

Troma

Rapport de soutenance 2

6 mai 2014



Thibault Dethi Deutsch (*deutsc_t*)
Rémy Shadows Bernier (*bernie_r*)
Marc Leshlague Fresne (*fresne_m*)
Anthony AnthonySG Belthier (*belthi_a*)

Table des matières

| | | |
|------|--|------|
| I. | Modélisation | V |
| 1. | La conception | V |
| 2. | La modélisation | V |
| 3. | Le normal mapping | V |
| 4. | Les animations | VII |
| 4.1 | Première étape : le “rigging” | VIII |
| 4.2 | Deuxième étape : le “skinning” | VIII |
| 4.3 | Troisième étape : la création de l'animation | IX |
| 4.4 | Exploitation des animations dans XNA | X |
| 5. | Pour la prochaine soutenance | X |
| II. | Moteur graphique | XI |
| 1. | Les différentes shader | XI |
| 2. | Ciel | XI |
| 3. | Optimisations | XI |
| 4. | Pour la prochaine soutenance | XII |
| III. | Moteur physique | XIII |
| 1. | Pour la prochaine soutenance | XIII |
| IV. | Menu | XIV |
| 1. | Écran de démarrage | XIV |
| 2. | Menu principale | XIV |
| 3. | Menu de pause | XV |
| 4. | Pour la prochaine soutenance | XV |
| V. | Réseau | XVI |

| | | |
|-------|--|-------|
| VI. | Gameplay | XVII |
| 1. | Pour la prochaine soutenance | XVII |
| VII. | Audio | XVIII |
| 1. | Les effets sonores | XVIII |
| 2. | La musique de fond | XVIII |
| VIII. | Site web | XIX |

Introduction

Ce document est le rapport de la seconde soutenance et a pour but de présenter les avancées réalisées depuis la première sur le projet Troma ainsi que les prochains objectifs de l'équipe d'Emagine Studio. Pour rappel, le jeu réalisé est un FPS (First-Person Shooter) développé en C# dont le thème principal est la seconde guerre mondiale. Celui-ci comportera à terme deux modes de jeu : un mode solo, qui sera un contre la montre, et un mode multi-joueurs. La répartition des tâches de la seconde soutenance est présentée dans le tableau ci-dessous.

| Tâches | Thibault | Rémy | Marc | Anthony |
|------------------|----------|------|------|---------|
| Modélisation | | | X | |
| Moteur graphique | X | | X | |
| Moteur physique | | X | | X |
| Menu | | X | | X |
| Réseau | X | | X | X |
| Gameplay | X | X | | |
| Audio | | X | | X |
| Site web | X | X | | |

FIGURE 1 – Répartition des tâches de la deuxième soutenance

Légende :

- Rouge : Commencé
- Orange : Avancé
- Vert : Terminé

Lors de la dernière soutenance nous vous avions présenté le jeu au tout début de son développement. Celui ne comportait pas de mode jeu a proprement parlé mais uniquement une première carte sur laquelle il était possible de se déplacer. Quelques modélisations étaient présentes et nous avions implémenté un menu basique.

Pour cette seconde soutenance nous souhaitons présenter les améliorations apportés au moteur graphique ainsi qu'au moteur physique de notre jeu. Nous avons aussi amélioré le menu et créer le mode de jeu solo.

Dans la suite du rapport vous trouverez des explications détaillées à propos du travail fournit par chacun des membres du groupe, des avancées réalisées mais aussi des difficultés rencontrées. Les conclusions de chaque partie comporteront une liste de prochains objectifs à remplir.

I. Modélisation

1. La conception

Après la soutenance 1 nous avions déjà une première carte destinée au mode solo du jeu. Cependant nous étions limités au niveau de la taille de celle-ci. En effet comme expliqué dans le rapport précédent nous avions utilisé une heightmap pour la gestion du sol et du relief. L'implémentation que nous avions choisie nous limitait à un carré de 513 par 513 unités. Afin d'augmenter la surface nous avons donc dans un premier temps construit un tableau à deux dimensions contenant 9 heightmap (3×3). La taille est donc passée à 1536 par 1536 unités. Ce premier changement a entraîné une modification de la carte solo de manière superficielle, mais a permis de poser les bases pour la conception de la carte multijoueur.

