USA, 2007. ACM. (document), 4.3.3, 4.10

[ID90] A. Inselberg and B. Dimsdale. Parallel coordinates : a tool for visualizing multidimensional geometry. In *Proceedings of the 1st conference on Visualization'90*, pages 361–378. IEEE Computer Society Press, 1990. 2.4.1

[Jac] C. Jacquemin. Virtual Choreographer Reference Guide (version 1.4). *LIMSI-CNRS and Université Paris*, 11. (document), 2.2.1, 2.2

[JHM04] Wesley M. Johnston, J. R. Paul Hanna, and Richard J. Millar. Advances in dataflow programming languages. *ACM Comput. Surv.*, 36(1) :1–34, 2004. 5.2.3, 8

[JS02] B. Johnson and B. Shneiderman. Tree-maps : A space-filling approach to the visualization of hierarchical information structures. In *Visualization, 1991. Visualization'91, Proceedings., IEEE Conference on*, pages 284–291. IEEE, 2002. (document), 2.4.1, 2.15

[JS03] J.A. Jacko and A. Sears. *The human-computer interaction handbook : fundamentals, evolving technologies, and emerging applications*. Lawrence Erlbaum Assoc Inc, 2003. 2.1.1

[Kar09] Kartoo. http ://www.kartoo.com/. 2009. 2.2.3

[Kay10] Kayak. http ://www.kayak.fr/. 2010. 3.2.2

[Kei02a] D.A. Keim. Designing pixel-oriented visualization techniques : Theory and applications. *Visualization and Computer Graphics, IEEE Transactions on*, 6(1) :59–78, 2002. 2.4.1

[Kei02b] Daniel A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1) :1–8, 2002. (document), 2.4.1, 2.14, 3.3.1

[KMH02] R. Kosara, S. Miksch, and H. Hauser. Focus+ context taken literally. *IEEE Computer Graphics and Applications*, 22(1) :22–29, 2002. 3.3.1

[KMSZ06] Daniel A. Keim, Florian Mansmann, Jorn Schneidewind, and Hartmut Ziegler. Challenges in visual data analysis. In *IV '06 : Proceedings of the conference on Information Visualization*, pages 9–16, Washington, DC, USA, 2006. IEEE Computer Society. (document), 4.3.1, 4.3.2, 4.6, 4.3.2

[Kni10] Knime. http ://www.knime.org/. 2010. (document), 4.3.1, 4.5

[KO07] DA Keim and D. Oelke. Literature fingerprinting : A new method for visual literary analysis. In *IEEE Symposium on Visual Analytics Science and Technology, 2007. VAST 2007*, pages 115–122, 2007. 2.4.1

[Koh82] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1) :59–69, 1982. 2.2.3

[KP88] G.E. Krasner and S.T. Pope. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of Object Oriented Programming*, 1(3) :26–49, 1988. 2.3.1, 5.1.2

[LA94] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Trans. Comput.-Hum. Interact.*, 1(2) :126–160, 1994. 3.3.1

[LAC+08] X.Á. Llorà, B. Auvil, L.S. Capitanu, B. Welge, M.E. Goldberg, and E. David. Meandre : Semantic-Driven Data-Intensive Flows in the Clouds. In *IEEE Fourth International Conference on eScience, 2008. eScience'08*, pages 238–245, 2008. 6.5.1

[Las87] J. Lasseter. Principles of traditional animation applied to 3D computer animation. *ACM SIGGRAPH Computer Graphics*, 21(4) :35–44, 1987. (document), 2.2.1, 2.3

[Lay10] Layar. http ://www.layar.com/. 2010. 7.2.1

[LPP+06] Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. Task taxonomy for graph visualization. In *BELIV '06 : Proceedings of the 2006 AVI workshop on BEyond time and errors*, pages 1–5, New York, NY, USA, 2006. ACM. 4.2.1, 4.2.2

[Mac86] Jock Mackinlay. Automating the design of graphical presentations of relational information. *ACM Trans. Graph.*, 5(2) :110–141, 1986. (document), 2.4.1, 2.4.2, 2.4.2, 2.19, 2.4.3

[Man09] ManyEyes. http ://manyeyes.alphaworks.ibm.com/manyeyes/. 2009. (document), 1.1, 1.2, 7.2.2, 7.5

