'*SCP Camera*'
Shoot, Cut & Play

X.Gouchet ; R. Quittard ; N.Serikoff
Departement Arts et Technologies des Nouvelles Image
Master 1 - Université Paris VIII

http ://scp.camera.free.fr

Chapitre 1

Presentation du projet

1.1 Principe du projet

Notre projet trouve son origine à la fin du mois d'avril 2005, suite au festival de réalité virtuelle Laval Virtual. Le but était de réaliser une application permettant à des etudiants, cadreurs ou réalisatuers de prototyper de façon simple et rapide des cadrages et angles de vue.

En effet, grâce à l'outil 3D, nous avons la possibilité de se déplacer dans des scènes virtuelles et de manipuler facilement l'environnement (décors, lumières, acteurs).

Ce projet était pour nous un moyen de mettre nos connaissances au service d'un domaine que nous apprécions : le cinéma et la réalisation.

Partant de cette idée, le projet a subit plusieurs modifications au fur et à mesure de sa réalisation.

Au départ, il était question d'enregistrer les mouvements de caméra de l'utilisateur pour le réutiliser dans un logiciel 3D (Maya). Puis, nous avons compris qu'il serait plus utile et plus novateur de créer un logiciel de montage complet, permettant d'éditer les mouvements enregistrés.

Puis nous avons eu l'idée de créer des figurants (foule pseudo-dynamique), ce qui rajoute de la vie dans la scène.

Au final, notre application permet d'évoluer dans une scène 3D, soit au moyen d'un capteur de positions et d'orientations, soit en utilisant le clavier et la souris, et de filmer une scène virtuelle comme avec une vraie camera. L'aspect novateur du projet est, qu'au lieu d'enregistrer les images issues des mouvements créés, nous enregistrons directement les mouvements, pour pouvoir les éditer et les réutiliser ensuite dans d'autres logiciels (Maya, Virtools,...), et pouvoir sortir une vidéo de qualité.

1.2 Présentation en Detail

Cette installation est composée de l'application Virtools, du plug d'export Maya et d'une caméra servant d'interface, munie d'un écran LCD et d'un capteur de mouvements.

Materiel

Du point de vue matériel, nous avons développé une caméra factice, utilisable sur notre application pour ajouter une dimension immersive au projet.

Cette caméra est une ancienne caméra VHS, dont nous avons retiré les composants éléctroniques qui permettaient l'enregistrement de vidéo, pour intégrer notre propre interface.

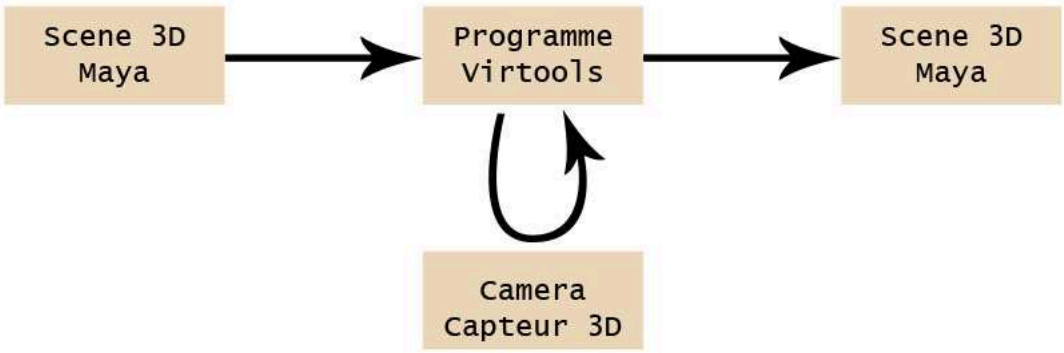

FIG. 1.1 – Principe de l'Installation

Grâce à un cable USB, nous pouvons récupérer la pression sur les boutons de la caméra, ainsi que l'utilisation de la bague de l'optique (que nous utilisons pour la profondeur de champ).

Ensuite, nous avons intégré à cette caméra un écran LCD (récupéré sur un lecteur DVD portable), afin d'avoir un retour visuel directe de l'interaction entre l'utilisateur et l'application.

La caméra est reliée à un boîtier via un câble parallèle (ou un câble RJ45). Le boîtier permet de récupérer les informations video et audio pour les transmettre à la caméra, et transmet l'utilisation des boutons de la caméra à l'ordinateur.

Enfin, un capteur de position peut être placé sur la caméra, indépendamment des câbles utilisés pour l'écran et les boutons de la caméra. Cela permet d'avoir un matériel très portable, pouvant être aussi bien utilisé avec un capteur magnétique (Polhemus par exemple) ou un capteur optique (infrarouge, Cyclope par exemple).

Application Virtools

Le programme virtools s'articule autour des trois fonctions de notre projet : Shoot, Cut and Play (Enregister, monter et rejouer).

Dans le programme Virtools, chaque mode est compartimenté et distinct des deux autres. Cela offre un moyen d'orienter l'utilisateur dans l'étapes de la réalisation d'un projet.

Utilisation

La première partie du programme est le mode Shoot [F2], permettant de se déplacer librement dans la scène et d'enregistrer le mouvement de caméra (touche [R] ou bouton enregistrer).

Le déplacement peut se faire avec la souris (pour l'orientation) et le clavier (touches [Z][Q][S][D]), ou un capteur de positions / orientations (capteur magnétique ou infrarouge).

Chaque enregistrement apparait dans la timeline située en bas de l'écran. Ces clips peuvent ensuite être modifiés dans le mode Cut [F3].

Nous avons quatre outils de montage. Tout d'abord le Cut, qui permet de couper un clip en deux. Cela permet, par exemple de ne garder que la partie qui nous intéresse et supprimer l'autre, ou de placer un autre clip entre les deux parties coupées.

L'outil Crop permet de modifier les bornes de lecture d'un clip, afin de ne garder que l'intervalle utile.

L'outil Drag and Drop (glisser-deposer) permet de changer l'ordre de lecture des clips, en les déplaçant simplement à l'endroit désiré sur la timeline.

Enfin l'outil Delete permet de supprimer un clip.

Enfin avec le mode Play [F4], on peut relire le montage effectué.

Ajoutés à ces options de base, il est possible de sauver, exporter ou ouvrir un montage. Une fois exporté, un montage peut être utilisé dans Maya, où le mouvement de caméra pourra toujours être modifié à loisir.

Import Maya

La derniere partie de notre projet permet d'utiliser dans Maya des mouvements enregistrés dans Virtools.

Si la scène comporte des animations utilisées comme références dans virtools, les mouvements importés peuvent être synchronisés pour obtenir un résultat similaire à celui créé dans Virtools.

En outre, le plug permet de créer le fichier batch, automatisant le rendu des mouvements créés.

Chapitre 2

Notre travail

2.1 Présentation de l'équipe

Nous avons été une petite équipe de trois personnes à travailler sur le projet SCPCamera.

Remi Quittard
Xavier Gouchet
Nicolas Serikoff

De part, les exigences du projet, nous avons naturellement constitué un groupe aux compétences variées touchant à des domaines très différents (éléctronique, programmation, graphisme) mais, dans notre cas, complémentaires. Ainsi, au vue des parcours et des spécialisations de chacun, la répartition des rôles et des tâches s'est faite facilement.

Rémi Quittard, avec un bac STI (Sciences et Techniques Industrielles) et passionné d'éléctronique, a travaillé sur toute la partie très technique du projet qui a consisté à la mise au point du système de la caméra factice. Il a aussi réalisé les animations présentent dans la scène 3D et grâce à ses connaissances en flash, le site web du projet.

Xavier Gouchet avec une licence d'Informatique a travaillé sur toute la partie programmation qui comprend aussi bien le noyau principale de l'application fait sous Virtools que des scripts Mel ou du développement réseau.

Nicolas Serikoff, avec une licence d'Arts Plastiques et un profil artistique, a travaillé sur toute la partie graphique et principalement sur la partie 3D, qui comprend la modélisation et les textures d'une scène 3D optimisée pour le temps-réel et le pré-calculé.

L'effectif réduit de notre équipe et la synergie de travail qui s'est installée naturellement ne nous a pas poussé à avoir ou à établir de chef de projet. Les relations de travail ont toujours été de type horizontale sans aucune notion de hiérarchie. Les grandes décisions et les choix techniques ou artistiques ont toujours été prises d'un commun accord après argumentation de chacun.

Il est donc important de souligner que la réussite de ce projet tient au travail d'équipe fourni par chacun, à la mise à disposition et aux partages de nos savoir-faire. Le rôle de chacun a été cruciale pour l'aboutissement du projet qui aurait été compromis sans l'un de nous.

Notre formation universitaire à A.T.I s'est efforcée de nous donner les clefs indispensables pour comprendre les contraintes et le travail de l'autre dans l'univers dans lequel nous évoluons tout les jours : le numérique. Quel que soit notre profil (programmeur ou graphiste) nous avons tous une base commune de connaissances qui nous permet de mieux appréhender le travail en équipe.
Aussi la réussite de ce projet aux compétences variées en est sans doute une excellente illustration.

2.2 Déroulement du projet

Le projet s'est déroulé en trois phases ou étapes disctinctes qui en caractérise l'évolution.

Tout d'abord nous avons commencé par une maquette débutée en octobre et qui s'est achevée fin mars. Il s'agit à la fois de la phase de préparation et de recherche la plus en amont du projet.

Bien que très difficilement quantifiable en terme de travail, ceci nous a permis, à travers de nombreuses réunions de travail de discuter et de poser les grandes lignes du projet.

Le rôle de chacun et la répartition des tâches a principalement été décidé à ce moment.

La maquette

Le capteur

La première chose dont nous nous soyons occupé a été de répondre à la problématique du capteur de position.

Intégrer ce genre de technologie dans le projet a été un élément motivant et représente un des aspects innovants de SCPCamera.

Mais cela a soulevé de nombreuses inquiètudes tout au long de la réalisation.

Grâces à nos recherches nous avons pu avoir un aperçu des capteurs qui correspondaient à notre volonté de récupérer la position et la rotation d'un objet dans l'espace.

Optique ? Nous avons rapidement éliminé l'idée d'utiliser un système optique qui consiste à placer plusieurs caméras et à retrouver par algorithme la position de l'objet.

Un des principales intérêts de ce système réside dans son faible coût car il peut fonctionner avec des webcams, mais il demande un développement important notamment pour traiter et analyser le flux vidéo (bien que l'on puisse trouver aujourd'hui des librairies libres de droits facilitant le travail).

Mais l'inconvénient principale est sans doute le fait que ce genre de système est très sensible aux conditions de lumière.

Un des leitmotiv du projet a toujours été que le système soit le plus facilement utilisable partout, avec le moins d'aléas possible.

Il nous aurait fallu alors développer un studio de tournage spécifique au projet, avec des conditions de lumière toujours identiques : chose nullement envisageable ni réalisable.

Nous avons eu par la suite confirmation de ces contraites lors du festival de Laval Virtual 2006 où des équipes ayant intégrées ce type de technologie se retrouvaient à qualibrer constament leur matériel selon les conditions d'éclairage.

Magnétique ? Nous nous sommes fixés sur les capteurs magnétiques de positions de type Polhemus ou Fastrack/ Patriot pour deux raisons.

La première est que ces systèmes permettent d'obtenir excatement les données que nous cherchions à avoir : positions et rotations, c'est à dire 6 axes de liberté.

La seconde est que nous disposions d'un système Polhemus Flock Of Bird à l'université ce qui évacuait, ainsi la question de l'achat du système (environs 3500 E).

Toutefois ce genre de dispositif impose quand même quelques contraintes.

Bien que les dispositisf magnétiques soient sensibles aux ondes et aux éléments magnétiques (barres et charpente métaliques), nous n'avons pas constaté de perturbation de ce genre avec notre matériel.

Le rayon d'action ou de liberté est théoriquement limité par un câble qui relie le capteur (en mouvement) à la borne (fixe) ce qui fait un diamètre avoisinant les 8 mètres.

Mais lors des testes que nous avons effectués pour la maquette, nous avons constaté une longueur de déplacement maximale du capteur de l'ordre de 1 mètre avant de perdre tout signal correct.

A ce problème de distance, c'est ajouté un problème d'intégration dans Virtools. L'utilisation du Polhemus dans une scène Virtools faisait automatiquement tomber les fps à 10 ou 14. Autrement dit, inutilisable en tant que tel, dans une scène temps-réel.

Scène 3D

Pour pouvoir montrer au mieux l'utilisation du projet, nous avons pensé à créer notre propre environnement 3D dans lequel l'utlisateur évoluerait.

L'idée Il a été retenu le choix d'une scène de rue réaliste car cet environnment nous permettait de rajouter facilement des scènes d'actions qui serviraient de matière à filmer.

Le but était d'avoir une scène plus démonstrative qu'originale.

L'idée d'un système de foule a été abordé et retenu pour donner un aspect plus vivant et plus dynamique.

Le principe d'avoir des acteurs princpaux effectuants des actions personnalisées a aussi été adopté. Ces acteurs sont les centres d'intérêts de l'action.

Répérages photos En parallèle des recherches sur les capteurs, nous avons commencé un travail de repérages de lieux qui nous semblaient pertinants à reconstituer et à placer dans note scène 3D.

L'idée est ainsi de s'inspirer d'éléments réels, et de nous servir de ces photos comme bases de textures.

Munis d'un appareil numérique compact, nous avons pris près de 250 photos de facades, d'immeubles et d'espaces urbains qui nous ont servi à la conception de notre rue.

Chaque élément a été pris plusieurs fois pour s'assurer le choix d'une photos référence de qualité.

Les Choix techniques Nous avons déterminé lors de cette maquette les principaux choix techniques en matière de 3D.

Une passe d'occlusion baké servirait pour les ombres de contact.

Le fait de baker cette passe nous permettait à la fois de gagner du temps de calcul pour la scène pré-calculée mais aussi d'améliorer la qualité graphique et le réalisme pour la version temps-réel.

Les personnages seraient animés à partir de bibliothèques de motion captures. Ceci permet notamment dans le cas des animations secondaires de la foule d'avoir un résultat rapide (cycle de marche court) et dans les animations principales d'avoir des mouvements très précis et réalistes.

Sous Virtools Le principe de base d'enregistrement des mouvements de la caméra dans Virtools est défini.
La notion de timeline, pourtant absente en temps-réel est crée.
Le système d'export vers maya en fichier texte aussi.

La premiere version

C'est dans un laps de temps très court (15 jours) que nous avons réalisée cette première version.

C'est grâce à tout le travail préalable de réflexion et au travail préliminaire que chacun avait fait dans son domaine que nous avons pu tenir les délais imposés par le festival de Laval Virtual 2006.

Rémi a travaillé exclusivement sur la réalisation de la caméra et de notre propre interface éléctronique.

Toutes les principales fonctions nécessaires à notre applications(enregistrement, lecture, profondeur de champs) ont été reliées par éléctronique aux boutons déjà existants de la caméra VHS récupérée. *(cf Fig. 3.18 p.24)*

Xavier a approfondi le système de montage en rajoutant des options comme l'outil "couper" ou "déplacer".

Il a développé aussi un système client/serveur pour faire fonctionner le Polhemus sur une seconde machine et permettre ainsi de ne pas ralentir Virtools.

Nicolas a modélisé et texturé l'ensemble de la scène 3D et intégré le shaders HLSL de profondeur de champs à la scène virtools.

Laval Virtual

C'est lors du festival de réalité Virtuelle de Laval Virtual 2006 que nous avons fait nos premiers tests et notre première présentation au public.

Pour nous, ce salon représentait l'aboutissement du projet. Des retours qu'on aurait sur le salon, dépendaient la poursuite du projet.

Deux points positifs sont à noter à propos de ce salon.

Tout d'abord l'aide que nous avons pu avoir sur place de la part de la société Virtools et Immersion (spécialisée dans les capteurs 3D).

David Nahon, ancien de A.T.I et désormais en charge de la réalité virtuelle chez Virtools, nous a mis en relation avec la société Immersion présente sur le salon après avoir vu nos problèmes liés au capteur Polhemus.

