

Mandelbox documentation

Generated by Doxygen 1.9.4

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 resource_manager Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Member Data Documentation	5
3.1.2.1 shader_names_chunk	5
3.1.2.2 shader_names_offset	6
3.1.2.3 shader_names_size	6
3.1.2.4 work_dir	6
3.2 shader Struct Reference	6
3.2.1 Detailed Description	6
3.2.2 Member Data Documentation	6
3.2.2.1 shader_hash	7
3.2.2.2 shader_id	7
3.2.2.3 shader_path	7
3.2.2.4 shader_type	7
3.2.2.5 status	7
3.3 shader_program Struct Reference	7
3.3.1 Detailed Description	8
3.3.2 Member Data Documentation	8
3.3.2.1 shader_prog_hash	8
3.3.2.2 shader_prog_id	8
3.3.2.3 shaders	8
3.3.2.4 status	8
4 File Documentation	9
4.1 log.h	9
4.2 murmurhash.h	9
4.3 src/resource-manager/src/resource_manager.h File Reference	10
4.3.1 Detailed Description	11
4.4 resource_manager.h	11
4.5 WinMain.h	12
Index	13

Chapter 1

Class Index

1.1 Class List

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

resource_manager	5
shader	6
shader_program	7

Chapter 2

File Index

2.1 File List

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

lib/ log.h	9
src/ WinMain.h	12
src/resource-manager/src/ murmurhash.h	9
src/resource-manager/src/ resource_manager.h	
Something	10

Chapter 3

Class Documentation

3.1 resource_manager Struct Reference

```
#include <resource_manager.h>
```

Public Attributes

- char * [work_dir](#)
- char * [shader_names_chunk](#)
- int [shader_names_offset](#)
- int [shader_names_size](#)
- [shader_program](#) * **programs**
- int **shader_program_size**
- int **shader_program_count**
- long long **status**
- int **is_binded**
- int **is_initialized**

3.1.1 Detailed Description

Structure to control your resource such as shaders.

3.1.2 Member Data Documentation

3.1.2.1 shader_names_chunk

```
char* resource_manager::shader_names_chunk
```

save all relative path to shaders. Used to print debug info.

3.1.2.2 shader_names_offset

```
int resource_manager::shader_names_offset
```

offset shader_names_chunk buffer. Used to save relative path of shaders.

3.1.2.3 shader_names_size

```
int resource_manager::shader_names_size
```

used to control shader_name_chunk size

3.1.2.4 work_dir

```
char* resource_manager::work_dir
```

Save current work directory. This is the path where all resource will be checked. Note that in function make_shader_prog() all pathes are relative.

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

- src/resource-manager/src/[resource_manager.h](#)

3.2 shader Struct Reference

```
#include <resource_manager.h>
```

Public Attributes

- char * [shader_path](#)
- hash_t [shader_hash](#)
- GLenum [shader_type](#)
- GLuint [shader_id](#)
- long long [status](#)

3.2.1 Detailed Description

Structure to simplify working with shaders.

3.2.2 Member Data Documentation

3.2.2.1 shader_hash

```
hash_t shader::shader_hash
```

hash of current shader. Used to search shader in [shader_program](#).

3.2.2.2 shader_id

```
GLuint shader::shader_id
```

shader_id is return value of `glCreateShader()` function. For more see GLFW documentation.

3.2.2.3 shader_path

```
char* shader::shader_path
```

path to the shader source file

3.2.2.4 shader_type

```
GLenum shader::shader_type
```

One of this shader type: `GL_VERTEX_SHADER`, `GL_FRAGMENT_SHADER`, `GL_GEOMETRY_SHADER`, `GL_TESS_CONTROL_SHADER`, `GL_TESS_EVALUATION_SHADER`. For more see GLFW documentation.

