Project-base documentation

Written by Aldrik Ramaekers

This document pertains to version 2.0.0 of the project-base library.

Content

Content

1 array

array.h

1.1 Definitions

1.1.1 Constants

```
#define ASSERT(e_) {if(!(e_)){*(int*)0=0;}}
```

1.1.2 Structures

Comment 1
Comment 2

```
struct t_array
{
        u32 length;
        u32 reserved_length;
        u64 entry_size;
        u32 reserve_jump;
        void *data;
        mutex mutex;
} array;
```

2 assets

assets.h

2.1 Definitions

2.1.1 Constants

```
#define ASSET_IMAGE_COUNT 10
#define ASSET_FONT_COUNT 10
#define ASSET_QUEUE_COUNT 20
#define ASSET_WORKER_COUNT 2
#define TEXT_CHARSET_START 0
#define TEXT_CHARSET_END 2000
#define TOTAL_GLYPHS TEXT_CHARSET_END-TEXT_CHARSET_START
#define load_image(_name, _inmem) assets_load_image(_binary____data_imgs_##_name##_start,_binary____data_im
#define load_font(_name, _size) assets_load_font(_binary____data_fonts_##_name##_start,_binary____data_font
#define load_bitmap(_name) assets_load_bitmap(_binary____data_imgs_##_name##_start,_binary____data_imgs_##_
```

2.1.2 Structures

```
struct t_image {
        u8 *start_addr;
        u8 *end_addr;
        bool loaded;
        s32 width;
        s32 height;
        s32 channels;
        void *data;
        s16 references;
        u32 textureID;
} image;
```

```
struct t_glyph
{
        s32 width;
        s32 height;
        s32 advance;
        s32 lsb;
        s32 xoff;
        s32 yoff;
        void *bitmap;
        u32 textureID;
} glyph;
```

```
struct t_font
{
        u8 *start_addr;
        u8 *end_addr;
        bool loaded;
        s16 references;
        s16 size;
        s32 px_h;
        float32 scale;
        stbtt_fontinfo info;
        glyph glyphs[TOTAL_GLYPHS];
} font;
```

```
enum t_asset_task_type
{
        ASSET_IMAGE,
        ASSET_BITMAP,
        ASSET_FONT,
} asset_task_type;
```

```
struct t_asset_task
{
        s8 type;
        bool valid;
        union {
                image *image;
                font *font;
        };
} asset_task;
```

```
struct t_asset_queue {
        array queue;
} asset_queue;
```

```
struct t_assets {
        array images;
        array fonts;
        asset_queue queue;
        array post_process_queue;
        bool valid;
        bool done_loading_assets;
} assets;
```