Immersion nous a gracieusement prêter un capteur optique qui est une caméra infrarouge (prototype caméra Cyclope) permettant de récupérer la position et la rotation d'un objet dans l'espace.

A partir de là, nous avons pu faire fonctionner notre caméra en intégrant très facilement (20 minutes en tout) le dispositif d'Immersion avec notre scène 3D virtools grâce au pack VR de Virtools.

Malgré les défauts de l'application (pas d'animation, pas d'interface) mais avec un système immersif opérationnel, nous avons pu rencontrer des professionnels qui ont vu dans notre projet une réponse pertinante à des problématiques qu'ils pouvaient rencontrer dans leurs domaines (audiovisuel principalement).

Le Futuroscope s'est dit intéréssé par une orientation de notre projet pour les scolaires.

Le Siggraph nous a encouragé à écrire pour l'édition 2007.

Nous restons en contacte avec Immersion en espérant développer un partenariat plus approfondi autour de notre application et de leur caméra cyclope.

Ce salon de Laval a été pour nous l'occasion de faire un premier état des lieux du projet après plusieurs mois dessus.

Des failles sur plusieurs aspects ont été mises à nues tandis que les encouragements et les remarques nous ont redonné la motivation et la volonté de pousser le projet.

Nous avons pris conscience de l'intérêt d'aller jusqu'à une version réellement plus aboutie.

La version Finale

Cette version finale a été l'occasion de se nourrir des critiques et des défauts présents dans la version de Laval.

Le but de cette version n'a pas été de simplement corriger des bugs mais bel et bien d'avoir une application utilisable par tous.

La prise de contact avec le Futuroscope nous a poussée à repenser de manière plus large le public visé par notre application.

Nous avons décidé de ne pas se restreindre aux seuls professionnels et ainsi de mettre au point une version ludo-éducative permettant à tous, grâce à un didacticiel, de se familiariser avec le maniement d'une caméra.

Le web

Pour gagner en visibilité et faciliter la présentation et la communication autour du projet, nous avons décidé de créer un site web, réalisé par Rémi.

Nous voyons plusieurs intérêts majeurs à posséder un site web :
- communiquer plus facilement sur le projet
- accéder à tout moment et de n'importe où, à du contenu (descriptifs, images et vidéos)
- appuyer la crédibilité du projet en montrant que c'est un projet vivant qui évolue.
- montrer la multiplicité de nos compétences.

L'interface

En parallèle, nous avons créée une véritable interface d'utilisation :
- 4 menus dont celui des options
- la TimeLine, barre du temps, sur laquelle s'effectue toute les opérations de montage.

Cette phase a été une période de travail collaboratif et d'intégration importante entre le graphiste (Nicolas) et le développeur (Xavier).

L'ensemble des différentes fonctions a été redébattue pour s'assurer de leurs sens et de leurs utilités.

Le code couleur a été adopté pour permettre de plus facilement identifier les 3 modes d'utilisations :
- Shoot, enregistrement =<bleue
- Cut, montage =<orangé
- Play, prévisualisation =<vert

De la clareté de l'interface dépend la lisibilité et l'accroche de l'utilisateur.

Améliorations graphiques

En termes, de modélisation relativement peu de changement ont été apporté.

La nouveauté vient plutôt des changement du pipeline de production.

Nicolas a amélioré le rendu pré-calculé, retravaillant l'ensemble des textures pour les adapter en deux formats :
1 fichier optmisé pour le temps-réel avec des tailles réduite de textures (taille max 128*128)
1 fichier de meilleur qualité pour la scène précalculée (taille max 1024*1024)

Un système de "Resolution Node" permet de permutter l'utilisation et l'affichage entre les textures basse et haute résolution.

La technique du BumpMapping a été implémenté dans les rendus maya.

Aussi pour permettre d'utiliser toujours une seule et même scène 3D, Xavier a dû crée des outils (scripts Mel) facilitant l'export de la scène maya vers Virtools en tenant compte d'optimisations pour le temps-réel.

Rémi a réalisé, sous MotionBuilder et à partir de motion capture, l'action qui sert de scène à filmer.

Une partie de l'animation a été réalisée à l'aide du logiciel endorphine afin de rajouter un aspect naturelle et dynamique.

Pour rajouter un aspect plus vivant, une simulation de foule a été ajouté. Il s'agit d'acteurs secondaires qui suivent une multitudes de courbes.

Le didacticiel

Afin de répondre à la demande du Futuroscope, nous avons ajouté un mode didacticiel

Il s'agit simplement de présenter les différentes valeur de cadrages, la notion de champs/contre-champs et de plongée /contre-plongée.

Pour cela, nous expliquons l'effet à l'aide d'une illustration puis l'utilisateur doit essayer de refaire l'effet.

Pour les enfants c'est un moyen d'appréhender de manière simple des notions parfois complexes et de rentrer dans la peau des grands réalisateurs.

Ce mode permet d'ouvrir notre application a un public qui n'est pas forcément familier avec l'univers de l'audiovisuel ou du cinéma.

Le code

L'ensemble du projet Virtools et du plug-in Maya a été refait de A à Z pour plusieurs raison.

La première est que la version faite pour Laval a été programmée en 15 jours. De fait, le code était souvent non optimisé, non commenté, et surtout difficile à modifier.

Ensuite, Xavier voulais reprendre le coeur du programme afin d'avoir un code propre et optimisé. Ayant deja codé l'essentiel de l'application une première fois, les besoins étaient clairement definis et plus facile à mettre en place.

Enfin, certaines fonctionnalités (notamment le drag'n drop) présente dans la version précédente était bancale et necessitait d'avoir une structure des données differentes.

Chapitre 3

Le rôle de Remi

Sur ce projet, le rôle de Rémi fut varié puisqu'il eu à s'occuper de la modélisation et de l'animation des personnages, de toute la partie électronique, ainsi que de la création du site internet en Flash.

3.1 Création des personnages

Les personnages humains

C'est juste après avoir réalisé la premiere maquette qu'a débuté la modélisation du personnage temps réel.

C'était la première fois que Rémi s'essayait à la modélisation d'un être humain « lowpoly », il a donc profité de cette execrcice pour augmenter ses compétences dans ce domaine.

Ici le but n'était pas de faire un personnage ultra réaliste, mais plutot un modèle de base passe-partout, facilement animable et peu coûteux en polygones.

Nous avons procédé de façon très classique en partant de primitives simples, cubes, sphères et cylindres. Puis nous avons donné les volumes générale des différentes partie du corps en subdivisant et en plaçant les Vertex *(cf Fig. 3.1 p.11)* .

Une fois les bases de la morphologie fixées, les détails du visage et des mains sont venus se rajouter *(cf Fig. 3.2 p.11)* .

Notez qu'afin d'avoir la possiblilité de générer facilement un grand nombre de figurants virtuels, plusieurs petits accessoires de personalisation ont été modélisés séparéments, et la texture est faite de telle manière que plusieurs coloris de vêtements puissent être créés dynamiquement dans Virtools. Ainsi les passants peuvent arborer différentes barbes, coupes de cheveux, au choix beret et/ou baguette de pain *(cf Fig. 3.3 p.12)* .

En ce qui concerne la texture, nous avons choisi de seulement mettre une map d'AmbientOcclusion pour ajouté du réalisme. Cette texture fut placée sur un canal texture supplémentaire et fusionné en mode « Multiplication ».

De cette façon le personnage avait une couche d'AO (appelé aussi Dirtmap) indépendante des couleurs des matériaux de peau et de vêtements. Cela permettant d'avoir des personnages aux tenues differentes *(cf Fig. 3.4 p.12)* .

Pour la préparation à l'animation (rigging et skinning) nous avons utilisé comme modèle le squelette Mbskel de MotionBuilder. En effet, la hierarchie et la convention de nomage permettent une reconnaissance direct de l'ossature dans Motion Builder.

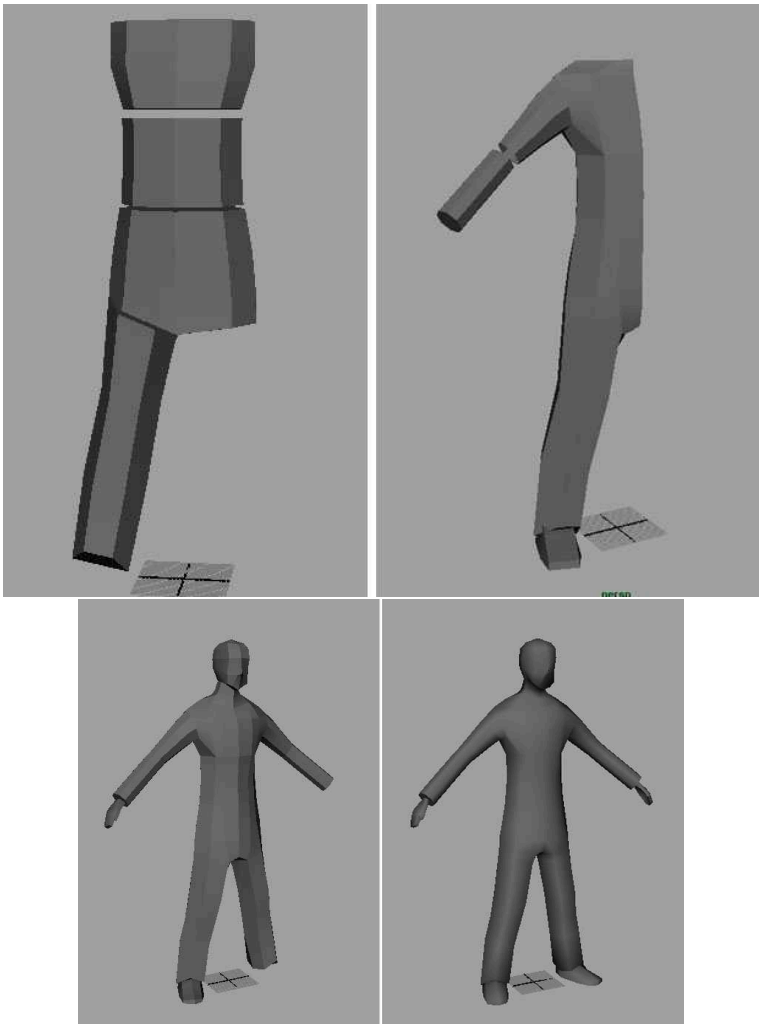

FIG. 3.1 – Evolution de la modelisation du corps

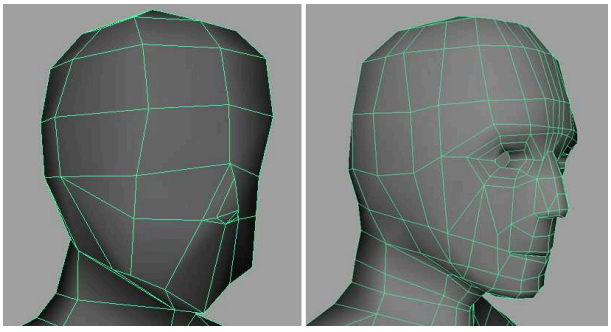

FIG. 3.2 – Evolution de la modelisation du visage

FIG. 3.3 – Accessoires de personalisation des personnages

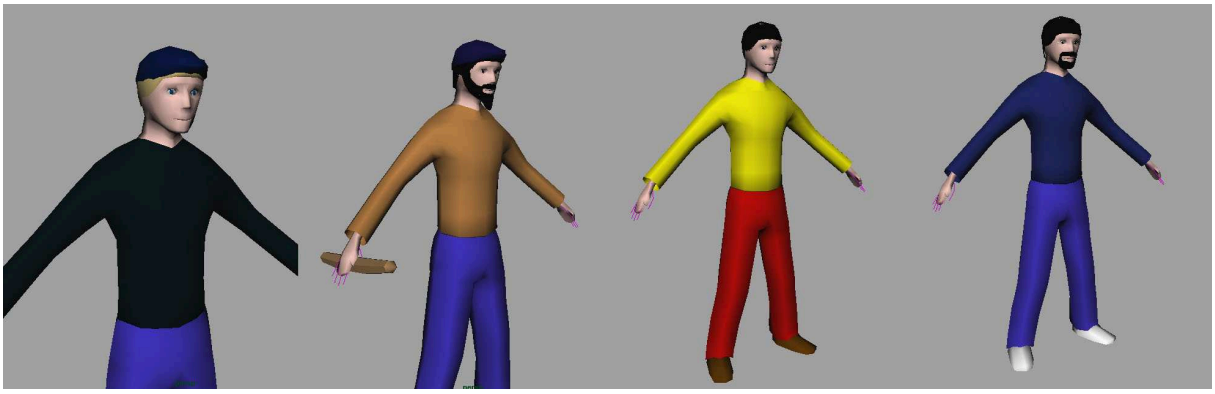

FIG. 3.4 – Variation de couleurs de personnages

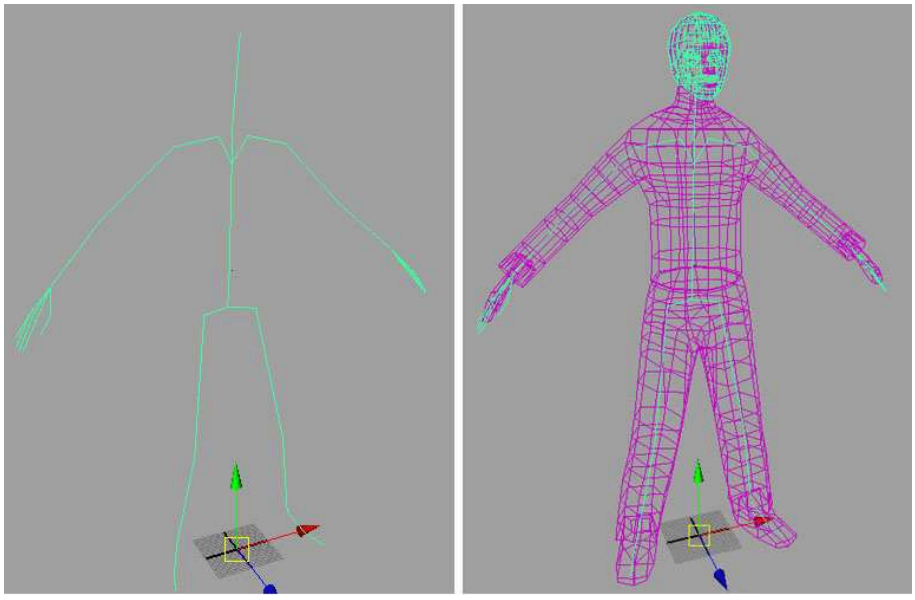

FIG. 3.5 – Phase de rigging

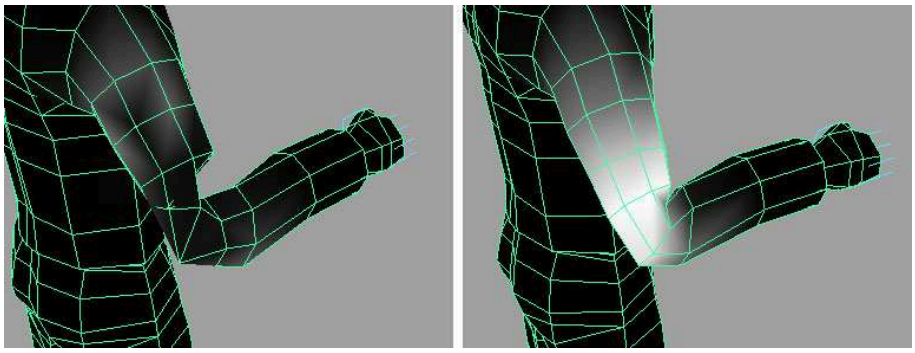

FIG. 3.6 – Phase de skinning

Le tyranosaure

Le dinosaure a nécessité moins de travail car il a été repris d'une scène exemple de Motion Builder. Nous lui avons apporté des modification de maillage (optimisation), une texture d'Ambient Occlusion, et un skinning entièrement refait *(cf Fig. 3.7 p.14)* .

