Table des matières

Introduction

Voila plus de quatre mois que nous avons débuté nos études à EPITA et nous sommes déjà à la première soutenance de notre projet. Notre projet est de construire un fps en 3D dans l'univers de South Park, il y donc énormément à faire ! Heureusement, nous sommes une équipe soudée et très motivée, nous sommes armés de suffisamment de patience pour aller au bout de notre jeu. Pour avoir un jeu plus complet, nous avons donc choisis d'utiliser Xna.

Nous avons consacré une bonne partie de nos vacances à la réalisation de ce projet, nous nous sommes rendu compte de la complexité de la chose. Nous allons pouvoir vous expliquer dans ce rapport de soutenance les difficultés que nous avons rencontrées et celles que nous avons pu surmontées. Nous avons beaucoup à apprendre, et nous sommes pressés de voir ce jeu aboutir .

Nous allons vous présenter ce rapport de première soutenance en quatre parties, d'abord le moteur graphique, puis le moteur physique, ensuite nous verrons nos modélisations 3D et pour finir un aperçu de notre site.

Chapitre 1

Moteur Graphique(Basile et Rémi)

1.1 XNA

Pourquoi XNA ? La création d'un jeu en 3D est évidemment ambitieuse. De ce fait, il nous fallait un outil puissant, adapté, XNA apparu alors comme une évidence à nos yeux. Ainsi, nous pouvons utiliser les classes et leurs méthodes fournies par XNA pour gérer le moteur graphique, c'est-à-dire l'affichage des modèles 3D crées sur Maya, des modèles 2D, des polices, des textures et plus tard des animations.

1.2 Affichage

Comme nous l'avons dit, XNA nous permet de tout afficher, comme par exemple la map que Théo et Basile ont modélisé. Pour cela nous parcourons la collection de meshs du modèle :

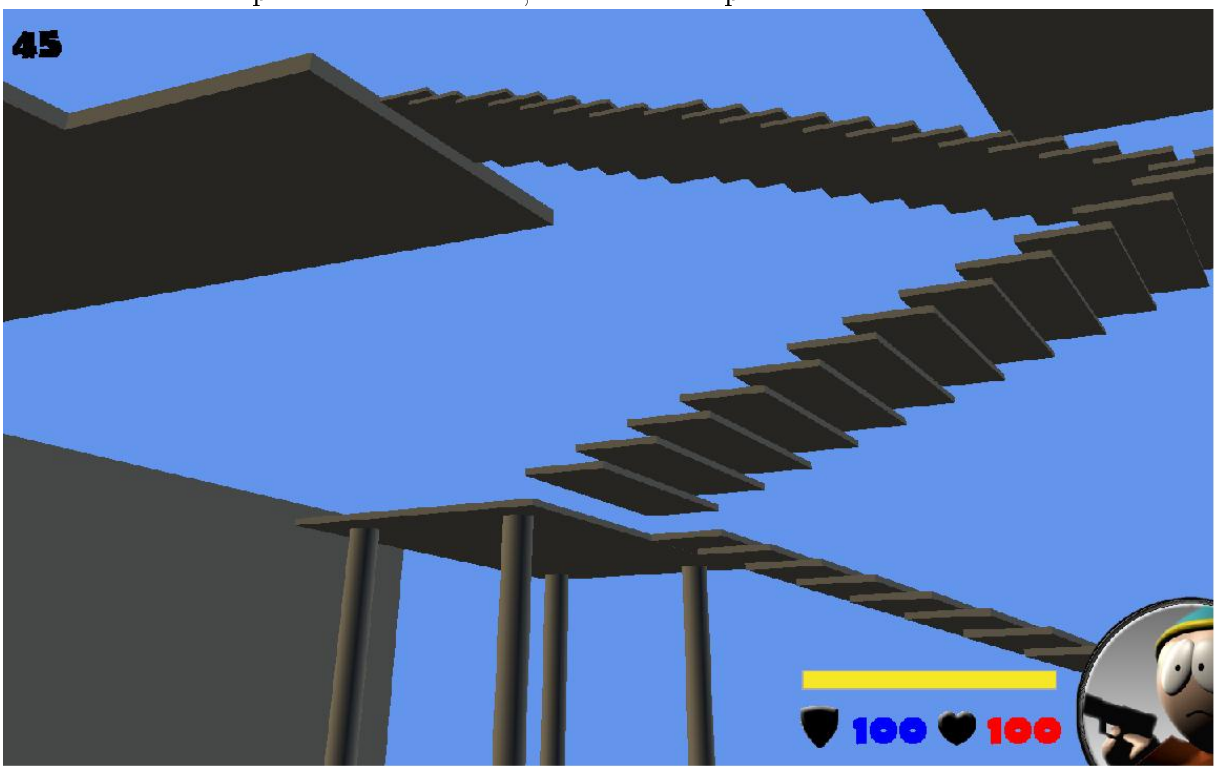
```
foreach (ModelMesh mesh in model.Meshes)
```

et après, à l'aide de 3 matrices(une de vue, une pour le monde et une de projection) on affiche le modèle à l'écran :

