

VoXen - the VoXel engine

0.1

Generated by Doxygen 1.8.3.1

Thu Mar 14 2013 19:55:47

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	_cam_thread_t Struct Reference	5
3.2	_oct_file_node Struct Reference	5
3.3	_oct_file_node_leaf Struct Reference	5
3.4	_stack_item Struct Reference	6
3.5	oct_stack_item Struct Reference	6
3.6	raw_data Struct Reference	6
3.7	VOX_box Struct Reference	6
3.8	VX_block Struct Reference	6
3.8.1	Detailed Description	7
3.9	VX_camera Struct Reference	7
3.9.1	Detailed Description	7
3.9.2	Member Data Documentation	7
3.9.2.1	buffer	7
3.9.2.2	data	7
3.9.2.3	destroy	8
3.9.2.4	draw	8
3.9.2.5	position	8
3.9.2.6	rg	8
3.9.2.7	rotation_matrix	8
3.9.2.8	surface	8
3.9.2.9	workers	8
3.10	VX_CL_device Struct Reference	9
3.11	VX_CL_program Struct Reference	9
3.12	VX_cube Struct Reference	9
3.12.1	Detailed Description	9

3.13	VX_dline3 Struct Reference	10
3.13.1	Detailed Description	10
3.14	VX_dpoint2 Struct Reference	10
3.14.1	Detailed Description	10
3.15	VX_dpoint3 Struct Reference	10
3.15.1	Detailed Description	10
3.16	VX_field Struct Reference	11
3.16.1	Detailed Description	11
3.16.2	Member Data Documentation	11
3.16.2.1	destroy	11
3.16.2.2	destroy_p	11
3.16.2.3	enter	11
3.16.2.4	peasants_count	11
3.16.2.5	t	11
3.16.2.6	workers	12
3.17	VX_fline3 Struct Reference	12
3.17.1	Detailed Description	12
3.18	VX_format Struct Reference	12
3.18.1	Detailed Description	12
3.18.2	Member Data Documentation	12
3.18.2.1	ARGB32	12
3.18.2.2	Asz	13
3.18.2.3	bit_pp	13
3.18.2.4	Bsz	13
3.18.2.5	byte_pp	13
3.18.2.6	colorkey	13
3.18.2.7	Gsz	13
3.18.2.8	Rsz	13
3.19	VX_fpoint2 Struct Reference	13
3.19.1	Detailed Description	14
3.20	VX_fpoint3 Struct Reference	14
3.20.1	Detailed Description	14
3.21	VX_iline2 Struct Reference	14
3.21.1	Detailed Description	14
3.22	VX_iline3 Struct Reference	14
3.22.1	Detailed Description	15
3.23	VX_ipoint2 Struct Reference	15
3.23.1	Detailed Description	15
3.24	VX_ipoint3 Struct Reference	15
3.24.1	Detailed Description	15

3.25	VX_light Struct Reference	15
3.25.1	Detailed Description	16
3.26	VX_machine Struct Reference	16
3.26.1	Member Data Documentation	16
3.26.1.1	custom_dta	16
3.26.1.2	init	16
3.26.1.3	make_surface	16
3.26.1.4	quit	17
3.26.1.5	redraw	17
3.27	VX_model Struct Reference	17
3.27.1	Detailed Description	17
3.27.2	Member Data Documentation	18
3.27.2.1	compile	18
3.27.2.2	data	18
3.27.2.3	dump	18
3.27.2.4	free	18
3.27.2.5	get_voxel	18
3.27.2.6	inspect	19
3.27.2.7	load	19
3.27.2.8	present	19
3.27.2.9	ray_voxel	19
3.27.2.10	set_voxel	20
3.28	VX_ms_block Struct Reference	20
3.28.1	Detailed Description	20
3.29	VX_OCT_header Struct Reference	20
3.30	VX_oct_info Struct Reference	20
3.30.1	Detailed Description	21
3.31	VX_oct_node Struct Reference	21
3.31.1	Detailed Description	21
3.32	VX_oct_node_leaf Struct Reference	21
3.32.1	Detailed Description	21
3.33	VX_peasant Struct Reference	22
3.33.1	Detailed Description	22
3.34	VX_promise Struct Reference	22
3.35	VX_raw Struct Reference	22
3.36	VX_rect Struct Reference	22
3.36.1	Detailed Description	23
3.37	VX_stream Struct Reference	23
3.38	VX_surface Struct Reference	23
3.38.1	Detailed Description	23

3.38.2	Member Data Documentation	24
3.38.2.1	custom_dta	24
3.38.2.2	free	24
3.38.2.3	get_pixel	24
3.38.2.4	h	24
3.38.2.5	lock_surface	24
3.38.2.6	native_surf	24
3.38.2.7	set_pixel	24
3.38.2.8	unlock_surface	25
3.38.2.9	w	25
3.39	vxl_header Struct Reference	25
4	File Documentation	27
4.1	camera.h File Reference	27
4.1.1	Typedef Documentation	27
4.1.1.1	VX_camera	27
4.1.2	Function Documentation	27
4.1.2.1	VX_camera_new	27
4.2	display.h File Reference	28
4.2.1	Macro Definition Documentation	28
4.2.1.1	VX_FULLSCREEN	28
4.2.2	Function Documentation	28
4.2.2.1	VX_init	28
4.2.2.2	VX_machine_default_init	29
4.2.3	Variable Documentation	29
4.2.3.1	VX_DRV_null	29
4.2.3.2	VX_DRV_sdl	29
4.2.3.3	VX_lib	29
4.3	lighting.h File Reference	29
4.3.1	Typedef Documentation	29
4.3.1.1	VX_light	29
4.3.2	Function Documentation	29
4.3.2.1	VX_ambient_light	29
4.3.2.2	VX_ambient_light_set	30
4.3.2.3	VX_light_dynamic_add	30
4.3.2.4	VX_light_remove	30
4.3.2.5	VX_lights_clear	30
4.3.2.6	VX_lights_count	30
4.3.2.7	VX_lights_enumerate	30
4.4	matrix.h File Reference	31

4.4.1	Function Documentation	31
4.4.1.1	VX_dvector3_normalize	31
4.4.1.2	VX_dvector4_multiply	31
4.4.1.3	VX_matrix4_rotate	31
4.4.1.4	VX_matrix4_rotated	31
4.5	octree.h File Reference	31
4.5.1	Typedef Documentation	32
4.5.1.1	VX_oct_info	32
4.5.1.2	VX_oct_node	32
4.5.1.3	VX_oct_node_leaf	32
4.5.2	Function Documentation	32
4.5.2.1	VX_model_octree_new	32
4.5.2.2	VX_oct_fill_region	32
4.5.2.3	VX_oct_set_size	33
4.6	peasant.h File Reference	33
4.6.1	Typedef Documentation	33
4.6.1.1	VX_field	33
4.6.1.2	VX_peasant	33
4.6.2	Function Documentation	33
4.6.2.1	VX_field_new	33
4.7	raw.h File Reference	34
4.7.1	Function Documentation	34
4.7.1.1	VX_model_raw_new	34
4.8	slab.h File Reference	34
4.8.1	Function Documentation	34
4.8.1.1	VX_cube_in_point_fpu	34
4.8.1.2	VX_cube_out_point	35
4.8.1.3	VX_cube_out_point_fpu	35
4.8.1.4	VX_permutation_inspect	35
4.8.2	Variable Documentation	36
4.8.2.1	VX_line_sort	36
4.9	voxel.h File Reference	36
4.9.1	Typedef Documentation	36
4.9.1.1	VX_model	36
4.10	vx_fundamental.h File Reference	36
4.10.1	Macro Definition Documentation	38
4.10.1.1	A	38
4.10.1.2	APPLY3	38
4.10.1.3	B	38
4.10.1.4	BOX_TO_MS_BLOCK	39

4.10.1.5	CHECK_INTERVAL	39
4.10.1.6	CROSS_PRODUCT	39
4.10.1.7	FLOAT_SIGNUM	39
4.10.1.8	G	39
4.10.1.9	MAX	39
4.10.1.10	MAX_ALLOCABLE	39
4.10.1.11	MIN	39
4.10.1.12	MS_BLOCK_TO_BOX	39
4.10.1.13	PI	39
4.10.1.14	PRINT_BLOCK	39
4.10.1.15	PRINT_CUBE	40
4.10.1.16	PRINT_MS_BLOCK	40
4.10.1.17	PRINT_POINT3	40
4.10.1.18	PRINT_RAW_FPOINT3	40
4.10.1.19	PRINT_RAW_POINT3	40
4.10.1.20	R	40
4.10.1.21	X	40
4.10.1.22	Y	40
4.10.1.23	Z	40
4.10.2	Typedef Documentation	40
4.10.2.1	VX_block	40
4.10.2.2	VX_cube	40
4.10.2.3	VX_dline3	40
4.10.2.4	VX_dpoint2	41
4.10.2.5	VX_dpoint3	41
4.10.2.6	VX_fline3	41
4.10.2.7	VX_format	41
4.10.2.8	VX_fpoint2	41
4.10.2.9	VX_fpoint3	41
4.10.2.10	VX_iline2	41
4.10.2.11	VX_iline3	41
4.10.2.12	VX_ipoint2	41
4.10.2.13	VX_ipoint3	41
4.10.2.14	VX_ms_block	41
4.10.2.15	VX_rect	41
4.10.2.16	VX_surface	42
4.10.3	Function Documentation	42
4.10.3.1	color_add	42
4.10.3.2	color_mul	42
4.10.3.3	deg_to_radians	42

