FDF

Wireframe model

Summary: This project involves representing a wireframe landscape in 3D.

Version: 5.0

Contents

I Foreword 2

II Introduction 5

III Objectives 6

IV Common Instructions 7

V AI Instructions 9

VI Mandatory part 11

VI.1 Rendering 13

VI.2 Graphic management 15

VII Readme Requirements 16

VIII Bonus part 17

IX Submission and peer-evaluation 18

Chapter I

Foreword

This is what *Wikipedia* says about *Ghosts'n Goblins*:

Ghosts'n Goblins is a platform game where the player controls a knight, named Sir Arthur, who must defeat zombies, ogres, demons, cyclops, dragons and other monsters in order to rescue Princess Prin Prin, who has been kidnapped by Satan, king of Demon World. Along the way the player can pick up new weapons, bonuses and extra suits of armor that can help in this task.

The game is often considered very difficult by arcade standards and is commonly regarded as **one of the most difficult games ever released**.

The player can only be hit twice before losing a life (the first hit takes away Arthur's armor, and the player must continue on wearing underwear until completing the level or finding replacement armor). If the player loses a life, they start all over again, or from the halfway point if they managed to get that far.

Furthermore, each life can only last a fixed duration (generally around three minutes), the clock being reset at the start of a level. If the clock does run out, the player instantly loses that life.

After defeating the final boss, but only with the cross weapon (if the player does not have the cross weapon, they will be prompted that it is needed to defeat the boss and restart at the beginning of level 5 and must repeat round 5 and 6 again regardless if the weapon is obtained immediately or not) for the first time the player is informed that the battle was a trap devised by Satan. The player must then replay the entire game on a higher difficulty level to reach the genuine final battle.

- **Ports**

  Many conversions to home computers were produced by `Elite Systems`.

  - **The Commodore 64 version**, released in 1986, is known for its music by Mark Cooksey, which borrows from Frédéric Chopin's *Prelude No. 20*. Due to the limited resources on the Commodore 64, it was somewhat different from the arcade version. It only features the Graveyard and Forest, The Ice Palace, The Floating Platforms and Firebridge and The Caves in that order. The player also starts the game with five lives. The demon that kidnapped the princess replaces Astaroth in the title screen. Additionally, the `cyclops` (or *Unicorn*) is the boss of levels one to three and the dragon is the final boss.
  - **The version for Commodore 16/116 and Commodore Plus/4**, also released in 1986 by `Elite Systems`, was even more limited than the C64 version. It was written to work on a Commodore 16, which had only 16 KB of RAM. Therefore, this version features only two levels and no music. In addition, the remaining two levels and the gameplay are simplified. For example, in the graveyard level, the attacking bird, the *plant monsters* and the winged demon are all missing from the C16 version, and there is only one weapon. The title screen features no graphics besides the stylised *Ghosts 'n Goblins* lettering.
  - **A version for the Commodore Amiga was released in 1990**. While the advanced hardware of the Amiga allowed an almost perfect conversion of the arcade game, it failed to emulate the success of the Commodore 64 version. The player starts the game with six lives and no music is played unless the Amiga was equipped with at least 1 Megabyte of RAM. The standard configuration of an Amiga 500 was 512 Kilobytes.
  - **The NES version** was developed by `Micronics`. This also serves as the basis for the Game Boy Color version, which uses passwords to allow the player to jump to certain levels. The NES version was ported to the Game Boy Advance as part of *Classic NES series*, but only in Japan.
  - **The NES version** was also re-released for download for Nintendo's Virtual Console in North America on December 10, 2007 (Wii) and October 25, 2012 (Nintendo 3DS) and in the PAL region on October 31, 2008 (Wii) and January 3, 2013 (Nintendo 3DS) while the Wii U version was released in both regions on May 30, 2013. The arcade version was released on the Wii's Virtual Console Arcade in Japan on November 16, 2010, the PAL region on January 7, 2011 and in North America on January 10, 2011.
  - `Ghosts'n Goblins` was also ported to the ZX Spectrum, Amstrad CPC, MSX, Atari ST, IBM PC compatibles, Game Boy Color, Game Boy Advance. The original arcade version of the game was also included in the compilation *Capcom Generations Vol.2: Chronicles of Arthur* for the PlayStation (in Japan and Europe) and Sega Saturn (in Japan only), which also contained *Ghouls'n Ghosts* and *Super Ghouls'n Ghosts*. The three games (based on their `Capcom Generation` versions) were later collected as part of *Capcom Classics Collection*. The game was also featured in the compilation *Capcom Arcade Cabinet* for the PlayStation 3 and Xbox 360.

