Cuda Ray Tracer

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

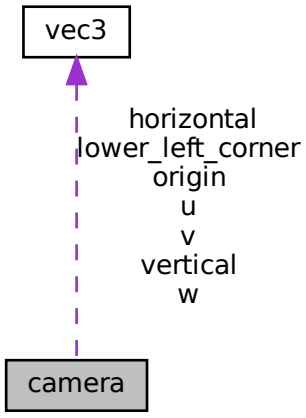
Chapter 4

Class Documentation

4.1 camera Class Reference

The class representing the camera (POV)

Collaboration diagram for camera:



Public Member Functions

- __device__ camera (vec3 lookfrom, vec3 lookat, vec3 vup, float vfov, float aspect, float aperture, float focus↩ _dist)

  The camera constructor.
- __device__ ray get_ray (float s, float t, curandState ∗state)

  Generates a ray from the camera.

Public Attributes

- vec3 origin
- vec3 lower_left_corner
- vec3 horizontal
- vec3 vertical
- vec3 v
- vec3 u
- vec3 w
- float lens_radius

4.1.1 Detailed Description

The class representing the camera (POV)

This class represents the camera with its point of view, with its origin, focus, and render plane, it also provides a ray generator.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 camera()

```
__device__ camera::camera (
            vec3 lookfrom,
            vec3 lookat,
            vec3 vup,
            float vfov,
            float aspect,
            float aperture,
            float focus_dist )  [inline]
```

The camera constructor.

Parameters

lookfrom	the origin point
lookat	where to look at
vup	the camera relative up direction
vfov	the vertical fov
aspect	the aspect ratio of the camera
aperture	the aperture of the lens of the camera
focus_dist	the distance of perfect focus from the camera

given the initialization parameters, it creates the needed vectors to then generate new rays from the camera

4.1.3 Member Function Documentation

4.1.3.1 get_ray()

```
__device__ ray camera::get_ray (
            float s,
            float t,
            curandState * state )  [inline]
```

Generates a ray from the camera.

Parameters

s	the uniform pointed x coordinate
t	the uniform pointed y coordinate
state	the random state

Returns

the ray pointing (x,y)

It generates a ray from the camera in the direction (x,y) with a slight shift given by the lens focus

4.1.4 Member Data Documentation

4.1.4.1 horizontal

```
vec3 camera::horizontal
```

4.1.4.2 lens_radius

```
float camera::lens_radius
```

4.1.4.3 lower_left_corner

```
vec3 camera::lower_left_corner
```

4.1.4.4 origin

```
vec3 camera::origin
```

4.1.4.5 u

```
vec3 camera::u
```

4.1.4.6 v

```
vec3 camera::v
```

4.1.4.7 vertical

```
vec3 camera::vertical
```

4.1.4.8 w

```
vec3 camera::w
```

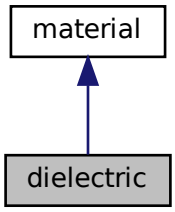
The documentation for this class was generated from the following file:

- includes/camera.cuh

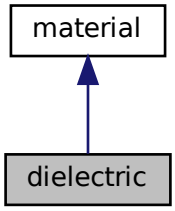
4.2 dielectric Class Reference

A dielectric material derived from material class.

Inheritance diagram for dielectric:



Collaboration diagram for dielectric:



Public Member Functions

- __device__ dielectric (float refraction_index)

  A dielectric constructor.
- virtual __device__ bool scatter (const ray &r_in, const t_hit_record &rec, vec3 &attenuation, ray &scattered, curandState *state) const override

  Computes the scatter of a given ray.

Public Attributes

- float ir

4.2.1 Detailed Description

A dielectric material derived from material class.

A dielectric material derived from material class

4.2.2 Constructor & Destructor Documentation

4.2.2.1 dielectric()

```
__device__ dielectric::dielectric (
            float refraction_index )  [inline]
```

A dielectric constructor.

Parameters

ir	the dielectric refraction index

4.2.3 Member Function Documentation

4.2.3.1 scatter()

```
virtual __device__ bool dielectric::scatter (
            const ray & r_in,
            const t_hit_record & rec,
            vec3 & attenuation,
            ray & scattered,
            curandState * state ) const  [inline], [override], [virtual]
```

Computes the scatter of a given ray.

Parameters

r_in	the input ray
rec	the hit point informations
attenuation	filled with material attenuation
scattered	the scattered ray
state	the random state

Returns

true if scattered else flase

Computes the scatter of a given ray

Implements material.

4.2.4 Member Data Documentation

4.2.4.1 ir

```
float dielectric::ir
```

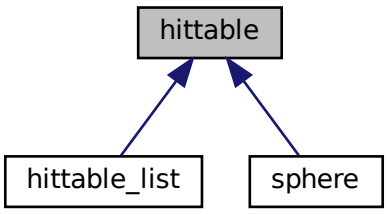
The documentation for this class was generated from the following file:

- includes/material.cuh

4.3 hittable Class Reference

this class represents an hittable object

Inheritance diagram for hittable:



Public Member Functions

- virtual __device__ bool hit (const ray &r, float t_min, float t_max, t_hit_record &rec) const =0

  A virtual function to know if hitted.

4.3.1 Detailed Description

this class represents an hittable object

This is an abstract class with only virtual methods which will get overrided by derived classes.

4.3.2 Member Function Documentation

4.3.2.1 hit()

```
virtual __device__ bool hittable::hit (
            const ray & r,
            float t_min,
            float t_max,
            t_hit_record & rec ) const  [pure virtual]
```

A virtual function to know if hitted.

Parameters

r	the ray to analyze
t_min	the minimum range span to hit
t_max	the maximum range span to hit
rec	the output structure filled with hit record data

Returns

the boolean value of true if hitten or false if not

This function is called by derived classes to know if a certain ray hitted the caller object, and fills out the t_hit_record struct with useful informations

Implemented in sphere, and hittable_list.

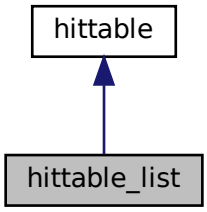
The documentation for this class was generated from the following file:

- includes/hittable.cuh

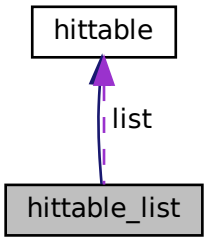
4.4 hittable_list Class Reference

A list of hittables.

Inheritance diagram for hittable_list:



Collaboration diagram for hittable_list:



Public Member Functions

- __device__ hittable_list ()

  Empty constructor.
- __device__ hittable_list (hittable $**$l, int n)

  Inizialized constructor.
- virtual __device__ bool hit (const ray &r, float t_min, float t_max, t_hit_record &rec) const override

  Check if some of the listed objects is hitted and returns the closest.

Public Attributes

- hittable $**$ list
- int size

4.4.1 Detailed Description

A list of hittables.

This class contains a list of hittables objects, it also provides an hit function to find the closest hitted object from the list

4.4.2 Constructor & Destructor Documentation

4.4.2.1 hittable_list() [1/2]

```
__device__ hittable_list::hittable_list ( )  [inline]
```

Empty constructor.

4.4.2.2 hittable_list() [2/2]

```
__device__ hittable_list::hittable_list (
            hittable ** l,
            int n )  [inline]
```

Inizialized constructor.

Parameters

l	a list of hittables
n	the size of the list

A constructor that is initialized with given values

4.4.3 Member Function Documentation

4.4.3.1 hit()

```
__device__ bool hittable_list::hit (
            const ray & r,
            float t_min,
            float t_max,
            t_hit_record & rec ) const  [override], [virtual]
```

Check if some of the listed objects is hitted and returns the closest.

Parameters

r	the ray to analyze
t_min	the minimum range span to hit
t_max	the maximum range span to hit
rec	the output structure filled with hit record data

Returns

the boolean value of true if hitten or false if not

it searches the closest hittable object from the given ray

Implements hittable.

4.4.4 Member Data Documentation

4.4.4.1 list

```
hittable** hittable_list::list
```

4.4.4.2 size

```
int hittable_list::size
```

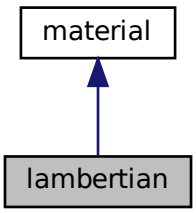
The documentation for this class was generated from the following file:

- includes/hittable_list.cuh

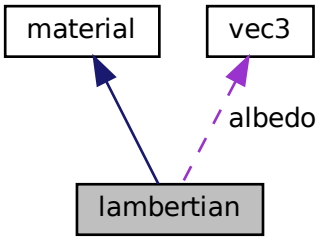
4.5 lambertian Class Reference

A lambertian material derived from material class.

