Premier Rapport de Soutenance

Projet Stick 'n' Rage

Groupe Slash

5 janvier 2009

Table des matières

Introduction

Dans notre cahier des charges, nous avons élaboré un planning en prévision de cette première Soutenance concernant le moteur Physique, le moteur Graphique ainsi que les Mesh, nous allons donc vous exposer à l'oral l'avancé de ces trois parties ainsi que nos previsions pour la prochaine soutenance.

Chapitre 1

Le Projet

1.1 Précisions du Sujet

Ce projet sera donc, comme nous l'avons déjà dit dans notre cahier des charges, un jeu de combat 2D orienté vers des combats de type Tekken Like. Nous utiliserons pour ceci le personnage du Stickman qui nous évitera d'avoir un rendu moche et hazardeux.

1.2 Rappel des tâches :

Taches	Rippalka	Pull	Msb	Turgal
Moteur Graphique	+	+		
Moteur Physique			+	+
Son			+	+
Site Web	+		+	
Menu		+		+
IA				+
Mesh	+	+		

Chapitre 2

Moteur Physique

2.1 Avancées

Pour cette première soutenance, nous avions décidé de gérer les collisions avec un environnement quelconque avec de simples formes comme un carré. Nous pouvons donc dire "Mission accomplie" car nous avons réussi à animer un carré pouvant bouger et sauter (bien que l'interaction avec l'obstacle soit encore à améliorer). De plus, nous sommes en mesure d'afficher les coordonnées d'un objet soumis à une vitesse quelconque et un angle précis (bien que l'interaction avec l'obstacle soit encore à améliorer).

2.1.1 Théorie de Newton

Grâce à la théorie de Newton, que nous avons appris en cours, et plus précisement grâce à la seconde loi, nous avons réussi à modéliser des trajectoires d'objets ainsi que des collisions de ce même objet avec les éléments du décor.

La trajectoire a été réalisée en utilisant des vecteurs vitesses, des angles ainsi que des coordonnées x/y dans un repère à partir des formules suivantes :

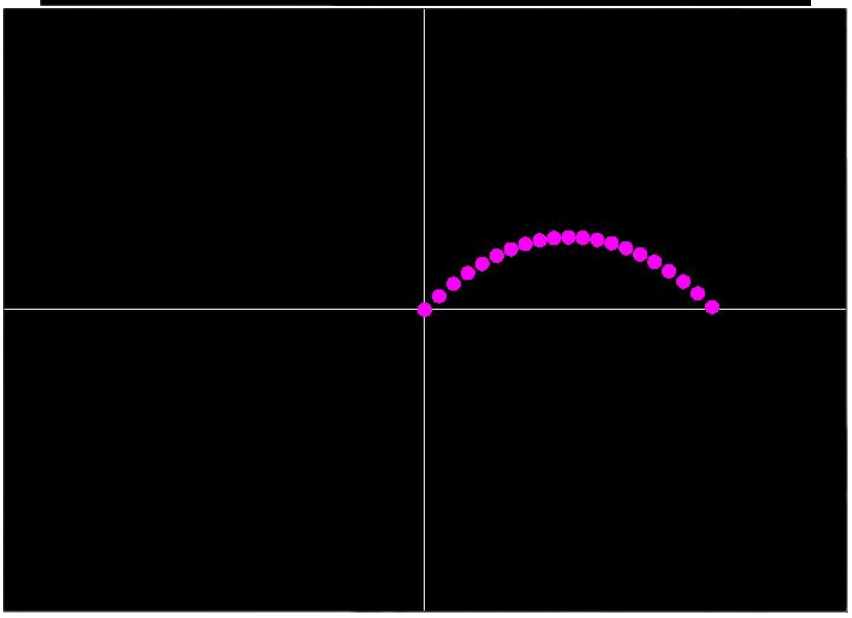
Soit V le vecteur vitesse et V.X, V.Y respectivement les composantes X et Y du vecteur et g la constante gravitationnelle.

