Approches directes et planifiées de l'interaction 3D sur terminaux mobiles

Fabrice Decle

N° d'ordre : 3836

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ BORDEAUX 1

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par **Fabrice Decle**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Approches Directes et Planifiées de l'Interaction 3D sur Terminaux Mobiles

---

Soutenue le : 23 Septembre 2009

Après avis des rapporteurs :

Bruno Arnaldi	Professeur
Guillaume Moreau	Maître de Conférences

Devant la commission d'examen composée de :

Bruno Arnaldi	Professeur	Rapporteur
Guillaume Moreau	Maître de Conférences	Rapporteur
Géry Casiez	Maître de Conférences	Examinateur
Christophe Schlick	Professeur	Président
Pascal Guitton	Professeur	Directeur de Thèse
Martin Hachet	Chargé de Recherche .	Co-Directeur de Thèse

– 2009 –

Remerciements

Je tiens tout d'abord à remercier mes directeurs de thèse, Martin Hachet et Pascal Guitton, qui m'ont énormément aidé pendant ces trois années. Je les remercie de m'avoir soutenu dans les moments difficiles, et d'avoir toujours su me guider.

Je remercie également les membres du jury, Bruno Arnaldi et Guillaume Moreau, qui ont accepté d'évaluer mon rapport, ainsi que Géry Casiez et Christophe Schlick pour avoir accepté d'assister à ma soutenance.

Qu'auraient été ces trois années sans la présence de mes chers collègues, qui ont du supporter mes blagues douteuse et mes bourrins des demis ? A ce titre, je remercie chaleureusement Joachim Pouderoux et Mickael Raynaud, mes deux compères, sans qui ces trois années auraient été bien fades ! Merci à vous deux pour tous ces moments inoubliables ! Merci également à Jérôme Baril pour m'avoir si souvent laissé gagner au baby, Pascal Barla, pour ses monologues interminables mais toujours instructifs, Sebastian Knödel pour tous ces conseils et ces bons moments, Émilie Lalagüe pour notre goût commun pour certaines séries, Romain Pacanowsky pour cette course effrénée à la rédaction, Yann Savoye pour le Moonwalk qui n'a jamais eu lieu, David Vanderhaeghe pour son fils si adorable, Mariam Amyra pour sa bonne humeur permanente, Benoît Bossavit pour ses blagues aussi douteuses que les miennes, Romain Vergne pour avoir retrouvé mon portefeuille, Vincent Lebret Soler pour ce monde de merde et pour avoir découvert que la parole nous faisait gagner au baby, Rémi Synave pour avoir été un sujet exemplaire, et à tous les autres membres de l'équipe Iparla.

Je remercie également l'équipe d'SC2X, Jean-Paul, Fabrice et les autres, qui m'ont beaucoup appris en me permettant de connaître un nouvel environnement de travail.

Je tiens également à remercier les étudiants avec qui j'ai pu travailler, qui ont porté Navidget sous Blender et SketchUp : Pierre Faure, Pierre Rouanet, Maher Finianos, Yassine El Ouardani, Anthony Battel, Jonathan Depiets, Benjamin Orsini et David Palanchon.

Ces remerciements ne seraient pas complets sans une pensée pour tous mes amis, qui ont fait de moi ce que je suis. Merci à Nicolas "Space Boy", pour ces innombrables moments inoubliables, ces peines partagées et ces nombreuses preuves d'amitié. Merci coco ! Je salue également tous les membres de PiTiFaYa, Pedro, Ti et Ya pour ces fous-rires à la limite de l'asphyxie. Kikikiki à vous quatre les gars ! Et merci à Ti'Cochon et à toute sa famille pour nous avoir si souvent fait rire !

Je n'oublie bien sur pas mes amis de longue date, Manu, Simon, Guillaume et sa tendre

épouse, qui ont été et sont toujours des amis exceptionnels, ainsi que Laurent “El shaman”, Christophe “Punky” et Julien “P’tit Bouchon” pour ce projet si rondement mené !

Bien évidemment, je remercie mes parents et ma famille, qui ont toujours cru en moi et m’ont toujours soutenu. Merci papa, sans toi je n’aurais jamais pu goûter aux joies de l’informatique ! Merci maman, sans toi je n’aurais jamais pu goûter aux joies de la cuisine ! Merci Steph, sans toi je n’aurais jamais pu goûter aux joies des jeux vidéos, et à tant d’autres choses aussi ! Merci également à ma belle famille, pour cet accueil toujours exceptionnel, et pour tous ces bons moments partagés. Merci Bali pour ces bons moments passés chez nous, merci Ghislaine et Jean Pierre d’être si affectueux, merci à toute la famille Loulier pour ces nombreux fous rires !

Enfin, je tiens à saluer mon grand père, un homme doté d’une grande créativité et d’une imagination débordante, qui aurait certainement pu être un grand “inventeur”.

Pour finir, je veux remercier celle qui a toujours été là pour moi, qui m’a soutenu et motivé sans jamais perdre son magnifique sourire, celle qui a été, est, et restera à mes côtés... Merci ma Nadège, merci pour tout.

D’une façon générale cette thèse est dédicacée à tous les gens que j’ai rencontrés, avec qui mes rapports furent aussi divers qu’enrichissants.

Résumé

Les récentes évolutions technologiques des terminaux mobiles tels que les téléphones portables, assistants personnels ou GPS, ont été très importantes. Il est aujourd'hui possible d'afficher en temps réel des scènes 3D complètes sur de tels appareils. Cependant, les spécificités ergonomiques et techniques des appareils mobiles risquent de rendre l'interaction avec l'environnement 3D difficile à réaliser efficacement. En particulier, les techniques d'interaction 3D développées pour les ordinateurs de bureau risquent de souffrir de l'absence de souris.

Dans ce mémoire, nous nous intéresserons aux contraintes et aux spécificités des terminaux mobiles et étudierons leur influence pour des tâches d'interaction 3D. La première partie du mémoire constitue une introduction aux différents domaines abordés dans cette thèse. Nous y définirons le terme de "terminal mobile" et aborderons les spécificités technologiques et ergonomique de ces appareils. Nous introduirons également les notions essentielles de géométrie et d'interaction 3D.

Nous nous intéresserons dans un second temps à l'interaction 3D avec un contrôle direct, utilisé dans beaucoup d'applications 3D. Dans ce mode d'interaction, les actions de l'utilisateur produisent des modifications dans l'environnement 3D en temps réel. Après une analyse de l'existant, nous présenterons nos travaux avec une technique d'inspection de modèles 3D qui utilise des informations telles que la profondeur ou la présence de contours dans l'image afin de calculer des mouvements de caméra adaptés au contexte. La suite de la deuxième partie sera consacrée à l'étude de l'apport de l'utilisation de deux doigts sur un écran tactile multi-points.

La troisième et dernière partie de ce mémoire est consacrée à l'interaction 3D "planifiée". Ce mode d'interaction est plus adapté aux contraintes des terminaux mobiles et possède l'avantage d'offrir une interaction d'un plus haut niveau que ne l'offre le contrôle direct. Nous verrons dans cette partie la définition exacte de ce mode d'interaction, ainsi que les différences, les forces et les faiblesses de cette manière d'interagir avec un environnement 3D. Nous présenterons dans cette partie nos contributions, à savoir une technique de sélection de point 3D utilisable à l'aide des touches de l'appareil, ainsi qu'une technique de placement de caméra simple à utiliser mais offrant un contrôle avancé, qui permet aux utilisateurs, novices ou experts, de spécifier un point de vue à l'aide de quelques gestes simples seulement.

Abstract

MOBILE devices, such as phones, PDAs or GPS, are getting more and more powerful. They are now able to compute and to display 3D scenes in real time. However, mobile devices' ergonomics and technical limitations may make 3D interaction hard to achieve efficiently. In particular, 3D interaction techniques developed for desktop computers may be affected by the lack of mouse.

In this thesis, we will study mobile devices' constraints and specificities, and their influences on 3D interaction tasks. The first part of this thesis will be an introduction to the domains addressed in this thesis. We will define the term "mobile device" and analyze their technical and ergonomic specificities. We will also define some 3D geometry and 3D interaction basis.

In the second part of the thesis, we will talk about 3D interaction with direct control, which is used in many 3D applications, where the user's actions induce modifications into the 3D environment in real time. After presenting the related works, we will introduce a 3D inspection technique we have developed. This technique uses depth and contours information in order to compute camera movements adapted to the current context. The following of this second part will be dedicated to a study about benefits of using two fingers on a multitouch screen.

The third part of this thesis is dedicated to "planned 3D interaction". This kind of control is more adapted to the mobile devices' constraints and offers a higher level of interaction than direct control. We will define and analyze planned 3D interaction in this part. We will then present our contributions : a key based 3D navigation technique, and a 3D camera positioning technique based on some gestures which allows both expert and novice users to easily specify a point of view.

Table des matières

Table des figures

Liste des tableaux

Introduction

Motivations

LES importantes évolutions technologiques de ces dernières années ont permis une miniaturisation des ordinateurs. Initialement, machines fixes et encombrantes, ils sont maintenant devenus portables : PDA, téléphones portables, GPS, consoles de jeux : les Terminaux Mobiles (TM) occupent désormais une place importante de notre vie quotidienne.

Si les premières générations d'appareils mobiles ne servaient qu'à traiter de faibles quantités de données, telles du texte, des rendez-vous ou des agendas, il en est tout autre aujourd'hui. En effet, désormais, ces appareils peuvent effectuer des calculs plus importants, au point de pouvoir afficher en temps réel des objets voire des scènes complètes en trois dimensions (3D).. Plusieurs études ont montré qu'il était possible d'envisager l'utilisation de terminaux mobiles pour des applications de Réalité Virtuelle (RV) (Fitzmaurice *et al.*, 1993; Hwang *et al.*, 2006). Au delà de la performance technologique que cela implique, une question fondamentale doit être posée : si, avec les configurations classiques des ordinateurs de bureau, les techniques d'interaction 3D possèdent une certaine maturité, qu'en est-il dans le monde des appareils mobiles ? La démarche qui a été suivie jusqu'ici fut principalement d'adapter les techniques d'interaction 3D classiques sur les terminaux mobiles, en se basant sur l'hypothèse que si cela fonctionne bien sur PC, pourquoi cela ne marcherait-il pas sur un PC miniature ?

Malheureusement, comme nous le verrons dans ce mémoire, les contraintes inhérentes aux terminaux mobiles sont différentes de celles des stations de travail classiques. Ainsi, par exemple, si la souris est devenue un outil de pointage indispensable aux ordinateurs de bureau, elle n'est pas utilisable dans le contexte de mobilité qu'implique l'utilisation d'un terminal mobile. Une solution possible était d'avoir recours à un écran tactile, ce qui a permis leur succès.

La frontière technologique séparant les terminaux mobiles des stations de travail est toujours importante, mais s'amenuise au fil du temps. Aujourd'hui, beaucoup d'applications, de bureautique principalement, existent sur ces deux types d'appareils. Ceci permet à l'utilisateur mobile de continuer à travailler lorsqu'il n'est pas à son bureau ou de rester en contact

avec son entourage grâce au développement de l'Internet mobile. Toutefois, les terminaux mobiles possèdent une force absente des stations de travail : ils sont petits, légers, et peuvent être déplacés librement. Il est alors possible d'imaginer un spectre d'applications beaucoup plus grand que celui des stations de travail.

Dans le cas précis des applications 3D, afficher une scène 3D sur un TM représente toujours un axe de recherche important. L'équipe Iparla, dans laquelle a été effectuée cette thèse, s'intéresse à la visualisation et à l'interaction pour des utilisateurs mobiles. Le développement de techniques de visualisation et d'interaction 3D sur un terminal mobile est donc un des axes de recherche de cette équipe. Afficher et interagir avec des données 3D sur un terminal mobile peut être utile dans diverses situations, dont voici quelques exemples :

Divertissement Sur les consoles de jeux vidéos de salon, les jeux utilisent depuis plusieurs années des données 3D, et les joueurs ont maintenant l'habitude de se déplacer et d'interagir avec l'environnement virtuel qui leur est présenté. A cause des contraintes technologiques des premières années, les jeux présents sur les TM tels les téléphones portables se sont bien souvent limités à des jeux 2D comme des jeux de cartes ou des jeux de réflexion.

Avec les progrès technologiques de ces dernières années, les premiers jeux en 3D mobiles sont apparus sur des consoles portables, puis sur certains téléphones ou assistants personnels. Dans les prochaines années, on pourra envisager offrir à l'utilisateur mobile des expériences vidéo-ludique proches de celles offertes par sa console de salon.

Aide à la navigation Les systèmes de navigation personnels, qui utilisent généralement le système GPS (*Global Positioning System*) afin de positionner l'utilisateur, ont connu un nombre de ventes exceptionnel en peu de temps, dépassant même les chiffres des téléphones portables en terme de progression. Les nouveaux modèles peuvent afficher la carte en 3D, donnant au voyageur des informations plus pertinentes sur son environnement. Autoriser l'utilisateur à interagir avec l'environnement virtuel affiché sur son GPS lui permettrait, par exemple, de placer son point d'arrivée plus précisément, ou de visiter son voisinage afin de repérer les lieux intéressants.

De plus, le développement important des services de cartographie basés sur des relevés topographiques et satellites, tels *Google Maps* ou *Microsoft Live Maps*, et leur portage sur terminaux mobiles, fournissent à l'utilisateur un confort et une efficacité d'utilisation jusque là inexistants.

Tourisme Les fascicules des sites historiques ne donnent aux visiteurs que des informations statiques. Fournir, par exemple, aux visiteurs d'un site archéologique un ordinateur de poche avec des représentations 3D des lieux dans leur état d'origine faciliterait leur compréhension de ce qui les entoure.

Maintenance Dans de nombreux domaines de l'industrie, des techniciens sont amenés à inspecter des machines complexes. Habituellement, le repérage d'une pièce particulière se fait sur un plan papier. Cependant, les mécanismes étant de plus en plus complexes, fournir une représentation 3D permettrait au technicien de se repérer plus facilement, et d'obtenir plus de renseignements sur l'appareillage étudié. Ce modèle 3D pourrait alors être affiché sur un terminal mobile, afin d'afficher les informations dans leur contexte. Il est alors important de fournir des mécanismes d'observation et de manipulation de l'objet 3D

qui soient efficaces.

Si la technologie permet aujourd'hui d'afficher en temps réel des scènes 3D complètes sur TM, le problème du contrôle de ces scènes reste ouvert.

Objectifs

L'objectif principal de cette thèse est d'analyser les particularités technologiques et d'utilisation des terminaux mobiles afin d'étudier et de développer des techniques d'interaction 3D adaptées à ces appareils. En effet, interagir avec une scène 3D est une tâche complexe car elle demande de contrôler beaucoup de paramètres simultanément. Il sera donc important de prendre en compte la difficulté d'*apprentissage*, la difficulté d'*utilisation* ainsi que les *performances* d'utilisation. Par exemple, dans beaucoup de logiciels destinés au grand public, les techniques d'interaction présentes sont faciles d'apprentissage afin d'être utilisables par un grand nombre de personnes, mais peuvent ne pas être performantes. Par contre, beaucoup de logiciels de visualisation 3D actuels sont destinés à des experts du domaine, et sont de ce fait difficiles à utiliser pour un novice, mais sont très performantes une fois maîtrisées. Un de nos objectifs est de rendre l'interaction 3D accessible à tous, c'est à dire de développer des techniques d'interaction 3D qui soient à la fois performantes et faciles d'apprentissage et d'utilisation.

Notre réflexion a suivi les évolutions importantes qu'ont subies les terminaux mobiles au cours de ces dernières années : des premiers modèles ne possédant que des touches comme seul périphérique d'interaction aux appareils actuels équipés d'écrans tactiles pouvant détecter la présence de plusieurs doigts, les terminaux mobiles, et leur utilisation, se sont profondément transformés en peu de temps.

Plan du mémoire

Le présent mémoire est divisé en trois grandes parties.

La partie I constitue une introduction aux différents domaines abordés dans ce mémoire. Nous définirons le terme de "terminal mobile" dans le chapitre 1, où la gamme des périphériques existants pour ces appareils sera présentée. Les problèmes d'ergonomie, matérielle et logicielle, sont abordés dans le chapitre 2. Enfin, le chapitre 3 est consacré aux notions liées à l'interaction 3D.

Dans la partie II, nous nous intéresserons à l'interaction 3D par contrôle direct. Après une analyse des techniques existantes (chapitre 5), nous présenterons au chapitre 6 une technique d'inspection que nous avons développée appelée ScrutiCam. Puis, nous proposerons un ensemble de techniques d'interaction 3D utilisables à l'aide de plusieurs doigts au chapitre 7. Enfin, le chapitre 8 est consacré à une étude concernant l'apport de l'utilisation de plusieurs doigts pour des tâches basiques d'interaction 3D.

La troisième partie est dédiée à l'interaction 3D planifiée. Une analyse des techniques d'interaction 3D utilisant ce principe est réalisée au chapitre 10. Puis, nous présenterons les différences existantes entre l'interaction 3D par contrôle direct et par contrôle planifié au chapitre 11. Enfin, les chapitres 12 et 13 présentent deux techniques de navigation que nous avons développées et qui utilisent un contrôle planifié.

Première partie

Introduction Générale

CHAPITRE 1

Les terminaux mobiles

CE mémoire est consacré à l'étude de l'interaction 3D sur terminaux mobiles. Afin de pleinement comprendre les buts et les enjeux de cette étude, il convient de définir préalablement ce qu'est un terminal mobile, ainsi que les caractéristiques que possède un tel appareil.

Dans ce chapitre, nous définirons dans un premier temps le concept de terminal mobile. Nous verrons également les différentes classes d'appareils appartenant à la famille des terminaux mobiles. Nous étudierons par la suite les différents périphériques d'entrée existant sur de telles machines.

1.1 Les différentes familles de terminaux mobiles

Les "terminaux mobiles", abrégés en *TM* tout au long de ce mémoire, sont des ordinateurs possédant un certain nombre de caractéristiques qui les rendent différents des ordinateurs *de bureau*. En effet, ces derniers sont destinés à être utilisés de manière *statique*, la plupart du temps en étant assis face à un bureau. Ainsi, les ordinateurs traditionnels, composés d'une unité centrale et de périphériques, ainsi que les ordinateurs portables, entrent dans la catégorie des ordinateurs de bureau.

Le terme "terminal mobile" a longtemps été utilisé pour désigner des appareils présents par exemple dans des voitures de police ou dans les taxis. Ne possédant pas ou peu de ressources et de puissance de calcul, ils étaient dotés d'un écran sur lequel apparaissent les informations envoyées par un serveur, où tous les calculs étaient effectués.

Néanmoins, ce terme est de nos jours utilisé pour désigner une plus large gamme d'appareils mobiles. Il correspond à une traduction possible de l'expression *mobile device* en anglais, et est synonyme de *dispositif mobile* ou *appareil mobile*. Les terminaux mobiles appartiennent à une famille de matériels possédant des caractéristiques communes :

- Une taille et un poids prévus pour une utilisation en mobilité. Il est nécessaire que l'utilisateur puisse se servir de l'appareil tout en étant debout, voire en marchant.
- Les composants (affichage, calcul, périphérique de saisie) regroupés en un seul appareil.
- Un fonctionnement autonome, sans câbles et sans branchements extérieurs (sauf temporairement, pour le rechargement ou la synchronisation par exemple).
- Une connexion, que ce soit GSM, GPRS, 3G Wifi, ou autre ; l'appareil doit pouvoir

communiquer avec l'extérieur.
- Une architecture matérielle basée sur une faible consommation électrique, afin d'offrir une autonomie importante. Les processeurs ARM ont par exemple longtemps été utilisés sur de nombreux terminaux mobiles.

Il existe différentes familles de TM, que nous présentons dans cette section. Nous verrons ensuite les périphériques d'entrée les plus courants sur ces appareils.

1.1.1 Téléphones portables

Les *téléphones portables*, ont initialement été créés pour la seule tâche de passer et recevoir des appels. Au fil des années, ils se sont enrichis de fonctionnalités comme l'envoi de messages textuels, l'accès à Internet, la gestion d'agendas, etc. Ces téléphones évolués portent aujourd'hui le nom de *smartphones*. Ils ne sont plus limités à la seule fonction de téléphone, et possèdent aujourd'hui des fonctions d'agenda, de navigation web, de client de messagerie, de caméra numérique, de lecteur de musique et de vidéos, de console de jeux, et bien d'autres encore. Il est souvent possible d'installer de nouvelles applications, élargissant encore le spectre des utilisations de ces appareils.

1.1.2 Assistants personnels

Les *assistants personnels* (ou *PDA : Personal Digital Assistant*) sont des petits ordinateurs de poche, apparus au début des années 1990. Ils disposent généralement d'un écran tactile (de résolution VGA la plupart des cas) et de quelques touches de fonction. Beaucoup de PDA ne sont pas pourvus de fonction de téléphonie.

Les PDA sont principalement utilisés pour leur fonction d'agenda, de répertoire téléphonique ou de bloc-notes, mais aussi, depuis leur avancée technologique de ces dernières années, des fonctions multimédia telles lecteur de musique ou de vidéo. Il est, dans la plupart des cas, possible d'installer de nouvelles applications facilement.

Il est à noter que nous assistons aujourd'hui à une convergence des téléphones portables et des assistants personnels, qui possèdent des caractéristiques techniques et logicielles comparables. Les dernières générations de téléphones portables, comme l'iPhone, le Palm Pre ou la gamme de téléphones Android possèdent autant de fonctions que les assistants personnels, tandis que ces derniers se sont, de leur côté, dotés de fonctions téléphoniques.

1.1.3 Consoles de jeux-vidéos portables

Les consoles de jeux portables existent depuis le début des années 1980, mais ont connu leur essor à partir de 1989, avec l'arrivée du "Game Boy" de Nintendo. De nombreux modèles sont apparus depuis, avec des performances graphiques toujours grandissantes. Aujourd'hui, les consoles portables permettent d'afficher des scènes 3D de qualité proche de la génération précédente des consoles de salon.

De nos jours, les deux principales consoles portables sont la *PSP* de Sony, commercialisée en 2004 et affichant de très bonnes capacités 3D, et la *DS* de Nintendo, possédant la particularité d'être équipée d'un écran tactile. La *N-Gage* est un téléphone portable développé par

Nokia qui possède également toutes les caractéristiques d'une console de jeu. Notons également l'arrivée prochaine de la console "Open Pandora", une console portable tournant sur un système GNU/Linux, équipée d'une puce graphique OpenGL|ES 2.0.

1.1.4 PC ultra portables

Ces ordinateurs sont simplement des versions réduites des ordinateurs de bureau. Ils sont basés sur la même architecture et fonctionnent avec les mêmes systèmes d'exploitation. Il en existe plusieurs formes :

Les UMPC et TIM Le terme *UMPC* (Ultra Mobile PC) désigne un ordinateur portable de taille très réduite, répondant aux spécifications du concept "Origami" de Microsoft. Ces ordinateurs doivent être équipés d'un écran tactile de moins de 20 cm de diagonale. Le même concept désigne les *Terminaux Internet Mobiles* (ou *MID*, *Mobile Internet Devices* en anglais). La particularité de ces derniers est de fonctionner sous un système Linux plutôt que Windows.

Les Tablet PC Ce sont des PC portables équipés d'un stylet qui permet d'écrire ou de dessiner sur l'écran, comme on le ferait avec un bloc note. Il en existe deux sortes : les *convertibles*, qui possèdent un clavier rabattable, et les *slates*, qui ne sont constitués que d'un écran, avec le stylet comme seul moyen d'interaction.

1.2 Périphériques d'entrée des TM

Les TM possèdent différents périphériques d'entrée, qui diffèrent de ceux disponibles sur des ordinateurs de bureau. Parmi ces périphériques, certains fournissent un *signal numérique*, c'est à dire qu'ils ne fournissent qu'une information à deux états (par exemple, bouton appuyé ou non). Dans le cas contraire, on parle de *signal analogique*.

Il convient de distinguer trois catégories de périphériques : *isotoniques*, *isométriques* et *élastiques*.

Isotoniques Ils peuvent être déplacés librement par l'utilisateur, et ne possèdent pas de résistance. Leur principal inconvénient est qu'ils ne fournissent pas de retour haptique. Le stylet d'un TabletPC est un exemple de périphérique isotonique.

Isométriques Ils possèdent la particularité d'être immobiles. Leur résistance est donc infinie. Ils mesurent le mouvement à partir de la force qui leur est appliquée. Le *trackpoint*, présent sur certains ordinateurs portables est un exemple de périphérique isométrique.

Élastiques Ils sont à mi-chemin entre les isométriques et les isotoniques. La force à appliquer varie en fonction du déplacement. De plus, ils possèdent la particularité de se centrer automatiquement, ce qui leur permet de revenir dans leur position stable de référence. Le joystick est un bon exemple de périphérique élastique.

La suite de cette section présente différents périphériques existants sur les terminaux mobiles actuels.

FIGURE 1.1 – Les différentes classes de terminaux mobiles.

1.2.1 Clavier

Parmi la grande diversité des TM actuels, une caractéristique apparaît comme étant un facteur commun à presque tous ces appareils : la présence de boutons. Ces derniers fournissent un signal numérique au système, et permettent donc un contrôle *discret* de l'application. Dans la plupart des cas, plus particulièrement sur les téléphones portables ou smartphones, moins sur les PDA, les appareils mobiles possèdent un *pavé numérique*. Parallèlement, des *touches de fonction* et un *pavé directionnel* sont souvent présents. Le pavé directionnel permet de se déplacer de manière discrète dans les menus. Les touches de fonction, situées autour du pavé directionnel, permettent de valider, d'annuler, ou d'effectuer une action particulière définie par l'application. la Figure 1.2 présente quelques exemples de claviers de TM.

Ces touches sont parfaitement adaptées à la réalisation de tâches discrètes telles la sélection d'un élément dans une liste ou dans un tableau ou la saisie de texte. Cependant, manipuler et interagir avec une scène 3D avec ce genre de périphérique peut se révéler complexe.

Il est à noter que la tendance actuelle, notamment au niveau des téléphones portables, est de limiter le nombre de boutons "physiques" au profit de boutons "virtuels". L'iPhone par exemple ne possède qu'un seul bouton.

1.2.2 Stylet / écran tactile

Sur la plupart des PDA, et sur tous les TabletPC et UMPC, l'écran est tactile. Il y a dans ce cas deux familles d'écrans tactiles : certains nécessitant l'utilisation d'un stylet (comme sur les TabletPC) , d'autres peuvent être utilisés au doigt. Les écrans tactiles sont des périphériques isotoniques. Il existe de nombreuses technologies d'écrans tactiles, mais les quatre principales sont le *résistif*, le *capacitif*, l'*infrarouge* et les *ondes de surface* (voir Figure 1.3).

1.2.2.1 Résistif

Cette technique est la plus répandue et est utilisée sur la plupart des PDA actuels. Elle utilise la mise en contact de deux couches conductrices par l'action du doigt ou du stylet. L'écran est ici constitué de plusieurs couches : une *dalle de verre*, une *couche électriquement conductrice* puis une *couche de polyester souple* dont la face inférieure est recouverte d'une couche conductrice. Les deux dernières couches sont isolées l'une de l'autre par un grand nombre de petits plots isolants et transparents.

FIGURE 1.2 – Exemples de claviers de téléphones portables.

Lorsque l'on touche l'écran, on enfonce légèrement la couche de polyester qui vient en contact avec une couche conductrice. Le contact des deux couches conductrices induit un changement dans la distribution du courant électrique dans la dalle. Le contrôleur associé à la dalle calcule alors la position du point de contact.

L'avantage de cette technologie est qu'elle est très résistante et qu'elle fonctionne avec n'importe quel dispositif de pointage (doigt, stylet). Le principal inconvénient est qu'elle réduit légèrement la luminosité de l'écran.

1.2.2.2 Capacitif

C'est la technologie la plus ancienne. Elle consiste à rajouter un revêtement conducteur métallique sur la surface de verre d'un écran classique. Une tension est ensuite appliquée aux quatre coins de ce revêtement. Lorsqu'il est au repos, un champ électrique uniforme existe sur la surface conductrice. Lorsqu'il touche l'écran, l'utilisateur devient lui même une partie du circuit électrique, créant un couplage avec la surface de l'écran. Ainsi, une quantité de courant est captée par le point de contact. La valeur de courant mesurée aux quatre coins de l'écran est alors proportionnelle à la distance par rapport au point de contact. Cette technologie est utilisée dans l'iPhone d'Apple et lui permet de bénéficier d'un écran tactile en plusieurs points simultanément.

La principale limitation de cette technologie est que le contact avec l'écran doit être fait avec une surface conductrice. Ainsi, il est impossible de l'utiliser avec des gants en extérieur ou avec un stylet non métallique. Par contre, cette technologie ne réduit pas la luminosité de l'écran.

1.2.2.3 Infrarouge

Cette technologie est la plus simple à mettre en oeuvre. En effet, elle consiste à créer une grille de lumière devant l'écran de l'ordinateur : sur les bords de l'écran (horizontal et vertical) sont installés des diodes infrarouges et des phototransistors. Lorsque l'utilisateur touche l'écran, il intercepte des rayons lumineux, et la position est alors calculée en fonction des photo-transistors ayant détecté l'absence de lumière. L'avantage ici est que cette grille peut être installée sur n'importe quel type d'écran. Malheureusement, la précision est relativement faible et le coût plutôt élevé.

1.2.2.4 Ondes de surface

Cette technologie est la plus récente des quatre. Elle utilise des transducteurs piézo-électriques placés sur un bord horizontal et vertical de l'écran. Ils transforment un signal électrique en une onde ultrasonique qui se propage le long d'une dalle de verre superposée à l'écran. Grâce à des réflecteurs placés le long des bords de l'écran, cette onde est redirigée vers des transducteurs de réception, placés à l'opposé des transducteurs d'émission. Comme la vitesse de propagation et la taille de l'écran sont connues, le temps d'arrivée des ondes est déterminé avec précision. Si l'on touche l'écran, une partie de l'onde acoustique est absorbée. L'analyse du temps d'arrivée de l'onde atténuée permet alors de déduire la position du

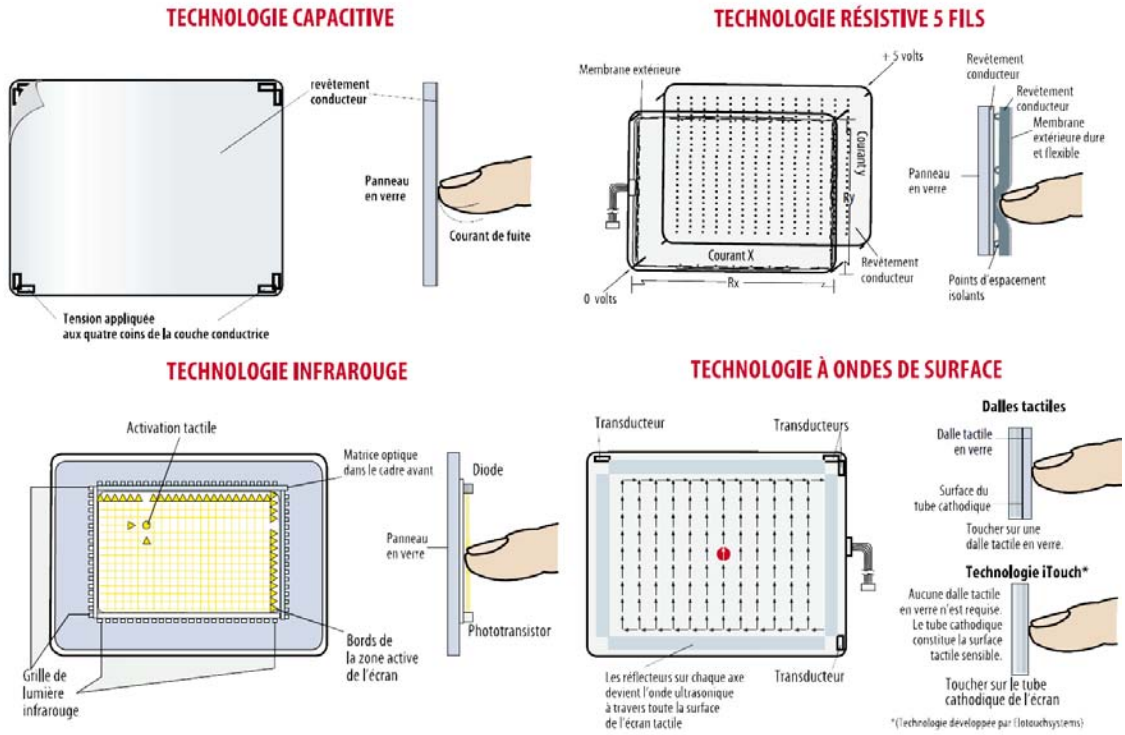


FIGURE 1.3 – Les différentes technologies d'écran tactile (Mesures, 2001).

point de contact. Notons qu'en mesurant la quantité d'énergie absorbée, il est possible d'obtenir une information sur la composante de profondeur, ce qui permet de modifier l'action en fonction de la distance entre le doigt et l'écran.

1.2.2.5 Autre solution : utilisation d'un stylet *radio*

Sur les TabletPC, en revanche, la technologie utilisée est tout autre, puisque l'interaction ne peut se faire qu'avec un stylet bien particulier, et non avec le doigt. Dans ce cas, on ne parle pas d'écran tactile mais de *surface active*. La technologie utilisée ici se base sur des ondes radio.

1.2.3 Écrans tactiles multi-points

Apparus il y a quelques années, mais popularisés récemment, les écrans multi-points permettent d'utiliser plusieurs doigts comme moyen d'interaction. En effet, ces écrans sont capables de détecter la présence et de calculer la position de plusieurs points de contacts. Cette caractéristique est présente sur l'iPhone d'Apple, et a contribué à son succès commercial. Toutefois, les premiers prototypes d'écrans multi-points sont plus anciens. Plusieurs technologies d'écrans tactiles multi-points ont été développées.

1.2.3.1 Électronique

Les technologies d'écran tactile citées précédemment permettent, au prix de quelques améliorations, de détecter la présence de plusieurs points de contact. Ainsi, la société Stantum a développé une technologie, basée sur un écran tactile résistif, qui permet de détecter un nombre infini de points de contacts. Leur premier produit, appelé Lemur, était destiné au contrôle d'applications musicales. Ils ont par la suite commercialisé un écran tactile multi-points indépendant muni de leur technologie. L'écran multi-points de l'iPhone et le tabletPc Latitude Xt de Dell sont, eux, basés sur la technologie d'écrans capacitifs.

Enfin, la société Touchco(Touchco) a développé la technologie IFSR (*Interpolating Force-Sensitive Resistance*), qui possède l'avantage de détecter la pression exercée avec une grande précision. La dalle tactile utilisée est souple, ce qui permet de la placer sur n'importe quelle surface, plane ou non.

Ce type d'écran possède l'avantage d'offrir une très bonne réactivité, car les calculs de position sont réalisés par un contrôleur électronique intégré à l'écran. Cependant, leur fabrication est onéreuse et ils ne fournissent souvent que l'information de position, et pas de pression ou de taille.

1.2.3.2 Vidéo

Il est également possible de détecter la présence de plusieurs doigts sur un écran en utilisant une caméra vidéo. Ce principe est utilisé dans plusieurs tables de travail, souvent appelées *tabletops*, utilisées dans un cadre de travail collaboratif. Cette technologie, appelée *FTIR* pour "Frustrated Total Internal Reflexion", est basée sur la projection arrière de l'image. Elle a été développée par Han (2005). Le principe est d'éclairer l'arrière d'une surface semi-transparente à l'aide de LED infrarouges : lorsque l'utilisateur pose un doigt sur l'écran, la zone de contact sera éclaircie par les LED et détectée par une caméra située en arrière.

Cette technique possède l'avantage d'être facile à fabriquer et peu onéreuse. Elle est utilisé par plusieurs produits commerciaux, comme Surface de Microsoft ou ILight et CubeTile (de la Rivière *et al.*, 2008) d'Immersion. Il est également possible, avec cette technologie, de connaître la taille et la forme du contact. Malheureusement, les traitements vidéo nécessaires à l'identification des points de contact sont assez importants, et un temps de latence est souvent présent. De plus, cette technologie de pourra être utilisée sur un terminal mobile, car elle occupe beaucoup de place.

1.2.3.3 Simulation logicielle

Il est parfois possible de simuler la détection de plusieurs points de contact sur un écran tactile résistif. Beaucoup d'appareils mobiles, pour des raisons de coût de fabrication et de facilité d'utilisation, sont équipés d'un écran tactile basé sur la technologie d'écran *résistive*. Cette technologie offre l'avantage, contrairement aux écrans *capacitifs* de pouvoir être utilisée avec n'importe quel dispositif de pointage, que ce soit un stylet ou un doigt (ganté ou non).

Cette technologie possède une caractéristique intéressante : lorsque l'on appuie sur l'écran en plusieurs points simultanément, la position du curseur correspond au barycentre de tous ces points. En effet, la position du doigt sur l'écran étant basée sur la diffusion de courant le long de la surface tactile, si plusieurs points de contact sont appliqués en même temps, la tension mesurée renverra la position moyenne des points de contact.

Matsushita *et al.* (2000) ont utilisé ces caractéristiques pour calculer la position approximative d'un deuxième doigt sur un écran tactile. Le principe, illustré sur la Figure 1.4 est simple : une fois le premier doigt positionné sur l'écran, sa position P_1 est enregistrée par le système. Ensuite, lorsque l'on placera un deuxième doigt à la position P_2 sur l'écran, le curseur sera déplacé en P_m, qui correspond au milieu des points P_1 et P_2. Le déplacement du curseur de P_1 en P_m produit ainsi un changement brutal de coordonnées. La position du point P_2 peut donc être calculée par $P_2 = P_1 + 2 \times \overrightarrow{P_1 P_m}$. Cette technique peut être considérée comme un *pseudo multi-points*.

Dans leurs applications, la position du premier point de contact est utilisée comme modificateur, pour changer de mode ou pour définir un centre de rotation, et la position du deuxième point de contact est envoyée au système comme position du curseur.

Loviscach (2007) est allé plus loin dans cette direction. Il a décrit les moyens électroniques et logiciels nécessaires pour récupérer des informations telles que la pression exercée en un point ou la position approximative de deux points de contact sur l'écran. Dans ce cas, la position du deuxième doigt peut être calculée indépendamment de celle du premier point de contact, contrairement à la technique de Matsushita et al, qui demandait à ce que le premier point de contact reste immobile. Ces informations sont utilisées pour diverses applications, telles la peinture numérique, la sélection de plusieurs entités ou la manipulation de cartes (voir Figure 1.5). Cependant, cette technique requiert l'adjonction d'un circuit électronique, connecté au port USB de la machine.

Contraintes d'utilisation

La détection de la présence de plusieurs points de contact sur l'écran est approximative. En effet, elle est basée sur l'apparition d'un changement brutal des coordonnées du curseur. Par conséquent, la détection de la présence d'un nouveau point de contact dépend d'un seuil à la fois spatial δ_p et temporel δ_t : si le curseur de la souris a effectué un déplacement supérieur à δ_p dans un temps inférieur à δ_t, un saut est détecté. Il convient donc de régler ces deux paramètres correctement, afin de ne pas détecter de sauts lorsque l'utilisateur déplace son doigt rapidement. De plus, il sera difficile de détecter la présence d'un nouveau point de contact si celui ci est proche du premier.

Une des conséquences de cette méthode de détection est que les différents points de contact doivent être appuyés successivement, et non simultanément, afin de pouvoir détecter le saut de coordonnées entre chaque nouvel appui.

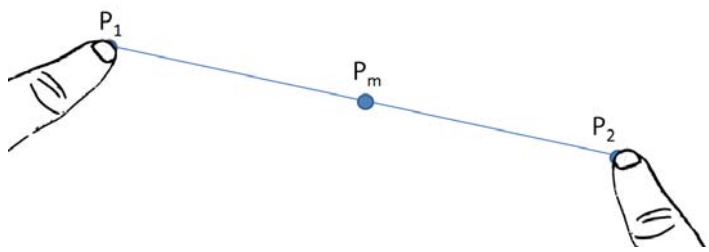


FIGURE 1.4 – **Estimation de la position du deuxième point de contact.** Les points P_1 et P_2 correspondent au premier et au second point de contact. Le point P_m est la position du curseur calculée par le système.

FIGURE 1.5 – Sélection de plusieurs éléments sur un écran tactile avec la technique de Loviscach (2007).

Enfin, seule la position du dernier point de contact est mise à jour, les autres sont supposés rester fixes par l'algorithme. De même, le relâchement d'un point de contact produit lui aussi un saut. Par conséquent, il n'est pas possible de différencier un relâchement d'un nouvel appui. Il faut donc bien définir la tâche d'interaction et le nombre de points de contact associés, afin d'éviter toute ambiguïté.

1.2.4 Caméra

Depuis l'apparition et le développement des appareils photo numériques, beaucoup de progrès ont été effectués, tant au niveau matériel, avec des capteurs optiques de plus en plus précis, qu'au niveau logiciel, avec le développement d'algorithmes de traitement et d'analyse d'images toujours plus efficaces. Au début des années 2000 sont apparus les premiers téléphones portables équipés d'un appareil photo numérique. Celui-ci avait la plupart du temps une résolution assez faible (VGA), et une profondeur de couleur elle aussi limitée (256 couleurs la plupart du temps). Cependant, d'énormes progrès ont été effectués dans ce domaine, et la grande majorité des appareils mobiles d'aujourd'hui sont équipés d'une caméra possédant une résolution comparable aux appareils photo numériques. De plus, ils sont également capables de capturer des flux vidéo en temps réel, pour faire de la visioconférence par exemple.

1.2.5 Joysticks

Les joysticks, présents depuis de nombreuses années sur les consoles de jeux vidéo et les ordinateurs de bureau, sont des périphériques élastiques ou isométriques fournissant un signal analogique. Une étude récente (Casiez et Vogel, 2008) a évalué l'influence de l'élasticité du joystick pour des tâches dites de "contrôle de vitesse", afin de calculer la meilleure résistance à appliquer à un joystick, en fonction de la tâche à accomplir. Certains terminaux mobiles sont équipés de joysticks. C'est le cas de la console portable PSP™ de Sony, qui

intègre un joystick analogique, ainsi que de prototypes expérimentaux (Kawachiya et Ishikawa, 1998). Des brevets ont été déposés concernant la mise en place de joysticks au sein des appareils. Il est par exemple envisagé d'utiliser le stylet comme manche du joystick (Wong John Patrick, 2007), d'intégrer le joystick dans les touches de direction (IndustrialDesign, 2009), voire même de transformer le clavier entier en joystick (Kim, 2009). Il est aussi possible de rajouter un joystick à n'importe quel appareil mobile équipé de BlueTooth. Enfin, RubberEdge (Casiez *et al.*, 2007) est un projet visant à regrouper, dans un même périphérique, un contrôle isométrique et un contrôle élastique, qui possèderait alors les avantages d'un écran tactile et d'un joystick.

La Figure 1.6 présente différents joysticks pour terminaux mobiles, commercialisés ou à l'état de prototypes.

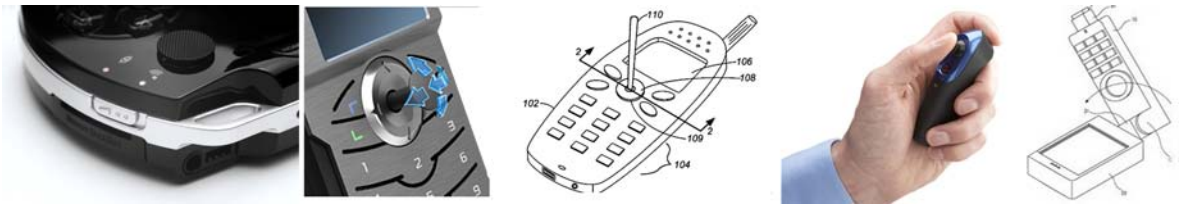


FIGURE 1.6 – Différents Joysticks présents sur des terminaux mobiles : PSP™ de Sony, le projet CompactAnalogThumbstick (IndustrialDesign, 2009), un brevet de Nokia(Wong John Patrick, 2007), un joystick BlueTooth, ainsi qu'un brevet de Samsung (Kim, 2009).

1.2.6 Capteurs

De plus en plus présent sur les smartphones et PDA récents, le GPS permet d'obtenir les coordonnées géographiques de l'utilisateur. Cependant, sa précision n'est que de 20m dans la plupart des cas (5m dans le cas particulier du GPS différentiel), ce qui ne permet pas de détecter des mouvements précis. De plus, les versions d'entrée de gamme, souvent présentes dans les smartphones ou les PDA, ne donnent aucune information quand à l'orientation de l'appareil, ce qui réduit ici aussi son utilité dans notre thématique. Il existe cependant des GPS couplés à une boussole numérique qui permettent de connaître l'orientation de l'utilisateur, comme par exemple l'iPhone 3GS.

D'autres types de capteurs existent sur le marché. Par exemple, Hinckley et al.(Hinckley *et al.*, 2000) utilisent un capteur d'orientation et des accéléromètres afin de détecter l'orientation du téléphone ou de faire défiler les pages affichées à l'écran. L'iPhone d'Apple possède un système similaire afin de détecter comment l'utilisateur tient son appareil et tourner l'écran en conséquence.

1.2.7 Microphone

Enfin, la plupart des appareils mobiles sont avant tout destinés à téléphoner. Ils intègrent de ce fait tous un microphone. Les recherches en acoustique et en traitement de signal sonore ont permis de développer des méthodes de reconnaissance de sons, de mots ou de phrases. Cette "commande vocale" est utilisée sur beaucoup d'appareils pour composer automatiquement le numéro de téléphone d'un contact en prononçant son nom par exemple.

Le chapitre suivant est consacré aux particularités et aux problèmes ergonomiques pouvant être rencontrés lors de l'utilisation de terminaux mobiles. Nous y aborderons des questions concernant l'ergonomie générale des terminaux mobiles, les différentes manières de les manipuler, ainsi que les problèmes d'ergonomie logicielle.

CHAPITRE 2

Ergonomie Mobile

Les terminaux mobiles possèdent des caractéristiques particulières qui les rendent très différents des ordinateurs de bureau. Ces différences peuvent provenir des périphériques d'entrée, de limitations techniques ou de différences dans l'ergonomie et la manière d'utiliser ces appareils. Dans ce chapitre, nous verrons les principales caractéristiques ergonomiques des terminaux mobiles, ainsi que les différences pouvant exister par rapport aux ordinateurs de bureau. Cette étude nous permettra de donner explicitement des contraintes fortes des terminaux mobiles, ainsi que des directives concernant le développement de techniques d'interactions 3D qui prennent en compte ces contraintes.

De nombreux travaux ont été consacrés à l'étude de l'ergonomie des terminaux mobiles. En effet, ceux-ci possèdent des contraintes et des particularités qui font qu'il est difficile d'adapter les applications développées pour des ordinateurs de bureau sur de tels appareils. Des études ont ainsi été réalisées sur la conception d'applications mobiles (Gong et Tarasewich, 2004; Kjeldskov et Kolbe, 2002), proposant des conseils et donnant des directives sur l'ergonomie de telles applications.

Dans ce chapitre, nous aborderons les problèmes d'ergonomie mobile en parlant, dans un premier temps, de concepts généraux de l'ergonomie mobile. Nous verrons dans cette section certaines des contraintes matérielles des terminaux mobiles. Puis, dans la section suivante, nous aborderons les problèmes logiciels pouvant survenir, ainsi que quelques possibles solutions.

2.1 Ergonomie générale

Les caractéristiques physiques des terminaux mobiles font que leur ergonomie générale, la façon de les utiliser, de les tenir, de les manipuler, diffèrent énormément de l'ergonomie des ordinateurs de bureau.

Une première contrainte inhérente aux terminaux mobiles concerne la **taille** de l'appareil. En effet, afin d'être portable, le terminal mobile doit être de petite taille. Il doit par exemple être possible de le tenir dans une main sans peine. C'est pour cette raison que les ordinateurs portables n'entrent pas, dans ce mémoire, dans la catégorie des terminaux mobiles. La **préhension** de l'appareil sera donc particulière.

Il y a deux manières principales d'utiliser un terminal mobile : à une main ou à deux mains. L'utilisation *unimanuelle* est la plus employée (Karlson *et al.*, 2007), car elle permet

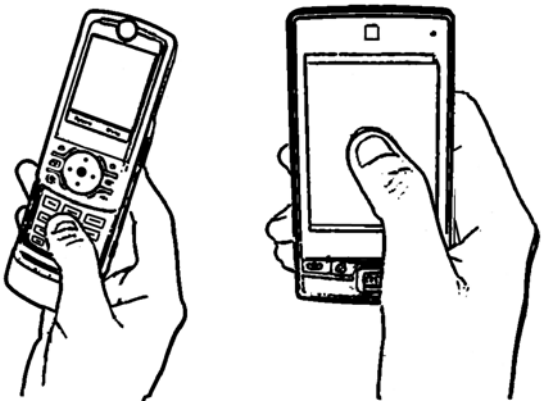


FIGURE 2.1 – Exemples de contrôle unimanuel de terminaux mobiles.

de garder la seconde main libre pour d'autres tâches. Ici, l'utilisateur tient l'appareil dans une main et l'interaction se fait à l'aide du pouce, comme l'illustre la Figure 2.1. Ce type de manipulation est par exemple utilisé lors de la saisie d'un numéro de téléphone.

Cette utilisation pose cependant des problèmes de précision, que nous aborderons dans la section suivante. De plus, il ne sera possible d'utiliser qu'une seule touche du clavier à la fois, car seul le pouce peut interagir avec l'appareil dans ce cas de figure. Perry et Hourcade (2008) ont montré que les utilisateurs étaient plus rapides et plus précis en utilisant leur main dominante pour interagir. Ils ont également montré que les performances étaient comparables lorsque les sujets de l'étude étaient immobiles ou étaient en train de marcher.

L'utilisation *bimanuelle* est principalement employée sur les terminaux à écran tactile. Dans ce cas, l'utilisateur tiens l'appareil avec sa main non dominante tandis que l'autre main est occupée à interagir, à l'aide d'un stylet ou d'un doigt par exemple. Cette position permet une plus grande liberté d'action. Cependant, dans cette configuration, une grande partie de l'écran est occultée : une étude récente a montré que 47% d'un écran de 12 pouces était occulté lors de l'utilisation d'un stylet sur un TabletPC (Vogel *et al.*, 2009). Une possible solution à ce problème a été proposée par Wigdor *et al.* (2007) : interagir par l'arrière de l'appareil, ce qui permet de libérer l'affichage.

Il est également courant de tenir l'appareil dans les deux mains et d'utiliser les deux pouces pour interagir, comme par exemple sur les terminaux mobiles non tactiles lors de la saisie de texte. La Figure 2.2 montre ces différents types d'utilisations bimanuelles.

Dans cette configuration, il sera parfois possible d'utiliser deux touches de l'appareil à la fois. Cependant, les systèmes d'exploitation des terminaux mobiles ne prennent pas tous en compte cette possibilité. De plus, sur beaucoup de terminaux, les touches directionnelles sont physiquement liées les unes aux autres (elles appartiennent par exemple à la même pièce) et il n'est pas possible d'en presser plusieurs à la fois.

