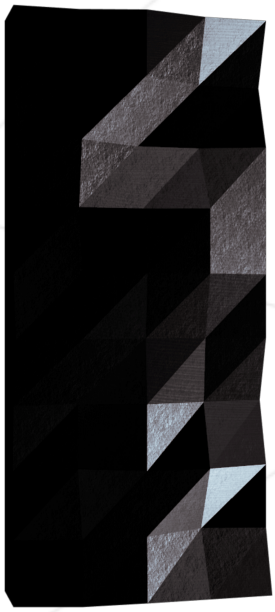

# DooM-Nukem' 3D

Pedago Team pedago@42.fr

*Résumé: A la suite du wolf3D, vous êtes désormais dans un univers plus complexe fait de pieces aux formes variées, parsemmé de décors et mobilier, rempli d'adversaires.*


# Table des matières

# Chapitre I

# Préambule

A la suite de `Wolfenstein 3D`, John Carmack et John Romero ont poursuivit leur aventure commune. Une fois avoir réalisé quelques niveaux supplémentaires pour des éditions spéciales de `Wolf3D`, ils se sont attaqués à la suite. Plus gros, plus grand, plus fort. Fin 1993, ils publient `Doom`, toujours un shoot'em up à la première personne. L'immersion est encore plus impactante, l'ambiance est encore plus prenante, et `Doom` devient immédiatement, et reste toujours, une référence en matière de jeux vidéos.

Dans les locaux sur Mars de l'UAC, l'Union Aerospace Corporation, vous incarnez un ancien marine qui doit nettoyer la base de tous un tas de monstres venus des enfers. Ceux-ci sont apparus à la suite d'expériences radioactives ayant créé une déchirure de l'espace-temps. Plus moches les uns que les autres, et jusqu'au gigantesque boss final, il vous faut toutes les armes disponibles pour en venir à bout. Ultra violent, rythmé par du Heavy Métal, même aujourd'hui il est indispensable d'y avoir joué au moins une fois dans sa vie.

En parallèle des aventures d'ID Software, un autre univers s'est construit dans une autre société de jeux vidéos : Apogee Software. Dès 1991, dans un Los Angeles envahit par les extraterrestres, `Duke Nukem` se bat pour les mettre au tapis. Il s'agit alors uniquement d'un jeu de plateforme en 2D à la Mario. Après un passage dans l'espace pour une seconde aventure en 1993, on retrouve en 1996 un troisième jeu qui cette fois-ci est en 3D et à la première personne. Avec davantage d'effets techniques que `Doom`, des décors très cartoon, du sexe, de la violence, des répliques cultes, et beaucoup de second degré, ce troisième opus est une vraie réussite. Edité sous la marque 3d Realms d'Apogee Software, `Duke Nukem 3D` se place comme successeur légitime de Doom dans la lignée des jeux en fausse 3D avant l'avènement des GPU et de l'accélération matérielle pour faire de la 3D complète.

A nouveau, il est temps pour vous de revivre l'Histoire...

# Chapitre II

# Introduction

C'est un mélange de `Doom` et de `Duke Nukem 3D` qu'il vous faudra réaliser. De nombreux éléments fonctionnels de base sont identiques. Ils n'ont cependant pas été implémentés de la même façon par les concepteurs des deux jeux. A vous de trouver celle qui vous conviendra et qui vous permettra d'ajouter les features demandées comme celles que vous souhaitez y voir.

Toujours en utilisant la technique du raycasting, il vous faudra davantage de travail que votre `Wolf3D`. Ce qui n'était que bonus précédemment devient élément à part entière du sujet.