$$\overrightarrow{V} = \begin{cases} X = v0 * cos(\alpha) * t \\ Y = v0 * sin(\alpha) * t + g * \frac{t^2}{2} \end{cases}$$

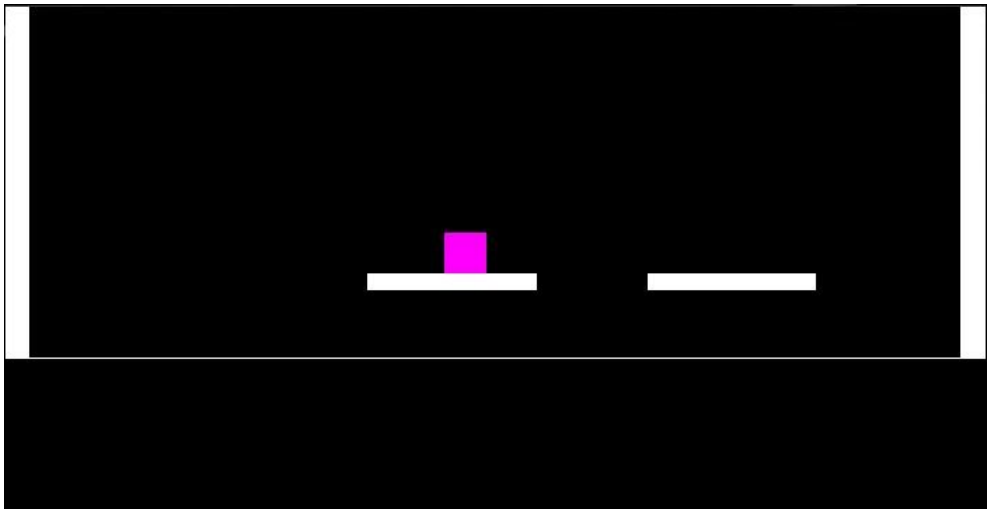
Voici une partie de notre implantation des vecteurs vitesses nous permettant de placer chaque coordonnée du vecteur dans un tableau :

```
while P.Y >= sol do
  begin
    P.X := VX(coeff, v0)*t;
    P.Y := VY(coeff, v0)*t+g.Y*t*t/2;
   t:=t+dt;
    tab[li, col] := P.X;
    tab[li, col+1] := P.Y;
    li := li+1;
  end;
```

