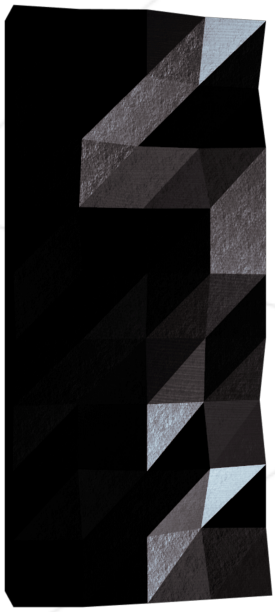

RTv1

My first Ray Tracer

Pedago team pedago@42.fr

Summary: This mini-project is the first step to create a Raytracing program, for you to finally be able to render computer-generated images.

Contents

Chapter I

Foreword

"I think that in Europe, research is pretty good, but the weaknesses come from concrete applications, a step which in itself is a source of innovation. This, I think comes from a lack of companies ready to take risks. What we call development rarely comes from the big corporations, it's the smaller ones or the SMEs that will do that. So, what we all need are loads of SMEs with gifted students, venture capital investments, more efficient in the private sector, and also some role models that we can all take in example, saying "That's in-novation" But there is also another more intricate factor: The cultural factor. In Europe failure is a very big deal. If you leave the UNI and fail, it will follow you your whole life, where as in the US, in Silicon Valley we are constantly failing. When you fall on your face, you get up and start again. What you need for the computer industries to develop in Europe and in France, is a solid computer programming industry, because programming is the oil of the 80s and 90s, of this digital revolution. You need hundreds of mini computer programming companies and France could dominate Europe in software. It has the most brilliant students, a solid understanding of technology, what we need to do now is to encourage the kids to create software companies. We, we can't buy them out, and the government really cannot do that either. They need to belong to those ready to take risks. "

Steve Jobs, 1984, interview given to French TV Antenne 2

Chapter II

Introduction

For once, the topic is related to the foreword. To succeed, you must fail. That's why you will start with some sort of proof-of-concept on the theory of Raytracing. It's the objective of `RTv1`: to get familiar with Ray-Tracing, geometric elements that you will need to manipulate, the description of the scene... You will therefore succeed or fail this project, with the end goal to do it again later bigger, better, more beautiful with all sorts of additional features (it will be the RT, do not start RT if you did not do RTv1, it would not be reasonable.)

Watch the demo video to understand what we have to start with, and what your program is meant to do. Resources on the net are important to gain a better understanding of Raytracing. There are different ways to understand it, find the one that works for you. RTv1 is still a simple version, see what's requested so that you don't get lost in the maze of the numerous functionalities that such a program can contain. Keep the complex features for the RT project.

Chapter III

Objectives

When it comes to render 3 dimensions computer generated images there is 2 possible approaches: “rasterization”, which is used by almost all graphic engines because of it’s efficiency and “ray tracing.” The ray tracing method is more expensive and as a result not adapted to real time but it produces a high degree of visual realism.

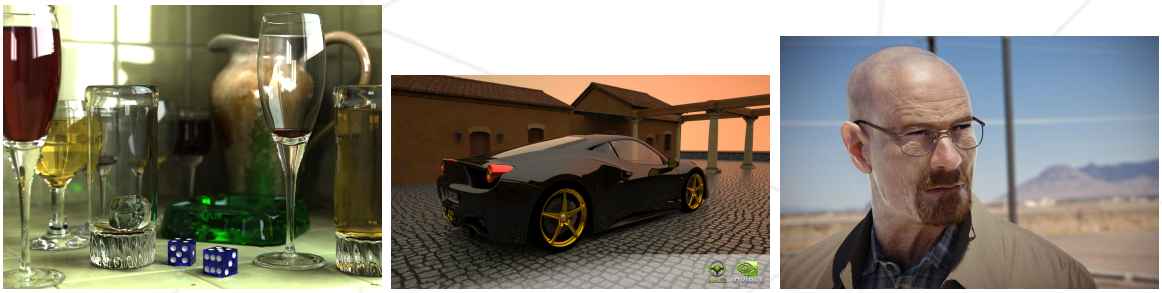

Figure III.1: The pictures above are rendered with the ray tracing technique . Impressive isn’t it?

Before you can even begin to produce such high quality graphics, you must master the basics: `RTv1` is your first ray tracer coded in `C`, normed, humble but functionnal.

At least you’ll then be able to show off nice looking pictures to justify the amount of hours you’re spending at school instead of spending time with your special person :-).

