Vally

Generated by Doxygen 1.9.4

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Module Documentation

4.1 Keyboard keys

Macros

- #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
- #define KEY_WORLD_2 162
- #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 KEYS_MAX_KEYS KEY_MENU

4.1.1 Detailed Description

4.1.2 Macro Definition Documentation

4.1.2.1 KEY_0

```
#define KEY_0 48
```

4.1.2.2 KEY_1

```
#define KEY_1 49
```

4.1.2.3 KEY_2

```
#define KEY_2 50
```

4.1.2.4 KEY_3

```
#define KEY_3 51
```

4.1.2.5 KEY_4

```
#define KEY_4 52
```

4.1.2.6 KEY_5

```
#define KEY_5 53
```

4.1.2.7 KEY_6

```
#define KEY_6 54
```

4.1.2.8 KEY_7

```
#define KEY_7 55
```

4.1.2.9 KEY_8

```
#define KEY_8 56
```

4.1.2.10 KEY_9

```
#define KEY_9 57
```

4.1.2.11 KEY_A

```
#define KEY_A 65
```

4.1.2.12 KEY_APOSTROPHE

```
#define KEY_APOSTROPHE 39
```

4.1.2.13 KEY_B

```
#define KEY_B 66
```

4.1.2.14 KEY_BACKSLASH

```
#define KEY_BACKSLASH 92
```

4.1.2.15 KEY_BACKSPACE

```
#define KEY_BACKSPACE 259
```

4.1.2.16 KEY_C

```
#define KEY_C 67
```

4.1.2.17 KEY_CAPS_LOCK

```
#define KEY_CAPS_LOCK 280
```

4.1.2.18 KEY_COMMA

```
#define KEY_COMMA 44
```

4.1.2.19 KEY_D

```
#define KEY_D 68
```

4.1.2.20 KEY_DELETE

```
#define KEY_DELETE 261
```

4.1.2.21 KEY_DOWN

```
#define KEY_DOWN 264
```

4.1.2.22 KEY_E

```
#define KEY_E 69
```

4.1.2.23 KEY_END

```
#define KEY_END 269
```

4.1.2.24 KEY_ENTER

```
#define KEY_ENTER 257
```

4.1.2.25 KEY_EQUAL

```
#define KEY_EQUAL 61
```

4.1.2.26 KEY_ESCAPE

```
#define KEY_ESCAPE 256
```

4.1.2.27 KEY_F

```
#define KEY_F 70
```

4.1.2.28 KEY_F1

```
#define KEY_F1 290
```

4.1.2.29 KEY_F10

```
#define KEY_F10 299
```

4.1.2.30 KEY_F11

```
#define KEY_F11 300
```

4.1.2.31 KEY_F12

```
#define KEY_F12 301
```

4.1.2.32 KEY_F13

```
#define KEY_F13 302
```

4.1.2.33 KEY_F14

```
#define KEY_F14 303
```

4.1.2.34 KEY_F15

```
#define KEY_F15 304
```

4.1.2.35 KEY_F16

```
#define KEY_F16 305
```

4.1.2.36 KEY_F17

```
#define KEY_F17 306
```

4.1.2.37 KEY_F18

```
#define KEY_F18 307
```

4.1.2.38 KEY_F19

```
#define KEY_F19 308
```

4.1.2.39 KEY_F2

```
#define KEY_F2 291
```

4.1.2.40 KEY_F20

```
#define KEY_F20 309
```

4.1.2.41 KEY_F21

```
#define KEY_F21 310
```

4.1.2.42 KEY_F22

```
#define KEY_F22 311
```

4.1.2.43 KEY_F23

```
#define KEY_F23 312
```

4.1.2.44 KEY_F24

```
#define KEY_F24 313
```

4.1.2.45 KEY_F25

```
#define KEY_F25 314
```

4.1.2.46 KEY_F3

```
#define KEY_F3 292
```

4.1.2.47 KEY_F4

```
#define KEY_F4 293
```

4.1.2.48 KEY_F5

```
#define KEY_F5 294
```

4.1.2.49 KEY_F6

```
#define KEY_F6 295
```

4.1.2.50 KEY_F7

```
#define KEY_F7 296
```

4.1.2.51 KEY_F8

```
#define KEY_F8 297
```

4.1.2.52 KEY_F9

```
#define KEY_F9 298
```

4.1.2.53 KEY_G

```
#define KEY_G 71
```

4.1.2.54 KEY_GRAVE_ACCENT

```
#define KEY_GRAVE_ACCENT 96
```

4.1.2.55 KEY_H

```
#define KEY_H 72
```

4.1.2.56 KEY_HOME

```
#define KEY_HOME 268
```

4.1.2.57 KEY_I

```
#define KEY_I 73
```

4.1.2.58 KEY_INSERT

```
#define KEY_INSERT 260
```

4.1.2.59 KEY_J

```
#define KEY_J 74
```

4.1.2.60 KEY_K

```
#define KEY_K 75
```

4.1.2.61 KEY_KP_0

```
#define KEY_KP_0 320
```

4.1.2.62 KEY_KP_1

```
#define KEY_KP_1 321
```

4.1.2.63 KEY_KP_2

```
#define KEY_KP_2 322
```

4.1.2.64 KEY_KP_3

```
#define KEY_KP_3 323
```

4.1.2.65 KEY_KP_4

```
#define KEY_KP_4 324
```

4.1.2.66 KEY_KP_5

```
#define KEY_KP_5 325
```

4.1.2.67 KEY_KP_6

```
#define KEY_KP_6 326
```

4.1.2.68 KEY_KP_7

```
#define KEY_KP_7 327
```

4.1.2.69 KEY_KP_8

```
#define KEY_KP_8 328
```

4.1.2.70 KEY_KP_9

```
#define KEY_KP_9 329
```

4.1.2.71 KEY_KP_ADD

```
#define KEY_KP_ADD 334
```

4.1.2.72 KEY_KP_DECIMAL

```
#define KEY_KP_DECIMAL 330
```

4.1.2.73 KEY_KP_DIVIDE

```
#define KEY_KP_DIVIDE 331
```

4.1.2.74 KEY_KP_ENTER

```
#define KEY_KP_ENTER 335
```

4.1.2.75 KEY_KP_EQUAL

```
#define KEY_KP_EQUAL 336
```

4.1.2.76 KEY_KP_MULTIPLY

```
#define KEY_KP_MULTIPLY 332
```

4.1.2.77 KEY_KP_SUBTRACT

```
#define KEY_KP_SUBTRACT 333
```

4.1.2.78 KEY_L

```
#define KEY_L 76
```

4.1.2.79 KEY_LEFT

```
#define KEY_LEFT 263
```

4.1.2.80 KEY_LEFT_ALT

```
#define KEY_LEFT_ALT 342
```

4.1.2.81 KEY_LEFT_BRACKET

```
#define KEY_LEFT_BRACKET 91
```

4.1.2.82 KEY_LEFT_CONTROL

```
#define KEY_LEFT_CONTROL 341
```

4.1.2.83 KEY_LEFT_SHIFT

```
#define KEY_LEFT_SHIFT 340
```

4.1.2.84 KEY_LEFT_SUPER

```
#define KEY_LEFT_SUPER 343
```

4.1.2.85 KEY_M

```
#define KEY_M 77
```

4.1.2.86 KEY_MENU

```
#define KEY_MENU 348
```

4.1.2.87 KEY_MINUS

```
#define KEY_MINUS 45
```

4.1.2.88 KEY_N

```
#define KEY_N 78
```

4.1.2.89 KEY_NUM_LOCK

```
#define KEY_NUM_LOCK 282
```

4.1.2.90 KEY_O

```
#define KEY_O 79
```

4.1.2.91 KEY_P

```
#define KEY_P 80
```

4.1.2.92 KEY_PAGE_DOWN

```
#define KEY_PAGE_DOWN 267
```

4.1.2.93 KEY_PAGE_UP

```
#define KEY_PAGE_UP 266
```

4.1.2.94 KEY_PAUSE

```
#define KEY_PAUSE 284
```

4.1.2.95 KEY_PERIOD

```
#define KEY_PERIOD 46
```

4.1.2.96 KEY_PRINT_SCREEN

```
#define KEY_PRINT_SCREEN 283
```

4.1.2.97 KEY_Q

```
#define KEY_Q 81
```

4.1.2.98 KEY_R

```
#define KEY_R 82
```

4.1.2.99 KEY_RIGHT

```
#define KEY_RIGHT 262
```

4.1.2.100 KEY_RIGHT_ALT

```
#define KEY_RIGHT_ALT 346
```

4.1.2.101 KEY_RIGHT_BRACKET

```
#define KEY_RIGHT_BRACKET 93
```

4.1.2.102 KEY_RIGHT_CONTROL

```
#define KEY_RIGHT_CONTROL 345
```

4.1.2.103 KEY_RIGHT_SHIFT

```
#define KEY_RIGHT_SHIFT 344
```

4.1.2.104 KEY_RIGHT_SUPER

```
#define KEY_RIGHT_SUPER 347
```

4.1.2.105 KEY_S

```
#define KEY_S 83
```

4.1.2.106 KEY_SCROLL_LOCK

```
#define KEY_SCROLL_LOCK 281
```

4.1.2.107 KEY_SEMICOLON

```
#define KEY_SEMICOLON 59
```

4.1.2.108 KEY_SLASH

```
#define KEY_SLASH 47
```

4.1.2.109 KEY_SPACE

```
#define KEY_SPACE 32
```

4.1.2.110 KEY_T

```
#define KEY_T 84
```

4.1.2.111 KEY_TAB

```
#define KEY_TAB 258
```

4.1.2.112 KEY_U

```
#define KEY_U 85
```

4.1.2.113 KEY_UP

