Université Nice SOPHIA ANTIPOLIS

Polytech'Nice-Sophia
Département Sciences Informatiques

INGENIEUR EN SCIENCES INFORMATIQUES

RAPPORT DE STAGE DE QUATRIEME ANNEE

Développement et intégration de nouvelles fonctionnalités sur le logiciel ShiVa

Stagiaire : **MAXIMILIEN MOUSSALLI** (SI4)

Maître de stage : **Nicolas PERI**

ENTREPRISE : **STONETRIP** (Villeneuve-Loubet)

21 juin 2010 - 21 septembre 2010

Résumé

L'entreprise "Stonetrip" qui m'a accueilli a développé un logiciel nommé "ShiVa" qui permet aux utilisateurs de produire des jeux vidéo 3D. Le travail qui m'a été confié consistait à implémenter/étendre le logiciel "ShiVa" en ajoutant de nouvelles fonctionnalités, principalement l'intégration de librairies disponibles sur le marché, offrant des services tels que la gestion du son (lecture de tous les formats de sons, simulation de sons 3D, etc), ou la gestion des phénomènes physiques (collisions, forces, gravité, etc). Le langage de programmation utilisé est le C/C++. J'ai de plus utilisé le logiciel "ShiVa" pour vérifier le bon fonctionnement des fonctionnalités que j'ai implémentées.

Stonetrip

1 Remerciements

Je tiens avant toute chose à remercier plusieurs personnes qui ont participé au bon déroulement de mon stage.

Tout d'abord M. Belhassen, Directeur général de Stonetrip, pour m'avoir accueilli au sein de son entreprise et m'avoir confié un travail attrayant.

M. Peri, M. Giraud et M. Maccini pour l'aide efficace qu'ils m'ont apporté tout au long de ce stage.

Ainsi qu'à tous ceux qui ont contribué à l'ambiance chaleureuse qui règne dans cette petite entreprise.

Sommaire

1 Remerciements 2

2 Introduction 4

3 Description du travail proposé 5

4 Description du travail réalisé 7

4.1 FMOD 7

4.1.1 1ère semaine : Analyse 7

4.1.2 2ème et 3ème semaines : Développement 8

4.1.3 4ème semaine : Démonstration 9

4.1.4 5ème semaine : Tutoriels 10

4.2 Clothing 11

4.2.1 1ère semaine : Analyse et développement 11

4.2.2 2ème semaine : Développement et démonstration 12

4.3 Physx 14

4.3.1 1ère semaine : Analyse et développement basique 14

4.3.2 2ème semaine : Développement du “clothing” et des “joints” 15

4.3.3 3ème semaine : Développement des particules 16

4.3.4 4ème semaine : Développement des “soft body” 16

4.3.5 5ème semaine : Documentation, tutoriels et démonstration 18

5 Planning du stage 19

6 Conclusion 20

7 Bibliographie 21

7.1 Rapport 21

7.2 Stage 21

8 Abstract 22

2 Introduction

Dans le cadre de ma 4 ème année en cycle ingénieur à Polytech'Nice Sophia dans le département Sciences Informatiques de Sophia-Antipolis, j'ai été amené à effectuer un stage d'une durée de treize semaines en entreprise, du 21 juin au 21 Septembre 2010, afin d'acquérir une expérience professionnelle.

L'entreprise "Stonetrip" qui m'a accueilli est basée à Villeneuve-Loubet a conceptualisé et développé un logiciel nommé "ShiVa" qui permet aux utilisateurs de produire principalement des jeux vidéos 3D de qualité, rapidement, et sur de nombreuses plate-formes du marché (PC, MAC, iPhone, Linux, etc...). L'unité dans laquelle j'ai travaillé était composée de sept employés et de trois stagiaires dont deux de Polytech'Nice Sophia (M. Giraud et moi même) avec qui j'ai travaillé en collaboration. Le dernier stagiaire M. Maccini travaillait principalement sur le développement d'applications produites avec "ShiVa", il m'a été d'un grand secours pour l'apprentissage de ce logiciel.

Le travail qui m'a été confié consistait à implémenter/étendre le logiciel "ShiVa" en ajoutant de nouvelles fonctionnalités, principalement l'intégration de librairies disponibles sur le marché, offrant des services tels que la gestion du son (lecture de tous les formats de sons, simulation de sons 3D, etc), ou la gestion des phénomènes physiques (collisions, forces, gravité, etc). Le langage de programmation utilisé dans le cadre du projet proposé est le C/C++, langage dans lequel l'École Polytech'Nice Sophia donne une bonne formation, ce qui m'a permis une adaptation au projet très rapide. J'ai de plus utilisé le logiciel "ShiVa" pour vérifier le bon fonctionnement des fonctionnalités que j'ai implémentées. J'ai donc appris à maîtriser les outils propres au logiciel, et principalement le langage de script LUA qui est utilisé pour permettre à l'utilisateur de programmer les divers éléments qui composeront son jeu (gestion caméra, déplacements, intelligence artificielle, etc). Pour m'aider j'avais mon tuteur de stage M. Peri qui m'a expliqué les détails du projet et un exemple simple d'intégration de fonctionnalités sur lequel m'appuyer.

