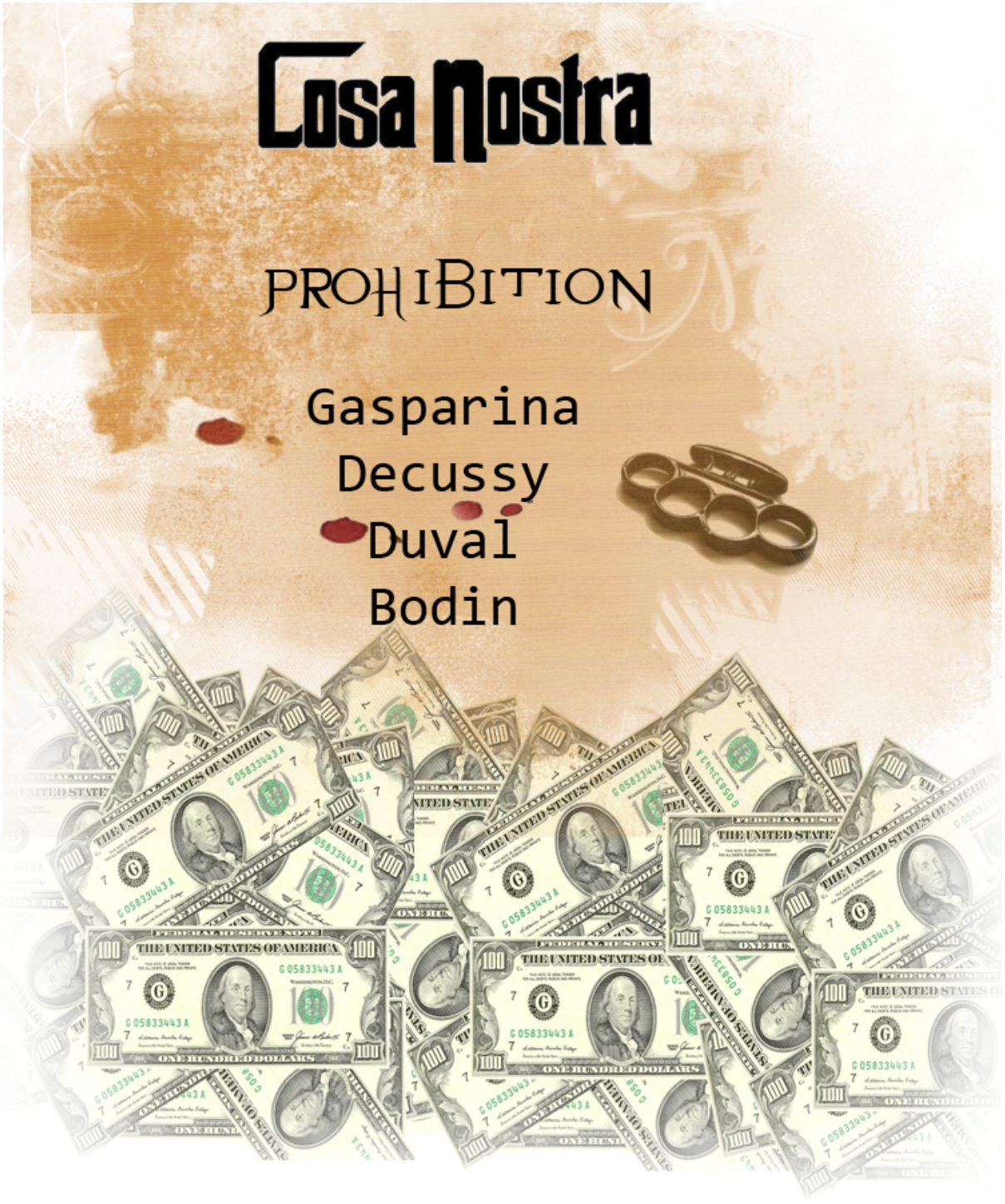


Rapport de la première soutenance

Table des matières

1 Groupe **4**

2 Impressions personnelles **5**
2.1 Damien 5
2.2 Fabien 5
2.3 Alexandre 5
2.4 Raphaële 5

3 Avancement par rapport au cahier des charges **6**
3.1 Modification 6

4 Travail effectué **7**
4.1 Intellligence Artificielle 7
4.2 Moteur Graphique 7
4.2.1 Les Mesh 7
4.3 Affichage de la carte 8
4.3.1 La Caméra 9
4.4 Gestion des collisions 9
4.5 Le Clavier et la Souris 10
4.6 La lumière 10