Inheritance diagram for lambertian:



Collaboration diagram for lambertian:



Public Member Functions

- __device__ lambertian (const vec3 &a)

  A lambertian constructor.
- virtual __device__ bool scatter (const ray &r_in, const t_hit_record &rec, vec3 &attenuation, ray &scattered, curandState *state) const override

  Computes the scatter of a given ray.

Public Attributes

- vec3 albedo

4.5.1 Detailed Description

A lambertian material derived from material class.

A lambertian material derived from material class

4.5.2 Constructor & Destructor Documentation

4.5.2.1 lambertian()

```
__device__ lambertian::lambertian (
            const vec3 & a )  [inline]
```

A lambertian constructor.

Parameters

a	the lambertian albedo

4.5.3 Member Function Documentation

4.5.3.1 scatter()

```
virtual __device__ bool lambertian::scatter (
            const ray & r_in,
            const t_hit_record & rec,
            vec3 & attenuation,
            ray & scattered,
            curandState * state ) const  [inline], [override], [virtual]
```

Computes the scatter of a given ray.

Parameters

r_in	the input ray
rec	the hit point informations
attenuation	filled with material attenuation
scattered	the scattered ray
state	the random state

Returns

true if scattered else flase

Computes the scatter of a given ray

Implements material.

4.5.4 Member Data Documentation

4.5.4.1 albedo

```
vec3 lambertian::albedo
```

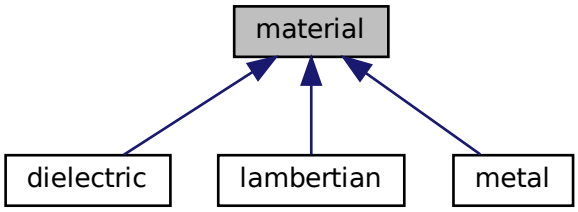
The documentation for this class was generated from the following file:

- includes/material.cuh

4.6 material Class Reference

The abstract class of materials.

Inheritance diagram for material:



Public Member Functions

- virtual __device__ bool scatter (const ray &r_in, const t_hit_record &rec, vec3 &attenuation, ray &scattered, curandState *state) const =0

  Computes the scatter of a given ray.

4.6.1 Detailed Description

The abstract class of materials.

Represents the materials with their ray scattering virtual function.

4.6.2 Member Function Documentation

4.6.2.1 scatter()

```
virtual __device__ bool material::scatter (
            const ray & r_in,
            const t_hit_record & rec,
            vec3 & attenuation,
            ray & scattered,
            curandState * state ) const  [pure virtual]
```

Computes the scatter of a given ray.

Parameters

r_in	the input ray
rec	the hit point informations
attenuation	filled with material attenuation
scattered	the scattered ray
state	the random state

Returns

true if scattered else flase

Computes the scatter of a given ray

Implemented in dielectric, metal, and lambertian.

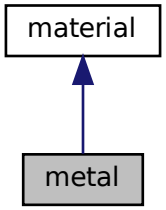
The documentation for this class was generated from the following file:

- includes/material.cuh

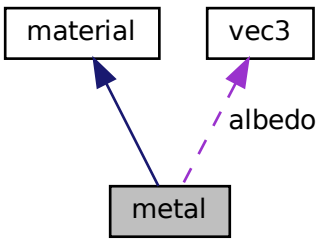
4.7 metal Class Reference

A metal material derived from material class.

Inheritance diagram for metal:



Collaboration diagram for metal:



Public Member Functions

- __device__ metal (const vec3 &a, float f)

  A metal constructor.
- virtual __device__ bool scatter (const ray &r_in, const t_hit_record &rec, vec3 &attenuation, ray &scattered, curandState *state) const override

  Computes the scatter of a given ray.

Public Attributes

- vec3 albedo
- float fuzz

4.7.1 Detailed Description

A metal material derived from material class.

A metal material derived from material class

4.7.2 Constructor & Destructor Documentation

4.7.2.1 metal()

```
__device__ metal::metal (
            const vec3 & a,
            float f )  [inline]
```

A metal constructor.

Parameters

a	the metal albedo
f	the fuzz uniform value

4.7.3 Member Function Documentation

4.7.3.1 scatter()

```
virtual __device__ bool metal::scatter (
            const ray & r_in,
            const t_hit_record & rec,
```

```
            vec3 & attenuation,
            ray & scattered,
            curandState * state ) const  [inline], [override], [virtual]
```

Computes the scatter of a given ray.

Parameters

r_in	the input ray
rec	the hit point informations
attenuation	filled with material attenuation
scattered	the scattered ray
state	the random state

Returns

true if scattered else flase

Computes the scatter of a given ray

Implements material.

4.7.4 Member Data Documentation

4.7.4.1 albedo

```
vec3 metal::albedo
```

4.7.4.2 fuzz

```
float metal::fuzz
```

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

- includes/material.cuh

4.8 ray Class Reference

Class representing rays.

Collaboration diagram for ray:

Public Member Functions

- __device__ ray ()

  Empty Costructor.
- __device__ ray (const vec3 &origin, const vec3 &direction)

  Standard Costructor.
- __device__ vec3 origin () const

  A getter for origin.
- __device__ vec3 direction () const

  A getter for direction.
- __device__ vec3 at (const double t) const

  A scalar product of the ray.

Public Attributes

- vec3 orig
- vec3 dir

4.8.1 Detailed Description

Class representing rays.

A class that represents rays as a tuple of origin and direction as vec3 and some utilities functions

4.8.2 Constructor & Destructor Documentation

4.8.2.1 ray() [1/2]

```
__device__ ray::ray ( )  [inline]
```

Empty Costructor.

4.8.2.2 ray() [2/2]

```
__device__ ray::ray (
            const vec3 & origin,
            const vec3 & direction )  [inline]
```

Standard Costructor.

Parameters

origin	the origin point of the ray
direction	the direction of the ray

4.8.3 Member Function Documentation

4.8.3.1 at()

```
__device__ vec3 ray::at (
            const double t ) const  [inline]
```

A scalar product of the ray.

Returns

the product of the scalar and the ray

4.8.3.2 direction()

```
__device__ vec3 ray::direction ( ) const  [inline]
```

A getter for direction.

Returns

the direction of the ray

4.8.3.3 origin()

```
__device__ vec3 ray::origin ( ) const  [inline]
```

A getter for origin.

Returns

the origin of the ray

4.8.4 Member Data Documentation

4.8.4.1 dir

```
vec3 ray::dir
```

4.8.4.2 orig

```
vec3 ray::orig
```

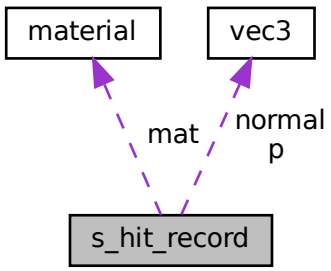
The documentation for this class was generated from the following file:

- includes/ray.cuh

4.9 s_hit_record Struct Reference

this struct contains hit record informations

Collaboration diagram for s_hit_record:



Public Attributes

- vec3 p
- vec3 normal
- float t
- material * mat

4.9.1 Detailed Description

this struct contains hit record informations

This struct is made of useful informations about hits like the hitted material, the hit distance, the point of impact and the normal of the hitten point.

4.9.2 Member Data Documentation

4.9.2.1 mat

```
material* s_hit_record::mat
```

4.9.2.2 normal

```
vec3 s_hit_record::normal
```

4.9.2.3 p

```
vec3 s_hit_record::p
```

4.9.2.4 t

```
float s_hit_record::t
```

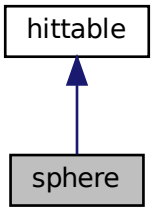
The documentation for this struct was generated from the following file:

- includes/hittable.cuh

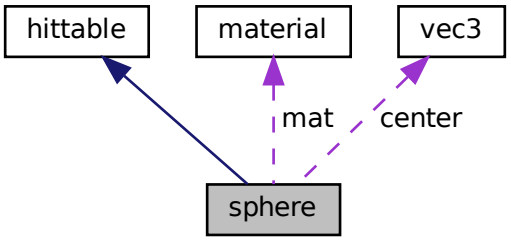
4.10 sphere Class Reference

A sphere object derived by hittable.