La taille réduite des terminaux mobiles implique également une **taille d'écran réduite**. Ceci rend la lecture à l'écran plus difficile que sur un écran de bureau, et il n'est pas possible d'y afficher autant d'informations. En effet, la résolution de l'écran est souvent réduite (VGA sur la plupart des PDA) pour des raisons de coût de fabrication des écrans et d'économie de ressources. La perception est alors différente par rapport à un écran de bureau, car l'angle de vue est limité, principalement à cause de la taille et de l'orientation "portrait" de l'écran,

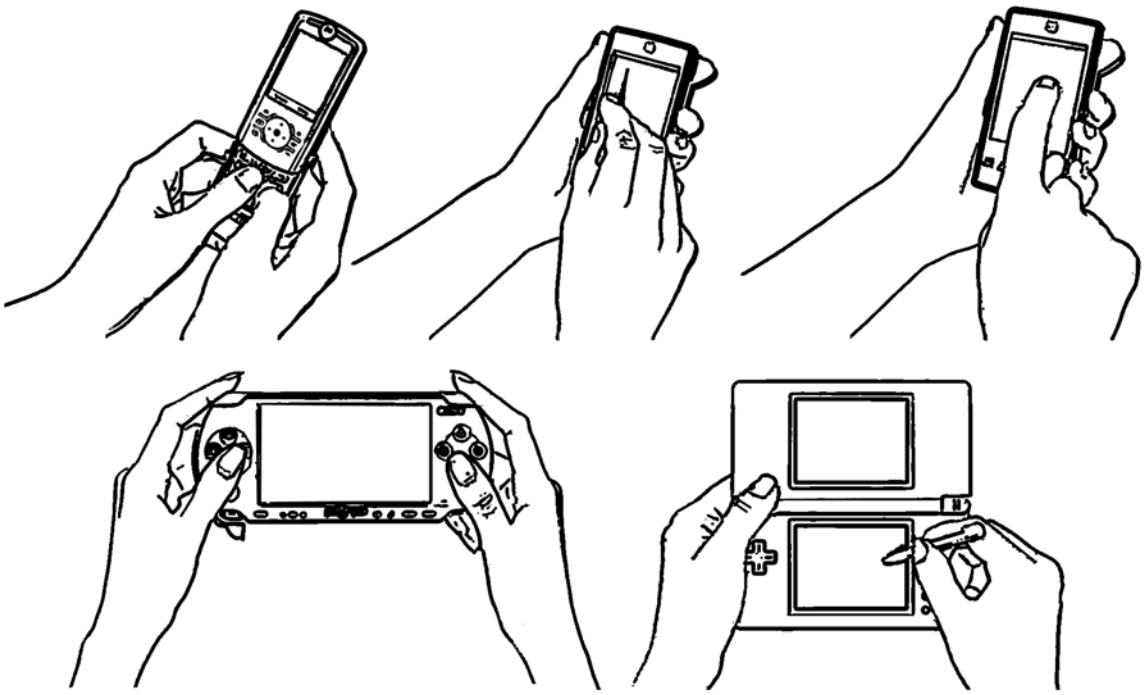


FIGURE 2.2 – Exemples de contrôles bimanuels de terminaux mobiles et de consoles portables.

et la distance de vue doit être réduite, pour que les informations affichées soient toujours visibles.

Il sera donc important de ne pas surcharger l'affichage, notamment en limitant l'utilisation de *menus*, qui occupent une partie importante de l'écran.

Des études ont été réalisées sur les manières d'optimiser l'espace d'affichage (Chittaro, 2006). Une solution possible est l'utilisation de techniques multimodales : il a par exemple été montré que l'utilisation conjointe du son et de l'image permettait de réduire la taille des boutons affichés, afin de gagner de l'espace sur l'écran (Brewster, 2002).

2.2 Ergonomie logicielle

Beaucoup de problèmes ergonomiques des terminaux mobiles peuvent être résolus par des solutions logicielles. Ces programmes permettent de pallier aux contraintes des TM, ainsi que de tirer parti des avantages que procure l'utilisation de ces appareils.

2.2.1 Saisie de texte

L'ergonomie particulière des terminaux a été étudiée depuis l'apparition des premiers appareils portables. Un des premiers problèmes importants à résoudre concernait la saisie de texte. En effet, les premiers téléphones portables ne possédaient qu'un pavé numérique comme unique moyen d'interaction, ce qui est toujours le cas pour beaucoup de téléphones aujourd'hui. Plusieurs méthodes de saisie de texte à l'aide d'un clavier de téléphone ont ainsi été développées (Silfverberg *et al.*, 2000), se basant pour la plupart sur l'utilisation des

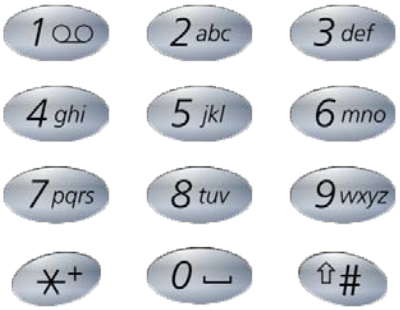


FIGURE 2.3 – Correspondance entre les touches du clavier et les lettres ISO 9995-8 (1994).

12 touches du pavé numérique : les caractères A à Z sont répartis sur touches 2 à 9 dans l'ordre alphabétique. Le placement de ces lettres sur les touches, visible sur la Figure 2.3, est similaire d'un téléphone à l'autre car il est basé sur un standard ISO (9995-8, 1994). Une des premières méthodes de saisie de texte a été le *multi-press*, basé sur l'appui successif d'une même touche afin de sélectionner une lettre particulière : par exemple, la saisie du mot "rue" se fait en pressant la séquence "7-7-7 (pour la lettre "r") 8-8 (pour le "u") 3-3 (pour le "e"). D'autres méthodes, plus efficaces, ont été développées, comme le "T9" (Grover *et al.*, 1998), qui se base sur la prédiction de texte : par exemple, la saisie du mot "rue" se fait en pressant les touches "7-8-3". Cependant, lorsque plusieurs candidats sont possibles, dans notre cas les mots "que" et "sud", l'utilisateur devra choisir dans une liste le mot qui lui convient.

Certains TM actuels sont pourvus d'un clavier alphabétique complet, ce qui facilite la saisie de texte. Cependant, il est alors nécessaire d'augmenter la taille du terminal, car de trop petites touches seraient difficiles à utiliser.

Avec l'avènement des écrans tactiles sur les terminaux mobiles, il a été possible d'utiliser des "claviers virtuels". Ils possèdent l'avantage de présenter toutes les touches d'un clavier standard. Cependant, les icônes des touches risquent d'être de petite taille. D'autres systèmes sont basés sur la reconnaissance d'écriture et sont implémentés dans les systèmes d'exploitation tels que Windows Mobile. Cependant, la vitesse de saisie du texte est plutôt lente car l'humain écrit en moyenne à la vitesse de 15 mots par minute (Gibbs, 1993). Afin de permettre une saisie plus rapide, les techniques basées sur le dessin d'esquisses (*strokes* en anglais) sont souvent utilisées, comme *unistrokes* (Goldberg et Richardson, 1993) (Figure 2.4) ou le système *Graffiti*, développé pour les PDA Palm. Le principe est de créer une grammaire de gestes simples et rapides à dessiner, que le système reconnaîtra plus rapidement. Wobbrock *et al.* (2003a,b) ont développé une méthode de saisie de texte qui utilise les bords de l'écran : le retour physique permet ainsi d'accroître les performances de l'utilisateur par rapport aux systèmes *unistroke* et *Graffiti*.

Une étude (MacKenzie et Soukoreff, 2002) résume la plupart des méthodes de saisie de texte développées pour les terminaux mobile.

La saisie de texte a été la principale difficulté rencontrée pour la conception des premiers téléphones portables. En effet, leurs capacités techniques ne leur permettait pas de traiter de tâches plus complexes. De nos jours, les appareils mobiles sont capables d'exécuter toute

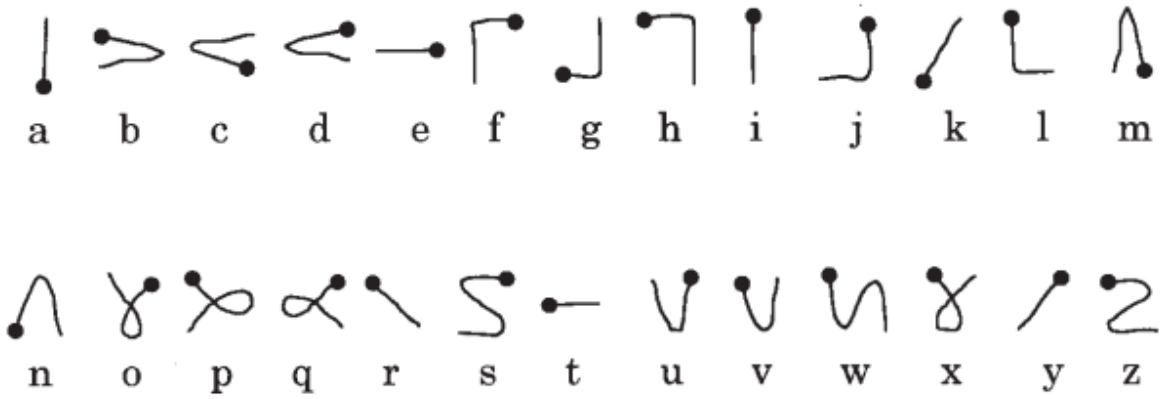


FIGURE 2.4 – Grammaire de gestes d'*unistrokes* (Goldberg et Richardson, 1993).

sorte d'application. Il a donc été important d'étudier comment contrôler une application à l'aide d'un terminal mobile.

2.2.2 Contrôle d'application

Tandis que les premières applications des téléphones portables se limitaient à appeler un numéro ou écrire un message, des applications plus riches sont apparues sur les premiers PDA, puis sur les smartphones. Aujourd'hui, la plupart des terminaux mobiles sont multimédia et permettent d'utiliser plusieurs types d'applications.

Sur les terminaux mobiles seulement équipés d'un clavier, le contrôle des applications est principalement basé sur l'utilisation de menus. En effet, ce mode d'affichage est tout à fait adapté au signal binaire que produit l'appui sur une touche. Cependant, des études ont été menées pour des tâches plus spécifiques, comme l'agrandissement d'une zone particulière (Robbins *et al.*, 2004) ou la sélection d'un point à l'écran (Hachet *et al.*, 2007). Ces techniques sont basées sur une décomposition de l'écran en plusieurs zones, que l'utilisateur sélectionne à l'aide des touches du pavé numérique ou des touches directionnelles.

Lorsque le mobile est équipé d'un écran tactile, il est alors possible d'utiliser des applications se basant sur le paradigme *WIMP* (*Windows, Icons, Menu, Pointing device*), utilisé par la plupart des applications de bureau. Cependant, les contraintes des terminaux mobiles font qu'il n'est pas recommandé, voire qu'il n'est pas possible, d'effectuer un simple portage des applications de bureau sur un terminal mobile.

Tout d'abord, lorsque l'appareil est utilisé de manière unimanuelle, il sera parfois difficile de contrôler une application. En effet, il se peut qu'une partie de l'écran ne soit pas facilement accessible par le pouce, ce qui rend les contrôles situés dans cette zone difficiles à utiliser. Karlson *et al.* (2007) ont analysé les zones accessibles par le pouce sur un terminal mobile utilisé de manière unimanuelle et ont ainsi pu dicter quelques principes. Ils conseillent de privilégier les zones centrales de l'écran, plus faciles d'accès, de bien étudier la position des cibles à atteindre car cela a une influence importante sur les performances des utilisateurs, et d'éviter les mouvements NO-SE (pour les droitiers), qui sont difficiles à réaliser. Ils ont également développé de nouvelles techniques de contrôle d'application qui

prennent en compte ces principes : Applens et LaunchTile (Karlson *et al.*, 2005) (Figure 2.5). Ces techniques sont principalement basées sur l'utilisation de différents niveaux de zoom pour afficher plusieurs applications simultanément, et sur le contrôle de l'application à base de gestes.

Un des avantages des écrans tactiles est qu'il est plus facile de dessiner des formes simples qu'avec une souris. Ainsi, il sera intéressant d'utiliser des techniques d'interaction basées sur l'*esquisse*, aussi appelé *sketching* en anglais. Le principe de ces techniques est de proposer à l'utilisateur de contrôler l'application de manière simple et rapide en dessinant des gestes au sein même de la fenêtre du programme. Scriboli (Hinckley *et al.*, 2005) par exemple est une application de prise de notes qui utilise ce principe. La sélection d'un ou de plusieurs éléments est réalisée en les encerclant et il est possible de faire apparaître un menu contextuel circulaire à l'aide d'un geste appelé "pigtail" (voir Figure 2.6).

2.2.3 Écran tactile et souris

Pour beaucoup de systèmes d'exploitation, un écran tactile n'est pas considéré comme un périphérique particulier, mais émule des évènements de souris, comme le clic ou le déplacement du curseur à une position donnée. Toutefois, des différences importantes existent au niveau ergonomique entre une souris et un écran tactile. Ces différences apparaissent à plusieurs niveaux : au niveau du type de positionnement et de mouvements, de l'utilisation de boutons ainsi qu'au niveau de la précision de la sélection d'un point particulier.

2.2.3.1 Positionnement et mouvements

Sur un écran tactile, la position du curseur est absolue, et correspond à la position pointée, alors qu'avec une souris, elle est relative et est déterminée par les déplacements de l'utilisateur. Ainsi, la position absolue du pointeur rend la manipulation et la sélection directe,

FIGURE 2.5 – Exemples de contrôle unimanuel d'applications : LaunchTile (Karlson *et al.*, 2005).

FIGURE 2.6 – Sélection et affichage d'un menu contextuel dans Scriboli (Hinckley *et al.*, 2005).

car l'utilisateur peut désigner son centre d'intérêt directement. On parle alors d'une **colocalisation** entre le dispositif de pointage (stylet, doigt) et le curseur (ou le point cible).

Parallèlement à cela, avec une souris, il est possible de déplacer le curseur sans cliquer. Ceci permet par exemple de se positionner précisément sur un point avant d'effectuer une action. Cela permet également de réaliser des actions différentes suivant si le bouton est enfoncé ou non. Sur un écran tactile, le déplacement du curseur ne peut se faire que lorsque l'on touche l'écran. Il est donc important de considérer ce point, qui modifie grandement l'ergonomie et l'utilisation de l'appareil.

Des études ont été menées concernant l'émulation de la souris à l'aide d'un écran tactile multi-points (Matejka *et al.*, 2009; Esenther et Ryall, 2006), en utilisant plusieurs doigts : par exemple, l'utilisation d'un seul doigt déplace le curseur sans cliquer, tandis que l'ajout d'un second doigt permet de simuler l'appui sur le bouton gauche de la souris.

2.2.3.2 Utilisation de boutons

Les souris actuelles possèdent plusieurs boutons : les "boutons gauche et droit", ainsi qu'un bouton central qui est, dans la plupart des cas, présenté sous la forme d'une "molette" rotative. Ces boutons permettent d'accéder à des fonctions particulières, comme par exemple ouvrir un menu contextuel. La molette centrale permet également de contrôler une dimension supplémentaire, par exemple pour faire défiler un texte de manière continue.

Les interfaces tactiles se basant sur l'appui du doigt sur l'écran ne possèdent, elles, pas de bouton. Elles se contentent de simuler l'appui sur le bouton gauche de la souris. Ainsi, le fait de poser un doigt sur l'écran envoie un évènement "déplacement de souris" vers la position pointée ainsi qu'un évènement "bouton appuyé" au système, simulant l'appui sur le bouton gauche de la souris avec le curseur positionné à l'endroit désigné.

Afin d'avoir un comportement similaire aux souris, il a été nécessaire de simuler l'utilisation du plusieurs boutons, pour ouvrir le menu contextuel par exemple. Plusieurs méthodes sont possibles(Li *et al.*, 2005) : utiliser un bouton présent sur le stylet (dans le cas des Tablet PC), presser et attendre (comme c'est le cas sur les systèmes Windows Mobile), utiliser une touche de l'appareil (par exemple une touche auxiliaire sur le côté de l'appareil), se baser sur la pression exercée, ou enfin, si cela est possible, utiliser les deux cotés du stylet, chacun correspondant à un bouton particulier. Un contrôle à base de gestes est également utilisé par Roudaut *et al.* (2009) qui distinguent les mouvements de roulement des mouvements de

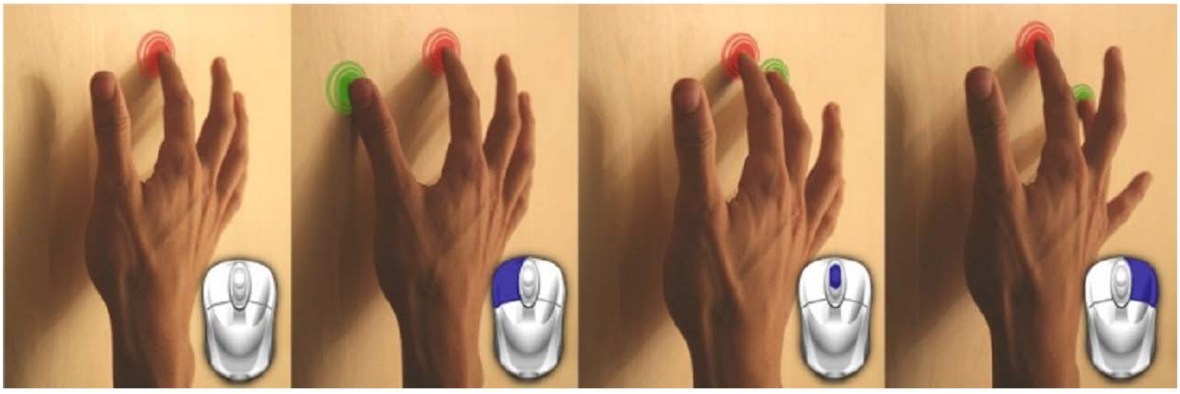


FIGURE 2.7 – Émulation des mouvements de la souris à l'aide d'un écran multi-points (Matejka *et al.*, 2009).

glissement du pouce afin d'augmenter le vocabulaire des gestes possibles.

Il est également possible d'utiliser plusieurs doigts pour simuler la présence de plusieurs boutons (Matejka *et al.*, 2009). Dans ce cas, la différenciation se fait en fonction du doigt de la main qui touche l'écran : l'index sert à pointer, le pouce à déclencher un clic du bouton gauche, le majeur un clic du bouton central, et l'auriculaire un clic du bouton droit, comme l'illustre la Figure 2.7.

2.2.3.3 Précision

Enfin, la principale différence existante concerne la précision de la sélection d'un point particulier de l'écran. En effet, l'écran tactile de l'appareil mobile est souvent utilisé à l'aide d'un doigt, l'index pour une utilisation bimanuelle ou le pouce pour une utilisation unimanuelle. Dans ces deux cas, de gros problèmes de précision apparaissent, dus à la taille des doigts.

Des recherches ont ainsi été conduites de manière à évaluer la taille optimale des cibles à sélectionner à l'aide d'un doigt (Parhi *et al.*, 2006). Cet article donne des recommandations sur la taille optimale des contrôles à utiliser, en fonction du type de tâche concerné. L'étude menée par Perry et Hourcade (2008) a montré que les utilisateurs trouvaient que la sélection de cibles placées au centre de l'écran était plus facile et confortable que la sélection de cibles placées sur les bords de l'écran, même si ces derniers étaient plus précis dans le second cas.

Beaucoup de techniques ont été développées de manière à améliorer la précision de la sélection de points ou d'objets sur un écran tactile, en suivant les recommandations des études menées auparavant. Par exemple, Escape (Yatani *et al.*, 2008) permet de sélectionner de petites cibles très proches en utilisant des gestes : les différentes cibles possèdent une icône indiquant une direction à suivre afin de la sélectionner. Shift (Vogel et Baudisch, 2007) permet d'éviter l'occultation du doigt en décalant la position du curseur au dessus du doigt. De même, MagStick (Roudaut *et al.*, 2008) utilise un principe similaire, mais permet un meilleur contrôle car l'utilisateur peut choisir la direction et la distance du décalage à appliquer. Enfin, TapTap (Roudaut *et al.*, 2008) utilise un agrandissement du voisinage de la cible à sélectionner afin d'offrir une plus grande précision.

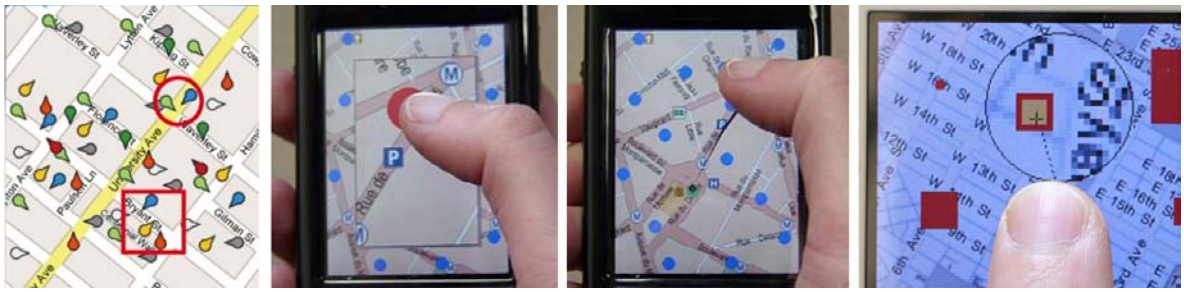


FIGURE 2.8 – Sélection précise sur écran tactile : Escape (Yatani *et al.*, 2008), Taptap, MagStick (Roudaut *et al.*, 2008) et Shift (Vogel et Baudisch, 2007).

Lorsqu'un écran tactile multi-points est disponible, il est possible d'employer des techniques utilisant deux doigts (Benko *et al.*, 2006; Olwal *et al.*, 2008). Il est alors possible de positionner le curseur entre deux doigts ou d'appliquer de petits ajustements à l'aide de mouvements de la main non dominante.

2.3 Conclusion

Dans ce chapitre, nous avons abordé les différentes caractéristiques qui font des terminaux mobiles des appareils nécessitant une attention particulière. Il ne sera pas possible, dans beaucoup de cas, de se contenter d'adapter les applications et les techniques d'interaction développées pour une utilisation sur un ordinateur de bureau. De ces analyses, nous proposons d'énumérer quelques contraintes fortes des terminaux mobiles, qu'il sera important de prendre en compte lors du développement d'application et de l'étude de techniques d'interaction.

- La première contrainte des terminaux mobile concerne la TAILLE de l'appareil. Les dimensions de l'écran sont réduites et il convient donc de ne pas surcharger l'affichage avec des informations inutiles.

- La seconde contrainte intervient au niveau de l'ergonomie générale, et particulièrement de la PRÉHENSION de l'appareil : l'utilisateur doit tenir le terminal dans une de ses main, et l'interaction se fera alors principalement à l'aide d'un ou de deux doigts. Il ne sera ainsi pas possible d'utiliser plus de deux doigts simultanément. Lorsque l'on utilise les touches de l'appareil, il ne sera donc pas possible d'utiliser plus deux touches simultanément.

- La troisième contrainte est spécifique aux terminaux possédant un écran tactile. Dans ce cas, lors d'une utilisation à l'aide d'un doigt, la PRÉCISION de la sélection sera faible à cause de la taille du doigt.

- Enfin, la PUISSANCE de calcul des terminaux mobiles est souvent limitée. Il sera donc important de concevoir des techniques peu gourmandes en calcul, afin de laisser le plus de puissance disponible à l'application.

Le chapitre suivant est consacré à la présentation du domaine de l'interaction 3D. Il comporte une brève introduction aux notions essentielles de géométrie tridimensionnelle et présente les différentes caractéristiques de l'interaction 3D.

CHAPITRE 3

Interaction 3D

Il y a plusieurs définitions possibles de l'interaction 3D. Elle peut par exemple être définie par une "interaction Homme-machine dans laquelle les tâches de l'utilisateur sont réalisées dans un contexte spatial en 3D" (Bowman *et al.*, 2004).

Le terme "interaction" sous entend une action entre deux acteurs : dans notre cas, l'utilisateur et l'environnement 3D sont les acteurs impliqués dans l'interaction. Cette notion sous-entend également une action *bidirectionnelle* : l'utilisateur agit sur l'environnement 3D, en déplaçant son doigt sur l'écran tactile par exemple. En réponse, le retour (principalement visuel, mais aussi sonore, haptique, etc.) correspondant à son action montre les changements effectués dans l'environnement 3D.

Il est important de garder à l'esprit cette notion d'*inter-action*, et de considérer le retour visuel comme faisant partie de la tâche d'interaction.

Il est généralement admis que l'interaction 3D regroupe trois catégories de tâches : la tâche de *manipulation*, la *navigation* et le *contrôle de l'application* (Bowman *et al.*, 2001b). Ces trois tâches sont essentielles au pilotage d'une application 3D. Nous avons abordé le contrôle d'application dans le chapitre précédent, et nous nous concentrerons à présent les tâches propres à l'interaction 3D que sont la *manipulation* et la *navigation* car elles constituent le thème central de cette thèse.

3.1 Rappels mathématiques et géométriques

3.1.1 Repères

Afin de positionner un point 3D dans l'espace, il est nécessaire de définir un système de coordonnées. Le plus couramment employé est le système de coordonnées *cartésiennes*, où chaque point de l'espace est repéré par rapport à un *repère* défini par une origine O et trois vecteurs non coplanaires $\overrightarrow{x}, \overrightarrow{y}, \overrightarrow{z}$ (la plupart du temps, ces vecteurs seront orthonormés). Dans les applications 3D, quatre repères sont principalement utilisés, illustrés sur la Figure 3.1 :

- **Le repère monde** Il permet de définir l'origine de l'environnement 3D ainsi que son orientation globale. Généralement, l'axe Y est dirigé vers le haut et le plan formé par les axes X et Z est alors utilisé pour représenter le sol de l'environnement 3D.

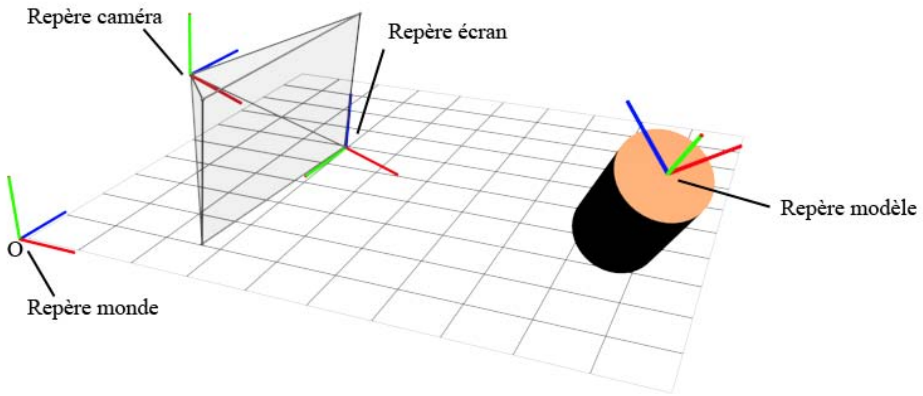


FIGURE 3.1 – Différents repères couramment utilisés.

- **Le repère caméra** Ici, le point de vue courant est considéré comme l'origine du repère, et possède donc les coordonnées $(0, 0, 0)$. Généralement, l'axe Z est utilisé pour désigner la direction de la vue ; on parle alors d'*axe des profondeurs*.
- **Le repère modèle** Chaque modèle de la scène 3D possède son propre repère : la position du point de pivot de l'objet définit l'origine du repère, et l'orientation du modèle sert à définir la base.
- **Le repère écran** Il permet de connaître la position d'un pixel à l'écran, ainsi que sa profondeur. Cette donnée correspond à la valeur du pixel dans le *tampon de profondeur*, aussi appelé *ZBuffer*. La phase consistant à passer d'un repère 3D au repère écran s'appelle la *rastérisation* ou *projection*. Il est également possible, à partir des coordonnées écran d'un pixel, de retrouver la position 3D du point correspondant.

3.1.2 Transformations

Le placement d'un modèle 3D dans l'espace est effectué en spécifiant les coordonnées de son *point de pivot*. Ce point correspond généralement au centre de gravité de l'objet, mais peut correspondre à n'importe quel point de l'espace 3D, et n'appartient pas forcément au modèle. Ce point sera ensuite utilisé pour calculer les transformations à appliquer au modèle 3D.

3.1.2.1 Translation

Une fois la position d'un modèle 3D déterminée, la transformation le plus élémentaire à appliquer est la **translation**. Cette opération permet de déplacer un point 3D. La distance et la direction de la translation sont représentées par un *vecteur*.

3.1.2.2 Rotation

Une seconde transformation 3D est essentielle : la **rotation**. Elle permet d'orienter un modèle dans l'espace. Une rotation peut être effectuée de deux manières : autour d'un *axe de rotation* ou autour d'un *point de pivot* (on parle alors de *rotation centrale*). L'axe de la rotation est déterminé par un point appartenant à cet axe ainsi qu'un vecteur donnant la direction de cet axe. La rotation est alors déterminée par l'angle de rotation à appliquer autour de cet axe.

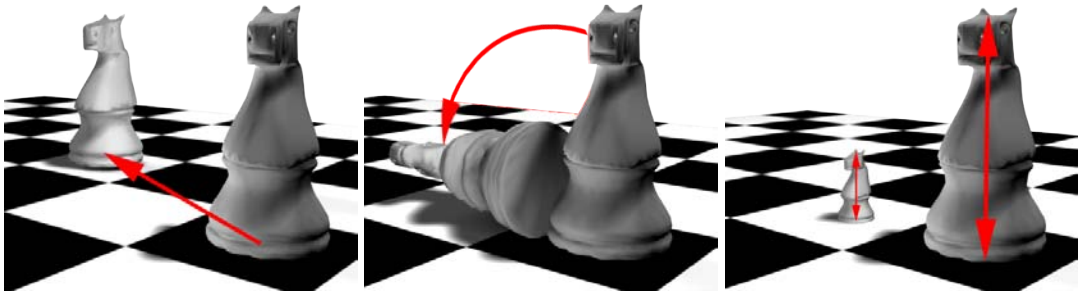


FIGURE 3.2 – Translation, rotation, homothétie : les trois transformations élémentaires.

Lors d'une rotation centrale, il y a plusieurs manières de noter la transformation à appliquer. Il est courant d'utiliser les *angles d'Euler*, qui sont les angles utilisés dans le repère sphérique décrit auparavant. Il est également possible de représenter une rotation en utilisant la notion de *quaternion* (Bobick, 1998). Les quaternions fournissent une notation mathématique utile pour la représentation de l'orientation et des rotations d'un modèle 3D. L'un des avantages de cette représentation est qu'il est facile de composer les rotations, et de retrouver les angles d'Euler ou la matrice de transformation associées.

3.1.2.3 Homothétie

Enfin, la *taille* d'un modèle 3D est également une caractéristique essentielle. Il est ici aussi possible d'utiliser un triplet pour représenter la taille, chaque coordonnée du triplet déterminant la taille du modèle selon l'axe correspondant. La transformation associée au changement de taille d'un modèle se nomme l'*homothétie*, *mise à l'échelle* ou *redimensionnement*. Celle-ci peut être appliquée selon les trois axes : on parle alors d'*homothétie centrale*. Si la transformation est appliquée sur deux axes seulement, l'homothétie est dire *planaire*. Enfin, une homothétie n'affectant qu'un axe est dite *axiale*.

La Figure 3.2 montre des exemples de translation, rotation et homothétie.

3.1.2.4 Degrés de liberté

La manipulation des modèles 3D correspond à l'application d'une ou plusieurs transformations simultanément au modèle. Ainsi, si un modèle est totalement libre d'être déplacé dans toutes les directions et pivoté selon tous les axes, on dira qu'il possède **6 degrés de liberté** (DDL). On appelle *degré de liberté* les mouvements **indépendants** qu'un objet peut réaliser. Ainsi, les six DDL élémentaires sont les translations selon les axes X, Y et Z, ainsi que les rotations autour des mêmes axes. Avec ces 6 transformations élémentaires, il est possible de placer un objet n'importe où dans l'espace 3D, et avec n'importe quelle orientation.

Le terme de degré de liberté est normalement utilisé en mécanique, pour désigner les mouvements possibles dans une liaison. La définition originale ne prend donc en compte que les translations et les rotations. Cependant, nous pouvons étendre cette notion à n'importe quelle transformation, comme l'homothétie. La manipulation d'un objet comprenant

translation, rotation et redimensionnement nécessitera donc 9 DDL (3 translations, 3 rotations, 3 redimensionnements).

3.2 Techniques d'interaction

3.2.1 Définitions

Il est possible, afin de manipuler un modèle 3D quelconque, de spécifier sa position, son orientation et sa taille de manière explicite, grâce à ses coordonnées et à la valeur de des angles par exemple. Ceci est possible dans beaucoup d'applications de modélisation, en saisissant les coordonnées et les valeurs des transformations dans une boîte de dialogue spécifique. Cependant, cette approche n'est pas intuitive et est plutôt utilisée par les experts, pour qui les notions de transformation 3D, de repère et de degrés de liberté sont parfaitement connues. Pour un novice en revanche, il convient de développer des méthodes lui permettant d'effectuer sa tâche de manière plus intuitive : on parle alors de *technique d'interaction*. Une technique d'interaction décrit la *manière de se servir d'un dispositif d'entrée pour accomplir une tâche sur l'ordinateur* (Foley *et al.*, 1990). Dans le domaine de l'interaction 3D, les techniques d'interaction seront donc des méthodes permettant, à l'aide des dispositifs d'entrée, d'interagir avec l'environnement 3D.

Si la technique d'interaction fait une analogie avec un geste, une méthode ou une technique utilisée dans la vie réelle ou dans un monde imaginaire, on parlera alors de *métaphore* (*métaphore magique* dans le cas d'une analogie avec un monde imaginaire). Nous verrons des exemples de métaphores dans les sections suivantes, consacrées à la description des différentes tâches d'interaction 3D.

3.2.2 Classification

Devant le grand nombre de techniques d'interaction existantes, il a été nécessaire d'établir des classifications en fonction de différents critères. Ces classifications portent le nom de **taxonomies** et sont structurées de manière évolutive. Beaucoup de taxonomies ont été proposée, en fonction du type de tâche à accomplir(Grossman et Wigdor, 2007; Bowman *et al.*, 1997), des périphériques d'interaction disponibles ou du type de métaphore recherché.

Dans ce mémoire, nous avons choisi de classifier les techniques en fonction du degré de la directivité du contrôle proposé à l'utilisateur : dans le cas du **contrôle direct**, l'interaction est réalisée de manière directe et en temps réel. Ainsi, le résultat d'une action ou d'un mouvement réalisé par l'utilisateur est visible immédiatement. Ceci est opposé au **contrôle planifié**, dans lequel le contrôle se fait de manière asynchrone : l'utilisateur donne un *ordre* à l'application, par exemple désigner un point vers lequel se déplacer ou esquisser l'orientation d'un modèle, qui va calculer les transformations nécessaires de manière à obtenir le résultat demandé par l'utilisateur.

3.3 Manipulation

La manipulation de modèles 3D est une tâche essentielle à la conception et à l'interaction avec des scènes 3D. Il est primordial de proposer à l'utilisateur des techniques de manipu-

lation efficaces, auquel cas il risque de ne pas pouvoir effectuer certaines tâches. La tâche de manipulation regroupe en fait trois sous-tâches : le *pointage*, la *sélection* et la *transformation* (déplacement et déformation) de modèles 3D.

3.3.1 Pointage / Sélection

Les tâches de pointage et de sélection sont souvent liées. Le pointage consiste à désigner une partie de la scène 3D, par exemple un point ou un modèle 3D. La sélection est destinée à valider le choix effectué par le pointage.

Cette tâche est donc essentielle car elle permet de définir le contexte de travail dans lequel les transformations seront appliquées. Il est important de bien mettre en valeur les objets sélectionnés en fournissant un *retour visuel*, par exemple un changement de couleur ou une surbrillance des modèles 3D sélectionnés.

La sélection peut se faire par un *pointage direct*, c'est à dire en désignant la cible à l'aide du dispositif de pointage (curseur, écran tactile) ou par une sélection *indirecte*, en passant par un menu par exemple.

3.3.2 Transformation

Cette tâche consiste à appliquer une transformation géométrique au modèle 3D. Cette dernière peut être une translation, une rotation, ou n'importe quelle modification (déformation, redimensionnement, etc.).

Dans le monde réel, nous manipulons les objets grâce à notre main : il est ainsi possible de les déplacer et de les tourner dans toutes les directions. Une telle combinaison de translations et de rotations implique une manipulation simultanée de 6 DDL. Malheureusement, la plupart des périphériques d'entrée, tels que les souris ou les écrans tactiles, ne fournissent que 2 DDL. Il est donc important de développer des techniques permettant de manipuler ces 6 DDL à l'aide de 2 DDL.

La technique la plus classiquement implémentée dans les logiciels de modélisation 3D est de décomposer la tâche de manipulation en fonction des degrés de liberté invoqués. Ainsi, afin de placer un objet dans une scène 3D, il faudra par exemple l'orienter à l'aide d'un premier outil de rotation, puis le déplacer vers sa position finale grâce à l'outil "translation". Cette décomposition de la tâche en sous-tâches n'est cependant pas la solution optimale, car continuité du mouvement est interrompue, ce qui mène à une charge cognitive plus importante (Buxton, 1986). Nous verrons dans ce mémoire différentes approches développées au cours des dernières années consistant à intégrer intelligemment les rotations et les translations en un même outil.

3.4 Navigation

La navigation est une tâche fondamentale de l'interaction 3D. Elle permet à l'utilisateur de se déplacer au sein de l'environnement 3D, d'observer les modèles et d'apprendre à connaître le monde virtuel qui l'entoure. L'expérience des utilisateurs sera en grande partie déterminée par la qualité des techniques de navigation développées.

La notion de *navigation* est large, et peut être aussi bien étudiée dans le monde réel que dans un monde virtuel. Ainsi, on distingue généralement deux composantes dans la tâche de navigation : la composante *motrice*, qui correspond au déplacement effectif d'un point à un autre, et la composante *cognitive*, qui, elle, est invoquée lors de phases de recherche d'itinéraire et permet à l'utilisateur de se repérer, d'appréhender l'environnement qui l'entoure et de trouver la trajectoire à suivre pour atteindre un point précis.

3.4.1 Naviguer, trouver son chemin, se repérer

La navigation ne correspond donc pas seulement au déplacement de la vue, mais également à tous les processus cognitifs qui entrent en jeu lors de ce déplacement. La recherche d'itinéraire est définie comme étant "le procédé cognitif consistant à définir un chemin à travers un environnement, en utilisant et en acquérant des connaissances spatiales, en étant aidé par des aides (artificielles ou non)" (Bowman *et al.*, 2001b).

Ainsi, la connaissance d'un lieu qui nous est préalablement inconnu passe par plusieurs phases d'apprentissage, qui sont la *connaissance de points de repères*, la *connaissance procédurale* et la *connaissance topologique* (Thorndyke et Hayes-Roth, 1982). Ceci permet de construire notre *carte cognitive*, qui correspond à la représentation mentale que nous faisons de notre environnement.

Il existe plusieurs types de tâches dans la recherche d'itinéraire (Bowman *et al.*, 2001b) :

- **L'exploration** implique un déplacement libre à travers l'environnement. L'utilisateur n'a pas de but particulier. L'exploration aide à la construction de la carte cognitive.
- **La recherche**, qui peut être divisée en *recherche naïve* (*naive search* en anglais), où l'utilisateur ne connaît pas la position exacte de sa cible, et la *recherche préparée* (*primed search*), où l'utilisateur connaît la position de sa cible. Dans le premier cas, la carte cognitive de l'utilisateur n'est pas assez complète pour l'emmener jusqu'au point souhaité du premier coup. Dans le deuxième cas, la capacité de l'utilisateur à se servir de sa carte cognitive pour faire la relation entre sa position courante et la position de sa cible sera déterminante dans le succès de la recherche de la cible.
- **La manoeuvre**, où l'utilisateur fait de petits mouvements, afin d'atteindre un point bien précis. Ceci peut être une sous-tâche de la recherche. Elle apparaît par exemple lorsque l'on souhaite identifier clairement un point de repère, ou pour apercevoir de petites cibles.
- **Le mouvement prédéfini**. Dans cette tâche, l'utilisateur est guidé automatiquement à travers l'environnement, le long d'un chemin prédéfini afin d'obtenir un aperçu global de l'environnement. Cette tâche permet à l'utilisateur de construire une carte cognitive basique en peu de temps, à condition que la trajectoire définie soit efficace.

Une technique de navigation sera alors efficace si elle aide l'utilisateur à construire sa carte cognitive. Ainsi, il n'est pas recommandé d'utiliser de la téléportation, car la désorientation serait alors importante. De même, il peut être judicieux de concevoir des techniques différentes en fonction du type de tâche (exploration, recherche, manoeuvre, mouvement prédéfini), car les degrés de liberté invoqués peuvent ne pas être les mêmes.

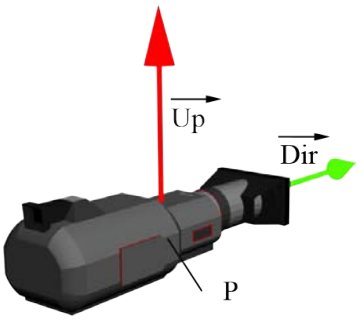


FIGURE 3.3 – Représentation d'une caméra 3D.

3.4.2 Navigation 3D

La navigation dans une scène 3D représente certainement une part importante de l'interaction 3D. Elle est invoquée en permanence dans tous les types d'application 3D, que ce soit dans la modélisation afin d'observer les différentes parties d'un modèle, la RV immersive afin de se déplacer dans l'environnement 3D ou dans les jeux vidéo afin de suivre le personnage virtuel.

Dans beaucoup d'applications 3D, on parlera de *caméra 3D* afin de désigner la position et l'orientation de la vue courante. Naviguer dans la scène correspond donc à manipuler la caméra 3D, la déplacer et la pivoter. Techniquement parlant, il est courant de représenter une caméra 3D à l'aide de trois composantes : une *position* dans l'espace P, représentée par les trois coordonnées (x, y, z), ainsi que deux *vecteurs* : un vecteur donne la *direction de vue* $\overrightarrow{Dir}$ et un autre, appelé *vecteur up* ou *vecteur haut*, noté $\overrightarrow{Up}$, permet de désigner la verticalité (Figure 3.3).

Les déplacements au sein de la scène 3D peuvent être effectués de manière *égocentrique*, c'est à dire en considérant que le point de vue appartient à la scène, ou de manière *exocentrique*, en considérant la scène 3D dans son intégralité et en gardant une vision globale de celle-ci. Lors de déplacements égocentriques, les mouvements sont calculés et réalisés par rapport au repère de la caméra, tandis que les déplacements exocentriques font référence à un repère extérieur au point de vue.

Au cours de ces dernières années, beaucoup de techniques de navigation ont été développées. Cependant, quelques mouvements essentiels représentent un facteur commun à beaucoup d'application de visualisation 3D. Elles ont pour certaines été créées pour une utilisation à l'aide de capteurs à 6 DDL (Ware et Osborne, 1990), mais ont été adaptées à une utilisation à l'aide de 2 DDL. Nous utiliserons la terminologie anglaise afin de nommer ces techniques, car elle est couramment utilisée dans la littérature et dans les applications.

- **Panning** Ce mouvement, qui vient du terme "panoramique" utilisé dans le cinéma, correspond à une translation de la caméra selon ses axes X et Y. Les mouvements sont donc parallèles au plan de l'écran.
- **Look Around** Ce mouvement égocentrique ne déplace pas le point de vue, mais se contente de changer l'orientation de la vue. Il permet à l'utilisateur d'observer l'environnement qui l'entoure.
- **Trackball** Ce mouvement est plus complexe. Le principe ici est de faire tourner la scène 3D entièrement, de manière à l'observer comme si elle était placée à l'intérieur d'une

sphère virtuelle. Cette technique a connu beaucoup d'implémentations différentes, que nous étudierons par la suite. Elle est essentielle à l'observation de modèles 3D de manière globale.

– **Flying** Ce mode de navigation permet de se déplacer sur un sol virtuel, souvent représenté par le plan XZ. Il est utilisé dans les jeux vidéos du type "First Person Shooter" et représente la technique de base du déplacement 3D. Cette technique permet d'avancer / reculer ainsi que de tourner. Il est alors possible de se déplacer librement dans l'environnement 3D en restant sur un sol virtuel. Cette technique possède l'avantage de pouvoir être utilisée à l'aide de beaucoup de périphériques, car elle ne nécessite que 2 DDL.

Ces quatre contrôles de caméra représentent les contrôles standards, auxquels nous ferons souvent référence lors de ce mémoire. Ils sont présents dans la plupart des applications 3D.

CHAPITRE 4

Bilan

Cette première partie du mémoire a été consacrée à l'introduction des différents domaines de cette thèse que sont les terminaux mobiles, l'ergonomie mobile et l'interaction 3D. A partie des analyses effectuées, nous proposons quelques **règles** concernant le développement de techniques d'interaction 3D destinées à une utilisation sur terminaux mobiles.

COLOCALISATION La colocalisation est très présente lorsque l'on utilise un écran tactile. Ceci permet une interaction plus directe, mais risque d'occulter une partie de l'écran. Il sera donc important de prendre en compte la colocalisation lors de l'étude d'une technique d'interaction.

UTILISATION UNIMANUELLE Lors d'un usage en mobilité, par exemple en marchant, les utilisateurs préfèrent généralement se servir de leur terminal à l'aide d'une seule main. Il est donc préférable de pouvoir utiliser la technique à une seule main.

PRÉCISION La sélection d'un point précis peut être difficile, particulièrement lorsque l'on utilise un doigt pour interagir. Il faudra donc veiller à adapter la taille des éléments afin de ne pas demander une trop grande précision.

ÉCONOMIE Les ressources matérielles des terminaux mobiles étant limitées, il est important de développer des techniques d'interaction qui ne nécessitent pas de calculs trop importants, afin de laisser le maximum de puissance de calcul pour l'application principale.

ORIENTATION Il peut être difficile, lorsque l'on est en extérieur par exemple, de trouver une orientation permettant une bonne visualisation de l'écran. La prise en compte du déplacement de l'appareil pour interagir devra donc être utilisée avec parcimonie.

AFFICHAGE La taille de l'écran étant réduite, il est important de ne pas surcharger l'affichage et de ne pas occulter une trop grande partie de l'écran.

Les analyses de techniques d'interaction 3D présentées dans la suite de ce mémoire feront référence à ces principes, afin de comprendre pourquoi une technique est, ou non, adaptée à une utilisation sur terminaux mobiles.

Deuxième partie

Interaction 3D Avec Contrôle Direct

Résumé

La puissance grandissante des terminaux mobiles permet aujourd'hui d'y afficher une scène 3D en temps réel. Il est alors possible pour l'utilisateur d'interagir de manière directe avec l'environnement 3D. Dans ce cas, les actions de l'utilisateur, appui sur une touche, déplacement de l'appareil ou mouvement du doigt sur l'écran tactile par exemple, modifient en temps réel l'environnement 3D en déplaçant la caméra ou en transformant les modèles 3D.

Dans cette partie, nous nous intéresserons à différentes techniques d'interaction 3D implémentées dans les applications 3D ou décrites dans la littérature qui utilisent un contrôle direct. Nous étudierons en particulier leur possible adaptation aux contraintes des terminaux mobiles. Nous présenterons ensuite une technique d'observation de modèles 3D que nous avons développée appelée ScrutiCam. Puis, nous décrirons un ensemble de techniques de navigation 3D libre utilisables sur un écran tactile multi-points. Enfin, nous terminerons cette partie par une étude que nous avons menée afin d'étudier l'influence de l'utilisation de deux doigts pour des tâches de positionnement et de contrôle de vitesse.

CHAPITRE 5

Introduction et analyse

La puissance grandissante des terminaux mobiles permet aujourd'hui d'obtenir un rendu en temps réel de scènes 3D sur de tels appareils. Il est ainsi possible de contrôler une application 3D de manière interactive et, surtout, de manière directe : lorsque l'utilisateur réalise une action, son résultat est immédiatement visible. Il est ainsi possible d'avoir un *contrôle direct* sur l'environnement 3D.

De nombreux travaux ont été consacrés au contrôle direct d'applications 3D, dans différents contextes d'application. Une partie des travaux a été développée pour des applications *immersives*. De ce fait, elles utilisent des périphériques particuliers, comme des capteurs de position ou des périphériques offrant un grand nombre de degrés de liberté. Ces techniques, bien que reconnues comme efficaces dans leurs configurations d'origine, risquent de ne pas être adaptables aux terminaux mobiles.

Il y a également eu de nombreuses recherches dans le domaine de l'interaction 3D pour les *ordinateurs de bureau*. Ici, les périphériques utilisés sont beaucoup plus classiques et proches des périphériques présents sur les TM, et se limitent pour la plupart au duo clavier + souris. L'adaptation de ces techniques n'est pas triviale dans beaucoup de cas. En effet, certaines de ces techniques utilisent des caractéristiques spécifiques de l'ergonomie de bureau, comme l'utilisation simultanée de la souris et du clavier, qui rend leur adaptation aux TM difficile. En revanche, d'autres techniques pourront être adaptées facilement aux appareils mobiles équipés d'écrans tactiles.

Enfin, quelques techniques d'interaction 3D ont été développées spécifiquement pour les terminaux mobiles. Elles sont peu nombreuses, car les appareils mobiles ont plutôt été destinés à une utilisation de bureautique, mais sont les plus intéressantes dans notre contexte, car elles utilisent et considèrent les caractéristiques intrinsèques des terminaux mobiles.

Dans cette section, nous présenterons des techniques d'interaction 3D continues qui sont adaptables aux contraintes des TM, qu'elles aient été développées spécifiquement pour des TM ou non.

Afin de présenter ces différentes techniques d'interaction 3D, nous les classons en fonction du périphérique qui sera utilisé. Il y a dans ce cas deux grandes familles de périphériques : les *touches* de l'appareil, qui fournissent un signal binaire et l'*écran tactile*, qui est présent sur une grande variété de terminaux mobiles.

Cependant, les terminaux actuels sont parfois équipés d'autres périphériques qui apportent un réel avantage pour l'interaction 3D. Nous étudierons donc l'utilisation d'*écran*

tactile multi-points, qui permet de détecter la présence et la position de plusieurs points de contact, ainsi que l'utilisation d'autres périphériques comme la *caméra vidéo*, qui permet de filmer en temps réel l'environnement de l'utilisateur, ou différents capteurs dont peuvent être équipés les terminaux mobiles actuels.

Pour chacun de ces périphériques, nous présenterons et étudierons les principales techniques de manipulation et de navigation qui utilisent de tels périphériques. Nous verrons de même les contraintes inhérentes à l'adaptation de ces techniques aux terminaux mobiles.

5.1 Interagir avec une scène 3D en utilisant des touches

Comme nous l'avons vu dans la partie introductive, la présence d'un clavier est un facteur commun à la très grande majorité des terminaux mobiles ; certains ne possèdent même que ce périphérique comme moyen d'interaction. Il est donc important d'étudier comment interagir avec un environnement 3D à l'aide du clavier de l'appareil.

5.1.1 Naviguer

L'utilisation des touches pour se déplacer dans un environnement 3D est une technique classique, qui est couramment utilisée dans les jeux vidéo. Elle utilise la métaphore du *flying vehicle*, présentée dans le chapitre 3. Dans ce mode de déplacement, les touches *haut* et *bas* contrôlent l'avancement de la caméra, et les touches *gauche* et *droite* permettent de tourner la caméra autour de son axe y. Lorsque l'utilisateur relâche la touche, le mouvement s'arrête.

Le principal problème qui intervient ici est que la contrainte PRÉHENSION empêche l'utilisateur d'avancer et de tourner simultanément. Cela produit une navigation discontinue, car il sera nécessaire de s'arrêter lorsque l'on veut changer l'orientation de la caméra.

Dans ce genre d'approche, il convient de régler convenablement la distance de déplacement que produira l'appui sur une touche. Mackinlay *et al.* (1990) isolent ainsi trois cas qui peuvent potentiellement poser problème. Si le pas est trop grand, il sera difficile de se placer précisément, et l'utilisateur devra réaliser plusieurs tentatives pour arriver à un point précis (stratégie *fast&overshoot*). Si, au contraire, le pas est trop petit, les déplacements seront lents, mais très précis (stratégie *slow&sure*). L'utilisateur est alors assuré d'arriver jusqu'à son point de destination, mais au prix d'un long déplacement. Enfin, il est possible de faire varier la vitesse en fonction du temps de déplacement (stratégie *pulse*). Ainsi, plusieurs appuis seront nécessaires pour avoir un mouvement précis, mais il sera tout à fait possible de se déplacer rapidement ou, au contraire, précisément. Cette dernière méthode est la plus pratique, car elle permet de combiner un déplacement précis et un déplacement rapide. La Figure 5.1, où chaque pas de déplacement est représenté par un vecteur dont la norme dépend de la distance parcourue, schématise ces trois stratégies.

