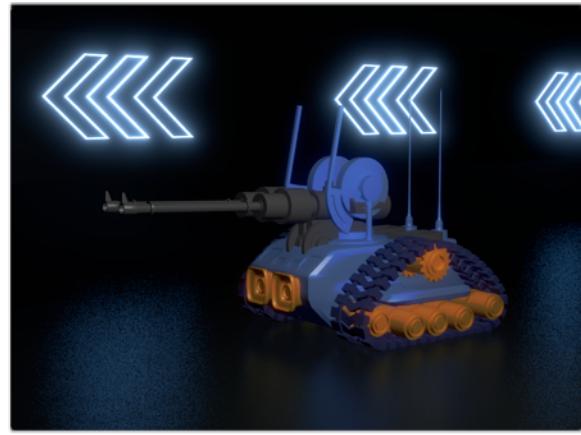


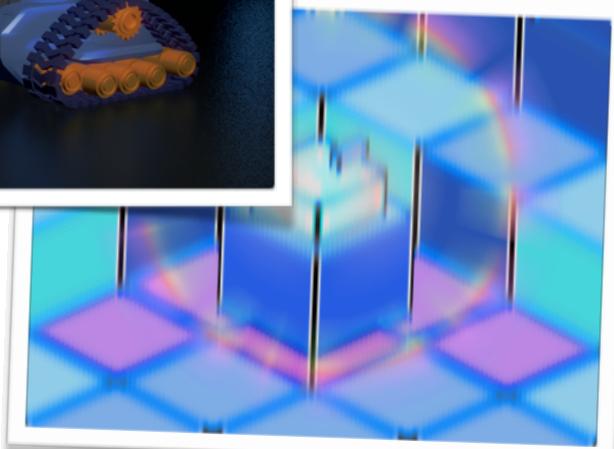
# MagnetiK TanK

ReconfigureD



```
struct SMouseState {
    core::position2di Position;
    bool LeftButtonDown;
    bool appui;
    SMouseState() : LeftButtonDown(false), appui(false) {}
} MouseState;

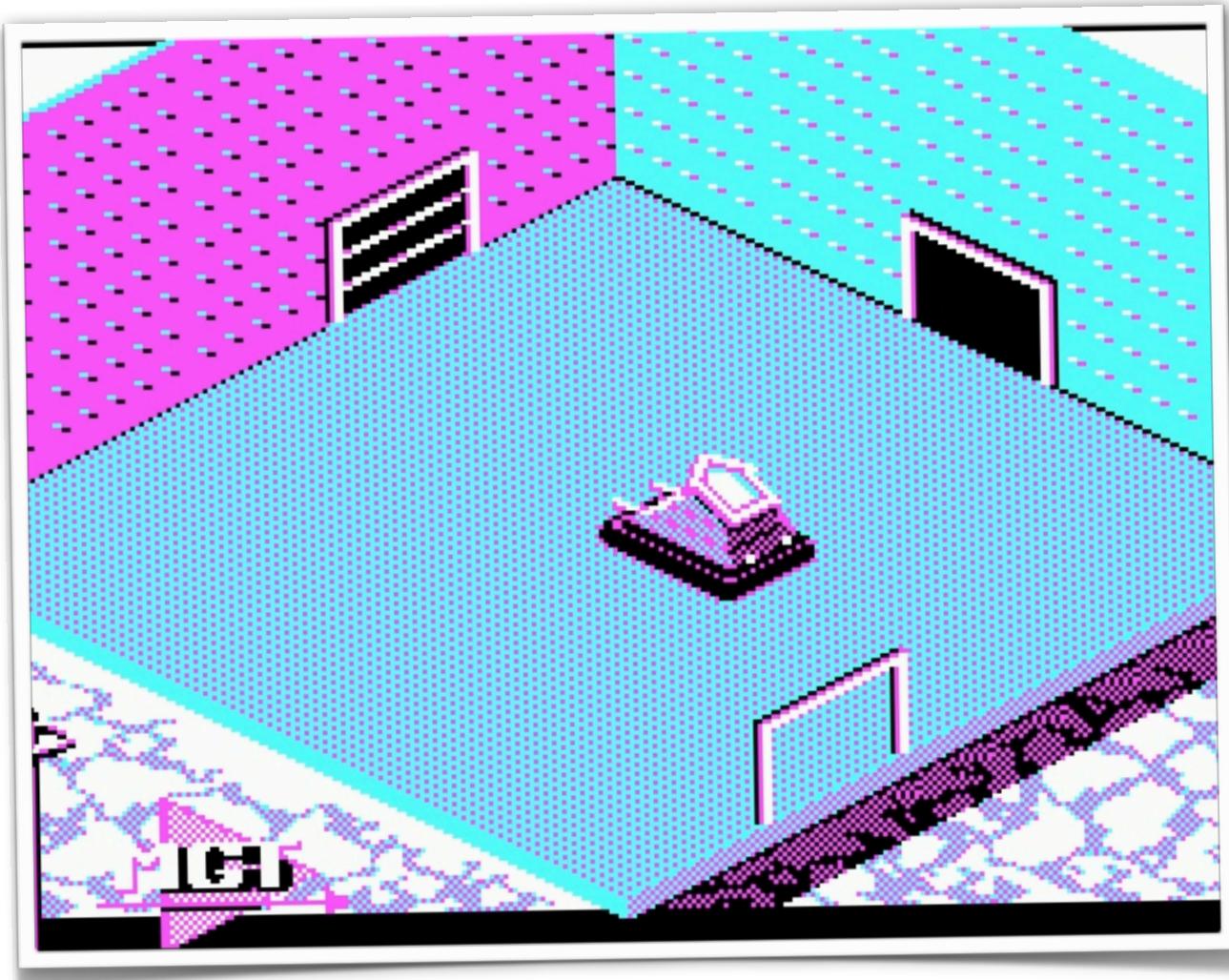
virtual bool OnEvent(const SEvent &Event,
const SEvent::SJoystickEvent &GetJoystickEvent,
const SMouseState &GetMouseState)
```



