

Travail d'Etudes et de Recherches New interaction tools : an hyperbolic interface

Sommaire

SOMMAIRE	2
INTRODUCTION	4
I. BUT DU T.E.R.	4
II. CONTRAINTES JURIDIQUES [DROIT1]	4
III. BASES MATHÉMATIQUES DE L'ESPACE HYPERBOLIQUE	5
1. INTRODUCTION AUX MODÈLES HYPERBOLIQUES PLAN [MATHS]	5
2. LA PROJECTION DE POINCARÉ : INTÉRÊT DE L'ESPACE HYPERBOLIQUE [INTERET]	5
A. OUTILS D'INTERACTION ET INTERFACES UTILISANT L'ESPACE HYPERBOLIQUE	7
I. LES ARBRES HYPERBOLIQUES OU FISHEYES-VIEW	7
1. DÉFINITION	7
2. PRINCIPE :	7
3. DICTIONNAIRE MULTILINGUES	8
4. OUTILS DE VISUALISATION DE RESSOURCES MULTIMÉDIA	9
5. NAVIGATEURS ET EXPLORATION DE SITES INTERNET	11
6. VISUALISATION DE RÉSEAUX DE LIENS DE PETITE TAILLE	16
7. NAVIGATION ET EXPLORATION DE BASES DE DONNÉES [PRODUCTS_INXIGHT]	18
8. EXPLORATION DE GRANDS GRAPHES DANS L'ESPACE HYPERBOLIQUE EN 3D	20
II. LES ARBRES CONIQUES HYPERBOLIQUES OU CONE-TREE	22
1. HISTORIQUE	22
2. PRINCIPE	22
3. EXEMPLE	23
4. REMARQUE	23
III. LES MURS EN PERSPECTIVES OU PERSPECTIVE-WALL	24
1. POUR LA VISUALISATION DE DONNÉES TEMPORELLES [MUR_TEMPS]	24
2. LE "ZOOM" POUR TABLEAUX À GRAND NOMBRE DE CELLULES [FICELL]	24
IV. LES PAVAGES DANS UN ESPACE HYPERBOLIQUE	26
B. ÉTUDE D'OUTILS EXISTANTS	27
I. APPLET JAVA DE RÉALISATION D'UN PAVAGE DANS UN ESPACE HYPERBOLIQUE	27
II. MODÈLES POUR LE PROGRAMMATION WIN32 ET MFC	32
1. BIBLIOTHÈQUE MFC	32
2. OUTIL LOUPE AVEC MASQUE CIRCULAIRE	32
3. EXEMPLE AVEC DE LA SEMI TRANSPARENCE	33
C. RÉALISATION DE L'INTERFACE	35
I. RÉALISATION D'UNE FENÊTRE RONDE ET ALPHABLENDING	35
1. PROGRAMMATION DE FENÊTRES WIN32	35
2. CRÉATION D'UNE FENÊTRE RONDE	36
3. ALPHABLENDING	39
II. RÉALISATION DU PAVAGE	40
1. ALGORITHME DE PAVAGE	40
2. MÉTHODE DE DESSIN	44

3. CLASSES ET METHODES	47
III. REALISATION DE LA NUMEROTATION BASEE SUR LA SUITE DE FIBONACCI	50
1. SUITE DE FIBONACCI	50
2. APPLICATION A LA NUMEROTATION DES CELLULES	50
IV. AFFICHAGE DES DONNEES DANS LE PAVAGE	53
1. AFFICHAGE DE NOMBRES	53
2. AFFICHAGE D'ICONES	54
3. AFFICHAGE DE COULEURS	57
V. DEPLACEMENT DES DONNEES DANS LE PAVAGE	57
 CONCLUSION	 58
 I. LIMITES	 58
II. SYNTHESE	58
III. SUGGESTIONS	58
 BIBLIOGRAPHIE	 60
 ANNEXES	 62
 I. LE DISQUE D'EUCLID ET DE POINCARÉ : BASES MATHÉMATIQUES DE L'ESPACE HYPERBOLIQUE	 62
1. INTRODUCTION AUX MODELES HYPERBOLIQUES PLAN [MATHS]	62
2. LE DISQUE DE POINCARÉ [POINCARÉ]	65
3. PAVAGES DANS L'ESPACE HYPERBOLIQUE	66
II. GLOSSAIRE	67

Introduction

I. But du T.E.R.

Les interfaces ont fortement évoluées jusqu'à devenir exclusivement graphiques. Afin d'interagir avec les différents objets d'une application, les développeurs ont mis à la disposition de tous un certain nombre de techniques, comme par exemple : **magic lens** ou des **menus en étoile**.

Ces travaux d'étude et de recherche ont pour but de mettre en place une nouvelle interface Humain - machine, basée sur la représentation de l'espace hyperbolique dans l'espace euclidien. L'intérêt principal de l'utilisation de l'espace hyperbolique dans la création de ce type d'interface, est qu'il permet de représenter un espace infini (espace hyperbolique) dans un espace fini (espace euclidien) qui est l'interface.

Tout d'abord nous chercherons à comprendre les principes mathématiques de base de l'espace hyperbolique.

Puis nous réaliserons un état de l'art des différents outils basés sur l'espace hyperbolique mis à la disposition des utilisateurs afin d'interagir le plus naturellement et le plus simplement possible. Nous étudierons principalement un programme permettant de représenter un espace hyperbolique pavé et plusieurs programmes permettant de créer des fenêtres interactives et élaborées.

Enfin, la dernière étape repose sur la réalisation d'un outil d'interaction de type disque sous l'interface Windows XP, basé sur l'utilisation des espaces hyperboliques.

Cette réalisation sera testée ultérieurement par plusieurs utilisateurs potentiels pour en déterminer l'ergonomie, et notamment, elle permettra d'interagir avec un logiciel de démonstration automatique.

II. Contraintes juridiques [droit1]

La visualisation à l'aide d'un arbre hyperbolique a été entièrement brevetée par Xerox aux Etats-Unis. L'utilisation d'une implémentation hyperbolique, quelque soit la personne l'ayant développée, est illégale dans les pays où le brevet de logiciels est applicable, si Xerox ne donne pas son accord. Inxight, la branche composants logiciels de Xerox donne apparemment le droit d'utiliser un arbre hyperbolique aux organisations à but non lucratif et aux membres de l'éducation, pour les américains. [droit2]

Mais les logiciels ne sont pas brevetables en Europe, donc ce brevet n'a aucune valeur en France d'après l'article 52, 2-c du code pénal ou civil [droit3 art 52, 2c]. [droit4]

III. Bases mathématiques de l'espace hyperbolique

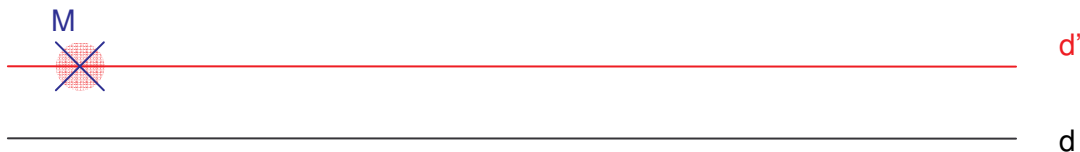
1. Introduction aux modèles hyperboliques plan [maths]

a. Le cinquième axiome

Il y a 2300 ans le mathématicien grec Euclide a fondé sa géométrie (la géométrie euclidienne ou classique) sur cinq axiomes. Le cinquième, l'axiome des parallèles, apparaît comme superflu pour beaucoup de ses collègues. Et ils essayèrent, mais sans succès, de prouver qu'il était faux. Après plus de 2000 Lobachevsky (1793-1853), Bolyai (1777-1855) et Gauss (1802-1860) ont réfuté cet axiome et ont indépendamment découvert la géométrie non-Euclidienne.

Cet axiome est :

Il existe une et une seule droite d' parallèle à une droite d donnée passant par un point M donné.



b. La géométrie hyperbolique

A la suite des travaux de Klein, on appelle hyperbolique une géométrie de Lobatchevsky pour laquelle par un point il passe deux droites non sécantes et sans perpendiculaire commune à une droite donnée. Bachmann nomme cette situation "droites non connectables". D'autres axiomatiques sont possibles, plus directes quant à l'étude des propriétés métriques : celle de Jacqueline Lelong Ferrand (JLF) qui permet de présenter la "géométrie absolue" au sens de Bolyai, c'est-à-dire qui vérifie les quatre premiers axiomes d'Euclide. Comme le fait remarquer JLF, la géométrie hyperbolique est à 80% ... euclidienne.

2. La projection de Poincaré : intérêt de l'espace hyperbolique [interet]

a. Compatibilité de la représentation et encerclement de l'infini

L'aire infinie du plan hyperbolique est entièrement incluse dans un disque : le disque de Poincaré. Le périmètre du disque représente donc l'infini et tous les points appartiennent à ce disque, sans jamais appartenir au périmètre du disque.

On peut donc représenter un espace infini dans l'espace fini d'un disque du plan euclidien.

b. Focus + Context

Le focus, c'est-à-dire la donnée centrale ou importante sur laquelle l'utilisateur porte toute son attention (le centre du disque), peut être déplacé dans tout le disque de l'espace hyperbolique. Le coefficient de zoom est de 0,5 au centre et il diminue exponentiellement avec la distance, vers le périmètre du disque. C'est pourquoi, le contexte apparaît comme étant vraiment naturel. Plus les données sont éloignées du centre, donc du focus, moins leur représentation spatiale est visible dans la représentation courante.

c. Les droites deviennent des cercles

Toutes les droites dans l'espace hyperbolique apparaissent comme des segments d'arc de cercle ou des lignes droites centrées dans le disque de Poincaré. Leur prolongement coupe toujours le périmètre du disque de Poincaré perpendiculairement en deux endroits.

d. L'axiome des parallèles

Pendant que l'espace euclidien accepte une et une seule parallèle à une droite passant par un point donné, l'espace euclidien en offre infiniment plus. Dans le disque de Poincaré, une droite est un arc de cercle qui fini perpendiculairement au périmètre du disque.

e. Une surface exponentielle

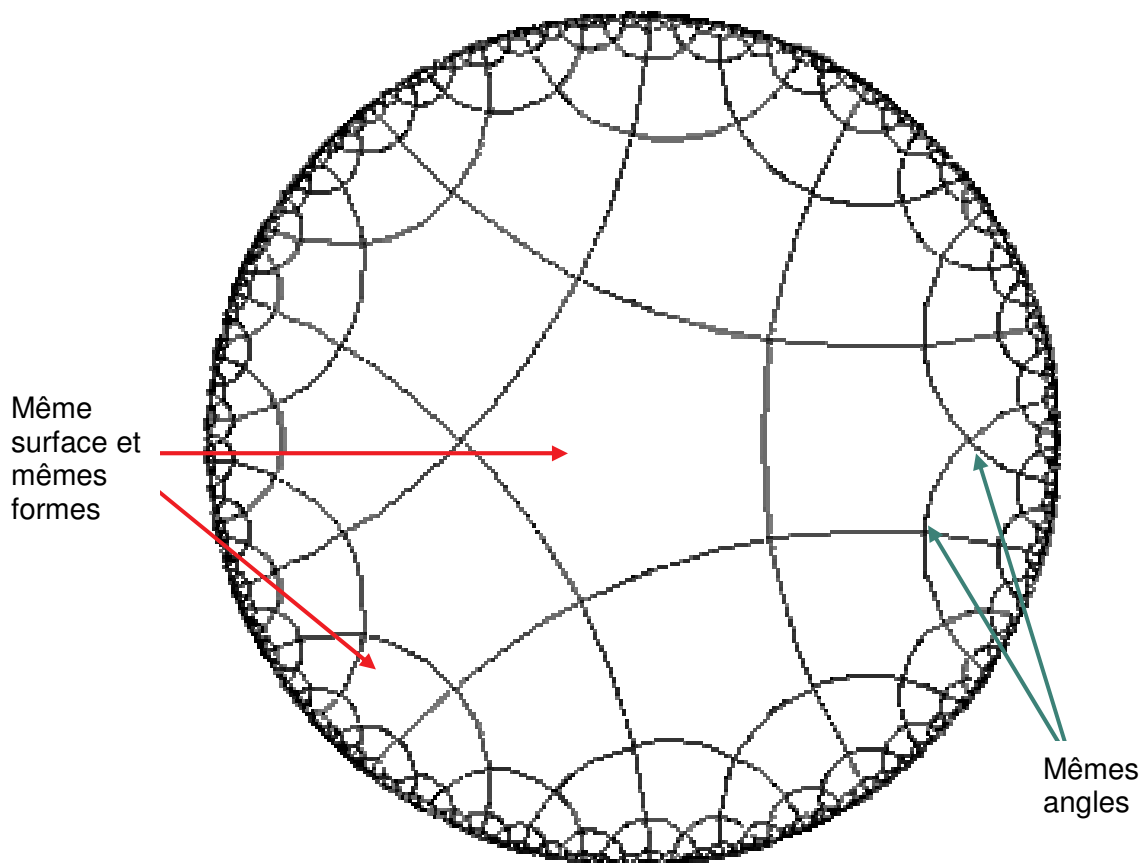
Il existe deux sortes de "parallèles" :

- les parallèles asymptotiques : les cercles qui touchent le périmètre du disque unité ont le même point infini.
- les ultra parallèles : les arcs de cercle qui n'intersectionnent pas avec le disque de Poincaré.

Il y a plus d'espace à l'infini qu'habituellement (dans l'espace euclidien). Quelles que fois l'espace hyperbolique est aussi appelé "infini plus intense" que l'espace des réels $\mathbb{R}^2$. Cet extra-espace est convoité pour trouver un bon arrangement des données.

f. Une isométrie déformée par l'espace euclidien

Les polygones représentés ont tous la même surface et les mêmes mesures d'angles. Ils sont donc isométriques dans le plan hyperbolique. Cette isométrie est déformée par l'espace euclidien qui représente des polygones de plus en plus petits en s'éloignant du centre.



Pour plus d'informations sur la géométrie hyperbolique, voir les annexes.

A. Outils d'interaction et interfaces utilisant l'espace hyperbolique

I. Les arbres hyperboliques ou fisheyes-view

1. Définition

Il s'agit de projeter un arbre dans une représentation de l'espace hyperbolique en deux dimensions afin d'obtenir la visualisation intégrale de l'arbre (quelque soit sa taille, pouvant aller jusqu'à plusieurs centaines de nœuds) dans le disque unité. L'avantage apparaît clairement : tout l'arbre est visualisable d'un seul coup d'œil.

Souvent, plusieurs fonctions dynamiques ont été ajoutées, afin de faire glisser l'arbre au centre du disque, pouvant ainsi agrandir la partie voulue grâce à la fonction "zoom" ou aux actions click & drag de la souris (cliquer/déplacer).

C'est un outil visuel avancé de navigation et d'exploration d'une hiérarchie de données.

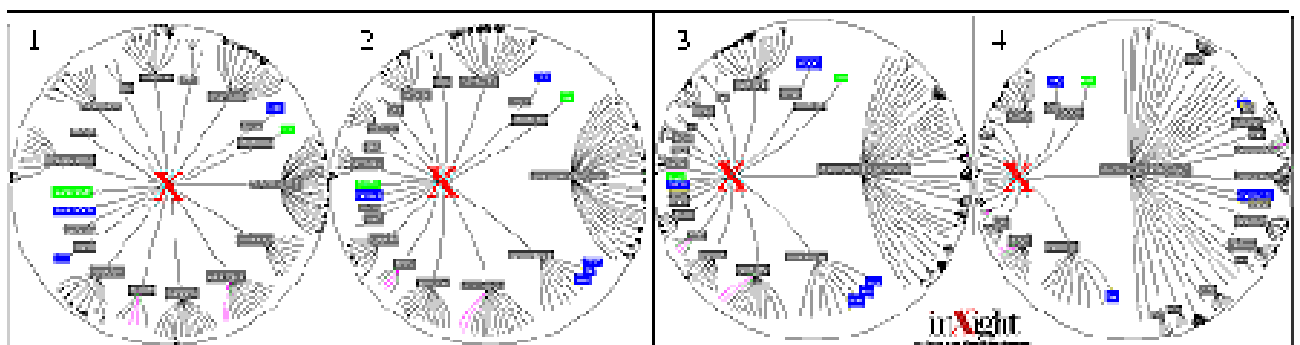
Avec cette technologie d'arbre hyperbolique ou en étoile, l'organisation la plus complexe et le contenu peuvent être visualisés et explorés, rapidement et facilement.

Les arbres hyperboliques ont pour but de faciliter la navigation à l'intérieur de la structure et permettent :

- de garder des informations sur le contexte et la périphérie immédiate de l'objet principal : context
- d'afficher autant d'informations pertinentes que possible en choisissant une représentation adaptée au type de données et en mettant en avant une donnée principale : focus.

2. Principe :

Par un simple clic sur un noeud, on change le **focus** et on obtient une vue centralisée autour de notre centre d'intérêt. On peut rapprocher cette méthode de la vue en oeil de poisson ou "**fisheyes-view**". [inxight 1]



Un arbre hyperbolique [inxight1]

Dans l'exemple ci-dessus nous pouvons voir la visualisation de données hiérarchiques. Dans le cadre 1, nous pouvons voir la configuration initiale. L'utilisateur déplace progressivement (cadre 2, 3, 4) son centre d'intérêt de l'objet central vers le **sous-arbre** de droite. Plusieurs choses sont à remarquer.

Par exemple, à la fin de la manipulation, l'utilisateur a complètement conscience des objets à sa droite et a des informations complémentaires par rapport à la configuration initiale.

On peut aussi remarquer que les objets en bas à droite au cours du déplacement se sont rapprochés du centre de la fenêtre pour finalement s'en éloigner. La granularité de la description de ce sous-arbre a reflété ces différents changements d'états.

Ce type de visualisation de données hiérarchiques peut être utilisé dans des applications telles que :

- les dictionnaires multilingues
- les outils multimédia de visualisations divers
- La visualisation de la structure de navigation
 - les browsers et l'exploration de sites Internet
 - l'exploration de larges bases de données
- l'exploration de grands graphes dans l'espace hyperbolique en 3D

Remarque d'un point de vue ergonomique

L'inconvénient majeur de cette approche est qu'elle est peu intuitive, et qu'elle reste relativement difficile à appréhender pour une personne non habituée ; si l'arbre a trop de niveaux, il devient vite difficile de visualiser le chemin parcouru et donc de s'y retrouver.

3. Dictionnaire multilingues

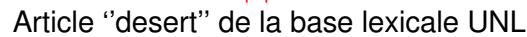
a. Principe [dictionnaire]

Le projet UNL utilise à l'heure actuelle environ 16 langues. Les bases lexicales UNL de chaque langue sont composées d'un dictionnaire bilingue reliant les unités lexicales UNL aux lemmes (proposition qui prépare à la démonstration d'une autre proposition) de la langue. Une unité lexicale (UW) est composée d'un mot-vedette (headword) anglais suivi éventuellement d'une liste de restrictions illustrant un sens précis du mot-vedette. Un seul mot-vedette regroupe donc plusieurs sens avec des restrictions différentes. Chaque sens peut avoir des traductions dans chaque langue du projet.

Pour visualiser un mot-vedette UNL, ses différentes acceptions et leurs différentes traductions, nous avons utilisé un visualisateur d'arbres hyperboliques développé par la société InXight. Ce visualisateur permet de naviguer dans l'arbre en cliquant sur les nœuds, et de relier des nœuds à des pages html grâce à des liens hypertextes. C'est une applet programmée en java qui lit un fichier texte représentant un arbre en entrée et l'affiche à l'écran comme un arbre hyperbolique. Il est possible de spécifier les couleurs de chaque nœud et de chaque arc.

b. Exemple d'arbre hyperbolique [dictionnaire]

L'utilisateur demande un mot-vedette. La maquette consulte alors les dictionnaires UNL disponibles pour chaque langue et extrait toutes les acceptions correspondant au mot-vedette demandé ainsi que leurs traductions. Un fichier texte représentant l'arbre est ensuite construit à la volée, puis affiché à l'aide de l'applet d'arbre hyperbolique sur l'écran de l'utilisateur. On peut ensuite naviguer dans l'arbre avec la souris.



Un principe ergonomique, qui est d'associer à chaque langue une couleur, a été adopté. Les traductions françaises des acceptions sont coloriées en bleu, par exemple : "désert2", "faire_défection", "délaisser", "abandonner". Les traductions italiennes sont coloriées en vert : "desertico" & "disertare". Les traductions japonaises sont coloriées en rouge et comme le logiciel utilise Unicode, les traductions japonaises sont donc correctement codées.

- identification automatique de la langue (automatic language identification)
- la segmentation en mots (tokenization)
- la lemmatisation (stemming)
- l'étiquetage des parties du discours (part-of-speech tagging)
- et l'extraction de groupes nominaux (noun/phrase extraction) ;

4. Outils de visualisation de ressources multimédia

Un exemple de liens externes toujours dans le cadre de l'édition d'un mémoire est donné par les liens associés aux références bibliographiques et qui permettent d'accéder aux articles cités.

b. Treebolic [treebolic]

Treebolic est un applet Java, s'insérant dans des pages Web, destinées à visualiser un ensemble hiérarchisé de données sous forme d'arbre hyperbolique.

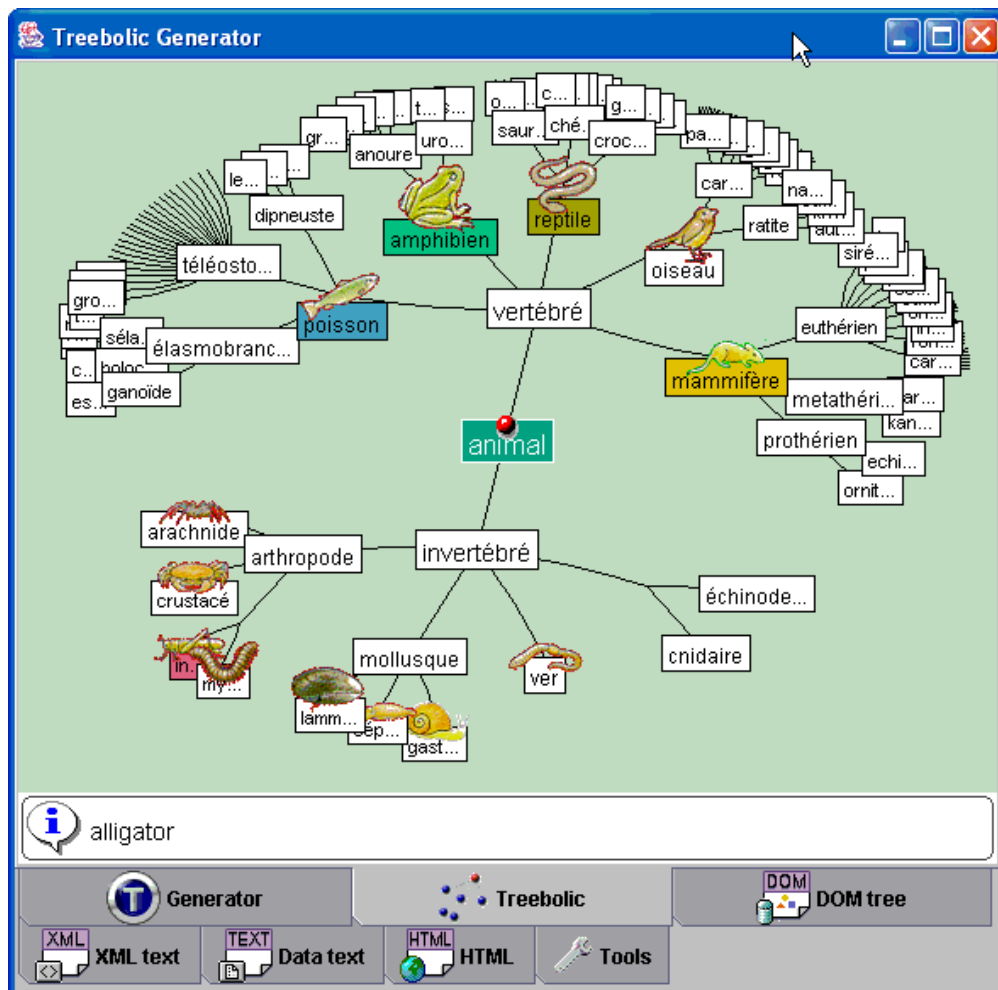
Un arbre est affiché avec ses noeuds et ses arcs mais l'espace de visualisation subit une courbure en allouant davantage de place au centre, le noeud parent et les descendants immédiats apparaissent légèrement plus grands. Les grands-parents et petits fils sont toujours visibles mais sont encore plus petits.

