

Project-base Technical Reference Manual

Written by Aldrik Ramaekers

This document is distributed under the BSD 2-Clause 'Simplified' License.

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

Introduction

This document gives a technical description for the Project-base library. The Project-base library is a general purpose library intended for creating graphical programs for the Windows and Linux operating system. This document describes all the components of the Project-base library and gives examples for using these components.

Content

<u>1 array</u>	1
<u>1.1 Definitions</u>	1
<u>1.1.1 Constants</u>	1
<u>1.1.2 Structures</u>	1
<u>1.1.3 Methods</u>	1
<u>1.2 Explanation</u>	1
<u>2 assets</u>	2
<u>2.1 Definitions</u>	2
<u>2.1.1 Constants</u>	2
<u>2.1.2 Structures</u>	2
<u>2.1.3 Methods</u>	3
<u>3 camera</u>	4
<u>3.1 Definitions</u>	4
<u>3.1.1 Structures</u>	4
<u>3.1.2 Methods</u>	4
<u>4 input</u>	5
<u>4.1 Definitions</u>	5
<u>4.1.1 Constants</u>	5
<u>4.1.2 Structures</u>	7
<u>4.1.3 Methods</u>	8
<u>4.2 Explanation</u>	8
<u>5 localization</u>	9
<u>5.1 Definitions</u>	9
<u>5.1.1 Structures</u>	9
<u>5.1.2 Methods</u>	9
<u>5.2 Explanation</u>	10
<u>6 memory</u>	11
<u>6.1 Definitions</u>	11
<u>6.1.1 Constants</u>	11
<u>6.1.2 Structures</u>	11
<u>6.1.3 Methods</u>	11
<u>7 memory bucket</u>	12
<u>7.1 Definitions</u>	12
<u>7.1.1 Constants</u>	12
<u>7.1.2 Structures</u>	12
<u>7.1.3 Methods</u>	12
<u>8 notification</u>	13
<u>8.1 Definitions</u>	13
<u>8.1.1 Structures</u>	13
<u>8.1.2 Methods</u>	13
<u>9 platform</u>	14
<u>9.1 Definitions</u>	14
<u>9.1.1 Constants</u>	14
<u>9.1.2 Structures</u>	14
<u>9.1.3 Methods</u>	17
<u>9.2 Explanation</u>	18

Content

<u>10 project_base</u>	19
<u>10.1 Definitions</u>	19
<u>10.1.1 Constants</u>	19
<u>11 render</u>	20
<u>11.1 Definitions</u>	20
<u>11.1.1 Constants</u>	20
<u>11.1.2 Structures</u>	20
<u>11.1.3 Methods</u>	20
<u>11.2 Explanation</u>	21
<u>12 settings_config</u>	22
<u>12.1 Definitions</u>	22
<u>12.1.1 Structures</u>	22
<u>12.1.2 Methods</u>	22
<u>13 string_utils</u>	23
<u>13.1 Definitions</u>	23
<u>13.1.1 Constants</u>	23
<u>13.1.2 Structures</u>	23
<u>13.1.3 Methods</u>	23
<u>14 thread</u>	25
<u>14.1 Definitions</u>	25
<u>14.1.1 Methods</u>	25
<u>15 timer</u>	26
<u>15.1 Definitions</u>	26
<u>15.1.1 Constants</u>	26
<u>15.1.2 Methods</u>	26
<u>16 ui</u>	27
<u>16.1 Definitions</u>	27
<u>16.1.1 Constants</u>	27
<u>16.1.2 Structures</u>	27
<u>16.1.3 Methods</u>	30
<u>16.2 Explanation</u>	30
<u>17 History</u>	31

1 array

array.h

1.1 Definitions

1.1.1 Constants

```
1d1    #define ASSERT(e_) {if(!(e_)){*(int*)0=0;}}
```

1.1.2 Structures

```
1s1  
struct t_array  
{  
    u32 length;  
    u32 reserved_length;  
    u64 entry_size;  
    u32 reserve_jump;  
    void *data;  
    mutex mutex;  
} array;
```

1.1.3 Methods

```
1f1    array array_create(u64 entry_size);  
1f2    bool array_exists(array *array);  
1f3    int array_push(array *array, void *data);  
1f4    int array_push_size(array *array, void *data, s32 data_size);  
1f5    void array_remove_at(array *array, u32 at);  
1f6    void array_remove(array *array, void *ptr);  
1f7    void array_remove_by(array *array, void *data);  
1f8    void *array_at(array *array, u32 at);  
1f9    void array_destroy(array *array);  
1f10   void array_swap(array *array, u32 swap1, u32 swap2);  
1f11   void array_reserve(array *array, u32 reserve_count);  
1f12   array array_copy(array *array);
```

