Software structure

Generated by Doxygen 1.9.8

1 Source content	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Class Documentation	9
5.1 box Class Reference	9
5.1.1 Detailed Description	9
5.1.2 Constructor & Destructor Documentation	9
5.1.2.1 box()	9
5.1.3 Member Function Documentation	10
5.1.3.1 hit()	10
5.2 camera Class Reference	10
5.2.1 Detailed Description	11
5.2.2 Constructor & Destructor Documentation	11
5.2.2.1 camera()	11
5.2.3 Member Function Documentation	11
5.2.3.1 get_ray()	11
5.2.3.2 getLensRadius()	12
5.2.3.3 getOrigin()	12
5.3 dielectric Class Reference	12
5.3.1 Detailed Description	13
5.3.2 Constructor & Destructor Documentation	13
5.3.2.1 dielectric()	13
5.3.3 Member Function Documentation	13
5.3.3.1 scatter()	13
5.4 diffuse_light Class Reference	14
5.4.1 Detailed Description	14
5.4.2 Constructor & Destructor Documentation	14
5.4.2.1 diffuse_light()	14
5.4.3 Member Function Documentation	15
5.4.3.1 emitted()	15
5.4.3.2 scatter()	15
5.5 hit_record Class Reference	15
5.5.1 Detailed Description	16
5.5.2 Member Function Documentation	16
5.5.2.1 set_face_normal()	16
5.5.3 Member Data Documentation	16

5.5.3.1 front_face	16
5.5.3.2 mat_ptr	16
5.5.3.3 normal	17
5.5.3.4 p	17
5.5.3.5 t	17
5.5.3.6 v	17
5.6 hittable Class Reference	17
5.6.1 Detailed Description	18
5.6.2 Member Function Documentation	18
5.6.2.1 hit()	18
5.7 hittable_list Class Reference	18
5.7.1 Detailed Description	19
5.7.2 Constructor & Destructor Documentation	19
5.7.2.1 hittable_list()	19
5.7.3 Member Function Documentation	19
5.7.3.1 add()	19
5.7.3.2 hit()	20
5.7.4 Member Data Documentation	20
5.7.4.1 objects	20
5.8 interval Class Reference	20
5.8.1 Detailed Description	21
5.8.2 Constructor & Destructor Documentation	21
5.8.2.1 interval()	21
5.8.3 Member Function Documentation	21
5.8.3.1 clamp()	21
5.8.3.2 contains()	22
5.8.3.3 surrounds()	22
5.8.4 Member Data Documentation	22
5.8.4.1 max	22
5.9 Is Class Reference	23
5.9.1 Detailed Description	23
5.10 lambertian Class Reference	23
5.10.1 Detailed Description	23
5.10.2 Constructor & Destructor Documentation	23
5.10.2.1 lambertian()	23
5.10.3 Member Function Documentation	24
5.10.3.1 scatter()	24
5.11 material Class Reference	24
5.11.1 Detailed Description	25
5.11.2 Member Function Documentation	25
5.11.2.1 emitted()	25
5.11.2.2 scatter()	25

5.12 metal Class Reference	25
5.12.1 Detailed Description	26
5.12.2 Constructor & Destructor Documentation	26
5.12.2.1 metal()	26
5.12.3 Member Function Documentation	26
5.12.3.1 scatter()	26
5.13 moving_sphere Class Reference	27
5.13.1 Detailed Description	27
5.13.2 Constructor & Destructor Documentation	27
5.13.2.1 moving_sphere()	27
5.13.3 Member Function Documentation	28
5.13.3.1 hit()	28
5.14 PixelInfo Struct Reference	28
5.14.1 Detailed Description	29
5.15 ray Class Reference	29
5.15.1 Detailed Description	29
5.15.2 Constructor & Destructor Documentation	30
5.15.2.1 ray()	30
5.15.3 Member Function Documentation	30
5.15.3.1 at()	30
5.15.3.2 direction()	30
5.15.3.3 origin()	31
5.15.3.4 time()	31
5.16 sphere Class Reference	31
5.16.1 Detailed Description	32
5.16.2 Constructor & Destructor Documentation	32
5.16.2.1 sphere()	32
5.16.3 Member Function Documentation	32
5.16.3.1 hit()	32
5.17 vec3 Class Reference	33
5.17.1 Detailed Description	34
5.17.2 Constructor & Destructor Documentation	34
5.17.2.1 vec3()	34
5.17.3 Member Function Documentation	34
5.17.3.1 length()	34
5.17.3.2 length_squared()	34
5.17.3.3 near_zero()	35
5.17.3.4 operator*=()	35
5.17.3.5 operator+=()	35
5.17.3.6 operator-()	36
5.17.3.7 operator/=()	36
5.17.3.8 operator[]() [1/2]	36

	5.17.3.9 operator[]() [2/2]	37
	5.17.3.10 random() [1/2]	37
	<b>5.17.3.11 random()</b> [2/2]	37
	5.17.3.12 x()	38
	5.17.3.13 y()	38
	5.17.3.14 z()	38
	5.18 xy_rect Class Reference	38
	5.18.1 Constructor & Destructor Documentation	39
	5.18.1.1 xy_rect()	39
	5.18.2 Member Function Documentation	39
	5.18.2.1 hit()	39
	5.19 xz_rect Class Reference	40
	5.19.1 Constructor & Destructor Documentation	40
	5.19.1.1 xz_rect()	40
	5.19.2 Member Function Documentation	41
	5.19.2.1 hit()	41
	5.20 yz_rect Class Reference	41
	5.20.1 Constructor & Destructor Documentation	42
	5.20.1.1 yz_rect()	42
	5.20.2 Member Function Documentation	42
	5.20.2.1 hit()	42
۱ ۵	File Desumentation	42
6 I	File Documentation	<b>43</b>
6 I	6.1 box.hpp	43
6 I	6.1 box.hpp	43 43
6 I	6.1 box.hpp	43 43 44
6 I	6.1 box.hpp	43 43 44 44
6 I	6.1 box.hpp  6.2 src/camera.hpp File Reference  6.2.1 Detailed Description  6.3 camera.hpp  6.4 hittable.hpp	43 43 44 44 45
6 I	6.1 box.hpp 6.2 src/camera.hpp File Reference 6.2.1 Detailed Description 6.3 camera.hpp 6.4 hittable.hpp 6.5 hittable_list.hpp	43 43 44 44 45 45
6 1	6.1 box.hpp 6.2 src/camera.hpp File Reference 6.2.1 Detailed Description 6.3 camera.hpp 6.4 hittable.hpp 6.5 hittable_list.hpp 6.6 interval.hpp	43 43 44 44 45 45
6 I	6.1 box.hpp 6.2 src/camera.hpp File Reference 6.2.1 Detailed Description 6.3 camera.hpp 6.4 hittable.hpp 6.5 hittable_list.hpp 6.6 interval.hpp 6.7 src/material.hpp File Reference	43 43 44 44 45 45 45
6 I	6.1 box.hpp 6.2 src/camera.hpp File Reference 6.2.1 Detailed Description 6.3 camera.hpp 6.4 hittable.hpp 6.5 hittable_list.hpp 6.6 interval.hpp 6.7 src/material.hpp File Reference 6.7.1 Detailed Description	43 43 44 44 45 45 46 46
6 I	6.1 box.hpp 6.2 src/camera.hpp File Reference 6.2.1 Detailed Description 6.3 camera.hpp 6.4 hittable.hpp 6.5 hittable_list.hpp 6.6 interval.hpp 6.7 src/material.hpp File Reference 6.7.1 Detailed Description 6.8 material.hpp	43 43 44 44 45 45 46 46 47
6 I	6.1 box.hpp 6.2 src/camera.hpp File Reference 6.2.1 Detailed Description 6.3 camera.hpp 6.4 hittable.hpp 6.5 hittable_list.hpp 6.6 interval.hpp 6.7 src/material.hpp File Reference 6.7.1 Detailed Description 6.8 material.hpp 6.9 moving_sphere.hpp	43 43 44 45 45 45 46 46 47 47
6 I	6.1 box.hpp 6.2 src/camera.hpp File Reference 6.2.1 Detailed Description 6.3 camera.hpp 6.4 hittable.hpp 6.5 hittable_list.hpp 6.6 interval.hpp 6.7 src/material.hpp File Reference 6.7.1 Detailed Description 6.8 material.hpp 6.9 moving_sphere.hpp 6.10 src/ray.cpp File Reference	43 43 44 44 45 45 46 46 47 47 48
6	6.1 box.hpp 6.2 src/camera.hpp File Reference 6.2.1 Detailed Description 6.3 camera.hpp 6.4 hittable.hpp 6.5 hittable_list.hpp 6.6 interval.hpp 6.7 src/material.hpp File Reference 6.7.1 Detailed Description 6.8 material.hpp 6.9 moving_sphere.hpp 6.10 src/ray.cpp File Reference 6.10.1 Detailed Description	43 43 44 44 45 45 46 46 47 47 48 48
6 I	6.1 box.hpp 6.2 src/camera.hpp File Reference 6.2.1 Detailed Description 6.3 camera.hpp 6.4 hittable.hpp 6.5 hittable_list.hpp 6.6 interval.hpp 6.7 src/material.hpp File Reference 6.7.1 Detailed Description 6.8 material.hpp 6.9 moving_sphere.hpp 6.10 src/ray.cpp File Reference 6.10.1 Detailed Description 6.11 src/ray.hpp File Reference	43 43 44 45 45 45 46 47 47 47 48 48
6 I	6.1 box.hpp 6.2 src/camera.hpp File Reference 6.2.1 Detailed Description 6.3 camera.hpp 6.4 hittable.hpp 6.5 hittable_list.hpp 6.6 interval.hpp 6.7 src/material.hpp File Reference 6.7.1 Detailed Description 6.8 material.hpp 6.9 moving_sphere.hpp 6.10 src/ray.cpp File Reference 6.10.1 Detailed Description 6.11 src/ray.hpp File Reference 6.11.1 Detailed Description	43 43 44 44 45 45 46 46 47 47 48 48 48 48
661	6.1 box.hpp 6.2 src/camera.hpp File Reference 6.2.1 Detailed Description 6.3 camera.hpp 6.4 hittable.hpp 6.5 hittable_list.hpp 6.6 interval.hpp 6.7 src/material.hpp File Reference 6.7.1 Detailed Description 6.8 material.hpp 6.9 moving_sphere.hpp 6.10 src/ray.cpp File Reference 6.10.1 Detailed Description 6.11 src/ray.hpp File Reference 6.11.1 Detailed Description 6.12 ray.hpp	43 43 44 44 45 45 46 47 47 48 48 48 49
661	6.1 box.hpp 6.2 src/camera.hpp File Reference 6.2.1 Detailed Description 6.3 camera.hpp 6.4 hittable.hpp 6.5 hittable_list.hpp 6.6 interval.hpp 6.7 src/material.hpp File Reference 6.7.1 Detailed Description 6.8 material.hpp 6.9 moving_sphere.hpp 6.10 src/ray.cpp File Reference 6.10.1 Detailed Description 6.11 src/ray.hpp File Reference 6.11.1 Detailed Description 6.12 ray.hpp 6.13 rect.hpp	43 43 44 44 45 45 46 47 47 48 48 48 49 49
61	6.1 box.hpp 6.2 src/camera.hpp File Reference 6.2.1 Detailed Description 6.3 camera.hpp 6.4 hittable.hpp 6.5 hittable_list.hpp 6.6 interval.hpp 6.7 src/material.hpp File Reference 6.7.1 Detailed Description 6.8 material.hpp 6.9 moving_sphere.hpp 6.10 src/ray.cpp File Reference 6.10.1 Detailed Description 6.11 src/ray.hpp File Reference 6.11.1 Detailed Description 6.12 ray.hpp	43 43 44 44 45 45 45 46 47 47 48 48 48 49 49 49

