CS162 - Visualizer

Generated by Doxygen 1.9.6

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

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 SceneId {
  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 SceneId

```
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)

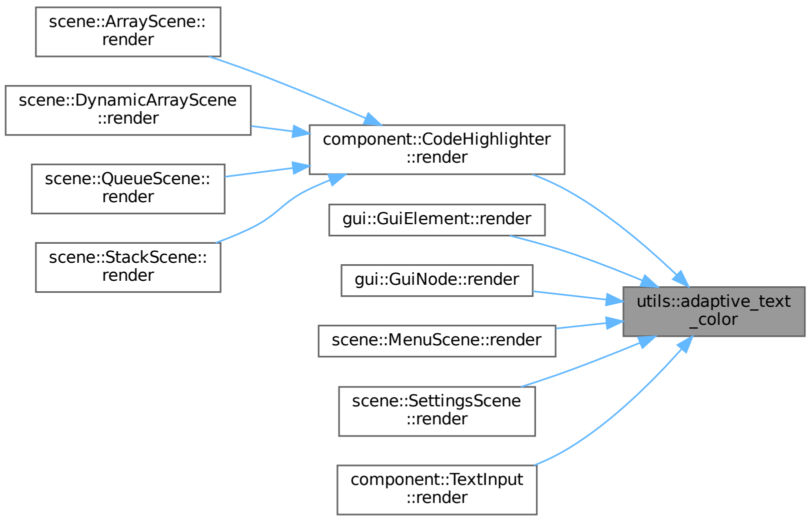
5.8.1 Function Documentation

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:


scene::ArrayScene::render
scene::DynamicArrayScene::render
scene::QueueScene::render
scene::StackScene::render
component::CodeHighlighter::render
gui::GuiElement::render
gui::GuiNode::render
scene::MenuScene::render
scene::SettingsScene::render
component::TextInput::render
utils::adaptive_text_color

5.8.1.2 color_from_hex()

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

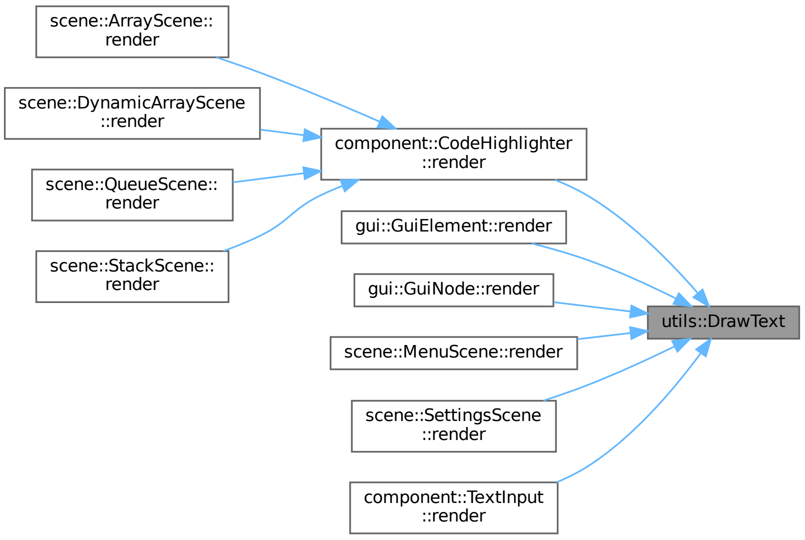
Definition at line 82 of file utils.cpp.

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:

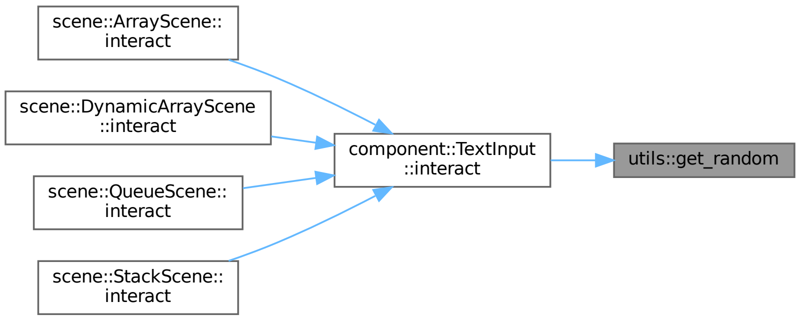

scene::ArrayScene::render
scene::DynamicArrayScene::render
scene::QueueScene::render
scene::StackScene::render
component::CodeHighlighter::render
gui::GuiElement::render
gui::GuiNode::render
scene::MenuScene::render
scene::SettingsScene::render
component::TextInput::render
utils::DrawText

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:

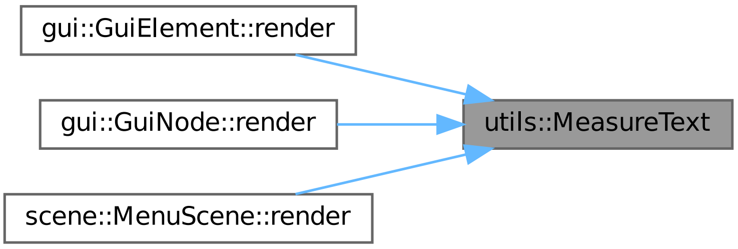

scene::ArrayScene::interact
scene::DynamicArrayScene::interact
scene::QueueScene::interact
scene::StackScene::interact
component::TextInput::interact
utils::get_random

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:

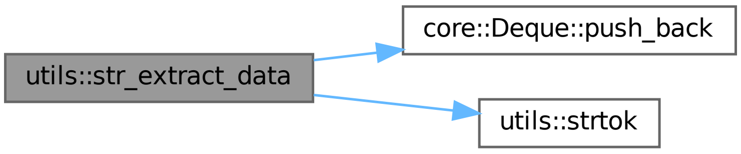


5.8.1.6 str_extract_data()

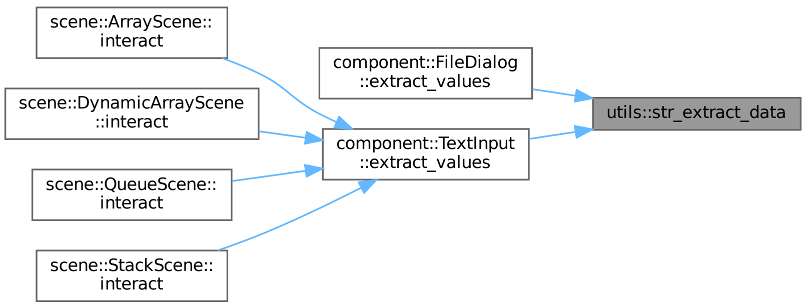
```
core::Deque< int > utils::str_extract_data (
             char str[constants::text_buffer_size] )
```

Definition at line 30 of file utils.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

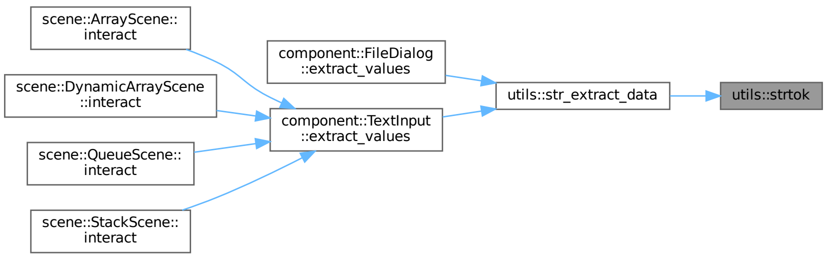


5.8.1.7 strtok()

```
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:

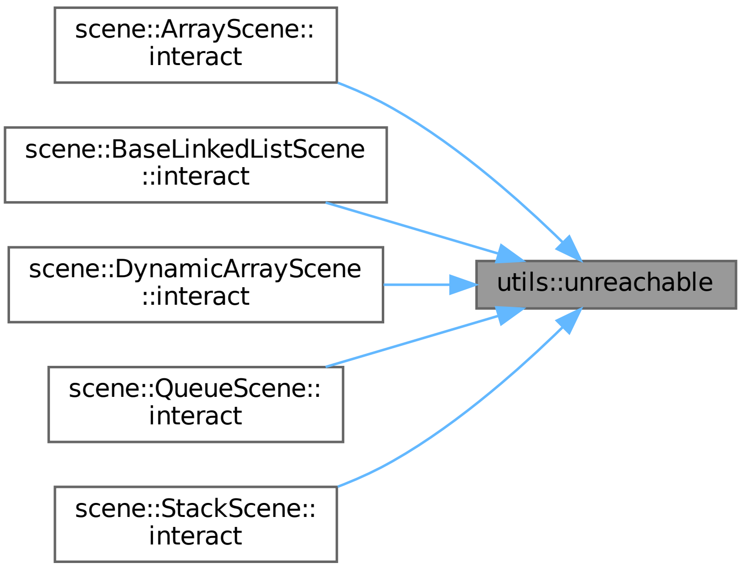


5.8.1.8 unreachable()

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

Definition at line 65 of file utils.cpp.

Here is the caller graph for this function:


scene::ArrayScene::interact
scene::BaseLinkedListScene::interact
scene::DynamicArrayScene::interact
scene::QueueScene::interact
scene::StackScene::interact
utils::unreachable

5.8.1.9 val_in_range()

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

Definition at line 61 of file utils.cpp.

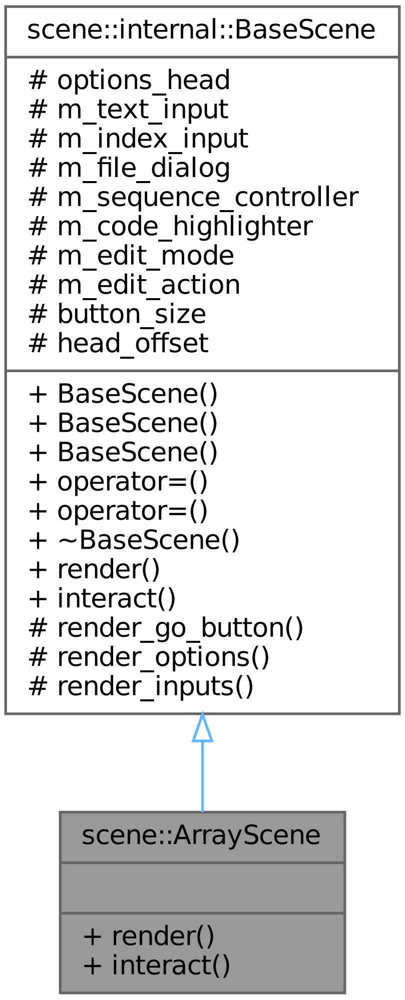
Chapter 6

Class Documentation

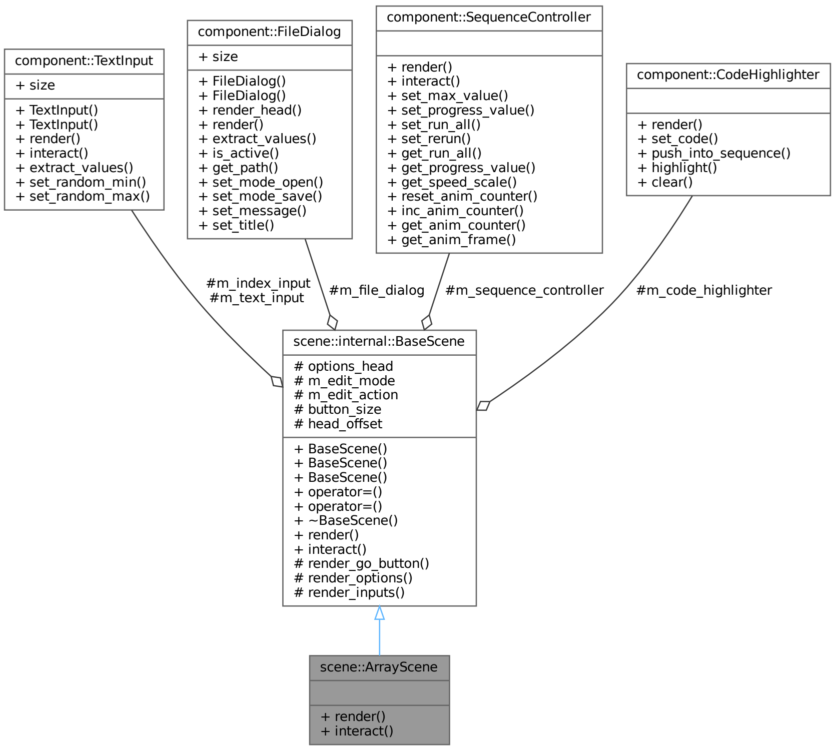
6.1 scene::ArrayScene Class Reference

```
#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

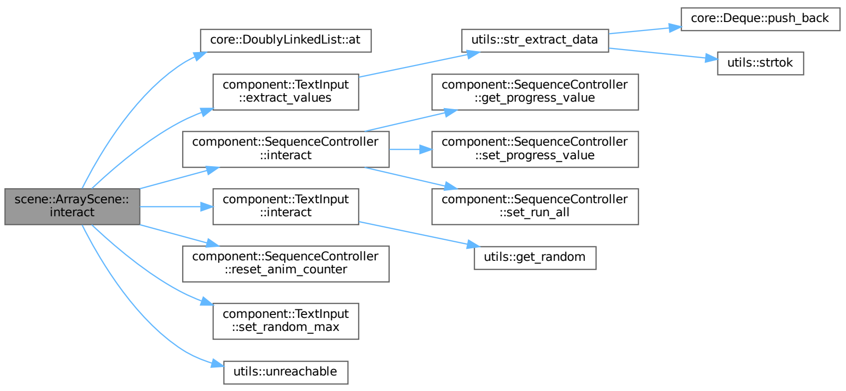
6.1.2.1 interact()

```
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:



scene::ArrayScene::interact
core::DoublyLinkedList::at
component::TextInput ::extract_values
component::SequenceController ::interact
component::TextInput ::interact
component::SequenceController ::reset_anim_counter
component::TextInput ::set_random_max
utils::unreachable
utils::str_extract_data
component::SequenceController ::get_progress_value
component::SequenceController ::set_progress_value
component::SequenceController ::set_run_all
utils::get_random
core::Deque::push_back
utils::strtok

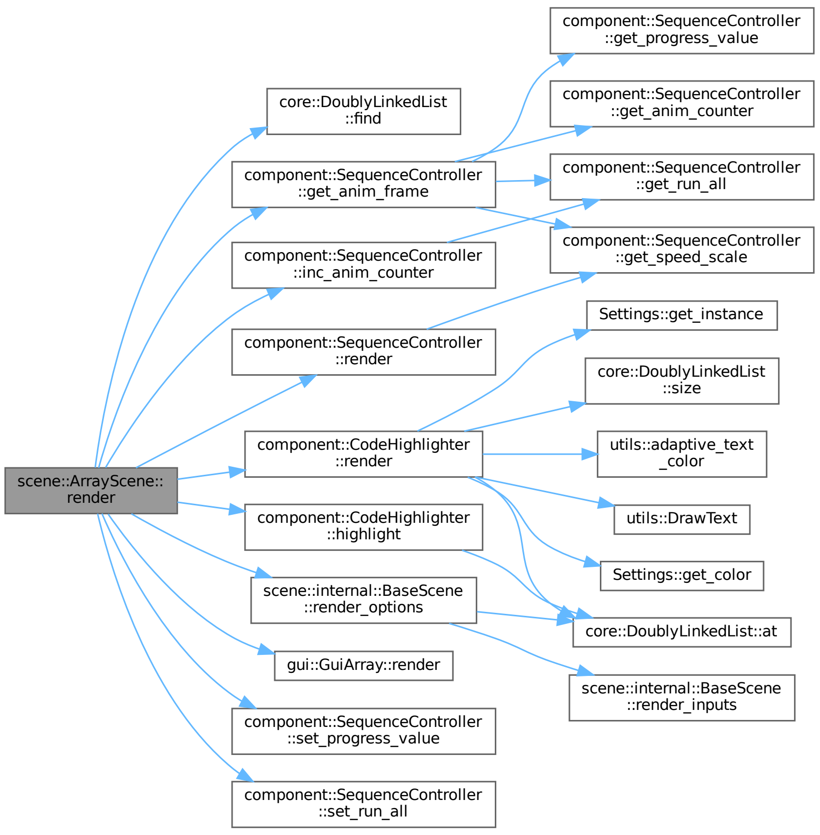
6.1.2.2 render()

```
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:


scene::ArrayScene:: render
core::DoublyLinkedList ::find
component::SequenceController ::get_anim_frame
component::SequenceController ::inc_anim_counter
component::SequenceController ::render
component::CodeHighlighter ::render
component::CodeHighlighter ::highlight
scene::internal::BaseScene ::render_options
gui::GuiArray::render
component::SequenceController ::set_progress_value
component::SequenceController ::set_run_all
component::SequenceController ::get_progress_value
component::SequenceController ::get_anim_counter
component::SequenceController ::get_run_all
component::SequenceController ::get_speed_scale
Settings::get_instance
core::DoublyLinkedList ::size
utils::adaptive_text _color
utils::DrawText
Settings::get_color
core::DoublyLinkedList::at
scene::internal::BaseScene ::render_inputs

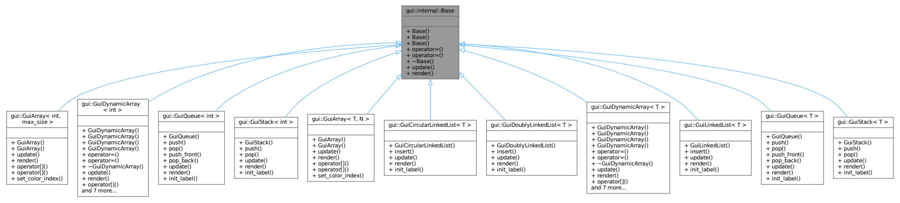
The documentation for this class was generated from the following files:

- src/scene/array_scene.hpp
- src/scene/array_scene.cpp

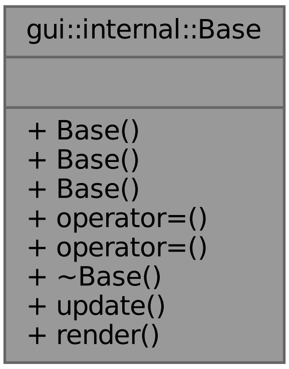
6.2 gui::internal::Base Class Reference

```
#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 >.

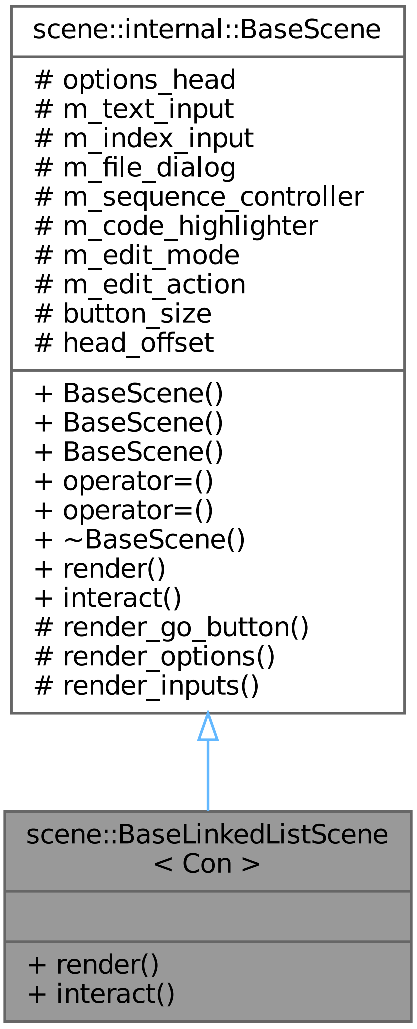
The documentation for this class was generated from the following file:

- src/gui/base_gui.hpp

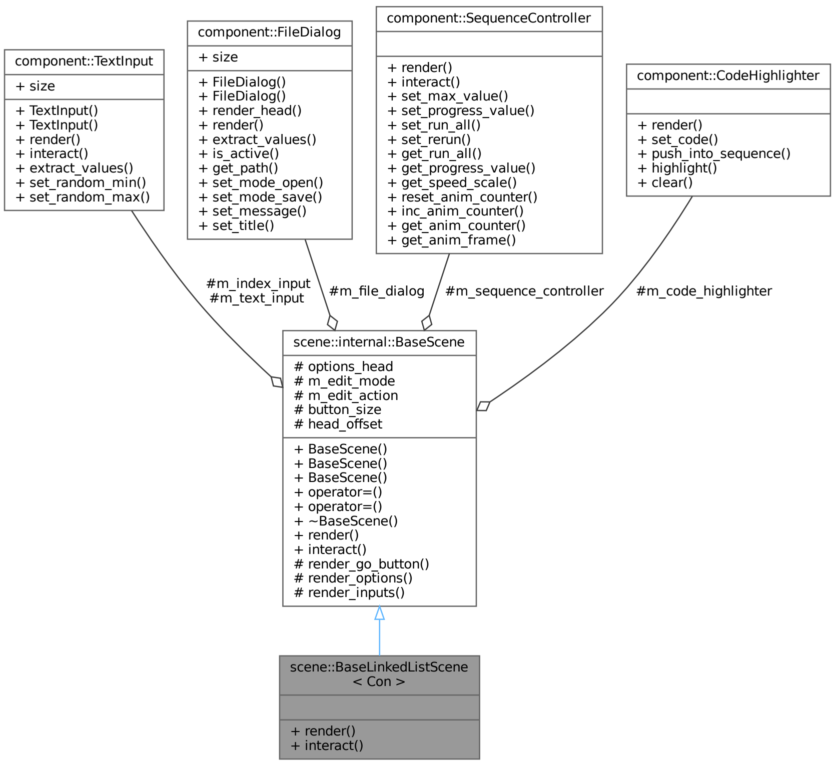
6.3 scene::BaseLinkedListScene< Con > Class Template Reference

```
#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

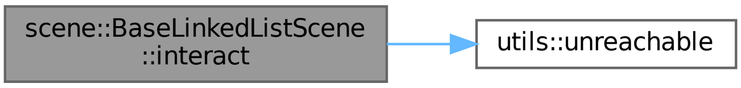
6.3.2.1 interact()

```
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:



6.3.2.2 render()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 149 of file base_linked_list_scene.hpp.

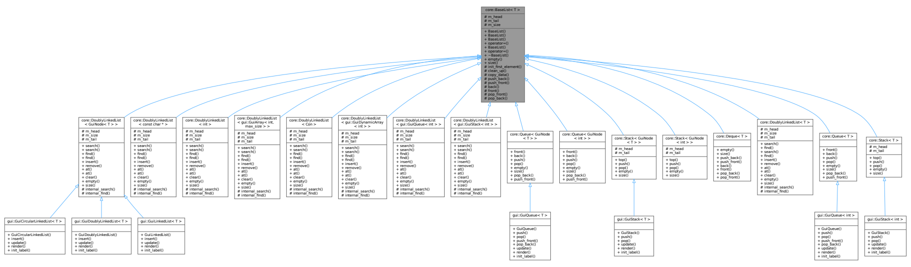
The documentation for this class was generated from the following file:

- src/scene/base_linked_list_scene.hpp

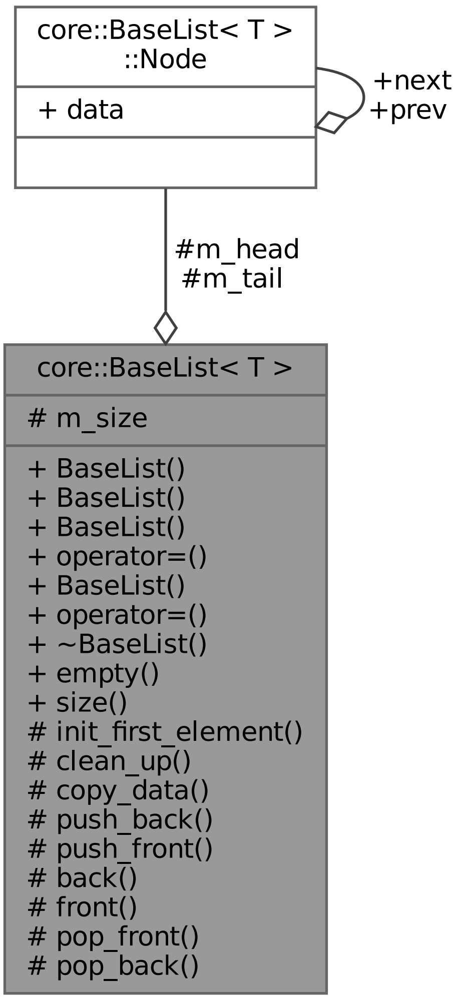
6.4 core::BaseList< T > Class Template Reference

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

Inheritance diagram for core::BaseList< T >:



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.

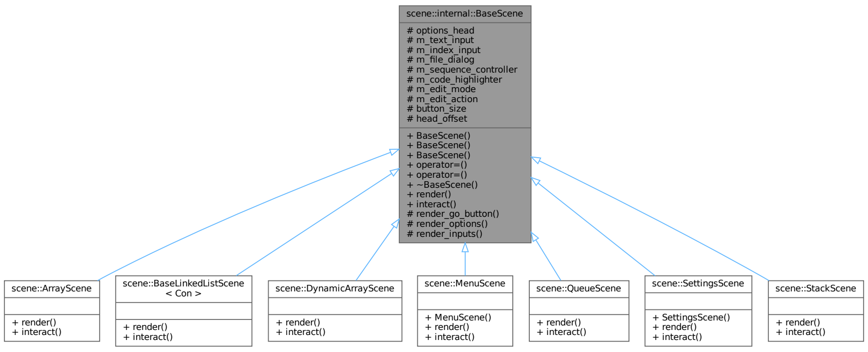
The documentation for this class was generated from the following file:

- src/core/base_list.hpp

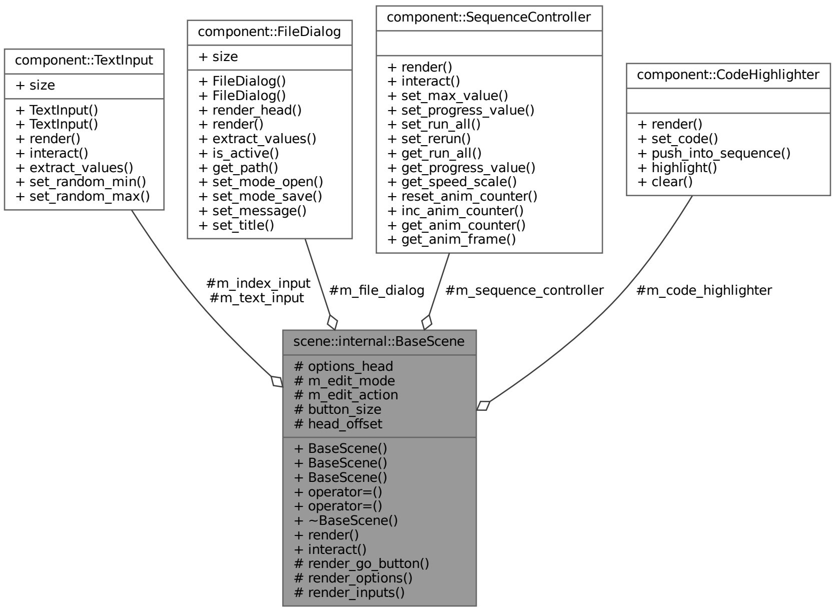
6.5 scene::internal::BaseScene Class Reference

```
#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

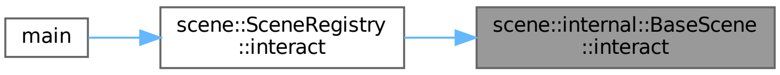
6.5.3.1 interact()

```
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:


main
scene::SceneRegistry ::interact
scene::internal::BaseScene ::interact

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]
```

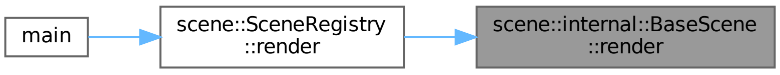
6.5.3.4 render()

```
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:


main
scene::SceneRegistry ::render
scene::internal::BaseScene ::render

6.5.3.5 render_go_button()

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

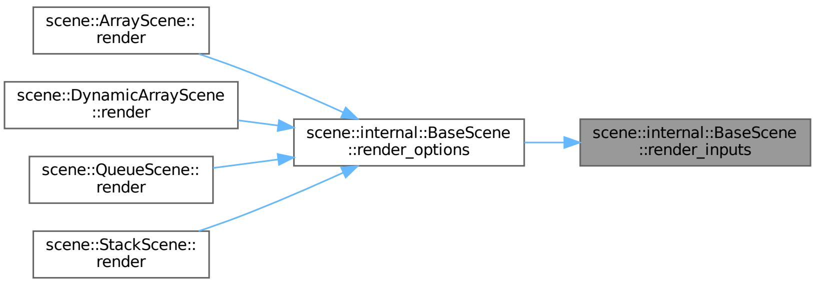
Definition at line 10 of file base_scene.cpp.

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:

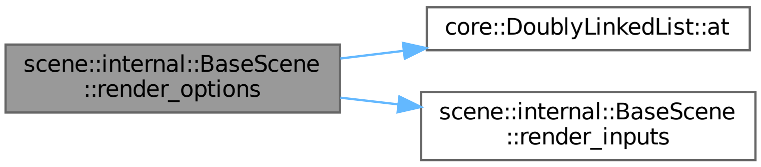

scene::ArrayScene:: render
scene::DynamicArrayScene ::render
scene::QueueScene:: render
scene::StackScene:: render
scene::internal::BaseScene ::render_options
scene::internal::BaseScene ::render_inputs

6.5.3.7 render_options()

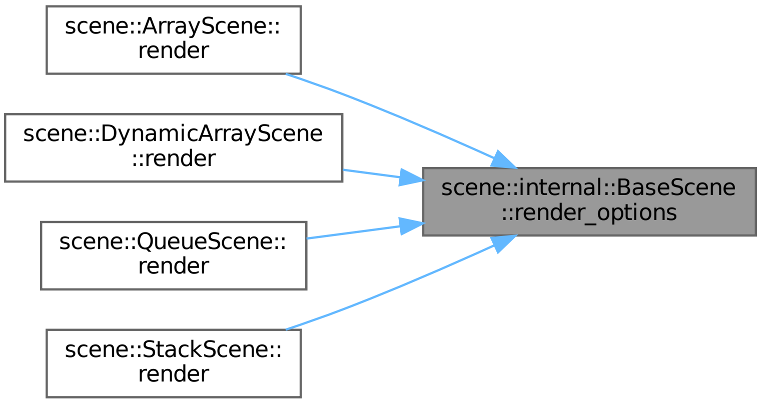
```
void scene::internal::BaseScene::render_options (
             SceneOptions & scene_config )  [protected], [virtual]
```

Definition at line 16 of file base_scene.cpp.

Here is the call graph for this function:


scene::internal::BaseScene ::render_options
core::DoublyLinkedList::at
scene::internal::BaseScene ::render_inputs

Here is the caller graph for this function:


scene::ArrayScene:: render
scene::DynamicArrayScene ::render
scene::QueueScene:: render
scene::StackScene:: render
scene::internal::BaseScene ::render_options

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.

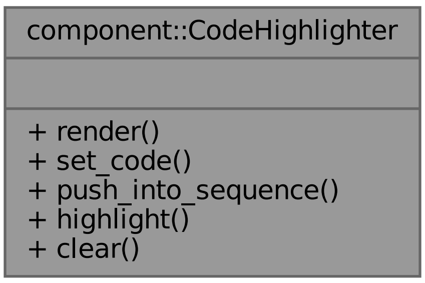
The documentation for this class was generated from the following files:

- 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.

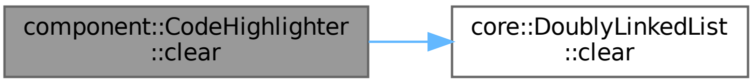
6.6.2 Member Function Documentation

6.6.2.1 clear()

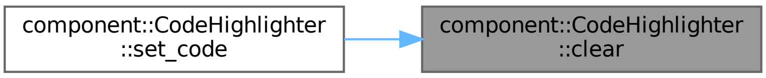
```
void component::CodeHighlighter::clear ( )
```

Definition at line 38 of file code_highlighter.cpp.

Here is the call graph for this function:


component::CodeHighlighter ::clear
core::DoublyLinkedList ::clear

Here is the caller graph for this function:

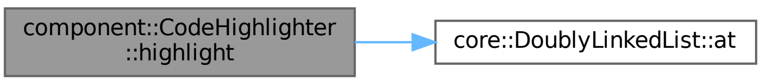

component::CodeHighlighter ::set_code
component::CodeHighlighter ::clear

6.6.2.2 highlight()

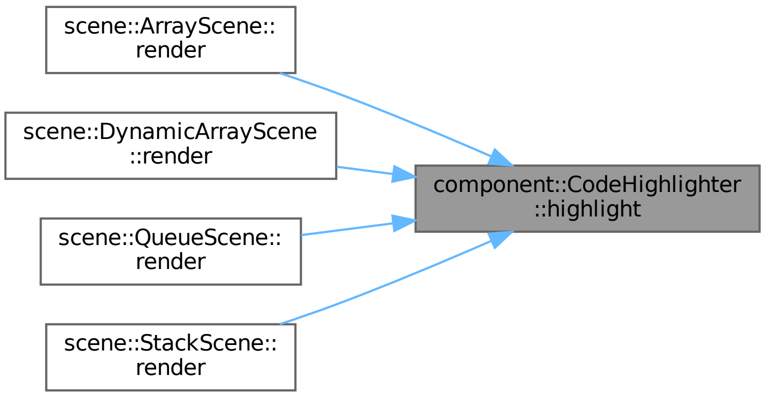
```
void component::CodeHighlighter::highlight (
             int frame_idx )
```

Definition at line 34 of file code_highlighter.cpp.

Here is the call graph for this function:


component::CodeHighlighter ::highlight
core::DoublyLinkedList::at

Here is the caller graph for this function:

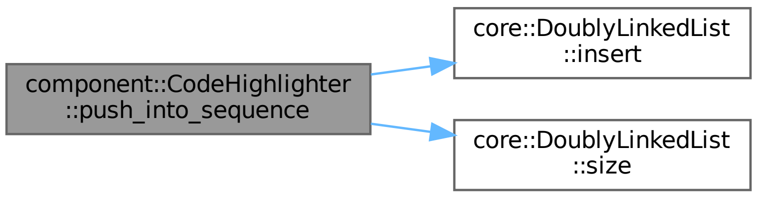

scene::ArrayScene:: render
scene::DynamicArrayScene ::render
scene::QueueScene:: render
scene::StackScene:: render
component::CodeHighlighter ::highlight

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:

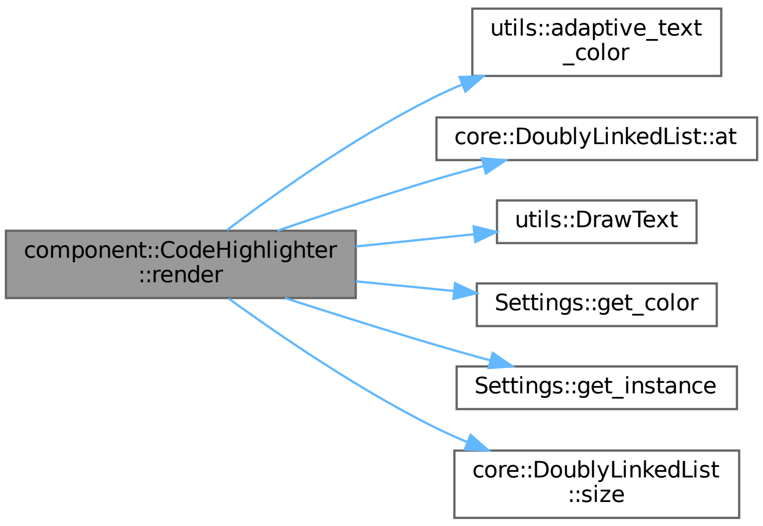

component::CodeHighlighter ::push_into_sequence
core::DoublyLinkedList ::insert
core::DoublyLinkedList ::size

6.6.2.4 render()

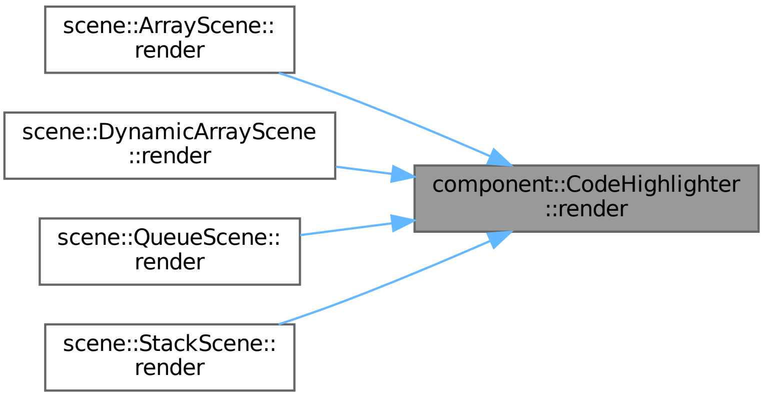
```
void component::CodeHighlighter::render ( )
```

Definition at line 9 of file code_highlighter.cpp.

Here is the call graph for this function:


component::CodeHighlighter ::render
utils::adaptive_text _color
core::DoublyLinkedList::at
utils::DrawText
Settings::get_color
Settings::get_instance
core::DoublyLinkedList ::size

Here is the caller graph for this function:

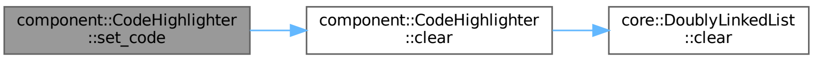

scene::ArrayScene::render
scene::DynamicArrayScene::render
scene::QueueScene::render
scene::StackScene::render
component::CodeHighlighter::render

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:


component::CodeHighlighter::set_code
component::CodeHighlighter::clear
core::DoublyLinkedList::clear

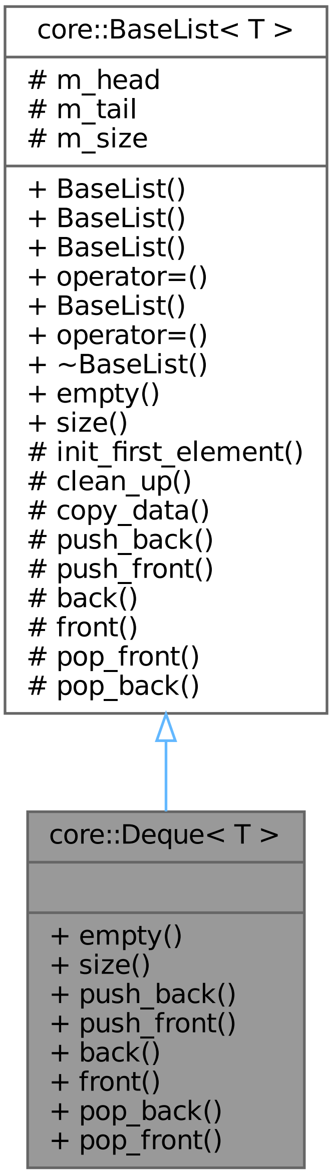
The documentation for this class was generated from the following files:

- src/component/code_highlighter.hpp
- src/component/code_highlighter.cpp

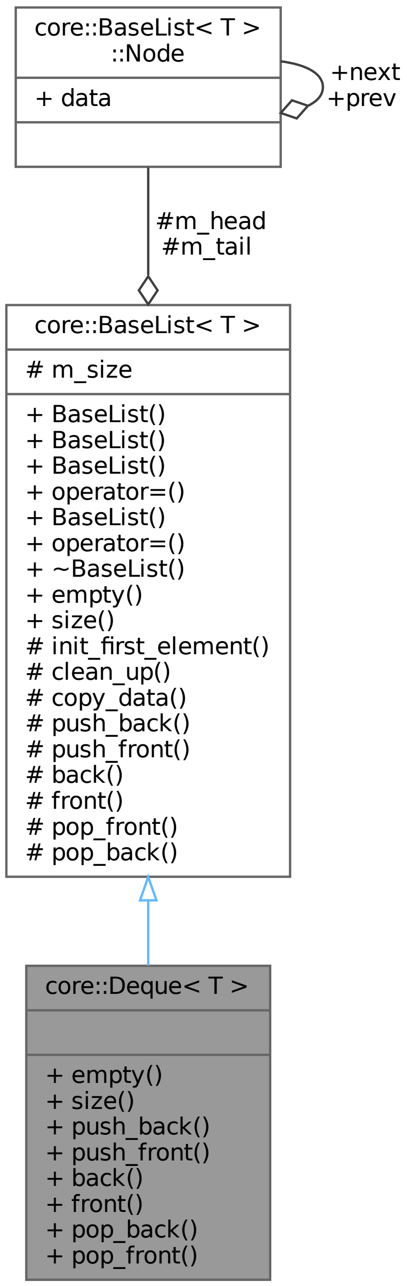
6.7 core::Deque< T > Class Template Reference

```
#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.

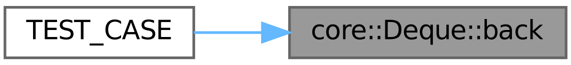
6.7.2 Member Function Documentation

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:

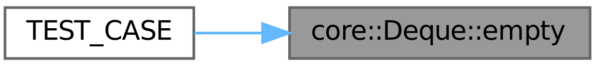


6.7.2.2 empty()

```
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:

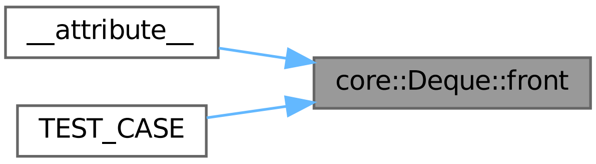


6.7.2.3 front()

```
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:

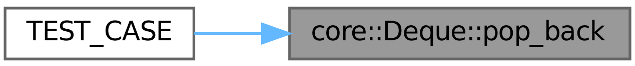


6.7.2.4 pop_back()

```
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:

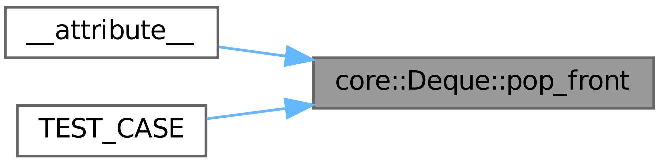


6.7.2.5 pop_front()

```
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:

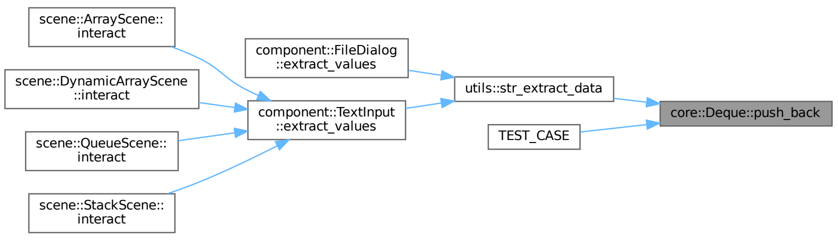


6.7.2.6 push_back()

```
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:

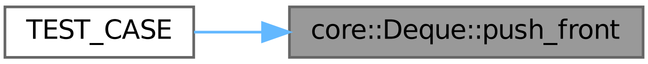

scene::ArrayScene::interact
scene::DynamicArrayScene::interact
scene::QueueScene::interact
scene::StackScene::interact
component::FileDialog::extract_values
component::TextInput::extract_values
utils::str_extract_data
TEST_CASE
core::Deque::push_back

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:

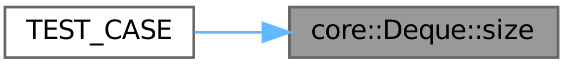


6.7.2.8 size()

```
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:



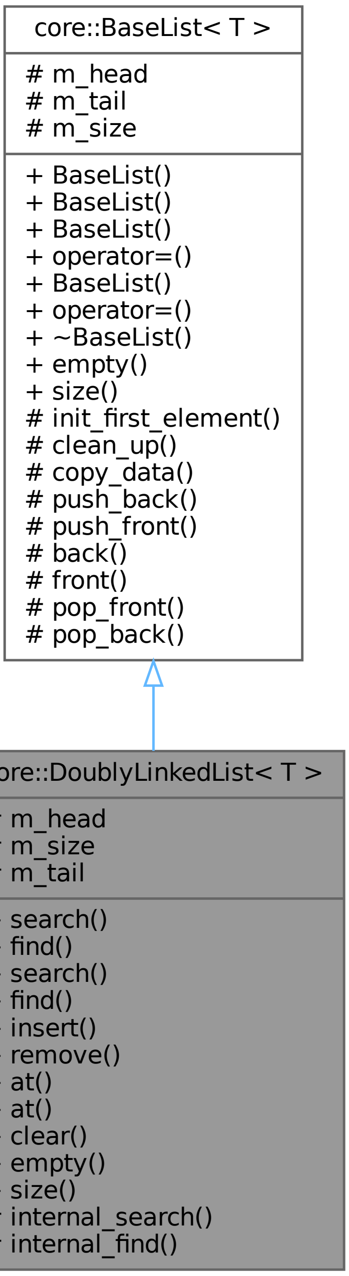
The documentation for this class was generated from the following file:

- src/core/deque.hpp

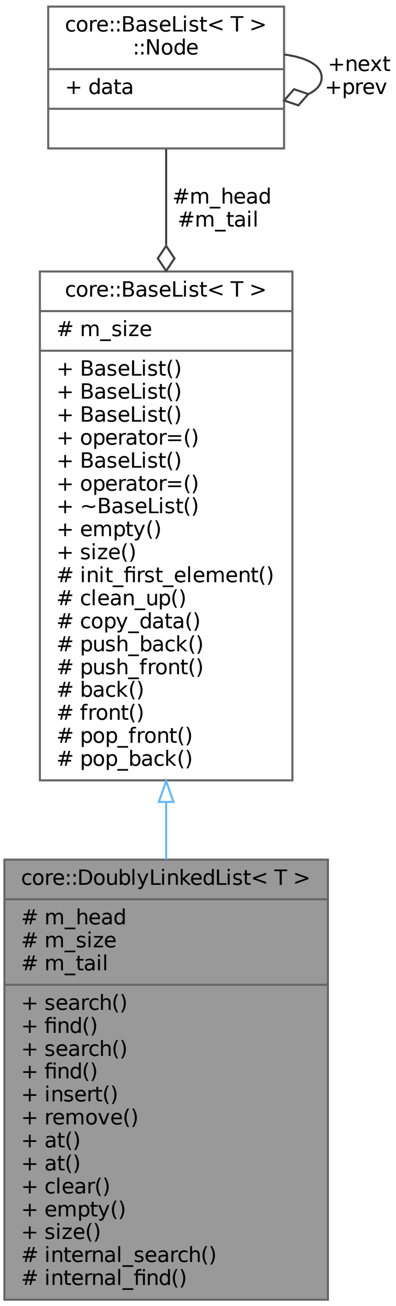
6.8 core::DoublyLinkedList< T > Class Template Reference

```
#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.

6.8.2.4 Node_ptr

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

Definition at line 15 of file doubly_linked_list.hpp.

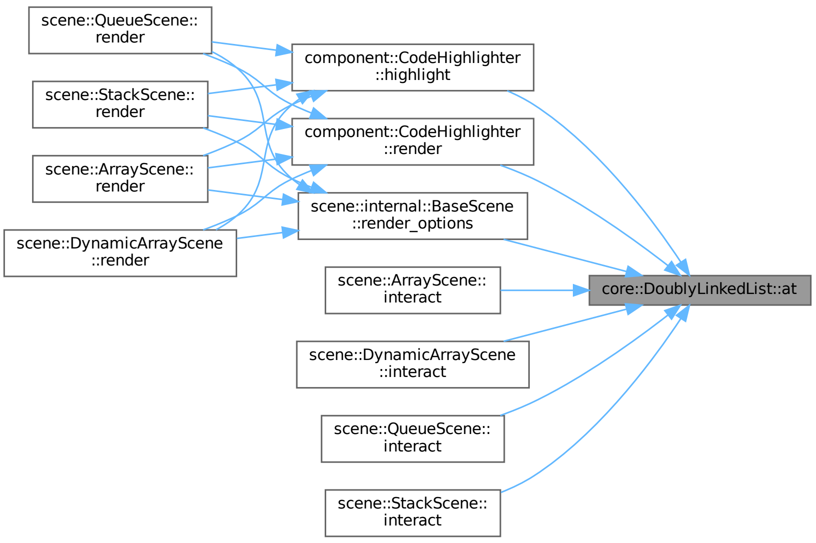
6.8.3 Member Function Documentation

6.8.3.1 at() [1/2]

```
template<typename T >
T & core::DoublyLinkedList< T >::at (
             std::size_t index )
```

Definition at line 153 of file doubly_linked_list.hpp.

Here is the caller graph for this function:


scene::QueueScene::render
scene::StackScene::render
scene::ArrayScene::render
scene::DynamicArrayScene::render
component::CodeHighlighter::highlight
component::CodeHighlighter::render
scene::internal::BaseScene::render_options
scene::ArrayScene::interact
scene::DynamicArrayScene::interact
scene::QueueScene::interact
scene::StackScene::interact
core::DoublyLinkedList::at

6.8.3.2 at() [2/2]

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

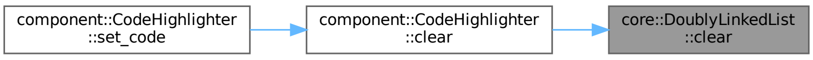
Definition at line 158 of file doubly_linked_list.hpp.

