Université de Lille 1 SCIENCES ET TECHNOLOGIES

Université Lille 1

Création et développement au sein de Sofa

Rapport de stage de Master 2

soutenu le 1er Septembre 2015

par

Tony GALLOY

Remerciements

Table des matières

Introduction

1 Introduction générale

1.1 Introduction

1.2 Problématique

1.3 Choix du sujet

1.4 Interet

2 Présentation de l ?entreprise

2.1 Inria

2.2 place et role de l ?inria

L ?INRIA est un institut de recherche national sur les domaines de l ?informatique et de l ?automatique. Il est réparti sur 9 centres dans toute la France. Il est dirigé par Antoine PETIT, mais chaque centre possède un sous-directeur qui gère la politique de son centre.

Les chercheurs employés dans ce centre sont divisés en équipes qui ont chacun un thème qui leur est propre. Ces équipes ont une durée de vie de 4 ans, renouvelable 2 fois durant le cours de sa vie, pour un total de 12 ans.

2.3 Son organisation

En plus des chercheurs implémentant de nouveaux algorithmes, INRIA dispose d ?une division, appelé SED. Elle est composé de développeurs expérimentés qui ont pour tache de maintenir les différentes plate-formes expérimentales développés par les equipes inria et assurer un suivi technologique.

2.4 Defrost

J ?ai la chance de faire cette année mon stage de seconde année de master dans l ?une de ces équipes, DEFROsT, spécialisé dans les robots déformables. Cette équipe a été crée suite a la séparation en deux d ?une précédente équipe, SHACRA. En effet, cette dernière a été démantelé suite au terme de sa vie, et une partie des chercheurs qui la composaient ont été réassignés au centre de Strasbourg, tandis qu ?une autre partie a créé une startup.

Shacra

L ?équipe était spécialisé dans l ?informatique au service de la médecine. Elle souhaitait utiliser des méthodes mathématiques et informatiques afin de simuler des déformations d ?organes d'êtres vivants. C ?est ainsi qu ?elle améliora Sofa, un moteur graphique et physique afin d'implémenter les différents algorithmes de déformations de maillages. Afin de l ?aider dans ses recherches, l ?équipe a collaboré avec certains médecins pour se rapprocher des enjeux et besoins de ces derniers.

Suite au succès de l ?une de leurs expériences, simulant une opération de la cataracte, certains membres ont quittés l ?équipe afin de créer leur entreprise, appelé INSIMO. Le reste des membres se sont regroupés au sein d ?une nouvelle équipe, DEFROsT, acronyme de DEFormable RObotic SofTware.

Defrost

Tandis qu ?une partie de l ?équipe ont débutés la création d ?une entreprise, d ?autres membres ont déménagés a Strasbourg, (Insérer une raison quelconque). Le reste de l ?équipe resté a Lille s ?est regroupé autour d ?une nouvelle équipe dirigé par Christian Duriez. Cette équipe, nommé Defrost, a changé de thématique, dorénavant ?Contrôle des robots déformables ?. L ?équipe possède 2 chercheurs dans ses rangs. Ces chercheurs sont responsables de l ?équipe, et l ?un de ces chercheurs est chef de DEFROST. Elle dispose également d ?un ingénieur développeur chargé de la maintenance et du support informatique des solutions développés. Cela inclut l ?amélioration du logiciel existant tel que de nouvelles options de compilation, de créations d ?outils informatiques ou même de gestion du travail partagé. Le reste des collaborateurs sont des doctorants en recherche avec des financements tiers et des ingénieurs en informatique et en mathématique.

Actuellement, la recherche possède deux axes principaux dans le domaine. Le premier est la construction de robots déformables grâce a divers techniques tel que la pneumatique, ou encore des matériaux mous. Cependant, ces robots ne sont pas encore totalement contrôlables, car les degrés de liberté d ?un robot mou sont théoriquement infinis. C ?est pourquoi d ?autres chercheurs recherchent des moyens de contrôler ces robots mous.