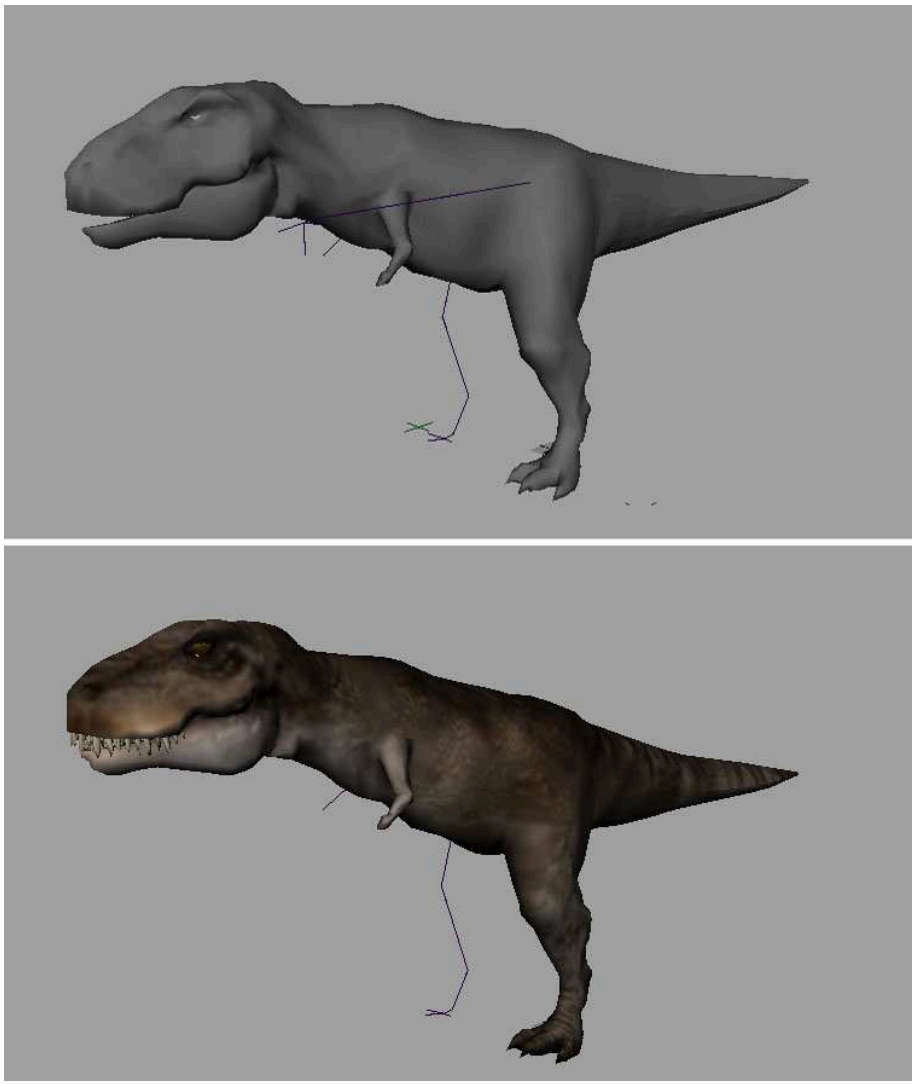

FIG. 3.7 – Modelisation du Tyranosaure

3.2 Animation

De toutes les parties relatives aux personnages, celle-ci fût la plus longue et complexe.

Pour animer, nous avons choisi d'utiliser principalement deux logiciels : Motion Builder qui est une référence dans le domaine, et Endorphine qui permet de faire de l'animation dynamique et comportementale.

Pour la grande partie des anims notre choix s'est porté vers la Motion Capture (capture de mouvements). Cela pour deux raisons, premièrement le souhait de pratiquer concrétement cette technique qui est de plus en plus utilisé dans la production de films et jeux vidéos, et deuxièment Rémi n'étant pas spécialisé en animation, il lui aurait été plus difficile de créer des animations de qualité entièrement en keyframing.

Recherche

Une fois décidée l'utilisation de la « Mocap », la première tâche fût de trouver des fichiers de capture de mouvements libres de droit et surtout exploitables. Par exploitable, il faut comprendre que les mouvements doivent être assez proches de ce que l'on veut pour notre personnage. De même, il est nécessaire que la carrure et la démarche de l'acteur ayant servi de modèle pour la capture se rapproche du perso 3D.

Par exemple, une marche capturée sur une femme d'un mètre soixante sera difficilement transférable sur le modèle 3D d'un homme d'un mètre quatre vingt dix (la démarche féminine lui ferait perdre toute crédibilité).

Transfert et correction des mouvements

Une fois les animations trouvées, on procède à un « retargeting » (adaptation) permettant à MotionBuilder de reconnaître le squelette importé et d'y appliquer un « character ». Ainsi on peu transférer la Mocap sur notre perso *(cf Fig. 3.8 p.15)* .

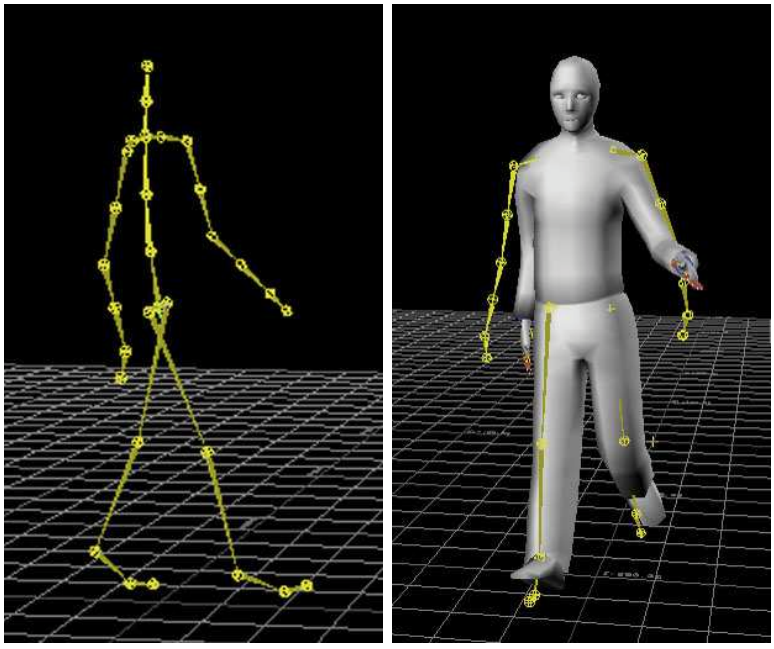

FIG. 3.8 – Transfert d'une motion capture sur un personnage

Lors du transfert on remarque très souvent des problèmes de décalages des bras et des jambes dus à une différence de position de référence entre la mocap et notre personnage.

Afin de résoudre ce défaut, il faut « Ploter » l'animation sur un control Rig, une sorte de manipulateur gérant le squelette et autorisant la cinématique inverse. Pour information, le « Plot » est une procédure qui ajoute une clé d'animation par frame sur chaque objet animé *(cf Fig. 3.9 p.16)* .

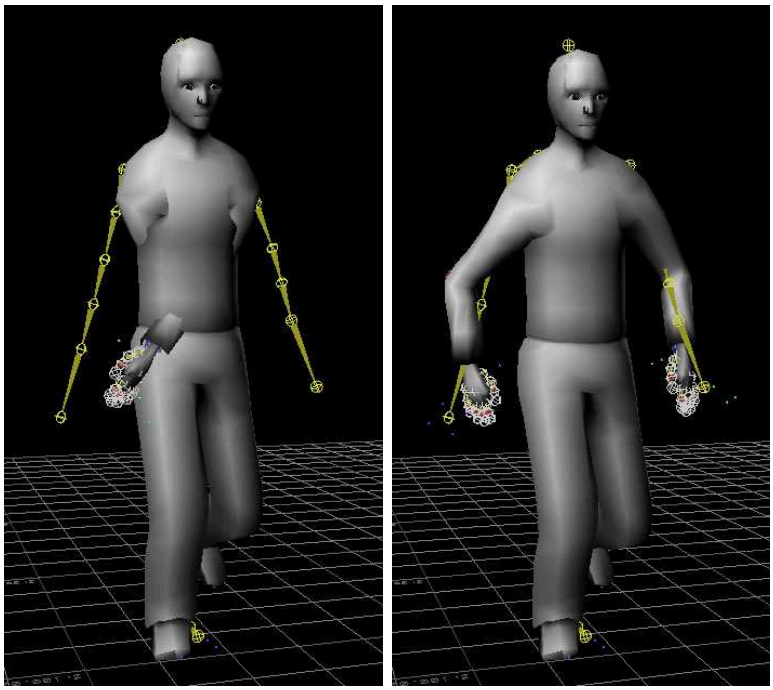

FIG. 3.9 – Correction d'un bug lors du transfert

Une fois ceci effectué, on ajoute de nouvelles couches d'animation (Layers) pour chaque élément du corps problématique.

Précisons que pour le dynosaure un gros travail supplémentaire de « déshumanisation » fut nécéssaire. En effet, ses animations principales venant d'un humain il en avait les mimiques, et il a donc fallut lui transmettre une impression de lourdeur et de grandeur.

Transformation en cycles

Il a fallut ensuite convertir les animations de marche et de courses en cycles bouclant sur eux-mêmes à l'infini.

Pour cela nous avons utilisé les fonctions de mélangeur d'anim (Motion Blend) de Motion Builder. Ici l'important était de bien choisir des positions de départ et d'arrivée suffisament similaires dans le but d'éviter des interpolation de mouvements trop brutales *(cf Fig. 3.10 p.18)* .

Au final six capture de mouvements ont été utilisé sur le projet.
Une fois toutes les animation nécessaires créées, on les sauve avec les personnage dans un même fichier pour procéder à la mise en scène.

La mise en scène

Pour appuyer notre projet et tirer pleinement partit des fonctions de montage nous souhaitions avoir une scène d'action.

Ainsi nous nous sommes fixé sur le scénario suivant : Un Tyranosaurus Rex arrive dans la ville et dévore un passant.

La premiere étape fut de définir les trajectoires, et les actions pour chacun des acteurs *(cf Fig. 3.11 p.18)* .

Dans ce travail de metteur en scène virtuel, nous avons du tenir compte des rythmes de marches des personnage afin d'éviter les effets de flottement.

De même, l'environnement 3D étant restreint il fallut bien choisir les lieux d'entrée en scène et de sortie des personnages.

Le gros de l'animation a été l'enchaînement des différents cycles de marche et de course, ainsi que le keyframing du parcours des trajectoires.

Une des difficultés dans l'enchaîenement est de trouver les bons points d'entrée/sortie pour le mélange d'animation (Blending), sans quoi des interpolations hazardeuses auront lieux.

Une premiere étape a consisté à fixer grossièrement le début de l'action, de l'arrivé du dinosaure jusqu'à la poursuite. Ensuite, nous nous sommes occupé du moment critique où le T-rex attaque la victime et la dévore. Nous avons donc préparé les mouvements du Trex (dans sa course il se penche et tourne la tête) avant de les exporter vers Endorphine. Dans Endorphine nous avons créé l'animation dynamique en essayant plusieurs réglages, contraintes et comportement *(cf Fig. 3.12 p.19)* .

Une fois le résultat satisfaisant, la scène a été réexportée vers Motion Builder.

Les animations venant d'Endorphine sont traitées comme une Mocap dans MotionBuilder. Elles nécessitent exactement les mêmes étapes de retargeting et retouche.

L'animation transférée sur notre acteur, elle est ensuite insérée dans la séquence à l'aide du mode « Story ».

La dynamique effectuée et intégrée, les étapes précédentes sont refaites en se concentrant sur l'arrivé du second personnage humain. Celui-ci essai de passer discrètement, mais fini par attirer l'attention du dinosaure qui lui cours après *(cf Fig. 3.13 p.19)* .

Finition et export

Une fois les bases de l'animation complète, la totalité de la scène (trajectoire et actions des personnages) a été plotée sur les « Control Rigs ». Ainsi a suivi un affinage des enchaînements et une correction des interpénétrations trop flagrantes. Nous en avons profité aussi pour ajouter des détails d'animation, des mouvements de tête et de bras.

On peut noter l'utilisation de relations pour animer la queue et la mâchoire du T-rex avec plus de facilité *(cf Fig. 3.15 p.21)* .

Une fois l'animation finie, l'ensemble est fixé en keyframing (plot) avant une derniere sauvegarde.

Le résultat est ouvert dans Maya où la scène subit un dernier nettoyage, en ne gardant que les animations de personnages.

Pour finir, l'animation définitive est exportée vers Virtools.