3.2.2.5 status

```
long long shader::status
```

Flag used to check the shader status, used in printing debug info

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

- `src/resource-manager/src/resource_manager.h`

3.3 shader_program Struct Reference

```
#include <resource_manager.h>
```

Public Attributes

- [shader shaders](#) [MAX_SHADER_TYPES]
- `hash_t` [shader_prog_hash](#)
- `GLuint` [shader_prog_id](#)
- `long long` [status](#)

3.3.1 Detailed Description

Structure that contain array of shaders (shader structure objects), and info from GLFW functions

3.3.2 Member Data Documentation

3.3.2.1 shader_prog_hash

```
hash_t shader_program::shader_prog_hash
```

hash of this shader program. Used to search [shader_program](#) in [resource_manager](#).

3.3.2.2 shader_prog_id

```
GLuint shader_program::shader_prog_id
```

return ovalue of `glCreateProgram()`. For more see GLFW documentation.

3.3.2.3 shaders

```
shader shader_program::shaders[MAX_SHADER_TYPES]
```

array of shader objects. Can be used to print debug info

3.3.2.4 status

```
long long shader_program::status
```

Flag for debug info

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

- `src/resource-manager/src/resource_manager.h`

Chapter 4

File Documentation

4.1 log.h

```
1 #ifndef LOG_INCLUDED
2 #define LOG_INCLUDED
3 #include <stdio.h>
4 #include <stdarg.h>
5 #include <stdlib.h>
6 #include <execinfo.h>
7
8 // #define __PRINT_ALL_INFO__ (...)
9 // PrintToLog("Error occured in file: %s, function backtrace: %s, line: %d\n", __FILE__,
10 // __PRETTY_FUNCTION__, __LINE__);
11 // fprintf(GetCurrentLogFile(), __VA_ARGS__);
12 #define RED "\u001b[31m"
13
14 #define FATAL_RED "\u001b[31;1m"
15 #define GREEN "\u001b[32m"
16 #define YELLOW "\u001b[33m"
17 #define BLUE "\u001b[34m"
18 #define MAGENTA "\u001b[35m"
19 #define CYAN "\u001b[36m"
20 #define END "\u001b[0m"
21
22 #define ERROR_IF(condition, ret_val, ...) \
23     do { \
24         if(condition) { \
25             ErrorPrint(__VA_ARGS__); \
26             return ret_val; \
27         } \
28     } while(0);
29
30 #define RET_IF(condition, ret_val) \
31 if(condition) { \
32     return ret_val; \
33 }
34
35 void SetLogFile(FILE* log_file = nullptr);
36
37 void ResetLogFile();
38
39 void ResetAllLogFiles();
40
41 int PrintToLog(const char* format, ...);
42
43 FILE* GetCurrentLogFile();
44
45 #define ErrorPrint(...) \
46 ErrorPrint_(__PRETTY_FUNCTION__, __LINE__, __FILE__, __VA_ARGS__);
47
48 int ErrorPrint_(const char* function, const int line, const char* file, const char* format, ...);
49
50 #endif
```

4.2 murmurhash.h

```
1 #if !defined MURMURHASH_INCLUDED
```

```

2
3 #define MURMURHASH_INCLUDED
4 #include <stdlib.h>
5
6 typedef unsigned long long hash_t;
7
8 hash_t MurmurHash(const char* key, size_t data_size);
9
10 #endif

```

4.3 src/resource-manager/src/resource_manager.h File Reference

something

```

#include <stdarg.h>
#include <glad/glad.h>
#include <murmurhash.h>
#include <stdio.h>

```

Classes

- struct [shader](#)
- struct [shader_program](#)
- struct [resource_manager](#)

