Mandelbox documentation

Generated by Doxygen 1.9.4

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

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_GEOMENTRY_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

```
#ifndef LOG_INCLUDED
#define LOG_INCLUDED
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <execinfo.h>

//#define __PRINT_ALL_INFO__(...)                    \
//PrintToLog("Error occured in file: %s, function backtrace: %s, line: %d\n", __FILE__, __PRETTY_FUNCTION__, __LINE__);     \
//fprintf(GetCurrentLogFile(), __VA_ARGS__);

#define RED "\u001b[31m"

#define FATAL_RED "\u001b[31;1m"
#define GREEN "\u001b[32m"
#define YELLOW "\u001b[33m"
#define BLUE "\u001b[34m"
#define MAGENTA "\u001b[35m"
#define CYAN    "\u001b[36m"
#define END  "\u001b[0m"

#define ERROR_IF(condition, ret_val,...)                        \
    do {                                                        \
    if(condition) {                                             \
        ErrorPrint(__VA_ARGS__);                                \
        return ret_val;                                         \
    }                                                           \
    } while(0);

#define RET_IF(condition, ret_val)               \
if(condition) {                                  \
    return ret_val;                              \
}

void  SetLogFile(FILE* log_file = nullptr);

void  ResetLogFile();

void  ResetAllLogFiles();

int   PrintToLog(const char* format, ...);

FILE* GetCurrentLogFile();

#define ErrorPrint(...)                                                 \
ErrorPrint_(__PRETTY_FUNCTION__, __LINE__, __FILE__, __VA_ARGS__);

int ErrorPrint_(const char* function, const int line, const char* file, const char* format, ...);

#endif
```

4.2 murmurhash.h

```
#if !defined MURMURHASH_INCLUDED
```

```

#define MURMURHASH_INCLUDED
#include <stdlib.h>

typedef unsigned long long hash_t;

hash_t MurmurHash(const char* key, size_t data_size);

#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** = -0xDED , **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_BUILDED** ,
  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.

```c

#ifndef SHADER_HANDLER_H_
#define SHADER_HANDLER_H_

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

const int MAX_SHADER_TYPES = 6;

struct shader
{
        char*     shader_path;
        hash_t    shader_hash;
        GLenum    shader_type;
        GLuint    shader_id;
        long long status;
};


struct shader_program
{
        shader      shaders[MAX_SHADER_TYPES];
        hash_t      shader_prog_hash;
        GLuint      shader_prog_id;
        long long   status;
};

struct resource_manager
{
        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;
};

enum RESOURCE_MANAGER_ERRORS
{
        RME_NULLPTR = -0xDED,
        RME_NO_BIND,
        RME_INCCORECT_MANAGER,
        RME_MALLOC_ERROR,
};

enum SHADER_STATUS
{
// GLOBAL STATUS
        SHADERS_DESTROYED   = -1,
        SHADERS_NO_STATUS   = 0,
//        SHADERS_INITIALIZED = 0,
        SHADERS_COMPILED,
        SHADERS_LINKED,

        SHADERS_COMPILATION_FILE_LOAD_ERROR,
        SHADERS_COMPILATION_SYNTAX_ERROR,

        SHADERS_LINKING_COMPILE_ERROR,
        SHADERS_LINKING_GL_ATTACH_ERROR,

// BIN_SHADERS LINKING ERRORS
};

enum SHADER_PROGRAM_STATUS
{
        SHADER_PROG_DESTROYED = -1,
        SHADER_PROG_NO_STATUS,
        SHADER_PROG_BUILDED,
        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,
};

extern shader_program* NOT_FOUNDED;


enum RET_VAL_ERRORS
{
        NULLPTR = -0xEBAFF,
        ERROR_RET
};

char*   file_to_buffer(FILE* source, int* buffer_size);
char*   load_file_source(const char *const src_file_path);
//int     validate_shader(shader* curr_shader, const char* shader_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);


#endif // SHADER_HANDLER_H_
```

4.5 WinMain.h

```

#include <glad/glad.h>
#include <GLFW/glfw3.h>
#include <stdio.h>
#include <log.h>
#include <resource_manager.h>


void glfw_key_callback(GLFWwindow* p_window, int key, int scancode, int action, int mode);

int make_ultra_shader(GLuint* p_shader_prog, GLuint* p_vao, const char* path);

GLFWwindow* make_fullscreen_window();

int make_shad_prog_n_res_man(const char *const execution_path, const char *const shader_prog_name);

void WinMain(GLFWwindow* window, GLuint shader_program, GLuint vao);
```

Index