6.17 srotvec3.cpp File Reference 6.17.1 Detailed Description 6.17.2 Function Documentation 6.17.2 rorost) 6.17.2.1 cross) 6.17.2.3 operator+() [1/3] 6.17.2.4 operator+() [2/3] 6.17.2.5 operator+() (2/3] 6.17.2.5 operator+() 6.17.2.7 operator-() 6.17.2.9 operator-() 6.17.2.9 operator-() 6.17.2.10 random_in_hemisphere() 6.17.2.11 random_in_unit_disk() 6.17.2.12 random_in_unit_disk() 6.17.2.13 random_unit_vector() 6.18 srotvec3.hpp File Reference 6.18.1 Detailed Description 6.18.2 Function Documentation 6.18.2.1 cross() 6.18.2.3 operator+() [1/3] 6.18.2.4 operator+() [1/3] 6.18.2.5 operator+() [1/3] 6.18.2.5 operator+() 6.18.2.7 operator+() 6.18.2.9 operator+() 6.18.2.9 operator+() 6.18.2.10 random_in_hemisphere() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_disk() 6.18.2.13 random_in_unit_disk() 6.18.2.14 random_in_unit_disk() 6.18.2.13 random_in_unit_disk() 6.18.2.14 random_in_unit_disk() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector()	6.16 utility.hpp	
6.17.2 Function Documentation 6.17.2.1 cross() 6.17.2.2 dot() 6.17.2.3 operator+() [1/3] 6.17.2.4 operator+() [2/3] 6.17.2.5 operator+() [3/3] 6.17.2.6 operator+() 6.17.2.7 operator-() 6.17.2.9 operator-() 6.17.2.9 operator-() 6.17.2.11 random_in_unit_disk() 6.17.2.12 random_in_unit_disk() 6.17.2.13 random_unit_vector() 6.17.2.15 unit_vector() 6.18 src/vec3.hpp File Reference 6.18.1 Detailed Description 6.18.2 Function Documentation 6.18.2.1 cross() 6.18.2.3 operator-() [1/3] 6.18.2.4 operator-() [2/3] 6.18.2.5 operator-() [3/3] 6.18.2.5 operator-() 6.18.2.7 operator-() 6.18.2.9 operator-() 6.18.2.9 operator-() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_hemisphere() 6.18.2.12 random_in_hemisphere() 6.18.2.13 random_in_unit_sphere() 6.18.2.14 refract() 6.18.2.15 unit_vector()	6.17 src/vec3.cpp File Reference	
6.17.2.1 cross() 6.17.2.2 dot() 6.17.2.3 operator*() [1/3] 6.17.2.4 operator*() [2/3] 6.17.2.5 operator*() [2/3] 6.17.2.5 operator*() [3/3] 6.17.2.6 operator*() 6.17.2.7 operator*() 6.17.2.9 operator*() 6.17.2.9 operator*() 6.17.2.10 random_in_hemisphere() 6.17.2.11 random_in_unit_disk() 6.17.2.12 random_in_unit_vector() 6.17.2.13 random_unit_vector() 6.18 src/vec3.hpp File Reference 6.18.1 Detailed Description 6.18.2 Function Documentation 6.18.2.1 cross() 6.18.2.3 operator*() [1/3] 6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [2/3] 6.18.2.6 operator*() [3/3] 6.18.2.6 operator*() 6.18.2.8 operator*() 6.18.2.9 operator*() 6.18.2.9 operator*() 6.18.2.9 operator*() 6.18.2.9 operator*() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector()	6.17.1 Detailed Description	
6.17.2.2 dot() 6.17.2.3 operator*() [1/3] 6.17.2.4 operator*() [2/3] 6.17.2.5 operator*() [3/3] 6.17.2.5 operator*() [3/3] 6.17.2.7 operator*() 6.17.2.7 operator*() 6.17.2.8 operator*() 6.17.2.9 operator*() 6.17.2.10 random_in_hemisphere() 6.17.2.11 random_in_unit_disk() 6.17.2.12 random_unit_vector() 6.17.2.13 random_unit_vector() 6.17.2.15 unit_vector() 6.18 srcvvec3.hpp File Reference 6.18.1 Detailed Description 6.18.2 Function Documentation 6.18.2.1 cross() 6.18.2.3 operator*() [1/3] 6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator*() [3/3] 6.18.2.6 operator*() 6.18.2.7 operator*() 6.18.2.8 operator*() 6.18.2.9 operator*() 6.18.2.1 orandom_in_hemisphere() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector()	6.17.2 Function Documentation	
6.17.2.3 operator*() [1/3] 6.17.2.4 operator*() [2/3] 6.17.2.5 operator*() [3/3] 6.17.2.6 operator+() . 6.17.2.7 operator-() . 6.17.2.8 operator-() . 6.17.2.9 operator-() . 6.17.2.10 random_in_hemisphere() . 6.17.2.11 random_in_unit_disk() . 6.17.2.12 random_in_unit_sphere() . 6.17.2.13 random_unit_vector() . 6.17.2.14 refract() . 6.18 src/vec3.hpp File Reference . 6.18.1 Detailed Description . 6.18.2 Function Documentation . 6.18.2.1 cross() . 6.18.2.3 operator*() [1/3] . 6.18.2.4 operator*() [1/3] . 6.18.2.5 operator*() [1/3] . 6.18.2.6 operator*() [1/3] . 6.18.2.7 operator-() . 6.18.2.8 operator*() . 6.18.2.9 operator-() . 6.18.2.10 random_in_hemisphere() . 6.18.2.11 random_in_unit_disk() . 6.18.2.12 random_in_unit_sphere() . 6.18.2.13 random_unit_vector() . 6.18.2.14 refract() . 6.18.2.15 unit_vector() .	6.17.2.1 cross()	
6.17.2.4 operator*() [2/3] 6.17.2.5 operator*() [3/3] 6.17.2.6 operator+(). 6.17.2.7 operator-() 6.17.2.8 operator-() 6.17.2.9 operator<() 6.17.2.10 random_in_hemisphere() 6.17.2.11 random_in_unit_disk() 6.17.2.12 random_in_unit_sphere() 6.17.2.13 random_unit_vector() 6.17.2.14 refract() 6.17.2.15 unit_vector() 6.18 src/vec3.hpp File Reference 6.18.1 Detailed Description 6.18.2 Function Documentation 6.18.2.1 cross() 6.18.2.2 dot() 6.18.2.3 operator*() [1/3] 6.18.2.3 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator*() 6.18.2.7 operator-() 6.18.2.9 operator-() 6.18.2.9 operator-() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector()	6.17.2.2 dot()	
6.17.2.5 operator*() [3/3] 6.17.2.6 operator+(). 6.17.2.7 operator-() 6.17.2.8 operator/(). 6.17.2.9 operator<<() 6.17.2.10 random_in_hemisphere() 6.17.2.11 random_in_unit_disk(). 6.17.2.12 random_in_unit_sphere() 6.17.2.13 random_unit_vector() 6.17.2.14 refract(). 6.17.2.15 unit_vector() 6.18 src(vec3.hpp File Reference 6.18.1 Detailed Description 6.18.2 Function Documentation 6.18.2.1 cross(). 6.18.2.3 operator*() [1/3] 6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator*() [3/3] 6.18.2.7 operator-(). 6.18.2.9 operator*(). 6.18.2.1 random_in_temisphere(). 6.18.2.11 random_in_temisphere(). 6.18.2.12 random_in_unit_disk(). 6.18.2.13 random_unit_vector(). 6.18.2.14 refract(). 6.18.2.15 unit_vector().	6.17.2.3 operator*() [1/3]	
6.17.2.6 operator+() 6.17.2.7 operator-() 6.17.2.8 operator/() 6.17.2.9 operator<<() 6.17.2.10 random_in_hemisphere() 6.17.2.11 random_in_unit_disk() 6.17.2.12 random_in_unit_sphere() 6.17.2.13 random_unit_vector() 6.17.2.14 refract() 6.17.2.15 unit_vector() 6.18.1 Detailed Description 6.18.2 Function Documentation 6.18.2.1 cross() 6.18.2.2 dot() 6.18.2.3 operator*() [1/3] 6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator() 6.18.2.9 operator<<() 6.18.2.9 operator<<() 6.18.2.1 random_in_hemisphere() 6.18.2.11 random_in_hemisphere() 6.18.2.11 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_sphere() 6.18.2.14 refract() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector()	6.17.2.4 operator*() [2/3]	
6.17.2.7 operator-() 6.17.2.8 operator/() 6.17.2.9 operator<<() 6.17.2.10 random_in_hemisphere() 6.17.2.11 random_in_unit_disk() 6.17.2.12 random_in_unit_sphere() 6.17.2.13 random_unit_vector() 6.17.2.14 refract() 6.17.2.15 unit_vector() 6.18 src/vec3.hpp File Reference 6.18.1 Detailed Description 6.18.2 Function Documentation 6.18.2.1 cross() 6.18.2.2 dot() 6.18.2.3 operator+() [2/3] 6.18.2.4 operator+() [2/3] 6.18.2.5 operator+() [3/3] 6.18.2.6 operator+() (3/3] 6.18.2.7 operator-() 6.18.2.9 operator-() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector()	<b>6.17.2.5 operator*()</b> [3/3]	
6.17.2.8 operator/() 6.17.2.9 operator<<() 6.17.2.10 random_in_hemisphere() 6.17.2.11 random_in_unit_disk() 6.17.2.12 random_in_unit_sphere() 6.17.2.13 random_unit_vector() 6.17.2.14 refract() 6.17.2.15 unit_vector() 6.18 src/vec3.hpp File Reference 6.18.1 Detailed Description 6.18.2 Function Documentation 6.18.2.1 cross() 6.18.2.2 dot() 6.18.2.3 operator*() [2/3] 6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator+() 6.18.2.7 operator-() 6.18.2.9 operator-() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_unit_sphere() 6.18.2.13 random_unit_sphere() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector()	6.17.2.6 operator+()	
6.17.2.9 operator<<() 6.17.2.10 random_in_hemisphere() 6.17.2.11 random_in_unit_disk() 6.17.2.12 random_in_unit_sphere() 6.17.2.13 random_unit_vector() 6.17.2.14 refract() 6.17.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.1 cross() 6.18.2.2 dot() 6.18.2.2 dot() 6.18.2.3 operator*() [1/3] 6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator*() 6.18.2.9 operator*() 6.18.2.9 operator*() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_unit_vector() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector()	6.17.2.7 operator-()	
6.17.2.10 random_in_hemisphere() 6.17.2.11 random_in_unit_disk() 6.17.2.12 random_in_unit_sphere() 6.17.2.13 random_unit_vector() 6.17.2.14 refract() 6.17.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2 Function Documentation 6.18.2 Function Documentation 6.18.2.2 dot() 6.18.2.3 operator*() [1/3] 6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator*() 6.18.2.7 operator-() 6.18.2.9 operator-() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_unit_vector() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector()	6.17.2.8 operator/()	
6.17.2.11 random_in_unit_disk() 6.17.2.12 random_in_unit_sphere() 6.17.2.13 random_unit_vector() 6.17.2.14 refract() 6.17.2.15 unit_vector() 6.18 src/vec3.hpp File Reference 6.18.1 Detailed Description 6.18.2 Function Documentation 6.18.2.1 cross() 6.18.2.2 dot() 6.18.2.2 dot() 6.18.2.3 operator*() [1/3] 6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator*() 6.18.2.7 operator*() 6.18.2.9 operator*() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector()	6.17.2.9 operator<<()	
6.17.2.12 random_in_unit_sphere() 6.17.2.13 random_unit_vector() 6.17.2.14 refract() 6.17.2.15 unit_vector() 6.18 src/vec3.hpp File Reference 6.18.1 Detailed Description 6.18.2 Function Documentation 6.18.2.1 cross() 6.18.2.2 dot() 6.18.2.3 operator*() [1/3] 6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator*() 6.18.2.7 operator*() 6.18.2.9 operator*() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_unit_vector() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.19 vec3.hpp	6.17.2.10 random_in_hemisphere()	
6.17.2.13 random_unit_vector() 6.17.2.14 refract() 6.17.2.15 unit_vector() 6.18 src/vec3.hpp File Reference 6.18.1 Detailed Description 6.18.2 Function Documentation 6.18.2.1 cross() 6.18.2.2 dot() 6.18.2.3 operator*() [1/3] 6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator+() 6.18.2.7 operator-() 6.18.2.8 operator/() 6.18.2.9 operator<() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.19 vec3.hpp	6.17.2.11 random_in_unit_disk()	
6.17.2.14 refract() 6.17.2.15 unit_vector() 6.18 src/vec3.hpp File Reference 6.18.1 Detailed Description 6.18.2 Function Documentation 6.18.2.2 dot() 6.18.2.3 operator*() [1/3] 6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator+() 6.18.2.7 operator-() 6.18.2.8 operator/() 6.18.2.9 operator<() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_sphere() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.18.2.15 unit_vector() 6.19 vec3.hpp	6.17.2.12 random_in_unit_sphere()	
6.17.2.15 unit_vector() 6.18 src/vec3.hpp File Reference 6.18.1 Detailed Description 6.18.2 Function Documentation 6.18.2.1 cross() 6.18.2.2 dot() 6.18.2.3 operator*() [1/3] 6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator+() 6.18.2.7 operator-() 6.18.2.8 operator/() 6.18.2.9 operator/() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.19 vec3.hpp	6.17.2.13 random_unit_vector()	
6.18 src/vec3.hpp File Reference 6.18.1 Detailed Description 6.18.2 Function Documentation 6.18.2.1 cross() 6.18.2.2 dot() 6.18.2.3 operator*() [1/3] 6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator*() 6.18.2.7 operator-() 6.18.2.8 operator/() 6.18.2.9 operator<() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.19 vec3.hpp	6.17.2.14 refract()	
6.18.1 Detailed Description 6.18.2 Function Documentation 6.18.2.1 cross() 6.18.2.2 dot() 6.18.2.3 operator*() [1/3] 6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator+() 6.18.2.7 operator-() 6.18.2.9 operator/() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.19 vec3.hpp	6.17.2.15 unit_vector()	
6.18.2 Function Documentation 6.18.2.1 cross() 6.18.2.2 dot() 6.18.2.3 operator*() [1/3] 6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator+() 6.18.2.7 operator-() 6.18.2.8 operator/() 6.18.2.9 operator<() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.19 vec3.hpp	6.18 src/vec3.hpp File Reference	
6.18.2.1 cross() 6.18.2.2 dot() 6.18.2.3 operator*() [1/3] 6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator+() 6.18.2.7 operator-() 6.18.2.8 operator/() 6.18.2.9 operator<() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.19 vec3.hpp	6.18.1 Detailed Description	
6.18.2.2 dot() 6.18.2.3 operator*() [1/3] 6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator+() 6.18.2.7 operator-() 6.18.2.8 operator/() 6.18.2.9 operator<<() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.19 vec3.hpp	6.18.2 Function Documentation	
6.18.2.3 operator*() [1/3] 6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator+() 6.18.2.7 operator-() 6.18.2.8 operator/() 6.18.2.9 operator<<() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.19 vec3.hpp	6.18.2.1 cross()	
6.18.2.4 operator*() [2/3] 6.18.2.5 operator*() [3/3] 6.18.2.6 operator+() 6.18.2.7 operator-() 6.18.2.8 operator/() 6.18.2.9 operator<<() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.19 vec3.hpp	6.18.2.2 dot()	
6.18.2.5 operator*() [3/3] 6.18.2.6 operator+() 6.18.2.7 operator-() 6.18.2.8 operator/() 6.18.2.9 operator<<() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.19 vec3.hpp	<b>6.18.2.3 operator*()</b> [1/3]	
6.18.2.6 operator+()	6.18.2.4 operator*() [2/3]	
6.18.2.7 operator-() 6.18.2.8 operator/() 6.18.2.9 operator<<() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector()	<b>6.18.2.5 operator*()</b> [3/3]	
6.18.2.8 operator/() 6.18.2.9 operator<<() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.19 vec3.hpp	6.18.2.6 operator+()	
6.18.2.9 operator<<() 6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.19 vec3.hpp	6.18.2.7 operator-()	
6.18.2.10 random_in_hemisphere() 6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.19 vec3.hpp	6.18.2.8 operator/()	
6.18.2.11 random_in_unit_disk() 6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.19 vec3.hpp	6.18.2.9 operator<<()	
6.18.2.12 random_in_unit_sphere() 6.18.2.13 random_unit_vector() 6.18.2.14 refract() 6.18.2.15 unit_vector() 6.19 vec3.hpp	6.18.2.10 random_in_hemisphere()	
6.18.2.13 random_unit_vector() 6.18.2.14 refract()	6.18.2.11 random_in_unit_disk()	
6.18.2.14 refract()	6.18.2.12 random_in_unit_sphere()	
6.18.2.15 unit_vector()	6.18.2.13 random_unit_vector()	
6.19 vec3.hpp	6.18.2.14 refract()	
	6.18.2.15 unit_vector()	
lov	6.19 vec3.hpp	
45A	dex	