Inheritance diagram for sphere:



Collaboration diagram for sphere:



Public Member Functions

- __device__ sphere ()
  Empty Constructor.
- __device__ sphere (vec3 c, float r, material *m)
  Standard Constructor.
- virtual __device__ bool hit (const ray &r, float t_min, float t_max, t_hit_record &rec) const override
  A function to check if hitted.

Public Attributes

- vec3 center
- float radius
- material * mat

4.10.1 Detailed Description

A sphere object derived by hittable.

This class represents a sphere with its core functions

4.10.2 Constructor & Destructor Documentation

4.10.2.1 sphere() [1/2]

```
__device__ sphere::sphere ( )  [inline]
```

Empty Constructor.

4.10.2.2 sphere() [2/2]

```
__device__ sphere::sphere (
            vec3 c,
            float r,
            material * m )  [inline]
```

Standard Constructor.

Parameters

c	the center of the sphere
r	the radius of the sphere
m	the material of the sphere

4.10.3 Member Function Documentation

4.10.3.1 hit()

```
__device__ bool sphere::hit (
            const ray & r,
            float t_min,
            float t_max,
            t_hit_record & rec ) const  [override], [virtual]
```

A function to check if hitted.

Parameters

r	the ray to analyze
t_min	the minimum range span to hit
t_max	the maximum range span to hit
rec	the output structure filled with hit record data

Returns

the boolean value of true if hitten or false if not

This function is called to check if a certain ray is hitting the current sphere

Implements hittable.

4.10.4 Member Data Documentation

4.10.4.1 center

```
vec3 sphere::center
```

4.10.4.2 mat

```
material* sphere::mat
```

4.10.4.3 radius

```
float sphere::radius
```

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

- includes/sphere.cuh

4.11 vec3 Class Reference

Public Member Functions

- __host__ __device__ vec3 ()
  Empty Constructor.
- __host__ __device__ vec3 (float e0, float e1, float e2)
  Standard Constructor.
- __host__ __device__ float x () const
  A getter for x.
- __host__ __device__ float y () const
  A getter for y.
- __host__ __device__ float z () const
  A getter for z.
- __host__ __device__ float r () const
  A getter for x.
- __host__ __device__ float g () const
  A getter for y.
- __host__ __device__ float b () const
  A getter for z.
- __host__ __device__ vec3 operator- () const
  A vec3 inverter.
- __host__ __device__ float operator[ ] (int i) const
  An indexed getter.
- __host__ __device__ float & operator[ ] (int i)
  An indexed getter.
- __host__ __device__ vec3 & operator+= (const vec3 &v)
  The sum operator.
- __host__ __device__ vec3 & operator*= (const vec3 &v)
  The multiplication operator.
- __host__ __device__ vec3 & operator*= (const float t)
  The multiplication operator.
- __host__ __device__ vec3 & operator/= (const float t)
  The division operator.
- __host__ __device__ float length () const
  Computes the absolute length of the vec3.
- __host__ __device__ float length_squared () const
  Computes the squared length of the vec3.

Public Attributes

- float e [3]

4.11.1 Constructor & Destructor Documentation

4.11.1.1 vec3() [1/2]

```
__host__ __device__ vec3::vec3 ( )  [inline]
```

Empty Constructor.

it initializes all the three values at 0

4.11.1.2 vec3() [2/2]

```
__host__ __device__ vec3::vec3 (
            float e0,
            float e1,
            float e2 )  [inline]
```

Standard Constructor.

Parameters

e0	the x of the vec3
e1	the y of the vec3
e2	the z of the vec3

4.11.2 Member Function Documentation

4.11.2.1 b()

```
__host__ __device__ float vec3::b ( ) const  [inline]
```

A getter for z.

Returns

the z of the vec3

4.11.2.2 g()

```
__host__ __device__ float vec3::g ( ) const  [inline]
```

A getter for y.

Returns

the y of the vec3

4.11.2.3 length()

```
__host__ __device__ float vec3::length ( ) const  [inline]
```

Computes the absolute length of the vec3.

Returns

the absolute length of the vec3

4.11.2.4 length_squared()

```
__host__ __device__ float vec3::length_squared ( ) const  [inline]
```

Computes the squared length of the vec3.

Returns

the squared length of the vec3

4.11.2.5 operator*=() [1/2]

```
__host__ __device__ vec3& vec3::operator*= (
            const float t )  [inline]
```

The multiplication operator.

Parameters

t	the scalar to multiply with

Returns

the multiplied result

4.11.2.6 operator*=() [2/2]

```
__host__ __device__ vec3& vec3::operator*= (
            const vec3 & v )  [inline]
```

The multiplication operator.

Parameters

v	the vec3 to multiply with

Returns

the multiplied result

4.11.2.7 operator+=()

```
__host__ __device__ vec3& vec3::operator+= (
            const vec3 & v )  [inline]
```

The sum operator.

Parameters

v	the vec3 to sum with

Returns

the summed result

4.11.2.8 operator-()

```
__host__ __device__ vec3 vec3::operator- ( ) const  [inline]
```

A vec3 inverter.

Returns

the inverted vec3

4.11.2.9 operator/=()

```
__host__ __device__ vec3& vec3::operator/= (
            const float t )  [inline]
```

The division operator.

Parameters

t	the scalar to divide with

Returns

the divided result

4.11.2.10 operator[]() [1/2]

```
__host__ __device__ float& vec3::operator[] (
            int i )  [inline]
```

An indexed getter.

Parameters

i	the index of the value to get

Returns

the value indexed at i

4.11.2.11 operator[]() [2/2]

```
__host__ __device__ float vec3::operator[] (
            int i ) const  [inline]
```

An indexed getter.

Parameters

i	the index of the value to get

Returns

the value indexed at i

4.11.2.12 r()

```
__host__ __device__ float vec3::r ( ) const  [inline]
```

A getter for x.

Returns

the x of the vec3

4.11.2.13 x()

```
__host__ __device__ float vec3::x ( ) const  [inline]
```

A getter for x.

Returns

the x of the vec3

4.11.2.14 y()

```
__host__ __device__ float vec3::y ( ) const  [inline]
```

A getter for y.

Returns

the y of the vec3

4.11.2.15 z()

```
__host__ __device__ float vec3::z ( ) const  [inline]
```

A getter for z.

Returns

the z of the vec3

4.11.3 Member Data Documentation

4.11.3.1 e

```
float vec3::e[3]
```

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

- includes/vec3.cuh

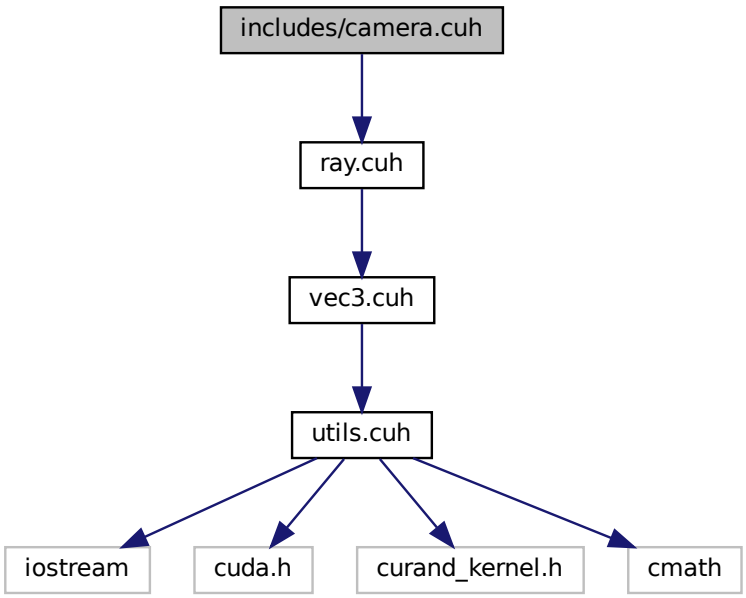
Chapter 5

File Documentation

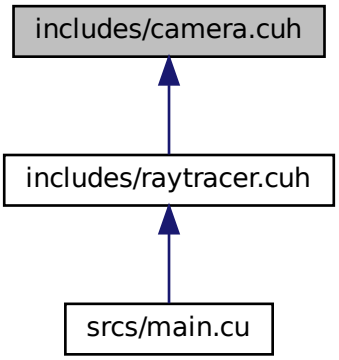
5.1 includes/camera.cuh File Reference

```
#include "ray.cuh"
```