TD n°3

Valentin Mercier  
Arthur Busser

# Introduction



**Objectif de ce projet d'informatique :** Ce projet a pour principal but pédagogique l'apprentissage de la bibliothèque graphique Allegro. Il nous est demandé de réadapter **en un mois** le jeu Magnetik Tank développé en 1986 par la société Loriciels sur les ordinateurs actuels. Cependant le cahier des charges reste quand même assez souple et nous permet d'y apporter notre touche personnelle. C'est pourquoi nous avons décidé de baptiser notre projet : MagnetiK TanK ReconfigureD. Ce nom est le symbole de notre désir d'apporter notre imagination à ce merveilleux jeu.

**Présentation du jeu originel :** Afin de bien réaliser notre projet, nous avons jugé utile d'étudier le fonctionnement du jeu de Loriciels. Nous l'avons donc téléchargé sur leur site, puis nous avons pu le lancer grâce à un émulateur Windows 95. Nous avons ainsi pu noter plusieurs points importants que nous nous sommes efforcés de respecter.

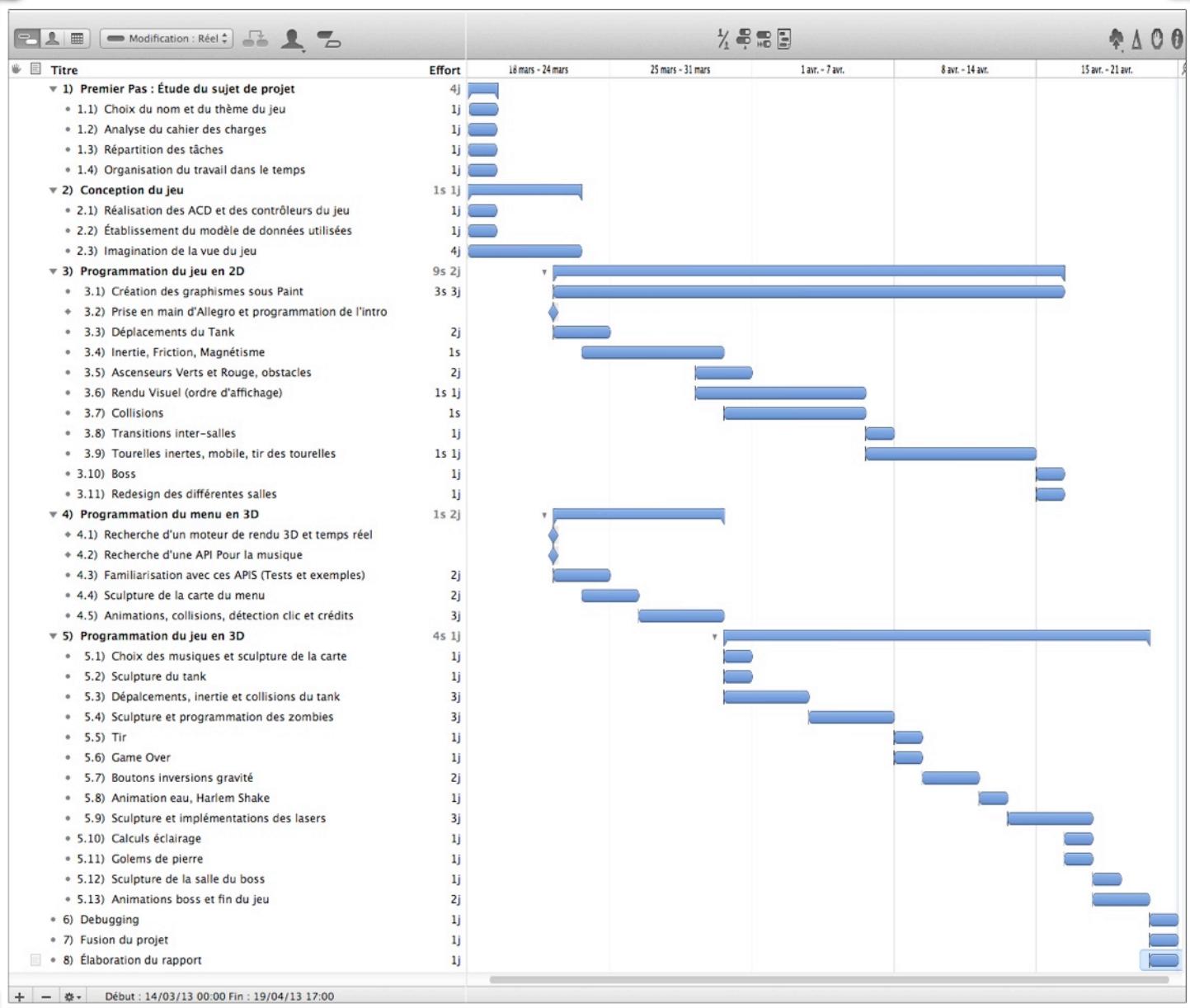
- L'aspect visuel du jeu nous donne une impression de vue en trois dimensions. On appelle cela de la 3D isométrique
- Le tank possède une inertie rendant ainsi le contrôle des déplacements plus séduisant.
- Le jeu ne possède pas de checkpoint, quand l'on perd on recommence au début.
- Le jeu est un labyrinthe de plusieurs salles avec un boss final. Des clés sont nécessaires pour ouvrir certaines salles.
- Enfin la complexité du jeu est accrue par la présence d'estrades en verre dans certaines salles.