L ?équipe pars du principe que le robot peut lui même se déformer pour répondre a un objectif, mais pour cela il a besoin de connaître à tout moment l ?état dans lequel il se trouve. Il a ainsi besoin de pouvoir calculer les déformations apportés a son squelette a chaque pas de temps, et en temps réel. Il peut ainsi déplacer ses parties déplaçable en fonction des paramètres calculés. L ?équipe a créé un prototype, ainsi que la simulation associé. Il s ?agit d ?un robot a quatre ressorts, lié chacun a un moteur par un fil. En fonction du déplacement demandé du bout du robot, les moteurs vont tirer ou relâcher les câbles, et la résistance globale du matériau va placer l ?effecteur (Définir effecteur).

3 Projets

3.1 Anévrisme

L ?anévrisme est un gonflement de certaines artères du corps, qui vont se gonfler de sang. Cet anévrisme grossi avec le temps, et il arrive qu ?il se romp, déclanchant une hémorragie locale. Il est possible de soigner un anévrisme par chirurgie, soit en amenant un cathéter jusqu ?à l ?anévrisme et en insérant un câble particulier qui va durcir et ainsi boucher l ?anévrisme, soit en accédant par diverses manières chirurgicales a l ?anévrisme, et en venant clipper l ?anévrisme, c ?est a dire insérer une pince très puissante et petite autour du collier de l ?anévrisme. Cependant, cet anévrisme peut se situer a des endroits très difficiles d ?accès, tel que le cerveau. Le neurochirurgien doit alors opérer pendant plusieurs heures afin d ?ouvrir le crane du patient, inciser les différentes liaisons entre les parties du cerveau, et sectionner l ?anévrisme. Ces opérations sont très délicates, et sont généralement apprises par les étudiants en assistant des neurochirurgiens expérimentés lors de l ?opération. Seulement, en salle d ?opération, la vie du patient est mise en jeu, et la moindre erreur peut couter la vie du dit patient. Ce genre d ?opération peut également etre exercé sur des corps sans vie, limitant ainsi les risques au maximum. Malheureusement, les quantités de cadavres sont limités, et leur conservation est délicate et demande des installations particulières. C ?est pourquoi l ?équipe s ?est intérressé a la création de simulations basé sur les pratiques actuelles en neurochirurgie afin de simuler une opération de l ?anévrisme. Cette simulation a été séparé en 4 étapes différentes La séparation des lobes afin de dégager le passage La séparation des liens entre l ?anévrisme et les parois autour Le clipagge de l ?anévrisme Le clampage de l ?anévrisme, en cas d ?hémorragie

3.2 Autre projet en cours

1

Sofa

Créé en (Insérer année) , Sofa est un moteur physique servant de base pour implémenter des algorithmes. Il est maintenu par plusieurs équipes de INRIA, et est utilisé par plusieurs entreprises privées.

1.1 Objectif de Sofa

Sofa a pour objectif de fournir aux différentes équipes un environnement maîtrisé complètement. Elles peuvent implémenter les algorithmes sur lesquels elles travailles, et les ingénieurs du SED maintiennent le code source de SOFA afin de fournir de nouvelles possibilités. Sofa est implémenté en C++, est compilable et compatible sur Windows, Mac et Linux, et le code est libre. Il est sauvegardé sur un dépôt git, et la procédure d ?installation est décrite sur le site du logiciel ici. Mais le logiciel est avant tout destiné a la recherche, et le dépôt n ?est disponible qu ?en lecture seule, et les visiteurs ne peuvent pousser les modifications qu ?ils effectuent sur le dépôt principal. Il faut pour cela être inscrit sur le site de la Gforge, un site fourni par INRIA, et récupérer le logiciel grâce à une adresse personnalisé.

1.2 Mécanismes simples

Sofa fonctione par principe de scène. Ces scènes sont des niveaux chargés par sofa afin de créer les composants correspondants. Ces scènes peuvent etre écrites en XML, en python, ou en C++. Elles disposent d ?une architecture particulière. Chaque scène possède un noeud principal. Ces noeud servent de conteneurs de composants, et peuvent être eux même contenus dans d ?autres noeuds. Ces noeuds sont souvent utilisés dans les scènes afin de rassembler les composants avec des thèmes communs, tel que le modèle de collision d ?un objet. Les composants sont des concepts ou des fonctionnalités qui ont été ajoutés a sofa. Parmi les composants que possède sofa, on trouve par exemple des chargeurs de maillages, des outils de collisions, des moteurs de