Il est également possible de contrôler la vitesse d'avancement par le nombre d'appui sur une touche : une touche sert alors à accélérer, une autre sert à décélérer. Dans ce cas, lorsque l'utilisateur relâche une touche, le déplacement continue, à vitesse constante. Il est ainsi possible d'orienter la caméra pendant le déplacement. Cependant, arrêter le mouvement peut être compliqué, car il sera nécessaire de décélérer, par plusieurs appuis successifs, jusqu'à trouver une vitesse nulle.

L'ajout d'un capteur de pression sous chaque touche de l'appareil permettrait d'avoir un contrôle plus précis (Clarkson *et al.*, 2006). Malheureusement, ce type de dispositif n'est pas

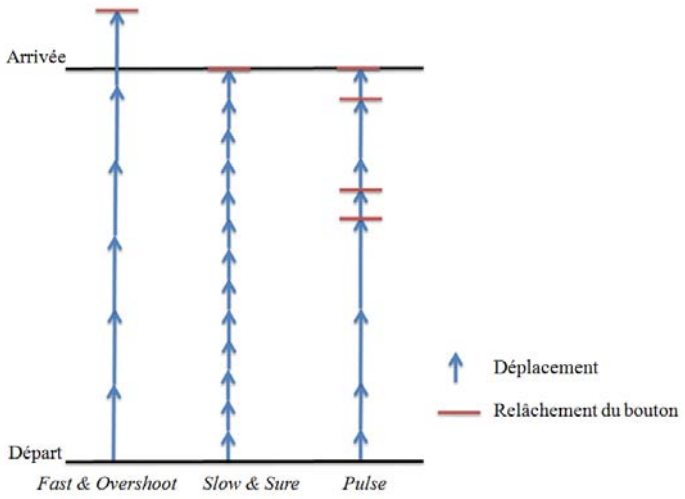


FIGURE 5.1 – Différentes stratégies de contrôle du pas de déplacement.

présent sur les terminaux actuels.

Une fois la vitesse de déplacement correctement définie, il est tout à fait possible d'utiliser des techniques d'interaction plus avancées, afin d'avoir un meilleur contrôle. Dans le cas de la navigation 3D, ces techniques sont bien souvent dérivées de la technique du *flying*. Il est par exemple possible de contraindre la caméra à rester le long de surfaces prédéfinies (Hanson et Wernert, 1997). Ceci permet de rester à une certaine distance du sol, ou de n'importe quelle surface visible ou non définie par l'utilisateur. Cette approche est intéressante lorsque l'on souhaite interdire à l'utilisateur d'aller vers certaines parties d'une scène.

Afin de favoriser la construction de la carte cognitive de l'utilisateur lors de son déplacement, Tan *et al.* (2001) proposent d'élever la caméra 3D dans les airs lors de déplacements rapides. Ceci permet à l'utilisateur de mieux percevoir son environnement.

5.1.2 Manipuler

Le principal problème que pose la manipulation de modèles 3D est que plusieurs DDL sont impliqués. Ainsi, il est nécessaire de contrôler au moins 6 DDL si l'on souhaite déplacer et pivoter un modèle, et 9 DDL si la tâche concerne également un redimensionnement. Dans tous les cas, plusieurs stratégies sont possibles.

Il est tout d'abord possible d'assigner chaque degré de liberté à contrôler à un couple de touches particulier : par exemple, si deux touches sont assignées à la translation selon l'axe X, la première touche sert à **augmenter** la valeur de la coordonnées X du modèle, la seconde sert à **diminuer** cette valeur. Il est néanmoins nécessaire de disposer de beaucoup de touches, ce qui n'est pas le cas sur la plupart des terminaux mobiles, qui ne possèdent que quelques touches. De plus, cela demande un effort de mémorisation à l'utilisateur.

La seconde stratégie permet de résoudre ce problème. L'idée ici est de ne contrôler que quelques DDL simultanément, à l'aide de quelques touches. la plupart des terminaux étant équipés de touches directionnelles, il sera possible de contrôler deux DDL à l'aide de ces

touches. Dans cette situation, il est nécessaire de proposer une méthode permettant de choisir le type de transformation à appliquer (translation, rotation, redimensionnement) ainsi que l'axe ou le plan sur lequel appliquer cette transformation. Ceci peut être fait à l'aide d'un menu ou de raccourcis clavier par exemple.

Une fois la transformation sélectionnée, le contrôle de la transformation subit les mêmes contraintes que celles de la navigation, à savoir le réglage de la vitesse de variation du paramètre à contrôler et l'impossibilité d'utiliser plusieurs touches. Cela implique donc de décomposer la tâche en plusieurs sous-tâches élémentaires.

Dans tous les cas, il sera important d'étudier la disposition des touches au préalable, afin de proposer une interaction qui soit adaptée : Jacob et Sibert (1992) soulignent en effet qu'il est important d'adapter la structure de l'interface à la structure de la tâche.

Beaucoup de terminaux actuels sont pourvus d'un écran tactile, qui donne à l'utilisateur une plus grande interactivité et un meilleur contrôle de l'application 3D. De ce fait, de nombreuses techniques d'interaction 3D peuvent tirer parti de ce dispositif.

5.2 Interagir avec une scène 3D en utilisant l'écran tactile

La présence d'un écran tactile sur un terminal mobile représente en réel avantage par rapport aux touches. Cela permet de rapprocher son ergonomie de celle des ordinateurs de bureau, car l'écran tactile fournit alors la possibilité de manipuler un curseur de façon continue. Cependant, comme nous l'avons vu dans la partie introductive, il existe un certain nombre de différences significatives entre l'utilisation d'un écran tactile et celle d'une souris, qui peuvent compliquer l'adaptation des techniques initialement développées pour une utilisation à la souris.

5.2.1 Naviguer

Concernant la navigation, il y a généralement deux tâches distinctes : l'exploration, qui permet de se déplacer au sein d'un environnement 3D, et l'observation de modèles 3D.

5.2.1.1 Exploration

La technique d'exploration de référence est la technique du *flying vehicle* définie par Ware et Osborne (1990). Elle est souvent associée à la technique appelée *Eye in Hand*, qui permet de manipuler la caméra virtuelle afin de regarder autour de soi. Ces techniques ont été développées pour une utilisation à l'aide de capteurs à 6 DDL. Cependant, de nombreuses adaptations de ces techniques ont été réalisées afin de permettre leur utilisation à l'aide de périphériques à 2 DDL.

L'adaptation de la technique *Eye in Hand* ne pose pas de problème particulier. En effet, regarder autour de soi ne nécessite que 2 DDL (rotation autour des axes X et Y de la caméra), qui peuvent être assignés aux mouvements de la souris. Cette technique est parfois appelée "*Look Around*" dans les applications 3D.

FIGURE 5.2 – Prototypes d'ajout de contrôle élastique à un TM (Hachet et Kulik, 2008).

L'adaptation du *Flying Vehicle* nécessite, elle, une étude plus approfondie. Il est bien entendu possible d'utiliser les techniques de *flying*, décrites précédemment (Hanson et Wernert, 1997; Tan *et al.*, 2001). Dans ce cas, les mouvements verticaux de la souris contrôlent la vitesse de déplacement, tandis que les mouvements horizontaux permettent de tourner la caméra vers la gauche ou la droite. Ceci permet d'avoir des mouvements de caméra plus fluides que ne le permet l'utilisation des touches, car l'utilisateur peut régler précisément la vitesse de son déplacement et contrôler simultanément l'avancement et la rotation de la vue.

Cependant, il a été montré par Zhai (1995) que l'utilisation d'un périphérique isotonique n'est pas adaptée aux tâches de contrôle de vitesse. Pour ce type de tâches, les périphériques élastiques ou isométriques sont plus performants. Cette différence est principalement due à la difficulté, avec des périphériques isotoniques, à retrouver la *position de référence*, c'est à dire la position initiale, à laquelle la vitesse est nulle.

Ainsi, d'après Hachet et Kulik (2008), l'ajout d'un contrôle élastique aux écrans tactiles permet d'obtenir de meilleures performances pour des tâches de navigation 3D. Ils ont proposé plusieurs prototypes, visibles sur la Figure 5.2, utilisables au stylet ou au doigt.

Afin d'avoir une vitesse de déplacement adaptée à l'environnement, Ware et Fleet (1997) ont défini le concept de navigation "sensible au contexte". Le principe ici est d'utiliser l'information de profondeur contenue dans le *tampon de profondeur* afin de définir la vitesse du déplacement.

5.2.1.2 Observation

Concernant l'observation de modèles 3D, la technique de référence est le *Scene in Hand* de Ware et Osborne (1990). Cette technique est essentielle, car elle permet à l'utilisateur de manipuler un modèle 3D, de le tourner dans toutes les directions et de l'appréhender dans son intégralité. Il est à noter qu'il est également possible d'obtenir un tel contrôle en maintenant la caméra fixe, mais en appliquant des rotations à la scène 3D.

Dès la fin des années 1980, Chen *et al.* (1988) a étudié différentes méthodes permettant de contrôler des rotations 3D à l'aide de contrôleurs 2D. De cette étude est apparue la *Virtual Sphere*, qui permet de contrôler les rotations d'un modèle 3D comme s'il était à l'intérieur d'une sphère virtuelle. Ceci permet ainsi de déplacer la caméra autour du modèle, car les mouvements de la caméra sont contraints à la surface d'une sphère virtuelle englobant l'objet manipulé. Cette technique est aussi connue sous le nom de *Virtual Trackball*.

Suite à cette étude, plusieurs améliorations de cette technique ont été réalisées. Une des plus connues est le *ArcBall* (Shoemake, 1992), qui possède l'avantage de maintenir une bonne

correspondance entre la position de la souris et la rotation de l'objet. Cette caractéristique est importante, car elle satisfait le principe COLOCALISATION.

Henriksen *et al.* (2004) ont étudié les fondements mathématiques des techniques précédemment citées, en plus de la technique de Bell (non publiée) développée pour la démonstration "flip" des capacités des stations Silicon Graphics. Ils ont montré les forces et les faiblesses de chacune, et ont proposé des améliorations à ces techniques.

Enfin, une nouvelle étude comparative des différentes techniques de rotation 3D a été menée plus récemment (Bade *et al.*, 2005), avec des critères basés sur les attentes des utilisateurs :

- **Principe 1 :** Des actions similaires doivent produire des résultats similaires.
- **Principe 2 :** La direction de la rotation doit correspondre à la direction du mouvement de la souris.
- **Principe 3 :** La rotation 3D doit être transitive.
- **Principe 4 :** Le rapport *Control To Display* (C/D) doit être paramétrable.

Il ressort de leurs expérimentations qu'aucune des techniques ne satisfait ces quatre principes. Cependant, la technique de Chen *et al.* (1988) (ici appelée "Two Axis Valuator") possède l'avantage d'être la seule technique à satisfaire le principe 1.

Il est à noter que d'autres études similaires ont été effectuées (Hinckley *et al.*, 1997; Jacob et Oliver, 1995; Partala, 1999), mais pour des tâches de rotation plus simples.

La technique du Trackball, et ses dérivées, possède des avantages indéniables, et permet d'avoir un aperçu rapide d'un modèle 3D. Elle possède cependant des problèmes d'utilisation importants.

Tout d'abord, cette technique possède le désavantage de nécessiter le choix d'un *point de pivot*, qui détermine le centre de la sphère virtuelle. Ce point par exemple peut être :

- L'origine du repère. Ce choix peut être mauvais dans le cas où la scène n'est pas centrée sur l'origine du repère.
- Le centre du modèle manipulé ou sélectionné ou le centre de la scène entière. Le problème ici est que ce point peut être loin de la position courante de la caméra.
- Un point sélectionné par l'utilisateur : il faut dans ce cas fournir à l'utilisateur le moyen de sélectionner un point 3D, ce qui peut être une tâche complexe.

Il sera donc important de choisir un point de pivot en fonction de l'application ou du type de modèle manipulé.

Un second problème important du Trackball est que cette technique n'est pas adaptée à des modèles possédant des proportions allongées, comme la statue visible sur la Figure 5.3. En effet, sur ce genre de modèles, la caméra sera tantôt proche de la surface du modèle (a), tantôt éloignée (b). Ceci est gênant car cela oblige l'utilisateur à se rapprocher manuellement de l'objet, à l'aide d'un autre outil. Cette technique n'est également pas adaptée à la visualisation de grands modèles comme des terrains pour les mêmes raisons.

Enfin, le Trackball n'est pas adapté à un mode de navigation libre, étant donné que la caméra est toujours dirigée vers le point de pivot. Il est aussi difficile de prévoir les mouvements de la caméra lorsque l'on est placé à l'intérieur d'un modèle 3D.

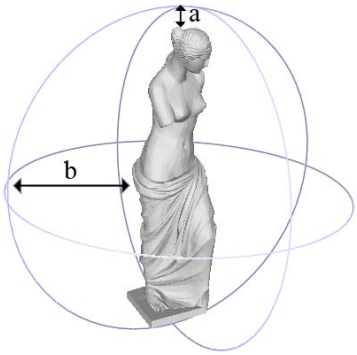


FIGURE 5.3 – Représentation d'un Trackball, utilisé dans l'application MeshLab. La distance entre la caméra virtuelle et le modèle n'est pas constante.

Ces problèmes sont abordés dans l'article "Safe 3D Navigation" (Fitzmaurice *et al.*, 2008). Les auteurs expliquent les problèmes que peuvent créer l'utilisation des outils standards, comme le pan, le zoom, le trackball, et proposent un concept de "Navigation 3D Sure", ainsi que des outils, sous la forme d'un widget, qui prennent en compte ces conseils.

Afin de pallier à ces problèmes, des techniques d'inspection de modèles 3D ont été développées. StyleCam (Burtnyk *et al.*, 2002), par exemple, utilise des surfaces virtuelles, définies par le graphiste, afin de proposer à l'utilisateur des mouvements de caméra adaptés au modèle qu'il visualise (voir Figure 5.4). De même, HoverCam (Khan *et al.*, 2005) est une caméra virtuelle particulièrement adaptée à l'inspection de modèles. Grâce à un algorithme de recherche du point le plus proche, cette technique aligne la caméra avec la surface du modèle, et permet à l'utilisateur de se déplacer le long de celui-ci, en n'utilisant qu'un périphérique 2 DDL. Cette technique sera détaillée et étudiée plus en détails dans le chapitre 6.

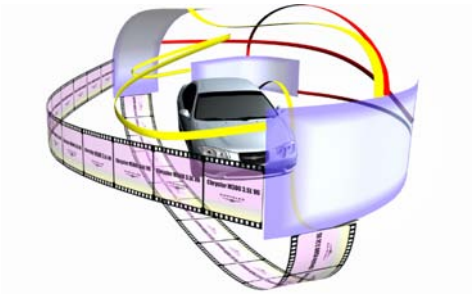


FIGURE 5.4 – StylCam(Burtnyk *et al.*, 2002) permet de se déplacer le long de surfaces virtuelles.

5.2.2 Manipuler

Une technique couramment employée pour manipuler des modèles 3D est d'utiliser des *widgets*. L'utilisation de tels contrôles pour manipuler des modèles 3D a été introduite dès les années 1980, avec le concept des "Skitters" et "Jacks" de Bier (1986). Les "Skitters" représentent un curseur 3D, que l'utilisateur déplace dans la scène. Ces curseurs définissent

une *position 3D* ainsi qu'une *orientation*. Les "Jacks" sont des repères que l'utilisateur place au sein de la scène 3D, afin de mettre en relation différentes parties d'un objet à assembler.

Bier (1990) a également montré, quelques années plus tard, que l'utilisation de "magnétisme" (*snapping*), qui attire le curseur de l'utilisateur vers des points particuliers (sommets des modèles, "Jacks" définis par l'utilisateur), améliorait les performances lors de tâches d'assemblage.

Dans le même contexte d'application, Oh et Stuerzlinger (2005) ont réalisé une technique permettant de placer des objets 3D dans des applications de CAO en calculant les collisions éventuelles entre les différentes entités. Ainsi, les objets ne pouvant s'intersecter "glissent" les uns sur les autres, et il devient facile de placer des objets relativement à d'autres.

De même, Nielson et Dan R. Olsen (1987) ont défini des *widgets* 3D permettant la sélection, le déplacement, la rotation et la mise à l'échelle d'objets 3D. Leur fonctionnement est très proche des widgets utilisés dans les applications 3D d'aujourd'hui, comme par exemple les widgets de Blender (Blender), visibles sur la Figure 5.5.

L'avantage que procure l'utilisation de widgets pour une telle tâche est qu'il est plus facile de spécifier la transformation à appliquer, ainsi que l'axe ou le plan de la transformation, car tout est réalisé à l'aide d'un seul et même outil. De plus, l'affichage du *widget* est intégré à la vue 3D. Ceci diminue la surcharge cognitive de l'utilisateur (Conner *et al.*, 1992).

Le principal inconvénient de ces *widgets* est que la sélection de certains éléments du contrôle peut demander de la précision. Ceci peut être un problème lorsque l'on souhaite manipuler une scène 3D à l'aide des doigts par exemple. De plus, la main de l'utilisateur, ainsi que le doigt ou le stylet, masquent une partie de l'écran, ce qui rend la sélection plus difficile encore.

L'utilisation de *widgets* sur un terminal mobile demandera donc une attention particulière. La sélection de ses différents éléments, par exemple l'axe de rotation ou de translation, ne doit pas nécessiter une grande précision, car cela irait à l'encontre du principe de PRÉCISION. Il sera ainsi préférable de privilégier des transformations où la co-localisation est maintenue, comme c'est le cas avec les *widgets* de Nielson et Dan R. Olsen (1987).

Le choix du repère de la transformation est également un point important. En effet, dans le cas d'une translation, si l'axe de la transformation est colinéaire, ou presque, avec la direction de vue, la précision des déplacements sera faible, et un petit mouvement créera un déplacement important. De plus, la colocalisation risque de ne pas être maintenue. Il

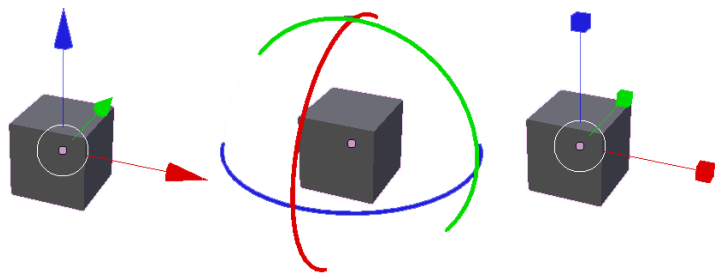


FIGURE 5.5 – Exemple de *widgets* utilisés pour la manipulation d'objets 3D (Image de Blender) : translation, rotation et mise à l'échelle.

sera donc préférable de travailler dans le repère de la caméra. Dans ce cas, le modèle se déplacera parallèlement au plan écran, et la colocalisation sera toujours possible. Cela ne permet cependant pas de déplacer le modèle en profondeur : pour ce type de translation, il ne sera pas possible de bénéficier de la colocalisation.

Une contrainte forte de beaucoup d'écrans tactiles est qu'ils ne détectent la présence que d'un seul point de contact. Cependant, certains écrans, comme celui de l'iPhone d'Apple par exemple, sont capables de détecter la présence de plusieurs points de contacts, ce qui permet d'envisager des niveaux d'interaction plus grands qu'avec un simple point de contact.

5.3 Interagir avec une scène 3D en utilisant un écran tactile multi-points

Le succès commercial du téléphone portable d'Apple, l'iPhone, est entre autres dû à la présence d'une caractéristique inédite dans le monde des terminaux mobiles : la présence d'un écran tactile *multi-points* (*multitouch* en anglais). Ce type d'écrans possède l'avantage de détecter la présence et la position de plusieurs points de contact, afin de permettre l'utilisation de plusieurs doigts simultanément pour interagir avec l'application.

Toutefois, les technologies utilisées et les techniques d'interaction présentes sur cet appareil sont inspirées de travaux plus anciens : en effet, des surfaces tactiles multi-points existent depuis plus longtemps, mais étaient présents sur des dispositifs encombrants, tels les *Tabletops*, plutôt destinés à une utilisation collective pour permettre à chaque personne présente autour de la table d'interagir avec les éléments affichés. Ces applications s'inspirent également de travaux réalisés en RV immersive, où des études se sont intéressées au contrôle bimanuel d'applications 3D.

Les travaux effectués sur l'utilisation de surfaces multi-points se sont plutôt intéressés à la manipulation d'objets 2D (photos, documents), et peu d'investigations ont été réalisées concernant la manipulation de modèles 3D. Cependant, Grossman et Wigdor (2007) dressent une taxonomie de l'utilisation d'applications 3D sur des Tabletops, que ce soit au niveau des contraintes d'affichage, des propriétés physiques ou des types d'applications, et proposent quelques conseils d'interaction. Les problèmes abordés ici sont toutefois éloignés du problème de l'interaction 3D sur terminaux mobiles, étant donné que les Tabletops ont des contraintes différentes des nôtres.

Benko et Feiner (2007) ont développé une technique de sélection 3D qui se contrôle comme un ballon au bout d'une ficelle. Ainsi, comme le montre la figure 5.6, la main dominante définit la position du ballon sur le plan XY, tandis que la distance entre cette main et la main non dominante définit la hauteur du ballon. Cette technique, bien que très intuitive car basée sur une manipulation réelle, n'est pas adaptée à une utilisation sur TM, car elle nécessite un système de réalité augmentée, pour superposer les mains et les images 3D.

Dans un cadre bien précis de "profondeur superficielle" (*Shallow-Depth*), Hancock *et al.* (2007) étudient la manipulation d'objets 3D à l'aide d'un, de deux ou de trois points de contact. Le terme de "profondeur superficielle" correspond à une interaction 3D dans un en-

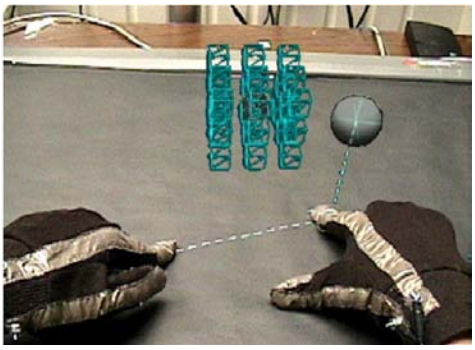


FIGURE 5.6 – *Balloon Selection* (Benko et Feiner, 2007) permet de sélectionner un point 3D en utilisant la métaphore d'un ballon tenu par une ficelle.

vironnement 3D à faible profondeur, typiquement un bureau où l'on empile des documents et des objets. Les DDL invoqués sont ainsi limités aux rotations selon les 3 axes, ainsi qu'à des translations dans le plan XY. Les translations dans l'axe Z sont possibles, mais limitées à une faible profondeur.

Leurs techniques d'interaction sont étudiées pour être les plus intuitives possibles, et sont principalement basées sur une différenciation de la position à laquelle l'utilisateur sélectionne l'objet à manipuler.

Ainsi, s'il attrape un objet par un coté, il sera possible de le faire pivoter autour de ses axes tout en le déplaçant, comme si l'objet était posé sur une table. Si, au contraire, on sélectionne l'objet en son centre, les déplacements vont être restreints à des translations le long du plan XY de la caméra. La manipulation est ainsi basée sur des mouvements physiques proches des mouvements naturels réalisés dans le monde réel. Il est par contre difficile de bien séparer les DDL, par exemple effectuer seulement une rotation.

L'utilisation de plusieurs doigts permet d'avoir un contrôle plus précis, en définissant un point de pivot à l'aide du premier point de contact, ou la translation le long de l'axe Z à l'aide de la distance séparant les deux doigts. Enfin, l'utilisation d'un troisième doigt permet de passer facilement entre différents modes (translations seulement, rotations seulement par exemple). Il ressort de cette étude que l'utilisation de trois doigts est plus efficace et précis. Cependant, la faible taille des écrans de terminaux mobiles empêche une utilisation de trois doigts. Il vaudra donc mieux se concentrer sur une utilisation à l'aide de deux doigts maximum.

Pour une tâche de sélection et de manipulation 3D, Schlattmann et Klein (2009) utilisent les deux mains pour "encercler" l'objet à sélectionner, puis pour le manipuler comme s'il était placé entre les mains de l'utilisateur.

Une approche intéressante a été développée pour de la navigation dans des environnements virtuels : le "Finger Walking In Place" (FWIP) (Kim *et al.*, 2008). Il s'agit d'une technique originale, où l'utilisateur contrôle l'avancement et les rotations de la caméra en faisant "marcher" ses doigts sur place. Il est ainsi possible de contrôler la vitesse et l'orientation du déplacement. Cette technique possède l'avantage de nécessiter un faible apprentissage, la métaphore employée étant très proche du déplacement naturel.

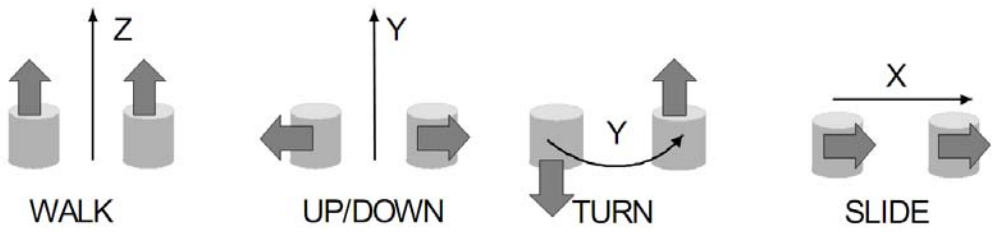


FIGURE 5.7 – Mouvements proposés par la métaphore du "*Magic Carpet*"(Zhai *et al.*, 1999).

Les écrans tactiles multi-points permettent à l'utilisateur de contrôler plusieurs curseurs simultanément. Cette approche a été étudiée dans des contextes de RV immersive à travers la manipulation *bimanuelle*. Dans ce cas, les utilisateurs pouvaient, à l'aide de capteurs de position situés sur leurs mains gauche et droite, avoir deux curseurs 3D affichés à l'écran. Ainsi, Balakrishnan et Kurtenbach (1999) ont utilisé la main non dominante (MND) pour contrôler la caméra 3D, et la main dominante (MD) pour se concentrer sur la tâche principale dans la scène 3D. Leur étude montre que les utilisateurs préfèrent les techniques bimanuelles pour une tâche de peinture de modèle 3D.

Une telle approche serait envisageable sur un terminal mobile : un premier point de contact contrôlerait les rotations du modèle 3D, tandis que le second point de contact serait dédié à une tâche différente. Il y aurait cependant des problèmes d'occultation dus à la taille réduite des écrans.

De même, Zeleznik *et al.* (1997) ont proposé un ensemble de techniques d'interaction 3D qui utilisent deux pointeurs de souris. Ces techniques sont principalement basées sur la sélection d'un point de pivot à l'aide du pointeur de la MND, et d'un point à manipuler avec la MD. Le principe ici est de toujours maintenir la corrélation entre la position du curseur de la MND et le point 3D sélectionné. L'objet (ou la caméra) est alors translaté et pivoté de façon à maintenir cette corrélation. Cette approche possède l'avantage de tirer parti de la propriété de co localisation, qui est très forte lorsque l'on utilise un écran tactile. Elle se rapproche et s'inspire d'une méthode développée par Mapes et Moshell (1995).

Plus récemment, Shen *et al.* (2009) ont cherché comment manipuler un modèle 3D lorsque l'on possède un écran *multi-points double face*. Ils ont ainsi pu isoler quelques gestes élémentaires qui permettent de déplacer ou de pivoter un objet 3D affiché sur l'écran.

Le problème de cette méthode est qu'elle est basée sur la sélection de deux points de la scène 3D, ce qui n'est pas propice à de la navigation libre. Dans ce cas, une technique telle le "Magic Carpet" (Zhai *et al.*, 1999) peut être envisagée. Elle est basée sur le mode de pilotage d'un bulldozer, et est destinée à être contrôlée à l'aide de deux joysticks. La Figure 5.7 montre les différents mouvements possibles avec cette interface : *Avancer* en poussant les deux joysticks vers l'avant, *Monter* en écartant les joysticks horizontalement, *Tourner* en opposant les joysticks verticalement, et *Translater* en poussant les deux joysticks vers la gauche ou la droite.

Cette technique utilise dans son implémentation deux joysticks élastiques, afin d'avoir un mécanisme de re-centrage efficace. En effet, elle est basée sur du contrôle de vitesse, dans lequel les périphériques élastiques sont plus performants (Zhai, 1995). Cependant, il est tout à fait possible, au prix de performances réduites, d'utiliser une telle technique avec

une surface tactile multipoints.

Le principal problème que pose l'utilisation de plusieurs doigts sur un écran tactile réside dans l'occultation d'une grande partie de l'écran qui en résulte. Cependant, les recherches actuelles en ergonomie voient se développer un nouveau moyen d'interagir sans occulter l'écran : l'interaction par l'arrière de l'appareil. Les premiers prototypes (Wigdor *et al.*, 2007) utilisaient une caméra et une surface tactile placé derrière l'appareil afin de détecter la position des doigts et d'afficher l'"'ombre" des doigts en transparence sur l'écran. Le problème d'une telle méthode est qu'elle risque d'être gourmande en ressources système. De plus, elle est plutôt encombrante. Les prototypes actuels utilisent seulement une surface tactile placée derrière l'appareil. Ceci permet une interaction tactile facilitée, même sur de très petits appareils (Baudisch et Chu, 2009), avec une plus grande précision qu'en interagissant directement sur l'écran (Yang *et al.*, 2009). Enfin, cela permet d'imaginer de nouvelles techniques de manipulation 3D (Shen *et al.*, 2009) qui utilisent les deux mains et les deux côtés de l'appareil, comme l'illustre la Figure 5.8.

Il est important de noter que les techniques employant un écran multi-points doivent obligatoirement être utilisées à deux mains, une main tenant l'appareil tandis que les doigts de la seconde main servent à interagir, ou bien les deux tenant l'appareil, l'interaction étant réalisée à l'aide des pouces. Le principe d'UTILISATION UNIMANUELLE n'est donc pas respecté ici.

5.4 Interagir avec une scène 3D en utilisant une caméra

Une grande majorité des terminaux mobiles actuels, téléphones portables, Smartphones, PDA, sont équipés d'une caméra vidéo. Leur résolution est sans cesse grandissante et est aujourd'hui tout à fait convenable, se rapprochant de la résolution des appareils photo numériques.

Les possibilités d'utilisation d'une caméra pour interagir avec un appareil mobile sont grandes. Il est par exemple possible, à partir de l'analyse du flux vidéo, de détecter et de suivre au cours du temps la position de points caractéristiques de l'image. Cette opération n'est cependant pas triviale, et peut être coûteuse en termes de temps de calcul, comme nous le verrons dans la suite.

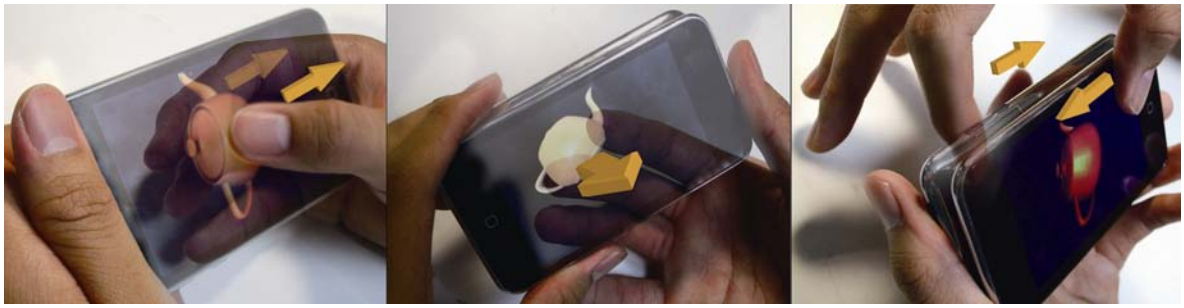


FIGURE 5.8 – Interaction 3D par l'arrière de l'appareil (Shen *et al.*, 2009) (translation XY, translation Z, rotation).

Une fois les points caractéristiques de l'image détectés, il est alors possible de reporter ces mouvements sur l'environnement 3D, afin de se déplacer et d'explorer son environnement virtuel (Capin *et al.*, 2006). Cette technologie est couramment employée dans les applications de *Réalité Augmentée*. En effet, il est crucial dans de telles applications de détecter la position de l'utilisateur, et de savoir ce qu'il est en train de regarder, afin d'y ajouter des informations. ARToolKit (Billinghurst *et al.*, 1999) par exemple, porté par la suite sur PocketPC (Wagner et Schmalstieg, 2003, 2006; Wagner *et al.*, 2009), détecte dans le flux vidéo la position et l'orientation de marqueurs positionnés dans l'environnement réel. Ceci permet de créer des applications où l'utilisateur interagit avec son environnement réel, comme l'Invisible Train (Wagner *et al.*, 2004) ou LevelHead (Oliver, 2008), visibles sur la Figure 5.9.

FIGURE 5.9 – Applications de Réalité Augmentée : Invisible Train (gauche) et LevelHead (droite).

Le principal obstacle de l'utilisation de la vidéo concerne les performances et la robustesse des algorithmes employés. En effet, la détection des points caractéristiques ou de motifs dans une image est une tâche coûteuse en termes de temps de calcul, ce qui ne satisfait pas le principe d' **ÉCONOMIE**. L'algorithme de détection utilisé devra être simplifié, afin de ne pas demander trop de ressources. Il faudra toutefois trouver un bon compromis entre rapidité et précision de l'algorithme, afin de ne pas avoir une technique lente et inutilement précise ou, au contraire, rapide mais fournissant des données erronées.

Beaucoup de ces techniques nécessitent de déplacer l'appareil, et ne respectent donc pas le principe d'**ORIENTATION** : l'angle de vision des écrans d'appareils mobiles est parfois faible, et l'utilisateur risque de perdre en visibilité lors des mouvements.

Afin de ne pas avoir à déplacer l'appareil, et respecter le principe d'**ORIENTATION**, il peut être préférable de devoir déplacer un objet devant la caméra. C'est ce principe qui est utilisé dans Tangimap (Hachet *et al.*, 2005a,b). Il est alors possible de manipuler 3 DDL en déplaçant une cible devant la caméra. Une version plus compacte, qui présente l'avantage de fournir un retour élastique, a été présentée récemment (Hachet *et al.*, 2008). Elle s'apparente à un joystick à 3 DDL et se base sur la reconnaissance d'un motif simple par la caméra.

Ces techniques s'inspirent de travaux tels que MagicMouse (Woods *et al.*, 2003) ou Visual Panel (Wu *et al.*, 2000; Zhang *et al.*, 2001), qui permettent de détecter plusieurs DDL en fonction des mouvements d'un marqueur ARToolKit (pour MagicMouse) ou d'une simple feuille de papier (dans le cas de Visual Panel). Enfin, Bretzner et Lindeberg (1998) ont proposé d'utiliser la main de l'utilisateur comme repère 3D, et de détecter ses mouvements

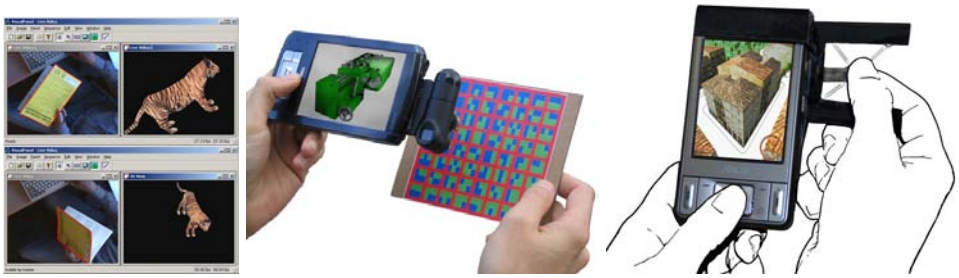


FIGURE 5.10 – Différentes techniques utilisant la caméra de l'appareil pour contrôler l'application 3D : Visual Panel (Zhang *et al.*, 2001) (gauche), Tangimap (Hachet *et al.*, 2005a) (centre) et Joystick 3DOF (Hachet *et al.*, 2008) (droite).

afin de manipuler un objet 3D. Ces différentes techniques sont illustrées sur la Figure 5.10.

Utiliser une caméra pour manipuler une scène 3D possède l'avantage de libérer l'affichage de tout contrôle superflu, et satisfait ainsi le principe d'AFFICHAGE, ce qui est un point important lorsque l'écran est de petite taille.

Il y a cependant quelques inconvénients :

- Les calculs (lire l'image depuis le flux vidéo, traiter et analyser l'image) prennent un temps non négligeable. Ceci réduit d'autant plus la puissance disponible pour l'application. Il convient donc d'utiliser une méthode aussi légère que possible, sans que cela ne pénalise la précision de la détection. Le principe de ÉCONOMIE n'est donc pas satisfait.
- Les conditions d'éclairement de l'environnement de l'utilisateur peuvent être changeantes. Il est alors parfois difficile pour l'algorithme de détecter les marqueurs, et de trop fortes variations peuvent perturber le système.
- L'utilisation est souvent bimanuelle, ce qui empêche son utilisation si une main est occupée. Le principe d'UTILISATION UNIMANUELLE n'est donc pas satisfait si la technique nécessite l'emploi des deux mains.

Le nombre de fonctionnalités offertes par les terminaux mobiles augmente régulièrement. La présence d'une caméra vidée est aujourd'hui devenue un standard. Cependant, les constructeurs d'appareils mobiles se sont également intéressés à d'autres types de périphériques, afin d'accroître le degré d'interactivité offert à l'utilisateur.

5.5 Interagir avec une scène 3D en utilisant divers périphériques

Certains appareils mobiles sont équipés de périphériques particuliers, qui peuvent être très utiles pour interagir avec un environnement 3D. Ces périphériques permettent de rajouter des degrés de liberté à l'appareil ou de proposer une interaction plus fluide ou plus intuitive.

FIGURE 5.11 – Jeu iPhone "Raging Thunder" tirant parti des capteurs d'inclinaison.

5.5.1 Joysticks

Les *joysticks* sont utilisés depuis de nombreuses années, notamment sur des ordinateurs de salon ou des consoles de jeux vidéos. Utiliser un joystick procure un avantage indéniable, car il fournit un signal analogique avec un retour élastique (Silfverberg *et al.*, 2001), ce qui permet de contrôler efficacement une vitesse de déplacement. De plus, son utilisation peut être unimanuelle (principe d'UTILISATION UNIMANUELLE), ne surcharge pas l'affichage, ne crée par d'occultations (principe AFFICHAGE) et ne nécessite pas de déplacer l'appareil (principe d'ORIENTATION). C'est donc un choix tout à fait intéressant. Malheureusement, il est rare de trouver des terminaux mobiles équipés d'un tel dispositif (voir section 1.2.5 pour des exemples de joysticks pour TM).

5.5.2 Accéléromètres et systèmes de positionnement

Certains appareils mobiles sont pourvus d'accéléromètres. Ils sont utilisés pour contrôler l'appareil : passer à la chanson suivante dans la liste de lecture en agitant l'appareil vers la droite ou pivoter une photo en changeant l'orientation de l'appareil par exemple. Ils sont également utilisés dans des jeux en 3D : ainsi, il est possible de piloter une voiture en pivotant l'appareil vers la gauche ou vers la droite (voir Figure 5.11).

Des applications de visualisation de modèles 3D tirant parti de capteurs d'orientation ont également été développées (Rekimoto, 1996), ainsi que des applications dédiées à la navigation libre (Fitzmaurice *et al.*, 1993; Cancrinus, 2004; Marsden et Tip, 2005). Les techniques utilisées ici sont très semblables, et utilisent les mouvements de l'appareil pour orienter et déplacer la caméra virtuelle. Il est également possible de rajouter les informations fournies par un système de positionnement, comme le GPS par exemple, afin de placer l'utilisateur dans un environnement 3D géoréférencé (Rantakokko et Plomp, 2003). Il est à noter que le GPS seul ne peut pas déterminer l'orientation de l'utilisateur. Pour cela, il est nécessaire de coupler le GPS à un *magnétomètre*, qui mesurera le champ magnétique terrestre afin de connaître la direction du pôle nord magnétique. Cependant, la précision actuelle des GPS est trop faible pour permettre de détecter de petits mouvements.

Ces techniques possèdent l'avantage de produire une interaction très intuitive, car basée sur des mouvements réels. Elles sont de même compatibles avec le principe d'UTILISATION

UNIMANUELLE. Cependant, elles ne respectent pas le principe d'ORIENTATION, car elles nécessitent de déplacer l'appareil. De plus, il risque d'y avoir des mouvements parasites si l'utilisateur est mobile.

5.6 Bilan

La variété des périphériques d'entrée que peuvent posséder les terminaux mobiles permet d'imaginer de nombreuses techniques d'interaction 3D. La table 5.1 dresse un récapitulatif des principaux périphériques que nous avons présentés dans ce chapitre, ainsi que les avantages et inconvénients de chacun pour interagir avec un environnement 3D.

Périphérique	**Avantages**	**Inconvénients**
Touches	– Utilisation unimanuelle – Plusieurs touches disponibles	– Signal binaire – 1 touche à la fois
Écran tactile	– Co-localisation – Contrôle continu et fluide – Beaucoup de techniques existantes	– La main occulte l'écran – Problème de précision avec une utilisation au doigt – Performances faibles en contrôle de vitesse
Multi-points	– Contrôle de plusieurs degrés de liberté simultanément – Interaction plus "naturelle"	– Occultation importante de l'écran
Caméra	– Écran libéré – Contrôle de plusieurs DDL	– Utilisation bimanuelle ou nécessité de déplacer l'appareil – Gourmand en ressources
Capteurs	– Écran libéré – Contrôle de plusieurs DDL – Mouvements "naturels"	– Nécessité de déplacer l'appareil – Mouvements parasites
Joysticks	– Écran libéré – Contrôle isométrique ou élastique – Habitude des utilisateurs de jeux vidéos	– Difficile à utiliser pour des tâches de positionnement – Peu répandus sur les TM

TABLE 5.1 – Récapitulatif

Dans la suite, nous nous concentrerons sur le contrôle direct d'une application 3D. Le chapitre 6 présente une technique de manipulation semi-automatique de caméra que nous avons développée appelée ScrutiCam. Elle permet à l'utilisateur de spécifier son point d'intérêt et de déplacer la caméra virtuelle vers une position propice à la visualisation de ce point et de son voisinage. Nous présenterons ensuite quelques exemples de techniques d'interaction 3D utilisant plusieurs points de contact. Puis nous présenterons, dans le chapitre 8, une étude sur l'utilisation de surfaces tactiles multi-points pour des tâches de contrôle de vitesse.

CHAPITRE 6

Manipulation semi-automatique de caméra 3D

FIGURE 6.1 – Les mouvements de caméra de ScrutiCam.

6.1 Introduction

Comme nous l'avons vu dans l'introduction, contrôler une caméra virtuelle à l'aide d'un écran tactile ou d'une souris n'est pas une tâche aisée. Il est donc important de développer des techniques d'interaction qui assistent l'utilisateur, lui permettant de manipuler facilement la caméra, et ainsi de se concentrer sur sa tâche principale, et non sur le contrôle de la caméra.

ScrutiCam(Decle *et al.*, 2009) est une technique de manipulation de caméra 3D que nous avons développée et qui est particulièrement adaptée aux tâches d'inspection. En effet, elle permet à l'utilisateur de manipuler la caméra en fonction d'un point d'intérêt sélectionné. Il est par exemple possible de le déplacer parallèlement au plan écran ou d'aligner la caméra avec la surface autour de ce point d'intérêt sélectionné par l'utilisateur avec un seul mouvement.

Ces mouvements de caméra sont réalisés en sélectionnant un point d'intérêt sur le modèle 3D, puis en déplaçant ce dernier sur le plan écran. Si la sélection est déplacée vers le centre de l'écran, la caméra est transformée de façon à faire face à la surface environnant le point sélectionné. Dans le cas contraire, la caméra est translatée parallèlement au plan écran (voir Figure 6.2). Tous les mouvements de caméra de ScrutiCam maintiennent une correspondance entre le curseur de la souris et la projection sur le plan écran du point cible. Ceci rend cette technique d'interaction particulièrement adaptée aux écrans tactiles, car elle satisfait le principe de **COLOCALISATION** que nous avons défini en introduction.

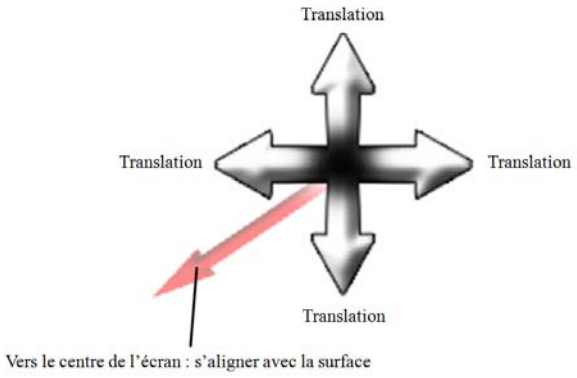


FIGURE 6.2 – Le *widget* de ScrutiCam. Un déplacement du curseur vers le centre de l'écran, représenté par la flèche rouge, aligne la caméra avec la surface. Dans le cas contraire, la caméra est translatée parallèlement au plan écran.

L'inspection est une tâche de navigation très utilisée notamment dans les applications de Conception Assistée par Ordinateur (CAO), où les utilisateurs ont besoin d'observer une zone particulière, de se rapprocher ou de s'éloigner de celle-ci, ou de tourner autour d'un point précis. Une telle tâche nécessite habituellement l'utilisation de plusieurs outils ce qui augmente le temps d'apprentissage des utilisateurs : une technique de *trackball* afin de pivoter autour du modèle, une technique de *panning*, afin de translater la caméra parallèlement au plan écran et une technique permettant de se rapprocher du modèle, comme le *flying* ou un *zoom*.

Buxton (1986) explique que chaque changement de contrôle divise l'action en *unités* séparées, car la continuité du mouvement est interrompue. Ceci mène à une séparation de la tâche en sous-tâches et à une charge cognitive plus importante.

La plupart des logiciels de visualisation 3D assignent différentes techniques de navigation aux boutons de la souris ou à des raccourcis clavier. Par exemple, la molette de la souris peut être utilisée pour zoomer, tandis qu'un clic avec le bouton droit sert à translater la caméra. Les écrans tactiles ne possédant pas de boutons ni de molette, il n'est pas possible d'adopter une telle stratégie lorsque l'on utilise ce genre de périphérique. Par conséquent, l'utilisation des techniques standards sur de tels dispositifs demande une attention particulière. Il est bien entendu possible d'avoir recours à des menus, à des barres d'outils ou à un système de commandes gestuelles pour passer d'une technique à une autre. Cependant, il est tout de même préférable d'éviter de changer de technique trop fréquemment. Pour cette raison, des techniques de navigation ont été développées, afin de donner à l'utilisateur la possibilité d'effectuer des mouvements de caméra complexes à l'aide d'un seul et même outil.

Michael Gleicher et Andrew Witkin ont proposé un contrôle de la caméra à base de contraintes (Gleicher et Witkin, 1992). L'utilisateur peut sélectionner des sommets dans la scène. Lors des déplacements de caméra, le système calcule une position et une orientation de caméra qui maintiennent ces sommets à la même position à l'écran. Ceci permet de réaliser des mouvements de caméra intéressants, comme par exemple tourner autour d'une arête d'un objet. Cependant, l'utilisation peut être contraignante à cause de la nécessité

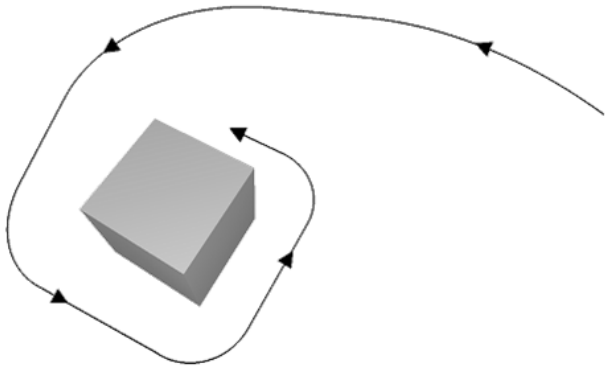


FIGURE 6.3 – La technique HoverCam de Khan *et al.* (2005) permet d'aligner la caméra avec la surface d'un objet 3D.

d'utiliser plusieurs outils (sélection et désélection de sommets, déplacement de la caméra).

HoverCam (Khan *et al.*, 2005) est une technique de manipulation de caméra qui a été étudiée pour des tâches d'inspection. Le principe de cette technique est de chercher dans la direction du mouvement de la caméra le point le plus proche sur la géométrie et d'aligner progressivement la caméra avec ce point. Ceci permet de tourner autour des objets de la scène et de se déplacer le long de la surface à l'aide d'un seul outil (voir Figure 6.3). Le principal défaut de cette technique est qu'elle utilise un algorithme de "recherche du point le plus proche" qui est gourmand en ressources. Ainsi, il est difficile, voire impossible, d'utiliser une telle technique sur une scène complexe, où les objets possèdent un grand nombre de polygones. Ceci est bien évidemment un problème déterminant pour une utilisation sur un terminal mobile, car cela ne satisfait pas le principe d'**ÉCONOMIE**.

Nous proposons une technique de manipulation de caméra pour des tâches d'inspection appelée **ScrutiCam**. Elle peut être utilisée avec n'importe quel dispositif de pointage, souris ou écrans tactiles. Son algorithme est "basé image", et ne dépend donc pas de la complexité géométrique de la scène.

6.2 Présentation

Le principe de ScrutiCam est d'analyser la surface autour du point sélectionné par l'utilisateur afin de calculer une "vue cible" appropriée. Ensuite, l'utilisateur pourra, en fonction de ses mouvements sur l'écran, se déplacer progressivement vers cette vue cible.

La section 6.2.1 décrit les données que nous extrayons de l'analyse de la surface du voisinage et comment nous les utilisons pour calculer la vue cible. La section suivante détaille les contrôles possibles ainsi que les mouvements de caméra associés.

6.2.1 Analyse de surface

Beaucoup de techniques de manipulation de caméra basent leurs calculs sur la position du centre du modèle étudié. C'est le cas de la technique du *trackball*, où le centre du modèle est utilisé comme point de pivot de la caméra virtuelle. Certaines techniques de déplacement utilisent aussi la distance entre la caméra et le centre de modèle pour calculer une vitesse de déplacement appropriée. Cependant, une telle stratégie peut ne pas être optimale. C'est le cas lorsque l'on travaille sur des modèles aux dimensions disproportionnées, un grand bâtiment par exemple, ou sur de très grands modèles, comme un terrain. Dans de tels cas, le centre du modèle peut être situé très loin du centre d'intérêt sur lequel l'utilisateur est concentré. Ceci risque de produire des mouvements de caméra inappropriés car trop rapides ou inattendus.

Un des principes de ScrutiCam est de considérer la surface visible du modèle 3D afin de calculer les mouvements de caméra. L'analyse de la surface visible du modèle peut fournir des informations utiles comme les *vecteurs normaux* à la surface, représentés sur la Figure 6.4, ou la présence d'*arêtes* ou de *contours*. Ces informations permettront de calculer une vue adaptée à la visualisation de la surface considérée.

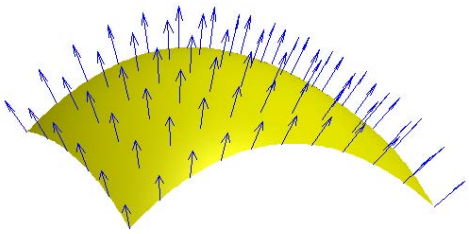


FIGURE 6.4 – Les vecteurs normaux à une surface 3D.

