

CS162 - Visualizer

Generated by Doxygen 1.9.6

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	9
5.1 component Namespace Reference	9
5.2 constants Namespace Reference	9
5.2.1 Variable Documentation	9
5.2.1.1 ani_speed	10
5.2.1.2 default_color_path	10
5.2.1.3 default_font_size	10
5.2.1.4 frames_per_second	10
5.2.1.5 max_val	10
5.2.1.6 min_val	10
5.2.1.7 scene_height	11
5.2.1.8 scene_width	11
5.2.1.9 sidebar_width	11
5.2.1.10 text_buffer_size	11
5.3 core Namespace Reference	11
5.4 gui Namespace Reference	11
5.5 gui::internal Namespace Reference	12
5.6 scene Namespace Reference	12
5.6.1 Typedef Documentation	13
5.6.1.1 CircularLinkedListScene	13
5.6.1.2 DoublyLinkedListScene	13
5.6.1.3 LinkedListScene	13
5.6.2 Enumeration Type Documentation	13
5.6.2.1 Sceneld	13
5.7 scene::internal Namespace Reference	14
5.8 utils Namespace Reference	14
5.8.1 Function Documentation	14
5.8.1.1 adaptive_text_color()	14
5.8.1.2 color_from_hex()	15
5.8.1.3 DrawText()	15
5.8.1.4 get_random()	16
5.8.1.5 MeasureText()	17

5.8.1.6 str_extract_data()	17
5.8.1.7 strtok()	18
5.8.1.8 unreachable()	18
5.8.1.9 val_in_range()	19
6 Class Documentation	21
6.1 scene::ArrayScene Class Reference	21
6.1.1 Detailed Description	24
6.1.2 Member Function Documentation	24
6.1.2.1 interact()	24
6.1.2.2 render()	25
6.2 gui::internal::Base Class Reference	25
6.2.1 Detailed Description	26
6.2.2 Constructor & Destructor Documentation	26
6.2.2.1 Base() [1/3]	27
6.2.2.2 Base() [2/3]	27
6.2.2.3 Base() [3/3]	27
6.2.2.4 ~Base()	27
6.2.3 Member Function Documentation	27
6.2.3.1 operator=() [1/2]	27
6.2.3.2 operator=() [2/2]	27
6.2.3.3 render()	28
6.2.3.4 update()	28
6.3 scene::BaseLinkedListScene< Con > Class Template Reference	28
6.3.1 Detailed Description	31
6.3.2 Member Function Documentation	31
6.3.2.1 interact()	31
6.3.2.2 render()	32
6.4 core::BaseList< T > Class Template Reference	32
6.4.1 Detailed Description	34
6.4.2 Member Typedef Documentation	34
6.4.2.1 Node_ptr	34
6.4.3 Constructor & Destructor Documentation	34
6.4.3.1 BaseList() [1/4]	35
6.4.3.2 BaseList() [2/4]	35
6.4.3.3 BaseList() [3/4]	35
6.4.3.4 BaseList() [4/4]	35
6.4.3.5 ~BaseList()	35
6.4.4 Member Function Documentation	35
6.4.4.1 back()	36
6.4.4.2 clean_up()	36
6.4.4.3 copy_data()	36

6.4.4.4 empty()	36
6.4.4.5 front()	36
6.4.4.6 init_first_element()	37
6.4.4.7 operator=() [1/2]	37
6.4.4.8 operator=() [2/2]	37
6.4.4.9 pop_back()	37
6.4.4.10 pop_front()	37
6.4.4.11 push_back()	38
6.4.4.12 push_front()	38
6.4.4.13 size()	38
6.4.5 Member Data Documentation	38
6.4.5.1 m_head	38
6.4.5.2 m_size	38
6.4.5.3 m_tail	39
6.5 scene::internal::BaseScene Class Reference	39
6.5.1 Detailed Description	40
6.5.2 Constructor & Destructor Documentation	40
6.5.2.1 BaseScene() [1/3]	40
6.5.2.2 BaseScene() [2/3]	41
6.5.2.3 BaseScene() [3/3]	41
6.5.2.4 ~BaseScene()	41
6.5.3 Member Function Documentation	41
6.5.3.1 interact()	41
6.5.3.2 operator=() [1/2]	41
6.5.3.3 operator=() [2/2]	42
6.5.3.4 render()	42
6.5.3.5 render_go_button()	42
6.5.3.6 render_inputs()	42
6.5.3.7 render_options()	43
6.5.4 Member Data Documentation	43
6.5.4.1 button_size	43
6.5.4.2 head_offset	44
6.5.4.3 m_code_highlighter	44
6.5.4.4 m_edit_action	44
6.5.4.5 m_edit_mode	44
6.5.4.6 m_file_dialog	44
6.5.4.7 m_index_input	44
6.5.4.8 m_sequence_controller	45
6.5.4.9 m_text_input	45
6.5.4.10 options_head	45
6.6 component::CodeHighlighter Class Reference	45
6.6.1 Detailed Description	46

6.6.2 Member Function Documentation	46
6.6.2.1 clear()	46
6.6.2.2 highlight()	47
6.6.2.3 push_into_sequence()	47
6.6.2.4 render()	48
6.6.2.5 set_code()	49
6.7 core::Deque< T > Class Template Reference	49
6.7.1 Detailed Description	52
6.7.2 Member Function Documentation	53
6.7.2.1 back()	53
6.7.2.2 empty()	53
6.7.2.3 front()	54
6.7.2.4 pop_back()	54
6.7.2.5 pop_front()	55
6.7.2.6 push_back()	55
6.7.2.7 push_front()	56
6.7.2.8 size()	56
6.8 core::DoublyLinkedList< T > Class Template Reference	57
6.8.1 Detailed Description	60
6.8.2 Member Typedef Documentation	60
6.8.2.1 Base	60
6.8.2.2 cNode_ptr	60
6.8.2.3 Node	60
6.8.2.4 Node_ptr	61
6.8.3 Member Function Documentation	61
6.8.3.1 at() [1/2]	61
6.8.3.2 at() [2/2]	61
6.8.3.3 clear()	62
6.8.3.4 empty()	62
6.8.3.5 find() [1/2]	62
6.8.3.6 find() [2/2]	63
6.8.3.7 insert()	63
6.8.3.8 internal_find()	63
6.8.3.9 internal_search()	63
6.8.3.10 remove()	64
6.8.3.11 search() [1/2]	64
6.8.3.12 search() [2/2]	64
6.8.3.13 size()	65
6.8.4 Member Data Documentation	65
6.8.4.1 m_head	65
6.8.4.2 m_size	65
6.8.4.3 m_tail	65

6.9 scene::DynamicArrayScene Class Reference	66
6.9.1 Detailed Description	68
6.9.2 Member Function Documentation	68
6.9.2.1 interact()	68
6.9.2.2 render()	69
6.10 component::FileDialog Class Reference	69
6.10.1 Detailed Description	70
6.10.2 Constructor & Destructor Documentation	70
6.10.2.1 FileDialog() [1/2]	71
6.10.2.2 FileDialog() [2/2]	71
6.10.3 Member Function Documentation	71
6.10.3.1 extract_values()	71
6.10.3.2 get_path()	72
6.10.3.3 is_active()	72
6.10.3.4 render()	72
6.10.3.5 render_head()	73
6.10.3.6 set_message()	73
6.10.3.7 set_mode_open()	73
6.10.3.8 set_mode_save()	73
6.10.3.9 set_title()	74
6.10.4 Member Data Documentation	74
6.10.4.1 size	74
6.11 gui::GuiArray< T, N > Class Template Reference	74
6.11.1 Detailed Description	77
6.11.2 Constructor & Destructor Documentation	77
6.11.2.1 GuiArray() [1/2]	77
6.11.2.2 GuiArray() [2/2]	77
6.11.3 Member Function Documentation	77
6.11.3.1 operator[]() [1/2]	78
6.11.3.2 operator[]() [2/2]	78
6.11.3.3 render()	78
6.11.3.4 set_color_index()	78
6.11.3.5 update()	79
6.12 gui::GuiCircularLinkedList< T > Class Template Reference	79
6.12.1 Detailed Description	83
6.12.2 Constructor & Destructor Documentation	83
6.12.2.1 GuiCircularLinkedList()	84
6.12.3 Member Function Documentation	84
6.12.3.1 init_label()	84
6.12.3.2 insert()	84
6.12.3.3 render()	85
6.12.3.4 update()	85

6.13 <code>gui::GuiDoublyLinkedList< T ></code> Class Template Reference	85
6.13.1 Detailed Description	89
6.13.2 Constructor & Destructor Documentation	89
6.13.2.1 <code>GuiDoublyLinkedList()</code>	90
6.13.3 Member Function Documentation	90
6.13.3.1 <code>init_label()</code>	90
6.13.3.2 <code>insert()</code>	90
6.13.3.3 <code>render()</code>	91
6.13.3.4 <code>update()</code>	91
6.14 <code>gui::GuiDynamicArray< T ></code> Class Template Reference	91
6.14.1 Detailed Description	94
6.14.2 Constructor & Destructor Documentation	94
6.14.2.1 <code>GuiDynamicArray()</code> [1/4]	94
6.14.2.2 <code>GuiDynamicArray()</code> [2/4]	94
6.14.2.3 <code>GuiDynamicArray()</code> [3/4]	95
6.14.2.4 <code>GuiDynamicArray()</code> [4/4]	95
6.14.2.5 <code>~GuiDynamicArray()</code>	95
6.14.3 Member Function Documentation	95
6.14.3.1 <code>capacity()</code>	95
6.14.3.2 <code>operator=()</code> [1/2]	95
6.14.3.3 <code>operator=()</code> [2/2]	96
6.14.3.4 <code>operator[]()</code> [1/2]	96
6.14.3.5 <code>operator[]()</code> [2/2]	96
6.14.3.6 <code>pop()</code>	96
6.14.3.7 <code>push()</code>	96
6.14.3.8 <code>realloc()</code>	97
6.14.3.9 <code>render()</code>	97
6.14.3.10 <code>set_color_index()</code>	98
6.14.3.11 <code>size()</code>	98
6.14.3.12 <code>update()</code>	98
6.15 <code>gui::GuiElement< T ></code> Class Template Reference	99
6.15.1 Detailed Description	99
6.15.2 Constructor & Destructor Documentation	100
6.15.2.1 <code>GuiElement()</code> [1/2]	100
6.15.2.2 <code>GuiElement()</code> [2/2]	100
6.15.3 Member Function Documentation	100
6.15.3.1 <code>get_pos()</code>	100
6.15.3.2 <code>get_value()</code> [1/2]	100
6.15.3.3 <code>get_value()</code> [2/2]	100
6.15.3.4 <code>render()</code>	101
6.15.3.5 <code>set_color_index()</code>	101
6.15.3.6 <code>set_index()</code>	102

6.15.3.7 set_pos()	102
6.15.3.8 set_value()	102
6.15.4 Member Data Documentation	102
6.15.4.1 init_pos	102
6.15.4.2 side	103
6.16 gui::GuiLinkedList< T > Class Template Reference	103
6.16.1 Detailed Description	107
6.16.2 Constructor & Destructor Documentation	107
6.16.2.1 GuiLinkedList()	108
6.16.3 Member Function Documentation	108
6.16.3.1 init_label()	108
6.16.3.2 insert()	108
6.16.3.3 render()	109
6.16.3.4 update()	109
6.17 gui::GuiNode< T > Class Template Reference	109
6.17.1 Detailed Description	110
6.17.2 Constructor & Destructor Documentation	110
6.17.2.1 GuiNode()	110
6.17.3 Member Function Documentation	110
6.17.3.1 get_pos()	110
6.17.3.2 get_value()	111
6.17.3.3 render()	111
6.17.3.4 set_color_index()	111
6.17.3.5 set_label()	112
6.17.3.6 set_pos()	112
6.17.3.7 set_value()	112
6.17.4 Member Data Documentation	112
6.17.4.1 radius	112
6.18 gui::GuiQueue< T > Class Template Reference	113
6.18.1 Detailed Description	116
6.18.2 Constructor & Destructor Documentation	116
6.18.2.1 GuiQueue()	116
6.18.3 Member Function Documentation	117
6.18.3.1 init_label()	117
6.18.3.2 pop()	117
6.18.3.3 pop_back()	117
6.18.3.4 push()	117
6.18.3.5 push_front()	118
6.18.3.6 render()	118
6.18.3.7 update()	118
6.19 gui::GuiStack< T > Class Template Reference	119
6.19.1 Detailed Description	122

6.19.2 Constructor & Destructor Documentation	122
6.19.2.1 GuiStack()	122
6.19.3 Member Function Documentation	122
6.19.3.1 init_label()	123
6.19.3.2 pop()	123
6.19.3.3 push()	123
6.19.3.4 render()	124
6.19.3.5 update()	124
6.20 component::MenuItem Class Reference	124
6.20.1 Detailed Description	125
6.20.2 Constructor & Destructor Documentation	125
6.20.2.1 MenuItem() [1/2]	126
6.20.2.2 MenuItem() [2/2]	126
6.20.3 Member Function Documentation	126
6.20.3.1 clicked()	126
6.20.3.2 render()	126
6.20.3.3 reset()	126
6.20.3.4 x()	127
6.20.3.5 y()	127
6.20.4 Member Data Documentation	127
6.20.4.1 block_height	127
6.20.4.2 block_width	127
6.20.4.3 button_height	127
6.20.4.4 button_width	128
6.21 scene::MenuScene Class Reference	128
6.21.1 Detailed Description	130
6.21.2 Constructor & Destructor Documentation	130
6.21.2.1 MenuScene()	130
6.21.3 Member Function Documentation	130
6.21.3.1 interact()	131
6.21.3.2 render()	131
6.22 core::BaseList< T >::Node Struct Reference	132
6.22.1 Detailed Description	132
6.22.2 Member Data Documentation	132
6.22.2.1 data	133
6.22.2.2 next	133
6.22.2.3 prev	133
6.23 core::Queue< T > Class Template Reference	133
6.23.1 Detailed Description	136
6.23.2 Member Function Documentation	137
6.23.2.1 back()	137
6.23.2.2 empty()	137

6.23.2.3 front()	137
6.23.2.4 pop()	137
6.23.2.5 pop_back()	137
6.23.2.6 push()	138
6.23.2.7 push_front()	138
6.23.2.8 size()	138
6.24 scene::QueueScene Class Reference	139
6.24.1 Detailed Description	141
6.24.2 Member Function Documentation	141
6.24.2.1 interact()	141
6.24.2.2 render()	142
6.25 scene::internal::SceneOptions Struct Reference	142
6.25.1 Detailed Description	144
6.25.2 Member Data Documentation	144
6.25.2.1 action_labels	144
6.25.2.2 action_selection	144
6.25.2.3 max_size	144
6.25.2.4 mode_labels	144
6.25.2.5 mode_selection	144
6.26 scene::SceneRegistry Class Reference	145
6.26.1 Detailed Description	145
6.26.2 Constructor & Destructor Documentation	146
6.26.2.1 SceneRegistry() [1/2]	146
6.26.2.2 SceneRegistry() [2/2]	146
6.26.2.3 ~SceneRegistry()	146
6.26.3 Member Function Documentation	146
6.26.3.1 close_window()	146
6.26.3.2 get_instance()	147
6.26.3.3 get_scene()	147
6.26.3.4 interact()	148
6.26.3.5 operator=() [1/2]	148
6.26.3.6 operator=() [2/2]	148
6.26.3.7 render()	149
6.26.3.8 set_scene()	149
6.26.3.9 should_close()	150
6.27 component::SequenceController Class Reference	150
6.27.1 Detailed Description	151
6.27.2 Member Function Documentation	151
6.27.2.1 get_anim_counter()	151
6.27.2.2 get_anim_frame()	152
6.27.2.3 get_progress_value()	152
6.27.2.4 get_run_all()	153

6.27.2.5 get_speed_scale()	154
6.27.2.6 inc_anim_counter()	154
6.27.2.7 interact()	155
6.27.2.8 render()	156
6.27.2.9 reset_anim_counter()	156
6.27.2.10 set_max_value()	157
6.27.2.11 set_progress_value()	157
6.27.2.12 set_rerun()	158
6.27.2.13 set_run_all()	158
6.28 Settings Class Reference	159
6.28.1 Detailed Description	160
6.28.2 Constructor & Destructor Documentation	160
6.28.2.1 Settings() [1/2]	160
6.28.2.2 Settings() [2/2]	160
6.28.2.3 ~Settings()	161
6.28.3 Member Function Documentation	161
6.28.3.1 get_color() [1/2]	161
6.28.3.2 get_color() [2/2]	162
6.28.3.3 get_instance()	162
6.28.3.4 operator=() [1/2]	162
6.28.3.5 operator=() [2/2]	163
6.28.3.6 save_to_file()	163
6.28.4 Member Data Documentation	163
6.28.4.1 default_color	163
6.28.4.2 num_color	164
6.29 scene::SettingsScene Class Reference	164
6.29.1 Detailed Description	166
6.29.2 Constructor & Destructor Documentation	166
6.29.2.1 SettingsScene()	166
6.29.3 Member Function Documentation	166
6.29.3.1 interact()	167
6.29.3.2 render()	167
6.30 component::SideBar Class Reference	168
6.30.1 Detailed Description	168
6.30.2 Member Function Documentation	168
6.30.2.1 interact()	169
6.30.2.2 render()	169
6.31 core::Stack< T > Class Template Reference	170
6.31.1 Detailed Description	174
6.31.2 Member Typedef Documentation	174
6.31.2.1 Base	174
6.31.3 Member Function Documentation	174

6.31.3.1 empty()	174
6.31.3.2 pop()	174
6.31.3.3 push()	174
6.31.3.4 size()	175
6.31.3.5 top()	175
6.31.4 Member Data Documentation	175
6.31.4.1 m_head	175
6.31.4.2 m_tail	175
6.32 scene::StackScene Class Reference	176
6.32.1 Detailed Description	178
6.32.2 Member Function Documentation	178
6.32.2.1 interact()	178
6.32.2.2 render()	179
6.33 component::TextInput Class Reference	179
6.33.1 Detailed Description	180
6.33.2 Constructor & Destructor Documentation	180
6.33.2.1 TextInput() [1/2]	180
6.33.2.2 TextInput() [2/2]	181
6.33.3 Member Function Documentation	181
6.33.3.1 extract_values()	181
6.33.3.2 interact()	182
6.33.3.3 render()	182
6.33.3.4 set_random_max()	183
6.33.3.5 set_random_min()	184
6.33.4 Member Data Documentation	184
6.33.4.1 size	184
7 File Documentation	185
7.1 src/component/code_highlighter.cpp File Reference	185
7.2 code_highlighter.cpp	185
7.3 src/component/code_highlighter.hpp File Reference	186
7.4 code_highlighter.hpp	187
7.5 src/component/file_dialog.cpp File Reference	188
7.6 file_dialog.cpp	188
7.7 src/component/file_dialog.hpp File Reference	189
7.8 file_dialog.hpp	190
7.9 src/component/menu_item.cpp File Reference	191
7.10 menu_item.cpp	191
7.11 src/component/menu_item.hpp File Reference	192
7.12 menu_item.hpp	193
7.13 src/component/sequence_controller.cpp File Reference	193
7.14 sequence_controller.cpp	194

7.15 src/component/sequence_controller.hpp File Reference	195
7.16 sequence_controller.hpp	196
7.17 src/component/sidebar.cpp File Reference	197
7.18 sidebar.cpp	197
7.19 src/component/sidebar.hpp File Reference	198
7.20 sidebar.hpp	199
7.21 src/component/text_input.cpp File Reference	199
7.22 text_input.cpp	200
7.23 src/component/text_input.hpp File Reference	201
7.24 text_input.hpp	202
7.25 src/constants.hpp File Reference	202
7.26 constants.hpp	203
7.27 src/core/base_list.hpp File Reference	203
7.28 base_list.hpp	204
7.29 src/core/deque.hpp File Reference	207
7.30 deque.hpp	207
7.31 src/core/deque.test.cpp File Reference	208
7.31.1 Function Documentation	209
7.31.1.1 __attribute__((__test__()))	209
7.31.1.2 TEST_CASE() [1/2]	209
7.31.1.3 TEST_CASE() [2/2]	210
7.31.2 Variable Documentation	210
7.31.2.1 list	210
7.32 deque.test.cpp	211
7.33 src/core/doubly_linked_list.hpp File Reference	212
7.34 doubly_linked_list.hpp	213
7.35 src/core/doubly_linked_list.test.cpp File Reference	215
7.35.1 Function Documentation	216
7.35.1.1 TEST_CASE()	216
7.36 doubly_linked_list.test.cpp	217
7.37 src/core/queue.hpp File Reference	218
7.38 queue.hpp	219
7.39 src/core/stack.hpp File Reference	220
7.40 stack.hpp	221
7.41 src/doctest_main.cpp File Reference	221
7.41.1 Macro Definition Documentation	222
7.41.1.1 DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN	222
7.42 doctest_main.cpp	222
7.43 src/gui/array_gui.hpp File Reference	222
7.44 array_gui.hpp	223
7.45 src/gui/base_gui.hpp File Reference	224
7.46 base_gui.hpp	225

7.47 src/gui/circular_linked_list_gui.hpp File Reference	226
7.48 circular_linked_list_gui.hpp	227
7.49 src/gui/doubly_linked_list_gui.hpp File Reference	228
7.50 doubly_linked_list_gui.hpp	230
7.51 src/gui/dynamic_array_gui.hpp File Reference	231
7.52 dynamic_array_gui.hpp	232
7.53 src/gui/element_gui.hpp File Reference	235
7.54 element_gui.hpp	236
7.55 src/gui/linked_list_gui.hpp File Reference	238
7.56 linked_list_gui.hpp	239
7.57 src/gui/node_gui.hpp File Reference	240
7.58 node_gui.hpp	241
7.59 src/gui/queue_gui.hpp File Reference	243
7.60 queue_gui.hpp	244
7.61 src/gui/stack_gui.hpp File Reference	246
7.62 stack_gui.hpp	247
7.63 src/main.cpp File Reference	249
7.63.1 Function Documentation	249
7.63.1.1 main()	249
7.64 main.cpp	250
7.65 src/raygui_impl.cpp File Reference	250
7.65.1 Macro Definition Documentation	251
7.65.1.1 GUI_FILE_DIALOG_IMPLEMENTATION	251
7.65.1.2 RAYGUI_IMPLEMENTATION	251
7.66 raygui_impl.cpp	251
7.67 src/scene/array_scene.cpp File Reference	251
7.68 array_scene.cpp	252
7.69 src/scene/array_scene.hpp File Reference	255
7.70 array_scene.hpp	256
7.71 src/scene/base_linked_list_scene.hpp File Reference	257
7.72 base_linked_list_scene.hpp	258
7.73 src/scene/base_scene.cpp File Reference	267
7.74 base_scene.cpp	267
7.75 src/scene/base_scene.hpp File Reference	268
7.76 base_scene.hpp	269
7.77 src/scene/dynamic_array_scene.cpp File Reference	270
7.78 dynamic_array_scene.cpp	270
7.79 src/scene/dynamic_array_scene.hpp File Reference	274
7.80 dynamic_array_scene.hpp	275
7.81 src/scene/menu_scene.cpp File Reference	276
7.82 menu_scene.cpp	277
7.83 src/scene/menu_scene.hpp File Reference	279

7.84 menu_scene.hpp	280
7.85 src/scene/queue_scene.cpp File Reference	280
7.86 queue_scene.cpp	281
7.87 src/scene/queue_scene.hpp File Reference	284
7.88 queue_scene.hpp	285
7.89 src/scene/scene_options.hpp File Reference	286
7.90 scene_options.hpp	287
7.91 src/scene/scene_registry.cpp File Reference	287
7.92 scene_registry.cpp	288
7.93 src/scene/scene_registry.hpp File Reference	288
7.94 scene_registry.hpp	289
7.95 src/scene/settings_scene.cpp File Reference	290
7.96 settings_scene.cpp	290
7.97 src/scene/settings_scene.hpp File Reference	293
7.98 settings_scene.hpp	294
7.99 src/scene/stack_scene.cpp File Reference	294
7.100 stack_scene.cpp	295
7.101 src/scene/stack_scene.hpp File Reference	298
7.102 stack_scene.hpp	299
7.103 src/settings.cpp File Reference	300
7.104 settings.cpp	300
7.105 src/settings.hpp File Reference	300
7.106 settings.hpp	301
7.107 src/utils.cpp File Reference	302
7.108 utils.cpp	303
7.109 src/utils.hpp File Reference	304
7.110 utils.hpp	305

Index**307**

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

component	9
constants	9
core	11
gui	11
gui::internal	12
scene	12
scene::internal	14
utils	14

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

gui::internal::Base	25
gui::GuiArray< int, max_size >	74
gui::GuiDynamicArray< int >	91
gui::GuiQueue< int >	113
gui::GuiStack< int >	119
gui::GuiArray< T, N >	74
gui::GuiCircularLinkedList< T >	79
gui::GuiDoublyLinkedList< T >	85
gui::GuiDynamicArray< T >	91
gui::GuiLinkedList< T >	103
gui::GuiQueue< T >	113
gui::GuiStack< T >	119
core::BaseList< T >	32
core::DoublyLinkedList< GuiNode< T > >	57
gui::GuiCircularLinkedList< T >	79
gui::GuiDoublyLinkedList< T >	85
gui::GuiLinkedList< T >	103
core::DoublyLinkedList< const char * >	57
core::DoublyLinkedList< int >	57
core::DoublyLinkedList< gui::GuiArray< int, max_size > >	57
core::DoublyLinkedList< Con >	57
core::DoublyLinkedList< gui::GuiDynamicArray< int > >	57
core::DoublyLinkedList< gui::GuiQueue< int > >	57
core::DoublyLinkedList< gui::GuiStack< int > >	57
core::Queue< GuiNode< T > >	133
gui::GuiQueue< T >	113
core::Queue< GuiNode< int > >	133
core::Stack< GuiNode< T > >	170
gui::GuiStack< T >	119
core::Stack< GuiNode< int > >	170
core::Deque< T >	49
core::DoublyLinkedList< T >	57
core::Queue< T >	133
gui::GuiQueue< int >	113

core::Stack< T >	170
gui::GuiStack< int >	119
core::BaseList< Con >	32
core::BaseList< const char * >	32
core::BaseList< gui::GuiArray< int, max_size > >	32
core::BaseList< gui::GuiDynamicArray< int > >	32
core::BaseList< gui::GuiQueue< int > >	32
core::BaseList< gui::GuiStack< int > >	32
core::BaseList< GuiNode< int > >	32
core::BaseList< GuiNode< T > >	32
core::BaseList< int >	32
scene::internal::BaseScene	39
scene::ArrayScene	21
scene::BaseLinkedListScene< Con >	28
scene::DynamicArrayScene	66
scene::MenuScene	128
scene::QueueScene	139
scene::SettingsScene	164
scene::StackScene	176
component::CodeHighlighter	45
component::FileDialog	69
gui::GuiElement< T >	99
gui::GuiElement< int >	99
gui::GuiNode< T >	109
component::MenuItem	124
core::BaseList< T >::Node	132
scene::internal::SceneOptions	142
scene::SceneRegistry	145
component::SequenceController	150
Settings	159
component::SideBar	168
component::TextInput	179

Chapter 3

Class Index

3.1 Class List

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

scene::ArrayScene	21
gui::internal::Base	25
scene::BaseLinkedListScene< Con >	28
core::BaseList< T >	32
scene::internal::BaseScene	39
component::CodeHighlighter	45
core::Deque< T >	49
core::DoublyLinkedList< T >	57
scene::DynamicArrayScene	66
component::FileDialog	69
gui::GuiArray< T, N >	74
gui::GuiCircularLinkedList< T >	79
gui::GuiDoublyLinkedList< T >	85
gui::GuiDynamicArray< T >	91
gui::GuiElement< T >	99
gui::GuiLinkedList< T >	103
gui::GuiNode< T >	109
gui::GuiQueue< T >	113
gui::GuiStack< T >	119
component::MenuItem	124
scene::MenuScene	128
core::BaseList< T >::Node	132
core::Queue< T >	133
scene::QueueScene	139
scene::internal::SceneOptions	142
scene::SceneRegistry	145
component::SequenceController	150
Settings	159
scene::SettingsScene	164
component::SideBar	168
core::Stack< T >	170
scene::StackScene	176
component::TextInput	179

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/constants.hpp	202
src/doctest_main.cpp	221
src/main.cpp	249
src/raygui_impl.cpp	250
src/settings.cpp	300
src/settings.hpp	300
src/utils.cpp	302
src/utils.hpp	304
src/component/code_highlighter.cpp	185
src/component/code_highlighter.hpp	186
src/component/file_dialog.cpp	188
src/component/file_dialog.hpp	189
src/component/menu_item.cpp	191
src/component/menu_item.hpp	192
src/component/sequence_controller.cpp	193
src/component/sequence_controller.hpp	195
src/component/sidebar.cpp	197
src/component/sidebar.hpp	198
src/component/text_input.cpp	199
src/component/text_input.hpp	201
src/core/base_list.hpp	203
src/core/deque.hpp	207
src/core/deque.test.cpp	208
src/core/doubly_linked_list.hpp	212
src/core/doubly_linked_list.test.cpp	215
src/core/queue.hpp	218
src/core/stack.hpp	220
src/gui/array_gui.hpp	222
src/gui/base_gui.hpp	224
src/gui/circular_linked_list_gui.hpp	226
src/gui/doubly_linked_list_gui.hpp	228
src/gui/dynamic_array_gui.hpp	231
src/gui/element_gui.hpp	235
src/gui/linked_list_gui.hpp	238
src/gui/node_gui.hpp	240

src/gui/queue_gui.hpp	243
src/gui/stack_gui.hpp	246
src/scene/array_scene.cpp	251
src/scene/array_scene.hpp	255
src/scene/base_linked_list_scene.hpp	257
src/scene/base_scene.cpp	267
src/scene/base_scene.hpp	268
src/scene/dynamic_array_scene.cpp	270
src/scene/dynamic_array_scene.hpp	274
src/scene/menu_scene.cpp	276
src/scene/menu_scene.hpp	279
src/scene/queue_scene.cpp	280
src/scene/queue_scene.hpp	284
src/scene/scene_options.hpp	286
src/scene/scene_registry.cpp	287
src/scene/scene_registry.hpp	288
src/scene/settings_scene.cpp	290
src/scene/settings_scene.hpp	293
src/scene/stack_scene.cpp	294
src/scene/stack_scene.hpp	298

Chapter 5

Namespace Documentation

5.1 component Namespace Reference

Classes

- class [CodeHighlighter](#)
- class [FileDialog](#)
- class [MenuItem](#)
- class [SequenceController](#)
- class [SideBar](#)
- class [TextInput](#)

5.2 constants Namespace Reference

Variables

- constexpr int [scene_width](#) = 1366
- constexpr int [scene_height](#) = 768
- constexpr int [frames_per_second](#) = 30
- constexpr int [sidebar_width](#) = 256
- constexpr int [ani_speed](#) = 8
- constexpr int [text_buffer_size](#) = 512
- constexpr int [min_val](#) = 0
- constexpr int [max_val](#) = 999
- constexpr int [default_font_size](#) = 60
- constexpr const char * [default_color_path](#) = "data/color.bin"

5.2.1 Variable Documentation

5.2.1.1 ani_speed

```
constexpr int constants::ani_speed = 8 [constexpr]
```

Definition at line 11 of file [constants.hpp](#).

5.2.1.2 default_color_path

```
constexpr const char* constants::default_color_path = "data/color.bin" [constexpr]
```

Definition at line 20 of file [constants.hpp](#).

5.2.1.3 default_font_size

```
constexpr int constants::default_font_size = 60 [constexpr]
```

Definition at line 18 of file [constants.hpp](#).

5.2.1.4 frames_per_second

```
constexpr int constants::frames_per_second = 30 [constexpr]
```

Definition at line 8 of file [constants.hpp](#).

5.2.1.5 max_val

```
constexpr int constants::max_val = 999 [constexpr]
```

Definition at line 16 of file [constants.hpp](#).

5.2.1.6 min_val

```
constexpr int constants::min_val = 0 [constexpr]
```

Definition at line 15 of file [constants.hpp](#).

5.2.1.7 scene_height

```
constexpr int constants::scene_height = 768 [constexpr]
```

Definition at line 7 of file [constants.hpp](#).

5.2.1.8 scene_width

```
constexpr int constants::scene_width = 1366 [constexpr]
```

Definition at line 6 of file [constants.hpp](#).

5.2.1.9 sidebar_width

```
constexpr int constants::sidebar_width = 256 [constexpr]
```

Definition at line 10 of file [constants.hpp](#).

5.2.1.10 text_buffer_size

```
constexpr int constants::text_buffer_size = 512 [constexpr]
```

Definition at line 13 of file [constants.hpp](#).

5.3 core Namespace Reference

Classes

- class [BaseList](#)
- class [Deque](#)
- class [DoublyLinkedList](#)
- class [Queue](#)
- class [Stack](#)

5.4 gui Namespace Reference

Namespaces

- namespace [internal](#)

Classes

- class [GuiArray](#)
- class [GuiCircularLinkedList](#)
- class [GuiDoublyLinkedList](#)
- class [GuiDynamicArray](#)
- class [GuiElement](#)
- class [GuiLinkedList](#)
- class [GuiNode](#)
- class [GuiQueue](#)
- class [GuiStack](#)

5.5 gui::internal Namespace Reference

Classes

- class [Base](#)

5.6 scene Namespace Reference

Namespaces

- namespace [internal](#)

Classes

- class [ArrayScene](#)
- class [BaseLinkedListScene](#)
- class [DynamicArrayScene](#)
- class [MenuScene](#)
- class [QueueScene](#)
- class [SceneRegistry](#)
- class [SettingsScene](#)
- class [StackScene](#)

Typedefs

- using [LinkedListScene](#) = [BaseLinkedListScene](#)< [gui::GuiLinkedList](#)< int > >
- using [DoublyLinkedListScene](#) = [BaseLinkedListScene](#)< [gui::GuiDoublyLinkedList](#)< int > >
- using [CircularLinkedListScene](#) = [BaseLinkedListScene](#)< [gui::GuiCircularLinkedList](#)< int > >

Enumerations

- enum [Sceneld](#) {
 [Array](#) , [DynamicArray](#) , [LinkedList](#) , [DoublyLinkedList](#) ,
 [CircularLinkedList](#) , [Stack](#) , [Queue](#) , [Menu](#) ,
 [Settings](#) }

5.6.1 Typedef Documentation

5.6.1.1 CircularLinkedListScene

```
using scene::CircularLinkedListScene = typedef BaseLinkedListScene<gui::GuiCircularLinkedList<int>  
>
```

Definition at line 98 of file [base_linked_list_scene.hpp](#).

5.6.1.2 DoublyLinkedListScene

```
using scene::DoublyLinkedListScene = typedef BaseLinkedListScene<gui::GuiDoublyLinkedList<int>  
>
```

Definition at line 96 of file [base_linked_list_scene.hpp](#).

5.6.1.3 LinkedListScene

```
using scene::LinkedListScene = typedef BaseLinkedListScene<gui::GuiLinkedList<int> >
```

Definition at line 95 of file [base_linked_list_scene.hpp](#).

5.6.2 Enumeration Type Documentation

5.6.2.1 Sceneld

```
enum scene::SceneId
```

Enumerator

Array	
DynamicArray	
LinkedList	
DoublyLinkedList	
CircularLinkedList	
Stack	
Queue	
Menu	
Settings	

Definition at line 18 of file [scene_registry.hpp](#).

5.7 scene::internal Namespace Reference

Classes

- class [BaseScene](#)
- struct [SceneOptions](#)

5.8 utils Namespace Reference

Functions

- void [DrawText](#) (const char *text, Vector2 pos, Color color, float font_size, float spacing)
- Vector2 [MeasureText](#) (const char *text, float font_size, float spacing)
- [core::Deque](#)< int > [str_extract_data](#) (char str[[constants::text_buffer_size](#)])
- bool [val_in_range](#) (int num)
- void [unreachable](#) ()
- char * [strtok](#) (char *str, const char *delim, char **save_ptr)
- Color [color_from_hex](#) (const std::string &hex)
- Color [adaptive_text_color](#) (Color bg_color)
- template<typename T >
T [get_random](#) (T low, T high)

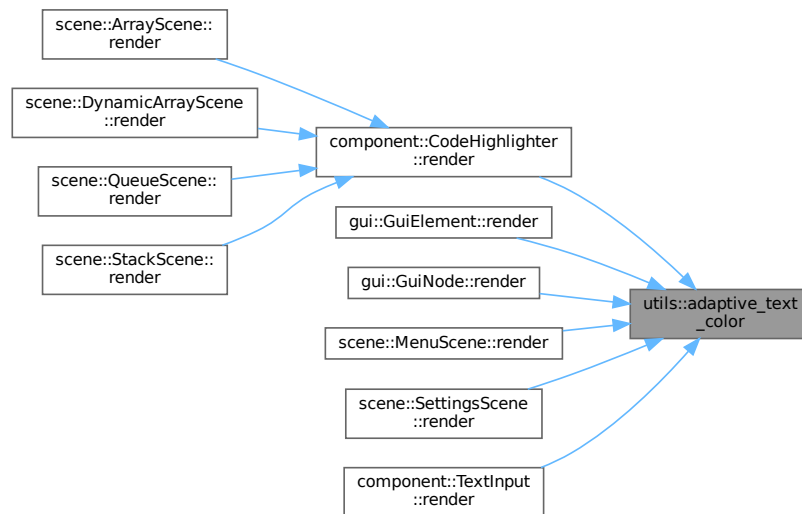
5.8.1 Function Documentation

5.8.1.1 adaptive_text_color()

```
Color utils::adaptive_text_color (  
    Color bg_color )
```

Definition at line 90 of file [utils.cpp](#).

Here is the caller graph for this function:



5.8.1.2 color_from_hex()

```
Color utils::color_from_hex (
    const std::string & hex )
```

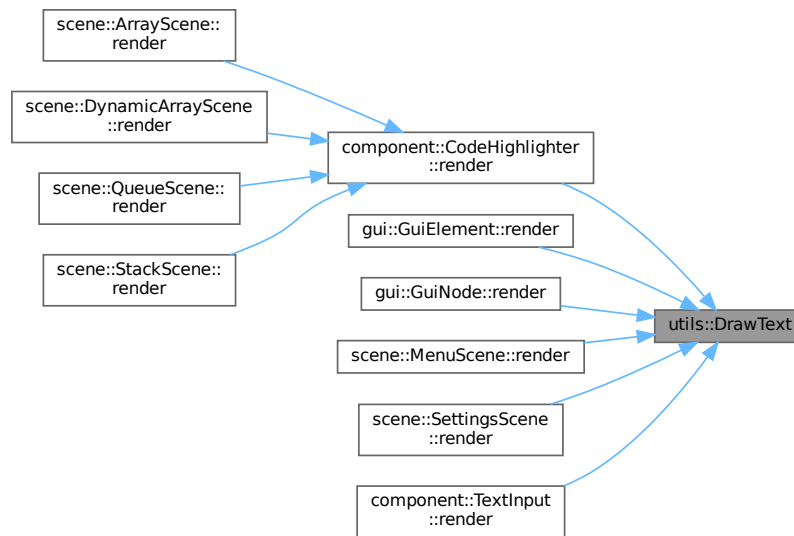
Definition at line 82 of file [utils.cpp](#).

5.8.1.3 DrawText()

```
void utils::DrawText (
    const char * text,
    Vector2 pos,
    Color color,
    float font_size,
    float spacing )
```

Definition at line 14 of file [utils.cpp](#).

Here is the caller graph for this function:



5.8.1.4 get_random()

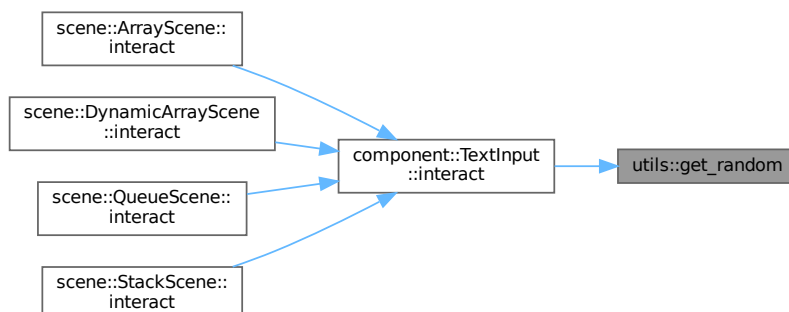
```

template<typename T>
T utils::get_random (
    T low,
    T high )

```

Definition at line 19 of file [utils.hpp](#).

Here is the caller graph for this function:

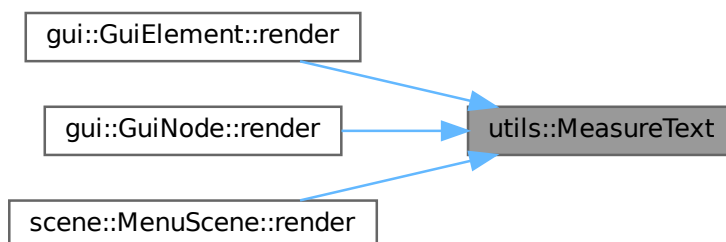


5.8.1.5 MeasureText()

```
Vector2 utils::MeasureText (
    const char * text,
    float font_size,
    float spacing )
```

Definition at line 23 of file [utils.cpp](#).

Here is the caller graph for this function:

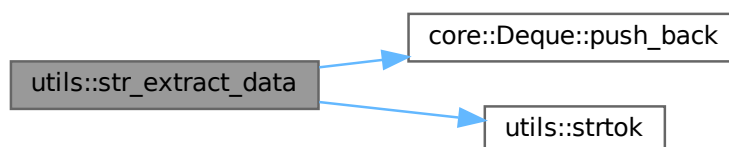


5.8.1.6 str_extract_data()

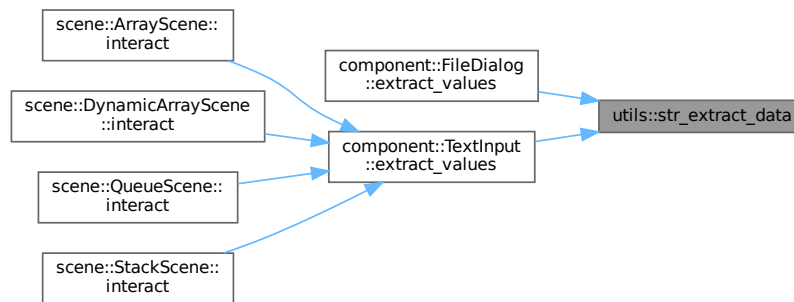
```
core::Deque< int > utils::str_extract_data (
    char str[constants::text_buffer_size] )
```

Definition at line 30 of file [utils.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

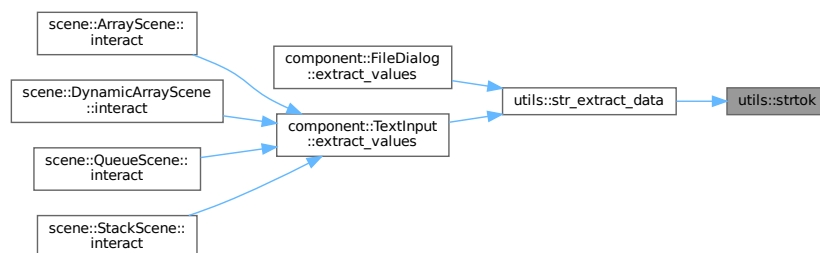


5.8.1.7 strtok()

```
char * utils::strtok (
    char * str,
    const char * delim,
    char ** save_ptr )
```

Definition at line 73 of file [utils.cpp](#).

Here is the caller graph for this function:

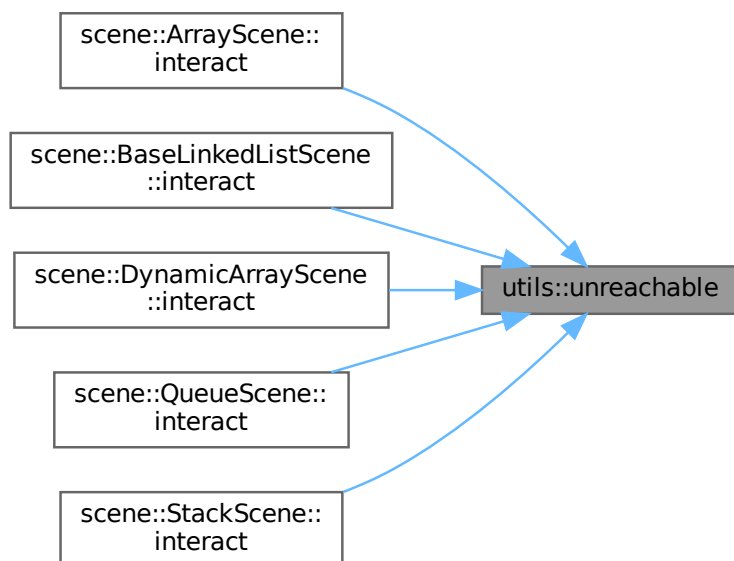


5.8.1.8 unreachable()

```
void utils::unreachable ( )
```

Definition at line 65 of file [utils.cpp](#).

Here is the caller graph for this function:



5.8.1.9 val_in_range()

```
bool utils::val_in_range (
    int num )
```

Definition at line 61 of file [utils.cpp](#).

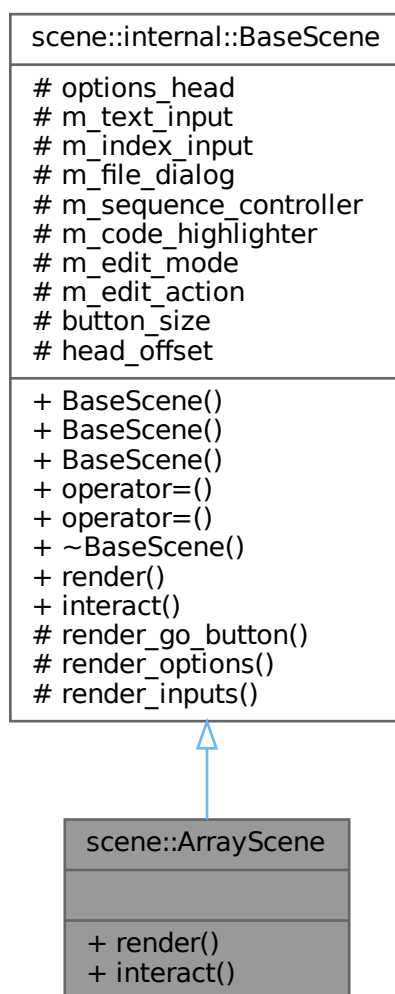
Chapter 6

Class Documentation

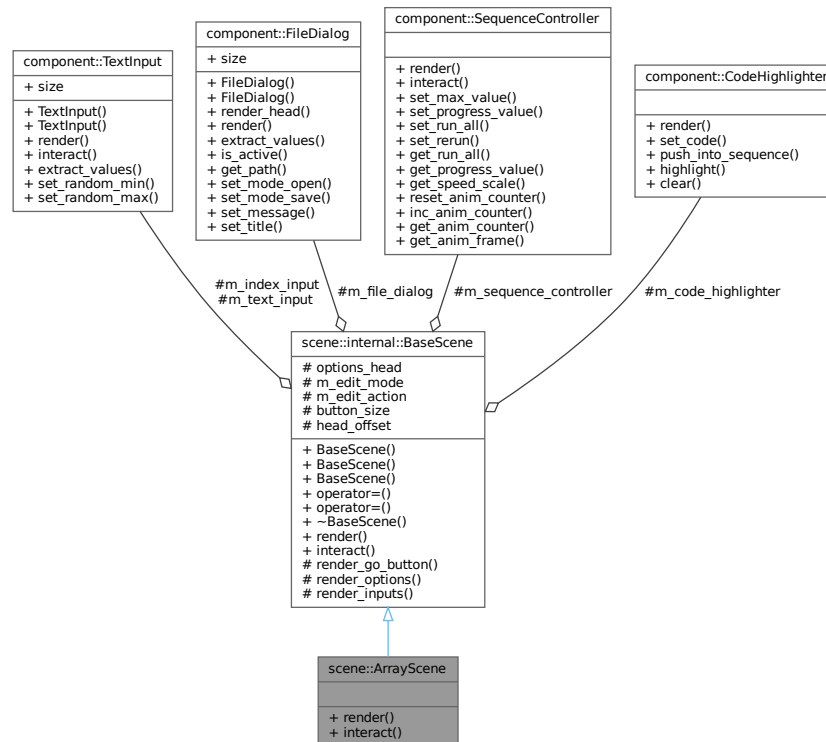
6.1 scene::ArrayScene Class Reference

```
#include <array_scene.hpp>
```

Inheritance diagram for scene::ArrayScene:



Collaboration diagram for scene::ArrayScene:



Public Member Functions

- void [render](#) () override
- void [interact](#) () override

Public Member Functions inherited from [scene::internal::BaseScene](#)

- [BaseScene](#) ()=default
- [BaseScene](#) (const [BaseScene](#) &)=delete
- [BaseScene](#) ([BaseScene](#) &&)=delete
- [BaseScene](#) & [operator=](#) (const [BaseScene](#) &)=delete
- [BaseScene](#) & [operator=](#) ([BaseScene](#) &&)=delete
- virtual [~BaseScene](#) ()=default
- virtual void [render](#) ()
- virtual void [interact](#) ()

Additional Inherited Members

Protected Member Functions inherited from [scene::internal::BaseScene](#)

- virtual bool [render_go_button](#) () const
- virtual void [render_options](#) ([SceneOptions](#) &scene_config)
- virtual void [render_inputs](#) ()

Protected Attributes inherited from [scene::internal::BaseScene](#)

- float [options_head](#) {}
- [component::TextInput](#) [m_text_input](#) {"value"}
- [component::TextInput](#) [m_index_input](#) {"index"}
- [component::FileDialog](#) [m_file_dialog](#)
- [component::SequenceController](#) [m_sequence_controller](#)
- [component::CodeHighlighter](#) [m_code_highlighter](#)
- bool [m_edit_mode](#) {}
- bool [m_edit_action](#) {}

Static Protected Attributes inherited from [scene::internal::BaseScene](#)

- static constexpr [Vector2](#) [button_size](#) {200, 50}
- static constexpr int [head_offset](#) = 20

6.1.1 Detailed Description

Definition at line 18 of file [array_scene.hpp](#).

6.1.2 Member Function Documentation

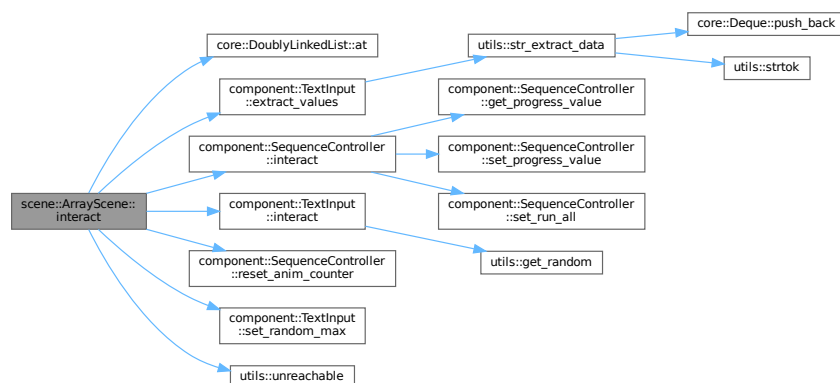
6.1.2.1 [interact\(\)](#)

```
void scene::ArrayScene::interact ( ) [override], [virtual]
```

Reimplemented from [scene::internal::BaseScene](#).

Definition at line 74 of file [array_scene.cpp](#).

Here is the call graph for this function:



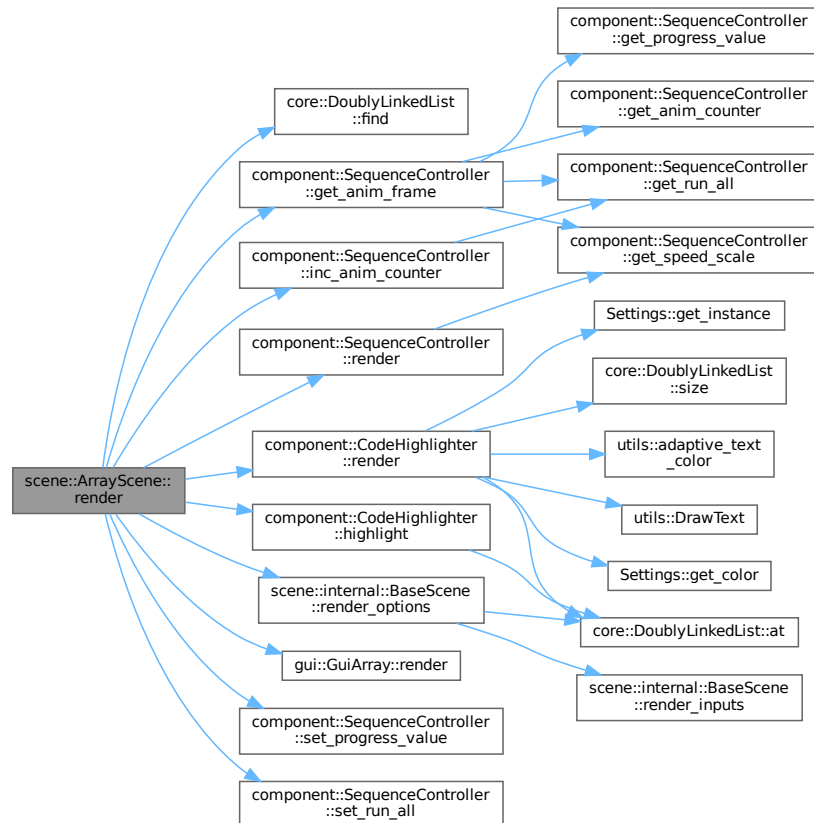
6.1.2.2 render()

```
void scene::ArrayScene::render ( ) [override], [virtual]
```

Reimplemented from [scene::internal::BaseScene](#).

Definition at line 54 of file [array_scene.cpp](#).

Here is the call graph for this function:



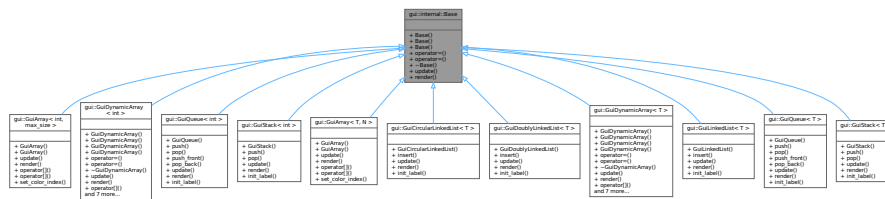
The documentation for this class was generated from the following files:

- [src/scene/array_scene.hpp](#)
- [src/scene/array_scene.cpp](#)

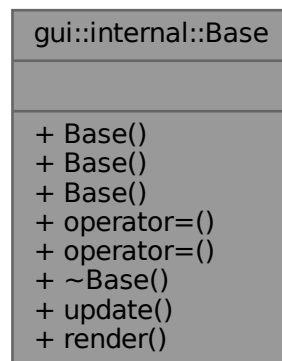
6.2 gui::internal::Base Class Reference

```
#include <base_gui.hpp>
```

Inheritance diagram for `gui::internal::Base`:



Collaboration diagram for `gui::internal::Base`:



Public Member Functions

- `Base()`=default
- `Base(const Base &)=default`
- `Base(Base &&)=default`
- `Base & operator= (const Base &)=default`
- `Base & operator= (Base &&)=default`
- virtual `~Base()`=default
- virtual void `update()`=0
- virtual void `render()`=0

6.2.1 Detailed Description

Definition at line 8 of file `base_gui.hpp`.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Base() [1/3]

```
gui::internal::Base::Base ( ) [default]
```

6.2.2.2 Base() [2/3]

```
gui::internal::Base::Base (
    const Base & ) [default]
```

6.2.2.3 Base() [3/3]

```
gui::internal::Base::Base (
    Base && ) [default]
```

6.2.2.4 ~Base()

```
virtual gui::internal::Base::~Base ( ) [virtual], [default]
```

6.2.3 Member Function Documentation

6.2.3.1 operator=() [1/2]

```
Base & gui::internal::Base::operator= (
    Base && ) [default]
```

6.2.3.2 operator=() [2/2]

```
Base & gui::internal::Base::operator= (
    const Base & ) [default]
```

6.2.3.3 render()

```
virtual void gui::internal::Base::render ( ) [pure virtual]
```

Implemented in [gui::GuiArray< T, N >](#), [gui::GuiArray< int, max_size >](#), [gui::GuiCircularLinkedList< T >](#), [gui::GuiDoublyLinkedList< T >](#), [gui::GuiDynamicArray< T >](#), [gui::GuiDynamicArray< int >](#), [gui::GuiLinkedList< T >](#), [gui::GuiQueue< T >](#), [gui::GuiQueue< int >](#), [gui::GuiStack< T >](#), and [gui::GuiStack< int >](#).

6.2.3.4 update()

```
virtual void gui::internal::Base::update ( ) [pure virtual]
```

Implemented in [gui::GuiArray< T, N >](#), [gui::GuiArray< int, max_size >](#), [gui::GuiCircularLinkedList< T >](#), [gui::GuiDoublyLinkedList< T >](#), [gui::GuiDynamicArray< T >](#), [gui::GuiDynamicArray< int >](#), [gui::GuiLinkedList< T >](#), [gui::GuiQueue< T >](#), [gui::GuiQueue< int >](#), [gui::GuiStack< T >](#), and [gui::GuiStack< int >](#).

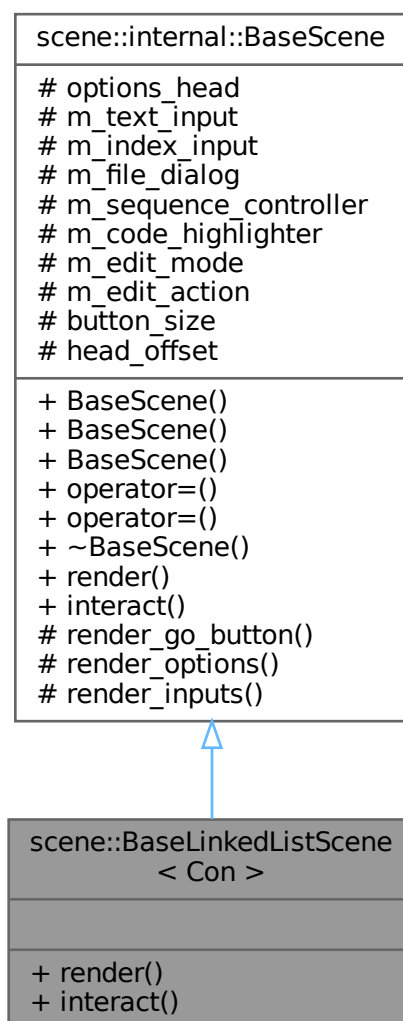
The documentation for this class was generated from the following file:

- [src/gui/base_gui.hpp](#)

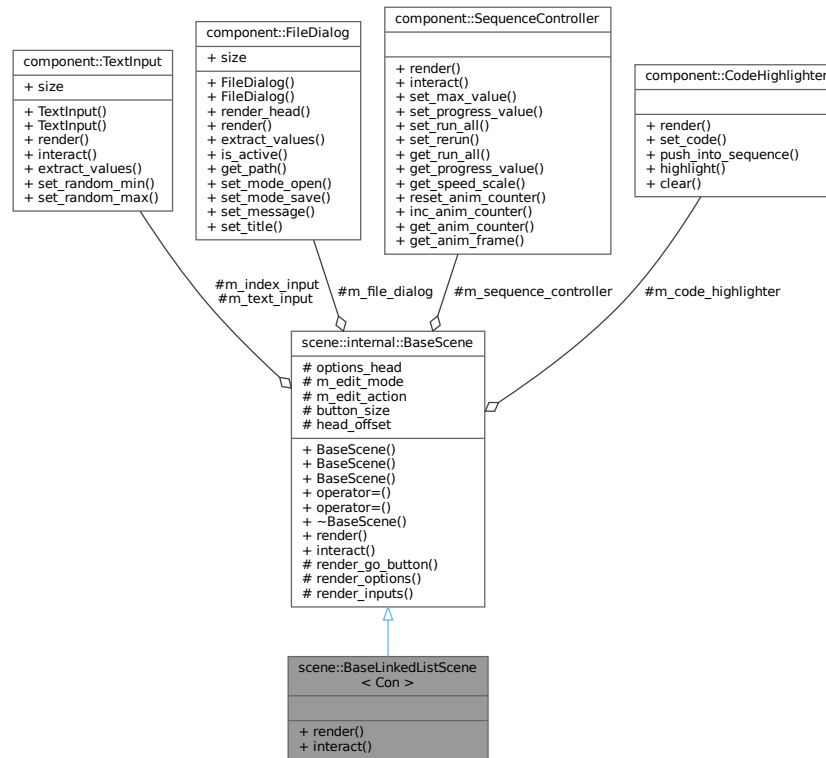
6.3 scene::BaseLinkedListScene< Con > Class Template Reference

```
#include <base_linked_list_scene.hpp>
```

Inheritance diagram for scene::BaseLinkedListScene< Con >:



Collaboration diagram for `scene::BaseLinkedListScene< Con >`:



Public Member Functions

- void `render` () override
- void `interact` () override

Public Member Functions inherited from `scene::internal::BaseScene`

- `BaseScene` ()=default
- `BaseScene` (const `BaseScene` &)=delete
- `BaseScene` (`BaseScene` &&)=delete
- `BaseScene` & `operator=` (const `BaseScene` &)=delete
- `BaseScene` & `operator=` (`BaseScene` &&)=delete
- virtual `~BaseScene` ()=default
- virtual void `render` ()
- virtual void `interact` ()

Additional Inherited Members

Protected Member Functions inherited from `scene::internal::BaseScene`

- virtual bool `render_go_button` () const
- virtual void `render_options` (`SceneOptions` &scene_config)
- virtual void `render_inputs` ()

Protected Attributes inherited from [scene::internal::BaseScene](#)

- float [options_head](#) {}
- [component::TextInput](#) [m_text_input](#) {"value"}
- [component::TextInput](#) [m_index_input](#) {"index"}
- [component::FileDialog](#) [m_file_dialog](#)
- [component::SequenceController](#) [m_sequence_controller](#)
- [component::CodeHighlighter](#) [m_code_highlighter](#)
- bool [m_edit_mode](#) {}
- bool [m_edit_action](#) {}

Static Protected Attributes inherited from [scene::internal::BaseScene](#)

- static constexpr [Vector2](#) [button_size](#) {200, 50}
- static constexpr int [head_offset](#) = 20

6.3.1 Detailed Description

```
template<typename Con>
class scene::BaseLinkedListScene< Con >
```

Definition at line 17 of file [base_linked_list_scene.hpp](#).

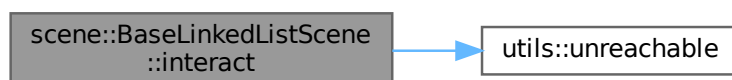
6.3.2 Member Function Documentation**6.3.2.1 [interact\(\)](#)**

```
template<typename Con >
void scene::BaseLinkedListScene< Con >::interact [override], [virtual]
```

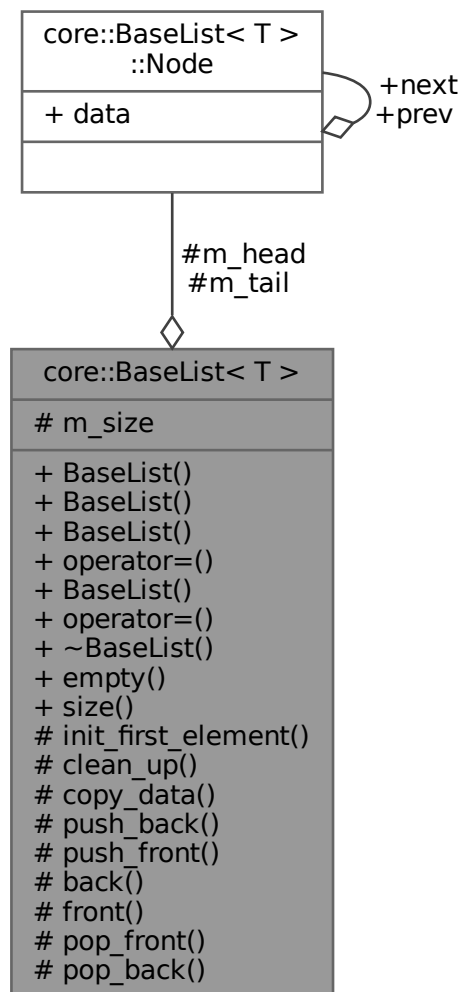
Reimplemented from [scene::internal::BaseScene](#).

Definition at line 170 of file [base_linked_list_scene.hpp](#).

Here is the call graph for this function:



Collaboration diagram for core::BaseList< T >:



Classes

- struct [Node](#)

Public Member Functions

- [BaseList](#) ()=default
- [BaseList](#) (std::initializer_list< T > init_list)
- [BaseList](#) (const [BaseList](#) &rhs)
- [BaseList](#) & [operator=](#) (const [BaseList](#) &rhs)
- [BaseList](#) ([BaseList](#) &&rhs) noexcept
- [BaseList](#) & [operator=](#) ([BaseList](#) &&rhs) noexcept
- [~BaseList](#) ()
- bool [empty](#) () const
- std::size_t [size](#) () const

Protected Types

- using [Node_ptr](#) = [Node](#) *

Protected Member Functions

- void [init_first_element](#) (const T &elem)
- void [clean_up](#) ()
- void [copy_data](#) (const [BaseList](#) &rhs)
- void [push_back](#) (const T &elem)
- void [push_front](#) (const T &elem)
- T & [back](#) () const
- T & [front](#) () const
- void [pop_front](#) ()
- void [pop_back](#) ()

Protected Attributes

- [Node_ptr](#) [m_head](#) {nullptr}
- [Node_ptr](#) [m_tail](#) {nullptr}
- std::size_t [m_size](#) {}

6.4.1 Detailed Description

```
template<typename T>  
class core::BaseList< T >
```

Definition at line 11 of file [base_list.hpp](#).

6.4.2 Member Typedef Documentation

6.4.2.1 Node_ptr

```
template<typename T >  
using core::BaseList< T >::Node_ptr = Node* [protected]
```

Definition at line 14 of file [base_list.hpp](#).

6.4.3 Constructor & Destructor Documentation

6.4.3.1 BaseList() [1/4]

```
template<typename T >
core::BaseList< T >::BaseList ( ) [default]
```

6.4.3.2 BaseList() [2/4]

```
template<typename T >
core::BaseList< T >::BaseList (
    std::initializer_list< T > init_list )
```

Definition at line 58 of file [base_list.hpp](#).

6.4.3.3 BaseList() [3/4]

```
template<typename T >
core::BaseList< T >::BaseList (
    const BaseList< T > & rhs )
```

Definition at line 53 of file [base_list.hpp](#).

6.4.3.4 BaseList() [4/4]

```
template<typename T >
core::BaseList< T >::BaseList (
    BaseList< T > && rhs ) [noexcept]
```

Definition at line 74 of file [base_list.hpp](#).

6.4.3.5 ~BaseList()

```
template<typename T >
core::BaseList< T >::~~BaseList
```

Definition at line 99 of file [base_list.hpp](#).

6.4.4 Member Function Documentation

6.4.4.1 back()

```
template<typename T >
T & core::BaseList< T >::back [protected]
```

Definition at line 166 of file [base_list.hpp](#).

6.4.4.2 clean_up()

```
template<typename T >
void core::BaseList< T >::clean_up [protected]
```

Definition at line 121 of file [base_list.hpp](#).

6.4.4.3 copy_data()

```
template<typename T >
void core::BaseList< T >::copy_data (
    const BaseList< T > & rhs ) [protected]
```

Definition at line 135 of file [base_list.hpp](#).

6.4.4.4 empty()

```
template<typename T >
bool core::BaseList< T >::empty
```

Definition at line 104 of file [base_list.hpp](#).

6.4.4.5 front()

```
template<typename T >
T & core::BaseList< T >::front [protected]
```

Definition at line 171 of file [base_list.hpp](#).

6.4.4.6 `init_first_element()`

```
template<typename T >
void core::BaseList< T >::init_first_element (
    const T & elem ) [protected]
```

Definition at line 114 of file [base_list.hpp](#).

6.4.4.7 `operator=()` [1/2]

```
template<typename T >
BaseList< T > & core::BaseList< T >::operator= (
    BaseList< T > && rhs ) [noexcept]
```

Definition at line 82 of file [base_list.hpp](#).

6.4.4.8 `operator=()` [2/2]

```
template<typename T >
BaseList< T > & core::BaseList< T >::operator= (
    const BaseList< T > & rhs )
```

Definition at line 65 of file [base_list.hpp](#).

6.4.4.9 `pop_back()`

```
template<typename T >
void core::BaseList< T >::pop_back [protected]
```

Definition at line 176 of file [base_list.hpp](#).

6.4.4.10 `pop_front()`

```
template<typename T >
void core::BaseList< T >::pop_front [protected]
```

Definition at line 189 of file [base_list.hpp](#).

6.4.4.11 push_back()

```
template<typename T >
void core::BaseList< T >::push_back (
    const T & elem ) [protected]
```

Definition at line 142 of file [base_list.hpp](#).

6.4.4.12 push_front()

```
template<typename T >
void core::BaseList< T >::push_front (
    const T & elem ) [protected]
```

Definition at line 154 of file [base_list.hpp](#).

