cub3D

My first RayCaster with miniLibX

Summary: This project is inspired by the world-famous Wolfenstein 3D game, which was the first FPS ever. It will enable you to explore ray-casting. Your goal will be to make a dynamic view inside a maze, in which you'll have to find your way.

Version: 10

Contents

Chapter I

Foreword

Developed by `Id Software` by the über famous John Carmack and John Romero, published in 1992 by `Apogee Software`, `Wolfenstein 3D` is the first true "First Person Shooter" in the history of video games.

Figure I.1: John Romero (left) and John Carmack (right) posing for posterity.

`Wolfenstein 3D` is the ancestor of games like `Doom` (`Id Software`, 1993), `Doom II` (`Id Software`, 1994), `Duke Nukem 3D` (`3D Realm`, 1996) and `Quake` (`Id Software`, 1996), that are additional eternal milestones in the world of video games.

Now, it's your turn to relive History...

The game Wolfenstein 3D originally takes place in the Nazi Germany, that could be eventually disturbing. Pictures and history of this game are only brought to you for technical reasons and pop/geek culture reasons, as the game was considered as a masterpiece for both.

Chapter II

Goals

This project's objectives are similar to all this first year's objectives: Rigor, use of `C`, use of basic algorithms, information research etc.

As a graphic design project, `cub3D` will enable you to improve your skills in these areas: windows, colors, events, fill shapes, etc.

To conclude `cub3D` is a remarkable playground to explore the playful practical applications of mathematics without having to understand the specifics.

With the help of the numerous documents available on the internet, you will use mathematics as a tool to create elegant and efficient algorithms.

If this suits you, you can test the original game before starting this project:
http://users.atw.hu/wolf3d/

Chapter III

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 inside.

- Your functions should not quit unexpectedly (segmentation fault, bus error, double free, etc) apart from undefined behaviors. If this happens, your project will be considered non functional and will receive a `0` during the evaluation.

- All heap allocated memory space must be properly freed when necessary. No leaks will be tolerated.

- If the subject requires it, you must submit a `Makefile` which will compile your source files to the required output with the flags `-Wall`, `-Wextra` and `-Werror`, use cc, and your Makefile must not relink.

- Your `Makefile` must at least contain the rules `$(NAME)`, `all`, `clean`, `fclean` and `re`.

- To turn in bonuses to your project, you must include a rule `bonus` to your Makefile, which will add all the various headers, libraries or functions that are forbidden on the main part of the project. Bonuses must be in a different file `_bonus.{c/h}` if the subject does not specify anything else. Mandatory and bonus part evaluation is done separately.

- If your project allows you to use your `libft`, you must copy its sources and its associated `Makefile` in a `libft` folder with its associated Makefile. 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 **won't have to be submitted and won't be graded**. It will give you a chance to easily test your work and your peers' work. You will find those 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 your assigned git repository. Only the work in the git repository will be graded. If Deepthought is assigned to grade your work, it will be done