# SyntheseImage : RayTracing

**Authors: LOPES MENDES Ailton, LAMBERT-DELAVAQUERIE Fabien**

## Introduction

Ray tracing is a rendering technique for generating an image by tracing the path of light as pixels in an image plane and simulating the effects of its encounters with virtual objects. This technique will produce a high degree of visual realism.

## Installation

```
/SyntheseImage/Kit.M1.M2.2020$  chmod a+x install.sh
/SyntheseImage/Kit.M1.M2.2020$  cd libg2x
/SyntheseImage/Kit.M1.M2.2020/libg2x$ make
/SyntheseImage/Kit.M1.M2.2020$  cd libg3x
/SyntheseImage/Kit.M1.M2.2020/libg3x$ make
```

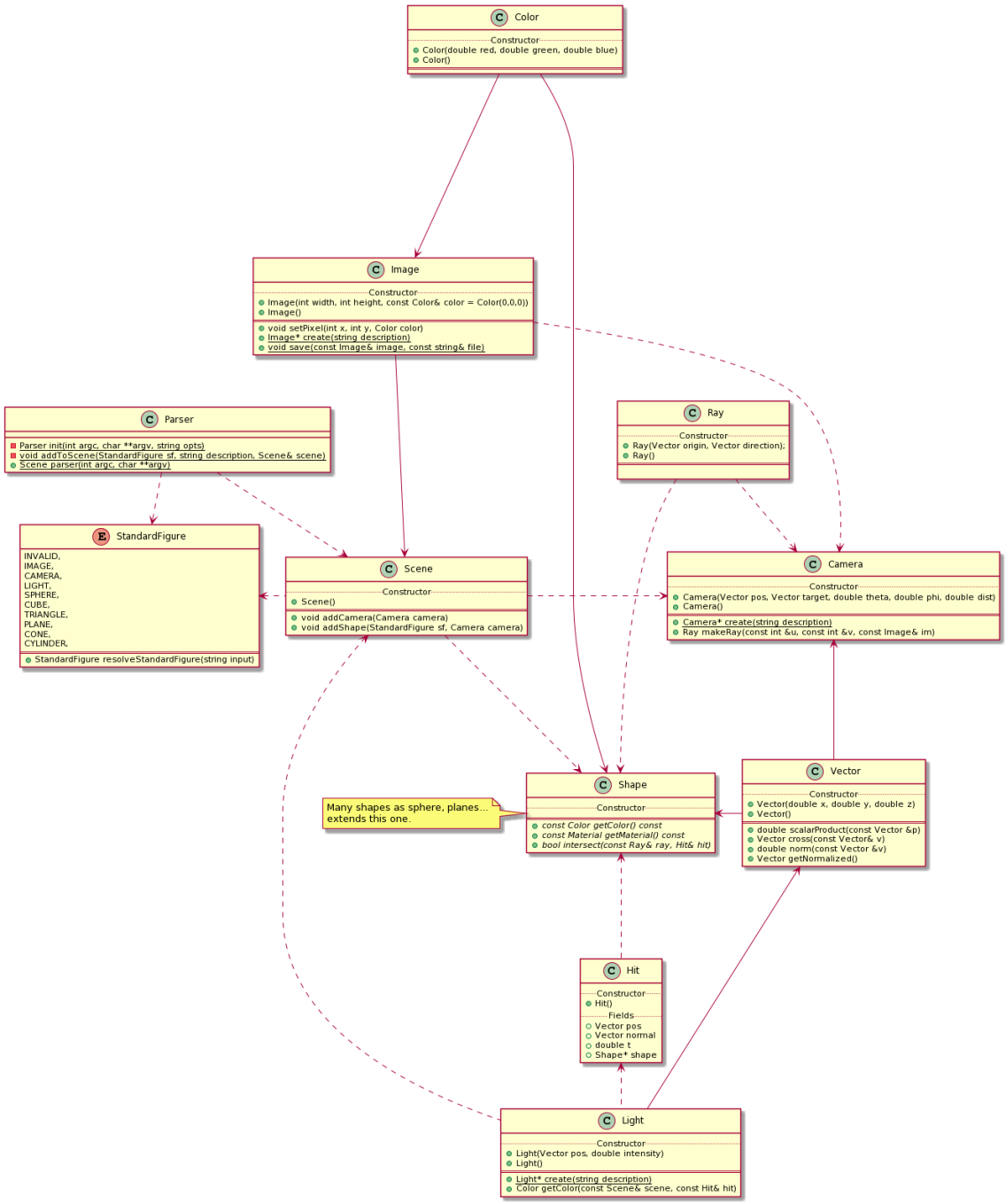
## Usage

| opt | description |
|---|---|
| -n | 1 to 3 |
| -i | input file format .txt |
| -o | output file format .ppm |
| -ps | number of rays > 0 |

**Exemple**

```
/SyntheseImage/Kit.M1.M2.2020$  cd 3D
/SyntheseImage/Kit.M1.M2.2020/3D$  make
/SyntheseImage/Kit.M1.M2.2020/3D$  ./lray -n 1 -i input/sphere.txt -o image.ppm
/SyntheseImage/Kit.M1.M2.2020/3D$  ./lray -n 3 -i input/sphere.txt -o i64.ppm -ps 64
```

## UML


Color
Constructor
Color(double red, double green, double blue)
Color()
Image
Constructor
Image(int width, int height, const Color& color = Color(0,0,0))
Image()
void setPixel(int x, int y, Color color)
Image* create(string description)
void save(const Image& image, const string& file)
Parser
Parser init(int argc, char **argv, string opts)
void addToScene(StandardFigure sf, string description, Scene& scene)
Scene parser(int argc, char **argv)
Ray
Constructor
Ray(Vector origin, Vector direction);
Ray()
StandardFigure
INVALID,
IMAGE,
CAMERA,
LIGHT,
SPHERE,
CUBE,
TRIANGLE,
PLANE,
CONE,
CYLINDER,
StandardFigure resolveStandardFigure(string input)
Scene
Constructor
Scene()
void addCamera(Camera camera)
void addShape(StandardFigure sf, Camera camera)
Camera
Constructor
Camera(Vector pos, Vector target, double theta, double phi, double dist)
Camera()
Camera* create(string description)
Ray makeRay(const int &u, const int &v, const Image& im)
Shape
Constructor
const Color getColor() const
const Material getMaterial() const
bool intersect(const Ray& ray, Hit& hit)
Many shapes as sphere, planes...
extends this one.
Vector
Constructor
Vector(double x, double y, double z)
Vector()
double scalarProduct(const Vector &p)
Vector cross(const Vector& v)
double norm(const Vector &v)
Vector getNormalized()
Hit
Constructor
Hit()
Fields
Vector pos
Vector normal
double t
Shape* shape
Light
Constructor
Light(Vector pos, double intensity)
Light()
Light* create(string description)
Color getColor(const Scene& scene, const Hit& hit)