6.4.4.13 size()

```
template<typename T >
std::size_t core::BaseList< T >::size
```

Definition at line 109 of file [base_list.hpp](#).

6.4.5 Member Data Documentation

6.4.5.1 m_head

```
template<typename T >
Node_ptr core::BaseList< T >::m_head {nullptr} [protected]
```

Definition at line 22 of file [base_list.hpp](#).

6.4.5.2 m_size

```
template<typename T >
std::size_t core::BaseList< T >::m_size {} [protected]
```

Definition at line 24 of file [base_list.hpp](#).

6.4.5.3 m_tail

```
template<typename T >
Node_ptr core::BaseList< T >::m_tail {nullptr} [protected]
```

Definition at line 23 of file [base_list.hpp](#).

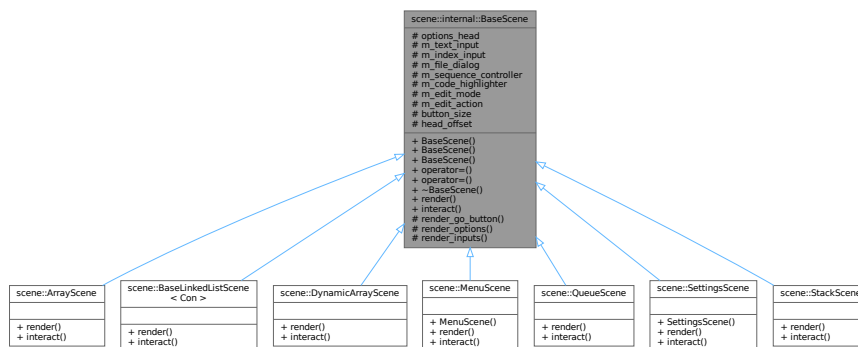
The documentation for this class was generated from the following file:

- [src/core/base_list.hpp](#)

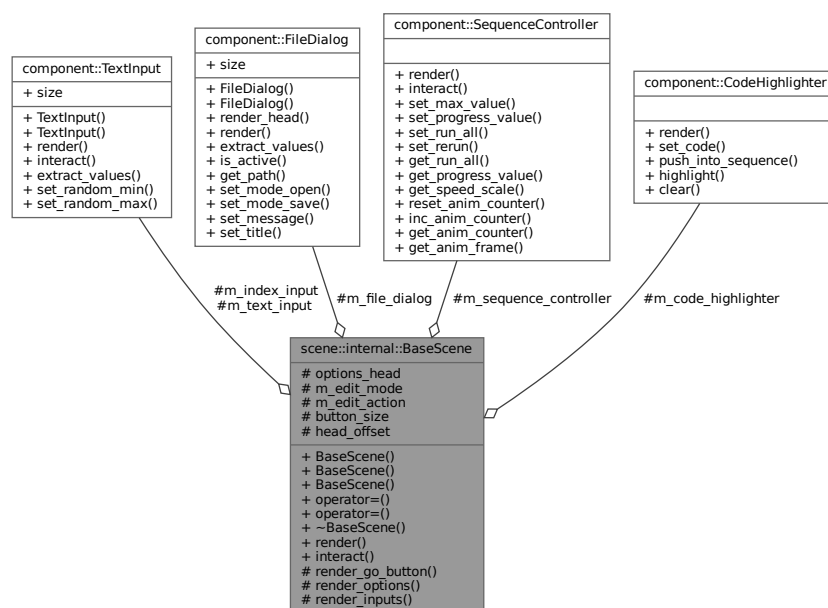
6.5 scene::internal::BaseScene Class Reference

```
#include <base_scene.hpp>
```

Inheritance diagram for scene::internal::BaseScene:



Collaboration diagram for scene::internal::BaseScene:



Public Member Functions

- [BaseScene](#) ()=default
- [BaseScene](#) (const [BaseScene](#) &)=delete
- [BaseScene](#) ([BaseScene](#) &&)=delete
- [BaseScene](#) & [operator=](#) (const [BaseScene](#) &)=delete
- [BaseScene](#) & [operator=](#) ([BaseScene](#) &&)=delete
- virtual [~BaseScene](#) ()=default
- virtual void [render](#) ()
- virtual void [interact](#) ()

Protected Member Functions

- virtual bool [render_go_button](#) () const
- virtual void [render_options](#) ([SceneOptions](#) &scene_config)
- virtual void [render_inputs](#) ()

Protected Attributes

- float [options_head](#) {}
- [component::TextInput](#) [m_text_input](#) {"value"}
- [component::TextInput](#) [m_index_input](#) {"index"}
- [component::FileDialog](#) [m_file_dialog](#)
- [component::SequenceController](#) [m_sequence_controller](#)
- [component::CodeHighlighter](#) [m_code_highlighter](#)
- bool [m_edit_mode](#) {}
- bool [m_edit_action](#) {}

Static Protected Attributes

- static constexpr [Vector2](#) [button_size](#) {200, 50}
- static constexpr int [head_offset](#) = 20

6.5.1 Detailed Description

Definition at line 13 of file [base_scene.hpp](#).

6.5.2 Constructor & Destructor Documentation

6.5.2.1 [BaseScene](#)() [1/3]

```
scene::internal::BaseScene::BaseScene ( ) [default]
```


6.5.2.2 BaseScene() [2/3]

```
scene::internal::BaseScene::BaseScene (
    const BaseScene & ) [delete]
```

6.5.2.3 BaseScene() [3/3]

```
scene::internal::BaseScene::BaseScene (
    BaseScene && ) [delete]
```

6.5.2.4 ~BaseScene()

```
virtual scene::internal::BaseScene::~~BaseScene ( ) [virtual], [default]
```

6.5.3 Member Function Documentation

6.5.3.1 interact()

```
virtual void scene::internal::BaseScene::interact ( ) [inline], [virtual]
```

Reimplemented in [scene::ArrayScene](#), [scene::BaseLinkedListScene< Con >](#), [scene::DynamicArrayScene](#), [scene::MenuScene](#), [scene::QueueScene](#), [scene::SettingsScene](#), and [scene::StackScene](#).

Definition at line 42 of file [base_scene.hpp](#).

Here is the caller graph for this function:



6.5.3.2 operator=() [1/2]

```
BaseScene & scene::internal::BaseScene::operator= (
    BaseScene && ) [delete]
```

6.5.3.3 operator=() [2/2]

```
BaseScene & scene::internal::BaseScene::operator= (
    const BaseScene & ) [delete]
```

6.5.3.4 render()

```
virtual void scene::internal::BaseScene::render ( ) [inline], [virtual]
```

Reimplemented in [scene::ArrayScene](#), [scene::BaseLinkedListScene< Con >](#), [scene::DynamicArrayScene](#), [scene::MenuScene](#), [scene::QueueScene](#), [scene::SettingsScene](#), and [scene::StackScene](#).

Definition at line 41 of file [base_scene.hpp](#).

Here is the caller graph for this function:



6.5.3.5 render_go_button()

```
bool scene::internal::BaseScene::render_go_button ( ) const [protected], [virtual]
```

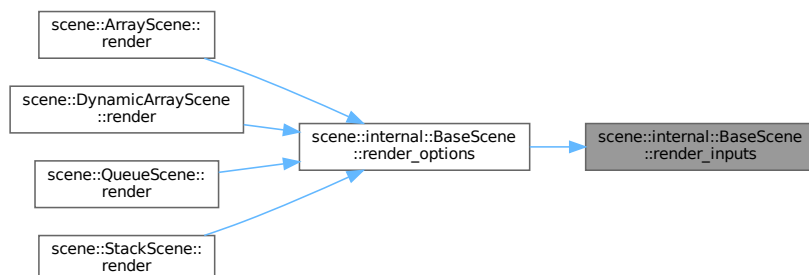
Definition at line 10 of file [base_scene.cpp](#).

6.5.3.6 render_inputs()

```
virtual void scene::internal::BaseScene::render_inputs ( ) [inline], [protected], [virtual]
```

Definition at line 21 of file [base_scene.hpp](#).

Here is the caller graph for this function:

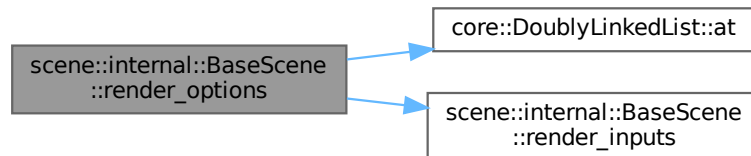


6.5.3.7 render_options()

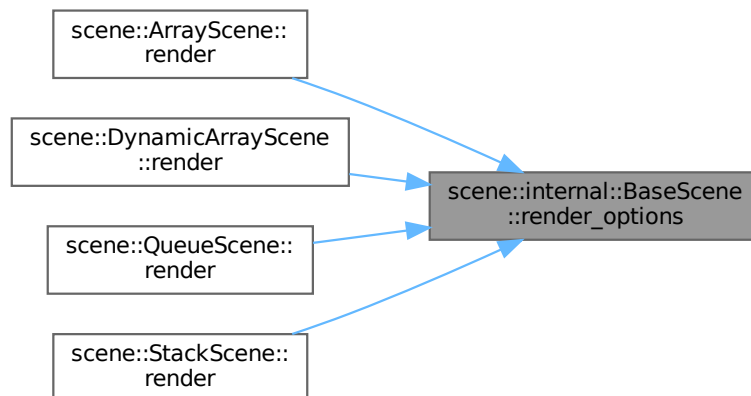
```
void scene::internal::BaseScene::render_options (
    SceneOptions & scene_config ) [protected], [virtual]
```

Definition at line 16 of file [base_scene.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.4 Member Data Documentation

6.5.4.1 button_size

```
constexpr Vector2 scene::internal::BaseScene::button_size {200, 50} [static], [constexpr],
[protected]
```

Definition at line 15 of file [base_scene.hpp](#).

6.5.4.2 head_offset

```
constexpr int scene::internal::BaseScene::head_offset = 20 [static], [constexpr], [protected]
```

Definition at line 16 of file [base_scene.hpp](#).

6.5.4.3 m_code_highlighter

```
component::CodeHighlighter scene::internal::BaseScene::m_code_highlighter [protected]
```

Definition at line 27 of file [base_scene.hpp](#).

6.5.4.4 m_edit_action

```
bool scene::internal::BaseScene::m_edit_action {} [protected]
```

Definition at line 30 of file [base_scene.hpp](#).

6.5.4.5 m_edit_mode

```
bool scene::internal::BaseScene::m_edit_mode {} [protected]
```

Definition at line 29 of file [base_scene.hpp](#).

6.5.4.6 m_file_dialog

```
component::FileDialog scene::internal::BaseScene::m_file_dialog [protected]
```

Definition at line 25 of file [base_scene.hpp](#).

6.5.4.7 m_index_input

```
component::TextInput scene::internal::BaseScene::m_index_input {"index"} [protected]
```

Definition at line 24 of file [base_scene.hpp](#).

6.5.4.8 m_sequence_controller

```
component::SequenceController scene::internal::BaseScene::m_sequence_controller [protected]
```

Definition at line 26 of file [base_scene.hpp](#).

6.5.4.9 m_text_input

```
component::TextInput scene::internal::BaseScene::m_text_input {"value"} [protected]
```

Definition at line 23 of file [base_scene.hpp](#).

6.5.4.10 options_head

```
float scene::internal::BaseScene::options_head {} [protected]
```

Definition at line 17 of file [base_scene.hpp](#).

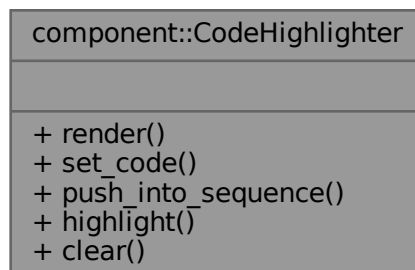
The documentation for this class was generated from the following files:

- [src/scene/base_scene.hpp](#)
- [src/scene/base_scene.cpp](#)

6.6 component::CodeHighlighter Class Reference

```
#include <code_highlighter.hpp>
```

Collaboration diagram for component::CodeHighlighter:



Public Member Functions

- void [render](#) ()
- void [set_code](#) ([core::DoublyLinkedList](#)< const char * > &&src_code)
- void [push_into_sequence](#) (int line_number)
- void [highlight](#) (int frame_idx)
- void [clear](#) ()

6.6.1 Detailed Description

Definition at line 10 of file [code_highlighter.hpp](#).

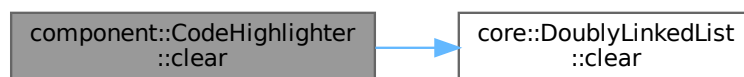
6.6.2 Member Function Documentation

6.6.2.1 [clear\(\)](#)

```
void component::CodeHighlighter::clear ( )
```

Definition at line 38 of file [code_highlighter.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.2.2 highlight()

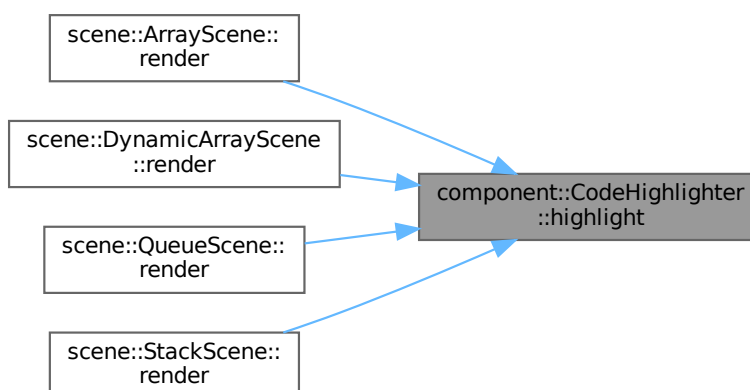
```
void component::CodeHighlighter::highlight (
    int frame_idx )
```

Definition at line 34 of file [code_highlighter.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

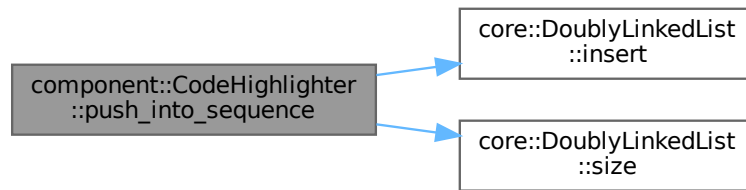


6.6.2.3 push_into_sequence()

```
void component::CodeHighlighter::push_into_sequence (
    int line_number )
```

Definition at line 30 of file [code_highlighter.cpp](#).

Here is the call graph for this function:

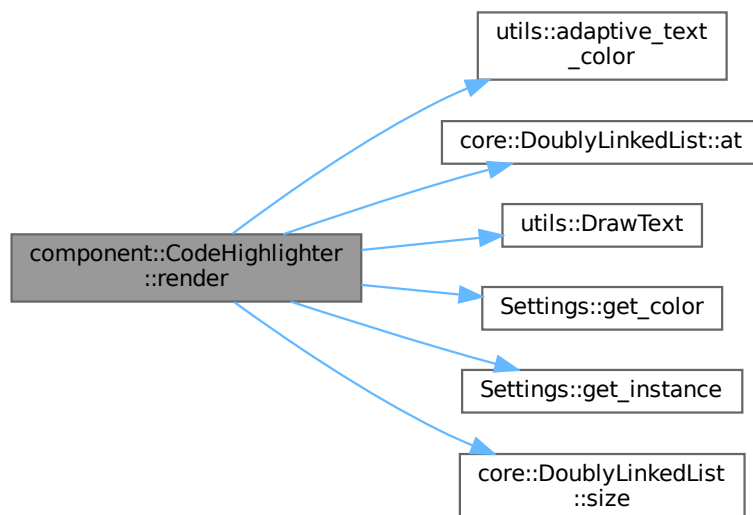


6.6.2.4 render()

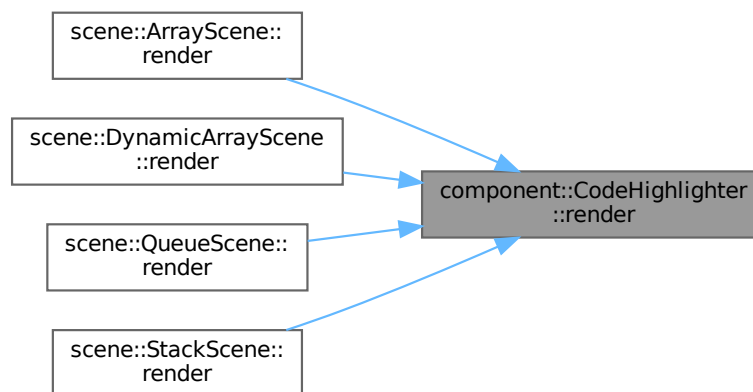
```
void component::CodeHighlighter::render ( )
```

Definition at line 9 of file [code_highlighter.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

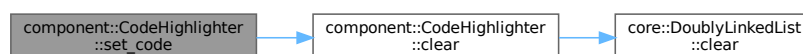


6.6.2.5 set_code()

```
void component::CodeHighlighter::set_code (
    core::DoublyLinkedList< const char * > && src_code )
```

Definition at line 25 of file [code_highlighter.cpp](#).

Here is the call graph for this function:



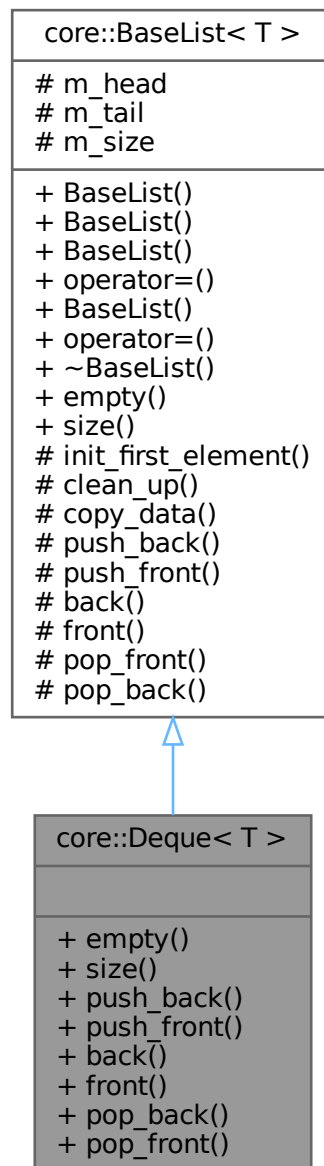
The documentation for this class was generated from the following files:

- [src/component/code_highlighter.hpp](#)
- [src/component/code_highlighter.cpp](#)

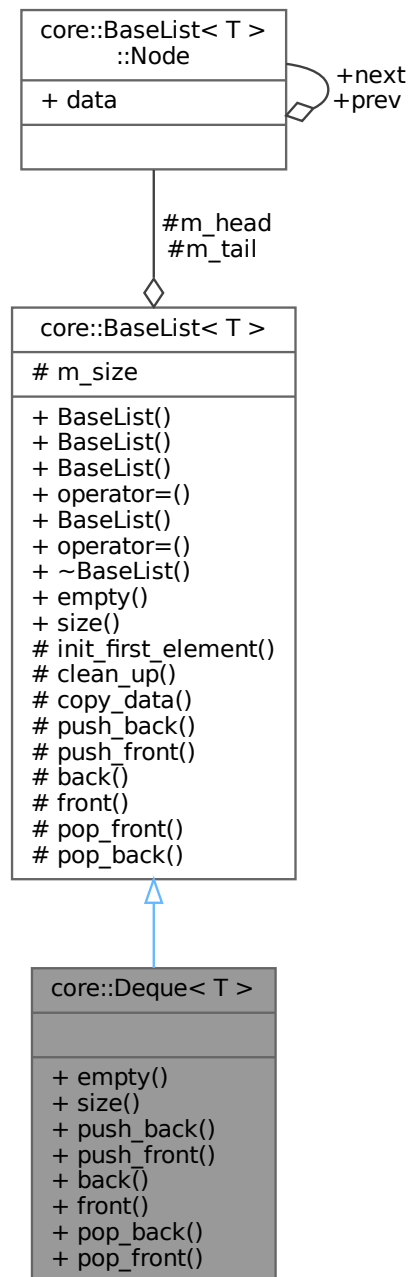
6.7 core::Deque< T > Class Template Reference

```
#include <deque.hpp>
```

Inheritance diagram for `core::Deque< T >`:



Collaboration diagram for core::Deque< T >:



Public Member Functions

- `bool empty () const`
- `std::size_t size () const`
- `void push_back (const T &elem)`
- `void push_front (const T &elem)`
- `T & back () const`

- T & [front](#) () const
- void [pop_back](#) ()
- void [pop_front](#) ()

Public Member Functions inherited from [core::BaseList< T >](#)

- [BaseList](#) ()=default
- [BaseList](#) (std::initializer_list< T > init_list)
- [BaseList](#) (const [BaseList](#) &rhs)
- [BaseList](#) & [operator=](#) (const [BaseList](#) &rhs)
- [BaseList](#) ([BaseList](#) &&rhs) noexcept
- [BaseList](#) & [operator=](#) ([BaseList](#) &&rhs) noexcept
- [~BaseList](#) ()
- bool [empty](#) () const
- std::size_t [size](#) () const

Additional Inherited Members

Protected Types inherited from [core::BaseList< T >](#)

- using [Node_ptr](#) = [Node](#) *

Protected Member Functions inherited from [core::BaseList< T >](#)

- void [init_first_element](#) (const T &elem)
- void [clean_up](#) ()
- void [copy_data](#) (const [BaseList](#) &rhs)
- void [push_back](#) (const T &elem)
- void [push_front](#) (const T &elem)
- T & [back](#) () const
- T & [front](#) () const
- void [pop_front](#) ()
- void [pop_back](#) ()

Protected Attributes inherited from [core::BaseList< T >](#)

- [Node_ptr](#) [m_head](#) {nullptr}
- [Node_ptr](#) [m_tail](#) {nullptr}
- std::size_t [m_size](#) {}

6.7.1 Detailed Description

```
template<typename T>
class core::Deque< T >
```

Definition at line 9 of file [deque.hpp](#).

6.7.2 Member Function Documentation

6.7.2.1 back()

```
template<typename T >  
T & core::BaseList< T >::back
```

Definition at line 33 of file [base_list.hpp](#).

Here is the caller graph for this function:



6.7.2.2 empty()

```
template<typename T >  
bool core::BaseList< T >::empty
```

Definition at line 48 of file [base_list.hpp](#).

Here is the caller graph for this function:

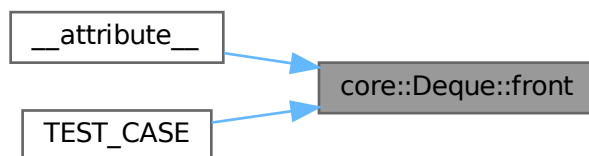


6.7.2.3 front()

```
template<typename T >  
T & core::BaseList< T >::front
```

Definition at line 34 of file [base_list.hpp](#).

Here is the caller graph for this function:



6.7.2.4 pop_back()

```
template<typename T >  
void core::BaseList< T >::pop_back
```

Definition at line 37 of file [base_list.hpp](#).

Here is the caller graph for this function:

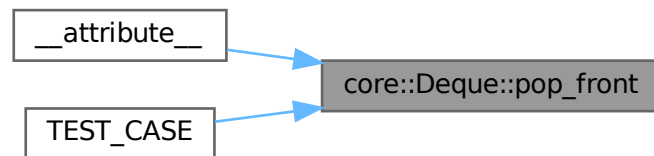


6.7.2.5 pop_front()

```
template<typename T >  
void core::BaseList< T >::pop_front
```

Definition at line 36 of file [base_list.hpp](#).

Here is the caller graph for this function:

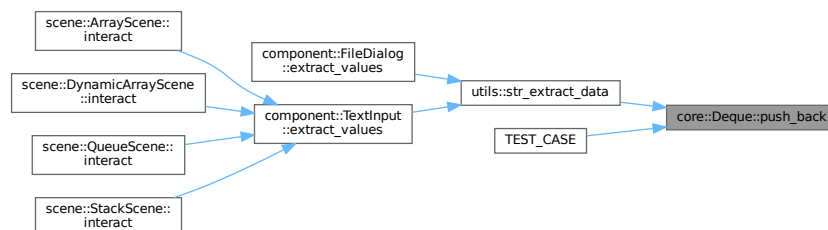


6.7.2.6 push_back()

```
template<typename T >  
void core::BaseList< T >::push_back (  
    const T & elem )
```

Definition at line 30 of file [base_list.hpp](#).

Here is the caller graph for this function:



6.7.2.7 push_front()

```
template<typename T >
void core::BaseList< T >::push_front (
    const T & elem )
```

Definition at line 31 of file [base_list.hpp](#).

Here is the caller graph for this function:



6.7.2.8 size()

```
template<typename T >
std::size_t core::BaseList< T >::size
```

Definition at line 49 of file [base_list.hpp](#).

Here is the caller graph for this function:



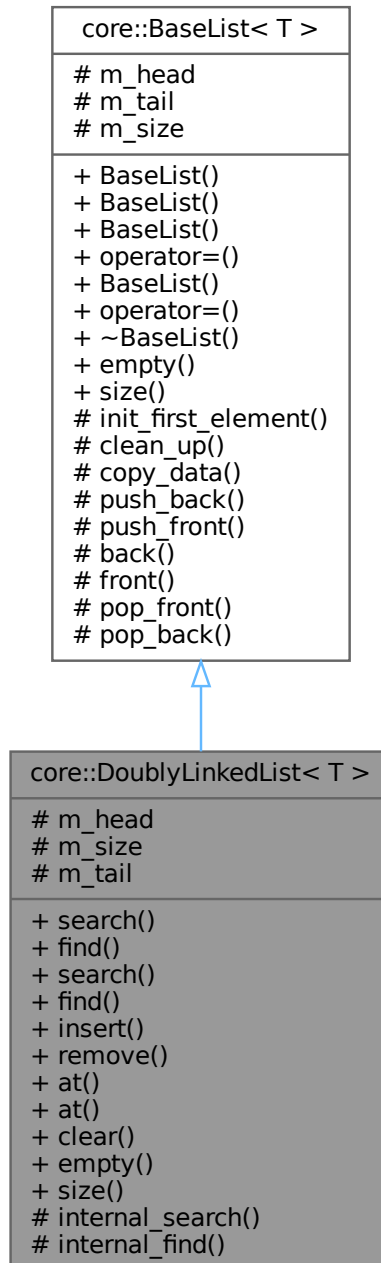
The documentation for this class was generated from the following file:

- [src/core/deque.hpp](#)

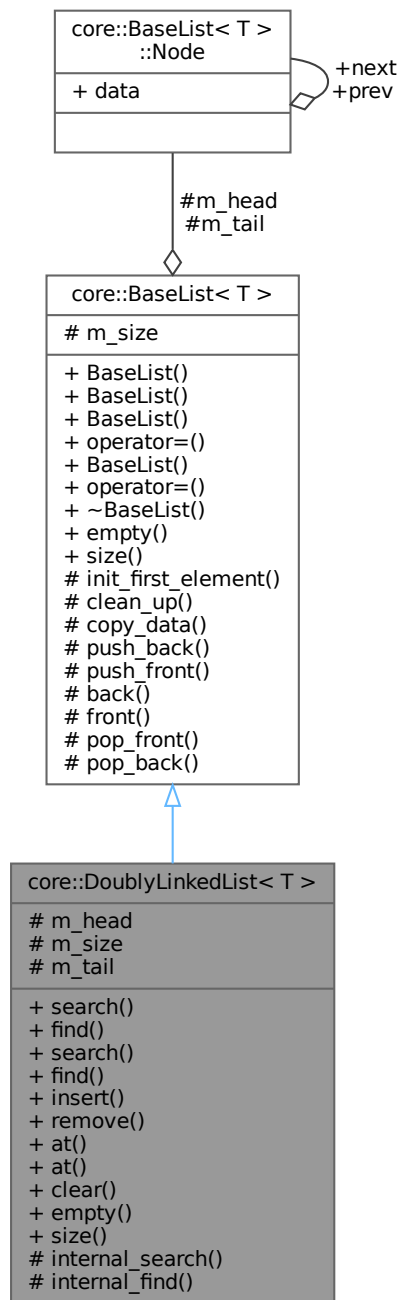
6.8 core::DoublyLinkedList< T > Class Template Reference

```
#include <doubly_linked_list.hpp>
```

Inheritance diagram for core::DoublyLinkedList< T >:



Collaboration diagram for `core::DoublyLinkedList< T >`:



Public Member Functions

- [Node_ptr search](#) (const T &elem)
- [Node_ptr find](#) (std::size_t index)
- [cNode_ptr search](#) (const T &elem) const
- [cNode_ptr find](#) (std::size_t index) const
- [Node_ptr insert](#) (std::size_t index, const T &elem)

- [Node_ptr remove](#) (std::size_t index)
- T & [at](#) (std::size_t index)
- T [at](#) (std::size_t index) const
- void [clear](#) ()
- bool [empty](#) () const
- std::size_t [size](#) () const

Public Member Functions inherited from [core::BaseList< T >](#)

- [BaseList](#) ()=default
- [BaseList](#) (std::initializer_list< T > init_list)
- [BaseList](#) (const [BaseList](#) &rhs)
- [BaseList](#) & [operator=](#) (const [BaseList](#) &rhs)
- [BaseList](#) ([BaseList](#) &&rhs) noexcept
- [BaseList](#) & [operator=](#) ([BaseList](#) &&rhs) noexcept
- [~BaseList](#) ()
- bool [empty](#) () const
- std::size_t [size](#) () const

Protected Types

- using [Base](#) = [BaseList](#)< T >
- using [Node](#) = typename [Base::Node](#)
- using [Node_ptr](#) = [Node](#) *
- using [cNode_ptr](#) = const [Node](#) *

Protected Types inherited from [core::BaseList< T >](#)

- using [Node_ptr](#) = [Node](#) *

Protected Member Functions

- [Node_ptr internal_search](#) (const T &elem)
- [Node_ptr internal_find](#) (std::size_t index)

Protected Member Functions inherited from [core::BaseList< T >](#)

- void [init_first_element](#) (const T &elem)
- void [clean_up](#) ()
- void [copy_data](#) (const [BaseList](#) &rhs)
- void [push_back](#) (const T &elem)
- void [push_front](#) (const T &elem)
- T & [back](#) () const
- T & [front](#) () const
- void [pop_front](#) ()
- void [pop_back](#) ()

Protected Attributes

- [Node_ptr m_head](#)
- [std::size_t m_size](#)
- [Node_ptr m_tail](#)

Protected Attributes inherited from [core::BaseList< T >](#)

- [Node_ptr m_head](#) {nullptr}
- [Node_ptr m_tail](#) {nullptr}
- [std::size_t m_size](#) {}

6.8.1 Detailed Description

```
template<typename T>
class core::DoublyLinkedList< T >
```

Definition at line 11 of file [doubly_linked_list.hpp](#).

6.8.2 Member Typedef Documentation

6.8.2.1 Base

```
template<typename T >
using core::DoublyLinkedList< T >::Base = BaseList<T> [protected]
```

Definition at line 13 of file [doubly_linked_list.hpp](#).

6.8.2.2 cNode_ptr

```
template<typename T >
using core::DoublyLinkedList< T >::cNode\_ptr = const Node\* [protected]
```

Definition at line 16 of file [doubly_linked_list.hpp](#).

6.8.2.3 Node

```
template<typename T >
using core::DoublyLinkedList< T >::Node = typename Base::Node [protected]
```

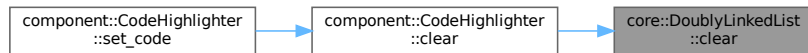
Definition at line 14 of file [doubly_linked_list.hpp](#).

6.8.3.3 clear()

```
template<typename T >
void core::DoublyLinkedList< T >::clear
```

Definition at line 163 of file [doubly_linked_list.hpp](#).

Here is the caller graph for this function:



6.8.3.4 empty()

```
template<typename T >
bool core::BaseList< T >::empty
```

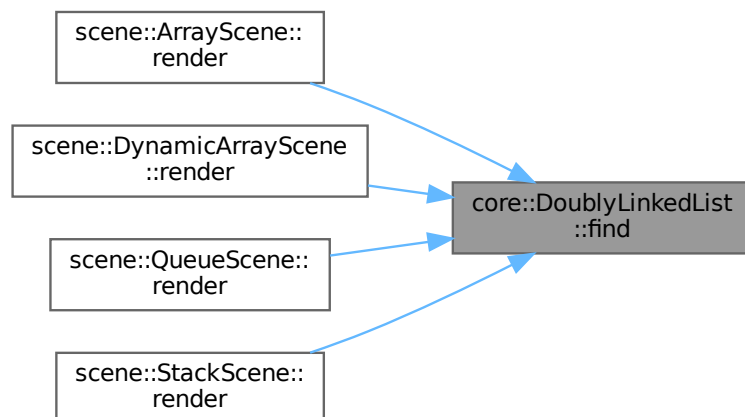
Definition at line 48 of file [base_list.hpp](#).

6.8.3.5 find() [1/2]

```
template<typename T >
DoublyLinkedList< T >::Node_ptr core::DoublyLinkedList< T >::find (
    std::size_t index )
```

Definition at line 83 of file [doubly_linked_list.hpp](#).

Here is the caller graph for this function:



6.8.3.6 find() [2/2]

```
template<typename T >
DoublyLinkedList< T >::cNode_ptr core::DoublyLinkedList< T >::find (
    std::size_t index ) const
```

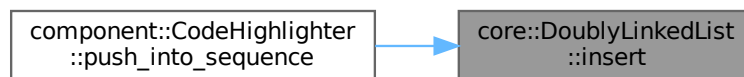
Definition at line 95 of file [doubly_linked_list.hpp](#).

6.8.3.7 insert()

```
template<typename T >
DoublyLinkedList< T >::Node_ptr core::DoublyLinkedList< T >::insert (
    std::size_t index,
    const T & elem )
```

Definition at line 101 of file [doubly_linked_list.hpp](#).

Here is the caller graph for this function:

**6.8.3.8 internal_find()**

```
template<typename T >
DoublyLinkedList< T >::Node_ptr core::DoublyLinkedList< T >::internal_find (
    std::size_t index ) [protected]
```

Definition at line 63 of file [doubly_linked_list.hpp](#).

6.8.3.9 internal_search()

```
template<typename T >
DoublyLinkedList< T >::Node_ptr core::DoublyLinkedList< T >::internal_search (
    const T & elem ) [protected]
```

Definition at line 47 of file [doubly_linked_list.hpp](#).

6.8.3.10 remove()

```
template<typename T >
DoublyLinkedList< T >::Node_ptr core::DoublyLinkedList< T >::remove (
    std::size_t index )
```

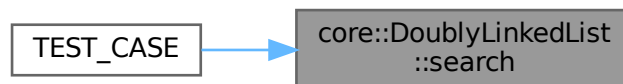
Definition at line 124 of file [doubly_linked_list.hpp](#).

6.8.3.11 search() [1/2]

```
template<typename T >
DoublyLinkedList< T >::Node_ptr core::DoublyLinkedList< T >::search (
    const T & elem )
```

Definition at line 77 of file [doubly_linked_list.hpp](#).

Here is the caller graph for this function:



6.8.3.12 search() [2/2]

```
template<typename T >
DoublyLinkedList< T >::cNode_ptr core::DoublyLinkedList< T >::search (
    const T & elem ) const
```

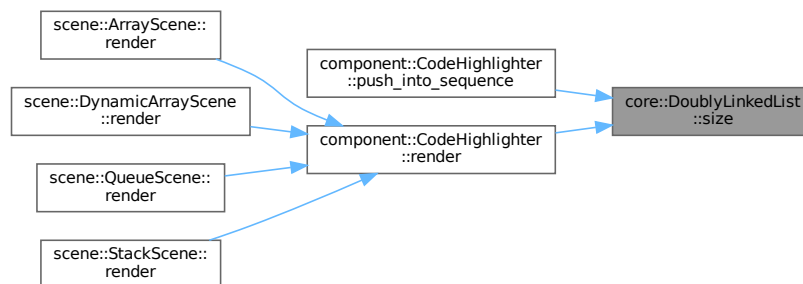
Definition at line 89 of file [doubly_linked_list.hpp](#).

6.8.3.13 size()

```
template<typename T >
std::size_t core::BaseList< T >::size
```

Definition at line 49 of file [base_list.hpp](#).

Here is the caller graph for this function:



6.8.4 Member Data Documentation

6.8.4.1 m_head

```
template<typename T >
Node_ptr core::BaseList< T >::m_head [protected]
```

Definition at line 22 of file [base_list.hpp](#).

6.8.4.2 m_size

```
template<typename T >
std::size_t core::BaseList< T >::m_size [protected]
```

Definition at line 24 of file [base_list.hpp](#).

6.8.4.3 m_tail

```
template<typename T >
Node_ptr core::BaseList< T >::m_tail [protected]
```

Definition at line 23 of file [base_list.hpp](#).

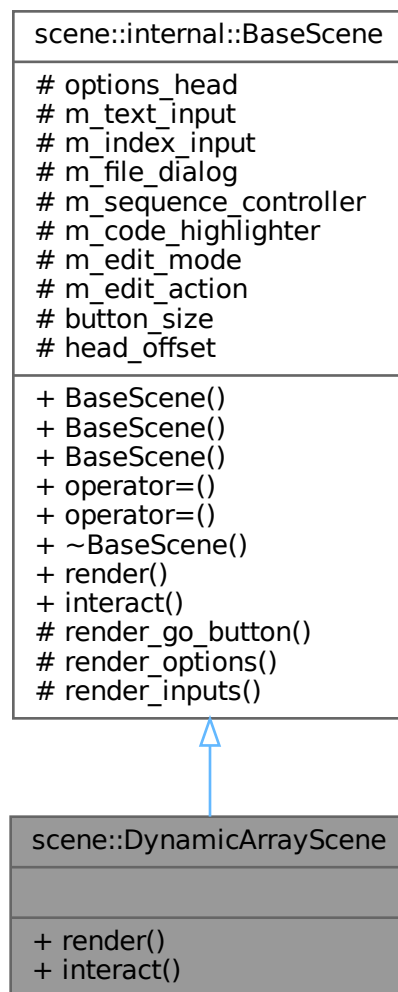
The documentation for this class was generated from the following file:

- [src/core/doubly_linked_list.hpp](#)

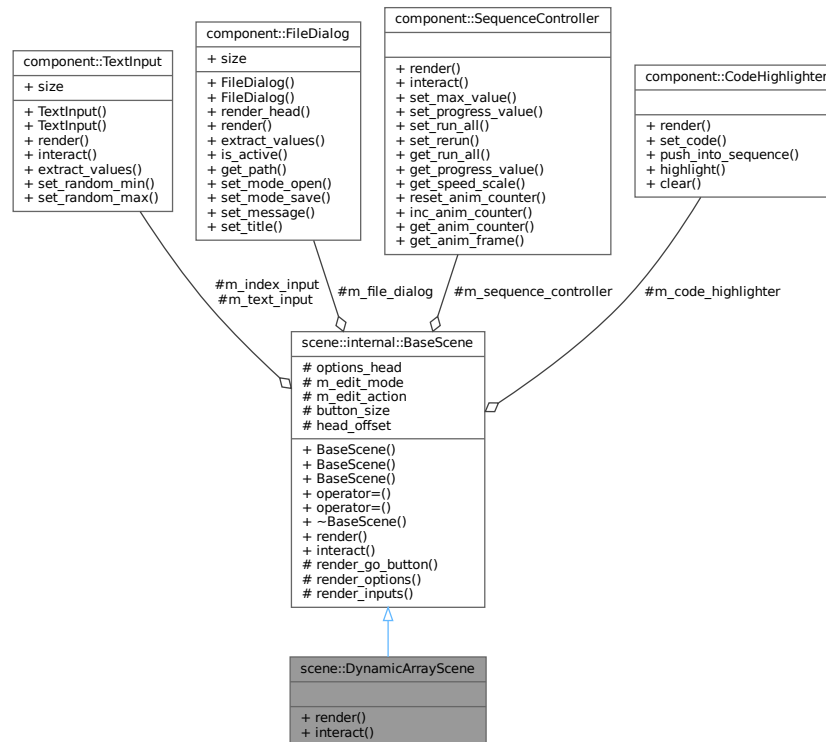
6.9 scene::DynamicArrayScene Class Reference

```
#include <dynamic_array_scene.hpp>
```

Inheritance diagram for scene::DynamicArrayScene:



Collaboration diagram for scene::DynamicArrayScene:



Public Member Functions

- void [render](#) () override
- void [interact](#) () override

Public Member Functions inherited from [scene::internal::BaseScene](#)

- [BaseScene](#) ()=default
- [BaseScene](#) (const [BaseScene](#) &)=delete
- [BaseScene](#) ([BaseScene](#) &&)=delete
- [BaseScene](#) & [operator=](#) (const [BaseScene](#) &)=delete
- [BaseScene](#) & [operator=](#) ([BaseScene](#) &&)=delete
- virtual [~BaseScene](#) ()=default
- virtual void [render](#) ()
- virtual void [interact](#) ()

Additional Inherited Members

Protected Member Functions inherited from [scene::internal::BaseScene](#)

- virtual bool [render_go_button](#) () const
- virtual void [render_options](#) ([SceneOptions](#) &scene_config)
- virtual void [render_inputs](#) ()

Protected Attributes inherited from [scene::internal::BaseScene](#)

- float [options_head](#) {}
- [component::TextInput](#) [m_text_input](#) {"value"}
- [component::TextInput](#) [m_index_input](#) {"index"}
- [component::FileDialog](#) [m_file_dialog](#)
- [component::SequenceController](#) [m_sequence_controller](#)
- [component::CodeHighlighter](#) [m_code_highlighter](#)
- bool [m_edit_mode](#) {}
- bool [m_edit_action](#) {}

Static Protected Attributes inherited from [scene::internal::BaseScene](#)

- static constexpr [Vector2](#) [button_size](#) {200, 50}
- static constexpr int [head_offset](#) = 20

6.9.1 Detailed Description

Definition at line 18 of file [dynamic_array_scene.hpp](#).

6.9.2 Member Function Documentation

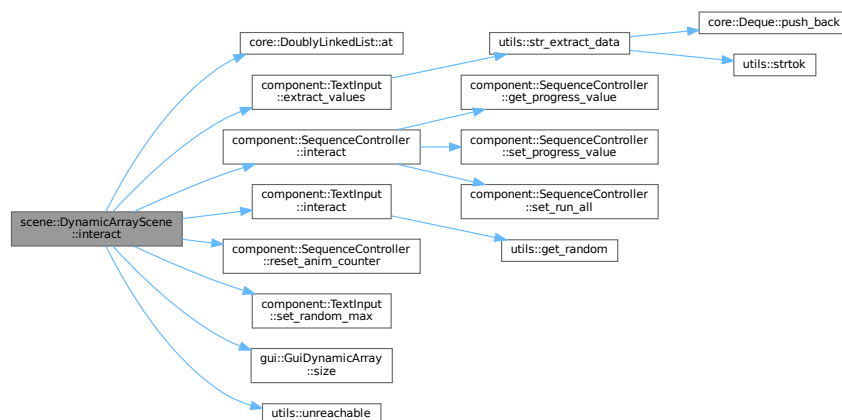
6.9.2.1 [interact\(\)](#)

```
void scene::DynamicArrayScene::interact ( ) [override], [virtual]
```

Reimplemented from [scene::internal::BaseScene](#).

Definition at line 78 of file [dynamic_array_scene.cpp](#).

Here is the call graph for this function:



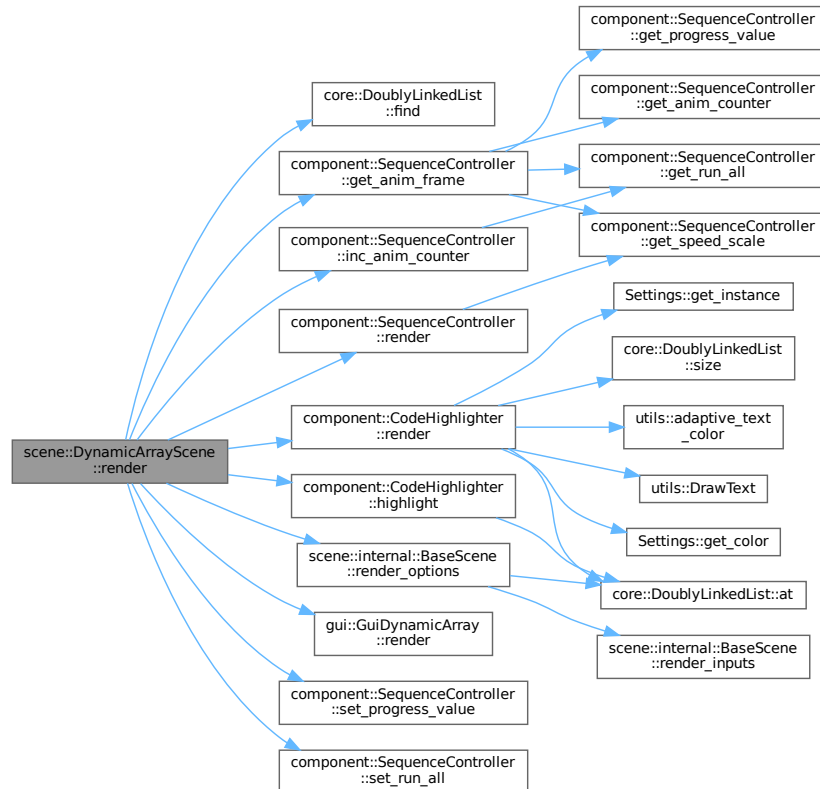
6.9.2.2 render()

```
void scene::DynamicArrayScene::render ( ) [override], [virtual]
```

Reimplemented from [scene::internal::BaseScene](#).

Definition at line 58 of file [dynamic_array_scene.cpp](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- [src/scene/dynamic_array_scene.hpp](#)
- [src/scene/dynamic_array_scene.cpp](#)

6.10 component::FileDialog Class Reference

```
#include <file_dialog.hpp>
```

Collaboration diagram for component::FileDialog:

component::FileDialog
+ size
+ FileDialog() + FileDialog() + render_head() + render() + extract_values() + is_active() + get_path() + set_mode_open() + set_mode_save() + set_message() + set_title()

Public Member Functions

- [FileDialog](#) ()
- [FileDialog](#) (int mode, const char *title, const char *message)
- int [render_head](#) (float &options_head, float head_offset)
- int [render](#) (float x, float y)
- [core::Deque](#)< int > [extract_values](#) ()
- bool [is_active](#) () const
- std::string [get_path](#) ()
- void [set_mode_open](#) ()
- void [set_mode_save](#) ()
- void [set_message](#) (const char *message)
- void [set_title](#) (const char *title)

Static Public Attributes

- static constexpr Vector2 [size](#) {200, 50}

6.10.1 Detailed Description

Definition at line 13 of file [file_dialog.hpp](#).

6.10.2 Constructor & Destructor Documentation

6.10.2.1 FileDialog() [1/2]

```
component::FileDialog::FileDialog ( )
```

Definition at line 16 of file [file_dialog.cpp](#).

6.10.2.2 FileDialog() [2/2]

```
component::FileDialog::FileDialog (
    int mode,
    const char * title,
    const char * message )
```

Definition at line 13 of file [file_dialog.cpp](#).

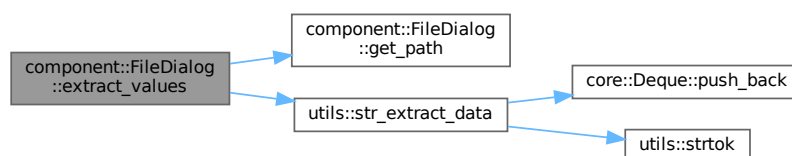
6.10.3 Member Function Documentation

6.10.3.1 extract_values()

```
core::Deque< int > component::FileDialog::extract_values ( )
```

Definition at line 49 of file [file_dialog.cpp](#).

Here is the call graph for this function:

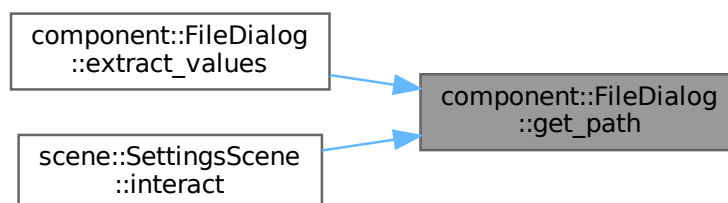


6.10.3.2 get_path()

```
std::string component::FileDialog::get_path ( )
```

Definition at line 66 of file [file_dialog.cpp](#).

Here is the caller graph for this function:



6.10.3.3 is_active()

```
bool component::FileDialog::is_active ( ) const
```

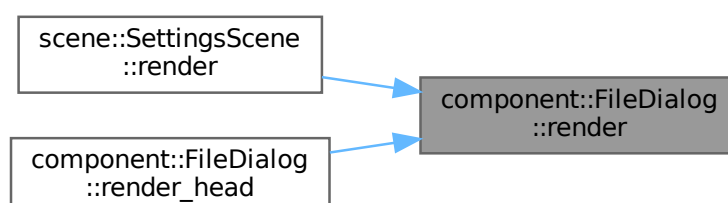
Definition at line 57 of file [file_dialog.cpp](#).

6.10.3.4 render()

```
int component::FileDialog::render (
    float x,
    float y )
```

Definition at line 18 of file [file_dialog.cpp](#).

Here is the caller graph for this function:



6.10.3.5 render_head()

```
int component::FileDialog::render_head (
    float & options_head,
    float head_offset )
```

Definition at line 43 of file [file_dialog.cpp](#).

Here is the call graph for this function:



6.10.3.6 set_message()

```
void component::FileDialog::set_message (
    const char * message )
```

Definition at line 63 of file [file_dialog.cpp](#).

6.10.3.7 set_mode_open()

```
void component::FileDialog::set_mode_open ( )
```

Definition at line 59 of file [file_dialog.cpp](#).

6.10.3.8 set_mode_save()

```
void component::FileDialog::set_mode_save ( )
```

Definition at line 61 of file [file_dialog.cpp](#).

6.10.3.9 set_title()

```
void component::FileDialog::set_title (
    const char * title )
```

Definition at line 65 of file [file_dialog.cpp](#).

6.10.4 Member Data Documentation

6.10.4.1 size

```
constexpr Vector2 component::FileDialog::size {200, 50} [static], [constexpr]
```

Definition at line 25 of file [file_dialog.hpp](#).

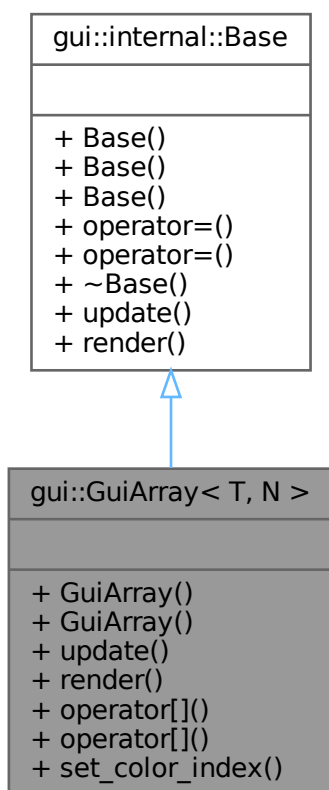
The documentation for this class was generated from the following files:

- [src/component/file_dialog.hpp](#)
- [src/component/file_dialog.cpp](#)

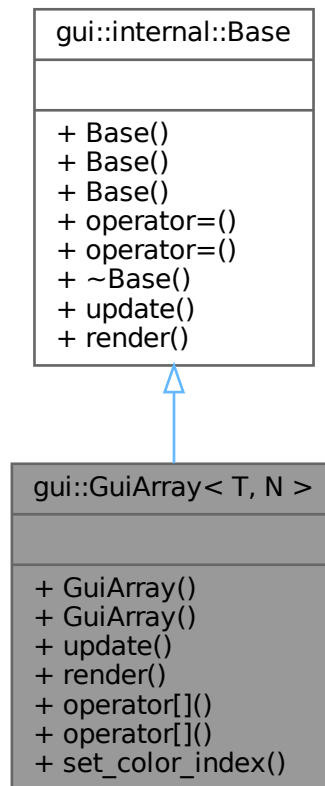
6.11 gui::GuiArray< T, N > Class Template Reference

```
#include <array_gui.hpp>
```

Inheritance diagram for gui::GuiArray< T, N >:



Collaboration diagram for `gui::GuiArray< T, N >`:



Public Member Functions

- [GuiArray](#) ()
- [GuiArray](#) (std::array< [GuiElement](#)< T >, N > &&init_list)
- void [update](#) () override
- void [render](#) () override
- T & [operator\[\]](#) (std::size_t idx)
- T [operator\[\]](#) (std::size_t idx) const
- void [set_color_index](#) (std::size_t idx, int color_index)

Public Member Functions inherited from [gui::internal::Base](#)

- [Base](#) ()=default
- [Base](#) (const [Base](#) &)=default
- [Base](#) ([Base](#) &&)=default
- [Base](#) & [operator=](#) (const [Base](#) &)=default
- [Base](#) & [operator=](#) ([Base](#) &&)=default
- virtual [~Base](#) ()=default
- virtual void [update](#) ()=0
- virtual void [render](#) ()=0

6.11.1 Detailed Description

```
template<typename T, std::size_t N>
class gui::GuiArray< T, N >
```

Definition at line 16 of file [array_gui.hpp](#).

6.11.2 Constructor & Destructor Documentation

6.11.2.1 GuiArray() [1/2]

```
template<typename T , std::size_t N>
gui::GuiArray< T, N >::GuiArray
```

Definition at line 39 of file [array_gui.hpp](#).

Here is the call graph for this function:



6.11.2.2 GuiArray() [2/2]

```
template<typename T , std::size_t N>
gui::GuiArray< T, N >::GuiArray (
    std::array< GuiElement< T >, N > && init_list )
```

Definition at line 47 of file [array_gui.hpp](#).

6.11.3 Member Function Documentation

6.11.3.1 operator[]() [1/2]

```
template<typename T , std::size_t N>
T & gui::GuiArray< T, N >::operator[] (
    std::size_t idx )
```

Definition at line 73 of file [array_gui.hpp](#).

6.11.3.2 operator[]() [2/2]

```
template<typename T , std::size_t N>
T gui::GuiArray< T, N >::operator[] (
    std::size_t idx ) const
```

Definition at line 78 of file [array_gui.hpp](#).

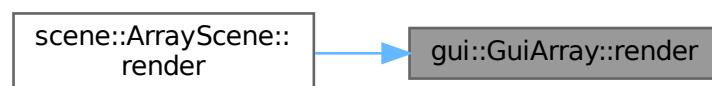
6.11.3.3 render()

```
template<typename T , std::size_t N>
void gui::GuiArray< T, N >::render [override], [virtual]
```

Implements [gui::internal::Base](#).

Definition at line 54 of file [array_gui.hpp](#).

Here is the caller graph for this function:

**6.11.3.4 set_color_index()**

```
template<typename T , std::size_t N>
void gui::GuiArray< T, N >::set_color_index (
    std::size_t idx,
    int color_index )
```

Definition at line 83 of file [array_gui.hpp](#).

6.11.3.5 update()

```
template<typename T , std::size_t N>  
void gui::GuiArray< T, N >::update [override], [virtual]
```

Implements [gui::internal::Base](#).

Definition at line 63 of file [array_gui.hpp](#).

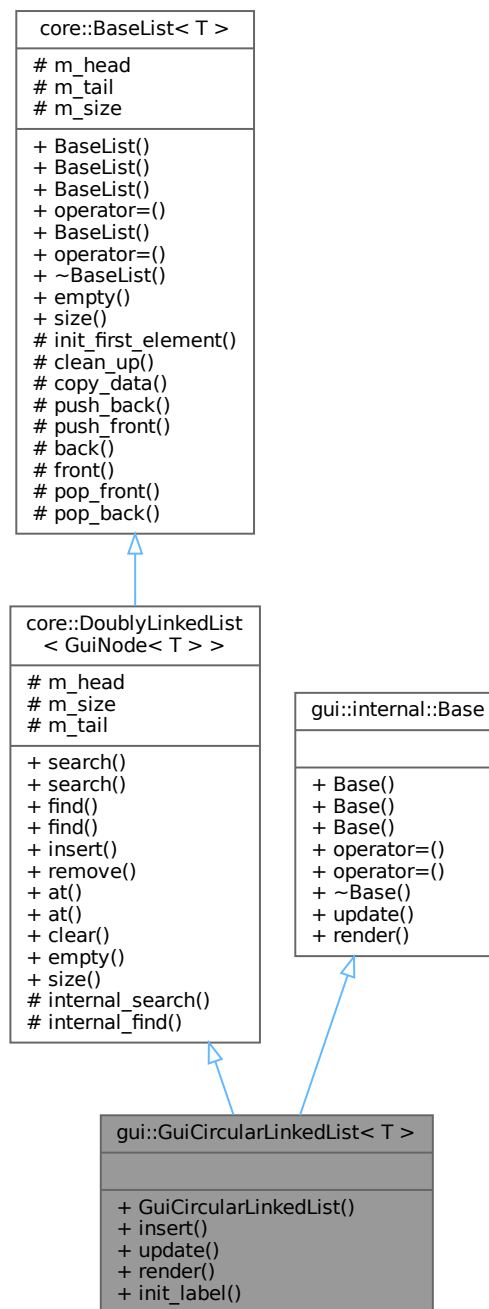
The documentation for this class was generated from the following file:

- [src/gui/array_gui.hpp](#)

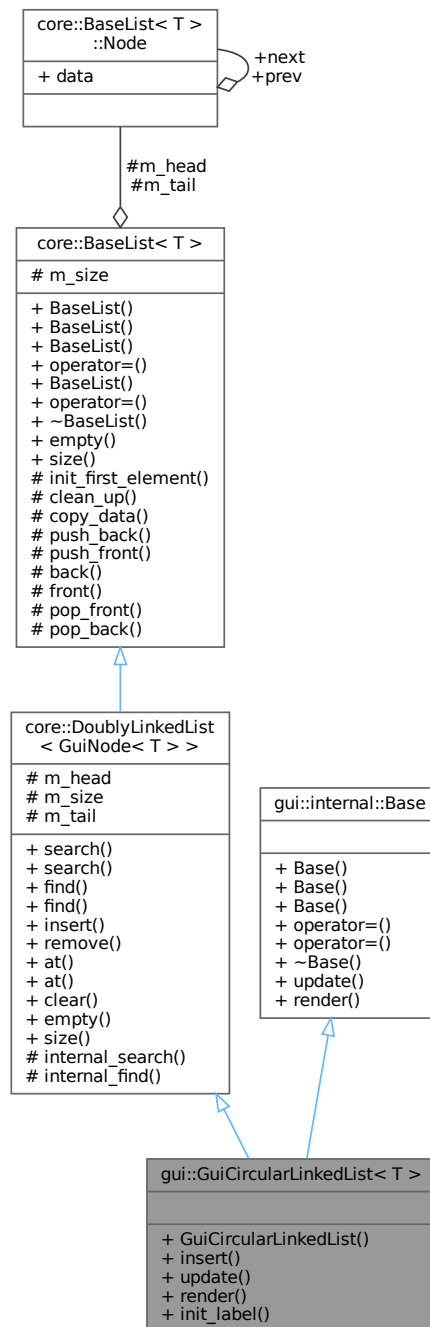
6.12 gui::GuiCircularLinkedList< T > Class Template Reference

```
#include <circular_linked_list_gui.hpp>
```

Inheritance diagram for gui::GuiCircularLinkedList< T >:



Collaboration diagram for gui::GuiCircularLinkedList< T >:



Public Member Functions

- [GuiCircularLinkedList](#) (std::initializer_list< [GuiNode< T > > init_list\)](#)
- void [insert](#) (std::size_t index, const T &elem)
- void [update](#) () override
- void [render](#) () override
- void [init_label](#) ()

Public Member Functions inherited from `core::DoublyLinkedList< GuiNode< T > >`

- `Node_ptr search` (const GuiNode< T > &elem)
- `cNode_ptr search` (const GuiNode< T > &elem) const
- `Node_ptr find` (std::size_t index)
- `cNode_ptr find` (std::size_t index) const
- `Node_ptr insert` (std::size_t index, const GuiNode< T > &elem)
- `Node_ptr remove` (std::size_t index)
- GuiNode< T > & `at` (std::size_t index)
- GuiNode< T > `at` (std::size_t index) const
- void `clear` ()
- bool `empty` () const
- std::size_t `size` () const

Public Member Functions inherited from `core::BaseList< T >`

- `BaseList` ()=default
- `BaseList` (std::initializer_list< T > init_list)
- `BaseList` (const `BaseList` &rhs)
- `BaseList` & `operator=` (const `BaseList` &rhs)
- `BaseList` (`BaseList` &&rhs) noexcept
- `BaseList` & `operator=` (`BaseList` &&rhs) noexcept
- `~BaseList` ()
- bool `empty` () const
- std::size_t `size` () const

Public Member Functions inherited from `gui::internal::Base`

- `Base` ()=default
- `Base` (const `Base` &)=default
- `Base` (`Base` &&)=default
- `Base` & `operator=` (const `Base` &)=default
- `Base` & `operator=` (`Base` &&)=default
- virtual `~Base` ()=default
- virtual void `update` ()=0
- virtual void `render` ()=0

Additional Inherited Members

Protected Types inherited from `core::DoublyLinkedList< GuiNode< T > >`

- using `Base` = `BaseList`< GuiNode< T > >
- using `Node` = typename `Base::Node`
- using `Node_ptr` = `Node` *
- using `cNode_ptr` = const `Node` *

Protected Types inherited from `core::BaseList< T >`

- using `Node_ptr` = `Node` *

Protected Member Functions inherited from [core::DoublyLinkedList< GuiNode< T > >](#)

- [Node_ptr internal_search](#) (const GuiNode< T > &elem)
- [Node_ptr internal_find](#) (std::size_t index)

Protected Member Functions inherited from [core::BaseList< T >](#)

- void [init_first_element](#) (const T &elem)
- void [clean_up](#) ()
- void [copy_data](#) (const [BaseList](#) &rhs)
- void [push_back](#) (const T &elem)
- void [push_front](#) (const T &elem)
- T & [back](#) () const
- T & [front](#) () const
- void [pop_front](#) ()
- void [pop_back](#) ()

Protected Attributes inherited from [core::DoublyLinkedList< GuiNode< T > >](#)

- [Node_ptr m_head](#)
- std::size_t [m_size](#)
- [Node_ptr m_tail](#)

Protected Attributes inherited from [core::BaseList< T >](#)

- [Node_ptr m_head](#) {nullptr}
- [Node_ptr m_tail](#) {nullptr}
- std::size_t [m_size](#) {}

6.12.1 Detailed Description

```
template<typename T>
class gui::GuiCircularLinkedList< T >
```

Definition at line 19 of file [circular_linked_list_gui.hpp](#).

6.12.2 Constructor & Destructor Documentation

6.12.2.1 GuiCircularLinkedList()

```
template<typename T >
gui::GuiCircularLinkedList< T >::GuiCircularLinkedList (
    std::initializer_list< GuiNode< T > > init_list )
```

Definition at line 65 of file [circular_linked_list_gui.hpp](#).

Here is the call graph for this function:



6.12.3 Member Function Documentation

6.12.3.1 init_label()

```
template<typename T >
void gui::GuiCircularLinkedList< T >::init_label
```

Definition at line 50 of file [circular_linked_list_gui.hpp](#).

Here is the caller graph for this function:



6.12.3.2 insert()

```
template<typename T >
void gui::GuiCircularLinkedList< T >::insert (
    std::size_t index,
    const T & elem )
```

Definition at line 72 of file [circular_linked_list_gui.hpp](#).

6.12.3.3 render()

```
template<typename T >
void gui::GuiCircularLinkedList< T >::render [override], [virtual]
```

Implements [gui::internal::Base](#).

Definition at line 129 of file [circular_linked_list_gui.hpp](#).

6.12.3.4 update()

```
template<typename T >
void gui::GuiCircularLinkedList< T >::update [override], [virtual]
```

Implements [gui::internal::Base](#).

Definition at line 143 of file [circular_linked_list_gui.hpp](#).

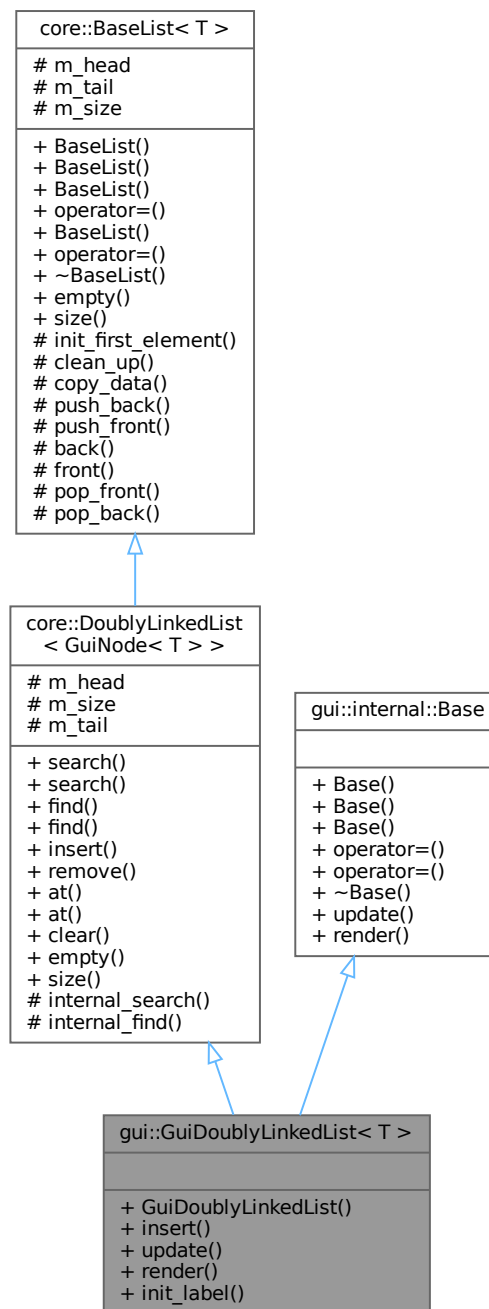
The documentation for this class was generated from the following file:

- [src/gui/circular_linked_list_gui.hpp](#)

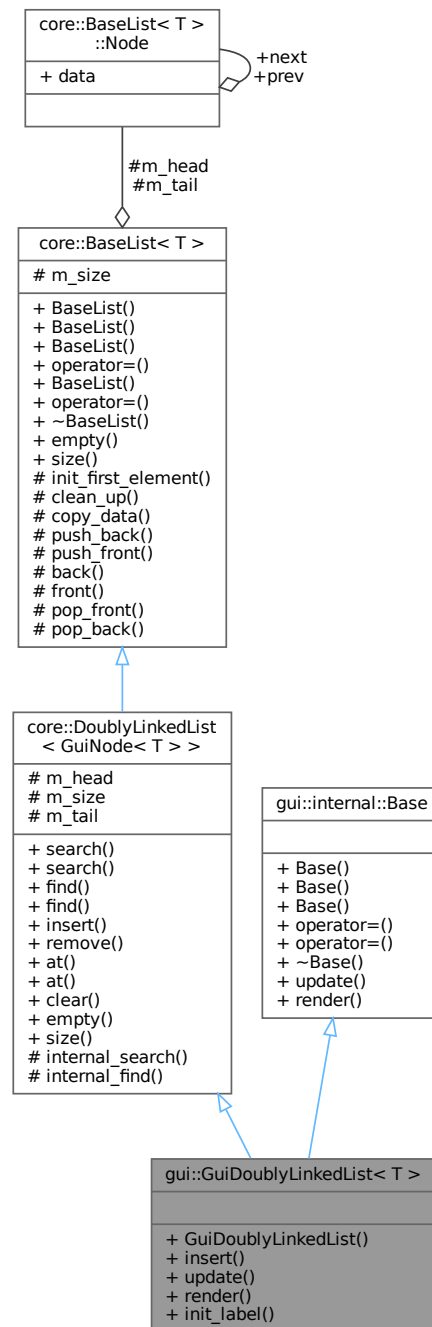
6.13 gui::GuiDoublyLinkedList< T > Class Template Reference

```
#include <doubly_linked_list_gui.hpp>
```

Inheritance diagram for gui::GuiDoublyLinkedList< T >:



Collaboration diagram for gui::GuiDoublyLinkedList< T >:



Public Member Functions

- `GuiDoublyLinkedList` (`std::initializer_list< GuiNode< T > > init_list`)
- `void insert` (`std::size_t index, const T &elem`)
- `void update` () override
- `void render` () override
- `void init_label` ()

Public Member Functions inherited from [core::DoublyLinkedList< GuiNode< T > >](#)

- [Node_ptr search](#) (const GuiNode< T > &elem)
- [cNode_ptr search](#) (const GuiNode< T > &elem) const
- [Node_ptr find](#) (std::size_t index)
- [cNode_ptr find](#) (std::size_t index) const
- [Node_ptr insert](#) (std::size_t index, const GuiNode< T > &elem)
- [Node_ptr remove](#) (std::size_t index)
- GuiNode< T > & [at](#) (std::size_t index)
- GuiNode< T > [at](#) (std::size_t index) const
- void [clear](#) ()
- bool [empty](#) () const
- std::size_t [size](#) () const

Public Member Functions inherited from [core::BaseList< T >](#)

- [BaseList](#) ()=default
- [BaseList](#) (std::initializer_list< T > init_list)
- [BaseList](#) (const [BaseList](#) &rhs)
- [BaseList](#) & [operator=](#) (const [BaseList](#) &rhs)
- [BaseList](#) ([BaseList](#) &&rhs) noexcept
- [BaseList](#) & [operator=](#) ([BaseList](#) &&rhs) noexcept
- [~BaseList](#) ()
- bool [empty](#) () const
- std::size_t [size](#) () const

Public Member Functions inherited from [gui::internal::Base](#)

- [Base](#) ()=default
- [Base](#) (const [Base](#) &)=default
- [Base](#) ([Base](#) &&)=default
- [Base](#) & [operator=](#) (const [Base](#) &)=default
- [Base](#) & [operator=](#) ([Base](#) &&)=default
- virtual [~Base](#) ()=default
- virtual void [update](#) ()=0
- virtual void [render](#) ()=0

Additional Inherited Members

Protected Types inherited from [core::DoublyLinkedList< GuiNode< T > >](#)

- using [Base](#) = [BaseList](#)< GuiNode< T > >
- using [Node](#) = typename Base::Node
- using [Node_ptr](#) = [Node](#) *
- using [cNode_ptr](#) = const [Node](#) *

Protected Types inherited from [core::BaseList< T >](#)

- using [Node_ptr](#) = [Node](#) *

Protected Member Functions inherited from [core::DoublyLinkedList< GuiNode< T > >](#)

- [Node_ptr internal_search](#) (const GuiNode< T > &elem)
- [Node_ptr internal_find](#) (std::size_t index)

Protected Member Functions inherited from [core::BaseList< T >](#)

- void [init_first_element](#) (const T &elem)
- void [clean_up](#) ()
- void [copy_data](#) (const [BaseList](#) &rhs)
- void [push_back](#) (const T &elem)
- void [push_front](#) (const T &elem)
- T & [back](#) () const
- T & [front](#) () const
- void [pop_front](#) ()
- void [pop_back](#) ()

Protected Attributes inherited from [core::DoublyLinkedList< GuiNode< T > >](#)

- [Node_ptr m_head](#)
- std::size_t [m_size](#)
- [Node_ptr m_tail](#)

Protected Attributes inherited from [core::BaseList< T >](#)

- [Node_ptr m_head](#) {nullptr}
- [Node_ptr m_tail](#) {nullptr}
- std::size_t [m_size](#) {}

6.13.1 Detailed Description

```
template<typename T>
class gui::GuiDoublyLinkedList< T >
```

Definition at line 17 of file [doubly_linked_list_gui.hpp](#).