## **Chapter 1**

## **Source content**

This folder should contain only hpp/cpp files of your implementation. You can also place hpp files in a separate directory include.

You can create a summary of files here. It might be useful to describe file relations, and brief summary of their content.

2 Source content

# **Chapter 2**

# **Hierarchical Index**

## 2.1 Class Hierarchy

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

camera	. 10
hit_record	. 15
hittable	. 17
box	9
hittable_list	18
moving_sphere	27
sphere	31
xy_rect	38
xz_rect	40
yz_rect	41
interval	. 20
ls	. 23
material	. 24
dielectric	12
diffuse light	
lambertian	23
metal	25
PixelInfo	. 28
ray	. 29
vec3	. 33

4 Hierarchical Index

# **Chapter 3**

# **Class Index**

## 3.1 Class List

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

DOX		
	epresents a axis-aligned box in 3D space	ç
camera R	epresents a movable camera in the world	10
dielectric		
С	lass representing dielectric material	12
diffuse_ligh		
	epresents a light source material	14
hit_record		
Si hittable	tores values related to a hit	15
	ase class for different kind of objects	17
hittable list	•	
R	epresents a list of objects the ray can hit	18
interval		
R	epresents an interval between two points	20
ls		
	epresents a rectangle parallel to the xy-plane	23
lambertian		
	lass representing Lambertian diffusion	23
material		_
	irtual class for all the materials	24
metal	loss representing metal metarial	0.5
moving spl	lass representing metal material	25
		27
PixelInfo	oprosonte moving opnoto	
	epresents one pixel in the picture	28
ray		
•	class representing a ray with an origin point and a direction vector. Heavily inspired by Pe-	
te	er Shirley https://raytracing.github.io/v3/books/RayTracingInOne↔	
M	eekend.html	29
sphere		
R	epresents a sphere	31
vec3		
Α	class representing a 3D vector with x, y, and z components	33

6 Class Index

xy_rect															 						 	(	38
xz_rect															 						 	4	40
yz_rect												-			 						 	4	41

# **Chapter 4**

# **File Index**

## 4.1 File List

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

src/box.hpp	43
The camera class represents a camera used for ray tracing. Heavily inspired by Peter Shirley	
https://raytracing.github.io/v3/books/RayTracingInOneWeekend.↔	
html	43
src/hittable.hpp	45
src/hittable_list.hpp	45
src/interval.hpp	45
src/material.hpp	
The material class represents materials used for ray tracing. Heavily inspired by Peter Shirley	
https://raytracing.github.io/v3/books/RayTracingInOneWeekend.↔	
html	46
	47
src/moving_sphere.hpp	47
src/ray.cpp	
Necessary class definitions of ray with an origin point and a direction vector	48
src/ray.hpp	48
src/rect.hpp	49
src/render.hpp	50
src/sphere.hpp	50
src/utility.hpp	50
src/vec3.cpp	
Cpp files with definitions of vec3 members and non-members	51
src/vec3.hpp	
A C++ header file defining a 3D vector class and related utility functions. Heavily inspired	
by Peter Shirley https://raytracing.github.io/v3/books/RayTracingIn	
OneWeekend.html	E0
UNEWEEKENG. III III	ാറ

8 File Index

## **Chapter 5**

## **Class Documentation**

## 5.1 box Class Reference

Represents a axis-aligned box in 3D space.

```
#include <box.hpp>
```

Inheritance diagram for box:



## **Public Member Functions**

- box (const point3 &p0, const point3 &p1, shared\_ptr< material > ptr)
   Constructs a box with specified parameters.
- virtual bool hit (const ray &r, interval ray\_t, hit\_record &rec) const override
   Determines if a ray hits the box.

#### **Public Attributes**

- point3 box\_min
- point3 box\_max
- hittable\_list sides

## 5.1.1 Detailed Description

Represents a axis-aligned box in 3D space.

## 5.1.2 Constructor & Destructor Documentation

## 5.1.2.1 box()

Constructs a box with specified parameters.

#### **Parameters**

p0	Minimum point of the box.
p1	Maximum point of the box.
ptr	Material of the box.

## 5.1.3 Member Function Documentation

#### 5.1.3.1 hit()

Determines if a ray hits the box.

Given a ray, this function checks if it intersects with the box within the specified ray parameter interval.

#### **Parameters**

r	The ray.	
ray⊷	The valid interval for the ray parameter.	
_t		
rec	The hit record to be filled if the ray hits the box.	

### Returns

True if the ray hits the box, false otherwise.

Implements hittable.

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

- src/box.hpp
- src/box.cpp

## 5.2 camera Class Reference

Represents a movable camera in the world.

```
#include <camera.hpp>
```

#### **Public Member Functions**

• camera (point3 lookfrom, point3 lookat, vec3 vup, double vfov, double aspect\_ratio, double aperture, double focus\_dist, double \_time0, double \_time1)

Constructs a camera with the specified parameters.

• ray get\_ray (double s, double t) const

Generates a ray from the camera for a given screen position.

• point3 getOrigin () const

Gets the origin of the camera.

· double getLensRadius () const

Gets the radius of the lens.

## 5.2.1 Detailed Description

Represents a movable camera in the world.

#### 5.2.2 Constructor & Destructor Documentation

#### 5.2.2.1 camera()

Constructs a camera with the specified parameters.

#### **Parameters**

lookfrom	The camera's position.
lookat	The point the camera is looking at.
vup	The "up" vector indicating the camera's orientation.
vfov	The vertical field-of-view in degrees.
aspect_ratio	The aspect ratio of the viewport.
_time0	Time when the shutter opens for blur
_time1	Time when the shutter closes for blur

## 5.2.3 Member Function Documentation

#### 5.2.3.1 get\_ray()

```
ray camera::get_ray (
```

```
\begin{array}{ll} \mbox{double } s, \\ \mbox{double } t \mbox{ ) const} \end{array}
```

Generates a ray from the camera for a given screen position.

#### **Parameters**

s	The horizontal coordinate on the screen (0 to	
t	The vertical coordinate on the screen (0 to 1).	

#### Returns

The generated ray.

## 5.2.3.2 getLensRadius()

```
double camera::getLensRadius ( ) const [inline]
```

Gets the radius of the lens.

#### Returns

Lens radius

## 5.2.3.3 getOrigin()

```
point3 camera::getOrigin ( ) const [inline]
```

Gets the origin of the camera.

#### Returns

The camera's origin point.