```
#define KEY_UP 265
```

4.1.2.114 KEY_V

```
#define KEY_V 86
```

4.1.2.115 KEY_W

```
#define KEY_W 87
```

4.1.2.116 KEY_WORLD_1

```
#define KEY_WORLD_1 161
```

4.1.2.117 KEY_WORLD_2

```
#define KEY_WORLD_2 162
```

4.1.2.118 KEY_X

```
#define KEY_X 88
```

4.1.2.119 KEY_Y

```
#define KEY_Y 89
```

4.1.2.120 KEY_Z

```
#define KEY_Z 90
```

4.1.2.121 KEYS_MAX_KEYS

```
#define KEYS_MAX_KEYS KEY_MENU
```

4.2 Mouse buttons

Macros

- #define MOUSE_BUTTON_1 0
- #define MOUSE_BUTTON_2 1
- #define MOUSE_BUTTON_3 2
- #define MOUSE_BUTTON_4 3
- #define MOUSE_BUTTON_5 4
- #define MOUSE_BUTTON_6 5
- #define MOUSE_BUTTON_7 6
- #define MOUSE_BUTTON_8 7
- #define MOUSE_BUTTON_MAX MOUSE_BUTTON_8
- #define MOUSE_BUTTON_LEFT MOUSE_BUTTON_1
- #define MOUSE_BUTTON_RIGHT MOUSE_BUTTON_2
- #define MOUSE_BUTTON_MIDDLE MOUSE_BUTTON_3

4.2.1 Detailed Description

4.2.2 Macro Definition Documentation

4.2.2.1 MOUSE_BUTTON_1

```
#define MOUSE_BUTTON_1 0
```

4.2.2.2 MOUSE_BUTTON_2

```
#define MOUSE_BUTTON_2 1
```

4.2.2.3 MOUSE_BUTTON_3

```
#define MOUSE_BUTTON_3 2
```

4.2.2.4 MOUSE_BUTTON_4

```
#define MOUSE_BUTTON_4 3
```

4.2.2.5 MOUSE_BUTTON_5

```
#define MOUSE_BUTTON_5 4
```

4.2.2.6 MOUSE_BUTTON_6

```
#define MOUSE_BUTTON_6 5
```

4.2.2.7 MOUSE_BUTTON_7

```
#define MOUSE_BUTTON_7 6
```

4.2.2.8 MOUSE_BUTTON_8

```
#define MOUSE_BUTTON_8 7
```

4.2.2.9 MOUSE_BUTTON_LEFT

```
#define MOUSE_BUTTON_LEFT MOUSE_BUTTON_1
```

4.2.2.10 MOUSE_BUTTON_MAX

```
#define MOUSE_BUTTON_MAX MOUSE_BUTTON_8
```

4.2.2.11 MOUSE_BUTTON_MIDDLE

```
#define MOUSE_BUTTON_MIDDLE MOUSE_BUTTON_3
```

4.2.2.12 MOUSE_BUTTON_RIGHT

```
#define MOUSE_BUTTON_RIGHT MOUSE_BUTTON_2
```

Chapter 5

Data Structure Documentation

5.1 animation Struct Reference

```
#include <animator.h>
```

Data Fields

- texture ∗ texture
- vector2f tiling
- u32 first_frame
- u32 last_frame
- f32 frametime

5.1.1 Field Documentation

5.1.1.1 first_frame

```
u32 animation::first_frame
```

5.1.1.2 frametime

```
f32 animation::frametime
```

5.1.1.3 last_frame

```
u32 animation::last_frame
```

5.1.1.4 texture

```
texture* animation::texture
```

5.1.1.5 tiling

```
vector2f animation::tiling
```

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

- include/vally/ecs/components/animator.h

5.2 animator Struct Reference

```
#include <animator.h>
```

Data Fields

- b8 playing
- animation ∗ animation
- u32 current_frame
- f32 time

5.2.1 Field Documentation

5.2.1.1 animation

```
animation* animator::animation
```

5.2.1.2 current_frame

```
u32 animator::current_frame
```

5.2.1.3 playing

```
b8 animator::playing
```

5.2.1.4 time

```
f32 animator::time
```

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

- include/vally/ecs/components/animator.h

5.3 atlas Struct Reference

```
#include <spriterenderer.h>
```

Data Fields

- vector2f tiling
- u32 sprite_count
- u32 sprites_per_row
- uvs * uvs

5.3.1 Field Documentation

5.3.1.1 sprite_count

```
u32 atlas::sprite_count
```

5.3.1.2 sprites_per_row

```
u32 atlas::sprites_per_row
```

5.3.1.3 tiling

```
vector2f atlas::tiling
```

5.3.1.4 uvs

```
uvs* atlas::uvs
```

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

- include/vally/ecs/components/spriterenderer.h

5.4 event_context Struct Reference

Context send alongside the event.

```
#include <event.h>
```

Data Fields

- union {
  i64 i64 [2]
  u64 u64 [2]
  f64 f64 [2]
  i32 i32 [4]
  u32 u32 [4]
  f32 f32 [4]
  i16 i16 [8]
  u16 u16 [8]
  i8 i8 [8]
  u8 u8 [8]
  char c [16]
  } data

  Stores 128 bytes of date.

5.4.1 Detailed Description

Context send alongside the event.

5.4.2 Field Documentation

5.4.2.1 c

```
char event_context::c[16]
```

5.4.2.2

```
union { ...  } event_context::data
```

Stores 128 bytes of date.

5.4.2.3 f32

```
f32 event_context::f32[4]
```

5.4.2.4 f64

```
f64 event_context::f64[2]
```

5.4.2.5 i16

```
i16 event_context::i16[8]
```

5.4.2.6 i32

```
i32 event_context::i32[4]
```

5.4.2.7 i64

```
i64 event_context::i64[2]
```

5.4.2.8 i8

```
i8 event_context::i8[8]
```

5.4.2.9 u16

```
u16 event_context::u16[8]
```

5.4.2.10 u32

```
u32 event_context::u32[4]
```

5.4.2.11 u64

```
u64 event_context::u64[2]
```

5.4.2.12 u8

```
u8 event_context::u8[8]
```

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

- include/vally/core/event.h

5.5 image Struct Reference

```
#include <resource_types.h>
```

Data Fields

- i8 ∗ buffer
- i32 width
- i32 height
- i32 channels

5.5.1 Field Documentation

5.5.1.1 buffer

```
i8* image::buffer
```

5.5.1.2 channels

```
i32 image::channels
```

5.5.1.3 height

```
i32 image::height
```

5.5.1.4 width

```
i32 image::width
```

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

- include/vally/resources/resource_types.h

5.6 resource Struct Reference

```
#include <resource_types.h>
```

Data Fields

- void * data
- destructor_fun destructor

5.6.1 Field Documentation

5.6.1.1 data

```
void* resource::data
```

5.6.1.2 destructor

```
destructor_fun resource::destructor
```

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

- include/vally/resources/resource_types.h

5.7 spriterenderer Struct Reference

```
#include <spriterenderer.h>
```

Data Fields

- entity entity
- texture ∗ texture
- vector2f size
- uvs uvs
- atlas atlas

5.7.1 Field Documentation

5.7.1.1 atlas

```
atlas spriterenderer::atlas
```

5.7.1.2 entity

```
entity spriterenderer::entity
```

5.7.1.3 size

```
vector2f spriterenderer::size
```

5.7.1.4 texture

```
texture* spriterenderer::texture
```

5.7.1.5 uvs

```
uvs spriterenderer::uvs
```

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

- include/vally/ecs/components/spriterenderer.h

5.8 texture Struct Reference

Texture struct.

```
#include <texture.h>
```

Data Fields

- u32 id
- i32 width
- i32 height

5.8.1 Detailed Description

Texture struct.

Contains OpenGL texture and its dimensions.

INFO: Recommended declaring a pointer.