- La matrice de vue permet de voir, de regarder un point :
  ```
  viewMatrix = Matrix.CreateLookAt(cameraPosition, finalTarget,
   rotatedUpVector);
  ```
- La matrice de projection permet de rendre la 3D visible sur un écran 2D :
  ```
  Matrix.CreatePerspectiveFieldOfView(MathHelper.PiOver4,
  aspectRatio, 0.001f, 1000f, out projection);
  ```

– La matrice du monde, qui est utilisée pour situer le modèle dans le monde par rapport à l'origine.

La réalisation d'un HUB est aussi faite à l'aide des méthodes d'XNA, il est seulement composé de modèles 2D, et de chiffres pour la vie et l'armure.



On peut remarquer ici la présence d'un compteur de fps, qui permet d'avoir en permanence le nombre d'images par secondes et ainsi de voir si notre affichage est bien optimisé.
PS : La synchronisation verticale est activée, ce qui permet d'éviter le "screen tearing", fortement visible lorsque celle-ci est désactivée.

Chapitre 2

Moteur Physique (Robin et Théo)

2.1 Déplacement de la caméra

La caméra est à la première personne et on peut la déplacer à l'aide de la souris et du clavier.

- Le clavier permet de faire avancer, reculer, bouger vers la gauche ou vers la droite (comme dans un fps normal) la caméra, donc ceci est gérer simplement avec un vecteur de déplacement *moveVector* qui est ajouté, pour l'instant, à un vecteur gravité et multiplié par la vitesse de déplacement du joueur, différente selon si le joueur court ou pas. Le vecteur obtenu *vector* est "transformé" par une matrice de rotation *cameraRotation* (utilisée pour les déplacements de la souris) qui permet une rotation uniquement autour de l'axe des ordonnées (en effet, on ne peut pas voler) et est ensuite ajouté à la position de la caméra :

  ```
  Matrix cameraRotation = Matrix.CreateRotationY(RLrot);
  Vector3 rotatedVector = Vector3.Transform(vector, cameraRotation);
  cameraPosition += moveSpeed * rotatedVector;
  ```

  On met ensuite à jour la matrice de vue :

  ```
  UpdateViewMatrix();
  ```

- La souris effectue une rotation de la caméra autour de l'axe des ordonnées et autour de l'axe des abscisses. La position du pointeur se trouve dès le départ au milieu de l'écran :

  ```
  Mouse.SetPosition(graphics.GraphicsDevice.Viewport.Width / 2,
  graphics.GraphicsDevice.Viewport.Height / 2);
  ```

  et on modifie la vue si et seulement si la position actuelle du pointeur est différente de la position initiale, en modifiant les angles de rotation

selon la différence entre la position de la souris et sa position initiale :

```
RLrot -= rotationSpeed * mouseX * time;
UDrot -= rotationSpeed * mouseY * time;
```

et en remettant le pointeur de la souris au milieu de l'écran, comme ci-dessus.

2.2 Collisions

Nous n'avons pas encore débuté la gestion des collisions, seul la limite de la map est gérée, si on dépasse celle-ci, la caméra tombe sous l'effet de la gravité. Ces limites sont pour l'instant uniques, c'est-à-dire qu'elles ne marchent que pour une seul map que nous avons modéliser. Nous stockerons plus tard ces limites dans un fichier à part.

Chapitre 3

Modélisation 3D (Théo et Basile)

C'est une étape qui peut paraitre futile mais nous avons pour ambition d'élargir notre culture personnelle en apprenant les bases de la modélisation 3D et de l'animation objet. En ayant conscience de la dose de travail très importante que cela peut représenter, nous avons décidé de s'y attaquer avant même la première soutenance afin de mieux cerner le temps qui nous serait nécessaire pour l'élaboration des nos objets.