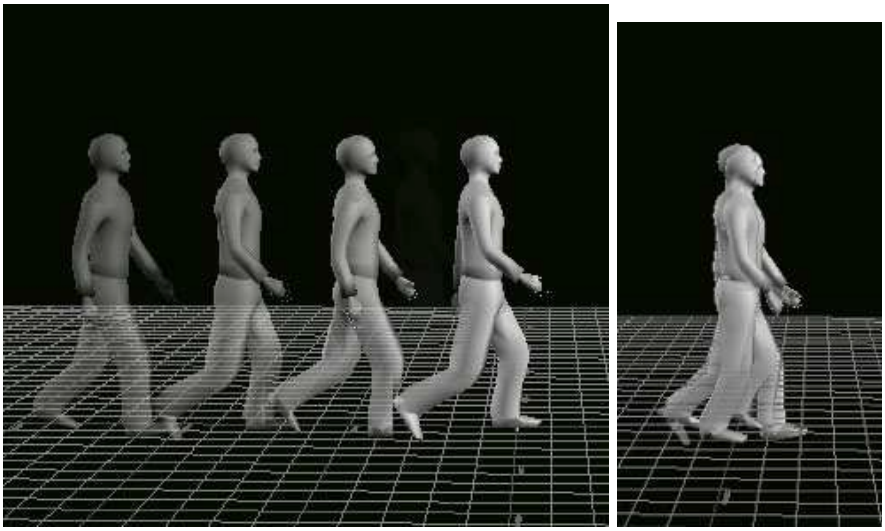

FIG. 3.10 – Mocap non cyclique et Mocap cyclique

FIG. 3.11 – Trajectoires des acteurs

FIG. 3.12 – Scène dynamique créée sous Endorphine

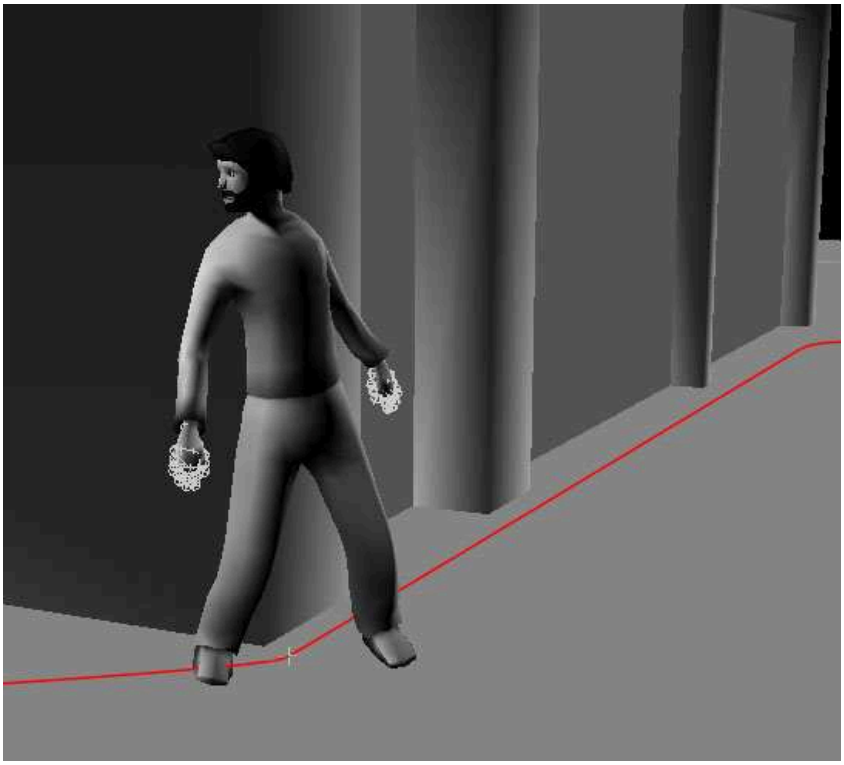

FIG. 3.13 – L'arrivée du second personnage

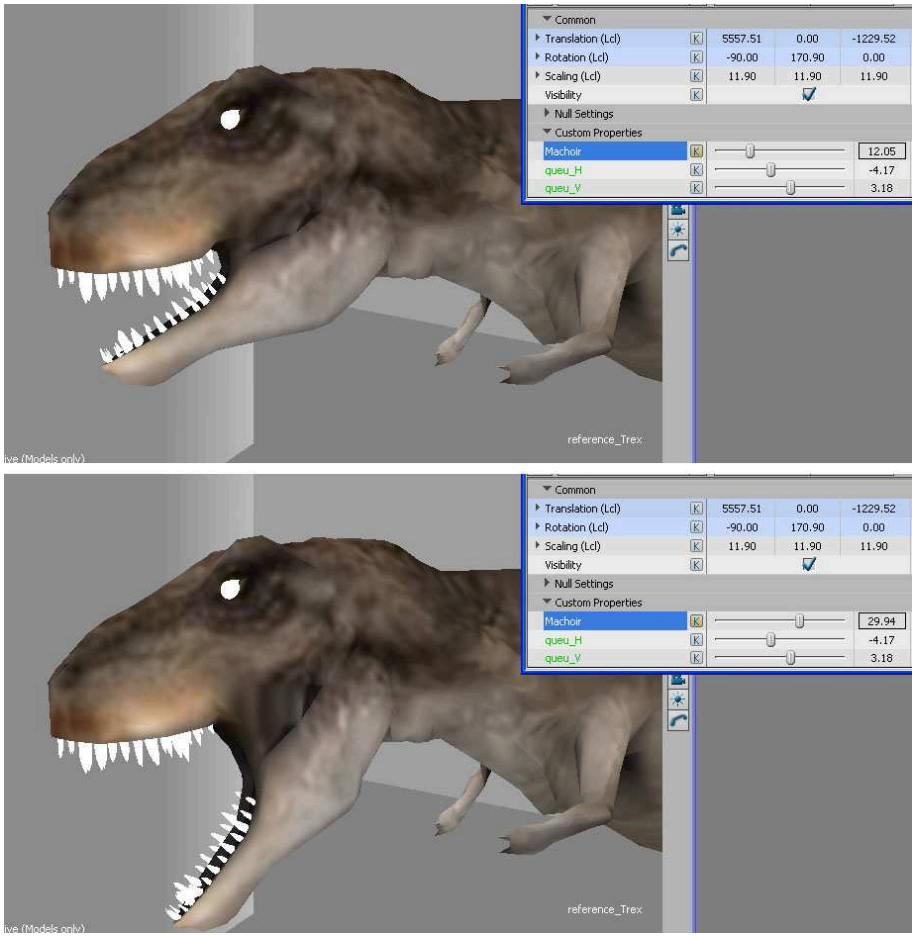

FIG. 3.14 – Contrôle de la machoire et de la queue du T-rex

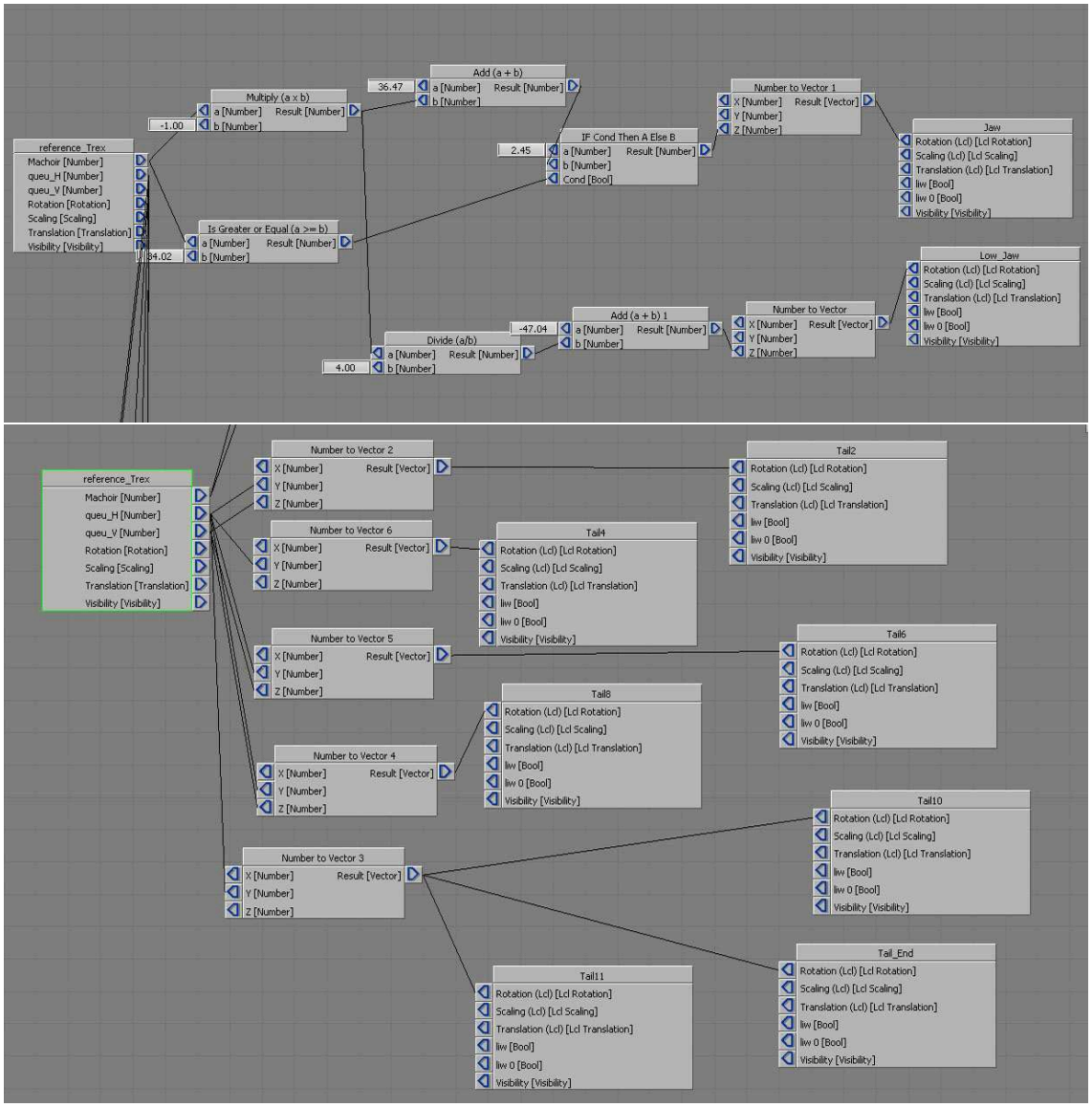

FIG. 3.15 – Les relations contrôllant la queue et la machoire du dinosaure

3.3 Éléctronique

Comme dit précédement l'idée de l'installation est de permettre à l'utilisateur de filmer du virtuel avec les fonctionnalités d'une caméra. Pour une interactivité optimale, il nous fallait créer un périphérique en forme de caméra capable de communiquer avec Virtools.

Dans l'équipe Rémi était le seul à avoir des notions d'électronique, c'est donc à lui qu'est revenu cette tâche.

La réflexion concernant ce matériel a commencé dés le début du projet.

Trouver une caméra

La premiere étape était de trouver une base pour l'électronique. Par chance, un réparateur électronique nous a grâcieusement donné une caméra hors d'usage *(cf Fig. 3.16 p.22)* .

FIG. 3.16 – La caméra originale

Analyse du system existant

Une fois la caméra trouvée, il a fallu la démonter entièrement pour comprendre son fonctionnement mécanique et électronique.

Ainsi, après une analyse et un repérage des composants, Rémi a pu trouver une stratégie pour dériver les circuits existants et en faire ce que nous souhaitions.

Recherche d'une interface connectable

A cette étape il nous manquait un élément important, une interface capable de convertir les signaux logiques et analogiques venant de la caméra en données compréhensibles par Virtools.

Notre première idée fut d'essayer la norme MIDI grâce aux cartes interfaces dont dispose la formation ATI. Mais après de nombreux tests plutôt décevants, Rémi eu l'idée d'utiliser une manette de jeu vidéo (gamepad). Nous avons donc fait l'acquisition d'une manette PC avec deux leviers anologiques. Une fois

démonté, c'est devenu l'interface parfaite, avec douze entrées logiques, quatre analogiques fonctionnant sans pilote (Plug'N'Play) sur port USB et parfaitement reconu par Virtools *(cf Fig. 3.17 p.24)* .

Ainsi en connectant les boutons de la caméra sur la manette de jeu, ceux-ci sont interprétés comme des boutons de la manette. Pour la bague de la focale, nous avons utilisé un potentiomètre, collé au mécanisme *(cf Fig. 3.18 p.24)* .

Recherche d'une solution pour le retour visuel

La caméra ne disposant pas d'un écran intégré, il fallait absolument trouver un autre système d'affichage.

Après de nombreuses recherches, nous avons finalement acheté un lecteur DVD portable disposant d'un écran $16/9^{ème}$ de sept pouces et d'une entrée composite *(cf Fig. 3.19 p.25)* .

Avantage, l'appareil permet d'utiliser directement la sortie vidéo de la carte graphique de l'ordinateur.

Le démontage fut sensible surtout en ce qui concerne l'ecran qui est la partie fragile du système *(cf Fig. 3.20 p.25)* .

Mais le résultat était très encourageant, il fallait ensuite intégrer les cartes électronique dans la caméra et créer un support pour l'écran.

Pour le support, une barre d'aluminium a directement été fixéé dans la caméra à l'aide de vis et d'écrous, l'écran lui même étant collé sur une tige rainuréé et une équaire métalique *(cf Fig. 3.21 p.26)* .

Boitier et cable

Une fois le plus complexe passé, il ne reste qu'à réaliser un câble pour faire passer les différentes données (video, usb , alimentation electrique...) et un boîtier propre pour connecter au PC *(cf Fig. 3.22 p.26)* .

FIG. 3.17 – Le Joystick

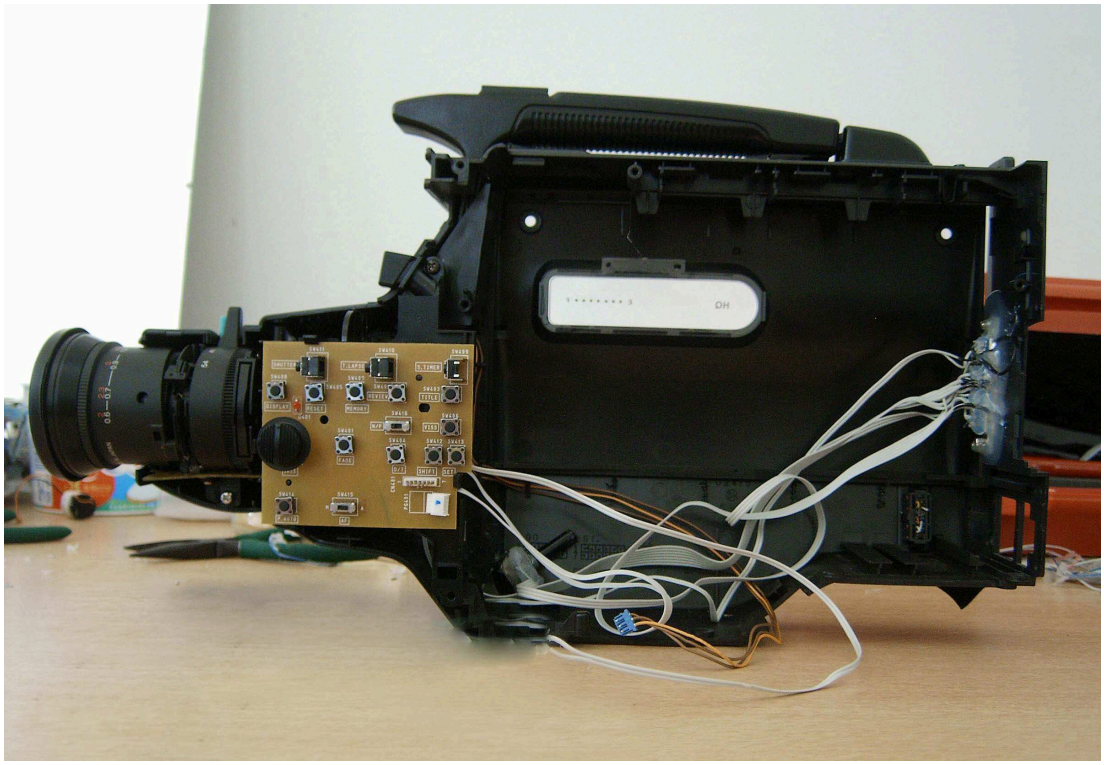

FIG. 3.18 – L'interieur de la caméra

FIG. 3.19 – Le lecteur DVD

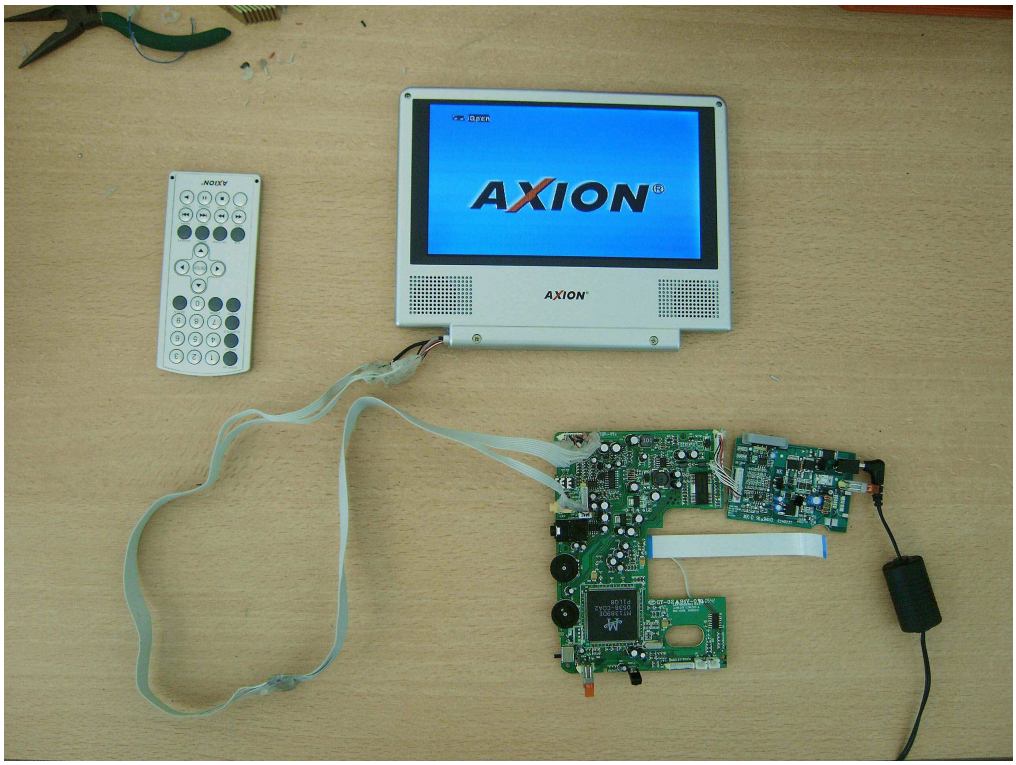

FIG. 3.20 – L'ecran démonté

FIG. 3.21 – La Caméra avec l'écran

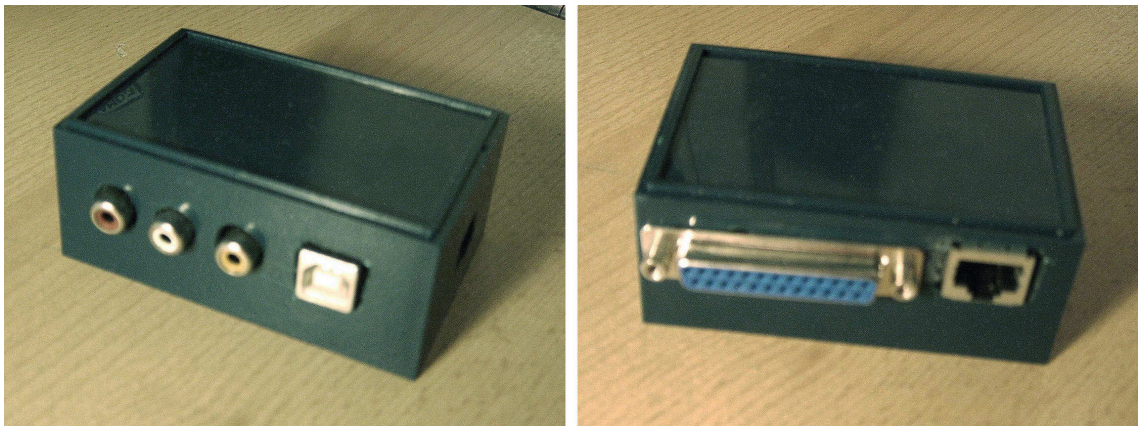

FIG. 3.22 – Le boitier

3.4 Création du site Internet

Après Laval, nous avons senti la nécessité d'être présents sur internet afin de mieux communiquer sur le projet. Rémi qui avait des compétences en Flash a donc mis en place un site web.