A mesure que l'on s'éloigne du foyer, les noeuds se voient allouer moins d'espace et disparaissent ainsi virtuellement vers la limite du disque de visualisation, comme si toute cette hiérarchie était vue à travers un **objectif fisheye**, visible dans sa totalité mais sans que ce soit au détriment du contexte.

L'arbre hyperbolique est dynamique : une animation amène un noeud visité au centre de l'affichage. Les noeuds peuvent contenir des liens hypertextes et mener le navigateur vers d'autres pages.

Treebolic est piloté par une description **XML** de l'arbre, ce qui rend les données faciles à écrire et à vérifier. Cette description doit se conformer à un schéma XML précis et documenté ici et peut être générée grâce à des outils XML standards. Parmi ces derniers, le Générateur Treebolic.

Treebolic est un applet écrit en Java et est donc indépendant de la plateforme (Windows, Mac, Linux, Unix) et du navigateur (Internet Explorer ou Netscape) pourvu qu'il possède Java.



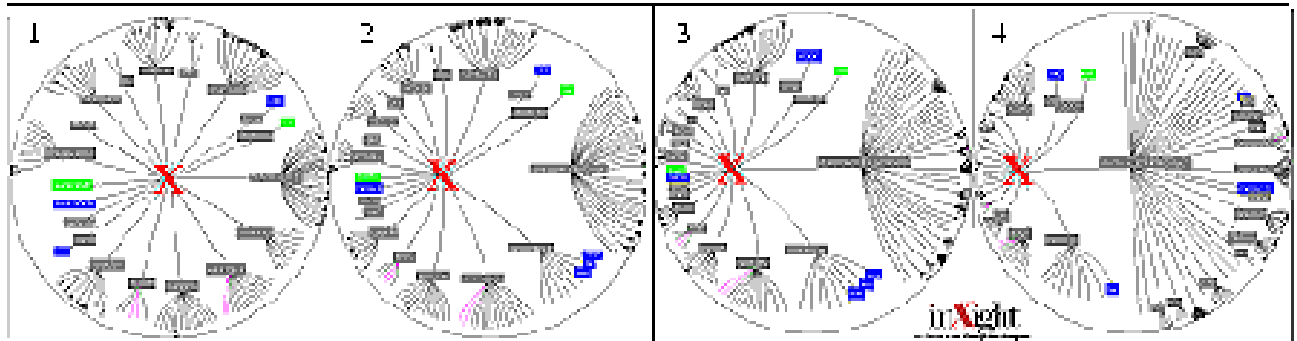
Treebolic : logiciel de construction et de visualisation des arbres hyperboliques grâce à XML [treebolic]

c. Madeus InXight

Au sein de Madeus, le scénario temporel est fondé sur :

- l'organisation hiérarchique des données ;
- l'utilisation de contraintes de placement.

L'objectif est d'aider l'auteur d'un document multimédia en lui offrant une vue graphique de son scénario qui lui permette de visualiser l'ensemble des placements possibles des objets.



Un arbre hyperbolique [inxight1]

5. Navigateurs et exploration de sites Internet

a. Visualisation de réseaux de liens de grande taille

La visualisation de nombreux liens hypertextes ou hypermédia nécessite l'utilisation de méthodes de visualisation adaptées pour permettre une visualisation claire et éviter des vues trop complexes et incompréhensibles (voir un contre exemple sur la Fig 4).

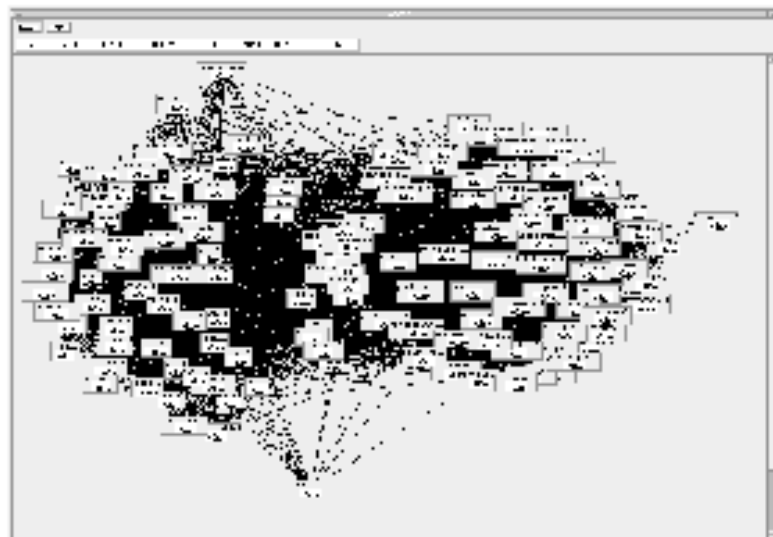


Figure 1. Nécessité d'avoir des méthodes de visualisation adaptées

La visualisation des liens entre les documents revient du fait de la grande taille du graphe, à visualiser une véritable toile d'araignée. Différentes techniques pour améliorer la visibilité de ces graphes ont été proposées. On peut par exemple, citer les méthodes de vue en oeil de poisson (Figure. 2).

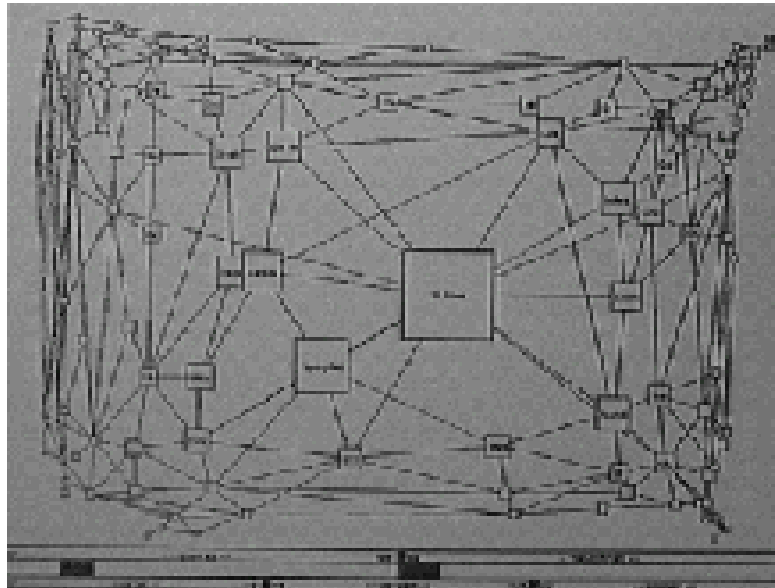


Figure 2. Vue en oeil de poisson

La technique de la vue en oeil de poisson (Figure 2), a pour objectif d'améliorer la lisibilité du graphe en centralisant la vue sur un élément particulier.

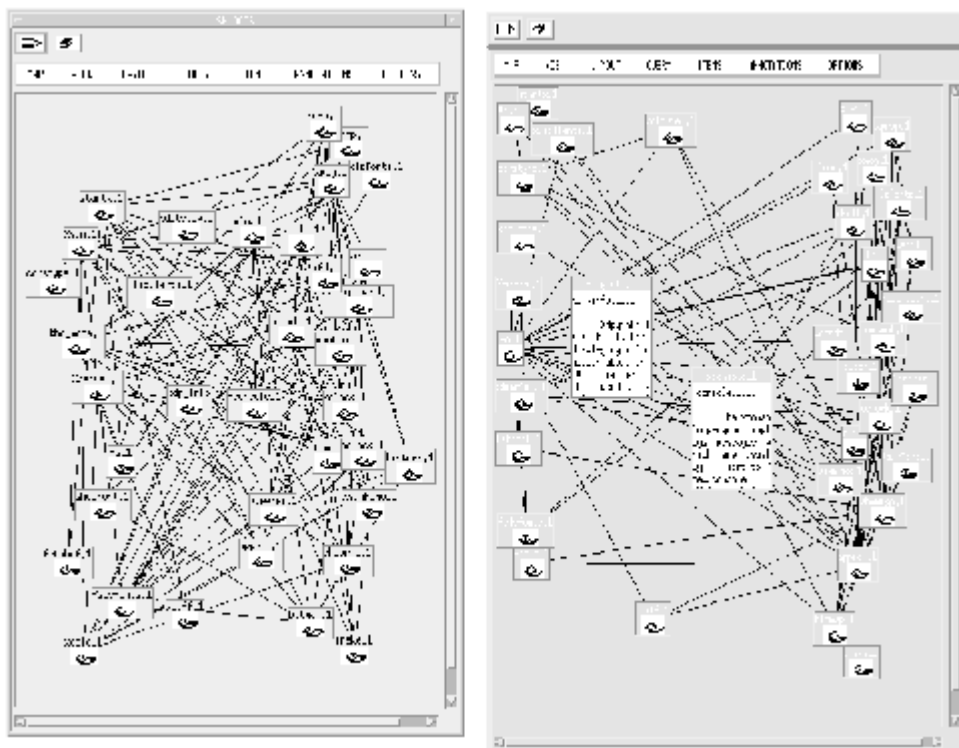


Figure 3. Réseau avec et sans vue en oeil de poisson

Exemples d'utilisation :

- Structure de sites Internet (liens entre les différentes pages du site)
- Représentation physique de réseaux informatiques

b. Navigateur hyperbolique

Le navigateur hyperbolique utilise la technique **focus**+contexte pour visualiser des données hiérarchiques avec un affichage dynamique. L'utilisateur peut fixer son attention sur une partie **détaillée des données, celle qui l'intéresse, comme si il la regardait au travers d'une loupe, tandis que les autres données, mises en retrait, apparaissent plus discrètement.**

<http://www.object-everywhere.com/OBJECT-EVERYWHERE/articles/NavigateurHyperbolique.pdf>

Problème :

Se perdre en suivant des liens hypertextes est l'un des problèmes associés au Web. Webviz [Webviz] utilise la géométrie hyperbolique pour donner une vue ultra-grand-angulaire des liens à destination et en provenance des pages Web. [nlc]

c. Exemples de navigateurs hyperboliques

Les exemples requièrent un navigateur qui accepte JavaScript

Principe :

Permet de visualiser la structure d'un site Internet et les liens entre les différentes pages Web qui le composent.

- Arborescence des sites SVT [svt]

Mode d'emploi :

- 1 - Cliquer sur un noeud. Il se positionne au milieu de l'écran.
- 2 - Cliquer et tirer sur un noeud pour voir le sous arbre.
- 3 - Utiliser les icônes pour rapprocher ou éloigner les noeuds, augmenter ou diminuer la taille des caractères, chercher un mot dans les titres des pages HTML.
- 4 - Double-cliquer sur un noeud pour afficher la page HTML correspondante (voir la barre des applications en bas de page).
- 5 - Dans l'arborescence des sites SVT, double-cliquer sur un noeud pour ouvrir l'arbre du site dans une nouvelle fenêtre.

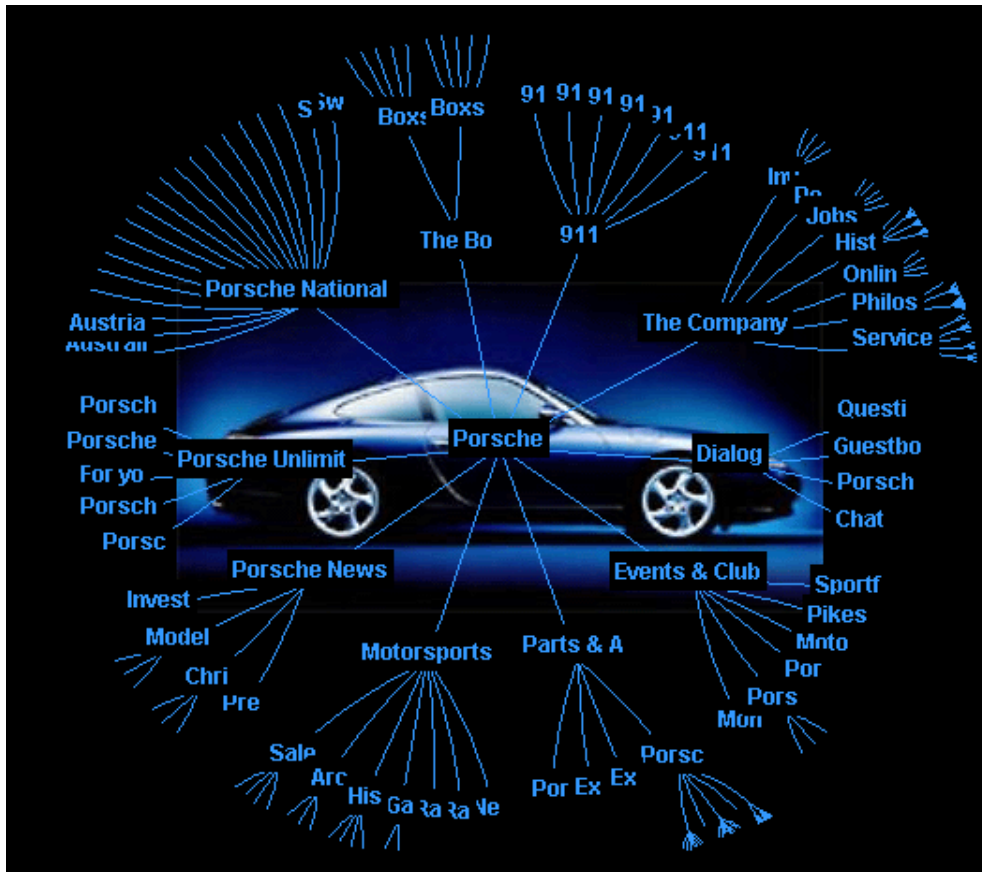
Figure : <http://www.inrp.fr/Acces/Biogeo/inxight/slsbeta/menrt/menrt.html>

Ou <http://www.inrp.fr/Acces/Biogeo/inxight/slsbeta/ressvt/ressvt.html>

- Inxight Star Tree SDK [inxight2] [products inxight]

Le site Internet de InXight offre la possibilité de visualiser les liens entre les pages avec un arbre hyperbolique sous la forme d'un applet Java : l'arbre SDKs d'étoile d'Inxight (autrefois connu sous le nom d'arbre hyperbolique SDKs) un outil visuel avancé de navigation et d'exploration. Avec la technologie d'arbre d'étoile, une organisation complexe des catégories et le contenu peuvent être visualisés et explorés, rapidement et facilement.

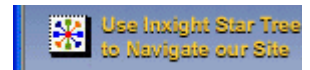
Inxight Star Tree est la technologie brevetée d'Inxight pour la navigation et la visualisation de sites Internet et autres collections hiérarchiques d'information. Star Tree est particulièrement efficace pour organiser et montrer d'énormes quantités de données qui peuvent être rencontrées dans les domaines du commerce virtuel, de l'édition électronique, ou du développement Web. Une interface graphique présente les hiérarchies complexes dans une vue facile à naviguer. La technologie Star Tree forme la base pour le Inxight Star Tree SDK dont l'API intègre les capacités de Star Tree afin de visualiser et de naviguer dans les applications d'entreprise ainsi qu'OEM.



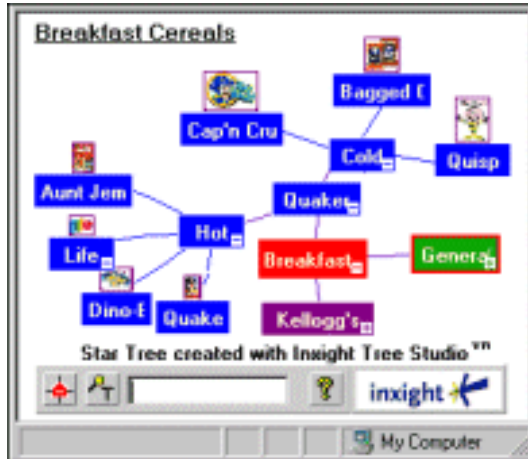
Un exemple de l'utilisation de Inxight Star Tree SDK avec le site de Porsche

Voir aussi les réalisations suivantes avec Inxight tree Studio

- [BestBuy](#)
- [Porsche](#)
- [CIGNA](#)
- ou la navigation sur le site de Inxight : <http://www.inxight.com/>



Utilisation de Inxight Star Tree :



Clicker sur un noeud pour le centrer à l'écran.

Double-clicker sur un noeud pour ouvrir la page HTML associée.

Passer le curseur sur un noeud pour lire sa description.

Clicker sur le bouton du centre de l'arbre Star Tree dans la barre d'outils et l'arbre et recentré.

Clicker sur le signe "+" d'un noeud pour développer sa branche et voir ses enfants. Clicker sur le signe "-" du noeud pour contracter la branche et cacher ses enfants.

Tous les noeuds et les espaces entre les noeuds de l'arbre peuvent être déplacés avec la souris maintenue appuyée.

La barre d'outils

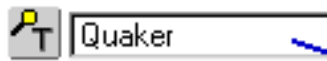
La barre d'outils contient un groupe de boutons qui aident à la visualisation de l'arbre et à la recherche de noeuds dans l'arbre. L'applet montre ou cache les boutons en fonction des préférences du webmaster.



	Centre le noeud racine, origine, au milieu de l'applet.
	Contracte ou étend l'arbre, en fonction de la flèche du bouton sur laquelle on clique.
	Montre ou cache les vignettes des nœuds.
	Change le layout (agencement) de l'arbre.
	Augmente ou diminue la taille de la police des noeud en fonction de la flèche du bouton sur laquelle on clique.
	Montre les noms longs en étendant les étiquettes des noeuds.
	Lance une recherche textuelle ou numérique dans les descriptions des noeuds, ainsi que les adresses URL dans tous les champs de texte. Cf. utiliser la recherche.

Utiliser la recherche

Entrez la chaîne de caractères recherchée dans la zone de saisie de texte



Tous les nœuds contenant la chaîne de caractères recherchée sont surlignés



Il existe 3 options de recherche, qui apparaissent cycliquement par click sur le bouton de lancement de la recherche



La recherche s'effectue juste sur les étiquettes des nœuds.



La recherche s'effectue juste dans les URL des nœuds



La recherche s'effectue dans tous les champs de texte des

Images et quantités

Montre quel est le degré de différence entre les nœuds.



Des images peuvent illustrer ou remplacer les étiquettes des nœuds pour donner des indices visuels sur le contenu.

6. Visualisation de réseaux de liens de petite taille

Dans le cas de petites structures de navigation, fournir des moyens visuels est un moyen efficace pour visualiser et naviguer au travers de ces structures.

Bien que ce besoin existe, ces moyens visuels sont rarement utilisés dans des systèmes de conception de document multimédia. Ils sont issus d'une autre communauté : l'édition de document hypermédia.

Un document hypermédia est un document hypertexte usuel dans lequel des objets peuvent être de nature "multimédia" (vidéo, son, ...). Dans ces documents l'organisation temporelle des objets est entièrement portée par la structure de navigation offerte au lecteur. La différence entre un document multimédia et un document hypermédia est l'existence du scénario temporel.

Exemple :

AOL Press autorise un découpage hiérarchique des fichiers, à chaque niveau de la hiérarchie on peut lier les fichiers entre eux (par exemple entre Introduction et Conclusion dans la Fig 1). Néanmoins ces liens ne sont pas limités à un seul niveau de la hiérarchie, ils peuvent être mis entre des fichiers de niveaux différents (par exemple entre Introduction et Section 1.1 dans la Fig 1). Le système n'offre pas de vue globale de ces liens, il offre seulement une vue par niveaux. Pour résoudre le problème de la visualisation des liens entre des fichiers de niveaux différents, une image grisée du fichier du niveau différent est ramenée au niveau courant de la hiérarchie (la Section 1.1 de la Figure 1).

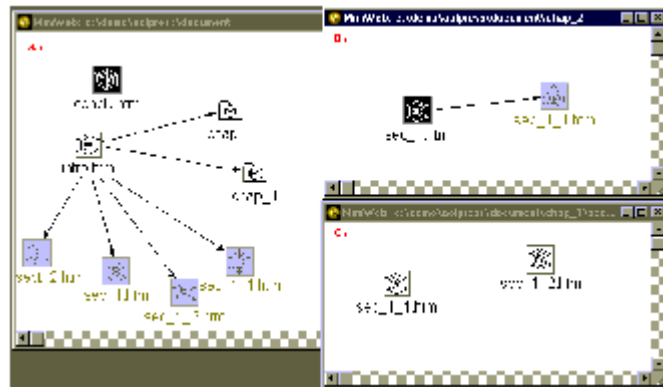


Figure 1. Un niveau de hiérarchie avec des liens

On peut mettre en doute la capacité de cette solution de visualisation à passer à l'échelle. La Figure 2 permet de mettre en évidence qu'une représentation adaptée à de petit volume d'information ne s'adapte pas directement à de gros volume.

