

# Cuda Ray Tracer

Generated by Doxygen 1.9.1



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 camera Class Reference	7
4.1.1 Detailed Description	8
4.1.2 Constructor & Destructor Documentation	8
4.1.2.1 camera()	8
4.1.3 Member Function Documentation	9
4.1.3.1 get_ray()	9
4.1.4 Member Data Documentation	9
4.1.4.1 horizontal	9
4.1.4.2 lens_radius	9
4.1.4.3 lower_left_corner	9
4.1.4.4 origin	10
4.1.4.5 u	10
4.1.4.6 v	10
4.1.4.7 vertical	10
4.1.4.8 w	10
4.2 dielectric Class Reference	11
4.2.1 Detailed Description	12
4.2.2 Constructor & Destructor Documentation	12
4.2.2.1 dielectric()	12
4.2.3 Member Function Documentation	12
4.2.3.1 scatter()	12
4.2.4 Member Data Documentation	13
4.2.4.1 ir	13
4.3 hittable Class Reference	13
4.3.1 Detailed Description	14
4.3.2 Member Function Documentation	14
4.3.2.1 hit()	14
4.4 hittable_list Class Reference	14
4.4.1 Detailed Description	15
4.4.2 Constructor & Destructor Documentation	16
4.4.2.1 hittable_list() [1/2]	16
4.4.2.2 hittable_list() [2/2]	16
4.4.3 Member Function Documentation	16

4.4.3.1 hit()	16
4.4.4 Member Data Documentation	17
4.4.4.1 list	17
4.4.4.2 size	17
4.5 lambertian Class Reference	17
4.5.1 Detailed Description	18
4.5.2 Constructor & Destructor Documentation	18
4.5.2.1 lambertian()	18
4.5.3 Member Function Documentation	19
4.5.3.1 scatter()	19
4.5.4 Member Data Documentation	19
4.5.4.1 albedo	19
4.6 material Class Reference	20
4.6.1 Detailed Description	20
4.6.2 Member Function Documentation	20
4.6.2.1 scatter()	20
4.7 metal Class Reference	21
4.7.1 Detailed Description	22
4.7.2 Constructor & Destructor Documentation	22
4.7.2.1 metal()	22
4.7.3 Member Function Documentation	22
4.7.3.1 scatter()	22
4.7.4 Member Data Documentation	23
4.7.4.1 albedo	23
4.7.4.2 fuzz	23
4.8 ray Class Reference	24
4.8.1 Detailed Description	24
4.8.2 Constructor & Destructor Documentation	24
4.8.2.1 ray() [1/2]	25
4.8.2.2 ray() [2/2]	25
4.8.3 Member Function Documentation	25
4.8.3.1 at()	25
4.8.3.2 direction()	25
4.8.3.3 origin()	26
4.8.4 Member Data Documentation	26
4.8.4.1 dir	26
4.8.4.2 orig	26
4.9 s_hit_record Struct Reference	26
4.9.1 Detailed Description	27
4.9.2 Member Data Documentation	27
4.9.2.1 mat	27
4.9.2.2 normal	27

4.9.2.3 p . . . . .	27
4.9.2.4 t . . . . .	27
4.10 sphere Class Reference . . . . .	28
4.10.1 Detailed Description . . . . .	29
4.10.2 Constructor & Destructor Documentation . . . . .	29
4.10.2.1 sphere() [1/2] . . . . .	29
4.10.2.2 sphere() [2/2] . . . . .	29
4.10.3 Member Function Documentation . . . . .	29
4.10.3.1 hit() . . . . .	29
4.10.4 Member Data Documentation . . . . .	30
4.10.4.1 center . . . . .	30
4.10.4.2 mat . . . . .	30
4.10.4.3 radius . . . . .	30
4.11 vec3 Class Reference . . . . .	31
4.11.1 Constructor & Destructor Documentation . . . . .	31
4.11.1.1 vec3() [1/2] . . . . .	32
4.11.1.2 vec3() [2/2] . . . . .	32
4.11.2 Member Function Documentation . . . . .	32
4.11.2.1 b() . . . . .	32
4.11.2.2 g() . . . . .	32
4.11.2.3 length() . . . . .	33
4.11.2.4 length_squared() . . . . .	33
4.11.2.5 operator*=( ) [1/2] . . . . .	33
4.11.2.6 operator*=( ) [2/2] . . . . .	33
4.11.2.7 operator+=( ) . . . . .	34
4.11.2.8 operator-( ) . . . . .	34
4.11.2.9 operator/=( ) . . . . .	34
4.11.2.10 operator[]() [1/2] . . . . .	35
4.11.2.11 operator[]() [2/2] . . . . .	35
4.11.2.12 r() . . . . .	35
4.11.2.13 x() . . . . .	36
4.11.2.14 y() . . . . .	36
4.11.2.15 z() . . . . .	36
4.11.3 Member Data Documentation . . . . .	36
4.11.3.1 e . . . . .	36
<b>5 File Documentation</b>	<b>37</b>
5.1 includes/camera.cuh File Reference . . . . .	37
5.1.1 Macro Definition Documentation . . . . .	38
5.1.1.1 ASPECT_RATIO . . . . .	39
5.1.1.2 CAMERA_CUH . . . . .	39
5.1.1.3 FOCAL_LEN . . . . .	39

5.1.1.4 HORIZONTAL . . . . .	39
5.1.1.5 LOWER_LEFT_CORNER . . . . .	39
5.1.1.6 ORIGIN . . . . .	39
5.1.1.7 VERTICAL . . . . .	39
5.1.1.8 VIEW_H . . . . .	39
5.1.1.9 VIEW_W . . . . .	40
5.1.2 Function Documentation . . . . .	40
5.1.2.1 O_get_ray() . . . . .	40
5.1.2.2 unit_disk_rand() . . . . .	40
5.2 includes/hitable.cuh File Reference . . . . .	41
5.2.1 Macro Definition Documentation . . . . .	42
5.2.1.1 HITTABLE_CUH . . . . .	42
5.2.2 Typedef Documentation . . . . .	42
5.2.2.1 t_hit_record . . . . .	42
5.3 includes/hitable_list.cuh File Reference . . . . .	43
5.3.1 Macro Definition Documentation . . . . .	44
5.3.1.1 HITTABLE_LIST_CUH . . . . .	44
5.3.2 Function Documentation . . . . .	44
5.3.2.1 O_hit() . . . . .	45
5.4 includes/material.cuh File Reference . . . . .	45
5.4.1 Macro Definition Documentation . . . . .	47
5.4.1.1 MATERIAL_CUH . . . . .	47
5.4.2 Function Documentation . . . . .	47
5.4.2.1 reflect() . . . . .	47
5.4.2.2 refract() . . . . .	48
5.4.2.3 schlick() . . . . .	48
5.4.2.4 unit_sphere_rand() . . . . .	49
5.5 includes/ray.cuh File Reference . . . . .	49
5.5.1 Macro Definition Documentation . . . . .	50
5.5.1.1 RAY_CUH . . . . .	50
5.6 includes/raytracer.cuh File Reference . . . . .	50
5.6.1 Macro Definition Documentation . . . . .	52
5.6.1.1 BLOCK_H . . . . .	52
5.6.1.2 BLOCK_W . . . . .	52
5.6.1.3 BSIZE . . . . .	52
5.6.1.4 H . . . . .	52
5.6.1.5 MANEGED . . . . .	52
5.6.1.6 PIXELS . . . . .	52
5.6.1.7 RAYTRACER_CUH . . . . .	53
5.6.1.8 REFRACTION . . . . .	53
5.6.1.9 SAMPLES . . . . .	53
5.6.1.10 SEED . . . . .	53

