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 ablige to a specified coding styke,
to check if the code is complient with the norm execut
the make coding command or the ./styleChecker.sh.
To understand the errors and how to fix them please
refers to the coding-cpp.txt.
```

1.3 Documentation :

1.3.1 Docusorus :

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

1.3.2 Doxygen :

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

1.4 Commit norm :

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

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

Gitmojies:

```
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 deja été push :
  ```
  git commit --amend -m "New commit message"
  git push --force
  ```
- Un-add un ficher 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:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

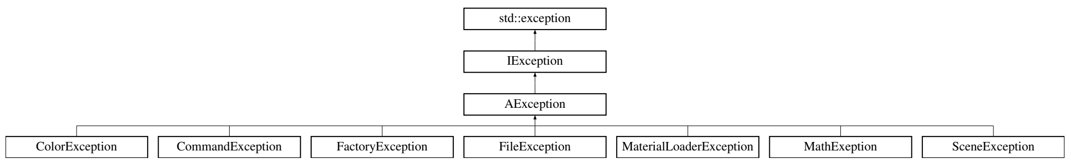
Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 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]
```

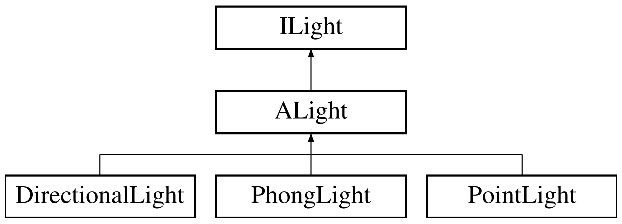
Implements IException.

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]
```

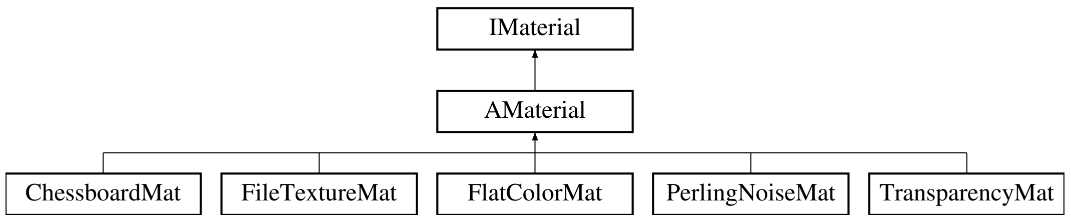
Implements ILight.

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]
```

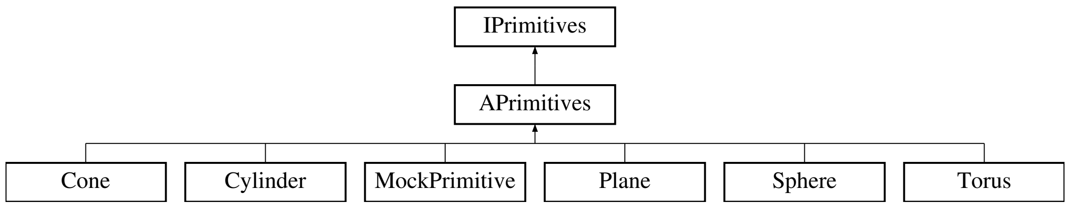
Implements IMaterial.

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 PixelInfo 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** (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)

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

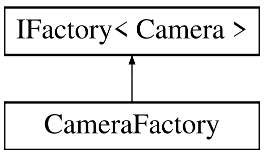
- int **width**
- int **height**
- float **fieldOfView**
- Math::Vector3D **_rotation**
- Math::Vector3D **_position**

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]
```

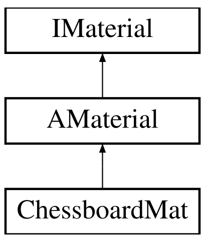
Implements IFactory< Camera >.

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 PixelInfo &pixelInfo, float radius, float height, const IPrimitives &primitive) const
- Color **applyPlaneChessboard** (const PixelInfo &pixelInfo, float radius, float height, const IPrimitives &primitive) const
- Color **applyCylinderChessboard** (const PixelInfo &pixelInfo, float radius, float height, const IPrimitives &primitive) const
- Color **applyConeChessboard** (const PixelInfo &pixelInfo, 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

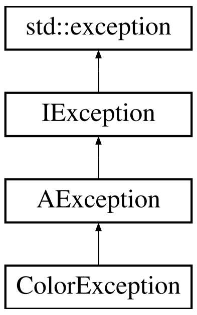
- uint8_t **_red**
- uint8_t **_green**
- uint8_t **_blue**
- uint8_t **_transparency**

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

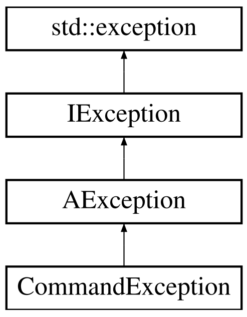
- **AException** (const std::string &type, const std::string &message)
- const char $*$ what () const noexcept override
- std::string getType () const noexcept override
- std::string getMessage () const noexcept override
- std::string getFormattedMessage () const noexcept override

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

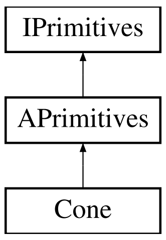
- **AException** (const std::string &type, const std::string &message)
- const char ∗ what () const noexcept override
- std::string getType () const noexcept override
- std::string getMessage () const noexcept override
- std::string getFormattedMessage () const noexcept override

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
- 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
- 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** (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 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

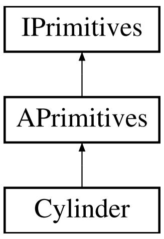
- Config **file**

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]
```

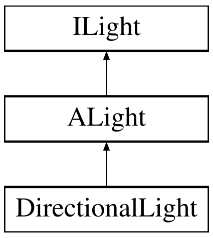
Implements APrimitives.

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]
```

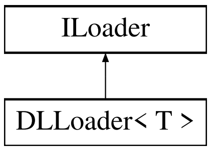
Implements ALight.

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

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

5.16 DLLoader< T > Class Template Reference

Inheritance diagram for DLLoader< 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 DLLoader< T >::Close ( )  [inline], [override], [virtual]
```

Implements ILoader.

5.16.1.2 Error()

```
template<typename T >
const char * DLLoader< 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

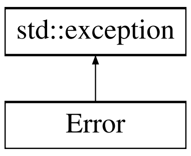
- Math::Vector3D **position**
- float **darkness**

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 **ErrorFaillureException** () const
- void **exitCode** (int code)

Private Attributes

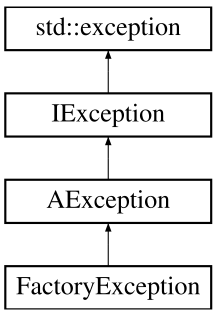
- std::string **_message**
- std::string **_file**
- int **_line**

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

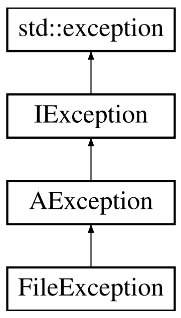
- std::shared_ptr< PrimitiveFactory > **_primitiveFactory**
- std::shared_ptr< CameraFactory > **_cameraFactory**
- std::shared_ptr< LightFactory > **_lightsFactory**
- std::shared_ptr< GraphsNodePrimitive > **_primitives**
- std::vector< std::shared_ptr< ILight > > **_lights**
- std::shared_ptr< Camera > **_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

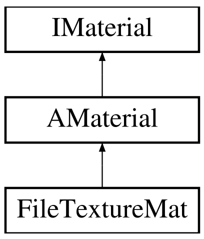
- **AException** (const std::string &type, const std::string &message)
- const char ∗ what () const noexcept override
- std::string getType () const noexcept override
- std::string getMessage () const noexcept override
- std::string getFormattedMessage () const noexcept override

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 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

- 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]
```

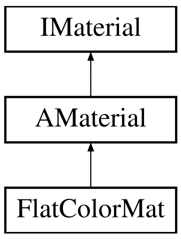
Implements AMaterial.

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]
```

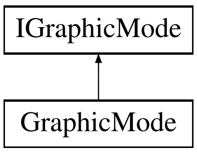
Implements AMaterial.

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

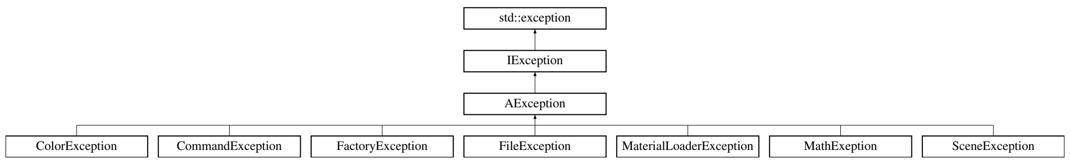
- std::shared_ptr< IPrimitives > **_primitives**
- std::vector< std::shared_ptr< GraphsNodePrimitive > > **_children**

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

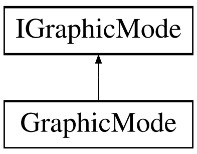
- virtual std::shared_ptr< T > **create** (const std::string &type, std::shared_ptr< std::map< ValueType_↩ t, ValueType > > config, const std::vector< std::shared_ptr< std::map< ValueType_t, ValueType > > > &graphSceneList, const std::vector< std::shared_ptr< ILight > > &lights)=0
- virtual std::shared_ptr< T > **createSimple** (const std::string &type, std::shared_ptr< std::map< Value↩ Type_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

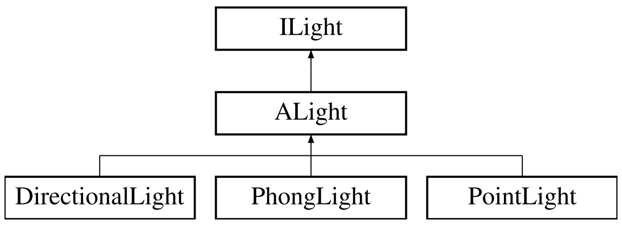
- virtual void **createText** (const std::string &text, int size, int x, int y)=0
- virtual void **createRectangle** (const std::string &id, int x, int y, int width, int height)=0
- virtual bool **getRenderingComplete** () const =0
- virtual void **setWindow** (int width, int height)=0
- virtual void **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