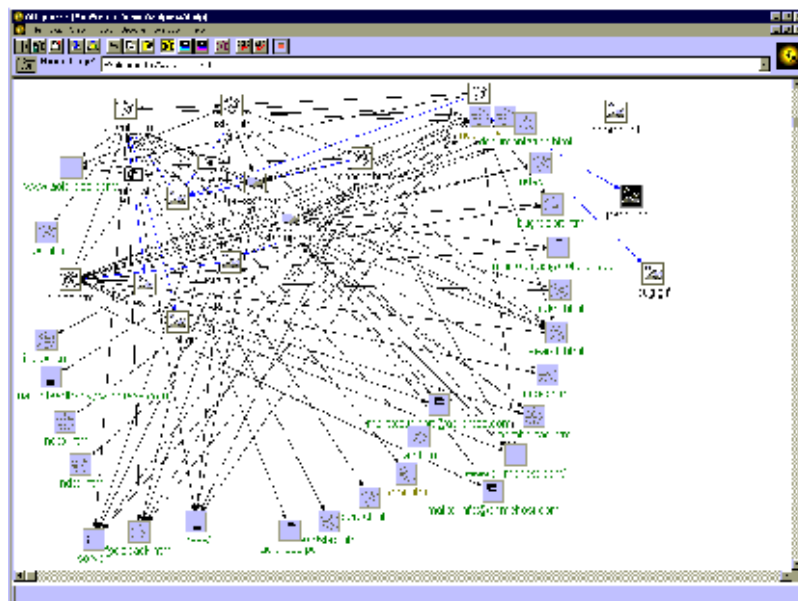


Figure 2. Représentation d'une scène complexe

7. Navigation et exploration de bases de données [products_inxight]

a. Inxight SmartDiscovery : recherche d'information en entreprise

Fruits des découvertes du PARC et du Centre Européen de Recherche de Xerox, les produits Inxight sont utilisés pour les applications à fort contenu informatif tel qu'on en trouve dans les entreprises notamment celles des services financiers, des produits pharmaceutiques, de la publication électronique et de l'administration. Les solutions Inxight sont des logiciels puissants et simples d'emploi pour organiser, comprendre, retrouver et utiliser au mieux la valeur du contenu non structuré des données dans toutes les plus grandes langues du monde. Les solutions Inxight s'appuient sur les technologies de traitement des langages naturels afin d'être rapides et exactes lors de l'automatisation de la recherche et de la délivrance automatique des données.

Inxight SmartDiscovery 3.0 est une solution complète, puissante et simple d'utilisation pour automatiser la recherche et la délivrance de texte inter-entreprise en plusieurs langues. Grâce à des dispositifs majeurs comme la gestion de taxonomie (utilisation des unités systémiques dans la science des classifications : espèce, genre,...), la classification par classes d'entreprise et un environnement de recherche guidée de l'information, SmartDiscovery exploite la richesse de l'information : fichiers MS-Office, pages Intranet, e-mail.

SmartDiscovery s'intègre sans problème aux autres systèmes de gestion des documents, pages Web et portails de l'entreprise et offre un système d'analyse de l'information.

b. Inxight VizServer : accélère et simplifie l'exploration et la découverte du contenu de larges banques de données

Grâce à l'utilisation de puissante représentation visuelle des données (les arbres hyperboliques), VizServer donne des hiérarchies de documents les plus complexes et les plus volumineuses une vue facile à comprendre qui permet aux utilisateurs de trouver rapidement et facilement ce qu'ils cherchent.

Les techniques utilisées dans les VizServer d'Inxight (arbres hyperboliques) se sont montrées plus de 60% plus rapides que les méthodes traditionnelles de recherche d'information dans de vastes bases de données. Inxight VizServer peut importer des données depuis des bases de données relationnelles permettant une intégration des données structurées et non structurées.

c. EBI (European Bioinformatics Institute) : système de visualisation pour la bioinformatique

Le système de visualisation hyperbolique d'EBI utilise CORBA pour afficher l'arbre NCBI taxonomique situé dans une base de données SYBASE.

Mode d'emploi:

<Bouton souris> ou <Bouton gauche souris> = déplacement

<Bouton souris + ALT> ou <Bouton central souris> = rotation de l'origine

<Bouton souris + META> ou <Bouton droit souris> = modifie la position courante en position d'origine

<Roulette souris> = Zoom

Légende:

Rouge = Accession Number

Orange = Feature Type

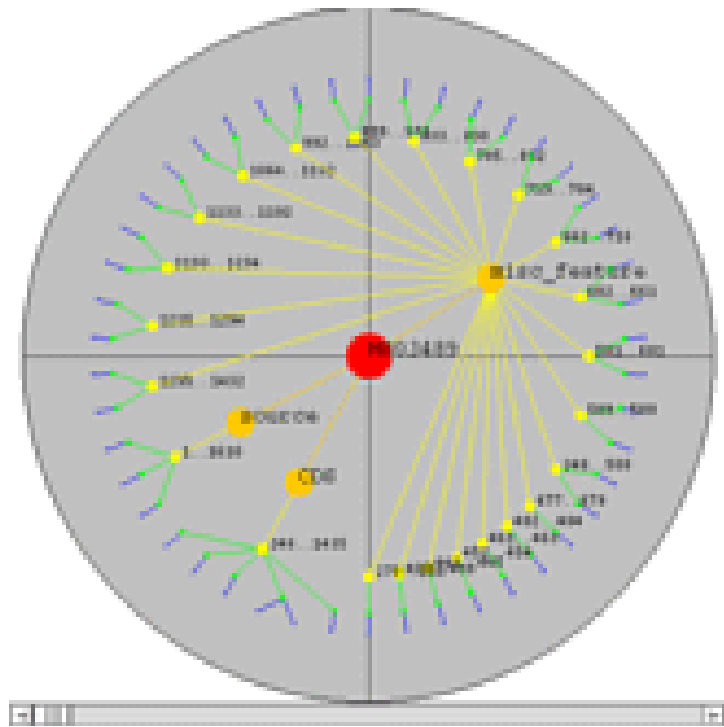
Jaune = Feature Location

Vert = Feature Qualifier Name

Bleu = Feature Qualifier Value (where applicable)

Key: imprime écran à faire

Exemples :



The feature table of [an EMBL entry](#) (Accession number = MS03489) was turned into a hierarchical tree and then displayed in the EBI hyperbolic viewer.

8. Exploration de grands graphes dans l'espace hyperbolique en 3D

a. H3 : configuration de grands graphes orientés dans l'espace hyperbolique en 3D [H3]

H3 permet de représenter de plus grandes structures que celles gérées par les techniques de représentation traditionnelles puisqu'il assume la nature hiérarchisée des données. La hiérarchie du graphe est imposée par l'utilisation de connaissances spécifiques au domaine afin de trouver un arbre approprié. Les liens qui ne font pas partie de l'arbre ne doivent pas influencer la représentation mais peuvent être dessinés sur requête de l'utilisateur.

Le volume de l'espace 3D hyperbolique augmente de façon exponentielle, contrairement à l'espace 3D de la géométrie classique : l'espace euclidien. H3 exploite cette augmentation exponentielle de l'espace en implémentant la représentation en accord avec la métrique hyperbolique. H3 optimise l'algorithme de représentation d'arbre conique pour l'espace 3D hyperbolique en plaçant les enfants sur une hémisphère autour du cône central au lieu de les placer sur son périmètre. La navigation hyperbolique fournit une vue Focus+Context de la structure avec un minimum d'encombrement et de désordre visuels. H3 a été testé avec des hiérarchies de plus de 20000 nœuds. H3 permet une navigation à travers des graphes trop grands pour être interactifs sans que l'utilisateur ne soit obligé de les tronquer ou de déployer au fur et à mesure les sous-arbres.

Site Manager

An implementation of the system described in the paper is included in this free web publishing package from SGI.

H3/H3Viewer libraries

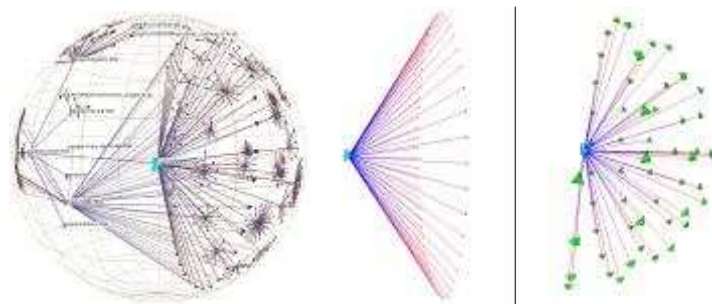
available for free noncommercial use

H3 Video

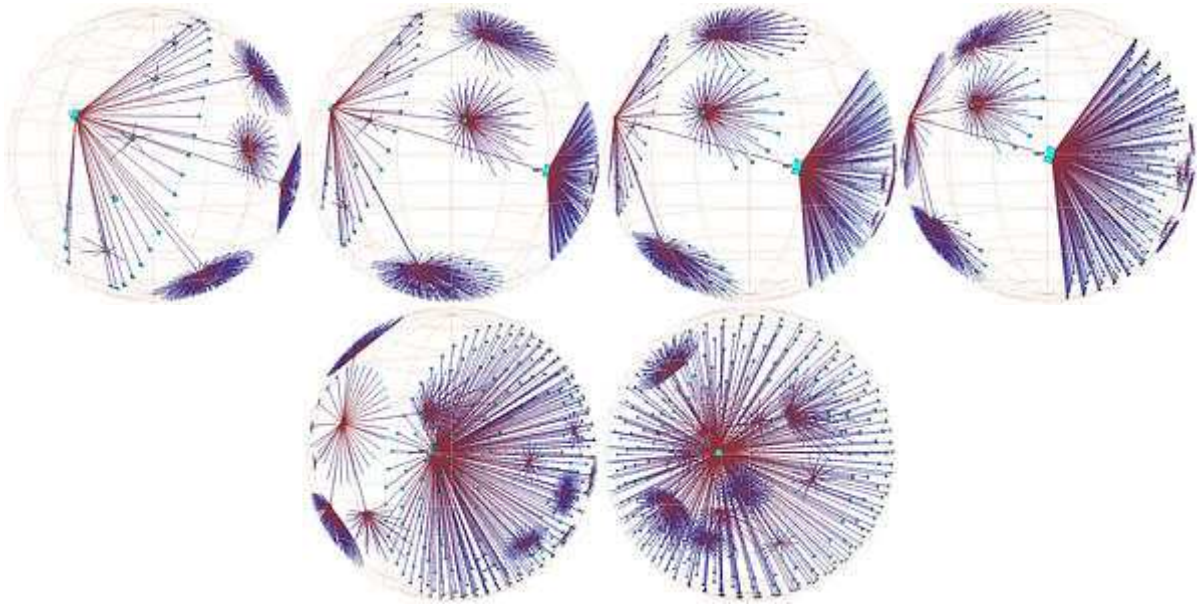
downloadable

- [Drawing Large Graphs with H3Viewer and Site Manager](#)
- [Exploring Large Graphs in 3D Hyperbolic Space](#)

Exemples de grands graphes orientés représentés avec H3 :



Exemple de déplacement et d'exploration dans un grand graphe orienté avec H3 :



II. Les arbres coniques hyperboliques ou cone-tree

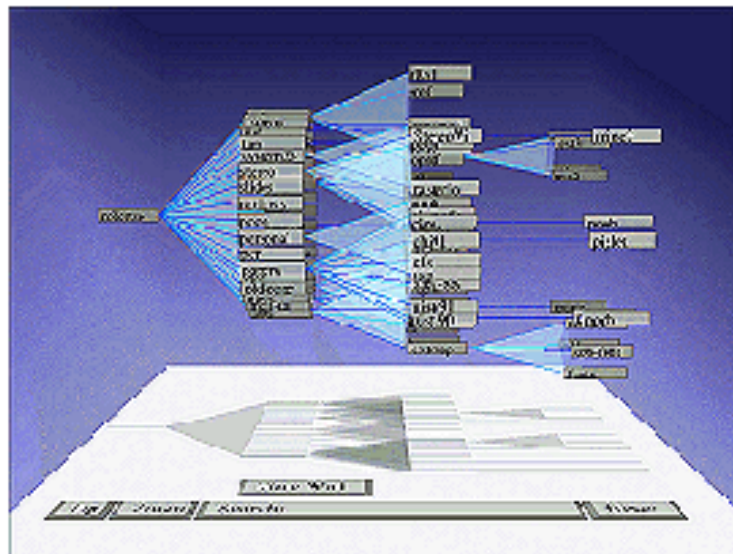
1. Historique

Élaborés au Palo Alto Research Center (PARC) de Xerox, les **arbres coniques** servent à représenter les hiérarchies telles que les répertoires de fichiers et les sites Web. On peut représenter beaucoup d'autres liens sur un écran en utilisant trois dimensions plutôt que deux. Il est possible de faire pivoter l'arbre conique pour voir les objets en détail.

2. Principe

La première méthode de visualisation que nous allons présenter fait partie des systèmes de visualisation en 3D, les arbres coniques. Un arbre conique est une représentation tridimensionnelle d'une hiérarchie dans laquelle on associe à chaque noeud le sommet d'un cône, et on arrange ses fils autour de la base circulaire du cône. C'est ce que nous pouvons voir dans la figure ci-dessous.

Le but de cette représentation est de présenter la structure de façon telle que toute la hiérarchie ou du moins la majeure partie puisse être visualisée sans que l'utilisateur ait besoin d'utiliser les barres de défilement (scroll bars). Cette approche souffre d'une limitation en termes de nombre de noeuds affichables.



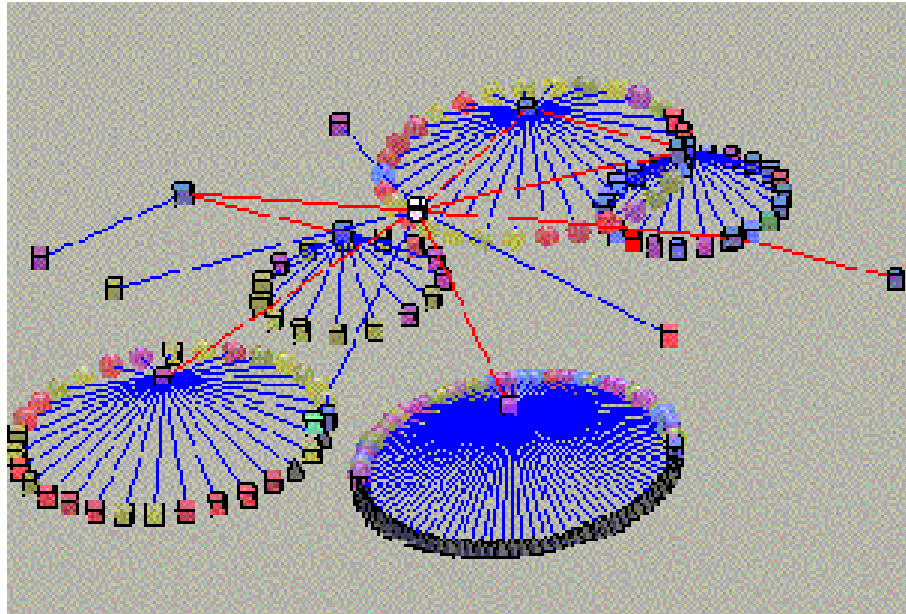
Un arbre conique

Dans l'exemple ci-dessus, nous pouvons voir l'arborescence d'un système de fichiers Unix. Cette représentation nous permet d'avoir une vue globale et assez explicite de l'arborescence.

Une approche vise à étendre cette visualisation en utilisant des métaphores de couleurs et de formes pour permettre à l'utilisateur de mieux naviguer dans la structure. De plus, les concepteurs de cette approche ont proposé différentes améliorations pour faciliter le travail de compréhension de l'utilisateur et améliorer le temps de réponse lors de la manipulation de la structure. Par exemple, ils ont proposé une vue réduite de la hiérarchie qui permet de voir la forme de la hiérarchie (ou d'une partie) sans avoir les détails.

3. Exemple

Une variante de la visualisation de données hiérarchiques par l'intermédiaire des arbres coniques (figure ci-dessous) a été proposée par Xerox . Cette approche offre des facilités de manipulation et des agréments de visualisation comme des ombres et des mouvements continus lors des rotations d'une partie de la hiérarchie.



Un arbre conique de Xerox

4. Remarque

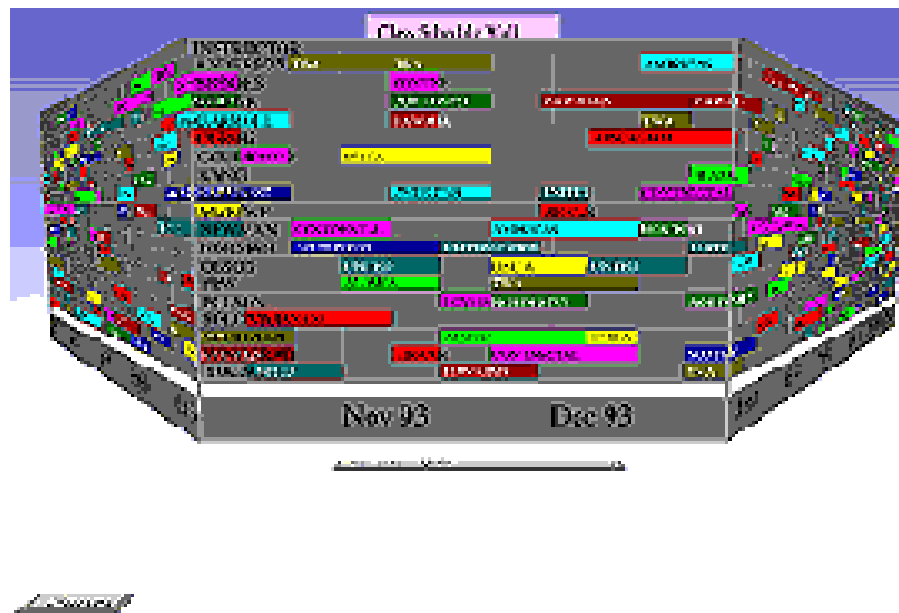
L'utilisation des arbres coniques semble bien adaptée à la visualisation de grosses hiérarchies. Cependant cette représentation circulaire semble peu adaptée au cas où les éléments de la hiérarchie sont reliés par des informations temporelles linéaires et non pas circulaires.

III. Les murs en perspectives ou Perspective-Wall

1. Pour la visualisation de données temporelles [mur_temps]

Le mur en perspective est une méthode de visualisation bien adaptée pour visualiser des grandes quantités d'informations que l'on peut ordonner selon un critère suivant l'axe des X. Cette méthode semble bien adaptée pour représenter des informations temporelles, dans ce cas le critère utilisé est le temps. L'idée à la base de ce concept est d'utiliser une vue en perspective (figure ci-dessous) pour palier les limitations physiques imposées par les tailles des fenêtres d'un système de fenêtrage classique. De cette manière, l'auteur a conscience de la totalité des informations.

Un des murs en perspective a été présenté à Xerox (ci-dessous) pour visualiser des informations classées chronologiquement.



Un perspective wall "classique"

Le mur en perspective, dans le cas de visualisation d'informations temporelles a l'avantage de permettre la visualisation d'un plus grand volume d'informations tout en gardant au premier plan une vue proche d'un "time-line" qui permet de visualiser la période temporelle au centre de l'intérêt du lecteur.

2. Le "zoom" pour tableaux à grand nombre de cellules [FiCell]

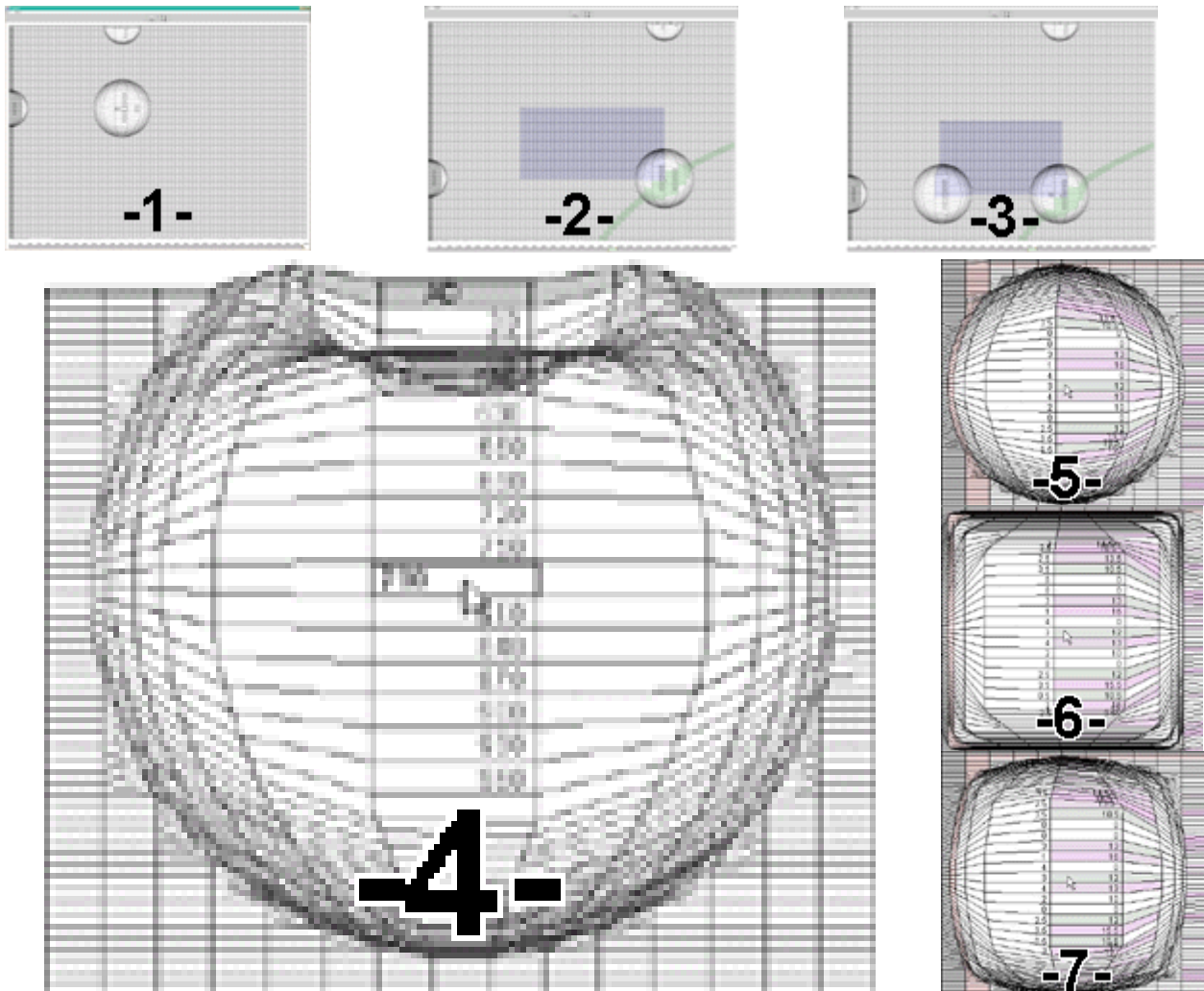
a. Principe

FiCell est un sigle signifiant Fisheye Cell.

FiCell est un système de visualisation graphique et interactif d'une table de données (de type tableur Excel par exemple). Il reprend l'idée de la vue en colline et l'interaction permet à l'utilisateur de déplacer le focus en même temps que son curseur de souris. La vue est donc toujours détaillée sous le curseur et l'utilisateur peut utiliser sa souris pour pointer (sélectionner, éditer) les cellules grossies. La vue en colline offre également une vue globale que l'utilisateur utilise à tout moment pour se déplacer. Afin de rendre saillant cette vue, FiCell offre un outil pour colorier les cellules du tableur selon la valeur du chiffre contenu dans la cellule.