Le décor de la seconde carte de notre jeu (multijoueurs) a pour cadre un milieu urbain, un petit village européen. Nous nous sommes inspirés d'un quartier de Cracovie (Pologne) pour la disposition des bâtiments.

2. La modélisation

Une fois le plan conçu, Marc a pu commencer la modélisation toujours à l'aide de Blender et Gimp. Dans l'ensemble les éléments des décors sont toujours basés sur des formes géométriques simples, cependant il dispose d'un peu plus de détails, tels que des portes, des fenêtres en relief.

Afin de faciliter la création de l'ensemble de bâtiments, la construction de la ville s'est déroulée en plusieurs étapes. Un premier temps a été consacré à la création et la mise en place sur la carte des murs des bâtiments texturés (grâce à l'UV mapping expliqué dans le rapport précédent). Un second a été l'ajout sur chaque bâtiment du toit, des fenêtres et des portes. Plusieurs modèles de fenêtres et de portes ont été réalisés au préalable (eux aussi texturés) et dupliqués sur chaque bâtiment.

A ce moment, tous les modèles étaient composés de plusieurs "meshes" (objets en français), mais chaque modèle avait besoin de plusieurs textures.

3. Le normal mapping

En parallèle à la modélisation, Thibault s'est attelé à l'implémentation d'un "shader" afin d'améliorer le rendu.

"Un shader (le mot est issu du verbe anglais to shade pris dans le sens de « nuancer ») est un programme informatique, utilisé en image de synthèse, pour paramétriser une partie du processus de rendu réalisé par une carte graphique ou un moteur de rendu logiciel.", *Wikipédia*

La technique de normal mapping ressemble à la technique de l'heightmap que nous vous avions présenté lors de la soutenance 1. Cependant alors que le shader utilisé pour l'heightmap déforme la géométrie de l'objet sur lequel il est appliqué, le normal mapping feind cette déformation. En effet, la déformation n'est que visuelle dans le cas du normal mapping.

Cette technique permet sans ajouter de polygones supplémentaires aux modèles, de donner l'impression de relief. Afin de permettre cette modification, le shader se base sur une image en couleur (RVB).

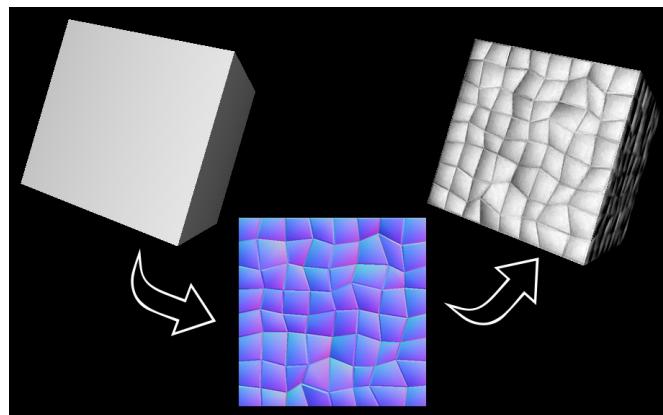


FIGURE 2 – Illustration de l'effet du normal mapping

TODO : detail explication

L'utilisation de ce shader a nécessité la réunification des textures des modèles. Pour ce faire, Marc a dû fusionner tous les meshes de chaque modèle pour modifier l'UV mapping, puis réunir les textures en une seule en suivant la carte UV.

Ensuite est arrivé l'étape de création des normal map. La technique utilisé habituellement consiste à réalisé deux modèles. Un dit low poly (composé de peu de polygones) et un dit hight poly (composé de beaucoup de polygones).