1.2 Explanation

#	Comments
<i>1s1</i>	Comment 1 Comment 2
<i>1f1</i>	// Comment // Comment 1 // Comment 2

2 assets

assets.h

2.1 Definitions

2.1.1 Constants

```
2d1    #define ASSET_IMAGE_COUNT 10
2d2    #define ASSET_FONT_COUNT 10
2d3    #define ASSET_QUEUE_COUNT 20
2d4    #define ASSET_WORKER_COUNT 2
2d5    #define TEXT_CHARSET_START 0
2d6    #define TEXT_CHARSET_END 2000
2d7    #define TOTAL_GLYPHS TEXT_CHARSET_END-TEXT_CHARSET_START
2d8    #define load_image(_name, _inmem) assets_load_image(_binary____data_imgs_##_name##_start,_binary____d
2d9    #define load_font(_name, _size) assets_load_font(_binary____data_fonts_##_name##_start,_binary____d
2d10   #define load_bitmap(_name) assets_load_bitmap(_binary____data_imgs_##_name##_start,_binary____data_
```

2.1.2 Structures

```
2s1
struct t_image {
    u8 *start_addr;
    u8 *end_addr;
    bool loaded;
    s32 width;
    s32 height;
    s32 channels;
    void *data;
    s16 references;
    u32 textureID;
} image;
```

```
2s2
struct t_glyph
{
    s32 width;
    s32 height;
    s32 advance;
    s32 lsb;
    s32 xoff;
    s32 yoff;
    void *bitmap;
    u32 textureID;
} glyph;
```

```
2s3
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;
```

2s4

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

2s5

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

2s6

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

2s7

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

2.1.3 Methods

```
2f1    image *assets_load_image(u8 *start_addr, u8 *end_addr);
2f2    void assets_destroy_image(image *image);
2f3    image *assets_load_bitmap(u8 *start_addr, u8 *end_addr);
2f4    void assets_destroy_bitmap(image *image);
2f5    font *assets_load_font(u8 *start_addr, u8 *end_addr, s16 size);
2f6    void assets_destroy_font(font *font);
```

3 camera

camera.h

3.1 Definitions

3.1.1 Structures

```
3s1  
struct t_camera  
{  
    float32 x;  
    float32 y;  
    float32 rotation;  
} camera;
```

3.1.2 Methods

```
3f1    void camera_apply_transformations(platform_window *window, camera *camera);
```

4 input

input.h

4.1 Definitions

4.1.1 Constants

```
4d1    #define KEY_UNKNOWN -1
4d2    #define MOUSE_OFFSCREEN 32767
4d3    #define KEY_SPACE 32
4d4    #define KEY_APOSTROPHE 39 /* ' */
4d5    #define KEY_COMMA 44 /* , */
4d6    #define KEY_MINUS 45 /* - */
4d7    #define KEY_PERIOD 46 /* . */
4d8    #define KEY_SLASH 47 /* / */
4d9    #define KEY_0 48
4d10   #define KEY_1 49
4d11   #define KEY_2 50
4d12   #define KEY_3 51
4d13   #define KEY_4 52
4d14   #define KEY_5 53
4d15   #define KEY_6 54
4d16   #define KEY_7 55
4d17   #define KEY_8 56
4d18   #define KEY_9 57
4d19   #define KEY_SEMICOLON 59 /* ; */
4d20   #define KEY_EQUAL 61 /* = */
4d21   #define KEY_A 65
4d22   #define KEY_B 66
4d23   #define KEY_C 67
4d24   #define KEY_D 68
4d25   #define KEY_E 69
4d26   #define KEY_F 70
4d27   #define KEY_G 71
4d28   #define KEY_H 72
4d29   #define KEY_I 73
4d30   #define KEY_J 74
4d31   #define KEY_K 75
4d32   #define KEY_L 76
4d33   #define KEY_M 77
4d34   #define KEY_N 78
4d35   #define KEY_O 79
4d36   #define KEY_P 80
4d37   #define KEY_Q 81
4d38   #define KEY_R 82
4d39   #define KEY_S 83
4d40   #define KEY_T 84
4d41   #define KEY_U 85
4d42   #define KEY_V 86
4d43   #define KEY_W 87
4d44   #define KEY_X 88
4d45   #define KEY_Y 89
4d46   #define KEY_Z 90
4d47   #define KEY_LEFT_BRACKET 91 /* [ */
4d48   #define KEY_BACKSLASH 92 /* \ */
4d49   #define KEY_RIGHT_BRACKET 93 /* ] */
4d50   #define KEY_GRAVE_ACCENT 96 /* ` */
4d51   #define KEY_WORLD_1 161 /* non-US #1 */
4d52   #define KEY_WORLD_2 162 /* non-US #2 */
```