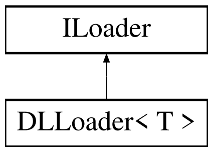
- virtual void **addLight** (PixelInfo &pixelInfo, const Math::Ray &ray) const =0
- virtual Color **getColor** () const =0
- virtual Math::Vector3D **getDirection** () const =0
- virtual float **getIntensity** () const =0
- virtual float **getRadius** () const =0
- virtual TypeLight **getTypeLight**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

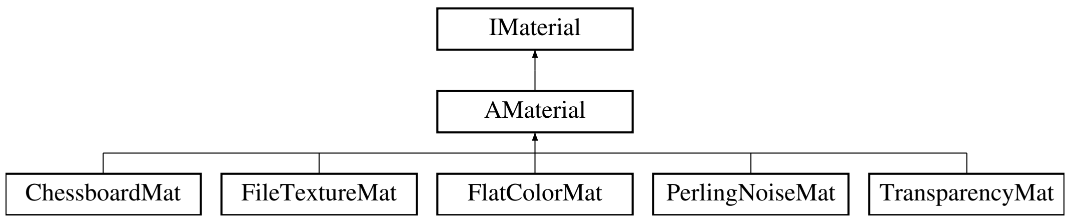
- int **width**
- int **height**
- int **maxColorValue**
- std::vector< Color > **data**

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

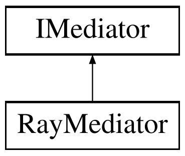
- virtual void **setAmbient** (const Math::Vector3D &a)=0
- virtual void **setDiffuse** (const Math::Vector3D &d)=0
- virtual void **setSpecular** (const Math::Vector3D &s)=0
- virtual void **setShininess** (float s)=0
- virtual void **setReflectivity** (float r)=0
- virtual void **setTransparency** (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

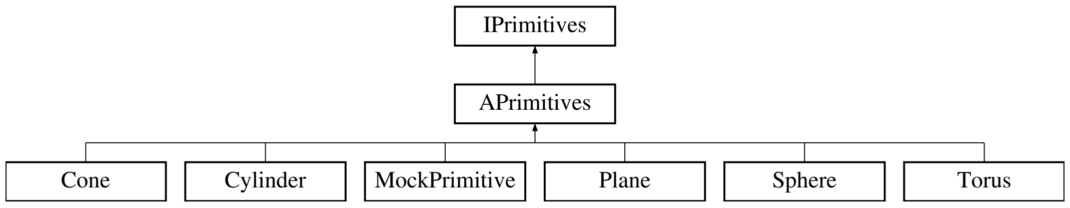
- double **distance**
- Color **color**
- float **transparency**

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

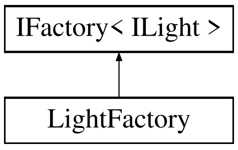
- virtual std::optional< double > **distance** (const Math::Ray &ray) const =0
- virtual PixelInfo **distanceInfo** (const Math::Ray &ray)=0
- virtual std::optional< Math::Point3D > **getIntersection** (const Math::Ray &ray) const =0
- virtual std::optional< Math::Vector3D > **getNormal** (const Math::Point3D &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, 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< 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, ValueType > > > &)> 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< 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], [virtual]
```

Implements IFactory< ILight >.

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

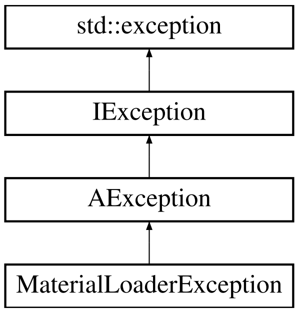
- Math::Vector3D **ambient**
- Math::Vector3D **diffuse**
- Math::Vector3D **specular**
- float **shininess**
- Color **OptionalColor1** = Color(0.0f, 0.0f, 0.0f)
- Color **OptionalColor2** = Color(0.0f, 0.0f, 0.0f)
- float **scale** = 5.0f
- float **reflectivity**
- float **transparency**
- float **refractiveIndex**
- 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

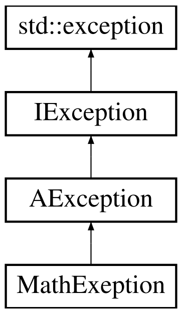
- **AException** (const std::string &type, const std::string &message)
- const char * what () const noexcept override
- std::string getType () const noexcept override
- std::string getMessage () const noexcept override
- std::string getFormattedMessage () const noexcept override

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

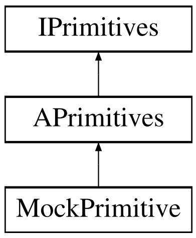
- **AException** (const std::string &type, const std::string &message)
- const char ∗ what () const noexcept override
- std::string getType () const noexcept override
- std::string getMessage () const noexcept override
- std::string getFormattedMessage () const noexcept override

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 > importedRoot↩ Node) const
- void **importObjectsFromScene** (const std::shared_ptr< ConfigNode > importedRootNode)

Private Attributes

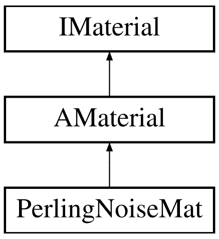
- ConfigNode **rootNode**
- ConfigParser **configParser**
- ObjectConstructor **_objectConstructor**

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 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 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]
```

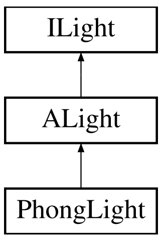
Implements AMaterial.

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

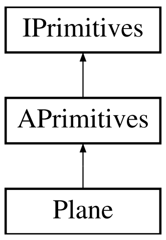
- Color **_color**
- Math::Vector3D **_normalizedVector**
- double **_distance**
- bool **_isHit**
- Math::Vector3D **_pos**
- float **_light_intensity**
- Color **_light_color**

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** (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

- 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

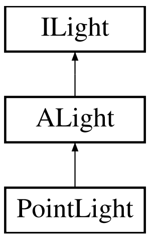
- double **x**
- double **y**
- double **z**

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]
```

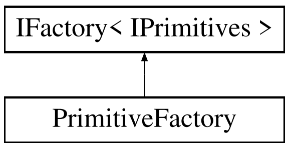
Implements ALight.

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< Value↵Type_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< Value↵Type_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

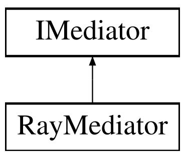
- Math::Ray **_ray**
- Color **_color**
- double **_intensity**

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 **averageImages** (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 **cameraWidth** () 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

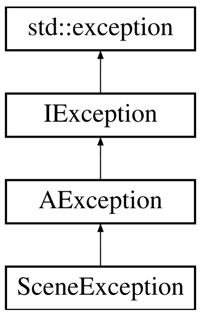
- std::shared_ptr< Camera > **_camera**
- std::vector< std::shared_ptr< IPrimitives > > **_primTemp**
- std::shared_ptr< GraphsNodePrimitive > **_primitives**
- std::shared_ptr< GraphsNodeLight > **_lights**
- float **_ambientLight**

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

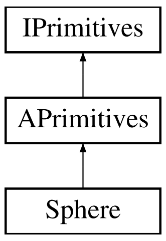
- std::string **name**
- std::vector$<$ std::string $>$ **mandatory**
- std::vector$<$ std::string $>$ **optional**
- ObjectType **objectType**

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
- 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

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

- 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]
```

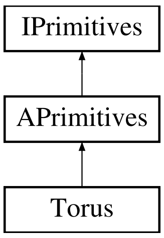
Implements APrimitives.

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** (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 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]
```

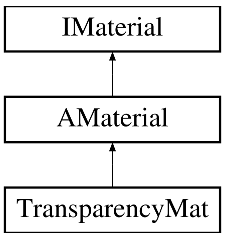
Implements APrimitives.

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 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

5.66.1 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

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** ALight
*/

#ifndef ALIGHT_HPP_
#define ALIGHT_HPP_

#include "ILight.hpp"
#include "ValueType.hpp"
#include "Point3D.hpp"

class ALight : public ILight {
    public:
        ALight();
        ALight(const Color &color, float intensity);
        ~ALight() override;

        /* Virtual methods */
        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;

        /* Getter */
        Color getColor() const override;
        float getIntensity() const override;

        /* Setter */
        void setColor(const Color &color) override;
        void setIntensity(float intensity) override;

    protected:
        Color _color;
        float _intensity;
};

#endif /* !ALIGHT_HPP_ */
```

6.2 APrimitives.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** APrimitives
*/

#include <memory>
#include <map>
#include <string>
#include <vector>
```

```

#include "IPrimitives.hpp"
#include "ValueType.hpp"
#include "material/IMaterial.hpp"
#include "ILight.hpp"


#ifndef APRIMITIVES_HPP_
#define APRIMITIVES_HPP_

class APrimitives : public IPrimitives {
    public:
        APrimitives() : _position(0, 0, 0), scale(1), material(nullptr),
            _color(-1, -1, -1), _scales(1, 1, 1), _invScales(1, 1, 1), _graph("") {}
        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 ~APrimitives() = default;

        /*Method */
        virtual PixelInfo 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;

        /* Getter */
        Math::Point3D getPosition() const override { return _position; }
        Math::Rot3D getRotation() const override { return _rotation; }
        float getScale() const override { return scale; }
        std::shared_ptr<IMaterial> getMaterial() const override { return material; }
        Math::Vector3D getInvScales() const { return _invScales; }
        const std::vector<std::shared_ptr<ILight»& getLights() const { return _lights; }

