

## Arcade architecture

Generated by Doxygen 1.10.0



# Chapter 1

## Project Detail

### Raytracer Project

This project is a modular and extensible raytracer written in C++, designed to render 3D scenes described in external configuration files. The goal is to lay the foundation for a full-featured rendering engine, supporting interface interaction, modularity, and advanced design practices.

#### Features

#### Core Requirements

```
Scene Configuration via external file (recommended: libconfig++)
Image Display during and after generation
Exit Handling during or after rendering
Basic Scene Preview using a fast renderer
Live Scene Reloading on file change
Uses interfaces for lights and primitives to support extensibility
Implements at least 2 design patterns (e.g., Factory, Builder, Decorator)
```

#### Architecture

```
Clean modular structure allowing runtime extensibility
Optional plugin system (./plugins/) for primitives, lights, and renderers
```

#### Libraries

Only authorized:

```
libconfig++ for scene parsing
SFML for image display
Standard C++ library
```

## 1.1 Compilation :

- make / make re
- make clean / make fclean
- make coding

## 1.2 Coding Style :

The Cpp code needs to abide to a specified coding style, to check if the code is compliant with the norm execute the make coding command or the ./styleChecker.sh. To understand the errors and how to fix them please refer to the coding-cpp.txt.

## 1.3 Documentation :

### 1.3.1 Docusaurus :

To start the docusaurus documentation : cd documentation/my-website npx docusaurus start

### 1.3.2 Doxygen :

The basic documentation for the project is generated using the doxygen, to run the doxygen executable, please make sure you installed the pdf-latex library. To generate the PDF : ./generateDoc.sh

## 1.4 Commit norm :

<Gitmoji> : [Element / Module] : [MESSAGE]

Gitmoji = The emoji appropriate for the current modification. [Element / Module] = The element you applied the modification. [MESSAGE] = A detail message of what you did.

Gitmojis:

```
Code feature :
- :sparkles: (): Introduce new features
- :recycle: (): Refactor / update code
- :bug: (): Fix a bug
- :poop: () : Remove Coding style or temporary fix
- :rotating_light: () : Fix Compiling Warning
- :fire: (): Remove code or files

Test feature :
- :white_check_mark: (): Add, update, or pass tests

Architecture :
- :see_no_evil: (): Add or update .gitignore files
- :construction_worker: (): Add or update CI build system
- :building_construction: () : Make Architectural changes
- :memo: () : Add or update documentation

...
```

### 1.4.1 Pull Request

- :tada: (): This Gitmoji must be used for each PR created!
- :lipstick: (): This Gitmoji must be used for each PR merged!
- :rewind: (): This Gitmoji must be used for each revert done!

## 1.5 Git-Cli :

- Changer message de commit, avant qu'il soit push :  
`git commit --amend -m "New commit message"`
- Changer le message de commit, si il a déjà été push :  
`git commit --amend -m "New commit message"`  
`git push --force`
- Un-add un fichier add par erreur qui est pas encore push:  
`git restore --staged <file>`
- Un-add un fichier qui a été commit :  
`git reset --soft HEAD~1`  
`git restore --staged fichier-a-retirer.txt`  
`git commit -m "Nouveau message de commit (sans le fichier)"`



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

Camera	??
Color	??
ConfigNode	??
ConfigParser	??
DropShadowInfo	??
std::exception	
Error	??
IException	??
AException	??
ColorException	??
CommandException	??
FactoryException	??
FileException	??
MaterialLoaderException	??
MathException	??
SceneException	??
FactoryManager	??
GraphsNodeLight	??
GraphsNodePrimitive	??
IFactory< T >	??
IFactory< Camera >	??
CameraFactory	??
IFactory< ILight >	??
LightFactory	??
IFactory< IPrimitives >	??
PrimitiveFactory	??
IGraphicMode	??
GraphicMode	??
ILight	??
ALight	??
DirectionalLight	??
PhongLight	??
PointLight	??
ILoader	??

DLoader< T > . . . . .	??
Image . . . . .	??
IMaterial . . . . .	??
AMaterial . . . . .	??
ChessboardMat . . . . .	??
FileTextureMat . . . . .	??
FlatColorMat . . . . .	??
PerlingNoiseMat . . . . .	??
TransparencyMat . . . . .	??
IMediator . . . . .	??
RayMediator . . . . .	??
InfoPixelDisplay . . . . .	??
IPrimitives . . . . .	??
APrimitives . . . . .	??
Cone . . . . .	??
Cylinder . . . . .	??
MockPrimitive . . . . .	??
Plane . . . . .	??
Sphere . . . . .	??
Torus . . . . .	??
Material . . . . .	??
ObjectConstructor . . . . .	??
ObjectErrorHandling . . . . .	??
Parser . . . . .	??
PixelInfo . . . . .	??
Math::Point3D . . . . .	??
PropertyConfig . . . . .	??
PropertyInfo . . . . .	??
Math::Random . . . . .	??
Math::Ray . . . . .	??
Ray . . . . .	??
Raytracer . . . . .	??
Rectangle3D . . . . .	??
Math::Rot3D . . . . .	??
Scene . . . . .	??
ShapeDefinition . . . . .	??
Utils . . . . .	??
ValueConverter . . . . .	??
Math::Vector2D . . . . .	??



# Chapter 3

## Class Index

### 3.1 Class List

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

<a href="#">AException</a>	??
<a href="#">ALight</a>	??
<a href="#">AMaterial</a>	??
<a href="#">APrimitives</a>	??
<a href="#">Camera</a>	??
<a href="#">CameraFactory</a>	??
<a href="#">ChessboardMat</a>	??
<a href="#">Color</a>	??
<a href="#">ColorException</a>	??
<a href="#">CommandException</a>	??
<a href="#">Cone</a>	??
<a href="#">ConfigNode</a>	??
<a href="#">ConfigParser</a>	??
<a href="#">Cylinder</a>	??
<a href="#">DirectionalLight</a>	??
<a href="#">DLLoader&lt; T &gt;</a>	??
<a href="#">DropShadowInfo</a>	??
<a href="#">Error</a>	??
<a href="#">FactoryException</a>	??
<a href="#">FactoryManager</a>	??
<a href="#">FileException</a>	??
<a href="#">FileTextureMat</a>	??
<a href="#">FlatColorMat</a>	??
<a href="#">GraphicMode</a>	??
<a href="#">GraphsNodeLight</a>	??
<a href="#">GraphsNodePrimitive</a>	??
<a href="#">IException</a>	??
<a href="#">IFactory&lt; T &gt;</a>	??
<a href="#">IGraphicMode</a>	??
<a href="#">ILight</a>	??
<a href="#">ILoader</a>	??
<a href="#">Image</a>	??
<a href="#">IMaterial</a>	??
<a href="#">IMediator</a>	??
<a href="#">InfoPixelDisplay</a>	??

<a href="#">IPrimitives</a>	??
<a href="#">LightFactory</a>	??
<a href="#">Material</a>	??
<a href="#">MaterialLoaderException</a>	??
<a href="#">MathExeption</a>	??
<a href="#">MockPrimitive</a>	??
<a href="#">ObjectConstructor</a>	??
<a href="#">ObjectErrorHandling</a>	??
<a href="#">Parser</a>	??
<a href="#">PerlingNoiseMat</a>	??
<a href="#">PhongLight</a>	??
<a href="#">PixelInfo</a>	??
<a href="#">Plane</a>	??
<a href="#">Math::Point3D</a>	??
<a href="#">PointLight</a>	??
<a href="#">PrimitiveFactory</a>	??
<a href="#">PropertyConfig</a>	??
<a href="#">PropertyInfo</a>	??
<a href="#">Math::Random</a>	??
<a href="#">Math::Ray</a>	??
<a href="#">Ray</a>	??
<a href="#">RayMediator</a>	??
<a href="#">Raytracer</a>	??
<a href="#">Rectangle3D</a>	??
<a href="#">Math::Rot3D</a>	??
<a href="#">Scene</a>	??
<a href="#">SceneException</a>	??
<a href="#">ShapeDefinition</a>	??
<a href="#">Sphere</a>	??
<a href="#">Torus</a>	??
<a href="#">TransparencyMat</a>	??
<a href="#">Utils</a>	??
<a href="#">ValueConverter</a>	??
<a href="#">Math::Vector2D</a>	??

# Chapter 4

## File Index

### 4.1 File List

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

common/ALight.hpp	??
common/APrimitives.hpp	??
common/Camera.hpp	??
common/Color.hpp	??
common/Error.hpp	??
common/Graphs.hpp	??
common/IGraphicMode.hpp	??
common/ILight.hpp	??
common/Image.hpp	??
common/IPrimitives.hpp	??
common/Material.hpp	??
common/PixelInfo.hpp	??
common/Point3D.hpp	??
common/Random.hpp	??
common/Ray.hpp	??
common/Rectangle3D.hpp	??
common/Rot3D.hpp	??
common/Scene.hpp	??
common/ValueType.hpp	??
common/Vector2D.hpp	??
common/Vector3D.hpp	??
common/Exception/AException.hpp	??
common/Exception/ColorException.hpp	??
common/Exception/CommandException.hpp	??
common/Exception/FactoryException.hpp	??
common/Exception/FileException.hpp	??
common/Exception/IException.hpp	??
common/Exception/materialLoaderException.hpp	??
common/Exception/MathException.hpp	??
common/Exception/SceneException.hpp	??
common/material/AMaterial.hpp	??
common/material/chessboardMat.hpp	??
common/material/fileTextureMat.hpp	??
common/material/flatColorMat.hpp	??
common/material/IMaterial.hpp	??

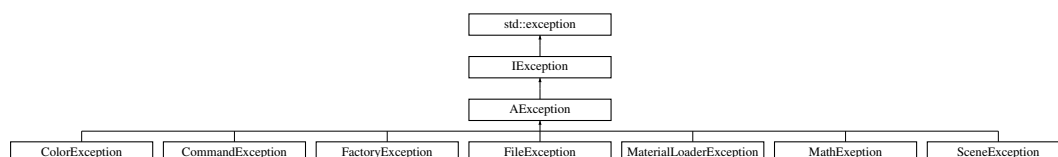
common/material/ <a href="#">perlingNoiseMat.hpp</a>	??
common/material/ <a href="#">transparencyMat.hpp</a>	??
lib/ <a href="#">DLLoader.hpp</a>	??
lib/ <a href="#">ILoader.hpp</a>	??
lib/SFML/ <a href="#">GraphicMode.hpp</a>	??
src/ <a href="#">Raytracer.hpp</a>	??
src/factory/ <a href="#">CameraFactory.hpp</a>	??
src/factory/ <a href="#">FactoryManager.hpp</a>	??
src/factory/ <a href="#">IFactory.hpp</a>	??
src/factory/ <a href="#">LightFactory.hpp</a>	??
src/factory/ <a href="#">PrimitiveFactory.hpp</a>	??
src/lights/ <a href="#">DirectionalLight.hpp</a>	??
src/lights/ <a href="#">PhongLight.hpp</a>	??
src/lights/ <a href="#">PointLight.hpp</a>	??
src/mediator/ <a href="#">IMediator.hpp</a>	??
src/mediator/ <a href="#">RayMediator.hpp</a>	??
src/parser/ <a href="#">ConfigNode.hpp</a>	??
src/parser/ <a href="#">ConfigParser.hpp</a>	??
src/parser/ <a href="#">ObjectConstructor.hpp</a>	??
src/parser/ <a href="#">ObjectErrorHandling.hpp</a>	??
src/parser/ <a href="#">Parser.hpp</a>	??
src/parser/ <a href="#">PropertyTypes.hpp</a>	??
src/parser/ <a href="#">ValueConverter.hpp</a>	??
src/primitives/ <a href="#">Cone.hpp</a>	??
src/primitives/ <a href="#">Cylinder.hpp</a>	??
src/primitives/ <a href="#">Plane.hpp</a>	??
src/primitives/ <a href="#">Sphere.hpp</a>	??
src/primitives/ <a href="#">Torus.hpp</a>	??
src/utils/ <a href="#">Utils.hpp</a>	??

## Chapter 5

# Class Documentation

### 5.1 AException Class Reference

Inheritance diagram for AException:



#### Public Member Functions

- **AException** (const std::string &type, const std::string &message)
- const char \* [what](#) () const noexcept override
- std::string [getType](#) () const noexcept override
- std::string [getMessage](#) () const noexcept override
- std::string [getFormattedMessage](#) () const noexcept override

#### Private Attributes

- std::string **\_message**
- std::string **\_type**

#### 5.1.1 Member Function Documentation

##### 5.1.1.1 getFormattedMessage()

```
std::string AException::getFormattedMessage ( ) const [inline], [override], [virtual], [noexcept]
```

Implements [IException](#).

### 5.1.1.2 getMessage()

```
std::string AException::getMessage ( ) const [inline], [override], [virtual], [noexcept]
```

Implements [IException](#).

### 5.1.1.3 getType()

```
std::string AException::getType ( ) const [inline], [override], [virtual], [noexcept]
```

Implements [IException](#).

### 5.1.1.4 what()

```
const char * AException::what ( ) const [inline], [override], [virtual], [noexcept]
```

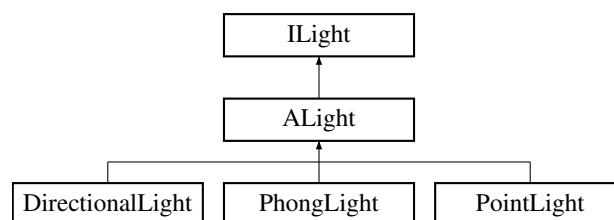
Implements [IException](#).

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

- common/Exception/AException.hpp

## 5.2 ALight Class Reference

Inheritance diagram for ALight:



### Public Member Functions

- **ALight** (const [Color](#) &color, float intensity)
- virtual void [addLight](#) ([PixelInfo](#) &pixelInfo, const [Math::Ray](#) &ray) const =0
- virtual [TypeLight](#) [getTypeLight](#) ( ) const override=0
- virtual [Math::Vector3D](#) [getDirection](#) ( ) const override=0
- virtual float [getRadius](#) ( ) const override=0
- [Color](#) [getColor](#) ( ) const override
- float [getIntensity](#) ( ) const override
- void [setColor](#) (const [Color](#) &color) override
- void [setIntensity](#) (float intensity) override

### Protected Attributes

- [Color](#) \_color
- float \_intensity

## 5.2.1 Member Function Documentation

### 5.2.1.1 addLight()

```
virtual void ALight::addLight (
    PixelInfo & pixelInfo,
    const Math::Ray & ray ) const [pure virtual]
```

Implements [ILight](#).

### 5.2.1.2 getColor()

```
Color ALight::getColor ( ) const [override], [virtual]
```

Implements [ILight](#).

### 5.2.1.3 getDirection()

```
virtual Math::Vector3D ALight::getDirection ( ) const [override], [pure virtual]
```

Implements [ILight](#).

### 5.2.1.4 getIntensity()

```
float ALight::getIntensity ( ) const [override], [virtual]
```

Implements [ILight](#).

### 5.2.1.5 getRadius()

```
virtual float ALight::getRadius ( ) const [override], [pure virtual]
```

Implements [ILight](#).

### 5.2.1.6 getTypeLight()

```
virtual TypeLight ALight::getTypeLight ( ) const [override], [pure virtual]
```

Implements [ILight](#).

### 5.2.1.7 setColor()

```
void ALight::setColor (
    const Color & color ) [override], [virtual]
```

Implements [ILight](#).

### 5.2.1.8 setIntensity()

```
void ALight::setIntensity (
    float intensity ) [override], [virtual]
```

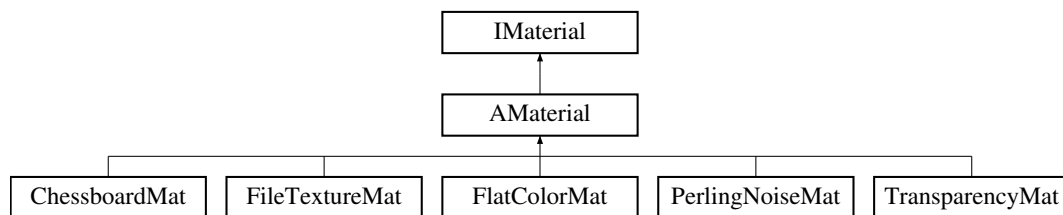
Implements [ILight](#).

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

- common/ALight.hpp
- common/ALight.cpp

## 5.3 AMaterial Class Reference

Inheritance diagram for AMaterial:



### Public Member Functions

- void [setAmbient](#) (const Math::Vector3D &a) override
- void [setDiffuse](#) (const Math::Vector3D &d) override
- void [setSpecular](#) (const Math::Vector3D &s) override
- void [setShininess](#) (float s) override
- void [setReflectivity](#) (float r) override
- void [setTransparency](#) (float t) override
- void [setRefractiveIndex](#) (float i) override
- void [setOpacity](#) (float o) override
- void [setColorTexture](#) (const std::shared\_ptr< std::string > &texture) override
- void [setNormalMap](#) (const std::shared\_ptr< std::string > &map) override
- void [setOptionalColor1](#) (const Color &color) override
- void [setOptionalColor2](#) (const Color &color) override
- void [setScale](#) (float s) override
- void [setMaterialType](#) (MaterialType type) override
- Math::Vector3D [getAmbient](#) () const override
- Math::Vector3D [getDiffuse](#) () const override
- Math::Vector3D [getSpecular](#) () const override



- float [getShininess](#) () const override
- float [getReflectivity](#) () const override
- float [getTransparency](#) () const override
- float [getRefractiveIndex](#) () const override
- float [getOpacity](#) () const override
- std::shared\_ptr< std::string > [getColorTexture](#) () const override
- std::shared\_ptr< std::string > [getNormalMap](#) () const override
- [Color](#) [getOptionalColor1](#) () const override
- [Color](#) [getOptionalColor2](#) () const override
- float [getScale](#) () const override
- MaterialType [getMaterialType](#) () const override
- virtual [Color](#) [applyMaterial](#) (const [PixelInfo](#) &pixelInfo, float radius, float height, const [IPrimitives](#) &primitive) const =0

### Public Attributes

- Math::Vector3D **ambient**
- Math::Vector3D **diffuse**
- Math::Vector3D **specular**
- float **shininess**
- [Color](#) **OptionalColor1** = [Color](#)(0.0f, 0.0f, 0.0f)
- [Color](#) **OptionalColor2** = [Color](#)(0.0f, 0.0f, 0.0f)
- float **scale** = 5.0f
- float **reflectivity**
- float **transparency**
- float **refractiveIndex**
- float **opacity**
- std::shared\_ptr< std::string > **colorTexture**
- std::shared\_ptr< std::string > **normalMap**
- MaterialType **materialType** = MaterialType::FLAT\_COLOR

## 5.3.1 Member Function Documentation

### 5.3.1.1 applyMaterial()

```
virtual Color AMaterial::applyMaterial (
    const PixelInfo & pixelInfo,
    float radius,
    float height,
    const IPrimitives & primitive ) const [pure virtual]
```

Implements [IMaterial](#).

### 5.3.1.2 getAmbient()

```
Math::Vector3D AMaterial::getAmbient ( ) const [inline], [override], [virtual]
```

Implements [IMaterial](#).

### 5.3.1.3 getColorTexture()

```
std::shared_ptr< std::string > AMaterial::getColorTexture ( ) const [inline], [override], [virtual]
```

Implements [IMaterial](#).

### 5.3.1.4 getDiffuse()

```
Math::Vector3D AMaterial::getDiffuse ( ) const [inline], [override], [virtual]
```

Implements [IMaterial](#).

### 5.3.1.5 getMaterialType()

```
MaterialType AMaterial::getMaterialType ( ) const [inline], [override], [virtual]
```

Implements [IMaterial](#).

### 5.3.1.6 getNormalMap()

```
std::shared_ptr< std::string > AMaterial::getNormalMap ( ) const [inline], [override], [virtual]
```

Implements [IMaterial](#).

### 5.3.1.7 getOpacity()

```
float AMaterial::getOpacity ( ) const [inline], [override], [virtual]
```

Implements [IMaterial](#).

### 5.3.1.8 getOptionalColor1()

```
Color AMaterial::getOptionalColor1 ( ) const [inline], [override], [virtual]
```

Implements [IMaterial](#).

### 5.3.1.9 getOptionalColor2()

```
Color AMaterial::getOptionalColor2 ( ) const [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.10 getReflectivity()

```
float AMaterial::getReflectivity ( ) const [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.11 getRefractiveIndex()

```
float AMaterial::getRefractiveIndex ( ) const [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.12 getScale()

```
float AMaterial::getScale ( ) const [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.13 getShininess()

```
float AMaterial::getShininess ( ) const [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.14 getSpecular()

```
Math::Vector3D AMaterial::getSpecular ( ) const [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.15 getTransparency()

```
float AMaterial::getTransparency ( ) const [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.16 setAmbient()

```
void AMaterial::setAmbient (
    const Math::Vector3D & a ) [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.17 setColorTexture()

```
void AMaterial::setColorTexture (
    const std::shared_ptr< std::string > & texture ) [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.18 setDiffuse()

```
void AMaterial::setDiffuse (
    const Math::Vector3D & d ) [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.19 setMaterialType()

```
void AMaterial::setMaterialType (
    MaterialType type ) [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.20 setNormalMap()

```
void AMaterial::setNormalMap (
    const std::shared_ptr< std::string > & map ) [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.21 setOpacity()

```
void AMaterial::setOpacity (
    float o ) [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.22 setOptionalColor1()

```
void AMaterial::setOptionalColor1 (
    const Color & color ) [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.23 setOptionalColor2()

```
void AMaterial::setOptionalColor2 (
    const Color & color ) [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.24 setReflectivity()

```
void AMaterial::setReflectivity (
    float r ) [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.25 setRefractiveIndex()

```
void AMaterial::setRefractiveIndex (
    float i ) [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.26 setScale()

```
void AMaterial::setScale (
    float s ) [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.27 setShininess()

```
void AMaterial::setShininess (
    float s ) [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.28 setSpecular()

```
void AMaterial::setSpecular (
    const Math::Vector3D & s ) [inline], [override], [virtual]
```

Implements [IMaterial](#).

#### 5.3.1.29 setTransparency()

```
void AMaterial::setTransparency (
    float t ) [inline], [override], [virtual]
```

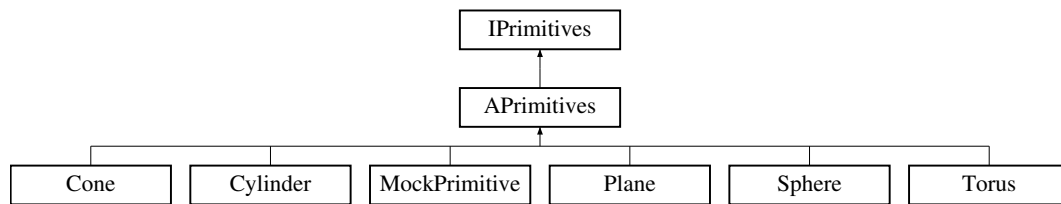
Implements [IMaterial](#).

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

- common/material/AMaterial.hpp
- common/material/AMaterial.cpp

## 5.4 APrimitives Class Reference

Inheritance diagram for APrimitives:



### Public Member Functions

- **APrimitives** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList, std::vector< std::shared\_ptr< [ILight](#) > > light)
- virtual [PixellInfo](#) **distanceInfo** (const [Math::Ray](#) &ray)=0
- virtual std::optional< double > **distance** (const [Math::Ray](#) &ray) const =0
- virtual Type **getType** () const override=0
- [Math::Ray](#) **computeScaledRay** (const [Math::Ray](#) &ray) const override
- [Math::Point3D](#) **getPosition** () const override
- [Math::Rot3D](#) **getRotation** () const override
- float **getScale** () const override
- std::shared\_ptr< [IMaterial](#) > **getMaterial** () const override
- [Math::Vector3D](#) **getInvScales** () const
- const std::vector< std::shared\_ptr< [ILight](#) > > & **getLights** () const
- void **setPosition** (const [Math::Point3D](#) &newPosition) override
- void **setRotation** (const [Math::Rot3D](#) &newRotation) override
- void **setScale** (const float newScale) override
- void **setMaterial** (std::shared\_ptr< [IMaterial](#) > newMaterial) override
- void **addLight** (std::shared\_ptr< [ILight](#) > light)
- void **clearLights** ()
- void **applyLights** ([PixellInfo](#) &pixellInfo, const [Math::Ray](#) &ray) const
- void **getPos** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getRot** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getCol** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getScales** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getGraph** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList)
- void **computeInvScales** ()
- void **getGraphScale** (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)
- void **getPosGraph** (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)
- void **getRotGraph** (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)

### Public Member Functions inherited from [IPrimitives](#)

- virtual std::optional< [Math::Point3D](#) > **getIntersection** (const [Math::Ray](#) &ray) const =0
- virtual std::optional< [Math::Vector3D](#) > **getNormal** (const [Math::Point3D](#) &point) const =0

**Protected Attributes**

- [Math::Point3D](#) **\_position**
- [Math::Rot3D](#) **\_rotation**
- float **scale**
- std::shared\_ptr< [IMaterial](#) > **material**
- [Color](#) **\_color**
- [Math::Vector3D](#) **\_scales**
- [Math::Vector3D](#) **\_invScales**
- std::string **\_graph**
- std::vector< std::shared\_ptr< [ILight](#) > > **\_lights**

**5.4.1 Member Function Documentation****5.4.1.1 computeScaledRay()**

```
Math::Ray APrimitives::computeScaledRay (
    const Math::Ray & ray ) const [override], [virtual]
```

Implements [IPrimitives](#).

**5.4.1.2 distance()**

```
virtual std::optional< double > APrimitives::distance (
    const Math::Ray & ray ) const [pure virtual]
```

Implements [IPrimitives](#).

**5.4.1.3 distanceInfo()**

```
virtual PixelInfo APrimitives::distanceInfo (
    const Math::Ray & ray ) [pure virtual]
```

Implements [IPrimitives](#).

**5.4.1.4 getMaterial()**

```
std::shared_ptr< IMaterial > APrimitives::getMaterial ( ) const [inline], [override], [virtual]
```

Implements [IPrimitives](#).

**5.4.1.5 getPosition()**

```
Math::Point3D APrimitives::getPosition ( ) const [inline], [override], [virtual]
```

Implements [IPrimitives](#).

#### 5.4.1.6 getRotation()

```
Math::Rot3D APrimitives::getRotation ( ) const [inline], [override], [virtual]
```

Implements [IPrimitives](#).

#### 5.4.1.7 getScale()

```
float APrimitives::getScale ( ) const [inline], [override], [virtual]
```

Implements [IPrimitives](#).

#### 5.4.1.8 getType()

```
virtual Type APrimitives::getType ( ) const [override], [pure virtual]
```

Implements [IPrimitives](#).

#### 5.4.1.9 setMaterial()

```
void APrimitives::setMaterial (
    std::shared_ptr< IMaterial > newMaterial ) [inline], [override], [virtual]
```

Implements [IPrimitives](#).

#### 5.4.1.10 setPosition()

```
void APrimitives::setPosition (
    const Math::Point3D & newPosition ) [inline], [override], [virtual]
```

Implements [IPrimitives](#).

#### 5.4.1.11 setRotation()

```
void APrimitives::setRotation (
    const Math::Rot3D & newRotation ) [inline], [override], [virtual]
```

Implements [IPrimitives](#).

#### 5.4.1.12 setScale()