## Présentation du groupe

Nous sommes deux élèves, **Valentin Mercier** et **Arthur Busser**, en première année du cycle préparatoire intégré de l'école d'ingénieurs ECE Paris. Notre curiosité et notre désir d'apprendre ont fait que nous avons sans cesse pris part au travail de l'autre, soit en donnant des conseils, soit en posant des questions sur comment certains modules étaient codés. Chaque fonctionnalité de notre projet a été vérifiée et améliorée par nos nombreuses discussions. C'est pourquoi nous sommes tous les deux fiers de notre travail.

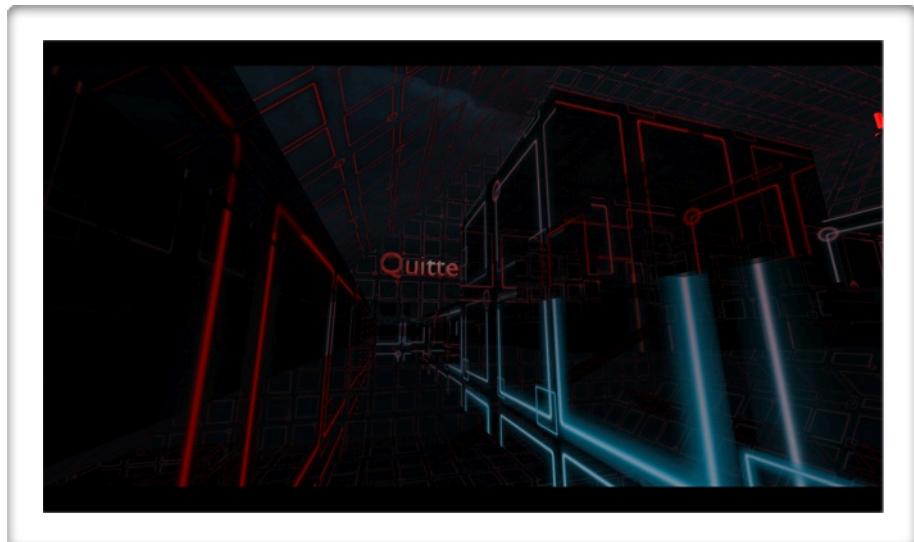
Malgré notre volonté de toucher à tout il a bien fallu que l'on se répartisse les tâches. C'est pourquoi après avoir décidé que nous réaliserons une version 3D et une version 2D de ce jeu nous avons établi qu'Arthur ferait toute la logique du jeu sous Allegro ainsi que le design de nombreux éléments graphiques du jeu 3D (tank, lasers, golems), tandis que Valentin s'occupera de la partie vue du jeu sous Allegro ainsi que de la logique du jeu en 3D.

# Planning



# Présentation du jeu

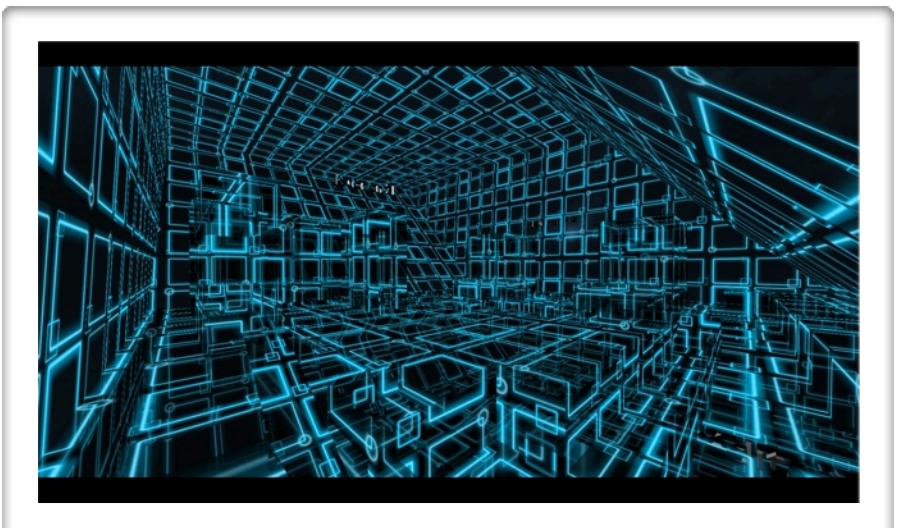
**Le menu principal**, est un environnement en trois dimensions dans lequel on peut se déplacer à l'aide des touches Z, Q, S et D ainsi que la barre d'espace.



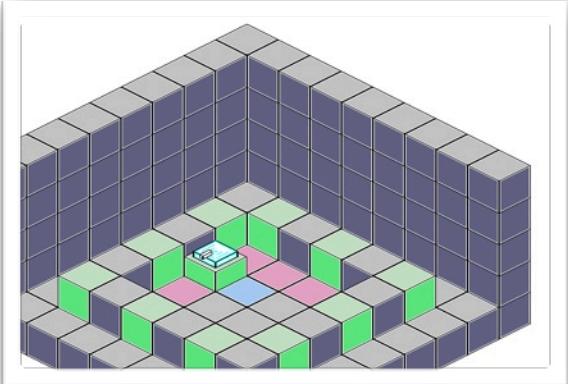
Il faut trouver du texte flottant en l'air indiquant chacune des options du menu. Le clic gauche de la souris permet ensuite de valider l'option.