5.8.2 Field Documentation

5.8.2.1 height

```
i32 texture::height
```

5.8.2.2 id

```
u32 texture::id
```

5.8.2.3 width

```
i32 texture::width
```

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

- include/vally/renderer/texture.h

5.9 transform Struct Reference

```
#include <transform.h>
```

Data Fields

- entity entity
- vector2f position
- vector2f scale

5.9.1 Field Documentation

5.9.1.1 entity

```
entity transform::entity
```

5.9.1.2 position

```
vector2f transform::position
```

5.9.1.3 scale

```
vector2f transform::scale
```

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

- include/vally/ecs/components/transform.h

5.10 uvs Struct Reference

```
#include <spriterenderer.h>
```

Data Fields

- vector2f uv [4]

5.10.1 Field Documentation

5.10.1.1 uv

```
vector2f uvs::uv[4]
```

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

- include/vally/ecs/components/spriterenderer.h

5.11 vector2f Struct Reference

```
#include <math_types.h>
```

Data Fields

- f32 x
- f32 y

5.11.1 Field Documentation

5.11.1.1 x

```
f32 vector2f::x
```

5.11.1.2 y

```
f32 vector2f::y
```

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

- include/vally/math/math_types.h

5.12 vector2i Struct Reference

```
#include <math_types.h>
```

Data Fields

- i32 x
- i32 y

5.12.1 Field Documentation

5.12.1.1 x

```
i32 vector2i::x
```

5.12.1.2 y

```
i32 vector2i::y
```

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

- include/vally/math/math_types.h

5.13 vector3f Struct Reference

```
#include <math_types.h>
```

Data Fields

- f32 x
- f32 y
- f32 z

5.13.1 Field Documentation

5.13.1.1 x

```
f32 vector3f::x
```

5.13.1.2 y

```
f32 vector3f::y
```

5.13.1.3 z

```
f32 vector3f::z
```

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

- include/vally/math/math_types.h

5.14 vector3i Struct Reference

```
#include <math_types.h>
```

Data Fields

- i32 x
- i32 y
- i32 z

5.14.1 Field Documentation

5.14.1.1 x

```
i32 vector3i::x
```

5.14.1.2 y

```
i32 vector3i::y
```

5.14.1.3 z

```
i32 vector3i::z
```

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

- include/vally/math/math_types.h

Chapter 6

File Documentation

6.1 include/vally/config.h File Reference

This header file contains main declarations and macros.

```
#include <stdint.h>
```

Macros

- #define VALLY_API

  Identify OS.
- #define TRUE 1

  Boolean values.
- #define FALSE 0
- #define NULL ((void ∗)0)
- #define BIT_MASK(n) (1 << n)

  Creates a bit mask with certain bit toggled on.
- #define MIN(a, b) (((a)<(b))?(a):(b))
- #define MAX(a, b) (((a)>(b))?(a):(b))

Typedefs

- typedef uint8_t u8

  Fixed-size unsigned types.
- typedef uint16_t u16
- typedef uint32_t u32
- typedef uint64_t u64
- typedef int8_t i8

  Fixed-size integer types.
- typedef int16_t i16
- typedef int32_t i32
- typedef int64_t i64
- typedef float f32

  Fixed-size decimal types.
- typedef double f64
- typedef int32_t b32

  Custom booleans.
- typedef uint8_t b8

6.1.1 Detailed Description

This header file contains main declarations and macros.

Author

Dmytro Zykov

Date

12 Jan 2022

6.1.2 Macro Definition Documentation

6.1.2.1 BIT_MASK

```
#define BIT_MASK(
              n ) (1 << n)
```

Creates a bit mask with certain bit toggled on.

6.1.2.2 FALSE

```
#define FALSE 0
```

6.1.2.3 MAX

```
#define MAX(
              a,
              b ) (((a)>(b))?(a):(b))
```

6.1.2.4 MIN

```
#define MIN(
              a,
              b ) (((a)<(b))?(a):(b))
```

6.1.2.5 NULL

```
#define NULL ((void *)0)
```

6.1.2.6 TRUE

```
#define TRUE 1
```

Boolean values.

6.1.2.7 VALLY_API

```
#define VALLY_API
```

Identify OS.

Define import and export macros.

6.1.3 Typedef Documentation

6.1.3.1 b32

```
typedef int32_t b32
```

Custom booleans.

6.1.3.2 b8

```
typedef uint8_t b8
```

6.1.3.3 f32

```
typedef float f32
```

Fixed-size decimal types.

6.1.3.4 f64

```
typedef double f64
```

6.1.3.5 i16

```
typedef int16_t i16
```

6.1.3.6 i32

```
typedef int32_t i32
```

6.1.3.7 i64

```
typedef int64_t i64
```

6.1.3.8 i8

```
typedef int8_t i8
```

Fixed-size integer types.

6.1.3.9 u16

```
typedef uint16_t u16
```

6.1.3.10 u32

```
typedef uint32_t u32
```

6.1.3.11 u64

```
typedef uint64_t u64
```

6.1.3.12 u8

```
typedef uint8_t u8
```

Fixed-size unsigned types.

6.2 config.h

Go to the documentation of this file.

```
/*************************************************************************
 * config.h                                                              *
 *                                                                       *
 * Copyright (c) 2022 Dmytro Zykov                                       *
 *                                                                       *
 * This file is a part of the vally project, and may only be used,      *
 * modified and distributed under the terms of the MIT License,         *
 * LICENSE.md. By continuing to use, modify and distribute this file *
 * you inidicate that you have read the license and accept it fully. *
 *************************************************************************/

#ifndef VALLY_CONFIG_H_
#define VALLY_CONFIG_H_

#include <stdint.h>

#if defined(_WIN32)

  // Windows
  #define VALLY_PLATFORM_WINDOWS

#elif defined(__linux__)

  // Linux
  #define VALLY_PLATFORM_LINUX

#else

  #error This OS is not supported by Vally

#endif

#if !defined(VALLY_STATIC)

  #if defined(VALLY_PLATFORM_WINDOWS)

    // Special keywords for Windows
    #if defined(VALLY_BUILD_DLL)

      #define VALLY_API __declspec(dllexport)

    #else

      #define VALLY_API __declspec(dllimport)

    #endif

    #if defined(_MSC_VER)

      #pragma warning(disable: 4251)

    #endif

  #else // Linux

    #if defined(VALLY_BUILD_DLL)

      #define VALLY_API __attribute__((visibility("default")))
    #else

      #define VALLY_API

    #endif

  #endif

#else

  // Static library does not require import and export macros
  #define VALLY_API

#endif

typedef uint8_t u8;
typedef uint16_t u16;
typedef uint32_t u32;
typedef uint64_t u64;

typedef int8_t  i8;
typedef int16_t i16;
typedef int32_t i32;
typedef int64_t i64;

typedef float f32;
typedef double f64;

typedef int32_t b32;
typedef uint8_t b8;


#define TRUE 1
#define FALSE 0

#define NULL ((void *)0)

#define BIT_MASK(n) (1 « n)

#define MIN(a,b) (((a)<(b))?(a):(b))
#define MAX(a,b) (((a)>(b))?(a):(b))

#endif // VALLY_CONFIG_H_
```

6.3 include/vally/core/engine.h File Reference

```
#include <vally/config.h>
```

Typedefs

- typedef b8(∗ engine_start) (void)

  Client API function pointer.
- typedef b8(∗ engine_update) (f32 delta_time)

  Client API function pointer.
- typedef b8(∗ engine_render) (f32 delta_time)

  Client API function pointer.

Functions

- VALLY_API b8 engine_create (i16 width, i16 height, const char ∗title)

  Initializes the vally engine.
- VALLY_API b8 engine_run (engine_start start, engine_update update, engine_render render)

  Runs the vally engine.

6.3.1 Typedef Documentation

6.3.1.1 engine_render

```
typedef b8(* engine_render) (f32 delta_time)
```

Client API function pointer.

Called inside main loop in rendering phase.

6.3.1.2 engine_start

```
typedef b8(* engine_start) (void)
```

Client API function pointer.

Called after engine is initialized, before main loop.

6.3.1.3 engine_update

```
typedef b8(* engine_update) (f32 delta_time)
```

Client API function pointer.

Called inside main loop before rendering phase.

6.3.2 Function Documentation

6.3.2.1 engine_create()

```
VALLY_API b8 engine_create (
             i16 width,
             i16 height,
             const char * title )
```

Initializes the vally engine.

This function initializes the vally engine. Must be called before any other vally function calls.

Parameters

width	Target window width
height	Target window height
title	Target window title

Returns

> TRUE - if success, FALSE - in case of fatal errors

6.3.2.2 engine_run()

```
VALLY_API b8 engine_run (
             engine_start start,
             engine_update update,
             engine_render render )
```

Runs the vally engine.

This function runs the vally engine.

