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 RandomTextInput
- 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 97 of file base_linked_list_scene.hpp.

5.6.1.2 DoublyLinkedListScene

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

Definition at line 95 of file base_linked_list_scene.hpp.

5.6.1.3 LinkedListScene

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

Definition at line 94 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)

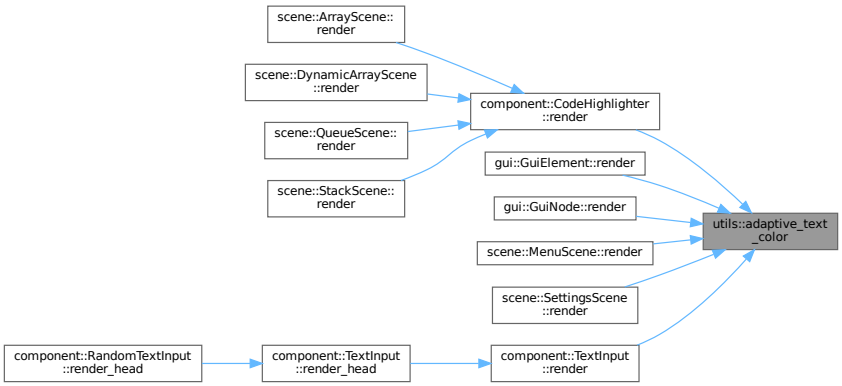
5.8.1 Function Documentation

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:



5.8.1.2 color_from_hex()

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

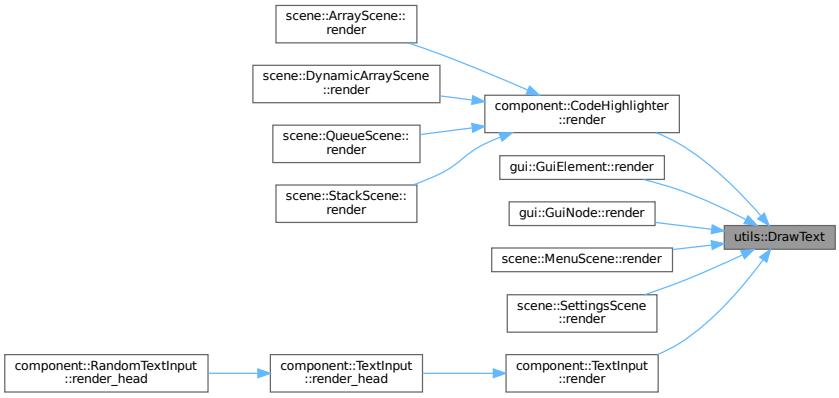
Definition at line 82 of file utils.cpp.

5.8.1.3 DrawText()

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

Definition at line 14 of file utils.cpp.

Here is the caller graph for this function:

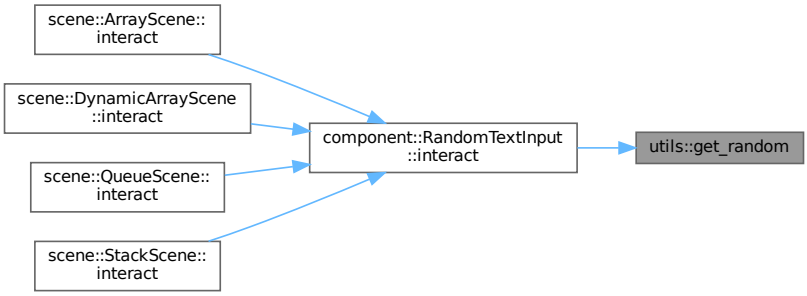


5.8.1.4 get_random()

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

Definition at line 19 of file utils.hpp.

Here is the caller graph for this function:

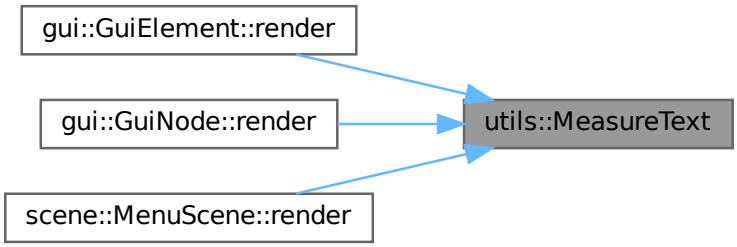


5.8.1.5 MeasureText()

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

Definition at line 23 of file utils.cpp.

