Sommaire

Intro

Déroulement du projet (pourquoi ce découpage, déroulement des tâches)

Ma partie (Picking (pas encore en place mais je sais comment faire☺), modélisation pioche, insertion pioche dans l’environnement, suivi de la caméra par la pioche).

Ta partie (Collision, gestion des blocs …)

Conclusion

Le tout en 10 pages mini.

# Intro :

Le projet BlockBuster s’inscrit dans le cadre des projets d’option de cinquième année, ici l’option Réalité virtuelle, il nous permet de mettre en place les notions vu en cours et de développer une branche de notre métier peu ou pas aborder en cours : les jeux vidéos.

Ce rapport va développer en première partie le sujet ainsi que les choix effectués pour le mener à bien (découpage des tâches, choix des outils…). Ensuite dans les parties suivantes nous parlerons des différentes fonctions implémentées dans notre projet : leur mise en place et leur fonctionnement. Enfin, en conclusion, une partie sur les difficultés rencontrées et les possibles améliorations à apporter.

# Déroulement du projet :

## Présentation du projet :

BlockBuster est un projet de Mr Aupetit, l’objectif principal est de créer un jeu type Minecraft. Il s’agit de créer un monde de blocs ou ceux-ci sont générer aléatoirement par le moteur du jeu à la demande du joueur. C'est-à-dire que le jeu choisi la taille, la fragilité, poids… du bloc demandé par le joueur. Celui-ci peut, ensuite, s’amuser à sculpter ce bloc afin de lui donner la forme qu’il souhaite. Ces blocs peuvent être bougés et placés où le joueur le veut. Par contre chaque bloc subit la gravité et peut supporter un poids maximum ; si celui-ci est dépassé le bloc disparaît.

Ce jeu doit être développé en Java donc nous avons décidé d’utiliser Jmonkey et son SDK pour travailler. De plus le découpage en tâches nous permet d’utiliser un système de versionnage : Git. Nous avons donc créé un dépôt sur Github qui contient notre projet.

## Découpage :

Suite à la présentation du projet par Mr Aupetit nous avons décidé de découper le projet en 2 parties :

* Collision et Gestion des blocs
* Sculpture

Ce découpage est surtout dû à l’avancement de chacun d’entre nous dans ces autres projets (surtout le PFE). En effet, Thibault ayant une bonne avance dans le sien, il a été décidé qu’il s’occuperait de la gestion des blocs et avec de la gestion des collisions. La sculpture des bocs étant un peu plus simple Pierre a décidé de s’en occuper.

## Déroulement des tâches :

Comme dit dans le paragraphe précédent le développement s’est retrouvé en concurrence avec l’avancement de nos PFE respectifs. Il a donc fallu négocier cela, nous avons donc décidé de « négliger » le déplacement des blocs pour nous concentrer sur les collisions, la génération aléatoire de bloc et la sculpture.

Les collisions ont eu un peu de retard du à un manque d’information disponible sur Internet et donc Thibault a du tâtonner pur trouver une solution viable. Nous verrons plus loin sa solution. La sculpture et la gestion aléatoire des blocs ce sont mieux passé.

# Partie Pierre

## Modélisation d’une pioche et intégration

Le jeu se rapprochant de Minecraft nous avons voulu modéliser une pioche afin de permettre au joueur de sculpter ses blocs. Pour cela nous avons utilisé un logiciel de modélisation : Blender.

Blender est un logiciel de modélisation, d’animation et de rendu 3D libre. Il permet de dessiner assez vite des formes simples et, ensuite de les animés simplement. Sa communauté est très importante il est donc facile de trouver des tutoriels pour apprendre à l’utiliser ou plus généralement des modèles d’objet déjà fait.

Pour la pioche, suite à notre expérience de projet collectif, nous avons décidé de trouver un modèle tout fait et de l’animé par la suite.

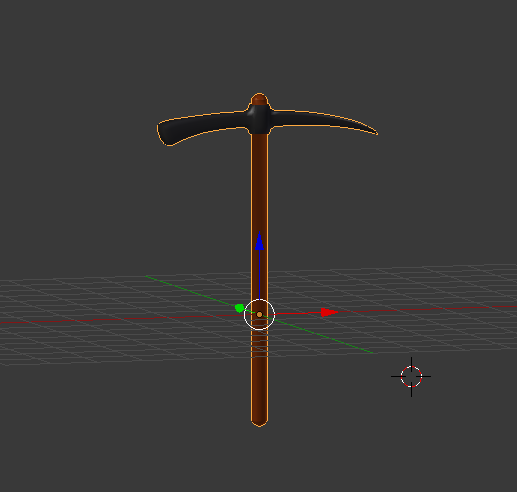


Figure : Pioche

Une fois le modèle trouvé et un peu transformé (ajout des deux matériaux), il a fallu mettre en place l’animation d’un coup de pioche, c'est-à-dire une rotation de la pioche selon l’axe y de 90°. Pour cela nous avons ajouté une armature à notre pioche composé d’un seul Bone (car une pioche ne se déforme pas il n’y a donc pas besoin de plus de Bone).