3 camera

camera.h

3.1 Definitions

3.1.1 Constants

3.1.2 Structures

```
struct t_camera
{
        float32 x;
        float32 y;
        float32 rotation;
} camera;
```

4 input

input.h

4.1 Definitions

4.1.1 Constants

```
#define KEY_UNKNOWN -1
#define MOUSE_OFFSCREEN 32767
#define KEY_SPACE              32
#define KEY_APOSTROPHE         39  /* ' */
#define KEY_COMMA              44  /* , */
#define KEY_MINUS              45  /* - */
#define KEY_PERIOD             46  /* . */
#define KEY_SLASH              47  /* / */
#define KEY_0                  48
#define KEY_1                  49
#define KEY_2                  50
#define KEY_3                  51
#define KEY_4                  52
#define KEY_5                  53
#define KEY_6                  54
#define KEY_7                  55
#define KEY_8                  56
#define KEY_9                  57
#define KEY_SEMICOLON          59  /* ; */
#define KEY_EQUAL              61  /* = */
#define KEY_A                  65
#define KEY_B                  66
#define KEY_C                  67
#define KEY_D                  68
#define KEY_E                  69
#define KEY_F                  70
#define KEY_G                  71
#define KEY_H                  72
#define KEY_I                  73
#define KEY_J                  74
#define KEY_K                  75
#define KEY_L                  76
#define KEY_M                  77
#define KEY_N                  78
#define KEY_O                  79
#define KEY_P                  80
#define KEY_Q                  81
#define KEY_R                  82
#define KEY_S                  83
#define KEY_T                  84
#define KEY_U                  85
#define KEY_V                  86
#define KEY_W                  87
#define KEY_X                  88
#define KEY_Y                  89
#define KEY_Z                  90
#define KEY_LEFT_BRACKET       91  /* [ */
#define KEY_BACKSLASH          92  /* \ */
#define KEY_RIGHT_BRACKET      93  /* ] */
#define KEY_GRAVE_ACCENT       96  /* ` */
#define KEY_WORLD_1            161 /* non-US #1 */
#define KEY_WORLD_2            162 /* non-US #2 */
```

```
#define KEY_ESCAPE            256
#define KEY_ENTER             257
#define KEY_TAB               258
#define KEY_BACKSPACE         259
#define KEY_INSERT            260
#define KEY_DELETE            261
#define KEY_RIGHT             262
#define KEY_LEFT              263
#define KEY_DOWN              264
#define KEY_UP                265
#define KEY_PAGE_UP           266
#define KEY_PAGE_DOWN         267
#define KEY_HOME              268
#define KEY_END               269
#define KEY_CAPS_LOCK         280
#define KEY_SCROLL_LOCK       281
#define KEY_NUM_LOCK          282
#define KEY_PRINT_SCREEN      283
#define KEY_PAUSE             284
#define KEY_F1                290
#define KEY_F2                291
#define KEY_F3                292
#define KEY_F4                293
#define KEY_F5                294
#define KEY_F6                295
#define KEY_F7                296
#define KEY_F8                297
#define KEY_F9                298
#define KEY_F10               299
#define KEY_F11               300
#define KEY_F12               301
#define KEY_F13               302
#define KEY_F14               303
#define KEY_F15               304
#define KEY_F16               305
#define KEY_F17               306
#define KEY_F18               307
#define KEY_F19               308
#define KEY_F20               309
#define KEY_F21               310
#define KEY_F22               311
#define KEY_F23               312
#define KEY_F24               313
#define KEY_F25               314
#define KEY_KP_0              320
#define KEY_KP_1              321
#define KEY_KP_2              322
#define KEY_KP_3              323
#define KEY_KP_4              324
#define KEY_KP_5              325
#define KEY_KP_6              326
#define KEY_KP_7              327
#define KEY_KP_8              328
#define KEY_KP_9              329
#define KEY_KP_DECIMAL        330
#define KEY_KP_DIVIDE         331
#define KEY_KP_MULTIPLY       332
#define KEY_KP_SUBTRACT       333
#define KEY_KP_ADD            334
#define KEY_KP_ENTER          335
#define KEY_KP_EQUAL          336
#define KEY_LEFT_SHIFT        340
#define KEY_LEFT_CONTROL      341
#define KEY_LEFT_ALT          342
```

```
#define KEY_LEFT_SUPER          343
#define KEY_RIGHT_SHIFT         344
#define KEY_RIGHT_CONTROL       345
#define KEY_RIGHT_ALT           346
#define KEY_RIGHT_SUPER         347
#define KEY_MENU                348
#define KEY_LAST KEY_MENU
#define MAX_KEYCODE 512
#define MOUSE_DOWN (1   <<1)
#define MOUSE_RELEASE (1    <<2)
#define MOUSE_DOUBLE_CLICK (1     <<3)
#define MOUSE_CLICK (1    <<4)
#define SCROLL_UP 1
#define SCROLL_DOWN -1
#define MAX_INPUT_LENGTH 4096+1
#define MAX_PATH_LENGTH 255+1
#define MAX_INPUT_LENGTH 4096+1
#define MAX_PATH_LENGTH MAX_PATH+1
```

