

Voxel World

0.02

Generated by Doxygen 1.8.14

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	Camera Class Reference	5
3.1.1	Constructor & Destructor Documentation	6
3.1.1.1	Camera() [1/2]	6
3.1.1.2	Camera() [2/2]	6
3.1.1.3	~Camera()	6
3.1.2	Member Function Documentation	6
3.1.2.1	getForward()	7
3.1.2.2	getFov()	7
3.1.2.3	getPerspectiveMatrix()	7
3.1.2.4	getPosition()	7
3.1.2.5	getSide()	7
3.1.2.6	getUp()	7
3.1.2.7	getViewMatrix()	7
3.1.2.8	incline()	7
3.1.2.9	move()	8
3.1.2.10	resetCamera()	8
3.1.2.11	resetView()	8

3.1.2.12	setLookDirection()	8
3.1.2.13	setPosition()	8
3.1.2.14	turnH()	8
3.1.2.15	turnV()	8
3.1.3	Member Data Documentation	9
3.1.3.1	forward	9
3.1.3.2	fov	9
3.1.3.3	height	9
3.1.3.4	orig_forward	9
3.1.3.5	orig_position	9
3.1.3.6	orig_side	9
3.1.3.7	orig_up	10
3.1.3.8	position	10
3.1.3.9	side	10
3.1.3.10	up	10
3.1.3.11	width	10
3.1.3.12	zFar	10
3.1.3.13	zNear	11
3.2	Chunk Class Reference	11
3.2.1	Constructor & Destructor Documentation	11
3.2.1.1	Chunk() [1/3]	12
3.2.1.2	Chunk() [2/3]	12
3.2.1.3	Chunk() [3/3]	12
3.2.1.4	~Chunk()	12
3.2.2	Member Function Documentation	12
3.2.2.1	check_neighbour()	12
3.2.2.2	create_cubes()	12
3.2.2.3	operator()()	13
3.2.2.4	send_render_data()	13
3.2.2.5	update()	13

3.2.2.6	update_visible_faces()	13
3.2.3	Member Data Documentation	13
3.2.3.1	chunk_cubes	13
3.2.3.2	cubes_info	13
3.2.3.3	position	13
3.2.3.4	render_data	14
3.2.3.5	world	14
3.3	Chunk_Holder Class Reference	14
3.3.1	Constructor & Destructor Documentation	14
3.3.1.1	Chunk_Holder() [1/2]	14
3.3.1.2	Chunk_Holder() [2/2]	15
3.3.1.3	~Chunk_Holder()	15
3.3.2	Member Function Documentation	15
3.3.2.1	operator>()	15
3.3.2.2	shift()	15
3.3.3	Member Data Documentation	15
3.3.3.1	chunkBox	15
3.3.3.2	world	16
3.4	cirArray< T > Class Template Reference	16
3.4.1	Detailed Description	16
3.4.2	Constructor & Destructor Documentation	16
3.4.2.1	cirArray() [1/2]	16
3.4.2.2	cirArray() [2/2]	17
3.4.3	Member Function Documentation	17
3.4.3.1	operator=()	17
3.4.3.2	operator[]()	17
3.4.3.3	shift()	17
3.4.3.4	size()	17
3.4.4	Member Data Documentation	17
3.4.4.1	array	18

3.4.4.2	start	18
3.5	Cube Class Reference	18
3.5.1	Constructor & Destructor Documentation	19
3.5.1.1	Cube() [1/3]	19
3.5.1.2	Cube() [2/3]	19
3.5.1.3	Cube() [3/3]	19
3.5.1.4	~Cube()	19
3.5.2	Member Function Documentation	19
3.5.2.1	cleanup()	19
3.5.2.2	getMesh()	19
3.5.2.3	initialize()	20
3.5.2.4	update()	20
3.5.3	Member Data Documentation	20
3.5.3.1	cube_type	20
3.5.3.2	meshes	20
3.5.3.3	position	20
3.5.3.4	textures	20
3.5.3.5	transparent	20
3.6	Light Struct Reference	21
3.6.1	Member Data Documentation	21
3.6.1.1	color	21
3.6.1.2	intensity	21
3.6.1.3	position	21
3.7	Mesh Struct Reference	21
3.7.1	Constructor & Destructor Documentation	22
3.7.1.1	~Mesh()	22
3.7.2	Member Data Documentation	22
3.7.2.1	indices	22
3.7.2.2	normals	22
3.7.2.3	uvs	22

3.7.2.4	vertices	23
3.8	Object_3D Class Reference	23
3.8.1	Constructor & Destructor Documentation	23
3.8.1.1	Object_3D()	23
3.8.2	Member Function Documentation	24
3.8.2.1	set_instance_data()	24
3.8.3	Member Data Documentation	24
3.8.3.1	layouts	24
3.8.3.2	mesh_indices	24
3.8.3.3	render_instances	24
3.8.3.4	types	24
3.8.3.5	VAO	25
3.8.3.6	VBOs	25
3.9	Renderer Class Reference	25
3.9.1	Constructor & Destructor Documentation	26
3.9.1.1	Renderer() [1/2]	26
3.9.1.2	Renderer() [2/2]	26
3.9.1.3	~Renderer()	26
3.9.2	Member Function Documentation	26
3.9.2.1	add_data()	26
3.9.2.2	add_Shader()	27
3.9.2.3	change_active_program()	27
3.9.2.4	clear()	27
3.9.2.5	find_shader()	27
3.9.2.6	make_program()	27
3.9.2.7	multi_render()	27
3.9.2.8	render()	28
3.9.2.9	set_camera()	28
3.9.2.10	update()	28
3.9.3	Member Data Documentation	28

3.9.3.1	busy_queue	28
3.9.3.2	cam	28
3.9.3.3	current_program	28
3.9.3.4	fragment_shaders	29
3.9.3.5	render_queue	29
3.9.3.6	shading_programs	29
3.9.3.7	tessellation_shaders	29
3.9.3.8	vertex_shaders	29
3.10	Shader Class Reference	29
3.10.1	Constructor & Destructor Documentation	30
3.10.1.1	Shader() [1/2]	30
3.10.1.2	Shader() [2/2]	30
3.10.1.3	~Shader()	30
3.10.2	Member Function Documentation	31
3.10.2.1	clear()	31
3.10.2.2	load_from_file()	31
3.10.3	Member Data Documentation	31
3.10.3.1	fileName	31
3.10.3.2	shaderID	31
3.10.3.3	type	32
3.11	Texture Class Reference	32
3.11.1	Constructor & Destructor Documentation	32
3.11.1.1	Texture()	32
3.11.1.2	~Texture()	33
3.11.2	Member Function Documentation	33
3.11.2.1	clear()	33
3.11.2.2	load_to_GPU()	33
3.11.3	Member Data Documentation	33
3.11.3.1	height	33
3.11.3.2	target	34

3.11.3.3	texture	34
3.11.3.4	textureID	34
3.11.3.5	width	34
3.12	World Class Reference	34
3.12.1	Constructor & Destructor Documentation	35
3.12.1.1	World()	35
3.12.1.2	~World()	35
3.12.2	Member Function Documentation	35
3.12.2.1	center_frame()	35
3.12.2.2	operator()()	35
3.12.2.3	send_render_data()	36
3.12.3	Member Data Documentation	36
3.12.3.1	h_radius	36
3.12.3.2	loaded_chunks	36
3.12.3.3	loaded_lights	36
3.12.3.4	origin	36
3.12.3.5	v_radius	36
4	File Documentation	37
4.1	Cube.cpp File Reference	37
4.1.1	Variable Documentation	37
4.1.1.1	obj_source_files	37
4.1.1.2	texture_source_files	37
4.2	Cube.hpp File Reference	37
4.2.1	Enumeration Type Documentation	38
4.2.1.1	CubeID	38
4.2.2	Variable Documentation	38
4.2.2.1	cube_types	38
4.3	Helpers/cout-definitions.cpp File Reference	38
4.3.1	Detailed Description	39
4.3.2	Function Documentation	39

4.3.2.1	<code>operator<<()</code> [1/4]	39
4.3.2.2	<code>operator<<()</code> [2/4]	39
4.3.2.3	<code>operator<<()</code> [3/4]	39
4.3.2.4	<code>operator<<()</code> [4/4]	39
4.4	Helpers/cout-definitions.hpp File Reference	40
4.4.1	Detailed Description	40
4.4.2	Function Documentation	40
4.4.2.1	<code>operator<<()</code> [1/4]	40
4.4.2.2	<code>operator<<()</code> [2/4]	41
4.4.2.3	<code>operator<<()</code> [3/4]	41
4.4.2.4	<code>operator<<()</code> [4/4]	41
4.5	Helpers/system-libraries.hpp File Reference	41
4.5.1	Detailed Description	42
4.5.2	Macro Definition Documentation	42
4.5.2.1	<code>GLEW_DYNAMIC</code>	42
4.6	Helpers/tools.cpp File Reference	42
4.6.1	Detailed Description	43
4.6.2	Function Documentation	43
4.6.2.1	<code>fade()</code>	43
4.6.2.2	<code>length()</code>	43
4.6.2.3	<code>noise_2D()</code>	43
4.6.2.4	<code>perlin_noise()</code>	43
4.6.2.5	<code>surflet()</code>	44
4.6.2.6	<code>vec_field_init()</code>	44
4.6.3	Variable Documentation	44
4.6.3.1	<code>mask</code>	44
4.6.3.2	<code>perm</code>	44
4.6.3.3	<code>size</code>	44
4.6.3.4	<code>vec_field_x</code>	44
4.6.3.5	<code>vec_field_y</code>	45