Ce site entièrement fait sous Flash Mx 8 est composé de cinq rubriques.

Chacune de ces rubrique est accessible depuis la page d'accueil et arrive avec une animation de déplacement.

Les trois premières sont les même que celle de l'application, à savoir Shoot, Cut, et Play. Chacune intègre une vidéo téléchargeable, et agrandissable.

La partie suivante, la rubrique Média est une galerie Photos et Vidéos. Chaque médias est chargé en dynamique via un fiché XML et est représenté par un petite vignette. Chacune d'entre elles est affichée dans une barre de navigation rétractable.

Enfin, la rubrique contact redirige le visiteur sur nos e-mail et le site Artweb de l'université Paris8.

Chapitre 4

Le rôle de Nicolas

Nicolas Serikoff a été chargé de tout le travail 3D et graphique du projet exception faite du site web et des personnages. De plus, suite aux contacts établis à Laval, il s'est chargé de tout l'ascept relationnnel du projet.

4.1 L'idée

L'utilité de la scène 3D est de pouvoir illustrer et montrer de manière ludique l'utilisation du projet SCPCamera.

A partir de là, nous avons cherché des idées d'environnements qui seraient les plus adaptés à nos attentes.

Nous avons retenu la scène dans la ville car cela permettait d'avoir de nombreux scénarios différents et originaux tout en restant cohérent à l'environnement.

Une des images qui vient en premier lorque l'on évoque la ville, c'est celle de grands buildings, de facades modernes et d'embouteillages. Nous avons voulu nous détacher le possible de cela en créant une ville plus intime et chaleureuse.

Pour cela, Nicolas a effectué un gros travail de recherche préalable et de repérages. Il a sillonné de nombreux quartiers de Paris (1er, 4ème, 5ème, 13ème...) qu'il connaissait pour aller y trouver les bâtiments et les détails qui l'ont nourri pour l'élaboration et la cohésion de la ville.

Près de 250 clichés ont été pris, allant du mur abimé, à la porte d'immeuble, en passant par la façade ou au bloc d'immeubles *(cf Fig. 4.1 p.29)* .

L'intérêt de cette démarche a été triple :
- aide à la prévisualisation , permettre de mieux se faire une idée de l'ambiance générale de la rue 3D ;
- aide à la modélisation , se servir des photos comme références pour la modélisation 3D ;
- aider à la création de texture , avoir de la matière pour les textures.

Ces nombreux clichés ont permis donc de sélectionner les éléments qui semblaient intéréssants à mettre dans notre scène.

Tout les éléments ont été photographiés plusieurs fois et sous des angles légèrement différents pour s'assurer d'avoir suffisament de clichés exploitables.

A partir de là, nous avons pu débuter le travail de modélisation à partir des photos.

FIG. 4.1 – Quelques exemples de sources d'inspiration

4.2 La modélisation

La modélisation a toujours été faite à partir de polygones, parfaitement adaptés à la modélisation architecturale dont nous avions besoin.

Par ailleurs, les Nurbs ou les subdivisions de surfaces de maya ne sont pas des techniques de modélisation gérées ou optimisées par les moteurs temps-réel.

Pour éviter d'avoir une scène trop cubique, de nombreux Biseaux (Bevel) ont été fait sur les arêtes des murs ou des fenetres.

Cette technique qui ajoute des polygones trouve son intérêt dans le fait de casser les angles droits et ainsi de permetttre à la lumière de mieux éclairer l'objet.

L'effet de spéculaire est plus visible et adoucie le modéle 3D *(cf Fig. 4.2 p.30)* .

FIG. 4.2 – Comparaison Biseauté/non biseauté

Afin de ne pas avoir à refaire une modélisation spécifique pour le temps-réel et pour le précalculé, nous avons décidé d'utiliser une modélisation low-poly pour les deux modes.

Toutefois avec les performances accrues fournies par le hardware, la limite polygonale n'a pas été un véritable souci pour nous. De plus, l'étendue modélisée et la complexité de la scène ne rivalise pas avec celle de jeux vidéos où la contrainte reste plus forte.

Et même si les dernières générations de cartes graphiquse(GPU) allègent le processeur(CPU) de certains calcul 3D complexes, la taille des textures reste plus limitée en temps-réel.

4.3 Les textures

Pour Maya, la taille des texures ne représente pas une limite mais allonge le temps de rendu. Nous nous sommes donc limité à une taille de 1024*1024 pixel *(cf Fig. 4.4 p.32)* , car avec une taille superieur sur un écran en 1024x768 on n'aurait pas vu la différence.

Afin d'optimiser la scène Virtools, j'ai limité la taille des textures à 128 pixels maximum pour environs 110 textures *(cf Fig. 4.3 p.31)* . Les textures varies donc entre 32 et 128 pour Virtools.

FIG. 4.3 – Exemples de textures optimisées pour le temps réel

Pour ajouter du réalisme à la scène, nous avons utilisé une passe d'AmbienteOcclusion (ou Dirtmap) qui a servie pour les ombres de contacts. D'habitude, ce genre de technique s'utilise en précalculé, donc pour réutiliser ces informations d'ombrage nous avons dû rendre sur une texture spécifique ces informations (baking).

Ensuite, dans Maya j'ai mis en color des Layered texture qui permettent la superposition de plusieurs textures :
- la première texture étant toujours ma dirtmap ;
- la seconde étant la texture de diffuse.

Cette technique est extrêment pertinente car elle permet d'exporter le shading sans aucun souci vers Virtools qui à son tour combine les textures. On retrouve ainsi le même résultat d'affichage entre Maya et Virtools.

A la façon des calques de Photoshop, on peut spécifier différents modes de fusion (superposition, multiplication...) *(cf Fig. 4.5 p.33)* .

Un des défis était donc de travaillller sur une seule scène qui servait en simultanée pour le temps-réel (Virtools) et pour le précalculé (Maya).

Nous venons de voir que grâce aux outils spécifiques de Maya, il nous avait été possible de mélanger des techniques du précalculé avec le temps-réel.

Toutefois, nous avons dû développer quelques outils spécifiques pour nous permettre de travailler de manière optimale.

FIG. 4.4 – Exemples de textures grand format

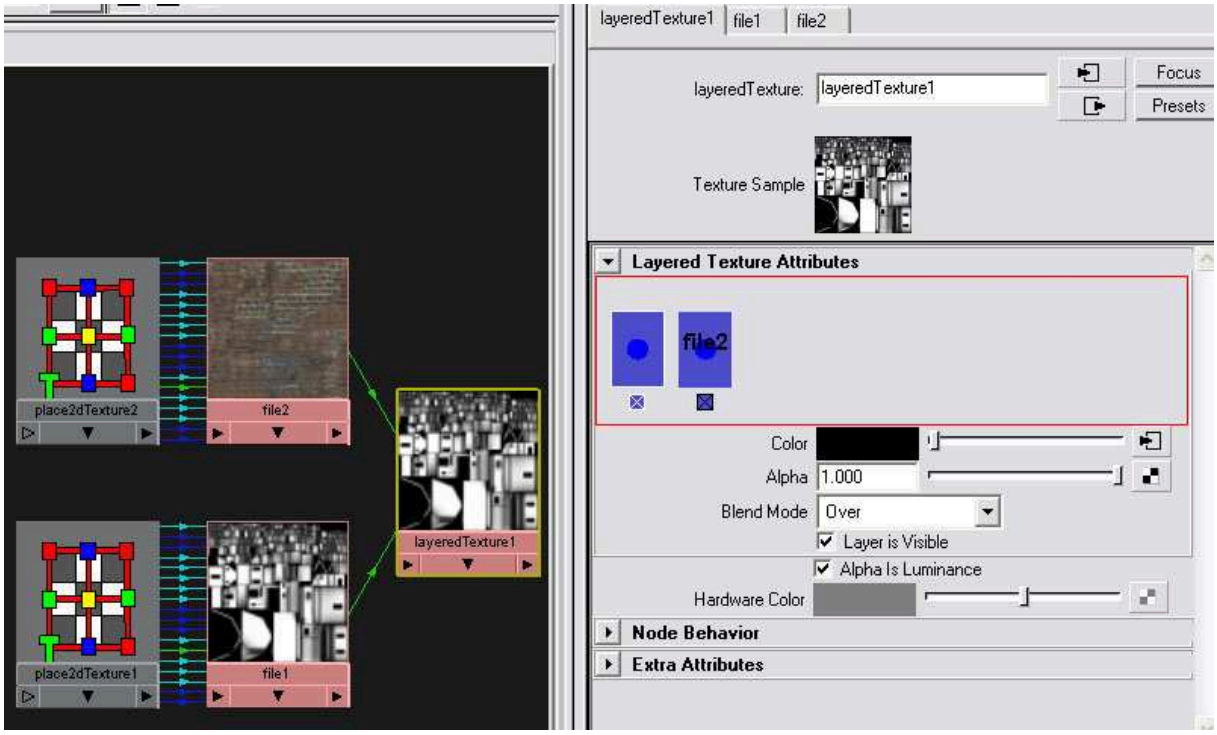

FIG. 4.5 – Utilisation des LayeredTexture

4.4 Scripts

Assez rapidement nous nous sommes apercu que certaines fonctions été utilisées très régulièrement.

Plutôt que de parcourir sans cesse l'interface Maya, Nico a décidé de créer sa propre boîte à outils en Mel *(cf Fig. 4.6 p.34)* qui donnait un accès direct aux fonctions les plus utiles : outliner, hypergraph, UvEditor, outils de modélisation...

Au fur et à mesure de l'avancée du projet, les fonctions se sont étoffées et cela a permis aussi d'intégrer facilement les outils développés par Xavier.

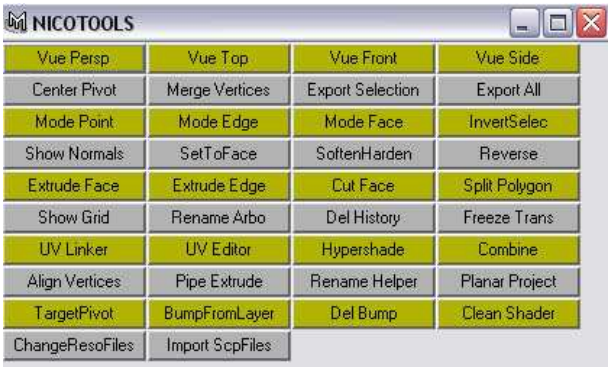

FIG. 4.6 – Les célèbres Nico'sTool

Avec l'utilisation des grandes textures, s'est posé la question d'optimisation pour Virtools.

Pour répondre à cela, Xavier a crée un script, le ResolutionFiles qui faisait permutter les connections des matériaux entre des textures hautes résolutions pour Maya et basses définitions pour le temps-réel.

C'est ce système qui nous a réellement permis d'utiliser la même scène quelque soit la sortie car ainsi nous n'avions qu'à permutter le node approprié en BasseResolution pour préparer l'export Virtools.

Pour augmenter la qualité des rendus, nous avons décidé d'utiliser du BumpMapping sur les nodes de Diffuses des objets.

Xavier a confectionné un script reliant automatiquement un node de Bump à partir du 2ème Layered-Texture (texture de diffuse) d'un matériau *(cf Fig. 4.7 p.35)* .

Ce script a permis de faire évoluer rapidement la scène précalculée et d'en augmenter la qualité finale.

Et à l'inverse pour préparer l'export vers Virtools, il a fallu créer un script permettant de supprimer seulement les informations rajoutées par le ResolutionFiles et le BumpFromLayer.

Le système du Resolution Files a été appliqué ici uniquement à des textures mais il est tout à fait possible de l'adapter pour gérer une sorte de Level Of Detail(L.O.D) qui offrirait la possibilité de permutter entre des mesh high-poly pour du précalculé et des mesh low-poly pour du temps-réel.

L'automatisation par script nous a permis deux choses :
– gagner du temps de production ;
– éviter toute mauvaise manipulation dû à une tâche délicate ou répétitive.