5.6.1.11 SHARED . . . . .	53
5.6.1.12 W . . . . .	53
5.6.1.13 WEIGHT . . . . .	53
5.7 includes/sphere.cuh File Reference . . . . .	54
5.7.1 Macro Definition Documentation . . . . .	55
5.7.1.1 SPHERE_CUH . . . . .	55
5.7.2 Function Documentation . . . . .	56
5.7.2.1 O_hit() . . . . .	56
5.8 includes/utils.cuh File Reference . . . . .	56
5.8.1 Macro Definition Documentation . . . . .	57
5.8.1.1 CHECK . . . . .	57
5.8.1.2 UTILS_CUH . . . . .	58
5.8.2 Function Documentation . . . . .	58
5.8.2.1 check_cuda() . . . . .	58
5.9 includes/vec3.cuh File Reference . . . . .	58
5.9.1 Macro Definition Documentation . . . . .	60
5.9.1.1 VEC3_CUH . . . . .	60
5.9.2 Function Documentation . . . . .	60
5.9.2.1 cross() . . . . .	60
5.9.2.2 dot() . . . . .	60
5.9.2.3 operator*() [1/3] . . . . .	61
5.9.2.4 operator*() [2/3] . . . . .	61
5.9.2.5 operator*() [3/3] . . . . .	62
5.9.2.6 operator+() . . . . .	62
5.9.2.7 operator-() . . . . .	62
5.9.2.8 operator/() [1/3] . . . . .	63
5.9.2.9 operator/() [2/3] . . . . .	63
5.9.2.10 operator/() [3/3] . . . . .	63
5.9.2.11 operator<<() . . . . .	64
5.9.2.12 operator>>() . . . . .	64
5.9.2.13 unit_vector() . . . . .	65
5.10 srcs/main.cu File Reference . . . . .	65
5.10.1 Macro Definition Documentation . . . . .	66
5.10.1.1 RND . . . . .	66
5.10.2 Function Documentation . . . . .	66
5.10.2.1 create_world() . . . . .	66
5.10.2.2 free_world() . . . . .	67
5.10.2.3 main() . . . . .	67
5.10.2.4 print() . . . . .	67
5.10.2.5 rand_init() . . . . .	67
5.10.2.6 ray_color() . . . . .	68
5.10.2.7 render() . . . . .	68

5.10.2.8 <code>write_color()</code> . . . . .	69
---	----

<b>Index</b>	<b>71</b>
--------------	-----------



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

camera . . . . .	7
hittable . . . . .	13
hittable_list . . . . .	14
sphere . . . . .	28
material . . . . .	20
dielectric . . . . .	11
lambertian . . . . .	17
metal . . . . .	21
ray . . . . .	24
s_hit_record . . . . .	26
vec3 . . . . .	31



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">camera</a>	The class representing the camera (POV) . . . . .	7
<a href="#">dielectric</a>	A dielectric material derived from material class . . . . .	11
<a href="#">hittable</a>	This class represents an hittable object . . . . .	13
<a href="#">hittable_list</a>	A list of hittables . . . . .	14
<a href="#">lambertian</a>	A lambertian material derived from material class . . . . .	17
<a href="#">material</a>	The abstract class of materials . . . . .	20
<a href="#">metal</a>	A metal material derived from material class . . . . .	21
<a href="#">ray</a>	Class representing rays . . . . .	24
<a href="#">s_hit_record</a>	This struct contains hit record informations . . . . .	26
<a href="#">sphere</a>	A sphere object derived by hittable . . . . .	28
<a href="#">vec3</a>	. . . . .	31



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

includes/camera.cuh . . . . .	37
includes/hitable.cuh . . . . .	41
includes/hitable_list.cuh . . . . .	43
includes/material.cuh . . . . .	45
includes/ray.cuh . . . . .	49
includes/raytracer.cuh . . . . .	50
includes/sphere.cuh . . . . .	54
includes/utils.cuh . . . . .	56
includes/vec3.cuh . . . . .	58
srcs/main.cu . . . . .	65



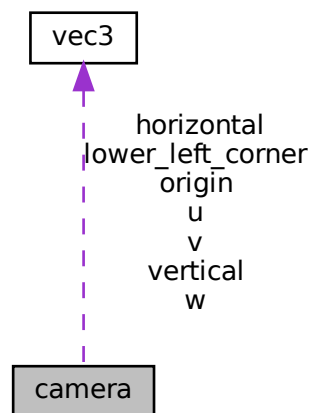
## Chapter 4

# 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, float 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

<i>lookfrom</i>	the origin point
<i>lookat</i>	where to look at
<i>vup</i>	the camera relative up direction
<i>vfov</i>	the vertical fov
<i>aspect</i>	the aspect ratio of the camera
<i>aperture</i>	the aperture of the lens of the camera
<i>focus_dist</i>	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

<i>s</i>	the uniform pointed x coordinate
<i>t</i>	the uniform pointed y coordinate
<i>state</i>	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

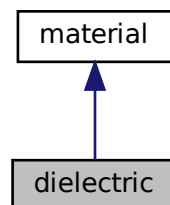
The documentation for this class was generated from the following file:

- includes/[camera.cuh](#)

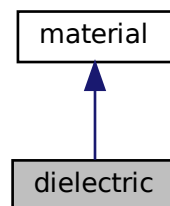
## 4.2 dielectric Class Reference

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

<i>ir</i>	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

<i>r_in</i>	the input ray
<i>rec</i>	the hit point informations
<i>attenuation</i>	filled with material attenuation
<i>scattered</i>	the scattered ray
<i>state</i>	the random state

#### Returns

true if scattered else false

Computes the scatter of a given ray

Implements [material](#).

