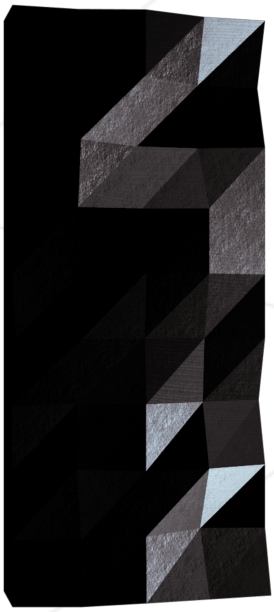


FDF

Fil De Fer

Résumé:
Ce projet consiste à créer graphiquement la représentation schématique d'un terrain en relief.

Version: 3

Table des matières

Chapitre I

Préambule

Voici ce que *Wikipédia* a à dire sur *Ghosts'n Goblins* :

Ghosts'n Goblins (Makaimura, Demon World Village) au Japon est un jeu vidéo de plateformes développé et édité par **Capcom** en 1985 sur borne d'arcade. Trois suites officielles virent ensuite le jour : *Ghouls'n Ghosts*, *Super Ghouls'n Ghosts* et *Ultimate Ghosts'n Goblins*. Ce premier opus a été porté vers de nombreuses plateformes. *Ghosts'n Goblins* est considéré comme l'un des jeux les plus difficiles de tous les temps.

- **Système de jeu**
  Ghosts'n Goblins est un jeu de plateformes où le joueur contrôle un chevalier, nommé Arthur, qui doit combattre des zombies, des démons et autres morts-vivants dans le but de sauver une princesse. Durant la partie, le joueur récolte diverses nouvelles armes, ainsi que des bonus et des pièces d'armure qui l'aideront dans sa tâche. Ce jeu est souvent considéré comme très difficile dans les standards du jeu d'arcade et cela vaut aussi pour les versions consoles.

- **Contrôles**
  La borne d'arcade permet au joueur de se diriger dans quatre directions grâce à un joystick huit directions, au côté duquel se trouvent deux boutons : l'un pour utiliser l'arme, l'autre pour sauter.

- **Vies**
  Le joueur débute avec trois vies et peut gagner des vies supplémentaires lorsqu'il dépasse 20 000 et 70 000 points. Une autre vie est offerte à chaque 70 000 points par la suite. Le personnage perd une vie s'il se fait toucher deux fois par ses ennemis. Après le premier coup, Arthur perd son armure et se retrouve en caleçon. Au second, il devient un squelette et meurt, le joueur perd alors une vie. Au début de chaque niveau, Arthur est vêtu d'une armure, même s'il n'en portait pas à la fin du niveau précédent. À certains endroits du jeu, Arthur peut mourir d'un coup qu'il porte une armure ou non. Lorsque le joueur perd une vie, il reprend le jeu au début du niveau ou au *checkpoint* du milieu de niveau. De plus, chaque vie ne dure qu'un certain temps, généralement trois minutes ; un décompte apparaît à l'écran et le joueur perd une vie lorsqu'il touche à sa fin. Il est relancé à chaque début de niveau.

- **Armes**
  Arthur ne peut posséder qu'une seule arme à la fois. Toutes les armes de jet peuvent se lancer indéfiniment. Arthur a la possibilité d'avoir les armes suivantes :

  - **Lance** : le joueur débute avec cette arme.
  - **Dague** : une arme puissante, plus rapide que la lance.
  - **Torche enflammée** : Elle forme un arc de feu lorsqu'elle est utilisée et brûle momentanément le sol, détruisant tout ennemis entrant en contact avec. Elle est plus efficace que la lance et la dague, mais est plus difficile à employer.
  - **Hache** : elle forme un arc de cercle tout comme la torche, mais elle continue à travers les ennemis. Cela permet de provoquer des dégâts à des ennemis multiples.
  - **Bouclier** (ou Crucifix selon la version) : semblable à la lance, mais part moins loin. Cependant, contrairement aux autres armes, elle peut aussi bloquer les attaques des ennemis. C'est la seule arme qui peut battre le boss final.

