My Project

Generated by Doxygen 1.8.16

1 Hierarchical Index 1

1.1 Class Hierarchy 1

2 Class Index 3

2.1 Class List 3

3 Class Documentation 5

3.1 RayTracer Class Reference 5

3.1.1 Detailed Description 5

3.1.2 Constructor & Destructor Documentation 5

3.1.2.1 RayTracer() `[1/2]` 5

3.1.2.2 RayTracer() `[2/2]` 5

3.1.3 Member Function Documentation 6

3.1.3.1 getPixels() 6

3.1.3.2 getScene() `[1/2]` 7

3.1.3.3 getScene() `[2/2]` 7

3.1.3.4 getView() 7

3.2 Shape3D Class Reference 7

3.2.1 Member Function Documentation 8

3.2.1.1 ambient() 8

3.2.1.2 color() 8

3.2.1.3 intersect() 8

3.2.1.4 normal() 9

3.2.1.5 position() 9

3.3 Sphere Class Reference 9

3.3.1 Constructor & Destructor Documentation 10

3.3.1.1 Sphere() `[1/2]` 10

3.3.1.2 Sphere() `[2/2]` 10

3.3.2 Member Function Documentation 10

3.3.2.1 ambient() 10

3.3.2.2 color() 11

3.3.2.3 intersect() 11

3.3.2.4 normal() 11

3.3.2.5 position() 12

3.3.2.6 radius() 12

3.4 Vector Class Reference 12

3.4.1 Detailed Description 13

3.4.2 Constructor & Destructor Documentation 13

3.4.2.1 Vector() `[1/2]` 13

3.4.2.2 Vector() `[2/2]` 13

3.4.3 Member Function Documentation 14

3.4.3.1 add() 14

3.4.3.2 angle() 14

3.4.3.3 cross() 15
3.4.3.4 dot() 15
3.4.3.5 equal() 16
3.4.3.6 getI() 16
3.4.3.7 getJ() 16
3.4.3.8 getK() 16
3.4.3.9 norm() 17
3.4.3.10 output() 17
3.4.3.11 setI() 17
3.4.3.12 setJ() 18
3.4.3.13 setK() 18
3.4.3.14 sub() 18

Index **21**

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

RayTracer 5
Shape3D 7
    Sphere 9
Vector 12

Chapter 2

Class Index

2.1 Class List

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

RayTracer 5
Shape3D 7
Sphere 9
Vector 12

Chapter 3

Class Documentation

3.1 RayTracer Class Reference

```
#include <RayTracer.hpp>
```

Public Member Functions

- RayTracer ()
- RayTracer (Vector light, Vector camera, Vector target, const Shape3D &shape)
- bool getScene (std::string filename)
- bool getScene ()
- std::vector< Vector > & getView ()
- std::vector< unsigned char > & getPixels ()

3.1.1 Detailed Description

A simple ray tracer in C++: currently only supports one object in scene

3.1.2 Constructor & Destructor Documentation

3.1.2.1 RayTracer() [1/2]

```
RayTracer::RayTracer ( )
```

Default constructor. should render a scene with default sphere, light source at (0,10,0), and camera at (5,0,0) with target at (0,0,0), and white background

3.1.2.2 RayTracer() [2/2]

```
RayTracer::RayTracer (
            Vector light,
            Vector camera,
            Vector target,
            const Shape3D & shape )
```

Use the parameters to set the scene components and render scene; use white background.

Parameters

light	- position (Vector w/r/t (0,0,0)) of light source
camera	- position (Vector w/r/t (0,0,0)) of camera source
target	- position (Vector w/r/t (0,0,0)) of target (where camera is looking)
shape	- the shape we wish to render in the scene

3.1.3 Member Function Documentation

3.1.3.1 getPixels()

```
std::vector<unsigned char>& RayTracer::getPixels ( )
```

Grading function: allow access to private data to compare pixel values with reference

Returns

a pointer to the vector holding the pixels

3.1.3.2 getScene() [1/2]

```
bool RayTracer::getScene ( )
```

Create png of rendered scene with name of file ‘scene.png’ Required function: don't delete this (I wrote it for you); may need to modify it

Returns

whether was scene was successfully written to disk as .png

3.1.3.3 getScene() [2/2]

```
bool RayTracer::getScene (
            std::string filename )
```

Create png of rendered scene with name of the file given by filename (use scene.png if filename is empty) Required function: don't delete this (I wrote it for you); may need to modify it

Returns

whether was scene was successfully written to disk as .png

3.1.3.4 getView()

```
std::vector<Vector>& RayTracer::getView ( )
```

Grading function: allow access to private data to compare view rays with reference

Returns

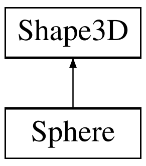
address of the vector holding the view rays

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

- RayTracer.hpp

3.2 Shape3D Class Reference

Inheritance diagram for Shape3D:



Public Member Functions

- virtual std::vector< unsigned char > color () const =0
- virtual Vector position () const =0
- virtual double ambient () const =0
- virtual Vector intersect (const Vector &s, const Vector &d) const =0
- virtual Vector normal (const Vector &pos) const =0

3.2.1 Member Function Documentation

3.2.1.1 ambient()

```
virtual double Shape3D::ambient ( ) const  [pure virtual]
```

The ambient of the shape

Returns

ambience of shape on the interval [0,1]

Implemented in Sphere.