4.7	Helpers/tools.hpp File Reference	45
4.7.1	Detailed Description	45
4.7.2	Function Documentation	45
4.7.2.1	noise_2D()	45
4.7.2.2	vec_field_init()	46
4.8	Helpers/wavefront-loader.cpp File Reference	46
4.8.1	Detailed Description	46
4.8.2	Function Documentation	46
4.8.2.1	load_obj()	46
4.9	Helpers/wavefront-loader.hpp File Reference	47
4.9.1	Detailed Description	47
4.9.2	Function Documentation	47
4.9.2.1	load_obj()	47
4.10	main.cpp File Reference	47
4.10.1	Typedef Documentation	48
4.10.1.1	frame_duration	48
4.10.1.2	world_duration	48
4.10.2	Function Documentation	48
4.10.2.1	main()	48
4.10.2.2	render_loop()	48
4.10.2.3	update_loop()	49
4.11	Rendering/Camera/Camera.cpp File Reference	49
4.11.1	Detailed Description	49
4.12	Rendering/Camera/Camera.hpp File Reference	49
4.12.1	Detailed Description	49
4.13	Rendering/OpenGL-Wrappers.cpp File Reference	50
4.13.1	Detailed Description	50
4.13.2	Macro Definition Documentation	50
4.13.2.1	STB_IMAGE_IMPLEMENTATION	50
4.13.2.2	STB_IMAGE_WRITE_IMPLEMENTATION	50

4.13.3	Variable Documentation	51
4.13.3.1	Rendering_Handler	51
4.14	Rendering/OpenGL-Wrappers.hpp File Reference	51
4.14.1	Detailed Description	51
4.14.2	Enumeration Type Documentation	52
4.14.2.1	PROGRAM	52
4.14.3	Function Documentation	52
4.14.3.1	openGLError()	52
4.14.4	Variable Documentation	52
4.14.4.1	Rendering_Handler	52
4.15	Rendering/Window-Management.cpp File Reference	52
4.15.1	Macro Definition Documentation	53
4.15.1.1	CAM_SPEED	53
4.15.2	Function Documentation	53
4.15.2.1	callBackInit()	53
4.15.2.2	create_context()	53
4.15.2.3	createWindow()	53
4.15.2.4	cursor_pos_callback()	53
4.15.2.5	cursorSelectNode()	54
4.15.2.6	error_callback()	54
4.15.2.7	key_callback()	54
4.15.2.8	mouse_button_callback()	54
4.15.2.9	openGLError()	54
4.16	Rendering/Window-Management.hpp File Reference	54
4.16.1	Function Documentation	55
4.16.1.1	calculateFPS()	55
4.16.1.2	callBackInit()	55
4.16.1.3	create_context()	55
4.16.1.4	createWindow()	55
4.16.1.5	cursor_pos_callback()	56
4.16.1.6	error_callback()	56
4.16.1.7	key_callback()	56
4.16.1.8	mouse_button_callback()	56
4.17	World.cpp File Reference	56
4.17.1	Macro Definition Documentation	57
4.17.1.1	MESH	57
4.17.2	Variable Documentation	57
4.17.2.1	the_world	57
4.18	World.hpp File Reference	57
4.18.1	Macro Definition Documentation	58
4.18.1.1	CHUNK_DIMS	58
4.18.2	Variable Documentation	58
4.18.2.1	the_world	58

Chapter 1

Class Index

1.1 Class List

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

Camera	5
Chunk	11
Chunk_Holder	14
cirArray< T >	16
Cube	18
Light	21
Mesh	21
Object_3D	23
Renderer	25
Shader	29
Texture	32
World	34

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Cube.cpp	37
Cube.hpp	37
main.cpp	47
World.cpp	56
World.hpp	57
Helpers/ cout-definitions.cpp	
Implementation of the output functions for I/O debugging	38
Helpers/ cout-definitions.hpp	
Header defining some output methods to print structures to the terminal	40
Helpers/ system-libraries.hpp	
General header for system libraries	41
Helpers/ tools.cpp	
Implementation of miscellaneous helping functions and structures	42
Helpers/ tools.hpp	
Header for the definition of a generic chunk object	45
Helpers/ wavefront-loader.cpp	
Defines methods needed to load wavefront (.obj) meshes	46
Helpers/ wavefront-loader.hpp	
Header declaration of methods needed to load wavefront (.obj) meshes	47
Rendering/ OpenGL-Wrappers.cpp	
Wrapper structures to abstract OpenGL function calls	50
Rendering/ OpenGL-Wrappers.hpp	
Header to define variables, structure definitions, include libraries... Shared among all rendering functions	51
Rendering/ Window-Management.cpp	52
Rendering/ Window-Management.hpp	54
Rendering/Camera/ Camera.cpp	
Implementation of the camera header. Defines the behaviour for a generic camera	49
Rendering/Camera/ Camera.hpp	
Header declaration of functions and members for a generic camera class	49

Chapter 3

Class Documentation

3.1 Camera Class Reference

```
#include <Camera.hpp>
```

Public Member Functions

- [Camera](#) (mat3 frame, vec3 pos, float w, float h)
- [Camera](#) ()
- [~Camera](#) ()
- mat4 [getViewMatrix](#) ()
- mat4 [getPerspectiveMatrix](#) ()
- void [setLookDirection](#) (vec3 v)
- void [move](#) (vec3 v)
- void [setPosition](#) (vec3 p)
- void [turnH](#) (float angle)
- void [turnV](#) (float angle)
- void [incline](#) (float angle)
- void [resetView](#) ()
- void [resetCamera](#) ()
- vec3 [getPosition](#) ()
- vec3 [getForward](#) ()
- vec3 [getUp](#) ()
- vec3 [getSide](#) ()
- float [getFov](#) ()

Private Attributes

- vec3 [forward](#)
vector indicating forward direction of the camera
- vec3 [up](#)
vector indicating up direction of the camera
- vec3 [side](#)
vector indicating side direction of the camera
- vec3 [position](#)

- vector indicating the position of the camera*
- `vec3 orig_forward`
 - vector indicating forward direction of the camera*
- `vec3 orig_up`
 - vector indicating up direction of the camera*
- `vec3 orig_side`
 - vector indicating side direction of the camera*
- `vec3 orig_position`
 - vector indicating the position of the camera*
- `float fov`
 - field of view, dimensions, near clipping angle, far clipping angle*
- `float width`
- `float height`
- `float zNear`
- `float zFar`

3.1.1 Constructor & Destructor Documentation

3.1.1.1 `Camera()` [1/2]

```
Camera::Camera (
    mat3 frame,
    vec3 pos,
    float w,
    float h )
```

Parameter constructor

3.1.1.2 `Camera()` [2/2]

```
Camera::Camera ( )
```

Default constructor

3.1.1.3 `~Camera()`

```
Camera::~~Camera ( )
```

Destructor

3.1.2 Member Function Documentation

3.1.2.1 getForward()

```
vec3 Camera::getForward ( )
```

Return the direction in which the camera is looking

3.1.2.2 getFov()

```
float Camera::getFov ( )
```

Return the field of view of the camera

3.1.2.3 getPerspectiveMatrix()

```
mat4 Camera::getPerspectiveMatrix ( )
```

Get the perspective matrix of the camera

3.1.2.4 getPosition()

```
vec3 Camera::getPosition ( )
```

Return the global camera position

3.1.2.5 getSide()

```
vec3 Camera::getSide ( )
```

Return the side direction of the camera

3.1.2.6 getUp()

```
vec3 Camera::getUp ( )
```

Return the up direction of the camera

3.1.2.7 getViewMatrix()

```
mat4 Camera::getViewMatrix ( )
```

Get the view matrix of the camera

3.1.2.8 incline()

```
void Camera::incline (
    float angle )
```

Rotate the camera around its forward direction

3.1.2.9 move()

```
void Camera::move (
    vec3 v )
```

Move the camera by an offset v

3.1.2.10 resetCamera()

```
void Camera::resetCamera ( )
```

Hard reset all camera values to defaults

3.1.2.11 resetView()

```
void Camera::resetView ( )
```

Orient the camera to it's default looking direction and orientation

3.1.2.12 setLookDirection()

```
void Camera::setLookDirection (
    vec3 v )
```