4.10.3.4	fvector3_raw_normalize	42
4.10.3.5	nearest_pow2	43
4.10.3.6	rad_to_degrees	43
4.10.3.7	VX_cubes_intersection	43
4.10.3.8	VX_cubes_intersects_p	43
4.10.3.9	VX_fvector4_multiply	43
4.10.3.10	VX_matrix4_add	44
4.10.3.11	VX_matrix4_identity	44
4.10.3.12	VX_matrix4_multiply	44
4.10.3.13	VX_matrix4_print	44
4.10.3.14	VX_matrix4_xrotation	44
4.10.3.15	VX_matrix4_yrotation	45
4.10.3.16	VX_matrix4_zrotation	45
4.10.3.17	VX_point_in_cube_p	45
4.10.3.18	VX_vector4_print	45
4.11	VX_lib.h File Reference	45
4.11.1	Detailed Description	46
4.12	vxlfmt.h File Reference	46
4.12.1	Function Documentation	46
4.12.1.1	VX_vxlfmt_load	46

Chapter 1

Class Index

1.1 Class List

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

_cam_thread_t	5
_oct_file_node	5
_oct_file_node_leaf	5
_stack_item	6
oct_stack_item	6
raw_data	6
VOX_box	6
VX_block	6
VX_camera	7
VX_CL_device	9
VX_CL_program	9
VX_cube	9
VX_dline3	10
VX_dpoint2	10
VX_dpoint3	10
VX_field	11
VX_fline3	12
VX_format	12
VX_fpoint2	13
VX_fpoint3	14
VX_iline2	14
VX_iline3	14
VX_ipoint2	15
VX_ipoint3	15
VX_light	15
VX_machine	16
VX_model	17
VX_ms_block	20
VX_OCT_header	20
VX_oct_info	20
VX_oct_node	21
VX_oct_node_leaf	21
VX_peasant	22
VX_promise	22
VX_raw	22
VX_rect	22
VX_stream	23
VX_surface	23

vxl_header	25
----------------------------	-------	--------------------

Chapter 2

File Index

2.1 File List

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

camera.h	27
display.h	28
fpu_octree_traversal.h	??
lighting.h	29
matrix.h	31
octree.h	31
octree_traversal.h	??
peasant.h	33
raw.h	34
slab.h	34
stream.h	??
voxel.h	36
vx_fundamental.h	36
VX_lib.h	
VX_lib.h is the only one header file, which can be explicitly included in your project using the	
VoXen library	45
vxlfmt.h	46
vxocl.h	??

Chapter 3

Class Documentation

3.1 `_cam_thread_t` Struct Reference

Public Attributes

- `VX_model` * **m**
- `VX_camera` * **c**
- `VX_rect clip_rect`

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

- `camera.c`

3.2 `_oct_file_node` Struct Reference

Public Attributes

- `VX_uint32 flags`
- `VX_uint32 color`
- `VX_uint32 childs_offsets` [8]

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

- `octree.c`

3.3 `_oct_file_node_leaf` Struct Reference

Public Attributes

- `VX_uint32 flags`
- `VX_uint32 color`

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

- `octree.c`

3.4 `_stack_item` Struct Reference

Public Attributes

- `VX_cube` **b**
- `VX_oct_node` * **n**

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

- `octree_traversal.c`

3.5 `oct_stack_item` Struct Reference

Public Attributes

- float `cube_point` [4]
- `VX_oct_node` * **n**

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

- `fpu_octree_traversal_stck.c`

3.6 `raw_data` Struct Reference

Public Attributes

- `VX_uint32` **pitchw**
- `VX_uint32` **pitchd**

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

- `raw.c`

3.7 `VOX_box` Struct Reference

Public Attributes

- int **x**
- int **y**
- int **z**
- int **w**

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

- `octree_traversal.c`

3.8 `VX_block` Struct Reference

```
#include <vx_fundamental.h>
```


Public Attributes

- VX_int32 **x**
- VX_int32 **y**
- VX_int32 **z**
- VX_int32 **w**
- VX_int32 **h**
- VX_int32 **d**

3.8.1 Detailed Description

Structure representing cube in 3D $\langle x,y,z \rangle$ is left bottom near corner, $\langle w,h,d \rangle$ is width, height and depth of cube.

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

- [vx_fundamental.h](#)

3.9 VX_camera Struct Reference

```
#include <camera.h>
```

Public Attributes

- [VX_ipoint3](#) **position**
- double [rotation_matrix](#) [4][4]
- [VX_fpoint3](#)(* [rg](#))(struct [VX_camera](#) *self, VX_uint32 x, VX_uint32 y)
- [VX_surface](#) * **surface**
- [VX_field](#) * **workers**
- void(* [draw](#))(struct [VX_camera](#) *self, void *model)
- void(* [destroy](#))(struct [VX_camera](#) *self)
- [VX_fpoint3](#) * **buffer**
- void * **data**

3.9.1 Detailed Description

Structure representing camera in raytracing model.

3.9.2 Member Data Documentation

3.9.2.1 [VX_fpoint3](#)* [VX_camera::buffer](#)

Commonly used buffer for storing rays used by rg function

3.9.2.2 void* [VX_camera::data](#)

Other user data

3.9.2.3 void(* VX_camera::destroy)(struct VX_camera *self)

Frees camera object.

Parameters

<i>self</i>	
-------------	--

3.9.2.4 void(* VX_camera::draw)(struct VX_camera *self, void *model)

Runs raytracing for camera on model.

Parameters

<i>self</i>	
<i>model</i>	model to run raytracing on

3.9.2.5 VX_ipoint3 VX_camera::position

holds position of camera

3.9.2.6 VX_fpoint3(* VX_camera::rg)(struct VX_camera *self, VX_uint32 x, VX_uint32 y)

Function that generates rays for given x and y coordinates

Parameters

<i>self</i>	
<i>x</i>	x-position of camera
<i>y</i>	y-position of camera

Returns

transformed [VX_fpoint3](#) by rotation matrix for <x,y> point.

3.9.2.7 double VX_camera::rotation_matrix[4][4]

stores 4x4 viewing matrix, last row and column should be 0

3.9.2.8 VX_surface* VX_camera::surface

surface output surface - framebuffer (read_only)

3.9.2.9 VX_field* VX_camera::workers

thread object [VX_field](#) (read_only)

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

- [camera.h](#)

3.10 VX_CL_device Struct Reference

Public Attributes

- VX_uint32 **exec_cores**
- cl_platform_id **platform**
- cl_context **context**
- cl_command_queue **queue**
- cl_device_id **device**
- void(* **release**)(struct [VX_CL_device](#) *)

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

- vxocl.h

3.11 VX_CL_program Struct Reference

Public Attributes

- [VX_CL_device](#) * **dev**
- cl_program **p**
- cl_kernel **ker**
- int **arg_top**
- int **cpu_blocking_p**
- sem_t **block**
- const char * **entry_point**
- int(* **push_arg**)(struct [VX_CL_program](#) *, void *, size_t)
- int(* **enter**)(struct [VX_CL_program](#) *, VX_uint32 wi_count)
- int(* **release**)(struct [VX_CL_program](#) *)

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

- vxocl.h

3.12 VX_cube Struct Reference

```
#include <vx_fundamental.h>
```

Public Attributes

- VX_int32 **x**
- VX_int32 **y**
- VX_int32 **z**
- VX_int32 **w**

3.12.1 Detailed Description

Structure representing cube in 3D, <x,y,z> is left bottom near corner, w is width.

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

- [vx_fundamental.h](#)

3.13 VX_dline3 Struct Reference

```
#include <vx_fundamental.h>
```

Public Attributes

- [VX_dpoint3](#) p1
- [VX_dpoint3](#) p2

3.13.1 Detailed Description

Structure representing double line in 3D.

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

- [vx_fundamental.h](#)

3.14 VX_dpoint2 Struct Reference

```
#include <vx_fundamental.h>
```

Public Attributes

- double x
- double y

3.14.1 Detailed Description

Structure representing double point in 2D.

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

- [vx_fundamental.h](#)

3.15 VX_dpoint3 Struct Reference

```
#include <vx_fundamental.h>
```

Public Attributes

- double x
- double y
- double z

3.15.1 Detailed Description

Structure representing double point in 3D.

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

- [vx_fundamental.h](#)

3.16 VX_field Struct Reference

```
#include <peasant.h>
```

Public Attributes

- int [peasants_count](#)
- int [destroy_p](#)
- pthread_t * [t](#)
- [VX_peasant](#) * [workers](#)
- void(* [enter](#))(struct [VX_field](#) *, void **params, void(*funcs[])(void *))
- void(* [destroy](#))(struct [VX_field](#) *)

3.16.1 Detailed Description

Structure holding set of threads

3.16.2 Member Data Documentation

3.16.2.1 void(* VX_field::destroy)(struct VX_field *)

Standard destructor of [VX_field](#), stops all threads and frees them

Parameters

<i>self</i>	analogy to this pointer
-------------	-------------------------

3.16.2.2 int VX_field::destroy_p

is private and should not be changed

3.16.2.3 void(* VX_field::enter)(struct VX_field *, void **params, void(*funcs[])(void *))

Executes all threads, this call blocks calling thread until done

Parameters

<i>self</i>	this pointer analogy
<i>params</i>	array of parameters, must be same the same size as peasants_count
<i>funcs</i>	array of functions, that returns void and takes one void* parameter

3.16.2.4 int VX_field::peasants_count

information about count of threads

3.16.2.5 pthread_t* VX_field::t

is array of threads and should not be changed

3.16.2.6 `VX_peasant*` `VX_field::workers`

workers array of threads, should not be changed

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

- [peasant.h](#)

3.17 `VX_fline3` Struct Reference

```
#include <vx_fundamental.h>
```

Public Attributes

- `VX_fpoint3` `p1`
- `VX_fpoint3` `p2`

3.17.1 Detailed Description

Structure representing float line in 3D.

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

- [vx_fundamental.h](#)

3.18 `VX_format` Struct Reference

```
#include <vx_fundamental.h>
```

Public Attributes

- `VX_byte` `bit_pp`
- `VX_byte` `byte_pp`
- `VX_uint32` `Rsz`
- `VX_uint32` `Gsz`
- `VX_uint32` `Bsz`
- `VX_uint32` `Asz`
- `VX_uint32` `colorkey`
- `VX_uint32`(`* ARGB32`)(`struct VX_format *`, `VX_byte *ptr`)

3.18.1 Detailed Description

Structure representing color format.