<u>4d53</u>	#define KEY_ESCAPE	256
<u>4d54</u>	#define KEY_ENTER	257
<u>4d55</u>	#define KEY_TAB	258
<u>4d56</u>	#define KEY_BACKSPACE	259
<u>4d57</u>	#define KEY_INSERT	260
<u>4d58</u>	#define KEY_DELETE	261
<u>4d59</u>	#define KEY_RIGHT	262
<u>4d60</u>	#define KEY_LEFT	263
<u>4d61</u>	#define KEY_DOWN	264
<u>4d62</u>	#define KEY_UP	265
<u>4d63</u>	#define KEY_PAGE_UP	266
<u>4d64</u>	#define KEY_PAGE_DOWN	267
<u>4d65</u>	#define KEY_HOME	268
<u>4d66</u>	#define KEY_END	269
<u>4d67</u>	#define KEY_CAPS_LOCK	280
<u>4d68</u>	#define KEY_SCROLL_LOCK	281
<u>4d69</u>	#define KEY_NUM_LOCK	282
<u>4d70</u>	#define KEY_PRINT_SCREEN	283
<u>4d71</u>	#define KEY_PAUSE	284
<u>4d72</u>	#define KEY_F1	290
<u>4d73</u>	#define KEY_F2	291
<u>4d74</u>	#define KEY_F3	292
<u>4d75</u>	#define KEY_F4	293
<u>4d76</u>	#define KEY_F5	294
<u>4d77</u>	#define KEY_F6	295
<u>4d78</u>	#define KEY_F7	296
<u>4d79</u>	#define KEY_F8	297
<u>4d80</u>	#define KEY_F9	298
<u>4d81</u>	#define KEY_F10	299
<u>4d82</u>	#define KEY_F11	300
<u>4d83</u>	#define KEY_F12	301
<u>4d84</u>	#define KEY_F13	302
<u>4d85</u>	#define KEY_F14	303
<u>4d86</u>	#define KEY_F15	304
<u>4d87</u>	#define KEY_F16	305
<u>4d88</u>	#define KEY_F17	306
<u>4d89</u>	#define KEY_F18	307
<u>4d90</u>	#define KEY_F19	308
<u>4d91</u>	#define KEY_F20	309
<u>4d92</u>	#define KEY_F21	310
<u>4d93</u>	#define KEY_F22	311
<u>4d94</u>	#define KEY_F23	312
<u>4d95</u>	#define KEY_F24	313
<u>4d96</u>	#define KEY_F25	314
<u>4d97</u>	#define KEY_KP_0	320
<u>4d98</u>	#define KEY_KP_1	321
<u>4d99</u>	#define KEY_KP_2	322
<u>4d100</u>	#define KEY_KP_3	323
<u>4d101</u>	#define KEY_KP_4	324
<u>4d102</u>	#define KEY_KP_5	325
<u>4d103</u>	#define KEY_KP_6	326
<u>4d104</u>	#define KEY_KP_7	327
<u>4d105</u>	#define KEY_KP_8	328
<u>4d106</u>	#define KEY_KP_9	329
<u>4d107</u>	#define KEY_KP_DECIMAL	330
<u>4d108</u>	#define KEY_KP_DIVIDE	331
<u>4d109</u>	#define KEY_KP_MULTIPLY	332
<u>4d110</u>	#define KEY_KP_SUBTRACT	333
<u>4d111</u>	#define KEY_KP_ADD	334
<u>4d112</u>	#define KEY_KP_ENTER	335
<u>4d113</u>	#define KEY_KP_EQUAL	336
<u>4d114</u>	#define KEY_LEFT_SHIFT	340
<u>4d115</u>	#define KEY_LEFT_CONTROL	341
<u>4d116</u>	#define KEY_LEFT_ALT	342

```

4d117 #define KEY_LEFT_SUPER          343
4d118 #define KEY_RIGHT_SHIFT        344
4d119 #define KEY_RIGHT_CONTROL       345
4d120 #define KEY_RIGHT_ALT          346
4d121 #define KEY_RIGHT_SUPER        347
4d122 #define KEY_MENU               348
4d123 #define KEY_LAST KEY_MENU
4d124 #define MAX_KEYCODE 512
4d125 #define MOUSE_DOWN (1 <<1)
4d126 #define MOUSE_RELEASE (1 <<2)
4d127 #define MOUSE_DOUBLE_CLICK (1 <<3)
4d128 #define MOUSE_CLICK (1 <<4)
4d129 #define SCROLL_UP 1
4d130 #define SCROLL_DOWN -1
4d131 #define MAX_INPUT_LENGTH 4096+1
4d132 #define MAX_PATH_LENGTH 255+1
4d133 #define MAX_INPUT_LENGTH 4096+1
4d134 #define MAX_PATH_LENGTH MAX_PATH+1

```

4.1.2 Structures

4s1

```

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;

```

4s2

```

enum t_keyboard_input_mode
{
    INPUT_NUMERIC,
    INPUT_FULL,
} keyboard_input_mode;

```

4s3

```

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;