Contrairement à HoverCam, qui analyse la surface du modèle dans la direction du mouvement de la caméra, ScrutiCam laisse à l'utilisateur le choix de son *point d'intérêt*, qu'il sélectionne par un simple clic. L'algorithme de ScrutiCam analyse alors le *voisinage* de ce point d'intérêt afin de calculer un *vecteur normal moyen* (c'est à dire l'orientation moyenne de la surface dans le voisinage du point cible), de détecter la présence de contours et de calculer leur éventuelle orientation. La méthode utilisée pour estimer le vecteur normal moyen ainsi que pour détecter la présence de contours est détaillée dans la section 6.3

Le fait de prendre en considération le voisinage du point d'intérêt possède des avantages intéressants. Tout d'abord, ceci permet de "lisser" les surfaces irrégulières. En effet, si la continuité de la surface du modèle n'est pas C^1, l'angle entre deux vecteurs normaux proches peut être grand. Dans ce cas, deux points cibles proches l'un de l'autre produiraient des caméras cibles très éloignées l'une de l'autre. La Figure 6.5 (gauche) montre ce problème, qui serait très perturbant pour l'utilisateur qui ne serait pas en mesure de prédire où la caméra se placerait. En considérant le voisinage dans les calculs, ce problème est résolu (voir Figure 6.5 (droite)).
Enfin, la présence d'arêtes et de contours dans le voisinage du point d'intérêt donne des informations sur la discontinuité de la surface ou sur la présence d'autres objets proches de la zone d'intérêt. Par conséquent, si un contour est détecté dans le voisinage, nous utilisons

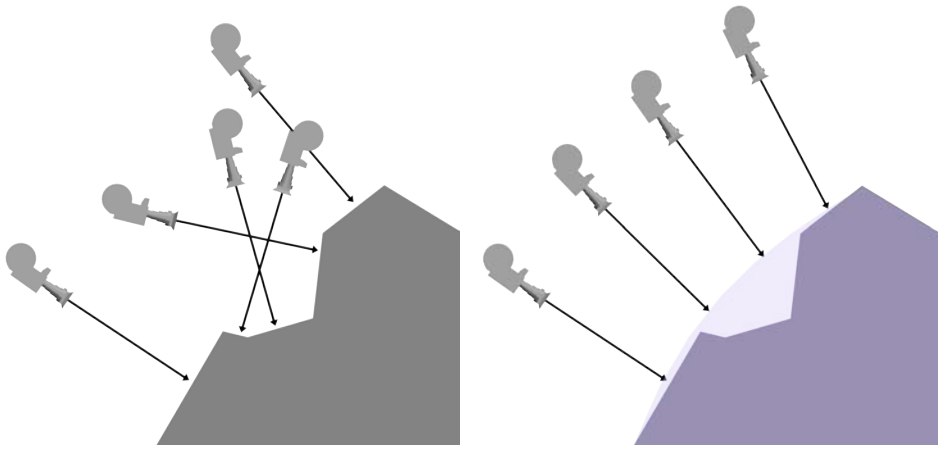


FIGURE 6.5 – Problème provoqué par l'utilisation du vecteur normal des faces. A gauche, la caméra utilise le véritable vecteur normal de la surface. A droite, nous prenons en compte le voisinage du point cible pour lisser les mouvements de la caméra.

cette information pour produire une caméra cible qui fait face à ce contour, afin d'avoir une bonne vision de celui-ci.

6.2.2 Calcul de la *vue cible*

A partir des données extraites de l'analyse de la surface, nous calculons une *caméra cible* C_t, c'est à dire une position P_t, une direction de vue $\overrightarrow{D_t}$ et un *vecteur Haut* $\overrightarrow{U_t}$ permettant une bonne visibilité de la zone d'intérêt.

Position cible P_t

La caméra cible doit faire face à la zone pointée par l'utilisateur. Par conséquent, elle sera placée le long de la demi-droite $[T\overrightarrow{N})$, où T représente la position du point cible et $\overrightarrow{N}$ le *vecteur normal moyen* calculé à partir de l'analyse de la surface.

Nous choisissons de conserver la distance d entre la caméra et le point cible T fixe. Par conséquent, la position de la caméra cible sera donnée par l'équation suivante :

$$P_t = T + d * \overrightarrow{N}. \tag{6.1}$$

Direction cible D_t

De même, étant donné que C_t doit faire face au point sélectionné, la direction $\overrightarrow{D_t}$ sera égale à l'opposé du vecteur normal moyen $\overrightarrow{N}$:

$$\overrightarrow{D_t} = -\overrightarrow{N}. \tag{6.2}$$

Vecteur *haut* cible P_t

Enfin, il nous faut également calculer un *vecteur haut* (*up vector* en anglais) cible, qui définira l'orientation de la caméra, et qui désigne la verticalité. Il y a plusieurs choix de vec-

teurs "haut" possibles. L'article décrivant la technique HoverCam (Khan *et al.*, 2005) identifie quatre modèles de vecteur haut :

- *Global* Dans le modèle *global*, la notion de "haut" est donnée par la direction du Pôle Nord. Ainsi, quelle que soit la position de la caméra sur le globe, le Pôle Nord sera toujours vers le haut de l'écran. Si l'utilisateur dépasse le Pôle Nord, la caméra effectue alors une rotation de 180°.
- *Local* Ce modèle dépend de la vue courante de l'utilisateur et est toujours dirigé vers le haut de la fenêtre de vue. Dans ce cas, si l'on dépasse le Pôle Nord, la Terre apparaîtra à l'envers.
- *Dérivant* Ce modèle oriente le vecteur Haut dans la direction du mouvement. On peut comparer ce modèle au pilotage d'un avion.
- *Personnalisé* Enfin, certains objets nécessitent l'utilisation de vecteurs Haut particuliers. Par exemple, dans le cas de l'observation d'un modèle de voiture, il apparaît normal d'avoir un vecteur Haut pointant vers le haut de la voiture lorsque l'on observe les cotés de celle-ci. Par contre, lorsque l'on observe par dessus ou par dessous, il peut être plus approprié d'avoir le *haut* pointant vers l'avant de la voiture.

Nous avons seulement retenu les modèles *Global*, *Local* et *Personnalisé*. L'utilisateur peut changer de modèle de vecteur Haut à travers les paramètres de l'application.

6.2.3 Contrôle et mouvements de caméra

Les mouvements de ScrutiCam sont contrôlés en effectuant un "cliquer-glisser", c'est à dire en cliquant sur le point cible et en le déplaçant tout en maintenant le bouton gauche de la souris enfoncé. Cette approche est fréquemment utilisée dans les applications afin, par exemple, de déplacer des icônes ou de dessiner une forme. De plus, ce geste est tout à fait adapté aux écrans tactiles car il profite de la *co-localisation* du doigt et du curseur, comme le suggère le principe de COLOCALISATION. Par conséquent, la manipulation est plus directe et semble plus intuitive.

Nous voulions que ScrutiCam utilise cette co-localisation. De ce fait, la zone sélectionnée par l'utilisateur reste constamment sous son doigt lors de la manipulation.

Dès que l'utilisateur sélectionne son point d'intérêt, une caméra cible est calculée, ainsi qu'une trajectoire allant de la caméra courante à la caméra cible. Cette trajectoire est une interpolation linéaire de la position et de l'orientation de la caméra. Ainsi, la position de la caméra sera interpolée entre sa position courante P_0 et sa position cible P_1, à l'aide de la formule suivante :

$$P(t) = P_0 + t * (P_1 - P_0), t \in [0; 1] \tag{6.3}$$

Afin de déplacer la caméra le long de la trajectoire, le paramètre t doit donc varier entre 0 et 1 (voir Figure 6.6). Si nous analysons la position du point cible à l'écran, nous remarquons que :

- à t = 0, la position à l'écran du point cible correspond à la position initiale du point sélectionné.
- à t = 1, la caméra est face au point sélectionné. Par conséquent, le point sélectionné est situé au centre de l'écran.

Ainsi, afin de conserver la relation entre la position du point d'intérêt et la position du curseur, le paramètre t est contrôlé par la distance entre la position initiale du curseur et le

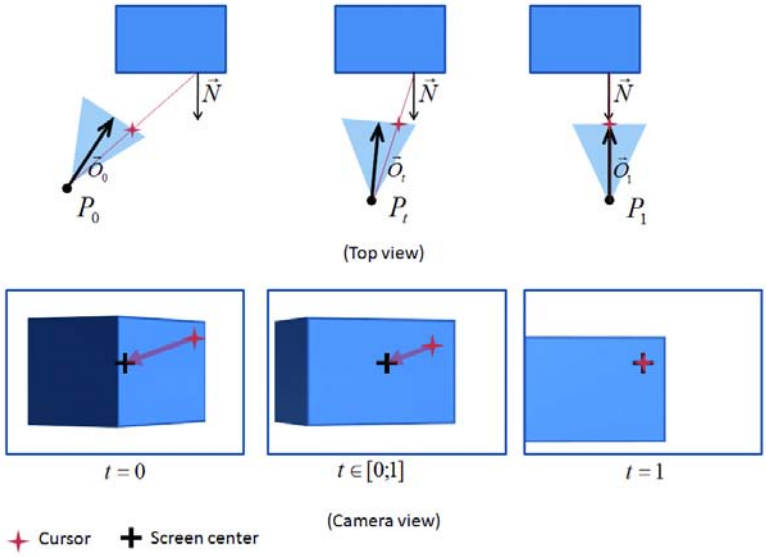


FIGURE 6.6 – Interpolation de la position P et de l'orientation $\vec{O}$ de la caméra.

centre de l'écran : ainsi, $t = 0$ lorsque le curseur est à sa position initiale, $t = 1$ lorsqu'il est au centre de l'écran.

6.3 Détails d'implémentation

ScrutiCam a été développé et testé sur différents systèmes : un PDA (avec écran tactile, utilisation au stylet et au doigt), un UMPC (avec écran tactile, utilisation au doigt), une station de travail standard (avec souris standard) et un écran tactile de grande taille (en utilisant un tableau blanc électronique *e-Beam*). Nous l'avons développé avec OpenGL, pour les versions Windows et Linux, et avec OpenGL|ES, pour la version Windows Mobile. Il est à noter que pour la version OpenGL|ES, nous avons modifié la bibliothèque *Vincent 3D*(Will.), une implémentation libre d'OpenGL|ES 1.1. En effet, OpenGL|ES n'autorise pas la lecture du contenu du tampon de profondeur, ce qui est indispensable pour pouvoir connaître la position de la projection 3D d'un point 2D de l'écran. Nous avons donc rajouté cette fonctionnalité, bien que cela nous éloigne des spécifications d'OpenGL|ES.

Une solution possible serait de calculer la position à l'aide d'une technique de *lancer de rayon*. Une telle approche sera décrite plus tard, dans la section 10.1.1.2 de la partie suivante.

6.3.1 Choix de l'espace de calcul

Afin de calculer le vecteur normal moyen et de détecter la présence de contours dans le voisinage de la sélection, deux solutions s'offraient à nous. Une première solution est "basée géométrie". Elle consiste à parcourir les triangles limitrophes au triangle auquel appartient le point sélectionné. Une seconde solution consiste à se baser uniquement sur ce qui est affiché à l'écran, l'approche "basée image". Dans notre cas, l'approche "image" est bien mieux adaptée que la "géométrique", pour les raisons suivantes :

– Les calculs sont indépendants de la complexité de la scène. A cet effet, notre technique

aura les mêmes performances si la scène affichée est composée de quelques triangles ou de plusieurs millions de polygones. Ceci est particulièrement important dans le cas de la version pour appareils mobiles, où la puissance de calcul est limitée.
- L'utilisation de *shaders* permet de détecter facilement la présence de contours. De nombreuses méthodes existent, utilisées par exemple pour faire du rendu expressif. Cependant, il n'est pas possible, à l'heure actuelle, d'utiliser de tels systèmes sur des terminaux mobiles, car ceux-ci ne possèdent pas de carte graphique programmable. Il est à noter que les prochaines générations de terminaux mobiles possèderont cette fonctionnalité, grâce à l'utilisation de la version 2.0 d'OpenGL|ES.
- Aucune connaissance *a priori* du modèle n'est nécessaire. Il n'y a pas non plus besoin d'avoir une structure particulière pour le maillage (sphere-trees, octree par exemple) pour optimiser la recherche des points ou des faces voisines.

Il y a cependant quelques inconvénients mineurs :
- Il n'est pas possible d'avoir des informations sur les parties qui ne sont pas affichées. Par conséquent, il nous faudra donc "deviner", ou tout du moins estimer, ce que cache un contour ou une bordure par exemple.
- La précision des calculs est au pixel près, alors qu'une approche géométrique peut être plus précise. Cependant, cette précision suffit dans la grande majorité des cas.

Nous avons donc tout naturellement choisi d'utiliser une approche basée "image" pour calculer le vecteur normal moyen ainsi que pour détecter la présence de contours.

6.3.2 Approximation locale du modèle

La technique que nous utilisons pour calculer le vecteur normal moyen consiste à projeter un motif, visible sur la figure 6.7, autour du point cible. Le motif étant composé de points et de triangles, la projection de ceux-ci sur le modèle 3D crée un maillage que nous utilisons pour calculer le vecteur normal moyen. Il est obtenu en calculant la moyenne normalisée des vecteurs normaux des triangles du motif projeté. La partie gauche de la figure 6.7 montre le motif dessiné sur le plan écran. La projection de ce motif sur le modèle 3D est visible sur la partie droite de la figure. Dans notre implémentation, le diamètre du motif était de 1/10 de la hauteur de l'écran.

Afin d'avoir une plus grande précision, le nombre de triangles et le nombre d'itérations concentriques du motif sont paramétrables. Ainsi, si la scène affichée est très discontinue, il peut être préférable d'avoir un grand nombre de triangles et deux ou trois itérations. Ceci aura pour effet de lisser l'approximation locale du modèle. Au contraire, sur des scènes simples, une seule itération et quelques triangles seront suffisants.

6.3.3 Normale des parties cachées

Lorsque le point sélectionné est proche de la bordure du modèle, cela peut signifier que l'utilisateur souhaite voir la partie cachée de celui-ci. Nous proposons une manière de réaliser un tel mouvement de caméra en prenant en compte la présence d'une bordure ou d'un contour dans le calcul du vecteur normal moyen.

La détection de contours est également réalisée lors de la projection du motif sur le modèle 3D. Si certains points du motif n'ont pas pu être projetés, ou sont trop éloignés du point

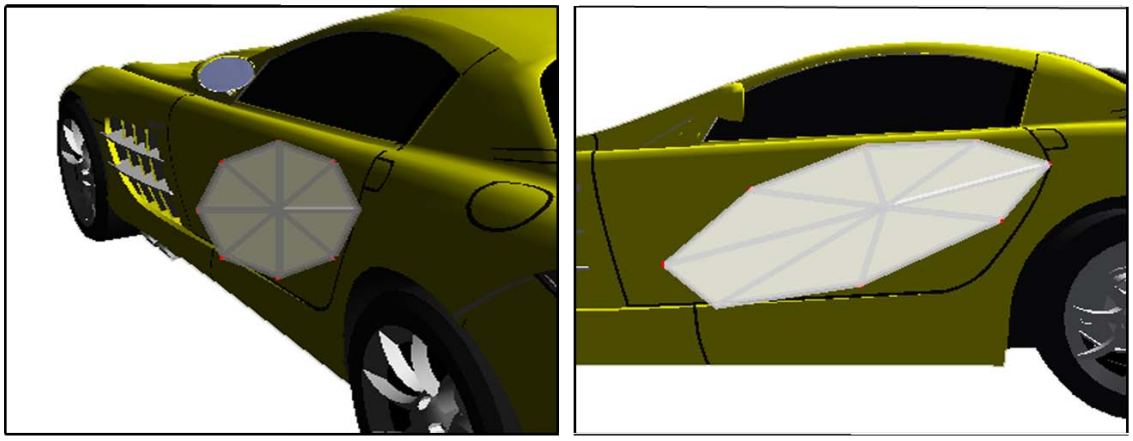


FIGURE 6.7 – L'approximation locale du modèle est réalisée en projetant un motif autour du point cible.

cible, cela signifie que le point sélectionné est proche d'une bordure. Ils sont alors considérés comme "nuls", et les triangles qui les contiennent aussi. Les points voisins des points "nuls" définissent la *bordure estimée* du modèle. Il est alors facile d'obtenir la "direction de la bordure", c'est à dire un vecteur perpendiculaire à la bordure estimée.

Le vecteur normal moyen est alors calculé comme la moyenne des vecteurs des triangles "non nuls" du motif et du vecteur de direction de la bordure.

6.3.4 Performances

L'utilisation de ScrutiCam ne pénalise pas les performances de l'application. En effet, le calcul de projection, utilisé pour calculer la position du point d'intérêt, n'est effectué qu'une seule fois, lorsque l'utilisateur touche l'écran. Cette opération peut être coûteuse car elle nécessite de lire le tampon de profondeur et d'effectuer des calculs matriciels afin de déterminer la position 3D de la projection d'un point 2D. Il est donc important de ne pas l'effectuer à chaque image, mais plutôt de manière ponctuelle.

Une fois la position du point d'intérêt calculée, les seules opérations effectuées à chaque image sont des interpolations linéaires, peu coûteuses, afin de déterminer la position et l'orientation courante de la caméra par rapport au point d'intérêt. Ceci est une différence fondamentale par rapport à HoverCam, dont l'algorithme cherche le point le plus proche de l'utilisateur à chaque image.

6.4 Discussions

ScrutiCam permet d'observer et de se déplacer le long d'objets 3D quelle que soit leur géométrie. Son algorithme d'estimation de la surface du modèle étant basé image, il est totalement indépendant de la complexité de la scène et est peu gourmand en ressources, ce qui satisfait le principe d'**ÉCONOMIE**.

Nous n'avons pas encore réalisé de véritable d'étude utilisateur visant à évaluer ScrutiCam. Cependant, les personnes qui l'ont essayé ont apprécié la facilité avec laquelle il était

possible de se déplacer et d'observer des parties d'un modèle. Cette technique a été intégrée à un démonstrateur pour le projet ANR RaxEnv [1], et fait partie des techniques de navigation proposées à l'utilisateur.

Quelques problèmes subsistent cependant. Si la surface du modèle est très irrégulière, le vecteur normal local risque de voir son orientation varier. Ceci produit des mouvements de caméra inappropriés. De plus, si un des points du motif est projeté sur une zone du modèle à forte discontinuité, le calcul du vecteur normal moyen risque d'être faussé. Une solution possible à ce genre de problèmes serait d'utiliser une méthode de calcul du vecteur normal moyen plus précise. Il serait par exemple intéressant d'étudier l'utilisation de *shaders*. En effet, les techniques permettant de détecter la présence de contours à l'aide de *shaders* sont nombreuses, et il serait alors facile de choisir les points du voisinage à considérer ou, au contraire, à ne pas prendre en compte dans les calculs.

Le problème principal de cette technique est qu'elle ne fonctionnerait que sur des ordinateurs de bureau. En effet, les terminaux mobiles actuels ne possèdent pas encore de cartes graphiques programmables, et ne permettent donc pas d'utiliser des *shaders*.

Un autre problème de ScrutiCam est qu'il n'est pas possible de se rapprocher ou de s'éloigner de la surface à observer. Il est possible de définir une distance, mais celle-ci restera constante lors des déplacements. Une solution possible est de combiner cette technique à une méthode permettant de sélectionner facilement plusieurs outils. Dans nos expérimentations, nous avons combiné ScrutiCam à une technique de zoom : un "cliquer-glisser" simple permettait d'utiliser ScrutiCam, tandis qu'un "double cliquer-glisser", c'est à dire un double clic dont le second clic est maintenu, permettait d'activer la fonction de zoom. Cependant, le geste "double cliquer-glisser" n'est pas facilement réalisable, particulièrement sur un écran tactile.

6.5 Conclusion

ScrutiCam fournit un contrôle direct assisté de la caméra. En analysant la surface autour du point d'intérêt de l'utilisateur, une nouvelle vue est calculée de façon à faire face à la surface sélectionnée. Une telle approche possède l'avantage de permettre des mouvements rapides, et d'observer facilement différentes parties d'un modèle. Elle est cependant réservée à l'observation de modèles, et n'est pas utilisable dans un cadre de navigation libre. Pour ce type de tâches, il est préférable d'utiliser des techniques basées sur du contrôle de vitesse. Toutefois, un des inconvénients de ces techniques est qu'elles ne fournissent que peu de degrés de liberté à contrôler. Il est alors nécessaire d'utiliser plusieurs techniques afin de contrôler plus librement les mouvements de la caméra.

Nous nous sommes intéressés à l'utilisation de surfaces multi-points pour le contrôle de ce type de tâches. Ainsi, dans le chapitre suivant, nous présentons quelques techniques de navigation libre que nous avons développées. Ces techniques sont contrôlées par plusieurs doigts simultanément afin d'offrir plus de liberté et plus de précision à l'utilisateur.

1. http ://raxenv.brgm.fr/

CHAPITRE 7

Étude de cas : Utilisation de plusieurs doigts pour diverses taches d'interaction 3D

Les écrans multi-points recèlent un fort potentiel pour l'interaction 3D, notamment grâce au grand nombre de degrés de libertés qui peuvent être contrôlés simultanément lors de l'utilisation de plusieurs doigts.

Dans ce chapitre, nous présentons quelques techniques que nous avons étudiées qui tirent parti de l'un ou de l'autre de ces avantages. Ces techniques peuvent bien entendu être utilisées sur des écrans tactiles multipoints. Il est néanmoins possible, avec quelques restrictions, de les adapter à des écrans tactiles résistifs classiques, en utilisant la technique de "*pseudo multi-points*" décrite en Section 1.2.3.3.

7.1 Contrôle simultané de plusieurs degrés de liberté : Navigation libre dans des environnements 3D

Le contrôle de plusieurs degrés de liberté a longtemps été le principal enjeu de la recherche en interaction 3D. En effet, la plupart des transformations de base de l'interaction 3D peut être réalisée selon plusieurs axes, et une manipulation totalement libre des objets 3D nécessiterait un contrôle de tous ces degrés de liberté simultanément.

L'utilisation d'un écran tactile classique ne fournissant que deux degrés de liberté, il est alors nécessaire d'utiliser des techniques permettant de choisir facilement les degrés de liberté sur lesquels travailler.

Cependant, les écrans multi-points permettent d'envisager le contrôle d'un plus grand nombre de degrés de liberté simultanément. En effet, il est possible sur ce type d'écrans d'interagir à l'aide de plusieurs doigts. Comme chaque doigt peut se mouvoir dans plusieurs directions, ils peuvent potentiellement contrôler plusieurs degrés de liberté chacun. Cependant, la mobilité des différents doigts n'est pas totale, et il faut prendre en considérations les limites anatomiques de la main afin de proposer un contrôle qui soit confortable et facile à effectuer.

Deux doigts prédominent chez l'être humain : le pouce et l'index. Ces doigts possèdent une disposition particulière qui nous permet d'appréhender des objets, de les manipuler et de les utiliser. Il est courant, dans nos mouvements, que l'un de ces deux doigts soit utilisé comme *point de référence* pour les autres doigts de la main. Par exemple, le pouce est considéré comme le point de référence lorsque l'on doit attraper des objets. En revanche, la désignation d'un point ou d'une direction se fait à l'aide de l'index.

En partant de ces observations, nous avons développé et étudié deux manières de contrôler plusieurs degrés de liberté, l'une considérant le pouce comme point de référence, l'autre utilisant l'index. Nous les avons appliquées à une tâche de navigation libre, qui nécessite le contrôle de plusieurs degrés de liberté afin de permettre à la caméra virtuelle de se déplacer selon les trois axes et de s'orienter librement dans l'espace.

7.1.1 Utilisation du pouce comme point de référence

Nous avons défini et développé une première technique permettant de contrôler plusieurs degrés de liberté à l'aide des doigts. Cette technique considère le pouce comme étant un point de référence. L'index est alors mobile et peut effectuer des rotations autour du pouce. Deux degrés de liberté peuvent alors être contrôlés par l'écartement du pouce et de l'index, et par l'angle formé par ces doigts et l'axe vertical. Il est néanmoins possible de rajouter deux degrés de liberté supplémentaires, contrôlés par le déplacement des deux doigts simultanément. Ceci permet alors de contrôler quatre degrés de liberté.

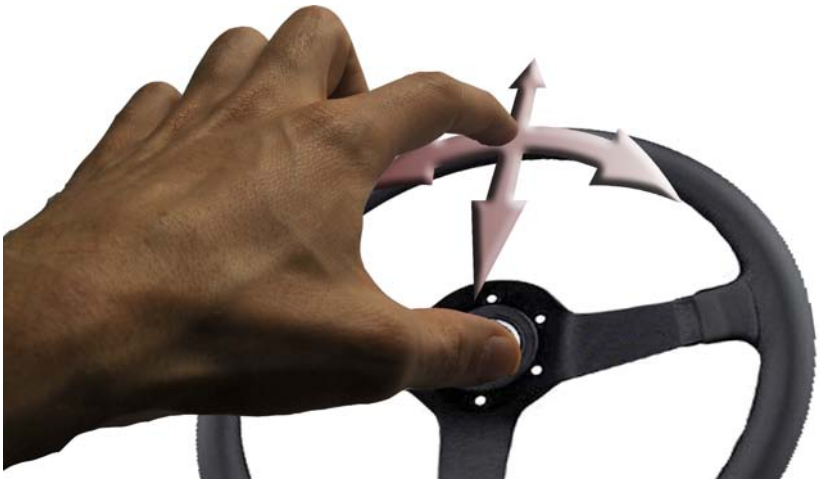


FIGURE 7.1 – Navigation libre à l'aide de deux doigts : métaphore du volant.

L'application de cette technique à la navigation 3D est simple. Il est par exemple possible de contrôler la vitesse du déplacement à l'aide de l'écartement des deux doigts. L'angle de rotation horizontal, qui permet de regarder vers la gauche ou vers la droite, est contrôlé par l'angle que forment les deux doigts par rapport à la verticale. Les deux degrés de liberté supplémentaires peuvent alors être assignés au contrôle des déplacements latéraux et à la rotation verticale de la caméra.

Cette technique peut être considérée comme utilisant une métaphore de volant de

voiture, car l'utilisateur contrôle l'angle des roues de la voiture virtuelle ainsi que la vitesse de déplacement de celle-ci (voir Figure 7.1).

Cette configuration n'est bien sur pas unique. Il est tout à fait possible d'utiliser les mouvements simultanés des deux doigts pour contrôler les déplacements d'une technique de *flying*. L'écartement des deux doigts pourrait alors contrôler l'orientation verticale de la caméra.

7.1.2 Utilisation de l'index comme point de référence

La technique que nous présentons ici permet de contrôler plusieurs degrés de liberté simultanément. Elle est basée sur une observation faite sur l'anatomie de la main, et des degrés de liberté du pouce, de l'index et du majeur : comme la Figure 7.2 l'illustre, l'index, situé entre le pouce et le majeur, est couramment utilisé pour pointer ou pour sélectionner. Lorsque l'index est tendu, les mouvements du pouce sont principalement horizontaux, tandis que les mouvements du majeur sont verticaux.

Il apparaît alors intéressant d'attribuer des contrôles de différents degrés de liberté aux trois doigts : 2 DDL sont contrôlés par l'index (comme pour un écran tactile simple), 1 DDL est contrôlé par les mouvements horizontaux du pouce, et 1 DDL est contrôlé par les mouvements verticaux du majeur, comme l'illustre la Figure 7.2.

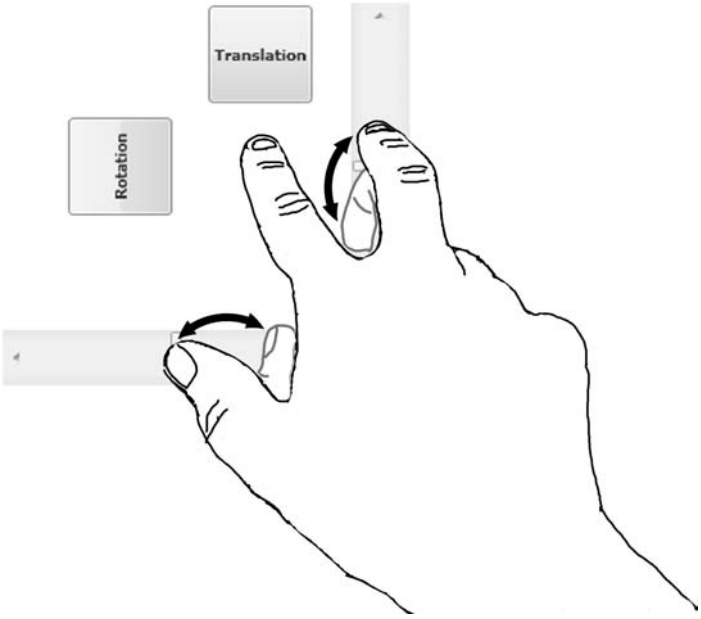


FIGURE 7.2 – Mouvements des doigts lorsque l'index est utilisé comme point de référence.

L'utilisation de cette technique permet d'avoir un contrôle plus précis des mouvements de caméra, en permettant par exemple de se déplacer à l'aide d'une méthode de *flying*, contrôlée grâce aux mouvements de l'index, tout en orientant la caméra verticalement à l'aide des mouvements du majeur.

Afin de pouvoir paramétrer facilement cette interface, nous avons développé, avec l'aide de Joachim Pouderoux (ingénieur de recherche au sein de l'équipe Iparla), un outil qui permet de configurer ce type d'approches. Cet outil permet de créer un *widget* qui apparaîtra autour de l'index de l'utilisateur lorsque celui-ci touchera l'écran.

Ce widget est composé de deux types d'éléments :
- Des boutons, afin de changer de mode ou d'activer une option par exemple.
- Des glissières, afin d'avoir un contrôle continu d'un paramètre particulier.

Les différents éléments présents autour de l'index, ainsi que leur position et l'action qui leur est associée, sont paramétrables *via* un fichier *XML*. Il est alors facile de décrire plusieurs types d'interfaces, en fonction de l'application utilisée par exemple, et de passer de l'une à l'autre.

7.1.3 Contrôle sans occultations des doigts

Le principal problème des techniques précédentes est que les doigts occultent une partie importante de l'écran. Afin de limiter au maximum la surface cachée par les doigts lors de l'interaction, il peut être préférable de ne pas avoir à placer ses doigts au centre de l'écran mais plutôt sur les côtés. Ainsi, nous proposons d'utiliser les bords de l'écran de l'appareil comme contrôleurs de différents degrés de liberté : il est alors possible de contrôler quatre degrés de liberté différents.

7.2 Utilisation de plusieurs curseurs 3D : navigation et manipulation de modèles 3D

L'un des avantages que procure l'utilisation d'un écran tactile réside dans la colocalisation. Les techniques que nous avons décrites dans la section précédente ne tirent pas parti de cet avantage. Dans cette section, nous présentons une technique permettant d'observer des scènes ou des modèles 3D, l'interaction étant réalisée à l'aide de plusieurs points de contact. Nous avons cherché à conserver le principe de colocalisation, afin de créer une interaction plus directe et intuitive.

La technique présentée ici s'inspire des idées formulées par Zeleznik *et al.* (1997). Cependant, ces techniques sont destinées à une utilisation à l'aide de deux souris. Une analyse des contraintes d'utilisation lors de l'emploi d'une tablette y est toutefois abordée. Dans notre contexte, l'interaction se fait à l'aide de deux doigts, ce qui engendre des contraintes supplémentaires, comme par exemple des problèmes de précision ou de placement des doigts, qui ne sont pas abordés dans l'article.

7.2.1 Description

Comme nous l'avons rappelé dans l'introduction, nous cherchons à développer une technique de manipulation de caméra tirant parti de la propriété de colocalisation, qui est très forte lorsque l'on utilise un écran tactile. Pour cela, nous avons mis au point une technique

de manipulation de caméra basée sur la sélection d'un *point d'intérêt*. Le deuxième point de contact est alors considéré comme un *point de contrôle*.

Le point d'intérêt est sélectionné par une opération de *picking* simple. Le système calcule alors la position 3D de ce point de contact, par une projection des coordonnées dans l'environnement 3D. Une fois cette opération effectuée, les mouvements du point de contrôle permettent de déplacer la caméra virtuelle de manière à ce que la colocalisation du point d'intérêt soit toujours assurée.

Les mouvements de caméra élémentaires qu'il est possible de réaliser à l'aide de cette technique sont :

- **Se rapprocher d'un point** Ce mouvement de caméra est réalisé en rapprochant le second point de contact du premier. Ainsi, les mouvements de la caméra virtuelle sont calculés de manière à ce que le rapprochement soit total (c'est à dire que la caméra se trouve face au point d'intérêt) lorsque le point de contrôle a rejoint le point d'intérêt sur l'écran. Il est également possible de s'éloigner en réalisant le mouvement inverse.
- **Tourner autour d'un point** Afin de réaliser ce mouvement de caméra, il suffit d'effectuer une rotation du point de contrôle autour du point d'intérêt. La caméra réalise alors une rotation autour du point d'intérêt, à la manière d'un *trackball* dont le centre est défini par l'utilisateur. L'angle de la rotation correspond à l'angle effectué par le point de contrôle autour du point d'intérêt. Il est à noter que le point d'intérêt peut ne plus être visible, à cause d'obstacles pouvant apparaître entre le point et la caméra.

Ces deux gestes sont tout à fait réalisables indépendamment ou simultanément. Il est alors possible de se rapprocher d'un point d'intérêt tout en tournant autour.

Afin de faciliter l'utilisation de ces deux gestes, un *widget* est affiché, visible sur la Figure 7.3, montrant la direction à suivre pour se rapprocher du point d'intérêt. Il est alors aisé de suivre cette direction.

7.2.2 Contraintes d'utilisation

Cette technique a été développée pour être utilisée sur un terminal mobile équipé d'un écran multi-points. Cependant, étant donné que le premier point de contact, le point d'intérêt, reste immobile tout au long de l'interaction, il est tout à fait possible d'utiliser "pseudo multi-points" décrit en Section 1.2.3.3.

Cette technique permet de contrôler quelques mouvements de caméra basiques mais essentiels, que sont la rotation autour d'un modèle 3D et le rapprochement vers un point particulier. Cependant, l'utilisation d'une telle technique sur un terminal mobile ne satisfait pas le principe d'AFFICHAGE, car elle nécessite de poser deux doigts sur l'écran, ce qui masque une grande partie de celui-ci. De plus, comme l'ont montré Moscovich et Hughes (2008), il est préférable, pour le contrôle de deux points séparés, d'utiliser ses deux mains. Ceci rend alors impossible une utilisation unimanuelle. L'utilisation devra alors se faire à l'aide des deux pouces par exemple.

7.3 Conclusion

Nous avons présenté dans ce chapitre quelques techniques de navigation ou de manipulation de modèles 3D tirant parti des écrans multi-points. Ces techniques possèdent l'avan-

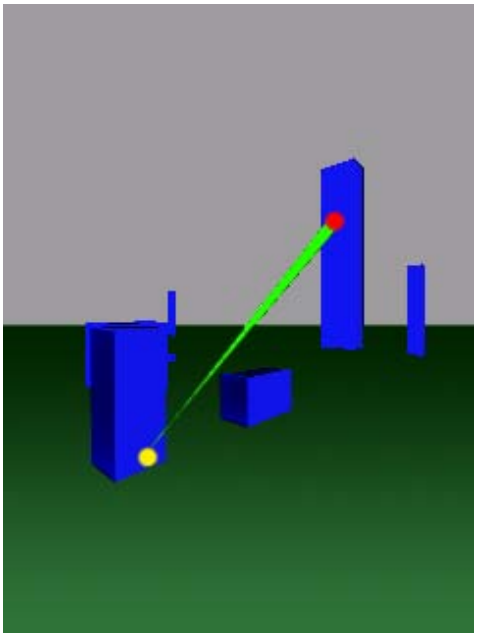


FIGURE 7.3 – Utilisation de plusieurs points de contact pour naviguer. Le *point d'intérêt* est en rouge, le *point de contrôle* est en jaune.

tage d'offrir à l'utilisateur un contrôle plus précis ou plus direct de l'environnement 3D, et permettent de contrôler plusieurs degrés de liberté simultanément. Cependant, leur utilisation masque une grande partie de l'écran, ce qui risque d'être gênant compte tenu de la taille réduite des écrans des terminaux mobiles.

Nous avons cherché à évaluer l'approche consistant à utiliser deux doigts pour des tâches de contrôle de vitesse. Nous avons donc mené une expérimentation, décrite dans le chapitre suivant.

CHAPITRE 8

Apport du sens proprioceptif lors d'une interaction à deux doigts

8.1 Introduction

L'emploi des deux mains pour interagir avec notre environnement est un geste naturel, que nous utilisons tous les jours. Il est par exemple extrêmement difficile de lacer une chaussure à l'aide d'une seule main. Les deux mains travaillent alors de façon coopérative, et sont toutes les deux assignées à une même tâche. Cependant, à part pour les personnes ambidextres, il peut être difficile de contrôler nos deux mains indépendamment.

Guiard (1987) a montré que pour des tâches de la vie courante, les deux mains travaillent de façon coopérative et asymétrique. Il a également montré que la main *non dominante* occupe un rôle important, car elle sert de référentiel spatial. Par exemple, la main non dominante tient le carnet à croquis d'un dessinateur, tandis que la main dominante est occupée à dessiner.

Cette particularité a été étudiée en ergonomie informatique, car elle est souvent employée afin d'accroître les performances des utilisateurs. Il est par exemple courant que la main *non dominante* manipule le clavier (afin d'utiliser des raccourcis clavier ou des *touches modifiantes* comme *Ctrl* ou *Shift*) tandis que la main *dominante* déplace la souris. Ceci est intensément utilisé par les joueurs, qui contrôlent les déplacements du personnage à l'aide des touches du clavier, tandis que la souris sert à contrôler la direction de vue.

Une première étude, menée par Buxton et Myers (1986), s'est intéressée à l'utilisation de deux mains pour une tâche simple de positionnement et de redimensionnement : la main gauche de l'utilisateur contrôlait une glissière afin de régler la taille de l'objet à manipuler, tandis que la main droite le déplaçait à l'aide d'une tablette graphique. Pour cette tâche, les sujets ont naturellement utilisé les deux mains simultanément et de manière continue. De même, Balakrishnan et Kurtenbach (1999) se sont concentrés sur l'apport de la main non dominante pour une tâche de contrôle de caméra : la main non dominante contrôle la caméra virtuelle, tandis que la main dominante s'occupe des autres tâches.

Plus récemment, Balakrishnan et Hinckley (1999) ont validé les principes énoncés par

Guiard et ont souligné le rôle du choix d'un référentiel dans des tâches bimanuelles, en insistant sur l'importance du retour visuel, qui aide l'utilisateur à comprendre le référentiel dans lequel il se trouve.

Il a ensuite été montré qu'il était plus efficace de limiter les contrôles bimanuels à des tâches *symétriques* (Balakrishnan et Hinckley, 2000), c'est à dire où les deux mains ont un rôle similaire. Il ressort ainsi que des facteurs tels que la *distance* séparant les deux curseurs, la *vitesse* ainsi que le *retour visuel* qui relie les deux points ont une influence sur les performances des utilisateurs : les performances diminuent lorsque les deux curseurs sont éloignés ou se déplacent à des vitesses différentes. De plus, le fait de relier visuellement les deux curseurs aide l'utilisateur à contrôler sa main non dominante, celle-ci se référant à la main dominante plus facilement.

Partant de cette affirmation, Latulipe *et al.* (2006) ont développé SymSpline, un outil de manipulation de courbes qui utilise ce principe de symétrie. De même, une manipulation *symétrique* d'images (Latulipe *et al.*, 2005) est bien plus performante qu'en utilisant une méthode simple, qui nécessite un changement de mode afin de translater ou de pivoter et redimensionner l'image à aligner.

L'utilisation de deux mains pour interagir avec une application a également été étudiée pour le contrôle d'environnements 3D (voir Section 5.3 du Chapitre 5). Ces études ont été menées dans un cadre de RV immersive, qui se prête bien à une utilisation des deux mains. Elles ont toutes conclu sur l'apport en terme de performances de la bimanualité [1]. Ce gain de performances est en partie dû à l'apport du *sens proprioceptif*.

Ce sens permet la perception, consciente ou non, de la position relative des parties du corps (Sherrington, 1911; Delmas, 1981). Il est à mettre en relation avec le *sens kinesthésique* dont Yves Guiard a souligné l'importance, qui met quand à lui l'accent sur le mouvement du corps.

Une étude de Mine *et al.* (1997) a étudié l'apport du sens proprioceptif pour le déplacement d'objets dans des applications de RV immersive.

Cependant, de telles études ont été menées dans des contextes d'application où l'utilisateur pouvait déplacer librement ses deux mains. Ce n'est pas notre cas, car l'utilisation typique d'un appareil mobile est soit unimanuelle, l'interaction se faisant à l'aide du pouce par exemple, soit bimanuelle, la main non dominante tenant l'appareil tandis que la main dominante s'occupe de l'interaction.

Dans notre contexte, il est donc plus judicieux d'étudier l'apport de l'interaction à l'aide de plusieurs doigts. Une étude récente, menée par Moscovich et Hughes (2008), a étudié les différences existantes entre l'interaction à deux mains et l'interaction à deux doigts d'une seule main (voir Figure 8.1). Leur étude révèle que l'utilisation de deux mains est recommandée pour des tâches où l'on contrôle explicitement deux points, et l'usage de deux doigts d'une seule main est préférable dans le cas du contrôle de positions, rotations et écartement des doigts. Une autre étude s'est intéressée au cas particulier de la sélection de plusieurs éléments (Kin *et al.*, 2009).

Comme nous l'avons vu, le sens proprioceptif et le sens kinesthésique aident l'utilisateur à interagir lorsqu'il utilise ses deux mains. Ces mêmes sens peuvent intervenir au niveau du

1. Le terme "bimanualité" est utilisé en médecine pour désigner l'utilisation des deux mains.

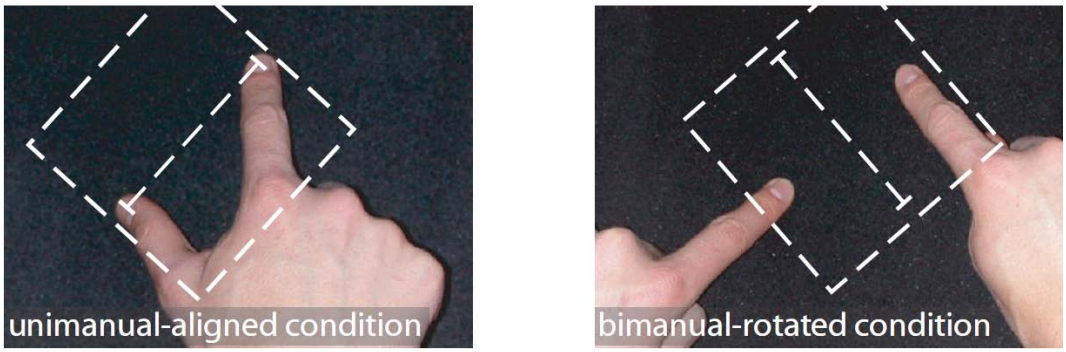


FIGURE 8.1 – Utilisation de deux doigts d'une seule main ou de deux mains (Moscovich et Hughes, 2008).

positionnement des doigts : le succès de l'iPhone d'Apple est entre autres dû à sa capacité à détecter la présence de plusieurs doigts sur l'écran. Cela a permis de développer des applications très intuitives. Nous pensons que cette approche peut également être utilisée pour interagir avec un environnement 3D, notamment pour des tâches de contrôle de vitesse.

Dans ce chapitre, nous proposons d'évaluer l'apport de l'utilisation de deux doigts pour le contrôle d'un degré de liberté. Dans ce cas, l'un des doigts sert alors de *référentiel*, de la même manière dont la main non dominante sert de référentiel à la main dominante. Nous discuterons ensuite de l'extension de ce principe au contrôle de deux degrés de liberté.

8.2 Contrôle d'une seule dimension

Les premières expérimentations que nous avons menées concernaient le contrôle d'une seule dimension. Dans des applications 3D interactives, cette dimension peut par exemple être la vitesse du déplacement de la caméra virtuelle, la taille d'un modèle, ou l'angle de rotation autour d'un axe.

Lorsque l'on utilise un écran tactile, le contrôle d'une dimension unique peut se faire à l'aide de mouvements *verticaux* ou *horizontaux*. C'est par exemple ce qui est employé dans les techniques de *flying* pour contrôler la vitesse de déplacement de la caméra.

Ces mouvements peuvent alors contrôler la *valeur* correspondant à cette dimension ; on parle dans ce cas d'une tâche de *contrôle de position* (*position control* en anglais). Si, par contre, les mouvements contrôlent la *vitesse de variation* de cette valeur, la tâche correspond à un *contrôle de vitesse* (*rate control* en anglais). Dans les deux cas, le paramètre contrôlé (la valeur ou la vitesse) dépend de la distance séparant le point de contact courant et un *point de référence*. Ce point, que nous nommerons P_0, correspond à une vitesse de variation nulle, ou à la valeur initiale du paramètre à contrôler.

Classiquement, ce point correspond au premier point de contact avec l'écran tactile lorsque l'utilisateur a initialisé son geste.

Nous proposons de comparer ce mode de contrôle avec une technique utilisant deux doigts, le pouce et l'index. La valeur de la dimension à contrôler dépend alors de la distance

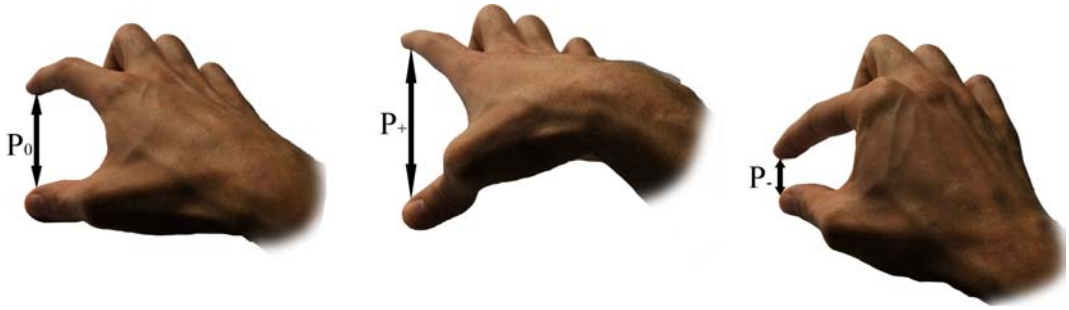


FIGURE 8.2 – Utilisation de deux doigts pour contrôler une dimension : position de référence P_0, écartement des doigts pour augmenter la valeur, rapprochement des doigts pour diminuer la valeur.

séparant les deux doigts. Le pouce, immobile, sert de point de repère pour l'index. La *position de référence* P_0 correspond à l'écartement initial des deux doigts. Le contrôle, indépendamment du type de tâche, se fait en écartant les doigts pour augmenter et en rapprochant les doigts pour diminuer la valeur contrôlée, comme le montre la Figure 8.2.

Pour cette étude, nous ne prenons pas en considération la taille réduite des écrans de terminaux mobiles. Nous prenons comme hypothèse que l'écran est assez grand pour que l'utilisateur puisse réaliser les gestes qu'il souhaite.

8.2.1 Étude 1 : Contrôle de position

Nous avons mené une étude préliminaire afin d'évaluer les différences de performances qui pouvaient exister entre un périphérique *élastique* (un Joystick), un périphérique *isotonique simple-point* et un périphérique *isotonique multi-points* pour une tâche de contrôle de position. Pour ce type de tâche, le joystick n'est théoriquement pas performant (Zhai, 1995). Par contre, les périphériques isotoniques devraient avoir de meilleures performances.

Notre hypothèse est que le fait d'utiliser deux doigts plutôt qu'un permet de "mémoriser" l'écartement des doigts, et ainsi de revenir plus facilement et plus rapidement à une position donnée. Les performances devraient donc être meilleures pour le périphérique isotonique multi-points.

Nous avons réalisé une étude visant à confirmer cette hypothèse. Ainsi, pour cette expérimentation, il était demandé aux utilisateurs de contrôler une valeur affichée à l'écran, et d'atteindre quatre *valeurs cibles*. Pour chaque sujet, elle a été réalisée à l'aide de trois interfaces : *joystick*, *simple-point*, multi-points. Le but de cette expérimentation étant de mesurer la capacité du cerveau à mémoriser des positions, les valeurs à atteindre étaient répétées cinq fois, afin d'évaluer les effets de l'apprentissage.

8.2.1.1 Description de la tâche

L'interface du programme est visible sur la Figure 8.3. La valeur courante est affichée à gauche de la fenêtre, la valeur cible est affichée à droite. Les valeurs peuvent aller de -100 à

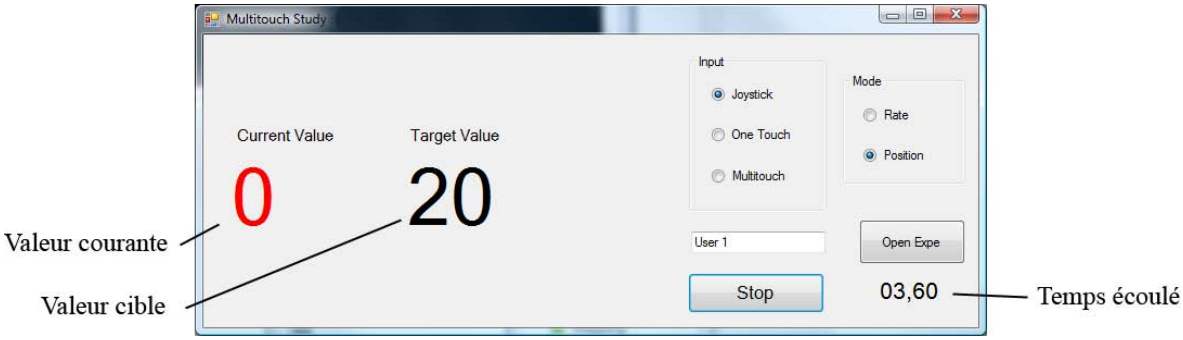


FIGURE 8.3 – Interface de l'expérimentation.

100, par pas de 1. Lorsque l'utilisateur est assez proche de la valeur cible (la tolérance était de ± 5), la valeur courante est affichée en vert. Sinon, elle est affichée en rouge. Une cible est considérée comme atteinte lorsque l'utilisateur reste pendant *une seconde* dans cette plage de tolérance. Le temps écoulé depuis le début de la tâche est affiché en bas à droite.

Le contrôle de la valeur était le suivant :

- Avec le joystick, la valeur correspondant au joystick "au repos", c'est à dire lorsqu'aucune pression n'est exercée, est de 0. Lorsque le joystick est poussé au maximum, la valeur est à 100, et à -100 lorsqu'il est tiré au maximum.
- Avec le simple-point, la valeur est déterminée par la distance entre le doigt de l'utilisateur et la position de référence. Une distance de 10 cm vers le haut (respectivement vers le bas) correspond à une valeur de 100 (respectivement de -100). Le choix de cette distance sera expliqué ultérieurement.
- Avec le multi-points, la valeur était calculée de la même manière que pour le simple-point. Il était cependant demandé à l'utilisateur de poser le pouce en premier, puis l'index. La distance entre le pouce et l'index est alors considérée comme distance de référence P_0. Nous avons constaté, lors de tests préliminaires, que la distance P_0 était généralement proche de 10 cm, ce qui nous a conduit à utiliser cette même distance comme distance maximale (à la fois pour le simple-point et pour le multi-point).

Pendant l'expérimentation, nous avons mesuré le temps nécessaire pour arriver à chaque valeur cible. Nous avons aussi enregistré la valeur courante toutes les 100 ms, afin de pouvoir visualiser l'évolution des valeurs au cours du temps pour chaque utilisateur.

Une tâche consistait à atteindre une série de 20 valeurs cibles. Étant donné que nous cherchons à évaluer l'effet d'apprentissage, cette série était composée d'une suite de quatre valeurs à atteindre (20, 60, 20, 0) répétée 5 fois.