Chapter IV

General Instructions

- This project will be corrected by humans only. You're allowed to organize and name your files as you want, but you must follow the following rules.
- The executable file must be named `RTv1`.
- Your `Makefile` must compile the project and must contain the usual rules. It must recompile and re-link the program only if necessary.
- If you are clever, you will use your library for your `RTv1`. Submit also your folder `libft` including its own `Makefile` at the root of your repository. Your `Makefile` will have to compile the library, and then compile your project.
- Global variables are prohibited.
- Your project must be written in accordance with the Norm. Only norminette is authoritative.
- You have to handle errors carefully. In no way can your program quit in an unexpected manner (Segmentation fault, bus error, double free, etc).
- Your program cannot have memory leaks.
- You'll have to submit a file called `author` containing your username followed by a '\n' at the root of your repository.

```
$>cat -e author
xlogin$
```

- You can use `MacOS` native `MinilibX` library already installed on the imacs, or you can use MinilibX from its sources that you'll need to integrate similarly to libft. Last option, you can use additional graphic libraries (`X11`, `SDL`, etc...). If the library you are using is not installed on the imacs, you will have to submit the sources of this library in your repository, and it will have to be automatically compiled, without doing anything more than compiling your project, exactly like `MinilibX` or like your `libft`. No matter which graphic library you can only use its basic drawings functions similar to `MinilibX`: Open a window, lit a pixel and manage events.

- Within the mandatory part, you are allowed to use only the following libc functions:
  - `open`
  - `read`
  - `write`
  - `close`
  - `malloc`
  - `free`
  - `perror`
  - `strerror`
  - `exit`
  - All functions of the math library (`-lm` man `man 3 math`)
  - All functions of the `MinilibX` or their equivalent in another graphic library.
- You are allowed to use other functions or other libraries to complete the bonus part as long as their use is justified during your defense. Be smart!
- You can ask your questions on the forum, on slack...

Chapter V

Mandatory part

Your goal is to be able, with the help of your program, to generate images according to Raytracing protocol.

Those computer generated images will each represent a scene, as seen from a specific angle and position, defined by simple geometric objects, and each with its own lighting system.

- Implement the Ray-Tracing method to create a computer generated image.
- You need at least 4 simple geometric objects as a base (not composed): plane, sphere, cylinder and cone.
- Your program must be able to apply translation and rotation transformation to objects before displaying them. For example a sphere declared at `(0, 0, 0)` must be able to successfully translate to `(42, 42, 42)`.
- Manage the redraw view without recalculating (basically with MinilibX, you can manage the expose properly): if a part of the window needs to be redrawn, it is best if you don't have to recalculate everything.
- Position and direction of any point of vision and of simple objects.
- Smallest light management : different brightness, shadows.

For the defence, it would be ideal for you to have a whole set of scenes with the focus on what is functional, facilitating that way the control of the elements to create. For example:

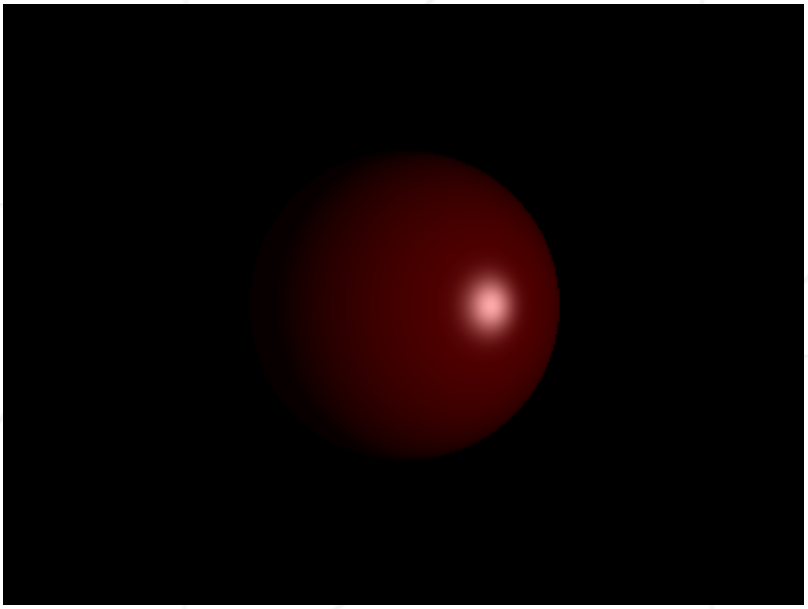

Figure V.1: A sphere, one spot, some shine (optional)