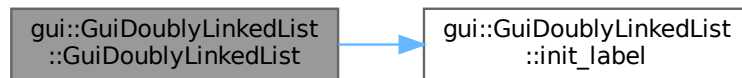
6.13.2 Constructor & Destructor Documentation

6.13.2.1 GuiDoublyLinkedList()

```
template<typename T >
gui::GuiDoublyLinkedList< T >::GuiDoublyLinkedList (
    std::initializer_list< GuiNode< T > > init_list )
```

Definition at line 62 of file [doubly_linked_list_gui.hpp](#).

Here is the call graph for this function:



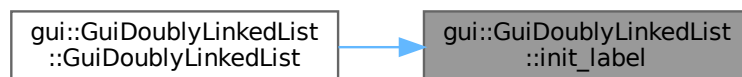
6.13.3 Member Function Documentation

6.13.3.1 init_label()

```
template<typename T >
void gui::GuiDoublyLinkedList< T >::init_label
```

Definition at line 47 of file [doubly_linked_list_gui.hpp](#).

Here is the caller graph for this function:



6.13.3.2 insert()

```
template<typename T >
void gui::GuiDoublyLinkedList< T >::insert (
    std::size_t index,
    const T & elem )
```

Definition at line 69 of file [doubly_linked_list_gui.hpp](#).

6.13.3.3 render()

```
template<typename T >
void gui::GuiDoublyLinkedList< T >::render [override], [virtual]
```

Implements [gui::internal::Base](#).

Definition at line 105 of file [doubly_linked_list_gui.hpp](#).

6.13.3.4 update()

```
template<typename T >
void gui::GuiDoublyLinkedList< T >::update [override], [virtual]
```

Implements [gui::internal::Base](#).

Definition at line 118 of file [doubly_linked_list_gui.hpp](#).

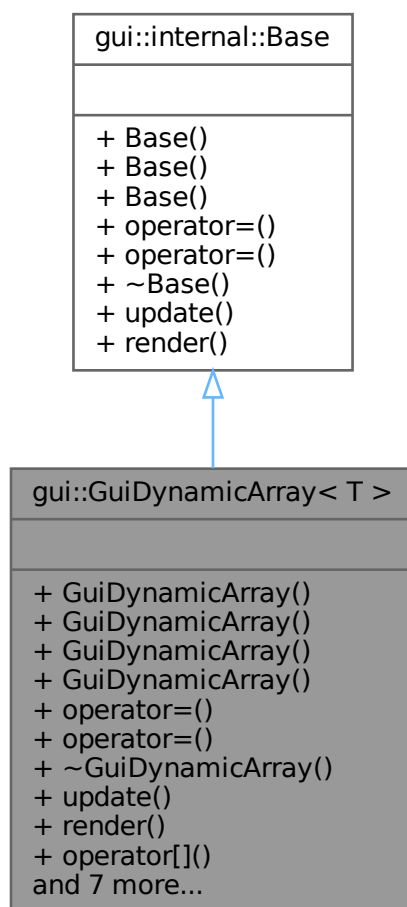
The documentation for this class was generated from the following file:

- [src/gui/doubly_linked_list_gui.hpp](#)

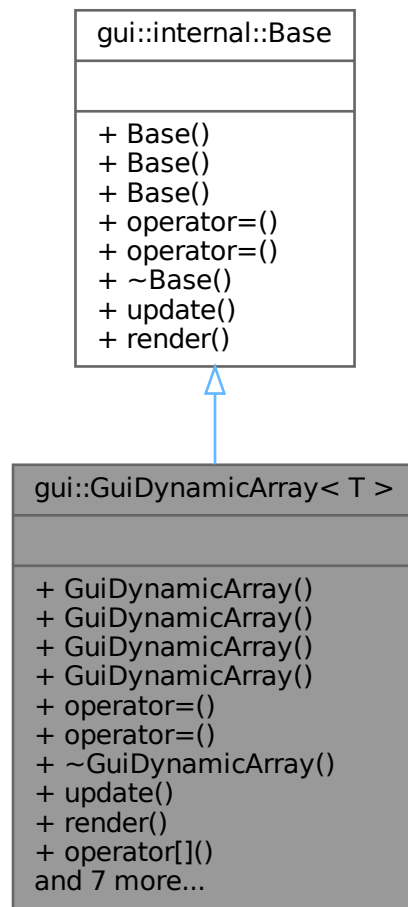
6.14 gui::GuiDynamicArray< T > Class Template Reference

```
#include <dynamic_array_gui.hpp>
```

Inheritance diagram for gui::GuiDynamicArray< T >:



Collaboration diagram for gui::GuiDynamicArray< T >:



Public Member Functions

- `GuiDynamicArray ()`
- `GuiDynamicArray (std::initializer_list< T > init_list)`
- `GuiDynamicArray (const GuiDynamicArray &other)`
- `GuiDynamicArray (GuiDynamicArray &&other) noexcept`
- `GuiDynamicArray & operator= (const GuiDynamicArray &other)`
- `GuiDynamicArray & operator= (GuiDynamicArray &&other) noexcept`
- `~GuiDynamicArray ()` override
- `void update ()` override
- `void render ()` override
- `T & operator[] (std::size_t idx)`
- `T operator[] (std::size_t idx) const`
- `void set_color_index (std::size_t idx, int color_index)`
- `void realloc (std::size_t capacity)`
- `std::size_t capacity () const`
- `std::size_t size () const`
- `void push (const T &value)`
- `void pop ()`

Public Member Functions inherited from [gui::internal::Base](#)

- [Base](#) ()=default
- [Base](#) (const [Base](#) &)=default
- [Base](#) ([Base](#) &&)=default
- [Base](#) & [operator=](#) (const [Base](#) &)=default
- [Base](#) & [operator=](#) ([Base](#) &&)=default
- virtual [~Base](#) ()=default
- virtual void [update](#) ()=0
- virtual void [render](#) ()=0

6.14.1 Detailed Description

```
template<typename T>
class gui::GuiDynamicArray< T >
```

Definition at line 17 of file [dynamic_array_gui.hpp](#).

6.14.2 Constructor & Destructor Documentation

6.14.2.1 GuiDynamicArray() [1/4]

```
template<typename T >
gui::GuiDynamicArray< T >::GuiDynamicArray
```

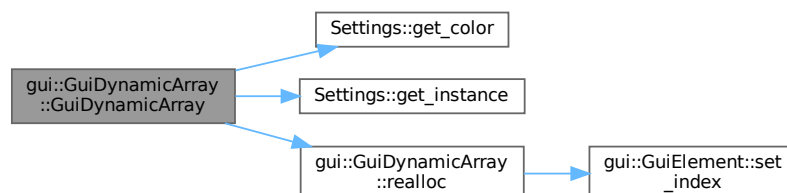
Definition at line 77 of file [dynamic_array_gui.hpp](#).

6.14.2.2 GuiDynamicArray() [2/4]

```
template<typename T >
gui::GuiDynamicArray< T >::GuiDynamicArray (
    std::initializer_list< T > init_list )
```

Definition at line 84 of file [dynamic_array_gui.hpp](#).

Here is the call graph for this function:



6.14.2.3 GuiDynamicArray() [3/4]

```
template<typename T >
gui::GuiDynamicArray< T >::GuiDynamicArray (
    const GuiDynamicArray< T > & other )
```

Definition at line 95 of file [dynamic_array_gui.hpp](#).

6.14.2.4 GuiDynamicArray() [4/4]

```
template<typename T >
gui::GuiDynamicArray< T >::GuiDynamicArray (
    GuiDynamicArray< T > && other ) [noexcept]
```

Definition at line 105 of file [dynamic_array_gui.hpp](#).

6.14.2.5 ~GuiDynamicArray()

```
template<typename T >
gui::GuiDynamicArray< T >::~~GuiDynamicArray [override]
```

Definition at line 143 of file [dynamic_array_gui.hpp](#).

6.14.3 Member Function Documentation

6.14.3.1 capacity()

```
template<typename T >
std::size_t gui::GuiDynamicArray< T >::capacity
```

Definition at line 187 of file [dynamic_array_gui.hpp](#).

6.14.3.2 operator=() [1/2]

```
template<typename T >
GuiDynamicArray< T > & gui::GuiDynamicArray< T >::operator= (
    const GuiDynamicArray< T > & other )
```

Definition at line 113 of file [dynamic_array_gui.hpp](#).

6.14.3.3 operator=() [2/2]

```
template<typename T >
GuiDynamicArray< T > & gui::GuiDynamicArray< T >::operator= (
    GuiDynamicArray< T > && other ) [noexcept]
```

Definition at line 129 of file [dynamic_array_gui.hpp](#).

6.14.3.4 operator[]() [1/2]

```
template<typename T >
T & gui::GuiDynamicArray< T >::operator[] (
    std::size_t idx )
```

Definition at line 172 of file [dynamic_array_gui.hpp](#).

6.14.3.5 operator[]() [2/2]

```
template<typename T >
T gui::GuiDynamicArray< T >::operator[] (
    std::size_t idx ) const
```

Definition at line 177 of file [dynamic_array_gui.hpp](#).

6.14.3.6 pop()

```
template<typename T >
void gui::GuiDynamicArray< T >::pop
```

Definition at line 208 of file [dynamic_array_gui.hpp](#).

6.14.3.7 push()

```
template<typename T >
void gui::GuiDynamicArray< T >::push (
    const T & value )
```

Definition at line 197 of file [dynamic_array_gui.hpp](#).

6.14.3.8 realloc()

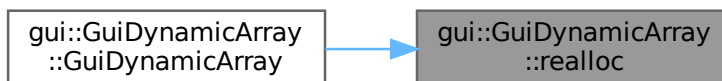
```
template<typename T >
void gui::GuiDynamicArray< T >::realloc (
    std::size_t capacity )
```

Definition at line 55 of file [dynamic_array_gui.hpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



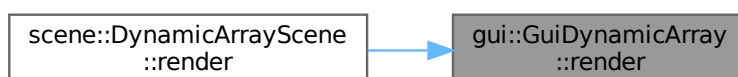
6.14.3.9 render()

```
template<typename T >
void gui::GuiDynamicArray< T >::render [override], [virtual]
```

Implements [gui::internal::Base](#).

Definition at line 151 of file [dynamic_array_gui.hpp](#).

Here is the caller graph for this function:



6.14.3.10 set_color_index()

```
template<typename T >
void gui::GuiDynamicArray< T >::set_color_index (
    std::size_t idx,
    int color_index )
```

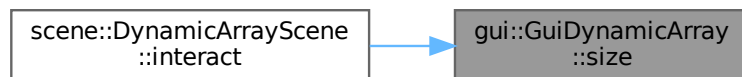
Definition at line 182 of file [dynamic_array_gui.hpp](#).

6.14.3.11 size()

```
template<typename T >
std::size_t gui::GuiDynamicArray< T >::size
```

Definition at line 192 of file [dynamic_array_gui.hpp](#).

Here is the caller graph for this function:



6.14.3.12 update()

```
template<typename T >
void gui::GuiDynamicArray< T >::update [override], [virtual]
```

Implements [gui::internal::Base](#).

Definition at line 162 of file [dynamic_array_gui.hpp](#).

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

- [src/gui/dynamic_array_gui.hpp](#)

6.15 gui::GuiElement< T > Class Template Reference

```
#include <element_gui.hpp>
```

Collaboration diagram for gui::GuiElement< T >:

gui::GuiElement< T >
+ side + init_pos
+ GuiElement() + GuiElement() + render() + set_pos() + set_color_index() + get_pos() + get_value() + get_value() + set_value() + set_index()

Public Member Functions

- [GuiElement](#) ()=default
- [GuiElement](#) (const T &value, std::size_t index)
- void [render](#) ()
- void [set_pos](#) (Vector2 pos)
- void [set_color_index](#) (int color_index)
- Vector2 [get_pos](#) () const
- T & [get_value](#) ()
- T [get_value](#) () const
- void [set_value](#) (const T &value)
- void [set_index](#) (std::size_t index)

Static Public Attributes

- static constexpr int [side](#) = 20
- static constexpr Vector2 [init_pos](#)

6.15.1 Detailed Description

```
template<typename T>
class gui::GuiElement< T >
```

Definition at line 17 of file [element_gui.hpp](#).

6.15.2 Constructor & Destructor Documentation

6.15.2.1 GuiElement() [1/2]

```
template<typename T >  
gui::GuiElement< T >::GuiElement ( ) [default]
```

6.15.2.2 GuiElement() [2/2]

```
template<typename T >  
gui::GuiElement< T >::GuiElement (   
    const T & value,  
    std::size_t index )
```

Definition at line 50 of file [element_gui.hpp](#).

6.15.3 Member Function Documentation

6.15.3.1 get_pos()

```
template<typename T >  
Vector2 gui::GuiElement< T >::get_pos ( ) const
```

6.15.3.2 get_value() [1/2]

```
template<typename T >  
T & gui::GuiElement< T >::get_value
```

Definition at line 100 of file [element_gui.hpp](#).

6.15.3.3 get_value() [2/2]

```
template<typename T >  
T gui::GuiElement< T >::get_value
```

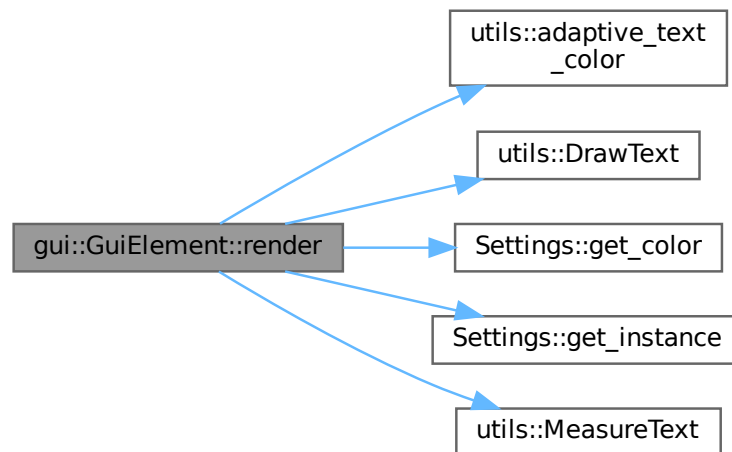
Definition at line 105 of file [element_gui.hpp](#).

6.15.3.4 render()

```
template<typename T >
void gui::GuiElement< T >::render
```

Definition at line 54 of file [element_gui.hpp](#).

Here is the call graph for this function:



6.15.3.5 set_color_index()

```
template<typename T >
void gui::GuiElement< T >::set_color_index (
    int color_index )
```

Definition at line 95 of file [element_gui.hpp](#).

Here is the caller graph for this function:

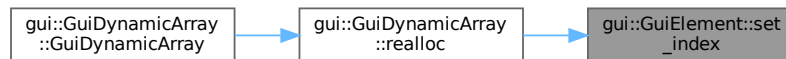


6.15.3.6 set_index()

```
template<typename T >
void gui::GuiElement< T >::set_index (
    std::size_t index )
```

Definition at line 115 of file [element_gui.hpp](#).

Here is the caller graph for this function:



6.15.3.7 set_pos()

```
template<typename T >
void gui::GuiElement< T >::set_pos (
    Vector2 pos )
```

Definition at line 90 of file [element_gui.hpp](#).

6.15.3.8 set_value()

```
template<typename T >
void gui::GuiElement< T >::set_value (
    const T & value )
```

Definition at line 110 of file [element_gui.hpp](#).

6.15.4 Member Data Documentation

6.15.4.1 init_pos

```
template<typename T >
constexpr Vector2 gui::GuiElement< T >::init_pos [static], [constexpr]
```

Initial value:

```
{
    constants::sidebar_width +
    static_cast<float>(constants::scene_width -
                      constants::sidebar_width) /
    2,
    0}
```

Definition at line 28 of file [element_gui.hpp](#).

6.15.4.2 side

```
template<typename T >
constexpr int gui::GuiElement< T >::side = 20 [static], [constexpr]
```

Definition at line 27 of file [element_gui.hpp](#).

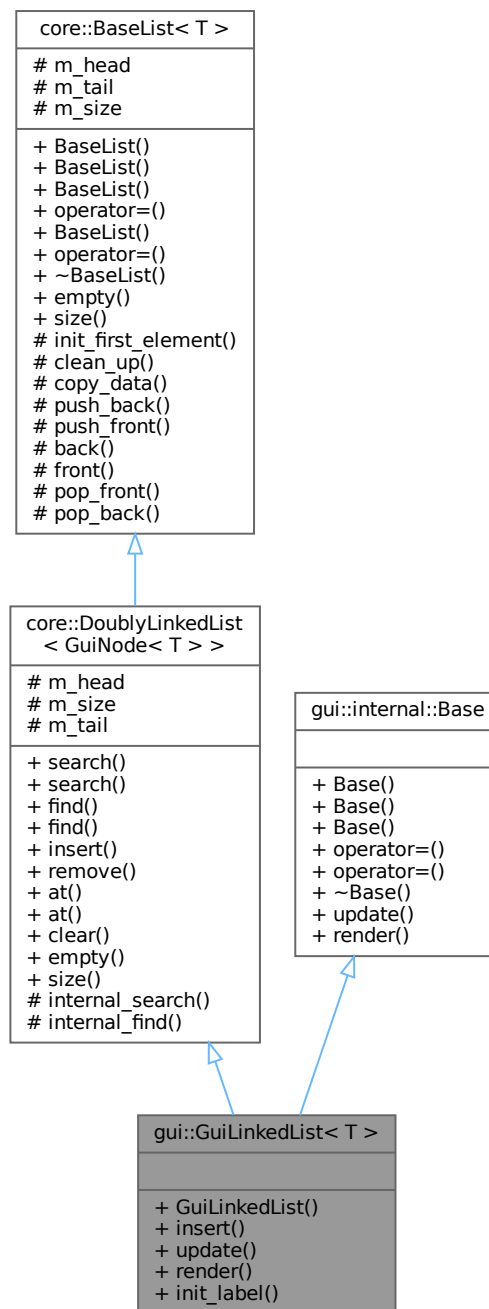
The documentation for this class was generated from the following file:

- [src/gui/element_gui.hpp](#)

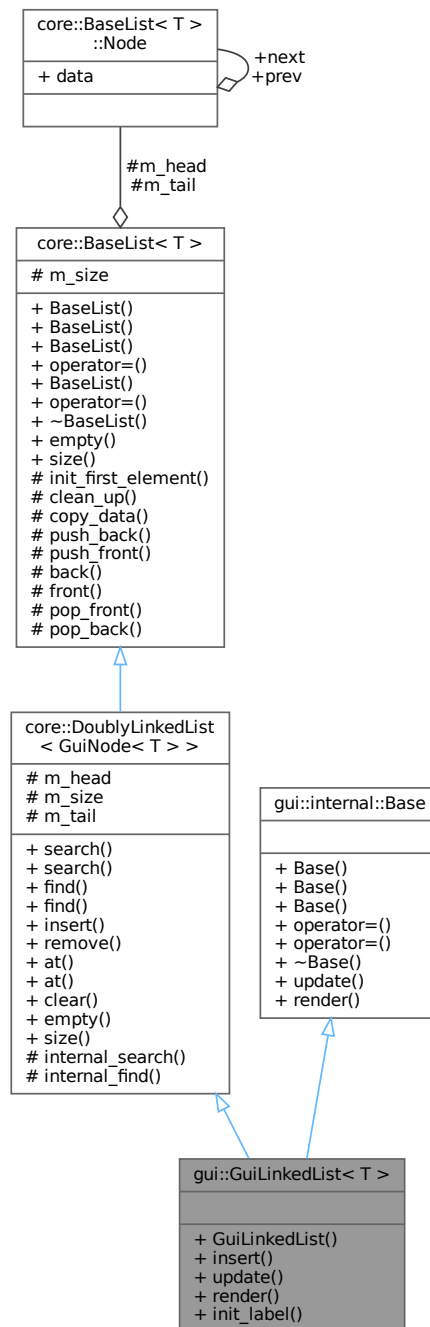
6.16 gui::GuiLinkedList< T > Class Template Reference

```
#include <linked_list_gui.hpp>
```

Inheritance diagram for gui::GuiLinkedList< T >:



Collaboration diagram for gui::GuiLinkedList< T >:



Public Member Functions

- `GuiLinkedList` (`std::initializer_list< GuiNode< T > > init_list`)
- `void insert` (`std::size_t index, const T &elem`)
- `void update` () override
- `void render` () override
- `void init_label` ()

Public Member Functions inherited from [core::DoublyLinkedList< GuiNode< T > >](#)

- [Node_ptr search](#) (const GuiNode< T > &elem)
- [cNode_ptr search](#) (const GuiNode< T > &elem) const
- [Node_ptr find](#) (std::size_t index)
- [cNode_ptr find](#) (std::size_t index) const
- [Node_ptr insert](#) (std::size_t index, const GuiNode< T > &elem)
- [Node_ptr remove](#) (std::size_t index)
- GuiNode< T > & [at](#) (std::size_t index)
- GuiNode< T > [at](#) (std::size_t index) const
- void [clear](#) ()
- bool [empty](#) () const
- std::size_t [size](#) () const

Public Member Functions inherited from [core::BaseList< T >](#)

- [BaseList](#) ()=default
- [BaseList](#) (std::initializer_list< T > init_list)
- [BaseList](#) (const [BaseList](#) &rhs)
- [BaseList](#) & [operator=](#) (const [BaseList](#) &rhs)
- [BaseList](#) ([BaseList](#) &&rhs) noexcept
- [BaseList](#) & [operator=](#) ([BaseList](#) &&rhs) noexcept
- [~BaseList](#) ()
- bool [empty](#) () const
- std::size_t [size](#) () const

Public Member Functions inherited from [gui::internal::Base](#)

- [Base](#) ()=default
- [Base](#) (const [Base](#) &)=default
- [Base](#) ([Base](#) &&)=default
- [Base](#) & [operator=](#) (const [Base](#) &)=default
- [Base](#) & [operator=](#) ([Base](#) &&)=default
- virtual [~Base](#) ()=default
- virtual void [update](#) ()=0
- virtual void [render](#) ()=0

Additional Inherited Members

Protected Types inherited from [core::DoublyLinkedList< GuiNode< T > >](#)

- using [Base](#) = [BaseList](#)< GuiNode< T > >
- using [Node](#) = typename Base::Node
- using [Node_ptr](#) = [Node](#) *
- using [cNode_ptr](#) = const [Node](#) *

Protected Types inherited from [core::BaseList< T >](#)

- using [Node_ptr](#) = [Node](#) *

Protected Member Functions inherited from [core::DoublyLinkedList< GuiNode< T > >](#)

- [Node_ptr internal_search](#) (const GuiNode< T > &elem)
- [Node_ptr internal_find](#) (std::size_t index)

Protected Member Functions inherited from [core::BaseList< T >](#)

- void [init_first_element](#) (const T &elem)
- void [clean_up](#) ()
- void [copy_data](#) (const [BaseList](#) &rhs)
- void [push_back](#) (const T &elem)
- void [push_front](#) (const T &elem)
- T & [back](#) () const
- T & [front](#) () const
- void [pop_front](#) ()
- void [pop_back](#) ()

Protected Attributes inherited from [core::DoublyLinkedList< GuiNode< T > >](#)

- [Node_ptr m_head](#)
- std::size_t [m_size](#)
- [Node_ptr m_tail](#)

Protected Attributes inherited from [core::BaseList< T >](#)

- [Node_ptr m_head](#) {nullptr}
- [Node_ptr m_tail](#) {nullptr}
- std::size_t [m_size](#) {}

6.16.1 Detailed Description

```
template<typename T>
class gui::GuiLinkedList< T >
```

Definition at line 18 of file [linked_list_gui.hpp](#).

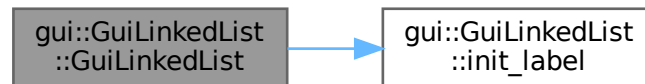
6.16.2 Constructor & Destructor Documentation

6.16.2.1 GuiLinkedList()

```
template<typename T >
gui::GuiLinkedList< T >::GuiLinkedList (
    std::initializer_list< GuiNode< T > > init_list )
```

Definition at line 63 of file [linked_list_gui.hpp](#).

Here is the call graph for this function:



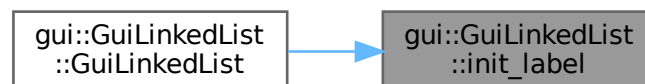
6.16.3 Member Function Documentation

6.16.3.1 init_label()

```
template<typename T >
void gui::GuiLinkedList< T >::init_label
```

Definition at line 48 of file [linked_list_gui.hpp](#).

Here is the caller graph for this function:



6.16.3.2 insert()

```
template<typename T >
void gui::GuiLinkedList< T >::insert (
    std::size_t index,
    const T & elem )
```

Definition at line 69 of file [linked_list_gui.hpp](#).

6.16.3.3 render()

```
template<typename T >
void gui::GuiLinkedList< T >::render [override], [virtual]
```

Implements [gui::internal::Base](#).

Definition at line 95 of file [linked_list_gui.hpp](#).

6.16.3.4 update()

```
template<typename T >
void gui::GuiLinkedList< T >::update [override], [virtual]
```

Implements [gui::internal::Base](#).

Definition at line 108 of file [linked_list_gui.hpp](#).

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

- [src/gui/linked_list_gui.hpp](#)

6.17 gui::GuiNode< T > Class Template Reference

```
#include <node_gui.hpp>
```

Collaboration diagram for gui::GuiNode< T >:

gui::GuiNode< T >
+ radius
+ GuiNode() + render() + set_pos() + get_pos() + set_color_index() + set_value() + get_value() + set_label()

Public Member Functions

- [GuiNode](#) (const T &value)
- void [render](#) ()
- void [set_pos](#) (Vector2 pos)
- Vector2 [get_pos](#) () const
- void [set_color_index](#) (int color_index)
- void [set_value](#) (const T &value)
- T & [get_value](#) ()
- void [set_label](#) (const char *label)

Static Public Attributes

- static constexpr int [radius](#) = 20

6.17.1 Detailed Description

```
template<typename T>
class gui::GuiNode< T >
```

Definition at line 16 of file [node_gui.hpp](#).

6.17.2 Constructor & Destructor Documentation

6.17.2.1 GuiNode()

```
template<typename T >
gui::GuiNode< T >::GuiNode (
    const T & value ) [explicit]
```

Definition at line 44 of file [node_gui.hpp](#).

6.17.3 Member Function Documentation

6.17.3.1 get_pos()

```
template<typename T >
Vector2 gui::GuiNode< T >::get_pos
```

Definition at line 97 of file [node_gui.hpp](#).

6.17.3.2 get_value()

```
template<typename T >  
T & gui::GuiNode< T >::get_value
```

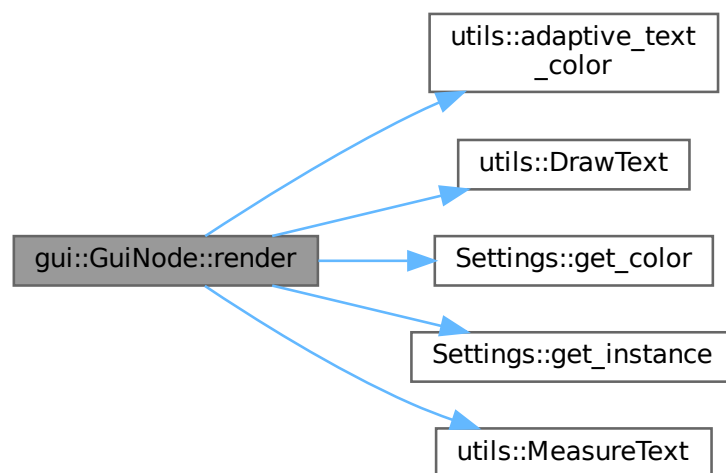
Definition at line 87 of file [node_gui.hpp](#).

6.17.3.3 render()

```
template<typename T >  
void gui::GuiNode< T >::render
```

Definition at line 47 of file [node_gui.hpp](#).

Here is the call graph for this function:



6.17.3.4 set_color_index()

```
template<typename T >  
void gui::GuiNode< T >::set_color_index (  
    int color_index )
```

Definition at line 77 of file [node_gui.hpp](#).

6.17.3.5 set_label()

```
template<typename T >
void gui::GuiNode< T >::set_label (
    const char * label )
```

Definition at line 102 of file [node_gui.hpp](#).

6.17.3.6 set_pos()

```
template<typename T >
void gui::GuiNode< T >::set_pos (
    Vector2 pos )
```

Definition at line 92 of file [node_gui.hpp](#).

6.17.3.7 set_value()

```
template<typename T >
void gui::GuiNode< T >::set_value (
    const T & value )
```

Definition at line 82 of file [node_gui.hpp](#).

6.17.4 Member Data Documentation

6.17.4.1 radius

```
template<typename T >
constexpr int gui::GuiNode< T >::radius = 20 [static], [constexpr]
```

Definition at line 30 of file [node_gui.hpp](#).

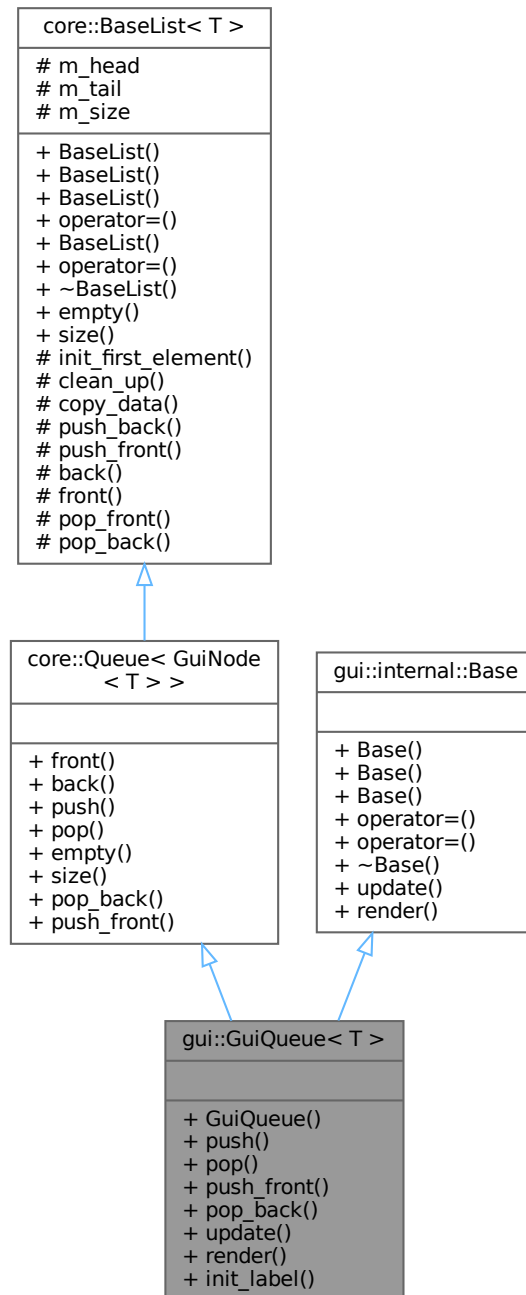
The documentation for this class was generated from the following file:

- [src/gui/node_gui.hpp](#)

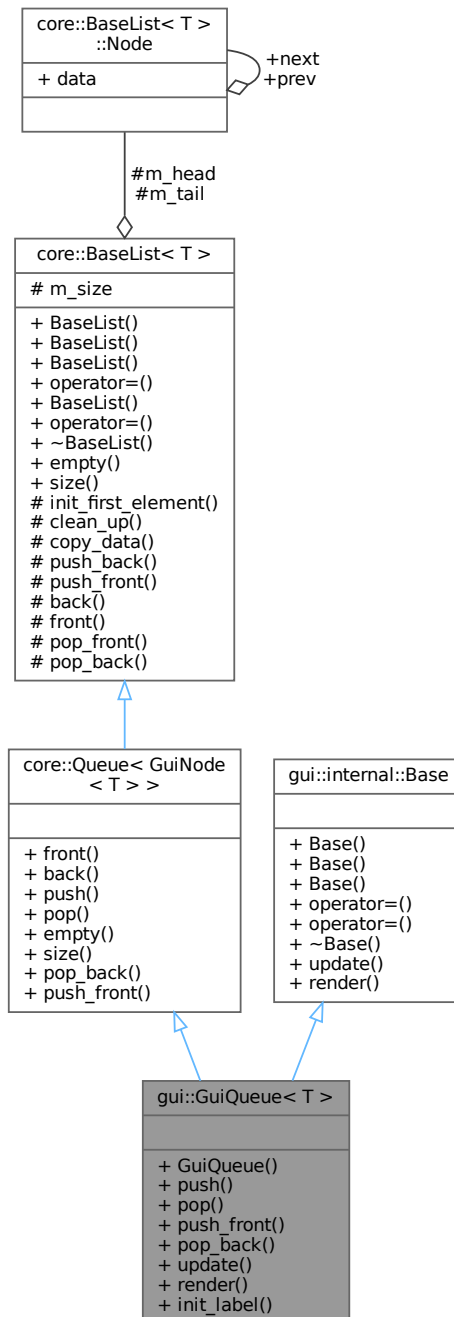
6.18 gui::GuiQueue< T > Class Template Reference

```
#include <queue_gui.hpp>
```

Inheritance diagram for gui::GuiQueue< T >:



Collaboration diagram for `gui::GuiQueue< T >`:



Public Member Functions

- `GuiQueue` (`std::initializer_list< GuiNode < T > > init_list`)
- `void push` (`const T &elem`)
- `void pop` ()
- `void push_front` (`const T &elem`)
- `void pop_back` ()

- void [update](#) () override
- void [render](#) () override
- void [init_label](#) ()

Public Member Functions inherited from [core::Queue< GuiNode< T > >](#)

- GuiNode< T > & [front](#) () const
- GuiNode< T > & [back](#) () const
- void [push](#) (const GuiNode< T > &elem)
- void [pop](#) ()
- bool [empty](#) () const
- std::size_t [size](#) () const
- void [pop_back](#) ()
- void [push_front](#) (const GuiNode< T > &elem)

Public Member Functions inherited from [core::BaseList< T >](#)

- [BaseList](#) ()=default
- [BaseList](#) (std::initializer_list< T > init_list)
- [BaseList](#) (const [BaseList](#) &rhs)
- [BaseList](#) & [operator=](#) (const [BaseList](#) &rhs)
- [BaseList](#) ([BaseList](#) &&rhs) noexcept
- [BaseList](#) & [operator=](#) ([BaseList](#) &&rhs) noexcept
- [~BaseList](#) ()
- bool [empty](#) () const
- std::size_t [size](#) () const

Public Member Functions inherited from [gui::internal::Base](#)

- [Base](#) ()=default
- [Base](#) (const [Base](#) &)=default
- [Base](#) ([Base](#) &&)=default
- [Base](#) & [operator=](#) (const [Base](#) &)=default
- [Base](#) & [operator=](#) ([Base](#) &&)=default
- virtual [~Base](#) ()=default
- virtual void [update](#) ()=0
- virtual void [render](#) ()=0

Additional Inherited Members

Protected Types inherited from [core::BaseList< T >](#)

- using [Node_ptr](#) = [Node](#) *

Protected Member Functions inherited from `core::BaseList< T >`

- void `init_first_element` (const T &elem)
- void `clean_up` ()
- void `copy_data` (const `BaseList` &rhs)
- void `push_back` (const T &elem)
- void `push_front` (const T &elem)
- T & `back` () const
- T & `front` () const
- void `pop_front` ()
- void `pop_back` ()

Protected Attributes inherited from `core::BaseList< T >`

- `Node_ptr m_head` {nullptr}
- `Node_ptr m_tail` {nullptr}
- `std::size_t m_size` {}

6.18.1 Detailed Description

```
template<typename T>
class gui::GuiQueue< T >
```

Definition at line 17 of file `queue_gui.hpp`.

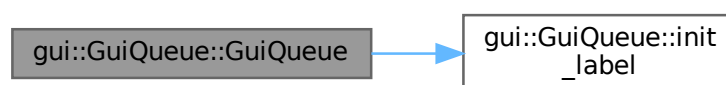
6.18.2 Constructor & Destructor Documentation

6.18.2.1 `GuiQueue()`

```
template<typename T >
gui::GuiQueue< T >::GuiQueue (
    std::initializer_list< GuiNode< T > > init_list )
```

Definition at line 66 of file `queue_gui.hpp`.

Here is the call graph for this function:



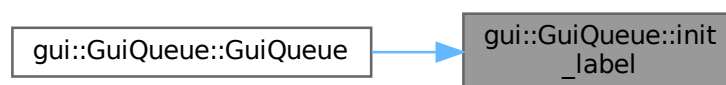
6.18.3 Member Function Documentation

6.18.3.1 init_label()

```
template<typename T >  
void gui::GuiQueue< T >::init_label
```

Definition at line 51 of file [queue_gui.hpp](#).

Here is the caller graph for this function:



6.18.3.2 pop()

```
template<typename T >  
void gui::GuiQueue< T >::pop
```

Definition at line 77 of file [queue_gui.hpp](#).

6.18.3.3 pop_back()

```
template<typename T >  
void gui::GuiQueue< T >::pop_back
```

Definition at line 87 of file [queue_gui.hpp](#).

6.18.3.4 push()

```
template<typename T >  
void gui::GuiQueue< T >::push (  
    const T & elem )
```

Definition at line 72 of file [queue_gui.hpp](#).

6.18.3.5 push_front()

```
template<typename T >
void gui::GuiQueue< T >::push_front (
    const T & elem )
```

Definition at line 82 of file [queue_gui.hpp](#).

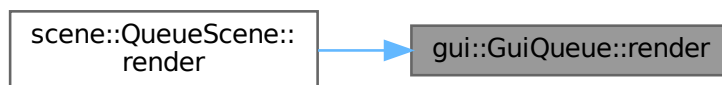
6.18.3.6 render()

```
template<typename T >
void gui::GuiQueue< T >::render [override], [virtual]
```

Implements [gui::internal::Base](#).

Definition at line 113 of file [queue_gui.hpp](#).

Here is the caller graph for this function:



6.18.3.7 update()

```
template<typename T >
void gui::GuiQueue< T >::update [override], [virtual]
```

Implements [gui::internal::Base](#).

Definition at line 126 of file [queue_gui.hpp](#).

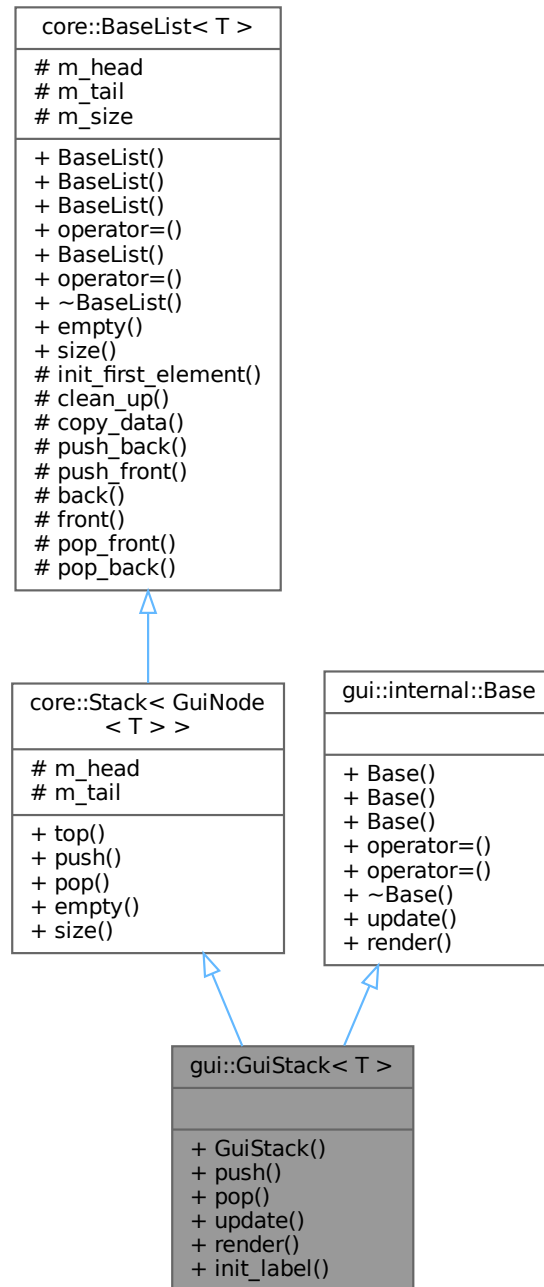
The documentation for this class was generated from the following file:

- [src/gui/queue_gui.hpp](#)

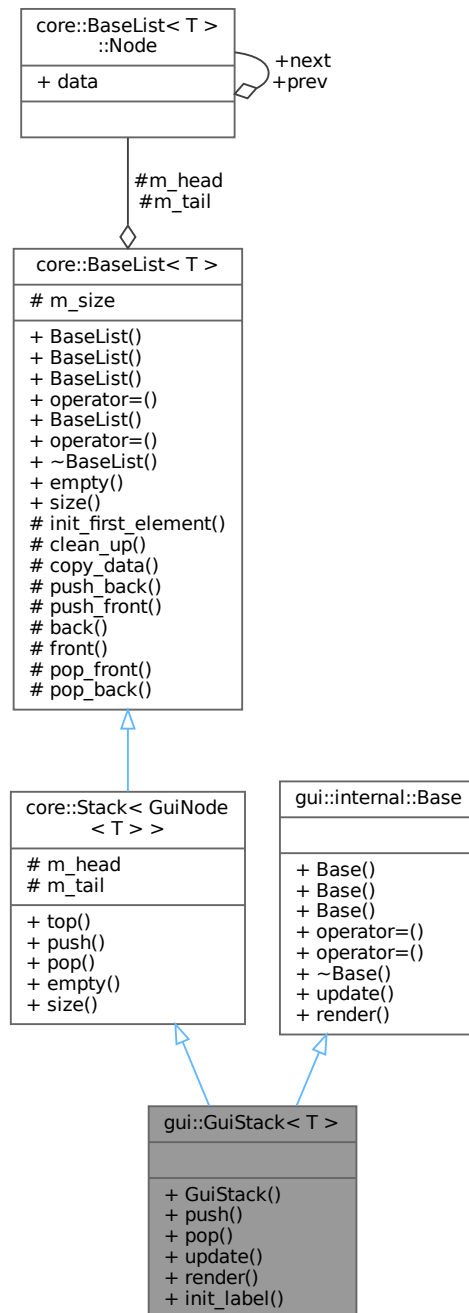
6.19 gui::GuiStack< T > Class Template Reference

```
#include <stack_gui.hpp>
```

Inheritance diagram for gui::GuiStack< T >:



Collaboration diagram for `gui::GuiStack< T >`:



Public Member Functions

- `GuiStack` (`std::initializer_list< GuiNode< T > >` `init_list`)
- `void push` (`const T &elem`)
- `void pop` ()
- `void update` () override
- `void render` () override
- `void init_label` ()

Public Member Functions inherited from [core::Stack< GuiNode< T > >](#)

- [GuiNode< T > & top](#) () const
- void [push](#) (const [GuiNode< T > &elem](#))
- void [pop](#) ()
- bool [empty](#) () const
- std::size_t [size](#) () const

Public Member Functions inherited from [core::BaseList< T >](#)

- [BaseList](#) ()=default
- [BaseList](#) (std::initializer_list< T > init_list)
- [BaseList](#) (const [BaseList](#) &rhs)
- [BaseList](#) & [operator=](#) (const [BaseList](#) &rhs)
- [BaseList](#) ([BaseList](#) &&rhs) noexcept
- [BaseList](#) & [operator=](#) ([BaseList](#) &&rhs) noexcept
- [~BaseList](#) ()
- bool [empty](#) () const
- std::size_t [size](#) () const

Public Member Functions inherited from [gui::internal::Base](#)

- [Base](#) ()=default
- [Base](#) (const [Base](#) &)=default
- [Base](#) ([Base](#) &&)=default
- [Base](#) & [operator=](#) (const [Base](#) &)=default
- [Base](#) & [operator=](#) ([Base](#) &&)=default
- virtual [~Base](#) ()=default
- virtual void [update](#) ()=0
- virtual void [render](#) ()=0

Additional Inherited Members**Protected Types inherited from [core::Stack< GuiNode< T > >](#)**

- using [Base](#) = [BaseList< GuiNode< T > >](#)

Protected Types inherited from [core::BaseList< T >](#)

- using [Node_ptr](#) = [Node](#) *

Protected Member Functions inherited from [core::BaseList< T >](#)

- void [init_first_element](#) (const T &elem)
- void [clean_up](#) ()
- void [copy_data](#) (const [BaseList](#) &rhs)
- void [push_back](#) (const T &elem)
- void [push_front](#) (const T &elem)
- T & [back](#) () const
- T & [front](#) () const
- void [pop_front](#) ()
- void [pop_back](#) ()

Protected Attributes inherited from [core::Stack< GuiNode< T > >](#)

- [Node_ptr m_head](#)
- [Node_ptr m_tail](#)

Protected Attributes inherited from [core::BaseList< T >](#)

- [Node_ptr m_head](#) {nullptr}
- [Node_ptr m_tail](#) {nullptr}
- [std::size_t m_size](#) {}

6.19.1 Detailed Description

```
template<typename T>
class gui::GuiStack< T >
```

Definition at line 17 of file [stack_gui.hpp](#).

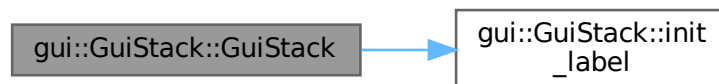
6.19.2 Constructor & Destructor Documentation

6.19.2.1 GuiStack()

```
template<typename T >
gui::GuiStack< T >::GuiStack (
    std::initializer_list< GuiNode< T > > init_list )
```

Definition at line 54 of file [stack_gui.hpp](#).

Here is the call graph for this function:



6.19.3 Member Function Documentation

6.19.3.1 init_label()

```
template<typename T >
void gui::GuiStack< T >::init_label
```

Definition at line 47 of file [stack_gui.hpp](#).

Here is the caller graph for this function:



6.19.3.2 pop()

```
template<typename T >
void gui::GuiStack< T >::pop
```

Definition at line 65 of file [stack_gui.hpp](#).

6.19.3.3 push()

```
template<typename T >
void gui::GuiStack< T >::push (
    const T & elem )
```

Definition at line 60 of file [stack_gui.hpp](#).

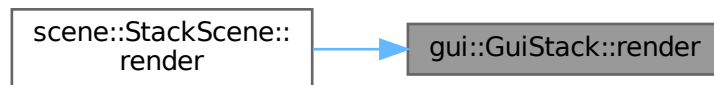
6.19.3.4 render()

```
template<typename T >
void gui::GuiStack< T >::render [override], [virtual]
```

Implements [gui::internal::Base](#).

Definition at line 91 of file [stack_gui.hpp](#).

Here is the caller graph for this function:



6.19.3.5 update()

```
template<typename T >
void gui::GuiStack< T >::update [override], [virtual]
```

Implements [gui::internal::Base](#).

Definition at line 104 of file [stack_gui.hpp](#).

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

- [src/gui/stack_gui.hpp](#)

6.20 component::MenuItem Class Reference

```
#include <menu_item.hpp>
```

Collaboration diagram for component::MenuItem:

component::MenuItem
+ block_width + block_height + button_width + button_height
+ MenuItem() + MenuItem() + x() + y() + render() + clicked() + reset()

Public Member Functions

- [MenuItem](#) ()=default
- [MenuItem](#) (int scene, const char *text, int [x](#), int [y](#), const char *img_path)
- int [x](#) () const
- int [y](#) () const
- void [render](#) ()
- bool [clicked](#) () const
- void [reset](#) ()

Static Public Attributes

- static constexpr int [block_width](#) = 300
- static constexpr int [block_height](#) = 200
- static constexpr int [button_width](#) = [block_width](#)
- static constexpr int [button_height](#) = 50

6.20.1 Detailed Description

Definition at line 8 of file [menu_item.hpp](#).

6.20.2 Constructor & Destructor Documentation

6.20.2.1 MenuItem() [1/2]

```
component::MenuItem::MenuItem ( ) [default]
```

6.20.2.2 MenuItem() [2/2]

```
component::MenuItem::MenuItem (
    int scene,
    const char * text,
    int x,
    int y,
    const char * img_path )
```

Definition at line 8 of file [menu_item.cpp](#).

6.20.3 Member Function Documentation

6.20.3.1 clicked()

```
bool component::MenuItem::clicked ( ) const
```

Definition at line 38 of file [menu_item.cpp](#).

6.20.3.2 render()

```
void component::MenuItem::render ( )
```

Definition at line 19 of file [menu_item.cpp](#).

6.20.3.3 reset()

```
void component::MenuItem::reset ( )
```

Definition at line 40 of file [menu_item.cpp](#).

6.20.3.4 x()

```
int component::MenuItem::x ( ) const
```

Definition at line 16 of file [menu_item.cpp](#).

6.20.3.5 y()

```
int component::MenuItem::y ( ) const
```

Definition at line 17 of file [menu_item.cpp](#).

6.20.4 Member Data Documentation

6.20.4.1 block_height

```
constexpr int component::MenuItem::block_height = 200 [static], [constexpr]
```

Definition at line 20 of file [menu_item.hpp](#).

6.20.4.2 block_width

```
constexpr int component::MenuItem::block_width = 300 [static], [constexpr]
```

Definition at line 19 of file [menu_item.hpp](#).

6.20.4.3 button_height

```
constexpr int component::MenuItem::button_height = 50 [static], [constexpr]
```

Definition at line 22 of file [menu_item.hpp](#).

6.20.4.4 button_width

```
constexpr int component::MenuItem::button_width = block_width [static], [constexpr]
```

Definition at line 21 of file [menu_item.hpp](#).

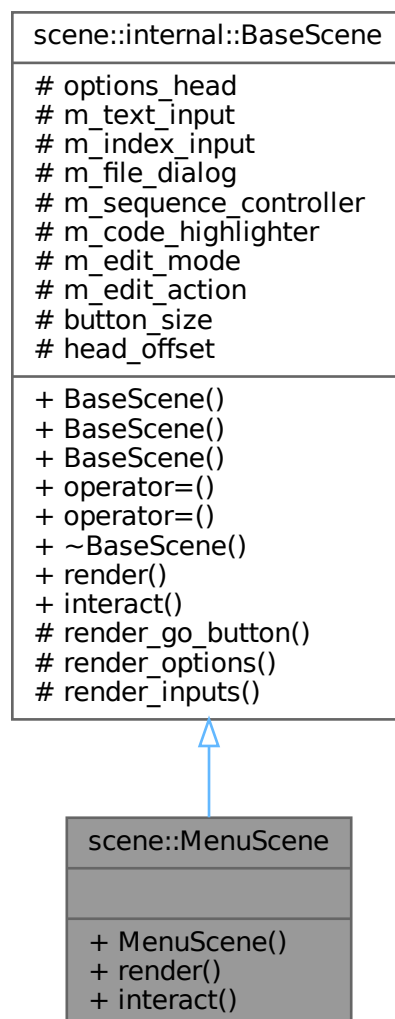
The documentation for this class was generated from the following files:

- [src/component/menu_item.hpp](#)
- [src/component/menu_item.cpp](#)

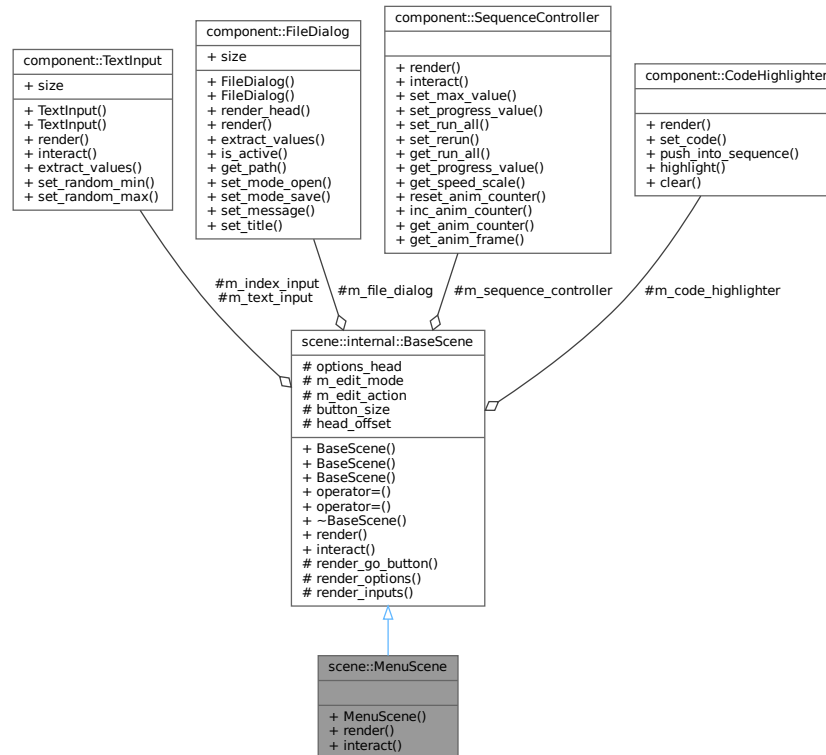
6.21 scene::MenuScene Class Reference

```
#include <menu_scene.hpp>
```

Inheritance diagram for scene::MenuScene:



Collaboration diagram for scene::MenuScene:



Public Member Functions

- [MenuScene](#) ()
- void [render](#) () override
- void [interact](#) () override

Public Member Functions inherited from [scene::internal::BaseScene](#)

- [BaseScene](#) ()=default
- [BaseScene](#) (const [BaseScene](#) &)=delete
- [BaseScene](#) ([BaseScene](#) &&)=delete
- [BaseScene](#) & [operator=](#) (const [BaseScene](#) &)=delete
- [BaseScene](#) & [operator=](#) ([BaseScene](#) &&)=delete
- virtual [~BaseScene](#) ()=default
- virtual void [render](#) ()
- virtual void [interact](#) ()

Additional Inherited Members

Protected Member Functions inherited from [scene::internal::BaseScene](#)

- virtual bool [render_go_button](#) () const
- virtual void [render_options](#) ([SceneOptions](#) &scene_config)
- virtual void [render_inputs](#) ()

Protected Attributes inherited from [scene::internal::BaseScene](#)

- float [options_head](#) {}
- [component::TextInput](#) [m_text_input](#) {"value"}
- [component::TextInput](#) [m_index_input](#) {"index"}
- [component::FileDialog](#) [m_file_dialog](#)
- [component::SequenceController](#) [m_sequence_controller](#)
- [component::CodeHighlighter](#) [m_code_highlighter](#)
- bool [m_edit_mode](#) {}
- bool [m_edit_action](#) {}

Static Protected Attributes inherited from [scene::internal::BaseScene](#)

- static constexpr Vector2 [button_size](#) {200, 50}
- static constexpr int [head_offset](#) = 20

6.21.1 Detailed Description

Definition at line 11 of file [menu_scene.hpp](#).

6.21.2 Constructor & Destructor Documentation**6.21.2.1 MenuScene()**

```
scene::MenuScene::MenuScene ( )
```

Definition at line 14 of file [menu_scene.cpp](#).

6.21.3 Member Function Documentation

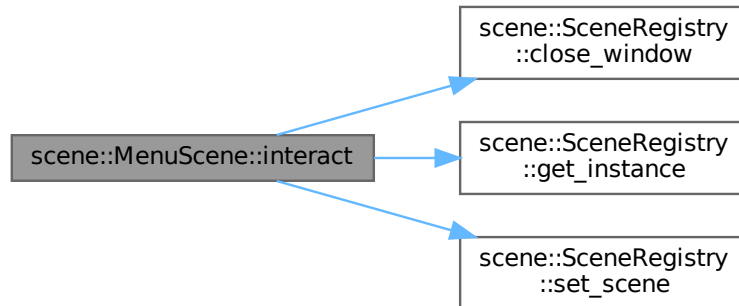
6.21.3.1 interact()

```
void scene::MenuScene::interact ( ) [override], [virtual]
```

Reimplemented from [scene::internal::BaseScene](#).

Definition at line 125 of file [menu_scene.cpp](#).

Here is the call graph for this function:



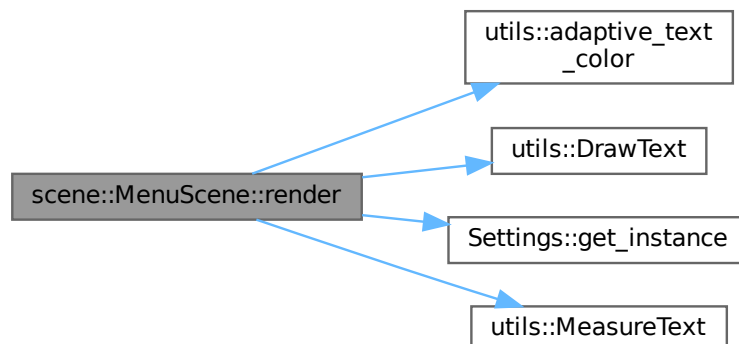
6.21.3.2 render()

```
void scene::MenuScene::render ( ) [override], [virtual]
```

Reimplemented from [scene::internal::BaseScene](#).

Definition at line 52 of file [menu_scene.cpp](#).

Here is the call graph for this function:



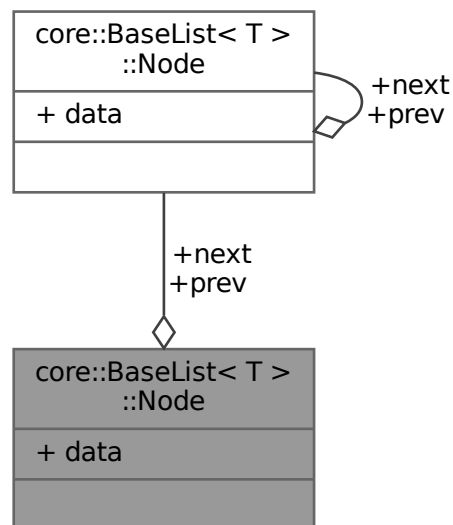
The documentation for this class was generated from the following files:

- [src/scene/menu_scene.hpp](#)
- [src/scene/menu_scene.cpp](#)

6.22 core::BaseList< T >::Node Struct Reference

```
#include <base_list.hpp>
```

Collaboration diagram for core::BaseList< T >::Node:



Public Attributes

- [T data](#) {}
- [Node_ptr prev](#) {}
- [Node_ptr next](#) {}

6.22.1 Detailed Description

```
template<typename T>
struct core::BaseList< T >::Node
```

Definition at line 16 of file [base_list.hpp](#).

6.22.2 Member Data Documentation

6.22.2.1 data

```
template<typename T >
T core::BaseList< T >::Node::data {}
```

Definition at line 17 of file [base_list.hpp](#).

6.22.2.2 next

```
template<typename T >
Node_ptr core::BaseList< T >::Node::next {}
```

Definition at line 19 of file [base_list.hpp](#).

6.22.2.3 prev

```
template<typename T >
Node_ptr core::BaseList< T >::Node::prev {}
```

Definition at line 18 of file [base_list.hpp](#).

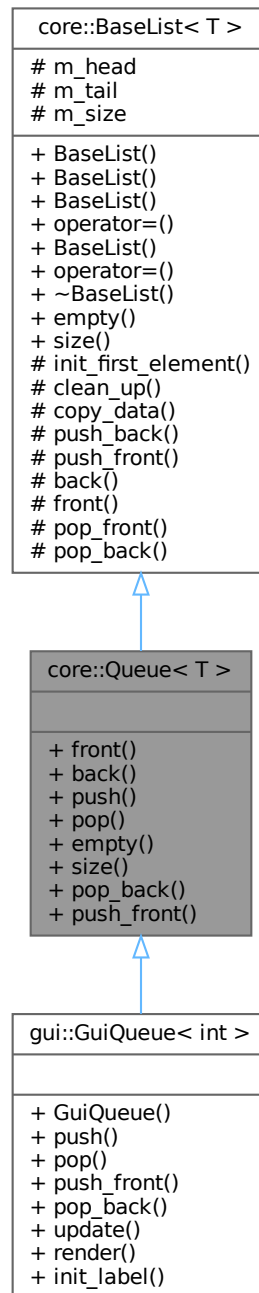
The documentation for this struct was generated from the following file:

- [src/core/base_list.hpp](#)

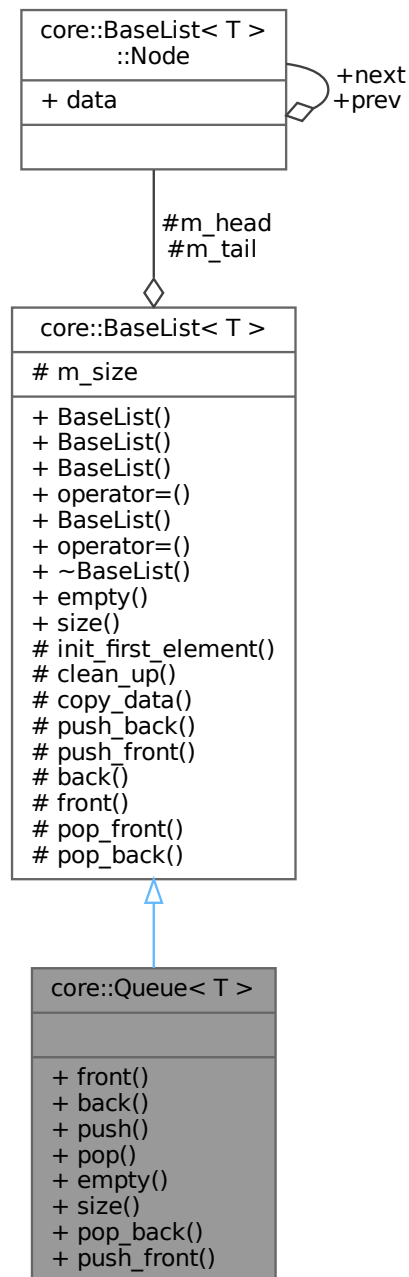
6.23 core::Queue< T > Class Template Reference

```
#include <queue.hpp>
```

Inheritance diagram for `core::Queue< T >`:



Collaboration diagram for core::Queue< T >:



Public Member Functions

- `T & front () const`
- `T & back () const`
- `void push (const T &elem)`
- `void pop ()`
- `bool empty () const`

- `std::size_t size () const`
- `void pop_back ()`
- `void push_front (const T &elem)`

Public Member Functions inherited from `core::BaseList< T >`

- `BaseList ()=default`
- `BaseList (std::initializer_list< T > init_list)`
- `BaseList (const BaseList &rhs)`
- `BaseList & operator= (const BaseList &rhs)`
- `BaseList (BaseList &&rhs) noexcept`
- `BaseList & operator= (BaseList &&rhs) noexcept`
- `~BaseList ()`
- `bool empty () const`
- `std::size_t size () const`

Additional Inherited Members

Protected Types inherited from `core::BaseList< T >`

- using `Node_ptr = Node *`

Protected Member Functions inherited from `core::BaseList< T >`

- `void init_first_element (const T &elem)`
- `void clean_up ()`
- `void copy_data (const BaseList &rhs)`
- `void push_back (const T &elem)`
- `void push_front (const T &elem)`
- `T & back () const`
- `T & front () const`
- `void pop_front ()`
- `void pop_back ()`

Protected Attributes inherited from `core::BaseList< T >`

- `Node_ptr m_head {nullptr}`
- `Node_ptr m_tail {nullptr}`
- `std::size_t m_size {}`

6.23.1 Detailed Description

```
template<typename T>
class core::Queue< T >
```

Definition at line 9 of file [queue.hpp](#).

6.23.2 Member Function Documentation

6.23.2.1 back()

```
template<typename T >  
T & core::Queue< T >::back
```

Definition at line 36 of file [queue.hpp](#).

6.23.2.2 empty()

```
template<typename T >  
bool core::BaseList< T >::empty
```

Definition at line 48 of file [base_list.hpp](#).

6.23.2.3 front()

```
template<typename T >  
T & core::Queue< T >::front
```

Definition at line 31 of file [queue.hpp](#).

6.23.2.4 pop()

```
template<typename T >  
void core::Queue< T >::pop
```

Definition at line 46 of file [queue.hpp](#).

6.23.2.5 pop_back()

```
template<typename T >  
void core::BaseList< T >::pop_back
```

Definition at line 37 of file [base_list.hpp](#).

6.23.2.6 push()

```
template<typename T >
void core::Queue< T >::push (
    const T & elem )
```

Definition at line 41 of file [queue.hpp](#).

6.23.2.7 push_front()

```
template<typename T >
void core::BaseList< T >::push_front (
    const T & elem )
```

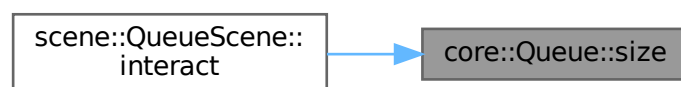
Definition at line 31 of file [base_list.hpp](#).

6.23.2.8 size()

```
template<typename T >
std::size_t core::BaseList< T >::size
```

Definition at line 49 of file [base_list.hpp](#).