Include dependency graph for camera.cuh:



This graph shows which files directly or indirectly include this file:



Classes

- class camera

  The class representing the camera (POV)

Macros

- #define CAMERA_CUH
- #define ASPECT_RATIO float(W) / float(H)
- #define VIEW_H 2.0f
- #define VIEW_W ASPECT_RATIO ∗ VIEW_H
- #define FOCAL_LEN 1.0f
- #define ORIGIN vec3(0, 0, 0)
- #define HORIZONTAL vec3(VIEW_W, 0, 0)
- #define VERTICAL vec3(0, VIEW_H, 0)
- #define LOWER_LEFT_CORNER vec3(-VIEW_W/2, -VIEW_H/2, -FOCAL_LEN)

Functions

- __device__ vec3 unit_disk_rand (curandState ∗s)

  generates a random uniform vec3
- __device__ ray O_get_ray (camera ∗c, float s, float t, curandState ∗state)

  An optimized version of camera->get_ray.

5.1.1 Macro Definition Documentation

5.1.1.1 ASPECT_RATIO

```
#define ASPECT_RATIO float(W) / float(H)
```

5.1.1.2 CAMERA_CUH

```
#define CAMERA_CUH
```

5.1.1.3 FOCAL_LEN

```
#define FOCAL_LEN 1.0f
```

5.1.1.4 HORIZONTAL

```
#define HORIZONTAL vec3(VIEW_W, 0, 0)
```

5.1.1.5 LOWER_LEFT_CORNER

```
#define LOWER_LEFT_CORNER vec3(-VIEW_W/2, -VIEW_H/2, -FOCAL_LEN)
```

5.1.1.6 ORIGIN

```
#define ORIGIN vec3(0, 0, 0)
```

5.1.1.7 VERTICAL

```
#define VERTICAL vec3(0, VIEW_H, 0)
```

5.1.1.8 VIEW_H

```
#define VIEW_H 2.0f
```

5.1.1.9 VIEW_W

```
#define VIEW_W ASPECT_RATIO * VIEW_H
```

5.1.2 Function Documentation

5.1.2.1 O_get_ray()

```
__device__ ray O_get_ray (
            camera * c,
            float s,
            float t,
            curandState * state )
```

An optimized version of camera->get_ray.

Parameters

c	the camera
s	the uniform pointed x coordinate
t	the uniform pointed y coordinate
state	the random state

Returns

the ray pointing (x,y)

An optimized version of camera->get_ray used to skip some operations of fetching from the class vtable at runtime to gain performance

5.1.2.2 unit_disk_rand()

```
__device__ vec3 unit_disk_rand (
            curandState * s )
```

generates a random uniform vec3

Parameters

s	the pointer of the curandState

Returns

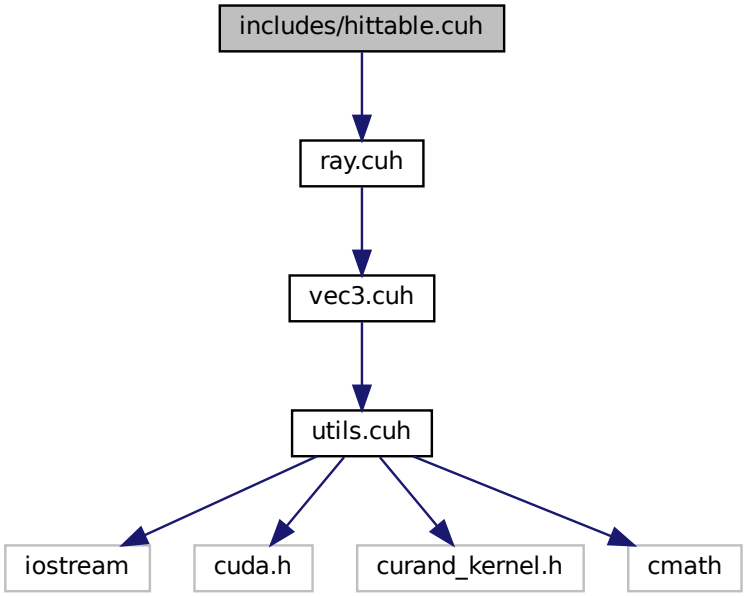
a random uniform vec3

Given the current cuda random state, it generates a uniform vec3

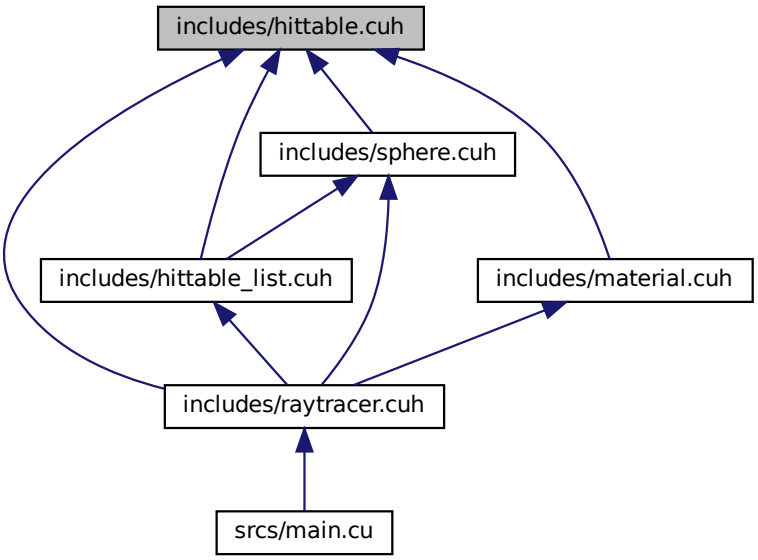
5.2 includes/hittable.cuh File Reference

```
#include "ray.cuh"
```

Include dependency graph for hittable.cuh:



This graph shows which files directly or indirectly include this file:



Classes

- struct s_hit_record

  this struct contains hit record informations
- class hittable

  this class represents an hittable object

Macros

- #define HITTABLE_CUH

Typedefs

- typedef struct s_hit_record t_hit_record

  this struct contains hit record informations

5.2.1 Macro Definition Documentation

5.2.1.1 HITTABLE_CUH

```
#define HITTABLE_CUH
```

5.2.2 Typedef Documentation

5.2.2.1 t_hit_record

```
typedef struct s_hit_record t_hit_record
```

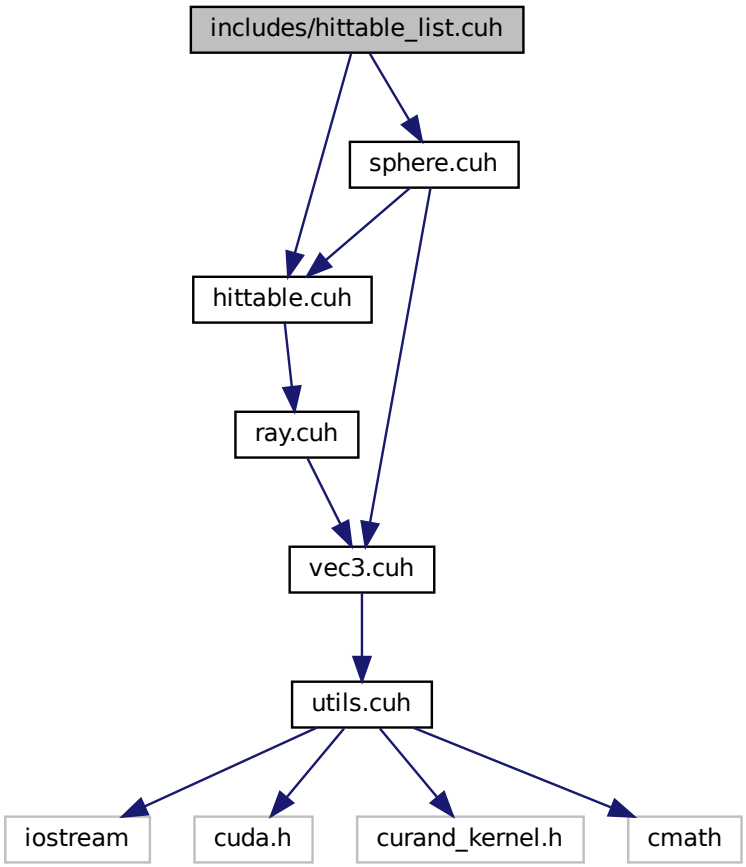
this struct contains hit record informations

This struct is made of useful informations about hits like the hitted material, the hit distance, the point of impact and the normal of the hitten point.