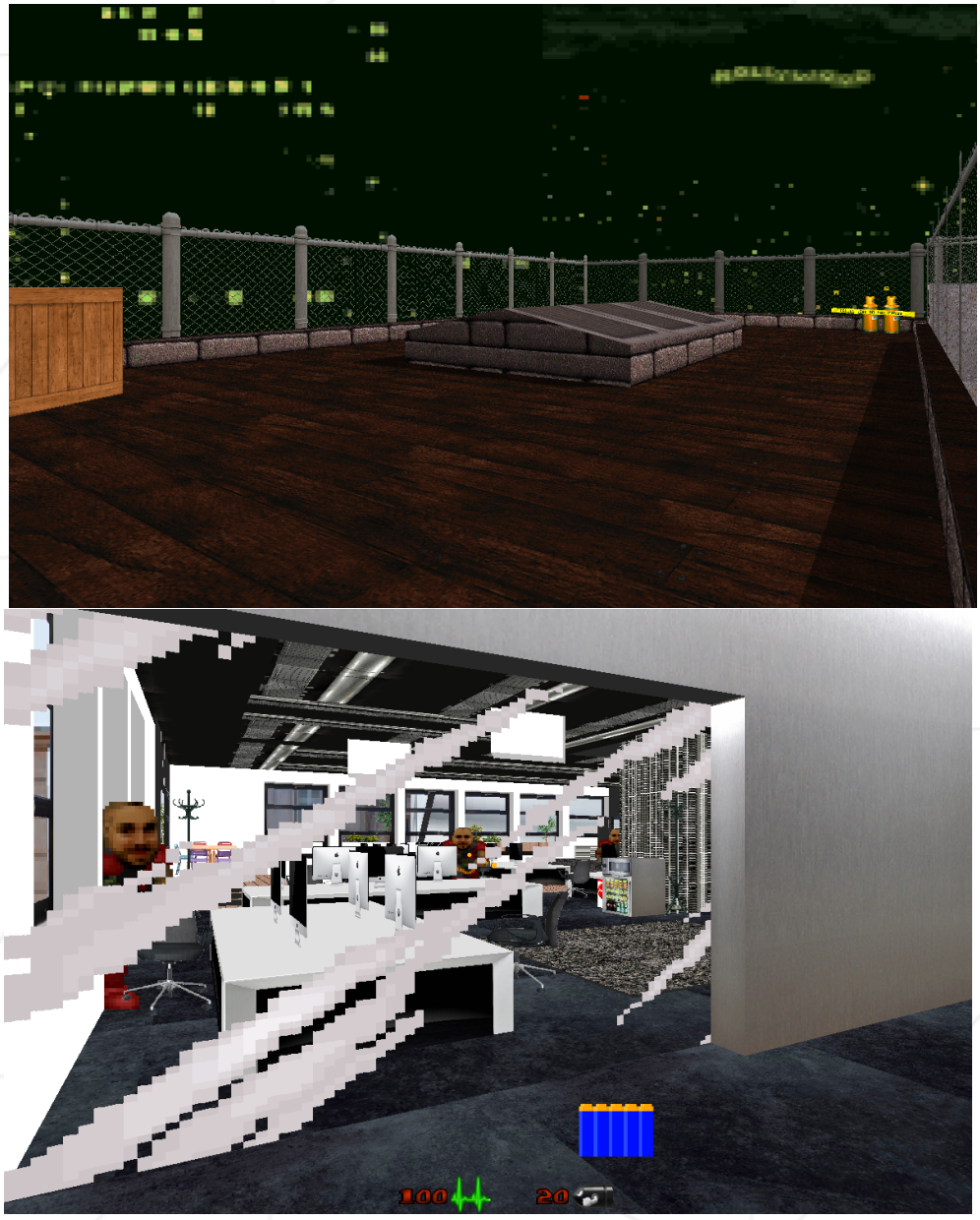


FIGURE II.1 – Utilisation du raycasting pour le rendu graphique du projet `Doom-Nukem`.

# Chapitre III

# Objectifs

Les objectifs de ce projet comprennent toujours les objectifs communs à tous les premiers projets ayant lieu avant le stage : rigueur, pratique du `C`, pratique d'algos classiques, recherche d'infos, exploitation de documentations, etc.

Étant un projet de programmation graphique, `Doom-Nukem` vous permettra également de consolider vos acquis specifiques à cette discipline : les fenêtres, les images, les events, remplir des formes, etc. Mais ça devrait commencer à ne plus être un problème...