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:


component::CodeHighlighter ::set_code
component::CodeHighlighter ::clear
core::DoublyLinkedList ::clear

6.8.3.4 empty()

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

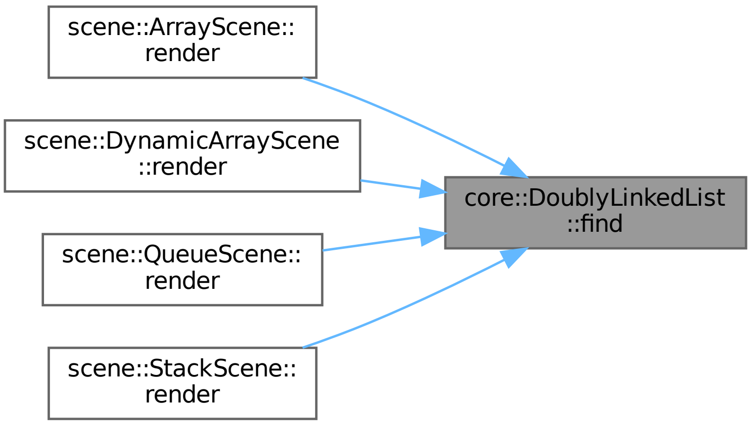
Definition at line 48 of file base_list.hpp.

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:


scene::ArrayScene:: render
scene::DynamicArrayScene ::render
scene::QueueScene:: render
scene::StackScene:: render
core::DoublyLinkedList ::find

6.8.3.6 find() [2/2]

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

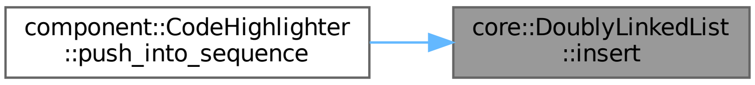
Definition at line 95 of file doubly_linked_list.hpp.

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:


component::CodeHighlighter ::push_into_sequence
core::DoublyLinkedList ::insert

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 )
```

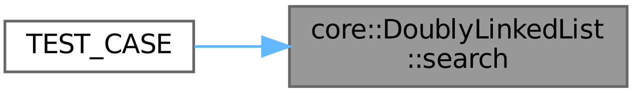
Definition at line 124 of file doubly_linked_list.hpp.

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
```

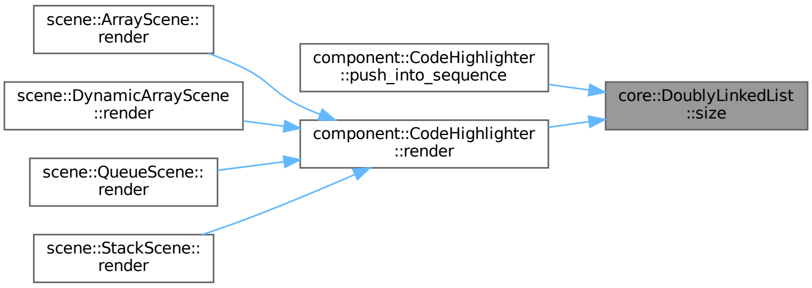
Definition at line 89 of file doubly_linked_list.hpp.

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.

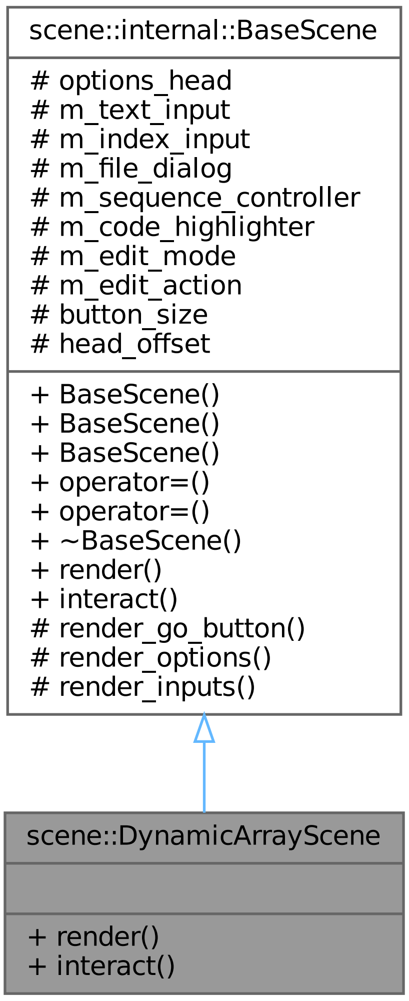
The documentation for this class was generated from the following file:

- src/core/doubly_linked_list.hpp

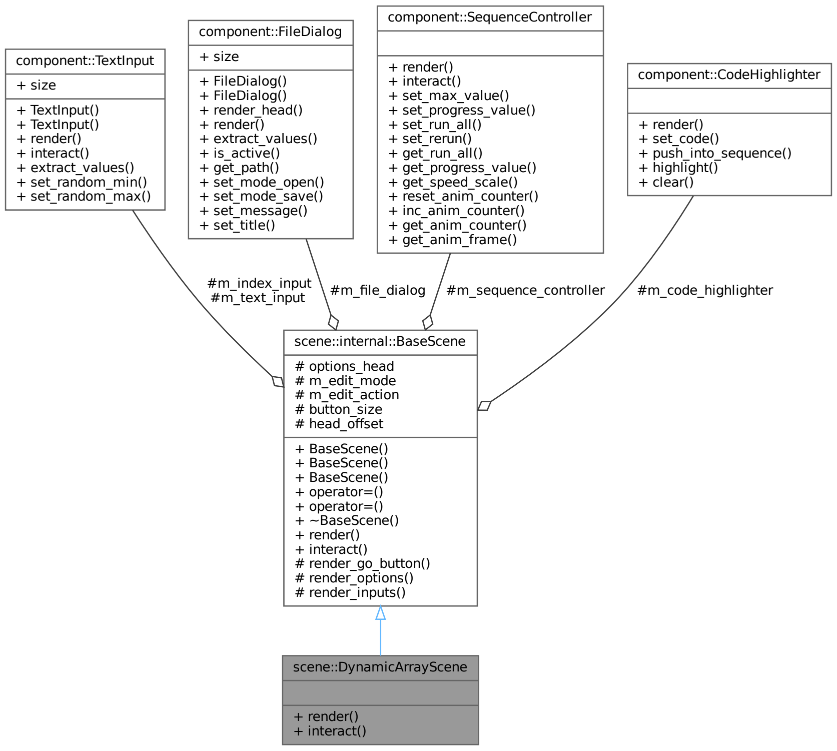
6.9 scene::DynamicArrayScene Class Reference

```
#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

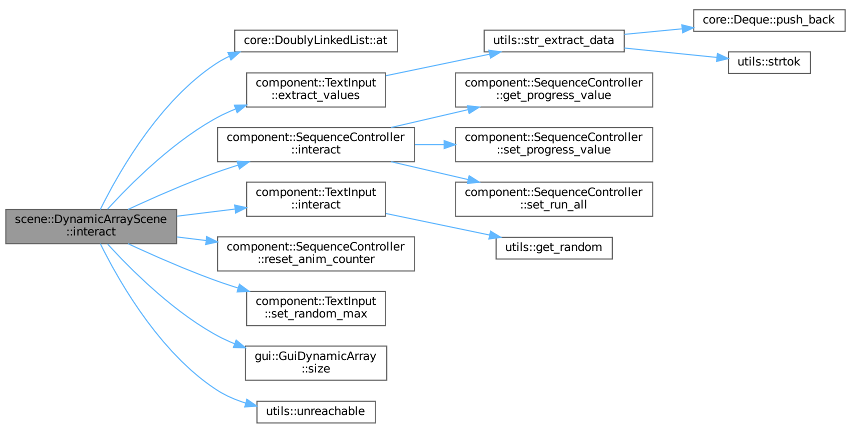
6.9.2.1 interact()

```
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:


scene::DynamicArrayScene ::interact
core::DoublyLinkedList::at
component::TextInput ::extract_values
component::SequenceController ::interact
component::TextInput ::interact
component::SequenceController ::reset_anim_counter
component::TextInput ::set_random_max
gui::GuiDynamicArray ::size
utils::unreachable
utils::str_extract_data
component::SequenceController ::get_progress_value
component::SequenceController ::set_progress_value
component::SequenceController ::set_run_all
utils::get_random
core::Deque::push_back
utils::strtok

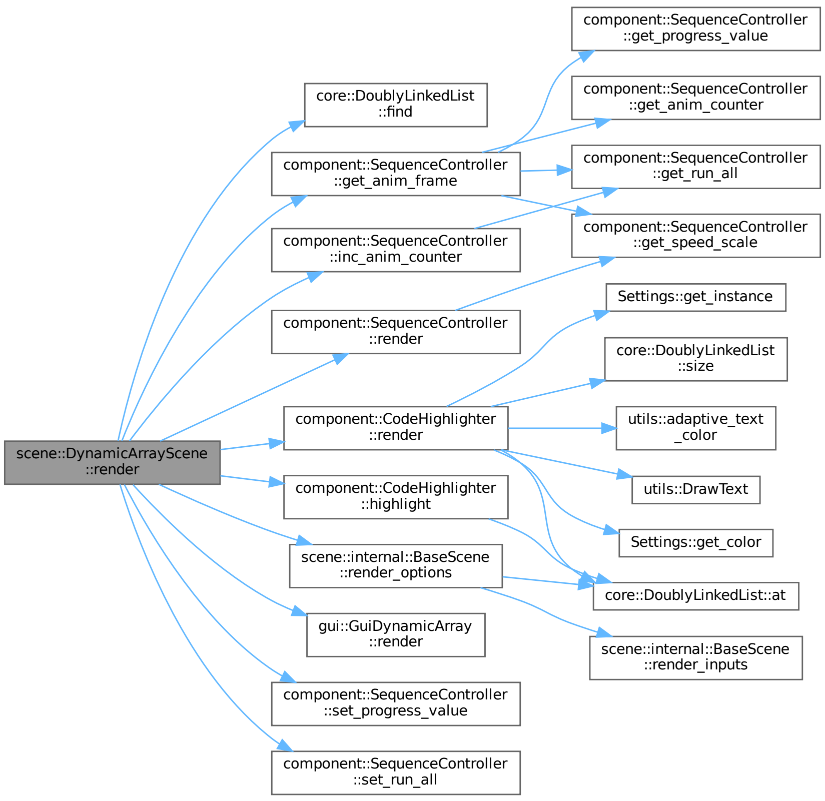
6.9.2.2 render()

```
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:


scene::DynamicArrayScene ::render
core::DoublyLinkedList ::find
component::SequenceController ::get_anim_frame
component::SequenceController ::inc_anim_counter
component::SequenceController ::render
component::CodeHighlighter ::render
component::CodeHighlighter ::highlight
scene::internal::BaseScene ::render_options
gui::GuiDynamicArray ::render
component::SequenceController ::set_progress_value
component::SequenceController ::set_run_all
component::SequenceController ::get_progress_value
component::SequenceController ::get_anim_counter
component::SequenceController ::get_run_all
component::SequenceController ::get_speed_scale
Settings::get_instance
core::DoublyLinkedList ::size
utils::adaptive_text _color
utils::DrawText
Settings::get_color
core::DoublyLinkedList::at
scene::internal::BaseScene ::render_inputs

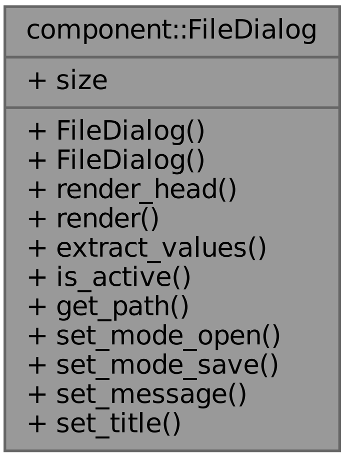
The documentation for this class was generated from the following files:

- 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:



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.

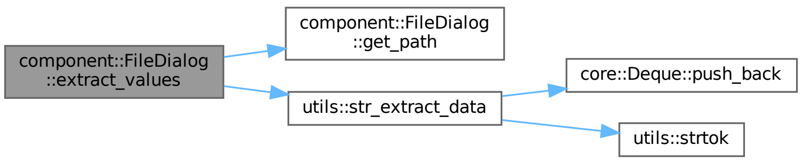
6.10.3 Member Function Documentation

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:

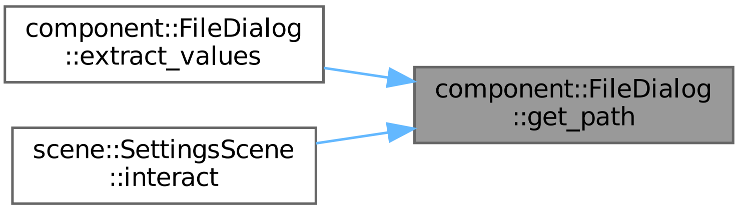

component::FileDialog ::extract_values
component::FileDialog ::get_path
utils::str_extract_data
core::Deque::push_back
utils::strtok

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:


component::FileDialog ::extract_values
scene::SettingsScene ::interact
component::FileDialog ::get_path

6.10.3.3 is_active()

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

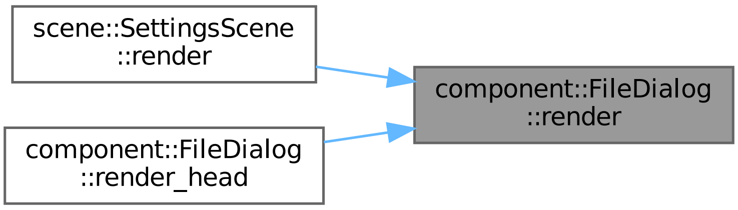
Definition at line 57 of file file_dialog.cpp.

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:

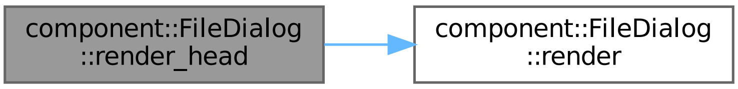

scene::SettingsScene ::render
component::FileDialog ::render_head
component::FileDialog ::render

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:


component::FileDialog ::render_head
component::FileDialog ::render

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.

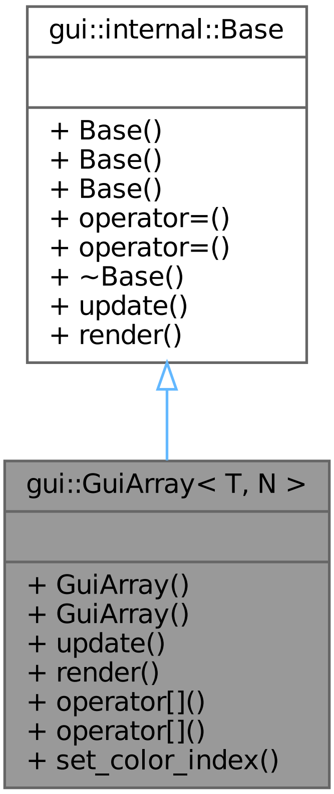
The documentation for this class was generated from the following files:

- src/component/file_dialog.hpp
- src/component/file_dialog.cpp

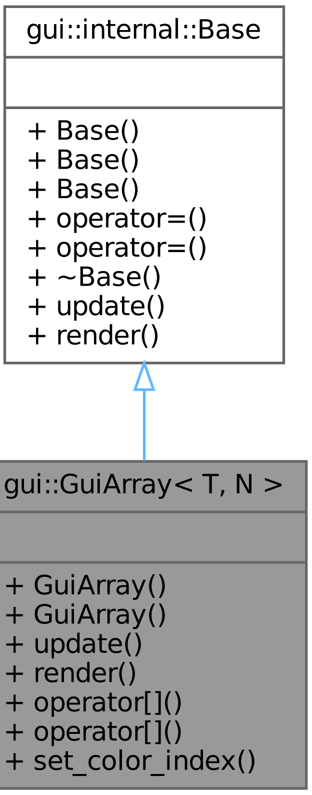
6.11 gui::GuiArray< T, N > Class Template Reference

```
#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.

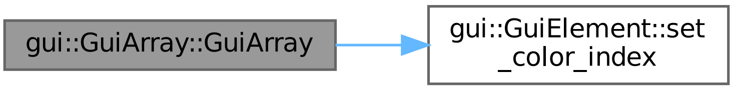
6.11.2 Constructor & Destructor Documentation

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.

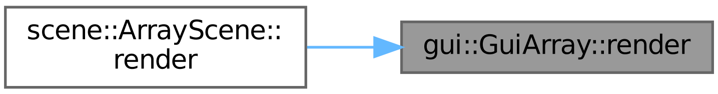
6.11.3.3 render()

```
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.

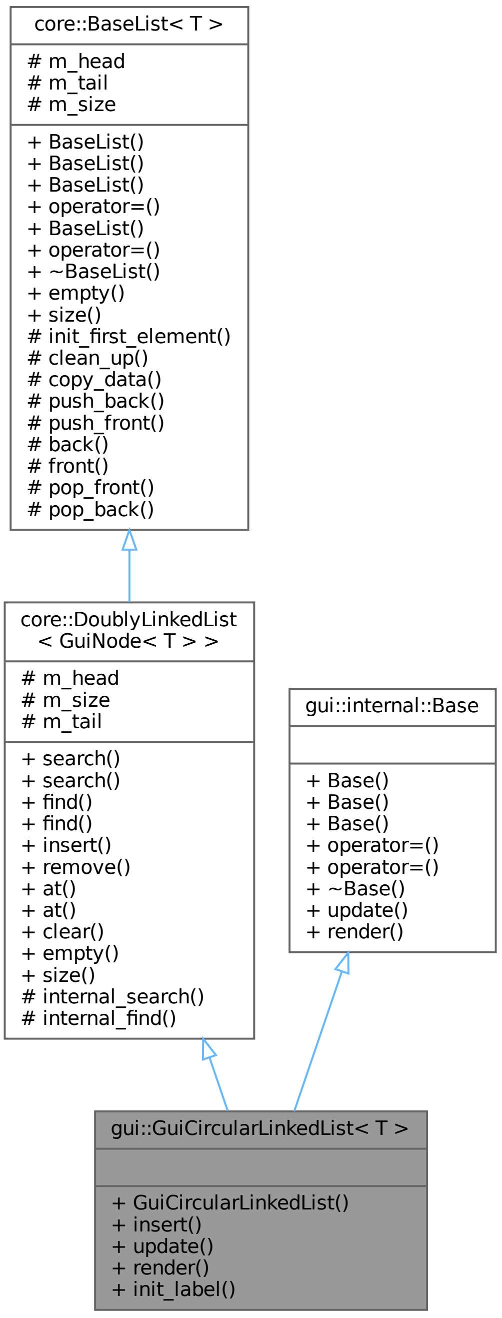
The documentation for this class was generated from the following file:

- src/gui/array_gui.hpp

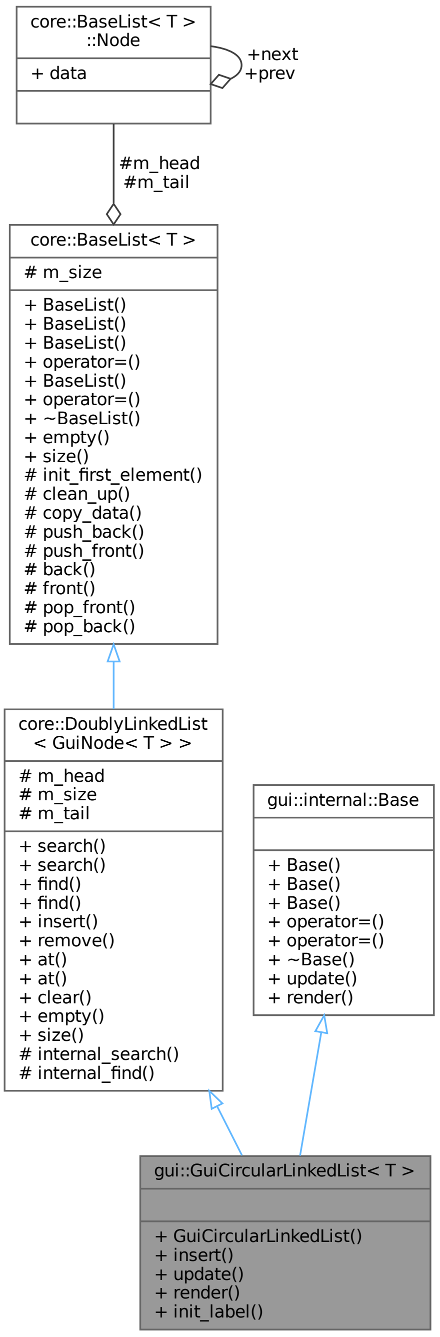
6.12 gui::GuiCircularLinkedList< T > Class Template Reference

```
#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.

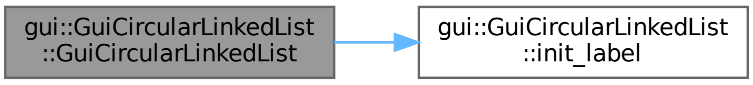
6.12.2 Constructor & Destructor Documentation

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:


gui::GuiCircularLinkedList ::GuiCircularLinkedList
gui::GuiCircularLinkedList ::init_label

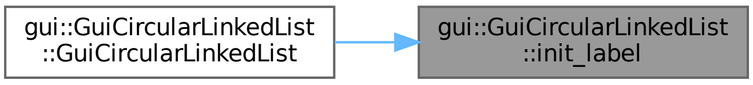
6.12.3 Member Function Documentation

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:


gui::GuiCircularLinkedList ::GuiCircularLinkedList
gui::GuiCircularLinkedList ::init_label

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.

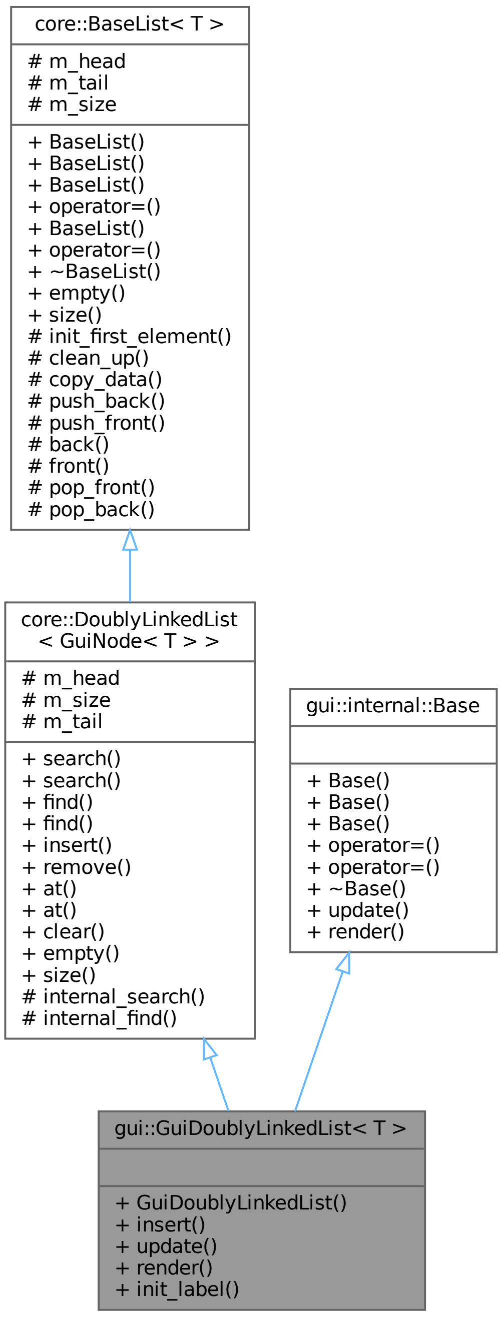
The documentation for this class was generated from the following file:

- src/gui/circular_linked_list_gui.hpp

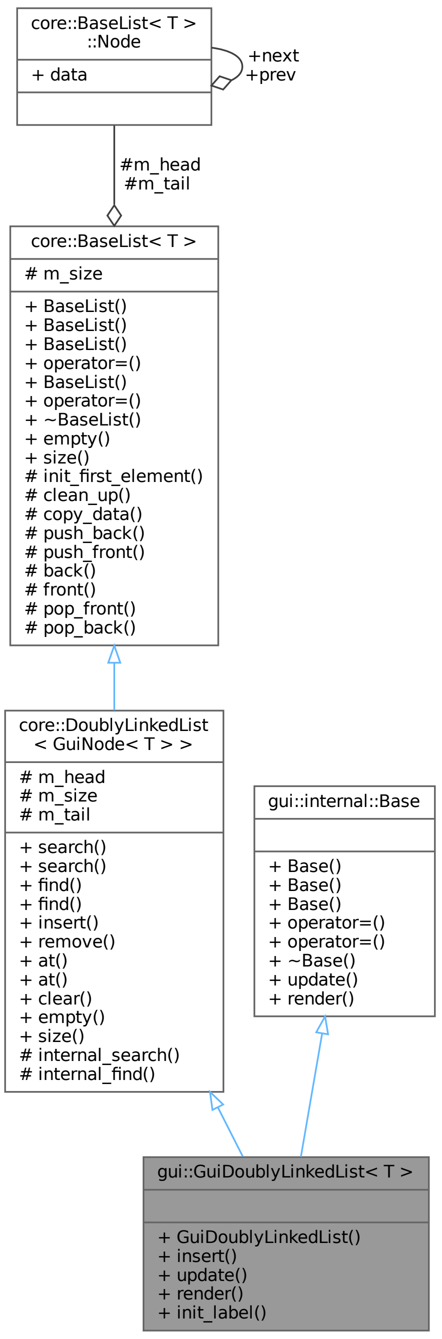
6.13 gui::GuiDoublyLinkedList< T > Class Template Reference

```
#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.

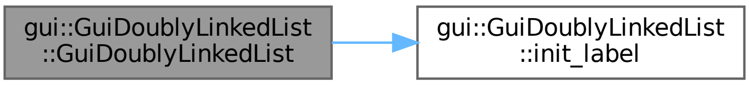
6.13.2 Constructor & Destructor Documentation

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:


gui::GuiDoublyLinkedList ::GuiDoublyLinkedList
gui::GuiDoublyLinkedList ::init_label

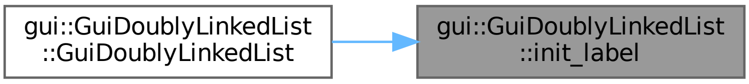
6.13.3 Member Function Documentation

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:


gui::GuiDoublyLinkedList ::GuiDoublyLinkedList
gui::GuiDoublyLinkedList ::init_label

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.

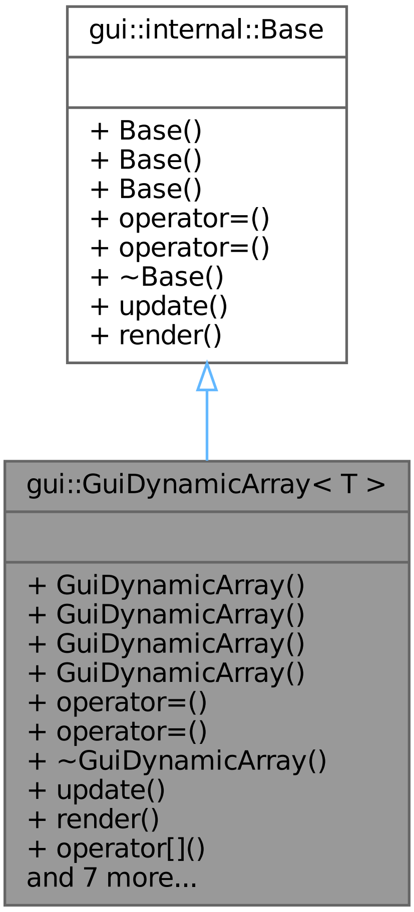
The documentation for this class was generated from the following file:

- src/gui/doubly_linked_list_gui.hpp

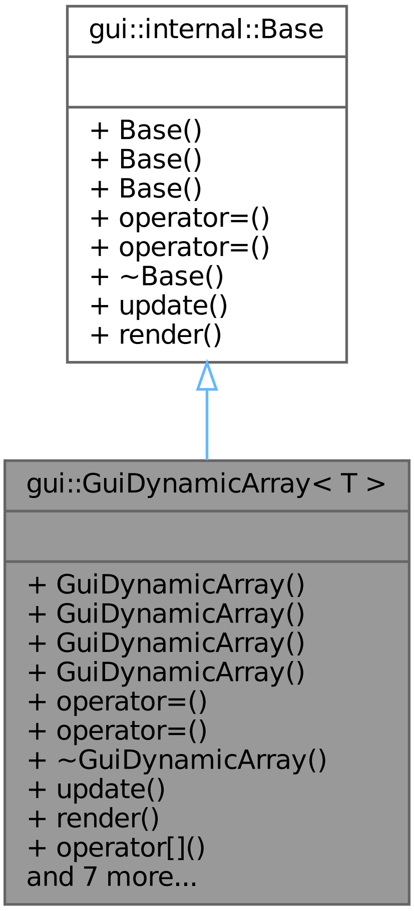
6.14 gui::GuiDynamicArray< T > Class Template Reference

```
#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
```

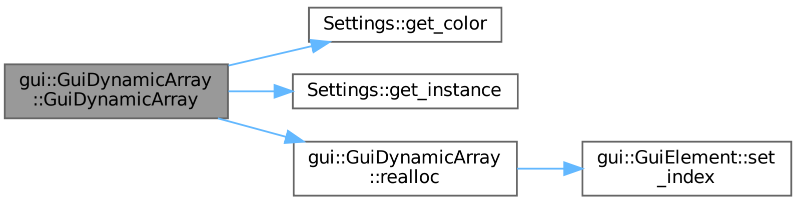
Definition at line 77 of file dynamic_array_gui.hpp.

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:


gui::GuiDynamicArray ::GuiDynamicArray
Settings::get_color
Settings::get_instance
gui::GuiDynamicArray ::realloc
gui::GuiElement::set _index

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 )
```

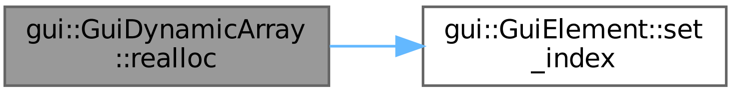
Definition at line 197 of file dynamic_array_gui.hpp.

6.14.3.8 realloc()

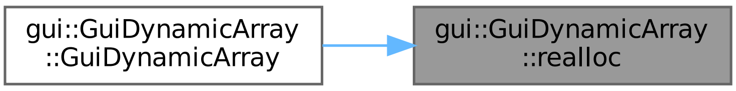
```
template<typename T >
void gui::GuiDynamicArray< T >::realloc (
             std::size_t capacity )
```

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:


gui::GuiDynamicArray ::GuiDynamicArray
gui::GuiDynamicArray ::realloc

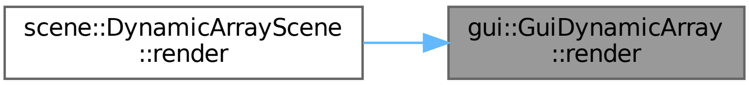
6.14.3.9 render()

```
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:


scene::DynamicArrayScene ::render
gui::GuiDynamicArray ::render

6.14.3.10 set_color_index()

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

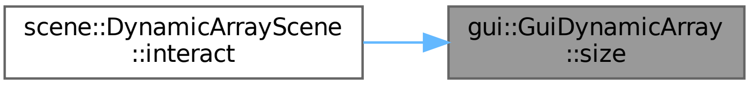
Definition at line 182 of file dynamic_array_gui.hpp.

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:


scene::DynamicArrayScene ::interact
gui::GuiDynamicArray ::size

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.

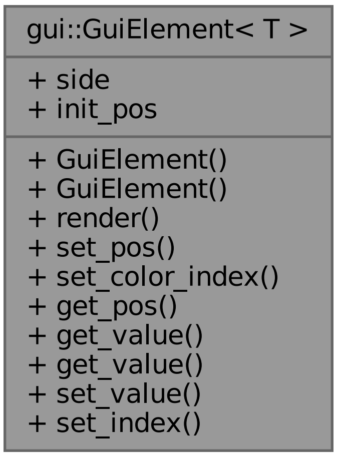
The documentation for this class was generated from the following file:

- src/gui/dynamic_array_gui.hpp

6.15 gui::GuiElement< T > Class Template Reference

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

Collaboration diagram for gui::GuiElement< T >:



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
```

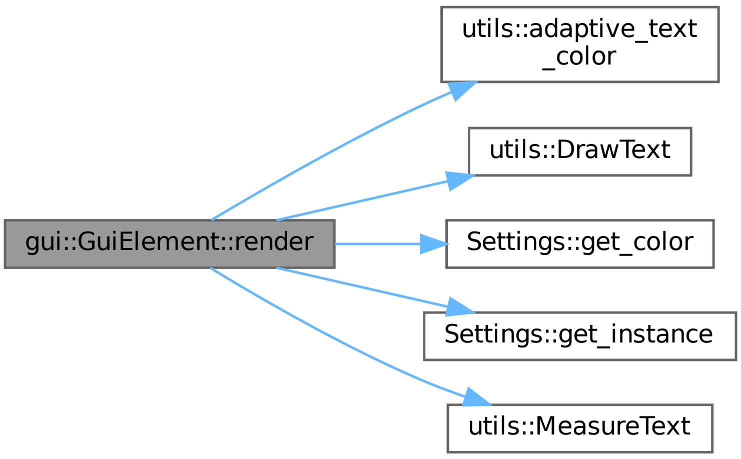
Definition at line 105 of file element_gui.hpp.

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:

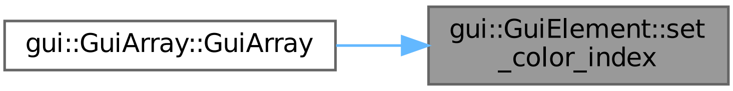


6.15.3.5 set_color_index()

```
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:

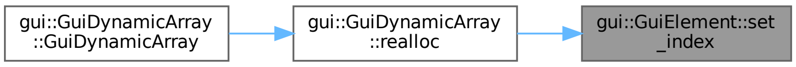


6.15.3.6 set_index()

```
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.

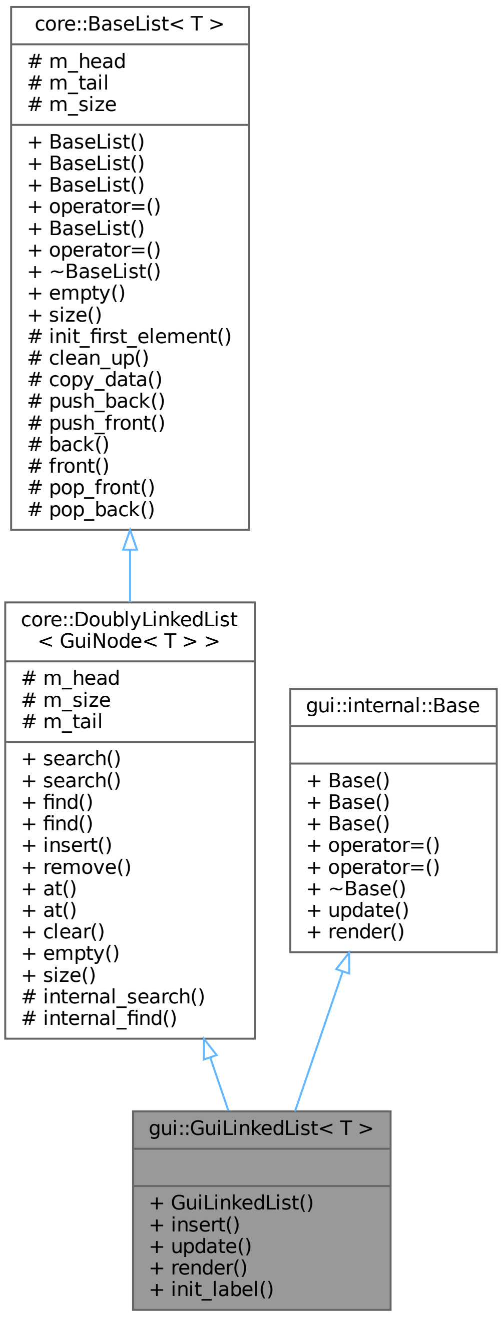
The documentation for this class was generated from the following file:

- src/gui/element_gui.hpp

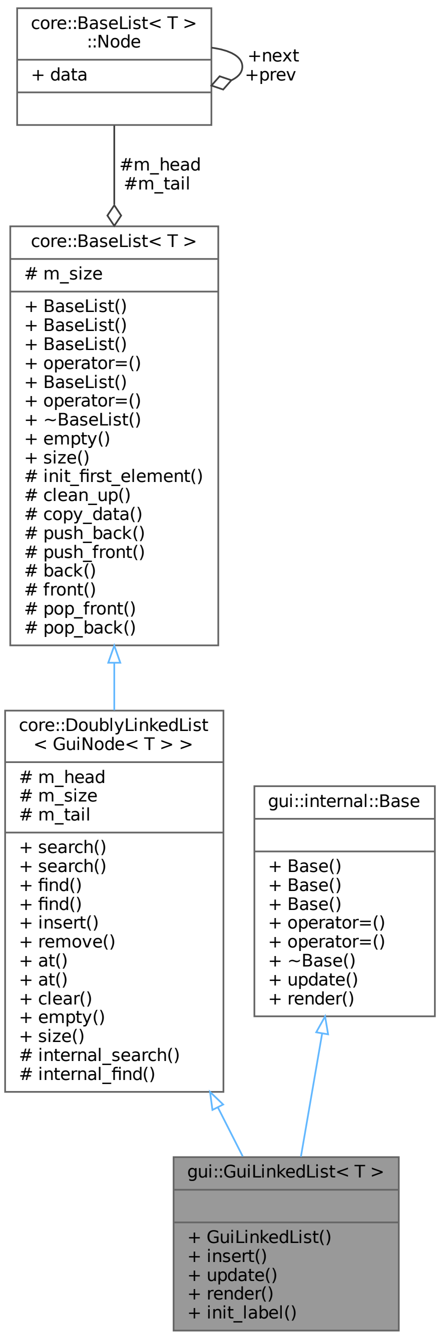
6.16 gui::GuiLinkedList< T > Class Template Reference

```
#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.

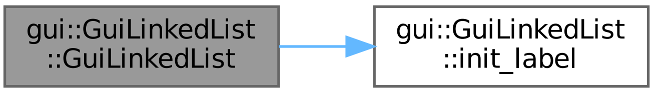
6.16.2 Constructor & Destructor Documentation

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:



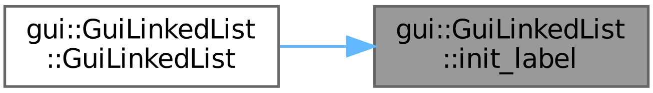
6.16.3 Member Function Documentation

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.

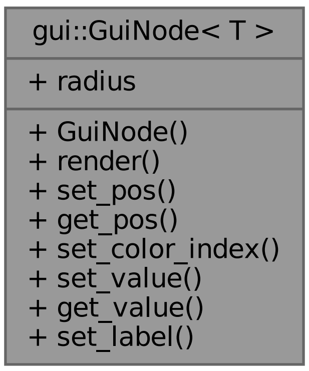
The documentation for this class was generated from the following file:

- src/gui/linked_list_gui.hpp

6.17 gui::GuiNode< T > Class Template Reference

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

Collaboration diagram for gui::GuiNode< T >:



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
```

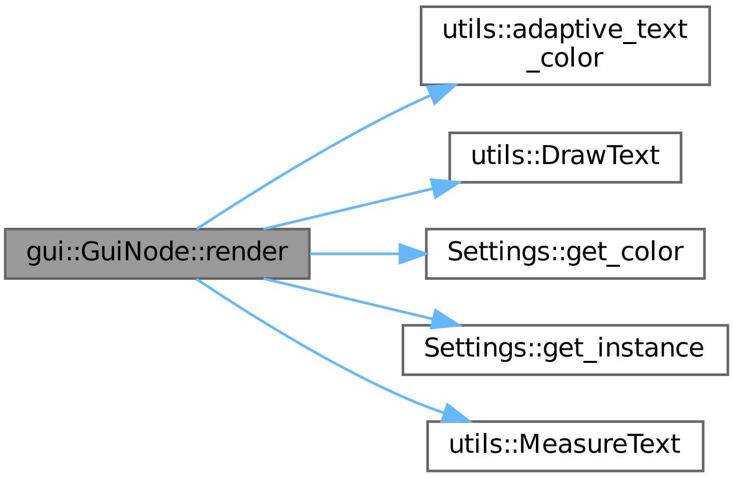
Definition at line 87 of file node_gui.hpp.

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.

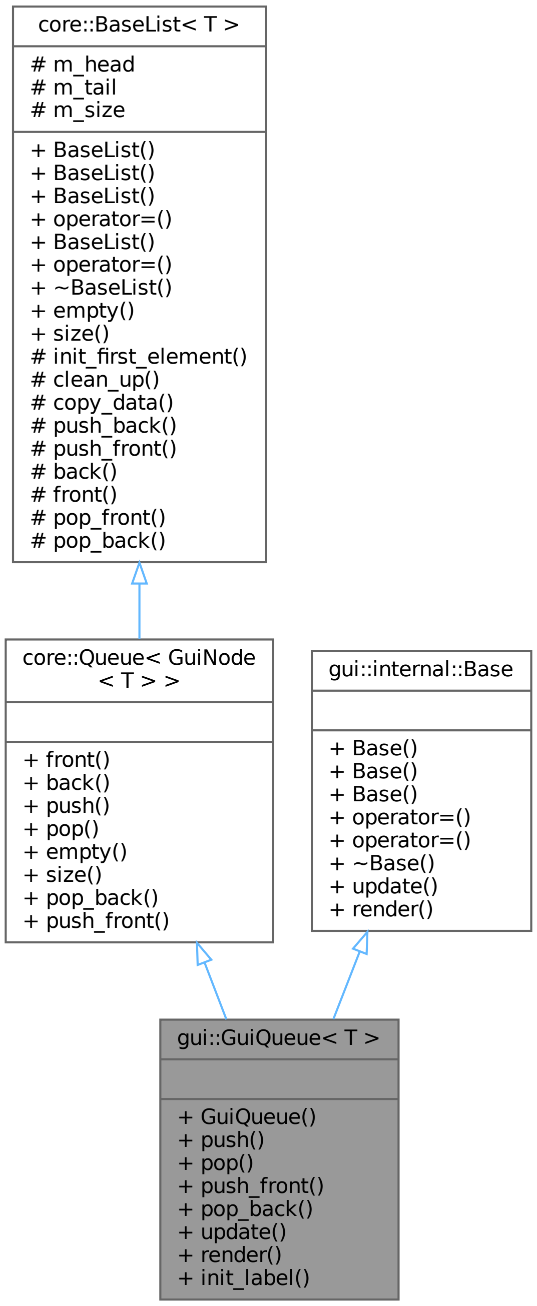
The documentation for this class was generated from the following file:

- src/gui/node_gui.hpp

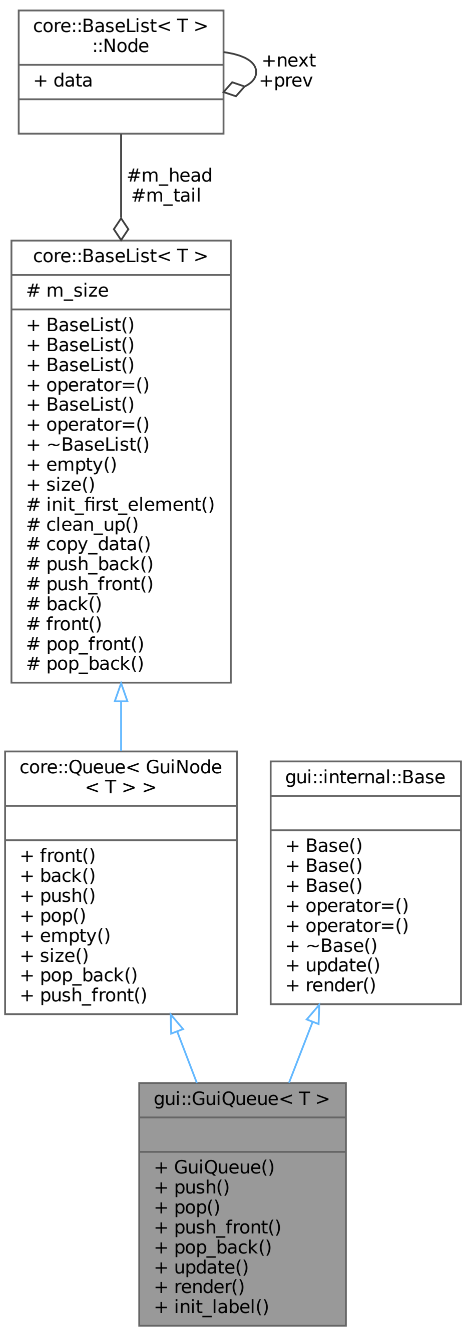
6.18 gui::GuiQueue< T > Class Template Reference

```
#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.

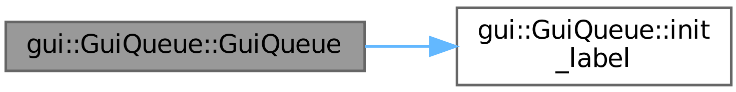
6.18.2 Constructor & Destructor Documentation

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:



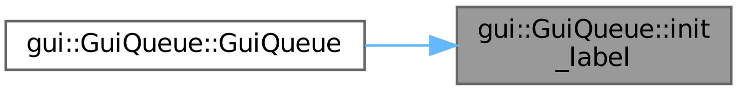
6.18.3 Member Function Documentation

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.

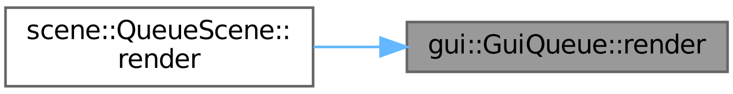
6.18.3.6 render()

```
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.

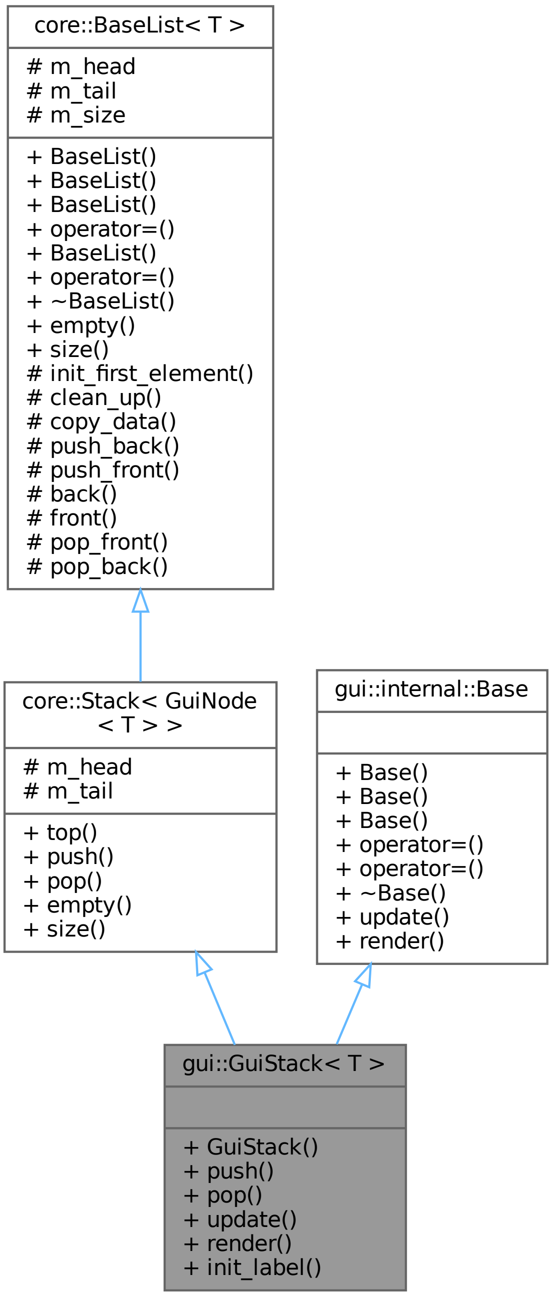
The documentation for this class was generated from the following file:

- src/gui/queue_gui.hpp

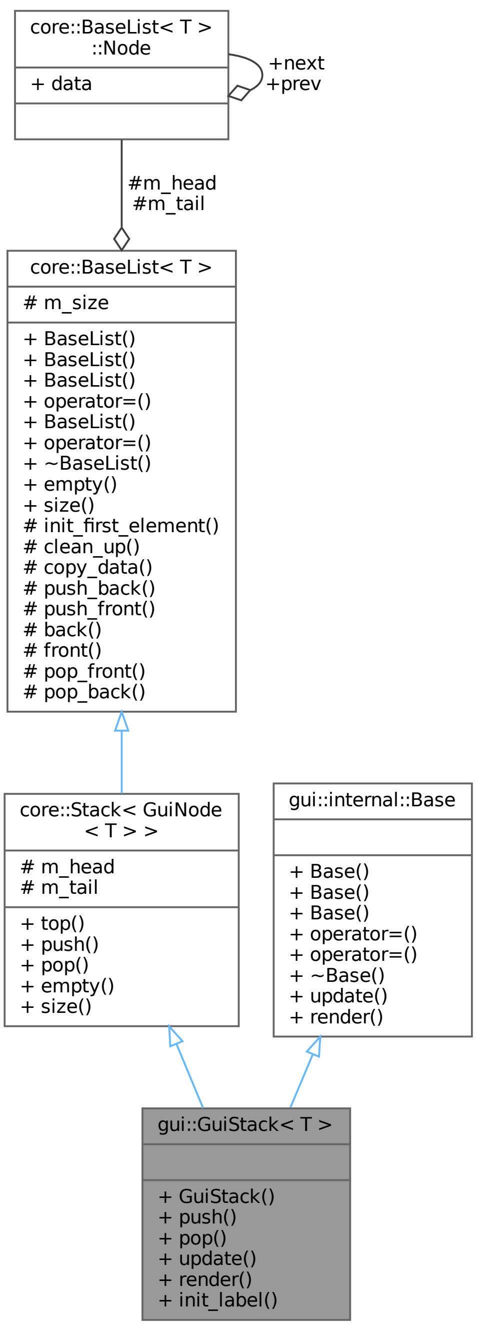
6.19 gui::GuiStack< T > Class Template Reference

```
#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.

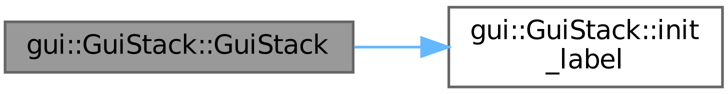
6.19.2 Constructor & Destructor Documentation

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:



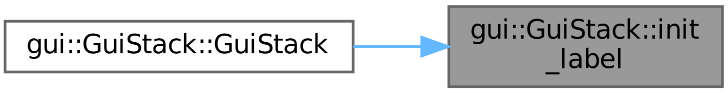
6.19.3 Member Function Documentation

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.

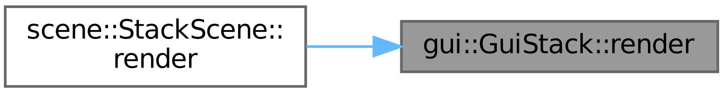
6.19.3.4 render()

```
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.

