ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE
M.Ing.

PAR
David GUILMAINE

CONTRÔLE ET PRÉSENTATION INTERACTIFS DES TRANSITIONS FLUIDES DANS LES VISUALISATIONS

MONTRÉAL, LE 30 AOÛT 2011

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Michael McGuffin, directeur de mémoire
Département de génie logiciel et des TI à l'École de technologie supérieure

M. Pierre Bourque, président du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

M. Luc Duong, membre du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 25 AOÛT 2011

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

J'aimerais d'abord remercier M. Michael McGuffin, mon directeur de recherche, pour son inspiration, ses idées, son support, ses précieux conseils, ses commentaires et son aide fort appréciée dans la réalisation de ce projet. C'est le contenu d'un de ses cours qui a motivé le choix de poursuivre mes études en réalisant un projet dans un tout nouveau domaine très intéressant. Je voudrais aussi souligner la collaboration de mes collègues du groupe de recherche HIFIV (*Human Interaction For Information Visualization*) pour avoir participé de près ou de loin au début de ce projet ou encore pour leur participation dans l'expérimentation.

Je veux aussi souligner la grande confiance accordée par M. Pierre Bourque en tant qu'auxiliaire d'enseignement et chargé de cours. Grâce à vous, j'ai pu acquérir une expérience personnelle et professionnelle qui n'est pas accessible à moins d'avoir une opportunité très privilégiée.

De plus, je remercie les membres du jury pour leur collaboration dans le contexte accéléré de l'évaluation de mon mémoire et de ma soutenance.

Toutefois, la réalisation de cette maîtrise n'aurait pas été possible sans l'appui de mes parents, France Gazaille et Jean Guilmaine, tout au long de mes études. Les valeurs et la persévérance que vous m'avez inculquées ont une fois de plus porté fruit et j'espère qu'il en sera de même pour le début de ma carrière d'ingénieur. Finalement, je voudrais adresser un merci tout particulier à ma copine, Vanessa Maltais, pour sa patience, sa motivation et ses nombreux encouragements.

CONTRÔLE ET PRÉSENTATION INTERACTIFS DES TRANSITIONS FLUIDES DANS LES VISUALISATIONS

David GUILMAINE

RÉSUMÉ

Le but de cette étude est de proposer de nouveaux types de transitions fluides qui permettent d'améliorer le suivi des éléments et la compréhension de l'utilisateur pour la visualisation de données. Elle propose aussi de nouveaux types de *widgets* plus complets et flexibles pour le contrôle des animations en intégrant les propriétés essentielles des techniques existantes.

Pour analyser les types d'animations, un prototype de visualisation de données sous la forme d'une arborescence a été développé. Deux types d'animations existantes, les animations linéaire et par étapes, ont été réalisés dans le prototype selon trois types de changements sur les nœuds (fermetures, ouvertures et permutations) pouvant s'opérer sur les différents niveaux de l'arborescence. De plus, deux nouveaux types d'animations, les transitions hiérarchique et hybride, ont été conçus et réalisés dans le même prototype. Ces dernières sont conçues pour exploiter la structure hiérarchique de l'arborescence, et pour mieux diriger l'attention de l'utilisateur selon les différents niveaux.

Pour évaluer ces quatre techniques d'animation, une expérimentation contrôlée a été réalisée où les participants devaient identifier un nœud à suivre et des nœuds ayant subi un type de changement en particulier. Les résultats ont démontré un avantage significatif pour les animations hiérarchique et hybride face aux animations linéaire et par étapes pour les permutations. De plus, ils confirment que l'animation par étapes ne présente pas d'avantages face à une simple animation linéaire.

Aussi, l'analyse des techniques existantes de contrôles interactifs a fait ressortir trois propriétés essentielles que les composants d'interface contextuels pour le contrôle des animations devraient posséder : le contrôle discret, le contrôle continu et le repérage de la position en cours. Pour les intégrer, des maquettes de nouveaux *widgets* ont été présentées en rapport avec une taxonomie les classifiant selon le type de passage du contrôle discret à continu et la forme du glisseur.

Mots-clés : animations, transitions fluides, visualisation, arborescences, *widgets*, menu contextuel, contrôle

INTERACTIVE PRESENTATION AND CONTROL OF SMOOTH TRANSITIONS IN VISUALIZATIONS

David GUILMAINE

ABSTRACT

The purpose of this work is to propose new types of smoothly animated transitions for data visualization that facilitate visual tracking of items and overall perception by a user. The work also proposes new types of widgets for interactively controlling transitions that are more complete and flexible than previous techniques.

To analyze different types of animated transitions, a prototype has been developed to visualize tree structures. Two types of previously published transitions, linear and staged animations, have been implemented in the prototype, allowing for three types of changes on the nodes (collapsing, expanding and permuting) that can occur at different levels of the tree. Additionally, two novel transition techniques, hierarchical animations and hybrid animations, were designed and implemented in the prototype. These new transition techniques are designed to leverage the hierarchical structure of the data, and better direct the user's attention according to the levels of the tree.

To evaluate these four animation techniques, a controlled experiment was conducted where participants were asked to track a specific node and also identify other nodes that had been collapsed, expanded or permutated over the course of a transition. The results show a significant advantage for hierarchical and hybrid animations over linear and staged animations for observing permutations. The results also found no advantage of staged animations over simple linear animations.

Next, an analysis of existing techniques for interactive control of animations highlighted three key functionalities that popup widgets for animation control should have: enabling discrete control, enabling continuous control, and displaying the current position in the transition. To combine these functionalities, several novel widget designs are presented and organized within a taxonomy classifying them according to the type of interaction used to switch between discrete control and continuous control, and also according to the direction of movement used for continuous control.

Keywords: animations, smooth transitions, visualizations, trees, widgets, popup menu, control, interaction

TABLE DES MATIÈRES

Page

LISTE DES FIGURES

Page

INTRODUCTION

La visualisation de l'information émerge de travaux réalisés dans le domaine de la visualisation scientifique dans le but de développer des techniques d'analyse des données pour la physique, la chimie, les mathématiques, les statistiques ou l'ingénierie. En complémentarité avec les approches traditionnelles de traitement des données, la visualisation a permis de faire ressortir des éléments intéressants qui n'auraient peut-être pas été découverts autrement. Elle permet d'établir des liens qui sont parfois au-delà des compétences cognitives de l'être humain (Card, 2008). C'est pourquoi elle s'est vite répandue à d'autres domaines comme le commerce et les études de marché, l'analyse financière, l'industrie manufacturière pour le contrôle de la production et même le divertissement. À un niveau restreint, des éléments propres à la visualisation de données se retrouvent souvent dans les journaux ou à la télévision en complément de l'explication d'une situation quelconque. Certains logiciels utilisent aussi certains éléments pour classifier les fichiers présents sur le disque dur de l'ordinateur ou encore pour montrer le niveau d'utilisation d'un réseau informatique. Toutefois, les images employées pour présenter les informations peuvent devenir très complexes et peuvent perdre leur efficacité si les données doivent être présentées avec plusieurs dimensions différentes ou si leur nombre est considérable.

Souvent, des représentations statiques d'une vue sur des données peuvent être suffisantes pour bien les analyser. Cependant, pour les cas plus difficiles, les utilisateurs ont besoin d'interagir avec un logiciel pour fureter les données, choisir différents sous-ensembles, changer de vue, afficher plusieurs vues en même temps, ou même voir des vues animées. Les animations servent donc parfois de moyen pour montrer des données à plusieurs dimensions, mais aussi comme façon de montrer des transitions fluides entre différentes vues statiques, soit en réponse aux interactions de l'utilisateur ou bien pour montrer des données qui évoluent dans le temps ou selon d'autres facteurs. Ainsi, le mouvement des informations à l'écran, grâce aux animations, a permis aux utilisateurs de mieux percevoir et interpréter les

changements par rapport à une transition soudaine (non animée). De plus, les animations ont apporté un côté plus attrayant à la visualisation de données, et ces animations sont parfois même contrôlables de façon interactive par la personne qui les observe. En effet, des *widgets* (composants d'interface) ont été créés pour que cette dernière puisse démarrer les animations et les rejouer en modifiant leur vitesse ou leur sens de progression pour mieux comprendre les détails d'une transition animée. Cela ajoute une autre possibilité pour mieux assimiler les images qui se succèdent. Cependant, le même problème s'est posé lorsque les données à animer deviennent trop complexes. Si trop d'éléments bougent de façon simultanée et désorganisée, l'utilisateur n'est donc plus en mesure de discerner la nature des changements qui s'opèrent et peut même perdre la trace de ceux-ci. C'est pourquoi la conception des animations doit être analysée et adaptée pour rendre plus efficace le suivi des éléments et améliorer la compréhension des utilisateurs. Également, les techniques d'interaction pour contrôler les animations doivent être enrichies pour les rendre plus flexibles et complètes en combinant plusieurs méthodes existantes.

Cette présente étude tentera donc de répondre au problème concernant la complexité des animations en proposant plusieurs types de transitions fluides pour la visualisation de données, particulièrement pour visualiser des arborescences. De par la quantité d'information et les liens qui unissent les données entre elles, ce contexte est propice à rendre les animations complexes, mais en même temps structurées par les liens entre les nœuds. Dans les chapitres qui suivent, les techniques d'animation existantes seront adaptées à ce contexte et de nouvelles seront proposées pour faciliter la perception et la compréhension des animations en exploitant la structure en niveaux des arborescences. Par la suite, deux des nouvelles techniques d'animation proposées, l'animation hiérarchique et l'animation hybride, seront évaluées et comparées avec des approches antérieures dans une expérimentation contrôlée qui permettra de mieux comprendre les facteurs qui influencent la réussite ou non d'un type d'animation en particulier. Ainsi, après l'analyse des résultats obtenus, des directives de conception pourront être émises dans le but de guider le développement ou l'amélioration des transitions fluides. De plus, l'inventaire des caractéristiques des techniques

d'interaction existantes pour le contrôle des animations permettra de les définir et les classifier. Ainsi, de nouveaux *widgets* seront conçus pour améliorer le contrôle de l'utilisateur sur les transitions fluides en intégrant plusieurs concepts qui sont présents dans les techniques antérieures, mais qui n'ont pas été intégrés, jusqu'à maintenant, en une seule technique d'interaction cohérente.

Pour détailler l'approche adoptée, ce mémoire présente quatre chapitres. Le premier comporte une revue de la littérature dans le domaine des animations et des techniques d'interaction ou de contrôle sur celles-ci. Par la suite, le deuxième chapitre traitera de l'analyse et de la conception des animations spécifiques aux arborescences en proposant quatre types de transitions différents. Le troisième chapitre portera sur l'expérimentation menée avec des participants pour mesurer la performance des animations, l'analyse des résultats obtenus et les directives de conception énoncées à la suite de l'interprétation. Finalement, le dernier chapitre se consacrera sur l'analyse et la conception des *widgets* interactifs pour le contrôle des animations en proposant des maquettes de nouveaux composants d'interface.

CHAPITRE 1

REVUE DE LA LITTÉRATURE

La revue de la littérature présente d'abord un survol sur l'utilisation et la conception des animations dans les interfaces graphiques en plus d'évaluer leur utilité et leur bénéfice par rapport à une interface statique. Par la suite, les animations seront analysées dans le contexte de la visualisation de données statistiques et la visualisation d'arborescences en mettant l'emphase sur les transitions animées par étapes. Finalement, quelques *widgets* (composantes d'interfaces) pour contrôler des variables ou des animations seront présentés afin de tracer les bases d'un nouveau mécanisme de contrôle pour les transitions animées.

1.1 L'utilisation des animations dans les interfaces graphiques

L'animation est définie comme étant l'art de dessiner le mouvement en plus de manipuler et d'interpréter les actions sous-entendues qui se déroulent entre chaque cadre clé. Cette définition indique que les transitions entre deux cadres peuvent être plus importantes que les cadres eux-mêmes (Baecker et Small, 1990). En effet, les domaines des sciences et des arts visuels affirment que le mouvement possède un potentiel de communication plus riche que les éléments statiques. Le mouvement et les animations peuvent être efficaces d'un point de vue perceptuel pour le recueil et le traitement de l'information en plus d'être facilement interprétables par l'homme. Bien qu'ils soient utiles pour aider à faire ressortir visuellement des changements tout en augmentant la capacité d'interprétation des systèmes complexes, ces concepts sont souvent sous-utilisés lors de la conception des interfaces graphiques (Bartram, 2007). Les animations peuvent être employées dans divers contextes comme pour montrer le déroulement d'une histoire, d'un processus ou d'une simulation, ou encore pour amuser à l'intérieur d'un jeu vidéo. Toutefois, cet ouvrage s'intéressera particulièrement à l'utilisation des animations pour montrer des changements d'état ou des transitions entre différentes vues de données.

En se basant sur les principes utilisés par les créateurs de dessins animés, Chang et Ungar (1993) ont créé un modèle appliqué aux éléments de l'interface graphique d'un système d'exploitation. En remplaçant les transitions soudaines par des transitions animées, ils ont démontré que les animations peuvent augmenter la compréhension, l'attrait et l'expérience des interfaces utilisateur. Plus tard, Thomas et Calder (2001) ont repris ce modèle afin d'étendre les principes proposés dans un contexte réaliste, soit dans un prototype de logiciel pour le dessin par ordinateur. Leurs études démontrent que les animations peuvent aider à diriger et concentrer l'attention de l'utilisateur sur des éléments clés de l'interface graphique. Cependant, ils notent qu'une animation inappropriée pourrait distraire l'utilisateur en attirant son attention sur l'animation elle-même et non sur la complétion efficace de la tâche en cours.

1.1.1 Présentation statique ou animée

À l'intérieur d'un logiciel de visualisation ou de furetage de données, il arrive souvent que l'utilisateur change la vue des données. La transition entre deux vues de données peut se produire de deux façons. Premièrement, la transition peut être soudaine (aussi appelée une transition statique ou immédiate), c'est-à-dire que la nouvelle vue s'affiche directement sans aucune animation (voir Figure 1.1, en haut). L'autre méthode consiste à utiliser une transition fluide ou animée (voir Figure 1.1, en bas) qui montre de façon constante la progression des changements entre la vue initiale et finale. Cette dernière permet de mieux comprendre la direction des mouvements et de bien associer la position de chaque élément du début à la fin de la transition. La Figure 1.1 montre un exemple où les données sont représentées par des points, mais une transition de vue peut arriver dans n'importe quelle visualisation de données par exemple dans les visualisations d'arborescences, de graphes ou d'histogrammes. Ces transitions peuvent impliquer un changement du positionnement, des couleurs, de la taille ou de l'orientation des éléments. Du point de vue de l'utilisateur, elles peuvent aussi opérer sur un défilement ou un zoom en deux dimensions, un changement de caméra dans une

visualisation en trois dimensions, ou encore un changement dans le genre de représentation visuelle.

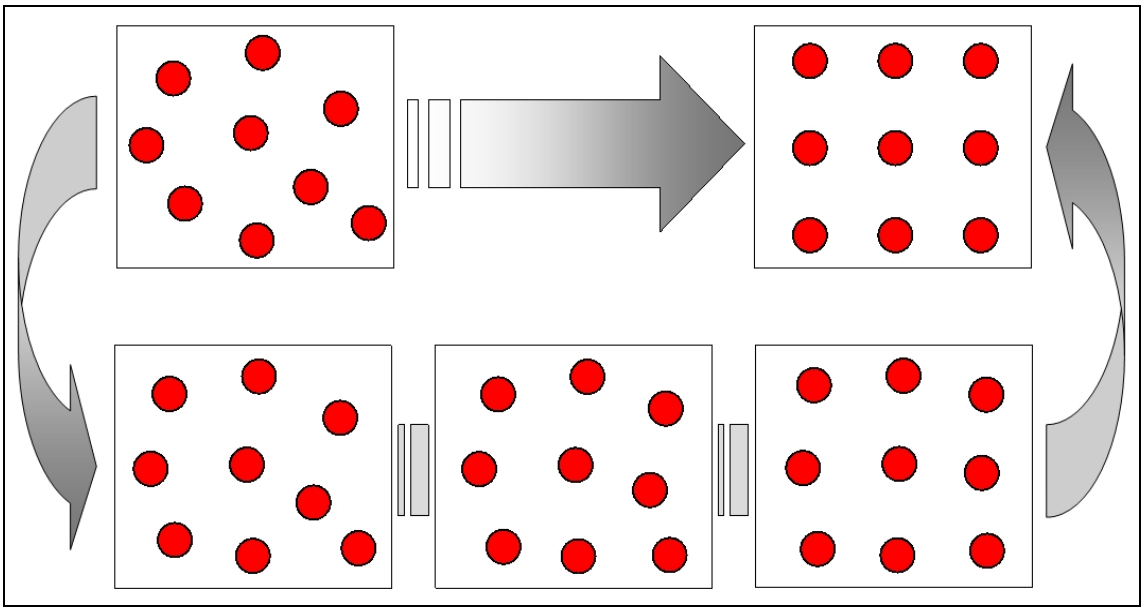

Figure 1.1 Comparaison entre une transition soudaine (en haut) et une transition animée (en bas)

Plusieurs études ont été menées afin de démontrer si l'utilisation des animations apporte un réel avantage à la manipulation d'une interface graphique ou à la visualisation de données. Parmi celles-ci, Robertson *et al.* (2002) ont trouvé que les participants préfèrent des animations pour visualiser des changements entre les points de vue d'un ensemble de hiérarchies en plus de réduire le temps de complétion des tâches par rapport à une transition soudaine. Klein et Bederson (2005) démontrent que le défilement vertical animé dans un document contenant du texte augmente l'efficacité des tâches de lecture et de repérage d'éléments précis comparativement à un défilement direct non animé. Selon Heer et Robertson (2007), il est plus facile d'identifier le déplacement d'un élément et d'estimer le pourcentage de changement d'une valeur entre deux dispositions différentes d'un graphique de données statistiques à l'aide d'une transition animée plutôt qu'une transition soudaine. Finalement, Shanmugasundaram *et al.* (2007) ont montré que les utilisateurs ont plus de facilité à percevoir les connexions entre les éléments d'un diagramme nœuds-liens quand des

transformations sont appliquées avec une transition fluide. Leurs résultats indiquent aussi qu'il est ainsi plus facile de maintenir mentalement les informations sur la structure d'un graphe à travers divers changements de points de vue tout au long de l'animation.

Toutefois, d'autres chercheurs avancent une vision différente sur la réelle pertinence des animations et leurs études viennent appuyer leurs propos. Bederson et Boltman (1999) ont élaboré une expérimentation à l'aide d'un navigateur d'arbre généalogique pour vérifier si les animations aident à construire une image mentale et spatiale des informations. Les résultats ont démontré un avantage significatif pour l'utilisation des animations seulement pour le taux d'erreur lors de la reconstruction de l'arbre, mais pas pour le temps de complétion de la tâche. De plus, l'analyse statistique n'a pas relevé d'avantages pour des tâches de navigation et d'exploration. Selon Tversky et Betrancourt (2002), les animations d'événements successifs ou qui montrent l'évolution des données à travers le temps peuvent être inefficaces parce qu'elles violent des principes fondamentaux de l'infographie au niveau de la perception et de la conception. Souvent, les animations sont trop complexes ou les transitions sont trop rapides pour permettre à l'utilisateur de bien comprendre les changements d'information. Par exemple, pour montrer l'évolution de l'inflation économique à travers le temps, il est mieux d'utiliser un graphique statique à lignes brisées au lieu d'une animation. Donc, lorsqu'il est possible d'utiliser une représentation statique pour montrer les mêmes informations, cette dernière est préférable. Ce point de vue contraire est appuyé par Robertson *et al.* (2008) avec une étude portant sur l'efficacité des animations pour la visualisation du mouvement dans les données. Ses résultats énoncent que les participants ont été confondus par les animations parce qu'il y avait trop de données ou bien parce que les items à l'écran ne se déplaçaient pas de façon synchronisée. De plus, les résultats supposent que, pour être efficace, l'animation doit être rejouée plusieurs fois pour que l'utilisateur découvre où porter son attention. Bien que les animations soient plus intéressantes et amusantes au niveau visuel, les représentations statiques des mêmes informations se sont avérées plus efficaces lorsqu'il est temps de les analyser. Il faut mentionner que les travaux antérieurs de Tversky et Betrancourt (2002) et de Robertson *et al.* (2008) concernent des cas où l'animation est utilisée pour présenter des

informations supplémentaires aux vues statiques. Dans le cas où l'utilisateur (ou le logiciel) tient simplement à faire la transition d'une vue vers une autre, l'utilisation d'une animation reste préférable par rapport à une transition soudaine.

En résumé, pour qu'une animation soit utile, sa conception doit être simple tout en tenant compte de la pertinence du contexte ou du modèle mental de l'utilisateur. De plus, les transitions animées devraient être fluides et interactives (Gonzalez, 1996). L'utilisation judicieuse de l'interactivité permettant à l'utilisateur de contrôler la vitesse de l'animation, par exemple, pourrait permettre de palier aux désavantages des animations et même de rendre bénéfique l'utilisation de celles-ci (Tversky et Betrancourt, 2002).

1.2 La conception des animations

Dans le but de concevoir des transitions animées pertinentes, efficaces et adaptées aux interfaces graphiques, il est nécessaire de comprendre les principes de base que respectent les animateurs lors de la réalisation des dessins animés (Chang et Ungar, 1993). Premièrement, le principe de solidité dicte que tout objet animé doit se déplacer et agir comme s'il était un objet physique réel en trois dimensions. Elle permet aussi à un objet de se distinguer de sa simple représentation imagée à l'écran en étant tangible et en ayant un comportement qui lui est propre. Le deuxième principe est que l'exagération d'un mouvement ou d'une caractéristique particulière d'un personnage ou d'un objet tend à le rendre plus réaliste. Cette technique est aussi utilisée pour mettre l'emphase sur certains points précis dans le but d'attirer l'attention du public ou bien pour rendre des détails plus visibles. Finalement, le renforcement de l'illusion de la réalité regroupe toutes les techniques permettant à l'animation d'être altérée dans le but de la rendre plus réaliste (Chang et Ungar, 1993). La prochaine section présente quelques-unes de ces techniques.

1.2.1 Quelques techniques de base

Les ralentissements en début et en fin de mouvement (*slow-in* et *slow-out*) tentent de reproduire le déplacement réel d'un objet physique qui a une masse et une inertie. Le mouvement sera alors plus lent au début avant d'atteindre une vitesse maximale puis ralentir jusqu'à un arrêt complet. Cette technique permet de minimiser l'effet de surprise lors du début de l'animation et aussi de préparer l'utilisateur à un déplacement plus rapide par la suite (Chang et Ungar, 1993). Utilisée par les animateurs, cette technique s'est répandue chez les chercheurs dont Chang et Ungar (1993), Bederson et Boltman (1999), Yee *et al.* (2001) ainsi que Heer et Robertson (2007) pour développer des animations par ordinateur dans plusieurs prototypes de recherche. De plus, Dragicevic *et al.* (2011) comparent différentes distorsions temporelles pour les animations dont une vitesse constante pour toute la durée de l'animation, un ralentissement en début et en fin de mouvement, une accélération en début et en fin de mouvement ainsi qu'une vitesse adaptée en fonction des parties les plus complexes de l'animation. Le ralentissement en début et en fin de mouvement s'avère le plus efficace parce qu'il maximise la prédiction du mouvement.

Un autre principe qui favorise le renforcement de la réalité est le mouvement des objets selon des trajectoires arquées. Lorsqu'un objet se déplace de façon non interactive selon ses propres forces, il a tendance à adopter un mouvement légèrement curviligne plutôt que rectiligne (Chang et Ungar, 1993). Un exemple concret de ce principe est utilisé par Yee *et al.* (2001) pour l'animation de nœuds dans un graphe de données avec une disposition circulaire sur plusieurs orbites. Au lieu de faire une simple translation linéaire entre deux positions, ils utilisent une transition arquée suivant la trajectoire de l'orbite de chaque nœud. Ainsi, le mouvement est non seulement plus naturel, mais cela empêche aussi les deux trajectoires de se croiser apportant donc moins de confusion lors de la transition et moins d'occlusion entre les éléments. Pour y arriver, les chercheurs utilisent des coordonnées polaires plutôt que des coordonnées rectangulaires étant donné que la disposition du graphe est circulaire.

Parmi les autres techniques, Chang et Ungar (1993) mentionnent le flou directionnel, l'anticipation du mouvement et le suivi en fin de déplacement.

1.3 Les transitions visuelles différenciées et les transitions par étapes

Le but des animations dans le contexte de la visualisation de données est de permettre à l'utilisateur de mieux comprendre les changements entre des dispositions différentes d'un même ensemble ou encore de lui faire connaître les modifications qui ont été apportées. Les chercheurs combinent plusieurs principes ou enchaînements d'animations adaptées au contexte et à l'information présentée dans le but de créer des transitions fluides plus complètes.

1.3.1 Les transitions visuelles différenciées

En plus de répondre aux principes de base des animations, les transitions visuelles différenciées ont le mandat de présenter de l'information. En effet, elles regroupent dans une même séquence des éléments propres au mouvement entre les éléments tout en rajoutant une dimension adaptée aux données.

Pour concevoir des transitions différenciées efficaces, Schlienger *et al.* (2006) proposent un guide comportant quelques éléments à suivre. Ils énoncent qu'il est nécessaire de privilégier la continuité, de jouer sur les paramètres des trajectoires (vitesse, accélération, oscillations et déformations), d'exploiter les apparitions et les disparitions, de bien combiner les transitions entre elles et de favoriser le travail avec un animateur. Ils conseillent aussi de rester honnête pour éviter de confondre une animation avec une évolution réelle du système et ainsi venir biaiser l'interprétation de l'utilisateur.

Pour démontrer visuellement ce concept, deux exemples de transitions différenciées ont été implémentés : les Graviticônes et les Mobilistes. Le premier exemple est utilisé dans le contexte de la copie ou du téléchargement d'un fichier dans une interface de bureau virtuel. L'animation, en plus de montrer la progression de la copie, évoque également la taille du fichier en question et la vitesse de transfert. De plus, elle permet de suivre le déplacement du fichier depuis son endroit d'origine jusqu'à sa destination finale. Ainsi, pour chaque copie, l'animation utilisera des déformations différentes et appropriées en fonction des informations du système (taille du fichier, emplacement, vitesse initiale et vitesse finale). De leur côté, les Mobilistes, utilisées dans la gestion du transport d'un aéroport, ont pour but de notifier le contrôleur lorsqu'une modification se produit dans l'horaire des déplacements prévus afin qu'il puisse adapter ses directives. Ainsi, les animations utilisées prennent en compte le type de véhicule impliqué dans la modification afin de rendre une trajectoire cohérente aux objets qui se sont déplacés dans l'interface. L'avion va donc survoler la liste par la droite, un autobus va faire marche arrière vers la gauche puis s'insérer au bon endroit tandis qu'un véhicule de service effectuera un demi-tour vers la gauche. Si plusieurs changements surviennent en même temps, ces transitions différentiées pourront être juxtaposées pour se produire l'une après l'autre. La Figure 1.2 présente la séquence d'animation d'une inversion de deux éléments de la liste représentés par un avion et un véhicule de service.

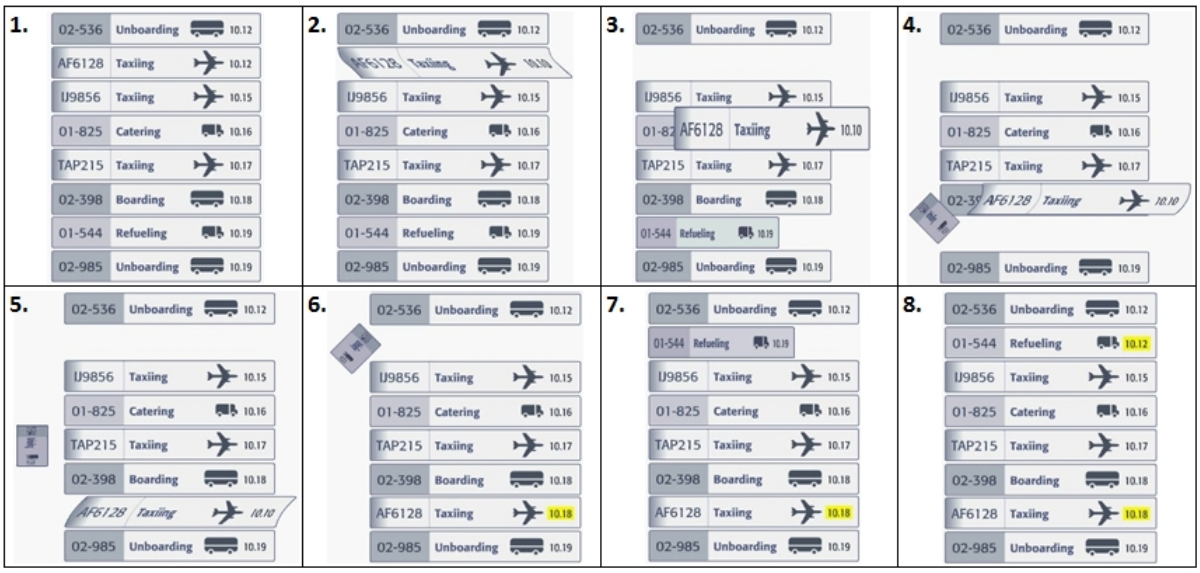

Figure 1.2 Séquence d'animation d'une inversion d'éléments entre un avion et un véhicule de service dans une Mobiliste
Adaptée de Céline Schlienger et Pierre Dragicevic

1.3.2 Les transitions par étapes

Les transitions animées par étapes combinent plusieurs animations subséquentes présentant chacune des caractéristiques particulières dans le but de construire une transition animée complète. Il existe plusieurs façons de procéder selon le contexte ou la disposition des informations en plus de la nature des changements qui sont appliqués. La prochaine section transporte les transitions par étapes dans le contexte de la visualisation d'arborescences et de données statistiques.

1.3.2.1 Les transitions animées pour la visualisation des arborescences

Les arborescences peuvent adopter différents types de présentations et contenir un nombre important de données. Jürgensmann et Schulz (2010) répertorient plus de 100 dispositions de graphes et d'arborescences et les classent selon diverses catégories, dont le nombre de dimensions (2, 3 ou hybride), l'alignement (radial, parallèle à des axes ou libre) et la

représentation (explicite, implicite ou hybride). Bien que certaines techniques facilitent la visualisation des informations, il peut s'avérer très compliqué de bien suivre l'évolution de l'état d'un graphe pendant des changements. Afin de montrer des modifications ou encore appliquer un changement de disposition, plusieurs chercheurs adoptent les animations.