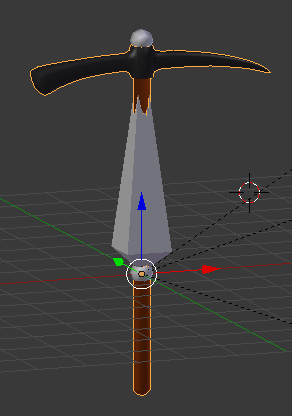


Figure : Pioche avec Bone

Une fois le bone ajouter et lié à la pioche il suffit d’ajouter les différentes phases de l’animation pour que Blender calcule les phases manquantes.

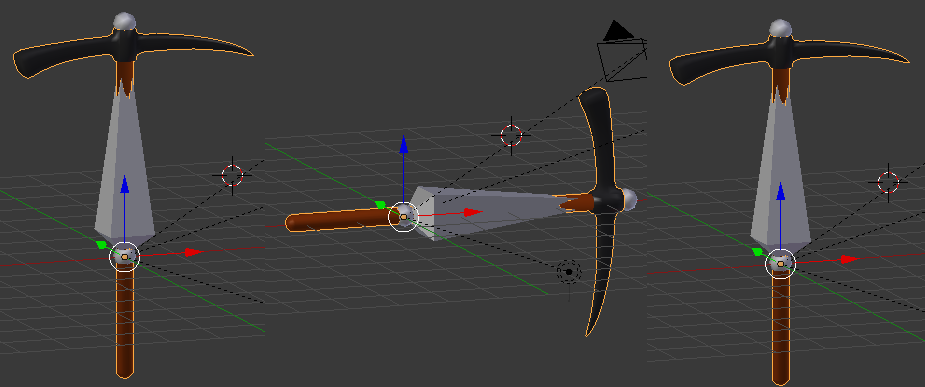
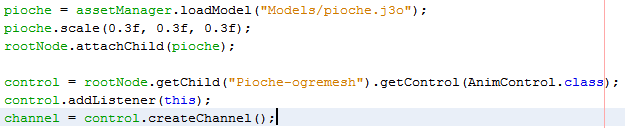


Figure : Les 3 phases de l'animation

Une fois cela terminé il faut exporter le modèle au format ogre pour permettre à Jmonkey de le convertir dans un format qu’il connait (j3o).

L’intégration dans notre jeu et l’animation se fait grâce à ses lignes de code :



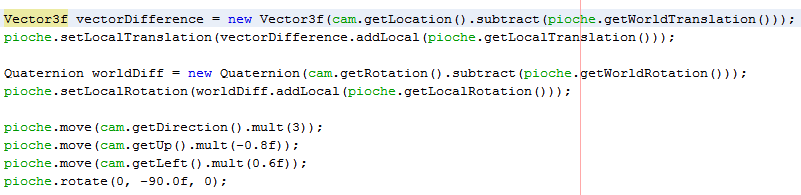
Le scale de la pioche permet de la mettre à l’échelle du jeu et ensuite on l’attache au nœud racine afin de la faire apparaître dans la scène.

Le control et le channel sont les deux variables utilisées pour lancer l’animation quand le joueur clique.

Ensuite il a fallu connecter la pioche à notre joueur, c'est-à-dire que celle-ci devait le suivre tout le temps. Cela n’a pas été facile car notre joueur n’est pas représenté par un nœud simple c’est un PhysicsCharacter. Il était donc impossible de lui attaché un objet. Pour palier à ce problème nous avons lié la position physique du joueur et la position de la pioche grâce à ce code :

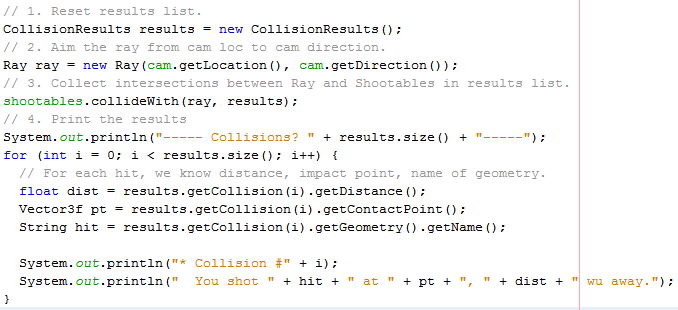


Mais cela a entrainé un autre problème : la pioche suivait bien le joueur mais pas les mouvements de rotation de la caméra. Il a donc fallu calculer les mouvements de rotation de la caméra et les retranscrire à la pioche :



## Picking (Sculpture)

Le picking avec Jmonkey est simple il suffit de ces lignes de codes :



Ceci n’est qu’un exemple car pour l’instant elle n’est pas implémentée dans notre code car c’est plus compliqué avec la gestion des collisions.

Le principe est d’envoyer un rayon et de supprimer le premier bloc rencontré par ce rayon.