4.1.2 Structures

```
struct t_mouse_input
{
        s16 x;
        s16 y;
        s16 move_x;
        s16 move_y;
        s16 total_move_x;
        s16 total_move_y;
        s8 left_state;
        s8 right_state;
        s8 scroll_state;
        bool last_state_released;
} mouse_input;
```

```
enum t_keyboard_input_mode
{
        INPUT_NUMERIC,
        INPUT_FULL,
} keyboard_input_mode;
```

```
struct t_keyboard_input
{
        keyboard_input_mode input_mode;
        int modifier_state;
        bool take_input;
        u32 cursor;

        // input
        bool text_changed; // is set when text is pasted in, incase the new text is the same length as the
        bool has_selection;
        s32 selection_begin_offset;
        s32 selection_length;
        char *input_text;
        // input

        s32 input_text_len;
        bool keys[MAX_KEYCODE];
        bool input_keys[MAX_KEYCODE];
} keyboard_input;
```

5 localization

localization.h

5.1 Definitions

5.1.1 Constants

5.1.2 Structures

```
struct t_mo_entry
{
        s32 length;
        s32 offset;
} mo_entry;
```

```
struct t_mo_translation
{
        s32 identifier_len;
        char *identifier;
        char *translation;
} mo_translation;
```

```
struct t_mo_header
{
        s32 magic_number;
        s32 file_format_revision;
        s32 number_of_strings;
        s32 identifier_table_offset;
        s32 translation_table_offset;
        s32 hashtable_size;
        s32 hashtable_offset;
} mo_header;
```

```
struct t_mo_file
{
        mo_header header;
        array translations;
        char *locale_id;
        char *locale_full;
        image *icon;
} mo_file;
```

```
struct t_localization
{
        array mo_files;
        mo_file *active_localization;
        bool loaded;
} localization;
```

6 memory

memory.h

6.1 Definitions

6.1.1 Constants

```
#define MEM_ENTRY_BUFFER_SIZE 50000
#define mem_alloc(size) __custom_alloc(size)
#define mem_free(p) __custom_free(p)
#define mem_realloc(p, size) __custom_realloc(p, size);
#define memory_print_leaks() __custom_print_leaks()
#define mem_alloc(size) malloc(size)
#define mem_free(p) free(p)
#define mem_realloc(p, size) realloc(p, size)
#define memory_print_leaks() {}
#define STBI_MALLOC(sz) mem_alloc(sz)
#define STBI_REALLOC(p, newsz) mem_realloc(p, newsz)
#define STBI_FREE(p) mem_free(p)
```

6.1.2 Structures

```
struct t_mem_entry
{
        bool valid;
        void *p;
        s32 size;
        char *stacktrace;
} __mem_entry;
```

7 memory_bucket

memory_bucket.h

7.1 Definitions

7.1.1 Constants

```
#define kilobytes(num) num*1000
#define megabytes(num) kilobytes(num*1000)
```

7.1.2 Structures

```
struct t_memory_bucket_entry
{
        char *data;
        s32 length;
        s32 cursor;
} memory_bucket_entry;
```

```
struct t_memory_bucket
{
        mutex bucket_mutex;
        array buckets;
} memory_bucket;
```

8 notification

notification.h

8.1 Definitions

8.1.1 Constants

8.1.2 Structures

```
struct t_notification
{
        char *message;
        u16 duration;
} notification;
```

9 platform

platform.h

9.1 Definitions

9.1.1 Constants

```
#define platform_open_window(name, width, height, max_w, max_h, min_w, min_h) platform_open_window_ex(name,
```

9.1.2 Structures

```
struct t_platform_window platform_window;

/////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////

typedef struct t_found_file
{
        char *matched_filter;
        char *path;
} found_file;
```

```
struct t_file_match
{
        found_file file;
        s16 file_error;
        s32 file_size;

        u32 line_nr;
        s32 word_match_offset;
        s32 word_match_length;
        s32 word_match_offset_x; // highlight render offset
        s32 word_match_width; // highlight render width
        char *line_info; // will be null when no match is found
} file_match;
```

```
struct t_search_info
{
        u64 file_count;
        u64 dir_count;
} search_info;
```