```
void APrimitives::setScale (
    const float newScale ) [inline], [override], [virtual]
```

Implements [IPrimitives](#).

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

- common/APrimitives.hpp
- common/APrimitives.cpp



## 5.5 Camera Class Reference

### Public Member Functions

- **Camera** (const [Camera](#) &)=default
- **Camera** (const [Rectangle3D](#) &screen)
- **Camera** & **operator=** (const [Camera](#) &)=default
- void **updateScreen** ()
- [Math::Ray](#) **ray** (double u, double v) const
- [Math::Point3D](#) **getOrigin** () const
- [Math::Point3D](#) **getRotation** () const
- [Rectangle3D](#) **getScreen** () const
- [Math::Vector3D](#) **getPosition** () const
- int **getWidth** () const
- int **getHeight** () const
- float **getFieldOfView** () const
- void **setRotation** ([Math::Point3D](#) rotation)
- void **setPosition** ([Math::Vector3D](#) position)
- void **setResolution** (int x, int y)
- void **setHeight** (int h)
- void **setFieldOfView** (float fov)

### Public Attributes

- [Math::Point3D](#) **\_origin**
- [Rectangle3D](#) **\_screen**

### Protected Attributes

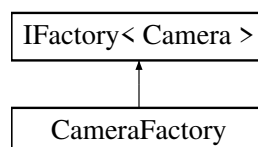
- int **width**
- int **height**
- float **fieldOfView**
- [Math::Vector3D](#) **\_rotation**
- [Math::Vector3D](#) **\_position**

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

- common/Camera.hpp
- common/Camera.cpp

## 5.6 CameraFactory Class Reference

Inheritance diagram for CameraFactory:



## Public Member Functions

- `std::shared_ptr< Camera > create` (const std::string &type, std::shared\_ptr< std::map< ValueType\_  
\_t, ValueType > > config, const std::vector< std::shared\_ptr< std::map< ValueType\_  
\_t, ValueType > > &graphSceneList, const std::vector< std::shared\_ptr< [ILight](#) > > &lights) override
- `std::shared_ptr< Camera > createSimple` (const std::string &type, std::shared\_ptr< std::map< ValueType\_  
\_t, ValueType > > config, const std::vector< std::shared\_ptr< std::map< ValueType\_  
\_t, ValueType > > &graphSceneList) override
- `void registerCreator` (const std::string &type, std::function< std::shared\_ptr< [Camera](#) >(std::shared\_ptr<  
std::map< ValueType\_  
\_t, ValueType > >, const std::vector< std::shared\_ptr< std::map< ValueType\_  
\_t, ValueType > > &)> creator) override
- `void registerCreatorLight` (const std::string &type, std::function< std::shared\_ptr< [Camera](#) >(std::shared\_  
ptr< std::map< ValueType\_  
\_t, ValueType > >, const std::vector< std::shared\_ptr< std::map< ValueType\_  
\_t, ValueType > > &, const std::vector< std::shared\_ptr< [ILight](#) > > &)> creator) override
- `bool loadPlugin` (const std::string &path) override
- `void loadAllPlugins` (const std::string &directory="plugins/")
- `ObjectType getTypeFromPlugin` (const std::string &path, [DLLoader](#)< void \* > loader)
- `std::string getNameFromPlugin` (const std::string &path, [DLLoader](#)< void \* > loader)

## Private Attributes

- `std::map< std::string, std::function< std::shared_ptr< Camera >(std::shared_ptr< std::map< ValueType_  
_t, ValueType > >, const std::vector< std::shared_ptr< std::map< ValueType_  
_t, ValueType > > &)> _creators )`
- `std::vector< DLLoader< void * > > _dlLoaders`

## 5.6.1 Member Function Documentation

### 5.6.1.1 create()

```
std::shared_ptr< Camera > CameraFactory::create (
    const std::string & type,
    std::shared_ptr< std::map< ValueType_t, ValueType > > config,
    const std::vector< std::shared_ptr< std::map< ValueType_t, ValueType > > > &
graphSceneList,
    const std::vector< std::shared_ptr< ILight > > & lights ) [override], [virtual]
```

Implements [IFactory< Camera >](#).

### 5.6.1.2 createSimple()

```
std::shared_ptr< Camera > CameraFactory::createSimple (
    const std::string & type,
    std::shared_ptr< std::map< ValueType_t, ValueType > > config,
    const std::vector< std::shared_ptr< std::map< ValueType_t, ValueType > > > &
graphSceneList ) [override], [virtual]
```

Implements [IFactory< Camera >](#).

### 5.6.1.3 loadPlugin()

```
bool CameraFactory::loadPlugin (
    const std::string & path ) [override], [virtual]
```

Implements [IFactory< Camera >](#).

### 5.6.1.4 registerCreator()

```
void CameraFactory::registerCreator (
    const std::string & type,
    std::function< std::shared_ptr< Camera >(std::shared_ptr< std::map< ValueType←
_t, ValueType > >, const std::vector< std::shared_ptr< std::map< ValueType_t, ValueType > >
> &)> creator ) [override], [virtual]
```

Implements [IFactory< Camera >](#).

### 5.6.1.5 registerCreatorLight()

```
void CameraFactory::registerCreatorLight (
    const std::string & type,
    std::function< std::shared_ptr< Camera >(std::shared_ptr< std::map< ValueType←
_t, ValueType > >, const std::vector< std::shared_ptr< std::map< ValueType_t, ValueType > >
> &, const std::vector< std::shared_ptr< ILight > > &)> creator ) [override], [virtual]
```

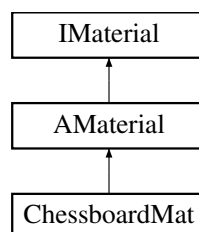
Implements [IFactory< Camera >](#).

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

- `src/factory/CameraFactory.hpp`
- `src/factory/CameraFactory.cpp`

## 5.7 ChessboardMat Class Reference

Inheritance diagram for ChessboardMat:



### Public Member Functions

- [Color applyMaterial](#) (const [PixelInfo](#) &pixelInfo, float radius, float height, const [IPrimitives](#) &primitive) const override

## Public Member Functions inherited from **AMaterial**

- void [setAmbient](#) (const Math::Vector3D &a) override
- void [setDiffuse](#) (const Math::Vector3D &d) override
- void [setSpecular](#) (const Math::Vector3D &s) override
- void [setShininess](#) (float s) override
- void [setReflectivity](#) (float r) override
- void [setTransparency](#) (float t) override
- void [setRefractiveIndex](#) (float i) override
- void [setOpacity](#) (float o) override
- void [setColorTexture](#) (const std::shared\_ptr< std::string > &texture) override
- void [setNormalMap](#) (const std::shared\_ptr< std::string > &map) override
- void [setOptionalColor1](#) (const [Color](#) &color) override
- void [setOptionalColor2](#) (const [Color](#) &color) override
- void [setScale](#) (float s) override
- void [setMaterialType](#) (MaterialType type) override
- Math::Vector3D [getAmbient](#) () const override
- Math::Vector3D [getDiffuse](#) () const override
- Math::Vector3D [getSpecular](#) () const override
- float [getShininess](#) () const override
- float [getReflectivity](#) () const override
- float [getTransparency](#) () const override
- float [getRefractiveIndex](#) () const override
- float [getOpacity](#) () const override
- std::shared\_ptr< std::string > [getColorTexture](#) () const override
- std::shared\_ptr< std::string > [getNormalMap](#) () const override
- [Color](#) [getOptionalColor1](#) () const override
- [Color](#) [getOptionalColor2](#) () const override
- float [getScale](#) () const override
- MaterialType [getMaterialType](#) () const override

## Private Member Functions

- [Color](#) [applySphereChessboard](#) (const [PixellInfo](#) &pixellInfo, float radius, float height, const [IPrimitives](#) &primitive) const
- [Color](#) [applyPlaneChessboard](#) (const [PixellInfo](#) &pixellInfo, float radius, float height, const [IPrimitives](#) &primitive) const
- [Color](#) [applyCylinderChessboard](#) (const [PixellInfo](#) &pixellInfo, float radius, float height, const [IPrimitives](#) &primitive) const
- [Color](#) [applyConeChessboard](#) (const [PixellInfo](#) &pixellInfo, float radius, float height, const [IPrimitives](#) &primitive) const

## Additional Inherited Members

## Public Attributes inherited from **AMaterial**

- Math::Vector3D **ambient**
- Math::Vector3D **diffuse**
- Math::Vector3D **specular**
- float **shininess**
- [Color](#) **OptionalColor1** = [Color](#)(0.0f, 0.0f, 0.0f)
- [Color](#) **OptionalColor2** = [Color](#)(0.0f, 0.0f, 0.0f)
- float **scale** = 5.0f

- float **reflectivity**
- float **transparency**
- float **refractiveIndex**
- float **opacity**
- std::shared\_ptr< std::string > **colorTexture**
- std::shared\_ptr< std::string > **normalMap**
- MaterialType **materialType** = MaterialType::FLAT\_COLOR

## 5.7.1 Member Function Documentation

### 5.7.1.1 applyMaterial()

```
Color ChessboardMat::applyMaterial (
    const PixelInfo & pixelInfo,
    float radius,
    float height,
    const IPrimitives & primitive ) const [override], [virtual]
```

Implements [AMaterial](#).

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

- common/material/chessboardMat.hpp
- common/material/chessboardMat.cpp

## 5.8 Color Class Reference

### Public Member Functions

- **Color** (uint8\_t red, uint8\_t green, uint8\_t blue)
- **Color** (Math::Vector3D vec)
- **Color** (uint8\_t red, uint8\_t green, uint8\_t blue, uint8\_t transparency)
- **Color & operator=** (const Math::Vector3D &vec)
- **Color operator\*** (float scalar) const
- **Color operator\*** (const **Color** &other) const
- **Color operator\*=** (float scalar)
- **Color operator\*=** (const **Color** &other)
- **Color operator+** (const **Color** &other) const
- **Color operator+=** (const **Color** &other)
- **Color operator-** (const **Color** &other) const
- **Color operator-=** (const **Color** &other)
- void **setTransparency** (float transparencyValue)
- uint8\_t **getRed** () const
- uint8\_t **getGreen** () const
- uint8\_t **getBlue** () const
- uint8\_t **getTransparency** () const
- void **setRed** (uint8\_t red)
- void **setGreen** (uint8\_t green)
- void **setBlue** (uint8\_t blue)

### Private Attributes

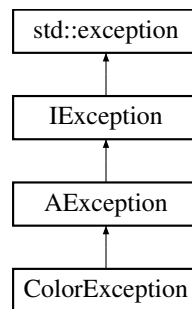
- `uint8_t _red`
- `uint8_t _green`
- `uint8_t _blue`
- `uint8_t _transparency`

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

- `common/Color.hpp`

## 5.9 ColorException Class Reference

Inheritance diagram for ColorException:



### Public Member Functions

- **ColorException** (const std::string &message)

### Public Member Functions inherited from **AException**

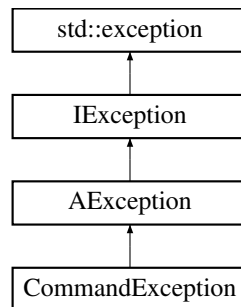
- **AException** (const std::string &type, const std::string &message)
- const char \* **what** () const noexcept override
- std::string **getType** () const noexcept override
- std::string **getMessage** () const noexcept override
- std::string **getFormattedMessage** () const noexcept override

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

- `common/Exception/ColorException.hpp`

## 5.10 CommandException Class Reference

Inheritance diagram for CommandException:



### Public Member Functions

- **CommandException** (const std::string &message)

### Public Member Functions inherited from [AException](#)

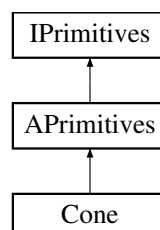
- **AException** (const std::string &type, const std::string &message)
- const char \* [what](#) () const noexcept override
- std::string [getType](#) () const noexcept override
- std::string [getMessage](#) () const noexcept override
- std::string [getFormattedMessage](#) () const noexcept override

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

- common/Exception/CommandException.hpp

## 5.11 Cone Class Reference

Inheritance diagram for Cone:



## Public Member Functions

- **Cone** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList, const std::vector< std::shared\_ptr< [ILight](#) > > &lights)
- void **setBaseRadius** (float radius)
- void **setHeight** (float height)
- std::optional< double > **distance** (const [Math::Ray](#) &ray) const override
- [PixellInfo](#) **distanceInfo** (const [Math::Ray](#) &ray) override
- std::optional< [Math::Point3D](#) > **getIntersection** (const [Math::Ray](#) &ray) const override
- std::optional< [Math::Vector3D](#) > **getNormal** (const [Math::Point3D](#) &point) const override
- float **getBaseRadius** () const
- Type **getType** () const override
- float **getHeight** () const

## Public Member Functions inherited from [APrimitives](#)

- **APrimitives** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList, std::vector< std::shared\_ptr< [ILight](#) > > light)
- [Math::Ray](#) **computeScaledRay** (const [Math::Ray](#) &ray) const override
- [Math::Point3D](#) **getPosition** () const override
- [Math::Rot3D](#) **getRotation** () const override
- float **getScale** () const override
- std::shared\_ptr< [IMaterial](#) > **getMaterial** () const override
- [Math::Vector3D](#) **getInvScales** () const
- const std::vector< std::shared\_ptr< [ILight](#) > > & **getLights** () const
- void **setPosition** (const [Math::Point3D](#) &newPosition) override
- void **setRotation** (const [Math::Rot3D](#) &newRotation) override
- void **setScale** (const float newScale) override
- void **setMaterial** (std::shared\_ptr< [IMaterial](#) > newMaterial) override
- void **addLight** (std::shared\_ptr< [ILight](#) > light)
- void **clearLights** ()
- void **applyLights** ([PixellInfo](#) &pixellInfo, const [Math::Ray](#) &ray) const
- void **getPos** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getRot** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getCol** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getScales** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getGraph** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList)
- void **computeInvScales** ()
- void **getGraphScale** (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)
- void **getPosGraph** (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)
- void **getRotGraph** (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)

## Private Member Functions

- [Math::Vector3D](#) **transformToLocal** (const [Math::Ray](#) &ray) const
- [Math::Vector3D](#) **localDirectionVector** (const [Math::Ray](#) &ray) const
- std::optional< double > **intersectConeBody** (const [Math::Vector3D](#) &localOrigin, const [Math::Vector3D](#) &localDirection) const
- std::optional< double > **intersectConeBase** (const [Math::Vector3D](#) &localOrigin, const [Math::Vector3D](#) &localDirection) const
- bool **isPointOnConeBody** (const [Math::Point3D](#) &hitPoint) const



**Private Attributes**

- float **\_baseRadius**
- float **\_height**
- double **\_distance**

**Additional Inherited Members****Protected Attributes inherited from [APrimitives](#)**

- [Math::Point3D](#) **\_position**
- [Math::Rot3D](#) **\_rotation**
- float **scale**
- std::shared\_ptr< [IMaterial](#) > **material**
- [Color](#) **\_color**
- [Math::Vector3D](#) **\_scales**
- [Math::Vector3D](#) **\_invScales**
- std::string **\_graph**
- std::vector< std::shared\_ptr< [ILight](#) > > **\_lights**

**5.11.1 Member Function Documentation****5.11.1.1 distance()**

```
std::optional< double > Cone::distance (
    const Math::Ray & ray ) const [override], [virtual]
```

Implements [APrimitives](#).

**5.11.1.2 distanceInfo()**

```
PixelInfo Cone::distanceInfo (
    const Math::Ray & ray ) [override], [virtual]
```

Implements [APrimitives](#).

**5.11.1.3 getIntersection()**

```
std::optional< Math::Point3D > Cone::getIntersection (
    const Math::Ray & ray ) const [override], [virtual]
```

Implements [IPrimitives](#).

**5.11.1.4 getNormal()**

```
std::optional< Math::Vector3D > Cone::getNormal (
    const Math::Point3D & point ) const [override], [virtual]
```

Implements [IPrimitives](#).

### 5.11.1.5 getType()

```
Type Cone::getType ( ) const [override], [virtual]
```

Implements [APrimitives](#).

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

- src/primitives/Cone.hpp
- src/primitives/Cone.cpp

## 5.12 ConfigNode Class Reference

### Public Member Functions

- bool **hasChild** (const std::string &name) const

### Public Attributes

- std::map< std::string, [ConfigNode](#) > **children**
- ValueType **value**
- bool **isValue**
- NodeType **type**
- std::string **\_name**

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

- src/parser/ConfigNode.hpp
- src/parser/ConfigNode.cpp

## 5.13 ConfigParser Class Reference

### Public Member Functions

- bool **loadConfig** (const std::string &filename, [ConfigNode](#) &rootNode)

### Protected Member Functions

- void **buildConfigTree** (const Setting &setting, std::shared\_ptr< [ConfigNode](#) > node)

### Private Member Functions

- void **handleGroupType** (const Setting &child, const std::string &childName, std::shared\_ptr< [ConfigNode](#) > node)
- void **handleArrayType** (const Setting &child, const std::string &childName, std::shared\_ptr< [ConfigNode](#) > node)
- void **handleListType** (const Setting &child, const std::string &childName, std::shared\_ptr< [ConfigNode](#) > node)
- void **handleValueType** (const Setting &child, const std::string &childName, std::shared\_ptr< [ConfigNode](#) > node)

**Private Attributes**

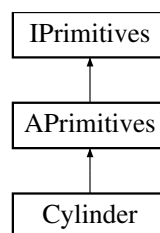
- Config file

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

- src/parser/ConfigParser.hpp
- src/parser/ConfigParser.cpp

## 5.14 Cylinder Class Reference

Inheritance diagram for Cylinder:

**Public Member Functions**

- **Cylinder** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList, const std::vector< std::shared\_ptr< [ILight](#) > > &lights)
- void **setBaseRadius** (float radius)
- void **setHeight** (float height)
- float **getBaseRadius** () const
- float **getHeight** () const
- Type **getType** () const override
- std::optional< double > **distance** (const [Math::Ray](#) &ray) const override
- [PixelInfo](#) **distanceInfo** (const [Math::Ray](#) &ray) override
- std::optional< [Math::Point3D](#) > **getIntersection** (const [Math::Ray](#) &ray) const override
- std::optional< [Math::Vector3D](#) > **getNormal** (const [Math::Point3D](#) &point) const override

**Public Member Functions inherited from [APrimitives](#)**

- **APrimitives** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList, std::vector< std::shared\_ptr< [ILight](#) > > light)
- [Math::Ray](#) **computeScaledRay** (const [Math::Ray](#) &ray) const override
- [Math::Point3D](#) **getPosition** () const override
- [Math::Rot3D](#) **getRotation** () const override
- float **getScale** () const override
- std::shared\_ptr< [IMaterial](#) > **getMaterial** () const override
- [Math::Vector3D](#) **getInvScales** () const
- const std::vector< std::shared\_ptr< [ILight](#) > > & **getLights** () const
- void **setPosition** (const [Math::Point3D](#) &newPosition) override

- void **setRotation** (const [Math::Rot3D](#) &newRotation) override
- void **setScale** (const float newScale) override
- void **setMaterial** (std::shared\_ptr< [IMaterial](#) > newMaterial) override
- void **addLight** (std::shared\_ptr< [ILight](#) > light)
- void **clearLights** ()
- void **applyLights** ([PixelInfo](#) &pixelInfo, const [Math::Ray](#) &ray) const
- void **getPos** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getRot** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getCol** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getScales** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getGraph** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList)
- void **computeInvScales** ()
- void **getGraphScale** (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)
- void **getPosGraph** (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)
- void **getRotGraph** (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)

### Private Attributes

- float **\_baseRadius**
- float **\_height**
- double **\_distance**

### Additional Inherited Members

### Protected Attributes inherited from [APrimitives](#)

- [Math::Point3D](#) **\_position**
- [Math::Rot3D](#) **\_rotation**
- float **scale**
- std::shared\_ptr< [IMaterial](#) > **material**
- [Color](#) **\_color**
- [Math::Vector3D](#) **\_scales**
- [Math::Vector3D](#) **\_invScales**
- std::string **\_graph**
- std::vector< std::shared\_ptr< [ILight](#) > > **\_lights**

## 5.14.1 Member Function Documentation

### 5.14.1.1 distance()

```
std::optional< double > Cylinder::distance (
    const Math::Ray & ray ) const [override], [virtual]
```

Implements [APrimitives](#).

### 5.14.1.2 distanceInfo()

```
PixelInfo Cylinder::distanceInfo (
    const Math::Ray & ray ) [override], [virtual]
```

Implements [APrimitives](#).

#### 5.14.1.3 getIntersection()

```
std::optional< Math::Point3D > Cylinder::getIntersection (
    const Math::Ray & ray ) const [override], [virtual]
```

Implements [IPrimitives](#).

#### 5.14.1.4 getNormal()

```
std::optional< Math::Vector3D > Cylinder::getNormal (
    const Math::Point3D & point ) const [override], [virtual]
```

Implements [IPrimitives](#).

#### 5.14.1.5 getType()

```
Type Cylinder::getType ( ) const [override], [virtual]
```

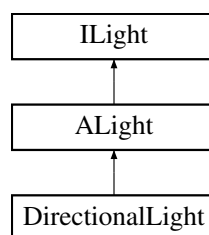
Implements [APrimitives](#).

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

- src/primitives/Cylinder.hpp
- src/primitives/Cylinder.cpp

## 5.15 DirectionalLight Class Reference

Inheritance diagram for DirectionalLight:



### Public Member Functions

- **DirectionalLight** (const [Color](#) &color, float intensity, const Math::Vector3D &direction, float radius)
- **DirectionalLight** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void [addLight](#) ([PixelInfo](#) &pixelInfo, const [Math::Ray](#) &ray) const override
- TypeLight [getTypeLight](#) () const override
- Math::Vector3D [getDirection](#) () const override
- float [getRadius](#) () const override
- void **setDirection** (const Math::Vector3D &direction)

## Public Member Functions inherited from [ALight](#)

- **ALight** (const [Color](#) &color, float intensity)
- [Color](#) **getColor** () const override
- float **getIntensity** () const override
- void **setColor** (const [Color](#) &color) override
- void **setIntensity** (float intensity) override

## Private Attributes

- [Math::Vector3D](#) **\_direction**
- [Color](#) **\_color**
- float **\_intensity**
- float **\_radius**

## Additional Inherited Members

## Protected Attributes inherited from [ALight](#)

- [Color](#) **\_color**
- float **\_intensity**

## 5.15.1 Member Function Documentation

### 5.15.1.1 addLight()

```
void DirectionalLight::addLight (
    PixelInfo & pixelInfo,
    const Math::Ray & ray ) const [override], [virtual]
```

Implements [ALight](#).

### 5.15.1.2 getDirection()

```
Math::Vector3D DirectionalLight::getDirection ( ) const [override], [virtual]
```

Implements [ALight](#).

### 5.15.1.3 getRadius()

```
float DirectionalLight::getRadius ( ) const [override], [virtual]
```

Implements [ALight](#).

#### 5.15.1.4 getTypeLight()

```
TypeLight DirectionalLight::getTypeLight ( ) const [override], [virtual]
```

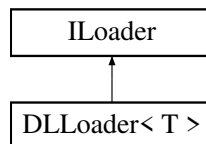
Implements [ALight](#).

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

- src/lights/DirectionalLight.hpp
- src/lights/DirectionalLight.cpp

## 5.16 DLoader< T > Class Template Reference

Inheritance diagram for DLoader< T >:



### Public Member Functions

- void \* [getHandler](#) () const override
- void \* [Open](#) (const char \*path, int flag) override
- void \* [Symbol](#) (const char \*symbolName) override
- T [getSymbol](#) (const char \*symbolName)
- int [Close](#) () override
- const char \* [Error](#) () override

### Private Attributes

- void \* [\\_handler](#) = nullptr

### 5.16.1 Member Function Documentation

#### 5.16.1.1 Close()

```
template<typename T >
int DLoader< T >::Close ( ) [inline], [override], [virtual]
```

Implements [ILoader](#).

#### 5.16.1.2 Error()

```
template<typename T >
const char * DLoader< T >::Error ( ) [inline], [override], [virtual]
```

Implements [ILoader](#).

### 5.16.1.3 getHandler()

```
template<typename T >
void * DLLoader< T >::getHandler ( ) const [inline], [override], [virtual]
```

Implements [ILoader](#).

### 5.16.1.4 Open()

```
template<typename T >
void * DLLoader< T >::Open (
    const char * path,
    int flag ) [inline], [override], [virtual]
```

Implements [ILoader](#).

### 5.16.1.5 Symbol()

```
template<typename T >
void * DLLoader< T >::Symbol (
    const char * symbolName ) [inline], [override], [virtual]
```

Implements [ILoader](#).

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

- lib/DLLoader.hpp

## 5.17 DropShadowInfo Struct Reference

### Public Attributes

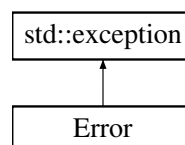
- [Math::Vector3D](#) **position**
- float **darkness**

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

- src/Raytracer.hpp

## 5.18 Error Class Reference

Inheritance diagram for Error:





**Public Member Functions**

- **Error** (const std::string &msg, const std::string &file, int line)
- const char \* **what** () const noexcept override
- const char \* **where** () const noexcept
- int **line** () const noexcept
- void **ErrorFailureException** () const
- void **exitCode** (int code)

**Private Attributes**

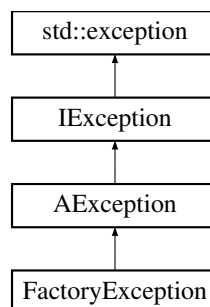
- std::string **\_message**
- std::string **\_file**
- int **\_line**

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

- common/Error.hpp

## 5.19 FactoryException Class Reference

Inheritance diagram for FactoryException:

**Public Member Functions**

- **FactoryException** (const std::string &message)

**Public Member Functions inherited from AException**

- **AException** (const std::string &type, const std::string &message)
- const char \* **what** () const noexcept override
- std::string **getType** () const noexcept override
- std::string **getMessage** () const noexcept override
- std::string **getFormattedMessage** () const noexcept override

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

- common/Exception/FactoryException.hpp

## 5.20 FactoryManager Class Reference

### Public Member Functions

- `std::shared_ptr< PrimitiveFactory > getPrimitiveFactory ()`
- `std::shared_ptr< CameraFactory > getCameraFactory ()`
- `void initializeFactories ()`
- `void createObjectsFromConfig (const std::vector< std::shared_ptr< std::map< ValueType_t, ValueType > > > &objectsConfig)`
- `std::shared_ptr< GraphsNodePrimitive > getPrimitives () const`
- `std::vector< std::shared_ptr< ILight > > getLights () const`
- `std::shared_ptr< Camera > getCamera () const`
- `float getAmbientLight () const`

### Private Attributes

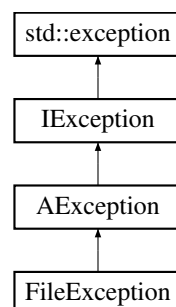
- `std::shared_ptr< PrimitiveFactory > _primitiveFactory`
- `std::shared_ptr< CameraFactory > _cameraFactory`
- `std::shared_ptr< LightFactory > _lightsFactory`
- `std::shared_ptr< GraphsNodePrimitive > _primitives`
- `std::vector< std::shared_ptr< ILight > > _lights`
- `std::shared_ptr< Camera > _camera`
- `float _ambientLight`

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

- `src/factory/FactoryManager.hpp`
- `src/factory/FactoryManager.cpp`

## 5.21 FileException Class Reference

Inheritance diagram for FileException:



### Public Member Functions

- `FileException (const std::string &message)`

### Public Member Functions inherited from [AException](#)

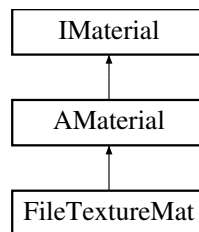
- **AException** (const std::string &type, const std::string &message)
- const char \* [what](#) () const noexcept override
- std::string [getType](#) () const noexcept override
- std::string [getMessage](#) () const noexcept override
- std::string [getFormattedMessage](#) () const noexcept override

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

- common/Exception/FileException.hpp

## 5.22 FileTextureMat Class Reference

Inheritance diagram for FileTextureMat:



### Public Member Functions

- [Color](#) [applyMaterial](#) (const [PixellInfo](#) &pixellInfo, float radius, float height, const [IPrimitives](#) &primitive) const override

### Public Member Functions inherited from [AMaterial](#)

- void [setAmbient](#) (const Math::Vector3D &a) override
- void [setDiffuse](#) (const Math::Vector3D &d) override
- void [setSpecular](#) (const Math::Vector3D &s) override
- void [setShininess](#) (float s) override
- void [setReflectivity](#) (float r) override
- void [setTransparency](#) (float t) override
- void [setRefractiveIndex](#) (float i) override
- void [setOpacity](#) (float o) override
- void [setColorTexture](#) (const std::shared\_ptr< std::string > &texture) override
- void [setNormalMap](#) (const std::shared\_ptr< std::string > &map) override
- void [setOptionalColor1](#) (const [Color](#) &color) override
- void [setOptionalColor2](#) (const [Color](#) &color) override
- void [setScale](#) (float s) override
- void [setMaterialType](#) (MaterialType type) override
- Math::Vector3D [getAmbient](#) () const override
- Math::Vector3D [getDiffuse](#) () const override
- Math::Vector3D [getSpecular](#) () const override
- float [getShininess](#) () const override

- float [getReflectivity](#) () const override
- float [getTransparency](#) () const override
- float [getRefractiveIndex](#) () const override
- float [getOpacity](#) () const override
- std::shared\_ptr< std::string > [getColorTexture](#) () const override
- std::shared\_ptr< std::string > [getNormalMap](#) () const override
- [Color](#) [getOptionalColor1](#) () const override
- [Color](#) [getOptionalColor2](#) () const override
- float [getScale](#) () const override
- MaterialType [getMaterialType](#) () const override

### Private Member Functions

- void **loadTextureFromFile** (const std::string &filePath)
- [Color](#) **getTextureFromFile** (const [PixelInfo](#) &pixelInfo, std::shared\_ptr< float > u, std::shared\_ptr< float > v) const
- void **calculUVCoordinates** (const [IPrimitives](#) &primitive, const [PixelInfo](#) &pixelInfo, float radius, float height, std::shared\_ptr< float > u, std::shared\_ptr< float > v) const
- void **calculUVCoordinatesSphere** (const [IPrimitives](#) &primitive, const [PixelInfo](#) &pixelInfo, float radius, std::shared\_ptr< float > u, std::shared\_ptr< float > v) const
- void **calculUVCoordinatesCylinder** (const [IPrimitives](#) &primitive, const [PixelInfo](#) &pixelInfo, float radius, float height, std::shared\_ptr< float > u, std::shared\_ptr< float > v) const
- void **calculUVCoordinatesCone** (const [IPrimitives](#) &primitive, const [PixelInfo](#) &pixelInfo, float radius, float height, std::shared\_ptr< float > u, std::shared\_ptr< float > v) const
- void **calculUVCoordinatesPlane** (const [IPrimitives](#) &primitive, const [PixelInfo](#) &pixelInfo, float radius, float height, std::shared\_ptr< float > u, std::shared\_ptr< float > v) const

### Additional Inherited Members

### Public Attributes inherited from [AMaterial](#)

- Math::Vector3D **ambient**
- Math::Vector3D **diffuse**
- Math::Vector3D **specular**
- float **shininess**
- [Color](#) **OptionalColor1** = [Color](#)(0.0f, 0.0f, 0.0f)
- [Color](#) **OptionalColor2** = [Color](#)(0.0f, 0.0f, 0.0f)
- float **scale** = 5.0f
- float **reflectivity**
- float **transparency**
- float **refractiveIndex**
- float **opacity**
- std::shared\_ptr< std::string > **colorTexture**
- std::shared\_ptr< std::string > **normalMap**
- MaterialType **materialType** = MaterialType::FLAT\_COLOR

## 5.22.1 Member Function Documentation

### 5.22.1.1 applyMaterial()