3.18.2 Member Data Documentation

3.18.2.1 `VX_uint32`(`* VX_format::ARGB32`)(`struct VX_format *`, `VX_byte *ptr`)

Conversion function, from this format to ARGB32 standard format, this function must be set at least once by user.

Parameters

<i>self</i>	this pointer
<i>ptr</i>	pointer to color

Returns

valid ARGB32 color

3.18.2.2 VX_uint32 VX_format::Asz

Asz count of bits used for alpha

3.18.2.3 VX_byte VX_format::bit_pp

count of bits used by one pixel.

3.18.2.4 VX_uint32 VX_format::Bsz

Bsz count of bits used for blue

3.18.2.5 VX_byte VX_format::byte_pp

count of bytes used by pixel

3.18.2.6 VX_uint32 VX_format::colorkey

colorkey is colorkey value used as transparent color

3.18.2.7 VX_uint32 VX_format::Gsz

count of bits used for green

3.18.2.8 VX_uint32 VX_format::Rsz

count of bits used for red

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

- [vx_fundamental.h](#)

3.19 VX_fpoint2 Struct Reference

```
#include <vx_fundamental.h>
```

Public Attributes

- float **x**
- float **y**

3.19.1 Detailed Description

Structure representing float point in 2D.

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

- [vx_fundamental.h](#)

3.20 VX_fpoint3 Struct Reference

```
#include <vx_fundamental.h>
```

Public Attributes

- float **x**
- float **y**
- float **z**

3.20.1 Detailed Description

Structure representing float point in 3D.

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

- [vx_fundamental.h](#)

3.21 VX_iline2 Struct Reference

```
#include <vx_fundamental.h>
```

Public Attributes

- [VX_ipoint2](#) **p1**
- [VX_ipoint2](#) **p2**

3.21.1 Detailed Description

Structure representing integer line in 2D.

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

- [vx_fundamental.h](#)

3.22 VX_iline3 Struct Reference

```
#include <vx_fundamental.h>
```

Public Attributes

- [VX_ipoint3](#) **p1**
- [VX_ipoint3](#) **p2**

3.22.1 Detailed Description

Structure representing integer line in 3D.

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

- [vx_fundamental.h](#)

3.23 VX_ipoint2 Struct Reference

```
#include <vx_fundamental.h>
```

Public Attributes

- int **x**
- int **y**

3.23.1 Detailed Description

Structure representing integer point in 2D.

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

- [vx_fundamental.h](#)

3.24 VX_ipoint3 Struct Reference

```
#include <vx_fundamental.h>
```

Public Attributes

- int **x**
- int **y**
- int **z**

3.24.1 Detailed Description

Structure representing integer point in 3D.

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

- [vx_fundamental.h](#)

3.25 VX_light Struct Reference

```
#include <lighting.h>
```

Public Attributes

- float **pos** [3]
- VX_uint32 **clr**

3.25.1 Detailed Description

Dynamic light structure

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

- [lighting.h](#)

3.26 VX_machine Struct Reference

Public Attributes

- `int(* init)(struct VX_machine *self, VX_rect resolution, VX_uint32 flags)`
- `int(* quit)(struct VX_machine *self)`
- `VX_surface *(* make_surface)(VX_uint32 w, VX_uint32 h)`
- `void(* redraw)(struct VX_machine *self)`
- `VX_rect resolution`
- `VX_uint32 flags`
- `VX_surface * native_surface`
- `int cores_count`
- `void * custom_dta`

3.26.1 Member Data Documentation

3.26.1.1 `void* VX_machine::custom_dta`

Custom data pointer, may be accessed only by programmer of driver

3.26.1.2 `int(* VX_machine::init)(struct VX_machine *self, VX_rect resolution, VX_uint32 flags)`

Initialize `VX_machine` self with resolution given by resolution and with flags.

Parameters

<i>self</i>	
<i>resolution</i>	rectangle where is important only information about width and height, x and y values are ignored
<i>flags</i>	is set of possible VoXen flags (commonly <code>VX_FULLSCREEN</code>)

Returns

returns 0 on success, not 0 on fail, some implementations may exit with -1

3.26.1.3 `VX_surface *(* VX_machine::make_surface)(VX_uint32 w, VX_uint32 h)`

Allocates new device-independent surface with size w,h and format ARGB32.

Parameters

<i>w</i>	width of image
<i>h</i>	height of image

Returns

valid [VX_surface](#) pointer on success, NULL when failed.

3.26.1.4 `int(* VX_machine::quit)(struct VX_machine *self)`

Quits machine self and does all needed frees and deallocations.

Parameters

<i>self</i>	
-------------	--

Returns

0 on success, not 0 on fail

3.26.1.5 `void(* VX_machine::redraw)(struct VX_machine *self)`

Redraws framebuffer of the machine.

Parameters

<i>self</i>	
-------------	--

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

- [display.h](#)

3.27 VX_model Struct Reference

```
#include <voxel.h>
```

Public Attributes

- VX_node **root**
- void * **chunk_ptr**
- [VX_block](#) **dim_size**
- [VX_format](#) * **fmt**
- int(* **compile**)(struct [VX_model](#) *self, struct [VX_model](#) *m)
- int(* **dump**)(struct [VX_model](#) *self, const char *file)
- void(* **inspect**)(struct [VX_model](#) *self)
- VX_byte *(**get_voxel**)(struct [VX_model](#) *self, VX_uint32 x, VX_uint32 y, VX_uint32 z, int *lod)
- void(* **set_voxel**)(struct [VX_model](#) *self, VX_uint32 x, VX_uint32 y, VX_uint32 z, VX_uint32 color)
- VX_uint32(* **ray_voxel**)(struct [VX_model](#) *self, float *position, float *norm_vect, float *end_point, int lod)
- void(* **present**)(struct [VX_model](#) *self, [VX_camera](#) *, [VX_rect](#) clip_rect)
- int(* **load**)(struct [VX_model](#) *self, const char *path)
- void(* **free**)(struct [VX_model](#) *self)
- void * **data**

3.27.1 Detailed Description

[VX_model](#) is general interface for implementing various models and doing raytracing on them.

3.27.2 Member Data Documentation

3.27.2.1 `int(* VX_model::compile)(struct VX_model *self, struct VX_model *m)`

Compiles model *m* into model *self* (meaning is to copy one model to other - the models can be different type).

Parameters

<i>self</i>	
<i>m</i>	arbitrary model to compile into <i>self</i>

Returns

0 on success, not 0 on fail

3.27.2.2 `void* VX_model::data`

User data

3.27.2.3 `int(* VX_model::dump)(struct VX_model *self, const char *file)`

Saves model *self* into file named *file*.

Parameters

<i>self</i>	
<i>path</i>	to file which will be created or rewritten.

Returns

0 on success, not 0 on fail

3.27.2.4 `void(* VX_model::free)(struct VX_model *self)`

Correctly frees memory allocated by model.

Parameters

<i>self</i>	
-------------	--

3.27.2.5 `VX_byte*(* VX_model::get_voxel)(struct VX_model *self, VX_uint32 x, VX_uint32 y, VX_uint32 z, int *lod)`

Obtains pointer to color of voxel placed on position <*x*,*y*,*z*>

Parameters

<i>self</i>	
<i>x</i>	
<i>y</i>	
<i>z</i>	
<i>lod</i>	input/output specifies level of detail, in some models (raw) must not be implemented, but for other models should be big enough. Output is then <i>*lod</i> minus (how much levels was traversed).

Returns

pointer to color

3.27.2.6 void(* VX_model::inspect)(struct VX_model *self)

Prints some information and stats about model on the stdout

Parameters

<i>self</i>	
-------------	--

3.27.2.7 int(* VX_model::load)(struct VX_model *self, const char *path)

Loads model from file path.

Parameters

<i>self</i>	
<i>path</i>	path to a file

Returns

0 on success, not 0 on fail

3.27.2.8 void(* VX_model::present)(struct VX_model *self, VX_camera *, VX_rect clip_rect)

Doing raytracing for camera on model self and on area specified as clip_rect. This method may be replaced any other well implementing presenting process (For example OpenCL implementation).

Parameters

<i>self</i>	
<i>camera</i>	
<i>clip_rect</i>	specifies area of framebuffer which should be redrawn

3.27.2.9 VX_uint32(* VX_model::ray_voxel)(struct VX_model *self, float *position, float *norm_vect, float *end_point, int lod)

Doing ray trace for ray, that starts in position, with vector norm_vect and with lod limit.

Parameters

<i>self</i>	
<i>position</i>	ray starting point
<i>norm_vect</i>	direction of ray
<i>end_point</i>	unused
<i>lod</i>	level of detail, may restrict using small voxels (quality drop occurs then)

Returns

color in ARGB32 format

3.27.2.10 `void(* VX_model::set_voxel)(struct VX_model *self, VX_uint32 x, VX_uint32 y, VX_uint32 z, VX_uint32 color)`

Sets voxel at <x,y,z> with color.

Parameters

<i>self</i>	
<i>x</i>	
<i>y</i>	
<i>z</i>	
<i>color</i>	

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

- [voxel.h](#)

3.28 VX_ms_block Struct Reference

```
#include <vx_fundamental.h>
```

Public Attributes

- VX_int32 **x1**
- VX_int32 **y1**
- VX_int32 **z1**
- VX_int32 **x2**
- VX_int32 **y2**
- VX_int32 **z2**

3.28.1 Detailed Description

Structure representing block in 3D in Microsoft way (like RECT from WinAPI) <x1,y1,z1> is minimal corner and <x2,y2,z2> is maximal corner.