### 4.2.4 Member Data Documentation

#### 4.2.4.1 ir

```
float dielectric::ir
```

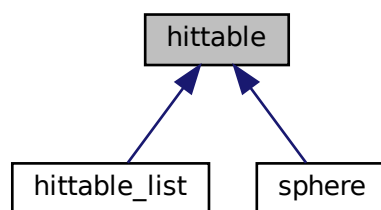
The documentation for this class was generated from the following file:

- 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 hit.*

### 4.3.1 Detailed Description

this class represents an hittable object

This is an abstract class with only virtual methods which will get overridden 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

<i>r</i>	the ray to analyze
<i>t_min</i>	the minimum range span to hit
<i>t_max</i>	the maximum range span to hit
<i>rec</i>	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](#).

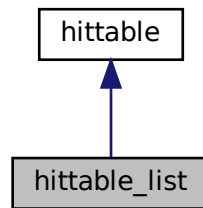
The documentation for this class was generated from the following file:

- includes/[hittable.cuh](#)

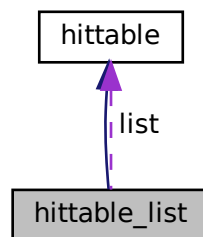
## 4.4 hittable\_list Class Reference

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 **, int n)`  
*Initialized 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]
```

Initialized constructor.

#### Parameters

<i>l</i>	a list of hittables
<i>n</i>	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

<i>r</i>	the ray to analyze
<i>t_min</i>	the minimum range span to hit
<i>t_max</i>	the maximum range span to hit
<i>rec</i>	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
```

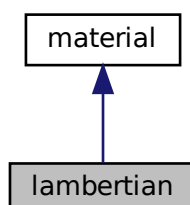
The documentation for this class was generated from the following file:

- includes/[hittable\\_list.cuh](#)

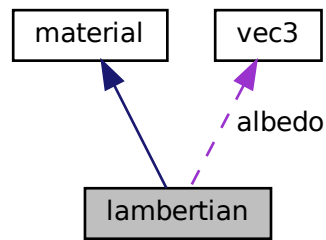
## 4.5 lambertian Class Reference

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

<i>a</i>	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

<i>r_in</i>	the input ray
<i>rec</i>	the hit point informations
<i>attenuation</i>	filled with material attenuation
<i>scattered</i>	the scattered ray
<i>state</i>	the random state

## Returns

true if scattered else false

Computes the scatter of a given ray

Implements [material](#).

### 4.5.4 Member Data Documentation

#### 4.5.4.1 albedo

```
vec3 lambertian::albedo
```

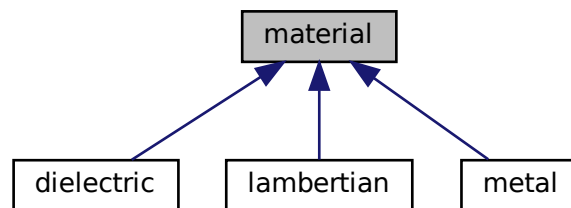
The documentation for this class was generated from the following file:

- 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

<i>r_in</i>	the input ray
<i>rec</i>	the hit point informations
<i>attenuation</i>	filled with material attenuation
<i>scattered</i>	the scattered ray
<i>state</i>	the random state

## Returns

true if scattered else false

Computes the scatter of a given ray

Implemented in [dielectric](#), [metal](#), and [lambertian](#).

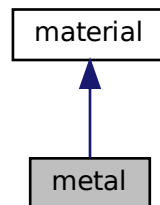
The documentation for this class was generated from the following file:

- includes/[material.cuh](#)

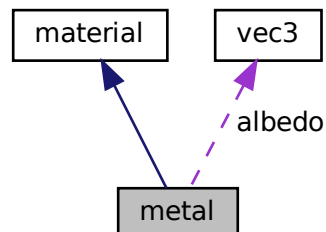
## 4.7 metal Class Reference

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

<i>a</i>	the metal albedo
<i>f</i>	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

<i>r_in</i>	the input ray
<i>rec</i>	the hit point informations
<i>attenuation</i>	filled with material attenuation
<i>scattered</i>	the scattered ray
<i>state</i>	the random state

#### Returns

true if scattered else false

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
```

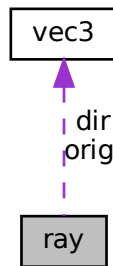
The documentation for this class was generated from the following file:

- 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**

<i>origin</i>	the origin point of the ray
<i>direction</i>	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
```

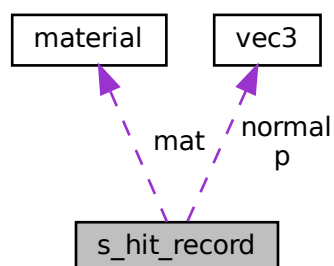
The documentation for this class was generated from the following file:

- 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
```

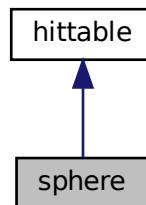
The documentation for this struct was generated from the following file:

- [includes/hitable.cuh](#)

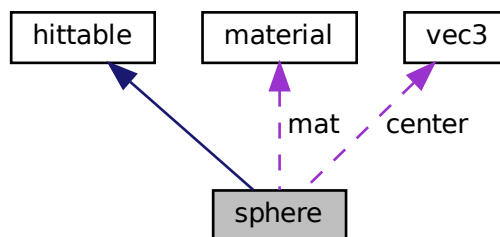
## 4.10 sphere Class Reference

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 hit.*

### 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

<i>c</i>	the center of the sphere
<i>r</i>	the radius of the sphere
<i>m</i>	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 hit.

**Parameters**

<i>r</i>	the ray to analyze
<i>t_min</i>	the minimum range span to hit
<i>t_max</i>	the maximum range span to hit
<i>rec</i>	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**

<i>e0</i>	the x of the <a href="#">vec3</a>
<i>e1</i>	the y of the <a href="#">vec3</a>
<i>e2</i>	the z of the <a href="#">vec3</a>

**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

<i>t</i>	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 <a href="#">vec3</a> 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 <a href="#">vec3</a> 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

<i>t</i>	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>i</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>i</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](#)

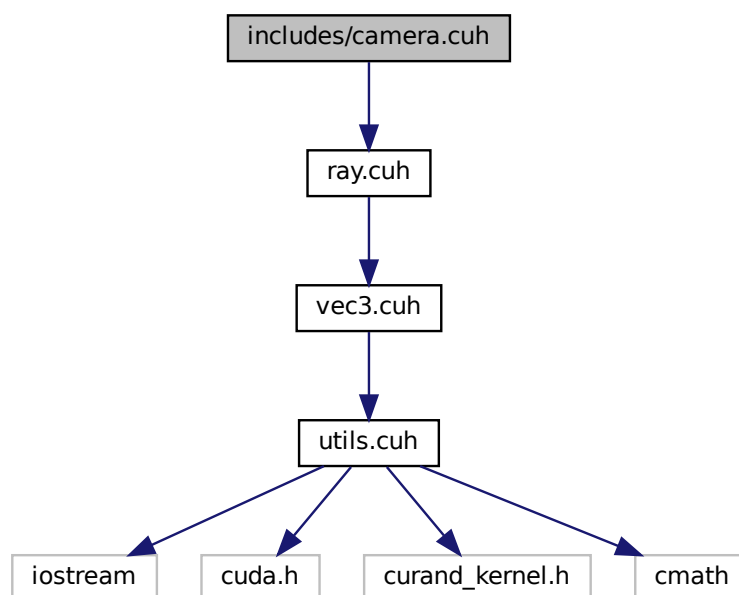
## Chapter 5

# File Documentation

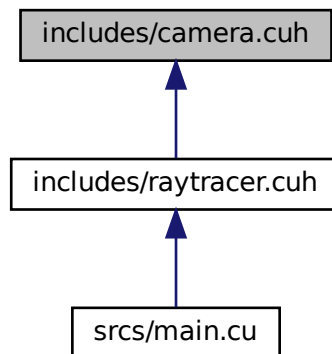
### 5.1 includes/camera.cuh File Reference

