



# Winter 2022 Review Campaign

term3d

*Summary: A 3D rendering program... in the terminal!*

# Contents

<b>I</b>	<b>Foreword</b>	<b>2</b>
<b>II</b>	<b>Common Instructions</b>	<b>3</b>
<b>III</b>	<b>Subject</b>	<b>4</b>
<b>IV</b>	<b>Bonus</b>	<b>6</b>
<b>V</b>	<b>Resources</b>	<b>7</b>

# Chapter I

## Foreword

Ever wondered how they make those cool ASCII graphics?  
In this project, you will build a program that renders 3D objects in the terminal!

# Chapter II

## 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.
- Mandatory and bonus part evaluation is done separately.
- You must have completed the mandatory section to receive bonus points.

# Chapter III

## Subject

<b>Program name</b>	term3d
<b>Turn in files</b>	All necessary files, makefile if present, and at least one object file to render
<b>Makefile</b>	Optional
<b>Arguments</b>	Any 3D file format
<b>External functs.</b>	All standard C library functions
<b>Libft authorized</b>	Yes
<b>Description</b>	Your goal for this project is to render a 3D object with ASCII text.

The constraints are as follows:

- Your program should accept a file as an argument, containing a 3D object representation.
- Your object files can be in any format, as long they contain 3D coordinates representing an object. (.ply, .obj, .stl, or your own custom format)
- Given a file as an argument, your program must draw the represented 3D object to the terminal.
- Your program should handle errors in the case of an invalid file or no file at all. Make sure you submit at least one valid file for review.
- Requirements for the render:
  - The object must be 3 dimensional. Not a line, or a plane.
  - The object must be animated to rotate along an axis.
  - The animation should be displayed in-place, each frame being printed over the last.

- A sample file, `torus.3d`, is attached to the subject. The file contains a torus represented as a set of 3D points along its surface.
- Here is a basic flow to get you started with simple perspective projection.
  - Step 1: Load a set of 3D points representing an object's surface (sample file provided in attachments).
  - Step 2: Loop through each of the points, mapping them to a position on the screen via perspective projection.
  - Step 3: Print characters at the projected positions on your terminal.
  - Step 4: Apply a rotation to the points along an axis.
  - Step 5: Loop and repeat steps 2 - 4.
- This is just one of many ways to render an object. If you're ambitious enough to go for bonus points, you can try line-drawing, rasterization, or even ray-tracing!

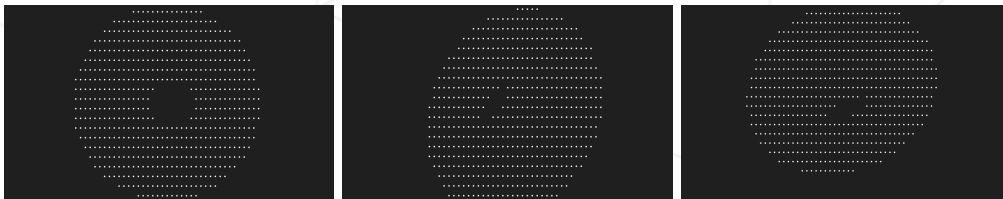


Figure III.1: The pictures above show what the sample file might look like when rendered properly.

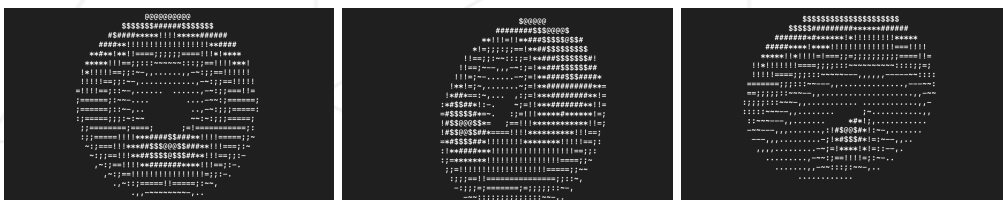


Figure III.2: And here's with one of the bonuses (normal shading).

# Chapter IV

## Bonus

- The bonus section is where the fun begins. You have a lot of creative freedom for bonus points.
- Each of the listed bonuses will provides the amount of points listed below. Maximum bonus is 25 points.
  - Rendering other 3D objects besides the provided sample. (+5)
  - Rendering multiple objects at once. (+5)
  - Handling multiple different file formats. (+5)
  - Full camera control: camera translation and rotation in real-time through user interaction. (+10)
  - Shading: use different characters to show the angles of surfaces. (+15)
  - Line-drawing: use a line-drawing algorithm to draw edges of objects. (+15)
  - Rasterization: render files with only edges or vertices, drawing polygons instead of points. (+25)
  - Raytracing: render files with only edges or vertices, computing rays through each cursor position. (+25)
- Any additional creative bonuses will provide 5 points each. For something to be considered for bonus, it must be transformative. In other words, it must impact the user experience in a meaningful way. The final decision is up to the reviewer, so be ready to defend!
- Examples of things that should NOT be considered for bonus:
  - Changing the color of ASCII characters (unless combined with shading).
  - Using a custom file format.

# Chapter V

## Resources

- Resources for the mandatory part:
  - [Computing the Pixel Coordinates of a 3D Point](#)
  - [The Perspective and Orthographic Projection Matrix](#)
  - [Video Lesson: Perspective Projection](#)
  - [3D Rotation in Computer Graphics](#)
- Resources for bonuses:
  - [Introduction to Shading](#)
  - [Rasterization: a Practical Implementation](#)
  - [An Overview of the Ray-Tracing Rendering Technique](#)
  - [Ray-Tracing: Generating Camera Rays](#)