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

- [vx_fundamental.h](#)

3.29 VX_OCT_header Struct Reference

Public Attributes

- [VX_block](#) **dim_size**
- VX_uint32 **control_size**

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

- [octree.c](#)

3.30 VX_oct_info Struct Reference

```
#include <octree.h>
```

Public Attributes

- VX_uint32 **stack_len**
- VX_uint32 **item_len**
- VX_uint32 **buffer_size**

3.30.1 Detailed Description

Private undocumented structure holding information about octree

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

- [octree.h](#)

3.31 VX_oct_node Struct Reference

```
#include <octree.h>
```

Public Attributes

- VX_uint32 **flags**
- VX_uint32 **color**
- struct [VX_oct_node](#) * **childs** [8]

3.31.1 Detailed Description

Private undocumented structure holding octree node

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

- [octree.h](#)

3.32 VX_oct_node_leaf Struct Reference

```
#include <octree.h>
```

Public Attributes

- VX_uint32 **flags**
- VX_uint32 **color**

3.32.1 Detailed Description

Private undocumented structure holding leaf octree node

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

- [octree.h](#)

3.33 VX_peasant Struct Reference

```
#include <peasant.h>
```

Public Attributes

- sem_t **sem_in**
- sem_t **sem_out**
- void(* **plant**)(void *)
- void * **param**
- int **destroy_p**

3.33.1 Detailed Description

Undocumented structure for almost inner purposes

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

- [peasant.h](#)

3.34 VX_promise Struct Reference

Public Attributes

- void(* **func_call**)(void *shared_args, void *user_args, void *ret)
- void * **args**

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

- [stream.h](#)

3.35 VX_raw Struct Reference

Public Attributes

- VX_byte * **chunk**
- [VX_block](#) **bounding_box**
- [VX_format](#) **fmt**
- VX_uint32 **pitchd**
- VX_uint32 **pitchw**

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

- [raw.c](#)

3.36 VX_rect Struct Reference

```
#include <vx_fundamental.h>
```


Public Attributes

- VX_int32 **x**
- VX_int32 **y**
- VX_int32 **w**
- VX_int32 **h**

3.36.1 Detailed Description

Structure representing a 2D rectangle, <x,y> is left-top corner and w is width and h is height.

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

- [vx_fundamental.h](#)

3.37 VX_stream Struct Reference

Public Attributes

- [VX_promise](#) ** **stack_promises**
- int **stack_top**
- int **stack_max_size**
- void(* **call**)(struct [VX_stream](#) *self, void *shared_args, void *ret)
- void(* **promise**)(struct [VX_stream](#) *self, [VX_promise](#) *p)

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

- stream.h

3.38 VX_surface Struct Reference

```
#include <vx_fundamental.h>
```

Public Attributes

- void(* **free**)(struct [VX_surface](#) *)
- VX_uint32(* **get_pixel**)(struct [VX_surface](#) *, VX_uint32 x, VX_uint32 y)
- void(* **lock_surface**)(struct [VX_surface](#) *)
- void(* **unlock_surface**)(struct [VX_surface](#) *)
- void(* **set_pixel**)(struct [VX_surface](#) *self, VX_uint32 x, VX_uint32 y, VX_uint32 color)
- VX_uint32 **w**
- VX_uint32 **h**
- void * **native_surf**
- void * **custom_dta**

3.38.1 Detailed Description

2D surface device independent interface.

3.38.2 Member Data Documentation

3.38.2.1 void* VX_surface::custom_dta

pointer for any other data used by driver, user may modify this only if writing driver

3.38.2.2 void(* VX_surface::free)(struct VX_surface *)

Destructor of surface.

Parameters

<i>self</i>	
-------------	--

3.38.2.3 VX_uint32(* VX_surface::get_pixel)(struct VX_surface *, VX_uint32 x, VX_uint32 y)

Returns pixel in ARGB32 format from x, y position.

Parameters

<i>self</i>	
<i>x</i>	x position of pixel
<i>y</i>	y position of pixel

Returns

ARGB32 color

3.38.2.4 VX_uint32 VX_surface::h

height of surface

3.38.2.5 void(* VX_surface::lock_surface)(struct VX_surface *)

Locks surface for writing, returns raw pointer to image data.

Parameters

<i>self</i>	
-------------	--

Returns

raw pointer to image data

3.38.2.6 void* VX_surface::native_surf

pointer to device dependent surface, user may modify this only if writing driver

3.38.2.7 void(* VX_surface::set_pixel)(struct VX_surface *self, VX_uint32 x, VX_uint32 y, VX_uint32 color)

Sets pixel to color (ARGB32 format) on x, y position.

Parameters

<i>self</i>	
<i>x</i>	x position of pixel
<i>y</i>	y position of pixel
<i>color</i>	valid ARGB32 color

3.38.2.8 void(* VX_surface::unlock_surface)(struct VX_surface *)

Unlocks surface for writing, returns raw pointer to image data.

Parameters

<i>self</i>	
-------------	--

3.38.2.9 VX_uint32 VX_surface::w

width of surface

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

- [vx_fundamental.h](#)

3.39 vxl_header Struct Reference

Public Attributes

- VX_int32 **magic_number**
- VX_int32 **width**
- VX_int32 **height**
- [VX_dpoint3](#) **cam**
- [VX_dpoint3](#) **right_vect**
- [VX_dpoint3](#) **down_vect**
- [VX_dpoint3](#) **forward_vect**

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

- vox_fmt.c

Chapter 4

File Documentation

4.1 camera.h File Reference

Classes

- struct [VX_camera](#)

Typedefs

- typedef struct [VX_camera](#) [VX_camera](#)

Functions

- [VX_camera](#) * [VX_camera_new](#) ([VX_surface](#) *surf, [VX_fpoint3](#)(*ray_generator)([VX_camera](#) *self, [VX_uint32](#) x, [VX_uint32](#) y), int cpus_count)

4.1.1 Typedef Documentation

4.1.1.1 typedef struct [VX_camera](#) [VX_camera](#)

Structure representing camera in raytracing model.

4.1.2 Function Documentation

4.1.2.1 [VX_camera](#)* [VX_camera_new](#) ([VX_surface](#) * *surf*, [VX_fpoint3](#)(*)([VX_camera](#) *self, [VX_uint32](#) x, [VX_uint32](#) y) *ray_generator*, int *cpus_count*)

Constructor for [VX_camera](#) object.

Parameters

<i>surf</i>	some surface of type VX_surface , can not be NULL, otherwise undefined behavior occurs
<i>ray_generator</i>	function used for generating rays, may be NULL, then standard perspective correction will be used
<i>cpus_count</i>	specifying number of threads for raytracing, when value is less than 1 is passed, then camera will create as same threads as your CPU has, otherwise the passed value will be used

Returns

valid [VX_camera](#) pointer on success, NULL on fail

4.2 display.h File Reference

Classes

- struct [VX_machine](#)

Macros

- `#define VX_FULLSCREEN 0x1`

Typedefs

- typedef struct [VX_machine](#) **VX_machine**

Functions

- int [VX_init](#) ([VX_machine](#) driver, VX_uint32 w, VX_uint32 h, VX_uint32 flags)
- void [VX_machine_default_init](#) ([VX_machine](#) *m)

Variables

- [VX_machine](#) [VX_DRV_sdl](#)
- [VX_machine](#) [VX_DRV_null](#)
- [VX_machine](#) [VX_lib](#)

4.2.1 Macro Definition Documentation

4.2.1.1 `#define VX_FULLSCREEN 0x1`

Flag specifying if initialize with fullscreen

4.2.2 Function Documentation

4.2.2.1 int [VX_init](#) ([VX_machine](#) driver, VX_uint32 w, VX_uint32 h, VX_uint32 flags)

Initialize driver and set it as main object ([VX_lib](#)).

Parameters

<i>driver</i>	may be arbitrary structure of type VX_machine , common use is VX_DRV_sdl or VX_DRV_null or any other self written driver.
<i>w</i>	width of framebuffer
<i>h</i>	height of framebuffer
<i>flags</i>	flags for machine, commonly 0 or VX_FULLSCREEN

4.2.2.2 void `VX_machine_default_init (VX_machine * m)`

Undocumented inner function

4.2.3 Variable Documentation

4.2.3.1 `VX_machine VX_DRV_null`

Predefined 2D driver using raw `VX_uint32` arrays as its backend.

4.2.3.2 `VX_machine VX_DRV_sdl`

Predefined 2D driver using SDL (Simple direct layer) as its backend.

4.2.3.3 `VX_machine VX_lib`

Global main object of library, which operates with machine.

4.3 lighting.h File Reference

Classes

- struct `VX_light`

Typedefs

- typedef struct `VX_light VX_light`

Functions

- int `VX_light_dynamic_add (VX_light *l)`
- void `VX_light_remove (int idx)`
- void `VX_lights_clear ()`
- `VX_uint32 VX_lights_count ()`
- `VX_light * VX_lights_enumerate ()`
- void `VX_ambient_light_set (VX_uint32 mask)`
- `VX_uint32 VX_ambient_light ()`

4.3.1 Typedef Documentation

4.3.1.1 typedef struct `VX_light VX_light`

Dynamic light structure

4.3.2 Function Documentation

4.3.2.1 `VX_uint32 VX_ambient_light ()`

Inspects current ambient light.

Returns

ARB32 color of ambient light

4.3.2.2 void VX_ambient_light_set (VX_uint32 mask)

Sets ambient color for current scene (default is 0xffffffff)

Parameters

<i>mask</i>	color of ambient light in ARGB32 format
-------------	---

4.3.2.3 int VX_light_dynamic_add (VX_light * l)

Adds a dynamic light to scene.

Parameters

<i>l</i>	valid pointer to light, invalid may lead to unexpected behavior of raytracing (NULL is also invalid pointer)
----------	--

Returns

index/identifier of light

4.3.2.4 void VX_light_remove (int idx)

Removes dynamic light from scene identified by idx.

Parameters

<i>idx</i>	identifier of light, if invalid, no light will be removed.
------------	--

4.3.2.5 void VX_lights_clear ()

Removes all lights in scene.