Parameters

start	Function that will be called before main loop
update	Function that will be called before rendering
render	Function that will be called in rendering

Returns

> TRUE - if success, FALSE - in case of fatal errors

6.4 engine.h

Go to the documentation of this file.

```
/*********************************************************************
 * engine.h                                                          *
 *                                                                   *
 * Copyright (c) 2022 Dmytro Zykov                                   *
 *                                                                   *
 * This file is a part of the vally project, and may only be used,   *
 * modified and distributed under the terms of the MIT License,      *
 * LICENSE.md. By continuing to use, modify and distribute this file *
 * you inidicate that you have read the license and accept it fully. *
 *********************************************************************/

#ifndef VALLY_ENGINE_H_
#define VALLY_ENGINE_H_

#include <vally/config.h>

typedef b8 (*engine_start)(void);


typedef b8 (*engine_update)(f32 delta_time);

typedef b8 (*engine_render)(f32 delta_time);

VALLY_API b8 engine_create(i16 width, i16 height, const char *title);

VALLY_API b8 engine_run(engine_start start, engine_update update, engine_render render);

#endif // VALLY_ENGINE_H_
```

6.5 include/vally/core/event.h File Reference

```
#include <vally/config.h>
```

Data Structures

- struct event_context

  Context send alongside the event.

Typedefs

- typedef b8(∗ on_event_fun) (u16 code, void ∗sender, void ∗listener, event_context context)

  Pointer to event function.

Enumerations

- enum engine_event_code {
  EVENT_CODE_APP_QUIT , EVENT_CODE_KEY_PRESSED , EVENT_CODE_KEY_RELEASED ,
  EVENT_CODE_MOUSE_BUTTON_PRESSED ,
  EVENT_CODE_MOUSE_BUTTON_RELEASED , EVENT_CODE_MOUSE_MOVED , EVENT_CODE_MOUSE_WHEEL
  , EVENT_CODE_WINDOW_RESIZED ,
  EVENT_CODE_CAMERA_MOVED , MAX_EVENT_CODE = 255 }

  Event type codes.

Functions

- b8 event_init ()
- void event_terminate ()
- VALLY_API b8 event_subscribe (u16 code, void ∗listener, on_event_fun callback)

  Subscribes for certain event types.
- VALLY_API b8 event_unsubscribe (u16 code, void ∗listener, on_event_fun callback)

  Unsubscribes from certain event types.
- VALLY_API b8 event_publish (u16 code, void ∗sender, event_context context)

  Publishes event of certain type.

6.5.1 Typedef Documentation

6.5.1.1 on_event_fun

```
typedef b8(∗ on_event_fun) (u16 code, void ∗sender, void ∗listener, event_context context)
```

Pointer to event function.

6.5.2 Enumeration Type Documentation

6.5.2.1 engine_event_code

```
enum engine_event_code
```

Event type codes.

User defined event codes must have code larger then 255

Enumerator

EVENT_CODE_APP_QUIT	
EVENT_CODE_KEY_PRESSED	
EVENT_CODE_KEY_RELEASED	
EVENT_CODE_MOUSE_BUTTON_PRESSED	
EVENT_CODE_MOUSE_BUTTON_RELEASED	
EVENT_CODE_MOUSE_MOVED	
EVENT_CODE_MOUSE_WHEEL	
EVENT_CODE_WINDOW_RESIZED	
EVENT_CODE_CAMERA_MOVED	
MAX_EVENT_CODE	

6.5.3 Function Documentation

6.5.3.1 event_init()

```
b8 event_init ( )
```

6.5.3.2 event_publish()

```
VALLY_API b8 event_publish (
             u16 code,
             void * sender,
             event_context context )
```

Publishes event of certain type.

Parameters

code	Event code
sender	Object that published the event
context	Context information sent alongside the event

Returns

TRUE - if success, FALSE - in case of fatal errors

6.5.3.3 event_subscribe()

```
VALLY_API b8 event_subscribe (
            u16 code,
            void * listener,
            on_event_fun callback )
```

Subscribes for certain event types.

Executes the callback function when event is published

Parameters

code	Event code
listener	Object that subscribes for an event
callback	Event callback function

Returns

TRUE - if success, FALSE - in case of fatal errors

6.5.3.4 event_terminate()

```
void event_terminate ( )
```

6.5.3.5 event_unsubscribe()

```
VALLY_API b8 event_unsubscribe (
            u16 code,
            void * listener,
            on_event_fun callback )
```

Unsubscribes from certain event types.

Stops executing the callback function when event is published

Parameters

code	Event code
listener	Object that unsubscribes from an event
callback	Event callback function

Returns

TRUE - if success, FALSE - in case of fatal errors

6.6 event.h

Go to the documentation of this file.

```
//==================================================================
// event.h
//
// Copyright (c) 2022 Dmytro Zykov
//
// This file is a part of the vally project, and may only be used,
// modified and distributed under the terms of the MIT License,
// LICENSE.md. By continuing to use, modify and distribute this file
// you inidicate that you have read the license and accept it fully.
//==================================================================

#ifndef VALLY_EVENT_H_
#define VALLY_EVENT_H_

#include <vally/config.h>

typedef struct {
  union {
    i64 i64[2];
    u64 u64[2];
    f64 f64[2];

    i32 i32[4];
    u32 u32[4];
    f32 f32[4];

    i16 i16[8];
    u16 u16[8];

    i8 i8[8];
    u8 u8[8];

    char c[16];
  } data;
} event_context;

typedef b8 (*on_event_fun)(u16 code, void *sender, void *listener, event_context context);

b8 event_init();
void event_terminate();

VALLY_API b8 event_subscribe(u16 code, void *listener, on_event_fun callback);

VALLY_API b8 event_unsubscribe(u16 code, void *listener, on_event_fun callback);

VALLY_API b8 event_publish(u16 code, void *sender, event_context context);

typedef enum {
  EVENT_CODE_APP_QUIT,
  EVENT_CODE_KEY_PRESSED,
  EVENT_CODE_KEY_RELEASED,
  EVENT_CODE_MOUSE_BUTTON_PRESSED,
  EVENT_CODE_MOUSE_BUTTON_RELEASED,
  EVENT_CODE_MOUSE_MOVED,
  EVENT_CODE_MOUSE_WHEEL,
  EVENT_CODE_WINDOW_RESIZED,
  EVENT_CODE_CAMERA_MOVED,

  MAX_EVENT_CODE = 255
} engine_event_code;

#endif // VALLY_EVENT_H_
```

6.7 include/vally/core/input.h File Reference

```
#include <vally/config.h>
```

Macros

- #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
- #define KEY_WORLD_2 162
- #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 KEYS_MAX_KEYS KEY_MENU
- #define MOUSE_BUTTON_1 0
- #define MOUSE_BUTTON_2 1
- #define MOUSE_BUTTON_3 2
- #define MOUSE_BUTTON_4 3
- #define MOUSE_BUTTON_5 4
- #define MOUSE_BUTTON_6 5
- #define MOUSE_BUTTON_7 6
- #define MOUSE_BUTTON_8 7
- #define MOUSE_BUTTON_MAX MOUSE_BUTTON_8
- #define MOUSE_BUTTON_LEFT MOUSE_BUTTON_1
- #define MOUSE_BUTTON_RIGHT MOUSE_BUTTON_2
- #define MOUSE_BUTTON_MIDDLE MOUSE_BUTTON_3

Functions

- b8 input_init ()
- void input_terminate ()
- VALLY_API b8 input_key_pressed (u16 key)

  Checks if keyboard key is pressed.
- VALLY_API b8 input_key_released (u16 key)

  Checks if keyboard key is released.
- VALLY_API b8 input_mouse_button_pressed (u16 button)

  Checks if mouse button is pressed.
- VALLY_API b8 input_mouse_button_released (u16 button)

  Checks if mouse button is released.
- VALLY_API void input_mouse_position (f64 ∗x, f64 ∗y)

  Returns current mouse position.
- VALLY_API void input_mouse_prev_position (f64 ∗x, f64 ∗y)

  Returns current previous position.

6.7.1 Function Documentation

6.7.1.1 input_init()

```
b8 input_init ( )
```

6.7.1.2 input_key_pressed()

```
VALLY_API b8 input_key_pressed (
            u16 key )
```

Checks if keyboard key is pressed.

Parameters

key	Keyboard key code

Returns

TRUE - if pressed, FALSE - if released

6.7.1.3 input_key_released()

```
VALLY_API b8 input_key_released (
            u16 key )
```

Checks if keyboard key is released.

Parameters

key	Keyboard key code

Returns

TRUE - if released, FALSE - if pressed

6.7.1.4 input_mouse_button_pressed()

```
VALLY_API b8 input_mouse_button_pressed (
            u16 button )
```

Checks if mouse button is pressed.

Parameters

button	Mouse button code

Returns

TRUE - if pressed, FALSE - if released

6.7.1.5 input_mouse_button_released()

```
VALLY_API b8 input_mouse_button_released (
             u16 button )
```

Checks if mouse button is released.

