Cub3d

sueno-te

February 2025

1 Introduction

A RayCaster with miniLibX.

[a4paper,12pt]article graphicx hyperref xcolor

cub3D - My First RayCaster with miniLibX 42 Project February 15, 2025

2 Introduction

The **cub3D** project is an introduction to the fundamentals of 3D rendering using the ray-casting technique. Inspired by the mechanics of the classic game *"Wolfenstein 3D"*, this project challenges us to create a simple 3D engine from a first-person perspective using a lightweight graphics library. Such as "42" **miniLibX** and "codam" **Mlx**.

The goal is to generate a realistic visual representation of a maze, respecting specific constraints and handling user interactions.

3 Project Requirements

3.1 Program Information

- **Program Name**: cub3D
- **Turn-in Files**: All necessary source files
- **Makefile Targets**: all, clean, fclean, re, bonus
- **Arguments**: A map file in `.cub` format

3.2 Allowed External Functions

- `open`, `close`, `read`, `write`, `printf`, `malloc`, `free`, `perror`, `strerror`, `exit`
- All functions from the math library (`-lm`)
- All functions from the **miniLibX** or **Mlx**

3.3 Authorized Libraries

- **Libft** (custom C library)

4 Project Description

The program must generate a 3D visualization of a maze, following the ray-casting technique. The key features of the project include:

- Rendering walls with textures based on their facing direction (North, South, East, West).
- Configuring floor and ceiling colors independently.
- Smooth interaction handling, including minimizing and switching between windows.
- Basic movement controls:
  - **Arrow Keys**: Rotate the player's viewpoint.
  - **W, A, S, D**: Move through the maze.
  - **ESC**: Exit the program.
  - Clicking the **window close button** must properly terminate the program.
- The program must read a `.cub` file containing:
  - Map layout (walls, empty spaces, player position and orientation).
  - Textures for walls.
  - Floor and ceiling colors.
- The map must be properly formatted, enclosed by walls, and free of misconfigurations.

5 Example of a Simple Valid Map

```
111111
100101
101001
1100N1
111111
```

6 Error Handling

If any misconfiguration is detected in the input file, the program must exit cleanly and display `"Error\n"` followed by an explicit error message.

7 Bonus Features

If the mandatory part is completed perfectly, additional features can be implemented, including:

- Wall collisions.
- Minimap system.
- Doors that open and close.
- Animated sprites.
- Mouse-based viewpoint rotation.

8 Initial Phase - DDA