        /* Setter */
        void setPosition(const Math::Point3D &newPosition) override { _position = newPosition; }
        void setRotation(const Math::Rot3D &newRotation) override { _rotation = newRotation; }
        void setScale(const float newScale) override {
            scale = newScale;
            computeInvScales();
        }
        void setMaterial(std::shared_ptr<IMaterial> newMaterial) override { material = newMaterial; }
        /* Light Management */
        void addLight(std::shared_ptr<ILight> light) { _lights.push_back(std::move(light)); }
        void clearLights() { _lights.clear(); }
        void applyLights(PixelInfo &pixelInfo, const Math::Ray &ray) const;

        /* Sub Functions */
        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();

        /* Graph Method */
        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);

    protected:
        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;
};

#endif /* !APRIMITIVES_HPP_ */
```

6.3 Camera.hpp

```
/*
```

```
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** ICamera
*/

#include "Point3D.hpp"
#include "Vector3D.hpp"
#include "Rectangle3D.hpp"
#include "Ray.hpp"

#ifndef ICAMERA_HPP_
#define ICAMERA_HPP_

class Camera {
    public:
        Camera();
        Camera(const Camera&) = default;
        Camera(const Rectangle3D& screen);
        ~Camera();

        Math::Point3D _origin;
        Rectangle3D _screen;

        Camera& operator=(const Camera&) = default;

        void updateScreen();
        Math::Ray ray(double u, double v) const;

        /* Getter */
        Math::Point3D getOrigin() const { return _origin; }
        Math::Point3D getRotation() const { return _rotation; }
        Rectangle3D getScreen() const { return _screen; }
        Math::Vector3D getPosition() const { return _position; }
        int getWidth() const { return width; }
        int getHeight() const { return height; }
        float getFieldOfView() const { return fieldOfView; }
        /* Setter */

        void setRotation(Math::Point3D rotation) { _rotation = rotation; }
        void setPosition(Math::Vector3D position) { _position = position; }
        void setResolution(int x, int y) { width = x; height = y; }
        void setHeight(int h) { height = h; }
        void setFieldOfView(float fov) { fieldOfView = fov; }

    protected:
        int width;
        int height;
        float fieldOfView;
        Math::Vector3D _rotation;
        Math::Vector3D _position;

    private:
};

#endif /* !ICAMERA_HPP_ */
```

6.4 Color.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** Color
*/

#ifndef COLOR_HPP_
#define COLOR_HPP_

#include <cstdint>
#include "Vector3D.hpp"
#include "Exception/ColorException.hpp"

class Color  {
 public:
    Color() {
        this->_red = 0;
        this->_green = 0;
        this->_blue = 0;
        this->_transparency = 255;
    }
    ~Color() = default;

```

```cpp
    Color(uint8_t red, uint8_t green, uint8_t blue) {
        this->_red = red;
        this->_green = green;
        this->_blue = blue;
        this->_transparency = 255;
    }

    Color(Math::Vector3D vec) {
        vec.setX(std::min(255.0, std::max(0.0, vec.getX())));
        vec.setY(std::min(255.0, std::max(0.0, vec.getY())));
        vec.setZ(std::min(255.0, std::max(0.0, vec.getZ())));
        this->_red = static_cast<uint8_t>(vec.getX());
        this->_green = static_cast<uint8_t>(vec.getY());
        this->_blue = static_cast<uint8_t>(vec.getZ());
        this->_transparency = 255;
    }

    Color(uint8_t red, uint8_t green, uint8_t blue, uint8_t transparency) {
        this->_red = red;
        this->_green = green;
        this->_blue = blue;
        this->_transparency = transparency;
    }

    Color& operator=(const Math::Vector3D &vec) {
        if (vec.getX() < 0.0 || vec.getX() > 255.0 ||
            vec.getY() < 0.0 || vec.getY() > 255.0 ||
            vec.getZ() < 0.0 || vec.getZ() > 255.0) {
            throw ColorException("Color values must be between 0 and 255");
        }

        this->_red = static_cast<uint8_t>(vec.getX());
        this->_green = static_cast<uint8_t>(vec.getY());
        this->_blue = static_cast<uint8_t>(vec.getZ());
        return *this;
    }

    Color operator*(float scalar) const {
        uint8_t red = static_cast<uint8_t>(std::min(255.0f, std::max(0.0f, static_cast<float>(_red) * scalar)));
        uint8_t green = static_cast<uint8_t>(std::min(255.0f, std::max(0.0f, static_cast<float>(_green) * scalar)));
        uint8_t blue = static_cast<uint8_t>(std::min(255.0f, std::max(0.0f, static_cast<float>(_blue) * scalar)));
        return Color(red, green, blue, _transparency);
    }

    Color operator*(const Color& other) const {
        uint8_t red = static_cast<uint8_t>(std::min(255.0f, (static_cast<float>(_red) * other._red) / 255.0f));
        uint8_t green = static_cast<uint8_t>(std::min(255.0f, (static_cast<float>(_green) * other._green) / 255.0f));
        uint8_t blue = static_cast<uint8_t>(std::min(255.0f, (static_cast<float>(_blue) * other._blue) / 255.0f));
        uint8_t transparency = static_cast<uint8_t>(std::min(255.0f, (static_cast<float>(_transparency) * other._transparency) / 255.0f));
        return Color(red, green, blue, transparency);
    }

    Color operator*=(float scalar) {
        this->_red = static_cast<uint8_t>(std::min(255.0f, std::max(0.0f, static_cast<float>(_red) * scalar)));
        this->_green = static_cast<uint8_t>(std::min(255.0f, std::max(0.0f, static_cast<float>(_green) * scalar)));
        this->_blue = static_cast<uint8_t>(std::min(255.0f, std::max(0.0f, static_cast<float>(_blue) * scalar)));
        return *this;
    }

    Color operator*=(const Color& other) {
        this->_red = static_cast<uint8_t>(std::min(255.0f, (static_cast<float>(_red) * other._red) / 255.0f));
        this->_green = static_cast<uint8_t>(std::min(255.0f, (static_cast<float>(_green) * other._green) / 255.0f));
        this->_blue = static_cast<uint8_t>(std::min(255.0f, (static_cast<float>(_blue) * other._blue) / 255.0f));
        this->_transparency = static_cast<uint8_t>(std::min(255.0f, (static_cast<float>(_transparency) * other._transparency) / 255.0f));
        return *this;
    }

    Color operator+(const Color& other) const {
        uint8_t red = static_cast<uint8_t>(std::min(255, static_cast<int>(_red) + static_cast<int>(other._red)));
        uint8_t green = static_cast<uint8_t>(std::min(255, static_cast<int>(_green) + static_cast<int>(other._green)));
        uint8_t blue = static_cast<uint8_t>(std::min(255, static_cast<int>(_blue) +
static_cast<int>(other._blue)));
        uint8_t transparency = static_cast<uint8_t>(std::min(255, static_cast<int>(_transparency) +
static_cast<int>(other._transparency)));
        return Color(red, green, blue, transparency);
    }

    Color operator+=(const Color& other) {
        this->_red= static_cast<uint8_t>(std::min(255, static_cast<int>(_red) +
static_cast<int>(other._red)));
        this->_green = static_cast<uint8_t>(std::min(255, static_cast<int>(_green) +
static_cast<int>(other._green)));
        this->_blue = static_cast<uint8_t>(std::min(255, static_cast<int>(_blue) +
static_cast<int>(other._blue)));
        this->_transparency = static_cast<uint8_t>(std::min(255, static_cast<int>(_transparency) +
static_cast<int>(other._transparency)));
        return *this;
    }

    Color operator-(const Color& other) const {
        uint8_t red = static_cast<uint8_t>(std::max(0, static_cast<int>(_red) -
static_cast<int>(other._red)));
        uint8_t green = static_cast<uint8_t>(std::max(0, static_cast<int>(_green) -
static_cast<int>(other._green)));
        uint8_t blue = static_cast<uint8_t>(std::max(0, static_cast<int>(_blue) -
static_cast<int>(other._blue)));
        uint8_t transparency = static_cast<uint8_t>(std::max(0, static_cast<int>(_transparency) -
static_cast<int>(other._transparency)));
        return Color(red, green, blue, transparency);
    }

    Color operator-=(const Color& other) {
        this->_red = static_cast<uint8_t>(std::max(0, static_cast<int>(_red) -
static_cast<int>(other._red)));
        this->_green = static_cast<uint8_t>(std::max(0, static_cast<int>(_green) -
static_cast<int>(other._green)));
        this->_blue = static_cast<uint8_t>(std::max(0, static_cast<int>(_blue) -
static_cast<int>(other._blue)));
        this->_transparency = static_cast<uint8_t>(std::max(0, static_cast<int>(_transparency) -
static_cast<int>(other._transparency)));
        return *this;
    }



    void setTransparency(float transparencyValue) {
        this->_transparency = static_cast<uint8_t>(255 * (1.0f - transparencyValue));
    }
    /* Getter */
    uint8_t getRed() const { return this->_red; }
    uint8_t getGreen() const { return this->_green; }
    uint8_t getBlue() const { return this->_blue; }
    uint8_t getTransparency() const { return this->_transparency; }

    /* Setter */
    void setRed(uint8_t red) { this->_red = red; }
    void setGreen(uint8_t green) { this->_green = green; }
    void setBlue(uint8_t blue) { this->_blue = blue; }

 protected:
 private:
    uint8_t _red;
    uint8_t _green;
    uint8_t _blue;

    uint8_t _transparency;
};

#endif /* !COLOR_HPP_ */
```

6.5 Error.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** Error
*/

#ifndef ERROR_HPP_
#define ERROR_HPP_

#include <exception>
#include <iostream>
```

```cpp
#include <string>

class Error : public std::exception {
 public:
    Error(const std::string &msg, const std::string &file, int line)
        : _message(msg), _file(file), _line(line) {
        ErrorFaillureException();
    }

    ~Error() noexcept override = default;

    const char *what() const noexcept override { return _message.c_str(); }

    const char *where() const noexcept { return _file.c_str(); }

    int line() const noexcept { return _line; }

    void ErrorFaillureException() const {
        std::cerr « "Error: " « what() « " in file: " « where()
                  « " at line: " « line() « std::endl;
        exit(84);
    }
    void exitCode(int code) { exit(code); }

 private:
    std::string _message;
    std::string _file;
    int _line;
};