Parameters

button	Mouse button code

Returns

TRUE - if released, FALSE - if pressed

6.7.1.6 input_mouse_position()

```
VALLY_API void input_mouse_position (
             f64 * x,
             f64 * y )
```

Returns current mouse position.

Parameters

x	pointer where mouse x coordinate will be written
y	pointer where mouse y coordinate will be written

6.7.1.7 input_mouse_prev_position()

```
VALLY_API void input_mouse_prev_position (
             f64 * x,
             f64 * y )
```

Returns current previous position.

Parameters

x	pointer where mouse x coordinate will be written
y	pointer where mouse y coordinate will be written

6.7.1.8 input_terminate()

```
void input_terminate ( )
```

6.8 input.h

Go to the documentation of this file.

```
//===================================================================
// input.h
//
// Copyright (c) 2022 Dmytro Zykov
//
// This file is a part of the vally project, and may only be used,
// modified and distributed under the terms of the MIT License,
// LICENSE.md. By continuing to use, modify and distribute this file
// you inidicate that you have read the license and accept it fully.
//===================================================================

#ifndef VALLY_INPUT_H_
#define VALLY_INPUT_H_

#include <vally/config.h>

#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
#define KEY_WORLD_2            162
#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 KEYS_MAX_KEYS          KEY_MENU
#define MOUSE_BUTTON_1         0
#define MOUSE_BUTTON_2         1
#define MOUSE_BUTTON_3         2
#define MOUSE_BUTTON_4         3
#define MOUSE_BUTTON_5         4
#define MOUSE_BUTTON_6         5
#define MOUSE_BUTTON_7         6
#define MOUSE_BUTTON_8         7
#define MOUSE_BUTTON_MAX       MOUSE_BUTTON_8
#define MOUSE_BUTTON_LEFT      MOUSE_BUTTON_1
#define MOUSE_BUTTON_RIGHT     MOUSE_BUTTON_2
#define MOUSE_BUTTON_MIDDLE    MOUSE_BUTTON_3
b8 input_init();
void input_terminate();

VALLY_API b8 input_key_pressed(u16 key);

VALLY_API b8 input_key_released(u16 key);

VALLY_API b8 input_mouse_button_pressed(u16 button);

VALLY_API b8 input_mouse_button_released(u16 button);

VALLY_API void input_mouse_position(f64 *x, f64 *y);

VALLY_API void input_mouse_prev_position(f64 *x, f64 *y);

#endif // VALLY_INPUT_H_

```

6.9 include/vally/ecs/components/animator.h File Reference

```
#include <vally/config.h>
#include <vally/renderer/texture.h>
#include <vally/ecs/ecs.h>
```

Data Structures

- struct animation
- struct animator

Functions

- VALLY_API animation animation_create (texture *texture, vector2f tiling, u32 first_frame, u32 last_frame, u32 framerate)
- VALLY_API void ecs_animator_add (entity entity)
- VALLY_API void ecs_animator_play (entity entity, animation *animation)
- VALLY_API void ecs_animator_stop (entity entity)
- void ecs_animator_update (f32 dt)

6.9.1 Function Documentation

6.9.1.1 animation_create()

```
VALLY_API animation animation_create (
            texture * texture,
            vector2f tiling,
            u32 first_frame,
            u32 last_frame,
            u32 framerate )
```

6.9.1.2 ecs_animator_add()

```
VALLY_API void ecs_animator_add (
            entity entity )
```

6.9.1.3 ecs_animator_play()

```
VALLY_API void ecs_animator_play (
              entity entity,
              animation * animation )
```

6.9.1.4 ecs_animator_stop()

```
VALLY_API void ecs_animator_stop (
              entity entity )
```

6.9.1.5 ecs_animator_update()

```
void ecs_animator_update (
              f32 dt )
```

6.10 animator.h

Go to the documentation of this file.

```
#ifndef VALLY_ECS_COMPONENTS_ANIMATOR_H_
#define VALLY_ECS_COMPONENTS_ANIMATOR_H_

#include <vally/config.h>

#include <vally/renderer/texture.h>
#include <vally/ecs/ecs.h>

typedef struct {
  texture *texture;
  vector2f tiling;
  u32 first_frame;
  u32 last_frame;
  f32 frametime;
} animation;

typedef struct {
  b8 playing;
  animation *animation;
  u32 current_frame;
  f32 time;
} animator;

VALLY_API animation animation_create(texture *texture, vector2f tiling, u32 first_frame, u32 last_frame,
    u32 framerate);

VALLY_API void ecs_animator_add(entity entity);

VALLY_API void ecs_animator_play(entity entity, animation *animation);

VALLY_API void ecs_animator_stop(entity entity);

void ecs_animator_update(f32 dt);

#endif // !VALLY_ECS_COMPONENTS_ANIMATOR_H_
```

6.11 include/vally/ecs/components/spriterenderer.h File Reference

```
#include <vally/config.h>
#include <vally/renderer/texture.h>
#include <vally/ecs/ecs.h>
#include <vally/math/math.h>
```

Data Structures

- struct uvs
- struct atlas
- struct spriterenderer

Functions

- VALLY_API void ecs_spriterenderer_add (entity entity, texture ∗texture)
- VALLY_API void ecs_spriterenderer_add_from_atlas (entity entity, texture ∗texture, vector2f tiling)
- VALLY_API void ecs_spriterenderer_atlas_select (entity entity, u32 index)
- VALLY_API spriterenderer ∗ ecs_spriterenderer_get (entity entity)
- void ecs_spriterenderer_update ()

6.11.1 Function Documentation

6.11.1.1 ecs_spriterenderer_add()

```
VALLY_API void ecs_spriterenderer_add (
            entity entity,
            texture ∗ texture )
```

6.11.1.2 ecs_spriterenderer_add_from_atlas()

```
VALLY_API void ecs_spriterenderer_add_from_atlas (
            entity entity,
            texture ∗ texture,
            vector2f tiling )
```

6.11.1.3 ecs_spriterenderer_atlas_select()

```
VALLY_API void ecs_spriterenderer_atlas_select (
            entity entity,
            u32 index )
```

6.11.1.4 ecs_spriterenderer_get()

```
VALLY_API spriterenderer * ecs_spriterenderer_get (
                entity entity )
```

6.11.1.5 ecs_spriterenderer_update()

```
void ecs_spriterenderer_update ( )
```

6.12 spriterenderer.h

Go to the documentation of this file.

```
#ifndef VALLY_ECS_COMPONENTS_SPRITERENDERER_H_
#define VALLY_ECS_COMPONENTS_SPRITERENDERER_H_

#include <vally/config.h>
#include <vally/renderer/texture.h>
#include <vally/ecs/ecs.h>
#include <vally/math/math.h>

typedef struct {
  vector2f uv[4];
} uvs;

typedef struct {
  vector2f tiling;
  u32 sprite_count;
  u32 sprites_per_row;
  uvs *uvs;
} atlas;

typedef struct {
  entity entity;
  texture *texture;
  vector2f size;
  uvs uvs;
  atlas atlas;
} spriterenderer;

VALLY_API void ecs_spriterenderer_add(entity entity, texture *texture);

VALLY_API void ecs_spriterenderer_add_from_atlas(entity entity, texture *texture, vector2f tiling);

VALLY_API void ecs_spriterenderer_atlas_select(entity entity, u32 index);

VALLY_API spriterenderer *ecs_spriterenderer_get(entity entity);

void ecs_spriterenderer_update();

#endif // !VALLY_ECS_COMPONENTS_SPRITERENDERER_H_
```

6.13 include/vally/ecs/components/transform.h File Reference

```
#include <vally/config.h>
#include <vally/ecs/ecs.h>
#include <vally/math/math.h>
```

Data Structures

- struct transform

Functions

- VALLY_API void ecs_transform_add (entity entity, vector2f position, vector2f scale)
- VALLY_API transform ∗ ecs_transform_get (entity entity)

6.13.1 Function Documentation

6.13.1.1 ecs_transform_add()

```
VALLY_API void ecs_transform_add (
            entity entity,
            vector2f position,
            vector2f scale )
```

6.13.1.2 ecs_transform_get()

```
VALLY_API transform * ecs_transform_get (
            entity entity )
```

6.14 transform.h

Go to the documentation of this file.

```
#ifndef VALLY_ECS_COMPONENTS_TRANSFORM_H_
#define VALLY_ECS_COMPONENTS_TRANSFORM_H_

#include <vally/config.h>
#include <vally/ecs/ecs.h>
#include <vally/math/math.h>

typedef struct {
  entity entity;
  vector2f position;
  vector2f scale;
} transform;

VALLY_API void ecs_transform_add(entity entity, vector2f position, vector2f scale);

VALLY_API transform *ecs_transform_get(entity entity);

#endif // !VALLY_ECS_COMPONENTS_TRANSFORM_H_
```

6.15 include/vally/ecs/ecs.h File Reference

```
#include <vally/config.h>
#include <vally/ecs/ecs_types.h>
#include <vally/ecs/components/spriterenderer.h>
#include <vally/ecs/components/transform.h>
#include <vally/ecs/components/animator.h>
```