Here is the caller graph for this function:



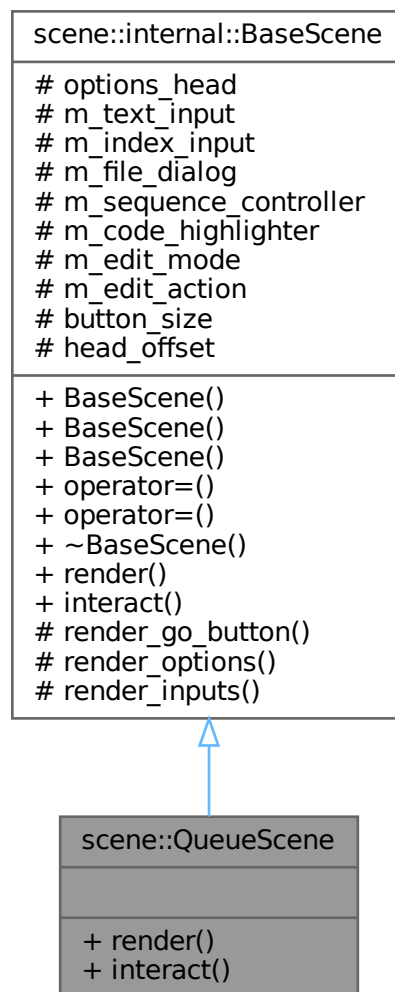
The documentation for this class was generated from the following file:

- [src/core/queue.hpp](#)

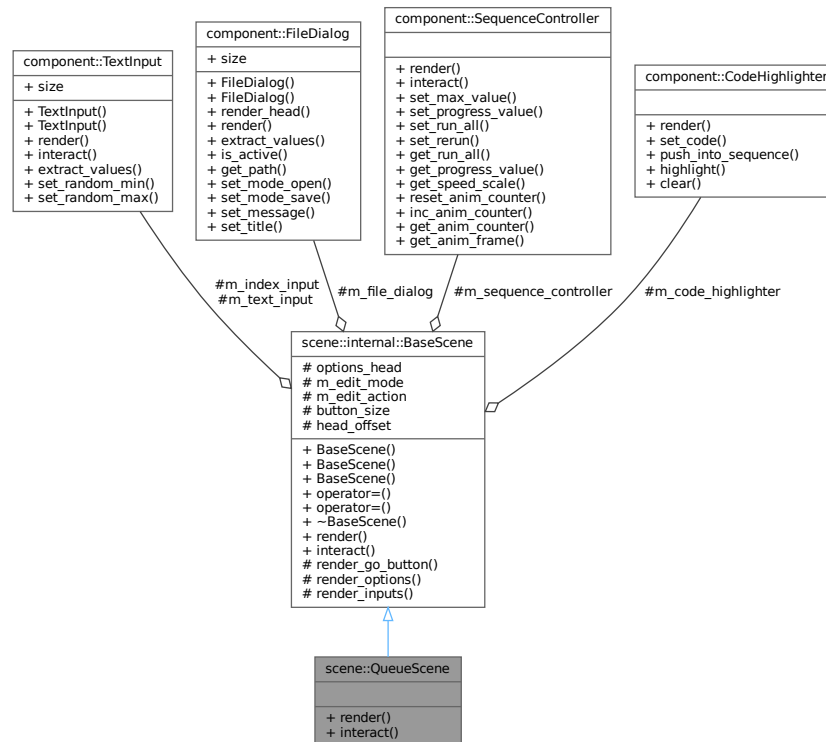
6.24 scene::QueueScene Class Reference

```
#include <queue_scene.hpp>
```

Inheritance diagram for scene::QueueScene:



Collaboration diagram for `scene::QueueScene`:



Public Member Functions

- void [render](#) () override
- void [interact](#) () override

Public Member Functions inherited from [scene::internal::BaseScene](#)

- [BaseScene](#) ()=default
- [BaseScene](#) (const [BaseScene](#) &)=delete
- [BaseScene](#) ([BaseScene](#) &&)=delete
- [BaseScene](#) & [operator=](#) (const [BaseScene](#) &)=delete
- [BaseScene](#) & [operator=](#) ([BaseScene](#) &&)=delete
- virtual [~BaseScene](#) ()=default
- virtual void [render](#) ()
- virtual void [interact](#) ()

Additional Inherited Members

Protected Member Functions inherited from [scene::internal::BaseScene](#)

- virtual bool [render_go_button](#) () const
- virtual void [render_options](#) ([SceneOptions](#) &scene_config)
- virtual void [render_inputs](#) ()

Protected Attributes inherited from [scene::internal::BaseScene](#)

- float [options_head](#) {}
- [component::TextInput](#) [m_text_input](#) {"value"}
- [component::TextInput](#) [m_index_input](#) {"index"}
- [component::FileDialog](#) [m_file_dialog](#)
- [component::SequenceController](#) [m_sequence_controller](#)
- [component::CodeHighlighter](#) [m_code_highlighter](#)
- bool [m_edit_mode](#) {}
- bool [m_edit_action](#) {}

Static Protected Attributes inherited from [scene::internal::BaseScene](#)

- static constexpr [Vector2](#) [button_size](#) {200, 50}
- static constexpr int [head_offset](#) = 20

6.24.1 Detailed Description

Definition at line 16 of file [queue_scene.hpp](#).

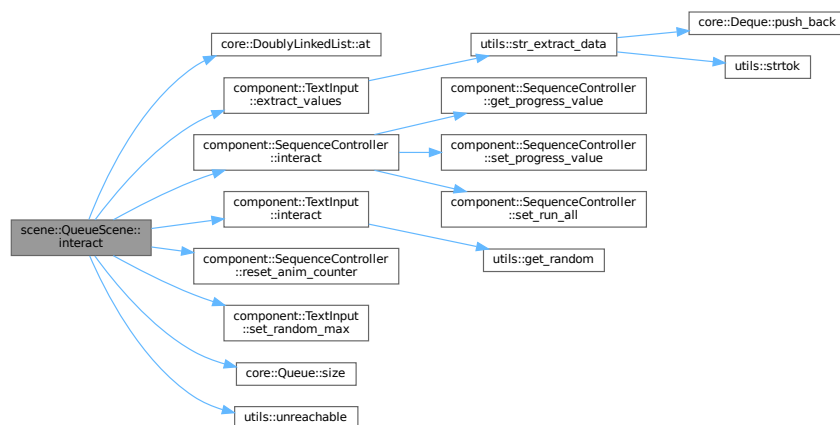
6.24.2 Member Function Documentation**6.24.2.1 [interact\(\)](#)**

```
void scene::QueueScene::interact ( ) [override], [virtual]
```

Reimplemented from [scene::internal::BaseScene](#).

Definition at line 71 of file [queue_scene.cpp](#).

Here is the call graph for this function:



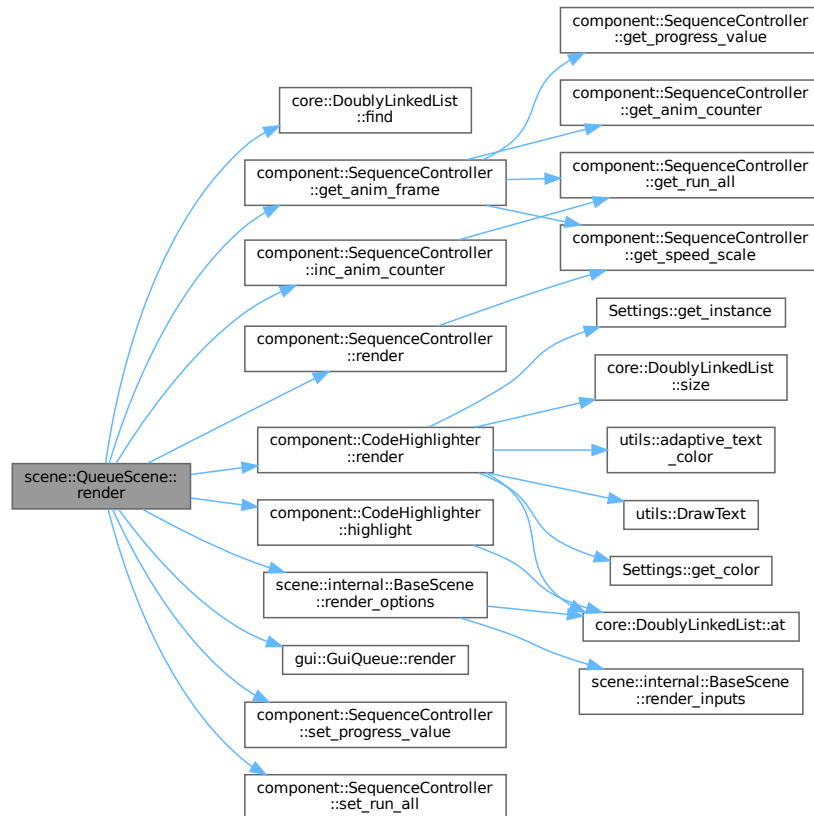
6.24.2.2 render()

```
void scene::QueueScene::render ( ) [override], [virtual]
```

Reimplemented from [scene::internal::BaseScene](#).

Definition at line 51 of file [queue_scene.cpp](#).

Here is the call graph for this function:



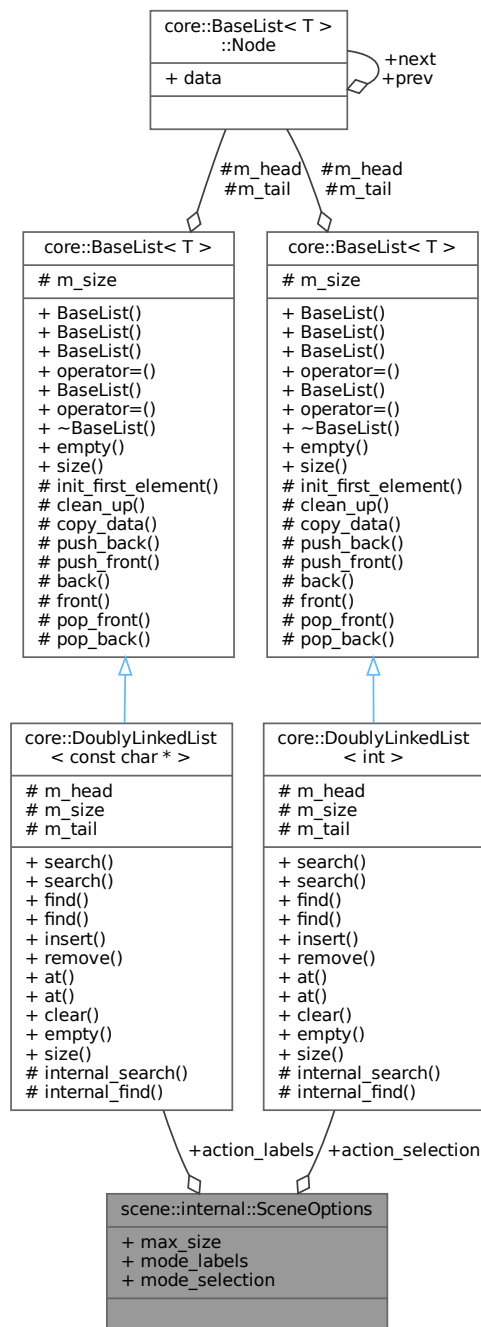
The documentation for this class was generated from the following files:

- [src/scene/queue_scene.hpp](#)
- [src/scene/queue_scene.cpp](#)

6.25 scene::internal::SceneOptions Struct Reference

```
#include <scene_options.hpp>
```

Collaboration diagram for scene::internal::SceneOptions:



Public Attributes

- `const std::size_t max_size {}`
- `const char * mode_labels {}`
- `int mode_selection {}`
- `core::DoublyLinkedList< const char * > action_labels`
- `core::DoublyLinkedList< int > action_selection`

6.25.1 Detailed Description

Definition at line 10 of file [scene_options.hpp](#).

6.25.2 Member Data Documentation

6.25.2.1 action_labels

```
core::DoublyLinkedList<const char*> scene::internal::SceneOptions::action_labels
```

Definition at line 14 of file [scene_options.hpp](#).

6.25.2.2 action_selection

```
core::DoublyLinkedList<int> scene::internal::SceneOptions::action_selection
```

Definition at line 15 of file [scene_options.hpp](#).

6.25.2.3 max_size

```
const std::size_t scene::internal::SceneOptions::max_size {}
```

Definition at line 11 of file [scene_options.hpp](#).

6.25.2.4 mode_labels

```
const char* scene::internal::SceneOptions::mode_labels {}
```

Definition at line 12 of file [scene_options.hpp](#).

6.25.2.5 mode_selection

```
int scene::internal::SceneOptions::mode_selection {}
```

Definition at line 13 of file [scene_options.hpp](#).

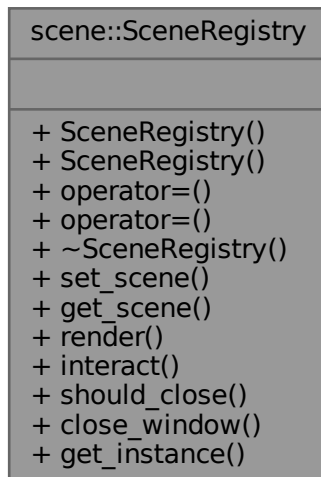
The documentation for this struct was generated from the following file:

- [src/scene/scene_options.hpp](#)

6.26 scene::SceneRegistry Class Reference

```
#include <scene_registry.hpp>
```

Collaboration diagram for scene::SceneRegistry:



Public Member Functions

- [SceneRegistry](#) (const [SceneRegistry](#) &)=delete
- [SceneRegistry](#) ([SceneRegistry](#) &&)=delete
- [SceneRegistry](#) & [operator=](#) (const [SceneRegistry](#) &)=delete
- [SceneRegistry](#) & [operator=](#) ([SceneRegistry](#) &&)=delete
- [~SceneRegistry](#) ()=default
- void [set_scene](#) (int scene_type)
- int [get_scene](#) () const
- void [render](#) ()
- void [interact](#) ()
- bool [should_close](#) () const
- void [close_window](#) ()

Static Public Member Functions

- static [SceneRegistry](#) & [get_instance](#) ()

6.26.1 Detailed Description

Definition at line 30 of file [scene_registry.hpp](#).

6.26.2 Constructor & Destructor Documentation

6.26.2.1 SceneRegistry() [1/2]

```
scene::SceneRegistry::SceneRegistry (
    const SceneRegistry & ) [delete]
```

6.26.2.2 SceneRegistry() [2/2]

```
scene::SceneRegistry::SceneRegistry (
    SceneRegistry && ) [delete]
```

6.26.2.3 ~SceneRegistry()

```
scene::SceneRegistry::~~SceneRegistry ( ) [default]
```

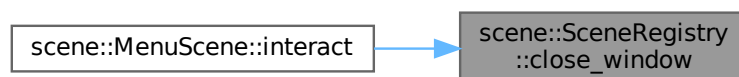
6.26.3 Member Function Documentation

6.26.3.1 close_window()

```
void scene::SceneRegistry::close_window ( )
```

Definition at line 25 of file [scene_registry.cpp](#).

Here is the caller graph for this function:

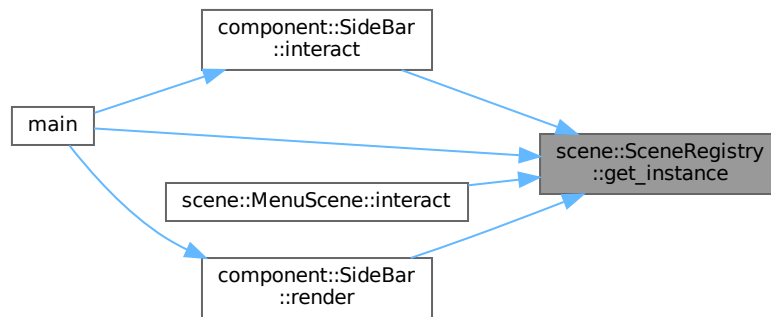


6.26.3.2 get_instance()

`SceneRegistry & scene::SceneRegistry::get_instance () [static]`

Definition at line 7 of file [scene_registry.cpp](#).

Here is the caller graph for this function:

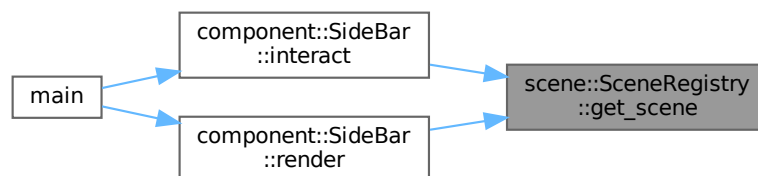


6.26.3.3 get_scene()

`int scene::SceneRegistry::get_scene () const`

Definition at line 17 of file [scene_registry.cpp](#).

Here is the caller graph for this function:

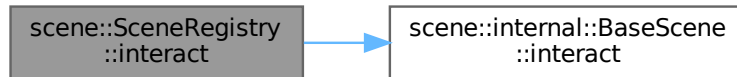


6.26.3.4 interact()

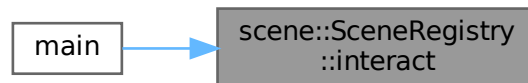
```
void scene::SceneRegistry::interact ( )
```

Definition at line 21 of file [scene_registry.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.26.3.5 operator=() [1/2]

```
SceneRegistry & scene::SceneRegistry::operator= (
    const SceneRegistry & ) [delete]
```

6.26.3.6 operator=() [2/2]

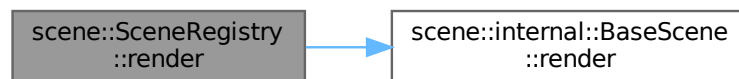
```
SceneRegistry & scene::SceneRegistry::operator= (
    SceneRegistry && ) [delete]
```

6.26.3.7 render()

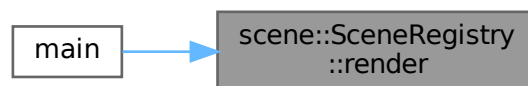
```
void scene::SceneRegistry::render ( )
```

Definition at line 19 of file [scene_registry.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

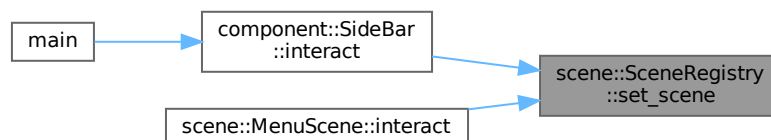


6.26.3.8 set_scene()

```
void scene::SceneRegistry::set_scene (
    int scene_type )
```

Definition at line 12 of file [scene_registry.cpp](#).

Here is the caller graph for this function:

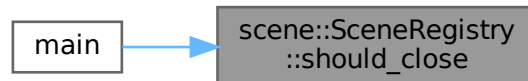


6.26.3.9 should_close()

```
bool scene::SceneRegistry::should_close ( ) const
```

Definition at line 23 of file [scene_registry.cpp](#).

Here is the caller graph for this function:



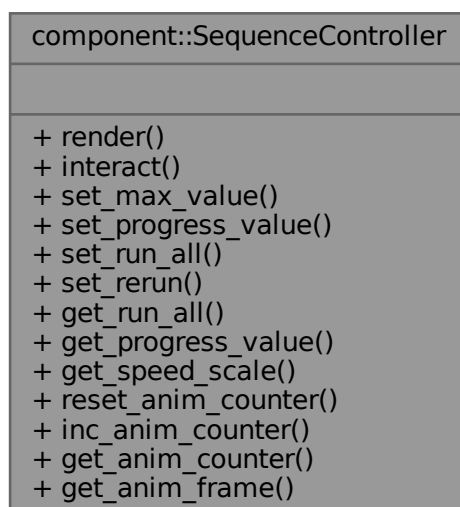
The documentation for this class was generated from the following files:

- [src/scene/scene_registry.hpp](#)
- [src/scene/scene_registry.cpp](#)

6.27 component::SequenceController Class Reference

```
#include <sequence_controller.hpp>
```

Collaboration diagram for component::SequenceController:



Public Member Functions

- void [render](#) ()
- bool [interact](#) ()
- void [set_max_value](#) (int num)
- void [set_progress_value](#) (int value)
- void [set_run_all](#) (bool run_all)
- void [set_rerun](#) ()
- bool [get_run_all](#) () const
- int [get_progress_value](#) () const
- float [get_speed_scale](#) () const
- void [reset_anim_counter](#) ()
- void [inc_anim_counter](#) ()
- int [get_anim_counter](#) () const
- int [get_anim_frame](#) () const

6.27.1 Detailed Description

Definition at line 8 of file [sequence_controller.hpp](#).

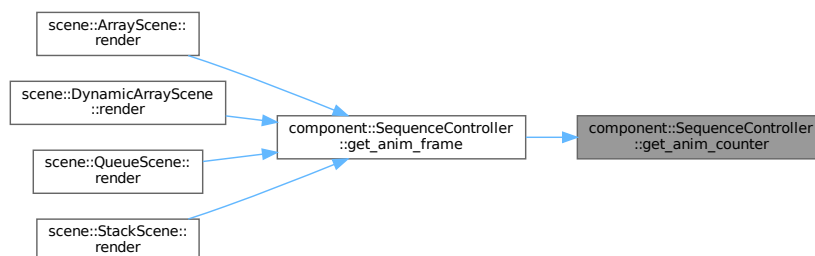
6.27.2 Member Function Documentation

6.27.2.1 [get_anim_counter\(\)](#)

```
int component::SequenceController::get_anim_counter ( ) const
```

Definition at line 35 of file [sequence_controller.cpp](#).

Here is the caller graph for this function:

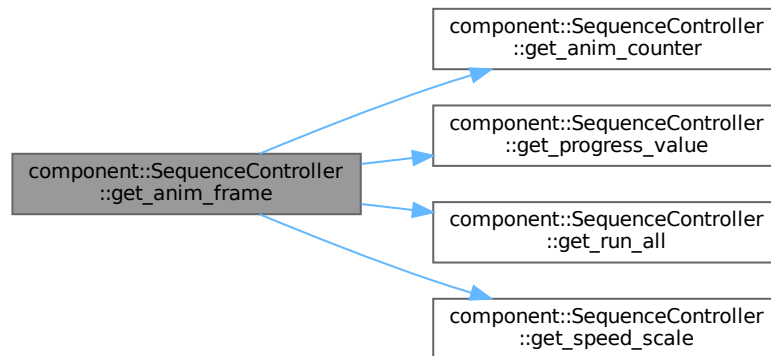


6.27.2.2 get_anim_frame()

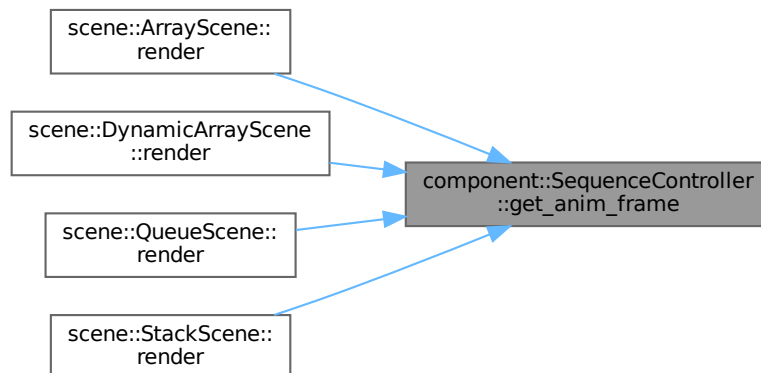
```
int component::SequenceController::get_anim_frame ( ) const
```

Definition at line 42 of file [sequence_controller.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

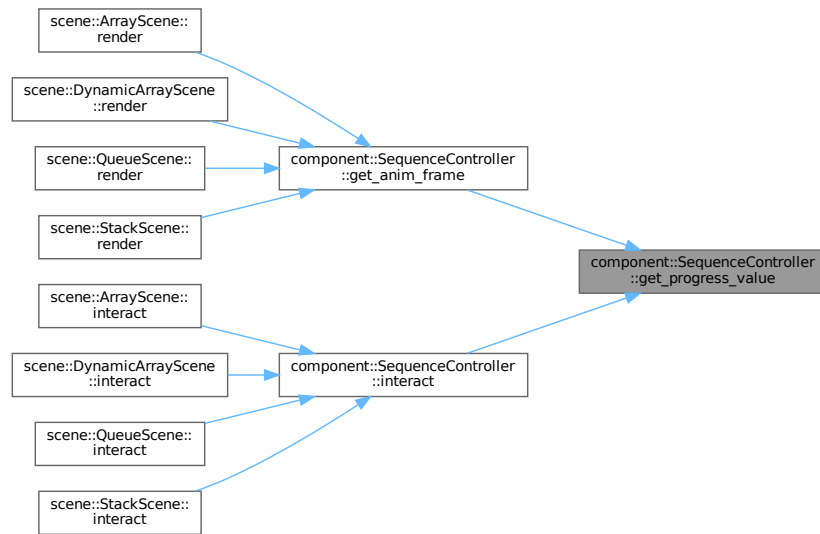


6.27.2.3 get_progress_value()

```
int component::SequenceController::get_progress_value ( ) const
```

Definition at line 21 of file [sequence_controller.cpp](#).

Here is the caller graph for this function:

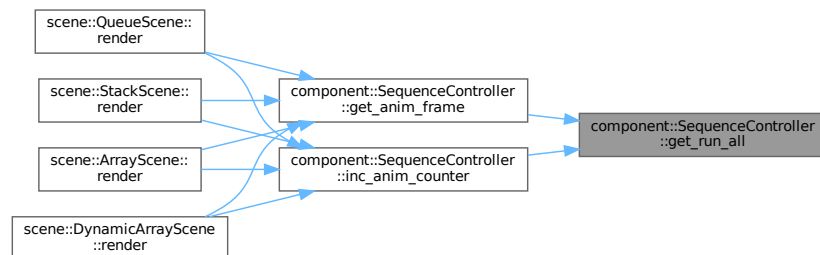


6.27.2.4 get_run_all()

```
bool component::SequenceController::get_run_all ( ) const
```

Definition at line 19 of file [sequence_controller.cpp](#).

Here is the caller graph for this function:

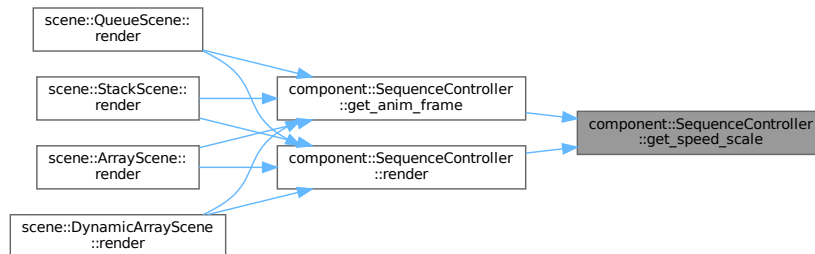


6.27.2.5 get_speed_scale()

```
float component::SequenceController::get_speed_scale ( ) const
```

Definition at line 23 of file [sequence_controller.cpp](#).

Here is the caller graph for this function:



6.27.2.6 inc_anim_counter()

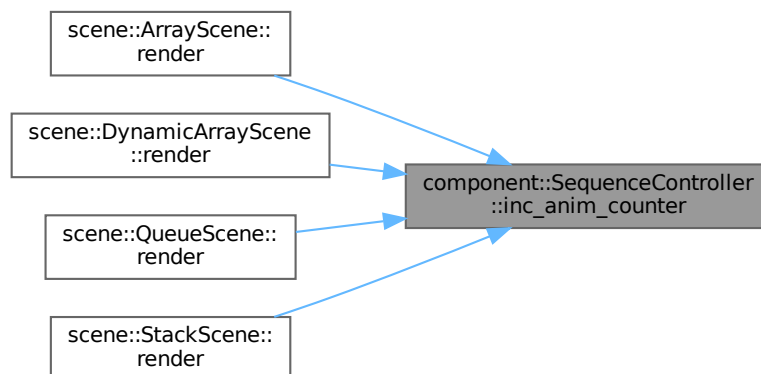
```
void component::SequenceController::inc_anim_counter ( )
```

Definition at line 29 of file [sequence_controller.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

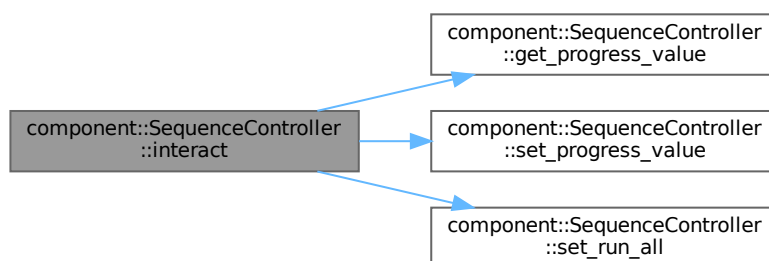


6.27.2.7 interact()

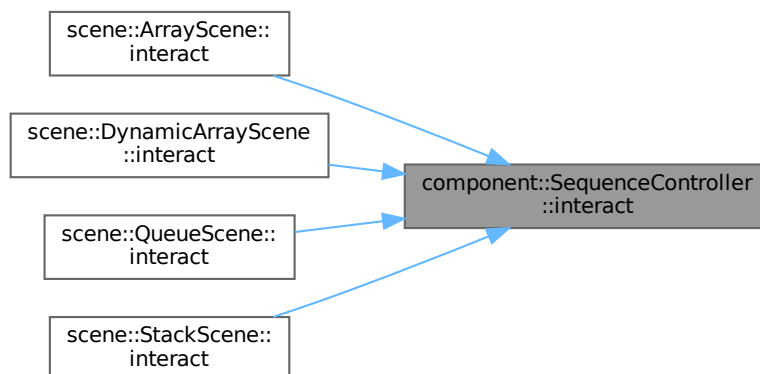
```
bool component::SequenceController::interact ( )
```

Definition at line 90 of file [sequence_controller.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.27.2.8 render()

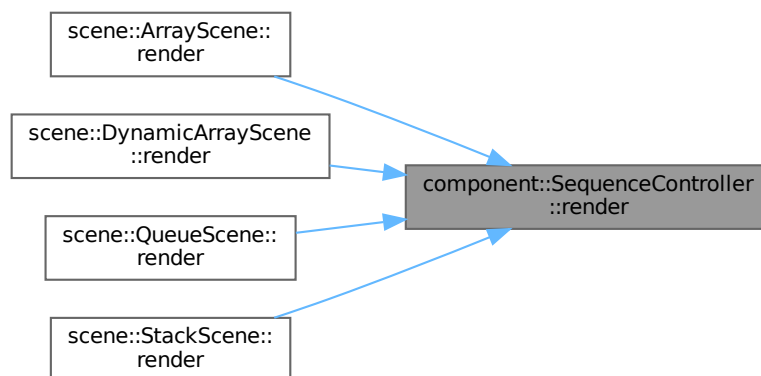
```
void component::SequenceController::render ( )
```

Definition at line 51 of file [sequence_controller.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

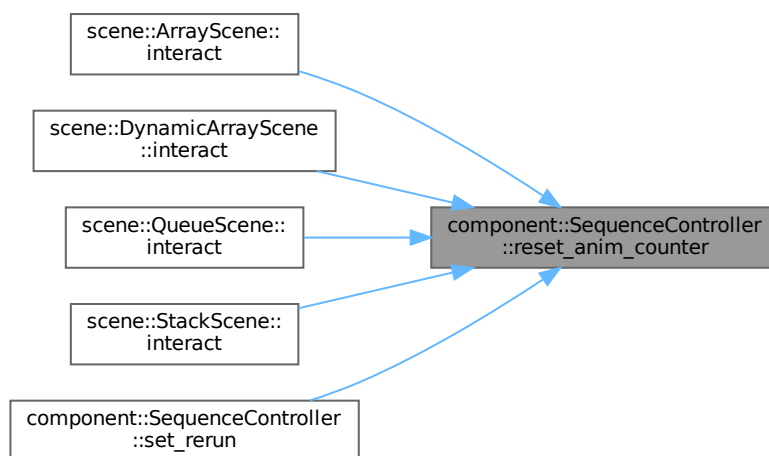


6.27.2.9 reset_anim_counter()

```
void component::SequenceController::reset_anim_counter ( )
```

Definition at line 27 of file [sequence_controller.cpp](#).

Here is the caller graph for this function:



6.27.2.10 set_max_value()

```
void component::SequenceController::set_max_value (
    int num )
```

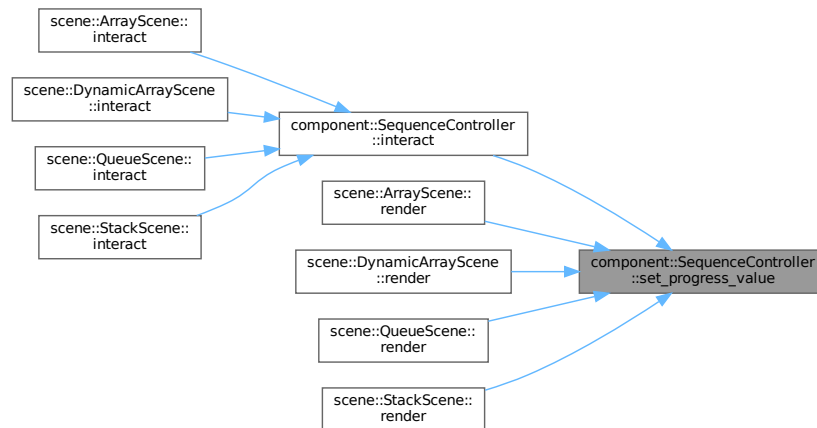
Definition at line 11 of file [sequence_controller.cpp](#).

6.27.2.11 set_progress_value()

```
void component::SequenceController::set_progress_value (
    int value )
```

Definition at line 13 of file [sequence_controller.cpp](#).

Here is the caller graph for this function:

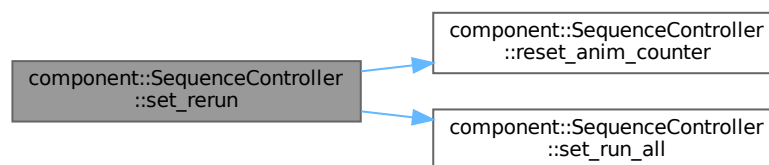


6.27.2.12 set_rerun()

```
void component::SequenceController::set_rerun ( )
```

Definition at line 37 of file [sequence_controller.cpp](#).

Here is the call graph for this function:

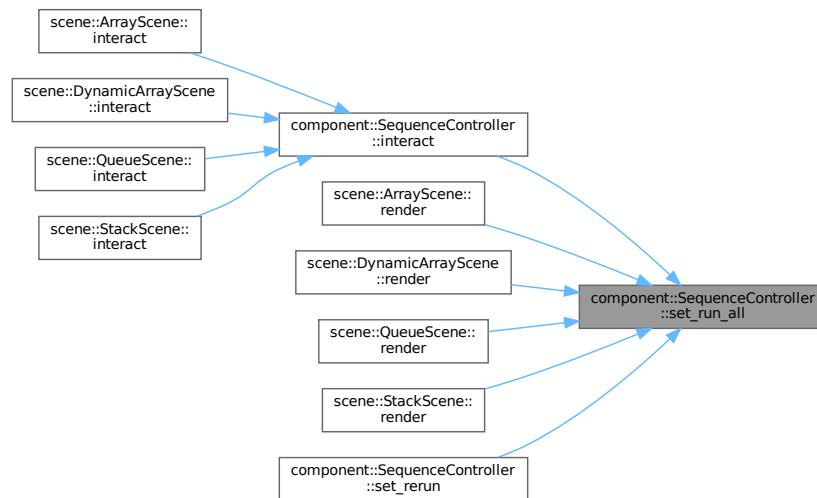


6.27.2.13 set_run_all()

```
void component::SequenceController::set_run_all (
    bool run_all )
```

Definition at line 17 of file [sequence_controller.cpp](#).

Here is the caller graph for this function:



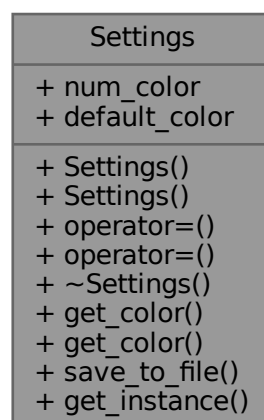
The documentation for this class was generated from the following files:

- [src/component/sequence_controller.hpp](#)
- [src/component/sequence_controller.cpp](#)

6.28 Settings Class Reference

```
#include <settings.hpp>
```

Collaboration diagram for Settings:



Public Member Functions

- [Settings](#) (const [Settings](#) &)=delete
- [Settings](#) ([Settings](#) &&)=delete
- [Settings](#) & [operator=](#) (const [Settings](#) &)=delete
- [Settings](#) & [operator=](#) ([Settings](#) &&)=delete
- [~Settings](#) ()
- Color & [get_color](#) (std::size_t index)
- Color [get_color](#) (std::size_t index) const
- void [save_to_file](#) (const std::string &path)

Static Public Member Functions

- static [Settings](#) & [get_instance](#) ()

Static Public Attributes

- static constexpr int [num_color](#) = 9
- static constexpr std::array< unsigned, [num_color](#) > [default_color](#)

6.28.1 Detailed Description

Definition at line 10 of file [settings.hpp](#).

6.28.2 Constructor & Destructor Documentation

6.28.2.1 Settings() [1/2]

```
Settings::Settings (  
    const Settings & ) [delete]
```

6.28.2.2 Settings() [2/2]

```
Settings::Settings (  
    Settings && ) [delete]
```


6.28.2.3 ~Settings()

```
Settings::~~Settings ( )
```

Definition at line 24 of file [settings.cpp](#).

Here is the call graph for this function:



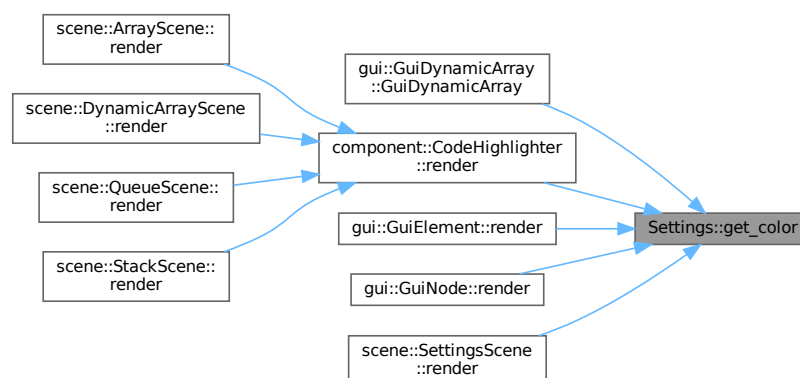
6.28.3 Member Function Documentation

6.28.3.1 get_color() [1/2]

```
Color & Settings::get_color (
    std::size_t index )
```

Definition at line 26 of file [settings.cpp](#).

Here is the caller graph for this function:



6.28.3.2 `get_color()` [2/2]

```
Color Settings::get_color (
    std::size_t index ) const
```

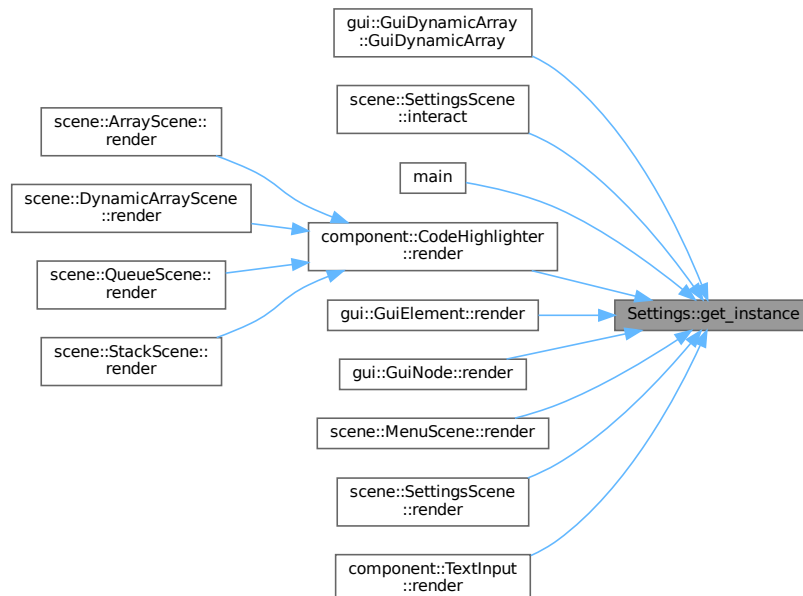
Definition at line 28 of file [settings.cpp](#).

6.28.3.3 `get_instance()`

```
Settings & Settings::get_instance ( ) [static]
```

Definition at line 10 of file [settings.cpp](#).

Here is the caller graph for this function:



6.28.3.4 `operator=()` [1/2]

```
Settings & Settings::operator= (
    const Settings & ) [delete]
```

6.28.3.5 operator=() [2/2]

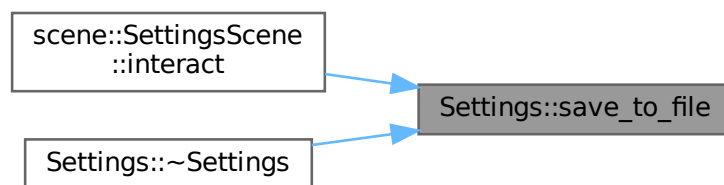
```
Settings & Settings::operator= (
    Settings && ) [delete]
```

6.28.3.6 save_to_file()

```
void Settings::save_to_file (
    const std::string & path )
```

Definition at line 15 of file [settings.cpp](#).

Here is the caller graph for this function:



6.28.4 Member Data Documentation

6.28.4.1 default_color

```
constexpr std::array<unsigned, num_color> Settings::default_color [static], [constexpr]
```

Initial value:

```
{{
    0x00000000,
    0x82828200,
    0xfffa1000,
    0x00e43000,
    0x873cbe00,
    0xe6293700,
    0x0079f100,
    0xff6dc200,
    0xf5f5f500,
}}
```

Definition at line 13 of file [settings.hpp](#).

6.28.4.2 num_color

```
constexpr int Settings::num_color = 9 [static], [constexpr]
```

Definition at line 12 of file [settings.hpp](#).

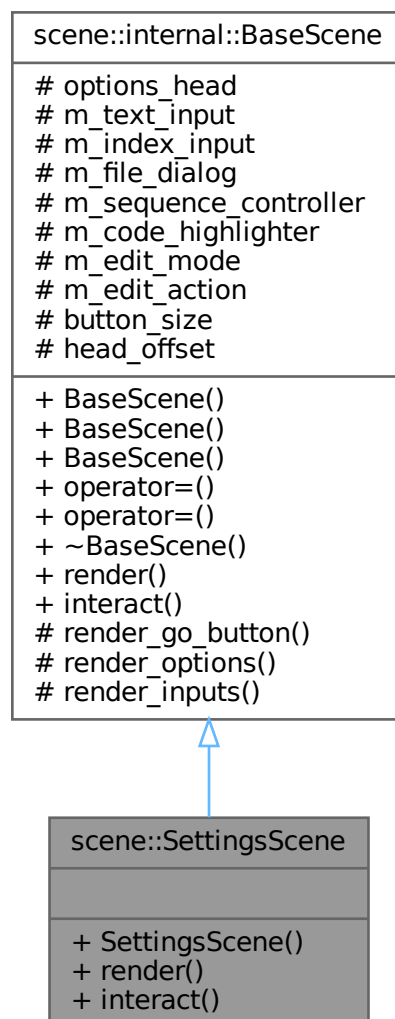
The documentation for this class was generated from the following files:

- [src/settings.hpp](#)
- [src/settings.cpp](#)

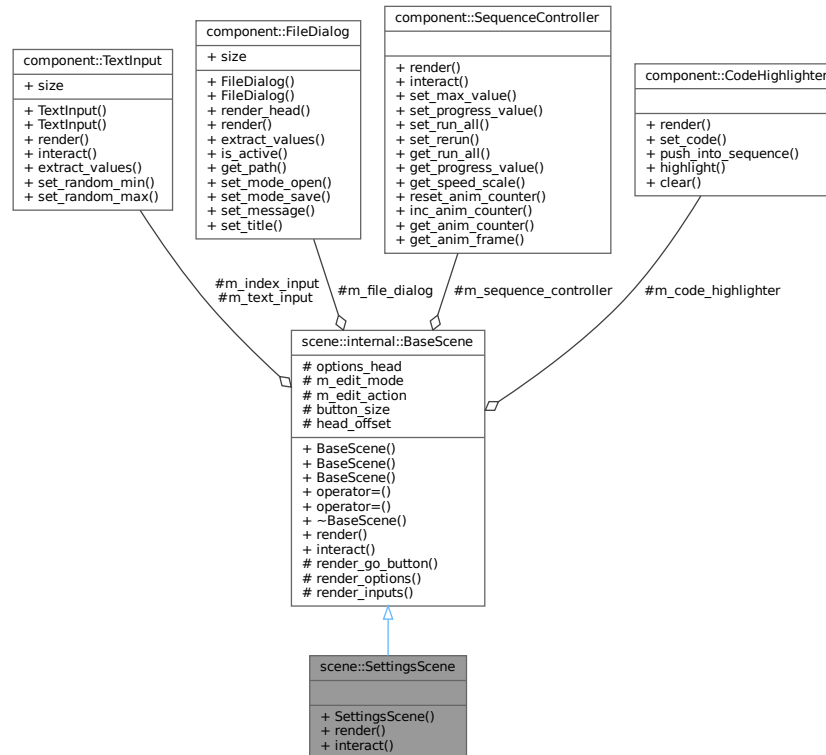
6.29 scene::SettingsScene Class Reference

```
#include <settings_scene.hpp>
```

Inheritance diagram for scene::SettingsScene:



Collaboration diagram for scene::SettingsScene:



Public Member Functions

- [SettingsScene](#) ()
- void [render](#) () override
- void [interact](#) () override

Public Member Functions inherited from [scene::internal::BaseScene](#)

- [BaseScene](#) ()=default
- [BaseScene](#) (const [BaseScene](#) &)=delete
- [BaseScene](#) ([BaseScene](#) &&)=delete
- [BaseScene](#) & [operator=](#) (const [BaseScene](#) &)=delete
- [BaseScene](#) & [operator=](#) ([BaseScene](#) &&)=delete
- virtual [~BaseScene](#) ()=default
- virtual void [render](#) ()
- virtual void [interact](#) ()

Additional Inherited Members

Protected Member Functions inherited from [scene::internal::BaseScene](#)

- virtual bool [render_go_button](#) () const
- virtual void [render_options](#) ([SceneOptions](#) &scene_config)
- virtual void [render_inputs](#) ()

Protected Attributes inherited from [scene::internal::BaseScene](#)

- float [options_head](#) {}
- [component::TextInput](#) [m_text_input](#) {"value"}
- [component::TextInput](#) [m_index_input](#) {"index"}
- [component::FileDialog](#) [m_file_dialog](#)
- [component::SequenceController](#) [m_sequence_controller](#)
- [component::CodeHighlighter](#) [m_code_highlighter](#)
- bool [m_edit_mode](#) {}
- bool [m_edit_action](#) {}

Static Protected Attributes inherited from [scene::internal::BaseScene](#)

- static constexpr Vector2 [button_size](#) {200, 50}
- static constexpr int [head_offset](#) = 20

6.29.1 Detailed Description

Definition at line 15 of file [settings_scene.hpp](#).

6.29.2 Constructor & Destructor Documentation

6.29.2.1 SettingsScene()

```
scene::SettingsScene::SettingsScene ( )
```

Definition at line 47 of file [settings_scene.cpp](#).

6.29.3 Member Function Documentation

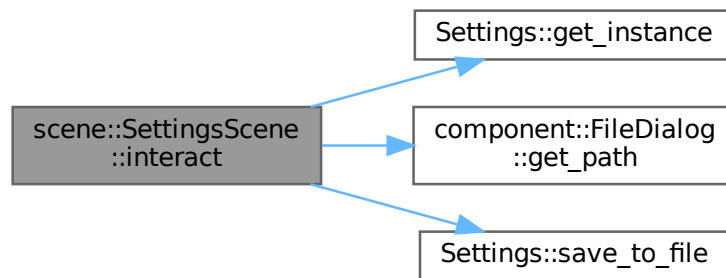
6.29.3.1 interact()

```
void scene::SettingsScene::interact ( ) [override], [virtual]
```

Reimplemented from [scene::internal::BaseScene](#).

Definition at line 145 of file [settings_scene.cpp](#).

Here is the call graph for this function:



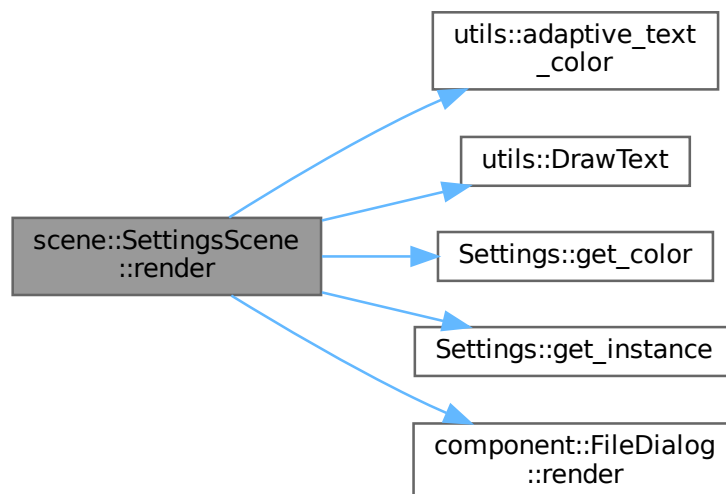
6.29.3.2 render()

```
void scene::SettingsScene::render ( ) [override], [virtual]
```

Reimplemented from [scene::internal::BaseScene](#).

Definition at line 70 of file [settings_scene.cpp](#).

Here is the call graph for this function:



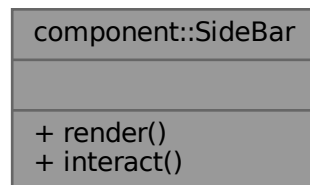
The documentation for this class was generated from the following files:

- [src/scene/settings_scene.hpp](#)
- [src/scene/settings_scene.cpp](#)

6.30 component::SideBar Class Reference

```
#include <sidebar.hpp>
```

Collaboration diagram for component::SideBar:



Public Member Functions

- void [render](#) ()
- void [interact](#) ()

6.30.1 Detailed Description

Definition at line [11](#) of file [sidebar.hpp](#).

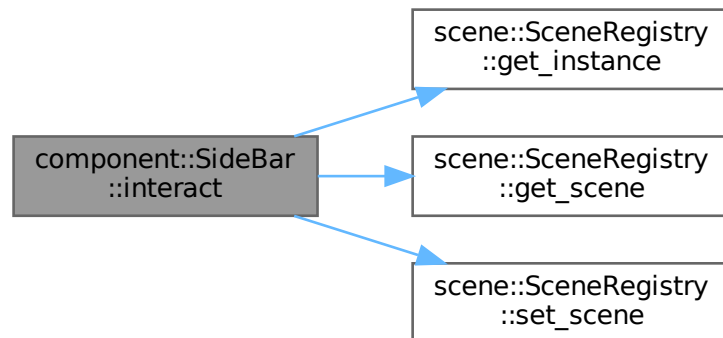
6.30.2 Member Function Documentation

6.30.2.1 interact()

```
void component::SideBar::interact ( )
```

Definition at line 48 of file [sidebar.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

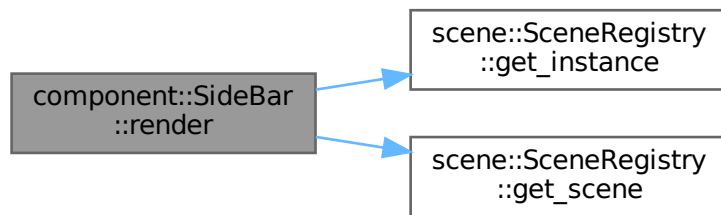


6.30.2.2 render()

```
void component::SideBar::render ( )
```

Definition at line 11 of file [sidebar.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



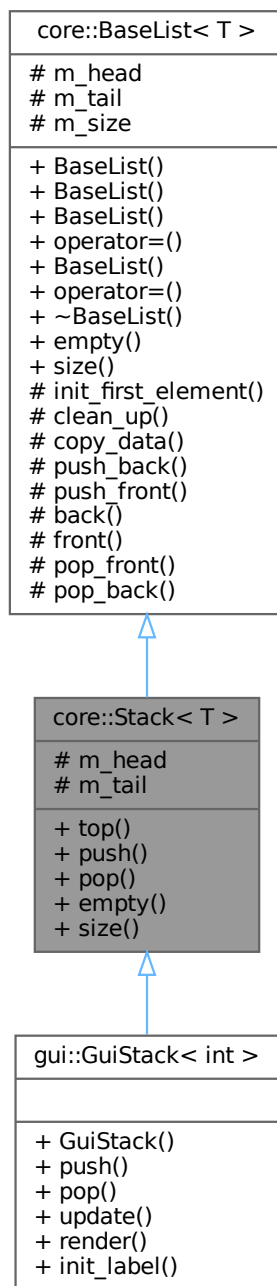
The documentation for this class was generated from the following files:

- [src/component/sidebar.hpp](#)
- [src/component/sidebar.cpp](#)

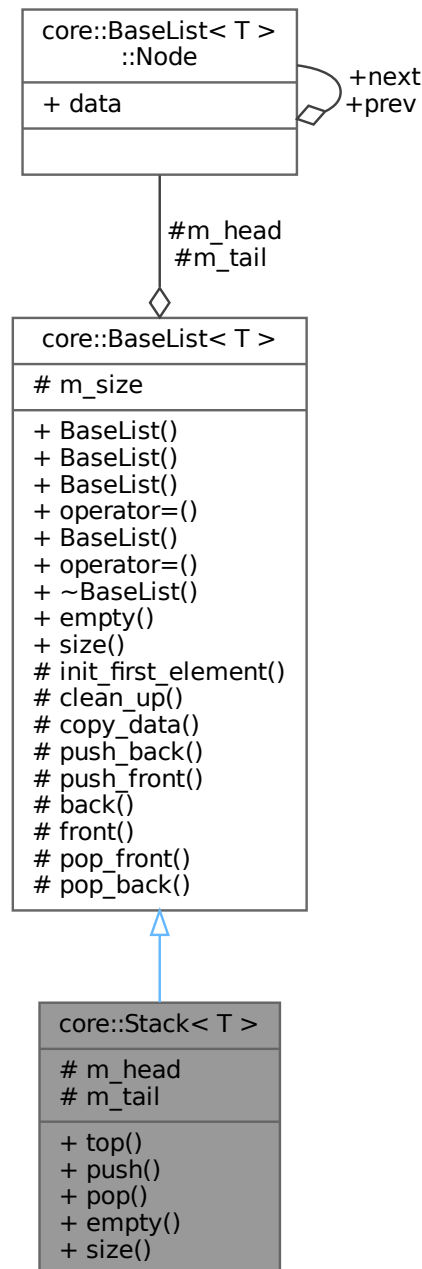
6.31 `core::Stack< T >` Class Template Reference

```
#include <stack.hpp>
```

Inheritance diagram for core::Stack< T >:



Collaboration diagram for `core::Stack< T >`:



Public Member Functions

- `T & top () const`
- `void push (const T &elem)`
- `void pop ()`
- `bool empty () const`
- `std::size_t size () const`

Public Member Functions inherited from [core::BaseList< T >](#)

- [BaseList](#) ()=default
- [BaseList](#) (std::initializer_list< T > init_list)
- [BaseList](#) (const [BaseList](#) &rhs)
- [BaseList](#) & [operator=](#) (const [BaseList](#) &rhs)
- [BaseList](#) ([BaseList](#) &&rhs) noexcept
- [BaseList](#) & [operator=](#) ([BaseList](#) &&rhs) noexcept
- [~BaseList](#) ()
- bool [empty](#) () const
- std::size_t [size](#) () const

Protected Types

- using [Base](#) = [BaseList](#)< T >

Protected Types inherited from [core::BaseList< T >](#)

- using [Node_ptr](#) = [Node](#) *

Protected Attributes

- [Node_ptr](#) m_head
- [Node_ptr](#) m_tail

Protected Attributes inherited from [core::BaseList< T >](#)

- [Node_ptr](#) m_head {nullptr}
- [Node_ptr](#) m_tail {nullptr}
- std::size_t m_size {}

Additional Inherited Members**Protected Member Functions inherited from [core::BaseList< T >](#)**

- void [init_first_element](#) (const T &elem)
- void [clean_up](#) ()
- void [copy_data](#) (const [BaseList](#) &rhs)
- void [push_back](#) (const T &elem)
- void [push_front](#) (const T &elem)
- T & [back](#) () const
- T & [front](#) () const
- void [pop_front](#) ()
- void [pop_back](#) ()

6.31.1 Detailed Description

```
template<typename T>
class core::Stack< T >
```

Definition at line 9 of file [stack.hpp](#).

6.31.2 Member Typedef Documentation

6.31.2.1 Base

```
template<typename T >
using core::Stack< T >::Base = BaseList<T> [protected]
```

Definition at line 11 of file [stack.hpp](#).

6.31.3 Member Function Documentation

6.31.3.1 empty()

```
template<typename T >
bool core::BaseList< T >::empty
```

Definition at line 48 of file [base_list.hpp](#).

6.31.3.2 pop()

```
template<typename T >
void core::Stack< T >::pop
```

Definition at line 38 of file [stack.hpp](#).

6.31.3.3 push()

```
template<typename T >
void core::Stack< T >::push (
    const T & elem )
```

Definition at line 33 of file [stack.hpp](#).

6.31.3.4 size()

```
template<typename T >
std::size_t core::BaseList< T >::size
```

Definition at line 49 of file [base_list.hpp](#).

Here is the caller graph for this function:



6.31.3.5 top()

```
template<typename T >
T & core::Stack< T >::top
```

Definition at line 28 of file [stack.hpp](#).

6.31.4 Member Data Documentation

6.31.4.1 m_head

```
template<typename T >
Node_ptr core::BaseList< T >::m_head [protected]
```

Definition at line 22 of file [base_list.hpp](#).

6.31.4.2 m_tail

```
template<typename T >
Node_ptr core::BaseList< T >::m_tail [protected]
```

Definition at line 23 of file [base_list.hpp](#).

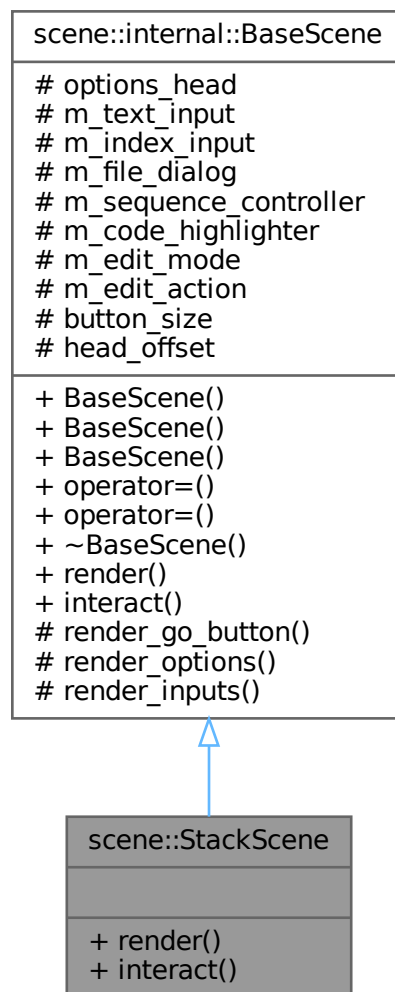
The documentation for this class was generated from the following file:

- [src/core/stack.hpp](#)

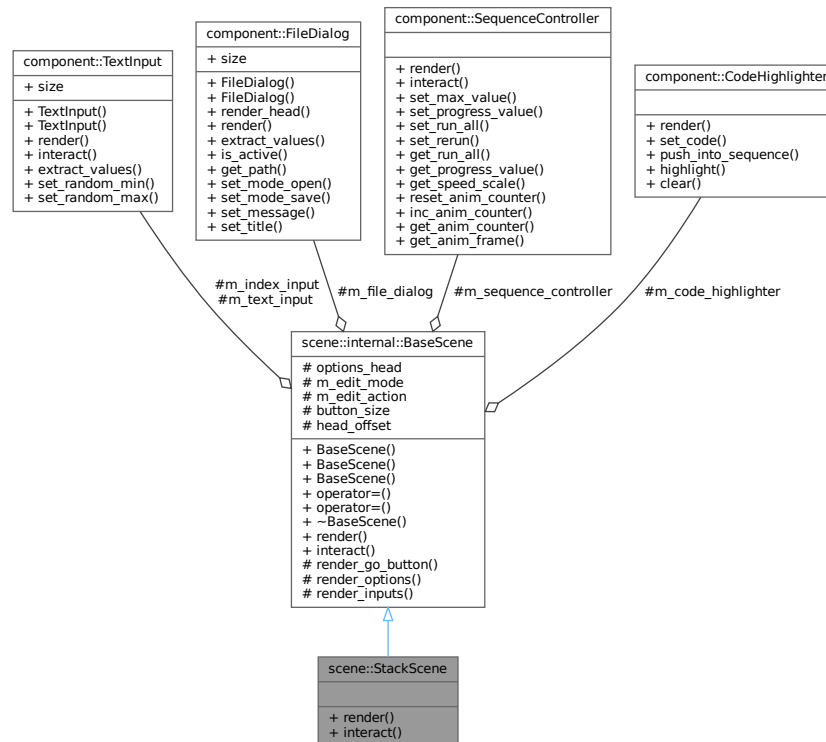
6.32 scene::StackScene Class Reference

```
#include <stack_scene.hpp>
```

Inheritance diagram for scene::StackScene:



Collaboration diagram for scene::StackScene:



Public Member Functions

- void [render](#) () override
- void [interact](#) () override

Public Member Functions inherited from [scene::internal::BaseScene](#)

- [BaseScene](#) ()=default
- [BaseScene](#) (const [BaseScene](#) &)=delete
- [BaseScene](#) ([BaseScene](#) &&)=delete
- [BaseScene](#) & [operator=](#) (const [BaseScene](#) &)=delete
- [BaseScene](#) & [operator=](#) ([BaseScene](#) &&)=delete
- virtual [~BaseScene](#) ()=default
- virtual void [render](#) ()
- virtual void [interact](#) ()

Additional Inherited Members

Protected Member Functions inherited from [scene::internal::BaseScene](#)

- virtual bool [render_go_button](#) () const
- virtual void [render_options](#) ([SceneOptions](#) &scene_config)
- virtual void [render_inputs](#) ()

Protected Attributes inherited from [scene::internal::BaseScene](#)

- float [options_head](#) {}
- [component::TextInput](#) [m_text_input](#) {"value"}
- [component::TextInput](#) [m_index_input](#) {"index"}
- [component::FileDialog](#) [m_file_dialog](#)
- [component::SequenceController](#) [m_sequence_controller](#)
- [component::CodeHighlighter](#) [m_code_highlighter](#)
- bool [m_edit_mode](#) {}
- bool [m_edit_action](#) {}

Static Protected Attributes inherited from [scene::internal::BaseScene](#)

- static constexpr [Vector2](#) [button_size](#) {200, 50}
- static constexpr int [head_offset](#) = 20

6.32.1 Detailed Description

Definition at line 14 of file [stack_scene.hpp](#).

6.32.2 Member Function Documentation

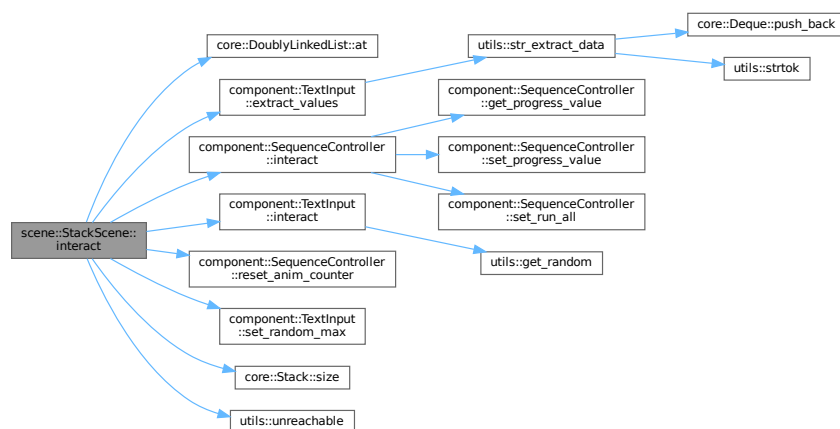
6.32.2.1 `interact()`

```
void scene::StackScene::interact ( ) [override], [virtual]
```

Reimplemented from [scene::internal::BaseScene](#).

Definition at line 71 of file [stack_scene.cpp](#).

Here is the call graph for this function:



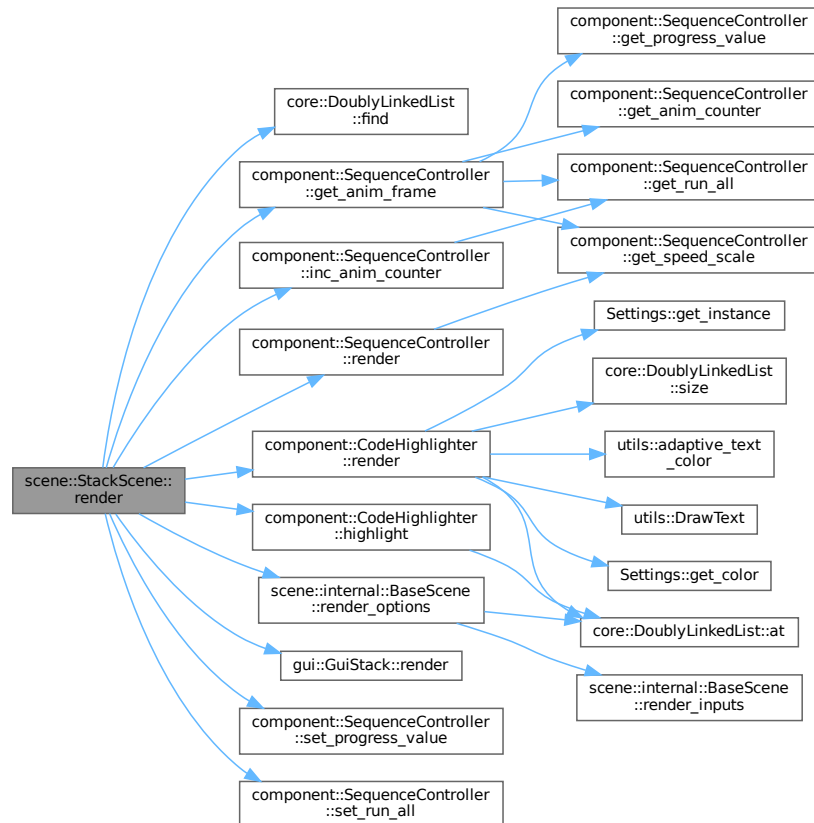
6.32.2.2 render()

```
void scene::StackScene::render ( ) [override], [virtual]
```

Reimplemented from [scene::internal::BaseScene](#).

Definition at line 17 of file [stack_scene.cpp](#).

Here is the call graph for this function:



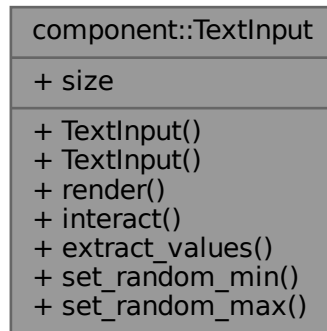
The documentation for this class was generated from the following files:

- [src/scene/stack_scene.hpp](#)
- [src/scene/stack_scene.cpp](#)

6.33 component::TextInput Class Reference

```
#include <text_input.hpp>
```

Collaboration diagram for component::TextInput:



Public Member Functions

- [TextInput](#) ()=default
- [TextInput](#) (const char *label)
- void [render](#) (float &options_head, float head_offset)
- bool [interact](#) ()
- [core::Deque](#)< int > [extract_values](#) ()
- void [set_random_min](#) (int value)
- void [set_random_max](#) (int value)

Static Public Attributes

- static constexpr Vector2 [size](#) {200, 50}

6.33.1 Detailed Description

Definition at line 12 of file [text_input.hpp](#).

6.33.2 Constructor & Destructor Documentation

6.33.2.1 TextInput() [1/2]

```
component::TextInput::TextInput ( ) [default]
```

6.33.2.2 TextInput() [2/2]

```
component::TextInput::TextInput (
    const char * label )
```

Definition at line 14 of file [text_input.cpp](#).

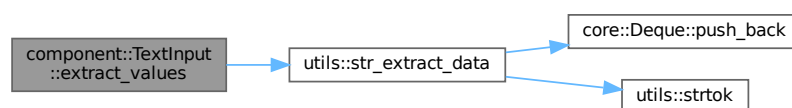
6.33.3 Member Function Documentation

6.33.3.1 extract_values()

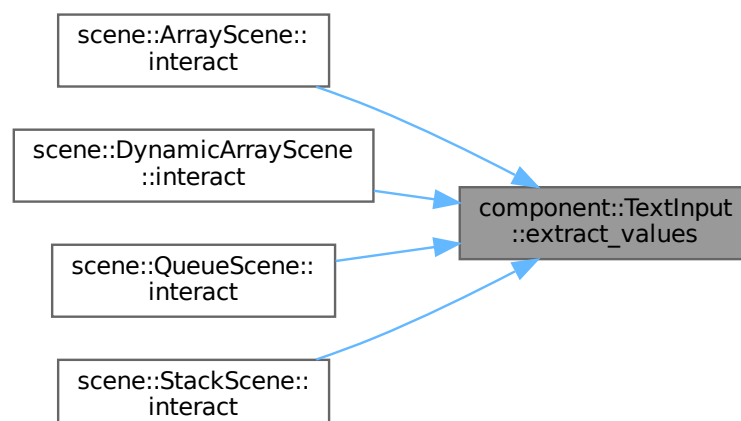
```
core::Deque< int > component::TextInput::extract_values ( )
```

Definition at line 58 of file [text_input.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

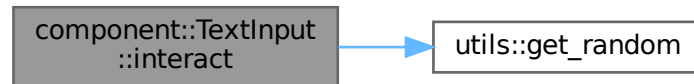


6.33.3.2 interact()

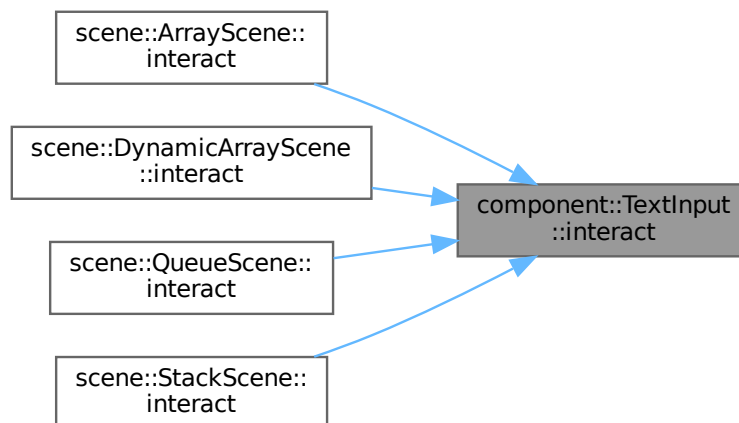
```
bool component::TextInput::interact ( )
```

Definition at line 46 of file [text_input.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

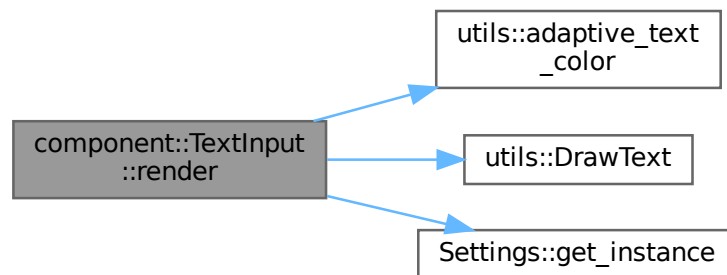


6.33.3.3 render()

```
void component::TextInput::render (
    float & options_head,
    float head_offset )
```

Definition at line 20 of file [text_input.cpp](#).