#endif /* !ERROR_HPP_ */
```

6.6 AException.hpp

```cpp
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** AException
*/

#ifndef AEXEPTION_HPP_
#define AEXEPTION_HPP_

#include "IException.hpp"
#include <string>

class AException : public IException {
    public:
        AException(const std::string& type, const std::string& message)
        : _message(message), _type(type) {}
        virtual ~AException() noexcept = default;

        const char* what() const noexcept override {
            return getFormattedMessage().c_str();
        }

        std::string getType() const noexcept override {
            return _type;
        }

        std::string getMessage() const noexcept override {
            return _message;
        }

        std::string getFormattedMessage() const noexcept override {
            return "[" + _type + "] " + _message;
        }

    private:
        std::string _message;
        std::string _type;
};

#endif /* !AEXEPTION_HPP_ */
```

6.7 ColorException.hpp

```cpp
#ifndef COLOREXCEPTION_HPP_
```

```
#define COLOREXCEPTION_HPP_

#include "AException.hpp"

class ColorException : public AException {
    public:
        ColorException(const std::string &message)
            : AException("ColorError", message) {}
};

#endif /* !COLOREXCEPTION_HPP_ */
```

6.8 CommandException.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** CommandException
*/

#ifndef COMMANDEXCEPTION_HPP_
#define COMMANDEXCEPTION_HPP_

#include "AException.hpp"

class CommandException : public AException {
    public:
        CommandException(const std::string& message)
            : AException("CommandError", message) {};
    protected:
    private:
};

#endif /* !COMMANDEXCEPTION_HPP_ */
```

6.9 FactoryException.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** FactoryException
*/

#ifndef FACTORYEXCEPTION_HPP_
#define FACTORYEXCEPTION_HPP_

#include "AException.hpp"

class FactoryException : public AException {
    public:
        FactoryException(const std::string& message)
                : AException("FactoryError", message) {};
    protected:
    private:
};

#endif /* !FACTORYEXCEPTION_HPP_ */
```

6.10 FileException.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** FileConstructionExeption
*/

#ifndef FILECONSTRUCTIONEXEPTION_HPP_
#define FILECONSTRUCTIONEXEPTION_HPP_

#include "AException.hpp"

class FileException : public AException {
    public:
        FileException(const std::string& message)
            : AException("FileError", message) {};
    protected:
    private:
};

#endif /* !FILECONSTRUCTIONEXEPTION_HPP_ */
```

6.11 IException.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** IExeption
*/

#ifndef IEXEPTION_HPP_
#define IEXEPTION_HPP_

#include <exception>
#include <string>


class IException : public std::exception {
    public:
        virtual ~IException() noexcept = default;
        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;
};

#endif /* !IEXEPTION_HPP_ */
```

6.12 materialLoaderException.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** materialLoaderException
*/

#ifndef MATERIALLOADEREXCEPTION_HPP_
#define MATERIALLOADEREXCEPTION_HPP_

#include "AException.hpp"

class MaterialLoaderException : public AException {
    public:
        MaterialLoaderException(const std::string& message)
            : AException("MaterialLoaderError", message) {};
    protected:
    private:
};

#endif /* !MATERIALLOADEREXCEPTION_HPP_ */
```

6.13 MathExeption.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** MathException
*/

#ifndef MATHEXEPTION_HPP_
#define MATHEXEPTION_HPP_

#include "AException.hpp"

class MathException  : public AException {
    public:
        MathExeption(const std::string& message)
            : AException("MathError", message) {};

    protected:
    private:
};

#endif /* !MATHEXEPTION_HPP_ */
```

6.14 SceneException.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** SceneException
*/

#ifndef SCENEEXCEPTION_HPP_
#define SCENEEXCEPTION_HPP_

#include "AException.hpp"

class SceneException : public AException {
    public:
        SceneException(const std::string &message)
            : AException("SceneError", message) {}

    protected:
    private:
};

#endif /* !SCENEEXCEPTION_HPP_ */
```

6.15 Graphs.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** Graphs
*/

#ifndef GRAPHS_HPP_
#define GRAPHS_HPP_

#include "IPrimitives.hpp"
#include "ILight.hpp"

struct GraphsNodePrimitive {
    std::shared_ptr<IPrimitives> _primitives;
    std::vector<std::shared_ptr<GraphsNodePrimitive» _children;

    GraphsNodePrimitive& operator=(const GraphsNodePrimitive& other) {
        if (this == &other)
            return *this;
        _primitives = other._primitives;
        _children.clear();
        for (const auto& child : other._children) {
            if (child)
                _children.push_back(std::make_shared<GraphsNodePrimitive>(*child));
            else
                _children.push_back(nullptr);
        }
        return *this;
    }
    template<typename Func>
    void traverseGraph(const std::shared_ptr<GraphsNodePrimitive>& node, Func &&func) {
        if (!node) return;
        if (node->_primitives)
            func(node->_primitives);
        for (const auto& child : node->_children)
            traverseGraph(child, func);
    }
};


struct GraphsNodeLight {
    std::shared_ptr<ILight> _primitives;
    std::vector<std::shared_ptr<GraphsNodeLight» _children;

    GraphsNodeLight &operator=(const GraphsNodeLight& other) {
        if (this == &other)
            return *this;
        _primitives = other._primitives;
        _children.clear();
        for (const auto& child : other._children) {
            if (child)
                _children.push_back(std::make_shared<GraphsNodeLight>(*child));
            else
                _children.push_back(nullptr);
        }
        return *this;
    }
    template<typename Func>
    void traverseGraph(const std::shared_ptr<GraphsNodeLight>& node, Func &&func) {
        if (!node) return;
        if (node->_primitives)
            func(node->_primitives);
        for (const auto& child : node->_children)
            traverseGraph(child, func);
    }
};



#endif /* !GRAPHS_HPP_ */
```

6.16 IGraphicMode.hpp

```
/*
** EPITECH PROJECT, 2025
** Raytracer
** File description:
** Header
*/

#include <memory>
#include <string>
#include <map>
#include <vector>
#include <SFML/Graphics.hpp>

#ifndef IGRAPHICMODE_HPP_
    #define IGRAPHICMODE_HPP_

class IGraphicMode {
    public:
        virtual ~IGraphicMode() = default;

        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;
};

#endif /* !IGRAPHICMODE_HPP_ */
```

6.17 ILight.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** ILight
*/
```

```

#include <memory>
#include "Color.hpp"
#include "PixelInfo.hpp"
#include "Ray.hpp"
#include "Vector3D.hpp"

#ifndef ILIGHT_HPP_
#define ILIGHT_HPP_


enum class TypeLight { POINT, DIRECTIONAL, PHONG, AMBIENT };

class ILight {
    public:
        virtual ~ILight() = default;

        virtual void addLight(PixelInfo &pixelInfo, const Math::Ray &ray) const = 0;

        // Geters
        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;

        // Setters
        virtual void setColor(const Color &color) = 0;
        virtual void setIntensity(float intensity) = 0;
};

#endif /* !ILIGHT_HPP_ */
```

6.18 Image.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** Raytracer
*/

#include <fstream>
#include <iostream>
#include <vector>

#include "../common/Error.hpp"
#include "Color.hpp"

#ifndef IMAGE_HPP_
#define IMAGE_HPP_

class Image {
 public:
    Image(int width, int height, int maxColorValue = 255);
    ~Image() = default;

    int getWidth() const { return width; }

    int getHeight() const { return height; }

    int getMaxColorValue() const { return maxColorValue; }

    const std::vector<Color> &getData() const { return data; }

    void writeToFilePPM(const std::string &fileName) const;
    void setPixel(int x, int y, const Color &color);
    Color getPixel(int x, int y) const;

 private:
    int width;
    int height;
    int maxColorValue;
    std::vector<Color> data;
};

#endif /* !IMAGE_HPP_ */
```

6.19 IPrimitives.hpp

```
/*
```

```
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** IPrimitives
*/

#include <memory>
#include <optional>

#include "material/IMaterial.hpp"
#include "Point3D.hpp"
#include "Ray.hpp"
#include "Rot3D.hpp"
#include "PixelInfo.hpp"


#ifndef IPRIMITIVES_HPP_
#define IPRIMITIVES_HPP_

enum class Type { SPHERE, PLANE, CYLINDER, CONE, CUBE, TORUS };

class IPrimitives {
 public:
    virtual ~IPrimitives() = default;

    virtual std::optional<double> distance(const Math::Ray &ray) const = 0;
    virtual PixelInfo 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;

    /* Getter */
    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;

    /* Setter */
    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;
 protected:
 private:
};

#endif /* !IPRIMITIVES_HPP_ */
```

6.20 Material.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** Material
*/

#ifndef MATERIAL_HPP_
#define MATERIAL_HPP_
#include "Vector3D.hpp"
#include "Color.hpp"

enum MaterialType {
    FLAT_COLOR,
    TRANSPARENCY_MAT,
    CHESSBOARD,
    FILE_TEXTURE_MAT,
    PERLING_NOISE_MAT
};

struct Material {
    // Basic colors/components (Vec3: r,g,b in [0,1])
    Math::Vector3D ambient; // Ambient reflection coefficient
    Math::Vector3D diffuse; // Diffuse (Lambertian) reflection coefficient
    Math::Vector3D specular; // Specular reflection coefficient
    float shininess;         // Phong exponent (higher = smaller, sharper highlights)

    Color OptionalColor1 = Color(0.0f, 0.0f, 0.0f); // Color for the first chessboard square
    Color OptionalColor2 = Color(0.0f, 0.0f, 0.0f); // Color for the second chessboard square
    float scale = 5.0f; // Scale for the chessboard pattern

    // Reflection & Refraction
    float reflectivity;      // [0,1] Fraction of reflection ray contribution
    float transparency;      // [0,1] Fraction of refracted ray contribution
    float refractiveIndex;      // Snell's law index (1.0 = vacuum/air)
    float opacity;      // [0,1] Fraction of light that is absorbed by the material

    // Optional textures (nullptr if unused)
    std::shared_ptr<std::string> colorTexture;     // Overrides diffuse when present
    std::shared_ptr<std::string> normalMap;        // Perturbs surface normals for bump mapping

    MaterialType materialType = FLAT_COLOR;

    // Constructors
    Material()
        : ambient(0.0f, 0.0f, 0.0f)
        , diffuse(0.0f, 0.0f, 0.0f)
        , specular(0.0f, 0.0f, 0.0f)
        , shininess(32.0f)
        , reflectivity(0.0f)
        , transparency(0.0f)
        , refractiveIndex(1.0f)
        , opacity(1.0f)
        , colorTexture(nullptr)
        , normalMap(nullptr)
    {}

    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)
        : ambient(a)
        , diffuse(d)
        , specular(s)
        , shininess(shin)
        , reflectivity(refl)
        , transparency(trans)
        , refractiveIndex(ior)
        , opacity(1.0f)
        , colorTexture(nullptr)
        , normalMap(nullptr)
    {}
};


#endif /* !MATERIAL_HPP_ */
```