4.3.2.6 VX_uint32 VX_lights_count ()

Obtains number of lights in scene.

Returns

count of lights

4.3.2.7 VX_light* VX_lights_enumerate ()

Enumerates all dynamic lights in scene by returning array of lights

Returns

array of lights, NULL will be returned iff [VX_lights_count\(\)](#) returns 0;

4.4 matrix.h File Reference

Functions

- void [VX_dvector3_normalize](#) (double inout[3])
- void [VX_dvector4_multiply](#) (double outv[4], double matrix[4][4], double vector[4])
- void [VX_matrix4_rotate](#) (double out[4][4], float v[3], float angle)
- void [VX_matrix4_rotated](#) (double out[4][4], double v[3], double angle)
- void [VX_matrix4_lookat](#) (double out[4][4], float at[3], float pos[3], float up[3])

4.4.1 Function Documentation

4.4.1.1 void VX_dvector3_normalize (double *inout*[3])

Normalize inout double raw vector.

Parameters

<i>inout</i>	double raw vector, after execution it is the same vector of length 1.0f
--------------	---

4.4.1.2 void VX_dvector4_multiply (double *outv*[4], double *matrix*[4][4], double *vector*[4])

Multiplies matrix matrix with vector in double precision, like this matrix*vector and stores output in outv.

Parameters

<i>outv</i>	output vector in format float[4]
<i>matrix</i>	matrix in format double[4][4]
<i>vector</i>	vector in format double[4]

4.4.1.3 void VX_matrix4_rotate (double *out*[4][4], float *v*[3], float *angle*)

Initialize out to a matrix of rotation around an arbitrary axis (single precision) v and angle.

Parameters

<i>out</i>	output matrix
<i>v</i>	arbitrary axis
<i>angle</i>	angle in radians

4.4.1.4 void VX_matrix4_rotated (double *out*[4][4], double *v*[3], double *angle*)

Initialize out to a matrix of rotation around an arbitrary axis (double precision) v and angle.

Parameters

<i>out</i>	output matrix
<i>v</i>	arbitrary axis
<i>angle</i>	angle in radians

4.5 octree.h File Reference

Classes

- struct [VX_oct_node](#)
- struct [VX_oct_node_leaf](#)
- struct [VX_oct_info](#)

Typedefs

- typedef struct [VX_oct_node](#) [VX_oct_node](#)
- typedef struct [VX_oct_node_leaf](#) [VX_oct_node_leaf](#)
- typedef struct [VX_oct_info](#) [VX_oct_info](#)

Functions

- int [VX_oct_set_size](#) ([VX_model](#) *self, int size)
- int [VX_oct_fill_region](#) ([VX_model](#) *self, [VX_ms_block](#) b, [VX_uint32](#) color)
- [VX_model](#) * [VX_model_octree_new](#) ()

4.5.1 Typedef Documentation

4.5.1.1 typedef struct [VX_oct_info](#) [VX_oct_info](#)

Private undocumented structure holding information about octree

4.5.1.2 typedef struct [VX_oct_node](#) [VX_oct_node](#)

Private undocumented structure holding octree node

4.5.1.3 typedef struct [VX_oct_node_leaf](#) [VX_oct_node_leaf](#)

Private undocumented structure holding leaf octree node

4.5.2 Function Documentation

4.5.2.1 [VX_model](#)* [VX_model_octree_new](#) ()

Constructor of octree model format, color format can not be modified and is set to ARGB32.

Returns

valid [VX_model](#) pointer (when malloc can not fail - on UNIX)

4.5.2.2 int [VX_oct_fill_region](#) ([VX_model](#) * *self*, [VX_ms_block](#) *b*, [VX_uint32](#) *color*)

Additional function for octree for fast filling of large regions.

Parameters

<i>self</i>	
<i>b</i>	block to fill
<i>color</i>	color to fill region b with

4.5.2.3 int VX_oct_set_size (VX_model * self, int size)

Additional function that modifies size of valid octree.

Parameters

<i>self</i>	
<i>size</i>	size of all edges

Returns

1

4.6 peasant.h File Reference

Classes

- struct [VX_peasant](#)
- struct [VX_field](#)

Typedefs

- typedef struct [VX_peasant](#) [VX_peasant](#)
- typedef struct [VX_field](#) [VX_field](#)

Functions

- [VX_field](#) * [VX_field_new](#) (VX_uint32 peasants_count)
Constructor that creates peasants_count-1 threads, one job runs in calling thread.

4.6.1 Typedef Documentation

4.6.1.1 typedef struct VX_field VX_field

Structure holding set of threads

4.6.1.2 typedef struct VX_peasant VX_peasant

Undocumented structure for almost inner purposes

4.6.2 Function Documentation

4.6.2.1 VX_field* VX_field_new (VX_uint32 peasants_count)

Constructor that creates peasants_count-1 threads, one job runs in calling thread.

Parameters

<i>peasants_count</i>	is not 0 argument, specifying count of threads
-----------------------	--

Returns

returns [VX_field](#) pointer, or NULL if error occurred or peasants_count is 0

4.7 raw.h File Reference

Functions

- [VX_model](#) * [VX_model_raw_new](#) ([VX_format](#) *fmt, [VX_block](#) size)

4.7.1 Function Documentation

4.7.1.1 [VX_model](#)* [VX_model_raw_new](#) ([VX_format](#) * *fmt*, [VX_block](#) *size*)

Constructor for raw models. Raw model has some limitations, the load argument has no impact, set_voxel and get_voxel should write only voxels in respect to model size (otherwise expect undefined behavior).

Parameters

<i>fmt</i>	valid pointer to VX_format , any other will lead to unexpected behavior
<i>size</i>	size of raw model, only w, h, d values will be used

Returns

valid [VX_model](#) pointer on success, NULL when fail

4.8 slab.h File Reference

Functions

- void [VX_permutation_inspect](#) ([VX_uint32](#) p)
- int [VX_cube_out_point](#) (int box_min[3], int box_max[3], [VX_fpoint3](#) *point, [VX_fpoint3](#) *ray, float out[3], [VX_byte](#) line_perm[3])
- int [VX_cube_out_point_fpu](#) (float box_min[3], float box_max[3], [VX_fpoint3](#) *point, [VX_fpoint3](#) *ray, float out[3], [VX_byte](#) line_perm[3])
- int [VX_cube_in_point_fpu](#) (float box_min[3], float box_max[3], float point[3], float ray[3], float out[3], [VX_byte](#) line_perm[3])

Variables

- [VX_uint32](#)(* [VX_line_sort](#))([VX_fpoint3](#) dir)

4.8.1 Function Documentation

4.8.1.1 int [VX_cube_in_point_fpu](#) (float *box_min*[3], float *box_max*[3], float *point*[3], float *ray*[3], float *out*[3], [VX_byte](#) *line_perm*[3])

Computes input point for a ray to a cube defined by box_min and box_max points in floating point arithmetics

Parameters

<i>box_min</i>	integer raw point representing minimal box point
<i>box_max</i>	integer raw point representing maximal box point
<i>point</i>	float raw point representing position of observer
<i>ray</i>	float raw direction of ray
<i>out</i>	output raw float point
<i>line_perm</i>	sorting permutation of ray given by VX_line_sort

Returns

-1 if no intersection found, 0-2 which is index in which intersection occurred

4.8.1.2 `int VX_cube_out_point (int box_min[3], int box_max[3], VX_fpoint3 * point, VX_fpoint3 * ray, float out[3], VX_byte line_perm[3])`

Computes output point for a ray from cube defined by *box_min* and *box_max* points in mostly integer arithmetics

Parameters

<i>box_min</i>	integer raw point representing minimal box point
<i>box_max</i>	integer raw point representing maximal box point
<i>point</i>	float raw point representing position of observer
<i>ray</i>	float raw direction of ray
<i>out</i>	output raw float point
<i>line_perm</i>	sorting permutation of ray given by VX_line_sort

Returns

-1 if no intersection found, 0-2 which is index in which intersection occurred

4.8.1.3 `int VX_cube_out_point_fpu (float box_min[3], float box_max[3], VX_fpoint3 * point, VX_fpoint3 * ray, float out[3], VX_byte line_perm[3])`

Computes output point for a ray from cube defined by *box_min* and *box_max* points in floating point arithmetics

Parameters

<i>box_min</i>	integer raw point representing minimal box point
<i>box_max</i>	integer raw point representing maximal box point
<i>point</i>	float raw point representing position of observer
<i>ray</i>	float raw direction of ray
<i>out</i>	output raw float point
<i>line_perm</i>	sorting permutation of ray given by VX_line_sort

Returns

-1 if no intersection found, 0-2 which is index in which intersection occurred

4.8.1.4 `void VX_permutation_inspect (VX_uint32 p)`

Prints information about sorting permutation to stdout

Parameters

<i>p</i>	a valid sorting permutation
----------	-----------------------------

4.8.2 Variable Documentation

4.8.2.1 `VX_uint32(* VX_line_sort)(VX_fpoint3 dir)`

Returns sorting permutation of floating point vector *dir*, the index with the biggest value will be in the third byte of returned value, the index with the second biggest value will be in the second byte, and index with the least value is in the first byte of returned value.

Parameters

<i>dir</i>	floating point 3D vector
------------	--------------------------

Returns

sorting permutation, where zero byte is index of the least value, byte 1 is index of the second biggest value and byte 2 holds the index with the biggest value.

4.9 voxel.h File Reference

Classes

- struct [VX_model](#)

Macros

- `#define VX_node void *`

Typedefs

- typedef struct [VX_model](#) [VX_model](#)

4.9.1 Typedef Documentation

4.9.1.1 typedef struct [VX_model](#) [VX_model](#)

[VX_model](#) is general interface for implementing various models and doing raytracing on them.