Orient the camera so that it looks in the direction of v

3.1.2.13 setPosition()

```
void Camera::setPosition (
    vec3 p )
```

Place the camera at specified position p

3.1.2.14 turnH()

```
void Camera::turnH (
    float angle )
```

Rotate the camera around it's *up direction*

3.1.2.15 turnV()

```
void Camera::turnV (
    float angle )
```

Rotate the camera around it's *side direction*

3.1.3 Member Data Documentation

3.1.3.1 forward

```
vec3 Camera::forward [private]
```

vector indicating forward direction of the camera

3.1.3.2 fov

```
float Camera::fov [private]
```

field of view, dimensions, near clipping angle, far clipping angle

3.1.3.3 height

```
float Camera::height [private]
```

3.1.3.4 orig_forward

```
vec3 Camera::orig_forward [private]
```

vector indicating forward direction of the camera

3.1.3.5 orig_position

```
vec3 Camera::orig_position [private]
```

vector indicating the position of the camera

3.1.3.6 orig_side

```
vec3 Camera::orig_side [private]
```

vector indicating side direction of the camera

3.1.3.7 orig_up

```
vec3 Camera::orig_up [private]
```

vector indicating up direction of the camera

3.1.3.8 position

```
vec3 Camera::position [private]
```

vector indicating the position of the camera

3.1.3.9 side

```
vec3 Camera::side [private]
```

vector indicating side direction of the camera

3.1.3.10 up

```
vec3 Camera::up [private]
```

vector indicating up direction of the camera

3.1.3.11 width

```
float Camera::width [private]
```

3.1.3.12 zFar

```
float Camera::zFar [private]
```


3.1.3.13 zNear

```
float Camera::zNear [private]
```

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

- Rendering/Camera/[Camera.hpp](#)
- Rendering/Camera/[Camera.cpp](#)

3.2 Chunk Class Reference

```
#include <World.hpp>
```

Public Member Functions

- [Cube](#) * [operator\(\)](#) (int, int, int)
- [Chunk](#) ()
- [Chunk](#) (vec3)
- [Chunk](#) (vec3, [World](#) *)
- [~Chunk](#) ()
- void [create_cubes](#) (vec3)
- void [update](#) ()
- void [send_render_data](#) ([Renderer](#) *)

Public Attributes

- vec3 [position](#)

Private Member Functions

- void [update_visible_faces](#) ()
- bool [check_neighbour](#) ([Cube](#) *c, [Cube](#) *n)

Private Attributes

- [World](#) * [world](#)
- [Cube](#) * [chunk_cubes](#) [[CHUNK_DIMS](#) *[CHUNK_DIMS](#) *[CHUNK_DIMS](#)] = {}
- [Object_3D](#) * [render_data](#)
- vector< vec4 > [cubes_info](#)

3.2.1 Constructor & Destructor Documentation

3.2.1.1 `Chunk()` [1/3]

```
Chunk::Chunk ( )
```

3.2.1.2 `Chunk()` [2/3]

```
Chunk::Chunk (
    vec3 offset )
```

3.2.1.3 `Chunk()` [3/3]

```
Chunk::Chunk (
    vec3 offset,
    World * w )
```

3.2.1.4 `~Chunk()`

```
Chunk::~~Chunk ( )
```

3.2.2 Member Function Documentation

3.2.2.1 `check_neighbour()`

```
bool Chunk::check_neighbour (
    Cube * c,
    Cube * n ) [inline], [private]
```

3.2.2.2 `create_cubes()`

```
void Chunk::create_cubes (
    vec3 offset )
```

3.2.2.3 operator()

```
Cube * Chunk::operator() (
    int x,
    int y,
    int z )
```

3.2.2.4 send_render_data()

```
void Chunk::send_render_data (
    Renderer * handler ) [inline]
```

3.2.2.5 update()

```
void Chunk::update ( )
```

3.2.2.6 update_visible_faces()

```
void Chunk::update_visible_faces ( ) [private]
```

3.2.3 Member Data Documentation

3.2.3.1 chunk_cubes

```
Cube* Chunk::chunk_cubes[CHUNK_DIMS *CHUNK_DIMS *CHUNK_DIMS] = {} [private]
```

3.2.3.2 cubes_info

```
vector<vec4> Chunk::cubes_info [private]
```

3.2.3.3 position

```
vec3 Chunk::position
```

3.2.3.4 render_data

```
Object_3D* Chunk::render_data [private]
```

3.2.3.5 world

```
World* Chunk::world [private]
```

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

- [World.hpp](#)
- [World.cpp](#)

3.3 Chunk_Holder Class Reference

```
#include <World.hpp>
```

Public Member Functions

- [Chunk_Holder](#) ()
- [Chunk_Holder](#) (int, int, int, [World](#) *)
- [~Chunk_Holder](#) ()
- [Chunk](#) * [operator\(\)](#) (int, int, int)
- void [shift](#) (ivec3)

Private Attributes

- [cirArray](#)< [cirArray](#)< [cirArray](#)< [Chunk](#) * > > > [chunkBox](#)
- [World](#) * [world](#)

3.3.1 Constructor & Destructor Documentation

3.3.1.1 Chunk_Holder() [1/2]

```
Chunk_Holder::Chunk_Holder ( )
```

3.3.1.2 Chunk_Holder() [2/2]

```
Chunk_Holder::Chunk_Holder (
    int x_dim,
    int y_dim,
    int z_dim,
    World * w )
```

3.3.1.3 ~Chunk_Holder()

```
Chunk_Holder::~~Chunk_Holder ( )
```

3.3.2 Member Function Documentation

3.3.2.1 operator>()

```
Chunk * Chunk_Holder::operator() (
    int x,
    int y,
    int z )
```

3.3.2.2 shift()

```
void Chunk_Holder::shift (
    ivec3 offset )
```

3.3.3 Member Data Documentation

3.3.3.1 chunkBox

```
cirArray<cirArray<cirArray<Chunk*> > > Chunk_Holder::chunkBox [private]
```

3.3.3.2 world

```
World* Chunk_Holder::world [private]
```

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

- [World.hpp](#)
- [World.cpp](#)

3.4 cirArray< T > Class Template Reference

```
#include <tools.hpp>
```

Public Member Functions

- [cirArray](#) ()
- [cirArray](#) (uint [size](#))
- void [shift](#) (int)
- T & [operator\[\]](#) (int)
- void [operator=](#) (T)
- uint [size](#) ()

Private Attributes

- vector< T > [array](#)
- int [start](#)

3.4.1 Detailed Description

```
template<typename T>
class cirArray< T >
```

A generic circular array class, can be considered a ciircular vector

3.4.2 Constructor & Destructor Documentation

3.4.2.1 cirArray() [1/2]

```
template<typename T >
cirArray< T >::cirArray ( )
```

Default constructor of the class

3.4.2.2 cirArray() [2/2]

```
template<typename T >
cirArray< T >::cirArray (
    uint size )
```

Parametrized constructor of the class, creates a circular array of *size elements*

3.4.3 Member Function Documentation

3.4.3.1 operator=()

```
template<typename T>
void cirArray< T >::operator= (
    T )
```

3.4.3.2 operator[]()

```
template<typename T >
T & cirArray< T >::operator[] (
    int i )
```

[] operator, returns the element at index *i*

3.4.3.3 shift()

```
template<typename T >
void cirArray< T >::shift (
    int i )
```

shift the circular array by *i units*

3.4.3.4 size()

```
template<typename T >
uint cirArray< T >::size ( )
```

Returns the current size (number of elements) of the circular array

3.4.4 Member Data Documentation

3.4.4.1 array

```
template<typename T>
vector<T> cirArray< T >::array [private]
```

3.4.4.2 start

```
template<typename T>
int cirArray< T >::start [private]
```

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

- [Helpers/tools.hpp](#)

3.5 Cube Class Reference

```
#include <Cube.hpp>
```

Public Member Functions

- void [update](#) (vec3 offset)
- [Cube](#) (vec3 p, [CubeID](#) type)
- [Cube](#) (vec3 p)
- [Cube](#) ()
- [~Cube](#) ()
- [Mesh](#) [getMesh](#) ()

Static Public Member Functions

- static void [initialize](#) ()
- static void [cleanup](#) ()

Public Attributes

- vec3 [position](#)
- [CubeID](#) [cube_type](#) = [DEFAULT](#)
- bool [transparent](#) = false

Static Public Attributes

- static vector< [Mesh](#) * > [meshes](#)
- static vector< [Texture](#) * > [textures](#)

3.5.1 Constructor & Destructor Documentation

3.5.1.1 Cube() [1/3]

```
Cube::Cube (
    vec3 p,
    CubeID type )
```

3.5.1.2 Cube() [2/3]

```
Cube::Cube (
    vec3 p )
```

3.5.1.3 Cube() [3/3]

```
Cube::Cube ( )
```

3.5.1.4 ~Cube()

```
Cube::~Cube ( )
```

3.5.2 Member Function Documentation

3.5.2.1 cleanup()

```
void Cube::cleanup ( ) [static]
```

3.5.2.2 getMesh()

```
Mesh Cube::getMesh ( )
```

3.5.2.3 initialize()

```
void Cube::initialize ( ) [static]
```

3.5.2.4 update()

```
void Cube::update (
    vec3 offset )
```

3.5.3 Member Data Documentation

3.5.3.1 cube_type

```
CubeID Cube::cube_type = DEFAULT
```

3.5.3.2 meshes

```
vector< Mesh * > Cube::meshes [static]
```

3.5.3.3 position

```
vec3 Cube::position
```

3.5.3.4 textures

```
vector< Texture * > Cube::textures [static]
```

3.5.3.5 transparent

```
bool Cube::transparent = false
```

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

- [Cube.hpp](#)
- [Cube.cpp](#)

3.6 Light Struct Reference

```
#include <World.hpp>
```

Public Attributes

- `vec3` [position](#)
- `vec4` [color](#)
- `double` [intensity](#)

3.6.1 Member Data Documentation

3.6.1.1 color

```
vec4 Light::color
```

3.6.1.2 intensity

```
double Light::intensity
```

3.6.1.3 position

```
vec3 Light::position
```

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

- [World.hpp](#)

3.7 Mesh Struct Reference

```
#include <OpenGL-Wrappers.hpp>
```

Public Member Functions

- [~Mesh](#) ()

Public Attributes

- `vector< vec3 > vertices`
Vertex data.
- `vector< vec3 > normals`
Normal data.
- `vector< uint > indices`
Element data (sequence in which data will be read)
- `vector< vec2 > uvs`

3.7.1 Constructor & Destructor Documentation

3.7.1.1 `~Mesh()`

```
Mesh::~Mesh ( )
```

Class destructor

3.7.2 Member Data Documentation

3.7.2.1 `indices`

```
vector<uint> Mesh::indices
```

Element data (sequence in which data will be read)

3.7.2.2 `normals`

```
vector<vec3> Mesh::normals
```

Normal data.