déformations ou encore des utilitaires de shaders. Certains sont autonomes, mais d ?autres en requiers des spécifiques afin de fonctionner correctement. Chaque composant possède des propriétés communes, tel que son nom, et éventuellement des propriétés particulières. l ?une de ces propriétés communes est le tag. Il s ?agit d ?une case ou l ?utilisateur peut insérer un ou plusieurs mots séparés par des espaces. De manière générale, ces tags vont faire office de marqueur pour certains autres composants. Sofa possède de nombreux composants de base, mais il permet également de charger des plugins, qui sont des packs de composants, afin d ?augmenter les possibilités de sofa. Ce système de plugin a le gros avantage de permettre la création de composants séparément, et également l ?ajout de ces composants par la suite a sofa. C ?est également très pratique lors de la compilation, car il est possible de ne sélectionner que certains plugins, réduisant drastiquement le temps nécessaire à cette compilation.

1.2.1 Les exécutables

Sofa possède deux exécutables principaux. Le premier est le Modeler. Il a pour rôle de donner une interface simple et efficace afin de créer et d ?éditer des scènes. Il dispose pour cela de plusieurs menus.

Au centre se trouve le détail de la scène actuellement ouverte. Chaque noeud fils est décalé par rapport au noeud parent. Un système d ?onglet permet de switcher entre plusieurs scènes ouvertes en meme temps. A gauche se trouve la librairie de composants actuellement disponibles et reconnu par sofa. Ces composants sont triés par catégorie, et il est possible de rechercher par mot les composants dont on connait une partie du nom. Un simple glisser-déposer ajoute le composant au même niveau si on lâche le clic sur un autre composant, soit dans le noeud sélectionné. Par un double clic, il est possible d ?accéder à une fenêtre d ?édition du composant. Cette fenêtre liste l ?intégralité des datas publiques du composant, et liste les valeurs déjà existantes.

Il est possible d ?ajouter des composants dans le gestionnaire de plugins accessible dans le menu ?Edit ?. Tous les composants disponibles dans les plugins ajoutés seront immédiatement ajoutés dans la librairie.

1.3 Mécanismes complexes

1.3.1 Les Datas

Certaines données en sofa sont contenus dans un type particulier, les Datas. Ce sont des encapsuleurs permettant plusieurs choses. Tout d ?abord, Le contenu des datas peux etre partagé entre plusieurs datas, y compris des datas de composants differents. Et c ?est leur utilisation la plus courante. Cela évite la copie de données parfois très volumineuses.

1.3.2 Les MechanicalObjects

Ces composants représentent les états mécaniques des objets modélisés. La totalité des comopsants qui appliquent des forces ou des modifications Les engines Un engine, c ?est un composant sofa qui va calculer un lot de données de sortie depuis un lot d ?entrée. La stratégie de gestion de ces données est la stratégie paresseuse. C ?est a dire que les données d ?un engine ne seront (re)calculés qu ?a condition qu ?un autre composant y fasse appel durant l ?execution

2

HMD

Les opérations menés par les neurochirurgiens sont minimalement invasive, c ?est a dire qu ?elles doivent au minimum toucher au corps des patients. Afin d ?atteindre cet objectif, ils effectuent le moins d ?interactions possibles. C ?est pourquoi ils se sont équipés d ?outils divers, parfois changeant drastiquement les méthodes de travail, tels que les microscopes sur pied. C ?est pourquoi on m ?a demandé de rendre compatible Sofa et les casques de réalité virtuelle.

2.1 La stéréovision

La stéréovision est le prinipe de reconstituer une scène 3D a partir d ?images 2D. Il s ?agit la d ?une technique communément utilisé chez les animaux. Les yeux capturent une image chacun, et l ?envoient au cerveau par l ?intermédiaire des nerfs optiques. Ces images vont être analysés par plusieurs parties du cerveau, dont l ?une va se charger de reconstruire la scène mentalement en 3D. Cela permet d ?avoir une notion de profondeur de champ, malgré la capture en 2D.

