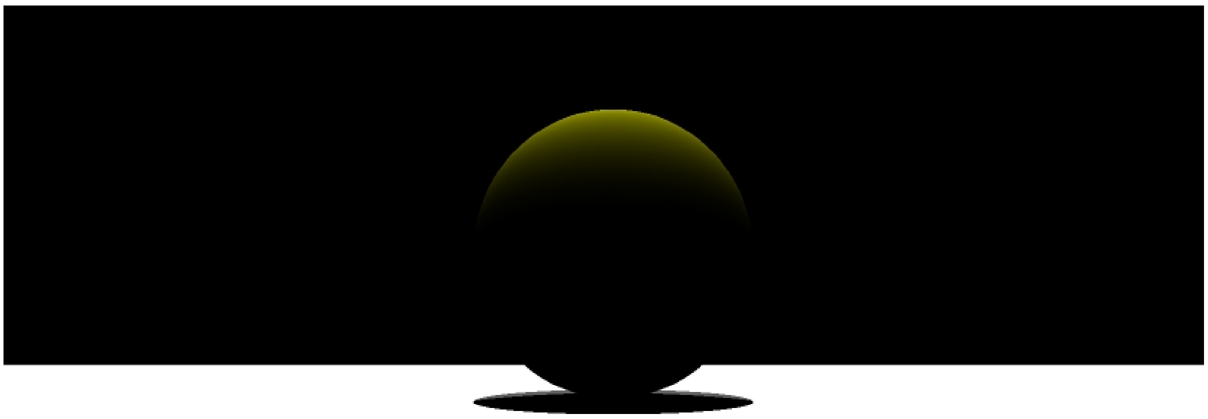
Raytracer

Technical documentation



Directed by :

- Axel Fradet axel.fradet@epitech.eu
- Mathieu Robert mathieu1.robert@epitech.eu
- Kylian Tranchet kylian.tranchet@epitech.eu
- Théophile Jérôme-Rocher theophile.jerome-rocher@epitech.eu
    - **Epitech Nantes PGE 2027. 2nd year.**

Useful links (technical):

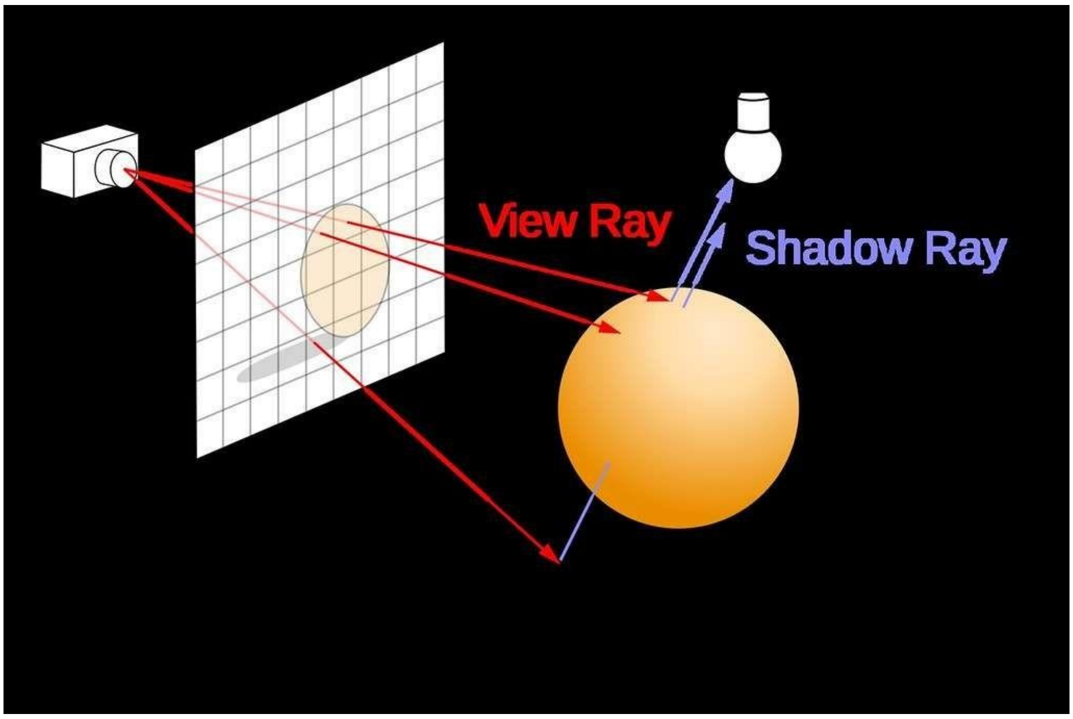
- Github: https://github.com/Njord201/Raytracer
- Github Project: https://github.com/users/Njord201/projects/7
- Commits standard: https://www.conventionalcommits.org/en/v1.0.0/
- Octree algorithm Documentation: https://www.geeksforgeeks.org/octree-insertion-and-searching/