:Cleanup: move to text_search.c.. what is this doing here?

```
struct t_search_result
{
        array work_queue;
        array files;
        array matches;
        s32 match_count;
        u64 find_duration_us;
        array errors;
        bool show_error_message; // error occured
        bool found_file_matches; // found/finding file matches
        s32 files_searched;
        s32 files_matched;
        s32 search_result_source_dir_len;
        bool match_found; // found text match
        mutex mutex;
        bool walking_file_system;
        bool cancel_search;
        bool done_finding_matches;
        s32 search_id;
        u64 start_time;
        bool done_finding_files;
        memory_bucket mem_bucket;
        bool is_command_line_search;
        bool threads_closed;
        search_info search_info;
        char *export_path;
        char *file_filter;
        char *directory_to_search;
        char *text_to_find;
        s32 max_thread_count;
        s32 max_file_size;
        bool is_recursive;
} search_result;
```

```
struct t_find_text_args
{
        file_match file;
        search_result *search_result_buffer;
} find_text_args;
```

```
struct t_file_content
{
        s64 content_length;
        void *content;
        s16 file_error;
} file_content;
```

```
enum t_time_type
{
        TIME_FULL,     // realtime
        TIME_THREAD,   // run time for calling thread
        TIME_PROCESS,  // run time for calling process
} time_type;
```

```
enum t_time_precision
{
        TIME_NS, // nanoseconds
        TIME_US, // microseconds
        TIME_MILI_S, // miliseconds
        TIME_S,  // seconds
} time_precision;
```

```
struct t_cpu_info
{
        s32 model;
        char model_name[255];
        float32 frequency;
        u32 cache_size;
        u32 cache_alignment;
} cpu_info;



enum t_file_dialog_type
{
        OPEN_FILE,
        OPEN_DIRECTORY,
        SAVE_FILE,
} file_dialog_type;



enum t_file_open_error
{
        FILE_ERROR_TOO_MANY_OPEN_FILES_PROCESS = 1,
        FILE_ERROR_TOO_MANY_OPEN_FILES_SYSTEM = 2,
        FILE_ERROR_NO_ACCESS = 3,
        FILE_ERROR_NOT_FOUND = 4,
        FILE_ERROR_CONNECTION_ABORTED = 5,
        FILE_ERROR_CONNECTION_REFUSED = 6,
        FILE_ERROR_NETWORK_DOWN = 7,
        FILE_ERROR_REMOTE_IO_ERROR = 8,
        FILE_ERROR_STALE = 9, // NFS server file is removed/renamed
        FILE_ERROR_GENERIC = 10,
        FILE_ERROR_TOO_BIG = 11,
} file_open_error;



struct t_list_file_args
{
        array *list;
        char *start_dir;
        char *pattern;
        bool recursive;
        bool include_directories;
        bool *state;
        bool *is_cancelled;
        memory_bucket *bucket;
        search_info *info;
} list_file_args;



enum t_cursor_type
{
        CURSOR_DEFAULT,
        CURSOR_POINTER,
        CURSOR_TEXT,
        CURSOR_DRAG,
} cursor_type;
```

```
struct t_vec2
{
        s32 x;
        s32 y;
} vec2;
```

```
struct t_backbuffer_pixel
{
        s32 color;
        u8 depth;
} backbuffer_pixel;
```

```
struct t_backbuffer
{
        s32 width;
        s32 height;
        u8 *buffer; // 4bytes color + 1byte depth
#ifdef OS_WIN
        BITMAPINFO bitmapInfo;
#endif
#ifdef OS_LINUX
        XImage * s_image;
#endif
} backbuffer;
```

NOT IMPLEMENTED ON LINUX: USE FLAGS_NONE

```
enum t_window_flags
{
        FLAGS_NONE = 0,
        FLAGS_BORDERLESS = 1,
        FLAGS_TOPMOST = 2,
        FLAGS_GLOBAL_MOUSE = 4,
        FLAGS_HIDDEN = 8,
        FLAGS_NO_TASKBAR = 16,
} window_flags;
```

10 Project-base

This is that entry point of the project_base library. This is the only file you will have to include to use this library. All files will be imported by including this file.