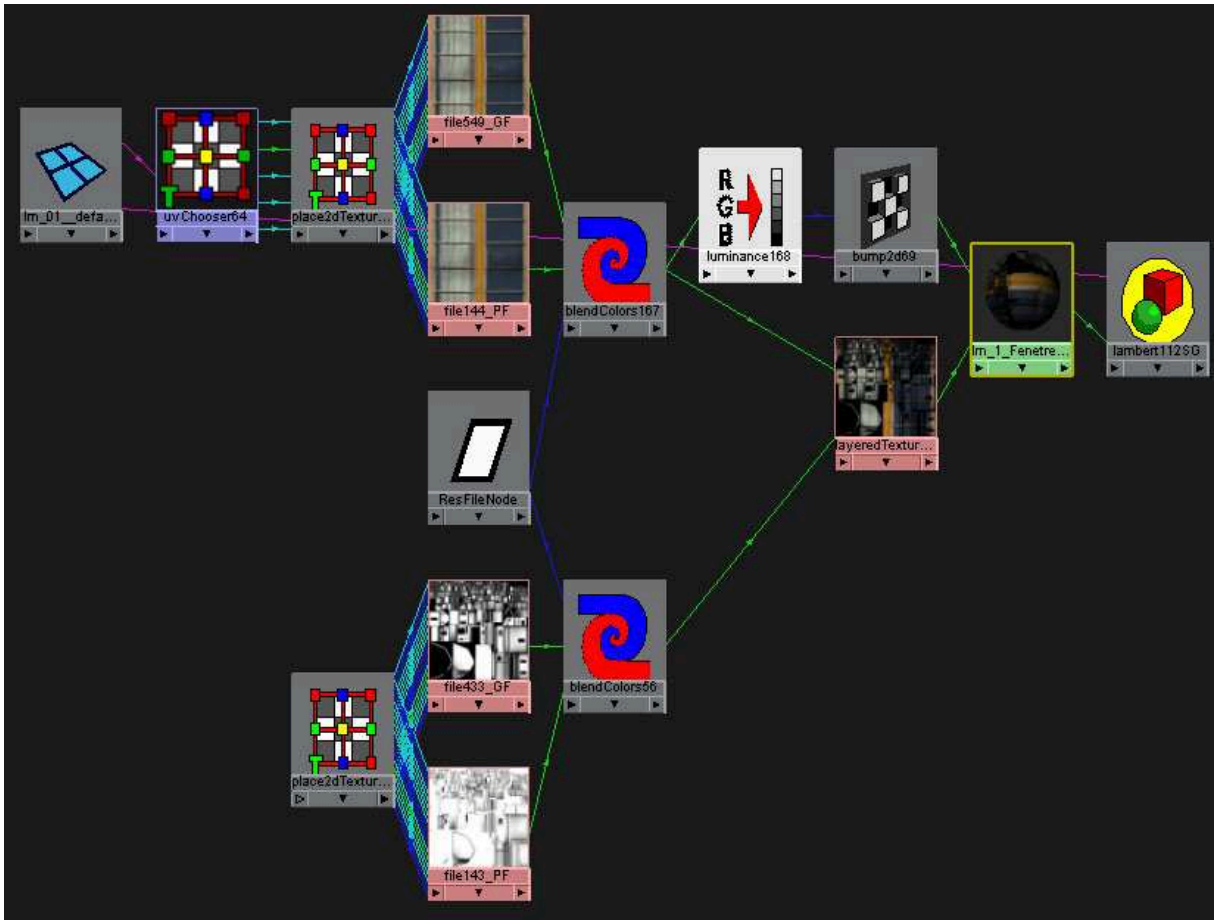

FIG. 4.7 – Graphe d'un shader avec ResolutionFiles et Bump appliqués par script

FIG. 4.8 – Textures Petit Format Sans Bump

FIG. 4.9 – Textures Grand Format avec Bump

4.5 Evolution de la chaine de production graphique

FIG. 4.10 – Photo référence d'un immeuble

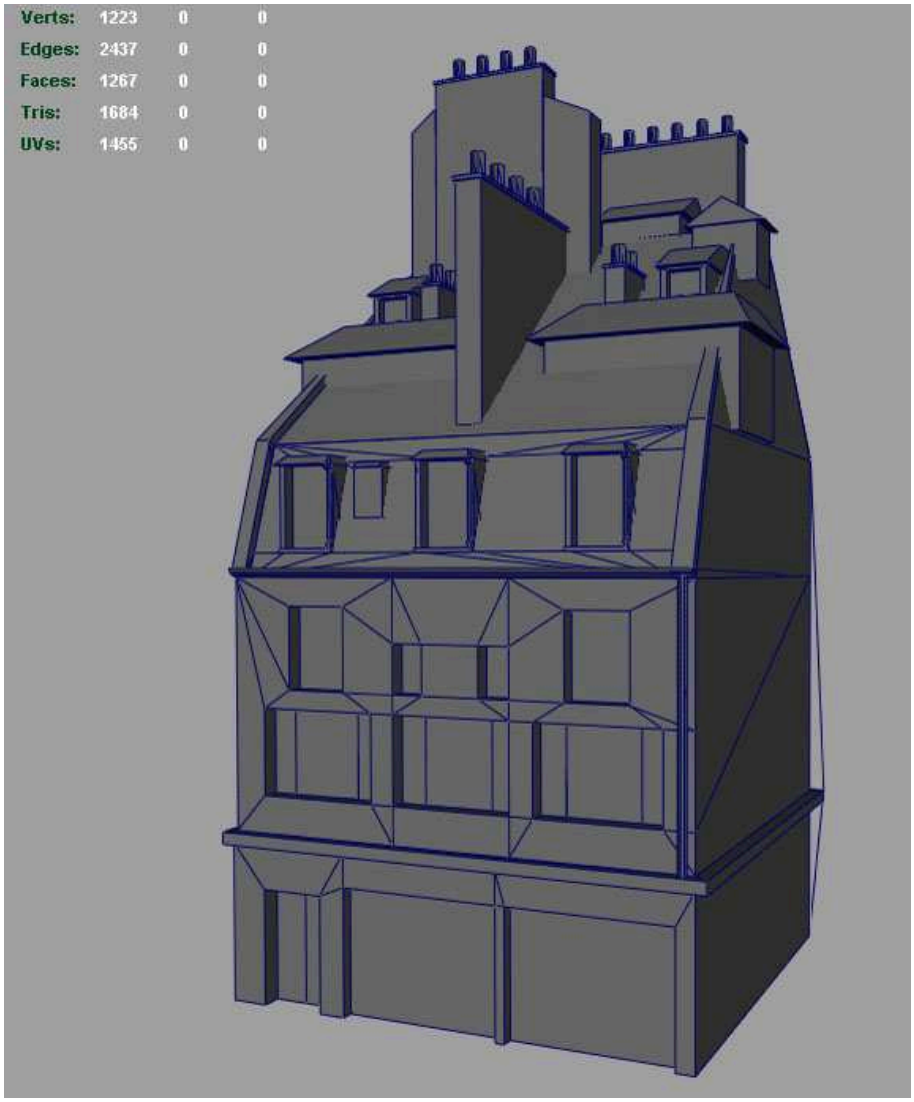

FIG. 4.11 – Modélisation sous Maya

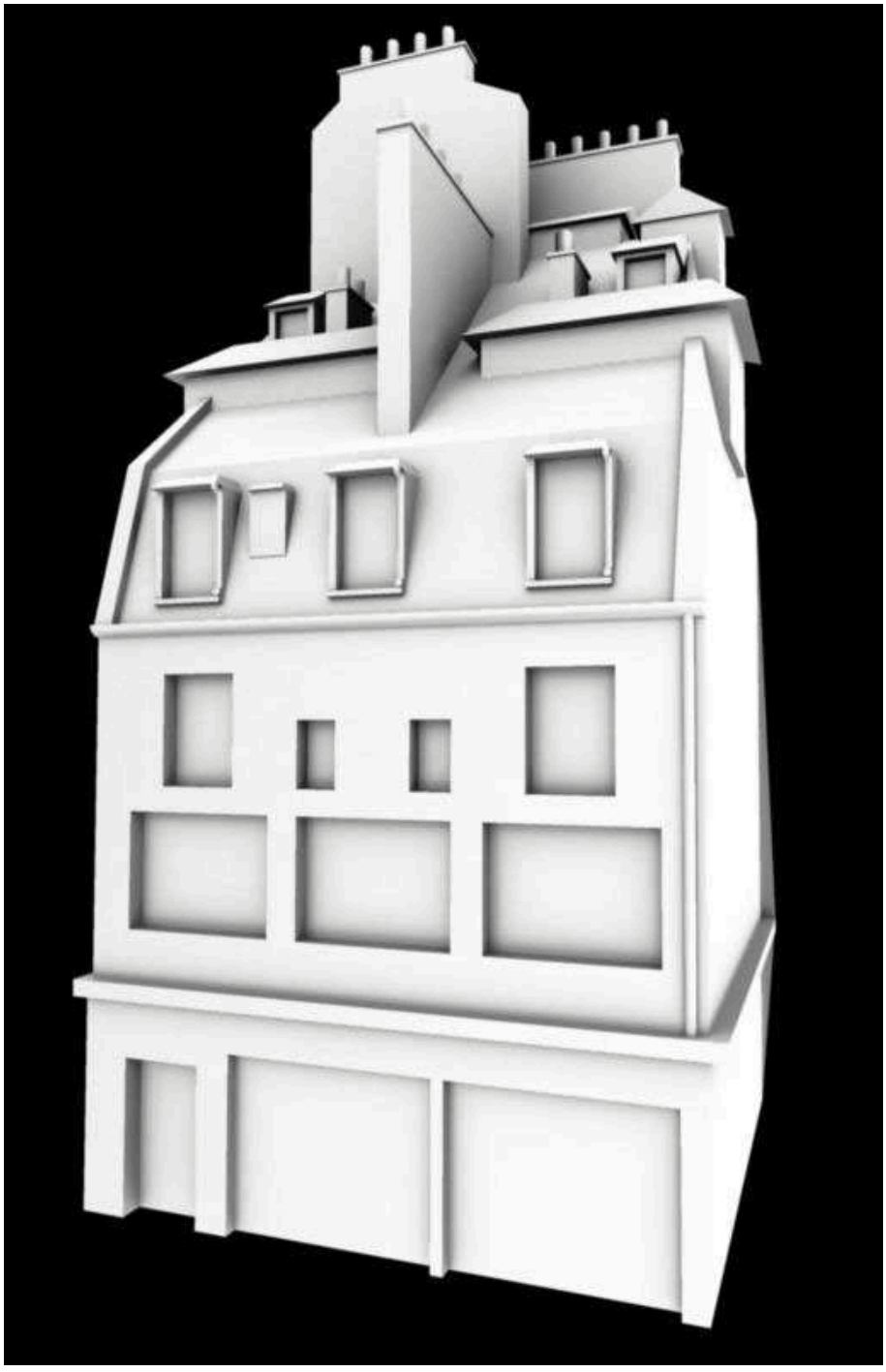

FIG. 4.12 – Ombres de contactes avec passe d'Ambiante Occlusion

FIG. 4.13 – Textures Temps-réel Maya

FIG. 4.14 – Temps-réel Virtools

FIG. 4.15 – Rendu précalculé Maya

4.6 L'interface

Après Laval, il était évident que si nous voulions pousser encore plus loin l'application, nous devions créer une interface utilisateur.

Dans un premier temps, nous avons mis sur papier toutes les fonctions qui devaient figurer dans le projet, afin d'avoir un apercu clair.

Nous avons choisis de différencier les trois modes d'utilisation grâce à l'utilisation de trois couleurs.

Chaque élément de l'interface peut donc s'afficher dans les trois couleurs selon le mode utilisé.

Ainsi grâce au code couleurs, l'utilisateur doit pouvoir savoir dans quel mode il est uniquement en jetant un coup rapide à un des éléments de l'interface.

L'ensemble des éléments de l'interface a été réalisé sous maya avec des primitves comme les cylindre ou les spheres.

Pour permettre de faire mes réglages de colorimétrie de manière plus souple, les rendus étaient fait en noir et blanc (Dirtmap) et étaient ensuite retouchés sous photoshop pour la couleur *(cf Fig. 4.16 p.43)* .

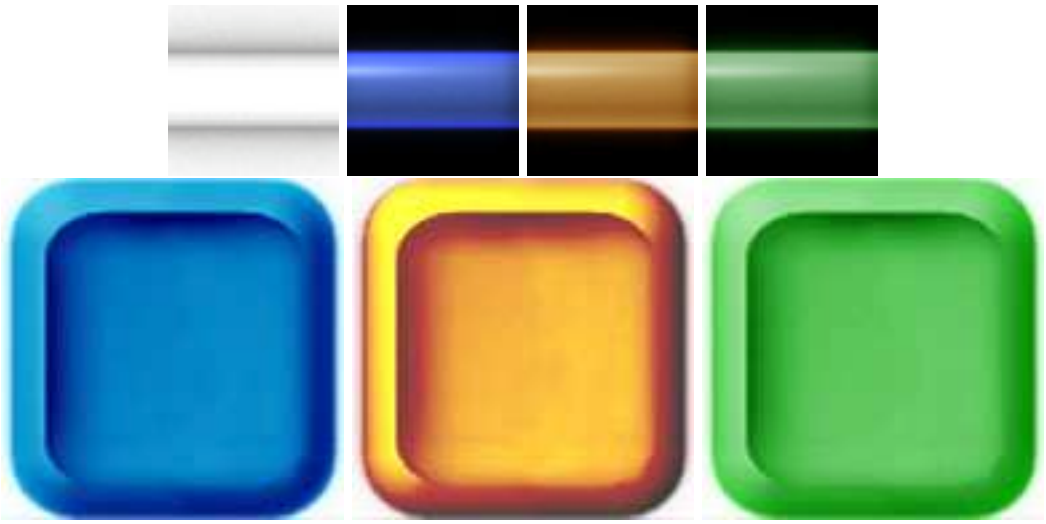

FIG. 4.16 – Les principaux elements de l'interface en differentes couleurs

Avec cette méthode, nous avons pu aussi créer sans problème les différents états des boutons car comme sur un site web, les boutons changent selon l'action du curseur *(cf Fig. 4.17 p.44)* .

Nous avons également créé un code d'icônes, renseignant l'utilisateur sur l'effet de chaque bouton/fonction *(cf Fig. 4.18 p.44)* .

L'interface a été un moment d'intense collaboration entre Xavier (développeur) et Nicolas (Graphiste) car aussitôt les éléments graphiques créés, ils étaient intégrés dans Virtools. Nous précedions ainsi à de nombreux tests avant d'arriver au résultat final.

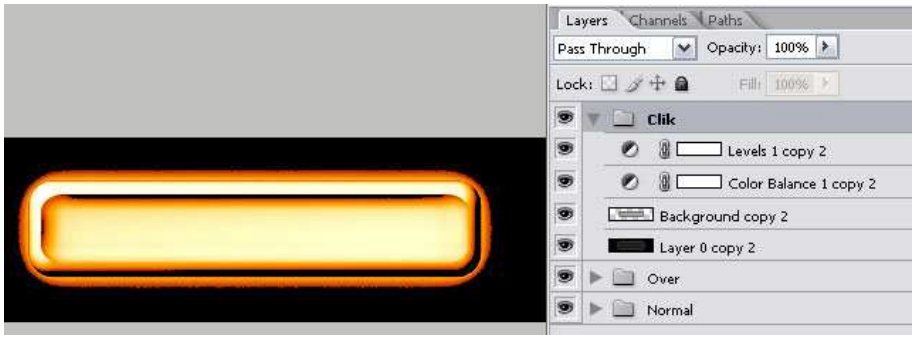

FIG. 4.17 – Création des differents état des boutons

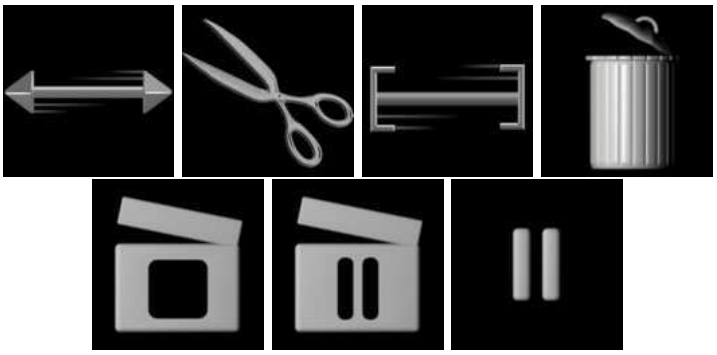

FIG. 4.18 – Les Icones des differentes options

Chapitre 5

Le rôle de Xavier

Le rôle principal de Xavier a été de créer le programme sous virtools, le plug-in d'import sous Maya, et tous les outils nécessaires pour l'organisation et le travail des collègues.

5.1 Programmation sous Virtools

Enregistrement, sauvegarde et lecture

La première partie qui a été programmée fut la partie d'enregistrement des mouvements de caméra. Pour cela, à chaque frame on enregistre la position et l'orientation (c'est-à-dire la Matrice) de la caméra séléctionnée dans un tableau donc chaque ligne correspond à une frame. Dans ce tableau nous gardons également le zoom, la distance focale.

A chaque fois que l'utilisateur arrête d'enregistrer, le tableau est sauvegardé dans un fichier temporaire unique, lequel est réutilisé pour le montage.

Pour gérer le montage, nous utilisons un second tableau qui contient l'organisation des differents enregistrements. Chaque ligne du tableau correspond à un clip, et contient le fichier correspondant, la première et dernière ligne du fichier qui sont utilisées, le numéro unique du clip, ainsi que sa position (qui permet de déterminer l'ordre dans lequel sont disposés les clips).

Enfin, à chaque fois qu'un clip est ajouté, supprimé ou modifié, le mouvement "final" est recalculé. Il contient les mêmes informations que le tableau d'enregistrement, avec en plus pour chaque frame, le numéro du clip correspondant.

Animation

Il est possible d'importer des objets/personnages avec des animations pour avoir un repère lors de l'enregistrement des mouvements de cameras.

Pour gérer l'ensemble des animations (pour le cas où il y en a plusieurs), et éviter les conflits lors du passage à Maya, nous avons utilisé une variable contenant le code temporel des animations. Ainsi, toutes les animations sont synchroniséesz, et les mouvements de caméras se placent au bon endroit dans la timeline de Maya.

Montage

Pour le montage, les seuls tableaux modifiés sont celui du montage et l'enregistrement final.

Pour le cut, on duplique le clip selectionné et on modifie juste la première/dernière ligne utile du fichier correspondant.