Mon rapport de stage se découpe en trois parties bien distinctes. Tout d'abord, je tacherai de présenter plus précisément le travail que l'on m'a proposé au sein de "Stonetrip". Ensuite je parlerai du travail que j'ai réalisé en décrivant le contexte, les problèmes que j'ai rencontrés, les diverses solutions trouvées, les résultats obtenus, les éléments qui m'ont soutenus dans mon travail qu'ils soient humains, matériels ou logiciels. Enfin, pour finir, je ferai une conclusion en essayant de faire une critique personnelle de mon travail, en me situant d'une part du coté de l'entreprise, d'une autre part de mon point de vue et puis sur la formation que j'ai acquise à Polytech'Nice Sophia au cours de ces deux dernières années.

3 Description du travail proposé

Le logiciel “ShiVa” (anciennement appelé “Stonetrip development kit”) est capable de produire des jeux vidéos 3D multi-plate-formes, multi-joueurs et de grande qualité.

Logiciel ShiVa

Toujours en quête d’améliorer le service du logiciel, de nouvelles fonctionnalités sont implémentées chaque jour pour offrir aux utilisateurs le meilleur confort et les meilleures performances d’utilisation du logiciel.

Pour la dernière version de “ShiVa”, qui n’est pas encore sortie, l’équipe de développement offre la possibilité aux utilisateurs de créer et développer leurs propres fonctionnalités à travers ce qu’on appelle dans notre jargon d'informaticien un “plugin”.

Voici une brève définition : *En informatique, un “plugin” est un logiciel qui complète un logiciel hôte pour lui apporter de nouvelles fonctionnalités.*

Je pense que pour que ce soit plus clair, un petit exemple ne ferait pas de mal. Prenons un logiciel qui lit des vidéos. Il existe un grand nombre de formats de vidéos et chacun se décode d’une façon particulière. C’est dans ce genre de situations que les “plugins” sont très utiles. Plutôt que d'intégrer tous les outils de décodage directement dans le logiciel de lecture de vidéos (logiciel de base), on peut mettre à disposition tout un éventail de “plugins” dont chacun se chargera de décoder un format vidéo particulier. En effet les “plugins” ne sont pas nécessaires au fonctionnement du logiciel de base et donc ne sont pas obligés de prendre une place inutile chez le client qui n’a pas forcément besoin de lire des formats très peu

répandus. Il peut donc décider des “plugins” qu’il veut utiliser ou non pour que le logiciel corresponde à ses besoins.

Les “plugins” dans “ShiVa” permettent aux utilisateurs d’utiliser dans le langage de script LUA les nouvelles fonctionnalités qu’offrent ces “plugins”.

Mon travail a consisté à développer trois “plugins” dont deux pour mettre à disposition les possibilités offertes par des librairies disponibles sur le marché. En informatique, une librairie (ou bibliothèque) est un ensemble de fonctions utilitaires, regroupées et mises à disposition afin de pouvoir être utilisées sans avoir à les réécrire. Les fonctions sont regroupées de par leur appartenance à un même domaine conceptuel (mathématiques, graphique, tris, etc). Le premier “plugin” consistait à intégrer la librairie FMOD, qui est une bibliothèque multi-plate-formes de gestion du son. Le second “plugin” devait ajouter des fonctionnalités permettant la simulation des comportements de tissus/vêtements en temps réel dans le logiciel “ShiVa”. Le dernier “plugin” devait intégrer la librairie “Physx” qui est un moteur physique (NB : Un moteur physique est, en informatique, une bibliothèque logicielle indépendante appliquée à la résolution de problèmes de la mécanique classique. Les résolutions typiques sont les collisions, la chute des corps, les forces, la cinétique, etc...). A coté du développement de ces trois “plugins” j’ai réalisé des démonstrations, qui serviront d’accroche publicitaire à Stonetrip pour montrer les possibilités qu’offrent ces nouveaux “plugins” dans le développement de jeux sur “ShiVa”. J’ai aussi réalisé des tutoriels ainsi qu’une documentation en anglais pour chacun des trois “plugins” pour que les utilisateurs de “ShiVa” puissent apprendre à les utiliser dans leurs jeux.

4 Description du travail réalisé

La description des trois “plugins” réalisés sera divisée en trois sous-parties propres à chaque “plugin”.

Le but des plugins qui intègrent une librairie (comme FMOD et Physx) est d’offrir aux utilisateurs de ShiVa, la possibilité d'accéder à toutes les fonctionnalités offertes par la librairie dans le langage utilisé par ShiVa : le script LUA. Le problème est que l'intégration de librairies dans ce langage est impossible. Sans rentrer dans les détails, il faut coder ce que l’on appelle dans notre milieu des “wrappers” sur chaque fonction que fournit la librairie. Un “wrapper” permet de convertir l'interface d'une fonction (paramètres d’entrée et de sortie) en une autre interface dont l’utilisateur a besoin. Dans notre cas transformer du code C/C++ en code LUA.