10.1 Introduction

10.2 Definitions

10.2.1 Constants

```
#define PROJECT_BASE_VERSION "2.0.0"
#define TARGET_FRAMERATE (1000/24.0)
#define s8 int8_t
#define s16 int16_t
#define s32 int32_t
#define s64 int64_t
#define u8 uint8_t
#define u16 uint16_t
#define u32 uint32_t
#define u64 uint64_t
#define float32 float
#define float64 double
#define f32 float
#define f64 double
#define bool uint8_t
#define bool _Bool
#define true 1
#define false 0
```

10.2.2 Structures

11 render

render.h

11.1 Definitions

11.1.1 Constants

```
#define rgb(r_,g_,b_) (color){ r_, g_, b_, 255 }
#define rgba(r_,g_,b_,a_) (color){r_,g_,b_,a_}
```

11.1.2 Structures

```
struct t_color {
        u8 r;
        u8 g;
        u8 b;
        u8 a;
} color;
```

```
struct t_vec4
{
        s32 x;
        s32 y;
        s32 w;
        s32 h;
} vec4;
```

```
struct t_render_target
{
        s32 x;
        s32 y;
        s32 w;
        s32 h;

        s32 offset_x;
        s32 offset_y;
} render_target;
```

```
enum t_triangle_direction
{
        TRIANGLE_DOWN,
        TRIANGLE_UP,
        TRIANGLE_LEFT,
        TRIANGLE_RIGHT,
} triangle_direction;
```

12 resources

resources.h

12.1 Definitions

12.1.1 Constants

12.1.2 Structures

13 settings_config

settings_config.h

13.1 Definitions

13.1.1 Constants

13.1.2 Structures

```
struct t_config_setting
{
        char *name;
        char *value;
} config_setting;
```

```
struct t_settings_config
{
        char *path;
        array settings;
        bool loaded;
} settings_config;
```

14 string_utils

string_utils.h

14.1 Definitions

14.1.1 Constants

```
#define string_contains(big, small) string_contains_ex(big, small, 0, 0)
```

14.1.2 Structures

```
struct t_text_match
{
        u32 line_nr;
        s32 word_offset;
        s32 word_match_len;
        char *line_start;
        char *line_info;
} text_match;
```

15 thread

thread.h

15.1 Definitions

15.1.1 Constants

15.1.2 Structures

16 timer

timer.h

16.1 Definitions

16.1.1 Constants

```
#define debug_print_elapsed_title(_title) printf("%.*s", _indent_c+1, "|----------------------"); printf("%s
#define debug_print_elapsed_indent() _indent_c+=2;
#define debug_print_elapsed_undent() _indent_c-=2;
#define debug_print_elapsed(_stamp,_title) printf("|%*s%s: %.2fms\n", _indent_c, "", _title, timer_elapsed_
#define debug_print_elapsed_title(_title) do { } while(0);
#define debug_print_elapsed_indent() do { } while(0);
#define debug_print_elapsed_undent() do { } while(0);
#define debug_print_elapsed(_stamp,_title) do { } while(0);
```

16.1.2 Structures

17 ui

ui.h

17.1 Definitions

17.1.1 Constants

```
#define SCROLL_SPEED 20
#define BLOCK_HEIGHT 25
#define MENU_BAR_HEIGHT 25
#define MENU_HORIZONTAL_PADDING 10
#define WIDGET_PADDING 8
#define BUTTON_HORIZONTAL_TEXT_PADDING 15
#define MENU_ITEM_WIDTH 220
#define CHECKBOX_SIZE BLOCK_HEIGHT - 8
#define TEXTBOX_HEIGHT BLOCK_HEIGHT
#define BUTTON_HEIGHT BLOCK_HEIGHT
#define BUTTON_IMAGE_PADDING 5
#define BUTTON_IMAGE_SPACING 8
#define DROPDOWN_WIDTH 225
#define DROPDOWN_ITEM_WIDTH 225
#define TEXTBOX_SCROLL_X_SPEED 32
```

