RayTracer

0.9

Generated by Doxygen 1.13.2

Chapter 1

Raytracer Project

1.1 Project Overview

This project is a raytracer implementation designed to render 3D scenes based on configuration files. It uses a modular architecture to separate concerns such as mathematical operations, scene parsing, and rendering primitives. The project is written in C++ and follows object-oriented principles.

1.2 Project Structure

Here is the directory tree of the project:

```
raytracer/
  include/
    Math/
      Color.hpp
      Ray.hpp
      Transform.hpp
      Vector3D.hpp
    Raytracer/
      Lights/
        AmbientLight.hpp
        DirectionalLight.hpp
        ILight.hpp
      Materials/
        FlatMaterial.hpp
        IMaterial.hpp
      Primitives/
        Cone.hpp
        Cylinder.hpp
        IPrimitive.hpp
        Plane.hpp
        Sphere.hpp
      Scene/
        Camera.hpp
        Image.hpp
        Renderer.hpp
        Scene.hpp
        SceneLoader.hpp

    Utils/
      ConfigParser.hpp
      Error.hpp
      Logger.hpp
      Parser.hpp
      PPMWriter.hpp
      Timer.hpp
  src/
    main.cpp
    Math/
      Color.cpp
      Ray.cpp
      Transform.cpp
      Vector3D.cpp
    Raytracer/
      Lights/
        AmbientLight.cpp
```

```
            DirectionalLight.cpp
            ILight.cpp
        Materials/
            FlatMaterial.cpp
            IMaterial.cpp
        Primitives/
            Cone.cpp
            Cylinder.cpp
            IPrimitive.cpp
            Plane.cpp
            Sphere.cpp
        Scene/
            Camera.cpp
            Image.cpp
            Renderer.cpp
            Scene.cpp
            SceneLoader.cpp
    Utils/
        ConfigParser.cpp
        Error.cpp
        Logger.cpp
        Parser.cpp
        PPMWriter.cpp
        Timer.cpp
scenes/
    scene1.json
    scene2.json
    scene3.json
screenshots/
    scene1.ppm
Makefile
README.md
```

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

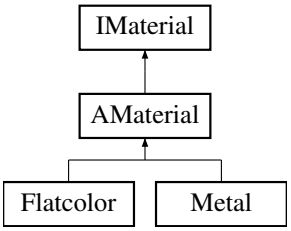
Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 AMaterial Class Reference

Inheritance diagram for AMaterial:



Public Member Functions

- AMaterial (const std::string &name="")
- bool scatter (const Ray &rayIn, const Structs::hitRecord &rec, Vector3D &attenuation, Ray &scattered) const =0

Public Attributes

- std::string name

5.1.1 Member Function Documentation

5.1.1.1 scatter()

bool AMaterial::scatter (
const Ray & rayIn,
const Structs::hitRecord & rec,
Vector3D & attenuation,
Ray & scattered) const [pure virtual]

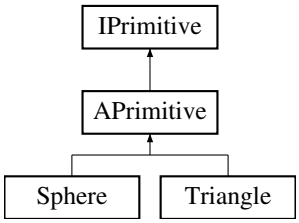
Implements IMaterial.

The documentation for this class was generated from the following file:

- include/Abstracts/AMaterial.hpp

5.2 APrimitive Class Reference

Inheritance diagram for APrimitive:



Public Member Functions

- APrimitive (const Point3D &c=Point3D(0, 0, 0), Structs::Color col=Structs::Color{255, 255, 255}, std::shared_ptr< AMaterial > material=nullptr)
- bool hit (const Ray &r, double t_min, double t_max, Structs::hitRecord &rec) const override=0

Public Attributes

- Point3D center
- Structs::Color color
- std::shared_ptr< AMaterial > material

5.2.1 Member Function Documentation

5.2.1.1 hit()

bool APrimitive::hit (
const Ray & r,
double t_min,
double t_max,
Structs::hitRecord & rec) const [override], [pure virtual]

Implements IPrimitive.

The documentation for this class was generated from the following file:

- include/Abstracts/APrimitive.hpp

5.3 Camera Class Reference

Public Member Functions

- Camera (const Point3D &pos, double fieldOfView)
- Camera (const Vector3D &origin, const Vector3D &lower_l, const Vector3D &horiz, const Vector3D &verti)
- Ray getRay (double u, double v)

Public Attributes

- Vector3D lower_left
- Vector3D horizontal
- Vector3D vertical
- Vector3D origin

The documentation for this class was generated from the following file:

- include/Camera.hpp

5.4 Structs::Camera Struct Reference

Public Attributes

- int width

- int height
- double pos_x
- double pos_y
- double pos_z
- double rot_x
- double rot_y
- double rot_z
- double fieldOfView

The documentation for this struct was generated from the following file:

- include/Utils/structs.hpp

5.5 ConfigStruct::Color Struct Reference

Public Attributes

- int r
- int g
- int b

The documentation for this struct was generated from the following file:

- include/structs/PrimitesStructs.hpp

5.6 Structs::Color Struct Reference

Public Attributes

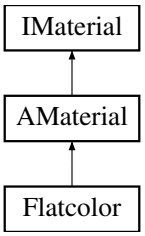
- int r
- int g
- int b

The documentation for this struct was generated from the following file:

- include/Utils/structs.hpp

5.7 Flatcolor Class Reference

Inheritance diagram for Flatcolor:



Public Member Functions

- Flatcolor (const std::string &n="flatcolor")
- bool scatter (const Ray &rayIn, const Structs::hitRecord &rec, Vector3D &attenuation, Ray &scattered) const

Public Member Functions inherited from AMaterial

- AMaterial (const std::string &name="")

Additional Inherited Members

Public Attributes inherited from AMaterial

- std::string name

5.7.1 Member Function Documentation

5.7.1.1 scatter()

bool Flatcolor::scatter (
const Ray & rayIn,
const Structs::hitRecord & rec,
Vector3D & attenuation,
Ray & scattered) const [inline], [virtual]

Implements AMaterial.