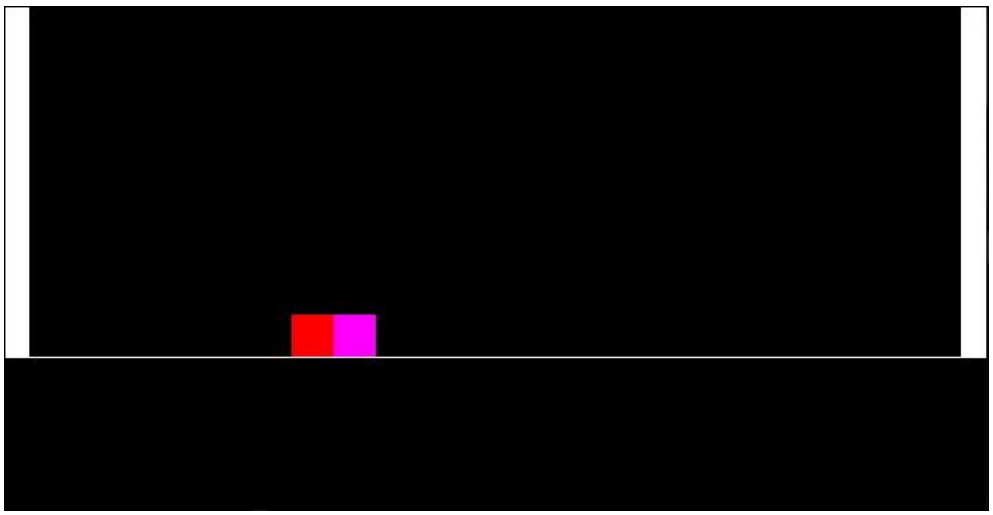
```
P.X :  0.00000000000000E+0000    P.Y :  0.00000000000000E+0000
P.X :  6.92964645562817E-0001    P.Y :  6.68954645562816E-0001
P.X :  1.38592929112563E+0000    P.Y :  1.28988929112563E+0000
P.X :  2.07889393668845E+0000    P.Y :  1.86280393668845E+0000
P.X :  2.77185858225127E+0000    P.Y :  2.38769858225127E+0000
P.X :  3.46482322781408E+0000    P.Y :  2.86457322781408E+0000
P.X :  4.15778787337690E+0000    P.Y :  3.29342787337690E+0000
P.X :  4.85075251893972E+0000    P.Y :  3.67426251893972E+0000
P.X :  5.54371716450253E+0000    P.Y :  4.00707716450253E+0000
P.X :  6.23668181006535E+0000    P.Y :  4.29187181006535E+0000
P.X :  6.92964645562817E+0000    P.Y :  4.52864645562816E+0000
P.X :  7.62261110119098E+0000    P.Y :  4.71740110119098E+0000
P.X :  8.31557574675380E+0000    P.Y :  4.85813574675380E+0000
P.X :  9.00854039231662E+0000    P.Y :  4.95085039231661E+0000
P.X :  9.70150503787944E+0000    P.Y :  4.99554503787943E+0000
P.X :  1.03944696834422E+0001    P.Y :  4.99221968344225E+0000
P.X :  1.10874343290050E+0001    P.Y :  4.94087432900506E+0000
P.X :  1.17803989745678E+0001    P.Y :  4.84150897456788E+0000
P.X :  1.24733636201307E+0001    P.Y :  4.69412362013069E+0000
P.X :  1.31663282656935E+0001    P.Y :  4.49871826569351E+0000
P.X :  1.38592929112563E+0001    P.Y :  4.25529291125633E+0000
P.X :  1.45522575568191E+0001    P.Y :  3.96384755681914E+0000
P.X :  1.52452222023819E+0001    P.Y :  3.62438220238196E+0000
P.X :  1.59381868479447E+0001    P.Y :  3.23689684794477E+0000
P.X :  1.66311514935076E+0001    P.Y :  2.80139149350759E+0000
P.X :  1.73241161390704E+0001    P.Y :  2.31786613907040E+0000
P.X :  1.80170807846332E+0001    P.Y :  1.78632078463322E+0000
P.X :  1.87100454301961E+0001    P.Y :  1.20675543019603E+0000
P.X :  1.94030100757589E+0001    P.Y :  5.79170075758847E-0001
P.X :  0.00000000000000E+0000    P.Y :  0.00000000000000E+0000
```



Nous avons ensuite réalisé grâce à la trajectoire, des collisions avec des éléments du décor comme des obstacles où l'environnement dans lequel l'objet est enfermé. Il nous a suffit de récupérer dans un tableau les coordonnées successives de l'objet dans un intervalle de temps prédéfini et d'utiliser nos fonctions génériques pour pouvoir faire n'importe quelle application.



Grâce à ces mêmes fonctions génériques, gérant les calculs de coordonnées, la gravité et la réaction avec d'éventuels obstacles, nous pouvons faire de multiples applications assez rapidement. Ainsi, nous avons pu faire cette application qui consiste tout simplement à pousser une "caisse".



2.2 Prévisions en vue de la Seconde Soutenance

Voici ce que nous prévoyons lors de notre seconde soutenance :

- amélioration de la réaction des obstacles.
- application du moteur physique au moteur graphique, cest-à-dire que nous serons capable de faire réagir les différents stickman lorsqu'ils se prendront des coups ou simplement de ne pas pouvoir traverser l'autre joueur et les bords du "ring".