```

4.1.3 Methods

```

4f1    bool is_left_down(mouse_input *input);
4f2    bool is_left_released(mouse_input *input);
4f3    bool is_left_clicked(mouse_input *input);
4f4    bool is_left_double_clicked(mouse_input *input);
4f5    bool is_right_down(mouse_input *input);
4f6    bool is_right_released(mouse_input *input);
4f7    bool is_right_clicked(mouse_input *input);
4f8    bool keyboard_is_key_down(keyboard_input *keyboard, s16 key);
4f9    bool keyboard_is_key_pressed(keyboard_input *keyboard, s16 key);
4f10   void keyboard_set_input_text(keyboard_input *keyboard, char *text);
4f11   void keyboard_set_input_mode(keyboard_input *keyboard, keyboard_input_mode mode);
4f12   bool keyboard_is_shortcut_down(s32 shortcut_keys[2]);
4f13   void keyboard_handle_input_string(platform_window *window, keyboard_input *keyboard, char *text);
4f14   void keyboard_input_destroy(keyboard_input *keyboard);

```

4.2 Explanation

#	Comments
4f1	<pre> // should be max path length // input bool text_changed; // is set when text is pasted in, incase the new text is the same length as // input </pre>

5 localization

localization.h

5.1 Definitions

5.1.1 Structures

5s1

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

5s2

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

5s3

```
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;
```

5s4

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

5s5

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

5.1.2 Methods

5f1 char* localize(const char *identifier);

5f2 bool set_locale(char *country_id);

5.2 Explanation

#	Comments
5f1	// https://www.science.co.il/language/Locale-codes.php

6 memory

memory.h

The Project-base library does not help the user manage memory in any way. It does however provide the functions `mem_alloc(s32 size)`, `mem_realloc(void* ptr, s32 size)` and `mem_free(void* ptr)`. These functions work identical to the standard library memory management functions, but provides the ability to track allocated memory. By specifying the `MODE_DEBUGMEM` flag all allocations will be tracked until they are free'd using `mem_free()`. All allocations that are being tracked can be printed to stdout using `memory_print_leaks()`.

6.1 Definitions

6.1.1 Constants

```
6d1    #define MEM_ENTRY_BUFFER_SIZE 50000
6d2    #define mem_alloc(size) __custom_alloc(size)
6d3    #define mem_free(p) __custom_free(p)
6d4    #define mem_realloc(p, size) __custom_realloc(p, size);
6d5    #define memory_print_leaks() __custom_print_leaks()
6d6    #define mem_alloc(size) malloc(size)
6d7    #define mem_free(p) free(p)
6d8    #define mem_realloc(p, size) realloc(p, size)
6d9    #define memory_print_leaks() {}
6d10   #define STBI_MALLOC(sz) mem_alloc(sz)
6d11   #define STBI_REALLOC(p, newsz) mem_realloc(p, newsz)
6d12   #define STBI_FREE(p) mem_free(p)
```

6.1.2 Structures

```
6s1
struct t_mem_entry
{
    bool valid;
    void *p;
    s32 size;
    char *stacktrace;
} __mem_entry;
```

6.1.3 Methods

```
6f1    void* newp = malloc(size);
6f2                mem_entries[i].stacktrace = malloc(4000);
6f3                symbol->SizeOfStruct = sizeof(SYMBOL_INFO);
6f4    if (!found) assert(0 && "memory entry buffer too small");
6f5    return realloc(p, size);
```

7 memory_bucket

memory_bucket.h

7.1 Definitions

7.1.1 Constants

```
7d1    #define kilobytes(num)  num*1000
7d2    #define megabytes(num) kilobytes(num*1000)
```

7.1.2 Structures

```
7s1
struct t_memory_bucket_entry
{
    char *data;
    s32 length;
    s32 cursor;
} memory_bucket_entry;

7s2
struct t_memory_bucket
{
    mutex bucket_mutex;
    array buckets;
} memory_bucket;
```

7.1.3 Methods

```
7f1    memory_bucket memory_bucket_init(s32 bucket_size);
7f2    void* memory_bucket_reserve(memory_bucket *bucket, s32 reserve_length);
7f3    void memory_bucket_reset(memory_bucket *bucket);
```

8 notification

notification.h

8.1 Definitions

8.1.1 Structures

```
8s1  
struct t_notification  
{  
    char *message;  
    ul6 duration;  
} notification;
```

8.1.2 Methods

```
8f1    void push_notification(char *message);
```


9 platform

platform.h

9.1 Definitions

9.1.1 Constants

```
9d1      #define platform_open_window(name, width, height, max_w, max_h, min_w, min_h) platform_open_window_e
```

9.1.2 Structures

```
9s1  
struct t_platform_window platform_window;  
  
////////////////////////////////////  
////////////////////////////////////
```

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

```
9s2  
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;
```

```
9s3  
struct t_search_info  
{  
    u64 file_count;  
    u64 dir_count;  
} search_info;
```

```
9s4  
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 occurred  
    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;