```
Color FileTextureMat::applyMaterial (
    const PixelInfo & pixelInfo,
    float radius,
    float height,
    const IPrimitives & primitive ) const [override], [virtual]
```

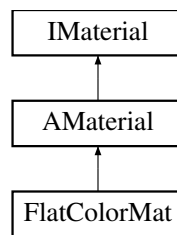
Implements [AMaterial](#).

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

- common/material/fileTextureMat.hpp
- common/material/fileTextureMat.cpp

## 5.23 FlatColorMat Class Reference

Inheritance diagram for FlatColorMat:



### Public Member Functions

- [Color applyMaterial](#) (const [PixelInfo](#) &pixelInfo, float radius, float height, const [IPrimitives](#) &primitive) const override

### Public Member Functions inherited from [AMaterial](#)

- void [setAmbient](#) (const Math::Vector3D &a) override
- void [setDiffuse](#) (const Math::Vector3D &d) override
- void [setSpecular](#) (const Math::Vector3D &s) override
- void [setShininess](#) (float s) override
- void [setReflectivity](#) (float r) override
- void [setTransparency](#) (float t) override
- void [setRefractiveIndex](#) (float i) override
- void [setOpacity](#) (float o) override
- void [setColorTexture](#) (const std::shared\_ptr< std::string > &texture) override
- void [setNormalMap](#) (const std::shared\_ptr< std::string > &map) override
- void [setOptionalColor1](#) (const [Color](#) &color) override
- void [setOptionalColor2](#) (const [Color](#) &color) override
- void [setScale](#) (float s) override

- void [setMaterialType](#) (MaterialType type) override
- Math::Vector3D [getAmbient](#) () const override
- Math::Vector3D [getDiffuse](#) () const override
- Math::Vector3D [getSpecular](#) () const override
- float [getShininess](#) () const override
- float [getReflectivity](#) () const override
- float [getTransparency](#) () const override
- float [getRefractiveIndex](#) () const override
- float [getOpacity](#) () const override
- std::shared\_ptr< std::string > [getColorTexture](#) () const override
- std::shared\_ptr< std::string > [getNormalMap](#) () const override
- [Color](#) [getOptionalColor1](#) () const override
- [Color](#) [getOptionalColor2](#) () const override
- float [getScale](#) () const override
- MaterialType [getMaterialType](#) () const override

### Additional Inherited Members

### Public Attributes inherited from [AMaterial](#)

- Math::Vector3D **ambient**
- Math::Vector3D **diffuse**
- Math::Vector3D **specular**
- float **shininess**
- [Color](#) **OptionalColor1** = [Color](#)(0.0f, 0.0f, 0.0f)
- [Color](#) **OptionalColor2** = [Color](#)(0.0f, 0.0f, 0.0f)
- float **scale** = 5.0f
- float **reflectivity**
- float **transparency**
- float **refractiveIndex**
- float **opacity**
- std::shared\_ptr< std::string > **colorTexture**
- std::shared\_ptr< std::string > **normalMap**
- MaterialType **materialType** = MaterialType::FLAT\_COLOR

## 5.23.1 Member Function Documentation

### 5.23.1.1 [applyMaterial\(\)](#)

```
Color FlatColorMat::applyMaterial (
    const PixelInfo & pixelInfo,
    float radius,
    float height,
    const IPrimitives & primitive ) const [override], [virtual]
```

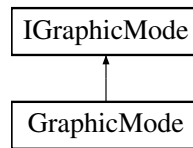
Implements [AMaterial](#).

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

- common/material/flatColorMat.hpp
- common/material/flatColorMat.cpp

## 5.24 GraphicMode Class Reference

Inheritance diagram for GraphicMode:



### Public Member Functions

- void [createText](#) (const std::string &text, int size, int x, int y) override
- void [createRectangle](#) (const std::string &id, int x, int y, int width, int height) override
- bool [getRenderingComplete](#) () const override
- void [setWindow](#) (int width, int height) override
- void [setRenderingComplete](#) (bool renderingComplete) override
- std::string [getButtonPressed](#) () override
- void [updateTexture](#) () override
- void [drawPixelColor](#) (int x, int y, uint8\_t r, uint8\_t g, uint8\_t b) override
- void [drawImage](#) () override
- void [drawButtons](#) () override
- void [display](#) () override
- bool [isOpen](#) () override
- void [closeWindow](#) () override

### Private Attributes

- sf::Event **\_event**
- std::shared\_ptr< sf::RenderWindow > **\_window**
- std::shared\_ptr< sf::Image > **\_image**
- std::map< std::string, sf::RectangleShape > **\_buttons**
- std::shared\_ptr< sf::Font > **\_font**
- std::vector< sf::Text > **\_texts**
- sf::Texture **\_texture**
- std::string **\_title**
- int **\_width**
- int **\_height**
- bool **\_renderingComplete**

### 5.24.1 Member Function Documentation

#### 5.24.1.1 closeWindow()

```
void GraphicMode::closeWindow ( ) [override], [virtual]
```

Implements [IGraphicMode](#).

#### 5.24.1.2 createRectangle()

```
void GraphicMode::createRectangle (
    const std::string & id,
    int x,
    int y,
    int width,
    int height ) [override], [virtual]
```

Implements [IGraphicMode](#).

#### 5.24.1.3 createText()

```
void GraphicMode::createText (
    const std::string & text,
    int size,
    int x,
    int y ) [override], [virtual]
```

Implements [IGraphicMode](#).

#### 5.24.1.4 display()

```
void GraphicMode::display ( ) [override], [virtual]
```

Implements [IGraphicMode](#).

#### 5.24.1.5 drawButtons()

```
void GraphicMode::drawButtons ( ) [override], [virtual]
```

Implements [IGraphicMode](#).

#### 5.24.1.6 drawImage()

```
void GraphicMode::drawImage ( ) [override], [virtual]
```

Implements [IGraphicMode](#).

#### 5.24.1.7 drawPixelColor()

```
void GraphicMode::drawPixelColor (
    int x,
    int y,
    uint8_t r,
    uint8_t g,
    uint8_t b ) [override], [virtual]
```

Implements [IGraphicMode](#).



#### 5.24.1.8 getButtonPressed()

```
std::string GraphicMode::getButtonPressed ( ) [override], [virtual]
```

Implements [IGraphicMode](#).

#### 5.24.1.9 getRenderingComplete()

```
bool GraphicMode::getRenderingComplete ( ) const [override], [virtual]
```

Implements [IGraphicMode](#).

#### 5.24.1.10 isOpen()

```
bool GraphicMode::isOpen ( ) [override], [virtual]
```

Implements [IGraphicMode](#).

#### 5.24.1.11 setRenderingComplete()

```
void GraphicMode::setRenderingComplete (
    bool renderingComplete ) [override], [virtual]
```

Implements [IGraphicMode](#).

#### 5.24.1.12 setWindow()

```
void GraphicMode::setWindow (
    int width,
    int height ) [override], [virtual]
```

Implements [IGraphicMode](#).

#### 5.24.1.13 updateTexture()

```
void GraphicMode::updateTexture ( ) [override], [virtual]
```

Implements [IGraphicMode](#).

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

- lib/SFML/GraphicMode.hpp
- lib/SFML/GraphicMode.cpp

## 5.25 GraphsNodeLight Struct Reference

### Public Member Functions

- [GraphsNodeLight](#) & **operator=** (const [GraphsNodeLight](#) &other)
- template<typename Func >  
void **traverseGraph** (const std::shared\_ptr< [GraphsNodeLight](#) > &node, Func &&func)

### Public Attributes

- std::shared\_ptr< [ILight](#) > **\_primitives**
- std::vector< std::shared\_ptr< [GraphsNodeLight](#) > > **\_children**

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

- common/Graphs.hpp

## 5.26 GraphsNodePrimitive Struct Reference

### Public Member Functions

- [GraphsNodePrimitive](#) & **operator=** (const [GraphsNodePrimitive](#) &other)
- template<typename Func >  
void **traverseGraph** (const std::shared\_ptr< [GraphsNodePrimitive](#) > &node, Func &&func)

### Public Attributes

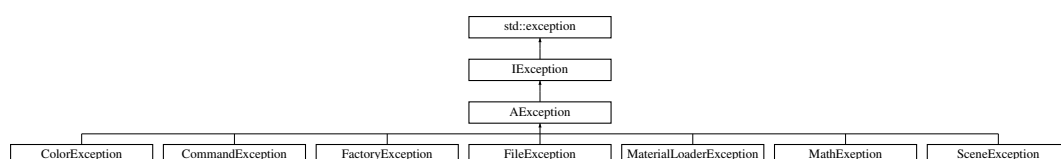
- std::shared\_ptr< [IPrimitives](#) > **\_primitives**
- std::vector< std::shared\_ptr< [GraphsNodePrimitive](#) > > **\_children**

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

- common/Graphs.hpp

## 5.27 IException Class Reference

Inheritance diagram for IException:



**Public Member Functions**

- const char \* **what** () const noexcept override=0
- virtual std::string **getType** () const noexcept=0
- virtual std::string **getMessage** () const noexcept=0
- virtual std::string **getFormattedMessage** () const noexcept=0

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

- common/Exception/IException.hpp

**5.28 IFactory< T > Class Template Reference****Public Member Functions**

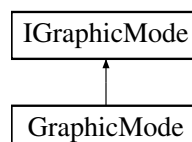
- virtual std::shared\_ptr< T > **create** (const std::string &type, std::shared\_ptr< std::map< ValueType\_t, ValueType > > config, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList, const std::vector< std::shared\_ptr< [ILight](#) > > &lights)=0
- virtual std::shared\_ptr< T > **createSimple** (const std::string &type, std::shared\_ptr< std::map< ValueType\_t, ValueType > > config, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList)=0
- virtual void **registerCreator** (const std::string &type, std::function< std::shared\_ptr< T >(std::shared\_ptr< std::map< ValueType\_t, ValueType > >, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &)> creator)=0
- virtual void **registerCreatorLight** (const std::string &type, std::function< std::shared\_ptr< T >(std::shared\_ptr< std::map< ValueType\_t, ValueType > >, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &, const std::vector< std::shared\_ptr< [ILight](#) > > &)> creator)=0
- virtual bool **loadPlugin** (const std::string &path)=0

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

- src/factory/IFactory.hpp

**5.29 IGraphicMode Class Reference**

Inheritance diagram for IGraphicMode:



### Public Member Functions

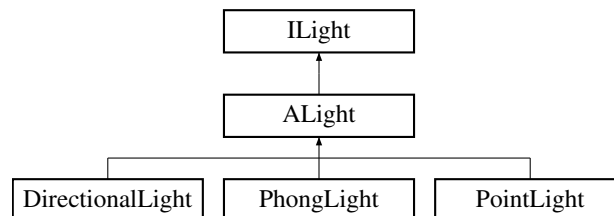
- virtual void **createText** (const std::string &text, int size, int x, int y)=0
- virtual void **createRectangle** (const std::string &id, int x, int y, int width, int height)=0
- virtual bool **getRenderingComplete** () const =0
- virtual void **setWindow** (int width, int height)=0
- virtual void **setRenderingComplete** (bool renderingComplete)=0
- virtual std::string **getButtonPressed** ()=0
- virtual void **updateTexture** ()=0
- virtual void **drawPixelColor** (int x, int y, uint8\_t r, uint8\_t g, uint8\_t b)=0
- virtual void **drawImage** ()=0
- virtual void **drawButtons** ()=0
- virtual void **display** ()=0
- virtual bool **isOpen** ()=0
- virtual void **closeWindow** ()=0

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

- common/IGraphicMode.hpp

## 5.30 ILight Class Reference

Inheritance diagram for ILight:



### Public Member Functions

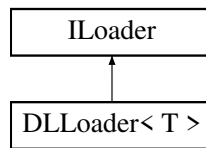
- virtual void **addLight** (PixelInfo &pixelInfo, const Math::Ray &ray) const =0
- virtual Color **getColor** () const =0
- virtual Math::Vector3D **getDirection** () const =0
- virtual float **getIntensity** () const =0
- virtual float **getRadius** () const =0
- virtual TypeLight **getTypeLight** () const =0
- virtual void **setColor** (const Color &color)=0
- virtual void **setIntensity** (float intensity)=0

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

- common/ILight.hpp

## 5.31 ILoader Class Reference

Inheritance diagram for ILoader:



### Public Member Functions

- virtual void \* **Open** (const char \*path, int flag)=0
- virtual void \* **Symbol** (const char \*symbolName)=0
- virtual int **Close** ()=0
- virtual const char \* **Error** ()=0
- virtual void \* **getHandler** () const =0

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

- lib/ILoader.hpp

## 5.32 Image Class Reference

### Public Member Functions

- **Image** (int width, int height, int maxColorValue=255)
- int **getWidth** () const
- int **getHeight** () const
- int **getMaxColorValue** () const
- const std::vector< [Color](#) > & **getData** () const
- void **writeToFilePPM** (const std::string &fileName) const
- void **setPixel** (int x, int y, const [Color](#) &color)
- [Color](#) **getPixel** (int x, int y) const

### Private Attributes

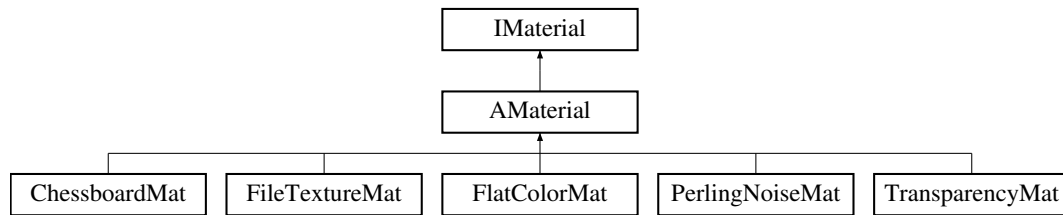
- int **width**
- int **height**
- int **maxColorValue**
- std::vector< [Color](#) > **data**

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

- common/Image.hpp
- common/Image.cpp

## 5.33 IMaterial Class Reference

Inheritance diagram for IMaterial:



### Public Member Functions

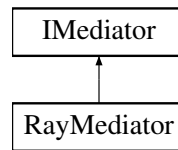
- virtual void **setAmbient** (const Math::Vector3D &a)=0
- virtual void **setDiffuse** (const Math::Vector3D &d)=0
- virtual void **setSpecular** (const Math::Vector3D &s)=0
- virtual void **setShininess** (float s)=0
- virtual void **setReflectivity** (float r)=0
- virtual void **setTransparency** (float t)=0
- virtual void **setRefractiveIndex** (float i)=0
- virtual void **setOpacity** (float o)=0
- virtual void **setColorTexture** (const std::shared\_ptr< std::string > &texture)=0
- virtual void **setNormalMap** (const std::shared\_ptr< std::string > &map)=0
- virtual void **setOptionalColor1** (const Color &color)=0
- virtual void **setOptionalColor2** (const Color &color)=0
- virtual void **setScale** (float s)=0
- virtual void **setMaterialType** (MaterialType type)=0
- virtual Math::Vector3D **getAmbient** () const =0
- virtual Math::Vector3D **getDiffuse** () const =0
- virtual Math::Vector3D **getSpecular** () const =0
- virtual float **getShininess** () const =0
- virtual float **getReflectivity** () const =0
- virtual float **getTransparency** () const =0
- virtual float **getRefractiveIndex** () const =0
- virtual float **getOpacity** () const =0
- virtual std::shared\_ptr< std::string > **getColorTexture** () const =0
- virtual std::shared\_ptr< std::string > **getNormalMap** () const =0
- virtual Color **getOptionalColor1** () const =0
- virtual Color **getOptionalColor2** () const =0
- virtual float **getScale** () const =0
- virtual MaterialType **getMaterialType** () const =0
- virtual Color **applyMaterial** (const PixelInfo &pixelInfo, float radius, float height, const IPrimitives &primitive) const =0

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

- common/material/IMaterial.hpp

## 5.34 IMediator Class Reference

Inheritance diagram for IMediator:



### Public Member Functions

- virtual void **addTask** (std::function< void()> task)=0
- virtual void **executeTasks** ()=0
- virtual void **waitForCompletion** ()=0

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

- src/mediator/IMediator.hpp

## 5.35 InfoPixelDisplay Struct Reference

### Public Attributes

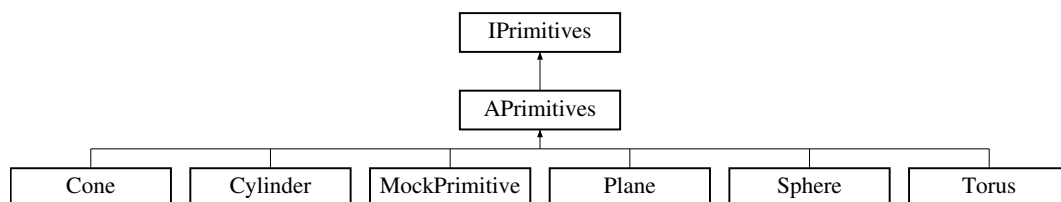
- double **distance**
- [Color](#) **color**
- float **transparency**

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

- src/Raytracer.hpp

## 5.36 IPrimitives Class Reference

Inheritance diagram for IPrimitives:



## Public Member Functions

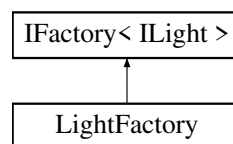
- virtual std::optional< double > **distance** (const [Math::Ray](#) &ray) const =0
- virtual [PixellInfo](#) **distanceInfo** (const [Math::Ray](#) &ray)=0
- virtual std::optional< [Math::Point3D](#) > **getIntersection** (const [Math::Ray](#) &ray) const =0
- virtual std::optional< [Math::Vector3D](#) > **getNormal** (const [Math::Point3D](#) &point) const =0
- virtual [Math::Ray](#) **computeScaledRay** (const [Math::Ray](#) &ray) const =0
- virtual Type **getType** () const =0
- virtual [Math::Point3D](#) **getPosition** () const =0
- virtual [Math::Rot3D](#) **getRotation** () const =0
- virtual float **getScale** () const =0
- virtual std::shared\_ptr< [IMaterial](#) > **getMaterial** () const =0
- virtual void **setPosition** (const [Math::Point3D](#) &position)=0
- virtual void **setRotation** (const [Math::Rot3D](#) &rotation)=0
- virtual void **setScale** (float scale)=0
- virtual void **setMaterial** (std::shared\_ptr< [IMaterial](#) > material)=0

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

- common/IPrimitives.hpp

## 5.37 LightFactory Class Reference

Inheritance diagram for LightFactory:



## Public Member Functions

- std::shared\_ptr< [ILight](#) > **create** (const std::string &type, std::shared\_ptr< std::map< ValueType\_t, Value↵  
Type > > config, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graph↵  
SceneList, const std::vector< std::shared\_ptr< [ILight](#) > > &lights) override
- std::shared\_ptr< [ILight](#) > **createSimple** (const std::string &type, std::shared\_ptr< std::map< ValueType↵  
\_t, ValueType > > config, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > ↵  
&graphSceneList) override
- void **registerCreator** (const std::string &type, std::function< std::shared\_ptr< [ILight](#) >(std::shared\_ptr< std↵  
::map< ValueType\_t, ValueType > >, const std::vector< std::shared\_ptr< std::map< ValueType\_t, Value↵  
Type > > > &)> creator) override
- void **registerCreatorLight** (const std::string &type, std::function< std::shared\_ptr< [ILight](#) >(std::shared\_↵  
ptr< std::map< ValueType\_t, ValueType > >, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ↵  
ValueType > > > &, const std::vector< std::shared\_ptr< [ILight](#) > > &)> creator) override
- bool **loadPlugin** (const std::string &path) override
- void **loadAllPlugins** (const std::string &directory="plugins/")
- ObjectType **getTypeFromPlugin** (const std::string &path, [DLLoader](#)< void \* > loader)
- std::string **getNameFromPlugin** (const std::string &path, [DLLoader](#)< void \* > loader)



## Private Attributes

- `std::map< std::string, std::function< std::shared_ptr< ILight >(std::shared_ptr< std::map< ValueType_t, ValueType > >, const std::vector< std::shared_ptr< std::map< ValueType_t, ValueType > > > &)>_creators)`
- `std::vector< DLLoader< void * > > _dlLoaders`

## 5.37.1 Member Function Documentation

### 5.37.1.1 create()

```
std::shared_ptr< ILight > LightFactory::create (
    const std::string & type,
    std::shared_ptr< std::map< ValueType_t, ValueType > > config,
    const std::vector< std::shared_ptr< std::map< ValueType_t, ValueType > > > &
graphSceneList,
    const std::vector< std::shared_ptr< ILight > > & lights ) [override], [virtual]
```

Implements [IFactory< \[ILight\]\(#\) >](#).

### 5.37.1.2 createSimple()

```
std::shared_ptr< ILight > LightFactory::createSimple (
    const std::string & type,
    std::shared_ptr< std::map< ValueType_t, ValueType > > config,
    const std::vector< std::shared_ptr< std::map< ValueType_t, ValueType > > > &
graphSceneList ) [override], [virtual]
```

Implements [IFactory< \[ILight\]\(#\) >](#).

### 5.37.1.3 loadPlugin()

```
bool LightFactory::loadPlugin (
    const std::string & path ) [override], [virtual]
```

Implements [IFactory< \[ILight\]\(#\) >](#).

### 5.37.1.4 registerCreator()

```
void LightFactory::registerCreator (
    const std::string & type,
    std::function< std::shared_ptr< ILight >(std::shared_ptr< std::map< ValueType_t, ValueType > >, const std::vector< std::shared_ptr< std::map< ValueType_t, ValueType > > > &)> creator ) [override], [virtual]
```

Implements [IFactory< \[ILight\]\(#\) >](#).

### 5.37.1.5 registerCreatorLight()

```
void LightFactory::registerCreatorLight (
    const std::string & type,
    std::function< std::shared_ptr< TLight >(std::shared_ptr< std::map< ValueType←
_t, ValueType > >, const std::vector< std::shared_ptr< std::map< ValueType_t, ValueType > >
> &, const std::vector< std::shared_ptr< TLight > > &)> creator ) [override], [virtual]
```

Implements [IFactory< TLight >](#).

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

- src/factory/LightFactory.hpp
- src/factory/LightFactory.cpp

## 5.38 Material Struct Reference

### Public Member Functions

- **Material** (const Math::Vector3D &a, const Math::Vector3D &d, const Math::Vector3D &s, float shin, float refl=0.0f, float trans=0.0f, float ior=1.0f)

### Public Attributes

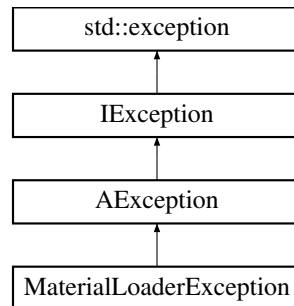
- Math::Vector3D **ambient**
- Math::Vector3D **diffuse**
- Math::Vector3D **specular**
- float **shininess**
- **Color OptionalColor1** = [Color](#)(0.0f, 0.0f, 0.0f)
- **Color OptionalColor2** = [Color](#)(0.0f, 0.0f, 0.0f)
- float **scale** = 5.0f
- float **reflectivity**
- float **transparency**
- float **refractiveIndex**
- float **opacity**
- std::shared\_ptr< std::string > **colorTexture**
- std::shared\_ptr< std::string > **normalMap**
- MaterialType **materialType** = FLAT\_COLOR

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

- common/Material.hpp

## 5.39 MaterialLoaderException Class Reference

Inheritance diagram for MaterialLoaderException:



### Public Member Functions

- **MaterialLoaderException** (const std::string &message)

### Public Member Functions inherited from **AException**

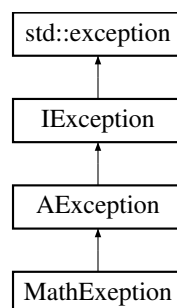
- **AException** (const std::string &type, const std::string &message)
- const char \* **what** () const noexcept override
- std::string **getType** () const noexcept override
- std::string **getMessage** () const noexcept override
- std::string **getFormattedMessage** () const noexcept override

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

- common/Exception/materialLoaderException.hpp

## 5.40 MathExeption Class Reference

Inheritance diagram for MathExeption:



### Public Member Functions

- **MathExeption** (const std::string &message)

## Public Member Functions inherited from [AException](#)

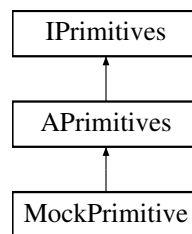
- **AException** (const std::string &type, const std::string &message)
- const char \* [what](#) () const noexcept override
- std::string [getType](#) () const noexcept override
- std::string [getMessage](#) () const noexcept override
- std::string [getFormattedMessage](#) () const noexcept override

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

- common/Exception/MathExeption.hpp

## 5.41 MockPrimitive Class Reference

Inheritance diagram for MockPrimitive:



### Public Member Functions

- [PixelInfo distanceInfo](#) (const [Math::Ray](#) &ray) override
- std::optional< double > [distance](#) (const [Math::Ray](#) &ray) const override
- std::optional< [Math::Point3D](#) > [getIntersection](#) (const [Math::Ray](#) &ray) const override
- std::optional< [Math::Vector3D](#) > [getNormal](#) (const [Math::Point3D](#) &point) const override
- Type [getType](#) () const override

## Public Member Functions inherited from [APrimitives](#)

- **APrimitives** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList, std::vector< std::shared\_ptr< [ILight](#) > > light)
- [Math::Ray computeScaledRay](#) (const [Math::Ray](#) &ray) const override
- [Math::Point3D getPosition](#) () const override
- [Math::Rot3D getRotation](#) () const override
- float [getScale](#) () const override
- std::shared\_ptr< [IMaterial](#) > [getMaterial](#) () const override
- [Math::Vector3D getInvScales](#) () const
- const std::vector< std::shared\_ptr< [ILight](#) > > & [getLights](#) () const
- void [setPosition](#) (const [Math::Point3D](#) &newPosition) override
- void [setRotation](#) (const [Math::Rot3D](#) &newRotation) override
- void [setScale](#) (const float newScale) override
- void [setMaterial](#) (std::shared\_ptr< [IMaterial](#) > newMaterial) override
- void **addLight** (std::shared\_ptr< [ILight](#) > light)

- void **clearLights** ()
- void **applyLights** ([PixelInfo](#) &pixelInfo, const [Math::Ray](#) &ray) const
- void **getPos** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getRot** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getCol** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getScales** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getGraph** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList)
- void **computeInvScales** ()
- void **getGraphScale** (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)
- void **getPosGraph** (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)
- void **getRotGraph** (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)

### Additional Inherited Members

### Protected Attributes inherited from [APrimitives](#)

- [Math::Point3D](#) **\_position**
- [Math::Rot3D](#) **\_rotation**
- float **scale**
- std::shared\_ptr< [IMaterial](#) > **material**
- [Color](#) **\_color**
- [Math::Vector3D](#) **\_scales**
- [Math::Vector3D](#) **\_invScales**
- std::string **\_graph**
- std::vector< std::shared\_ptr< [ILight](#) > > **\_lights**

## 5.41.1 Member Function Documentation

### 5.41.1.1 distance()

```
std::optional< double > MockPrimitive::distance (
    const Math::Ray & ray ) const [inline], [override], [virtual]
```

Implements [APrimitives](#).

### 5.41.1.2 distanceInfo()

```
PixelInfo MockPrimitive::distanceInfo (
    const Math::Ray & ray ) [inline], [override], [virtual]
```

Implements [APrimitives](#).

### 5.41.1.3 getIntersection()

```
std::optional< Math::Point3D > MockPrimitive::getIntersection (
    const Math::Ray & ray ) const [inline], [override], [virtual]
```

Implements [IPrimitives](#).

#### 5.41.1.4 getNormal()

```
std::optional< Math::Vector3D > MockPrimitive::getNormal (
    const Math::Point3D & point ) const [inline], [override], [virtual]
```

Implements [IPrimitives](#).

#### 5.41.1.5 getType()