# Chapitre IV

# Consignes générales

- Ce projet ne sera corrigé que par des humains. Vous êtes donc libres d'organiser et de nommer vos fichiers comme vous le désirez, en respectant néanmoins les contraintes listées ici.
- L'exécutable principal doit s'appeller `doom-nukem`.
- Vous devez rendre un `Makefile`. Ce `Makefile` doit compiler le projet, et doit contenir les règles habituelles. Il ne doit recompiler et relinker le programme qu'en cas de nécessité.
- Si vous êtes malin et que vous utilisez votre biliothèque `libft` pour votre `Doom-Nukem`, vous devez en copier les sources et le `Makefile` associé dans un dossier nommé `libft` qui devra être à la racine de votre dépôt de rendu. Votre `Makefile` devra compiler la bibliothèque, en appelant son `Makefile`, puis compiler votre projet.
- Les variables globales sont interdites.
- Votre projet doit être en `C` et à la Norme. La norminette fait foi.
- Vous devez gérer les erreurs de façon raisonnée. En aucun cas votre programme ne doit quitter de façon inattendue (segmentation fault, bus error, double free, etc...).
- Votre programme ne doit pas avoir de fuites mémoire.
- Vous devez rendre, à la racine de votre dépôt de rendu, un fichier `auteur` contenant vos login suivi d'un '\n' :

```
$>cat -e auteur
wlogin$
xlogin$
ylogin$
zlogin$
$>
```

- Vous pouvez utiliser la `MinilibX` native `MacOS` qui est sur les dumps, ou alors, vous pouvez utiliser la `MinilibX` à partir de ses sources qui seront à intégrer de la même manière que celles de la `libft`. Dernière possibilité, vous pouvez aussi utiliser d'autres bibliothèques d'affichage (`SDL`, etc...). Si la bibliothèque que vous utilisez n'est pas installée sur les machines de cluster, vous devrez fournir les sources de cette bibliothèque dans votre rendu, et elle devra se compiler automatiquement

exactement comme la `MinilibX` ou votre `libft`, sans autre action que de compiler votre rendu. Quelle que soit la bibliothèque d'affichage, vous ne devez utiliser que ses fonctions de dessin basiques similaires à celles de la `MinilibX` : ouvrir une fenêtre, manipuler des images à partir du buffer mémoire, et gérer les évènements.

- Dans le cadre de ce projet, vous avez le droit d'utiliser tout appel système de la libC (man 2), ainsi que `malloc`, `free`, `perror`, `strerror`, `exit`, toute la lib math (`-lm`), toutes les fonctions de la `MinilibX` ou leurs équivalents dans une autre bibliothèque d'affichage. On vous laisse recoder un reader de png ou tga (ou autre), si vous ne l'avez pas déjà fait pour le `wolf3d`.
- Vous avez l'autorisation d'utiliser d'autres fonctions, voire d'autres bibliothèques, dans le cadre de vos bonus à condition que leur utilisation soit dûment justifiée lors de votre correction. Soyez malins.
- Vous pouvez poser vos questions sur le forum, sur slack, ...

# Chapitre V

# Ce qu'il faut faire

Tout le fun de ce projet réside dans ses fonctionnalités liées au gameplay. Mais avant de compter les points face aux monstres et aliens ainsi que des flots d'XP à gagner, vous devez quand même bosser un peu. Une partie du boulot a heureusement déjà été faite, il s'agit de votre `wolf3d` (notez que bien qu'indispensable, cette partie ne rapporte aucun point).

Vous devez créer la représentation graphique "réaliste" en 3D que l'on pourrait avoir à l'intérieur d'un univers en vue subjective. Vous devez réaliser cette représentation en utilisant le principe du Ray-Casting. Aucune accélération matérielle n'est permise, ni bibliothèque 3D.

On trouve comme fonctionnalités déjà présentes dans le `wolf3d` :

- Les flèches du clavier doivent permettre de se déplacer en temps réel dans l'univers, comme dans les deux jeux d'origine. Tourner à 360 degrés, avancer, reculer.
- Appuyer sur la touche `ESC` doit fermer la fenêtre et quitter le programme proprement.
- Cliquer sur la croix rouge sur la bordure de la fenêtre doit fermer la fenêtre et quitter le programme proprement.
- Il y a des textures sur les murs.

Les nouveautés sur le plan graphique :

- Il est possible de regarder vers le haut et vers le bas.
- Les espaces parcourus ont des formes quelconques, par exemple les pièces ont un nombre quelconque de murs avec toutes les orientations possibles.
- Le sol et le plafond ont une hauteur réglable, définissant ainsi des dénivelés entre différents espaces.
- Le sol et le plafond peuvent ne pas être horizontaux, mais des plans inclinés.
- Il y a des textures sur les sols et plafonds.
- Un ciel peut être présent à la place d'un plafond.
- Des murs peuvent être partiellement transparents, permettant d'entrevoir un espace derrière.
- Des éléments de décor peuvent être disposés sur les murs au dessus de la texture

principale (tableaux, posters ... ).