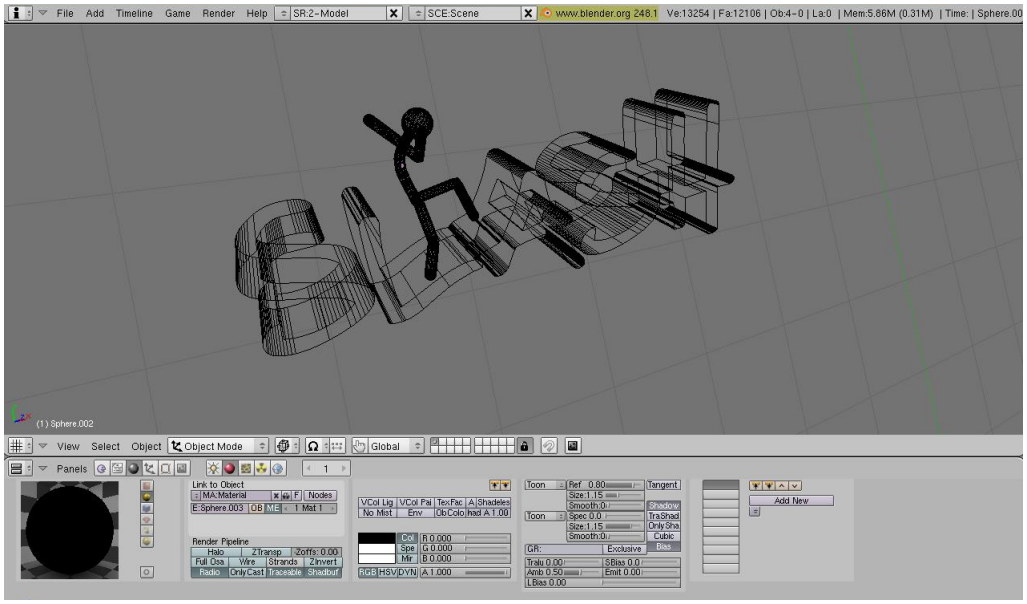
Chapitre 3

Moteur Graphique

3.1 Modélisation

3.1.1 Réalisation du Logo

Nous nous somme tout d'abord servi de Blender pour réaliser un Logo 3D comprenant notre Stickman et le nom 'SLASH'.