```
#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

<i>c</i>	the camera
<i>s</i>	the uniform pointed x coordinate
<i>t</i>	the uniform pointed y coordinate
<i>state</i>	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

<i>s</i>	the pointer of the curandState
----------	--------------------------------

#### Returns

a random uniform [vec3](#)

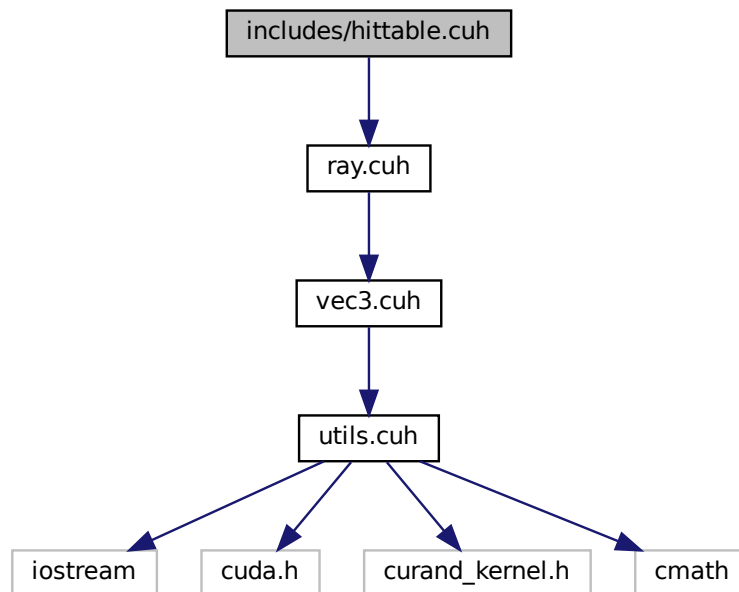
Given the current cuda random state, it generates a uniform [vec3](#)



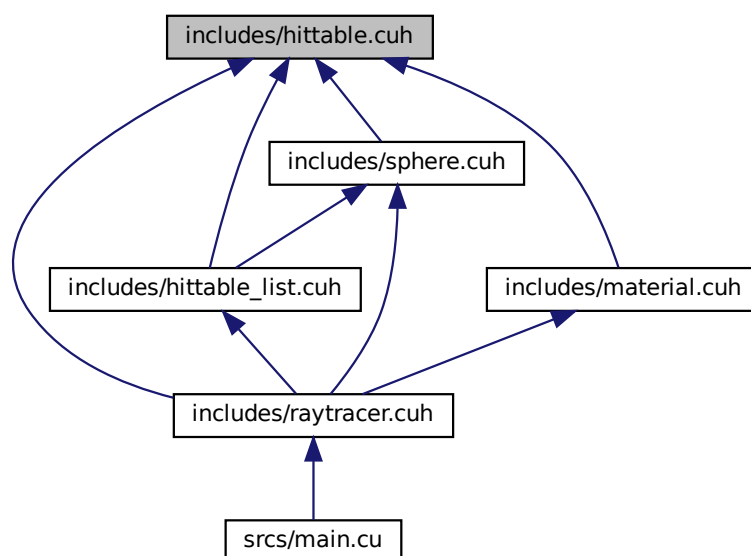
## 5.2 includes/hitable.cuh File Reference

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

Include dependency graph for hitable.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

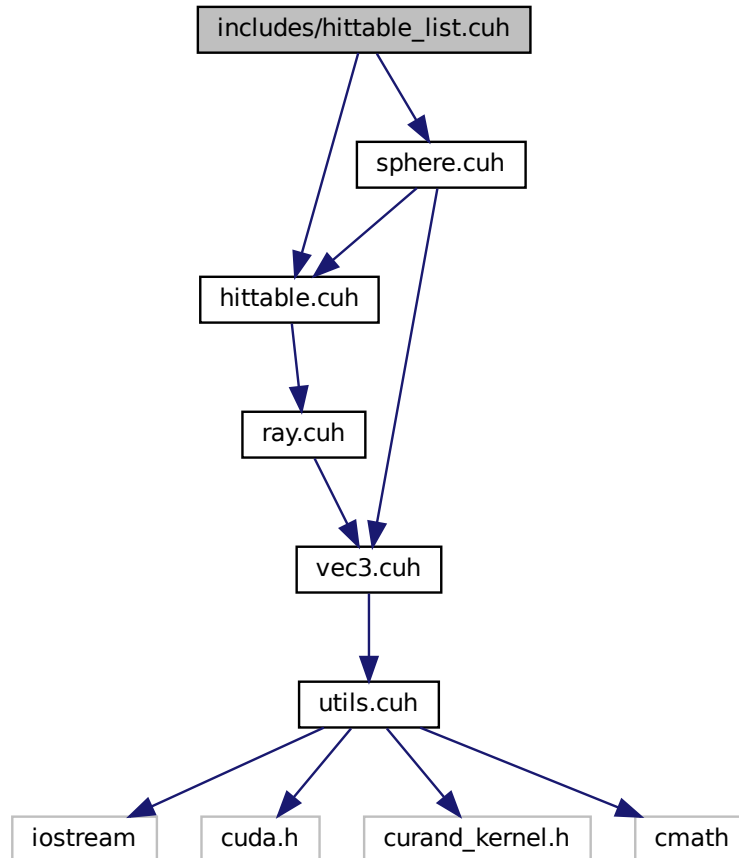
This struct is made of useful informations about hits like the hitted material, the hit distance, the point of impact and the normal of the hitten point.