FiCell peut copier ou coller un morceau de tableau qui est compatible avec Excel ainsi que sauver/charger un type de fichier qui lui est propre. La deuxième version permet de visualiser de plus grands tableaux (18000 cellules sur un écran 1600*1200).

b. Photos d'écran



- 1- La vue en colline sur un tableau de valeurs avec 2 autres collines pour les titres de colonne et de ligne
- 2- Une grande sélection en bleu et une colorisation des valeurs entre 9500 et 10000 en vert
- 3- Deux collines sur la première et la dernière colonne de la sélection pour une tâche de comparaison
- 4- Gros plan sur la fusion de deux collines (ici le focus et le titre de colonne)
- 5- Focus sphérique
- 6- Focus Pyramidale
- 7- Focus mi sphérique mi pyramidale

c. Fonctionnalités

FiCell peut copier ou coller un morceau de tableau qui est compatible avec Excel.

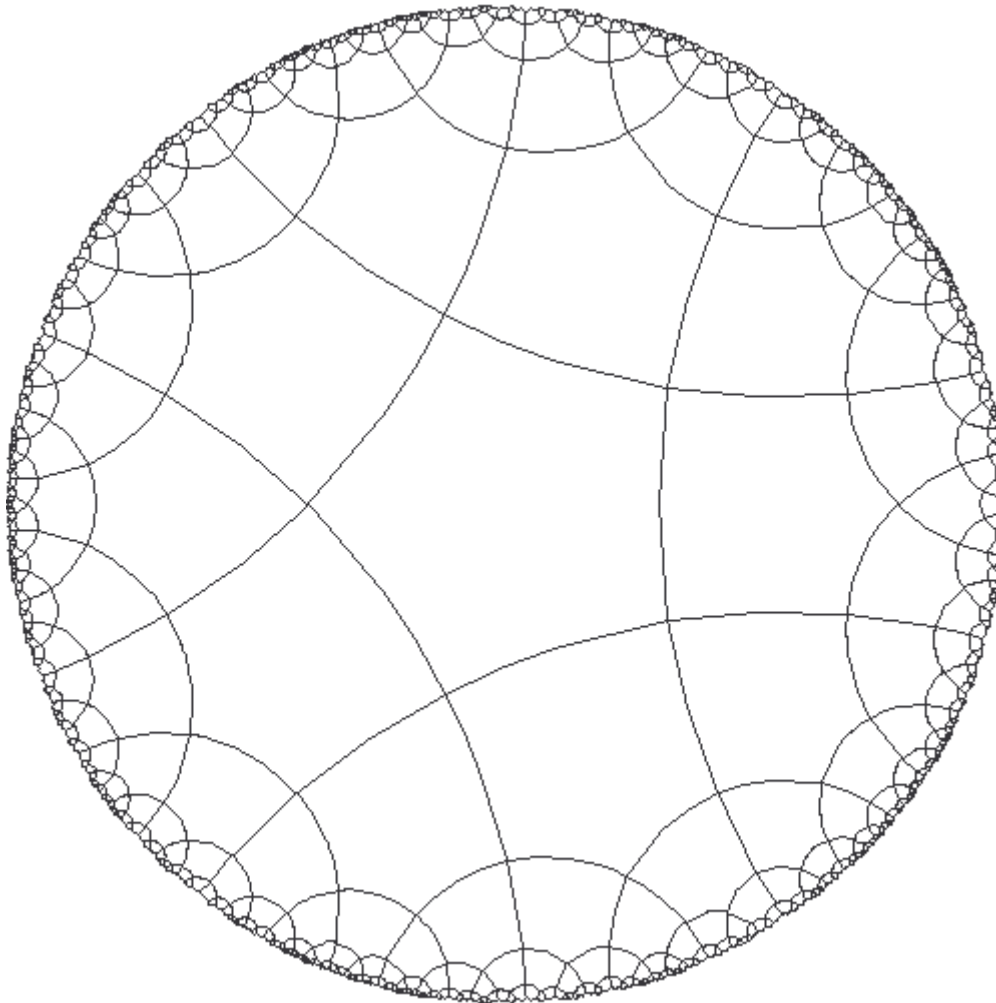
FiCell permet de :

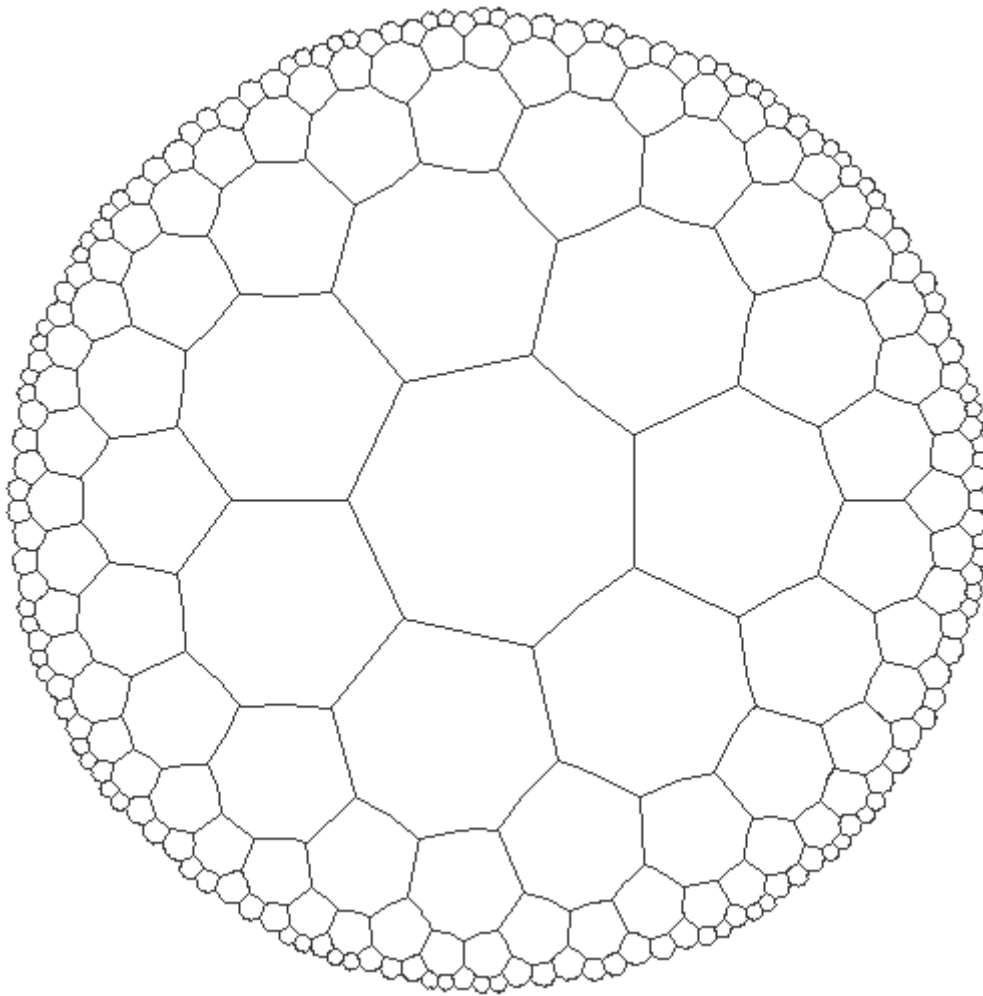
- sélectionner de grandes zones sans avoir à utiliser les scrollbars
- colorier les cellules en fonctions de leurs valeurs
- déplacer la colline avec le clavier ou la souris
- visualiser la colonne et la ligne grâce à deux fisheyes plus petites
- comparer deux colonnes ou deux lignes éloignées grâce à une seconde colline.
- changer la forme du focus (voir photos d'écran 5, 6 et 7)

IV. Les pavages dans un espace hyperbolique

B. Etude d'outils existants

I. Applet Java de réalisation d'un pavage dans un espace hyperbolique





Hyperbolic Tessellations

Some printable tilings

Regular tilings. These regular tilings were all created by the applet. They're drawn in black on white backgrounds so that they can be printed for later use.

The selection of available tilings is shown in the following table. The row indicates n , the number of sides on a polygon, while the column indicates k , the number of polygons that meet at each vertex.

Some entries in the table are colored. They indicate tilings that are not hyperbolic. There are five elliptic tilings that correspond to the five regular solids. They are: the tetrahedron $\{3,3\}$, the octahedron $\{3,4\}$, the icosahedron $\{3,5\}$, the cube $\{4,3\}$, and the dodecahedron $\{5,3\}$. There are also three parabolic tilings, which are tilings of the ordinary Euclidean plane. They are $\{3,6\}$ (tiling by equilateral triangles), $\{4,4\}$ (tiling by squares), and $\{6,3\}$ (tiling by hexagons).

Regular tilings								
	3	4	5	6	7	8	10	12

3					{3,7}	{3,8}	{3,10}	{3,12}
4			{4,5}	{4,6}	{4,7}	{4,8}	{4,10}	
5		{5,4}	{5,5}	{5,6}				
6		{6,4}	{6,5}	{6,6}				
7	{7,3}	{7,4}			{7,7}			
8	{8,3}	{8,4}						
10	{10,3}	{10,4}						
12	{12,3}							

Quasiregular tilings. There are also a few quasiregular tilings available. They're listed in the table below. The quasiregular tiling $\{n|k\}$ is made out of n -gons and k -gons. Each n -gon is bounded by k -gons, and each k -gon is bounded by n -gons. Two n -gons and two k -gons alternate around each vertex.

Note that the quasiregular tilings $\{n|k\}$ and $\{k|n\}$ are actually the same tiling of the hyperbolic plane. They appear different because a n -gon is placed in the center of the Poincaré disk for the first tiling, but a k -gon is placed in the center for the second tiling.

A quasiregular tiling $\{n|n\}$ is the same as the regular tiling $\{n|4\}$, and so the diagonal entries in the table below refer to tilings mentioned in the first table.

Quasiregular tilings								
	3	4	5	6	7	8	10	12
3					{3 7}	{3 8}	{3 10}	{3 12}
4			{4 5}	{4 6}				
5		{5 4}	{5 5}	{5 6}				
6		{6 4}	{6 5}	{6 6}				
7	{7 3}				{7 7}			
8	{8 3}					{8 8}		
10	{10 3}						{10 10}	
12	{12 3}							

[Create your own black & white tilings.](#)

[Return to the index.](#)

Dec 2002

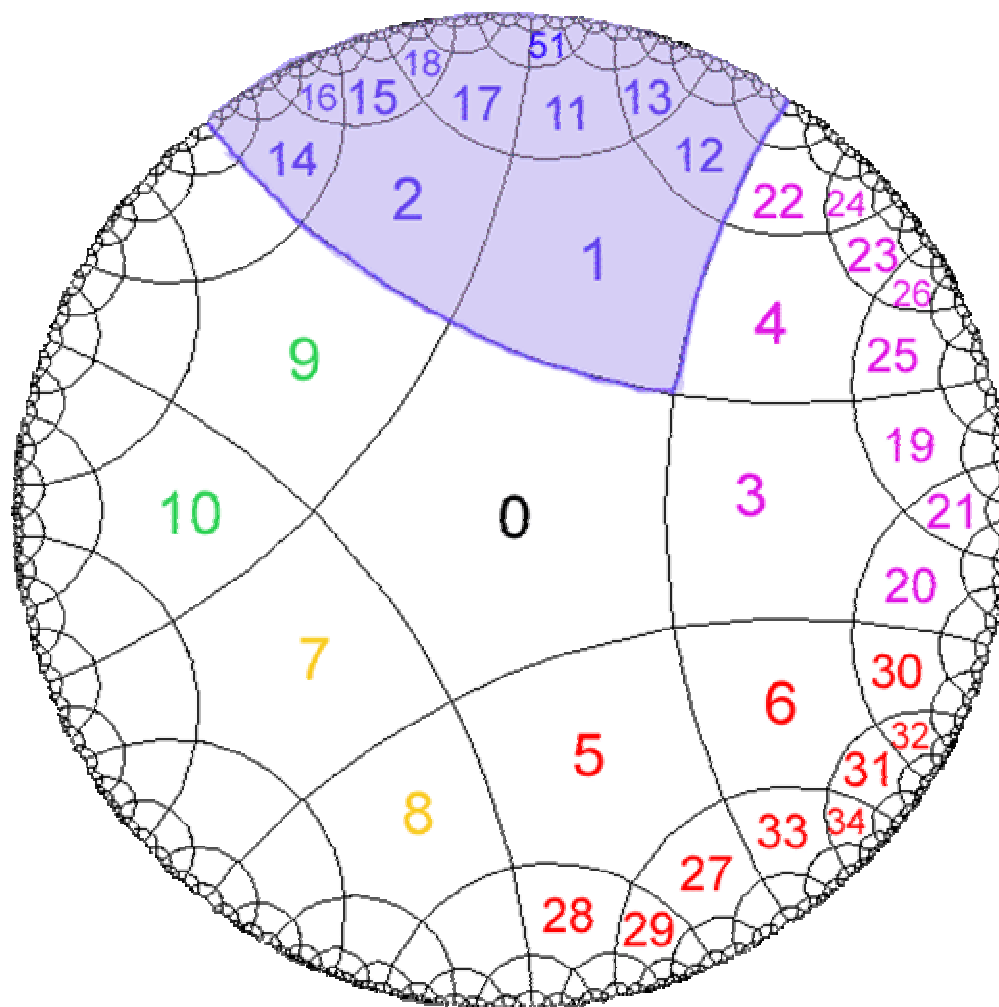
[David E. Joyce](#)

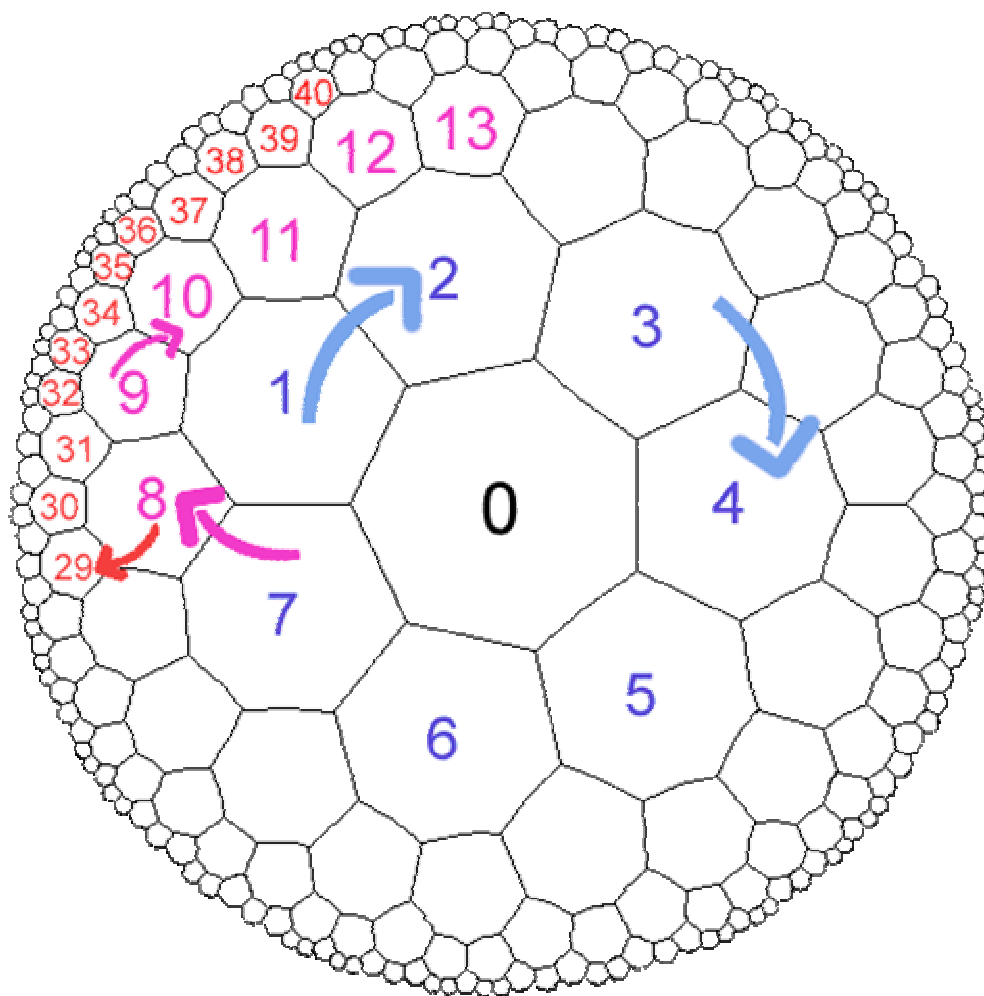
[Department of Mathematics and Computer Science](#)

[Clark University](#)

Worcester, MA 01610

The address of this file is <http://aleph0.clarku.edu/~djoyce/poincare/poincare.html>





II. Modeles pour le programmation Win32 et MFC

Etant totalement novices en programmation Windows au début de ce TER (maintenant ça va un peu mieux), nous avons commencé la réalisation pratique par une série d'études de codes récupérés sur Internet et qui présentaient certaines fonctionnalités que nous avons implémenté dans notre projet.

1. Bibliothèque MFC

Les exemples que nous avons trouvés sur Internet sont pour la plupart écrits avec la Bibliothèque de classes MFC (Microsoft Foundation Classes).

Les MFC constituent un ensemble de classes prédéfinies autour desquelles s'articule la programmation Windows avec Visual C++. L'écriture d'un programme Windows entraîne la création et l'utilisation d'objets MFC ou d'objets de classes dérivées des MFC. Les objets créés contiennent des fonctions membres permettant la communication avec Windows, l'échange de messages entre eux et le traitement des messages Windows.

Les MFC représentent un vaste domaine et font intervenir un grand nombre de classes. Elles composent une structure complète de développement d'application où il suffit d'effectuer des opérations de personnalisation dont on a besoin pour adapter nos programmes.

Toutes les classes MFC portent des noms qui commencent par C, comme CBrush ou CLabel ; toutes les données membres d'une classe MFC commencent par le préfixe m_.

Les classes MFC reposent intégralement sur l'interface Win32 de Windows qui est programmée en C. MFC est donc une simple abstraction, une couche objets ajoutée pour manipuler des structures Win32 classiques.

Nous avons commencé l'étude de code et la formation à la programmation Windows par MFC mais l'apprentissage de cette bibliothèque ne peut pas être fait en peu de temps et le développement demande une gymnastique d'esprit pour être en phase avec la méthode de programmation MFC. La lecture de codes Win32 nous a en comparaison paru beaucoup plus claire et compréhensible.

La surcouche objet MFC ne parvient pas non plus totalement à cacher ses origines Win32 et l'on se retrouve finalement à initialiser des structures type C plus qu'à programmer réellement en objet.

2. Outil loupe avec masque circulaire

L'outil loupe nous a été proposé par M. Martin comme base de travail au début de notre TER. C'est une loupe écran que l'on active en cliquant et que l'on peut balader à travers l'écran pour agrandir les éléments que l'on traverse.

Elle est programmée à partir de la bibliothèque MFC. Le principal intérêt de ce code, pour nous, résidait dans la gestion des fenêtres rondes (ou rectangulaire à bord transparent) et l'affichage de données dans cette fenêtre.

En fait, l'application "triche" en ne travaillant pas réellement avec le bureau mais fait tous ces calculs de dessins à partir d'une image (un imprim-écran). Lors du clic pour activer l'outil, une photo est prise et affichée et tout ce qui se passe en arrière plan n'est plus visible. Impossible par exemple de faire un zoom sur une séquence animée puisqu'une image fixe est toujours imposée.

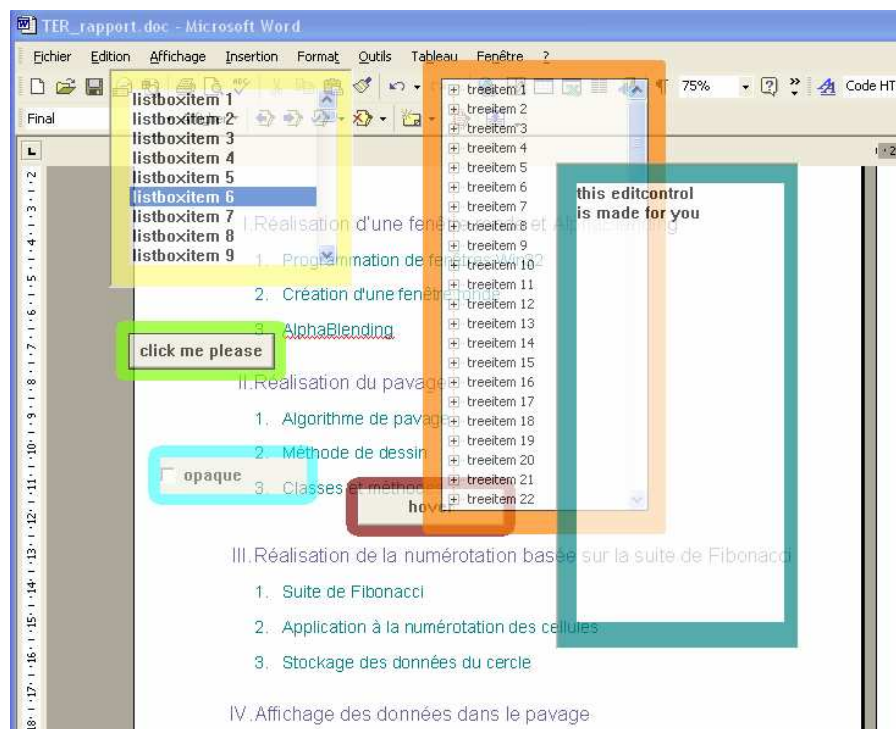
Pour nous, cette technique n'est pas envisageable car nous avons besoin de pouvoir interagir avec les fenêtres en arrière plan.

De plus, le code, particulièrement obscur pour des non habitués de la programmation MFC, n'a pas facilité la compréhension de l'ensemble.



3. Exemple avec de la semi transparence

Trouvé sur Internet, le code pellucid est un petit ensemble de classes permettant d'ajouter de l'alphablending à une boîte de dialogue et à ces composants.



Le code est composé de 3 classes à inclure dans le projet pour pouvoir être utilisées :

PellucidChild dérive du composant MFC CWnd en remplace les procédures de contrôles de la fenêtre principale par ses propres contrôles (Redéfinition du drag de la fenetre).

PellucidSelf dérive de la classe CFrameWnd. C'est la fenêtre en elle-même. Dériver de CFrameWnd permet d'être placée n'importe où sur l'écran.

Pellucid met en relation tous les signaux entre la fenêtre originale et PellucidChild et PellucidSelf. Elle génère aussi la transparence par un appel paramétré à Pellucid::CreateSelf()

La transparence s'effectue sans problème et même assez rapidement pour la plupart des widgets MFC. Nous avons testé l'utilisation avec CLabel, CEdit, CListBox ... les seuls widgets qui ne passent pas en transparence sont les objets du GDI, autrement dit les objet des dessins graphiques, les seuls dont nous ayons un besoin vital...

Après avoir quelque temps tenté d'adapter la librairie pour qu'elle supporte les objets du GDI (sans grand succes) , nous avons abandonné les Classes MFC et repris la programmation Windows en Win32.