- **Personnages**
  Le personnage principal, Arthur, apparaît dans le jeu *Marvel vs. Capcom : Clash of Super Heroes*. Il apparait également en tant que combattant dans *Marvel vs. Capcom 3 : Fate of Two Worlds*.
  Firebrand devint ensuite le héros d'une nouvelle série du nom de *Gargoyle's Quest* et *Demon's Crest*. Il est aussi jouable dans *SNK vs. Capcom : SVC Chaos*.
  Arthur, Astaroth, ainsi que d'autres ennemis du jeu apparaissent dans le jeu vidéo *Namco x Capcom*. Certains lieux sont fortement inspirés des niveaux de *Ghosts'n Goblins*.

- **Musique**
  La musique du premier niveau peut être jouée dans le niveau *Shade Man* de *Megaman 7* sur Super Nintendo à la place de la musique originale. Pour cela, il faut appuyer sur le bouton `B` en même temps que l'on sélectionne le niveau.

- **Équipe de développement**
  - **Concepteur de jeux** : Tokuro Fujiwara
  - **Programmeur en chef** : Toshio Arima
  - **Musique et effets sonores** : Ayako Mori

- **Accueil**
  Ce jeu est classé *"88ème meilleur jeu de tous les temps"* selon le site français `jeuxvideo.com`.

- **Exploitation**
  Avec le succès du jeu sur borne d'arcade en 1985, de nombreuses adaptations ont été réalisées sur console de jeux vidéo. Le jeu a plus tard été réédité sur des plateformes de générations suivantes.

- **Portages**
  - La version Commodore 64 est sorti en 1987. Programmée par Chris Butler,

elle est aussi célèbre pour sa bande originale réalisée par Mark Cooksey. Étant donné le peu de ressources du Commodore 64, elle est un peu différente de la version arcade.

- Une version Commodore Amiga est sorti en 1990. Bien que la technologie avancée de l'Amiga permettait à l'époque des conversions fidèles des jeux d'arcade, ce portage pourtant tardif (sorti en fait quelques mois après l'adaptation de *Ghouls'n Ghosts*) ne vaut pas l'original. Dans cette version, le joueur commence avec six vies.
- *Ghosts'n Goblins* fut aussi porté sur les ordinateurs personnels Atari ST, Amstrad CPC, ZX Spectrum, DOS, FM-7 (1987, ASCII), Sharp X68000 et les consoles Game Boy Color (1999, Digital Eclipse) et la NES et WonderSwan.

- **Rééditions**
  - La version originale fut incluse dans la compilation *Capcom Generations Vol.2 : Chronicles of Arthur* sur PlayStation (au Japon et en Europe) et sur Saturn (au Japon uniquement), puis dans la compilation *Capcom Classics Collection*.
  - La version NES a été réédité en 2004 sur Game Boy Advance dans la gamme NES Classics. Il fut aussi rendu disponible sur des petites consoles Sega Genesis à jeux fermés. Il était inclus avec 1942 et 1943 : The Battle of Midway dans une mini console Play TV et sa suite, *Ghouls'n Ghosts* est disponible avec *Street Fighter II : Champion Edition* sur la console Sega Play TV.