The documentation for this class was generated from the following file:

- include/RayTracer/Materials/Flatcolor.hpp

5.8 Structs::hitRecord Struct Reference

Public Attributes

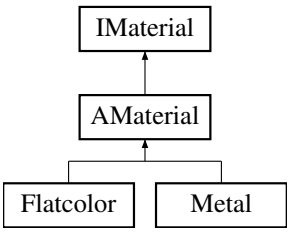
- double t
- Vector3D point
- Vector3D normal
- std::shared_ptr< AMaterial > material = nullptr
- Structs::Color color

The documentation for this struct was generated from the following file:

- include/Utils/structs.hpp

5.9 IMaterial Class Reference

Inheritance diagram for IMaterial:



Public Member Functions

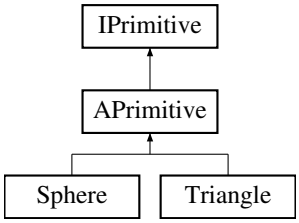
- virtual bool scatter (const Ray &rayIn, const Structs::hitRecord &rec, Vector3D &attenuation, Ray &scattered) const =0

The documentation for this class was generated from the following file:

- include/Interfaces/IMaterial.hpp

5.10 IPrimitive Class Reference

Inheritance diagram for IPrimitive:



Public Member Functions

- virtual bool hit (const Ray &r, double t_min, double t_max, Structs::hitRecord &rec) const =0

The documentation for this class was generated from the following file:

- include/Interfaces/IPrimitive.hpp

5.11 ConfigStruct::Light Struct Reference

Classes

- struct point

Public Attributes

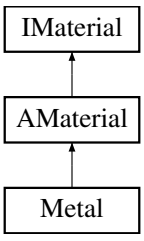
- double ambient
- double diffuse

The documentation for this struct was generated from the following file:

- include/structs/LightStruct.hpp

5.12 Metal Class Reference

Inheritance diagram for Metal:



Public Member Functions

- Metal (const std::string &n="metal")
- bool scatter (const Ray &rayIn, const Structs::hitRecord &rec, Vector3D &attenuation, Ray &scattered) const

Public Member Functions inherited from AMaterial

- AMaterial (const std::string &name="")

Additional Inherited Members

Public Attributes inherited from AMaterial

- std::string name

5.12.1 Member Function Documentation

5.12.1.1 scatter()

bool Metal::scatter (
const Ray & rayIn,
const Structs::hitRecord & rec,
Vector3D & attenuation,
Ray & scattered) const [inline], [virtual]

Implements AMaterial.

The documentation for this class was generated from the following file:

- include/RayTracer/Materials/Metal.hpp

5.13 Params Struct Reference

Public Attributes

- Point3D center
- Structs::Color color
- std::shared_ptr< AMaterial > material
- double radius
- int size_x
- int size_y

The documentation for this struct was generated from the following file:

- include/PrimitiveBuilder.hpp

5.14 Parser Class Reference

Public Member Functions

- Parser (Parser &other)=delete
- void operator= (const Parser &other)=delete
- void ParseConfig (Screen *s)
- Structs::Camera & getCameraConfig (void)

Static Public Member Functions

- static Parser * GetInstance (const std::string &path)

Public Attributes

- Structs::Camera * mCameraConfig
- int antiAliasing

Protected Member Functions

- Parser (const std::string &path)
- void ParseCamera (const ConfSetting &cam, const ConfSetting &res, const ConfSetting &pos, const ConfSetting &rota)
- void ParseSphere (Screen *s, const ConfSetting &sphere)

Protected Attributes

- std::string mConfigPath

Static Protected Attributes

- static Parser * mParser = nullptr

The documentation for this class was generated from the following files:

- include/Parser.hpp
- src/Parser.cpp

5.15 ConfigStruct::Light::point Struct Reference

Public Attributes

- double x
- double y
- double z

The documentation for this struct was generated from the following file:

- include/structs/LightStruct.hpp

5.16 PrimitiveBuilder Class Reference

Public Member Functions

- PrimitiveBuilder & setCenter (const Point3D &center)
- PrimitiveBuilder & setMaterial (std::shared_ptr< AMaterial > material)
- PrimitiveBuilder & setRadius (double radius)
- PrimitiveBuilder & setColor (Structs::Color color)
- std::unique_ptr< APrimitive > createSphere (void)

The documentation for this class was generated from the following file:

- include/PrimitiveBuilder.hpp

5.17 Ray Class Reference

Public Member Functions

- Ray (const Vector3D &a, const Vector3D &b)
- Vector3D getOrigin (void) const
- Vector3D getDirection (void) const
- Vector3D pointAtParameter (double t) const

The documentation for this class was generated from the following files:

- include/Math/ray.hpp
- src/ray.cpp

5.18 Screen Class Reference

Public Member Functions

- bool checkForHit (const Ray &r, double t_min, double t_max, Structs::hitRecord &rec) const
- Vector3D getColor (const Ray &ray, int depth)
- void startRendering (void)

Public Attributes

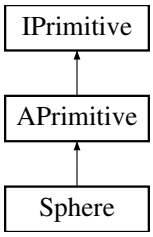
- std::vector< std::unique_ptr< APrimitive > > mPrimitives
- std::map< std::string, std::shared_ptr< AMaterial > > mMaterials

The documentation for this class was generated from the following file:

- include/Screen.hpp

5.19 Sphere Class Reference

Inheritance diagram for Sphere:



Public Member Functions

- Sphere (const Point3D &center, Structs::Color color, std::shared_ptr< AMaterial > material, double radius)
- bool hit (const Ray &r, double t_min, double t_max, Structs::hitRecord &rec) const override

Public Member Functions inherited from APrimitive

- APrimitive (const Point3D &c=Point3D(0, 0, 0), Structs::Color col=Structs::Color{255, 255, 255}, std::shared_ptr< AMaterial > material=nullptr)

Public Attributes

- double radius

Public Attributes inherited from APrimitive

- Point3D center
- Structs::Color color
- std::shared_ptr< AMaterial > material

5.19.1 Member Function Documentation

5.19.1.1 hit()

bool Sphere::hit (
const Ray & r,
double t_min,
double t_max,
Structs::hitRecord & rec) const [inline], [override], [virtual]

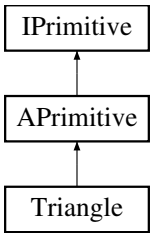
Implements APrimitive.

The documentation for this class was generated from the following file:

- include/RayTracer/Primitives/Sphere.hpp

5.20 Triangle Class Reference

Inheritance diagram for Triangle:



Public Member Functions

- Triangle (const Vector3D &center, Structs::Color color, std::shared_ptr< AMaterial > material, const Vector3D &v0, const Vector3D &v1, const Vector3D &v2)
- bool hit (const Ray &ray, double t_min, double t_max, Structs::hitRecord &rec) const override

Public Member Functions inherited from APrimitive

- APrimitive (const Point3D &c=Point3D(0, 0, 0), Structs::Color col=Structs::Color{255, 255, 255}, std::shared_ptr< AMaterial > material=nullptr)

Additional Inherited Members

Public Attributes inherited from APrimitive

- Point3D center
- Structs::Color color
- std::shared_ptr< AMaterial > material

5.20.1 Member Function Documentation

5.20.1.1 hit()

```
bool Triangle::hit (
            const Ray & ray,
            double t_min,
            double t_max,
            Structs::hitRecord & rec) const    [override], [virtual]
```

Implements APrimitive.
The documentation for this class was generated from the following file:

- plugins/primitives/Triangle.hpp

5.21 Vector3D Class Reference

Public Member Functions

- Vector3D (double xa, double ya, double za)
- Vector3D & operator+= (const Vector3D &v)
- Vector3D & operator*= (const Vector3D &v)
- Vector3D & operator/= (const Vector3D &v)
- Vector3D & operator* (const Vector3D &v)
- Vector3D & operator/ (const Vector3D &v)
- Vector3D & operator*= (const double t)
- Vector3D & operator/= (const double t)
- Vector3D & operator* (const double t)

- Vector3D & operator/ (const double a)
- double length (void) const
- double length_sqrt (void) const

Public Attributes

- double x
- double y
- double z

Friends

- Vector3D operator+= (const Vector3D &v, const Vector3D &a)
- Vector3D operator*= (const Vector3D &v, const Vector3D &a)
- Vector3D operator/= (const Vector3D &v, const Vector3D &a)
- Vector3D operator* (const Vector3D &v, const Vector3D &a)
- Vector3D operator/ (const Vector3D &v, const Vector3D &a)
- Vector3D operator*= (const double t, const Vector3D &v)
- Vector3D operator/= (const double t, const Vector3D &v)
- Vector3D operator* (const double t, const Vector3D &v)
- Vector3D operator/ (const double a, const Vector3D &v)

The documentation for this class was generated from the following files:

- include/Math/vector3D.hpp
- src/vector3D.cpp

Chapter 6

File Documentation

6.1 Color.hpp

00001

6.2 Ray.hpp

00001

6.3 Transform.hpp

00001

6.4 Vector3D.hpp

00001

6.5 AmbientLight.hpp

00001

6.6 DirectionalLight.hpp

00001

6.7 FlatMaterial.hpp

00001

6.8 Cone.hpp

00001

6.9 Cylinder.hpp

00001

6.10 Plane.hpp

00001

6.11 Image.hpp

00001

6.12 Renderer.hpp

00001

6.13 Scene.hpp

00001

6.14 SceneLoader.hpp

00001

6.15 ConfigParser.hpp

00001

6.16 Error.hpp

00001

6.17 Logger.hpp

00001

6.18 PPMWriter.hpp

00001

6.19 Timer.hpp

00001

6.20 ALight.hpp

00001

6.21 AMaterial.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** AMaterial
00006 */
00007
00008 #ifndef AMATERIAL_HPP_
00009 #define AMATERIAL_HPP_
00010
00011 #include "Math/ray.hpp"
00012 #include "Utils/structs.hpp"
00013 #include "Interfaces/IMaterial.hpp"
00014
00015 class AMaterial : public IMaterial {
00016     public:
```