5 Structure **11**

6 3D **12**
6.1 Le son 15
6.2 site web 16

Introduction

Voilà maintenant deux mois que l'aventure a commencé et quelle aventure ! Un projet d'envergure dans lequel nous nous sommes lancés corps et âmes... Mais nous fûmes vite stoppés par les embûches dispersées sur notre chemin, comme si un démon invisible avait parcouru internet et lui avait arraché toutes formes d'informations. Devant l'adversité, nous avons persistés, nous avons dû alors traduire des dialectes jusqu'alors inconnus dont le C ! Fouiller dans un nombre de documents qui nous submergaient et qui pour la plupart ne nous convenaient pas. Pourtant, nous voici en ce jour à vous présenter fièrement notre début, à vous conter nos joies et nos peines, la grande epopée d'une petite équipe face à son destin : La Prohibition's Team.

1 Groupe

Le travail ne s'est pas vraiment passé en groupe, vacances de Noël obligent. Chacun est retourné chez lui, surtout Alexandre et Fabien, les deux bretons qui sont retournés chez eux assez longtemps pour profiter de leur famille. Damien et Raphaële se sont occupés de la caméra pendant une journée avant leur retour. Pourtant la distance n'a pas été un obstacle chacun à travaillé de son côté malgré la difficulté des nouvelles connaissances à apprendre, mais msn et svn nous ont bien aidés à rester en contact et à progresser. Les deux dernières journées ont été occupées à la mise en commun, à régler quelques problèmes et à répéter l'oral avant la soutenance de lundi.

Une petite pensée émue aux paquets de cookies et de chocolat Milka qui sont tombés ces journées là...

.

2 Impressions personnelles

2.1 Damien

J'ai réalisé toute la partie graphique du jeu, une grosse partie mais néanmoins, comme je code depuis longtemps il était normal que je prenne en charge cette partie lourde et qui demande beaucoup d'organisation.
J'ai eu beaucoup de mal à m'habituer à SlimDX, peu de tutoriels et trop récents pour obtenir de l'aide sur internet.
La documentation de SlimDX me fut très utile, j'ai réussi à comprendre la structure du code et les dépendances des fonctions, ainsi j'ai pu aider tout mon groupe à utiliser SlimDX et à traduire les codes ManagedDirectX.
J'ai obtenu l'aide de Raphy lors de la partie de l'interactivité de la caméra, en effet l'utilisation des matrix est déconcertantes aux débuts.

2.2 Fabien

Le projet a, à priori, l'air intéressant, la documentation nécessaire est parfois un peu difficile à trouver, mais on finit toujours par la trouver après quelques heures de recherches. Me concernant, j'ai implémenté un algorithme de pathfinding, ou du moins j'ai tenté de l'implémenter, j'ai pour cela choisi d'implémenter en Delphi le très connu algorithme de dijkstra. J'ai fait ce choix car il semble que c'est le seul algorithme qui trouve le chemin le plus court. Mais ce que j'ai pour le moment fait ne me semble pas vraiment optimisé, et ne fonctionne pas. Je tenterai donc de l'optimiser une fois que j'aurai eu la preuve que ce que j'ai fait fonctionne vraiment une fois fini et que j'aurai pu le tester. Ma déception est grande, après avoir passé une cinquantaine d'heure sur l'implémentation de cet algorithme.

2.3 Alexandre

Je me suis donc occupé de la partie son. Le début fut... comment dire... laborieux ? J'ai passé énormément de temps sur la recherche. Il y a très peu d'informations concernant la gestion du son avec Delphi. Ainsi, deux principaux choix s'offraient à nous. Utiliser soit FMOD, soit Managed DirectSound. J'ai tout d'abord tenter d'utiliser FMOD. Il paraissait simple dans son utilisation... Je me suis rendu compte par la suite que cet API ne fonctionnait pas avec Delphi .NET. Je me suis donc rabattu vers SlimDX mais j'ai rapidement réalisé que le module DirectSound de cet API était toujours en Construction. J'ai donc abandonné cet API. J'ai donc utilisé Managed DirectSound un peu plus compliqué selon moi. J'ai donc dû traduire du C en Delphi.