## 5.3 includes/hittable\_list.cuh File Reference

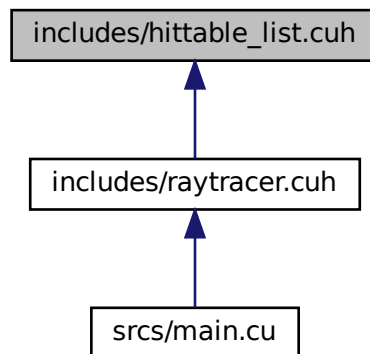
```
#include "hittable.cuh"
```

```
#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

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

#### Returns

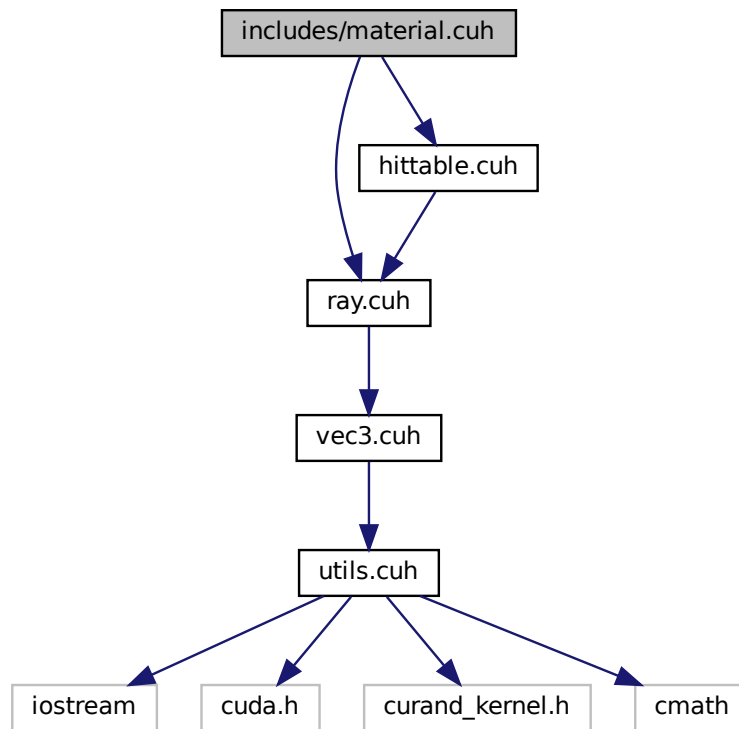
the boolean value of true if hitten or false if not

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

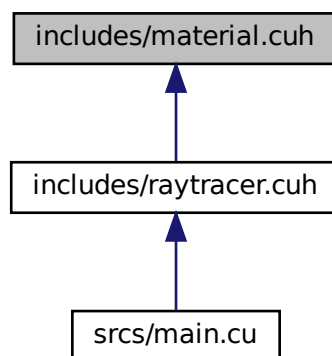
## 5.4 includes/material.cuh File Reference

```
#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 approximated 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**

<i>v</i>	the direction of the ray
<i>u</i>	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**

<i>v</i>	direction unit vector
<i>u</i>	outward unit vector
<i>ni_over_nt</i>	normals refraction indices
<i>refracted</i>	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 approximated refraction.

**Parameters**

<i>cos</i>	cosine of the refraction
<i>ir</i>	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**

<b>s</b>	the pointer of the curandState
----------	--------------------------------

**Returns**

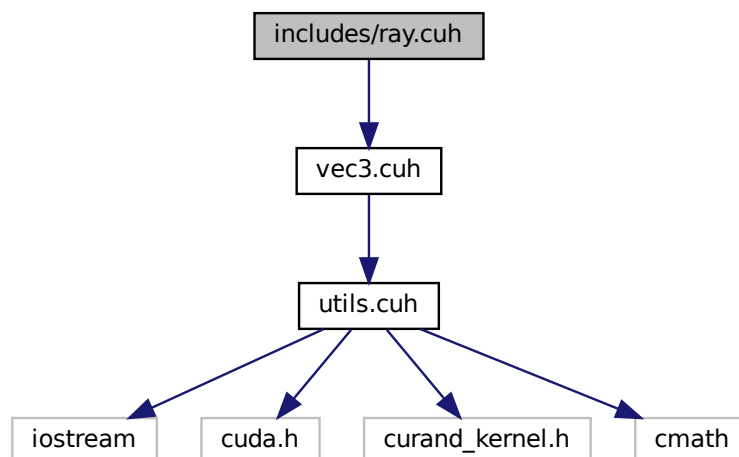
a random uniform [vec3](#)

Given the current cuda random state, it generates a uniform [vec3](#)

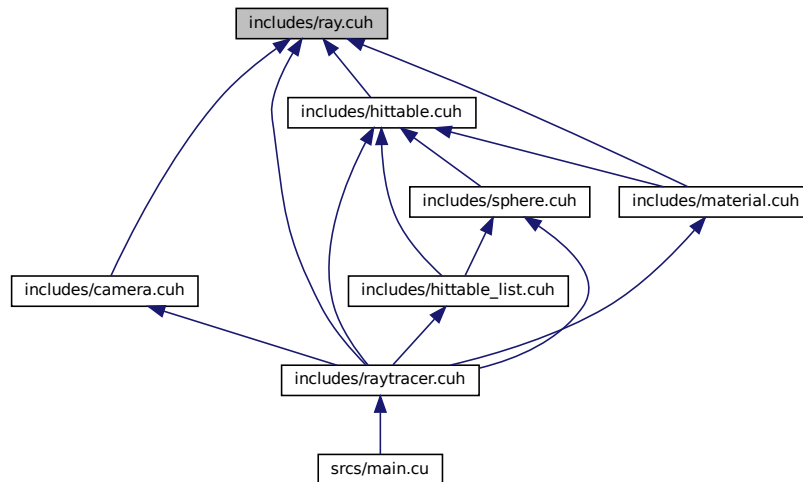
**5.5 includes/ray.cuh File Reference**

```
#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

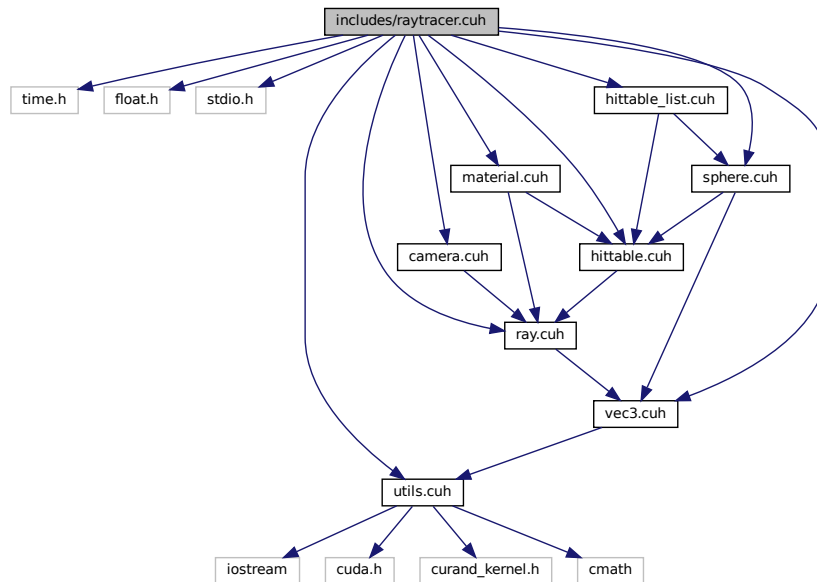
#### 5.5.1.1 RAY\_CUH