Les animations linéaires, où une interpolation directe est faite d'une disposition du graphe vers une autre, peuvent être une technique facile et rapide à concevoir. Teoh et Ma (2002) utilisent une seule transition animée pour montrer le changement de focus d'un nœud vers un autre. Cependant, les nombreux mouvements et l'occlusion entre plusieurs éléments rendent l'animation difficile à assimiler. Pour sa part, Yee *et al.* (2001) ont développé une transition linéaire adaptée à un système de coordonnées polaires dans le but de minimiser l'occlusion. Bien que cette amélioration soit bénéfique pour la visualisation et l'exploration d'un graphe dans son contexte, il est souvent nécessaire de mieux découper les transitions en étapes plus simples.

La philosophie générale des chercheurs lors de la conception des animations par étapes est de faire disparaître en premier les éléments qui ne se retrouvent pas dans la disposition finale du graphe, d'appliquer des translations pour repositionner les éléments et finalement de faire apparaître les nouveaux qui se sont ajoutés ou qui sont maintenant visibles. Plaisant *et al.* (2002) proposent une animation qui élague d'abord les branches de l'arbre qui se trouveraient cachées par les nouvelles à apparaître. Par la suite, ils procèdent à une translation de l'arbre vers le centre de l'écran pour faire de la place pour les nouvelles branches. Puis, ces dernières apparaissent dans l'espace qui était prévu. McGuffin et Balakrishnan (2005) adoptent la même approche pour la visualisation d'arbres généalogiques. Dans leur animation pour visualiser une hiérarchie de fichiers sur un disque dur (voir Figure 1.3), McGuffin *et al.* (2004) rajoutent une étape supplémentaire entre la disparition (étape 2) et la translation de nœuds (étapes 4 et 5) afin de fermer la structure d'un dossier (étape 3) et font de même plus tard pour l'ouverture d'un dossier avant d'en afficher le contenu (étape 6). Cela porte donc l'animation à cinq étapes distinctes (étapes 2 à 6). Lee

et al. (2006) utilisent une animation en trois étapes avec d'abord une translation de certaines parties de l'arbre pour créer un espace pour l'agencement des nouveaux nœuds puis une translation des nœuds présents pour les repositionner dans la structure existante. En dernier, les nouveaux nœuds sont introduits par une translation à partir d'une zone de prévisualisation vers leur position finale.

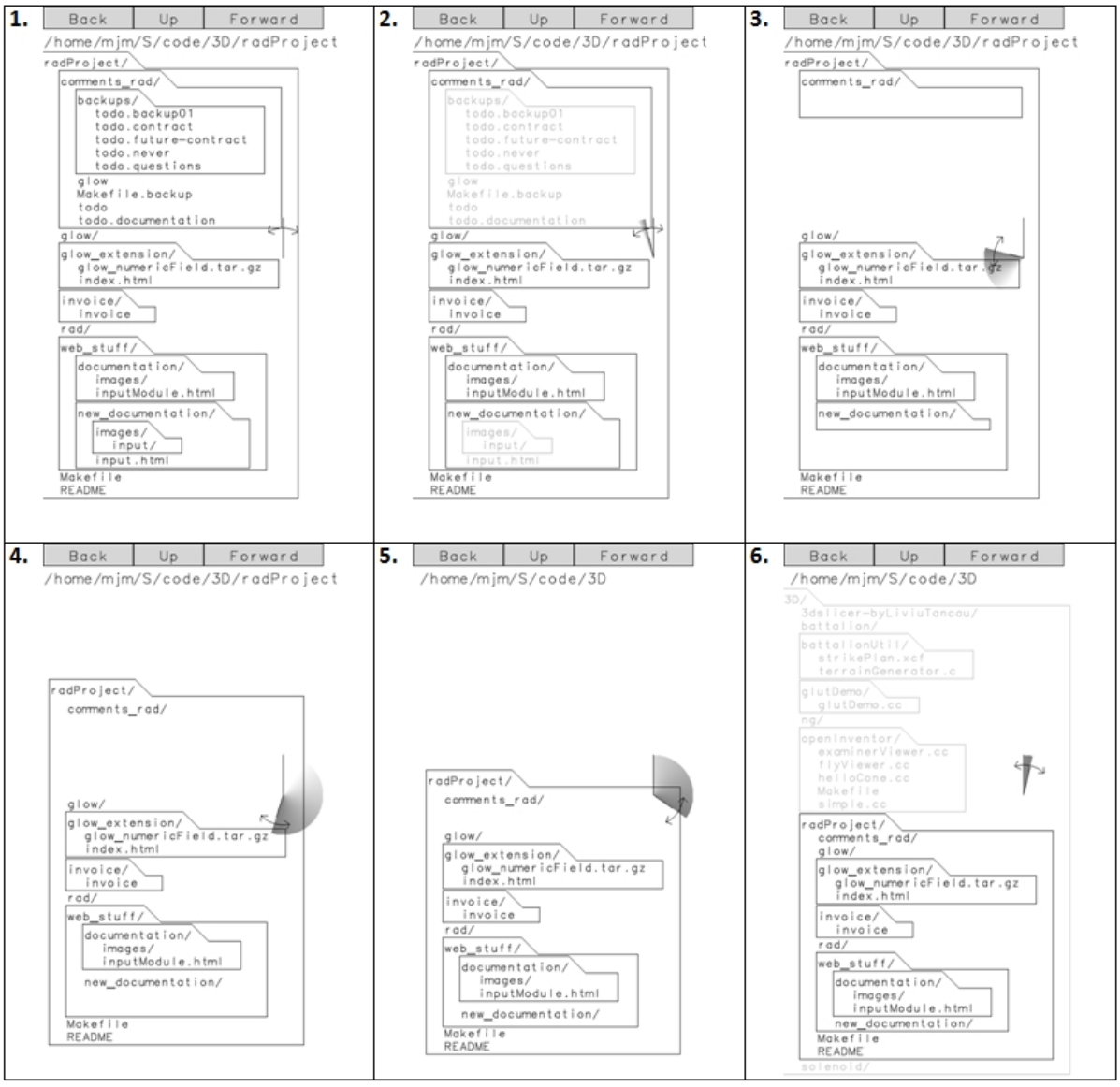

Figure 1.3 Animation par étapes, contrôlée par un mouvement circulaire de la souris, pour la visualisation d'une hiérarchie de fichiers sur le disque dur d'un ordinateur
Adaptée de Michael McGuffin

1.3.2.2 Les transitions animées pour la visualisation des graphiques statistiques

Dans le but d'étudier le processus de création des animations par étapes et leurs éventuels avantages, Heer et Robertson (2007) ont bâti un prototype dans lequel un utilisateur peut visualiser des transitions entre différentes représentations de graphiques réalisés à partir de données statistiques. Avant de procéder, les chercheurs proposent une taxonomie pour identifier les types de transitions à l'aide de critères comme la transformation de la vue (translation et zoom), la transformation de l'espace (changement d'échelle et déformation), le filtrage des données, l'ordonnancement, le temps, le changement de visualisation et le changement du modèle de données. De plus, ils énoncent deux principes à considérer lors du design d'une transition : la congruence et la compréhension. La congruence consiste à maintenir une représentation valide des données à travers la transition dans le but d'éviter l'ambiguïté. Pour sa part, la compréhension dicte d'utiliser des transitions simples en groupant celles qui sont similaires tout en évitant l'occlusion des données et en maximisant la prédictibilité. Enfin, elle propose d'utiliser des transitions par étapes pour décomposer une transition trop longue ou complexe.

Afin de mettre en pratique cette série de principes, deux expériences ont été réalisées concernant le suivi visuel d'objets et l'estimation du changement d'une valeur statistique. La première présente à l'utilisateur une représentation d'un graphique statistique avec deux cibles mises en évidence. Après une durée de 3 secondes, une transition soudaine (immédiate) ou une transition animée de 1,5 seconde transforme le graphique vers une autre disposition par exemple d'un nuage de points vers un diagramme à bandes verticales. Les transitions animées sont soient linéaires (directes) ou par étapes (voir Figure 1.4). Par la suite, l'écran est masqué en devenant complètement noir et le participant doit cliquer approximativement où il pense que les deux cibles se sont déplacées. La deuxième étude demande à l'utilisateur de suivre une cible unique et d'estimer le pourcentage de changement de la donnée qu'elle représente toujours à travers les trois types de transitions entre des graphiques. La méthodologie est la même que la première étude sauf pour la dernière étape.

Après avoir masqué l'écran, une série de boutons apparaît pour permettre d'identifier le pourcentage de changement de la valeur ciblée.

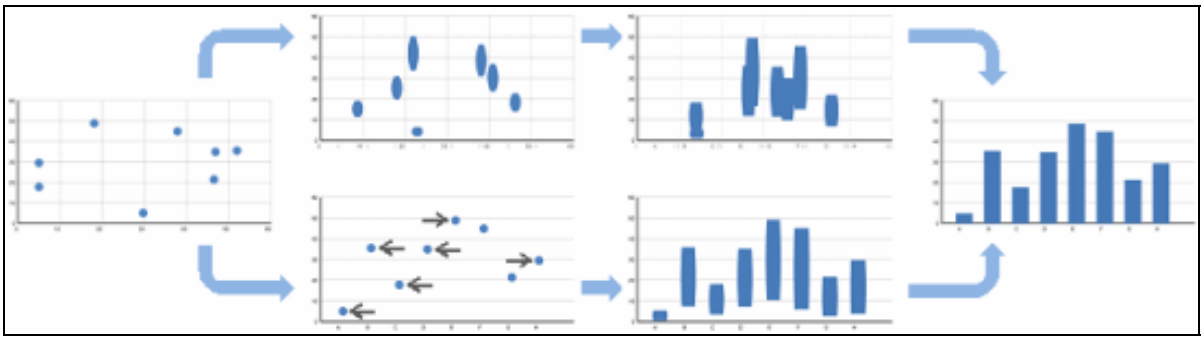

Figure 1.4 Séquence de l'animation linéaire (en haut) et par étapes (en bas)
d'un changement entre deux vues d'un graphique statistique
Tirée de Stanford Visualization Group (2007) avec l'autorisation de Jeffrey Heer

Dans les deux cas, les résultats démontrent une diminution significative du pourcentage d'erreur en utilisant des animations (linéaires ou par étapes) plutôt que des transitions statiques. Les résultats obtenus varient selon les types de graphiques utilisés et la grosseur des objets à suivre dans certains cas. Cependant, seulement une condition sur les huit étudiées dans la première expérimentation démontre un avantage significatif pour les animations par étapes. De plus, les animations linéaires se sont montrées plus efficaces de façon significative pour deux conditions face à l'animation par étapes. Toutefois, les chercheurs en viennent à des recommandations sur la création des animations par étapes. La conception des animations doit être basée sur la réduction de l'occlusion des éléments et les étapes doivent être simples (Heer et Robertson, 2007).

1.3.3 Opportunités pour des recherches futures

Un défi actuel dans l'état de l'art est de rendre les animations plus faciles à comprendre lorsqu'il y a plusieurs sortes de changements en même temps ou lorsqu'un grand nombre d'éléments change pendant la transition. Autrement dit, il manque des approches pour rendre les animations plus extensibles (*scalable*). Les transitions visuelles différenciées de

Schlienger *et al.* (2006) réussissent à montrer des informations supplémentaires pendant une transition en présentant différents types de changements visuels. Cependant, ces types d'animations ont besoin d'être conçus par un humain à l'avance en utilisant des métaphores qui sont particulières au genre de données qui sera montré, et ne semblent donc pas particulièrement génériques ni extensibles.

Le découpage d'une transition en étapes pourrait être une façon de rendre les transitions plus compréhensibles et donc extensibles. Mais, l'étude de Heer et Robertson (2007) a seulement trouvé un faible avantage pour les animations par étapes comparativement aux animations linéaires, malgré que la majorité des participants ait préféré les animations par étapes. Il se pourrait que le changement complet de disposition entre deux graphiques de données statistiques dans leur étude reste difficile à assimiler puisque beaucoup d'éléments bougent en même temps, rendant ainsi difficile la création d'une image mentale et la compréhension du contexte. Il faut noter que les données statistiques dont il était question n'ont pas de liens entre elles contrairement à des données sous la forme d'un graphe ou d'une arborescence où les éléments sont reliés. Ceci amène une nouvelle possibilité : dans le cas de visualisations de données avec des liens, est-ce qu'il serait possible de mesurer plus de différences entre les animations linéaires et les animations par étapes? Aussi, est-ce qu'il pourrait y avoir des façons alternatives de découper une transition en étapes en fonction de la structure des liens entre les éléments, rendant ainsi la transition plus facile à comprendre et donc plus extensible? Ces questions seront étudiées dans les prochains chapitres.

1.4 Les *widgets* et le contrôle des animations

Avec n'importe quelle transition animée, il existe un compromis entre le temps et la compréhensibilité. Si la transition est trop lente, le coût en temps sera trop grand et l'utilisateur devra attendre à la fin avant de continuer ses interactions. Si la transition est trop rapide et que l'utilisateur s'intéresse à surveiller les détails des changements, il pourra manquer de temps pour tout remarquer. Une façon de permettre à l'utilisateur de maîtriser ce

compromis est de lui fournir un ou plusieurs contrôles interactifs lui permettant d'interagir avec une transition animée en modifiant sa vitesse ou encore son sens de progression (avant ou arrière). L'utilisateur pourrait ainsi rejouer certaines parties d'une séquence dans le but de mieux comprendre les changements à l'écran en repérant des détails supplémentaires qui peuvent être subtils à observer directement, ou encore laisser jouer une transition rapidement lorsque les détails ne l'intéressent pas.

Pour agir ainsi, l'utilisateur a besoin de composants d'interface nommés *widgets* qui procurent une manipulation directe des données à l'écran. De façon générale, les *widgets* sont des objets graphiques composant une interface utilisateur et qui peuvent prendre la forme d'icônes, de boutons, de listes, d'onglets, de curseurs, de zones de texte ou de menus. Pour contrôler le progrès d'une animation, un curseur défilant sur une barre linéaire est souvent utilisé, mais il est aussi possible de simplement offrir un ou plusieurs boutons pour permettre à l'utilisateur de jouer, reculer ou avancer une animation à vitesse fixe.

Les *widgets* utilisés pour contrôler une transition peuvent être situés à l'intérieur d'un panneau de *widgets* dans la périphérie de la fenêtre principale, mais il pourrait être plus judicieux d'accéder directement à des *widgets* contextuels (ou *popup widgets*) qui s'affichent de façon dynamique près du curseur de la souris. Ces *widgets* ou menus contextuels apparaissent seulement à la demande de l'utilisateur et ne requièrent pas d'espace à l'écran lorsqu'ils ne sont pas affichés. De plus, en les invoquant en maintenant un bouton de la souris enfoncé par exemple, cela procure un retour kinesthésique qui évite les erreurs de mode causées par une mauvaise interprétation du mode d'utilisation en cours dans le logiciel par l'utilisateur (Sellen *et al.*, 1992). Par exemple, l'utilisateur pourrait vouloir contrôler l'animation d'une transition lorsque c'est impossible, soit pendant que le logiciel est en mode de sélection des données ou l'inverse. Également, étant donné que le ou les *widgets* s'affichent près du curseur de la souris, il n'est pas nécessaire de faire un aller-retour entre une barre de menus ou un panneau d'outils fixe et les données à contrôler. De plus, les contrôles d'animation peuvent être intégrés à l'intérieur d'un menu contextuel permettant à

l'utilisateur de non seulement contrôler la progression d'une transition, mais aussi de choisir le type de transition ou encore d'accéder à d'autres options. Parmi les différents types de menus contextuels, le menu radial procure un avantage par rapport à un menu linéaire traditionnel, car il permet la sélection d'options de façon gestuelle et rapide (Kurtenbach et Buxton, 1993). Grâce à leurs nombreux avantages, cette section va se concentrer principalement sur les menus et autres *widgets* contextuels qui apparaissent à la demande de l'utilisateur ou encore qui permettent de contrôler des variables, comme le temps dans une animation, de façon discrète ou continue.

1.4.1 Les menus contextuels pour le contrôle continu

Bailly *et al.* (2007) présentent un inventaire de plus de 20 types de menus contextuels qu'ils soient linéaires ou circulaires. Parmi ceux-ci, certains permettent de faire la sélection d'un paramètre puis d'en faire le contrôle dans un même geste. Bien qu'il soit conçu pour un autre contexte, le principe de l'ajustement continu d'une valeur peut être assimilé au contrôle d'une animation, où le paramètre à contrôler est le temps. La prochaine section fait un survol des menus les plus pertinents qui permettent ce genre d'interaction.

Pook *et al.* (2000) proposent le « Control Menu » qui est formé d'items disposés en cercle autour du centre où est le curseur de la souris (voir Figure 1.5). Pour faire la sélection d'une opération comme un zoom, l'utilisateur dirige le curseur vers l'élément en question. Quand il a passé la zone d'activation de l'item, le menu disparaît et l'utilisateur peut alors contrôler de façon continue et linéaire la valeur du zoom jusqu'à ce qu'il relâche le bouton de la souris.

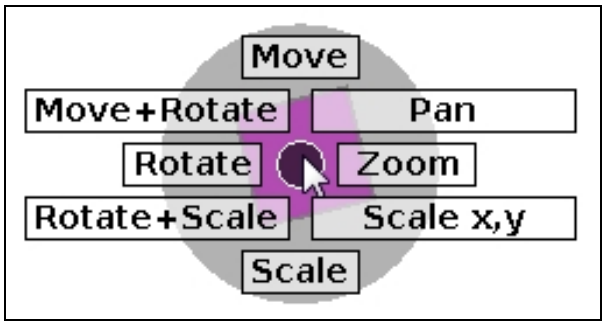

Figure 1.5 Exemple d'un « Control Menu » à l'intérieur d'un logiciel de dessin
Adaptée de Michael McGuffin

De son côté, le « Flow Menu » (Guimbretière et Winograd, 2000) accomplit sensiblement la même tâche (voir Figure 1.6, étapes 1 et 2) sauf qu'il faut repasser par le centre (3 et 4) pour entrer dans le mode de contrôle continu après avoir choisi l'opération à effectuer ou le paramètre à modifier. Cependant, pour ce dernier, le contrôle continu se fait de façon circulaire autour du centre un peu à la façon d'un bouton de l'intensité du volume sur une radio (5). De plus, les deux menus donnent la possibilité d'avoir des sous-menus et de les utiliser dans deux modes : novice et expert. Dans le mode novice, le menu s'affiche et l'utilisateur prend le temps de lire les items avant de faire une sélection. Dans le mode expert, l'utilisateur qui se souvient de l'emplacement des items peut faire une sélection immédiate sans même attendre l'apparition du menu. Cela permet donc d'effectuer des gestes de souris et de terminer une tâche plus rapidement.

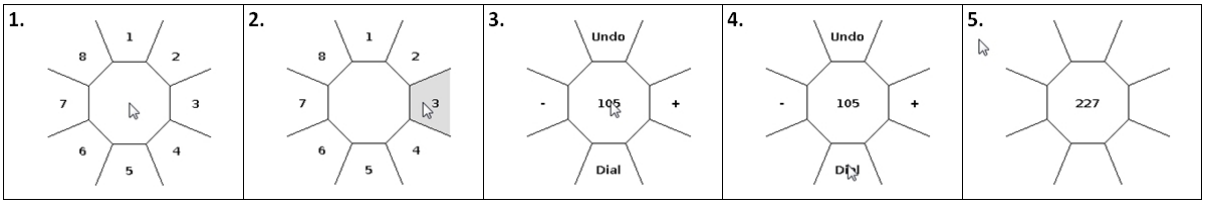

Figure 1.6 Séquence d'utilisation d'un « Flow Menu » pour la sélection et la modification d'un paramètre
Adaptée de Michael McGuffin

Également, McGuffin *et al.* (2002) s'inspirent du « Control Menu » avec une série d'items en cercle et une zone d'activation pour sélectionner une opération (voir Figure 1.7, étapes 1 et 2). Toutefois, une fois la zone d'activation dépassée, un curseur vertical, semblable à un potentiomètre linéaire sur une table de mixage, apparaît pour faire le contrôle continu d'une valeur (3). L'utilisateur peut alors monter ou descendre avec le curseur pour modifier la valeur (4). Si un mouvement perpendiculaire au curseur est appliqué, une série de boutons s'affiche également pour effectuer un contrôle plus fin par échelons ou encore pour annuler la modification en cours (5).

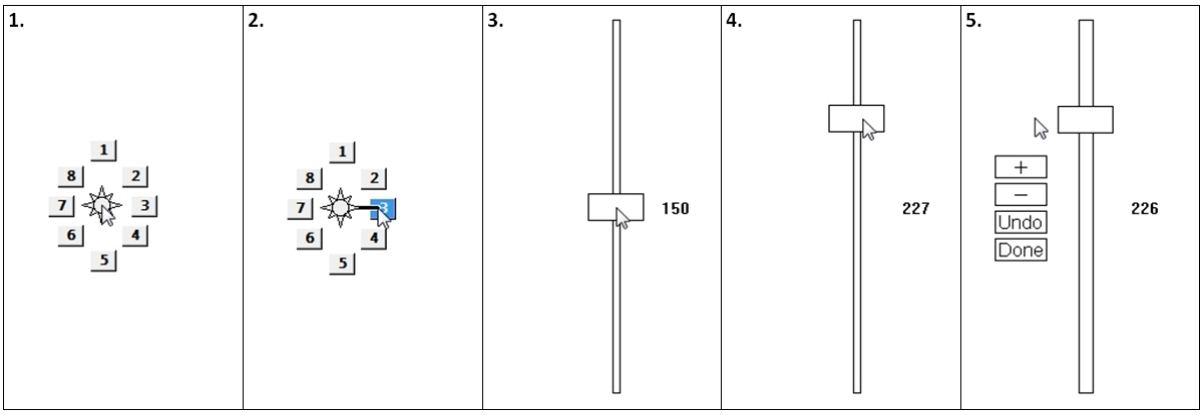

Figure 1.7 Séquence d'utilisation d'un « FaST Slider » pour la sélection et la modification d'un paramètre
Adaptée de Michael McGuffin

Comme ils sont génériques et peuvent servir à contrôler n'importe quels paramètres continus, les trois menus ci-dessus pourraient aussi servir à contrôles des transitions animées. L'option sélectionnée avec un mouvement radial pourrait correspondre à un type de transition quelconque puis le glissement continu du curseur servirait à contrôler le temps, c'est-à-dire le progrès de l'animation.

1.4.2 Les *widgets* conçus pour contrôler des animations

La technique classique pour le contrôle d'une animation est semblable à celle des séquences vidéo qui possède un bouton pour démarrer ou faire une pause, un pour avancer rapidement et un autre pour revenir en arrière. Elle peut s'accompagner aussi d'un curseur qui permet de changer la vitesse d'animation et de faire un contrôle continu interactif avant et arrière comme utilisent Robertson *et al.* (2008) dans leur prototype. Cependant, les *widgets* expressément conçus pour les animations sont mieux adaptés à leur contexte et permettent à l'utilisateur de se concentrer sur sa tâche.

Contrairement aux menus de la section précédente qui demandent tous un contrôle continu du temps via un glissement de curseur, McGuffin *et al.* (2003) utilisent une méthode de contrôle discrète pour débuter des animations à vitesse constante dans leur prototype de visualisation de données volumiques (voir Figure 1.8). Ils utilisent un menu radial pour faire la sélection d'opérations qui démarrent l'animation par un geste rapide vers la droite pour avancer (1 et 2) et vers la gauche pour reculer. Le même geste peut être répété rapidement plusieurs fois pour démarrer les animations en parallèle. Par exemple, chaque animation peut montrer l'épluchage d'une couche de tissu du crâne. Ainsi, un lancement de plusieurs animations en parallèle aurait comme effet de montrer plusieurs couches qui se font éplucher en même temps.

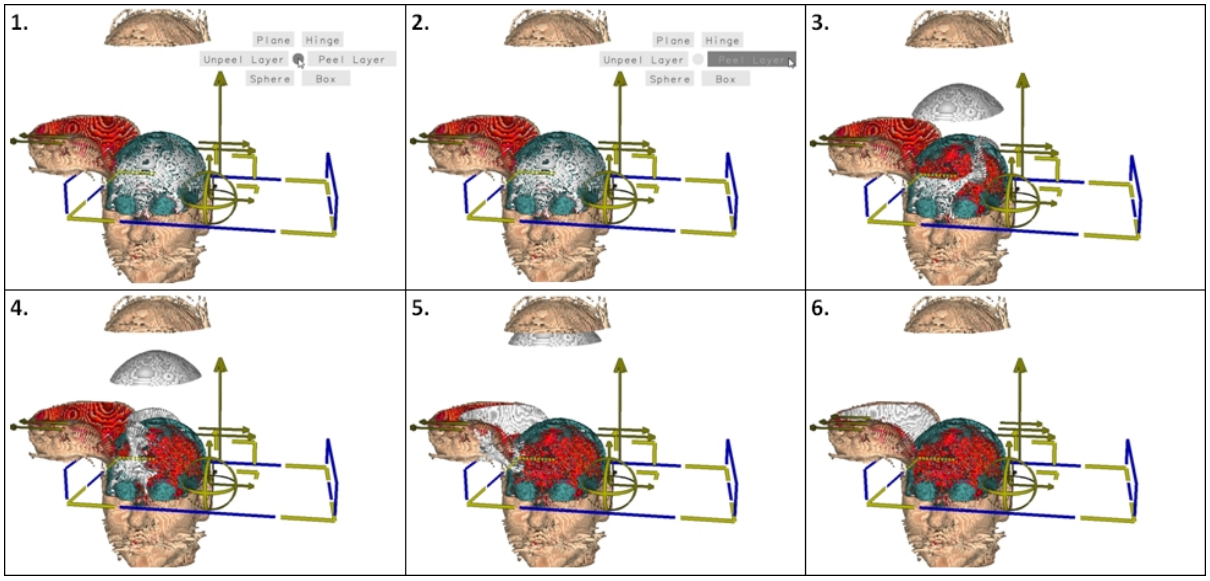

Figure 1.8 Démarrage d'une animation de façon discrète dans un prototype de visualisation de données volumiques
Adaptée de Michael McGuffin

Elmqvist *et al.* (2008) proposent plusieurs techniques d'interaction pour contrôler un changement de point de vue en trois dimensions entre différents nuages de points. Ces techniques s'opèrent à l'intérieur d'une matrice de nuages de points qui montre un aperçu des combinaisons des multiples dimensions des données. Pour passer d'une vue à l'autre, l'utilisateur peut simplement utiliser les flèches du clavier pour naviguer dans la matrice et ainsi démarrer les animations de façon discrète. Il peut aussi contrôler de façon continue la transition en maintenant le bouton gauche de la souris enfoncé tout en glissant le curseur entre les cases de la matrice représentant les deux points de vue adjacents. Une autre approche permet de jouer une animation successive de plusieurs points de vue selon un chemin particulier dans la matrice. Il peut être choisi de façon automatique par le logiciel qui trouve le chemin le plus court entre deux vues ou par l'utilisateur qui peut dessiner le chemin à parcourir lors de la transition. Finalement, il est également possible de sauter directement d'un point de vue à un autre sans passer par un chemin intermédiaire en sélectionnant deux cases distantes.

Dans un prototype de visualisation d'arbres généalogiques, McGuffin et Balakrishnan (2005) utilisent un nouveau *widget* linéaire pour animer l'ouverture ou la fermeture des différents niveaux de l'arborescence (voir Figure 1.9). À l'aide d'un menu contextuel en forme de sablier, il est d'abord possible de sélectionner l'arbre des ancêtres d'un nœud en glissant le curseur de la souris vers le haut ou l'arbre des descendants vers le bas (1 et 2). Par la suite, l'utilisateur peut ouvrir ou fermer l'arbre sélectionné (3 à 6) niveau par niveau en glissant le curseur de la souris vers le haut ou vers le bas de façon continue. En plus de contrôler l'animation, la grandeur du *widget* s'adapte au nombre d'arborescences qu'il est possible d'ouvrir ou de fermer et montre l'avancement de l'animation.

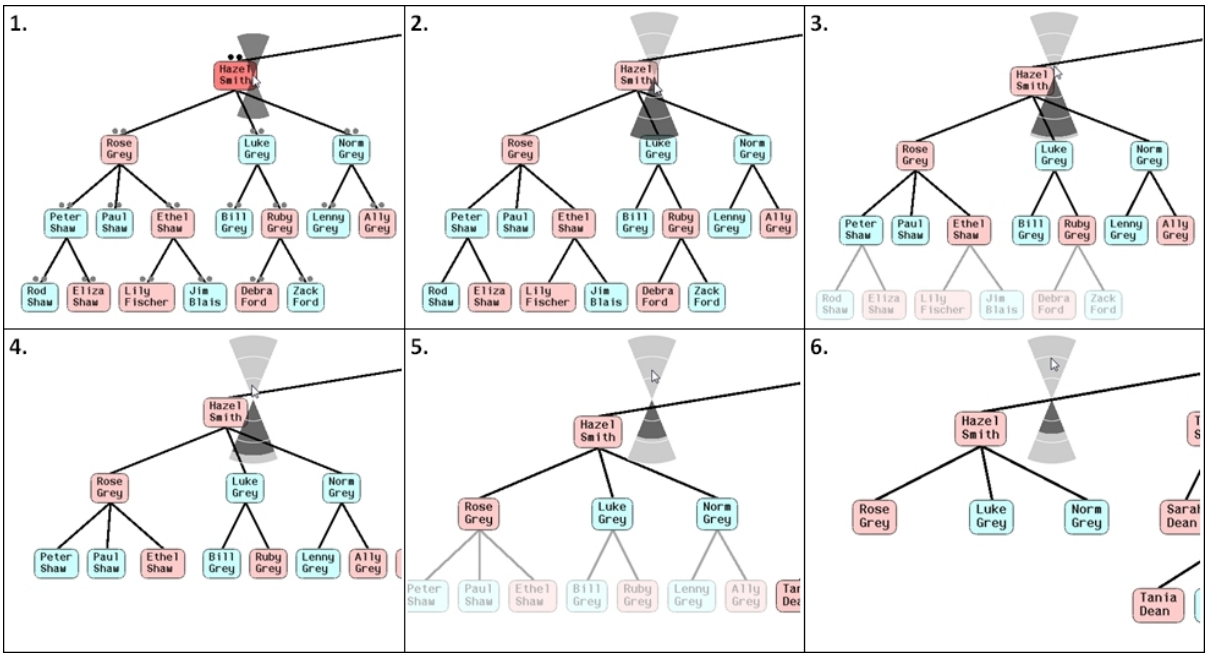

Figure 1.9 Séquence du contrôle continu d'une animation avec le *widget* du prototype de visualisation d'un arbre généalogique
Adaptée de Michael McGuffin

Finalement, McGuffin *et al.* (2004) proposent un *widget* contextuel circulaire pour animer les transitions dans un prototype de visualisation d'une arborescence de fichiers sur le disque dur (voir Figure 1.3). Quand le menu apparaît, l'utilisateur peut avancer ou reculer l'animation avec un mouvement circulaire du curseur de la souris autour du centre en sens horaire ou

antihoraire. Un tour complet représente une transition et il est possible de poursuivre le mouvement pour avancer ou reculer dans l'historique des transitions déjà effectuées. Le *widget* indique l'état d'avancement de l'animation à l'intérieur d'un seul tour, mais pas pour l'historique complet des animations sauf si le début ou la fin est atteint. Il est donc impossible pour l'utilisateur de connaître le nombre de transitions dans l'historique et la position de l'animation en cours par rapport aux autres. Également, pour savoir dans quel niveau de l'arborescence l'animation courante s'applique, il faut regarder ailleurs à l'écran pour voir le chemin complet d'un dossier sur le disque. Il peut alors être difficile pour l'utilisateur de savoir le nombre de tours à effectuer pour atteindre un état précédent désiré ou encore un niveau en particulier dans l'arborescence. En plus du *widget*, l'interface possède des boutons semblables à un navigateur Internet pour avancer et reculer dans l'historique des transitions ou encore un bouton pour accéder au niveau supérieur de l'arbre. Ces boutons déclenchent de manière discrète une animation à vitesse constante.