2.4 Raphaële

N'ayant aucune connaissance en delphi au départ, lors de la distribution des tâches, je me suis donc attelée à la 3D, pour apprendre à maîtriser Blender. Le départ fut un peu difficile, je n'ai jamais vraiment touché à un logiciel de graphisme et encore moins de 3D. Du moins, j'ai suivi des tutoriels vidéos qui m'ont été d'une grande aide. Après je me suis occupée de la gestion caméra avec Damien, le plus dur fut de trouver comment ça marchait surtout sans tutoriel sous la main.

3 Avancement par rapport au cahier des charges

Ce que nous avions prévu				
Point	**Raphaële**	**Damien**	**Alex**	**Fabien**
Inteligence Artificielle		X		XX
Moteur Graphique		XX		
3D	XX		X	
Audio	X		XX	
Editeur de map				
Site Web	X	XX		

Ce que nous avons fait :				
Point	**Raphaële**	**Damien**	**Alex**	**Fabien**
Inteligence Artificielle		X		XX
Moteur Graphique	X	XX		
3D	XX		X	
Audio			XX	
Editeur de map				
Site Web	X	XX		

3.1 Modification

Nous avons bien entendu essayé de respecter au maximum le cahier des charges. La répartition des tâches n'a pas été totalement respectée. Aucune autre modification n'a été effectuée pour cette première soutenance

4 Travail effectué

4.1 Intellligence Artificielle

J'ai d'abord chercher à comprendre comment fonctionnait une IA avant de finalement comprendre que l'on ne pouvait pas décrire comment celle-ci fonctionne, tout dépend de ce que l'on souhaite que celle-ci fasse. J'ai donc fait des recherches pour trouver un algorithme de pathfinding. Celui de dijkstra semble être le meilleur que je puisse trouver pour notre utilisation, celui-ci étant de complexité n, le nombre de case et trouvant le chemin le plus court contrairement à l'algorithme A star qui trouve un chemin, mais pas forcement le plus court.
Après quelques heures passées à tenter de comprendre dijkstra, je suis finalement tombé sur cette page : http ://www.techno-science.net/ ?onglet=glossaire&definition=6470. L'algorithme étant déjà écrit et n'arrivant pas à comprendre son fonctionnement, je me suis dis que je pourrais finalement l'implémanter sans le comprendre. Mais c'est finalement en commençant à le coder que j'ai compris son fonctionnement.
L'algorithme général est celui-ci :

```
Dijkstra(G ,sdeb)
 1 Initialisation(G,sdeb)
 2 P:= ensemble vide
 3 Q:= ensemble de tous les noeuds
 4 tant que Q n'est pas un ensemble vide
 5       faire s1:= Trouve_min(Q)
 6          P:= P union {s1}
 7          pour chaque noeud  s2 voisin de s1
 8              faire maj_distances(s1,s2)
```

G étant l'ensemble des sommets ainsi que des arrêtes, sdeb le point de départ.

4.2 Moteur Graphique

4.2.1 Les Mesh

Notre première mission, charger un mesh et l'afficher !
Notre périple fut dur, sur notre route nous avons rencontré plein de méchants ! Néanmoins, un mercredi matin, notre mesh s'est enfin affiché !

Charger un Mesh Charger le mesh n'est pas bien difficile, nous utilisons le constructor 'FromFile' de mesh pour le charger correctement

```
my_mesh := Mesh.FromFile(my_device,path,MeshFlags.SystemMemory);
```

Nous récupérons ensuite le 'extended material' du mesh auquel nous récupérons le chemin de la texture et le 'Material' du mesh.