```

9s5

```

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

```

9s6

```

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

```

9s7

```

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

```

9s8

```

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

```

9s9

```

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

```

9s10

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

9s11

```
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;
```

9s12

```
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;
```

9s13

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

9s14

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

9s15

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

9s16

```
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;

9s17
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;

```

9.1.3 Methods

```

9f1   platform_window* platform_open_window_ex(char *name, u16 width, u16 height, u16 max_w, u16 max_h, u16 flags);
9f2   bool platform_window_is_valid(platform_window *window);
9f3   void platform_get_focus(platform_window *window);
9f4   void platform_show_window(platform_window *window);
9f5   void platform_hide_window(platform_window *window);
9f6   bool platform_set_clipboard(platform_window *window, char *buffer);
9f7   bool platform_get_clipboard(platform_window *window, char *buffer);
9f8   void platform_window_set_size(platform_window *window, u16 width, u16 height);
9f9   void platform_window_set_position(platform_window *window, u16 x, u16 y);
9f10  void platform_destroy_window(platform_window *window);
9f11  void platform_handle_events(platform_window *window);
9f12  void platform_window_swap_buffers(platform_window *window);
9f13  void platform_set_cursor(platform_window *window, cursor_type type);
9f14  void platform_window_set_title(platform_window *window, char *name);
9f15  file_content platform_read_file_content(char *path, const char *mode);
9f16  s32 platform_get_file_size(char *path);
9f17  bool platform_write_file_content(char *path, const char *mode, char *buffer, s32 len);
9f18  void platform_destroy_file_content(file_content *content);
9f19  bool get_active_directory(char *buffer);
9f20  bool set_active_directory(char *path);
9f21  void platform_show_message(platform_window *window, char *message, char *title);
9f22  array get_filters(char *filter);
9f23  void platform_list_files_block(array *list, char *start_dir, array filters, bool recursive, memory_bucket *bucket);
9f24  void platform_list_files(array *list, char *start_dir, char *filter, bool recursive, memory_bucket *bucket);
9f25  void platform_open_file_dialog(file_dialog_type type, char *buffer, char *file_filter, char *start_dir);
9f26  bool platform_get_mac_address(char *buffer, s32 buf_size);
9f27  void *platform_open_file_dialog_block(void *arg);
9f28  char *platform_get_full_path(char *file);
9f29  void platform_open_url(char *command);
9f30  bool platform_send_http_request(char *url, char *params, char *response_buffer);
9f31  void platform_run_command(char *command);
9f32  void platform_window_make_current(platform_window *window);
9f33  void platform_init(int argc, char **argv);
9f34  void platform_setup_backbuffer(platform_window *window);
9f35  void platform_set_icon(platform_window *window, image *img);
9f36  void platform_autocomplete_path(char *buffer, bool want_dir);
9f37  bool platform_directory_exists(char *path);
9f38  void platform_create_directory(char *path);
9f39  bool platform_file_exists(char *path);

```

```

9f40 void platform_show_alert(char *title, char *message);
9f41 char *get_config_save_location(char *buffer, char *directory);
9f42 char *get_file_extension(char *path);
9f43 void get_name_from_path(char *buffer, char *path);
9f44 void get_directory_from_path(char *buffer, char *path);
9f45 vec2 platform_get_window_size(platform_window *window);
9f46 s32 filter_matches(array *filters, char *string, char **matched_filter);
9f47 void platform_delete_file(char *path);
9f48 bool platform_keep_running(platform_window *window);
9f49 void platform_init_shared(int argc, char **argv);
9f50 u64 platform_get_time(time_type time_type, time_precision precision);
9f51 void _platform_register_window(platform_window* window);
9f52 void _platform_unregister_window(platform_window* window);
9f53 u64 platform_get_time(time_type time_type, time_precision precision);
9f54 void _platform_register_window(platform_window* window);
9f55 void _platform_unregister_window(platform_window* window);

```

9.2 Explanation

#	Comments
9s4	:Cleanup: move to text_search.c.. what is this doing here?
9s17	NOT IMPLEMENTED ON LINUX: USE FLAGS_NONE
	<pre> //////////////////////////////////// //////////////////////////////////// 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 // :Cleanup: move to text_search.c.. what is this doing here? bool show_error_message; // error occurred bool found_file_matches; // found/finding file matches bool match_found; // found text match //////////////////////////////////// //////////////////////////////////// TIME_FULL, // realtime TIME_THREAD, // run time for calling thread TIME_PROCESS, // run time for calling process TIME_NS, // nanoseconds TIME_US, // microseconds TIME_MILI_S, // milliseconds TIME_S, // seconds FILE_ERROR_STALE = 9, // NFS server file is removed/renamed u8 *buffer; // 4bytes color + 1byte depth // NOT IMPLEMENTED ON LINUX: USE FLAGS_NONE // NOT IMPLEMENTED ON LINUX: USE FLAGS_NONE </pre>
9f1	

