Raytracing

Jan Görgens, Mihails Jersovs, Michael Baden

Version 1.0
Fri Sep 30 2022

Raytracing

Installation

Clone the repository and run the **main.cpp** file.

Change Settings

Before you run the **main.cpp** file, you can change the settings with editing the settings.json file.

Parameters:

width: width of the final image.

height: height of the final image.

camera_pos_x: x coordinate shift of the camera. This value will be added to the center of the width.

camera_pos_y: y coordinate shift of the camera. This value will be added to the center of the height.

background_color: color of the background in hex format.

image_folder: Optional path to a directory where the generated image should be saved at. Use double \\ to enter a path, for example "C:\\\\Documents".

light_sources: list of the light sources (only one light source supported yet).

pos_x: x coordinate of the light source.

pos_y: y coordinate of the light source.

pos_z: z coordinate of the light source.

intensity: intensity of the light source.

spheres: list of the light sources (only one light source supported yet)

pos_x: x coordinate of the sphere.

pos_y: y coordinate of the sphere.

pos_z: z coordinate of the sphere.

radius: radius of the sphere.

color: color of the sphere in hex format.

To generate the image shown above, use the following settings.json file:

```
  "width": 1600,
  "height": 800,
  "camera_pos_x": 0,
  "camera_pos_y": 0,
  "background_color": "#2874b2",
  "image_folder": "",
  "light_sources": [
    {
      "pos_x": 800,
      "pos_y": 300,
      "pos_z": 0,
      "intensity": 0.8
    }
  ],
  "spheres": [
    {
      "pos_x": 800,
      "pos_y": 450,
      "pos_z": 300,
      "radius": 200,
      "color": "#804000"
    },
    {
      "pos_x": 670,
      "pos_y": 250,
      "pos_z": 300,
      "radius": 100,
      "color": "#804000"
    },
    {
      "pos_x": 930,
      "pos_y": 250,
      "pos_z": 300,
      "radius": 100,
      "color": "#804000"
    },
    {
      "pos_x": 750,
      "pos_y": 350,
      "pos_z": 100,
      "radius": 30,
      "color": "#FFFFFF"
    },
    {
      "pos_x": 850,
      "pos_y": 350,
      "pos_z": 100,
      "radius": 30,
      "color": "#FFFFFF"
    },
    {
      "pos_x": 800,
      "pos_y": 410,
      "pos_z": 100,
      "radius": 30,
      "color": "#FE0202"
    },
    {
      "pos_x": 715,
      "pos_y": 470,
      "pos_z": 100,
      "radius": 30,
      "color": "#b1b0ae"
    },
    {
      "pos_x": 755,
      "pos_y": 500,
      "pos_z": 100,
      "radius": 30,
      "color": "#b1b0ae"
    },
    {
      "pos_x": 810,
      "pos_y": 500,
      "pos_z": 100,
      "radius": 30,
      "color": "#b1b0ae"
    },
    {
      "pos_x": 865,
      "pos_y": 500,
      "pos_z": 100,
      "radius": 30,
      "color": "#b1b0ae"
    },
    {
      "pos_x": 905,
      "pos_y": 470,
    "pos_z": 100,
    "radius": 30,
    "color": "#b1b0ae"
  },
  {
    "pos_x": 800,
    "pos_y": 800,
    "pos_z": 300,
    "radius": 300,
    "color": "#804000"
  },
  {
    "pos_x": 500,
    "pos_y": 650,
    "pos_z": 300,
    "radius": 40,
    "color": "#804000"
  },
  {
    "pos_x": 450,
    "pos_y": 590,
    "pos_z": 300,
    "radius": 40,
    "color": "#804000"
  },
  {
    "pos_x": 400,
    "pos_y": 530,
    "pos_z": 300,
    "radius": 40,
    "color": "#804000"
  },
  {
    "pos_x": 350,
    "pos_y": 590,
    "pos_z": 300,
    "radius": 40,
    "color": "#804000"
  },
  {
    "pos_x": 1100,
    "pos_y": 650,
    "pos_z": 300,
    "radius": 40,
    "color": "#804000"
  },
  {
    "pos_x": 1150,
    "pos_y": 590,
    "pos_z": 300,
    "radius": 40,
    "color": "#804000"
  },
  {
    "pos_x": 1200,
    "pos_y": 530,
    "pos_z": 300,
    "radius": 40,
    "color": "#804000"
  },
  {
    "pos_x": 1250,
    "pos_y": 470,
    "pos_z": 300,
    "radius": 40,
    "color": "#804000"
  },
  {
    "pos_x": 1500,
    "pos_y": 100,
    "pos_z": 200,
    "radius": 200,
    "color": "#fce903"
  },
  {
    "pos_x": 740,
    "pos_y": 340,
            "pos_z": 50,
            "radius": 10,
            "color": "#000000"
        },
        {
            "pos_x": 860,
            "pos_y": 360,
            "pos_z": 50,
            "radius": 10,
            "color": "#000000"
        }
    ]
}
```