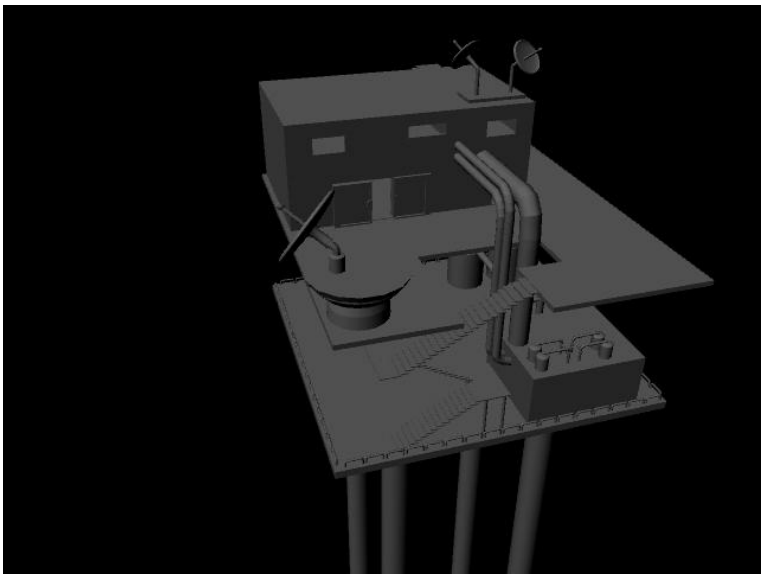
Maya est un logiciel très complexe du au nombre d'options qui sont a disposition, mais contrairement à ce que l'on pourrait croire il est très ergonomique et intuitif d'utilisation, une fois que l'on a bien différencié les grandes parties. La seule qualité requise en modélisation 3D est une très grande rigueur (qui nous fit malheureusement default lors de nos premières esquisses, nous obligeant à recommencer plusieurs travaux) afin d'éviter que certains points, arrêtes, faces ne se superposent.

Il faut savoir qu'il existe trois grands types de modélisation : modélisation polygonale, par NURBS et par surfaces. Nous avons choisi d'utiliser principalement les polygones, qui sont beaucoup plus simple d'utilisation et réduisent le nombre d'outils nécessaire pour réaliser n'importe quelle forme à une petite dizaine. L'inconvénient de cette méthode est l'absence de formes arrondies (lesquelles sont créées grâce à la modélisation par surface) ; il faut générer un nombre important de points, d'arrêtes et faces pour arriver à quelque chose qui ressemble de loin à une sphère, ce qui augmente la complexité des objets du même coup.

Notre première erreur fut d'essayer de modéliser entièrement une carte, avec des bâtiments et autres objets, mais nous nous sommes rendus compte plus tard des problèmes que cela impliquent au niveau de la gestion des colli-

sions. Nous avons donc décidé de modéliser les objets un par un pour ensuite les placer sur notre carte a travers le code.

Voici quelques unes de nos créations :



Chapitre 4

Site (Robin et Rémi)

Nous avons commencé à faire le site web dès le début de l'année car ceci nous paraissait le plus simple, s'attaquer d'abord au plus simple avant d'attaquer le gros morceau ! Grâce au site du zéro et quelques connaissances en XHTML et CSS, ce fut assez rapide de faire un site propre. Il propose plusieurs sections qui permettent de :

- Tenir les visiteurs au courant du développement du projet grâce aux news.
- Des renseignements et images /photos sur le groupe et le projet.
- Une section pour les téléchargements des fichiers de chaque soutenance.
- Bonus pour les plus grands fans.


MENU
Accueil
Télécharger
Le Groupe
Le Projet
Ressources
Contact

Chapitre 5

Ce qui est à faire

5.1 Moteur Physique

C'est vraiment ici que le plus dur reste à faire, et le plus long vu que pour l'instant celui-ci n'est pas vraiment performant. Donc pour la prochaine soutenance nous souhaiterions avoir une gestion correcte des collisions, avec le système des bounding box/bounding sphere délivré par XNA. Nous pensons donc dans un premier temps, mettre notre joueur dans une bounding sphere ou bounding box, et chaque autre élément dans une bounding box. Ainsi, lors de la rencontre de bounding boxes on pourra voir de plus près ce qui se passe.

5.2 Moteur Graphique

Au niveau du moteur graphique, nous allons réaliser un meilleur HUB pour le joueur, avec une arme qui se déplace en même temps que la caméra. La gestion de ce moteur sera sûrement simplifiée avec la création d'une nouvelle classe pour celui-ci, au lieu de tout mettre directement dans l'appel à la fonction *Draw()*. Ensuite, nous allons faire un nouveau système pour les maps qui ne seront plus un gros modèle seulement, mais différents modèles rassemblés selon un fichier texte.

5.3 Modélisation 3D

Comme nous l'avons dit, la modélisation des objets sera faite différemment à présent, nous allons essayer de simplifier la réalisation des maps en créant plusieurs modèles que nous mettrons un par un dans le monde.

Conclusion

On ne peut que dire de l'enthousiasme généré par un tel projet ! Etant tous grands joueurs de FPS, voir une camera se déplacer, modéliser des armes en trois dimensions c'est vraiment quelque chose ! Peu importe le temps passé pendant les quelques jours de repos d'hiver, ca en vaut la peine. Nous allons par la suite essayer de redoubler d'efforts pour lutter et résoudre les nombreux problèmes qui sont apparus lors du développement, affiner les modèles 3D comme les armes ou les différents personnages à venir, diffuser l'avancée du jeu avec des moyens publicitaires...Enfin, nous tenons à souligner la bonne humeur du groupe qui tiendra, nous espérons, dans toutes les conditions !