4.1 FMOD

FMOD a été le premier “plugin” que j’ai développé. C’est une bibliothèque multi-plate-formes de gestion du son, qui est très utilisée dans le développement de jeux vidéos car elle permet entre autre, de gérer facilement les sons 3D, c’est à dire qu’un son dans un jeu 3D aura un volume différent selon la distance entre le joueur et la source sonore, ou encore si un son est atténué par un mur...
Voici une liste des jeux vidéo les plus connus qui utilisent FMOD :

- Starcraft II : Wings of Liberty
- Crysis 2
- Lara Croft and the Guardian of Light
- Need for Speed: Shift
- Warhammer 40,000 Dawn of War 2: Chaos Rising
- Bioshock 2
- Dragon Age
- World of Warcraft: Cataclysm

Il m’a fallu cinq semaines pour réaliser ce “plugin”, une démonstration et des tutoriels.

4.1.1 1ère semaine : Analyse

J’ai travaillé en collaboration avec M. Giraud (stagiaire) lors de la première semaine de stage, puis des “plugins” particuliers lui ont été attribués, et j’ai continué le développement seul tout le reste du stage, mais régulièrement on s'entraidait quand l’un ou l’autre avait des

difficultés.

Dans un premier temps il a été nécessaire d'installer la librairie FMOD et de découvrir les possibilités qu'elle offre à travers la documentation et les exemples fournis lors de l'installation.

On a ensuite fait quelques essais d'intégration très basiques pour vérifier si la lecture de sons était possible et surtout pour apprendre comment développer un "plugin" pour ShiVa. Heureusement M. Peri (tuteur) a été d'un grand secours au début, pour nous expliquer les fonctionnalités de ShiVa. De plus il nous a fourni un exemple de "plugin" montrant quelques ajouts de fonctions dans ShiVa via des fonctions C/C++.

Après ce petit test d'intégration, nous avons analysé l'interface de toutes les fonctions de FMOD et trouvé une solution pour que chaque fonction puisse être utilisée en LUA dans ShiVa, car le développement des "plugins" se fait en C/C++ et les fonctions que doit fournir le "plugin" sont en LUA et cela pose des problèmes majeurs car, par exemple, en LUA l'utilisation de pointeurs est impossible (*NB : un pointeur est une variable contenant l'adresse mémoire où est stockée la vraie variable)*. il est à noter que FMOD utilise beaucoup les pointeurs.

Pendant cette analyse, nous reportions chaque jour à M. Peri tous les problèmes que nous avions trouvés la veille, nous proposions les solutions que nous pensions implémenter et il nous en proposait d'autres si les nôtres ne lui convenaient pas.

A coté nous ajoutions dans le code C/C++ du "plugin" la définition des fonctions LUA (à ce moment là, les fonctions ne font aucun traitement) en précisant pour chaque fonction le nom de la fonction en LUA, ses paramètres d'entrée, ses paramètres de sortie, et une brève description du rôle de la fonction (généralement calquée sur la documentation de FMOD). A la fin de la semaine nous avions défini toutes les fonctions (environ 300).

4.1.2 2ème et 3ème semaines : Développement

Après avoir imaginé toutes les solutions pour réussir l'intégration des fonctions. Il m'a fallu les développer. Ce travail a été très long et fastidieux. Devant le grand nombre de fonctions à produire, j'ai décidé pour avoir un endroit où commencer, de développer les fonctions qui étaient utilisées dans les exemples fournis par FMOD. Ainsi j'implémentais en priorité les fonctions les plus utiles, et je pouvais les tester en LUA en suivant l'exemple et vérifier le bon fonctionnement de mon code.

Lors de la réalisation des 300 fonctions, je tombais parfois sur des problèmes que l'on n'avait pas prévus ou pas vus. J'ai dû pour certaines fonctions laisser tomber leur intégration car impossible, trop compliquée pour le peu d'utilité qu'elle fournit, voir inutile. Bien évidement je

ne faisais pas cela sans avoir l'accord de M. Peri à qui je rapportais régulièrement les derniers problèmes rencontrés.

4.1.3 4ème semaine : Démonstration

Pour montrer aux utilisateurs les possibilités offertes par le "plugin" j'ai fait une démonstration des possibilités de FMOD dans ShiVa. Je me suis inspiré d'une démonstration faite par FMOD. Cette démonstration montre les possibilités des sons 3D dans un jeu vidéo. Ma démonstration était simple: avoir un décor avec quelques murs, et permettre au joueur de se déplacer librement.

M. Maccini un autre stagiaire à été chargé de réaliser le développement de la base de cette démonstration sur ShiVa. Puis j'ai rajouté moi-même des sources sonores sur le terrain et j'ai développé le code LUA via le "plugin" FMOD et ses fonctionnalités utiles à la gestion de sons 3D. Selon la position du joueur celui-ci entend le son plus ou moins fort en fonction de la distance entre lui et la source. Il peut entendre le son en stéréo plus fort du coté droit s'il est situé à gauche de la source sonore. S'il y a un mur situé entre le joueur et la source sonore le son est atténué, on appelle cela "l'occlusion". J'ai placé aussi quelques effets de réverbération du son qu'offre FMOD pour simuler la réverbération spécifique que certaines pièces produisent comme une salle de bain ou un hall.