Ce principe est a l ?étude chez l ?humain depuis plus d ?un siècle, et a été porté sur ordinateur depuis les années 80, naissance de la vision par ordinateur.

2.1.1 Les casques de réalité virtuelle

Depuis quelques années, les casques de réalité virtuels(HMD = Head-mounted devices) ont eu un regain de popularité. La cause est simple : La technologie s ?est développé a un point tel que ces casques sont devenus performants, tout en conservant un prix tout public. C ?est pourquoi les membres de DEFROST se sont intéressés a l'intégrer dans sofa.

2.1.2 L ?intégration

Dans le cas de la stéréovision, nous avons besoin d ?au moins deux images. Malheureusement, certains casques ne possèdent qu ?un seul écran. Ce problème est contourné par une simple sépa-

ration d ?écran. Afin de réaliser cela dans Sofa, il est necessaire d ?utiliser un concept d ?OPENGL particulier : Les framebuffers. Les framebuffers sont des tableaux gérés par OPENGL, et contenant des pixels. Le plus utilisé est celui de l ?écran, et c ?est justement celui la que l ?on va utiliser. On va tout d ?abord réaliser un premier dessin sur une partie de l ?écran, puis nous allons soit sélectionner une seconde caméra, soit utiliser la première afin de réaliser une seconde image, sur l ?autre moitié de l ?image. Tous les HMD possèdent soit des doubles écrans alignés sur l ?horizontale, soit un seul écran bien plus long que haut. De plus, les yeux sont également alignés sur l ?horizontale, nous allons donc dessiner les images l ?une a gauche de l ?autre. Maintenant que nous savons comment dessiner les images, il nous faut nous intéresser a comment positionner les images afin d ?avoir la stéréo-vision la plus réussi

Il existe pour cela 2 types de vision : L ?approche Toed-in Les deux caméras sont séparés de quelques centimètres, et les normales du plan de chaque caméra se croise en un point de l ?espace. L ?approche parallèle Les plans des deux caméras sont parallèles l ?un a l ?autre.

Un dernier problème se pose alors. Les lentilles des HMD sont courbes, et cela implique que l ?image qui sera diffusé a l ?utilisateur comportera des aberrations chromatiques. Ces aberrations sont intimement liés a la forme de la lentille. Plus elle sera courbée, plus les composantes seront soit approchés du centre dans le cas de la composante rouge, soit éloigné du centre dans le cas de la composante bleu. La composante verte de l ?image n'étant pas affecté significativement, elle sert donc de référence pour les modifications des deux autres composantes. Une solution efficace permet de corriger ce problème. Il s ?agit des shaders. Les shaders sont des programmes qui sont compilés a l'exécution du programme par les pilotes du chipset graphique, qu ?il s ?agisse d ?une carte graphique intégré a un processeur, ou une carte graphique dédié. L ?un des atouts des shaders est que le langage de programmation de ces shaders est standard, c ?est a dire qu ?il est reconnu par tous les constructeurs de cartes graphiques, ils sont donc universels. Les shaders sont intégrés a Sofa, et il est possible d ?utiliser des composants pour les ajouter a la scène. Cela ce réalise en plusieurs étapes Tout d ?abord, il est nécessaire d ?ajouter a une scène un VisualManagerPass. Ce composant va assurer le dessin de tous les objets qui seront taggués avec le même mot que le VMP. Il est ensuite nécessaire d ?ajouter un VisualManagerSecondaryPass afin de traiter l ?image complète. Ce composant va récupérer les images de sortie des VMP ayant les tags identiques à ceux indiqués comme tags d ?entrée du composant, et va d ?abord les fusionner en une seule image, dans l ?ordre des tags. Il va ensuite appliquer le shader indiqué en lui passant comme paramètre l ?image créé, en tant que texture. A partir de cette étape, il devient aisé d ?appliquer notre modification. En effet, il suffit pour chaque pixel de sélectionner une composante plus ou moins loin, en fonction d ?un ratio depuis le centre de l ?image. Cette transformation est donné par la formule :

(st.x-0.5)*factX +0.5 ;

Ou ST est les coordonnées du pixel courant, et FactX est le facteur attribué a la composante.