Enumerations

- enum **RESOURCE_MANAGER_ERRORS** { **RME_NULLPTR** = -0xDEDED , **RME_NO_BIND** , **RME_INCCORECT_MANAGER** , **RME_MALLOC_ERROR** }
- enum **SHADER_STATUS** { **SHADERS_DESTROYED** = -1 , **SHADERS_NO_STATUS** = 0 , **SHADERS_COMPILED** , **SHADERS_LINKED** , **SHADERS_COMPILATION_FILE_LOAD_ERROR** , **SHADERS_COMPILATION_SYNTAX_ERROR** , **SHADERS_LINKING_COMPILE_ERROR** , **SHADERS_LINKING_GL_ATTACH_ERROR** }
- enum **SHADER_PROGRAM_STATUS** { **SHADER_PROG_DESTROYED** = -1 , **SHADER_PROG_NO_STATUS** , **SHADER_PROG_BUILT** , **SHADER_PROG_BUILDING_ERROR** , **SHADER_PROG_GL_VALIDATE_ERROR** , **SHADER_PROG_LOG_MALLOC_ERROR** }
- enum **RESOURCE_MANAGER_STATUS** { **RES_MAN_NO_STATUS** , **RES_MAN_INITIALIZED** , **RES_MAN_BINDED** , **RES_MAN_SHADER_PROG_MALLOC_ERROR** }
- enum **RET_VAL_ERRORS** { **NULLPTR** = -0xEBAFF , **ERROR_RET** }

Functions

- char * **file_to_buffer** (FILE *source, int *buffer_size)
- char * **load_file_source** (const char *const src_file_path)
- int **init_resource_manager** ([resource_manager](#) *res_manager, const char *exec_path)
- int **bind_resource_manager** ([resource_manager](#) res_manager)
- int **make_shader_prog** (const char *prog_name, int binary_count,...)
- int **shader_prog_log** ([shader_program](#) *prog)
- int **destroy_programs** ()
- int **destroy_resource_manager** ()
- int **resource_manager_log** ()
- int **create_shader_prog** (const char *const shader_prog_name, const char *const vert_s=nullptr, const char *const frag_s=nullptr, const char *const geom_s=nullptr, const char *const tess_ctl_s=nullptr, const char *const tess_eval_s=nullptr, const char *const comp_s=nullptr)
- int **resource_manager_shader_log** ()
- GLuint **get_shader_prog_id** (const char *prog_name)
- [shader_program](#) * **find_shader_prog** (const char *prog_name)

Variables

- const int **MAX_SHADER_TYPES** = 6
- **shader_program** * **NOT_FOUNDED**

4.3.1 Detailed Description

something

4.4 resource_manager.h

[Go to the documentation of this file.](#)

```

1
2 #ifndef SHADER_HANDLER_H_
3 #define SHADER_HANDLER_H_
4
5 #include <stdarg.h>
6 #include <glad/glad.h>
7 #include <murmurhash.h>
8 #include <stdio.h>
9
10 const int MAX_SHADER_TYPES = 6;
11
12 struct shader
13 {
14     char*      shader_path;
15     hash_t     shader_hash;
16     GLenum     shader_type;
17     GLuint     shader_id;
18     long long  status;
19 };
20
21 struct shader_program
22 {
23     shader      shaders[MAX_SHADER_TYPES];
24     hash_t      shader_prog_hash;
25     GLuint      shader_prog_id;
26     long long    status;
27 };
28
29 struct resource_manager
30 {
31     char*      work_dir;
32     char*      shader_names_chunk;
33     int        shader_names_offset;
34     int        shader_names_size;
35     shader_program* programs;
36
37     int        shader_program_size;
38     int        shader_program_count;
39
40     long long  status;
41
42     int        is_binded;
43     int        is_initialized;
44 };
45
46 enum RESOURCE_MANAGER_ERRORS
47 {
48     RME_NULLPTR = -0xDEAD,
49     RME_NO_BIND,
50     RME_INCORRECT_MANAGER,
51     RME_MALLOC_ERROR,
52 };
53
54 enum SHADER_STATUS
55 {
56     // GLOBAL STATUS
57     SHADERS_DESTROYED = -1,
58     SHADERS_NO_STATUS = 0,
59     // SHADERS_INITIALIZED = 0,
60     SHADERS_COMPILED,
61     SHADERS_LINKED,
62 }