Cette démonstration a abouti sur l'ajout de fonctions "magiques" dans le "plugin". C'est à dire des fonctions qui ne sont pas dans FMOD mais que j'ai rajoutées moi-même et qui simplifient grandement le code à écrire en LUA pour gérer tout ce qui est occlusion du son par les murs, la position et le déplacement des sources sonores, etc... Ces fonctions sont très spécifiques aux outils ShiVa.

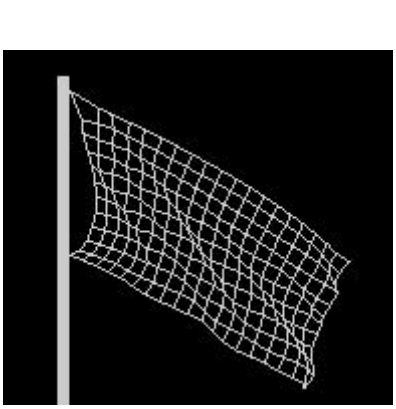
Par contre l'occlusion des murs avec des propriétés spécifiques comme une rotation posaient un problème sur ma démonstration car l'occlusion du son n'était pas cohérente par rapport à la position/rotation du mur, après avoir passé beaucoup de temps à chercher dans mon code ce qui provoquait l'erreur avec l'aide de M. Peri et M. Giraud nous n'avons pas trouvé. Nous avons donc expliqué notre problème sur le forum officiel de FMOD pour recevoir de l'aide. Il s'est avéré en fait que le problème venait de la librairie FMOD. Cela m'a vraiment surpris car c'est une librairie très utilisée dans le jeu vidéo depuis plusieurs années, et il me semblait impensable qu'il subsiste un bug de cette envergure dans une librairie professionnelle à 3000 dollars le projet...

4.1.4 5ème semaine : Tutoriels

Lors de cette semaine j'ai écrit en anglais plusieurs tutoriels pour expliquer aux utilisateurs de ShiVa comment utiliser le plugin FMOD que j'ai réalisé.

4.2 Clothing

Le “clothing” est le nom donné à la simulation de comportements des tissus en temps réel sur ordinateur, comme par exemple un drapeau qui flotte au bout d’un mât, ou une cape. Un tissu sera représenté au niveau graphique par plusieurs points. En appliquant des forces pour simuler l’élasticité entre chaque point du tissu, on peut obtenir un résultat réaliste.



Le 2ème “plugin” que je devais développer devait permettre l’utilisation du “clothing” dans ShiVa. Il m’a fallu deux semaines pour réaliser ce “plugin”, ainsi qu’une démonstration. Comparé au “plugin” de FMOD, la réalisation du “plugin” de “Clothing” n’a pas du tout été la même car je n’ai pas intégré de librairie déjà existante mais j’ai dû coder les algorithmes de simulations du “clothing” moi-même. Pour m’aider M. Peri a mis à ma disposition un code source gratuit qu’un internaute avait développé.

4.2.1 1ère semaine : Analyse et développement

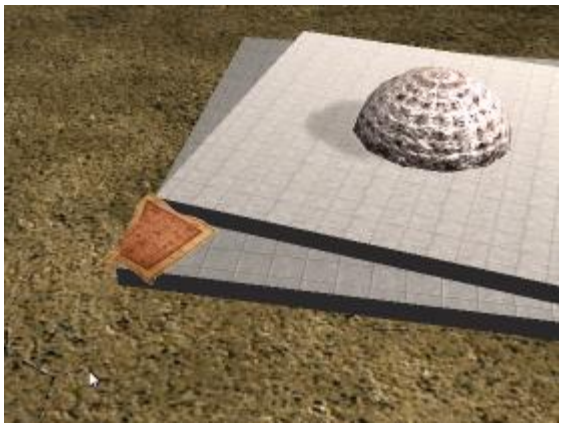
Durant cette semaine j’ai tout d’abord regardé et analysé le code fourni pour comprendre déjà le fonctionnement du “clothing”, puis pour décider les fonctions que j’allais ajouter en LUA pour permettre d’exploiter toutes les possibilités du “clothing”. J’ai ensuite mis en place une version du “plugin” de façon assez primaire pour avoir une première approche avec le “clothing”. J’ai assez rapidement obtenu un tissu qui pendait accroché par un point fixe. J’ai ensuite rajouté des objets de collision, tout d’abord ceux proposés par le code qui m’a été fourni, les plans et les sphères, les deux objets de collisions les plus basiques. J’ai ensuite retravaillé mon “plugin” pour qu’il soit plus pratique et offre plus de fonctionnalités que la version primaire. J’ai fait tourner la partie qui calcule les mises à jour du tissu dans un “thread”, c’est à dire un processus indépendant qui tourne en parallèle du logiciel ShiVa. L’intérêt est que les calculs se font aussi rapidement que possible par le PC, et on y gagne en précision. J’ai également ajouté de nouveaux types de collisions notamment avec n’importe quelle géométrie convexe. J’ai fait aussi de nouvelles fonctions qui permettent de

connaître le point du tissu qui est touché par la souris via une projection 3D sur le tissu en fonction de l'angle de vision de la caméra et de la position de la souris pour permettre d'attraper le tissu avec la souris.