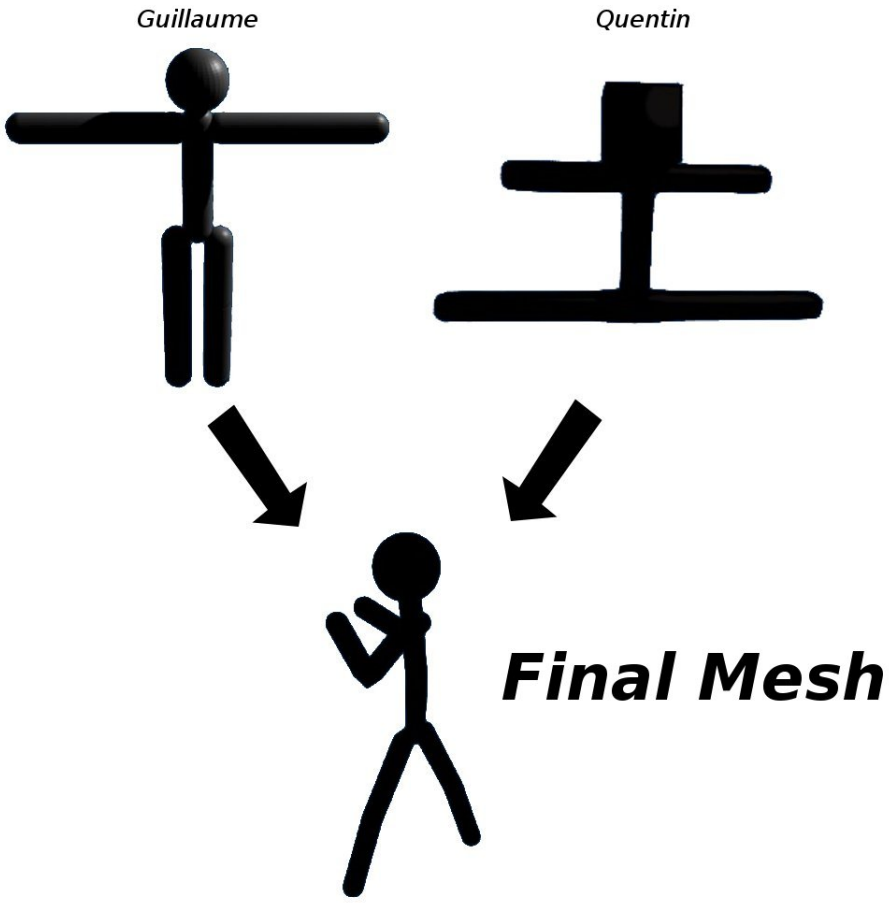
3.1.2 Outils Graphiques

Après de nombreuses heures passées à comprendre les utilisations d'OpenGL et de Blender, nous avons pris conscience de la puissance de ces deux outils, et par conséquent nous allons poursuivre notre projet avec ceux-ci. Malgré cela, ces deux outils ont vite laissé paraitre leurs défauts, notamment au niveau des animations. En effet Blender ne gère l'export d'animation qu'avec quelques formats comme le MD2

(Format de Fichier de Quake II), et ceux-ci s'importent très mal avec OpenGL. Or, notre jeu reposera entièrement sur des animations, c'est pourquoi nous sommes toujours à la recherche d'une solution.

3.1.3 Mesh

Dans un premier temps, nous nous sommes concentrés sur la réalisation de notre personnage principal, à savoir un Stickman. Nous nous sommes documentés séparement, puis avons synthètisé nos recherches pour obtenir le FINAL MESH, ci-dessous :



Comme on peut le voir, Quentin était parti sur un mesh à base de cubes, Guillaume à base de cyclindres et de sphères. Les deux étant composés d'énormément de points (celui de Quentin dépassait les 215 000 Vertex ! ! !) et n'ayant pas un rendu acceptable, nous avons dû uniformiser et optimiser tout cela. Le rendu final contient près de

6000 Vertex, ce qui est toujours un nombre élevés mais le rendu est plus vrai que nature.

3.1.4 Animations

Grâce à ce Mesh, nous avons réalisé plusieurs animations basiques sous Blender, à savoir : coup de poing, coup de pied, animation au repos et marche. Nous les avons réalisées en vue de les importer dans OpenGL par la suite. Le format d'export est toujours à déterminer car malgré plusieurs tentatives avec Blender aux formats : .3ds .md2 .md3 .md5 .obj, les animations ne sont toujours pas exportées. Nous avons néanmoins fait une tentative en rechargeant le mesh pour chaque Frame (le mesh ayant précedement été exporté en plusieurs fichiers .3ds pour chaque Frame), mais ceci ne nous parait pas être une solution optimale car cela équivaut à charger 25 fois le mesh en une seconde (25FPS). Par exemple, en appliquant des mouvements aux deux personnages sur la scène, nous constatons un ralentissement.

```
if Coupdepied then // coup de pied variable 'boolean' lors de la pression de la touche G
   begin
       if Actual2 <= 20 then
         begin
             GLFreeForm1.LoadFromFile(appdir+'Coup de pied\'+IntToStr(Actual2)+'.3ds');
             Actual2 := Actual2 + 1;
         end;
       if Actual2 = 20 then
         begin
             Actual2 := 1;
             Coupdepied := false;
             Pause := true; // permet l'animation du mesh
         end;
      end;
```