Une des différences intéressantes entre les méthodes de contrôle ci-dessus est que certaines utilisent une entrée discrète, comme les sélections d'options dans le menu de la Figure 1.8 ou encore l'utilisation de boutons, pour lancer la transition animée avec une vitesse fixe, tandis que d'autres utilisent une entrée continue avec un glissement pour faire progresser l'animation. Chacune de ces approches possède des avantages et des inconvénients. L'entrée discrète est plus simple et peut être lancée plus rapidement, mais empêche l'utilisateur de ralentir ou d'accélérer la transition contrairement à l'entrée continue. De plus, pour le contrôle continu, le glissement peut se faire de façon linéaire ou circulaire. Dans le premier cas, il permet à l'utilisateur de se rendre plus rapidement au début ou à la fin d'une transition ou une série de transitions avec un seul glissement. Toutefois, l'autre a l'avantage de permettre à l'utilisateur de faire un geste circulaire répété en peu d'espace tout en permettant de se rapprocher ou de s'éloigner du centre de rotation pour faire varier le gain en degrés par pixel. Il existe alors des opportunités pour développer des nouveaux *widgets* ou techniques d'interaction qui combinent les avantages de ces différentes modalités de contrôle.

1.5 Résumé

À la suite des lectures effectuées, les animations ont plusieurs applications possibles dans les interfaces graphiques. En particulier, les transitions fluides sont une meilleure approche pour montrer des changements de vues entre des données comparativement aux transitions soudaines. L'effet de mouvement apporte une meilleure compréhension de la nature des changements et de la position des éléments. Plusieurs techniques peuvent être utilisées pour mieux concevoir des animations comme les accélérations en début et fin de mouvement. Les transitions visuelles différenciées exposent des informations supplémentaires sur les données en ajoutant une dimension propre au mouvement de chaque élément. De leur côté, les transitions par étapes juxtaposent plusieurs animations différentes où sont regroupées des actions semblables dans le but de bâtir une transition fluide complète.

Des études comme celles de Heer et Robertson (2007) confirment que les transitions animées (linéaires ou par étapes) sont mieux que les transitions soudaines, mais n'ont pas trouvé de grand avantage pour les animations par étapes par rapport aux animations linéaires, et ont seulement étudié des données n'ayant pas de liens entre eux. Deux questions peuvent alors se poser : est-ce que les animations par étapes pourraient avoir un plus grand avantage par rapport aux animations linéaires dans des visualisations de données ayant des liens entre elles comme avec les arborescences? De plus, est-ce que la visualisation de telles données interreliées ou structurées permet de nouvelles variantes d'animations en fonction des liens entre elles, et peuvent-elles être plus faciles à comprendre?

Une troisième question se pose aussi : y a-t-il une façon de concevoir de nouveaux *widgets* interactifs qui pourraient combiner les avantages des *widgets* antérieurs en permettant, par exemple, à l'utilisateur de choisir d'effectuer soit un contrôle discret, donc rapide, en lançant une animation à vitesse fixe ou bien de faire une entrée continue en ayant le plein contrôle sur la vitesse et le sens de progression de la transition?

Des réponses positives à ces trois questions pourraient rendre les transitions animées plus faciles à comprendre lorsqu'il y a plusieurs changements complexes ou plus d'éléments à montrer, rendant donc les transitions plus extensibles. Les prochains chapitres font une investigation de ces questions et proposent des solutions possibles.

CHAPITRE 2

ANALYSE ET CRÉATION D'ANIMATIONS POUR LES ARBORESCENCES

La revue de littérature a permis de connaître les types de transitions animées proposées par la littérature antérieure. Essentiellement, les chercheurs utilisent des animations linéaires qui montrent une seule transition entre un état initial et final ou des animations par étapes qui découpent une transition complète en plusieurs segments comportant des attributs particuliers. Ces études n'ont pas trouvé de grande différence de performance entre les animations linéaires et les animations par étapes (malgré la préférence subjective des utilisateurs pour ces dernières), et les ont seulement évaluées avec des données non structurées ne possédant pas de liens entre les éléments. Ce chapitre étudie la question pour découvrir quelles transitions animées sont possibles pour des données structurées, en particulier dans une visualisation d'arborescence. Il présente d'abord un prototype servant à tester différentes approches de transitions animées, puis analyse et présente les transitions linéaires et par étapes. Ensuite, il propose quelques nouveaux types de transitions animées qui exploitent la structure hiérarchique d'une arborescence et qui pourraient être plus faciles à comprendre. Le chapitre suivant présentera une évaluation expérimentale de ces techniques d'animation.

2.1 Implémentation d'un prototype

Un prototype réalisé avec le langage de programmation Java et la librairie Java OpenGL (JOGL) pour le rendu graphique permettra de mettre en pratique les animations développées pour cette étude. Le type d'arborescence choisi est radial (voir Figure 2.1) et sa représentation est semblable à un arbre de ballons vu de haut (*balloon tree*) (Boardman, 2000, Melançon et Herman, 1998 et Teoh et Ma, 2002). Un nœud est représenté par un cercle et chaque nœud enfant est dessiné autour du nœud parent qui est au centre. La grosseur de chaque nœud est directement déterminée en fonction de celle de son parent et du nombre

d'enfants présents à l'intérieur de ce premier. Ainsi, il est possible que des nœuds de même niveau dans l'arborescence soient de tailles différentes si ce ne sont pas des nœuds frères.

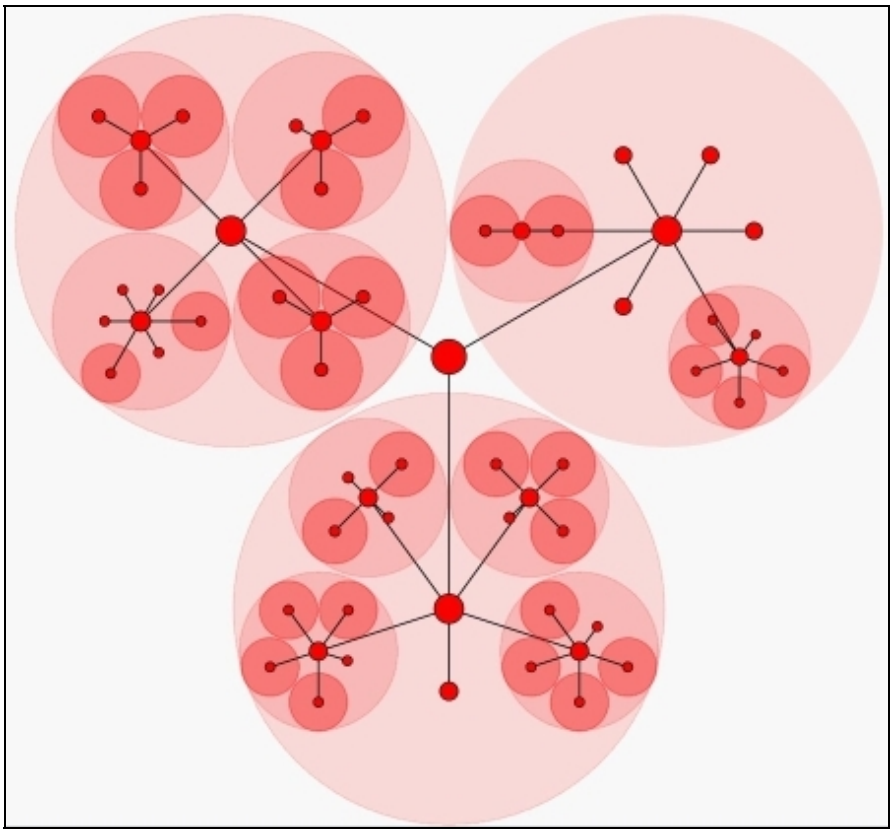

Figure 2.1 Arborescence radiale sous la forme d'un arbre à ballons (*balloon layout*)

Un problème possible de la présentation radiale est le risque d'occlusion des éléments. Étant donné que l'arbre a un seul point central et que tous les nœuds sont positionnés par rapport à celui-ci, le chevauchement des nœuds lors des permutations qui passent par le centre est très important et augmente la confusion pour suivre un ou plusieurs nœuds qui se déplacent. Cependant, l'agencement radial de cet arbre est simple et montre de façon claire la hiérarchie des nœuds sur plusieurs niveaux. De plus, il a aussi l'avantage d'être plus efficace en termes d'espace alloué aux feuilles (nœuds enfants sur le niveau le plus profond) que certains types d'agencements traditionnels. En effet, les formes qui représentent les nœuds auront une plus grande aire en utilisant un agencement radial comparativement à un agencement traditionnel

lorsque les deux arbres sont dessinés à l'intérieur de la même superficie (voir Figure 2.2). Toutefois, l'objectif du prototype n'est pas de proposer un nouvel algorithme d'agencement des nœuds qui maximise l'espace à l'écran pour présenter le plus d'informations possible ou qui réduit l'occlusion. En utilisant une présentation de base déjà existante, les efforts d'implémentation sont concentrés principalement sur les techniques d'animation. Ainsi, les avantages perçus ou non dépendront uniquement de ces techniques et non de la présentation de l'arbre qui ne changera pas.

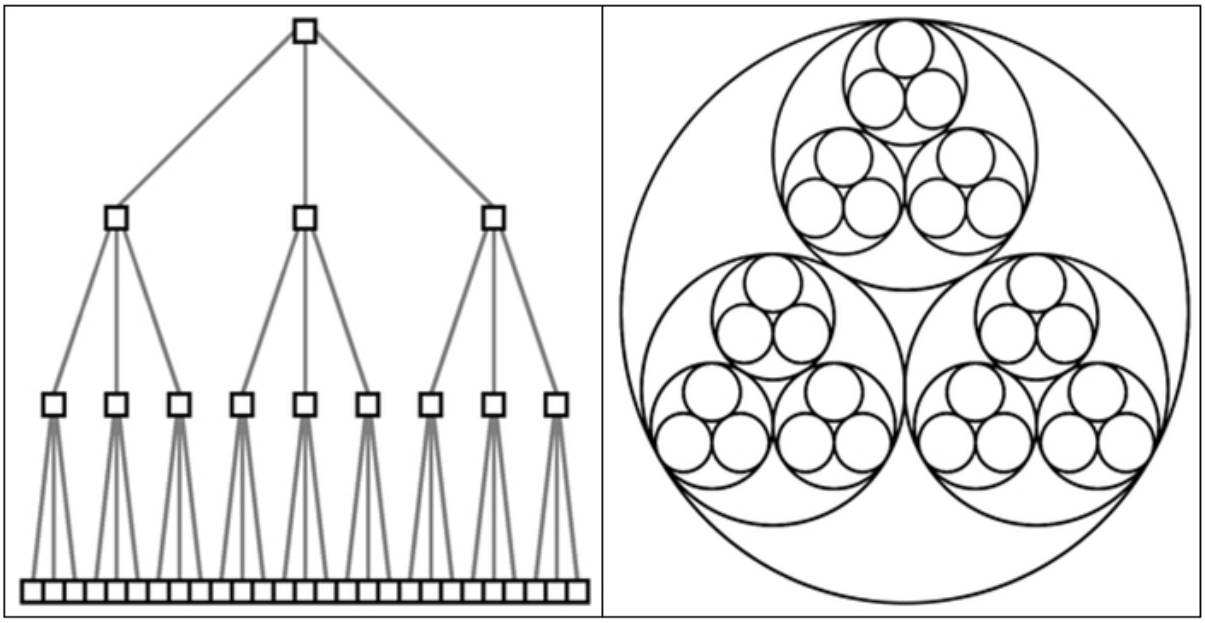

Figure 2.2 Agencement traditionnel (à gauche) comparé à l'agencement radial (à droite) qui présente une plus grande aire pour les feuilles et les autres nœuds de l'arborescence
Adaptée de Michael McGuffin

Le fonctionnement général du prototype se base sur une structure hiérarchique de nœuds sous forme d'arbre qui contient pour chaque nœud les informations suivantes : un identifiant unique, une valeur numérique pouvant être utilisée pour un tri, l'état initial (ouvert ou fermé) et les liens vers les nœuds enfants. La disposition sous forme radiale ainsi que la taille et la position des nœuds sont calculées indépendamment ce qui permet de réutiliser la même structure pour d'autres types de présentations. Toutes les informations relatives à la

disposition sont stockées dans une liste qui peut contenir plusieurs dispositions différentes en même temps. Cette liste est utilisée pour appliquer les changements à l'intérieur de l'arbre, sans modifier sa structure initiale, puis pour les animer en passant d'une disposition à une autre. Par exemple, la disposition initiale (*layout* 0) contient les informations sur l'état de l'arborescence en cours. En copiant ces informations et en appliquant des changements dans une autre disposition (*layout* 1), il y a maintenant un état final et initial. En analysant les différences entre les deux états, il est possible de connaître les nœuds qui vont s'ouvrir, se fermer ou permuter et ainsi calculer leur nouvelle position avant de démarrer l'animation. Ces deux premières dispositions servent de base pour manipuler les états, mais dans le cadre des animations par étapes, il est possible d'en utiliser plusieurs pour bâtir une transition complète. Ainsi, les modifications de la disposition initiale (*layout* 0) seront appliquées dans une autre (*layout* 1), mais des dispositions intermédiaires seront calculées tout dépendant du type d'animation et de sa conception. L'état final de l'arborescence pourrait alors se retrouver dans une troisième disposition (*layout* 3) après avoir passé par deux dispositions intermédiaires (*layout* 1 et 2). Dans ce contexte, l'animation aurait trois étapes distinctes.

Toutefois, avant d'aller plus loin avec les animations complètes, illustrons d'abord les trois types de changements qu'un nœud peut subir à l'intérieur de l'arbre : les ouvertures, les fermetures et les permutations. Un nœud qui est ouvert permet de voir la hiérarchie de ses enfants contrairement à un nœud qui est fermé. Lors d'une transition, un nœud peut s'ouvrir ou se fermer tout dépendant des informations désirées soit pour en montrer plus ou en cacher d'autres. Premièrement, l'animation de la fermeture d'un nœud (voir Figure 2.3, 1 à 6) débute par la disparition immédiate des enfants puis par la fermeture du cercle qui entoure le nœud en sens antihoraire (2 à 4). Par la suite, le nœud fermé se rapproche légèrement du centre de l'arborescence par une translation (5 et 6). L'animation de l'ouverture (voir Figure 2.3, 6 à 1) s'effectue de façon inverse soit par l'éloignement du nœud fermé par une translation vers sa position finale (6 et 5). Ensuite, l'ouverture du cercle entourant le nœud se fait en sens horaire (4 à 2) puis les enfants du nœud apparaissent en fin d'animation (1).

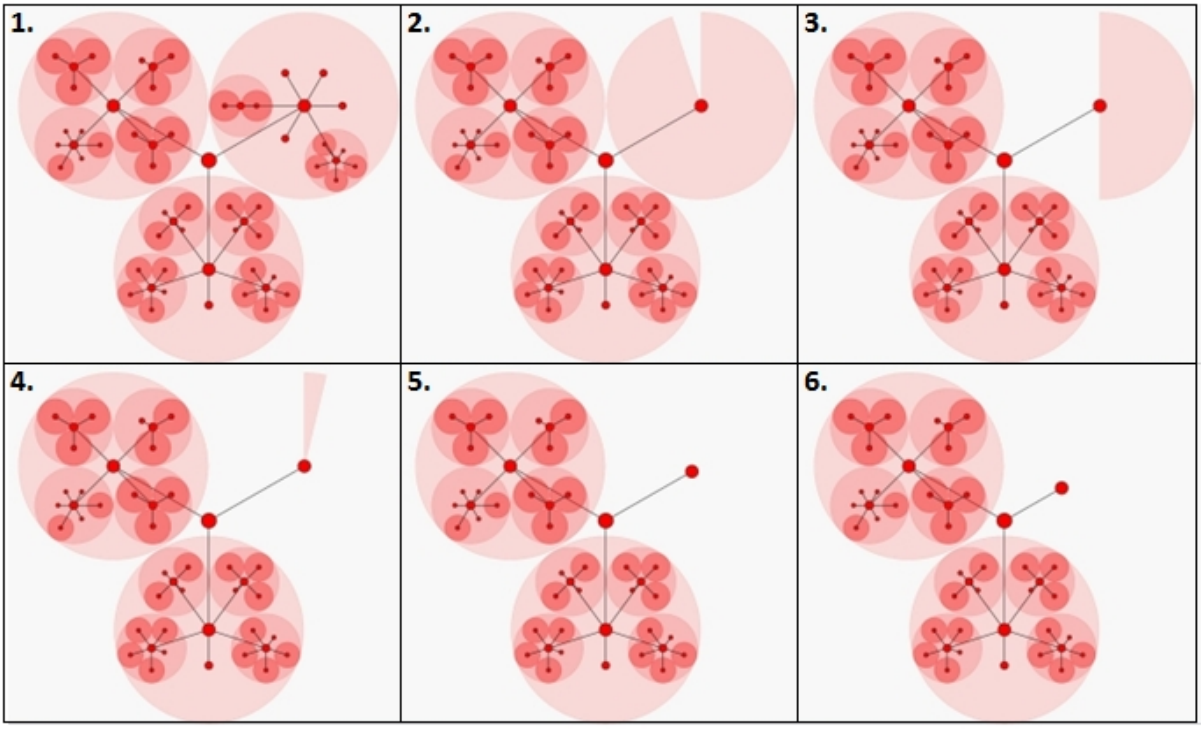

Figure 2.3 Séquence d'animation de la fermeture (1 à 6) et de l'ouverture (6 à 1) du nœud supérieur droit de l'arborescence

Les permutations de nœuds peuvent s'effectuer sur un niveau de l'arborescence ou encore à l'intérieur même d'un nœud parent. Elles peuvent aussi s'appliquer de façon récursive sur plusieurs niveaux ou sur l'arbre en entier. Pendant une animation, l'ordre d'affichage des enfants d'un nœud peut être modifié pour représenter un tri sur les valeurs de chaque élément par exemple. L'animation des permutations d'un nœud et de ses enfants, dans sa forme initiale, se réalise de façon simultanée (voir Figure 2.4). Il s'agit d'une interpolation directe des coordonnées rectangulaires entre la position initiale et la position finale de chacun des nœuds sur toute la durée de l'animation. Ainsi, il se peut que les trajectoires de certains nœuds se croisent entraînant ainsi la superposition des cercles qui les entourent (3 et 4). Étant donné que chaque nœud est semi-transparent, il est toujours possible d'en suivre un en particulier.

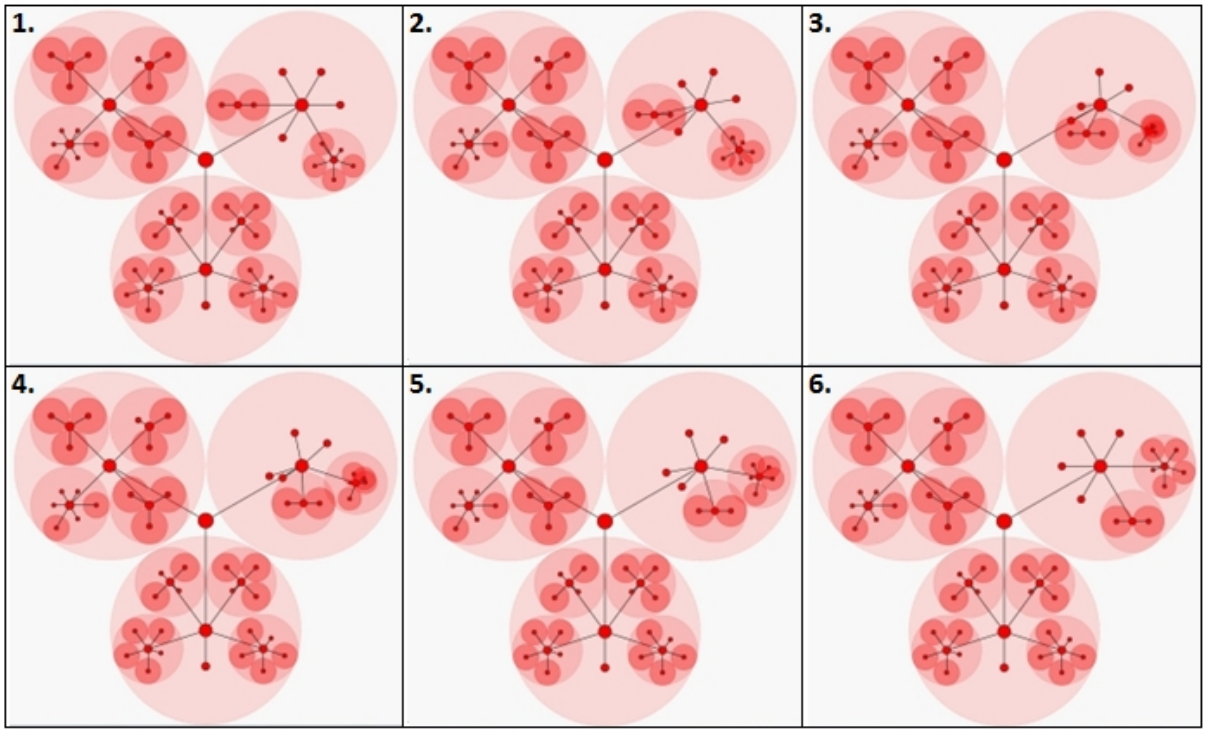

Figure 2.4 Séquence d'animation de la permutation des descendants du nœud supérieur droit

Bien que ces deux séquences d'animation ne présentent seulement qu'un changement à la fois, plusieurs changements sur différents niveaux de l'arborescence peuvent s'appliquer en même temps et ils s'effectueront de la même façon. Les prochaines sections de ce chapitre vont évoquer cette possibilité à l'intérieur d'une analyse théorique et d'une démonstration de chaque type d'animation. L'analyse qui suit permettra d'explorer l'espace de conception autour des transitions animées dans les visualisations d'arborescences, tant pour mieux comprendre les animations linéaires et par étapes, que pour identifier de nouvelles sortes de transitions animées.

2.2 Les animations linéaires et par étapes

Les deux types d'animations déjà utilisées dans le domaine de la visualisation d'informations sont les animations linéaires et par étapes. Bien qu'elles soient déjà définies par certains chercheurs dans leur contexte d'application, il est nécessaire d'avoir recours à une nouvelle

méthode de notation pour la présente analyse. Deux facteurs sont à considérer dans le cas de changements multiples et simultanés à l'intérieur d'une arborescence : le type de changement et le niveau du nœud auquel il s'applique. Pour ce faire, une matrice sera utilisée avec deux dimensions soit une verticale pour le type de changement et l'autre horizontale pour le niveau en profondeur du nœud sur lequel le changement s'applique. Par exemple, un nœud de niveau 1 est de descendance plus proche de la racine de l'arbre (nœud central) qu'un nœud de niveau 2. Dans la présentation de l'arbre choisie, un nœud de niveau moindre sera donc plus gros qu'un nœud de niveau plus élevé pour le même parent. Ainsi, chaque case de la matrice contiendra l'ordre dans lequel l'animation d'un type de changement sur un niveau en particulier sera appliquée. Il se peut qu'une seule ou plusieurs combinaisons de changements et de niveaux s'animent en même temps ou de façon séquentielle selon le type d'animation.

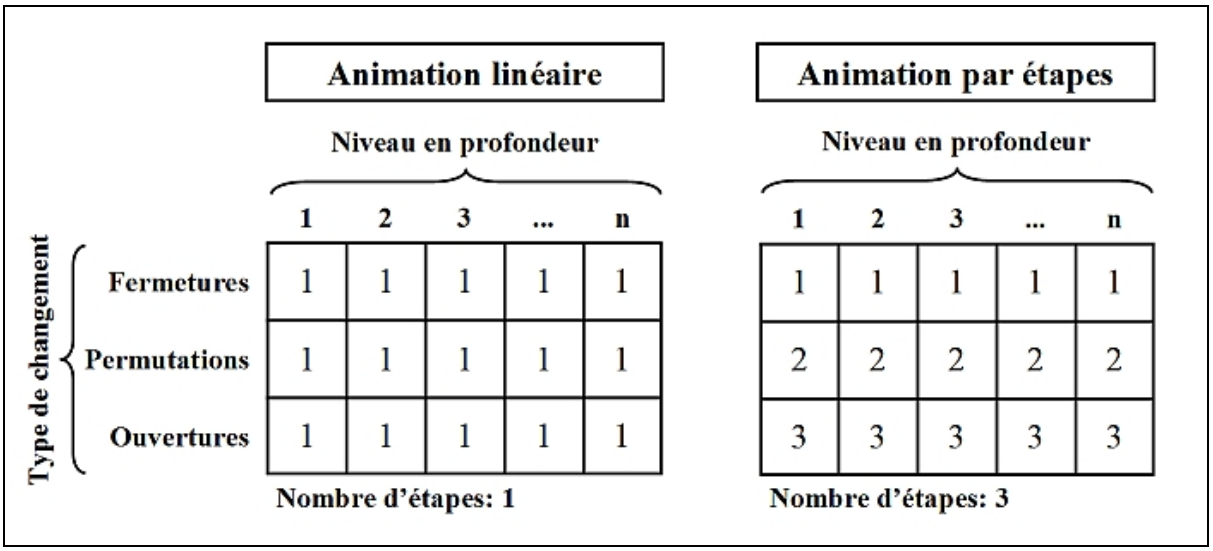

Figure 2.5 Matrices de la décomposition des animations linéaire et par étapes selon le type de changement et le niveau en profondeur de l'arborescence

L'animation linéaire possède une seule étape pour montrer les changements entre deux états. Ce concept est adaptable facilement au contexte des arborescences puisque tous les types de changements sur tous les niveaux se produisent en même temps. Donc, peu importe le nombre de changements et la profondeur de l'arbre, l'animation linéaire contient toujours une seule étape (voir Figure 2.5). La Figure 2.6 montre la séquence de l'animation linéaire

dans un contexte réel comportant une ouverture, une fermeture et plusieurs permutations sur différents niveaux de l'arbre. Les mêmes changements seront utilisés pour présenter la séquence de tous les types d'animations qui vont suivre.

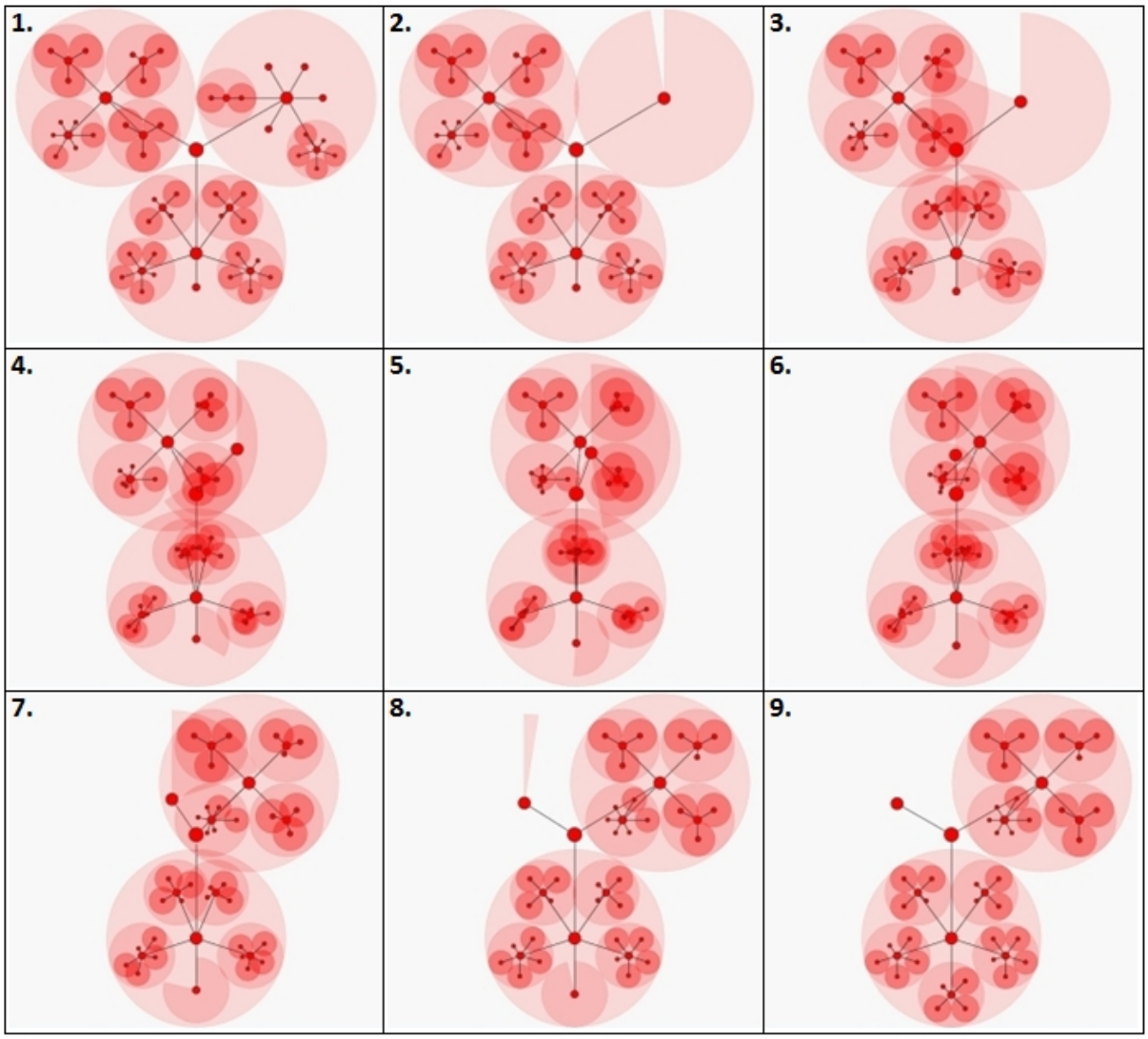

Figure 2.6 Séquence de l'animation linéaire

De son côté, l'animation par étapes (voir Figure 2.5) pourrait se faire de plusieurs façons différentes si tous les types de changements sont combinés avec des ordres différents. Toutefois, en se basant sur la technique de Plaisant *et al.* (2002), il est préférable de débuter

la séquence (voir Figure 2.7) par la fermeture des nœuds (1 à 4) pour empêcher qu'ils interfèrent avec les autres pendant l'animation étant donné qu'ils ne seront plus visibles à la fin. Par la suite, les permutations (5 à 8) pourront s'effectuer pour replacer les nœuds dans leur position finale puis viendront les ouvertures (9 à 12) en dernier puisqu'ils présentent de nouvelles informations par rapport à l'état initial. Peu importe la profondeur de l'arbre, le nombre maximal d'étapes pour ce type d'animation est de trois et chaque type de changement s'appliquera sur tous les niveaux en même temps.