3.7.2.3 `uvs`

```
vector<vec2> Mesh::uvs
```

[Texture](#) data for this geometry (the associated coordinates of the mesh)

3.7.2.4 vertices

```
vector<vec3> Mesh::vertices
```

Vertex data.

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

- [Rendering/OpenGL-Wrappers.hpp](#)
- [Rendering/OpenGL-Wrappers.cpp](#)

3.8 Object_3D Class Reference

```
#include <OpenGL-Wrappers.hpp>
```

Public Member Functions

- [Object_3D](#) ([Mesh](#) *)
- `template<class T >`
void [set_instance_data](#) ([Renderer](#) *, `vector< T >`)

Public Attributes

- GLuint [VAO](#)
Vertex Array Object.
- `vector< GLuint >` [VBOs](#)
array of VBO Ids
- `vector< GLuint >` [types](#)
Array of VBO types.
- uint [layouts](#)
The number of layouts to activate.
- uint [render_instances](#)
Number of instances to render current object.
- uint [mesh_indices](#)
Indices for index rendering, if any.

3.8.1 Constructor & Destructor Documentation

3.8.1.1 Object_3D()

```
Object_3D::Object_3D (
    Mesh * mesh )
```

Create a 3D rendereable object from a mesh

3.8.2 Member Function Documentation

3.8.2.1 set_instance_data()

```
template<class T >
void Object_3D::set_instance_data (
    Renderer * handler,
    vector< T > info )
```

Set the visual data for the current 3D object (SSBO data)

3.8.3 Member Data Documentation

3.8.3.1 layouts

```
uint Object_3D::layouts
```

The number of layouts to activate.

3.8.3.2 mesh_indices

```
uint Object_3D::mesh_indices
```

Indices for index rendering, if any.

3.8.3.3 render_instances

```
uint Object_3D::render_instances
```

Number of instances to render current object.

3.8.3.4 types

```
vector<GLuint> Object_3D::types
```

Array of VBO types.

3.8.3.5 VAO

```
GLuint Object_3D::VAO
```

Vertex Array Object.

3.8.3.6 VBOs

```
vector<GLuint> Object_3D::VBOs
```

array of VBO ids

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

- Rendering/[OpenGL-Wrappers.hpp](#)
- Rendering/[OpenGL-Wrappers.cpp](#)

3.9 Renderer Class Reference

```
#include <OpenGL-Wrappers.hpp>
```

Public Member Functions

- [Renderer](#) ()
- [Renderer](#) (int width, int height)
- [~Renderer](#) ()
- [Shader](#) * [find_shader](#) (string shader_name)
- void [update](#) (GLFWwindow *window)
- void [add_Shader](#) (string shader, GLuint type)
- void [make_program](#) (vector< uint > *shaders)
- void [set_camera](#) ([Camera](#) *new_cam)
- void [multi_render](#) (GLuint VAO, vector< GLuint > *VBOs, vector< GLuint > *buffer_types, GLuint layout↵_num, GLuint index_num, GLuint instances)
- void [change_active_program](#) (GLuint newProgram)
- void [add_data](#) ([Object_3D](#) *)
- void [render](#) ()
- void [clear](#) ()

Public Attributes

- mutex [busy_queue](#)
Lock to synchronize queue W/R.
- [Camera](#) * [cam](#)
Main (player) camera object.
- GLuint [current_program](#)
Current shading program (program used to render)

Private Attributes

- vector< GLuint > [shading_programs](#)
Shading programs IDs.
- vector< [Shader](#) > [vertex_shaders](#)
Vertex shader IDs.
- vector< [Shader](#) > [fragment_shaders](#)
Fragment shader IDs.
- vector< [Shader](#) > [tessellation_shaders](#)
Tessellation shader IDs.
- vector< [Object_3D](#) * > [render_queue](#)

3.9.1 Constructor & Destructor Documentation

3.9.1.1 [Renderer\(\)](#) [1/2]

```
Renderer::Renderer ( )
```

Default constructor for the [Renderer](#) Class

3.9.1.2 [Renderer\(\)](#) [2/2]

```
Renderer::Renderer (
    int width,
    int height )
```

Constructor for the [Renderer](#) class. Creates a renderer object that handles all render calls. It's intended to be unique but has not been implemented as a singleton be weary!

3.9.1.3 [~Renderer\(\)](#)

```
Renderer::~~Renderer ( )
```

Class destructor

3.9.2 Member Function Documentation

3.9.2.1 [add_data\(\)](#)

```
void Renderer::add_data (
    Object\_3D * data )
```

Add a rendereable 3D object to the current render queue

3.9.2.2 add_Shader()

```
void Renderer::add_Shader (
    string shader,
    GLuint type )
```

Add a new shader to the set of all shaders

3.9.2.3 change_active_program()

```
void Renderer::change_active_program (
    GLuint newProgram )
```

3.9.2.4 clear()

```
void Renderer::clear ( )
```

Clear all objects in the render queue

3.9.2.5 find_shader()

```
Shader * Renderer::find_shader (
    string shader_name )
```

Find a shader through a string

3.9.2.6 make_program()

```
void Renderer::make_program (
    vector< uint > * shaders )
```

3.9.2.7 multi_render()

```
void Renderer::multi_render (
    GLuint VAO,
    vector< GLuint > * VBOs,
    vector< GLuint > * buffer_types,
    GLuint layout_num,
    GLuint index_num,
    GLuint instances )
```

Function to render multiple instances of the same mesh *index_num* is the number of indices in the mesh (for drawing elements) *layout_num* is the number of layouts to enable (always 0 to *layout_num*-1)

3.9.2.8 render()

```
void Renderer::render ( )
```

Render all elements in the current render queue

3.9.2.9 set_camera()

```
void Renderer::set_camera (
    Camera * new_cam )
```

Initialize the main rendering camera

3.9.2.10 update()

```
void Renderer::update (
    GLFWwindow * window )
```

Update general rendering values

3.9.3 Member Data Documentation

3.9.3.1 busy_queue

```
mutex Renderer::busy_queue
```

Lock to synchronize queue W/R.

3.9.3.2 cam

```
Camera* Renderer::cam
```

Main (player) camera object.

3.9.3.3 current_program

```
GLuint Renderer::current_program
```

Current shading program (program used to render)

3.9.3.4 fragment_shaders

```
vector<Shader> Renderer::fragment_shaders [private]
```

Fragment shader IDs.

3.9.3.5 render_queue

```
vector<Object_3D*> Renderer::render_queue [private]
```

Queue of objects to render in the current frame

3.9.3.6 shading_programs

```
vector<GLuint> Renderer::shading_programs [private]
```

Shading programs IDs.

3.9.3.7 tessellation_shaders

```
vector<Shader> Renderer::tessellation_shaders [private]
```

Tessellation shader IDs.

3.9.3.8 vertex_shaders

```
vector<Shader> Renderer::vertex_shaders [private]
```

Vertex shader IDs.

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

- [Rendering/OpenGL-Wrappers.hpp](#)
- [Rendering/OpenGL-Wrappers.cpp](#)

3.10 Shader Class Reference

```
#include <OpenGL-Wrappers.hpp>
```

Public Member Functions

- [Shader](#) ()
- [Shader](#) (string file, GLenum [type](#))
- [~Shader](#) ()
- string [load_from_file](#) (string &)
- void [clear](#) ()

Public Attributes

- string [fileName](#)
source file
- GLuint [shaderID](#)
generated OpenGL shader ID
- GLuint [type](#)
shader type

3.10.1 Constructor & Destructor Documentation

3.10.1.1 [Shader](#)() [1/2]

```
Shader::Shader ( )
```

Default Constructor

3.10.1.2 [Shader](#)() [2/2]

```
Shader::Shader (
    string file,
    GLenum type )
```

Initialize the fields of a shader object using a glsl shader file

Parameters

<i>file</i>	the file path (relative or absolute) where the shader program is defined
<i>type</i>	the type of shader (e.g vertex,fragment, tessellation...)