Il y a quatre options dans le menu :

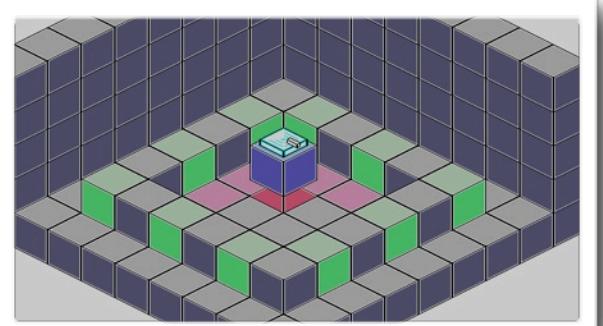
- Jouer 2D
- Jouer 3D
- Crédits
- Quitter



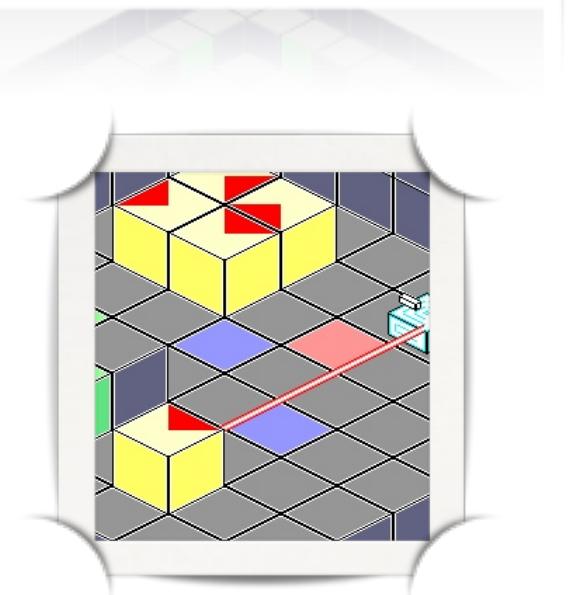
**Le jeu sous Allegro :** Le jeu 2D fait sous allegro met en place un tank au design très sobre et futuriste perdu dans un puzzle de salles dans lesquelles nous pouvons trouver des ascenseurs et des tourelles.



Le joueur contrôle le tank à l'aide des touches fléchées. Il remarquera que le tank possède une inertie assez particulière à laquelle vient s'ajouter un phénomène de magnétisme. En effet chaque dalle agit comme un aimant attirant ainsi le tank vers son centre quand celui-ci passe sur elles. Le but du jeu est de résoudre l'énigme de chaque salle afin de pouvoir accéder à l'ascenseur menant à la salle suivante.

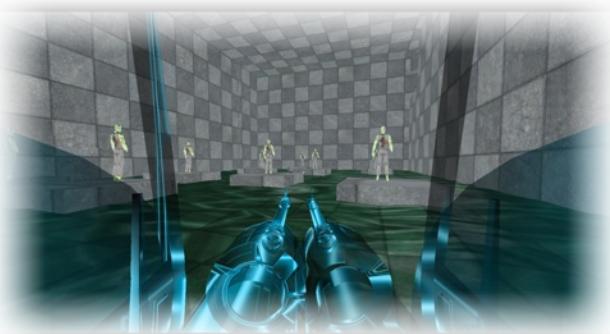


Il y a aussi des tourelles aux lasers mortels qu'il faudra éviter afin de pouvoir se hisser jusqu'à la dernière salle : celle du boss.  
La partie se finit quand l'on bat le boss



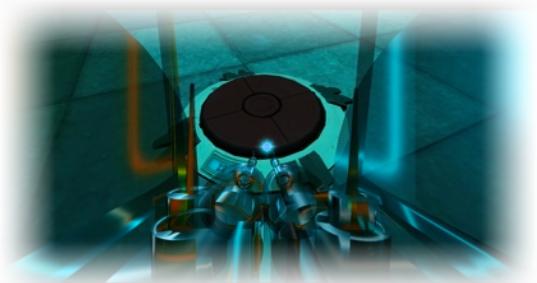
**Notre bonus, un jeu réalisé en trois dimensions :** Nous voulions faire un projet original. C'est pourquoi nous nous sommes dit : «Pourquoi ne pas faire un jeu en réelle 3D?» Et nous l'avons fait.

Le jeu en 3D ressemble globalement à celui que nous avons réalisé sous Allegro. Le joueur est embarqué dans un tank avec une redoutable inertie. La carte s'apparente à un réel labyrinthe où il faut parfois inverser la gravité pour sortir de certaines salles. Le joueur doit trouver la sortie afin de pouvoir affronter le boss et terminer le jeu. Nous avons repensé nos ennemis afin d'y ajouter un peu de folie. C'est pourquoi le joueur devra faire face à des zombies, éviter des lasers mortels et triompher des golems de pierre.



Pour cela le joueur pourra compter sur son stock illimité de munitions, pouvant assommer des zombies par dizaines et éclairer les recoins les plus sombres et les plus mystérieux du labyrinthe.

Une fois sorti du labyrinthe, le joueur se fera agresser par un méchant golem de pierre (boss) qui en profitera pour lui voler son tank. Ainsi un combat épique commencera entre les deux protagonistes. Tandis que le golem utilisera les canons du tank pour attaquer il faudra trouver et activer des mines pour faire exploser l'arène et mettre ainsi fin au jeu.



# Choix de programmation

**DÉVELOPPEMENT DU JEU 3D :** Nous avons utilisé le moteur de rendu 3D Irrlicht pour notre jeu en trois dimensions. Irrlicht présente plusieurs avantages. Il est développé en C++ qui est un langage très proche du C. Ainsi relativement aisément orienté objet ce structurer notre lire de nombreux images (jpeg, png, fichiers 3D (3ds, md2, obj). Enfin il est totalement open source et libre de droit. C'est pourquoi nous n'avons pas hésité une seconde à le choisir.



sa prise en main fut  
Il est totalement  
qui permet de bien  
code. De plus il peut  
types de fichiers  
tga, bmp) et de

## Modèle des données :

Données	Traitements
Manager de Scène	Objet parent de tous les autres objets. Crée le rendu 3D à partir des positions, éclairages et animations des autres objets
Zombies	S'animer, se déplacer vers le tank, mourir, tuer le joueur
Boutons gravité	Inverser ou rétablir la gravité
Lasers	Tuer le joueur
Golems	Grogner, se déplacer vers le tank, tuer le joueur, s'envoler
Lumières	Éclairer la carte, s'allumer, s'éteindre
Munitions	Éclairer la carte, infliger des dégâts
Labyrinthe	Créer des collisions