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

- src/camera.hpp
- src/camera.cpp

## 5.3 dielectric Class Reference

Class representing dielectric material.

```
#include <material.hpp>
```

Inheritance diagram for dielectric:



#### **Public Member Functions**

• dielectric (double index\_of\_reftraction)

Constructor for dielectric material.

• virtual bool scatter (const ray &r\_in, const hit\_record &rec, color &attenuation, ray &scattered) const override Calculates the correct scatter of the vector for dielectric material. This should look like glass.

## **Public Member Functions inherited from material**

• virtual color emitted (double u, double v, const point3 &p) const

## 5.3.1 Detailed Description

Class representing dielectric material.

### 5.3.2 Constructor & Destructor Documentation

## 5.3.2.1 dielectric()

Constructor for dielectric material.

### Parameters

index_of_refraction	Index of refraction.
---------------------	----------------------

## 5.3.3 Member Function Documentation

## 5.3.3.1 scatter()

Calculates the correct scatter of the vector for dielectric material. This should look like glass.

#### **Parameters**

r_in	The incoming vector
rec	hit record of hit point.
attenuation	The color change of the ray after hitting.
scattered	The scattered ray.

#### Returns

true if hit.

false otherwise.

Implements material.

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

- src/material.hpp
- · src/material.cpp

## 5.4 diffuse\_light Class Reference

Represents a light source material.

```
#include <material.hpp>
```

Inheritance diagram for diffuse\_light:



### **Public Member Functions**

- diffuse\_light (color c)
   Contructor with a color.
- virtual bool scatter (const ray &r\_in, const hit\_record &rec, color &attenuation, ray &scattered) const override
- virtual color emitted (double u, double v, const point3 &p) const override
   *Emits light*.

## 5.4.1 Detailed Description

Represents a light source material.

#### 5.4.2 Constructor & Destructor Documentation

#### 5.4.2.1 diffuse\_light()

```
\begin{tabular}{ll} \tt diffuse\_light::diffuse\_light (\\ \tt color c) \end{tabular}
```

Contructor with a color.

#### **Parameters**

```
a Given color for the light.
```

#### 5.4.3 Member Function Documentation

## 5.4.3.1 emitted()

```
color diffuse_light::emitted ( \label{eq:color} \mbox{double } u, \\ \mbox{double } v, \\ \mbox{const point3 & $p$ ) const [override], [virtual]}
```

Emits light.

#### **Parameters**

и	Coordinate of object surface.	
V	Coordinate of object surface.	
р	Coordinate of the 3D world.	

#### Returns

Coordinates for the light ray.

Reimplemented from material.

## 5.4.3.2 scatter()

Implements material.

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

- src/material.hpp
- · src/material.cpp

## 5.5 hit\_record Class Reference

Stores values related to a hit.

```
#include <hittable.hpp>
```

## **Public Member Functions**

void set\_face\_normal (const ray &r, const vec3 &outward\_normal)

Sets the nomal for the surface of the object. And also checks if the ray is coming from the outside of object and changes front\_face accordingly.

#### **Public Attributes**

- point3 p
- · vec3 normal
- shared\_ptr< material > mat\_ptr
- double t
- double **u**
- double v
- · bool front face

## 5.5.1 Detailed Description

Stores values related to a hit.

#### 5.5.2 Member Function Documentation

#### 5.5.2.1 set\_face\_normal()

Sets the nomal for the surface of the object. And also checks if the ray is coming from the outside of object and changes front\_face accordingly.

#### **Parameters**

r	The coming ray.
outward_normal	The outward normal of the surface.

## 5.5.3 Member Data Documentation

#### 5.5.3.1 front\_face

```
bool hit_record::front_face
```

Holds information about whether the ray is coming from outside (true).

#### 5.5.3.2 mat\_ptr

```
shared_ptr<material> hit_record::mat_ptr
```

Pointer to the object material.

5.6 hittable Class Reference

#### 5.5.3.3 normal

```
vec3 hit_record::normal
```

The normal of object surface at the hit point.

## 5.5.3.4 p

```
point3 hit_record::p
```

Position of the hit point.

#### 5.5.3.5 t

```
double hit_record::t
```

The size of t or the length when hit.

#### 5.5.3.6 v

```
double hit_record::v
```

Surface coordinates of the object.

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

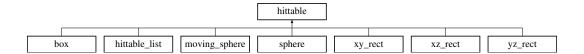
- src/hittable.hpp
- · src/hittable.cpp

## 5.6 hittable Class Reference

a base class for different kind of objects.

```
#include <hittable.hpp>
```

Inheritance diagram for hittable:



### **Public Member Functions**

virtual bool hit (const ray &r, interval ray\_t, hit\_record &rec) const =0
 Chekcs if ray hits. And updates hit record.

## 5.6.1 Detailed Description

a base class for different kind of objects.

#### 5.6.2 Member Function Documentation

## 5.6.2.1 hit()

Chekcs if ray hits. And updates hit record.

#### **Parameters**

r	The coming ray.	
ray⊷	The interval to be checked for intersection.	
_t		
rec	The hit_record of the possible hit	

Implemented in box, hittable\_list, moving\_sphere, xy\_rect, xz\_rect, yz\_rect, and sphere.

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

· src/hittable.hpp

## 5.7 hittable\_list Class Reference

Represents a list of objects the ray can hit.

```
#include <hittable_list.hpp>
```

Inheritance diagram for hittable\_list:



#### **Public Member Functions**

hittable\_list ()

Default constructor.

hittable\_list (shared\_ptr< hittable > object)

Constructor with an initial hittable object.

· void clear ()

Clears the list of objects.

void add (shared\_ptr< hittable > object)

Adds an object to the end of the list.

• bool hit (const ray &r, interval ray\_t, hit\_record &rec) const override

Checks if a ray hits an object in the list given the parameters. And updates the record of that object.

#### **Public Attributes**

std::vector< shared\_ptr< hittable >> objects

## 5.7.1 Detailed Description

Represents a list of objects the ray can hit.

#### 5.7.2 Constructor & Destructor Documentation

#### 5.7.2.1 hittable\_list()

Constructor with an initial hittable object.

#### **Parameters**

object	The object to be added to the list.
Cojoot	The deject to be added to the net.

#### 5.7.3 Member Function Documentation

#### 5.7.3.1 add()

Adds an object to the end of the list.

#### **Parameters**

object	The object to be added to the list.
--------	-------------------------------------

#### 5.7.3.2 hit()

Checks if a ray hits an object in the list given the parameters. And updates the record of that object.

#### **Parameters**

r	The ray.	
ray⊷	Interval or length of ray to be checked for intersection.	
_t		
rec	The hit record to be filled if the ray hits an object.	

#### Returns

True if the ray hits any object in the list, false otherwise.

Implements hittable.

#### 5.7.4 Member Data Documentation

#### 5.7.4.1 objects

```
std::vector<shared_ptr<hittable> > hittable_list::objects
```

Vector containing shared pointers to hittable objects.

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

- · src/hittable list.hpp
- · src/hittable\_list.cpp

## 5.8 interval Class Reference

Represents an interval between two points.

```
#include <interval.hpp>
```

#### **Public Member Functions**

· interval ()

Default constructor with empty interval.

• interval (double \_min, double \_max)

Constructor which sets min and max.

• bool contains (double x) const

Checks if the interval contains certain value within range.

• bool surrounds (double x) const

Chekcs if x is between the pramaters. Not including min and max.

• double clamp (double x) const

Checks if x is outside parameters and returns closest min or max to it.

5.8 interval Class Reference 21

#### **Public Attributes**

- double min
- double max

#### **Static Public Attributes**

- static const interval empty
- · static const interval universe

## 5.8.1 Detailed Description

Represents an interval between two points.

#### 5.8.2 Constructor & Destructor Documentation

## 5.8.2.1 interval()

Constructor which sets min and max.

## **Parameters**

_min	Min value
_max	Max value

## 5.8.3 Member Function Documentation

## 5.8.3.1 clamp()

```
double interval::clamp ( double x ) const
```

Checks if x is outside parameters and returns closest min or max to it.

#### **Parameters**



#### Returns

double

## 5.8.3.2 contains()

```
bool interval::contains ( \mbox{double } x \mbox{ ) const}
```

Checks if the interval contains certain value within range.

#### **Parameters**

```
x The value to be checked for.
```

#### Returns

true if contains within range false otherwise

#### 5.8.3.3 surrounds()

```
bool interval::surrounds ( double x ) const
```

Chekcs if x is between the pramaters. Not including min and max.

#### **Parameters**



#### Returns

true if contains false otherwise

## 5.8.4 Member Data Documentation

#### 5.8.4.1 max

```
double interval::max
```

The parameters for min and max.

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

- src/interval.hpp
- · src/interval.cpp

5.9 Is Class Reference 23

#### 5.9 Is Class Reference

Represents a rectangle parallel to the xy-plane.

## 5.9.1 Detailed Description

Represents a rectangle parallel to the xy-plane.

Represents a rectangle parallel to the yz-plane.

Represents a rectangle parallel to the xz-plane.

aligned rectangle.

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

· src/rect.hpp

## 5.10 lambertian Class Reference

Class representing Lambertian diffusion.

```
#include <material.hpp>
```

Inheritance diagram for lambertian:



#### **Public Member Functions**

- lambertian (const color &a)
   Constructor for Lambertian material.
- virtual bool scatter (const ray &r\_in, const hit\_record &rec, color &attenuation, ray &scattered) const override Calculates the correct scatter of the vector for lambertian material.

#### Public Member Functions inherited from material

• virtual color emitted (double u, double v, const point3 &p) const

#### 5.10.1 Detailed Description

Class representing Lambertian diffusion.

#### 5.10.2 Constructor & Destructor Documentation

#### 5.10.2.1 lambertian()

Constructor for Lambertian material.

#### **Parameters**

```
a Albedo color.
```

### 5.10.3 Member Function Documentation

## 5.10.3.1 scatter()

Calculates the correct scatter of the vector for lambertian material.

#### **Parameters**

r_in	The incoming vector
rec	hit record of hit point.
attenuation	The color change of the ray after hitting.
scattered	The scattered ray.

#### **Returns**

true if hit.

false otherwise.

Implements material.

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

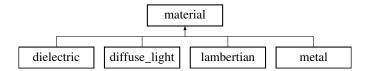
- src/material.hpp
- · src/material.cpp

## 5.11 material Class Reference

virtual class for all the materials.

```
#include <material.hpp>
```

Inheritance diagram for material:



5.12 metal Class Reference 25

#### **Public Member Functions**

- virtual color emitted (double u, double v, const point3 &p) const
- virtual bool scatter (const ray &r\_in, const hit\_record &rec, color &attenuation, ray &scattered) const =0

## 5.11.1 Detailed Description

virtual class for all the materials.

#### 5.11.2 Member Function Documentation

#### 5.11.2.1 emitted()

```
virtual color material::emitted ( \label{eq:color} \mbox{double } u, \\ \mbox{double } v, \\ \mbox{const point3 & $p$ ) const [inline], [virtual]}
```

Reimplemented in diffuse\_light.

#### 5.11.2.2 scatter()

Implemented in lambertian, metal, and dielectric.

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

• src/material.hpp

## 5.12 metal Class Reference

Class representing metal material.

```
#include <material.hpp>
```

Inheritance diagram for metal:



#### **Public Member Functions**

• metal (const color &a, double f)

Constructor for metal material.

• virtual bool scatter (const ray &r\_in, const hit\_record &rec, color &attenuation, ray &scattered) const override Calculates the correct scatter of the vector for metal material. Mertal should acct like a mirror.

## **Public Member Functions inherited from material**

• virtual color emitted (double u, double v, const point3 &p) const

## 5.12.1 Detailed Description

Class representing metal material.

## 5.12.2 Constructor & Destructor Documentation

#### 5.12.2.1 metal()

Constructor for metal material.

#### **Parameters**

а	Albedo color.
f	Fuzziness factor.

