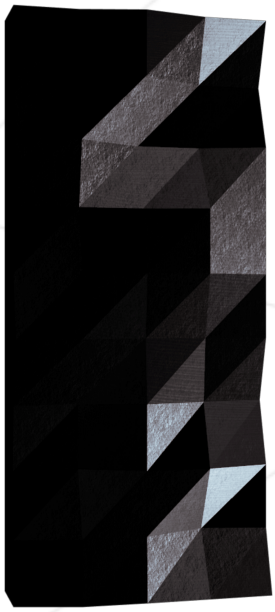


FDF

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

Version: 2

Table des matières

Chapitre I

Préambule

Voici ce que Wikipedia a à dire sur `Ghosts'n Goblins` :

`Ghosts'n Goblins` (Makaimura, Demon World Village) au Japon est un jeu vidéo de plates-formes 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 plates-formes. `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 plates-formes 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 plates-formes 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.

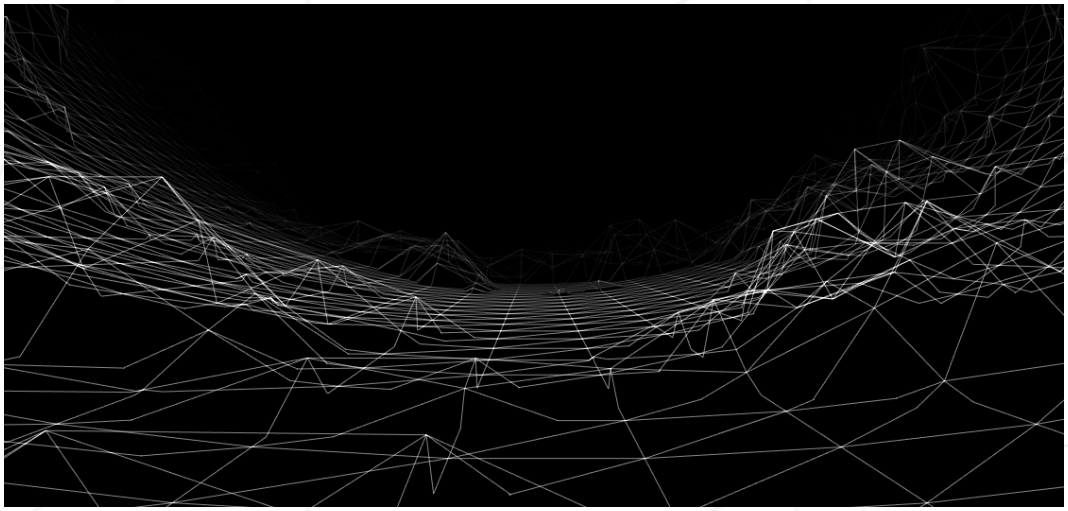
Figure I.1 – Game's cover

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 ou l'activité tectonique est importante permet de mieux comprendre ces phénomènes et leur évolution, permettant ainsi d'être mieux préparés.

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



Chapitre III

Objectifs

Vous decouvrirez dans ce projet les bases de la programmation graphique, et en particulier, le placement de points dans l'espace, comment les relier avec des segments et surtout comment observer la scène depuis un certain point de vue.

Vous découvrirez également votre première bibliothèque graphique : La `miniLibX`. Cette bibliothèque développée en interne rassemble le minimum nécéssaire pour ouvrir une fenêtre, allumer un pixel et gérer les évènements lié à cette fenêtre : le clavier et la souris. Ce sera pour vous l'occasion de vous initier à la programmation dite "evenementielle".

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 de 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.

- 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 doit s'appeller `fdf`.
- Vous devez rendre un `Makefile`.
- Vous ne devez pas utiliser de variables globales.
- Vous devez `obligatoirement` utiliser la `miniLibX`. Soit dans sa version présente sur les dumps, soit à partir de ses sources. Si vous choisissez de travailler à partir de ses sources, vous devez appliquer les mêmes règles que pour votre `libft` telles que décrites au dessus.
- Dans le cadre de votre partie obligatoire, vous avez le droit d'utiliser les fonctions suivantes :
  - `open, read, write, close`
  - `malloc, free`
  - `perror, strerror`
  - `exit`
  - Toutes les fonctions de la lib `math` (-lm et man 3 math)
  - Toutes les fonctions de la `miniLibX`.
- Vous avez l'autorisation d'utiliser d'autres fonctions dans le cadre de vos bonus, à condition que leur utilisation soit dûment justifiée lors de votre évaluation. Soyez malins.
- Vous pouvez poser vos questions sur le forum, Slack, etc.