```
#define RAY_CUH
```

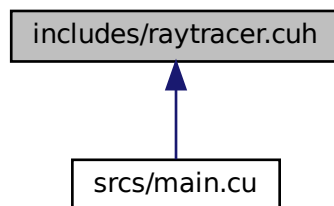
## 5.6 includes/raytracer.cuh File Reference

```
#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

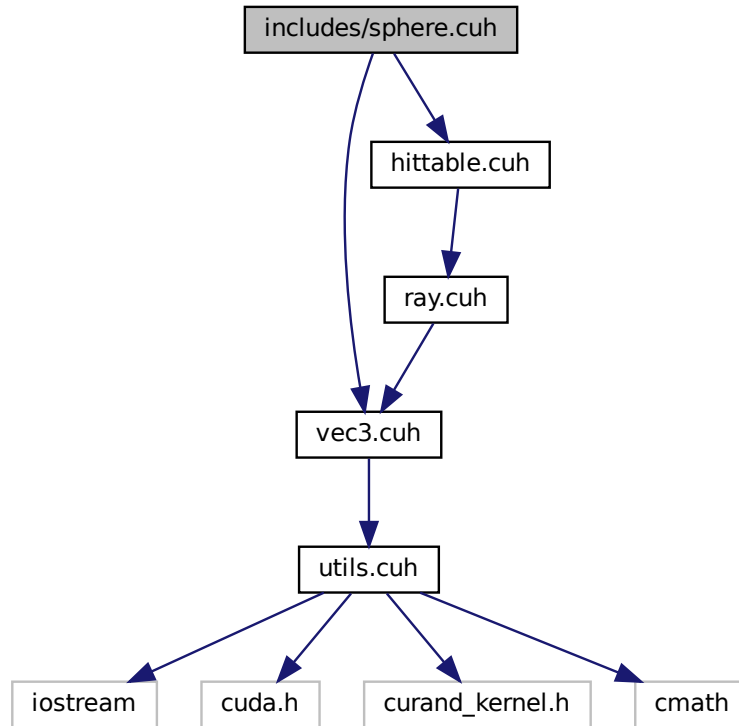
```
#define WEIGHT 0.5f
```

## 5.7 includes/sphere.cuh File Reference

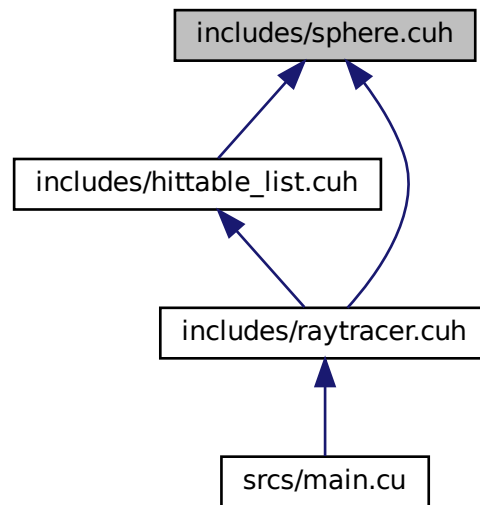
```
#include "vec3.cuh"
```

```
#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

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

#### Returns

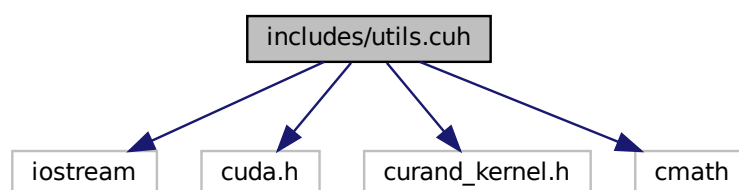
the boolean value of true if hitten or false if not

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:







### 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.

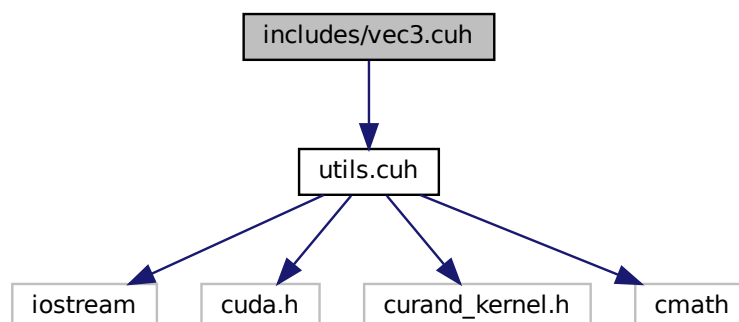
#### Parameters

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

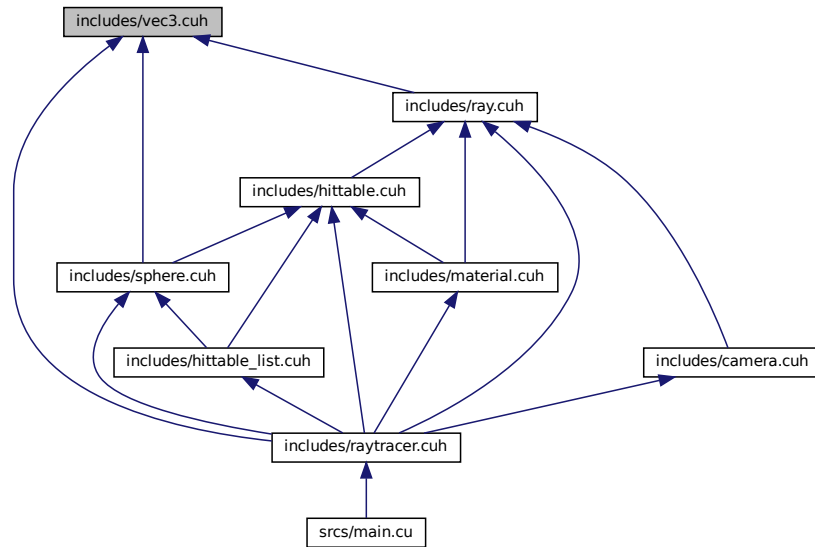
This function checks the result of a called function and throws an error waring with detailed info and stops the execution

## 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

<i>u</i>	the first <code>vec3</code>
<i>v</i>	the second <code>vec3</code>

#### 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**

<i>u</i>	the first <code>vec3</code>
<i>v</i>	the second <code>vec3</code>

**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**

<i>t</i>	the scalar
<i>v</i>	the <code>vec3</code>

**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**

<i>u</i>	the first <code>vec3</code>
<i>v</i>	the second <code>vec3</code>

**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

<i>v</i>	the <a href="#">vec3</a>
<i>t</i>	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

<i>u</i>	the first <a href="#">vec3</a>
<i>v</i>	the second <a href="#">vec3</a>

#### 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

<i>u</i>	the first <a href="#">vec3</a>
<i>v</i>	the second <a href="#">vec3</a>

**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**

<i>t</i>	the scalar
<i>v</i>	the <a href="#">vec3</a>

**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**

<i>u</i>	the first <a href="#">vec3</a>
<i>v</i>	the second <a href="#">vec3</a>

**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**

<i>v</i>	the <a href="#">vec3</a>
<i>t</i>	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**

<i>out</i>	the ostream
<i>v</i>	the <a href="#">vec3</a>

**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**

<i>in</i>	the istream
<i>v</i>	the <a href="#">vec3</a>

**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

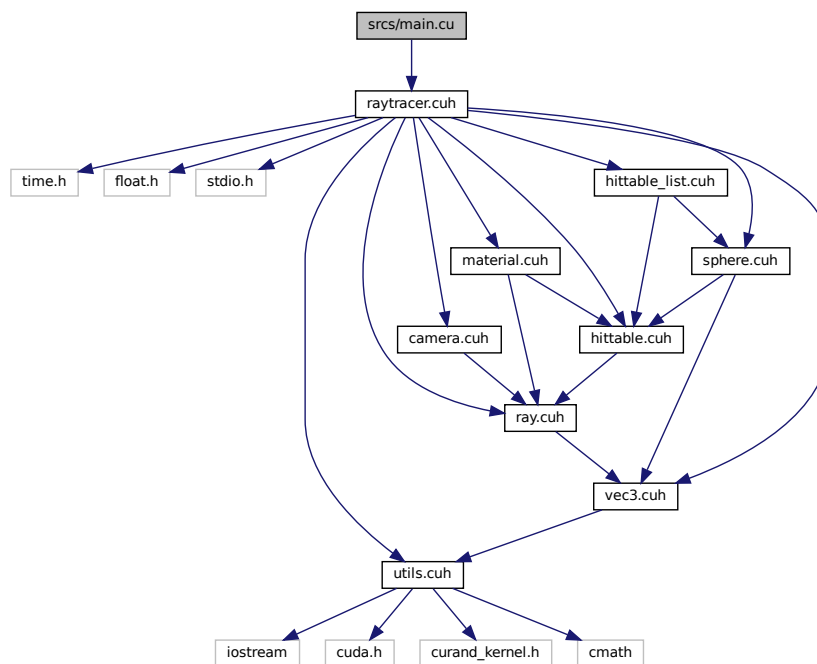
<code>v</code>	the <code>vec3</code>
----------------	-----------------------

#### 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

<i>d_list</i>	the list of objects in the world
<i>d_world</i>	the world to render
<i>d_camera</i>	the camera to render from
<i>rand_state</i>	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

<i>d_list</i>	the list of objects in the world
<i>d_world</i>	the world to render
<i>d_camera</i>	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

<i>buf</i>	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**

<i>rand_state</i>	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**

<i>r</i>	the ray to analyze