Here is the call graph for this function:

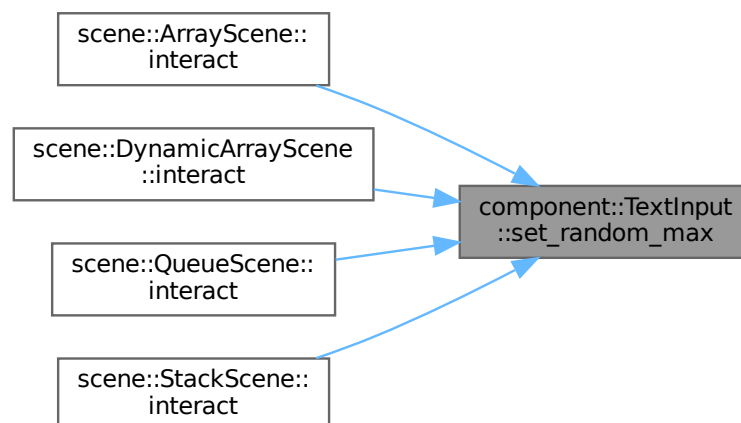


6.33.3.4 set_random_max()

```
void component::TextInput::set_random_max (
    int value )
```

Definition at line 18 of file [text_input.cpp](#).

Here is the caller graph for this function:



6.33.3.5 set_random_min()

```
void component::TextInput::set_random_min (
    int value )
```

Definition at line 16 of file [text_input.cpp](#).

6.33.4 Member Data Documentation

6.33.4.1 size

```
constexpr Vector2 component::TextInput::size {200, 50} [static], [constexpr]
```

Definition at line 23 of file [text_input.hpp](#).

The documentation for this class was generated from the following files:

- [src/component/text_input.hpp](#)
- [src/component/text_input.cpp](#)

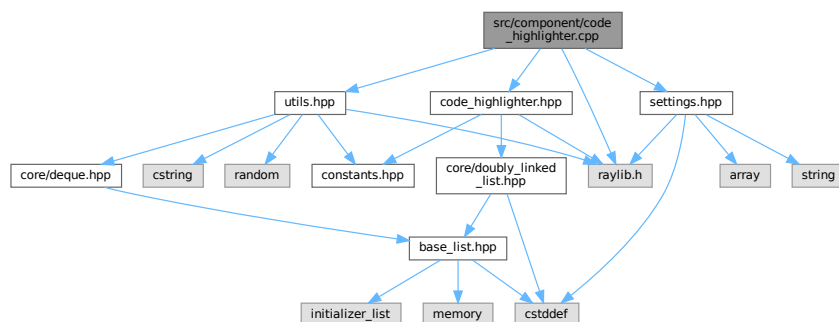
Chapter 7

File Documentation

7.1 src/component/code_highlighter.cpp File Reference

```
#include "code_highlighter.hpp"
#include "raylib.h"
#include "settings.hpp"
#include "utils.hpp"
```

Include dependency graph for code_highlighter.cpp:



Namespaces

- namespace `component`

7.2 code_highlighter.cpp

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

```
00001 #include "code_highlighter.hpp"
00002
00003 #include "raylib.h"
00004 #include "settings.hpp"
00005 #include "utils.hpp"
00006
00007 namespace component {
00008
```

```

00009 void CodeHighlighter::render() {
00010     for (int i = 0; i < m_src_code.size(); ++i) {
00011         const Settings& settings = Settings::get_instance();
00012
00013         int color_index = (i == m_highlighted_line) ? 4 : 0;
00014         Color bg_color = settings.get_color(color_index);
00015         Color text_color = utils::adaptive_text_color(bg_color);
00016
00017         Rectangle shape{head_pos.x, head_pos.y + i * height, width, height};
00018         Vector2 text_head = {head_pos.x + 10, head_pos.y + i * height + 5};
00019
00020         DrawRectangleRec(shape, bg_color);
00021         utils::DrawText(m_src_code.at(i), text_head, text_color, 20, 2);
00022     }
00023 }
00024
00025 void CodeHighlighter::set_code(core::DoublyLinkedList<const char*>&& src_code) {
00026     clear();
00027     m_src_code = src_code;
00028 }
00029
00030 void CodeHighlighter::push_into_sequence(int line_number) {
00031     m_sequence.insert(m_sequence.size(), line_number);
00032 }
00033
00034 void CodeHighlighter::highlight(int frame_idx) {
00035     m_highlighted_line = m_sequence.at(frame_idx);
00036 }
00037
00038 void CodeHighlighter::clear() {
00039     m_src_code.clear();
00040     m_sequence.clear();
00041 }
00042
00043 } // namespace component

```

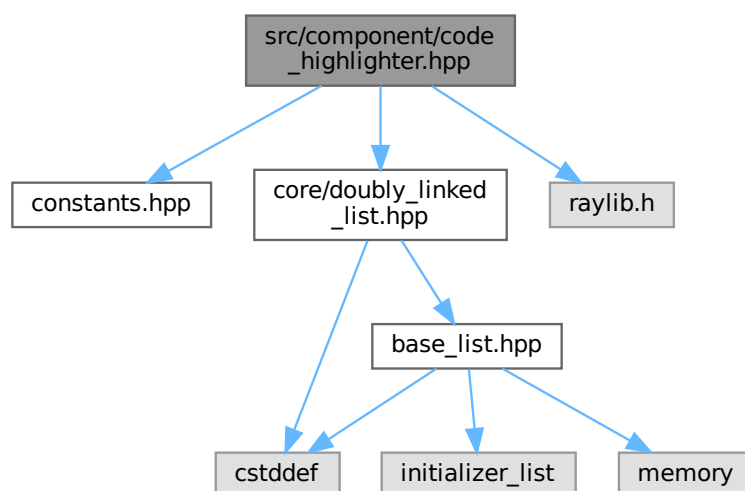
7.3 src/component/code_highlighter.hpp File Reference

```

#include "constants.hpp"
#include "core/doubly_linked_list.hpp"
#include "raylib.h"

```

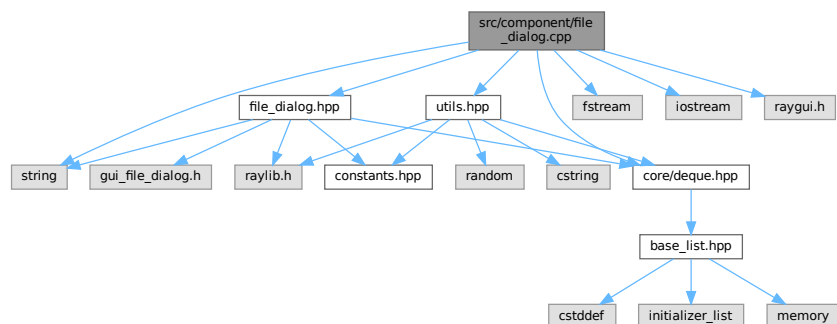
Include dependency graph for code_highlighter.hpp:



7.5 src/component/file_dialog.cpp File Reference

```
#include "file_dialog.hpp"
#include <fstream>
#include <iostream>
#include <string>
#include "core/deque.hpp"
#include "raygui.h"
#include "utils.hpp"
```

Include dependency graph for file_dialog.cpp:



Namespaces

- namespace [component](#)

7.6 file_dialog.cpp

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

```
00001 #include "file_dialog.hpp"
00002
00003 #include <fstream>
00004 #include <iostream>
00005 #include <string>
00006
00007 #include "core/deque.hpp"
00008 #include "raygui.h"
00009 #include "utils.hpp"
00010
00011 namespace component {
00012
00013 FileDialog::FileDialog(int mode, const char* title, const char* message)
00014     : m_mode{mode}, m_title{title}, m_message{message} {}
00015
00016 FileDialog::FileDialog() : FileDialog(0, "Open file...", "Open file") {}
00017
00018 int FileDialog::render(float x, float y) {
00019     m_file_dialog_state.title = m_title;
00020     m_file_dialog_state.fileName = m_file_input;
00021     m_file_dialog_state.message = m_message;
00022     m_file_dialog_state.dialogType = m_mode;
00023
00024     int result = -1;
00025     if (m_file_dialog_state.windowActive) {
00026         GuiLock();
00027         result = GuiFileDialog(&m_file_dialog_state);
00028         if (result >= 0) {
00029             m_file_dialog_state.windowActive = false;
00030         }
00031     }
```

```

00032
00033     const Rectangle shape{x, y, size.x, size.y};
00034
00035     if (GuiButton(shape, GuiIconText(ICON_FILE_OPEN, "Select file"))) {
00036         m_file_dialog_state.windowActive = true;
00037     }
00038
00039     GuiUnlock();
00040     return result;
00041 }
00042
00043 int FileDialog::render_head(float& options_head, float head_offset) {
00044     int ret = render(options_head, constants::scene_height - size.y);
00045     options_head += (size.x + head_offset);
00046     return ret;
00047 }
00048
00049 core::Deque<int> FileDialog::extract_values() {
00050     std::ifstream ifs(get_path());
00051     char buffer[constants::text_buffer_size]{}; // NOLINT
00052     ifs » buffer;
00053
00054     return utils::str_extract_data(buffer); // NOLINT
00055 }
00056
00057 bool FileDialog::is_active() const { return m_file_dialog_state.windowActive; }
00058
00059 void FileDialog::set_mode_open() { m_mode = DIALOG_OPEN_FILE; }
00060
00061 void FileDialog::set_mode_save() { m_mode = DIALOG_SAVE_FILE; }
00062
00063 void FileDialog::set_message(const char* message) { m_message = message; }
00064
00065 void FileDialog::set_title(const char* title) { m_title = title; }
00066 std::string FileDialog::get_path() { return m_file_input; }
00067
00068 } // namespace component

```

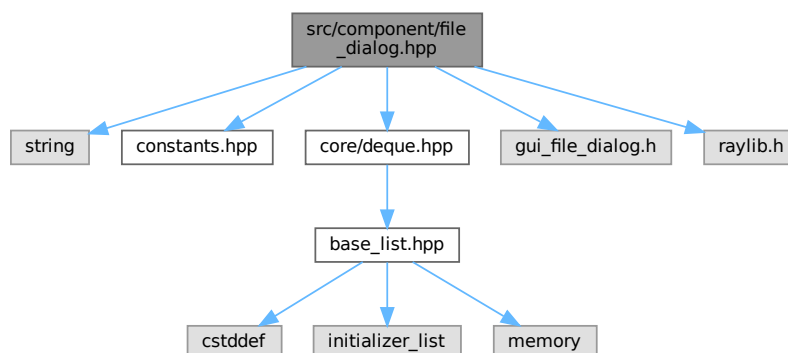
7.7 src/component/file_dialog.hpp File Reference

```

#include <string>
#include "constants.hpp"
#include "core/deque.hpp"
#include "gui_file_dialog.h"
#include "raylib.h"

```

Include dependency graph for file_dialog.hpp:



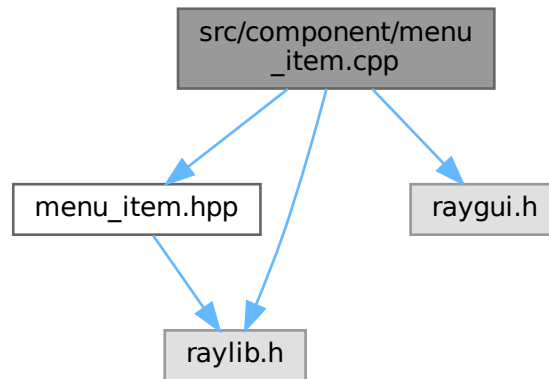
7.9 src/component/menu_item.cpp File Reference

```
#include "menu_item.hpp"
```

```
#include "raygui.h"
```

```
#include "raylib.h"
```

Include dependency graph for menu_item.cpp:



Namespaces

- namespace `component`

7.10 menu_item.cpp

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

```

00001 #include "menu_item.hpp"
00002
00003 #include "raygui.h"
00004 #include "raylib.h"
00005
00006 namespace component {
00007
00008 MenuItem::MenuItem(int scene, const char* text, int x, int y,
00009                   const char* img_path)
00010     : m_scene{scene},
00011       m_text{text},
00012       m_x{x},
00013       m_y{y},
00014       m_texture(LoadTextureFromImage(LoadImage(img_path))) {}
00015
00016 int MenuItem::x() const { return m_x; }
00017 int MenuItem::y() const { return m_y; }
00018
00019 void MenuItem::render() {
00020     auto mouse = GetMousePosition();
00021     const Rectangle bound{(float)m_x, (float)m_y, block_width, block_height};
00022     const Rectangle text_bound{(float)m_x + 20,
00023                               (float)m_y + block_height - button_height,
00024                               button_width - 20, button_height};
00025
00026     DrawRectangleRec(bound, RAYWHITE);
00027     DrawTexture(m_texture, m_x, m_y, WHITE);
00028     GuiLabelButton(text_bound, m_text);
00029     DrawRectangleLinesEx(bound, 2, BLACK);

```

```

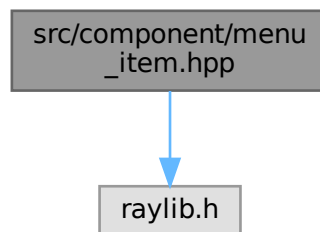
00030
00031     if (CheckCollisionPointRec(mouse, bound)) {
00032         DrawRectangle(m_x, m_y, block_width, block_height,
00033             ColorAlpha(BLUE, 0.3));
00034         m_clicked = IsMouseButtonPressed(MOUSE_LEFT_BUTTON);
00035     }
00036 }
00037
00038 bool MenuItem::clicked() const { return m_clicked; }
00039
00040 void MenuItem::reset() { m_clicked = false; }
00041
00042 } // namespace component

```

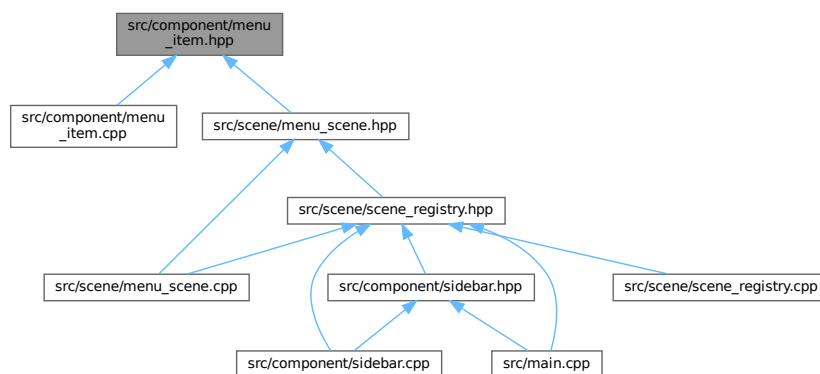
7.11 src/component/menu_item.hpp File Reference

```
#include "raylib.h"
```

Include dependency graph for menu_item.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class `component::MenuItem`

Namespaces

- namespace `component`

7.12 menu_item.hpp

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

```

00001 #ifndef COMPONENT_MENU_ITEM_HPP_
00002 #define COMPONENT_MENU_ITEM_HPP_
00003
00004 #include "raylib.h"
00005
00006 namespace component {
00007
00008 class MenuItem {
00009 private:
00010     int m_scene{};
00011     int m_x{};
00012     int m_y{};
00013     Texture2D m_texture{};
00014     const char* m_text{};
00015
00016     bool m_clicked{};
00017
00018 public:
00019     static constexpr int block_width = 300;
00020     static constexpr int block_height = 200;
00021     static constexpr int button_width = block_width;
00022     static constexpr int button_height = 50;
00023
00024     MenuItem() = default;
00025     MenuItem(int scene, const char* text, int x, int y, const char* img_path);
00026
00027     int x() const;
00028     int y() const;
00029
00030     void render();
00031     bool clicked() const;
00032     void reset();
00033 };
00034
00035 } // namespace component
00036
00037 #endif // COMPONENT_MENU_ITEM_HPP_

```

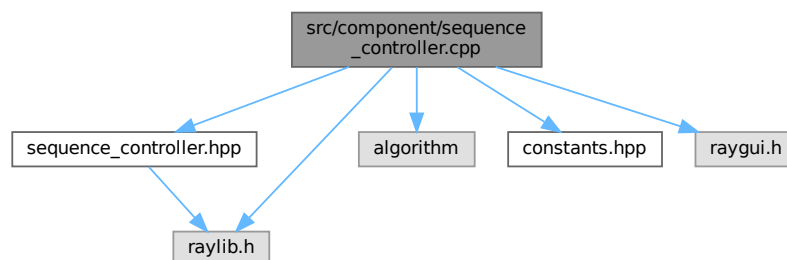
7.13 src/component/sequence_controller.cpp File Reference

```

#include "sequence_controller.hpp"
#include <algorithm>
#include "constants.hpp"
#include "raygui.h"
#include "raylib.h"

```

Include dependency graph for sequence_controller.cpp:



Namespaces

- namespace [component](#)

7.14 sequence_controller.cpp

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

```

00001 #include "sequence_controller.hpp"
00002
00003 #include <algorithm>
00004
00005 #include "constants.hpp"
00006 #include "raygui.h"
00007 #include "raylib.h"
00008
00009 namespace component {
00010
00011 void SequenceController::set_max_value(int num) { m_num_steps = num; }
00012
00013 void SequenceController::set_progress_value(int value) {
00014     m_progress_value = value;
00015 }
00016
00017 void SequenceController::set_run_all(bool run_all) { m_run_all = run_all; }
00018
00019 bool SequenceController::get_run_all() const { return m_run_all; }
00020
00021 int SequenceController::get_progress_value() const { return m_progress_value; }
00022
00023 float SequenceController::get_speed_scale() const {
00024     return (float)m_speed / speed_scale;
00025 }
00026
00027 void SequenceController::reset_anim_counter() { m_anim_counter = 0; }
00028
00029 void SequenceController::inc_anim_counter() {
00030     if (get_run_all()) {
00031         ++m_anim_counter;
00032     }
00033 }
00034
00035 int SequenceController::get_anim_counter() const { return m_anim_counter; }
00036
00037 void SequenceController::set_rerun() {
00038     reset_anim_counter();
00039     set_run_all(true);
00040 }
00041
00042 int SequenceController::get_anim_frame() const {
00043     if (get_run_all()) {
00044         return 2.0F * get_anim_counter() * get_speed_scale() /
00045             constants::frames_per_second;
00046     } else {
00047         return get_progress_value();
00048     }
00049 }
00050
00051 void SequenceController::render() {
00052     Rectangle replay_shape{button_size.x * 0.5F,
00053         constants::scene_height - 1.5F * button_size.x,
00054         button_size.x, button_size.y};
00055
00056     Rectangle prev_frame_shape{
00057         replay_shape.x + replay_shape.width + button_size.x * 0.5F,
00058         replay_shape.y, button_size.x, button_size.y};
00059
00060     Rectangle progress_shape{prev_frame_shape.x + button_size.x * 1.5F,
00061         replay_shape.y, 360, button_size.y};
00062
00063     Rectangle next_frame_shape{
00064         progress_shape.x + progress_shape.width + button_size.x * 0.5F,
00065         replay_shape.y, button_size.x, button_size.y};
00066
00067     Rectangle prev_speed_shape{prev_frame_shape.x + 240,
00068         prev_frame_shape.y - 1.5F * button_size.y,
00069         button_size.x, button_size.y};
00070
00071     Rectangle next_speed_shape{next_frame_shape.x,
00072         next_frame_shape.y - 1.5F * button_size.y,
00073         button_size.x, button_size.y};

```

```

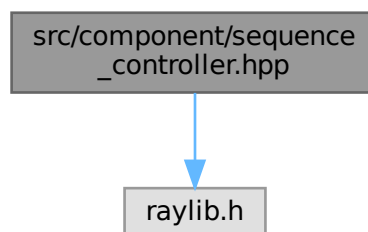
00074
00075     Rectangle speed_shape{prev_speed_shape.x + 1.5F * button_size.x,
00076                          prev_speed_shape.y, 120, button_size.y};
00077
00078     m_prev_speed = GuiButton(prev_speed_shape, "#114#");
00079     m_next_speed = GuiButton(next_speed_shape, "#115#");
00080     GuiStatusBar(speed_shape, TextFormat("Speed: %.2fx", get_speed_scale()));
00081
00082     m_replay = GuiButton(replay_shape, "#75#");
00083     m_prev_frame = GuiButton(prev_frame_shape, "#72#");
00084     m_progress_value =
00085         (int)GuiProgressBar(progress_shape, nullptr, nullptr,
00086                             (float)m_progress_value, 0, (float)m_num_steps);
00087     m_next_frame = GuiButton(next_frame_shape, "#73#");
00088 }
00089
00090 bool SequenceController::interact() {
00091     if (m_replay) {
00092         set_progress_value(0);
00093         set_run_all(true);
00094         return true;
00095     }
00096
00097     if (m_prev_frame) {
00098         set_progress_value(std::max(get_progress_value() - 1, 0));
00099         return true;
00100     }
00101
00102     if (m_next_frame) {
00103         set_progress_value(std::min(get_progress_value() + 1, m_num_steps));
00104         return true;
00105     }
00106
00107     if (m_prev_speed) {
00108         m_speed = std::max(m_speed - 1, 2);
00109         return true;
00110     }
00111
00112     if (m_next_speed) {
00113         m_speed = std::min(m_speed + 1, 6);
00114         return true;
00115     }
00116
00117     return false;
00118 }
00119
00120 } // namespace component

```

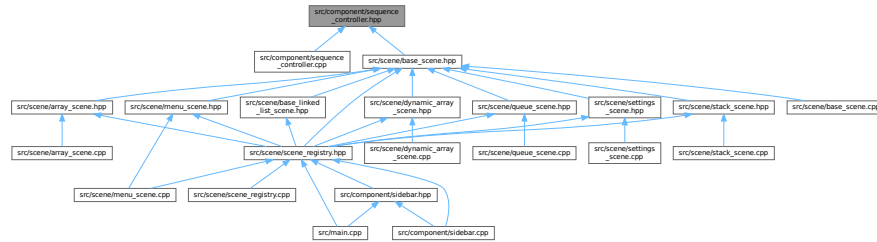
7.15 src/component/sequence_controller.hpp File Reference

#include "raylib.h"

Include dependency graph for sequence_controller.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [component::SequenceController](#)

Namespaces

- namespace [component](#)

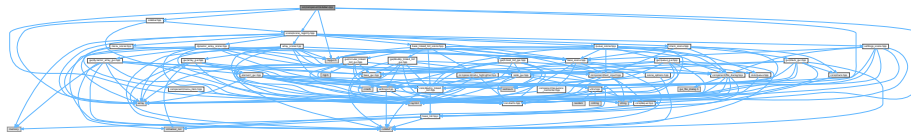
7.16 sequence_controller.hpp

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

```
00001 #ifndef COMPONENT_SEQUENCE_CONTROLLER_HPP_
00002 #define COMPONENT_SEQUENCE_CONTROLLER_HPP_
00003
00004 #include "raylib.h"
00005
00006 namespace component {
00007
00008 class SequenceController {
00009 private:
00010     static constexpr Vector2 button_size{25, 25};
00011     static constexpr int speed_scale = 4;
00012
00013     bool m_replay{};
00014     bool m_prev_frame{};
00015     bool m_next_frame{};
00016     int m_progress_value{};
00017     int m_num_steps{};
00018     bool m_run_all{};
00019     int m_anim_counter{};
00020
00021     bool m_prev_speed{};
00022     bool m_next_speed{};
00023     int m_speed{speed_scale};
00024
00025 public:
00026     void render();
00027     bool interact();
00028
00029     void set_max_value(int num);
00030     void set_progress_value(int value);
00031     void set_run_all(bool run_all);
00032     void set_rerun();
00033
00034     bool get_run_all() const;
00035     int get_progress_value() const;
00036     float get_speed_scale() const;
00037
00038     void reset_anim_counter();
00039     void inc_anim_counter();
00040     int get_anim_counter() const;
00041     int get_anim_frame() const;
00042 };
00043
00044 } // namespace component
00045
00046 #endif // COMPONENT_SEQUENCE_CONTROLLER_HPP_
```

7.17 src/component/sidebar.cpp File Reference

```
#include "sidebar.hpp"
#include "constants.hpp"
#include "raygui.h"
#include "raylib.h"
#include "scene/scene_registry.hpp"
#include "utils.hpp"
Include dependency graph for sidebar.cpp:
```



Namespaces

- namespace `component`

7.18 sidebar.cpp

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

```
00001 #include "sidebar.hpp"
00002
00003 #include "constants.hpp"
00004 #include "raygui.h"
00005 #include "raylib.h"
00006 #include "scene/scene_registry.hpp"
00007 #include "utils.hpp"
00008
00009 namespace component {
00010
00011 void SideBar::render() {
00012     (m_edit_mode) ? GuiLock() : GuiUnlock();
00013
00014     scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();
00015     int options_head = 2 * constants::sidebar_width;
00016
00017     constexpr float scale = 0.2;
00018
00019     constexpr Rectangle menu_button_shape{20, 20, button_height * 2,
00020                                           button_height};
00021
00022     constexpr Rectangle selection_shape{
00023         menu_button_shape.x + menu_button_shape.width + 10, menu_button_shape.y,
00024         button_width, button_height};
00025
00026     constexpr Rectangle settings_button_shape{
00027         constants::scene_width - button_height - 20, 20, button_height,
00028         button_height};
00029
00030     m_next_scene = registry.get_scene();
00031
00032     bool menu_is_next = m_next_scene == scene::Menu;
00033     bool settings_is_next = m_next_scene == scene::Settings;
00034
00035     if (!menu_is_next) {
00036         m_return_menu = GuiButton(menu_button_shape, "#118#Menu");
00037     }
00038
00039     if (!menu_is_next && !settings_is_next) {
00040         if (GuiDropdownBox(selection_shape, sidebar_labels, &m_next_scene,
00041                           m_edit_mode)) {
00042             m_pressed = true;
00043             m_edit_mode ^= 1;
00044         }
00045     }
00046
00047     m_return_settings = GuiButton(settings_button_shape, "#142#");
```

```

00046 }
00047
00048 void SideBar::interact() {
00049     scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();
00050     bool menu_is_current = registry.get_scene() == scene::Menu;
00051     bool settings_is_current = registry.get_scene() == scene::Settings;
00052
00053     if (!menu_is_current) {
00054         if (m_return_menu) {
00055             registry.set_scene(scene::Menu);
00056             m_return_menu = false;
00057             return;
00058         }
00059     }
00060
00061     if (!menu_is_current && !settings_is_current) {
00062         if (m_pressed) {
00063             registry.set_scene(m_next_scene);
00064             m_pressed = false;
00065             return;
00066         }
00067     }
00068
00069     if (m_return_settings) {
00070         if (settings_is_current) {
00071             registry.set_scene(m_scene_before_settings);
00072         } else {
00073             m_scene_before_settings = registry.get_scene();
00074             registry.set_scene(scene::Settings);
00075         }
00076         m_return_settings = false;
00077         return;
00078     }
00079 }
00080
00081 } // namespace component

```

7.19 src/component/sidebar.hpp File Reference

```

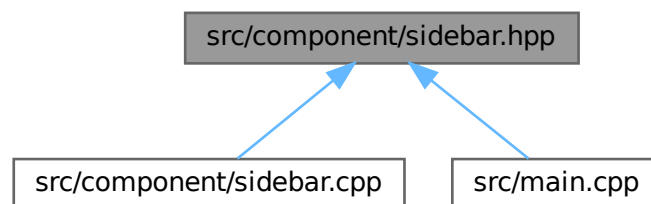
#include <array>
#include "constants.hpp"
#include "scene/scene_registry.hpp"

```

Include dependency graph for sidebar.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [component::SideBar](#)

Namespaces

- namespace [component](#)

7.20 sidebar.hpp

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

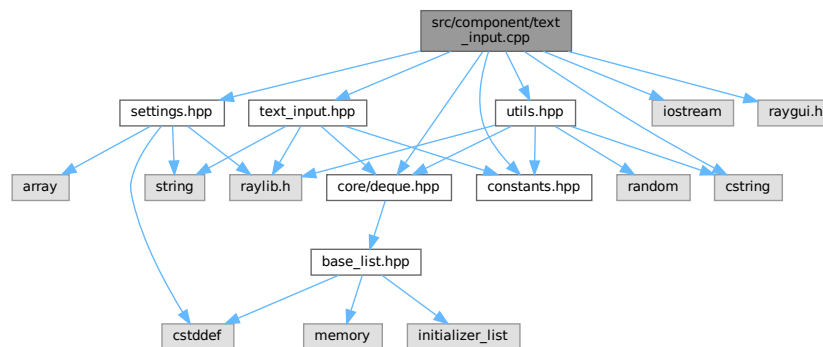
```
00001 #ifndef COMPONENT_SIDEBAR_HPP_
00002 #define COMPONENT_SIDEBAR_HPP_
00003
00004 #include <array>
00005
00006 #include "constants.hpp"
00007 #include "scene/scene_registry.hpp"
00008
00009 namespace component {
00010
00011 class SideBar {
00012 private:
00013     static constexpr int num_scenes = 8;
00014
00015     static constexpr int button_width = constants::sidebar_width;
00016     static constexpr int button_height = 50;
00017
00018     static constexpr const char* sidebar_labels =
00019         "Array;"
00020         "Dynamic Array;"
00021         "Linked List;"
00022         "Doubly Linked List;"
00023         "Circular Linked List;"
00024         "Stack;"
00025         "Queue";
00026
00027     int m_next_scene{};
00028     bool m_edit_mode{};
00029     bool m_return_menu{};
00030     bool m_return_settings{};
00031     int m_scene_before_settings{};
00032     bool m_pressed{};
00033
00034 public:
00035     void render();
00036     void interact();
00037 };
00038
00039 } // namespace component
00040
00041 #endif // COMPONENT_SIDEBAR_HPP_
```

7.21 src/component/text_input.cpp File Reference

```
#include "text_input.hpp"
#include <cstring>
#include <iostream>
#include "constants.hpp"
#include "core/deque.hpp"
#include "raygui.h"
#include "settings.hpp"
```

```
#include "utils.hpp"
```

Include dependency graph for text_input.cpp:



Namespaces

- namespace `component`

7.22 text_input.cpp

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

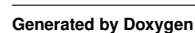
```

00001 #include "text_input.hpp"
00002
00003 #include <cstring>
00004 #include <iostream>
00005
00006 #include "constants.hpp"
00007 #include "core/deque.hpp"
00008 #include "raygui.h"
00009 #include "settings.hpp"
00010 #include "utils.hpp"
00011
00012 namespace component {
00013
00014 TextInput::TextInput(const char* label) : m_label{label} {}
00015
00016 void TextInput::set_random_min(int value) { m_random_min = value; }
00017
00018 void TextInput::set_random_max(int value) { m_random_max = value; }
00019
00020 void TextInput::render(float& options_head, float head_offset) {
00021     Rectangle shape{options_head, constants::scene_height - size.y, size.x,
00022                     size.y};
00023
00024     utils::DrawText(
00025         m_label, {options_head, constants::scene_height - size.y - 25},
00026         utils::adaptive_text_color(
00027             Settings::get_instance().get_color(Settings::num_color - 1)),
00028         20, 2);
00029
00030     DrawRectangleRec(shape, RAYWHITE);
00031
00032     if (GuiTextBox(shape, static_cast<char*>(m_text_input),
00033                  constants::text_buffer_size, m_is_active)) {
00034         m_is_active ^= 1;
00035     }
00036
00037     options_head += (shape.width + head_offset);
00038
00039     shape = {options_head, constants::scene_height - size.y, size.y, size.y};
00040
00041     m_set_random = GuiButton(shape, "#78#");
00042

```


7.23 src/component/text_input.hpp File Reference

Include dependency graph for text_input.hpp:



Classes

- class [component::TextInput](#)

Namespaces

- namespace [component](#)

7.24 text_input.hpp

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

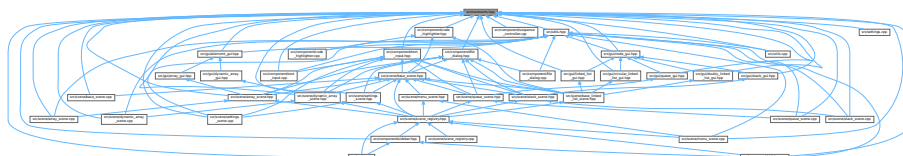
```

00001 #ifndef COMPONENT_TEXT_INPUT_HPP_
00002 #define COMPONENT_TEXT_INPUT_HPP_
00003
00004 #include <string>
00005
00006 #include "constants.hpp"
00007 #include "core/deque.hpp"
00008 #include "raylib.h"
00009
00010 namespace component {
00011
00012 class TextInput {
00013 private:
00014     char m_text_input[constants::text_buffer_size] = ""; // NOLINT
00015     bool m_is_active{};
00016     const char* m_label{};
00017
00018     int m_random_min{constants::min_val};
00019     int m_random_max{constants::max_val};
00020     bool m_set_random{};
00021
00022 public:
00023     static constexpr Vector2 size{200, 50};
00024
00025     TextInput() = default;
00026     TextInput(const char* label);
00027
00028     void render(float& options_head, float head_offset);
00029     bool interact();
00030     core::Deque<int> extract_values();
00031     void set_random_min(int value);
00032     void set_random_max(int value);
00033 };
00034
00035 } // namespace component
00036
00037 #endif // COMPONENT_TEXT_INPUT_HPP_

```

7.25 src/constants.hpp File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [constants](#)

Variables

- constexpr int `constants::scene_width` = 1366
- constexpr int `constants::scene_height` = 768
- constexpr int `constants::frames_per_second` = 30
- constexpr int `constants::sidebar_width` = 256
- constexpr int `constants::ani_speed` = 8
- constexpr int `constants::text_buffer_size` = 512
- constexpr int `constants::min_val` = 0
- constexpr int `constants::max_val` = 999
- constexpr int `constants::default_font_size` = 60
- constexpr const char * `constants::default_color_path` = "data/color.bin"

7.26 constants.hpp

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

```

00001 #ifndef CONSTANTS_HPP_
00002 #define CONSTANTS_HPP_
00003
00004 namespace constants {
00005
00006 constexpr int scene_width = 1366;
00007 constexpr int scene_height = 768;
00008 constexpr int frames_per_second = 30;
00009
00010 constexpr int sidebar_width = 256;
00011 constexpr int ani_speed = 8;
00012
00013 constexpr int text_buffer_size = 512;
00014
00015 constexpr int min_val = 0;
00016 constexpr int max_val = 999;
00017
00018 constexpr int default_font_size = 60;
00019
00020 constexpr const char* default_color_path = "data/color.bin";
00021
00022 } // namespace constants
00023
00024 #endif // CONSTANTS_HPP_

```

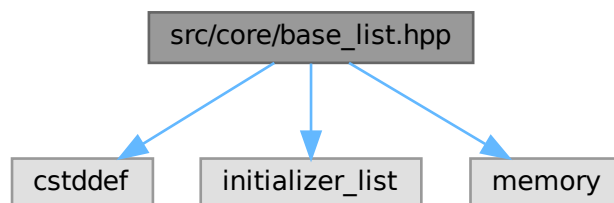
7.27 src/core/base_list.hpp File Reference

```

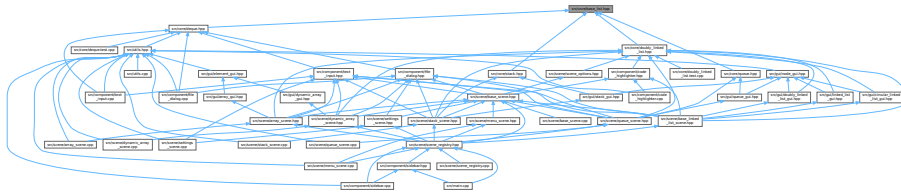
#include <cstddef>
#include <initializer_list>
#include <memory>

```

Include dependency graph for base_list.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [core::BaseList< T >](#)
- struct [core::BaseList< T >::Node](#)

Namespaces

- namespace [core](#)

7.28 base_list.hpp

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

```

00001 #ifndef CORE_BASE_LIST_HPP_
00002 #define CORE_BASE_LIST_HPP_
00003
00004 #include <cstddef>
00005 #include <initializer_list>
00006 #include <memory>
00007
00008 namespace core {
00009
00010 template<typename T>
00011 class BaseList {
00012 protected:
00013     struct Node;
00014     using Node_ptr = Node*;
00015
00016     struct Node {
00017         T data{};
00018         Node_ptr prev{};
00019         Node_ptr next{};
00020     };
00021
00022     Node_ptr m_head{nullptr};
00023     Node_ptr m_tail{nullptr};
00024     std::size_t m_size{};
00025
00026     void init_first_element(const T& elem);
00027     void clean_up();
00028     void copy_data(const BaseList& rhs);
00029
00030     void push_back(const T& elem);
00031     void push_front(const T& elem);
00032
00033     T& back() const;
00034     T& front() const;
00035
00036     void pop_front();
00037     void pop_back();
00038
00039 public:
00040     BaseList() = default;
00041     BaseList(std::initializer_list<T> init_list);
00042     BaseList(const BaseList& rhs);
00043     BaseList& operator=(const BaseList& rhs);
00044     BaseList(BaseList&& rhs) noexcept;
00045     BaseList& operator=(BaseList&& rhs) noexcept;

```

```

00046     ~BaseList();
00047
00048     [[nodiscard]] bool empty() const;
00049     [[nodiscard]] std::size_t size() const;
00050 };
00051
00052 template<typename T>
00053 BaseList<T>::BaseList(const BaseList& rhs) {
00054     copy_data(rhs);
00055 }
00056
00057 template<typename T>
00058 BaseList<T>::BaseList(std::initializer_list<T> init_list) {
00059     for (const auto& elem : init_list) {
00060         push_back(elem);
00061     }
00062 }
00063
00064 template<typename T>
00065 BaseList<T>& BaseList<T>::operator=(const BaseList& rhs) {
00066     if (this != &rhs) {
00067         copy_data(rhs);
00068     }
00069
00070     return *this;
00071 }
00072
00073 template<typename T>
00074 BaseList<T>::BaseList(BaseList&& rhs) noexcept
00075     : m_head{rhs.m_head}, m_tail{rhs.m_tail}, m_size{rhs.m_size} {
00076     rhs.m_head = nullptr;
00077     rhs.m_tail = nullptr;
00078     rhs.m_size = 0;
00079 }
00080
00081 template<typename T>
00082 BaseList<T>& BaseList<T>::operator=(BaseList&& rhs) noexcept {
00083     if (this != &rhs) {
00084         clean_up();
00085
00086         m_head = rhs.m_head;
00087         m_tail = rhs.m_tail;
00088         m_size = rhs.m_size;
00089
00090         rhs.m_head = nullptr;
00091         rhs.m_tail = nullptr;
00092         rhs.m_size = 0;
00093     }
00094
00095     return *this;
00096 }
00097
00098 template<typename T>
00099 BaseList<T>::~~BaseList() {
00100     clean_up();
00101 }
00102
00103 template<typename T>
00104 bool BaseList<T>::empty() const {
00105     return m_size == 0;
00106 }
00107
00108 template<typename T>
00109 std::size_t BaseList<T>::size() const {
00110     return m_size;
00111 }
00112
00113 template<typename T>
00114 void BaseList<T>::init_first_element(const T& elem) {
00115     m_head = new Node{elem, nullptr, nullptr};
00116     m_tail = m_head;
00117     m_size = 1;
00118 }
00119
00120 template<typename T>
00121 void BaseList<T>::clean_up() {
00122     Node_ptr ptr{nullptr};
00123
00124     while (m_head != nullptr) {
00125         ptr = m_head->next;
00126         delete m_head;
00127         m_head = ptr;
00128     }
00129
00130     m_tail = m_head;
00131     m_size = 0;
00132 }

```

```

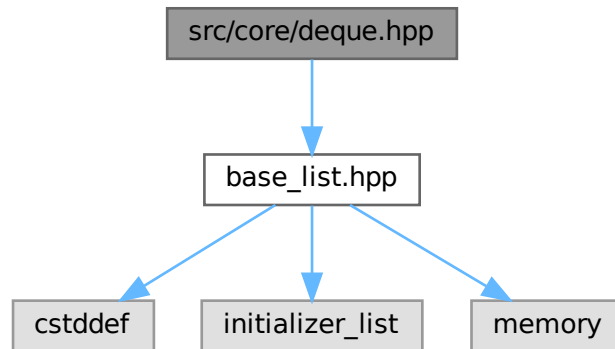
00133
00134 template<typename T>
00135 void BaseList<T>::copy_data(const BaseList& rhs) {
00136     for (Node_ptr ptr = rhs.m_head; ptr != nullptr; ptr = ptr->next) {
00137         push_back(ptr->data);
00138     }
00139 }
00140
00141 template<typename T>
00142 void BaseList<T>::push_back(const T& elem) {
00143     if (empty()) {
00144         init_first_element(elem);
00145         return;
00146     }
00147
00148     m_tail->next = new Node{elem, m_tail, nullptr};
00149     m_tail = m_tail->next;
00150     ++m_size;
00151 }
00152
00153 template<typename T>
00154 void BaseList<T>::push_front(const T& elem) {
00155     if (empty()) {
00156         init_first_element(elem);
00157         return;
00158     }
00159
00160     m_head->prev = new Node{elem, nullptr, m_head};
00161     m_head = m_head->prev;
00162     ++m_size;
00163 }
00164
00165 template<typename T>
00166 T& BaseList<T>::back() const {
00167     return m_tail->data;
00168 }
00169
00170 template<typename T>
00171 T& BaseList<T>::front() const {
00172     return m_head->data;
00173 }
00174
00175 template<typename T>
00176 void BaseList<T>::pop_back() {
00177     if (size() <= 1) {
00178         clean_up();
00179         return;
00180     }
00181
00182     m_tail = m_tail->prev;
00183     delete m_tail->next;
00184     m_tail->next = nullptr;
00185     --m_size;
00186 }
00187
00188 template<typename T>
00189 void BaseList<T>::pop_front() {
00190     if (size() <= 1) {
00191         clean_up();
00192         return;
00193     }
00194
00195     m_head = m_head->next;
00196     delete m_head->prev;
00197     m_head->prev = nullptr;
00198     --m_size;
00199 }
00200
00201 } // namespace core
00202
00203 #endif // CORE_BASE_LIST_HPP_

```

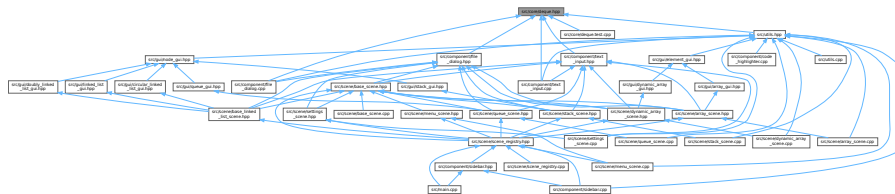
7.29 src/core/deque.hpp File Reference

```
#include "base_list.hpp"
```

Include dependency graph for deque.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class `core::Deque< T >`

Namespaces

- namespace `core`

7.30 deque.hpp

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

```

00001 #ifndef CORE_DEQUE_HPP_
00002 #define CORE_DEQUE_HPP_
00003
00004 #include "base_list.hpp"
00005
00006 namespace core {
00007
```

```

00008 template<typename T>
00009 class Deque : public BaseList<T> {
00010 private:
00011     using Base = BaseList<T>;
00012
00013 public:
00014     using Base::Base;
00015
00016     using Base::empty;
00017     using Base::size;
00018
00019     using Base::push_back;
00020     using Base::push_front;
00021
00022     using Base::back;
00023     using Base::front;
00024
00025     using Base::pop_back;
00026     using Base::pop_front;
00027 };
00028
00029 } // namespace core
00030
00031 #endif // CORE_DEQUE_HPP_

```

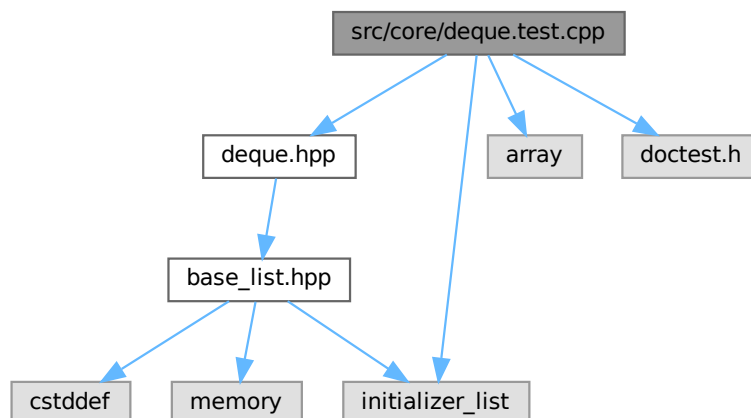
7.31 src/core/deque.test.cpp File Reference

```

#include "deque.hpp"
#include <array>
#include <initializer_list>
#include "doctest.h"

```

Include dependency graph for deque.test.cpp:



Functions

- `TEST_CASE` ("core::Deque functionality")
- `__attribute__((always_inline)) void check_match(core`
- `TEST_CASE` ("core::Deque special member functions")

Variables

- `constexpr std::array< int, 3 > list {1, 2, 3}`

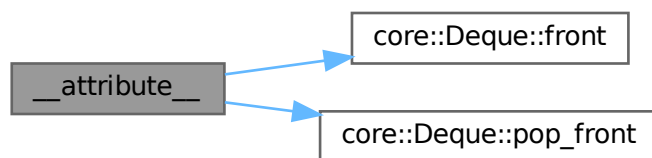
7.31.1 Function Documentation

7.31.1.1 `__attribute__()`

```
__attribute__ (  
    (always_inline) ) [inline]
```

Definition at line 38 of file [deque.test.cpp](#).

Here is the call graph for this function:

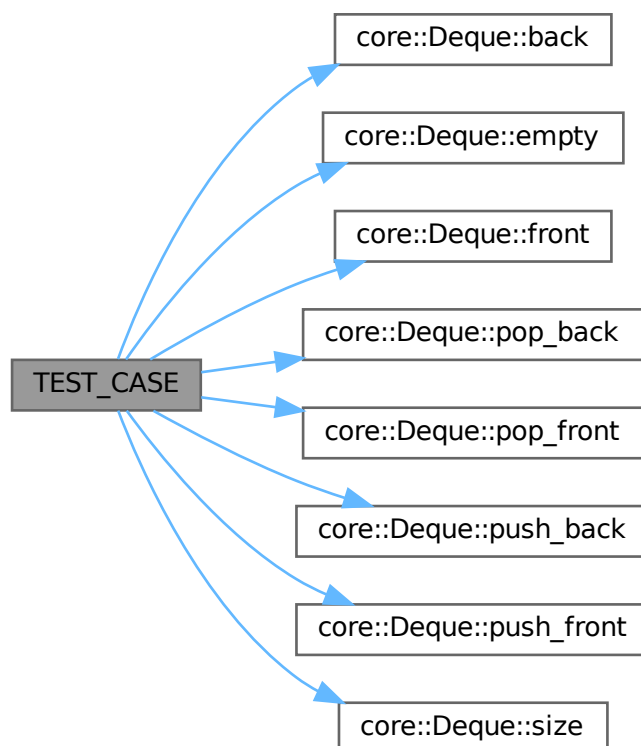


7.31.1.2 `TEST_CASE()` [1/2]

```
TEST_CASE (  
    "core::Deque functionality" )
```

Definition at line 8 of file [deque.test.cpp](#).

Here is the call graph for this function:



7.31.1.3 TEST_CASE() [2/2]

```
TEST_CASE (
    "core::Deque special member functions" )
```

Definition at line 45 of file [deque.test.cpp](#).

7.31.2 Variable Documentation

7.31.2.1 list

```
constexpr std::array<int, 3> list {1, 2, 3} [constexpr]
```

Definition at line 36 of file [deque.test.cpp](#).

7.32 deque.test.cpp

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

```

00001 #include "deque.hpp"
00002
00003 #include <array>
00004 #include <initializer_list>
00005
00006 #include "doctest.h"
00007
00008 TEST_CASE("core::Deque functionality") {
00009     core::Deque<int> deque;
00010     CHECK(deque.empty());
00011
00012     deque.push_back(2);
00013     deque.push_back(3);
00014     deque.push_front(1);
00015
00016     CHECK(deque.front() == 1);
00017     CHECK(deque.back() == 3);
00018     CHECK(deque.size() == 3);
00019
00020     deque.pop_back();
00021     CHECK(deque.back() == 2);
00022     CHECK(deque.size() == 2);
00023
00024     deque.pop_front();
00025     CHECK(deque.front() == 2);
00026     CHECK(deque.size() == 1);
00027
00028     deque.front() += 3;
00029     CHECK(deque.front() == 5);
00030
00031     deque.push_back(0);
00032     deque.back() -= 2;
00033     CHECK(deque.back() == -2);
00034 }
00035
00036 constexpr std::array<int, 3> list{1, 2, 3};
00037
00038 inline __attribute__((always_inline)) void check_match(core::Deque<int> deque) {
00039     for (int elem : list) {
00040         CHECK(deque.front() == elem);
00041         deque.pop_front();
00042     }
00043 }
00044
00045 TEST_CASE("core::Deque special member functions") {
00046     std::initializer_list<int> init_list{1, 2, 3};
00047
00048     SUBCASE("core::Deque(std::initializer_list<T>)" ) {
00049         core::Deque<int> deque{init_list};
00050         check_match(deque);
00051     }
00052
00053     SUBCASE("core::Deque(const core::Deque&)" ) {
00054         core::Deque<int> deque1{init_list};
00055         core::Deque<int> deque2{deque1}; // NOLINT
00056
00057         check_match(deque2);
00058         check_match(deque1);
00059     }
00060
00061     SUBCASE("core::Deque& operator=(const core::Deque&) (single)" ) {
00062         core::Deque<int> deque1{init_list};
00063         core::Deque<int> deque2 = deque1; // NOLINT
00064
00065         check_match(deque2);
00066         check_match(deque1);
00067     }
00068
00069     SUBCASE("core::Deque& operator=(const core::Deque&) (multiple)" ) {
00070         core::Deque<int> deque1{init_list};
00071         core::Deque<int> deque2;
00072         core::Deque<int> deque3;
00073         deque3 = deque2 = deque1;
00074
00075         check_match(deque3);
00076         check_match(deque2);
00077         check_match(deque1);
00078     }
00079
00080     SUBCASE("core::Deque(core::Deque&& rhs)" ) {
00081         {
00082             core::Deque<int> deque1{core::Deque<int>{init_list}};

```

```

00083         check_match(deque1);
00084     }
00085     {
00086         core::Deque<int> deque1{init_list};
00087         core::Deque<int> deque2{std::move(deque1)};
00088         check_match(deque2);
00089         CHECK(deque1.empty()); // NOLINT
00090     }
00091 }
00092
00093 SUBCASE("core::Deque& operator=(core::Deque&& rhs)") {
00094     {
00095         core::Deque<int> deque1{1, 2, 3};
00096         core::Deque<int> deque2 = std::move(deque1);
00097
00098         check_match(deque2);
00099         CHECK(deque1.empty()); // NOLINT
00100     }
00101     {
00102         core::Deque<int> deque{init_list};
00103         deque = std::move(deque);
00104         check_match(deque); // NOLINT
00105     }
00106 }
00107 }

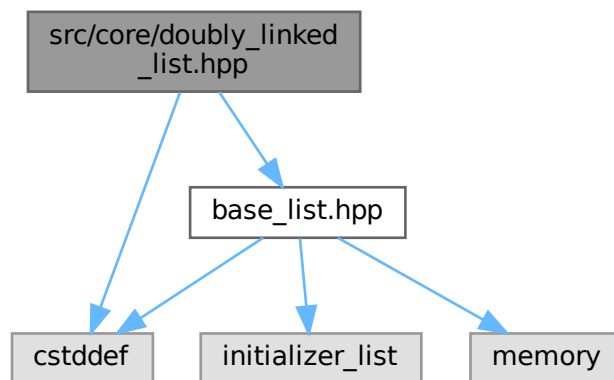
```

7.33 src/core/doubly_linked_list.hpp File Reference

```
#include <cstddef>
```

```
#include "base_list.hpp"
```

Include dependency graph for doubly_linked_list.hpp:



[illegible]

- class `core::DoublyLinkedList< T >`

- namespace **core**

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

Generated by Doxygen

```

00042
00043     void clear();
00044 };
00045
00046 template<typename T>
00047 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::internal_search(
00048     const T& elem) {
00049     Node_ptr ptr{m_head};
00050
00051     while (ptr != nullptr) {
00052         if (ptr->data == elem) {
00053             break;
00054         }
00055
00056         ptr = ptr->next;
00057     }
00058
00059     return ptr;
00060 }
00061
00062 template<typename T>
00063 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::internal_find(
00064     std::size_t index) {
00065     Node_ptr ptr{m_head};
00066     std::size_t pos = 0;
00067
00068     while (ptr != nullptr && pos < index) {
00069         ptr = ptr->next;
00070         ++pos;
00071     }
00072
00073     return ptr;
00074 }
00075
00076 template<typename T>
00077 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::search(
00078     const T& elem) {
00079     return internal_search(elem);
00080 }
00081
00082 template<typename T>
00083 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::find(
00084     std::size_t index) {
00085     return internal_find(index);
00086 }
00087
00088 template<typename T>
00089 typename DoublyLinkedList<T>::cNode_ptr DoublyLinkedList<T>::search(
00090     const T& elem) const {
00091     return internal_search(elem);
00092 }
00093
00094 template<typename T>
00095 typename DoublyLinkedList<T>::cNode_ptr DoublyLinkedList<T>::find(
00096     std::size_t index) const {
00097     return internal_find(index);
00098 }
00099
00100 template<typename T>
00101 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::insert(
00102     std::size_t index, const T& elem) {
00103     if (index == 0) {
00104         Base::push_front(elem);
00105         return m_head;
00106     }
00107
00108     if (index >= m_size) {
00109         Base::push_back(elem);
00110         return m_tail;
00111     }
00112
00113     Node_ptr ptr = find(index);
00114     auto new_node = new Node(elem, ptr->prev, ptr);
00115
00116     ptr->prev->next = new_node;
00117     ptr->prev = new_node;
00118     ++m_size;
00119
00120     return new_node;
00121 }
00122
00123 template<typename T>
00124 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::remove(
00125     std::size_t index) {
00126     if (index >= m_size) {
00127         return nullptr;
00128     }

```

```

00129
00130     if (index == 0) {
00131         Base::pop_front();
00132         return m_head;
00133     }
00134
00135     if (index + 1 == m_size) {
00136         Base::pop_back();
00137         return nullptr;
00138     }
00139
00140     Node_ptr ptr = find(index);
00141     Node_ptr ret = ptr->next;
00142
00143     ptr->next->prev = ptr->prev;
00144     ptr->prev->next = ptr->next;
00145
00146     delete ptr;
00147     --m_size;
00148
00149     return ret;
00150 }
00151
00152 template<typename T>
00153 T& DoublyLinkedList<T>::at(std::size_t index) {
00154     return find(index)->data;
00155 }
00156
00157 template<typename T>
00158 T DoublyLinkedList<T>::at(std::size_t index) const {
00159     return find(index)->data;
00160 }
00161
00162 template<typename T>
00163 void DoublyLinkedList<T>::clear() {
00164     while (!empty()) {
00165         Base::pop_front();
00166     }
00167 }
00168
00169 } // namespace core
00170
00171 #endif // CORE_DOUBLY_LINKED_LIST_HPP_

```

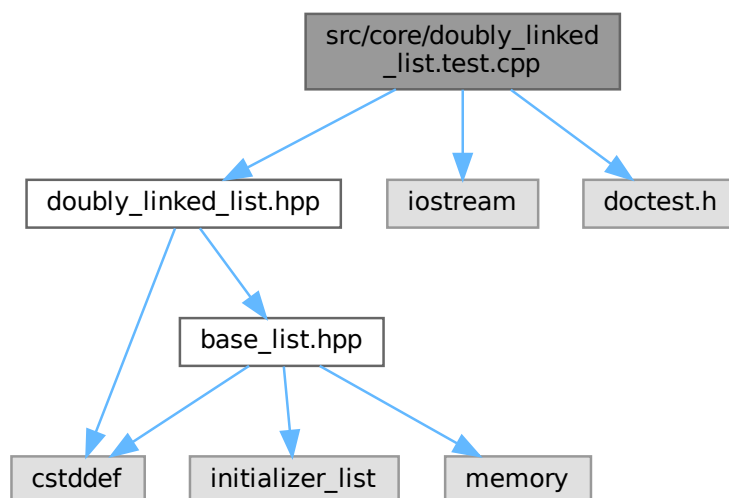
7.35 src/core/doubly_linked_list.test.cpp File Reference

```

#include "doubly_linked_list.hpp"
#include <iostream>
#include "doctest.h"

```

Include dependency graph for doubly_linked_list.test.cpp:



Functions

- [TEST_CASE](#) ("core::DoublyLinkedList functionality")

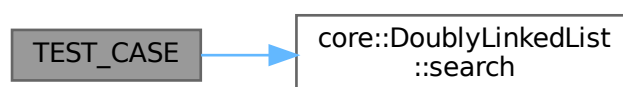
7.35.1 Function Documentation

7.35.1.1 TEST_CASE()

```
TEST_CASE (
    "core::DoublyLinkedList functionality" )
```

Definition at line 7 of file [doubly_linked_list.test.cpp](#).

Here is the call graph for this function:



7.36 doubly_linked_list.test.cpp

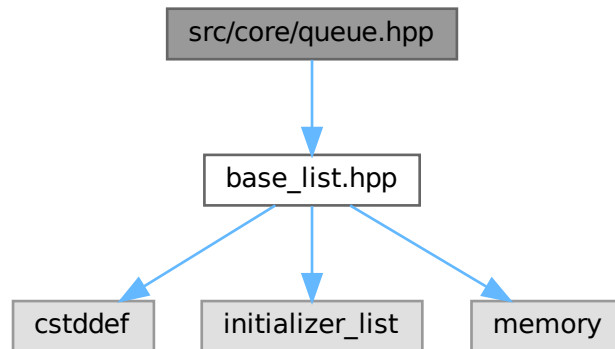
[Go to the documentation of this file.](#)

```
00001 #include "doubly_linked_list.hpp"
00002
00003 #include <iostream>
00004
00005 #include "doctest.h"
00006
00007 TEST_CASE("core::DoublyLinkedList functionality") {
00008     // List: {1, 2, 3}
00009     SUBCASE("Node_ptr search(const T& elem)") {
00010         core::DoublyLinkedList<int> dll{1, 2, 3};
00011         CHECK(dll.search(4) == nullptr);
00012         CHECK(dll.search(3)->data == 3);
00013     }
00014
00015     // List: {1, 2, 3}
00016     SUBCASE("Node_ptr find(std::size_t index)") {
00017         core::DoublyLinkedList<int> dll{1, 2, 3};
00018         CHECK(dll.find(8) == nullptr);
00019
00020         auto* ptr1 = dll.search(3);
00021         auto* ptr2 = dll.find(1);
00022
00023         CHECK(ptr1->data == 3);
00024         CHECK(ptr2->data == 2);
00025
00026         CHECK(ptr1->prev == ptr2);
00027         CHECK(ptr2->next == ptr1);
00028     }
00029
00030     SUBCASE("Node_ptr insert(std::size_t index, const T& elem)") {
00031         core::DoublyLinkedList<int> dll{1, 2, 3};
00032         auto* ptr0 = dll.search(1);
00033
00034         // List: {-1, 1, 2, 3}
00035         auto* ptr = dll.insert(0, -1);
00036
00037         CHECK(dll.size() == 4);
00038         CHECK(ptr->next == ptr0);
00039
00040         auto* ptrN = dll.search(3);
00041         // List: {-1, 1, 2, 3, 4}
00042         ptr = dll.insert(4, 4);
00043
00044         CHECK(dll.size() == 5);
00045         CHECK(ptr->prev == ptrN);
00046
00047         // List: {-1, 1, 20, 2, 3, 4}
00048         ptr = dll.insert(2, 20); // NOLINT
00049         CHECK(ptr->prev == dll.find(1));
00050         CHECK(ptr->next == dll.find(3));
00051         CHECK(dll.size() == 6);
00052
00053         // List: {-1, 1, 20, 2, 3, 4, 69}
00054         dll.insert(69, 69); // NOLINT
00055         CHECK(dll.search(69) == dll.find(69));
00056         CHECK(dll.size() == 7);
00057     }
00058
00059     // List: {-1, 1, 20, 2, 3, 4, 69}
00060     SUBCASE("Node_ptr remove(std::size_t index)") {
00061         core::DoublyLinkedList<int> dll{-1, 1, 20, 2, 3, 4, 69}; // NOLINT
00062
00063         CHECK(dll.remove(1000) == nullptr);
00064         CHECK(dll.size() == 7);
00065
00066         // List: {-1, 1, 20, 2, 3, 4}
00067         CHECK(dll.remove(6) == nullptr);
00068         CHECK(dll.size() == 6);
00069
00070         // List: {1, 20, 2, 3, 4}
00071         auto* ptr = dll.remove(0);
00072         CHECK(dll.size() == 5);
00073         CHECK(ptr->data == 1);
00074
00075         // List: {1, 2, 3, 4}
00076         ptr = dll.remove(1);
00077         CHECK(dll.size() == 4);
00078         CHECK(ptr->data == 2);
00079     }
00080 }
```

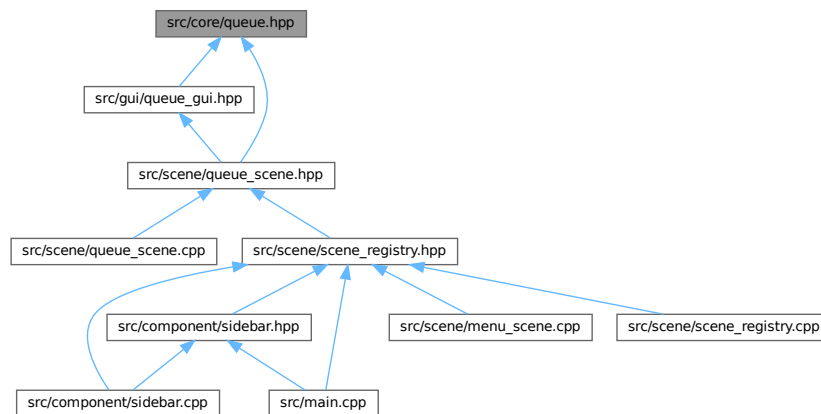
7.37 src/core/queue.hpp File Reference

```
#include "base_list.hpp"
```

Include dependency graph for queue.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class `core::Queue< T >`

Namespaces

- namespace `core`

7.38 queue.hpp

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

```

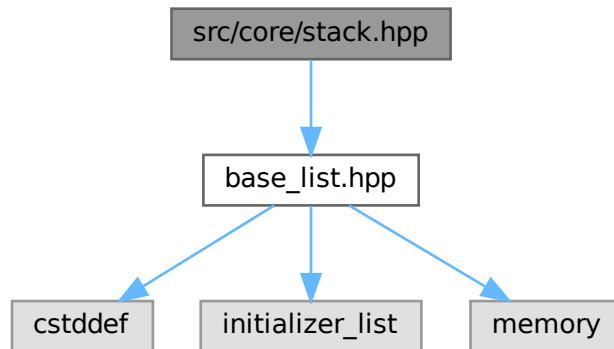
00001 #ifndef CORE_QUEUE_HPP_
00002 #define CORE_QUEUE_HPP_
00003
00004 #include "base_list.hpp"
00005
00006 namespace core {
00007
00008 template<typename T>
00009 class Queue : public BaseList<T> {
00010 private:
00011     using Base = BaseList<T>;
00012
00013 public:
00014     using Base::Base;
00015
00016     using Base::empty;
00017     using Base::size;
00018
00019     // for animation purpose only, not for real use
00020     using Base::pop_back;
00021     using Base::push_front;
00022
00023     T& front() const;
00024     T& back() const;
00025
00026     void push(const T& elem);
00027     void pop();
00028 };
00029
00030 template<typename T>
00031 T& Queue<T>::front() const {
00032     return Base::front();
00033 }
00034
00035 template<typename T>
00036 T& Queue<T>::back() const {
00037     return Base::back();
00038 }
00039
00040 template<typename T>
00041 void Queue<T>::push(const T& elem) {
00042     Base::push_back(elem);
00043 }
00044
00045 template<typename T>
00046 void Queue<T>::pop() {
00047     Base::pop_front();
00048 }
00049
00050 } // namespace core
00051
00052 #endif // CORE_QUEUE_HPP_

```

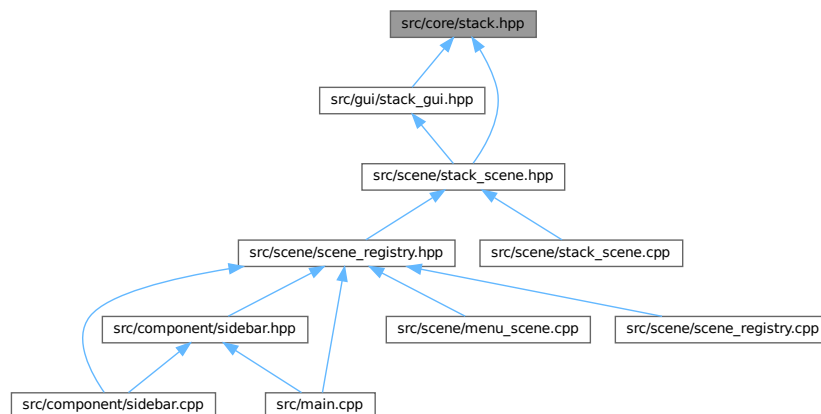
7.39 src/core/stack.hpp File Reference

```
#include "base_list.hpp"
```

Include dependency graph for stack.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class `core::Stack< T >`

Namespaces

- namespace `core`

7.40 stack.hpp

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

```

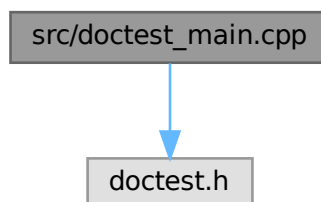
00001 #ifndef CORE_STACK_HPP_
00002 #define CORE_STACK_HPP_
00003
00004 #include "base_list.hpp"
00005
00006 namespace core {
00007
00008 template<typename T>
00009 class Stack : public BaseList<T> {
00010 protected:
00011     using Base = BaseList<T>;
00012     using Base::m_head;
00013     using Base::m_tail;
00014
00015 public:
00016     using Base::Base;
00017
00018     using Base::empty;
00019     using Base::size;
00020
00021     T& top() const;
00022
00023     void push(const T& elem);
00024     void pop();
00025 };
00026
00027 template<typename T>
00028 T& Stack<T>::top() const {
00029     return Base::front();
00030 }
00031
00032 template<typename T>
00033 void Stack<T>::push(const T& elem) {
00034     Base::push_front(elem);
00035 }
00036
00037 template<typename T>
00038 void Stack<T>::pop() {
00039     Base::pop_front();
00040 }
00041
00042 } // namespace core
00043
00044 #endif // CORE_STACK_HPP_

```

7.41 src/doctest_main.cpp File Reference

```
#include "doctest.h"
```

Include dependency graph for doctest_main.cpp:



Macros

- `#define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN`

7.41.1 Macro Definition Documentation

7.41.1.1 DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN

```
#define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN
```

Definition at line 1 of file [doctest_main.cpp](#).