- **Reception**

  Computer Gaming World called *Ghosts'n Goblins "an excellent example of what the [NES] can do ... while hardly groundbreaking, represents the kind of game that made Nintendo famous"*.
  Ghosts 'n Goblins was runner-up in the category of **Arcade-Style Game of the Year** at the Golden Joystick Awards. The NES version of *Ghosts 'n Goblins* was rated the 129th best game made on a Nintendo System in Nintendo Power's Top 200 Games list. It was also a best seller for the NES, selling 1.64 million units.
  Ghosts 'n Goblins is often cited as an example of one of the **most difficult games of all time** to beat, due to its extreme level of difficulty and the fact the player must play through the game twice in order to beat the game.

- **Legacy**

  Ghosts'n Goblins was followed by a series of sequels and spin-offs eventually becoming **Capcom's 8th best-selling game franchise**, selling over 4.4 million units. Its sequels include *Ghouls'n Ghosts*, *Super Ghouls'n Ghosts*, and *Ultimate Ghosts'n Goblins* in addition to producing the *Gargoyle's Quest* and Maximo spin-off series. Though originating as an arcade title, the franchise has been featured on a variety of PC and video game consoles with the latest entries in the series, *Ghosts 'n Goblins: Gold Knights*, released on the iOS. Additionally, the franchise frequently makes cameo appearances — the character of Arthur in particular — in other Capcom titles, the latest of which being *Ultimate Marvel vs. Capcom 3*.

Figure I.1: The game's cover

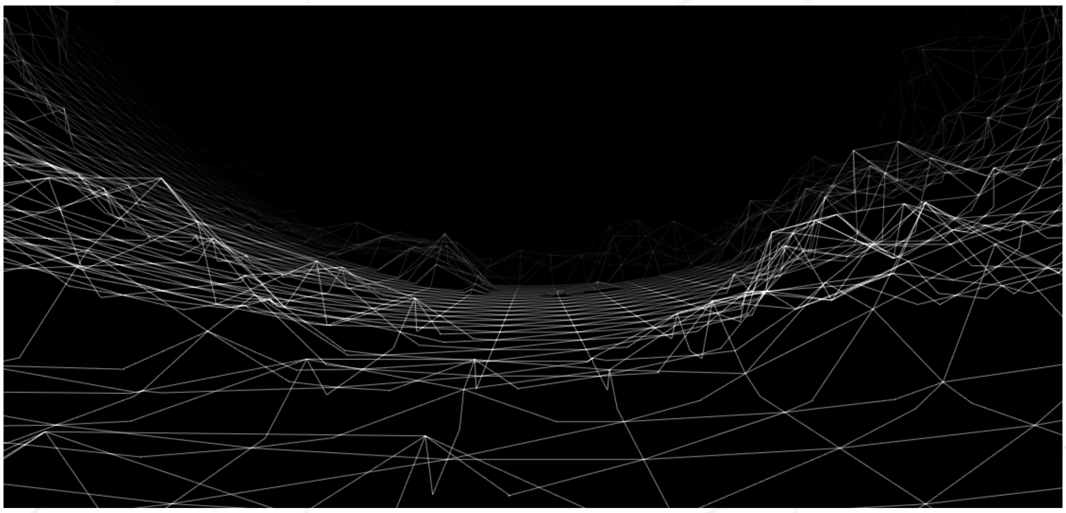
Chapter II

Introduction

The representation in 3D of a landscape is a critical aspect of modern mapping. For example, in these times of space exploration, having a 3D representation of Mars is a prerequisite for its conquest.

As another example, comparing various 3D representations of an area of high tectonic activity will allow you to better understand these phenomena and their evolution, and as a result, be better prepared.