4.10 vx_fundamental.h File Reference

Classes

- struct [VX_rect](#)
- struct [VX_cube](#)
- struct [VX_block](#)
- struct [VX_ms_block](#)
- struct [VX_ipoint2](#)
- struct [VX_iline2](#)
- struct [VX_ipoint3](#)

- struct [VX_fpoint2](#)
- struct [VX_fpoint3](#)
- struct [VX_iline3](#)
- struct [VX_fline3](#)
- struct [VX_dpoint2](#)
- struct [VX_dpoint3](#)
- struct [VX_dline3](#)
- struct [VX_format](#)
- struct [VX_surface](#)

Macros

- `#define FAST_AND &`
- `#define FAST_OR |`
- `#define PI 3.14159265`
- `#define PRINT_POINT3(X) {printf("p(%f , %f , %f)\n" , (float)X.x ,(float)X.y,(float)X.z);}`
- `#define PRINT_RAW_POINT3(P) {printf("p_raw(%i , %i , %i)\n" , P[X] , P[Y] , P[Z]);}`
- `#define PRINT_RAW_FPOINT3(P) {printf("p_raw(%f , %f , %f)\n" , P[X] , P[Y] , P[Z]);}`
- `#define PRINT_MS_BLOCK(B) {printf("ms_block(%d , %d , %d , %d , %d , %d)\n" , B.x1 , B.y1 , B.z1 , B.x2 , B.y2 , B.z2);}`
- `#define PRINT_BLOCK(B) {printf("block(%d , %d , %d , %d , %d , %d)\n" , B.x , B.y , B.z , B.w , B.h , B.d);}`
- `#define BOX_TO_MS_BLOCK(BOX) (VX_ms_block){ BOX.x, BOX.y, BOX.z , BOX.x + BOX.w , BOX.y + BOX.h , BOX.z + BOX.d }`
- `#define MS_BLOCK_TO_BOX(BLO) (VX_block){ BLO.x1 , BLO.y1 , BLO.z1 , BLO.x2 - BLO.x1 , BLO.y2 - BLO.y1 , BLO.z2 - BLO.z1 }`
- `#define PRINT_CUBE(C) { printf("cube(%d , %d , %d , .w = %d)\n" , C.x , C.y , C.z , C.w); }`
- `#define CHECK_INTERVAL(INSIDE_P, LEFT, RIGHT) (INSIDE_P <= RIGHT && INSIDE_P >= LEFT)`
- `#define APPLY3(lvar, rvar, op)`
- `#define CROSS_PRODUCT(a, b, c)`
- `#define MAX_ALLOCABLE 0x80000000`
- `#define VX_int32 int32_t`
- `#define VX_int16 int16_t`
- `#define VX_byte uint8_t`
- `#define VX_uint32 uint32_t`
- `#define VX_uint64 uint64_t`
- `#define X 0`
- `#define Y 1`
- `#define Z 2`
- `#define A 3`
- `#define R 2`
- `#define G 1`
- `#define B 0`
- `#define MAX(x, y) (x > y ? x : y)`
- `#define MIN(x, y) (x < y ? x : y)`
- `#define FLOAT_SIGNUM(N) (*((VX_uint32*)((void*)&N)) & (0x1 << 31))`

Typedefs

- `typedef struct VX_rect VX_rect`
- `typedef struct VX_cube VX_cube`
- `typedef struct VX_block VX_block`
- `typedef struct VX_ms_block VX_ms_block`
- `typedef struct VX_ipoint2 VX_ipoint2`
- `typedef struct VX_iline2 VX_iline2`

- typedef struct [VX_ipoint3](#) [VX_ipoint3](#)
- typedef struct [VX_fpoint2](#) [VX_fpoint2](#)
- typedef struct [VX_fpoint3](#) [VX_fpoint3](#)
- typedef struct [VX_iline3](#) [VX_iline3](#)
- typedef struct [VX_fline3](#) [VX_fline3](#)
- typedef struct [VX_dpoint2](#) [VX_dpoint2](#)
- typedef struct [VX_dpoint3](#) [VX_dpoint3](#)
- typedef struct [VX_dline3](#) [VX_dline3](#)
- typedef struct [VX_format](#) [VX_format](#)
- typedef struct [VX_surface](#) [VX_surface](#)

Functions

- void [VX_matrix4_print](#) (double m[4][4])
- void [VX_vector4_print](#) (float v[4])
- void [VX_fvector4_multiply](#) (float outv[4], double matrix[4][4], float vector[4])
- void [VX_matrix4_multiply](#) (double out[4][4], double m1[4][4], double m2[4][4])
- void [VX_matrix4_add](#) (double out[4][4], double m1[4][4], double m2[4][4])
- void [VX_matrix4_identity](#) (double out[4][4])
- void [VX_matrix4_xrotation](#) (double out[4][4], double rad_angle)
- void [VX_matrix4_yrotation](#) (double out[4][4], double rad_angle)
- void [VX_matrix4_zrotation](#) (double out[4][4], double rad_angle)
- double [rad_to_degrees](#) (double rad)
- double [deg_to_radians](#) (double deg)
- VX_uint32 [nearest_pow2](#) (VX_uint32 a)
- int [VX_point_in_cube_p](#) (VX_ms_block b, [VX_ipoint3](#) p)
- int [VX_cubes_intersects_p](#) (VX_ms_block b1, VX_ms_block b2)
- VX_ms_block [VX_cubes_intersection](#) (VX_ms_block b1, VX_ms_block b2)
- void [fvector3_raw_normalize](#) (float v[3], float len)
- VX_uint32 [color_add](#) (VX_uint32 c1, VX_uint32 c2)
- VX_uint32 [color_mul](#) (VX_uint32 c1, VX_uint32 c2)

4.10.1 Macro Definition Documentation

4.10.1.1 `#define A 3`

Index A in 4-item color array

4.10.1.2 `#define APPLY3(lvar, rvar, op)`

Value:

```
lvar[X] op rvar[X]; lvar[Y] op rvar[Y]; lvar[Z] op rvar[Z] \
```

Applies operation op on all items of 3-item array lvar and rvar

4.10.1.3 `#define B 0`

Index B in 4-item color array

4.10.1.4 `#define BOX_TO_MS_BLOCK(BOX) (VX_ms_block){ BOX.x, BOX.y, BOX.z , BOX.x + BOX.w , BOX.y + BOX.h , BOX.z + BOX.d }`

Conversion [VX_block](#) to [VX_ms_block](#).

4.10.1.5 `#define CHECK_INTERVAL(INSIDE_P, LEFT, RIGHT) (INSIDE_P <= RIGHT && INSIDE_P >= LEFT)`

Checking if INSIDE_P lies inside interval [LEFT,RIGHT]

4.10.1.6 `#define CROSS_PRODUCT(a, b, c)`

Value:

```
(a)[0] = (b)[1] * (c)[2] - (c)[1] * (b)[2]; \
(a)[1] = (b)[2] * (c)[0] - (c)[2] * (b)[0]; \
(a)[2] = (b)[0] * (c)[1] - (c)[0] * (b)[1]; \
```

4.10.1.7 `#define FLOAT_SIGNUM(N) (((VX_uint32*)((void*)&N)) & (0x1 << 31))`

Faster signum by using type punning - returns 1 << 31 if N is negative 0 otherwise.

4.10.1.8 `#define G 1`

Index G in 4-item color array

4.10.1.9 `#define MAX(x, y) (x > y ? x : y)`

Maximum macro.

4.10.1.10 `#define MAX_ALLOCABLE 0x80000000`

Constant of maximal allocable bytes - 3GB

4.10.1.11 `#define MIN(x, y) (x < y ? x : y)`

Minimum macro.

4.10.1.12 `#define MS_BLOCK_TO_BOX(BLO) (VX_block){ BLO.x1 , BLO.y1 , BLO.z1 , BLO.x2 - BLO.x1 , BLO.y2 - BLO.y1 , BLO.z2 - BLO.z1 }`

Conversion [VX_ms_block](#) to [VX_block](#).

4.10.1.13 `#define PI 3.14159265`

Ludolf constant.

4.10.1.14 `#define PRINT_BLOCK(B) { printf("block(%d , %d , %d , %d , %d , %d)\n" , B.x , B.y , B.z , B.w , B.h , B.d); }`

Print a [VX_block](#) structure.

4.10.1.15 `#define PRINT_CUBE(C) { printf("cube(%d , %d , %d , .w = %d)\n" , C.x , C.y , C.z , C.w); }`

Print `VX_cube` structure.

4.10.1.16 `#define PRINT_MS_BLOCK(B) { printf("ms_block(%d , %d , %d , %d , %d , %d)\n" , B.x1 , B.y1 , B.z1 , B.x2 , B.y2 , B.z2); }`

Print a `VX_ms_block` structure.

4.10.1.17 `#define PRINT_POINT3(X) { printf("p(%f , %f , %f)\n" , (float)X.x , (float)X.y , (float)X.z); }`

Print `VX_fpoint3`.

4.10.1.18 `#define PRINT_RAW_FPOINT3(P) { printf("p_raw(%f , %f , %f)\n" , P[X] , P[Y] , P[Z]); }`

Print array of 3 floats.

4.10.1.19 `#define PRINT_RAW_POINT3(P) { printf("p_raw(%i , %i , %i)\n" , P[X] , P[Y] , P[Z]); }`

Print array of 3 integers.

4.10.1.20 `#define R 2`

Index R in 4-item color array

4.10.1.21 `#define X 0`

Index X in 3-item vector array

4.10.1.22 `#define Y 1`

Index Y in 3-item vector array

4.10.1.23 `#define Z 2`

Index Z in 3-item vector array

4.10.2 Typedef Documentation

4.10.2.1 `typedef struct VX_block VX_block`

Structure representing cube in 3D $\langle x,y,z \rangle$ is left bottom near corner, $\langle w,h,d \rangle$ is width, height and depth of cube.

4.10.2.2 `typedef struct VX_cube VX_cube`

Structure representing cube in 3D, $\langle x,y,z \rangle$ is left bottom near corner, w is width.

4.10.2.3 `typedef struct VX_dline3 VX_dline3`

Structure representing double line in 3D.