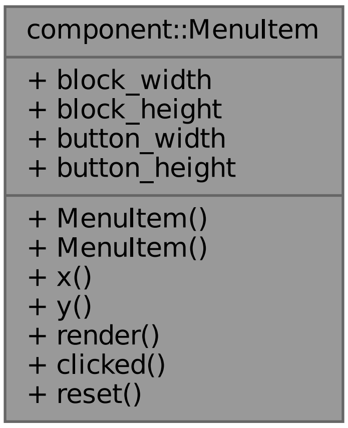
The documentation for this class was generated from the following file:

- src/gui/stack_gui.hpp

6.20 component::MenuItem Class Reference

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

Collaboration diagram for component::MenuItem:



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.

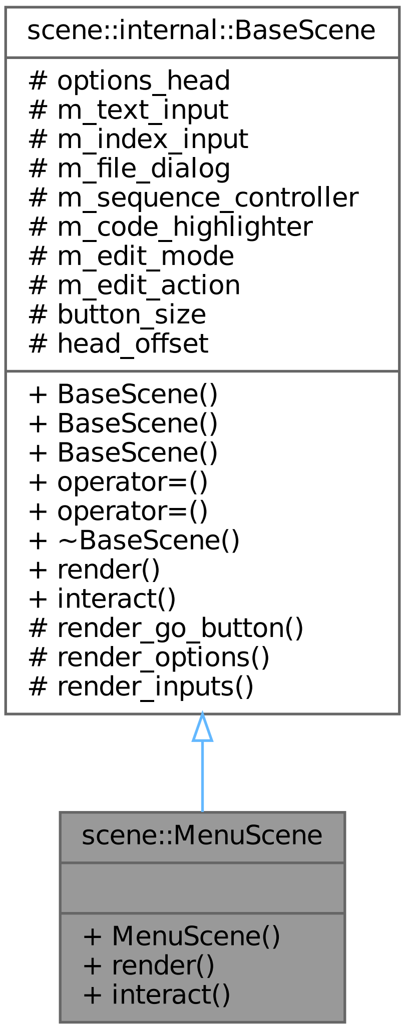
The documentation for this class was generated from the following files:

- src/component/menu_item.hpp
- src/component/menu_item.cpp

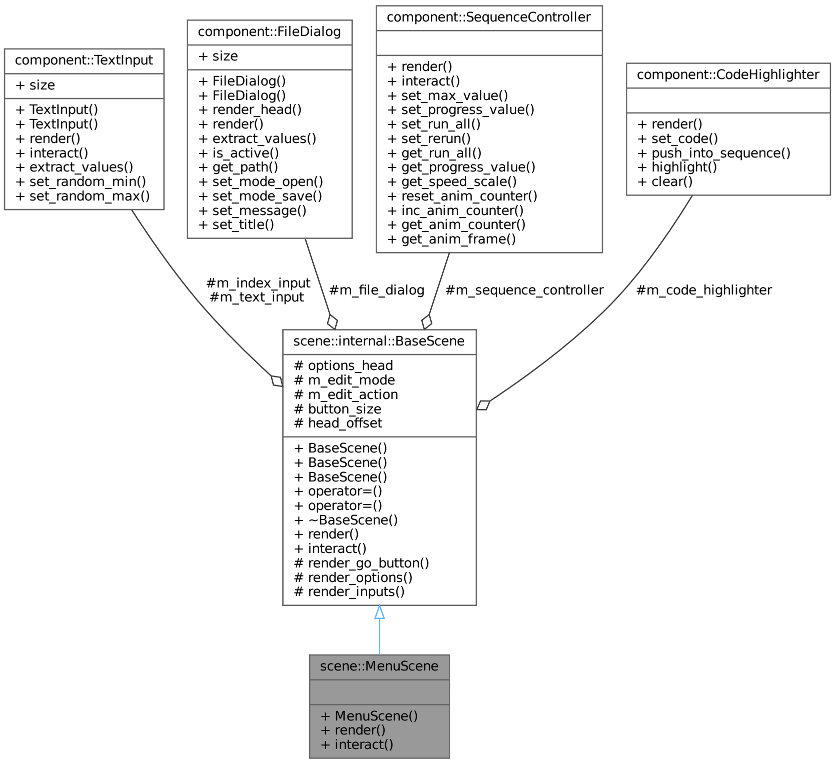
6.21 scene::MenuScene Class Reference

```
#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

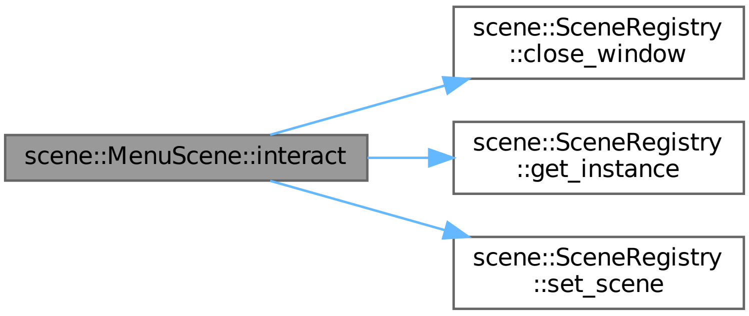
6.21.3.1 interact()

```
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:


scene::MenuScene::interact
scene::SceneRegistry ::close_window
scene::SceneRegistry ::get_instance
scene::SceneRegistry ::set_scene

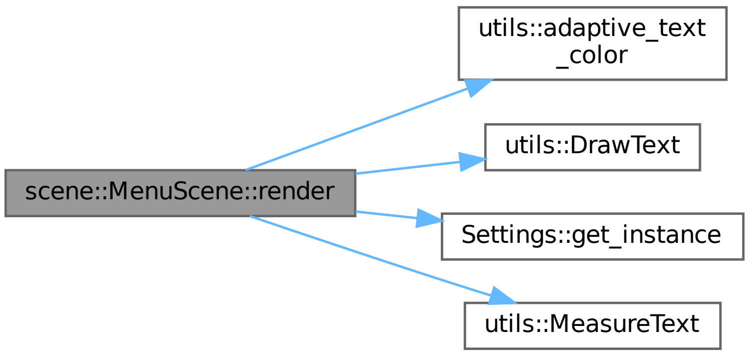
6.21.3.2 render()

```
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:



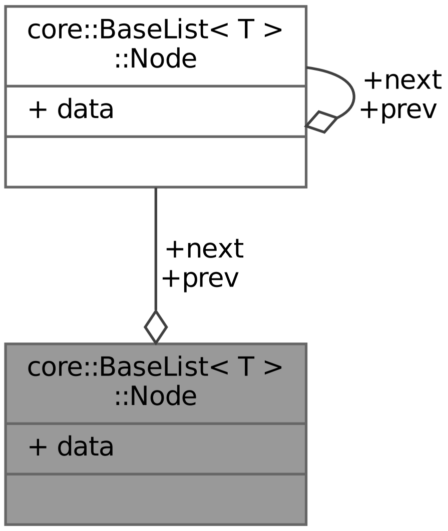
The documentation for this class was generated from the following files:

- 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.

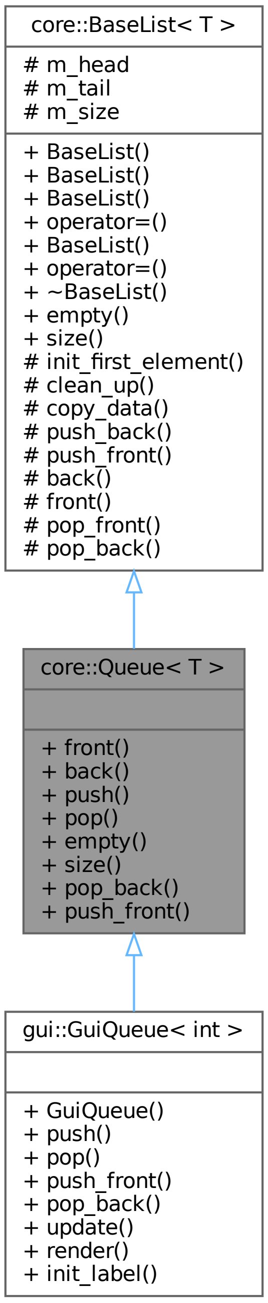
The documentation for this struct was generated from the following file:

- src/core/base_list.hpp

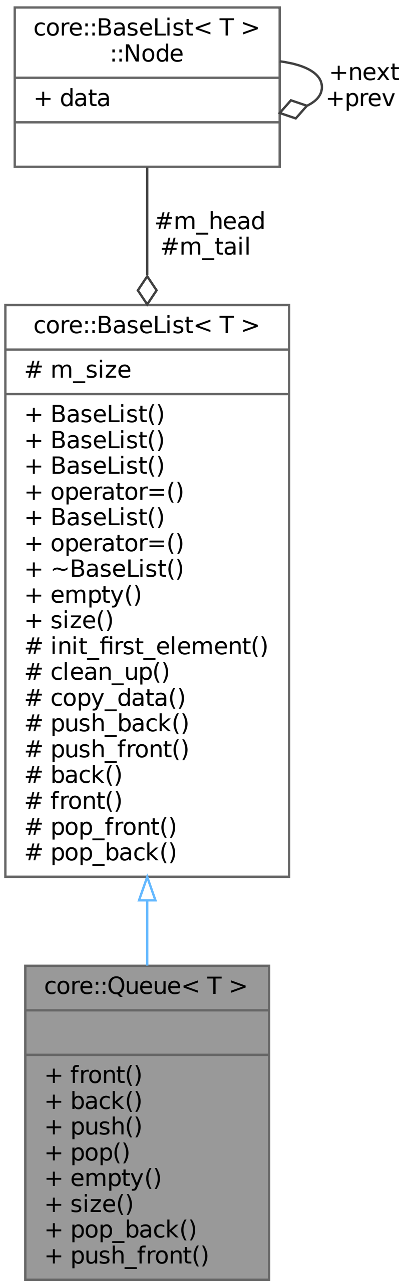
6.23 core::Queue< T > Class Template Reference

```
#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 )
```

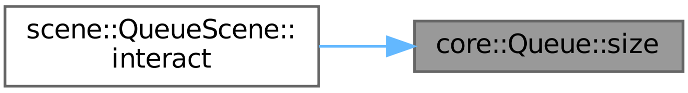
Definition at line 31 of file base_list.hpp.

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:



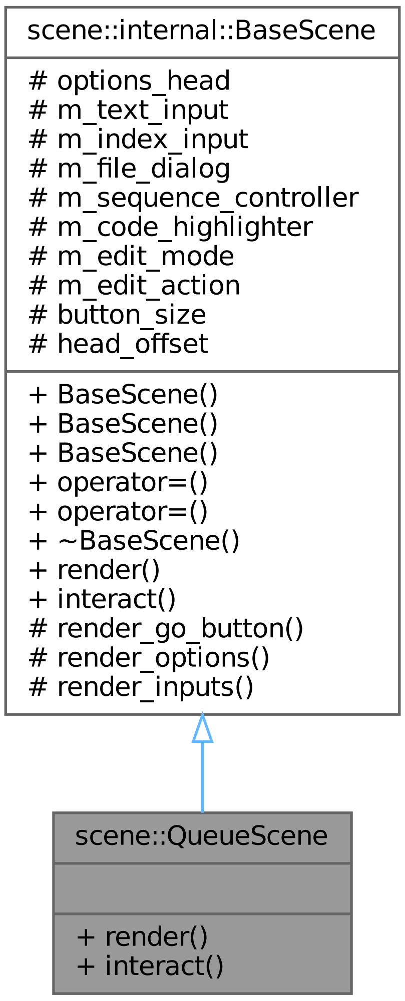
The documentation for this class was generated from the following file:

- src/core/queue.hpp

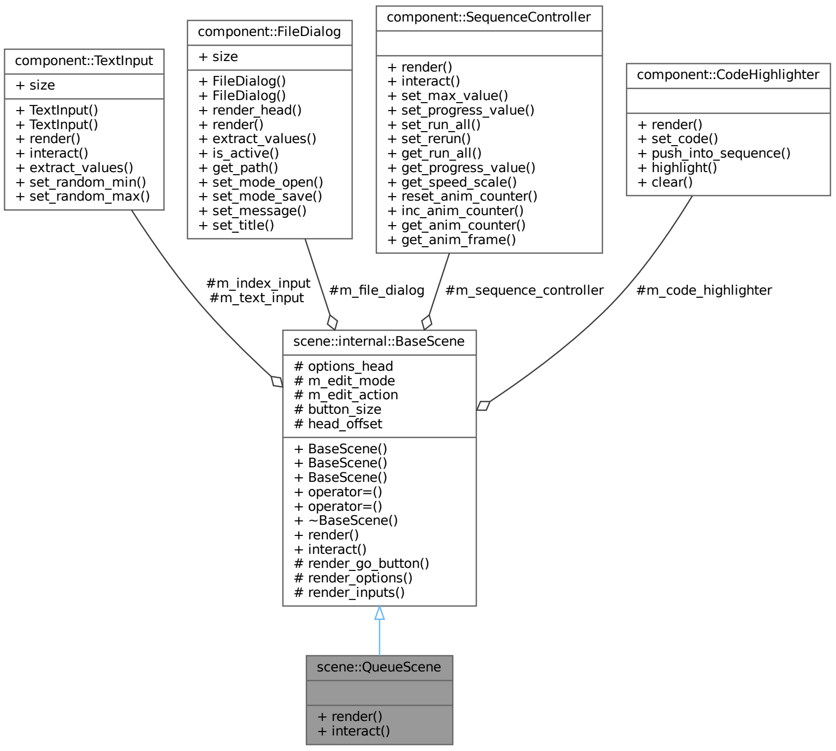
6.24 scene::QueueScene Class Reference

```
#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

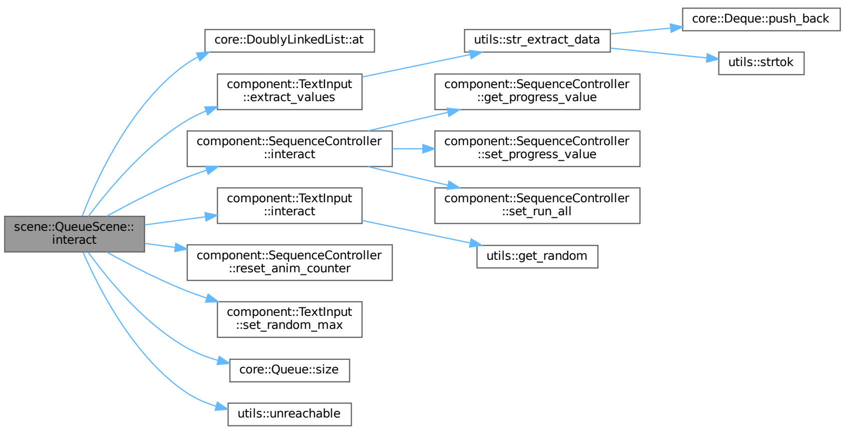
6.24.2.1 interact()

```
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:


scene::QueueScene::interact
core::DoublyLinkedList::at
component::TextInput::extract_values
component::SequenceController::interact
component::TextInput::interact
component::SequenceController::reset_anim_counter
component::TextInput::set_random_max
core::Queue::size
utils::unreachable
utils::str_extract_data
component::SequenceController::get_progress_value
component::SequenceController::set_progress_value
component::SequenceController::set_run_all
utils::get_random
core::Deque::push_back
utils::strtok

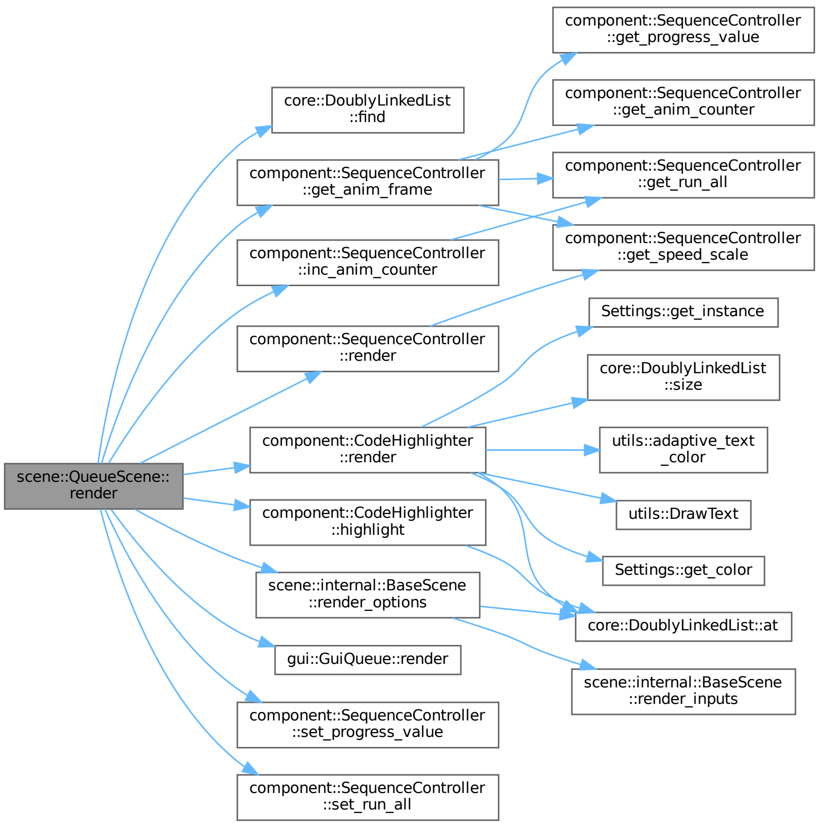
6.24.2.2 render()

```
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:


scene::QueueScene::render
core::DoublyLinkedList::find
component::SequenceController::get_anim_frame
component::SequenceController::inc_anim_counter
component::SequenceController::render
component::CodeHighlighter::render
component::CodeHighlighter::highlight
scene::internal::BaseScene::render_options
gui::GuiQueue::render
component::SequenceController::set_progress_value
component::SequenceController::set_run_all
component::SequenceController::get_progress_value
component::SequenceController::get_anim_counter
component::SequenceController::get_run_all
component::SequenceController::get_speed_scale
Settings::get_instance
core::DoublyLinkedList::size
utils::adaptive_text_color
utils::DrawText
Settings::get_color
core::DoublyLinkedList::at
scene::internal::BaseScene::render_inputs

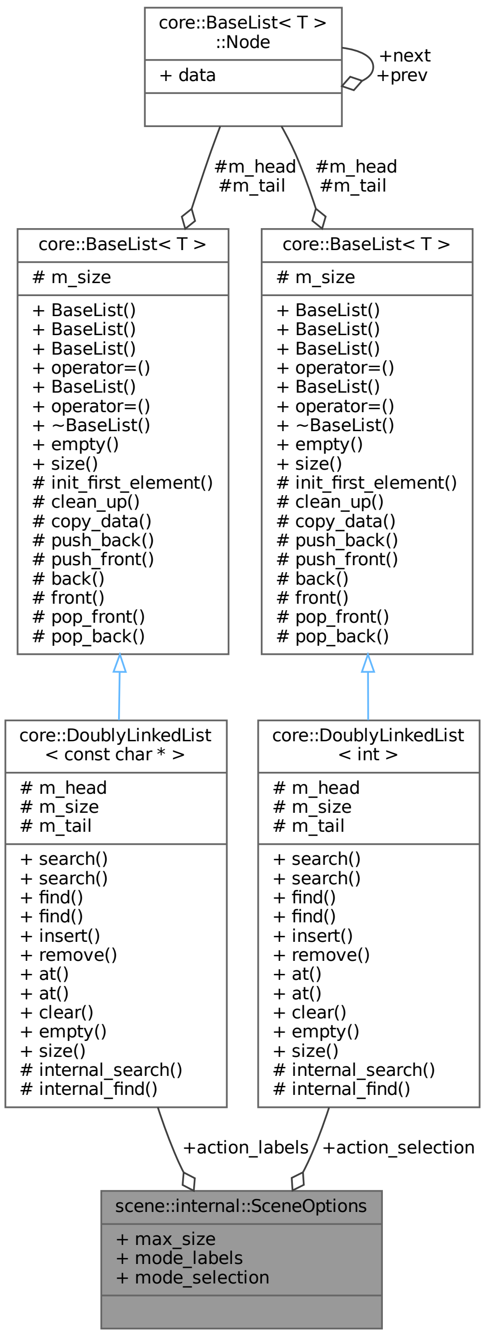
The documentation for this class was generated from the following files:

- 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.

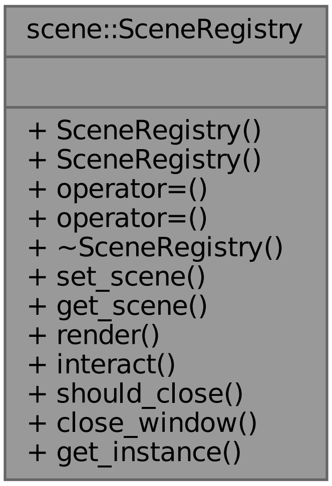
The documentation for this struct was generated from the following file:

- 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]
```

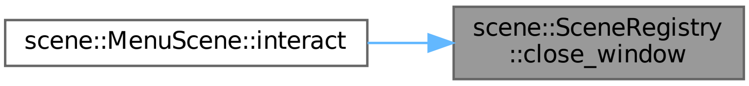
6.26.3 Member Function Documentation

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:

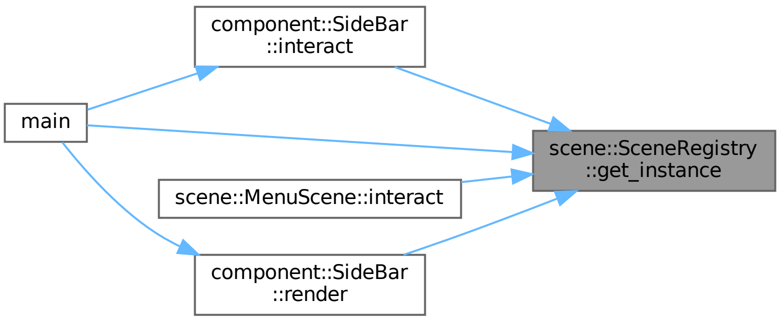


6.26.3.2 get_instance()

```
SceneRegistry & scene::SceneRegistry::get_instance ( )  [static]
```

Definition at line 7 of file scene_registry.cpp.

Here is the caller graph for this function:

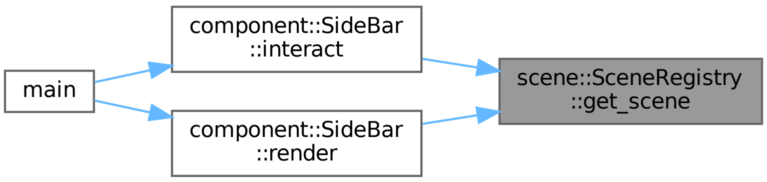

component::SideBar ::interact
main
scene::SceneRegistry ::get_instance
scene::MenuScene::interact
component::SideBar ::render

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:

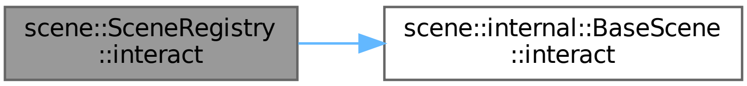

component::SideBar ::interact
main
scene::SceneRegistry ::get_scene
component::SideBar ::render

6.26.3.4 interact()

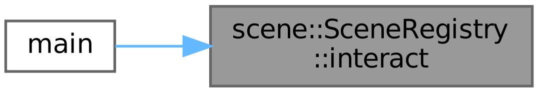
```
void scene::SceneRegistry::interact ( )
```

Definition at line 21 of file scene_registry.cpp.

Here is the call graph for this function:


scene::SceneRegistry ::interact
scene::internal::BaseScene ::interact

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]

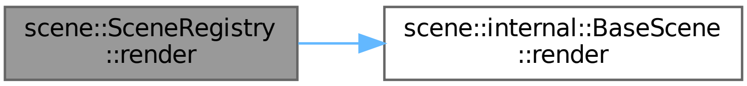
```
SceneRegistry & scene::SceneRegistry::operator= (
             SceneRegistry && )  [delete]
```

6.26.3.7 render()

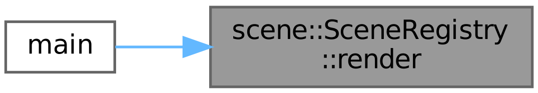
```
void scene::SceneRegistry::render ( )
```

Definition at line 19 of file scene_registry.cpp.

Here is the call graph for this function:


scene::SceneRegistry ::render
scene::internal::BaseScene ::render

Here is the caller graph for this function:

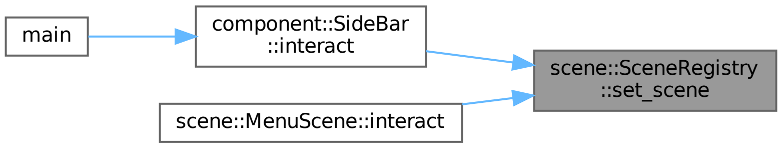


6.26.3.8 set_scene()

```
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:

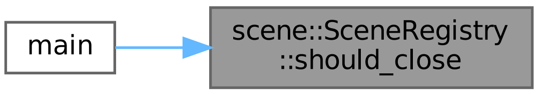

main
component::SideBar ::interact
scene::MenuScene::interact
scene::SceneRegistry ::set_scene

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:



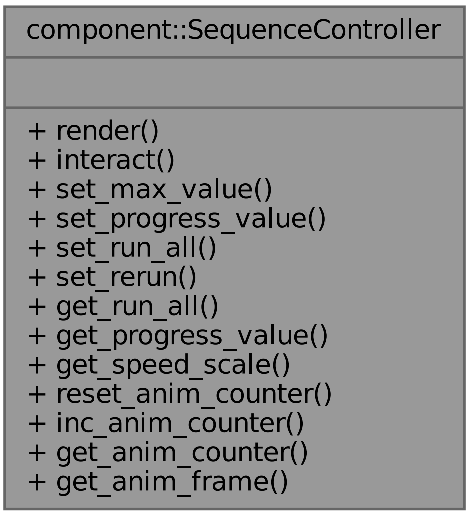
The documentation for this class was generated from the following files:

- 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.

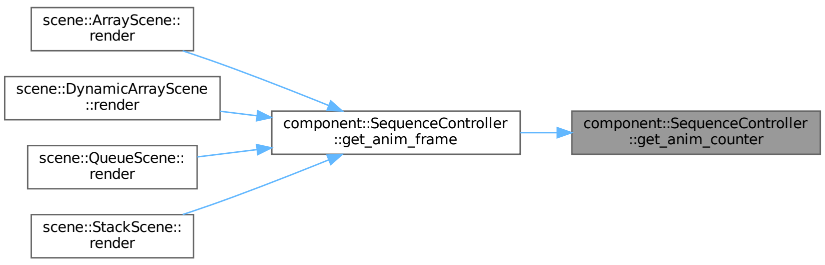
6.27.2 Member Function Documentation

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:

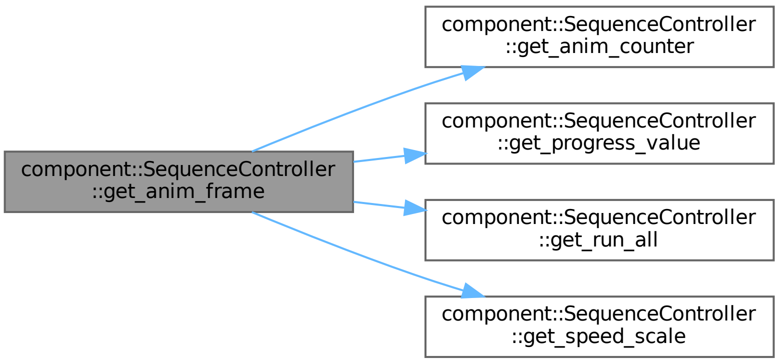


6.27.2.2 get_anim_frame()

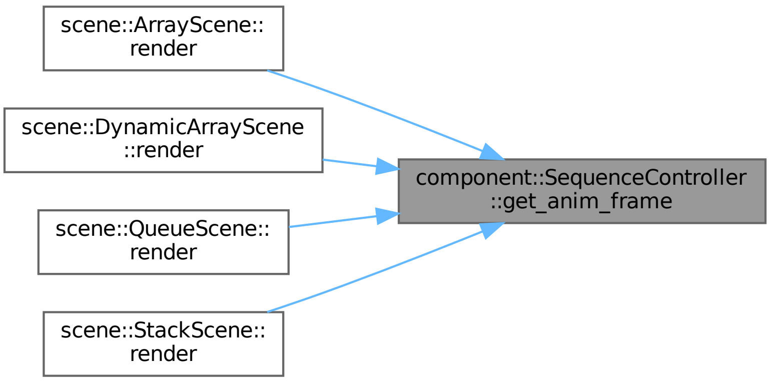
```
int component::SequenceController::get_anim_frame ( ) const
```

Definition at line 42 of file sequence_controller.cpp.

Here is the call graph for this function:


component::SequenceController ::get_anim_frame
component::SequenceController ::get_anim_counter
component::SequenceController ::get_progress_value
component::SequenceController ::get_run_all
component::SequenceController ::get_speed_scale

Here is the caller graph for this function:

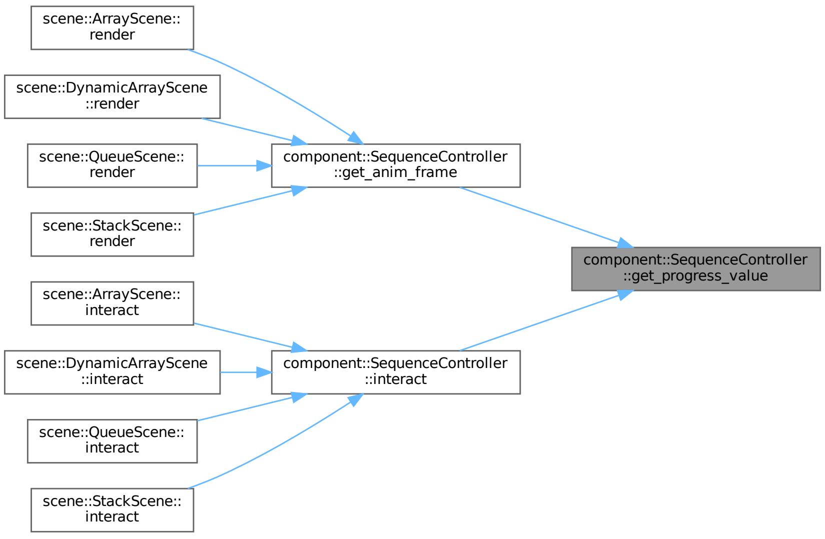

scene::ArrayScene:: render
scene::DynamicArrayScene ::render
scene::QueueScene:: render
scene::StackScene:: render
component::SequenceController ::get_anim_frame

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:

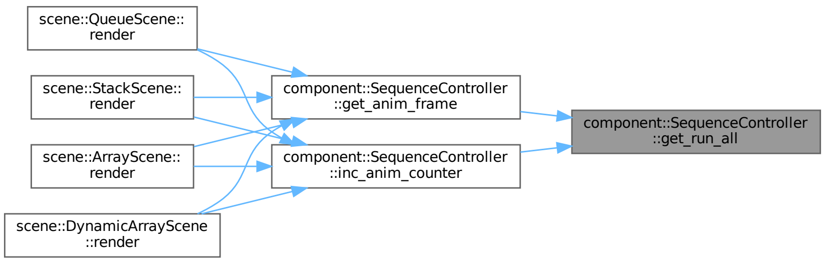


6.27.2.4 get_run_all()

```
bool component::SequenceController::get_run_all ( ) const
```

Definition at line 19 of file sequence_controller.cpp.

Here is the caller graph for this function:

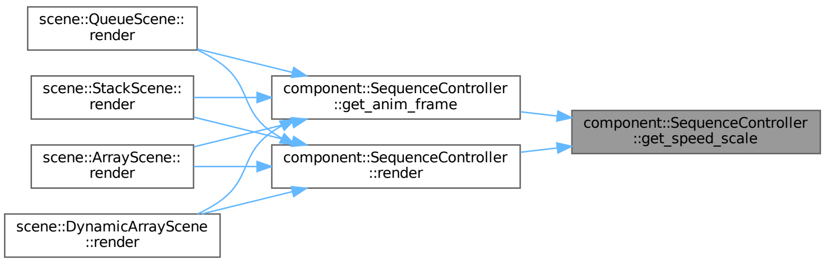


6.27.2.5 get_speed_scale()

```
float component::SequenceController::get_speed_scale ( ) const
```

Definition at line 23 of file sequence_controller.cpp.

Here is the caller graph for this function:

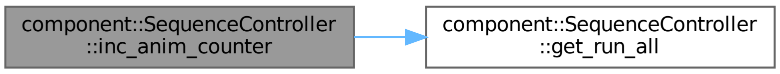

scene::QueueScene:: render
scene::StackScene:: render
scene::ArrayScene:: render
scene::DynamicArrayScene ::render
component::SequenceController ::get_anim_frame
component::SequenceController ::render
component::SequenceController ::get_speed_scale

6.27.2.6 inc_anim_counter()

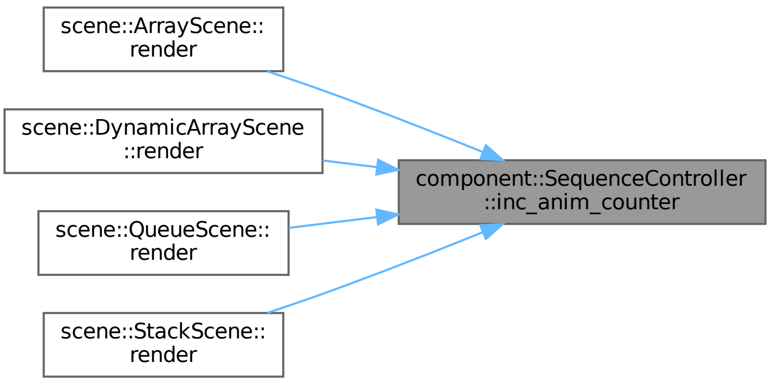
```
void component::SequenceController::inc_anim_counter ( )
```

Definition at line 29 of file sequence_controller.cpp.

Here is the call graph for this function:


component::SequenceController ::inc_anim_counter
component::SequenceController ::get_run_all

Here is the caller graph for this function:

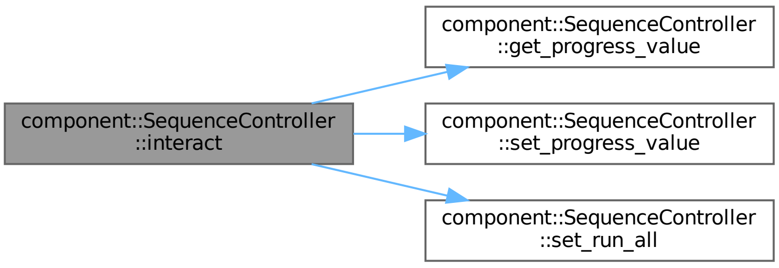

scene::ArrayScene:: render
scene::DynamicArrayScene ::render
scene::QueueScene:: render
scene::StackScene:: render
component::SequenceController ::inc_anim_counter

6.27.2.7 interact()

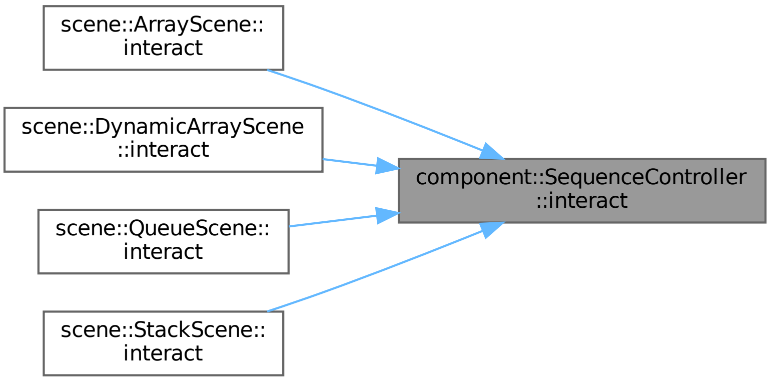
```
bool component::SequenceController::interact ( )
```

Definition at line 90 of file sequence_controller.cpp.

Here is the call graph for this function:


component::SequenceController ::interact
component::SequenceController ::get_progress_value
component::SequenceController ::set_progress_value
component::SequenceController ::set_run_all

Here is the caller graph for this function:

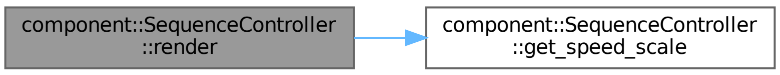

scene::ArrayScene:: interact
scene::DynamicArrayScene ::interact
scene::QueueScene:: interact
scene::StackScene:: interact
component::SequenceController ::interact

6.27.2.8 render()

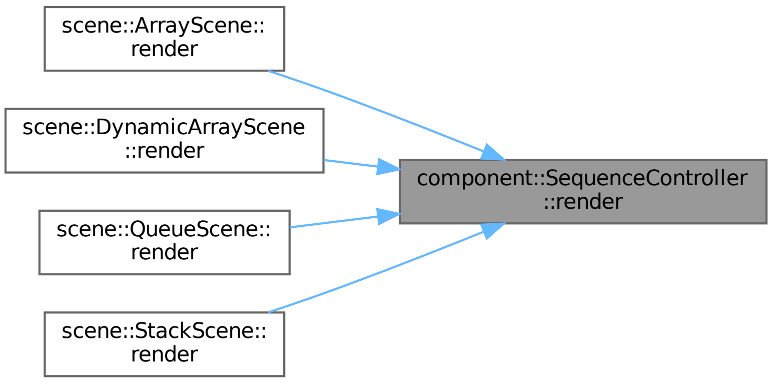
```
void component::SequenceController::render ( )
```

Definition at line 51 of file sequence_controller.cpp.

Here is the call graph for this function:


component::SequenceController ::render
component::SequenceController ::get_speed_scale

Here is the caller graph for this function:

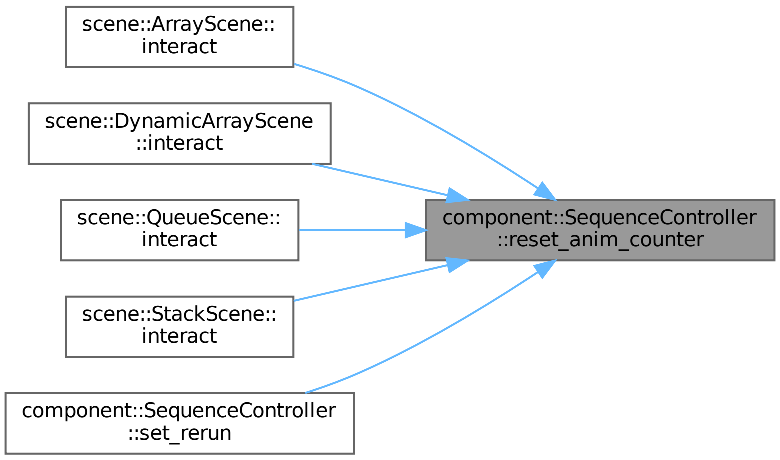

scene::ArrayScene:: render
scene::DynamicArrayScene ::render
scene::QueueScene:: render
scene::StackScene:: render
component::SequenceController ::render

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:


scene::ArrayScene::interact
scene::DynamicArrayScene::interact
scene::QueueScene::interact
scene::StackScene::interact
component::SequenceController::set_rerun
component::SequenceController::reset_anim_counter

6.27.2.10 set_max_value()

```
void component::SequenceController::set_max_value (
             int num )
```

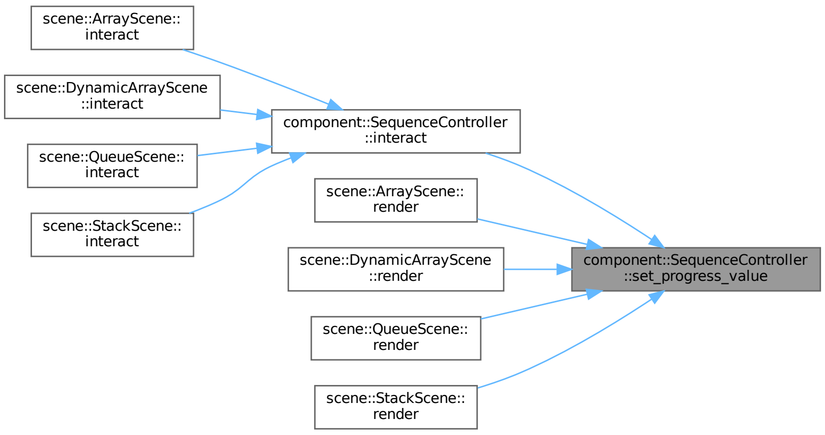
Definition at line 11 of file sequence_controller.cpp.

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:

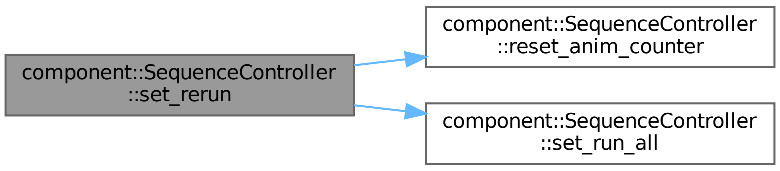


6.27.2.12 set_rerun()

```
void component::SequenceController::set_rerun ( )
```

Definition at line 37 of file sequence_controller.cpp.

Here is the call graph for this function:

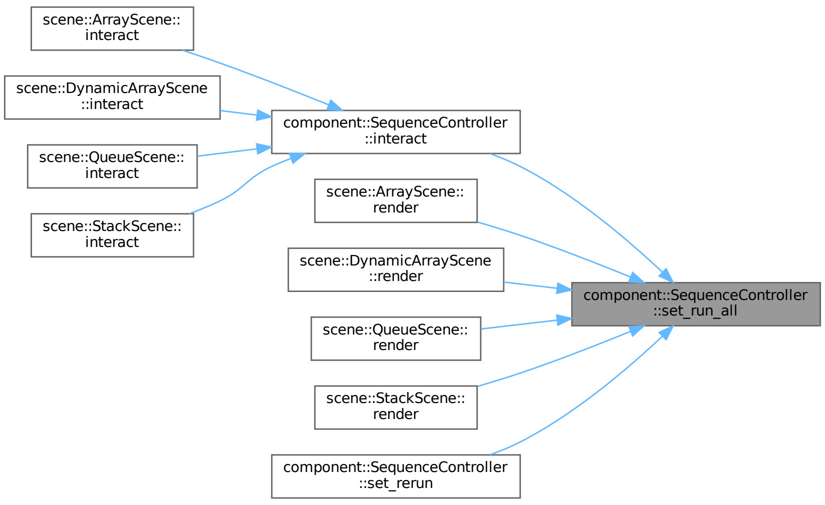

component::SequenceController
::set_rerun
component::SequenceController
::reset_anim_counter
component::SequenceController
::set_run_all

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:


scene::ArrayScene::interact
scene::DynamicArrayScene::interact
scene::QueueScene::interact
scene::StackScene::interact
component::SequenceController::interact
scene::ArrayScene::render
scene::DynamicArrayScene::render
scene::QueueScene::render
scene::StackScene::render
component::SequenceController::set_rerun
component::SequenceController::set_run_all

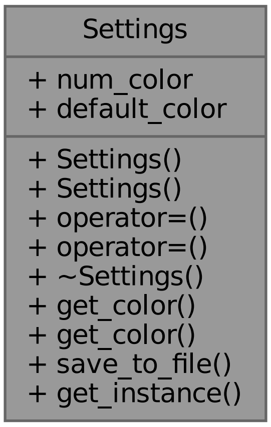
The documentation for this class was generated from the following files:

- 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]

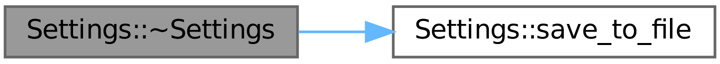
```
Settings::Settings (
             Settings &&  )  [delete]
```

6.28.2.3 ∼Settings()

```
Settings::∼Settings ( )
```

Definition at line 24 of file settings.cpp.

Here is the call graph for this function:



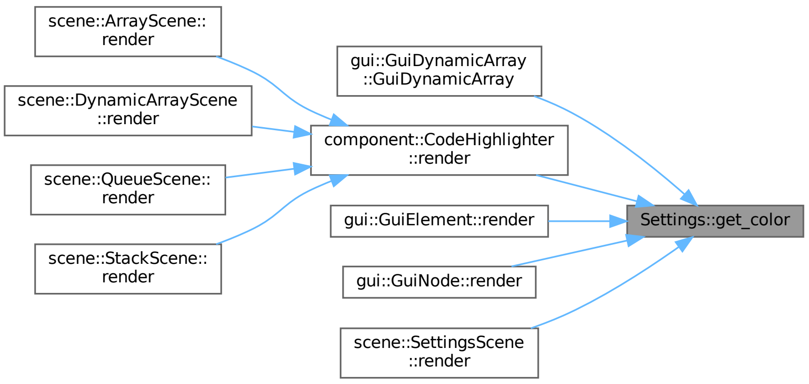
6.28.3 Member Function Documentation

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
```

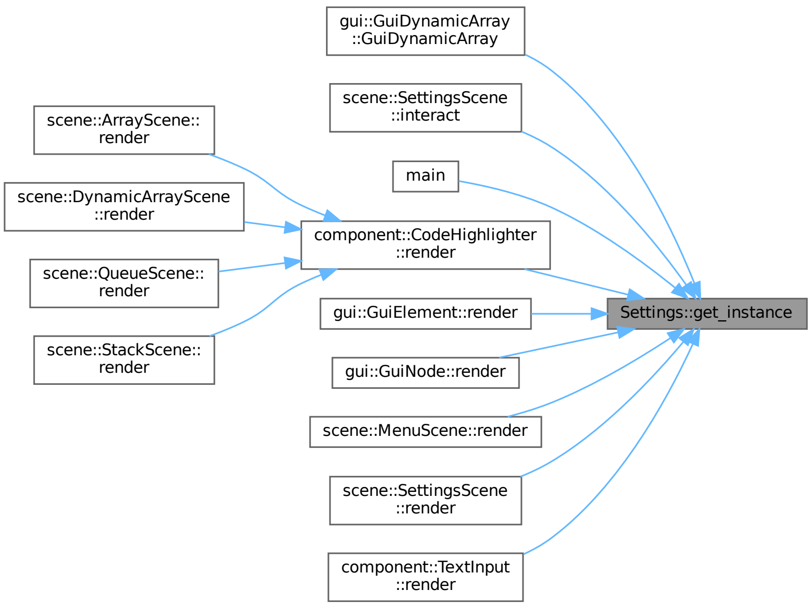
Definition at line 28 of file settings.cpp.

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:


gui::GuiDynamicArray ::GuiDynamicArray
scene::SettingsScene ::interact
scene::ArrayScene:: render
main
scene::DynamicArrayScene ::render
component::CodeHighlighter ::render
scene::QueueScene:: render
gui::GuiElement::render
Settings::get_instance
scene::StackScene:: render
gui::GuiNode::render
scene::MenuScene::render
scene::SettingsScene ::render
component::TextInput ::render

6.28.3.4 operator=() [1/2]

```
Settings & Settings::operator= (
            const Settings & )  [delete]
```

6.28.3.5 operator=() [2/2]

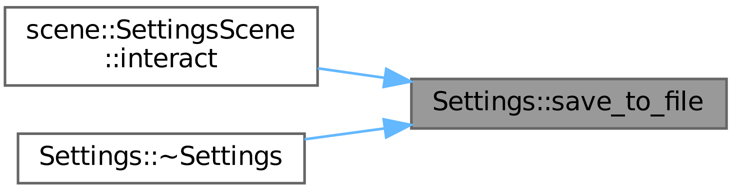
```
Settings & Settings::operator= (
             Settings && ) [delete]
```

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,
        0xffa10000,
        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.

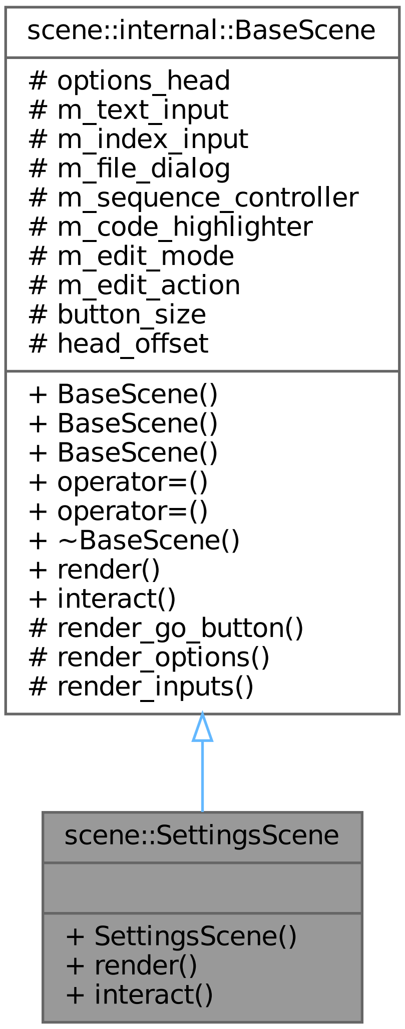
The documentation for this class was generated from the following files:

- src/settings.hpp
- src/settings.cpp

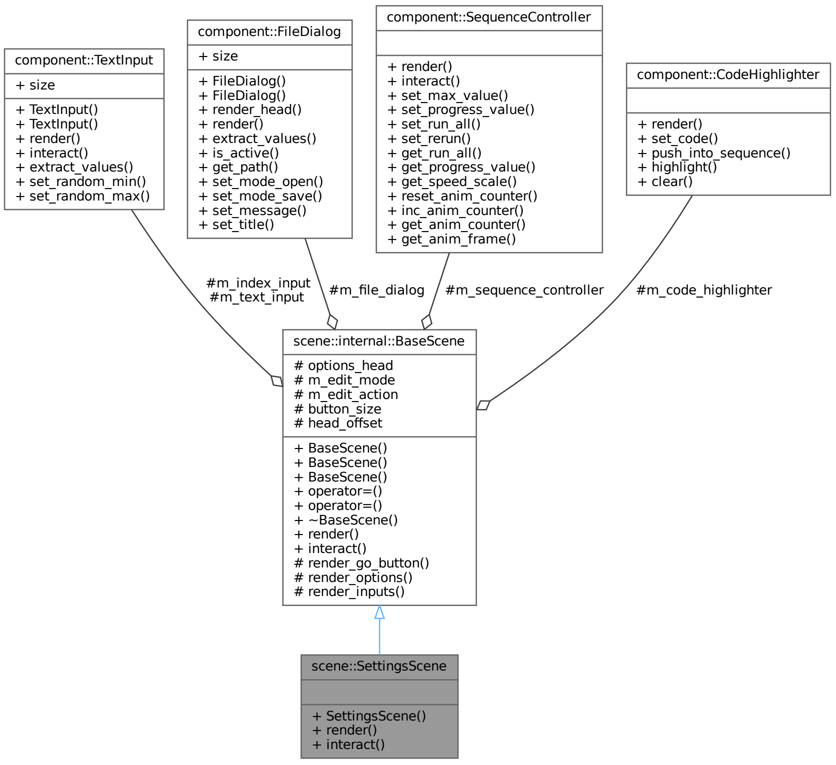
6.29 scene::SettingsScene Class Reference

```
#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

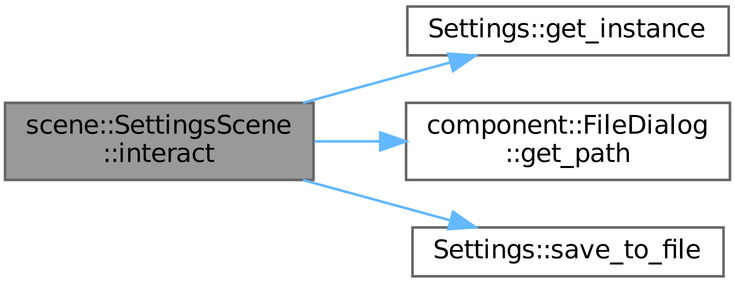
6.29.3.1 interact()

```
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:


Settings::get_instance
scene::SettingsScene ::interact
component::FileDialog ::get_path
Settings::save_to_file

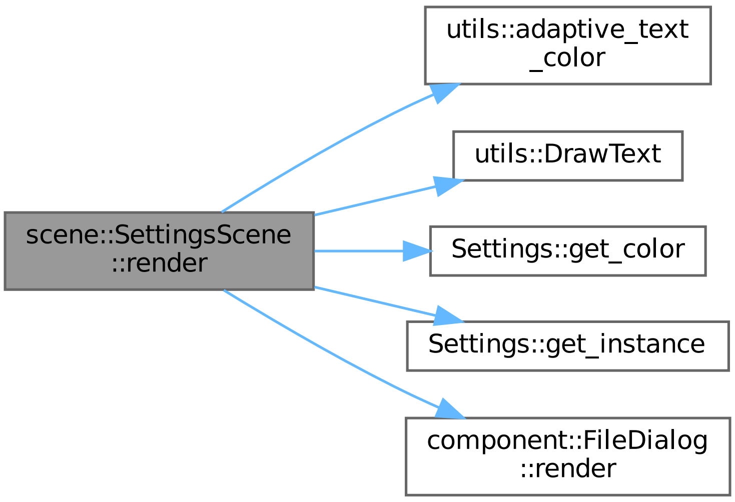
6.29.3.2 render()

```
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:


utils::adaptive_text _color
utils::DrawText
scene::SettingsScene ::render
Settings::get_color
Settings::get_instance
component::FileDialog ::render

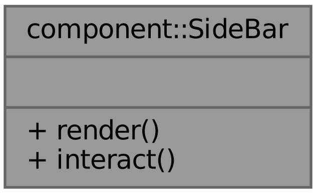
The documentation for this class was generated from the following files:

- 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.