FIGURE I.1 – La couverture du jeu

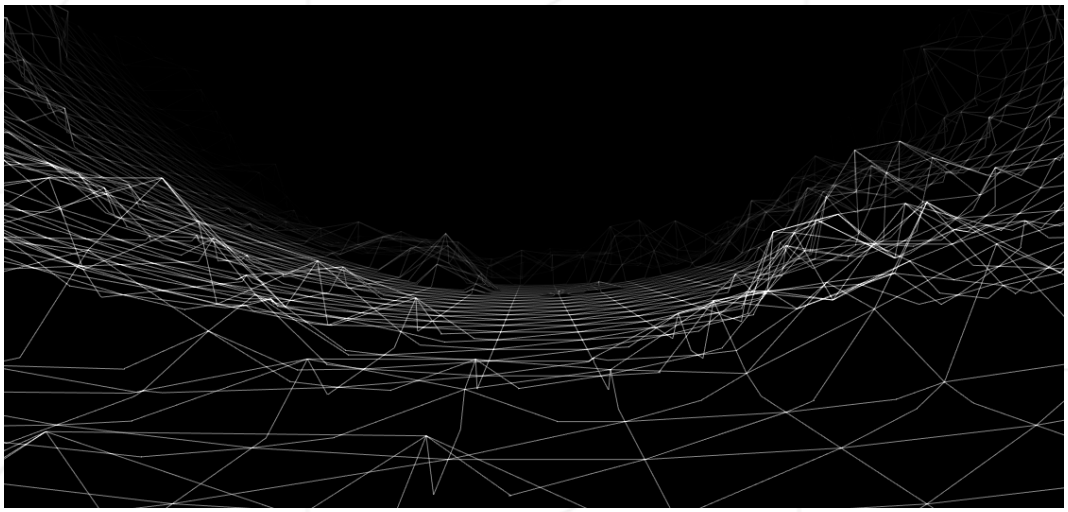
Chapitre II

Introduction

La représentation en relief d'un terrain est une pratique clef de la cartographie moderne. Par exemple, en cette ère d'exploration spatiale, avoir une reproduction en trois dimensions de la surface de Mars est un prérequis indispensable à la conquête de cette planète.

Autre exemple, comparer des représentations en trois dimensions d'une zone où l'activité tectonique est importante permet de mieux comprendre ces phénomènes et leur évolution, donc d'être mieux préparé.

À vous aujourd'hui de vous essayer à cette pratique et de modéliser de magnifiques terrains en trois dimensions, imaginaires ou non.



Chapitre III

Objectifs

Il est temps pour vous d'attaquer votre premier projet graphique !

Vous allez maintenant prendre en main la bibliothèque graphique de l'école : la `MiniLibX` ! Cette bibliothèque a été développée en interne et inclut des outils basiques permettant d'ouvrir une fenêtre, de créer des images et de gérer des événements clavier et souris.

Ce sera l'occasion de vous familiariser avec la `MiniLibX`, de découvrir les bases de la **programmation graphique**, en particulier placer des points dans l'espace, les relier et, surtout, comment observer la scène d'un certain point de vue.

Chapitre IV

Règles communes

- Votre projet doit être écrit en C.
- Votre projet doit être codé à la Norme. Si vous avez des fichiers ou fonctions bonus, celles-ci seront inclues dans la vérification de la norme et vous aurez 0 au projet en cas de faute de norme.
- Vos fonctions ne doivent pas s'arrêter de manière inattendue (segmentation fault, bus error, double free, etc) mis à part dans le cas d'un comportement indéfini. Si cela arrive, votre projet sera considéré non fonctionnel et vous aurez 0 au projet.
- Toute mémoire allouée sur la heap doit être libéré lorsque c'est nécessaire. Aucun leak ne sera toléré.
- Si le projet le demande, vous devez rendre un Makefile qui compilera vos sources pour créer la sortie demandée, en utilisant les flags `-Wall`, `-Wextra` et `-Werror`. Votre Makefile ne doit pas relink.
- Si le projet demande un Makefile, votre Makefile doit au minimum contenir les règles `$(NAME)`, `all`, `clean`, `fclean` et `re`.
- Pour rendre des bonus, vous devez inclure une règle `bonus` à votre Makefile qui ajoutera les divers headers, librairies ou fonctions qui ne sont pas autorisées dans la partie principale du projet. Les bonus doivent être dans un fichier différent : `_bonus.{c/h}`. L'évaluation de la partie obligatoire et de la partie bonus sont faites séparément.
- Si le projet autorise votre `libft`, vous devez copier ses sources et son Makefile associé dans un dossier libft contenu à la racine. Le Makefile de votre projet doit compiler la librairie à l'aide de son Makefile, puis compiler le projet.
- Nous vous recommandons de créer des programmes de test pour votre projet, bien que ce travail **ne sera pas rendu ni noté**. Cela vous donnera une chance de tester facilement votre travail ainsi que celui de vos pairs.
- Vous devez rendre votre travail sur le git qui vous est assigné. Seul le travail déposé sur git sera évalué. Si Deepthought doit corriger votre travail, cela sera fait à la fin des peer-evaluations. Si une erreur se produit pendant l'évaluation Deepthought, celle-ci s'arrête.