[MAT10] MATHLAB. http ://www.mathworks.com/. 2010. 4.3.1

[MB07] Francis T. Marchese and Natasha Brajkovska. Fostering asynchronous collaborative visualization. In *IV '07 : Proceedings of the 11th International Conference Information Visualization*, pages 185–190, Washington, DC, USA, 2007. IEEE Computer Society. 4.3.3

[McC88] B. H. McCormick. Visualization in scientific computing. *SIGBIO Newsl.*, 10(1) :15–21, 1988. 2.1.1

[ME09] Bryan McDonnel and Niklas Elmqvist. Towards utilizing gpus in information visualization : A model and implementation of image-space operations. *IEEE Transactions on Visualization and Computer Graphics*, 15 :1105–1112, 2009. 2.2.1, 2.3.1, 2.4

[Met10] Metapixel - A Photomosaic Generator. http ://www.complang.tuwien.ac.at/schani/metapixel/. 2010. 6.4.1, B.1

[Mic10a] Microsoft Excel. http ://office.microsoft.com/excel/. 2010. 1.2, 2.4.1, 5.1.4, 7.2.1

[Mic10b] Microsoft Photosynth. http ://www.vtk.org/. 2010. (document), 1.2, 1.4, 7.2.2

[Moo10] MooTools - a compact javascript framework. http ://mootools.net/. 2010. 6.6

[MRC91] Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The perspective wall : detail and context smoothly integrated. In *CHI '91 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 173–176, New York, NY, USA, 1991. ACM. (document), 3.3.1, 3.11

[mys10] mysql. http ://www.mysql.com/. 2010. B.1

[NB02] D.A. Norman and Inc Books24x7. *The design of everyday things*. Basic Books New York, 2002. 3.2.2

[NC96] L. Nigay and J. Coutaz. Espaces conceptuels pour l'interaction multimédia et multimodale. *Technique et science informatiques*, 15(9) :1195–1225, 1996. 8.2

[Nei93] J. Neilsen. Usability engineering. *Boston : AP Professional*, 1993. 2.4.2, 3.2.2, 5.4

[net10] netflix. http ://www.netflix.com/. 2010. 6.2

[Nor05] C. North. Information visualization. *Handbook of Human Factors and Ergonomics*,, pages 1222–1246, 2005. 2.4.3

[NS00] Chris North and Ben Shneiderman. Snap-together visualization : a user interface for coordinating visualizations via relational schemata. In *AVI '00 : Proceedings of the working conference on Advanced visual interfaces*, pages 128–135, New York, NY, USA, 2000. ACM. 3.3.1

[NVD10] NVDIA. http ://www.nvidia.com/. 2010. 2.2.1

[OJN+00] Dan R. Olsen, Jr., Sean Jefferies, Travis Nielsen, William Moyes, and Paul Fredrickson. Cross-modal interaction using xweb. In *UIST '00 : Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 191–200, New York, NY, USA, 2000. ACM. 3

[Ols07] Dan R. Olsen, Jr. Evaluating user interface systems research. In *UIST '07 : Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 251–258, New York, NY, USA, 2007. ACM. 5.2.1, 7.3.1

[Ope10a] OpenCV - Open Source Computer Vision. http ://opencv.willowgarage.com/wiki/. 2010. 6.4.1, 6.4.3, B.1

[Ope10b] OpenGL. http ://www.opengl.org/. 2010. 2.2.1

[Ope10c] OpenViz. http ://www.openviz.com/enterprise.html. 2010. 1.1

[Ora10] Oracle Database 10g Express Edition. http ://www.oracle.com/technetwork/database/express-edition/overview/index.html. 2010. B.1

[PC05] P. Pirolli and S. Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of International Conference on Intelligence Analysis*, volume 2005, pages 2–4, 2005. (document), 4.3.2, 4.7

[Pet07] Gabriele Peters. Aesthetic primitives of images for visualization. In *IV '07 : Proceedings of the 11th International Conference Information Visualization*, pages 316–325, Washington, DC, USA, 2007. IEEE Computer Society. 2.4.2

[Pip10] Yahoo ! Pipes. http ://pipes.yahoo.com/pipes/. 2010. 5.2.3

[Pla04] Catherine Plaisant. The challenge of information visualization evaluation. In *AVI '04 : Proceedings of the working conference on Advanced visual interfaces*, pages 109–116, New York, NY, USA, 2004. ACM. 1.2, 8.1

[Pro10a] Processing. http ://processing.org/. 2010. 1.1

[Pro10b] GNU Image Manipulation Program. http ://www.gimp.org/. 2010. 5.1.3, 5.1.4, 4, 6.2.1, B.1