7.42 doctest_main.cpp

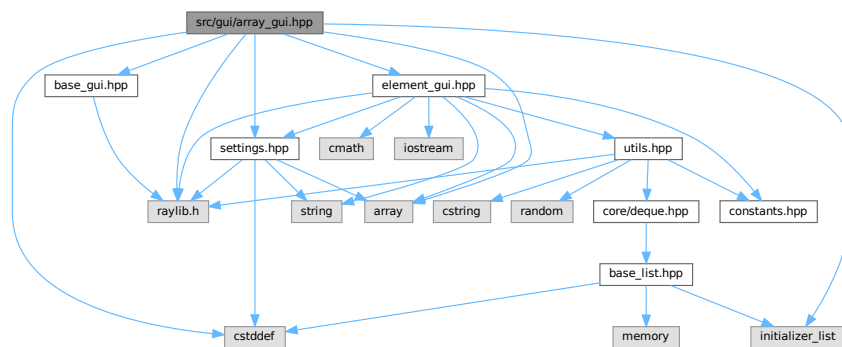
[Go to the documentation of this file.](#)

```
00001 #define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN
00002 #include "doctest.h"
```

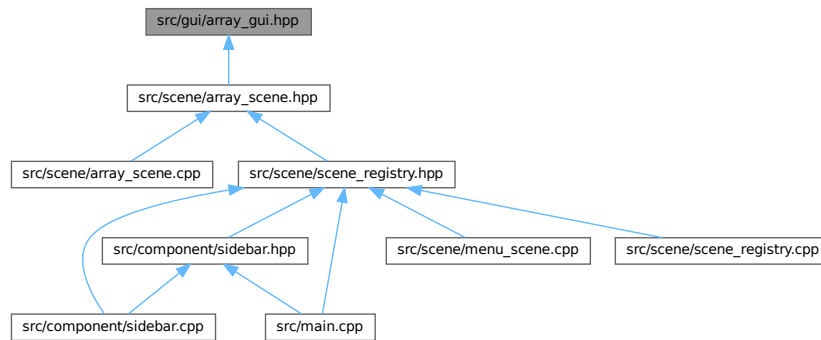
7.43 src/gui/array_gui.hpp File Reference

```
#include <array>
#include <cstddef>
#include <initializer_list>
#include "base_gui.hpp"
#include "element_gui.hpp"
#include "raylib.h"
#include "settings.hpp"
```

Include dependency graph for array_gui.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class `gui::GuiArray< T, N >`

Namespaces

- namespace `gui`

7.44 array_gui.hpp

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

```

00001 #ifndef GUI_ARRAY_GUI_HPP_
00002 #define GUI_ARRAY_GUI_HPP_
00003
00004 #include <array>
00005 #include <cstdint>
00006 #include <initializer_list>
00007
00008 #include "base_gui.hpp"
00009 #include "element_gui.hpp"
00010 #include "raylib.h"
00011 #include "settings.hpp"
00012
00013 namespace gui {
00014
00015 template<typename T, std::size_t N>
00016 class GuiArray : public internal::Base {
00017 private:
00018     static constexpr Vector2 head_pos{
00019         constants::scene_width / 2.0F - 15 * GuiElement<T>::side,
00020         constants::scene_height / 2.0F};
00021
00022     std::array<GuiElement<T>, N> m_array{};
00023
00024     void render_link(Vector2 src, Vector2 dest) override;
00025
00026 public:
00027     GuiArray();
00028     GuiArray(std::array<GuiElement<T>, N>&& init_list);
00029     void update() override;
00030     void render() override;
00031
00032     T& operator[](std::size_t idx);
00033     T operator[](std::size_t idx) const;
00034
00035     void set_color_index(std::size_t idx, int color_index);
00036 };
  
```

```

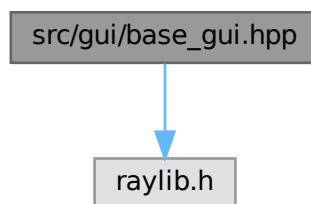
00037
00038 template<typename T, std::size_t N>
00039 GuiArray<T, N>::GuiArray() {
00040     for (std::size_t i = 0; i < N; ++i) {
00041         m_array[i] = GuiElement<T>{0, i};
00042         m_array[i].set_color_index(0);
00043     }
00044 }
00045
00046 template<typename T, std::size_t N>
00047 GuiArray<T, N>::GuiArray(std::array<GuiElement<T>, N>&& init_list)
00048     : m_array{init_list} {}
00049
00050 template<typename T, std::size_t N>
00051 void GuiArray<T, N>::render_link(Vector2 src, Vector2 dest) {}
00052
00053 template<typename T, std::size_t N>
00054 void GuiArray<T, N>::render() {
00055     update();
00056
00057     for (std::size_t i = 0; i < N; ++i) {
00058         m_array[i].render();
00059     }
00060 }
00061
00062 template<typename T, std::size_t N>
00063 void GuiArray<T, N>::update() {
00064     // TODO: if not outdated then return
00065
00066     for (std::size_t i = 0; i < N; ++i) {
00067         m_array[i].set_pos(
00068             {head_pos.x + 4 * GuiElement<T>::side * i, head_pos.y});
00069     }
00070 }
00071
00072 template<typename T, std::size_t N>
00073 T& GuiArray<T, N>::operator[](std::size_t idx) {
00074     return m_array[idx].get_value();
00075 }
00076
00077 template<typename T, std::size_t N>
00078 T GuiArray<T, N>::operator[](std::size_t idx) const {
00079     return m_array[idx].get_value();
00080 }
00081
00082 template<typename T, std::size_t N>
00083 void GuiArray<T, N>::set_color_index(std::size_t idx, int color_index) {
00084     m_array[idx].set_color_index(color_index);
00085 }
00086
00087 } // namespace gui
00088
00089 #endif // GUI_ARRAY_GUI_HPP_

```

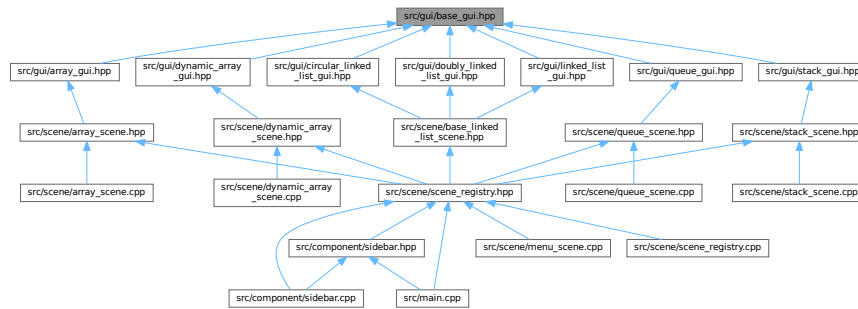
7.45 src/gui/base_gui.hpp File Reference

#include "raylib.h"

Include dependency graph for base_gui.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [gui::internal::Base](#)

Namespaces

- namespace [gui](#)
- namespace [gui::internal](#)

7.46 base_gui.hpp

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

```

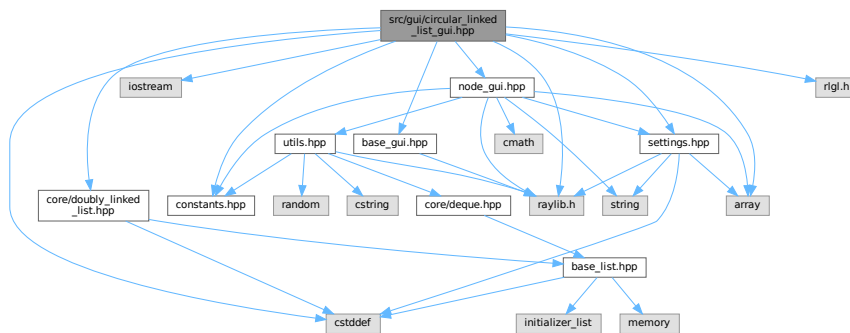
00001 #ifndef GUI_BASE_GUI_HPP_
00002 #define GUI_BASE_GUI_HPP_
00003
00004 #include "raylib.h"
00005
00006 namespace gui::internal {
00007
00008 class Base {
00009     virtual void render_link(Vector2 src, Vector2 dest) = 0;
00010
00011 public:
00012     Base() = default;
00013     Base(const Base&) = default;
00014     Base(Base&&) = default;
00015     Base& operator=(const Base&) = default;
00016     Base& operator=(Base&&) = default;
00017
00018     virtual ~Base() = default;
00019
00020     virtual void update() = 0;
00021     virtual void render() = 0;
00022 };
00023
00024 } // namespace gui::internal
00025
00026 #endif // GUI_BASE_GUI_HPP_

```

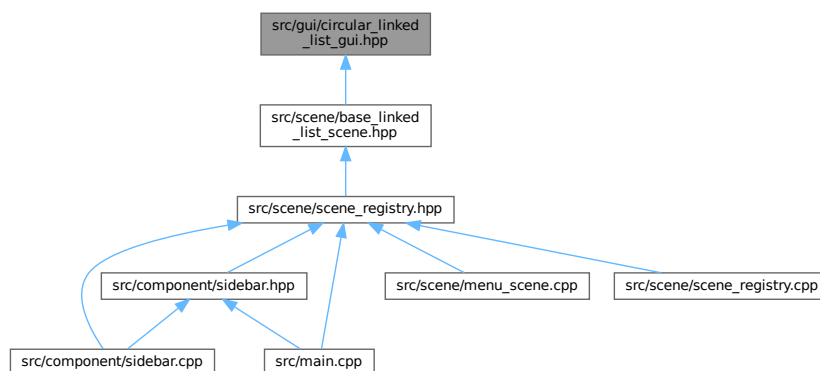
7.47 src/gui/circular_linked_list_gui.hpp File Reference

```
#include <array>
#include <cstdint>
#include <iostream>
#include "base_gui.hpp"
#include "constants.hpp"
#include "core/doubly_linked_list.hpp"
#include "node_gui.hpp"
#include "raylib.h"
#include "rlgl.h"
#include "settings.hpp"
```

Include dependency graph for circular_linked_list_gui.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [gui::GuiCircularLinkedList< T >](#)

Namespaces

- namespace [gui](#)

7.48 circular_linked_list_gui.hpp

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

```

00001 #ifndef GUI_CIRCULAR_LINKED_LIST_GUI_HPP_
00002 #define GUI_CIRCULAR_LINKED_LIST_GUI_HPP_
00003
00004 #include <array>
00005 #include <cstdint>
00006 #include <iostream>
00007
00008 #include "base_gui.hpp"
00009 #include "constants.hpp"
00010 #include "core/doubly_linked_list.hpp"
00011 #include "node_gui.hpp"
00012 #include "raylib.h"
00013 #include "rlgl.h"
00014 #include "settings.hpp"
00015
00016 namespace gui {
00017
00018 template<typename T>
00019 class GuiCircularLinkedList : public core::DoublyLinkedList<GuiNode<T>,
00020                                     public internal::Base {
00021 private:
00022     using Base = core::DoublyLinkedList<GuiNode<T>>;
00023
00024     static constexpr Vector2 head_pos{
00025         constants::scene_width / 2.0F - 15 * GuiNode<T>::radius,
00026         constants::scene_height / 2.0F};
00027
00028     using Base::m_head;
00029     using Base::m_tail;
00030
00031     void render_link(Vector2 src, Vector2 dest) override;
00032     void render_back_link();
00033
00034 public:
00035     using Base::Base;
00036
00037     using Base::empty;
00038     using Base::size;
00039
00040     GuiCircularLinkedList(std::initializer_list<GuiNode<T>> init_list);
00041
00042     void insert(std::size_t index, const T& elem);
00043
00044     void update() override;
00045     void render() override;
00046     void init_label();
00047 };
00048
00049 template<typename T>
00050 void GuiCircularLinkedList<T>::init_label() {
00051     if (m_head != nullptr) {
00052         m_head->data.set_label("head");
00053     }
00054
00055     if (m_tail != nullptr) {
00056         if (m_head == m_tail) {
00057             m_tail->data.set_label("head/tail");
00058         } else {
00059             m_tail->data.set_label("tail");
00060         }
00061     }
00062 }
00063
00064 template<typename T>
00065 GuiCircularLinkedList<T>::GuiCircularLinkedList(
00066     std::initializer_list<GuiNode<T>> init_list)
00067     : core::DoublyLinkedList<GuiNode<T>>(init_list) {
00068     init_label();
00069 }
00070
00071 template<typename T>
00072 void GuiCircularLinkedList<T>::insert(std::size_t index, const T& elem) {
00073     Base::insert(index, GuiNode{elem});
00074 }
00075
00076 template<typename T>
00077 void GuiCircularLinkedList<T>::render_link(Vector2 src, Vector2 dest) {
00078     constexpr int radius = GuiNode<T>::radius;
00079     constexpr float scaled_len = radius / 8.0F;
00080
00081     // straight line
00082     Vector2 link_pos{src.x + radius, src.y - scaled_len};

```

```

00083     Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};
00084
00085     // arrow
00086     constexpr int arrow_size = scaled_len * 5;
00087     Vector2 head{dest.x - radius + scaled_len / 2, src.y};
00088     Vector2 side_top{head.x - arrow_size, head.y - arrow_size};
00089     Vector2 side_bot{head.x - arrow_size, head.y + arrow_size};
00090
00091     // draw both
00092     const Settings& settings = Settings::get_instance();
00093     DrawRectangleV(link_pos, link_size, settings.get_color(1));
00094     DrawTriangle(head, side_top, side_bot, settings.get_color(1));
00095 }
00096
00097 template<typename T>
00098 void GuiCircularLinkedList<T>::render_back_link() {
00099     if (m_head == nullptr && m_tail == nullptr) {
00100         return;
00101     }
00102
00103     constexpr int num_points = 5;
00104     const Vector2 head_pos = m_head->data.get_pos();
00105     const Vector2 tail_pos = m_tail->data.get_pos();
00106     constexpr int radius = GuiNode<T>::radius;
00107     constexpr float scaled_len = radius / 8.0F;
00108
00109     std::array<Vector2, num_points> points{{
00110         tail_pos,
00111         {tail_pos.x + 2 * radius, tail_pos.y},
00112         {tail_pos.x + 2 * radius, tail_pos.y + 3 * radius},
00113         {head_pos.x, tail_pos.y + 3 * radius},
00114         head_pos,
00115     }};
00116
00117     constexpr int arrow_size = scaled_len * 5;
00118     Vector2 head{head_pos.x, head_pos.y + radius - scaled_len / 2};
00119     Vector2 side_left{head.x - arrow_size, head.y + arrow_size};
00120     Vector2 side_right{head.x + arrow_size, head.y + arrow_size};
00121
00122     const Settings& settings = Settings::get_instance();
00123     rlSetLineWidth(2 * scaled_len);
00124     DrawLineStrip(points.data(), num_points, settings.get_color(1));
00125     DrawTriangle(head, side_left, side_right, settings.get_color(1));
00126 }
00127
00128 template<typename T>
00129 void GuiCircularLinkedList<T>::render() {
00130     update();
00131
00132     render_back_link();
00133     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00134         if (ptr->next != nullptr) {
00135             render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
00136         }
00137         ptr->data.render();
00138     }
00139 }
00140 }
00141
00142 template<typename T>
00143 void GuiCircularLinkedList<T>::update() {
00144     // TODO: if not outdated then return
00145
00146     std::size_t pos = 0;
00147
00148     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00149         ptr->data.set_pos(
00150             {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
00151         ++pos;
00152     }
00153 }
00154
00155 } // namespace gui
00156
00157 #endif // GUI_CIRCULAR_LINKED_LIST_GUI_HPP_

```

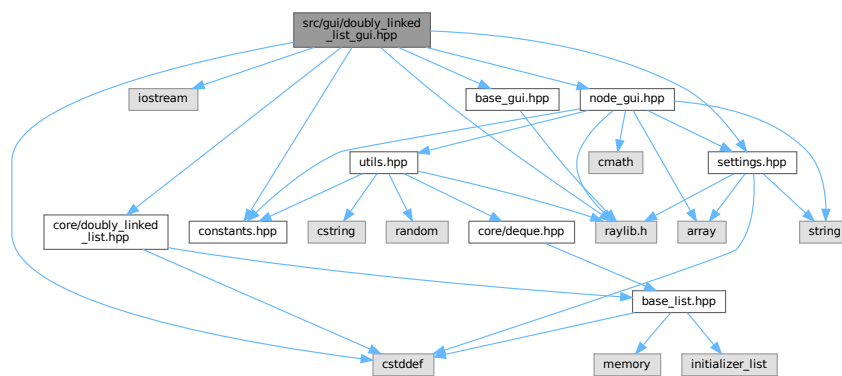
7.49 src/gui/doubly_linked_list_gui.hpp File Reference

```

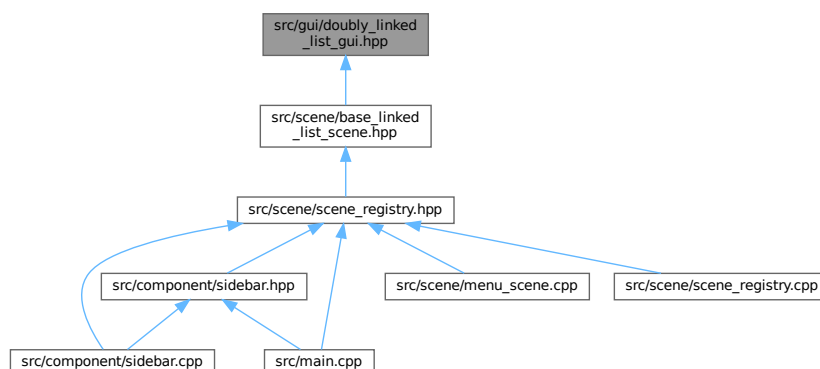
#include <cstdint>
#include <iostream>

```

```
#include "base_gui.hpp"
#include "constants.hpp"
#include "core/doubly_linked_list.hpp"
#include "node_gui.hpp"
#include "raylib.h"
#include "settings.hpp"
Include dependency graph for doubly_linked_list_gui.hpp:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [gui::GuiDoublyLinkedList< T >](#)

Namespaces

- namespace [gui](#)

7.50 doubly_linked_list_gui.hpp

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

```

00001 #ifndef GUI_DOUBLY_LINKED_LIST_GUI_HPP_
00002 #define GUI_DOUBLY_LINKED_LIST_GUI_HPP_
00003
00004 #include <cstdlib>
00005 #include <iostream>
00006
00007 #include "base_gui.hpp"
00008 #include "constants.hpp"
00009 #include "core/doubly_linked_list.hpp"
00010 #include "node_gui.hpp"
00011 #include "raylib.h"
00012 #include "settings.hpp"
00013
00014 namespace gui {
00015
00016 template<typename T>
00017 class GuiDoublyLinkedList : public core::DoublyLinkedList<GuiNode<T>,
00018                               public internal::Base {
00019 private:
00020     using Base = core::DoublyLinkedList<GuiNode<T>>;
00021
00022     static constexpr Vector2 head_pos{
00023         constants::scene_width / 2.0F - 15 * GuiNode<T>::radius,
00024         constants::scene_height / 2.0F};
00025
00026     using Base::m_head;
00027     using Base::m_tail;
00028
00029     void render_link(Vector2 src, Vector2 dest) override;
00030
00031 public:
00032     using Base::Base;
00033
00034     using Base::empty;
00035     using Base::size;
00036
00037     GuiDoublyLinkedList(std::initializer_list<GuiNode<T>> init_list);
00038
00039     void insert(std::size_t index, const T& elem);
00040
00041     void update() override;
00042     void render() override;
00043     void init_label();
00044 };
00045
00046 template<typename T>
00047 void GuiDoublyLinkedList<T>::init_label() {
00048     if (m_head != nullptr) {
00049         m_head->data.set_label("head");
00050     }
00051
00052     if (m_tail != nullptr) {
00053         if (m_head == m_tail) {
00054             m_tail->data.set_label("head/tail");
00055         } else {
00056             m_tail->data.set_label("tail");
00057         }
00058     }
00059 }
00060
00061 template<typename T>
00062 GuiDoublyLinkedList<T>::GuiDoublyLinkedList(
00063     std::initializer_list<GuiNode<T>> init_list)
00064     : core::DoublyLinkedList<GuiNode<T>>(init_list) {
00065     init_label();
00066 }
00067
00068 template<typename T>
00069 void GuiDoublyLinkedList<T>::insert(std::size_t index, const T& elem) {
00070     Base::insert(index, GuiNode{elem});
00071 }
00072
00073 template<typename T>
00074 void GuiDoublyLinkedList<T>::render_link(Vector2 src, Vector2 dest) {
00075     constexpr int radius = GuiNode<T>::radius;
00076     constexpr float scaled_len = radius / 8.0F;
00077
00078     // straight line
00079     Vector2 link_pos{src.x + radius, src.y - scaled_len};
00080     Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};
00081
00082     // right arrow

```

```

00083     constexpr int arrow_size = scaled_len * 5;
00084     Vector2 right_head{dest.x - radius + scaled_len / 2, src.y};
00085     Vector2 right_side_top{right_head.x - arrow_size,
00086                           right_head.y - arrow_size};
00087     Vector2 right_side_bot{right_head.x - arrow_size,
00088                           right_head.y + arrow_size};
00089
00090     // left arrow
00091     Vector2 left_head{src.x + radius - scaled_len / 2, src.y};
00092     Vector2 left_side_top{left_head.x + arrow_size, left_head.y - arrow_size};
00093     Vector2 left_side_bot{left_head.x + arrow_size, left_head.y + arrow_size};
00094
00095     // draw all
00096     const Settings& settings = Settings::get_instance();
00097     DrawRectangleV(link_pos, link_size, settings.get_color(1));
00098     DrawTriangle(right_head, right_side_top, right_side_bot,
00099                 settings.get_color(1));
00100     DrawTriangle(left_head, left_side_bot, left_side_top,
00101                 settings.get_color(1));
00102 }
00103
00104 template<typename T>
00105 void GuiDoublyLinkedList<T>::render() {
00106     update();
00107
00108     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00109         if (ptr->next != nullptr) {
00110             render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
00111         }
00112
00113         ptr->data.render();
00114     }
00115 }
00116
00117 template<typename T>
00118 void GuiDoublyLinkedList<T>::update() {
00119     // TODO: if not outdated then return
00120
00121     std::size_t pos = 0;
00122
00123     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00124         ptr->data.set_pos(
00125             {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
00126         ++pos;
00127     }
00128 }
00129
00130 } // namespace gui
00131
00132 #endif // GUI_DOUBLY_LINKED_LIST_GUI_HPP_

```

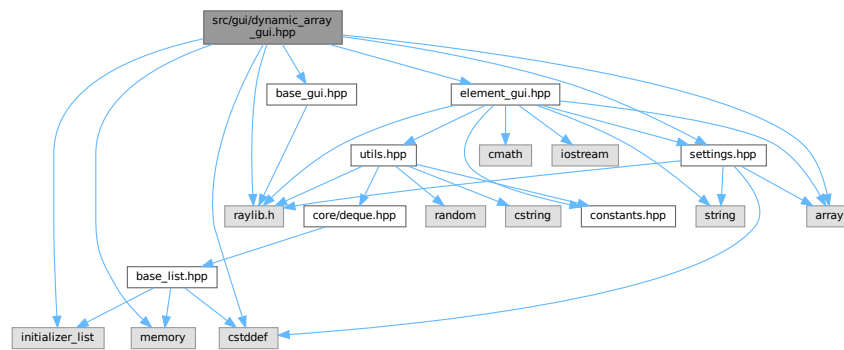
7.51 src/gui/dynamic_array_gui.hpp File Reference

```

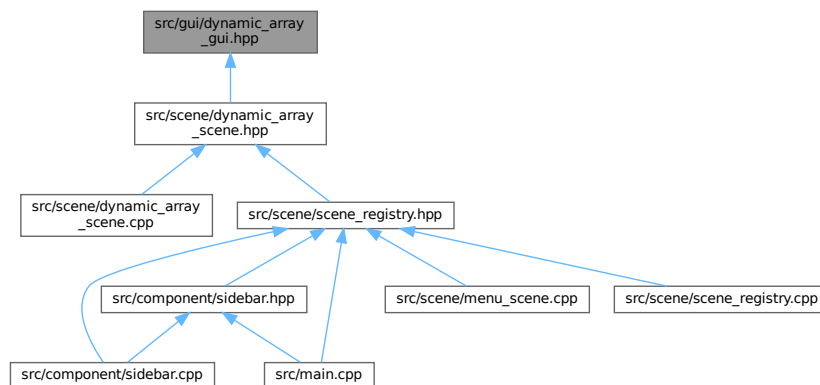
#include <array>
#include <cstdint>
#include <initializer_list>
#include <memory>
#include "base_gui.hpp"
#include "element_gui.hpp"
#include "raylib.h"
#include "settings.hpp"

```

Include dependency graph for `dynamic_array_gui.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class `gui::GuiDynamicArray< T >`

Namespaces

- namespace `gui`

7.52 dynamic_array_gui.hpp

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

```

00001 #ifndef GUI_DYNAMIC_ARRAY_GUI_HPP_
00002 #define GUI_DYNAMIC_ARRAY_GUI_HPP_
00003
00004 #include <array>
00005 #include <cstdint>
00006 #include <initializer_list>
00007 #include <memory>

```



```

00008
00009 #include "base_gui.hpp"
00010 #include "element_gui.hpp"
00011 #include "raylib.h"
00012 #include "settings.hpp"
00013
00014 namespace gui {
00015
00016 template<typename T>
00017 class GuiDynamicArray : public internal::Base {
00018 private:
00019     static constexpr Vector2 head_pos{
00020         constants::scene_width / 2.0F - 15 * GuiElement<T>::side,
00021         constants::scene_height / 2.0F};
00022
00023     std::size_t m_capacity{2};
00024     std::size_t m_size{};
00025     GuiElement<T>* m_ptr{nullptr};
00026
00027     void render_link(Vector2 src, Vector2 dest) override;
00028
00029 public:
00030     GuiDynamicArray();
00031     GuiDynamicArray(std::initializer_list<T> init_list);
00032     GuiDynamicArray(const GuiDynamicArray& other);
00033     GuiDynamicArray(GuiDynamicArray&& other) noexcept;
00034     GuiDynamicArray& operator=(const GuiDynamicArray& other);
00035     GuiDynamicArray& operator=(GuiDynamicArray&& other) noexcept;
00036     ~GuiDynamicArray() override;
00037
00038     void update() override;
00039     void render() override;
00040
00041     T& operator[](std::size_t idx);
00042     T operator[](std::size_t idx) const;
00043
00044     void set_color_index(std::size_t idx, int color_index);
00045     void realloc(std::size_t capacity);
00046
00047     std::size_t capacity() const;
00048     std::size_t size() const;
00049
00050     void push(const T& value);
00051     void pop();
00052 };
00053
00054 template<typename T>
00055 void GuiDynamicArray<T>::realloc(std::size_t capacity) {
00056     if (m_capacity > capacity) {
00057         return;
00058     }
00059
00060     while (m_capacity < capacity) {
00061         m_capacity *= 2;
00062     }
00063
00064     auto* new_ptr = new GuiElement<T>[m_capacity];
00065     for (auto i = 0; i < m_size; ++i) {
00066         new_ptr[i] = m_ptr[i];
00067     }
00068     for (auto i = m_size; i < m_capacity; ++i) {
00069         new_ptr[i].set_index(i);
00070     }
00071
00072     delete[] m_ptr;
00073     m_ptr = new_ptr;
00074 }
00075
00076 template<typename T>
00077 GuiDynamicArray<T>::GuiDynamicArray() : m_ptr(new GuiElement<T>[m_capacity]) {
00078     for (auto i = 0; i < m_capacity; ++i) {
00079         m_ptr[i].set_index(i);
00080     }
00081 }
00082
00083 template<typename T>
00084 GuiDynamicArray<T>::GuiDynamicArray(std::initializer_list<T> init_list)
00085     : m_size{init_list.size()}, m_ptr(new GuiElement<T>[m_capacity]) {
00086     realloc(m_size);
00087
00088     for (std::size_t idx = 0; auto elem : init_list) {
00089         *(m_ptr + idx).set_value(elem);
00090         *(m_ptr + idx).set_color(Settings::get_instance().get_color(0));
00091     }
00092 }
00093
00094 template<typename T>

```

```

00095 GuiDynamicArray<T>::GuiDynamicArray(const GuiDynamicArray<T>& other)
00096     : m_capacity{other.m_capacity},
00097       m_size{other.m_size},
00098       m_ptr{new GuiElement<T>[m_capacity]} {
00099     for (auto i = 0; i < m_capacity; ++i) {
00100         m_ptr[i] = other.m_ptr[i];
00101     }
00102 }
00103
00104 template<typename T>
00105 GuiDynamicArray<T>::GuiDynamicArray(GuiDynamicArray<T>&& other) noexcept
00106     : m_capacity{other.m_capacity}, m_size{other.m_size}, m_ptr{other.m_ptr} {
00107     other.m_capacity = 0;
00108     other.m_size = 0;
00109     other.m_ptr = nullptr;
00110 }
00111
00112 template<typename T>
00113 GuiDynamicArray<T>& GuiDynamicArray<T>::operator=(
00114     const GuiDynamicArray<T>& other) {
00115     if (&other != this) {
00116         m_capacity = other.m_capacity;
00117         m_size = other.m_size;
00118
00119         m_ptr = new GuiDynamicArray<T>[m_capacity];
00120         for (auto i = 0; i < m_capacity; ++i) {
00121             m_ptr[i] = other.m_ptr[i];
00122         }
00123     }
00124
00125     return *this;
00126 }
00127
00128 template<typename T>
00129 GuiDynamicArray<T>& GuiDynamicArray<T>::operator=(
00130     GuiDynamicArray&& other) noexcept {
00131     m_capacity = other.m_capacity;
00132     m_size = other.m_size;
00133     m_ptr = other.m_ptr;
00134
00135     other.m_capacity = 0;
00136     other.m_size = 0;
00137     other.m_ptr = nullptr;
00138
00139     return *this;
00140 }
00141
00142 template<typename T>
00143 GuiDynamicArray<T>::~GuiDynamicArray() {
00144     delete[] m_ptr;
00145 }
00146
00147 template<typename T>
00148 void GuiDynamicArray<T>::render_link(Vector2 src, Vector2 dest) {}
00149
00150 template<typename T>
00151 void GuiDynamicArray<T>::render() {
00152     update();
00153
00154     std::size_t idx = 0;
00155
00156     for (std::size_t i = 0; i < m_capacity; ++i) {
00157         m_ptr[i].render();
00158     }
00159 }
00160
00161 template<typename T>
00162 void GuiDynamicArray<T>::update() {
00163     // TODO: if not outdated then return
00164
00165     for (std::size_t i = 0; i < m_capacity; ++i) {
00166         m_ptr[i].set_pos(
00167             {head_pos.x + 4 * GuiElement<T>::side * i, head_pos.y});
00168     }
00169 }
00170
00171 template<typename T>
00172 T& GuiDynamicArray<T>::operator[](std::size_t idx) {
00173     return m_ptr[idx].get_value();
00174 }
00175
00176 template<typename T>
00177 T GuiDynamicArray<T>::operator[](std::size_t idx) const {
00178     return m_ptr[idx].get_value();
00179 }
00180
00181 template<typename T>

```

```

00182 void GuiDynamicArray<T>::set_color_index(std::size_t idx, int color_index) {
00183     m_ptr[idx].set_color_index(color_index);
00184 }
00185
00186 template<typename T>
00187 std::size_t GuiDynamicArray<T>::capacity() const {
00188     return m_capacity;
00189 }
00190
00191 template<typename T>
00192 std::size_t GuiDynamicArray<T>::size() const {
00193     return m_size;
00194 }
00195
00196 template<typename T>
00197 void GuiDynamicArray<T>::push(const T& value) {
00198     if (m_size == m_capacity) {
00199         realloc(m_size + 1);
00200     }
00201
00202     m_ptr[m_size].set_color_index(0);
00203     m_ptr[m_size].set_value(value);
00204     ++m_size;
00205 }
00206
00207 template<typename T>
00208 void GuiDynamicArray<T>::pop() {
00209     if (m_size >= 1) {
00210         m_ptr[m_size - 1].set_color_index(1);
00211         m_ptr[m_size - 1].set_value(0);
00212         --m_size;
00213     }
00214 }
00215
00216 } // namespace gui
00217
00218 #endif // GUI_DYNAMIC_ARRAY_GUI_HPP_

```

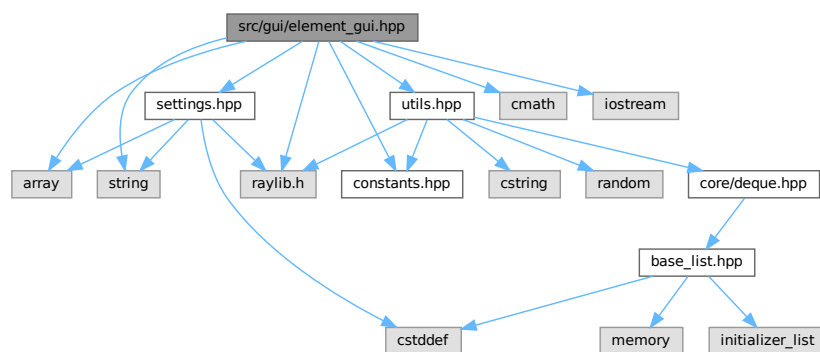
7.53 src/gui/element_gui.hpp File Reference

```

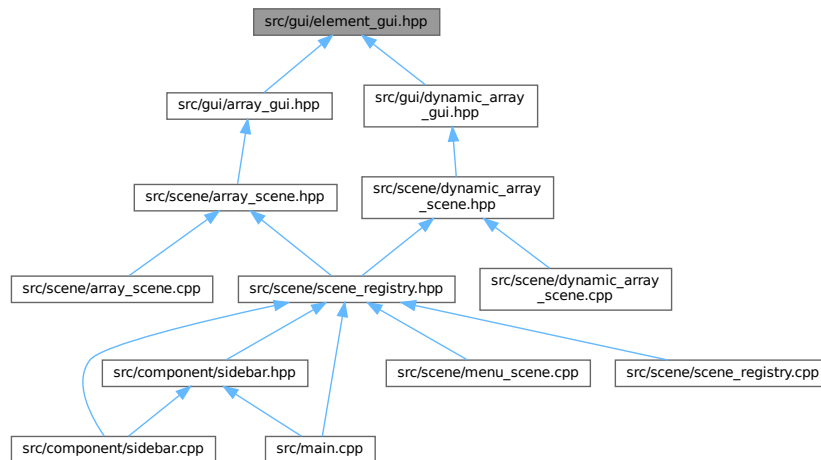
#include <array>
#include <cmath>
#include <iostream>
#include <string>
#include "constants.hpp"
#include "raylib.h"
#include "settings.hpp"
#include "utils.hpp"

```

Include dependency graph for element_gui.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class `gui::GuiElement< T >`

Namespaces

- namespace `gui`

7.54 element_gui.hpp

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

```

00001 #ifndef GUI_ELEMENT_GUI_HPP_
00002 #define GUI_ELEMENT_GUI_HPP_
00003
00004 #include <array>
00005 #include <cmath>
00006 #include <iostream>
00007 #include <string>
00008
00009 #include "constants.hpp"
00010 #include "raylib.h"
00011 #include "settings.hpp"
00012 #include "utils.hpp"
00013
00014 namespace gui {
00015
00016 template<typename T>
00017 class GuiElement {
00018 private:
00019     T m_value{};
00020     std::size_t m_index{};
00021
00022     Vector2 m_pos{init_pos};
00023     static constexpr float eps = 1e-3;
00024     int m_color_index{1};
00025
00026 public:
00027     static constexpr int side = 20;
00028     static constexpr Vector2 init_pos{
00029         constants::sidebar_width +
00030         static_cast<float>(constants::scene_width -

```

```

00031         constants::sidebar_width) /
00032         2,
00033     0};
00034
00035     GuiElement() = default;
00036     GuiElement(const T& value, std::size_t index);
00037
00038     void render();
00039     void set_pos(Vector2 pos);
00040     void set_color_index(int color_index);
00041     [[nodiscard]] Vector2 get_pos() const;
00042
00043     T& get_value();
00044     T get_value() const;
00045     void set_value(const T& value);
00046     void set_index(std::size_t index);
00047 };
00048
00049 template<typename T>
00050 GuiElement<T>::GuiElement(const T& value, std::size_t index)
00051     : m_value{value}, m_index{index} {}
00052
00053 template<typename T>
00054 void GuiElement<T>::render() {
00055     constexpr int label_font_size = 25;
00056     constexpr int label_font_spacing = 2;
00057     const std::string label = std::to_string(m_value);
00058     const std::string index = std::to_string(m_index);
00059
00060     const Vector2 label_size =
00061         utils::MeasureText(label.c_str(), label_font_size, label_font_spacing);
00062
00063     const Vector2 label_pos{m_pos.x - label_size.x / 2,
00064                             m_pos.y - label_size.y / 2};
00065
00066     const Vector2 index_size =
00067         utils::MeasureText(index.c_str(), label_font_size, label_font_spacing);
00068
00069     const Vector2 index_pos{m_pos.x - index_size.x / 2,
00070                             m_pos.y - 2 * side - index_size.y / 2};
00071
00072     const Color value_color =
00073         utils::adaptive_text_color(Settings::get_instance().get_color(0));
00074     const Color index_color = utils::adaptive_text_color(
00075         Settings::get_instance().get_color(Settings::num_color - 1));
00076
00077     DrawRectangle(m_pos.x - side, // NOLINT
00078                 m_pos.y - side, // NOLINT
00079                 2 * side, 2 * side,
00080                 Settings::get_instance().get_color(m_color_index));
00081
00082     utils::DrawText(label.c_str(), label_pos, value_color, label_font_size,
00083                     label_font_spacing);
00084
00085     utils::DrawText(index.c_str(), index_pos, index_color, label_font_size,
00086                     label_font_spacing);
00087 }
00088
00089 template<typename T>
00090 void GuiElement<T>::set_pos(Vector2 pos) {
00091     m_pos = pos;
00092 }
00093
00094 template<typename T>
00095 void GuiElement<T>::set_color_index(int color_index) {
00096     m_color_index = color_index;
00097 }
00098
00099 template<typename T>
00100 T& GuiElement<T>::get_value() {
00101     return m_value;
00102 }
00103
00104 template<typename T>
00105 T GuiElement<T>::get_value() const {
00106     return m_value;
00107 }
00108
00109 template<typename T>
00110 void GuiElement<T>::set_value(const T& value) {
00111     m_value = value;
00112 }
00113
00114 template<typename T>
00115 void GuiElement<T>::set_index(std::size_t index) {
00116     m_index = index;
00117 }

```

```

00118
00119 } // namespace gui
00120
00121 #endif // GUI_ELEMENT_GUI_HPP_

```

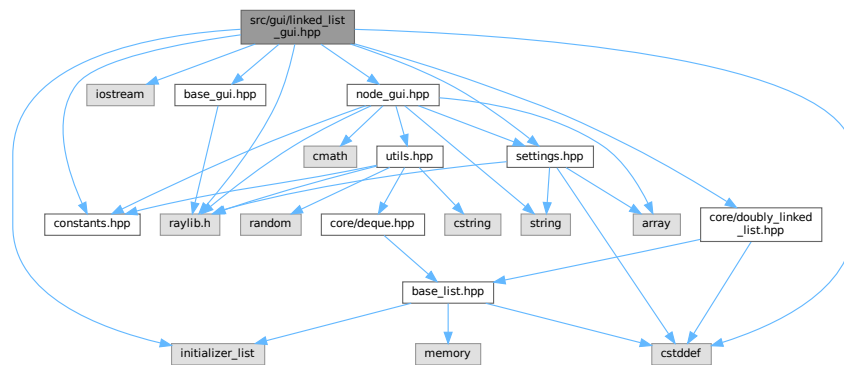
7.55 src/gui/linked_list_gui.hpp File Reference

```

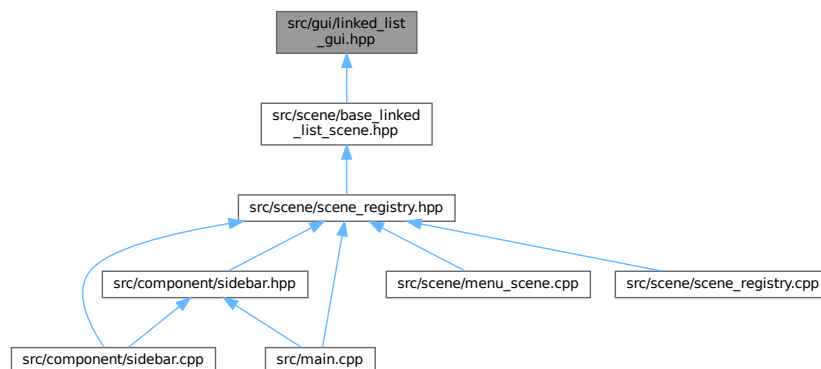
#include <cstdint>
#include <initializer_list>
#include <iostream>
#include "base_gui.hpp"
#include "constants.hpp"
#include "core/doubly_linked_list.hpp"
#include "node_gui.hpp"
#include "raylib.h"
#include "settings.hpp"

```

Include dependency graph for linked_list_gui.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class `gui::GuiLinkedList< T >`

Namespaces

- namespace `gui`

7.56 linked_list_gui.hpp

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

```

00001 #ifndef GUI_LINKED_LIST_GUI_HPP_
00002 #define GUI_LINKED_LIST_GUI_HPP_
00003
00004 #include <cstdlib>
00005 #include <initializer_list>
00006 #include <iostream>
00007
00008 #include "base_gui.hpp"
00009 #include "constants.hpp"
00010 #include "core/doubly_linked_list.hpp"
00011 #include "node_gui.hpp"
00012 #include "raylib.h"
00013 #include "settings.hpp"
00014
00015 namespace gui {
00016
00017     template<typename T>
00018     class GuiLinkedList : public core::DoublyLinkedList<GuiNode<T>,
00019                                     public internal::Base {
00020     private:
00021         using Base = core::DoublyLinkedList<GuiNode<T>>;
00022
00023         static constexpr Vector2 head_pos{
00024             constants::scene_width / 2.0F - 15 * GuiNode<T>::radius,
00025             constants::scene_height / 2.0F};
00026
00027         using Base::m_head;
00028         using Base::m_tail;
00029
00030         void render_link(Vector2 src, Vector2 dest) override;
00031
00032     public:
00033         using Base::Base;
00034
00035         using Base::empty;
00036         using Base::size;
00037
00038         GuiLinkedList(std::initializer_list<GuiNode<T>> init_list);
00039
00040         void insert(std::size_t index, const T& elem);
00041
00042         void update() override;
00043         void render() override;
00044         void init_label();
00045     };
00046
00047     template<typename T>
00048     void GuiLinkedList<T>::init_label() {
00049         if (m_head != nullptr) {
00050             m_head->data.set_label("head");
00051         }
00052
00053         if (m_tail != nullptr) {
00054             if (m_head == m_tail) {
00055                 m_tail->data.set_label("head/tail");
00056             } else {
00057                 m_tail->data.set_label("tail");
00058             }
00059         }
00060     }
00061
00062     template<typename T>
00063     GuiLinkedList<T>::GuiLinkedList(std::initializer_list<GuiNode<T>> init_list)
00064         : core::DoublyLinkedList<GuiNode<T>>(init_list) {
00065         init_label();
00066     }
00067
00068     template<typename T>
00069     void GuiLinkedList<T>::insert(std::size_t index, const T& elem) {
00070         Base::insert(index, GuiNode{elem});
00071     }
00072
00073     template<typename T>

```

```

00074 void GuiLinkedList<T>::render_link(Vector2 src, Vector2 dest) {
00075     constexpr int radius = GuiNode<T>::radius;
00076     constexpr float scaled_len = radius / 8.0F;
00077
00078     // straight line
00079     Vector2 link_pos{src.x + radius, src.y - scaled_len};
00080     Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};
00081
00082     // arrow
00083     constexpr int arrow_size = scaled_len * 5;
00084     Vector2 head{dest.x - radius + scaled_len / 2, src.y};
00085     Vector2 side_top{head.x - arrow_size, head.y - arrow_size};
00086     Vector2 side_bot{head.x - arrow_size, head.y + arrow_size};
00087
00088     // draw both
00089     DrawRectangleV(link_pos, link_size, Settings::get_instance().get_color(1));
00090     DrawTriangle(head, side_top, side_bot,
00091                 Settings::get_instance().get_color(1));
00092 }
00093
00094 template<typename T>
00095 void GuiLinkedList<T>::render() {
00096     update();
00097
00098     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00099         if (ptr->next != nullptr) {
00100             render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
00101         }
00102
00103         ptr->data.render();
00104     }
00105 }
00106
00107 template<typename T>
00108 void GuiLinkedList<T>::update() {
00109     // TODO: if not outdated then return
00110
00111     std::size_t pos = 0;
00112
00113     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00114         ptr->data.set_pos(
00115             {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
00116         ++pos;
00117     }
00118 }
00119
00120 } // namespace gui
00121
00122 #endif // GUI_LINKED_LIST_GUI_HPP_

```

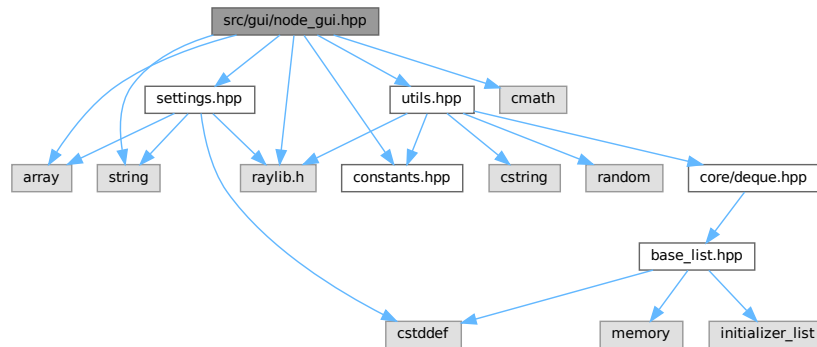
7.57 src/gui/node_gui.hpp File Reference

```

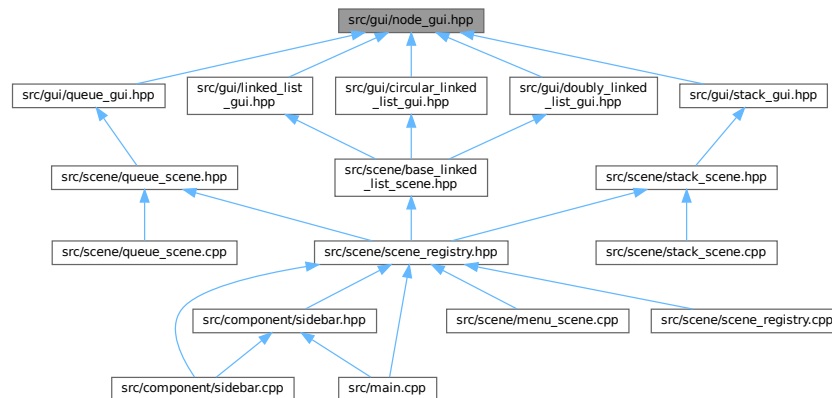
#include <array>
#include <cmath>
#include <string>
#include "constants.hpp"
#include "raylib.h"
#include "settings.hpp"
#include "utils.hpp"

```


Include dependency graph for node_gui.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class `gui::GuiNode< T >`

Namespaces

- namespace `gui`

7.58 node_gui.hpp

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

```

00001 #ifndef GUI_NODE_GUI_HPP_
00002 #define GUI_NODE_GUI_HPP_
00003
00004 #include <array>
00005 #include <cmath>

```

```

00006 #include <string>
00007
00008 #include "constants.hpp"
00009 #include "raylib.h"
00010 #include "settings.hpp"
00011 #include "utils.hpp"
00012
00013 namespace gui {
00014
00015 template<typename T>
00016 class GuiNode {
00017 private:
00018     T m_value{};
00019     int m_color_index{0};
00020
00021     Vector2 m_pos{constants::sidebar_width +
00022                  static_cast<float>(constants::scene_width -
00023                                    constants::sidebar_width) /
00024                  2,
00025                  0};
00026     static constexpr float eps = 1e-3;
00027     const char* m_label{};
00028
00029 public:
00030     static constexpr int radius = 20;
00031
00032     explicit GuiNode(const T& value);
00033
00034     void render();
00035     void set_pos(Vector2 pos);
00036     [[nodiscard]] Vector2 get_pos() const;
00037     void set_color_index(int color_index);
00038     void set_value(const T& value);
00039     T& get_value();
00040     void set_label(const char* label);
00041 };
00042
00043 template<typename T>
00044 GuiNode<T>::GuiNode(const T& value) : m_value{value} {}
00045
00046 template<typename T>
00047 void GuiNode<T>::render() {
00048     constexpr int label_font_size = 25;
00049     constexpr int label_font_spacing = 2;
00050     const std::string value = std::to_string(m_value);
00051     const Settings& settings = Settings::get_instance();
00052
00053     const Vector2 value_size =
00054         utils::MeasureText(value.c_str(), label_font_size, label_font_spacing);
00055
00056     const Vector2 value_pos{m_pos.x - value_size.x / 2,
00057                             m_pos.y - value_size.y / 2};
00058
00059     const Vector2 label_size =
00060         utils::MeasureText(m_label, label_font_size, label_font_spacing);
00061
00062     const Vector2 label_pos{m_pos.x - label_size.x / 2,
00063                             m_pos.y - 2 * label_size.y};
00064
00065     const Color value_color =
00066         utils::adaptive_text_color(Settings::get_instance().get_color(0));
00067
00068     DrawCircleV(m_pos, radius, settings.get_color(m_color_index));
00069     utils::DrawText(value.c_str(), value_pos, value_color, label_font_size,
00070                     label_font_spacing);
00071
00072     utils::DrawText(m_label, label_pos, settings.get_color(5), label_font_size,
00073                     label_font_spacing);
00074 }
00075
00076 template<typename T>
00077 void GuiNode<T>::set_color_index(int color_index) {
00078     m_color_index = color_index;
00079 }
00080
00081 template<typename T>
00082 void GuiNode<T>::set_value(const T& value) {
00083     m_value = value;
00084 }
00085
00086 template<typename T>
00087 T& GuiNode<T>::get_value() {
00088     return m_value;
00089 }
00090
00091 template<typename T>
00092 void GuiNode<T>::set_pos(Vector2 pos) {

```

```

00093     m_pos = pos;
00094 }
00095
00096 template<typename T>
00097 Vector2 GuiNode<T>::get_pos() const {
00098     return m_pos;
00099 }
00100
00101 template<typename T>
00102 void GuiNode<T>::set_label(const char* label) {
00103     m_label = label;
00104 }
00105
00106 } // namespace gui
00107
00108 #endif // GUI_NODE_GUI_HPP_

```

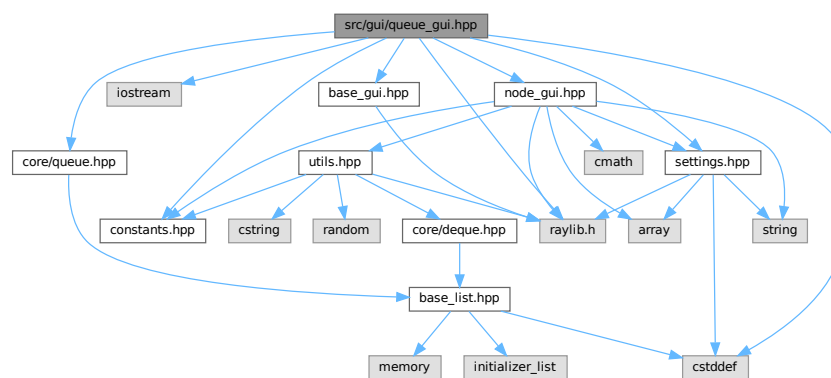
7.59 src/gui/queue_gui.hpp File Reference

```

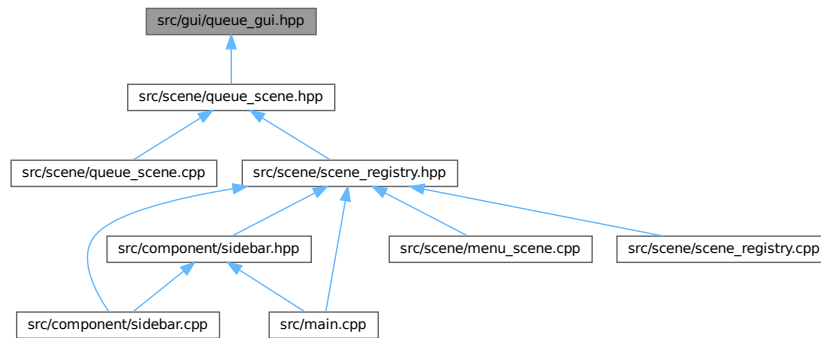
#include <cstdint>
#include <iostream>
#include "base_gui.hpp"
#include "constants.hpp"
#include "core/queue.hpp"
#include "node_gui.hpp"
#include "raylib.h"
#include "settings.hpp"

```

Include dependency graph for queue_gui.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class `gui::GuiQueue< T >`

Namespaces

- namespace `gui`

7.60 queue_gui.hpp

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

```

00001 #ifndef GUI_QUEUE_GUI_HPP_
00002 #define GUI_QUEUE_GUI_HPP_
00003
00004 #include <cstdlib>
00005 #include <iostream>
00006
00007 #include "base_gui.hpp"
00008 #include "constants.hpp"
00009 #include "core/queue.hpp"
00010 #include "node_gui.hpp"
00011 #include "raylib.h"
00012 #include "settings.hpp"
00013
00014 namespace gui {
00015
00016 template<typename T>
00017 class GuiQueue : public core::Queue<GuiNode<T>, public internal::Base {
00018 private:
00019     using Base = core::Queue<GuiNode<T>>;
00020
00021     static constexpr Vector2 head_pos{
00022         constants::scene_width / 2.0F - 15 * GuiNode<T>::radius,
00023         constants::scene_height / 2.0F};
00024
00025     using Base::m_head;
00026     using Base::m_tail;
00027
00028     void render_link(Vector2 src, Vector2 dest) override;
00029
00030 public:
00031     using Base::Base;
00032
00033     using Base::empty;
00034     using Base::size;
00035
00036     GuiQueue(std::initializer_list<GuiNode<T>> init_list);

```

```

00037
00038     void push(const T& elem);
00039     void pop();
00040
00041     // for animation purpose only, not for real use
00042     void push_front(const T& elem);
00043     void pop_back();
00044
00045     void update() override;
00046     void render() override;
00047     void init_label();
00048 };
00049
00050 template<typename T>
00051 void GuiQueue<T>::init_label() {
00052     if (m_head != nullptr) {
00053         m_head->data.set_label("head");
00054     }
00055
00056     if (m_tail != nullptr) {
00057         if (m_head == m_tail) {
00058             m_tail->data.set_label("head/tail");
00059         } else {
00060             m_tail->data.set_label("tail");
00061         }
00062     }
00063 }
00064
00065 template<typename T>
00066 GuiQueue<T>::GuiQueue(std::initializer_list<GuiNode<T>> init_list)
00067     : core::Queue<GuiNode<T>>(init_list) {
00068     init_label();
00069 }
00070
00071 template<typename T>
00072 void GuiQueue<T>::push(const T& elem) {
00073     Base::push(GuiNode<T>(elem));
00074 }
00075
00076 template<typename T>
00077 void GuiQueue<T>::pop() {
00078     Base::pop();
00079 }
00080
00081 template<typename T>
00082 void GuiQueue<T>::push_front(const T& elem) {
00083     Base::push_front(GuiNode<T>(elem));
00084 }
00085
00086 template<typename T>
00087 void GuiQueue<T>::pop_back() {
00088     Base::pop_back();
00089 }
00090
00091 template<typename T>
00092 void GuiQueue<T>::render_link(Vector2 src, Vector2 dest) {
00093     constexpr int radius = GuiNode<T>::radius;
00094     constexpr float scaled_len = radius / 8.0F;
00095
00096     // straight line
00097     Vector2 link_pos{src.x + radius, src.y - scaled_len};
00098     Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};
00099
00100     // arrow
00101     constexpr int arrow_size = scaled_len * 5;
00102     Vector2 head{dest.x - radius + scaled_len / 2, src.y};
00103     Vector2 side_top{head.x - arrow_size, head.y - arrow_size};
00104     Vector2 side_bot{head.x - arrow_size, head.y + arrow_size};
00105
00106     // draw both
00107     DrawRectangleV(link_pos, link_size, Settings::get_instance().get_color(1));
00108     DrawTriangle(head, side_top, side_bot,
00109                 Settings::get_instance().get_color(1));
00110 }
00111
00112 template<typename T>
00113 void GuiQueue<T>::render() {
00114     update();
00115
00116     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00117         if (ptr->next != nullptr) {
00118             render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
00119         }
00120         ptr->data.render();
00121     }
00122 }
00123 }

```

```

00124
00125 template<typename T>
00126 void GuiQueue<T>::update() {
00127     // TODO: if not outdated then return
00128
00129     std::size_t pos = 0;
00130
00131     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00132         ptr->data.set_pos(
00133             {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
00134         ++pos;
00135     }
00136 }
00137
00138 } // namespace gui
00139
00140 #endif // GUI_QUEUE_GUI_HPP_

```

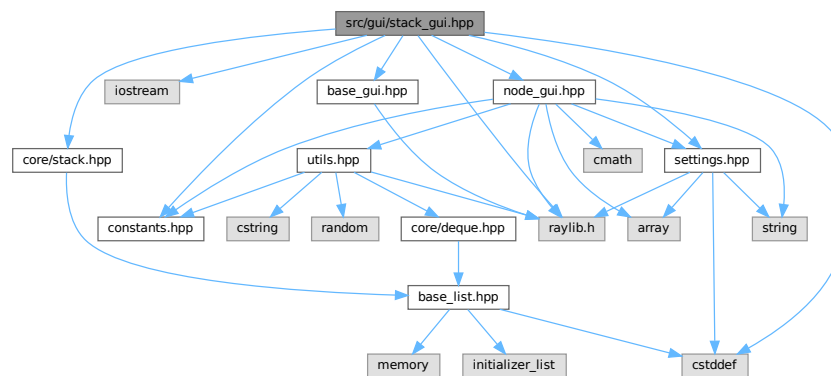
7.61 src/gui/stack_gui.hpp File Reference

```

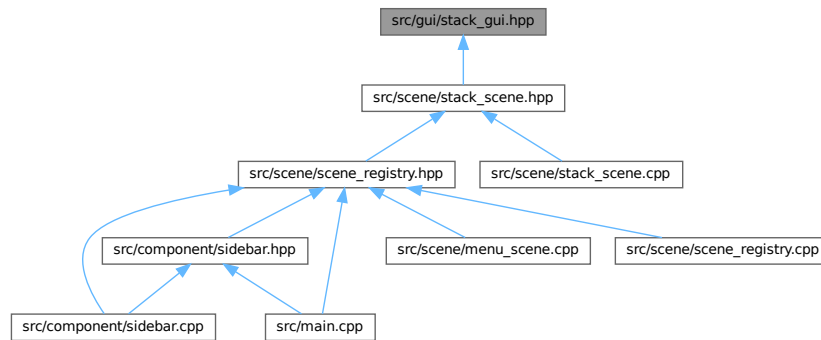
#include <cstdint>
#include <iostream>
#include "base_gui.hpp"
#include "constants.hpp"
#include "core/stack.hpp"
#include "node_gui.hpp"
#include "raylib.h"
#include "settings.hpp"

```

Include dependency graph for stack_gui.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class `gui::GuiStack< T >`

Namespaces

- namespace `gui`

7.62 stack_gui.hpp

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

```

00001 #ifndef GUI_STACK_GUI_HPP_
00002 #define GUI_STACK_GUI_HPP_
00003
00004 #include <cstdlib>
00005 #include <iostream>
00006
00007 #include "base_gui.hpp"
00008 #include "constants.hpp"
00009 #include "core/stack.hpp"
00010 #include "node_gui.hpp"
00011 #include "raylib.h"
00012 #include "settings.hpp"
00013
00014 namespace gui {
00015
00016 template<typename T>
00017 class GuiStack : public core::Stack<GuiNode<T>, public internal::Base {
00018 private:
00019     using Base = core::Stack<GuiNode<T>>;
00020
00021     static constexpr Vector2 head_pos{
00022         constants::scene_width / 2.0F - GuiNode<T>::radius / 2.0F,
00023         GuiNode<T>::radius * 4.0F};
00024
00025     using Base::m_head;
00026     using Base::m_tail;
00027
00028     void render_link(Vector2 src, Vector2 dest) override;
00029
00030 public:
00031     using Base::Base;
00032
00033     using Base::empty;
00034     using Base::size;
00035
00036     GuiStack(std::initializer_list<GuiNode<T>> init_list);

```

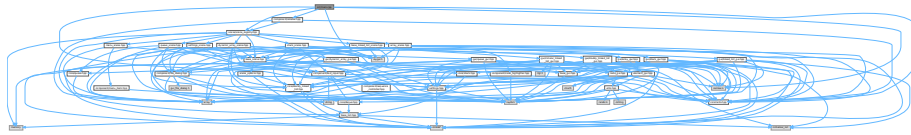
```

00037
00038     void push(const T& elem);
00039     void pop();
00040
00041     void update() override;
00042     void render() override;
00043     void init_label();
00044 };
00045
00046 template<typename T>
00047 void GuiStack<T>::init_label() {
00048     if (m_head != nullptr) {
00049         m_head->data.set_label("head");
00050     }
00051 }
00052
00053 template<typename T>
00054 GuiStack<T>::GuiStack(std::initializer_list<GuiNode<T>> init_list)
00055     : core::Stack<GuiNode<T>>(init_list) {
00056     init_label();
00057 }
00058
00059 template<typename T>
00060 void GuiStack<T>::push(const T& elem) {
00061     Base::push(GuiNode<T>(elem));
00062 }
00063
00064 template<typename T>
00065 void GuiStack<T>::pop() {
00066     Base::pop();
00067 }
00068
00069 template<typename T>
00070 void GuiStack<T>::render_link(Vector2 src, Vector2 dest) {
00071     constexpr int radius = GuiNode<T>::radius;
00072     constexpr float scaled_len = radius / 8.0F;
00073
00074     // straight line
00075     Vector2 link_pos{src.x - scaled_len, src.y + radius};
00076     Vector2 link_size{2 * scaled_len, dest.y - src.y - 2 * radius};
00077
00078     // arrow
00079     constexpr int arrow_size = scaled_len * 5;
00080     Vector2 head{src.x, dest.y - radius + scaled_len / 2};
00081     Vector2 side_left{head.x - arrow_size, head.y - arrow_size};
00082     Vector2 side_right{head.x + arrow_size, head.y - arrow_size};
00083
00084     // draw both
00085     DrawRectangleV(link_pos, link_size, Settings::get_instance().get_color(1));
00086     DrawTriangle(head, side_right, side_left,
00087         Settings::get_instance().get_color(1));
00088 }
00089
00090 template<typename T>
00091 void GuiStack<T>::render() {
00092     update();
00093
00094     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00095         if (ptr->next != nullptr) {
00096             render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
00097         }
00098
00099         ptr->data.render();
00100     }
00101 }
00102
00103 template<typename T>
00104 void GuiStack<T>::update() {
00105     // TODO: if not outdated then return
00106
00107     std::size_t pos = 0;
00108
00109     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00110         ptr->data.set_pos(
00111             {head_pos.x, head_pos.y + 4 * GuiNode<T>::radius * pos});
00112         ++pos;
00113     }
00114 }
00115
00116 } // namespace gui
00117
00118 #endif // GUI_STACK_GUI_HPP_

```


7.63 src/main.cpp File Reference

```
#include <iostream>
#include "component/sidebar.hpp"
#include "constants.hpp"
#include "raygui.h"
#include "scene/scene_registry.hpp"
#include "settings.hpp"
Include dependency graph for main.cpp:
```



Functions

- int [main](#) ()

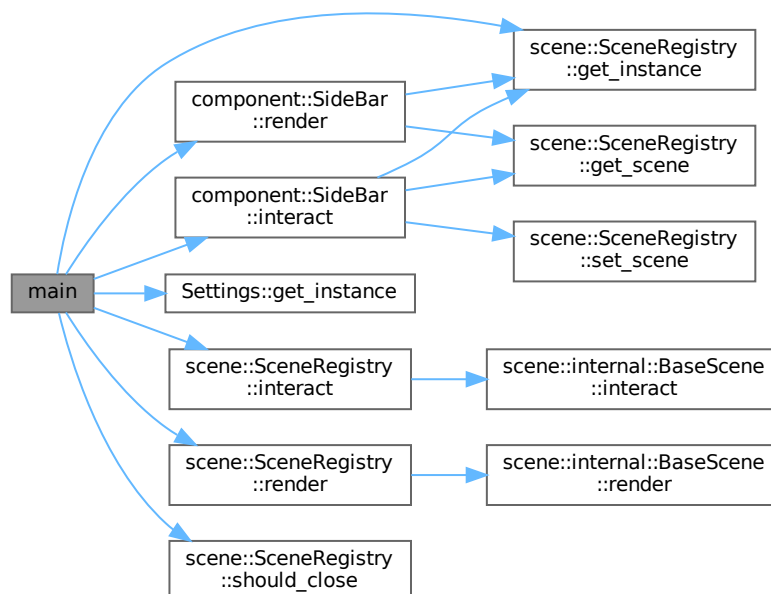
7.63.1 Function Documentation

7.63.1.1 main()

```
int main ( )
```

Definition at line 9 of file [main.cpp](#).

Here is the call graph for this function:



7.64 main.cpp

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

```

00001 #include <iostream>
00002
00003 #include "component/sidebar.hpp"
00004 #include "constants.hpp"
00005 #include "raygui.h"
00006 #include "scene/scene_registry.hpp"
00007 #include "settings.hpp"
00008
00009 int main() {
00010     InitWindow(constants::scene_width, constants::scene_height,
00011         "VisuAlgo.net clone in C++ by @jalsol");
00012     SetTargetFPS(constants::frames_per_second);
00013
00014     GuiLoadStyle("data/bluish_open_sans.rgs");
00015
00016     scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();
00017     component::SideBar sidebar;
00018
00019     bool should_close = false;
00020
00021     do {
00022         // NOTE: The order is important
00023         sidebar.interact();
00024         registry.interact();
00025
00026         BeginDrawing();
00027         {
00028             ClearBackground(
00029                 Settings::get_instance().get_color(Settings::num_color - 1));
00030
00031             // NOTE: The order is important
00032             registry.render();
00033             sidebar.render();
00034         }
00035         EndDrawing();
00036
00037         should_close = registry.should_close() || WindowShouldClose();
00038     } while (!should_close);
00039
00040     CloseWindow();
00041
00042     return 0;
00043 }

```

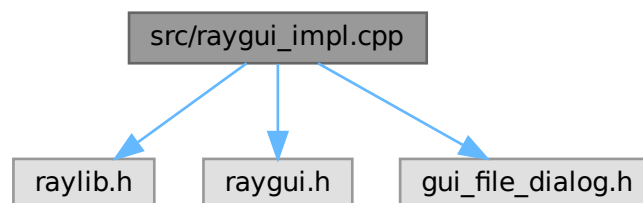
7.65 src/raygui_impl.cpp File Reference

```

#include "raylib.h"
#include "raygui.h"
#include "gui_file_dialog.h"

```

Include dependency graph for raygui_impl.cpp:



Namespaces

- namespace [scene](#)

7.68 array_scene.cpp

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

```

00001 #include "array_scene.hpp"
00002
00003 #include <cstdlib>
00004 // #include <cstdlib>
00005 // #include <cstring>
00006 #include <fstream>
00007 // #include <iostream>
00008 // #include <limits>
00009 // #include <string>
00010
00011 #include "constants.hpp"
00012 #include "raygui.h"
00013 #include "utils.hpp"
00014
00015 namespace scene {
00016
00017 void ArrayScene::render_inputs() {
00018     int& mode = scene_options.mode_selection;
00019
00020     switch (mode) {
00021         case 0: {
00022             switch (scene_options.action_selection.at(mode)) {
00023                 case 0:
00024                     break;
00025                 case 1: {
00026                     m_text_input.render(options_head, head_offset);
00027                 } break;
00028                 case 2: {
00029                     m_go = (m_file_dialog.render_head(options_head,
00030                                                         head_offset) > 0);
00031                     return;
00032                 } break;
00033                 default:
00034                     utils::unreachable();
00035             }
00036         } break;
00037
00038         case 1: {
00039             m_index_input.render(options_head, head_offset);
00040             m_text_input.render(options_head, head_offset);
00041         } break;
00042
00043         case 2: {
00044             m_text_input.render(options_head, head_offset);
00045         } break;
00046
00047         default:
00048             utils::unreachable();
00049     }
00050
00051     m_go |= render_go_button();
00052 }
00053
00054 void ArrayScene::render() {
00055     m_sequence_controller.inc_anim_counter();
00056
00057     int frame_idx = m_sequence_controller.get_anim_frame();
00058     auto* const frame_ptr = m_sequence.find(frame_idx);
00059     m_sequence_controller.set_progress_value(frame_idx);
00060
00061     if (frame_ptr != nullptr) {
00062         frame_ptr->data.render();
00063         m_code_highlighter.highlight(frame_idx);
00064     } else { // end of sequence
00065         m_array.render();
00066         m_sequence_controller.set_run_all(false);
00067     }
00068
00069     m_code_highlighter.render();
00070     m_sequence_controller.render();
00071     render_options(scene_options);
00072 }
00073

```

```

00074 void ArrayScene::interact() {
00075     if (m_sequence_controller.interact()) {
00076         m_sequence_controller.reset_anim_counter();
00077         return;
00078     }
00079
00080     m_index_input.set_random_max(max_size);
00081
00082     if (m_text_input.interact() || m_index_input.interact()) {
00083         return;
00084     }
00085
00086     if (!m_go) {
00087         return;
00088     }
00089
00090     int& mode = scene_options.mode_selection;
00091
00092     switch (mode) {
00093     case 0: {
00094         switch (scene_options.action_selection.at(mode)) {
00095             case 0: {
00096                 interact_random();
00097             } break;
00098
00099             case 1: {
00100                 interact_import(m_text_input.extract_values());
00101             } break;
00102
00103             case 2: {
00104                 interact_file_import();
00105             } break;
00106
00107             default:
00108                 utils::unreachable();
00109         }
00110     } break;
00111
00112     case 1: {
00113         interact_update();
00114     } break;
00115
00116     case 2: {
00117         interact_search();
00118     } break;
00119
00120     default:
00121         utils::unreachable();
00122     }
00123
00124     m_go = false;
00125 }
00126
00127 void ArrayScene::interact_random() {
00128     m_array = {};
00129
00130     for (std::size_t i = 0; i < max_size; ++i) {
00131         m_array[i] = utils::get_random(constants::min_val, constants::max_val);
00132     }
00133 }
00134
00135 void ArrayScene::interact_import(core::Deque<int> nums) {
00136     m_array = {};
00137     std::size_t i; // NOLINT
00138
00139     for (i = 0; i < max_size && !nums.empty(); ++i) {
00140         m_array[i] = nums.front();
00141         nums.pop_front();
00142     }
00143
00144     for (; i < max_size; ++i) {
00145         m_array[i] = 0;
00146     }
00147 }
00148
00149 void ArrayScene::interact_update() {
00150     auto index_container = m_index_input.extract_values();
00151     if (index_container.empty()) {
00152         return;
00153     }
00154
00155     auto value_container = m_text_input.extract_values();
00156     if (value_container.empty()) {
00157         return;
00158     }
00159
00160     int index = index_container.front();

```