3.2.1.2 color()

```
virtual std::vector<unsigned char> Shape3D::color ( ) const  [pure virtual]
```

The color of the shape

Returns

color shape: order is rgba with rgb in the interval [0,255] and a = 255;

Implemented in Sphere.

3.2.1.3 intersect()

```
virtual Vector Shape3D::intersect (
             const Vector & s,
             const Vector & d ) const  [pure virtual]
```

The position (as a Vector w/r/t Vector(0,0,0)) at which the unit vector d, originating from s, would or would not intersect with the shape

Returns

position at which user-supplied vector intersects with the shape (Vector(INFINITY, INFINITY, INFINITY) if no intersection)

Implemented in Sphere.

3.2.1.4 normal()

```
virtual Vector Shape3D::normal (
            const Vector & pos ) const  [pure virtual]
```

The (unit) normal vector of the shape at the user supplied position on the surface of the shape (as a Vector w/r/t Vector(0,0,0))

Returns

vector normal to the shape's surface at user-supplied position (Vector(INFINITY, INFINITY, INFINITY) if no normal available)

Implemented in Sphere.

3.2.1.5 position()

```
virtual Vector Shape3D::position ( ) const  [pure virtual]
```

The position of the shape

Returns

position of shape (center) w/r/t Vector(0,0,0)

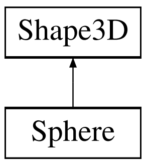
Implemented in Sphere.

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

- Shape3D.hpp

3.3 Sphere Class Reference

Inheritance diagram for Sphere:



Public Member Functions

- Sphere ()
- Sphere (double rad, Vector pos, std::vector< unsigned char > col, double amb)
- std::vector< unsigned char > color () const
- Vector position () const
- double ambient () const
- Vector intersect (const Vector &s, const Vector &d) const
- Vector normal (const Vector &pos) const
- double radius () const

3.3.1 Constructor & Destructor Documentation

3.3.1.1 Sphere() [1/2]

```
Sphere::Sphere ( )
```

default constructor: creates a purely red sphere with radius one at position (0,0,0) and with ambience of 0.2

Returns

sets data fields appropriately

3.3.1.2 Sphere() [2/2]

```
Sphere::Sphere (
            double rad,
            Vector pos,
            std::vector< unsigned char > col,
            double amb )
```

parameterized constructor: creates a sphere with user supplied color at given position with specified radius

Returns

sets data fields appropriately

3.3.2 Member Function Documentation

3.3.2.1 ambient()

```
double Sphere::ambient ( ) const  [virtual]
```

The ambient of the sphere

Returns

ambience of sphere on the interval [0,1]

Implements Shape3D.

3.3.2.2 color()

```
std::vector<unsigned char> Sphere::color ( ) const  [virtual]
```

The color of the sphere

Returns

color sphere: order is rgba with rgb in the interval [0,255] and a = 255;

Implements Shape3D.

3.3.2.3 intersect()

```
Vector Sphere::intersect (
            const Vector & s,
            const Vector & d ) const  [virtual]
```

calculates the intersection point, if one exists, between the sphere surface and the ray originating from position s with direction d (a unit vector)

Returns

position (as Vector w/r/t (0,0,0)) of where ray with origin s and unit direction d intersects with sphere, (inf,inf,inf) for no intersection

Implements Shape3D.

3.3.2.4 normal()

```
Vector Sphere::normal (
            const Vector & pos ) const  [virtual]
```

determines the unit normal vector on the surface of the sphere at the position given by the vector pos (w/r/t (0,0,0))

Returns

unit normal vector for the surface of the sphere at a user-specified position on the surface

Implements Shape3D.

3.3.2.5 position()

```
Vector Sphere::position ( ) const  [virtual]
```

The position of the sphere

Returns

position of sphere (center) w/r/t Vector(0,0,0)

Implements Shape3D.

3.3.2.6 radius()

```
double Sphere::radius ( ) const
```

This shape (but not all shapes) has a radius, so we add a new member function to allow the user to query it

Returns

the radius of the sphere

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

- Sphere.hpp

3.4 Vector Class Reference

```
#include <Vector.hpp>
```

Public Member Functions

- Vector ()
- Vector (double vx, double vy, double vz)
- double getI () const
- double getJ () const
- double getK () const
- void setI (double newVx)
- void setJ (double newVy)
- void setK (double newVz)
- bool equal (const Vector &rhs) const
- Vector add (const Vector &rhs) const
- Vector sub (const Vector &rhs) const
- Vector cross (const Vector &rhs) const
- double dot (const Vector &rhs) const
- double norm () const
- double angle (const Vector &rhs) const
- void output (std::ostream &out) const

3.4.1 Detailed Description

This is a basic C++ class to represent three-dimensional numbers. It's not meant to be difficult but as a refresher on classes.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Vector() [1/2]

```
Vector::Vector ( )
```

Default constructor. It should set the scalar components to 0.

