

ARCHITECTURE LOGICIELLE POUR UN SYSTÈME D'EXPLOITATION DES ENVIRONNEMENTS DE RÉALITÉ AUGMENTÉE COMPLEXES

Vincent LE LIGEOUR, Samir OTMANE et Malik MALLEM

Laboratoire Systèmes Complexes

EVRY

{leligeour,otmane,malle}@lsc.univ-evry.fr

Résumé - Le développement d'un système de Réalité Augmentée (RA) nécessite souvent la réécriture d'un grand nombre de modules qui ne sont pas nécessairement réutilisables au sein d'un laboratoire. Pour palier à ce problème il faut donc opter pour un environnement de développement. Il est intéressant de noter les similitudes entre un tel environnement et un système d'exploitation.

Mots clé - Réalité Augmentée, Système d'exploitation, Architecture distribuée, multi agents, Fusion de données hétérogènes.

1 Introduction

Un système de réalité augmentée pour l'assistance au travail/télétravail peut se représenter sous forme de relations entre trois pôles bien distincts [Fig. 1] :

- La perception
- La communication
- Les interactions

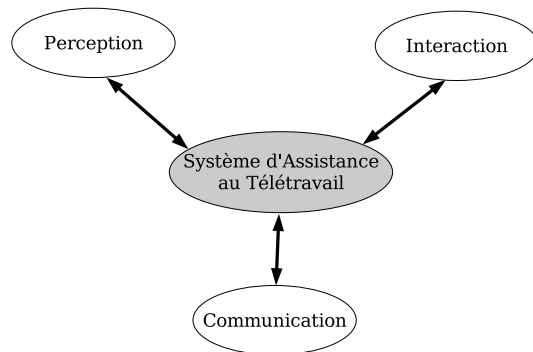


FIG. 1 – Graphe Conceptuel du système d'assistance au travail/télétravail

Actuellement dans les laboratoires de RA/Réalité Virtuelle (RV), il existe un très grand nombre de développements parallèles. Ces développements ont bien souvent des points communs (Affichage de vidéo à l'écran, calibration de caméra, applications de filtres standard, récupérations de positions à partir de capteurs, etc.). Mais malheureusement très peu d'entre eux sont réutilisables en l'état. C'est ainsi que depuis quelques temps est apparu la notion de Framework (environnement de développement) dans les systèmes de réalité augmentée. A cet effet un groupe de travail a été créé : STARS 2003 *The International Workshop on Software Technology for Augmented Reality Systems*[1],

qui vise à regrouper l'ensemble des développements et des chercheurs sur ce sujet.

L'avantage de tels environnements de développement est d'homogénéiser les développements d'une équipe à travers la contrainte de l'utilisation d'un environnement de développement unifié. Ceci permet alors un développement très rapide d'applications et l'expérimentation quasiment instantanée de procédures dont on a déjà tous les composants qu'il ne reste alors plus qu'à assembler. A partir de là, il est très rapide de concevoir des composants clients/serveurs pour distribuer un tel système. De plus un tel environnement de développement peut aussi être vu comme une base de connaissance.

2 Les architectures de référence

Il existe en Réalité Augmentée deux environnements de travail qui font référence actuellement : DWARF[3] et ARVIKA[2]. L'étude du système ARVIKA est relativement ardue car cet environnement n'est pas public, il est donc impossible d'en faire une étude interne. On peut cependant avoir la vision plus globale du système d'information utilisé. Il existe ainsi un comparatif des deux systèmes, mais d'un point de vue relativement élevé dans la conception (Comparatif [6]).

Le système DWARF quand à lui est diffusé de façon libre. L'environnement de développement DWARF (Distributed Wearable Augmented Reality Framework) est basé sur le concept de services distribués collaboratifs [Fig. 2]. Les services sont indépendants et distribuent leurs nécessités appelés Besoins et leurs offres, appelé Habilités, à l'aide du gestionnaire de services. Sur chaque nœud du réseau DWARF, il y a un gestionnaire de service ; il n'y a pas de composant central. Le gestionnaire de service contrôle ses services locaux et maintient une description de ces derniers. Chaque gestionnaire est en liaison avec les autres sur le réseau pour établir les connections entre chaque services. Chaque service a une description XML de ses Habilités, Besoins et de ses connections (protocoles de communi-

tion). Une part importante est aussi donnée à la qualité de service donné par un service par rapport à la qualité de service attendu par un autre service. Une telle modularité permet de faire communiquer différents systèmes à travers des modules de communications [5].

DWARF est de plus conçu pour être portable (au sens logiciel et matériel du terme). C'est à dire qu'il existe un grand nombre de langages supportés (C, C++, Java, Python) sur un grand nombre de plate forme (Linux principalement, MacOS X, Windows). De plus grâce au Wifi et à des ordinateurs portables il y a possibilité de pouvoir se déplacer librement dans un environnement.