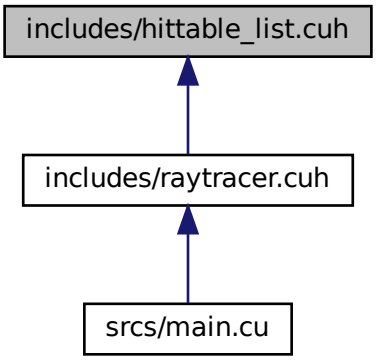
5.3 includes/hittable_list.cuh File Reference

```
#include "hittable.cuh"
#include "sphere.cuh"
```

Include dependency graph for hittable_list.cuh:



This graph shows which files directly or indirectly include this file:



Classes

- class hittable_list

  A list of hittables.

Macros

- #define HITTABLE_LIST_CUH

Functions

- __device__ bool O_hit (hittable_list *h, ray &r, float t_min, float t_max, t_hit_record &rec)

  An optimized version of hittable_list->hit.

5.3.1 Macro Definition Documentation

5.3.1.1 HITTABLE_LIST_CUH

```
#define HITTABLE_LIST_CUH
```

5.3.2 Function Documentation

5.3.2.1 O_hit()

```
__device__ bool O_hit (
             hittable_list * h,
             ray & r,
             float t_min,
             float t_max,
             t_hit_record & rec )
```

An optimized version of hittable_list->hit.

Parameters

r	the ray to analyze
t_min	the minimum range span to hit
t_max	the maximum range span to hit
rec	the output structure filled with hit record data

Returns

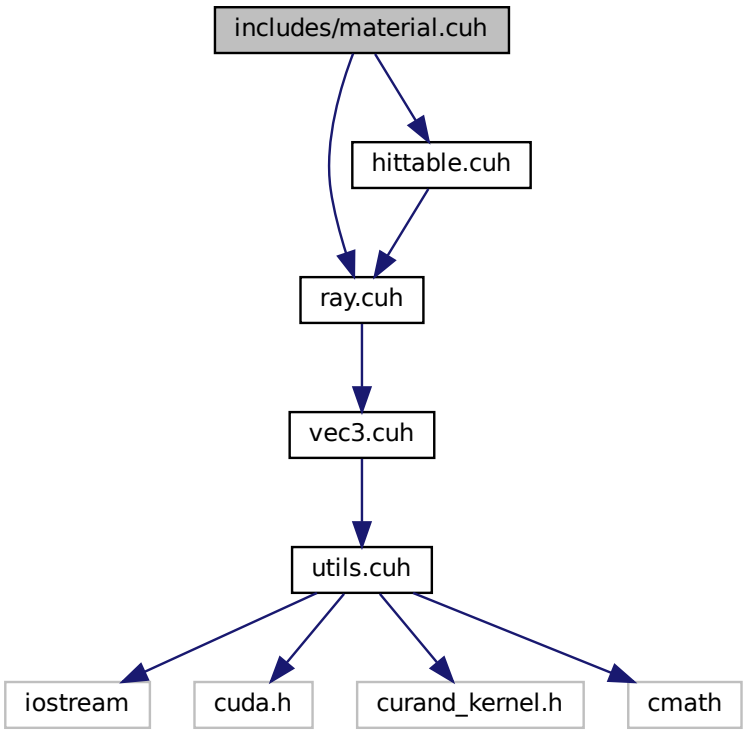
the boolean value of true if hitten or false if not

An optimized version of hittable_list->hit which skips a lot of vtable fetches and som function calls to gain performances at runtime

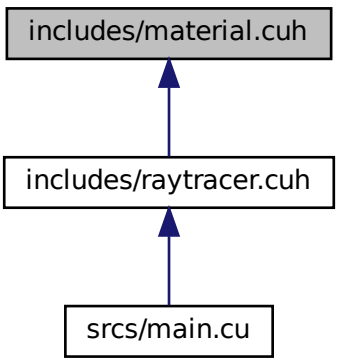
5.4 includes/material.cuh File Reference

```
#include "ray.cuh"
#include "hittable.cuh"
```

Include dependency graph for material.cuh:



This graph shows which files directly or indirectly include this file:



Classes

- class material

The abstract class of materials.

- class lambertian

  A lambertian material derived from material class.

- class metal

  A metal material derived from material class.

- class dielectric

  A dielectric material derived from material class.

Macros

- #define MATERIAL_CUH

Functions

- __device__ vec3 unit_sphere_rand (curandState *s)

  generates a random uniform vec3

- __device__ vec3 reflect (const vec3 &v, const vec3 &u)

  Computes a reflection.

- __device__ float schlick (float cos, float ir)

  Computes an approssimated refraction.

- __device__ bool refract (const vec3 &v, const vec3 &u, float ni_over_nt, vec3 &refracted)

  Computes a refraction.

5.4.1 Macro Definition Documentation

5.4.1.1 MATERIAL_CUH

```
#define MATERIAL_CUH
```

5.4.2 Function Documentation

5.4.2.1 reflect()

```
__device__ vec3 reflect (
            const vec3 & v,
            const vec3 & u )
```

Computes a reflection.

Parameters

v	the direction of the ray
u	the normal of the ray

Returns

the reflected ray

Computes the reflection of a given ray

5.4.2.2 refract()

```
__device__ bool refract (
            const vec3 & v,
            const vec3 & u,
            float ni_over_nt,
            vec3 & refracted )
```

Computes a refraction.

Parameters

v	direction unit vector
u	outward unit vector
ni_over_nt	normals refraction indices
refracted	the output refraction

Returns

a bool: true if the ray is refracted else false

Computes a refraction by its parameters and fills the refracted parameter with the output refraction given by the Snell's law.

5.4.2.3 schlick()

```
__device__ float schlick (
            float cos,
            float ir )
```

Computes an approssimated refraction.

Parameters

cos	cosine of the refraction
ir	the refraction index

Returns

the refraction probability

An algorithm of approximation of glass refraction by Christophe Schlick

5.4.2.4 unit_sphere_rand()

```
__device__ vec3 unit_sphere_rand (
             curandState * s )
```

generates a random uniform vec3

Parameters

s	the pointer of the curandState

Returns

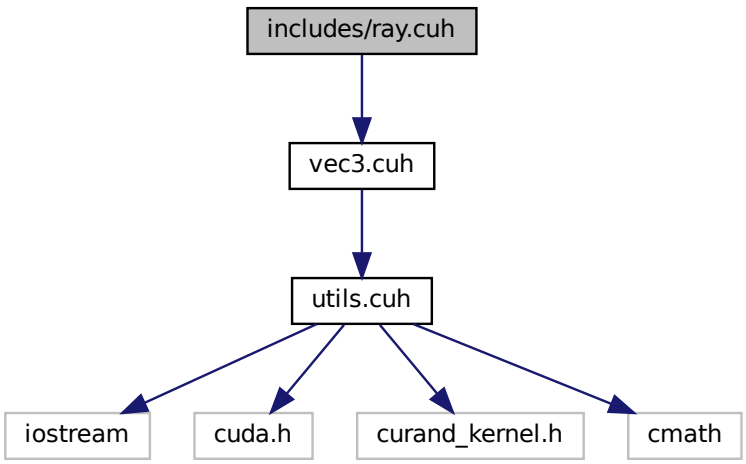
a random uniform vec3

Given the current cuda random state, it generates a uniform vec3

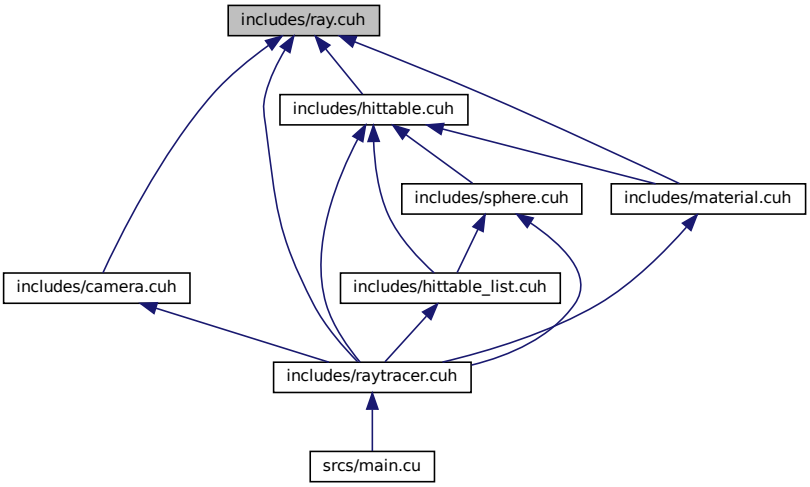
5.5 includes/ray.cuh File Reference

```
#include "vec3.cuh"
```

Include dependency graph for ray.cuh:



This graph shows which files directly or indirectly include this file:



Classes

- class ray

  Class representing rays.