```

00161     int value = value_container.front();
00162
00163     if (!(0 <= index && index < max_size) || !utils::val_in_range(value)) {
00164         return;
00165     }
00166
00167     m_code_highlighter.set_code({
00168         "array[index] = value;",
00169     });
00170
00171     m_sequence.clear();
00172
00173     // initial state (before update)
00174     m_sequence.insert(m_sequence.size(), m_array);
00175     m_code_highlighter.push_into_sequence(-1);
00176
00177     // highlight
00178     m_array.set_color_index(index, 2);
00179     m_sequence.insert(m_sequence.size(), m_array);
00180     m_code_highlighter.push_into_sequence(0);
00181
00182     // update
00183     m_array[index] = value;
00184     m_array.set_color_index(index, 3);
00185     m_sequence.insert(m_sequence.size(), m_array);
00186     m_code_highlighter.push_into_sequence(0);
00187
00188     // undo highlight
00189     m_array.set_color_index(index, 0);
00190
00191     m_sequence_controller.set_max_value((int)m_sequence.size());
00192     m_sequence_controller.set_rerun();
00193 }
00194
00195 void ArrayScene::interact_file_import() {
00196     interact_import(m_file_dialog.extract_values());
00197 }
00198
00199 void ArrayScene::interact_search() {
00200     auto value_container = m_text_input.extract_values();
00201     if (value_container.empty()) {
00202         return;
00203     }
00204
00205     int value = value_container.front();
00206     if (!utils::val_in_range(value)) {
00207         return;
00208     }
00209
00210     m_code_highlighter.set_code({
00211         "for (i = 0; i < size; i++)",
00212         "    if (array[i] == value)",
00213         "        return i;",
00214         "return not_found",
00215     });
00216
00217     m_sequence.clear();
00218     m_sequence.insert(m_sequence.size(), m_array);
00219     m_code_highlighter.push_into_sequence(0);
00220
00221     bool found = false;
00222
00223     for (std::size_t i = 0; i < max_size; ++i) {
00224         m_array.set_color_index(i, 3);
00225         m_sequence.insert(m_sequence.size(), m_array);
00226         m_code_highlighter.push_into_sequence(1);
00227
00228         if (m_array[i] == value) {
00229             found = true;
00230             m_array.set_color_index(i, 4);
00231             m_sequence.insert(m_sequence.size(), m_array);
00232             m_code_highlighter.push_into_sequence(2);
00233             m_array.set_color_index(i, 0);
00234             break;
00235         }
00236
00237         m_array.set_color_index(i, 0);
00238         m_sequence.insert(m_sequence.size(), m_array);
00239         m_code_highlighter.push_into_sequence(0);
00240     }
00241
00242     if (!found) {
00243         m_sequence.insert(m_sequence.size(), m_array);
00244         m_code_highlighter.push_into_sequence(3);
00245     }
00246
00247     m_sequence_controller.set_max_value((int)m_sequence.size());

```

```

00248     m_sequence_controller.set_rerun();
00249 }
00250
00251 } // namespace scene

```

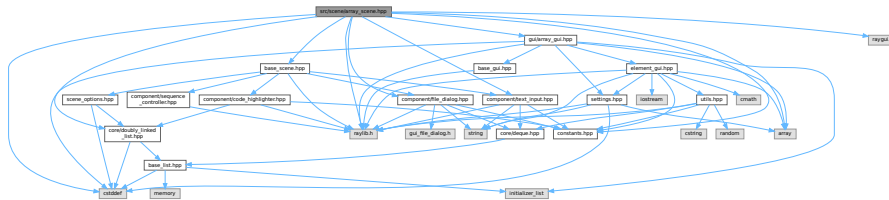
7.69 src/scene/array_scene.hpp File Reference

```

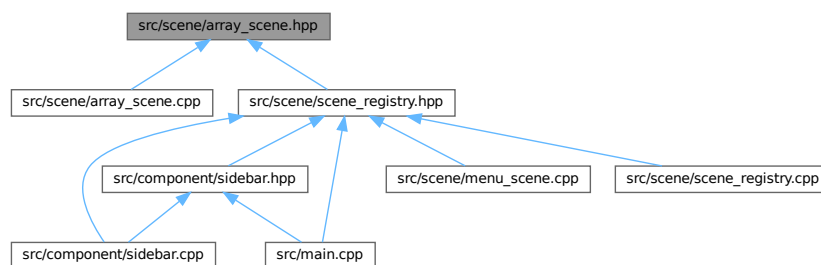
#include <array>
#include <cstddef>
#include "base_scene.hpp"
#include "component/file_dialog.hpp"
#include "component/text_input.hpp"
#include "constants.hpp"
#include "core/doubly_linked_list.hpp"
#include "gui/array_gui.hpp"
#include "raygui.h"
#include "raylib.h"

```

Include dependency graph for array_scene.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [scene::ArrayScene](#)

Namespaces

- namespace [scene](#)

7.70 array_scene.hpp

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

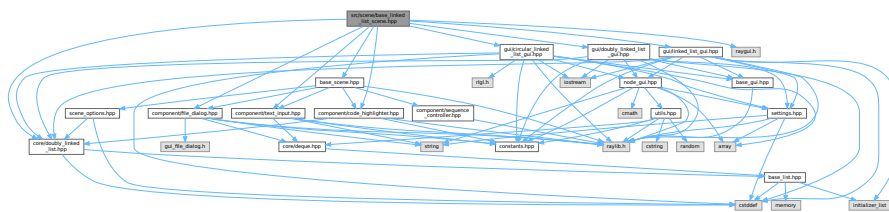
```

00001 #ifndef SCENE_ARRAY_SCENE_HPP_
00002 #define SCENE_ARRAY_SCENE_HPP_
00003
00004 #include <array>
00005 #include <cstdint>
00006
00007 #include "base_scene.hpp"
00008 #include "component/file_dialog.hpp"
00009 #include "component/text_input.hpp"
00010 #include "constants.hpp"
00011 #include "core/doubly_linked_list.hpp"
00012 #include "gui/array_gui.hpp"
00013 #include "raygui.h"
00014 #include "raylib.h"
00015
00016 namespace scene {
00017
00018 class ArrayScene : public internal::BaseScene {
00019 private:
00020     static constexpr std::size_t max_size = 8;
00021
00022     internal::SceneOptions scene_options{
00023         // max_size
00024         max_size,
00025
00026         // mode_labels
00027         "Mode: Create;",
00028         "Mode: Update;",
00029         "Mode: Search",
00030
00031         // mode_selection
00032         0,
00033
00034         // action_labels
00035         {
00036             // Mode: Create
00037             "Action: Random;",
00038             "Action: Input;",
00039             "Action: File",
00040
00041             // Mode: Update
00042             "",
00043
00044             // Mode: Search
00045             "",
00046         },
00047
00048         // action_selection
00049         core::DoublyLinkedList<int>{0, 0, 0},
00050     };
00051
00052     using internal::BaseScene::button_size;
00053     using internal::BaseScene::head_offset;
00054     using internal::BaseScene::options_head;
00055
00056     gui::GuiArray<int, max_size> m_array{};
00057     core::DoublyLinkedList<gui::GuiArray<int, max_size>> m_sequence;
00058
00059     bool m_go{};
00060
00061     using internal::BaseScene::m_code_highlighter;
00062     using internal::BaseScene::m_file_dialog;
00063     using internal::BaseScene::m_index_input;
00064     using internal::BaseScene::m_sequence_controller;
00065     using internal::BaseScene::m_text_input;
00066
00067     using internal::BaseScene::render_go_button;
00068     using internal::BaseScene::render_options;
00069     void render_inputs() override;
00070
00071     void interact_random();
00072     void interact_import(core::Deque<int> nums);
00073     void interact_file_import();
00074     void interact_update();
00075     void interact_search();
00076
00077 public:
00078     void render() override;
00079     void interact() override;
00080 };
00081
00082 } // namespace scene

```


7.71 src/scene/base_linked_list_scene.hpp File Reference

Include dependency graph for base_linked_list_scene.hpp:



```

graph BT
    src_main[src/main.cpp] --> src_sidebar[src/component/sidebar.cpp]
    src_main --> src_registry[src/scene/scene_registry.hpp]
    src_sidebar --> src_registry
    src_menu[src/scene/menu_scene.cpp] --> src_registry
    src_registry_cpp[src/scene/scene_registry.cpp] --> src_registry
    src_registry_hpp[src/scene/scene_registry.hpp] --> src_base[src/scene/base_linked_list_scene.hpp]
  
```

- class scene::BaseLinkedListScene< Con >

- namespace **scene**

Typedefs

- using `scene::LinkedListScene` = `BaseLinkedListScene< gui::GuiLinkedList< int > >`
- using `scene::DoublyLinkedListScene` = `BaseLinkedListScene< gui::GuiDoublyLinkedList< int > >`
- using `scene::CircularLinkedListScene` = `BaseLinkedListScene< gui::GuiCircularLinkedList< int > >`

7.72 base_linked_list_scene.hpp

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

```
00001 #ifndef SCENE_BASE_LINKED_LIST_SCENE_HPP_
00002 #define SCENE_BASE_LINKED_LIST_SCENE_HPP_
00003
00004 #include "base_scene.hpp"
00005 #include "component/code_highlighter.hpp"
00006 #include "component/file_dialog.hpp"
00007 #include "component/text_input.hpp"
00008 #include "core/doubly_linked_list.hpp"
00009 #include "gui/circular_linked_list_gui.hpp"
00010 #include "gui/doubly_linked_list_gui.hpp"
00011 #include "gui/linked_list_gui.hpp"
00012 #include "raygui.h"
00013
00014 namespace scene {
00015
00016 template<typename Con>
00017 class BaseLinkedListScene : public internal::BaseScene {
00018 private:
00019     internal::SceneOptions scene_options{
00020         // max_size
00021         8, // NOLINT
00022
00023         // mode_labels
00024         "Mode: Create;"
00025         "Mode: Add;"
00026         "Mode: Delete;"
00027         "Mode: Update;"
00028         "Mode: Search",
00029
00030         // mode_selection
00031         0,
00032
00033         // action_labels
00034         {
00035             // Mode: Create
00036             "Action: Random;Action: Input;Action: File",
00037             // Mode: Add
00038             "",
00039             // Mode: Delete
00040             "",
00041             // Mode: Update
00042             "",
00043             // Mode: Search
00044             ""
00045         },
00046
00047         // action_selection
00048         core::DoublyLinkedList<int>{0, 0, 0, 0, 0},
00049     };
00050
00051     using internal::BaseScene::button_size;
00052     using internal::BaseScene::head_offset;
00053     using internal::BaseScene::options_head;
00054
00055     Con m_list{
00056         gui::GuiNode<int>{1},
00057         gui::GuiNode<int>{2},
00058         gui::GuiNode<int>{3},
00059     };
00060     core::DoublyLinkedList<Con> m_sequence;
00061
00062     bool m_go{};
00063     using internal::BaseScene::m_code_highlighter;
00064     using internal::BaseScene::m_file_dialog;
00065     using internal::BaseScene::m_index_input;
00066     using internal::BaseScene::m_sequence_controller;
00067     using internal::BaseScene::m_text_input;
00068
00069     using internal::BaseScene::render_go_button;
00070     using internal::BaseScene::render_options;
```

```

00071     void render_inputs() override;
00072
00073     void interact_random();
00074     void interact_import(core::Deque<int> nums);
00075     void interact_file_import();
00076
00077     void interact_add();
00078     void interact_add_head(int value);
00079     void interact_add_tail(int value);
00080     void interact_add_middle(int index, int value);
00081
00082     void interact_delete();
00083     void interact_delete_head();
00084     void interact_delete_tail();
00085     void interact_delete_middle(int index);
00086
00087     void interact_update();
00088     void interact_search();
00089
00090 public:
00091     void render() override;
00092     void interact() override;
00093 };
00094
00095 using LinkedListScene = BaseLinkedListScene<gui::GuiLinkedList<int>>;
00096 using DoublyLinkedListScene =
00097     BaseLinkedListScene<gui::GuiDoublyLinkedList<int>>;
00098 using CircularLinkedListScene =
00099     BaseLinkedListScene<gui::GuiCircularLinkedList<int>>;
00100
00101 template<typename Con>
00102 void BaseLinkedListScene<Con>::render_inputs() {
00103     int& mode = scene_options.mode_selection;
00104
00105     switch (mode) {
00106     case 0: {
00107         switch (scene_options.action_selection.at(mode)) {
00108             case 0:
00109                 break;
00110             case 1: {
00111                 m_text_input.render(options_head, head_offset);
00112             } break;
00113             case 2: {
00114                 m_go = (m_file_dialog.render_head(options_head,
00115                                                     head_offset) > 0);
00116                 return;
00117             } break;
00118             default:
00119                 utils::unreachable();
00120         }
00121     } break;
00122
00123     case 1: {
00124         m_index_input.render(options_head, head_offset);
00125         m_text_input.render(options_head, head_offset);
00126     } break;
00127
00128     case 2: {
00129         m_index_input.render(options_head, head_offset);
00130     } break;
00131
00132     case 3: {
00133         m_index_input.render(options_head, head_offset);
00134         m_text_input.render(options_head, head_offset);
00135     } break;
00136
00137     case 4: {
00138         m_text_input.render(options_head, head_offset);
00139     } break;
00140
00141     default:
00142         utils::unreachable();
00143     }
00144
00145     m_go |= render_go_button();
00146 }
00147
00148 template<typename Con>
00149 void BaseLinkedListScene<Con>::render() {
00150     m_sequence_controller.inc_anim_counter();
00151
00152     int frame_idx = m_sequence_controller.get_anim_frame();
00153     auto* const frame_ptr = m_sequence.find(frame_idx);
00154     m_sequence_controller.set_progress_value(frame_idx);
00155
00156     if (frame_ptr != nullptr) {
00157         frame_ptr->data.render();
00158     }
00159 }

```

```

00158         m_code_highlighter.highlight(frame_idx);
00159     } else { // end of sequence
00160         m_list.render();
00161         m_sequence_controller.set_run_all(false);
00162     }
00163
00164     m_code_highlighter.render();
00165     m_sequence_controller.render();
00166     render_options(scene_options);
00167 }
00168
00169 template<typename Con>
00170 void BaseLinkedListScene<Con>::interact() {
00171     if (m_sequence_controller.interact()) {
00172         m_sequence_controller.reset_anim_counter();
00173         return;
00174     }
00175
00176     m_index_input.set_random_max((int)m_list.size() - 1);
00177
00178     if (m_text_input.interact() || m_index_input.interact()) {
00179         return;
00180     }
00181
00182     if (!m_go) {
00183         return;
00184     }
00185
00186     int& mode = scene_options.mode_selection;
00187
00188     switch (mode) {
00189     case 0: {
00190         switch (scene_options.action_selection.at(mode)) {
00191             case 0: {
00192                 interact_random();
00193             } break;
00194
00195             case 1: {
00196                 interact_import(m_text_input.extract_values());
00197             } break;
00198
00199             case 2: {
00200                 interact_file_import();
00201             } break;
00202
00203             default:
00204                 utils::unreachable();
00205         }
00206     } break;
00207
00208     case 1: {
00209         m_index_input.set_random_max((int)m_list.size());
00210         interact_add();
00211     } break;
00212
00213     case 2: {
00214         interact_delete();
00215     } break;
00216
00217     case 3: {
00218         interact_update();
00219     } break;
00220
00221     case 4: {
00222         interact_search();
00223     } break;
00224
00225     default:
00226         utils::unreachable();
00227     }
00228
00229     m_go = false;
00230 }
00231
00232 template<typename Con>
00233 void BaseLinkedListScene<Con>::interact_random() {
00234     std::size_t size =
00235         utils::get_random(std::size_t{1}, scene_options.max_size);
00236     m_list = Con();
00237
00238     for (auto i = 0; i < size; ++i) {
00239         m_list.insert(
00240             i, utils::get_random(constants::min_val, constants::max_val));
00241     }
00242     m_list.init_label();
00243 }
00244

```

```

00245 template<typename Con>
00246 void BaseLinkedListScene<Con>::interact_import(core::Deque<int> nums) {
00247     m_sequence.clear();
00248     m_list = Con();
00249
00250     while (!nums.empty()) {
00251         if (utils::val_in_range(nums.front())) {
00252             m_list.insert(m_list.size(), nums.front());
00253         }
00254         nums.pop_front();
00255     }
00256     m_list.init_label();
00257 }
00258
00259 template<typename Con>
00260 void BaseLinkedListScene<Con>::interact_file_import() {
00261     interact_import(m_file_dialog.extract_values());
00262 }
00263
00264 template<typename Con>
00265 void BaseLinkedListScene<Con>::interact_add() {
00266     auto index_container = m_index_input.extract_values();
00267     if (index_container.empty()) {
00268         return;
00269     }
00270
00271     auto value_container = m_text_input.extract_values();
00272     if (value_container.empty()) {
00273         return;
00274     }
00275
00276     int index = index_container.front();
00277     int value = value_container.front();
00278
00279     if (!(0 <= index && index <= m_list.size())) {
00280         return;
00281     }
00282
00283     if (!utils::val_in_range(value)) {
00284         return;
00285     }
00286
00287     m_sequence.clear();
00288     m_sequence.insert(m_sequence.size(), m_list);
00289
00290     if (index == 0) {
00291         interact_add_head(value);
00292     } else if (index == m_list.size()) {
00293         interact_add_tail(value);
00294     } else {
00295         interact_add_middle(index, value);
00296     }
00297
00298     m_sequence_controller.set_max_value((int)m_sequence.size());
00299     m_sequence_controller.set_rerun();
00300 }
00301
00302 template<typename Con>
00303 void BaseLinkedListScene<Con>::interact_add_head(int value) {
00304     m_code_highlighter.set_code({
00305         "Node* node = new Node(value);",
00306         "node->next = head;",
00307         "head = next;",
00308     });
00309     m_code_highlighter.push_into_sequence(-1);
00310
00311     m_list.insert(0, value);
00312
00313     m_list.at(0).set_color_index(6);
00314     m_list.at(0).set_label("node");
00315     m_sequence.insert(m_sequence.size(), m_list);
00316     m_code_highlighter.push_into_sequence(0);
00317
00318     if (m_list.size() > 1) {
00319         m_list.at(1).set_color_index(4);
00320     }
00321
00322     m_sequence.insert(m_sequence.size(), m_list);
00323     m_code_highlighter.push_into_sequence(1);
00324
00325     if (m_list.size() > 1) {
00326         m_list.at(1).set_color_index(0);
00327         m_list.at(1).set_label("");
00328     }
00329
00330     m_list.at(0).set_color_index(4);
00331     m_list.at(0).set_label("head");

```

```

00332     m_sequence.insert(m_sequence.size(), m_list);
00333     m_code_highlighter.push_into_sequence(2);
00334
00335     m_list.at(0).set_color_index(0);
00336 }
00337
00338 template<typename Con>
00339 void BaseLinkedListScene<Con>::interact_add_tail(int value) {
00340     m_code_highlighter.set_code({
00341         "Node* node = new Node(value);",
00342         "tail->next = node;",
00343         "tail = tail->next;",
00344     });
00345     m_code_highlighter.push_into_sequence(-1);
00346
00347     std::size_t size = m_list.size();
00348
00349     m_list.insert(size, value);
00350     m_list.at(size).set_color_index(6);
00351     m_sequence.insert(m_sequence.size(), m_list);
00352     m_code_highlighter.push_into_sequence(0);
00353
00354     m_list.at(size - 1).set_color_index(4);
00355     m_sequence.insert(m_sequence.size(), m_list);
00356     m_code_highlighter.push_into_sequence(1);
00357
00358     m_list.at(size - 1).set_color_index(0);
00359     m_list.at(size - 1).set_label("");
00360     m_list.at(size).set_color_index(4);
00361     m_list.at(size).set_label("tail");
00362     m_sequence.insert(m_sequence.size(), m_list);
00363     m_code_highlighter.push_into_sequence(2);
00364
00365     m_list.at(size).set_color_index(0);
00366 }
00367
00368 template<typename Con>
00369 void BaseLinkedListScene<Con>::interact_add_middle(int index, int value) {
00370     m_code_highlighter.set_code({
00371         "Node* pre = head;",
00372         "for (i = 0; i < index - 1; ++i)",
00373         "    pre = pre->next;",
00374         "",
00375         "Node* nxt = pre->next;",
00376         "Node* node = new Node(value);",
00377         "node->next = nxt;",
00378         "pre->next = node;",
00379     });
00380     m_code_highlighter.push_into_sequence(-1);
00381
00382     m_list.at(0).set_color_index(4);
00383     m_list.at(0).set_label("head/pre");
00384     m_sequence.insert(m_sequence.size(), m_list);
00385     m_code_highlighter.push_into_sequence(0);
00386
00387     // search until index - 1
00388     for (int i = 0; i < index - 1; ++i) {
00389         m_list.at(i).set_color_index(2);
00390         m_sequence.insert(m_sequence.size(), m_list);
00391         m_code_highlighter.push_into_sequence(1);
00392
00393         m_list.at(i).set_color_index(0);
00394         m_list.at(i).set_label(i == 0 ? "head" : "");
00395         m_list.at(i + 1).set_color_index(2);
00396         m_list.at(i + 1).set_label("pre");
00397         m_sequence.insert(m_sequence.size(), m_list);
00398         m_code_highlighter.push_into_sequence(2);
00399     }
00400
00401     m_sequence.insert(m_sequence.size(), m_list);
00402     m_code_highlighter.push_into_sequence(1);
00403
00404     // reaching index - 1
00405     // cur
00406     m_list.at(index - 1).set_color_index(2);
00407     m_sequence.insert(m_sequence.size(), m_list);
00408     m_code_highlighter.push_into_sequence(3);
00409
00410     // cur->next
00411     m_list.at(index).set_color_index(7);
00412     m_list.at(index).set_label(index + 1 == m_list.size() ? "tail/nxt" : "nxt");
00413     m_sequence.insert(m_sequence.size(), m_list);
00414     m_code_highlighter.push_into_sequence(4);
00415
00416     // insert between cur and cur->next
00417     m_list.insert(index, value);
00418     m_list.at(index).set_color_index(6);

```

```

00419     m_list.at(index).set_label("node");
00420     m_sequence.insert(m_sequence.size(), m_list);
00421     m_code_highlighter.push_into_sequence(5);
00422
00423     m_list.at(index - 1).set_color_index(2);
00424     m_list.at(index + 1).set_color_index(0);
00425     m_sequence.insert(m_sequence.size(), m_list);
00426     m_code_highlighter.push_into_sequence(6);
00427
00428     m_list.at(index - 1).set_color_index(0);
00429     m_list.at(index + 1).set_color_index(7);
00430     m_list.init_label();
00431     m_sequence.insert(m_sequence.size(), m_list);
00432     m_code_highlighter.push_into_sequence(7);
00433
00434     // done
00435     m_list.at(index - 1).set_color_index(0);
00436     m_list.at(index - 1).set_label("");
00437     m_list.at(index).set_color_index(0);
00438     m_list.at(index).set_label("");
00439     m_list.at(index + 1).set_color_index(0);
00440     m_list.at(index + 1).set_label("");
00441     m_list.init_label();
00442 }
00443
00444 template<typename Con>
00445 void BaseLinkedListScene<Con>::interact_delete() {
00446     if (m_list.empty()) {
00447         return;
00448     }
00449
00450     auto index_container = m_index_input.extract_values();
00451     if (index_container.empty()) {
00452         return;
00453     }
00454
00455     int index = index_container.front();
00456
00457     if (!(0 <= index && index < m_list.size())) {
00458         return;
00459     }
00460
00461     m_sequence.clear();
00462     m_sequence.insert(m_sequence.size(), m_list);
00463
00464     if (index == 0) {
00465         interact_delete_head();
00466     } else if (index + 1 == m_list.size()) {
00467         interact_delete_tail();
00468     } else {
00469         interact_delete_middle(index);
00470     }
00471
00472     m_sequence_controller.set_max_value((int)m_sequence.size());
00473     m_sequence_controller.set_rerun();
00474 }
00475
00476 template<typename Con>
00477 void BaseLinkedListScene<Con>::interact_delete_head() {
00478     m_code_highlighter.set_code({
00479         "Node* temp = head;",
00480         "head = head->next;",
00481         "delete temp;",
00482     });
00483     m_code_highlighter.push_into_sequence(-1);
00484
00485     m_list.at(0).set_color_index(4);
00486     m_sequence.insert(m_sequence.size(), m_list);
00487     m_code_highlighter.push_into_sequence(0);
00488
00489     m_list.at(0).set_color_index(5);
00490     m_list.at(0).set_label("");
00491     if (m_list.size() > 1) {
00492         m_list.at(1).set_color_index(4);
00493         m_list.at(1).set_label("head");
00494     }
00495     m_sequence.insert(m_sequence.size(), m_list);
00496     m_code_highlighter.push_into_sequence(1);
00497
00498     m_list.remove(0);
00499     m_sequence.insert(m_sequence.size(), m_list);
00500     m_code_highlighter.push_into_sequence(2);
00501
00502     if (m_list.size() > 0) {
00503         m_list.at(0).set_color_index(0);
00504     }
00505 }

```

```

00506
00507 template<typename Con>
00508 void BaseLinkedListScene<Con>::interact_delete_tail() {
00509     m_code_highlighter.set_code({
00510         "Node* pre = head;",
00511         "Node* nxt = pre->next;",
00512         "while (nxt->next != nullptr)",
00513         "    pre = pre->next, nxt = nxt->next;",
00514         "",
00515         "delete nxt;",
00516         "tail = pre;",
00517     });
00518     m_code_highlighter.push_into_sequence(-1);
00519
00520     m_list.at(0).set_color_index(2);
00521     m_list.at(0).set_label("head/pre");
00522     m_sequence.insert(m_sequence.size(), m_list);
00523     m_code_highlighter.push_into_sequence(0);
00524
00525     m_list.at(1).set_color_index(3);
00526     if (m_list.size() == 2) {
00527         m_list.at(1).set_label("tail/nxt");
00528     } else {
00529         m_list.at(1).set_label("nxt");
00530     }
00531     m_sequence.insert(m_sequence.size(), m_list);
00532     m_code_highlighter.push_into_sequence(1);
00533
00534     int idx = 0;
00535     for (; idx + 2 < m_list.size(); ++idx) {
00536         m_sequence.insert(m_sequence.size(), m_list);
00537         m_code_highlighter.push_into_sequence(2);
00538
00539         m_list.at(idx).set_color_index(0);
00540         if (idx == 0) {
00541             m_list.at(idx).set_label("head");
00542         } else {
00543             m_list.at(idx).set_label("");
00544         }
00545
00546         m_list.at(idx + 1).set_color_index(2);
00547         m_list.at(idx + 1).set_label("pre");
00548         m_list.at(idx + 2).set_color_index(3);
00549         if (idx + 3 == m_list.size()) {
00550             m_list.at(idx + 2).set_label("tail/nxt");
00551         } else {
00552             m_list.at(idx + 2).set_label("nxt");
00553         }
00554
00555         m_sequence.insert(m_sequence.size(), m_list);
00556         m_code_highlighter.push_into_sequence(3);
00557     }
00558
00559     m_sequence.insert(m_sequence.size(), m_list);
00560     m_code_highlighter.push_into_sequence(2);
00561
00562     m_list.at(idx).set_color_index(2);
00563     m_list.at(idx).set_label("pre");
00564     m_list.at(idx + 1).set_color_index(3);
00565     m_list.at(idx + 1).set_label("tail/nxt");
00566     m_sequence.insert(m_sequence.size(), m_list);
00567     m_code_highlighter.push_into_sequence(4);
00568
00569     m_list.remove(idx + 1);
00570     m_list.at(idx).set_label("tail/pre");
00571     m_sequence.insert(m_sequence.size(), m_list);
00572     m_code_highlighter.push_into_sequence(5);
00573
00574     m_list.at(idx).set_color_index(4);
00575     m_list.init_label();
00576     m_sequence.insert(m_sequence.size(), m_list);
00577     m_code_highlighter.push_into_sequence(6);
00578
00579     m_list.at(idx).set_color_index(0);
00580 }
00581
00582 template<typename Con>
00583 void BaseLinkedListScene<Con>::interact_delete_middle(int index) {
00584     m_code_highlighter.set_code({
00585         "Node* pre = head;",
00586         "for (i = 0; i < index - 1; i++)",
00587         "    pre = pre->next;",
00588         "",
00589         "Node* node = pre->next;",
00590         "Node* nxt = node->next;",
00591         "delete node;",
00592         "pre->next = nxt;",

```



```

00593     });
00594     m_code_highlighter.push_into_sequence(-1);
00595
00596     m_list.at(0).set_color_index(4);
00597     m_list.at(0).set_label("head/pre");
00598     m_sequence.insert(m_sequence.size(), m_list);
00599     m_code_highlighter.push_into_sequence(0);
00600
00601     int idx = 0;
00602     for (; idx + 1 < index; ++idx) {
00603         m_list.at(idx).set_color_index(2);
00604         m_sequence.insert(m_sequence.size(), m_list);
00605         m_code_highlighter.push_into_sequence(1);
00606
00607         m_list.at(idx).set_color_index(0);
00608         m_list.at(idx).set_label("");
00609         m_list.at(idx + 1).set_color_index(2);
00610         m_list.init_label();
00611         m_list.at(idx + 1).set_label("pre");
00612         m_sequence.insert(m_sequence.size(), m_list);
00613         m_code_highlighter.push_into_sequence(2);
00614     }
00615
00616     m_list.at(idx).set_color_index(2);
00617     m_list.at(idx).set_label("pre");
00618     m_sequence.insert(m_sequence.size(), m_list);
00619     m_code_highlighter.push_into_sequence(3);
00620
00621     m_list.at(idx + 1).set_color_index(5);
00622     m_list.at(idx + 1).set_label("node");
00623     m_sequence.insert(m_sequence.size(), m_list);
00624     m_code_highlighter.push_into_sequence(4);
00625
00626     m_list.at(idx + 2).set_color_index(3);
00627     if (idx + 3 == m_list.size()) {
00628         m_list.at(idx + 2).set_label("tail/nxt");
00629     } else {
00630         m_list.at(idx + 2).set_label("nxt");
00631     }
00632     m_sequence.insert(m_sequence.size(), m_list);
00633     m_code_highlighter.push_into_sequence(5);
00634
00635     m_list.remove(idx + 1);
00636     m_sequence.insert(m_sequence.size(), m_list);
00637     m_code_highlighter.push_into_sequence(6);
00638
00639     m_list.at(idx + 1).set_color_index(7);
00640     m_sequence.insert(m_sequence.size(), m_list);
00641     m_code_highlighter.push_into_sequence(7);
00642
00643     m_list.at(idx).set_color_index(0);
00644     m_list.at(idx).set_label("");
00645     m_list.at(idx + 1).set_color_index(0);
00646     m_list.at(idx + 1).set_label("");
00647 }
00648
00649 template<typename Con>
00650 void BaseLinkedListScene<Con>::interact_update() {
00651     auto index_container = m_index_input.extract_values();
00652     if (index_container.empty()) {
00653         return;
00654     }
00655
00656     auto value_container = m_text_input.extract_values();
00657     if (value_container.empty()) {
00658         return;
00659     }
00660
00661     int index = index_container.front();
00662     int value = value_container.front();
00663
00664     if (!(0 <= index && index < m_list.size())) {
00665         return;
00666     }
00667
00668     m_code_highlighter.set_code({
00669         "Node* node = head;",
00670         "for (i = 0; i < index; i++)",
00671         "    node = node->next;",
00672         "",
00673         "node->value = value;",
00674     });
00675
00676     m_sequence.clear();
00677     m_sequence.insert(m_sequence.size(), m_list);
00678     m_code_highlighter.push_into_sequence(-1);
00679

```

```

00680     m_list.at(0).set_color_index(4);
00681     m_list.at(0).set_label("head/node");
00682     m_sequence.insert(m_sequence.size(), m_list);
00683     m_code_highlighter.push_into_sequence(0);
00684
00685     for (int i = 0; i < index; ++i) {
00686         m_list.at(i).set_color_index(2);
00687         m_sequence.insert(m_sequence.size(), m_list);
00688         m_code_highlighter.push_into_sequence(1);
00689
00690         m_list.at(i).set_color_index(0);
00691         m_list.at(i).set_label(i == 0 ? "head" : "");
00692         m_list.at(i + 1).set_color_index(2);
00693         m_list.at(i + 1).set_label(i + 2 == m_list.size() ? "tail/node"
00694                                     : "node");
00695         m_sequence.insert(m_sequence.size(), m_list);
00696         m_code_highlighter.push_into_sequence(2);
00697     }
00698
00699     m_sequence.insert(m_sequence.size(), m_list);
00700     m_code_highlighter.push_into_sequence(1);
00701     m_sequence.insert(m_sequence.size(), m_list);
00702     m_code_highlighter.push_into_sequence(3);
00703
00704     m_list.at(index).set_color_index(3);
00705     m_list.at(index).set_value(value);
00706     m_sequence.insert(m_sequence.size(), m_list);
00707     m_code_highlighter.push_into_sequence(4);
00708
00709     m_list.at(index).set_color_index(0);
00710     m_list.at(index).set_label("");
00711     m_list.init_label();
00712
00713     m_sequence_controller.set_max_value((int)m_sequence.size());
00714     m_sequence_controller.set_rerun();
00715 }
00716
00717 template<typename Con>
00718 void BaseLinkedListScene<Con>::interact_search() {
00719     auto value_container = m_text_input.extract_values();
00720     if (value_container.empty()) {
00721         return;
00722     }
00723
00724     int value = value_container.front();
00725     if (!utils::val_in_range(value)) {
00726         return;
00727     }
00728
00729     m_code_highlighter.set_code({
00730         "Node* node = head;",
00731         "while (node != nullptr) {",
00732             "    if (node->value == value)",
00733             "        return node;",
00734             "    node = node->next;",
00735         "}",
00736         "return not_found",
00737     });
00738
00739     m_sequence.clear();
00740     m_sequence.insert(m_sequence.size(), m_list);
00741     m_code_highlighter.push_into_sequence(-1);
00742
00743     m_list.at(0).set_color_index(4);
00744     m_list.at(0).set_label("head/node");
00745     m_sequence.insert(m_sequence.size(), m_list);
00746     m_code_highlighter.push_into_sequence(0);
00747
00748     std::size_t idx = 0;
00749
00750     while (idx < m_list.size()) {
00751         m_list.at(idx).set_color_index(2);
00752         m_sequence.insert(m_sequence.size(), m_list);
00753         m_code_highlighter.push_into_sequence(1);
00754
00755         m_sequence.insert(m_sequence.size(), m_list);
00756         m_code_highlighter.push_into_sequence(2);
00757         if (m_list.at(idx).get_value() == value) {
00758             m_list.at(idx).set_color_index(3);
00759             m_sequence.insert(m_sequence.size(), m_list);
00760             m_code_highlighter.push_into_sequence(3);
00761             m_list.at(idx).set_color_index(0);
00762             m_list.at(idx).set_label(idx + 1 == m_list.size() ? "tail" : "");
00763             break;
00764         }
00765
00766         m_list.at(idx).set_color_index(0);

```

```

00767         m_list.at(idx).set_label("");
00768         m_list.init_label();
00769         ++idx;
00770         if (idx < m_list.size()) {
00771             m_list.at(idx).set_color_index(2);
00772             m_list.at(idx).set_label(idx + 1 == m_list.size() ? "tail/node"
00773                                     : "node");
00774         }
00775         m_sequence.insert(m_sequence.size(), m_list);
00776         m_code_highlighter.push_into_sequence(4);
00777     }
00778
00779     if (idx >= m_list.size()) {
00780         m_sequence.insert(m_sequence.size(), m_list);
00781         m_code_highlighter.push_into_sequence(1);
00782
00783         m_sequence.insert(m_sequence.size(), m_list);
00784         m_code_highlighter.push_into_sequence(5);
00785
00786         m_sequence.insert(m_sequence.size(), m_list);
00787         m_code_highlighter.push_into_sequence(6);
00788     }
00789
00790     m_sequence_controller.set_max_value((int)m_sequence.size());
00791     m_sequence_controller.set_rerun();
00792 }
00793
00794 } // namespace scene
00795
00796 #endif // SCENE_BASE_LINKED_LIST_SCENE_HPP_

```

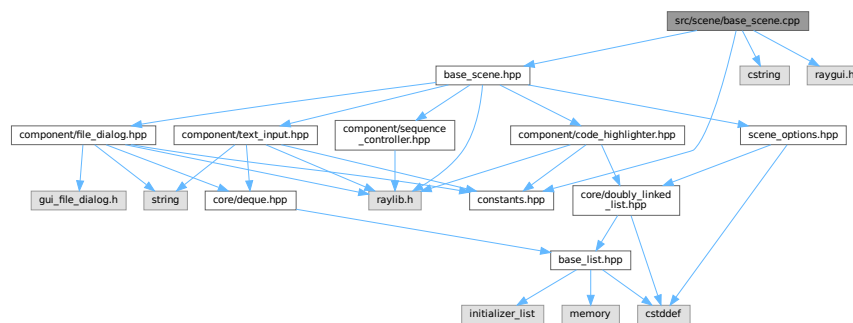
7.73 src/scene/base_scene.cpp File Reference

```

#include "base_scene.hpp"
#include <cstring>
#include "constants.hpp"
#include "raygui.h"

```

Include dependency graph for base_scene.cpp:



Namespaces

- namespace `scene`
- namespace `scene::internal`

7.74 base_scene.cpp

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

```

00001 #include "base_scene.hpp"
00002
00003 #include <cstring>
00004
00005 #include "constants.hpp"
00006 #include "raygui.h"
00007
00008 namespace scene::internal {
00009
00010 bool BaseScene::render_go_button() const {
00011     Rectangle shape{options_head, constants::scene_height - button_size.y,
00012                    button_size.y, button_size.y};
00013     return GuiButton(shape, "Go");
00014 }
00015
00016 void BaseScene::render_options(SceneOptions& scene_config) {
00017     (m_edit_mode || m_edit_action) ? GuiLock() : GuiUnlock();
00018
00019     options_head = 2 * constants::sidebar_width;
00020
00021     Rectangle mode_button_shape{options_head,
00022                                constants::scene_height - button_size.y,
00023                                button_size.x, button_size.y};
00024
00025     options_head += (button_size.x + head_offset);
00026
00027     int& mode = scene_config.mode_selection;
00028
00029     if (GuiDropupBox(mode_button_shape, scene_config.mode_labels, &mode,
00030                     m_edit_mode)) {
00031         m_edit_mode ^= 1;
00032     }
00033
00034     if (std::strlen(scene_config.action_labels.at(mode)) != 0) {
00035         Rectangle action_button_shape{options_head,
00036                                      constants::scene_height - button_size.y,
00037                                      button_size.x, button_size.y};
00038
00039         options_head += (button_size.x + head_offset);
00040
00041         int& action_selection = scene_config.action_selection.at(mode);
00042
00043         if (GuiDropupBox(action_button_shape,
00044                         scene_config.action_labels.at(mode), &action_selection,
00045                         m_edit_action)) {
00046             m_edit_action ^= 1;
00047         }
00048
00049         // scene_config.action_selection.at(mode) = GuiComboBox(
00050         //     action_button_shape, scene_config.action_labels.at(mode),
00051         //     scene_config.action_selection.at(mode));
00052     }
00053
00054     render_inputs();
00055 }
00056
00057 } // namespace scene::internal

```

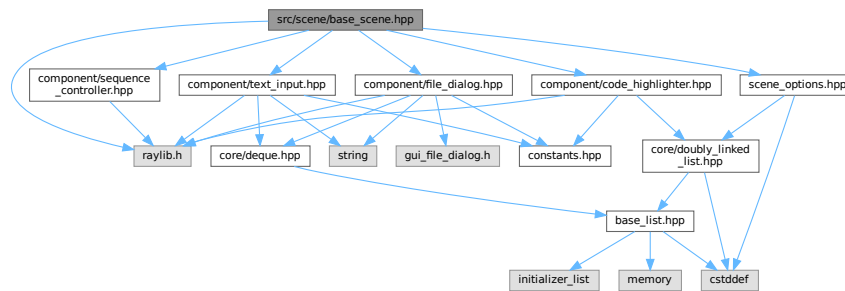
7.75 src/scene/base_scene.hpp File Reference

```

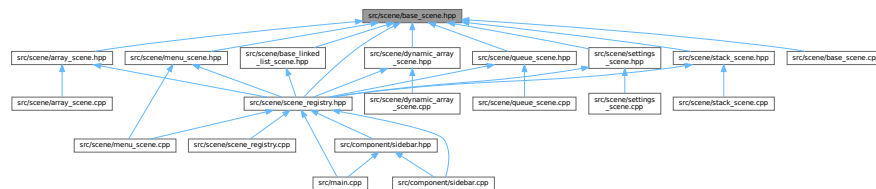
#include "component/code_highlighter.hpp"
#include "component/file_dialog.hpp"
#include "component/sequence_controller.hpp"
#include "component/text_input.hpp"
#include "raylib.h"
#include "scene_options.hpp"

```

Include dependency graph for base_scene.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [scene::internal::BaseScene](#)

Namespaces

- namespace [scene](#)
- namespace [scene::internal](#)

7.76 base_scene.hpp

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

```

00001 #ifndef SCENE_BASE_SCENE_HPP_
00002 #define SCENE_BASE_SCENE_HPP_
00003
00004 #include "component/code_highlighter.hpp"
00005 #include "component/file_dialog.hpp"
00006 #include "component/sequence_controller.hpp"
00007 #include "component/text_input.hpp"
00008 #include "raylib.h"
00009 #include "scene_options.hpp"
00010
00011 namespace scene::internal {
00012
00013 class BaseScene {
00014 protected:
00015     static constexpr Vector2 button_size{200, 50};
00016     static constexpr int head_offset = 20;
00017     float options_head{};
00018
00019     virtual bool render_go_button() const;
00020     virtual void render_options(SceneOptions& scene_config);
  
```



```

00009 // #include <string>
00010
00011 #include "constants.hpp"
00012 #include "raygui.h"
00013 #include "utils.hpp"
00014
00015 namespace scene {
00016
00017 void DynamicArrayScene::render_inputs() {
00018     int& mode = scene_options.mode_selection;
00019
00020     switch (mode) {
00021         case 0: {
00022             switch (scene_options.action_selection.at(mode)) {
00023                 case 0:
00024                     break;
00025                 case 1: {
00026                     m_text_input.render(options_head, head_offset);
00027                 } break;
00028                 case 2: {
00029                     m_go = (m_file_dialog.render_head(options_head,
00030                                                         head_offset) > 0);
00031                     return;
00032                 } break;
00033                 default:
00034                     utils::unreachable();
00035             }
00036         } break;
00037
00038         case 1: {
00039             m_index_input.render(options_head, head_offset);
00040             m_text_input.render(options_head, head_offset);
00041         } break;
00042
00043         case 2:
00044         case 3: {
00045             m_text_input.render(options_head, head_offset);
00046         } break;
00047
00048         case 4:
00049             break;
00050
00051         default:
00052             utils::unreachable();
00053     }
00054
00055     m_go |= render_go_button();
00056 }
00057
00058 void DynamicArrayScene::render() {
00059     m_sequence_controller.inc_anim_counter();
00060
00061     int frame_idx = m_sequence_controller.get_anim_frame();
00062     auto* const frame_ptr = m_sequence.find(frame_idx);
00063     m_sequence_controller.set_progress_value(frame_idx);
00064
00065     if (frame_ptr != nullptr) {
00066         frame_ptr->data.render();
00067         m_code_highlighter.highlight(frame_idx);
00068     } else { // end of sequence
00069         m_array.render();
00070         m_sequence_controller.set_run_all(false);
00071     }
00072
00073     m_code_highlighter.render();
00074     m_sequence_controller.render();
00075     render_options(scene_options);
00076 }
00077
00078 void DynamicArrayScene::interact() {
00079     if (m_sequence_controller.interact()) {
00080         m_sequence_controller.reset_anim_counter();
00081         return;
00082     }
00083
00084     m_index_input.set_random_max((int)m_array.size() - 1);
00085
00086     if (m_text_input.interact() || m_index_input.interact()) {
00087         return;
00088     }
00089
00090     if (!m_go) {
00091         return;
00092     }
00093
00094     int& mode = scene_options.mode_selection;
00095

```

```

00096     switch (mode) {
00097     case 0: {
00098         switch (scene_options.action_selection.at(mode)) {
00099             case 0: {
00100                 interact_random();
00101             } break;
00102
00103             case 1: {
00104                 interact_import(m_text_input.extract_values());
00105             } break;
00106
00107             case 2: {
00108                 interact_file_import();
00109             } break;
00110
00111             default:
00112                 utils::unreachable();
00113         }
00114     } break;
00115
00116     case 1: {
00117         interact_update();
00118     } break;
00119
00120     case 2: {
00121         interact_search();
00122     } break;
00123
00124     case 3: {
00125         interact_push();
00126     } break;
00127
00128     case 4: {
00129         interact_pop();
00130     } break;
00131
00132     default:
00133         utils::unreachable();
00134 }
00135
00136 m_go = false;
00137 }
00138
00139 void DynamicArrayScene::interact_random() {
00140     std::size_t size =
00141         utils::get_random(std::size_t{1}, scene_options.max_size);
00142     m_array = {};
00143
00144     for (std::size_t i = 0; i < size; ++i) {
00145         m_array.push(utils::get_random(constants::min_val, constants::max_val));
00146     }
00147 }
00148
00149 void DynamicArrayScene::interact_import(core::Deque<int> nums) {
00150     m_array = {};
00151     std::size_t i; // NOLINT
00152
00153     for (i = 0; i < max_size && !nums.empty(); ++i) {
00154         m_array.push(nums.front());
00155         nums.pop_front();
00156     }
00157 }
00158
00159 void DynamicArrayScene::interact_update() {
00160     auto index_container = m_index_input.extract_values();
00161     if (index_container.empty()) {
00162         return;
00163     }
00164
00165     auto value_container = m_text_input.extract_values();
00166     if (value_container.empty()) {
00167         return;
00168     }
00169
00170     int index = index_container.front();
00171     int value = value_container.front();
00172
00173     if (!(0 <= index && index < m_array.size()) ||
00174         !utils::val_in_range(value)) {
00175         return;
00176     }
00177
00178     m_code_highlighter.set_code({
00179         "array[index] = value;",
00180     });
00181
00182     m_sequence.clear();

```



```

00183
00184 // initial state (before update)
00185 m_sequence.insert(m_sequence.size(), m_array);
00186 m_code_highlighter.push_into_sequence(-1);
00187
00188 // highlight
00189 m_array.set_color_index(index, 2);
00190 m_sequence.insert(m_sequence.size(), m_array);
00191 m_code_highlighter.push_into_sequence(0);
00192
00193 // update
00194 m_array[index] = value;
00195 m_array.set_color_index(index, 3);
00196 m_sequence.insert(m_sequence.size(), m_array);
00197 m_code_highlighter.push_into_sequence(0);
00198
00199 // undo highlight
00200 m_array.set_color_index(index, 0);
00201
00202 m_sequence_controller.set_max_value((int)m_sequence.size());
00203 m_sequence_controller.set_rerun();
00204 }
00205
00206 void DynamicArrayScene::interact_file_import() {
00207     interact_import(m_file_dialog.extract_values());
00208 }
00209
00210 void DynamicArrayScene::interact_search() {
00211     auto value_container = m_text_input.extract_values();
00212     if (value_container.empty()) {
00213         return;
00214     }
00215
00216     int value = value_container.front();
00217     if (!utils::val_in_range(value)) {
00218         return;
00219     }
00220
00221     m_code_highlighter.set_code({
00222         "for (i = 0; i < size; i++)",
00223         "    if (array[i] == value)",
00224         "        return i;",
00225         "return not_found",
00226     });
00227
00228     m_sequence.clear();
00229     m_sequence.insert(m_sequence.size(), m_array);
00230     m_code_highlighter.push_into_sequence(0);
00231
00232     bool found = false;
00233
00234     for (std::size_t i = 0; i < m_array.size(); ++i) {
00235         m_array.set_color_index(i, 3);
00236         m_sequence.insert(m_sequence.size(), m_array);
00237         m_code_highlighter.push_into_sequence(1);
00238
00239         if (m_array[i] == value) {
00240             found = true;
00241             m_array.set_color_index(i, 4);
00242             m_sequence.insert(m_sequence.size(), m_array);
00243             m_code_highlighter.push_into_sequence(2);
00244             m_array.set_color_index(i, 0);
00245             break;
00246         }
00247
00248         m_array.set_color_index(i, 0);
00249         m_sequence.insert(m_sequence.size(), m_array);
00250         m_code_highlighter.push_into_sequence(0);
00251     }
00252
00253     if (!found) {
00254         m_sequence.insert(m_sequence.size(), m_array);
00255         m_code_highlighter.push_into_sequence(3);
00256     }
00257
00258     m_sequence_controller.set_max_value((int)m_sequence.size());
00259     m_sequence_controller.set_rerun();
00260 }
00261
00262 void DynamicArrayScene::interact_push() {
00263     int value = m_text_input.extract_values().front();
00264
00265     if (m_array.size() >= max_size) {
00266         return;
00267     }
00268
00269     m_code_highlighter.set_code({

```

```

00270         "if (size == capacity)",
00271         "     capacity *= 2;",
00272         "array[size] = value;",
00273         "size++;",
00274     });
00275
00276     m_sequence.clear();
00277     m_sequence.insert(m_sequence.size(), m_array);
00278     m_code_highlighter.push_into_sequence(-1);
00279
00280     m_sequence.insert(m_sequence.size(), m_array);
00281     m_code_highlighter.push_into_sequence(0);
00282
00283     if (m_array.size() == m_array.capacity()) {
00284         m_array.realloc(m_array.size() + 1);
00285         m_sequence.insert(m_sequence.size(), m_array);
00286         m_code_highlighter.push_into_sequence(1);
00287     }
00288
00289     m_array.push(value);
00290     m_array.set_color_index(m_array.size() - 1, 4);
00291     m_sequence.insert(m_sequence.size(), m_array);
00292     m_code_highlighter.push_into_sequence(2);
00293
00294     m_array.set_color_index(m_array.size() - 1, 0);
00295     m_sequence.insert(m_sequence.size(), m_array);
00296     m_code_highlighter.push_into_sequence(3);
00297
00298     m_sequence_controller.set_max_value((int)m_sequence.size());
00299     m_sequence_controller.set_rerun();
00300 }
00301
00302 void DynamicArrayScene::interact_pop() {
00303     if (m_array.size() == 0) {
00304         return;
00305     }
00306
00307     m_code_highlighter.set_code({
00308         "array[size - 1] = 0;",
00309         "size--;",
00310     });
00311
00312     m_sequence.clear();
00313     m_sequence.insert(m_sequence.size(), m_array);
00314     m_code_highlighter.push_into_sequence(-1);
00315
00316     m_array.set_color_index(m_array.size() - 1, 3);
00317     m_sequence.insert(m_sequence.size(), m_array);
00318     m_code_highlighter.push_into_sequence(0);
00319
00320     m_array.pop();
00321     m_sequence.insert(m_sequence.size(), m_array);
00322     m_code_highlighter.push_into_sequence(1);
00323
00324     m_sequence_controller.set_max_value((int)m_sequence.size());
00325     m_sequence_controller.set_rerun();
00326 }
00327
00328 } // namespace scene

```

7.79 src/scene/dynamic_array_scene.hpp File Reference

```

#include <array>
#include <cstddef>
#include "base_scene.hpp"
#include "component/file_dialog.hpp"
#include "component/text_input.hpp"
#include "constants.hpp"
#include "core/doubly_linked_list.hpp"
#include "gui/dynamic_array_gui.hpp"
#include "raygui.h"
#include "raylib.h"

```



```

00021
00022     internal::SceneOptions scene_options{
00023         // max_size
00024         max_size,
00025
00026         // mode_labels
00027         "Mode: Create;"
00028         "Mode: Update;"
00029         "Mode: Search;"
00030         "Mode: Push;"
00031         "Mode: Pop",
00032
00033         // mode_selection
00034         0,
00035
00036         // action_labels
00037         {
00038             // Mode: Create
00039             "Action: Random;Action: Input;Action: File",
00040
00041             // Mode: Update
00042             "",
00043
00044             // Mode: Search
00045             "",
00046
00047             // Mode: Push
00048             "",
00049
00050             // Mode: Pop
00051             "",
00052         },
00053
00054         // action_selection
00055         core::DoublyLinkedList<int>{0, 0, 0, 0, 0},
00056     };
00057
00058     using internal::BaseScene::button_size;
00059     using internal::BaseScene::head_offset;
00060     using internal::BaseScene::options_head;
00061
00062     gui::GuiDynamicArray<int> m_array{};
00063     core::DoublyLinkedList<gui::GuiDynamicArray<int>> m_sequence;
00064
00065     bool m_go{};
00066     using internal::BaseScene::m_file_dialog;
00067     using internal::BaseScene::m_index_input;
00068     using internal::BaseScene::m_sequence_controller;
00069     using internal::BaseScene::m_text_input;
00070
00071     using internal::BaseScene::render_go_button;
00072     using internal::BaseScene::render_options;
00073     void render_inputs() override;
00074
00075     void interact_random();
00076     void interact_import(core::Deque<int> nums);
00077     void interact_file_import();
00078     void interact_update();
00079     void interact_search();
00080     void interact_push();
00081     void interact_pop();
00082
00083 public:
00084     void render() override;
00085     void interact() override;
00086 };
00087
00088 } // namespace scene
00089
00090 #endif // SCENE_DYNAMIC_ARRAY_SCENE_HPP_

```

7.81 src/scene/menu_scene.cpp File Reference

```

#include "menu_scene.hpp"
#include <iostream>
#include "constants.hpp"
#include "raygui.h"
#include "raylib.h"
#include "scene_registry.hpp"

```

```
#include "settings.hpp"
#include "utils.hpp"
Include dependency graph for menu_scene.cpp:
```



Namespaces

- namespace `scene`

7.82 menu_scene.cpp

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

```
00001 #include "menu_scene.hpp"
00002
00003 #include <iostream>
00004
00005 #include "constants.hpp"
00006 #include "raygui.h"
00007 #include "raylib.h"
00008 #include "scene_registry.hpp"
00009 #include "settings.hpp"
00010 #include "utils.hpp"
00011
00012 namespace scene {
00013
00014 MenuScene::MenuScene() {
00015     constexpr int block_width = component::MenuItem::block_width;
00016     constexpr int block_height = component::MenuItem::block_height;
00017     constexpr int button_width = component::MenuItem::button_width;
00018     constexpr int button_height = component::MenuItem::button_height;
00019     constexpr int gap = 20;
00020
00021     constexpr int first_row_y =
00022         constants::scene_height / 16.0F * 5 - block_height / 2.0F;
00023
00024     // first row
00025     {
00026         constexpr int row_width =
00027             3 * component::MenuItem::block_width + 2 * gap;
00028         constexpr int row_x = constants::scene_width / 2.0F - row_width / 2.0F;
00029         constexpr int row_y = first_row_y;
00030
00031         for (auto i = 0; i < 3; ++i) {
00032             m_menu_items[i] = component::MenuItem(
00033                 i, labels[i], row_x + i * (block_width + gap), row_y,
00034                 img_paths[i]);
00035         }
00036     }
00037
00038     // second row
00039     {
00040         constexpr int row_width = 4 * block_width + 3 * gap;
00041         constexpr int row_x = constants::scene_width / 2.0F - row_width / 2.0F;
00042         constexpr int row_y = first_row_y + block_height + gap;
00043
00044         for (auto i = 3; i < 7; ++i) {
00045             m_menu_items[i] = component::MenuItem(
00046                 i, labels[i], row_x + (i - 3) * (block_width + gap), row_y,
00047                 img_paths[i]);
00048         }
00049     }
00050 }
00051
00052 void MenuScene::render() {
00053     const Color text_color = utils::adaptive_text_color(
00054         Settings::get_instance().get_color(Settings::num_color - 1));
00055
00056     // Menu text
```

```

00057     constexpr int menu_font_size = 60;
00058     constexpr int menu_font_spacing = 5;
00059
00060     constexpr const char* menu_text = "CS162 - VisuAlgo.net clone in C++";
00061
00062     const Vector2 menu_text_size =
00063         utils::MeasureText(menu_text, menu_font_size, menu_font_spacing);
00064
00065     const Vector2 menu_text_pos{
00066         constants::scene_width / 2.0F - menu_text_size.x / 2,
00067         constants::scene_height / 16.0F - menu_text_size.y / 2};
00068
00069     utils::DrawText(menu_text, menu_text_pos, text_color, menu_font_size,
00070         menu_font_spacing);
00071
00072     // Sub text
00073     constexpr int sub_font_size = 30;
00074     constexpr int sub_font_spacing = 2;
00075
00076     constexpr const char* sub_text = "By Quang-Truong Nguyen (@jalsol)";
00077
00078     const Vector2 sub_text_size =
00079         utils::MeasureText(sub_text, sub_font_size, sub_font_spacing);
00080
00081     const Vector2 sub_text_pos{
00082         constants::scene_width / 2.0F - sub_text_size.x / 2,
00083         menu_text_pos.y + menu_text_size.y / 2 + sub_text_size.y};
00084
00085     utils::DrawText(sub_text, sub_text_pos, text_color, sub_font_size,
00086         sub_font_spacing);
00087
00088     // Button
00089     constexpr int block_width = 300;
00090     constexpr int block_height = 200;
00091     constexpr int button_width = block_width;
00092     constexpr int button_height = 50;
00093     constexpr int gap = 20;
00094     constexpr int first_row_y =
00095         constants::scene_height / 16.0F * 5 - block_height / 2.0F;
00096
00097     for (auto i = 0; i < 7; ++i) {
00098         m_menu_items[i].render();
00099     }
00100
00101     const Rectangle quit_button_shape{
00102         constants::scene_width / 2.0F - 128,
00103         constants::scene_height / 16.0F * 15 - block_height / 2.0F, 256, 64};
00104
00105     m_quit = GuiButton(quit_button_shape, "Quit");
00106
00107     // Bottom text
00108     constexpr int bot_font_size = 20;
00109     constexpr int bot_font_spacing = 2;
00110
00111     constexpr const char* bot_text =
00112         "(pls read the src code, i tried so hard for this)";
00113
00114     const Vector2 bot_text_size =
00115         utils::MeasureText(bot_text, bot_font_size, bot_font_spacing);
00116
00117     const Vector2 bot_text_pos{
00118         constants::scene_width / 2.0F - bot_text_size.x / 2,
00119         constants::scene_height - 1.5F * bot_text_size.y};
00120
00121     utils::DrawText(bot_text, bot_text_pos, text_color, bot_font_size,
00122         bot_font_spacing);
00123 }
00124
00125 void MenuScene::interact() {
00126     scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();
00127
00128     if (m_quit) {
00129         registry.close_window();
00130         return;
00131     }
00132
00133     for (auto i = 0; i < 7; ++i) {
00134         if (m_menu_items[i].clicked()) {
00135             m_next_scene = i;
00136             m_start = true;
00137         }
00138     }
00139
00140     for (auto i = 0; i < 7; ++i) {
00141         m_menu_items[i].reset();
00142     }
00143 }

```

```

00144     if (m_start) {
00145         registry.set_scene(m_next_scene);
00146         m_start = false;
00147     }
00148 }
00149
00150 } // namespace scene

```

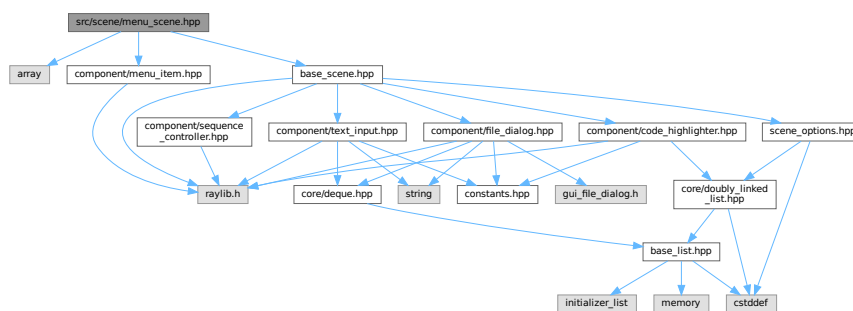
7.83 src/scene/menu_scene.hpp File Reference

```

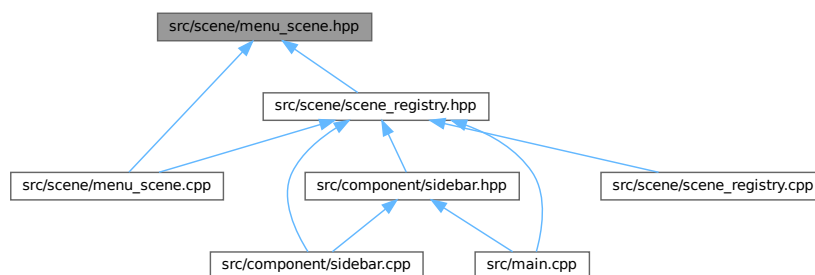
#include <array>
#include "base_scene.hpp"
#include "component/menu_item.hpp"

```

Include dependency graph for menu_scene.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [scene::MenuScene](#)

Namespaces

- namespace [scene](#)

7.84 menu_scene.hpp

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

```

00001 #ifndef SCENE_MENU_SCENE_HPP_
00002 #define SCENE_MENU_SCENE_HPP_
00003
00004 #include <array>
00005
00006 #include "base_scene.hpp"
00007 #include "component/menu_item.hpp"
00008
00009 namespace scene {
00010
00011 class MenuScene : public internal::BaseScene {
00012 private:
00013     bool m_start{};
00014     bool m_quit{};
00015     int m_next_scene{};
00016
00017     static constexpr std::array<const char*, 7> labels = {{
00018         "Array",
00019         "Dynamic Array",
00020         "Linked List",
00021         "Doubly Linked List",
00022         "Circular Linked List",
00023         "Stack",
00024         "Queue",
00025     }};
00026
00027     static constexpr std::array<const char*, 7> img_paths = {{
00028         "data/preview/array.png",
00029         "data/preview/dynamic_array.png",
00030         "data/preview/linked_list.png",
00031         "data/preview/doubly_linked_list.png",
00032         "data/preview/circular_linked_list.png",
00033         "data/preview/stack.png",
00034         "data/preview/queue.png",
00035     }};
00036
00037     std::array<component::MenuItem, 7> m_menu_items{};
00038
00039 public:
00040     MenuScene();
00041     void render() override;
00042     void interact() override;
00043 };
00044
00045 } // namespace scene
00046
00047 #endif // SCENE_MENU_SCENE_HPP_

```

7.85 src/scene/queue_scene.cpp File Reference

```

#include "queue_scene.hpp"
#include <cstddef>
#include <cstdlib>
#include <cstring>
#include <fstream>
#include <iostream>
#include <limits>
#include <string>
#include "constants.hpp"
#include "raygui.h"
#include "utils.hpp"

```


[illegible]

- namespace **scene**

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

Generated by Doxygen

```

00055     auto* const frame_ptr = m_sequence.find(frame_idx);
00056     m_sequence_controller.set_progress_value(frame_idx);
00057
00058     if (frame_ptr != nullptr) {
00059         frame_ptr->data.render();
00060         m_code_highlighter.highlight(frame_idx);
00061     } else { // end of sequence
00062         m_queue.render();
00063         m_sequence_controller.set_run_all(false);
00064     }
00065
00066     m_code_highlighter.render();
00067     m_sequence_controller.render();
00068     render_options(scene_options);
00069 }
00070
00071 void QueueScene::interact() {
00072     if (m_sequence_controller.interact()) {
00073         m_sequence_controller.reset_anim_counter();
00074         return;
00075     }
00076
00077     m_index_input.set_random_max((int)m_queue.size() - 1);
00078
00079     if (m_text_input.interact() || m_index_input.interact()) {
00080         return;
00081     }
00082
00083     if (!m_go) {
00084         return;
00085     }
00086
00087     int& mode = scene_options.mode_selection;
00088
00089     switch (mode) {
00090     case 0: {
00091         switch (scene_options.action_selection.at(mode)) {
00092         case 0: {
00093             interact_random();
00094             break;
00095         }
00096         case 1: {
00097             interact_import(m_text_input.extract_values());
00098             break;
00099         }
00100         case 2: {
00101             interact_file_import();
00102             break;
00103         }
00104         default:
00105             utils::unreachable();
00106         }
00107         break;
00108     }
00109     case 1: {
00110         interact_push();
00111         break;
00112     }
00113     case 2: {
00114         interact_pop();
00115         break;
00116     }
00117     default:
00118         utils::unreachable();
00119     }
00120
00121     m_go = false;
00122 }
00123
00124 void QueueScene::interact_random() {
00125     std::size_t size =
00126         utils::get_random(std::size_t{1}, scene_options.max_size);
00127     m_queue = gui::GuiQueue<int>();
00128
00129     for (auto i = 0; i < size; ++i) {
00130         m_queue.push(utils::get_random(constants::min_val, constants::max_val));
00131     }
00132     m_queue.init_label();
00133 }
00134
00135 void QueueScene::interact_import(core::Deque<int> nums) {
00136     m_sequence.clear();
00137     m_queue = gui::GuiQueue<int>();
00138
00139     while (!nums.empty()) {
00140         if (utils::val_in_range(nums.front())) {
00141             m_queue.push(nums.front());

```

```

00142     }
00143     nums.pop_front();
00144 }
00145 m_queue.init_label();
00146 }
00147
00148 void QueueScene::interact_file_import() {
00149     interact_import(m_file_dialog.extract_values());
00150 }
00151
00152 void QueueScene::interact_push() {
00153     auto value_container = m_text_input.extract_values();
00154     if (value_container.empty()) {
00155         return;
00156     }
00157
00158     int value = value_container.front();
00159
00160     if (m_queue.size() >= scene_options.max_size) {
00161         return;
00162     }
00163
00164     m_code_highlighter.set_code({
00165         "Node* node = new Node(value);",
00166         "tail->next = node;",
00167         "tail = tail->next;",
00168     });
00169
00170     m_sequence.clear();
00171     m_sequence.insert(m_sequence.size(), m_queue);
00172     m_code_highlighter.push_into_sequence(-1);
00173
00174     m_queue.push(value);
00175     m_queue.back().set_color_index(6);
00176     m_sequence.insert(m_sequence.size(), m_queue);
00177     m_code_highlighter.push_into_sequence(0);
00178
00179     m_queue.pop_back();
00180     if (!m_queue.empty()) {
00181         m_queue.back().set_color_index(4);
00182     }
00183     m_queue.push(value);
00184     m_queue.back().set_color_index(6);
00185     m_sequence.insert(m_sequence.size(), m_queue);
00186     m_code_highlighter.push_into_sequence(1);
00187
00188     m_queue.pop_back();
00189     if (!m_queue.empty()) {
00190         m_queue.back().set_color_index(0);
00191         m_queue.back().set_label("");
00192     }
00193     m_queue.push(value);
00194     m_queue.back().set_color_index(3);
00195     m_queue.init_label();
00196     m_sequence.insert(m_sequence.size(), m_queue);
00197     m_code_highlighter.push_into_sequence(2);
00198
00199     m_queue.back().set_color_index(0);
00200
00201     m_sequence_controller.set_max_value((int)m_sequence.size());
00202     m_sequence_controller.set_rerun();
00203 }
00204
00205 void QueueScene::interact_pop() {
00206     if (m_queue.empty()) {
00207         return;
00208     }
00209
00210     m_code_highlighter.set_code({
00211         "Node* temp = head;",
00212         "head = head->next;",
00213         "delete temp;",
00214     });
00215
00216     m_sequence.clear();
00217     m_sequence.insert(m_sequence.size(), m_queue);
00218     m_code_highlighter.push_into_sequence(-1);
00219
00220     m_queue.front().set_color_index(5);
00221     m_sequence.insert(m_sequence.size(), m_queue);
00222     m_code_highlighter.push_into_sequence(0);
00223
00224     auto old_front = m_queue.front();
00225     m_queue.pop();
00226
00227     if (!m_queue.empty()) {
00228         m_queue.front().set_color_index(3);

```

```

00229         if (m_queue.size() == 1) {
00230             m_queue.front().set_label("head/tail");
00231         } else {
00232             m_queue.front().set_label("head");
00233         }
00234     }
00235
00236     m_queue.push_front(old_front.get_value());
00237     m_queue.front().set_color_index(5);
00238     m_sequence.insert(m_sequence.size(), m_queue);
00239     m_code_highlighter.push_into_sequence(1);
00240
00241     m_queue.pop();
00242     m_queue.init_label();
00243     m_sequence.insert(m_sequence.size(), m_queue);
00244     m_code_highlighter.push_into_sequence(2);
00245
00246     if (!m_queue.empty()) {
00247         m_queue.front().set_color_index(0);
00248     }
00249
00250     m_sequence_controller.set_max_value((int)m_sequence.size());
00251     m_sequence_controller.set_rerun();
00252 }
00253
00254 } // namespace scene

```

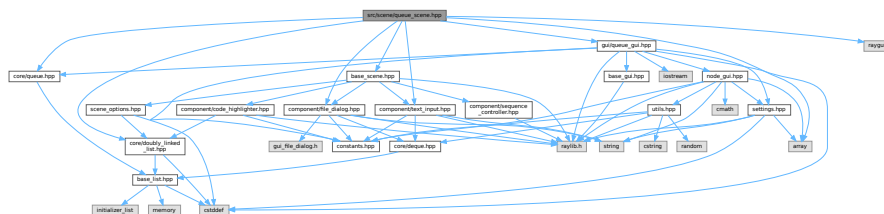
7.87 src/scene/queue_scene.hpp File Reference