Class Index

Class List

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

File Index

File List

Here is a list of all files with brief descriptions:

Class Documentation

Color Class Reference

`#include <Color.h>`

Public Member Functions

Color ()=default
Color (int **red**, int **green**, int **blue**)
Color (const char *hex)
Color (double c)
Color operator* (double d)
Color operator+ (**Color** other) const
Color operator- (**Color** other) const

Static Public Member Functions

static **Color red** ()
static **Color green** ()
static **Color blue** ()
static **Color white** ()
static **Color black** ()

Public Attributes

int **r**
int **g**
int **b**

Detailed Description

Class to represent a color in rgb format.

Constructor & Destructor Documentation

Color::Color () `[default]`

Color::Color (int *red*, int *green*, int *blue*) `[inline]`

Color::Color (const char * *hex*) `[inline]`

Color::Color (double *c*) `[inline]`

Constructor with only one double as input if each r,g,b value should be the same. When the input is 10, the according rgb color will be rgb(10,10,10)

Member Function Documentation

static Color Color::black () `[inline]`, `[static]`

Default color black

static Color Color::blue ()`[inline]`, `[static]`

Default color blue

static Color Color::green ()`[inline]`, `[static]`

Default color green

Color Color::operator* (double *d*)`[inline]`

Color Color::operator+ (Color *other*) const`[inline]`

Color Color::operator- (Color *other*) const`[inline]`

static Color Color::red ()`[inline]`, `[static]`

Default color red

static Color Color::white ()`[inline]`, `[static]`

Default color white

Member Data Documentation

int Color::b

int Color::g

int Color::r

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

Color.h

LightSource Class Reference

`#include <LightSource.h>`

Public Member Functions

LightSource (**Vec3** center, double intensity)
Vec3 get_center () const
double **get_intensity** ()

Detailed Description

This class represents a light source which can be added to a world

Constructor & Destructor Documentation

LightSource::LightSource (Vec3 *center*, double *intensity*) `[inline]`

Takes a **Vec3** and double as input. The **Vec3** represents the center of the light source. The double represents the intensity of the light source. The value of the intensity must be between 1 and 0 and will be limited if the value isn't in this interval.

Member Function Documentation

Vec3 LightSource::get_center () const `[inline]`

Returns the center of the light source

Returns

center of type **Vec3**

double LightSource::get_intensity () `[inline]`

Returns the intensity of the light source

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

LightSource.h

Ray Class Reference

`#include <ray.h>`

Public Member Functions

Ray (**Vec3** origin, **Vec3** shift)

*Constructor of the **Ray**. Accepts a **Vec3** origin and a **Vec3** shift. The shift vector will be added to the origin to make a shift of the image possible.*

Vec3 get_origin () const

This function returns the origin of the ray.

Vec3 get_direction () const

This function returns the direction of the ray.