4.2.2 2ème semaine : Développement et démonstration

Suite à l'ajout d'un "thread" dans mon "plugin" je devais gérer de nouveaux traitements dans celui ci. En effet les données du tissu doivent être accessibles à tout moment par ShiVa pour mettre à jour l'ensemble des points du tissu. Mais à coté mon "plugin" calcule en boucle dans un "thread" ces points. Or un accès simultané à une même donnée peut engendrer des conflits.
Il faut donc que j'utilise ce que l'on appelle des "mutex" dans notre jargon pour qu'il y ait un seul accès aux données simultanément. Un "mutex" est un outil de synchronisation qui permet d'éviter que des ressources partagées d'un système ne soient utilisées en même temps. Heureusement, cette année à Polytech'Nice Sophia nous avons eu beaucoup de cours sur la synchronisation de processus. Mais il m'a fallu sortir des solutions simples vues dans ces cours pour trouver des solutions plus complexes notamment pour optimiser au mieux l'accès aux données sans limiter la vitesse de calcul.
Pour réaliser ma solution, j'ai mis en place une copie des données de mon tissu. A la fin de chaque calcul des points de mon tissu par le "plugin", je mets à jour les données dans la copie. De son côté, Shiva récupère et met à jour le tissu à partir des données de la copie. Ainsi le seul moment où il y a un risque d'accès multiples aux données, c'est quand je mets à jour cette copie, c'est à dire rarement. Toujours pour réduire le nombre d'accès aux données partagées, j'ai dû mettre en place une liste d'attente des actions que l'utilisateur veut effectuer sur le tissu. En fait si l'utilisateur demande une action telle que changer la position d'un point du tissu, cette action ne s'effectuera qu'à la fin du calcul en cours juste avant la copie des données du tissu dans la copie.

Enfin pour finir cette semaine j'ai fait une démonstration pour décrire les possibilités offertes par mon "plugin" de "clothing". Après quelques semaines d'utilisation de ShiVa, j'ai pu me débrouiller seul pour réaliser cette démonstration. La démonstration est simple : Un décor avec quelques obstacles, plans, sphères et parallélépipèdes rectangles. Je fais tomber des tissus sur ces obstacles et on peut voir les tissus se déformer à leur contact. J'ai aussi rajouté un drapeau, sur un mat qui flotte grâce au vent. On peut attraper un point du tissu avec la souris et le déplacer.



4.3 Physx

Physx est un moteur physique 3D. Un moteur physique est, en informatique, une bibliothèque logicielle indépendante appliquée à la résolution de problèmes de la mécanique classique. Les résolutions typiques sont les collisions, la chute des corps, les forces, la cinétique, etc. Les moteurs physiques sont principalement utilisés dans des simulations scientifiques et dans les jeux vidéo.
Voici une liste des jeux vidéo les plus connus qui utilisent Physx :

- Mafia II
- Batman: Arkham Asylum
- Cryostasis
- Metro 2033
- Dark Void
- Mirror's Edge

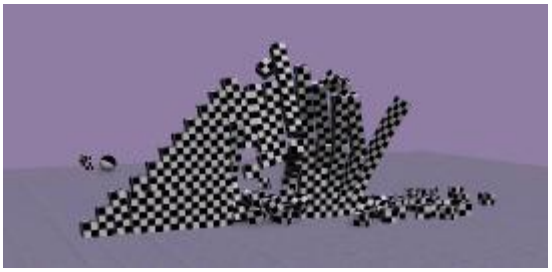
Physx a été le dernier "plugin" à développer. Il m'a fallu cinq semaines pour réaliser partiellement ce "plugin", une démonstration et des tutoriels. Je n'ai cependant pas pu finir d'intégrer entièrement cette librairie car elle contenait plus de 3000 fonctions. Je pense avoir fait un peu plus de 5% de l'intégration totale de Physx mais celà représente 30% des fonctions les plus utilisées pour développer une application avec Physx. On peut donc réaliser avec mon "plugin" actuel la plupart des phénomènes physiques.

4.3.1 1ère semaine : Analyse et développement basique

L'approche de ce "plugin" se rapproche de celui de FMOD. Mais est beaucoup plus gros et plus complexe que ce dernier. Je n'ai donc pas procédé de la même manière, l'analyse avant implementation de ce plugin a été moins approfondie, car si j'avais dû faire une analyse de chaque fonction comme pour FMOD je n'aurais pas eu le temps de passer à l'étape du développement. Le développement de ce "plugin" pour Stonetrip était une priorité, j'ai donc fait une analyse très rapide en regardant les exemples mis à disposition par Physx lors de son installation. Fort de mon expérience acquise au cours de mes deux premiers mois de travail sur les "plugins" j'arrivais à déterminer plus rapidement les fonctions problématiques et à trouver des solutions.