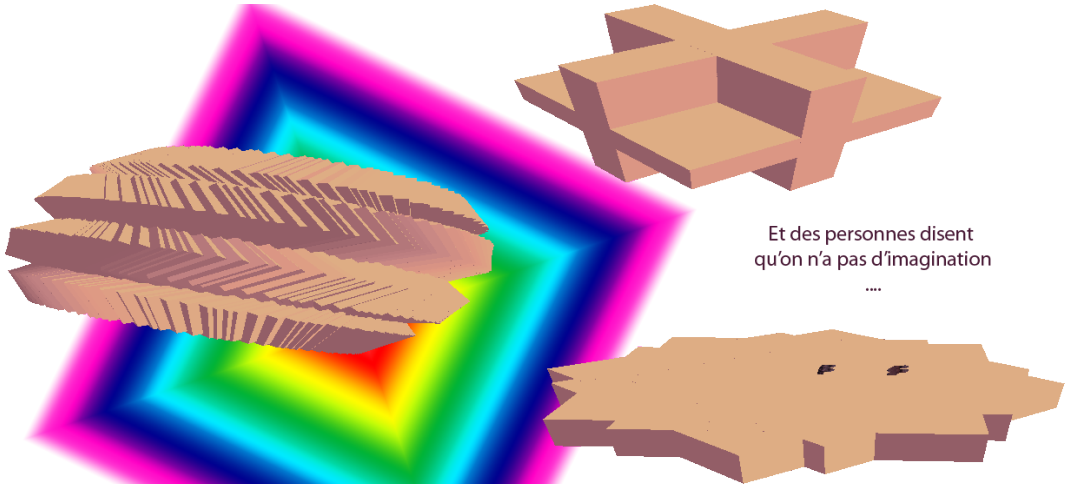
Afficher le Mesh Bien qu'il existe une fonction déjà créée pour afficher un mesh : 'drawsubsect', nous avons eu beaucoup de mal à avoir un rendu, en effet, même si l'utilisation de cette fonction est simpliste, elle a beaucoup de dépendance.
Nous devons avant tout initialiser correctement tout les paramètres graphiques, la caméra, le device, la fenêtre.
Nous avons donc du comprendre tout le fonctionement de l'environement graphique avant de réussir à avoir un rendu.

4.3 Affichage de la carte

Après de durs combats, notre carte s'est enfin affichée, notre carte est un carré composé de 110 carrés de petite taille qui délimitent chaque case du terrain
La plus grande partie du travail est réalisée, il ne reste plus qu'a créer une fonction qui permet de personnaliser la carte automatiquement à partir d'un fichier externe, mais cela ne devrait pas poser de problème.
Chaque case est créé grace à la fonction CreateBox du module Mesh de SlimDX, nous les positionnons en effectuant des translations sur la matrix du monde.

```
position := Matrix.Translation(Vector3.Create(WIDTH_MAP,0,0));
  my_device.MultiplyTransform(TransformState.World,position);
  Boucle D'affichange
  position := Matrix.Translation(Vector3.Create(- WIDTH_MAP * 11 ,0,HEIGHT_MAP))
```

Mais bien sûr, avant de réussir à afficher correctement notre grand carré, nous avons eu beaucoup de surprises...



4.3.1 La Caméra

Notre jolie caméra marche bien, on a eu beaucoup de mal à comprendre comment elle marchait, mais maintenant c'est un vrai bonheur de la coder.
Nous avons essayer, pour notre première soutenance nous avons éssayé de créer un maximum d'intéractivité avec la caméra.

```
procedure Cosa.SetupMatrix;
 begin
   my_mouse_input.getView(xrot,yrot,z);
   world   := Matrix.RotationY(xrot / 15);
   world   := Matrix.Multiply(world,Matrix.RotationX(yrot / 8));
   my_device.SetTransform(TransformState.World, Matrix.Identity);
```

Pour permettre les interactivités avec la caméra, nous utilisons des variables qui dépendent de la souris et du clavier, bien que le nom ne soit pas explicite, world représente l'utilisateur, on peut facilement lire sur le code que l'on peut faire pivoter la caméra, xRot et Yrot dépendant de la position de la souris.
La fonction GetView prend en référence Xrot Yrot et Z et permet de changer ses valeurs en fonction de la position de la souris et des clics.

Fonctionalité Le clique gauche maintenant nous permet de faire pivoter la map et la mesh.
La roulette ainsi que la commande clavier défilement page nous permet de zoomer.
Les flèches permettent de faire avancer le mesh sur la map.
Un clique gauche sur le mesh suivi d'un clique gauche sur une des cases de la map permet au mesh de s'y déplacer.

4.4 Gestion des collisions

Une partie complexe, nous devions être capable de sélectioner un objet et de le déplacer avec la souris. Les Matrix ont un comportement étrange et sont très durs à manipuler, finalement, après une journée entière à réfléchir dessus, nous avons réussis à obtenir ce que l'on voulait. Un objet, une fois selectionné peut être bougé vers une autre case du jeu.