```
Type MockPrimitive::getType ( ) const [inline], [override], [virtual]
```

Implements [APrimitives](#).

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

- tests/APrimitives-test.cpp

## 5.42 ObjectConstructor Class Reference

### Public Member Functions

- void **createObject** (const [ConfigNode](#) &node)
- void **createObjects** (const [ConfigNode](#) &node)
- bool **verifyObjectValidity** (const [ConfigNode](#) &node, const std::string &objectName)
- const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > & **getObjects** () const
- void **printObjectMap** () const
- bool **createMaterials** (const [ConfigNode](#) &node)

### Private Member Functions

- void **fillObject** (const [ConfigNode](#) &node, std::shared\_ptr< std::map< ValueType\_t, ValueType > > object)
- void **handleSimpleValue** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > object, const Value↵Type\_t &key, const ValueType &value, ValueDataType dataType)
- void **handleVector2DValue** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > object, const Value↵Type\_t &key, const [ConfigNode](#) &node, const std::vector< std::string > &components, ValueDataType data↵Type)
- void **handleVector3DValue** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > object, const Value↵Type\_t &key, const [ConfigNode](#) &node, const std::vector< std::string > &components, ValueDataType data↵Type)
- ValueType **convertValue** (const ValueType &value, ValueDataType dataType)
- void **initShapeDefinitions** ()

### Private Attributes

- [ObjectErrorHandling](#) **\_errorHandler**
- std::map< std::string, [PropertyConfig](#) > **\_propertyTypeMap**
- std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > **\_objects**
- std::vector< [ShapeDefinition](#) > **\_shapeDefinitions**

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

- src/parser/ObjectConstructor.hpp
- src/parser/ObjectConstructor.cpp

## 5.43 ObjectErrorHandling Class Reference

### Public Member Functions

- bool **checkArrayValidity** (const [ConfigNode](#) &node, const std::string &objectName)
- bool **checkGroupValidity** (const [ConfigNode](#) &node, const std::string &objectName)
- bool **checkListValidity** (const [ConfigNode](#) &node, const std::string &objectName)
- bool **checkValueValidity** (const [ConfigNode](#) &node, const std::string &objectName)
- bool **verifyObjectValidity** (const [ConfigNode](#) &node, const std::string &objectName)
- void **setShapeDefinitions** (std::vector< [ShapeDefinition](#) > shapeDefinitions)
- void **setPropertyTypeMap** (const std::map< std::string, [PropertyConfig](#) > &propertyTypeMap)

### Private Member Functions

- std::string **getDataTypeName** (ValueDataType type) const
- std::shared\_ptr< const [ShapeDefinition](#) > **getShapeDefinition** (const std::string &objectName) const
- bool **isParameterValid** (const std::string &parameter, const std::string &objectName) const
- bool **isParameterMandatory** (const std::string &parameter, const std::string &objectName) const
- bool **isParameterOptional** (const std::string &parameter, const std::string &objectName) const
- bool **checkParameterType** (const std::string &parameter, const [ConfigNode](#) &node) const
- bool **isValueTypeValid** (const ValueType &value, ValueDataType expectedType) const
- bool **checkSimpleValueValidity** (const [ConfigNode](#) &node, const std::string &parameter, const [PropertyConfig](#) &config) const
- bool **checkVector2DValueValidity** (const [ConfigNode](#) &node, const std::string &parameter, const [PropertyConfig](#) &config) const
- bool **checkVector3DValueValidity** (const [ConfigNode](#) &node, const std::string &parameter, const [PropertyConfig](#) &config) const
- bool **checkMandatoryParameters** (const [ConfigNode](#) &node, const std::string &objectName) const
- bool **checkOptionalParameters** (const [ConfigNode](#) &node, const std::string &objectName) const
- bool **checkUnknownParameters** (const [ConfigNode](#) &node, const std::string &objectName) const

### Private Attributes

- std::vector< [ShapeDefinition](#) > **\_shapeDefinitions**
- std::map< std::string, [PropertyConfig](#) > **\_propertyTypeMap**

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

- src/parser/ObjectErrorHandling.hpp
- src/parser/ObjectErrorHandling.cpp

## 5.44 Parser Class Reference

### Public Member Functions

- **Parser** (const std::string &filename)
- void **loadConfig** (const std::string &filename)
- void **parse** ()
- const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > & **getObjects** () const
- std::shared\_ptr< [ConfigNode](#) > **getRootNode** ()
- void **printMap** () const

### Private Member Functions

- void **importScene** ()
- bool **isValidFilePath** (const std::string &path) const
- bool **loadImportedScene** (const std::string &scenePath, std::shared\_ptr< [ConfigNode](#) > importedRootNode) const
- void **importObjectsFromScene** (const std::shared\_ptr< [ConfigNode](#) > importedRootNode)

### Private Attributes

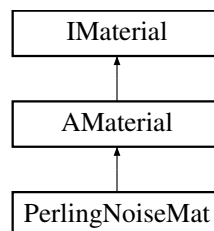
- [ConfigNode](#) **rootNode**
- [ConfigParser](#) **configParser**
- [ObjectConstructor](#) **\_objectConstructor**

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

- src/parser/Parser.hpp
- src/parser/Parser.cpp

## 5.45 PerlingNoiseMat Class Reference

Inheritance diagram for PerlingNoiseMat:



### Public Member Functions

- [Color](#) **applyMaterial** (const [PixellInfo](#) &pixellInfo, float radius, float height, const [IPrimitives](#) &primitive) const override

### Public Member Functions inherited from [AMaterial](#)

- void [setAmbient](#) (const Math::Vector3D &a) override
- void [setDiffuse](#) (const Math::Vector3D &d) override
- void [setSpecular](#) (const Math::Vector3D &s) override
- void [setShininess](#) (float s) override
- void [setReflectivity](#) (float r) override
- void [setTransparency](#) (float t) override
- void [setRefractiveIndex](#) (float i) override
- void [setOpacity](#) (float o) override
- void [setColorTexture](#) (const std::shared\_ptr< std::string > &texture) override
- void [setNormalMap](#) (const std::shared\_ptr< std::string > &map) override



- void [setOptionalColor1](#) (const [Color](#) &color) override
- void [setOptionalColor2](#) (const [Color](#) &color) override
- void [setScale](#) (float s) override
- void [setMaterialType](#) (MaterialType type) override
- Math::Vector3D [getAmbient](#) () const override
- Math::Vector3D [getDiffuse](#) () const override
- Math::Vector3D [getSpecular](#) () const override
- float [getShininess](#) () const override
- float [getReflectivity](#) () const override
- float [getTransparency](#) () const override
- float [getRefractiveIndex](#) () const override
- float [getOpacity](#) () const override
- std::shared\_ptr< std::string > [getColorTexture](#) () const override
- std::shared\_ptr< std::string > [getNormalMap](#) () const override
- [Color](#) [getOptionalColor1](#) () const override
- [Color](#) [getOptionalColor2](#) () const override
- float [getScale](#) () const override
- MaterialType [getMaterialType](#) () const override

#### Private Attributes

- float **frequency**
- float **amplitude**
- int **octaves**
- float **persistence**

#### Additional Inherited Members

#### Public Attributes inherited from [AMaterial](#)

- Math::Vector3D **ambient**
- Math::Vector3D **diffuse**
- Math::Vector3D **specular**
- float **shininess**
- [Color](#) **OptionalColor1** = [Color](#)(0.0f, 0.0f, 0.0f)
- [Color](#) **OptionalColor2** = [Color](#)(0.0f, 0.0f, 0.0f)
- float **scale** = 5.0f
- float **reflectivity**
- float **transparency**
- float **refractiveIndex**
- float **opacity**
- std::shared\_ptr< std::string > **colorTexture**
- std::shared\_ptr< std::string > **normalMap**
- MaterialType **materialType** = MaterialType::FLAT\_COLOR

## 5.45.1 Member Function Documentation

### 5.45.1.1 applyMaterial()

```
Color PerlingNoiseMat::applyMaterial (
    const PixelInfo & pixelInfo,
    float radius,
    float height,
    const IPrimitives & primitive ) const [override], [virtual]
```

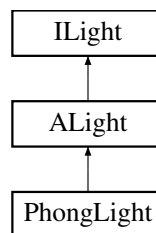
Implements [AMaterial](#).

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

- common/material/perlingNoiseMat.hpp
- common/material/perlingNoiseMat.cpp

## 5.46 PhongLight Class Reference

Inheritance diagram for PhongLight:



### Public Member Functions

- **PhongLight** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- **PhongLight** (const Color &color, float intensity, const Math::Vector3D &direction, float radius, float shininess)
- void **addLight** (PixelInfo &pixelInfo, const Math::Ray &ray) const override
- TypeLight **getTypeLight** () const override
- Math::Vector3D **getDirection** () const override
- float **getRadius** () const override
- float **getShininess** () const
- void **setDirection** (const Math::Vector3D &position)
- void **setShininess** (float shininess)

### Public Member Functions inherited from [ALight](#)

- **ALight** (const Color &color, float intensity)
- Color **getColor** () const override
- float **getIntensity** () const override
- void **setColor** (const Color &color) override
- void **setIntensity** (float intensity) override

### Private Attributes

- `Math::Vector3D _direction`
- `Color _color`
- `float _intensity`
- `float _radius`
- `float _shininess`

### Additional Inherited Members

### Protected Attributes inherited from [ALight](#)

- `Color _color`
- `float _intensity`

## 5.46.1 Member Function Documentation

### 5.46.1.1 `addLight()`

```
void PhongLight::addLight (
    PixelInfo & pixelInfo,
    const Math::Ray & ray ) const [override], [virtual]
```

Implements [ALight](#).

### 5.46.1.2 `getDirection()`

```
Math::Vector3D PhongLight::getDirection ( ) const [override], [virtual]
```

Implements [ALight](#).

### 5.46.1.3 `getRadius()`

```
float PhongLight::getRadius ( ) const [override], [virtual]
```

Implements [ALight](#).

### 5.46.1.4 `getTypeLight()`

```
TypeLight PhongLight::getTypeLight ( ) const [override], [virtual]
```

Implements [ALight](#).

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

- `src/lights/PhongLight.hpp`
- `src/lights/PhongLight.cpp`

## 5.47 PixelInfo Struct Reference

### Public Member Functions

- **PixelInfo** (const [Color](#) &color, const Math::Vector3D &normalVector, double distance, bool isHit, const Math::Vector3D &position, float lightIntensity, const [Color](#) &colorLight)

### Public Attributes

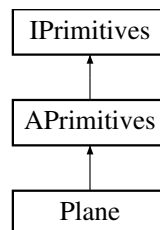
- [Color](#) \_color
- Math::Vector3D \_normalizedVector
- double \_distance
- bool \_isHit
- Math::Vector3D \_pos
- float \_light\_intensity
- [Color](#) \_light\_color

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

- common/PixelInfo.hpp

## 5.48 Plane Class Reference

Inheritance diagram for Plane:



### Public Member Functions

- **Plane** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList, const std::vector< std::shared\_ptr< [ILight](#) > > &lights)
- std::optional< double > [distance](#) (const [Math::Ray](#) &ray) const override
- [PixelInfo distanceInfo](#) (const [Math::Ray](#) &ray) override
- std::optional< [Math::Point3D](#) > [getIntersection](#) (const [Math::Ray](#) &ray) const override
- std::optional< Math::Vector3D > [getNormal](#) (const [Math::Point3D](#) &point) const override
- Type [getType](#) () const override
- void [setRotation](#) (const [Math::Rot3D](#) &newRotation) override
- void [updateNormal](#) ()

## Public Member Functions inherited from **APrimitives**

- **APrimitives** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList, std::vector< std::shared\_ptr< **ILight** > > light)
- **Math::Ray computeScaledRay** (const **Math::Ray** &ray) const override
- **Math::Point3D getPosition** () const override
- **Math::Rot3D getRotation** () const override
- float **getScale** () const override
- std::shared\_ptr< **IMaterial** > **getMaterial** () const override
- **Math::Vector3D getInvScales** () const
- const std::vector< std::shared\_ptr< **ILight** > > & **getLights** () const
- void **setPosition** (const **Math::Point3D** &newPosition) override
- void **setScale** (const float newScale) override
- void **setMaterial** (std::shared\_ptr< **IMaterial** > newMaterial) override
- void **addLight** (std::shared\_ptr< **ILight** > light)
- void **clearLights** ()
- void **applyLights** (**PixellInfo** &pixellInfo, const **Math::Ray** &ray) const
- void **getPos** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getRot** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getCol** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getScales** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getGraph** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList)
- void **computeInvScales** ()
- void **getGraphScale** (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)
- void **getPosGraph** (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)
- void **getRotGraph** (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)

## Private Attributes

- double **\_distance**
- **Math::Vector3D \_normal**
- std::string **\_axe**

## Additional Inherited Members

## Protected Attributes inherited from **APrimitives**

- **Math::Point3D \_position**
- **Math::Rot3D \_rotation**
- float **scale**
- std::shared\_ptr< **IMaterial** > **material**
- **Color \_color**
- **Math::Vector3D \_scales**
- **Math::Vector3D \_invScales**
- std::string **\_graph**
- std::vector< std::shared\_ptr< **ILight** > > **\_lights**

## 5.48.1 Member Function Documentation

### 5.48.1.1 distance()

```
std::optional< double > Plane::distance (
    const Math::Ray & ray ) const [override], [virtual]
```

Implements [APrimitives](#).

### 5.48.1.2 distanceInfo()

```
PixelInfo Plane::distanceInfo (
    const Math::Ray & ray ) [override], [virtual]
```

Implements [APrimitives](#).

### 5.48.1.3 getIntersection()

```
std::optional< Math::Point3D > Plane::getIntersection (
    const Math::Ray & ray ) const [override], [virtual]
```

Implements [IPrimitives](#).

### 5.48.1.4 getNormal()

```
std::optional< Math::Vector3D > Plane::getNormal (
    const Math::Point3D & point ) const [override], [virtual]
```

Implements [IPrimitives](#).

### 5.48.1.5 getType()

```
Type Plane::getType ( ) const [override], [virtual]
```

Implements [APrimitives](#).

### 5.48.1.6 setRotation()

```
void Plane::setRotation (
    const Math::Rot3D & newRotation ) [override], [virtual]
```

Reimplemented from [APrimitives](#).

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

- [src/primitives/Plane.hpp](#)
- [src/primitives/Plane.cpp](#)

## 5.49 Math::Point3D Class Reference

### Public Member Functions

- **Point3D** (double x, double y, double z)
- double **getX** () const
- double **getY** () const
- double **getZ** () const
- void **setX** (double x)
- void **setY** (double y)
- void **setZ** (double z)
- **Point3D normalize** () const
- **Point3D operator+** (const **Point3D** &other) const
- **Point3D operator-** (const **Point3D** &other) const
- **Point3D operator+** (const Vector3D &vector) const
- **Point3D operator-** (const Vector3D &vector) const
- **Point3D operator\*** (const Vector3D &vector) const
- **Point3D operator/** (const Vector3D &vector) const
- double **dot** (const **Point3D** &other) const
- double **dot** (const Vector3D &vector) const
- **Point3D** (const Vector3D &vector)
- **Point3D & operator+=** (const Vector3D &vector)
- **Point3D & operator-=** (const Vector3D &vector)

### Private Attributes

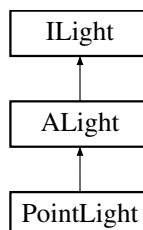
- double **x**
- double **y**
- double **z**

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

- common/Point3D.hpp
- common/Point3D.cpp

## 5.50 PointLight Class Reference

Inheritance diagram for PointLight:



## Public Member Functions

- **PointLight** (const [Color](#) &color, float intensity, const [Math::Point3D](#) &position, float radius)
- TypeLight [getTypeLight](#) () const override
- void [addLight](#) ([PixelInfo](#) &pixelInfo, const [Math::Ray](#) &ray) const override
- [Math::Point3D](#) [getPosition](#) () const
- float [getRadius](#) () const
- void [setPosition](#) (const [Math::Point3D](#) &position)
- void [setRadius](#) (float radius)

## Public Member Functions inherited from [ALight](#)

- **ALight** (const [Color](#) &color, float intensity)
- virtual [Math::Vector3D](#) [getDirection](#) () const override=0
- [Color](#) [getColor](#) () const override
- float [getIntensity](#) () const override
- void [setColor](#) (const [Color](#) &color) override
- void [setIntensity](#) (float intensity) override

## Private Attributes

- [Math::Point3D](#) [\\_position](#)
- float [\\_radius](#)

## Additional Inherited Members

## Protected Attributes inherited from [ALight](#)

- [Color](#) [\\_color](#)
- float [\\_intensity](#)

## 5.50.1 Member Function Documentation

### 5.50.1.1 [addLight\(\)](#)

```
void PointLight::addLight (  
    PixelInfo & pixelInfo,  
    const Math::Ray & ray ) const    [override], [virtual]
```

Implements [ALight](#).

### 5.50.1.2 [getRadius\(\)](#)

```
float PointLight::getRadius ( ) const    [virtual]
```

Implements [ALight](#).



### 5.50.1.3 getTypeLight()

```
TypeLight PointLight::getTypeLight ( ) const [override], [virtual]
```

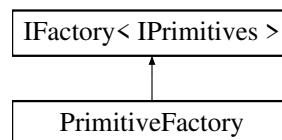
Implements [ALight](#).

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

- src/lights/PointLight.hpp
- src/lights/PointLight.cpp

## 5.51 PrimitiveFactory Class Reference

Inheritance diagram for PrimitiveFactory:



### Public Member Functions

- `std::shared_ptr< IPrimitives > create` (const std::string &type, std::shared\_ptr< std::map< ValueType\_t, ValueType > > config, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList, const std::vector< std::shared\_ptr< [ILight](#) > > &lights) override
- `std::shared_ptr< IPrimitives > createSimple` (const std::string &type, std::shared\_ptr< std::map< ValueType\_t, ValueType > > config, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList) override
- `void registerCreatorLight` (const std::string &type, std::function< std::shared\_ptr< [IPrimitives](#) >(std::shared\_ptr< std::map< ValueType\_t, ValueType > >, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &, const std::vector< std::shared\_ptr< [ILight](#) > > &)> creator) override
- `void registerCreator` (const std::string &type, std::function< std::shared\_ptr< [IPrimitives](#) >(std::shared\_ptr< std::map< ValueType\_t, ValueType > >, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &)> creator) override
- `bool loadPlugin` (const std::string &path) override
- `void loadAllPlugins` (const std::string &directory="plugins/")
- `ObjectType getTypeFromPlugin` (const std::string &path, [DLLoader](#)< void \* > loader)
- `std::string getNameFromPlugin` (const std::string &path, [DLLoader](#)< void \* > loader)
- `void setTexturePathIfNeeded` (std::shared\_ptr< [IPrimitives](#) > primitive, std::shared\_ptr< std::map< ValueType\_t, ValueType > > config)
- `std::shared_ptr< IMaterial > createMaterial` (const std::string &materialName)
- `std::shared_ptr< IMaterial > createMaterialByType` (MaterialType matType)

### Private Attributes

- `std::map< std::string, std::function< std::shared_ptr< IPrimitives >(std::shared_ptr< std::map< ValueType_t, ValueType > >, const std::vector< std::shared_ptr< std::map< ValueType_t, ValueType > > > &, const std::vector< std::shared_ptr< ILight > > &)> _creators` )
- `std::vector< DLLoader< void * > > _dlLoaders`
- `std::map< std::string, std::shared_ptr< IMaterial > > _materialList`

## 5.51.1 Member Function Documentation

### 5.51.1.1 create()

```
std::shared_ptr< IPrimitives > PrimitiveFactory::create (
    const std::string & type,
    std::shared_ptr< std::map< ValueType_t, ValueType > > config,
    const std::vector< std::shared_ptr< std::map< ValueType_t, ValueType > > > &
    graphSceneList,
    const std::vector< std::shared_ptr< ILight > > & lights ) [override], [virtual]
```

Implements [IFactory< IPrimitives >](#).

### 5.51.1.2 createSimple()

```
std::shared_ptr< IPrimitives > PrimitiveFactory::createSimple (
    const std::string & type,
    std::shared_ptr< std::map< ValueType_t, ValueType > > config,
    const std::vector< std::shared_ptr< std::map< ValueType_t, ValueType > > > &
    graphSceneList ) [override], [virtual]
```

Implements [IFactory< IPrimitives >](#).

### 5.51.1.3 loadPlugin()

```
bool PrimitiveFactory::loadPlugin (
    const std::string & path ) [override], [virtual]
```

Implements [IFactory< IPrimitives >](#).

### 5.51.1.4 registerCreator()

```
void PrimitiveFactory::registerCreator (
    const std::string & type,
    std::function< std::shared_ptr< IPrimitives >(std::shared_ptr< std::map< Value↵
Type_t, ValueType > >, const std::vector< std::shared_ptr< std::map< ValueType_t, ValueType
> > > &)> creator ) [override], [virtual]
```

Implements [IFactory< IPrimitives >](#).

### 5.51.1.5 registerCreatorLight()

```
void PrimitiveFactory::registerCreatorLight (
    const std::string & type,
    std::function< std::shared_ptr< IPrimitives >(std::shared_ptr< std::map< Value↵
Type_t, ValueType > >, const std::vector< std::shared_ptr< std::map< ValueType_t, ValueType
> > > &, const std::vector< std::shared_ptr< ILight > > &)> creator ) [override], [virtual]
```

Implements [IFactory< IPrimitives >](#).

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

- src/factory/PrimitiveFactory.hpp
- src/factory/PrimitiveFactory.cpp

## 5.52 PropertyConfig Struct Reference

### Public Attributes

- ValueType\_t **type**
- ValueFormat **format**
- std::vector< std::string > **components**
- ValueDataType **dataType**

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

- src/parser/PropertyTypes.hpp

## 5.53 PropertyInfo Struct Reference

### Public Attributes

- ValueType\_t **type**
- ValueFormat **format**
- std::vector< std::string > **components**
- ValueDataType **dataType**

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

- src/parser/PropertyTypes.hpp

## 5.54 Math::Random Class Reference

### Static Public Member Functions

- static double **normalDistribution** (uint32\_t &state)
- static float **pcg** (uint32\_t &state)

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

- common/Random.hpp
- common/Random.cpp

## 5.55 Math::Ray Class Reference

### Public Member Functions

- **Ray** ([Point3D](#) origin, Vector3D direction)
- **Ray** ([Point3D](#) origin, Vector3D direction, double Refraction\_index)
- void **setOrigin** ([Point3D](#) origin)
- void **setDirection** (Vector3D direction)
- void **setRefractionIndex** (double Refraction\_index)
- [Point3D](#) **getOrigin** () const
- Vector3D **getDirection** () const
- double **getRefractionIndex** () const

**Private Attributes**

- [Point3D](#) **\_origin**
- [Vector3D](#) **\_direction**
- double **\_refraction\_index**

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

- common/Ray.hpp

## 5.56 Ray Class Reference

**Public Member Functions**

- **Ray** ([Math::Point3D](#) origin, [Math::Vector3D](#) direction, [Color](#) color, double intensity)
- void **setOrigin** ([Math::Point3D](#) origin)
- void **setDirection** ([Math::Vector3D](#) direction)
- void **setColor** ([Color](#) color)
- void **setIntensity** (double intensity)
- [Math::Point3D](#) **getOrigin** () const
- [Math::Vector3D](#) **getDirection** () const
- [Color](#) **getColor** () const
- double **getIntensity** () const

**Private Attributes**

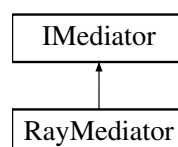
- [Math::Ray](#) **\_ray**
- [Color](#) **\_color**
- double **\_intensity**

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

- common/Ray.hpp
- common/Ray.cpp

## 5.57 RayMediator Class Reference

Inheritance diagram for RayMediator:



## Public Member Functions

- void [addTask](#) (std::function< void()> task) override
- void [executeTasks](#) () override
- void [waitForCompletion](#) () override

## Private Attributes

- std::vector< std::thread > **\_threads**
- std::vector< std::function< void()> > **\_tasks**
- std::mutex **\_mutex**
- std::condition\_variable **\_condition**
- bool **\_stop** = false

## 5.57.1 Member Function Documentation

### 5.57.1.1 addTask()

```
void RayMediator::addTask (
    std::function< void()> task ) [override], [virtual]
```

Implements [IMediator](#).

### 5.57.1.2 executeTasks()

```
void RayMediator::executeTasks ( ) [override], [virtual]
```

Implements [IMediator](#).

### 5.57.1.3 waitForCompletion()

```
void RayMediator::waitForCompletion ( ) [override], [virtual]
```

Implements [IMediator](#).

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

- src/mediator/RayMediator.hpp
- src/mediator/RayMediator.cpp

## 5.58 Raytracer Class Reference

### Public Member Functions

- `std::string` **getSceneFile** () const
- `std::string` **getOutputFile** () const
- `std::string` **getOutputFormat** () const
- `std::shared_ptr< Image >` **getImage** () const
- `bool` **getGraphicMode** () const
- `bool` **isDebug** () const
- `Scene` **getScene** () const
- `void` **setSceneFile** (std::string sceneFile)
- `void` **setOutputFile** (std::string outputFile)
- `void` **setOutputFormat** (std::string outputFormat)
- `void` **setImage** (std::shared\_ptr< Image > Image)
- `void` **setScene** (Scene scene)
- `bool` **setGraphicMode** ()
- `void` **writeToFilePPM** (std::string fileName)
- `void` **parseCmd** (int ac, char \*\*av)
- `void` **LoadAllformlibs** (const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &objectsConfig)
- `std::optional< PixelInfo >` **getClosestPrimitiveHit** (const Math::Ray &ray) const
- `Color` **TraceRay** (int x, int y, uint32\_t &state)
- `void` **InitParams** ()
- `void` **setScene** ()
- `void` **StartImage** ()
- `void` **printLoadingBar** (std::shared\_ptr< int > pixelCount, int totalPixels, int barWidth)
- `Color` **blendColors** (const Color &foreground, const Color &background, float transparency)
- `std::vector< InfoPixelDisplay >` **calculatePixel** (const Math::Ray &ray)
- `void` **averagelImages** (const std::vector< std::shared\_ptr< Image > > &images)
- `void` **loopThruType** ()
- `void` **initializeScene** ()
- `Color` **computePixelColor** (double u, double v, const Color &backgroundColor)
- `void` **displayGraphicMode** (std::shared\_ptr< int > pixelCount, int totalPixels)
- `void` **renderConsoleMode** (const Color &backgroundColor)
- `void` **renderGraphicMode** (int width, int height, std::shared\_ptr< int > pixelCount)
- `void` **finalizeRendering** ()
- `void` **generateDropShadows** ()
- `void` **setAntialiasingSamples** (int samples)
- `int` **getAntialiasingSamples** () const

### Private Attributes

- `bool` **graphicMode**
- `bool` **debugMode**
- `std::string` **\_sceneFile**
- `std::string` **\_outputFile**
- `std::string` **\_outputFormat**
- `int` **numRenders**
- `std::shared_ptr< Image >` **image**
- `Scene` **\_scene**
- `FactoryManager` **\_factoryManager**
- `std::shared_ptr< GraphicMode >` **\_display**

- int **\_width**
- int **\_height**
- int **\_antialiasingSamples** = 1

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

- src/Raytracer.hpp
- src/image/ConvertImage.cpp
- src/Raytracer.cpp
- src/RaytracerImage.cpp

## 5.59 Rectangle3D Class Reference

### Public Member Functions

- **Rectangle3D** (const [Math::Point3D](#) &origin, const [Math::Vector3D](#) &bottom\_side, const [Math::Vector3D](#) &left\_side)
- **Rectangle3D** (const [Math::Point3D](#) &point, const [Math::Rot3D](#) &rotation, double width=1, double height=1)
- int **getWidth** () const
- int **getHeight** () const
- [Math::Point3D](#) **pointAt** (double u, double v) const

### Public Attributes

- [Math::Point3D](#) **\_origin**
- [Math::Vector3D](#) **\_bottom\_side**
- [Math::Vector3D](#) **\_left\_side**

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

- common/Rectangle3D.hpp
- common/Rectangle3D.cpp

## 5.60 Math::Rot3D Class Reference

### Public Member Functions

- **Rot3D** (double x\_pitch=0, double z\_yaw=0, double y\_roll=0)
- **Rot3D** (const [Rot3D](#) &other)
- [Vector3D](#) **toVector** () const
- [Rot3D](#) **operator+** (const [Rot3D](#) &other) const
- [Rot3D](#) & **operator+=** (const [Rot3D](#) &other)
- [Rot3D](#) **operator-** (const [Rot3D](#) &other) const
- [Rot3D](#) & **operator-=** (const [Rot3D](#) &other)
- [Rot3D](#) **operator-** () const
- [Vector3D](#) **toUnitVector** () const
- [Vector3D](#) **rotate** (const [Vector3D](#) &vec) const
- [Vector3D](#) **inverseRotate** (const [Vector3D](#) &vec) const
- [Rot3D](#) & **operator=** (const [Rot3D](#) &other)
- double **dot** (const [Rot3D](#) &other) const
- double **getX** () const
- double **getY** () const
- double **getZ** () const

### Public Attributes

- double **x\_pitch**
- double **z\_yaw**
- double **y\_roll**

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

- common/Rot3D.hpp
- common/Rot3D.cpp

## 5.61 Scene Class Reference

### Public Member Functions

- **Scene** (std::shared\_ptr< [Camera](#) >, std::vector< std::shared\_ptr< [IPrimitives](#) > >)
- std::shared\_ptr< [Camera](#) > **getCamera** () const
- std::shared\_ptr< [GraphsNodePrimitive](#) > **getPrimitives** () const
- std::shared\_ptr< [GraphsNodeLight](#) > **getLights** () const
- int **camereaWidth** () const
- int **camereaHeight** () const
- float **getAmbientLight** () const
- void **setCamera** (std::shared\_ptr< [Camera](#) > camera)
- void **setPrimitives** (const std::vector< std::shared\_ptr< [IPrimitives](#) > > &primitives)
- void **setLights** (const std::vector< std::shared\_ptr< [ILight](#) > > &lights)

### Private Attributes

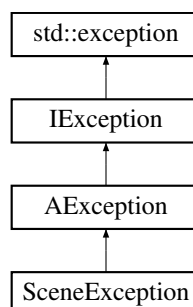
- std::shared\_ptr< [Camera](#) > **\_camera**
- std::vector< std::shared\_ptr< [IPrimitives](#) > > **\_primTemp**
- std::shared\_ptr< [GraphsNodePrimitive](#) > **\_primitives**
- std::shared\_ptr< [GraphsNodeLight](#) > **\_lights**
- float **\_ambientLight**

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

- common/Scene.hpp
- common/Scene.cpp

## 5.62 SceneException Class Reference

Inheritance diagram for SceneException:





### Public Member Functions

- **SceneException** (const std::string &message)

### Public Member Functions inherited from [AException](#)

- **AException** (const std::string &type, const std::string &message)
- const char \* [what](#) () const noexcept override
- std::string [getType](#) () const noexcept override
- std::string [getMessage](#) () const noexcept override
- std::string [getFormattedMessage](#) () const noexcept override

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

- common/Exception/SceneException.hpp

## 5.63 ShapeDefinition Struct Reference

### Public Attributes

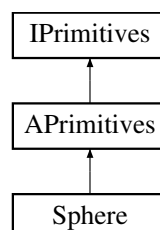
- std::string **name**
- std::vector< std::string > **mandatory**
- std::vector< std::string > **optional**
- ObjectType **objectType**

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

- src/parser/PropertyTypes.hpp

## 5.64 Sphere Class Reference

Inheritance diagram for Sphere:



## Public Member Functions

- **Sphere** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList, const std::vector< std::shared\_ptr< [ILight](#) > > &lights)
- std::optional< double > [distance](#) (const [Math::Ray](#) &ray) const override
- [PixellInfo distanceInfo](#) (const [Math::Ray](#) &ray) override
- std::optional< [Math::Point3D](#) > [getIntersection](#) (const [Math::Ray](#) &ray) const override
- std::optional< [Math::Vector3D](#) > [getNormal](#) (const [Math::Point3D](#) &point) const override
- Type [getType](#) () const override

## Public Member Functions inherited from [APrimitives](#)

- **APrimitives** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList, std::vector< std::shared\_ptr< [ILight](#) > > light)
- [Math::Ray computeScaledRay](#) (const [Math::Ray](#) &ray) const override
- [Math::Point3D getPosition](#) () const override
- [Math::Rot3D getRotation](#) () const override
- float [getScale](#) () const override
- std::shared\_ptr< [IMaterial](#) > [getMaterial](#) () const override
- [Math::Vector3D getInvScales](#) () const
- const std::vector< std::shared\_ptr< [ILight](#) > > & [getLights](#) () const
- void [setPosition](#) (const [Math::Point3D](#) &newPosition) override
- void [setRotation](#) (const [Math::Rot3D](#) &newRotation) override
- void [setScale](#) (const float newScale) override
- void [setMaterial](#) (std::shared\_ptr< [IMaterial](#) > newMaterial) override
- void [addLight](#) (std::shared\_ptr< [ILight](#) > light)
- void [clearLights](#) ()
- void [applyLights](#) ([PixellInfo](#) &pixellInfo, const [Math::Ray](#) &ray) const
- void [getPos](#) (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void [getRot](#) (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void [getCol](#) (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void [getScales](#) (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void [getGraph](#) (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList)
- void [computeInvScales](#) ()
- void [getGraphScale](#) (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)
- void [getPosGraph](#) (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)
- void [getRotGraph](#) (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)

## Private Attributes

- double [\\_distance](#)
- double [radius](#)

## Additional Inherited Members

### Protected Attributes inherited from [APrimitives](#)

- [Math::Point3D](#) **\_position**
- [Math::Rot3D](#) **\_rotation**
- float **scale**
- std::shared\_ptr< [IMaterial](#) > **material**
- [Color](#) **\_color**
- [Math::Vector3D](#) **\_scales**
- [Math::Vector3D](#) **\_invScales**
- std::string **\_graph**
- std::vector< std::shared\_ptr< [ILight](#) > > **\_lights**

## 5.64.1 Member Function Documentation

### 5.64.1.1 distance()