```

```

98     SHADERS_COMPILATION_FILE_LOAD_ERROR,
99     SHADERS_COMPILATION_SYNTAX_ERROR,
100
101     SHADERS_LINKING_COMPILE_ERROR,
102     SHADERS_LINKING_GL_ATTACH_ERROR,
103
104 // BIN_SHADERS LINKING ERRORS
105 };
106
107 enum SHADER_PROGRAM_STATUS
108 {
109     SHADER_PROG_DESTROYED = -1,
110     SHADER_PROG_NO_STATUS,
111     SHADER_PROG_BUILT,
112     SHADER_PROG_BUILDING_ERROR,
113     SHADER_PROG_GL_VALIDATE_ERROR,
114     SHADER_PROG_LOG_MALLOC_ERROR
115 };
116
117 enum RESOURCE_MANAGER_STATUS
118 {
119     RES_MAN_NO_STATUS,
120     RES_MAN_INITIALIZED,
121     RES_MAN_BINDED,
122     RES_MAN_SHADER_PROG_MALLOC_ERROR,
123 };
124
125 extern shader_program* NOT_FOUNDED;
126
127
128 enum RET_VAL_ERRORS
129 {
130     NULLPTR = -0xEBAFF,
131     ERROR_RET
132 };
133
134 char* file_to_buffer(FILE* source, int* buffer_size);
135 char* load_file_source(const char *const src_file_path);
136 //int validate_shader(shader* curr_shader, const char* shader_path);
137
138 int init_resource_manager(resource_manager* res_manager, const char* exec_path);
139 int bind_resource_manager(resource_manager res_manager);
140
141 int make_shader_prog(const char* prog_name, int binary_count, ...);
142 int shader_prog_log(shader_program* prog);
143 int destroy_programs();
144
145 int destroy_resource_manager();
146 int resource_manager_log();
147
148 int create_shader_prog(const char* const shader_prog_name, const char* const vert_s = nullptr,
149                     const char* const frag_s = nullptr, const char* const geom_s = nullptr,
150                     const char* const tess_ctl_s = nullptr, const char* const tess_eval_s =
151                     nullptr, const char* const comp_s = nullptr);
152 int resource_manager_shader_log();
153
154 GLuint get_shader_prog_id(const char* prog_name);
155
156 shader_program* find_shader_prog(const char* prog_name);
157
158
159 #endif // SHADER_HANDLER_H_

```

4.5 WinMain.h

```

1
2 #include <glad/glad.h>
3 #include <GLFW/glfw3.h>
4 #include <stdio.h>
5 #include <log.h>
6 #include <resource_manager.h>
7
8
9 void glfw_key_callback(GLFWwindow* p_window, int key, int scancode, int action, int mode);
10
11 int make_ultra_shader(GLuint* p_shader_prog, GLuint* p_vao, const char* path);
12
13 GLFWwindow* make_fullscreen_window();
14
15 int make_shad_prog_n_res_man(const char *const execution_path, const char *const shader_prog_name);
16
17 void WinMain(GLFWwindow* window, GLuint shader_program, GLuint vao);

```


Index

lib/log.h, [9](#)

resource_manager, [5](#)

 shader_names_chunk, [5](#)

 shader_names_offset, [5](#)

 shader_names_size, [6](#)

 work_dir, [6](#)

shader, [6](#)

 shader_hash, [6](#)

 shader_id, [7](#)

 shader_path, [7](#)

 shader_type, [7](#)

 status, [7](#)

shader_hash

 shader, [6](#)

shader_id

 shader, [7](#)

shader_names_chunk

 resource_manager, [5](#)

shader_names_offset

 resource_manager, [5](#)

shader_names_size

 resource_manager, [6](#)

shader_path

 shader, [7](#)

shader_prog_hash

 shader_program, [8](#)

shader_prog_id

 shader_program, [8](#)

shader_program, [7](#)

 shader_prog_hash, [8](#)

 shader_prog_id, [8](#)

 shaders, [8](#)

 status, [8](#)

shader_type

 shader, [7](#)

shaders

 shader_program, [8](#)

src/resource-manager/src/murmurhash.h, [9](#)

src/resource-manager/src/resource_manager.h, [10](#), [11](#)

src/WinMain.h, [12](#)

status

 shader, [7](#)

 shader_program, [8](#)

work_dir

 resource_manager, [6](#)