Figure V.2: A cylinder, one spot

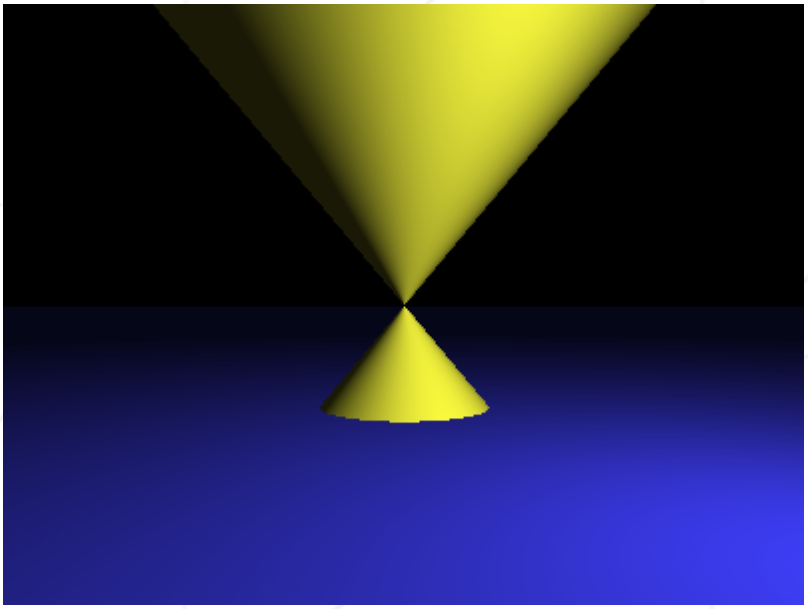

Figure V.3: A cone, a plane, one spot

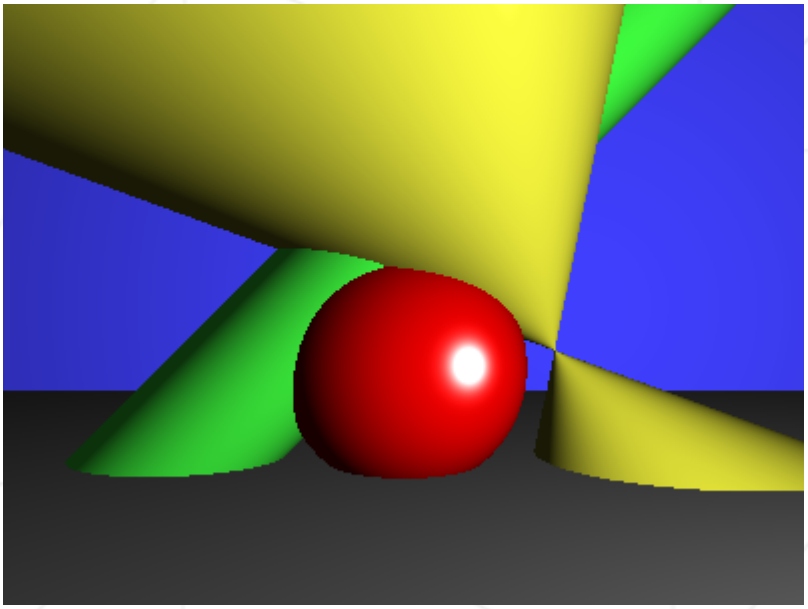

Figure V.4: A bit of everything, including 2 planes

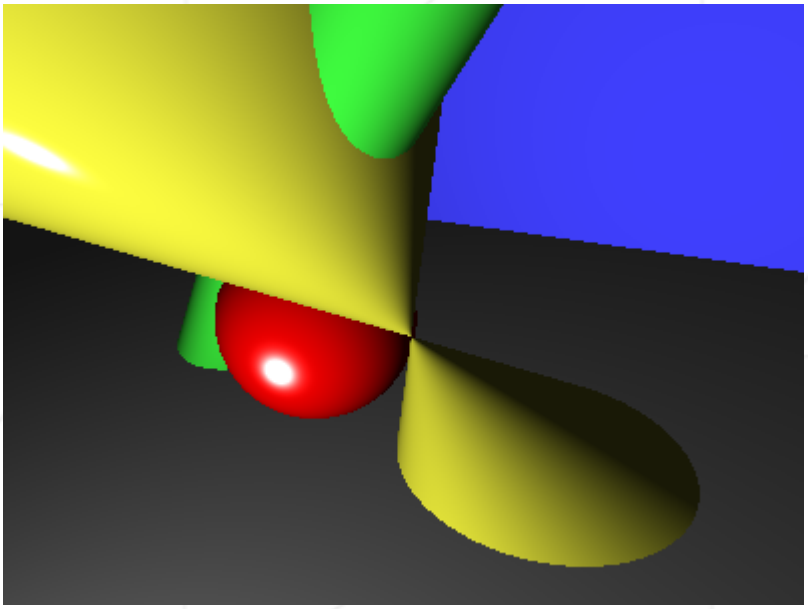

Figure V.5: Same scene different viewpoint

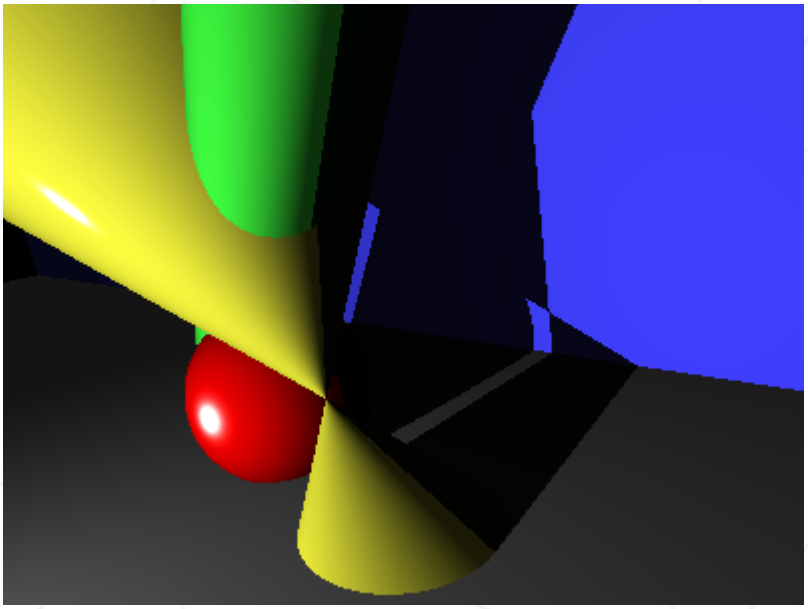

Figure V.6: This time with shadows

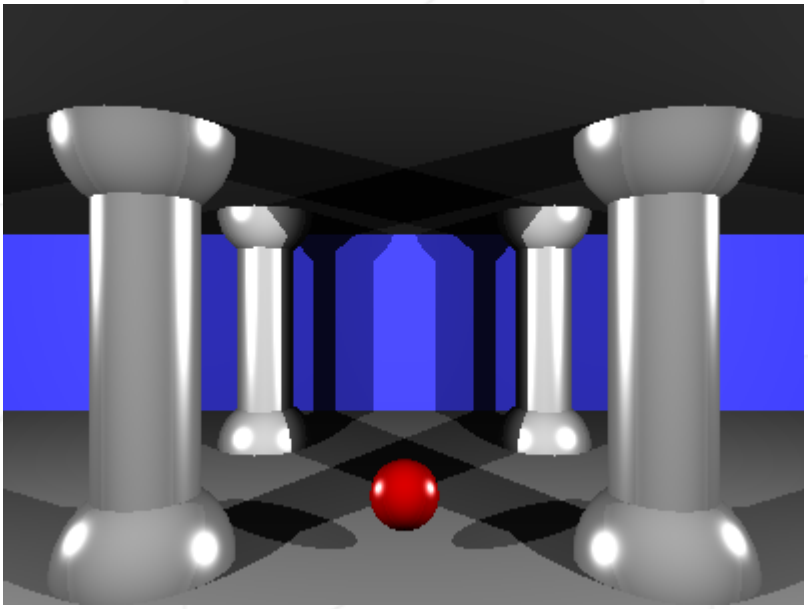

Figure V.7: And finally with multiple spots

Chapter VI

Bonus part

We will look at your bonuses if and only if your mandatory part is EXCELLENT. This means that your must complete the mandatory part, beginning to end, and your error management must be flawless, even in cases of twisted or bad usage. If that's not the case, your bonuses will be totally IGNORED.

Possible bonuses :

- Multi-spots
- Shine effect

No other bonuses will be accepted as they'll be part of the next project: RT.

Chapter VII

Submission and peer correction

Submit your work on your `GiT` repository as usual. Only the work on your repository will be graded.

Good luck to all and don't forget your author file!