3.10.1.3 [~Shader](#)()

```
Shader::~~Shader ( )
```

Destructor of a shader struct

3.10.2 Member Function Documentation

3.10.2.1 clear()

```
void Shader::clear ( )
```

Cleanup the shader OpenGL information

3.10.2.2 load_from_file()

```
string Shader::load_from_file (
    string & filepath )
```

Copy a file into a a string

Parameters

<i>filepath</i>	path to the file
-----------------	------------------

Returns

A string that is the copy of the source file

3.10.3 Member Data Documentation

3.10.3.1 fileName

```
string Shader::fileName
```

source file

3.10.3.2 shaderID

```
GLuint Shader::shaderID
```

generated OpenGL shader ID

3.10.3.3 type

```
GLuint Shader::type
```

shader type

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

- Rendering/[OpenGL-Wrappers.hpp](#)
- Rendering/[OpenGL-Wrappers.cpp](#)

3.11 Texture Class Reference

```
#include <OpenGL-Wrappers.hpp>
```

Public Member Functions

- [Texture](#) (const char *filename, GLuint [target](#)=GL_TEXTURE_2D)
- [~Texture](#) ()
- void [load_to_GPU](#) (GLuint)
- void [clear](#) ()

Public Attributes

- GLuint [textureID](#)
OpenGL generated ID for the texture.
- GLuint [target](#)
OpenGL target (Usually 2D texture or rectangle) check OpenGL doc.
- string [texture](#)
Texture data.
- int [width](#)
width of the texture
- int [height](#)
height of the texture

3.11.1 Constructor & Destructor Documentation

3.11.1.1 Texture()

```
Texture::Texture (
    const char * filename,
    GLuint targ = GL_TEXTURE_2D )
```

Initialize the fields of a texture object using arrays

Parameters

<i>filename</i>	the filepath to the texture file
<i>targ</i>	the OpenGL texture target (e.g 2D, rectangle...)

Returns

Boolean value indicating whether an error occurred (true means no error)

3.11.1.2 ~Texture()

```
Texture::~Texture ( )
```

Destructor of a texture struct

3.11.2 Member Function Documentation**3.11.2.1 clear()**

```
void Texture::clear ( )
```

Clear all OpenGL information of the texture object

3.11.2.2 load_to_GPU()

```
void Texture::load_to_GPU (
    GLuint program )
```

3.11.3 Member Data Documentation**3.11.3.1 height**

```
int Texture::height
```

height of the texture

3.11.3.2 target

```
GLuint Texture::target
```

OpenGL target (Usually 2D texture or rectangle) check OpenGL doc.

3.11.3.3 texture

```
string Texture::texture
```

[Texture](#) data.

3.11.3.4 textureID

```
GLuint Texture::textureID
```

OpenGL generated ID for the texture.

3.11.3.5 width

```
int Texture::width
```

width of the texture

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

- [Rendering/OpenGL-Wrappers.hpp](#)
- [Rendering/OpenGL-Wrappers.cpp](#)

3.12 World Class Reference

```
#include <World.hpp>
```

Public Member Functions

- [World](#) ()
- [~World](#) ()
- [Cube * operator\(\)](#) (int x, int y, int z)
- void [center_frame](#) (ivec3 offset)
- void [send_render_data](#) ([Renderer](#) *)

Public Attributes

- int `h_radius` = 7
- int `v_radius` = 4
- ivec3 `origin` = ivec3(0)

Private Attributes

- `Chunk_Holder` * `loaded_chunks`
- vector< `Light` > `loaded_lights`

3.12.1 Constructor & Destructor Documentation

3.12.1.1 World()

```
World::World ( )
```

3.12.1.2 ~World()

```
World::~~World ( )
```

3.12.2 Member Function Documentation

3.12.2.1 center_frame()

```
void World::center_frame (
    ivec3 offset )
```

3.12.2.2 operator>()()

```
Cube * World::operator() (
    int x,
    int y,
    int z )
```

3.12.2.3 send_render_data()

```
void World::send_render_data (
    Renderer * handler )
```

3.12.3 Member Data Documentation

3.12.3.1 h_radius

```
int World::h_radius = 7
```

3.12.3.2 loaded_chunks

```
Chunk\_Holder* World::loaded_chunks [private]
```

3.12.3.3 loaded_lights

```
vector<Light> World::loaded_lights [private]
```

3.12.3.4 origin

```
ivec3 World::origin = ivec3(0)
```

3.12.3.5 v_radius

```
int World::v_radius = 4
```

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

- [World.hpp](#)
- [World.cpp](#)

Chapter 4

File Documentation

4.1 Cube.cpp File Reference

```
#include "system-libraries.hpp"  
#include "Cube.hpp"  
#include "cout-definitions.hpp"
```

Variables

- `vector< string > texture_source_files` = {"Assets/Textures/white_cube.png"}
- `vector< string > obj_source_files` = {"Assets/Objs/cube.obj"}

4.1.1 Variable Documentation

4.1.1.1 obj_source_files

```
vector<string> obj_source_files = {"Assets/Objs/cube.obj"}
```

4.1.1.2 texture_source_files

```
vector<string> texture_source_files = {"Assets/Textures/white_cube.png"}
```

4.2 Cube.hpp File Reference

```
#include <string>  
#include "OpenGL-Wrappers.hpp"  
#include "wavefront-loader.hpp"
```

Classes

- class [Cube](#)

Enumerations

- enum [CubeID](#) { [DEFAULT](#) =0 }

Variables

- const uint [cube_types](#) = 1

4.2.1 Enumeration Type Documentation

4.2.1.1 CubeID

enum [CubeID](#)

Enumerator

DEFAULT	
---------	--

4.2.2 Variable Documentation

4.2.2.1 cube_types

```
const uint cube_types = 1
```

4.3 Helpers/cout-definitions.cpp File Reference

Implementation of the output functions for I/O debugging.

```
#include "cout-definitions.hpp"
```

Functions

- ostream & [operator<<](#) (ostream &os, vec2 &v)
- ostream & [operator<<](#) (ostream &os, vec3 &v)
- ostream & [operator<<](#) (ostream &os, vec4 &v)
- ostream & [operator<<](#) (ostream &os, vector< float > &v)

4.3.1 Detailed Description

Implementation of the output functions for I/O debugging.

Author

Camilo Talero

Version: 0.0.2

4.3.2 Function Documentation

4.3.2.1 `operator<<()` [1/4]

```
ostream& operator<< (
    ostream & os,
    vec2 & v )
```

Print a vec2

4.3.2.2 `operator<<()` [2/4]

```
ostream& operator<< (
    ostream & os,
    vec3 & v )
```

Print a vec3

4.3.2.3 `operator<<()` [3/4]

```
ostream& operator<< (
    ostream & os,
    vec4 & v )
```

Print a vec4

4.3.2.4 `operator<<()` [4/4]

```
ostream& operator<< (
    ostream & os,
    vector< float > & v )
```

Print a vector of floats

4.4 Helpers/cout-definitions.hpp File Reference

Header defining some output methods to print structures to the terminal.

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtx/transform.hpp>
#include <glm/gtc/type_ptr.hpp>
#include <string>
#include <vector>
#include <unistd.h>
```

Functions

- ostream & [operator<<](#) (ostream &os, vec2 &v)
- ostream & [operator<<](#) (ostream &os, vec3 &v)
- ostream & [operator<<](#) (ostream &os, vec4 &v)
- ostream & [operator<<](#) (ostream &os, vector< float > &v)

4.4.1 Detailed Description

Header defining some output methods to print structures to the terminal.

Author

Camilo Talero

Version: 0.0.2

4.4.2 Function Documentation

4.4.2.1 [operator<<\(\)](#) [1/4]

```
ostream& operator<< (
    ostream & os,
    vec2 & v )
```

Print a vec2

4.4.2.2 operator<<() [2/4]

```
ostream& operator<< (
    ostream & os,
    vec3 & v )
```

Print a vec3

4.4.2.3 operator<<() [3/4]

```
ostream& operator<< (
    ostream & os,
    vec4 & v )
```

Print a vec4

4.4.2.4 operator<<() [4/4]

```
ostream& operator<< (
    ostream & os,
    vector< float > & v )
```

Print a vector of floats

4.5 Helpers/system-libraries.hpp File Reference

General header for system libraries.

```
#include <GL/glew.h>
#include <GLFW/glfw3.h>
#include <string>
#include <iostream>
#include <vector>
#include <fstream>
#include <cstdlib>
#include <unistd.h>
#include <time.h>
#include <thread>
#include <mutex>
#include <math.h>
#include <chrono>
#include <ctime>
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtx/transform.hpp>
#include <glm/gtc/type_ptr.hpp>
#include <ft2build.h>
```

Macros

- `#define GLEW_DYNAMIC`

4.5.1 Detailed Description

General header for system libraries.

Author

Camilo Talero

Version: 0.0.2

4.5.2 Macro Definition Documentation

4.5.2.1 GLEW_DYNAMIC

```
#define GLEW_DYNAMIC
```

4.6 Helpers/tools.cpp File Reference

Implementation of miscellaneous helping functions and structures.