Macros

- #define ECS_NULL_ENTITY -1
- #define ECS_MAX_ENTITIES 1000
- #define ECS_MAX_COMPONENTS ECS_MAX_ENTITIES
- #define ecs_is_alive(entity) ecs_component_has(entity, ECS_ALIVE)

Functions

- b8 ecs_init ()
- VALLY_API entity ecs_entity_create ()
- VALLY_API void ecs_component_add (entity entity, component_mask mask)
- VALLY_API void ecs_component_remove (entity entity, component_mask mask)
- b8 ecs_component_has (entity entity, component_mask mask)
- entity ecs_get_last ()

6.15.1 Macro Definition Documentation

6.15.1.1 ecs_is_alive

```
#define ecs_is_alive(
              entity ) ecs_component_has(entity, ECS_ALIVE)
```

6.15.1.2 ECS_MAX_COMPONENTS

```
#define ECS_MAX_COMPONENTS ECS_MAX_ENTITIES
```

6.15.1.3 ECS_MAX_ENTITIES

```
#define ECS_MAX_ENTITIES 1000
```

6.15.1.4 ECS_NULL_ENTITY

```
#define ECS_NULL_ENTITY -1
```

6.15.2 Function Documentation

6.15.2.1 ecs_component_add()

```
VALLY_API void ecs_component_add (
            entity entity,
            component_mask mask )
```

6.15.2.2 ecs_component_has()

```
b8 ecs_component_has (
            entity entity,
            component_mask mask )
```

6.15.2.3 ecs_component_remove()

```
VALLY_API void ecs_component_remove (
            entity entity,
            component_mask mask )
```

6.15.2.4 ecs_entity_create()

```
VALLY_API entity ecs_entity_create ( )
```

6.15.2.5 ecs_get_last()

```
entity ecs_get_last ( )
```

6.15.2.6 ecs_init()

```
b8 ecs_init ( )
```

6.16 ecs.h

Go to the documentation of this file.

```
#ifndef VALLY_ECS_ECS_H_
#define VALLY_ECS_ECS_H_

#include <vally/config.h>
#include <vally/ecs/ecs_types.h>
#include <vally/ecs/components/spriterenderer.h>
#include <vally/ecs/components/transform.h>
#include <vally/ecs/components/animator.h>

#define ECS_NULL_ENTITY -1
#define ECS_MAX_ENTITIES 1000
#define ECS_MAX_COMPONENTS ECS_MAX_ENTITIES

b8 ecs_init();

VALLY_API entity ecs_entity_create();

VALLY_API void ecs_component_add(entity entity, component_mask mask);

VALLY_API void ecs_component_remove(entity entity, component_mask mask);

b8 ecs_component_has(entity entity, component_mask mask);

entity ecs_get_last();

#define ecs_is_alive(entity)  ecs_component_has(entity, ECS_ALIVE)

#endif // !VALLY_ECS_ECS_H_
```

6.17 include/vally/ecs/ecs_types.h File Reference

```
#include <vally/config.h>
```

Typedefs

- typedef i32 entity

Enumerations

- enum component_mask {
  ECS_ALIVE = BIT_MASK(0) , ECS_TRANSFORM_COMPONENT = BIT_MASK(1) , ECS_SPRITERENDERER_COMPONENT
  = BIT_MASK(2) , ECS_RIGIDBODY_COMPONENT = BIT_MASK(3) ,
  ECS_ANIMATOR_COMPONENT = BIT_MASK(4) }

6.17.1 Typedef Documentation

6.17.1.1 entity

```
typedef i32 entity
```

6.17.2 Enumeration Type Documentation

6.17.2.1 component_mask

```
enum component_mask
```

Enumerator

ECS_ALIVE	
ECS_TRANSFORM_COMPONENT	
ECS_SPRITERENDERER_COMPONENT	
ECS_RIGIDBODY_COMPONENT	
ECS_ANIMATOR_COMPONENT	

6.18 ecs_types.h

Go to the documentation of this file.

```
#ifndef VALLY_ECS_ECS_COMPONENTS_H_
#define VALLY_ECS_ECS_COMPONENTS_H_

#include <vally/config.h>

typedef i32 entity;

typedef enum {
  ECS_ALIVE = BIT_MASK(0),
  ECS_TRANSFORM_COMPONENT = BIT_MASK(1),
  ECS_SPRITERENDERER_COMPONENT = BIT_MASK(2),
  ECS_RIGIDBODY_COMPONENT = BIT_MASK(3),
  ECS_ANIMATOR_COMPONENT = BIT_MASK(4),
} component_mask;

#endif // !VALLY_ECS_ECS_COMPONENTS_H_
```

6.19 include/vally/math/math.h File Reference

```
#include <vally/config.h>
#include <vally/math/math_types.h>
```

Macros

- #define V_PI 3.141592653589793F
- #define V_2PI V_PI ∗ 2.0F
- #define V_PI_2 V_PI / 2.0F
- #define V_PI_3 V_PI / 3.0F
- #define V_PI_4 V_PI / 4.0F
- #define V_SQRT2 1.41421356237F
- #define LERP(w, v1, v2) ((1.0 - (w)) ∗ (v1) + (w) ∗ (v2))
- #define vector2f_create(x, y) ((vector2f){x, y})
- #define vector3f_create(x, y, z) ((vector3f){x, y, z})
- #define vector2i_create(x, y) ((vector2i){x, y})
- #define vector3i_create(x, y, z) ((vector3i){x, y, z})
- #define vector2f_zero vector2f_create(0.0f, 0.0f)
- #define vector2f_one vector2f_create(1.0f, 1.0f)
- #define vector2f_right vector2f_create(1.0f, 0.0f)
- #define vector2f_left vector2f_create(-1.0f, 0.0f)
- #define vector2f_up vector2f_create(0.0f, 1.0f)
- #define vector2f_down vector2f_create(0.0f, -1.0f)

Functions

- VALLY_API f32 vsin (f32 x)
- VALLY_API f32 vcos (f32 x)
- VALLY_API f32 vtan (f32 x)
- VALLY_API f32 vasin (f32 x)
- VALLY_API f32 vacos (f32 x)
- VALLY_API f32 vatan (f32 x)
- VALLY_API f32 vsqrt (f32 x)
- VALLY_API f32 vabs (f32 x)
- VALLY_INLINE vector2f vector2f_add (vector2f a, vector2f b)
- VALLY_INLINE vector2f vector2f_sub (vector2f a, vector2f b)
- VALLY_INLINE vector2f vector2f_mul (vector2f a, vector2f b)
- VALLY_INLINE vector2f vector2f_muls (vector2f vector, f32 scalar)
- VALLY_INLINE vector2f vector2f_divs (vector2f vector, f32 scalar)
- VALLY_INLINE vector2f vector2f_div (vector2f a, vector2f b)
- VALLY_INLINE f32 vector2f_length (vector2f vector)
- VALLY_INLINE vector2f vector2f_normalized (vector2f vector)
- VALLY_INLINE void vector2f_normalize (vector2f *vector)
- VALLY_INLINE f32 vector2f_distance (vector2f a, vector2f b)
- VALLY_INLINE vector3f vector3f_add (vector3f a, vector3f b)
- VALLY_INLINE vector3f vector3f_sub (vector3f a, vector3f b)
- VALLY_INLINE vector3f vector3f_cross (vector3f a, vector3f b)
- VALLY_INLINE f32 vector3f_dot (vector3f a, vector3f b)
- VALLY_INLINE vector3f vector3f_mul (vector3f a, vector3f b)
- VALLY_INLINE vector3f vector3f_div (vector3f a, vector3f b)
- VALLY_INLINE vector3f vector3f_muls (vector3f vector, f32 scalar)
- VALLY_INLINE vector3f vector3f_divs (vector3f vector, f32 scalar)
- VALLY_INLINE f32 vector3f_length (vector3f vector)
- VALLY_INLINE vector3f vector3f_normalized (vector3f vector)
- VALLY_INLINE void vector3f_normalize (vector3f *vector)
- VALLY_INLINE f32 vector3f_distance (vector3f a, vector3f b)
- VALLY_INLINE vector2i vector2i_add (vector2i a, vector2i b)
- VALLY_INLINE vector3i vector3i_add (vector3i a, vector3i b)
- VALLY_INLINE vector2i vector2i_sub (vector2i a, vector2i b)
- VALLY_INLINE vector3i vector3i_sub (vector3i a, vector3i b)

6.19.1 Macro Definition Documentation

6.19.1.1 LERP

```
#define LERP(
                w,
                v1,
                v2 ) ((1.0 - (w)) * (v1) + (w) * (v2))
```