8.2.1.2 Participants et matériel

Six membres de notre équipe ont participé à cette étude (5 hommes, 1 femme). Ils étaient tous droitiers, et avaient l'habitude d'utiliser les périphériques présentés. La durée totale nécessaire pour accomplir la tâche était de 15 minutes. Il aurait fallu, pour avoir des résultats plus fiables, doubler le nombre de participants. Cependant, les premiers résultats obtenus grâce à ces six participants ont permis de dessiner une tendance qui confirmait l'hypothèse de base, à savoir les mauvaises performances du périphérique élastique par

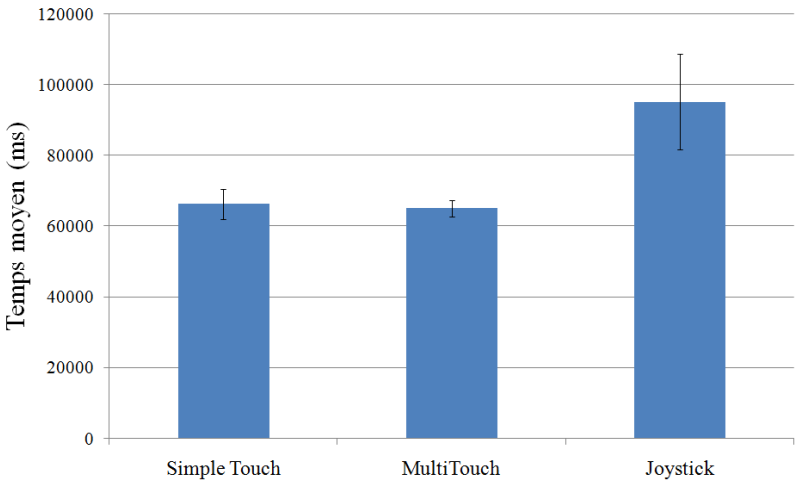


FIGURE 8.4 – Résultats de l'étude préliminaire.

rapport aux périphériques isotoniques.

Pour cette étude, nous avons utilisé un écran multi-points Stantum SMK. La dalle mesure 33 cm $\times$ 21 cm, sa résolution est de 1280 $\times$ 800 pixels. Le joystick utilisé est un WingMan de Logitech, muni d'une résolution de 32 valeurs possibles.

8.2.1.3 Résultats

Les Figures 8.4 et 8.5 présentent les temps moyens obtenus pour tous les participants, ainsi que les temps moyens pour atteindre les différentes valeurs cibles (20, 60, 20, 0).

L'analyse de ces figures suggère que le joystick est la moins performante des trois interfaces. En revanche, il n'y a pas de différence significative entre le simple et le multi-points. Un test PLSD de Fisher montre une différence significative entre le simple-point et le joystick ($p < 0.05$), entre le multi-points et le joystick ($p < 0.05$), mais pas entre le simple-point et le multi-points ($p = 0.9$).

L'analyse du graphique 8.5 nous fournit une information intéressante. Il nous montre qu'avec les techniques isotoniques, les utilisateurs ont, en moyenne, mis autant de temps pour aller vers une valeur élevée, que pour revenir vers une valeur basse. Par contre, ils ont eu plus de difficulté à aller vers la plus haute valeur avec le joystick. Ceci peut s'expliquer par l'élasticité du joystick qui rend difficile l'exercice d'une pression constante.

Les commentaires des utilisateurs ont permis de détecter les premiers problèmes de notre approche. Ils n'ont pas ressenti d'amélioration entre le fait d'utiliser un doigt ou deux. Ceci confirme les performances mesurées.

En revanche, utiliser les deux doigts a provoqué chez certain des douleurs, voire des crampes, au niveau du poignet ou de la main. Les raisons possibles à cela sont que la position de la main est différente entre les deux techniques, et que les muscles mis en oeuvre ne sont pas les mêmes : lorsque l'utilisateur n'utilise qu'un doigt, sa main est mue par la flexion de tout le bras ; le poignet est droit, et la main immobile.

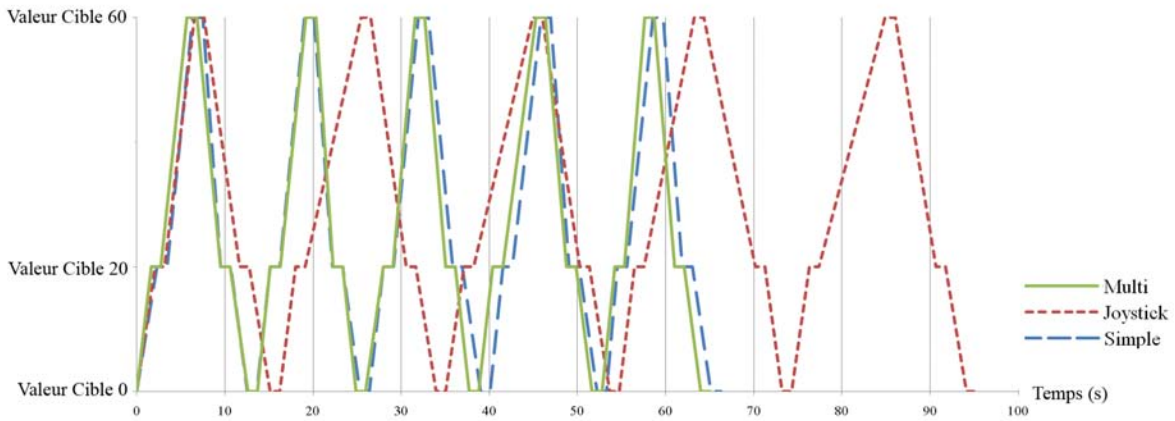


FIGURE 8.5 – Temps moyens pour atteindre les valeurs cibles.

FIGURE 8.6 – Mouvement effectué lors de l'utilisation de deux doigts.

Par contre, lorsque les deux doigts sont utilisés, le poignet est beaucoup plus "cassé", et les mouvements sont principalement dus au travail de muscles de la main tels que les *abducteurs et extenseurs du pouce* et les *fléchisseurs des doigts* (voir Figure 8.6). Ces extensions et contractions risquent de provoquer des douleurs, voire des crampes, lors d'une utilisation trop longue.

Ce dernier problème est réellement dérangeant, car il empêche une utilisation prolongée d'une telle technique. Néanmoins, nous avons cherché à évaluer si un apport existait tout de même lors de tâches de contrôle de vitesse.

8.2.2 Étude 2 : Contrôle de vitesse

Pour cette expérimentation, le protocole était le même que pour l'étude préliminaire. Cependant, au lieu de contrôler directement la valeur courante, les utilisateurs contrôlaient la *vitesse de variation* de la valeur courante. Ainsi, suivant la force exercée sur le joystick, la valeur variait par pas de 1 (pression faible), jusqu'à varier par pas de 10 (pression maximale). De même, pour les périphériques tactiles, la vitesse de variation était proportionnelle la longueur du déplacement (1cm correspondant à une variation de 1).

Dans ce cas de figure, les mouvements sont beaucoup plus importants que pour la tâche de contrôle de position, avec de nombreux aller-retour nécessaires pour ajuster précisément la valeur courante.

De même, il est nécessaire de revenir constamment à la position de référence, afin de rester à une valeur particulière. Ceci est bien entendu facile avec le joystick, car un relâchement suffit. Avec l'écran tactile en revanche, cette tâche est compliquée à cause du manque de repère. C'est sur ce point que nous pensons que l'utilisation de deux doigts plutôt qu'un seul permet d'améliorer les performances.

Malheureusement, les premiers sujets ayant participé à l'étude nous ont tout de suite fait part de la difficulté à réaliser les mouvements demandés. Les allers-retours à réaliser sur l'écran tactile demandaient une extension trop importante des doigts, même après un réglage approprié. De plus, il leur était difficile de maintenir les deux doigts appuyés en permanence, et ils préféraient relâcher les doigts afin d'annuler la vitesse de variation de la valeur, plutôt que de chercher à retrouver la position de référence.

8.2.3 Discussion

Au vue des résultats de cette étude, il apparaît que le contrôle d'un paramètre par l'écartement de deux doigts, le pouce et l'index en l'occurrence, doit être utilisé avec parcimonie, particulièrement lorsque l'amplitude du mouvement est grande. Lorsque ce mouvement doit être utilisé, il est préférable que ce soit pour une tâche ponctuelle, comme par exemple réduire les dimensions d'un objet, et non sur une tâche continue, comme la navigation libre.

L'utilisation de deux doigts sur l'écran d'un terminal mobile ne satisfait bien entendu pas le principe d'AFFICHAGE, car les deux doigts occultent alors une grande partie de l'écran. Cependant, de telles techniques peuvent être utiles lorsque l'on souhaite utiliser un terminal mobile comme *télécommande*, qui sert alors à interagir à distance. Dans ce cas, la scène 3D est visualisée sur un écran externe.

8.3 Contrôle de deux dimensions

La position des doigts utilisée dans l'expérimentation précédente, le pouce comme point de référence et l'index mobile, n'est pas propice à des déplacements verticaux ou horizontaux de l'index. En effet, cela impliquerait des mouvements difficiles à réaliser, et pouvant faire souffrir les muscles de la main. Il est donc préférable de ne pas demander à l'utilisateur de réaliser ce type de mouvement trop souvent. En revanche, certains mouvements semblent plus simples à réaliser, comme par exemple écarter les doigts, les rapprocher, ou effectuer une rotation autour du pouce.

Une étude similaire a été réalisée par Karlson *et al.* (2007) pour connaître les mouvements préférés des utilisateurs lors d'une utilisation unimanuelle d'un appareil mobile. Dans notre cas, nous cherchons à connaître les mouvements les plus confortables qui utilisent le pouce et l'index afin de contrôler deux degrés de liberté simultanément.

L'idée principale est d'identifier, parmi les mouvements que les utilisateurs peuvent réaliser sans peine à l'aide de deux doigts, deux degrés de liberté principaux.

8.3.1 Description des modèles

Nous avons réalisé des études pilotes afin de déterminer quels gestes, effectués à l'aide du pouce et de l'index, étaient facilement réalisables par les utilisateurs. Ces tests ont été réalisés auprès de quelques personnes de notre équipe, de façon informelle.

Il est ressorti de ces tests que trois gestes prédominaient :
- la rotation de l'index autour du pouce,
- l'écartement des doigts,
- la translation de l'index horizontalement.

De ces trois mouvements, visibles sur la Figure 8.7, nous avons identifié deux *modèles* permettant de contrôler 2 DDL simultanément : un modèle *polaire*, et un modèle *trapézoïdal*.

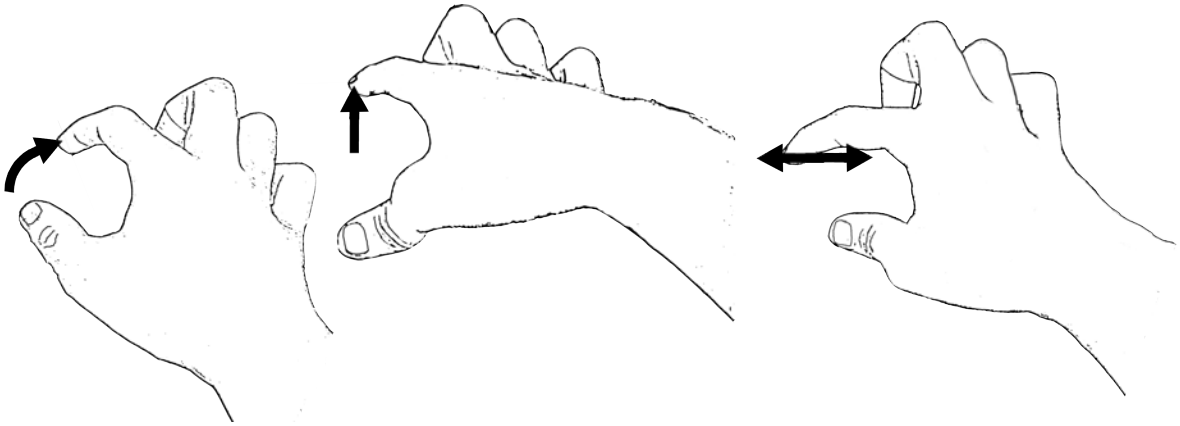


FIGURE 8.7 – Trois gestes réalisables à l'aide du pouce et de l'index : rotation, écartement, translation horizontale de l'index.

Le **modèle polaire** tire parti des mouvements d'écartement et de rotation décrits précédemment. Afin de contrôler 2 DDL, une dimension est alors contrôlée par l'écartement des doigts, l'autre par l'angle que forment les doigts par rapport à un *axe de référence*, par exemple l'axe vertical. La Figure 8.8 (gauche) présente les mesures associées à ce modèle. La surface accessible par l'index représente les limites accessibles par ce dernier en termes de rotation et d'écartement. La représentation ici présentée n'est qu'indicative, car cette surface, particulière à chaque individu, dépend de facteurs anatomiques tels que la taille et souplesse des doigts ou la taille de la main.

Les mouvements de rotation du modèle polaire peuvent être difficiles à réaliser par certaines personnes. Pour cela, nous avons également étudié un **modèle trapézoïdal**, qui favorise quant à lui les mouvements de translation. Dans ce cas, les 2 DDL à manipuler sont calculés en fonction de la distance horizontale Dx et verticale Dy séparant les deux points (voir Figure 8.8 (droite)). Dans ce cas, la surface accessible par l'index a la forme d'un trapèze.

8.3.2 Utilisation

Il est possible d'utiliser ces deux modèles pour des tâches de contrôle de vitesse. Dans ce cas, les points de référence sont à définir, et peuvent par exemple dépendre de la position initiale des doigts (angle, écartement, position spatiale). Une application du modèle polaire

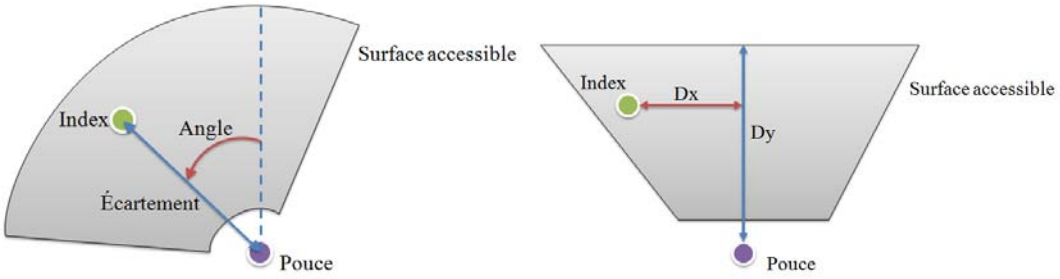


FIGURE 8.8 – Le modèle polaire (gauche) et le modèle trapézoïdal (droite).

à une tâche de navigation libre sera décrite dans la section 7.1.

L'utilisation de ces modèles pour une tâche de positionnement, celui d'un curseur par exemple, demande une attention particulière. En effet, une grande partie de l'écran n'est pas accessible par l'index si le pouce reste immobile. Il n'est donc pas possible d'utiliser la position de l'index sur l'écran pour positionner le curseur. De plus, le repère de l'écran est un repère *cartésien*, dont les coordonnées sont déterminées par la distance par rapport à l'origine du repère, tandis que les modèles opèrent dans un repère différent, avec des origines et des axes différents.

De ce fait, il est nécessaire d'effectuer une *correspondance* entre l'espace accessible par l'index et la surface de l'écran sur lequel le contrôle se fera.

Pour le modèle polaire, cette correspondance permet de faire un lien entre l'angle et la coordonnée X, et entre l'écartement et la coordonnée Y du curseur.

Ainsi, une abscisse nulle du curseur correspond à l'angle maximal vers la gauche que peut faire l'utilisateur, une abscisse maximale (en fonction de la résolution de l'écran) correspond à l'angle maximal vers la droite. De même, une ordonnée nulle correspond à l'écartement maximal des doigts, tandis qu'une ordonnée maximale correspond au rapprochement minimal des doigts de l'utilisateur.

Il en est de même pour le modèle trapézoïdal, les coordonnées X et Y dépendant alors des distances définies précédemment.

Nos premières expérimentations ont été réalisées sur des modèles dont la forme était générique. Ceci risque de ne pas être optimal, car chaque utilisateur possède une anatomie différente, car la taille et la souplesse des doigts n'est pas la même d'un individu à un autre.

Nous prévoyons d'utiliser une phase de *calibration*, où l'utilisateur définira, avant toute utilisation, l'espace accessible par ses doigts. La correspondance se fera alors en fonction des paramètres de chaque utilisateur. Une telle phase de calibration est utilisée dans ThumbSpace (Karlson et Bederson, 2007), afin de connaître la zone de l'écran que l'utilisateur atteint facilement.

8.4 Conclusion

Les travaux décrit dans ce chapitre ne sont pour l'instant qu'exploratoires. Nous avons cherché à identifier comment contrôler un, puis deux degrés de liberté simultanément à l'aide de deux doigts. Nous pensions que l'utilisation de deux doigts permettrait de favoriser

la mémorisation des positions importantes, comme la position de référence où la vitesse de variation du paramètre contrôlé est nulle. Il est cependant apparu que les mouvements que nous avons proposés n'étaient pas les plus adaptés, car ils pouvaient provoquer des douleurs dans le poignet et au niveau de certains muscles de la main lors d'utilisations prolongées.

Cependant, les premières expérimentations réalisées concernant le contrôle de deux dimensions sont encourageantes. Nous pensons qu'une telle approche peut être bénéfique pour un contrôle distant, où un écran mobile multi-points serait utilisé comme télécommande, l'environnement 3D étant affiché sur un écran externe.

CHAPITRE 9

Conclusion de la Partie II

Dans cette partie du mémoire, nous avons étudié l'interaction 3D basée sur un contrôle direct. Dans ce type d'interaction, l'utilisateur manipule l'environnement 3D de manière directe, c'est à dire que les actions qu'il effectue sont directement appliquées à l'environnement 3D. Il en résulte une grande interactivité, car le résultat d'un mouvement est visible immédiatement. Ce type d'interaction est utilisé dans la plupart des applications 3D temps réel tels que les jeux vidéos, la simulation 3D ou la visualisation de modèles 3D.

Cependant, lorsque l'on utilise un terminal mobile, certains problèmes peuvent survenir. Tout d'abord, la petite taille des écrans et l'utilisation du doigt pour interagir avec l'écran tactile créent de fortes occultations qui gênent l'utilisateur. De ce fait, la tâche à effectuer est plus ardue, car l'utilisateur peut masquer ce qui l'intéresse.

Afin d'éviter ces problèmes d'occultation, il peut être intéressant de considérer l'utilisation d'un terminal mobile comme *télécommande*. Dans ce cas, l'affichage est réalisé sur un écran externe. Ceci possède l'avantage de ne pas obstruer l'écran, et l'utilisateur reste alors concentré sur sa tâche. Les techniques de navigation basées sur l'utilisation de plusieurs doigts peuvent tout à fait utiliser ce mode de fonctionnement, car les sens proprioceptifs et kinesthésiques permettent de déplacer nos doigts "à l'aveugle", sans avoir besoin de voir ces derniers.

L'avènement des *picoprojecteurs*, des vidéo-projecteurs miniatures portatifs, permet d'imaginer de nouvelles façons d'interagir, où l'utilisateur projetterait son affichage sur n'importe quelle surface (mur, sol) et utiliserait son terminal pour interagir avec ce qui est projeté. Des téléphones portables intégrant ce type de projecteurs sont déjà à l'étude, et seront commercialisés très prochainement, notamment par Samsung, LG ou Tedacos.

La suite de ce mémoire est consacrée au contrôle *planifié* d'applications 3D. En effet, les problèmes d'occultation ou de précision évoqués plus haut peuvent rendre difficile l'utilisation de techniques directes. De plus, certains terminaux mobiles ont une puissance de calcul réduite, et le rendu en temps réel de scènes 3D complexes n'est pas toujours assuré sur ce genre d'appareil. Dans ces cas, il peut être plus efficace ou plus pertinent d'utiliser un contrôle *indirect*, c'est à dire où l'utilisateur "planifie" son mouvement.

Dans ce cas, le résultat de son action ne sera pas visible en temps réel, car l'interaction est effectuée à base d'*ordres* que l'utilisateur donne à l'application. De ce fait, l'interaction est souvent plus facile, car l'utilisateur spécifie ce qu'il souhaite obtenir, l'application étant alors

chargée de calculer les transformations nécessaire à l'obtention de ce résultat.

Nous verrons dans la partie suivante les différences existantes entre le contrôle direct et le contrôle planifié, ainsi que des techniques d'interaction 3D utilisant ce type de contrôle.

Troisième partie

Interaction 3D avec Contrôle Planifié

Résumé

Les terminaux mobiles possèdent des limitations qui peuvent rendre difficile le contrôle direct d'applications 3D. En effet, des paramètres tels que le nombre d'images par secondes ou l'occultation de l'écran par la main peuvent rendre les techniques d'interaction directe difficiles à utiliser, voire inefficaces. Dans cette partie, après une présentation des techniques d'interaction 3D planifiées existantes, nous étudierons les différences entre un contrôle direct et un contrôle planifié d'application 3D. Nous présenterons ensuite deux techniques de navigation 3D que nous avons développées : ZGoto, proposant un contrôle à l'aide des touches de l'appareil, puis Navidget, qui permet un contrôle avancé de la caméra virtuelle à l'aide d'un écran tactile.

CHAPITRE 10

Introduction et analyse

De nombreuses tâches d'interaction 3D peuvent être réalisées de façon *planifiée*, c'est à dire de manière à ce que l'interaction soit basée sur des *ordres* donnés par l'utilisateur. Ces ordres sont alors traduits par l'application en transformations dans l'environnement 3D. Ceci est en opposition avec le contrôle *continu*, que nous avons étudié dans la partie précédente, où le résultat des actions de l'utilisateur est visible en temps réel.

Une telle approche possède beaucoup d'avantages, particulièrement sur terminaux mobiles. En effet, de nombreuses contraintes des terminaux mobiles, comme la puissance de calcul limitée, la taille de l'écran ou la faible précision due à une utilisation des doigts sur l'écran tactile, peuvent rendre compliquée l'utilisation de techniques d'interaction continues.

Dans cette partie du mémoire, nous présenterons et étudierons des techniques d'interaction 3D qui utilisent un contrôle planifié.

Ce chapitre sera consacré à l'étude des techniques d'interaction 3D existantes qui utilisent un contrôle planifié. Ces techniques ayant pour la plupart été développées pour une utilisation sur un ordinateur de bureau, à l'aide d'une souris, leur possible adaptation aux terminaux mobile sera abordée. Ces différentes techniques seront présentées en fonction du type de tâche qu'elles opèrent. Nous présenterons ainsi tout d'abord des techniques de sélection de points 2D, de points 3D ou de modèles 3D. Puis, nous aborderons le problème de la manipulation de modèles 3D et du contrôle de l'application. Enfin, la navigation sera abordée dans la section 10.3.

10.1 Sélectionner

La sélection d'un élément de l'environnement 3D est une tâche primordiale. Cela permet de définir le modèle sur lequel travailler, d'isoler une partie du modèle afin de modéliser, ou encore de définir un point d'intérêt. Il est donc possible de sélectionner plusieurs types de données : un modèle, un sommet, une face ou une arête du modèle. Il est également utile de sélectionner un point sur l'écran qui servira, par une opération de projection, à retrouver la position 3D correspondant à ce point dans la scène.

Nous avons identifié deux principaux modes de sélection : la sélection *directe*, basée principalement sur l'opération de projection afin d'identifier l'élément situé sous le curseur par exemple, et la sélection *à base de liste*.

10.1.1 Sélection directe

Lorsqu'il est visible, le plus simple pour sélectionner un objet est de cliquer dessus. Cette technique, appelée *picking* en anglais, est le mode de sélection le plus intuitif et, de ce fait, le plus utilisé lorsque l'application 3D est contrôlée à l'aide d'une souris. Sur un terminal mobile équipé d'un écran tactile, cette technique de sélection semble encore plus directe, car l'apport de la colocalisation est ici très important. Cependant, l'utilisation d'une telle technique sur un terminal mobile pose quelques problèmes. En effet, si l'utilisateur se sert d'un doigt pour sélectionner un modèle ou un point, des problèmes de précision apparaîtront. De plus, si l'appareil ne possède pas d'écran tactile, il sera nécessaire de contrôler un curseur à l'aide des touches directionnelles. Ces points seront abordés dans la section 10.1.1.3.

Techniquement parlant, la sélection par *picking* peut être réalisée de deux principales façons : une méthode est *basée image*, c'est à dire qu'elle utilise les informations fournies par l'image courante, et la seconde est *basée géométrie*, c'est à dire qu'elle utilise des calculs géométriques pour déterminer l'objet ou le point sélectionné.

10.1.1.1 Sélection basée image

La sélection d'un élément dans l'image courante est une tâche essentielle, et est utilisée dans toutes les applications 3D. Nous différencions ici la sélection d'un modèle, ou d'une partie d'un modèle, et la sélection d'un point 3D de la scène.

Lorsque l'on souhaite sélectionner un **modèle**, ou une sous-partie d'un modèle, le but est d'identifier sur quel objet l'utilisateur a cliqué. Pour cela, on utilise un *tampon de sélection*. Cette technique consiste à effectuer une nouvelle passe de rendu au moment où l'utilisateur clique. Dans cette passe, les objets pouvant être sélectionnés ne sont pas visualisés avec leur apparence habituelle, mais le sont sans éclairage et sans ombrage, avec une couleur par objet sélectionnable, comme le montre la Figure 10.1. Le rendu est effectué dans un tampon non visible, appelé *tampon de sélection*. Il suffit alors de savoir quelle est la couleur située sous le curseur dans ce tampon afin d'identifier l'objet qui a été sélectionné.

Cette méthode possède l'avantage d'être facile à implémenter. Le tampon de sélection est disponible dans les versions actuelles d'OpenGL. Il n'est cependant pas disponible dans OpenGL|ES, mais il est facile d'effectuer soi-même le rendu de sélection. De plus, cette technique est uniquement basée "image", et ne nécessite pas de connaissances préalables des modèles 3D de la scène. Enfin, elle nécessite peu de calculs, mis à part le rendu de la passe supplémentaire. Cependant, cette passe de rendu est effectuée à faible résolution, ce qui réduit le temps de rendu.

Lorsque l'on souhaite sélectionner un **point 3D**, le *picking* se fait en utilisant le *tampon de profondeur*. En utilisant ce tampon, ainsi que les matrices de projection et de transformation du contexte courant, il est possible de retrouver la position 3D du point situé sous un point 2D quelconque de l'écran. Malheureusement, les spécifications d'OpenGL|ES interdisent la lecture du tampon de profondeur. Il n'est donc pas possible d'utiliser cette méthode sur un terminal mobile utilisant OpenGL|ES. Cependant, pour nos expérimentations, nous avons contourné les spécifications en modifiant le code de l'implémentation *Vincent* (Will.) afin de

FIGURE 10.1 – Rendu normal (gauche) et rendu du tampon de sélection (droite).

rendre possible une telle opération.

Le principal inconvénient de la sélection basée image est que la précision de la sélection n'est pas grande, car elle dépend entièrement de la résolution d'affichage. Il risque donc d'y avoir des ambiguïtés, particulièrement à l'approche des bordures des modèles. De plus, la précision de la sélection diminue avec la distance, car le tampon de profondeur ne possède pas une précision constante sur toute la profondeur de la scène (la précision est plus importante proche de l'utilisateur). Ces imprécisions risquent de produire des erreurs dans les sélections d'objets ou de points distants.

10.1.1.2 Sélection basée géométrie

La seconde méthode possible pour sélectionner un modèle ou un point 3D est d'utiliser un *lancer de rayon*, schématisé sur la Figure 10.2. Le principe ici est de calculer les intersections existantes entre un rayon défini par le point de vue et passant par le point 2D du curseur et tous les modèles de la scène, afin de déterminer quels objets sont intersectés, et de calculer les éventuels points d'intersection avec la géométrie du modèle intersecté (Bier, 1986). Le calcul se fera en deux phases :

1. **Test d'intersection**. Ce premier test sert à déterminer si une intersection existe avec un, ou plusieurs, des modèles de la scène. Pour ce premier test, il est courant d'utiliser des *volumes englobants*, sur lesquels les tests seront rapides à effectuer. Il est également conseillé d'organiser la scène de façon hiérarchique, afin de n'avoir à traiter qu'une partie des modèles : si un objet parent n'est pas intersecté, ses fils ne le seront pas non plus, il est donc inutile de les tester.
2. **Calcul de l'intersection**. Le test précédent permet d'obtenir une liste de modèles potentiellement intersectés. Afin de déterminer quels modèles sont réellement intersectés, ainsi que le point d'intersection, le calcul est réalisé sur les triangles de ces modèles. Il est à noter que des optimisations sont également possibles ici, par exemple en ne calculant pas d'intersection sur les faces cachées.

La recherche des objets intersectant est un processus complexe qui demande beaucoup de calculs et de ressources. De nombreuses optimisations ont été proposées, qui utilisent par exemple l'accélération matérielle (Batagelo et Wu, 2008).

La sélection basée sur la géométrie est beaucoup plus complexe à implémenter que la méthode utilisant le tampon de sélection ou le tampon de profondeur. Elle est en revanche beaucoup plus précise.

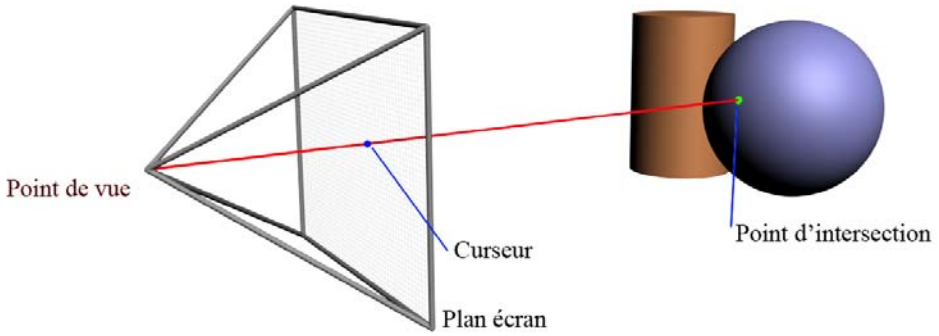


FIGURE 10.2 – Sélection directe par lancer de rayon.

10.1.1.3 Utilisation sur terminaux mobiles

Comme nous l'avons décrit dans la section 2, la sélection d'un point à l'aide d'un écran tactile peut souffrir d'un manque de précision, particulièrement lorsque la sélection se fait à l'aide d'un doigt. Des techniques telles que TapTap et MagStick (Roudaut *et al.*, 2008), Shift (Vogel et Baudisch, 2007) ou "Rubbing and Tapping" (Olwal *et al.*, 2008), décrites dans la partie introductive, peuvent aider à une plus grande précision. Si l'écran de l'appareil peut détecter plusieurs points de contact, il est alors possible d'utiliser des techniques basées sur l'emploi de plusieurs doigts (Benko *et al.*, 2006; Esenther et Ryall, 2006).

Certains terminaux mobiles ne sont pas équipés d'un écran tactile, mais seulement d'un clavier. La sélection d'un point à l'écran peut alors se faire en déplaçant un curseur à l'aide des touches de l'appareil. Cependant, le temps nécessaire pour atteindre la cible peut être long. Dans ce cas, il peut être bon d'utiliser une technique comme *Jump and Refine* (Hachet *et al.*, 2007), qui permet de sélectionner un point 2D plus rapidement, en proposant une sélection à deux niveaux : le premier niveau permet de déplacer rapidement le curseur sur une grille, tandis que le second niveau peut être utilisé pour le positionner précisément à l'intérieur de la zone sélectionnée (voir Figure 10.3).

10.1.2 Sélection à base de liste

Dans certaines situations, il peut être difficile de sélectionner un modèle en utilisant une sélection par curseur, notamment à cause de problèmes de précision. Dans ce cas, la sélection d'un modèle peut être effectuée à travers l'utilisation d'une *liste* répertoriant un ensemble d'objets 3D présents dans la scène. La sélection d'un élément en particulier est alors facilitée.

FIGURE 10.3 – La sélection à base de touches du Jump and Refine(Hachet *et al.*, 2007).

Les terminaux mobiles ont été créés pour traiter des tâches de bureautique tels que gérer un agenda, lire des courriers électroniques, lire des documents, etc. Parmi ces tâches, beaucoup sont semblables à la sélection d'un élément dans une liste. On peut donc penser qu'il sera aussi efficace d'utiliser de telles listes pour sélectionner un modèle 3D. Il y a cependant des contraintes importantes à prendre en compte.

10.1.2.1 Organisation de la liste

Il est tout d'abord important d'organiser et de nommer intelligemment la liste des objets à sélectionner. En effet, énumérer l'intégralité des modèles de la scène risque de produire une liste trop grande, dans laquelle la recherche d'un élément serait fastidieuse. Il est donc nécessaire d'effectuer un *tri* des éléments de la scène, afin de n'en afficher qu'un sous ensemble à la fois. Un tel tri parmi les objets 3D est fréquemment réalisé dans les moteurs 3D, notamment pour des raisons d'optimisation de performance : plutôt que d'envoyer à la carte graphique ou au processeur tous les triangles de la scène 3D, il est plus efficace d'effectuer un test rapide de visibilité afin de déterminer quels modèles sont visibles par l'utilisateur, et de n'envoyer que ces triangles là.

Il est alors possible de se baser sur une organisation similaire afin de ne présenter à l'utilisateur qu'une partie des objets de la scène, par exemple uniquement les objets visibles.

Il est aussi possible d'organiser les objets en fonction de critères géométriques, par exemple leur taille, leur couleur, leur forme.

Il est cependant possible qu'un tri ne suffise pas à rechercher efficacement un objet en particulier si la scène comporte un trop grand nombre d'objets. Il sera alors préférable d'organiser la scène de manière *hiérarchique*, afin que la recherche d'un élément soit basée sur une dichotomie, plutôt que sur une recherche linéaire. Cette organisation hiérarchique peut être directement issue de la scène 3D, si celle-ci est organisée en *graphe de scène*. Elle peut aussi être réalisée à la volée, en fonction de critères de l'utilisateur. Ceci nécessite néanmoins une analyse de la scène qui peut être fastidieuse.

10.1.2.2 Affichage de la liste

Il y a plusieurs façons de représenter une liste. La façon la plus simple, et la moins efficace, est d'utiliser un contrôle standard du système, énumérant les noms des différents éléments de la liste, comme le montre la Figure 10.4.

Ce mode n'est pas efficace pour plusieurs raisons. Tout d'abord, si beaucoup d'éléments sont affichés, le temps de recherche d'un élément particulier sera long, car l'utilisateur n'a qu'une vue *locale* et *restreinte* de la liste. Dans ce cas, il sera préférable d'adopter une stratégie où l'utilisateur peut visualiser à la fois les éléments voisins et avoir un aperçu des éléments les plus éloignés de l'élément courant. C'est ce que l'on appelle une visualisation *fisheye* (Furnas, 1986). Une telle approche a par exemple été étudiée pour l'affichage de menus (Bederson, 2000) (Figure 10.4).

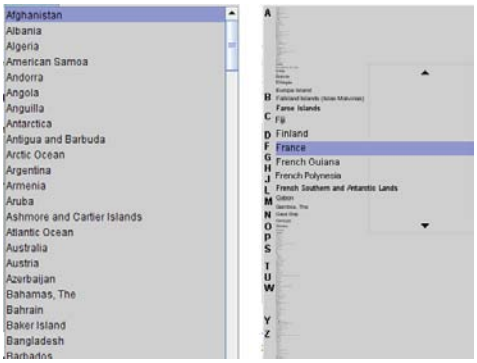


FIGURE 10.4 – Affichage d'une liste d'éléments : affichage standard (gauche) et affichage en *fisheye* (droite).

Un affichage purement textuel de la liste ne fournit que le nom de l'objet à sélectionner. Cela peut ne pas être le mode de représentation le plus adapté, car les modèles peuvent de pas avoir été nommés. Il est donc préférable d'afficher la liste d'une manière permettant d'avoir un aperçu du modèle sélectionné.

Il est un mode de présentation de données qui se prête bien à l'affichage d'une liste au sein de la fenêtre de visualisation 3D : le *carrousel*. L'affichage en carrousel, visible sur la Figure 10.5, organise les différents éléments d'une liste de manière circulaire. L'utilisateur voit alors l'élément courant ainsi que certains de ses voisins. Ce mode de représentation se rapproche des *fisheyes* présentés plus haut. Le rayon du cercle sur lequel sont déposés les éléments dépend de leur nombre. Il est ainsi possible d'afficher un grand nombre d'éléments, sans pour autant obstruer une trop grand partie de l'image (Wang *et al.*, 2005).

L'affichage de données en carrousel a été utilisé dans beaucoup d'applications, par exemple pour la sélection d'un album à écouter sur des baladeurs MP3 ou la sélection d'une image parmi un album photo.

Ce mode de représentation des données est parfaitement compatible avec un affichage de données hiérarchiques, les *Cone Trees* (Robertson *et al.*, 1991) en sont un exemple. De même, Patterson (Patterson, 2007) décrit deux autres agencements de données : le *Flow* et le *Circulatory*, qui tirent parti de la perception de la profondeur pour aider l'utilisateur dans sa tâche de sélection.

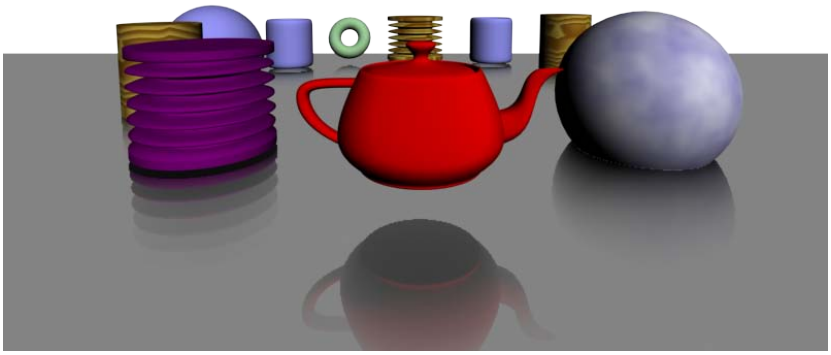


FIGURE 10.5 – Représentation en carrousel des objets de la scène.

10.2 Manipuler, Contrôler l'application

La manipulation de modèles 3D est une tâche complexe. En effet, plusieurs outils sont nécessaires (translation, rotation, redimensionnement), et chaque outil possède ses spécificités (plan ou axe de travail, point de pivot, etc.). Nous avons donc choisi de présenter ici des techniques de manipulation 3D et des techniques de contrôle d'application, car ces deux opérations sont bien souvent liées.

Un premier exemple de la liaison forte qui existe entre ces deux tâches concerne la *sélection du repère* de la transformation à appliquer. Dans beaucoup d'applications 3D, cette sélection peut se faire grâce à un menu ou à un bouton dans une barre d'outil. Il est également possible de sélectionner le repère grâce à l'utilisation de *widgets*. Cependant, comme nous l'avons vu dans la partie précédente, de tels contrôles peuvent être difficiles à utiliser sur un écran tactile lorsqu'ils nécessitent une trop grande précision.

Une approche intéressante, et qui convient bien à une utilisation à l'aide d'un écran tactile, est l'utilisation du *sketching* pour manipuler les objets 3D. Ces techniques traduisent les dessins et gestes effectués par l'utilisateur en ordre (Bimber *et al.*, 2000) ou en trajectoire (McCrae, 2008). Une telle approche a été utilisée pour plusieurs tâches, comme le contrôle de l'expression d'un avatar (Barrientos et Canny, 2002) ou des mouvements d'un personnage virtuel (Thorne *et al.*, 2004; Igarashi *et al.*, 1998) (voir Figure 10.6).

Schmidt *et al.* (2008) utilisent une telle méthode pour définir les axes de la transformation, ainsi que pour définir les points de pivot d'un modèle 3D. La Figure 10.7, présente un exemple d'assemblage de modèle 3D utilisant cette technique. A l'aide d'un geste "Cross" (a), l'utilisateur définit le point de pivot de l'objet à manipuler. Ensuite, un trait permet de définir le point d'accroche sur lequel placer la pièce et son orientation (b). Il est alors pos-

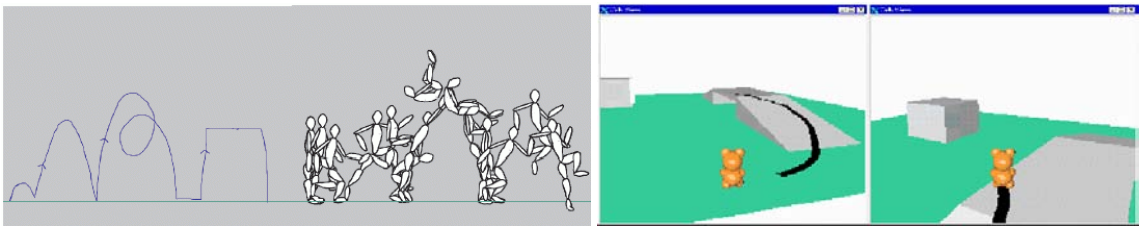


FIGURE 10.6 – Contrôle du déplacement de personnages : Motion Doodles (Thorne *et al.*, 2004) et Path Drawing (Igarashi *et al.*, 1998).

sible de modifier cette orientation en traçant un nouveau trait (c), pour finalement arriver au résultat attendu (d).

Cette approche présente l'avantage de ne pas surcharger l'écran, car peu d'informations sont affichées. Le *widget* ne possède que les fonctionnalités minimum, et n'occupe ainsi pas beaucoup de place à l'écran. De plus, l'utilisation de geste est tout à fait adaptée aux écrans tactiles. Enfin, le dessin de l'axe de transformation, ainsi que des différents gestes de commande, ne demande pas de précision particulière. Cette technique est donc tout à fait en accord avec le principe de **PRÉCISION** et d'**UTILISATION UNIMANUELLE**.

Par conséquent, ce genre de techniques est adapté aux terminaux mobiles équipés d'un écran tactile.

Cette approche nécessite néanmoins l'apprentissage des différents gestes qui permettent de contrôler l'application. Il faudra donc bien étudier les gestes indispensables, afin de ne pas perdre l'utilisateur devant un dictionnaire de gestes trop important.

La manipulation à proprement parler, c'est à dire l'utilisation de l'outil de transformation défini préalablement, peut également être effectuée de manière planifiée. Il est par exemple possible de définir la position, l'orientation et la taille d'un modèle de manière numérique, afin d'appliquer une transformation précise. Cette approche n'est cependant pas conseillée dans le cas où l'application est destinée aux novices, car elle nécessite une bonne connaissance des notions de transformation 3D et de repère de travail.

L'utilisation d'une technique basée sur le *pointage* peut en revanche être plus facile à utiliser pour un novice (Oh et Stuerzlinger, 2005; Zeleznik *et al.*, 1996). Dans ces approches, l'utilisateur sélectionne le point vers lequel il souhaite déplacer le modèle. Ce point étant sélectionné par une opération de *picking*, le placement des modèles se fait relativement aux autres modèles de la scène. Ceci permet un placement rapide, mais rend plus difficile le placement libre.

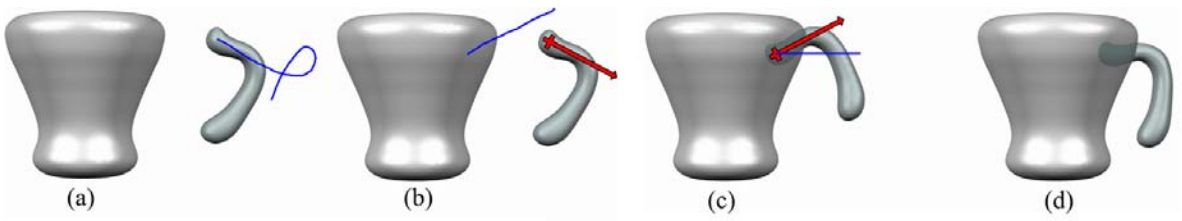


FIGURE 10.7 – Utilisation de *Sketching* pour la manipulation d'objets 3D(Schmidt *et al.*, 2008).

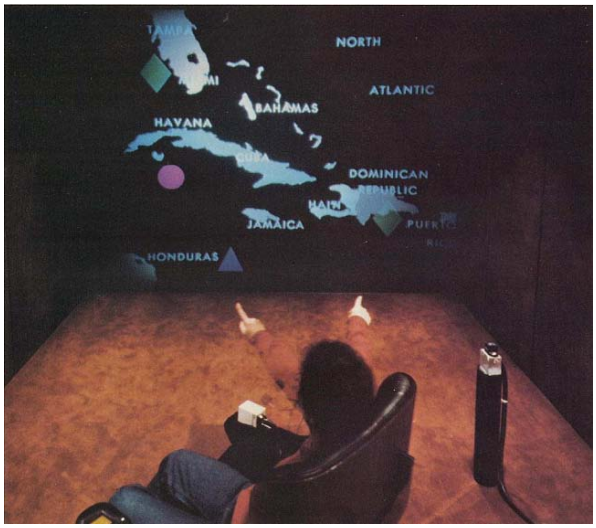


FIGURE 10.8 – Put-That-There (Bolt, 1980).

Concernant la sélection de l'outil à utiliser, plusieurs stratégies sont possibles. Richard Bolt (Bolt, 1980) a proposé, en 1980, un modèle d'interaction *multimodale*, qui utilise la voix et les gestes de l'utilisateur. Ainsi, celui-ci peut donner des ordres tels que "Mets *cet objet* ... *ici* !", tout en pointant l'objet en question puis sa position finale, afin de sélectionner et de déplacer des éléments graphiques (voir Figure 10.8). Une approche similaire a été utilisée récemment (Wang et van de Panne, 2006) pour le contrôle des déplacements et des actions de personnages, ainsi que pour des tâches de navigation simple, tels que zoomer vers un point ou tourner autour d'un personnage.

Les terminaux mobiles étant tous, ou presque, équipés d'un microphone, ils peuvent être adaptés à l'interaction *multimodale* basée aussi bien sur les gestes et la voix. Il est à noter qu'un système de reconnaissance vocale est présent sur beaucoup d'appareils mobiles, afin de composer le numéro de téléphone d'un contact en prononçant son nom par exemple. Il est également possible, avec de tels systèmes, de contrôler l'appareil. Cependant, l'utilisation d'un tel système peut déranger l'entourage de l'utilisateur et est peu propice à une utilisation collective à cause d'interférences pouvant survenir.

10.3 Naviguer

Le contrôle de la navigation de manière planifié peut se faire de plusieurs manières distinctes : à l'aide d'un *contrôle* de l'application, à base d'*esquisse*, ou par la sélection d'un *point d'intérêt*.

10.3.1 Navigation par l'application

L'utilisation d'un contrôle de l'application pour définir la vue permet, par exemple, de sélectionner une vue prédéfinie. Elle peut correspondre à une des vues canoniques (vue de face, de profil, de dessus, etc.) ou une vue oblique, mêlant deux ou trois de ces vues (par exemple une vue plongeante de trois-quarts est un mélange des vues de face, de profil et de dessus). Afin de sélectionner une de vues, il est possible de passer par un menu. Cependant,

FIGURE 10.9 – Le ViewCube (Khan *et al.*, 2008) permet de se déplacer vers des vues prédéfinies en sélectionnant une arête, une face ou un sommet du *widget*.

l'utilisation de menus n'est pas conseillée lorsque l'écran est de petite taille, car cela masquerait une trop grande partie de l'affichage. De plus, il n'est pas toujours simple de savoir à quelle vue se référer.

Lorsqu'un écran tactile est disponible, il vaut mieux, dans ce cas, utiliser un *widget* approprié, comme le ViewCube (Khan *et al.*, 2008). Ce contrôle, visible sur la Figure 10.9, permet de connaître l'orientation actuelle de la caméra 3D. Il permet également de contrôler la vue : il peut être tourné à la manière d'un *trackball*, et ses faces, arêtes et sommets peuvent être sélectionnés par l'utilisateur afin d'orienter la scène vers la vue correspondante.

Lorsque l'appareil n'est pas équipé d'écran tactile, il ne sera pas possible d'utiliser un *widget*. Dans ce cas, il sera préférable d'utiliser un menu ou de définir des *raccourcis clavier* permettant d'accéder facilement à ces vues.

10.3.2 Navigation par esquisse

L'utilisation d'**esquisse** pour naviguer correspond à une action habituelle pour l'utilisateur, par exemple pour montrer un chemin à suivre sur une carte, ou pour montrer une direction dans une image. Une approche similaire est possible, consistant à dessiner une trajectoire au sein même d'un environnement 3D (Hagedorn et Döllner, 2008; Knödel *et al.*, 2007; Igarashi *et al.*, 1998) afin de déplacer la caméra virtuelle une fois ce dessin achevé. Cette même approche a été utilisée pour déplacer un robot à l'aide d'un pda (Chronis et Skubic, 2003).

La navigation basée sur le dessin de trajectoire ou de gestes permet une interaction intuitive. Elle est à conseiller lorsque l'utilisateur possède un écran tactile. L'inconvénient principal d'une telle approche est que le calcul de la trajectoire peut être complexe, car plusieurs solutions sont possibles, et l'ambiguïté peut être difficile à résoudre.

10.3.3 Navigation par point d'intérêt

La troisième approche possible est de définir un **point d'intérêt** que le système utilisera pour calculer une trajectoire jusqu'à ce point. Cette approche a été introduite par Mackinlay et al.(Mackinlay *et al.*, 1990) et est maintenant implémentée dans de nombreuses applications de visualisation 3D. Elle porte le nom de *Goto*, *Zoom To Point*, *Point of Interest* (POI) ou *Seek*, et permet de se déplacer rapidement vers un point sélectionné sur le plan écran. Ce qui différencie les différentes techniques basées sur ce principe réside dans le choix de la distance

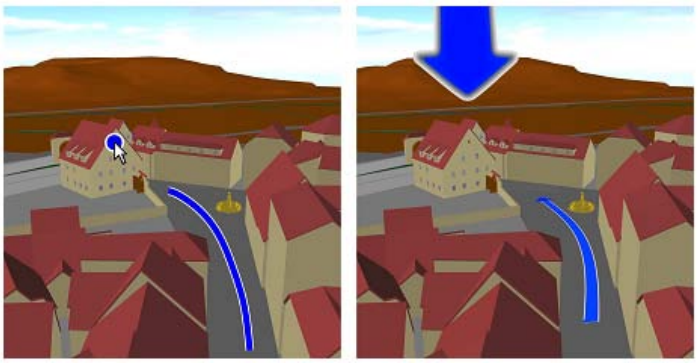


FIGURE 10.10 – Exemple de navigation basée sur l'esquisse : l'image de gauche montre le dessin effectué par l'utilisateur (la trajectoire et le bâtiment à sélectionner), l'image de droite montre l'interprétation faite par le système (Hagedorn et Döllner, 2008).

et de l'angle de visualisation du point d'intérêt, ainsi que dans la trajectoire à suivre. Ainsi, la technique de base de Mackinlay et al. aligne la caméra avec la normale au point d'intérêt, la distance d'arrêt dépendant de la distance initiale entre la caméra et le point d'intérêt. Afin de permettre à l'utilisateur de contrôler l'arrivée près du modèle, la vitesse de déplacement est logarithmique, ce qui produit une forte accélération au début du mouvement, puis une décélération à l'approche de la cible.

Unicam (Zeleznik et Forsberg, 1999) possède lui aussi la possibilité de se déplacer vers un point d'intérêt. Cette fonctionnalité, appelée "click-to-focus", place la caméra selon une vue *oblique* par rapport à la surface afin de permettre une meilleure visualisation de celle-ci. La distance d'arrêt peut être définie en fonction de la distance séparant le point d'intérêt et la caméra, ou bien en fonction de la taille du modèle sur lequel l'utilisateur a cliqué.

Il est également possible d'utiliser un *World In Miniature (WIM)* afin de déplacer la caméra (Stoakley *et al.*, 1995; Pausch *et al.*, 1995). Dans ce cas, l'utilisateur sélectionne son point d'intérêt dans une vue générale de son environnement. Cela permet de sélectionner des points non visibles, mais demande une plus grande précision. De plus, cela occupe une grande partie de l'écran, ce qui est contraire au principe d'**AFFICHAGE**.

Plus récemment, McCrae et al. (McCrae *et al.*, 2009) ont défini le concept de "navigation multi-échelle". Le principe repose simplement sur une technique de *point of interest* où la distance d'arrêt dépend de la distance séparant la vue et le point d'intérêt, afin de rendre la technique utilisable aussi bien pour de grandes distances que sur des trajectoires plus courtes. L'originalité de leur approche réside dans le calcul de la trajectoire, qui est effectué en fonction de l'environnement de vue actuelle, afin d'éviter toute collision.

D'autres approches ont été abordées, où les points d'intérêt et les vues correspondantes sont définis par le concepteur de l'environnement (Abásolo et Della, 2007; Haik *et al.*, 2002; Pierce et Pausch, 2004).