Chapitre V

Partie obligatoire

Nom du programme	`fdf`
Fichiers de rendu	`Makefile, *.h, *.c`
Makefile	`NAME, all, clean, fclean, re`
Arguments	`Un fichier au format *.fdf`
Fonctions externes autorisées	• `open, close, read, write, malloc, free, perror, strerror, exit` • `Toutes les fonctions de la bibliothèque mathématique (option de compilation -lm, man man 3 math)` • `Toutes les fonctions de la MiniLibX` • `ft_printf et tout équivalent que VOUS avez codé`
Libft autorisée	`Oui`
Description	`Ce projet consiste à créer le rendu fil de fer d'un paysage.`

Ce projet consiste à créer graphiquement la représentation schématique d'un terrain en relief en reliant différents points `(x, y, z)` par des segments.

Votre projet doit respecter les règles suivantes :

- Vous **devez** utiliser la `MiniLibX`. Soit la version disponible sur les machines de l'école, soit en l'installant par les sources.
- Vous devez rendre un `Makefile` qui compilera vos fichiers sources. Il ne doit pas `relink`.
- Les variables globales sont interdites.

V.1 Rendu

Le rendu doit être affiché en utilisant une **projection isométrique**.

Les coordonnées du terrain seront stockées dans un fichier passé en paramètre dont voici un exemple :

```
$>cat 42.fdf
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0 10 10  0  0 10 10  0  0  0 10 10 10 10 10  0  0  0
0  0 10 10  0  0 10 10  0  0  0  0  0  0  0 10 10  0  0
0  0 10 10  0  0 10 10  0  0  0  0  0  0  0 10 10  0  0
0  0 10 10 10 10 10 10  0  0  0  0 10 10 10 10  0  0  0
0  0  0 10 10 10 10 10  0  0  0 10 10  0  0  0  0  0  0
0  0  0  0  0  0 10 10  0  0  0 10 10  0  0  0  0  0  0
0  0  0  0  0  0 10 10  0  0  0 10 10 10 10 10 10  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
$>
```

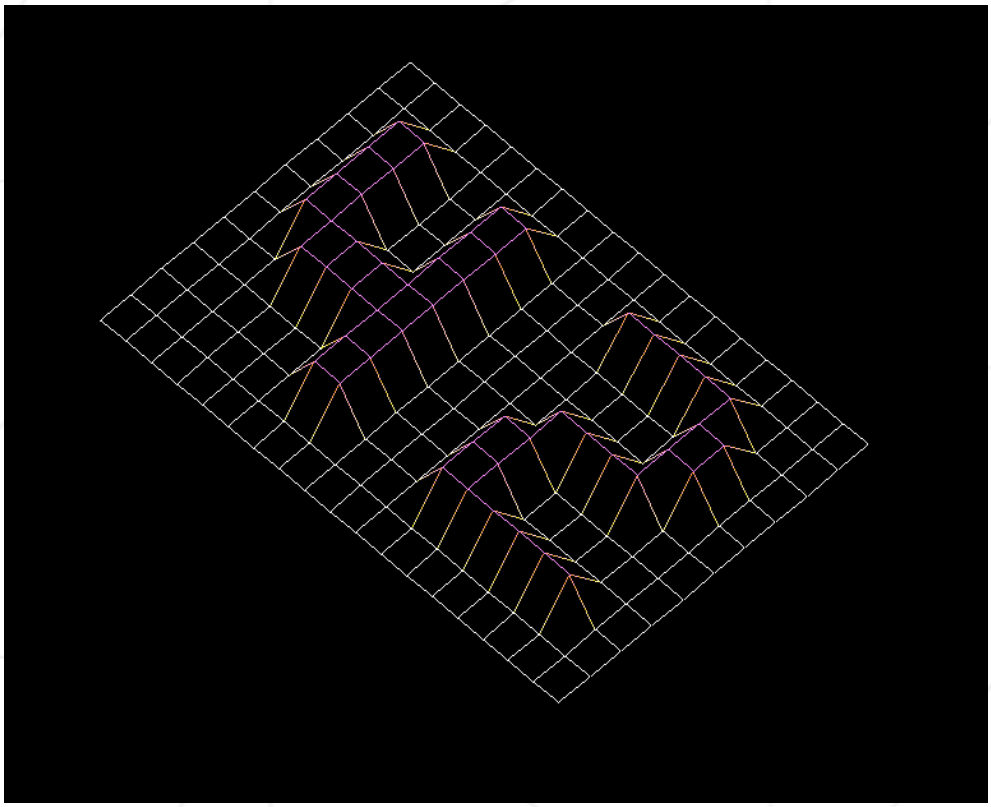
Chaque nombre représente un point dans l'espace :