J'ai ensuite commencé l'implémentation du "plugin" de la même manière que FMOD, c'est à dire en suivant les exemples mis à disposition dans un ordre croissant de difficultés. J'ai réussi à obtenir assez rapidement le résultat le plus basique qui soit, c'est à dire un plan et

des cubes qui tombent dessus, mettant en oeuvre les collisions et la gravité. J'ai eu cependant un problème avec les rotations de mes cubes. Car Physx me renvoyait les rotations de ses objets via un quaternion, un outil très utilisé dans la 3D . En effet la relation qui existe entre les quaternions et les rotations en dimension 3 simplifie grandement certains problèmes de rotation rencontrés avec d'autres systèmes de données pour connaître la rotation d'un objet. Or dans ShiVa il m'était impossible de toucher à la rotation de mon cube par un quaternion, j'ai donc dû contourner le problème pour transformer mon quaternion en de simples rotations en degrés sur les 3 axes X, Y et Z.



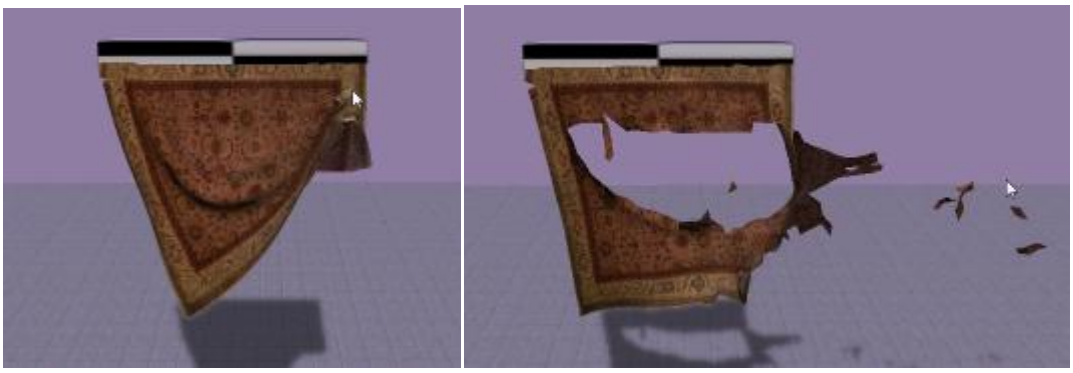
Pyramide de cubes qui s'écroule

La base interne de mon "plugin", c'est à dire la gestion des éléments de Physx que l'utilisateur ne voit pas, a été développée d'une manière plus poussée que FMOD car dans Physx un très grand nombre de données ne sont que temporaires, et donc l'utilisateur aurait dû dire dans son code LUA quand ses données ne sont plus utilisées pour pouvoir ainsi libérer la mémoire de l'ordinateur. Je me suis donc occupé en interne de libérer ces données en créant une liste des variables temporaires que l'utilisateur possède. Quand cette liste est pleine, la prochaine donnée temporaire à créer libérera et écrasera la plus ancienne donnée. Ainsi l'utilisateur n'a plus à se soucier de la gestion de la mémoire pour toutes les données temporaires.

J'ai ensuite continué le développement du "plugin" en ajoutant peu à peu les objets primaires utilisés dans la physique, sphères, capsules, géométries convexes et géométries quelconques.

4.3.2 2ème semaine : Développement du "clothing" et des "joints"

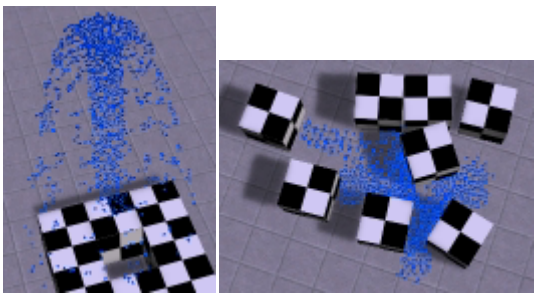
Physx offre la possibilité de faire du "clothing", mais d'une manière beaucoup plus poussée que sur mon "plugin" de "clothing". Par exemple il est possible de déchirer un tissu. La partie d'intégration du "clothing" m'a demandée beaucoup d'analyses pour qu'il soit possible de l'utiliser dans ShiVa, mais le résultat obtenu est très satisfaisant et même amusant quand on déchire un tissu à la souris ou en projetant un cube dessus.



J’ai ensuite implémenté la base des “joints”, les joints sont en physique des liaisons entre objets qui peuvent contraindre ceux-ci. Exemples : un “joint” de distance oblige deux objets à se trouver à une distance constante l’un de l’autre, un autre exemple est une contrainte d’axe, l’objet ne peut se déplacer que dans un seul axe.