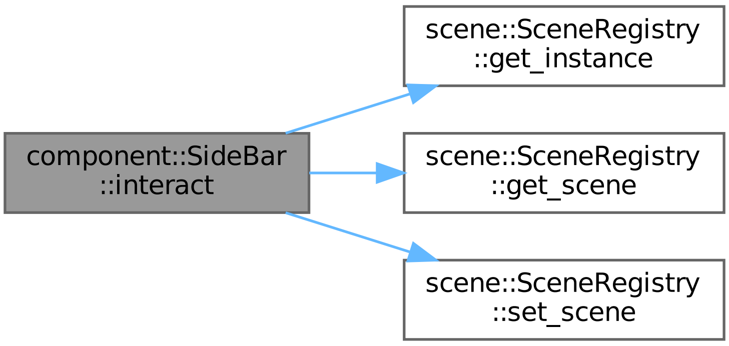
6.30.2 Member Function Documentation

6.30.2.1 interact()

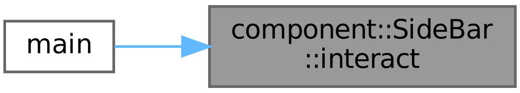
```
void component::SideBar::interact ( )
```

Definition at line 48 of file sidebar.cpp.

Here is the call graph for this function:


component::SideBar ::interact
scene::SceneRegistry ::get_instance
scene::SceneRegistry ::get_scene
scene::SceneRegistry ::set_scene

Here is the caller graph for this function:

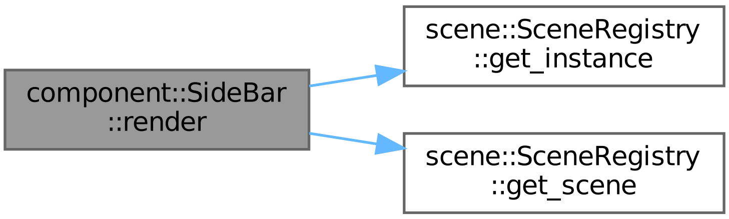


6.30.2.2 render()

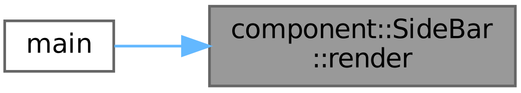
```
void component::SideBar::render ( )
```

Definition at line 11 of file sidebar.cpp.

Here is the call graph for this function:


component::SideBar ::render
scene::SceneRegistry ::get_instance
scene::SceneRegistry ::get_scene

Here is the caller graph for this function:



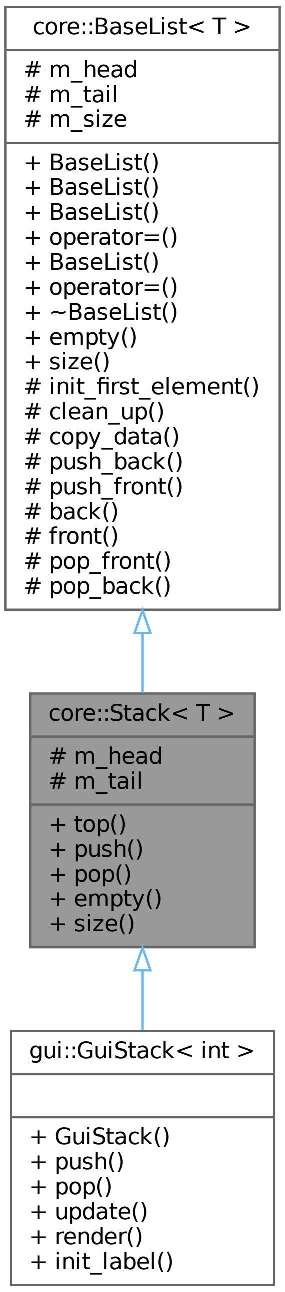
The documentation for this class was generated from the following files:

- src/component/sidebar.hpp
- src/component/sidebar.cpp

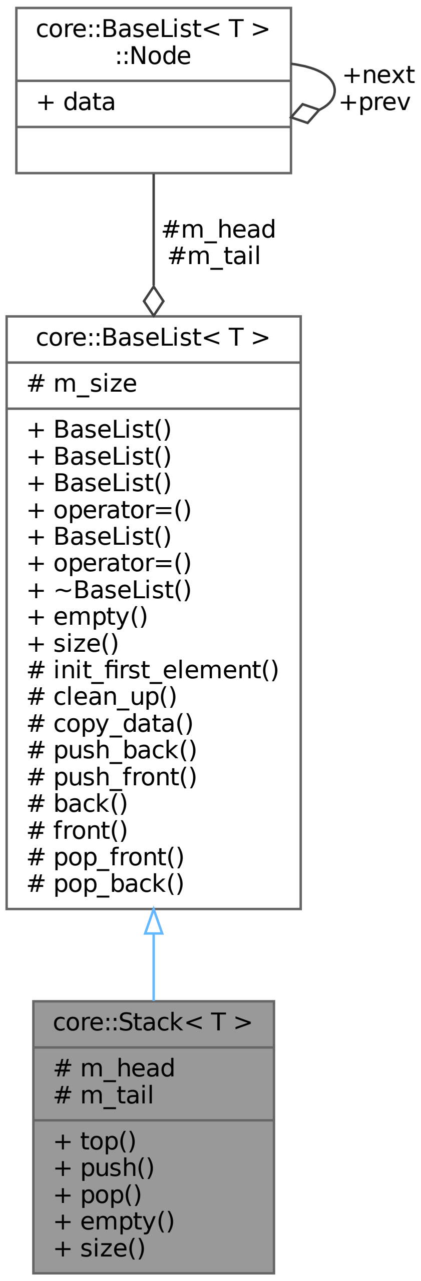
6.31 core::Stack< T > Class Template Reference

```
#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 )
```

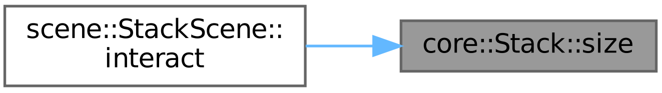
Definition at line 33 of file stack.hpp.

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.

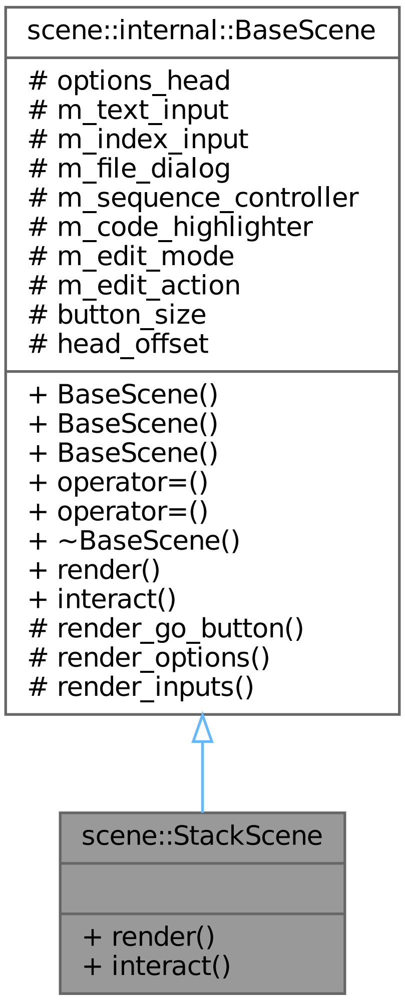
The documentation for this class was generated from the following file:

- src/core/stack.hpp

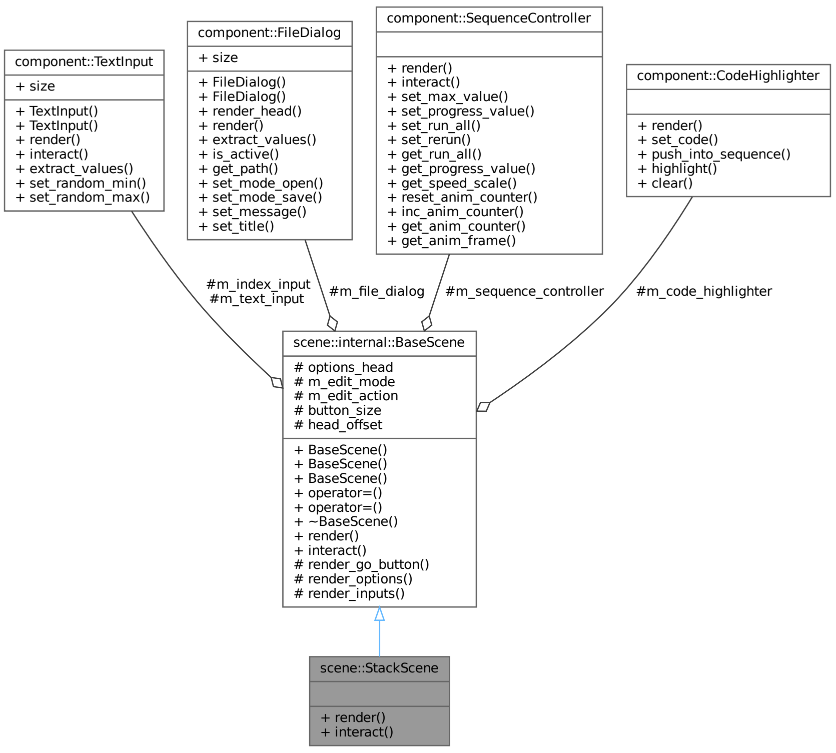
6.32 scene::StackScene Class Reference

```
#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

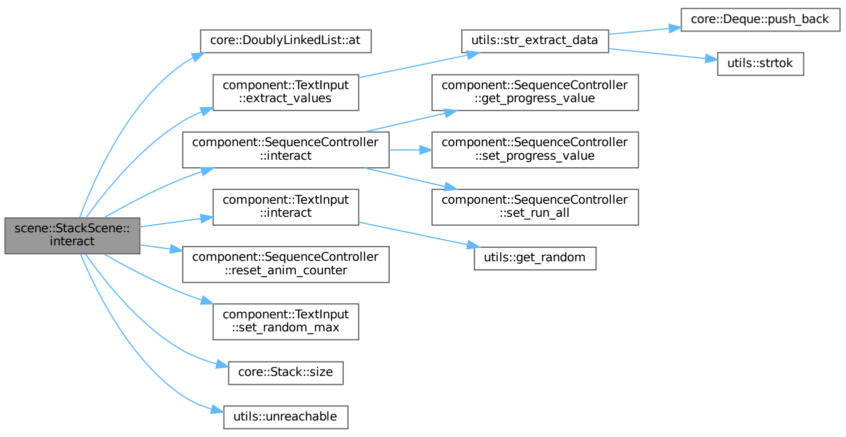
6.32.2.1 interact()

```
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:



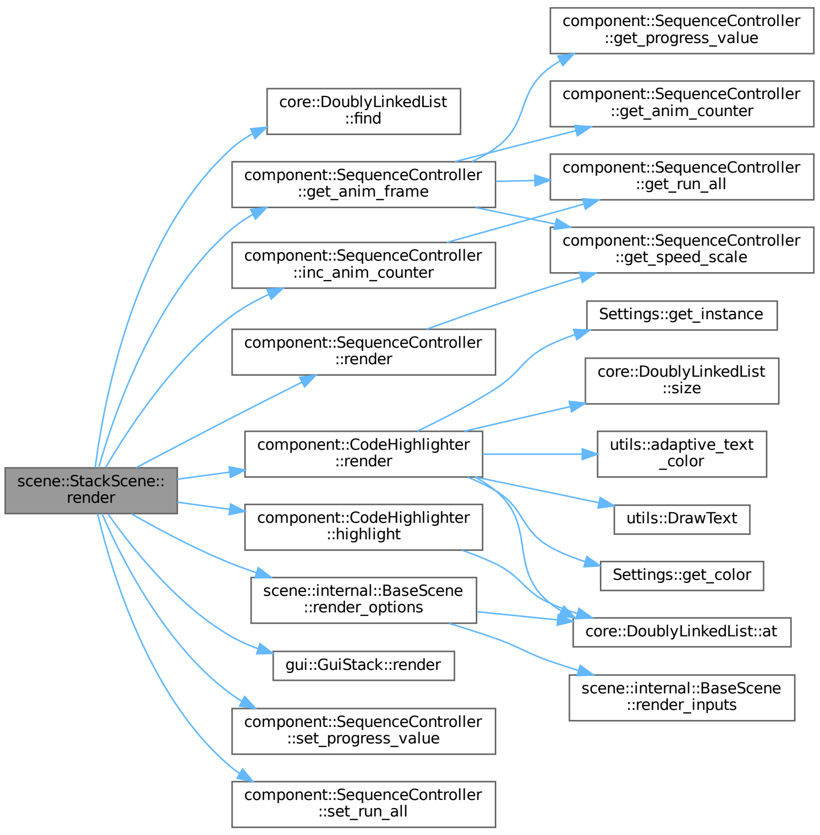
6.32.2.2 render()

```
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:


scene::StackScene::
render
core::DoublyLinkedList
::find
component::SequenceController
::get_anim_frame
component::SequenceController
::inc_anim_counter
component::SequenceController
::render
component::CodeHighlighter
::render
component::CodeHighlighter
::highlight
scene::internal::BaseScene
::render_options
gui::GuiStack::render
component::SequenceController
::set_progress_value
component::SequenceController
::set_run_all
component::SequenceController
::get_progress_value
component::SequenceController
::get_anim_counter
component::SequenceController
::get_run_all
component::SequenceController
::get_speed_scale
Settings::get_instance
core::DoublyLinkedList
::size
utils::adaptive_text
_color
utils::DrawText
Settings::get_color
core::DoublyLinkedList::at
scene::internal::BaseScene
::render_inputs

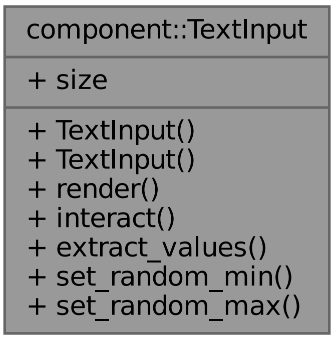
The documentation for this class was generated from the following files:

- 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.

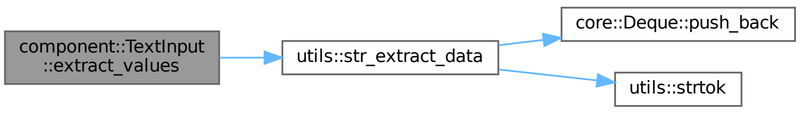
6.33.3 Member Function Documentation

6.33.3.1 extract_values()

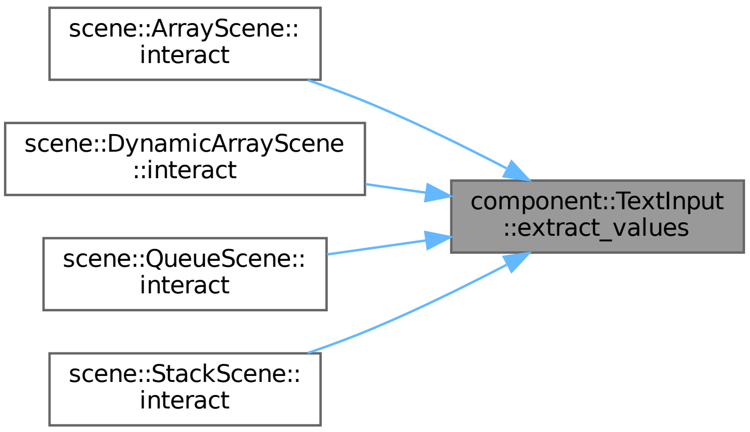
```
core::Deque< int > component::TextInput::extract_values ( )
```

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:

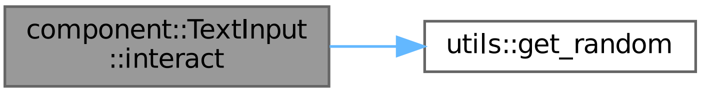

scene::ArrayScene:: interact
scene::DynamicArrayScene ::interact
scene::QueueScene:: interact
scene::StackScene:: interact
component::TextInput ::extract_values

6.33.3.2 interact()

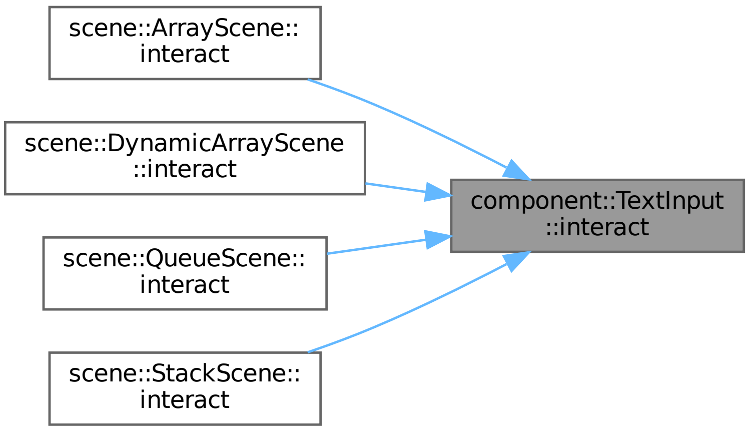
```
bool component::TextInput::interact ( )
```

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:

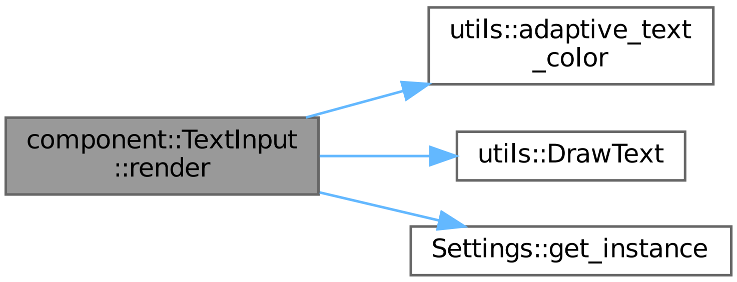

scene::ArrayScene:: interact
scene::DynamicArrayScene ::interact
scene::QueueScene:: interact
scene::StackScene:: interact
component::TextInput ::interact

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:

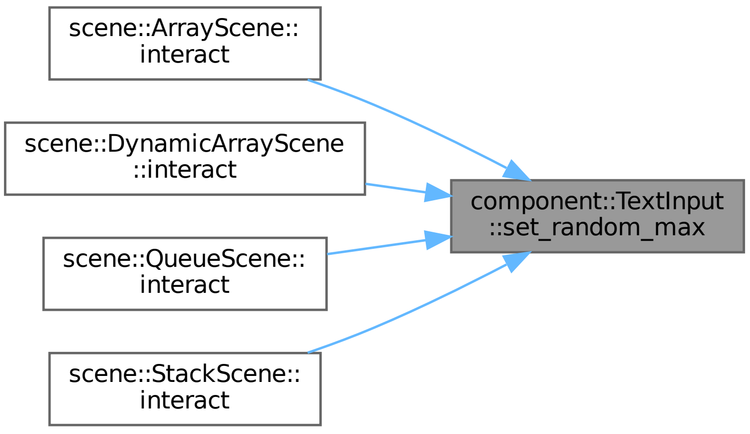


6.33.3.4 set_random_max()

```
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

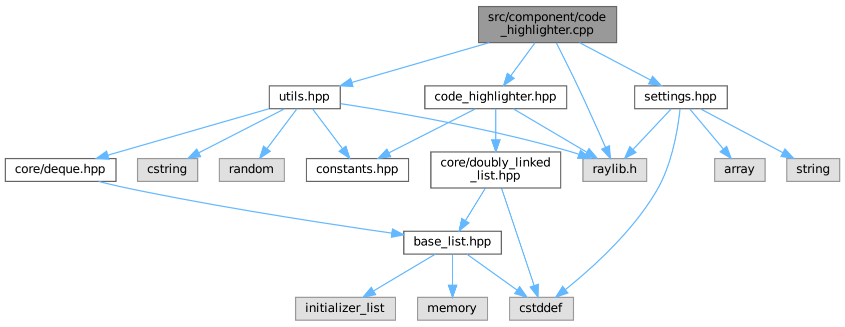
Chapter 7

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.

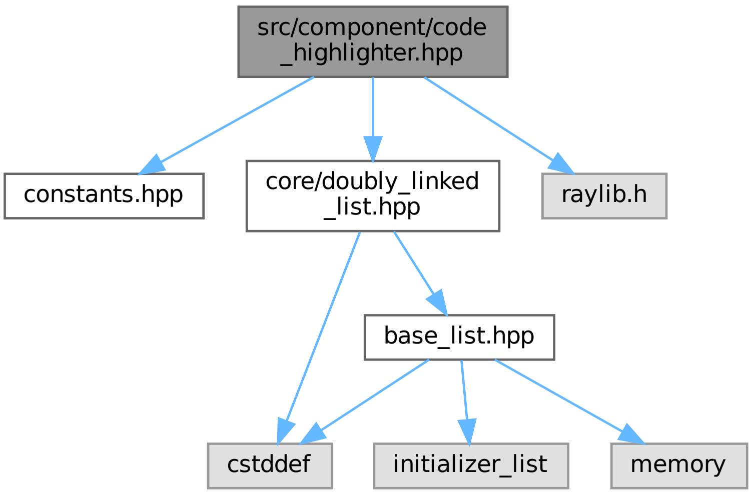
```
#include "code_highlighter.hpp"

#include "raylib.h"
#include "settings.hpp"
#include "utils.hpp"

namespace component {

```

```
void CodeHighlighter::render() {
    for (int i = 0; i < m_src_code.size(); ++i) {
        const Settings& settings = Settings::get_instance();

        int color_index = (i == m_highlighted_line) ? 4 : 0;
        Color bg_color = settings.get_color(color_index);
        Color text_color = utils::adaptive_text_color(bg_color);

        Rectangle shape{head_pos.x, head_pos.y + i * height, width, height};
        Vector2 text_head = {head_pos.x + 10, head_pos.y + i * height + 5};

        DrawRectangleRec(shape, bg_color);
        utils::DrawText(m_src_code.at(i), text_head, text_color, 20, 2);
    }
}

void CodeHighlighter::set_code(core::DoublyLinkedList<const char*>&& src_code) {
    clear();
    m_src_code = src_code;
}

void CodeHighlighter::push_into_sequence(int line_number) {
    m_sequence.insert(m_sequence.size(), line_number);
}

void CodeHighlighter::highlight(int frame_idx) {
    m_highlighted_line = m_sequence.at(frame_idx);
}

void CodeHighlighter::clear() {
    m_src_code.clear();
    m_sequence.clear();
}

}  // namespace component
```

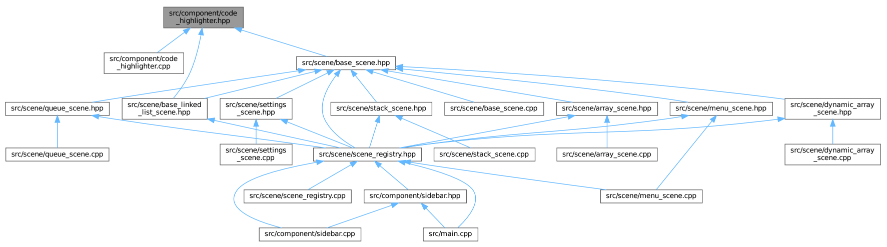
7.3 src/component/code_highlighter.hpp File Reference

```
#include "constants.hpp"
#include "core/doubly_linked_list.hpp"
#include "raylib.h"
```

Include dependency graph for code_highlighter.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class component::CodeHighlighter

Namespaces

- namespace component

7.4 code_highlighter.hpp

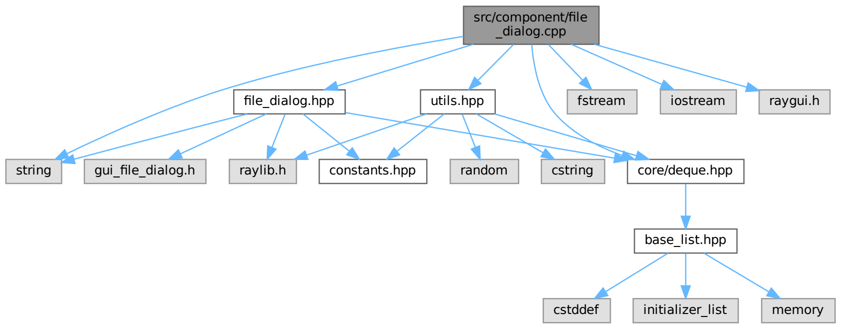
Go to the documentation of this file.

```
#ifndef COMPONENT_CODE_HIGHLIGHTER_HPP_
#define COMPONENT_CODE_HIGHLIGHTER_HPP_

#include "constants.hpp"
#include "core/doubly_linked_list.hpp"
#include "raylib.h"

namespace component {

class CodeHighlighter {
private:
    static constexpr int width = 400;
    static constexpr int height = 30;
    static constexpr Vector2 head_pos{constants::scene_width - width,
                                      2.5F * height};

    core::DoublyLinkedList<const char*> m_src_code;
    core::DoublyLinkedList<int> m_sequence;
    int m_highlighted_line{-1};

public:
    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();
};

}  // namespace component

#endif  // COMPONENT_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

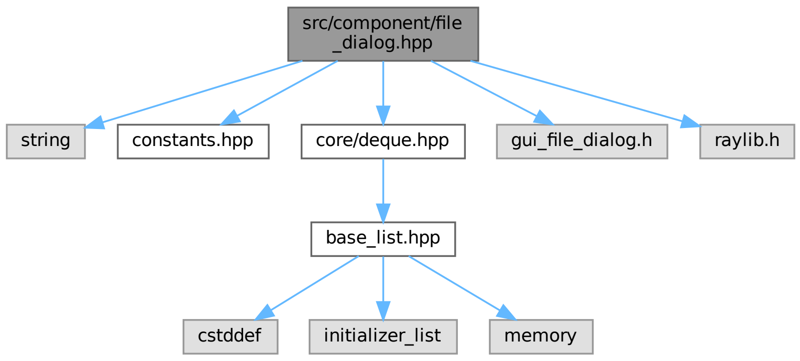
Go to the documentation of this file.

```
#include "file_dialog.hpp"

#include <fstream>
#include <iostream>
#include <string>

#include "core/deque.hpp"
#include "raygui.h"
#include "utils.hpp"

namespace component {

FileDialog::FileDialog(int mode, const char* title, const char* message)
    : m_mode{mode}, m_title{title}, m_message{message} {}

FileDialog::FileDialog() : FileDialog(0, "Open file...", "Open file") {}

int FileDialog::render(float x, float y) {
    m_file_dialog_state.title = m_title;
    m_file_dialog_state.fileName = m_file_input;
    m_file_dialog_state.message = m_message;
    m_file_dialog_state.dialogType = m_mode;

    int result = -1;
    if (m_file_dialog_state.windowActive) {
        GuiLock();
        result = GuiFileDialog(&m_file_dialog_state);
        if (result >= 0) {
            m_file_dialog_state.windowActive = false;
        }
    }

    const Rectangle shape{x, y, size.x, size.y};

    if (GuiButton(shape, GuiIconText(ICON_FILE_OPEN, "Select file"))) {
        m_file_dialog_state.windowActive = true;
    }

    GuiUnlock();
    return result;
}

int FileDialog::render_head(float& options_head, float head_offset) {
    int ret = render(options_head, constants::scene_height - size.y);
    options_head += (size.x + head_offset);
    return ret;
}

core::Deque<int> FileDialog::extract_values() {
    std::ifstream ifs(get_path());
    char buffer[constants::text_buffer_size]{};  // NOLINT
    ifs » buffer;

    return utils::str_extract_data(buffer);  // NOLINT
}

bool FileDialog::is_active() const { return m_file_dialog_state.windowActive; }

void FileDialog::set_mode_open() { m_mode = DIALOG_OPEN_FILE; }

void FileDialog::set_mode_save() { m_mode = DIALOG_SAVE_FILE; }

void FileDialog::set_message(const char* message) { m_message = message; }

void FileDialog::set_title(const char* title) { m_title = title; }
std::string FileDialog::get_path() { return m_file_input; }

}  // namespace component
```

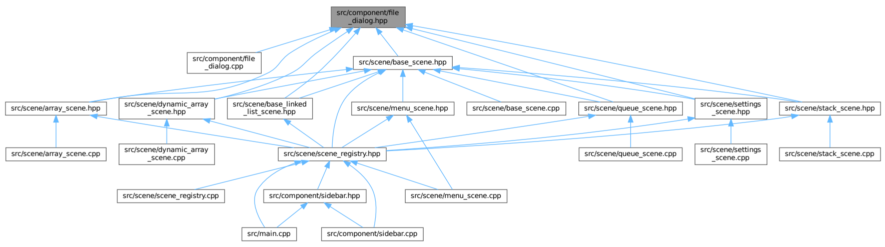
7.7 src/component/file_dialog.hpp File Reference

```
#include <string>
#include "constants.hpp"
#include "core/deque.hpp"
#include "gui_file_dialog.h"
#include "raylib.h"
```

Include dependency graph for file_dialog.hpp:



This graph shows which files directly or indirectly include this file:


src/component/file_dialog.hpp
src/component/file_dialog.cpp
src/scene/base_scene.hpp
src/scene/array_scene.hpp
src/scene/dynamic_array_scene.hpp
src/scene/base_linked_list_scene.hpp
src/scene/menu_scene.hpp
src/scene/base_scene.cpp
src/scene/queue_scene.hpp
src/scene/settings_scene.hpp
src/scene/stack_scene.hpp
src/scene/array_scene.cpp
src/scene/dynamic_array_scene.cpp
src/scene/scene_registry.hpp
src/scene/queue_scene.cpp
src/scene/settings_scene.cpp
src/scene/stack_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/main.cpp
src/component/sidebar.cpp

Classes

- class component::FileDialog

Namespaces

- namespace component

7.8 file_dialog.hpp

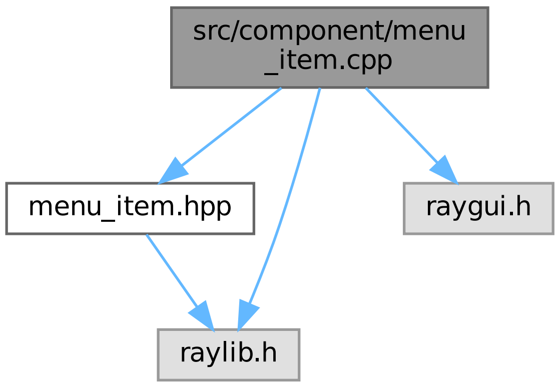
Go to the documentation of this file.

```
#ifndef COMPONENT_FILE_DIALOG_HPP_
#define COMPONENT_FILE_DIALOG_HPP_

#include <string>

#include "constants.hpp"
#include "core/deque.hpp"
#include "gui_file_dialog.h"
#include "raylib.h"

namespace component {

class FileDialog {
private:
    GuiFileDialogState m_file_dialog_state{
        InitGuiFileDialog(GetWorkingDirectory())};

    char m_file_input[constants::text_buffer_size] = "";  // NOLINT

    int m_mode{};
    const char* m_message;
    const char* m_title;

public:
    static constexpr Vector2 size{200, 50};

    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);
};

}  // namespace component

#endif  // COMPONENT_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

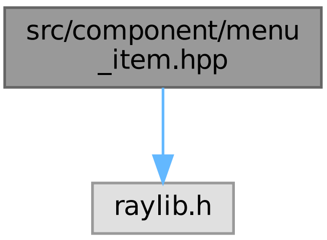
Go to the documentation of this file.

```
#include "menu_item.hpp"

#include "raygui.h"
#include "raylib.h"

namespace component {

MenuItem::MenuItem(int scene, const char* text, int x, int y,
                   const char* img_path)
    : m_scene{scene},
      m_text{text},
      m_x{x},
      m_y{y},
      m_texture{LoadTextureFromImage(LoadImage(img_path))} {}

int MenuItem::x() const { return m_x; }
int MenuItem::y() const { return m_y; }

void MenuItem::render() {
    auto mouse = GetMousePosition();
    const Rectangle bound{(float)m_x, (float)m_y, block_width, block_height};
    const Rectangle text_bound{(float)m_x + 20,
                               (float)m_y + block_height - button_height,
                               button_width - 20, button_height};

    DrawRectangleRec(bound, RAYWHITE);
    DrawTexture(m_texture, m_x, m_y, WHITE);
    GuiLabelButton(text_bound, m_text);
    DrawRectangleLinesEx(bound, 2, BLACK);

    if (CheckCollisionPointRec(mouse, bound)) {
        DrawRectangle(m_x, m_y, block_width, block_height,
                      ColorAlpha(BLUE, 0.3));
        m_clicked = IsMouseButtonPressed(MOUSE_LEFT_BUTTON);
    }
}

bool MenuItem::clicked() const { return m_clicked; }

void MenuItem::reset() { m_clicked = false; }

}  // namespace component
```

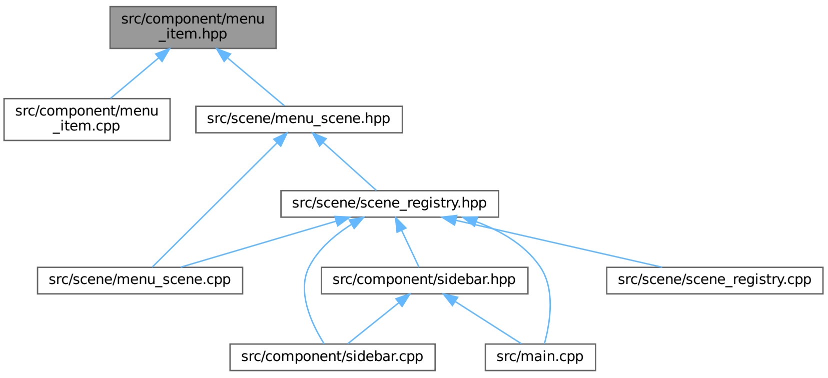
7.11 src/component/menu_item.hpp File Reference

`#include "raylib.h"`

Include dependency graph for menu_item.hpp:



This graph shows which files directly or indirectly include this file:


src/component/menu_item.hpp
src/component/menu_item.cpp
src/scene/menu_scene.hpp
src/scene/scene_registry.hpp
src/scene/menu_scene.cpp
src/component/sidebar.hpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class component::MenuItem

Namespaces

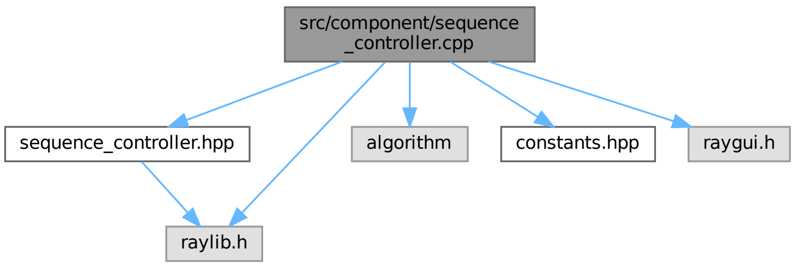
- namespace component

7.12 menu_item.hpp

Go to the documentation of this file.
```
#ifndef COMPONENT_MENU_ITEM_HPP_
#define COMPONENT_MENU_ITEM_HPP_

#include "raylib.h"

namespace component {

class MenuItem {
private:
    int m_scene{};
    int m_x{};
    int m_y{};
    Texture2D m_texture{};
    const char* m_text{};

    bool m_clicked{};

public:
    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;

    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();
};

}  // namespace component

#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

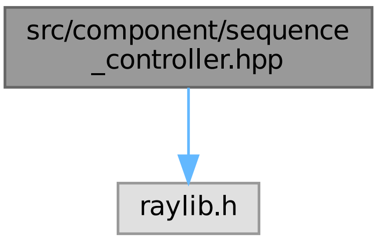
Go to the documentation of this file.

```cpp
#include "sequence_controller.hpp"

#include <algorithm>

#include "constants.hpp"
#include "raygui.h"
#include "raylib.h"

namespace component {

void SequenceController::set_max_value(int num) { m_num_steps = num; }

void SequenceController::set_progress_value(int value) {
    m_progress_value = value;
}

void SequenceController::set_run_all(bool run_all) { m_run_all = run_all; }

bool SequenceController::get_run_all() const { return m_run_all; }

int SequenceController::get_progress_value() const { return m_progress_value; }

float SequenceController::get_speed_scale() const {
    return (float)m_speed / speed_scale;
}

void SequenceController::reset_anim_counter() { m_anim_counter = 0; }

void SequenceController::inc_anim_counter() {
    if (get_run_all()) {
        ++m_anim_counter;
    }
}

int SequenceController::get_anim_counter() const { return m_anim_counter; }

void SequenceController::set_rerun() {
    reset_anim_counter();
    set_run_all(true);
}

int SequenceController::get_anim_frame() const {
    if (get_run_all()) {
        return 2.0F * get_anim_counter() * get_speed_scale() /
               constants::frames_per_second;
    } else {
        return get_progress_value();
    }
}

void SequenceController::render() {
    Rectangle replay_shape{button_size.x * 0.5F,
                           constants::scene_height - 1.5F * button_size.x,
                           button_size.x, button_size.y};

    Rectangle prev_frame_shape{
        replay_shape.x + replay_shape.width + button_size.x * 0.5F,
        replay_shape.y, button_size.x, button_size.y};

    Rectangle progress_shape{prev_frame_shape.x + button_size.x * 1.5F,
                             replay_shape.y, 360, button_size.y};

    Rectangle next_frame_shape{
        progress_shape.x + progress_shape.width + button_size.x * 0.5F,
        replay_shape.y, button_size.x, button_size.y};

    Rectangle prev_speed_shape{prev_frame_shape.x + 240,
                               prev_frame_shape.y - 1.5F * button_size.y,
                               button_size.x, button_size.y};

    Rectangle next_speed_shape{next_frame_shape.x,
                               next_frame_shape.y - 1.5F * button_size.y,
                               button_size.x, button_size.y};

    Rectangle speed_shape{prev_speed_shape.x + 1.5F * button_size.x,
                          prev_speed_shape.y, 120, button_size.y};

    m_prev_speed = GuiButton(prev_speed_shape, "#114#");
    m_next_speed = GuiButton(next_speed_shape, "#115#");
    GuiStatusBar(speed_shape, TextFormat("Speed: %.2fx", get_speed_scale()));

    m_replay = GuiButton(replay_shape, "#75#");
    m_prev_frame = GuiButton(prev_frame_shape, "#72#");
    m_progress_value =
        (int)GuiProgressBar(progress_shape, nullptr, nullptr,
                            (float)m_progress_value, 0, (float)m_num_steps);
    m_next_frame = GuiButton(next_frame_shape, "#73#");
}

bool SequenceController::interact() {
    if (m_replay) {
        set_progress_value(0);
        set_run_all(true);
        return true;
    }

    if (m_prev_frame) {
        set_progress_value(std::max(get_progress_value() - 1, 0));
        return true;
    }

    if (m_next_frame) {
        set_progress_value(std::min(get_progress_value() + 1, m_num_steps));
        return true;
    }

    if (m_prev_speed) {
        m_speed = std::max(m_speed - 1, 2);
        return true;
    }

    if (m_next_speed) {
        m_speed = std::min(m_speed + 1, 6);
        return true;
    }

    return false;
}

}  // namespace component
```

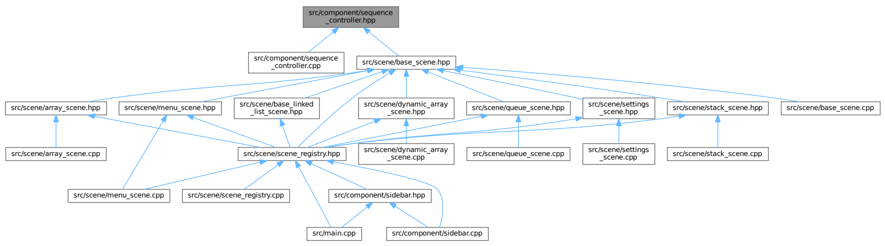
7.15 src/component/sequence_controller.hpp File Reference

`#include "raylib.h"`

Include dependency graph for sequence_controller.hpp:



This graph shows which files directly or indirectly include this file:


src/component/sequence_controller.hpp
src/component/sequence_controller.cpp
src/scene/base_scene.hpp
src/scene/array_scene.hpp
src/scene/menu_scene.hpp
src/scene/base_linked_list_scene.hpp
src/scene/dynamic_array_scene.hpp
src/scene/queue_scene.hpp
src/scene/settings_scene.hpp
src/scene/stack_scene.hpp
src/scene/base_scene.cpp
src/scene/array_scene.cpp
src/scene/scene_registry.hpp
src/scene/dynamic_array_scene.cpp
src/scene/queue_scene.cpp
src/scene/settings_scene.cpp
src/scene/stack_scene.cpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.hpp
src/main.cpp
src/component/sidebar.cpp

Classes

- class component::SequenceController

Namespaces

- namespace component

7.16 sequence_controller.hpp

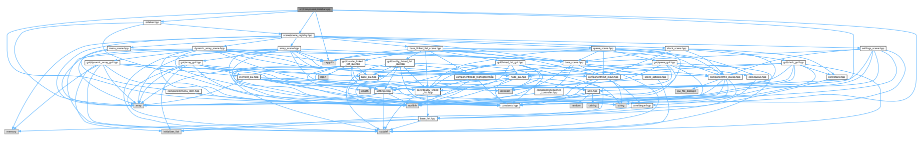
Go to the documentation of this file.