6.21 AMaterial.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** AMaterial
*/

#ifndef AMATERIAL_HPP_
#define AMATERIAL_HPP_
#include "IMaterial.hpp"
#include "../Vector3D.hpp"
#include "../Color.hpp"
#include <memory>

class AMaterial : public IMaterial {
    public:
        AMaterial();
        virtual ~AMaterial() = default;

        void setAmbient(const Math::Vector3D& a) override { ambient = a; }
        void setDiffuse(const Math::Vector3D& d) override { diffuse = d; }
        void setSpecular(const Math::Vector3D& s) override { specular = s; }
        void setShininess(float s) override { shininess = s; }
        void setReflectivity(float r) override { reflectivity = r; }
        void setTransparency(float t) override { transparency = t; }
        void setRefractiveIndex(float i) override { refractiveIndex = i; }
        void setOpacity(float o) override { opacity = o; }
        void setColorTexture(const std::shared_ptr<std::string>& texture) override { colorTexture =
texture; }
        void setNormalMap(const std::shared_ptr<std::string>& map) override { normalMap = map; }
        void setOptionalColor1(const Color& color) override { OptionalColor1 = color; }
        void setOptionalColor2(const Color& color) override { OptionalColor2 = color; }
        void setScale(float s) override { scale = s; }
        void setMaterialType(MaterialType type) override { materialType = type; }

        Math::Vector3D getAmbient() const override { return ambient; }
        Math::Vector3D getDiffuse() const override { return diffuse; }
        Math::Vector3D getSpecular() const override { return specular; }
        float getShininess() const override { return shininess; }
        float getReflectivity() const override { return reflectivity; }
        float getTransparency() const override { return transparency; }
        float getRefractiveIndex() const override { return refractiveIndex; }
        float getOpacity() const override { return opacity; }
        std::shared_ptr<std::string> getColorTexture() const override { return colorTexture; }
        std::shared_ptr<std::string> getNormalMap() const override { return normalMap; }
        Color getOptionalColor1() const override { return OptionalColor1; }
        Color getOptionalColor2() const override { return OptionalColor2; }
        float getScale() const override { return scale; }
        MaterialType getMaterialType() const override { return materialType; }

        virtual Color applyMaterial(const PixelInfo& pixelInfo, float radius, float height, const IPrimitives &primitive) const = 0;
        Math::Vector3D ambient; // Ambient reflection coefficient
        Math::Vector3D diffuse; // Diffuse (Lambertian) reflection coefficient
        Math::Vector3D specular; // Specular reflection coefficient
        float shininess;         // Phong exponent (higher = smaller, sharper highlights)

        Color OptionalColor1 = Color(0.0f, 0.0f, 0.0f); // Color for the first chessboard square
        Color OptionalColor2 = Color(0.0f, 0.0f, 0.0f); // Color for the second chessboard square
        float scale = 5.0f; // Scale for the chessboard pattern

        // Reflection & Refraction
        float reflectivity;      // [0,1] Fraction of reflection ray contribution
        float transparency;      // [0,1] Fraction of refracted ray contribution
        float refractiveIndex;   // Snell’s law index (1.0 = vacuum/air)
        float opacity;           // [0,1] Fraction of light that is absorbed by the material

        // Optional textures (nullptr if unused)
        std::shared_ptr<std::string> colorTexture;      // Overrides diffuse when present
        std::shared_ptr<std::string> normalMap;         // Perturbs surface normals for bump mapping
        MaterialType materialType = MaterialType::FLAT_COLOR;
};

#endif /* !AMATERIAL_HPP_ */
```

6.22 chessboardMat.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** chessboardMat
*/

#ifndef CHESSBOARDMAT_HPP_
#define CHESSBOARDMAT_HPP_
#include "AMaterial.hpp"
#include "../IPrimitives.hpp"
#include "../PixelInfo.hpp"

class ChessboardMat : public AMaterial {
    public:
        ChessboardMat();
        ~ChessboardMat() override = default;
        Color applyMaterial(const PixelInfo& pixelInfo, float radius, float height, const IPrimitives &primitive) const override;

    private:
        Color applySphereChessboard(const PixelInfo& pixelInfo, float radius, float height, const IPrimitives &primitive) const;
        Color applyPlaneChessboard(const PixelInfo& pixelInfo, float radius, float height, const IPrimitives &primitive) const;
        Color applyCylinderChessboard(const PixelInfo& pixelInfo, float radius, float height, const IPrimitives &primitive) const;
        Color applyConeChessboard(const PixelInfo& pixelInfo, float radius, float height, const IPrimitives &primitive) const;
};

#endif /* !CHESSBOARDMAT_HPP_ */
```

6.23 fileTextureMat.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** fileTextureMat
```

```
*/

#ifndef FILETEXTUREMAT_HPP_
#define FILETEXTUREMAT_HPP_

#include "AMaterial.hpp"

class FileTextureMat : public AMaterial {
    public:
        FileTextureMat();
        ~FileTextureMat() override = default;

        Color applyMaterial(const PixelInfo& pixelInfo, float radius, float height, const IPrimitives
&primitive) const override;

    private:
        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;
};

#endif /* !FILETEXTUREMAT_HPP_ */
```

6.24 flatColorMat.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** flatColorMat
*/

#ifndef FLATCOLORMAT_HPP_
#define FLATCOLORMAT_HPP_
#include "AMaterial.hpp"
#include "../IPrimitives.hpp"
#include "../PixelInfo.hpp"

class FlatColorMat : public AMaterial {
    public:
        FlatColorMat();
        ~FlatColorMat() override = default;

        Color applyMaterial(const PixelInfo& pixelInfo, float radius, float height, const IPrimitives
&primitive) const override;
};

#endif /* !FLATCOLORMAT_HPP_ */
```

6.25 IMaterial.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** IMaterial
*/

#ifndef IMATERIAL_HPP_
#define IMATERIAL_HPP_

#include "../Vector3D.hpp"
#include "../Color.hpp"
#include "../PixelInfo.hpp"
#include <memory>

// Forward declaration pour éviter les dépendances circulaires
class IPrimitives;

```

```
enum MaterialType {
    FLAT_COLOR,
    TRANSPARENCY_MAT,
    CHESSBOARD,
    FILE_TEXTURE_MAT,
    PERLING_NOISE_MAT
};

class IMaterial {
    public:
        virtual ~IMaterial() = default;

        // Setters
        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;

        // Getters
        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;

        // Material application methods
        virtual Color applyMaterial(const PixelInfo& pixelInfo, float radius, float height, const
IPrimitives &primitive) const = 0;
};

#endif /* !IMATERIAL_HPP_ */
```

6.26 perlingNoiseMat.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** perlingNoiseMat
*/

#ifndef PERLINGNOISEMAT_HPP_
#define PERLINGNOISEMAT_HPP_

#include "AMaterial.hpp"
#include <cmath>
#include <random>
#include <vector>

class PerlingNoiseMat : public AMaterial {
    public:
        PerlingNoiseMat();
        ~PerlingNoiseMat() override = default;

        Color applyMaterial(const PixelInfo& pixelInfo, float radius, float height, const IPrimitives
&primitive) const override;

    private:
        float frequency;
        float amplitude;
        int octaves;
        float persistence;
};

#endif /* !PERLINGNOISEMAT_HPP_ */
```

6.27 transparencyMat.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** transparencyMat
*/

#ifndef TRANSPARENCYMAT_HPP_
#define TRANSPARENCYMAT_HPP_

#include "AMaterial.hpp"

class TransparencyMat : public AMaterial {
    public:
        TransparencyMat();
        ~TransparencyMat() override = default;

        Color applyMaterial(const PixelInfo& pixelInfo, float radius, float height, const IPrimitives
&primitive) const override;
};
#endif /* !TRANSPARENCYMAT_HPP_ */
```

6.28 PixelInfo.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** PixelInfo
*/

#ifndef PIXELINFO_HPP_
#define PIXELINFO_HPP_

#include "Color.hpp"
#include "Vector3D.hpp"

struct PixelInfo {
    Color _color;
    Math::Vector3D _normalizedVector;
    double _distance;
    bool _isHit;
    Math::Vector3D _pos;
    float _light_intensity;
    Color _light_color;

    PixelInfo() :
        _color(),
        _normalizedVector(),
        _distance(0.0),
        _isHit(false),
        _pos(),
        _light_intensity(0.0f),
        _light_color()
    {}

    PixelInfo(const Color& color, const Math::Vector3D& normalVector, double distance,
              bool isHit, const Math::Vector3D& position, float lightIntensity, const Color&
colorLight) :
        _color(color),
        _normalizedVector(normalVector),
        _distance(distance),
        _isHit(isHit),
        _pos(position),
        _light_intensity(lightIntensity),
        _light_color(colorLight)
    {}
};

#endif /* !PIXELINFO_HPP_ */
```