Today, it is your turn to model magnificent 3D landscapes, whether imaginary or real.



FDF is short for 'fil de fer' in French which means 'wireframe model'.

Chapter III

Objectives

It's time for you to create a basic computer graphics project!

You will use the school's graphical library: the `MiniLibX`! This library was developed internally and includes basic necessary tools to open a window, create images and deal with keyboard and mouse events.

This will be an opportunity for you to become familiar with the `MiniLibX` library, discover the basics of **graphics programming**, particularly how to place points in space, connect them, and most importantly, how to view the scene from a specific viewpoint.

Chapter IV

Common Instructions

- Your project must be written in C.
- Your project must be written in accordance with the Norm. If you have bonus files/functions, they are included in the norm check, and you will receive a `0` if there is a norm error.
- Your functions should not quit unexpectedly (segmentation fault, bus error, double free, etc.) except for undefined behavior. If this occurs, your project will be considered non-functional and will receive a `0` during the evaluation.
- All heap-allocated memory must be properly freed when necessary. Memory leaks will not be tolerated.
- If the subject requires it, you must submit a `Makefile` that compiles your source files to the required output with the flags `-Wall`, `-Wextra`, and `-Werror`, using `cc`. Additionally, your `Makefile` must not perform unnecessary relinking.
- Your `Makefile` must contain at least the rules `$(NAME)`, `all`, `clean`, `fclean` and `re`.
- To submit bonuses for your project, you must include a `bonus` rule in your `Makefile`, which will add all the various headers, libraries, or functions that are not allowed in the main part of the project. Bonuses must be placed in `_bonus.{c/h}` files, unless the subject specifies otherwise. The evaluation of mandatory and bonus parts is conducted separately.
- If your project allows you to use your `libft`, you must copy its sources and its associated `Makefile` into a `libft` folder. Your project's `Makefile` must compile the library by using its `Makefile`, then compile the project.
- We encourage you to create test programs for your project, even though this work **does not need to be submitted and will not be graded**. It will give you an opportunity to easily test your work and your peers' work. You will find these tests especially useful during your defence. Indeed, during defence, you are free to use your tests and/or the tests of the peer you are evaluating.
- Submit your work to the assigned Git repository. Only the work in the Git repository will be graded. If Deepthought is assigned to grade your work, it will occur

after your peer-evaluations. If an error happens in any section of your work during Deepthought's grading, the evaluation will stop.

Chapter V

AI Instructions

● Context

During your learning journey, AI can assist with many different tasks. Take the time to explore the various capabilities of AI tools and how they can support your work. However, always approach them with caution and critically assess the results. Whether it's code, documentation, ideas, or technical explanations, you can never be completely sure that your question was well-formed or that the generated content is accurate. Your peers are a valuable resource to help you avoid mistakes and blind spots.

● Main message

- ☛ Use AI to reduce repetitive or tedious tasks.
- ☛ Develop prompting skills — both coding and non-coding — that will benefit your future career.
- ☛ Learn how AI systems work to better anticipate and avoid common risks, biases, and ethical issues.
- ☛ Continue building both technical and power skills by working with your peers.
- ☛ Only use AI-generated content that you fully understand and can take responsibility for.

● Learner rules:

- You should take the time to explore AI tools and understand how they work, so you can use them ethically and reduce potential biases.
- You should reflect on your problem before prompting — this helps you write clearer, more detailed, and more relevant prompts using accurate vocabulary.
- You should develop the habit of systematically checking, reviewing, questioning, and testing anything generated by AI.
- You should always seek peer review — don't rely solely on your own validation.

● Phase outcomes:

- Develop both general-purpose and domain-specific prompting skills.
- Boost your productivity with effective use of AI tools.
- Continue strengthening computational thinking, problem-solving, adaptability, and collaboration.

● Comments and examples:

- You'll regularly encounter situations — exams, evaluations, and more — where you must demonstrate real understanding. Be prepared, keep building both your technical and interpersonal skills.
- Explaining your reasoning and debating with peers often reveals gaps in your understanding. Make peer learning a priority.
- AI tools often lack your specific context and tend to provide generic responses. Your peers, who share your environment, can offer more relevant and accurate insights.
- Where AI tends to generate the most likely answer, your peers can provide alternative perspectives and valuable nuance. Rely on them as a quality checkpoint.