<i>world</i>	the current world where the ray is been analyzed
<i>rand_state</i>	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 occurs 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**

<i>buf</i>	the final buffer image
<i>cam</i>	the camera where the rays are coming from
<i>world</i>	the current world to analyze with the rays
<i>rand_state</i>	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 adjacent pixels via shared memory and executing some gamma correction to correct the output color.

### 5.10.2.8 write\_color()

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

Writes color in the out stream.

#### Parameters

<i>out</i>	the output stream
<i>pixel</i>	the color to write

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



# Index

- albedo
  - lambertian, [19](#)
  - metal, [23](#)
- ASPECT\_RATIO
  - camera.cuh, [38](#)
- at
  - ray, [25](#)
- b
  - vec3, [32](#)
- BLOCK\_H
  - raytracer.cuh, [52](#)
- BLOCK\_W
  - raytracer.cuh, [52](#)
- BSIZE
  - raytracer.cuh, [52](#)
- camera, [7](#)
  - camera, [8](#)
  - get\_ray, [9](#)
  - horizontal, [9](#)
  - lens\_radius, [9](#)
  - lower\_left\_corner, [9](#)
  - origin, [9](#)
  - u, [10](#)
  - v, [10](#)
  - vertical, [10](#)
  - w, [10](#)
- camera.cuh
  - ASPECT\_RATIO, [38](#)
  - CAMERA\_CUH, [39](#)
  - FOCAL\_LEN, [39](#)
  - HORIZONTAL, [39](#)
  - LOWER\_LEFT\_CORNER, [39](#)
  - O\_get\_ray, [40](#)
  - ORIGIN, [39](#)
  - unit\_disk\_rand, [40](#)
  - VERTICAL, [39](#)
  - VIEW\_H, [39](#)
  - VIEW\_W, [39](#)
- CAMERA\_CUH
  - camera.cuh, [39](#)
- center
  - sphere, [30](#)
- CHECK
  - utils.cuh, [57](#)
- check\_cuda
  - utils.cuh, [58](#)
- create\_world
  - main.cu, [66](#)
- cross
  - vec3.cuh, [60](#)
- dielectric, [11](#)
  - dielectric, [12](#)
  - ir, [13](#)
  - scatter, [12](#)
- dir
  - ray, [26](#)
- direction
  - ray, [25](#)
- dot
  - vec3.cuh, [60](#)
- e
  - vec3, [36](#)
- FOCAL\_LEN
  - camera.cuh, [39](#)
- free\_world
  - main.cu, [67](#)
- fuzz
  - metal, [23](#)
- g
  - vec3, [32](#)
- get\_ray
  - camera, [9](#)
- H
  - raytracer.cuh, [52](#)
- hit
  - hittable, [14](#)
  - hittable\_list, [16](#)
  - sphere, [29](#)
- hittable, [13](#)
  - hit, [14](#)
- hittable.cuh
  - HITTABLE\_CUH, [42](#)
  - t\_hit\_record, [42](#)
- HITTABLE\_CUH
  - hittable.cuh, [42](#)
- hittable\_list, [14](#)
  - hit, [16](#)
  - hittable\_list, [16](#)
  - list, [17](#)
  - size, [17](#)
- hittable\_list.cuh
  - HITTABLE\_LIST\_CUH, [44](#)
  - O\_hit, [44](#)
- HITTABLE\_LIST\_CUH

- hittable\_list.cuh, 44
- HORIZONTAL
  - camera.cuh, 39
- horizontal
  - camera, 9
- includes/camera.cuh, 37
- includes/hittable.cuh, 41
- includes/hittable\_list.cuh, 43
- includes/material.cuh, 45
- includes/ray.cuh, 49
- includes/raytracer.cuh, 50
- includes/sphere.cuh, 54
- includes/utils.cuh, 56
- includes/vec3.cuh, 58
- ir
  - dielectric, 13
- lambertian, 17
  - albedo, 19
  - lambertian, 18
  - scatter, 19
- length
  - vec3, 32
- length\_squared
  - vec3, 33
- lens\_radius
  - camera, 9
- list
  - hittable\_list, 17
- LOWER\_LEFT\_CORNER
  - camera.cuh, 39
- lower\_left\_corner
  - camera, 9
- main
  - main.cu, 67
- main.cu
  - create\_world, 66
  - free\_world, 67
  - main, 67
  - print, 67