[PSDB99] Catherine Plaisant, Ben Shneiderman, Khoa Doan, and Tom Bruns. Interface and data architecture for query preview in networked information systems. *ACM Trans. Inf. Syst.*, 17(3) :320–341, 1999. 3.2.2

[Pub10] Tableau Public. http ://www.tableausoftware.com/public/. 2010. 1.1

[R10] R. http ://www.r-project.org/. 2010. 4.3.1

[RCL+98] George Robertson, Mary Czerwinski, Kevin Larson, Daniel C. Robbins, David Thiel, and Maarten van Dantzich. Data mountain : using spatial memory for document management. In *UIST '98 : Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 153–162, New York, NY, USA, 1998. ACM. (document), 2.4.1, 2.16

[Rea10] RealVNC - VNC remote control software. http ://www.realvnc.com/. 2010. 1.1

[RLP10] N.H. Riche, B. Lee, and C. Plaisant. Understanding Interactive Legends : a Comparative Evaluation with Standard Widgets. In *Computer Graphics Forum*, volume 29, pages 1193–1202. Wiley-Blackwell, 2010. 3.2.1

[Rob07] Jonathan C. Roberts. State of the art : Coordinated & multiple views in exploratory visualization. In *CMV '07 : Proceedings of the Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization*, pages 61–71, Washington, DC, USA, 2007. IEEE Computer Society. 3.3.1

[S+02] J.C. Schlimmer et al. Web services description requirements. *World Wide Web Consortium (W3C) working draft. Oct*, 28, 2002. 5.2.1

[SA88] R.R. Swick and M.S. Ackerman. The X Toolkit : More Bricks for Building User-Interfaces- or- Widgets For Hire. In *Proceedings of the Usenix Winter 1988 Conference*, pages 221–228. Citeseer, 1988. 3.2.1

[SAP10] SAP Business Objects BI. http ://www.SAP.com/BusinessObjects.com/SAP. 2010. 1.1

[Scu09] Marian Scuturici. Dataspace API. Technical report, LIRIS, September 2009. 5.2.2, 5.2.3, 5.2.5, 1, 6.5.2, 7.2.1, 7.3.2

[SGL08] J. Stasko, C. G
"org, and Z. Liu. Jigsaw : Supporting investigative analysis through interactive visualization. *Information Visualization*, 7(2) :118–132, 2008. (document), 1.1, 4.3.2, 4.8

[Shn87] B. Shneiderman. Direct manipulation : A step beyond programming languages. pages 461–467, 1987. 1, 3.2.2

[Shn96] Ben Shneiderman. The eyes have it : A task by data type taxonomy for information visualizations. In *VL '96 : Proceedings of the 1996 IEEE Symposium on Visual Languages*, page 336, Washington, DC, USA, 1996. IEEE Computer Society. 2.4.1, 2.4.2, 2.4.3, 4.2.1, 4.2.1, 4.2.2, 4.3.2, 6.2.3

[SHR+08] Johannes Schöning, Brent Hecht, Martin Raubal, Antonio Krüger, Meredith Marsh, and Michael Rohs. Improving interaction with virtual globes through spatial thinking : helping users ask "why ?". In *IUI '08 : Proceedings of the 13th international conference on Intelligent user interfaces*, pages 129–138, New York, NY, USA, 2008. ACM. 6.2

[Sof10] Tableau Software. http ://www.tableausoftware.com/. 2010. (document), 1.1, 2.4.3, 2.23

[SP06] Ben Shneiderman and Catherine Plaisant. Strategies for evaluating information visualization tools : multi-dimensional in-depth long-term case studies. In *BELIV '06 : Proceedings of the 2006 AVI workshop on BEyond time and errors*, pages 1–7, New York, NY, USA, 2006. ACM. 5.3

[SPCJ09] Ben Shneiderman, Catherine Plaisant, Maxine Cohen, and Steven Jacobs. *Designing the User Interface : Strategies for Effective Human-Computer Interaction*. Addison-Wesley Publishing Company, USA, 2009. 1.2, 3.2.2, 7.2.1

[Spo10] Spotfire. http ://spotfire.tibco.com/. 2010. 1.1

[SRS97] K. Sowizral, K. Rushforth, and H. Sowizral. *The Java 3D API Specification*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1997. 2.2.1

[STH02] C. Stolte, D. Tang, and P. Hanrahan. Polaris : A System for Query, Analysis, and Visualization of Multidimensional Relational Databases. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, 8(1) :1, 2002. 2.4, 4.2.2