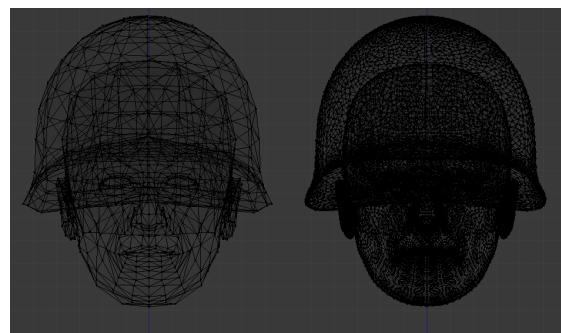


FIGURE 3 – Un mesh en low poly et un mesh en high poly

Ensuite on superpose les deux modèles puis le logiciel de modélisation est capable de générer une normal map en calculant la différence de relief en chaque point entre les deux modèles. Cette technique offre une normal map de qualité, mais est relativement longue puisqu'il est nécessaire de réaliser deux modèles dont un très détaillé.

Heureusement, il existe une autre solution. A partir d'une texture il est possible grâce à un logiciel de retouche d'image, de générer en fonction des couleurs et de la luminosité, une normal map. Cette technique permet de réaliser rapidement une normal map si la texture de l'objet a déjà été réalisée, et les résultats sont plutôt convaincants. Nous avons donc opté pour la deuxième technique, moins chronophage.

N.B. Une autre technique permet l'illusion de relief et utilise aussi un shader : le Bump mapping. Cependant, l'image utilisé pour le Bump mapping, a l'instar de celle utilisé pour l'heightmap est en noir et blanc. La modification du relief ne s'effectue que sur un axe (la normale à la surface concernée), le rendu est donc moins détaillé, c'est pourquoi nous opté pour le normal mapping.

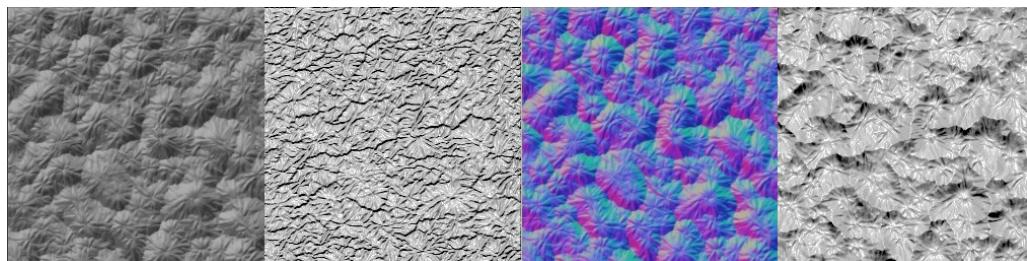


FIGURE 4 – A gauche le Bump Mapping, à droite le Normal Mapping

4. Les animations

Jusqu'à la première soutenance, les animations 3D n'avait pas fait leurs apparitions dans notre jeu.

Pour cette deuxième soutenance, quelques animations sont visibles. La première est celle des cibles, qui se "lèvent" comme si un ennemi apparaissait et qui se "couchent" lorsque le joueur leur tire dessus. Les suivantes concernent les personnages en lui-même, les mouvements tels que la marche ou la mise accroupie.

Pour obtenir les animations, deux techniques sont possibles. Soit réaliser le modèle et "programmer" manuellement les mouvements des différents meshes depuis Visual Studio. Soit réaliser les animations depuis blender et les exportés pour les "lire" ensuite.

Par commodité Marc a choisi d'utilisé la seconde méthode. La création d'une animation en utilisant des "bones" s'effectue en trois étapes : le "rigging", le "skinning", et enfin l'enregistrement de l'animation.

Tout d'abord, pourquoi utiliser une armature composée de “bones” ? Cela permet d'animer des modèles complexes, le personnage par exemple, en modifiant le maillage en fonction du mouvement.

4.1 Première étape : le “rigging”

Le rigging consiste à créer une armature, un squelette, et à organiser une hiérarchie entre les os. Il s'agit de disposer les os de façon à être capable d'effectuer les mouvements désirés et d'indiquer la dépendance des os entre eux.