Execution:

The root makefile offers make, make re, make fclean, make clean, make doc.

Technical point:

For this raytracing, we simply have a ray (origin + vector) emitted by a camera, passing through a "rectangle", and depending on whether this ray intersects with a primitive, we put colour on the rectangle. It's simply a matter of calculating vectors and intersections...



Technical point on the optimisation algorithm:

In order to optimise image generation, we had to implement an algorithm. In this case, the 'Octree' algorithm (see useful links). We also use anti-aliasing to obtain smoother renderings.

Technology:

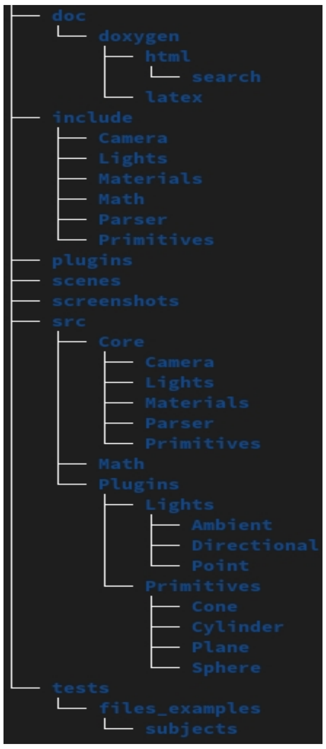
- C++ (Language), Make (Tool), SDL2 (Graphical user interface)
- For documentation: Doxygen, LaTeX, GraphViz
    - Run "**sudo yum install doxygen doxygen-latex graphviz".**

Developer constraints:

- Follow the commit standard (Conventional Commits).
- Follow the standard for variables, file names, etc. --> Camel case.
- Open an issue, create a branch from it, and push on main using pull requests only. Minimum 2 reviews approved.
- Rigorous coding (standard / cleanliness).
- Follow the current architecture of the project.
- Using and updating the github project.

A quick look at the structure:

- Primitives, lights, etc. must be "plugins", i.e. .so or shared objects, which are then re-injected when they are loaded. We therefore have a Makefile compiling the Core and the plugins in two stages, which themselves have their own Makefiles (makefile in Plugins/ or Core/)...



```
├── doc
│   └── doxygen
│       ├── html
│       │   └── search
│       └── latex
├── include
│   ├── Camera
│   ├── Lights
│   ├── Materials
│   ├── Math
│   ├── Parser
│   └── Primitives
├── plugins
├── scenes
├── screenshots
├── src
│   ├── Core
│   │   ├── Camera
│   │   ├── Lights
│   │   ├── Materials
│   │   ├── Parser
│   │   └── Primitives
│   ├── Math
│   └── Plugins
│       ├── Lights
│       │   ├── Ambient
│       │   ├── Directional
│       │   └── Point
│       └── Primitives
│           ├── Cone
│           ├── Cylinder
│           ├── Plane
│           └── Sphere
└── tests
    └── files_examples
        └── subjects
```

include/ all hpp files.

src/ all .cpp files.

scenes/ contains the .cfg files for the scenes.

screenshots/ images provided by the developers of this Raytracer.

plugins/the .so of compiled plugins.

tests/home tests.

doc/ documentation.

Create a new primitive (developers):

- Create a new primitive in include/Primitives, following the logic of existing primitives based on the IPrimitive interface.
- The same applies to C++ code in src/Plugins/Primitives.
- Remember to update the Plugins Makefile with the correct references.

Access technical documentation:

If you would like more information on the classes and different structures used in our programme. You can access **refman.pdf** in **/doc**. You can run **"make doc"** to re-generate the documentation, this pdf and also
/doc/doxygen/html which will then contain a site that you can launch to access the Raytracer doxygen documentation, including graphics.