Vec3 get_pos (double t)

Constructor & Destructor Documentation

Ray::Ray (Vec3 *origin*, Vec3 *shift*) `[inline]`

Constructor of the **Ray**. Accepts a **Vec3** origin and a **Vec3** shift. The shift vector will be added to the origin to make a shift of the image possible.

Member Function Documentation

Vec3 Ray::get_direction () const `[inline]`

This function returns the direction of the ray.

Returns

direction of type **Vec3**

Vec3 Ray::get_origin () const `[inline]`

This function returns the origin of the ray.

Returns

origin of type **Vec3**

Vec3 Ray::get_pos (double *t*) `[inline]`

Takes a double t as argument which is the multiplier of the direction. Calculates the position of the ray with adding the shift to the origin and with adding the direction multiplied with t. Returns the calculated **Vec3**.

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

ray.h

Sphere Class Reference

`#include <Sphere.h>`

Public Member Functions

Sphere ()=default
Sphere (**Vec3** center, double radius)
Sphere (**Vec3** center, double radius, **Color** c)
bool **intersect** (**Ray** ray, double &t) const
const **Vec3** **getCenter** () const
double **getRadius** () const
const **Color** & **getColor** () const

Detailed Description

This class represents a sphere object which can be added to a world

---

Constructor & Destructor Documentation

Sphere::Sphere () `[default]`

Sphere::Sphere (Vec3 *center*, double *radius*) `[inline]`

Takes a **Vec3** and double as input. The **Vec3** represents the center of the sphere. The double represents the radius of the sphere The default color is set to red.

Sphere::Sphere (Vec3 *center*, double *radius*, Color *c*) `[inline]`

Takes a **Vec3**, double and color as input. The **Vec3** represents the center of the sphere. The double represents the radius of the sphere. The color represents the color of the sphere.

Member Function Documentation

const Vec3 Sphere::getCenter () const `[inline]`

Returns the center of the sphere

const Color & Sphere::getColor () const `[inline]`

Returns the color of the sphere

double Sphere::getRadius () const `[inline]`

Returns the radius of the sphere

bool Sphere::intersect (Ray *ray*, double & *t*) const `[inline]`

Use equations from `https://collaborating.tuhh.de/cpf5546/oop-sose22/-/blob/master/project/doc/objects.md#primitive-objects`

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

Sphere.h

Vec3 Class Reference

`#include <Vec3.h>`

Public Member Functions

Vec3 ()=default
Vec3 (double x, double y, double z)
double **get_x** () const
double **get_y** () const
double **get_z** () const
virtual double **get_length** () const
virtual double **get_distance** (**Vec3** &other) const
virtual double **dot_product** () const
virtual **Vec3 normalize** ()
const **Vec3 operator-** (**Vec3** const &v) const
const **Vec3 operator+** (**Vec3** const &v) const
const **Vec3 operator/** (**Vec3** const &v) const
const **Vec3 operator/** (double n) const
const **Vec3 operator*** (**Vec3** const &v) const
const **Vec3 operator*** (double n) const
void **operator+=** (**Vec3** const &u)
void **operator-=** (**Vec3** const &u)

Constructor & Destructor Documentation

Vec3::Vec3 () `[default]`

Vec3::Vec3 (double *x*, double *y*, double *z*) `[inline]`

Member Function Documentation

double Vec3::dot_product () const `[virtual]`

Returns

dot product of the vector

double Vec3::get_distance (Vec3 & *other*) const `[virtual]`

Takes a second vector as input.