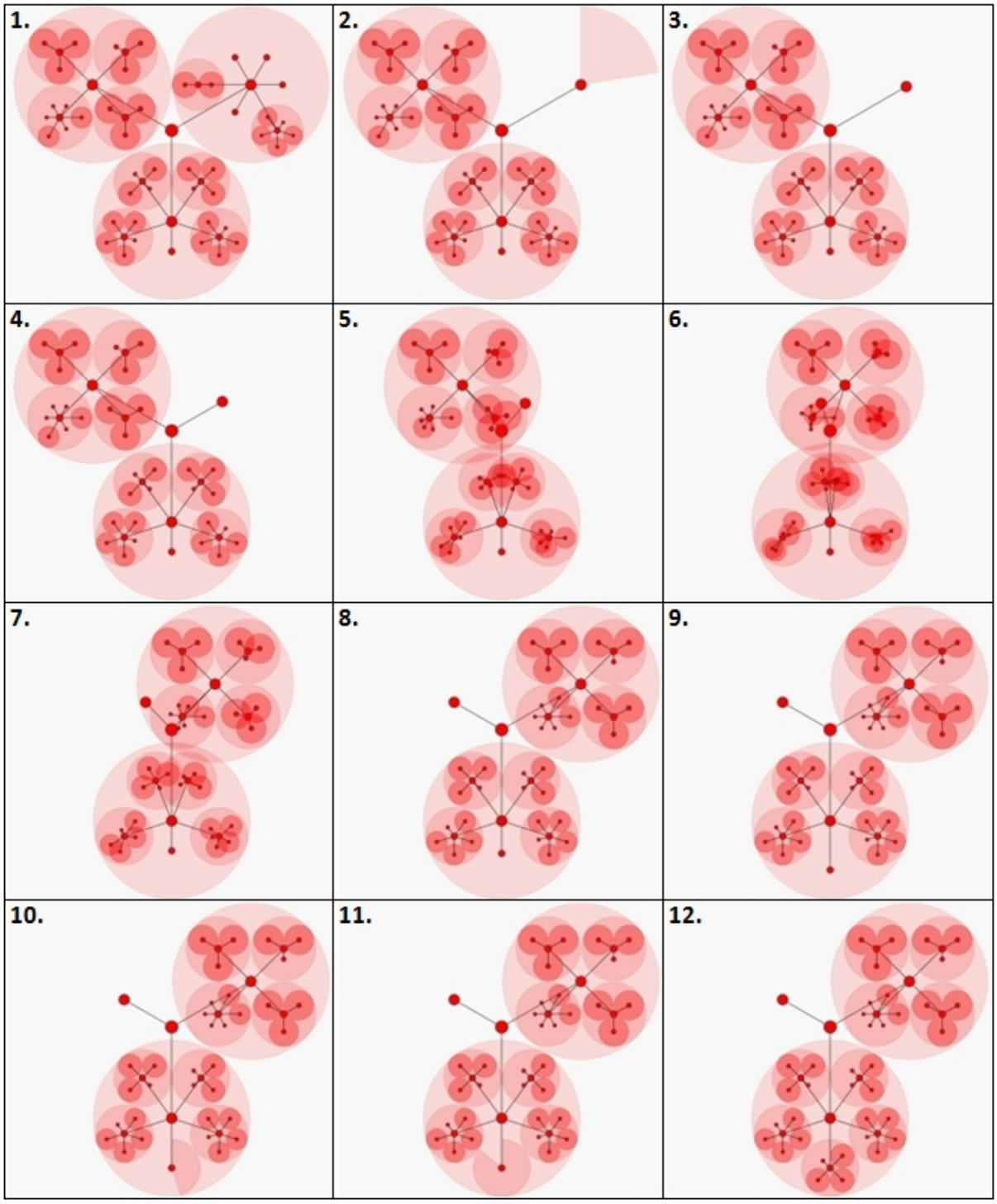

Figure 2.7 Séquence de l'animation par étapes qui montre les fermetures (1 à 4), les permutations (5 à 8) et les ouvertures (9 à 12) des nœuds

2.3 Les animations hiérarchiques

Les animations hiérarchiques représentent une technique de transition animée novatrice qui se base sur les niveaux de l'arborescence pour agencer les changements et ainsi bâtir une transition complète entre l'état initial et final. Elles se démarquent des animations existantes en apportant une nouvelle logique pour le suivi d'un nœud en rapport avec les liens qui unissent ses parents et ses enfants.

Premièrement, l'animation hiérarchique simple (voir Figure 2.8) débute par animer tous les changements dans le premier niveau de l'arborescence et fait de même jusqu'au plus profond. Les changements de premier niveau captent facilement l'attention de l'utilisateur puisqu'ils se produisent au début de l'animation et parce que la taille des nœuds est plus grande. De plus, comme les changements se produisent sur un seul niveau à la fois, la translation engendrée par une permutation d'un nœud parent cause la même translation sur tous les nœuds descendants de ce dernier. L'utilisateur peut donc suivre un nœud de premier niveau qui est en train de permuter puis, une fois arrivé à sa position finale, observer les permutations des enfants, et par la suite celles des petits-enfants. Son attention est ainsi dirigée d'un nœud de premier niveau vers les nœuds de niveaux plus profonds tout en limitant la zone à l'écran où l'utilisateur doit garder le focus. Cette logique peut aussi aider à prévoir les déplacements des nœuds et ainsi permettre à l'utilisateur d'en suivre plusieurs à la fois tout en changeant stratégiquement son focus d'attention. Comme l'animation hiérarchique se base directement sur les niveaux de l'arborescence, la transition complète comportera le même nombre d'étapes que la profondeur n de l'arbre.

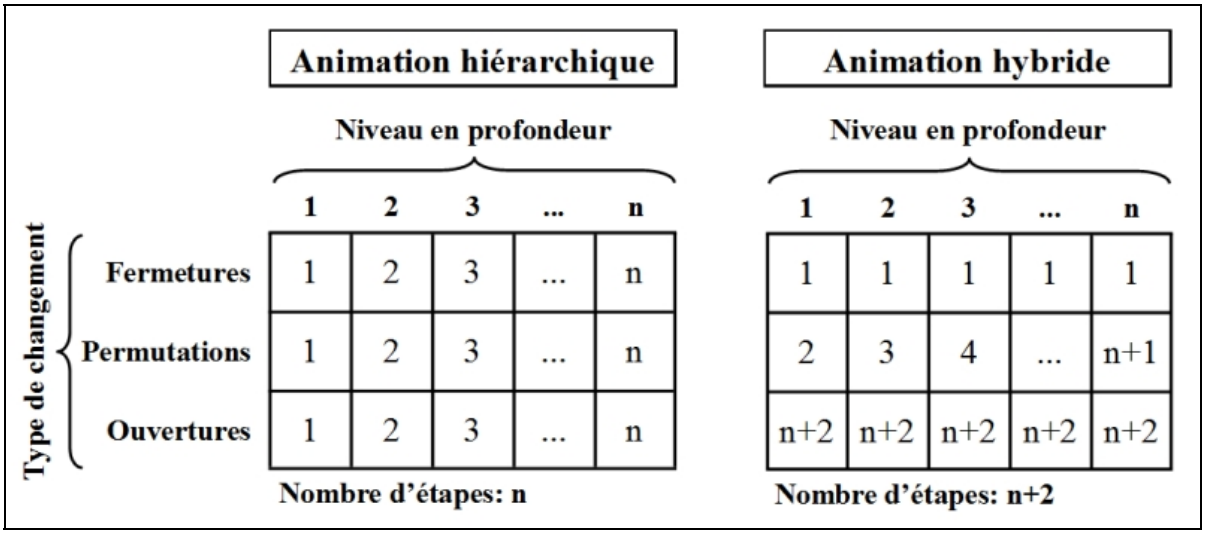

Figure 2.8 Matrices de la décomposition des animations hiérarchique et hybride selon le type de changement et le niveau en profondeur de l'arborescence

La Figure 2.9 démontre la séquence de l'animation hiérarchique simple. Étant donné que l'arbre en exemple contient trois niveaux de nœuds, l'animation comporte également trois étapes. Tout d'abord, un nœud de premier niveau se ferme en même temps qu'il permute avec un autre (1 à 5). Ensuite, deux nœuds de deuxième niveau permutent pendant qu'un troisième est en train d'ouvrir (6 à 9). Finalement, les nœuds de troisième niveau permutent en même temps et la transition se termine (10 à 12).

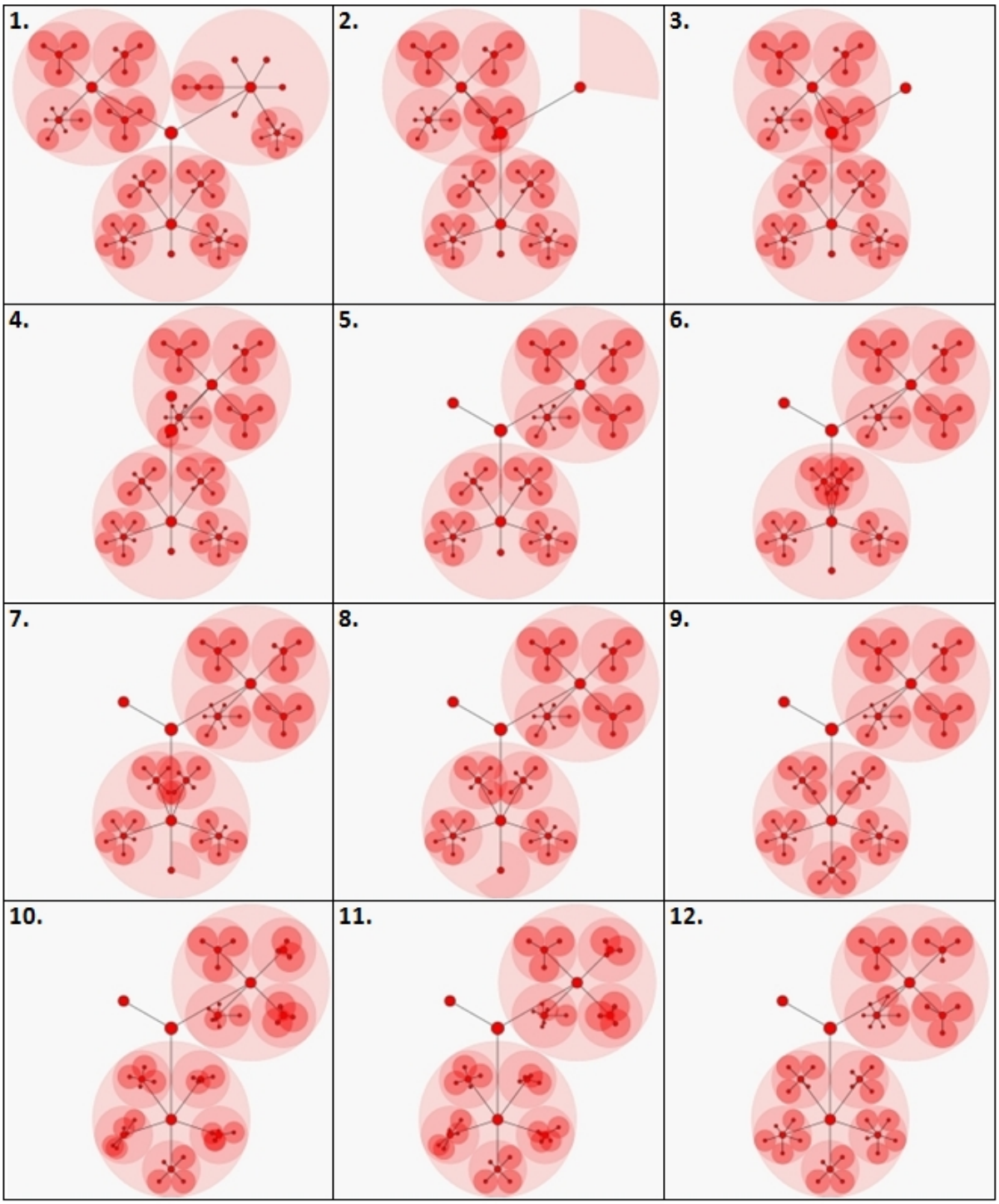

Figure 2.9 Séquence de l'animation hiérarchique du premier (1 à 5), deuxième (6 à 9) et troisième niveau (10 à 12) de l'arborescence

Vu que l'animation hiérarchique se base uniquement sur les niveaux de l'arborescence, il serait intéressant d'inclure les types de changements comme facteurs pour bâtir une transition. En regardant l'analyse effectuée pour l'animation par étapes, il serait possible de combiner le même principe avec l'animation hiérarchique simple pour en faire une de type hybride (voir Figure 2.8). De cette façon, dans l'exemple de la Figure 2.10, toutes les fermetures de nœuds se produisent simultanément au début de l'animation (1 à 3). Ensuite, les permutations sont effectuées de façon hiérarchique en débutant par le premier niveau (4 à 6), le deuxième niveau (7 à 9) et le troisième (10 à 12). L'animation se termine finalement par les ouvertures de nœuds sur tous les niveaux en même temps (11 à 15). La création de l'animation hybride ajoute deux étapes supplémentaires à l'animation hiérarchique simple, ce qui porte le nombre total des étapes de la transition de l'exemple à cinq.

Par rapport à l'animation hiérarchique, l'animation hybride pourrait avoir comme avantage que toutes les fermetures se font au début et que toutes les ouvertures sont retardées jusqu'à la fin, simplifiant donc la complexité visuelle des permutations au milieu. Par rapport à l'animation par étape, l'animation hybride pourrait avoir comme avantage de mieux guider l'œil de l'utilisateur pendant les permutations, car ce dernier pourra observer les mouvements des sous-arbres de façon grossière d'abord et ensuite détaillée à mesure que la transition avance.

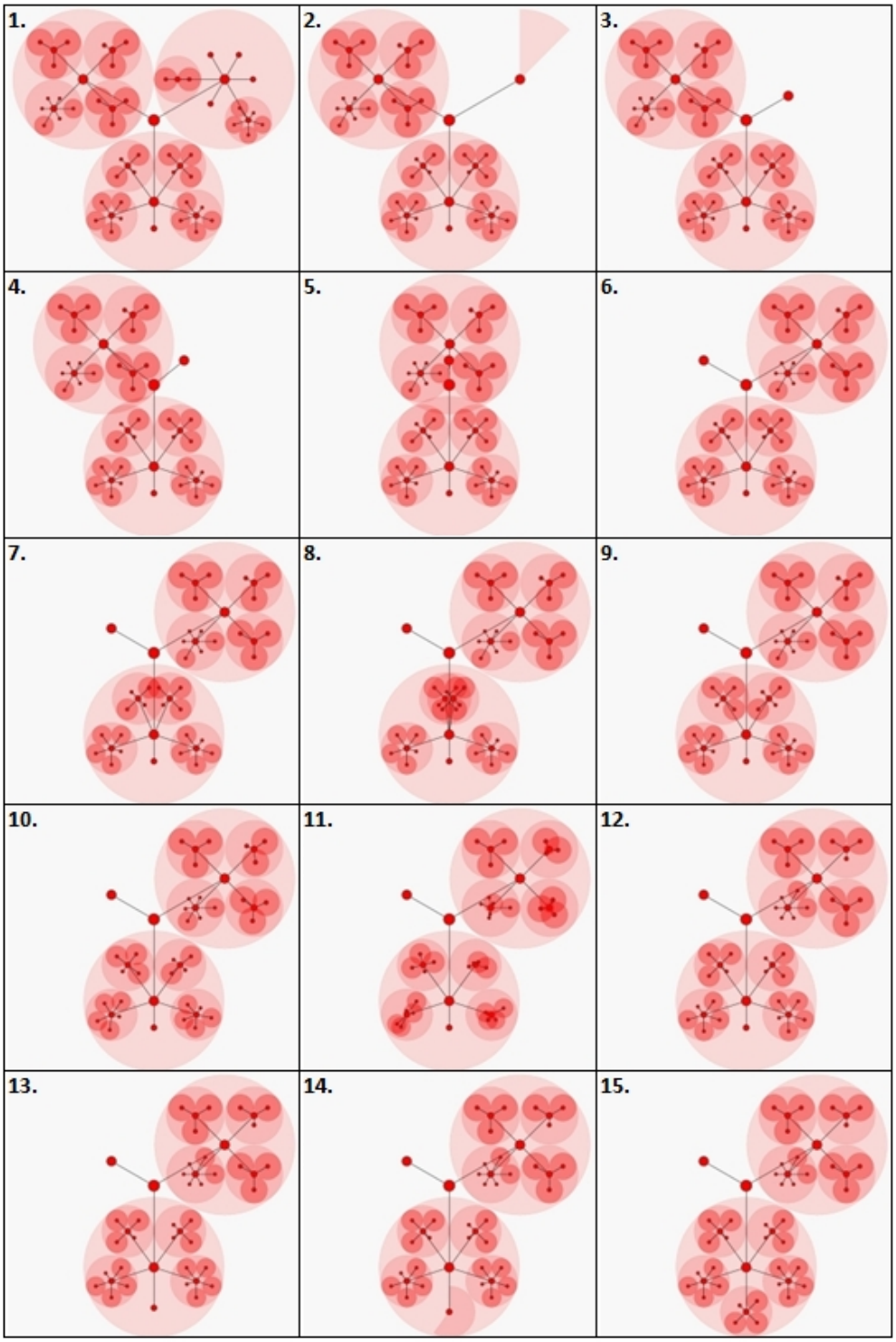

Figure 2.10 Séquence de l'animation hybride qui montre les ouvertures (1 à 3), les permutations de premier (4 à 6), deuxième (7 à 9) et troisième niveau (10 à 12) puis les fermetures de nœuds (11 à 15)

2.4 Analyse des possibilités et choix de compromis

La notation sous la forme de matrices pour montrer la décomposition des animations a permis de mieux définir l'ordre des étapes selon le type de changement et les différents niveaux de l'arbre. Cependant, cette approche apporte des dizaines de combinaisons possibles si un type de changement en particulier est animé avant un autre et si la séquence débute par un niveau peu profond ou non. Les animations par étapes éliminent déjà plusieurs possibilités en prônant l'animation des fermetures en premier et des ouvertures à la fin. De même, les animations hiérarchiques proposent de diriger l'attention de l'utilisateur en construisant une transition qui commence par le niveau le plus près de la racine de l'arbre vers les niveaux les plus profonds. Toutefois, quelques possibilités demeurent à être analysées.

Les animations 1 et 2 de la Figure 2.11 sont une combinaison de l'animation par étapes et de l'animation hiérarchique, mais qui se répète plusieurs fois. Pour faciliter la compréhension et éliminer la complexité de la notation, les matrices sont fixées à trois niveaux en profondeur ($n = 3$). L'animation 1 propose d'animer les types de changements selon l'ordre de l'animation par étapes, mais de façon répétée pour chaque niveau de l'arborescence. Ainsi, les fermetures de premier niveau vont s'effectuer au début, avant les permutations puis les ouvertures toujours à partir du niveau le moins profond. Ensuite, la même séquence se produit pour le deuxième niveau et finalement pour le troisième et dernier niveau. L'exemple porte le nombre d'étapes à neuf ce qui se généralise par trois fois plus d'étapes que le nombre de niveaux de l'arbre. L'animation 2 anime d'abord les fermetures de manière hiérarchique à partir du premier niveau vers le plus profond de l'arbre. La même séquence se répète pour les permutations et les ouvertures. Cet autre agencement d'animation hiérarchique par étapes amène aussi neuf étapes soit trois fois plus que le nombre de niveaux. Finalement, l'animation 3 tente d'apporter une direction différente aux animations analysées précédemment. Elle anime d'abord les fermetures de façon hiérarchique du premier niveau vers le plus profond, mais poursuit avec les permutations du niveau le plus profond en

remontant vers le premier. La séquence se termine ensuite par l'animation des ouvertures du premier niveau vers le dernier. Ce changement suppose qu'une fois les fermetures terminées, l'attention de l'utilisateur se trouve au dernier niveau de l'arbre. Il pourrait alors être logique de continuer la séquence à partir de ce point pour revenir vers le premier niveau avec les permutations et redescendre dans la profondeur de l'arbre avec les ouvertures. Une fois de plus, cette séquence comporte le même nombre d'étapes que les animations 1 et 2.

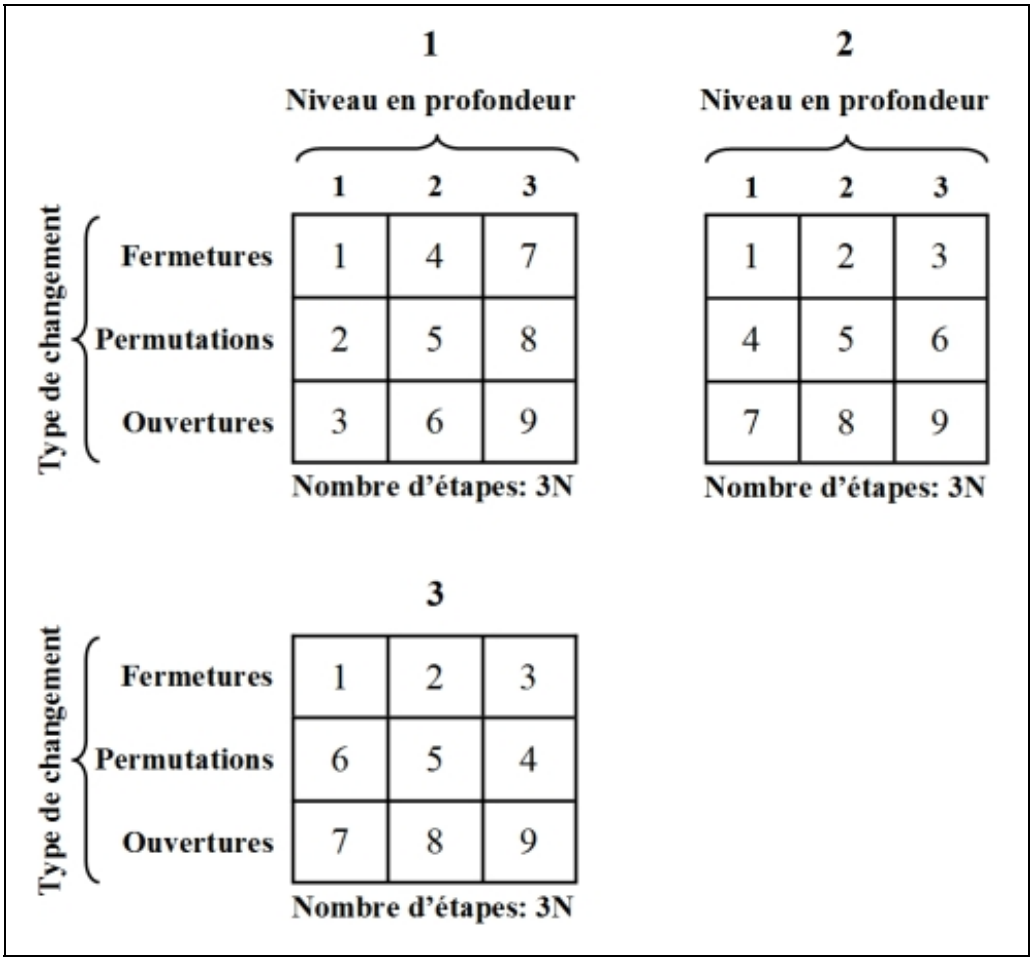

Figure 2.11 Matrices de la décomposition d'autres types d'animations possibles

Afin de déterminer la pertinence de chaque type d'animation énoncé dans ce chapitre, il est nécessaire de fixer certaines variables et d'étudier les compromis à choisir. D'abord, le temps total de l'animation ne doit pas être trop long surtout si le nombre d'étapes est élevé. Un

temps trop long pourrait nuire à la rapidité de la tâche en cours et pourrait aussi mener à une confusion, car l'utilisateur pourrait oublier la nature des premières étapes de la séquence. À l'inverse, un temps beaucoup trop court ne permettrait pas de bien voir tous les changements et l'utilisateur pourrait perdre de vue les nœuds qu'il veut suivre. Le but est donc de minimiser le temps total tout en minimisant la vitesse de chaque étape. Pour ce faire, il est nécessaire de fixer le temps total de la séquence pour chaque type d'animation dans le but de les comparer. Par exemple, pour un temps de cinq secondes, l'animation linéaire comportera une étape de cinq secondes et l'animation par étapes en aura trois de 1,66 seconde chacune environ. Pour l'animation hiérarchique, le temps doit être divisé par le nombre de niveaux de l'arbre ce qui donne donc pour un arbre de trois niveaux environ 1,66 seconde par étape, de quatre niveaux 1,25 seconde et de cinq niveaux une seconde par étape. De son côté, l'animation hybride possède deux étapes de plus que le nombre de niveaux de l'arborescence ce qui porte la durée d'une étape à une seconde pour un arbre de trois niveaux, environ 0,83 seconde pour un arbre de quatre niveaux et environ 0,71 seconde pour un arbre à cinq niveaux. En fixant ainsi le temps, il est donc facile de voir que la complexité de la transition a une incidence sur la performance, car un nombre élevé d'étapes amène une vitesse d'animation plus rapide. Pour optimiser la performance tout en minimisant le temps total et la vitesse de chaque étape, il faut donc éliminer les animations qui comportent un nombre d'étapes excessif. C'est pourquoi les animations 1, 2 et 3 de la Figure 2.11 ne seront pas retenues puisque chacune des neuf étapes de la séquence pour un arbre de seulement trois niveaux s'effectuerait en environ 0,55 seconde. De plus, le découpage de la transition en un grand nombre d'étapes peut être difficilement compris par un utilisateur.

Selon l'analyse effectuée, les animations hiérarchique et hybride pourraient offrir un avantage par rapport aux animations linéaire et par étapes pour les visualisations d'arborescences. Une expérimentation avec des utilisateurs sera en mesure de récolter des résultats concrets sur la performance de celles-ci et ainsi venir supporter ou non cette possibilité.

CHAPITRE 3

EXPÉRIMENTATION ET RÉSULTATS

Afin de mesurer les différences de performance entre les types d'animation et de connaître les avantages de chacun, une expérience contrôlée doit être réalisée avec des participants. Le but principal est de savoir quel type d'animation permet de mieux assimiler des changements dans une arborescence et faciliter le suivi des nœuds à l'écran. Avec l'analyse des résultats obtenus, les hypothèses pourront être vérifiées et il sera possible de connaître les circonstances qui amènent un avantage ou un désavantage au niveau de la réussite des tâches. Ce processus permettra de mieux comprendre comment bâtir des transitions fluides et ainsi proposer des directives de conception pour les animations avec plusieurs étapes.

3.1 Méthodologie de l'expérimentation

Avant de procéder à l'expérimentation, il est nécessaire de suivre une méthodologie précise pour s'assurer de la validité de l'étude. La prochaine section traite de l'identification des tâches et des hypothèses, du déroulement du test, des équipements utilisés, des participants recrutés et des conditions expérimentales.

3.1.1 Identification des tâches

Pour évaluer la performance des différents types d'animation, il est important de reproduire un environnement fidèle à une utilisation réelle à l'intérieur d'un logiciel. Ainsi, il faut identifier des tâches qu'un utilisateur est susceptible d'accomplir couramment afin de rendre les résultats plus pertinents et concrets. Premièrement, l'utilisateur peut être intéressé à observer un nœud en particulier dans l'interface. Entre deux états d'un graphe, la valeur représentée par un nœud peut changer en plus de sa position à l'écran et celle de ses nœuds parents ou enfants. Il est donc nécessaire pour l'utilisateur d'identifier la nouvelle position du

nœud après l'animation en suivant sa trajectoire des yeux. Deuxièmement, l'utilisateur peut s'intéresser aux changements qui ont eu lieu dans l'arborescence en général. Il peut vouloir connaître les nœuds qui ont subi un type de changement précis ou encore percevoir tous les changements (ouvertures, fermetures et permutations) sur tous les niveaux.

Dans une interface de furetage d'une arborescence, ces deux tâches sont souvent faciles à accomplir, car l'utilisateur déclenche lui-même chaque changement un à la fois. Par exemple, il pourrait ouvrir ou fermer seulement un nœud, causant ainsi une transition animée relativement simple, comme dans l'interface de « SpaceTree » (Plaisant *et al.*, 2002), où les changements d'état et de position de tous les nœuds sont faciles à prévoir et à interpréter. Par contre, d'autres interfaces pourraient montrer des transitions animées comme dans une interface qui permet de faire la transition entre deux versions d'une arborescence. Ces deux versions peuvent représenter deux arborescences phylogénétiques (qui montrent les liens de parenté entre les organismes vivants) calculées de manières différentes, ou encore un système de fichiers à deux moments différents. Il peut s'agir aussi d'une interface pour visualiser l'évolution d'une arborescence à travers le temps comme une hiérarchie de systèmes ou de machines dans une usine, et l'état de chaque nœud. Chaque transition dans de telles interfaces pourrait impliquer plusieurs fermetures, ouvertures ou autres changements simultanés sur les différents nœuds. Dans ce cas, il pourrait devenir difficile pour l'utilisateur d'accomplir les deux tâches mentionnées précédemment, c'est-à-dire de suivre un nœud en particulier tout en restant plus ou moins conscient des changements ailleurs dans l'arborescence. En effet, les changements qui auront lieu pendant la transition sont partiellement ou totalement inconnus par l'utilisateur.

L'évaluation expérimentale décrite dans ce chapitre va alors montrer des transitions du deuxième type, où plusieurs fermetures, ouvertures et permutations peuvent se passer en même temps, afin de mesurer à quel point chaque type d'animation permet l'accomplissement des deux tâches. Ceci permettra aussi d'évaluer à quel point un type

d'animation est facile à interpréter, ce qui le rend donc extensible, dans le contexte général des transitions complexes.