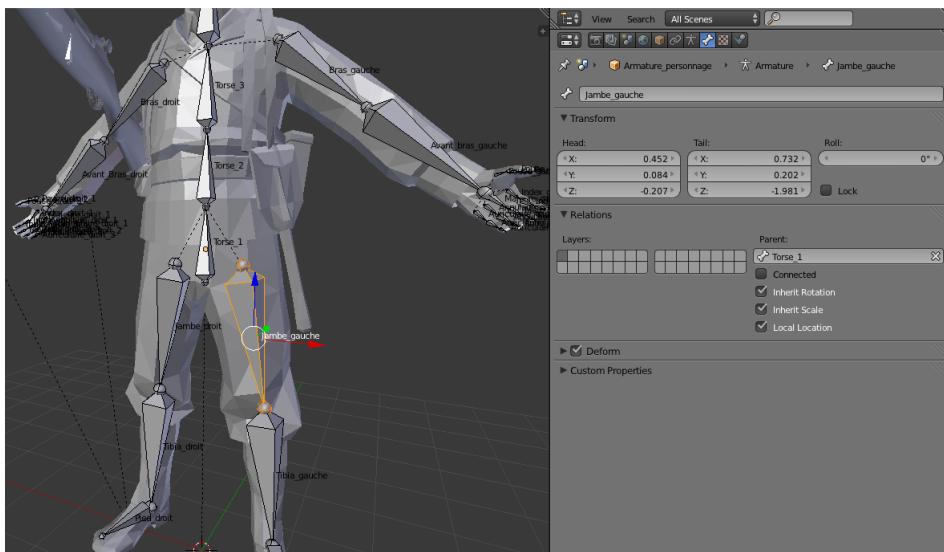


FIGURE 5 – Illustration du “rigging”

Un os peut avoir un parent ou non, et peut lui-même être père ou non. Si l'os est père d'un autre alors le mouvement du père entraîne le mouvement du fils.

4.2 Deuxième étape : le “skinning”

Le skinning consiste à affecter à chaque os les points du maillage sur lesquels il aura une influence, c'est la notion de weight (le poids d'influence). La valeur de ce poids est comprise entre 0 et 1, 0 étant l'influence nulle et 1 l'influence maximale.

Blender offre plusieurs possibilités sur ce point :

- Soit affecter automatiquement les points du maillage aux os en fonction de leur “envelope”, c'est à dire leur rayon d'action.
- Soit affecter automatiquement les points du maillage aux os en fonction de leur position par rapport aux os.
- Soit manuellement. Là encore deux solutions sont possibles :

- Sélection de points et affectation à un «vertex group», comprendre groupe de point influencé par l'os
- Selection par « Weight Painting », c'est le même principe sauf qu'au lieu de sélectionner les points du maillage, il suffit de « peindre » la surface qui dépendra de l'os.

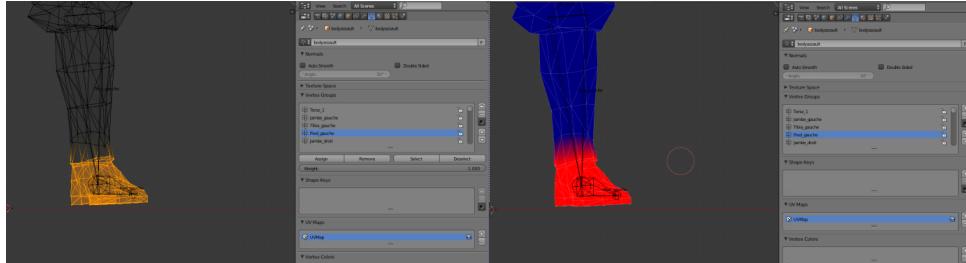


FIGURE 6 – Skinning selection maillage à gauche, Weight Painting à droite

Les deux premières méthodes de skinning ne sont à utiliser que pour des modèles très simples. Marc n'a donc pas pu les utiliser.

N.B. Il est très important que tous les points dépendent d'au moins un os. Une solution pour vérifier que tous les points ont un poids et de parcourir les “vertex groups” en mode Weight Painting et de chercher une région resté bleue tout le long du parcours.

4.3 Troisième étape : la création de l'animation

Une fois le squelette construit et les dépendances avec le maillage assignées, il faut maintenant définir les mouvements qui composeront l'animation.