#### 5.12.3 Member Function Documentation

#### 5.12.3.1 scatter()

Calculates the correct scatter of the vector for metal material. Mertal should acct like a mirror.

#### **Parameters**

r_in	The incoming vector
rec	hit record of hit point.
attenuation	The color change of the ray after hitting.
scattered	The scattered ray.

#### Returns

true if hit.

false otherwise.

Implements material.

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

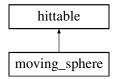
- src/material.hpp
- src/material.cpp

## 5.13 moving\_sphere Class Reference

Represents moving sphere.

```
#include <moving_sphere.hpp>
```

Inheritance diagram for moving\_sphere:



#### **Public Member Functions**

moving\_sphere (point3 cen0, point3 cen1, double \_time0, double \_time1, double r, shared\_ptr< material > m)

Constructor for the moving\_sphere. The moving happens from cen0 to cen1 linearily. \_time0 and \_time1 are used to calculate the ray intersection point.

• virtual bool hit (const ray &r, interval ray\_t, hit\_record &rec) const override

Determines if a ray hits the sphere.

• point3 center (double time) const

return the center the object. During certain time point.

## 5.13.1 Detailed Description

Represents moving sphere.

#### 5.13.2 Constructor & Destructor Documentation

#### 5.13.2.1 moving sphere()

Constructor for the moving\_sphere. The moving happens from cen0 to cen1 linearily. \_time0 and \_time1 are used to calculate the ray intersection point.

#### **Parameters**

cen0	start point of the movement.
cen1	end point of movement.
_time0	start time of movement.
_time1	end point of movement.
r	radius of the sphere
m	material of the sphere

## 5.13.3 Member Function Documentation

#### 5.13.3.1 hit()

Determines if a ray hits the sphere.

#### **Parameters**

r	The ray.
ray⇔	The valid interval for the ray parameter.
_t	
rec	The hit record to be filled if the ray hits the object.

#### Returns

True if the ray hits the box, false otherwise.

Implements hittable.

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

- src/moving\_sphere.hpp
- src/moving\_sphere.cpp

## 5.14 PixelInfo Struct Reference

Represents one pixel in the picture.

```
#include <render.hpp>
```

#### **Public Attributes**

- int x
- int y
- color pixel\_color

## 5.14.1 Detailed Description

Represents one pixel in the picture.

Holds information about the pixel, including its position (x, y) and color.

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

· src/render.hpp

## 5.15 ray Class Reference

A class representing a ray with an origin point and a direction vector. Heavily inspired by Peter Shirley https://raytracing.github.io/v3/books/RayTracingInOneWeekend.html.

```
#include <ray.hpp>
```

### **Public Member Functions**

• ray ()

Default constructor for a ray.

• ray (const point3 &origin, const vec3 &direction, double time=0.0)

Constructor to create a ray with an origin and direction.

• point3 origin () const

Get the origin point of the ray.

vec3 direction () const

Get the direction vector of the ray.

• double time () const

Get the time of ray appearing.

• point3 at (double t) const

Compute a point on the ray at a given parameter value.

## **Public Attributes**

- · point3 orig
- vec3 dir
- double tm

## 5.15.1 Detailed Description

A class representing a ray with an origin point and a direction vector. Heavily inspired by Peter Shirley https://raytracing.github.io/v3/books/RayTracingInOneWeekend.html.

Version

0.1

Date

2023-11-06

Copyright

Copyright (c) 2023

30 Class Documentation

## 5.15.2 Constructor & Destructor Documentation

### 5.15.2.1 ray()

Constructor to create a ray with an origin and direction.

### **Parameters**

origin	The origin point of the ray.
direction	The direction vector of the ray.
time	The time, when ray is existing

### **5.15.3 Member Function Documentation**

### 5.15.3.1 at()

```
point3 ray::at ( double t ) const
```

Compute a point on the ray at a given parameter value.

#### **Parameters**

t The parameter value that determines the point's position along the ray.

## Returns

The point on the ray at the specified parameter value.

### 5.15.3.2 direction()

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

Get the direction vector of the ray.

### Returns

The direction vector as a vec3.

### 5.15.3.3 origin()

```
point3 ray::origin ( ) const [inline]
```

Get the origin point of the ray.

Returns

The origin point as a point3.

## 5.15.3.4 time()

```
double ray::time ( ) const [inline]
```

Get the time of ray appearing.

Returns

The time of ray appearing.

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

- src/ray.hpp
- src/ray.cpp

## 5.16 sphere Class Reference

Represents a sphere.

```
#include <sphere.hpp>
```

Inheritance diagram for sphere:



## **Public Member Functions**

- sphere (point3 \_center, double \_radius, shared\_ptr< material > m)

  Constructor for the sphere class.
- bool hit (const ray &r, interval ray\_t, hit\_record &rec) const override

  Determines if a ray hits the sphere.

32 Class Documentation

## 5.16.1 Detailed Description

Represents a sphere.

This class inherits from the hittable class and defines a sphere with a center, radius, and material.

## 5.16.2 Constructor & Destructor Documentation

## 5.16.2.1 sphere()

Constructor for the sphere class.

### **Parameters**

_center	The center of the sphere.
_radius	The radius of the sphere.
m	A shared pointer to the material of the sphere.

### 5.16.3 Member Function Documentation

### 5.16.3.1 hit()

Determines if a ray hits the sphere.

### **Parameters**

r	The ray.
ray⊷	The valid interval for the ray parameter.
_t	
rec	The hit record to be filled if the ray hits the object.

### Returns

True if the ray hits the object, false otherwise.

Implements hittable.

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

- · src/sphere.hpp
- src/spehre.cpp

5.17 vec3 Class Reference 33

## 5.17 vec3 Class Reference

A class representing a 3D vector with x, y, and z components.

```
#include <vec3.hpp>
```

### **Public Member Functions**

• vec3 ()

Default constructor for a vec3, initializes all components to zero.

• vec3 (double e0, double e1, double e2)

Constructor to create a vec3 with specified components.

• double x () const

Get the x-component of the vec3.

• double y () const

Get the y-component of the vec3.

• double z () const

Get the z-component of the vec3.

• vec3 operator- () const

Unary negation operator, returns the negation of the vector.

• double operator[] (int i) const

Array subscript operator to access vector components.

• double & operator[] (int i)

Array subscript operator to access vector components (non-const version).

• vec3 & operator+= (const vec3 &v)

Compound addition assignment operator for vector addition.

vec3 & operator\*= (const double t)

Compound multiplication assignment operator for scalar multiplication.

vec3 & operator/= (const double t)

Compound division assignment operator for scalar division.

· double length () const

Calculate the length (magnitude) of the vector.

double length\_squared () const

Calculate the squared length (magnitude) of the vector.

- bool operator== (const vec3 &rhs) const
- bool near\_zero () const

Near\_zero bool check.

### **Static Public Member Functions**

• static vec3 random ()

creates a vec3 with random values from [0,1)

static vec3 random (double min, double max)

creates a vec3 with ranodom values in chosen range [min, max)

#### **Public Attributes**

• double e [3]

34 Class Documentation

## 5.17.1 Detailed Description

A class representing a 3D vector with x, y, and z components.

## 5.17.2 Constructor & Destructor Documentation

## 5.17.2.1 vec3()

Constructor to create a vec3 with specified components.

#### **Parameters**

e0	The x-component.
e1	The y-component.
e2	The z-component.

## 5.17.3 Member Function Documentation

## 5.17.3.1 length()

```
double vec3::length ( ) const [inline]
```

Calculate the length (magnitude) of the vector.

### Returns

The length of the vector as a double.

### 5.17.3.2 length\_squared()

```
double vec3::length_squared ( ) const [inline]
```

Calculate the squared length (magnitude) of the vector.

### Returns

The squared length of the vector as a double.

5.17 vec3 Class Reference 35

### 5.17.3.3 near\_zero()

```
bool vec3::near_zero ( ) const
```

Near\_zero bool check.

Returns

true

false

### 5.17.3.4 operator\*=()

Compound multiplication assignment operator for scalar multiplication.

Definition of an \*= operator.

### **Parameters**

t The scalar value to multiply this vector.

### Returns

A reference to this vector after multiplication.

#### **Parameters**

t

Returns

vec3&

### 5.17.3.5 operator+=()

Compound addition assignment operator for vector addition.

Definition of a += operator.

## **Parameters**

v The vector to be added to this vector.

36 Class Documentation

### Returns

A reference to this vector after addition.

### **Parameters**



### Returns

vec3&

### 5.17.3.6 operator-()

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

Unary negation operator, returns the negation of the vector.

### Returns

The negated vector as a vec3.

### 5.17.3.7 operator/=()

Compound division assignment operator for scalar division.

## **Parameters**

t The scalar value to divide this vector.

### Returns

A reference to this vector after division.

### 5.17.3.8 operator[]() [1/2]

Array subscript operator to access vector components (non-const version).

### **Parameters**

i The index (0 for x, 1 for y, 2 for z).

5.17 vec3 Class Reference 37

### Returns

A reference to the component at the specified index as a double.

### 5.17.3.9 operator[]() [2/2]

Array subscript operator to access vector components.

### **Parameters**

```
i The index (0 for x, 1 for y, 2 for z).
```

### Returns

The component at the specified index as a double (const version).

### 5.17.3.10 random() [1/2]

```
static vec3 vec3::random ( ) [inline], [static]
creates a vec3 with random values from [0,1)
```

## Returns

vec3

## 5.17.3.11 random() [2/2]

creates a vec3 with ranodom values in chosen range [min, max)

### **Parameters**



## Returns

vec3

38 Class Documentation

### 5.17.3.12 x()

```
double vec3::x ( ) const [inline]
```

Get the x-component of the vec3.

### Returns

The x-component as a double.

## 5.17.3.13 y()

```
double vec3::y ( ) const [inline]
```

Get the y-component of the vec3.

### Returns

The y-component as a double.

## 5.17.3.14 z()

```
double vec3::z ( ) const [inline]
```

Get the z-component of the vec3.

### Returns

The z-component as a double.

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

- src/vec3.hpp
- src/vec3.cpp

## 5.18 xy\_rect Class Reference

Inheritance diagram for xy\_rect:



### **Public Member Functions**

- xy\_rect (double \_x0, double \_x1, double \_y0, double \_y1, double \_k, shared\_ptr< material > mat)
   Constructs an xy rect with specified parameters.
- virtual bool hit (const ray &r, interval ray\_t, hit\_record &rec) const override

  Determines if a ray hits the rectangle.

## 5.18.1 Constructor & Destructor Documentation

### 5.18.1.1 xy\_rect()

Constructs an xy\_rect with specified parameters.

#### **Parameters**

_x0	Minimum x-coordinate.
_x1	Maximum x-coordinate.
_y0	Minimum y-coordinate.
_y1	Maximum y-coordinate.
_k	z-coordinate of the rectangle.
mat	Material of the rectangle.

## 5.18.2 Member Function Documentation

### 5.18.2.1 hit()

Determines if a ray hits the rectangle.

### **Parameters**

r	The ray.
ray⊷	The valid interval for the ray parameter.
_t	
rec	The hit record to be filled if the ray hits the object.

40 Class Documentation

### Returns

True if the ray hits the object, false otherwise.

Implements hittable.

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

- · src/rect.hpp
- src/rect.cpp

## 5.19 xz\_rect Class Reference

Inheritance diagram for xz\_rect:



### **Public Member Functions**

- xz\_rect (double \_x0, double \_x1, double \_z0, double \_z1, double \_k, shared\_ptr< material > mat)
   Constructs an xy\_rect with specified parameters.
- virtual bool hit (const ray &r, interval ray\_t, hit\_record &rec) const override
   Determines if a ray hits the rectangle.

### 5.19.1 Constructor & Destructor Documentation

## 5.19.1.1 xz\_rect()

Constructs an xy\_rect with specified parameters.