3.1.2 Hypothèses

En se basant sur la littérature et sur l'analyse réalisée sur les animations, quelques hypothèses sur la performance de celles-ci peuvent être formulées en rapport aux tâches énoncées dans la section précédente. Premièrement, l'animation par étape permettra de mieux suivre un nœud comparativement à l'animation linéaire à cause du découpage en trois étapes subséquentes (hypothèse H01). Deuxièmement, les ouvertures seront facilement identifiables avec l'animation par étapes puisqu'elles se produisent à la fin de la séquence (H02). Du côté des fermetures, bien qu'il n'y ait pas d'avantages marqués, l'hypothèse H01 suggère qu'il sera tout de même plus facile de les identifier avec l'animation par étapes même si elles se produisent en début de séquence (H03). Pour les permutations, les animations linéaires et par étapes causent beaucoup d'occlusion lors du déplacement des nœuds dans l'interface graphique. Toutefois, toujours selon l'hypothèse H01, l'animation par étapes devrait être plus performante pour identifier les permutations de nœuds dans l'arborescence (H04).

Comme les animations hiérarchiques et hybrides sont mieux adaptées au contexte des arborescences, certaines hypothèses peuvent définir leurs résultats par rapport à l'animation linéaire et par étapes. Tout d'abord, le suivi de nœuds sera plus facile à réaliser avec l'animation hybride comparativement aux autres animations puisqu'elle regroupe les avantages de l'animation par étapes et les permutations séquentielles de l'animation hiérarchique (H05). De plus, l'animation hiérarchique devrait être meilleure que l'animation linéaire et par étapes pour le suivi des nœuds à cause d'une moins grande occlusion entre les éléments à l'écran (H06). Pour les ouvertures, l'avantage de l'animation hybride est mitigé par rapport à l'animation par étapes puisqu'elles obéissent au même principe, mais l'hypothèse H05 propose que l'identification des ouvertures soit plus facile avec l'animation hybride (H07). Cette dernière sera également meilleure que l'animation linéaire. De son côté,

l'animation hiérarchique devrait moins bien performer pour les ouvertures comparativement à l'animation par étapes ou hybride vu qu'elles ne se produisent pas nécessairement en fin de séquence (H08). Le même raisonnement peut être appliqué pour les fermetures. L'animation hybride sera meilleure pour l'identification des fermetures que les autres types d'animation si l'hypothèse H05 est considérée (H09). Toutefois, l'animation hiérarchique devrait avoir un avantage sur l'animation par étapes et linéaire à cause de l'hypothèse H06 et par le fait que les fermetures ne sont pas toutes nécessairement au début de la séquence (H10). Concernant les permutations, selon H05, l'animation hybride devrait être plus efficace que l'animation hiérarchique, mais son nombre d'étapes plus grand pourrait la défavoriser. Strictement du point de vue des permutations, un nombre d'étapes égal au nombre de niveaux est suffisant pour bien assimiler ce type de changement. C'est pourquoi l'animation hiérarchique sera plus efficace que l'animation hybride et les autres types d'animation (H11). Finalement, l'animation hybride sera plus performante que l'animation linéaire et par étapes principalement à cause de la trop grande occlusion générée par ces deux dernières (H12).

Pour mieux répertorier et résumer les hypothèses, le Tableau 3.1 présente tous les énoncés de façon succincte avec le symbole « plus grand que » qui veut dire « mieux que » en termes d'efficacité ou de justesse à comprendre les changements ou à suivre les nœuds. Ainsi, pour l'hypothèse H01, l'animation par étapes est plus performante que l'animation linéaire pour le suivi de nœuds.

Tableau 3.1
Résumé des hypothèses sur la performance des animations

Hypothèse	Catégorie	Énoncé
H01	Suivi de nœuds	Par étapes > Linéaire
H02	Ouvertures	Par étapes > Linéaire
H03	Fermetures	Par étapes > Linéaire
H04	Permutations	Par étapes > Linéaire
H05	Suivi de nœuds	Hybride > Linéaire, par étapes et hiérarchique
H06	Suivi de nœuds	Hiérarchique > Linéaire et par étapes
H07	Ouvertures	Hybride > Linéaire, par étapes et hiérarchique
H08	Ouvertures	Par étapes > Hiérarchique
H09	Fermetures	Hybride > Linéaire, par étapes et hiérarchique
H10	Fermetures	Hiérarchique > Linéaire et par étapes
H11	Permutations	Hiérarchique > Linéaire, par étapes et hybride
H12	Permutations	Hybride > Linéaire et par étapes

3.1.3 Déroulement

Avant le test, un formulaire de consentement est remis au participant dans le but de l'informer sur l'expérimentation et pour obtenir son accord pour sa participation. De plus, un pré questionnaire lui est remis pour connaître son profil personnel (sexe, âge, main dominante), sa scolarité et son expérience en informatique (main qui utilise la souris et nombre d'heures d'utilisation d'un ordinateur par jour). Le but du questionnaire est aussi de savoir si le participant a déjà utilisé des logiciels pour visualiser des arborescences, des graphes ou des réseaux, ou encore des logiciels dans lesquels il y a des animations

d'information ou des transitions animées. Par la suite, l'expérimentateur donne les consignes à respecter et explique la nature des tâches à réaliser. Il invite aussi le participant à procéder à quelques essais d'entraînement pour chaque type d'animation et de changements dans le logiciel de test. En aucun cas la logique et l'ordre d'animation des types de changement ou des niveaux de l'arbre ne sont expliqués dans le but que l'utilisateur les découvre par lui-même au cours du test.

Dans le but de recueillir des résultats sur la performance des animations, chaque participant doit exécuter un grand nombre d'essais dans le logiciel de test. Chaque essai comporte deux tâches. La tâche principale consiste à suivre avec les yeux, sans l'aide du pointeur de la souris, un nœud cible dans une arborescence. Au début de chaque essai, ce nœud cible est indiqué par un effet de surbrillance. Lorsque l'utilisateur est prêt, il appuie sur la barre d'espacement du clavier de l'ordinateur, ce qui enlève la surbrillance du nœud cible (pour le confondre avec les autres nœuds obligeant l'utilisateur à faire un effort pour le suivre), et déclenche une transition animée préprogrammée et paramétrée pour faire passer l'arborescence d'un état à un autre. À la fin de la transition animée, l'utilisateur doit indiquer où le nœud cible s'est déplacé en cliquant sur sa nouvelle position, terminant ainsi la tâche principale.

La tâche secondaire s'effectue de façon parallèle à la première et consiste à identifier un type de changement qui s'est appliqué à un ou plusieurs autres nœuds de l'arbre pendant la transition. Avant même le début de l'animation, un message placé en haut de l'écran informe l'utilisateur qu'il doit non seulement suivre le nœud cible, mais qu'il doit aussi porter une attention particulière à un type de changement précis qui se passera ailleurs dans l'arbre (voir Figure 3.1). Il y a trois tâches secondaires possibles parmi lesquelles une est choisie par le logiciel pour chaque essai : surveiller pour des nœuds qui subissent des fermetures, des ouvertures ou des permutations. Par exemple, la tâche secondaire illustrée dans la Figure 3.1 consiste à surveiller pour des fermetures ailleurs dans l'arbre. À la fin de la transition animée, l'utilisateur indique la nouvelle position du nœud cible (accomplissant la tâche principale), et

ensuite l'utilisateur doit aussi indiquer la position des nœuds qui ont subi le type de changement mentionné (des fermetures par exemple), ce qui termine la deuxième tâche. Finalement, l'utilisateur appuie sur la barre d'espacement pour terminer l'essai. En cas de doute, l'utilisateur est invité à appuyer directement sur la barre d'espacement au lieu de deviner la position des nœuds pour ainsi éviter de fausser les résultats. Après la complétion de l'essai, le résultat pour les tâches principale et secondaire est affiché dans le coin inférieur gauche de l'écran informant l'utilisateur de sa réussite ou de son échec. Le nombre de nœuds correctement et mal identifiés accompagne l'étiquette de la tâche secondaire. De plus, en tout temps, l'utilisateur peut consulter l'état d'avancement du test en fonction de la condition, du bloc et de l'essai actuel (voir Figure 3.1).

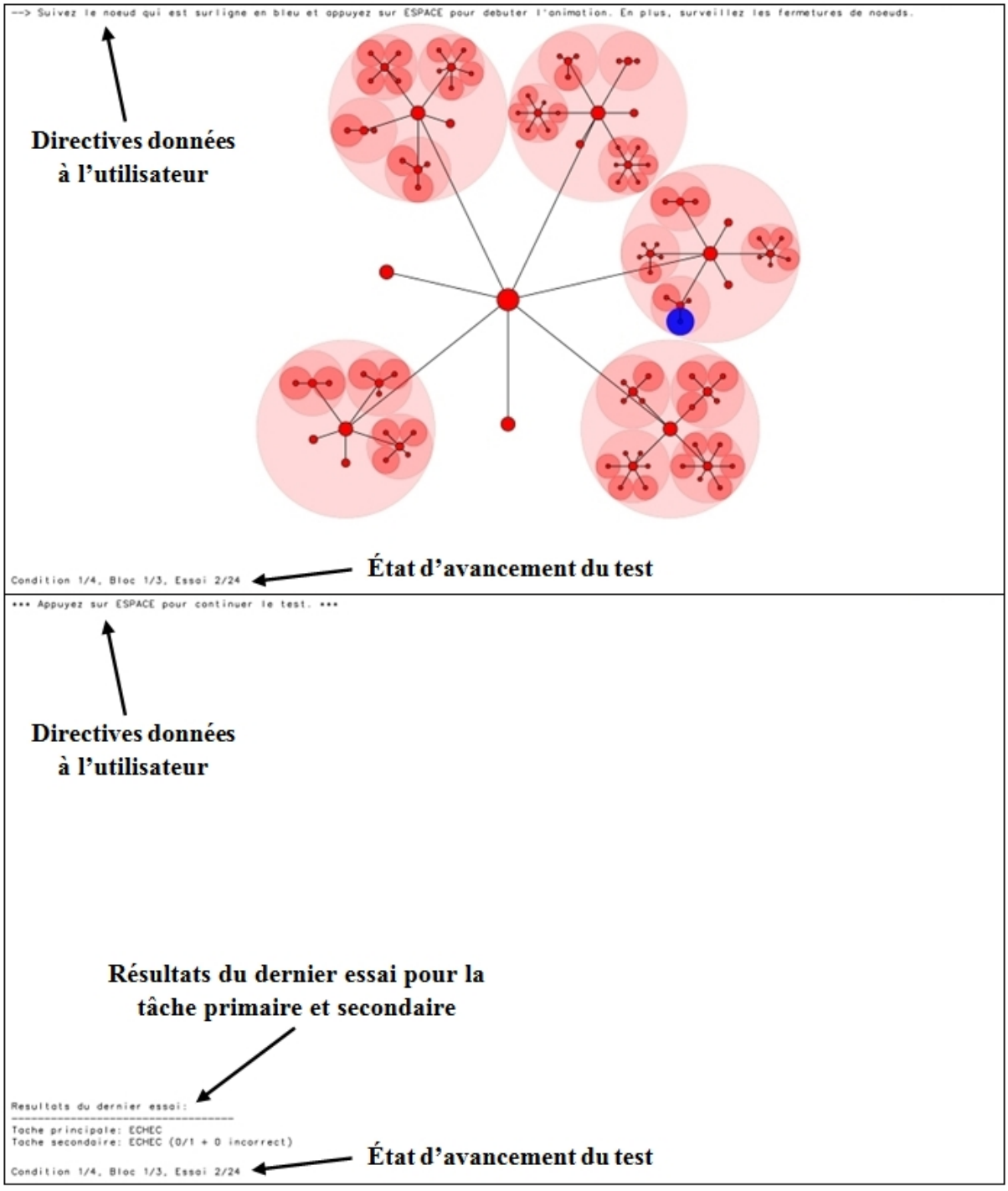

Figure 3.1 Captures d'écrans de l'interface de test avec les indications données à l'utilisateur

Après le test, un postquestionnaire est remis au participant dans le but de connaître son appréciation et ses commentaires sur les types d'animation. Il doit classer les différentes transitions selon leur capacité à permettre de bien comprendre le changement de position des nœuds et selon leur jugement subjectif de la vitesse d'animation. Le participant doit aussi indiquer le type d'animation qu'il préfère pour l'un ou l'autre des types de changements. Entre les blocs d'essais, le participant pouvait livrer ses commentaires à l'expérimentateur ou encore les noter sur le questionnaire à la fin du test. Afin de remercier le participant pour la complétion du test, un certificat-cadeau de 15 $ est remis en guise de compensation.

3.1.4 Équipements et environnement de test

Le prototype servant aux tests a été déployé sur un ordinateur portable qui comporte un processeur double cœurs de 2,53 gigahertz, 4 gigaoctets de mémoire vive et une carte graphique d'un gigaoctet de mémoire. L'interface graphique couvrait complètement un écran de 15,4 pouces (39,1 centimètres) avec une résolution de 1680 par 1050 pixels. Une souris sans fil avec une vitesse de défilement du curseur légèrement supérieure à la valeur par défaut (moyenne) était utilisée avec la main de l'utilisateur qui utilise normalement la souris pour cliquer sur des objets et les sélectionner à l'écran. Le clavier était utilisé avec l'autre main seulement pour changer d'essai ou pour confirmer la fin d'une tâche à l'aide de la barre d'espacement. Tous les tests se sont déroulés au même poste de travail dans un laboratoire de l'École de technologie supérieure où l'air ambiant est climatisé et où les distractions sont peu nombreuses. Cette section du laboratoire possède aussi un éclairage réduit et indirect assuré par des tubes fluorescents au plafond qui ne se reflète pas sur l'écran et qui n'indispose pas les yeux des utilisateurs.

3.1.5 Participants

Pour réaliser les tâches de l'expérimentation, 12 participants ont été recrutés parmi les étudiants de l'École de technologie supérieure, dont deux femmes et dix hommes. Ils sont âgés entre 21 et 45 ans (moyenne de 28,83 ans et écart type de 6,9 ans) dont deux sont gauchers et dix sont droitiers, mais ils utilisent tous la main droite pour contrôler la souris d'un ordinateur. Au niveau de la scolarité, cinq d'entre eux sont des étudiants au baccalauréat, quatre à la maîtrise et trois au doctorat. Sept participants étudient dans le domaine du génie des technologies de l'information, quatre dans le génie logiciel et un dans le génie électrique. Ils utilisent un ordinateur de cinq à 15 heures par jour (moyenne de neuf heures et écart type de 1,6 heure) et neuf ont indiqué qu'ils utilisent parfois des logiciels pour visualiser des arborescences, des graphes ou des réseaux. Parmi ces réponses, deux participants n'en utilisent que rarement. Finalement, quatre participants ont mentionné qu'ils utilisent parfois des logiciels dans lesquels il y a des animations d'information ou des transitions animées.

3.1.6 Conditions de l'expérimentation

Les essais sont réalisés sous quatre conditions principales soit une pour chaque type d'animation : linéaire, par étapes, hiérarchique et hybride. À l'intérieur de chaque condition, les trois types de changements sont évalués dans l'ordre en commençant par les fermetures, les ouvertures puis les permutations. Pour chaque type de changement impliqué pour la tâche secondaire, un nombre égal d'essais est alloué pour suivre un, deux ou trois nœuds à la fois. En effet, pour chaque nombre de nœuds à suivre, huit répétitions permettront d'avoir de multiples résultats à analyser. Avec cette planification d'expérimentation utilisée pour les 12 participants, le nombre total d'essais pour l'expérience est de 3 456 (voir Figure 3.2).

4 conditions (animation linéaire, par étapes, hiérarchique et hybride)
× 3 types de changements pour la tâche secondaire (ouvertures, fermetures et permutations)
× 3 nombres de nœuds à suivre pour la tâche secondaire (1, 2 ou 3 nœuds à la fois)
× 8 répétitions
× 12 participants
= 3 456 essais au total

Figure 3.2 Résumé de la planification de l'expérimentation

L'ordre de présentation des quatre conditions principales est contrebalancé selon le principe d'un carré latin pour répartir les effets d'apprentissage ou de fatigue des participants. Autrement dit, si les lettres A, B, C et D représentent les quatre types d'animation, un quart des participants ont fait les essais en suivant l'ordre [A, B, C, D], un autre quart l'ordre [B, C, D, A], etc. Ainsi, chaque quart des participants était assigné à une rangée du carré latin illustré par la Figure 3.3.

A	**B**	**C**	**D**
B	**C**	**D**	**A**
C	**D**	**A**	**B**
D	**A**	**B**	**C**

Figure 3.3 Carré latin montrant l'ordre de
présentation des quatre conditions principales

Étant donné que la comparaison principale de l'expérimentation se base sur les types d'animation, l'ordre de présentation des blocs d'essais pour chaque type de changement était toujours le même. De plus, l'ordre des essais dans chaque bloc a été déterminé à l'avance de façon aléatoire et le même a été utilisé pour tous les participants.

Pour uniformiser les essais et contrôler les variables de l'expérimentation, les arbres générés aléatoirement comportent toujours trois niveaux. Également, les arborescences peuvent compter de trois à huit enfants de premier niveau et chacun de ceux-ci peut compter de deux à six nœuds enfants sur les deuxième et troisième niveaux. Cette restriction permet d'avoir une complexité suffisante pour évaluer les tâches sans toutefois avoir de trop gros arbres dont les nœuds seraient trop petits et ainsi être plus difficiles à sélectionner par l'utilisateur. Les nœuds de l'arbre sont majoritairement ouverts et le nombre de nœuds fermés est déterminé selon une probabilité fixe qui est la même pour chaque niveau de l'arbre assurant ainsi d'avoir un nombre de nœuds fermés proportionnel au nombre de nœuds total sur un niveau. Pour les transitions animées, la durée est fixée à cinq secondes et la vitesse est constante du début à la fin peu importe le type d'animation utilisé. Pour la tâche principale, le nœud à suivre est toujours situé sur le niveau le plus profond de l'arbre soit le troisième. Pour être choisi, le nœud cible doit subir un déplacement d'une distance minimale de 100 pixels et il ne doit pas être impliqué dans une fermeture d'un de ses parents puisqu'il serait caché et non identifiable. Du côté de la tâche secondaire, les nœuds impliqués se retrouvent uniquement au premier niveau pour mieux permettre à l'utilisateur de percevoir les changements et faciliter leur suivi. Ainsi, le type de changement observé peut se produire uniquement sur le premier niveau sauf pour les permutations qui agissent également comme distraction et pour permettre le déplacement des nœuds à l'écran. Aussi, le nombre de nœuds pouvant subir le changement mentionné pour l'évaluation se situe entre un et trois. Ce maximum est choisi en fonction des études de Cavanagh et Alvarez (2005), Pylyshyn et Storm (1988) et Oksama et Hyönä (2004) qui démontrent qu'en moyenne un être humain n'a pas la capacité de suivre simultanément plus de quatre cibles en mouvement à l'écran. En combinant donc les deux tâches, ce maximum est atteint. De plus, ces nœuds doivent être espacés de 100 pixels entre eux et du nœud à suivre pour la tâche principale à la fin de l'animation. Cela évite de rendre la tâche trop facile si tous les nœuds à suivre se trouvent à proximité et dans le même focus d'attention de l'utilisateur. Pour l'identification des ouvertures, il doit y avoir plus d'un nœud dont son état est ouvert à la fin de l'animation et il en est de même avec les fermetures où deux nœuds enfants doivent minimalement être fermés. Cette mesure sert à éviter que

l'utilisateur procède par élimination et choisisse le seul nœud restant ouvert ou fermé selon le type de changement évalué. Finalement, en aucun cas un même nœud ne peut s'ouvrir et se fermer pendant la même animation pour ne pas confondre l'utilisateur.

Un pré test effectué avec deux participants a permis d'ajuster certaines variables pour le test final. Initialement, dans le but de rendre les tâches plus réalistes, le type de changement à observer pour la tâche secondaire n'était pas communiqué à l'utilisateur avant l'essai et ce changement pouvait se produire sur n'importe quel niveau de l'arbre. Il est vite apparu que les tâches étaient trop difficiles à réaliser et que le participant n'arrivait pas à remarquer les changements qui se produisaient. Toutefois, en mentionnant le type de changement à l'avance et en limitant les changements aux nœuds de premier niveau, le taux de réussite s'est amélioré permettant ainsi de comparer les différents types d'animations entre eux.

3.2 Analyse et discussion des résultats

Pendant la complétion des essais par les participants, le prototype expérimental enregistrait la réussite des tâches principale et secondaire en plus de comptabiliser le temps de complétion pour chaque tâche. Le nombre de nœuds correctement identifiés et mal identifiés (faux positifs) étaient aussi enregistrés. De plus, les questionnaires ont été compilés dans le but de connaître les préférences des participants pour chaque type d'animation.

3.2.1 Résultats pour la tâche principale

L'animation hiérarchique a généré un plus grand nombre de réussites pour la tâche principale (voir Figure 3.4) avec 699 sur 864 (80,9 %) comparativement à l'animation hybride (653 réussites, 75,58 %), linéaire (619 réussites, 71,64 %) et par étapes (608 réussites, 70,37 %). Une analyse de variance (ANOVA) a démontré un avantage significatif ($F = 10,317$) pour l'animation hiérarchique par rapport à l'animation linéaire ($p < 0,001$), par étapes ($p < 0,001$) et hybride ($p < 0,037$). De plus, un faible avantage est accordé à l'animation hybride face à

l'animation par étapes ($p < 0,07$). La meilleure réussite pour les animations hiérarchique et hybride par rapport aux animations linéaire et par étapes vient supporter la nouvelle méthode utilisée pour diriger l'attention de l'utilisateur. En animant les changements par niveau en commençant par le premier jusqu'au plus profond, cela limite la zone d'attention de l'utilisateur et permet un meilleur suivi des nœuds. Les résultats supportent donc partiellement les hypothèses H05 et H06. Toutefois, la combinaison utilisée par l'animation hybride ne permet pas un meilleur suivi de nœuds par rapport à l'animation hiérarchique. L'effet contraire confirme donc le rejet partiel de l'hypothèse H05. Pendant le test, quelques participants ont mentionné s'être trompés lors de la sélection du nœud à suivre à cause de la vitesse trop rapide de leur geste. Comme il n'était pas possible de corriger leur choix avant de passer à la tâche secondaire, quelques essais supplémentaires auraient pu être réussis.

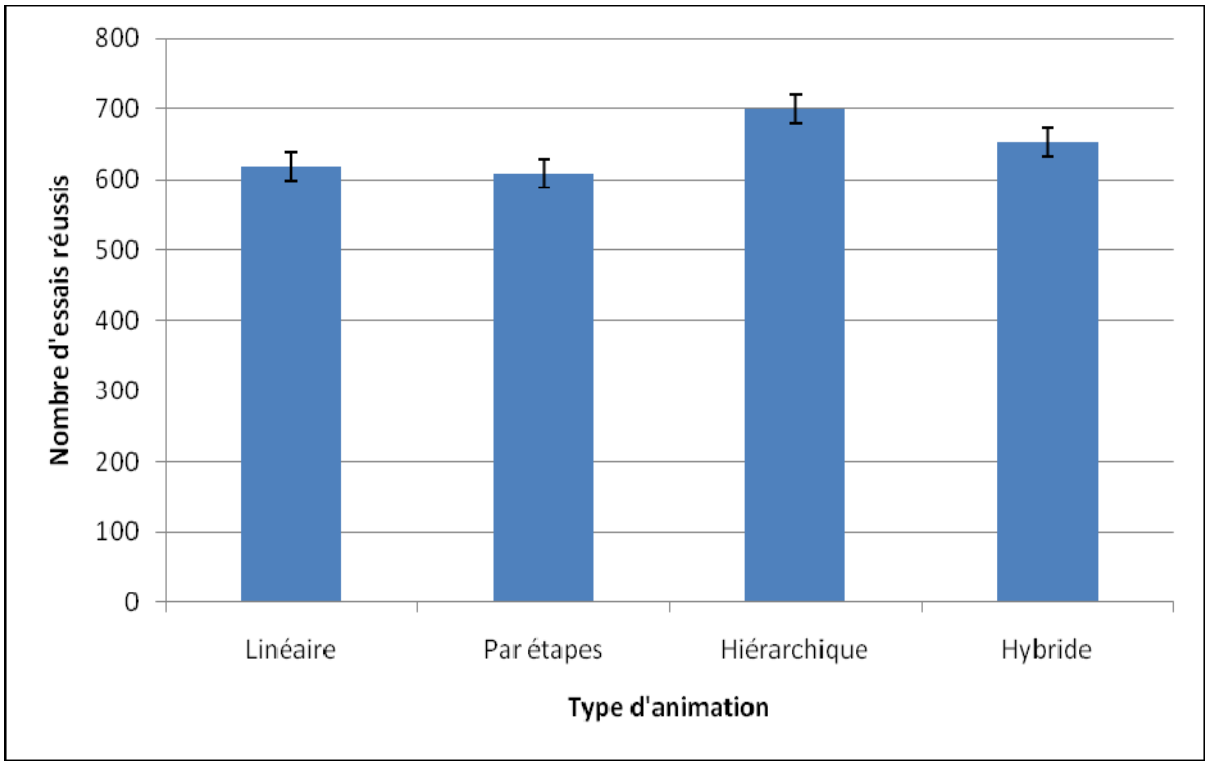

Figure 3.4 Nombre d'essais réussis pour la tâche principale selon le type d'animation

Le temps moyen pour la complétion des essais de la tâche principale (voir Figure 3.5), calculé à partir de la fin de l'animation, est significativement plus élevé (F = 7,077) pour l'animation linéaire avec 1,45 seconde en relation avec 1,28 seconde pour l'animation hiérarchique (p < 0,001), 1,26 seconde pour l'animation par étapes (p < 0,008) et 1,24 seconde pour l'animation hybride (p < 0,001). Un temps plus élevé peut s'expliquer par une hésitation du participant avant de sélectionner le nœud à suivre dans sa position finale. Étant donné que l'animation linéaire ne comporte qu'une seule étape et qu'elle génère beaucoup d'occlusion entre les éléments à l'écran lors des mouvements, l'utilisateur peut perdre la position du nœud qu'il est en train de suivre.

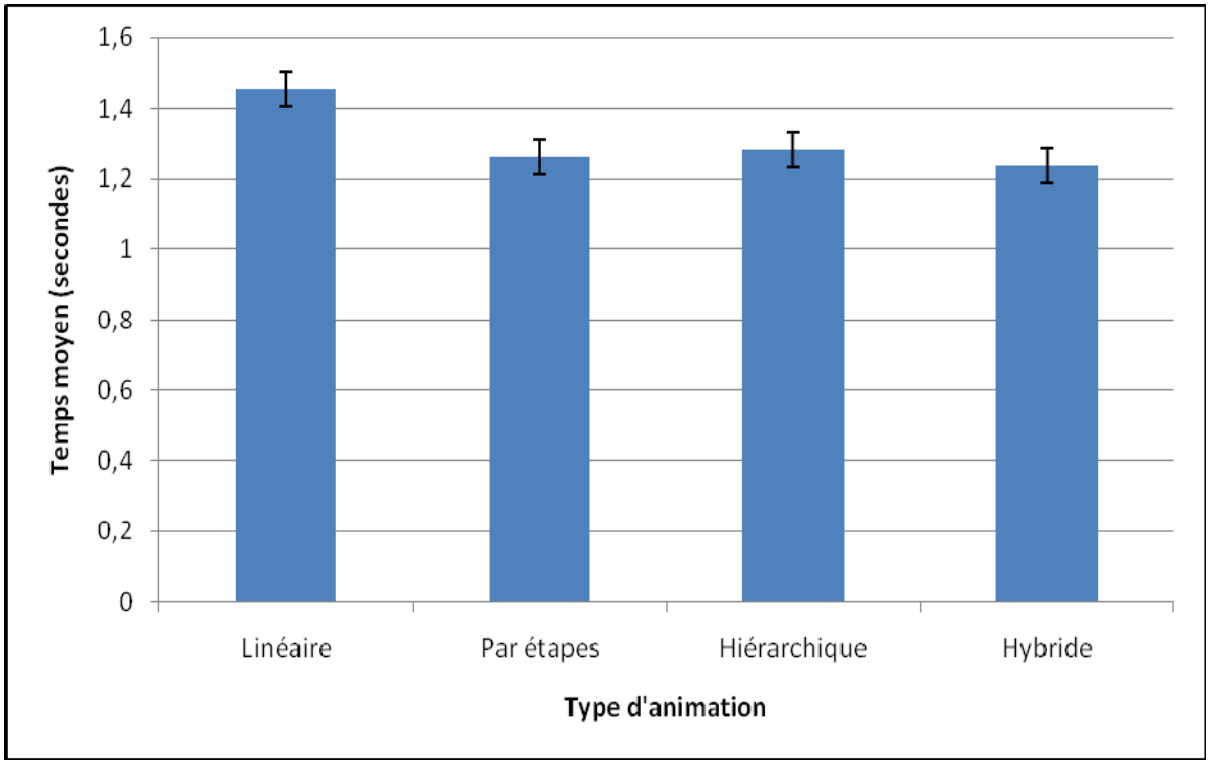

Figure 3.5 Temps moyen pour la complétion des essais de la tâche principale selon le type d'animation

3.2.2 Résultats pour la tâche secondaire

Le nombre d'essais réussis pour la tâche secondaire (voir Figure 3.6) est de 615 sur 864 (71,18 %) pour l'animation hybride, comparativement à l'animation linéaire (588 réussites, 68,06 %), l'animation hiérarchique (580 réussites, 67,13 %) et l'animation par étapes (467 réussites, 54,05 %). Une analyse de variance (ANOVA) a démontré un avantage significatif pour l'animation linéaire, hiérarchique et hybride par rapport à l'animation par étapes ($F = 22{,}187$, $p < 0{,}001$). Comme pour la tâche principale, cette analyse vient supporter partiellement les hypothèses H05 et H06. Au contraire, un effet inverse rejette partiellement l'hypothèse H01 donnant donc une meilleure performance à l'animation linéaire. Quelques participants ont eu de la difficulté à bien sélectionner certains nœuds, mais ils ont pu reprendre la sélection avant de confirmer la fin de la tâche.