Pour modifier le positionnement des os et que cela ait une influence sur le maillage il faut se placer en “pose mode”. Blender permet de diviser l'écran en plusieurs zones pour pouvoir travailler sur différents aspects en même temps. Par défaut sous la vue 3D il y a une zone où le mode “TimeLine” est activé. Il s'agit d'un axe représentant le temps sur lequel on peut se déplacer, et servant à indiquer les différents paramètres du modèle en fonction du temps.

C'est grâce à cette “TimeLine” que l'on réalise une animation. Il suffit de se placer à un instant t et d'enregistrer les informations voulues pour les liées. Ces instants sont appelées KeyFrames. Ainsi pour l'animation de la cible, à l'instant t1 la cible est relevée, à l'instant t2 la cible est baissée.

Grace à Blender ces deux instants suffisent pour que la cible se baisse et passe de la position-rotation de l'instant t1 à celle de l'instant t2 de manière progressive durant le temps qui sépare ces deux instants. En effet celui-ci va calculer les différents paramètres à appliquer au modèle à chaque instant pour que l'animation soit fluide.

Une fois l'action créée, il n'y a plus qu'à exporter le modèle.

4.4 Exploitation des animations dans XNA

Une fois le modèle et l'animation créés et exportés, il va falloir récupérer ces informations pour être capable de lancer l'animation dans XNA. Toutes les informations concernant le modèle sont conservées sous forme de matrice (ex position des points) ou de tableau (ex poids des points, os). Par default le ModelProcessor d'XNA pour les fichiers .X ne récupère pas d'information concernant les animations. Il a donc fallu « l'étendre » pour récupérer les informations de skinning, les keyframes ainsi que la relation entre les bones.

Pour ce faire nous avons récupérer le sample nommé « Skinned Model » mis à disposition par Microsoft¹.

Une fois l'extension du Model Processor intégrés à notre code, nous étions en mesure de lancer des animations et de les afficher.

5. Pour la prochaine soutenance

1. http://xbox.create.msdn.com/en-US/education/catalog/sample/skinned_model

II. Moteur graphique

1. Les différentes shader

TODO : lumière, normal map, et autre ci-dessous

2. Ciel

L'environnement de notre jeu se composait lors de la première soutenance d'un terrain et de modèle. Nous souhaitions rendre le ciel plus agréable à regarder que la couleur bleue par défaut d'XNA. Notre première idée a été d'utiliser un skydome. Il s'agit d'un modèle 3d prenant la forme d'une demi-sphère sur laquelle est apposé sur la partie intérieure, une image de ciel.



FIGURE 7 – Texture d'un skydome

TODO : figure skydome

Apres quelques tests, bien que le rendu soit meilleur qu'un fond bleu uni, nous avons décidé de laisser cette idée de côté. D'une part parce que la délimitation entre le skydome et le sol posait problème. Et d'autre part car nous avions un ciel statique.

TODO : ciel dynamique

Nous avons donc fait machine arrière et retiré le skydome. Nous nous avons à la place un fond uni, auquel Thibaut a ajouté des nuages.

3. Optimisations

Afin d'optimiser une fois de plus notre moteur de jeu, nous avons réécrit nos shaders afin qu'ils soient au maximum compatible avec les preshaders. Les preshaders sont

une optimisation du compilateur HLSL qui précalcule les constantes des shaders sur le processeur. Ainsi les calculs utilisant les même valeurs pour l'ensemble des shaders ne sont calculés qu'une seule fois.

Une autre optimisation a été réalisée au niveau des shaders au niveau du passage des variables au shader. En effet, le passage de variables au shader implique un transfert des données de la mémoire vive vers la carte graphique. Ce processus fait ralentir les calculs qui doivent attendre les transfers des données. Ce processus est d'autant plus long quand il s'agit de faire passer des variables. Pour contrer ce problème, nous avons décidé de n'actualiser que les données qui varient. Ainsi les textures ne sont transférées au shader que lors de l'initialisation. Nous utilisons donc plus de mémoire graphique mais nos rendus sont accélérés.