  - rand\_init, 67
  - ray\_color, 68
  - render, 68
  - RND, 66
  - write\_color, 68
- MANEGED
  - raytracer.cuh, 52
- mat
  - s\_hit\_record, 27
  - sphere, 30
- material, 20
  - scatter, 20
- material.cuh
  - MATERIAL\_CUH, 47
  - reflect, 47
  - refract, 48
  - schlick, 48
  - unit\_sphere\_rand, 49
- MATERIAL\_CUH
  - material.cuh, 47
- metal, 21
  - albedo, 23
  - fuzz, 23
  - metal, 22
  - scatter, 22
- normal
  - s\_hit\_record, 27
- O\_get\_ray
  - camera.cuh, 40
- O\_hit
  - hittable\_list.cuh, 44
  - sphere.cuh, 56
- operator<<
  - vec3.cuh, 64
- operator>>
  - vec3.cuh, 64
- operator\*
  - vec3.cuh, 61
- operator\*=
  - vec3, 33
- operator+
  - vec3.cuh, 62
- operator+=
  - vec3, 34
- operator-
  - vec3, 34
  - vec3.cuh, 62
- operator/
  - vec3.cuh, 63
- operator/=
  - vec3, 34
- operator[]
  - vec3, 35
- orig
  - ray, 26
- ORIGIN
  - camera.cuh, 39
- origin
  - camera, 9
  - ray, 25
- p
  - s\_hit\_record, 27
- PIXELS
  - raytracer.cuh, 52
- print
  - main.cu, 67
- r
  - vec3, 35
- radius
  - sphere, 30
- rand\_init
  - main.cu, 67



- ray, 24
  - at, 25
  - dir, 26
  - direction, 25
  - orig, 26
  - origin, 25
  - ray, 24, 25
- ray.cuh
  - RAY\_CUH, 50
- ray\_color
  - main.cu, 68
- RAY\_CUH
  - ray.cuh, 50
- raytracer.cuh
  - BLOCK\_H, 52
  - BLOCK\_W, 52
  - BSIZE, 52
  - H, 52
  - MANEGED, 52
  - PIXELS, 52
  - RAYTRACER\_CUH, 52
  - REFRACTION, 53
  - SAMPLES, 53
  - SEED, 53
  - SHARED, 53
  - W, 53
  - WEIGHT, 53
- RAYTRACER\_CUH
  - raytracer.cuh, 52
- reflect
  - material.cuh, 47
- refract
  - material.cuh, 48
- REFRACTION
  - raytracer.cuh, 53
- render
  - main.cu, 68
- RND
  - main.cu, 66
- s\_hit\_record, 26
  - mat, 27
  - normal, 27
  - p, 27
  - t, 27
- SAMPLES
  - raytracer.cuh, 53
- scatter
  - dielectric, 12
  - lambertian, 19
  - material, 20
  - metal, 22
- schlick
  - material.cuh, 48
- SEED
  - raytracer.cuh, 53
- SHARED
  - raytracer.cuh, 53
- size
  - hittable\_list, 17
- sphere, 28
  - center, 30
  - hit, 29
  - mat, 30
  - radius, 30
  - sphere, 29
- sphere.cuh
  - O\_hit, 56
  - SPHERE\_CUH, 55
- SPHERE\_CUH
  - sphere.cuh, 55
- srcs/main.cu, 65
- t
  - s\_hit\_record, 27
- t\_hit\_record
  - hittable.cuh, 42
- u
  - camera, 10
- unit\_disk\_rand
  - camera.cuh, 40
- unit\_sphere\_rand
  - material.cuh, 49
- unit\_vector
  - vec3.cuh, 64
- utils.cuh
  - CHECK, 57
  - check\_cuda, 58
  - UTILS\_CUH, 57
- UTILS\_CUH
  - utils.cuh, 57
- v
  - camera, 10
- vec3, 31
  - b, 32
  - e, 36
  - g, 32
  - length, 32
  - length\_squared, 33
  - operator\*=, 33
  - operator+=, 34
  - operator-, 34
  - operator/=: 34
  - operator[], 35
  - r, 35
  - vec3, 31, 32
  - x, 36
  - y, 36
  - z, 36
- vec3.cuh
  - cross, 60
  - dot, 60
  - operator<<, 64
  - operator>>, 64
  - operator\*, 61
  - operator+, 62

- operator-, [62](#)
  - operator/, [63](#)
  - unit\_vector, [64](#)
  - VEC3\_CUH, [60](#)
- VEC3\_CUH
  - vec3.cuh, [60](#)
- VERTICAL
  - camera.cuh, [39](#)
- vertical
  - camera, [10](#)
- VIEW\_H
  - camera.cuh, [39](#)
- VIEW\_W
  - camera.cuh, [39](#)
- W
  - raytracer.cuh, [53](#)
- w
  - camera, [10](#)
- WEIGHT
  - raytracer.cuh, [53](#)
- write\_color
  - main.cu, [68](#)
- x
  - vec3, [36](#)
- y
  - vec3, [36](#)
- z
  - vec3, [36](#)