3.4.2.2 Vector() [2/2]

```
Vector::Vector (
            double vx,
            double vy,
            double vz )
```

And a second one. Use the parameters to set the scalar components.

Parameters

vx	- the scalar value to use for i component
vy	- the scalar value to use for j component
vz	- the scalar value to use for k component

3.4.3 Member Function Documentation

3.4.3.1 add()

```
Vector Vector::add (
             const Vector & rhs ) const
```

Creates and returns a new Vector object representing the vector addition of two Vector objects

Returns

a new Vector object that contains the appropriate summed components

Parameters

rhs	- the Vector object to add to this object.

3.4.3.2 angle()

```
double Vector::angle (
             const Vector & rhs ) const
```

Returns the angle between two Vector objects in radians (over interval [0,2∗pi)).

Parameters

rhs	- the Vector object to find the angle be- tween with this object.

Returns

the angle (-1 if angle undefined)

3.4.3.3 cross()

```
Vector Vector::cross (
             const Vector & rhs ) const
```

Creates and returns a new Vector object that is cross product of this and the given Vector object.

Returns

a new Vector object that contains the cross product of this and the given Vector object.

Parameters

rhs	- the Vector object to cross with this object.

3.4.3.4 dot()

```
double Vector::dot (
             const Vector & rhs ) const
```

Returns the dot product of this and the given Vector object.

Returns

the dot product of this and the given Vector object.

Parameters

rhs	- the Vector object to dot with this object.

3.4.3.5 equal()

```
bool Vector::equal (
             const Vector & rhs ) const
```

Returns true if the scalar components for this object and rhs are the same, false otherwise.

Returns

true if scalar components in both objects are the same.

3.4.3.6 getI()

```
double Vector::getI ( ) const
```

Returns the scalar of the i component

Returns

vx.

3.4.3.7 getJ()

```
double Vector::getJ ( ) const
```

Returns the scalar of the j component

Returns

vy.

3.4.3.8 getK()

```
double Vector::getK ( ) const
```

Returns the scalar of the k component

Returns

vz.

3.4.3.9 norm()

```
double Vector::norm ( ) const
```

Returns the norm of the Vector object.

Returns

the norm (-1 if magnitude undefined)

3.4.3.10 output()

```
void Vector::output (
             std::ostream & out ) const
```

Outputs this Vector object on the given ostream. `‘vxi + vyj + vzk” (for debugging).

Parameters

out	- the ostream object to use to output.

3.4.3.11 setI()

```
void Vector::setI (
             double newVx )
```

Updates the scalar of the i component to the given newVx parameter.

Parameters

newVx	- the new value to use for the vx field.

3.4.3.12 setJ()

```
void Vector::setJ (
            double newVy )
```

Updates the scalar of the i component to the given newVx parameter.

Parameters

newVy	- the new value to use for the vx field.

3.4.3.13 setK()

```
void Vector::setK (
            double newVz )
```

Updates the scalar of the i component to the given newVx parameter.

Parameters

newVz	- the new value to use for the vx field.

3.4.3.14 sub()

```
Vector Vector::sub (
            const Vector & rhs ) const
```

Creates and returns a new Vector object representing the vector subtraction of two Vector objects

Returns

a new Vector object that contains the appropriate difference components

Parameters

rhs	- the Vector object to subtract from this object.

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

- Vector.hpp

Index

add
    Vector, 14
ambient
    Shape3D, 8
    Sphere, 10
angle
    Vector, 14

color
    Shape3D, 8
    Sphere, 10
cross
    Vector, 15

dot
    Vector, 15

equal
    Vector, 15

getI
    Vector, 16
getJ
    Vector, 16
getK
    Vector, 16
getPixels
    RayTracer, 6
getScene
    RayTracer, 6, 7
getView
    RayTracer, 7

intersect
    Shape3D, 8
    Sphere, 11

norm
    Vector, 16
normal
    Shape3D, 8
    Sphere, 11

output
    Vector, 17

position
    Shape3D, 9
    Sphere, 11

radius
    Sphere, 12
RayTracer, 5
    getPixels, 6
    getScene, 6, 7
    getView, 7
    RayTracer, 5

setI
    Vector, 17
setJ
    Vector, 17
setK
    Vector, 18
Shape3D, 7
    ambient, 8
    color, 8
    intersect, 8
    normal, 8
    position, 9
Sphere, 9
    ambient, 10
    color, 10
    intersect, 11
    normal, 11
    position, 11
    radius, 12
    Sphere, 10
sub
    Vector, 18

Vector, 12
    add, 14
    angle, 14
    cross, 15
    dot, 15
    equal, 15
    getI, 16
    getJ, 16
    getK, 16
    norm, 16
    output, 17
    setI, 17
    setJ, 17
    setK, 18
    sub, 18
    Vector, 13