C. Réalisation de l'interface

I. Réalisation d'une fenêtre ronde et AlphaBlending

1. Programmation de fenêtres Win32

a. API Win32

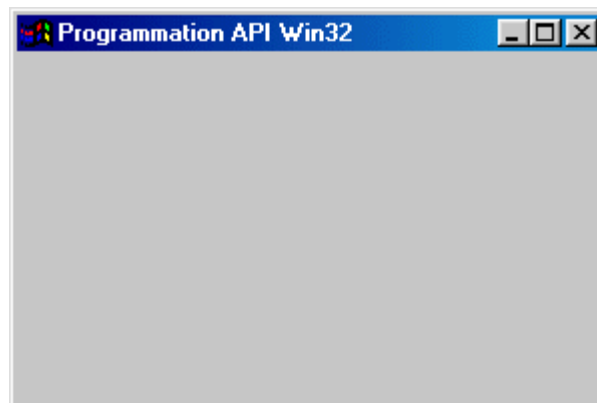
Windows 95, 98 NT et même les versions les plus récentes de l'OS de Microsoft sont toutes écrites avec le "même langage" si l'on peut s'exprimer ainsi. Toutes sont fondées sur l'API, mieux connues sous le nom de Win API. Ces trois versions 32 bits du système d'exploitation utilisent ce que l'on appelle Win32 API ou plus simplement Win32.

Win32 est une collection de plusieurs centaines de fonctions ; elle comporte un grand nombre de constantes, de macros, de structures, de types et d'autres éléments. Ces fonctions sont écrites en C mais on peut y faire appel depuis C++, l'assembleur, le Visual Basic. L'API Win32 est décrite dans la bibliothèque Windows.h

Les fonctions de Win32 API sont construites autour d'un certain nombre de concepts de base comme l'environnement graphique, les fenêtres, les menus, les icônes, les messages, les fichiers, les ressources ...

b. Fenêtres graphiques

Le concept fondamental est celui de fenêtre. De nombreuses fonctions de l'API concernent les fenêtres (c'est d'ailleurs le cas de tout environnement graphique) : création, dimensionnement, déplacement, etc. D'autres fonctions de l'API sont destinées à dessiner dans les fenêtres, gérer le placement des éléments dans ces fenêtres, etc.



Fenêtre basique

En programmation Win32, la base du programme est la fenêtre principale. Souvent celle-ci se nomme `hwnd` et est de type `HWND` (avec `visual c++`). Dans un programme Win32, la fenêtre se crée dans une fonction nommée `InitInstance()`. C'est dans celle-ci que l'on organise la taille, la couleur la forme, les onglets, tout le détail des fenêtres telles qu'elle doit apparaître lors de la première exécution

Exemple :

```
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
    HWND hWnd=CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
        CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);
```

```

//initialisation de la fenêtre
if (!hWnd){return FALSE;} // cas où l'on n'arrive pas à créer la fenêtre

ShowWindow(hWnd, nCmdShow); // affichage de fenêtre
UpdateWindow(hWnd); // update graphique des widgets
return TRUE;
}

```

Une autre fonction importante à écrire est `winProc()` qui sert à gérer les événements sur la fenêtre principale. C'est dans cette fonction que l'on va écrire le code à exécuter en cas d'événement souris ou clavier. Les événements sont des messages générés par Windows et transmis au programme concerné (le nôtre en fait). Ces messages sont définis comme des macros dans `windows.h`. Ils sont du type `WM_xxx`, par exemple `WM_QUIT`, `WM_CHAR`, etc.

Il est à noter aussi que les messages reçus par Windows sont ajoutés à une pile d'événements, sans distinction de priorité. `WM_QUIT` est traité comme un message de base.

Exemple :

```

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam,
                          LPARAM lParam){
    ...
    switch (message){
        case WM_COMMAND:
            ...
            break;
        case WM_PAINT:
            hdc = BeginPaint(hWnd, &ps);
            // Code pour peindre des éléments dans la fenêtre principale ici
            break;
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
        default:
            ...
    }
    return 0;
}

```

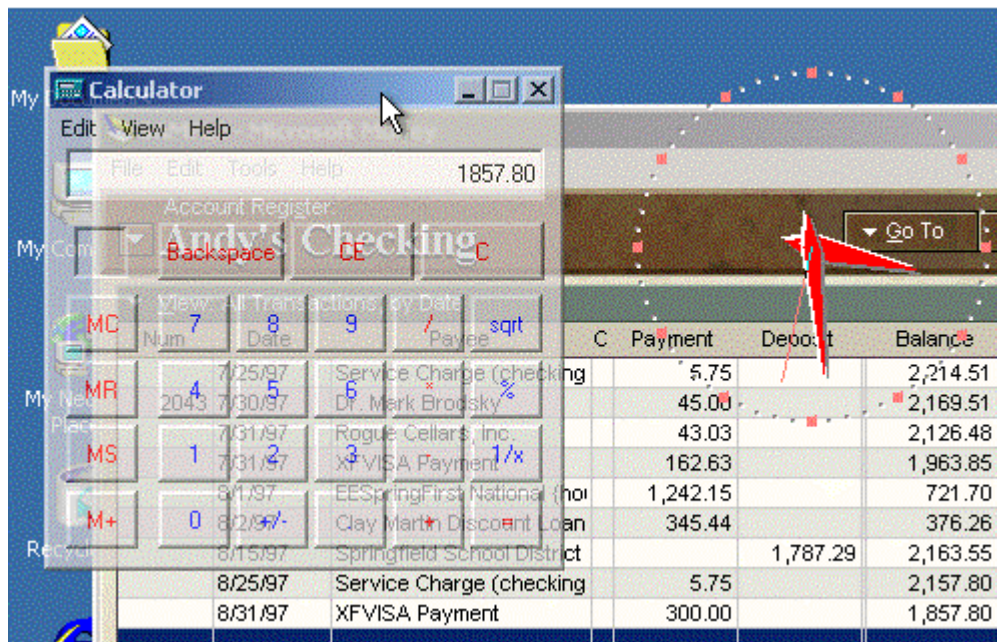
Ces fonctions sont appelées depuis la fonction principale, le point d'entrée du programme `Win32: WinMain()` (et non pas `main()` comme pour un programme C/C++ classique).

2. Création d'une fenêtre ronde

a. Layered windows

Il existe en programmation Win32 plusieurs types de fenêtres. `CreateWindow()` permet de créer des fenêtres pop-up, des fenêtres fils, une fenêtre principale. Elle permet de définir la géométrie de base (taille X et Y) ainsi que le style de la bordure, la présence ou non de barre de titre. Cela reste malgré tout assez classique et la fenêtre reste toujours un rectangle (plus ou moins personnalisable mais rectangle tout de même).

Depuis les versions 2000 de Windows, Win32 inclut un nouveau type de fenêtres, les fenêtres à style étendu (Layered Windows). Elles permettent d'ajouter des effets de transparence et de translucidité à des fenêtres principales. Windows 2000 a exploité ces capacités nouvelles dans les effets d'apparition et de transition du menu démarrer mais aussi dans l'ombre portée du curseur souris. C'est aussi la technique utilisée pour créer et animer les compagnons Office et Windows.



Calculatrice AlphaBlendée et horloge transparente

Il était possible de créer tous ces effets déjà avec les versions antérieures de Win32 mais les fonctions devaient être codées à la main et faire une transparence de zone consistait à calculer et appliquer un masque sur la zone en transparence et recopier celle-ci dans le contexte graphique écran de l'application. Opération fastidieuse et assez délicate à mettre en œuvre. Pour autant les Layered Windows ne sont qu'une abstraction permettant de manipuler ces fonctions beaucoup plus simplement.

Les fenêtres à style étendu consomment beaucoup de ressources système, c'est principalement cela qui induit la "lourdeur" souvent reprochée à Windows XP.

b. Création de la fenêtre ronde

La création de la fenêtre ronde s'effectue en trois étapes. Tout d'abord, on crée une "Layered Windows" dans la fonction d'initialisation de la fenêtre `InitInstance()` :

```

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
    HWND hWnd;
    hInst = hInstance; // Store instance handle in our global variable

    //creation de la fenetre
    hWnd = CreateWindowEx(WS_EX_LAYERED,szWindowClass, szTitle, WS_POPUP,
        CW_USEDEFAULT, 0, 300, 300, NULL, NULL, hInstance, NULL);

    if (!hWnd){return FALSE;}

    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);

    return TRUE;
}

```

On détermine ici aussi, comme pour une fenêtre classique le style de la forme. Ici, une fenêtre sans bordure ni barre de tâche ou de titre.

Ensuite, il s'agit de lui indiquer qu'elle doit prendre une forme ronde. Cela est fait lors de la création effective de la fenêtre au moment où Windows reçoit l'évènement `WM_CREATE` dans la fonction de gestion des événements `WndProc()` :

```

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam,
                          LPARAM lParam){
    int wmId, wmEvent;
    PAINTSTRUCT ps;
    HPEN hPen;
    HBRUSH hBrush;
    LOGBRUSH hLogBrush;
    HRGN hRgn;
    HDC hdc;
    RECT rt;

    switch (message)
    {
        case WM_COMMAND:
            ...
            break;
        case WM_CREATE :
            GetClientRect(hWnd,&rt);
            hRgn=CreateEllipticRgn(rt.left,rt.top,rt.right,rt.bottom);
            SetWindowRgn(hWnd,hRgn,true);
            break;
        ...
    }
    ...
}

```

Ici, on dit à l'application que sa fenêtre principale doit être de forme elliptique et doit prendre toute la taille du rectangle réservé pour la fenêtre lors de l'étape1. Il est possible de donner n'importe quelle forme à une fenêtre en utilisant une shape et la fonction SetWindowRgn().

Troisième et dernière étape, il faut maintenant colorier cette fenêtre. Cela s'écrit dans le code lié à l'évènement WM_PAINT, toujours dans la fonction WinProc :

```

case WM_PAINT:
    hdc = BeginPaint(hWnd, &ps);
    // TODO: Add any drawing code here...
    RECT rt;

    hPen=CreatePen(PS_SOLID,0,RGB(0,0,0));

    hLogBrush.lbStyle=BS_HOLLOW;
    hBrush=CreateBrushIndirect(& hLogBrush);

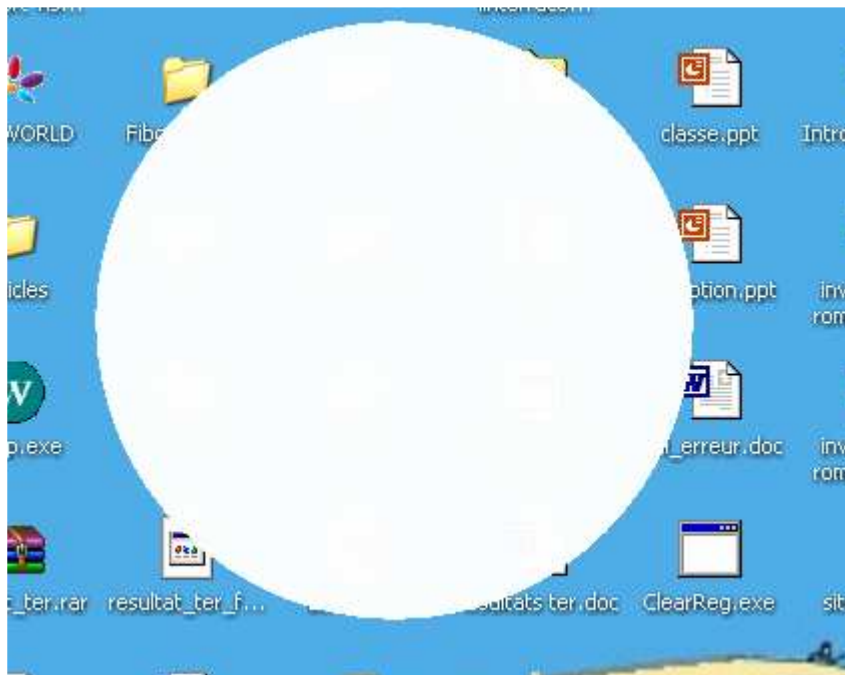
    SelectObject(hdc,hPen);
    SelectObject(hdc,hBrush);

    GetClientRect(hWnd,&rt);
    EndPaint(hWnd, &ps);
    break;

```

Pour cette fenêtre blanche, on associe simplement un crayon par défaut de cette couleur (hPen) avec CreatePen() à la fenêtre (représentée par son contexte graphique hdc) avec la fonction SelectObject().

Le résultat de ce programme est illustré en page suivante.



Fenêtre ronde

3. AlphaBlending

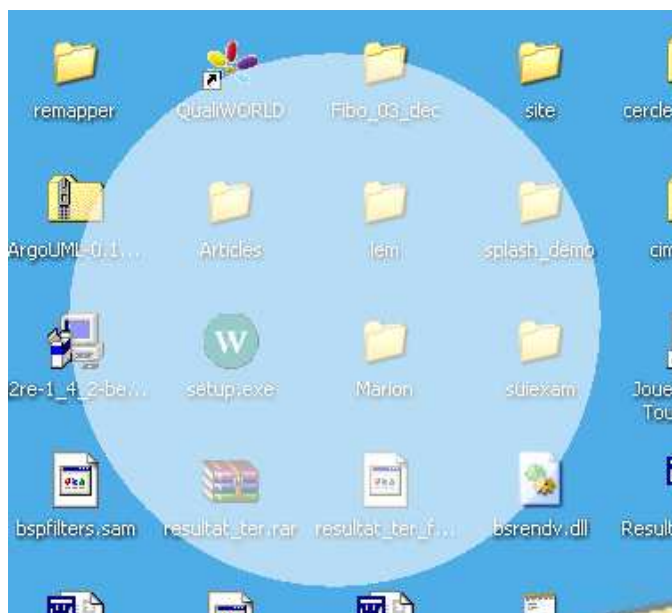
Une fois que l'on a trouvé le principe des Layered Windows, l'alpha blending n'est plus une chose très compliquée à mettre en place. Toutefois, cette option n'est disponible à la programmation qu'en utilisant le dernier SDK de chez Microsoft ou alors en programmant avec Visual studio .NET.

Il suffit en fait de définir une propriété particulière pour la fenêtre avec `SetLayeredWindowAttributes()` lors de la création de la fenêtre dans `InitInstance()`. C'est écrit en une ligne :

```
SetLayeredWindowAttributes(hWnd,NULL,150,LWA_ALPHA);
```

On passe la fenêtre, la valeur de l'alphaBlend (entre 0 et 255) et enfin le mode à appliquer à la fenêtre (ici `LWA_ALPHA` pour la semi-transparence).

Avec `SetLayeredWindowAttributes()`, il est aussi possible de définir une couleur de transparence, c'est le second paramètre que nous n'utilisons pas dans notre cas.



L'alphaBlending sur des grandes zones est très consommateur en ressources (c'est pas macOS), Microsoft lui même déconseille son utilisation abusive. Au delà d'une fenêtre de 400*400, l'alphaBlend n'est plus assez rapide pour permettre une bonne maniabilité de l'outil. Cette limitation nous a amené à considérer d'autres solutions pour rendre l'interface efficace, comme par exemple des petites zones de transparence par cellule plutôt que sur tout l'outil. L'alphaBlend est aussi nuisible à la représentation des couleurs. L'avantage de la méthode est que l'on voit avec quoi l'on interagit. Une solution permettant de combiner à la fois exigences techniques et contraintes d'interfaces serait de ne faire de l'alphablending que lors des déplacement de l'outils, pour voir où l'on se positionne et une fois le positionnement effectué, revenir en mode sans transparence pour une meilleure représentation des couleurs.

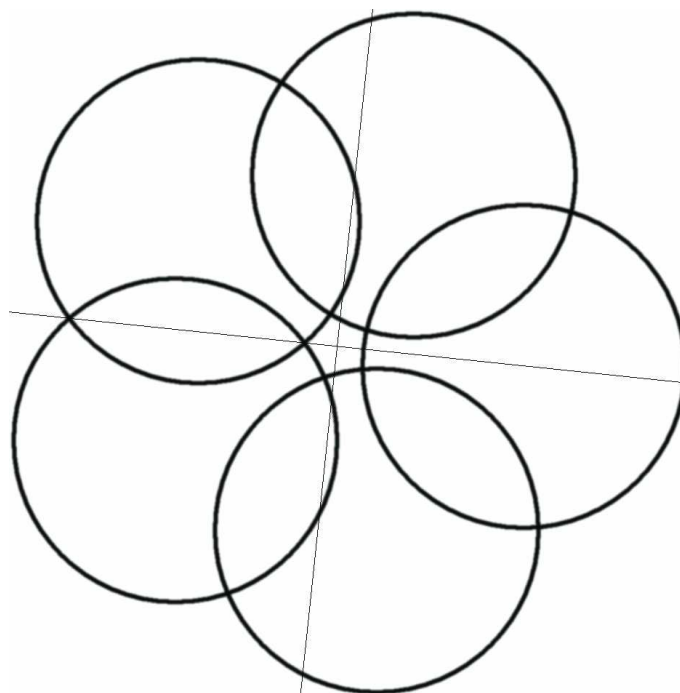
II. Réalisation du pavage

1. Algorithme de pavage

a. Technique utilisée pour la représentation d'un pentagone

Précision utile : dans le cas de notre outil, ce que nous appelons pentagone à un sens dans la géométrie hyperbolique, sa représentation en géométrie plane (euclidienne) est une figure à 5 cotés, ses cotés étant formés par des arcs de cercles incurvés vers l'intérieur. Les cotés des pentagones n'ont pas forcément les mêmes longueurs dans la représentation euclidienne. Dans le cercle hyperbolique, seul le pentagone central (ou pentagone principal) a ses 5 arcs de cercles de même longueur.

Pour composer un pentagone, on va chercher à dessiner les 5 cercles qui composent ses cotés.



Dessin du pentagone principal

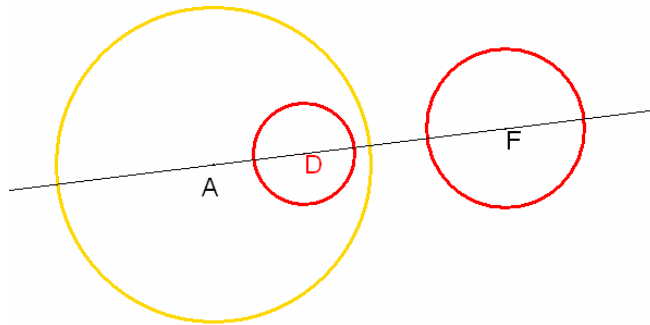
Le pentagone est le résultat du croisement des 5 cercles.

Il suffit donc de connaître les coordonnées des cercles pour construire les pentagones. Tout l'algorithme est basé sur la définition des pentagones comme ensemble de 5 cercles.

b. Symétrie par rapport aux cercles

Le pavage est réalisé à partir de construction de symétrie par rapport aux cercles composants les pentagones. En fait, toute la figure peut être obtenue en appliquant récursivement des règles de construction très simples à partir du dessin du pentagone principal. Une symétrie de cercle est une construction mathématique particulière permettant de définir l'image d'un cercle à partir d'un

autre. En fait en géométrie hyperbolique, cela revient à construire une symétrie axiale classique puisque nos cercles de papier y sont en fait des droites. En d'autres termes, une symétrie de cercles est la représentation euclidienne d'une symétrie hyperbolique.
Illustration :



Le symétrique du cercle F par rapport au cercle A est le cercle C. Pour la partie mathématique, voici les formules permettant de calculer les coordonnées du résultat d'une symétrie :

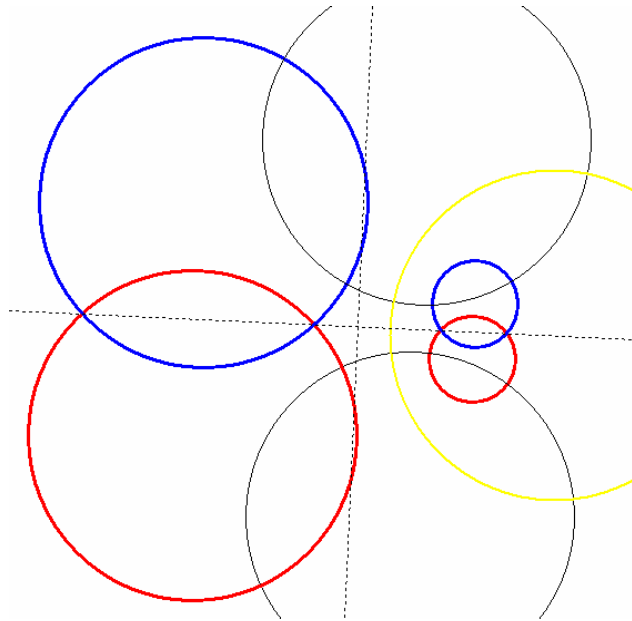
$$x_D = x_A + K^*(x_F - x_A)$$

$$y_D = y_A + K^*(y_F - y_A)$$

$$r_D = K^*r_F$$

Où (x,y) sont les coordonnées des centres des cercles et r les rayons respectifs

$$K = r_A^2 / ((x_F - x_A)^2 + (y_F - y_A)^2 - r_F^2)$$



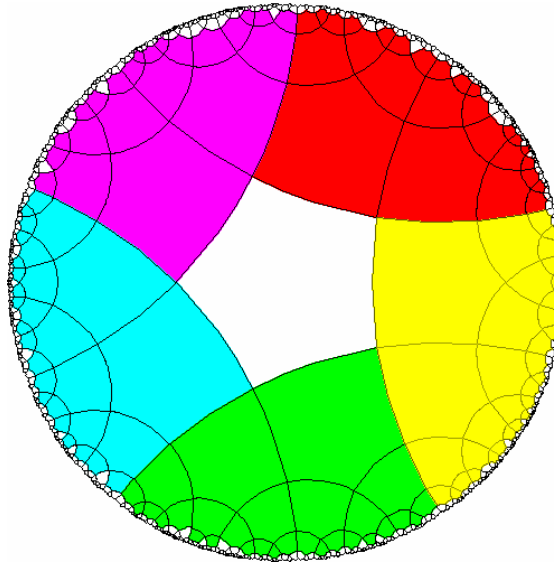
Les symétries de cercles permettent de représenter les pentagones descendant du pentagone principal.

Ici, les symétries des cercles bleus et rouges par rapport au cercle jaune permettent de construire l'image d'un second pentagone, descendant du pentagone principal.

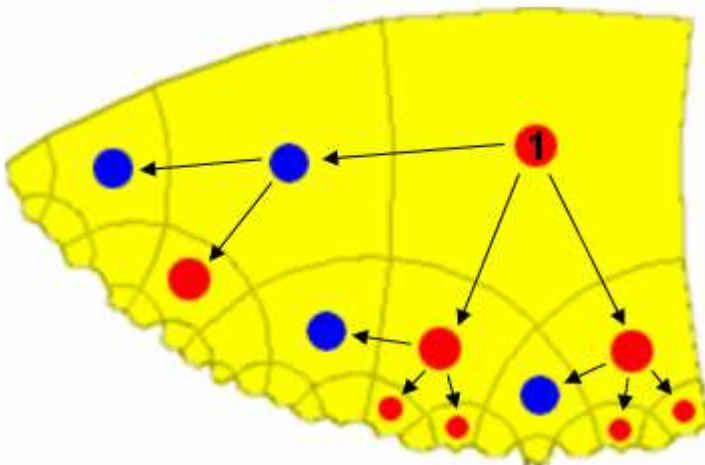
On peut noter qu'il n'est nécessaire d'effectuer que deux symétries pour déduire tous les cotés du pentagone. On calcule la symétrie à partir des cercles opposés au cercle de symétrie, les cercles adjacents constituent déjà des côtés du pentagone.

c. Découpages en du cercle en zones

Le cercle hyperbolique peut être découpé en 5 zones toutes semblables où l'on peut appliquer l'algorithme récursif de dessin



Découpage du cercle en 5 zones.