✓ Good practice:

I ask AI: "How do I test a sorting function?" It gives me a few ideas. I try them out and review the results with a peer. We refine the approach together.

✗ Bad practice:

I ask AI to write a whole function, copy-paste it into my project. During peer-evaluation, I can't explain what it does or why. I lose credibility — and I fail my project.

✓ Good practice:

I use AI to help design a parser. Then I walk through the logic with a peer. We catch two bugs and rewrite it together — better, cleaner, and fully understood.

✗ Bad practice:

I let Copilot generate my code for a key part of my project. It compiles, but I can't explain how it handles pipes. During the evaluation, I fail to justify and I fail my project.

Chapter VI

Mandatory part

Program Name	`fdf`
Files to Submit	`Makefile, *.h, *.c, demo .fdf files`
Makefile	`NAME, all, clean, fclean, re`
Arguments	`A file in format *.fdf`
External Function	• `open, close, read, write, malloc, free, perror, strerror, exit.` • `All functions of the math library (-lm compiler option,` *`man 3 math`* `on some operating systems).` • `All functions of the MiniLibX library.` • `gettimeofday()` • `ft_printf or any equivalent YOU coded.`
Libft authorized	`Yes`
Description	`This project involves creating a simple wireframe model of a landscape.`

This project is about creating a simple wireframe model representation of a 3D landscape by linking various points `(x, y, z)` thanks to line segments (edges).

Your project must comply with the following rules:

- You **must** use the `MiniLibX` library, either the version available on school machines, or by installing it using its sources.
- You have to turn in a `Makefile` which will compile your source files. It must not relink.

- Global variables are forbidden.

VI.1 Rendering

Your program has to represent the model in **isometric projection**.

The coordinates of the landscape are stored in a `.fdf` file, provided as a command line parameter to your program. Here is an example:

```
$>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 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
$>
```

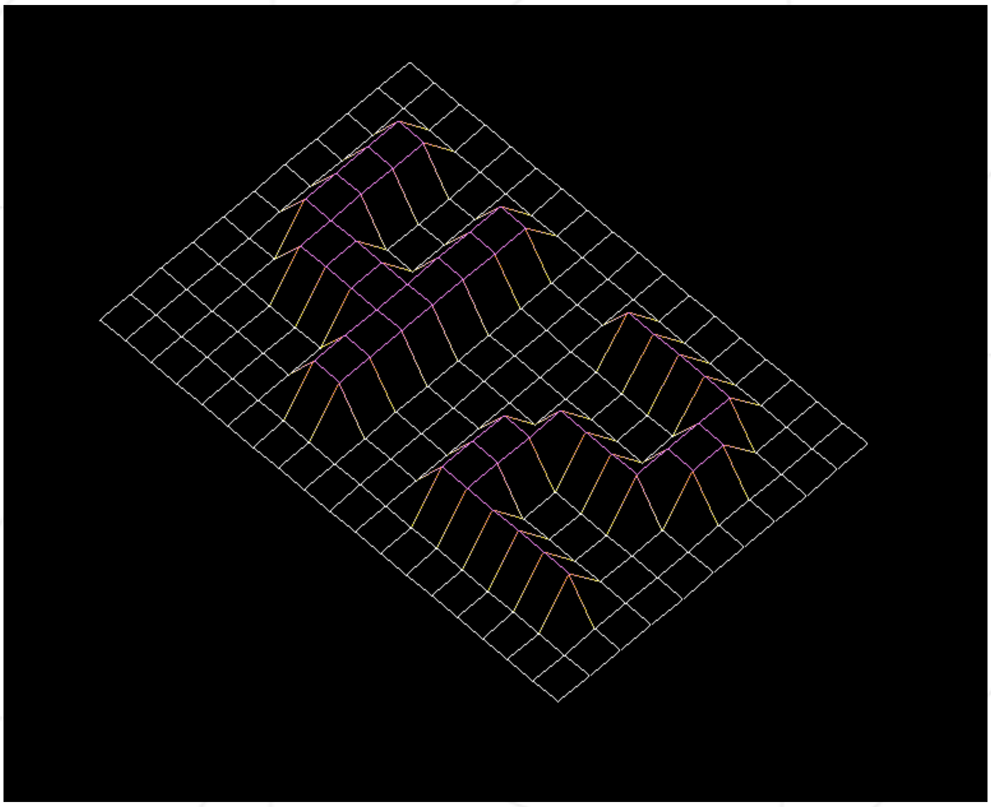
Each number represents a point in space:

- The horizontal position corresponds to its **abscissa**.
- The vertical position corresponds to its **ordinate**.
- The value corresponds to its **altitude**.

Executing your `fdf` program using the example file `42.fdf`:

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

Should render a landscape similar to:



Remember to use your `libft` in the best way possible! Using `get_next_line()`, `ft_split()` and other functions will allow you to read data from the file in a quick and simple way.

Keep in mind that the goal of this project is not to parse maps! However, this doesn't mean that your program should crash when run. It means that we assume the map contained in the file is properly formatted.

VI.2 Graphic management

- Your program has to display the image in a window.
- Window management must remain smooth (e.g., switching to another window, minimizing, etc.).
- Pressing `ESC` must close the window and quit the program in a clean way.
- Clicking on the cross on the window's frame must close the window and quit the program in a clean way.
- The use of the `images` of the `MiniLibX` library is mandatory.

`On the intranet project page, a binary file called fdf as well as the example file 42.fdf inside a fdf.zip are available to download.`

Chapter VII

Readme Requirements

A `README.md` file must be provided at the root of your Git repository. Its purpose is to allow anyone unfamiliar with the project (peers, staff, recruiters, etc.) to quickly understand what the project is about, how to run it, and where to find more information on the topic.
The `README.md` must include at least:

- The very first line must be italicized and read: *This project has been created as part of the 42 curriculum by <login1>[, <login2>[, <login3>[...]]].*
- A "**Description**" section that clearly presents the project, including its goal and a brief overview.
- An "**Instructions**" section containing any relevant information about compilation, installation, and/or execution.
- A "**Resources**" section listing classic references related to the topic (documentation, articles, tutorials, etc.), as well as a description of how AI was used — specifying for which tasks and which parts of the project.

➡ **Additional sections may be required depending on the project** (e.g., usage examples, feature list, technical choices, etc.).

Any required additions will be explicitly listed below.

`English is recommended; alternatively, you may use the main language of your campus.`

Chapter VIII

Bonus part

Typically, you would be encouraged to develop your own original additional features; however, more interesting graphic projects await you in the future. Don't spend too much time on this assignment!

You are allowed to use other functions to complete the bonus part, as long as their use is **justified** during your evaluation. Be smart!

You will get some extra points if you can:

- Include one extra projection (such as parallel or conic)!
- Zoom in and out.
- Translate your model.
- Rotate your model.
- Add one more bonus of your choice.

⚠ `The bonus part will only be assessed if the mandatory part is PERFECT. Perfect means the mandatory part has been integrally done and works without malfunctioning. If you have not passed ALL the mandatory requirements, your bonus part will not be evaluated at all.`

Chapter IX

Submission and peer-evaluation

Submit your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your files to ensure they are correct.

As these assignments are not verified by a program, feel free to organize your files as you wish, as long as you turn in the mandatory files and comply with the requirements.

During the evaluation, a brief **modification of the project** may occasionally be requested. This could involve a minor behavior change, a few lines of code to write or rewrite, or an easy-to-add feature.

While this step may **not be applicable to every project**, you must be prepared for it if it is mentioned in the evaluation guidelines.

This step is meant to verify your actual understanding of a specific part of the project. The modification can be performed in any development environment you choose (e.g., your usual setup), and it should be feasible within a few minutes — unless a specific timeframe is defined as part of the evaluation.
You can, for example, be asked to make a small update to a function or script, modify a display, or adjust a data structure to store new information, etc.

The details (scope, target, etc.) will be specified in the **evaluation guidelines** and may vary from one evaluation to another for the same project.