```
00017         AMaterial(const std::string &name = "") {};
00018         virtual ~AMaterial() = default;
00019         bool scatter(const Ray &rayIn, const Structs::hitRecord &rec, Vector3D &attenuation, Ray &scattered) const = 0;
00020
00021         std::string name;
00022 };
00023
00024 #endif /* !AMATERIAL_HPP_ */
```

6.22 APrimitive.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** APrimitive
00006 */
00007
00008 #ifndef APRIMITIVE_HPP_
00009 #define APRIMITIVE_HPP_
00010
00011 #include "Interfaces/IPrimitive.hpp"
00012 #include "Math/vector3D.hpp"
00013 #include "Abstracts/AMaterial.hpp"
00014
00015 class APrimitive : public IPrimitive {
00016     public:
00017         APrimitive(const Point3D &c = Point3D(0,0,0), Structs::Color col = Structs::Color{255,255,255},
      std::shared_ptr<AMaterial> material = nullptr) :
00018             center(c), color(col), material(material) {};
00019         virtual ~APrimitive() = default;
00020         bool hit(const Ray &r, double t_min, double t_max, Structs::hitRecord &rec) const override = 0;
00021
00022         /* Primitive's common data */
00023         Point3D center;
00024         Structs::Color color;
00025         std::shared_ptr<AMaterial> material;
00026 };
00027
00028 #endif /* !APRIMITIVE_HPP_ */
```

6.23 Camera.hpp

```
00001
```

6.24 Camera.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** Camera
00006 */
00007
00008 #ifndef CAMERA_HPP_
00009 #define CAMERA_HPP_
00010
00011 #include "Math/ray.hpp"
00012 #include "Math/vector3D.hpp"
00013
00014 class Camera {
00015     public:
00016         Camera(const Point3D &pos, double fieldOfView) : lower_left(pos + Vector3D(-2, -1, -1)), horizontal(pos +
      Vector3D(4, 0, 0)), vertical(pos + Vector3D(0, 2, 0)),
00017             origin(pos + Vector3D(0, 0, 0)) {};
00018         Camera(const Vector3D &origin, const Vector3D &lower_l, const Vector3D &horiz, const Vector3D &verti) :
00019             origin(origin), lower_left(lower_l), horizontal(horiz), vertical(verti) {};
00020         ~Camera() = default;
00021
00022         Ray getRay(double u, double v) {
00023             return Ray(origin, lower_left + u * horizontal + v * vertical);
00024         }
00025
00026         Vector3D lower_left;
00027         Vector3D horizontal;
00028         Vector3D vertical;
00029         Vector3D origin;
00030 };
00031 #endif /* !CAMERA_HPP_ */
```

6.25 ILight.hpp

00001

6.26 ILight.hpp

00001

6.27 IMaterial.hpp

00001

6.28 IMaterial.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** IMaterial
00006 */
00007
00008 #ifndef IMATERIAL_HPP_
00009 #define IMATERIAL_HPP_
00010
00011 #include "IPrimitive.hpp"
00012 #include "Utils/structs.hpp"
00013
00014 class IMaterial {
00015     public:
00016         virtual ~IMaterial() = default;
00017         virtual bool scatter(const Ray &rayIn, const Structs::hitRecord &rec, Vector3D &attenuation, Ray &scattered) const
      = 0;
00018 };
00019
00020 #endif /* !IMATERIAL_HPP_ */
```

6.29 IPrimitive.hpp

00001

6.30 IPrimitive.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** IPrimitive
00006 */
00007
00008 #ifndef IPRIMITIVE_HPP_
00009 #define IPRIMITIVE_HPP_
00010
00011 #include "Math/vector3D.hpp"
00012 #include "Math/ray.hpp"
00013
00014 class IPrimitive {
00015     public:
00016         virtual ~IPrimitive() = default;
00017         virtual bool hit(const Ray &r, double t_min, double t_max, Structs::hitRecord &rec) const = 0;
00018 };
00019
00020 #endif /* !IPRIMITIVE_HPP_ */
```

6.31 materialOperations.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
```

```
00005 ** materialOperations
00006 */
00007
00008 #ifndef MATERIALOPERATIONS_HPP_
00009 #define MATERIALOPERATIONS_HPP_
00010
00011 #include "Math/vector3D.hpp"
00012 #include "Math/operators.hpp"
00013
00014 inline Vector3D randomInUnitSphere() {
00015     Vector3D p;
00016     do {
00017         p = 2.9 * Vector3D(drand48(), drand48(), drand48()) - Vector3D(1, 1, 1);
00018     } while (p.length_sqrt() >= 1.0);
00019     return p;
00020 }
00021
00022 #endif /* !MATERIALOPERATIONS_HPP_ */
```