Puis nous avons pour l'instant provisoirement lié les touches du clavier pour déplacer simultanément les deux personnages selon les axes X/Y.

```
case key of
    '-' : GLCamera1.AdjustDistanceToTarget(1.1);
    '+' : GLCamera1.AdjustDistanceToTarget(0.9);
    'd' : GLFreeForm1.Position.X := GLFreeForm1.Position.X + 0.1;
    'q' : GLFreeForm1.Position.X := GLFreeForm1.Position.X - 0.1;
    '6' : GLFreeForm2.Position.X := GLFreeForm2.Position.X + 0.1;
    '4' : GLFreeForm2.Position.X := GLFreeForm2.Position.X - 0.1;
    'z' : GLFreeForm1.Position.Y := GLFreeForm1.Position.Y + 0.1;
    's' : GLFreeForm1.Position.Y := GLFreeForm1.Position.Y - 0.1;
    '8' : GLFreeForm2.Position.Y := GLFreeForm2.Position.Y + 0.1;
    '5' : GLFreeForm2.Position.Y := GLFreeForm2.Position.Y - 0.1;
end;

Form1.Paint;
   key := #0;
```

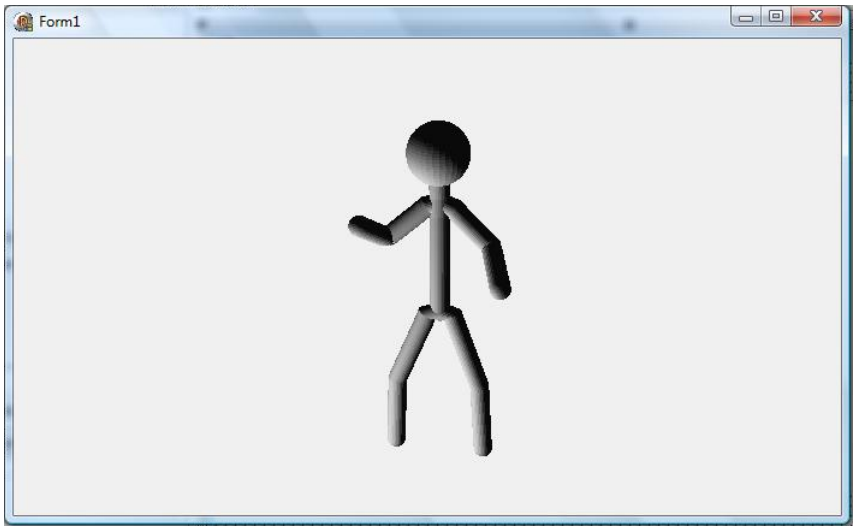
3.2 OpenGL / Delphi

3.2.1 Introduction

Après avoir difficilement affiché nos premières primitives sous OpenGL, nous nous sommes rendu compte du niveau de difficulté pour la réalisation d'un rendu graphique de jeu vidéo. C'est pourquoi nous avons cherché des librairies pour nous aider et c'est ainsi que nous sommes tombés sur GLScene et Glut.

3.2.2 GLScene

Nous avons choisi d'utiliser principalement la librairie 3D GLScene en dépis de Glut sur laquelle nous ne nous sommes pas encore vraiment interessés. La documentation pour cette librairie est pratiquement inexistante, mais néanmoins GLScene nous a permis de réaliser rapidement un rendu satisfaisant grâce à son fonctionnement intuitif. En effet, GLScene permet la création de scènes complètes : Camera, Source de lumière, Objets/Acteurs. Nous l'avons donc pour l'instant utilisé dans la majorité des tâches effectuées.



3.2.3 Level Design

Pour cette première soutenance, nous nous sommme principalement interessés au Level Design, c'est-à-dire à l'élaboration du terrain de jeu. Nous sommes tout d'abord partis sur l'idée d'un décor basic (montagne, hangar) mais nous avons pris conscience de la monotonie de la scène.

C'est pourquoi nous avons créé un plan de scène un peu plus complexe, que nous avons réalisé en trois parties :

- GLSkyBox
La SkyBox est en faite un cube englobant la caméra de la scène de telle manière qu'elle soit en permanence fixée au centre de ce cube. En chargeant une texture sur chaque face de cette SkyBox, elle donne un effet d'horizon à la scène.