- La position horizontale correspond à son **abscisse**.
- La position verticale correspond à son **ordonnée**.
- La valeur correspond à son **altitude**.

Exécuter votre programme `fdf` avec le fichier `42.fdf` ci-dessus :

```
$>./fdf 42.fdf
$>
```

Devra produire un rendu similaire à :



Pensez à tirer parti de votre `libft` ! L'utilisation de `get_next_line()`, `ft_split()` et d'autres fonctions vous permettra de lire rapidement et facilement les données du fichier.

Rappelez-vous que le but de ce projet n'est pas de parser des cartes ! Toutefois, cela ne signifie pas que votre programme peut crasher, mais qu'on part du principe qu'une carte sera correctement formatée dans le fichier passé en paramètre.

V.2 Gestion graphique

- Votre programme doit afficher une image dans une fenêtre.
- La gestion de la fenêtre doit rester fluide (changer de fenêtre, la réduire, etc.).
- Appuyer sur la touche `ESC` doit fermer la fenêtre et quitter le programme proprement.
- Cliquer sur la croix en haut de la fenêtre doit fermer celle-ci et quitter le programme proprement.
- Utiliser les `images` de la `MiniLibX` est obligatoire.

`Vous trouverez sur la page intranet du projet un binaire fdf et le fichier d'exemple 42.fdf dans un fdf.zip.`

Chapitre VI

Partie bonus

Habituellement, vous seriez encouragé(e) à développer vos propres bonus. Cependant, d'autres projets graphiques plus intéressants sont à venir. Ils vous attendent déjà ! Ne perdez pas de temps !

Vous aurez des points supplémentaires si vous :

- Incluez une projection supplémentaire (ex : parallèle ou conique) !
- Implémentez le zoom avant et arrière.
- Implémentez la translation (déplacement).
- Implémentez la rotation (faire pivoter votre rendu).
- Ajoutez un bonus supplémentaire de votre choix.

Les bonus ne seront évalués que si la partie obligatoire est PARFAITE. Par parfaite, nous entendons complète et sans aucun dysfonctionnement. Si vous n'avez pas réussi TOUS les points de la partie obligatoire, votre partie bonus ne sera pas prise en compte.

Chapitre VII

Rendu et peer-evaluation

Rendez votre travail sur votre dépot `Git` comme d'habitude. Seul le travail présent sur votre dépot sera évalué en soutenance. Vérifiez bien les noms de vos dossiers et de vos fichiers afin que ces derniers soient conformes aux demandes du sujet.

Vu que votre travail ne sera pas évalué par un programme, organisez vos fichiers comme bon vous semble du moment que vous rendez les fichiers obligatoires et respectez les consignes du sujet.

file.bfe:VADYjxBiOQSAWNqB652klCj13URaziELdHd+2Z38
XCMD9dvO9tSyFob6Il3NBX9YXrgZEiQK7JZJ7w5t0N80wMl7