```
#ifndef COMPONENT_SEQUENCE_CONTROLLER_HPP_
#define COMPONENT_SEQUENCE_CONTROLLER_HPP_

#include "raylib.h"

namespace component {

class SequenceController {
private:
    static constexpr Vector2 button_size{25, 25};
    static constexpr int speed_scale = 4;

    bool m_replay{};
    bool m_prev_frame{};
    bool m_next_frame{};
    int m_progress_value{};
    int m_num_steps{};
    bool m_run_all{};
    int m_anim_counter{};

    bool m_prev_speed{};
    bool m_next_speed{};
    int m_speed{speed_scale};

public:
    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;
};

}  // namespace component

#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

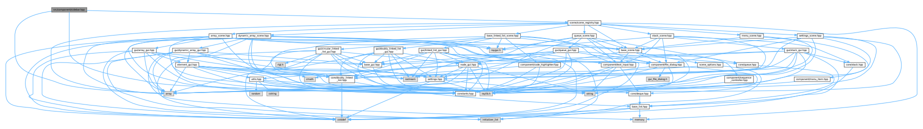
Go to the documentation of this file.

```
#include "sidebar.hpp"

#include "constants.hpp"
#include "raygui.h"
#include "raylib.h"
#include "scene/scene_registry.hpp"
#include "utils.hpp"

namespace component {

void SideBar::render() {
    (m_edit_mode) ? GuiLock() : GuiUnlock();

    scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();
    int options_head = 2 * constants::sidebar_width;

    constexpr float scale = 0.2;

    constexpr Rectangle menu_button_shape{20, 20, button_height * 2,
                                          button_height};
    constexpr Rectangle selection_shape{
        menu_button_shape.x + menu_button_shape.width + 10, menu_button_shape.y,
        button_width, button_height};
    constexpr Rectangle settings_button_shape{
        constants::scene_width - button_height - 20, 20, button_height,
        button_height};

    m_next_scene = registry.get_scene();

    bool menu_is_next = m_next_scene == scene::Menu;
    bool settings_is_next = m_next_scene == scene::Settings;

    if (!menu_is_next) {
        m_return_menu = GuiButton(menu_button_shape, "#118#Menu");
    }

    if (!menu_is_next && !settings_is_next) {
        if (GuiDropdownBox(selection_shape, sidebar_labels, &m_next_scene,
                           m_edit_mode)) {
            m_pressed = true;
            m_edit_mode ^= 1;
        }
    }

    m_return_settings = GuiButton(settings_button_shape, "#142#");
}

void SideBar::interact() {
    scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();
    bool menu_is_current = registry.get_scene() == scene::Menu;
    bool settings_is_current = registry.get_scene() == scene::Settings;

    if (!menu_is_current) {
        if (m_return_menu) {
            registry.set_scene(scene::Menu);
            m_return_menu = false;
            return;
        }
    }

    if (!menu_is_current && !settings_is_current) {
        if (m_pressed) {
            registry.set_scene(m_next_scene);
            m_pressed = false;
            return;
        }
    }

    if (m_return_settings) {
        if (settings_is_current) {
            registry.set_scene(m_scene_before_settings);
        } else {
            m_scene_before_settings = registry.get_scene();
            registry.set_scene(scene::Settings);
        }
        m_return_settings = false;
        return;
    }
}

}  // namespace component
```

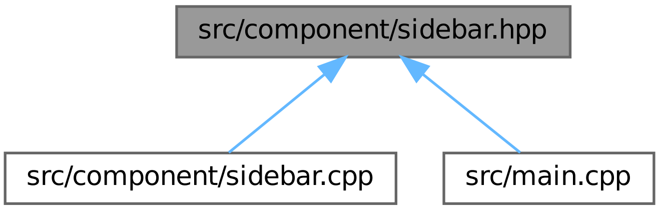
7.19 src/component/sidebar.hpp File Reference

```
#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.

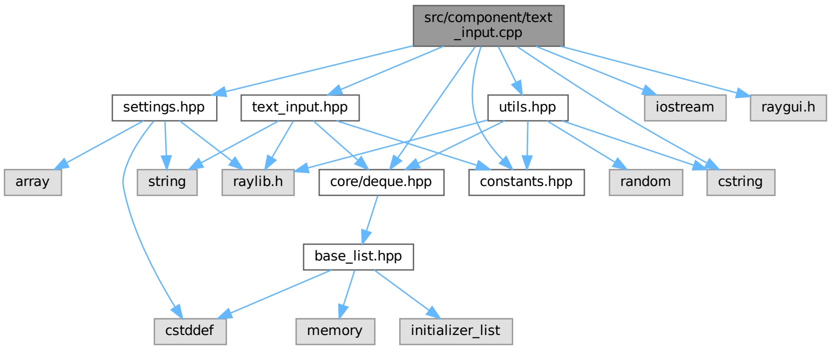
```
#ifndef COMPONENT_SIDEBAR_HPP_
#define COMPONENT_SIDEBAR_HPP_

#include <array>

#include "constants.hpp"
#include "scene/scene_registry.hpp"

namespace component {

class SideBar {
private:
    static constexpr int num_scenes = 8;

    static constexpr int button_width = constants::sidebar_width;
    static constexpr int button_height = 50;

    static constexpr const char* sidebar_labels =
        "Array;"
        "Dynamic Array;"
        "Linked List;"
        "Doubly Linked List;"
        "Circular Linked List;"
        "Stack;"
        "Queue";

    int m_next_scene{};
    bool m_edit_mode{};
    bool m_return_menu{};
    bool m_return_settings{};
    int m_scene_before_settings{};
    bool m_pressed{};

public:
    void render();
    void interact();
};

}  // namespace component

#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

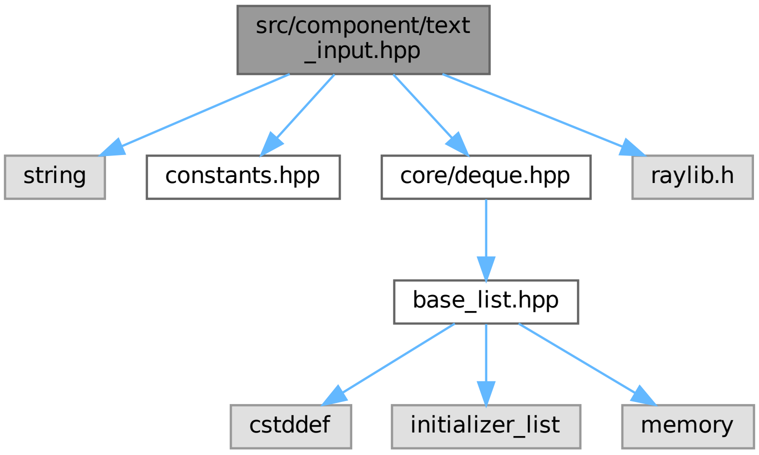
Go to the documentation of this file.

```
#include "text_input.hpp"

#include <cstring>
#include <iostream>

#include "constants.hpp"
#include "core/deque.hpp"
#include "raygui.h"
#include "settings.hpp"
#include "utils.hpp"

namespace component {

TextInput::TextInput(const char* label) : m_label{label} {}

void TextInput::set_random_min(int value) { m_random_min = value; }

void TextInput::set_random_max(int value) { m_random_max = value; }

void TextInput::render(float& options_head, float head_offset) {
    Rectangle shape{options_head, constants::scene_height - size.y, size.x,
                    size.y};

    utils::DrawText(
        m_label, {options_head, constants::scene_height - size.y - 25},
        utils::adaptive_text_color(
            Settings::get_instance().get_color(Settings::num_color - 1)),
        20, 2);

    DrawRectangleRec(shape, RAYWHITE);

    if (GuiTextBox(shape, static_cast<char*>(m_text_input),
                   constants::text_buffer_size, m_is_active)) {
        m_is_active ^= 1;
    }

    options_head += (shape.width + head_offset);

    shape = {options_head, constants::scene_height - size.y, size.y, size.y};

    m_set_random = GuiButton(shape, "#78#");

    options_head += (shape.width + head_offset);
}

bool TextInput::interact() {
    if (m_set_random) {
        auto value = utils::get_random(m_random_min, m_random_max);
        m_set_random = false;
        std::strncpy(m_text_input, std::to_string(value).c_str(),
                     constants::text_buffer_size);
        return true;
    }

    return false;
}

core::Deque<int> TextInput::extract_values() {
    core::Deque<int> nums = utils::str_extract_data(m_text_input);  // NOLINT
    return nums;
}

}  // namespace component
```

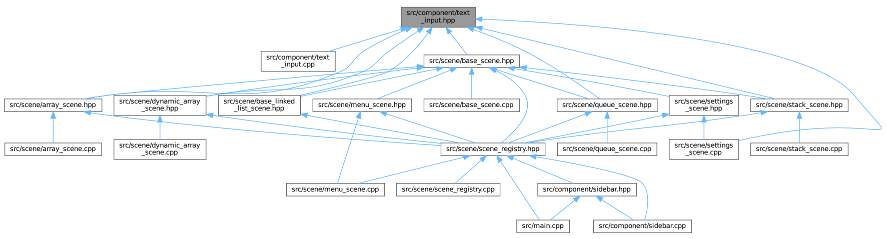
7.23 src/component/text_input.hpp File Reference

```
#include <string>
#include "constants.hpp"
#include "core/deque.hpp"
#include "raylib.h"
```

Include dependency graph for text_input.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class component::TextInput

Namespaces

- namespace component

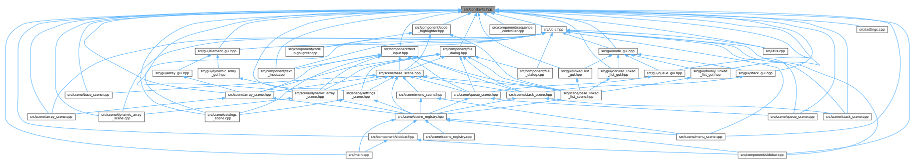
7.24 text_input.hpp

Go to the documentation of this file.

```
#ifndef COMPONENT_TEXT_INPUT_HPP_
#define COMPONENT_TEXT_INPUT_HPP_

#include <string>

#include "constants.hpp"
#include "core/deque.hpp"
#include "raylib.h"

namespace component {

class TextInput {
private:
    char m_text_input[constants::text_buffer_size] = "";  // NOLINT
    bool m_is_active{};
    const char* m_label{};

    int m_random_min{constants::min_val};
    int m_random_max{constants::max_val};
    bool m_set_random{};

public:
    static constexpr Vector2 size{200, 50};

    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);
};

}  // namespace component

#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

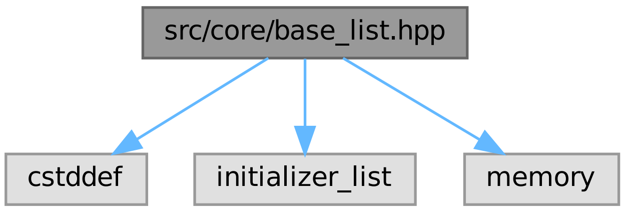
Go to the documentation of this file.

```
#ifndef CONSTANTS_HPP_
#define CONSTANTS_HPP_

namespace constants {

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";

}  // namespace constants

#endif  // CONSTANTS_HPP_
```

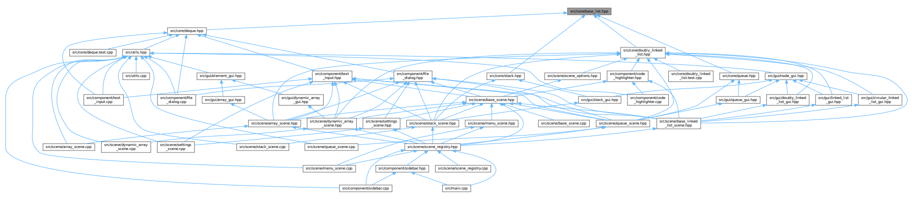
7.27 src/core/base_list.hpp File Reference

```
#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.

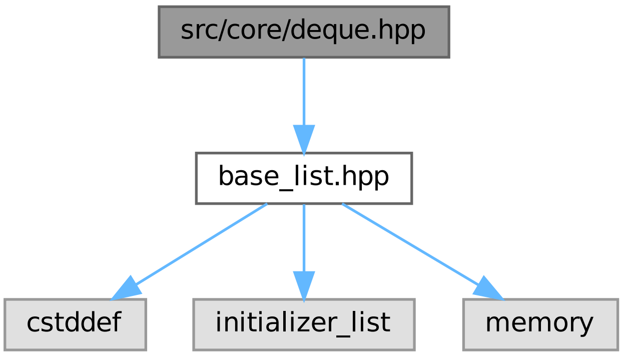
```
#ifndef CORE_BASE_LIST_HPP_
#define CORE_BASE_LIST_HPP_

#include <cstddef>
#include <initializer_list>
#include <memory>

namespace core {

template<typename T>
class BaseList {
protected:
    struct Node;
    using Node_ptr = Node*;

    struct Node {
        T data{};
        Node_ptr prev{};
        Node_ptr next{};
    };

    Node_ptr m_head{nullptr};
    Node_ptr m_tail{nullptr};
    std::size_t m_size{};

    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();

public:
    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();

    [[nodiscard]] bool empty() const;
    [[nodiscard]] std::size_t size() const;
};

template<typename T>
BaseList<T>::BaseList(const BaseList& rhs) {
    copy_data(rhs);
}

template<typename T>
BaseList<T>::BaseList(std::initializer_list<T> init_list) {
    for (const auto& elem : init_list) {
        push_back(elem);
    }
}

template<typename T>
BaseList<T>& BaseList<T>::operator=(const BaseList& rhs) {
    if (this != &rhs) {
        copy_data(rhs);
    }

    return *this;
}

template<typename T>
BaseList<T>::BaseList(BaseList&& rhs) noexcept
    : m_head{rhs.m_head}, m_tail{rhs.m_tail}, m_size{rhs.m_size} {
    rhs.m_head = nullptr;
    rhs.m_tail = nullptr;
    rhs.m_size = 0;
}

template<typename T>
BaseList<T>& BaseList<T>::operator=(BaseList&& rhs) noexcept {
    if (this != &rhs) {
        clean_up();

        m_head = rhs.m_head;
        m_tail = rhs.m_tail;
        m_size = rhs.m_size;

        rhs.m_head = nullptr;
        rhs.m_tail = nullptr;
        rhs.m_size = 0;
    }

    return *this;
}

template<typename T>
BaseList<T>::~BaseList() {
    clean_up();
}

template<typename T>
bool BaseList<T>::empty() const {
    return m_size == 0;
}

template<typename T>
std::size_t BaseList<T>::size() const {
    return m_size;
}

template<typename T>
void BaseList<T>::init_first_element(const T& elem) {
    m_head = new Node{elem, nullptr, nullptr};
    m_tail = m_head;
    m_size = 1;
}

template<typename T>
void BaseList<T>::clean_up() {
    Node_ptr ptr{nullptr};

    while (m_head != nullptr) {
        ptr = m_head->next;
        delete m_head;
        m_head = ptr;
    }

    m_tail = m_head;
    m_size = 0;
}
```

```

template<typename T>
void BaseList<T>::copy_data(const BaseList& rhs) {
    for (Node_ptr ptr = rhs.m_head; ptr != nullptr; ptr = ptr->next) {
        push_back(ptr->data);
    }
}

template<typename T>
void BaseList<T>::push_back(const T& elem) {
    if (empty()) {
        init_first_element(elem);
        return;
    }

    m_tail->next = new Node{elem, m_tail, nullptr};
    m_tail = m_tail->next;
    ++m_size;
}

template<typename T>
void BaseList<T>::push_front(const T& elem) {
    if (empty()) {
        init_first_element(elem);
        return;
    }

    m_head->prev = new Node{elem, nullptr, m_head};
    m_head = m_head->prev;
    ++m_size;
}

template<typename T>
T& BaseList<T>::back() const {
    return m_tail->data;
}

template<typename T>
T& BaseList<T>::front() const {
    return m_head->data;
}

template<typename T>
void BaseList<T>::pop_back() {
    if (size() <= 1) {
        clean_up();
        return;
    }

    m_tail = m_tail->prev;
    delete m_tail->next;
    m_tail->next = nullptr;
    --m_size;
}

template<typename T>
void BaseList<T>::pop_front() {
    if (size() <= 1) {
        clean_up();
        return;
    }

    m_head = m_head->next;
    delete m_head->prev;
    m_head->prev = nullptr;
    --m_size;
}

}  // namespace core

#endif  // CORE_BASE_LIST_HPP_
```

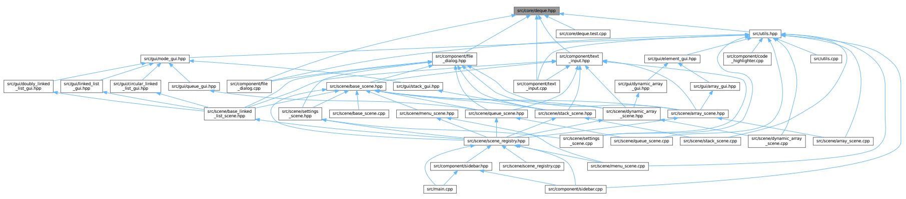
7.29 src/core/deque.hpp File Reference

`#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.

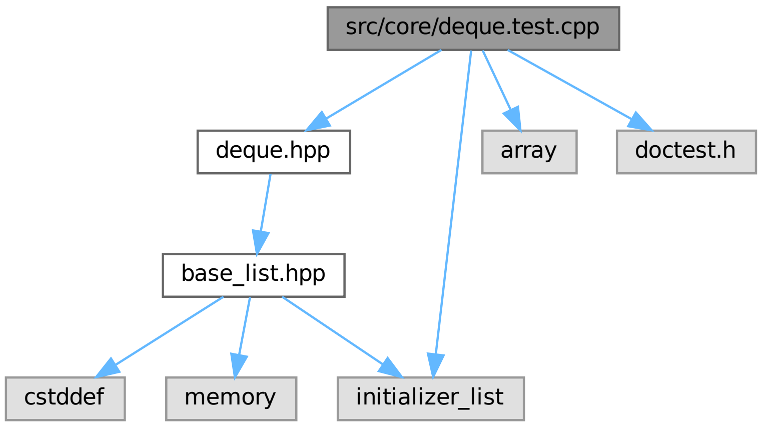
```
#ifndef CORE_DEQUE_HPP_
#define CORE_DEQUE_HPP_

#include "base_list.hpp"

namespace core {

```

```
template<typename T>
class Deque : public BaseList<T> {
private:
    using Base = BaseList<T>;

public:
    using Base::Base;

    using Base::empty;
    using Base::size;

    using Base::push_back;
    using Base::push_front;

    using Base::back;
    using Base::front;

    using Base::pop_back;
    using Base::pop_front;
};

}  // namespace core

#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}

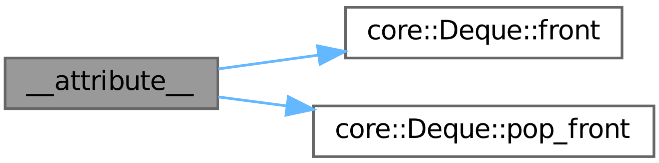
7.31.1 Function Documentation

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:

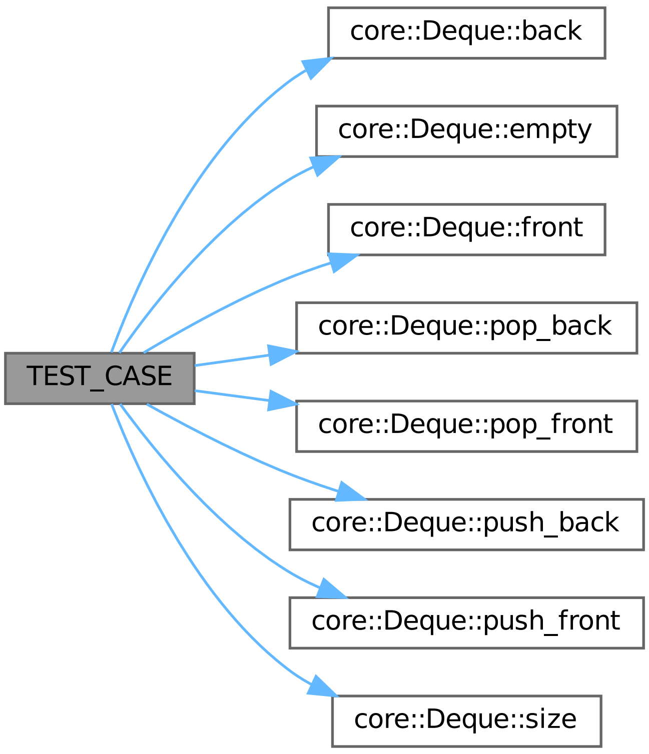


7.31.1.2 TEST_CASE() [1/2]

```
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

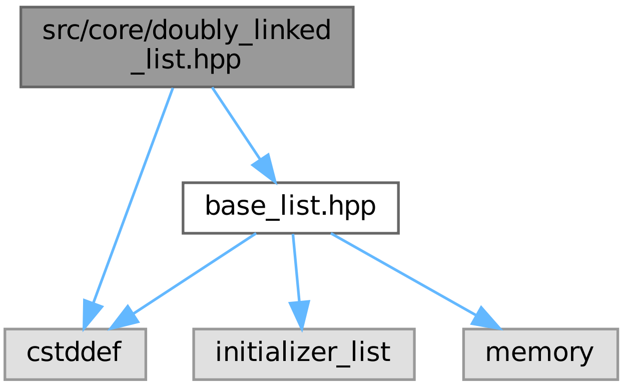
Go to the documentation of this file.

```cpp
#include "deque.hpp"

#include <array>
#include <initializer_list>

#include "doctest.h"

TEST_CASE("core::Deque functionality") {
    core::Deque<int> deque;
    CHECK(deque.empty());

    deque.push_back(2);
    deque.push_back(3);
    deque.push_front(1);

    CHECK(deque.front() == 1);
    CHECK(deque.back() == 3);
    CHECK(deque.size() == 3);

    deque.pop_back();
    CHECK(deque.back() == 2);
    CHECK(deque.size() == 2);

    deque.pop_front();
    CHECK(deque.front() == 2);
    CHECK(deque.size() == 1);

    deque.front() += 3;
    CHECK(deque.front() == 5);

    deque.push_back(0);
    deque.back() -= 2;
    CHECK(deque.back() == -2);
}

constexpr std::array<int, 3> list{1, 2, 3};

inline __attribute__((always_inline)) void check_match(core::Deque<int> deque) {
    for (int elem : list) {
        CHECK(deque.front() == elem);
        deque.pop_front();
    }
}

TEST_CASE("core::Deque special member functions") {
    std::initializer_list<int> init_list{1, 2, 3};

    SUBCASE("core::Deque(std::initializer_list<T>)") {
        core::Deque<int> deque{init_list};
        check_match(deque);
    }

    SUBCASE("core::Deque(const core::Deque&)") {
        core::Deque<int> deque1{init_list};
        core::Deque<int> deque2{deque1};  // NOLINT

        check_match(deque2);
        check_match(deque1);
    }

    SUBCASE("core::Deque& operator=(const core::Deque&) (single)") {
        core::Deque<int> deque1{init_list};
        core::Deque<int> deque2 = deque1;  // NOLINT

        check_match(deque2);
        check_match(deque1);
    }

    SUBCASE("core::Deque& operator=(const core::Deque&) (multiple)") {
        core::Deque<int> deque1{init_list};
        core::Deque<int> deque2;
        core::Deque<int> deque3;
        deque3 = deque2 = deque1;

        check_match(deque3);
        check_match(deque2);
        check_match(deque1);
    }

    SUBCASE("core::Deque(core::Deque&& rhs)") {
        {
            core::Deque<int> deque1{core::Deque<int>{init_list}};
            check_match(deque1);
        }
        {
            core::Deque<int> deque1{init_list};
            core::Deque<int> deque2{std::move(deque1)};
            check_match(deque2);
            CHECK(deque1.empty());  // NOLINT
        }
    }

    SUBCASE("core::Deque& operator=(core::Deque&& rhs)") {
        {
            core::Deque<int> deque1{1, 2, 3};
            core::Deque<int> deque2 = std::move(deque1);

            check_match(deque2);
            CHECK(deque1.empty());  // NOLINT
        }
        {
            core::Deque<int> deque{init_list};
            deque = std::move(deque);
            check_match(deque);  // NOLINT
        }
    }
}
```

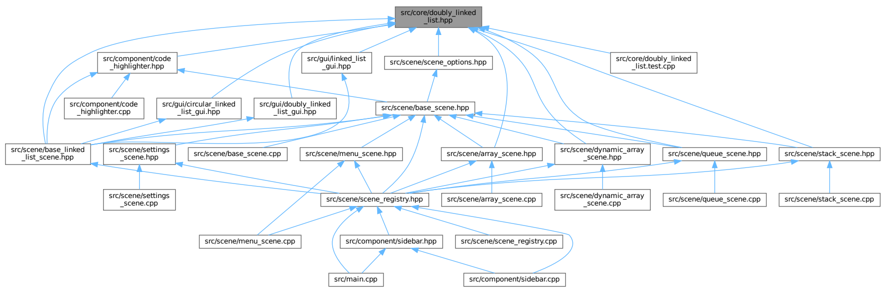
7.33 src/core/doubly_linked_list.hpp File Reference

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

Include dependency graph for doubly_linked_list.hpp:



This graph shows which files directly or indirectly include this file:


src/core/doubly_linked_list.hpp
src/component/code_highlighter.hpp
src/gui/linked_list_gui.hpp
src/scene/scene_options.hpp
src/core/doubly_linked_list.test.cpp
src/component/code_highlighter.cpp
src/gui/circular_linked_list_gui.hpp
src/gui/doubly_linked_list_gui.hpp
src/scene/base_scene.hpp
src/scene/base_linked_list_scene.hpp
src/scene/settings_scene.hpp
src/scene/base_scene.cpp
src/scene/menu_scene.hpp
src/scene/array_scene.hpp
src/scene/dynamic_array_scene.hpp
src/scene/queue_scene.hpp
src/scene/stack_scene.hpp
src/scene/settings_scene.cpp
src/scene/scene_registry.hpp
src/scene/array_scene.cpp
src/scene/dynamic_array_scene.cpp
src/scene/queue_scene.cpp
src/scene/stack_scene.cpp
src/scene/menu_scene.cpp
src/component/sidebar.hpp
src/scene/scene_registry.cpp
src/main.cpp
src/component/sidebar.cpp

Classes

- class core::DoublyLinkedList< T >

Namespaces

- namespace core

7.34 doubly_linked_list.hpp

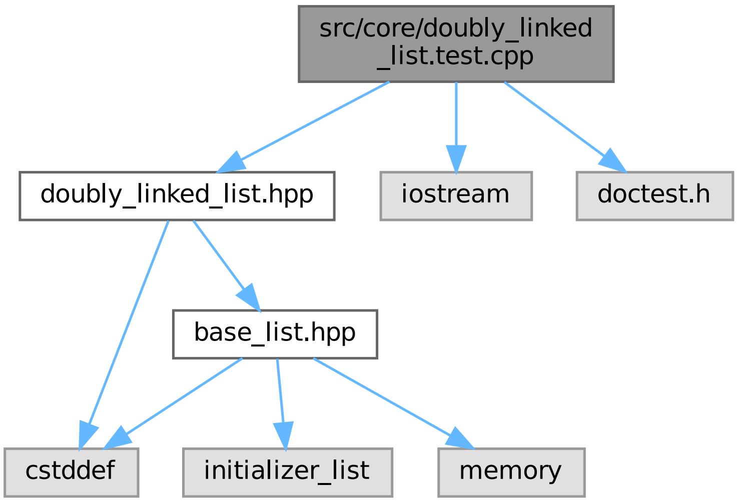
Go to the documentation of this file.

```
#ifndef CORE_DOUBLY_LINKED_LIST_HPP_
#define CORE_DOUBLY_LINKED_LIST_HPP_

#include <cstddef>

#include "base_list.hpp"

namespace core {

template<typename T>
class DoublyLinkedList : public BaseList<T> {
protected:
    using Base = BaseList<T>;
    using Node = typename Base::Node;
    using Node_ptr = Node*;
    using cNode_ptr = const Node*;

    using Base::m_head;
    using Base::m_size;
    using Base::m_tail;

    Node_ptr internal_search(const T& elem);
    Node_ptr internal_find(std::size_t index);

public:
    using Base::Base;

    using Base::empty;
    using Base::size;

    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();
};

template<typename T>
typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::internal_search(
    const T& elem) {
    Node_ptr ptr{m_head};

    while (ptr != nullptr) {
        if (ptr->data == elem) {
            break;
        }

        ptr = ptr->next;
    }

    return ptr;
}

template<typename T>
typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::internal_find(
    std::size_t index) {
    Node_ptr ptr{m_head};
    std::size_t pos = 0;

    while (ptr != nullptr && pos < index) {
        ptr = ptr->next;
        ++pos;
    }

    return ptr;
}

template<typename T>
typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::search(
    const T& elem) {
    return internal_search(elem);
}

template<typename T>
typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::find(
    std::size_t index) {
    return internal_find(index);
}

template<typename T>
typename DoublyLinkedList<T>::cNode_ptr DoublyLinkedList<T>::search(
    const T& elem) const {
    return internal_search(elem);
}

template<typename T>
typename DoublyLinkedList<T>::cNode_ptr DoublyLinkedList<T>::find(
    std::size_t index) const {
    return internal_find(index);
}

template<typename T>
typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::insert(
    std::size_t index, const T& elem) {
    if (index == 0) {
        Base::push_front(elem);
        return m_head;
    }

    if (index >= m_size) {
        Base::push_back(elem);
        return m_tail;
    }

    Node_ptr ptr = find(index);
    auto new_node = new Node{elem, ptr->prev, ptr};

    ptr->prev->next = new_node;
    ptr->prev = new_node;
    ++m_size;

    return new_node;
}

template<typename T>
typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::remove(
    std::size_t index) {
    if (index >= m_size) {
        return nullptr;
    }

    if (index == 0) {
        Base::pop_front();
        return m_head;
    }

    if (index + 1 == m_size) {
        Base::pop_back();
        return nullptr;
    }

    Node_ptr ptr = find(index);
    Node_ptr ret = ptr->next;

    ptr->next->prev = ptr->prev;
    ptr->prev->next = ptr->next;

    delete ptr;
    --m_size;

    return ret;
}

template<typename T>
T& DoublyLinkedList<T>::at(std::size_t index) {
    return find(index)->data;
}

template<typename T>
T DoublyLinkedList<T>::at(std::size_t index) const {
    return find(index)->data;
}

template<typename T>
void DoublyLinkedList<T>::clear() {
    while (!empty()) {
        Base::pop_front();
    }
}

}  // namespace core

#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")

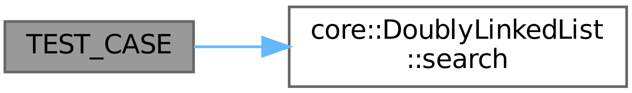
7.35.1 Function Documentation

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

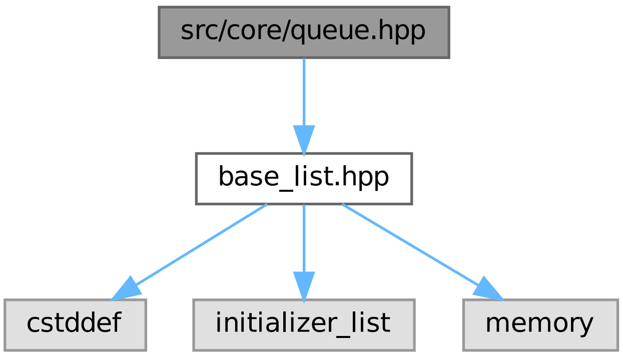
Go to the documentation of this file.

```
#include "doubly_linked_list.hpp"

#include <iostream>

#include "doctest.h"

TEST_CASE("core::DoublyLinkedList functionality") {
    // List: {1, 2, 3}
    SUBCASE("Node_ptr search(const T& elem)") {
        core::DoublyLinkedList<int> dll{1, 2, 3};
        CHECK(dll.search(4) == nullptr);
        CHECK(dll.search(3)->data == 3);
    }

    // List: {1, 2, 3}
    SUBCASE("Node_ptr find(std::size_t index)") {
        core::DoublyLinkedList<int> dll{1, 2, 3};
        CHECK(dll.find(8) == nullptr);

        auto* ptr1 = dll.search(3);
        auto* ptr2 = dll.find(1);

        CHECK(ptr1->data == 3);
        CHECK(ptr2->data == 2);

        CHECK(ptr1->prev == ptr2);
        CHECK(ptr2->next == ptr1);
    }

    SUBCASE("Node_ptr insert(std::size_t index, const T& elem)") {
        core::DoublyLinkedList<int> dll{1, 2, 3};
        auto* ptr0 = dll.search(1);

        // List: {-1, 1, 2, 3}
        auto* ptr = dll.insert(0, -1);

        CHECK(dll.size() == 4);
        CHECK(ptr->next == ptr0);

        auto* ptrN = dll.search(3);
        // List: {-1, 1, 2, 3, 4}
        ptr = dll.insert(4, 4);

        CHECK(dll.size() == 5);
        CHECK(ptr->prev == ptrN);

        // List: {-1, 1, 20, 2, 3, 4}
        ptr = dll.insert(2, 20);  // NOLINT
        CHECK(ptr->prev == dll.find(1));
        CHECK(ptr->next == dll.find(3));
        CHECK(dll.size() == 6);

        // List: {-1, 1, 20, 2, 3, 4, 69}
        dll.insert(69, 69);  // NOLINT
        CHECK(dll.search(69) == dll.find(6));
        CHECK(dll.size() == 7);
    }

    // List: {-1, 1, 20, 2, 3, 4, 69}
    SUBCASE("Node_ptr remove(std::size_t index)") {
        core::DoublyLinkedList<int> dll{-1, 1, 20, 2, 3, 4, 69};  // NOLINT

        CHECK(dll.remove(1000) == nullptr);
        CHECK(dll.size() == 7);

        // List: {-1, 1, 20, 2, 3, 4}
        CHECK(dll.remove(6) == nullptr);
        CHECK(dll.size() == 6);

        // List: {1, 20, 2, 3, 4}
        auto* ptr = dll.remove(0);
        CHECK(dll.size() == 5);
        CHECK(ptr->data == 1);

        // List: {1, 2, 3, 4}
        ptr = dll.remove(1);
        CHECK(dll.size() == 4);
        CHECK(ptr->data == 2);
    }
}
```

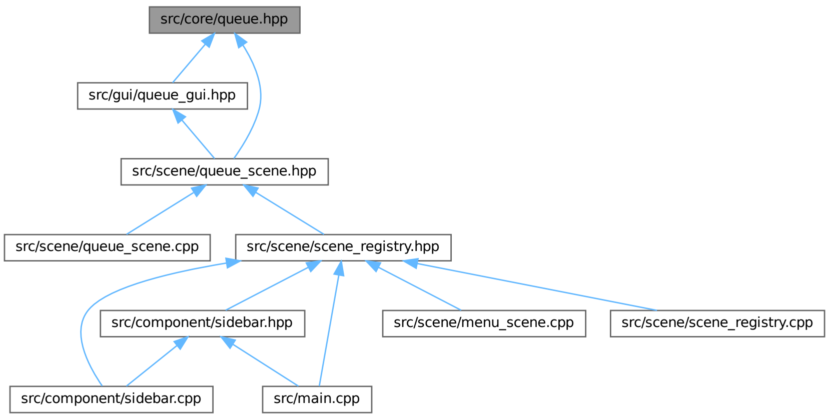
7.37 src/core/queue.hpp File Reference

```
#include "base_list.hpp"
```

Include dependency graph for queue.hpp:



This graph shows which files directly or indirectly include this file:


src/core/queue.hpp
src/gui/queue_gui.hpp
src/scene/queue_scene.hpp
src/scene/queue_scene.cpp
src/scene/scene_registry.hpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class core::Queue< T >

Namespaces

- namespace core

7.38 queue.hpp

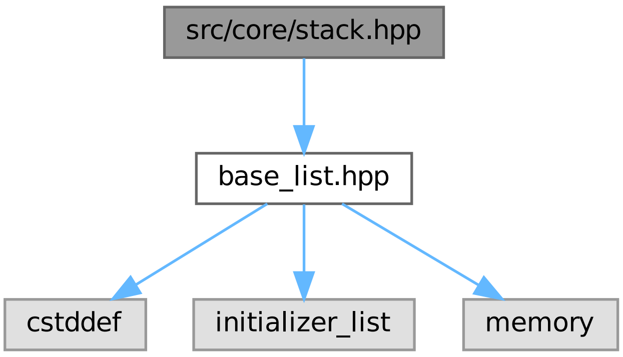
Go to the documentation of this file.

```
#ifndef CORE_QUEUE_HPP_
#define CORE_QUEUE_HPP_

#include "base_list.hpp"

namespace core {

template<typename T>
class Queue : public BaseList<T> {
private:
    using Base = BaseList<T>;

public:
    using Base::Base;

    using Base::empty;
    using Base::size;

    // for animation purpose only, not for real use
    using Base::pop_back;
    using Base::push_front;

    T& front() const;
    T& back() const;

    void push(const T& elem);
    void pop();
};

template<typename T>
T& Queue<T>::front() const {
    return Base::front();
}

template<typename T>
T& Queue<T>::back() const {
    return Base::back();
}

template<typename T>
void Queue<T>::push(const T& elem) {
    Base::push_back(elem);
}

template<typename T>
void Queue<T>::pop() {
    Base::pop_front();
}

}  // namespace core

#endif  // CORE_QUEUE_HPP_
```

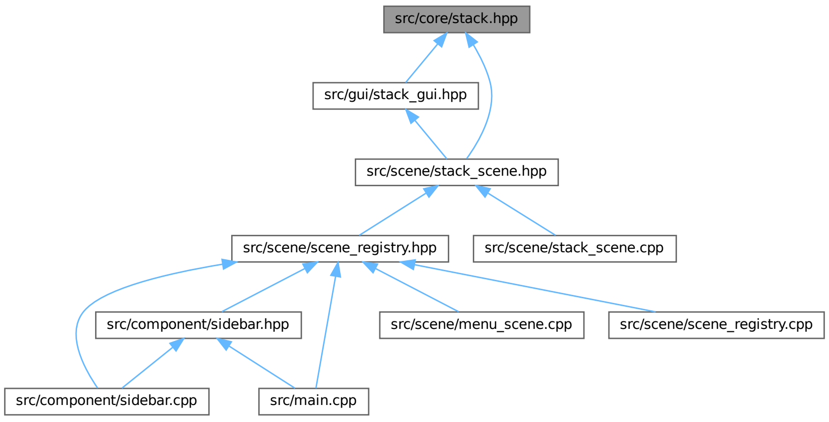
7.39 src/core/stack.hpp File Reference

```
#include "base_list.hpp"
```

Include dependency graph for stack.hpp:



This graph shows which files directly or indirectly include this file:


src/core/stack.hpp
src/gui/stack_gui.hpp
src/scene/stack_scene.hpp
src/scene/scene_registry.hpp
src/scene/stack_scene.cpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class core::Stack< T >

Namespaces

- namespace core

7.40 stack.hpp

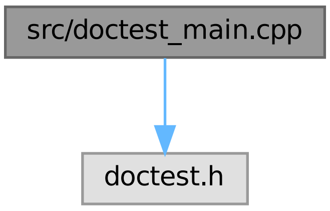
Go to the documentation of this file.

```
#ifndef CORE_STACK_HPP_
#define CORE_STACK_HPP_

#include "base_list.hpp"

namespace core {

template<typename T>
class Stack : public BaseList<T> {
protected:
    using Base = BaseList<T>;
    using Base::m_head;
    using Base::m_tail;

public:
    using Base::Base;

    using Base::empty;
    using Base::size;

    T& top() const;

    void push(const T& elem);
    void pop();
};

template<typename T>
T& Stack<T>::top() const {
    return Base::front();
}

template<typename T>
void Stack<T>::push(const T& elem) {
    Base::push_front(elem);
}

template<typename T>
void Stack<T>::pop() {
    Base::pop_front();
}

}  // namespace core

#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

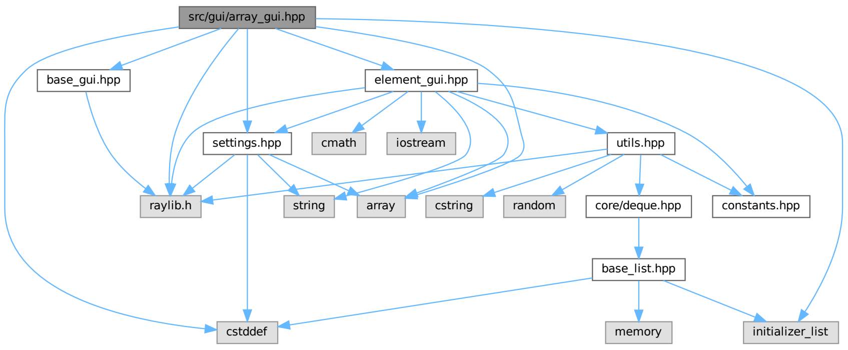
Go to the documentation of this file.

```
00001 #define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN
00002 #include "doctest.h"
```

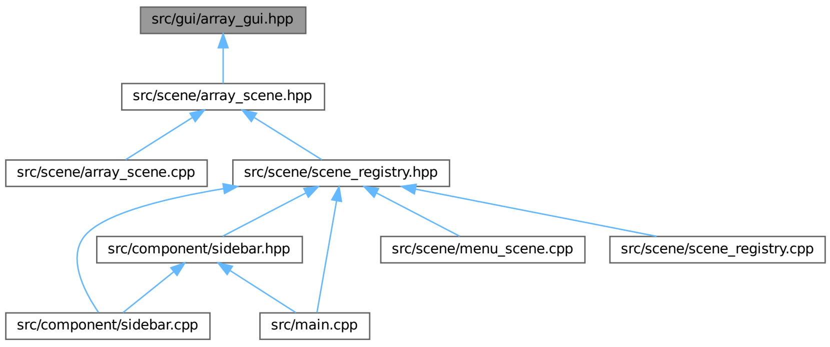
7.43 src/gui/array_gui.hpp File Reference

```
#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:


src/gui/array_gui.hpp
src/scene/array_scene.hpp
src/scene/array_scene.cpp
src/scene/scene_registry.hpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class gui::GuiArray< T, N >

Namespaces

- namespace gui

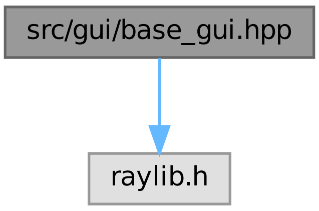
7.44 array_gui.hpp

Go to the documentation of this file.
```
#ifndef GUI_ARRAY_GUI_HPP_
#define GUI_ARRAY_GUI_HPP_

#include <array>
#include <cstddef>
#include <initializer_list>

#include "base_gui.hpp"
#include "element_gui.hpp"
#include "raylib.h"
#include "settings.hpp"

namespace gui {

template<typename T, std::size_t N>
class GuiArray : public internal::Base {
private:
    static constexpr Vector2 head_pos{
        constants::scene_width / 2.0F - 15 * GuiElement<T>::side,
        constants::scene_height / 2.0F};

    std::array<GuiElement<T>, N> m_array{};

    void render_link(Vector2 src, Vector2 dest) override;

public:
    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);
};

template<typename T, std::size_t N>
GuiArray<T, N>::GuiArray() {
    for (std::size_t i = 0; i < N; ++i) {
        m_array[i] = GuiElement<T>{0, i};
        m_array[i].set_color_index(0);
    }
}

template<typename T, std::size_t N>
GuiArray<T, N>::GuiArray(std::array<GuiElement<T>, N>&& init_list)
    : m_array{init_list} {}

template<typename T, std::size_t N>
void GuiArray<T, N>::render_link(Vector2 src, Vector2 dest) {}

template<typename T, std::size_t N>
void GuiArray<T, N>::render() {
    update();

    for (std::size_t i = 0; i < N; ++i) {
        m_array[i].render();
    }
}

template<typename T, std::size_t N>
void GuiArray<T, N>::update() {
    // TODO: if not outdated then return

    for (std::size_t i = 0; i < N; ++i) {
        m_array[i].set_pos(
            {head_pos.x + 4 * GuiElement<T>::side * i, head_pos.y});
    }
}

template<typename T, std::size_t N>
T& GuiArray<T, N>::operator[](std::size_t idx) {
    return m_array[idx].get_value();
}

template<typename T, std::size_t N>
T GuiArray<T, N>::operator[](std::size_t idx) const {
    return m_array[idx].get_value();
}

template<typename T, std::size_t N>
void GuiArray<T, N>::set_color_index(std::size_t idx, int color_index) {
    m_array[idx].set_color_index(color_index);
}

}  // namespace gui

#endif  // GUI_ARRAY_GUI_HPP_
```

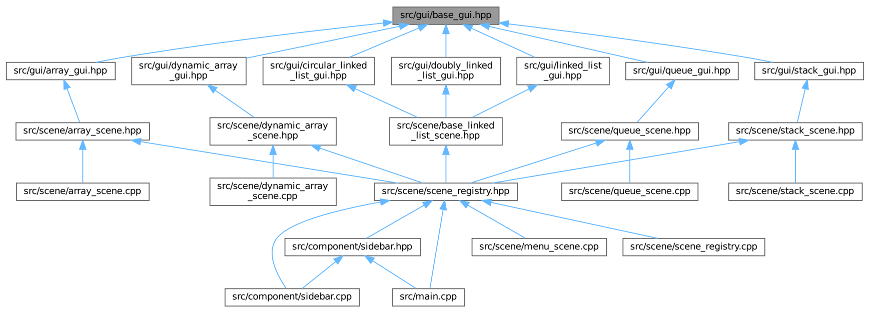
7.45 src/gui/base_gui.hpp File Reference

`#include "raylib.h"`

Include dependency graph for base_gui.hpp:



This graph shows which files directly or indirectly include this file:


src/gui/base_gui.hpp
src/gui/array_gui.hpp
src/gui/dynamic_array_gui.hpp
src/gui/circular_linked_list_gui.hpp
src/gui/doubly_linked_list_gui.hpp
src/gui/linked_list_gui.hpp
src/gui/queue_gui.hpp
src/gui/stack_gui.hpp
src/scene/array_scene.hpp
src/scene/dynamic_array_scene.hpp
src/scene/base_linked_list_scene.hpp
src/scene/queue_scene.hpp
src/scene/stack_scene.hpp
src/scene/array_scene.cpp
src/scene/dynamic_array_scene.cpp
src/scene/scene_registry.hpp
src/scene/queue_scene.cpp
src/scene/stack_scene.cpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class gui::internal::Base

Namespaces

- namespace gui
- namespace gui::internal

7.46 base_gui.hpp

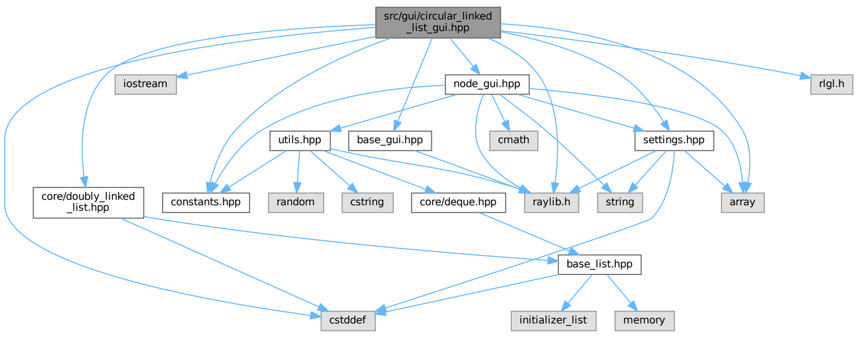
Go to the documentation of this file.

```
#ifndef GUI_BASE_GUI_HPP_
#define GUI_BASE_GUI_HPP_

#include "raylib.h"

namespace gui::internal {

class Base {
    virtual void render_link(Vector2 src, Vector2 dest) = 0;

public:
    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;
};

}  // namespace gui::internal

#endif  // GUI_BASE_GUI_HPP_
```

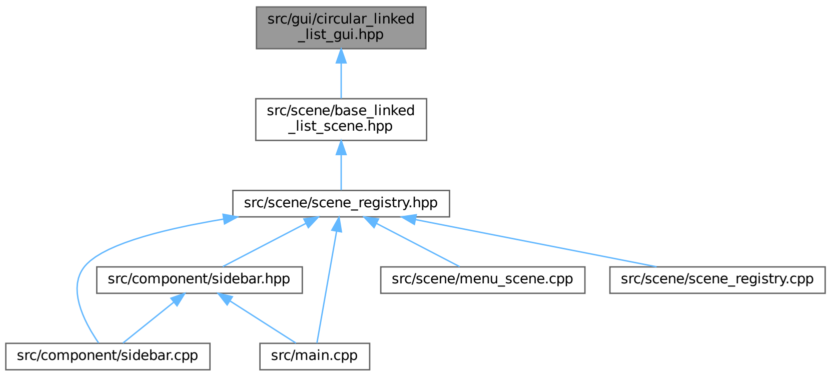
7.47 src/gui/circular_linked_list_gui.hpp File Reference

```
#include <array>
#include <cstddef>
#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:


src/gui/circular_linked_list_gui.hpp
src/scene/base_linked_list_scene.hpp
src/scene/scene_registry.hpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class gui::GuiCircularLinkedList< T >

Namespaces

- namespace gui

7.48 circular_linked_list_gui.hpp

Go to the documentation of this file.

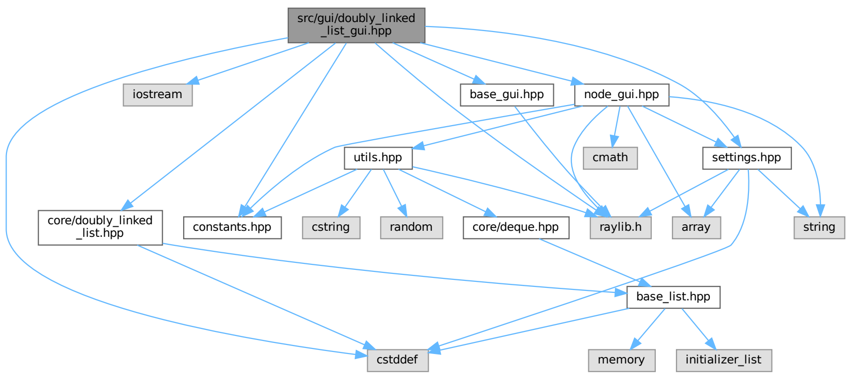
```
#ifndef GUI_CIRCULAR_LINKED_LIST_GUI_HPP_
#define GUI_CIRCULAR_LINKED_LIST_GUI_HPP_

#include <array>
#include <cstddef>
#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"

namespace gui {

template<typename T>
class GuiCircularLinkedList : public core::DoublyLinkedList<GuiNode<T>>,
                              public internal::Base {
private:
    using Base = core::DoublyLinkedList<GuiNode<T>>;

    static constexpr Vector2 head_pos{
        constants::scene_width / 2.0F - 15 * GuiNode<T>::radius,
        constants::scene_height / 2.0F};

    using Base::m_head;
    using Base::m_tail;

    void render_link(Vector2 src, Vector2 dest) override;
    void render_back_link();

public:
    using Base::Base;

    using Base::empty;
    using Base::size;

    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();
};

template<typename T>
void GuiCircularLinkedList<T>::init_label() {
    if (m_head != nullptr) {
        m_head->data.set_label("head");
    }

    if (m_tail != nullptr) {
        if (m_head == m_tail) {
            m_tail->data.set_label("head/tail");
        } else {
            m_tail->data.set_label("tail");
        }
    }
}

template<typename T>
GuiCircularLinkedList<T>::GuiCircularLinkedList(
    std::initializer_list<GuiNode<T>> init_list)
    : core::DoublyLinkedList<GuiNode<T>>(init_list) {
    init_label();
}

template<typename T>
void GuiCircularLinkedList<T>::insert(std::size_t index, const T& elem) {
    Base::insert(index, GuiNode{elem});
}

template<typename T>
void GuiCircularLinkedList<T>::render_link(Vector2 src, Vector2 dest) {
    constexpr int radius = GuiNode<T>::radius;
    constexpr float scaled_len = radius / 8.0F;

    // straight line
    Vector2 link_pos{src.x + radius, src.y - scaled_len};
    Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};

    // arrow
    constexpr int arrow_size = scaled_len * 5;
    Vector2 head{dest.x - radius + scaled_len / 2, src.y};
    Vector2 side_top{head.x - arrow_size, head.y - arrow_size};
    Vector2 side_bot{head.x - arrow_size, head.y + arrow_size};

    // draw both
    const Settings& settings = Settings::get_instance();
    DrawRectangleV(link_pos, link_size, settings.get_color(1));
    DrawTriangle(head, side_top, side_bot, settings.get_color(1));
}

template<typename T>
void GuiCircularLinkedList<T>::render_back_link() {
    if (m_head == nullptr && m_tail == nullptr) {
        return;
    }

    constexpr int num_points = 5;
    const Vector2 head_pos = m_head->data.get_pos();
    const Vector2 tail_pos = m_tail->data.get_pos();
    constexpr int radius = GuiNode<T>::radius;
    constexpr float scaled_len = radius / 8.0F;

    std::array<Vector2, num_points> points{{
        tail_pos,
        {tail_pos.x + 2 * radius, tail_pos.y},
        {tail_pos.x + 2 * radius, tail_pos.y + 3 * radius},
        {head_pos.x, tail_pos.y + 3 * radius},
        head_pos,
    }};

    constexpr int arrow_size = scaled_len * 5;
    Vector2 head{head_pos.x, head_pos.y + radius - scaled_len / 2};
    Vector2 side_left{head.x - arrow_size, head.y + arrow_size};
    Vector2 side_right{head.x + arrow_size, head.y + arrow_size};

    const Settings& settings = Settings::get_instance();
    rlSetLineWidth(2 * scaled_len);
    DrawLineStrip(points.data(), num_points, settings.get_color(1));
    DrawTriangle(head, side_left, side_right, settings.get_color(1));
}

template<typename T>
void GuiCircularLinkedList<T>::render() {
    update();

    render_back_link();
    for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
        if (ptr->next != nullptr) {
            render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
        }

        ptr->data.render();
    }
}

template<typename T>
void GuiCircularLinkedList<T>::update() {
    // TODO: if not outdated then return

    std::size_t pos = 0;

    for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
        ptr->data.set_pos(
            {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
        ++pos;
    }
}

}  // namespace gui

#endif  // GUI_CIRCULAR_LINKED_LIST_GUI_HPP_
```

7.49 src/gui/doubly_linked_list_gui.hpp File Reference

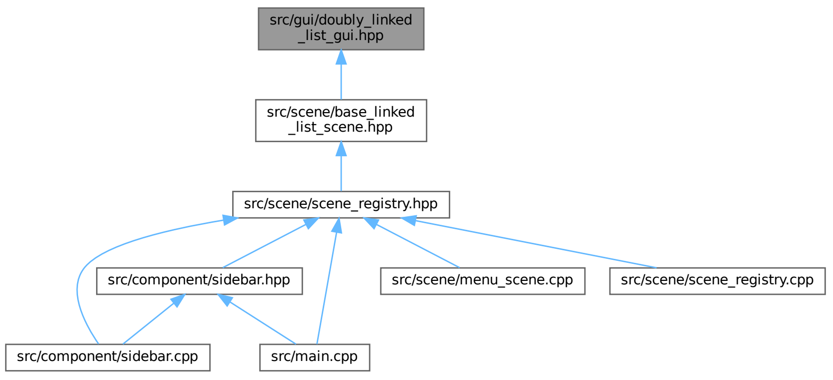
```
#include <cstddef>
#include <iostream>
```

```
#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:


src/gui/doubly_linked_list_gui.hpp
src/scene/base_linked_list_scene.hpp
src/scene/scene_registry.hpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class gui::GuiDoublyLinkedList< T >

Namespaces

- namespace gui

7.50 doubly_linked_list_gui.hpp

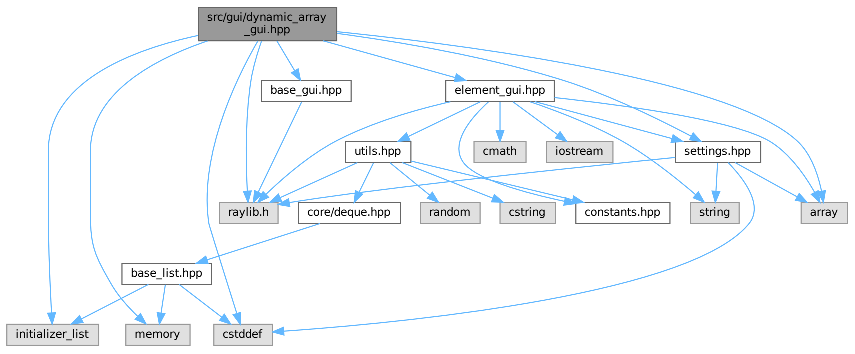
Go to the documentation of this file.