4.10.2.4 typedef struct VX_dpoint2 VX_dpoint2

Structure representing double point in 2D.

4.10.2.5 typedef struct VX_dpoint3 VX_dpoint3

Structure representing double point in 3D.

4.10.2.6 typedef struct VX_fline3 VX_fline3

Structure representing float line in 3D.

4.10.2.7 typedef struct VX_format VX_format

Structure representing color format.

4.10.2.8 typedef struct VX_fpoint2 VX_fpoint2

Structure representing float point in 2D.

4.10.2.9 typedef struct VX_fpoint3 VX_fpoint3

Structure representing float point in 3D.

4.10.2.10 typedef struct VX_iline2 VX_iline2

Structure representing integer line in 2D.

4.10.2.11 typedef struct VX_iline3 VX_iline3

Structure representing integer line in 3D.

4.10.2.12 typedef struct VX_ipoint2 VX_ipoint2

Structure representing integer point in 2D.

4.10.2.13 typedef struct VX_ipoint3 VX_ipoint3

Structure representing integer point in 3D.

4.10.2.14 typedef struct VX_ms_block VX_ms_block

Structure representing block in 3D in Microsoft way (like RECT from WinAPI) <x1,y1,z1> is minimal corner and <x2,y2,z2> is maximal corner.

4.10.2.15 typedef struct VX_rect VX_rect

Structure representing a 2D rectangle, <x,y> is left-top corner and w is width and h is height.

4.10.2.16 typedef struct VX_surface VX_surface

2D surface device independent interface.

4.10.3 Function Documentation

4.10.3.1 VX_uint32 color_add (VX_uint32 c1, VX_uint32 c2)

Adds two ARGB32 colors together by components, values are checked for not to overlap 255.

Parameters

<i>c1</i>	ARGB32 color
<i>c2</i>	ARGB32 color

Returns

ARGB32 color created by addition c1 and c2.

4.10.3.2 VX_uint32 color_mul (VX_uint32 c1, VX_uint32 c2)

Multiplies two ARGB32 colors together by components, values never overlaps 255 due to modular arithmetics.

Parameters

<i>c1</i>	ARGB32 color
<i>c2</i>	ARGB32 color

Returns

ARGB32 color created by addition c1 and c2.

4.10.3.3 double deg_to_radians (double deg)

Converts degrees to radians.

Parameters

<i>deg</i>	angle in degrees
------------	------------------

Returns

angle in radians

4.10.3.4 void fvector3_raw_normalize (float v[3], float len)

Makes vector v of length len

Parameters

<i>input/output</i>	vector
<i>len</i>	needed length of vector (commonly 1.0f)

4.10.3.5 `VX_uint32 nearest_pow2 (VX_uint32 a)`

Compute `log_2(a)` in integer precision.

Parameters

<code>a</code>	an unsigned integer
----------------	---------------------

Returns

`log_2(a)`

4.10.3.6 `double rad_to_degrees (double rad)`

Converts radians to degrees.

Parameters

<code>rad</code>	angle in radians
------------------	------------------

Returns

angle in degrees

4.10.3.7 `VX_ms_block VX_cubes_intersection (VX_ms_block b1, VX_ms_block b2)`

Returns block of intersection between two blocks b1 and b2, blocks b1 and b2 must overlap.

Parameters

<code>b1</code>	a valid VX_ms_block
<code>b2</code>	a valid VX_ms_block

Returns

block of intersection between b1 and b2.

4.10.3.8 `int VX_cubes_intersects_p (VX_ms_block b1, VX_ms_block b2)`

Test if two blocks b1 and b2 intersect.

Parameters

<code>b1</code>	valid VX_ms_block
<code>b2</code>	valid VX_ms_block

Returns

1 if intersection occurs, 0 otherwise.

4.10.3.9 `void VX_fvector4_multiply (float outv[4], double matrix[4][4], float vector[4])`

Multiplies matrix matrix with vector, like this `matrix*vector` and stores output in `outv`.

Parameters

<i>outv</i>	output vector in format float[4]
<i>matrix</i>	matrix in format double[4][4]
<i>vector</i>	vector in format float[4]

4.10.3.10 void VX_matrix4_add (double *out*[4][4], double *m1*[4][4], double *m2*[4][4])

Addition of two matrices m1 and m2 storing output to out addition is done in order $out = m1 + m2$.

Parameters

<i>out</i>	output matrix
<i>m1</i>	left matrix
<i>m2</i>	right matrix

4.10.3.11 void VX_matrix4_identity (double *out*[4][4])

Initialize matrix 4x4 out to identity matrix.

Parameters

<i>out</i>	output matrix
------------	---------------

4.10.3.12 void VX_matrix4_multiply (double *out*[4][4], double *m1*[4][4], double *m2*[4][4])

Multiplication of two matrices m1 and m2 storing output to out multiplication is done in order $out = m1 \cdot m2$.

Parameters

<i>out</i>	output matrix
<i>m1</i>	left matrix
<i>m2</i>	right matrix

4.10.3.13 void VX_matrix4_print (double *m*[4][4])

Print matrix on stdout.

Parameters

<i>m</i>	matrix in format double[4][4]
----------	-------------------------------

4.10.3.14 void VX_matrix4_xrotation (double *out*[4][4], double *rad_angle*)

Initialize matrix to a rotation matrix around x axis (left handed system).

Parameters

<i>out</i>	output matrix
<i>rad_angle</i>	angle of rotation in radians

4.10.3.15 void VX_matrix4_yrotation (double *out*[4][4], double *rad_angle*)

Initialize matrix to a rotation matrix around y axis (left handed system).

Parameters

<i>out</i>	output matrix
<i>rad_angle</i>	angle of rotation in radians

4.10.3.16 void VX_matrix4_zrotation (double *out*[4][4], double *rad_angle*)

Initialize matrix to a rotation matrix around z axis (left handed system).

Parameters

<i>out</i>	output matrix
<i>rad_angle</i>	angle of rotation in radians

4.10.3.17 int VX_point_in_cube_p (VX_ms_block *b*, VX_ipoint3 *p*)

Test if point *p* lies inside [VX_ms_block](#) *b*.

Parameters

<i>b</i>	a valid VX_ms_block
<i>p</i>	integer point in 3D

Returns

1 if *p* lies inside *b*, 0 otherwise.

4.10.3.18 void VX_vector4_print (float *v*[4])

Print vecotr on stdout.

Parameters

<i>v</i>	vector in format float[4]
----------	---------------------------

4.11 VX_lib.h File Reference

[VX_lib.h](#) is the only one header file, which can be explicitly included in your project using the VoXen library.

```

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <pthread.h>
#include <semaphore.h>
#include <CL/opencl.h>
#include "vx_fundamental.h"
#include "peasant.h"
#include "vxocl.h"
#include "slab.h"
#include "matrix.h"
#include "display.h"
#include "camera.h"
#include "voxel.h"
#include "lighting.h"
#include "octree.h"
#include "vxlfmt.h"

```

4.11.1 Detailed Description

[VX_lib.h](#) is the only one header file, which can be explicitly included in your project using the VoXen library.

4.12 vxlfmt.h File Reference

Functions

- `VX_int32` [VX_vxlfmt_load](#) ([VX_model](#) *m, const char *path)

4.12.1 Function Documentation

4.12.1.1 `VX_int32 VX_vxlfmt_load (VX_model * m, const char * path)`

Loader of Ken Silverman's .vxl format, thanks to Ken.

Parameters

<i>m</i>	destination model (octree will be very slow ~5-10 minutes!)
<i>path</i>	to a valid .vxl file

Returns

0 on success, not 0 when fail and let m unmodified

Index

[_cam_thread_t](#), 5
[_oct_file_node](#), 5
[_oct_file_node_leaf](#), 5
[_stack_item](#), 6

A

[vx_fundamental.h](#), 38
APPLY3
[vx_fundamental.h](#), 38
ARGB32
[VX_format](#), 12
Asz
[VX_format](#), 13

B

[vx_fundamental.h](#), 38
BOX_TO_MS_BLOCK
[vx_fundamental.h](#), 38

bit_pp

[VX_format](#), 13

Bsz

[VX_format](#), 13

buffer

[VX_camera](#), 7

byte_pp

[VX_format](#), 13

CHECK_INTERVAL

[vx_fundamental.h](#), 39

CROSS_PRODUCT

[vx_fundamental.h](#), 39

camera.h

[VX_camera](#), 27
[VX_camera_new](#), 27

color_add

[vx_fundamental.h](#), 42

color_mul

[vx_fundamental.h](#), 42

colorkey

[VX_format](#), 13

compile

[VX_model](#), 18

custom_dta

[VX_machine](#), 16
[VX_surface](#), 24

data

[VX_camera](#), 7
[VX_model](#), 18

deg_to_radians

[vx_fundamental.h](#), 42

destroy

[VX_camera](#), 7
[VX_field](#), 11

destroy_p

[VX_field](#), 11

display.h

[VX_DRV_null](#), 29
[VX_DRV_sdl](#), 29
[VX_FULLSCREEN](#), 28
[VX_init](#), 28
[VX_lib](#), 29
[VX_machine_default_init](#), 28

draw

[VX_camera](#), 8

dump

[VX_model](#), 18

enter

[VX_field](#), 11

FLOAT_SIGNUM

[vx_fundamental.h](#), 39

free

[VX_model](#), 18
[VX_surface](#), 24

fvector3_raw_normalize

[vx_fundamental.h](#), 42

G

[vx_fundamental.h](#), 39

get_pixel

[VX_surface](#), 24

get_voxel

[VX_model](#), 18

Gsz

[VX_format](#), 13

h

[VX_surface](#), 24

init

[VX_machine](#), 16

inspect

[VX_model](#), 19

lighting.h

[VX_ambient_light](#), 29
[VX_ambient_light_set](#), 30
[VX_light](#), 29
[VX_light_dynamic_add](#), 30