6.32 operators.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** operators
00006 */
00007
00008 #ifndef OPERATORS_HPP_
00009 #define OPERATORS_HPP_
00010
00011 #include "Math/vector3D.hpp"
00012
00013 /*
00014 ** inline sert à déclarer qu'il peut il y avoir plusieurs définitions de la fonction
00015 */
00016
00017 inline Vector3D operator*(double t, const Vector3D &v)
00018 {
00019     return Vector3D(v.x * t, v.y * t, v.z * t);
00020 }
00021
00022 inline Vector3D operator/(const Vector3D &v, double s)
00023 {
00024     return Vector3D{v.x / s, v.y / s, v.z / s};
00025 }
00026
00027 inline Vector3D operator+(const Vector3D &v1, const Vector3D &v2)
00028 {
00029     return Vector3D(v1.x + v2.x, v1.y + v2.y, v1.z + v2.z);
00030 }
00031
00032 inline Vector3D operator-(const Vector3D &v1, const Point3D &v2)
00033 {
00034     return Vector3D(v1.x - v2.x, v1.y - v2.y, v1.z - v2.z);
00035 }
00036
00037 #endif /* !OPERATORS_HPP_ */
```

6.33 ray.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** ray
00006 */
00007
00008 #ifndef RAY_HPP_
00009 #define RAY_HPP_
00010
00011 // -----------------
00012 // p(t) = a + t * b
00013 // -----------------
00014 // p(t) : Point à la distance t
00015 // t : distance
00016 // a : point d'origine
00017 // b : Direction
00018
00019 #include "Math/vector3D.hpp"
00020 #include "Utils/structs.hpp"
```

```
00021 #include "Math/operators.hpp"
00022
00023 class Ray {
00024     public:
00025         Ray() {};
00026         Ray(const Vector3D &a, const Vector3D &b) : mOrigin(a), mDirection(b) {};
00027
00028         Vector3D getOrigin(void) const;
00029         Vector3D getDirection(void) const;
00030
00031         Vector3D pointAtParameter(double t) const;
00032
00033     private:
00034         Vector3D mOrigin;
00035         Vector3D mDirection;
00036 };
00037
00038 #endif /* !RAY_HPP_ */
```

6.34 vector3D.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** vector3D
00006 */
00007
00008 #ifndef VECTOR3D_HPP_
00009 #define VECTOR3D_HPP_
00010
00011 #include <cmath>
00012
00013 class Vector3D
00014 {
00015     public:
00016         /* Constructors / Destructors */
00017         Vector3D() : x(0), y(0), z(0) {};
00018         Vector3D(double xa, double ya, double za) : x(xa), y(ya), z(za) {};
00019         ~Vector3D() = default;
00020
00021         /* Operations */
00022         Vector3D &operator+=(const Vector3D &v);
00023         Vector3D &operator*=(const Vector3D &v);
00024         Vector3D &operator/=(const Vector3D &v);
00025         Vector3D &operator*(const Vector3D &v);
00026         Vector3D &operator/(const Vector3D &v);
00027         Vector3D &operator*=(const double t);
00028         Vector3D &operator/=(const double t);
00029         Vector3D &operator*(const double t);
00030         Vector3D &operator/(const double a);
00031
00032         /* Get data */
00033         double length(void) const;
00034         double length_sqrt(void) const;
00035
00036         /* Variables */
00037         double x;
00038         double y;
00039         double z;
00040
00041         /* Extern the operators */
00042         friend Vector3D operator+=(const Vector3D &v, const Vector3D &a);
00043         friend Vector3D operator*=(const Vector3D &v, const Vector3D &a);
00044         friend Vector3D operator/=(const Vector3D &v, const Vector3D &a);
00045         friend Vector3D operator*(const Vector3D &v, const Vector3D &a);
00046         friend Vector3D operator/(const Vector3D &v, const Vector3D &a);
00047         friend Vector3D operator*=(const double t, const Vector3D &v);
00048         friend Vector3D operator/=(const double t, const Vector3D &v);
00049         friend Vector3D operator*(const double t, const Vector3D &v);
00050         friend Vector3D operator/(const double a, const Vector3D &v);
00051 };
00052
00053 /* Un point 3D est un vecteur fixe */
00054 using Point3D = Vector3D;
00055
00056 #endif /* !VECTOR3D_HPP_ */
```

6.35 vectorOperations.hpp

```
00001 /*
```

```
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** vectorOperations
00006 */
00007
00008 #ifndef VECTOROPERATIONS_HPP_
00009 #define VECTOROPERATIONS_HPP_
00010
00011 #include "operators.hpp"
00012 #include "vector3D.hpp"
00013
00014 inline double dot(const Vector3D &a, const Vector3D &b)
00015 {
00016     return (a.x * b.x + a.y * b.y + a.z * b.z);
00017 }
00018
00019 inline Vector3D reflect(const Vector3D &v, const Vector3D &n)
00020 {
00021     return v - 2 * dot(v, n) * n;
00022 }
00023
00024 inline Vector3D unit_vector(const Vector3D &v)
00025 {
00026     return v / v.length();
00027 }
00028
00029 #endif /* !VECTOROPERATIONS_HPP_ */
```

6.36 Parser.hpp

```
00001
```

6.37 Parser.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** Parser
00006 */
00007
00008 /* SINGLETON DESIGN PATTERN */
00009
00010 #ifndef PARSER_HPP_
00011 #define PARSER_HPP_
00012
00013 #include <string>
00014 #include <vector>
00015 #include <memory>
00016 #include <iostream>
00017 #include <libconfig.h++>
00018
00019 /* Structs used for the parsing */
00020 #include "Utils/structs.hpp"
00021
00022 class Screen;
00023
00024 using ConfSetting = libconfig::Setting;
00025
00026 class Parser {
00027     public:
00028         Parser(Parser &other) = delete;
00029         void operator=(const Parser &other) = delete;
00030
00031         static Parser *GetInstance(const std::string &path);
00032
00033         void ParseConfig(Screen *s);
00034
00035         Structs::Camera &getCameraConfig(void);
00036
00037         // Ajouter la récup des différents éléments
00038         Structs::Camera *mCameraConfig;
00039         int antiAliasing;
00040
00041     protected:
00042         Parser(const std::string &path) : mConfigPath(path) {};
00043
00044         void ParseCamera(const ConfSetting &cam, const ConfSetting &res,
00045             const ConfSetting &pos, const ConfSetting &rota);
00046         void ParseSphere(Screen *s, const ConfSetting &sphere);
```

```
00047
00048         static Parser* mParser;
00049         std::string mConfigPath;
00050 };
00051
00052 #endif /* !PARSER_HPP_ */
```