```cpp
#ifndef GUI_DOUBLY_LINKED_LIST_GUI_HPP_
#define GUI_DOUBLY_LINKED_LIST_GUI_HPP_

#include <cstddef>
#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"

namespace gui {

template<typename T>
class GuiDoublyLinkedList : public core::DoublyLinkedList<GuiNode<T>>,
                            public internal::Base {
private:
    using Base = core::DoublyLinkedList<GuiNode<T>>;

    static constexpr Vector2 head_pos{
        constants::scene_width / 2.0F - 15 * GuiNode<T>::radius,
        constants::scene_height / 2.0F};

    using Base::m_head;
    using Base::m_tail;

    void render_link(Vector2 src, Vector2 dest) override;

public:
    using Base::Base;

    using Base::empty;
    using Base::size;

    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();
};

template<typename T>
void GuiDoublyLinkedList<T>::init_label() {
    if (m_head != nullptr) {
        m_head->data.set_label("head");
    }

    if (m_tail != nullptr) {
        if (m_head == m_tail) {
            m_tail->data.set_label("head/tail");
        } else {
            m_tail->data.set_label("tail");
        }
    }
}

template<typename T>
GuiDoublyLinkedList<T>::GuiDoublyLinkedList(
    std::initializer_list<GuiNode<T>> init_list)
    : core::DoublyLinkedList<GuiNode<T>>(init_list) {
    init_label();
}

template<typename T>
void GuiDoublyLinkedList<T>::insert(std::size_t index, const T& elem) {
    Base::insert(index, GuiNode{elem});
}

template<typename T>
void GuiDoublyLinkedList<T>::render_link(Vector2 src, Vector2 dest) {
    constexpr int radius = GuiNode<T>::radius;
    constexpr float scaled_len = radius / 8.0F;

    // straight line
    Vector2 link_pos{src.x + radius, src.y - scaled_len};
    Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};

    // right arrow
    constexpr int arrow_size = scaled_len * 5;
    Vector2 right_head{dest.x - radius + scaled_len / 2, src.y};
    Vector2 right_side_top{right_head.x - arrow_size,
                           right_head.y - arrow_size};
    Vector2 right_side_bot{right_head.x - arrow_size,
                           right_head.y + arrow_size};

    // left arrow
    Vector2 left_head{src.x + radius - scaled_len / 2, src.y};
    Vector2 left_side_top{left_head.x + arrow_size, left_head.y - arrow_size};
    Vector2 left_side_bot{left_head.x + arrow_size, left_head.y + arrow_size};

    // draw all
    const Settings& settings = Settings::get_instance();
    DrawRectangleV(link_pos, link_size, settings.get_color(1));
    DrawTriangle(right_head, right_side_top, right_side_bot,
                 settings.get_color(1));
    DrawTriangle(left_head, left_side_bot, left_side_top,
                 settings.get_color(1));
}

template<typename T>
void GuiDoublyLinkedList<T>::render() {
    update();

    for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
        if (ptr->next != nullptr) {
            render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
        }

        ptr->data.render();
    }
}

template<typename T>
void GuiDoublyLinkedList<T>::update() {
    // TODO: if not outdated then return

    std::size_t pos = 0;

    for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
        ptr->data.set_pos(
            {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
        ++pos;
    }
}

}  // namespace gui

#endif  // GUI_DOUBLY_LINKED_LIST_GUI_HPP_
```

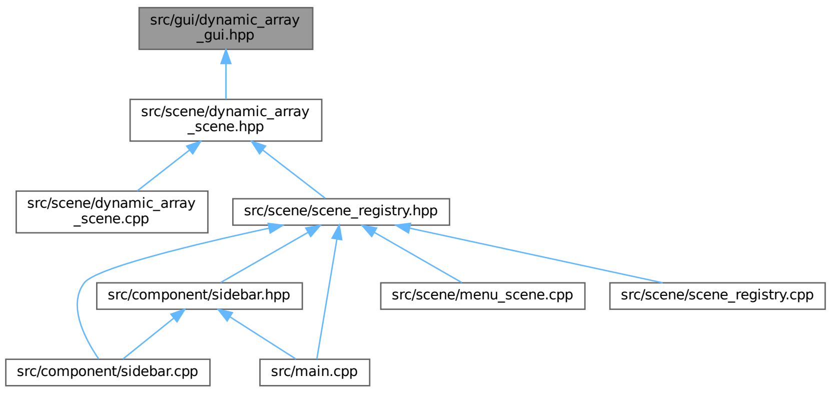
7.51 src/gui/dynamic_array_gui.hpp File Reference

```
#include <array>
#include <cstddef>
#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:


src/gui/dynamic_array_gui.hpp
src/scene/dynamic_array_scene.hpp
src/scene/dynamic_array_scene.cpp
src/scene/scene_registry.hpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class gui::GuiDynamicArray< T >

Namespaces

- namespace gui

7.52 dynamic_array_gui.hpp

Go to the documentation of this file.

```
#ifndef GUI_DYNAMIC_ARRAY_GUI_HPP_
#define GUI_DYNAMIC_ARRAY_GUI_HPP_

#include <array>
#include <cstddef>
#include <initializer_list>
#include <memory>
```

```

#include "base_gui.hpp"
#include "element_gui.hpp"
#include "raylib.h"
#include "settings.hpp"

namespace gui {

template<typename T>
class GuiDynamicArray : public internal::Base {
private:
    static constexpr Vector2 head_pos{
        constants::scene_width / 2.0F - 15 * GuiElement<T>::side,
        constants::scene_height / 2.0F};

    std::size_t m_capacity{2};
    std::size_t m_size{};
    GuiElement<T>* m_ptr{nullptr};

    void render_link(Vector2 src, Vector2 dest) override;

public:
    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();
};

template<typename T>
void GuiDynamicArray<T>::realloc(std::size_t capacity) {
    if (m_capacity > capacity) {
        return;
    }

    while (m_capacity < capacity) {
        m_capacity *= 2;
    }

    auto* new_ptr = new GuiElement<T>[m_capacity];
    for (auto i = 0; i < m_size; ++i) {
        new_ptr[i] = m_ptr[i];
    }
    for (auto i = m_size; i < m_capacity; ++i) {
        new_ptr[i].set_index(i);
    }

    delete[] m_ptr;
    m_ptr = new_ptr;
}

template<typename T>
GuiDynamicArray<T>::GuiDynamicArray() : m_ptr{new GuiElement<T>[m_capacity]} {
    for (auto i = 0; i < m_capacity; ++i) {
        m_ptr[i].set_index(i);
    }
}

template<typename T>
GuiDynamicArray<T>::GuiDynamicArray(std::initializer_list<T> init_list)
    : m_size{init_list.size()}, m_ptr{new GuiElement<T>[m_capacity]} {
    realloc(m_size);

    for (std::size_t idx = 0; auto elem : init_list) {
        *(m_ptr + idx).set_value(elem);
        *(m_ptr + idx).set_color(Settings::get_instance().get_color(0));
    }
}

template<typename T>
```

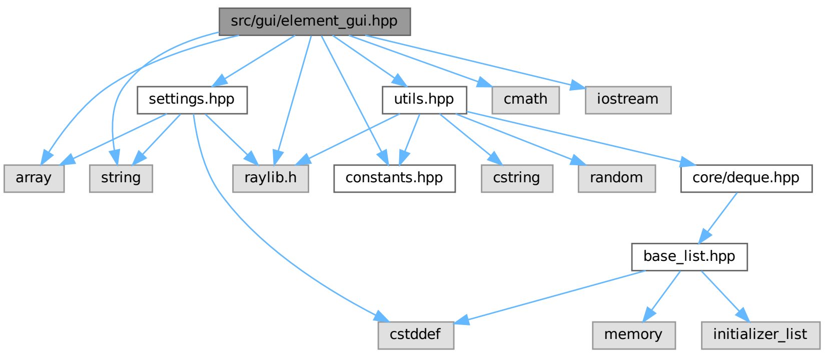
```
GuiDynamicArray<T>::GuiDynamicArray(const GuiDynamicArray<T>& other)
    : m_capacity{other.m_capacity},
      m_size{other.m_size},
      m_ptr{new GuiElement<T>[m_capacity]} {
    for (auto i = 0; i < m_capacity; ++i) {
        m_ptr[i] = other.m_ptr[i];
    }
}

template<typename T>
GuiDynamicArray<T>::GuiDynamicArray(GuiDynamicArray<T>&& other) noexcept
    : m_capacity{other.m_capacity}, m_size{other.m_size}, m_ptr{other.m_ptr} {
    other.m_capacity = 0;
    other.m_size = 0;
    other.m_ptr = nullptr;
}

template<typename T>
GuiDynamicArray<T>& GuiDynamicArray<T>::operator=(
    const GuiDynamicArray<T>& other) {
    if (&other != this) {
        m_capacity = other.m_capacity;
        m_size = other.m_size;

        m_ptr = new GuiDynamicArray<T>[m_capacity];
        for (auto i = 0; i < m_capacity; ++i) {
            m_ptr[i] = other.m_ptr[i];
        }
    }

    return *this;
}

template<typename T>
GuiDynamicArray<T>& GuiDynamicArray<T>::operator=(
    GuiDynamicArray&& other) noexcept {
    m_capacity = other.m_capacity;
    m_size = other.m_size;
    m_ptr = other.m_ptr;

    other.m_capacity = 0;
    other.m_size = 0;
    other.m_ptr = nullptr;

    return *this;
}

template<typename T>
GuiDynamicArray<T>::~GuiDynamicArray() {
    delete[] m_ptr;
}

template<typename T>
void GuiDynamicArray<T>::render_link(Vector2 src, Vector2 dest) {}

template<typename T>
void GuiDynamicArray<T>::render() {
    update();

    std::size_t idx = 0;

    for (std::size_t i = 0; i < m_capacity; ++i) {
        m_ptr[i].render();
    }
}

template<typename T>
void GuiDynamicArray<T>::update() {
    // TODO: if not outdated then return

    for (std::size_t i = 0; i < m_capacity; ++i) {
        m_ptr[i].set_pos(
            {head_pos.x + 4 * GuiElement<T>::side * i, head_pos.y});
    }
}

template<typename T>
T& GuiDynamicArray<T>::operator[](std::size_t idx) {
    return m_ptr[idx].get_value();
}

template<typename T>
T GuiDynamicArray<T>::operator[](std::size_t idx) const {
    return m_ptr[idx].get_value();
}

template<typename T>
```

```
void GuiDynamicArray<T>::set_color_index(std::size_t idx, int color_index) {
    m_ptr[idx].set_color_index(color_index);
}

template<typename T>
std::size_t GuiDynamicArray<T>::capacity() const {
    return m_capacity;
}

template<typename T>
std::size_t GuiDynamicArray<T>::size() const {
    return m_size;
}

template<typename T>
void GuiDynamicArray<T>::push(const T& value) {
    if (m_size == m_capacity) {
        realloc(m_size + 1);
    }

    m_ptr[m_size].set_color_index(0);
    m_ptr[m_size].set_value(value);
    ++m_size;
}

template<typename T>
void GuiDynamicArray<T>::pop() {
    if (m_size >= 1) {
        m_ptr[m_size - 1].set_color_index(1);
        m_ptr[m_size - 1].set_value(0);
        --m_size;
    }
}

}  // namespace gui

#endif  // GUI_DYNAMIC_ARRAY_GUI_HPP_
```

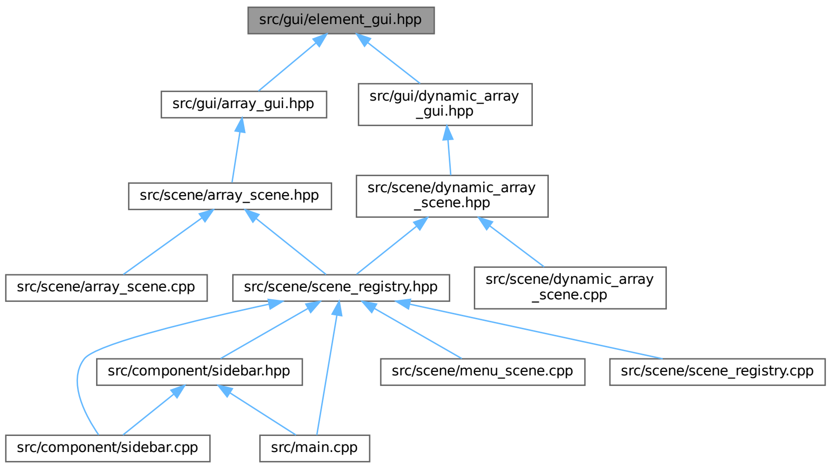
7.53 src/gui/element_gui.hpp File Reference

```
#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:


src/gui/element_gui.hpp
src/gui/array_gui.hpp
src/gui/dynamic_array_gui.hpp
src/scene/array_scene.hpp
src/scene/dynamic_array_scene.hpp
src/scene/array_scene.cpp
src/scene/scene_registry.hpp
src/scene/dynamic_array_scene.cpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class gui::GuiElement< T >

Namespaces

- namespace gui

7.54 element_gui.hpp

Go to the documentation of this file.

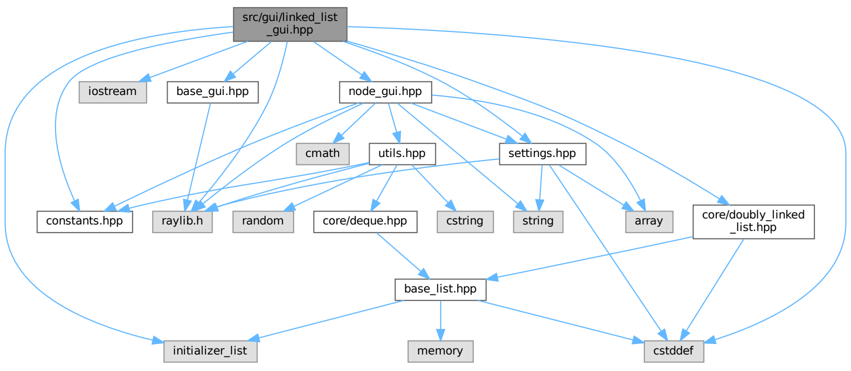
```
#ifndef GUI_ELEMENT_GUI_HPP_
#define GUI_ELEMENT_GUI_HPP_

#include <array>
#include <cmath>
#include <iostream>
#include <string>

#include "constants.hpp"
#include "raylib.h"
#include "settings.hpp"
#include "utils.hpp"

namespace gui {

template<typename T>
class GuiElement {
private:
    T m_value{};
    std::size_t m_index{};

    Vector2 m_pos{init_pos};
    static constexpr float eps = 1e-3;
    int m_color_index{1};

public:
    static constexpr int side = 20;
    static constexpr Vector2 init_pos{
        constants::sidebar_width +
            static_cast<float>(constants::scene_width -
                                constants::sidebar_width) /
                   2,
           0};

    GuiElement() = default;
    GuiElement(const T& value, std::size_t index);

    void render();
    void set_pos(Vector2 pos);
    void set_color_index(int color_index);
    [[nodiscard]] Vector2 get_pos() const;

    T& get_value();
    T get_value() const;
    void set_value(const T& value);
    void set_index(std::size_t index);
};

template<typename T>
GuiElement<T>::GuiElement(const T& value, std::size_t index)
    : m_value{value}, m_index{index} {}

template<typename T>
void GuiElement<T>::render() {
    constexpr int label_font_size = 25;
    constexpr int label_font_spacing = 2;
    const std::string label = std::to_string(m_value);
    const std::string index = std::to_string(m_index);

    const Vector2 label_size =
        utils::MeasureText(label.c_str(), label_font_size, label_font_spacing);

    const Vector2 label_pos{m_pos.x - label_size.x / 2,
                            m_pos.y - label_size.y / 2};

    const Vector2 index_size =
        utils::MeasureText(index.c_str(), label_font_size, label_font_spacing);

    const Vector2 index_pos{m_pos.x - index_size.x / 2,
                            m_pos.y - 2 * side - index_size.y / 2};

    const Color value_color =
        utils::adaptive_text_color(Settings::get_instance().get_color(0));
    const Color index_color = utils::adaptive_text_color(
        Settings::get_instance().get_color(Settings::num_color - 1));

    DrawRectangle(m_pos.x - side,  // NOLINT
                  m_pos.y - side,  // NOLINT
                  2 * side, 2 * side,
                  Settings::get_instance().get_color(m_color_index));

    utils::DrawText(label.c_str(), label_pos, value_color, label_font_size,
                    label_font_spacing);

    utils::DrawText(index.c_str(), index_pos, index_color, label_font_size,
                    label_font_spacing);
}

template<typename T>
void GuiElement<T>::set_pos(Vector2 pos) {
    m_pos = pos;
}

template<typename T>
void GuiElement<T>::set_color_index(int color_index) {
    m_color_index = color_index;
}

template<typename T>
T& GuiElement<T>::get_value() {
    return m_value;
}

template<typename T>
T GuiElement<T>::get_value() const {
    return m_value;
}

template<typename T>
void GuiElement<T>::set_value(const T& value) {
    m_value = value;
}

template<typename T>
void GuiElement<T>::set_index(std::size_t index) {
    m_index = index;
}
```

```

} // namespace gui

#endif // GUI_ELEMENT_GUI_HPP_
```

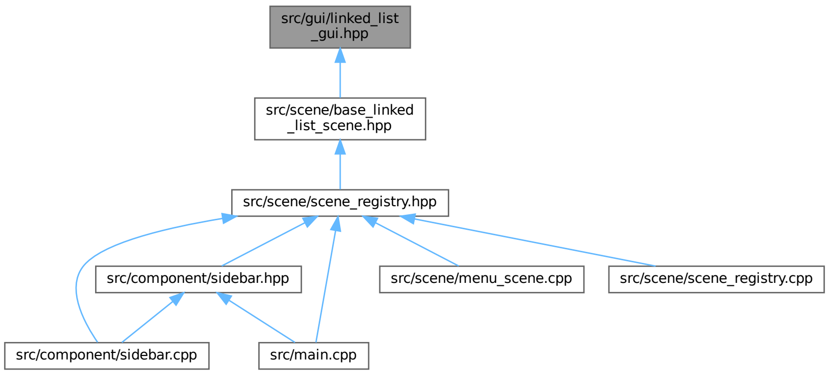
7.55 src/gui/linked_list_gui.hpp File Reference

```
#include <cstddef>
#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:


src/gui/linked_list
_gui.hpp
src/scene/base_linked
_list_scene.hpp
src/scene/scene_registry.hpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class gui::GuiLinkedList< T >

Namespaces

- namespace gui

7.56 linked_list_gui.hpp

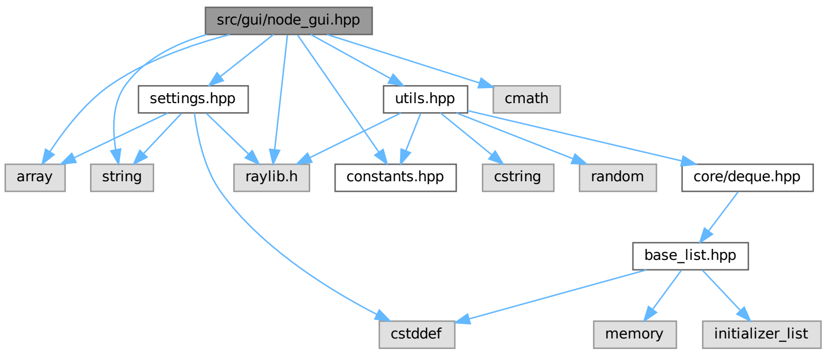
Go to the documentation of this file.

```
#ifndef GUI_LINKED_LIST_GUI_HPP_
#define GUI_LINKED_LIST_GUI_HPP_

#include <cstddef>
#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"

namespace gui {

template<typename T>
class GuiLinkedList : public core::DoublyLinkedList<GuiNode<T>»,
                      public internal::Base {
private:
    using Base = core::DoublyLinkedList<GuiNode<T>>;

    static constexpr Vector2 head_pos{
        constants::scene_width / 2.0F - 15 * GuiNode<T>::radius,
        constants::scene_height / 2.0F};

    using Base::m_head;
    using Base::m_tail;

    void render_link(Vector2 src, Vector2 dest) override;

public:
    using Base::Base;

    using Base::empty;
    using Base::size;

    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();
};

template<typename T>
void GuiLinkedList<T>::init_label() {
    if (m_head != nullptr) {
        m_head->data.set_label("head");
    }

    if (m_tail != nullptr) {
        if (m_head == m_tail) {
            m_tail->data.set_label("head/tail");
        } else {
            m_tail->data.set_label("tail");
        }
    }
}

template<typename T>
GuiLinkedList<T>::GuiLinkedList(std::initializer_list<GuiNode<T>> init_list)
    : core::DoublyLinkedList<GuiNode<T»(init_list) {
    init_label();
}

template<typename T>
void GuiLinkedList<T>::insert(std::size_t index, const T& elem) {
    Base::insert(index, GuiNode{elem});
}

template<typename T>
void GuiLinkedList<T>::render_link(Vector2 src, Vector2 dest) {
    constexpr int radius = GuiNode<T>::radius;
    constexpr float scaled_len = radius / 8.0F;

    // straight line
    Vector2 link_pos{src.x + radius, src.y - scaled_len};
    Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};

    // arrow
    constexpr int arrow_size = scaled_len * 5;
    Vector2 head{dest.x - radius + scaled_len / 2, src.y};
    Vector2 side_top{head.x - arrow_size, head.y - arrow_size};
    Vector2 side_bot{head.x - arrow_size, head.y + arrow_size};

    // draw both
    DrawRectangleV(link_pos, link_size, Settings::get_instance().get_color(1));
    DrawTriangle(head, side_top, side_bot,
                 Settings::get_instance().get_color(1));
}

template<typename T>
void GuiLinkedList<T>::render() {
    update();

    for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
        if (ptr->next != nullptr) {
            render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
        }

        ptr->data.render();
    }
}

template<typename T>
void GuiLinkedList<T>::update() {
    // TODO: if not outdated then return

    std::size_t pos = 0;

    for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
        ptr->data.set_pos(
            {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
        ++pos;
    }
}

}  // namespace gui

#endif  // GUI_LINKED_LIST_GUI_HPP_
```

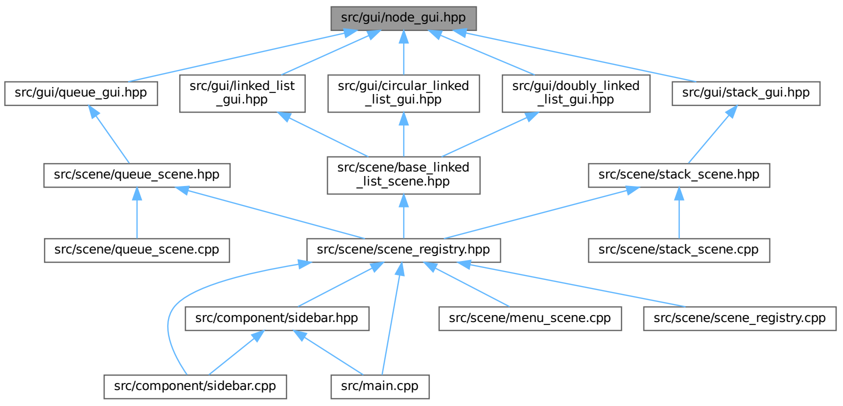
7.57 src/gui/node_gui.hpp File Reference

```
#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:


src/gui/node_gui.hpp
src/gui/queue_gui.hpp
src/gui/linked_list_gui.hpp
src/gui/circular_linked_list_gui.hpp
src/gui/doubly_linked_list_gui.hpp
src/gui/stack_gui.hpp
src/scene/queue_scene.hpp
src/scene/base_linked_list_scene.hpp
src/scene/stack_scene.hpp
src/scene/queue_scene.cpp
src/scene/scene_registry.hpp
src/scene/stack_scene.cpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class gui::GuiNode< T >

Namespaces

- namespace gui

7.58 node_gui.hpp

Go to the documentation of this file.

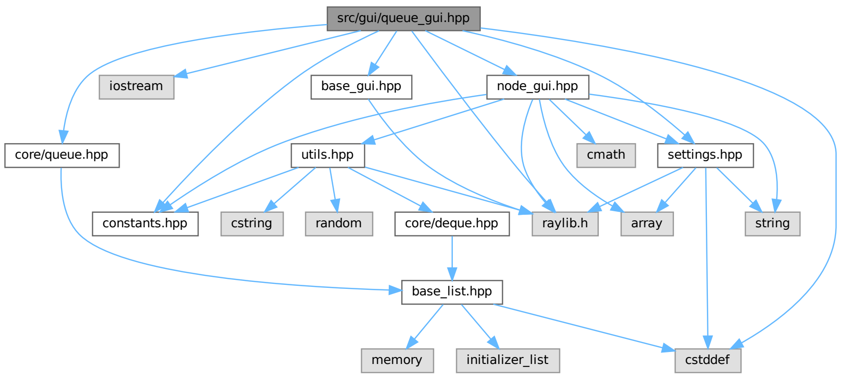
```
#ifndef GUI_NODE_GUI_HPP_
#define GUI_NODE_GUI_HPP_

#include <array>
#include <cmath>
```

```
#include <string>

#include "constants.hpp"
#include "raylib.h"
#include "settings.hpp"
#include "utils.hpp"

namespace gui {

template<typename T>
class GuiNode {
private:
    T m_value{};
    int m_color_index{0};

    Vector2 m_pos{constants::sidebar_width +
                      static_cast<float>(constants::scene_width -
                                         constants::sidebar_width) /
                          2,
                  0};
    static constexpr float eps = 1e-3;
    const char* m_label{};

public:
    static constexpr int radius = 20;

    explicit GuiNode(const T& value);

    void render();
    void set_pos(Vector2 pos);
    [[nodiscard]] 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);
};

template<typename T>
GuiNode<T>::GuiNode(const T& value) : m_value{value} {}

template<typename T>
void GuiNode<T>::render() {
    constexpr int label_font_size = 25;
    constexpr int label_font_spacing = 2;
    const std::string value = std::to_string(m_value);
    const Settings& settings = Settings::get_instance();

    const Vector2 value_size =
        utils::MeasureText(value.c_str(), label_font_size, label_font_spacing);

    const Vector2 value_pos{m_pos.x - value_size.x / 2,
                            m_pos.y - value_size.y / 2};

    const Vector2 label_size =
        utils::MeasureText(m_label, label_font_size, label_font_spacing);

    const Vector2 label_pos{m_pos.x - label_size.x / 2,
                            m_pos.y - 2 * label_size.y};

    const Color value_color =
        utils::adaptive_text_color(Settings::get_instance().get_color(0));

    DrawCircleV(m_pos, radius, settings.get_color(m_color_index));
    utils::DrawText(value.c_str(), value_pos, value_color, label_font_size,
                    label_font_spacing);

    utils::DrawText(m_label, label_pos, settings.get_color(5), label_font_size,
                    label_font_spacing);
}

template<typename T>
void GuiNode<T>::set_color_index(int color_index) {
    m_color_index = color_index;
}

template<typename T>
void GuiNode<T>::set_value(const T& value) {
    m_value = value;
}

template<typename T>
T& GuiNode<T>::get_value() {
    return m_value;
}

template<typename T>
void GuiNode<T>::set_pos(Vector2 pos) {
    m_pos = pos;
}

template<typename T>
Vector2 GuiNode<T>::get_pos() const {
    return m_pos;
}

template<typename T>
void GuiNode<T>::set_label(const char* label) {
    m_label = label;
}

}  // namespace gui

#endif  // GUI_NODE_GUI_HPP_
```

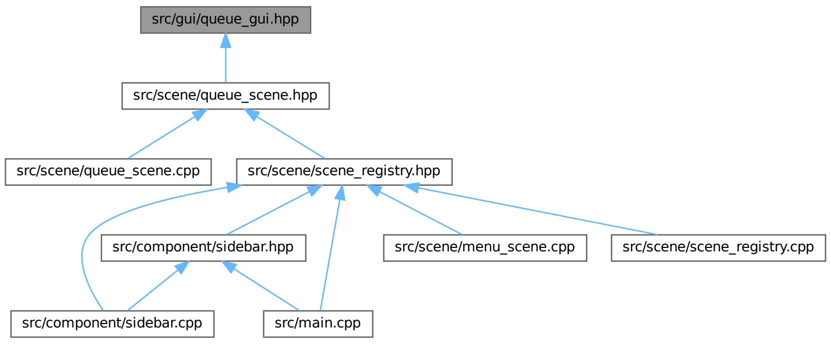
7.59 src/gui/queue_gui.hpp File Reference

```
#include <cstddef>
#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:


src/gui/queue_gui.hpp
src/scene/queue_scene.hpp
src/scene/queue_scene.cpp
src/scene/scene_registry.hpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class gui::GuiQueue< T >

Namespaces

- namespace gui

7.60 queue_gui.hpp

Go to the documentation of this file.

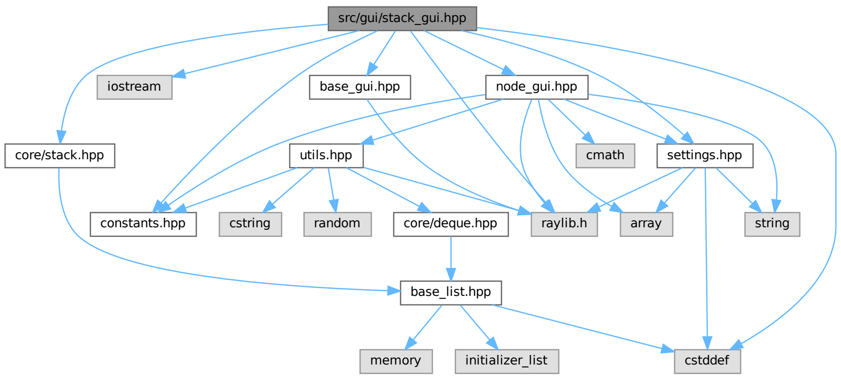
```
#ifndef GUI_QUEUE_GUI_HPP_
#define GUI_QUEUE_GUI_HPP_

#include <cstddef>
#include <iostream>

#include "base_gui.hpp"
#include "constants.hpp"
#include "core/queue.hpp"
#include "node_gui.hpp"
#include "raylib.h"
#include "settings.hpp"

namespace gui {

template<typename T>
class GuiQueue : public core::Queue<GuiNode<T>>, public internal::Base {
private:
    using Base = core::Queue<GuiNode<T>>;

    static constexpr Vector2 head_pos{
        constants::scene_width / 2.0F - 15 * GuiNode<T>::radius,
        constants::scene_height / 2.0F};

    using Base::m_head;
    using Base::m_tail;

    void render_link(Vector2 src, Vector2 dest) override;

public:
    using Base::Base;

    using Base::empty;
    using Base::size;

    GuiQueue(std::initializer_list<GuiNode<T>> init_list);

    void push(const T& elem);
    void pop();

    // for animation purpose only, not for real use
    void push_front(const T& elem);
    void pop_back();

    void update() override;
    void render() override;
    void init_label();
};

template<typename T>
void GuiQueue<T>::init_label() {
    if (m_head != nullptr) {
        m_head->data.set_label("head");
    }

    if (m_tail != nullptr) {
        if (m_head == m_tail) {
            m_tail->data.set_label("head/tail");
        } else {
            m_tail->data.set_label("tail");
        }
    }
}

template<typename T>
GuiQueue<T>::GuiQueue(std::initializer_list<GuiNode<T>> init_list)
    : core::Queue<GuiNode<T»(init_list) {
    init_label();
}

template<typename T>
void GuiQueue<T>::push(const T& elem) {
    Base::push(GuiNode<T>{elem});
}

template<typename T>
void GuiQueue<T>::pop() {
    Base::pop();
}

template<typename T>
void GuiQueue<T>::push_front(const T& elem) {
    Base::push_front(GuiNode<T>{elem});
}

template<typename T>
void GuiQueue<T>::pop_back() {
    Base::pop_back();
}

template<typename T>
void GuiQueue<T>::render_link(Vector2 src, Vector2 dest) {
    constexpr int radius = GuiNode<T>::radius;
    constexpr float scaled_len = radius / 8.0F;

    // straight line
    Vector2 link_pos{src.x + radius, src.y - scaled_len};
    Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};

    // arrow
    constexpr int arrow_size = scaled_len * 5;
    Vector2 head{dest.x - radius + scaled_len / 2, src.y};
    Vector2 side_top{head.x - arrow_size, head.y - arrow_size};
    Vector2 side_bot{head.x - arrow_size, head.y + arrow_size};

    // draw both
    DrawRectangleV(link_pos, link_size, Settings::get_instance().get_color(1));
    DrawTriangle(head, side_top, side_bot,
                 Settings::get_instance().get_color(1));
}

template<typename T>
void GuiQueue<T>::render() {
    update();

    for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
        if (ptr->next != nullptr) {
            render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
        }

        ptr->data.render();
    }
}
```

```
template<typename T>
void GuiQueue<T>::update() {
    // TODO: if not outdated then return

    std::size_t pos = 0;

    for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
        ptr->data.set_pos(
            {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
        ++pos;
    }
}

}  // namespace gui

#endif  // GUI_QUEUE_GUI_HPP_
```

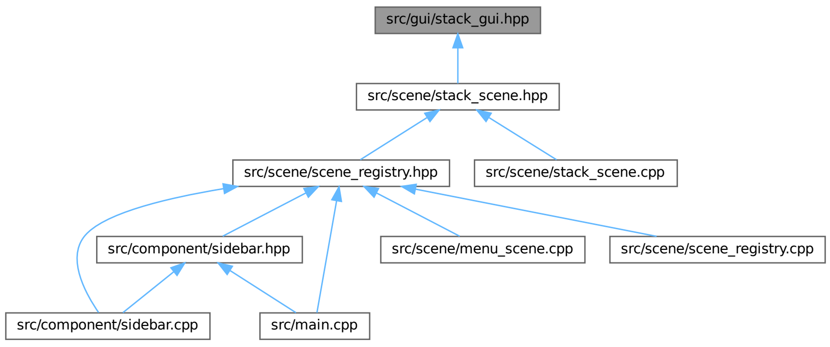
7.61 src/gui/stack_gui.hpp File Reference

```
#include <cstddef>
#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:


src/gui/stack_gui.hpp
src/scene/stack_scene.hpp
src/scene/scene_registry.hpp
src/scene/stack_scene.cpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class gui::GuiStack< T >

Namespaces

- namespace gui

7.62 stack_gui.hpp

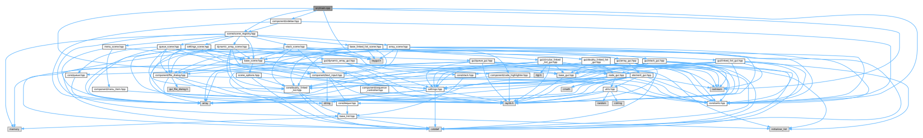
Go to the documentation of this file.

```
#ifndef GUI_STACK_GUI_HPP_
#define GUI_STACK_GUI_HPP_

#include <cstddef>
#include <iostream>

#include "base_gui.hpp"
#include "constants.hpp"
#include "core/stack.hpp"
#include "node_gui.hpp"
#include "raylib.h"
#include "settings.hpp"

namespace gui {

template<typename T>
class GuiStack : public core::Stack<GuiNode<T>>, public internal::Base {
private:
    using Base = core::Stack<GuiNode<T>>;

    static constexpr Vector2 head_pos{
        constants::scene_width / 2.0F - GuiNode<T>::radius / 2.0F,
        GuiNode<T>::radius * 4.0F};

    using Base::m_head;
    using Base::m_tail;

    void render_link(Vector2 src, Vector2 dest) override;

public:
    using Base::Base;

    using Base::empty;
    using Base::size;

    GuiStack(std::initializer_list<GuiNode<T>> init_list);

    void push(const T& elem);
    void pop();

    void update() override;
    void render() override;
    void init_label();
};

template<typename T>
void GuiStack<T>::init_label() {
    if (m_head != nullptr) {
        m_head->data.set_label("head");
    }
}

template<typename T>
GuiStack<T>::GuiStack(std::initializer_list<GuiNode<T>> init_list)
    : core::Stack<GuiNode<T»(init_list) {
    init_label();
}

template<typename T>
void GuiStack<T>::push(const T& elem) {
    Base::push(GuiNode<T>{elem});
}

template<typename T>
void GuiStack<T>::pop() {
    Base::pop();
}

template<typename T>
void GuiStack<T>::render_link(Vector2 src, Vector2 dest) {
    constexpr int radius = GuiNode<T>::radius;
    constexpr float scaled_len = radius / 8.0F;

    // straight line
    Vector2 link_pos{src.x - scaled_len, src.y + radius};
    Vector2 link_size{2 * scaled_len, dest.y - src.y - 2 * radius};

    // arrow
    constexpr int arrow_size = scaled_len * 5;
    Vector2 head{src.x, dest.y - radius + scaled_len / 2};
    Vector2 side_left{head.x - arrow_size, head.y - arrow_size};
    Vector2 side_right{head.x + arrow_size, head.y - arrow_size};

    // draw both
    DrawRectangleV(link_pos, link_size, Settings::get_instance().get_color(1));
    DrawTriangle(head, side_right, side_left,
                 Settings::get_instance().get_color(1));
}

template<typename T>
void GuiStack<T>::render() {
    update();

    for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
        if (ptr->next != nullptr) {
            render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
        }

        ptr->data.render();
    }
}

template<typename T>
void GuiStack<T>::update() {
    // TODO: if not outdated then return

    std::size_t pos = 0;

    for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
        ptr->data.set_pos(
            {head_pos.x, head_pos.y + 4 * GuiNode<T>::radius * pos});
        ++pos;
    }
}

}  // namespace gui

#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 ()

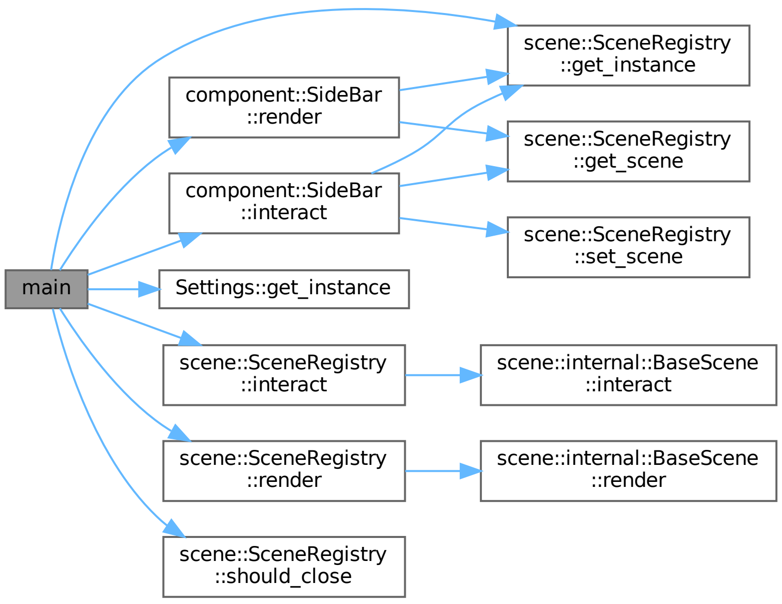
7.63.1 Function Documentation

7.63.1.1 main()

```
int main ( )
```

Definition at line 9 of file main.cpp.

Here is the call graph for this function:


scene::SceneRegistry ::get_instance
component::SideBar ::render
scene::SceneRegistry ::get_scene
component::SideBar ::interact
scene::SceneRegistry ::set_scene
main
Settings::get_instance
scene::SceneRegistry ::interact
scene::internal::BaseScene ::interact
scene::SceneRegistry ::render
scene::internal::BaseScene ::render
scene::SceneRegistry ::should_close

7.64 main.cpp

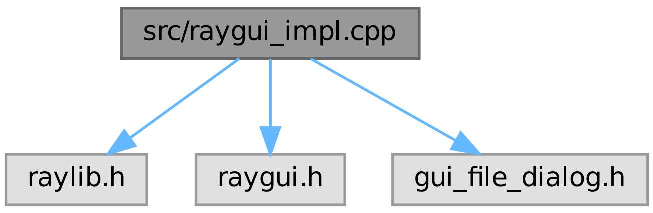
Go to the documentation of this file.

```
#include <iostream>

#include "component/sidebar.hpp"
#include "constants.hpp"
#include "raygui.h"
#include "scene/scene_registry.hpp"
#include "settings.hpp"

int main() {
    InitWindow(constants::scene_width, constants::scene_height,
               "VisuAlgo.net clone in C++ by @jalsol");
    SetTargetFPS(constants::frames_per_second);

    GuiLoadStyle("data/bluish_open_sans.rgs");

    scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();
    component::SideBar sidebar;

    bool should_close = false;

    do {
        // NOTE: The order is important
        sidebar.interact();
        registry.interact();

        BeginDrawing();
        {
            ClearBackground(
                Settings::get_instance().get_color(Settings::num_color - 1));

            // NOTE: The order is important
            registry.render();
            sidebar.render();
        }
        EndDrawing();

        should_close = registry.should_close() || WindowShouldClose();
    } while (!should_close);

    CloseWindow();

    return 0;
}
```

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:



Macros

- #define RAYGUI_IMPLEMENTATION
- #define GUI_FILE_DIALOG_IMPLEMENTATION

7.65.1 Macro Definition Documentation

7.65.1.1 GUI_FILE_DIALOG_IMPLEMENTATION

```
#define GUI_FILE_DIALOG_IMPLEMENTATION
```

Definition at line 6 of file raygui_impl.cpp.

7.65.1.2 RAYGUI_IMPLEMENTATION

```
#define RAYGUI_IMPLEMENTATION
```

Definition at line 2 of file raygui_impl.cpp.

7.66 raygui_impl.cpp

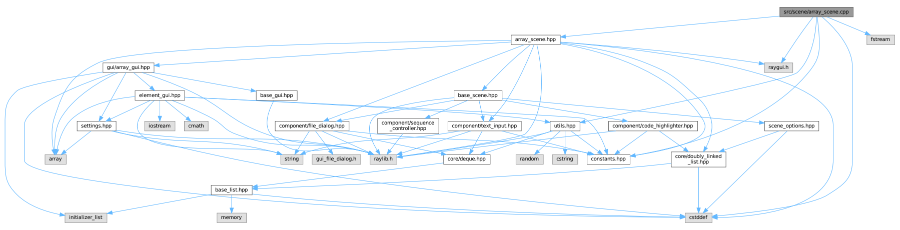
Go to the documentation of this file.

```
00001 #include "raylib.h"
00002 #define RAYGUI_IMPLEMENTATION
00003 #include "raygui.h"
00004
00005 #undef RAYGUI_IMPLEMENTATION
00006 #define GUI_FILE_DIALOG_IMPLEMENTATION
00007 #include "gui_file_dialog.h"
```

7.67 src/scene/array_scene.cpp File Reference

```
#include "array_scene.hpp"
#include <cstddef>
#include <fstream>
#include "constants.hpp"
#include "raygui.h"
#include "utils.hpp"
```

Include dependency graph for array_scene.cpp:



Namespaces

- namespace scene

7.68 array_scene.cpp

Go to the documentation of this file.

