INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque

THÈSE

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : "Informatique : Systèmes et Communications"

préparée au laboratoire INFORMATIQUE ET DISTRIBUTION
dans le cadre de ***l'École Doctorale "Mathématiques, Sciences et Technologies de l'Information, Informatique"***

présentée et soutenue publiquement

par

Jérémie ALLARD

le 25 novembre 2005

FlowVR : calculs interactifs et visualisation sur grappe

Directeur de thèse : Mme Brigitte Plateau

JURY

MME MARIE-PAULE CANI	, Présidente
M. ALAN CHALMERS	, Rapporteur
M. THIERRY PRIOL	, Rapporteur
MME BRIGITTE PLATEAU	, Directeur de thèse
M. BRUNO RAFFIN	, Co-encadrant

Remerciements

Je tiens à remercier Bruno Raffin qui m'a encadré pour ces travaux et dont les efforts ont permis de monter la plateforme expérimentale GrImage, sans laquelle une grande partie de ce travail n'aurait pas été possible.

J'ai travaillé avec de nombreuses personnes au cours de cette thèse, et je remercie plus particulièrement Clément Ménier qui a toujours été présent qu'en cela était nécessaire. J'ai eu de nombreux échanges trés bénéfiques avec François Faure autour des problèmes liés à l'animation et aux interactions et je lui en suis très reconnaissant. Je remercie aussi Nicolas Turro et Florian Geffray pour leurs contributions sur les aspects techniques des expérimentations.

Merci aux membres de mon jury, Marie-Paule Cani, Brigitte Plateau, Alan Chalmers et Thierry Priol pour avoir accepté d'en faire partie. Leurs remarques avisées m'ont permis de prendre conscience de nouveaux points de vue sur mon travail.

Je tiens également à remercier l'équipe du laboratoire ID pour la bonne ambiance de travail et les discussions fort intéressantes.

Enfin je ne serais pas la sans ma famille, qui m'a soutenu moralement pendant ces trois ans. Je remercie mes quatre grand-parents, qui ont eut la patience de relire tout ce document plusieurs fois. J'ai une pensée toute particulière pour mes deux neveux, Nathaël et Fabian, qui sont nés pendant cette thèse, et à qui je la dédie.

Table des matières

1 Introduction **1**
Contexte 1
Problématique 2
Contributions 3
Organisation 4

I Etat de l'art **5**

2 Logiciels pour la réalité virtuelle **7**
2.1 Lecture des entrées 9
2.2 Traitement des entrées 9
2.3 Calculs et simulations 10
2.3.1 Systèmes multi-agents 11
2.3.2 Simulations interactives 11
2.3.3 Simulations externes 12
2.4 Traitement des sorties 12
2.5 Rendu 13
2.6 Bilan 15

3 Communications et couplage de codes **17**
3.1 API de communication 18
3.1.1 Passage de messages 18
3.1.2 Appel de procédure à distance 19
3.1.3 Athapascan : graphe de flux de données par appels de procédure . 20
3.1.4 PadicoTM : Communications multi-protocoles 21
3.2 Couplage de codes 21
3.2.1 Composants logiciels 22
3.2.2 Couplage de codes parallèles 23
3.3 Bilan 25

4 Parallélisation pour la réalité virtuelle **27**
4.1 Rendu multi-projecteurs . . . 28
4.1.1 Sort-last . . . 28
4.1.2 Sort-first . . . 29
4.1.3 Graphes de scènes distribués . . . 29
4.1.4 Duplication . . . 30
4.2 Cadres applicatifs de réalité virtuelle . . . 30
4.2.1 CAVELib . . . 30
4.2.2 VR Juggler . . . 31
4.2.3 Syzygy . . . 32
4.2.4 OpenMASK . . . 32
4.3 Bilan . . . 33

II FlowVR **35**

5 Travaux préliminaires **37**
5.1 Net Juggler : duplication transparente d'applications VR Juggler . . . 37
5.1.1 Exécution sur grappe . . . 38
5.1.2 Résultats . . . 38
5.2 Simulations interactives sous Net Juggler . . . 39
5.2.1 Simulation de fluide 2D interactive . . . 39
5.2.2 Simulation de tissus préexistante . . . 41
5.3 Bilan . . . 42

6 Un modèle d'application distribuée interactive **45**
6.1 Introduction . . . 45
6.2 Le modèle choisi . . . 45
6.2.1 Granularité du découpage . . . 46
6.2.2 Connaissance des modules sur le reste de l'application . . . 47
6.2.3 Configuration de l'application . . . 47
6.2.4 Synchronisation et cohérence . . . 48
6.2.5 Placement et allocation des ressources . . . 49
6.3 Exemple d'application . . . 49
6.4 Des modules réutilisables . . . 50
6.4.1 Définition d'un module . . . 50
6.4.2 Données échangées par les modules . . . 50
6.4.3 Opérations utilisées par les modules . . . 51
6.4.4 Exemple . . . 54
6.5 Graphe de flux de données . . . 55
6.5.1 Connections simples . . . 55