Nous avons, pour cela, projeté les coordonnées de la souris sur le plan des objets et détecté si il y avait des collisions

```
vnear := Vector3.Unproject(vIn, my_device.Viewport, proj, view, world);
  vIn.Z := 1;
  vfar := Vector3.Unproject(vIn, my_device.Viewport, proj, view, world);
  vDir := Vector3.Subtract(vFar, vNear);
  my_ray := Ray.Create(vnear, vDir);
  result := my_mesh.Intersects(my_ray);
```

La procédure 'Unproject' permet de transformer un point 2D, ici notre souris, en un point 3D.
Ensuite, pour obtenir la direction de la souris, indispensable pour calculer les collisions on calcule la différence de deux projections de la souris qui n'ont pas la même profondeur.
Ainsi, en obtenant la direction et la projection, on peut facilement calculer si il y a collision avec un mesh grace à la procédure 'intersect' du type mesh.

4.5 Le Clavier et la Souris

Détecter les mouvements de la souris et les touches du clavier n'était pas bien compliqué.
Le plus dur dans cette partie fut d'initialiser tout les devices nécéssaire, nous avons du utiliser pour la souris 2 méthodes différentes, une pour récupérer les coordonnées en relatif, les déplacements en fonction de sa dernière position, et en absolu, sa position sur l'écran en fonction des pixels
Pour initialiser nos devices, nous avons utilisé une syntaxe plus ou moins étrange.

```
my_ms := device<mouseState>.Create(SystemGuid.Mouse);
my_kb := device<KeyboardState>.Create(SystemGuid.Keyboard);
```

Suite à des problèmes de périphérique, nous avons optimisé le code en créant des objets de type clavier et souris, en effet demander le status du clavier ou de la souris plusieurs fois par frame nous fait perdre des informations, nous avons donc restructuré notre code pour permettre de récupérer les positions de la souris et l'état du clavier n'importe où sans rappeler la fonction pour récupérer l'état des périphériques.

4.6 La lumière

Que la lumière soit !
La lumière est indispensable pour obtenir un bon rendu, nous l'avons appliquée simplement sur le monde, bien que nous ayons activé les ombres, nous n'avons pas encore eu le temps de travailler sur celles-ci. De plus, comme nous avons prévus une vue aérienne pour notre jeu, on ne pense pas que les ombres occupent une place très importante dans le rendu.

```
my_light.Direction := vecDir;
  my_light.Range := 3000;
  my_device.SetLight(0,my_light);
  my_device.EnableLight(0, true);
  my_device.SetRenderState(RenderState.Lighting, true);
  my_device.SetRenderState(RenderState.ShadeMode, true);
  my_device.SetRenderState(RenderState.Ambient,11110111);
```

5 Structure

La structure du code est posée, même si toute la partie graphique et audio ne sont pas encore modulable.
Beaucoup d'objets indispensables pour notre jeu sont créés, nous avons appris la programmation orientée objet en même temps.

```
type personnage = class
    public
      constructor Create(nom_ : string);
      procedure TestVariable;
    private
      pv : integer;
      exp : integer;
      nom : string;
end;
```

Bien sûr, nous avons utilisé des notions plus complexes de la POO pour rendre chaque personnage personnalisable, l'objet 'Personnage' est la base de tous nos personnages, nous utilisons l'héritage et la surcharge des opérateurs pour définir nos autres classes

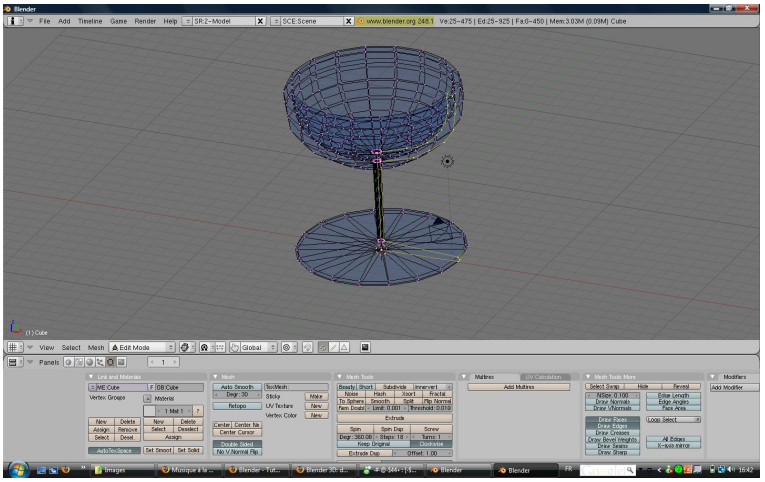
```
type tireur_distance = class (personnage)
  public
    constructor Create(nom_ : string);reintroduce;overload;
```