6.19.1.2 V_2PI

```
#define V_2PI V_PI * 2.0F
```

6.19.1.3 V_PI

```
#define V_PI 3.141592653589793F
```

6.19.1.4 V_PI_2

```
#define V_PI_2 V_PI / 2.0F
```

6.19.1.5 V_PI_3

```
#define V_PI_3 V_PI / 3.0F
```

6.19.1.6 V_PI_4

```
#define V_PI_4 V_PI / 4.0F
```

6.19.1.7 V_SQRT2

```
#define V_SQRT2 1.41421356237F
```

6.19.1.8 vector2f_create

```
#define vector2f_create(
              x,
              y ) ((vector2f){x, y})
```

6.19.1.9 vector2f_down

```
#define vector2f_down vector2f_create(0.0f, -1.0f)
```

6.19.1.10 vector2f_left

```
#define vector2f_left vector2f_create(-1.0f, 0.0f)
```

6.19.1.11 vector2f_one

```
#define vector2f_one vector2f_create(1.0f, 1.0f)
```

6.19.1.12 vector2f_right

```
#define vector2f_right vector2f_create(1.0f, 0.0f)
```

6.19.1.13 vector2f_up

```
#define vector2f_up vector2f_create(0.0f, 1.0f)
```

6.19.1.14 vector2f_zero

```
#define vector2f_zero vector2f_create(0.0f, 0.0f)
```

6.19.1.15 vector2i_create

```
#define vector2i_create(
              x,
              y ) ((vector2i){x, y})
```

6.19.1.16 vector3f_create

```
#define vector3f_create(
              x,
              y,
              z ) ((vector3f){x, y, z})
```

6.19.1.17 vector3i_create

```
#define vector3i_create(
              x,
              y,
              z ) ((vector3i){x, y, z})
```

6.19.2 Function Documentation

6.19.2.1 vabs()

```
VALLY_API f32 vabs (
             f32 x )
```

6.19.2.2 vacos()

```
VALLY_API f32 vacos (
             f32 x )
```

6.19.2.3 vasin()

```
VALLY_API f32 vasin (
             f32 x )
```

6.19.2.4 vatan()

```
VALLY_API f32 vatan (
             f32 x )
```

6.19.2.5 vcos()

```
VALLY_API f32 vcos (
             f32 x )
```

6.19.2.6 vector2f_add()

```
VALLY_INLINE vector2f vector2f_add (
             vector2f a,
             vector2f b )
```

6.19.2.7 vector2f_distance()

```
VALLY_INLINE f32 vector2f_distance (
             vector2f a,
             vector2f b )
```

6.19.2.8 vector2f_div()

```
VALLY_INLINE vector2f vector2f_div (
             vector2f a,
             vector2f b )
```

6.19.2.9 vector2f_divs()

```
VALLY_INLINE vector2f vector2f_divs (
             vector2f vector,
             f32 scalar )
```

6.19.2.10 vector2f_length()

```
VALLY_INLINE f32 vector2f_length (
             vector2f vector )
```

6.19.2.11 vector2f_mul()

```
VALLY_INLINE vector2f vector2f_mul (
             vector2f a,
             vector2f b )
```

6.19.2.12 vector2f_muls()

```
VALLY_INLINE vector2f vector2f_muls (
             vector2f vector,
             f32 scalar )
```

6.19.2.13 vector2f_normalize()

```
VALLY_INLINE void vector2f_normalize (
             vector2f * vector )
```

6.19.2.14 vector2f_normalized()

```
VALLY_INLINE vector2f vector2f_normalized (
             vector2f vector )
```

6.19.2.15 vector2f_sub()

```
VALLY_INLINE vector2f vector2f_sub (
             vector2f a,
             vector2f b )
```

6.19.2.16 vector2i_add()

```
VALLY_INLINE vector2i vector2i_add (
             vector2i a,
             vector2i b )
```

6.19.2.17 vector2i_sub()

```
VALLY_INLINE vector2i vector2i_sub (
             vector2i a,
             vector2i b )
```

6.19.2.18 vector3f_add()

```
VALLY_INLINE vector3f vector3f_add (
             vector3f a,
             vector3f b )
```

6.19.2.19 vector3f_cross()

```
VALLY_INLINE vector3f vector3f_cross (
             vector3f a,
             vector3f b )
```

6.19.2.20 vector3f_distance()

```
VALLY_INLINE f32 vector3f_distance (
             vector3f a,
             vector3f b )
```

6.19.2.21 vector3f_div()

```
VALLY_INLINE vector3f vector3f_div (
             vector3f a,
             vector3f b )
```

6.19.2.22 vector3f_divs()

```
VALLY_INLINE vector3f vector3f_divs (
             vector3f vector,
             f32 scalar )
```

6.19.2.23 vector3f_dot()

```
VALLY_INLINE f32 vector3f_dot (
            vector3f a,
            vector3f b )
```

6.19.2.24 vector3f_length()

```
VALLY_INLINE f32 vector3f_length (
            vector3f vector )
```

6.19.2.25 vector3f_mul()

```
VALLY_INLINE vector3f vector3f_mul (
            vector3f a,
            vector3f b )
```

6.19.2.26 vector3f_muls()

```
VALLY_INLINE vector3f vector3f_muls (
            vector3f vector,
            f32 scalar )
```

6.19.2.27 vector3f_normalize()

```
VALLY_INLINE void vector3f_normalize (
            vector3f * vector )
```

6.19.2.28 vector3f_normalized()

```
VALLY_INLINE vector3f vector3f_normalized (
            vector3f vector )
```

6.19.2.29 vector3f_sub()

```
VALLY_INLINE vector3f vector3f_sub (
              vector3f a,
              vector3f b )
```

6.19.2.30 vector3i_add()

```
VALLY_INLINE vector3i vector3i_add (
              vector3i a,
              vector3i b )
```

6.19.2.31 vector3i_sub()

```
VALLY_INLINE vector3i vector3i_sub (
              vector3i a,
              vector3i b )
```

6.19.2.32 vsin()

```
VALLY_API f32 vsin (
              f32 x )
```

6.19.2.33 vsqrt()

```
VALLY_API f32 vsqrt (
              f32 x )
```

6.19.2.34 vtan()

```
VALLY_API f32 vtan (
              f32 x )
```

6.20 math.h

Go to the documentation of this file.

```
#ifndef VALLY_MATH_VMATH_H_
#define VALLY_MATH_VMATH_H_

#include <vally/config.h>
#include <vally/math/math_types.h>

#define V_PI 3.141592653589793F
#define V_2PI V_PI * 2.0F
#define V_PI_2 V_PI / 2.0F
#define V_PI_3 V_PI / 3.0F
#define V_PI_4 V_PI / 4.0F
#define V_SQRT2 1.41421356237F

#define LERP(w, v1, v2) ((1.0 - (w)) * (v1) + (w) * (v2))

#define vector2f_create(x, y) ((vector2f){x, y})
#define vector3f_create(x, y, z) ((vector3f){x, y, z})
#define vector2i_create(x, y) ((vector2i){x, y})
#define vector3i_create(x, y, z) ((vector3i){x, y, z})

#define vector2f_zero vector2f_create(0.0f, 0.0f)
#define vector2f_one vector2f_create(1.0f, 1.0f)
#define vector2f_right vector2f_create(1.0f, 0.0f)
#define vector2f_left vector2f_create(-1.0f, 0.0f)
#define vector2f_up vector2f_create(0.0f, 1.0f)
#define vector2f_down vector2f_create(0.0f, -1.0f)

VALLY_API f32 vsin(f32 x);
VALLY_API f32 vcos(f32 x);
VALLY_API f32 vtan(f32 x);
VALLY_API f32 vasin(f32 x);
VALLY_API f32 vacos(f32 x);
VALLY_API f32 vatan(f32 x);
VALLY_API f32 vsqrt(f32 x);
VALLY_API f32 vabs(f32 x);

/* vector2f */
VALLY_INLINE vector2f vector2f_add(vector2f a, vector2f b) {
  return vector2f_create(a.x + b.x, a.y + b.y);
}

VALLY_INLINE vector2f vector2f_sub(vector2f a, vector2f b) {
  return vector2f_create(a.x - b.x, a.y - b.y);
}

VALLY_INLINE vector2f vector2f_mul(vector2f a, vector2f b) {
  return vector2f_create(a.x * b.x, a.y * b.y);
}

VALLY_INLINE vector2f vector2f_muls(vector2f vector, f32 scalar) {
  return vector2f_create(vector.x * scalar, vector.y * scalar);
}

VALLY_INLINE vector2f vector2f_divs(vector2f vector, f32 scalar) {
  return vector2f_create(vector.x / scalar, vector.y / scalar);
}

VALLY_INLINE vector2f vector2f_div(vector2f a, vector2f b) {
  return vector2f_create(a.x / b.x, a.y / b.y);
}

VALLY_INLINE f32 vector2f_length(vector2f vector) {
  return vsqrt(vector.x * vector.x + vector.y * vector.y);
}

VALLY_INLINE vector2f vector2f_normalized(vector2f vector) {
  const f32 length = vector2f_length(vector);
  if (length != 0.0f) {
    return vector2f_divs(vector, vector2f_length(vector));
  } else {
    return vector;
  }
}

VALLY_INLINE void vector2f_normalize(vector2f *vector) {
  const f32 length = vector2f_length(*vector);
  if (length != 0.0f) {
    vector->x /= length;
    vector->y /= length;
  }
}