6.5.2 Filtrage des données . . . 55
6.5.3 Communications collectives . . . 56
6.6 Asynchronisme contrôlé : ordonnancement par les données . . . 58
6.6.1 Synchronisation entre modules . . . 58
6.6.2 Exemple d'utilisation des filtres et synchroniseurs . . . 61
6.6.3 Exemples d'algorithmes de filtres et synchroniseurs . . . 62

7 Implantation **65**
7.1 Environnement d'exécution . . . 65
7.1.1 Communications inter-processus . . . 65
7.1.2 Démon . . . 66
7.1.3 Configuration de l'application : langage de commandes . . . 69
7.2 Mécanisme de contrôle d'applications . . . 69
7.2.1 Lancement des modules . . . 70
7.2.2 Construction du graphe de flux de données . . . 70
7.3 Passage à l'échelle : dépliage de graphes . . . 71
7.4 Performances . . . 73
7.4.1 Mesure et analyse de trace . . . 75
7.4.2 Signaux inter-processus . . . 76
7.4.3 Couplage parallèle Simulation / Visualisation . . . 76

8 Expérimentations **79**
8.1 La plateforme GrImage . . . 79
8.2 Applications graphiques simples . . . 79
8.3 Reconstruction 3D temps réel . . . 81
8.4 Interactions . . . 83
8.4.1 Sculpture . . . 84
8.4.2 Simulation de cheveux . . . 84
8.4.3 Collisions . . . 86
8.5 Contrôle et paramétrage . . . 86

9 Bilan **89**

III FlowVR Render **93**

10 Contexte et problématique **95**
10.1 OpenGL . . . 96
10.2 Chromium . . . 97
10.3 Shaders . . . 97
10.4 Bilan . . . 98

11 Description d'environnements virtuels distribués **99**
11.1 Principes fondamentaux . . . 99
11.2 Primitives graphiques . . . 100
11.2.1 Ressource . . . 101
11.2.2 Primitive . . . 104
11.3 Protocole de communication . . . 105
11.4 API . . . 107
11.5 Exemple . . . 107

12 Schémas de rendu parallèle **111**
12.1 Exemple . . . 111
12.2 Duplication . . . 112
12.3 Répartition des données vers plusieurs noeuds de rendu (*sort-first*) . . . 113
12.4 Rendu local et recomposition des pixels (*sort-last*) . . . 115
12.5 Description parallèle utilisant plusieurs viewers . . . 116
12.6 Asynchronisme . . . 117
12.7 Passage à l'échelle . . . 118

13 Applications **121**
13.1 Visualisation scientifique . . . 121
13.1.1 Couplage avec VTK . . . 121
13.1.2 Extraction de données . . . 122
13.1.3 Rendu volumique . . . 123
13.2 Autres applications . . . 126
13.2.1 Texturage du modèle reconstruit . . . 127
13.2.2 Vidéo haute-définition sur mur d'image . . . 128
13.2.3 Visualisation interactive de l'état d'une grille . . . 129

14 Bilan **131**

IV Couplage de simulations distribuées interactives **133**

15 Introduction **135**
15.1 Simulations physiques . . . 136
15.1.1 Simulation pour les applications graphiques interactives . . . 136
15.1.2 Simulations parallèles et distribuées . . . 137

16 Architecture de couplage **139**
16.1 Conception générale . . . 139
16.2 Objets et animators . . . 140
16.3 Interactors et filtrage des données . . . 141

16.4 Protocole de communication 142
16.4.1 Objects 142
16.4.2 Evènements 142
16.5 Construction de l'application 144
16.5.1 Exemple 144
16.5.2 Application séquentielle synchrone 144
16.5.3 Applications distribuées 146
16.5.4 Multi-fréquences 147

17 Implantation et résultats **149**
17.1 Implantation 149
17.1.1 Objets rigides 149
17.1.2 Fluide 149
17.1.3 Filet masses-ressorts 150
17.1.4 Interactions avec l'utilisateur 150
17.1.5 Visualisation 151
17.2 Résultats 151
17.2.1 Animation séquentielle hors-ligne 151
17.2.2 Parallélisation 151
17.2.3 Exécution interactive 151