- VX_light_remove, 30
- VX_lights_clear, 30
- VX_lights_count, 30
- VX_lights_enumerate, 30
- load
 - VX_model, 19
- lock_surface
 - VX_surface, 24
- MAX
 - vx_fundamental.h, 39
- MAX_ALLOCABLE
 - vx_fundamental.h, 39
- MIN
 - vx_fundamental.h, 39
- MS_BLOCK_TO_BOX
 - vx_fundamental.h, 39
- make_surface
 - VX_machine, 16
- matrix.h, 31
 - VX_dvector3_normalize, 31
 - VX_dvector4_multiply, 31
 - VX_matrix4_rotate, 31
 - VX_matrix4_rotated, 31
- native_surf
 - VX_surface, 24
- nearest_pow2
 - vx_fundamental.h, 42
- oct_stack_item, 6
- octree.h, 31
 - VX_model_octree_new, 32
 - VX_oct_fill_region, 32
 - VX_oct_info, 32
 - VX_oct_node, 32
 - VX_oct_node_leaf, 32
 - VX_oct_set_size, 32
- PI
 - vx_fundamental.h, 39
- PRINT_BLOCK
 - vx_fundamental.h, 39
- PRINT_CUBE
 - vx_fundamental.h, 39
- PRINT_MS_BLOCK
 - vx_fundamental.h, 40
- PRINT_POINT3
 - vx_fundamental.h, 40
- PRINT_RAW_FPOINT3
 - vx_fundamental.h, 40
- PRINT_RAW_POINT3
 - vx_fundamental.h, 40
- peasant.h, 33
 - VX_field, 33
 - VX_field_new, 33
 - VX_peasant, 33
- peasants_count
 - VX_field, 11
- position
 - VX_camera, 8
- present
 - VX_model, 19
- quit
 - VX_machine, 17
- R
 - vx_fundamental.h, 40
- rad_to_degrees
 - vx_fundamental.h, 43
- raw.h, 34
 - VX_model_raw_new, 34
- raw_data, 6
- ray_voxel
 - VX_model, 19
- redraw
 - VX_machine, 17
- rg
 - VX_camera, 8
- rotation_matrix
 - VX_camera, 8
- Rsz
 - VX_format, 13
- set_pixel
 - VX_surface, 24
- set_voxel
 - VX_model, 19
- slab.h, 34
 - VX_cube_in_point_fpu, 34
 - VX_cube_out_point, 35
 - VX_cube_out_point_fpu, 35
 - VX_line_sort, 36
 - VX_permutation_inspect, 35
- surface
 - VX_camera, 8
- t
 - VX_field, 11
- unlock_surface
 - VX_surface, 25
- VOX_box, 6
- VX_CL_device, 9
- VX_CL_program, 9
- VX_DRV_null
 - display.h, 29
- VX_DRV_sdl
 - display.h, 29
- VX_FULLSCREEN
 - display.h, 28
- VX_OCT_header, 20
- VX_ambient_light
 - lighting.h, 29
- VX_ambient_light_set
 - lighting.h, 30
- VX_block, 6

- vx_fundamental.h, 40
- VX_camera, 7
 - buffer, 7
 - camera.h, 27
 - data, 7
 - destroy, 7
 - draw, 8
 - position, 8
 - rg, 8
 - rotation_matrix, 8
 - surface, 8
 - workers, 8
- VX_camera_new
 - camera.h, 27
- VX_cube, 9
 - vx_fundamental.h, 40
- VX_cube_in_point_fpu
 - slab.h, 34
- VX_cube_out_point
 - slab.h, 35
- VX_cube_out_point_fpu
 - slab.h, 35
- VX_cubes_intersection
 - vx_fundamental.h, 43
- VX_cubes_intersects_p
 - vx_fundamental.h, 43
- VX_dline3, 10
 - vx_fundamental.h, 40
- VX_dpoint2, 10
 - vx_fundamental.h, 40
- VX_dpoint3, 10
 - vx_fundamental.h, 41
- VX_dvector3_normalize
 - matrix.h, 31
- VX_dvector4_multiply
 - matrix.h, 31
- VX_field, 11
 - destroy, 11
 - destroy_p, 11
 - enter, 11
 - peasant.h, 33
 - peasants_count, 11
 - t, 11
 - workers, 11
- VX_field_new
 - peasant.h, 33
- VX_fline3, 12
 - vx_fundamental.h, 41
- VX_format, 12
 - ARGB32, 12
 - Asz, 13
 - bit_pp, 13
 - Bsz, 13
 - byte_pp, 13
 - colorkey, 13
 - Gsz, 13
 - Rsz, 13
 - vx_fundamental.h, 41
- VX_fpoint2, 13
 - vx_fundamental.h, 41
- VX_fpoint3, 14
 - vx_fundamental.h, 41
- VX_fvector4_multiply
 - vx_fundamental.h, 43
- VX_iline2, 14
 - vx_fundamental.h, 41
- VX_iline3, 14
 - vx_fundamental.h, 41
- VX_init
 - display.h, 28
- VX_ipoint2, 15
 - vx_fundamental.h, 41
- VX_ipoint3, 15
 - vx_fundamental.h, 41
- VX_lib
 - display.h, 29
- VX_lib.h, 45
- VX_light, 15
 - lighting.h, 29
- VX_light_dynamic_add
 - lighting.h, 30
- VX_light_remove
 - lighting.h, 30
- VX_lights_clear
 - lighting.h, 30
- VX_lights_count
 - lighting.h, 30
- VX_lights_enumerate
 - lighting.h, 30
- VX_line_sort
 - slab.h, 36
- VX_machine, 16
 - custom_dta, 16
 - init, 16
 - make_surface, 16
 - quit, 17
 - redraw, 17
- VX_machine_default_init
 - display.h, 28
- VX_matrix4_add
 - vx_fundamental.h, 44
- VX_matrix4_identity
 - vx_fundamental.h, 44
- VX_matrix4_multiply
 - vx_fundamental.h, 44
- VX_matrix4_print
 - vx_fundamental.h, 44
- VX_matrix4_rotate
 - matrix.h, 31
- VX_matrix4_rotated
 - matrix.h, 31
- VX_matrix4_xrotation
 - vx_fundamental.h, 44
- VX_matrix4_yrotation
 - vx_fundamental.h, 44
- VX_matrix4_zrotation

- vx_fundamental.h, 45
- VX_model, 17
 - compile, 18
 - data, 18
 - dump, 18
 - free, 18
 - get_voxel, 18
 - inspect, 19
 - load, 19
 - present, 19
 - ray_voxel, 19
 - set_voxel, 19
 - voxel.h, 36
- VX_model_octree_new
 - octree.h, 32
- VX_model_raw_new
 - raw.h, 34
- VX_ms_block, 20
 - vx_fundamental.h, 41
- VX_oct_fill_region
 - octree.h, 32
- VX_oct_info, 20
 - octree.h, 32
- VX_oct_node, 21
 - octree.h, 32
- VX_oct_node_leaf, 21
 - octree.h, 32
- VX_oct_set_size
 - octree.h, 32
- VX_peasant, 22
 - peasant.h, 33
- VX_permutation_inspect
 - slab.h, 35
- VX_point_in_cube_p
 - vx_fundamental.h, 45
- VX_promise, 22
- VX_raw, 22
- VX_rect, 22
 - vx_fundamental.h, 41
- VX_stream, 23
- VX_surface, 23
 - custom_dta, 24
 - free, 24
 - get_pixel, 24
 - h, 24
 - lock_surface, 24
 - native_surf, 24
 - set_pixel, 24
 - unlock_surface, 25
 - vx_fundamental.h, 41
 - w, 25
- VX_vector4_print
 - vx_fundamental.h, 45
- VX_vxlfmt_load
 - vxlfmt.h, 46
- voxel.h, 36
 - VX_model, 36
- vx_fundamental.h, 36
- A, 38
- APPLY3, 38
- B, 38
- BOX_TO_MS_BLOCK, 38
- CHECK_INTERVAL, 39
- CROSS_PRODUCT, 39
- color_add, 42
- color_mul, 42
- deg_to_radians, 42
- FLOAT_SIGNUM, 39
- fvector3_raw_normalize, 42
- G, 39
- MAX, 39
- MAX_ALLOCABLE, 39
- MIN, 39
- MS_BLOCK_TO_BOX, 39
- nearest_pow2, 42
- PI, 39
- PRINT_BLOCK, 39
- PRINT_CUBE, 39
- PRINT_MS_BLOCK, 40
- PRINT_POINT3, 40
- PRINT_RAW_FPOINT3, 40
- PRINT_RAW_POINT3, 40
- R, 40
- rad_to_degrees, 43
- VX_block, 40
- VX_cube, 40
- VX_cubes_intersection, 43
- VX_cubes_intersects_p, 43
- VX_dline3, 40
- VX_dpoint2, 40
- VX_dpoint3, 41
- VX_fline3, 41
- VX_format, 41
- VX_fpoint2, 41
- VX_fpoint3, 41
- VX_fvector4_multiply, 43
- VX_iline2, 41
- VX_iline3, 41
- VX_ipoint2, 41
- VX_ipoint3, 41
- VX_matrix4_add, 44
- VX_matrix4_identity, 44
- VX_matrix4_multiply, 44
- VX_matrix4_print, 44
- VX_matrix4_xrotation, 44
- VX_matrix4_yrotation, 44
- VX_matrix4_zrotation, 45
- VX_ms_block, 41
- VX_point_in_cube_p, 45
- VX_rect, 41
- VX_surface, 41
- VX_vector4_print, 45
- X, 40
- Y, 40
- Z, 40
- vxl_header, 25

vxlfmt.h, [46](#)

VX_vxlfmt_load, [46](#)

w

VX_surface, [25](#)

workers

VX_camera, [8](#)

VX_field, [11](#)

X

vx_fundamental.h, [40](#)

Y

vx_fundamental.h, [40](#)

Z

vx_fundamental.h, [40](#)