```
std::optional< double > Sphere::distance (
    const Math::Ray & ray ) const [override], [virtual]
```

Implements [APrimitives](#).

### 5.64.1.2 distanceInfo()

```
PixelInfo Sphere::distanceInfo (
    const Math::Ray & ray ) [override], [virtual]
```

Implements [APrimitives](#).

### 5.64.1.3 getIntersection()

```
std::optional< Math::Point3D > Sphere::getIntersection (
    const Math::Ray & ray ) const [override], [virtual]
```

Implements [IPrimitives](#).

### 5.64.1.4 getNormal()

```
std::optional< Math::Vector3D > Sphere::getNormal (
    const Math::Point3D & point ) const [override], [virtual]
```

Implements [IPrimitives](#).

### 5.64.1.5 getType()

```
Type Sphere::getType ( ) const [override], [virtual]
```

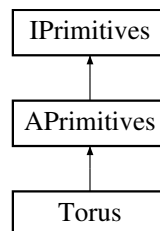
Implements [APrimitives](#).

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

- src/primitives/Sphere.hpp
- src/primitives/Sphere.cpp

## 5.65 Torus Class Reference

Inheritance diagram for Torus:



### Public Member Functions

- **Torus** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList, const std::vector< std::shared\_ptr< [ILight](#) > > &lights)
- **Torus** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList)
- void **setMajorRadius** (float radius)
- void **setMinorRadius** (float radius)
- float **getMajorRadius** () const
- float **getMinorRadius** () const
- Type **getType** () const override
- std::optional< double > **distance** (const [Math::Ray](#) &ray) const override
- [PixelInfo](#) **distanceInfo** (const [Math::Ray](#) &ray) override
- std::optional< [Math::Point3D](#) > **getIntersection** (const [Math::Ray](#) &ray) const override
- std::optional< [Math::Vector3D](#) > **getNormal** (const [Math::Point3D](#) &point) const override

## Public Member Functions inherited from **APrimitives**

- **APrimitives** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList, std::vector< std::shared\_ptr< [ILight](#) > > light)
- [Math::Ray](#) **computeScaledRay** (const [Math::Ray](#) &ray) const override
- [Math::Point3D](#) **getPosition** () const override
- [Math::Rot3D](#) **getRotation** () const override
- float **getScale** () const override
- std::shared\_ptr< [IMaterial](#) > **getMaterial** () const override
- [Math::Vector3D](#) **getInvScales** () const
- const std::vector< std::shared\_ptr< [ILight](#) > > & **getLights** () const
- void **setPosition** (const [Math::Point3D](#) &newPosition) override
- void **setRotation** (const [Math::Rot3D](#) &newRotation) override
- void **setScale** (const float newScale) override
- void **setMaterial** (std::shared\_ptr< [IMaterial](#) > newMaterial) override
- void **addLight** (std::shared\_ptr< [ILight](#) > light)
- void **clearLights** ()
- void **applyLights** ([PixellInfo](#) &pixellInfo, const [Math::Ray](#) &ray) const
- void **getPos** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getRot** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getCol** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getScales** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map)
- void **getGraph** (std::shared\_ptr< std::map< ValueType\_t, ValueType > > map, const std::vector< std::shared\_ptr< std::map< ValueType\_t, ValueType > > > &graphSceneList)
- void **computeInvScales** ()
- void **getGraphScale** (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)
- void **getPosGraph** (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)
- void **getRotGraph** (const std::shared\_ptr< std::map< ValueType\_t, ValueType > > &graph)

## Private Member Functions

- std::tuple< double, double, double, double, double > **computeQuarticCoefficients** (const [Math::Vector3D](#) &localOrigin, const [Math::Vector3D](#) &localDir) const
- double **evaluateQuartic** (double t, double a, double b, double c, double d\_coef, double e) const
- std::vector< double > **findRootCandidates** (double a, double b, double c, double d\_coef, double e) const
- double **refineRoot** (double t, double a, double b, double c, double d\_coef, double e) const

## Private Attributes

- float **\_majorRadius**
- float **\_minorRadius**
- double **\_distance**

## Additional Inherited Members

## Protected Attributes inherited from **APrimitives**

- [Math::Point3D](#) **\_position**
- [Math::Rot3D](#) **\_rotation**
- float **scale**
- std::shared\_ptr< [IMaterial](#) > **material**
- [Color](#) **\_color**
- [Math::Vector3D](#) **\_scales**
- [Math::Vector3D](#) **\_invScales**
- std::string **\_graph**
- std::vector< std::shared\_ptr< [ILight](#) > > **\_lights**

### 5.65.1 Member Function Documentation

#### 5.65.1.1 distance()

```
std::optional< double > Torus::distance (
    const Math::Ray & ray ) const [override], [virtual]
```

Implements [APrimitives](#).

#### 5.65.1.2 distanceInfo()

```
PixelInfo Torus::distanceInfo (
    const Math::Ray & ray ) [override], [virtual]
```

Implements [APrimitives](#).

#### 5.65.1.3 getIntersection()

```
std::optional< Math::Point3D > Torus::getIntersection (
    const Math::Ray & ray ) const [override], [virtual]
```

Implements [IPrimitives](#).

#### 5.65.1.4 getNormal()

```
std::optional< Math::Vector3D > Torus::getNormal (
    const Math::Point3D & point ) const [override], [virtual]
```

Implements [IPrimitives](#).

#### 5.65.1.5 getType()

```
Type Torus::getType ( ) const [override], [virtual]
```

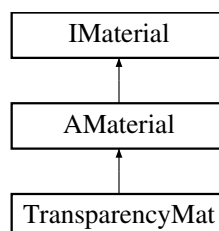
Implements [APrimitives](#).

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

- src/primitives/Torus.hpp
- src/primitives/Torus.cpp

## 5.66 TransparencyMat Class Reference

Inheritance diagram for TransparencyMat:



## Public Member Functions

- [Color applyMaterial](#) (const [PixellInfo](#) &pixellInfo, float radius, float height, const [IPrimitives](#) &primitive) const override

## Public Member Functions inherited from [AMaterial](#)

- void [setAmbient](#) (const Math::Vector3D &a) override
- void [setDiffuse](#) (const Math::Vector3D &d) override
- void [setSpecular](#) (const Math::Vector3D &s) override
- void [setShininess](#) (float s) override
- void [setReflectivity](#) (float r) override
- void [setTransparency](#) (float t) override
- void [setRefractiveIndex](#) (float i) override
- void [setOpacity](#) (float o) override
- void [setColorTexture](#) (const std::shared\_ptr< std::string > &texture) override
- void [setNormalMap](#) (const std::shared\_ptr< std::string > &map) override
- void [setOptionalColor1](#) (const [Color](#) &color) override
- void [setOptionalColor2](#) (const [Color](#) &color) override
- void [setScale](#) (float s) override
- void [setMaterialType](#) (MaterialType type) override
- Math::Vector3D [getAmbient](#) () const override
- Math::Vector3D [getDiffuse](#) () const override
- Math::Vector3D [getSpecular](#) () const override
- float [getShininess](#) () const override
- float [getReflectivity](#) () const override
- float [getTransparency](#) () const override
- float [getRefractiveIndex](#) () const override
- float [getOpacity](#) () const override
- std::shared\_ptr< std::string > [getColorTexture](#) () const override
- std::shared\_ptr< std::string > [getNormalMap](#) () const override
- [Color](#) [getOptionalColor1](#) () const override
- [Color](#) [getOptionalColor2](#) () const override
- float [getScale](#) () const override
- MaterialType [getMaterialType](#) () const override

## Additional Inherited Members

## Public Attributes inherited from [AMaterial](#)

- Math::Vector3D **ambient**
- Math::Vector3D **diffuse**
- Math::Vector3D **specular**
- float **shininess**
- [Color](#) **OptionalColor1** = [Color](#)(0.0f, 0.0f, 0.0f)
- [Color](#) **OptionalColor2** = [Color](#)(0.0f, 0.0f, 0.0f)
- float **scale** = 5.0f
- float **reflectivity**
- float **transparency**
- float **refractiveIndex**
- float **opacity**
- std::shared\_ptr< std::string > **colorTexture**
- std::shared\_ptr< std::string > **normalMap**
- MaterialType **materialType** = MaterialType::FLAT\_COLOR

## 5.66.1 Member Function Documentation

### 5.66.1.1 `applyMaterial()`

```
Color TransparencyMat::applyMaterial (
    const PixelInfo & pixelInfo,
    float radius,
    float height,
    const IPrimitives & primitive ) const [override], [virtual]
```

Implements [AMaterial](#).

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

- common/material/transparencyMat.hpp
- common/material/transparencyMat.cpp

## 5.67 Utils Class Reference

### Static Public Member Functions

- static void **helper** ()

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

- src/Utils/Utils.hpp
- src/Utils/Utils.cpp

## 5.68 ValueConverter Class Reference

### Static Public Member Functions

- static float **getFloatFromVariant** (const ValueType &value)
- static [Math::Vector2D](#) **getVector2DFromComponents** (const ValueType &x, const ValueType &y)
- static [Math::Vector3D](#) **getVector3DFromComponents** (const ValueType &x, const ValueType &y, const ValueType &z)

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

- src/parser/ValueConverter.hpp
- src/parser/ValueConverter.cpp



## 5.69 Math::Vector2D Class Reference

### Public Member Functions

- **Vector2D** (double x, double y)
- double **getX** () const
- double **getY** () const
- void **setX** (double x)
- void **setY** (double y)

### Private Attributes

- double **\_x**
- double **\_y**

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

- common/Vector2D.hpp
- common/Vector2D.cpp



# Chapter 6

## File Documentation

### 6.1 ALight.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** ALight
00006 */
00007
00008 #ifndef ALIGHT_HPP_
00009 #define ALIGHT_HPP_
00010
00011 #include "ILight.hpp"
00012 #include "ValueType.hpp"
00013 #include "Point3D.hpp"
00014
00015 class ALight : public ILight {
00016     public:
00017         ALight();
00018         ALight(const Color &color, float intensity);
00019         ~ALight() override;
00020
00021         /* Virtual methods */
00022         virtual void addLight(PixelInfo &pixelInfo, const Math::Ray &ray) const = 0;
00023         virtual TypeLight getTypeLight() const override = 0;
00024         virtual Math::Vector3D getDirection() const override = 0;
00025         virtual float getRadius() const override = 0;
00026
00027         /* Getter */
00028         Color getColor() const override;
00029         float getIntensity() const override;
00030
00031         /* Setter */
00032         void setColor(const Color &color) override;
00033         void setIntensity(float intensity) override;
00034
00035     protected:
00036         Color _color;
00037         float _intensity;
00038 };
00039
00040 #endif /* !ALIGHT_HPP_ */
```

### 6.2 APrimitives.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** APrimitives
00006 */
00007
00008 #include <memory>
00009 #include <map>
00010 #include <string>
00011 #include <vector>
```

```

00012
00013 #include "IPrimitives.hpp"
00014 #include "ValueType.hpp"
00015 #include "material/IMaterial.hpp"
00016 #include "ILight.hpp"
00017
00018
00019 #ifndef APRIMITIVES_HPP_
00020 #define APRIMITIVES_HPP_
00021
00022 class APrimitives : public IPrimitives {
00023 public:
00024     APrimitives() : _position(0, 0, 0), scale(1), material(nullptr),
00025         _color(-1, -1, -1), _scales(1, 1, 1), _invScales(1, 1, 1), _graph("") {}
00026     APrimitives(std::shared_ptr<std::map<ValueType_t, ValueType> map,
00027         const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>
00028             &graphSceneList, std::vector<std::shared_ptr<ILight> light);
00029     virtual ~APrimitives() = default;
00030
00031     /*Method */
00032     virtual PixelInfo distanceInfo(const Math::Ray &ray) = 0;
00033     virtual std::optional<double> distance(const Math::Ray &ray) const = 0;
00034     virtual Type getType() const override = 0;
00035     Math::Ray computeScaledRay(const Math::Ray &ray) const override;
00036
00037     /* Getter */
00038     Math::Point3D getPosition() const override { return _position; }
00039     Math::Rot3D getRotation() const override { return _rotation; }
00040     float getScale() const override { return scale; }
00041     std::shared_ptr<IMaterial> getMaterial() const override { return material; }
00042     Math::Vector3D getInvScales() const { return _invScales; }
00043     const std::vector<std::shared_ptr<ILight>& getLights() const { return _lights; }
00044
00045     /* Setter */
00046     void setPosition(const Math::Point3D &newPosition) override { _position = newPosition; }
00047     void setRotation(const Math::Rot3D &newRotation) override { _rotation = newRotation; }
00048     void setScale(const float newScale) override {
00049         scale = newScale;
00050         computeInvScales();
00051     }
00052     void setMaterial(std::shared_ptr<IMaterial> newMaterial) override { material = newMaterial; }
00053     /* Light Management */
00054     void addLight(std::shared_ptr<ILight> light) { _lights.push_back(std::move(light)); }
00055     void clearLights() { _lights.clear(); }
00056     void applyLights(PixelInfo &pixelInfo, const Math::Ray &ray) const;
00057
00058     /* Sub Functions */
00059     void getPos(std::shared_ptr<std::map<ValueType_t,
00060         ValueType> map);
00061     void getRot(std::shared_ptr<std::map<ValueType_t,
00062         ValueType> map);
00063     void getCol(std::shared_ptr<std::map<ValueType_t,
00064         ValueType> map);
00065     void getScales(std::shared_ptr<std::map<ValueType_t,
00066         ValueType> map);
00067     void getGraph(std::shared_ptr<std::map<ValueType_t,
00068         ValueType> map, const std::vector<std::shared_ptr<std::map<ValueType_t,
00069         ValueType>> &graphSceneList);
00070     void computeInvScales();
00071
00072     /* Graph Method */
00073     void getGraphScale(const std::shared_ptr<std::map<ValueType_t, ValueType> &graph);
00074     void getPosGraph(const std::shared_ptr<std::map<ValueType_t,
00075         ValueType> &graph);
00076     void getRotGraph(const std::shared_ptr<std::map<ValueType_t,
00077         ValueType> &graph);
00078
00079 protected:
00080     Math::Point3D _position;
00081     Math::Rot3D _rotation;
00082     float scale;
00083     std::shared_ptr<IMaterial> material;
00084     Color _color;
00085     Math::Vector3D _scales;
00086     Math::Vector3D _invScales;
00087     std::string _graph;
00088     std::vector<std::shared_ptr<ILight> _lights;
00089 };
00090
00091 #endif /* !APRIMITIVES_HPP_ */

```

## 6.3 Camera.hpp

```

00001 /*

```

```

00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** ICamera
00006 */
00007
00008 #include "Point3D.hpp"
00009 #include "Vector3D.hpp"
00010 #include "Rectangle3D.hpp"
00011 #include "Ray.hpp"
00012
00013 #ifndef ICAMERA_HPP_
00014 #define ICAMERA_HPP_
00015
00016 class Camera {
00017     public:
00018         Camera();
00019         Camera(const Camera&) = default;
00020         Camera(const Rectangle3D& screen);
00021         ~Camera();
00022
00023         Math::Point3D _origin;
00024         Rectangle3D _screen;
00025
00026         Camera& operator=(const Camera&) = default;
00027
00028         void updateScreen();
00029         Math::Ray ray(double u, double v) const;
00030
00031         /* Getter */
00032         Math::Point3D getOrigin() const { return _origin; }
00033         Math::Point3D getRotation() const { return _rotation; }
00034         Rectangle3D getScreen() const { return _screen; }
00035         Math::Vector3D getPosition() const { return _position; }
00036         int getWidth() const { return width; }
00037         int getHeight() const { return height; }
00038         float getFieldOfView() const { return fieldOfView; }
00039         /* Setter */
00040
00041         void setRotation(Math::Point3D rotation) { _rotation = rotation; }
00042         void setPosition(Math::Vector3D position) { _position = position; }
00043         void setResolution(int x, int y) { width = x; height = y; }
00044         void setHeight(int h) { height = h; }
00045         void setFieldOfView(float fov) { fieldOfView = fov; }
00046
00047     protected:
00048         int width;
00049         int height;
00050         float fieldOfView;
00051         Math::Vector3D _rotation;
00052         Math::Vector3D _position;
00053
00054     private:
00055 };
00056
00057 #endif /* !ICAMERA_HPP_ */

```

## 6.4 Color.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** Color
00006 */
00007
00008 #ifndef COLOR_HPP_
00009 #define COLOR_HPP_
00010
00011 #include <cstdint>
00012 #include "Vector3D.hpp"
00013 #include "Exception/ColorException.hpp"
00014
00015 class Color {
00016     public:
00017         Color() {
00018             this->_red = 0;
00019             this->_green = 0;
00020             this->_blue = 0;
00021             this->_transparency = 255;
00022         }
00023         ~Color() = default;
00024

```

```

00025     Color(uint8_t red, uint8_t green, uint8_t blue) {
00026         this->_red = red;
00027         this->_green = green;
00028         this->_blue = blue;
00029         this->_transparency = 255;
00030     }
00031
00032     Color(Math::Vector3D vec) {
00033         vec.setX(std::min(255.0, std::max(0.0, vec.getX())));
00034         vec.setY(std::min(255.0, std::max(0.0, vec.getY())));
00035         vec.setZ(std::min(255.0, std::max(0.0, vec.getZ())));
00036         this->_red = static_cast<uint8_t>(vec.getX());
00037         this->_green = static_cast<uint8_t>(vec.getY());
00038         this->_blue = static_cast<uint8_t>(vec.getZ());
00039         this->_transparency = 255;
00040     }
00041
00042     Color(uint8_t red, uint8_t green, uint8_t blue, uint8_t transparency) {
00043         this->_red = red;
00044         this->_green = green;
00045         this->_blue = blue;
00046         this->_transparency = transparency;
00047     }
00048
00049     Color& operator=(const Math::Vector3D &vec) {
00050         if (vec.getX() < 0.0 || vec.getX() > 255.0 ||
00051             vec.getY() < 0.0 || vec.getY() > 255.0 ||
00052             vec.getZ() < 0.0 || vec.getZ() > 255.0) {
00053             throw ColorException("Color values must be between 0 and 255");
00054         }
00055
00056         this->_red = static_cast<uint8_t>(vec.getX());
00057         this->_green = static_cast<uint8_t>(vec.getY());
00058         this->_blue = static_cast<uint8_t>(vec.getZ());
00059         return *this;
00060     }
00061
00062     Color operator*(float scalar) const {
00063         uint8_t red = static_cast<uint8_t>(std::min(255.0f, std::max(0.0f, static_cast<float>(_red) *
00064 scalar)));
00065         uint8_t green = static_cast<uint8_t>(std::min(255.0f, std::max(0.0f,
00066 static_cast<float>(_green) * scalar)));
00067         uint8_t blue = static_cast<uint8_t>(std::min(255.0f, std::max(0.0f, static_cast<float>(_blue)
00068 * scalar)));
00069         return Color(red, green, blue, _transparency);
00070     }
00071
00072     Color operator*(const Color& other) const {
00073         uint8_t red = static_cast<uint8_t>(std::min(255.0f, (static_cast<float>(_red) * other._red) /
00074 255.0f));
00075         uint8_t green = static_cast<uint8_t>(std::min(255.0f, (static_cast<float>(_green) *
00076 other._green) / 255.0f));
00077         uint8_t blue = static_cast<uint8_t>(std::min(255.0f, (static_cast<float>(_blue) * other._blue)
00078 / 255.0f));
00079         uint8_t transparency = static_cast<uint8_t>(std::min(255.0f,
00080 (static_cast<float>(_transparency) * other._transparency) / 255.0f));
00081         return Color(red, green, blue, transparency);
00082     }
00083
00084     Color operator+=(float scalar) {
00085         this->_red = static_cast<uint8_t>(std::min(255.0f, std::max(0.0f, static_cast<float>(_red) *
00086 scalar)));
00087         this->_green = static_cast<uint8_t>(std::min(255.0f, std::max(0.0f, static_cast<float>(_green)
00088 * scalar)));
00089         this->_blue = static_cast<uint8_t>(std::min(255.0f, std::max(0.0f, static_cast<float>(_blue) *
00090 scalar)));
00091         return *this;
00092     }
00093
00094     Color operator+=(const Color& other) {
00095         this->_red = static_cast<uint8_t>(std::min(255.0f, (static_cast<float>(_red) * other._red) /
00096 255.0f));
00097         this->_green = static_cast<uint8_t>(std::min(255.0f, (static_cast<float>(_green) *
00098 other._green) / 255.0f));
00099         this->_blue = static_cast<uint8_t>(std::min(255.0f, (static_cast<float>(_blue) * other._blue)
00100 / 255.0f));
00101         this->_transparency = static_cast<uint8_t>(std::min(255.0f, (static_cast<float>(_transparency)
00102 * other._transparency) / 255.0f));
00103         return *this;
00104     }
00105
00106     Color operator+(const Color& other) const {
00107         uint8_t red = static_cast<uint8_t>(std::min(255, static_cast<int>(_red) +
00108 static_cast<int>(other._red)));
00109         uint8_t green = static_cast<uint8_t>(std::min(255, static_cast<int>(_green) +
00110 static_cast<int>(other._green)));
00111         uint8_t blue = static_cast<uint8_t>(std::min(255, static_cast<int>(_blue) +

```

```

        static_cast<int>(other._blue)));
00096         uint8_t transparency = static_cast<uint8_t>(std::min(255, static_cast<int>(_transparency) +
        static_cast<int>(other._transparency)));
00097         return Color(red, green, blue, transparency);
00098     }
00099
00100     Color operator+=(const Color& other) {
00101         this->_red= static_cast<uint8_t>(std::min(255, static_cast<int>(_red) +
        static_cast<int>(other._red)));
00102         this->_green = static_cast<uint8_t>(std::min(255, static_cast<int>(_green) +
        static_cast<int>(other._green)));
00103         this->_blue = static_cast<uint8_t>(std::min(255, static_cast<int>(_blue) +
        static_cast<int>(other._blue)));
00104         this->_transparency = static_cast<uint8_t>(std::min(255, static_cast<int>(_transparency) +
        static_cast<int>(other._transparency)));
00105         return *this;
00106     }
00107
00108     Color operator-(const Color& other) const {
00109         uint8_t red = static_cast<uint8_t>(std::max(0, static_cast<int>(_red) -
        static_cast<int>(other._red)));
00110         uint8_t green = static_cast<uint8_t>(std::max(0, static_cast<int>(_green) -
        static_cast<int>(other._green)));
00111         uint8_t blue = static_cast<uint8_t>(std::max(0, static_cast<int>(_blue) -
        static_cast<int>(other._blue)));
00112         uint8_t transparency = static_cast<uint8_t>(std::max(0, static_cast<int>(_transparency) -
        static_cast<int>(other._transparency)));
00113         return Color(red, green, blue, transparency);
00114     }
00115
00116     Color operator-=(const Color& other) {
00117         this->_red = static_cast<uint8_t>(std::max(0, static_cast<int>(_red) -
        static_cast<int>(other._red)));
00118         this->_green = static_cast<uint8_t>(std::max(0, static_cast<int>(_green) -
        static_cast<int>(other._green)));
00119         this->_blue = static_cast<uint8_t>(std::max(0, static_cast<int>(_blue) -
        static_cast<int>(other._blue)));
00120         this->_transparency = static_cast<uint8_t>(std::max(0, static_cast<int>(_transparency) -
        static_cast<int>(other._transparency)));
00121         return *this;
00122     }
00123
00124
00125     void setTransparency(float transparencyValue) {
00126         this->_transparency = static_cast<uint8_t>(255 * (1.0f - transparencyValue));
00127     }
00128
00129     /* Getter */
00130     uint8_t getRed() const { return this->_red; }
00131     uint8_t getGreen() const { return this->_green; }
00132     uint8_t getBlue() const { return this->_blue; }
00133     uint8_t getTransparency() const { return this->_transparency; }
00134
00135     /* Setter */
00136     void setRed(uint8_t red) { this->_red = red; }
00137     void setGreen(uint8_t green) { this->_green = green; }
00138     void setBlue(uint8_t blue) { this->_blue = blue; }
00139
00140 protected:
00141 private:
00142     uint8_t _red;
00143     uint8_t _green;
00144     uint8_t _blue;
00145
00146     uint8_t _transparency;
00147 };
00148
00149 #endif /* !COLOR_HPP_ */