L'approche basée sur la sélection d'un point d'intérêt présente plusieurs avantages :

Indépendance du périphérique d'entrée. Il n'y a besoin que d'un périphérique de pointage, comme une souris, et d'un signal, comme l'appui sur un bouton de la souris, pour sélectionner le point d'intérêt et démarrer l'animation de la caméra. Il n'y a donc pas be-

soin d'autres périphériques (clavier, manette, ...). Par conséquent, une approche basée sur la sélection du point d'intérêt sera utilisable sur une grande variété d'appareils, allant des appareils mobiles (comme par exemple les téléphones portables ou les assistants personnels) aux grands écrans, dans le cas de travail collaboratif. En effet, l'interaction avec ce genre d'équipements ne se fait bien souvent que par du pointage. Par conséquent, ce type de techniques est particulièrement adapté pour la navigation 3D sur des systèmes alternatifs.

Facilité d'utilisation. Les techniques basées sur le pointage du point d'intérêt sont très simple et directes. Les utilisateurs n'ont qu'à sélectionner le point vers lequel ils désirent aller. Il n'y a pas de temps d'apprentissage, et toutes les parties visibles de la scène peuvent être atteintes directement.

Rapidité de déplacement dans la scène. La vitesse des mouvements de caméra au sein de l'environnement 3D sont entièrement contrôlés et paramétrables. Ceci permet de parcourir de grandes distances rapidement.

Peu de surcharge cognitive. Les cartes cognitives, qui jouent un rôle important dans le repérage dans l'espace et dans les tâches de recherche d'itinéraire, peuvent être difficiles à construire dans des applications 3D, particulièrement dans de grandes scènes et lorsque des appareils peu puissants sont utilisés, comme par exemple des appareils mobiles. Dans ce cas, le contrôle direct de la caméra doit être évité, car le délai entre l'action de l'utilisateur et le mouvement de caméra correspondant peut être long et devenir perturbant. L'approche "go to" permet d'éviter ce désagrément car l'utilisateur sélectionne sa cible et attend que le mouvement de la caméra soit terminé. Dans ce cas, le délai entre l'action de l'utilisateur et la réaction du système est moins ennuyeux, car aucun retour immédiat n'est nécessaire.

Cependant, les techniques basées sur la sélection du point d'intérêt possèdent deux défauts majeurs :

Dépendant de la surface. Le point cible vers lequel la caméra va se déplacer est toujours un point appartenant à une surface. Par conséquent, les seuls mouvements de caméra possibles sont des déplacements vers une surface de la scène 3D. Ceci est une restriction importante. Par exemple, si plusieurs objets sont affichés, il n'est pas possible de se déplacer de façon à avoir une vue proche de tous les objets à la fois.

Contrôle limité. Les utilisateurs sont limités à spécifier le point vers lequel ils souhaitent se déplacer. Ils ne contrôlent pas comment y aller. Plus particulièrement, ils ne contrôlent pas les paramètres tels que la distance entre le point d'arrivée de la trajectoire de la caméra et le point sélectionné sur la surface, ni l'angle sous lequel observer le point cible. Ces paramètres sont généralement fixés ou calculés automatiquement, en fonction du type de scène affiché. Ceci amène souvent à des situations bloquantes, comme par exemple se retrouver en face d'une surface qui occupe l'intégralité de la vue.

Malgré ces deux défauts, les techniques de navigation basées sur la sélection du point d'intérêt sont souvent utilisées car elles sont bien souvent les techniques les plus faciles à utiliser. Ces techniques sont implémentées dans la plupart des applications 3D interactives, et sont utiles aussi bien aux novices qu'aux experts.

10.4 Bilan

L'interaction 3D par contrôle planifié, à laquelle cette partie du mémoire est consacrée, représente une alternative intéressante au contrôle direct. En effet, dans ce mode d'interaction, le contrôle est beaucoup plus centré sur la tâche à accomplir que sur les moyens à mettre en oeuvre pour y arriver. Ceci crée une interaction d'un plus haut niveau qu'avec un contrôle direct, dont les techniques d'interaction sont plus souvent réservées à des utilisateurs experts. L'utilisateur reste alors concentré sur sa tâche, et non sur les outils dont il devra se servir pour parvenir à l'achever.

Bien qu'offrant une liberté de mouvements plus faibles que les techniques qui utilisent un contrôle direct, les techniques d'interaction 3D qui utilisent un contrôle planifié permettent aux utilisateurs d'effectuer la plupart des tâches d'interaction 3D de manière rapide et concise.

Le chapitre suivant est consacré à la comparaison du contrôle direct et du contrôle planifié. Nous y étudierons la faisabilité de la planification d'une technique essentielle de l'interaction 3D : le *trackball*. Ensuite, dans le chapitre 12, nous présenterons une technique de navigation que nous avons développée qui est basée sur la sélection d'un point 3D à l'aide de touches. Le chapitre 13 présentera Navidget, une technique de placement de caméra basée sur l'esquisse.

CHAPITRE 11

Étude comparative du contrôle direct et du contrôle planifié sur un terminal mobile

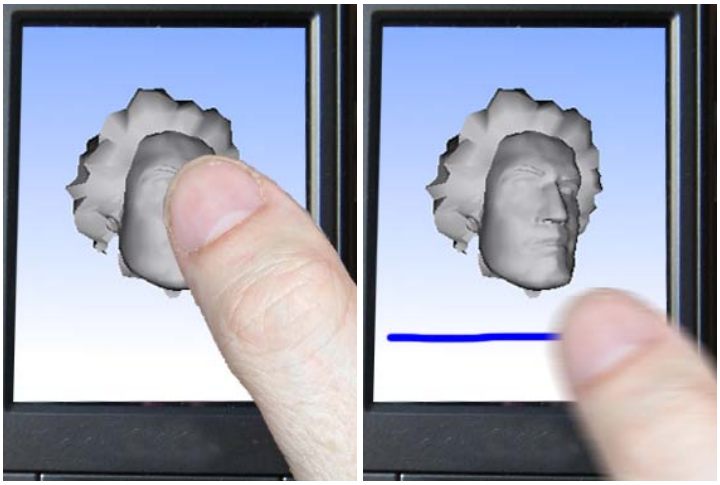


FIGURE 11.1 – L'utilisation d'un *trackball* standard occulte une partie importante de l'écran (gauche).
Cette même technique, utilisée de façon planifiée, améliore la visualisation du modèle (droite).

11.1 Introduction

Les limitations et les contraintes des terminaux mobiles font qu'il est parfois préférable de contrôler une application 3D de manière *planifiée* plutôt que *directe*. En effet, un contrôle planifié possède des avantages par rapport à un contrôle direct, particulièrement sur terminaux mobiles : les occultations de l'écran sont moins présentes et moins pénalisantes, l'expertise nécessaire est la plupart du temps moindre et cette approche est bien adaptée aux appareils peu puissants. Cependant, le contrôle peut être moins précis, et l'utilisateur est moins libre qu'avec un contrôle direct.

Nous avons cherché à identifier les différences qui existaient entre ces deux modes d'interaction, afin d'évaluer les situations dans lesquelles une approche peut être préférable à l'autre.

Dans ce chapitre, nous étudierons les différences existantes entre une interaction planifiée et une interaction directe. La suite du chapitre sera consacrée à une étude utilisateur que nous avons menée (Decle et Hachet, 2009) concernant la planification de la technique du *trackball*. Il ressort de cette étude que les performances des deux approches (*trackball direct* et *trackball planifié*) sont proches, mais que les utilisateurs se sont sentis aidés et moins perdus à l'aide de la technique de *trackball* planifiée.

11.2 Contrôle planifié et contrôle direct

Comme nous l'avons vu dans le chapitre 10, le contrôle planifié est une approche couramment utilisée dans de nombreuses tâches d'interaction 3D. La plupart des tâches d'interaction 3D peuvent être contrôlées de manière planifiée ou de manière directe, mais les performances et les contraintes d'utilisation ne seront pas les mêmes car des différences significatives existent entre ces deux modes d'interaction.

Tout d'abord, les techniques directes permettent souvent une plus grande liberté que les techniques planifiées. En effet, l'utilisateur est moins restreint, car il peut spécifier lui-même de nombreux paramètres qui seraient calculés automatiquement dans le cas du contrôle planifié. De plus, les techniques d'interaction planifiées se basent parfois sur une *discrétisation* de l'interaction, comme nous l'avons fait pour le contrôle planifié du *trackball*, décrit dans la section suivante, ce qui n'est pas le cas des techniques directes.

Cependant, l'utilisation d'un contrôle direct sur un écran tactile crée un problème important : une grande partie de l'écran est occultée par le dispositif de pointage servant à toucher l'écran (stylet, doigt, etc.). Cette limitation est particulièrement gênante sur un terminal mobile, où la taille de l'écran est réduite, car l'utilisateur masquerait alors une grande partie du modèle 3D qui l'intéresse.

Un autre problème risque de survenir si l'appareil ne possède pas assez de ressources pour pouvoir afficher une scène 3D en temps réel. Dans ce cas, le temps de latence séparant l'action de l'utilisateur et le résultat à l'écran peut être grand, à cause d'un temps d'affichage d'une image trop important. Prenons l'exemple d'un utilisateur souhaitant effectuer un quart de tour à la caméra avec une technique classique, telle le *flying*. Dans cette technique directe, l'utilisateur contrôle la vitesse de rotation de la caméra par des mouvements horizontaux : plus l'utilisateur s'écarte de l'axe vertical de l'écran, plus la caméra tourne vite. Si la latence est grande, le risque ici est que l'utilisateur continue de s'écarter de l'axe médian, augmentant la vitesse de rotation. Lorsque la nouvelle image sera affichée, l'angle de rotation sera trop grand et la rotation sera plus importante que prévue. Il risque alors de se perdre dans l'environnement. Avec un contrôle planifié, l'utilisateur est assuré que la caméra effectuera une rotation de l'angle qu'il a spécifié.

Ce sentiment d'égarement peut également provoquer un désagrément important : dans le cas de la navigation par exemple, des mouvements de caméra trop brusques et incontrôlés risques de provoquer des nausées. Avec les techniques planifiées, les mouvements de caméra sont calculés par le système. Elles sont de ce fait parfaitement fluides, sans à-coups.

Les techniques d'interaction qui utilisent un contrôle direct demandent souvent une certaine expertise. En effet, il est recommandé d'avoir de bonnes connaissances en manipulation

3D afin d'utiliser ces techniques, car elles sont souvent basées sur une décomposition de la tâche en fonction des degrés de liberté invoquée. Par exemple, beaucoup de logiciels de visualisation 3D utilisent une technique de *flying*, une technique de *trackball* et une technique de *panning* afin de permettre un déplacement totalement libre dans la scène 3D. De ce fait, il est nécessaire pour l'utilisateur d'apprendre plusieurs techniques afin d'arriver à ses fins. Cette décomposition de la tâche n'est pas adaptée à une utilisation par des novices.

Avec les techniques planifiées, la tâche est simplifiée et, surtout, elle est d'un plus haut niveau : plutôt que de contrôler les transformations à effectuer, l'utilisateur spécifie ce qu'il souhaite obtenir, afin que le système calcule les transformations invoquées afin d'arriver à ce résultat.

11.3 Étude de cas : Adaptation du *trackball* aux contraintes des appareils mobiles

Observer un modèle 3D dans son intégralité, faisant pivoter la caméra autour de celui-ci, est une tâche fondamentale. Ce mouvement de caméra permet à l'utilisateur de comprendre la structure du modèle 3D affiché à l'écran. Parmi les techniques de manipulation de caméra existantes, la technique du *trackball*, dont nous avons parlée dans la section 5.2, est la plus propice à ce genre de tâches. Elle permet d'accomplir des rotations autour d'un modèle 3D de manière efficace et est implémentée dans tous les logiciels de visualisation 3D. Malheureusement, elle a été développée et évaluée pour des ordinateurs de bureau, où elle est contrôlée à l'aide de la souris.

Les contraintes des appareils mobiles rendent cette technique plus difficile à contrôler, particulièrement lorsque l'on utilise le pouce afin d'interagir. En particulier, l'occultation causée par le pouce crée une situation contradictoire où l'utilisateur cache le modèle 3D qu'il est en train d'observer (voir Figure 11.1, gauche). De plus, des contraintes anatomiques ainsi que des problèmes de précision rendent le *trackball* plus difficile à contrôler au pouce qu'à la souris.

Afin de résoudre ces problèmes, nous proposons de contrôler le *trackball* de manière planifiée, à l'aide de gestes horizontaux ou verticaux dessinés par l'utilisateur (voir Figure 11.1, droite). Nous avons ainsi développé une version planifiée de la technique "Two-Axis Valuator", qui est considérée comme le *trackball* le plus efficace (Bade *et al.*, 2005) et qui est implémentée dans de nombreuses applications 3D.

Ainsi, dessiner un geste *horizontal* fait tourner la caméra autour de l'axe défini par le centre du modèle et le vecteur *up* de la caméra, tandis qu'un mouvement *vertical* produira une rotation autour de l'axe défini par le même centre et le vecteur *right* de la caméra (voir Figure 11.2).

Une fois le geste de l'utilisateur terminé, c'est à dire lorsqu'il relâche son pouce de l'écran, la caméra est transformée en fonction du geste effectué. Il est ainsi possible de tourner autour du modèle en dessinant plusieurs gestes successifs.

L'angle de la rotation entre deux vues successives doit être réglé en fonction de la précision requise. Un angle trop petit produit un grand nombre de vues possibles. Dans ce cas, beaucoup de gestes sont nécessaires. Parallèlement à cela, un angle trop grand réduit le nombre de gestes nécessaires à l'observation d'un modèle. Cependant, cela limite aussi le nombre de vues obtenues. L'utilisation de cette technique nous a montré qu'un angle de 45

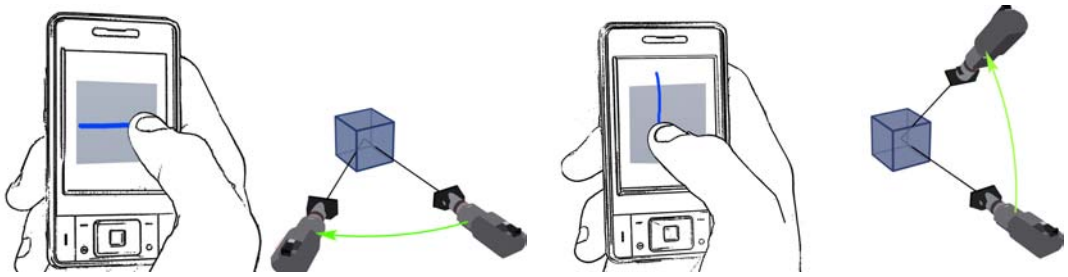


FIGURE 11.2 – Un geste horizontal produit une rotation autour du vecteur "up" de la caméra (image de gauche). Un geste vertical produit une rotation autour du vecteur "right" de la caméra (image de droite).

degrés était un bon compromis. Il permet de bien observer le modèle 3D, et ne requiert pas un grand nombre de gestes.

La durée de l'animation de rotation dépend du temps mis par l'utilisateur pour tracer son geste : plus le geste a été dessiné vite, plus l'animation sera rapide. Nous n'avons pas autorisé le dessin de gestes diagonaux, qui auraient produit une rotation combinant une rotation verticale et une rotation horizontale, afin de suivre les recommandations de Karlson *et al.* (2007) sur l'interaction mobile à l'aide du pouce.

Le *trackball* planifié que nous proposons nécessite une discrétisation des vues possibles. Par conséquent, il n'est pas adapté à des tâches nécessitant un placement et une orientation précise. Notre approche est en accord avec la philosophie du *sketching*, où des résultats simples mais imprécis sont obtenus grâce à des commandes simples et rapides. Dans le contexte des applications 3D interactives sur terminaux mobiles, nous pensons qu'un positionnement précis est rarement nécessaire. Des approches produisant des résultats approximatifs semblent mieux adaptées tant qu'ils permettent à l'utilisateur de bien comprendre la structure 3D des objets qu'il manipule.

Cette approche consistant à effectuer des mouvements verticaux et horizontaux simples à l'aide du pouce permet à l'utilisateur d'observer un modèle 3D facilement. Elle est inspirée des interfaces utilisateur actuelles (comme celle de l'iPhone par exemple) où l'utilisateur contrôle son appareil à l'aide de quelques gestes simples.

11.4 Évaluation du *trackball* planifié

La version planifiée du *trackball* permet d'observer facilement et rapidement un modèle 3D à l'aide de quelques gestes. Cependant, une telle approche discrétisée réduit l'aspect direct et continu de l'interaction, ce qui risque d'affecter les performances des utilisateurs lors de tâches d'observation de modèles 3D. Nous avons ainsi mené une expérimentation visant à évaluer l'influence de l'aspect continu du contrôle pour la technique de *trackball*.

11.4.1 Procédure

L'unique tâche de cette expérimentation consistait à compter le nombre de *motifs* (des "smileys" jaunes) dessinés sur certaines faces d'un modèle 3D, visible sur la Figure 11.3. Il

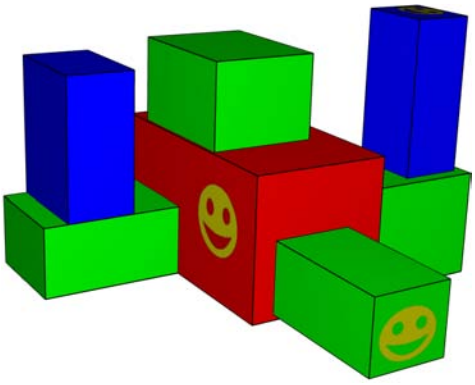


FIGURE 11.3 – Le modèle 3D utilisé pour la tâche expérimentale.

était alors nécessaire d'observer le modèle 3D sous tous les angles afin de trouver les motifs. Nous avons demandé aux sujets de réaliser la tâche deux fois : une fois avec le *Trackball Direct* (TD) et une fois avec le *Trackball Planifié* (TP) que nous avons proposé, avec un angle fixé à 45 degrés.

Avant de commencer l'expérimentation, les sujets ont pu essayer les deux techniques. Puis, nous leur avons demandé de compléter la tâche le plus vite possible avec chacune des deux techniques. L'expérimentation durait environ 10 minutes par participant. Un essai commençait dès que le modèle 3D était chargé sur l'appareil mobile, et terminait dès que le participant donnait sa réponse. Nous avons ainsi enregistré l'exactitude de la réponse de l'utilisateur afin de calculer le *taux d'erreur* ($TauxErreur = \frac{NombreMauvaisesReponses}{NombreQuestions}$) , ainsi que le temps nécessaire à l'accomplissement de la tâche (temps séparant le premier contact de l'utilisateur sur l'écran et le moment où il donnait sa réponse). Après l'expérimentation, nous avons demandé aux participants de remplir un questionnaire.

Les participants devaient réaliser six essais par technique. Le même modèle 3D était utilisé pour toutes les tâches, mais six distributions de motifs différentes étaient utilisées. Nous avons organisé les six distributions en deux *séries*. L'ordre de la technique ainsi que la série utilisée étaient alternées d'un sujet à l'autre. La seule variable indépendante était la *technique* (Trackball Direct TD ou Trackball Planifié TP).

11.4.2 Matériel et participants

L'expérimentation a été conduite sur un PDA Asus P535, muni d'un écran de 44 $\times$ 60 mm et d'une résolution de 240 $\times$ 320 pixels. La résolution effective était alors de 5.33 pixels / mm. Le taux de rafraîchissement moyen était de 15 images par secondes. Dix sept personnes (14 hommes et 3 femmes), âgées de 24 à 50 ans (moyenne 28 ans) et habitués à l'interaction 3D ont participé à l'étude. Tous les sujets étaient droitiers et avaient l'habitude d'utiliser des téléphones mobiles. Afin de les motiver, nous avons récompensé le participant qui réalisait le taux d'erreur le plus bas en un minimum de temps. Nous avons donc 6 *essais* * 2 *techniques* * 17 *participants* = 204 mesures.

11.5 Résultats et discussions

Les résultats obtenus sont de deux ordres : des **performances mesurées**, à savoir le temps nécessaire à l'accomplissement de la tâche et le taux d'erreur, et des **commentaires subjectifs** résultants des réponses données au questionnaire et des commentaires libres donnés par les sujets.

11.5.1 Performances mesurées

Un test t de Student apparié n'a pas révélé de différence significative entre les taux d'erreur moyens des deux techniques (TD : 53% / TP : 49%). Ce fort taux d'erreur peut être expliqué par le fait que, dans les deux conditions, les sujets avaient tendance à compter certaines cibles deux fois, ce qui rappelle que comprendre la structure d'un modèle 3D est une tâche difficile.

En revanche, la comparaison des temps nécessaires à l'accomplissement de la tâche, a permis de montrer que les sujets ont été plus rapides avec le contrôle direct qu'avec le contrôle planifié (test t de student apparié : Moyenne des temps TD : 106 164 ms / TP : 125 241 ms , $t_{(16)}$ = -2.180, $p < 0.05$). Cette différence peut être expliquée par la durée de l'animation qui intervient après le geste de l'utilisateur.

11.5.2 Commentaires subjectifs

Une échelle de Likert à 5 niveaux (1 = Pas du tout d'accord, 5 = Tout à fait d'accord) a été utilisée pour le questionnaire. L'analyse des réponses données par les utilisateurs nous a permis de connaître les impressions des utilisateurs à propos des deux techniques, *Trackball Direct* et *Trackball Planifié*. Afin de comparer les réponses des sujets, nous avons utilisé un test de Wilcoxon.

Il n'y a pas eu de différence concernant l'utilisabilité générale des techniques (TD : 3.06, TP : 3.68).

De même, les sujets ne se sont pas sentis plus rapides (TD : 3.37, TP : 3.68), plus libres (TD : 3.93, TP : 3.18), ou plus précis (TD : 2.5, TP : 3.6) en utilisant le contrôle direct qu'en utilisant le contrôle planifié.

Les participants se sont en revanche sentis perdus lorsqu'ils utilisaient la technique de Trackball Direct. L'analyse du questionnaire révèle une différence significative à la question "Je ne me suis pas senti perdu" (TD : 1.2, TP : 3.4 ; test Z de Wilocxon : Z = -3.319, $p < 0.001$). Cette différence peut être expliquée par le fait qu'un contrôle planifié aide l'utilisateur à structurer les mouvements.

De plus, l'approche planifiée permet à l'utilisateur de savoir exactement quels mouvements sont effectués : ainsi, en effectuant deux gestes successifs dans la même direction, l'utilisateur sait qu'il obtiendra une rotation de 90 degrés. Il peut alors revenir à l'orientation originale en effectuant l'opération inverse. Ceci limite la désorientation.

Enfin, les sujets ont rapporté que l'occultation de l'écran par le pouce était perturbante pour le Trackball Direct, ce qui n'est pas le cas pour le contrôle planifié (TD : 1.6, TP : 3.6 ; test

Z de Wilocxon : Z = -3.093, $p < 0.005$). Avec le contrôle direct, les sujets devaient fréquemment écarter leur pouce de l'écran afin d'observer le modèle.

11.6 Adaptation d'autres mouvements de caméra

Nous avons dans ce chapitre évalué la planification de la technique du *trackball*. Cette planification peut tout à fait être réalisée pour d'autres mouvements de caméra, comme par exemple la **rotation égocentrique** de la caméra : un geste fait alors tourner la caméra dans la direction désignée par le dessin, avec un angle fixé par l'application.

L'angle de la rotation peut être calculé de manière à ce que la rotation s'arrête lorsque la caméra possède une certaine orientation. Ainsi, pour les rotations égocentriques de la caméra, la rotation peut par exemple s'arrêter dès que l'angle que forme la caméra avec un des axes est un multiple de 45 : cela permet "corriger" un mouvement imprécis obtenu par une autre technique. De plus, avec un angle de 45 degrés, un point situé sur un bord de l'écran se retrouvera, après la rotation, au centre de la vue, ce qui permet d'observer la scène environnante dans son intégralité.

11.7 Conclusion

Au vu des résultats de cette étude, il apparaît qu'une planification de tâches simples est envisageable. La compréhension du modèle n'est pas affectée par cette planification, et les utilisateurs ne ressentent pas la limitation des libertés comme une contrainte forte. De plus, les sujets ayant participé à cette étude ont généralement été plus à l'aise avec la technique planifiée, car celle-ci les aidait à s'orienter.

Dans beaucoup de cas, les techniques d'interaction qui utilisent un contrôle direct sont plus puissantes et plus précises que les techniques planifiées. Mais les techniques planifiées peuvent représenter une bonne alternative, comme par exemple pour une orientation approximative comme nous l'avons montré dans ce chapitre.

Il serait intéressant d'évaluer l'influence des performances de la machine pour une tâche similaire à celle présentée ici. Nous pensons que si le nombre d'images par seconde de l'application est faible, cela aura une influence sur les utilisateurs, car la latence induite par le manque de puissance risque de créer des mouvements inattendus.

CHAPITRE 12

Sélection d'un point 3D pour de la navigation à base de touches

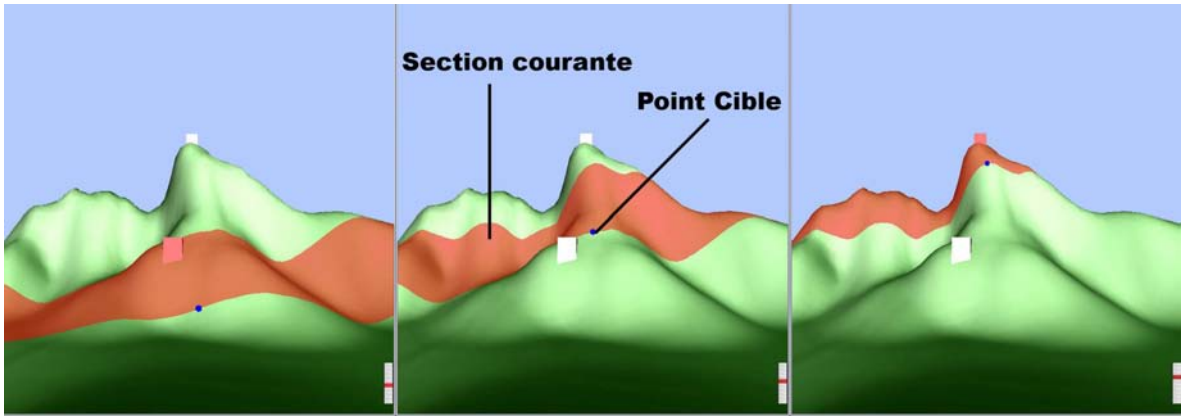


FIGURE 12.1 – Sélection d'un point 3D à l'aide du Z-Goto : l'utilisateur définit la *profondeur* du point, avant de sélectionner un point situé à cette profondeur.

La sélection d'un point est une tâche importante de l'interaction 3D, comme nous l'avons vu dans le chapitre 10. Elle permet entre autres à l'utilisateur de définir le point final d'une trajectoire afin de se déplacer vers ce lieu lors de l'utilisation de techniques de navigation basées sur la technique de "*point of interest*", ou "go to", décrite par Mackinlay *et al.* (1990). Alors que cette technique est directe sur un terminal mobile équipé d'un écran tactile, l'utilisation du "go to" sur un terminal mobile équipé de touches n'est pas triviale. En effet, la sélection d'un point à l'écran à l'aide d'un tel périphérique est difficile, car il est alors nécessaire de déplacer un curseur à l'écran à partir d'appuis successifs des touches de direction.

Malheureusement, le signal binaire fourni par les appareils mobiles implique un nombre de frappe de touches important avant d'atteindre un point précis de l'écran. Or, il se peut que seule une partie restreinte de l'écran soit réellement intéressante (ou atteignable). Par exemple, une grande partie de l'écran peut correspondre à des zones vides (typiquement, le ciel), ou à des objets situés au premier plan. Sur la figure 12.2, la partie gauche de l'écran ne possède qu'un objet de premier plan, qui ne représente qu'une seule cible intéressante pour le déplacement. Le quart haut-droit représente le ciel, qui ne peut être atteint. En fin de

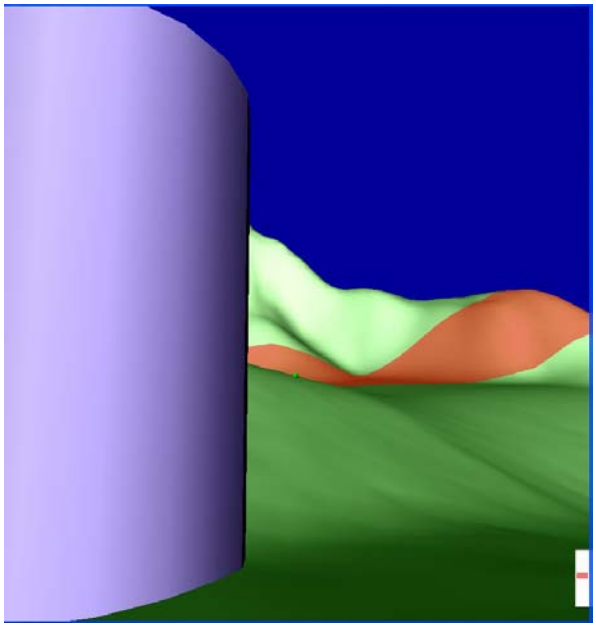


FIGURE 12.2 – Exemple de scène 3D où le nombre de cibles potentielles est réduit par rapport à la taille de l'image.

compte, seul un quart de l'écran est réellement intéressant sur le plan de la navigation. Par conséquent, les actions possibles de l'utilisateur doivent être réduites aux cibles potentielles afin de minimiser le nombre d'actions nécessaires.

De plus, une des caractéristiques des terminaux mobiles est leur petite taille : la résolution et la taille des écrans rend parfois difficile la perception des profondeurs, ce qui réduit l'immersion de l'utilisateur. Par conséquent, la carte cognitive est plus longue et plus difficile à construire, ce qui affecte les performances lors des tâches de navigation.

Partant de ces constats, nous avons développé une nouvelle technique de navigation adaptée aux contraintes des terminaux mobiles. Cette technique, appelée *Z-Goto*(Hachet *et al.*, 2006; Decle *et al.*, 2006), est basée sur la technique du "go to". Cependant, la sélection du point d'intérêt ne se fait pas à l'aide d'un curseur déplacé sur le plan 2D de l'écran, mais se fait dans la *profondeur* de la scène. En effet, avec le "go to" classique, le déplacement du curseur sur le plan xy (plan de l'écran) n'aide pas l'utilisateur à comprendre l'environnement 3D qui l'entoure. Pis, les utilisateurs doivent passer en permanence du repère 2D de l'écran au repère 3D de la scène, ce qui peut créer une charge cognitive supplémentaire.

12.1 Description générale

La technique du Z-Goto que nous avons développée a été étudiée pour pallier aux problèmes des terminaux mobiles évoqués plus haut. L'idée principale consiste à sélectionner le point d'arrivée du déplacement directement dans le plan xz de l'environnement 3D, c'est à dire en prenant en compte l'information de profondeur de la vue. Ceci réduit intrinsèquement le nombre de frappes de touches nécessaires étant donné que seules les

zones pouvant être atteintes sont concernées. De plus, ceci peut améliorer l'immersion et les processus cognitif de l'utilisateur, car la technique travaille directement dans l'espace 3D.

Le Z-Goto opère de la façon suivante. A chaque instant, l'axe z, correspondant à l'axe des profondeurs de la vue courante, est divisé en un certain nombre de "sections" (ce nombre étant paramétrable). La section courante est mise en valeur par une bande colorée, visible sur la figure 12.1. Par conséquent, avec seulement un petit nombre d'actions, un utilisateur peut se déplacer en profondeur, de la plus proche à la plus éloignée. Ce degré de liberté correspond à la distance à laquelle l'utilisateur souhaite aller. Déplacer la section courante le long de l'axe z offre une bonne perception des profondeurs à l'utilisateur, ce qui l'aide dans la construction de sa carte cognitive. De plus, le déplacement des bandes colorées favorise la perception des contours et des reliefs qui peuvent être difficilement perçus sur de petits écrans. Par exemple, dans le cas de terrains 3D, le relief du sol peut être mieux "senti".

Une fois la profondeur du déplacement sélectionnée, un point-cible est affiché (le cercle au centre de la figure 12.1). Sa taille dépend de sa distance par rapport à la caméra, afin d'offrir à l'utilisateur un élément supplémentaire pour la perception de la profondeur. Ce point-cible peut être déplacé latéralement (voir section 12.3.1 pour les détails sur le contrôle du Z-Goto).

Enfin, la validation du point-cible commence le déplacement de la caméra dans la scène. La caméra est translatée jusqu'au point d'arrivée. L'orientation de la caméra peut suivre deux scenarii : effectuer une rotation afin de se placer face au point-cible, ou rester sur un plan horizontal. Lors de navigation dans de grandes scènes, nous avons noté qu'il était préférable de garder la caméra sur un plan horizontal.

Nous avons choisi d'adapter la précision du déplacement en fonction de la distance. Pour les profondeurs les plus proches, la taille des sections et le pas de mouvement en x sont petits. Ceci permet des déplacements fins et précis. Pour de plus grandes distances, la taille des sections et le pas de mouvement latéral sont grands. Ceci permet de se déplacer rapidement. Par conséquent, les utilisateurs peuvent à tout moment atteindre des points éloignés rapidement, pour ensuite être plus précis. Ceci est une des forces du Z-Goto.

12.2 Limitations et extensions

La taille et le nombre de sections est un point important. Actuellement, nous avons fixé ces paramètres en fonction des paramètres *zNear* et *zFar* de la vue d'OpenGL, en utilisant une fonction non linéaire, afin de partitionner l'espace en sections non uniformes (grandes sections pour grandes distances). Par conséquent, le partitionnement de l'espace dépend de l'application. Lors de nos expérimentations, ces paramètres étaient tout à fait satisfaisants. Cependant, afin d'améliorer le découpage de la scène, des algorithmes plus complexes pourraient être développés. Par exemple, un regroupement des zones possédant une profondeur proche dans le *tampon de profondeur* est une voie à envisager.

Dans notre implémentation, la profondeur est divisée seulement selon l'axe z. Ce découpage ne dépend pas de ce qui est réellement visible. Ainsi, il arrive que des sections entières ne soient pas visibles, si aucun pixel visible n'appartient à cette plage de profondeur. (par exemple dans des environnements vallonnés). Afin de pallier à ce problème, nous avons ajouté un *indicateur de section*, qui donne à l'utilisateur la position de la section courante par rapport à l'ensemble des sections de la scène (voir figure 12.2, en bas à droite).

Un autre problème est que, dans certains cas, le point-cible peut ne pas être visible, lorsqu'aucun pixel de la section courante ne possède la coordonnée x désirée. Par exemple, sur la figure 12.3, un objet au premier plan masque la vue. Ceci obligeait l'utilisateur à presser les touches *gauche* et *droite* plusieurs fois jusqu'à faire apparaître le point-cible à nouveau. Afin de corriger ce problème, nous avons décidé de placer le point-cible au plus proche candidat visible : lorsque la touche *droite* ou *gauche* est pressée, le point-cible est directement déplacé vers le premier point candidat, dans la direction souhaitée. Ceci est effectué en parcourant le *tampon de profondeur* ou le *tampon RGBA* (voir section 12.3) de la coordonnée x courante jusqu'aux bords de l'écran ou jusqu'à ce que le nouveau point-cible soit trouvé. Cette optimisation permet de sauter les objets (ou parties d'objets) qui occultent la vue à une profondeur donnée, comme illustré sur la figure 12.3.

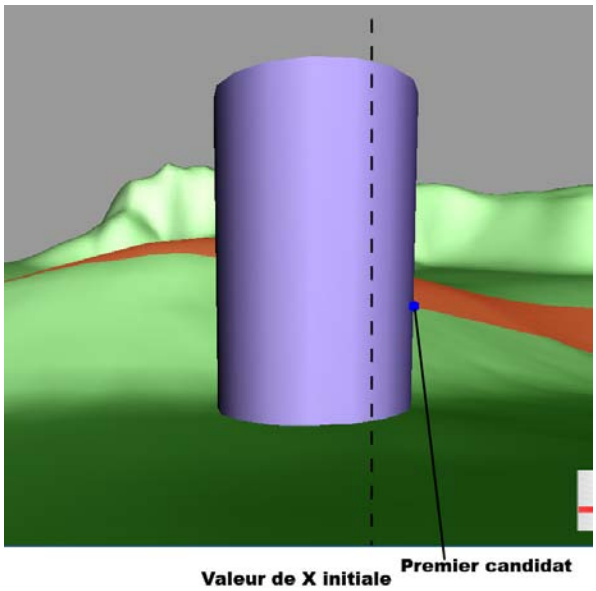


FIGURE 12.3 – Occlusion de la vue par un objet au premier plan ; une cible alternative est automatiquement proposée à l'utilisateur.

Enfin, un problème commun à toutes les approches basées sur le "go to" concerne le "demi-tour" ; lorsque l'utilisateur souhaite revenir en arrière, il doit tourner la vue. Dans notre implémentation, si le point-cible se rapproche du bord de l'écran, la caméra tourne dans cette direction. Cependant, effectuer un demi-tour complet nécessite beaucoup de temps, ce qui peut pénaliser les performances. Une façon de contourner le problème est d'introduire un raccourci permettant d'effectuer cette action.

12.3 Implémentation

Nous avons implémenté le Z-Goto en deux versions : une première pour PC (utilisant OpenGL) et une pour PocketPC (utilisant l'implémentation Vincent d'OpenGL|ES (Will.)).

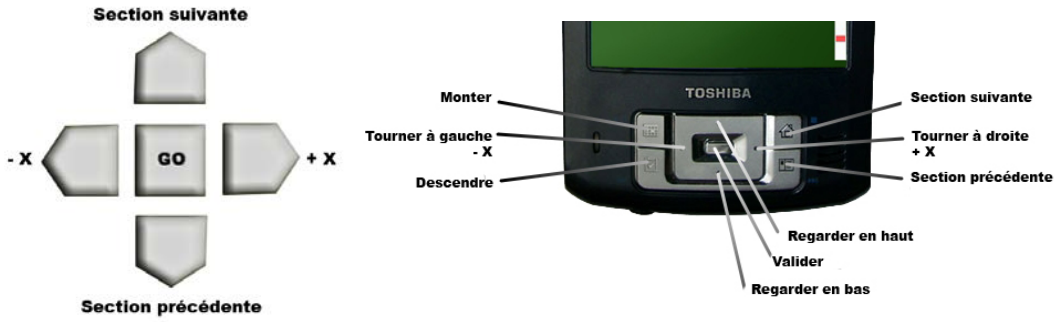


FIGURE 12.4 – Deux configurations possibles : un contrôle avec 4 touches de direction ou un contrôle plus avancé.

12.3.1 Contrôle

Plusieurs configurations de touches sont possibles pour le contrôle du Z-Goto. Avec seulement quatre touches directionnelles et une touche de validation, il est possible de se déplacer n'importe où dans l'environnement : les touches *haut* et *bas* servent à choisir la section courante, et les touches *gauche* et *droite* déplacent le point-cible latéralement. Enfin, la touche de validation est utilisée pour commencer le déplacement vers le point-cible. La figure 12.4 (gauche) illustre une telle configuration.

Si plus de touches sont disponibles (la plupart des téléphones portables ou des PDA possèdent des "touches auxiliaires" situés autour des touches de direction), une autre configuration est possible : les touches de direction peuvent être assignées à la gestion de l'orientation de la caméra, et les touches de fonctions utilisées pour sélectionner la section courante et, par exemple, modifier la hauteur de la caméra (voir figure 12.4, droite). Avec une telle configuration, les utilisateurs possèdent des DDLs supplémentaire. En contrepartie, le temps d'apprentissage est plus long.

12.3.2 Mise en valeur de la section courante

Afin de donner à l'utilisateur un retour visuel durant la sélection de la profondeur, la section courante doit être mise en valeur. Cette opération est faite en colorant les pixels correspondant. Une première approche pour effectuer cette coloration est de parcourir le tampon de profondeur (en utilisant *glReadPixels*), et de changer la couleur des pixels concernés dans le tampon de couleur (en utilisant *glDrawPixels*).

Malheureusement, cette technique est extrêmement lente, glDrawPixels étant certainement une des fonctions les plus lentes et coûteuses d'OpenGL. Nous avons donc développé une approche alternative, basée sur l'utilisation du *tampon de masque* (*stencil buffer* en anglais). Ce tampon permet de donner une valeur à chaque pixel. Dans notre cas, les pixels appartenant à une section donnée sont les projections des points 3D situés à une distance de la caméra comprise entre les paramètres *zMin* et *zMax* de la section. Nous allons assigner une valeur de masque particulière à ces pixels.

– Tout d'abord, le tampon de masque est créé et le "test de masque" est activé en mode *GL_INCR* pour le test de *zPass*, ce qui signifie que la valeur de masque est incrémentée si le pixel est affiché.

- Puis, un plan orthogonal à la direction de vue est dessiné à une distance de *zMax*. Chaque pixel devant ce plan a donc sa valeur de masque incrémentée (donc à "1"). Les valeurs de masque des points derrière le plan ne sont pas modifiées (donc "0").
- Un second plan orthogonal à la direction de vue est dessiné à une distance de *zMin*. Par conséquent, les points situés à une distance inférieurs à *zMin* ont une valeur de masque de "2". Les valeurs de masque des points situés à une distance entre *zMin* et *zMax* sont donc "1" (voir figure 12.5).
- Enfin, *glStencilFunc (GL_EQUAL, 1, 1)* demande à OpenGL de ne dessiner que sur les pixels possédant 1 comme valeur de masque. Un plan semi-transparent dessiné devant la caméra colore donc uniquement les pixels concernés.

Cette méthode est couramment utilisée dans des applications 3D temps-réel car elle ne dépend pas de la complexité de la scène.

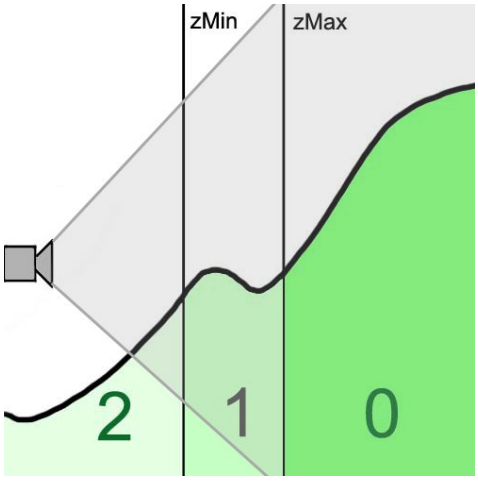


FIGURE 12.5 – Le tampon de masque, utilisé pour colorer les pixels correspondant à la section courante.

12.3.3 Obtenir les coordonnées de la cible

Comme pour la technique du "go to", les trajectoires dans le Z-Goto sont toujours effectuées de la caméra vers des points visibles. Dans notre cas, il est aisé d'obtenir les coordonnées écran du point cible. Puis, en utilisant *gluUnProject*, le lien entre les coordonnées écran et les coordonnées de la scène 3D réalisé.

Sur la version PC, étant donné que tous les pixels de la section courante ont une valeur de masque égale à 1, obtenir les coordonnées des pixels d'une section donnée est effectué directement en lisant le tampon de masque. Malheureusement, OpenGL|ES ne permet pas la lecture de ce tampon. La méthode alors utilisée, pour la version mobile, est de lire le tampon de couleur RGBA. Ainsi, plutôt que de considérer les pixels possédant une valeur particulière de masque, nous avons considéré les pixels possédant une couleur spécifique (par exemple, du rouge pur) qui est assignée durant la phase de coloration de la section courante. Par conséquent, cette couleur doit être bannie des couleurs de la scène.

Il est bien entendu possible d'effectuer le calcul en deux passes : une première passe, effectuée dans une résolution plus faible, ne servira qu'au calcul des coordonnées de la cible,

où la scène est dessinée sans ses couleurs. La couleur de la section sera alors la seule couleur de l'image. Puis, une fois les coordonnées de la cible calculées, le rendu normal de la scène est effectué. Cette technique possède l'avantage d'éliminer les possibles interférences entre les couleurs de la scène et la couleur de la section. Cependant, elle nécessite de rendre la scène deux fois, ce qui diminue bien évidemment les performances du programme.

12.4 Évaluation

Nous avons cherché à évaluer l'efficacité du Z-Goto pour la navigation dans des environnements 3D. Nous avons mené une expérimentation où les sujets devaient atteindre des cibles en utilisant les deux techniques différentes : le "go to" classique et le Z-Goto. Dans cette section, nous donnons une description technique de notre expérimentation (participants, environnement technique, tâches et procédure). Nous analysons ensuite les résultats et discutons les performances et la satisfaction des utilisateurs.

12.4.1 Participants

10 droitiers, habitués à utiliser un téléphone portable mais novices en interaction 3D, ont participé à l'étude. Le groupe était composé de 7 hommes et 3 femmes. Les âges étaient compris entre 20 et 26 ans (moyenne = 23). Tous les sujets ont utilisé leur main droite pour tenir l'appareil mobile, et leur pouce pour presser les touches. Avant l'expérimentation, les participants ont pu utiliser pendant quelques moments les deux techniques afin de se familiariser avec l'environnement matériel et logiciel. Après l'expérimentation, nous avons demandé aux sujets de remplir un questionnaire. Les résultats seront étudiés par la suite.

12.4.2 Environnement technique

Le terminal mobile utilisé pour l'expérimentation était un Toshiba e800 possédant une résolution d'écran de 320×240. Cet appareil est équipé de touches directionnelles et de touches de fonction. La touche de validation est située au centre des touches de direction. La vitesse de rafraîchissement lors de l'expérimentation était environ de 10 images par seconde.

12.4.3 Tâche

La tâche consiste à atteindre des zones distantes dans un environnement 3D. La scène est un large terrain possédant des hauteurs aléatoires. Le rendu du terrain est partiel, c'est à dire que seuls les objets proches de l'utilisateur sont affichés. Par conséquent, les zones à atteindre ne sont pas toujours directement visibles par les sujets ; plusieurs déplacements sont nécessaires avant de voir les dites zones (colorées en jaune). Lorsque la cible n'est pas visible, une indication donne la direction à suivre (voir figure 12.6). Dans cette expérimentation, nous avons principalement voulu évaluer la partie motrice de la navigation, la tâche de recherche étant fortement réduite. Cette tâche est une *tâche de recherche primaire* comme définie par Bowman dans (Bowman *et al.*, 2001a).

Chaque sujet devait atteindre les cibles le plus rapidement possible avec les deux différentes techniques. En utilisant le "go to" classique, un curseur est déplacé sur l'écran afin d'indiquer le point d'arrivée du déplacement de la caméra. Avec le Z-Goto, la profondeur était contrôlée en sélectionnant la section courante à l'aide des touches haut/bas, les touches

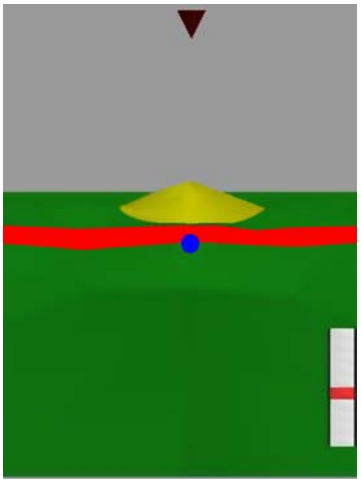


FIGURE 12.6 – Exemple de cible à atteindre.

gauche/droite étant utilisées pour déplacer latéralement le point-cible. Pour les deux techniques, la validation implique le déplacement de la caméra vers le point sélectionné. La hauteur de la caméra suit la hauteur du terrain. L'orientation de la caméra est ramenée vers le plan horizontal.

12.4.4 Procédure

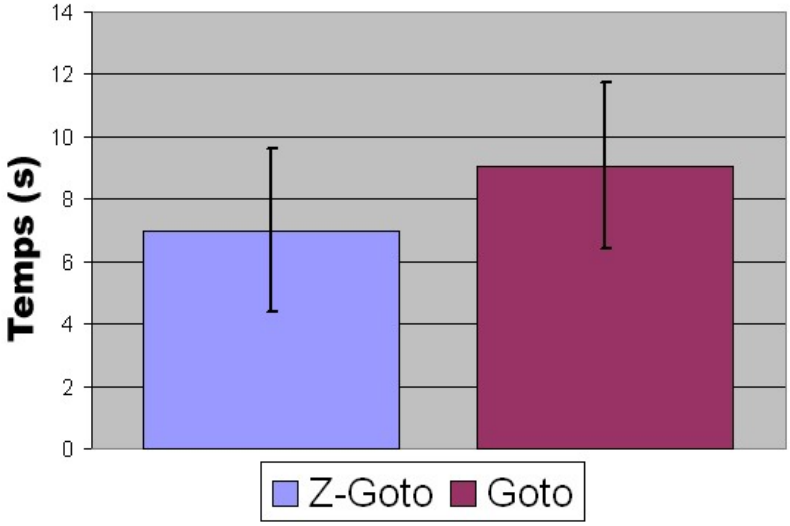
Nous avons demandé à chaque sujet d'utiliser les deux techniques pour atteindre les cibles. L'ordre des techniques à utiliser était choisi aléatoirement. Pour chaque technique, six cibles devaient être atteintes Nous avons demandé aux sujets d'effectuer l'expérimentation deux fois afin d'évaluer l'effet d'apprentissage. La position des cibles implique d'aller vers des points proches puis éloignés, à gauche puis à droite de la vue. Pour chaque essai, nous avons mesuré le temps nécessaire pour atteindre les cibles. Nous avons ainsi obtenu 12 temps par sujet, soit un total de 120 mesures.

12.5 Résultats

12.5.1 Performance

Pour l'analyse statistique des moyennes de temps obtenues, nous avons utilisé le test "t de student". Les résultats ont montré que le Z-Goto a été significativement plus rapide que le "go to" classique. Les résultats, illustrés sur la figure 12.7, prennent en compte tous les temps obtenus (ie. le premier et le second temps des sujets).

La différence entre le premier et le deuxième essai n'a pas montré de différence statistique significative. Par conséquent nous ne pouvons rien conclure au sujet de l'effet d'apprentissage.



	Z-Goto	Goto
Moyenne de temps (s)	6.98	9.07
Écart type	5.23	5.29
Signification	$t(9) = -4.149, p = 0.0001$	

FIGURE 12.7 – Moyennes de temps effectués.

12.5.2 Satisfaction

Afin de compléter notre analyse, nous avons mené une étude qualitative à propos de la satisfaction des utilisateurs. Après l'expérimentation, nous avons demandé à chaque sujet leur avis sur les techniques. Leur opinion était notée de 1 à 4, 1 exprimant une opinion négative, 4 une opinion positive. Les réponses des sujets sont résumées sur la figure 12.8. Tous les sujets ont préféré utiliser le Z-Goto par rapport au "go to" classique.

12.6 Discussion

Les sujets ont été plus rapides avec le Z-Goto qu'avec le "go to" classique pour la tâche que nous avons proposée. Le résultat est cohérent avec les impressions des utilisateurs exprimés dans le questionnaire. La différence peut être liée au nombre de frappes de touches nécessaires pour la sélection du point final du déplacement. En effet, le Z-Goto limite le nombre de touches nécessaires en réduisant le champ d'action à ce qui peut être atteint. Ainsi, une approche basée sur la scène semble plus rapide et plus intuitive qu'une approche basée uniquement sur l'écran.