```
#include "tools.hpp"
```

Functions

- void [vec_field_init](#) ()
- double [fade](#) (double d)
- double [length](#) (double x, double y)
- double [surflet](#) (double x, double y, double grad_x, double grad_y)
- double [perlin_noise](#) (double x, double y)
- double [noise_2D](#) (double x, double y)

Variables

- int const [size](#) = 256
- int const [mask](#) = [size](#)-1
- int [perm](#) [[size](#)]
- float [vec_field_x](#) [[size](#)]
- float [vec_field_y](#) [[size](#)]

4.6.1 Detailed Description

Implementation of miscellaneous helping functions and structures.

Author

Camilo Talero

Version: 0.0.2

Perlin noise implementation was done following the information at: <http://eastfarthing.com/blog/2015-04-21-nois>

4.6.2 Function Documentation

4.6.2.1 fade()

```
double fade (
    double d ) [inline]
```

Function to smooth out the transition from each grid cell to another $f(x)=1-6*|x|^5-15|x|^4+10|x|^3$

4.6.2.2 length()

```
double length (
    double x,
    double y ) [inline]
```

Return the length of the vector (x,y) for radial fading.

4.6.2.3 noise_2D()

```
double noise_2D (
    double x,
    double y )
```

Composite 2D noise function. Combines multiple iterations of Perlin noise at different sampling rates and amplitudes and merges them using octaves to create more complex noise functions

4.6.2.4 perlin_noise()

```
double perlin_noise (
    double x,
    double y )
```

2D Perlin Noise funtion

4.6.2.5 surflet()

```
double surflet (
    double x,
    double y,
    double grad_x,
    double grad_y ) [inline]
```

2D convolution surflet function, returns a scalar based on the gradient at (x,y)

4.6.2.6 vec_field_init()

```
void vec_field_init ( )
```

Initialize the perlin noise grid. We basically rotate a 2D vector 2PI units in the counter clockwise direction and assign a random location to it in a lookup table

4.6.3 Variable Documentation

4.6.3.1 mask

```
int const mask = size-1
```

4.6.3.2 perm

```
int perm[size]
```

4.6.3.3 size

```
int const size = 256
```

4.6.3.4 vec_field_x

```
float vec_field_x[size]
```

4.6.3.5 vec_field_y

```
float vec_field_y[size]
```

4.7 Helpers/tools.hpp File Reference

Header for the definition of a generic chunk object.

```
#include "system-libraries.hpp"
```

Classes

- class `cirArray< T >`

Functions

- double `noise_2D` (double x, double y)
- void `vec_field_init` ()

4.7.1 Detailed Description

Header for the definition of a generic chunk object.

Author

Camilo Talero

Version: 0.0.2

4.7.2 Function Documentation

4.7.2.1 noise_2D()

```
double noise_2D (  
    double x,  
    double y )
```

Composite 2D noise function. Combines multiple iterations of Perlin noise at different sampling rates and amplitudes and merges them using octaves to create more complex noise functions

4.7.2.2 `vec_field_init()`

```
void vec_field_init ( )
```

Initialize the perlin noise grid. We basically rotate a 2D vector 2PI units in the counter clockwise direction and assign a random location to it in a lookup table

4.8 Helpers/wavefront-loader.cpp File Reference

Defines methods needed to load wavefront (.obj) meshes.

```
#include "wavefront-loader.hpp"  
#include <algorithm>
```

Functions

- void `load_obj` (string filename, vector< float > *vertices, vector< float > *normals, vector< float > *texture_coords)

4.8.1 Detailed Description

Defines methods needed to load wavefront (.obj) meshes.

Author

Camilo Talero

Version: 0.0.2

4.8.2 Function Documentation

4.8.2.1 `load_obj()`

```
void load_obj (   
    string filename,   
    vector< float > * vertices,   
    vector< float > * normals,   
    vector< float > * texture_coords )
```

Function to load the mesh information from a .obj file, it assumes triangular meshes only. All return arrays must be cleared before using the function, else information will be returned at the end of the arrays.

Params: filename: the path to the file to be loaded. vertices: a pointer to a vector of floats where the vertex information will be loaded normals: a pointer to a vector of floats where the normal information will be loaded texture_coords: a pointer to a vector of floats where the texture mapping information will be loaded

4.9 Helpers/wavefront-loader.hpp File Reference

Header declaration of methods needed to load wavefront (.obj) meshes.

```
#include <fstream>
#include <iostream>
#include <sstream>
#include <vector>
#include <stdlib.h>
#include <string>
```

Functions

- void [load_obj](#) (std::string filename, std::vector< float > *vertices, std::vector< float > *normals, std::vector< float > *texture_coords)

4.9.1 Detailed Description

Header declaration of methods needed to load wavefront (.obj) meshes.

Author

: Camilo Talero

Version: 0.0.2

4.9.2 Function Documentation

4.9.2.1 load_obj()

```
void load_obj (
    std::string filename,
    std::vector< float > * vertices,
    std::vector< float > * normals,
    std::vector< float > * texture_coords )
```

4.10 main.cpp File Reference

```
#include "system-libraries.hpp"
#include "Window-Management.hpp"
#include "Cube.hpp"
#include "World.hpp"
```

Typedefs

- typedef std::chrono::duration< int, std::ratio< 1, 60 > > [frame_duration](#)
- typedef std::chrono::duration< int, std::ratio< 1, 600 > > [world_duration](#)

Functions

- void [render_loop](#) (GLFWwindow *window)
- void [update_loop](#) (GLFWwindow *, GLFWwindow *)
- int [main](#) (int argc, char **argv)

4.10.1 Typedef Documentation

4.10.1.1 [frame_duration](#)

```
typedef std::chrono::duration<int, std::ratio<1, 60> > frame\_duration
```

4.10.1.2 [world_duration](#)

```
typedef std::chrono::duration<int, std::ratio<1, 600> > world\_duration
```

4.10.2 Function Documentation

4.10.2.1 [main\(\)](#)

```
int main (  
    int argc,  
    char ** argv )
```

4.10.2.2 [render_loop\(\)](#)

```
void render_loop (  
    GLFWwindow * window )
```

4.10.2.3 update_loop()

```
void update_loop (
    GLFWwindow * window,
    GLFWwindow * o_window )
```

4.11 Rendering/Camera/Camera.cpp File Reference

Implementation of the camera header. Defines the behaviour for a generic camera.

```
#include "Camera.hpp"
```

4.11.1 Detailed Description

Implementation of the camera header. Defines the behaviour for a generic camera.

Author

Camilo Talero

Version: 0.0.2

4.12 Rendering/Camera/Camera.hpp File Reference

Header declaration of functions and members for a generic camera class.

```
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtx/transform.hpp>
```

Classes

- class [Camera](#)

4.12.1 Detailed Description

Header declaration of functions and members for a generic camera class.

Author

Camilo Talero

Version: 0.0.2

4.13 Rendering/OpenGL-Wrappers.cpp File Reference

Wrapper structures to abstract OpenGL function calls.

```
#include <stb/stb_image.h>
#include <stb/stb_image_write.h>
#include "system-libraries.hpp"
#include "OpenGL-Wrappers.hpp"
```

Macros

- `#define STB_IMAGE_IMPLEMENTATION`
- `#define STB_IMAGE_WRITE_IMPLEMENTATION`

Variables

- `Renderer * Rendering_Handler`
The global render handler.

4.13.1 Detailed Description

Wrapper structures to abstract OpenGL function calls.

Author

Camilo Talero

Version: 0.0.2

4.13.2 Macro Definition Documentation

4.13.2.1 STB_IMAGE_IMPLEMENTATION

```
#define STB_IMAGE_IMPLEMENTATION
```

4.13.2.2 STB_IMAGE_WRITE_IMPLEMENTATION

```
#define STB_IMAGE_WRITE_IMPLEMENTATION
```


4.13.3 Variable Documentation

4.13.3.1 Rendering_Handler

`Renderer*` `Rendering_Handler`

The global render handler.

4.14 Rendering/OpenGL-Wrappers.hpp File Reference

Header to define variables, structure definitions, include libraries... Shared among all rendering functions.

```
#include "system-libraries.hpp"
#include "Camera.hpp"
#include "cout-definitions.hpp"
```

Classes

- class `Shader`
- class `Texture`
- struct `Mesh`
- class `Renderer`
- class `Object_3D`

Enumerations

- enum `PROGRAM`

Functions

- int `openGLerror` ()

Variables

- `Renderer *` `Rendering_Handler`
The global render handler.

4.14.1 Detailed Description

Header to define variables, structure definitions, include libraries... Shared among all rendering functions.

Author

Camilo Talero

Version: 0.0.2

4.14.2 Enumeration Type Documentation

4.14.2.1 PROGRAM

enum [PROGRAM](#)

4.14.3 Function Documentation

4.14.3.1 openGLerror()

```
int openGLerror ( )
```

4.14.4 Variable Documentation

4.14.4.1 Rendering_Handler

[Renderer*](#) Rendering_Handler

The global render handler.

4.15 Rendering/Window-Management.cpp File Reference

```
#include "Window-Management.hpp"
```

Macros

- `#define` [CAM_SPEED](#) 0.3f

Functions

- GLFWwindow * [create_context](#) (GLFWwindow *other_window, bool visible)
- int [openGLerror](#) ()
- void [callBackInit](#) (GLFWwindow *window)
- GLFWwindow * [createWindow](#) (GLFWwindow *other_window, bool visible)
- int [cursorSelectNode](#) (GLFWwindow *window)
- void [error_callback](#) (int error, const char *description)
- void [cursor_pos_callback](#) (GLFWwindow *window, double xpos, double ypos)
- void [mouse_button_callback](#) (GLFWwindow *window, int button, int action, int mods)
- void [key_callback](#) (GLFWwindow *window, int key, int scancode, int action, int mods)

4.15.1 Macro Definition Documentation

4.15.1.1 CAM_SPEED