Données	Traitements
Particules dorées	Bouger aléatoirement
Donut	Animer ses textures violettes, créer des collisions, tourner,
Mines	Exploser le donut, mettre fin au jeu
Halo violet	Animation visuelle dans le donut
Tank	Se déplacer, avoir de l'inertie
Caméra	Embarquée dans le tank, détermine ce que l'on doit afficher à l'écran
Manager de Son	Objet parent à tous les autres sons, joue de la musique
Musiques	Charger, régler son volume, être jouées par le manager de son
Évènements	Déclencher un évènement, déterminer si un évènement a déjà eu lieu.
Curseur	Se projeter contre les murs

## Analyse Chronologique et descendante du jeu

- Initialisation
  - Chargement des musiques
  - Chargement des modèles 3D
  - Chargement des textures
- Boucle de jeu (tant qu'on est pas mort ou qu'on appuie pas sur échap)
  - Déetecter et lancer un évènement si nécessaire (ex : Arrivée du boss, changement de musique etc...)
  - Repositionner la caméra dans le tank
  - Positionner le curseur bleu sur le premier polygone vu par la caméra
  - Tirer si l'on appuie sur le bouton gauche de la souris
  - Infliger des dégâts si les tirs atteignent le zombie
  - Déplacer les zombie

- Déplacer le tank en calculant aussi l'inertie
  - Si l'on est contre le boss
  - Faire tirer le boss
  - Activer une mine si l'on est proche d'elle
  - Faire exploser l'arène si toutes les mines sont activées
- Swapper le buffer à l'écran
- Déetecter si l'on est mort soit à cause des zombies/golems soit à cause des lasers
- Menu Quitter ou Rejouer
  - Afficher un fondu rouge progressif pour faire comprendre au joueur qu'on est mort
  - Si quitter alors on libère toute la mémoire et on quitte
  - Si rejouer alors on réinitialise tout la mémoire et on recommence une partie

### **Découpage modulaire :**

**main.cpp** Lance le menu 3D, récupère l'option choisie. Permet ensuite le lancement du jeu 2D ou du jeu 3D

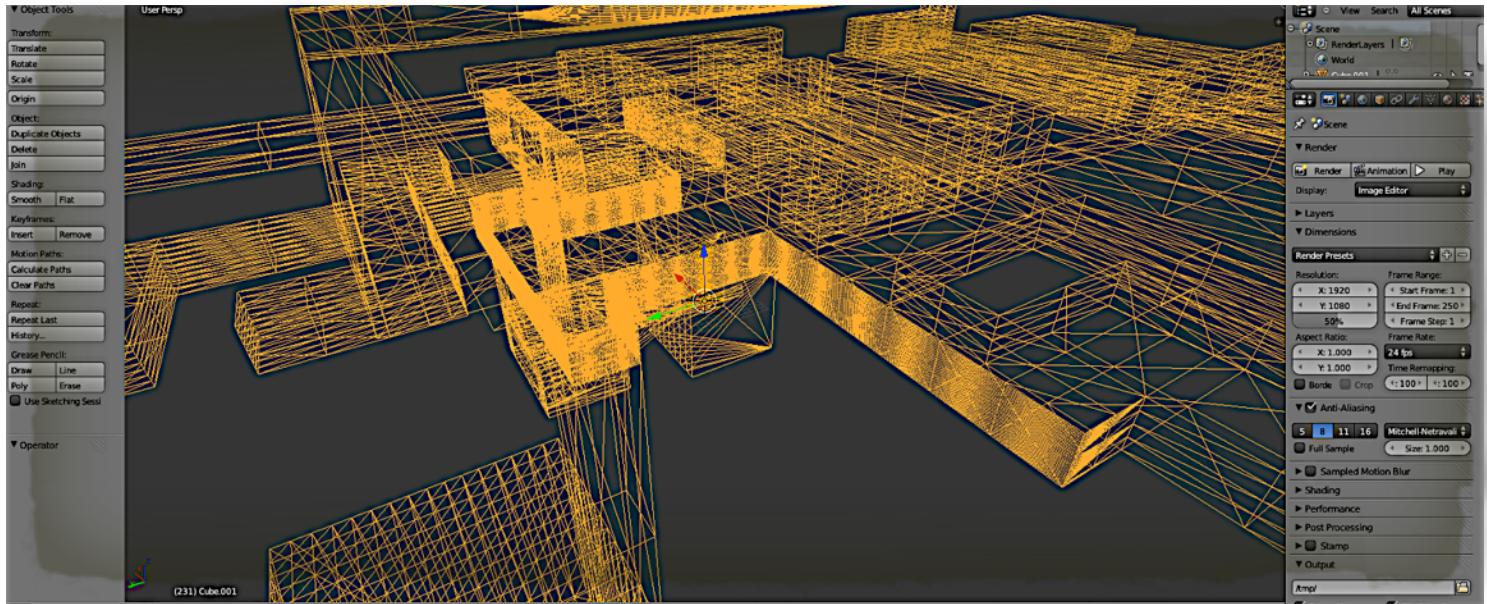
**DetectionSouris.cpp** Classe permettant de détecter les clics des souris. Elle sera ensuite intégrée dans le manager de scène afin qu'il puisse détecter les clics de souris

**LectureMedia.cpp** Toutes les ressources sont cryptées dans un énorme gros fichier appelé media ne possédant aucune extension. Ce module s'occupe de la vérification de l'intégrité de ce fichier et permet d'obtenir des données empaquetées dans ce fichier (ex : musiques)

**MenuPrincipal.cpp** Grande classe contenant les sous-programmes d'initialisation, d'affichage, de déplacement et de sélection d'option du menu