Pour le crop (modification des bornes d'un clip), on ne fait que modifier la première ou dernière ligne utile du fichier correspondant.

Pour le drag and drop (glisser-deposer), on modifie juste la position du clip qui a été déplacée, ainsi que les clips dont la position est modifiée.

Gestion des codes temporels

Dans la partie Virtools, nous avons 3 codes temporels differents.

Le temps absolu

(ou temps d'animation). Ce code correspond au temps gérant l'animation des personnages dans virtools. Il correspond ainsi à la timeline de Maya et permet d'avoir une synchronisation entre les mouvements de caméra et les mouvements des personnages, que ce soit lors de la relecture dans Virtools ou à l'export dans Maya.

Le temps relatif

(ou temps d'enregistrement). Ce code temporel commence à zéro au début d'un enregistrement de mouvement. Il permet d'avoir des valeurs plus cohérentes et manipulables lors de l'édition d'un clip (borne du clip, longueur etc...).

Le temps de lecture.

Ce code est uniquement créé pour faciliter la compréhension de l'utilisateur lors de la relecture d'un montage dans virtools. Il commence à zero au début du montage, et a pour longueur celle du montage.

Foule Pseudo Dynamique

Lorsque le coeur du programme fut opérationnel et que nous avons eu l'occasion de tester notre projet, notre scène démo semblait un peu vide. Nous avons donc décidé d'ajouter un peu de vie avec une foule que nous voulions dynamique.

La première idée que nous avons essayé fut de créer un système de path finding pour avoir une foule réelement dynamique. Mais cette methode fut rapidement écartée, car trop lourde à utiliser.

Ensuite nous avons décidé d'utiliser des courbes, préparés dans Maya, pour avoir un contrôle sur la foule. Cela résolvait non seulement le choix des trajectoires mais aussi les soucis de collisions avec le décors.

Le soucis était que l'export de Maya vers Virtools ne fonctionnait pas avec les courbes. Il a donc fallut créer un script Mel et un script VSL pour pouvoir importer des courbes dans Virtools (voir plus loin).

Ensuite, il a fallu trouver un moyen pour que la foule soit enregistrée en même temps que le mouvement de caméra. En effet, des utilisateurs trouvaient frustrant de faire leurs cadrages en fonction de la foule, et de ne pas retrouver les mêmes images lors de la relecture.

Pour cela, nous avons créé un tableau qui lors d'un enregistrement, récupere la position de chaque personnage sur une courbe (laquelle est aléatoire). Chaque personnage, pour faciliter la gestion de la foule, ne se déplace que sur une seule courbe.

Ensuite, lors de la relecture, il nous suffit de replacer chaque personnage à sa position sur la courbe.

Le didactitiel

Suite au succès obtenu lors du Festival de Laval, nous avons décidé d'intégrer un didactitiel, permettant à un utilisateur néophyte d'apprendre l'utilisation de base d'une caméra (Plan large, gros plan, Champs/contrechamps, plongée/contreplongée, etc...).

Il a donc fallut pour cela trouver un moyen de vérification des actions de l'utilisateur. L'idée est d'utiliser des boîtes centrées sur une action (deux personnages face à face) plus ou moins grandes. Nous plaçons ensuite des locators aux extrémités d'une diagonale et au centre de chaque boîte.

Une fois importés dans Virtools, il suffit de vérifier que les locators aux extrémités ne sont pas visibles et que le locator centrale l'est. Cela indique que "globalement", la caméra est pointée dans la bonne direction, et placée au bon endroit.

Lorsque l'utilisateur doit effectuer une action (se déplacer/zoomer), une boucle teste à chaque fois la visibilité dans le cadre de la caméra des locators "utiles", et ne passe la main que lorsque les conditions sont remplies.

5.2 Plug-in Virtools

Filtre anti-bruit

L'un des premier plugs créé pour l'application fut le filtre anti-bruit, permettant d'atténuer les effets de saut lors de l'utilisation de la souris pour contrôler la camera, et le bruit dû au capteur 3D (Magnétique ou infrarouge).

Le principe qui a été utilisé est de conserver en mémoire les 3, 5 ou 7 dernières positions récupérées à l'aide d'un capteur, ou de la souris, puis d'en faire cette moyenne. Nous utilisons alors cette moyenne comme position de la caméra, que nous enregistrons le cas écheant.

Ce type de filtre reste classique mais efficace, pour peû que la machine soit suffisament puissante. En effet, nous avons remarqués sur certaines machines un ralentissement lors de l'utilisation du shader de Depth Of Field. De fait, lorsque nous utilisons un filtre avec 7 valeurs, on observe un retard entre le mouvement capturé et le retour video.

Client Serveur

Pour la version de Laval (Avril 006), nous avons tenté de palier au problème du capteur (qui ralentissait notre application). Il a donc fallut développer un serveur non bloquant, permettant de récuperer les informations du capteur (Polhemus Isotrack II).

5.3 Plug-in Maya

Charger un montage

Le principal intérêt du plug-in Maya est de récuperer des mouvements de caméra créés sous Virtools. Pour cela, on lit le fichier texte correspondant à l'export, qui contient sur chaque ligne les codes temporels enregistrés dans virtools, la matrice de la caméra, les informations de zoom et de profondeur de champs et le numéro du clip.

Ainsi à chaque ligne correspond une clef sur la timeline. De plus, pour éviter des conflits et des résultats imprévisibles, nous avons décidé de créer une caméra par clip.

FIG. 5.1 – Le Plug In Maya

Convertir les données

Entre Virtools et Maya, il y a une différence dans le type de donnée. En effet, l'axe Z n'est pas orienté de la même manière dans Virtools et dans Maya.

Pour parer à ce problème, il a fallu modifier la position et l'orientation de la caméra importée afin d'avoir un résultat identique à celui obtenu dans Virtools.

Options de chargement

La première option de chargement (Animation Import) est utilisée pour définir la disposition des clips par rapport à la timeline de Maya. On peut décider d'utiliser le temps absolu comme référence (les mouvements de caméras sont alors synchronisés avec les animations de la scène Maya) en cochant la case "Whole Camera". Par contre dans ce cas, si aucune animation n'a été chargée dans Virtools, le résultat dans Maya sera imprévisble et globalement incohérent avec les mouvements créés dans Virtools. On peut également poser les clefs sur le temps relatif, ou temps de lecture (Option "Camera Only"). Les clips sont alors "posés" les uns après les autres sur la timeline de Maya, sans considération pour les éventuelles animations présente dans la scène.

La seconde option est la création de fichier batch (Create Batch File), evitant à l'utilisateur de lancer ses rendus. Le batch lance les rendus dans l'ordre, avec les bons intervalles de frames, et renomme les images en fonction du numero de clip correspondant.

On peut choisir lors de l'import, d'utiliser ou non les informations de profondeur de champs (Use Depth of FIeld). La correspondance entre la profondeur de champs sous virtools et dans Maya est approximative, mais fonctionne relativement bien. Les informations récupérées dans virtools sont cependant enregistrés dans la camera, et utilisable si le paramètre Depth of Field est coché sur la caméra voulue.

Les clefs posées sont dépendantes de la rapidité d'execution sous virtools. Plus l'ordinateur sur lequel les mouvements sont enregistrés est puissant, plus il y aura de clefs sur un même intervalle de temps. Pour éviter d'avoir trop de clefs, le paramètre keystep permet de n'utiliser qu'une partie des clefs. Si ce paramètre est mis à 4 par exemple, lors du chargement, on n'utilisera qu'une clefs sur 4.

Le dernier paramètre, le "Correct Warp error" permet de palier à un problème sur les rotations *(cf Fig. 5.2 p.49)* . En effet, les valeurs récupérées pour chaque axe sont situées entre 180^o et -180^o. Le résultat est que pour des mouvements censé être continu, on assiste à des sautes, se traduisant par des mouvements désagréables. Cette option tente de supprimer cet effet en analysant chaque clef par rapport à la précédente. L'effet est le plus souvent corrigé, sauf dans certains cas. Cependant, si le nombre de clefs est suffisament important (une ou deux par frame minimum), l'option est inutile car les sautes ne se voient pas.

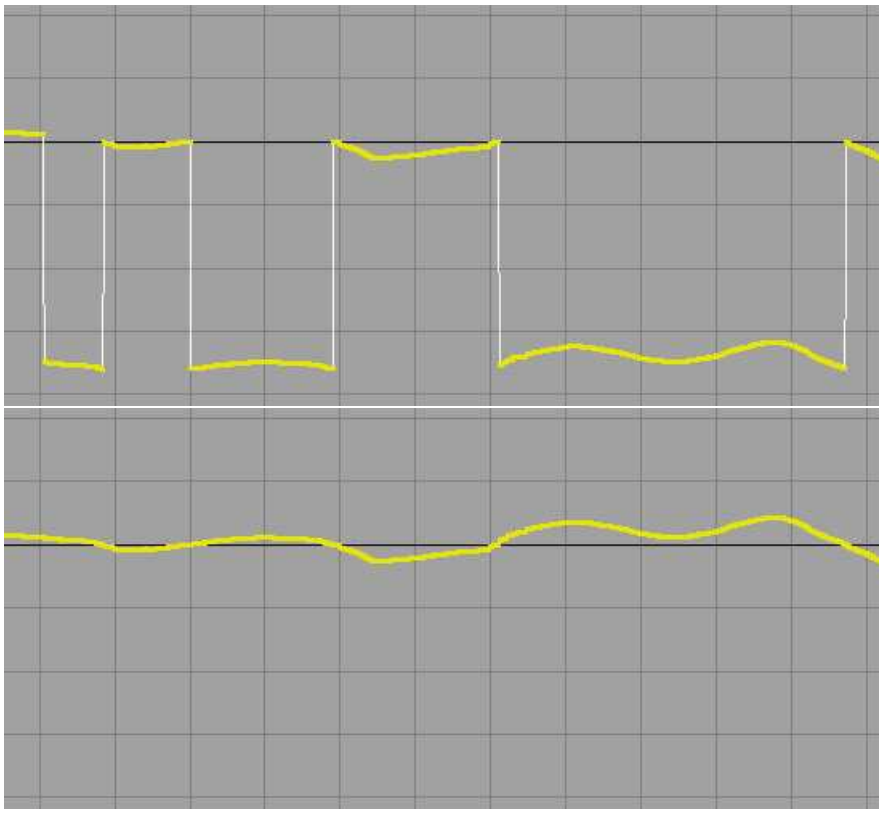

FIG. 5.2 – Avant et apres la correction des rotations

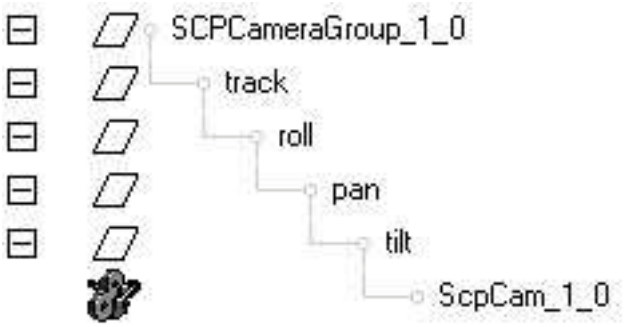

FIG. 5.3 – La hierarchie de notre camera dans Maya

Une camera décomposée

Suite à quelques recherches, nous avons décidé d'utiliser la caméra décrite lors d'une Conférence Alias-Wavefront *(cf Fig. 5.3 p.49)* .

La caméra en question est en fait une hiérarchie de nodes qui permet de décomposer le mouvement de la caméra en action rappelant la manipulation d'une caméra réelle.

Le node "track" est utilisé pour déterminer la position de la caméra. Ensuite, le noeud roll sert pour le tanguage (rotateZ). Puis le pan sert à déterminer l'orientation de la caméra (rotateY). Enfin, le tilt est utilisé pour le roulis (rotateX).

L'ensemble des paramètres des nodes (translate et rotate), ainsi que la focale et le depth of field sont controllés par le node à la racine de la hiérarchie.

5.4 Scripts et outils

Durant le dévelopement du projet, plusieurs scripts et outils ont dû être créés afin de faciliter notre travail, et de réduire les temps de production.

Gestions de Shaders dans Maya

Bump from Layer

Ce petit script permet d'utiliser le $n^{ième}$ layer d'un "Layered Texture" comme texture de Bump.

Resolution Files

La scène 3D que nous avons créé comme démonstration contenait deux niveaux de détails : un pour l'export vers Virtools et un pour les rendus. Ces deux niveaux de détails se traduisaient par deux textures de résolutions différentes pour chaque objet.

Le script permettait de créer un shading groupe qui contenait les deux texture, mais une seule était utilisé. Un node contenant un parametre modifiable permettait de changer la texture utilisée pour tous les shaders de la scène.

Clean Shaders

Ce script était utilisé conjointement avec le précédent. Il permettait de supprimer tous les nodes inutiles à l'export vers Virtools (Blend colors, Bump, etc...)

Aide à la modélisation

Pipe

Ce script automatise la création de tuyau à partir d'une seule courbe. Il place un cercle à la base de la(les) courbe(s) seléctionnée(s), et applique un extrude. Ce script a été utilisé notamment pour les gouttières de la ville et les fils éléctriques.

Align Vertices

Ce script permet d'aligner automatique un groupe de vertex, edges, face ou control vertex selon un axe, avec différents paramètres (les aligner sur le vertex ayant la coordonnée la plus grand, la plus petite, ou à une valeur paramétrable).

Target Pivot

Ce script permet de placer le point pivot d'un objet à l'emplacement d'un vertex, d'un point sur une courbe, d'une face ou tout autre point dans Maya.

Outils pour le passage de Maya à Virtools

Renamer

Cet outil a été créé pour la version de Laval (Avril 2006). Il a été utilisé principalement pour marquer les objets dans Maya, afin de pouvoir leur appliquer des paramètres dans virtools.

Par exemple nous avons utilisé le flag BS pour marquer les objets "Both Sided", c'est a dire les objets comme les parasoles, que nous devont pouvoir voir des deux côtés.

A chaque flag correspondait une action faite à l'import dans Virtools.

Export de courbes

Lorsque nous avons décidé de créer une foule pseudo dynamique, nous avons décidé d'utiliser des courbes pour déterminer le chemin des personnages. Mais l'outil de création de courbes sous virtools n'était pas suffisament complet à nos yeux, et il n'existait pas d'export de courbes entre Maya et Virtools.

Curve to Locator Nous savions que l'on pouvait exporter des locators depuis Maya vers Virtools. La première partie était donc de décomposer la courbe dans maya et locators, correspondant aux "edit points" de ladite courbe *(cf script p.54)* .

Frame to Curve Ensuite une fois les locators importés dans virtools, un script VSL *(cf script p.55)* permet de les reconvertir en courbes, avec la possibilité de créer des courbes linéaires ou béziers, ouvertes ou fermées.

Traitement des données par lots

Enfin certaines opérations devaient être faites sur un grand nombre d'objets lors de l'import dans Virtools. Pour éviter un travail long et fastidieux, il a fallut créer des actions en VSL permettant de faire ces opérations sur une sélection multiple.

La première de ces operations fut de passer la couleur "Diffuse" des matériaux d'un gris clair à blanc. En effet, lors de l'import par défaut depuis Maya, nous nous sommes retrouvés avec tous les matériaux ayant une diffuse en gris et non en blanc, ce qui obscurcissait beaucoup la scène.

Ensuite il a fallut modifier le mode de rendu de certains objets, en les mettant "Both Sided", c'est à dire visible des deux côtés *(cf script p.56)* .

Puis il a fallut (du moins lors de la premiere version) selectionner les objets utilisés lors du calcul des ombres.

Enfin, toujours dans la première version, nous avions un flag permettant de reconnaître la situation géographique de chaque élément de la scène *(cf script p.56)* . Dans un but d'optimisation, les éléments appartenant à un même bloc (en général correspondant à un batiment) était regroupé en "place", lesquels n'étant affiché que si nécessaire. Ceci permettait d'optimiser l'utilisation de la carte graphique.