Returns

distance between both vectors

double Vec3::get_length () const `[virtual]`

Returns

the length of the vector

double Vec3::get_x () const

Returns the coordinates of the vector

double Vec3::get_y () const

double Vec3::get_z () const

Vec3 Vec3::normalize ()`[virtual]`

Returns

normalized vector

const Vec3 Vec3::operator* (double ***n*****) const**

const Vec3 Vec3::operator* (Vec3 const & ***v*****) const**

const Vec3 Vec3::operator+ (Vec3 const & ***v*****) const**

void Vec3::operator+= (Vec3 const & ***u*****)**`[inline]`

const Vec3 Vec3::operator- (Vec3 const & ***v*****) const**

void Vec3::operator-= (Vec3 const & ***u*****)**`[inline]`

const Vec3 Vec3::operator/ (double ***n*****) const**

const Vec3 Vec3::operator/ (Vec3 const & ***v*****) const**

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

Vec3.h
Vec3.cpp

World Class Reference

Public Member Functions

World (int width, int height, **Vec3** shift, **Color** background)
void **render_image** (std::string path)
void **add_sphere** (**Sphere** s)
void **add_light_source** (**LightSource** ls)
vector< **Sphere** > & **get_spheres** ()
LightSource & **get_light_source** ()
double **get_width** ()
double **get_height** ()
const **Color** & **get_background** ()

Detailed Description

The **World** class contains the core of the project like the main calculations and the image generation.

Constructor & Destructor Documentation

World::World (int *width*, int *height*, Vec3 *shift*, Color *background*) `[inline]`

Member Function Documentation

void World::add_light_source (LightSource *ls*) `[inline]`

Takes a light source as argument and adds it to the world

void World::add_sphere (Sphere *s*) `[inline]`

Takes a sphere as argument and adds it to the world

const Color & World::get_background () `[inline]`

Returns the background color

double World::get_height () `[inline]`

Returns the height

LightSource & World::get_light_source () `[inline]`

Returns the first light source of all light sources

vector< Sphere > & World::get_spheres () `[inline]`

Returns the vector of all spheres

double World::get_width () `[inline]`

Returns the width

void World::render_image (std::string *path*) `[inline]`

Main function to render the image with all the data

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

World.cpp

File Documentation

Color.h File Reference

```
#include <string>
#include <stdio.h>
```

Classes

class **Color**

Color.h

Go to the documentation of this file.

```
#include <string>
#include <stdio.h>

class Color {
public:
    int r;
    int g;
    int b;


    Color() = default;

    // Constructor to initialize the rgb values
    Color(int red, int green, int blue) : r(red), g(green), b(blue) {};

    // Constructor with a hex color as input which will be formatted into the rgb format.
    Color(const char *hex) {
        std::sscanf(hex, "#%02x%02x%02x", &r, &g, &b);
    }


    Color(double c) : r(c), g(c), b(c) {};


    Color operator*(double d) {
        return Color(r * d, g * d, b * d);
    }

    Color operator+(Color other) const {
        return Color(r + other.r, g + other.g, b + other.b);
    }

    Color operator-(Color other) const {
        return Color(r - other.r, g - other.g, b - other.b);
    }

    static Color red() { return Color(255, 0, 0); }
    static Color green() { return Color(0, 255, 0); }
    static Color blue() { return Color(0, 0, 255); }
    static Color white() { return Color(255, 255, 255); }
    static Color black() { return Color(0, 0, 0); }
};
```

LightSource.h File Reference

```
#include "../utils/include/Vec3.h"
```

Classes

class **LightSource**

LightSource.h

Go to the documentation of this file.

```
#ifndef RAYTRACING_LIGHTSOURCE_H
#define RAYTRACING_LIGHTSOURCE_H


#include "../utils/include/Vec3.h"

class LightSource {

private:
      Vec3 center;
      double intensity;

public:
      LightSource(Vec3 center, double intensity) : center(center),
intensity(intensity) {
            if(intensity > 1) intensity = 1;
            if(intensity < 0) intensity = 0;
      };

      Vec3 get_center() const {
            return center;
      }
      double get_intensity() {
            return intensity;
      }

};


#endif //RAYTRACING_LIGHTSOURCE_H
```

main.cpp File Reference

```
#include "objects/World.cpp"
#include <fstream>
#include <string>
#include "lib/json.hpp"
```

Typedefs

using **json** = nlohmann::json

Functions

int **main** ()

Main function to read all the data from the settings.json file and generate the according image.