Les zones sont constituées d'un ensemble de pentagones dérivant chacun du premier pentagone de la zone (noté 1) lui-même descendant du pentagone principal du cercle hyperbolique.

Pour dessiner toute la zone sans doublons ni oubli, chaque pentagone doit avoir soit un fils soit deux fils (aussi appelés nœuds).

Le pentagone 1 de la zone a 3 fils, on répartit ensuite les 2 fils et 3 fils suivant l'illustration de droite (bleu = 2 nœuds, rouge = 3 nœuds).

On constate que le premier fils d'une zone (celui de gauche est systématiquement un deux nœuds.

Les autres fils sont des trois nœuds.

Si on génère à partir d'un deux nœuds, alors il n'y a que deux pentagones à dessiner, dans le cas d'un trois nœuds, on dessinera trois pentagones.

d. Exécution de l'algorithme

En premier lieu, il s'agit de construire et stocker notre pentagone principal, puisque tout découle de lui. Les coordonnées des cercles porteurs ont été préalablement calculées, nous n'avons plus qu'à nous servir des formules pour générer les cercles.

Le rayon vaut

$$\omega = \sqrt{1/\cos 2\pi/5}$$

Les coordonnées pour un cercle hyperbolique unité sont :

$$c1 = [\sqrt{1/\cos 2\pi/5}, 0]$$

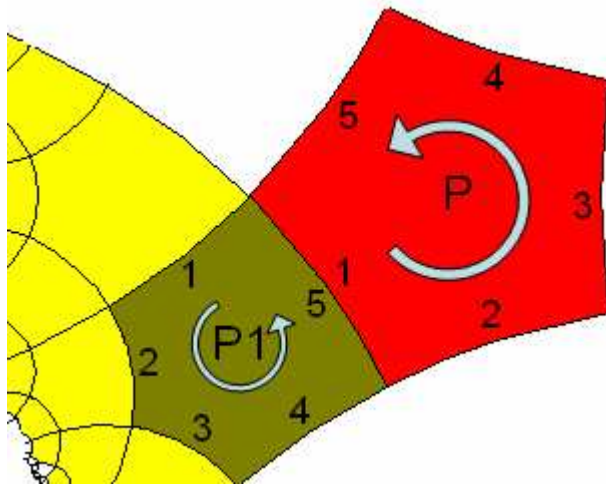
$$c2 = [\sqrt{1/\cos 2\pi/5} * \cos 2\pi/5, \sqrt{1/\cos 2\pi/5} * \sin 2\pi/5]$$

$$c3 = [\sqrt{1/\cos 2\pi/5} * \cos 4\pi/5, \sqrt{1/\cos 2\pi/5} * \sin 4\pi/5]$$

$$c4 = [\sqrt{1/\cos 2\pi/5} * \cos 4\pi/5, -\sqrt{1/\cos 2\pi/5} * \sin 4\pi/5]$$

$$c5 = [\sqrt{1/\cos 2\pi/5} * \cos 2\pi/5, -\sqrt{1/\cos 2\pi/5} * \sin 2\pi/5]$$

A partir de ce pentagone principal, nous allons générer les premiers pentagones de chaque zone. L'algorithme se déroule sur une zone, il est juste reproduit 5 fois pour générer l'ensemble des zones.



On base la construction du pentagone fils sur une symétrie suivant l'un des cotés du pentagone père. Pour la zone en exemple, le pentagone fils P1 est la symétrie du père P (le pentagone Main) suivant le côté 1 de P.

Le côté 1 de P est en fait le premier cercle porteur du pentagone.

Terminologie : le côté d'un pentagone est une HLine, ainsi, le côté un du pentagone P s'abrège de cette façon : H^P_1 , le côté 4 de P1 est H^{P1}_4 .

Définissons les cotés du pentagone fils par rapport au père :

$$\begin{aligned} H^{P1}_1 &= H^P_5 \\ H^{P1}_2 &= \text{symétrie de } H^P_4 \text{ par rapport à } H^P_1 \\ H^{P1}_3 &= \text{symétrie de } H^P_3 \text{ par rapport à } H^P_1 \\ H^{P1}_4 &= H^P_2 \\ H^{P1}_5 &= H^P_1 \end{aligned}$$

Il n'y a que deux symétries à réaliser, elles se font toujours pour générer les cotés 2 et 3 du fils, les autres cotés sont déjà dessinés et n'ont pas à être générés plusieurs fois.

Une fois que l'on a le premier pentagone de la zone, on entre dans l'algorithme récursif de pavage. Chaque boucle récursive va dessiner une génération ou un niveau dans l'arbre des pentagones.

On distingue 2 cas lors du dessin : soit on est en présence d'un pentagone de type 3 nœuds dont on doit générer les fils, soit il s'agit d'un deux nœuds. Le premier pentagone est toujours un 3 nœuds.

Algorithme général :

Si P est un 3 nœuds faire

Dessiner ses pentagones fils suivant les cotés H^P_1 , H^P_2 et H^P_3

Le côté symétrique de H^P_1 est un deux nœud

Le côté symétrique de H^P_2 est un trois nœud

Le côté symétrique de H^P_3 est un trois nœud

Sinon faire //P est un deux nœuds

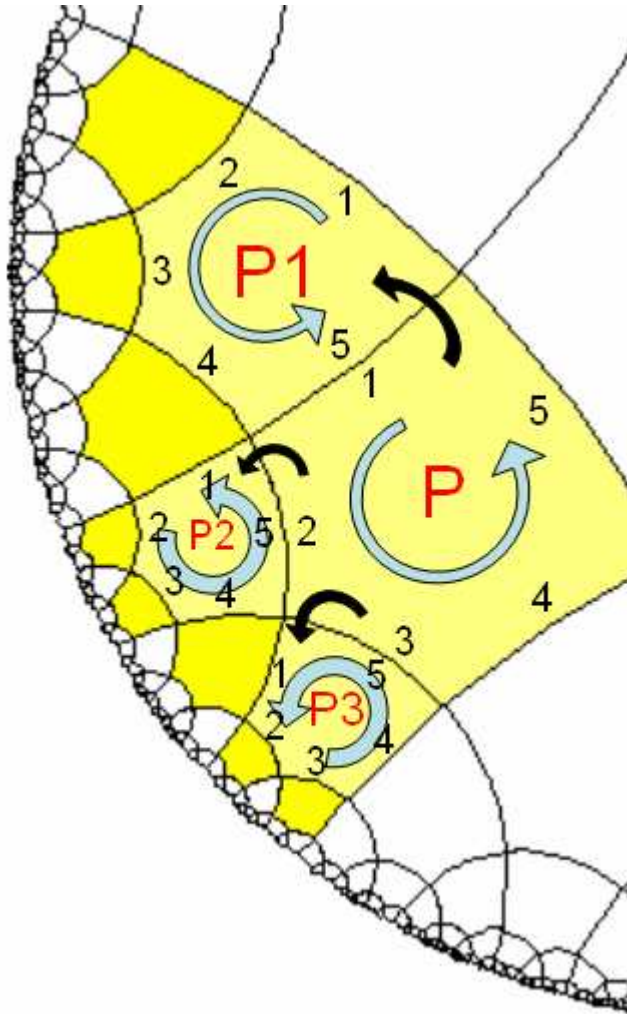
Dessiner ses pentagones fils suivant les cotés H^P_1 , H^P_2

Le côté symétrique de H^P_1 est un deux nœud

Le côté symétrique de H^P_2 est un trois nœud

Fin si

Lancer la génération suivante.

Déroulement de l'algorithme pour les fils 1, 2 et 3 du premier pentagone de la zone :

On commence par dessiner les trois fils en utilisant les symétries

HLines de P1:

$$H^{P1}_1 = H^P_5$$

$$H^{P1}_2 = \text{symétrie de } H^P_4 \text{ par rapport à } H^P_1$$

$$H^{P1}_3 = \text{symétrie de } H^P_3 \text{ par rapport à } H^P_1$$

$$H^{P1}_4 = H^P_2$$

$$H^{P1}_5 = H^P_1$$

Hlines de P2

$$H^{P1}_1 = H^P_1$$

$$H^{P1}_2 = \text{symétrie de } H^P_5 \text{ par rapport à } H^P_2$$

$$H^{P1}_3 = \text{symétrie de } H^P_4 \text{ par rapport à } H^P_2$$

$$H^{P1}_4 = H^P_3$$

$$H^{P1}_5 = H^P_2$$

Hlines de P3

$$H^{P1}_1 = H^P_2$$

$$H^{P1}_2 = \text{symétrie de } H^P_1 \text{ par rapport à } H^P_3$$

$$H^{P1}_3 = \text{symétrie de } H^P_5 \text{ par rapport à } H^P_3$$

$$H^{P1}_4 = H^P_4$$

$$H^{P1}_5 = H^P_3$$

Ensuite, comme P est un 3 nœuds, les fils sont désignés :

$$P1 \Rightarrow 2 \text{ nœuds}$$

$$P2 \Rightarrow 3 \text{ nœuds}$$

$$P3 \Rightarrow 3 \text{ nœuds}$$

Il est important de repérer les correspondances entre les HLines générées et les HLines génératrices ; Pour **P2** on génère les cotés à partir de **1,5,4,3,2** Pour **P3** on génère à partir de **2,1,5,4,3**.

e. Règles de constructions (abstraction) :

ATTENTION, on prend pour hypothèse que 1-1=5, 1-2=4 ... On ne passe pas à 0, on reboucle sur le dernier coté.

Le premier H^{PX}_1 du pentagone fils PX est la HLine du pentagone P indicé H^P_{X-1}

Pour les autres pentagones, l'indice des cotés HLines fils est incrémenté (on génère 1,2,3,4 et 5 dans cet ordre) et les indices des HLines génératrices du pentagone père sont décrémentés

$$H^{PX}_2 = \text{symétrique de } H^P_{X-2} \text{ par rapport à } H^P_X$$

$$H^{PX}_3 = \text{symétrique de } H^P_{X-3} \text{ par rapport à } H^P_X$$

$$H^{PX}_4 = H^P_{X-4}$$

$$H^{PX}_5 = H^P_{X-5}$$

2. Méthode de dessin

Après avoir été préalablement intégralement calculé et enregistré en mémoire (grâce à la hiérarchie d'objets mise en place), le programme va pouvoir demander un affichage écran du cercle.

a. Win32 et le GDI

GDI (Graphics Device Interface) est une interface de périphérique graphique. C'est lui qui permet l'affichage de tout ce qui est graphique à l'écran et sur imprimante (ou traceur). Chaque fois qu'une application Win32 dessine sur l'écran elle doit utiliser le GDI. Le GDI fournit des fonctions

pour tracer des points, des traits, des rectangles, des polygones, des ellipses, des bitmaps et du texte.

Lorsque l'on dessine à l'écran, on utilise un périphérique matériel. Dans les programmes Win32, pour pouvoir utiliser un périphérique matériel, on doit demander un handle de contexte pour ce périphérique. Il contient des attributs généraux utilisés par les objets du GDI définissant par exemple la couleur du texte, la taille des pinceaux pour le dessin. A chaque fois, des fonctions permettent de changer les attributs définis dans le contexte de périphérique.

Il existe deux solutions pour récupérer le contexte de périphérique.

Première méthode :

Nous avons vu que la fenêtre se dessinait sur l'appel WM_PAINT, nous pouvons récupérer le contexte au moment de cet appel et dessiner les objets en même temps :

```
case WM_PAINT:
    HDC hdc;
    PAINTSTRUCT ps;
    hdc=BeginPaint(hwnd,&ps); // obtient un handle de contexte de périphérique
    // Affichage de bitmap, pixel...
    EndPaint(hwnd,&ps); // libère le handle de contexte de périphérique
    return 0;
```

hdc de type HDC est le contexte de périphérique (Handle Device Context)

Ps, structure PAINTSTRUCT stocke les informations générales du dessin, il est défini par défaut par Windows, nous n'avons pas besoin de changer les valeurs des champs. PAINTSTRUCT doit être modifié si l'on veut redéfinir la taille de la zone à dessiner

Après avoir obtenu un handle de contexte de périphérique avec **BeginPaint**, il ne faut pas oublier de le libérer immédiatement après avec **EndPaint** avant de quitter la procédure de fenêtre. Ces deux fonctions prennent en argument l'handle de la fenêtre et une structure **PAINTSTRUCT**. **EndPaint** retourne un **BOOL** pour indiquer une réussite ou un échec.

Tout le code permettant d'afficher tous les objets graphiques de votre fenêtre doit être placé entre **BeginPaint** et **EndPaint** pour pouvoir être redessiné par Windows.

Deuxième méthode :

Elle utilise deux fonctions :

```
hdc=GetDC(hwnd); // obtient un handle de contexte de périphérique
... // affichage de bitmap, pixel...
ReleaseDC(hwnd,hdc); // libère l'handle
```

Ces fonctions peuvent être appelées presque n'importe où dans notre programme (après que la fenêtre ait été créée) et n'ont pas besoin d'être utilisées avec un message WM_PAINT. Mais dans ce cas si on a écrit un texte ou affiché une image au début de notre programme et si la fenêtre est redessinée par Windows (Ex: à cause d'un chevauchement), les objets graphiques seront effacés. Il faut donc trouver un moyen de redessiner les graphismes en rappelant ces fonctions (Une solution courante est de les rappeler lors du WM_PAINT).

GetDC et ReleaseDC reçoivent en argument l'handle d'une fenêtre. La seconde fonction nécessite aussi l'handle de contexte de périphérique.

D'autres variantes comme GetWindowDC accède à l'intégralité de la fenêtre, même les barres de titres, contrairement à GetDC qui n'accède qu'à la zone cliente. CreateDC est aussi utilisable pour dessiner sur tout l'écran (même en dehors de votre fenêtre).

b. Dessiner des cercles

L'ensemble du disque est constitué par des cercles. Ils sont dessinés les uns après les autres suivant l'ordre de descente dans l'algorithme de pavage. Ils doivent être en transparence, c'est-à-dire peint avec un pinceau transparent, pour éviter que les HLines de plus grand diamètre ne recouvre visuellement les plus petites.



Le disque avec des cercles non transparents : mauvais

Un cercle se dessine en utilisant la fonction ellipse() de Win32

```

BOOL Ellipse(
    HDC hdc,      // handle to DC
    int nLeftRect, // x-coord du coin haut droit du rectangle
    int nTopRect,  // y-coord du coin haut gauche du rectangle
    int nRightRect, // x-coord du coin bas droit du rectangle
    int nBottomRect // y-coord du coin bas gauche du rectangle
);
    
```

Ellipse() dessine dans le contexte hdc une ellipse contenue dans le rectangle défini par les 4 coordonnées.

Il faut définir quelques variables de GDI pour pouvoir peindre en transparence. Le pinceau à utiliser est défini par la structure LogBrush :

```

typedef struct tag LOGBRUSH {
    UINT    Style;
    COLORREF Couleur;
    LONG    Motif;
} LOGBRUSH, *PLOGBRUSH;
    
```

On peut changer le style, la couleur et le motif de remplissage du pinceau. Dans notre cas, il suffit de définir le style à BS_HOLLOW pour dessiner en transparence :

```

LOGBRUSH hLogBrush;
hLogBrush.lbStyle=BS_HOLLOW;
    
```

Ensuite nous n'avons plus qu'à assigner ce pinceau au GDI avec la fonction CreateBrushIndirect()

```

HBRUSH hBrush;
hBrush=CreateBrushIndirect(& hLogBrush);
    
```

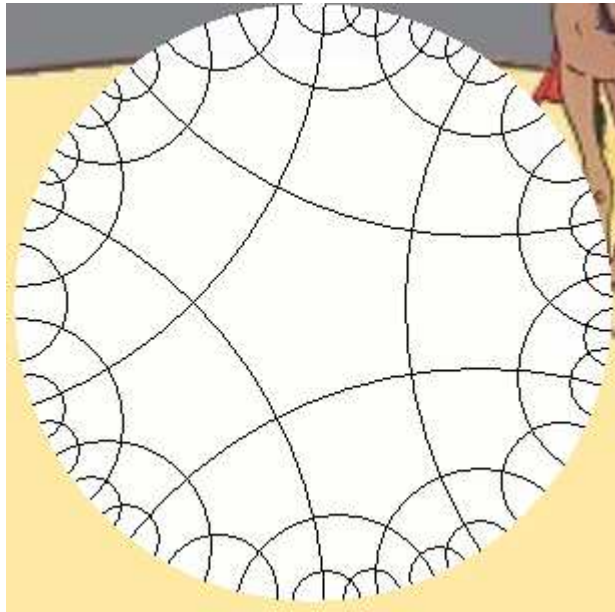
Les paramètres de dessin étant définis, on remonte la hiérarchie des objets en passant en paramètre le contexte HDC pour arriver à la méthode draw de Circle.

```

void Circle::draw(const HDC & hdc,int facteur)
{
    Point p1=Point(centre.x+rayon,centre.y+rayon);
    Point p2=Point(centre.x-rayon,centre.y-rayon);

    Ellipse(hdc,p1.x*facteur+facteur,p1.y*facteur+facteur,p2.x*facteur+facteur,
    p2.y*facteur+facteur);
    return;
}
    
```

Voir *Classes et méthodes* pour le membre rayon. Facteur est le facteur d'agrandissement de la figure. Avant l'affichage, le disque est calculé pour être affiché dans un cercle unité. Pour avoir quelque chose de visible à l'écran, il faut multiplier la taille de l'ensemble de la figure par facteur.



Une fois le pavage effectué

3. Classes et méthodes

Un disque hyperbolique n'est pas simplement un très gros tableau de coordonnées de cercles, nous sommes passés par une structure objet pour définir un niveau acceptable d'abstraction permettant de comprendre le système simplement.

Nous détaillerons les classes servant à l'affichage et au calcul du cercle en partant du plus haut niveau (classe DisquePoincare) vers le plus bas niveau des classes d'objets graphiques (classe Point).

ATTENTION : Les classes présentées sont simplifiées par rapport au code pour ne présenter que ce qui est utile à l'affichage du disque.

Classe DisquePoincare

Est une classe abstraite permettant de définir un disque de Poincaré. Un disque de Poincaré est un ensemble de pentagones.

```
class disquePoincare
{
    private :
        unsigned int m_hauteur; //profondeur du pavage
        noeud *m_mainPentagon;
        Penta **disquePoincareDraw;
        void paverRecuratif( int, const Penta &, noeud *, int);
        ...
    Public :
        void paver(unsigned int hauteur);
        void toscreen(const HDC &,int);
        ...
}
```

disquePoincareDraw est un tableau à deux dimensions contenant tous les pentagones (type Penta) du disque.

```
[0][0]      ->pentagone main
[1][1->size] ->region1
[2][1->size] ->region2
...
```

Les pentagones sont indexés dans ce tableau suivant l'ordre des codes de Fibonacci (cf section suivante : "Réalisation de la numérotation basée sur la suite de Fibonacci")

m_mainPentagon est le nœud principal du disque (le pentagone Main) Il est de type abstrait noeud.

paverRecuratif() est la fonction de calcul et enregistrement des pentagones dans la structure *disquePoincareDraw* On lui donne en argument le pentagone principal de la zone que l'on veut dessiner. Ce pentagone est donné par *m_mainPentagon->Fils(X)* ou X est entre 1 et 5 le numéro d'index de la zone à dessiner.

Les méthodes publiques *paver()* et *toscreen()* sont à appeler respectivement pour enregistrer le pavage en mémoire et dessiner dans le contexte graphique le disque enregistré.

Paver() ne prend pas d'argument, c'est la première méthode à appeler juste après la construction d'un pentagone. Cette méthode va construire et enregistrer le pentagone main dans *disquePoincareDraw[0][0]* et lancer la récursion des zones en appelant 5 fois la méthode privée *paverRecuratif()*.

Toscreen() prend en argument un contexte graphique HDC et un facteur d'agrandissement pour le disque. La méthode va appeler la méthode *draw()* de chacun des pentagones enregistrés dans *disquePoincareDraw[][]*.

Classe Penta

Classe représentant les pentagones en terme de géométrie et représentation dans l'espace. Un penta stocke les cercles porteurs et les points remarquables des pentagones (centre, sommets)

Un pentagone est un ensemble de cercle, un point centre et 5 sommets.

```
class Penta
{
    public:
        Penta(Circle,Circle,Circle,Circle,Circle,Point);
        Penta(Penta,int,int);
        void setType(int tp) {type=tp;};
        int getType()        {return type;};
        void draw(const HDC &,int);
        ...
    private:
        int type;
        Point Centre(Point,Circle);
        Circle Symetrie(Circle,Circle);
        Circle h[5]; //cotés de 0 à 4
        Point sommet[5];
        Point center;
};
```

Le premier constructeur est appelé par *DisquePoincare::Paver()* lors de la construction du main Pentagone. C'est le seul pentagone où l'on a besoin de donner tous ces cotés, tous les autres sont des symétries que l'on construit avec le second constructeur qui prend pour paramètre le pentagone père, le coté de symétrie du pentagone Père (son numéro) et le type du pentagone à générer (2 ou 3 nœuds). Le constructeur utilise deux appels à la méthode privée *Symetrie()* pour générer les HLines 2 et 3 et fait simplement des recopies par rapport aux cotés du père pour les autres HLines :

```
h[1]=Symetrie(pere.GetH(cible[1]),pere.GetH(cote));
h[2]=Symetrie(pere.GetH(cible[2]),pere.GetH(cote));
temp=pere.GetH(cible[0]);
h[0]=Circle(temp.GetCenter(),temp.GetRayon());
temp=pere.GetH(cible[3]);
h[3]=Circle(temp.GetCenter(),temp.GetRayon());
...
```

Draw() va appeler la méthode *Draw* des Circles composant le pentagone et nécessitant d'être dessinés (toujours les cotés 2 et 3 sauf dans le cas du Main pentagone ou il faut tout dessiner).

Centre() donne le centre du pentagone, elle prends en argument le centre du pentagone père.
h[] contient les côtés du pentagone, sommet[] les sommets et center le point centre.