Chapitre V

Partie obligatoire

Ce projet consiste à créer graphiquement la representation schématique (en “fils de fer” ou “wireframe” en anglais) d’un terrain en relief en reliant différents points `(x, y, z)` par des segments. Les coordonnées du terrain seront stockées dans un fichier passé en paramètre, dont voici un exemple :

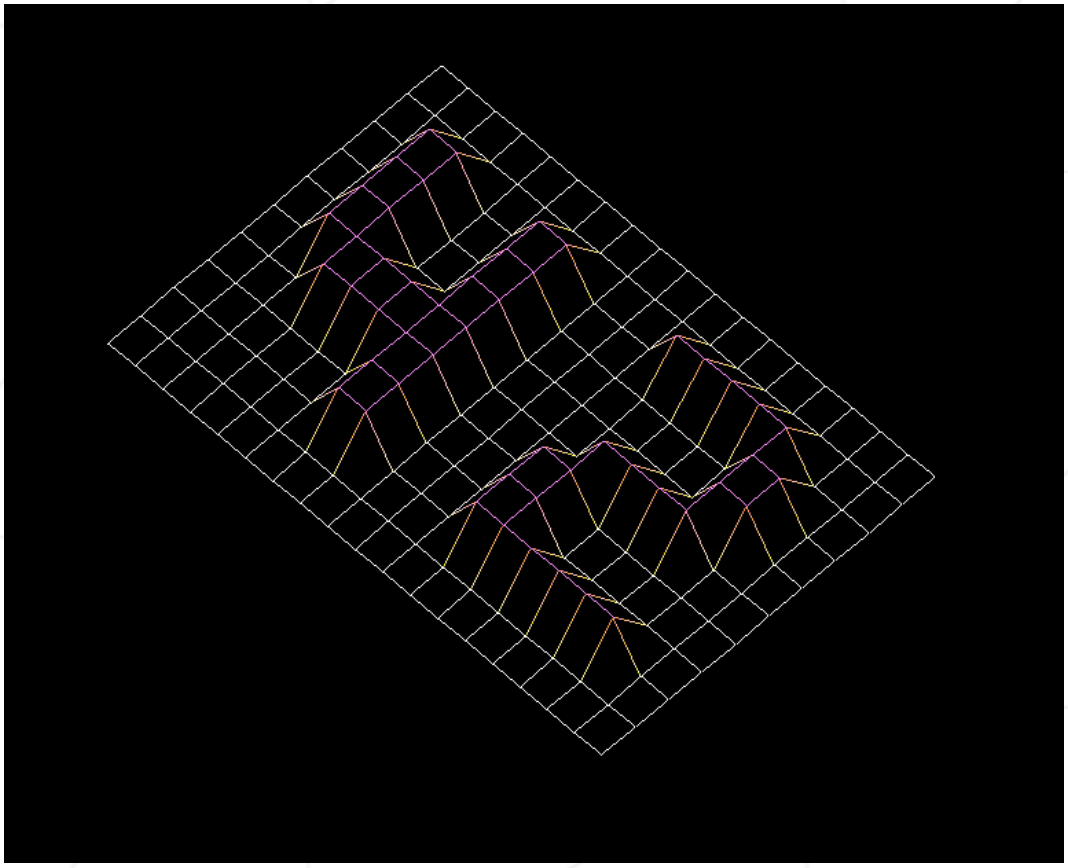
```
$>cat 42.fdf
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0 10 10  0  0 10 10  0  0  0 10 10 10 10 10  0  0  0
0  0 10 10  0  0 10 10  0  0  0  0  0  0  0 10 10  0  0
0  0 10 10  0  0 10 10  0  0  0  0  0  0  0 10 10  0 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

Si on éxécute votre programme `fdf` sur ce fichier, on devra voir quelque chose similaire à :

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



Pensez à exploiter votre `libft` ! L'utilisation de `get_next_line`, `ft_split` et `ft_getnbr` vous permettra de faire une lecture rapide et simple des données du fichier.

Rappel toi que le but de ce projet n'est pas d'analyser les cartes ! Ca ne veut pas dire que ton programme peut segfault, cela signifie que la carte contenue dans le fichier sera correctement formatée.

En ce qui concerne la représentation graphique :

- Votre fdf doit afficher la carte en utilisant une projection isométrique.
- Vous devez pouvoir quitter le programme en appuyant sur `'esc'`.
- L'utilisation de `images` de `minilibX` n'est pas nécessaire pour valider le projet, même si nous vous encourageons fortement à les utiliser.
- Vous trouverez dans les fichiers associés au sujet sur l'intranet un binaire de test (`fdf` dans `fdf.zip`) et le fichier d'exemple `42.fdf`

`man mlx`

Chapitre VI

Partie bonus

Habituellement, nous vous encourageons à ajouter vos propres bonus originaux, mais il y a d'autres projets graphiques bien plus intéressants en attente !! Ne perdez pas trop de temps ici, avancez vite !

Vous gagnerez des points supplémentaires si vous parvenez à :

- Incluez une projection supplémentaire (ex : parallèle ou conique) !
- Cela pourrait être très agréable de pouvoir zoomer, non ?
- Qu'en est-il de la rotation de ta carte ?
- Le dernier détail sympa que vous pouvez ajouter sont, bien sûr, les COULEURS !!

Chapitre VII

Rendu et peer-évaluation

Rendez-votre travail sur votre dépot `GiT` comme d'habitude. Seul le travail présent sur votre dépot sera évalué en soutenance.

Bonne Chance ! :)