6.29 Point3D.hpp

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

```
** Point3D
*/

#ifndef POINT3D_HPP_
#define POINT3D_HPP_

#include "Vector3D.hpp"

namespace Math {

class Point3D {
   public:
      Point3D();
      Point3D(double x, double y, double z);
      ~Point3D() = default;

      /* Getter */
      double getX() const;
      double getY() const;
      double getZ() const;

      /* Setter */
      void setX(double x);
      void setY(double y);
      void setZ(double z);

      /* Methods */
      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);
      /* Operators */
      Point3D &operator+=(const Vector3D &vector);
      Point3D &operator-=(const Vector3D &vector);

   protected:
   private:
      double x;
      double y;
      double z;
};

}  // namespace Math

#endif /* !POINT3D_HPP_ */
```

6.30 Random.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** Random
*/

#ifndef RANDOM_HPP_
#define RANDOM_HPP_

#include <cmath>
#include <random>
#include <functional>

namespace Math {

class Random {
 public:
    static double normalDistribution(uint32_t& state);
    static float pcg(uint32_t& state);
};

}  // namespace Math

#endif /* !RANDOM_HPP_ */
```

6.31 Ray.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** Ray
*/

#include "Color.hpp"
#include "Point3D.hpp"
#include "Vector3D.hpp"

#ifndef RAY_HPP_
#define RAY_HPP_

namespace Math {
class Ray {
 public:
    Ray() : _origin(), _direction() {}
    Ray(Point3D origin, Vector3D direction) : _origin(origin), _direction(direction),
_refraction_index(1) {}
    Ray(Point3D origin, Vector3D direction, double Refraction_index) : _origin(origin),
_direction(direction), _refraction_index( Refraction_index) {}
    ~Ray() = default;

    /* Setter */
    void setOrigin(Point3D origin) { this->_origin = origin; }
    void setDirection(Vector3D direction) { this->_direction = direction; }
    void setRefractionIndex(double Refraction_index) { this->_refraction_index = Refraction_index; }

    /* Getter */
    Point3D getOrigin() const { return this->_origin; }
    Vector3D getDirection() const { return this->_direction; }
    double getRefractionIndex() const { return this->_refraction_index; }

 private:
    Point3D _origin;
    Vector3D _direction;
    double _refraction_index;
};
} // namespace Math

class Ray {
 public:
    Ray() : _ray(), _color(), _intensity(0) {}
    Ray(Math::Point3D origin, Math::Vector3D direction, Color color, double intensity)
        : _ray(origin, direction), _color(color), _intensity(intensity) {}
    ~Ray() = default;

    /* Setter */
    void setOrigin(Math::Point3D origin);
    void setDirection(Math::Vector3D direction);
    void setColor(Color color);
    void setIntensity(double intensity);

    /* Getter */
    Math::Point3D getOrigin() const;
    Math::Vector3D getDirection() const;
    Color getColor() const;
    double getIntensity() const;

 private:
    Math::Ray _ray;
    Color _color;
    double _intensity;
};

#endif /* !RAY_HPP_ */
```

6.32 Rectangle3D.hpp

```
/*
** EPITECH PROJECT, 2025
** Raytracer
** File description:
** Header
*/

#include "Point3D.hpp"
#include "Rot3D.hpp"
#include "Vector3D.hpp"

```

```
#ifndef RECTANGLE3D_HPP_
    #define RECTANGLE3D_HPP_

class Rectangle3D {
    public:
        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);
        ~Rectangle3D();

        Math::Point3D _origin;
        Math::Vector3D _bottom_side;
        Math::Vector3D _left_side;
        int getWidth() const;
        int getHeight() const;
        Math::Point3D pointAt(double u, double v) const;

    protected:
    private:
};

#endif /* !RECTANGLE3D_HPP_ */
```

6.33 Rot3D.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** Raytracer
*/

#ifndef ROT3D_HPP
#define ROT3D_HPP

#include "Vector3D.hpp"

namespace Math {
class Rot3D {
 public:
    double x_pitch;
    double z_yaw;
    double y_roll;

    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 { return x_pitch; }
    double getY() const { return y_roll; }
    double getZ() const { return z_yaw; }
};
}  // namespace Math

#endif  // ROT3D_HPP
```

6.34 Scene.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** IScene
*/

#include <memory>
```

```
#include <vector>
#include "Camera.hpp"
#include "APrimitives.hpp"
#include "Graphs.hpp"

#ifndef ISCENE_HPP_
#define ISCENE_HPP_

class Scene {
    public:
        Scene();
        Scene(std::shared_ptr<Camera>, std::vector<std::shared_ptr<IPrimitives»);
        ~Scene();

        /* Getter */
        std::shared_ptr<Camera> getCamera() const;
        std::shared_ptr<GraphsNodePrimitive> getPrimitives() const;
        std::shared_ptr<GraphsNodeLight> getLights() const;
        int cameraWidth() const;
        int cameraHeight() const;
        float getAmbientLight() const;

        /* Setter */
        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);
      //  void setAmbientLight(float ambientLight);

       private:
        std::shared_ptr<Camera> _camera;
        std::vector<std::shared_ptr<IPrimitives» _primTemp;
       // std::vector<std::shared_ptr<IPrimitives» _primitives;
        std::shared_ptr<GraphsNodePrimitive> _primitives;
        std::shared_ptr<GraphsNodeLight> _lights;
      //  std::vector<std::shared_ptr<ILight» _lights;
        float _ambientLight;
};

#endif /* !ISCENE_HPP_ */
```

6.35 ValueType.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** ValueType
*/


#include <memory>
#include <string>
#include <variant>

#include "IPrimitives.hpp"
#include "Vector2D.hpp"
#include "Vector3D.hpp"

#ifndef VALUETYPE_HPP_
#define VALUETYPE_HPP_

enum ObjectType {
    TYPE_CAMERA,
    TYPE_LIGHT,
    TYPE_PRIMITIVE,
    TYPE_IMPORTED_SCENE,
    TYPE_GRAPH,
    TYPE_MATERIAL,
    TYPE_UNDEFINED
};

using ValueType = std::variant<int, float, double, std::string, bool, Math::Vector2D, Math::Vector3D,
ObjectType>;

enum ValueFormat { FORMAT_SIMPLE, FORMAT_VECTOR2D, FORMAT_VECTOR3D};

typedef enum ValueType_s {
    NAME,
    TYPE,
    POSITION,
    ROTATION,
    SCALE,
    SCALES,
    COLOR,
    COLOR_CHESS_1,
    COLOR_CHESS_2,
    RADIUS,
    FIELD_OF_VIEW,
    RESOLUTION,
    AXIS,
    HEIGHT,
    MATERIAL,
    GRAPH,
    PATH,
    AMBIENT,
    SPECULAR,
    DIFFUSE,
    REFLECTION,
    TRANSPARENCY,
    REFRACTION_INDEX,
    DIRECTION,
    INTENSITY,
    SHININESS,
    MAJOR_RADIUS,
    MINOR_RADIUS
} ValueType_t;



#endif /* !VALUETYPE_HPP_ */
```

6.36 Vector2D.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** Vector2D
*/

#ifndef VECTOR2D_HPP_
#define VECTOR2D_HPP_

namespace Math {

class Vector2D {
    public:
        Vector2D();
        Vector2D(double x, double y) : _x(x), _y(y) {}
        ~Vector2D() = default;

    // getter
    double getX() const { return _x; }
    double getY() const { return _y; }

    // setter
    void setX(double x) { _x = x; }
    void setY(double y) { _y = y; }

 protected:
 private:
    double _x;
    double _y;
};

}  // namespace Math

#endif /* !VECTOR2D_HPP_ */
```

6.37 Vector3D.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** Vector3D
*/

#include <cmath>
#include <iostream>
#include <vector>
#include <functional>
```

```

#include "Point3D.hpp"
#include "Random.hpp"

#ifndef VECTOR3D_HPP_
#define VECTOR3D_HPP_

class Color;

namespace Math {

class Point3D;

class Vector3D {
 public:
    Vector3D();
    Vector3D(uint32_t& state);
    Vector3D(double x, double y, double z);
    Vector3D(const Math::Point3D &point);
    ~Vector3D() = default;

    /* Getter */
    double getX() const;
    double getY() const;
    double getZ() const;

    /* Setter */
    void setX(double x);
    void setY(double y);
    void setZ(double z);

    /* Methods */
    double length() const;
    double dot(const Vector3D &other) const;

    // Add cross product
    Vector3D cross(const Vector3D &other) const;

    Vector3D normalize() const;

    Vector3D RandomInHemisphere(uint32_t& state) const;

    Vector3D getAnyPerpendicular() const;

    /* Operators */
    Vector3D operator-() const;
    Vector3D operator+(const Vector3D &other) const;
    Vector3D operator-(const Vector3D &other) const;

    Vector3D operator+=(const Vector3D &other);
    Vector3D operator+=(const Color &other);
    Vector3D operator-=(const Vector3D &other);

    Vector3D operator*(const Vector3D &other) const;
    Vector3D operator*=(const Vector3D &other);

    Vector3D operator/(const Vector3D &other) const;
    Vector3D operator/=(const Vector3D &other);

    /* Operator and Scalar */
    Vector3D operator*(double scalar) const;
    Vector3D &operator*=(double scalar);
    Vector3D operator/(double scalar) const;
    Vector3D &operator/=(double scalar);

 protected:
    double x;
    double y;
    double z;

 private:
};

inline Vector3D operator*(double scalar, const Vector3D &vec) {
    return Vector3D(vec.getX() * scalar, vec.getY() * scalar,
                    vec.getZ() * scalar);
}

inline Vector3D operator/(double scalar, const Vector3D &vec) {
   return Vector3D(vec.getX() / scalar, vec.getY() / scalar,
                   vec.getZ() / scalar);
}