Macros

- #define RAY_CUH

5.5.1 Macro Definition Documentation

5.5.1.1 RAY_CUH

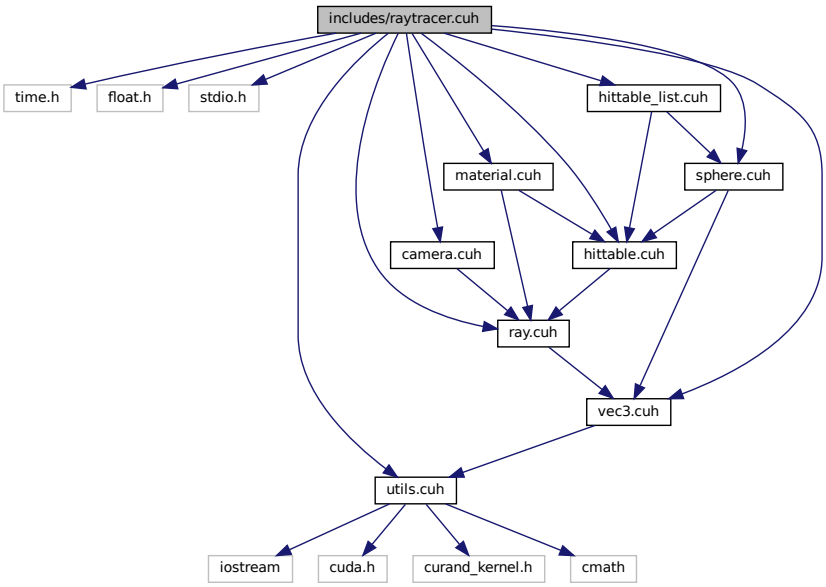
```
#define RAY_CUH
```

5.6 includes/raytracer.cuh File Reference

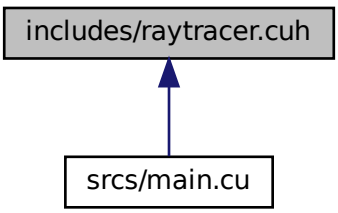
```
#include <time.h>
#include <float.h>
#include <stdio.h>
#include "utils.cuh"
#include "vec3.cuh"
#include "ray.cuh"
#include "hittable.cuh"
```

```
#include "hittable_list.cuh"
#include "sphere.cuh"
#include "camera.cuh"
#include "material.cuh"
```

Include dependency graph for raytracer.cuh:



This graph shows which files directly or indirectly include this file:



Macros

- #define RAYTRACER_CUH
- #define W 1200
- #define H 800
- #define PIXELS W ∗ H
- #define SAMPLES 32
- #define BSIZE PIXELS ∗ sizeof(vec3)
- #define BLOCK_W 8

- #define BLOCK_H 8
- #define SEED 42
- #define REFRACTION 100
- #define SHARED 1
- #define WEIGHT 0.5f
- #define MANEGED 1

5.6.1 Macro Definition Documentation

5.6.1.1 BLOCK_H

```
#define BLOCK_H 8
```

5.6.1.2 BLOCK_W

```
#define BLOCK_W 8
```

5.6.1.3 BSIZE

```
#define BSIZE PIXELS * sizeof(vec3)
```

5.6.1.4 H

```
#define H 800
```

5.6.1.5 MANEGED

```
#define MANEGED 1
```

5.6.1.6 PIXELS

```
#define PIXELS W * H
```

5.6.1.7 RAYTRACER_CUH

```
#define RAYTRACER_CUH
```

5.6.1.8 REFRACTION

```
#define REFRACTION 100
```

5.6.1.9 SAMPLES

```
#define SAMPLES 32
```

5.6.1.10 SEED

```
#define SEED 42
```

5.6.1.11 SHARED

```
#define SHARED 1
```

5.6.1.12 W

```
#define W 1200
```

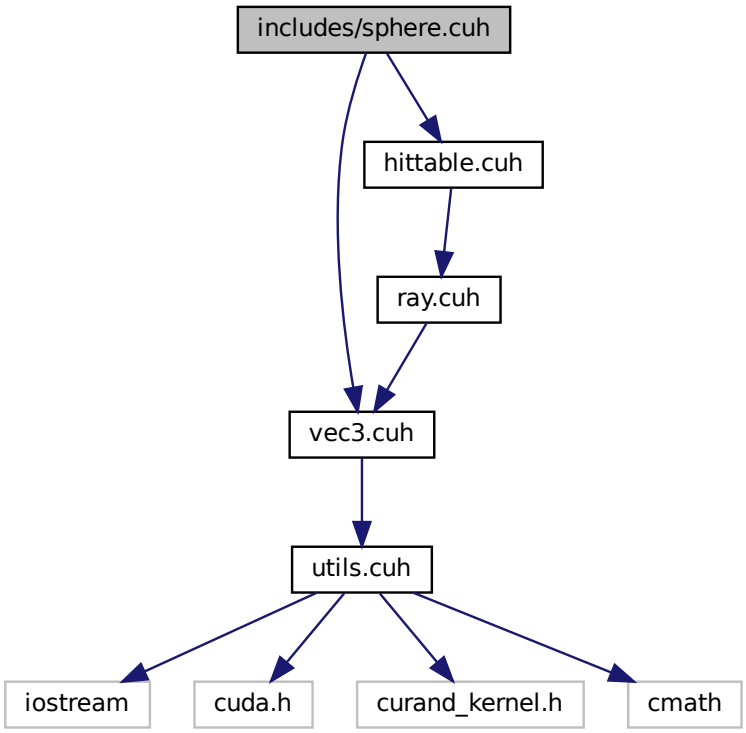
5.6.1.13 WEIGHT

```
#define WEIGHT 0.5f
```

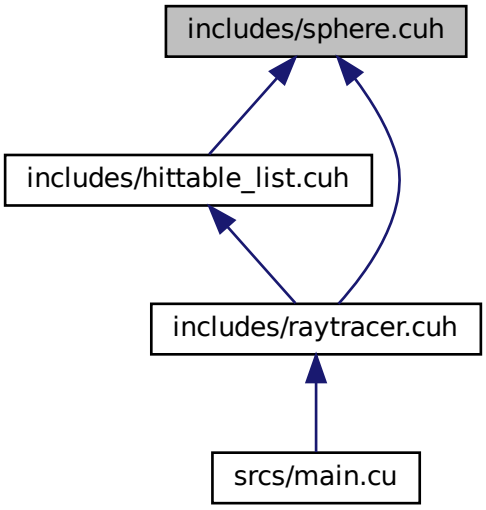
5.7 includes/sphere.cuh File Reference

```
#include "vec3.cuh"
#include "hittable.cuh"
```

Include dependency graph for sphere.cuh:



This graph shows which files directly or indirectly include this file:



Classes

- class sphere

  A sphere object derived by hittable.

Macros

- #define SPHERE_CUH

Functions

- __device__ bool O_hit (sphere *s, const ray &r, float t_min, float t_max, t_hit_record &rec)

  An optimized version of sphere->hit.