L'expérimentation proposée consistait en une tâche de recherche primaire (c'est à dire que le sujet était guidé par des indications), la recherche d'itinéraire était ainsi fortement réduite. Par conséquent, nous avons principalement évalué la composante motrice de la navigation. Cependant, nous sommes persuadés que le Z-Goto aide à la construction de la carte cognitive et, par conséquent, améliore les performances de l'utilisateur dans ses tâches de navigation. En particulier, les sujets ont mieux perçu la profondeur avec le Z-Goto (voir figure 12.8). Nous devons étudier plus en détail les relations entre la technique utilisée et l'ef-

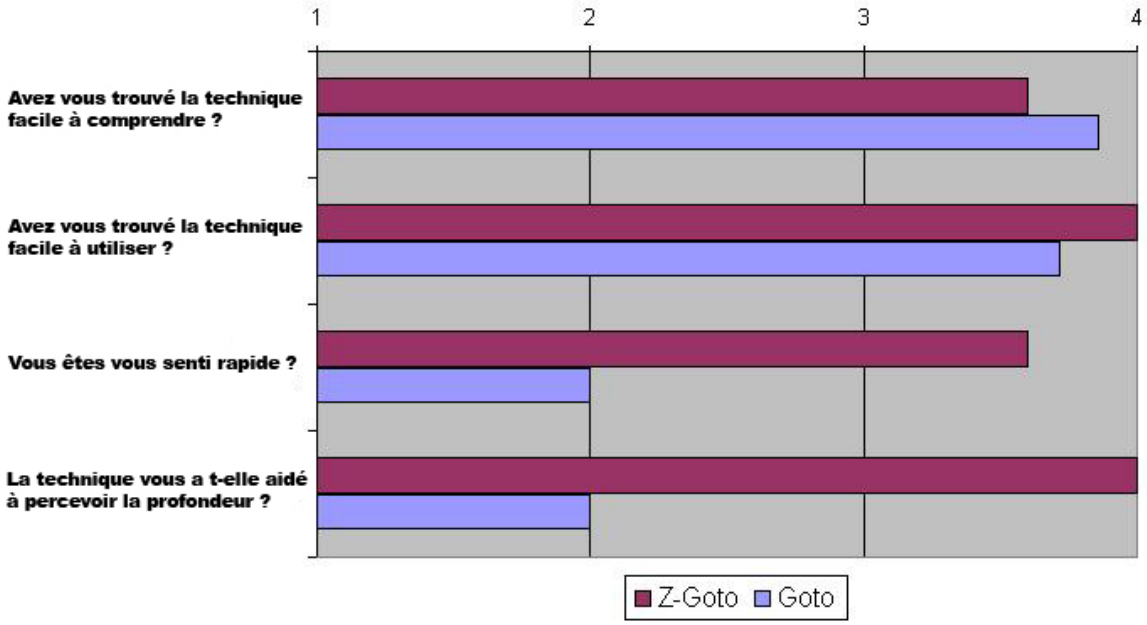


FIGURE 12.8 – Les préférences des utilisateurs.

ficacité de la construction de la carte cognitive de l'utilisateur. Une expérimentation adaptée doit donc être menée. Parallèlement à cela, il serait intéressant de tenter de mesurer le degré d'immersion cognitive de l'utilisateur.

12.7 Conclusion et travaux futurs

Dans ce chapitre, nous nous sommes intéressés à la seule tâche de navigation dans une scène 3D à l'aide des touches de l'appareil. Nous avons proposé une nouvelle technique appelée Z-Goto, permettant une navigation efficace en n'utilisant que les touches directionnelles du clavier de l'appareil. Cette technique est une extension du "go to" classique. Étant donné que le Z-Goto opère dans l'espace 3D, il s'adapte à l'environnement. Ceci permet de réduire le nombre de frappes de touches nécessaires, les déplacements de l'utilisateur étant réduits aux seules parties intéressantes. Ainsi, les déplacements dans la scène 3D peuvent être accélérés. De plus, comparé au "go to" classique, le Z-Goto améliore l'immersion cognitive de l'utilisateur. Tout d'abord, n'opérer que dans un seul référentiel (l'espace 3D) évite à l'utilisateur de passer du référentiel de l'écran à celui de la scène, qui impliquerait une charge cognitive supplémentaire. Enfin, le retour visuel que fournit la mise en valeur de la section courante aide l'utilisateur à percevoir et à appréhender l'environnement. En particulier, la profondeur et les distances peuvent être mieux perçues. Ceci favorise la construction de la carte cognitive et, par conséquent, améliore les performances de l'utilisateur dans ses tâches de navigation.

La technique que nous avons proposée a initialement été étudiée en regard des contraintes imposées par les terminaux mobiles. En particulier, nous voulions que la technique n'utilise que quelques touches binaires. Mais au delà des appareils mobiles, nous

sommes persuadés que le Z-Goto peut être utile dans d'autres contextes. Par exemple, utiliser le Z-Goto avec quelques touches représente une bonne alternative aux techniques usuelles. La meilleure connaissance de l'environnement est également bénéfique aux performances des utilisateurs. Un autre contexte où le Z-Goto peut être particulièrement intéressant est la visualisation 3D collaborative. En effet, améliorer l'interaction 3D collective dans un contexte de grands écrans actuellement un axe de recherche important. Une idée serait ainsi d'équiper les fauteuils d'une salle de visualisation collaborative avec des touches de direction. Les participants pourraient ainsi naviguer au sein de l'environnement 3D projeté en utilisant la technique du Z-Goto.

L'un des défauts majeurs des techniques basées sur le "go to" est qu'elles ne fournissent que peu de contrôles à l'utilisateur. En effet, il n'est pas possible de définir la distance à laquelle s'arrêter du point d'intérêt, ni l'angle avec lequel le visualiser. Le chapitre suivant est consacré à une technique de navigation appelée Navidget que nous avons développée. Cette technique permet de combler les lacunes du "go to" en proposant un moyen simple et intuitif de contrôler la distance et l'angle de la caméra avec le point d'intérêt.

CHAPITRE 13

Contrôle planifié avancé de la caméra

Le problème du contrôle des mouvements de caméra dans les applications 3D a été étudié dès l'apparition des premières applications 3D interactives. Beaucoup d'interfaces ont été proposées afin d'optimiser les performances des utilisateurs. Cependant, contrôler la caméra d'un environnement 3D reste une tâche difficile, et il est toujours nécessaire d'étudier et de développer de nouvelles techniques d'interaction.

Navidget (Hachet *et al.*, 2008, 2009) est une technique de positionnement de caméra dans un environnement 3D que nous avons développé. Cette technique est particulièrement adaptée à l'utilisation sur TM, car elle a été étudiée pour être utilisée à l'aide d'un écran tactile. De plus, elle peut fonctionner même si un rendu temps réel n'est pas assuré, car elle est basée sur la planification des mouvements de caméra, introduite au chapitre 11.

Cette technique dérive du "Point Of Interest" (POI), décrit par Mackinlay *et al.* (1990), dans lequel l'utilisateur choisit à l'écran le point vers lequel déplacer la caméra. L'idée principale de Navidget est de laisser à l'utilisateur le contrôle de l'endroit vers lequel il souhaite aller, ainsi que la façon d'y aller. Il pourra ainsi définir simplement la distance le séparant de son point d'intérêt, ainsi que l'angle sous lequel l'observer, contrairement aux techniques de "go to" traditionnelles, qui essaient de deviner comment l'utilisateur pourrait souhaiter observer son centre d'intérêt. Le contrôle de Navidget se fait à l'aide d'une interface graphique dédiée, aussi appelée un *widget*.

13.1 Contrôles

Navidget possède plusieurs *niveaux d'utilisation*, en fonction du type de cible ou de tâche

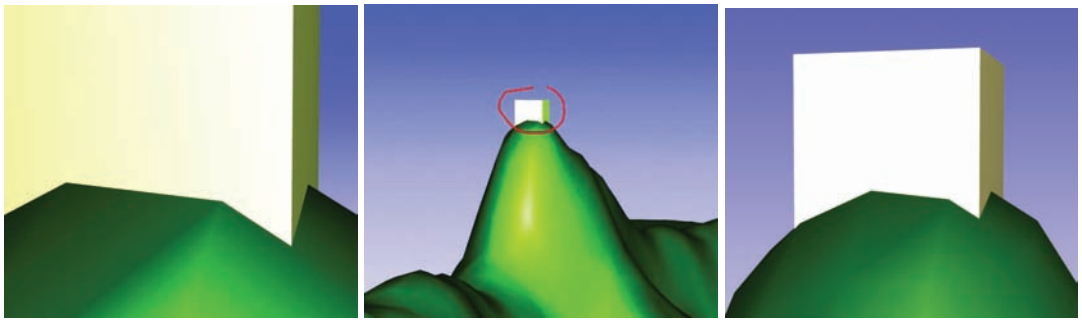


FIGURE 13.1 – Sélectionner un point d'intérêt amène souvent à une vue trop proche (gauche). Encercler la zone d'intérêt (centre) déplace la caméra vers une vue appropriée (droite).

de l'utilisateur. Ces niveaux permettent un contrôle des mouvements de caméra plus ou moins avancé, mais respectent tous la même philosophie de base de Navidget, à savoir sélectionner un point (ou une zone) d'intérêt, et définir la façon dont la caméra devra observer cette cible. Il est alors possible de simplement se rapprocher d'un point ou d'une zone d'intérêt ou bien de définir précisément l'angle avec lequel observer son point d'intérêt. De plus, Navidget permet, à l'aide de gestes simples, d'appliquer une rotation à la caméra.

13.1.1 Se rapprocher d'un point ou d'une zone d'intérêt

Le niveau de contrôle le plus simple de Navidget correspond au "go to" standard. Il est contrôlé simplement en cliquant sur le point d'intérêt. La caméra se déplace alors vers ce point. Afin de ne pas obtenir une caméra qui soit trop proche de la surface du point cible, la caméra s'arrête lorsqu'elle a parcouru 90% de la distance séparant la position initiale de la caméra et le point d'intérêt.

Cette technique offre l'avantage d'être très facile à contrôler. Cependant, le problème principal de cette méthode est que la distance entre la caméra et le point cible est calculée par le système. Ceci conduit à des situations où l'utilisateur se retrouve face à une surface occupant l'intégralité de la vue, avec pour seul recours possible le changement d'outil. L'image de gauche de la Figure 13.1 montre ce phénomène.

Un autre point négatif précédemment cité est l'impossibilité de sélectionner une *zone d'intérêt*, plutôt qu'un simple point. Il est alors impossible de se rapprocher d'un groupe d'objets. Afin de résoudre ces problèmes, nous proposons une première extension du "go to" consistant à *encercler la zone d'intérêt*.

Encercler est un geste très intuitif, étant donné que l'utilisateur dessine directement ce qu'il souhaite observer, comme illustré sur la Figure 13.1. Encercler est particulièrement adapté aux systèmes tactiles, qu'ils soient basés sur l'utilisation d'un stylet ou du doigt. De plus, ce geste est réalisé dans la vie de tous les jours, par exemple pour mettre en valeur un détail dans un texte ou dans une image. La métaphore de l'encerclement a précédemment été utilisée par Schmalstieg *et al.* (1999), mais pour des tâches de sélection plutôt que pour de la navigation.

L'avantage de cette métaphore est qu'elle ne nécessite pas de périphérique particulier,

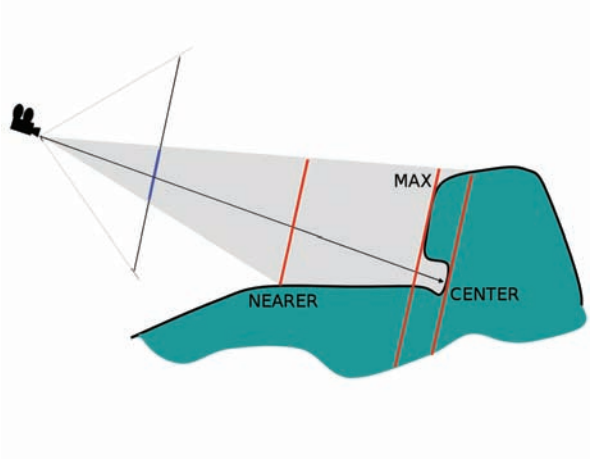


FIGURE 13.2 – Les trois distances possibles entre la caméra et la cible : CENTER, qui utilise le centre du cercle comme point cible, AX, qui calcule la profondeur la plus représentée, et NEARER, qui sélectionne le point le plus proche dans la sélection.

un simple dispositif de pointage suffit. Cependant, cette tâche peut être difficile à réaliser à l'aide d'une souris. Il sera alors préférable d'utiliser une sélection "rectangulaire", qui est couramment utilisée dans de nombreuses applications 2D ou dans quelques applications 3D orthographiques.

Une fois le geste réalisé et le stylet relâché, la caméra commence son déplacement vers le centre du cercle. Un problème apparaît alors : à quelle distance faut-il que la caméra s'arrête de la cible ?

Il y a plusieurs réponses possibles à cette question, schématisés sur la Figure 13.2. La première, et la plus simple, consiste à projeter le centre du cercle dans la vue courante (CENTER dans la figure), et de récupérer les coordonnées 3D de ce point. Le problème de cette approche est que ce point n'existe pas forcément. Il peut par exemple correspondre à un point dans le ciel, dont la profondeur est infinie. De plus, cela ne permet pas de sélectionner plusieurs objets comme zone d'intérêt. Afin de résoudre ce problème, nous avons développé une approche alternative (MAX), basée sur la valeur de profondeur la plus représentée dans la zone d'intérêt sélectionné. Malheureusement, cette deuxième approche risque d"'oublier" certains détails, et considérer le fond de la sélection comme étant la profondeur la plus représentée. L'approche la plus convaincante a été de choisir la profondeur la plus proche de la zone sélectionnée dans la vue courante (NEARER). Ceci assure à l'utilisateur que toute sa sélection sera visible.

A cause de la perspective, la vue résultante ne correspond pas exactement à ce qui a été encerclé. Il est à noter que Zeleznik et Forsberg (1999) ont proposé un concept de *Region zooming*, qui peut être relié à notre technique. Cependant, leur méthode nécessitent de l'utilisateur de sélectionner un point avant de modifier la taille de la fenêtre de zoom. Par conséquent, la zone cible est toujours centrée sur un point particulier de la scène 3D. Nous pensons qu'encercler est une métaphore très directe, étant donné que les utilisateurs désignent leur zone d'intérêt à l'aide d'un geste simple et familier.

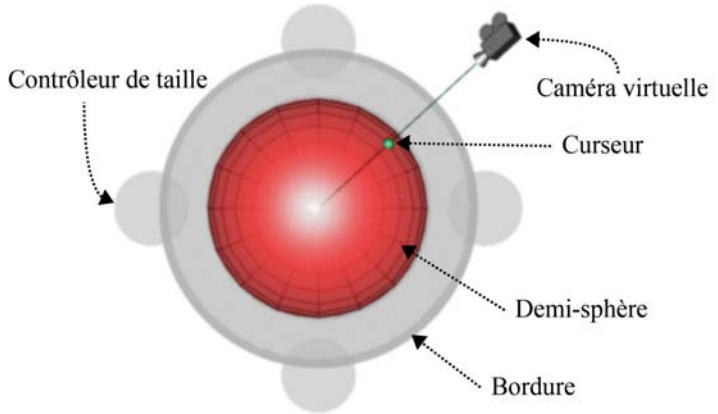


FIGURE 13.3 – Le *widget* de Navidget.

13.1.2 Définir la direction de vue

Dans les techniques de navigation existantes basées sur le POI, la direction de la vue cible est calculée automatiquement. La caméra est généralement dirigée vers le point cible avec une direction de vue alignée avec la normale de la face correspondante, ou bien avec une vue oblique pré définie (Zeleznik et Forsberg, 1999). Par conséquent, l'utilisateur ne peut définir comment il souhaite voir son point d'intérêt. Pire encore, un calcul automatique peut conduire à une vue inadaptée, car trop proche ou trop éloignée de la cible.

Afin de permettre à l'utilisateur de contrôler la direction de la vue cible, nous utilisons un *widget 3D*. Ce *widget*, illustré sur la Figure 13.3, est composé d'une *demi-sphère* dessinée face à la caméra, d'une bordure (*border ring*), d'un ensemble de *contrôleurs de taille* (*size actuators*) et d'une *caméra virtuelle*.

13.1.2.1 Description des éléments du *widget*

Demi-sphère La *demi-sphère* met en valeur la partie de la scène qui sera visible à la fin du mouvement de la caméra.

Curseur et Caméra 3D En déplaçant le *curseur 3D* le long de la surface de la demi-sphère, l'utilisateur contrôle l'angle avec lequel il souhaite observer son point d'intérêt. La représentation d'une *caméra 3D* montre la position et l'angle de la caméra de destination. Ceci fournit intrinsèquement un retour visuel à l'utilisateur. De plus, les occlusions possibles peuvent être vues, et sont donc naturellement évitées (voir Figure 13.4). La section 13.2.2 présente une extension, appelée *caméra intelligente*, qui améliore encore le retour visuel que donne Navidget.

Border Ring L'étude utilisateur préliminaire que nous avons menée a révélé le besoin de sélectionner facilement une vue qui soit orthogonale à la direction de vue courante. Nous avons donc rajouté le *border ring*, un anneau situé autour de la sphère. Il est ainsi plus facile de se positionner sur cet anneau que de tenter de placer précisément le curseur au bord de

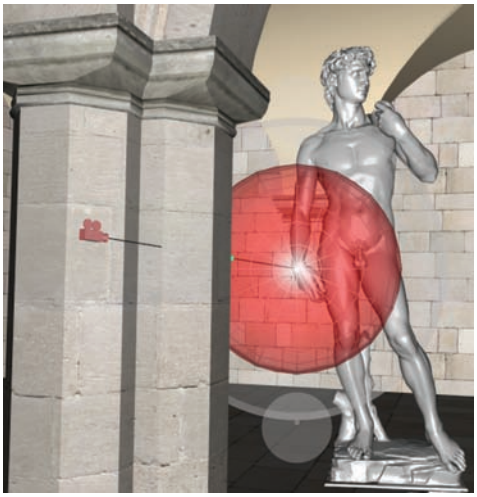


FIGURE 13.4 – Le retour visuel que fournit Navidget permet de détecter les occlusions.

la demi-sphère. Ainsi, déplacer le curseur le long du *border ring* revient à le déplacer le long des limites visibles de la demi-sphère.

Retournement de la demi-sphère L'étude préalable a également révélé que les utilisateurs souhaitaient pouvoir déplacer la caméra derrière la demi-sphère. Par conséquent, nous avons introduit un mécanisme permettant d'inverser la direction de la demi-sphère, devant ou derrière la cible. Lorsque le curseur sort puis ré-entre dans le *widget*, celui-ci réalise une rotation de 180°. Cette rotation est animée afin de suggérer ce retournement (voir Figure 13.5). Ceci permet de déplacer la caméra virtuelle tout autour du point cible.

FIGURE 13.5 – Un geste "Sortir Entrer" (tracé bleu) permet d'inverser l'orientation de la demi-sphère, afin de placer la caméra virtuelle derrière le point cible.

Contrôleurs de taille Nous avons ajouté des contrôleurs de taille qui permettent de redimensionner le *widget*, sans avoir à relâcher le bouton de la souris ou le stylet. Ils sont situés dans les directions cardinales de la demi-sphère. Pour redimensionner le *widget*, l'utilisateur peut alors "attraper" un des contrôleurs, c'est à dire déplacer son curseur vers ce dernier. Le système passe alors dans un état **REDIMENSIONNEMENT**. L'utilisateur peut alors modifier la taille du *widget*, et donc la distance entre la caméra virtuelle et le point cible, en réalisant des mouvements verticaux (respectivement horizontaux). Le mode REDIMENSIONNEMENT est quitté si un mouvement horizontal (respectivement horizontal) est

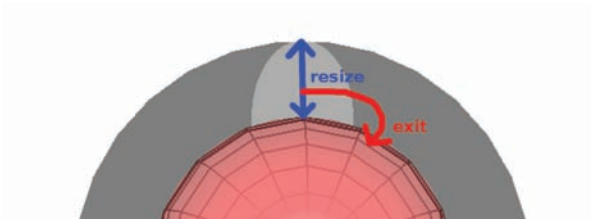


FIGURE 13.6 – les contrôleurs de taille, situés aux points cardinaux de la demi-sphère, permettent de redimensionner le *widget*.

détecté (voir Figure 13.6). Ceci signifie que l'utilisateur souhaite revenir dans la demi-sphère ou sur le *border ring*. Par conséquent, en contrôlant une direction et une distance, l'utilisateur peut positionner la caméra dans l'espace 3D.

13.1.2.2 Contrôle du *widget*

Le *widget* est centré autour du point cible. Il apparaît si aucun évènement "relâchement" n'est reçu après le pointage. En d'autres termes, si l'utilisateur clique et attend un court instant, le *widget* apparaît. Sa taille par défaut est calculée de façon à occuper une certaine proportion de la fenêtre de l'application.

Il est également possible de définir la taille du *widget* en esquissant celui-ci. Ainsi, le *widget* apparaît si l'utilisateur ne relâche pas le bouton de la souris après avoir dessiné un cercle. Il est alors affiché avec une taille donnée par le rayon du cercle dessiné par l'utilisateur. Il est donc possible de contrôler à la fois la distance avec la cible et la direction de vue. L'approche basée sur l'esquisse augmente l'aspect intuitif de l'interaction.

Il serait bien entendu possible d'utiliser un bouton supplémentaire, la main non dominante ou un périphérique sensible à la pression afin de faire apparaître le *widget*. Cependant, notre technique étant basée sur l'utilisation de gestes simples, en une unique séquence, nous voulons éviter l'utilisation de menus ou de manipulation à effectuer par la main non dominante. De plus, l'utilisation d'un bouton supplémentaire n'est pas possible lorsque l'on interagit à l'aide d'un doigt ou d'un stylet passif.

Lorsque l'utilisateur relâche la souris ou le stylet et que le curseur est sur la demi-sphère ou sur le *border ring*, la vue est déplacée vers la caméra virtuelle. Si le curseur est en dehors du *widget*, l'action est annulée et le *widget* disparaît.

Afin d'obtenir des mouvements de caméra lisses et continus, nous réalisons une animation en interpolant la position et l'orientation de la caméra initiale vers la caméra cible spécifiée. L'animation est réalisée en temps constant, indépendamment de la distance à parcourir. La formule de base est : $P(t) = P_0 + f(t) \times (P_1 - P_0)$, avec $t \in [0; 1]$.

Plusieurs choix sont possibles concernant le choix de la fonction $f(t)$. Cette fonction doit être continue C^1, afin que les mouvements de la caméra soient fluides et sans à-coups. De plus faut que la fonction vérifie $f(0) = 0$ et $f(1) = 1$.

– **Interpolation linéaire**. Dans ce cas, la vitesse de déplacement est constante tout

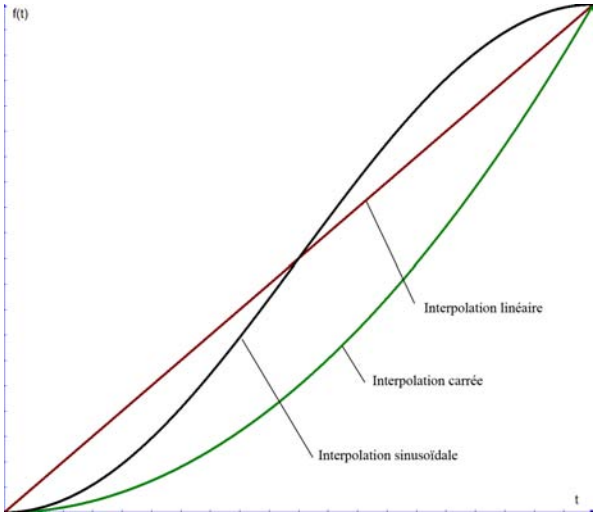


FIGURE 13.7 – Trois fonctions d'interpolation possibles : linéaire, carrée ou sinusoïdale.

au long de l'animation. Cette méthode est la plus simple, mais possède l'inconvénient de produire des mouvements de caméra brusques. La fonction associée est : $f_{lineaire}(t) = t$.

– **Interpolation carrée**. Cette interpolation est basée sur une fonction polynomiale de degré 2. Ici, la vitesse est faible au début de l'animation, et accélère de manière constante. Ici, le mouvement commence doucement mais finit de manière brutale. Dans ce cas, la fonction est : $f_{carre}(t) = t^2$.

– **Interpolation sinusoïdale**. Cette interpolation permet d'obtenir un mouvement doux : le mouvement démarre de façon douce, puis accélère en milieu de parcours, pour finalement ralentir en approchant le point d'arrivée. Une fonction possible est : $f_{sinusoidale}(t) = \frac{1}{2}(1 + \sin(\pi * t - \frac{1}{2}))$.

Les courbes correspondant aux différentes interpolations sont visibles sur la Figure 13.7. Dans notre implémentation, nous avons utilisé l'interpolation sinusoïdale, qui permet d'obtenir les mouvements de caméra les plus fluides.

13.1.3 Autres mouvements de caméra

En plus des contrôles précédemment cités, nous avons permis le déclenchement d'autres mouvements de caméra à l'aide de gestes simples. Ces gestes permettent d'appliquer une rotation à la caméra, ou de retourner à une précédente position.

13.1.3.1 Gestes de rotations

Il est souvent utile de tourner la caméra autour d'un de ses axes. Ceci permet d'observer la scène 3D située autour de soi. Ce mouvement de caméra est généralement réalisé de manière continu, les mouvements de la souris contrôlant les rotations de la caméra autour de ses axes : par exemple, un mouvement horizontal produit une rotation de la caméra autour

de son axe vertical, la faisant "regarder vers la gauche ou vers la droite". Un mouvement vertical permet lui de regarder vers le haut ou vers le bas. Cependant, comme nous l'avons montré dans le chapitre 11, un contrôle direct de la caméra peut ne pas être optimal dans certains cas, surtout lorsque les performances sont réduites. Nous avons donc ajouté à Navidget un *contrôle planifié* des rotations de la caméra. Ainsi, il est possible de tourner la caméra de vers la gauche ou la droite en dessinant un geste horizontal. De même, un geste vertical permet d'appliquer une rotation verticale, comme le décrit la section 11.6.

13.1.3.2 Revenir à une position précédente

Un geste, visible sur la Table 13.1, a été défini afin de permettre à l'utilisateur de revenir à la position et à l'orientation précédente de la caméra. Ceci peut être considéré comme une commande "Annuler", présente dans la grande majorité des logiciels actuels. Afin de permettre de revenir à une caméra antérieure, chaque caméra (c'est à dire une position et une orientation) est empilée dans une pile. Lorsque l'on souhaite revenir à la caméra précédente, il suffit alors de dépiler le dernier élément de la pile, et de calculer une trajectoire allant de la caméra courante à la caméra récupérée de la pile.

En rajoutant cette fonctionnalité à Navidget, ceci permet à l'utilisateur d'explorer l'environnement 3D en réalisant des allers-retours. Ainsi, lorsque plusieurs objets doivent être observés successivement, les techniques standard nécessitent de se déplacer d'une cible à une autre, et de devoir trouver son chemin entre chaque. Avec Navidget, il est facile d'aller observer un objet puis de revenir à une vue plus globale afin de se déplacer vers une nouvelle cible. Ce mode de navigation se rapproche de la technique du *ZoomBack*(Zeleznik *et al.*, 2002). Cependant, cette technique nécessitait l'utilisation d'un bouton à deux niveaux de pression, et n'est donc pas adaptée à une utilisation sur un terminal mobile.

13.1.4 Résumé des gestes de Navidget

Les gestes reconnus par Navidget sont résumés sur la Table 13.1. Le nombre de gestes que l'utilisateur doit apprendre est faible, et ils sont basés sur des gestes couramment utilisés. Par conséquent, le temps d'apprentissage de Navidget est court.

13.2 Améliorations et contrôle avancé de Navidget

Les fonctionnalités de Navidget que nous avons décrites permettent de placer une caméra virtuelle de façon précise. Cependant, les tests préliminaires que nous avons réalisés nous ont permis d'améliorer encore cette technique, en proposant des fonctionnalités supplémentaires : la *prévisualisation* de la vue ainsi que la *caméra intelligente*.

13.2.1 Fenêtre de prévisualisation

Navidget a initialement été conçu pour le positionnement rapide de caméras dans un environnement 3D. Nous avons cependant remarqué que l'ajout d'une fenêtre de prévisualisation affichant la vue depuis la caméra virtuelle, visible sur la Figure 13.8, rendait cette technique utile pour des tâches d'inspection distante. Les utilisateurs peuvent ainsi observer et inspecter des lieux distants, et décider de se déplacer ou non vers ces cibles. Concrè-

Geste	Action résultante
 Pointer et encercler	Se rapprocher d'une zone
 Geste "Undo"	Revenir à la position précédente
 Gestes horizontaux ou verticaux	Tourner la vue

TABLE 13.1 – Résumé des gestes de Navidget.

tement, si un utilisateur désire seulement inspecter les objets d'une scène sans avoir à se déplacer, il lui suffit de relâcher en dehors du *widget* afin d'annuler ce dernier. Si la vue cible lui convient, il peut bien entendu se déplacer vers celle-ci.

Il est à noter que cette technique se rapproche de travaux existants, comme par exemple le *Magic Mirror* (Grosjean et Coquillart, 1999), où l'inspection était réalisée à l'aide d'un miroir contrôlé par un périphérique à 6 DDL. Grâce à Navidget associé à une fenêtre de pré-visualisation, il est possible d'observer une cible de toutes parts simplement à l'aide d'un périphérique à 2 DDL.

Une telle fenêtre n'est cependant pas adaptée lorsque l'écran utilisé est de petite taille. En

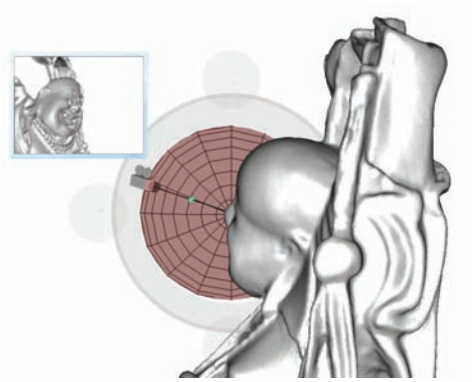


FIGURE 13.8 – La fenêtre de pré-visualisation. Dans cet exemple, la demi-sphère a été retournée.

effet, elle occuperait une part trop importante de l'image. De plus, elle nécessite de rendre la scène deux fois, ce qui réduit les performances. Cependant, lorsqu'il est possible d'utiliser une telle fonctionnalité, l'utilisation de la fenêtre de pré-visualisation avec Navidget peut être très utile. Cela permet d'explorer une scène 3D sans avoir à changer de position, et permet ainsi de se concentrer sur des parties spécifiques de la scène sans changer de contexte global. Cette approche peut être importante pour le processus cognitif qui peut être mis en oeuvre lors de tâches de navigation.

13.2.2 Caméra "intelligente"

Comme nous l'avons décrit dans la section 13.1.2, le retour visuel de Navidget aide les utilisateurs à détecter les possibles intersections et occlusions entre la caméra virtuelle et la scène 3D. Nous proposons cependant une extension qui aide les utilisateurs en fournissant des informations supplémentaires lors du positionnement de la caméra. Cette extension est utile lorsqu'un objet est situé entre le point d'intérêt et la caméra virtuelle, ou lorsque la caméra est cachée par un occultant. Ces deux cas sont représentés sur les Figures 13.9 (a) et 13.9 (c).

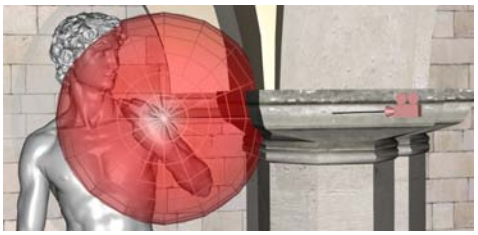
Étant donné que nous connaissons la position du point d'intérêt ainsi que la position de la caméra virtuelle, il nous est possible de détecter la présence de géométrie entre ces deux points qui pourrait gêner l'utilisateur. De plus, nous pouvons mettre en valeur la partie de l'objet qui occulte la vue (voir Figure 13.9 (b)) afin de le rendre bien visible. Il est également possible d'éviter à la caméra d'être à l'intérieur ou derrière un objet en rendant cet objet transparent, comme illustré sur la Figure 13.9 (d).

Il est à noter que les calculs nécessaires à la détection d'occlusions augmente avec la complexité de la scène. De plus, le calcul précis des occultations implique beaucoup de calculs d'intersection. Afin de préserver le rendu temps-réel, nous utilisons actuellement une méthode simple basée sur deux tests d'intersection. Le premier test évalue la visibilité du centre du *widget* depuis la caméra virtuelle(Figure 13.9 (b)). Le second test évalue la visibilité de la caméra virtuelle depuis le point de vue courant (Figure 13.9 (d)).

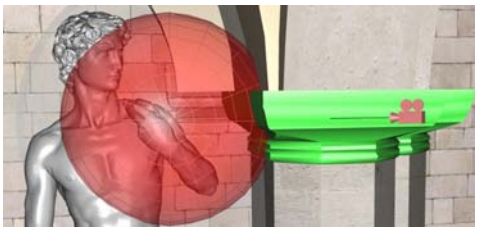
Il nous a été suggéré, lors des tests préliminaires, de déplacer la caméra vers une position plus convenable dans le cas où une intersection apparaîtrait. Nous avons choisi de ne pas implémenter cette fonctionnalité, car cela réduirait le niveau de contrôle offert à l'utilisateur. De plus, les choix effectués par le système pourraient ne pas être bons, par exemple une intersection détectée à tort, et il serait perturbant pour l'utilisateur de ne pas pouvoir placer la caméra où il le souhaite à cause d'imprécisions de calculs.

13.3 Étude utilisateur

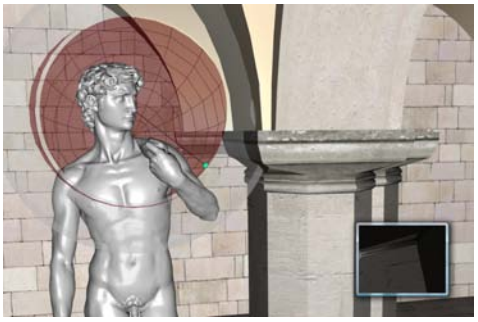
Dans cette section, nous décrivons une étude préliminaire que nous avons menée afin d'évaluer l'utilisabilité de Navidget. Les retours que nous avons reçus des utilisateurs de cette étude nous ont permis d'améliorer et de faire progresser Navidget vers sa version actuelle. Pour nos expérimentations, nous avons utilisé un Tablet PC. Notre étude se concentrait sur des tâches où Navidget peut être utile, par exemple lorsque l'utilisateur souhaite visualiser une zone particulière d'une scène 3D.



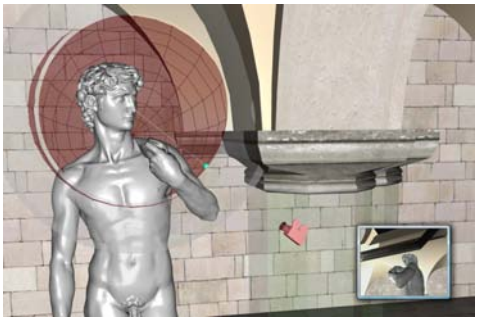
(a) Le point cible n'est pas visible depuis la caméra virtuelle.



(b) La *caméra intelligente* met en valeur l'objet occultant.



(c) La caméra virtuelle n'est plus visible depuis le point de vue courant.



(d) La *Caméra Intelligente* masque l'objet occultant.

FIGURE 13.9 – Lorsque des occultations interviennent, la *caméra intelligente* permet de mettre en valeur ces occultations (b) ou de supprimer les objets occultant (d).

13.3.1 Commentaires généraux

Le but principal de cette étude était d'évaluer l'utilisabilité de chaque contrôle de Navidget. De plus, nous voulions avoir une idée des préférences des utilisateurs entre Navidget et les contrôles standards, à savoir *fly*, *pan*, *look around*, *go to* et *orbit*. Nous avons donc mis en place une expérimentation où les sujets devaient compléter des tâches de positionnement de caméra. Après avoir achevé toutes les tâches, nous avons demandé aux utilisateurs de répondre à un questionnaire concernant l'utilisabilité de Navidget et des contrôles standard. Les réponses à ce questionnaires utilisaient une échelle de Likert en 4 points (0 = Pas du tout d'accord, 3 = Tout à fait d'accord).

Dans cette expérimentation préliminaire, nous n'avons pas considéré le temps comme critère d'évaluation de Navidget face aux contrôles standard. En effet, le temps nécessaire à l'accomplissement d'une tâche dépend énormément de la vitesse de déplacement choisie dans le paramétrage de Navidget. Cela dépend également de la sensibilité choisie pour les contrôles standards. Par conséquent, les résultats auraient été fortement dépendants des paramètres initiaux.

13.3.2 Procédure

30 sujets, divisés en deux groupes, ont participé à cette étude (21 hommes, 9 femmes, moyenne d'âge = 26 ans). Pour le premier groupe, 15 experts en 3D ont été recrutés. Le second groupe était composé de 15 novices (c'est à dire qui jouent aux jeux vidéos ou qui

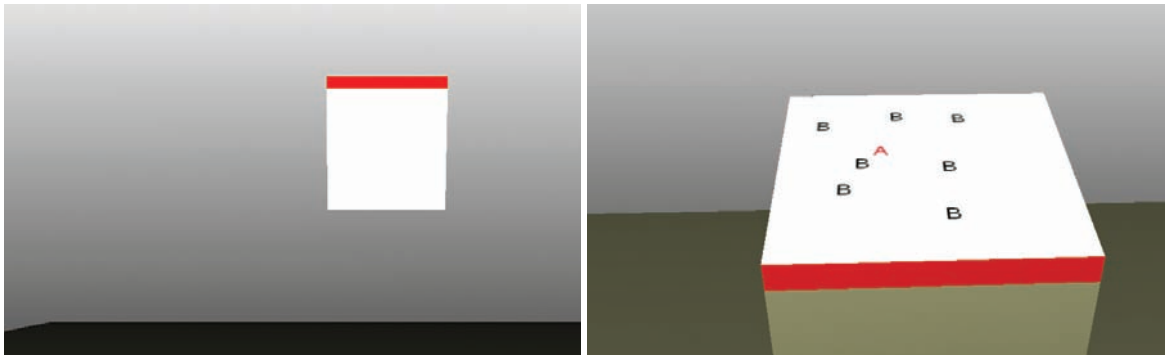


FIGURE 13.10 – Exemple de scène expérimentale utilisée pour l'étude préliminaire. Depuis la vue initiale (*gauche*), les sujets devaient se rapprocher de la face mise en valeur (*droite*) afin de compter le nombre de "A" présents sur la face.

utilisent une application 3D moins d'une fois par mois). Pour chaque groupe, la moitié des sujets a effectué les tâches à l'aide de Navidget puis des contrôles standards. L'autre moitié a commencé avec les contrôles standards.

Les scènes de l'étude étaient composées de cubes dont une face était mise en valeur. Sur cette face, des lettres (des "A" rouges et des "B" noirs) étaient écrites. La tâche consistait à compter le nombre de "A". Pour compléter la tâche, les sujets devaient se déplacer afin d'obtenir une vue appropriée (voir Figure 13.10). Un essai était achevé à partir du moment où le sujet répondait à la question en sélectionnant une réponse sur un pavé numérique.

Avant le début de l'expérimentation, les deux interfaces étaient présentées aux sujets. Ils ont eu la possibilité d'essayer chaque contrôle à l'aide d'une scène de test.

Durant l'expérimentation qui utilisait Navidget, il était demandé explicitement aux sujets d'utiliser un contrôle particulier de Navidget (c'est à dire *pointer*, *encercler*, *pointer* + widget, *encercler* + widget et *widget* + *contrôleurs de taille*). Chaque contrôle a donc été utilisé trois fois pour des positions et des tailles de cible différentes.

13.3.3 Résultats

L'analyse du questionnaire a montré que l'utilisabilité générale de Navidget est bonne. Les experts et les novices n'ont pas eu de difficulté à utiliser les contrôles de Navidget (voir Table 13.2). La métaphore de l'encerclement a été appréciée. En effet, le lien entre encercler et mettre en valeur une zone est très fort. Par conséquent, cette métaphore a été très souvent utilisée par les sujets, même si un simple pointage aurait été suffisant. De même, esquisser le *widget* à l'aide d'un geste simple a bien été apprécié. Le lien entre le dessin et le *widget* correspondant a bien été compris par les sujets. Ils ont pu définir leur orientation de vue facilement. Ils ont rapidement compris le rôle de la demi-sphère. De même, ils ont utilisé les *contrôleurs de taille* efficacement, et ce dès la première utilisation.

Cette étude préliminaire a montré que la grande majorité des sujets ont préféré utiliser Navidget plutôt que les contrôles standard pour les tâches proposées. Ceci est visible sur les résultats de la Figure 13.11.

Un test des rangs signés de Wilcoxon a montré que cette préférence était significative

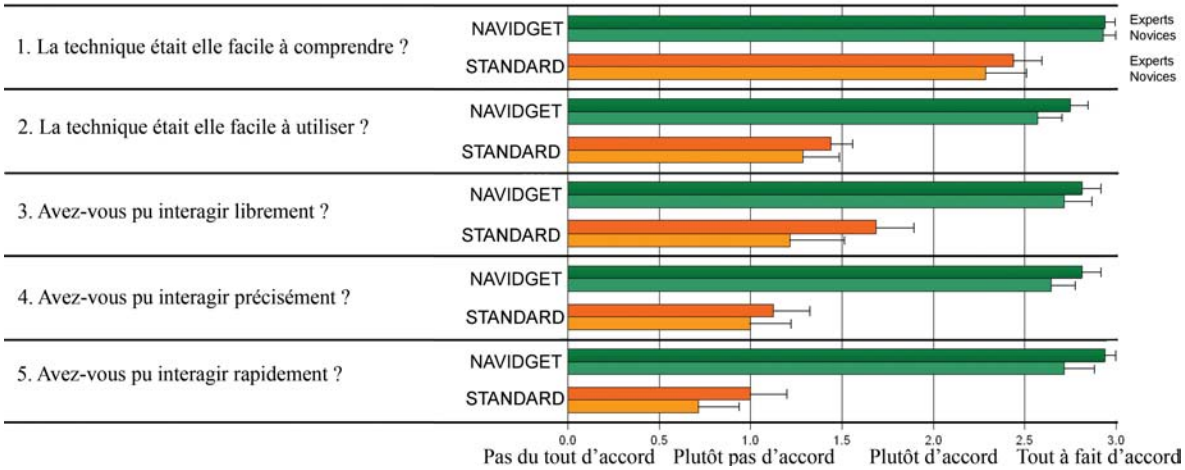


FIGURE 13.11 – Satisfaction générale des utilisateurs pour Navidget pour les contrôles standard.

	Encercler	*widget*	Taille	Encercler + *widget*
La technique était elle facile à comprendre ?	3.0, 2.9 *(3.0)*	2.9, 2.8 *(2.8)*	2.6, 2.6 *(2.6)*	3.0, 2.9 *(3.0)*
La technique était elle facile à utiliser ?	2.9, 2.9 *(2.9)*	2.7, 2.7 *(2.7)*	2.2, 2.4 *(2.3)*	2.7, 2.8 *(2.8)*
Avez vous pu interagir librement ?	2.9, 2.9 *(2.9)*	2.5, 2.9 *(2.7)*	2.3, 2.8 *(2.5)*	2.9, 2.9 *(2.9)*
Avez vous eu l'impression d'être précis ?	2.9, 2.6 *(2.7)*	2.6, 2.7 *(2.6)*	2.3, 2.8 *(2.5)*	2.6, 2.8 *(2.7)*
Avez vous eu l'impression d'être rapide ?	2.8, 2.9 *(2.8)*	2.8, 2.9 *(2.8)*	2.4, 2.6 *(2.5)*	2.7, 2.9 *(2.8)*

TABLE 13.2 – Satisfaction pour chaque contrôle de Navidget [Novices, Experts *(moyenne)*]. (0 = Pas du tout d'accord, 3 = Tout à fait d'accord).

pour les questions que nous avons posées, aussi bien pour les experts que pour les novices, sauf pour la question concernant la compréhension de la technique. En effet, les deux interfaces ont été bien comprises par les sujets.

Parmi ces résultats, il est intéressant de noter que les experts et les novices ont trouvé que Navidget était plus facile à utiliser que les contrôles standard (Expert *test Z de Wilcoxon : -3.52,* $p < 0.05$; Novices *test Z de Wilcoxon : -2.99,* $p < 0.05$) Ils ont également trouvé Navidget plus rapide (Expert *test Z de Wilcoxon : -3.45* $p<.05$; Novice *test Z de Wilcoxon : -3.34,* $p<.05$).

Avec Navidget, une action simple est utilisée pour compléter la tâche. La même tâche nécessite l'utilisation de plusieurs contrôles avec les techniques standards, ce qui peut prendre du temps.

13.3.4 Discussion

Les utilisateurs novices ont eu quelques difficultés en utilisant les techniques standards, avec lesquelles ils n'étaient pas familiers. En particulier, ils devaient se souvenir de l'action de chaque contrôle. Souvent, ils essayaient un outil sans savoir quel mouvement de caméra il allait produire dans la scène 3D. Ils se sont souvent retrouvés perdus.

Parallèlement à cela, avec Navidget, les sujets ont aimé manipuler une technique où plusieurs paramètres pouvaient être contrôlés à l'aide d'un geste simple. Contrairement aux contrôles standard, où les deux degrés de liberté du périphérique d'entrée contrôlent direc-

tement deux degrés de liberté de la caméra (par exemple, translation selon l'axe Z + rotation autour de l'axe Y pour la technique de *fly*), les gestes 2D de Navidget permettent de réaliser des mouvements de caméra élémentaires, mais qui réclament le contrôle de plusieurs DDL simultanément. Le retour visuel que fournit Navidget favorise une interaction efficace. Durant l'expérimentation, les sujets savaient exactement vers où la caméra allait se déplacer. Ils étaient également capables de définir où ils souhaitaient aller.

13.4 Diffusion de Navidget

Nous pensons que Navidget peut être utile dans beaucoup de situations différentes, aussi bien pour des experts que pour des novices de l'interaction 3D. Nous avons ainsi diffusé la technique Navidget à travers plusieurs voies.

Tout d'abord, afin de permettre l'intégration de Navidget dans d'autres applications, nous avons développé une bibliothèque C++. Nous avons également créé des *plugins* pour deux logiciels de modélisation : Blender et SketchUp. Enfin, Navidget a été utilisé dans plusieurs démonstrateurs de projets ANR dans lesquels nous sommes impliqués.

Ces différentes diffusions ont permis de développer de nouvelles utilisations possibles de Navidget, et d'étendre les possibilités de cette technique.

13.4.1 LibNavidget

Afin de permettre à n'importe quel développeur d'intégrer Navidget dans son application, nous avons développé une *bibliothèque* appelée "LibNavidget", écrite en C++ et utilisable avec OpenGL et OpenGL|ES. Cette bibliothèque a été compilée pour Windows, Windows Mobile et Linux et diffusée à travers un site web dédié [1]. Le code source de cette bibliothèque a fait l'objet d'un dépôt APP (Agence de Protection des Programmes) sous le numéro IDDN : 09-090033.000.

L'idiome "pimpl" (*Private IMPLementation*) a été utilisé afin de ne fournir à l'utilisateur que les fonctions nécessaires. Il permet de masquer l'architecture interne de Navidget, et de ne rendre visible que les fonctions essentielles. Cela permet également de garder une API propre et constante, afin que les utilisateurs n'aient pas à faire de modification dans leur code en cas de nouvelle version.

L'architecture est présentée sur la Figure 13.12 :

- La classe **NvgCam** gère les mouvements de caméra.
- La classe **NvgGest** analyse les mouvements de la souris et détecte les gestes de l'utilisateur.
- La classe **NvgGraph** est dédiée à l'affichage du *widget*.
- La classe **NvgImpl** est le contrôleur, et gère les différents états en fonction des évènements reçus par les autres classes.
- Enfin, la classe **Navidget**, qui utilise l'idiome "pimpl", représente la partie publique de LibNavidget.

1. http ://iparla.labri.fr/software/navidget/

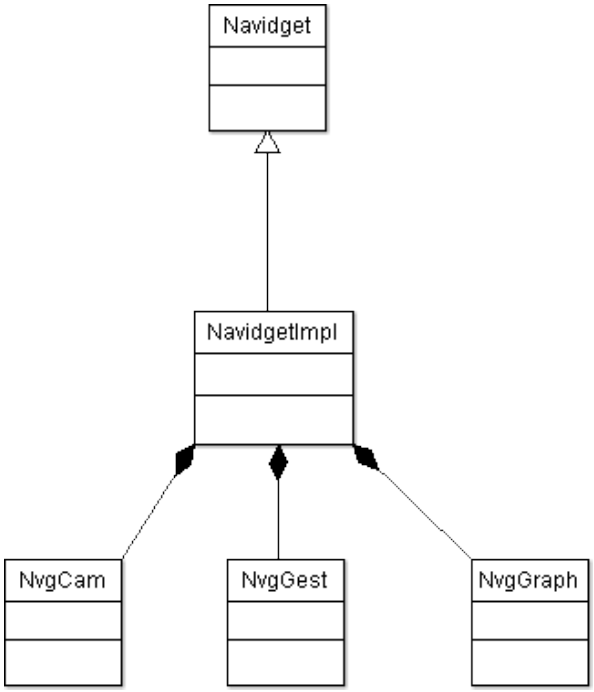


FIGURE 13.12 – Architecture de Navidget, utilisant l'idiome "pimpl".

L'API visible par l'utilisateur ne possède que quelques fonctions :

- `void init( void )` permet d'initialiser Navidget.
- `void setMouse( int x, int y, int button, bool pressed )` permet d'envoyer les évènements souris à Navidget.
- `void setMouseMotion( int x, int y )` permet d'envoyer les déplacements de la souris à Navidget.
- `void draw()` permet de dessiner le *widget*.
- `void draw( bool sphere, bool cam, bool cursor, bool actuator )` permet de dessiner le *widget* en sélectionnant les parties à afficher.
- `void getCamera( Camera &c )` permet de récupérer la position et l'orientation de la caméra calculée par Navidget.
- `void setCamera( const Camera &c )` permet d'informer Navidget d'un changement de caméra par l'application.
- `void getPreviewCamera( Camera &c )` permet de récupérer la position et l'orientation de la caméra virtuelle, afin d'afficher la fenêtre de prévisualisation par exemple.

A ce jour, la démonstration de Navidget a été téléchargée 240 fois, la bibliothèque 180 fois. Cette bibliothèque a été utilisée par des membres de notre équipe afin d'intégrer Navidget dans des démonstrateurs pour des projets ANR dans lesquels nous étions impliqués.

13.4.2 Utilisation dans des démonstrateurs de projets ANR

Navidget a été utilisé dans trois démonstrateurs pour des projets ANR. Ces projets nous ont permis de spécialiser Navidget pour des tâches pour lesquelles il n'était pas initialement destiné, ce qui montre que l'approche proposée par Navidget peut aider les utilisateurs dans beaucoup de situations.

13.4.2.1 Projet ANR RaxEnv

Le projet RaxEnv [2] a pour but de démontrer la faisabilité d'un système de réalité augmentée en extérieur dans le domaine des sciences et techniques de l'environnement. Dans ce projet, des modèles numériques sont superposés aux images réelles en temps réel, afin de permettre la visualisation de différents types de données : données géologiques, mesures, informations sur les matériaux utilisés, etc.

Dans ce projet, il est primordial d'offrir à l'utilisateur des moyens simples pour se déplacer et observer les modèles numériques affichés. En effet, les applications développées lors de ce projet sont destinées à être utilisées par des professionnels spécialisés, et non par des informaticiens ou des graphistes. Il faut donc fournir des techniques d'interaction les plus intuitives et simples possibles. Navidget a été intégré au démonstrateur, en temps que technique de navigation. Le démonstrateur est destiné à être utilisé sur un tabletPC, l'interaction se faisant alors à l'aide d'un stylet, ce qui permet d'utiliser Navidget de manière optimale.

13.4.2.2 Projet ANR Part@ge

Le projet Part@ge [3] a pour but de développer l'aspect collaboratif de l'interaction 3D. Dans ce projet, notre équipe a été sollicitée afin d'étudier l'utilisation de terminaux mobiles dans de tels contextes de travail. Un des démonstrateurs du projet montre la possibilité de faire collaborer plusieurs utilisateurs possédant des périphériques et des dispositifs hétérogènes : regroupés dans une salle de travail disposant d'un écran de RV immersive, certains utilisateurs disposent de stations de travail, d'autres de terminaux mobiles. Ces derniers reçoivent sur leur écran un aperçu du modèle 3D projeté sur l'écran de la salle. Il peuvent alors, à l'aide Navidget, spécifier un point de vue dont ils recevront l'aperçu une fois le geste terminé.