TODO : optimisation shader + optimisation carte abandonnée

4. Pour la prochaine soutenance

III. Moteur physique

Nous avons logiquement décidé d'implémenter un système de collision entre le joueur et les objets de la carte afin de mettre un place un mode solo et multijoueur réaliste. Nous pensions y arriver directement mais suite à des difficultés nous avons décomposé cela en 2 étapes : la création puis l'affichage des bounding box

Nous créons une bounding sphère pour le personnage et une bounding box par mesh pour chaque objet présent sur la map. Les objets étant fixes, nous créons ces box une seule fois par partie lors du chargement de celle-ci. Pour ce faire nous récupérons le vecteur minimum et maximum de chaque mesh afin de pouvoir créer une box la plus proche possible des dimensions de l'objet.

Par la suite, l'affichage de ces box nous a permis de comprendre d'où venaient les problèmes et ainsi de debugger notre jeu beaucoup plus facilement.

Enfin nous avons du trouver comment faire pour pouvoir véritablement appliquer un système de collision entre ces box, c'est à dire, empêchant le joueur de rentrer dans un bâtiment par exemple. Ainsi à chaque update nous testons la bounding sphere avec toutes les box de la map et en cas de collision nous recalculons les nouvelles coordonnées du joueur à la suite de son déplacement

1. Pour la prochaine soutenance

IV. Menu

1. Écran de démarrage

Nous avons mis une vidéo au début du jeu qui permet de contrôler l'âge des joueurs qui voudront y jouer pour le rendre plus réaliste. Car, sachant que le jeu est basé sur la guerre, les armes et la destruction pendant la seconde guerre mondiale, il est préférable d'avertir les utilisateurs du contenu qui va leur être présenté. Ainsi nous nous sommes mis d'accord que l'âge requis pour y jouer est de 18 ans.

D'autres vidéos seront bien sûr rajoutées au lancement du jeu tel que l'animation de notre groupe Emagine Studio ou encore une cinématique !

2. Menu principale

Pour la soutenance 1, nous avions un menu basique avec des boutons et un fond mais il était trop statique, il manquait du mouvement pour qu'il soit plus vivant. Nous avons donc repensé le menu ce qui nous a amené à le refaire entièrement. Le menu est donc devenu dynamique pour cette deuxième soutenance. Un menu composé de boutons repensé, de balles traversant l'écran à des vitesses différentes. Tout un joli paquet de dynamisme qui rend ce menu plus vivant à regarder.

De plus nous avons supprimé le bouton jouer par deux boutons qui sont le solo et le multijoueur.



FIGURE 8 – Capture d'écran du menu de démarrage

Nous avons aussi apportés les options. Et oui nous avons implémentés des options capables de changer le langage en Anglais ou Français, d'avoir le jeu en pleine écran ou non, de changer le volume du jeu et de changer le clavier en QWERTY ou AZERTY selon la configuration de votre clavier.

Le joueur pourra ainsi définir ses propres configurations de jeu qui lui permettra de se mettre le plus à l'aise possible et de mieux se plonger dans la peau du personnage.



FIGURE 9 – Capture d'écran des options du jeu

3. Menu de pause

Nous n'avons pas tellement évolué le menu quand le jeu est en pause. Juste quelque configuration différente telle que le bouton « quitter » qui ramène au menu principal alors qu'avant il quittait directement le jeu. Cela est plus pratique dans le cadre où le joueur voudrait juste changer une configuration dans les options, il pourra donc retourner dans le menu principal au lieu de quitter le jeu. Le seul inconvénient est lorsqu'il relancera le mode solo, sa partie sera remise au début.

Nous pourrons donc rajouter l'accès au menu option directement depuis le menu pause lors de la prochaine soutenance.



FIGURE 10 – Capture d'écran du menu de pause

4. Pour la prochaine soutenance

Nous prévoyons d'implémenter le menu multijoueur afin de permettre au joueur de sélectionner une partie. De plus nous implémenterons une page “Record” récapitulant toutes les anciennes performances du joueur.

V. Réseau

Depuis la dernière soutenance, nous avons commencé l'implémentation du réseau dans notre moteur de jeu. Pour cela, il nous a d'abord fallu réécrire plus de la moitié de notre code pour le rendre compatible avec la logique de notre réseau.