5.7.1 Macro Definition Documentation

5.7.1.1 SPHERE_CUH

```
#define SPHERE_CUH
```

5.7.2 Function Documentation

5.7.2.1 O_hit()

```
__device__ bool O_hit (
            sphere * s,
            const ray & r,
            float t_min,
            float t_max,
            t_hit_record & rec )
```

An optimized version of sphere->hit.

Parameters

r	the ray to analyze
t_min	the minimum range span to hit
t_max	the maximum range span to hit
rec	the output structure filled with hit record data

Returns

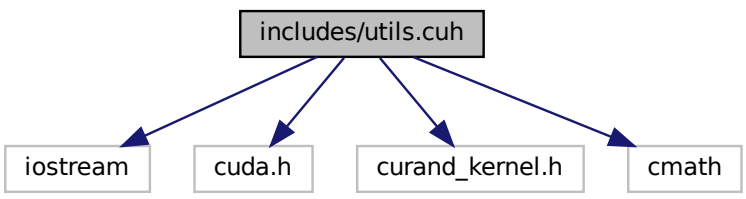
the boolean value of true if hitten or false if not

An optimized version of sphere->hit which skips a lot of vtable fetches and som function calls to gain performances at runtime

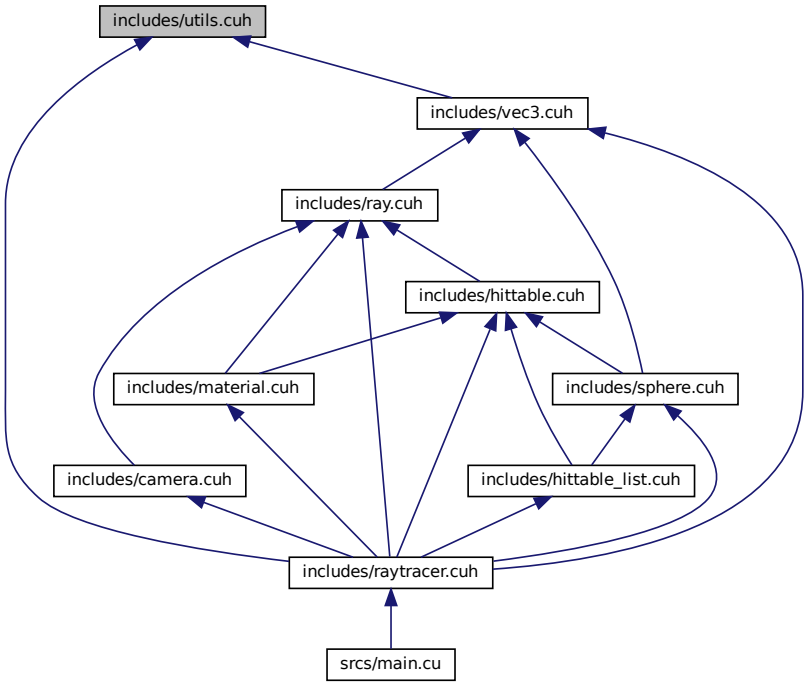
5.8 includes/utils.cuh File Reference

```
#include <iostream>
#include <cuda.h>
#include <curand_kernel.h>
#include <cmath>
```

Include dependency graph for utils.cuh:



This graph shows which files directly or indirectly include this file:



Macros

- #define UTILS_CUH
- #define CHECK(val) check_cuda((val), #val, __FILE__, __LINE__)

Functions

- void check_cuda (cudaError_t res, const char ∗func, const char ∗file, const int line)

  A function to check for errors.

5.8.1 Macro Definition Documentation

5.8.1.1 CHECK

```
#define CHECK(
              val ) check_cuda( (val), #val, __FILE__, __LINE__ )
```

5.8.1.2 UTILS_CUH

```
#define UTILS_CUH
```

5.8.2 Function Documentation

5.8.2.1 check_cuda()

```
void check_cuda (
            cudaError_t res,
            const char * func,
            const char * file,
            const int line )
```

A function to check for errors.

Parameters

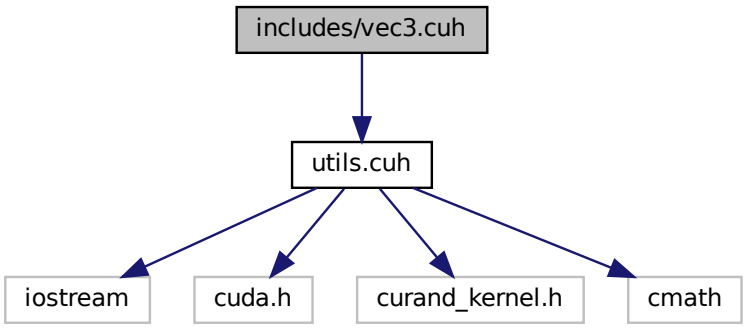
res	the cudaError result of the checked function
func	the function that was called
file	the string of the origin file
line	the number of line where the error occurred

This function checks the result of a called function and throws an error waring with detailed info and stops the execution

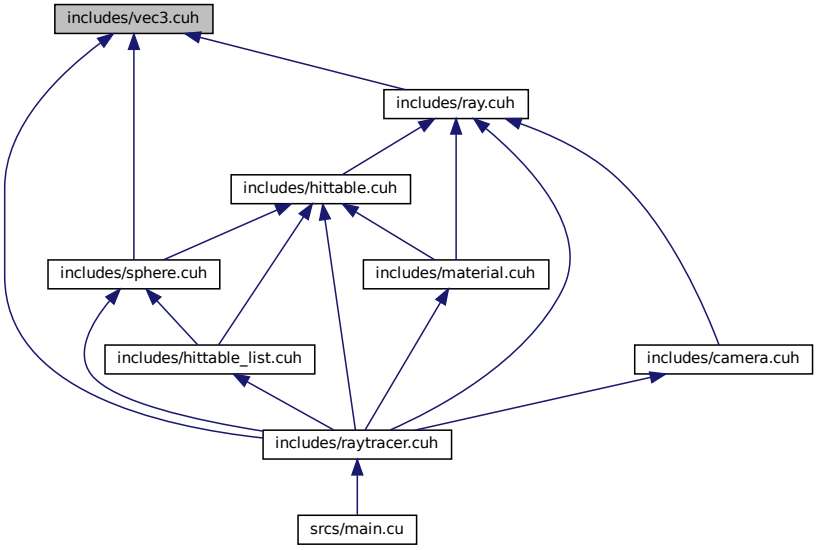
5.9 includes/vec3.cuh File Reference

```
#include "utils.cuh"
```

Include dependency graph for vec3.cuh:



This graph shows which files directly or indirectly include this file:



Classes

- class vec3

Macros

- #define VEC3_CUH

Functions

- std::istream & operator>> (std::istream &in, vec3 &v)
  The istream operator.
- std::ostream & operator<< (std::ostream &out, const vec3 &v)
  The ostream operator.
- __host__ __device__ vec3 operator+ (const vec3 &u, const vec3 &v)
  The sum operator.
- __host__ __device__ vec3 operator- (const vec3 &u, const vec3 &v)
  The sub operator.
- __host__ __device__ vec3 operator* (const vec3 &u, const vec3 &v)
  The mul operator.
- __host__ __device__ vec3 operator* (const float t, const vec3 &v)
  The mul operator.
- __host__ __device__ vec3 operator* (const vec3 &v, const float t)
  The mul operator.
- __host__ __device__ vec3 operator/ (const vec3 &v, const float t)
  The div operator.
- __host__ __device__ vec3 operator/ (const float t, const vec3 &v)

The div operator.

- __host__ __device__ vec3 operator/ (const vec3 &u, const vec3 &v)

  The div operator.
- __host__ __device__ float dot (const vec3 &u, const vec3 &v)

  The dot operator.
- __host__ __device__ vec3 cross (const vec3 &u, const vec3 &v)

  The cross operator.
- __host__ __device__ vec3 unit_vector (vec3 v)

  Computes the unit of a vector.

5.9.1 Macro Definition Documentation

5.9.1.1 VEC3_CUH

```
#define VEC3_CUH
```

5.9.2 Function Documentation

5.9.2.1 cross()

```
__host__ __device__ vec3 cross (
            const vec3 & u,
            const vec3 & v )  [inline]
```

The cross operator.

Parameters

u	the first vec3
v	the second vec3

Returns

The cross of u and v

5.9.2.2 dot()

```
__host__ __device__ float dot (
            const vec3 & u,
            const vec3 & v )  [inline]
```

The dot operator.