La classe Circle

Classe stockant les coordonnées d'un cercle. Un Circle contient un Point centre et un rayon.

```
class Circle
{
    public:
        Circle(Point,double);
        void draw(const HDC &,int);
        ...
    private:
        Point centre;
        double rayon;
};
```

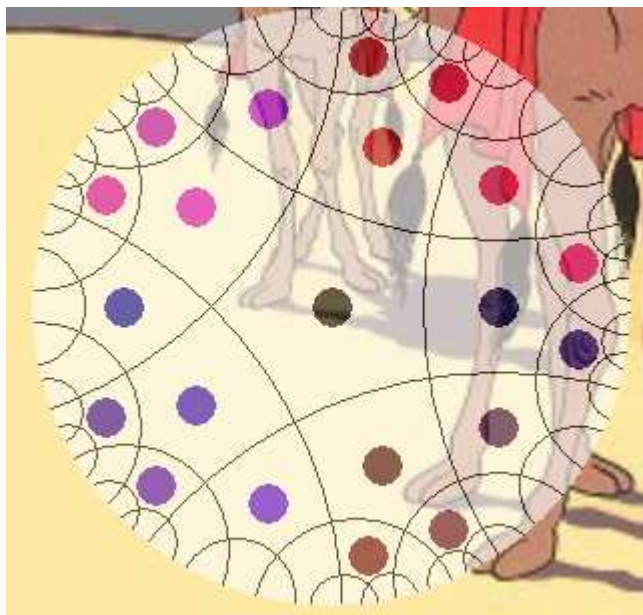
La méthode draw récupère le contexte graphique et calcule le rectangle dans lequel le cercle est inscrit et trace l'ellipse dans le contexte graphique en multipliant les coordonnées par le facteur d'agrandissement

La classe Point

Classe Stockant les données relatives à un point, c'est-à-dire sa coordonnée X et sa coordonnée Y. Win32 propose une classe HPoint pour utiliser des points mais dans notre cas nous n'avons pas pu l'utiliser, HPoint ne supporte pas des coordonnées type Double mais prend uniquement des Int.

```
class Point
{
    public:
        Point(double cx,double cy) {x=cx;y=cy;};
        Point() {x=y=0;};
        void draw(const HDC & hDC,int facteur)
        double x,y;
};
```

La méthode draw dessine le point sous forme d'un cercle coloré, la couleur étant fonction des coordonnées du point.

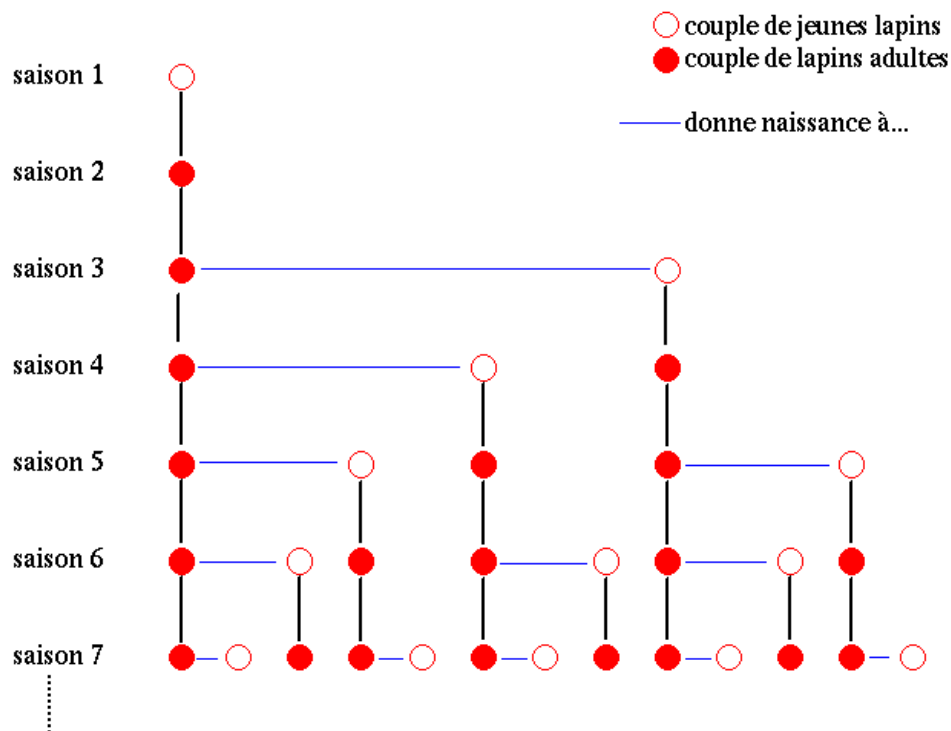


L'application test finale avec les centres et l'alphablend

III. Réalisation de la numérotation basée sur la suite de Fibonacci

1. Suite de Fibonacci

Léonard de Pise, plus connu sous le nom de Fibonacci (bien que ce nom ne lui ait été donné qu'au XIX^e siècle par l'historien des mathématiques Guillaume Libri), est à l'origine du premier modèle mathématique de la croissance des populations. Il étudia du point de vue numérique la reproduction des lapins. L'unité de base est un couple de lapins, il considère qu'un couple de jeunes lapins met une saison à devenir adulte, attend une deuxième saison de gestation, puis met au monde un couple de jeunes lapins à chaque saison suivante. En supposant que les lapins ne meurent jamais, on obtient donc le schéma ci-dessous :



Lorsque l'on met côte à côte le nombre de couples de lapins à chaque saison, cela donne la suite de Fibonacci:

{1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, ...}

Définition mathématique de la suite

Le nombre de couples de lapins U_n à la saison n est égal au nombre de couples de lapins adultes, c'est-à-dire le nombre total de lapins qu'il y avait à la saison précédente $n-1$, auquel on ajoute le nombre de couples de jeunes lapins, qui est égal au nombre de couples de lapins adultes à la saison précédente, donc au nombre total de couples à la saison $n-2$.

C'est pourquoi on a :

$$U_n = U_{n-1} + U_{n-2}$$

Pour être complet, il faut noter que $U_0 = U_1 = 1$ (ceci constitue l'initialisation de la suite)

2. Application à la numérotation des cellules

Après avoir dessiné le cercle hyperbolique, on voudrait bien réussir à afficher des données dans ces pentagones. Il faut pour cela trouver un moyen d'ordonnancement des pentagones dans la structure *disquePointcareDraw* pour que l'on ait un moyen de retrouver facilement quel pentagone graphique correspond à quel pentagone dans la structure hiérarchique dérivante du Main pentagone.

a. Taille du disque

La suite de Fibonacci est utilisée d'abord pour calculer le nombre de pentagone que contient une zone du disque pour une hauteur donnée :

$U_{2*hauteur}-1$ = nombre de pentagones sur le disque

Pour un cercle de hauteur 3, on aura donc $U_{3*2}-1=12$ pentagones dans chacune des zones du disque, pour une hauteur de 4, on passe à 33 pentagones et ainsi de suite.

Le nombre total de pentagones est donné par :

$$5*U_{2*hauteur} - 4$$

On détermine ainsi la taille du tableau de Pentagones *disquePointcareDraw* que l'on alloue dynamiquement lors du pavage (méthode *DisquePoincare::Paver()*).

b. Tri des pentagones dans le tableau

La suite de Fibonacci va servir dans un second temps à ranger les pentagones dans les bonnes cases du tableau *disquePointcareDraw*. On ne peut pas se contenter simplement de ranger les pentagones au fur et à mesure de leur construction car pour afficher et manipuler les données qu'ils contiennent, on a besoin de pouvoir les adresser précisément. Il faut un lien entre la hiérarchie de pentagones et le stockage mémoire de ceux-ci. La solution consiste à trouver un système de numérotation des pentagones pour que l'on puisse les ranger suivant leur numéro dans le tableau.

Le pentagone 1 de la zone 1 va dans *disquePointcareDraw[1][1]*

Le pentagone 2 de la zone 1 va dans *disquePointcareDraw[1][1]*

Le pentagone 65 de la zone 3 va dans *disquePointcareDraw[3][65]*

Le pentagone 10 de la zone 2 va dans *disquePointcareDraw[2][10]*

...

Le principal problème reste qu'un numéro d'identifiant doit pouvoir être déduit du numéro d'identifiant père et que celui-ci soit unique pour toute la zone, sinon on va écrire plusieurs pentagones dans la même case.

Une zone doit être marquée comme suit

Principe des masques

On va utiliser la suite de Fibonacci comme un masque en associant des bits forts ou faibles aux numéros de la suite. La somme des numéros associés à un bit fort donnera un identifiant pour un pentagone.

Quelques exemples : (en index, la suite de Fibonacci inversée)

13	8	5	3	2	1	Total
					1	1
				1	0	2
				1	1	3
		1	0	0	0	5
		1	0	0	1	6

Le pentagone ayant pour identifiant le code Fibonacci "1001" ira en case 6 du tableau pour sa zone respective.

6 aurait malgré tout pu être composé autrement, c'est aussi la solution du code "111". Il faut trouver une méthode algorithmique pour obtenir simplement les codes des descendants à partir de notre propre code.

Un code de Fibonacci est donc une suite de 0 et de 1 (pour le côté pratique, il est stocké en chaîne de caractères). On peut généraliser l'écriture d'un code *w* en écrivant

$$w = \alpha_n \dots \alpha_1$$

Où *n* est la taille en bits du code, α_n représente le bit de poids fort, α_1 est le bit de poids faible. Pour le code "1001", *n* vaut 4, α_n vaut 1, α_{n-1} vaut 0 ...

Nous représenterons les codes sous cette forme dans l'algorithme général

Algorithme de réalisation des masques

Le premier pentagone d'une zone va prendre le code fibonacci "1". C'est un trois nœuds, son ancêtre a le code fibonacci "0", c'est le Main pentagone.

Le pentagone Courant est P, ses fils sont P_1 à P_3 , le code de P est ω_P , les codes des fils sont ω_{P_1} à ω_{P_3} .

```

Si P est un deux nœuds faire
     $\omega_{P_1} = \omega_P \cdot "00"$ 
     $\omega_{P_2} = \omega_P \cdot "01"$ 
sinon faire                //P est un trois noeuds
     $\omega_{P_2} = \omega_P \cdot "00"$ 
     $\omega_{P_3} = \omega_P \cdot "01"$ 
     $\omega_{P_1} = (\omega_P) - 1 \cdot "10"$ 
fin faire
  
```

Reste à savoir calculer $(\omega_P) - 1$. L'opération doit nous donner les résultats suivants :

Pour le code fibo valant 8, on retourne le code fibo valant 7 ("10000" -> "1010")

Pour le code fibo valant 12, on retourne le code fibo valant 11 ("10101" -> "10100")

...

Écriture du code fibo de P :

$\omega_P = \alpha_{n(p)} \dots \alpha_{1(p)}$

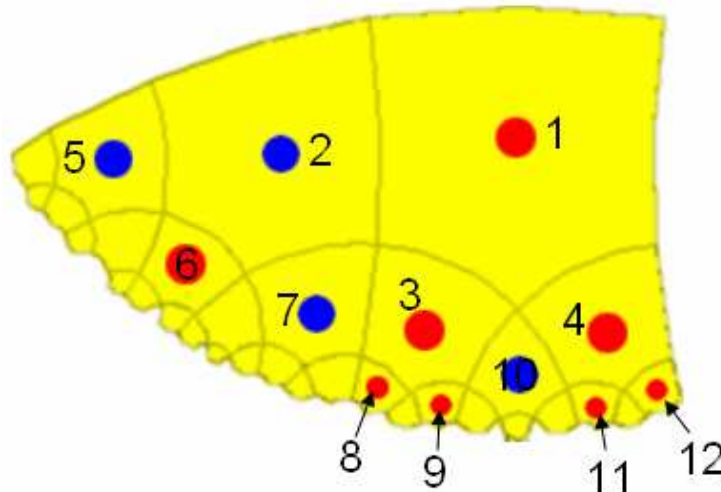
```

Si  $\alpha_{1(p)} = "0"$  faire
     $(\omega_P) - 1 = \alpha_{n(p)} \dots \alpha_{2(p)} \cdot "0"$ 
Sinon faire                // le code se termine par un 0
                            // et est de type  $\alpha_{n(p)} \dots "1" \cdot ("00")^n$ 
     $(\omega_P) - 1$  = Remplacer les couples ("00") de niveau inférieur au
                    "1" de plus bas niveau par ("10")
                    Remplacer le "1" de plus bas niveau par un "0"
Fin faire
  
```

En utilisant cet algorithme, nous pouvons localiser très simplement les fils et les ancêtres d'un pentagone dans le tableau en générant leurs codes de Fibonacci et en calculant la somme associée à l'aide de la suite.

Numérotation des pavés par zones :

L'application de l'algorithme dans une zone nous donne cet adressage :



On observe que les identifiants sont bien uniques mais en plus, les pentagones sont ordonnés par génération (ou par hauteur suivant la formulation) :

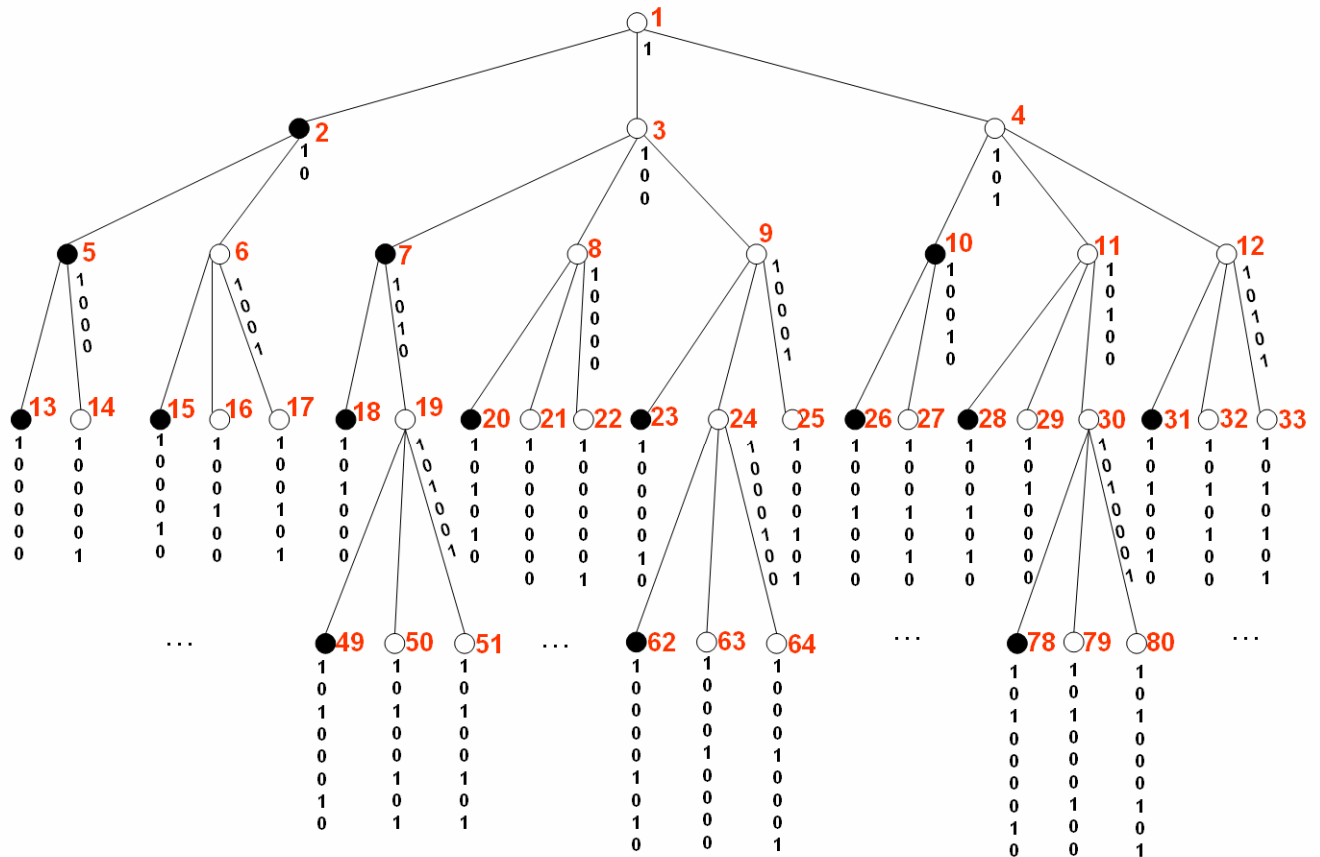
Génération 1 : P{1}

Génération 2 : P{2,3,4}

Génération 3 : P{5,6,7,8,9,10,11,12}

Lors du parcours du tableau de stockage des pentagones pour par exemple redessiner le contenu des cellules ou pour redessiner le disque, on va redessiner de façon homogène en commençant par le centre pour finir par les bords de la zone.

c. Hiérarchie des codes de Fibonacci pour le disque sur 4 niveau (presque 5)



IV. Affichage des données dans le pavage

1. Affichage de nombres

Pour afficher des nombres, notamment de la numérotation de Fibonacci, il faut surdéfinir les méthodes d'affichage des Polygones et des Points.

La méthode d'affichage des Polygones appelle la méthode d'affichage des Points pour afficher le point central du Polygone.

La méthode d'affichage des Points, convertie le nombre (qui est un unsigned long int s'il s'agit du numéro de Fibonacci représentant le Polygones que l'on est en train d'afficher) en une chaîne de caractères grâce à la méthode *sprintf*, puis on affiche cette chaîne de caractères grâce à la méthode *TextOut*

Syntaxe :

- Dans la méthode draw de la classe Polygones :
appel de la méthode draw de la classe Point sur le point central du polygone
`center.draw(hdc,facteur,value);`
- Dans la méthode draw de la classe Point :
On modifie le nombre de Fibonacci en chaîne de caractères. Remarque : il faut inclure la bibliothèque <stdio.h>

```
int sprintf( char *buffer, const char *format [, argument] ... );
```

On affiche la chaîne de caractères obtenue à l'écran grâce à la fonction suivante. Si la fonction réussit, la valeur de retour est différente de zéro. Si la fonction faillit, la valeur de retour est zéro.

```
BOOL TextOut(
    HDC hdc,           // handle vers la fenêtre (device context)
    int nXStart,       // coordonnée X de début de l'affichage
                        // x-coordinate of starting position
    int nYStart,       // coordonnée Y de début de l'affichage
                        // y-coordinate of starting position
    LPCTSTR lpString,  // pointeur de la chaîne de caractères
    int cbString        // nombre de caractères de la chaîne
);
```

2. Affichage d'icônes

Pour afficher les icônes, il faut, comme pour les nombres, surdéfinir la méthode draw de la classe Point.

```
//LOU début draw le Penta avec une icone
void Penta::draw(const HDC & hdc,int facteur, HICON icon)
{
    if(type==2)
    {
        for(int i=0;i<5;i++)
        {
            h[i].draw(hdc,facteur);
            sommet[i].draw(hdc,facteur);
        }
    }
    else
    {
        h[1].draw(hdc,facteur);
        h[2].draw(hdc,facteur);
    }
    center.draw(hdc,facteur,icon);
} //LOU fin
```

```
void draw(const HDC & hdc, int facteur, HICON icon)
{
    //Permet d'afficher l'icone "icon" à partir des coordonnées du Point
    DrawIcon(hdc, (int)(x*facteur+facteur-20), (int)(y*facteur+facteur-20), icon);
}
```

```
HICON ** iconeCentralePenta; //LOU
};
```

```
void initialiselconeCentralePenta()
{
    iconeCentralePenta = new HICON *[6];
    iconeCentralePenta[0] = new HICON;
    /*
    #define IDI_REGION_0          129
    #define IDI_REGION_1          130
    #define IDI_REGION_2          131
    #define IDI_REGION_3          132
    #define IDI_REGION_4          133
    #define IDI_REGION_5          134
    */
    for (int r=1; r<6; r++)
    {
        //on alloue la mémoire pour créer notre structure
        iconeCentralePenta[r] = new HICON [structSize];
    }
    for (int i=0; i<6; i++)
    {
        for (int j=0; j<(int)structSize; j++)
        {
            int numerolconeRef; // numéro identifiant l'icône dans ressource.h

            HINSTANCE hExe; // handle to loaded .EXE file
            HRSRC hResource; // handle for FindResource
            HRSRC hMem; // handle for LoadResource
            BYTE *lpResource; // address of resource data
            int nID; // ID of resource that best fits current screen

            HICON hlcon1; // icon handle
            //l'icone fait 32*32 pixels de côté
            int wlcon = 32; // width of icon
            int hlcon = 32; // height of icon

            switch (i) //C'est juste pour voir si on peut afficher des icônes différentes
            {
                case 1 :
                {
                    numerolconeRef = 130;
                    break;
                }
                case 2 :
                {
                    numerolconeRef = 131;
                    break;
                }
                case 3 :
                {
                    numerolconeRef = 132;
                    break;
                }
            }
        }
    }
}
```

```

    }
case 4 :
    {
        numerolconeRef = 133;
        break;
    }
case 5 :
    {
        numerolconeRef = 134;
        break;
    }
default :
    {
        numerolconeRef = 129;
        break;
    }
}

// Load the file from which to copy the icon.
//on a besoin d'un handle vers la fenetre d'execution pour loadRessource
hExe = LoadLibrary("cerclehyp.exe");

// Find the icon directory whose identifier is numerolconeRef.
//On charge le dossier contenant l'icone selon son numerolconeRef
hResource = FindResource(hExe, MAKEINTRESOURCE(numerolconeRef),
RT_GROUP_ICON);

// Load and lock the icon directory.
//on charge l'icone
hMem = (HRSRC)LoadResource(hExe, hResource);
lpResource = (BYTE*)LockResource((HRSRC)hMem)
;
// Get the identifier of the icon that is most appropriate
// for the video display.
nID = LookupIconIdFromDirectoryEx((PBYTE) lpResource, TRUE,
                                wlcon,
                                hlcon,
                                LR_DEFAULTCOLOR);

// Find the bits for the nID icon.
hResource = FindResource(hExe, MAKEINTRESOURCE(nID),
MAKEINTRESOURCE(RT_ICON));

// Load and lock the icon.
hMem = (HRSRC)LoadResource(hExe, hResource);
lpResource = (BYTE*)LockResource(hMem);

// Create a handle to the icon.
hlcon1 = CreateIconFromResourceEx((PBYTE) lpResource,
                                SizeofResource(hExe, hResource), TRUE,
                                0x00030000,
                                wlcon,
                                hlcon,
                                LR_DEFAULTCOLOR);

iconeCentralePenta[i][j]= hlcon1;

/*

```

```

HANDLE LoadImage(
    HINSTANCE hinst, // handle of the instance containing the image
    LPCTSTR lpszName, // name or identifier of image
    UINT uType, // type of image
    int cxDesired, // desired width
    int cyDesired, // desired height
    UINT fuLoad // load flags
);
*/
}
}
};

```

```

DrawIcon( HDC hDC, //
    int coordonnée_x, //
    int coordonnée_y, //
    HICON icône); //

```

3. Affichage de couleurs

extFloodFill

soit donner une couleur de départ

Soit donner une couleur de bord à laquelle s'arrêter de remplir

V.Déplacement des données dans le pavage

Conclusion

I.Limites

La principale limite dans l'élaboration de ce travail a été la complexité technique du domaine puisque la connaissance de plusieurs sciences est nécessaire à sa compréhension. La deuxième limite fut le manque de temps puisqu'il s'agit d'un domaine très actif qui regorge d'informations sur le Web. Il aurait donc fallu des semaines intensives de recherche sur Internet pour en faire à peu près le tour. Par conséquent, ce travail est loin d'être exhaustif et des sites très intéressants restent certainement à découvrir. D'autre part, seules les langues françaises et anglaises ont été traitées, en insistant beaucoup sur la langue française.