DWARF supporte de base un grand nombre de services, comme le tracking par système IR (AR-Tracking), par marqueurs (ART), des visualisateurs, des reconnaissances de voix, etc. Ceci est en fait à première vu un outil relativement intéressant à utiliser.

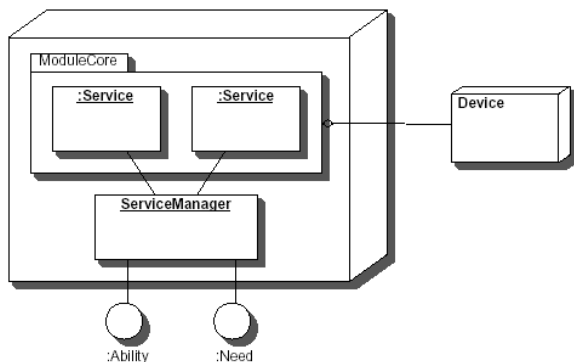


FIG. 2 – Service du système DWARF [4]

DWARF possède cependant un inconvénient majeur, il est extrêmement lourd à installer et quasiment impossible à mettre en oeuvre en dehors de son laboratoire d'origine (Technische Universität München).

De plus que ce soit DWARF ou ARVIKA, ces deux systèmes sont très orientés système d'informations, autrement dit ces framework génèrent plus des systèmes différents à partir de composants matériels identiques que de réelles applications à partir de capteurs hétérogènes. Ce qui fait qu'ils perdent en flexibilité au niveau de l'implémentation. Ils sont plus axés sur des démonstrations que sur de réels développements.

3 Architecture proposée

Comme on peut donc le remarquer, les outils existants ne sont pas parfaits. Il manque en particulier un méta langage pouvant servir à décrire un système de RA. Un tel langage servira alors à créer le système en lui même.

De plus ce système [Fig. 3] devra permettre générer l'application. Pour cela il serait nécessaire de gérer un tel système comme un système d'exploitation. En effet un système d'exploitation est fait pour gérer de façon plus ou moins temps réel (suivant le besoin) la fusion multi capteurs. De plus la possibilité ensuite d'afficher des résultats

sur des périphériques de sorties d'acquérir sur diverses entrées de générer des applications à partir d'un langage comme un compilateur. Pour cela il faut rendre le système au maximum modulaire pour pouvoir arranger ce que l'on peut appeler des briques d'une façon la plus libre possible. Il faut également veiller à travailler à travers des interfaces et éviter d'accéder directement au matériel et ainsi éviter les problèmes de portabilité.

Principe d'un système d'exploitation pour la réalité augmentée

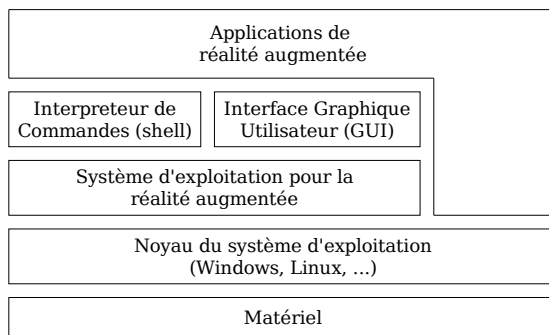


FIG. 3 – Architecture globale d'un système d'exploitation en RA

Un autre problème apparent de DWARF est qu'il ne permet pas l'approche du système par une personne étrangère à l'informatique. Pour ceci il serait possible d'intégrer une interface utilisateur graphique (GUI) permettant de rapidement décrire un système. La représentation serait semblable à un graphe. Les sommets représentant les traitements ou les périphériques les arcs les échanges de donnée possible.

Concernant les briques disponibles dans un tel systèmes on peut également imaginer des briques réseau permettant de distribuer les traitements ou les capteurs et augmenté la mobilité d'un tel système. Mais contrairement au système de référence DWARF la brique réseau n'est pas une partie intégrante d'un service mais uniquement une portion que l'on peut greffer ou non à un système qui deviendra un service si ce dernier vient à être une partie d'un système distribué.

Bien entendu une telle description pose un certain nombre de problèmes. Il faut notamment trouver un système le plus générique possible, et dans le cas parfait totalement générique. Ceci explique qu'un tel système doit posséder une phase d'étude relativement importante avant d'être développé. Il ne faut pas que l'apparition d'un nouveau système rende son intégration impossible. De plus il faut qu'un tel système puisse être utilisé sur un grand nombre de machine pour des tests. Il est donc important également d'accompagner chacun des capteurs de capteurs virtuels produisant des données à partir de données aléatoires, calculées ou alors acquises lors de mesures.

Un autre problème est l'aspect temporel d'un tel système. La solution se trouve dans les systèmes d'exploitation importants : l'ordonnanceur. On peut voir l'ordonnanceur comme

un superviseur donnant la parole à chacune des composantes de l'application. Mais contrairement au gestionnaire de services de DWARF il sera incorporé directement à l'application. Car dans tous les cas lors de l'élaboration d'une application permettant d'obtenir des résultats il existe un coeur dans l'application, il est donc inutile de multiplier les ordonnanceurs (ou superviseurs)..