**Jeu3D.cpp** Grande classe contenant tous les sous-programmes nécessaires au fonctionnement du jeu.

**Graphismes du jeu en 3D** Nous avons utilisé deux logiciels très utiles afin de pouvoir créer nos modèles 3D ainsi que leur textures : Blender et PhotoShop



Une vue de notre carte 3D dans Blender, 12672 polygones ;)

**Implémentation des musiques** Pour pouvoir jouer des musiques pendant le jeu nous avons utilisé la bibliothèque IrrKlang réalisée par les mêmes développeurs que IrrLicht. Son utilisation est aussi libre de droit et étant aussi association avec le

développée en C++ son code IrrLicht fut très



**DÉVELOPPEMENT DU JEU SOUS ALLEGRO** Nous avons eu à cœur de bien organiser notre code en structures et tableaux de pointeurs. Nous l'avons aussi orienté de manière très modulaire afin qu'une fois réalisé il puisse nous permettre de modifier nos pièces et nos éléments très facilement. Sa lisibilité s'en est retrouvé accrue. Ainsi il fut facile pour nous de pouvoir jeter des coups d'œil au code de l'autre.

### Structure principale du manager de carte

```
typedef struct map
{
    int x;
    int y;
    int z;
    int ** ground;
    int ** northWall;
    int ** westWall;
    ELEVATOR *** elevators;
    TURRET *** turrets;
    LASER *** lasers;
} MAP;
```

### Autres structures utiles

```
typedef struct elevator
{
```

```
    int x;
    int y;
    int z;
    int electrical;
    int on;
    int up;
    int delai;
    int delaiMax;
    int zMax;
    int zMin;
```

```
} ELEVATOR;
```

```
typedef struct turret
{
```

```
    int x;
    int y;
    int z;
    int zMin;
    int zMax;
    int direction;
    int on;
    int canMove;
    int canRotate;
    int delai;
    int delaiMax;
```

```
} TURRET;
```

```

typedef struct laser
{
    int x;
    int y;
    int z;
    int direction;
    int delai;
    int delaiMax;
} LASER;

typedef struct tank
{
    int userControlsMove;
    int userControlsRotate;
    int x;
    int y;
    int z;
    int vx;
    int vy;
    int vz;
    int direction;
    int nbSprites;
    BITMAP ** sprites;
    int alive;
} TANK;

```

## Modèle de données

Données	Traitements
Map	Contient tous les éléments de la scène
Ascenseurs	Monter, Descendre, changer de salle
Tourelles	S'animer, rechercher le tank, tirer, tuer
Tank	Se déplacer, inertie, magnétisme
Obstacles	Crée un système de collisions
Boss	Tuer, s'animer, gérer la salle du boss

## Analyse Chronologique et descendante du jeu

- Initialisation

- Chargement des BITMAPS
- Chargement de la map
- Chargement du tank

- Boucle de jeu (tant qu'on est pas mort ou qu'on appuie pas sur échap)
  - Calcul de la vitesse en fonction des commandes, de la friction, du magnétisme et des collisions, dans cet ordre.
  - Calcul de la position en fonction de la vitesse.
  - Calcul de l'orientation du tank.
  - Gestion des ascenseurs
    - Activer/Désactiver les ascenseurs si le tank passe dessus
    - Actualiser la hauteur des ascenseurs
    - Actualiser la hauteur du tank
  - Gestion des tourelles
    - Désactiver les tourelles si le tank passe dessus
    - Actualiser la hauteur des tourelles
    - Actualiser la hauteur du tank
    - Tourner les tourelles rotatives
    - Tirer sur le tank s'il passe devant
    - Enlever les lasers qui sont l‡ depuis un moment.
- Affichage
  - Image de fond
  - Blocks, lasers, tank
  - Passage du buffer à l'écran
- Transitions
  - Assombrir si changement de salle
  - Changer de salle si nécessaire
  - Animation de game-over si tank mort
- Libération mémoire de la map et du tank en fin de partie

## Découpage modulaire

Notre code s'organise en plusieurs modules

**main.c** Lance le gestionnaire de jeu

**Main\_Game\_Functions.c** Gestionnaire de sous-programmes et les Menu et jeu

**Load\_Functions.c** Chargement et allocations mémoire

**Free\_Functions.c** Libérations mémoire

**Elevator\_Functions.c** Fonctions concernant les ascenseurs

**Turret\_Functions.c** Fonctions concernant les tourelles

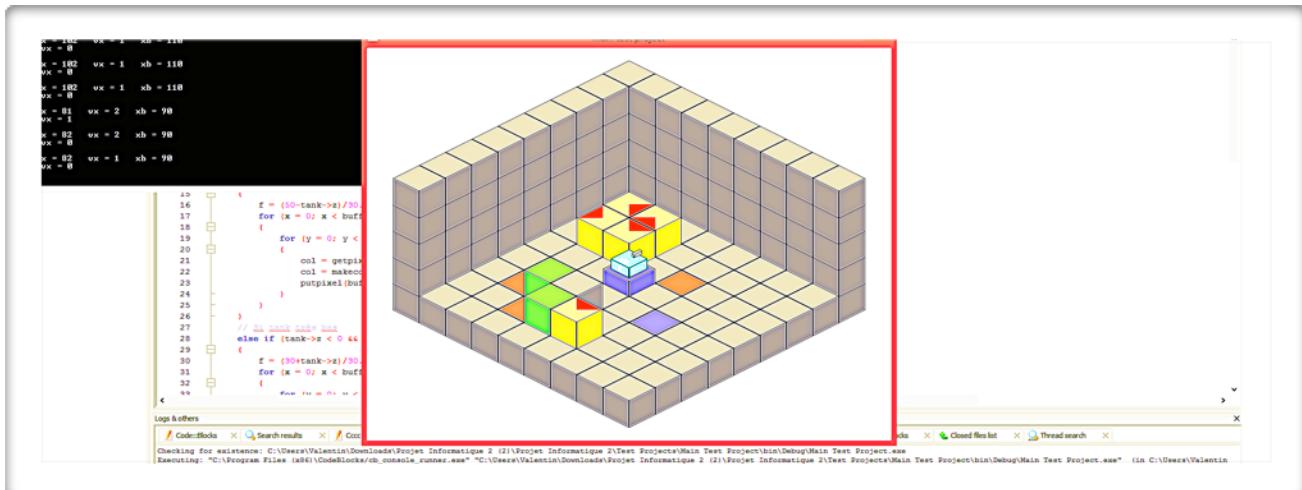
**Laser\_Functions.c** Les lasers des tourelles