Une fois cette longue opération fini, nous nous sommes longtemps documenté et nous avons fini par fixer le mode de fonctionnement de notre mode multijoueur. Premièrement, nous avons choisi d'implémenter une architecture client-serveur.

Le serveur sera chargé de recevoir l'ensemble des entrées de tout les joueurs et faire le calcul de la physique ainsi que de la logique du jeu. Une fois les calculs effectués, le serveur renverra l'ensemble des résultats à l'ensemble des joueurs.

De leur côté, les clients s'occuperont d'envoyer l'ensemble des entrées au joueur ainsi que de faire des calculs de physique approximative. Ainsi le joueur ne devrait pas être gêné par le temps de latence entre lui et le serveur. Les résultat des calculs effectués par le serveur permettront au client de corriger les approximation afin de rester un maximum synchronisé avec le serveur. C'est ce qu'on appelle la technique du "client side prediction". Si le client se retrouve totalement désynchroniser par rapport au serveur, par exemple après une perte de paquet trop importante, le serveur utilisera sa temporisation des calculs pour envoyer une commande au joueur avec la position à laquelle il doit se positionner pour être de nouveau synchronisé.

Bien entendu pour réaliser l'ensemble de ces concepts, nous nous baserons sur le protocole UDP, "user datagram protocol". Il s'agit d'un protocole de transport de la pile de protocole TCP/IP qui fonctionne sans connexion préalable, par rapport au TCP. Ainsi le protocole UDP ne garantie pas la bonne transmission des données, ni l'ordre d'arrivée. Cependant ce protocole à l'avantage d'être le plus rapide. En effet si un paquet est perdu, le client n'est pas obligé d'attendre que le serveur le renvoi, il recevra directement les paquets suivants. Ainsi il n'y pas de blocage du flux de données par rapport au TCP qui maintient un flux de donnée ordonné dans l'ordre de transmission.

Au niveau de l'implémentation concrète, nous avons choisi d'utiliser la bibliothèque Lidgren². Il s'agit d'une bibliothèque réseau réputé et conçu en C#.

Actuellement, nous avons presque fini l'implémentation du client mais nous rencontrons diverses difficultés avec la transmission de donnée et l'implémentation du serveur. C'est pourquoi le multijoueur n'est actuellement pas opérationnelle.

2. <https://code.google.com/p/lidgren-network-gen3/>

VI. Gameplay

1. Pour la prochaine soutenance

VII. Audio

Nous avons recodé le gestionnaire du son que ce soit pour la musique dans les menus ou les effets sonores car il ne suffisait plus par rapport au reste du projet.

1. Les effets sonores

Nous avons rajouté des effets sonores dans les menus pour donner un peu plus de vie dans ceux-ci. Les effets s'appliquent sur les boutons. Quand l'utilisateur utilise les touches directionnelles pour se déplacer entre les boutons, un effet sonore s'applique. De même quand l'utilisateur appuie sur un bouton (par exemple il veut changer la configuration du clavier, quand il va appuyer sur le bouton option, un effet sonore se fera entendre).

Concernant le jeu, nous avons implémenté les bruits de pas pour donner encore plus de réalisme au gameplay. Celui-ci diffère selon l'action, qui est soit de marcher, soit de courir.

Les nouveaux effets sonores ont été pris sur le même site web que les autres qui est <http://www.freesound.org/>. Il s'agit d'une banque de sons libre et gratuite.

2. La musique de fond

Nous avons trouvé une nouvelle musique pour le menu. Celle-ci est une musique de l'époque de la seconde guerre mondiale. Cela permettra au joueur de se mettre tout de suite dans la peau du personnage et dans le style de l'époque. Nous avons eu beaucoup de problème comme la musique qui se lançait à chaque fois que l'on appuyait sur la touche « entré » du clavier. Mais nous avons trouvé d'où venait le problème et nous avons pu le résoudre. Cette musique pourra être changé grâce à l'option volume.