}  // namespace Math

#endif /* !VECTOR3D_HPP_ */
```

6.38 DLLoader.hpp

```cpp
/*
** EPITECH PROJECT, 2025
** arcade
** File description:
** DLLoader
*/

#ifndef DLLOADER_HPP_
#define DLLOADER_HPP_

#include <dlfcn.h>
#include <iostream>
#include <ostream>
#include "ILoader.hpp"

template <typename T>

class DLLoader  : public ILoader {
    private:
        void *_handler = nullptr;

    public:
        ~DLLoader() = default;

        void *getHandler() const override {
            return _handler;
        };
        void *Open(const char *path, int flag) override {
            _handler = dlopen(path, flag);
            return _handler;
        };
        void *Symbol(const char *symbolName) override {
            void *symbol = dlsym(_handler, symbolName);
            const char *error = dlerror();
            if (error) {
                std::cerr « "dlerror: " « error « std::endl;
                return nullptr;
            }
            return symbol;
        };
        T getSymbol(const char *symbolName) {
            return reinterpret_cast<T>(dlsym(_handler, symbolName));
        };
        int Close() override{
            if (_handler == nullptr)
                return -1;
            return dlclose(_handler);
        };
        const char *Error() override {
            return dlerror();
        };
};

#endif /* !DLLOADER_HPP_ */
```

6.39 ILoader.hpp

```cpp
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** ILoader
*/

#ifndef ILoader_HPP_
#define ILoader_HPP_


class ILoader {
    public:
        ~ILoader() = default;

        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;

    protected:
    private:
};
```

```

#endif /* !ILoader_HPP_ */
```

6.40 GraphicMode.hpp

```
/*
** EPITECH PROJECT, 2025
** Raytracer
** File description:
** Graphic Mode
*/

#include <memory>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include "../../common/IGraphicMode.hpp"

#ifndef GRAPHICMODE_HPP_
    #define GRAPHICMODE_HPP_

class GraphicMode : public IGraphicMode {
    public:
        GraphicMode();
        ~GraphicMode();

        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:
        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;
};

extern "C" {
    std::shared_ptr<GraphicMode> createInstance();
}

#endif /* !GRAPHICMODE_HPP_ */
```

6.41 CameraFactory.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** PrimitiveFactory
*/

#ifndef CAMERAFACTORY_HPP_
#define CAMERAFACTORY_HPP_

#include <vector>
#include <functional>
#include "IFactory.hpp"
#include "../../lib/DLLoader.hpp"
#include "../../common/Camera.hpp"
#include "../../common/Exception/FactoryException.hpp"
```

```
class CameraFactory : public IFactory<Camera> {
public:
    CameraFactory();
    ~CameraFactory();

    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:
    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;
};

#endif /* !CAMERAFACTORY_HPP_ */
```

6.42 FactoryManager.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** FactoryManager
*/

#ifndef FACTORYMANAGER_HPP_
#define FACTORYMANAGER_HPP_

#include <memory>
#include "PrimitiveFactory.hpp"
#include "CameraFactory.hpp"
#include "LightFactory.hpp"
#include "../../common/Graphs.hpp"
#include "../../common/Exception/FactoryException.hpp"

class FactoryManager {
public:
    FactoryManager();
    ~FactoryManager() = default;

    std::shared_ptr<PrimitiveFactory> getPrimitiveFactory() { return _primitiveFactory; }
    std::shared_ptr<CameraFactory> getCameraFactory() { return _cameraFactory; }

    void initializeFactories();

    void createObjectsFromConfig(const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType»>&
objectsConfig);

    std::shared_ptr<GraphsNodePrimitive> getPrimitives() const { return _primitives; }
    std::vector<std::shared_ptr<ILight» getLights() const { return _lights; }
    std::shared_ptr<Camera> getCamera() const { return _camera; }
    float getAmbientLight() const { return _ambientLight; }

private:
    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;
};

#endif /* !FACTORYMANAGER_HPP_ */
```

6.43 IFactory.hpp

```
#ifndef IFACTORY_HPP_
#define IFACTORY_HPP_

#include <memory>
#include <string>
#include <map>
#include "../../common/ValueType.hpp"
#include "../../common/ILight.hpp"

template <typename T>
class IFactory {
public:
    virtual ~IFactory() = default;
    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;
};

#endif /* !IFACTORY_HPP_ */

```

6.44 LightFactory.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** LightFactory
*/

#ifndef LIGHTFACTORY_HPP_
#define LIGHTFACTORY_HPP_

#include <vector>
#include <memory>
#include <functional>
#include "IFactory.hpp"
#include "../../common/ALight.hpp"
#include "../../lib/DLLoader.hpp"
#include "../../common/Exception/FactoryException.hpp"


class LightFactory : public IFactory<ILight> {
    public:
        LightFactory();
        ~LightFactory();

        std::shared_ptr<ILight> 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<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, ValueType>>>&)> 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);
    protected:
    private:
        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;
};

#endif /* !LIGHTFACTORY_HPP_ */
```

6.45 PrimitiveFactory.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** PrimitiveFactory
*/

#ifndef PRIMITIVEFACTORY_HPP_
#define PRIMITIVEFACTORY_HPP_

#include <vector>
#include <memory>
#include <functional>
#include "IFactory.hpp"
#include "../../common/APrimitives.hpp"
#include "../../lib/DLLoader.hpp"
#include "../../common/Exception/FactoryException.hpp"

class PrimitiveFactory : public IFactory<IPrimitives> {
public:
    PrimitiveFactory();
    ~PrimitiveFactory();

    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:
    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;
};

#endif /* !PRIMITIVEFACTORY_HPP_ */
```

6.46 DirectionalLight.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** DirectionalLight
*/

#ifndef DIRECTIONALLIGHT_HPP_
#define DIRECTIONALLIGHT_HPP_

#include "../../common/ALight.hpp"
#include "../../common/Vector3D.hpp"
#include "../../common/ValueType.hpp"
#include <map>
#include <optional>

class DirectionalLight : public ALight {
    public:
        DirectionalLight();
        DirectionalLight(const Color &color, float intensity,
                        const Math::Vector3D &direction, float radius);
        DirectionalLight(std::shared_ptr<std::map<ValueType_t, ValueType» map);
        ~DirectionalLight();

        void addLight(PixelInfo &pixelInfo, const Math::Ray &ray) const override;
        TypeLight getTypeLight() const override;
        Math::Vector3D getDirection() const override;
        float getRadius() const override;

        // Setters
        void setDirection(const Math::Vector3D &direction);

    private:
        Math::Vector3D _direction;
        Color _color;
        float _intensity;
        float _radius;
};


#endif /* !DIRECTIONALLIGHT_HPP_ */
```

6.47 PhongLight.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** PhongLight
*/

#ifndef PHONGLIGHT_HPP_
#define PHONGLIGHT_HPP_

#include "../../common/ALight.hpp"
#include "../../common/Point3D.hpp"

#include <map>

class PhongLight : public ALight {
    public:
        PhongLight();
        PhongLight(std::shared_ptr<std::map<ValueType_t, ValueType» map);
        PhongLight(const Color &color, float intensity,
                   const Math::Vector3D &direction, float radius, float shininess);
        ~PhongLight();

        void addLight(PixelInfo &pixelInfo, const Math::Ray &ray) const override;
        TypeLight getTypeLight() const override;
        Math::Vector3D getDirection() const override;
        float getRadius() const override;
        // Getters
        float getShininess() const;

        // Setters
        void setDirection(const Math::Vector3D &position);
        void setShininess(float shininess);

    private:
        Math::Vector3D _direction;
        Color _color;
        float _intensity;
        float _radius;
        float _shininess;
};

#endif /* !PHONGLIGHT_HPP_ */
```

6.48 PointLight.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** PointLight
*/

#ifndef POINTLIGHT_HPP_
#define POINTLIGHT_HPP_

#include "../../common/ALight.hpp"
#include "../../common/Point3D.hpp"

class PointLight : public ALight {
    public:
        PointLight();
        PointLight(const Color &color, float intensity,
                   const Math::Point3D &position, float radius);
        ~PointLight();

        /* Override method */
        TypeLight getTypeLight() const override;
        void addLight(PixelInfo &pixelInfo, const Math::Ray &ray) const override;

        // Getters
        Math::Point3D getPosition() const;
        float getRadius() const;

        // Setters
        void setPosition(const Math::Point3D &position);
        void setRadius(float radius);

    private:
        Math::Point3D _position;
        float _radius;
};

#endif /* !POINTLIGHT_HPP_ */
```

6.49 IMediator.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** IMediator
*/

#ifndef IMEDIATOR_HPP_
```

```
#define IMEDIATOR_HPP_

#include <functional>

class IMediator {
    public:
        virtual void addTask(std::function<void()> task) = 0;
        virtual void executeTasks() = 0;
        virtual void waitForCompletion() = 0;
    protected:
    private:
};

#endif /* !IMEDIATOR_HPP_ */
```

6.50 RayMediator.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** RayMediator
*/

#ifndef RAYMEDIATOR_HPP_
#define RAYMEDIATOR_HPP_


#include <thread>
#include <functional>
#include <vector>
#include <mutex>
#include <condition_variable>

#include "IMediator.hpp"

class RayMediator : public IMediator {
    public:
        RayMediator();
        ~RayMediator();

        void addTask(std::function<void()> task) override;
        void executeTasks() override;
        void waitForCompletion() override;

    private:
        std::vector<std::thread> _threads;
        std::vector<std::function<void()» _tasks;
        std::mutex _mutex;
        std::condition_variable _condition;
        bool _stop = false;
};

#endif /* !RAYMEDIATOR_HPP_ */
```

6.51 ConfigNode.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** ConfigNode
*/

#ifndef CONFIGNODE_HPP_
#define CONFIGNODE_HPP_

#include <iostream>
#include <map>
#include <variant>

#include "../../common/ValueType.hpp"

enum NodeType { TypeUnknown, TypeGroup, TypeArray, TypeValue, TypeList };

class ConfigNode {
 public:
    ConfigNode() : isValue(false), type(TypeUnknown), _name("") {}
    ~ConfigNode() = default;
```

```

        std::map<std::string, ConfigNode> children;
        ValueType value;
        bool isValue;
        NodeType type;
        std::string _name;

    bool hasChild(const std::string &name) const;
   // template <typename T>
    //T getValue(const T &defaultValue = T()) const;
};