[STH03] Chris Stolte, Diane Tang, and Pat Hanrahan. Multiscale visualization using data cubes. *IEEE Transactions on Visualization and Computer Graphics*, 9(2) :176–187, 2003. 2.3.2, 3.3.2

[SVR08] Nutchanat Sattayakawee, Romain Vuillemot, and Béatrice Rumpler. VizOD API, v2.3. Technical report, LIRIS, September 2008. 6.2.1

[SvW08] Yedendra Babu Shrinivasan and Jarke J. van Wijk. Supporting the analytical reasoning process in information visualization. In *CHI '08 : Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1237–1246, New York, NY, USA, 2008. ACM. 4.3.2

[Swi09] Swivel. http ://www.swivel.com/. 2009. 1.1

[TC05] James J. Thomas and Kristin A. Cook. *Illuminating the Path : The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Ctr, 2005. 7

[TG02] M. Takatsuka and M. Gahegan. GeoVISTA Studio : a codeless visual programming environment for geoscientific data analysis and visualization* 1. *Computers & Geosciences*, 28(10) :1131–1144, 2002. (document), 1.1, 4.1, 4.1

[TIB10] TIBCO Spotfire Analytics Server. http ://spotfire.tibco.com/products/analytics-server.aspx. 2010. 1.1

[TM04a] Melanie Tory and Torsten Möller. Human factors in visualization research. *IEEE Transactions on Visualization and Computer Graphics*, 10(1) :72–84, 2004. 4.2.2

[TM04b] Melanie Tory and Torsten Moller. Rethinking visualization : A high-level taxonomy. In *INFOVIS '04 : Proceedings of the IEEE Symposium on Information Visualization*, pages 151–158, Washington, DC, USA, 2004. IEEE Computer Society. 2.1.1

[Tuf01] Edward R. Tufte. *The Visual Display of Quantitative Information, 2nd edition*. Graphics Press, May 2001. 2.4.1, 2.4.3

[Tuk77] J.W. Tukey. Exploratory data analysis. 1977. 4.3.1

[Twi10] Twitter API. http ://dev.twitter.com/. 2010. 1.1, 1.1

[VCPK09] Romain Vuillemot, Tanya Clement, Catherine Plaisant, and Amit Kumar. What's Being Said Near "Martha" ? Exploring Name Entities in Literary Text Collections. In *IEEE Symposium on Visual Analytics Science and Technology (VAST)*, october 2009. 6.1, 6.5, 6.5.2, 7.2.3

[VCPV09] Clement Plaisant Vuillemot, Tanya Clement, Catherine Plaisant, and Romain Vuillemot. The story of one : Humanity scholarship with visualization and text analysis. *Digital Humanities 2009*, 2009. 6.1, 6.5

[vD97] Andries van Dam. Post-wimp user interfaces. *Communications of the ACM*, 40(2) :63–67, 1997. 3.2.3, 3.3.3

[Vis10] IBM ILOG Visualization. http ://www-01.ibm.com/software/websphere/products/visualization/. 2010. 1.1

[VP07] Romain Vuillemot and Veronika Peralta. From beautiful to useful : A multiscale visualization of users movie rating. *Research Report submitted to IEEE Infovis 2007 Contest*, July 2007. 6.1, 6.2, 2, 6.2.4

[VR08] Romain Vuillemot and Béatrice Rumpler. Mapping visualization on-demand onto a virtual globe : an appealing complement to browser-based navigation. In *HT '08*, pages 249–250, New York, NY, USA, 2008. ACM. 6.1, 6.2, 6.2.3

[VR09a] R. Vuillemot and B. Rumpler. MosaiZ : Interactive Image Mosaics. Technical report, LIRIS, 2009. 6.1, 6.4, 6.4.2

[VR09b] Romain Vuillemot and Béatrice Rumpler. A web-based interface to design information visualization. *MEDES 2009*, October 2009. 5.2.3, 7.2.3

[VR09c] Romain Vuillemot and Béatrice Rumpler. Une interface de programmation visuelle pour la composition de services de visualisation d'information. In *IHM'09 : Actes de la 21ème Conférence Francophone sur l'Interaction Homme-Machine*, 2009. 5.2.3

[VRP09] Romain Vuillemot, Béatrice Rumpler, and Jean-Marie Pinon. *Enterprise Information Systems*, chapter Dissection of a Visualization On-Demand Server, pages 348–360. LNBIP. Springer Berlin Heidelberg, April 2009. 5.2.3