Bien que le rapport entre le rendu et la structure ne soit pas encore réalisé, nous comptons faire en sorte que les types puissent offrir un maximum d'options, rendus, changement des textures, déplacements, création.
Il nous faudra aussi créer un type spécial 'case' pour utiliser dijkstra et pour avoir une carte plus détaillée.

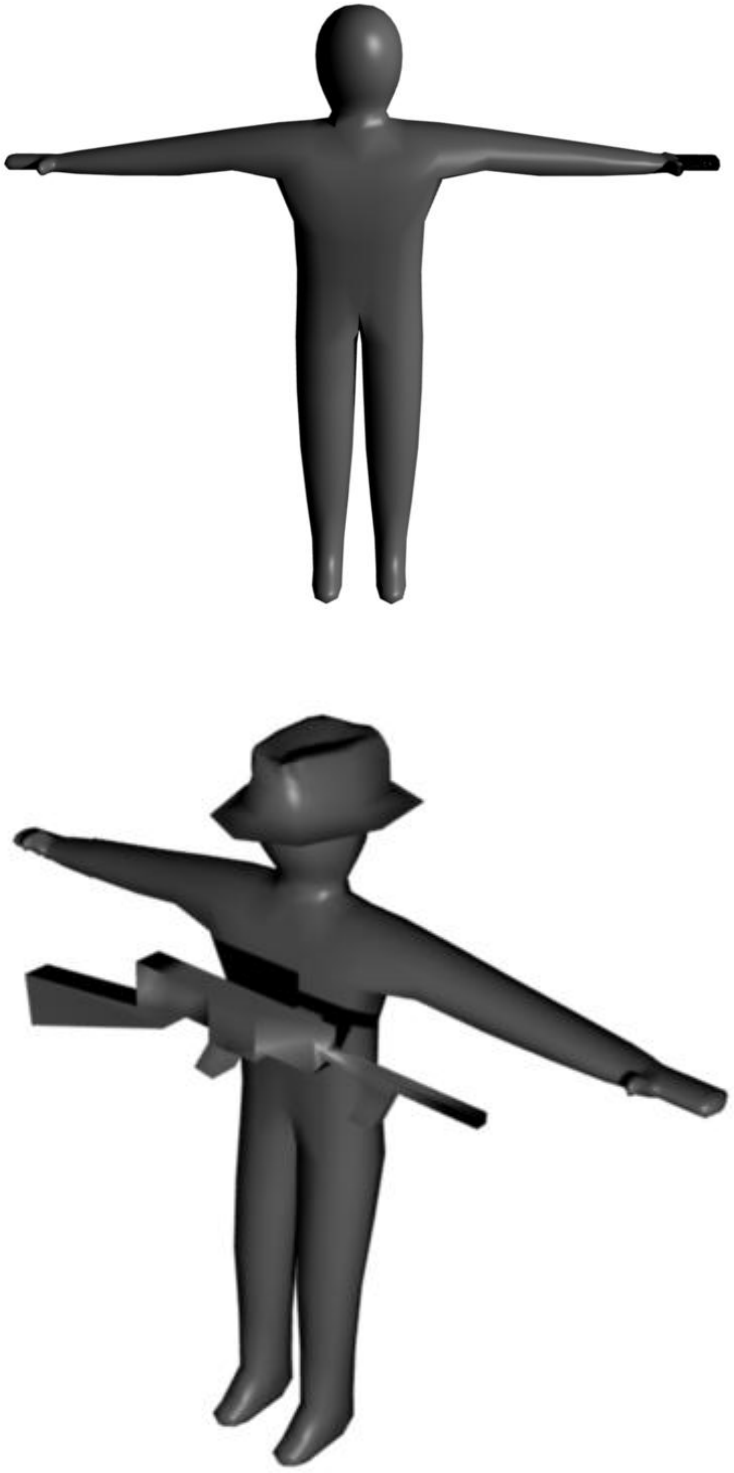
6 3D

Pour la partie 3D, il a d'abord fallu apprendre à maîtriser Blender, ce qui ne fut pas tout de suite évident puisque ce logiciel n'est pas vraiment intuitif au départ. Du moins, on a trouvé de bons tutoriels vidéos pour débuter et apprendre à gérer l'outil principal mais très pratique : l'extrude.