- Des objets/personnages peuvent être disposés dans l'univers, sous forme de sprite qui fait toujours face au joueur.
- Des objets/personnages peuvent être disposés dans l'univers, sous forme de multiples sprites affichés selon l'orientation entre l'objet et le joueur (voir un personnage de dos puis de profil lorsqu'on tourne autour).
- Chaque espace ou zone dispose d'un réglage de la luminosité ambiante, affectant les murs comme les objets qui s'y trouvent.
- Des messages texte peuvent être affichés pendant le jeu en surimpression.
- Un HUD doit être est présent (vie, monnaie, munitions, objets transportés ...) On entend par HUD plusieurs éléments disposés sur la vue du joueur et non pas un simple bandeau uniforme.

Sur le gameplay :

- La vue se fait à la souris de façon fluide, permettant de tourner sur soi-même à 360 et de regarder en haut et en bas.
- Il est possible de straffer.
- On interagit correctement avec les murs, les marches d'une hauteur raisonnable.
- On peut courrir, sauter, tomber, se baisser et se relever.
- Il est possible de voler ou nager (c'est pas bien différent...).
- Les objets présents peuvent bloquer le passage ou non, de façon proportionelle à leur représentation visuelle.
- Les objets présents peuvent être ramassés ou non, et alimenter un inventaire.
- Des interactions sont possibles avec des éléments du décor (murs, objets... ), à la fois par simple proximité (marcher sur une zone dangereuse) comme avec une action volontaire (appuyer sur un bouton).
- Des actions ou séries d'actions peuvent avoir lieu suite à des interactions, ou suite à des ramassages d'objets, ou bien de façon totalement indépendante, éventuellement timées.
- Les actions peuvent modifier tous les éléments du jeu, toutes les formes, toutes les propriétés (forme d'une piece, hauteur du plafond, texture d'un mur ou d'un objet, position d'une texture de porte pour son ouverture, etc).
- En corrolaire des 2 points précédents, votre jeu doit contenir des animations, des dispositifs du type ouverture de porte, utilisation de clef, ascenseur, et passage secret.
- Des personnages/objets peuvent avoir leurs propres actions, réactions, interactions dans l'univers.
- Pour reprendre l'univers de `Doom` et `Duke Nukem`, des projectiles peuvent être tirés avec interaction sur le décor, sur les objets, sur les personnages, y compris le joueur lui-même.
- Il y a un histoire avec une mission, un objectif à atteindre, un but (sauver la terre, devenir riche, ... même un truc simplissime, regardez Doom).
- Il y a un début et une fin de level.

- Il y a des bruitages.
- Il y a de la musique.

Autres considérations :

- Un éditeur de niveaux est indispensable. Il peut être intégré à l'exécutable principal et activé par une option en ligne de commande, ou bien il peut faire l'objet d'un binaire séparé. Il permet de définir complètement un level, les espaces, les dénivelés, les textures, les actions et interactions, etc.
- Comme dans les jeux originaux, vous devez packager dans un seul et unique fichier tous les éléments nécessaires au jeu : design du level, textures, éléments de gameplay. Un fichier par level est accepté, mais doit être autonome. Le binaire `doom-nukem` et le fichier d'un level quelconque doivent être autosuffisants.

# Chapitre VI

# Partie Bonus

Les bonus attendus :

- Un menu pour choisir le niveau ou des options de difficulté.
- Une esthétique, une ambiance, des décors soignés et détaillés.
- Une histoire et une scénarisation complexes, recherchées.
- De nombreuses interactions et animations useless qui participent à l'immersion dans l'univers (cabine téléphonique, distributeur de sodas, toilettes publiques, traces de projectiles sur les murs, etc).
- On peut jouer à plusieurs en réseau avec un lobby, des modes team/solo, un capture the flag, etc.
- Mode de rendu multithreadé avec l'utilisation de pthread.
- Plein d'autres trucs qu'on à mêmes pas imaginé !

# Chapitre VII

# Rendu et peer-évaluation

Rendez-votre travail sur le dépôt `GiT` du groupe comme d'habitude. Seul le travail présent sur ce dépot sera évalué.

Bon courage à tous et n'oubliez pas votre fichier auteur !