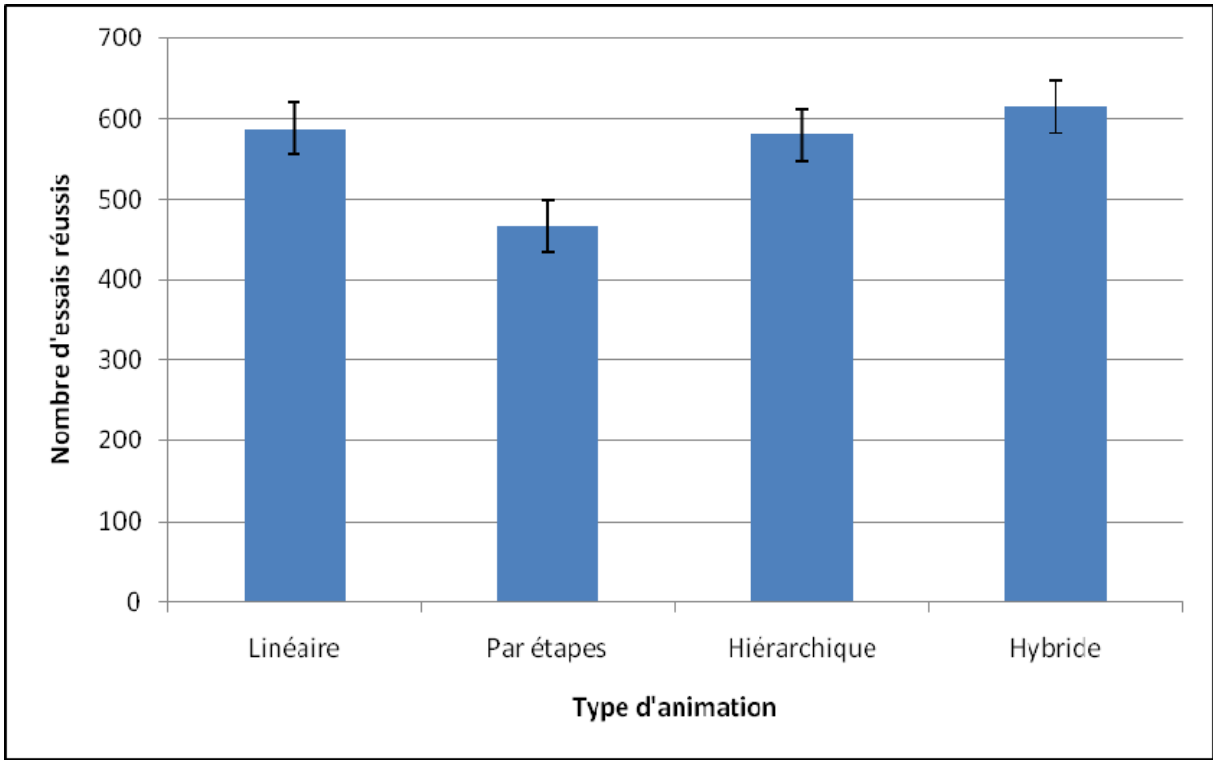

Figure 3.6 Nombre d'essais réussis pour la tâche secondaire selon le type d'animation

De son côté, le temps de complétion moyen des essais pour la tâche secondaire (calculé après la complétion de la tâche principale) n'a pas montré d'avantage significatif pour l'une ou l'autre des animations avec des temps de 2,54 secondes pour l'animation linéaire, 2,64 secondes pour l'animation par étapes, 2,65 secondes pour l'animation hiérarchique et 2,67 secondes pour l'animation hybride.

Pour détailler plus précisément les performances des animations pour la tâche secondaire, les essais sont comparés en fonction du type de changement observé (voir Figure 3.7). Dans les prochaines analyses, la réussite d'un essai consiste en l'identification exacte de tous les nœuds qui ont subi le changement observé (d'un à trois) sans erreur. Un échec est donc considéré lorsqu'un nœud impliqué par le changement n'a pas été identifié ou qu'un autre nœud non impliqué a été identifié. Une analyse de variance (ANOVA) a démontré un avantage significatif pour le type d'animation ($F = 29,849$, $p < 0,001$), le type de changement ($F = 429,097$, $p < 0,001$) et pour l'interaction de ces deux facteurs ($F = 56,970$, $p < 0,001$). D'abord, l'animation hybride a connu un meilleur succès pour les ouvertures avec 278 réussites sur 288 (96,53 %) comparativement à l'animation par étapes (272 réussites, 94,44 %), linéaire (266 réussites, 92,36 %) et hiérarchique (235 réussites, 81,6 %). Une analyse rétrospective a démontré un avantage significatif pour l'animation linéaire ($p < 0,002$), par étapes ($p < 0,001$) et hybride ($p < 0,001$) par rapport à l'animation hiérarchique. Ces résultats démontrent que les ouvertures animées à la fin de la transition pour les animations par étapes et hybride sont plus efficaces. Cela supporte donc l'hypothèse H08 et partiellement H07. Cependant, le fait de limiter les ouvertures sur le premier niveau de l'arborescence peut faire diminuer la performance de l'animation hiérarchique qui débute sa séquence d'animation par le premier niveau. La tâche est donc rendue plus difficile par rapport aux autres animations.

Pour les fermetures (voir Figure 3.7), l'animation linéaire a généré un nombre significativement plus grand de réussites ($p < 0,001$) avec 243 sur 288 (84,38 %) par rapport à l'animation hybride (166 réussites, 57,64), hiérarchique (160 réussites, 55,56%) et par étapes (149 réussites, 51,74%). De plus, un faible avantage est accordé à l'animation hybride

face à l'animation par étapes ($p < 0,082$) supportant partiellement l'hypothèse H09. Contrairement aux hypothèses H03, H09 et H10, l'animation linéaire a été la plus efficace malgré sa seule étape et sa plus grande occlusion confirmant ainsi un effet contraire à celui espéré. Cependant, certains participants ont mentionné que les fermetures étaient plus faciles à identifier avec l'animation linéaire, car l'animation de la fermeture d'un nœud s'étend sur toute la durée de la transition comparativement aux autres types d'animations. Ainsi, comme la fermeture est visible du début à la fin, l'utilisateur peut mieux suivre le nœud et anticiper sa trajectoire vers sa position finale. Toutefois, quelques participants ont trouvé difficile de bien distinguer les ouvertures des fermetures à la moitié de la transition linéaire puisque l'animation des deux changements se confond dans la forme du cercle qui s'ouvre ou se ferme. Comme seulement un demi-cercle est visible pour les deux types de changements à ce point, il faut se souvenir de l'origine de ceux-ci ou encore attendre que l'animation se poursuive pour les distinguer à nouveau.

Pour les permutations (voir Figure 3.7), les animations hiérarchique (185 réussites sur 288, 64,24 %) et hybride (171 réussites, 59,38 %) ont mieux performé de façon significative ($p < 0,001$) face à l'animation linéaire (79 réussites, 27,43 %) et par étapes (46 réussites, 15,98 %). De plus, l'animation linéaire a un avantage significatif ($p < 0,001$) par rapport à l'animation par étapes rejetant ainsi l'hypothèse H04. Le nombre de réussites plus bas pour les permutations en général peut s'expliquer par la difficulté de la tâche comparativement aux autres types de changements. Les permutations sont des changements moins évidents et peuvent être confondues avec les permutations d'autres niveaux qui ne font pas partie de l'essai. En effet, deux participants n'ont pas compris cette subtilité en affichant un très faible taux de réussite pour l'ensemble des rotations (moyenne de 11,46 %) comparativement aux résultats des autres participants (moyenne de 47,81 %). L'analyse confirme donc l'hypothèse H12 et partiellement H11 grâce à l'animation niveau par niveau des animations hiérarchique et hybride qui présentent de façon plus claire les permutations comparativement aux animations linéaire et par étapes. L'avantage de l'animation linéaire face à l'animation par

étapes peut s'expliquer par la vitesse d'animation moins élevée vu qu'il n'y a qu'une seule étape permettant de mieux suivre les permutations malgré l'occlusion.

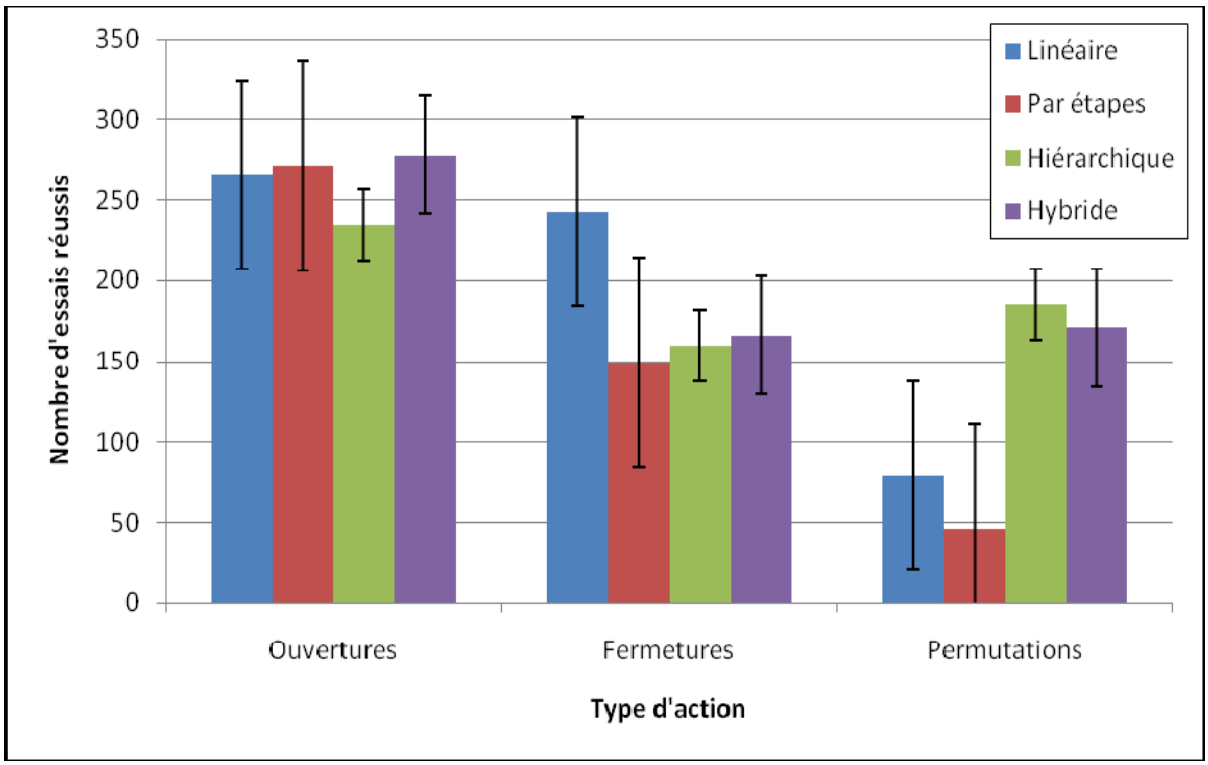

Figure 3.7 Nombre d'essais réussis pour la tâche secondaire pour chaque type d'animation selon le type de changement

Les résultats sur le nombre de nœuds correctement identifiés (à l'intérieur de tous les essais) pour chaque type de changement par rapport au type d'animation apportent les mêmes conclusions que le nombre d'essais réussis. L'analyse de variance (ANOVA) et les analyses rétrospectives ont démontré des différences significatives semblables à celles présentées précédemment. Il en va de même pour le nombre de nœuds identifiés à tort (faux positifs) qui est inversement proportionnel à la réussite des essais pour chaque type d'animation. Ainsi, une animation qui a un taux plus grand de réussite est susceptible de générer moins d'erreurs d'identification de nœuds comparativement à un autre type d'animation qui a un taux de réussite plus faible.

3.2.3 Appréciation et classement des animations selon les participants

Le classement des différents types d'animations selon la capacité de permettre à l'utilisateur de bien comprendre les changements de position des nœuds dans l'arborescence, sur une échelle d'appréciation de quatre points, a donné des résultats partagés. Ainsi, les animations hiérarchique et hybride ont obtenu une note moyenne de 2,97 points devant l'animation linéaire avec 2,42 points et l'animation par étapes avec deux points. Toutefois, une analyse de variance (ANOVA) n'a pas montré de différences significatives entre ces résultats. En effet, la répartition des points ne permet pas de voir une tendance claire puisque tous les types d'animations se sont vus attribuer des notes maximales et minimales. Cette constatation peut être expliquée par le fait que les participants devaient se rappeler des détails de chaque animation à la fin du test dans le but de remplir le questionnaire. Il est possible que l'ordre de présentation des animations ait influencé le classement final puisque la première était plus floue dans la mémoire des participants que la dernière.

Le même phénomène a été observé pour la perception de la vitesse d'animation selon les participants. Sur une échelle de quatre points, l'animation hiérarchique a semblé être plus rapide avec une note moyenne de 2,75 points devançant l'animation par étapes (2,67 points), hybride (2,17 points) et linéaire (2,08 points). L'absence de différence significative et de tendance dans la répartition des notes ne permet pas de tirer une conclusion claire. Il faut ajouter que la perception de la vitesse est très suggestive et peut être interprétée différemment d'un participant à l'autre.

Pour chaque type de changement, le participant était aussi invité à indiquer une ou plusieurs animations qu'il considère comme étant la meilleure (voir Figure 3.8). Pour les ouvertures, neuf participants ont désigné l'animation hybride et six ont opté pour l'animation par étapes contre seulement deux pour l'animation linéaire et un pour l'animation hiérarchique. Ils semblent avoir apprécié les ouvertures en fin de séquence des animations par étapes et hybride qui permettent d'identifier facilement ce type de changement. Pour les fermetures,

l'opinion des participants est partagée et accorde le même succès aux animations linéaire, hiérarchique et hybride. Bien que les résultats aient démontré un avantage significatif pour l'animation linéaire, celle-ci ne s'est pas démarquée selon les participants. Au niveau des permutations, six participants ont préféré l'animation hiérarchique et cinq ont choisi l'animation hybride contre seulement un pour l'animation linéaire et aucun pour l'animation par étapes. Ce point de vue démontre clairement la réussite des animations hiérarchique et hybride pour bien découper les permutations niveau par niveau et vient appuyer les résultats analysés précédemment.

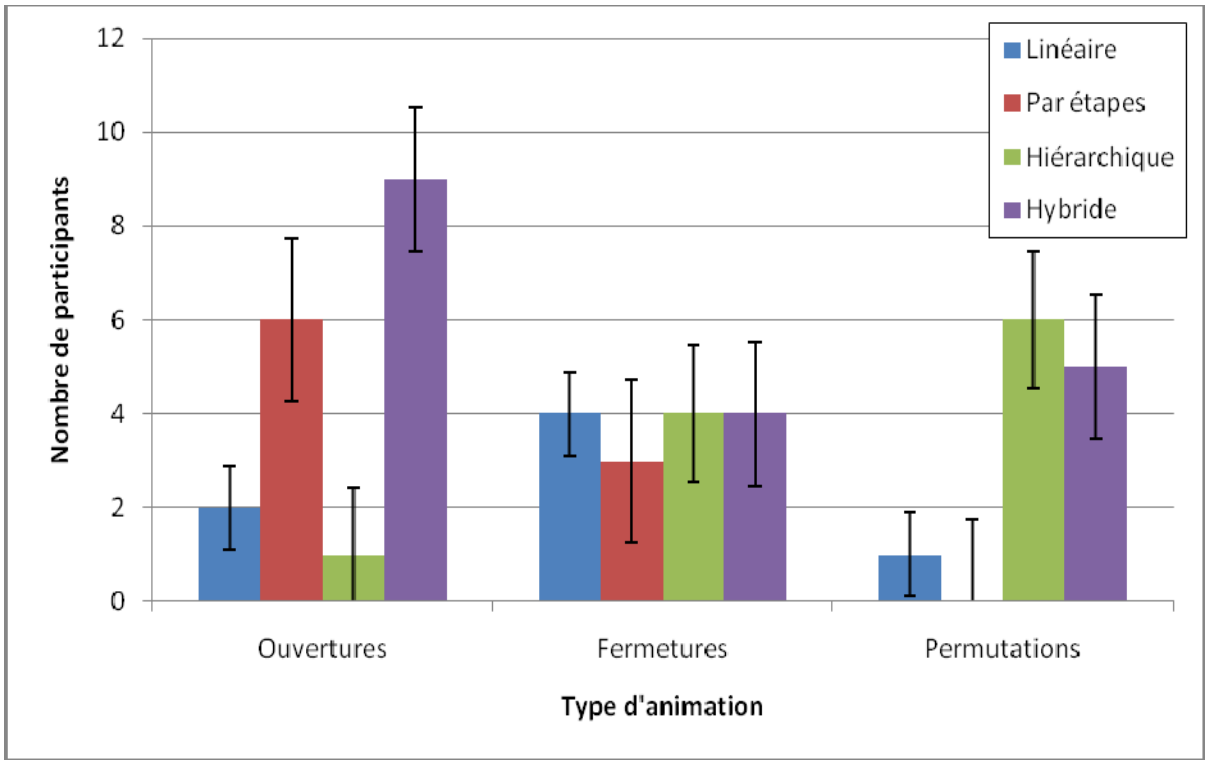

Figure 3.8 Meilleure animation pour chaque type de changement selon les participants

Finalement, une question demandait aux participants d'indiquer l'animation qu'ils préféreraient avoir dans l'interface d'un logiciel qu'ils ont à utiliser. Des 12 participants, sept aimeraient utiliser l'animation hybride, quatre l'animation hiérarchique, un l'animation par étapes et un l'animation linéaire. Dans ces résultats, un participant a indiqué préférer à la fois

l'animation hiérarchique et hybride. Les participants mentionnent généralement que l'animation hiérarchique est beaucoup plus facile pour identifier les permutations, mais permet aussi un bon compromis pour les ouvertures et les fermetures. Ils indiquent qu'elle permet aussi de se concentrer séquentiellement sur les différentes tâches à effectuer et qu'elle réduit l'occlusion des nœuds.

3.2.4 Retour sur les hypothèses

Les résultats ont permis de confirmer certaines hypothèses énoncées lors de l'analyse théorique de la conception des animations (voir Tableau 3.2). Ainsi, pour le suivi des nœuds, l'hypothèse H05 pour la tâche secondaire et H06 confirment un avantage significatif pour l'animation hiérarchique par rapport à l'animation par étapes. De plus, H06 donne un avantage à l'animation hiérarchique face à l'animation linéaire pour la tâche principale. Pour les ouvertures, les hypothèses H07 et H08 affirment que les animations par étapes et hybride sont significativement plus efficaces que l'animation hiérarchique. Finalement, les hypothèses H11 et H12 confirment que les animations hiérarchique et hybride sont meilleures que les animations linéaire et par étapes pour l'identification des permutations.

À l'inverse, certains effets contraires observés viennent rejeter quelques hypothèses de façon significative (voir Tableau 3.2). Pour le suivi de nœuds, l'hypothèse H01 pour la tâche secondaire démontre plutôt un avantage pour l'animation linéaire face à l'animation par étapes. De même, l'hypothèse H05 contredit l'efficacité de l'animation hybride face à l'animation hiérarchique. Pour les fermetures, les résultats rejettent les hypothèses H03, H09 et H10 en indiquant une meilleure performance pour l'animation linéaire face aux autres types d'animation. Pour les permutations, l'animation linéaire présente un avantage significatif sur l'animation par étape contrairement à l'énoncé de l'hypothèse H04.

Toutefois, l'absence de différences significatives empêche d'avancer un résultat de façon partielle ou totale sur les hypothèses H01, H02, H05, H06, H07, H09, H10, H11. Bien que les

résultats bruts peuvent montrer un avantage au niveau du nombre d'essais réussis pour l'une ou l'autre des animations, cette marge non représentative peut être due au hasard ou encore à des erreurs lors de l'expérimentation.

Tableau 3.2
Résumé du support des hypothèses par les résultats de l'expérimentation

Hypothèse	**Catégorie**	**Énoncé**	**Support de l'hypothèse***
H01	Suivi de nœuds[a]	Par étapes > Linéaire	TP: Non ($p > 0,1$) TS: **Non, effet contraire confirmé ($p < 0,001$)**
H02	Ouvertures	Par étapes > Linéaire	TS: Non ($p > 0,1$)
H03	Fermetures	Par étapes > Linéaire	TS: **Non, effet contraire confirmé ($p < 0,001$)**
H04	Permutations	Par étapes > Linéaire	TS: **Non, effet contraire confirmé ($p < 0,001$)**
H05	Suivi de nœuds[a]	Hybride > Linéaire	TP et TS: Non ($p > 0,1$)
H05	Suivi de nœuds[a]	Hybride > Par étapes	TP: Oui ($p < 0,07$) TS: **Oui ($p < 0,001$)**
H05	Suivi de nœuds[a]	Hybride > Hiérarchique	TP: **Non, effet contraire confirmé ($p < 0,037$)** TS: Non ($p > 0,1$)
H06	Suivi de nœuds[a]	Hiérarchique > Linéaire	TP: **Oui ($p < 0,001$)** TS: Non ($p > 0,1$)
H06	Suivi de nœuds[a]	Hiérarchique > Par étapes	TP et TS: **Oui ($p < 0,001$)**
H07	Ouvertures	Hybride > Linéaire	TS: Non ($p > 0,1$)
H07	Ouvertures	Hybride > Par étapes	TS: Non ($p > 0,1$)

Tableau 3.3
Résumé du support des hypothèses par les résultats de l'expérimentation (suite)

H07	Ouvertures	Hybride > Hiérarchique	TS: **Oui (p < 0,001)**
H08	Ouvertures	Par étapes > Hiérarchique	TS: **Oui (p < 0,001)**
H09	Fermetures	Hybride > Linéaire	TS: **Non, effet contraire confirmé (p < 0,001)**
H09	Fermetures	Hybride > Par étapes	TS: Oui (p < 0,082)
H09	Fermetures	Hybride > Hiérarchique	TS: Non (p > 0,1)
H10	Fermetures	Hiérarchique > Linéaire	TS: **Non, effet contraire confirmé (p < 0,001)**
H10	Fermetures	Hiérarchique > Par étapes	TS: Non (p > 0,1)
H11	Permutations	Hiérarchique > Linéaire	TS: **Oui (p < 0,001)**
H11	Permutations	Hiérarchique > Par étapes	TS: **Oui (p < 0,001)**
H11	Permutations	Hiérarchique > Hybride	TS: Non (p > 0,1)
H12	Permutations	Hybride > Linéaire	TS: **Oui (p < 0,001)**
H12	Permutations	Hybride > Par étapes	TS: **Oui (p < 0,001)**

[*] L'abréviation « TP » indique les résultats pour la tâche principale tandis que « TS » désigne ceux de la tâche secondaire. Les affirmations en caractères gras démontrent un avantage significatif à la suite d'une analyse de variance (ANOVA). [a] Le suivi des nœuds pour la tâche secondaire (TS) est évalué selon le taux de réussite pour l'identification de tous les types de changements confondus.

3.3 Directives de conception des animations

Les conclusions apportées par l'analyse des résultats et le retour sur les hypothèses permettent de mieux comprendre les facteurs qui ont mené au succès ou à l'échec de

certaines animations. En s'inspirant des données et des commentaires fournis par les participants lors de l'expérimentation, il est maintenant possible d'établir des directives de conception pour les transitions animées comportant plusieurs étapes distinctes. Ces directives pourront servir de guide pour l'amélioration des animations existantes ou encore pour en créer de nouvelles (voir Tableau 3.4).

Tableau 3.4
Résumé des directives de conception pour les transitions animées

Directive	**Énoncé**
D01	Attribuer une étape distincte ou animer en dernier un changement qui doit être facilement identifiable.
D02	Animer en premier un changement qui n'est pas important.
D03	Animer simultanément un changement sur plusieurs étapes s'il se produit en début de séquence et s'il doit être identifiable.
D04	Bâtir les transitions animées de façon à diriger l'attention de l'utilisateur.
D05	Minimiser le nombre d'étapes des transitions pour diminuer la vitesse d'animation.
D06	Éviter le croisement des trajectoires et réduire l'occlusion.
D07	Animer les changements ou les déplacements de façon séquentielle plutôt que simultanée et utiliser des décalages.
D08	Ajouter un élément visuel supportant la mise en évidence d'un changement.

Premièrement, pour permettre à l'utilisateur de bien identifier un changement en particulier pendant une transition animée, il est préférable de lui accorder une étape dédiée ou encore d'animer ce changement en dernier une fois que tous les autres nœuds sont déjà dans leur

position finale (directive D01). L'animation des ouvertures en dernier une fois que toutes les permutations sont terminées a permis aux animations par étapes et hybride d'augmenter la capacité de l'utilisateur à bien les observer comparativement à l'animation hiérarchique. De plus, le fait d'accorder une étape distincte pour chaque niveau de l'arborescence pour les permutations a aussi facilité l'identification de ces changements pour les animations hiérarchique et hybride.

Au contraire, si un changement n'est pas important pour l'utilisateur, il est préférable de le mettre au début de la transition (D02). C'est le cas pour l'animation par étapes et hybride si on suppose que les fermetures ne sont pas importantes pour l'utilisateur. En plus d'être plus difficilement identifiable, un nœud qui se ferme en premier élimine de l'information superflue lors de la transition, réduit l'occlusion des permutations et fait de la place pour l'ouverture de nouveaux nœuds.

Cependant, si un changement qui a lieu au début de la séquence doit tout de même être perçu par l'utilisateur, il est préférable que l'animation complète de ce changement se passe sur plusieurs étapes simultanément (D03). L'animation linéaire a démontré une meilleure performance pour l'identification des fermetures face aux autres types d'animations à cause de sa transition comportant une seule étape. Comme le changement est visible du début à la fin, il est plus facile de le repérer et de le suivre jusqu'à sa position finale. Or, comme l'animation par étapes comporte d'autres désavantages, ce principe peut être utilisé pour améliorer un autre type d'animation comme l'animation hybride qui s'inspire déjà des animations par étapes et hiérarchique (voir Figure 3.9). Au lieu d'animer les fermetures au début seulement, ce type de changement peut débuter en premier par une étape distincte pour se poursuivre simultanément aux autres étapes de permutations niveau par niveau jusqu'à la dernière pour les ouvertures. Cette nouveauté pourrait permettre de conserver le bon rendement de l'animation hybride pour l'identification des ouvertures et des permutations en plus d'augmenter celle des fermetures.

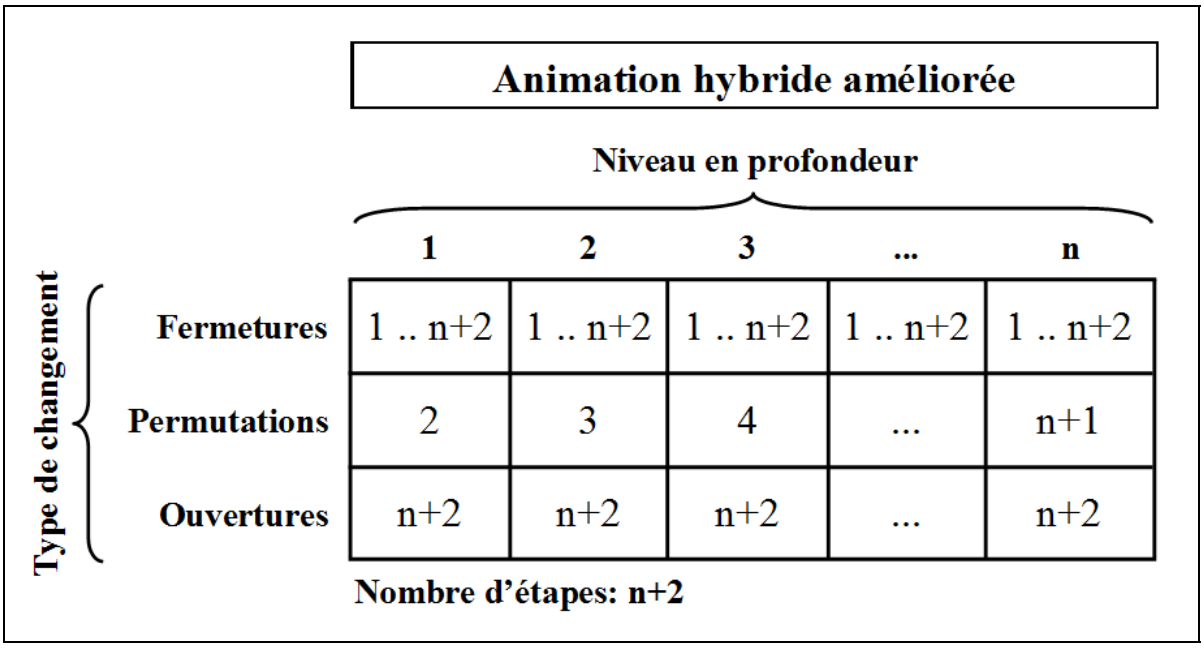

Figure 3.9 Matrice de la décomposition de l'animation hybride améliorée selon le type de changement et le niveau en profondeur de l'arborescence

D'un point de vue logique, la transition doit être bâtie de façon à diriger l'attention de l'utilisateur et ainsi faciliter le suivi des nœuds (D04). Le fait de créer un patron que l'utilisateur peut assimiler permet de se concentrer sur plusieurs focus d'attention à la fois. Il peut donc stratégiquement suivre certaines parties de la transition et d'autres non lorsqu'un certain changement s'anime selon la tâche à effectuer. La décomposition des permutations par niveau en commençant par le premier vers le plus profond a permis à l'utilisateur de mieux suivre les nœuds lors des permutations avec les animations hiérarchique et hybride.

Il est préférable de minimiser le nombre d'étapes de la transition pour éviter que la vitesse d'animation soit trop élevée (D05). En fixant le temps total de la transition, un nombre trop important d'étapes peut causer une vitesse d'animation plus grande diminuant ainsi l'efficacité de la transition. Ce phénomène peut expliquer l'avantage de l'animation linéaire face à l'animation par étapes pour le suivi des nœuds de la tâche secondaire et des permutations. Il va de même pour celle de l'animation hiérarchique par rapport à l'animation hybride pour le suivi des nœuds de la tâche principale. Bien qu'une animation puisse avoir

des faiblesses dans sa conception, une vitesse d'animation moins rapide pourrait lui donner un avantage, car cela laisse plus de temps à l'utilisateur pour assimiler les changements. Dans le cas où deux animations présentent sensiblement les mêmes caractéristiques, celle avec le moins d'étapes devrait être plus efficace.

Surtout dans le cas des arbres avec une disposition radiale, il est préférable d'éviter le croisement des trajectoires des nœuds et de minimiser le passage de ces trajectoires par le centre de l'arborescence pour réduire l'occlusion (D06). Bien que le but de la présente étude n'est pas de proposer un algorithme de présentation d'arborescence pour réduire l'occlusion, les animations hiérarchique et hybride ont quand même contribué à diminuer ce problème. En animant les permutations niveau par niveau, l'occlusion se produit seulement sur un niveau à la fois comparativement aux animations linéaire et par étapes où l'occlusion est générale. Pareillement, la décomposition permet d'éviter que tous les nœuds permutés passent par le centre de l'arborescence en même temps facilitant le suivi de la trajectoire d'un nœud.

De plus, il est préférable de ne pas animer plusieurs changements ou déplacements en même temps (D07). Lorsque trop d'éléments sont en mouvement à la fois il peut être difficile de bien les différencier entre eux. Toutefois, si le nombre d'éléments ne peut être limité il est possible d'utiliser un décalage spatial ou temporel pour faciliter le suivi. En animant des groupes d'éléments avec une vitesse, une trajectoire ou encore un temps de départ légèrement différent, l'utilisateur pourrait mieux les distinguer et suivre seulement les éléments qui l'intéressent tout en délaissant les autres. Cela pourrait donc réduire la charge d'attention nécessaire pour la complétion de la tâche en cours.