```
#define CAM_SPEED 0.3f
```

4.15.2 Function Documentation

4.15.2.1 callBackInit()

```
void callBackInit (  
    GLFWwindow * window )
```

4.15.2.2 create_context()

```
GLFWwindow* create_context (  
    GLFWwindow * other_window,  
    bool visible )
```

4.15.2.3 createWindow()

```
GLFWwindow* createWindow (  
    GLFWwindow * other_window,  
    bool visible )
```

4.15.2.4 cursor_pos_callback()

```
void cursor_pos_callback (  
    GLFWwindow * window,  
    double xpos,  
    double ypos )
```

4.15.2.5 cursorSelectNode()

```
int cursorSelectNode (
    GLFWwindow * window )
```

4.15.2.6 error_callback()

```
void error_callback (
    int error,
    const char * description )
```

4.15.2.7 key_callback()

```
void key_callback (
    GLFWwindow * window,
    int key,
    int scancode,
    int action,
    int mods )
```

4.15.2.8 mouse_button_callback()

```
void mouse_button_callback (
    GLFWwindow * window,
    int button,
    int action,
    int mods )
```

4.15.2.9 openGLError()

```
int openGLError ( )
```

4.16 Rendering/Window-Management.hpp File Reference

```
#include "system-libraries.hpp"
#include "OpenGL-Wrappers.hpp"
```

Functions

- void [error_callback](#) (int error, const char *description)
- void [key_callback](#) (GLFWwindow *window, int key, int scancode, int action, int mods)
- void [mouse_button_callback](#) (GLFWwindow *window, int button, int action, int mods)
- void [cursor_pos_callback](#) (GLFWwindow *window, double xpos, double ypos)
- void [callBackInit](#) (GLFWwindow *window)
- double [calculateFPS](#) (double prevTime, double currentTime)
- GLFWwindow * [createWindow](#) (GLFWwindow *other_window, bool)
- GLFWwindow * [create_context](#) (GLFWwindow *other_window, bool)

4.16.1 Function Documentation

4.16.1.1 [calculateFPS\(\)](#)

```
double calculateFPS (  
    double prevTime,  
    double currentTime )
```

4.16.1.2 [callBackInit\(\)](#)

```
void callBackInit (  
    GLFWwindow * window )
```

4.16.1.3 [create_context\(\)](#)

```
GLFWwindow* create_context (  
    GLFWwindow * other_window,  
    bool )
```

4.16.1.4 [createWindow\(\)](#)

```
GLFWwindow* createWindow (  
    GLFWwindow * other_window,  
    bool )
```

4.16.1.5 `cursor_pos_callback()`

```
void cursor_pos_callback (
    GLFWwindow * window,
    double xpos,
    double ypos )
```

4.16.1.6 `error_callback()`

```
void error_callback (
    int error,
    const char * description )
```

4.16.1.7 `key_callback()`

```
void key_callback (
    GLFWwindow * window,
    int key,
    int scancode,
    int action,
    int mods )
```

4.16.1.8 `mouse_button_callback()`

```
void mouse_button_callback (
    GLFWwindow * window,
    int button,
    int action,
    int mods )
```

4.17 World.cpp File Reference

```
#include "World.hpp"
#include "cout-definitions.hpp"
```

Macros

- `#define MESH Cube::meshes[0]`

Variables

- [World](#) * [the_world](#)

4.17.1 Macro Definition Documentation

4.17.1.1 MESH

```
#define MESH Cube::meshes[0]
```

4.17.2 Variable Documentation

4.17.2.1 the_world

```
World* the_world
```

4.18 World.hpp File Reference

```
#include "Cube.hpp"  
#include "tools.hpp"
```

Classes

- struct [Light](#)
- class [Chunk](#)
- class [Chunk_Holder](#)
- class [World](#)

Macros

- #define [CHUNK_DIMS](#) 16

Variables

- [World](#) * [the_world](#)

4.18.1 Macro Definition Documentation

4.18.1.1 CHUNK_DIMS