10 project_base

project_base.h

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 Definitions

10.1.1 Constants

```
10d1  #define PROJECT_BASE_NAME "Project-base"
10d2  #define PROJECT_BASE_VERSION "2.0.0"
10d3  #define TARGET_FRAMERATE (1000/24.0)
10d4  #define s8 int8_t
10d5  #define s16 int16_t
10d6  #define s32 int32_t
10d7  #define s64 int64_t
10d8  #define u8 uint8_t
10d9  #define u16 uint16_t
10d10 #define u32 uint32_t
10d11 #define u64 uint64_t
10d12 #define float32 float
10d13 #define float64 double
10d14 #define f32 float
10d15 #define f64 double
10d16 #define bool uint8_t
10d17 #define bool _Bool
10d18 #define true 1
10d19 #define false 0
```

11 render

render.h

11.1 Definitions

11.1.1 Constants

```
11d1  #define rgb(r_,g_,b_) (color){ r_, g_, b_, 255 }
11d2  #define rgba(r_,g_,b_,a_) (color){r_,g_,b_,a_}
```

11.1.2 Structures

```
11s1
struct t_color {
    u8 r;
    u8 g;
    u8 b;
    u8 a;
} color;

11s2
struct t_vec4
{
    s32 x;
    s32 y;
    s32 w;
    s32 h;
} vec4;

11s3
struct t_render_target
{
    s32 x;
    s32 y;
    s32 w;
    s32 h;

    s32 offset_x;
    s32 offset_y;
} render_target;

11s4
enum t_triangle_direction
{
    TRIANGLE_DOWN,
    TRIANGLE_UP,
    TRIANGLE_LEFT,
    TRIANGLE_RIGHT,
} triangle_direction;
```

11.1.3 Methods

```
11f1  void set_render_depth(s32 depth);
11f2  void render_clear(platform_window *window);
11f3  void render_image(image *image, s32 x, s32 y, s32 width, s32 height);
11f4  void render_image_tint(image *image, s32 x, s32 y, s32 width, s32 height, color tint);
11f5  s32 render_text(font *font, s32 x, s32 y, char *text, color tint);
```

```

11f6  s32 render_text_ellipsed(font *font, s32 x, s32 y, s32 maxw, char *text, color tint);
11f7  s32 render_text_cutoff(font *font, s32 x, s32 y, char *text, color tint, u16 cutoff_width);
11f8  s32 render_text_with_cursor(font *font, s32 x, s32 y, char *text, color tint, s32 cursor_pos);
11f9  s32 render_text_with_selection(font *font, s32 x, s32 y, char *text, color tint, s32 selection_start, s32 selection_end);
11f10 s32 calculate_cursor_position(font *font, char *text, s32 click_x);
11f11 s32 calculate_text_width(font *font, char *text);
11f12 s32 calculate_text_width_upto(font *font, char *text, s32 index);
11f13 s32 calculate_text_width_from_upto(font *font, char *text, s32 from, s32 index);
11f14 void render_rectangle(s32 x, s32 y, s32 width, s32 height, color tint);
11f15 void render_rectangle_outline(s32 x, s32 y, s32 width, s32 height, u16 outline_w, color tint);
11f16 void render_triangle(s32 x, s32 y, s32 w, s32 h, color tint, triangle_direction dir);
11f17 void render_set_scissor(platform_window *window, s32 x, s32 y, s32 w, s32 h);
11f18 void render_set_rotation(float32 rotation, float32 x, float32 y, s32 depth);
11f19 #endifd render_set_rotation(float32 rotation, float32 x, float32 y, s32 depth);

```

11.2 Explanation

#	Comments
11f3	// images
11f5	// text
11f14	// primitives
11f17	// utils

12 settings_config

settings_config.h

12.1 Definitions

12.1.1 Structures

```
13s1  
struct t_config_setting  
{  
    char *name;  
    char *value;  
} config_setting;
```

```
13s2  
struct t_settings_config  
{  
    char *path;  
    array settings;  
    bool loaded;  
} settings_config;
```

12.1.2 Methods

```
13f1 void settings_init(char *path);  
13f2 config_setting* settings_get_setting(char *name);  
13f3 char* settings_get_string(char *name);  
13f4 s64 settings_get_number(char *name);  
13f5 s64 settings_get_number_or_default(char *name, s64 def);  
13f6 void settings_set_string(char *name, char *value);  
13f7 void settings_set_number(char *name, s64 value);
```

13 string_utils

string_utils.h

13.1 Definitions

13.1.1 Constants

```
14d1  #define string_contains(big, small) string_contains_ex(big, small, 0, 0)
```

13.1.2 Structures

```
14s1  
struct t_text_match  
{  
    u32 line_nr;  
    s32 word_offset;  
    s32 word_match_len;  
    char *line_start;  
    char *line_info;  
} text_match;
```