4.3.3 3ème semaine : Développement des particules

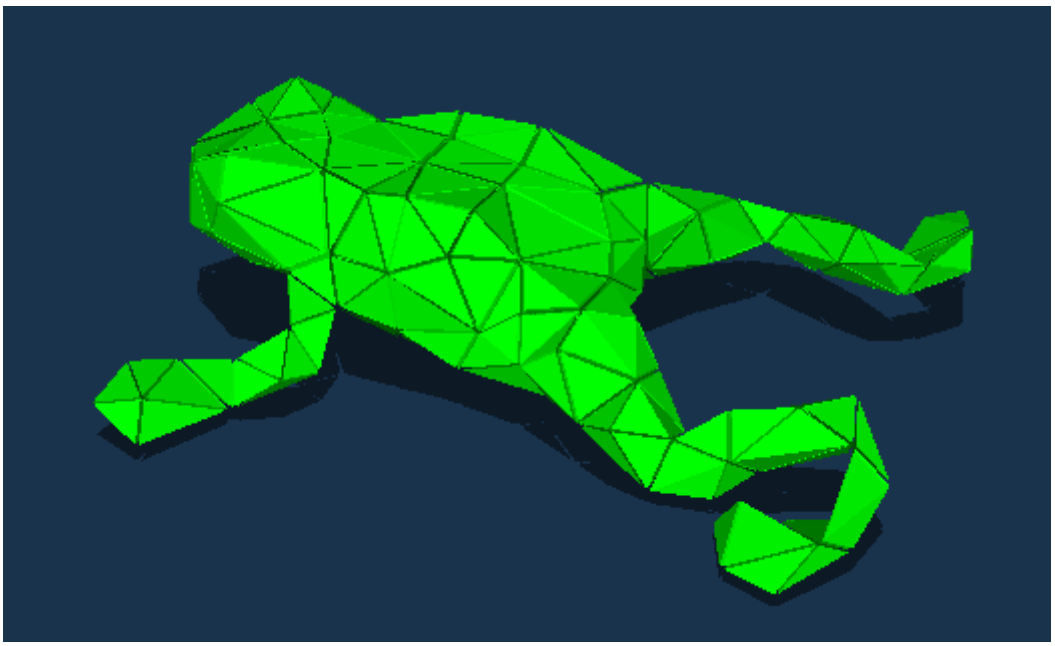
Je me suis ensuite attaqué aux particules, Physx permet de gérer via un système optimisé spécialement pour faire des effets de particules un très grand nombre d’objets (exemple d’utilisation : explosion qui projette des bouts de terre et de métal, fontaine qui projette de l’eau en l’air, etc). Dans un de leurs exemples Physx gère 8000 particules à la fois, en respectant des contraintes de collisions et de forces basiques, pour augmenter le réalisme. Les particules dans Physx peuvent aussi simuler le comportement des fluides, les particules se poussent entre elles comme une flaque qui s'étend sur le sol. Physx offre beaucoup de possibilités via les particules, j’ai donc passé beaucoup de temps sur cette partie.



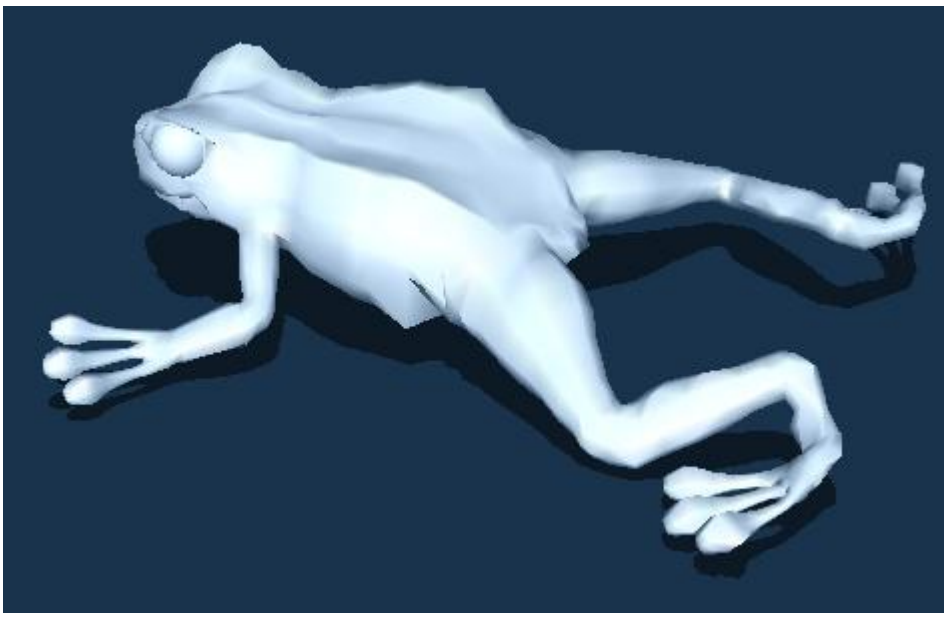
4.3.4 4ème semaine : Développement des “soft body”

Les “soft body” sont en fait des modèles 3D complexes comme une grenouille, mais qui sont mous, c’est un peu comme le “clothing” mais ils ont en plus une contrainte de volume. Le corps peut être déformé mais il revient généralement à son aspect d’origine. Quand on secoue notre grenouille c’est surtout les parties les moins solides, comme les pattes qui se déforment. J’ai eu beaucoup de mal à implémenter les “soft body” car c'était beaucoup plus

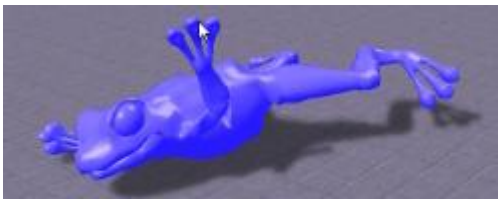
complexe que le “clothing”. En effet Physx ne déforme pas directement notre grenouille mais une version bien plus basique, pour optimiser les calculs. C’est à nous de nous débrouiller pour que notre vraie grenouille se déforme en suivant la version simplifiée. Mais le résultat en valait la chandelle car j’ai pu m’amuser à déformer ma grenouille.