Typedef Documentation

using json =   nlohmann::json

Function Documentation

int main ()

Main function to read all the data from the settings.json file and generate the according image.

test

ray.h File Reference

```
#include "include/Vec3.h"
```

Classes

class **Ray**

ray.h

Go to the documentation of this file.

```cpp
#include "include/Vec3.h"


class Ray {

private:
    Vec3 origin, direction, shift;

public:

    Ray(Vec3 origin, Vec3 shift) : origin(origin), shift(shift), direction(Vec3(0, 0, 1)) {};

    Vec3 get_origin() const {
        return origin;
    }

    Vec3 get_direction() const {
        return direction;
    }

    Vec3 get_pos(double t) {
        return get_origin() + get_direction() * t;
    }

};
```

readme.md File Reference

Sphere.h File Reference

```
#include "../utils/ray.h"
#include "../utils/Color.h"
#include <cmath>
```

Classes

class **Sphere**

Sphere.h

Go to the documentation of this file.

```cpp
#include "../utils/ray.h"
#include "../utils/Color.h"
#include <cmath>

class Sphere {


private:
     Vec3 center;
     double radius, radius2;
     Color color;


public:
     Sphere() = default;

     Sphere(Vec3 center, double radius) : center(center),
                                          radius(radius),
                                          radius2(radius * radius),
                                          color(Color::red()) {};

     Sphere(Vec3 center, double radius, Color c) : center(center),
                                                   radius(radius),
                                                   radius2(radius * radius),
                                                   color(c) {}


     bool intersect(Ray ray, double &t) const {
          Vec3 sc = center - ray.get_origin();
          double dp = dot_product(ray.get_direction(), sc); // dot product
          double delta = radius2 + pow(dp, 2) - pow(sc.get_length(), 2);
          if (delta < 0) return false;
          double t1, t2;
          t1 = dp + sqrt(delta);
          t2 = dp - sqrt(delta);
          t = t1 > t2 ? t2 : t1;
          return true;
     }


     const Vec3 getCenter() const {
          return center;
     }

     double getRadius() const {
          return radius;
     }

     const Color &getColor() const {
          return color;
     }

};

```

Vec3.cpp File Reference

```
#include <cmath>
#include "../include/Vec3.h"
```

Functions

double **dot_product** (**Vec3** const &u, **Vec3** const &v)

Function Documentation

double dot_product (Vec3 const & *u*, Vec3 const & *v*)

Returns

dot product of both input vectors

Vec3.h File Reference

Classes

class **Vec3**

Functions

double **dot_product** (**Vec3** const &u, **Vec3** const &v)

Function Documentation

double dot_product (Vec3 const & *u*, Vec3 const & *v*)

Returns

dot product of both input vectors

Vec3.h

Go to the documentation of this file.

```
#ifndef RAYTRACING_VEC3_H
#define RAYTRACING_VEC3_H


class Vec3 {
private:
     double x, y, z;

public:
     Vec3() = default;
     Vec3(double x, double y, double z) : x(x), y(y), z(z){};

     double get_x() const;

     double get_y() const;

     double get_z() const;

     virtual double get_length() const;

     virtual double get_distance(Vec3 &other) const;

     virtual double dot_product() const;

     virtual Vec3 normalize();

     const Vec3 operator-(Vec3 const &v) const;

     const Vec3 operator+(Vec3 const &v) const;

     const Vec3 operator/(Vec3 const &v) const;

     const Vec3 operator/(double n) const;

     const Vec3 operator*(Vec3 const &v) const;

     const Vec3 operator*(double n) const;

     void operator+=(Vec3 const &u) {
          x += u.x;
          y += u.y;
          z += u.z;
     }

     void operator-=(Vec3 const &u) {
          x -= u.x;
          y -= u.y;
          z -= u.z;
     }

};

double dot_product(Vec3 const &u, Vec3 const &v);

#endif //RAYTRACING_VEC3_H
```

World.cpp File Reference

```
#include <vector>
#include "Sphere.h"
#include "LightSource.h"
#include <string>
#include "../lib/CImg.h"
```

Classes

class **World**

Index

INDEX