17.1.2 Structures

```
enum t_ui_style_type
{
        UI_STYLE_LIGHT = 1,
        UI_STYLE_DARK = 2,
} ui_style_type;
```

```
struct t_ui_style
{
        u16 id;
        color foreground;
        color background;
        color border;
        color textbox_background;
        color textbox_active_border;
        color textbox_foreground;
        color image_outline_tint;
        color scrollbar_handle_background;
        color info_bar_background;
        color error_foreground;
        color item_hover_background;
        color scrollbar_background;
        color menu_background;
        color menu_hover_background;
        color menu_foreground;
        color widget_hover_background;
        color widget_background;
        color widget_confirm_background;
        color widget_confirm_hover_background;
        color hypertext_foreground;
        color hypertext_hover_foreground;
        color textbox_placeholder_foreground;
        color widget_confirm_border;
} ui_style;



enum t_layout_direction
{
        LAYOUT_HORIZONTAL,
        LAYOUT_VERTICAL,
} layout_direction;



struct t_dropdown_state
{
        bool state;
        int selected_index;
} dropdown_state;



struct t_scroll_state
{
        s32 height;
        s32 width;
        s32 x;
        s32 y;
        s32 scroll;
        s32 scroll_start_offset_y;
        bool in_scroll;
        bool mouse_scrolling;
} scroll_state;



struct t_ui_layout
{
        s32 dropdown_item_count;
        s32 dropdown_x;
        s32 offset_x;
        s32 offset_y;
        layout_direction layout_direction;
        s32 prev_offset_x;
        s32 width;
        s32 height;
        s32 menu_offset_y;
        s32 block_height;
        s32 start_offset_y;
        s32 start_offset_x;
        scroll_state *scroll;
        s32 padding;
        dropdown_state *active_dropdown_state;
} ui_layout;



struct t_textbox_history_entry
{
        char *text;
        s32 cursor_offset;
} textbox_history_entry;
```

```
struct t_textbox_state
{
        bool deselect_on_enter;
        bool accept_newline;
        char *buffer;
        s32 selection_start_index;
        bool state;
        s32 diff;
        bool double_clicked_to_select;
        s32 double_clicked_to_select_cursor_index;
        s32 max_len;
        s32 text_offset_x;
        bool attempting_to_select;
        array history;
        array future;
        s32 last_click_cursor_index;
} textbox_state;
```

```
struct t_checkbox_state
{
        bool state;
} checkbox_state;
```

```
struct t_button_state
{
        bool state;
} button_state;
```

```
struct t_submenu_state
{
        bool open;
        bool hovered;
        s32 item_count;
        s32 w;
        s32 x;
        s32 y;
} submenu_state;
```

```
struct t_submenus
{
        s32 count;
        submenu_state *submenu_stack[10];
} submenus;
```

```
struct t_ui_tooltip
{
        s32 x;
        s32 y;
        s32 w;
```

```
	s32 h;
} ui_tooltip;



struct t_ui_context
{
	platform_window *active_window;
	keyboard_input *keyboard;
	mouse_input *mouse;
	camera *camera;

	cursor_type cursor_to_set;
	ui_style style;
	ui_layout layout;
	font *font_small;
	s32 active_menu_id;
	u32 next_id;
	s32 menu_item_count;
	dropdown_state *active_dropdown;
	u32 confirming_button_id;
	textbox_state *current_active_textbox;
	submenus submenus;
	bool item_hovered;
	u32 item_hovered_id;
	u32 item_hovered_duration;
	ui_tooltip tooltip;
} ui_context;
```