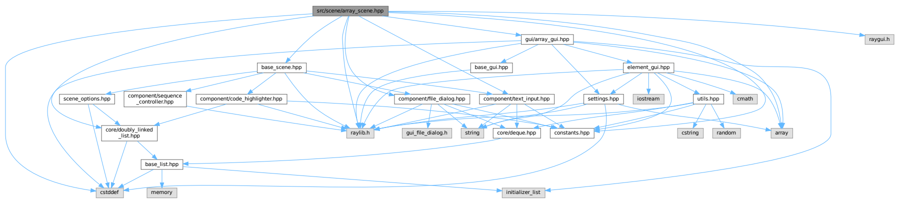
```
#include "array_scene.hpp"

#include <cstddef>
// #include <cstdlib>
// #include <cstring>
#include <fstream>
// #include <iostream>
// #include <limits>
// #include <string>

#include "constants.hpp"
#include "raygui.h"
#include "utils.hpp"

namespace scene {

void ArrayScene::render_inputs() {
    int& mode = scene_options.mode_selection;

    switch (mode) {
        case 0: {
            switch (scene_options.action_selection.at(mode)) {
                case 0:
                    break;
                case 1: {
                    m_text_input.render(options_head, head_offset);
                } break;
                case 2: {
                    m_go = (m_file_dialog.render_head(options_head,
                                                      head_offset) > 0);
                    return;
                } break;
                default:
                    utils::unreachable();
            }
        } break;

        case 1: {
            m_index_input.render(options_head, head_offset);
            m_text_input.render(options_head, head_offset);
        } break;

        case 2: {
            m_text_input.render(options_head, head_offset);
        } break;

        default:
            utils::unreachable();
    }

    m_go |= render_go_button();
}

void ArrayScene::render() {
    m_sequence_controller.inc_anim_counter();

    int frame_idx = m_sequence_controller.get_anim_frame();
    auto* const frame_ptr = m_sequence.find(frame_idx);
    m_sequence_controller.set_progress_value(frame_idx);

    if (frame_ptr != nullptr) {
        frame_ptr->data.render();
        m_code_highlighter.highlight(frame_idx);
    } else {  // end of sequence
        m_array.render();
        m_sequence_controller.set_run_all(false);
    }

    m_code_highlighter.render();
    m_sequence_controller.render();
    render_options(scene_options);
}

```

```
void ArrayScene::interact() {
    if (m_sequence_controller.interact()) {
        m_sequence_controller.reset_anim_counter();
        return;
    }

    m_index_input.set_random_max(max_size);

    if (m_text_input.interact() || m_index_input.interact()) {
        return;
    }

    if (!m_go) {
        return;
    }

    int& mode = scene_options.mode_selection;

    switch (mode) {
        case 0: {
            switch (scene_options.action_selection.at(mode)) {
                case 0: {
                    interact_random();
                } break;

                case 1: {
                    interact_import(m_text_input.extract_values());
                } break;

                case 2: {
                    interact_file_import();
                } break;

                default:
                    utils::unreachable();
            }
        } break;

        case 1: {
            interact_update();
        } break;

        case 2: {
            interact_search();
        } break;

        default:
            utils::unreachable();
    }

    m_go = false;
}

void ArrayScene::interact_random() {
    m_array = {};

    for (std::size_t i = 0; i < max_size; ++i) {
        m_array[i] = utils::get_random(constants::min_val, constants::max_val);
    }
}

void ArrayScene::interact_import(core::Deque<int> nums) {
    m_array = {};
    std::size_t i;  // NOLINT

    for (i = 0; i < max_size && !nums.empty(); ++i) {
        m_array[i] = nums.front();
        nums.pop_front();
    }

    for (; i < max_size; ++i) {
        m_array[i] = 0;
    }
}

void ArrayScene::interact_update() {
    auto index_container = m_index_input.extract_values();
    if (index_container.empty()) {
        return;
    }

    auto value_container = m_text_input.extract_values();
    if (value_container.empty()) {
        return;
    }

    int index = index_container.front();
    int value = value_container.front();

    if (!(0 <= index && index < max_size) || !utils::val_in_range(value)) {
        return;
    }

    m_code_highlighter.set_code({
        "array[index] = value;",
    });

    m_sequence.clear();

    // initial state (before update)
    m_sequence.insert(m_sequence.size(), m_array);
    m_code_highlighter.push_into_sequence(-1);

    // highlight
    m_array.set_color_index(index, 2);
    m_sequence.insert(m_sequence.size(), m_array);
    m_code_highlighter.push_into_sequence(0);

    // update
    m_array[index] = value;
    m_array.set_color_index(index, 3);
    m_sequence.insert(m_sequence.size(), m_array);
    m_code_highlighter.push_into_sequence(0);

    // undo highlight
    m_array.set_color_index(index, 0);

    m_sequence_controller.set_max_value((int)m_sequence.size());
    m_sequence_controller.set_rerun();
}

void ArrayScene::interact_file_import() {
    interact_import(m_file_dialog.extract_values());
}

void ArrayScene::interact_search() {
    auto value_container = m_text_input.extract_values();
    if (value_container.empty()) {
        return;
    }

    int value = value_container.front();
    if (!utils::val_in_range(value)) {
        return;
    }

    m_code_highlighter.set_code({
        "for (i = 0; i < size; i++)",
        "    if (array[i] == value)",
        "        return i;",
        "return not_found",
    });

    m_sequence.clear();
    m_sequence.insert(m_sequence.size(), m_array);
    m_code_highlighter.push_into_sequence(0);

    bool found = false;

    for (std::size_t i = 0; i < max_size; ++i) {
        m_array.set_color_index(i, 3);
        m_sequence.insert(m_sequence.size(), m_array);
        m_code_highlighter.push_into_sequence(1);

        if (m_array[i] == value) {
            found = true;
            m_array.set_color_index(i, 4);
            m_sequence.insert(m_sequence.size(), m_array);
            m_code_highlighter.push_into_sequence(2);
            m_array.set_color_index(i, 0);
            break;
        }

        m_array.set_color_index(i, 0);
        m_sequence.insert(m_sequence.size(), m_array);
        m_code_highlighter.push_into_sequence(0);
    }

    if (!found) {
        m_sequence.insert(m_sequence.size(), m_array);
        m_code_highlighter.push_into_sequence(3);
    }

    m_sequence_controller.set_max_value((int)m_sequence.size());
    m_sequence_controller.set_rerun();
}

}  // namespace scene
```

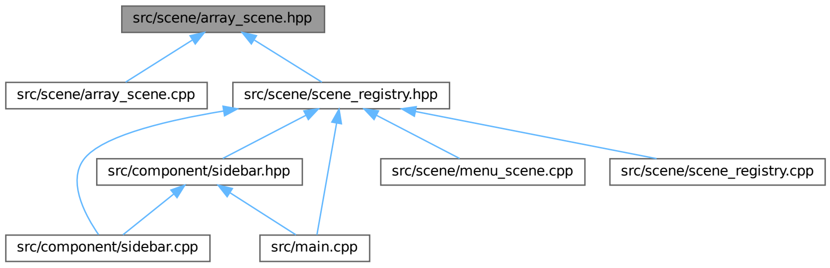
7.69 src/scene/array_scene.hpp File Reference

```
#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:


src/scene/array_scene.hpp
src/scene/array_scene.cpp
src/scene/scene_registry.hpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class scene::ArrayScene

Namespaces

- namespace scene

7.70 array_scene.hpp

Go to the documentation of this file.

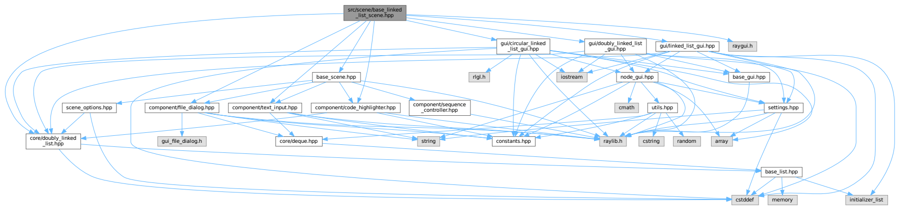
```
#ifndef SCENE_ARRAY_SCENE_HPP_
#define SCENE_ARRAY_SCENE_HPP_

#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"

namespace scene {

class ArrayScene : public internal::BaseScene {
private:
    static constexpr std::size_t max_size = 8;

    internal::SceneOptions scene_options{
        // max_size
        max_size,

        // mode_labels
        "Mode: Create;"
        "Mode: Update;"
        "Mode: Search",

        // mode_selection
        0,

        // action_labels
        {
            // Mode: Create
            "Action: Random;"
            "Action: Input;"
            "Action: File",

            // Mode: Update
            "",

            // Mode: Search
            "",
        },

        // action_selection
        core::DoublyLinkedList<int>{0, 0, 0},
    };

    using internal::BaseScene::button_size;
    using internal::BaseScene::head_offset;
    using internal::BaseScene::options_head;

    gui::GuiArray<int, max_size> m_array{};
    core::DoublyLinkedList<gui::GuiArray<int, max_size>> m_sequence;

    bool m_go{};

    using internal::BaseScene::m_code_highlighter;
    using internal::BaseScene::m_file_dialog;
    using internal::BaseScene::m_index_input;
    using internal::BaseScene::m_sequence_controller;
    using internal::BaseScene::m_text_input;

    using internal::BaseScene::render_go_button;
    using internal::BaseScene::render_options;
    void render_inputs() override;

    void interact_random();
    void interact_import(core::Deque<int> nums);
    void interact_file_import();
    void interact_update();
    void interact_search();

public:
    void render() override;
    void interact() override;
};

}  // namespace scene
```

```
00083
00084 #endif  // SCENE_ARRAY_SCENE_HPP_
```

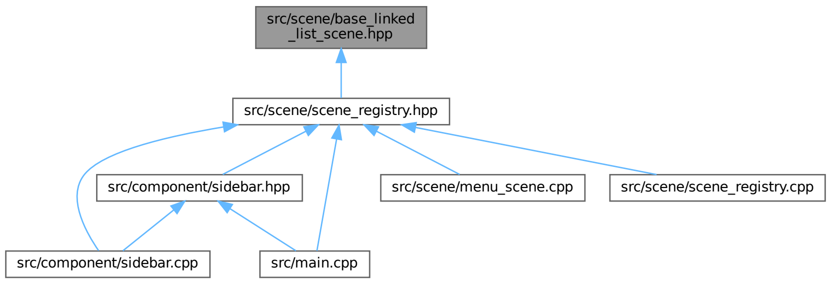
7.71 src/scene/base_linked_list_scene.hpp File Reference

```
#include "base_scene.hpp"
#include "component/code_highlighter.hpp"
#include "component/file_dialog.hpp"
#include "component/text_input.hpp"
#include "core/doubly_linked_list.hpp"
#include "gui/circular_linked_list_gui.hpp"
#include "gui/doubly_linked_list_gui.hpp"
#include "gui/linked_list_gui.hpp"
#include "raygui.h"
```

Include dependency graph for base_linked_list_scene.hpp:



This graph shows which files directly or indirectly include this file:


src/scene/base_linked_list_scene.hpp
src/scene/scene_registry.hpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class scene::BaseLinkedListScene< Con >

Namespaces

- 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.

```
#ifndef SCENE_BASE_LINKED_LIST_SCENE_HPP_
#define SCENE_BASE_LINKED_LIST_SCENE_HPP_

#include "base_scene.hpp"
#include "component/code_highlighter.hpp"
#include "component/file_dialog.hpp"
#include "component/text_input.hpp"
#include "core/doubly_linked_list.hpp"
#include "gui/circular_linked_list_gui.hpp"
#include "gui/doubly_linked_list_gui.hpp"
#include "gui/linked_list_gui.hpp"
#include "raygui.h"

namespace scene {

template<typename Con>
class BaseLinkedListScene : public internal::BaseScene {
private:
    internal::SceneOptions scene_options{
        // max_size
        8,  // NOLINT

        // mode_labels
        "Mode: Create;"
        "Mode: Add;"
        "Mode: Delete;"
        "Mode: Update;"
        "Mode: Search",

        // mode_selection
        0,

        // action_labels
        {
            // Mode: Create
            "Action: Random;Action: Input;Action: File",
            // Mode: Add
            "",
            // Mode: Delete
            "",
            // Mode: Update
            "",
            // Mode: Search
            "",
        },

        // action_selection
        core::DoublyLinkedList<int>{0, 0, 0, 0, 0},
    };

    using internal::BaseScene::button_size;
    using internal::BaseScene::head_offset;
    using internal::BaseScene::options_head;

    Con m_list{
        gui::GuiNode<int>{1},
        gui::GuiNode<int>{2},
        gui::GuiNode<int>{3},
    };
    core::DoublyLinkedList<Con> m_sequence;

    bool m_go{};
    using internal::BaseScene::m_code_highlighter;
    using internal::BaseScene::m_file_dialog;
    using internal::BaseScene::m_index_input;
    using internal::BaseScene::m_sequence_controller;
    using internal::BaseScene::m_text_input;

    using internal::BaseScene::render_go_button;
    using internal::BaseScene::render_options;
    void render_inputs() override;

    void interact_random();
    void interact_import(core::Deque<int> nums);
    void interact_file_import();

    void interact_add();
    void interact_add_head(int value);
    void interact_add_tail(int value);
    void interact_add_middle(int index, int value);

    void interact_delete();
    void interact_delete_head();
    void interact_delete_tail();
    void interact_delete_middle(int index);

    void interact_update();
    void interact_search();

public:
    void render() override;
    void interact() override;
};

using LinkedListScene = BaseLinkedListScene<gui::GuiLinkedList<int>>;
using DoublyLinkedListScene =
    BaseLinkedListScene<gui::GuiDoublyLinkedList<int>>;
using CircularLinkedListScene =
    BaseLinkedListScene<gui::GuiCircularLinkedList<int>>;

template<typename Con>
void BaseLinkedListScene<Con>::render_inputs() {
    int& mode = scene_options.mode_selection;

    switch (mode) {
        case 0: {
            switch (scene_options.action_selection.at(mode)) {
                case 0:
                    break;
                case 1: {
                    m_text_input.render(options_head, head_offset);
                } break;
                case 2: {
                    m_go = (m_file_dialog.render_head(options_head,
                                                      head_offset) > 0);
                    return;
                } break;
                default:
                    utils::unreachable();
            }
        } break;

        case 1: {
            m_index_input.render(options_head, head_offset);
            m_text_input.render(options_head, head_offset);
        } break;

        case 2: {
            m_index_input.render(options_head, head_offset);
        } break;

        case 3: {
            m_index_input.render(options_head, head_offset);
            m_text_input.render(options_head, head_offset);
        } break;

        case 4: {
            m_text_input.render(options_head, head_offset);
        } break;

        default:
            utils::unreachable();
    }

    m_go |= render_go_button();
}

template<typename Con>
void BaseLinkedListScene<Con>::render() {
    m_sequence_controller.inc_anim_counter();

    int frame_idx = m_sequence_controller.get_anim_frame();
    auto* const frame_ptr = m_sequence.find(frame_idx);
    m_sequence_controller.set_progress_value(frame_idx);

    if (frame_ptr != nullptr) {
        frame_ptr->data.render();
        m_code_highlighter.highlight(frame_idx);
    } else {  // end of sequence
        m_list.render();
        m_sequence_controller.set_run_all(false);
    }

    m_code_highlighter.render();
    m_sequence_controller.render();
    render_options(scene_options);
}

template<typename Con>
void BaseLinkedListScene<Con>::interact() {
    if (m_sequence_controller.interact()) {
        m_sequence_controller.reset_anim_counter();
        return;
    }

    m_index_input.set_random_max((int)m_list.size() - 1);

    if (m_text_input.interact() || m_index_input.interact()) {
        return;
    }

    if (!m_go) {
        return;
    }

    int& mode = scene_options.mode_selection;

    switch (mode) {
        case 0: {
            switch (scene_options.action_selection.at(mode)) {
                case 0: {
                    interact_random();
                } break;

                case 1: {
                    interact_import(m_text_input.extract_values());
                } break;

                case 2: {
                    interact_file_import();
                } break;

                default:
                    utils::unreachable();
            }
        } break;

        case 1: {
            m_index_input.set_random_max((int)m_list.size());
            interact_add();
        } break;

        case 2: {
            interact_delete();
        } break;

        case 3: {
            interact_update();
        } break;

        case 4: {
            interact_search();
        } break;

        default:
            utils::unreachable();
    }

    m_go = false;
}

template<typename Con>
void BaseLinkedListScene<Con>::interact_random() {
    std::size_t size =
        utils::get_random(std::size_t{1}, scene_options.max_size);
    m_list = Con();

    for (auto i = 0; i < size; ++i) {
        m_list.insert(
            i, utils::get_random(constants::min_val, constants::max_val));
    }
    m_list.init_label();
}

```

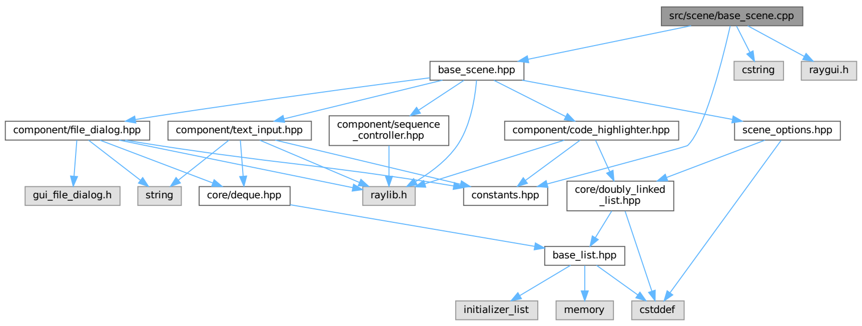
```
template<typename Con>
void BaseLinkedListScene<Con>::interact_import(core::Deque<int> nums) {
    m_sequence.clear();
    m_list = Con();

    while (!nums.empty()) {
        if (utils::val_in_range(nums.front())) {
            m_list.insert(m_list.size(), nums.front());
        }
        nums.pop_front();
    }
    m_list.init_label();
}

template<typename Con>
void BaseLinkedListScene<Con>::interact_file_import() {
    interact_import(m_file_dialog.extract_values());
}

template<typename Con>
void BaseLinkedListScene<Con>::interact_add() {
    auto index_container = m_index_input.extract_values();
    if (index_container.empty()) {
        return;
    }

    auto value_container = m_text_input.extract_values();
    if (value_container.empty()) {
        return;
    }

    int index = index_container.front();
    int value = value_container.front();

    if (!(0 <= index && index <= m_list.size())) {
        return;
    }

    if (!utils::val_in_range(value)) {
        return;
    }

    m_sequence.clear();
    m_sequence.insert(m_sequence.size(), m_list);

    if (index == 0) {
        interact_add_head(value);
    } else if (index == m_list.size()) {
        interact_add_tail(value);
    } else {
        interact_add_middle(index, value);
    }

    m_sequence_controller.set_max_value((int)m_sequence.size());
    m_sequence_controller.set_rerun();
}

template<typename Con>
void BaseLinkedListScene<Con>::interact_add_head(int value) {
    m_code_highlighter.set_code({
        "Node* node = new Node(value);",
        "node->next = head;",
        "head = next;",
    });
    m_code_highlighter.push_into_sequence(-1);

    m_list.insert(0, value);

    m_list.at(0).set_color_index(6);
    m_list.at(0).set_label("node");
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(0);

    if (m_list.size() > 1) {
        m_list.at(1).set_color_index(4);
    }

    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(1);

    if (m_list.size() > 1) {
        m_list.at(1).set_color_index(0);
        m_list.at(1).set_label("");
    }

    m_list.at(0).set_color_index(4);
    m_list.at(0).set_label("head");
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(2);

    m_list.at(0).set_color_index(0);
}

template<typename Con>
void BaseLinkedListScene<Con>::interact_add_tail(int value) {
    m_code_highlighter.set_code({
        "Node* node = new Node(value);",
        "tail->next = node;",
        "tail = tail->next;",
    });
    m_code_highlighter.push_into_sequence(-1);

    std::size_t size = m_list.size();

    m_list.insert(size, value);
    m_list.at(size).set_color_index(6);
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(0);

    m_list.at(size - 1).set_color_index(4);
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(1);

    m_list.at(size - 1).set_color_index(0);
    m_list.at(size - 1).set_label("");
    m_list.at(size).set_color_index(4);
    m_list.at(size).set_label("tail");
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(2);

    m_list.at(size).set_color_index(0);
}

template<typename Con>
void BaseLinkedListScene<Con>::interact_add_middle(int index, int value) {
    m_code_highlighter.set_code({
        "Node* pre = head;",
        "for (i = 0; i < index - 1; ++i)",
        "    pre = pre->next;",
        "",
        "Node* nxt = pre->next;",
        "Node* node = new Node(value);",
        "node->next = nxt;",
        "pre->next = node;",
    });
    m_code_highlighter.push_into_sequence(-1);

    m_list.at(0).set_color_index(4);
    m_list.at(0).set_label("head/pre");
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(0);

    // search until index - 1
    for (int i = 0; i < index - 1; ++i) {
        m_list.at(i).set_color_index(2);
        m_sequence.insert(m_sequence.size(), m_list);
        m_code_highlighter.push_into_sequence(1);

        m_list.at(i).set_color_index(0);
        m_list.at(i).set_label(i == 0 ? "head" : "");
        m_list.at(i + 1).set_color_index(2);
        m_list.at(i + 1).set_label("pre");
        m_sequence.insert(m_sequence.size(), m_list);
        m_code_highlighter.push_into_sequence(2);
    }

    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(1);

    // reaching index - 1
    // cur
    m_list.at(index - 1).set_color_index(2);
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(3);

    // cur->next
    m_list.at(index).set_color_index(7);
    m_list.at(index).set_label(index + 1 == m_list.size() ? "tail/nxt" : "nxt");
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(4);

    // insert between cur and cur->next
    m_list.insert(index, value);
    m_list.at(index).set_color_index(6);
    m_list.at(index).set_label("node");
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(5);

    m_list.at(index - 1).set_color_index(2);
    m_list.at(index + 1).set_color_index(0);
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(6);

    m_list.at(index - 1).set_color_index(0);
    m_list.at(index + 1).set_color_index(7);
    m_list.init_label();
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(7);

    // done
    m_list.at(index - 1).set_color_index(0);
    m_list.at(index - 1).set_label("");
    m_list.at(index).set_color_index(0);
    m_list.at(index).set_label("");
    m_list.at(index + 1).set_color_index(0);
    m_list.at(index + 1).set_label("");
    m_list.init_label();
}

template<typename Con>
void BaseLinkedListScene<Con>::interact_delete() {
    if (m_list.empty()) {
        return;
    }

    auto index_container = m_index_input.extract_values();
    if (index_container.empty()) {
        return;
    }

    int index = index_container.front();

    if (!(0 <= index && index < m_list.size())) {
        return;
    }

    m_sequence.clear();
    m_sequence.insert(m_sequence.size(), m_list);

    if (index == 0) {
        interact_delete_head();
    } else if (index + 1 == m_list.size()) {
        interact_delete_tail();
    } else {
        interact_delete_middle(index);
    }

    m_sequence_controller.set_max_value((int)m_sequence.size());
    m_sequence_controller.set_rerun();
}

template<typename Con>
void BaseLinkedListScene<Con>::interact_delete_head() {
    m_code_highlighter.set_code({
        "Node* temp = head;",
        "head = head->next;",
        "delete temp;",
    });
    m_code_highlighter.push_into_sequence(-1);

    m_list.at(0).set_color_index(4);
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(0);

    m_list.at(0).set_color_index(5);
    m_list.at(0).set_label("");
    if (m_list.size() > 1) {
        m_list.at(1).set_color_index(4);
        m_list.at(1).set_label("head");
    }
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(1);

    m_list.remove(0);
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(2);

    if (m_list.size() > 0) {
        m_list.at(0).set_color_index(0);
    }
}
```

```

template<typename Con>
void BaseLinkedListScene<Con>::interact_delete_tail() {
    m_code_highlighter.set_code({
        "Node* pre = head;",
        "Node* nxt = pre->next;",
        "while (nxt->next != nullptr)",
        "    pre = pre->next, nxt = nxt->next;",
        "",
        "delete nxt;",
        "tail = pre;",
    });
    m_code_highlighter.push_into_sequence(-1);

    m_list.at(0).set_color_index(2);
    m_list.at(0).set_label("head/pre");
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(0);

    m_list.at(1).set_color_index(3);
    if (m_list.size() == 2) {
        m_list.at(1).set_label("tail/nxt");
    } else {
        m_list.at(1).set_label("nxt");
    }
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(1);

    int idx = 0;
    for (; idx + 2 < m_list.size(); ++idx) {
        m_sequence.insert(m_sequence.size(), m_list);
        m_code_highlighter.push_into_sequence(2);

        m_list.at(idx).set_color_index(0);
        if (idx == 0) {
            m_list.at(idx).set_label("head");
        } else {
            m_list.at(idx).set_label("");
        }

        m_list.at(idx + 1).set_color_index(2);
        m_list.at(idx + 1).set_label("pre");
        m_list.at(idx + 2).set_color_index(3);
        if (idx + 3 == m_list.size()) {
            m_list.at(idx + 2).set_label("tail/nxt");
        } else {
            m_list.at(idx + 2).set_label("nxt");
        }

        m_sequence.insert(m_sequence.size(), m_list);
        m_code_highlighter.push_into_sequence(3);
    }

    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(2);

    m_list.at(idx).set_color_index(2);
    m_list.at(idx).set_label("pre");
    m_list.at(idx + 1).set_color_index(3);
    m_list.at(idx + 1).set_label("tail/nxt");
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(4);

    m_list.remove(idx + 1);
    m_list.at(idx).set_label("tail/pre");
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(5);

    m_list.at(idx).set_color_index(4);
    m_list.init_label();
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(6);

    m_list.at(idx).set_color_index(0);
}

template<typename Con>
void BaseLinkedListScene<Con>::interact_delete_middle(int index) {
    m_code_highlighter.set_code({
        "Node* pre = head;",
        "for (i = 0; i < index - 1; i++)",
        "    pre = pre->next;",
        "",
        "Node* node = pre->next;",
        "Node* nxt = node->next;",
        "delete node;",
        "pre->next = nxt;",
    });
    m_code_highlighter.push_into_sequence(-1);

    m_list.at(0).set_color_index(4);
    m_list.at(0).set_label("head/pre");
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(0);

    int idx = 0;
    for (; idx + 1 < index; ++idx) {
        m_list.at(idx).set_color_index(2);
        m_sequence.insert(m_sequence.size(), m_list);
        m_code_highlighter.push_into_sequence(1);

        m_list.at(idx).set_color_index(0);
        m_list.at(idx).set_label("");
        m_list.at(idx + 1).set_color_index(2);
        m_list.init_label();
        m_list.at(idx + 1).set_label("pre");
        m_sequence.insert(m_sequence.size(), m_list);
        m_code_highlighter.push_into_sequence(2);
    }

    m_list.at(idx).set_color_index(2);
    m_list.at(idx).set_label("pre");
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(3);

    m_list.at(idx + 1).set_color_index(5);
    m_list.at(idx + 1).set_label("node");
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(4);

    m_list.at(idx + 2).set_color_index(3);
    if (idx + 3 == m_list.size()) {
        m_list.at(idx + 2).set_label("tail/nxt");
    } else {
        m_list.at(idx + 2).set_label("nxt");
    }
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(5);

    m_list.remove(idx + 1);
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(6);

    m_list.at(idx + 1).set_color_index(7);
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(7);

    m_list.at(idx).set_color_index(0);
    m_list.at(idx).set_label("");
    m_list.at(idx + 1).set_color_index(0);
    m_list.at(idx + 1).set_label("");
}

template<typename Con>
void BaseLinkedListScene<Con>::interact_update() {
    auto index_container = m_index_input.extract_values();
    if (index_container.empty()) {
        return;
    }

    auto value_container = m_text_input.extract_values();
    if (value_container.empty()) {
        return;
    }

    int index = index_container.front();
    int value = value_container.front();

    if (!(0 <= index && index < m_list.size())) {
        return;
    }

    m_code_highlighter.set_code({
        "Node* node = head;",
        "for (i = 0; i < index; i++)",
        "    node = node->next;",
        "",
        "node->value = value;",
    });

    m_sequence.clear();
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(-1);

    m_list.at(0).set_color_index(4);
    m_list.at(0).set_label("head/node");
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(0);

    for (int i = 0; i < index; ++i) {
        m_list.at(i).set_color_index(2);
        m_sequence.insert(m_sequence.size(), m_list);
        m_code_highlighter.push_into_sequence(1);

        m_list.at(i).set_color_index(0);
        m_list.at(i).set_label(i == 0 ? "head" : "");
        m_list.at(i + 1).set_color_index(2);
        m_list.at(i + 1).set_label(i + 2 == m_list.size() ? "tail/node"
                                                            : "node");
        m_sequence.insert(m_sequence.size(), m_list);
        m_code_highlighter.push_into_sequence(2);
    }

    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(1);
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(3);

    m_list.at(index).set_color_index(3);
    m_list.at(index).set_value(value);
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(4);

    m_list.at(index).set_color_index(0);
    m_list.at(index).set_label("");
    m_list.init_label();

    m_sequence_controller.set_max_value((int)m_sequence.size());
    m_sequence_controller.set_rerun();
}

template<typename Con>
void BaseLinkedListScene<Con>::interact_search() {
    auto value_container = m_text_input.extract_values();
    if (value_container.empty()) {
        return;
    }

    int value = value_container.front();
    if (!utils::val_in_range(value)) {
        return;
    }

    m_code_highlighter.set_code({
        "Node* node = head;",
        "while (node != nullptr) {",
        "    if (node->value == value)",
        "        return node;",
        "    node = node->next;",
        "}",
        "return not_found",
    });

    m_sequence.clear();
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(-1);

    m_list.at(0).set_color_index(4);
    m_list.at(0).set_label("head/node");
    m_sequence.insert(m_sequence.size(), m_list);
    m_code_highlighter.push_into_sequence(0);

    std::size_t idx = 0;

    while (idx < m_list.size()) {
        m_list.at(idx).set_color_index(2);
        m_sequence.insert(m_sequence.size(), m_list);
        m_code_highlighter.push_into_sequence(1);

        m_sequence.insert(m_sequence.size(), m_list);
        m_code_highlighter.push_into_sequence(2);
        if (m_list.at(idx).get_value() == value) {
            m_list.at(idx).set_color_index(3);
            m_sequence.insert(m_sequence.size(), m_list);
            m_code_highlighter.push_into_sequence(3);
            m_list.at(idx).set_color_index(0);
            m_list.at(idx).set_label(idx + 1 == m_list.size() ? "tail" : "");
            break;
        }

        m_list.at(idx).set_color_index(0);
        m_list.at(idx).set_label("");
        m_list.init_label();
        ++idx;
        if (idx < m_list.size()) {
            m_list.at(idx).set_color_index(2);
            m_list.at(idx).set_label(idx + 1 == m_list.size() ? "tail/node"
                                                              : "node");
        }
        m_sequence.insert(m_sequence.size(), m_list);
        m_code_highlighter.push_into_sequence(4);
    }

    if (idx >= m_list.size()) {
        m_sequence.insert(m_sequence.size(), m_list);
        m_code_highlighter.push_into_sequence(1);

        m_sequence.insert(m_sequence.size(), m_list);
        m_code_highlighter.push_into_sequence(5);

        m_sequence.insert(m_sequence.size(), m_list);
        m_code_highlighter.push_into_sequence(6);
    }

    m_sequence_controller.set_max_value((int)m_sequence.size());
    m_sequence_controller.set_rerun();
}

}  // namespace scene

#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

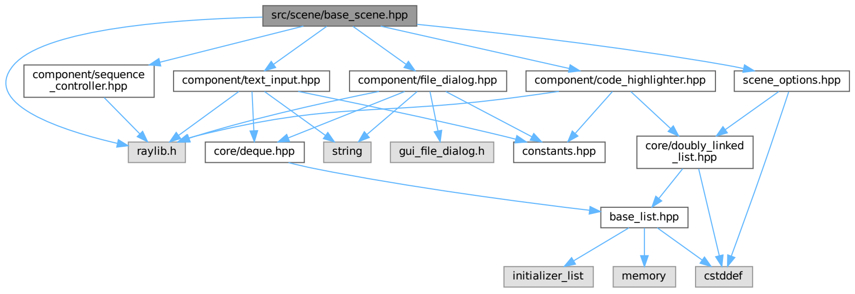
Go to the documentation of this file.

```
#include "base_scene.hpp"

#include <cstring>

#include "constants.hpp"
#include "raygui.h"

namespace scene::internal {

bool BaseScene::render_go_button() const {
    Rectangle shape{options_head, constants::scene_height - button_size.y,
                    button_size.y, button_size.y};
    return GuiButton(shape, "Go");
}

void BaseScene::render_options(SceneOptions& scene_config) {
    (m_edit_mode || m_edit_action) ? GuiLock() : GuiUnlock();

    options_head = 2 * constants::sidebar_width;

    Rectangle mode_button_shape{options_head,
                                constants::scene_height - button_size.y,
                                button_size.x, button_size.y};

    options_head += (button_size.x + head_offset);

    int& mode = scene_config.mode_selection;

    if (GuiDropupBox(mode_button_shape, scene_config.mode_labels, &mode,
                     m_edit_mode)) {
        m_edit_mode ^= 1;
    }

    if (std::strlen(scene_config.action_labels.at(mode)) != 0) {
        Rectangle action_button_shape{options_head,
                                      constants::scene_height - button_size.y,
                                      button_size.x, button_size.y};

        options_head += (button_size.x + head_offset);

        int& action_selection = scene_config.action_selection.at(mode);

        if (GuiDropupBox(action_button_shape,
                         scene_config.action_labels.at(mode), &action_selection,
                         m_edit_action)) {
            m_edit_action ^= 1;
        }

        // scene_config.action_selection.at(mode) = GuiComboBox(
        //     action_button_shape, scene_config.action_labels.at(mode),
        //     scene_config.action_selection.at(mode));
    }

    render_inputs();
}

}  // namespace scene::internal
```

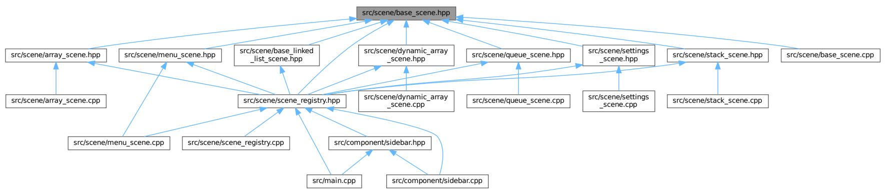
7.75 src/scene/base_scene.hpp File Reference

```
#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:


src/scene/base_scene.hpp
src/scene/array_scene.hpp
src/scene/menu_scene.hpp
src/scene/base_linked_list_scene.hpp
src/scene/dynamic_array_scene.hpp
src/scene/queue_scene.hpp
src/scene/settings_scene.hpp
src/scene/stack_scene.hpp
src/scene/base_scene.cpp
src/scene/array_scene.cpp
src/scene/scene_registry.hpp
src/scene/dynamic_array_scene.cpp
src/scene/queue_scene.cpp
src/scene/settings_scene.cpp
src/scene/stack_scene.cpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.hpp
src/main.cpp
src/component/sidebar.cpp

Classes

- class scene::internal::BaseScene

Namespaces

- namespace scene
- namespace scene::internal

7.76 base_scene.hpp

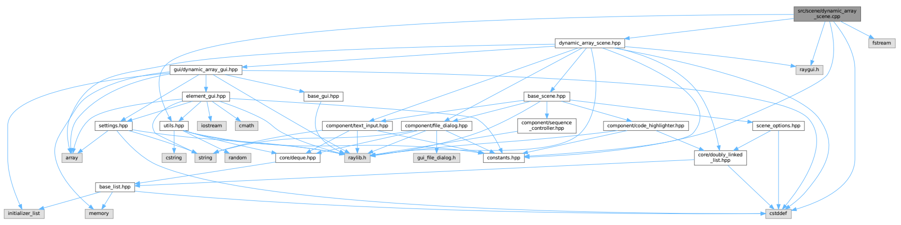
Go to the documentation of this file.

```cpp
#ifndef SCENE_BASE_SCENE_HPP_
#define SCENE_BASE_SCENE_HPP_

#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"

namespace scene::internal {

class BaseScene {
protected:
    static constexpr Vector2 button_size{200, 50};
    static constexpr int head_offset = 20;
    float options_head{};

    virtual bool render_go_button() const;
    virtual void render_options(SceneOptions& scene_config);
    virtual void render_inputs() {}

    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{};

public:
    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() {}
};

}  // namespace scene::internal

#endif  // SCENE_BASE_SCENE_HPP_
```

7.77 src/scene/dynamic_array_scene.cpp File Reference

```
#include "dynamic_array_scene.hpp"
#include <cstddef>
#include <fstream>
#include "constants.hpp"
#include "raygui.h"
#include "utils.hpp"
```

Include dependency graph for dynamic_array_scene.cpp:



Namespaces

- namespace scene

7.78 dynamic_array_scene.cpp

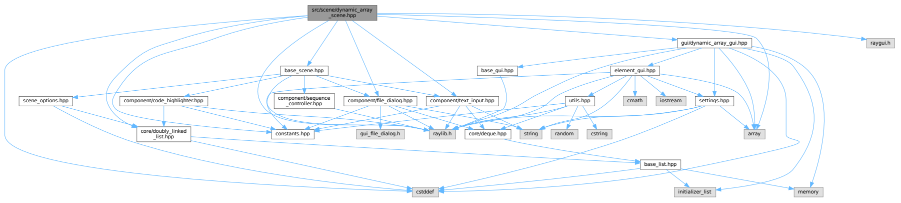
Go to the documentation of this file.

```
#include "dynamic_array_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"

namespace scene {

void DynamicArrayScene::render_inputs() {
    int& mode = scene_options.mode_selection;

    switch (mode) {
        case 0: {
            switch (scene_options.action_selection.at(mode)) {
                case 0:
                    break;
                case 1: {
                    m_text_input.render(options_head, head_offset);
                } break;
                case 2: {
                    m_go = (m_file_dialog.render_head(options_head,
                                                      head_offset) > 0);
                    return;
                } break;
                default:
                    utils::unreachable();
            }
        } break;

        case 1: {
            m_index_input.render(options_head, head_offset);
            m_text_input.render(options_head, head_offset);
        } break;

        case 2:
        case 3: {
            m_text_input.render(options_head, head_offset);
        } break;

        case 4:
            break;

        default:
            utils::unreachable();
    }

    m_go |= render_go_button();
}

void DynamicArrayScene::render() {
    m_sequence_controller.inc_anim_counter();

    int frame_idx = m_sequence_controller.get_anim_frame();
    auto* const frame_ptr = m_sequence.find(frame_idx);
    m_sequence_controller.set_progress_value(frame_idx);

    if (frame_ptr != nullptr) {
        frame_ptr->data.render();
        m_code_highlighter.highlight(frame_idx);
    } else {  // end of sequence
        m_array.render();
        m_sequence_controller.set_run_all(false);
    }

    m_code_highlighter.render();
    m_sequence_controller.render();
    render_options(scene_options);
}

void DynamicArrayScene::interact() {
    if (m_sequence_controller.interact()) {
        m_sequence_controller.reset_anim_counter();
        return;
    }

    m_index_input.set_random_max((int)m_array.size() - 1);

    if (m_text_input.interact() || m_index_input.interact()) {
        return;
    }

    if (!m_go) {
        return;
    }

    int& mode = scene_options.mode_selection;

    switch (mode) {
        case 0: {
            switch (scene_options.action_selection.at(mode)) {
                case 0: {
                    interact_random();
                } break;

                case 1: {
                    interact_import(m_text_input.extract_values());
                } break;

                case 2: {
                    interact_file_import();
                } break;

                default:
                    utils::unreachable();
            }
        } break;

        case 1: {
            interact_update();
        } break;

        case 2: {
            interact_search();
        } break;

        case 3: {
            interact_push();
        } break;

        case 4: {
            interact_pop();
        } break;

        default:
            utils::unreachable();
    }

    m_go = false;
}

void DynamicArrayScene::interact_random() {
    std::size_t size =
        utils::get_random(std::size_t{1}, scene_options.max_size);
    m_array = {};

    for (std::size_t i = 0; i < size; ++i) {
        m_array.push(utils::get_random(constants::min_val, constants::max_val));
    }
}

void DynamicArrayScene::interact_import(core::Deque<int> nums) {
    m_array = {};
    std::size_t i;  // NOLINT

    for (i = 0; i < max_size && !nums.empty(); ++i) {
        m_array.push(nums.front());
        nums.pop_front();
    }
}

void DynamicArrayScene::interact_update() {
    auto index_container = m_index_input.extract_values();
    if (index_container.empty()) {
        return;
    }

    auto value_container = m_text_input.extract_values();
    if (value_container.empty()) {
        return;
    }

    int index = index_container.front();
    int value = value_container.front();

    if (!(0 <= index && index < m_array.size()) ||
        !utils::val_in_range(value)) {
        return;
    }

    m_code_highlighter.set_code({
        "array[index] = value;",
    });

    m_sequence.clear();

    // initial state (before update)
    m_sequence.insert(m_sequence.size(), m_array);
    m_code_highlighter.push_into_sequence(-1);

    // highlight
    m_array.set_color_index(index, 2);
    m_sequence.insert(m_sequence.size(), m_array);
    m_code_highlighter.push_into_sequence(0);

    // update
    m_array[index] = value;
    m_array.set_color_index(index, 3);
    m_sequence.insert(m_sequence.size(), m_array);
    m_code_highlighter.push_into_sequence(0);

    // undo highlight
    m_array.set_color_index(index, 0);

    m_sequence_controller.set_max_value((int)m_sequence.size());
    m_sequence_controller.set_rerun();
}

void DynamicArrayScene::interact_file_import() {
    interact_import(m_file_dialog.extract_values());
}

void DynamicArrayScene::interact_search() {
    auto value_container = m_text_input.extract_values();
    if (value_container.empty()) {
        return;
    }

    int value = value_container.front();
    if (!utils::val_in_range(value)) {
        return;
    }

    m_code_highlighter.set_code({
        "for (i = 0; i < size; i++)",
        "    if (array[i] == value)",
        "        return i;",
        "return not_found",
    });

    m_sequence.clear();
    m_sequence.insert(m_sequence.size(), m_array);
    m_code_highlighter.push_into_sequence(0);

    bool found = false;

    for (std::size_t i = 0; i < m_array.size(); ++i) {
        m_array.set_color_index(i, 3);
        m_sequence.insert(m_sequence.size(), m_array);
        m_code_highlighter.push_into_sequence(1);

        if (m_array[i] == value) {
            found = true;
            m_array.set_color_index(i, 4);
            m_sequence.insert(m_sequence.size(), m_array);
            m_code_highlighter.push_into_sequence(2);
            m_array.set_color_index(i, 0);
            break;
        }

        m_array.set_color_index(i, 0);
        m_sequence.insert(m_sequence.size(), m_array);
        m_code_highlighter.push_into_sequence(0);
    }

    if (!found) {
        m_sequence.insert(m_sequence.size(), m_array);
        m_code_highlighter.push_into_sequence(3);
    }

    m_sequence_controller.set_max_value((int)m_sequence.size());
    m_sequence_controller.set_rerun();
}

void DynamicArrayScene::interact_push() {
    int value = m_text_input.extract_values().front();

    if (m_array.size() >= max_size) {
        return;
    }

    m_code_highlighter.set_code({
        "if (size == capacity)",
        "    capacity *= 2;",
        "array[size] = value;",
        "size++;",
    });

    m_sequence.clear();
    m_sequence.insert(m_sequence.size(), m_array);
    m_code_highlighter.push_into_sequence(-1);

    m_sequence.insert(m_sequence.size(), m_array);
    m_code_highlighter.push_into_sequence(0);

    if (m_array.size() == m_array.capacity()) {
        m_array.realloc(m_array.size() + 1);
        m_sequence.insert(m_sequence.size(), m_array);
        m_code_highlighter.push_into_sequence(1);
    }

    m_array.push(value);
    m_array.set_color_index(m_array.size() - 1, 4);
    m_sequence.insert(m_sequence.size(), m_array);
    m_code_highlighter.push_into_sequence(2);

    m_array.set_color_index(m_array.size() - 1, 0);
    m_sequence.insert(m_sequence.size(), m_array);
    m_code_highlighter.push_into_sequence(3);

    m_sequence_controller.set_max_value((int)m_sequence.size());
    m_sequence_controller.set_rerun();
}

void DynamicArrayScene::interact_pop() {
    if (m_array.size() == 0) {
        return;
    }

    m_code_highlighter.set_code({
        "array[size - 1] = 0;",
        "size--;",
    });

    m_sequence.clear();
    m_sequence.insert(m_sequence.size(), m_array);
    m_code_highlighter.push_into_sequence(-1);

    m_array.set_color_index(m_array.size() - 1, 3);
    m_sequence.insert(m_sequence.size(), m_array);
    m_code_highlighter.push_into_sequence(0);

    m_array.pop();
    m_sequence.insert(m_sequence.size(), m_array);
    m_code_highlighter.push_into_sequence(1);

    m_sequence_controller.set_max_value((int)m_sequence.size());
    m_sequence_controller.set_rerun();
}

}  // namespace scene
```

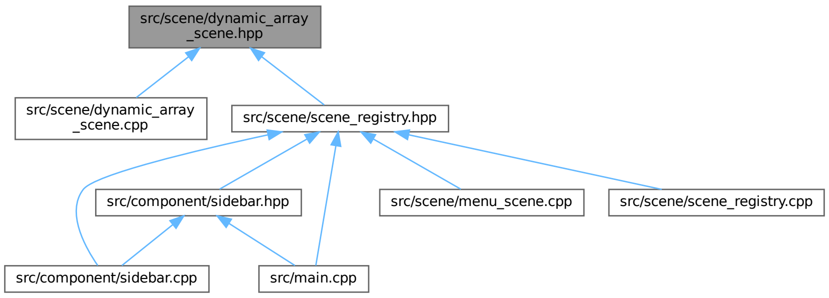
7.79 src/scene/dynamic_array_scene.hpp File Reference

```
#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"
```

Include dependency graph for dynamic_array_scene.hpp:



This graph shows which files directly or indirectly include this file:


src/scene/dynamic_array
_scene.hpp
src/scene/dynamic_array
_scene.cpp
src/scene/scene_registry.hpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class scene::DynamicArrayScene

Namespaces

- namespace scene

7.80 dynamic_array_scene.hpp

Go to the documentation of this file.

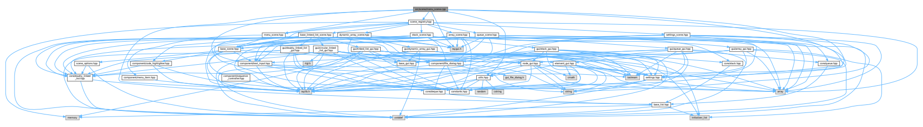
```
#ifndef SCENE_DYNAMIC_ARRAY_SCENE_HPP_
#define SCENE_DYNAMIC_ARRAY_SCENE_HPP_

#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"

namespace scene {

class DynamicArrayScene : public internal::BaseScene {
private:
    static constexpr std::size_t max_size = 8;

    internal::SceneOptions scene_options{
        // max_size
        max_size,

        // mode_labels
        "Mode: Create;"
        "Mode: Update;"
        "Mode: Search;"
        "Mode: Push;"
        "Mode: Pop",

        // mode_selection
        0,

        // action_labels
        {
            // Mode: Create
            "Action: Random;Action: Input;Action: File",

            // Mode: Update
            "",

            // Mode: Search
            "",

            // Mode: Push
            "",

            // Mode: Pop
            "",
        },

        // action_selection
        core::DoublyLinkedList<int>{0, 0, 0, 0, 0},
    };

    using internal::BaseScene::button_size;
    using internal::BaseScene::head_offset;
    using internal::BaseScene::options_head;

    gui::GuiDynamicArray<int> m_array{};
    core::DoublyLinkedList<gui::GuiDynamicArray<int>> m_sequence;

    bool m_go{};
    using internal::BaseScene::m_file_dialog;
    using internal::BaseScene::m_index_input;
    using internal::BaseScene::m_sequence_controller;
    using internal::BaseScene::m_text_input;

    using internal::BaseScene::render_go_button;
    using internal::BaseScene::render_options;
    void render_inputs() override;

    void interact_random();
    void interact_import(core::Deque<int> nums);
    void interact_file_import();
    void interact_update();
    void interact_search();
    void interact_push();
    void interact_pop();

public:
    void render() override;
    void interact() override;
};

}  // namespace scene

#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

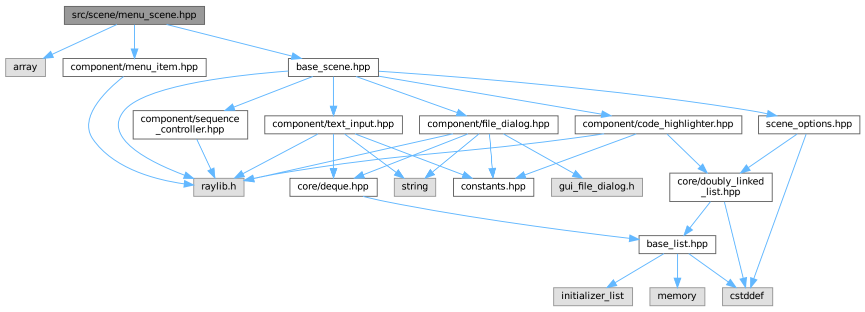
Go to the documentation of this file.

```
#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"

namespace scene {

MenuScene::MenuScene() {
    constexpr int block_width = component::MenuItem::block_width;
    constexpr int block_height = component::MenuItem::block_height;
    constexpr int button_width = component::MenuItem::button_width;
    constexpr int button_height = component::MenuItem::button_height;
    constexpr int gap = 20;

    constexpr int first_row_y =
        constants::scene_height / 16.0F * 5 - block_height / 2.0F;

    // first row
    {
        constexpr int row_width =
            3 * component::MenuItem::block_width + 2 * gap;
        constexpr int row_x = constants::scene_width / 2.0F - row_width / 2.0F;
        constexpr int row_y = first_row_y;

        for (auto i = 0; i < 3; ++i) {
            m_menu_items[i] = component::MenuItem(
                i, labels[i], row_x + i * (block_width + gap), row_y,
                img_paths[i]);
        }
    }

    // second row
    {
        constexpr int row_width = 4 * block_width + 3 * gap;
        constexpr int row_x = constants::scene_width / 2.0F - row_width / 2.0F;
        constexpr int row_y = first_row_y + block_height + gap;

        for (auto i = 3; i < 7; ++i) {
            m_menu_items[i] = component::MenuItem(
                i, labels[i], row_x + (i - 3) * (block_width + gap), row_y,
                img_paths[i]);
        }
    }
}

void MenuScene::render() {
    const Color text_color = utils::adaptive_text_color(
        Settings::get_instance().get_color(Settings::num_color - 1));

    // Menu text
    constexpr int menu_font_size = 60;
    constexpr int menu_font_spacing = 5;

    constexpr const char* menu_text = "CS162 - VisuAlgo.net clone in C++";

    const Vector2 menu_text_size =
        utils::MeasureText(menu_text, menu_font_size, menu_font_spacing);

    const Vector2 menu_text_pos{
        constants::scene_width / 2.0F - menu_text_size.x / 2,
        constants::scene_height / 16.0F - menu_text_size.y / 2};

    utils::DrawText(menu_text, menu_text_pos, text_color, menu_font_size,
                    menu_font_spacing);

    // Sub text
    constexpr int sub_font_size = 30;
    constexpr int sub_font_spacing = 2;

    constexpr const char* sub_text = "By Quang-Truong Nguyen (@jalsol)";

    const Vector2 sub_text_size =
        utils::MeasureText(sub_text, sub_font_size, sub_font_spacing);

    const Vector2 sub_text_pos{
        constants::scene_width / 2.0F - sub_text_size.x / 2,
        menu_text_pos.y + menu_text_size.y / 2 + sub_text_size.y};

    utils::DrawText(sub_text, sub_text_pos, text_color, sub_font_size,
                    sub_font_spacing);

    // Button
    constexpr int block_width = 300;
    constexpr int block_height = 200;
    constexpr int button_width = block_width;
    constexpr int button_height = 50;
    constexpr int gap = 20;
    constexpr int first_row_y =
        constants::scene_height / 16.0F * 5 - block_height / 2.0F;

    for (auto i = 0; i < 7; ++i) {
        m_menu_items[i].render();
    }

    const Rectangle quit_button_shape{
        constants::scene_width / 2.0F - 128,
        constants::scene_height / 16.0F * 15 - block_height / 2.0F, 256, 64};

    m_quit = GuiButton(quit_button_shape, "Quit");

    // Bottom text
    constexpr int bot_font_size = 20;
    constexpr int bot_font_spacing = 2;

    constexpr const char* bot_text =
        "(pls read the src code, i tried so hard for this)";

    const Vector2 bot_text_size =
        utils::MeasureText(bot_text, bot_font_size, bot_font_spacing);

    const Vector2 bot_text_pos{
        constants::scene_width / 2.0F - bot_text_size.x / 2,
        constants::scene_height - 1.5F * bot_text_size.y};

    utils::DrawText(bot_text, bot_text_pos, text_color, bot_font_size,
                    bot_font_spacing);
}

void MenuScene::interact() {
    scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();

    if (m_quit) {
        registry.close_window();
        return;
    }

    for (auto i = 0; i < 7; ++i) {
        if (m_menu_items[i].clicked()) {
            m_next_scene = i;
            m_start = true;
        }
    }

    for (auto i = 0; i < 7; ++i) {
        m_menu_items[i].reset();
    }

    if (m_start) {
        registry.set_scene(m_next_scene);
        m_start = false;
    }
}

} // namespace scene
```

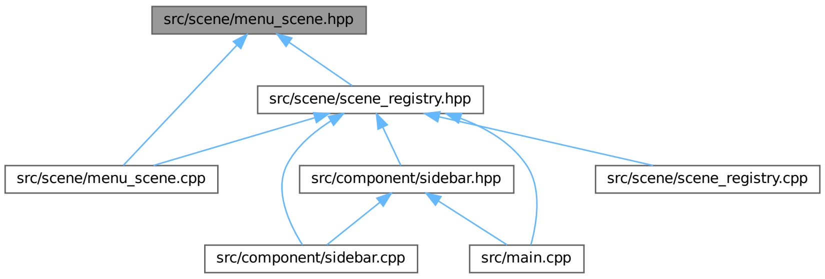
7.83 src/scene/menu_scene.hpp File Reference

```
#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:


src/scene/menu_scene.hpp
src/scene/scene_registry.hpp
src/scene/menu_scene.cpp
src/component/sidebar.hpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class scene::MenuScene

Namespaces

- namespace scene

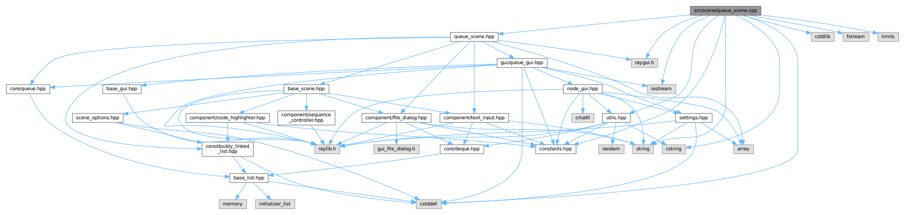
7.84 menu_scene.hpp

Go to the documentation of this file.
```
#ifndef SCENE_MENU_SCENE_HPP_
#define SCENE_MENU_SCENE_HPP_

#include <array>

#include "base_scene.hpp"
#include "component/menu_item.hpp"

namespace scene {

class MenuScene : public internal::BaseScene {
private:
    bool m_start{};
    bool m_quit{};
    int m_next_scene{};

    static constexpr std::array<const char*, 7> labels = {{
        "Array",
        "Dynamic Array",
        "Linked List",
        "Doubly Linked List",
        "Circular Linked List",
        "Stack",
        "Queue",
    }};

    static constexpr std::array<const char*, 7> img_paths = {{
        "data/preview/array.png",
        "data/preview/dynamic_array.png",
        "data/preview/linked_list.png",
        "data/preview/doubly_linked_list.png",
        "data/preview/circular_linked_list.png",
        "data/preview/stack.png",
        "data/preview/queue.png",
    }};

    std::array<component::MenuItem, 7> m_menu_items{};

public:
    MenuScene();
    void render() override;
    void interact() override;
};

}  // namespace scene

#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"
```

Include dependency graph for queue_scene.cpp:



Namespaces

- namespace scene

7.86 queue_scene.cpp

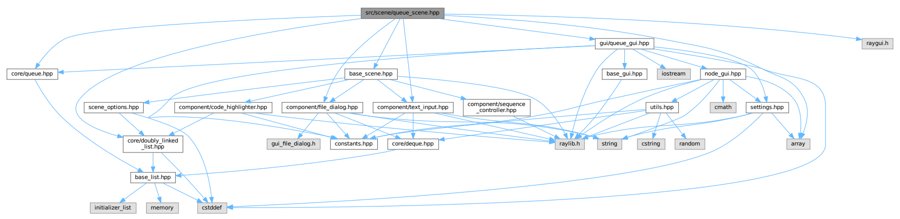
Go to the documentation of this file.