6.38 PrimitiveBuilder.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** PrimitiveBuilder
00006 */
00007
00008 #ifndef PRIMITIVEBUILDER_HPP_
00009 #define PRIMITIVEBUILDER_HPP_
00010
00011 #include "Math/vector3D.hpp"
00012 #include "RayTracer/Primitives/Sphere.hpp"
00013 #include "Math/ray.hpp"
00014 #include <memory>
00015 #include <vector>
00016 #include <optional> // à voir pk ça ne fonctionne pas
00017
00018 struct Params {
00019     Point3D center;
00020     Structs::Color color;
00021     std::shared_ptr<AMaterial> material;
00022
00023     /* Optionnal */
00024     double radius;
00025     int size_x;
00026     int size_y;
00027 };
00028
00029 class PrimitiveBuilder {
00030     public:
00031         PrimitiveBuilder() {};
00032         ~PrimitiveBuilder() {};
00033
00034         PrimitiveBuilder &setCenter(const Point3D &center) {
00035             mParams.center = center;
00036             return *this;
00037         };
00038         PrimitiveBuilder &setMaterial(std::shared_ptr<AMaterial> material) {
00039             mParams.material = material;
00040             return *this;
00041         };
00042         PrimitiveBuilder &setRadius(double radius) {
00043             mParams.radius = radius;
00044             return *this;
00045         };
00046         PrimitiveBuilder &setColor(Structs::Color color) {
00047             mParams.color = color;
00048             return *this;
00049         }
00050
00051         std::unique_ptr<APrimitive> createSphere(void) {
00052             return std::make_unique<Sphere>(mParams.center, mParams.color, mParams.material, mParams.radius);
00053         };
00054     private:
00055         Params mParams;
00056 };
00057
00058 #endif /* !PRIMITIVEBUILDER_HPP_ */
```

6.39 Flatcolor.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** Flatcolor
00006 */
00007
00008 #ifndef FLATCOLOR_HPP_
00009 #define FLATCOLOR_HPP_
00010
00011 #include "Abstracts/AMaterial.hpp"
```

```
00012 #include "Math/vectorOperations.hpp"
00013 #include "Math/materialOperations.hpp"
00014
00015 class Flatcolor : public AMaterial {
00016     public:
00017         Flatcolor(const std::string &n = "flatcolor") {
00018             name = n;
00019         };
00020         ~Flatcolor() = default;
00021
00022         bool scatter(const Ray &rayIn, const Structs::hitRecord &rec, Vector3D &attenuation, Ray &scattered) const
00023         {
00024             Vector3D target = rec.point + rec.normal + randomInUnitSphere();
00025             scattered = Ray(rec.point, target - rec.point);
00026             attenuation = Vector3D((rec.color.r / 255), (rec.color.g / 255), (rec.color.b / 255));
00027             return true;
00028         };
00029 };
00030
00031 #endif /* !FLATCOLOR_HPP_ */
```

6.40 Metal.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** Metal
00006 */
00007
00008 #ifndef METAL_HPP_
00009 #define METAL_HPP_
00010
00011 #include "Abstracts/AMaterial.hpp"
00012 #include "Math/vectorOperations.hpp"
00013
00014 class AMaterial;
00015
00016 class Metal : public AMaterial{
00017     public:
00018         Metal(const std::string &n = "metal") {
00019             name = n;
00020         };
00021         ~Metal() = default;
00022
00023         bool scatter(const Ray &rayIn, const Structs::hitRecord &rec, Vector3D &attenuation, Ray &scattered) const
00024         {
00025             Vector3D reflected = reflect(unit_vector(rayIn.getDirection()), rec.normal);
00026             scattered = Ray(rec.point, reflected);
00027             Vector3D objectColor = Vector3D(rec.color.r / 255.0, rec.color.g / 255.0, rec.color.b / 255.0);
00028             attenuation = objectColor;
00029             return true;
00030         };
00031 };
00032
00033 #endif /* !METAL_HPP_ */
```

6.41 Plate.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** Plate
00006 */
00007
00008 #ifndef PLATE_HPP_
00009 #define PLATE_HPP_
00010
00011 #include "Math/ray.hpp"
00012 #include "Math/vector3D.hpp"
00013 #include "Utils/structs.hpp"
00014 #include "Math/operators.hpp"
00015 #include "Abstracts/APrimitive.hpp"
00016
00017 // class Plate : public APrimitive {
00018 //     public:
00019 //         Plate(const Point3D &center, Structs::Color color, AMaterial *materialsx, double sy) : center(c), color(col),
      size_x(sx), size_y(sy) {};
00020 //         ~Plate() = default;
00021
```

```
00022 //         double hits(const Ray &ray, Structs::hitRecord &hit) {
00023 //             Vector3D normal(0, 1, 0);
00024 //             double d = dot (normal, ray.getDirection());
00025 //             if (std::abs(d) > 0) {
00026 //                 double t = dot((center - ray.getOrigin()), normal) / d;
00027 //                 if (t >= 0) {
00028 //                     Point3D hitPoint = ray.getOrigin() + ray.getDirection() * t;
00029 //                     if (std::abs(hitPoint.x - center.x) <= size_x / 2 &&
00030 //                         std::abs(hitPoint.z - center.z) <= size_y / 2) {
00031 //                         return t;
00032 //                     }
00033 //                 }
00034 //             }
00035 //             return -1.0;
00036 //         };
00037
00038 //         Point3D center;
00039 //         double size_x;
00040 //         double size_y;
00041 //         Structs::Color color;
00042 // };
00043
00044 #endif /* !PLATE_HPP_ */
```