Une valeur supérieure a 1 va attirer les composantes vers le centre de l ?image. A l ?inverse, une valeur inférieure va pousser les composantes vers les bordures. Ce facteur est intimement dépendant des caractéristiques des lentilles disponibles sur les HMD. Il est donc nécessaire de le recalculer pour chaque casque disponible. Mais il sera également le même quel que soit la scène. Naturellement, cette modification sera visible sur l ?image d ?origine, si elle est diffusé sur un écran. Mais elle compensera les distorsions créés par les lentilles du HMD. C ?est pourquoi il est conseillé de créer deux scènes, l ?une qui tournera sur un écran classique, et l ?autre qui sera sur un HMD.

VisualManagerPass = Dessiner les obgets taggés dans un FBO Render pass element : render the relevant tagged objects in a FBO CompositingVisualLoop = render multiple passes and composite them into one single rendered frame

3

Opération du crane

Durant l ?opération d ?un anévrisme, le neurochirurgien peut avoir a ouvrir le crane du patient afin d ?accéder a l ?anévrisme. Mais cette étape n ?a pas encore été modélisé. C ?est pourquoi on m ?a demandé de créer une scène ou l ?utilisateur pourrait s'exercer a cela.

3.1 Déroulement en bloc opératoire

Une fois la chair découpé et écarté, les neurochirurgiens vont devoir ouvrir le crane. Il s ?agit d ?une étape très importante qui donnera un chemin au docteur afin de pratiquer l ?opération. Elle est standard et fixe.

Le neurochirurgien va d ?abord bruler le crane grace a une pince électrique, afin de dessiner 3 cercles et les liens entre ces trois cercles. Ces dessins vont lui servir de guide afin de percer 3 trous grâce a une perceuse spéciale qui s'arrête dès que l ?os est complètement percé, afin d ?épargner le cerveau juste en dessous, puis découpe l ?os en suivant les courbures de l ?avant du crane grâce a une scie sauteuse avec frein afin la aussi d ?épargner le cerveau.

3.2 Modélisation des outils

Le seul outil à modéliser est la pince électrique. Cette pince peut être remplacé par un outil avec un simple embout, étant donné que les deux mâchoires doivent être quasiment en contact pour délivrer un courant, et que le dessin est simplement montré par un trait sur le crane. J ?ai donc choisi comme représentation un stylo, car il permet de symboliser le point de contact avec la pointe de celui ci.

3.2.1 Le chargement des données

Il existe énormément de méthodes pour stocker des maillages dans un fichier. Et pour chaque type de fichier existe un composant MeshXXXLoader, où le XXX désigne le type de fichier

que le composant est capable d ?ouvrir. Ces fichiers vont ensuite stocker dans leurs datas les informations du maillage donné. Il est par la suite possible de les afficher par l ?intermédiaire d ?un OGLModel. Ce composant peut soit utiliser les informations d ?un loader, soit une URL afin d ?afficher un maillage. Mais il ne peux charger que des fichiers enregistrés en ?.obj ?.

3.2.2 Le modèle de collision

La collision dans les moteurs de jeu ont toujours été des tests très couteux en temps de calcul. En effet, afin de tester une collision, on doit tester tous les triangles des deux maillages entre eux. Cela peux atteindre des temps de calculs en (Nr). C ?est pourquoi l ?une des solutions est d ?utiliser deux maillages. Le premier mailalge servira a afficher l ?objet. Ce maillage pourra etre détaillé afin de rendre quelque chose de fidèle. Le second sera un mailalge très épuré qui reprendra la forme globale du maillage visuel. J ?ai donc utilisé blender afin de créer un maillage épuré de l ?outil

3.2.3 Les mappings en Sofa

Le mapping est le mécanisme qui lie deux maillages entre eux. C ?est a dire que toute modification appliqué a l ?un des maillages va se transférer à l ?autre maillage. Cette liaison dépend du mapping utilisé dans les simulations. L ?identityMapping permet de mapper deux maillages identiques. Le rigidMapping permet quand a lui de lier deux objets par rapport a leur centre de gravité. Ainsi, tout mouvement dans l ?espace se résumera a un mouvement de translation et de rotation. L ?outil ne sera pas modifié, on peux donc utiliser un rigidMapping afin de lier le modèle visuel avec le mechanicalState de l ?objet. (Pourquoi ? Sa sort d ?ou ? Hein ?)