```

## 6.5 Error.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** Error
00006 */
00007
00008 #ifndef ERROR_HPP_
00009 #define ERROR_HPP_
00010
00011 #include <exception>
00012 #include <iostream>

```

```

00013 #include <string>
00014
00015 class Error : public std::exception {
00016 public:
00017     Error(const std::string &msg, const std::string &file, int line)
00018         : _message(msg), _file(file), _line(line) {
00019         ErrorFaillureException();
00020     }
00021
00022     ~Error() noexcept override = default;
00023
00024     const char *what() const noexcept override { return _message.c_str(); }
00025
00026     const char *where() const noexcept { return _file.c_str(); }
00027
00028     int line() const noexcept { return _line; }
00029
00030     void ErrorFaillureException() const {
00031         std::cerr << "Error: " << what() << " in file: " << where()
00032             << " at line: " << line() << std::endl;
00033         exit(84);
00034     }
00035     void exitCode(int code) { exit(code); }
00036
00037 private:
00038     std::string _message;
00039     std::string _file;
00040     int _line;
00041 };
00042
00043 #endif /* !ERROR_HPP_ */

```

## 6.6 AException.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** AException
00006 */
00007
00008 #ifndef AEXCEPTION_HPP_
00009 #define AEXCEPTION_HPP_
00010
00011 #include "IException.hpp"
00012 #include <string>
00013
00014 class AException : public IException {
00015 public:
00016     AException(const std::string& type, const std::string& message)
00017         : _message(message), _type(type) {}
00018     virtual ~AException() noexcept = default;
00019
00020     const char* what() const noexcept override {
00021         return getFormattedMessage().c_str();
00022     }
00023
00024     std::string getType() const noexcept override {
00025         return _type;
00026     }
00027
00028     std::string getMessage() const noexcept override {
00029         return _message;
00030     }
00031
00032     std::string getFormattedMessage() const noexcept override {
00033         return "[" + _type + "] " + _message;
00034     }
00035
00036 private:
00037     std::string _message;
00038     std::string _type;
00039 };
00040
00041 #endif /* !AEXCEPTION_HPP_ */

```

## 6.7 ColorException.hpp

```

00001 #ifndef COLOREXCEPTION_HPP_

```



```

00002 #define COLOREXCEPTION_HPP_
00003
00004 #include "AException.hpp"
00005
00006 class ColorException : public AException {
00007     public:
00008         ColorException(const std::string &message)
00009             : AException("ColorError", message) {}
00010 };
00011
00012 #endif /* !COLOREXCEPTION_HPP_ */

```

## 6.8 CommandException.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** CommandException
00006 */
00007
00008 #ifndef COMMANDEXCEPTION_HPP_
00009 #define COMMANDEXCEPTION_HPP_
00010
00011 #include "AException.hpp"
00012
00013 class CommandException : public AException {
00014     public:
00015         CommandException(const std::string& message)
00016             : AException("CommandError", message) {};
00017     protected:
00018     private:
00019 };
00020
00021 #endif /* !COMMANDEXCEPTION_HPP_ */

```

## 6.9 FactoryException.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** FactoryException
00006 */
00007
00008 #ifndef FACTORYEXCEPTION_HPP_
00009 #define FACTORYEXCEPTION_HPP_
00010
00011 #include "AException.hpp"
00012
00013 class FactoryException : public AException {
00014     public:
00015         FactoryException(const std::string& message)
00016             : AException("FactoryError", message) {};
00017     protected:
00018     private:
00019 };
00020
00021 #endif /* !FACTORYEXCEPTION_HPP_ */

```

## 6.10 FileException.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** FileConstructionException
00006 */
00007
00008 #ifndef FILECONSTRUCTIONEXEPTION_HPP_
00009 #define FILECONSTRUCTIONEXEPTION_HPP_
00010
00011 #include "AException.hpp"
00012
00013 class FileException : public AException {

```

```

00014     public:
00015         FileException(const std::string& message)
00016             : AException("FileError", message) {};
00017     protected:
00018     private:
00019 };
00020
00021 #endif /* !FILECONSTRUCTIONEXCEPTION_HPP_ */

```

## 6.11 IException.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** IException
00006 */
00007
00008 #ifndef IEXCEPTION_HPP_
00009 #define IEXCEPTION_HPP_
00010
00011 #include <exception>
00012 #include <string>
00013
00014
00015 class IException : public std::exception {
00016     public:
00017         virtual ~IException() noexcept = default;
00018         const char* what() const noexcept override = 0;
00019         virtual std::string getType() const noexcept = 0;
00020         virtual std::string getMessage() const noexcept = 0;
00021         virtual std::string getFormattedMessage() const noexcept = 0;
00022 };
00023
00024 #endif /* !IEXCEPTION_HPP_ */

```

## 6.12 materialLoaderException.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** materialLoaderException
00006 */
00007
00008 #ifndef MATERIALLOADEREXCEPTION_HPP_
00009 #define MATERIALLOADEREXCEPTION_HPP_
00010
00011 #include "AException.hpp"
00012
00013 class MaterialLoaderException : public AException {
00014     public:
00015         MaterialLoaderException(const std::string& message)
00016             : AException("MaterialLoaderError", message) {};
00017     protected:
00018     private:
00019 };
00020
00021 #endif /* !MATERIALLOADEREXCEPTION_HPP_ */

```

## 6.13 MathExeption.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** MathExeption
00006 */
00007
00008 #ifndef MATHEXCEPTION_HPP_
00009 #define MATHEXCEPTION_HPP_
00010
00011 #include "AException.hpp"
00012
00013 class MathExeption : public AException {

```

```

00014     public:
00015         MathException(const std::string& message)
00016             : AException("MathError", message) {};
00017
00018     protected:
00019     private:
00020 };
00021
00022 #endif /* !MATHEXCEPTION_HPP_ */

```

## 6.14 SceneException.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** SceneException
00006 */
00007
00008 #ifndef SCENEEXCEPTION_HPP_
00009 #define SCENEEXCEPTION_HPP_
00010
00011 #include "AException.hpp"
00012
00013 class SceneException : public AException {
00014     public:
00015         SceneException(const std::string &message)
00016             : AException("SceneError", message) {}
00017
00018     protected:
00019     private:
00020 };
00021
00022 #endif /* !SCENEEXCEPTION_HPP_ */

```

## 6.15 Graphs.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** Graphs
00006 */
00007
00008 #ifndef GRAPHS_HPP_
00009 #define GRAPHS_HPP_
00010
00011 #include "IPrimitives.hpp"
00012 #include "ILight.hpp"
00013
00014 struct GraphsNodePrimitive {
00015     std::shared_ptr<IPrimitives> _primitives;
00016     std::vector<std::shared_ptr<GraphsNodePrimitive>> _children;
00017
00018     GraphsNodePrimitive& operator=(const GraphsNodePrimitive& other) {
00019         if (this == &other)
00020             return *this;
00021         _primitives = other._primitives;
00022         _children.clear();
00023         for (const auto& child : other._children) {
00024             if (child)
00025                 _children.push_back(std::make_shared<GraphsNodePrimitive>(*child));
00026             else
00027                 _children.push_back(nullptr);
00028         }
00029         return *this;
00030     }
00031
00032     template<typename Func>
00033     void traverseGraph(const std::shared_ptr<GraphsNodePrimitive>& node, Func &&func) {
00034         if (!node) return;
00035         if (node->_primitives)
00036             func(node->_primitives);
00037         for (const auto& child : node->_children)
00038             traverseGraph(child, func);
00039     };
00040
00041
00042 struct GraphsNodeLight {

```

```

00043     std::shared_ptr<ILight> _primitives;
00044     std::vector<std::shared_ptr<GraphsNodeLight>> _children;
00045
00046     GraphsNodeLight &operator=(const GraphsNodeLight& other) {
00047         if (this == &other)
00048             return *this;
00049         _primitives = other._primitives;
00050         _children.clear();
00051         for (const auto& child : other._children) {
00052             if (child)
00053                 _children.push_back(std::make_shared<GraphsNodeLight>(*child));
00054             else
00055                 _children.push_back(nullptr);
00056         }
00057         return *this;
00058     }
00059     template<typename Func>
00060     void traverseGraph(const std::shared_ptr<GraphsNodeLight>& node, Func &&func) {
00061         if (!node) return;
00062         if (node->_primitives)
00063             func(node->_primitives);
00064         for (const auto& child : node->_children)
00065             traverseGraph(child, func);
00066     }
00067 };
00068
00069
00070
00071 #endif /* !GRAPHS_HPP_ */

```

## 6.16 IGraphicMode.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** Raytracer
00004 ** File description:
00005 ** Header
00006 */
00007
00008 #include <memory>
00009 #include <string>
00010 #include <map>
00011 #include <vector>
00012 #include <SFML/Graphics.hpp>
00013
00014 #ifndef IGRAPHICMODE_HPP_
00015     #define IGRAPHICMODE_HPP_
00016
00017     class IGraphicMode {
00018     public:
00019         virtual ~IGraphicMode() = default;
00020
00021         virtual void createText(const std::string &text, int size, int x, int y) = 0;
00022         virtual void createRectangle(const std::string &id, int x, int y, int width, int height) = 0;
00023
00024         virtual bool getRenderingComplete() const = 0;
00025         virtual void setWindow(int width, int height) = 0;
00026         virtual void setRenderingComplete(bool renderingComplete) = 0;
00027
00028         virtual std::string getButtonPressed() = 0;
00029         virtual void updateTexture() = 0;
00030         virtual void drawPixelColor(int x, int y, uint8_t r, uint8_t g, uint8_t b) = 0;
00031         virtual void drawImage() = 0;
00032         virtual void drawButtons() = 0;
00033         virtual void display() = 0;
00034         virtual bool isOpen() = 0;
00035         virtual void closeWindow() = 0;
00036     };
00037
00038 #endif /* !IGRAPHICMODE_HPP_ */

```

## 6.17 ILight.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** ILight
00006 */

```

```

00007
00008 #include <memory>
00009 #include "Color.hpp"
00010 #include "PixelInfo.hpp"
00011 #include "Ray.hpp"
00012 #include "Vector3D.hpp"
00013
00014 #ifndef ILIGHT_HPP_
00015 #define ILIGHT_HPP_
00016
00017
00018 enum class TypeLight { POINT, DIRECTIONAL, PHONG, AMBIENT };
00019
00020 class ILight {
00021     public:
00022         virtual ~ILight() = default;
00023
00024         virtual void addLight(PixelInfo &pixelInfo, const Math::Ray &ray) const = 0;
00025
00026         // Getters
00027         virtual Color getColor() const = 0;
00028         virtual Math::Vector3D getDirection() const = 0;
00029         virtual float getIntensity() const = 0;
00030         virtual float getRadius() const = 0;
00031         virtual TypeLight getTypeLight() const = 0;
00032
00033         // Setters
00034         virtual void setColor(const Color &color) = 0;
00035         virtual void setIntensity(float intensity) = 0;
00036 };
00037
00038 #endif /* !ILIGHT_HPP_ */

```

## 6.18 Image.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** Raytracer
00006 */
00007
00008 #include <fstream>
00009 #include <iostream>
00010 #include <vector>
00011
00012 #include "../common/Error.hpp"
00013 #include "Color.hpp"
00014
00015 #ifndef IMAGE_HPP_
00016 #define IMAGE_HPP_
00017
00018 class Image {
00019     public:
00020         Image(int width, int height, int maxColorValue = 255);
00021         ~Image() = default;
00022
00023         int getWidth() const { return width; }
00024
00025         int getHeight() const { return height; }
00026
00027         int getMaxColorValue() const { return maxColorValue; }
00028
00029         const std::vector<Color> &getData() const { return data; }
00030
00031         void writeToPPM(const std::string &fileName) const;
00032         void setPixel(int x, int y, const Color &color);
00033         Color getPixel(int x, int y) const;
00034
00035     private:
00036         int width;
00037         int height;
00038         int maxColorValue;
00039         std::vector<Color> data;
00040 };
00041
00042 #endif /* !IMAGE_HPP_ */

```

## 6.19 IPrimitives.hpp

```

00001 /*

```

```

00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** IPrimitives
00006 */
00007
00008 #include <memory>
00009 #include <optional>
00010
00011 #include "material/IMaterial.hpp"
00012 #include "Point3D.hpp"
00013 #include "Ray.hpp"
00014 #include "Rot3D.hpp"
00015 #include "PixelInfo.hpp"
00016
00017
00018 #ifndef IPRIMITIVES_HPP_
00019 #define IPRIMITIVES_HPP_
00020
00021 enum class Type { SPHERE, PLANE, CYLINDER, CONE, CUBE, TORUS };
00022
00023 class IPrimitives {
00024 public:
00025     virtual ~IPrimitives() = default;
00026
00027     virtual std::optional<double> distance(const Math::Ray &ray) const = 0;
00028     virtual PixelInfo distanceInfo(const Math::Ray &ray) = 0;
00029     virtual std::optional<Math::Point3D> getIntersection(const Math::Ray &ray) const = 0;
00030     virtual std::optional<Math::Vector3D> getNormal(const Math::Point3D &point) const = 0;
00031     virtual Math::Ray computeScaledRay(const Math::Ray &ray) const = 0;
00032
00033     /* Getter */
00034     virtual Type getType() const = 0;
00035     virtual Math::Point3D getPosition() const = 0;
00036     virtual Math::Rot3D getRotation() const = 0;
00037     virtual float getScale() const = 0;
00038     virtual std::shared_ptr<IMaterial> getMaterial() const = 0;
00039
00040     /* Setter */
00041     virtual void setPosition(const Math::Point3D &position) = 0;
00042     virtual void setRotation(const Math::Rot3D &rotation) = 0;
00043     virtual void setScale(float scale) = 0;
00044     virtual void setMaterial(std::shared_ptr<IMaterial> material) = 0;
00045 protected:
00046 private:
00047 };
00048
00049 #endif /* !IPRIMITIVES_HPP_ */

```

## 6.20 Material.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** Material
00006 */
00007
00008 #ifndef MATERIAL_HPP_
00009 #define MATERIAL_HPP_
00010 #include "Vector3D.hpp"
00011 #include "Color.hpp"
00012
00013 enum MaterialType {
00014     FLAT_COLOR,
00015     TRANSPARENCY_MAT,
00016     CHESSBOARD,
00017     FILE_TEXTURE_MAT,
00018     PERLING_NOISE_MAT
00019 };
00020
00021 struct Material {
00022     // Basic colors/components (Vec3: r,g,b in [0,1])
00023     Math::Vector3D ambient; // Ambient reflection coefficient
00024     Math::Vector3D diffuse; // Diffuse (Lambertian) reflection coefficient
00025     Math::Vector3D specular; // Specular reflection coefficient
00026     float shininess; // Phong exponent (higher = smaller, sharper highlights)
00027
00028     Color OptionalColor1 = Color(0.0f, 0.0f, 0.0f); // Color for the first chessboard square
00029     Color OptionalColor2 = Color(0.0f, 0.0f, 0.0f); // Color for the second chessboard square
00030     float scale = 5.0f; // Scale for the chessboard pattern
00031
00032     // Reflection & Refraction

```

```

00033     float reflectivity;      // [0,1] Fraction of reflection ray contribution
00034     float transparency;     // [0,1] Fraction of refracted ray contribution
00035     float refractiveIndex;   // Snell's law index (1.0 = vacuum/air)
00036     float opacity;          // [0,1] Fraction of light that is absorbed by the material
00037
00038     // Optional textures (nullptr if unused)
00039     std::shared_ptr<std::string> colorTexture; // Overrides diffuse when present
00040     std::shared_ptr<std::string> normalMap;    // Perturbs surface normals for bump mapping
00041
00042     MaterialType materialType = FLAT_COLOR;
00043
00044     // Constructors
00045     Material()
00046         : ambient(0.0f, 0.0f, 0.0f)
00047         , diffuse(0.0f, 0.0f, 0.0f)
00048         , specular(0.0f, 0.0f, 0.0f)
00049         , shininess(32.0f)
00050         , reflectivity(0.0f)
00051         , transparency(0.0f)
00052         , refractiveIndex(1.0f)
00053         , opacity(1.0f)
00054         , colorTexture(nullptr)
00055         , normalMap(nullptr)
00056     {}
00057
00058     Material(const Math::Vector3D& a, const Math::Vector3D& d, const Math::Vector3D& s, float shin,
00059             float refl = 0.0f, float trans = 0.0f, float ior = 1.0f)
00060         : ambient(a)
00061         , diffuse(d)
00062         , specular(s)
00063         , shininess(shin)
00064         , reflectivity(refl)
00065         , transparency(trans)
00066         , refractiveIndex(ior)
00067         , opacity(1.0f)
00068         , colorTexture(nullptr)
00069         , normalMap(nullptr)
00070     {}
00071 };
00072
00073
00074 #endif /* !MATERIAL_HPP_ */

```

## 6.21 AMaterial.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** AMaterial
00006 */
00007
00008 #ifndef AMATERIAL_HPP_
00009 #define AMATERIAL_HPP_
00010 #include "IMaterial.hpp"
00011 #include "../Vector3D.hpp"
00012 #include "../Color.hpp"
00013 #include <memory>
00014
00015 class AMaterial : public IMaterial {
00016     public:
00017         AMaterial();
00018         virtual ~AMaterial() = default;
00019
00020         void setAmbient(const Math::Vector3D& a) override { ambient = a; }
00021         void setDiffuse(const Math::Vector3D& d) override { diffuse = d; }
00022         void setSpecular(const Math::Vector3D& s) override { specular = s; }
00023         void setShininess(float s) override { shininess = s; }
00024         void setReflectivity(float r) override { reflectivity = r; }
00025         void setTransparency(float t) override { transparency = t; }
00026         void setRefractiveIndex(float i) override { refractiveIndex = i; }
00027         void setOpacity(float o) override { opacity = o; }
00028         void setColorTexture(const std::shared_ptr<std::string>& texture) override { colorTexture =
00029             texture; }
00030         void setNormalMap(const std::shared_ptr<std::string>& map) override { normalMap = map; }
00031         void setOptionalColor1(const Color& color) override { OptionalColor1 = color; }
00032         void setOptionalColor2(const Color& color) override { OptionalColor2 = color; }
00033         void setScale(float s) override { scale = s; }
00034         void setMaterialType(MaterialType type) override { materialType = type; }
00035
00036         Math::Vector3D getAmbient() const override { return ambient; }
00037         Math::Vector3D getDiffuse() const override { return diffuse; }
00038         Math::Vector3D getSpecular() const override { return specular; }

```

```

00038         float getShininess() const override { return shininess; }
00039         float getReflectivity() const override { return reflectivity; }
00040         float getTransparency() const override { return transparency; }
00041         float getRefractiveIndex() const override { return refractiveIndex; }
00042         float getOpacity() const override { return opacity; }
00043         std::shared_ptr<std::string> getColorTexture() const override { return colorTexture; }
00044         std::shared_ptr<std::string> getNormalMap() const override { return normalMap; }
00045         Color getOptionalColor1() const override { return OptionalColor1; }
00046         Color getOptionalColor2() const override { return OptionalColor2; }
00047         float getScale() const override { return scale; }
00048         MaterialType getMaterialType() const override { return materialType; }
00049
00050         virtual Color applyMaterial(const PixelInfo& pixelInfo, float radius, float height, const
IPrimitives &primitive) const = 0;
00051         Math::Vector3D ambient; // Ambient reflection coefficient
00052         Math::Vector3D diffuse; // Diffuse (Lambertian) reflection coefficient
00053         Math::Vector3D specular; // Specular reflection coefficient
00054         float shininess; // Phong exponent (higher = smaller, sharper highlights)
00055
00056         Color OptionalColor1 = Color(0.0f, 0.0f, 0.0f); // Color for the first chessboard square
00057         Color OptionalColor2 = Color(0.0f, 0.0f, 0.0f); // Color for the second chessboard square
00058         float scale = 5.0f; // Scale for the chessboard pattern
00059
00060         // Reflection & Refraction
00061         float reflectivity; // [0,1] Fraction of reflection ray contribution
00062         float transparency; // [0,1] Fraction of refracted ray contribution
00063         float refractiveIndex; // Snell's law index (1.0 = vacuum/air)
00064         float opacity; // [0,1] Fraction of light that is absorbed by the material
00065
00066         // Optional textures (nullptr if unused)
00067         std::shared_ptr<std::string> colorTexture; // Overrides diffuse when present
00068         std::shared_ptr<std::string> normalMap; // Perturbs surface normals for bump mapping
00069         MaterialType materialType = MaterialType::FLAT_COLOR;
00070     };
00071
00072 #endif /* !AMATERIAL_HPP_ */

```

## 6.22 chessboardMat.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** chessboardMat
00006 */
00007
00008 #ifndef CHESSBOARDMAT_HPP_
00009 #define CHESSBOARDMAT_HPP_
00010 #include "AMaterial.hpp"
00011 #include "../IPrimitives.hpp"
00012 #include "../PixelInfo.hpp"
00013
00014 class ChessboardMat : public AMaterial {
00015     public:
00016         ChessboardMat();
00017         ~ChessboardMat() override = default;
00018         Color applyMaterial(const PixelInfo& pixelInfo, float radius, float height, const IPrimitives
&primitive) const override;
00019
00020     private:
00021         Color applySphereChessboard(const PixelInfo& pixelInfo, float radius, float height, const
IPrimitives &primitive) const;
00022         Color applyPlaneChessboard(const PixelInfo& pixelInfo, float radius, float height, const
IPrimitives &primitive) const;
00023         Color applyCylinderChessboard(const PixelInfo& pixelInfo, float radius, float height, const
IPrimitives &primitive) const;
00024         Color applyConeChessboard(const PixelInfo& pixelInfo, float radius, float height, const
IPrimitives &primitive) const;
00025 };
00026
00027 #endif /* !CHESSBOARDMAT_HPP_ */

```

## 6.23 fileTextureMat.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** fileTextureMat

```



```

00006 */
00007
00008 #ifndef FILETEXTUREMAT_HPP_
00009 #define FILETEXTUREMAT_HPP_
00010
00011 #include "AMaterial.hpp"
00012
00013 class FileTextureMat : public AMaterial {
00014     public:
00015         FileTextureMat();
00016         ~FileTextureMat() override = default;
00017
00018         Color applyMaterial(const PixelInfo& pixelInfo, float radius, float height, const IPrimitives
&primitive) const override;
00019
00020     private:
00021         void loadTextureFromFile(const std::string& filePath);
00022         Color getTextureFromFile(const PixelInfo& pixelInfo, std::shared_ptr<float> u,
std::shared_ptr<float> v) const;
00023         void calculUVCoordinates(const IPrimitives &primitive, const PixelInfo& pixelInfo,
float radius, float height, std::shared_ptr<float> u, std::shared_ptr<float> v) const;
00024         void calculUVCoordinatesSphere(const IPrimitives &primitive, const PixelInfo& pixelInfo,
float radius, std::shared_ptr<float> u, std::shared_ptr<float> v) const;
00025         void calculUVCoordinatesCylinder(const IPrimitives &primitive, const PixelInfo& pixelInfo,
float radius, float height, std::shared_ptr<float> u, std::shared_ptr<float> v) const;
00026         void calculUVCoordinatesCone(const IPrimitives &primitive, const PixelInfo& pixelInfo,
float radius, float height, std::shared_ptr<float> u, std::shared_ptr<float> v) const;
00027         void calculUVCoordinatesPlane(const IPrimitives &primitive, const PixelInfo& pixelInfo,
float radius, float height, std::shared_ptr<float> u, std::shared_ptr<float> v) const;
00028         void calculUVCoordinatesSphere(const IPrimitives &primitive, const PixelInfo& pixelInfo,
float radius, float height, std::shared_ptr<float> u, std::shared_ptr<float> v) const;
00029         void calculUVCoordinatesCone(const IPrimitives &primitive, const PixelInfo& pixelInfo,
float radius, float height, std::shared_ptr<float> u, std::shared_ptr<float> v) const;
00030         void calculUVCoordinatesPlane(const IPrimitives &primitive, const PixelInfo& pixelInfo,
float radius, float height, std::shared_ptr<float> u, std::shared_ptr<float> v) const;
00031 };
00032
00033 #endif /* !FILETEXTUREMAT_HPP_ */

```

## 6.24 flatColorMat.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** flatColorMat
00006 */
00007
00008 #ifndef FLATCOLORMAT_HPP_
00009 #define FLATCOLORMAT_HPP_
00010 #include "AMaterial.hpp"
00011 #include "../IPrimitives.hpp"
00012 #include "../PixelInfo.hpp"
00013
00014 class FlatColorMat : public AMaterial {
00015     public:
00016         FlatColorMat();
00017         ~FlatColorMat() override = default;
00018
00019         Color applyMaterial(const PixelInfo& pixelInfo, float radius, float height, const IPrimitives
&primitive) const override;
00020 };
00021
00022 #endif /* !FLATCOLORMAT_HPP_ */

```

## 6.25 IMaterial.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** IMaterial
00006 */
00007
00008 #ifndef IMATERIAL_HPP_
00009 #define IMATERIAL_HPP_
00010
00011 #include "../Vector3D.hpp"
00012 #include "../Color.hpp"
00013 #include "../PixelInfo.hpp"
00014 #include <memory>
00015
00016 // Forward declaration pour éviter les dépendances circulaires
00017 class IPrimitives;
00018

```

```

00019 enum MaterialType {
00020     FLAT_COLOR,
00021     TRANSPARENCY_MAT,
00022     CHESSBOARD,
00023     FILE_TEXTURE_MAT,
00024     PERLING_NOISE_MAT
00025 };
00026
00027 class IMaterial {
00028 public:
00029     virtual ~IMaterial() = default;
00030
00031     // Setters
00032     virtual void setAmbient(const Math::Vector3D& a) = 0;
00033     virtual void setDiffuse(const Math::Vector3D& d) = 0;
00034     virtual void setSpecular(const Math::Vector3D& s) = 0;
00035     virtual void setShininess(float s) = 0;
00036     virtual void setReflectivity(float r) = 0;
00037     virtual void setTransparency(float t) = 0;
00038     virtual void setRefractiveIndex(float i) = 0;
00039     virtual void setOpacity(float o) = 0;
00040     virtual void setColorTexture(const std::shared_ptr<std::string>& texture) = 0;
00041     virtual void setNormalMap(const std::shared_ptr<std::string>& map) = 0;
00042     virtual void setOptionalColor1(const Color& color) = 0;
00043     virtual void setOptionalColor2(const Color& color) = 0;
00044     virtual void setScale(float s) = 0;
00045     virtual void setMaterialType(MaterialType type) = 0;
00046
00047     // Getters
00048     virtual Math::Vector3D getAmbient() const = 0;
00049     virtual Math::Vector3D getDiffuse() const = 0;
00050     virtual Math::Vector3D getSpecular() const = 0;
00051     virtual float getShininess() const = 0;
00052     virtual float getReflectivity() const = 0;
00053     virtual float getTransparency() const = 0;
00054     virtual float getRefractiveIndex() const = 0;
00055     virtual float getOpacity() const = 0;
00056     virtual std::shared_ptr<std::string> getColorTexture() const = 0;
00057     virtual std::shared_ptr<std::string> getNormalMap() const = 0;
00058     virtual Color getOptionalColor1() const = 0;
00059     virtual Color getOptionalColor2() const = 0;
00060     virtual float getScale() const = 0;
00061     virtual MaterialType getMaterialType() const = 0;
00062
00063     // Material application methods
00064     virtual Color applyMaterial(const PixelInfo& pixelInfo, float radius, float height, const
IPrimitives &primitive) const = 0;
00065 };
00066
00067 #endif /* !IMATERIAL_HPP_ */

```

## 6.26 perlingNoiseMat.hpp

```

00001 /*
00002  ** EPITECH PROJECT, 2025
00003  ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004  ** File description:
00005  ** perlingNoiseMat
00006  */
00007
00008 #ifndef PERLINGNOISEMAT_HPP_
00009 #define PERLINGNOISEMAT_HPP_
00010
00011 #include "AMaterial.hpp"
00012 #include <cmath>
00013 #include <random>
00014 #include <vector>
00015
00016 class PerlingNoiseMat : public AMaterial {
00017 public:
00018     PerlingNoiseMat();
00019     ~PerlingNoiseMat() override = default;
00020
00021     Color applyMaterial(const PixelInfo& pixelInfo, float radius, float height, const IPrimitives
&primitive) const override;
00022
00023 private:
00024     float frequency;
00025     float amplitude;
00026     int octaves;
00027     float persistence;
00028 };
00029
00030 #endif /* !PERLINGNOISEMAT_HPP_ */

```

## 6.27 transparencyMat.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** transparencyMat
00006 */
00007
00008 #ifndef TRANSPARENCYMAT_HPP_
00009 #define TRANSPARENCYMAT_HPP_
00010
00011 #include "AMaterial.hpp"
00012
00013 class TransparencyMat : public AMaterial {
00014     public:
00015         TransparencyMat();
00016         ~TransparencyMat() override = default;
00017
00018         Color applyMaterial(const PixelInfo& pixelInfo, float radius, float height, const IPrimitives
00019 &primitive) const override;
00019 };
00020 #endif /* !TRANSPARENCYMAT_HPP_ */

