What to Learn for the miniRT Project

1. Mathematics

a. Vectors and Their Operations

Why: Used for representing positions, directions, and calculations like lighting and object intersections.

What to Learn:

- Addition, subtraction, scalar multiplication.
- Dot product (for angles and lighting).
- Cross product (for normals and rotations).
- Vector normalization (unit vectors for directions).

b. 3D Geometry

Why: Essential for understanding how rays interact with objects.

What to Learn:

- Equations of planes, spheres, and cylinders.
- How to calculate ray-object intersections.
- Normal vectors and their importance in lighting.

c. Linear Algebra

Why: Used for transformations like rotations and translations.

What to Learn:

- Rotation matrices (for rotating objects in 3D).
- Transformations (translation, scaling).

d. Basic Trigonometry

Why: Needed for field of view (camera angles) and light calculations.

What to Learn:

- Sine, cosine, and tangent functions.
- Converting between degrees and radians.

2. Programming in C

a. Core C Concepts

Why: The entire project must be implemented in C.

What to Learn:

- Data structures (arrays, structs).
- Pointers (especially for managing dynamic memory).

- Functions and modular programming.
- File I/O (for parsing .rt scene files).

b. Memory Management

Why: To prevent memory leaks and ensure clean program exits.
What to Learn:
- malloc and free.
- How to track and manage allocated memory.

c. Debugging

Why: To quickly identify and fix bugs in your code.
What to Learn:
- Using tools like gdb or Valgrind.
- Debugging segmentation faults and logical errors.

d. Makefiles

Why: Required to compile your project efficiently.
What to Learn:
- Writing a Makefile with all, clean, fclean, and re rules.
- Using flags (-Wall, -Wextra, -Werror).

3. Graphics Programming

a. MiniLibX

Why: The project uses MiniLibX to create the window and render images.
What to Learn:
- Initializing a window and managing its lifecycle.
- Drawing pixels, lines, and shapes.
- Handling events (e.g., ESC to exit, window resizing).

b. Image Buffers

Why: To render scenes efficiently.
What to Learn:
- How to manipulate image buffers in MiniLibX.
- Updating the buffer with calculated pixel colors.

4. Ray Tracing Basics

a. What is Ray Tracing?

Why: This is the core of the project.

What to Learn:

- Casting rays from the camera through the screen.
- Checking if rays intersect objects (planes, spheres, cylinders).
- Calculating the color of pixels based on light and material properties.

b. Lighting Models

Why: To make scenes look realistic.

What to Learn:

- Ambient lighting (general illumination).
- Diffuse lighting (light reflecting off surfaces).
- Shadows (occluded areas).

5. File Parsing

Why: The scene data is provided in .rt files.

What to Learn:

- **Reading files line by line (open, read).**
- **Parsing text into meaningful data structures (e.g., spheres, planes).**
- **Validating input and handling errors gracefully.**