#endif /* !CONFIGNODE_HPP_ */
```

6.52 ConfigParser.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** ConfigParser
*/

#ifndef CONFIGPARSER_HPP_
#define CONFIGPARSER_HPP_

#include <libconfig.h++>
#include <string>
#include <memory>

#include "ConfigNode.hpp"

using namespace libconfig;

class ConfigParser {
 public:
    ConfigParser();
    ~ConfigParser();
    bool loadConfig(const std::string &filename, ConfigNode &rootNode);

 protected:
    void buildConfigTree(const Setting &setting, std::shared_ptr<ConfigNode> node);

 private:
    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);

    Config file;
};

#endif /* !CONFIGPARSER_HPP_ */
```

6.53 ObjectConstructor.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** ObjectFactory
*/

#ifndef OBJECTFACTORY_HPP_
#define OBJECTFACTORY_HPP_

#include <map>
#include <string>
#include <vector>
#include <memory>

#include "ConfigNode.hpp"
#include "PropertyTypes.hpp"
#include "ObjectErrorHandling.hpp"

```

```
class ObjectConstructor {
 public:
    ObjectConstructor();
    ~ObjectConstructor();

        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:
        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 ValueType_t& key,
                const ValueType& value,
                ValueDataType dataType);
        void handleVector2DValue(std::shared_ptr<std::map<ValueType_t,
                ValueType» object, const ValueType_t& key,
                const ConfigNode& node,
                const std::vector<std::string>& components,
                ValueDataType dataType);
        void handleVector3DValue(std::shared_ptr<std::map<ValueType_t, ValueType» object,
                const ValueType_t& key,
                const ConfigNode& node,
                const std::vector<std::string>& components,
                ValueDataType dataType);
        ValueType convertValue(const ValueType& value, ValueDataType dataType);

        void initShapeDefinitions();

        ObjectErrorHandling _errorHandler;
        std::map<std::string, PropertyConfig> _propertyTypeMap;
        std::vector<std::shared_ptr<std::map<ValueType_t, ValueType»> _objects;
        std::vector<ShapeDefinition> _shapeDefinitions;

};

#endif /* !OBJECTFACTORY_HPP_ */
```

6.54 ObjectErrorHandling.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** ObjectErrorHandling
*/

#ifndef OBJECTERRORHANDLING_HPP_
#define OBJECTERRORHANDLING_HPP_

#include <iostream>
#include <map>
#include <variant>
#include <memory>
#include "ConfigNode.hpp"
#include "PropertyTypes.hpp"
#include "../../common/ValueType.hpp"

class ObjectErrorHandling {
    public:
        ObjectErrorHandling();
        ~ObjectErrorHandling();

        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);

    protected:
    private:
        std::vector<ShapeDefinition> _shapeDefinitions;
        std::map<std::string, PropertyConfig> _propertyTypeMap;

        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;

};

#endif /* !OBJECTERRORHANDLING_HPP_ */
```

6.55 Parser.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** Parser
*/

#ifndef PARSER_HPP_
#define PARSER_HPP_

#include <memory>
#include <string>
#include <vector>
#include <filesystem>
#include "ConfigParser.hpp"
#include "ObjectConstructor.hpp"

class Parser {
 public:
    Parser();
    Parser(const std::string &filename);
    ~Parser();
    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() { return std::make_shared<ConfigNode>(rootNode); }

    void printMap() const;
 private:
    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);

    ConfigNode rootNode;
    ConfigParser configParser;
    ObjectConstructor _objectConstructor;
};
```

```

#endif /* !PARSER_HPP_ */
```

6.56 PropertyTypes.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** PropertyTypes
*/

#ifndef PROPERTYTYPES_HPP_
#define PROPERTYTYPES_HPP_

#include <map>
#include <string>
#include <vector>

#include "../../common/ValueType.hpp"

enum ValueDataType {
    TYPE_INT,
    TYPE_FLOAT,
    TYPE_STRING,
    TYPE_BOOL,
    TYPE_DOUBLE
};

struct PropertyInfo {
    ValueType_t type;
    ValueFormat format;
    std::vector<std::string> components;
    ValueDataType dataType;
};

struct ShapeDefinition {
    std::string name;
    std::vector<std::string> mandatory;
    std::vector<std::string> optional;
    ObjectType objectType;
};

struct PropertyConfig {
    ValueType_t type;
    ValueFormat format;
    std::vector<std::string> components;
    ValueDataType dataType;
};

#endif /* !PROPERTYTYPES_HPP_ */
```

6.57 ValueConverter.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** ValueConverter
*/

#ifndef VALUECONVERTER_HPP_
#define VALUECONVERTER_HPP_

#include "../../common/Vector2D.hpp"
#include "../../common/Vector3D.hpp"
#include "PropertyTypes.hpp"

class ValueConverter {
 public:
    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);
};

#endif /* !VALUECONVERTER_HPP_ */
```

6.58 Cone.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** Cone
*/

#ifndef CONE_HPP_
#define CONE_HPP_

#include <map>
#include <optional>
#include "../../common/APrimitives.hpp"
#include "../../common/ValueType.hpp"

class Cone : public APrimitives {
    public:
        Cone();
        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);
        ~Cone();
        /* Setter */
        void setBaseRadius(float radius);
        void setHeight(float height);

        /* Method */
        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;

        /* Getter */
        float getBaseRadius() const;
        Type getType() const override;
        float getHeight() const;


    protected:
    private:
        float _baseRadius;
        float _height;
        double _distance;

        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;
 };

#endif /* !CONE_HPP_ */
```

6.59 Cylinder.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** Cylinder
*/

#ifndef CYLINDER_HPP_
#define CYLINDER_HPP_

#include <map>
#include "../../common/APrimitives.hpp"
#include "../../common/ValueType.hpp"
// #include "../../common/Color.hpp"

class Cylinder : public APrimitives {
    public:
        Cylinder();
        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);

        ~Cylinder();

        /* Setter */
        void setBaseRadius(float radius);
        void setHeight(float height);

        /* Getter */
        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;


    private:
        float _baseRadius;
        float _height;
        double _distance;
};

#endif /* !CYLINDER_HPP_ */

```

6.60 Plane.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** Plane
*/

#include <memory>
#include <map>
#include "../../common/ValueType.hpp"
#include "../../common/material/IMaterial.hpp"
#include "../../common/APrimitives.hpp"
#include "../../common/Point3D.hpp"
#include "../../common/Rot3D.hpp"
#include "../../common/Vector3D.hpp"

class Plane : public APrimitives {
 public:
    Plane();
    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);
    ~Plane() 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;
    /* Getter */
    Type getType() const override;

    /* Setter */
    void setRotation(const Math::Rot3D &newRotation) override;
    void updateNormal();

 private:
    double _distance;
    Math::Vector3D _normal;
    std::string _axe;
};
```

6.61 Sphere.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** Sphere
*/

#include <memory>
#include <map>
#include "../../common/material/IMaterial.hpp"
#include "../../common/APrimitives.hpp"
```

```
#include "../../common/Point3D.hpp"
#include "../../common/Rot3D.hpp"
#include "../../common/Vector3D.hpp"
#include "../../common/ValueType.hpp"

class Sphere : public APrimitives {
 public:
    Sphere();
    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);
    ~Sphere() 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;

    /* Getter */
    Type getType() const override;

 private:
      double _distance;
      double radius;
};
```

6.62 Torus.hpp

```
/*
** EPITECH PROJECT, 2025
** Raytracer
** File description:
** Torus
*/

#include "../../common/APrimitives.hpp"
#include "../../common/Ray.hpp"
#include <map>
#include <memory>
#include <optional>
#include <vector>

#ifndef TORUS_HPP_
    #define TORUS_HPP_

class Torus : public APrimitives {
    public:
        Torus();
        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);
        // Add 2-argument constructor for plugin factory
        Torus(std::shared_ptr<std::map<ValueType_t, ValueType» map,
          const std::vector<std::shared_ptr<std::map<ValueType_t, ValueType»> &graphSceneList);
        ~Torus() override;

        /* Setter */
        void setMajorRadius(float radius);
        void setMinorRadius(float radius);

        /* Getter */
        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;

    private:
        float _majorRadius;
        float _minorRadius;
        double _distance;

        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;
};

#endif /* !TORUS_HPP_ */
```

6.63 Raytracer.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** Raytracer
*/

#include <iostream>
#include <map>
#include <memory>
#include <functional>
#include <vector>
#include <optional>

#include "../common/APrimitives.hpp"
#include "../common/ValueType.hpp"

#include "../common/Error.hpp"
#include "../common/Image.hpp"
#include "../common/Camera.hpp"
#include "../common/Scene.hpp"
#include "../lib/DLLoader.hpp"
#include "../lib/ILoader.hpp"
#include "factory/FactoryManager.hpp"
#include "../lib/SFML/GraphicMode.hpp"
#include "../common/Graphs.hpp"

#ifndef RAYTRACER_HPP_
#define RAYTRACER_HPP_

struct InfoPixelDisplay {
      double distance;
      Color color;
      float transparency;
};

struct DropShadowInfo {
    Math::Vector3D position;
    float darkness; // 0 to 1
};

class Raytracer {
      public:
         Raytracer();
         ~Raytracer();

         /* Getter */
         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;

         /* Setter */
         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();

         /* Methods */
         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);

         /* Image Method */
         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 averageImages(const std::vector<std::shared_ptr<Image»& images);
         void loopThruType();

```

```
        /* New image processing methods */
        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;
   protected:
   private:
      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; // Number of samples per pixel for antialiasing
   public:
};

#endif /* !RAYTRACER_HPP_ */
```

6.64 Utils.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** Utils
*/

#include <iostream>
#include <memory>

#ifndef UTILS_HPP_
#define UTILS_HPP_

class Utils {
 public:
    Utils();
    ~Utils();
    static void helper();

 protected:
 private:
};

#endif /* !UTILS_HPP_ */
```