### **Parameters**

_x0	Minimum x-coordinate.
_x1	Maximum x-coordinate.
_z0	Minimum z-coordinate.
_z1	Maximum z-coordinate.
_k	y-coordinate of the rectangle.
mat	Material of the rectangle.

## 5.19.2 Member Function Documentation

### 5.19.2.1 hit()

Determines if a ray hits the rectangle.

### **Parameters**

r	The ray.
ray⊷	The valid interval for the ray parameter.
_t	
rec	The hit record to be filled if the ray hits the object.

### Returns

True if the ray hits the object, false otherwise.

Implements hittable.

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

- · src/rect.hpp
- · src/rect.cpp

## 5.20 yz\_rect Class Reference

Inheritance diagram for yz\_rect:



### **Public Member Functions**

- yz\_rect (double \_y0, double \_y1, double \_z0, double \_z1, double \_k, shared\_ptr< material > mat)

  Constructs an xy\_rect with specified parameters.
- virtual bool hit (const ray &r, interval ray\_t, hit\_record &rec) const override
   Determines if a ray hits the rectangle.

42 Class Documentation

## 5.20.1 Constructor & Destructor Documentation

### 5.20.1.1 yz\_rect()

Constructs an xy\_rect with specified parameters.

#### **Parameters**

_y0	Minimum y-coordinate.
_y1	Maximum y-coordinate.
_z0	Minimum z-coordinate.
_z1	Maximum z-coordinate.
_k	x-coordinate of the rectangle.
mat	Material of the rectangle.

## 5.20.2 Member Function Documentation

### 5.20.2.1 hit()

Determines if a ray hits the rectangle.

### **Parameters**

r	The ray.
ray⇔	The valid interval for the ray parameter.
_t	
rec	The hit record to be filled if the ray hits the object.

### Returns

True if the ray hits the object, false otherwise.

Implements hittable.

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

- · src/rect.hpp
- src/rect.cpp

# **Chapter 6**

# **File Documentation**

## 6.1 box.hpp

```
00001 #pragma once
00002
00003 #include "rect.hpp"
00004 #include "hittable_list.hpp"
00005
00010 class box : public hittable {
00011 public:
00012 box() {}
00013
00020
          box(const point3& p0, const point3& p1, shared_ptr<material> ptr);
00021
00033
          virtual bool hit(const ray& r, interval ray_t, hit_record& rec) const override;
00034
00035 public:
        point3 box_min;
00036
         point3 box_max;
00038
         hittable_list sides;
00039 };
```

## 6.2 src/camera.hpp File Reference

```
#include "utility.hpp"
#include "ray.hpp"
#include "vec3.hpp"
```

#### **Classes**

• class camera

Represents a movable camera in the world.

## 6.2.1 Detailed Description

The camera class represents a camera used for ray tracing. Heavily inspired by Peter Shirley https://raytracing.github.io/v3/books/RayTracingInOneWeekend.html.

**Author** 

```
Lauri Wilppu ( lauri.wilppu@gmail.com)
```

Version

0.1

Date

2023-11-07

Copyright

Copyright (c) 2023

## 6.3 camera.hpp

Go to the documentation of this file.

```
00001 #pragma once
00002 #include "utility.hpp"
00003 #include "ray.hpp"
00004 #include "vec3.hpp"
00005
00021 class camera {
00022 public:
00034
         camera(
00035
             point3 lookfrom,
              point3 lookat,
00036
00037
              vec3 vup,
             double vfov,
double aspect_ratio,
00038
00039
00040
             double aperture,
00041
             double focus_dist,
             double _time0, double _time1
00042
00043
00044
00045
00053
         ray get_ray(double s, double t) const;
00054
00059
         point3 getOrigin() const {
         return origin;
}
00060
00061
00062
         double getLensRadius() const {
00067
00068
           return lens_radius;
00069
00070
00071 private:
        point3 origin;
point3 lower_left_corner;
00072
00073
00074
         vec3 horizontal:
00075
         vec3 vertical;
         vec3 u, v, w;
double lens_radius;
00077
00078
         double time0, time1;
00079 };
```

6.4 hittable.hpp 45

## 6.4 hittable.hpp

```
00001 #pragma once
00002 #include "ray.hpp"
00003 #include "interval.hpp"
00004 class material;
00005
00010 class hit_record {
00011
      public:
00012
          point3 p;
00013
          vec3 normal:
00014
          shared_ptr<material> mat_ptr;
          double t;
00016
          double u, v;
00017
          bool front_face;
00026
          void set_face_normal(const ray& r, const vec3& outward_normal);
00027 };
00028
00029
00034 class hittable {
       public:
00035
00036
          virtual ~hittable() = default;
00037
00045
          virtual bool hit(const ray& r, interval ray_t, hit_record& rec) const = 0;
00046 };
```

## 6.5 hittable\_list.hpp

```
00001 #pragma once
00002 #include "hittable.hpp"
00003
00004 #include <memory>
00005 #include <vector>
00006
00007 using std::shared_ptr;
00008 using std::make_shared;
00009
00014 class hittable_list : public hittable {
00015
       public:
00016
          std::vector<shared_ptr<hittable> objects;
00021
          hittable_list() {}
00022
00027
          hittable_list(shared_ptr<hittable> object);
00028
00032
          void clear();
00038
          void add(shared_ptr<hittable> object);
00039
00040
00049
          bool hit (const ray& r, interval ray_t, hit_record& rec) const override;
00050 };
```

## 6.6 interval.hpp

```
00001 #pragma once
00002 #include <limits>
00003
00004 const double infinity = std::numeric_limits<double>::infinity();
00005
00006
00011 class interval {
00012
      public:
00013
         double min. max:
00019
          interval();
00020
00028
          interval(double _min, double _max);
00029
00037
         bool contains(double x) const;
00038
00046
          bool surrounds (double x) const;
00047
00054
         double clamp(double x) const;
00055
00056
          static const interval empty, universe;
00057 };
00058
00059 const static interval empty (+infinity, -infinity);
00060 const static interval universe(-infinity, +infinity);
```

## 6.7 src/material.hpp File Reference

The material class represents materials used for ray tracing. Heavily inspired by Peter Shirley https://raytracing.github.io/v3/books/RayTracingInOneWeekend.html.

```
#include "ray.hpp"
#include "hittable.hpp"
#include "utility.hpp"
```

### Classes

· class material

virtual class for all the materials.

class lambertian

Class representing Lambertian diffusion.

· class metal

Class representing metal material.

· class dielectric

Class representing dielectric material.

· class diffuse\_light

Represents a light source material.

## 6.7.1 Detailed Description

The material class represents materials used for ray tracing. Heavily inspired by Peter Shirley https://raytracing.github.io/v3/books/RayTracingInOneWeekend.html.

Author

Verneri Hakkarainen

Version

0.1

Date

2023-11-17

Copyright

Copyright (c) 2023

6.8 material.hpp 47

## 6.8 material.hpp

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "ray.hpp"
00004 #include "hittable.hpp"
00005 #include "utility.hpp"
00006
00024 class material {
        public:
00025
             virtual color emitted(double u, double v, const point3& p) const {
00026
00027
                  return color(0,0,0);
00029
              virtual bool scatter(
00030
                 const ray& r_in, const hit_record& rec, color& attenuation, ray& scattered
00031
              ) const = 0;
00032 };
00033
00038 class lambertian : public material {
        public:
00039
00040
00046
              lambertian(const color& a);
00047
00058
             virtual bool scatter(
00059
                 const ray& r_in, const hit_record& rec, color& attenuation, ray& scattered
00060
              ) const override;
00061
         private:
00062
00063
             color albedo;
00064 };
00065
00070 class metal : public material {
        public:
00071
00072
00079
              metal(const color& a, double f);
08000
00092
              virtual bool scatter(
00093
                  const ray& r_in, const hit_record& rec, color& attenuation, ray& scattered
00094
             ) const override;
00095
         private:
00096
00097
             color albedo:
00098
              double fuzz;
00099 };
00100
00105 class dielectric : public material {
         public:
00106
00107
00113
              dielectric (double index of reftraction);
00114
00125
              virtual bool scatter(
00126
                  const ray& r_in, const hit_record& rec, color& attenuation, ray& scattered
00127
              ) const override;
00128
        private:
00129
00130
             double ir; // index of refraction
00131
00140
              static double reflectance(double cosine, double ref_idx) {
00141
                 auto r0 = (1-ref_idx) / (1+ref_idx);
                  r0 = r0*r0;
00142
                  return r0 + (1-r0)*pow((1-cosine),5);
00143
00144
              }
00145 };
00146
00151 class diffuse_light : public material {
00152
        public:
00153
00158
             diffuse_light(color c);
00159
00160
              virtual bool scatter(
00161
                  const ray& r_in, const hit_record& rec, color& attenuation, ray& scattered
00162
              ) const override;
00163
00172
              virtual color emitted (double u, double v, const point 3 % p) const override;
00173
         private:
00174
00175
             shared_ptr<color> emit;
00176 };
```

## 6.9 moving\_sphere.hpp

00001 #pragma once

```
00002
00003 #include "hittable.hpp"
00004
00009 class moving_sphere : public hittable {
         public:
00010
00011
             moving_sphere() {}
00012
00024
00025
                point3 cen0, point3 cen1, double _time0, double _time1, double r, shared_ptr<material> m)
                  : center0(cen0), center1(cen1), time0(_time0), time1(_time1), radius(r), mat_ptr(m)
00026
00027
              { };
00028
00039
             virtual bool hit(const ray& r, interval ray_t, hit_record& rec) const override;
00040
00045
              point3 center(double time) const;
00046
         private:
00047
00048
             point3 center0, center1;
              double time0, time1;
00050
              double radius;
00051
              shared_ptr<material> mat_ptr;
00052 };
```

## 6.10 src/ray.cpp File Reference

Necessary class definitions of ray with an origin point and a direction vector.

```
#include "ray.hpp"
```

## 6.10.1 Detailed Description

Necessary class definitions of ray with an origin point and a direction vector.

**Author** 

```
Teodors Kerimovs ( teodors.kerimovs@gmail.com)
```

Version

0.1

Date

2023-12-10

Copyright

Copyright (c) 2023

## 6.11 src/ray.hpp File Reference

```
#include "vec3.hpp"
```

6.12 ray.hpp 49

### Classes

· class ray

A class representing a ray with an origin point and a direction vector. Heavily inspired by Peter Shirley https←://raytracing.github.io/v3/books/RayTracingInOneWeekend.html.

### 6.11.1 Detailed Description

**Author** 

Teodors Kerimovs (teodors.kerimovs@gmail.com)

## 6.12 ray.hpp

## Go to the documentation of this file.