```
#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"

namespace scene {

void QueueScene::render_inputs() {
    int& mode = scene_options.mode_selection;

    switch (mode) {
        case 0: {
            switch (scene_options.action_selection.at(mode)) {
                case 0:
                    break;
                case 1: {
                    m_text_input.render(options_head, head_offset);
                } break;
                case 2: {
                    m_go = (m_file_dialog.render_head(options_head,
                                                      head_offset) > 0);
                    return;
                } break;
                default:
                    utils::unreachable();
            }
        } break;

        case 1: {
            m_text_input.render(options_head, head_offset);
        } break;

        case 2:
            break;
        default:
            utils::unreachable();
    }

    m_go |= render_go_button();
}

void QueueScene::render() {
    m_sequence_controller.inc_anim_counter();

    int frame_idx = m_sequence_controller.get_anim_frame();
    auto* const frame_ptr = m_sequence.find(frame_idx);
    m_sequence_controller.set_progress_value(frame_idx);

    if (frame_ptr != nullptr) {
        frame_ptr->data.render();
        m_code_highlighter.highlight(frame_idx);
    } else {  // end of sequence
        m_queue.render();
        m_sequence_controller.set_run_all(false);
    }

    m_code_highlighter.render();
    m_sequence_controller.render();
    render_options(scene_options);
}

void QueueScene::interact() {
    if (m_sequence_controller.interact()) {
        m_sequence_controller.reset_anim_counter();
        return;
    }

    m_index_input.set_random_max((int)m_queue.size() - 1);

    if (m_text_input.interact() || m_index_input.interact()) {
        return;
    }

    if (!m_go) {
        return;
    }

    int& mode = scene_options.mode_selection;

    switch (mode) {
        case 0: {
            switch (scene_options.action_selection.at(mode)) {
                case 0: {
                    interact_random();
                } break;

                case 1: {
                    interact_import(m_text_input.extract_values());
                } break;

                case 2: {
                    interact_file_import();
                } break;

                default:
                    utils::unreachable();
            }
        } break;

        case 1: {
            interact_push();
        } break;

        case 2: {
            interact_pop();
        } break;

        default:
            utils::unreachable();
    }

    m_go = false;
}

void QueueScene::interact_random() {
    std::size_t size =
        utils::get_random(std::size_t{1}, scene_options.max_size);
    m_queue = gui::GuiQueue<int>();

    for (auto i = 0; i < size; ++i) {
        m_queue.push(utils::get_random(constants::min_val, constants::max_val));
    }
    m_queue.init_label();
}

void QueueScene::interact_import(core::Deque<int> nums) {
    m_sequence.clear();
    m_queue = gui::GuiQueue<int>();

    while (!nums.empty()) {
        if (utils::val_in_range(nums.front())) {
            m_queue.push(nums.front());
        }
        nums.pop_front();
    }
    m_queue.init_label();
}

void QueueScene::interact_file_import() {
    interact_import(m_file_dialog.extract_values());
}

void QueueScene::interact_push() {
    auto value_container = m_text_input.extract_values();
    if (value_container.empty()) {
        return;
    }

    int value = value_container.front();

    if (m_queue.size() >= scene_options.max_size) {
        return;
    }

    m_code_highlighter.set_code({
        "Node* node = new Node(value);",
        "tail->next = node;",
        "tail = tail->next;",
    });

    m_sequence.clear();
    m_sequence.insert(m_sequence.size(), m_queue);
    m_code_highlighter.push_into_sequence(-1);

    m_queue.push(value);
    m_queue.back().set_color_index(6);
    m_sequence.insert(m_sequence.size(), m_queue);
    m_code_highlighter.push_into_sequence(0);

    m_queue.pop_back();
    if (!m_queue.empty()) {
        m_queue.back().set_color_index(4);
    }
    m_queue.push(value);
    m_queue.back().set_color_index(6);
    m_sequence.insert(m_sequence.size(), m_queue);
    m_code_highlighter.push_into_sequence(1);

    m_queue.pop_back();
    if (!m_queue.empty()) {
        m_queue.back().set_color_index(0);
        m_queue.back().set_label("");
    }
    m_queue.push(value);
    m_queue.back().set_color_index(3);
    m_queue.init_label();
    m_sequence.insert(m_sequence.size(), m_queue);
    m_code_highlighter.push_into_sequence(2);

    m_queue.back().set_color_index(0);

    m_sequence_controller.set_max_value((int)m_sequence.size());
    m_sequence_controller.set_rerun();
}

void QueueScene::interact_pop() {
    if (m_queue.empty()) {
        return;
    }

    m_code_highlighter.set_code({
        "Node* temp = head;",
        "head = head->next;",
        "delete temp;",
    });

    m_sequence.clear();
    m_sequence.insert(m_sequence.size(), m_queue);
    m_code_highlighter.push_into_sequence(-1);

    m_queue.front().set_color_index(5);
    m_sequence.insert(m_sequence.size(), m_queue);
    m_code_highlighter.push_into_sequence(0);

    auto old_front = m_queue.front();
    m_queue.pop();

    if (!m_queue.empty()) {
        m_queue.front().set_color_index(3);
        if (m_queue.size() == 1) {
            m_queue.front().set_label("head/tail");
        } else {
            m_queue.front().set_label("head");
        }
    }

    m_queue.push_front(old_front.get_value());
    m_queue.front().set_color_index(5);
    m_sequence.insert(m_sequence.size(), m_queue);
    m_code_highlighter.push_into_sequence(1);

    m_queue.pop();
    m_queue.init_label();
    m_sequence.insert(m_sequence.size(), m_queue);
    m_code_highlighter.push_into_sequence(2);

    if (!m_queue.empty()) {
        m_queue.front().set_color_index(0);
    }

    m_sequence_controller.set_max_value((int)m_sequence.size());
    m_sequence_controller.set_rerun();
}

}  // namespace scene
```

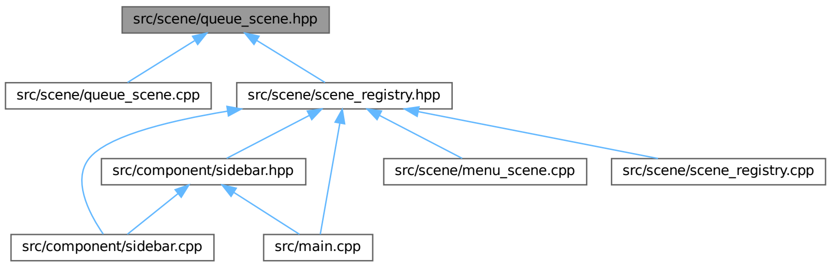
7.87 src/scene/queue_scene.hpp File Reference

```
#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:


src/scene/queue_scene.hpp
src/scene/queue_scene.cpp
src/scene/scene_registry.hpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class scene::QueueScene

Namespaces

- namespace scene

7.88 queue_scene.hpp

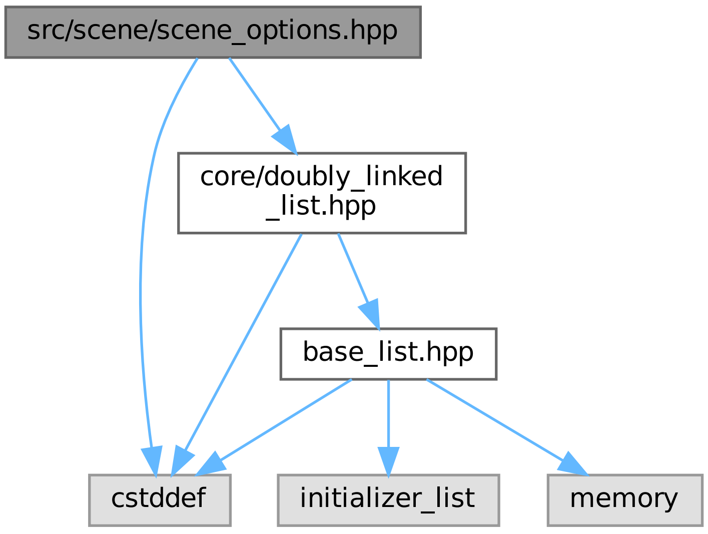
Go to the documentation of this file.

```
#ifndef SCENE_QUEUE_SCENE_HPP_
#define SCENE_QUEUE_SCENE_HPP_

#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"

namespace scene {

class QueueScene : public internal::BaseScene {
private:
    internal::SceneOptions scene_options{
        // max_size
        8,  // NOLINT

        // mode_labels
        "Mode: Create;"
        "Mode: Push;"
        "Mode: Pop",

        // mode_selection
        0,

        // action_labels
        {
            // Mode: Create
            "Action: Random;"
            "Action: Input;"
            "Action: File",

            // Mode: Push
            "",

            // Mode: Pop
            "",
        },

        // action_selection
        core::DoublyLinkedList<int>{0, 0, 0},
    };

    using internal::BaseScene::button_size;
    using internal::BaseScene::head_offset;
    using internal::BaseScene::options_head;

    gui::GuiQueue<int> m_queue{
        gui::GuiNode<int>{1},
        gui::GuiNode<int>{2},
        gui::GuiNode<int>{3},
    };
    core::DoublyLinkedList<gui::GuiQueue<int>> m_sequence;

    bool m_go{};
    using internal::BaseScene::m_code_highlighter;
    using internal::BaseScene::m_file_dialog;
    using internal::BaseScene::m_sequence_controller;
    using internal::BaseScene::m_text_input;

    using internal::BaseScene::render_go_button;
    using internal::BaseScene::render_options;
    void render_inputs() override;

    void interact_random();
    void interact_import(core::Deque<int> nums);
    void interact_file_import();
    void interact_push();
    void interact_pop();

public:
    void render() override;
    void interact() override;
};

}  // namespace scene

#endif  // SCENE_QUEUE_SCENE_HPP_
```

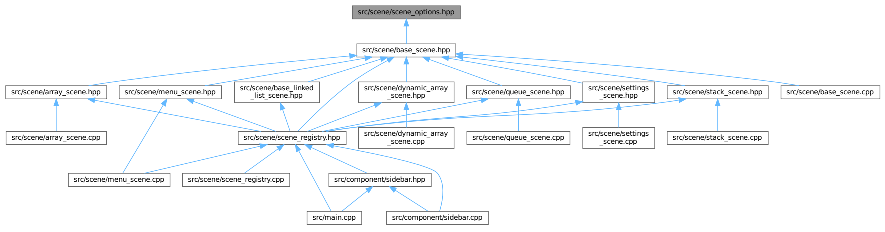
7.89 src/scene/scene_options.hpp File Reference

```
#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:


src/scene/scene_options.hpp
src/scene/base_scene.hpp
src/scene/array_scene.hpp
src/scene/menu_scene.hpp
src/scene/base_linked
_list_scene.hpp
src/scene/dynamic_array
_scene.hpp
src/scene/queue_scene.hpp
src/scene/settings
_scene.hpp
src/scene/stack_scene.hpp
src/scene/base_scene.cpp
src/scene/array_scene.cpp
src/scene/scene_registry.hpp
src/scene/dynamic_array
_scene.cpp
src/scene/queue_scene.cpp
src/scene/settings
_scene.cpp
src/scene/stack_scene.cpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.hpp
src/main.cpp
src/component/sidebar.cpp

Classes

- struct scene::internal::SceneOptions

Namespaces

- namespace scene
- namespace scene::internal

7.90 scene_options.hpp

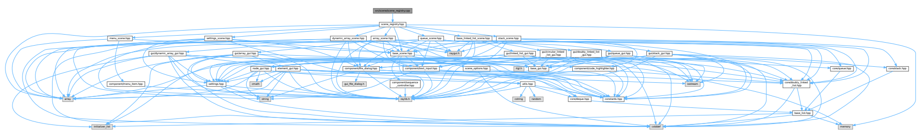
Go to the documentation of this file.

```
#ifndef SCENE_SCENE_OPTIONS_HPP_
#define SCENE_SCENE_OPTIONS_HPP_

#include <cstddef>

#include "core/doubly_linked_list.hpp"

namespace scene::internal {

struct SceneOptions {
    const std::size_t max_size{};
    const char* mode_labels{};
    int mode_selection{};
    core::DoublyLinkedList<const char*> action_labels;
    core::DoublyLinkedList<int> action_selection;
};

}  // namespace scene::internal

#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

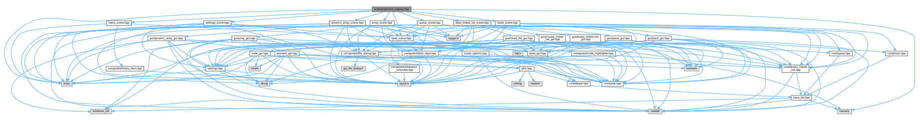
Go to the documentation of this file.

```
#include "scene_registry.hpp"

namespace scene {

SceneRegistry::SceneRegistry() { set_scene(Menu); }

SceneRegistry& SceneRegistry::get_instance() {
    static SceneRegistry registry;
    return registry;
}

void SceneRegistry::set_scene(int scene_type) {
    m_current_scene = scene_type;
    scene_ptr = m_registry.at(scene_type).get();
}

int SceneRegistry::get_scene() const { return m_current_scene; }

void SceneRegistry::render() { scene_ptr->render(); }

void SceneRegistry::interact() { scene_ptr->interact(); }

bool SceneRegistry::should_close() const { return m_should_close; }

void SceneRegistry::close_window() { m_should_close = true; }

}  // namespace scene
```

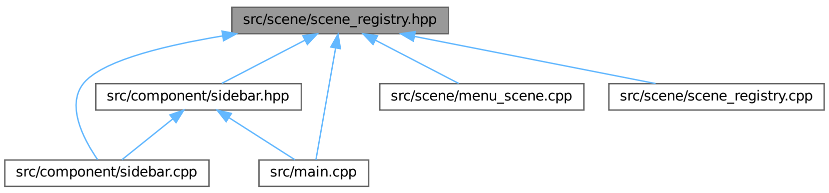
7.93 src/scene/scene_registry.hpp File Reference

```
#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:


src/scene/scene_registry.hpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

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

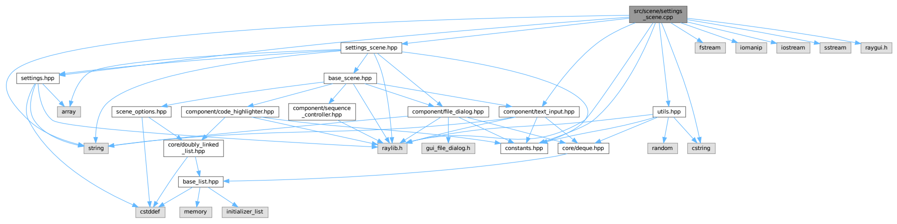
Go to the documentation of this file.

```
#ifndef SCENE_SCENE_REGISTRY_HPP_
#define SCENE_SCENE_REGISTRY_HPP_

#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"

namespace scene {

enum SceneId {
    Array,
    DynamicArray,
    LinkedList,
    DoublyLinkedList,
    CircularLinkedList,
    Stack,
    Queue,
    Menu,
    Settings,
};

class SceneRegistry {
private:
    internal::BaseScene* scene_ptr{};
    SceneRegistry();

    bool m_should_close{};
    int m_current_scene{};

    const std::array<const std::unique_ptr<internal::BaseScene>, 9> m_registry{{
        std::make_unique<ArrayScene>(),
        std::make_unique<DynamicArrayScene>(),
        std::make_unique<LinkedListScene>(),
        std::make_unique<DoublyLinkedListScene>(),
        std::make_unique<CircularLinkedListScene>(),
        std::make_unique<StackScene>(),
        std::make_unique<QueueScene>(),
        std::make_unique<MenuScene>(),
        std::make_unique<SettingsScene>(),
    }};

public:
    SceneRegistry(const SceneRegistry&) = delete;
    SceneRegistry(SceneRegistry&&) = delete;
    SceneRegistry& operator=(const SceneRegistry&) = delete;
    SceneRegistry& operator=(SceneRegistry&&) = delete;
    ~SceneRegistry() = default;

    static SceneRegistry& get_instance();

    void set_scene(int scene_type);
    int get_scene() const;
    void render();
    void interact();
    bool should_close() const;
    void close_window();
};

}  // namespace scene

#endif  // SCENE_SCENE_REGISTRY_HPP_
```

7.95 src/scene/settings_scene.cpp File Reference

```
#include "settings_scene.hpp"
#include <cstring>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <sstream>
#include <string>
#include "component/text_input.hpp"
#include "constants.hpp"
#include "raygui.h"
#include "raylib.h"
#include "settings.hpp"
#include "utils.hpp"
```

Include dependency graph for settings_scene.cpp:



Namespaces

- namespace scene

7.96 settings_scene.cpp

Go to the documentation of this file.

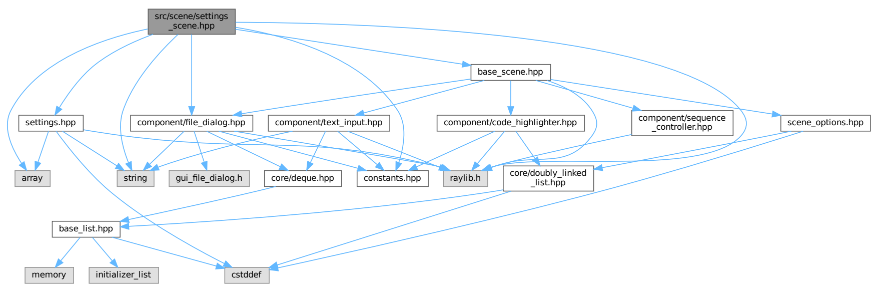
```
#include "settings_scene.hpp"

#include <cstring>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <sstream>
#include <string>
```

```cpp

#include "component/text_input.hpp"
#include "constants.hpp"
#include "raygui.h"
#include "raylib.h"
#include "settings.hpp"
#include "utils.hpp"

namespace scene {

void SettingsScene::open_from_file(const std::string& path) {
    Settings& settings = Settings::get_instance();
    std::ifstream file_in(path, std::ios::binary);

    if (!file_in.is_open()) {
        std::ofstream file_out(path, std::ios::binary);

        for (auto i = 0; i < Settings::num_color; ++i) {
            unsigned value = Settings::default_color.at(i);
            file_out.write(reinterpret_cast<const char*>(&value),
                           sizeof(value));
        }

        file_out.close();

        file_in.close();
        file_in.open(path, std::ios::binary);
    }

    unsigned hex_value;
    for (auto i = 0; i < Settings::num_color; ++i) {
        file_in.read(reinterpret_cast<char*>(&hex_value), sizeof(hex_value));
        settings.get_color(i) = GetColor(hex_value);
    }

    set_buffer();
}

SettingsScene::SettingsScene() {
    open_from_file(constants::default_color_path);
}

void SettingsScene::set_buffer() {
    std::stringstream sstr;

    for (auto i = 0; i < Settings::num_color; ++i) {
        sstr « std::setfill('0') « std::setw(6) « std::hex
             « ((unsigned)ColorToInt(Settings::get_instance().get_color(i)) »
                8);
        std::strncpy(m_buffers.at(i), sstr.str().c_str(), 7);
        sstr.str(std::string());
    }
}

void SettingsScene::set_color() {
    for (auto i = 0; i < Settings::num_color; ++i) {
        Settings::get_instance().get_color(i) =
            utils::color_from_hex(m_buffers.at(i));
    }
}

void SettingsScene::render() {
    Settings& settings = Settings::get_instance();
    constexpr int second_col_x = constants::scene_width / 2 + head_pos.y;
    int second_col_y = 100;
    constexpr int vertical_gap = 30;
    const Color text_color =
        utils::adaptive_text_color(settings.get_color(Settings::num_color - 1));

    auto [head_x, head_y] = head_pos;

    for (auto i = 0; i < m_buffers.size(); ++i) {
        Rectangle input_shape;
        const char* text = nullptr;

        if (i + 1 != m_buffers.size()) {
            input_shape = {(float)head_x, (float)head_y, input_size.x,
                           input_size.y};
            text = TextFormat("Color %d", i + 1);
        } else {
            input_shape = {(float)second_col_x, (float)second_col_y + 400,
                           input_size.x, input_size.y};
            text = "Background color";
        }

        utils::DrawText(text, {(float)input_shape.x, (float)input_shape.y - 25},
                        text_color, 20, 2);
        DrawRectangleRec(input_shape, RAYWHITE);
        if (GuiTextBox(input_shape, m_buffers.at(i), 7, m_edit_mode.at(i))) {
            m_edit_mode.at(i) ^= 1;
        }

        const Rectangle preview_shape{input_shape.x + input_size.x + 10,
                                      input_shape.y, input_size.y,
                                      input_size.y};

        DrawRectangleRec(preview_shape, settings.get_color(i));

        if (m_selected == i) {
            DrawRectangleLinesEx(preview_shape, 3, settings.get_color(5));
        } else {
            DrawRectangleLinesEx(preview_shape, 2, text_color);
        }

        head_y += input_size.y + vertical_gap;
    }

    {
        Color& color = settings.get_color(m_selected);
        auto new_color = GuiColorPicker({second_col_x, (float)second_col_y,
                                         4 * input_size.y, 4 * input_size.y},
                                        nullptr, color);

        if (ColorToInt(color) != ColorToInt(new_color)) {
            color = new_color;
            set_buffer();
        }
    }

    {
        second_col_y += 4 * input_size.y;
        utils::DrawText("Import config",
                        {second_col_x + 10, (float)second_col_y}, text_color,
                        20, 2);
        m_open = m_open_file.render(second_col_x, (float)second_col_y + 25);
    }

    {
        second_col_y += component::FileDialog::size.y + vertical_gap;
        utils::DrawText("Export config",
                        {second_col_x + 10, (float)second_col_y}, text_color,
                        20, 2);
        m_save = m_save_file.render(second_col_x, (float)second_col_y + 25);
    }
}

void SettingsScene::interact() {
    if (m_open > 0) {
        open_from_file(m_open_file.get_path());
        return;
    }

    if (m_save > 0) {
        Settings::get_instance().save_to_file(m_save_file.get_path());
        return;
    }

    const Vector2 mouse = GetMousePosition();
    const bool left_clicked = IsMouseButtonPressed(MOUSE_LEFT_BUTTON);
    auto [head_x, head_y] = head_pos;

    for (auto i = 0; i < m_buffers.size(); ++i) {
        const Rectangle input_shape{(float)head_x, (float)head_y, input_size.x,
                                    input_size.y};
        const Rectangle preview_shape{input_shape.x + input_size.x + 10,
                                      input_shape.y, input_size.y,
                                      input_size.y};

        if (m_edit_mode.at(i)) {
            m_selected = i;
        }
    }

    set_color();
}

}  // namespace scene
```

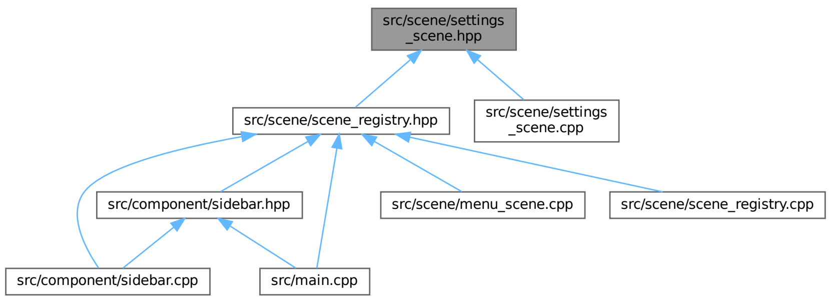
7.97 src/scene/settings_scene.hpp File Reference

```
#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:


src/scene/settings_scene.hpp
src/scene/scene_registry.hpp
src/scene/settings_scene.cpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class scene::SettingsScene

Namespaces

- namespace scene

7.98 settings_scene.hpp

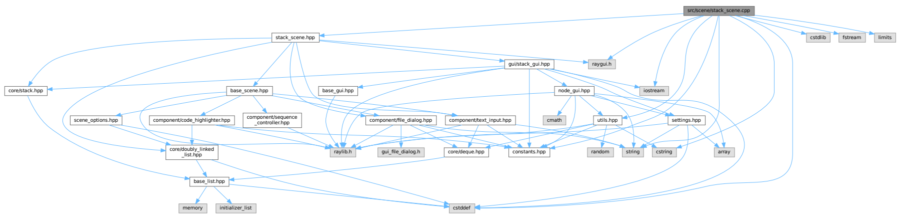
Go to the documentation of this file.

```
#ifndef SCENE_SETTINGS_SCENE_HPP_
#define SCENE_SETTINGS_SCENE_HPP_

#include <array>
#include <constants.hpp>
#include <string>

#include "base_scene.hpp"
#include "component/file_dialog.hpp"
#include "raylib.h"
#include "settings.hpp"

namespace scene {

class SettingsScene : public internal::BaseScene {
private:
    static constexpr Vector2 input_size{200, 50};
    static constexpr Vector2 head_pos{400, 70};
    std::array<char[7], Settings::num_color> m_buffers{};
    std::array<bool, Settings::num_color> m_edit_mode{};

    int m_selected{};

    component::FileDialog m_open_file;
    component::FileDialog m_save_file{3, "Save file...", "Save file"};
    int m_open{};
    int m_save{};

    void set_buffer();
    void set_color();
    void open_from_file(const std::string& path);

public:
    SettingsScene();

    void render() override;
    void interact() override;
};

}  // namespace scene

#endif  // SCENE_SETTINGS_SCENE_HPP_
```

7.99 src/scene/stack_scene.cpp File Reference

```
#include "stack_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"
```

Include dependency graph for stack_scene.cpp:



Namespaces

- namespace scene

7.100 stack_scene.cpp

Go to the documentation of this file.

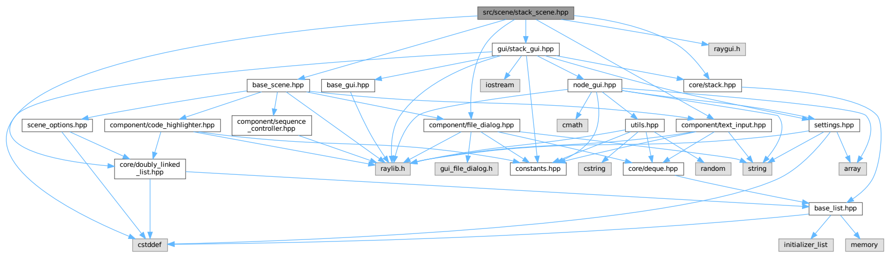
```
#include "stack_scene.hpp"

#include <cstddef>
#include <cstdlib>
#include <cstring>
#include <fstream>
#include <iostream>
#include <limits>
#include <string>

#include "constants.hpp"
#include "raygui.h"
#include "utils.hpp"

namespace scene {

void StackScene::render() {
    m_sequence_controller.inc_anim_counter();

    int frame_idx = m_sequence_controller.get_anim_frame();
    auto* const frame_ptr = m_sequence.find(frame_idx);
    m_sequence_controller.set_progress_value(frame_idx);

    if (frame_ptr != nullptr) {
        frame_ptr->data.render();
        m_code_highlighter.highlight(frame_idx);
    } else {  // end of sequence
        m_stack.render();
        m_sequence_controller.set_run_all(false);
    }

    m_code_highlighter.render();
    m_sequence_controller.render();
    render_options(scene_options);
}

void StackScene::render_inputs() {
    int& mode = scene_options.mode_selection;

    switch (mode) {
        case 0: {
            switch (scene_options.action_selection.at(mode)) {
                case 0:
                    break;
                case 1: {
                    m_text_input.render(options_head, head_offset);
                } break;
                case 2: {
                    m_go = (m_file_dialog.render_head(options_head,
                                                      head_offset) > 0);
                    return;
                } break;
                default:
                    utils::unreachable();
            }
        } break;

        case 1: {
            m_text_input.render(options_head, head_offset);
        } break;

        case 2:
            break;
        default:
            utils::unreachable();
    }

    m_go |= render_go_button();
}

void StackScene::interact() {
    if (m_sequence_controller.interact()) {
        m_sequence_controller.reset_anim_counter();
        return;
    }

    m_index_input.set_random_max((int)m_stack.size() - 1);
    if (m_text_input.interact() || m_index_input.interact()) {
        return;
    }

    if (!m_go) {
        return;
    }

    int& mode = scene_options.mode_selection;

    switch (mode) {
        case 0: {
            switch (scene_options.action_selection.at(mode)) {
                case 0: {
                    interact_random();
                } break;

                case 1: {
                    interact_import(m_text_input.extract_values());
                } break;

                case 2: {
                    interact_file_import();
                } break;

                default:
                    utils::unreachable();
            }
        } break;

        case 1: {
            interact_push();
        } break;

        case 2: {
            interact_pop();
        } break;

        default:
            utils::unreachable();
    }

    m_go = false;
}

void StackScene::interact_random() {
    std::size_t size =
        utils::get_random(std::size_t{1}, scene_options.max_size);
    m_stack = gui::GuiStack<int>();

    for (auto i = 0; i < size; ++i) {
        m_stack.push(utils::get_random(constants::min_val, constants::max_val));
    }
    m_stack.init_label();
}

void StackScene::interact_import(core::Deque<int> nums) {
    m_sequence.clear();
    m_stack = gui::GuiStack<int>();

    while (!nums.empty()) {
        if (utils::val_in_range(nums.back())) {
            m_stack.push(nums.back());
        }
        nums.pop_back();
    }
    m_stack.init_label();
}

void StackScene::interact_push() {
    auto value_container = m_text_input.extract_values();
    if (value_container.empty()) {
        return;
    }

    int value = value_container.front();

    if (m_stack.size() >= scene_options.max_size) {
        return;
    }

    m_code_highlighter.set_code({
        "Node* node = new Node(value);",
        "node->next = head;",
        "head = node;",
    });

    m_sequence.clear();
    m_sequence.insert(m_sequence.size(), m_stack);
    m_code_highlighter.push_into_sequence(-1);

    m_stack.push(value);
    m_stack.top().set_color_index(6);
    m_sequence.insert(m_sequence.size(), m_stack);
    m_code_highlighter.push_into_sequence(0);

    m_stack.pop();
    if (!m_stack.empty()) {
        m_stack.top().set_color_index(4);
    }
    m_stack.push(value);
    m_stack.top().set_color_index(6);
    m_sequence.insert(m_sequence.size(), m_stack);
    m_code_highlighter.push_into_sequence(1);

    m_stack.pop();
    if (!m_stack.empty()) {
        m_stack.top().set_color_index(0);
        m_stack.top().set_label("");
    }
    m_stack.push(value);
    m_stack.top().set_color_index(3);
    m_stack.init_label();
    m_sequence.insert(m_sequence.size(), m_stack);
    m_code_highlighter.push_into_sequence(2);

    m_stack.top().set_color_index(0);

    m_sequence_controller.set_max_value((int)m_sequence.size());
    m_sequence_controller.set_rerun();
}

void StackScene::interact_pop() {
    if (m_stack.empty()) {
        return;
    }

    m_code_highlighter.set_code({
        "Node* temp = head;",
        "head = head->next;",
        "delete temp;",
    });

    m_sequence.clear();
    m_sequence.insert(m_sequence.size(), m_stack);
    m_code_highlighter.push_into_sequence(-1);

    m_stack.top().set_color_index(5);
    m_sequence.insert(m_sequence.size(), m_stack);
    m_code_highlighter.push_into_sequence(0);

    auto old_top = m_stack.top();
    m_stack.pop();

    if (!m_stack.empty()) {
        m_stack.top().set_color_index(3);
        m_stack.top().set_label("head");
    }

    m_stack.push(old_top.get_value());
    m_stack.top().set_color_index(5);
    m_sequence.insert(m_sequence.size(), m_stack);
    m_code_highlighter.push_into_sequence(1);

    m_stack.pop();
    m_sequence.insert(m_sequence.size(), m_stack);
    m_code_highlighter.push_into_sequence(2);

    if (!m_stack.empty()) {
        m_stack.top().set_color_index(0);
    }

    m_sequence_controller.set_max_value((int)m_sequence.size());
    m_sequence_controller.set_rerun();
}

void StackScene::interact_file_import() {
    interact_import(m_file_dialog.extract_values());
}

```

```
00248 }  // namespace scene
```

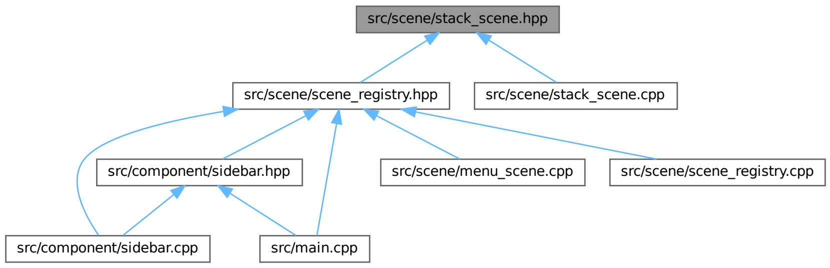
7.101 src/scene/stack_scene.hpp File Reference

```
#include "base_scene.hpp"
#include "component/file_dialog.hpp"
#include "component/text_input.hpp"
#include "core/doubly_linked_list.hpp"
#include "core/stack.hpp"
#include "gui/stack_gui.hpp"
#include "raygui.h"
```

Include dependency graph for stack_scene.hpp:



This graph shows which files directly or indirectly include this file:


src/scene/stack_scene.hpp
src/scene/scene_registry.hpp
src/scene/stack_scene.cpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class scene::StackScene

Namespaces

- namespace scene

7.102 stack_scene.hpp

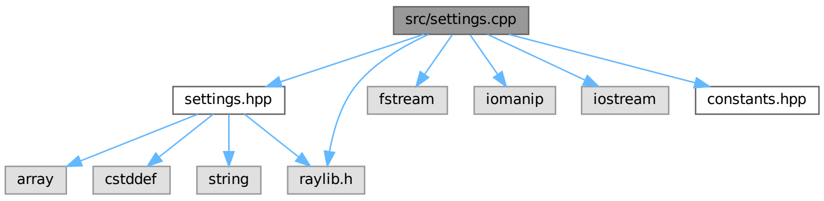
Go to the documentation of this file.

```
#ifndef SCENE_STACK_SCENE_HPP_
#define SCENE_STACK_SCENE_HPP_

#include "base_scene.hpp"
#include "component/file_dialog.hpp"
#include "component/text_input.hpp"
#include "core/doubly_linked_list.hpp"
#include "core/stack.hpp"
#include "gui/stack_gui.hpp"
#include "raygui.h"

namespace scene {

class StackScene : public internal::BaseScene {
private:
    internal::SceneOptions scene_options{
        // max_size
        8,  // NOLINT

        // mode_labels
        "Mode: Create;"
        "Mode: Push;"
        "Mode: Pop",

        // mode_selection
        0,

        // action_labels
        {
            // Mode: Create
            "Action: Random;"
            "Action: Input;"
            "Action: File",

            // Mode: Push
            "",

            // Mode: Pop
            "",
        },

        // action_selection
        core::DoublyLinkedList<int>{0, 0, 0},
    };

    using internal::BaseScene::button_size;
    using internal::BaseScene::head_offset;
    using internal::BaseScene::options_head;

    gui::GuiStack<int> m_stack{
        gui::GuiNode<int>{1},
        gui::GuiNode<int>{2},
        gui::GuiNode<int>{3},
    };
    core::DoublyLinkedList<gui::GuiStack<int>> m_sequence;

    bool m_go{};
    using internal::BaseScene::m_code_highlighter;
    using internal::BaseScene::m_file_dialog;
    using internal::BaseScene::m_sequence_controller;
    using internal::BaseScene::m_text_input;

    using internal::BaseScene::render_go_button;
    using internal::BaseScene::render_options;
    void render_inputs() override;

    void interact_random();
    void interact_import(core::Deque<int> nums);
    void interact_push();
    void interact_pop();
    void interact_file_import();

public:
    void render() override;
    void interact() override;
};

}  // namespace scene

#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.

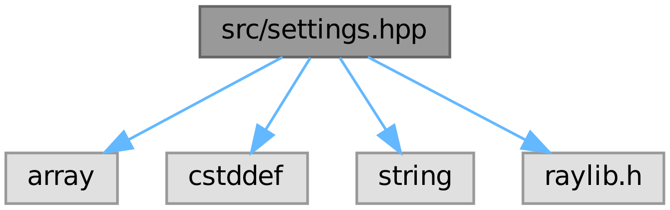
```
#include "settings.hpp"

#include <fstream>
#include <iomanip>
#include <iostream>

#include "constants.hpp"
#include "raylib.h"

Settings& Settings::get_instance() {
    static Settings settings;
    return settings;
}

void Settings::save_to_file(const std::string& path) {
    std::ofstream file_out(path, std::ios::binary);

    for (auto i = 0; i < num_color; ++i) {
        unsigned value = ColorToInt(m_colors.at(i));
        file_out.write(reinterpret_cast<const char*>(&value), sizeof(value));
    }
}

Settings::~Settings() { save_to_file(constants::default_color_path); }

Color& Settings::get_color(std::size_t index) { return m_colors.at(index); }

Color Settings::get_color(std::size_t index) const {
    return m_colors.at(index);
}
```

7.105 src/settings.hpp File Reference

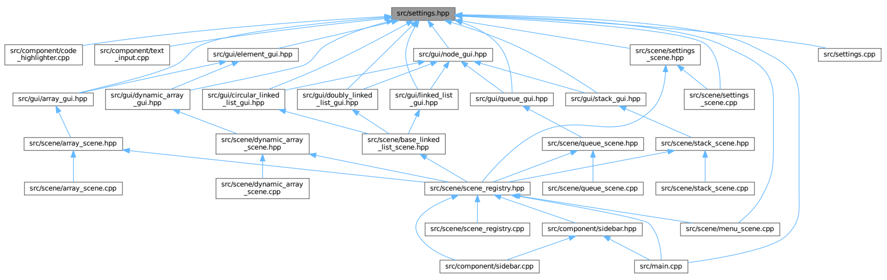
```
#include <array>
#include <cstddef>
#include <string>
```

```
#include "raylib.h"
```

Include dependency graph for settings.hpp:



This graph shows which files directly or indirectly include this file:


src/settings.hpp
src/component/code_highlighter.cpp
src/component/text_input.cpp
src/gui/element_gui.hpp
src/gui/node_gui.hpp
src/scene/settings_scene.hpp
src/settings.cpp
src/gui/array_gui.hpp
src/gui/dynamic_array_gui.hpp
src/gui/circular_linked_list_gui.hpp
src/gui/doubly_linked_list_gui.hpp
src/gui/linked_list_gui.hpp
src/gui/queue_gui.hpp
src/gui/stack_gui.hpp
src/scene/settings_scene.cpp
src/scene/array_scene.hpp
src/scene/dynamic_array_scene.hpp
src/scene/base_linked_list_scene.hpp
src/scene/queue_scene.hpp
src/scene/stack_scene.hpp
src/scene/array_scene.cpp
src/scene/dynamic_array_scene.cpp
src/scene/scene_registry.hpp
src/scene/queue_scene.cpp
src/scene/stack_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.hpp
src/scene/menu_scene.cpp
src/component/sidebar.cpp
src/main.cpp

Classes

- class Settings

7.106 settings.hpp

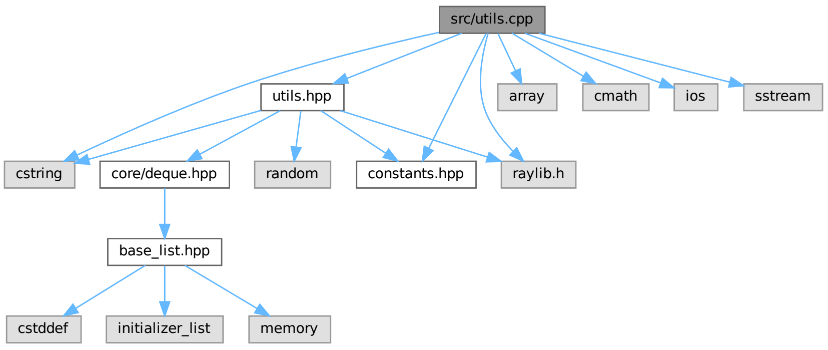
Go to the documentation of this file.

```
#ifndef SETTINGS_HPP_
#define SETTINGS_HPP_

#include <array>
#include <cstddef>
#include <string>

#include "raylib.h"

class Settings {
public:
    static constexpr int num_color = 9;
    static constexpr std::array<unsigned, num_color> default_color{{
        0x00000000,
        0x82828200,
        0xffa10000,
        0x00e43000,
        0x873cbe00,
        0xe6293700,
        0x0079f100,
        0xff6dc200,
        0xf5f5f500,
    }};

private:
    Settings() = default;
    std::array<Color, num_color> m_colors{};

public:
    Settings(const Settings&) = delete;
    Settings(Settings&&) = delete;
    Settings& operator=(const Settings&) = delete;
    Settings& operator=(Settings&&) = delete;
    ~Settings();

    static Settings& get_instance();

    Color& get_color(std::size_t index);
    Color get_color(std::size_t index) const;

    void save_to_file(const std::string& path);
};

#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

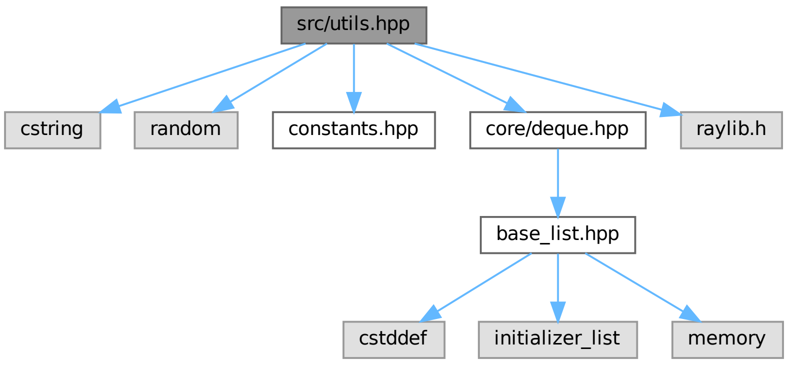
Go to the documentation of this file.

```
#include "utils.hpp"

#include <array>
#include <cmath>
#include <cstring>
#include <ios>
#include <sstream>

#include "constants.hpp"
#include "raylib.h"

namespace utils {

void DrawText(const char* text, Vector2 pos, Color color, float font_size,
              float spacing) {
    static Font font = LoadFontEx("data/open_sans.ttf",
                                  constants::default_font_size, nullptr, 0);

    Vector2 pos_vec{static_cast<float>(pos.x), static_cast<float>(pos.y)};
    DrawTextEx(font, text, pos_vec, font_size, spacing, color);
}

Vector2 MeasureText(const char* text, float font_size, float spacing) {
    static Font font = LoadFontEx("data/open_sans.ttf",
                                  constants::default_font_size, nullptr, 0);

    return MeasureTextEx(font, text, font_size, spacing);
}

core::Deque<int> str_extract_data(
    char str[constants::text_buffer_size]) {  // NOLINT
    char str_copy[constants::text_buffer_size];
    strncpy(str_copy, str, constants::text_buffer_size);

    char* save_ptr = nullptr;
    char* token = utils::strtok(str_copy, ",", &save_ptr);

    if (token == nullptr) {
        return {};
    }

    core::Deque<int> ret;

    constexpr int base = 10;
    int num = static_cast<int>(std::strtol(token, nullptr, base));
    ret.push_back(num);

    while (true) {
        token = utils::strtok(nullptr, ",", &save_ptr);
        if (token == nullptr) {
            break;
        }

        num = static_cast<int>(std::strtol(token, nullptr, base));
        ret.push_back(num);
    }

    return ret;
}

bool val_in_range(int num) {
    return constants::min_val <= num && num <= constants::max_val;
}

void unreachable() {
#if defined(_MSC_VER)
    __assume(0);
#else
    __builtin_unreachable();
#endif
}

char* strtok(char* str, const char* delim, char** save_ptr) {
    return
#if defined(_MSC_VER)
        strtok_s(str, delim, save_ptr);
#else
        strtok_r(str, delim, save_ptr);
#endif
}

Color color_from_hex(const std::string& hex) {
    std::stringstream stream(hex + "ff");
    unsigned int value;
    stream » std::hex » value;
    return GetColor(value);
}

// https://stackoverflow.com/a/3943023
Color adaptive_text_color(Color bg_color) {
    constexpr std::array<float, 3> threshold{{0.2126, 0.7152, 0.0722}};
    const std::array<int, 3> colors = {{bg_color.r, bg_color.g, bg_color.b}};
    float sum = 0;

    for (auto i = 0; i < 3; ++i) {
        float value = (float)colors.at(i) / 255.0F;
        if (value <= 0.04045) {
            value /= 12.92;
        } else {
            value = std::pow(((value + 0.055) / 1.055), 2.4);
        }

        sum += value;
    }

    return (sum > 0.179) ? BLACK : WHITE;
}

}  // namespace utils
```

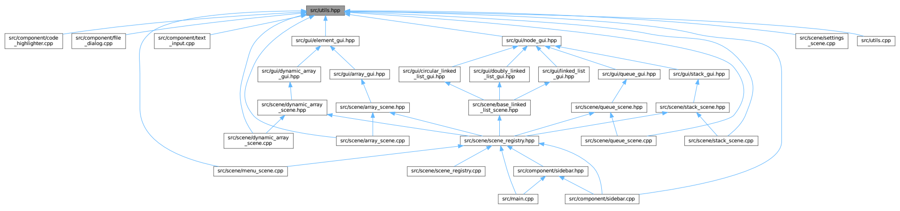
7.109 src/utils.hpp File Reference

```
#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:


src/utils.hpp
src/component/code_highlighter.cpp
src/component/file_dialog.cpp
src/component/text_input.cpp
src/gui/element_gui.hpp
src/gui/node_gui.hpp
src/scene/settings_scene.cpp
src/utils.cpp
src/gui/dynamic_array_gui.hpp
src/gui/array_gui.hpp
src/gui/circular_linked_list_gui.hpp
src/gui/doubly_linked_list_gui.hpp
src/gui/linked_list_gui.hpp
src/gui/queue_gui.hpp
src/gui/stack_gui.hpp
src/scene/dynamic_array_scene.hpp
src/scene/array_scene.hpp
src/scene/base_linked_list_scene.hpp
src/scene/queue_scene.hpp
src/scene/stack_scene.hpp
src/scene/dynamic_array_scene.cpp
src/scene/array_scene.cpp
src/scene/scene_registry.hpp
src/scene/queue_scene.cpp
src/scene/stack_scene.cpp
src/scene/menu_scene.cpp
src/scene/scene_registry.cpp
src/component/sidebar.hpp
src/main.cpp
src/component/sidebar.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)
- 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.

```
#ifndef UTILS_HPP_
#define UTILS_HPP_

#include <cstring>
#include <random>

#include "constants.hpp"
#include "core/deque.hpp"
#include "raylib.h"

namespace utils {

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

Vector2 MeasureText(const char* text, float font_size, float spacing);

template<typename T>
T get_random(T low, T high) {
    if (low > high) {
        return low;
    }

    static std::random_device ran_dev;
    static std::mt19937 prng(ran_dev());
    std::uniform_int_distribution<T> dist{low, high};
    return dist(prng);
}

core::Deque<int> str_extract_data(
    char str[constants::text_buffer_size]);  // NOLINT

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);

}  // namespace utils

#endif  // UTILS_HPP_
```

Index