Finalement, l'ajout d'un élément visuel ou un changement de couleur peuvent être utilisés pour mettre encore plus en évidence un changement (D08). Par exemple, pour mieux distinguer les fermetures en début de séquence de l'animation par étapes ou hybride, une bordure supplémentaire au cercle des nœuds qui se sont fermés peut être ajoutée pour bien

les repérer à la fin. Aussi, il est possible de mettre en surbrillance temporairement les nœuds d'une couleur différente selon le type de changement qu'ils ont subi.

Bien que ces directives puissent améliorer la justesse et la performance des transitions animées, une autre approche pour rendre les transitions plus faciles à comprendre est de fournir à l'utilisateur un moyen de contrôler de façon flexible et intuitive ces animations afin d'en retirer un maximum d'information. Le simple fait de rejouer une transition ou encore de se concentrer sur des parties plus spécifiques peut permettre à l'utilisateur de mieux comprendre les mouvements ou les changements de valeurs qui ont eu lieu. Le prochain chapitre se concentre sur l'analyse et la conception de telles techniques d'interaction en utilisant des *widgets* spécifiques aux animations.

CHAPITRE 4

ANALYSE ET CRÉATION DE *WIDGETS* POUR LE CONTRÔLE DES ANIMATIONS

Les deux chapitres précédents ont proposé et évalué différents types d'animations pour trouver comment rendre les transitions animées plus faciles à comprendre. Une autre façon de rendre les transitions plus compréhensibles est de fournir à l'utilisateur des contrôles interactifs permettant de ralentir ou rejouer une animation. Tel qu'expliqué dans le premier chapitre, les *widgets* proposés jusqu'à maintenant pour ce genre d'interaction ont chacun leurs avantages et inconvénients. Ce chapitre cherchera donc à combiner les avantages des différents *widgets* antérieurs. En particulier, il sera question de permettre à la fois une entrée simple et discrète, pour déclencher une transition à vitesse fixe, et aussi une entrée continue (par glissement) donnant le plein contrôle à l'utilisateur sur la progression de l'animation lorsque cela est désiré.

Le chapitre débutera par la définition des propriétés pour le contrôle des animations, puis par les caractéristiques de base des *widgets* contextuels pour le contrôle des transitions. Par la suite, de nouveaux types de contrôles seront présentés sous la forme de maquettes illustrant différentes techniques d'interaction qui intègrent les propriétés énoncées.

4.1 Définition des propriétés pour le contrôle des animations

L'analyse des méthodes existantes pour contrôler les animations a permis de faire ressortir trois propriétés essentielles, mais qui ne sont pas nécessairement incorporées dans un seul et même *widget*. Elles sont souvent présentes dans un logiciel grâce à plusieurs outils différents, mais certaines sont parfois oubliées. Les prochaines sections définissent l'utilisation du contrôle discret et continu des animations en plus du repérage de la position à l'intérieur de celles-ci.

4.1.1 Le contrôle discret des animations

Le contrôle discret des animations s'effectue de façon très simple en un seul geste ou une seule commande. Il permet de démarrer une animation dans un sens ou dans l'autre (avancer ou reculer) à vitesse constante sans toutefois pouvoir interagir avec elle une fois l'animation débutée. Selon le domaine d'application ou le type d'animation, le déclenchement discret des animations peut se faire de façon séquentielle, soit une après l'autre, ou encore de façon simultanée en démarrant plusieurs animations en parallèle. Par exemple, l'interface de la Figure 4.1 permet plusieurs épluchages des couches de tissu du crâne en même temps. Les techniques courantes pour débuter les animations de façon discrète consistent en l'utilisation des flèches directionnelles sur le clavier, un clic sur un bouton ou encore un petit geste rapide avec la souris (voir Figure 4.1).

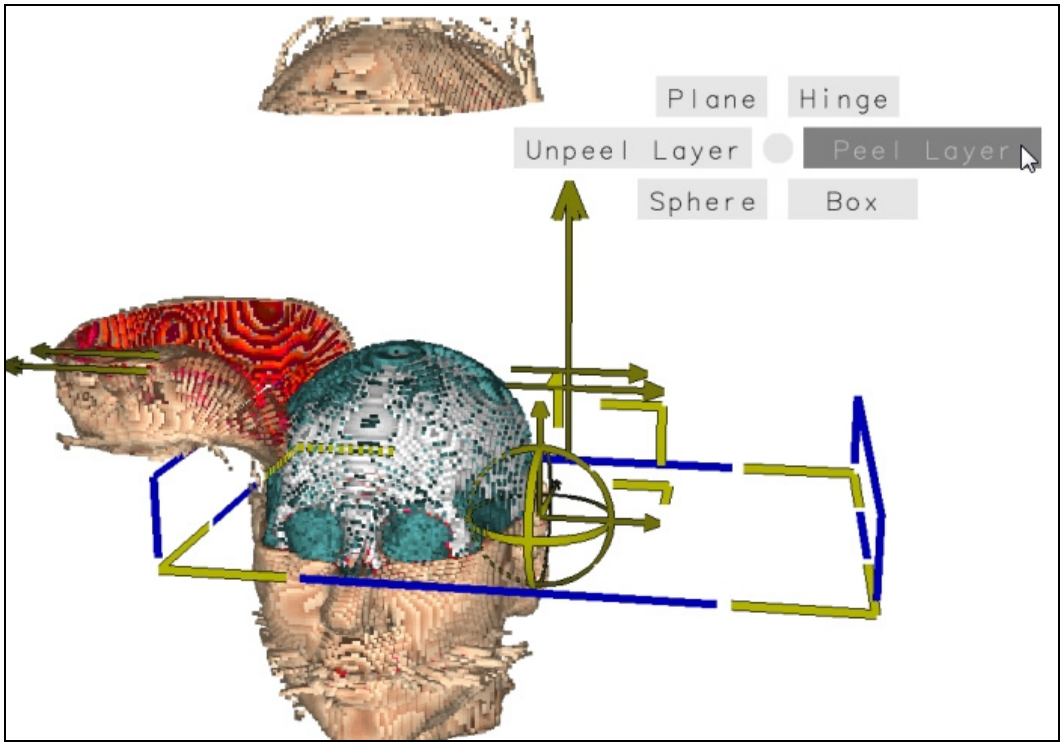

Figure 4.1 *Widget* contextuel pour le contrôle discret d'une transition
Adaptée de Michael McGuffin

4.1.2 Le contrôle continu des animations

Contrairement au contrôle discret, le contrôle continu des animations permet de faire avancer ou reculer celles-ci de façon interactive. L'utilisateur peut donc modifier la vitesse et le sens de l'animation pendant qu'elle s'effectue. Il peut ainsi décider de passer rapidement le début d'une transition pour observer plus attentivement la fin de celle-ci en progressant plus lentement ou encore en rejouant plusieurs fois cette partie vers l'avant ou vers l'arrière. Étant donné que l'utilisateur manipule directement la transition en cours, il ne peut pas en démarrer plusieurs de façon simultanée contrairement au contrôle discret. La technique courante pour interagir avec des animations de façon continue est l'utilisation d'une barre de défilement avec un curseur pour ajuster la position actuelle dans la transition (voir Figure 4.2) ou tout autre type de glisseur entre l'état initial et final de l'animation.

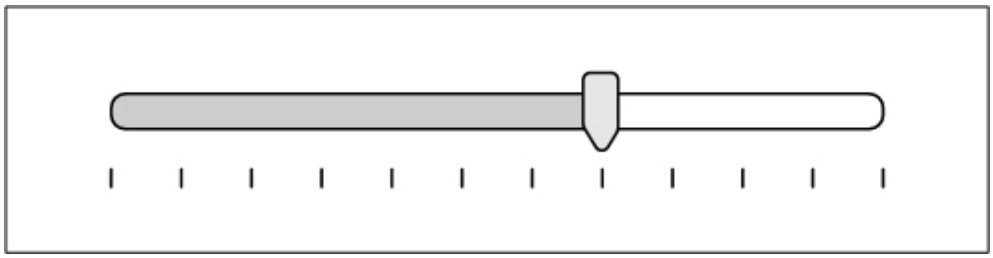

Figure 4.2 Glisseur linéaire horizontal avec curseur de défilement

4.1.3 Repérage de la position dans l'animation

Le repérage de la position dans l'animation permet à l'utilisateur d'avoir de l'information sur la progression de la transition. Cette propriété est particulièrement utile pour la manipulation continue des animations puisque l'utilisateur fait cheminer la transition de façon manuelle. En général, le repérage peut se faire à deux niveaux différents. Premièrement, le repérage indique la position actuelle de l'image à l'écran (cadre en cours) à l'intérieur de la transition par rapport à son début et à sa fin. Ce type de repérage est le plus courant et est utilisé pour connaître l'état d'avancement dans une vidéo par exemple. Le deuxième niveau de repérage,

lorsqu'il s'applique, s'opère sur toutes les transitions qui peuvent se rattacher à un élément ou encore sur l'ensemble des transitions passées ou futures comme un historique. Ainsi, à l'intérieur d'un logiciel qui permet de visualiser l'état d'un système à travers le temps, plusieurs transitions peuvent avoir eu lieu dans le passé. Le repérage doit donc indiquer à l'utilisateur combien de ces transitions sont accessibles et la position de la transition actuelle par rapport aux autres. Cela peut aussi s'appliquer pour ouvrir ou fermer les niveaux d'une arborescence à partir d'un nœud en particulier comme dans le prototype de McGuffin et Balakrishnan (2005). Ainsi, le repérage peut indiquer combien de transitions seront nécessaires pour ouvrir (vers le bas) ou fermer (vers le haut) complètement tous les niveaux (voir Figure 4.3).

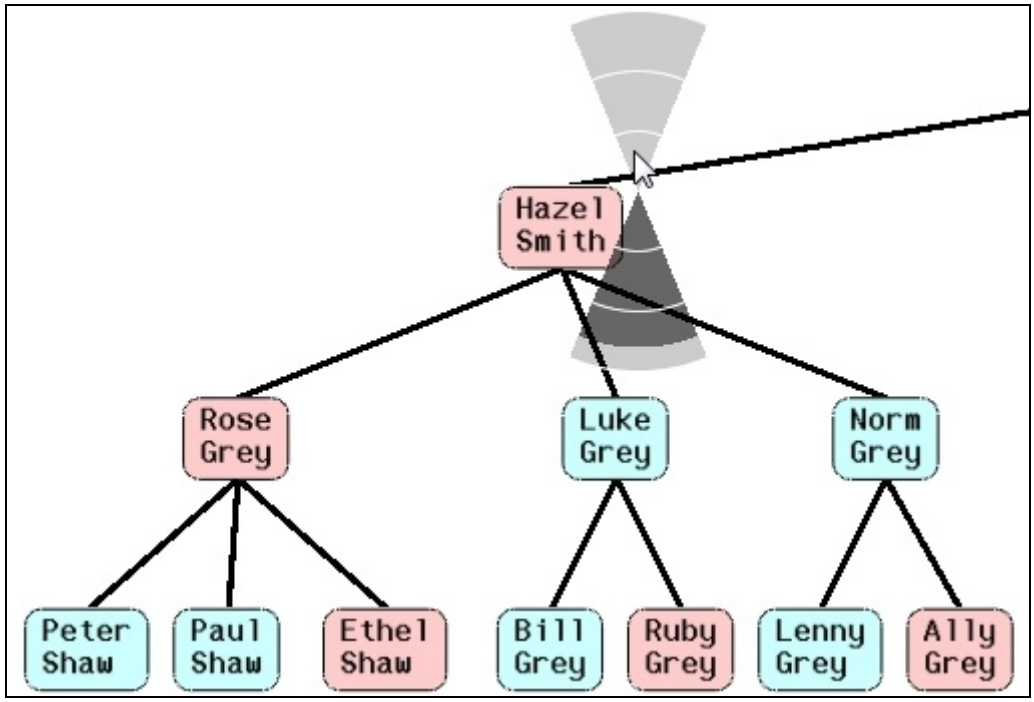

Figure 4.3 *Widget* contextuel pour le contrôle continu montrant un repérage pour la transition en cours et les transitions possibles à partir d'un nœud
Adaptée de Michael McGuffin

4.2 Conception des *widgets* pour le contrôle des animations

Un *widget* contextuel a la particularité de s'adapter au contexte dans lequel il est appelé et s'affiche près du curseur de la souris. Ainsi, l'utilisateur peut directement manipuler l'interface à travers ce composant qui doit être conçu de façon efficace. Pour ce faire, il devrait idéalement intégrer les trois propriétés du contrôle des animations, ce qui n'a pas été fait jusqu'à maintenant dans la littérature antérieure. Toutefois, plusieurs implémentations d'un *widget* intégrant ces propriétés sont possibles et comportent des avantages différents. Avant de proposer des maquettes illustrant de nouveaux composants de contrôle, les caractéristiques de base pour la conception des *widgets* doivent être établies.

4.2.1 Caractéristiques de base pour concevoir des *widgets* novateurs pour le contrôle des transitions

En intégrant le contrôle discret et le contrôle continu, un *widget* doit pouvoir passer d'un mode à un autre sans trop gêner la tâche que l'utilisateur doit accomplir. Comme le contrôle discret est direct, rapide et sans interaction suite au déclenchement, il convient de pouvoir démarrer une animation de cette façon en premier. Ainsi, le contrôle discret d'une animation à l'intérieur d'un *widget* contextuel peut s'effectuer par un geste rapide vers la droite pour avancer ou vers la gauche pour reculer à l'intérieur des différentes transitions. Toutefois, dans le cas où l'utilisateur veut utiliser directement le contrôle continu, trois méthodes sont possibles pour y arriver. Premièrement, un court délai peut être appliqué après l'appel du *widget* pour passer du mode discret au mode continu. Aucun mouvement n'est nécessaire avant l'apparition du glisseur pour faire l'entrée continue, mais ce délai peut entraîner un retard dans la complétion de la tâche surtout si une même action est répétée souvent. Deuxièmement, un mouvement perpendiculaire vers le haut ou le bas se différencie bien de ceux appliqués pour le contrôle discret. Ainsi, l'utilisateur qui veut passer directement au contrôle continu peut le faire sans délai dès qu'il fait l'appel du *widget* et ce dernier peut faire apparaître un glisseur immédiatement. Finalement, le contrôle continu peut s'activer lorsque

le pointeur de la souris s'immobilise dans une zone d'activation. À l'intérieur de cette zone, le *widget* s'adapte au nouveau type de contrôle et l'utilisateur peut donc faire une entrée continue.

Une autre caractéristique à considérer est la forme du *widget* qui peut être linéaire ou radiale. Un glisseur linéaire est traditionnellement utilisé pour le contrôle continu. Il a l'avantage de permettre un accès plus rapide au début ou à la fin de la transition en bougeant rapidement le curseur à l'une ou l'autre des extrémités du glisseur. Toutefois, un espace plus grand est nécessaire pour l'affichage et la manipulation de ce dernier. Par exemple, il pourrait être nécessaire de replacer la souris sur la surface de travail pour continuer le glissement dans le composant d'interface. De son côté, le *widget* radial consiste à faire progresser l'animation en bougeant le curseur en cercle autour de son point central. Donc, pour arriver au début ou la fin des transitions, l'utilisateur devra effectuer plusieurs tours contrairement au glisseur linéaire. Cependant, la forme radiale nécessite un espace plus restreint pour l'affichage et la manipulation du *widget* puisque le curseur se déplace selon une trajectoire circulaire fixe. Aussi, elle procure un niveau de contrôle qui s'adapte selon l'éloignement du curseur de la souris par rapport au centre du *widget*. En tournant de façon rapprochée du centre, la progression est plus rapide, mais moins précise contrairement à une progression plus lente avec un contrôle plus fin en s'éloignant du centre de plus en plus.

4.2.2 Maquettes de *widgets* novateurs pour le contrôle des animations

La combinaison des trois modes de passage du contrôle discret au contrôle continu et les deux formes de *widgets* apportent six choix possibles pour la conception d'un nouveau composant d'interface. Pour mettre en application quelques-uns de ses arrangements, cette section présente des maquettes proposant des techniques d'interaction pour le contrôle des animations. Dans tous les exemples suivants, les illustrations supposent que trois transitions sont possibles, soit une passée et deux futures. De plus, les éléments dessinés en bleu servent

de repères pour aider à la compréhension du fonctionnement du composant et ne seraient normalement pas visibles par l'utilisateur dans un logiciel.

Le premier *widget* présente un délai d'apparition et une forme linéaire. Lorsque l'utilisateur fait l'appel de celui-ci en maintenant enfoncé le bouton droit de la souris par exemple, il peut démarrer l'animation dans un mode expert (sans affichage du menu) avec un mouvement rapide vers la droite pour avancer ou vers la gauche pour reculer (voir Figure 4.4, étape 1). Si toutefois il veut utiliser le contrôle continu, il appelle le composant d'interface de la même façon, mais un court délai est nécessaire avant l'apparition d'un glisseur (2 et 3). Les trois sections du glisseur montrent les transitions existantes et la première partie ombragée montre que cette transition est déjà passée. Une fois le *widget* en mode continu, l'utilisateur peut glisser le curseur dans le sens désiré pour contrôler l'animation et le progrès s'affiche à l'intérieur de la transition en cours (4). Si l'utilisateur poursuit son mouvement, la transition en cours va se terminer et la prochaine, représentée par la troisième case, débutera (5).

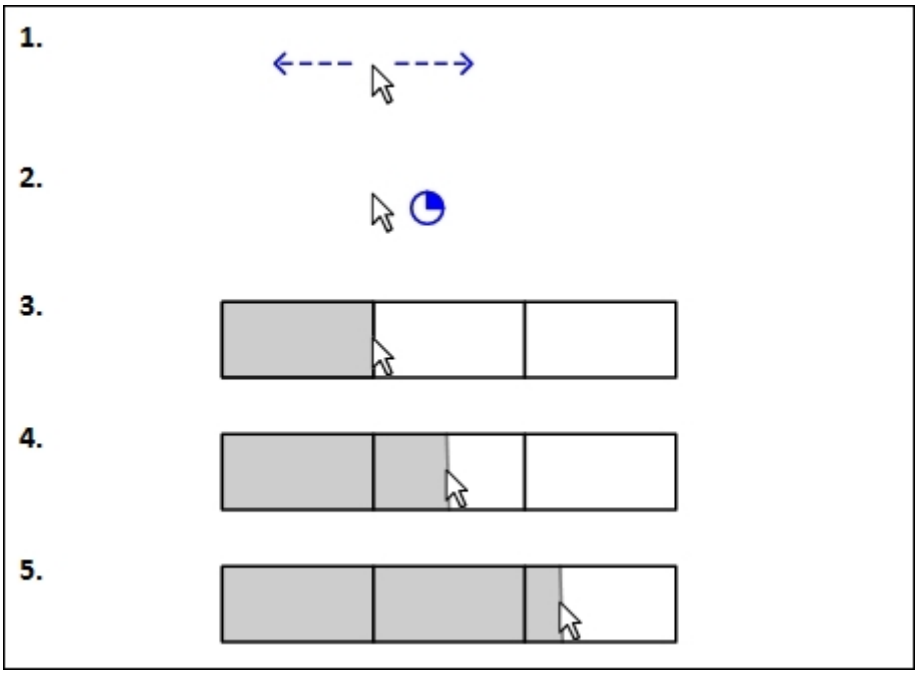

Figure 4.4 Technique d'interaction d'un *widget* linéaire combinant un contrôle discret (1) et un contrôle continu (3 à 5) après un délai (2)

Le prochain *widget* combine un mouvement perpendiculaire pour le changement de contrôle et une forme linéaire. Pour le contrôle discret, il peut s'effectuer comme le composant précédent ou encore dans un mode novice où un menu s'affiche pour démarrer l'animation (voir Figure 4.5). Lors de l'appel, le curseur se trouve au centre du menu (1), puis l'animation est débutée vers l'avant avec un geste vers la droite (2). Pour déclencher la commande, il n'est pas nécessaire de s'arrêter précisément sur l'item du menu, car il est possible de le dépasser dans le but de permettre un geste rapide. Pour le passage au mode continu, à partir de la position centrale du menu, l'utilisateur déplace le curseur de la souris vers le haut de façon perpendiculaire aux flèches (3). En cours de route, le glisseur linéaire va apparaître et le menu pour le contrôle discret va disparaître en même temps (4). L'utilisateur se retrouve maintenant avec le même glisseur que le *widget* proposé précédemment (5 et 6).

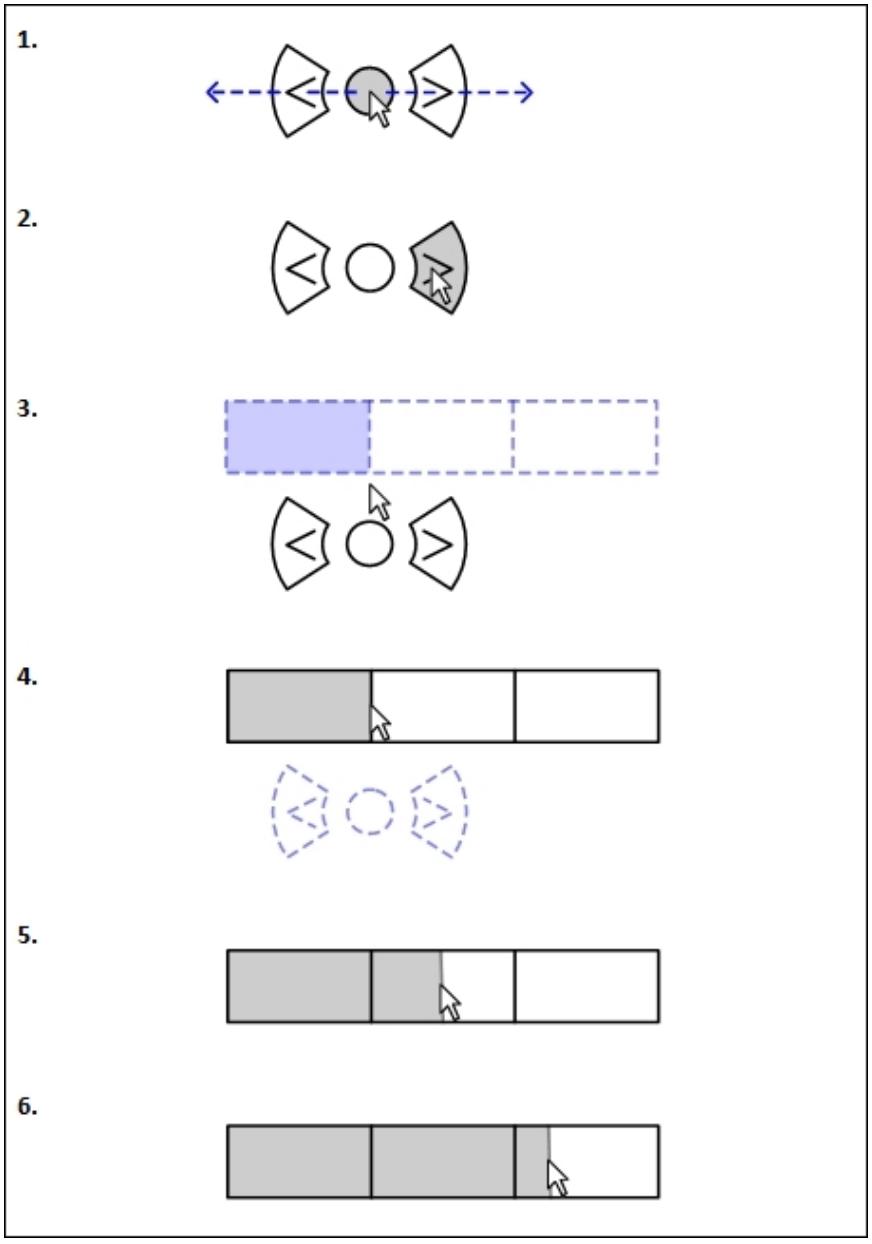

Figure 4.5 Technique d'interaction d'un *widget* linéaire combinant un contrôle discret (1 et 2) et un contrôle continu (5 et 6) après un mouvement perpendiculaire (3 et 4)

Dans le but d'enrichir la fonctionnalité du *widget* linéaire et ainsi donner plus d'information à l'utilisateur, le glisseur tel que présenté pourrait intégrer des vignettes de prévisualisations (voir Figure 4.6). Aux extrémités de chaque case représentant une transition, une petite image

semi-transparente, pour ne pas cacher complètement l'interface en dessous, montre l'état du système à chaque début ou fin de transition. Bien qu'il ne puisse pas présenter les données de façon détaillée en raison de la taille de l'image, cet ajout permet toutefois de différencier chacune des cases en donnant ainsi à l'utilisateur un meilleur aperçu de l'historique complet des transitions.

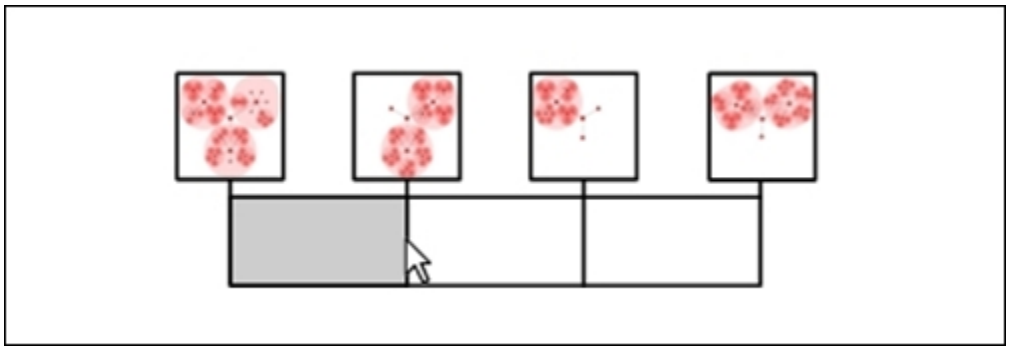

Figure 4.6 Intégration de vignettes de prévisualisation des transitions au glisseur d'un *widget* linéaire

Le dernier *widget* linéaire à être présenté utilise une zone d'activation pour basculer vers le contrôle continu (voir Figure 4.7). Il présente une zone centrale en forme de cercle et deux zones latérales qui représentent les transitions. Celle de gauche représente l'animation passée avec une seule case de forme rectangulaire et celle de droite en possède deux. Tout d'abord, le contrôle discret s'effectue de façon semblable aux techniques précédentes sauf que l'utilisateur doit absolument dépasser les limites du *widget* affiché à l'écran pour démarrer l'animation (1). Pour passer au contrôle continu, l'utilisateur doit immobiliser le curseur de la souris dans une zone d'activation ayant une forme triangulaire et qui est située de chaque côté de la zone centrale (2). Après un court moment, le glisseur va s'ouvrir permettant ainsi à l'utilisateur de contrôler l'animation (3 et 4). Même si sa conception est légèrement différente, le glisseur opère de la même façon que les précédents et permet de passer à une transition ultérieure (5).

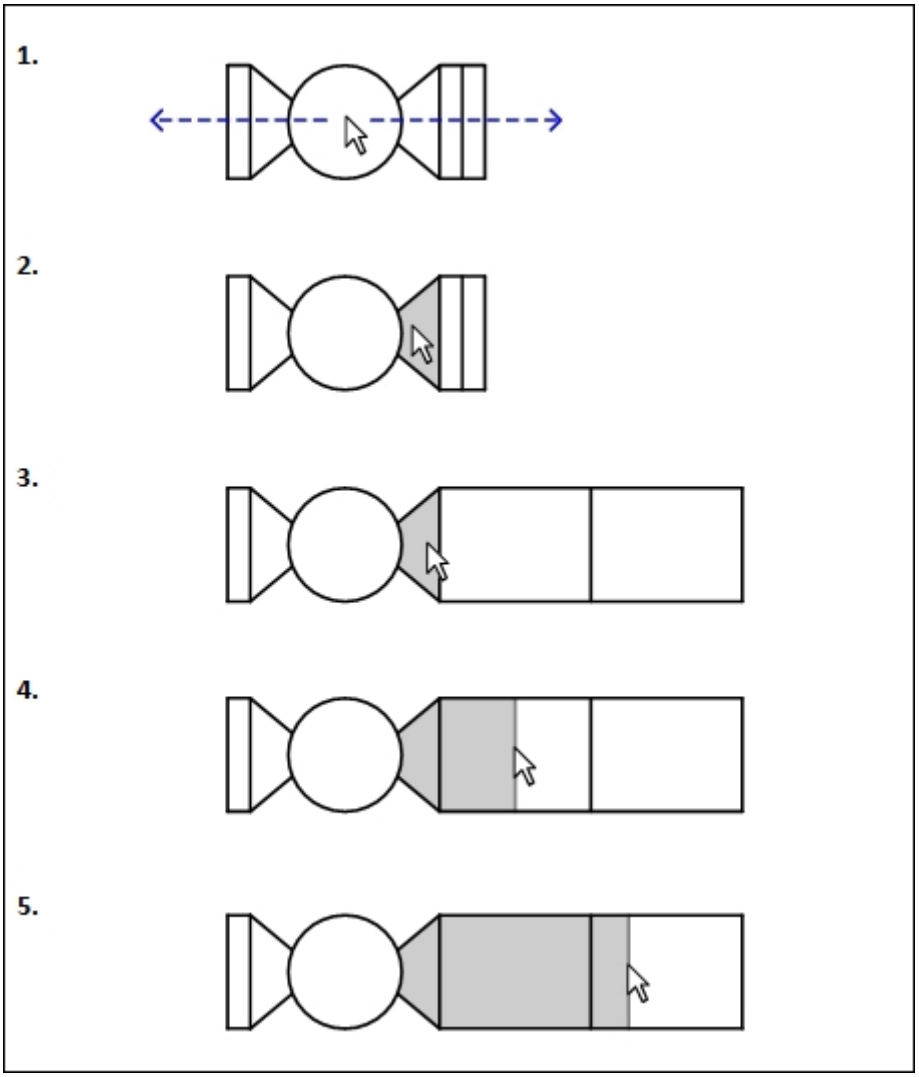

Figure 4.7 Technique d'interaction d'un widget linéaire combinant un contrôle discret (1) et un contrôle continu (3 à 5) à l'aide d'une zone d'activation (2)

La conception du dernier *widget* donne aussi la possibilité d'intégrer des items de menu dans l'espace qui n'est pas utilisé autour de la zone centrale (voir Figure 4.8). En s'inspirant du « Control Menu » (Pook *et al.*, 2000), six options peuvent être ajoutées pour effectuer des commandes ou des modifications sur un élément sélectionné comme une ouverture, une fermeture ou un tri sur les nœuds dans le contexte des arborescences. En combinant la sélection de commandes et le contrôle des animations, cela évite d'avoir deux méthodes

différentes pour appeler soit le *widget* de contrôle ou encore le menu avec les options. De plus, ces options pourraient avoir un lien et opérer dans le contexte de la transition en cours. Par exemple, l'utilisateur pourrait sélectionner l'option pour mettre en surbrillance les fermetures de nœuds avant de débuter l'animation de façon discrète ou continue dans une même interaction. Une fois que l'animation est débutée, les options du menu ne sont plus accessibles et peuvent disparaître pour ne pas cacher l'interface en dessous du *widget*.