```
#define CHUNK_DIMS 16
```

4.18.2 Variable Documentation

4.18.2.1 the_world

```
World* the_world
```


Index

- ~Camera
 - Camera, [6](#)
- ~Chunk
 - Chunk, [12](#)
- ~Chunk_Holder
 - Chunk_Holder, [15](#)
- ~Cube
 - Cube, [19](#)
- ~Mesh
 - Mesh, [22](#)
- ~Renderer
 - Renderer, [26](#)
- ~Shader
 - Shader, [30](#)
- ~Texture
 - Texture, [33](#)
- ~World
 - World, [35](#)
- add_Shader
 - Renderer, [26](#)
- add_data
 - Renderer, [26](#)
- array
 - cirArray, [17](#)
- busy_queue
 - Renderer, [28](#)
- CAM_SPEED
 - Window-Management.cpp, [53](#)
- CHUNK_DIMS
 - World.hpp, [58](#)
- calculateFPS
 - Window-Management.hpp, [55](#)
- callBackInit
 - Window-Management.cpp, [53](#)
 - Window-Management.hpp, [55](#)
- cam
 - Renderer, [28](#)
- Camera, [5](#)
 - ~Camera, [6](#)
 - Camera, [6](#)
 - forward, [9](#)
 - fov, [9](#)
 - getForward, [6](#)
 - getFov, [7](#)
 - getPerspectiveMatrix, [7](#)
 - getPosition, [7](#)
 - getSide, [7](#)
 - getUp, [7](#)
 - getViewMatrix, [7](#)
 - height, [9](#)
 - incline, [7](#)
 - move, [7](#)
 - orig_forward, [9](#)
 - orig_position, [9](#)
 - orig_side, [9](#)
 - orig_up, [9](#)
 - position, [10](#)
 - resetCamera, [8](#)
 - resetView, [8](#)
 - setLookDirection, [8](#)
 - setPosition, [8](#)
 - side, [10](#)
 - turnH, [8](#)
 - turnV, [8](#)
 - up, [10](#)
 - width, [10](#)
 - zFar, [10](#)
 - zNear, [10](#)
- center_frame
 - World, [35](#)
- change_active_program
 - Renderer, [27](#)
- check_neighbour
 - Chunk, [12](#)
- Chunk, [11](#)
 - ~Chunk, [12](#)
 - check_neighbour, [12](#)
 - Chunk, [11](#), [12](#)
 - chunk_cubes, [13](#)
 - create_cubes, [12](#)
 - cubes_info, [13](#)
 - operator(), [12](#)
 - position, [13](#)
 - render_data, [13](#)
 - send_render_data, [13](#)
 - update, [13](#)
 - update_visible_faces, [13](#)
 - world, [14](#)
- Chunk_Holder, [14](#)
 - ~Chunk_Holder, [15](#)
 - Chunk_Holder, [14](#)
 - chunkBox, [15](#)
 - operator(), [15](#)
 - shift, [15](#)
 - world, [15](#)
- chunk_cubes

- Chunk, 13
- chunkBox
 - Chunk_Holder, 15
- cirArray
 - array, 17
 - cirArray, 16
 - operator=, 17
 - operator[], 17
 - shift, 17
 - size, 17
 - start, 18
- cirArray< T >, 16
- cleanup
 - Cube, 19
- clear
 - Renderer, 27
 - Shader, 31
 - Texture, 33
- color
 - Light, 21
- cout-definitions.cpp
 - operator<<, 39
- cout-definitions.hpp
 - operator<<, 40, 41
- create_context
 - Window-Management.cpp, 53
 - Window-Management.hpp, 55
- create_cubes
 - Chunk, 12
- createWindow
 - Window-Management.cpp, 53
 - Window-Management.hpp, 55
- Cube, 18
 - ~Cube, 19
 - cleanup, 19
 - Cube, 19
 - cube_type, 20
 - getMesh, 19
 - initialize, 19
 - meshes, 20
 - position, 20
 - textures, 20
 - transparent, 20
 - update, 20
- Cube.cpp, 37
 - obj_source_files, 37
 - texture_source_files, 37
- Cube.hpp, 37
 - cube_types, 38
 - CubeID, 38
- cube_type
 - Cube, 20
- cube_types
 - Cube.hpp, 38
- CubeID
 - Cube.hpp, 38
- cubes_info
 - Chunk, 13
- current_program
 - Renderer, 28
- cursor_pos_callback
 - Window-Management.cpp, 53
 - Window-Management.hpp, 55
- cursorSelectNode
 - Window-Management.cpp, 53
- error_callback
 - Window-Management.cpp, 54
 - Window-Management.hpp, 56
- fade
 - tools.cpp, 43
- fileName
 - Shader, 31
- find_shader
 - Renderer, 27
- forward
 - Camera, 9
- fov
 - Camera, 9
- fragment_shaders
 - Renderer, 28
- frame_duration
 - main.cpp, 48
- GLEW_DYNAMIC
 - system-libraries.hpp, 42
- getForward
 - Camera, 6
- getFov
 - Camera, 7
- getMesh
 - Cube, 19
- getPerspectiveMatrix
 - Camera, 7
- getPosition
 - Camera, 7
- getSide
 - Camera, 7
- getUp
 - Camera, 7
- getViewMatrix
 - Camera, 7
- h_radius
 - World, 36
- height
 - Camera, 9
 - Texture, 33
- Helpers/cout-definitions.cpp, 38
- Helpers/cout-definitions.hpp, 40
- Helpers/system-libraries.hpp, 41
- Helpers/tools.cpp, 42
- Helpers/tools.hpp, 45
- Helpers/wavefront-loader.cpp, 46
- Helpers/wavefront-loader.hpp, 47
- incline

- Camera, [7](#)
- indices
 - Mesh, [22](#)
- initialize
 - Cube, [19](#)
- intensity
 - Light, [21](#)
- key_callback
 - Window-Management.cpp, [54](#)
 - Window-Management.hpp, [56](#)
- layouts
 - Object_3D, [24](#)
- length
 - tools.cpp, [43](#)
- Light, [21](#)
 - color, [21](#)
 - intensity, [21](#)
 - position, [21](#)
- load_from_file
 - Shader, [31](#)
- load_obj
 - wavefront-loader.cpp, [46](#)
 - wavefront-loader.hpp, [47](#)
- load_to_GPU
 - Texture, [33](#)
- loaded_chunks
 - World, [36](#)
- loaded_lights
 - World, [36](#)
- MESH
 - World.cpp, [57](#)
- main
 - main.cpp, [48](#)
- main.cpp, [47](#)
 - frame_duration, [48](#)
 - main, [48](#)
 - render_loop, [48](#)
 - update_loop, [48](#)
 - world_duration, [48](#)
- make_program
 - Renderer, [27](#)
- mask
 - tools.cpp, [44](#)
- Mesh, [21](#)
 - ~Mesh, [22](#)
 - indices, [22](#)
 - normals, [22](#)
 - uvs, [22](#)
 - vertices, [22](#)
- mesh_indices
 - Object_3D, [24](#)
- meshes
 - Cube, [20](#)
- mouse_button_callback
 - Window-Management.cpp, [54](#)
 - Window-Management.hpp, [56](#)
- move
 - Camera, [7](#)
- multi_render
 - Renderer, [27](#)
- noise_2D
 - tools.cpp, [43](#)
 - tools.hpp, [45](#)
- normals
 - Mesh, [22](#)
- obj_source_files
 - Cube.cpp, [37](#)
- Object_3D, [23](#)
 - layouts, [24](#)
 - mesh_indices, [24](#)
 - Object_3D, [23](#)
 - render_instances, [24](#)
 - set_instance_data, [24](#)
 - types, [24](#)
 - VAO, [24](#)
 - VBOs, [25](#)
- OpenGL-Wrappers.cpp
 - Rendering_Handler, [51](#)
 - STB_IMAGE_IMPLEMENTATION, [50](#)
 - STB_IMAGE_WRITE_IMPLEMENTATION, [50](#)
- OpenGL-Wrappers.hpp
 - openGLError, [52](#)
 - PROGRAM, [52](#)
 - Rendering_Handler, [52](#)
- openGLError
 - OpenGL-Wrappers.hpp, [52](#)
 - Window-Management.cpp, [54](#)
- operator<<
 - cout-definitions.cpp, [39](#)
 - cout-definitions.hpp, [40](#), [41](#)
- operator()
 - Chunk, [12](#)
 - Chunk_Holder, [15](#)
 - World, [35](#)
- operator=
 - cirArray, [17](#)
- operator[]
 - cirArray, [17](#)
- orig_forward
 - Camera, [9](#)
- orig_position
 - Camera, [9](#)
- orig_side
 - Camera, [9](#)
- orig_up
 - Camera, [9](#)
- origin
 - World, [36](#)
- PROGRAM
 - OpenGL-Wrappers.hpp, [52](#)
- perlin_noise
 - tools.cpp, [43](#)

- perm
 - tools.cpp, 44
- position
 - Camera, 10
 - Chunk, 13
 - Cube, 20
 - Light, 21
- render
 - Renderer, 27
- render_data
 - Chunk, 13
- render_instances
 - Object_3D, 24
- render_loop
 - main.cpp, 48
- render_queue
 - Renderer, 29
- Renderer, 25
 - ~Renderer, 26
 - add_Shader, 26
 - add_data, 26
 - busy_queue, 28
 - cam, 28
 - change_active_program, 27
 - clear, 27
 - current_program, 28
 - find_shader, 27
 - fragment_shaders, 28
 - make_program, 27
 - multi_render, 27
 - render, 27
 - render_queue, 29
 - Renderer, 26
 - set_camera, 28
 - shading_programs, 29
 - tessellation_shaders, 29
 - update, 28
 - vertex_shaders, 29
- Rendering/Camera/Camera.cpp, 49
- Rendering/Camera/Camera.hpp, 49
- Rendering/OpenGL-Wrappers.cpp, 50
- Rendering/OpenGL-Wrappers.hpp, 51
- Rendering/Window-Management.cpp, 52
- Rendering/Window-Management.hpp, 54
- Rendering_Handler
 - OpenGL-Wrappers.cpp, 51
 - OpenGL-Wrappers.hpp, 52
- resetCamera
 - Camera, 8
- resetView
 - Camera, 8
- STB_IMAGE_IMPLEMENTATION
 - OpenGL-Wrappers.cpp, 50
- STB_IMAGE_WRITE_IMPLEMENTATION
 - OpenGL-Wrappers.cpp, 50
- send_render_data
 - Chunk, 13
- World, 35
- set_camera
 - Renderer, 28
- set_instance_data
 - Object_3D, 24
- setLookDirection
 - Camera, 8
- setPosition
 - Camera, 8
- Shader, 29
 - ~Shader, 30
 - clear, 31
 - fileName, 31
 - load_from_file, 31
 - Shader, 30
 - shaderID, 31
 - type, 31
- shaderID
 - Shader, 31
- shading_programs
 - Renderer, 29
- shift
 - Chunk_Holder, 15
 - cirArray, 17
- side
 - Camera, 10
- size
 - cirArray, 17
 - tools.cpp, 44
- start
 - cirArray, 18
- surflet
 - tools.cpp, 43
- system-libraries.hpp
 - GLEW_DYNAMIC, 42
- target
 - Texture, 33
- tessellation_shaders
 - Renderer, 29
- Texture, 32
 - ~Texture, 33
 - clear, 33
 - height, 33
 - load_to_GPU, 33
 - target, 33
 - Texture, 32
 - texture, 34
 - textureID, 34
 - width, 34
- texture
 - Texture, 34
- texture_source_files
 - Cube.cpp, 37
- textureID
 - Texture, 34
- textures
 - Cube, 20
- the_world

- World.cpp, 57
- World.hpp, 58
- tools.cpp
 - fade, 43
 - length, 43
 - mask, 44
 - noise_2D, 43
 - perlin_noise, 43
 - perm, 44
 - size, 44
 - surflet, 43
 - vec_field_init, 44
 - vec_field_x, 44
 - vec_field_y, 44
- tools.hpp
 - noise_2D, 45
 - vec_field_init, 45
- transparent
 - Cube, 20
- turnH
 - Camera, 8
- turnV
 - Camera, 8
- type
 - Shader, 31
- types
 - Object_3D, 24
- up
 - Camera, 10
- update
 - Chunk, 13
 - Cube, 20
 - Renderer, 28
- update_loop
 - main.cpp, 48
- update_visible_faces
 - Chunk, 13
- uvs
 - Mesh, 22
- v_radius
 - World, 36
- VAO
 - Object_3D, 24
- VBOs
 - Object_3D, 25
- vec_field_init
 - tools.cpp, 44
 - tools.hpp, 45
- vec_field_x
 - tools.cpp, 44
- vec_field_y
 - tools.cpp, 44
- vertex_shaders
 - Renderer, 29
- vertices
 - Mesh, 22
- wavefront-loader.cpp
 - load_obj, 46
- wavefront-loader.hpp
 - load_obj, 47
- width
 - Camera, 10
 - Texture, 34
- Window-Management.cpp
 - CAM_SPEED, 53
 - callBackInit, 53
 - create_context, 53
 - createWindow, 53
 - cursor_pos_callback, 53
 - cursorSelectNode, 53
 - error_callback, 54
 - key_callback, 54
 - mouse_button_callback, 54
 - openGLerror, 54
- Window-Management.hpp
 - calculateFPS, 55
 - callBackInit, 55
 - create_context, 55
 - createWindow, 55
 - cursor_pos_callback, 55
 - error_callback, 56
 - key_callback, 56
 - mouse_button_callback, 56
- World, 34
 - ~World, 35
 - center_frame, 35
 - h_radius, 36
 - loaded_chunks, 36
 - loadedLights, 36
 - operator(), 35
 - origin, 36
 - send_render_data, 35
 - v_radius, 36
 - World, 35
- world
 - Chunk, 14
 - Chunk_Holder, 15
- World.cpp, 56
 - MESH, 57
 - the_world, 57
- World.hpp, 57
 - CHUNK_DIMS, 58
 - the_world, 58
- world_duration
 - main.cpp, 48
- zFar
 - Camera, 10
- zNear
 - Camera, 10