6.42 Sphere.hpp

00001

6.43 Sphere.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** Sphere
00006 */
00007
00008 #ifndef SPHERE_HPP_
00009 #define SPHERE_HPP_
00010
00011 #include "Abstracts/APrimitive.hpp"
00012 #include "Math/ray.hpp"
00013 #include "Math/vector3D.hpp"
00014 #include "Utils/structs.hpp"
00015 #include "Math/operators.hpp"
00016 #include "Math/vectorOperations.hpp"
00017
00018 class Sphere : public APrimitive {
00019     public:
00020         Sphere(const Point3D &center, Structs::Color color, std::shared_ptr<AMaterial> material, double radius)
00021             : APrimitive(center, color, material), radius(radius) {};
00022         ~Sphere() = default;
00023
00024         bool hit(const Ray &r, double t_min, double t_max, Structs::hitRecord &rec) const override
00025         {
00026             Vector3D oc = r.getOrigin() - center;
00027
00028             double a = dot(r.getDirection(), r.getDirection());
00029             double b = 2.0 * dot(oc, r.getDirection());
00030             double c = dot(oc, oc) - radius * radius;
00031
00032             /* Quadratic Equation */
00033             float discriminant = b * b - 4 * a * c;
00034
00035             if (discriminant > 0) {
00036                 double t = (-b - sqrt(discriminant)) / (2 * a);
00037                 if (t < t_max && t > t_min) {
00038                     rec.t = t;
00039                     rec.point = r.pointAtParameter(t);
00040                     rec.material = material;
00041                     rec.color = color;
00042                     rec.normal = (rec.point - center) / radius;
00043                     return true;
00044                 }
00045                 t = (-b + sqrt(discriminant)) / (2 * a);
00046                 if (t < t_max && t > t_min) {
00047                     rec.t = t;
00048                     rec.point = r.pointAtParameter(t);
00049                     rec.material = material;
00050                     rec.color = color;
00051                     rec.normal = (rec.point - center) / radius;
```

```
00052                 return true;
00053             }
00054         }
00055         return false;
00056     }
00057
00058     /* Params */
00059     double radius;
00060 };
00061
00062 #endif /* !SPHERE_HPP_ */
```

6.44 Screen.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** Screen
00006 */
00007
00008 #ifndef SCREEN_HPP_
00009 #define SCREEN_HPP_
00010
00011 #include "Camera.hpp"
00012 #include "Parser.hpp"
00013 #include "Math/ray.hpp"
00014 #include "Math/vector3D.hpp"
00015 #include "Math/operators.hpp"
00016 #include "PrimitiveBuilder.hpp"
00017 #include <algorithm>
00018 #include <map>
00019 /* Primitives */
00020 #include "Abstracts/APrimitive.hpp"
00021 #include "RayTracer/Primitives/Plate.hpp"
00022 #include "RayTracer/Primitives/Sphere.hpp"
00023 /* Materials */
00024 #include "Abstracts/AMaterial.hpp"
00025 #include "RayTracer/Materials/Metal.hpp"
00026 #include "RayTracer/Materials/Flatcolor.hpp"
00027
00028 inline bool compare(const std::unique_ptr<APrimitive> &a, const std::unique_ptr<APrimitive> &b)
00029 {
00030     return a->center.z < b->center.z;
00031 }
00032
00033 class Screen {
00034     public:
00035         Screen() {
00036             /* Create the default materials */
00037             mMaterials["flatcolor"] = std::make_shared<Flatcolor>();
00038             mMaterials["metal"] = std::make_shared<Metal>();
00039         };
00040         ~Screen() {};
00041
00042         bool checkForHit(const Ray &r, double t_min, double t_max, Structs::hitRecord &rec) const
00043         {
00044             Structs::hitRecord temp_rec;
00045             bool hitAnything = false;
00046
00047             for (const auto &primitive : mPrimitives) {
00048                 if (primitive->hit(r, t_min, t_max, temp_rec)) {
00049                     hitAnything = true;
00050                     rec = temp_rec;
00051                 }
00052             }
00053             return hitAnything;
00054         }
00055
00056         Vector3D getColor(const Ray &ray, int depth) {
00057             Structs::hitRecord rec;
00058             if (checkForHit(ray, 0.001, MAXFLOAT, rec)) {
00059                 Ray scattered;
00060                 Vector3D attenuation;
00061                 if (depth < 50 && rec.material->scatter(ray, rec, attenuation, scattered)) {
00062                     return attenuation * getColor(scattered, depth + 1);
00063                 } else {
00064                     return Vector3D(0, 0, 0);
00065                 }
00066             } else {
00067                 Vector3D unit = ray.getDirection() / ray.getDirection().length();
00068                 double t = 0.5 * (unit.y + 1);
00069                 return (1 - t) * Vector3D(1,1,1);
00070             }
```

```
00071         };
00072
00073         void startRendering(void) {
00074             Parser *p = Parser::GetInstance("");
00075
00076             /* Sort to get the closest to the camera on top */
00077             std::sort(mPrimitives.begin(), mPrimitives.end(), compare);
00078
00079             Camera cam(Point3D(p->mCameraConfig->pos_x, p->mCameraConfig->pos_y, p->mCameraConfig->pos_z),
    -1);
00080
00081             std::cout « "P3\n" « p->mCameraConfig->width « ' ' « p->mCameraConfig->height « "\n255\n";
00082
00083             for (int y = 0; y < p->mCameraConfig->height; y++) {
00084                 for (int x = 0; x < p->mCameraConfig->width; x++) {
00085                     Vector3D col = Vector3D(0, 0, 0);
00086                     for (int i = 0; i < p->antiAliasing ; i++) {
00087                         float co_x = float(x+drand48()) / float(p->mCameraConfig->width);
00088                         float co_y = float(y+drand48()) / float(p->mCameraConfig->height);
00089                         Ray ray = cam.getRay(co_x, co_y);
00090                         col += getColor(ray, 0);
00091                     }
00092                     col /= p->antiAliasing;
00093                     int r = 255 * col.x;
00094                     int g = 255 * col.y;
00095                     int b = 255 * col.z;
00096                     std::cout « r « ' ' « g « ' ' « b « '\n';
00097                 }
00098             }
00099         };
00100
00101         std::vector<std::unique_ptr<APrimitive» mPrimitives;
00102         std::map<std::string, std::shared_ptr<AMaterial» mMaterials;
00103     protected:
00104 };
00105
00106 #endif /* !SCREEN_HPP_ */
```