Version simplifiée de la grenouille que Physx modifie



La vraie grenouille



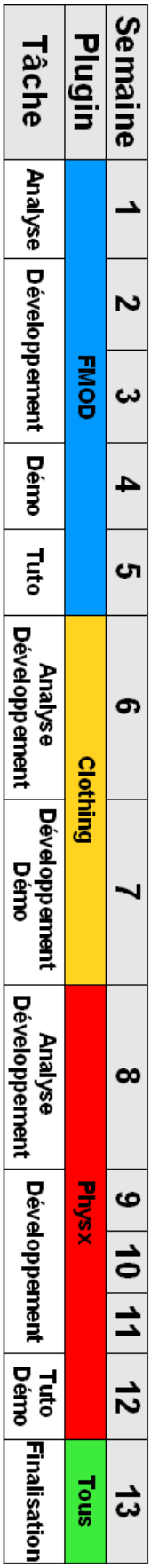
Rendu dans « ShiVa » de la grenouille en soft body

4.3.5 5ème semaine : Documentation, tutoriels et démonstration

Lors de la dernière semaine j'ai écrit en anglais une documentation sur mon "plugin" ainsi que 5 tutoriels pour que les utilisateurs puissent apprendre à se servir de Physx dans leurs jeux sous ShiVa. J'ai aussi fait une démonstration du "plugin" en mettant en place un tissu qui pend, et qui se fait déchirer par un cube qui tourne autour d'un axe via les "joints". Une grenouille en "soft body" est aussi posée de manière à ce que le cube l'écrase et la projette plus loin. L'utilisateur peut aussi attraper la grenouille et le tissu avec la souris. Il peut lancer des cubes, des sphères et des géométries aléatoires convexes.

Voilà le résumé de mon travail, le dernier jour il m'a été demandé de produire des versions exécutables de mes démonstrations des trois "plugins".

5 Planning du stage



6 Conclusion

Lors de ce stage j'ai pu tester mes compétences et même les améliorer comme en programmation C/C++, en anglais pour toutes les documentations que j'ai utilisées et celles que j'ai écrites, en communication, en analyse, etc... J'ai aussi appris énormément de chose dans cette entreprise. Les bases de la technique 3D, du son, du "clothing", d'un moteur physique, le langage LUA, l'utilisation de ShiVa, etc...

Je pense avoir fait un travail important pour Stonetrip et avoir rempli les objectifs de ce stage de façon professionnelle. J'ai su faire preuve d'autonomie, sans pour autant rester être solitaire, je prenais le plus possible en compte les conseils et avis de mon maître de stage, pour répondre au mieux aux besoins de l'entreprise. J'ai pu apprendre sur moi-même, et découvrir que j'étais capable de comprendre rapidement des choses dont je ne connaissais rien jusque là.

Au niveau critique de la formation que j'ai suivie au sein de Polytech'Nice Sophia ces deux dernières années, je pense que divers cours m'ont beaucoup aidés lors de mon travail tels que les cours de programmation, de communication, d'architecture logiciel, d'anglais, etc... Mais je pense que ce que l'école m'a le plus appris au cours de ses nombreux TP et projets, c'est à réussir à apprendre par moi même, à trouver les documents et outils nécessaires à la réalisation de mon travail et être capable de les assimiler. Le seul reproche que j'aurais à faire au niveau des cours de l'école, serait peut être de ne pas offrir de cours utilisant des librairies graphiques 3D dans la section logiciel que j'ai suivie.

Du point de vue de l'entreprise qui m'a accueilli, et des membres de l'équipe qui la composent, je pense qu'ils ont apprécié mes capacités d'adaptation au projet, mon investissement et ma ténacité à la tâche. En effet, M. Belhassen m'a proposé d'intégrer son équipe en poursuivant ma formation en alternance et/ou mon stage de 5ème année dans son entreprise.

7 Bibliographie

7.1 Rapport

- Pour les définitions : Wikipedia l'encyclopédie
- Pour les images : ShiVa + plugin, démonstration officielle de Physx

7.2 Stage

- Documentation : FMOD, Physx
- http://www.cplusplus.com/

8 Abstract

As part of my 4th year engineering course at Sophia Polytech'Nice in the Computer Science department of Sophia-Antipolis, I had to do an internship for a period of thirteen weeks in a company, from June 21 to September 21 2010, to gain work experience.

The company that welcomed me, is called “Stonetrip”. Overall they make a software named "Shiva", which is mainly used to produce quality 3D video games, quickly, and on many platforms in the market.

The work entrusted to me was to implement / extend the software "Shiva" by adding new features, mainly the integration of libraries available on the market, offering services such as management of sounds, or management of physical phenomena.

The programming language used in the proposed project is C / C + +. And I used the software "Shiva" to verify the proper functioning of the features I implemented.