18 Bilan **155**

V Conclusion et perspectives **157**

Bibliographie **163**

Figures couleur **177**

Introduction 1

Contexte

Depuis les années 1970 avec l'arrivée des premiers casques de visualisation et les projets précurseurs de simulateurs de vol, la réalité virtuelle n'a cessé d'évoluer vers des environnements de plus en plus complexes, nécessitant le développement de plateformes matérielles et logicielles performantes. Cette évolution est stimulée par des applications industrielles comme la conception de futurs produits (aéronautique, automobile) ou l'exploration de gros volumes de données (recherches pétrolières, mesures ou simulations scientifiques). Ces applications demandent des ressources en calculs importantes pour piloter les différents périphériques, faire face aux traitements nécessaires ainsi qu'aux contraintes liées à l'interactivité. Une tendance actuelle des plateformes de réalité virtuelle est d'exploiter la puissance apportée par des grappes de machines, afin de répondre à moindre coût à ces besoins. Cependant ceci requiert un travail supplémentaire lors du développement des outils et des applications lié à leur nécessaire parallélisation.

Dans le domaine du parallélisme les grappes sont désormais bien maîtrisées. Plusieurs paradigmes d'exploitation subsistent (passage de messages, mémoire virtuellement partagée, parallélisation semi-automatique avec annotation du code), chacun adapté à certains types d'applications. Pour obtenir plus de puissance de calcul les grappes sont désormais agrégées pour former des *grilles de calcul*, réparties à l'échelle régionale ou nationale. L'exploitation de ces plateformes implique de gérer les multiples réseaux (intra et inter grappes), ainsi que l'hétérogénéité des architectures. Ceci entraîne un rapprochement avec les outils d'applications distribuées, basés sur des concepts de couplage de codes et de déploiement d'applications. Ceux-ci doivent toutefois être adaptés pour supporter la parallélisation des éléments de l'application, ainsi que le volume de données échangées.

Problématique

L'application des méthodes du parallélisme aux applications de réalité virtuelle n'est pas immédiate. En effet, ces applications présentent des contraintes supplémentaires primordiales liées aux multiples périphériques utilisés ainsi qu'au besoin d'interactivité. Ainsi, les travaux de parallélisme se concentrent généralement sur le temps global nécessaire à l'exécution d'une application, alors qu'une application interactive nécessite une fréquence minimum de mise à jour de la boucle de calculs, ainsi qu'une bonne réactivité, c'est-à-dire une latence minimale entre une action de l'utilisateur et l'affichage du résultat. Par exemple, une parallélisation basée sur un pipeline permet d'augmenter la fréquence des calculs mais pas ce paramètre de latence.

Disposer de plus de ressources de calcul pour une application de réalité virtuelle permet d'augmenter la complexité du monde virtuel, par exemple en intégrant de plus en plus d'objets, ou en gérant des interactions de plus en plus complexes avec l'utilisateur ou entre objets. Cette évolution introduit une difficulté supplémentaire au niveau du design de l'application, car il est nécessaire d'intégrer un nombre croissant de composants, algorithmes, codes ou librairies. Cette intégration n'est pas triviale car chaque partie peut avoir des contraintes très différentes tant au niveau pratique (langage de programmation, dépendances, formats de données), que théorique (fréquence de fonctionnement, méthode de parallélisation).

Du fait de la complexité de telles applications, ainsi que la présence de nombreux outils ou codes existants, un besoin important concerne la réutilisation des éléments de l'application. Ainsi, la construction de l'application doit permettre l'intégration de composants préexistants, ainsi que la réutilisation des composants nouvellement développés dans les applications futures. Ceci implique d'utiliser des méthodes génériques de couplage de codes, permettant d'adapter aussi facilement que possible les différents composants aux conditions spécifiques liées à la plateforme d'exécution, aux périphériques utilisés, ainsi qu'aux demandes de l'utilisateur.

Un système permettant de résoudre les problèmes précédents permet d'envisager des applications nouvelles. Des programmes de simulation à l'origine *off-line* peuvent être couplés à des dispositifs d'interactions et ainsi permettre de manipuler directement les objets simulés. Plusieurs de ces simulations peuvent ensuite être intégrées dans un même monde virtuel pour gérer différents types d'objets et offrir un plus grand réalisme. Il faut alors trouver des techniques nouvelles pour prendre en compte les nouveaux types d'interactions entre ces objets. Ce genre de système peut être exploité par exemple pour simuler en temps réel une partie du corps humain, qui contient a la fois des objets rigides (os), déformables (muscles, veines), et liquides (sang), et ainsi permettre de simuler une opération chirurgicale.