**Calculate\_Functions.c** Les calculs en général (inertie, magnétisme, collisions)

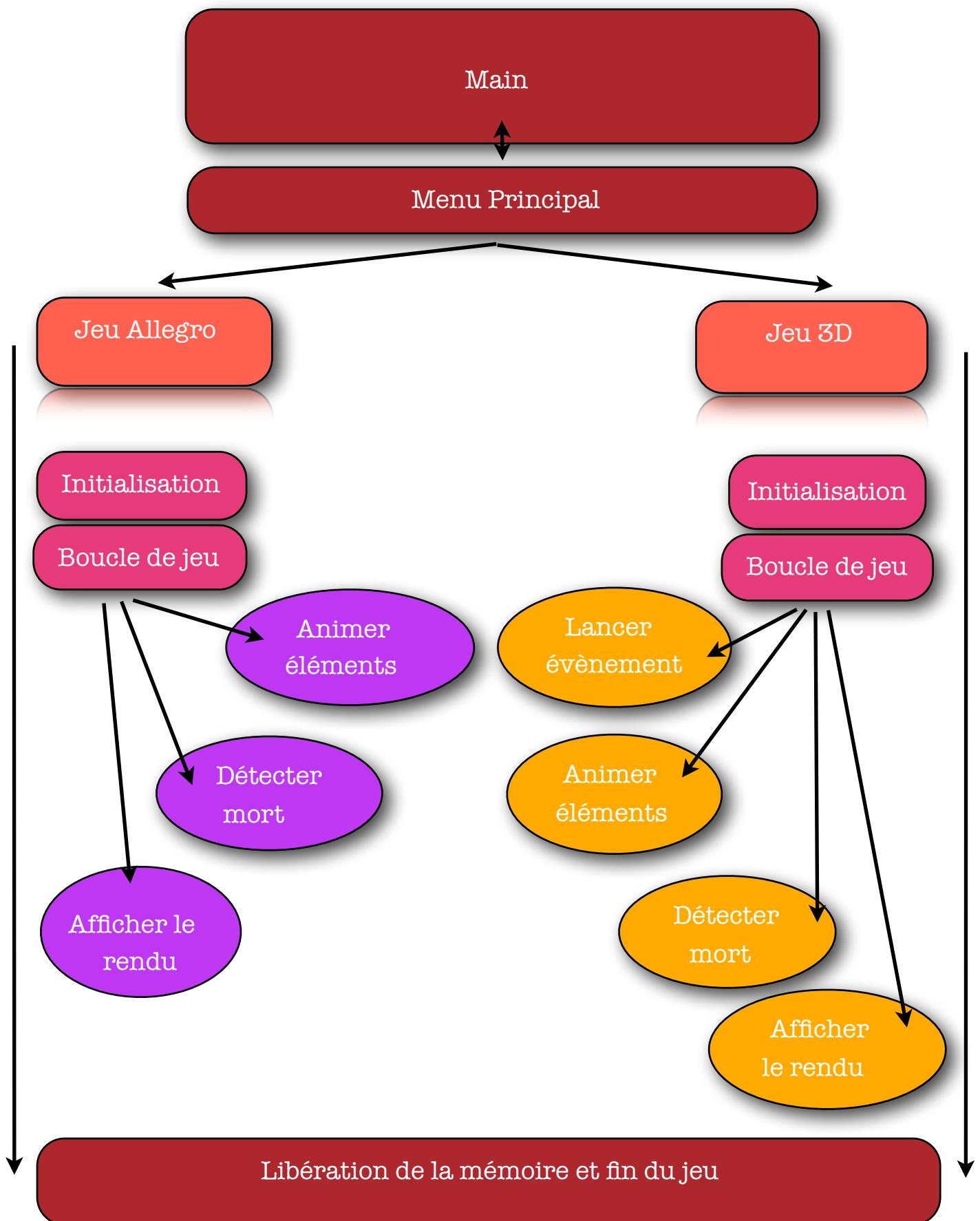
**Render\_Functions.c** Affichage

**Transition\_Functions.c** Transitions entre les salles et les game-overs

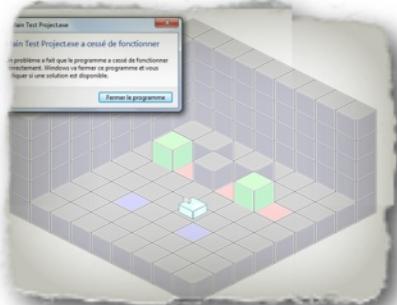
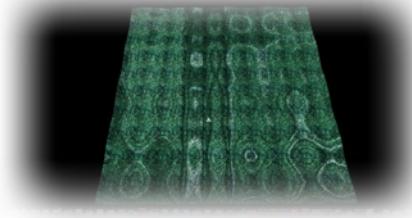
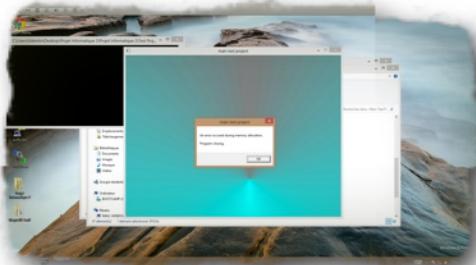
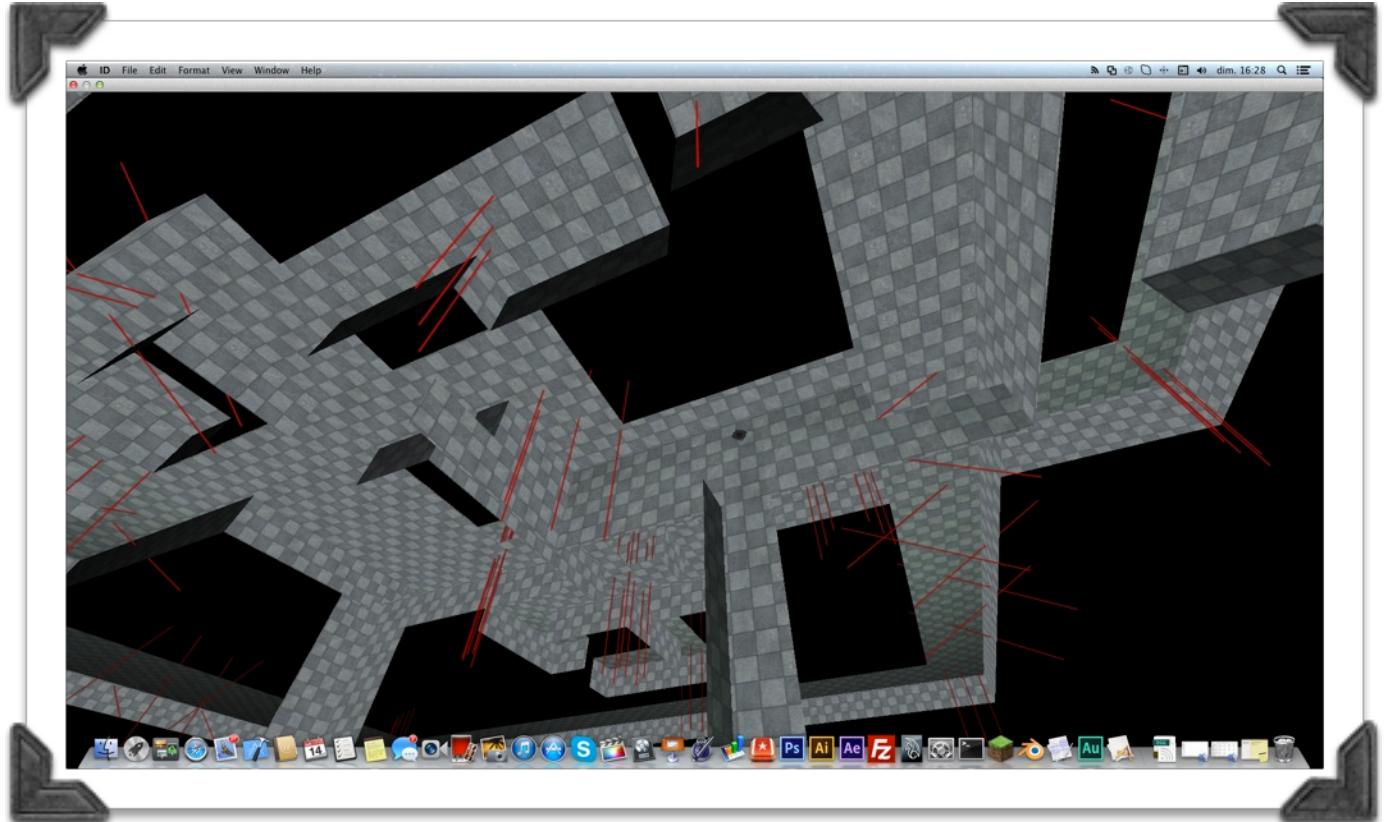
**Misc\_Functions.c** Autre. Ex: affichage de messages.



## Graphe d'appel représentant le fonctionnement de notre jeu



**Problèmes rencontrés :** Nous avons du faire face à de nombreux problèmes lors du développement de notre jeu. Les bugs les plus récurrents furent les erreurs de segmentation et les bugs d'affichages. Les capture d'écran qui suivent parlent d'elles mêmes.



# Bilans Personnels

**Valentin Mercier,** J'ai beaucoup aimé le sujet de ce projet car il nous a permis d'approfondir notre maîtrise de la bibliothèque graphique Allegro. De plus, étant assez libre, il nous a permis d'avoir l'audace de réaliser un jeu en 3D et ainsi découvrir de nouvelles notions de calcul de rendus visuels, de nouveaux logiciels (Blender) et de nouvelles APIs (IrrLicht & IrrKlang). Travailler avec Arthur fut un plaisir car nous nous entendons très bien et je trouve que nos compétences se complètent. Nous formons une très bonne équipe.

**Arthur Busser,** Ce projet n'était pas le premier jeu vidéo sur lequel j'ai travaillé, mais c'était la première fois que mon travail se faisait au niveau du code et pas simplement de la conception. Utiliser Allegro m'a permis de découvrir la complexité des fondations d'un jeu vidéo, aussi simple que ce jeu puisse paraître. La liberté du sujet a réellement été un point positif, donnant carte blanche à notre créativité. Valentin avait plus d'expérience que moi dans le codage de jeux, mais j'ai néanmoins su me rendre indispensable avec mes connaissances sur la modélisation 3D et sur la conception globale de jeux. Au final, ce projet m'aura beaucoup appris et m'aura donné davantage confiance vis-à-vis des projets, que ce soit un travail individuel ou en équipe.

**Notre Tank après avoir été modélisé pour la première fois dans un environnement virtuel**