Personnage dynamiques

Lors de la première version, les personnages avaient un matériau par élément (pantalon, pull, peau etc...). Un script a été créé afin de génerer une foule dynamique.

Ce script *(cf script p.57)* permettait de copier le personnage de base, en donnant une nouvelle couleur à chacune de ses parties. Ce script a permis en quelques minutes à peine de créer facilement une demi-douzaine de personnages différents, ajoutant ainsi de la vie à notre scène.

Chapitre 6

Conclusion et ouvertures

6.1 Conclusion

La réalisation de ce projet a pour nous trois été bénéfique à plusieurs niveaux.

Sur le plan technique tout d'abord, nous avons mis en pratique et enrichie notre savoir. Il a fallu faire appel à notre ingéniosité pour dépasser certaines contraintes (matériel, temps et moyens financiers) et résoudre les problèmes.

D'un point de vue artistique, nous avons pris conscience que dans une installation interactive la perception de l'environnement virtuel par l'utilisateur dépend de la façon dont il rentre en contact avec ce monde 3D. Dans notre cas l'objet usuel qu'est la camera rend plus instinctif le rapport entre l'homme et la synthèse.

Enfin, sur le plan humain ce fut la découverte de la gestion de projet et du travail d'équipe sur le long terme. D'ailleurs, face aux difficultés la concertation et la cohésion au sein du groupe ont été un atout important.

6.2 Avenir du Projet

Notre idée de départ était de faire une application destinée aux professionnels de l'audiovisuel et du cinéma. Mais il faut bien remarquer que dans le temps imparti et avec une équipe de seulement trois personnes, nous n'avons pu faire qu'une version moins ambitieuse et plus ludique.

A Laval Virtuel le projet (qui n'était alors qu'en version prototype) a rencontré un grand enthousiasme. Notamment, nous avons reçu l'aide de Virtools et d'Immersion (pour les problèmes de capteur), et avons pris contact avec un responsable du Futuroscope et un organisateur du Siggraph.

Actuellement, nous sommes en relation avec le Futuroscope pour éventuellement établir un partenariat technique concernant l'application SCP cam.

Prochainement, nous allons présenter le projet au 3D3, à la Villette Numérique, et peut-être au Lab des e-magiciens.

A plus long terme, nous comptons monter un dossier pour participer au SIGGRAPH 2007 et espérons aussi pouvoir exposé cette installation au centre des Arts d'Enghien les Bains.

Annexe A

Scripts

Curve to Locator (MEL)

```
// curve2locator.mel © Xavier Gouchet
global proc curve2locator(){
|    $sel = `ls -sl` ;
|    for ($curve in $sel){
|    |    // verifier que les objets selectionnés sont bien des courbes
|    |    $relatives = `listRelatives -s $curve` ;
|    |    if (!(`objectType -i nurbsCurve $relatives[0]`)){
|    |    |    error ($curve+" is not a nurbs curve") ;
|    |    }
|    |    // ensuite créer un groupe qui continedra les locators créés
|    |    group -em ;
|    |    rename ($curve +"Group") ;
|    |    int $spans = `eval("getAttr "+$curve+".spans")` ;
|    |    // pour chaque edit point, on crée un locator
|    |    for ($i=0 ;$i<=$spans ;$i++){
|    |    |    eval("select -r "+$curve+".ep["+$i+"]") ;
|    |    |    $pos = `xform -q -ws -t` ;
|    |    |    spaceLocator -p 0 0 0 -n ($curve+"_Point"+$i) ;
|    |    |    move -a $pos[0] $pos[1] $pos[2] ;
|    |    |    select -add ($curve +"Group") ;
|    |    |    parent ;
|    |    }
|    }
|    select -cl ;
|    print ("// Locators created succesfully \n") ;
}
```

Frame to Curve (VSL)

```
// Frame2curve.vsl © Xavier Gouchet
// Les paramètres pour convertir les frames en courbes
extern String curveName;
extern bool linear;
extern bool closed;
extern bool deleteEntities;

void main(){
|    int count = ac.selection.Size();
|    // on crée d'abord la courbe
|    Curve theNewCurve = ac.CreateCurve(curveName.Str(),false,true);
|    for (int i=0; i<count; ++i) {
|    |    // pour chaque frame selectionné, on recupere sa position
|    |    // puis on place un point sur la courbe à cet endroit
|    |    Entity3D ent = Entity3D.Cast(ac.selection[i]);
|    |    if (ent) {
|    |    |    Vector curPos;
|    |    |    ent.GetPosition(curPos);
|    |    |    CurvePoint curPoint = ac.CreateCurvePoint("curvpoint",false,true);
|    |    |    curPoint.SetPosition(curPos);
|    |    |    // definit si la courbe est linéaire ou bezier
|    |    |    curPoint.SetLinear(linear);
|    |    |    theNewCurve.AddControlPoint(curPoint);
|    |    |    if (deleteEntities){
|    |    |    |    ac.DestroyObject(ent);
|    |    |    }
|    |    }
|    }
|    theNewCurve.CreateLineMesh();
|    if (closed){
|    |    theNewCurve.Close();
|    }
}
```

MakePlaces (VSL)

```
// MakePlaces.vsl © Xavier Gouchet
extern String PlaceFlag;
extern Place place;

void main(){
|    int count = ac.selection.Size();
|    place.SetName(PlaceFlag.Str());
|    for (int i=0;i<count;i++){
|    |    Entity3D ent = Entity3D.Cast(ac.selection[i]);
|    |    if (ent) {
|    |    |    String name = ent.GetName();
|    |    |    BaseString bsFlag = PlaceFlag;
|    |    |    if (name.Contains(castStencil)){
|    |    |    |    place.AddChild(ent,true);
|    |    |    }
|    |    }
|    }
}
```

MakeBothSided (VSL)

```
// MakeBothSided.vsl © Xavier Gouchet
extern bool bothSided;
void main(){
|    int count = ac.selection.Size();
|    for (int i=0;i<count;i++){
|    |    Entity3D ent = Entity3D.Cast(ac.selection[i]);
|    |    if (ent) {
|    |    |    String name = ent.GetName();
|    |    |    String CastS = "BS_";
|    |    |    BaseString castStencil = CastS;
|    |    |    if (name.Contains(castStencil)){
|    |    |    |    Mesh mes = ent.GetCurrentMesh();
|    |    |    |    Material mat;
|    |    |    |    int matCount = mes.GetMaterialCount();
|    |    |    |    for (int j=0;j<matCount;j++){
|    |    |    |    |    mat = mes.GetMaterial(j);
|    |    |    |    |    mat.SetTwoSided(bothSided);
|    |    |    |    }
|    |    |    }
|    |    }
|    }
}
```

CrowdCreator (VSL)

```
// CrowdCreator.vsl © Xavier Gouchet
extern Character perso;
extern Color pull;
extern Color pantalon;
extern Color chaussure;
extern Material materiauOeil;
extern Texture dirtmap;
extern String nom;
void main(){
|    //  creation des variables de base
|    String PullM = "perso_pull";
|    BaseString pullBase = PullM;
|    String PantalonM = "perso_pantalon";
|    BaseString pantBase = PantalonM;
|    String ChaussureM = "perso_chaussures";
|    BaseString chauBase = ChaussureM;
|    String YeuxM = "perso_yeux";
|    BaseString yeuxBase = YeuxM;
|    //  copie du character selectionné
|    Character new = Character.Cast(ac.Copy(perso,false,true));
|    new.SetName(nom.Str());
|    int partCount = new.GetBodyPartCount();
|    //  pour chaque partie du character on récupère un mesh
|    for (int h=0;h<partCount;h++){
|    |    BodyPart curPart = new.GetBodyPart(h);
|    |    int meshCount = curPart.GetMeshCount();
|    |    //  pour chaque mesh recupéré on récupère un materiau
|    |    for (int i=0;i<meshCount;i++){
|    |    |    Mesh curMesh = curPart.GetMesh(i);
|    |    |    if (curMesh){
|    |    |    |    int materialCount = curMesh.GetMaterialCount();
|    |    |    |    //  pour chaque materiau on opere les changements adéquats
|    |    |    |    for (int j=0;j<materialCount;j++){
|    |    |    |    |    Material curMat = curMesh.GetMaterial(j);
|    |    |    |    |    Texture curTex = curMat.GetTexture(0);
|    |    |    |    |    if (curTex) curMat.SetTexture(dirtmap);
|    |    |    |    |    else {
|    |    |    |    |    |    String name = curMat.GetName();
|    |    |    |    |    |    if (name.Contains(pullBase))
|    |    |    |    |    |    |    curMat.SetDiffuse(pull);
|    |    |    |    |    |    if (name.Contains(pantBase))
|    |    |    |    |    |    |    curMat.SetDiffuse(pantalon);
|    |    |    |    |    |    if (name.Contains(chauBase))
|    |    |    |    |    |    |    curMat.SetDiffuse(chaussure);
|    |    |    |    |    |    if (name.Contains(yeuxBase))
|    |    |    |    |    |    |    curMesh.ApplyGlobalMaterial(materiauOeil);
|    |    |    |    |    }
|    |    |    |    }
|    |    |    }
|    |    }
|    }
}
```

Annexe B

Mode d'emploi du programme Virtools

Le programme '*SCP Camera*' se décompose en trois parties : Shoot, Cut et Play (enregistrement, montage et prévisualisation). Il permet de créer et monter rapidement et intuitivement des mouvements de caméra réutilisables dans une scène 3D pour le temps-réel (Virtools) ou une scène d'animation (Maya).

L'interface

L'interface principale *(cf Fig. B.1 p.61)* se compose de :

1. La boîte à outils
2. La timeline
3. Le code temporel

La Timeline

La timeline, située en bas de l'écran, permet de visualiser les clips (mouvements enregistrés). Chaque clip possède un numéro et une couleur permettant de le distinguer des autres clips.

Au dessus de la timeline se trouve la barre de défilement, permettant de déplacer l'ensemble des clips vers la droite ou la gauche lorsque ceux-ci ne sont pas tous visible à l'ecran. De plus, les touches [Page Up] et [Page Down] permettent de faire un zomm avant ou arriere sur la timeline, notamment pour éditer plus facilement des clips trop longs ou trop courts, et avoir plus de précision sur l'édition des clips. *(cf Fig. B.2 p.61)*

La boite à outils

Le bouton 1 (Options) *(cf Fig. B.3 p.61)* permet d'ouvrir le menu des options.

Les boutons 2, 3 et 4 permettent de changer le mode dans lequel on se trouve (respectivement Shoot, Cut et Play).

Les boutons 5, 6, 7 et 8 permettent d'effectuer des actions selon les modes dans lequel on se trouve.

Le mode Shoot

Le mode Shoot (Enregistrer) sert à enregistrer des mouvements de caméra. Lorsque la scène contient des personnages animés, on utilise les boutons 6 (Lancer l'animation des personnages) et 7 (Arrêter les animations) pour contrôler ces animations. Cela correspond au "Action" lors d'un tournage réel. Tant qu'il n'y a pas d'enregistrement de mouvement, l'action continue en boucle. Par contre, lorsqu'on arrive à la fin de l'animation pendant un enregistrement, ce dernier s'arrête automatiquement.

Le bouton 5 (Enregistrer) permet de commencer/arrêter l'enregistrement d'un mouvement de camera.

Le mode Cut

Le mode Cut (Montage) permet d'éditer et gérer les clips enregistrés.

Le bouton 5 (Razor) permet de couper un clip en deux. Cela a pour effet de dupliquer le clip choisi, et d'en modifier les bornes.

Le bouton 6 (Drag n Drop) permet de modifier l'ordre des clips en glissant les clips les uns par rapport aux autres.

Le bouton 7 (Crop) permet de modifier les bornes d'un clip. Cela ne modifie en aucun cas la vitesse de lecture de l'enregistrement, mais specifie juste quelle partie du clip sera prise en compte lors de la lecture et de l'export.

Le bouton 8 (Delete) permet de supprimer un clip.

Le mode Play

Le mode Play (Lecture) permet de prévisualiser le montage des enregistrements de mouvement de camera.

Le bouton 5 (Play/Pause) lance et met en pause la lecture. Le bouton 6 (Stop) arrête la lecture.

Les options

Le menu d'option permet de configurer le programme et d'en utiliser toutes les ressources.

Fichier

Cette partie permet de sauvegarder ou ouvrir un montage effectué avec le programme '*SCP Camera*'.

Nouveau : supprime tous les clips et le montage en cours afin de refaire des mouvements dans une scène vierge.

Enregistrer : Enregistre les données du montage, les fichiers temporaires et exporte le montage au format scp (utilisable sous Maya). Le programme crée deux fichiers (un pour l'export, un pour le montage), plus autant de fichiers que de clips, ces fichiers contenant l'animation complète correspondant au clip (indépendant des opérations de montage).

Exporter : Exporte le montage courant, sans l'enregistrer ni enregistrer les fichiers temporaires

Ouvrir : Ouvre un montage sauvegardé précédemment (nécessite que les fichiers correspondant à chaques clips soient toujours au même endroit).

Informations

Cette partie concerne les informations sur la scène courante.

Frame Rate : définie le nombre d'images par seconde de la scène. Attention ne pas changer ces informations entre deux enregistrement de mouvement. Le résultat donnerai des mouvements non homogènes une fois récupérés sous Maya.

Nom du plan : nom ou identifiant du plan correspondant aux mouvements / à la scène courante. Utilisé lors de l'import sous maya pour créer les fichiers Batch.

Commentaires : commentaires destinés à donner des informations supplémentaires sur le plan ou le montage.

Dossier temporaire : Dossier ou sont stockés les fichiers contenant les données de tous les mouvements enregistrés. Ces fichiers sont écrasés à chaque réouverture du programme '*SCP Camera*' et ne constituent en aucun cas des fichiers de sauvegarde. Pour conserver ces fichiers et pouvoir en faire un montage, utiliser la fonction enregistrer.

Données

Cette partie concerne le contenu de la scène dans laquelle les mouvements sont faits.

Importer Scene : permet d'importer un décor (ou des objets servant de référence) au format nmo.

Importer Animation : permet d'importer un personnage ayant une animation. Le personnage doit être enregistré "As Character", et doit contenir son animation dans le même fichier.

Vider la scène : supprime tous les décors et les personnages importés.

Affichage

Permet de modifier l'affichage de l'interface.

Afficher temps : affiche / cache les codes temporels (en haut de l'ecran).

Temps en Frame/Minutes - Secondes : Affiche les codes en frames ou en minutes secondes et millisecondes.

Gate 4/3 : affiche le cadre correspondant à la zone d'action au format 4/3.

Gate 16/9 : affiche le cadre correspondant à la zone d'action au format 16/9.

Divers

Cette partie contient les options n'entrant dans aucune autre catégorie.

Profondeur de Champ : affiche ou nom la profondeur de champ.

Langue : change les textes de l'interface en français et en anglais.

FIG. B.1 – L'interface

FIG. B.2 – La Timeline

FIG. B.3 – La boite à outil

Annexe C

Remerciements

A la suite de ce projet, nous tenons à remercier :

L'équipe enseignante d'A.T.I et tout particulièrement :
- Antoine Prévost, pour ses conseils pertinents et pour nous avoir très vite mis sur la bonne voie pour le développement du projet
- Marlène Puccini, pour ses conseils artistiques et son ouverture d'esprit.
- Cédric Plésier, pour ses conseils techniques, son soutien et ses contributions.

A tout ceux qui nous ont aidé et soutenu sur le salon de Laval Virtual, en particulier :
- la société VHP Maintenance qui nous a gracieusement fournie le boîtier de notre caméra.
- La société Virtools et David Nahon pour l'intérêt porté au projet et leur aide précieuse de dernière minute
- La société Immersion pour nous avoir permis d'utiliser leur système de tracking infrarouge.
- Le Futuroscope pour leur remarque dont nous avons tenue compte.
- Thanks Peter !

A ceux aussi qu'on aurait oublié, merci ;-)

L'ensemble de l'équipe de scp.camera

==========

scp.camera
Shoot Cut Play
scp.camera.free.fr

==========

Xavier Gouchet
Rémi Quittard
Nicolas Serikoff

Table des matières