6.45 CameraStruct.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** CameraStruct
00006 */
00007
00008 #ifndef CAMERASTRUCT_HPP_
00009 #define CAMERASTRUCT_HPP_
00010
00011 namespace ConfigStruct {
00012
00013 };
00014
00015 #endif /* !CAMERASTRUCT_HPP_ */
```

6.46 LightStruct.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** LightStruct
00006 */
00007
00008 #ifndef LIGHTSTRUCT_HPP_
00009 #define LIGHTSTRUCT_HPP_
00010
00011 namespace ConfigStruct {
00012     struct Light {
00013         double ambient;
00014         double diffuse;
00015         struct point {
00016             double x;
00017             double y;
00018             double z;
00019         };
00020     };
00021 };
00022
00023 #endif /* !LIGHTSTRUCT_HPP_ */
```

6.47 PrimitesStructs.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** PrimitesStructs
00006 */
00007
00008 #ifndef PRIMITESSTRUCTS_HPP_
00009 #define PRIMITESSTRUCTS_HPP_
00010
00011 #include <string>
00012
00013 namespace ConfigStruct {
00014     struct Color {
00015         int r;
00016         int g;
00017         int b;
00018     };
00019 };
00020
00021 #endif /* !PRIMITESSTRUCTS_HPP_ */
```

6.48 structs.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
00004 ** File description:
00005 ** structs
00006 */
00007
00008 #ifndef STRUCTS_HPP_
00009 #define STRUCTS_HPP_
00010
00011 #include "Math/vector3D.hpp"
00012 #include <memory>
00013
00014 class AMaterial;
00015
00016 namespace Structs {
00017     struct Color {
00018         int r;
00019         int g;
00020         int b;
00021     };
00022
00023     struct hitRecord {
00024         double t;
00025         Vector3D point;
00026         Vector3D normal;
00027         std::shared_ptr<AMaterial> material = nullptr;
00028         Structs::Color color;
00029     };
00030
00031     struct Camera {
00032         /* Resolution */
00033         int width;
00034         int height;
00035         /* Position */
00036         double pos_x;
00037         double pos_y;
00038         double pos_z;
00039         /* Rotation */
00040         double rot_x;
00041         double rot_y;
00042         double rot_z;
00043         /* Parameters */
00044         double fieldOfView;
00045     };
00046 }
00047
00048 #endif /* !STRUCTS_HPP_ */
```

6.49 Triangle.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-elias-josue.hajjar-llauquen
```

```
00004 ** File description:
00005 ** Triangle
00006 */
00007
00008 #ifndef TRIANGLE_HPP_
00009 #define TRIANGLE_HPP_
00010
00011 #include "Abstracts/APrimitive.hpp"
00012 #include "Abstracts/AMaterial.hpp"
00013 #include "Math/Vector3D.hpp"
00014 #include <memory>
00015
00016 class Triangle : public APrimitive {
00017     public:
00018         Triangle(const Vector3D &center, Structs::Color color, std::shared_ptr<AMaterial> material,
00019                 const Vector3D &v0, const Vector3D &v1, const Vector3D &v2);
00020         ~Triangle() noexcept override = default;
00021
00022         bool hit(const Ray &ray, double t_min, double t_max, Structs::hitRecord &rec) const override;
00023
00024     private:
00025         Vector3D mV0;
00026         Vector3D mV1;
00027         Vector3D mV2;
00028 };
00029
00030 #endif /* !TRIANGLE_HPP_ */
```

Index