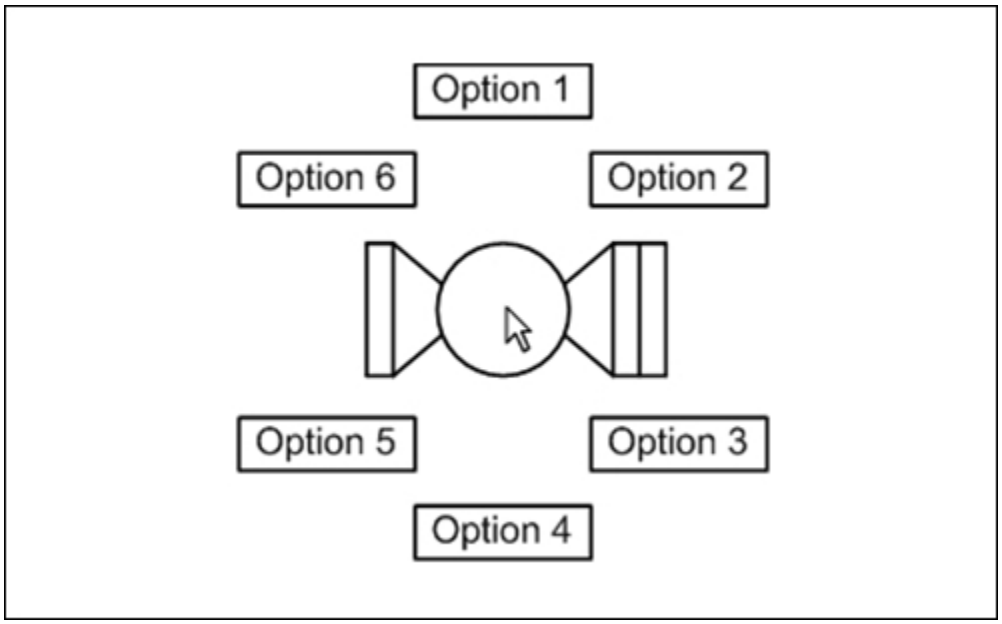

Figure 4.8 Intégration d'items de menu avec un *widget* linéaire comportant une zone d'activation pour le contrôle continu

Finalement, le dernier *widget* unique a être présenté est circulaire, et comporte lui aussi une zone d'activation pour le contrôle continu (voir Figure 4.9). Il est séparé en trois zones fidèlement au nombre de transitions existantes et la zone ombragée représente celle qui est déjà passée. Ainsi, la première transition débutera à midi, en se référant à la position des aiguilles sur une horloge, et les autres seront réparties dans le sens horaire. Semblable au *widget* linéaire de même conception, l'utilisateur doit dépasser l'affichage du contrôle pour démarrer une animation de façon discrète. Pour utiliser le mode continu, le curseur de la souris doit s'arrêter brièvement dans la zone d'activation représentée par les deux cercles

concentriques autour de la zone centrale (2). À ce moment, la zone de glissement va s'agrandir et la position du curseur va être enregistrée pour servir de référence pour le début de l'animation (3). Ainsi, l'utilisateur peut activer le contrôle continu à partir de n'importe quelle position dans la zone d'activation avant de faire progresser l'animation en suivant la trajectoire circulaire du glisseur (4). Pour passer d'une transition à une autre, le curseur de la souris doit effectuer un tour complet à partir de la position initiale enregistrée. C'est pourquoi l'indicateur de progression de l'animation ne suit pas directement la position du curseur à l'intérieur du cercle. De plus, il n'est pas nécessaire de garder le curseur à l'intérieur du glisseur pour poursuivre l'animation (5). En effet, plus le curseur s'éloigne du centre du *widget*, plus le contrôle de l'animation est lent donnant ainsi une meilleure précision de la manipulation à l'intérieur du glisseur. À l'inverse, si l'utilisateur veut passer rapidement d'une transition à l'autre, il n'a qu'à rapprocher le curseur du centre puisque la distance à parcourir sera moindre.

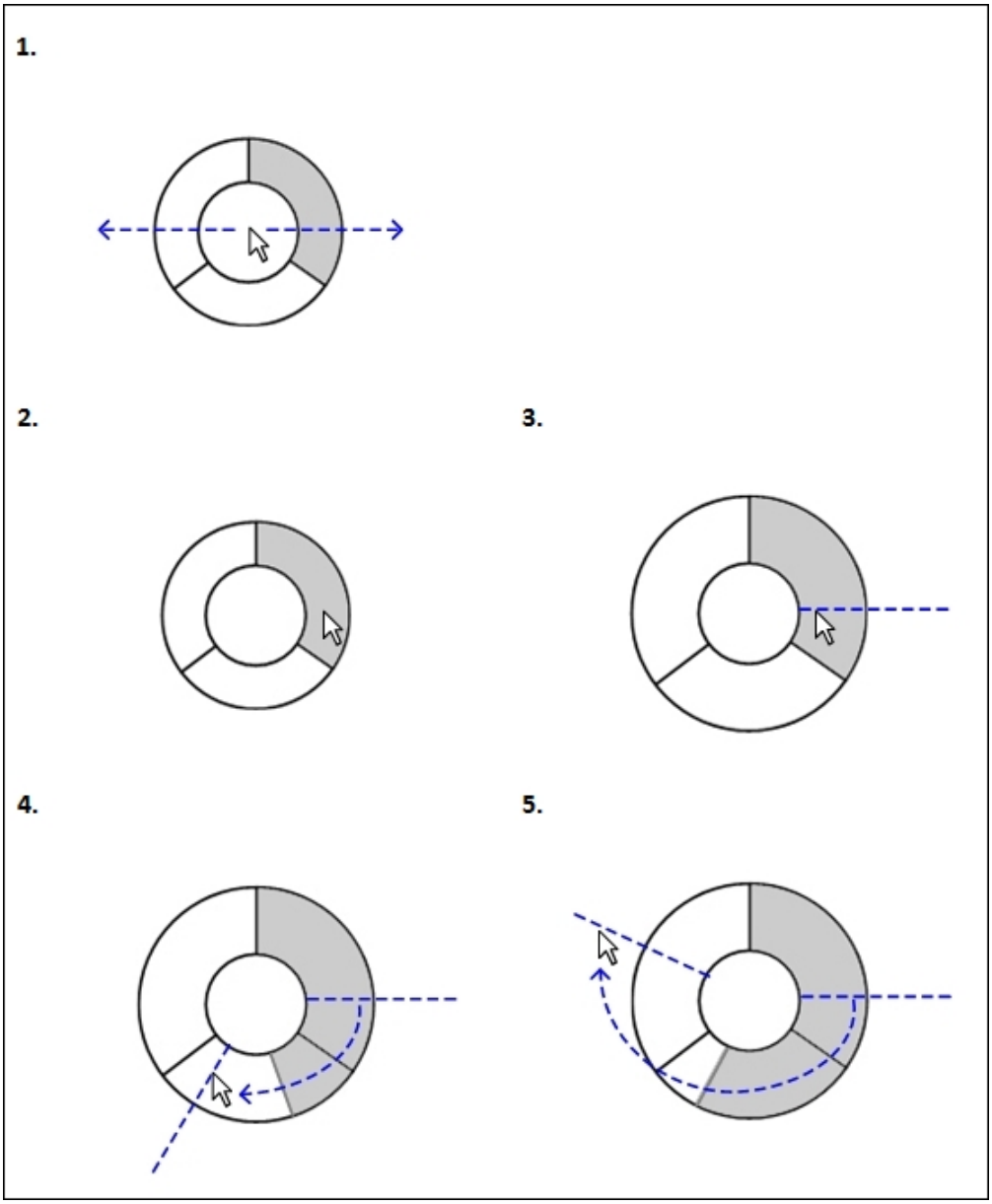

Figure 4.9 Technique d'interaction d'un *widget* radial combinant un contrôle discret (1) et un contrôle continu (3 à 5) à l'aide d'une zone d'activation (2)

Ces quatre maquettes ont démontré les caractéristiques distinctives des trois modes de passage des types de contrôle des animations et des deux formes que peuvent prendre les *widgets*. Cependant, deux arrangements n'ont pas encore été abordés. Toutefois, ils regroupent des éléments déjà présents dans les propositions précédentes. En effet, un *widget* de forme radiale peut aussi être utilisé après un délai d'apparition ou un mouvement

perpendiculaire de la même façon qu'un *widget* linéaire. À partir d'un contrôle discret en mode novice ou expert, un court délai ou un mouvement perpendiculaire fera afficher un glisseur radial plutôt que linéaire. Dans le but de faire l'inventaire complet des *widgets* et pour mieux les distinguer, une taxonomie (voir Figure 4.10) classifie les techniques discutées selon la combinaison entre chaque mode de passage d'un contrôle à l'autre (lignes) et la forme d'affichage du glisseur (colonnes).

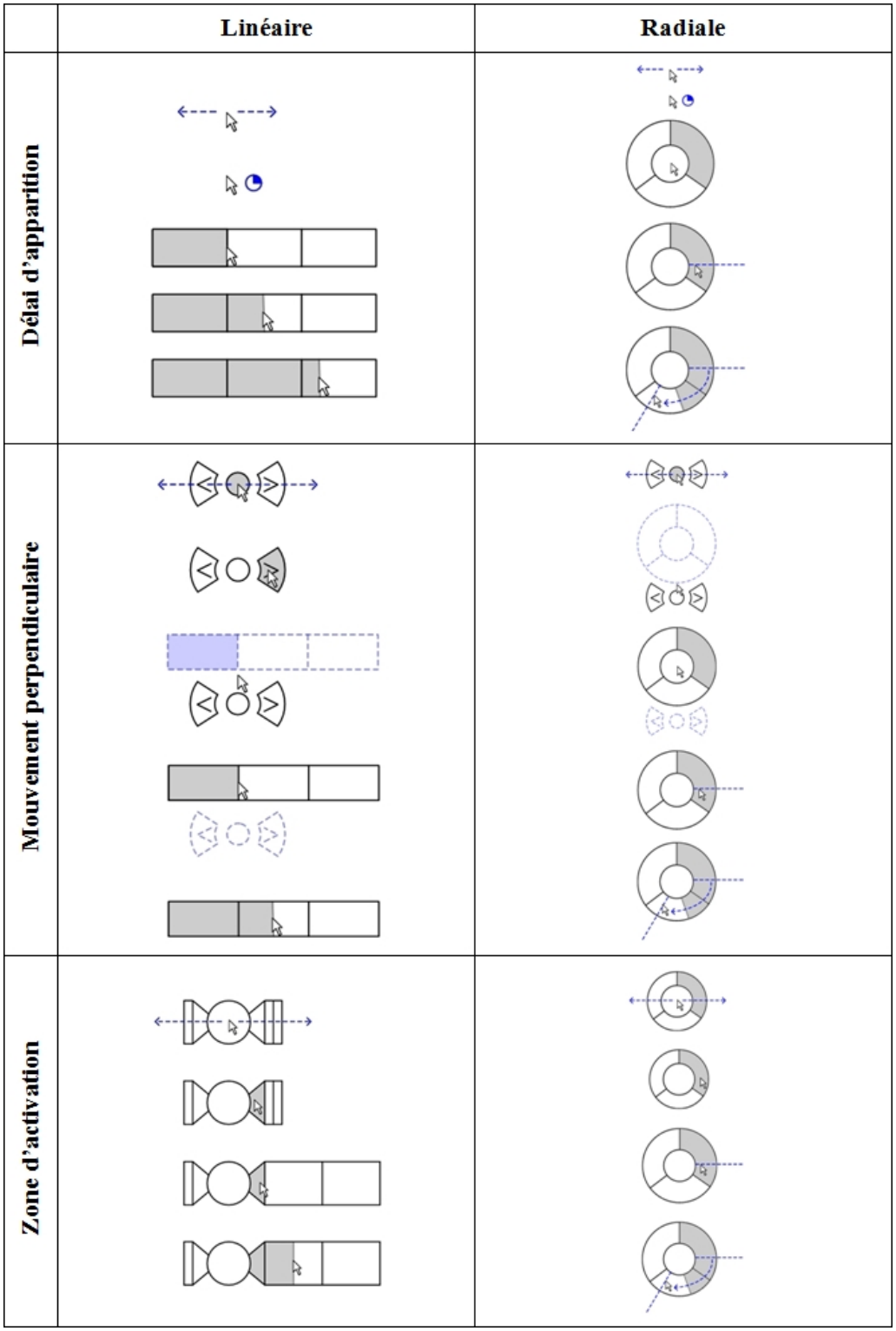

Figure 4.10 Taxonomie classifiant les *widgets* selon le mode de passage entre le contrôle discret et continu (lignes) et selon leur forme (colonnes)

4.3 Résumé

Pour être efficaces, les *widgets* pour le contrôle des animations doivent inclure une méthode de contrôle discret à vitesse constante et une autre de façon continue où l'utilisateur manipule lui-même la progression de la transition. De plus, ces composants d'interface doivent fournir un repérage qui permet de connaître la position de l'image en cours en fonction de la présente transition, mais aussi à l'intérieur de toutes les transitions existantes. Pour intégrer tous ces concepts, trois modes de passage du contrôle discret à continu (délai d'apparition, mouvement perpendiculaire et zone d'activation) et deux formes de glisseur (linéaire et radiale) peuvent être combinés pour créer de nouveaux *widgets*. Bien que les maquettes présentées permettent de bien comparer ces alternatives, il demeure une interrogation sur la réelle efficacité de chacune des techniques et leurs avantages entre elles. Il pourrait alors être pertinent, dans le cadre d'une nouvelle étude, d'expérimenter ces *widgets* avec les types d'animation proposés dans ce mémoire.

CONCLUSION

Le but de cette étude est, dans un premier temps, de proposer de nouveaux types de transitions fluides qui permettent d'améliorer le suivi des éléments et la compréhension de l'utilisateur pour la visualisation de données. De plus, elle propose de nouveaux types de *widgets* plus complets et flexibles pour le contrôle des animations en intégrant les propriétés essentielles des techniques existantes.

Pour ce faire, une revue de la littérature a permis de connaître les différentes techniques pour faire la conception des animations et qui sont utilisées pour bâtir des transitions fluides. Parmi celles-ci, les transitions par étapes associent plusieurs animations subséquentes pour découper les actions et les changements de position pour bâtir une seule transition fluide. Bien que cette méthode soit théoriquement plus efficace, une étude a démontré peu d'avantages par rapport à une transition linéaire simple en une seule étape. Une problématique a alors fait surface : comment tirer profit de la structure des données pour mieux concevoir une transition animée. Concernant le contrôle interactif des transitions, la revue a fait ressortir peu d'études proposant des techniques précisément adaptées aux animations. Bien qu'elles soient efficaces à différents niveaux, une meilleure intégration devait être faite pour regrouper les avantages de chacune à l'intérieur d'un seul et même composant d'interface.

Par la suite, pour analyser et mettre en pratique les différents types d'animations, un prototype de visualisation de données sous la forme d'une arborescence a été développé. Ce contexte particulier apporte une complexité accrue à cause du nombre d'éléments et des liens qui les unissent. Premièrement, les animations linéaire et par étapes ont été adaptées à ce nouveau contexte qui prend en compte trois types de changements sur les nœuds (fermetures, ouvertures et permutations) pouvant s'opérer sur les différents niveaux de l'arborescence. Ainsi, l'animation linéaire anime tous les types de changements en même temps sur tous les niveaux de l'arbre. De son côté, l'animation par étapes anime d'abord les fermetures suivies

des permutations et des ouvertures sur tous les niveaux simultanément. Ces deux derniers types de transitions prennent seulement en compte les types de changements et n'exploitent pas l'agencement des données. C'est pourquoi deux nouvelles transitions ont été conçues et réalisées : les transitions hiérarchique et hybride. L'animation hiérarchique anime tous les types de changements niveau par niveau à partir du moins profond jusqu'au dernier. De plus, l'animation hybride anime d'abord les fermetures sur tous les niveaux avant de procéder aux permutations de façon hiérarchique niveau par niveau en terminant avec toutes les ouvertures en même temps. Ces nouvelles transitions sont conçues pour mieux diriger l'attention de l'utilisateur et de mieux découper les types de changements dans le but de faciliter le suivi des nœuds et l'identification des types de changements. Trois autres transitions animées novatrices ont aussi été proposées, mais ont été éliminées dues au nombre élevé d'étapes qu'elles impliquent. Donc, plusieurs hypothèses ont été formulées et tendent généralement à donner un avantage à l'animation par étapes face à linéaire en plus de favoriser les animations hiérarchique et hybride par rapport à ces deux premières.

Dans le but de comparer et d'évaluer les types d'animations entre eux, une expérimentation avec 12 participants a été réalisée. Ainsi, de nombreux essais ont été effectués à l'aide de deux tâches à l'intérieur d'un logiciel de test. La tâche principale consistait à suivre un nœud préalablement désigné et à indiquer sa position à la fin de l'animation. De plus, la tâche secondaire qui s'exécutait en parallèle portait sur l'identification des nœuds qui ont subi une ouverture, une fermeture ou une permutation. Ces deux observations conjointes ont donc permis d'évaluer la prise de conscience du contexte par l'utilisateur tout en suivant un nœud en particulier. L'analyse des résultats a démontré un avantage significatif pour l'animation linéaire face à l'animation par étapes pour le suivi des nœuds et il en est de même avec l'animation hiérarchique et hybride face aux deux premières. Pour les ouvertures de nœuds, les animations par étapes et hybride ont montré un taux de réussite supérieur tandis que l'animation linéaire a mieux performé que les autres pour les fermetures. Pour les permutations, les animations hiérarchique et hybride ont montré un avantage significatif face aux animations linéaire et par étapes. Bien que certaines hypothèses aient pu être confirmées,

d'autres résultats n'ont pas été assez significatifs pour pouvoir en tirer une conclusion valable. De plus, des effets contraires ont été observés surtout au niveau de la meilleure performance de l'animation linéaire face à l'animation par étapes pour le suivi des nœuds, les permutations et les fermetures ainsi que face aux animations hiérarchique et hybride pour les fermetures uniquement. Par la suite, grâce à l'interprétation des résultats, les commentaires des participants et la revue de la littérature, huit directives de conception ont été rédigées dans le but de guider et d'améliorer la conception des transitions fluides dans le domaine de la visualisation de données.

Un moyen supplémentaire pour faciliter la compréhension des transitions animées est de permettre à l'utilisateur de les contrôler à sa guise. L'analyse des techniques existantes pour le contrôle interactif des animations a fait ressortir trois propriétés essentielles que tous les *widgets* contextuels employés pour manipuler les transitions fluides devraient idéalement posséder. En effet, ils doivent intégrer une méthode de contrôle discret (simple et rapide) pour débuter les animations à vitesse constante en plus de combiner un contrôle continu avec lequel l'utilisateur peut modifier la vitesse et le sens de l'animation comme il le souhaite. Également, ils doivent offrir une méthode de repérage de la position de l'image actuelle à l'intérieur de la transition en cours en plus de toutes celles qui sont accessibles dans le passé ou le futur. Pour mettre en illustration ces principes, des maquettes de nouveaux composants d'interface ont été présentées en rapport avec une taxonomie les classifiant selon le mode de passage du contrôle discret à continu (délai d'apparition, mouvement perpendiculaire et zone d'activation) et la forme du glisseur (linéaire et radiale).

Cette étude apporte plusieurs contributions tout d'abord en démontrant un net avantage pour les animations hiérarchique et hybride face aux types d'animations existants pour les permutations. De plus, elle confirme que l'animation par étapes traditionnelle n'est pas suffisante pour se démarquer face à une simple animation linéaire dans le contexte étudié. Elle propose aussi huit directives de conception pour les transitions fluides dans le but de guider les développeurs et les chercheurs qui désirent poursuivre ou améliorer les travaux

dans ce domaine. Elle propose aussi des maquettes de nouveaux composants d'interface, organisés selon une taxonomie, qui pourraient faciliter le contrôle des animations en fusionnant deux méthodes de contrôle et une de repérage comparativement aux techniques existantes.

Toutes ces propositions permettent d'améliorer la facilité de compréhension des transitions fluides, et par le fait même des changements sur les données, par l'utilisateur tout en les rendant extensibles. Les résultats obtenus par les animations hiérarchique et hybride démontrent que ces transitions permettraient de visualiser un plus grand nombre de données à la fois comparativement aux approches traditionnelles. Cela est d'autant plus vrai lorsque ces transitions peuvent être contrôlées de façon interactive à l'aide de *widgets* plus complets et adaptés au contexte. De plus, en basant les étapes des transitions sur la nature des changements et la structure des données, elles sont donc indépendantes du type d'implémentation et peuvent aussi s'appliquer dans plusieurs domaines différents. En effet, elles pourraient être utilisées pour visualiser les changements en temps réel sur un réseau informatique, l'état des machines ou le déplacement des produits dans une chaîne de montage et l'effet de la propagation d'un phénomène météorologique ou épidémiologique sur un territoire ou une population.

Afin de poursuivre la recherche, il conviendrait de réaliser une nouvelle étude qui pourrait présenter de nouvelles transitions fluides intégrant les directives de conception énoncées dans ce mémoire afin de les valider et de confirmer leurs propositions. En effet, les transitions montrées dans cette étude ont volontairement omis des principes comme la réduction de l'occlusion et la distorsion de la vitesse d'animation dans le but d'évaluer seulement l'ordonnancement des étapes des transitions selon le type de changement et les niveaux de l'arborescence. En se consacrant particulièrement sur des nouveaux types d'animations dits hiérarchiques, l'utilisation de vitesses d'animation variables ou de trajectoires de mouvement intelligentes pourrait démontrer une performance supérieure. De plus, les maquettes de *widgets* proposées à la fin de ce mémoire sont purement théoriques et nécessitent d'être

évaluées dans des conditions réelles. Ainsi, il sera possible de déterminer quelle implémentation est la plus efficace et selon quelles conditions, dans le but d'énoncer des directives de conception spécifiques aux *widgets* pour le contrôle interactif des transitions fluides.

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

Baecker, R., et I. Small. 1990. « Animation at the interface ». In *The Art of Human-Computer Interface Design*, sous la dir. de Laurel, Brenda. p. 251-267. New York : Addison-Wesley.

Bailly, Gilles, Eric Lecolinet et Laurence Nigay. 2007. « Quinze ans de recherche sur les menus : critères et propriétés des techniques de menus ». In *Conférence Internationale de l'Association Francophone d'Interaction Homme-Machine*. p. 119-126. Paris, France : ACM.

Bartram, L. 1997. « Can motion increase user interface bandwidth in complex systems? ». In *IEEE International Conference on Systems, Man, and Cybernetics*. Vol. 2, p. 1686-1692.

Bederson, B. B., et A. Boltman. 1999. « Does animation help users build mental maps of spatial information? ». In *Proceedings of IEEE Symposium on Information Visualization (InfoVis)*. p. 28-35.

Boardman, Richard. 2000. « Bubble trees the visualization of hierarchical information structures ». In *Extended abstracts of ACM Conference on Human Factors in Computing Systems (CHI)*. p. 315-316. La Haye, Pays-Bas : ACM.

Card, Stuart K. . 2008. « Information Visualization ». In *The Human-Computer Interaction Handbook : Fundamentals, Evolving Technologies and Emerging Applications*, sous la dir. de Sears, Andrew et Julie A. Jacko. p. 509-543. Lawrence Erlbaum Associates.

Cavanagh, P., et G.A. Alvarez. 2005. « Tracking multiple targets with multifocal attention ». *Trends in Cognitive Sciences,* vol. 9, n^{o} 7, p. 349-354.

Chang, Bay-Wei, et David Ungar. 1993. « Animation : from cartoons to the user interface ». In *Proceedings of ACM Symposium on User Interface Software and Technology (UIST)*. p. 45-55. Atlanta, Géorgie, États-Unis : ACM.

Dragicevic, Pierre, Anastasia Bezerianos, Waqas Javed, Niklas Elmqvist et Jean-Daniel Fekete. 2011. « Temporal Distortion for Animated Transitions ». In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI)*. Vancouver, Colombie-Britannique, Canada : ACM.

Elmqvist, N., P. Dragicevic et J. D. Fekete. 2008. « Rolling the Dice : Multidimensional Visual Exploration using Scatterplot Matrix Navigation ». *IEEE Transactions on Visualization and Computer Graphics (TVCG),* vol. 14, n^{o} 6, p. 1539-1148.

Gonzalez, Cleotilde. 1996. « Does animation in user interfaces improve decision making? ». In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI)*. p. 27-34. Vancouver, Colombie-Britannique, Canada : ACM.

Guimbretière, François, et Terry Winograd. 2000. « FlowMenu : combining command, text, and data entry ». In *Proceedings of ACM Symposium on User Interface Software and Technology (UIST)*. p. 213-216. San Diego, Californie, États-Unis : ACM.

Heer, J., et G. G. Robertson. 2007. « Animated Transitions in Statistical Data Graphics ». *IEEE Transactions on Visualization and Computer Graphics (TVCG),* vol. 13, n° 6, p. 1240-1247.

Jürgensmann, S., et H.J. Schulz. 2010. « Poster : a visual survey of tree visualization ». In *Proceedings of IEEE Information Visualization*. Salt Lake City, Utah, États-Unis : IEEE Press.

Klein, Christian, et Benjamin B. Bederson. 2005. « Benefits of animated scrolling ». In *Extended abstracts of ACM Conference on Human Factors in Computing Systems (CHI)*. p. 1965-1968. Portland, Oregon, États-Unis : ACM.

Kurtenbach, Gordon, et William Buxton. 1993. « Limits of expert performance using hierarchic marking menus ». In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI)*. p. 482-487. Amsterdam, Pays-Bas : ACM.

Lee, B., C.S. Parr, C. Plaisant, B.B. Bederson, V.D. Veksler, W.D. Gray et C. Kotfila. 2006. « Treeplus : Interactive exploration of networks with enhanced tree layouts ». *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, p. 1414-1426.

McGuffin, M., N. Burtnyk et G. Kurtenbach. 2002. « FaST Sliders : Integrating marking menus and the adjustment of continuous values ». *Proceedings of Graphics Interface (GI)*, p. 35-41.

McGuffin, M. J., et R. Balakrishnan. 2005. « Interactive visualization of genealogical graphs ». In *Proceedings of IEEE Symposium on Information Visualization (InfoVis)*. p. 16-23.

McGuffin, M. J., G. Davison et Balakrishnan Ravin. 2004. « Expand-Ahead : A Space-Filling Strategy for Browsing Trees ». In *Proceedings of IEEE Symposium on Information Visualization (InfoVis)*. p. 119-126.

McGuffin, Michael J., Liviu Tancau et Ravin Balakrishnan. 2003. « Using Deformations for Browsing Volumetric Data ». In *Proceedings of IEEE Symposium on Information Visualization (InfoVis)*. p. 53. IEEE Computer Society.

Melançon, G., et I. Herman. 1998. *Circular drawings of rooted trees*. Coll. « Information Systems », R 9817. CWI, 1-13 p.

Oksama, L., et J. Hyönä. 2004. « Is multiple object tracking carried out automatically by an early vision mechanism independent of higher-order cognition? An individual difference approach ». *Visual cognition,* vol. 11, n° 5, p. 631-671.

Plaisant, C., J. Grosjean et B. B. Bederson. 2002. « SpaceTree : supporting exploration in large node link tree, design evolution and empirical evaluation ». In *Proceedings of IEEE Symposium on Information Visualization (InfoVis)*. p. 57-64.

Pook, Stuart, Eric Lecolinet, Guy Vaysseix et Emmanuel Barillot. 2000. « Control menus : excecution and control in a single interactor ». In *Extended abstracts of ACM Conference on Human Factors in Computing Systems (CHI)*. p. 263-264. La Haye, Pays-Bas : ACM.

Pylyshyn, Z.W., et R.W. Storm. 1988. « Tracking multiple independent targets : Evidence for a parallel tracking mechanism ». *Spatial vision,* vol. 3, n° 3, p. 179-197.

Robertson, George, Kim Cameron, Mary Czerwinski et Daniel Robbins. 2002. « Polyarchy visualization : visualizing multiple intersecting hierarchies ». In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI)*. p. 423-430. Minneapolis, Minnesota, États-Unis : ACM.

Robertson, G., R. Fernandez, D. Fisher, B. Lee et J. Stasko. 2008. « Effectiveness of animation in trend visualization ». *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, p. 1325-1332.

Schlienger, Céline, Pierre Dragicevic, Claire Ollagnon et Stéphane Chatty. 2006. « Les transitions visuelles différenciées : principes et applications ». In *Conférence de l'Association Francophone d'Interaction Homme-Machine*. p. 59-66. Montréal, Québec, Canada : ACM.

Sellen, Abigail J., Gordon P. Kurtenbach et William A. S. Buxton. 1992. « Prevention of mode errors through sensory feedback ». *Human-Computer Interaction,* vol. 7, n° 2, p. 141-164.

Shanmugasundaram, Maruthappan, Pourang Irani et Carl Gutwin. 2007. « Can smooth view transitions facilitate perceptual constancy in node-link diagrams? ». In *Proceedings of Graphics Interface (GI)*. p. 71-78. Montréal, Québec, Canada : ACM.

Stanford Visualization Group. 2007. *Animated Transitions in Statistical Data Graphics*. En ligne. <http://vis.stanford.edu/papers/animated-transitions>. Consulté le 13 juillet 2011.

Teoh, Soon Tee, et Ma Kwan-Liu. 2002. « RINGS : A Technique for Visualizing Large Hierarchies ». In *Graph Drawing*, sous la dir. de Goodrich, Michael et Stephen Kobourov. Vol. 2528, p. 51-73. Coll. « Lecture Notes in Computer Science ». Springer Berlin / Heidelberg.

Thomas, B.H., et P. Calder. 2001. « Applying cartoon animation techniques to graphical user interfaces ». *ACM Transactions on Computer-Human Interaction (TOCHI),* vol. 8, n^{o} 3, p. 198-222.

Tversky, B., J.B. Morrison et M. Betrancourt. 2002. « Animation : can it facilitate? ». *International Journal of Human-Computer Studies,* vol. 57, n^{o} 4, p. 247-262.

Yee, Ka-Ping, Danyel Fisher, Rachna Dhamija et Marti Hearst. 2001. « Animated Exploration of Dynamic Graphs with Radial Layout ». In *Proceedings of IEEE Symposium on Information Visualization (InfoVis)*. p. 43. IEEE Computer Society.