```

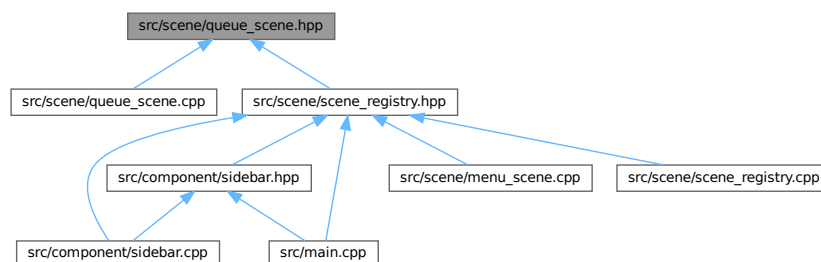
#include <array>
#include "base_scene.hpp"
#include "component/file_dialog.hpp"
#include "component/text_input.hpp"
#include "core/doubly_linked_list.hpp"
#include "core/queue.hpp"
#include "gui/queue_gui.hpp"
#include "raygui.h"

```

Include dependency graph for queue_scene.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [scene::QueueScene](#)

Namespaces

- namespace [scene](#)

7.88 queue_scene.hpp

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

```

00001 #ifndef SCENE_QUEUE_SCENE_HPP_
00002 #define SCENE_QUEUE_SCENE_HPP_
00003
00004 #include <array>
00005
00006 #include "base_scene.hpp"
00007 #include "component/file_dialog.hpp"
00008 #include "component/text_input.hpp"
00009 #include "core/doubly_linked_list.hpp"
00010 #include "core/queue.hpp"
00011 #include "gui/queue_gui.hpp"
00012 #include "raygui.h"
00013
00014 namespace scene {
00015
00016 class QueueScene : public internal::BaseScene {
00017 private:
00018     internal::SceneOptions scene_options{
00019         // max_size
00020         8, // NOLINT
00021
00022         // mode_labels
00023         "Mode: Create;"
00024         "Mode: Push;"
00025         "Mode: Pop",
00026
00027         // mode_selection
00028         0,
00029
00030         // action_labels
00031         {
00032             // Mode: Create
00033             "Action: Random;"
00034             "Action: Input;"
00035             "Action: File",
00036
00037             // Mode: Push
00038             "",
00039
00040             // Mode: Pop
00041             "",
00042         },
00043
00044         // action_selection
00045         core::DoublyLinkedList<int>{0, 0, 0},
00046     };
00047
00048     using internal::BaseScene::button_size;
00049     using internal::BaseScene::head_offset;
00050     using internal::BaseScene::options_head;
00051
00052     gui::GuiQueue<int> m_queue{
00053         gui::GuiNode<int>{1},
00054         gui::GuiNode<int>{2},
00055         gui::GuiNode<int>{3},
00056     };
00057     core::DoublyLinkedList<gui::GuiQueue<int>> m_sequence;
00058
00059     bool m_go{};
00060     using internal::BaseScene::m_code_highlighter;
00061     using internal::BaseScene::m_file_dialog;
00062     using internal::BaseScene::m_sequence_controller;
00063     using internal::BaseScene::m_text_input;
00064
00065     using internal::BaseScene::render_go_button;

```

```

00066     using internal::BaseScene::render_options;
00067     void render_inputs() override;
00068
00069     void interact_random();
00070     void interact_import(core::Deque<int> nums);
00071     void interact_file_import();
00072     void interact_push();
00073     void interact_pop();
00074
00075 public:
00076     void render() override;
00077     void interact() override;
00078 };
00079
00080 } // namespace scene
00081
00082 #endif // SCENE_QUEUE_SCENE_HPP_

```

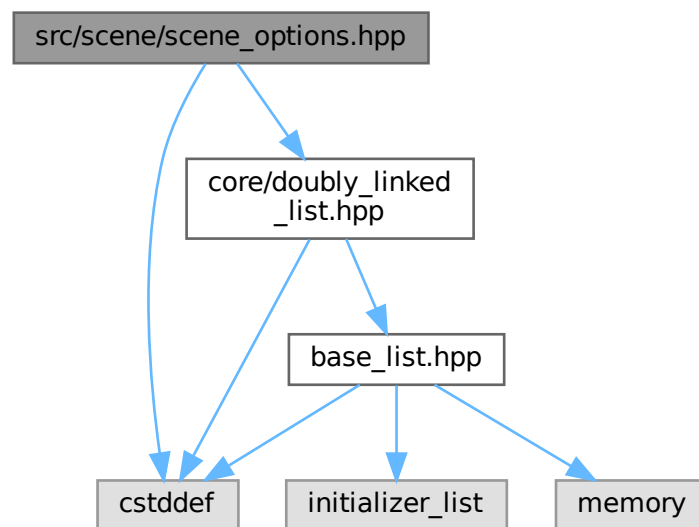
7.89 src/scene/scene_options.hpp File Reference

```

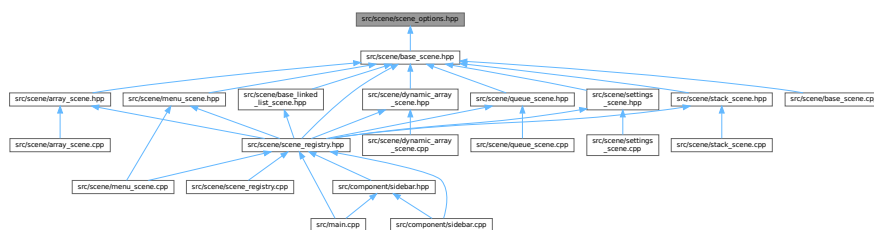
#include <cstddef>
#include "core/doubly_linked_list.hpp"

```

Include dependency graph for scene_options.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct `scene::internal::SceneOptions`

Namespaces

- namespace `scene`
- namespace `scene::internal`

7.90 scene_options.hpp

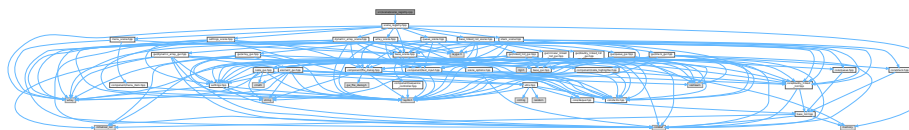
[Go to the documentation of this file.](#)

```
00001 #ifndef SCENE_SCENE_OPTIONS_HPP_
00002 #define SCENE_SCENE_OPTIONS_HPP_
00003
00004 #include <cstdint>
00005
00006 #include "core/doubly_linked_list.hpp"
00007
00008 namespace scene::internal {
00009
00010 struct SceneOptions {
00011     const std::size_t max_size{};
00012     const char* mode_labels{};
00013     int mode_selection{};
00014     core::DoublyLinkedList<const char*> action_labels;
00015     core::DoublyLinkedList<int> action_selection;
00016 };
00017
00018 } // namespace scene::internal
00019
00020 #endif // SCENE_SCENE_OPTIONS_HPP_
```

7.91 src/scene/scene_registry.cpp File Reference

```
#include "scene_registry.hpp"
```

Include dependency graph for scene_registry.cpp:



Namespaces

- namespace `scene`

7.92 scene_registry.cpp

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

```

00001 #include "scene_registry.hpp"
00002
00003 namespace scene {
00004
00005 SceneRegistry::SceneRegistry() { set_scene(Menu); }
00006
00007 SceneRegistry& SceneRegistry::get_instance() {
00008     static SceneRegistry registry;
00009     return registry;
00010 }
00011
00012 void SceneRegistry::set_scene(int scene_type) {
00013     m_current_scene = scene_type;
00014     scene_ptr = m_registry.at(scene_type).get();
00015 }
00016
00017 int SceneRegistry::get_scene() const { return m_current_scene; }
00018
00019 void SceneRegistry::render() { scene_ptr->render(); }
00020
00021 void SceneRegistry::interact() { scene_ptr->interact(); }
00022
00023 bool SceneRegistry::should_close() const { return m_should_close; }
00024
00025 void SceneRegistry::close_window() { m_should_close = true; }
00026
00027 } // namespace scene

```

7.93 src/scene/scene_registry.hpp File Reference

```

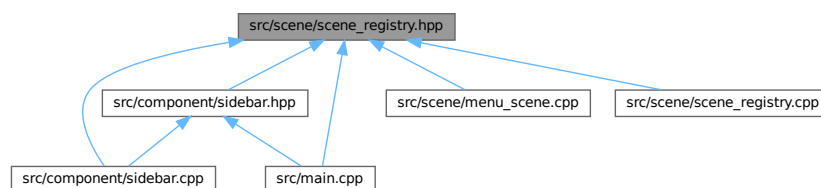
#include <array>
#include <memory>
#include "array_scene.hpp"
#include "base_linked_list_scene.hpp"
#include "base_scene.hpp"
#include "dynamic_array_scene.hpp"
#include "menu_scene.hpp"
#include "queue_scene.hpp"
#include "settings_scene.hpp"
#include "stack_scene.hpp"

```

Include dependency graph for scene_registry.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [scene::SceneRegistry](#)

Namespaces

- namespace [scene](#)

Enumerations

- enum [scene::SceneId](#) {
[scene::Array](#) , [scene::DynamicArray](#) , [scene::LinkedList](#) , [scene::DoublyLinkedList](#) ,
[scene::CircularLinkedList](#) , [scene::Stack](#) , [scene::Queue](#) , [scene::Menu](#) ,
[scene::Settings](#) }

7.94 scene_registry.hpp

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

```
00001 #ifndef SCENE_SCENE_REGISTRY_HPP_
00002 #define SCENE_SCENE_REGISTRY_HPP_
00003
00004 #include <array>
00005 #include <memory>
00006
00007 #include "array_scene.hpp"
00008 #include "base_linked_list_scene.hpp"
00009 #include "base_scene.hpp"
00010 #include "dynamic_array_scene.hpp"
00011 #include "menu_scene.hpp"
00012 #include "queue_scene.hpp"
00013 #include "settings_scene.hpp"
00014 #include "stack_scene.hpp"
00015
00016 namespace scene {
00017
00018     enum SceneId {
00019         Array,
00020         DynamicArray,
00021         LinkedList,
00022         DoublyLinkedList,
00023         CircularLinkedList,
00024         Stack,
00025         Queue,
00026         Menu,
00027         Settings,
00028     };
00029
00030     class SceneRegistry {
00031     private:
00032         internal::BaseScene* scene_ptr{};
00033         SceneRegistry();
00034
00035         bool m_should_close{};
00036         int m_current_scene{};
00037
00038         const std::array<const std::unique_ptr<internal::BaseScene>, 9> m_registry{{
00039             std::make_unique<ArrayScene>(),
00040             std::make_unique<DynamicArrayScene>(),
00041             std::make_unique<LinkedListScene>(),
00042             std::make_unique<DoublyLinkedListScene>(),
00043             std::make_unique<CircularLinkedListScene>(),
00044             std::make_unique<StackScene>(),
00045             std::make_unique<QueueScene>(),
00046             std::make_unique<MenuScene>(),
00047             std::make_unique<SettingsScene>(),
00048         }};
00049
00050     public:
00051         SceneRegistry(const SceneRegistry&) = delete;
00052         SceneRegistry(SceneRegistry&&) = delete;
00053         SceneRegistry& operator=(const SceneRegistry&) = delete;
```



```

00009
00010 #include "component/text_input.hpp"
00011 #include "constants.hpp"
00012 #include "raygui.h"
00013 #include "raylib.h"
00014 #include "settings.hpp"
00015 #include "utils.hpp"
00016
00017 namespace scene {
00018
00019 void SettingsScene::open_from_file(const std::string& path) {
00020     Settings& settings = Settings::get_instance();
00021     std::ifstream file_in(path, std::ios::binary);
00022
00023     if (!file_in.is_open()) {
00024         std::ofstream file_out(path, std::ios::binary);
00025
00026         for (auto i = 0; i < Settings::num_color; ++i) {
00027             unsigned value = Settings::default_color.at(i);
00028             file_out.write(reinterpret_cast<const char*>(&value),
00029                             sizeof(value));
00030         }
00031
00032         file_out.close();
00033
00034         file_in.close();
00035         file_in.open(path, std::ios::binary);
00036     }
00037
00038     unsigned hex_value;
00039     for (auto i = 0; i < Settings::num_color; ++i) {
00040         file_in.read(reinterpret_cast<char*>(&hex_value), sizeof(hex_value));
00041         settings.get_color(i) = GetColor(hex_value);
00042     }
00043
00044     set_buffer();
00045 }
00046
00047 SettingsScene::SettingsScene() {
00048     open_from_file(constants::default_color_path);
00049 }
00050
00051 void SettingsScene::set_buffer() {
00052     std::stringstream sstr;
00053
00054     for (auto i = 0; i < Settings::num_color; ++i) {
00055         sstr << std::setfill('0') << std::setw(6) << std::hex
00056             << ((unsigned)ColorToInt(Settings::get_instance().get_color(i)) >
00057                 8);
00058         std::strncpy(m_buffers.at(i), sstr.str().c_str(), 7);
00059         sstr.str(std::string());
00060     }
00061 }
00062
00063 void SettingsScene::set_color() {
00064     for (auto i = 0; i < Settings::num_color; ++i) {
00065         Settings::get_instance().get_color(i) =
00066             utils::color_from_hex(m_buffers.at(i));
00067     }
00068 }
00069
00070 void SettingsScene::render() {
00071     Settings& settings = Settings::get_instance();
00072     constexpr int second_col_x = constants::scene_width / 2 + head_pos.y;
00073     int second_col_y = 100;
00074     constexpr int vertical_gap = 30;
00075     const Color text_color =
00076         utils::adaptive_text_color(settings.get_color(Settings::num_color - 1));
00077
00078     auto [head_x, head_y] = head_pos;
00079
00080     for (auto i = 0; i < m_buffers.size(); ++i) {
00081         Rectangle input_shape;
00082         const char* text = nullptr;
00083
00084         if (i + 1 != m_buffers.size()) {
00085             input_shape = {(float)head_x, (float)head_y, input_size.x,
00086                             input_size.y};
00087             text = TextFormat("Color %d", i + 1);
00088         } else {
00089             input_shape = {(float)second_col_x, (float)second_col_y + 400,
00090                             input_size.x, input_size.y};
00091             text = "Background color";
00092         }
00093
00094         utils::DrawText(text, {(float)input_shape.x, (float)input_shape.y - 25},
00095                         text_color, 20, 2);
00096     }

```

```

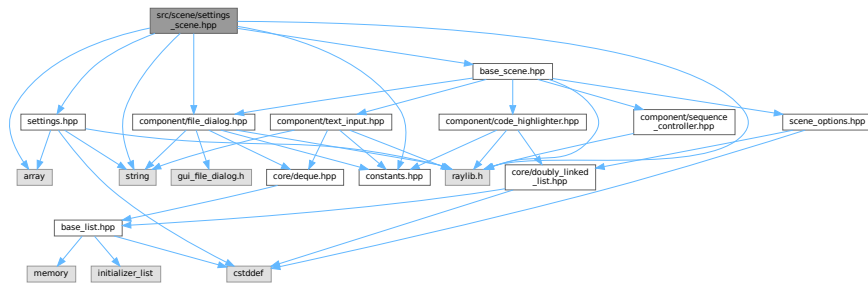
00096     DrawRectangleRec(input_shape, RAYWHITE);
00097     if (GuiTextBox(input_shape, m_buffers.at(i), 7, m_edit_mode.at(i))) {
00098         m_edit_mode.at(i) ^= 1;
00099     }
00100
00101     const Rectangle preview_shape{input_shape.x + input_size.x + 10,
00102                                   input_shape.y, input_size.y,
00103                                   input_size.y};
00104
00105     DrawRectangleRec(preview_shape, settings.get_color(i));
00106
00107     if (m_selected == i) {
00108         DrawRectangleLinesEx(preview_shape, 3, settings.get_color(5));
00109     } else {
00110         DrawRectangleLinesEx(preview_shape, 2, text_color);
00111     }
00112
00113     head_y += input_size.y + vertical_gap;
00114 }
00115
00116 {
00117     Color& color = settings.get_color(m_selected);
00118     auto new_color = GuiColorPicker({second_col_x, (float)second_col_y,
00119                                     4 * input_size.y, 4 * input_size.y},
00120                                     nullptr, color);
00121
00122     if (ColorToInt(color) != ColorToInt(new_color)) {
00123         color = new_color;
00124         set_buffer();
00125     }
00126 }
00127
00128 {
00129     second_col_y += 4 * input_size.y;
00130     utils::DrawText("Import config",
00131                     {second_col_x + 10, (float)second_col_y}, text_color,
00132                     20, 2);
00133     m_open = m_open_file.render(second_col_x, (float)second_col_y + 25);
00134 }
00135
00136 {
00137     second_col_y += component::FileDialog::size.y + vertical_gap;
00138     utils::DrawText("Export config",
00139                     {second_col_x + 10, (float)second_col_y}, text_color,
00140                     20, 2);
00141     m_save = m_save_file.render(second_col_x, (float)second_col_y + 25);
00142 }
00143 }
00144
00145 void SettingsScene::interact() {
00146     if (m_open > 0) {
00147         open_from_file(m_open_file.get_path());
00148         return;
00149     }
00150
00151     if (m_save > 0) {
00152         Settings::get_instance().save_to_file(m_save_file.get_path());
00153         return;
00154     }
00155
00156     const Vector2 mouse = GetMousePosition();
00157     const bool left_clicked = IsMouseButtonPressed(MOUSE_LEFT_BUTTON);
00158     auto [head_x, head_y] = head_pos;
00159
00160     for (auto i = 0; i < m_buffers.size(); ++i) {
00161         const Rectangle input_shape{(float)head_x, (float)head_y, input_size.x,
00162                                     input_size.y};
00163         const Rectangle preview_shape{input_shape.x + input_size.x + 10,
00164                                     input_shape.y, input_size.y,
00165                                     input_size.y};
00166
00167         if (m_edit_mode.at(i)) {
00168             m_selected = i;
00169         }
00170     }
00171
00172     set_color();
00173 }
00174
00175 } // namespace scene

```

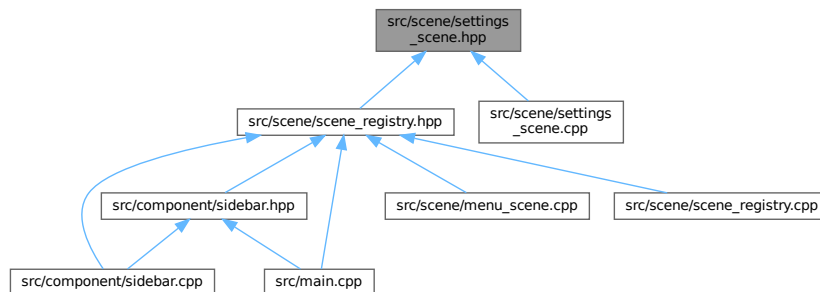
7.97 src/scene/settings_scene.hpp File Reference

```
#include <array>
#include <constants.hpp>
#include <string>
#include "base_scene.hpp"
#include "component/file_dialog.hpp"
#include "raylib.h"
#include "settings.hpp"
```

Include dependency graph for settings_scene.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [scene::SettingsScene](#)

Namespaces

- namespace [scene](#)

7.98 settings_scene.hpp

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

```

00001 #ifndef SCENE_SETTINGS_SCENE_HPP_
00002 #define SCENE_SETTINGS_SCENE_HPP_
00003
00004 #include <array>
00005 #include <constants.hpp>
00006 #include <string>
00007
00008 #include "base_scene.hpp"
00009 #include "component/file_dialog.hpp"
00010 #include "raylib.h"
00011 #include "settings.hpp"
00012
00013 namespace scene {
00014
00015 class SettingsScene : public internal::BaseScene {
00016 private:
00017     static constexpr Vector2 input_size{200, 50};
00018     static constexpr Vector2 head_pos{400, 70};
00019     std::array<char[7], Settings::num_color> m_buffers{};
00020     std::array<bool, Settings::num_color> m_edit_mode{};
00021
00022     int m_selected{};
00023
00024     component::FileDialog m_open_file;
00025     component::FileDialog m_save_file{3, "Save file...", "Save file"};
00026     int m_open{};
00027     int m_save{};
00028
00029     void set_buffer();
00030     void set_color();
00031     void open_from_file(const std::string& path);
00032
00033 public:
00034     SettingsScene();
00035
00036     void render() override;
00037     void interact() override;
00038 };
00039
00040 } // namespace scene
00041
00042 #endif // SCENE_SETTINGS_SCENE_HPP_

```

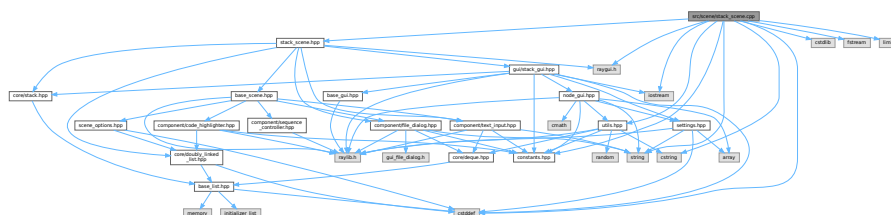
7.99 src/scene/stack_scene.cpp File Reference

```

#include "stack_scene.hpp"
#include <cstdint>
#include <cstdlib>
#include <cstring>
#include <fstream>
#include <iostream>
#include <limits>
#include <string>
#include "constants.hpp"
#include "raygui.h"
#include "utils.hpp"

```

Include dependency graph for stack_scene.cpp:



Namespaces

- namespace `scene`

7.100 stack_scene.cpp

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

```

00001 #include "stack_scene.hpp"
00002
00003 #include <cstdlib>
00004 #include <cstdlib>
00005 #include <cstring>
00006 #include <fstream>
00007 #include <iostream>
00008 #include <limits>
00009 #include <string>
00010
00011 #include "constants.hpp"
00012 #include "raygui.h"
00013 #include "utils.hpp"
00014
00015 namespace scene {
00016
00017 void StackScene::render() {
00018     m_sequence_controller.inc_anim_counter();
00019
00020     int frame_idx = m_sequence_controller.get_anim_frame();
00021     auto* const frame_ptr = m_sequence.find(frame_idx);
00022     m_sequence_controller.set_progress_value(frame_idx);
00023
00024     if (frame_ptr != nullptr) {
00025         frame_ptr->data.render();
00026         m_code_highlighter.highlight(frame_idx);
00027     } else { // end of sequence
00028         m_stack.render();
00029         m_sequence_controller.set_run_all(false);
00030     }
00031
00032     m_code_highlighter.render();
00033     m_sequence_controller.render();
00034     render_options(scene_options);
00035 }
00036
00037 void StackScene::render_inputs() {
00038     int& mode = scene_options.mode_selection;
00039
00040     switch (mode) {
00041         case 0: {
00042             switch (scene_options.action_selection.at(mode)) {
00043                 case 0:
00044                     break;
00045                 case 1: {
00046                     m_text_input.render(options_head, head_offset);
00047                 } break;
00048                 case 2: {
00049                     m_go = (m_file_dialog.render_head(options_head,
00050                                                         head_offset) > 0);
00051                     return;
00052                 } break;
00053                 default:
00054                     utils::unreachable();
00055             }
00056         } break;
00057
00058         case 1: {
00059             m_text_input.render(options_head, head_offset);
00060         } break;
00061
00062         case 2:
00063             break;
00064         default:
00065             utils::unreachable();
00066     }
00067
00068     m_go |= render_go_button();
00069 }
00070
00071 void StackScene::interact() {
00072     if (m_sequence_controller.interact()) {
00073         m_sequence_controller.reset_anim_counter();

```

```

00074         return;
00075     }
00076
00077     m_index_input.set_random_max((int)m_stack.size() - 1);
00078     if (m_text_input.interact() || m_index_input.interact()) {
00079         return;
00080     }
00081
00082     if (!m_go) {
00083         return;
00084     }
00085
00086     int& mode = scene_options.mode_selection;
00087
00088     switch (mode) {
00089         case 0: {
00090             switch (scene_options.action_selection.at(mode)) {
00091                 case 0: {
00092                     interact_random();
00093                 } break;
00094
00095                 case 1: {
00096                     interact_import(m_text_input.extract_values());
00097                 } break;
00098
00099                 case 2: {
00100                     interact_file_import();
00101                 } break;
00102
00103                 default:
00104                     utils::unreachable();
00105             }
00106         } break;
00107
00108         case 1: {
00109             interact_push();
00110         } break;
00111
00112         case 2: {
00113             interact_pop();
00114         } break;
00115
00116         default:
00117             utils::unreachable();
00118     }
00119
00120     m_go = false;
00121 }
00122
00123 void StackScene::interact_random() {
00124     std::size_t size =
00125         utils::get_random(std::size_t{1}, scene_options.max_size);
00126     m_stack = gui::GuiStack<int>();
00127
00128     for (auto i = 0; i < size; ++i) {
00129         m_stack.push(utils::get_random(constants::min_val, constants::max_val));
00130     }
00131     m_stack.init_label();
00132 }
00133
00134 void StackScene::interact_import(core::Deque<int> nums) {
00135     m_sequence.clear();
00136     m_stack = gui::GuiStack<int>();
00137
00138     while (!nums.empty()) {
00139         if (utils::val_in_range(nums.back())) {
00140             m_stack.push(nums.back());
00141         }
00142         nums.pop_back();
00143     }
00144     m_stack.init_label();
00145 }
00146
00147 void StackScene::interact_push() {
00148     auto value_container = m_text_input.extract_values();
00149     if (value_container.empty()) {
00150         return;
00151     }
00152
00153     int value = value_container.front();
00154
00155     if (m_stack.size() >= scene_options.max_size) {
00156         return;
00157     }
00158
00159     m_code_highlighter.set_code({
00160         "Node* node = new Node(value);",

```



```

00161         "node->next = head;",
00162         "head = node;",
00163     });
00164
00165     m_sequence.clear();
00166     m_sequence.insert(m_sequence.size(), m_stack);
00167     m_code_highlighter.push_into_sequence(-1);
00168
00169     m_stack.push(value);
00170     m_stack.top().set_color_index(6);
00171     m_sequence.insert(m_sequence.size(), m_stack);
00172     m_code_highlighter.push_into_sequence(0);
00173
00174     m_stack.pop();
00175     if (!m_stack.empty()) {
00176         m_stack.top().set_color_index(4);
00177     }
00178     m_stack.push(value);
00179     m_stack.top().set_color_index(6);
00180     m_sequence.insert(m_sequence.size(), m_stack);
00181     m_code_highlighter.push_into_sequence(1);
00182
00183     m_stack.pop();
00184     if (!m_stack.empty()) {
00185         m_stack.top().set_color_index(0);
00186         m_stack.top().set_label("");
00187     }
00188     m_stack.push(value);
00189     m_stack.top().set_color_index(3);
00190     m_stack.init_label();
00191     m_sequence.insert(m_sequence.size(), m_stack);
00192     m_code_highlighter.push_into_sequence(2);
00193
00194     m_stack.top().set_color_index(0);
00195
00196     m_sequence_controller.set_max_value((int)m_sequence.size());
00197     m_sequence_controller.set_rerun();
00198 }
00199
00200 void StackScene::interact_pop() {
00201     if (m_stack.empty()) {
00202         return;
00203     }
00204
00205     m_code_highlighter.set_code({
00206         "Node* temp = head;",
00207         "head = head->next;",
00208         "delete temp;",
00209     });
00210
00211     m_sequence.clear();
00212     m_sequence.insert(m_sequence.size(), m_stack);
00213     m_code_highlighter.push_into_sequence(-1);
00214
00215     m_stack.top().set_color_index(5);
00216     m_sequence.insert(m_sequence.size(), m_stack);
00217     m_code_highlighter.push_into_sequence(0);
00218
00219     auto old_top = m_stack.top();
00220     m_stack.pop();
00221
00222     if (!m_stack.empty()) {
00223         m_stack.top().set_color_index(3);
00224         m_stack.top().set_label("head");
00225     }
00226
00227     m_stack.push(old_top.get_value());
00228     m_stack.top().set_color_index(5);
00229     m_sequence.insert(m_sequence.size(), m_stack);
00230     m_code_highlighter.push_into_sequence(1);
00231
00232     m_stack.pop();
00233     m_sequence.insert(m_sequence.size(), m_stack);
00234     m_code_highlighter.push_into_sequence(2);
00235
00236     if (!m_stack.empty()) {
00237         m_stack.top().set_color_index(0);
00238     }
00239
00240     m_sequence_controller.set_max_value((int)m_sequence.size());
00241     m_sequence_controller.set_rerun();
00242 }
00243
00244 void StackScene::interact_file_import() {
00245     interact_import(m_file_dialog.extract_values());
00246 }
00247

```


7.102 stack_scene.hpp

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

```

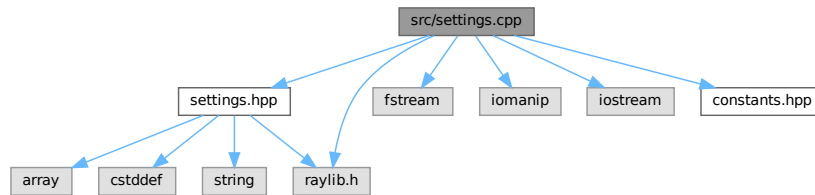
00001 #ifndef SCENE_STACK_SCENE_HPP_
00002 #define SCENE_STACK_SCENE_HPP_
00003
00004 #include "base_scene.hpp"
00005 #include "component/file_dialog.hpp"
00006 #include "component/text_input.hpp"
00007 #include "core/doubly_linked_list.hpp"
00008 #include "core/stack.hpp"
00009 #include "gui/stack_gui.hpp"
00010 #include "raygui.h"
00011
00012 namespace scene {
00013
00014 class StackScene : public internal::BaseScene {
00015 private:
00016     internal::SceneOptions scene_options{
00017         // max_size
00018         8, // NOLINT
00019
00020         // mode_labels
00021         "Mode: Create;",
00022         "Mode: Push;",
00023         "Mode: Pop",
00024
00025         // mode_selection
00026         0,
00027
00028         // action_labels
00029         {
00030             // Mode: Create
00031             "Action: Random;",
00032             "Action: Input;",
00033             "Action: File",
00034
00035             // Mode: Push
00036             "",
00037
00038             // Mode: Pop
00039             "",
00040         },
00041
00042         // action_selection
00043         core::DoublyLinkedList<int>{0, 0, 0},
00044     };
00045
00046     using internal::BaseScene::button_size;
00047     using internal::BaseScene::head_offset;
00048     using internal::BaseScene::options_head;
00049
00050     gui::GuiStack<int> m_stack{
00051         gui::GuiNode<int>{1},
00052         gui::GuiNode<int>{2},
00053         gui::GuiNode<int>{3},
00054     };
00055     core::DoublyLinkedList<gui::GuiStack<int>> m_sequence;
00056
00057     bool m_go{};
00058     using internal::BaseScene::m_code_highlighter;
00059     using internal::BaseScene::m_file_dialog;
00060     using internal::BaseScene::m_sequence_controller;
00061     using internal::BaseScene::m_text_input;
00062
00063     using internal::BaseScene::render_go_button;
00064     using internal::BaseScene::render_options;
00065     void render_inputs() override;
00066
00067     void interact_random();
00068     void interact_import(core::Deque<int> nums);
00069     void interact_push();
00070     void interact_pop();
00071     void interact_file_import();
00072
00073 public:
00074     void render() override;
00075     void interact() override;
00076 };
00077
00078 } // namespace scene
00079
00080 #endif // SCENE_STACK_SCENE_HPP_

```

7.103 src/settings.cpp File Reference

```
#include "settings.hpp"
#include <fstream>
#include <iomanip>
#include <iostream>
#include "constants.hpp"
#include "raylib.h"
```

Include dependency graph for settings.cpp:



7.104 settings.cpp

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

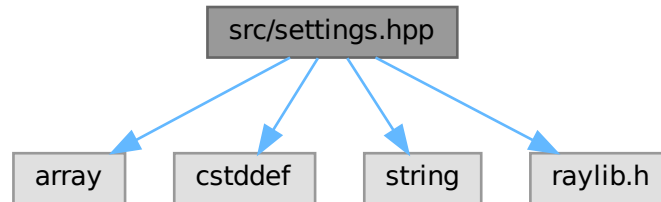
```
00001 #include "settings.hpp"
00002
00003 #include <fstream>
00004 #include <iomanip>
00005 #include <iostream>
00006
00007 #include "constants.hpp"
00008 #include "raylib.h"
00009
00010 Settings& Settings::get_instance() {
00011     static Settings settings;
00012     return settings;
00013 }
00014
00015 void Settings::save_to_file(const std::string& path) {
00016     std::ofstream file_out(path, std::ios::binary);
00017
00018     for (auto i = 0; i < num_color; ++i) {
00019         unsigned value = ColorToInt(m_colors.at(i));
00020         file_out.write(reinterpret_cast<const char*>(&value), sizeof(value));
00021     }
00022 }
00023
00024 Settings::~Settings() { save_to_file(constants::default_color_path); }
00025
00026 Color& Settings::get_color(std::size_t index) { return m_colors.at(index); }
00027
00028 Color Settings::get_color(std::size_t index) const {
00029     return m_colors.at(index);
00030 }
```

7.105 src/settings.hpp File Reference

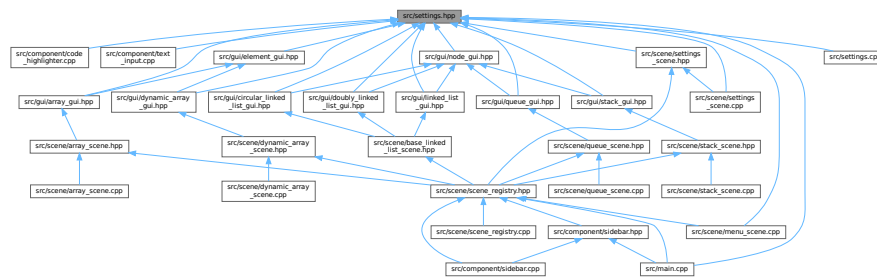
```
#include <array>
#include <cstdint>
#include <string>
```

```
#include "raylib.h"
```

Include dependency graph for settings.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class Settings

7.106 settings.hpp

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

```
00001 #ifndef SETTINGS_HPP_
00002 #define SETTINGS_HPP_
00003
00004 #include <array>
00005 #include <cstdint>
00006 #include <string>
00007
00008 #include "raylib.h"
00009
00010 class Settings {
00011 public:
00012     static constexpr int num_color = 9;
00013     static constexpr std::array<unsigned, num_color> default_color{
00014         0x00000000,
00015         0x82828200,
00016         0xffa10000,
00017         0x00e43000,
00018         0x873cbe00,
00019         0xe6293700,
00020         0x0079f100,
00021         0xff6dc200,
00022         0xf5f5f500,
```

```

00023     };
00024
00025 private:
00026     Settings() = default;
00027     std::array<Color, num_color> m_colors{};
00028
00029 public:
00030     Settings(const Settings&) = delete;
00031     Settings(Settings&&) = delete;
00032     Settings& operator=(const Settings&) = delete;
00033     Settings& operator=(Settings&&) = delete;
00034     ~Settings();
00035
00036     static Settings& get_instance();
00037
00038     Color& get_color(std::size_t index);
00039     Color get_color(std::size_t index) const;
00040
00041     void save_to_file(const std::string& path);
00042 };
00043
00044 #endif // SETTINGS_HPP_

```

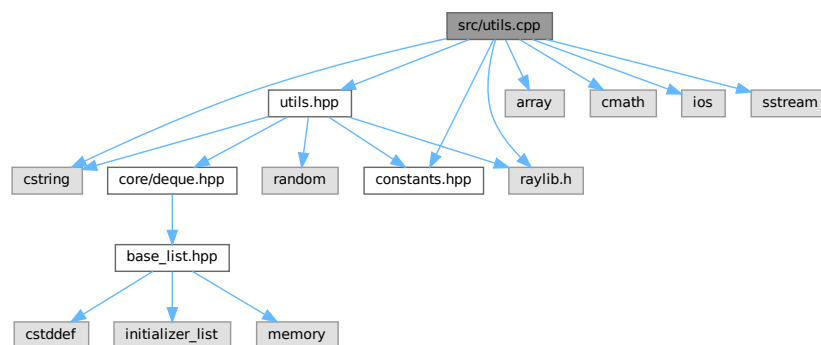
7.107 src/utils.cpp File Reference

```

#include "utils.hpp"
#include <array>
#include <cmath>
#include <cstring>
#include <ios>
#include <sstream>
#include "constants.hpp"
#include "raylib.h"

```

Include dependency graph for utils.cpp:



Namespaces

- namespace [utils](#)

Functions

- void [utils::DrawText](#) (const char *text, Vector2 pos, Color color, float font_size, float spacing)
- Vector2 [utils::MeasureText](#) (const char *text, float font_size, float spacing)
- [core::Deque< int > utils::str_extract_data](#) (char str[[constants::text_buffer_size](#)])

- bool `utils::val_in_range` (int num)
- void `utils::unreachable` ()
- char * `utils::strtok` (char *str, const char *delim, char **save_ptr)
- Color `utils::color_from_hex` (const std::string &hex)
- Color `utils::adaptive_text_color` (Color bg_color)

7.108 utils.cpp

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

```
00001 #include "utils.hpp"
00002
00003 #include <array>
00004 #include <cmath>
00005 #include <cstring>
00006 #include <ios>
00007 #include <sstream>
00008
00009 #include "constants.hpp"
00010 #include "raylib.h"
00011
00012 namespace utils {
00013
00014 void DrawText(const char* text, Vector2 pos, Color color, float font_size,
00015              float spacing) {
00016     static Font font = LoadFontEx("data/open_sans.ttf",
00017                                   constants::default_font_size, nullptr, 0);
00018
00019     Vector2 pos_vec{static_cast<float>(pos.x), static_cast<float>(pos.y)};
00020     DrawTextEx(font, text, pos_vec, font_size, spacing, color);
00021 }
00022
00023 Vector2 MeasureText(const char* text, float font_size, float spacing) {
00024     static Font font = LoadFontEx("data/open_sans.ttf",
00025                                   constants::default_font_size, nullptr, 0);
00026
00027     return MeasureTextEx(font, text, font_size, spacing);
00028 }
00029
00030 core::Deque<int> str_extract_data(
00031     char str[constants::text_buffer_size]) { // NOLINT
00032     char str_copy[constants::text_buffer_size];
00033     strncpy(str_copy, str, constants::text_buffer_size);
00034
00035     char* save_ptr = nullptr;
00036     char* token = utils::strtok(str_copy, ",", &save_ptr);
00037
00038     if (token == nullptr) {
00039         return {};
00040     }
00041
00042     core::Deque<int> ret;
00043
00044     constexpr int base = 10;
00045     int num = static_cast<int>(std::strtol(token, nullptr, base));
00046     ret.push_back(num);
00047
00048     while (true) {
00049         token = utils::strtok(nullptr, ",", &save_ptr);
00050         if (token == nullptr) {
00051             break;
00052         }
00053
00054         num = static_cast<int>(std::strtol(token, nullptr, base));
00055         ret.push_back(num);
00056     }
00057
00058     return ret;
00059 }
00060
00061 bool val_in_range(int num) {
00062     return constants::min_val <= num && num <= constants::max_val;
00063 }
00064
00065 void unreachable() {
00066     #if defined(_MSC_VER)
00067         __assume(0);
00068     #else
00069         __builtin_unreachable();
00070     #endif
00071 }
```

```

00071 }
00072
00073 char* strtok(char* str, const char* delim, char** save_ptr) {
00074     return
00075     #if defined(_MSC_VER)
00076         strtok_s(str, delim, save_ptr);
00077     #else
00078         strtok_r(str, delim, save_ptr);
00079     #endif
00080 }
00081
00082 Color color_from_hex(const std::string& hex) {
00083     std::stringstream stream(hex + "ff");
00084     unsigned int value;
00085     stream » std::hex » value;
00086     return GetColor(value);
00087 }
00088
00089 // https://stackoverflow.com/a/3943023
00090 Color adaptive_text_color(Color bg_color) {
00091     constexpr std::array<float, 3> threshold{{0.2126, 0.7152, 0.0722}};
00092     const std::array<int, 3> colors = {{bg_color.r, bg_color.g, bg_color.b}};
00093     float sum = 0;
00094
00095     for (auto i = 0; i < 3; ++i) {
00096         float value = (float)colors.at(i) / 255.0F;
00097         if (value <= 0.04045) {
00098             value /= 12.92;
00099         } else {
00100             value = std::pow(((value + 0.055) / 1.055), 2.4);
00101         }
00102
00103         sum += value;
00104     }
00105
00106     return (sum > 0.179) ? BLACK : WHITE;
00107 }
00108
00109 } // namespace utils

```

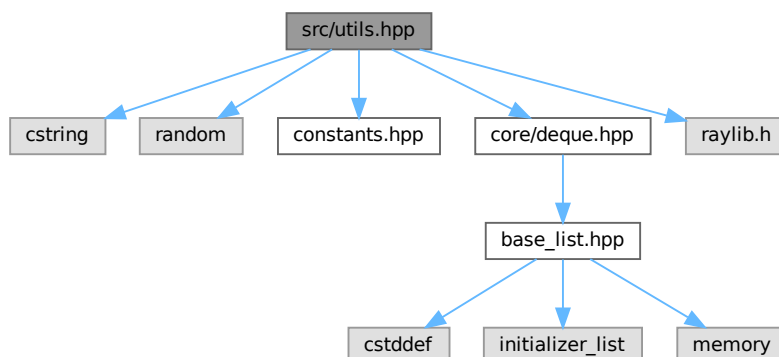
7.109 src/utils.hpp File Reference

```

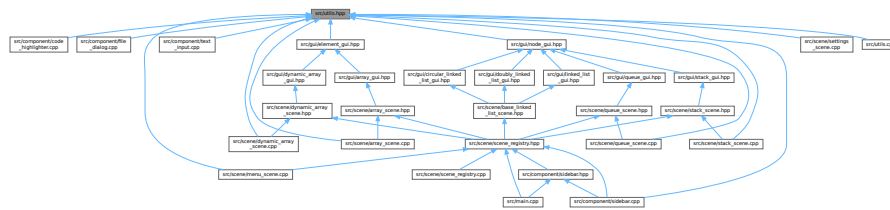
#include <cstring>
#include <random>
#include "constants.hpp"
#include "core/deque.hpp"
#include "raylib.h"

```

Include dependency graph for utils.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **utils**

Functions

- void `utils::DrawText` (const char *text, Vector2 pos, Color color, float font_size, float spacing)
- Vector2 `utils::MeasureText` (const char *text, float font_size, float spacing)
- template<typename T >
T `utils::get_random` (T low, T high)
- `core::Deque< int > utils::str_extract_data` (char str[constants::text_buffer_size])
- bool `utils::val_in_range` (int num)
- void `utils::unreachable` ()
- char * `utils::strtok` (char *str, const char *delim, char **save_ptr)
- Color `utils::color_from_hex` (const std::string &hex)
- Color `utils::adaptive_text_color` (Color bg_color)

7.110 utils.hpp

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

```
00001 #ifndef UTILS_HPP_
00002 #define UTILS_HPP_
00003
00004 #include <cstring>
00005 #include <random>
00006
00007 #include "constants.hpp"
00008 #include "core/deque.hpp"
00009 #include "raylib.h"
00010
00011 namespace utils {
00012
00013 void DrawText(const char* text, Vector2 pos, Color color, float font_size,
00014              float spacing);
00015
00016 Vector2 MeasureText(const char* text, float font_size, float spacing);
00017
00018 template<typename T>
00019 T get_random(T low, T high) {
00020     if (low > high) {
00021         return low;
00022     }
00023
00024     static std::random_device ran_dev;
00025     static std::mt19937 prng(ran_dev());
00026     std::uniform_int_distribution<T> dist{low, high};
00027     return dist(prng);
00028 }
00029
00030 core::Deque<int> str_extract_data(
00031     char str[constants::text_buffer_size]); // NOLINT
```

```
00032
00033 bool val_in_range(int num);
00034
00035 void unreachable();
00036
00037 char* strtok(char* str, const char* delim, char** save_ptr);
00038
00039 Color color_from_hex(const std::string& hex);
00040
00041 Color adaptive_text_color(Color bg_color);
00042
00043 } // namespace utils
00044
00045 #endif // UTILS_HPP_
```

Index

- `__attribute__`
 - `deque.test.cpp`, 209
- `~Base`
 - `gui::internal::Base`, 27
- `~BaseList`
 - `core::BaseList< T >`, 35
- `~BaseScene`
 - `scene::internal::BaseScene`, 41
- `~GuiDynamicArray`
 - `gui::GuiDynamicArray< T >`, 95
- `~SceneRegistry`
 - `scene::SceneRegistry`, 146
- `~Settings`
 - `Settings`, 160
- `action_labels`
 - `scene::internal::SceneOptions`, 144
- `action_selection`
 - `scene::internal::SceneOptions`, 144
- `adaptive_text_color`
 - `utils`, 14
- `ani_speed`
 - `constants`, 9
- `Array`
 - `scene`, 13
- `at`
 - `core::DoublyLinkedList< T >`, 61
- `back`
 - `core::BaseList< T >`, 35
 - `core::Deque< T >`, 53
 - `core::Queue< T >`, 137
- `Base`
 - `core::DoublyLinkedList< T >`, 60
 - `core::Stack< T >`, 174
 - `gui::internal::Base`, 26, 27
- `BaseList`
 - `core::BaseList< T >`, 34, 35
- `BaseScene`
 - `scene::internal::BaseScene`, 40, 41
- `block_height`
 - `component::MenuItem`, 127
- `block_width`
 - `component::MenuItem`, 127
- `button_height`
 - `component::MenuItem`, 127
- `button_size`
 - `scene::internal::BaseScene`, 43
- `button_width`
 - `component::MenuItem`, 127
- `capacity`
 - `gui::GuiDynamicArray< T >`, 95
- `CircularLinkedList`
 - `scene`, 13
- `CircularLinkedListScene`
 - `scene`, 13
- `clean_up`
 - `core::BaseList< T >`, 36
- `clear`
 - `component::CodeHighlighter`, 46
 - `core::DoublyLinkedList< T >`, 61
- `clicked`
 - `component::MenuItem`, 126
- `close_window`
 - `scene::SceneRegistry`, 146
- `cNode_ptr`
 - `core::DoublyLinkedList< T >`, 60
- `color_from_hex`
 - `utils`, 15
- `component`, 9
- `component::CodeHighlighter`, 45
 - `clear`, 46
 - `highlight`, 46
 - `push_into_sequence`, 47
 - `render`, 48
 - `set_code`, 49
- `component::FileDialog`, 69
 - `extract_values`, 71
 - `FileDialog`, 70, 71
 - `get_path`, 71
 - `is_active`, 72
 - `render`, 72
 - `render_head`, 72
 - `set_message`, 73
 - `set_mode_open`, 73
 - `set_mode_save`, 73
 - `set_title`, 73
 - `size`, 74
- `component::MenuItem`, 124
 - `block_height`, 127
 - `block_width`, 127
 - `button_height`, 127
 - `button_width`, 127
 - `clicked`, 126
 - `MenuItem`, 125, 126
 - `render`, 126
 - `reset`, 126
 - `x`, 126
 - `y`, 127

- component::SequenceController, 150
 - get_anim_counter, 151
 - get_anim_frame, 151
 - get_progress_value, 152
 - get_run_all, 153
 - get_speed_scale, 153
 - inc_anim_counter, 154
 - interact, 155
 - render, 155
 - reset_anim_counter, 156
 - set_max_value, 157
 - set_progress_value, 157
 - set_rerun, 158
 - set_run_all, 158
- component::SideBar, 168
 - interact, 168
 - render, 169
- component::TextInput, 179
 - extract_values, 181
 - interact, 181
 - render, 182
 - set_random_max, 183
 - set_random_min, 183
 - size, 184
 - TextInput, 180
- constants, 9
 - ani_speed, 9
 - default_color_path, 10
 - default_font_size, 10
 - frames_per_second, 10
 - max_val, 10
 - min_val, 10
 - scene_height, 10
 - scene_width, 11
 - sidebar_width, 11
 - text_buffer_size, 11
- copy_data
 - core::BaseList< T >, 36
- core, 11
- core::BaseList< T >, 32
 - ~BaseList, 35
 - back, 35
 - BaseList, 34, 35
 - clean_up, 36
 - copy_data, 36
 - empty, 36
 - front, 36
 - init_first_element, 36
 - m_head, 38
 - m_size, 38
 - m_tail, 38
 - Node_ptr, 34
 - operator=, 37
 - pop_back, 37
 - pop_front, 37
 - push_back, 37
 - push_front, 38
 - size, 38
- core::BaseList< T >::Node, 132
 - data, 132
 - next, 133
 - prev, 133
- core::Deque< T >, 49
 - back, 53
 - empty, 53
 - front, 53
 - pop_back, 54
 - pop_front, 54
 - push_back, 55
 - push_front, 55
 - size, 56
- core::DoublyLinkedList< T >, 57
 - at, 61
 - Base, 60
 - clear, 61
 - cNode_ptr, 60
 - empty, 62
 - find, 62
 - insert, 63
 - internal_find, 63
 - internal_search, 63
 - m_head, 65
 - m_size, 65
 - m_tail, 65
 - Node, 60
 - Node_ptr, 60
 - remove, 63
 - search, 64
 - size, 64
- core::Queue< T >, 133
 - back, 137
 - empty, 137
 - front, 137
 - pop, 137
 - pop_back, 137
 - push, 137
 - push_front, 138
 - size, 138
- core::Stack< T >, 170
 - Base, 174
 - empty, 174
 - m_head, 175
 - m_tail, 175
 - pop, 174
 - push, 174
 - size, 174
 - top, 175
- data
 - core::BaseList< T >::Node, 132
- default_color
 - Settings, 163
- default_color_path
 - constants, 10
- default_font_size
 - constants, 10
- deque.test.cpp

- `__attribute__`, 209
- `list`, 210
- `TEST_CASE`, 209, 210
- `DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN`
 - `doctest_main.cpp`, 222
- `doctest_main.cpp`
 - `DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN`, 222
- `doubly_linked_list.test.cpp`
 - `TEST_CASE`, 216
- `DoublyLinkedList`
 - `scene`, 13
- `DoublyLinkedListScene`
 - `scene`, 13
- `DrawText`
 - `utils`, 15
- `DynamicArray`
 - `scene`, 13
- `empty`
 - `core::BaseList< T >`, 36
 - `core::Deque< T >`, 53
 - `core::DoublyLinkedList< T >`, 62
 - `core::Queue< T >`, 137
 - `core::Stack< T >`, 174
- `extract_values`
 - `component::FileDialog`, 71
 - `component::TextInput`, 181
- `FileDialog`
 - `component::FileDialog`, 70, 71
- `find`
 - `core::DoublyLinkedList< T >`, 62
- `frames_per_second`
 - `constants`, 10
- `front`
 - `core::BaseList< T >`, 36
 - `core::Deque< T >`, 53
 - `core::Queue< T >`, 137
- `get_anim_counter`
 - `component::SequenceController`, 151
- `get_anim_frame`
 - `component::SequenceController`, 151
- `get_color`
 - `Settings`, 161
- `get_instance`
 - `scene::SceneRegistry`, 146
 - `Settings`, 162
- `get_path`
 - `component::FileDialog`, 71
- `get_pos`
 - `gui::GuiElement< T >`, 100
 - `gui::GuiNode< T >`, 110
- `get_progress_value`
 - `component::SequenceController`, 152
- `get_random`
 - `utils`, 16
- `get_run_all`
 - `component::SequenceController`, 153
- `get_scene`
 - `scene::SceneRegistry`, 147
- `get_speed_scale`
 - `component::SequenceController`, 153
- `get_value`
 - `gui::GuiElement< T >`, 100
 - `gui::GuiNode< T >`, 110
- `gui`, 11
- `gui::GuiArray< T, N >`, 74
 - `GuiArray`, 77
 - `operator[]`, 77, 78
 - `render`, 78
 - `set_color_index`, 78
 - `update`, 78
- `gui::GuiCircularLinkedList< T >`, 79
 - `GuiCircularLinkedList`, 83
 - `init_label`, 84
 - `insert`, 84
 - `render`, 84
 - `update`, 85
- `gui::GuiDoublyLinkedList< T >`, 85
 - `GuiDoublyLinkedList`, 89
 - `init_label`, 90
 - `insert`, 90
 - `render`, 90
 - `update`, 91
- `gui::GuiDynamicArray< T >`, 91
 - `~GuiDynamicArray`, 95
 - `capacity`, 95
 - `GuiDynamicArray`, 94, 95
 - `operator=`, 95
 - `operator[]`, 96
 - `pop`, 96
 - `push`, 96
 - `realloc`, 96
 - `render`, 97
 - `set_color_index`, 97
 - `size`, 98
 - `update`, 98
- `gui::GuiElement< T >`, 99
 - `get_pos`, 100
 - `get_value`, 100
 - `GuiElement`, 100
 - `init_pos`, 102
 - `render`, 100
 - `set_color_index`, 101
 - `set_index`, 101
 - `set_pos`, 102
 - `set_value`, 102
 - `side`, 102
- `gui::GuiLinkedList< T >`, 103
 - `GuiLinkedList`, 107
 - `init_label`, 108
 - `insert`, 108
 - `render`, 108
 - `update`, 109
- `gui::GuiNode< T >`, 109

- get_pos, 110
- get_value, 110
- GuiNode, 110
- radius, 112
- render, 111
- set_color_index, 111
- set_label, 111
- set_pos, 112
- set_value, 112
- gui::GuiQueue< T >, 113
 - GuiQueue, 116
 - init_label, 117
 - pop, 117
 - pop_back, 117
 - push, 117
 - push_front, 117
 - render, 118
 - update, 118
- gui::GuiStack< T >, 119
 - GuiStack, 122
 - init_label, 122
 - pop, 123
 - push, 123
 - render, 123
 - update, 124
- gui::internal, 12
- gui::internal::Base, 25
 - ~Base, 27
 - Base, 26, 27
 - operator=, 27
 - render, 27
 - update, 28
- GUI_FILE_DIALOG_IMPLEMENTATION
 - raygui_impl.cpp, 251
- GuiArray
 - gui::GuiArray< T, N >, 77
- GuiCircularLinkedList
 - gui::GuiCircularLinkedList< T >, 83
- GuiDoublyLinkedList
 - gui::GuiDoublyLinkedList< T >, 89
- GuiDynamicArray
 - gui::GuiDynamicArray< T >, 94, 95
- GuiElement
 - gui::GuiElement< T >, 100
- GuiLinkedList
 - gui::GuiLinkedList< T >, 107
- GuiNode
 - gui::GuiNode< T >, 110
- GuiQueue
 - gui::GuiQueue< T >, 116
- GuiStack
 - gui::GuiStack< T >, 122
- head_offset
 - scene::internal::BaseScene, 43
- highlight
 - component::CodeHighlighter, 46
- inc_anim_counter
 - component::SequenceController, 154
- init_first_element
 - core::BaseList< T >, 36
- init_label
 - gui::GuiCircularLinkedList< T >, 84
 - gui::GuiDoublyLinkedList< T >, 90
 - gui::GuiLinkedList< T >, 108
 - gui::GuiQueue< T >, 117
 - gui::GuiStack< T >, 122
- init_pos
 - gui::GuiElement< T >, 102
- insert
 - core::DoublyLinkedList< T >, 63
 - gui::GuiCircularLinkedList< T >, 84
 - gui::GuiDoublyLinkedList< T >, 90
 - gui::GuiLinkedList< T >, 108
- interact
 - component::SequenceController, 155
 - component::SideBar, 168
 - component::TextInput, 181
 - scene::ArrayScene, 24
 - scene::BaseLinkedListScene< Con >, 31
 - scene::DynamicArrayScene, 68
 - scene::internal::BaseScene, 41
 - scene::MenuScene, 130
 - scene::QueueScene, 141
 - scene::SceneRegistry, 147
 - scene::SettingsScene, 166
 - scene::StackScene, 178
- internal_find
 - core::DoublyLinkedList< T >, 63
- internal_search
 - core::DoublyLinkedList< T >, 63
- is_active
 - component::FileDialog, 72
- LinkedList
 - scene, 13
- LinkedListScene
 - scene, 13
- list
 - deque.test.cpp, 210
- m_code_highlighter
 - scene::internal::BaseScene, 44
- m_edit_action
 - scene::internal::BaseScene, 44
- m_edit_mode
 - scene::internal::BaseScene, 44
- m_file_dialog
 - scene::internal::BaseScene, 44
- m_head
 - core::BaseList< T >, 38
 - core::DoublyLinkedList< T >, 65
 - core::Stack< T >, 175
- m_index_input
 - scene::internal::BaseScene, 44
- m_sequence_controller
 - scene::internal::BaseScene, 44

- m_size
 - core::BaseList< T >, 38
 - core::DoublyLinkedList< T >, 65
- m_tail
 - core::BaseList< T >, 38
 - core::DoublyLinkedList< T >, 65
 - core::Stack< T >, 175
- m_text_input
 - scene::internal::BaseScene, 45
- main
 - main.cpp, 249
- main.cpp
 - main, 249
- max_size
 - scene::internal::SceneOptions, 144
- max_val
 - constants, 10
- MeasureText
 - utils, 16
- Menu
 - scene, 13
- MenuItem
 - component::MenuItem, 125, 126
- MenuScene
 - scene::MenuScene, 130
- min_val
 - constants, 10
- mode_labels
 - scene::internal::SceneOptions, 144
- mode_selection
 - scene::internal::SceneOptions, 144
- next
 - core::BaseList< T >::Node, 133
- Node
 - core::DoublyLinkedList< T >, 60
- Node_ptr
 - core::BaseList< T >, 34
 - core::DoublyLinkedList< T >, 60
- num_color
 - Settings, 163
- operator=
 - core::BaseList< T >, 37
 - gui::GuiDynamicArray< T >, 95
 - gui::internal::Base, 27
 - scene::internal::BaseScene, 41
 - scene::SceneRegistry, 148
 - Settings, 162
- operator[]
 - gui::GuiArray< T, N >, 77, 78
 - gui::GuiDynamicArray< T >, 96
- options_head
 - scene::internal::BaseScene, 45
- pop
 - core::Queue< T >, 137
 - core::Stack< T >, 174
 - gui::GuiDynamicArray< T >, 96
 - gui::GuiQueue< T >, 117
 - gui::GuiStack< T >, 123
- pop_back
 - core::BaseList< T >, 37
 - core::Deque< T >, 54
 - core::Queue< T >, 137
 - gui::GuiQueue< T >, 117
- pop_front
 - core::BaseList< T >, 37
 - core::Deque< T >, 54
- prev
 - core::BaseList< T >::Node, 133
- push
 - core::Queue< T >, 137
 - core::Stack< T >, 174
 - gui::GuiDynamicArray< T >, 96
 - gui::GuiQueue< T >, 117
 - gui::GuiStack< T >, 123
- push_back
 - core::BaseList< T >, 37
 - core::Deque< T >, 55
- push_front
 - core::BaseList< T >, 38
 - core::Deque< T >, 55
 - core::Queue< T >, 138
 - gui::GuiQueue< T >, 117
- push_into_sequence
 - component::CodeHighlighter, 47
- Queue
 - scene, 13
- radius
 - gui::GuiNode< T >, 112
- raygui_impl.cpp
 - GUI_FILE_DIALOG_IMPLEMENTATION, 251
 - RAYGUI_IMPLEMENTATION, 251
- RAYGUI_IMPLEMENTATION
 - raygui_impl.cpp, 251
- realloc
 - gui::GuiDynamicArray< T >, 96
- remove
 - core::DoublyLinkedList< T >, 63
- render
 - component::CodeHighlighter, 48
 - component::FileDialog, 72
 - component::MenuItem, 126
 - component::SequenceController, 155
 - component::SideBar, 169
 - component::TextInput, 182
 - gui::GuiArray< T, N >, 78
 - gui::GuiCircularLinkedList< T >, 84
 - gui::GuiDoublyLinkedList< T >, 90
 - gui::GuiDynamicArray< T >, 97
 - gui::GuiElement< T >, 100
 - gui::GuiLinkedList< T >, 108
 - gui::GuiNode< T >, 111
 - gui::GuiQueue< T >, 118
 - gui::GuiStack< T >, 123

- gui::internal::Base, 27
- scene::ArrayScene, 24
- scene::BaseLinkedListScene< Con >, 31
- scene::DynamicArrayScene, 68
- scene::internal::BaseScene, 42
- scene::MenuScene, 131
- scene::QueueScene, 141
- scene::SceneRegistry, 148
- scene::SettingsScene, 167
- scene::StackScene, 178
- render_go_button
 - scene::internal::BaseScene, 42
- render_head
 - component::FileDialog, 72
- render_inputs
 - scene::internal::BaseScene, 42
- render_options
 - scene::internal::BaseScene, 42
- reset
 - component::MenuItem, 126
- reset_anim_counter
 - component::SequenceController, 156
- save_to_file
 - Settings, 163
- scene, 12
 - Array, 13
 - CircularLinkedList, 13
 - CircularLinkedListScene, 13
 - DoublyLinkedList, 13
 - DoublyLinkedListScene, 13
 - DynamicArray, 13
 - LinkedList, 13
 - LinkedListScene, 13
 - Menu, 13
 - Queue, 13
 - Sceneld, 13
 - Settings, 13
 - Stack, 13
- scene::ArrayScene, 21
 - interact, 24
 - render, 24
- scene::BaseLinkedListScene< Con >, 28
 - interact, 31
 - render, 31
- scene::DynamicArrayScene, 66
 - interact, 68
 - render, 68
- scene::internal, 14
- scene::internal::BaseScene, 39
 - ~BaseScene, 41
 - BaseScene, 40, 41
 - button_size, 43
 - head_offset, 43
 - interact, 41
 - m_code_highlighter, 44
 - m_edit_action, 44
 - m_edit_mode, 44
 - m_file_dialog, 44
 - m_index_input, 44
 - m_sequence_controller, 44
 - m_text_input, 45
 - operator=, 41
 - options_head, 45
 - render, 42
 - render_go_button, 42
 - render_inputs, 42
 - render_options, 42
- scene::internal::SceneOptions, 142
 - action_labels, 144
 - action_selection, 144
 - max_size, 144
 - mode_labels, 144
 - mode_selection, 144
- scene::MenuScene, 128
 - interact, 130
 - MenuScene, 130
 - render, 131
- scene::QueueScene, 139
 - interact, 141
 - render, 141
- scene::SceneRegistry, 145
 - ~SceneRegistry, 146
 - close_window, 146
 - get_instance, 146
 - get_scene, 147
 - interact, 147
 - operator=, 148
 - render, 148
 - SceneRegistry, 146
 - set_scene, 149
 - should_close, 149
- scene::SettingsScene, 164
 - interact, 166
 - render, 167
 - SettingsScene, 166
- scene::StackScene, 176
 - interact, 178
 - render, 178
- scene_height
 - constants, 10
- scene_width
 - constants, 11
- Sceneld
 - scene, 13
- SceneRegistry
 - scene::SceneRegistry, 146
- search
 - core::DoublyLinkedList< T >, 64
- set_code
 - component::CodeHighlighter, 49
- set_color_index
 - gui::GuiArray< T, N >, 78
 - gui::GuiDynamicArray< T >, 97
 - gui::GuiElement< T >, 101
 - gui::GuiNode< T >, 111
- set_index

- gui::GuiElement< T >, 101
- set_label
 - gui::GuiNode< T >, 111
- set_max_value
 - component::SequenceController, 157
- set_message
 - component::FileDialog, 73
- set_mode_open
 - component::FileDialog, 73
- set_mode_save
 - component::FileDialog, 73
- set_pos
 - gui::GuiElement< T >, 102
 - gui::GuiNode< T >, 112
- set_progress_value
 - component::SequenceController, 157
- set_random_max
 - component::TextInput, 183
- set_random_min
 - component::TextInput, 183
- set_rerun
 - component::SequenceController, 158
- set_run_all
 - component::SequenceController, 158
- set_scene
 - scene::SceneRegistry, 149
- set_title
 - component::FileDialog, 73
- set_value
 - gui::GuiElement< T >, 102
 - gui::GuiNode< T >, 112
- Settings, 159
 - ~Settings, 160
 - default_color, 163
 - get_color, 161
 - get_instance, 162
 - num_color, 163
 - operator=, 162
 - save_to_file, 163
 - scene, 13
 - Settings, 160
- SettingsScene
 - scene::SettingsScene, 166
- should_close
 - scene::SceneRegistry, 149
- side
 - gui::GuiElement< T >, 102
- sidebar_width
 - constants, 11
- size
 - component::FileDialog, 74
 - component::TextInput, 184
 - core::BaseList< T >, 38
 - core::Deque< T >, 56
 - core::DoublyLinkedList< T >, 64
 - core::Queue< T >, 138
 - core::Stack< T >, 174
 - gui::GuiDynamicArray< T >, 98
 - src/component/code_highlighter.cpp, 185
 - src/component/code_highlighter.hpp, 186, 187
 - src/component/file_dialog.cpp, 188
 - src/component/file_dialog.hpp, 189, 190
 - src/component/menu_item.cpp, 191
 - src/component/menu_item.hpp, 192, 193
 - src/component/sequence_controller.cpp, 193, 194
 - src/component/sequence_controller.hpp, 195, 196
 - src/component/sidebar.cpp, 197
 - src/component/sidebar.hpp, 198, 199
 - src/component/text_input.cpp, 199, 200
 - src/component/text_input.hpp, 201, 202
 - src/constants.hpp, 202, 203
 - src/core/base_list.hpp, 203, 204
 - src/core/deque.hpp, 207
 - src/core/deque.test.cpp, 208, 211
 - src/core/doubly_linked_list.hpp, 212, 213
 - src/core/doubly_linked_list.test.cpp, 215, 217
 - src/core/queue.hpp, 218, 219
 - src/core/stack.hpp, 220, 221
 - src/doctest_main.cpp, 221, 222
 - src/gui/array_gui.hpp, 222, 223
 - src/gui/base_gui.hpp, 224, 225
 - src/gui/circular_linked_list_gui.hpp, 226, 227
 - src/gui/doubly_linked_list_gui.hpp, 228, 230
 - src/gui/dynamic_array_gui.hpp, 231, 232
 - src/gui/element_gui.hpp, 235, 236
 - src/gui/linked_list_gui.hpp, 238, 239
 - src/gui/node_gui.hpp, 240, 241
 - src/gui/queue_gui.hpp, 243, 244
 - src/gui/stack_gui.hpp, 246, 247
 - src/main.cpp, 249, 250
 - src/raygui_impl.cpp, 250, 251
 - src/scene/array_scene.cpp, 251, 252
 - src/scene/array_scene.hpp, 255, 256
 - src/scene/base_linked_list_scene.hpp, 257, 258
 - src/scene/base_scene.cpp, 267
 - src/scene/base_scene.hpp, 268, 269
 - src/scene/dynamic_array_scene.cpp, 270
 - src/scene/dynamic_array_scene.hpp, 274, 275
 - src/scene/menu_scene.cpp, 276, 277
 - src/scene/menu_scene.hpp, 279, 280
 - src/scene/queue_scene.cpp, 280, 281
 - src/scene/queue_scene.hpp, 284, 285
 - src/scene/scene_options.hpp, 286, 287
 - src/scene/scene_registry.cpp, 287, 288
 - src/scene/scene_registry.hpp, 288, 289
 - src/scene/settings_scene.cpp, 290
 - src/scene/settings_scene.hpp, 293, 294
 - src/scene/stack_scene.cpp, 294, 295
 - src/scene/stack_scene.hpp, 298, 299
 - src/settings.cpp, 300
 - src/settings.hpp, 300, 301
 - src/utils.cpp, 302, 303
 - src/utils.hpp, 304, 305
 - Stack
 - scene, 13
 - str_extract_data

- utils, [17](#)
- strtok
 - utils, [18](#)
- TEST_CASE
 - deque.test.cpp, [209](#), [210](#)
 - doubly_linked_list.test.cpp, [216](#)
- text_buffer_size
 - constants, [11](#)
- TextInput
 - component::TextInput, [180](#)
- top
 - core::Stack< T >, [175](#)
- unreachable
 - utils, [18](#)
- update
 - gui::GuiArray< T, N >, [78](#)
 - gui::GuiCircularLinkedList< T >, [85](#)
 - gui::GuiDoublyLinkedList< T >, [91](#)
 - gui::GuiDynamicArray< T >, [98](#)
 - gui::GuiLinkedList< T >, [109](#)
 - gui::GuiQueue< T >, [118](#)
 - gui::GuiStack< T >, [124](#)
 - gui::internal::Base, [28](#)
- utils, [14](#)
 - adaptive_text_color, [14](#)
 - color_from_hex, [15](#)
 - DrawText, [15](#)
 - get_random, [16](#)
 - MeasureText, [16](#)
 - str_extract_data, [17](#)
 - strtok, [18](#)
 - unreachable, [18](#)
 - val_in_range, [19](#)
- val_in_range
 - utils, [19](#)
- x
 - component::MenuItem, [126](#)
- y
 - component::MenuItem, [127](#)