13.1.3 Methods

```
14f1  bool string_match(char *first, char *second);  
14f2  bool string_contains_ex(char *big, char *small, array *text_matches, bool *cancel_search);  
14f3  void string_trim(char *string);  
14f4  bool string_equals(char *first, char *second);  
14f5  s32 string_length(char *buffer);  
14f6  void string_append(char *buffer, char *text);  
14f7  bool string_is_asteriks(char *text);  
14f8  void string_copyn(char *buffer, char *text, s32 bufferlen);  
14f9  void string_appendn(char *buffer, char *text, s32 bufferlen);  
14f10 void string_appendf(char *buffer, char *text);  
14f11 bool string_remove(char **buffer, char *text);  
14f12 char* string_get_json_literal(char **buffer, char *tmp);  
14f13 s32 string_get_json_number(char **buffer);  
14f14 s32 string_get_json_ulong_number(char **buffer);  
14f15 char *string_get_next(char *list, char *buffer, char separator);  
14f16 bool string_is_whitespace(char *text);  
14f17 utf8_int32_t utf8_str_at(char *str, s32 index);  
14f18 void utf8_str_remove_at(char *str, s32 at);  
14f19 void utf8_str_remove_range(char *str, s32 from, s32 to);  
14f20 void utf8_str_insert_at(char *str, s32 at, utf8_int32_t newval);  
14f21 void utf8_str_insert_utf8str(char *str, s32 at, char *toinsert);  
14f22 void utf8_str_replace_at(char *str, s32 at, utf8_int32_t newval);  
14f23 char* utf8_str_upto(char *str, s32 index);  
14f24 char *utf8_str_copy_upto(char *str, s32 roof, char *buffer);  
14f25 char *utf8_str_copy_range(char *str, s32 floor, s32 roof, char *buffer);  
14f26 bool string_is_numeric(char *str);  
14f27 void string_convert_crlf_to_lf(char *buffer);  
14f28 u64 string_to_u64(char *str);  
14f29 u32 string_to_u32(char *str);  
14f30 u16 string_to_u16(char *str);  
14f31 u8 string_to_u8(char *str);  
14f32 s64 string_to_s64(char *str);  
14f33 s32 string_to_s32(char *str);  
14f34 s16 string_to_s16(char *str);
```

```
14f35 s8 string_to_s8(char *str);  
14f36 s8 string_to_f32(char *str);  
14f37 s8 string_to_f64(char *str);
```

14 thread

thread.h

14.1 Definitions

14.1.1 Methods

```
15f1 void thread_join(thread *thread);  
15f2 bool thread_tryjoin(thread *thread);  
15f3 void thread_detach(thread *thread);  
15f4 void thread_stop(thread *thread);  
15f5 void thread_sleep(u64 microseconds);  
15f6 void mutex_lock(mutex *mutex);  
15f7 bool mutex_trylock(mutex *mutex);  
15f8 void mutex_unlock(mutex *mutex);  
15f9 void mutex_destroy(mutex *mutex);
```

15 timer

timer.h

15.1 Definitions

15.1.1 Constants

```
16d1  #define debug_print_elapsed_title(_title) printf("%.s", _indent_c+1, "|-----"); pr
16d2  #define debug_print_elapsed_indent() _indent_c+=2;
16d3  #define debug_print_elapsed_undent() _indent_c-=2;
16d4  #define debug_print_elapsed(_stamp,_title) printf("|%s%s: %.2fms\n", _indent_c, "", _title, timer_
16d5  #define debug_print_elapsed_title(_title) do { } while(0);
16d6  #define debug_print_elapsed_indent() do { } while(0);
16d7  #define debug_print_elapsed_undent() do { } while(0);
16d8  #define debug_print_elapsed(_stamp,_title) do { } while(0);
```

15.1.2 Methods

```
16f1  float32 timer_elapsed_ms(u64 start);
```

16 ui

ui.h

16.1 Definitions

16.1.1 Constants

```
17d1  #define SCROLL_SPEED 20
17d2  #define BLOCK_HEIGHT 25
17d3  #define MENU_BAR_HEIGHT 25
17d4  #define MENU_HORIZONTAL_PADDING 10
17d5  #define WIDGET_PADDING 8
17d6  #define BUTTON_HORIZONTAL_TEXT_PADDING 15
17d7  #define MENU_ITEM_WIDTH 220
17d8  #define CHECKBOX_SIZE BLOCK_HEIGHT - 8
17d9  #define TEXTBOX_HEIGHT BLOCK_HEIGHT
17d10 #define BUTTON_HEIGHT BLOCK_HEIGHT
17d11 #define BUTTON_IMAGE_PADDING 5
17d12 #define BUTTON_IMAGE_SPACING 8
17d13 #define DROPDOWN_WIDTH 225
17d14 #define DROPDOWN_ITEM_WIDTH 225
17d15 #define TEXTBOX_SCROLL_X_SPEED 32
```