```

```
VALLY_INLINE f32 vector2f_distance(vector2f a, vector2f b) {
  return vector2f_length(vector2f_sub(a, b));
}

/* vector3f */
VALLY_INLINE vector3f vector3f_add(vector3f a, vector3f b) {
  return vector3f_create(a.x + b.x, a.y + b.y, a.z + b.z);
}

VALLY_INLINE vector3f vector3f_sub(vector3f a, vector3f b) {
  return vector3f_create(a.x - b.x, a.y - b.y, a.z - b.z);
}

VALLY_INLINE vector3f vector3f_cross(vector3f a, vector3f b) {
  return vector3f_create((a.y * b.z - a.z * b.y), ( a.z * b.x - a.x * b.z), (a.x * b.y - a.y * b.x));
}

VALLY_INLINE f32 vector3f_dot(vector3f a, vector3f b) {
  return a.x * b.x + a.y * b.y + a.z * b.z;
}

VALLY_INLINE vector3f vector3f_mul(vector3f a, vector3f b) {
  return vector3f_create(a.x * b.x, a.y * b.y, a.z * b.z);
}

VALLY_INLINE vector3f vector3f_div(vector3f a, vector3f b) {
  return vector3f_create(a.x / b.x, a.y / b.y, a.z / b.z);
}

VALLY_INLINE vector3f vector3f_muls(vector3f vector, f32 scalar) {
  return vector3f_create(vector.x * scalar, vector.y * scalar, vector.z * scalar);
}

VALLY_INLINE vector3f vector3f_divs(vector3f vector, f32 scalar) {
  return vector3f_create(vector.x / scalar, vector.y / scalar, vector.z / scalar);
}

VALLY_INLINE f32 vector3f_length(vector3f vector) {
  return vsqrt(vector.x * vector.x + vector.y * vector.y + vector.z * vector.z);
}

VALLY_INLINE vector3f vector3f_normalized(vector3f vector) {
  return vector3f_divs(vector, vector3f_length(vector));
}

VALLY_INLINE void vector3f_normalize(vector3f *vector) {
  *vector = vector3f_normalized(*vector);
}

VALLY_INLINE f32 vector3f_distance(vector3f a, vector3f b) {
  return vector3f_length(vector3f_sub(a, b));
}

VALLY_INLINE vector2i vector2i_add(vector2i a, vector2i b) {
  return vector2i_create(a.x + b.x, a.y + b.y);
}

VALLY_INLINE vector3i vector3i_add(vector3i a, vector3i b) {
  return vector3i_create(a.x + b.x, a.y + b.y, a.z + b.z);
}

VALLY_INLINE vector2i vector2i_sub(vector2i a, vector2i b) {
  return vector2i_create(a.x - b.x, a.y - b.y);
}

VALLY_INLINE vector3i vector3i_sub(vector3i a, vector3i b) {
  return vector3i_create(a.x - b.x, a.y - b.y, a.z - b.z);
}

#endif // !VALLY_MATH_VMATH_H_
```

6.21 include/vally/math/math_types.h File Reference

```
#include <vally/config.h>
```

Data Structures

- struct vector2f

- struct vector3f
- struct vector2i
- struct vector3i

6.22 math_types.h

Go to the documentation of this file.

```
#ifndef VALLY_MATH_VMATH_TYPES_H_
#define VALLY_MATH_VMATH_TYPES_H_

#include <vally/config.h>

typedef struct {
  f32 x;
  f32 y;
} vector2f;

typedef struct {
  f32 x;
  f32 y;
  f32 z;
} vector3f;

typedef struct {
  i32 x;
  i32 y;
} vector2i;

typedef struct {
  i32 x;
  i32 y;
  i32 z;
} vector3i;

#endif // !VALLY_MATH_VMATH_TYPES_H_
```

6.23 include/vally/renderer/texture.h File Reference

```
#include <vally/config.h>
#include <vally/resources/image.h>
```

Data Structures

- struct texture

  Texture struct.

Functions

- VALLY_API texture ∗ texture_create (image ∗image)

  Creates new texture from image.
- texture ∗ texture_white_create ()
- void texture_destroy (texture ∗texture)
- void texture_bind (u32 unit, texture ∗texture)
- void texture_unbind ()

6.23.1 Function Documentation

6.23.1.1 texture_bind()

```
void texture_bind (
             u32 unit,
             texture * texture )
```

6.23.1.2 texture_create()

```
VALLY_API texture * texture_create (
             image * image )
```

Creates new texture from image.

Parameters

img	image source

Returns

textureS

6.23.1.3 texture_destroy()

```
void texture_destroy (
             texture * texture )
```

6.23.1.4 texture_unbind()

```
void texture_unbind ( )
```

6.23.1.5 texture_white_create()

```
texture * texture_white_create ( )
```

6.24 texture.h

Go to the documentation of this file.

```
//================================================================================
// Vally 0.1indev
//--------------------------------------------------------------------------------
// Copyright (c) 2022 Dmytro Zykov

// Permission is hereby granted, free of charge, to any person obtaining a copy
// of this software and associated documentation files (the "Software"), to deal
// in the Software without restriction, including without limitation the rights
// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
// copies of the Software, and to permit persons to whom the Software is
// furnished to do so, subject to the following conditions:

// The above copyright notice and this permission notice shall be included in all
// copies or substantial portions of the Software.

// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
// SOFTWARE.
//================================================================================

#pragma once

#include <vally/config.h>
#include <vally/resources/image.h>

typedef struct {
  u32 id;
  i32 width;
  i32 height;
} texture;

VALLY_API texture *texture_create(image *image);

// Creates white texture 1x1.
// Needed for colored quads "without texture".
texture *texture_white_create();

// Simple destructor
void texture_destroy(texture *texture);

// Binds OpenGL texture to given texture slot
void texture_bind(u32 unit, texture *texture);

// Unbinds texture (Actually, it’s not used, but maybe needed in the future)
void texture_unbind();
```

6.25 include/vally/resources/image.h File Reference

```
#include <vally/resources/resource_types.h>
#include <vally/config.h>
```

Functions

- VALLY_API image ∗ image_load (const char ∗path)
- void image_unload (image ∗img)

6.25.1 Function Documentation

6.25.1.1 image_load()

```
VALLY_API image * image_load (
            const char * path )
```

6.25.1.2 image_unload()

```
void image_unload (
            image * img )
```

6.26 image.h

Go to the documentation of this file.

```
#ifndef VALLY_RESOURCES_IMAGE_H_
#define VALLY_RESOURCES_IMAGE_H_

#include <vally/resources/resource_types.h>
#include <vally/config.h>

VALLY_API image *image_load(const char *path);

void image_unload(image *img);


#endif // !VALLY_RESOURCES_IMAGE_H_
```

6.27 include/vally/resources/resource_types.h File Reference

```
#include <vally/config.h>
```

Data Structures

- struct image
- struct resource

Typedefs

- typedef void(∗ destructor_fun) (void ∗)

6.27.1 Typedef Documentation

6.27.1.1 destructor_fun

```
typedef void(* destructor_fun) (void *)
```

6.28 resource_types.h

Go to the documentation of this file.

```
#ifndef VALLY_RESOURCES_RECOURCE_TYPES_H_
#define VALLY_RESOURCES_RECOURCE_TYPES_H_

#include <vally/config.h>

typedef void (*destructor_fun)(void *);

typedef struct {
  i8 *buffer;
  i32 width;
  i32 height;
  i32 channels;
} image;

typedef struct {
  void *data;
  destructor_fun destructor;
} resource;

#endif // !VALLY_RESOURCES_RECOURCE_TYPES_H_
```

6.29 include/vally/vally.h File Reference

```
#include <vally/core/engine.h>
#include <vally/core/input.h>
#include <vally/ecs/ecs.h>
```

6.30 vally.h

Go to the documentation of this file.

```
/*********************************************************************
 * vally.h                                                          *
 *                                                                   *
 * Copyright (c) 2022 Dmytro Zykov                                   *
 *                                                                   *
 * This file is a part of the vally project, and may only be used,   *
 * modified and distributed under the terms of the MIT License,      *
 * LICENSE.md. By continuing to use, modify and distribute this file *
 * you inidicate that you have read the license and accept it fully. *
 *********************************************************************/

#ifndef VALLY_H_
#define VALLY_H_

#include <vally/core/engine.h>
#include <vally/core/input.h>
#include <vally/ecs/ecs.h>

#endif // VALLY_H_
```

Index