II.Synthèse

La synthèse vocale, bien que peu connue du grand public, est utilisée dans de plus en plus d'applications informatiques ou de la vie de tous les jours. Elle reste cependant principalement utilisée pour des publics spécifiques : les personnes mal ou non voyantes, les personnes illettrées, par exemple.

Les différents utilisateurs ont des attentes spécifiques vis à vis de la synthèse vocale. Les déficients visuels recherchent plutôt une synthèse vocale très rapide, même si la voix et la prononciation ne sont pas parfaites, pour simuler une lecture rapide d'un écran qu'il ne peuvent pas voir (ordinateurs, écrans de domotique,...). Quant à un public dit normal, il n'utilise une synthèse vocale que si la voix est parfaite (la plus proche possible de la voix humaine au niveau de l'intonation, de la prononciation, ...) et dans des contextes particuliers : absence d'écran (voiture, téléphone, domotique), jeux et multimédia, apprentissage de la prononciation des mots (pour les personnes illettrées ou pour l'apprentissage de nouvelles langues par exemple), ou quand l'utilisation de moyens d'interactions visuels entre l'Homme et la machine est surchargée (simulations, milieu industriel et applications en temps réel).

Il existe une large offre de logiciels sur le marché qui permettent essentiellement l'utilisation des nouvelles technologies pour les mal et non voyants, l'apprentissage (langues, prononciation,...) et la "lecture" de tout type de textes. Certains de ces logiciels permettent aussi de redéfinir beaucoup de paramètres, voir d'intégrer ses propres morceaux de code informatique afin de mieux correspondre à nos besoins.

Pour développer ces logiciels, il existe plusieurs API qui définissent des méthodes de programmation, des interfaces et qui fournissent la documentation nécessaire à leur utilisation. Les principales, concernant la synthèse vocale, sont MSAPI (développée par Microsoft) et JSAPI (développée par Sun). Ces deux API font office de norme dans la synthèse vocale. Mais il existe bien d'autres API : SALT et ASAPI, des extensions de MSAPI. Ces API s'appuient sur des moteurs de synthèse vocale ou TTS qui sont TTS pour MSAPI et FreeTTS pour JSAPI. Enfin le langage de programmation VoiceXML, standardisé par le W3C, permet de développer des applications vocales essentiellement pour le Web et peut être considéré quasiment comme une API.

III.Suggestions

Pour en apprendre plus, on pourrait faire des recherches sur les noms de personnes ou d'institutions actives dans le domaine et qui figurent sur les sites répertoriés. D'autre part, les liens sur le site [Center for Spoken Language Understanding @ OGI](#) du "Oregon Graduate Institute of Science and Technology" aux États-Unis nous amène vers de nombreuses pistes intéressantes et variées à explorer.

Lacunes de la visualisation de l'information

Largeur de bande accrue

Les interfaces à forte intensité graphique nécessitent une largeur de bande accrue. Bien que cela ne pose pas de problème sur le réseau Intranet local, cela pourrait rendre la visualisation peu pratique pour l'utilisation générale sur Internet. On ne peut dépasser la vitesse à laquelle un simple texte ASCII peut être transmis même lorsque les interfaces de services populaires en direct ou sur disque sont claires.

L'application de visualisation «Butterfly» élaborée au PARC de Xerox est une solution 2 http://www.acm.org/sigchi/chi95/Electronic/documents/papers/jdm_bdy.htm. Essentiellement une fonction de navigation spécialisée, elle effectue un bon nombre des opérations de visualisation sur le système-client. L'autre solution consiste à créer les objets à l'aide du VRML (langage de modélisation de réalité virtuelle) qui permet la représentation d'espaces tridimensionnels complexes par le texte seulement.

Occlusion

L'une des difficultés techniques liées à la visualisation en trois dimensions est que certains objets peuvent être cachés périodiquement par d'autres objets au premier plan. Ce phénomène s'appelle l'**occlusion**. Dans des espaces d'information comprenant des centaines d'objets, cette obstruction peut nuire à l'interprétation de l'information par l'utilisateur. (Soulignons que cela reste préférable aux affichages actuels, dans lesquels tous les documents font l'objet d'«occlusion» sauf ceux qui apparaissent immédiatement à l'écran.)

L'une des solutions consiste à projeter une source lumineuse (virtuelle) à partir de la partie supérieure ou de la partie latérale de l'espace, en repoussant les ombres des objets à l'arrière-plan. Cela permet à l'utilisateur de savoir qu'il y a occlusion d'un objet même si ce dernier n'est pas visible directement.

Manque de graphisme normalisé

Actuellement, il existe trois grandes façons de donner une structure à l'information : le **lien hiérarchique** et le **lien relationnel** (p. ex., les renvois *voir aussi*) et la **liste séquentielle** (alphabétique et numérique). En plus d'être polyvalentes, ces méthodes ont comme principal avantage d'être comprises sans distinction. Toutes les personnes qui ont en commun le même alphabet et le même système de numérotation peuvent structurer l'information de la même manière.

Cependant, peu de méthodes sont unanimement reconnues pour structurer l'information en deux ou en trois dimensions. À moins que l'utilisateur puisse comprendre intuitivement la signification des objets et leur lien dans l'espace, toute visualisation de l'information comporte une valeur restreinte. Il est donc vraisemblable que la visualisation serve d'abord à compléter plutôt qu'à remplacer l'information textuelle.

Conclusion

Bien que certains aspects de la visualisation de l'information aient été intégrés à plus d'une application Internet et de gestion, ses incidences sur l'élaboration d'interfaces-utilisateurs sont inconnues jusqu'à maintenant. La plupart des obstacles techniques ont été franchis; la difficulté que doit maintenant surmonter la visualisation de l'information est d'être acceptée par les utilisateurs qui sont habitués aux représentations linéaires traditionnelles de l'information. À moins que l'on puisse créer un graphisme puissant pour structurer les métadonnées, les progrès de la recherche en visualisation passeront

Bibliographie

Introduction

- [droit1] <http://www.bouthier.net/>
[droit2] <http://slashdot.org/articles/99/03/19/096203.shtml>
[droit 3 art 52, 2c] <http://www.european-patent-office.org/legal/epc/e/ar52.html>
[droit4] <http://Hypertree.sf.net>
[interet] Article On interactive Visualization of Hight-Dimensional Data using the Hyperbolic Plane, Jörg A. Walter & Helge Ritter, 2002

A. Outils d'interaction et interfaces existants utilisant l'espace hyperbolique

Les arbres hyperboliques ou fisheyes-view

- [dictionnaire] <http://www-clips.imag.fr/geta/mathieu.mangeot/MM-These/partieB.html>
[inxight1] http://opera.inrialpes.fr/people/Laurent.Tardif/www_these/DEA/Chapitre_2/Chapitre_2.html
[inxight2] <http://www.inxight.com/>
[treebolic] <http://treebolic.sortilege.net/>
<http://treebolic.sortilege.net/fr/engine.htm>
<http://treebolic.sortilege.net/fr/generator.htm>
<http://treebolic.sortilege.net/fr/browser.htm>
[Webviz] <http://www.geom.umn.edu/docs/research/webviz/webviz/node9.html>
[nlc] <http://www.nlc-bnc.ca/>
[svt] <http://www.inrp.fr/Acces/Biogeo/inxight/slsbeta/menrt/menrt.html>
<http://www.inrp.fr/Acces/Biogeo/inxight/slsbeta/ressvt/ressvt.html>
[products_inxight] <http://www.inxight.com/int/fr/products/>
[ebi] <http://industry.ebi.ac.uk/~alan/VisSupp/>
<http://industry.ebi.ac.uk/index.html>
[H3] <http://graphics.stanford.edu/papers/h3cga/fig/dskwrk.gif>
<http://www-graphics.stanford.edu/papers/h3>

Les arbres coniques hyperboliques ou cone-tree

Les murs en perspectives ou Perspective-Wall

- [mur_temps] http://opera.inrialpes.fr/people/Laurent.Tardif/www_these/DEA/Chapitre_2/Chapitre_2.html
[FiCell] <http://iihm.imag.fr/vernier/ficell.html>

B. Annexes

Le disque d'Euclid et de Poincaré : bases mathématiques de l'espace hyperbolique

Introduction aux modèles hyperboliques plan [maths]

- [maths] <http://www-cabri.imag.fr/abracadabri/GeoNonE/GeoHyper/HIntro/KBversP2.htm>

Le disque de Poincaré [Poincare]

- [Poincare] http://www.sciences-en-ligne.com/momo/chronomath/anx3/disqu_beltrami.html

Annexes

I. Le disque d'Euclid et de Poincaré : bases mathématiques de l'espace hyperbolique

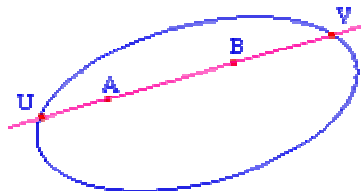
1. Introduction aux modèles hyperboliques plan [maths]

a. Trois modèles plans de géométrie hyperbolique

Historiquement, Cayley a d'abord montré que l'intérieur d'une conique (projective réelle) non dégénérée, munie de la distance

$$d(A,B) = k \cdot \left| \ln \left[\frac{AU/AV}{BU/BV} \right] \right|$$

est un modèle de géométrie hyperbolique dans lequel la droite (AB) est l'intersection de la droite euclidienne avec l'intérieur stricte de la conique.



Cette introduction utilise la distance de Cayley

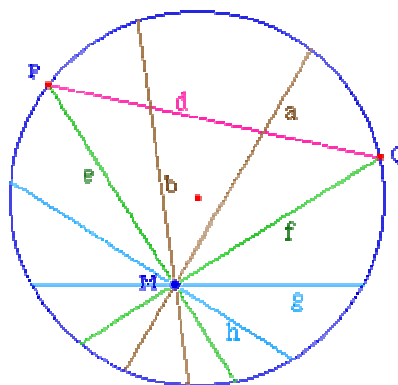
- Modèle de Klein-Beltrami

Suite à ses travaux sur la pseudo-sphère, Beltrami a proposé le modèle dit de "Klein-Beltrami" dans lequel la métrique est encore celle de Cayley (avec $k = 1$ par exemple)

Les droites sont les cordes du cercle, privées de leurs extrémités.

Parmi les droites du faisceau issu de M, on distingue :

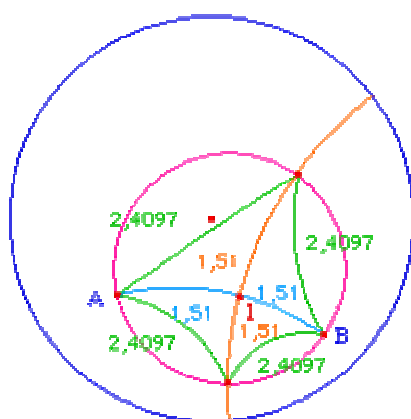
- les **droites sécantes à d** (exemple a et b)
- les **droites non connectables à d** (ex. e et f), celles que Lobachevsky appelle parallèles
- les **droites non sécantes à d** : elles sont connectables sans point commun, et donc ont une perpendiculaire commune avec d (exemple g et h).



Exemple

- Modèle du disque de Poincaré => conserve angles

Dans ce modèle, les **droites sont des arcs de cercle**, orthogonaux au **cercle horizon** (la conique absolue de Cayley). Par rapport au précédent, ce modèle est conforme : les angles hyperboliques sont les angles euclidiens des droites. On passe facilement du modèle de Klein-Beltrami au modèle de Poincaré, par le passage sur une sphère en perspective cavalière.

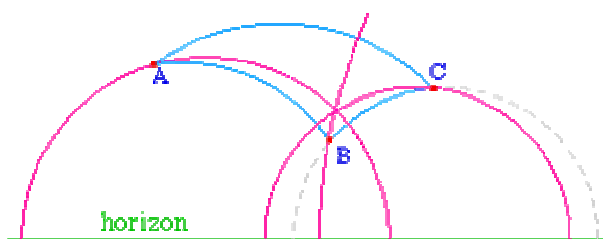


Un carré hyperbolique

- Modèle du demi-plan de Poincaré

Dans le demi plan des nombres complexes de la partie imaginaire positive, les droites sont les demi-cercles passant par deux points centrés sur l'axe réel (horizon) ou, s'il n'existe pas un tel cercle, la demi-droite, dans le demi-plan voulu, passant par ces deux points (elle est alors orthogonale à l'horizon).

Ce "ou exclusif" rend une Cabri-illustration moins simple (techniquement possible toutefois) puisque les droites ont deux formes différentes selon le contexte. C'est la raison pour laquelle on a privilégié le modèle précédent.



Un triangle ayant un orthocentre.

b. Les isométries de ces modèles au regard de l'axiomatique de Bachmann

Dans cette axiomatique, comme dans d'autres, le concept de réflexions et d'isométries remplace la congruence des axiomatiques de type Euclide-Hilbert : le groupe des isométries permettra d'écrire plus simplement - avec le vocabulaire de l'algèbre - certaines propriétés géométriques de base.

	Disque de Klein - Beltrami	Disque de Poincaré
Les droites	Cordes du disque sans les extrémités	Arcs de cercles orthogonaux au cercle horizon (sans les extrémités)
Symétrie orthogonale	Homologie harmonique d'axe cette droite et de centre son pôle par rapport à l'horizon	Inversion de cercle le support de cette droite
Pliage du disque selon la droite	Une telle homologie laisse le cercle globalement invariant (serait aussi vrai d'une conique pour le modèle de Klein) et envoie un des demi-plan sur l'autre.	Une inversion laisse globalement invariant un cercle qui lui est orthogonal et transforme un de ses "demi-plans" en l'autre.
	Car l'orthogonal d'une droite passe par le pôle et alors la droite est globalement invariante.	L'orthogonal étant un cercle orthogonal à l'axe de la symétrie, on a la même propriété que ci-contre.

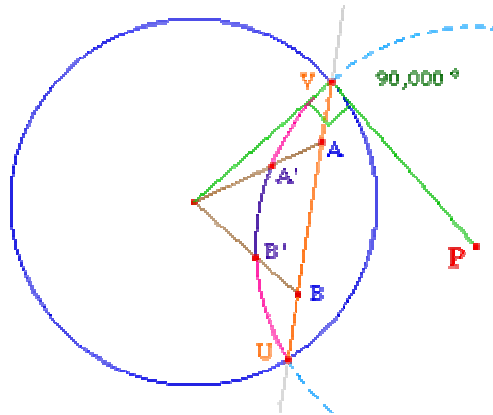
Remarque :

Le demi-plan de Poincaré est le cas où le cercle horizon a son centre à l'infini : les symétries orthogonales sont les mêmes inversions que dans le cas du disque fini.

On notera que géométriquement, deux points fondamentaux sont communs aux deux modèles : le **centre du cercle** orthogonal à l'horizon passant par **la droite hyperbolique** de points idéaux **U** et **V** dans le modèle de Poincaré est le pôle de cette même droite idéale (**UV**) dans le modèle de Beltrami.

Ainsi d'éventuelles propriétés sur les pôles de segments dans les figures du modèle de Klein-Beltrami seront donc aussitôt transférables en propriétés sur les centres des segments hyperboliques correspondants dans le modèle de Poincaré. Ceci sera illustré dans cette introduction sur les polygones réguliers.

Cette simple remarque aide à mieux percevoir le passage d'un modèle à l'autre, et permet même de réaliser, par conjugaison, des constructions dans un modèle en utilisant leur homologue dans l'autre modèle. Le détail du passage effectif de **[AB]** dans le modèle de Beltrami à **[A'B']** dans celui de Poincaré ainsi qu'un exemple de telle "conjugaison" sont proposés dans Passage d'un modèle à l'autre de l'introduction générale aux constructions hyperboliques.

**Voir :**

Java hyperbolique : <http://www-cabri.imag.fr/abracadabri/abraJava/IntroCJava.html>

3. Passage entre les différents modèles plans hyperboliques

<http://www-cabri.imag.fr/abracadabri/GeoNonE/GeoHyper/HIntro/KBversP.htm>

[Construction dans le modèle du disque de Poincaré]

<http://www-cabri.imag.fr/abracadabri/GeoNonE/GeoHyper/HDroite/HDroite1.htm>

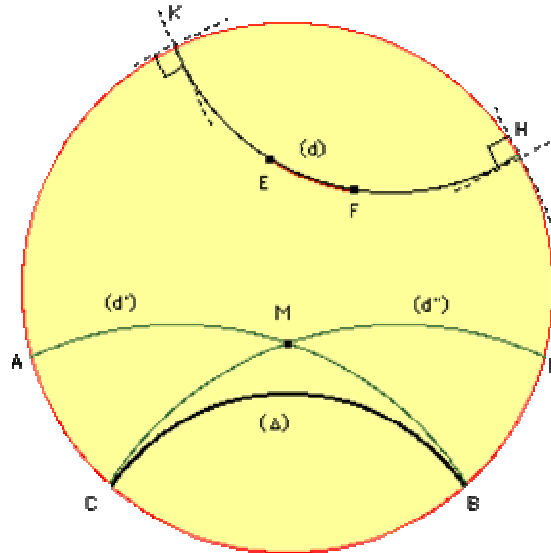
[Version Java]

<http://www-cabri.imag.fr/abracadabri/abraJava/HyperCJ/HIntroCJ.html>

2. Le disque de Poincaré [Poincare]

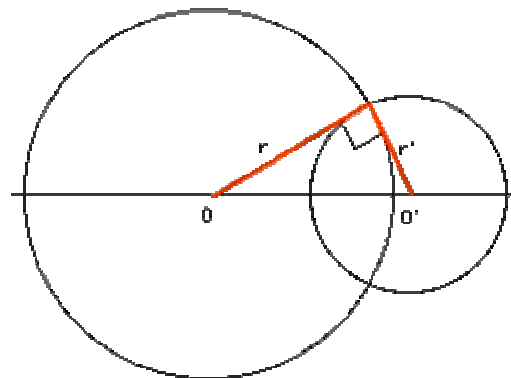
C'est le modèle de représentation de l'espace hyperbolique utilisé en générale dans les applications informatiques.

C'est au cours de ses recherches sur les fonctions automorphes que Poincaré exhiba plusieurs modèles, dans le plan et dans l'espace, susceptibles de décrire la géométrie hyperbolique. Le disque de Poincaré (1882) en est une représentation plane où le plan est encore réduit à un disque **(c)** de bord. Par définition, les droites sont des arcs de cercle orthogonaux à (c).



Deux cercles (c1) et (c2) de centres O et O', de rayons r et r', sont dits *orthogonaux* si leurs tangentes aux points d'intersection sont perpendiculaires. C'est dire que $OO'^2 = r^2 + r'^2$.

Deux droites seront dites parallèles si les arcs de cercles qui les définissent sont tangents en un point de (c).



Pour un point M situé hors de la droite (Δ), on peut mener deux parallèles à (Δ) : (d') et (d'').

La distance entre deux points E et F est comme précédemment calculée au moyen du rapport anharmonique [E,F,K,H].

Un modèle signale à cet égard que définir un sous-ensemble de l'espace euclidien nécessite les notions de "points", "lignes", "intersections", et "longueur" dans ce sous-ensemble, de sorte que les axiomes de la géométrie hyperbolique soient valides localement.

Plusieurs différents modèles ont été développés, et sont mieux connus comme le modèle de Klein et les modèles de Poincaré.

Le modèle de Klein utilise un disque ouvert (ou une sphère pour la 3d) comme un sous-ensemble, c.-à-d., le plan hyperbolique

dans ce modèle comprend les points dans le périmètre du disque. Des lignes hyperboliques sont représentées par des cordes du disque; l'intersection est juste l'intersection euclidienne. La seule différence principale est la longueur d'un segment de ligne. Nous ne donnerons pas ici une définition détaillée, puisqu'il suffit d'indiquer que cette longueur est définie en fonction de la position des points vis-à-vis du périmètre du disque: les segments qui sont conformes dans un sens hyperbolique sont exponentiellement plus petits (dans le sens euclidien) en approchant le périmètre.

Prouver la validité locale de tous les axiomes de la géométrie hyperbolique exige un certain travail non trivial. La validité de l'inversion du 5ème postulat d'Euclide est tout à fait évidente. Le modèle de Poincaré est tout à fait semblable bien que des lignes hyperboliques soient représentées par les arcs qui intersectent orthogonalement le périmètre du disque.

L'algorithme de dessin d'arbre sur le plan hyperbolique, utilise le modèle de Klein pour visualiser les résultats. Il est maintenant possible de donner une description plus exacte de ce que sont les dispositions hyperboliques des graphes: elles exécutent un algorithme de disposition dans le plan ou l'espace hyperbolique, puis affichent les résultats dans le plan ou l'espace euclidien familier en utilisant un des modèles de la géométrie hyperbolique. C'est-à-dire, ce que nous voyons n'est pas la géométrie hyperbolique intrinsèquement, mais ses représentations dans la géométrie euclidienne. Le papier initial de Lamping et autres a utilisé le modèle de Poincaré, tandis que Munzner utilise la plupart du temps le modèle de Klein.
<http://www.chez.com/stagerecherche/bibliohyperbol.htm>

3. Pavages dans l'espace hyperbolique

II. Glossaire

Arbre et sous-arbre ou arbre fils

Focus

Document hypermédia

C'est un document hypertexte usuel dans lequel des objets peuvent être de nature "multimédia" (video, son, ...). Dans ces documents l'organisation temporelle des objets est entièrement portée par la structure de navigation offerte au lecteur. La différence entre un document multimédia et un document hypermédia et l'existence du scénario temporel.

Magic lens

Menus en étoile

arbre hyperbolique, géométrie hyperbolique fisheye, disque de Poincaré

XML

Fisheye view ou objectif fisheyes : objectif ultra grand angulaire utilisé en photographie.

Par analogie avec l'«objectif ultra-grand-angulaire» utilisé en photographie, la technique ultra-grand-angulaire (aussi appelée mise au point + contexte) permet à l'utilisateur de voir en détail un objet au premier plan et bien d'autres objets de définition inférieure de façon périphérique. L'avantage tient à la conservation de la relation contextuelle entre un grand nombre d'objets tout en permettant à l'utilisateur un niveau de contrôle utile sur l'article sous la main.

PARC : Palo Alto Research Center

Une **API (Application Program Interface) [def_API]** est une interface de programmation d'application, contenant un ensemble de fonctions courantes de bas niveau, bien documentées, permettant de programmer des application de "Haut niveau". On obtient ainsi des bibliothèques de routines, stockées, par exemple dans des DLL ou des NLM.

DOM [dom] : le Document Object Model est une interface indépendante de la plate-forme et de la langue. Il offre la possibilité d'accéder et de mettre à jour dynamiquement le contenu, la structure et le style des documents.

T.E.R. : Travail ou travaux d'Etudes et de Recherches.

W3C [w3c] : le W3C Voice Browser Working Group s'occupe du développement d'une suite de spécifications qui devrait rendre possible l'interopérabilité des applications vocales. C'est ce qu'on appelle le W3C Speech Interface Framework. Ce framework contient des brouillons de travail sur : (The framework currently includes working drafts of:)

SRGS [srgs], un format de grammaire (grammar format)

SSML [ssml], un format pour la sortie vocale (a speech output format)

NLSML [nlsml], a natural language result format
CCXML [ccxml], and a telephony call control language

//ressources