```
00001 #pragma once
00015 #include "vec3.hpp"
00016
00017
00018 class ray {
00019 public:
00023
00024
            ray(const point3& origin, const vec3& direction, double time = 0.0)
00032
00033
                : orig(origin), dir(direction), tm(time)
00034
00035
00040
           point3 origin() const { return orig; }
00041
00046
           vec3 direction() const { return dir; }
00047
00052
           double time() const
                                    { return tm; }
00053
00059
           point3 at(double t) const;
00060
00061 public:
           point3 orig; // The origin point of the ray.
vec3 dir; // The direction vector of the ray.
double tm; // Time of vector appearing.
00062
00063
00064
00065 };
```

## 6.13 rect.hpp

```
00001 #pragma once
00003 #include "utility.hpp"
00004 #include "hittable.hpp"
00005
00006
00011 class xy_rect : public hittable {
00012 public:
00013
         xy_rect() {}
00014
00024
         xy_rect(double _x0, double _x1, double _y0, double _y1, double _k,
00025
            shared_ptr<material> mat);
00026
00036
          virtual bool hit (const ray& r, interval ray_t, hit_record& rec) const override;
00037
00038
       private:
00039
          shared_ptr<material> mp;
00040
          double x0, x1, y0, y1, k;
00041 };
00042
00048 class xz_rect : public hittable {
00049 public:
00050
          xz_rect() {}
00051
          xz_rect(double _x0, double _x1, double _z0, double _z1, double _k,
00061
00062
            shared_ptr<material> mat);
```

```
virtual bool hit(const ray& r, interval ray_t, hit_record& rec) const override;
00074
00075
00076
          shared_ptr<material> mp;
00077
          double x0, x1, z0, z1, k;
00078 };
00079
00085 class yz_rect : public hittable {
00086 public:
00087
        yz_rect() {}
00088
        yz_rect(double _y0, double _y1, double _z0, double _z1, double _k,
    shared_ptr<material> mat);
00098
00099
00100
00110
        virtual bool hit(const ray& r, interval ray_t, hit_record& rec) const override;
00111
        private:
00112
          shared_ptr<material> mp;
00113
          double y0, y1, z0, z1, k;
00114 };
```

## 6.14 render.hpp

```
00001 #pragma once
00002
00003 #include "vec3.hpp"
00004 #include "interval.hpp"
00005 #include "fstream"

00006 #include "ray.hpp"

00007 #include "hittable_list.hpp"
00008 #include "camera.hpp"
00009 #include "material.hpp"
00010 #include "interval.hpp"
00011 #include "fstream"
00012 #include <iostream>
00013 #include <thread>
00014 #include <mutex>
00015 #include <chrono>
00016
00022 struct PixelInfo {
        int x;
int y;
00023
00024
00025
           color pixel_color;
00026 };
00036 color ray_color(const ray& r, const color& background, const hittable& world, int depth, bool
      skylight);
00037
00038
00049 void render(int image_width, int image_height, int samples_per_pixel, int max_depth, camera cam,
      hittable_list& world, color background, bool skylight);
```

## 6.15 sphere.hpp

```
00001 #pragma once
00002 #include "hittable.hpp"
00003 #include "vec3.hpp"
00004 #include "interval.hpp"
00005
00011 class sphere : public hittable {
00012 public:
           sphere(point3 _center, double _radius, shared_ptr<material> m);
00019
00030
           bool hit(const ray& r, interval ray_t, hit_record& rec) const override;
00031
00032
        private:
00033
           point3 center;
00034
           double radius;
00035
           shared_ptr<material> mat_ptr;
00036 };
```

## 6.16 utility.hpp

```
00001 #pragma once
00002
00003
```

```
00004 #include <cmath>
00005 #include <cstdlib>
00006 #include <limits>
00007 #include <memory>
80000
00009
00010 // Usings
00011
00012 using std::shared_ptr;
00013 using std::make_shared;
00014 using std::sqrt;
00015
00016 // Constants
00017 const double pi = 3.1415926535897932385;
00018
00019 // Utility Functions
00020
00026 double degrees_to_radians(double degrees);
00035 double clamp(double x, double min, double max);
00036
00041 double random_double();
00042
00049 double random_double(double min, double max);
00050
00057 int random_int(int min, int max);
```

## 6.17 src/vec3.cpp File Reference

Cpp files with definitions of vec3 members and non-members.

```
#include "vec3.hpp"
```

### **Functions**

std::ostream & operator<< (std::ostream &out, const vec3 &v)</li>

Overloaded output stream operator to print a vec3.

• vec3 operator+ (const vec3 &u, const vec3 &v)

Overloaded addition operator for vector addition.

• vec3 operator- (const vec3 &u, const vec3 &v)

Overloaded subtraction operator for vector subtraction.

• vec3 operator\* (const vec3 &u, const vec3 &v)

Overloaded multiplication operator for component-wise vector multiplication.

• vec3 operator\* (double t, const vec3 &v)

Overloaded multiplication operator for scalar multiplication.

vec3 operator\* (const vec3 &v, double t)

Overloaded multiplication operator for scalar multiplication (reversed order).

• vec3 operator/ (vec3 v, double t)

Overloaded division operator for scalar division.

double dot (const vec3 &u, const vec3 &v)

Calculate the dot product of two vectors.

• vec3 cross (const vec3 &u, const vec3 &v)

Calculate the cross product of two vectors.

vec3 unit\_vector (vec3 v)

Calculate the unit vector (normalized vector) of a given vector.

vec3 refract (const vec3 &uv, const vec3 &n, double etai\_over\_etat)

Computes the refraction direction for a given incident vector (uv), surface normal (n), and the ratio of refractive indices (etai\_over\_etat).

vec3 random\_in\_unit\_sphere ()

chooses a random point inside a unit sphere. Used in reflections of different materials.

vec3 random\_unit\_vector ()

Chooses a random point on the surface unit circle cirlce.

• vec3 random\_in\_hemisphere (const vec3 &normal)

TODO different kind of diffuse.

- vec3 reflect (const vec3 &v, const vec3 &n)
- vec3 random\_in\_unit\_disk ()

Generates a random point in the unit disk.

## 6.17.1 Detailed Description

Cpp files with definitions of vec3 members and non-members.

**Author** 

```
Teodors Kerimovs ( teodors.kerimovs@gmail.com)
```

Version

0.1

Date

2023-12-10

Copyright

Copyright (c) 2023

## 6.17.2 Function Documentation

## 6.17.2.1 cross()

Calculate the cross product of two vectors.

### **Parameters**

и	The first vector.
V	The second vector.

### Returns

The cross product of the two vectors as a vec3.

### 6.17.2.2 dot()

Calculate the dot product of two vectors.

### **Parameters**

и	The first vector.
V	The second vector.

## Returns

The dot product of the two vectors as a double.

## 6.17.2.3 operator\*() [1/3]

Overloaded multiplication operator for component-wise vector multiplication.

## **Parameters**

и	The first vector.
V	The second vector.

### Returns

The result of multiplying the two vectors component-wise as a vec3.

## 6.17.2.4 operator\*() [2/3]

Overloaded multiplication operator for scalar multiplication (reversed order).

### **Parameters**

V	The vector to be multiplied.
t	The scalar value.

### Returns

The result of multiplying the vector by the scalar as a vec3.

## 6.17.2.5 operator\*() [3/3]

```
vec3 operator* ( \label{eq:const_vec3} \mbox{double } t, \\ \mbox{const vec3 & $v$ )}
```

Overloaded multiplication operator for scalar multiplication.

### **Parameters**

t	The scalar value.
V	The vector to be multiplied.

## Returns

The result of multiplying the vector by the scalar as a vec3.

### 6.17.2.6 operator+()

Overloaded addition operator for vector addition.

### **Parameters**

и	The first vector.
V	The second vector.

## Returns

The result of adding the two vectors as a vec3.

## 6.17.2.7 operator-()

Overloaded subtraction operator for vector subtraction.

### **Parameters**

и	The first vector.
V	The second vector.

#### Returns

The result of subtracting the second vector from the first as a vec3.

## 6.17.2.8 operator/()

```
{\tt vec3} operator/ ( {\tt vec3} v, double t )
```

Overloaded division operator for scalar division.

### **Parameters**

V	The vector to be divided.
t	The scalar value.

### Returns

The result of dividing the vector by the scalar as a vec3.

## 6.17.2.9 operator<<()

```
std::ostream & operator<< (  std::ostream \& out, \\ const vec3 \& v )
```

Overloaded output stream operator to print a vec3.

#### **Parameters**

out	The output stream.
V	The vec3 to be printed.

#### Returns

The output stream after printing the vec3.

## 6.17.2.10 random\_in\_hemisphere()

TODO different kind of diffuse.

#### **Parameters**

```
normal
```

#### Returns

```
** vec3
```

## 6.17.2.11 random\_in\_unit\_disk()

```
vec3 random_in_unit_disk ( )
```

Generates a random point in the unit disk.

This function continues generating points until it finds one within the unit disk.

Returns

A random point in the unit disk.

### 6.17.2.12 random\_in\_unit\_sphere()

```
vec3 random_in_unit_sphere ( )
```

chooses a random point inside a unit sphere. Used in reflections of different materials.

Returns

vec3

### 6.17.2.13 random\_unit\_vector()

```
vec3 random_unit_vector ( )
```

Chooses a random point on the surface unit circle cirlce.

Returns

vec3

### 6.17.2.14 refract()

Computes the refraction direction for a given incident vector (uv), surface normal (n), and the ratio of refractive indices (etai\_over\_etat).

This function ensures the cosine of the angle between the incident vector and the normal is within [-1, 1], and then computes the refracted ray using Snell's Law.

#### **Parameters**

uv	
n	
etai_over_etat	

### Returns

vec3

## 6.17.2.15 unit\_vector()

Calculate the unit vector (normalized vector) of a given vector.

#### **Parameters**

v The vector to be normalized.

#### Returns

The unit vector (normalized vector) of the input vector as a vec3.

## 6.18 src/vec3.hpp File Reference

A C++ header file defining a 3D vector class and related utility functions. Heavily inspired by Peter Shirley https://raytracing.github.io/v3/books/RayTracingInOneWeekend.html.

```
#include <cmath>
#include <iostream>
#include "utility.hpp"
```

### Classes

• class vec3

A class representing a 3D vector with x, y, and z components.

## **Typedefs**

- using **point3** = vec3
- using color = vec3

#### **Functions**

std::ostream & operator<< (std::ostream &out, const vec3 &v)</li>

Overloaded output stream operator to print a vec3.

vec3 operator+ (const vec3 &u, const vec3 &v)

Overloaded addition operator for vector addition.

vec3 operator- (const vec3 &u, const vec3 &v)

Overloaded subtraction operator for vector subtraction.

• vec3 operator\* (const vec3 &u, const vec3 &v)

Overloaded multiplication operator for component-wise vector multiplication.

vec3 operator\* (double t, const vec3 &v)

Overloaded multiplication operator for scalar multiplication.

vec3 operator\* (const vec3 &v, double t)

Overloaded multiplication operator for scalar multiplication (reversed order).

• vec3 operator/ (vec3 v, double t)

Overloaded division operator for scalar division.

double dot (const vec3 &u, const vec3 &v)

Calculate the dot product of two vectors.

• vec3 cross (const vec3 &u, const vec3 &v)

Calculate the cross product of two vectors.

vec3 unit vector (vec3 v)

Calculate the unit vector (normalized vector) of a given vector.

vec3 random\_in\_unit\_sphere ()

chooses a random point inside a unit sphere. Used in reflections of different materials.

• vec3 random unit vector ()

Chooses a random point on the surface unit circle cirlce.

· vec3 random\_in\_hemisphere (const vec3 &normal)

TODO different kind of diffuse.

- vec3 reflect (const vec3 &v, const vec3 &n)
- vec3 random\_in\_unit\_disk ()

Generates a random point in the unit disk.

vec3 refract (const vec3 &uv, const vec3 &n, double etai\_over\_etat)

Computes the refraction direction for a given incident vector (uv), surface normal (n), and the ratio of refractive indices (etai\_over\_etat).

### 6.18.1 Detailed Description

A C++ header file defining a 3D vector class and related utility functions. Heavily inspired by Peter Shirley https://raytracing.github.io/v3/books/RayTracingInOneWeekend.html.

**Author** 

```
Teodors Kerimovs ( teodors.kerimovs@gmail.com)
```

Version

0.1

Date

2023-11-06

Copyright

Copyright (c) 2023

## 6.18.2 Function Documentation

### 6.18.2.1 cross()

Calculate the cross product of two vectors.

### **Parameters**

и	The first vector.
V	The second vector.

### Returns

The cross product of the two vectors as a vec3.

### 6.18.2.2 dot()

Calculate the dot product of two vectors.

### **Parameters**

и	The first vector.
V	The second vector.

## Returns

The dot product of the two vectors as a double.

### 6.18.2.3 operator\*() [1/3]

Overloaded multiplication operator for component-wise vector multiplication.

### **Parameters**

и	The first vector.
V	The second vector.

### Returns

The result of multiplying the two vectors component-wise as a vec3.

### 6.18.2.4 operator\*() [2/3]

Overloaded multiplication operator for scalar multiplication (reversed order).

### **Parameters**

V	The vector to be multiplied.
t	The scalar value.

### Returns

The result of multiplying the vector by the scalar as a vec3.

### 6.18.2.5 operator\*() [3/3]