3.2.4 Le Phantom omni

L ?omni est un bras mécanique à retour d ?effort. Il possède cinq axes de rotation mais seulement 3 permettent de fournir un effort, sur leur axe de rotation. Il dispose malgré tout d ?un positionnement à six degrés de liberté. Sofa possède plusieurs modules de gestion de l ?omni, mais seulement l ?un des trois fonctionnait lors de mon stage. J ?ai donc choisi le NOD. Dans sofa, le NOD a un fonctionnement un peu particulier. Il va chercher un mechanicalObject de type Rigid dans le noeud courant, et va lui transférer tout mouvement subit par le stylet. Le calcul des forces est lui calculé par les XXXSpringForceField. Ces composants vont accrocher des ressorts entre deux MO. Plus ce ressort sera étiré, plus les forces calculés seront importantes, et plus le Phantom renverra de force. Afin de créer cette différence, ce ressort sera placé entre le MO de l ?omni et le MO de la collision de l ?instrument. Ainsi, quand notre instrument va entrer en collision avec un objet, il va se séparer du MO de l ?omni qui ne dispose pas de collision, ce qui va étirer le ressort, qui va engendrer des forces sur le phantom.

3.3 Modélisation du crane

Une fois l ?outil créé, il faut modéliser un crane humain. De la, plusieurs objectifs sont a réaliser.

3.3.1 Les Objectifs

Le principal objectif de la scène est que les utilisateurs doivent pouvoir voir leur tracé sur le crane. Un autre de ces objectifs est de pouvoir comparer le tracé de l ?utilisateur avec le tracé d ?un chirurgien expérimenté. Un troisième objectif est de permettre de percer le crane. Un dernier objectif consiste a atteindre les 60 images par seconde constants.

3.3.2 Une image

Avant une opération d ?anévrisme cérébral, chaque patient doit passer un IRM. Cet IRM enregistre couche par couche les différentes parties du cerveau et permet de naviguer parmi ces couches. Ces images sont par la suite enregistrés dans différents formats possibles pour pouvoir être utilisés par la suite. On m ?a confié plusieurs de ces fichiers. Tous avait une base commune : Un crane sans mâchoire. Les parties de l ?image a l'intérieur du crane sont a 1, tandis que les parties a l'extérieur sont a 0. Les autres fichiers contiennent un crane avec un énorme trou. Il s ?agit du modèle utilisé dans les autres scènes déjà existantes. Le trou a été réalisé bien plus grand afin de faciliter les démonstrations. Il existe un plugin Image dans sofa qui permet de traiter des images. Ce plugin contient plusieurs composants capables de charger des images, de les modifier ou de les transformer.

3.3.3 Le marching cube

Un composant particulier de sofa m ?a aidé a le transformer en mesh : le MarchingCubesEngine. Le MarchingCube est une technique publié en 1987. Elle permet de transformer une image 3D en maillage. L ?idée est de séparer l ?espace en cubes de taille identique, mais ces cubes n ?ont pas forcément besoin de correspondre à la taille d ?un pixel de l ?image. Dans ce cas, on utilisera une moyenne pour chaque sommet des cubes, qui pourront être utilisés pour chaque cube. On construit par la suite un tableau de référence. On va définir également une valeur, appelé Isovalue. Cette valeur va être une valeur seuil qui va nous indiquer si le sommet actuel est dans la figure ou ne l ?es pas. Ainsi, si le sommet possède une valeur supérieure a l ?isoValue, on placera sa valeur a 1. Sinon, sa valeur sera de 0. Pour chaque cube, on va récupérer les valeurs de chaque sommet. Étant donné que l ?on a huit sommets, on se retrouve avec un octet représentant les différentes possibilités de remplissage, soit 256 valeurs de référence. Mais il est possible de simplifier les calculs. Il est possible de retrouver certains modèles simplement en . Pour chaque octet

calculé, on prends la correspondance dans le tableau de référence. On obtient ainsi un maillage complet associé a l ?image de départ.

3.3.4 La collision

Comme dit précédemment, la collision est un