Ce projet nous a permis de développer, avec l'aide de Mariam Amyra (ingénieur au sein de l'équipe Iparla), une version *distante* de Navidget, qui ne nécessite pas d'afficher la scène 3D sur le terminal. Dans cette approche, la scène 3D est stockée et affichée sur un serveur. Lorsqu'un client se connecte, il ne reçoit qu'une capture d'écran de ce qu'affiche le serveur. Cela permet aux terminaux mobiles de bénéficier d'un rendu de bonne qualité.

Lorsqu'un utilisateur interagit à l'aide de Navidget, il reçoit une image correspondant à son point de vue, ainsi qu'une carte de profondeurs. Il est alors possible d'utiliser la version "distante" de Navidget afin de spécifier un point de vue sur le terminal mobile. Pendant qu'il manipule Navidget sur son terminal, un aperçu est affiché sur l'écran principal de la salle. Cela permet à plusieurs personnes de montrer leur point de vue en même temps (voir Figure 13.13).

2. http ://raxenv.brgm.fr/
3. http ://partage.ingenierium.com/

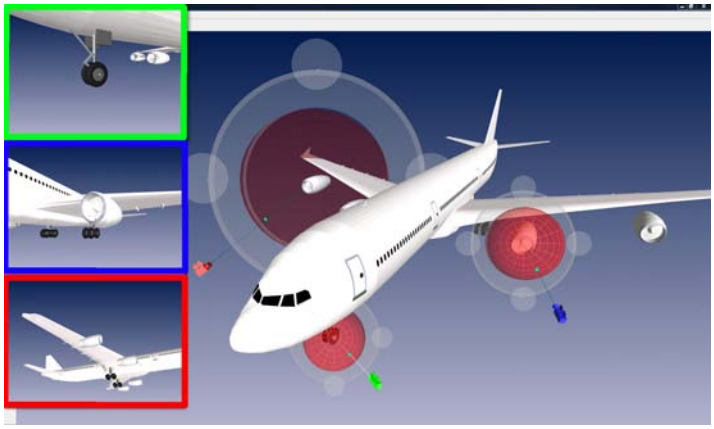


FIGURE 13.13 – Utilisation de plusieurs Navidget pour des tâches collaboratives.

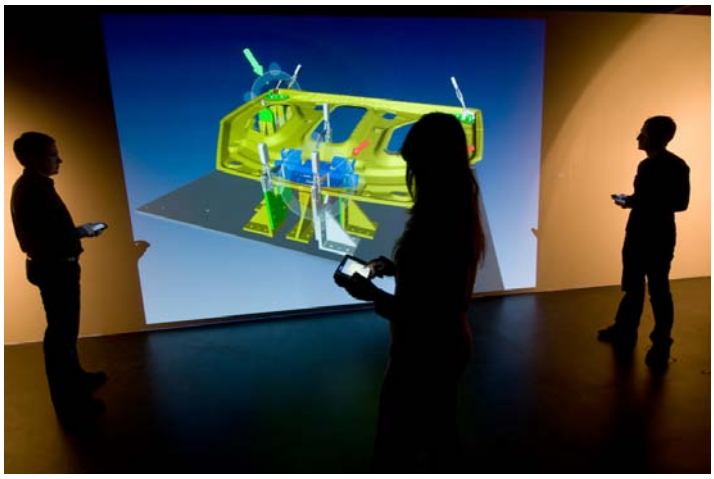


FIGURE 13.14 – Utilisation de Navidget pour des tâches de pointage.

Cette démonstration a également été utilisée lors du Salon du Bourget 2009, afin de montrer l'intérêt de la RV lors des phases de conception et d'étude de constructions de l'industrie aéronautique.

13.4.2.3 Projet ANR Dalia

Le projet Dalia [4] se concentre sur la visualisation, l'interaction et la collaboration dans des environnements hétérogènes. Pour ce projet, Navidget a été utilisé pour des tâches de *pointage*. En effet, les degrés de liberté que permet de manipuler Navidget sont tout à fait compatibles avec la représentation d'un vecteur tridimensionnel.

Ce travail a été réalisé par Benoît Bossavit (ingénieur au sein de l'équipe Iparla) à partir de la bibliothèque LibNavidget que nous avons développée. Un exemple de pointage à l'aide de Navidget est visible sur la Figure 13.14.

4. http ://dalia.gforge.inria.fr/

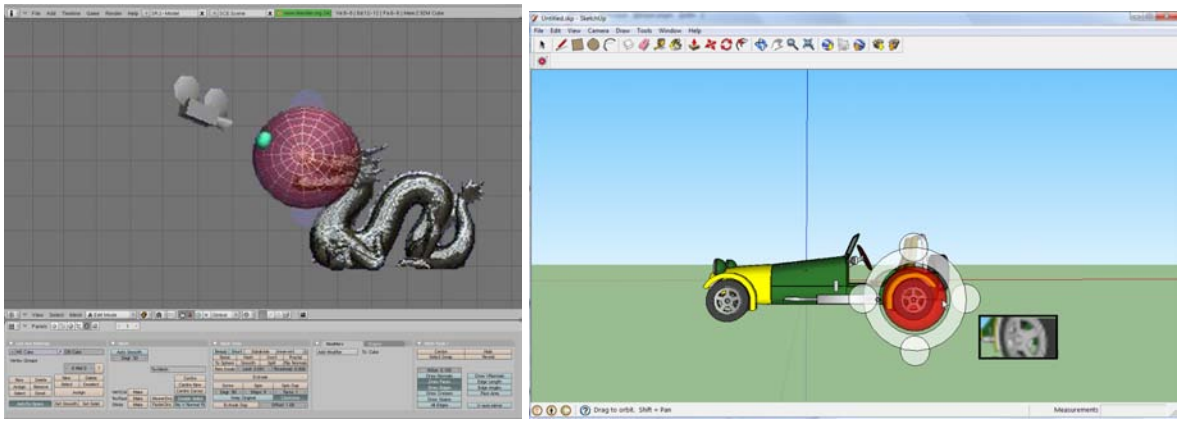


FIGURE 13.15 – Intégration de Navidget dans les logiciels Blender et SketchUp.

13.4.3 Intégration dans des logiciels de modélisation 3D

Afin de permettre une diffusion plus importante, et surtout d'offrir la possibilité d'utiliser Navidget à un plus grand nombre de personnes, nous avons encadré des projets étudiants visant à intégrer Navidget dans des logiciels de modélisation 3D du commerce. Cette intégration s'est effectuée à travers le développement d'un script pour le logiciel Blender et d'un script pour le logiciel Sketchup de Google.

13.4.3.1 Intégration de Navidget dans Blender

Le travail d'intégration de Navidget dans le logiciel Blender a été réalisé par quatre étudiants de Master 1 Informatique (Yassine El Ouardani, Pierre Faure, Maher Finianos et Pierre Rouanet) lors du module "projet de programmation". Leur travail a permis de développer un système de reconnaissance de gestes basé sur le calcul de la "distance de Levenshtein", qui permet de trouver quelle forme a été tracée par l'utilisateur. Ils ont également réalisé une version simplifiée de Navidget, sous la forme d'un script écrit en Python, permettant d'utiliser cette technique dans le logiciel Blender, visible sur la Figure 13.15.

13.4.3.2 Intégration de Navidget dans SketchUp

SketchUp de Google est un logiciel de modélisation très intéressant. Son approche est basée sur l'utilisation de gestes simples, et assiste l'utilisateur dans ses tâches de modélisation. Nous avons ainsi encadré quatre étudiants de Master 2 Multimédia (Anthony Battel, Jonathan Depiets, Benjamin Orsini, David Palanchon) afin d'intégrer Navidget dans ce logiciel. La majorité des fonctionnalités de Navidget ont été intégrées au module développé. L'API de SketchUp ne permettant pas de créer plusieurs *viewports*, une astuce a été utilisée afin de permettre de bénéficier de la fenêtre de prévisualisation : un rendu est effectué depuis la caméra virtuelle de Navidget et enregistré dans un fichier temporaire. Il suffit alors de texturer un polygone situé en face de l'utilisateur avec cette image. Cette technique étant assez lente, la qualité de l'image enregistrée et affichée a été réduite, comme nous pouvons le voir sur la Figure 13.15.

13.5 Conclusion

La technique de navigation Navidget possède beaucoup d'avantages par rapport aux techniques de "go to" classiques. Elle offre un contrôle total de la distance t de l'angle avec lesquels observer un point cible d'une scène 3D. Les différents tests utilisateurs réalisés ont montré qu'aussi bien les novices que les experts de l'interaction 3D trouvent un grand intérêt à utiliser Navidget. Cette technique est également intéressante car elle est utilisable sur n'importe quel appareil, avec une large gamme de périphérique. Ainsi, Knödel *et al.* (2008) ont développé une version immersive de Navidget qui utilise des capteurs à 3 DDL afin de sélectionner le point cible et d'interagir avec le *widget*. Ainsi, les utilisateurs n'ont pas besoin d'apprendre de nouvelles techniques d'interaction, et peuvent bénéficier du même outil quels que soient le périphérique et la configuration utilisés.

CHAPITRE 14

Conclusion de la Partie III

Cette troisième partie du mémoire a été consacrée à l'étude de l'interaction 3D à l'aide d'un contrôle planifié. Ce type de contrôle possède des avantages indéniables. Il permet à l'utilisateur de se concentrer sur le but de sa tâche, en s'affranchissant des transformations impliquées. Les expérimentations que nous avons menées nous ont permis de montrer que, pour certaines tâches simples, un contrôle planifié pouvait tout à fait être utilisé au lieu d'un contrôle direct.

Le principal inconvénient de ce genre d'approche est qu'il offre moins de liberté que le contrôle direct, car l'interaction y est bien souvent discrétisée. Il sera important, lors du développement de techniques d'interaction 3D utilisant un contrôle planifié, de trouver le bon compromis entre facilité d'utilisation et liberté offerte à l'utilisateur.

Des techniques telles que Navidget, présentée au chapitre 13, sont tout à fait adaptées à une utilisation sur un terminal mobile. Elles offrent une grande liberté d'action tout en étant simples à manipuler. Nous avons été sollicités pour présenter cette technique dans plusieurs salons, destinés aussi bien aux novices qu'aux professionnels, et l'approche a séduit beaucoup de ces personnes.

Dans le futur, nous pensons continuer nos investigations dans ce sens, afin de proposer de nouvelles techniques d'interaction qui soient utilisables par des novices, tout en offrant assez de contrôles pour être utile aux experts de l'interaction 3D.

Conclusion générale

Dans ce mémoire, nous avons abordé l'interaction 3D sur terminaux mobiles à travers deux approches. D'une part, le contrôle direct et d'autre part le contrôle planifié. Ce premier mode d'interaction est le plus couramment employé dans les applications 3D interactives telles que les jeux vidéo ou les logiciels de visualisation de modèles 3D. Ce deuxième mode offre souvent une interaction d'un plus haut niveau, où l'utilisateur reste concentré sur sa tâche, et non sur les outils qui lui permettent d'accomplir sa tâche.

Contributions

Durant cette thèse, nous avons développé plusieurs techniques d'interaction 3D, principalement pour la tâche de navigation. Ces techniques sont le fruit de réflexions concernant une des principales caractéristiques des terminaux mobiles : les périphériques d'entrée. En effet, les premiers terminaux mobiles ne possédant qu'un pavé numérique comme seul moyen d'interaction, nous nous sommes intéressés à la navigation à base de touches avec la technique du ZGoto (Hachet *et al.*, 2006; Decle *et al.*, 2006). Cette technique permet de sélectionner un point de l'espace 3D afin de se déplacer vers celui-ci. Le principe général de cette technique est de discrétiser les profondeurs de la vue courante en *sections* afin de réaliser une interaction en deux temps : une fois une section sélectionnée, le curseur 3D peut être déplacé le long de sa bordure. La sélection du point cible étant réalisée en fonction des éléments visibles, et non sur le plan écran comme cela se fait habituellement, les performances des utilisateurs sont meilleures.

Quelques temps après le début de cette thèse, les écrans tactiles se sont développés et répandus assez rapidement sur le marché. Nous avons alors proposé Navidget (Hachet *et al.*, 2008, 2009), qui peut être considéré comme un "go to" avancé, où l'utilisateur peut, à l'aide de quelques gestes simples, définir son point ou sa zone d'intérêt ainsi que l'angle et la distance de la vue cible par rapport à ce point. Cette technique a été présentée à la conférence 3DUI 2008 et a obtenu le prix du meilleur article.

Utilisant lui aussi l'écran tactile, ScrutiCam (Decle *et al.*, 2009) permet, lui, un contrôle direct semi-automatique, et se situe donc à mi-chemin entre une interaction libre et un mouvement automatique. Cette technique a été étudiée pour des tâches d'inspection et se base sur un algorithme qui analyse le voisinage d'un point cible sélectionné par l'utilisateur afin de proposer une vue adaptée. Le contrôle de la vue est réalisé par de simples glisser-

déposers sur l'écran, dont la direction déterminera le type de mouvements de caméra : un geste vers le centre de l'écran déplace la caméra vers la vue calculée par l'algorithme, tandis qu'un déplacement dans une autre direction permet de translater la caméra parallèlement au plan écran.

Étant persuadés de l'apport que pouvait représenter le contrôle planifié, nous avons étudié les différences existantes entre un contrôle direct et un contrôle planifié, ce qui a permis de publier l'étude présentée au chapitre 11 à MobileHCI 2009 (Decle et Hachet, 2009). Cette étude montre que, malgré des performances temporelles légèrement inférieures, il pouvait être intéressant d'utiliser un contrôle planifié plutôt qu'un contrôle direct sur un terminal mobile. Les sujets de cette étude ont pour la plupart préféré utiliser la technique planifiée, qui leur permettait de moins se perdre qu'avec une technique utilisant un contrôle direct.

Enfin, l'avènement des écrans tactiles multi-points nous a amené à nous demander si un tel périphérique pouvait améliorer les performances des utilisateurs pour des tâches simples de contrôle de vitesse. L'étude exploratoire que nous avons menée, présentée dans les chapitres 8 et 7, nous a montré que certains mouvements pouvaient être difficiles ou pénibles à réaliser, ce qui illustre la nécessité d'étudier attentivement les contraintes techniques de l'appareil ainsi que les contraintes anatomiques du corps humain lors du développement d'une technique d'interaction.

Toutes ces contributions sont résumées dans la Figure 14.1, qui classe ces techniques en fonction de la liberté de contrôle qu'elles offrent.

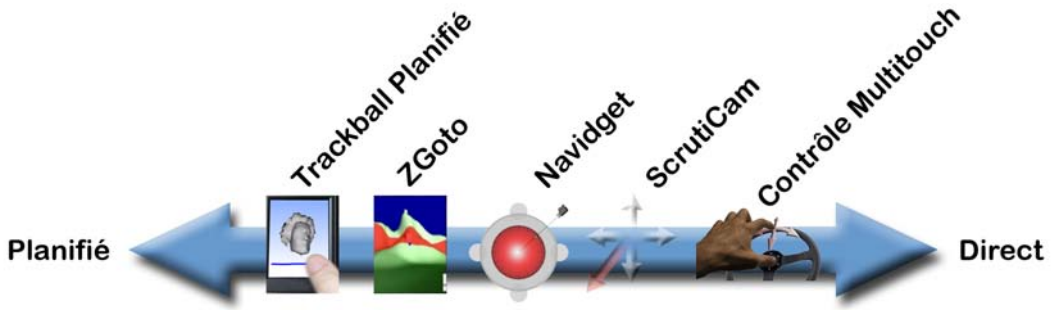


FIGURE 14.1 – Résumé des techniques proposées dans cette thèse, classées en fonction du degré de contrôle qu'elles offrent.

Perspectives

Le monde des terminaux mobiles est en constante évolution. De nouvelles fonctionnalités apparaissent chaque jour, ou presque, et permettent d'imaginer une multitude de nouvelles techniques d'interaction tirant parti de ces nouveautés. Parmi ces évolutions, le développement récent des *picoprojecteurs* qui, comme leur nom le suggère, sont des projecteurs vidéo miniatures, permet d'imaginer de nouvelles façons d'interagir en libérant totalement l'écran du terminal. L'image de l'appareil pourra alors être projetée sur n'importe quelle surface dis-

ponible : mur, sol, plafond, etc. On peut également penser que cela permettra d'avoir deux types d'informations simultanément : le projecteur affiche le modèle 3D à manipuler, tandis que l'écran de l'appareil donne des informations textuelles sur la pièce projetée.

Les écrans multi-points promettent également un bel avenir à l'interaction 3D. En effet, les premières applications de manipulation 2D utilisant ces écrans possèdent l'avantage d'offrir une interaction très intuitive, presque naturelle. Il y a fort à parier que ce type d'écrans peut également représenter un apport non négligeable pour l'interaction 3D.

Enfin, n'oublions pas que beaucoup de recherches peuvent encore être effectuées concernant l'interaction 3D en utilisant les périphériques actuels. L'approche que nous avons eue pour ScrutiCam, qui consiste à analyser la vue, les informations de profondeur ou la présence de contours, représente certainement une voie intéressante. Cela permet de proposer des vues pertinentes et adaptées au contexte tout en limitant les calculs géométriques nécessaires.

Pour conclure, cette thèse m'a permis d'explorer un domaine assez vaste qu'est l'interaction 3D sur terminaux mobiles. Si beaucoup de travaux ont été réalisés sur l'interaction 3D sur des stations de travail, peu se sont attelés à cette tâche sur des terminaux mobiles. En analysant les spécificités et les contraintes de ces appareils, nous avons proposé des techniques adaptées, ainsi que des conseils concernant le développement d'autres techniques d'interaction 3D. Le monde de la recherche est un univers passionnant, plein de découvertes et de soif de connaissances. Les compétences acquises durant ces trois années de thèse me permettent d'appréhender avec beaucoup de sérénité les différents problèmes scientifiques que je pourrais rencontrer dans mon futur parcours professionnel.

Références bibliographiques

ISO/IEC 9995-8 : Information systems - keyboard layouts for text and office systems - part 8 : Allocation of letters to the keys of a numeric keypad, international organisation for standardisation,, 1994.

María J. ABÁSOLO et José Mariano DELLA : Magallanes : 3D navigation for everybody. *In GRAPHITE '07 : Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, pages 135–142, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-912-8.

Ragnar BADE, Felix RITTER et Bernhard PREIM : Usability comparison of mouse-based interaction techniques for predictable 3d rotation. *In Smart Graphics*, pages 138–150, 2005.

Ravin BALAKRISHNAN et Ken HINCKLEY : The role of kinesthetic reference frames in two-handed input performance. *In UIST '99 : Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 171–178, New York, NY, USA, 1999. ACM. ISBN 1-58113-075-9.

Ravin BALAKRISHNAN et Ken HINCKLEY : Symmetric bimanual interaction. *In CHI '00 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 33–40, New York, NY, USA, 2000. ACM. ISBN 1-58113-216-6.

Ravin BALAKRISHNAN et Gordon KURTENBACH : Exploring bimanual camera control and object manipulation in 3D graphics interfaces. *In CHI '99 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 56–62, New York, NY, USA, 1999. ACM. ISBN 0-201-48559-1.

Francesca A. BARRIENTOS et John F. CANNY : Cursive : : controlling expressive avatar gesture using pen gesture. *In CVE '02 : Proceedings of the 4th international conference on Collaborative virtual environments*, pages 113–119, New York, NY, USA, 2002. ACM. ISBN 1-58113-489-4.

Harlen Costa BATAGELO et Shin-Ting WU : A framework for gpu-based application-independent 3D interactions. *Vis. Comput.*, 24(12):1003–1012, 2008. ISSN 0178-2789.

Patrick BAUDISCH et Gerry CHU : Back-of-device interaction allows creating very small touch devices. *In CHI '09 : Proceedings of the 27th international conference on Human factors in*

computing systems, pages 1923–1932, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-246-7.

Benjamin B. BEDERSON : Fisheye menus. *In UIST '00 : Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 217–225, New York, NY, USA, 2000. ACM. ISBN 1-58113-212-3.

Hrvoje BENKO et Steven FEINER : Balloon selection : A multi-finger technique for accurate low-fatigue 3d selection. *In Proceedings of Symposium on 3D User Interfaces (3DUI) 2007*, 2007.

Hrvoje BENKO, Andrew D. WILSON et Patrick BAUDISCH : Precise selection techniques for multi-touch screens. *In CHI '06 : Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1263–1272, New York, NY, USA, 2006. ACM. ISBN 1-59593-372-7.

Eric A. BIER : Skitters and jacks : Interactive 3D positioning tools. *In Workshop on Interactive 3D Graphics*, pages 183–196. ACM, 1986.

Eric A. BIER : Snap-dragging in three dimensions. *In Symposium on Interactive 3D Graphics*, pages 193–204. ACM, 1990.

M. BILLINGHURST, H. KATO, S. WEGHORST et TA FURNESS : A mixed reality 3D conferencing application. *Human Interface Technology Laboratory, University of Washington, Technical Report R-99-1*, 1999.

Oliver BIMBER, L. Miguel ENCARNAÇÃO et André STORK : A multi-layered architecture for sketch-based interaction within virtual environments. *Computers & Graphics*, 24(6):851–867, 2000.

Nick BOBICK : Rotating objects using quaternions. *Game Developer*, 2(26):21–31, 1998.

Richard A. BOLT : "put-that-there" : Voice and gesture at the graphics interface. *In SIGGRAPH '80 : Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 262–270, New York, NY, USA, 1980. ACM Press. ISBN 0-89791-021-4.

Doug A. BOWMAN, Donald B. JOHNSON et Larry F. HODGES : Testbed evaluation of virtual environment interaction techniques. *Presence*, 10(1):75–95, 2001a.

Doug A. BOWMAN, David KOLLER et Larry F. HODGES : Travel in immersive virtual environments : An evaluation of viewpoint motion control techniques. *In IEEE, Proceedings, 1997 Virtual Reality Annual International Symposium*, 1997.

Doug A. BOWMAN, Ernst KRUIJFF, Joseph J. LAVIOLA et Ivan POUPYREV : An introduction to 3D user interface design. *Presence 10 (1)*, 2001b.

Doug A. BOWMAN, Ernst KRUIJFF, Joseph J. LAVIOLA et Ivan POUPYREV : *3D User Interfaces : Theory and Practice*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004. ISBN 0201758679.

Lars BRETZNER et Tony LINDEBERG : Use your hand as a 3d mouse, or, relative orientation from extended sequences of sparse point and line correspondences using the affine trifocal tensor. *Lecture Notes in Computer Science*, 1406:141– ? ?, 1998.

Stephen BREWSTER : Overcoming the lack of screen space on mobile computers. *Personal Ubiquitous Comput.*, 6(3):188–205, 2002. ISSN 1617-4909.

Nicholas BURTNYK, Azam KHAN, George W. FITZMAURICE, Ravin BALAKRISHNAN et Gordon KURTENBACH : Stylecam : interactive stylized 3D navigation using integrated spatial & temporal controls. *In Proceedings of the 15th annual ACM symposium on User interface software and technology UIST '02*, pages 101–110. ACM Press, 2002.

William BUXTON : Chunking and phrasing and the design of human-computer dialogues (invited paper). *In IFIP Congress*, pages 475–480, 1986.

William BUXTON et Brad A. MYERS : A study in two-handed input. *In CHI '86 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 321–326, New York, NY, USA, 1986. ACM Press. ISBN 0-89791-180-6.

Bas E. CANCRINUS : Navigation of 3D worlds on mobile devices. Mémoire de D.E.A., Faculty of Science, University of Amsterdam, The Netherlands, 2004.

Tolga K. CAPIN, Antonio HARO, Vidya SETLUR et Stephen WILKINSON : Camera-based virtual environment interaction on mobile devices. *In ISCIS*, pages 765–773, 2006.

Géry CASIEZ et Daniel VOGEL : The effect of spring stiffness and control gain with an elastic rate control pointing device. *In CHI '08 : Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1709–1718, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-011-1.

Géry CASIEZ, Daniel VOGEL, Qing PAN et Christophe CHAILLOU : Rubberedge : reducing clutching by combining position and rate control with elastic feedback. *In UIST '07 : Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 129–138, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-679-2.

Michael CHEN, S. Joy MOUNTFORD et Abigail SELLEN : A study in interactive 3-d rotation using 2-d control devices. *In SIGGRAPH '88 : Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 121–129, New York, NY, USA, 1988. ACM Press. ISBN 0-89791-275-6.

Luca CHITTARO : Visualizing information on mobile devices. *Computer*, 39(3):40–45, 2006. ISSN 0018-9162.

George CHRONIS et Marjorie SKUBIC : Sketch-based navigation for mobile robots. *In Fuzzy Systems, 2003. FUZZ'03. The 12th IEEE International Conference on*, volume 1, 2003.

Edward C. CLARKSON, Shwetak N. PATEL, Jeffrey S. PIERCE et Gregory D. ABOWD : Exploring continuous pressure input for mobile phones. Rapport technique, Georgia Institute of Technology, 2006.

Brookshire D. CONNER, Scott S. SNIBBE, Kenneth P. HERNDON, Daniel C. ROBBINS, Robert C. ZELEZNIK et Andries van DAM : Three-dimensional widgets. *In SI3D '92 : Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 183–188, New York, NY, USA, 1992. ACM. ISBN 0-89791-467-8.

Jean-Baptiste de la RIVIÈRE, Cédric KERVÉGANT, Emmanuel ORVAIN et Nicolas DITTLO : Cubtile : a multi-touch cubic interface. *In VRST '08 : Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, pages 69–72, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-951-7.

André DELMAS : *Voies et centres nerveux : introduction à la neurologie*. 10e édition revue et augmentée, 1981.

Alan ESENTHER et Kathy RYALL : Fluid dtmouse : better mouse support for touch-based interactions. *In AVI '06 : Proceedings of the working conference on Advanced visual interfaces*, pages 112–115, New York, NY, USA, 2006. ACM. ISBN 1-59593-353-0.

George W. FITZMAURICE, Justin MATEJKA, Igor MORDATCH, Azam KHAN et Gordon KURTENBACH : Safe 3D navigation. *In SI3D '08 : Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pages 7–15, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-983-8.

George W. FITZMAURICE, Shumin ZHAI et Mark H. CHIGNELL : Virtual reality for palmtop computers. *ACM Trans. Inf. Syst.*, 11(3):197–218, 1993. ISSN 1046-8188.

James D. FOLEY, Andries van DAM, Steven K. FEINER et John F. HUGHES : *Computer graphics : principles and practice (2nd ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990. ISBN 0-201-12110-7.

G. W. FURNAS : Generalized fisheye views. *SIGCHI Bull.*, 17(4):16–23, 1986. ISSN 0736-6906.

Mark GIBBS : Handwriting recognition : A comprehensive comparison. *Pen, March/April*, pages 31–35, 1993.

Michael GLEICHER et Andrew WITKIN : Through-the-lens camera control. *In In Proceedings of ACM SIGGRAPH Computer Graphics 1992*, volume 26, pages 331–340. ACM Press New York, NY, USA, 1992.

David GOLDBERG et Cate RICHARDSON : Touch-typing with a stylus. *In CHI '93 : Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, pages 80–87, New York, NY, USA, 1993. ACM. ISBN 0-89791-575-5.

Jun GONG et Peter TARASEWICH : Guidelines for handheld mobile device interface design. *In Proceedings of DSI 2004 Annual Meeting*, 2004.

Jérôme GROSJEAN et Sabine COQUILLART : The magic mirror : A metaphor for assisting the exploration of virtual worlds. *In Proceedings of 15th Spring Conference on Computer Graphics. Bratislava*, pages 125–129, 1999.

Tovi GROSSMAN et Daniel WIGDOR : Going deeper : a taxonomy of 3D on the tabletop. *tabletop*, 0:137–144, 2007.

Dale L. GROVER, Martin T. KING et Clifford A. KUSHLER : Reduced keyboard disambiguating computer, octobre 6 1998. US Patent 5,818,437.

Yves GUIARD : Asymmetric division of labor in human skilled bimanual action : The kinematic chain as a model. *In Journal of Motor Behaviour*, volume 19, pages 486–517, 1987.

Martin HACHET et Alexander KULIK : Elastic control for navigation tasks on pen-based handheld computers. *In Proceedings of IEEE 3DUI - Symposium on 3D User Interfaces*, 2008. to appear.

Martin HACHET, Joachim POUDEROUX et Pascal GUITTON : A camera-based interface for interaction with mobile handheld computers. *In Proceedings of I3D'05 - ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games*, pages 65–71. ACM Press, 2005a.

Martin HACHET, Joachim POUDEROUX et Pascal GUITTON : 3d elastic control for mobile devices. *IEEE Computer Graphics and Applications*, 28(4):58–62, July/August 2008.

Martin HACHET, Joachim POUDEROUX, Pascal GUITTON et Jean-Christophe GONZATO : Tangimap - a tangible interface for visualization of large documents on handheld computers. *In* Kori INKPEN et Michiel Van De PANNE, éditeurs : *Proceedings of Graphics Interface*, pages 9–15. A.K. Peters, 2005b.

Martin HACHET, Joachim POUDEROUX, Florence TYNDIUK et Pascal GUITTON : "jump and refine" for rapid pointing on mobile phones. *In CHI '07 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 167–170, New York, NY, USA, 2007. ACM Press. ISBN 978-1-59593-593-9.

Benjamin HAGEDORN et Jürgen DÖLLNER : Sketch-based navigation in 3D virtual environments. *In 8th Int. Symposium on Smart Graphics 2008*, August 2008. to appear.

Eyal HAIK, Trevor BARKER, John SAPSFORD et Simon TRAINIS : Investigation into effective navigation in desktop virtual interfaces. *In Web3D '02 : Proceedings of the seventh international conference on 3D Web technology*, pages 59–66, New York, NY, USA, 2002. ACM. ISBN 1-58113-468-1.

Jefferson Y. HAN : Low-cost multi-touch sensing through frustrated total internal reflection. *In UIST '05 : Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118, New York, NY, USA, 2005. ACM. ISBN 1-59593-271-2.

Mark S. HANCOCK, Sheelagh Carpendale CARPENDALE et Andrew COCKBURN : Shallow depth 3D interaction : Design and evaluation of one-, two-, and three-touch techniques. *In Proceedings of ACM CHI'2007 Conference on Human Factors in Computing Systems. San Jose, CA*, 2007.

Andrew J. HANSON et Eric A. WERNERT : Constrained 3D navigation with 2d controllers. *In Proceedings of Visualization'97*, pages 175–182, 1997.

Knud HENRIKSEN, Jon SPORRING et Kasper HORNBAEK : Virtual trackballs revisited. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):206–216, 2004. ISSN 1077-2626.

Ken HINCKLEY, Patrick BAUDISCH, Gonzalo RAMOS et Francois GUIMBRETIERE : Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. *In CHI '05 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 451–460, New York, NY, USA, 2005. ACM Press. ISBN 1-58113-998-5.

Ken HINCKLEY, Jeff PIERCE, Mike SINCLAIR et Eric HORVITZ : Sensing techniques for mobile interaction. *In UIST '00 : Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 91–100, New York, NY, USA, 2000. ACM Press. ISBN 1-58113-212-3.

Ken HINCKLEY, Joe TULLIO, Randy PAUSCH, Dennis PROFFITT et Neal KASSELL : Usability analysis of 3D rotation techniques. *In UIST '97 : Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 1–10, New York, NY, USA, 1997. ACM. ISBN 0-89791-881-9.

Jane HWANG, Jaehoon JUNG et Gerard Jounghyun KIM : Hand-held virtual reality : a feasibility study. *In VRST '06 : Proceedings of the ACM symposium on Virtual reality software and technology*, pages 356–363, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-321-2.

Takeo IGARASHI, Rieko KADOBAYASHI, Kenji MASE et Hidehiko TANAKA : Path drawing for 3D walkthrough. *In UIST '98 : Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 173–174, New York, NY, USA, 1998. ACM Press. ISBN 1-58113-034-1.

Inés JACOB et Javier OLIVER : Evaluation of techniques for specifying 3D rotations with a 2d input device. *In HCI '95 : Proceedings of the HCI'95 conference on People and computers X*, pages 63–76, New York, NY, USA, 1995. Cambridge University Press. ISBN 0-521-56729-7.

Robert J. K. JACOB et Linda E. SIBERT : The perceptual structure of multidimensional input device selection. *In CHI '92 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 211–218, New York, NY, USA, 1992. ACM. ISBN 0-89791-513-5.

Amy K. KARLSON et Benjamin B. BEDERSON : Thumbspace : Generalized one-handed input for touchscreen-based mobile devices. *In* Maria Cecília Calani BARANAUSKAS, Philippe A. PALANQUE, Julio ABASCAL et Simone Diniz Junqueira BARBOSA, éditeurs : *INTERACT (1)*, volume 4662 de *Lecture Notes in Computer Science*, pages 324–338. Springer, 2007. ISBN 978-3-540-74794-9.

Amy K. KARLSON, Benjamin B. BEDERSON et Jose L. CONTRERAS-VIDAL : Understanding one handed use of mobile devices. *Handbook of Research on User Interface Design and Evaluation for Mobile Technology, Idea Group Reference*, 2007.

Amy K. KARLSON, Benjamin B. BEDERSON et John SANGIOVANNI : Applens and launchtile : two designs for one-handed thumb use on small devices. *In CHI '05 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 201–210, New York, NY, USA, 2005. ACM Press. ISBN 1-58113-998-5.

Kiyokuni KAWACHIYA et Hiroshi ISHIKAWA : Navipoint : an input device for mobile information browsing. *In CHI '98 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1–8, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-30987-4.

Azam KHAN, Ben KOMALO, Jos STAM, George FITZMAURICE et Gordon KURTENBACH : Hovercam : interactive 3D navigation for proximal object inspection. *In I3D '05 : Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 73–80, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-013-2.

Azam KHAN, Igor MORDATCH, George FITZMAURICE, Justin MATEJKA et Gordon KURTENBACH : Viewcube : a 3D orientation indicator and controller. *In SI3D '08 : Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pages 17–25, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-983-8.

Duck-soo KIM : Joystick apparatus for portable terminal, March 2009.

Ji-Sun KIM, Denis GRACANIN, Kresimir MATKOVIC et Francis QUEK : Finger walking in place (fwip) : a traveling technique in virtual environments. *In Proceedings of SmartGraphics 2008, Springer LNCS 5166/2008*, LNCS, pages ISBN 978-3-540-85410-4.

Kenrick KIN, Maneesh AGRAWALA et Tony DEROSE : Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. *In Proceedings of Graphics Interface 2009*, 2009.

Jesper KJELDSKOV et Nikolaj KOLBE : Interaction design for handheld computers. *In Proc. of the 5 thAPCHI'02*, 2002.

Sebastian KNÖDEL, Martin HACHET et Pascal GUITTON : Sketch-based route planning with mobile devices in immersive virtual environments. *In 2èmes journées de l'AFRV*. INRIA, 2007.

Sebastian KNÖDEL, Martin HACHET et Pascal GUITTON : Navidget for immersive virtual environments. *In VRST '08 : Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, pages 47–50, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-951-7.

Celine LATULIPE, Craig S. KAPLAN et Charles L. A. CLARKE : Bimanual and unimanual image alignment : an evaluation of mouse-based techniques. *In UIST*, pages 123–131, 2005.

Celine LATULIPE, Stephen MANN, Craig S. KAPLAN et Charles L. A. CLARKE : symspline : symmetric two-handed spline manipulation. *In CHI*, pages 349–358, 2006.

Yang LI, Ken HINCKLEY, Zhiwei GUAN et James A. LANDAY : Experimental analysis of mode switching techniques in pen-based user interfaces. *In CHI '05 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 461–470, New York, NY, USA, 2005. ACM Press. ISBN 1-58113-998-5.

Joern LOVISCACH : Two-finger input with a standard touch screen. *In UIST '07 : Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 169–172, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-679-2.

I. Scott MACKENZIE et R. William SOUKOREFF : Text Entry for Mobile Computing : Models and Methods, Theory and Practice. *Human-Computer Interaction*, 17(2):147–198, 2002.

Jock D. MACKINLAY, Stuart K. CARD et George G. ROBERTSON : Rapid controlled movement through a virtual 3D workspace. *In Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 171–176. ACM Press New York, NY, USA, 1990.

Daniel P. MAPES et J. Michael MOSHELL : A two handed interface for object manipulation in virtual environments. *Presence*, 4(4):403–416, 1995.

Gary MARSDEN et Nicholas TIP : Navigation control for mobile virtual environments. *In Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 279–282. ACM Press New York, NY, USA, 2005.

Justin MATEJKA, Tovi GROSSMAN, Jessica LO et George FITZMAURICE : The design and evaluation of multi-finger mouse emulation techniques. *In CHI '09 : Proceedings of the 27th international conference on Human factors in computing systems*, pages 1073–1082, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-246-7.

Nobuyuki MATSUSHITA, Yuji AYATSUKA et Jun REKIMOTO : Dual touch : a two-handed interface for pen-based pdas. *In UIST '00 : Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 211–212, New York, NY, USA, 2000. ACM. ISBN 1-58113-212-3.

James MCCRAE, Igor MORDATCH, Michael GLUECK et Azam KHAN : Multiscale 3D navigation. *In I3D '09 : Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 7–14, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-429-4.

James Palemr MCCRAE : Sketch-based path design. Mémoire de D.E.A., University of Toronto, 2008.

MESURES : Les écrans tactiles s'adaptent au besoin... grâce à la technologie. *Mesures*, 740:55–57, 2001.

Mark R. MINE, Jr. FREDERICK P. BROOKS et Carlo H. SEQUIN : Moving objects in space : exploiting proprioception in virtual-environment interaction. *In SIGGRAPH '97 : Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 19–26, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. ISBN 0-89791-896-7.

Tomer MOSCOVICH et John F. HUGHES : Indirect mappings of multi-touch input using one and two hands. *In CHI '08 : Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1275–1284, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-011-1.

Gregory M. NIELSON et Jr. DAN R. OLSEN : Direct manipulation techniques for 3D objects using 2d locator devices. *In SI3D '86 : Proceedings of the 1986 workshop on Interactive 3D graphics*, pages 175–182, New York, NY, USA, 1987. ACM Press. ISBN 0-89791-228-4.

Ji-Young OH et Wolfgang STUERZLINGER : Moving objects with 2d input devices in cad systems and desktop virtual environments. *In GI '05 : Proceedings of Graphics Interface 2005*, pages 195–202, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society. ISBN 1-56881-265-5.

Alex OLWAL, Steven FEINER et Susanna HEYMAN : Rubbing and tapping for precise and rapid selection on touch-screen displays. *In CHI '08 : Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 295–304, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-011-1.

Pekka PARHI, Amy K. KARLSON et Benjamin B. BEDERSON : Target size study for one-handed thumb use on small touchscreen devices. *In MobileHCI '06 : Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pages 203–210, New York, NY, USA, 2006. ACM. ISBN 1-59593-390-5.

Timo PARTALA : Controlling a single 3D object : Viewpoint metaphors, speed and subjective satisfaction. *In Sasse M.A. and Johnson C. (Ed.) Proceedings of INTERACT'99*, page 486–493. IOS Press, IOS Press, sep 1999.

Dale PATTERSON : 3d space : using depth and movement for selection tasks. *In Web3D '07 : Proceedings of the twelfth international conference on 3D web technology*, pages 147–155, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-652-3.

Randy PAUSCH, Tommy BURNETTE, Dan BROCKWAY et Michael E. WEIBLEN : Navigation and locomotion in virtual worlds via flight into hand-held miniatures. *In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 399–400. ACM Press New York, NY, USA, 1995.

Keith B. PERRY et Juan Pablo HOURCADE : Evaluating one handed thumb tapping on mobile touchscreen devices. *In GI '08 : Proceedings of graphics interface 2008*, pages 57–64, Toronto, Ont., Canada, Canada, 2008. Canadian Information Processing Society. ISBN 978-1-56881-423-0.

Jeffrey S. PIERCE et Randy PAUSCH : Navigation with place representations and visible landmarks. *In Virtual Reality, 2004. Proceedings. IEEE*, pages 173–288, 2004.

Tapani RANTAKOKKO et Johan PLOMP : An adaptive map-based interface for situated services. *In Proceedings of Smart Object Conference.*, 2003.

Jun REKIMOTO : Tilting operations for small screen interfaces. *In UIST '96 : Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 167–168, New York, NY, USA, 1996. ACM Press. ISBN 0-89791-798-7.

Daniel C. ROBBINS, Edward CUTRELL, Raman SARIN et Eric HORVITZ : Zonezoom : map navigation for smartphones with recursive view segmentation. *In AVI '04 : Proceedings of the working conference on Advanced visual interfaces*, pages 231–234, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-867-9.

George G. ROBERTSON, Jock D. MACKINLAY et Stuart K. CARD : Cone trees : animated 3D visualizations of hierarchical information. *In CHI '91 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 189–194, New York, NY, USA, 1991. ACM. ISBN 0-89791-383-3.

Anne ROUDAUT, Stéphane HUOT et Eric LECOLINET : Taptap and magstick : improving one-handed target acquisition on small touch-screens. *In AVI '08 : Proceedings of the working conference on Advanced visual interfaces*, pages 146–153, New York, NY, USA, 2008. ACM. ISBN 0-978-60558-141-5.

Anne ROUDAUT, Eric LECOLINET et Yves GUIARD : Microrolls : expanding touch-screen input vocabulary by distinguishing rolls vs. slides of the thumb. *In CHI '09 : Proceedings of the 27th international conference on Human factors in computing systems*, pages 927–936, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-246-7.

Markus SCHLATTMANN et Reinhard KLEIN : Efficient bimanual symmetric 3D manipulation for markerless hand- tracking. *In Virtual Reality International Conference (VRIC) (to appear)*, avril 2009.

Dieter SCHMALSTIEG, L. Miguel ENCARNAÇÃAO et Zsolt SZALAVÁRI : Using transparent props for interaction with the virtual table. *In Proceedings of the 1999 symposium on Interactive 3D graphics SI3D '99*, pages 147–153. ACM Press, 1999.

Ryan SCHMIDT, Karan SINGH et Ravin BALAKRISHNAN : Sketching and composing widgets for 3D manipulation. *In Eurographics 2008 (To Appear)*, 2008.

Erh-li Early SHEN, Sung-sheng Daniel TSAI, Hao-hua CHU, Yung-jen Jane HSU et Chi-wen Euro CHEN : Double-side multi-touch input for mobile devices. *In CHI EA '09 : Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 4339–4344, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-247-4.

Charles SHERRINGTON : *The integrative action of the nervous system / by Charles S. Sherrington*. New Haven,, 1911.

Ken SHOEMAKE : Arcball : a user interface for specifying three-dimensional orientation using a mouse. *In Proceedings of the conference on Graphics interface '92*, pages 151–156, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc. ISBN 0-9695338-1-0.

Miika SILFVERBERG, I. Scott MACKENZIE et Tatu KAUPPINEN : An isometric joystick as a pointing device for handheld information terminals. *In GRIN'01 : No description on Graphics interface 2001*, pages 119–126, Toronto, Ont., Canada, Canada, 2001. Canadian Information Processing Society. ISBN 0-9688808-0-0.

Miika SILFVERBERG, I. Scott MACKENZIE et Panu KORHONEN : Predicting text entry speed on mobile phones. *In CHI '00 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 9–16, New York, NY, USA, 2000. ACM Press. ISBN 1-58113-216-6.

Richard STOAKLEY, Matthew J. CONWAY et Randy PAUSCH : Virtual reality on a wim : interactive worlds in miniature. *In Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 265–272. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 1995.

Desney S. TAN, George G. ROBERTSON et Mary CZERWINSKI : Exploring 3D navigation : combining speed-coupled flying with orbiting. *In Proceedings of the SIGCHI conference on Human factors in computingsystems CHI 01*, pages 418–425. ACM Press, 2001.

Perry W. THORNDYKE et Barbara HAYES-ROTH : Differences in spatial knowledge acquired from maps and navigation. *In Cognitive Psychology*, 1982.

Matthew THORNE, David BURKE et Michiel van de PANNE : Motion doodles : an interface for sketching character motion. *ACM Trans. Graph.*, 23(3):424–431, 2004. ISSN 0730-0301.

Daniel VOGEL et Patrick BAUDISCH : Shift : a technique for operating pen-based interfaces using touch. *In CHI '07 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 657–666, New York, NY, USA, 2007. ACM Press. ISBN 978-1-59593-593-9.

Daniel VOGEL, Matthew CUDMORE, Géry CASIEZ, Ravin BALAKRISHNAN et Liam KELIHER : Hand occlusion with tablet-sized direct pen input. *In CHI '09 : Proceedings of the 27th international conference on Human factors in computing systems*, pages 557–566, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-246-7.

Daniel WAGNER, Thomas PINTARIC et Dieter SCHMALSTIEG : The invisible train : a collaborative handheld augmented reality demonstrator. *In SIGGRAPH '04 : ACM SIGGRAPH 2004 Emerging technologies*, page 12, New York, NY, USA, 2004. ACM Press. ISBN 1-59593-896-2.

Daniel. WAGNER et Dieter SCHMALSTIEG : Artoolkit on the pocketpc platform. *In Augmented Reality Toolkit Workshop, 2003. IEEE International*, pages 14–15, 2003.

Daniel WAGNER et Dieter SCHMALSTIEG : Handheld augmented reality displays. *In Proceedings of IEEE Virtual Reality 2006*. IEEE, March 2006.

Daniel WAGNER, Dieter SCHMALSTIEG et Bischof HORST : Multiple target detection and tracking with guaranteed framerates on mobile phones. *In Proceedings of Int. Symposium on Mixed and Augmented Reality 2009 (ISMAR 09)*, 2009.

Shuo WANG, Marcin POTURALSKI et David VRONAY : Designing a generalized 3D carousel view. *In CHI '05 : CHI '05 extended abstracts on Human factors in computing systems*, pages 2017–2020, New York, NY, USA, 2005. ACM. ISBN 1-59593-002-7.

Zhijin WANG et Michiel van de PANNE : Walk to here : a voice driven animation system. *In SCA '06 : Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 243–251, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association. ISBN 3-905673-34-7.

Colin WARE et Daniel FLEET : Context sensitive flying interface. *In SI3D '97 : Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 127–ff., New York, NY, USA, 1997. ACM. ISBN 0-89791-884-3.

Colin WARE et Steven OSBORNE : Exploration and virtual camera control in virtual three dimensional environments. *In Proceedings of the 1990 symposium on Interactive 3D graphics*, pages 175–183. ACM Press New York, NY, USA, 1990.

Daniel WIGDOR, Clifton FORLINES, Patrick BAUDISCH, John BARNWELL et Chia SHEN : Lucid touch : a see-through mobile device. *In UIST '07 : Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 269–278, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-679-2.

Jacob O. WOBBROCK, Brad A. MYERS et Scott E. HUDSON : Exploring edge-based input techniques for handheld text entry. *In ICDCSW '03 : Proceedings of the 23rd International Conference on Distributed Computing Systems*, page 280, Washington, DC, USA, 2003a. IEEE Computer Society. ISBN 0-7695-1921-0.

Jacob O. WOBBROCK, Brad A. MYERS et John A. KEMBEL : Edgewrite : a stylus-based text entry method designed for high accuracy and stability of motion. *In UIST '03 : Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 61–70, New York, NY, USA, 2003b. ACM Press. ISBN 1-58113-636-6.

Crampton Jeff Philip WONG JOHN PATRICK : Pointing device for handheld devices and method for implementing same, 2007.

Eric WOODS, Paul MASON et Mark BILLINGHURST : Magicmouse : an inexpensive 6-degree-of-freedom mouse. *In GRAPHITE '03 : Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 285–286, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-578-5.

Ying. WU, Ying SHAN, Zhengyou ZHANG et Steven SHAFER : Visual panel : From an ordinary paper to a wireless and mobile input device. Rapport technique, Technical Report, MSR-TR-2000 Microsoft Research Corporation, http ://www. research. microsoft. com, October 2000, 2000.

Xing-Dong YANG, Pourang IRANI, Pierre BOULANGER et Walter BISCHOF : One-handed behind-the-display cursor input on mobile devices. *In CHI EA '09 : Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 4501–4506, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-247-4.

Koji YATANI, Kurt PARTRIDGE, Marshall BERN et Mark W. NEWMAN : Escape : a target selection technique using visually-cued gestures. *In CHI '08 : Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 285–294, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-011-1.

Robert C. ZELEZNIK et Andrew S. FORSBERG : Unicam - 2d gestural camera controls for 3D environments. *In Proceedings of Symposium on Interactive 3D Graphics*, pages 169–173, 1999.

Robert C.. ZELEZNIK, Andrew S. FORSBERG et Paul S. STRAUSS : Two pointer input for 3D interaction. *In Proceedings of the 1997 symposium on Interactive 3D graphics*, 1997.

Robert C. ZELEZNIK, Kenneth P HERNDON et John F. HUGHES : Sketch : An interface for sketching 3D scenes. *In ACM Transactions on Graphics, Proceedings of SIGGRAPH*, 1996.

Robert C. ZELEZNIK, Joseph J. LAVIOLA JR, Daniel Acevedo FELIZ et Daniel F. KEEFE : Pop through button devices for ve navigation and interaction. *In VR '02 : Proceedings of the IEEE Virtual Reality Conference 2002*, page 127, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1492-8.

Shumin ZHAI : *Human Performance in Six Degree of Freedom Input Control*. Thèse de doctorat, University of Toronto, 1995.

Shumin ZHAI, Eser KANDOGAN, Barton A. SMITH et Ted SELKER : In search of the 'magic carpet' : Design and experimentation of a bimanual 3D navigation interface. *Journal of Visual Languages and Computing*, 10(1):3–17, 1999.

Zhengyou ZHANG, Ying WU, Ying SHAN et Steven SHAFER : Visual panel : virtual mouse, keyboard and 3D controller with an ordinary piece of paper. *In PUI '01 : Proceedings of the 2001 workshop on Perceptive user interfaces*, pages 1–8, New York, NY, USA, 2001. ACM Press.

Webographie

BLENDER : Site officiel du logiciel blender 3d. http://www.blender.org.

INDUSTRIALDESIGN : Site du studio industrial design. http://www.industrialdesign.ru/index_eng.html, 2009.

Julian OLIVER : Site officiel du jeu levelhead. http://julianoliver.com/levelhead, 2008.

STANTUM : Site de la société stantum. http://www.stantum.com.

TOUCHCO : Site de la société touchco. http://www.touchco.com.

Hans-Martin WILL. : Site de la bibliothèque opengl es vincent. http://ogl-es.sourceforge.net/.

Publications

Fabrice DECLE et Martin HACHET : A study of direct versus planned 3D camera manipulation on touch-based mobile phones. *In Proceedings of Mobile HCI 2009*, 2009.

Fabrice DECLE, Martin HACHET et Pascal GUITTON : Z-goto : pour une navigation efficace dans des environnements 3D sur terminaux mobiles. *In 1eres Journées de l'AFRV*. INRIA, 2006. URL `http://iparla.labri.fr/publications/2006/DHG06`.

Fabrice DECLE, Martin HACHET et Pascal GUITTON : Scruticam : Camera manipulation technique for 3D objects inspection. *In Proceedings of IEEE 3DUI 2009 - Symposium on 3D User Interfaces*, 2009. URL `http://iparla.labri.fr/publications/2009/DHG09`.

M. HACHET, F. DECLE et P. GUITTON : Z-goto for efficient navigation in 3D environments from discrete inputs. *In Proceedings of the ACM symposium on Virtual reality software and technology*, 2006.

Martin HACHET, Fabrice DECLE, Sebastian KNÖDEL et Pascal GUITTON : Navidget for 3D interaction : Camera positioning and further uses. *Int. J. Hum.-Comput. Stud.*, 67(3):225–236, 2009.

Martin HACHET, Fabrice DECLE, Sebastian KNOEDEL et Pascal GUITTON : Navidget for easy 3D camera positioning from 2d inputs. *In Proceedings of Symposium on 3D User Interfaces (3DUI) 2008*, 2008. To appear.

Imane ZENDJEBIL, Fakhr-Eddine ABABSA, Jean-Yves DIDIER, Emilie LALAGÜE, Fabrice DECLE, Romuald DELMONT, Luc FRAUCIEL et Jacques VAIRON : Réalité augmentée en extérieur : Etat de l'art. *Technique et Science Informatiques (TSI), Innovations en Réalité Virtuelle et Réalité Augmentée*, 2009.