VIII. Site web

Nous avons inclue une branche « actualité » qui permet de tenir au courant de l'avancer du projet pour les personnes qui nous suivent. Cette actualité nous sert de petit blog. Nous avons aussi mis à jour le site en changeant les anciennes photos du jeu par les nouvelles. Et nous avons actualisé la partie progression du site internet pour que les internautes voient où on se situe par rapport à notre jeu fini.

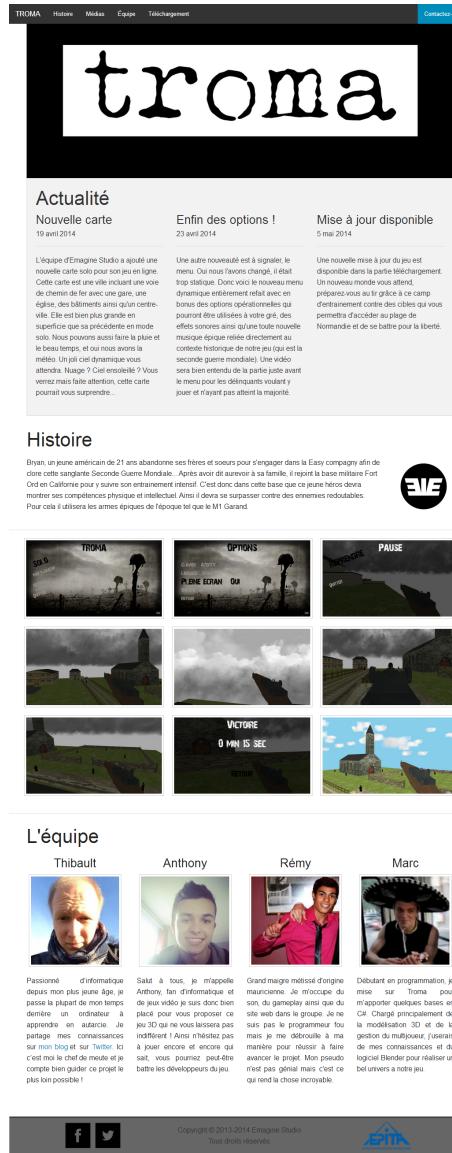


FIGURE 11 – Notre site web, le 05/05/14

Conclusion

Nous sommes une nouvelle fois à jour par rapport à tout ce qui était prévu. Nous avons, comme expliqué précédemment pris le temps de recoder la quasi totalité du projet afin de nous préserver de mauvaises surprises quant à l'implémentation du mode multijoueur.

Nous avons également améliorer ce qui était déjà présent à la dernière soutenance comme le menu, les sons et autres en nous servant de nos nouvelles connaissances fraîchement apprises !

Le mode solo est lui terminé, notre projet est d'ores et déjà jouable, cependant nous allons nous servir de cela comme tremplin pour le multijoueur et donc pour le jeu final. Le but étant une nouvelle fois d'améliorer ce que nous avons en y ajoutant en plus tous les éléments prévus pour le rendu final.

Nous souhaitons donc, pour la soutenance finale, vous présenter un jeu complètement fini, sans aucun bug, et qui propose une expérience de jeu de qualité. Nous voulons que les personnes prennent du plaisir à jouer à notre jeu !



FIGURE 12 – Capture d'écran du jeu pour la deuxième soutenance

Table des figures

| | | |
|----|--|------|
| 1 | Répartition des tâches de la deuxième soutenance | IV |
| 2 | Illustration de l'effet du normal mapping | VI |
| 3 | Un mesh en low poly puis en high poly | VI |
| 4 | A gauche le Bump Mapping, à droite le Normal Mapping | VII |
| 5 | Illustration du “rigging” | VIII |
| 6 | Skinning selection maillage à gauche, Weight Painting à droite | IX |
| 7 | Texture d'un skydome | XI |
| 8 | Capture d'écran du menu de démarrage | XIV |
| 9 | Capture d'écran des options du jeu | XV |
| 10 | Capture d'écran du menu de pause | XV |
| 11 | Notre site web, le 05/05/14 | XIX |
| 12 | Capture d'écran du jeu pour la deuxième soutenance | XX |