De plus pour permettre d'avoir des résultats fiables il sera nécessaire d'incorporer une datation dans l'ensemble des dialogues entre briques (ou agents). Ceci permettra d'obtenir des résultats et de gérer les situations critiques ou des données sont susceptibles de ne pas être pertinentes.

4 Architecture du pipeline graphique proposé

Un exemple intéressant à traiter dans le cadre d'un tel système serait les pipelines mis en place pour afficher un flux vidéo stéréographique [Fig 4].

L'acquisition peut se faire :

- à distance via une carte d'acquisition plus cameras bâton : cette configuration est celle qui permet d'obtenir la meilleure qualité, mais elle nécessite une installation coûteuse et lourde en matériel.
- à travers des webcams (par exemple USB) : l'image est de moins bonne qualité, mais le système est adaptable sur quasiment n'importe quel ordinateur ou PDA
- ou encore depuis une suite de fichiers : qui permet d'avoir des simulation off-line.

Concernant le transfert de fichier il y a également un grand nombre d'options disponibles :

- mémoire partagée : si les deux modules sont lancés sur la même machine
- compression Jpeg : pratique car permet une compression importante et une décompression standard (le client pouvant par exemple être un navigateur internet)
- compression Zip : permet de compresser les données de façon importante tout en ne nuisant pas à la qualité de l'image.

Vient ensuite le client qui lui doit écrire ces deux images du pipeline graphique réel dans un contexte 3D OpenGL. Pour cela on dispose d'un grand nombre d'outils (WxWindows, GLUT, Gtk, etc.). Par rapport au schéma on peut expliciter la brique « Vidéo Stéréo » qui comprend également d'éventuels rectifications sur l'image.

Enfin la partie concernant le monde virtuel permettra grâce aux données fournies par un système de tracking (Infra rouge, marqueurs, etc.) de faire évoluer de façon stéréographique le modèle virtuel sur les images vidéo.

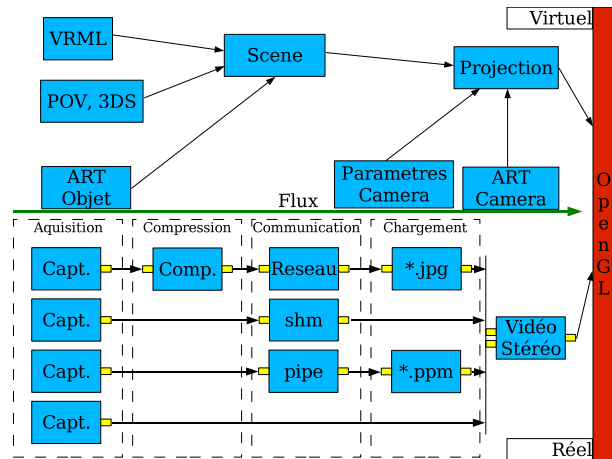


FIG. 4 – Organisation du pipeline graphique

5 Conclusion

Bien que DWARF soit un système très complet il pâtit d'un certain nombre de choix qui nuisent à l'architecture globale d'un environnement de développement permettant un prototypage rapide d'applications en RA. On peut ainsi noter la lourdeur des installations, la taille des modules sans doute prévue trop grosse.

D'un autre côté l'architecture telle que je la propose permettrait réellement de développer des tests et des applications relativement rapidement et ce avec des architectures internes pouvant grandement varier. Ainsi une même application pourra intérieurement conçu de façon totalement différente.

Il reste néanmoins encore un certain nombre de problèmes à régler comme ceux de généricité qu'il ne faut surtout pas manquer, et d'implémentation (communication en mémoire, ordonnancement des tâches, etc.). Finalement la durée de vie d'un tel système dépendra de la pertinence des choix conceptuels.

Références

- [1] <http://stars2003.cs.tum.edu/>. Internet.
- [2] <http://www.arvika.de>. Internet.
- [3] <http://www.augmentedreality.de>. Internet.
- [4] Martin Bauer, Bernd Bruegge, Gudrun Klinker, Asa MacWilliams, Thomas Reicher, Stefan Riss, Christian Sandor, and Martin Wagner. Design of a component-based augmented reality framework. In *Proceedings of ISAR 2001*, 2001.
- [5] Martin Bauer, Otmar Hilliges, Asa MacWilliams, Christian Sandor, Martin Wagner, Joe Newman, Gerhard Reitmayr, Tamer Fahmy, Gudrun Klinker, Thomas Pintaric, and Dieter Schmalstieg. Integrating studentstube and dwarf. In *International Workshop on Software Technology for Augmented Reality Systems (STARS 2003)*, 2003.

- [6] Thomas Reicher, Asa MacWilliams, Bernd Brügge, and Gudrun Klinker. Results of a study on software architectures for augmented reality systems. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, Tokio, Japan, October 2003.