[VTK10] VTK. http ://www.vtk.org/. 2010. 1.1

[Vui06] Romain Vuillemot. Modèle de représentation visuelle personnalisée de grands corpus de documents multimédia. *Rapport de Master Recherche. EDIIS INSA/Lyon1, Lyon, France*, June 2006. 1, 6.2.4

[Vui08] Romain Vuillemot. Visualisation personnalisée d'un corpus de conférences scientifiques. In *Atelier EGC'08 sur la modélisation utilisateur et la personnalisation d'interfaces Web (MUPIW'08)*, January 2008. 6.1, 6.3

[Vui10] Romain Vuillemot. Capture et modélisation de lactivité utilisateur pour lévaluation dapplications danalyse visuelle de données. In *8eme Atelier Visualisation et extraction de connaissances (AVEC), EGC 2010*, January 2010. 6.2.3

[Wal04] J. Walton. NAG's IRIS explorer. *Visualization handbook*, pages 633–654, 2004. 2.2.2, 5.2.2

[WBS+08] J. Wood, K. Brodlie, J. Seo, D. Duke, and J. Walton. A web services architecture for visualization. In *Proceedings of the IEEE Fourth International Conference on eScience, 2008.*, pages 1–7. IEEE Computer Society Press, 2008. 3, 5.2.2

[WBW96] Jason Wood, Ken Brodlie, and Helen Wright. Visualization over the world wide web and its application to environmental data. In *VIS '96 : Proceedings of the 7th conference on Visualization '96*, pages 81–ff., Los Alamitos, CA, USA, 1996. IEEE Computer Society Press. 2.3.1, 4.3.3

[WBWK00] Michelle Q. Wang Baldonado, Allison Woodruff, and Allan Kuchinsky. Guidelines for using multiple views in information visualization. In *AVI '00 : Proceedings of the working conference on Advanced visual interfaces*, pages 110–119, New York, NY, USA, 2000. ACM. 3.3.1, 3.3.1

[WDSC07] J. Wood, J. Dykes, A. Slingsby, and K. Clarke. Interactive visual exploration of a large spatio-temporal dataset : reflections on a geovisualization mashup. *IEEE Transactions on Visualization and Computer Graphics*, 13(6) :1176–1183, 2007. 2.2.3

[Wea04] Chris Weaver. Building highly-coordinated visualizations in improvise. In *INFOVIS '04 : Proceedings of the IEEE Symposium on Information Visualization*, pages 159–166, Washington, DC, USA, 2004. IEEE Computer Society. (document), 3.3.1, 3.8

[WHA07] Wesley Willett, Jeffrey Heer, and Maneesh Agrawala. Scented widgets : Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6) :1129–1136, 2007. (document), 3.2.1, 3.3

[Wil05] L. Wilkinson. *The grammar of graphics*. Springer Verlag, 2005. 2.4.2, 2.4.3

[WKM07] Martin Wattenberg, Jesse Kriss, and Matt McKeon. Manyeyes : a site for visualization at internet scale. *IEEE Transactions on Visualization and Computer Graphics*, 13(6) :1121–1128, 2007. Member-Viegas, Fernanda B. and Member-van Ham, Frank. (document), 4.3.3, 4.11

[WWB95] J. Wood, H. Wright, and K. Brodlie. CSCV-computer supported collaborative visualization. In *Proceedings of BCS Displays Group International Conference on Visualization and Modelling*. Citeseer, 1995. 4.3.3

[X3D10] X3D. http ://www.web3d.org/x3d/. 2010. 2.2.1

[YHN07] Beth Yost, Yonca Haciahmetoglu, and Chris North. Beyond visual acuity : the perceptual scalability of information visualizations for large displays. In *CHI '07 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 101–110, New York, NY, USA, 2007. ACM. 4.2.2

[YKSJ07] Ji Soo Yi, Youn ah Kang, John Stasko, and Julie Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6) :1224–1231, 2007. 4.2.1

[YSLH03] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. Faceted metadata for image search and browsing. In *CHI '03 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, New York, NY, USA, 2003. ACM. 3.3.1

[ZR08] Nan Zang and Mary Beth Rosson. What's in a mashup ? and why ? studying the perceptions of web-active end users. In *VLHCC '08 : Proceedings of the 2008 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 31–38, Washington, DC, USA, 2008. IEEE Computer Society. 5