- Environnement
Pour éviter d'obtenir une scène isolée, nous avons immergé notre arène de jeu au milieu d'une ville qui est en harmonie avec la SkyBox. En effet, le mélange de la Skybox et de la ville 3D renforce l'effet d'immertion. Cette ville est en réalité un mesh trouvé sur internet que nous avons redimentionné puis chargé dans OpenGL.

- Arène
Toujours dans l'optique d'immertion du joueur dans le jeu, l'arène est située au sommet d'un immmeuble, lui-même situé au milieu de la ville. Pour cela, nous avons simplement réalisé l'integralité de l'immeuble constituant l'arène à l'aide de cubes. De plus, l'arène comprend un DummyCube (Cube invisible sur la scène) à son centre sur lequel est centré la camera principale. Ceci permet une vue en permanence de l'arène.

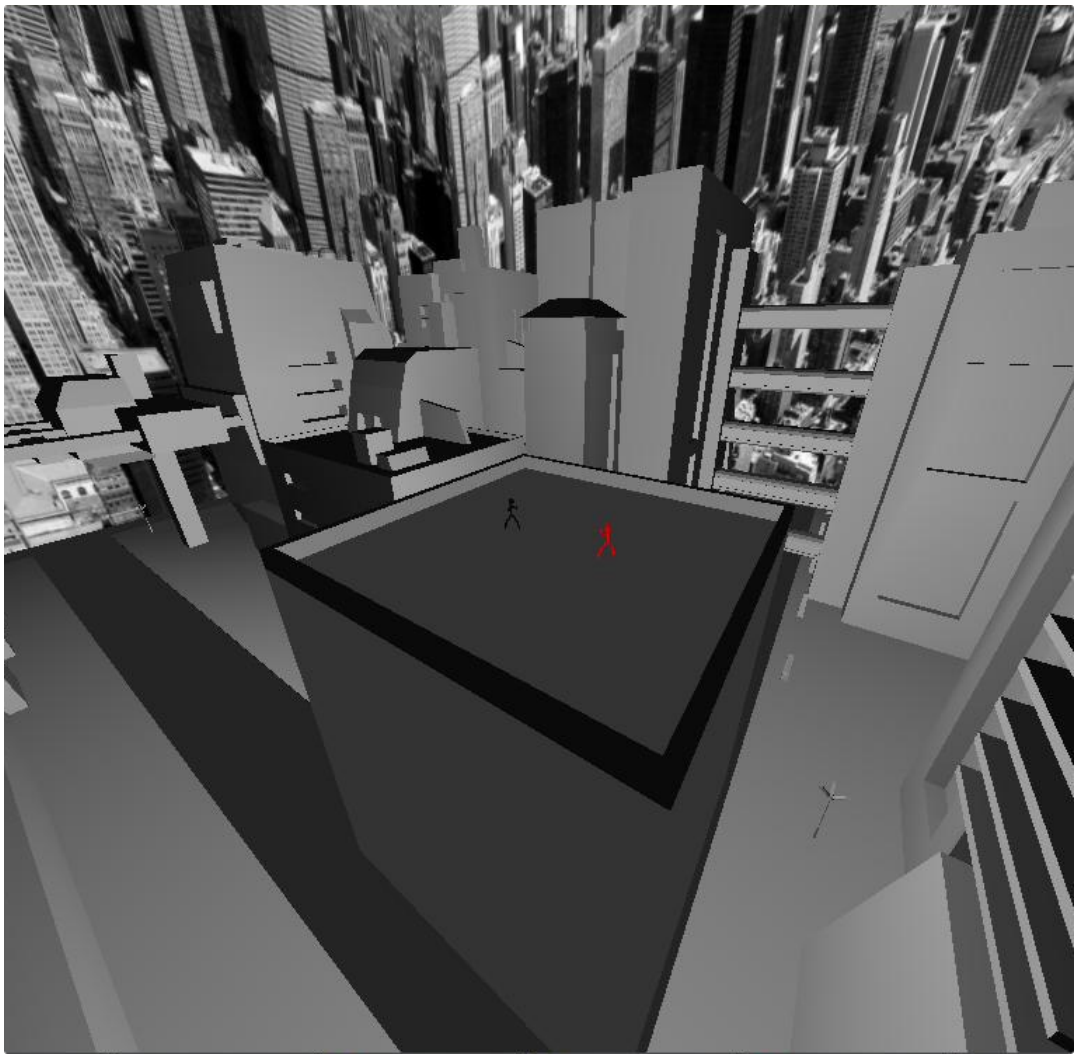
Par soucis de notre univers de Stickman, nous avons opté pour un environnement en noir et blanc. En effet, cela permet de mettre en avant les personnages qui sont eux en couleur.