```

## 6.28 PixelInfo.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** PixelInfo
00006 */
00007
00008 #ifndef PIXELINFO_HPP_
00009 #define PIXELINFO_HPP_
00010
00011 #include "Color.hpp"
00012 #include "Vector3D.hpp"
00013
00014 struct PixelInfo {
00015     Color _color;
00016     Math::Vector3D _normalizedVector;
00017     double _distance;
00018     bool _isHit;
00019     Math::Vector3D _pos;
00020     float _light_intensity;
00021     Color _light_color;
00022
00023     PixelInfo() :
00024         _color(),
00025         _normalizedVector(),
00026         _distance(0.0),
00027         _isHit(false),
00028         _pos(),
00029         _light_intensity(0.0f),
00030         _light_color()
00031     {}
00032
00033     PixelInfo(const Color& color, const Math::Vector3D& normalVector, double distance,
00034 bool isHit, const Math::Vector3D& position, float lightIntensity, const Color&
00035 colorLight) :
00036         _color(color),
00037         _normalizedVector(normalVector),
00038         _distance(distance),
00039         _isHit(isHit),
00040         _pos(position),
00041         _light_intensity(lightIntensity),
00042         _light_color(colorLight)
00043     {}
00044 };
00045 #endif /* !PIXELINFO_HPP_ */

```

## 6.29 Point3D.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:

```

```

00005 ** Point3D
00006 */
00007
00008 #ifndef POINT3D_HPP_
00009 #define POINT3D_HPP_
00010
00011 #include "Vector3D.hpp"
00012
00013 namespace Math {
00014
00015 class Point3D {
00016 public:
00017     Point3D();
00018     Point3D(double x, double y, double z);
00019     ~Point3D() = default;
00020
00021     /* Getter */
00022     double getX() const;
00023     double getY() const;
00024     double getZ() const;
00025
00026     /* Setter */
00027     void setX(double x);
00028     void setY(double y);
00029     void setZ(double z);
00030
00031     /* Methods */
00032     Point3D normalize() const;
00033     Point3D operator+(const Point3D &other) const;
00034     Point3D operator-(const Point3D &other) const;
00035     Point3D operator+(const Vector3D &vector) const;
00036     Point3D operator-(const Vector3D &vector) const;
00037     Point3D operator*(const Vector3D &vector) const;
00038     Point3D operator/(const Vector3D &vector) const;
00039     double dot(const Point3D &other) const;
00040     double dot(const Vector3D &vector) const;
00041     Point3D(const Vector3D &vector);
00042     /* Operators */
00043     Point3D &operator+=(const Vector3D &vector);
00044     Point3D &operator-=(const Vector3D &vector);
00045
00046 protected:
00047 private:
00048     double x;
00049     double y;
00050     double z;
00051 };
00052
00053 } // namespace Math
00054
00055 #endif /* !POINT3D_HPP_ */

```

## 6.30 Random.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** Random
00006 */
00007
00008 #ifndef RANDOM_HPP_
00009 #define RANDOM_HPP_
00010
00011 #include <cmath>
00012 #include <random>
00013 #include <functional>
00014
00015 namespace Math {
00016
00017 class Random {
00018 public:
00019     static double normalDistribution(uint32_t& state);
00020     static float pcg(uint32_t& state);
00021 };
00022
00023 } // namespace Math
00024
00025 #endif /* !RANDOM_HPP_ */

```

## 6.31 Ray.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** Ray
00006 */
00007
00008 #include "Color.hpp"
00009 #include "Point3D.hpp"
00010 #include "Vector3D.hpp"
00011
00012 #ifndef RAY_HPP_
00013 #define RAY_HPP_
00014
00015 namespace Math {
00016 class Ray {
00017 public:
00018     Ray() : _origin(), _direction() {}
00019     Ray(Point3D origin, Vector3D direction) : _origin(origin), _direction(direction),
00020     _refraction_index(1) {}
00021     Ray(Point3D origin, Vector3D direction, double Refraction_index) : _origin(origin),
00022     _direction(direction), _refraction_index( Refraction_index) {}
00023     ~Ray() = default;
00024
00025     /* Setter */
00026     void setOrigin(Point3D origin) { this->_origin = origin; }
00027     void setDirection(Vector3D direction) { this->_direction = direction; }
00028     void setRefractionIndex(double Refraction_index) { this->_refraction_index = Refraction_index; }
00029
00030     /* Getter */
00031     Point3D getOrigin() const { return this->_origin; }
00032     Vector3D getDirection() const { return this->_direction; }
00033     double getRefractionIndex() const { return this->_refraction_index; }
00034
00035 private:
00036     Point3D _origin;
00037     Vector3D _direction;
00038     double _refraction_index;
00039 };
00040 } // namespace Math
00041
00042 class Ray {
00043 public:
00044     Ray() : _ray(), _color(), _intensity(0) {}
00045     Ray(Math::Point3D origin, Math::Vector3D direction, Color color, double intensity)
00046     : _ray(origin, direction), _color(color), _intensity(intensity) {}
00047     ~Ray() = default;
00048
00049     /* Setter */
00050     void setOrigin(Math::Point3D origin);
00051     void setDirection(Math::Vector3D direction);
00052     void setColor(Color color);
00053     void setIntensity(double intensity);
00054
00055     /* Getter */
00056     Math::Point3D getOrigin() const;
00057     Math::Vector3D getDirection() const;
00058     Color getColor() const;
00059     double getIntensity() const;
00060
00061 private:
00062     Math::Ray _ray;
00063     Color _color;
00064     double _intensity;
00065 };
00066 #endif /* !RAY_HPP_ */

```

## 6.32 Rectangle3D.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** Raytracer
00004 ** File description:
00005 ** Header
00006 */
00007
00008 #include "Point3D.hpp"
00009 #include "Rot3D.hpp"
00010 #include "Vector3D.hpp"
00011

```

```

00012 #ifndef RECTANGLE3D_HPP_
00013     #define RECTANGLE3D_HPP_
00014
00015 class Rectangle3D {
00016     public:
00017         Rectangle3D(const Math::Point3D& origin, const Math::Vector3D& bottom_side, const
Math::Vector3D& left_side);
00018         Rectangle3D(const Math::Point3D &point, const Math::Rot3D &rotation, double width = 1, double
height = 1);
00019         ~Rectangle3D();
00020
00021         Math::Point3D _origin;
00022         Math::Vector3D _bottom_side;
00023         Math::Vector3D _left_side;
00024         int getWidth() const;
00025         int getHeight() const;
00026         Math::Point3D pointAt(double u, double v) const;
00027
00028     protected:
00029     private:
00030 };
00031
00032 #endif /* !RECTANGLE3D_HPP_ */

```

## 6.33 Rot3D.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** Raytracer
00006 */
00007
00008 #ifndef ROT3D_HPP
00009 #define ROT3D_HPP
00010
00011 #include "Vector3D.hpp"
00012
00013 namespace Math {
00014 class Rot3D {
00015     public:
00016         double x_pitch;
00017         double z_yaw;
00018         double y_roll;
00019
00020         Rot3D(double x_pitch = 0, double z_yaw = 0, double y_roll = 0);
00021         Rot3D(const Rot3D &other);
00022         Vector3D toVector() const;
00023
00024         Rot3D operator+(const Rot3D &other) const;
00025         Rot3D &operator+=(const Rot3D &other);
00026         Rot3D operator-(const Rot3D &other) const;
00027         Rot3D &operator-=(const Rot3D &other);
00028         Rot3D operator-() const;
00029         Vector3D toUnitVector() const;
00030         Vector3D rotate(const Vector3D &vec) const;
00031         Vector3D inverseRotate(const Vector3D &vec) const;
00032         Rot3D &operator=(const Rot3D &other);
00033
00034         double dot(const Rot3D &other) const;
00035
00036         double getX() const { return x_pitch; }
00037         double getY() const { return y_roll; }
00038         double getZ() const { return z_yaw; }
00039     };
00040 } // namespace Math
00041
00042 #endif // ROT3D_HPP

```

## 6.34 Scene.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** IScene
00006 */
00007
00008 #include <memory>

```

```

00009 #include <vector>
00010 #include "Camera.hpp"
00011 #include "APrimitives.hpp"
00012 #include "Graphs.hpp"
00013
00014 #ifndef ISCENE_HPP_
00015 #define ISCENE_HPP_
00016
00017 class Scene {
00018 public:
00019     Scene();
00020     Scene(std::shared_ptr<Camera>, std::vector<std::shared_ptr<IPrimitives>>);
00021     ~Scene();
00022
00023     /* Getter */
00024     std::shared_ptr<Camera> getCamera() const;
00025     std::shared_ptr<GraphsNodePrimitive> getPrimitives() const;
00026     std::shared_ptr<GraphsNodeLight> getLights() const;
00027     int cameraWidth() const;
00028     int cameraHeight() const;
00029     float getAmbientLight() const;
00030
00031     /* Setter */
00032     void setCamera(std::shared_ptr<Camera> camera);
00033     void setPrimitives(const std::vector<std::shared_ptr<IPrimitives>> &primitives);
00034     void setLights(const std::vector<std::shared_ptr<ILight>> &lights);
00035     // void setAmbientLight(float ambientLight);
00036
00037 private:
00038     std::shared_ptr<Camera> _camera;
00039     std::vector<std::shared_ptr<IPrimitives>> _primTemp;
00040     // std::vector<std::shared_ptr<IPrimitives>> _primitives;
00041     std::shared_ptr<GraphsNodePrimitive> _primitives;
00042     std::shared_ptr<GraphsNodeLight> _lights;
00043     // std::vector<std::shared_ptr<ILight>> _lights;
00044     float _ambientLight;
00045 };
00046
00047 #endif /* !ISCENE_HPP_ */

```

## 6.35 ValueType.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** ValueType
00006 */
00007
00008
00009 #include <memory>
00010 #include <string>
00011 #include <variant>
00012
00013 #include "IPrimitives.hpp"
00014 #include "Vector2D.hpp"
00015 #include "Vector3D.hpp"
00016
00017 #ifndef VALUETYPE_HPP_
00018 #define VALUETYPE_HPP_
00019
00020 enum ObjectType {
00021     TYPE_CAMERA,
00022     TYPE_LIGHT,
00023     TYPE_PRIMITIVE,
00024     TYPE_IMPORTED_SCENE,
00025     TYPE_GRAPH,
00026     TYPE_MATERIAL,
00027     TYPE_UNDEFINED
00028 };
00029
00030 using ValueType = std::variant<int, float, double, std::string, bool, Math::Vector2D, Math::Vector3D,
    ObjectType>;
00031
00032 enum ValueFormat { FORMAT_SIMPLE, FORMAT_VECTOR2D, FORMAT_VECTOR3D};
00033
00034 typedef enum ValueType_s {
00035     NAME,
00036     TYPE,
00037     POSITION,
00038     ROTATION,
00039     SCALE,
00040     SCALES,

```

```

00041     COLOR,
00042     COLOR_CHESS_1,
00043     COLOR_CHESS_2,
00044     RADIUS,
00045     FIELD_OF_VIEW,
00046     RESOLUTION,
00047     AXIS,
00048     HEIGHT,
00049     MATERIAL,
00050     GRAPH,
00051     PATH,
00052     AMBIENT,
00053     SPECULAR,
00054     DIFFUSE,
00055     REFLECTION,
00056     TRANSPARENCY,
00057     REFRACTION_INDEX,
00058     DIRECTION,
00059     INTENSITY,
00060     SHININESS,
00061     MAJOR_RADIUS,
00062     MINOR_RADIUS
00063 } ValueType_t;
00064
00065
00066
00067 #endif /* !VALUETYPE_HPP_ */

```

## 6.36 Vector2D.hpp

```

00001 /*
00002  ** EPITECH PROJECT, 2025
00003  ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004  ** File description:
00005  ** Vector2D
00006  */
00007
00008 #ifndef VECTOR2D_HPP_
00009 #define VECTOR2D_HPP_
00010
00011 namespace Math {
00012
00013 class Vector2D {
00014     public:
00015         Vector2D();
00016         Vector2D(double x, double y) : _x(x), _y(y) {}
00017         ~Vector2D() = default;
00018
00019         // getter
00020         double getX() const { return _x; }
00021         double getY() const { return _y; }
00022
00023         // setter
00024         void setX(double x) { _x = x; }
00025         void setY(double y) { _y = y; }
00026
00027     protected:
00028     private:
00029         double _x;
00030         double _y;
00031 };
00032
00033 } // namespace Math
00034
00035 #endif /* !VECTOR2D_HPP_ */

```

## 6.37 Vector3D.hpp

```

00001 /*
00002  ** EPITECH PROJECT, 2025
00003  ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004  ** File description:
00005  ** Vector3D
00006  */
00007
00008 #include <cmath>
00009 #include <iostream>
00010 #include <vector>
00011 #include <functional>

```



```

00012
00013 #include "Point3D.hpp"
00014 #include "Random.hpp"
00015
00016 #ifndef VECTOR3D_HPP_
00017 #define VECTOR3D_HPP_
00018
00019 class Color;
00020
00021 namespace Math {
00022
00023 class Point3D;
00024
00025 class Vector3D {
00026 public:
00027     Vector3D();
00028     Vector3D(uint32_t& state);
00029     Vector3D(double x, double y, double z);
00030     Vector3D(const Math::Point3D &point);
00031     ~Vector3D() = default;
00032
00033     /* Getter */
00034     double getX() const;
00035     double getY() const;
00036     double getZ() const;
00037
00038     /* Setter */
00039     void setX(double x);
00040     void setY(double y);
00041     void setZ(double z);
00042
00043     /* Methods */
00044     double length() const;
00045     double dot(const Vector3D &other) const;
00046
00047     // Add cross product
00048     Vector3D cross(const Vector3D &other) const;
00049
00050     Vector3D normalize() const;
00051
00052     Vector3D RandomInHemisphere(uint32_t& state) const;
00053
00054     Vector3D getAnyPerpendicular() const;
00055
00056     /* Operators */
00057     Vector3D operator-() const;
00058     Vector3D operator+(const Vector3D &other) const;
00059     Vector3D operator-(const Vector3D &other) const;
00060
00061     Vector3D operator+=(const Vector3D &other);
00062     Vector3D operator+=(const Color &other);
00063     Vector3D operator-=(const Vector3D &other);
00064
00065     Vector3D operator*(const Vector3D &other) const;
00066     Vector3D operator*=(const Vector3D &other);
00067
00068     Vector3D operator/(const Vector3D &other) const;
00069     Vector3D operator/=(const Vector3D &other);
00070
00071     /* Operator and Scalar */
00072     Vector3D operator*(double scalar) const;
00073     Vector3D &operator*=(double scalar);
00074     Vector3D operator/(double scalar) const;
00075     Vector3D &operator/=(double scalar);
00076
00077 protected:
00078     double x;
00079     double y;
00080     double z;
00081
00082 private:
00083 };
00084
00085 inline Vector3D operator*(double scalar, const Vector3D &vec) {
00086     return Vector3D(vec.getX() * scalar, vec.getY() * scalar,
00087         vec.getZ() * scalar);
00088 }
00089
00090 inline Vector3D operator/(double scalar, const Vector3D &vec) {
00091     return Vector3D(vec.getX() / scalar, vec.getY() / scalar,
00092         vec.getZ() / scalar);
00093 }
00094
00095 } // namespace Math
00096
00097 #endif /* !VECTOR3D_HPP_ */

```

## 6.38 DLoader.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** arcade
00004 ** File description:
00005 ** DLoader
00006 */
00007
00008 #ifndef DLOADER_HPP_
00009 #define DLOADER_HPP_
00010
00011 #include <dlfcn.h>
00012 #include <iostream>
00013 #include <ostream>
00014 #include "ILoader.hpp"
00015
00016 template <typename T>
00017
00018 class DLoader : public ILoader {
00019     private:
00020         void *_handler = nullptr;
00021
00022     public:
00023         ~DLoader() = default;
00024
00025         void *getHandler() const override {
00026             return _handler;
00027         };
00028         void *Open(const char *path, int flag) override {
00029             _handler = dlopen(path, flag);
00030             return _handler;
00031         };
00032         void *Symbol(const char *symbolName) override {
00033             void *symbol = dlsym(_handler, symbolName);
00034             const char *error = dlerror();
00035             if (error) {
00036                 std::cerr << "dlerror: " << error << std::endl;
00037                 return nullptr;
00038             }
00039             return symbol;
00040         };
00041         T getSymbol(const char *symbolName) {
00042             return reinterpret_cast<T>(dlsym(_handler, symbolName));
00043         };
00044         int Close() override {
00045             if (_handler == nullptr)
00046                 return -1;
00047             return dlclose(_handler);
00048         };
00049         const char *Error() override {
00050             return dlerror();
00051         };
00052 };
00053
00054 #endif /* !DLOADER_HPP_ */

```

## 6.39 ILoader.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** ILoader
00006 */
00007
00008 #ifndef ILoader_HPP_
00009 #define ILoader_HPP_
00010
00011
00012 class ILoader {
00013     public:
00014         ~ILoader() = default;
00015
00016         virtual void *Open(const char *path, int flag) = 0;
00017         virtual void *Symbol(const char *symbolName) = 0;
00018         virtual int Close() = 0;
00019         virtual const char *Error() = 0;
00020         virtual void *getHandler() const = 0;
00021
00022     protected:
00023     private:
00024 };

```

```

00025
00026 #endif /* !ILoader_HPP_ */

```

## 6.40 GraphicMode.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** Raytracer
00004 ** File description:
00005 ** Graphic Mode
00006 */
00007
00008 #include <memory>
00009 #include <SFML/Graphics.hpp>
00010 #include <SFML/Window.hpp>
00011 #include "../common/IGraphicMode.hpp"
00012
00013 #ifndef GRAPHICMODE_HPP_
00014     #define GRAPHICMODE_HPP_
00015
00016     class GraphicMode : public IGraphicMode {
00017     public:
00018         GraphicMode();
00019         ~GraphicMode();
00020
00021         void createText(const std::string &text, int size, int x, int y) override;
00022         void createRectangle(const std::string &id, int x, int y, int width, int height) override;
00023
00024         bool getRenderingComplete() const override;
00025         void setWindow(int width, int height) override;
00026         void setRenderingComplete(bool renderingComplete) override;
00027
00028         std::string getButtonPressed() override;
00029         void updateTexture() override;
00030         void drawPixelColor(int x, int y, uint8_t r, uint8_t g, uint8_t b) override;
00031         void drawImage() override;
00032         void drawButtons() override;
00033         void display() override;
00034         bool isOpen() override;
00035         void closeWindow() override;
00036
00037     private:
00038         sf::Event _event;
00039         std::shared_ptr<sf::RenderWindow> _window;
00040         std::shared_ptr<sf::Image> _image;
00041         std::map<std::string, sf::RectangleShape> _buttons;
00042         std::shared_ptr<sf::Font> _font;
00043         std::vector<sf::Text> _texts;
00044         sf::Texture _texture;
00045         std::string _title;
00046         int _width;
00047         int _height;
00048         bool _renderingComplete;
00049     };
00050
00051     extern "C" {
00052         std::shared_ptr<GraphicMode> createInstance();
00053     }
00054
00055 #endif /* !GRAPHICMODE_HPP_ */

```

## 6.41 CameraFactory.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** PrimitiveFactory
00006 */
00007
00008 #ifndef CAMERAFACTORY_HPP_
00009 #define CAMERAFACTORY_HPP_
00010
00011 #include <vector>
00012 #include <functional>
00013 #include "IFactory.hpp"
00014 #include "../lib/DLLoader.hpp"
00015 #include "../common/Camera.hpp"
00016 #include "../common/Exception/FactoryException.hpp"

```

```

00017
00018 class CameraFactory : public IFactory<Camera> {
00019 public:
00020     CameraFactory();
00021     ~CameraFactory();
00022
00023     std::shared_ptr<Camera> create(const std::string& type,
00024                                   std::shared_ptr<std::map<ValueType_t, ValueType> config,
00025                                   const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>& graphSceneList,
00026                                   const std::vector<std::shared_ptr<ILight>& lights) override;
00027
00028     std::shared_ptr<Camera> createSimple(const std::string& type,
00029                                          std::shared_ptr<std::map<ValueType_t, ValueType> config,
00030                                          const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>& graphSceneList) override;
00031
00032     void registerCreator(const std::string& type,
00033                         std::function<std::shared_ptr<Camera>(
00034                             std::shared_ptr<std::map<ValueType_t, ValueType>,
00035                             const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>&)> creator) override;
00036
00037     void registerCreatorLight(const std::string& type,
00038                              std::function<std::shared_ptr<Camera>(
00039                                  std::shared_ptr<std::map<ValueType_t, ValueType>,
00040                                  const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>&,
00041                                  const std::vector<std::shared_ptr<ILight>&)> creator) override;
00042
00043     bool loadPlugin(const std::string& path) override;
00044     void loadAllPlugins(const std::string& directory = "plugins/");
00045     ObjectType getTypeFromPlugin(const std::string& path, DLLoader<void*> loader);
00046     std::string getNameFromPlugin(const std::string& path, DLLoader<void*> loader);
00047
00048 private:
00049     std::map<std::string, std::function<std::shared_ptr<Camera>(
00050         std::shared_ptr<std::map<ValueType_t, ValueType>,
00051         const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>&)> _creators;
00052     std::vector<DLLoader<void*> _dlLoaders;
00053 };
00054
00055 #endif /* !CAMERAFACTORY_HPP_ */

```

## 6.42 FactoryManager.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** FactoryManager
00006 */
00007
00008 #ifndef FACTORYMANAGER_HPP_
00009 #define FACTORYMANAGER_HPP_
00010
00011 #include <memory>
00012 #include "PrimitiveFactory.hpp"
00013 #include "CameraFactory.hpp"
00014 #include "LightFactory.hpp"
00015 #include "../common/Graphs.hpp"
00016 #include "../common/Exception/FactoryException.hpp"
00017
00018 class FactoryManager {
00019 public:
00020     FactoryManager();
00021     ~FactoryManager() = default;
00022
00023     std::shared_ptr<PrimitiveFactory> getPrimitiveFactory() { return _primitiveFactory; }
00024     std::shared_ptr<CameraFactory> getCameraFactory() { return _cameraFactory; }
00025
00026     void initializeFactories();
00027
00028     void createObjectsFromConfig(const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>&
00029                                objectsConfig);
00030
00031     std::shared_ptr<GraphsNodePrimitive> getPrimitives() const { return _primitives; }
00032     std::vector<std::shared_ptr<ILight>> getLights() const { return _lights; }
00033     std::shared_ptr<Camera> getCamera() const { return _camera; }
00034     float getAmbientLight() const { return _ambientLight; }
00035
00036 private:
00037     std::shared_ptr<PrimitiveFactory> _primitiveFactory;
00038     std::shared_ptr<CameraFactory> _cameraFactory;
00039     std::shared_ptr<LightFactory> _lightsFactory;
00040     std::shared_ptr<GraphsNodePrimitive> _primitives;
00041     std::vector<std::shared_ptr<ILight>> _lights;

```

```

00041     std::shared_ptr<Camera> _camera;
00042     float _ambientLight;
00043 };
00044
00045 #endif /* !FACTORYMANAGER_HPP_ */

```

## 6.43 IFactory.hpp

```

00001 #ifndef IFACTORY_HPP_
00002 #define IFACTORY_HPP_
00003
00004 #include <memory>
00005 #include <string>
00006 #include <map>
00007 #include "../common/ValueType.hpp"
00008 #include "../common/ILight.hpp"
00009
00010 template <typename T>
00011 class IFactory {
00012 public:
00013     virtual ~IFactory() = default;
00014     virtual std::shared_ptr<T> create(const std::string& type,
00015         std::shared_ptr<std::map<ValueType_t, ValueType>> config,
00016         const std::vector<std::shared_ptr<std::map<ValueType_t,
00017             ValueType>>& graphSceneList, const std::vector<std::shared_ptr<ILight>> &lights) = 0;
00018
00019     virtual std::shared_ptr<T> createSimple(const std::string& type,
00020         std::shared_ptr<std::map<ValueType_t, ValueType>> config,
00021         const std::vector<std::shared_ptr<std::map<ValueType_t,
00022             ValueType>>>& graphSceneList) = 0;
00023
00024     virtual void registerCreator(
00025         const std::string& type,
00026         std::function<std::shared_ptr<T>(
00027             std::shared_ptr<std::map<ValueType_t, ValueType>>,
00028             const std::vector<std::shared_ptr<std::map<ValueType_t,
00029                 ValueType>>&)> creator) = 0;
00030     virtual void registerCreatorLight(
00031         const std::string& type,
00032         std::function<std::shared_ptr<T>(
00033             std::shared_ptr<std::map<ValueType_t, ValueType>>,
00034             const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>>&,
00035             const std::vector<std::shared_ptr<ILight>>&)> creator) = 0;
00036     virtual bool loadPlugin(const std::string& path) = 0;
00037 };
00038
00039 #endif /* !IFACTORY_HPP_ */
00040

```

## 6.44 LightFactory.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** LightFactory
00006 */
00007
00008 #ifndef LIGHTFACTORY_HPP_
00009 #define LIGHTFACTORY_HPP_
00010
00011 #include <vector>
00012 #include <memory>
00013 #include <functional>
00014 #include "IFactory.hpp"
00015 #include "../common/ALight.hpp"
00016 #include "../lib/DLLoader.hpp"
00017 #include "../common/Exception/FactoryException.hpp"
00018
00019
00020 class LightFactory : public IFactory<ILight> {
00021 public:
00022     LightFactory();
00023     ~LightFactory();
00024
00025     std::shared_ptr<ILight> create(const std::string& type,
00026         std::shared_ptr<std::map<ValueType_t, ValueType>> config,
00027         const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>>& graphSceneList,
00028         const std::vector<std::shared_ptr<ILight>>& lights) override;

```

```

00029
00030     std::shared_ptr<ILight> createSimple(const std::string& type,
00031     std::shared_ptr<std::map<ValueType_t, ValueType> config,
00032     const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>& graphSceneList)
00033     override;
00034
00034     void registerCreator(const std::string& type,
00035     std::function<std::shared_ptr<ILight>(
00036     std::shared_ptr<std::map<ValueType_t, ValueType>,
00037     const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>&)> creator) override;
00038
00039     void registerCreatorLight(const std::string& type,
00040     std::function<std::shared_ptr<ILight>(
00041     std::shared_ptr<std::map<ValueType_t, ValueType>,
00042     const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>&,>
00043     const std::vector<std::shared_ptr<ILight>&)> creator) override;
00044
00045     bool loadPlugin(const std::string& path) override;
00046     void loadAllPlugins(const std::string& directory = "plugins/");
00047     ObjectType getTypeFromPlugin(const std::string& path, DLLoader<void*> loader);
00048     std::string getNameFromPlugin(const std::string& path, DLLoader<void*> loader);
00049     protected:
00050     private:
00051         std::map<std::string, std::function<std::shared_ptr<ILight>(
00052         std::shared_ptr<std::map<ValueType_t, ValueType>,
00053         const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>&)> _creators;
00054         std::vector<DLLoader<void*> _dlLoaders;
00055     };
00056
00057 #endif /* !LIGHTFACTORY_HPP_ */

```

## 6.45 PrimitiveFactory.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** PrimitiveFactory
00006 */
00007
00008 #ifndef PRIMITIVEFACTORY_HPP_
00009 #define PRIMITIVEFACTORY_HPP_
00010
00011 #include <vector>
00012 #include <memory>
00013 #include <functional>
00014 #include "IFactory.hpp"
00015 #include "../common/APrimitives.hpp"
00016 #include "../lib/DLLoader.hpp"
00017 #include "../common/Exception/FactoryException.hpp"
00018
00019 class PrimitiveFactory : public IFactory<IPrimitives> {
00020 public:
00021     PrimitiveFactory();
00022     ~PrimitiveFactory();
00023
00024     std::shared_ptr<IPrimitives> create(const std::string& type,
00025     std::shared_ptr<std::map<ValueType_t, ValueType> config,
00026     const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>& graphSceneList,
00027     const std::vector<std::shared_ptr<ILight>& lights) override;
00028
00029     std::shared_ptr<IPrimitives> createSimple(const std::string& type,
00030     std::shared_ptr<std::map<ValueType_t, ValueType> config,
00031     const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>& graphSceneList) override;
00032
00033     void registerCreatorLight(const std::string& type,
00034     std::function<std::shared_ptr<IPrimitives>(
00035     std::shared_ptr<std::map<ValueType_t, ValueType>,
00036     const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>&,>
00037     const std::vector<std::shared_ptr<ILight>&)> creator) override;
00038
00039     void registerCreator(const std::string& type,
00040     std::function<std::shared_ptr<IPrimitives>(
00041     std::shared_ptr<std::map<ValueType_t, ValueType>,
00042     const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>&)> creator) override;
00043
00044     bool loadPlugin(const std::string& path) override;
00045
00046     void loadAllPlugins(const std::string& directory = "plugins/");
00047     ObjectType getTypeFromPlugin(const std::string& path, DLLoader<void*> loader);
00048     std::string getNameFromPlugin(const std::string& path, DLLoader<void*> loader);
00049
00050     void setTexturePathIfNeeded(

```