```
vec3 operator* ( \label{eq:const_vec3} \mbox{double } t, \\ \mbox{const vec3 & $v$ )}
```

Overloaded multiplication operator for scalar multiplication.

### **Parameters**

t	The scalar value.
V	The vector to be multiplied.

### Returns

The result of multiplying the vector by the scalar as a vec3.

### 6.18.2.6 operator+()

Overloaded addition operator for vector addition.

### **Parameters**

и	The first vector.	
V	The second vector.	

#### Returns

The result of adding the two vectors as a vec3.

## 6.18.2.7 operator-()

Overloaded subtraction operator for vector subtraction.

### **Parameters**

и	The first vector.	
V	The second vector.	

### Returns

The result of subtracting the second vector from the first as a vec3.

## 6.18.2.8 operator/()

Overloaded division operator for scalar division.

## **Parameters**

V	The vector to be divided.
t	The scalar value.

### Returns

The result of dividing the vector by the scalar as a vec3.

### 6.18.2.9 operator <<()

```
std::ostream & operator<< (  \mbox{std::ostream \& out,}   \mbox{const vec3 \& $v$ )}
```

Overloaded output stream operator to print a vec3.

### **Parameters**

out	The output stream.
V	The vec3 to be printed.

### Returns

The output stream after printing the vec3.

### 6.18.2.10 random\_in\_hemisphere()

TODO different kind of diffuse.

### **Parameters**

normal

### Returns

\*\* vec3

## 6.18.2.11 random\_in\_unit\_disk()

```
vec3 random_in_unit_disk ( )
```

Generates a random point in the unit disk.

This function continues generating points until it finds one within the unit disk.

## Returns

A random point in the unit disk.

## 6.18.2.12 random\_in\_unit\_sphere()

```
vec3 random_in_unit_sphere ( )
```

chooses a random point inside a unit sphere. Used in reflections of different materials.

### Returns

vec3

### 6.18.2.13 random\_unit\_vector()

```
vec3 random_unit_vector ( )
```

Chooses a random point on the surface unit circle cirlce.

### Returns

vec3

## 6.18.2.14 refract()

Computes the refraction direction for a given incident vector (uv), surface normal (n), and the ratio of refractive indices (etai\_over\_etat).

This function ensures the cosine of the angle between the incident vector and the normal is within [-1, 1], and then computes the refracted ray using Snell's Law.

### **Parameters**

uv	The incident vector.
n	The surface normal.
etai_over_etat	The ratio of refractive indices.

### Returns

The refracted ray direction.

This function ensures the cosine of the angle between the incident vector and the normal is within [-1, 1], and then computes the refracted ray using Snell's Law.

### **Parameters**

uv	
n	
etai_over_etat	

### Returns

vec3

## 6.18.2.15 unit\_vector()

```
vec3 unit_vector (
     vec3 v )
```

Calculate the unit vector (normalized vector) of a given vector.

#### **Parameters**

v The vector to be normalized.

#### Returns

The unit vector (normalized vector) of the input vector as a vec3.

## 6.19 vec3.hpp

#### Go to the documentation of this file.

```
00001 #pragma once
00014 #include <cmath>
00015 #include <iostream>
00016 #include "utility.hpp"
00017
00018 using std::sqrt;
00023 class vec3 {
00024
         public:
00028
              vec3() : e{0,0,0} {}
00035
              vec3(double e0, double e1, double e2) : e{e0, e1, e2} {}
00040
              double x() const { return e[0];
00045
              double y() const { return e[1];
00050
              double z() const { return e[2]; }
00051
00056
              vec3 operator-() const { return vec3(-e[0], -e[1], -e[2]); }
00057
00063
              double operator[](int i) const { return e[i]; }
00064
00070
              double& operator[](int i) { return e[i]; }
00071
00072
00078
              vec3& operator+=(const vec3 &v);
00079
00085
              vec3& operator*=(const double t);
00086
00092
              vec3& operator/=(const double t) {
00093
                 return *this *= 1/t;
00094
00095
00100
              double length() const {
00101
                 return sqrt(length_squared());
00102
00103
00108
              double length_squared() const {
00109
                 return e[0]*e[0] + e[1]*e[1] + e[2]*e[2];
00110
00111
00117
          inline static vec3 random() {
00118
          return vec3(random_double(), random_double());
00119
00120
00128
          inline static vec3 random(double min, double max) {
           return vec3(random_double(min, max), random_double(min, max), random_double(min, max));
00129
00130
00131
00132
          inline bool operator == (const vec3& rhs) const {
00133
            return e[0] == rhs.e[0] && e[1] == rhs.e[1] && e[2] == rhs.e[2];
00134
00135
00136
          bool near_zero() const;
00137
00138
          public:
00139
              double e[3]; // Components of the vec3.
00140 };
00141
00142 // Type aliases for vec3
00143 using point3 = vec3; // 3D point
00144 using color = vec3; // RGB color
                             // RGB color
00145
00146
00147
00148 // vec3 Utility Functions
00155 std::ostream& operator«(std::ostream &out, const vec3 &v);
00156
00163 vec3 operator+(const vec3 &u, const vec3 &v);
00164
```

6.19 vec3.hpp 67

```
00171 vec3 operator-(const vec3 &u, const vec3 &v);
00172
00179 vec3 operator*(const vec3 &u, const vec3 &v);
00180
00187 vec3 operator*(double t, const vec3 &v);
00188
00195 vec3 operator*(const vec3 &v, double t);
00196
00203 vec3 operator/(vec3 v, double t);
00204
00211 double dot(const vec3 &u, const vec3 &v);
00218 vec3 cross(const vec3 &u, const vec3 &v);
00219
00225 vec3 unit_vector(vec3 v);
00226
00227
00233 vec3 random_in_unit_sphere();
00234
00240 vec3 random_unit_vector();
00241
00248 vec3 random_in_hemisphere(const vec3& normal);
00249
00250 vec3 reflect(const vec3 \mbox{v} v, const vec3 \mbox{n});
00251
00259 vec3 random_in_unit_disk();
00260
00273 vec3 refract(const vec3& uv, const vec3& n, double etai_over_etat);
```

# Index

```
add
                                                                hittable, 18
     hittable_list, 19
                                                                hittable_list, 20
at
                                                                moving_sphere, 28
                                                                sphere, 32
     ray, 30
                                                                xy_rect, 39
box, 9
                                                                xz_rect, 41
     box, 9
                                                                yz_rect, 42
     hit, 10
                                                           hit_record, 15
                                                                front_face, 16
camera, 10
                                                                mat ptr, 16
     camera, 11
                                                                normal, 16
     get_ray, 11
                                                                p, 17
     getLensRadius, 12
                                                                set_face_normal, 16
     getOrigin, 12
                                                                t, 17
clamp
                                                                v, 17
     interval, 21
                                                           hittable, 17
contains
                                                                hit, 18
     interval, 21
                                                           hittable_list, 18
cross
                                                                add, 19
     vec3.cpp, 52
                                                                hit, 20
     vec3.hpp, 60
                                                                hittable_list, 19
                                                                objects, 20
dielectric, 12
     dielectric, 13
                                                           interval, 20
     scatter, 13
                                                                clamp, 21
diffuse light, 14
                                                                contains, 21
     diffuse light, 14
                                                                interval, 21
     emitted, 15
                                                                max, 22
     scatter, 15
                                                                surrounds, 22
direction
                                                           Is, 23
     ray, 30
dot
                                                           lambertian, 23
     vec3.cpp, 52
                                                                lambertian, 23
     vec3.hpp, 60
                                                                scatter, 24
                                                           length
emitted
                                                                vec3, 34
     diffuse_light, 15
                                                           length_squared
     material, 25
                                                                vec3, 34
front face
                                                           mat_ptr
     hit_record, 16
                                                                hit_record, 16
                                                           material, 24
get ray
                                                                emitted, 25
     camera, 11
                                                                scatter, 25
getLensRadius
                                                           max
     camera, 12
                                                                interval, 22
getOrigin
                                                           metal, 25
     camera, 12
                                                                metal, 26
                                                                scatter, 26
hit
                                                           moving_sphere, 27
     box, 10
```

70 INDEX

111.00	
hit, 28	ray, 30
moving_sphere, 27	time, 31
near zero	refract
vec3, 34	vec3.cpp, 57
normal	vec3.hpp, 64
hit record, 16	scatter
m_1000rd, 10	dielectric, 13
objects	diffuse_light, 15
hittable_list, 20	lambertian, 24
operator<<	material, 25
vec3.cpp, 55	metal, 26
vec3.hpp, 62	set face normal
operator+	hit_record, 16
vec3.cpp, 54	Source content, 1
vec3.hpp, 61	sphere, 31
operator+=	hit, 32
vec3, 35	sphere, 32
operator-	src/box.hpp, 43
vec3, 36	src/camera.hpp, 43, 44
vec3.cpp, 54	src/hittable.hpp, 45
vec3.hpp, 62	src/hittable_list.hpp, 45
operator/	src/interval.hpp, 45
vec3.cpp, 55	src/material.hpp, 46, 47
vec3.hpp, 62	src/moving_sphere.hpp, 47
operator/=	src/ray.cpp, 48
vec3, 36	src/ray.hpp, 48, 49
operator[]	src/rect.hpp, 49
vec3, 36, 37	src/render.hpp, 50
operator*	src/sphere.hpp, 50
vec3.cpp, 53, 54	src/utility.hpp, 50
vec3.hpp, 60, 61	src/vec3.cpp, 51
operator*=	src/vec3.hpp, 58, 66
vec3, 35	surrounds
origin	interval, 22
ray, 30	
p	t hit record 17
hit record, 17	hit_record, 17 time
PixelInfo, 28	•••••
1 1/011110, 20	ray, 31
random	unit vector
vec3, 37	vec3.cpp, 58
random_in_hemisphere	vec3.hpp, 64
vec3.cpp, 55	117
vec3.hpp, 63	V
random_in_unit_disk	hit_record, 17
vec3.cpp, 57	vec3, 33
vec3.hpp, 63	length, 34
random_in_unit_sphere	length_squared, 34
vec3.cpp, 57	near_zero, 34
vec3.hpp, 63	operator+=, 35
random_unit_vector	operator-, 36
vec3.cpp, 57	operator/=, 36
vec3.hpp, 63	operator[], 36, 37
ray, 29	operator*=, 35
at, 30	random, 37
direction, 30	vec3, 34
origin, 30	x, 37

INDEX 71

```
y, <mark>38</mark>
     z, 38
vec3.cpp
    cross, 52
     dot, 52
     operator<<, 55
     operator+, 54
     operator-, 54
     operator/, 55
     operator*, 53, 54
     random_in_hemisphere, 55
     random_in_unit_disk, 57
     random_in_unit_sphere, 57
     random_unit_vector, 57
     refract, 57
     unit_vector, 58
vec3.hpp
    cross, 60
    dot, 60
     operator << , 62
     operator+, 61
     operator-, 62
     operator/, 62
     operator*, 60, 61
     random_in_hemisphere, 63
     random_in_unit_disk, 63
     random_in_unit_sphere, 63
     random_unit_vector, 63
     refract, 64
     unit_vector, 64
    vec3, 37
xy_rect, 38
     hit, 39
     xy_rect, 39
xz_rect, 40
     hit, 41
     xz_rect, 40
у
     vec3, 38
yz_rect, 41
     hit, 42
     yz_rect, 42
```

vec3, 38