3.2.4 Personnages

Nous avons, grâce à GLScene, importé nos Mesh que nous avons placé au milieu de l'arène. Nous avons ensuite ré-appliqué les textures sur nos personnages, car celles-ci ont disparu lors de l'export au format .3ds à partir de Blender.

3.2.5 Fonctions de prévisualisation

Pour l'instant nous avons créé quelques fonctions pour avoir une meilleure perception de nos rendus. Nous avons par exemple créé une fonction permettant grâce à la souris de se déplacer autour de la scène, ou encore une fonction ZOOM -/+.



3.3 Prévisions pour la seconde soutenance

Pour la deuxième soutenance, nous avons prévu d'avancer le système d'animation afin d'obtenir des animations fluides, puis par conséquent de créer une fonction permettant d'adapter la distance de la camera par rapport à la scène en fonction de la position des personnages. Ensuite nous allons commencer l'interface de jeu comprenant timer, barres de vie, etc...

Chapitre 4

Retour sur le planning

Nous allons développé cette partie en plusieurs points comprenant le site web ainsi que le menu.

Le site internet, créé par Guillaume et Thomas, sera terminé en vu de la seconde soutenance, il comportera, comme cela a été dit précedement dans le cahier des charges de début de projet : une présentation du groupe et du projet, une page de téléchargement ainsi qu'un module de news.

Le menu, traité par Quentin et Sébastien, sera déjà bien entamé lors de la seconde soutenance où nous pourrons vous montrer l'avancée du travail.

Après mûre reflexion sur la conception de notre jeu, nous avons conclu qu'un éditeur de map était inutile puisque les mondes seront seulement un décor et non un parcours d'obstacle dans lequel le personnage auraît pu évoluer.

Conclusion

Pour conclure, nous avons gérer la gravité et la gestion des collisions sur des formes simples pour le côté physique. Au niveau graphique, nous avons un monde quasi-fini avec quelques animations non-optimisées. Il nous suffira donc d'appliquer la physique au graphique, de commencer les interfaces du jeu et les différentes attaques disponibles. Ainsi, nous dirons que nous ne sommes pas en retard sur le planning mais même plus avancés que nos espérances par rapport au cahier des charges. Passé le stade de la découverte des différents outils, notre groupe est prêt à partir sur de nos nouvelles bases pour mener à bien ce projet : Stick'n'Rage !

Chapitre 5

Bibliographie

5.1 Partie Graphique

5.1.1 Blender

- www.SiteDuZero.com
- www.tuto-blender.com
- www.daz3d.com (Ville 3D)
- www.dafont.com

5.1.2 OpenGL

- www.Developpez.com
- www.codes-sources.com
- www.delphipage.free.fr/opengl.html
- www.glscene.org
- www.eraquila.iquebec.com/site/delphi/opengl/gl1.html
- Google est notre ami

5.2 Partie Physique

5.2.1 Théorie

- www.Developpez.com
- www.codes-sources.com
- www.programmersheaven.com
- Google est notre amis nous aussi.