```

00051         std::shared_ptr<IPrimitives> primitive,
00052         std::shared_ptr<std::map<ValueType_t, ValueType> config>);
00053
00054     std::shared_ptr<IMaterial> createMaterial(const std::string& materialName);
00055     std::shared_ptr<IMaterial> createMaterialByType(MaterialType matType);
00056
00057 private:
00058     std::map<std::string, std::function<std::shared_ptr<IPrimitives>(
00059         std::shared_ptr<std::map<ValueType_t, ValueType>,
00060         const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>&,
00061         const std::vector<std::shared_ptr<ILight>&>> _creators;
00062     std::vector<DLLoader<void*> _dlLoaders;
00063     std::map<std::string, std::shared_ptr<IMaterial> _materialList;
00064 };
00065
00066 #endif /* !PRIMITIVEFACTORY_HPP_ */

```

## 6.46 DirectionalLight.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** DirectionalLight
00006 */
00007
00008 #ifndef DIRECTIONALLIGHT_HPP_
00009 #define DIRECTIONALLIGHT_HPP_
00010
00011 #include "../common/ALight.hpp"
00012 #include "../common/Vector3D.hpp"
00013 #include "../common/ValueType.hpp"
00014 #include <map>
00015 #include <optional>
00016
00017 class DirectionalLight : public ALight {
00018     public:
00019         DirectionalLight();
00020         DirectionalLight(const Color &color, float intensity,
00021             const Math::Vector3D &direction, float radius);
00022         DirectionalLight(std::shared_ptr<std::map<ValueType_t, ValueType> map);
00023         ~DirectionalLight();
00024
00025         void addLight(PixelInfo &pixelInfo, const Math::Ray &ray) const override;
00026         TypeLight getTypeLight() const override;
00027         Math::Vector3D getDirection() const override;
00028         float getRadius() const override;
00029
00030         // Setters
00031         void setDirection(const Math::Vector3D &direction);
00032
00033     private:
00034         Math::Vector3D _direction;
00035         Color _color;
00036         float _intensity;
00037         float _radius;
00038 };
00039
00040
00041 #endif /* !DIRECTIONALLIGHT_HPP_ */

```

## 6.47 PhongLight.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** PhongLight
00006 */
00007
00008 #ifndef PHONGLIGHT_HPP_
00009 #define PHONGLIGHT_HPP_
00010
00011 #include "../common/ALight.hpp"
00012 #include "../common/Point3D.hpp"
00013
00014 #include <map>
00015
00016 class PhongLight : public ALight {

```

```

00017     public:
00018         PhongLight();
00019         PhongLight(std::shared_ptr<std::map<ValueType_t, ValueType> map);
00020         PhongLight(const Color &color, float intensity,
00021                 const Math::Vector3D &direction, float radius, float shininess);
00022         ~PhongLight();
00023
00024         void addLight(PixelInfo &pixelInfo, const Math::Ray &ray) const override;
00025         TypeLight getTypeLight() const override;
00026         Math::Vector3D getDirection() const override;
00027         float getRadius() const override;
00028         // Getters
00029         float getShininess() const;
00030
00031         // Setters
00032         void setDirection(const Math::Vector3D &position);
00033         void setShininess(float shininess);
00034
00035     private:
00036         Math::Vector3D _direction;
00037         Color _color;
00038         float _intensity;
00039         float _radius;
00040         float _shininess;
00041 };
00042
00043 #endif /* !PHONGLIGHT_HPP_ */

```

## 6.48 PointLight.hpp

```

00001 /*
00002  ** EPITECH PROJECT, 2025
00003  ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004  ** File description:
00005  ** PointLight
00006  */
00007
00008 #ifndef POINTLIGHT_HPP_
00009 #define POINTLIGHT_HPP_
00010
00011 #include "../common/ALight.hpp"
00012 #include "../common/Point3D.hpp"
00013
00014 class PointLight : public ALight {
00015     public:
00016         PointLight();
00017         PointLight(const Color &color, float intensity,
00018                 const Math::Point3D &position, float radius);
00019         ~PointLight();
00020
00021         /* Override method */
00022         TypeLight getTypeLight() const override;
00023         void addLight(PixelInfo &pixelInfo, const Math::Ray &ray) const override;
00024
00025         // Getters
00026         Math::Point3D getPosition() const;
00027         float getRadius() const;
00028
00029         // Setters
00030         void setPosition(const Math::Point3D &position);
00031         void setRadius(float radius);
00032
00033     private:
00034         Math::Point3D _position;
00035         float _radius;
00036 };
00037
00038 #endif /* !POINTLIGHT_HPP_ */

```

## 6.49 IMediator.hpp

```

00001 /*
00002  ** EPITECH PROJECT, 2025
00003  ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004  ** File description:
00005  ** IMediator
00006  */
00007
00008 #ifndef IMEDIATOR_HPP_

```



```

00009 #define IMEDIATOR_HPP_
00010
00011 #include <functional>
00012
00013 class IMediator {
00014     public:
00015         virtual void addTask(std::function<void()> task) = 0;
00016         virtual void executeTasks() = 0;
00017         virtual void waitForCompletion() = 0;
00018     protected:
00019     private:
00020 };
00021
00022 #endif /* !IMEDIATOR_HPP_ */

```

## 6.50 RayMediator.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** RayMediator
00006 */
00007
00008 #ifndef RAYMEDIATOR_HPP_
00009 #define RAYMEDIATOR_HPP_
00010
00011 #include <thread>
00012 #include <functional>
00013 #include <vector>
00014 #include <mutex>
00015 #include <condition_variable>
00016
00017 #include "IMediator.hpp"
00018
00019 class RayMediator : public IMediator {
00020     public:
00021         RayMediator();
00022         ~RayMediator();
00023
00024         void addTask(std::function<void()> task) override;
00025         void executeTasks() override;
00026         void waitForCompletion() override;
00027
00028     private:
00029         std::vector<std::thread> _threads;
00030         std::vector<std::function<void()>> _tasks;
00031         std::mutex _mutex;
00032         std::condition_variable _condition;
00033         bool _stop = false;
00034 };
00035
00036 #endif /* !RAYMEDIATOR_HPP_ */

```

## 6.51 ConfigNode.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** ConfigNode
00006 */
00007
00008 #ifndef CONFIGNODE_HPP_
00009 #define CONFIGNODE_HPP_
00010
00011 #include <iostream>
00012 #include <map>
00013 #include <variant>
00014
00015 #include "../common/ValueType.hpp"
00016
00017 enum NodeType { TypeUnknown, TypeGroup, TypeArray, TypeValue, TypeList };
00018
00019 class ConfigNode {
00020     public:
00021         ConfigNode() : isValue(false), type(TypeUnknown), _name("") {}
00022         ~ConfigNode() = default;

```

```

00023
00024         std::map<std::string, ConfigNode> children;
00025         ValueType value;
00026         bool isValue;
00027         NodeType type;
00028         std::string _name;
00029
00030         bool hasChild(const std::string &name) const;
00031         // template <typename T>
00032         //T getValue(const T &defaultValue = T()) const;
00033     };
00034
00035 #endif /* !CONFIGNODE_HPP_ */

```

## 6.52 ConfigParser.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** ConfigParser
00006 */
00007
00008 #ifndef CONFIGPARSER_HPP_
00009 #define CONFIGPARSER_HPP_
00010
00011 #include <libconfig.h++>
00012 #include <string>
00013 #include <memory>
00014
00015 #include "ConfigNode.hpp"
00016
00017 using namespace libconfig;
00018
00019 class ConfigParser {
00020 public:
00021     ConfigParser();
00022     ~ConfigParser();
00023     bool loadConfig(const std::string &filename, ConfigNode &rootNode);
00024
00025 protected:
00026     void buildConfigTree(const Setting &setting, std::shared_ptr<ConfigNode> node);
00027
00028 private:
00029     void handleGroupType(const Setting &child, const std::string &childName,
00030                         std::shared_ptr<ConfigNode> node);
00031     void handleArrayType(const Setting &child, const std::string &childName,
00032                         std::shared_ptr<ConfigNode> node);
00033     void handleListType(const Setting &child, const std::string &childName,
00034                        std::shared_ptr<ConfigNode> node);
00035     void handleValueType(const Setting &child, const std::string &childName,
00036                        std::shared_ptr<ConfigNode> node);
00037
00038     Config file;
00039 };
00040
00041 #endif /* !CONFIGPARSER_HPP_ */

```

## 6.53 ObjectConstructor.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** ObjectFactory
00006 */
00007
00008 #ifndef OBJECTFACTORY_HPP_
00009 #define OBJECTFACTORY_HPP_
00010
00011 #include <map>
00012 #include <string>
00013 #include <vector>
00014 #include <memory>
00015
00016 #include "ConfigNode.hpp"
00017 #include "PropertyTypes.hpp"
00018 #include "ObjectErrorHandling.hpp"
00019

```

```

00020 class ObjectConstructor {
00021 public:
00022     ObjectConstructor();
00023     ~ObjectConstructor();
00024
00025     void createObject(const ConfigNode& node);
00026     void createObjects(const ConfigNode& node);
00027     bool verifyObjectValidity(const ConfigNode& node, const std::string& objectName);
00028     const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>&
00029         getObjects() const;
00030
00031     void printObjectMap() const;
00032     bool createMaterials(const ConfigNode &node);
00033
00034 private:
00035     void fillObject(const ConfigNode& node,
00036         std::shared_ptr<std::map<ValueType_t, ValueType> object);
00037     void handleSimpleValue(std::shared_ptr<std::map<ValueType_t,
00038         ValueType> object,
00039         const ValueType_t& key,
00040         const ValueType& value,
00041         ValueDataType dataType);
00042     void handleVector2DValue(std::shared_ptr<std::map<ValueType_t,
00043         ValueType> object, const ValueType_t& key,
00044         const ConfigNode& node,
00045         const std::vector<std::string>& components,
00046         ValueDataType dataType);
00047     void handleVector3DValue(std::shared_ptr<std::map<ValueType_t, ValueType> object,
00048         const ValueType_t& key,
00049         const ConfigNode& node,
00050         const std::vector<std::string>& components,
00051         ValueDataType dataType);
00052     ValueType convertValue(const ValueType& value, ValueDataType dataType);
00053
00054     void initShapeDefinitions();
00055
00056     ObjectErrorHandling _errorHandler;
00057     std::map<std::string, PropertyConfig> _propertyTypeMap;
00058     std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>> _objects;
00059     std::vector<ShapeDefinition> _shapeDefinitions;
00060
00061 };
00062
00063 #endif /* !OBJECTFACTORY_HPP_ */

```

## 6.54 ObjectErrorHandling.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** ObjectErrorHandling
00006 */
00007
00008 #ifndef OBJECTERRORHANDLING_HPP_
00009 #define OBJECTERRORHANDLING_HPP_
00010
00011 #include <iostream>
00012 #include <map>
00013 #include <variant>
00014 #include <memory>
00015 #include "ConfigNode.hpp"
00016 #include "PropertyTypes.hpp"
00017 #include "../common/ValueType.hpp"
00018
00019 class ObjectErrorHandling {
00020 public:
00021     ObjectErrorHandling();
00022     ~ObjectErrorHandling();
00023
00024     bool checkArrayValidity(const ConfigNode& node,
00025         const std::string& objectName);
00026     bool checkGroupValidity(const ConfigNode& node,
00027         const std::string& objectName);
00028     bool checkListValidity(const ConfigNode& node,
00029         const std::string& objectName);
00030     bool checkValueValidity(const ConfigNode& node,
00031         const std::string& objectName);
00032     bool verifyObjectValidity(const ConfigNode& node,
00033         const std::string& objectName);
00034
00035     void setShapeDefinitions
00036         (std::vector<ShapeDefinition> shapeDefinitions);

```

```

00037         void setPropertyTypeMap(const std::map<std::string,
00038                                 PropertyConfig>& propertyTypeMap);
00039
00040     protected:
00041     private:
00042         std::vector<ShapeDefinition> _shapeDefinitions;
00043         std::map<std::string, PropertyConfig> _propertyTypeMap;
00044
00045         std::string getDataTypeName(ValueDataType type) const;
00046         std::shared_ptr<const ShapeDefinition> getShapeDefinition
00047             (const std::string& objectName) const;
00048
00049         bool isParameterValid(const std::string& parameter,
00050                               const std::string& objectName) const;
00051         bool isParameterMandatory(const std::string& parameter,
00052                                   const std::string& objectName) const;
00053         bool isParameterOptional(const std::string& parameter,
00054                                  const std::string& objectName) const;
00055         bool checkParameterType(const std::string& parameter,
00056                                 const ConfigNode& node) const;
00057
00058         bool isValueTypeValid(const ValueType& value,
00059                               ValueDataType expectedType) const;
00060         bool checkSimpleValueValidity(const ConfigNode& node,
00061                                       const std::string &parameter, const PropertyConfig& config) const;
00062         bool checkVector2DValueValidity(const ConfigNode& node,
00063                                         const std::string &parameter, const PropertyConfig& config) const;
00064         bool checkVector3DValueValidity(const ConfigNode& node,
00065                                         const std::string &parameter, const PropertyConfig& config) const;
00066
00067         bool checkMandatoryParameters(const ConfigNode& node,
00068                                       const std::string& objectName) const;
00069         bool checkOptionalParameters(const ConfigNode& node,
00070                                      const std::string& objectName) const;
00071         bool checkUnknownParameters(const ConfigNode& node,
00072                                     const std::string& objectName) const;
00073
00074     };
00075
00076 #endif /* !OBJECTERRORHANDLING_HPP_ */

```

## 6.55 Parser.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** Parser
00006 */
00007
00008 #ifndef PARSER_HPP_
00009 #define PARSER_HPP_
00010
00011 #include <memory>
00012 #include <string>
00013 #include <vector>
00014 #include <filesystem>
00015 #include "ConfigParser.hpp"
00016 #include "ObjectConstructor.hpp"
00017
00018 class Parser {
00019 public:
00020     Parser();
00021     Parser(const std::string &filename);
00022     ~Parser();
00023     void loadConfig(const std::string &filename);
00024     void parse();
00025     const std::vector<std::shared_ptr<std::map<ValueT_t, ValueType>>
00026                 &getObjects() const;
00027     std::shared_ptr<ConfigNode> getRootNode() { return std::make_shared<ConfigNode>(rootNode); }
00028
00029     void printMap() const;
00030 private:
00031     void importScene();
00032     bool isValidFilePath(const std::string &path) const;
00033     bool loadImportedScene(const std::string &scenePath, std::shared_ptr<ConfigNode> importedRootNode)
00034         const;
00035     void importObjectsFromScene(const std::shared_ptr<ConfigNode> importedRootNode);
00036
00037     ConfigNode rootNode;
00038     ConfigParser configParser;
00039     ObjectConstructor _objectConstructor;
00040 };

```

```

00040
00041 #endif /* !PARSER_HPP_ */

```

## 6.56 PropertyTypes.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** PropertyTypes
00006 */
00007
00008 #ifndef PROPERTYTYPES_HPP_
00009 #define PROPERTYTYPES_HPP_
00010
00011 #include <map>
00012 #include <string>
00013 #include <vector>
00014
00015 #include "../common/ValueType.hpp"
00016
00017 enum ValueDataType {
00018     TYPE_INT,
00019     TYPE_FLOAT,
00020     TYPE_STRING,
00021     TYPE_BOOL,
00022     TYPE_DOUBLE
00023 };
00024
00025 struct PropertyInfo {
00026     ValueType_t type;
00027     ValueFormat format;
00028     std::vector<std::string> components;
00029     ValueDataType dataType;
00030 };
00031
00032 struct ShapeDefinition {
00033     std::string name;
00034     std::vector<std::string> mandatory;
00035     std::vector<std::string> optional;
00036     ObjectType objectType;
00037 };
00038
00039 struct PropertyConfig {
00040     ValueType_t type;
00041     ValueFormat format;
00042     std::vector<std::string> components;
00043     ValueDataType dataType;
00044 };
00045
00046 #endif /* !PROPERTYTYPES_HPP_ */

```

## 6.57 ValueConverter.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** ValueConverter
00006 */
00007
00008 #ifndef VALUECONVERTER_HPP_
00009 #define VALUECONVERTER_HPP_
00010
00011 #include "../common/Vector2D.hpp"
00012 #include "../common/Vector3D.hpp"
00013 #include "PropertyTypes.hpp"
00014
00015 class ValueConverter {
00016 public:
00017     static float getFloatFromVariant(const ValueType &value);
00018     static Math::Vector2D getVector2DFromComponents(const ValueType &x,
00019                                                     const ValueType &y);
00020     static Math::Vector3D getVector3DFromComponents(const ValueType &x,
00021                                                     const ValueType &y,
00022                                                     const ValueType &z);
00023 };
00024
00025 #endif /* !VALUECONVERTER_HPP_ */

```

## 6.58 Cone.hpp

```

00001  /*
00002  ** EPITECH PROJECT, 2025
00003  ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004  ** File description:
00005  ** Cone
00006  */
00007
00008  #ifndef CONE_HPP_
00009  #define CONE_HPP_
00010
00011  #include <map>
00012  #include <optional>
00013  #include "../common/APrimitives.hpp"
00014  #include "../common/ValueType.hpp"
00015
00016  class Cone : public APrimitives {
00017  public:
00018      Cone();
00019      Cone(std::shared_ptr<std::map<ValueType_t, ValueType>> map,
00020           const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>> &graphSceneList,
00021           const std::vector<std::shared_ptr<ILight>> &lights);
00022      ~Cone();
00023      /* Setter */
00024      void setBaseRadius(float radius);
00025      void setHeight(float height);
00026
00027      /* Method */
00028      std::optional<double> distance(const Math::Ray &ray) const override;
00029      PixelInfo distanceInfo(const Math::Ray &ray) override;
00030      std::optional<Math::Point3D> getIntersection(const Math::Ray &ray) const override;
00031      std::optional<Math::Vector3D> getNormal(const Math::Point3D &point) const override;
00032
00033      /* Getter */
00034      float getBaseRadius() const;
00035      Type getType() const override;
00036      float getHeight() const;
00037
00038  protected:
00039  private:
00040      float _baseRadius;
00041      float _height;
00042      double _distance;
00043
00044      Math::Vector3D transformToLocal(const Math::Ray &ray) const;
00045      Math::Vector3D localDirectionVector(const Math::Ray &ray) const;
00046      std::optional<double> intersectConeBody(const Math::Vector3D &localOrigin, const
00047      Math::Vector3D &localDirection) const;
00048      std::optional<double> intersectConeBase(const Math::Vector3D &localOrigin, const
00049      Math::Vector3D &localDirection) const;
00050      bool isPointOnConeBody(const Math::Point3D &hitPoint) const;
00051  };
00052  #endif /* !CONE_HPP_ */

```

## 6.59 Cylinder.hpp

```

00001  /*
00002  ** EPITECH PROJECT, 2025
00003  ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004  ** File description:
00005  ** Cylinder
00006  */
00007
00008  #ifndef CYLINDER_HPP_
00009  #define CYLINDER_HPP_
00010
00011  #include <map>
00012  #include "../common/APrimitives.hpp"
00013  #include "../common/ValueType.hpp"
00014  // #include "../common/Color.hpp"
00015
00016  class Cylinder : public APrimitives {
00017  public:
00018      Cylinder();
00019      Cylinder(std::shared_ptr<std::map<ValueType_t, ValueType>> map,
00020           const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>> &graphSceneList,
00021           const std::vector<std::shared_ptr<ILight>> &lights);
00022      ~Cylinder();
00023
00024

```

```

00025         /* Setter */
00026         void setBaseRadius(float radius);
00027         void setHeight(float height);
00028
00029         /* Getter */
00030         float getBaseRadius() const;
00031         float getHeight() const;
00032         Type getType() const override;
00033         std::optional<double> distance(const Math::Ray &ray) const override;
00034         PixelInfo distanceInfo(const Math::Ray &ray) override;
00035         std::optional<Math::Point3D> getIntersection(const Math::Ray &ray) const override;
00036         std::optional<Math::Vector3D> getNormal(const Math::Point3D &point) const override;
00037
00038     private:
00039         float _baseRadius;
00040         float _height;
00041         double _distance;
00042     };
00043
00044 #endif /* !CYLINDER_HPP_ */
00045
00046

```

## 6.60 Plane.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** Plane
00006 */
00007
00008 #include <memory>
00009 #include <map>
00010 #include "../common/ValueType.hpp"
00011 #include "../common/material/IMaterial.hpp"
00012 #include "../common/APrimitives.hpp"
00013 #include "../common/Point3D.hpp"
00014 #include "../common/Rot3D.hpp"
00015 #include "../common/Vector3D.hpp"
00016
00017 class Plane : public APrimitives {
00018 public:
00019     Plane();
00020     Plane(std::shared_ptr<std::map<ValueType_t, ValueType>> map,
00021          const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>> &graphSceneList,
00022          const std::vector<std::shared_ptr<ILight>> &lights);
00023     ~Plane() override;
00024
00025     std::optional<double> distance(const Math::Ray &ray) const override;
00026     PixelInfo distanceInfo(const Math::Ray &ray) override;
00027     std::optional<Math::Point3D> getIntersection(const Math::Ray &ray) const override;
00028     std::optional<Math::Vector3D> getNormal(const Math::Point3D &point) const override;
00029     /* Getter */
00030     Type getType() const override;
00031
00032     /* Setter */
00033     void setRotation(const Math::Rot3D &newRotation) override;
00034     void updateNormal();
00035
00036 private:
00037     double _distance;
00038     Math::Vector3D _normal;
00039     std::string _axe;
00040 };

```

## 6.61 Sphere.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** Sphere
00006 */
00007
00008 #include <memory>
00009 #include <map>
00010 #include "../common/material/IMaterial.hpp"
00011 #include "../common/APrimitives.hpp"

```

```

00012 #include "../common/Point3D.hpp"
00013 #include "../common/Rot3D.hpp"
00014 #include "../common/Vector3D.hpp"
00015 #include "../common/ValueType.hpp"
00016
00017 class Sphere : public APrimitives {
00018 public:
00019     Sphere();
00020     Sphere(std::shared_ptr<std::map<ValueType_t, ValueType>> map,
00021           const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>> &graphSceneList,
00022           const std::vector<std::shared_ptr<ILight>> &lights);
00023     ~Sphere() override;
00024
00025     std::optional<double> distance(const Math::Ray &ray) const override;
00026     PixelInfo distanceInfo(const Math::Ray &ray) override;
00027     std::optional<Math::Point3D> getIntersection(const Math::Ray &ray) const override;
00028     std::optional<Math::Vector3D> getNormal(const Math::Point3D &point) const override;
00029
00030     /* Getter */
00031     Type getType() const override;
00032
00033 private:
00034     double _distance;
00035     double radius;
00036 };

```

## 6.62 Torus.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** Raytracer
00004 ** File description:
00005 ** Torus
00006 */
00007
00008 #include "../common/APrimitives.hpp"
00009 #include "../common/Ray.hpp"
00010 #include <map>
00011 #include <memory>
00012 #include <optional>
00013 #include <vector>
00014
00015 #ifndef TORUS_HPP_
00016     #define TORUS_HPP_
00017
00018 class Torus : public APrimitives {
00019 public:
00020     Torus();
00021     Torus(std::shared_ptr<std::map<ValueType_t, ValueType>> map,
00022           const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>> &graphSceneList,
00023           const std::vector<std::shared_ptr<ILight>> &lights);
00024     // Add 2-argument constructor for plugin factory
00025     Torus(std::shared_ptr<std::map<ValueType_t, ValueType>> map,
00026           const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType>>> &graphSceneList);
00027     ~Torus() override;
00028
00029     /* Setter */
00030     void setMajorRadius(float radius);
00031     void setMinorRadius(float radius);
00032
00033     /* Getter */
00034     float getMajorRadius() const;
00035     float getMinorRadius() const;
00036     Type getType() const override;
00037
00038     std::optional<double> distance(const Math::Ray &ray) const override;
00039     PixelInfo distanceInfo(const Math::Ray &ray) override;
00040     std::optional<Math::Point3D> getIntersection(const Math::Ray &ray) const override;
00041     std::optional<Math::Vector3D> getNormal(const Math::Point3D &point) const override;
00042
00043 private:
00044     float _majorRadius;
00045     float _minorRadius;
00046     double _distance;
00047
00048     std::tuple<double, double, double, double, double> computeQuarticCoefficients(const
Math::Vector3D &localOrigin, const Math::Vector3D &localDir) const;
00049     double evaluateQuartic(double t, double a, double b, double c, double d_coef, double e) const;
00050     std::vector<double> findRootCandidates(double a, double b, double c, double d_coef, double e)
const;
00051     double refineRoot(double t, double a, double b, double c, double d_coef, double e) const;
00052 };
00053
00054 #endif /* !TORUS_HPP_ */

```



## 6.63 Raytracer.hpp

```

00001  /*
00002  ** EPITECH PROJECT, 2025
00003  ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004  ** File description:
00005  ** Raytracer
00006  */
00007
00008  #include <iostream>
00009  #include <map>
00010  #include <memory>
00011  #include <functional>
00012  #include <vector>
00013  #include <optional>
00014
00015  #include "../common/APrimitives.hpp"
00016  #include "../common/ValueType.hpp"
00017
00018  #include "../common/Error.hpp"
00019  #include "../common/Image.hpp"
00020  #include "../common/Camera.hpp"
00021  #include "../common/Scene.hpp"
00022  #include "../lib/DLLoader.hpp"
00023  #include "../lib/ILoader.hpp"
00024  #include "factory/FactoryManager.hpp"
00025  #include "../lib/SFML/GraphicMode.hpp"
00026  #include "../common/Graphs.hpp"
00027
00028  #ifndef RAYTRACER_HPP_
00029  #define RAYTRACER_HPP_
00030
00031  struct InfoPixelDisplay {
00032      double distance;
00033      Color color;
00034      float transparency;
00035  };
00036
00037  struct DropShadowInfo {
00038      Math::Vector3D position;
00039      float darkness; // 0 to 1
00040  };
00041
00042  class Raytracer {
00043  public:
00044      Raytracer();
00045      ~Raytracer();
00046
00047      /* Getter */
00048      std::string getSceneFile() const;
00049      std::string getOutputFile() const;
00050      std::string getOutputFormat() const;
00051      std::shared_ptr<Image> getImage() const;
00052      bool getGraphicMode() const;
00053      bool isDebug() const;
00054      Scene getScene() const;
00055
00056      /* Setter */
00057      void setSceneFile(std::string sceneFile);
00058      void setOutputFile(std::string outputFile);
00059      void setOutputFormat(std::string outputFormat);
00060      void setImage(std::shared_ptr<Image> image);
00061      void setScene(Scene scene);
00062      bool setGraphicMode();
00063
00064      /* Methods */
00065      void writeToFilePPM(std::string fileName);
00066
00067      void parseCmd(int ac, char **av);
00068
00069      void LoadAllformlibs(const std::vector<std::shared_ptr<std::map
00070                          <ValueType_t, ValueType>>& objectsConfig);
00071
00072      std::optional<PixelInfo> getClosestPrimitiveHit(const Math::Ray &ray) const;
00073
00074      Color TraceRay(int x, int y, uint32_t& state);
00075
00076      /* Image Method */
00077      void InitParams();
00078      void setScene();
00079      void StartImage();
00080      void printLoadingBar(std::shared_ptr<int> pixelCount, int totalPixels, int barWidth);
00081      Color blendColors(const Color& foreground, const Color& background, float transparency);
00082      std::vector<InfoPixelDisplay> calculatePixel(const Math::Ray& ray);
00083      void averageImages(const std::vector<std::shared_ptr<Image>>& images);
00084      void loopThruType();
00085

```

```

00086         /* New image processing methods */
00087         void initializeScene();
00088         Color computePixelColor(double u, double v, const Color& backgroundColor);
00089         void displayGraphicMode(std::shared_ptr<int> pixelCount, int totalPixels);
00090         void renderConsoleMode(const Color& backgroundColor);
00091         void renderGraphicMode(int width, int height, std::shared_ptr<int> pixelCount);
00092         void finalizeRendering();
00093         void generateDropShadows();
00094
00095         void setAntialiasingSamples(int samples);
00096         int getAntialiasingSamples() const;
00097     protected:
00098     private:
00099         bool graphicMode;
00100         bool debugMode;
00101         std::string _sceneFile;
00102         std::string _outputFile;
00103         std::string _outputFormat;
00104         int numRenders;
00105         std::shared_ptr<Image> image;
00106         Scene _scene;
00107         FactoryManager _factoryManager;
00108         std::shared_ptr<GraphicMode> _display;
00109         int _width;
00110         int _height;
00111         int _antialiasingSamples = 1; // Number of samples per pixel for antialiasing
00112     public:
00113 };
00114
00115 #endif /* !RAYTRACER_HPP_ */

```

## 6.64 Utils.hpp

```

00001 /*
00002  ** EPITECH PROJECT, 2025
00003  ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004  ** File description:
00005  ** Utils
00006  */
00007
00008 #include <iostream>
00009 #include <memory>
00010
00011 #ifndef UTILS_HPP_
00012 #define UTILS_HPP_
00013
00014 class Utils {
00015     public:
00016         Utils();
00017         ~Utils();
00018         static void helper();
00019
00020     protected:
00021     private:
00022 };
00023
00024 #endif /* !UTILS_HPP_ */

```