Contributions

Cette thèse présente un ensemble de modèles et d'outils pour la construction d'applications distribuées interactives. Ces réalisations concernent différents niveaux de l'application. Une série d'applications expérimentales permet d'évaluer leurs utilisations sur des cas concrets.

FlowVR

Notre premier travail repose sur la spécification de *FlowVR*, un nouveau modèle servant de base pour la construction d'applications interactives modulaires, inspiré des travaux en parallélisme et systèmes distribués mais adaptés pour la réalité virtuelle.

Les composants ont très peu d'informations sur le reste de l'application, afin d'en être le plus indépendant possible. Ils peuvent être parallélisés en interne, toutefois les communications entre composants distincts et les politiques de synchronisation et de couplage sont gérées de manière externe. L'accent est porté sur l'utilisation d'éléments simples et relativement indépendants pour gérer chaque fonctionnalité nécessaire. Ainsi il nous est possible d'explorer facilement différentes politiques de couplage, et de développer des extensions adaptées aux besoins de certaines parties de l'application.

L'objectif étant de construire des applications de plus en plus complexes, l'accent est mis sur le passage à l'échelle des mécanismes de développement de ces applications. En particulier, cette construction repose sur l'élaboration de graphes de flux de données générés par un système de scripts permettant un *dépliage* paramétrable des composants sur l'architecture cible. Cette approche permet d'obtenir une représentation visuelle de l'application très utile pour suivre sa conception et apporter des modifications.

FlowVR Render

Sur cette base nous avons travaillé sur la modularisation et l'optimisation de la partie visualisation de l'application, grâce à une description de la scène 3D en primitives indépendantes utilisant des *shaders* pour en contrôler le rendu. Cette représentation permet une grande souplesse sur la conception et le placement des composants de visualisation tout en garantissant une implantation efficace des traitements nécessaires aux techniques de rendu distribué. Grâce à une conception particulière du protocole de communication notre approche évite la plupart des surcoûts des systèmes classiques de rendu distribué lors des opérations de découpage et fusion des différents flux graphiques, liés en particulier aux problèmes de gestion des interdépendances entre les primitives classiques.

La description de l'apparence des objets utilise des *shaders*, permettant d'exploiter pleinement les fonctionnalités avancées des nouvelles générations de cartes graphiques. Grâce à cela une partie des calculs peut être déportée sur les machines de rendu, ce qui modifie le volume de données à transmettre sur le réseau.

Applications

Plusieurs applications ont pu être développées grâce à FlowVR. Nous avons ainsi intégré dans une application de réalité virtuelle des calculs complexes permet de calculer en temps réel une représentation 3D de l'utilisateur à partir d'un réseau de caméras. Cette application est ensuite devenue de plus en plus complexe par l'ajout de nombreux composants d'interactions et de simulations parallèles.

L'architecture de rendu distribuée *FlowVR Render* a abouti à des applications permettant un rendu volumique haute-résolution sur mur d'image, des algorithmes de texturage du modèle 3D de l'utilisateur par reprojection des flux vidéos des caméras, ou encore la visualisation de l'état d'une grille par un ensemble de composants gérant chacun un type d'information.

Enfin une application expérimentale très ambitieuse, combinant dans un même environnement un fluide 3D parallélisé, un moteur de collisions rigides, un filet déformable et la reconstruction 3D multi-caméras de l'utilisateur, constitue l'application la plus complexe développée avec FlowVR. Elle ouvre de nouvelles perspectives quand aux évolutions des approches de construction d'applications permettant de faciliter le développement d'environnement comportant de nombreuses interactions multi-physiques.

Organisation

L'état de l'art présente les principaux travaux en réalité virtuelle (chapitre 2), en parallélisme et systèmes distribués (chapitre 3), ainsi que les outils existants de conception d'applications de réalité virtuelle distribuées (chapitre 4).

La partie II s'attache tout d'abord aux travaux préliminaires liés à Net Juggler et aux premiers prototypes de couplages parallèles simulation/visualisation (chapitre 5), posant ainsi la problématique importante pour la suite, dédiée à la description du modèle de construction d'applications de réalité virtuelle distribuées (chapitre 6), son implantation (chapitre 7) ainsi que son évaluation au travers d'une étude de cas des applications développées (chapitre 8).

Une nouvelle approche de rendu distribué est présentée dans la partie III. Basé sur un modèle de description de la scène (chapitre 11) et des schémas de transmission de cette description (chapitre 12), ce travail permet de construire des applications de visualisation avancées et de les exécuter de manière efficace sur mur d'image (chapitre 13).

Enfin la partie IV présente une première architecture de couplage de haut niveau entre plusieurs simulations interdépendantes, permettant la construction d'applications ambitieuses gérant des environnements virtuels complexes et très interactifs.