Parameters

u	the first vec3
v	the second vec3

Returns

The dot of u and v

5.9.2.3 operator∗() [1/3]

```
__host__ __device__ vec3 operator* (
            const float t,
            const vec3 & v )  [inline]
```

The mul operator.

Parameters

t	the scalar
v	the vec3

Returns

The multiplication of t and v

5.9.2.4 operator∗() [2/3]

```
__host__ __device__ vec3 operator* (
            const vec3 & u,
            const vec3 & v )  [inline]
```

The mul operator.

Parameters

u	the first vec3
v	the second vec3

Returns

The multiplication of u and v

5.9.2.5 operator∗() [3/3]

```
__host__ __device__ vec3 operator* (
            const vec3 & v,
            const float t )  [inline]
```

The mul operator.

Parameters

v	the vec3
t	the scalar

Returns

The multiplication of v and t

5.9.2.6 operator+()

```
__host__ __device__ vec3 operator+ (
            const vec3 & u,
            const vec3 & v )  [inline]
```

The sum operator.

Parameters

u	the first vec3
v	the second vec3

Returns

The sum of u and v

5.9.2.7 operator-()

```
__host__ __device__ vec3 operator- (
            const vec3 & u,
            const vec3 & v )  [inline]
```

The sub operator.

Parameters

u	the first vec3
v	the second vec3

Returns

The subtraction of u and v

5.9.2.8 operator/() [1/3]

```
__host__ __device__ vec3 operator/ (
            const float t,
            const vec3 & v )  [inline]
```

The div operator.

Parameters

t	the scalar
v	the vec3

Returns

The division of v and t

5.9.2.9 operator/() [2/3]

```
__host__ __device__ vec3 operator/ (
            const vec3 & u,
            const vec3 & v )  [inline]
```

The div operator.

Parameters

u	the first vec3
v	the second vec3

Returns

The division of u and v

5.9.2.10 operator/() [3/3]

```
__host__ __device__ vec3 operator/ (
            const vec3 & v,
            const float t )  [inline]
```

The div operator.

Parameters

v	the vec3
t	the scalar

Returns

The division of v and t

5.9.2.11 operator<<()

```
std::ostream& operator<< (
            std::ostream & out,
            const vec3 & v )  [inline]
```

The ostream operator.

Parameters

out	the ostream
v	the vec3

Returns

The output streaming of the vec3

5.9.2.12 operator>>()

```
std::istream& operator>> (
            std::istream & in,
            vec3 & v )  [inline]
```

The istream operator.

Parameters

in	the istream
v	the vec3

Returns

The input streaming of the vec3

5.9.2.13 unit_vector()

```
__host__ __device__ vec3 unit_vector (
            vec3 v )  [inline]
```

Computes the unit of a vector.

Parameters

v	the vec3

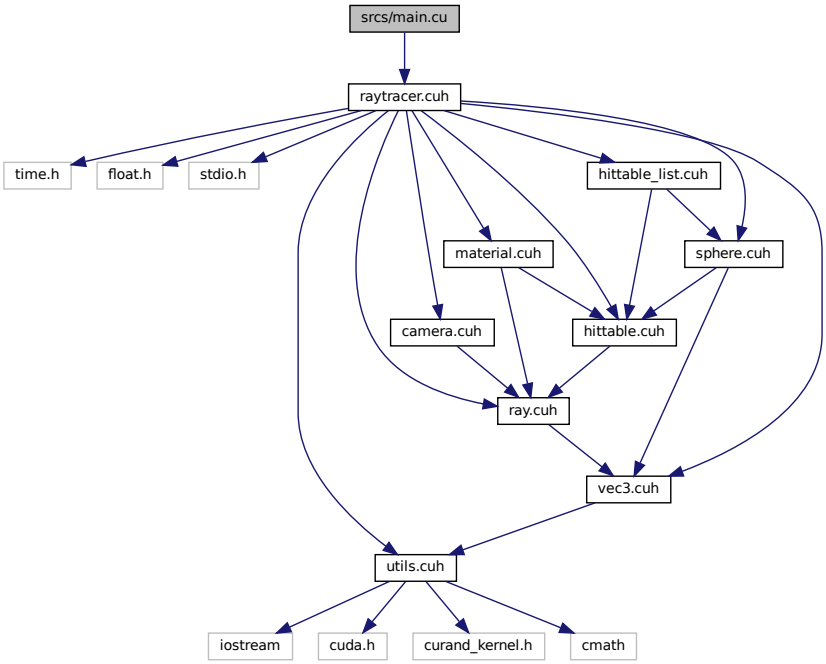
Returns

The unit of v

5.10 srcs/main.cu File Reference

```
#include "raytracer.cuh"
```

Include dependency graph for main.cu:



Macros

- #define RND (curand_uniform(&local_rand_state))

Functions

- __device__ vec3 ray_color (const ray &r, hittable **world, curandState *rand_state)

  Computes the path of a given ray.
- __global__ void render (vec3 *buf, camera **cam, hittable **world, curandState *rand_state)

  Computes the color of a given pixel.
- __global__ void rand_init (curandState *rand_state)

  Initialize a random state.
- __global__ void create_world (hittable **d_list, hittable **d_world, camera **d_camera, curandState *rand_state)

  Initialize the current world.
- __global__ void free_world (hittable **d_list, hittable **d_world, camera **d_camera)

  Deletes the world data on the device.
- void write_color (std::ostream &out, vec3 pixel)

  Writes color in the oust stream.
- void print (vec3 *buf)

  Writes the image into the output.
- int main (void)

  The main function.

5.10.1 Macro Definition Documentation

5.10.1.1 RND

```
#define RND (curand_uniform(&local_rand_state))
```

5.10.2 Function Documentation

5.10.2.1 create_world()

```
__global__ void create_world (
            hittable ** d_list,
            hittable ** d_world,
            camera ** d_camera,
            curandState * rand_state )
```

Initialize the current world.

Parameters

d_list	the list of objects in the world
d_world	the world to render
d_camera	the camera to render from
rand_state	the current random state

This function initialize the given random state

5.10.2.2 free_world()

```
__global__ void free_world (
            hittable ** d_list,
            hittable ** d_world,
            camera ** d_camera )
```

Deletes the world data on the device.

Parameters

d_list	the list of objects in the world
d_world	the world to render
d_camera	the camera to render from

This function deletes all the previously created objects in the world and the world itself with its list

5.10.2.3 main()

```
int main (
            void  )
```

The main function.

In this function anfter allocating the host needed memory, the world initializer is called, then a kernel is deployed for every pixel to compute the colors, that are printed after the coputation, ending with the memory deallocation.

5.10.2.4 print()

```
void print (
            vec3 * buf )  [inline]
```

Writes the image into the output.

Parameters

buf	the matrix representing the image

This function writes the image into the output stream by using ppm format

5.10.2.5 rand_init()

```
__global__ void rand_init (
            curandState * rand_state )
```

Initialize a random state.

Parameters

rand_state	the current random state

This function initialize the given random state

5.10.2.6 ray_color()

```
__device__ vec3 ray_color (
            const ray & r,
            hittable ** world,
            curandState * rand_state )
```

Computes the path of a given ray.

Parameters

r	the ray to analyze
world	the current world where the ray is been analyzed
rand_state	the current random state

Returns

the color of the resulting ray's path

This function computes the path of a give ray, by hitting a material and scattering until no hits occours or when approaching the refraction limit

5.10.2.7 render()

```
__global__ void render (
            vec3 * buf,
            camera ** cam,
            hittable ** world,
            curandState * rand_state )
```

Computes the color of a given pixel.

Parameters

buf	the final buffer image
cam	the camera where the rays are coming from
world	the current world to analyze with the rays
rand_state	the current random state

This function computes the color of a given pixel by scattering a fixed amount of sample rays to approximate the color of the given area, it can also improve the quality of the resulting image by using also the samples of the adiacent pixels via shared memory and executing some gamma corretion to correct the output color.

5.10.2.8 write_color()

```
void write_color (
            std::ostream & out,
            vec3 pixel )  [inline]
```

Writes color in the oust stream.

Parameters

out	the output stream
pixel	the color to write

This function writes a color given as uniform vec3 into an out ostream

Index