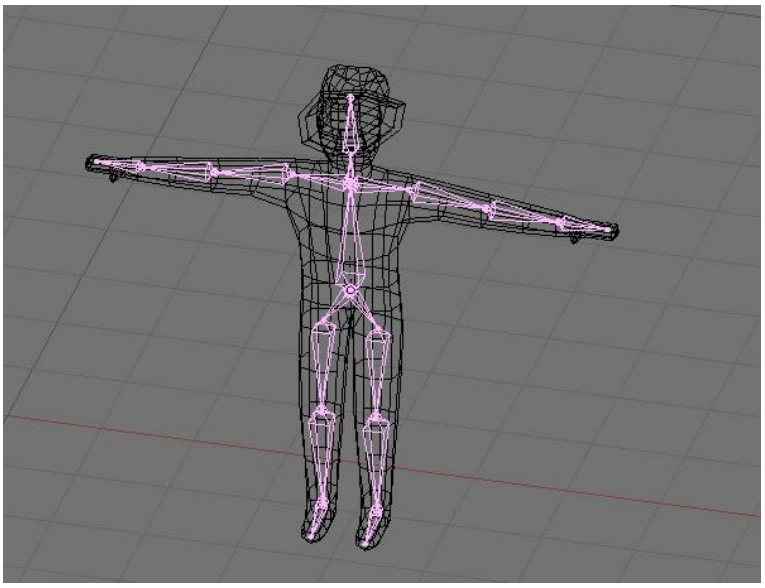
Quelques objets de base :



Et le plus important, notre mesh :



Ici, le mesh avec les armatures, lui permettant d'être animé... Ce qui n'est pas encore fait.



6.1 Le son

La première étape est d'initialiser la carte son de l'utilisateur à l'aide de ces lignes :

```
dsound := Device.Create();
  dSound.SetCooperativeLevel(my_form.handle, CooperativeLevel.Priority);
```

Le CooperativeLevel correspond ici à l'interaction de notre fenêtre de jeu ("my_form.handle" ici) avec les autres fenêtres de windows. Nous avons choisi "Priority", elle permet de (comme son nom l'indique...) donner la priorité aux sons de notre jeu face aux autres applications de Windows (celle-ci n'empêche en rien qu'une autre application puisse jouer des sons).
La deuxième étape est d'initialiser le buffer "son" (place réservée, ici, aux sons que l'on charge dans la mémoire).

```
d.ControlPan := true;
  d.ControlVolume := true;
  d.ControlFrequency := true;
  d.ControlEffects := true;
  d.GlobalFocus := true;
```

Le ControlPan permet de gérer la balance du son gauche/droite. Le ControlEffects permet d'appliquer des effets à nos sons. Le GlogbalFocus permet de continuer à DirectX de jouer les sons même si notre fenêtre de jeu n'est pas au premier plan.
Enfin, la procédure qui nous permet de jouer un son :

```
procedure audioDevice.play;
begin
   sound.Play(0, bufferplayflags.Default);
end;
```

Le 0 correspond ici à la priorité du Playback (c'est la valeur par défaut), le ".Default" signifie que l'on jouera une seul fois le son (".Looping" aurait jouer le son en boucle).

6.2 site web

Une première ébauche du site web a été réalisée. Cependant, certains membres du groupes ne le trouvant pas très beau, celui-ci sera refait. Voici un screenshot de ce qui a été réalisé pour le moment :

Son adresse : http ://la.cosa.nostra.free.fr.

Conclusion

Nous avançons donc petit à petit. Les connaissances s'accumulent au fur et à mesure à travers nos recherches. Les tâches se compliquent de plus en plus à travers le temps, mais notre motivation est sans limite et notre soif de connaissance s'accentue chaque nuit passée sur notre projet. Ce que nous avons fait nous encourage encore plus à poursuivre nos efforts. Notre petite "entreprise" continue donc sur sa lancée, prête à affronter de nouveaux défis.