16.1.2 Structures

```
17s1
enum t_ui_style_type
{
    UI_STYLE_LIGHT = 1,
    UI_STYLE_DARK = 2,
} ui_style_type;

17s2
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;

17s3
enum t_layout_direction
{
    LAYOUT_HORIZONTAL,
    LAYOUT_VERTICAL,
} layout_direction;

17s4
struct t_dropdown_state
{
    bool state;
    int selected_index;
} dropdown_state;

17s5
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;

17s6
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;

17s7
struct t_textbox_history_entry
{
    char *text;
    s32 cursor_offset;
} textbox_history_entry;

17s8
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;

```

17s9

```

struct t_checkbox_state
{
    bool state;
} checkbox_state;

```

17s10

```

struct t_button_state
{
    bool state;
} button_state;

```

17s11

```

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

```

17s12

```

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

```

17s13

```

struct t_ui_tooltip
{
    s32 x;
    s32 y;
    s32 w;
    s32 h;
} ui_tooltip;

```

17s14

```

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;

```

16.1.3 Methods

```

17f1 void ui_init(font *font_small);
17f2 void ui_set_active_window(platform_window *window);
17f3 void ui_begin(s32 id, platform_window *window);
17f4 bool ui_is_menu_active(u32 id);
17f5 char* name_of_day(s32 day);
17f6 char* name_of_month(s32 month);
17f7 void ui_set_style(u16 style);
17f8 void set_active_textbox(textbox_state *textbox);
17f9 void ui_set_textbox_text(textbox_state *textbox, char *text);
17f10 void ui_set_textbox_active(textbox_state *textbox);
17f11 checkbox_state ui_create_checkbox(bool selected);
17f12 textbox_state ui_create_textbox(u16 max_len);
17f13 scroll_state ui_create_scroll(s32 scroll);
17f14 void ui_destroy_textbox(textbox_state *state);
17f15 bool ui_push_menu(char *title);
17f16 bool ui_push_menu_item(char *title, char *shortcut);
17f17 void ui_begin_menu_submenu(submenu_state *state, char *title);
17f18 void ui_end_menu_submenu(char *empty_placeholder);
17f19 bool ui_push_dropdown(dropdown_state *state, char *title);
17f20 bool ui_push_dropdown_item(image *icon, char *title, s32 index);
17f21 void ui_push_rect(s32 w, color rec);
17f22 void ui_block_begin(layout_direction direction);
17f23 void ui_push_text(char *text);
17f24 bool ui_push_text_width(char *text, s32 maxw, bool active);
17f25 void ui_push_textf(font *f, char *text);
17f26 void ui_push_textf_width(font *f, char *text, s32 maxw);
17f27 bool ui_push_hypertext_link(char *text);
17f28 bool ui_push_color_button(char *text, bool selected, color color);
17f29 bool ui_push_image(image *img, s32 w, s32 h, s32 outline, color tint);
17f30 bool ui_push_checkbox(checkbox_state *state, char *title);
17f31 bool ui_push_textbox(textbox_state *state, char *title);
17f32 bool ui_push_button(button_state *button, char *title);
17f33 bool ui_push_button_image(button_state *button, char *title, image *img);
17f34 bool ui_push_button_image_with_confirmation(button_state *state, char *title, image *img);
17f35 void ui_scroll_begin(scroll_state *state);
17f36 void ui_push_tooltip(char *text);
17f37 bool ui_push_button_image_with_confirmation(button_state *state, char *title, image *img);
17f38 void ui_scroll_begin(scroll_state *state);
17f39 void ui_push_tooltip(char *text);

```

16.2 Explanation

#	Comments
17f11	// widget initialization
17f15	// widgets

17 History

2.0.0

- rename `is_shortcut_down` to `keyboard_is_shortcut_down`
- Remove unused function `is_platform_in_darkmode`
- Fix maximum size not working on linux
- Create makefile for creating lib and examples
- make keyboard, mouse and camera global
- let the library handle asset workerer management
- fix scissor not being reset at beginning of ui routine
- make example programs
- more customization
- make `settings_config` global
- get rid of definitions like `CONFIG_DIRECTORY_WINDOWS`
- move localization loading to user code
- refactor `settings_config`

1.2.0 [21-07-2020]

- fix issue where text and images were being cut off on scroll
- fix cursor being overwritten by underlying ui element
- refactor ui menu bar
- added shadow to dropdown menu item

1.1.0 [08-07-2020]

- fix image blending issue with background colors
- triangle rendering on cpu
- refactor submenu for localization
- fix issue where last row of pixels is empty
- cursor change when hovering textbox
- new cursors

1.0.0 [16-02-2020]

- initial release