Here is the caller graph for this function:

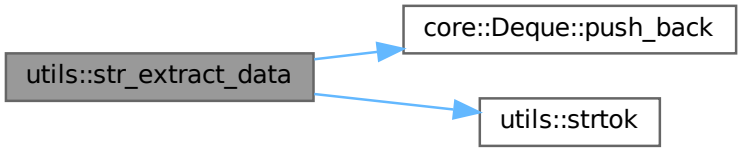


5.8.1.6 str_extract_data()

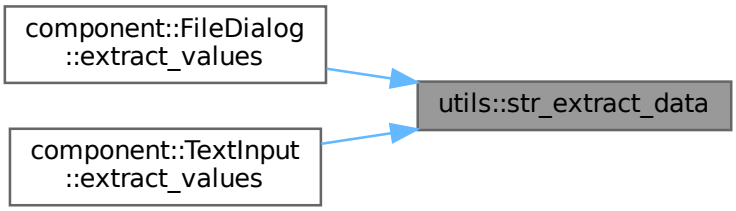
```
core::Deque< int > utils::str_extract_data (
            char str[constants::text_buffer_size] )
```

Definition at line 30 of file utils.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

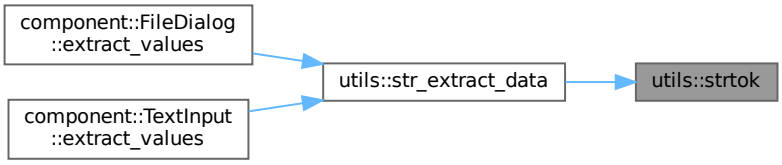


5.8.1.7 strtok()

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

Definition at line 73 of file utils.cpp.

Here is the caller graph for this function:

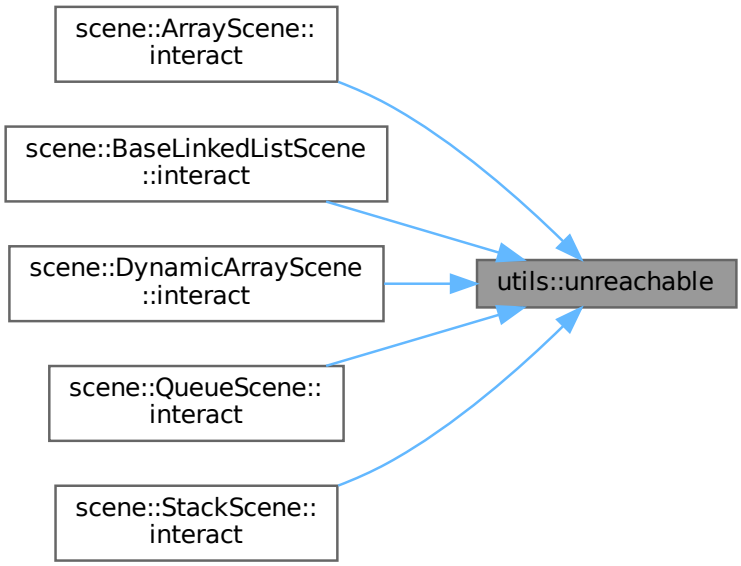


5.8.1.8 unreachable()

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

Definition at line 65 of file utils.cpp.

Here is the caller graph for this function:



5.8.1.9 val_in_range()

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

Definition at line 61 of file utils.cpp.

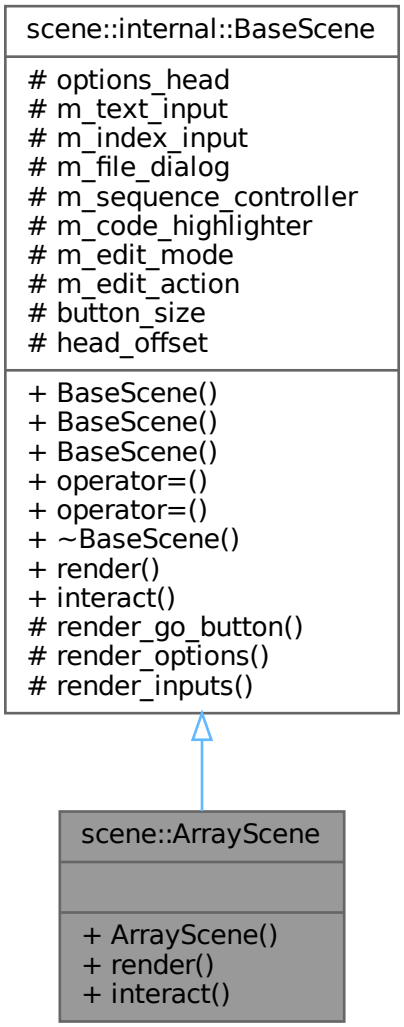
Chapter 6

Class Documentation

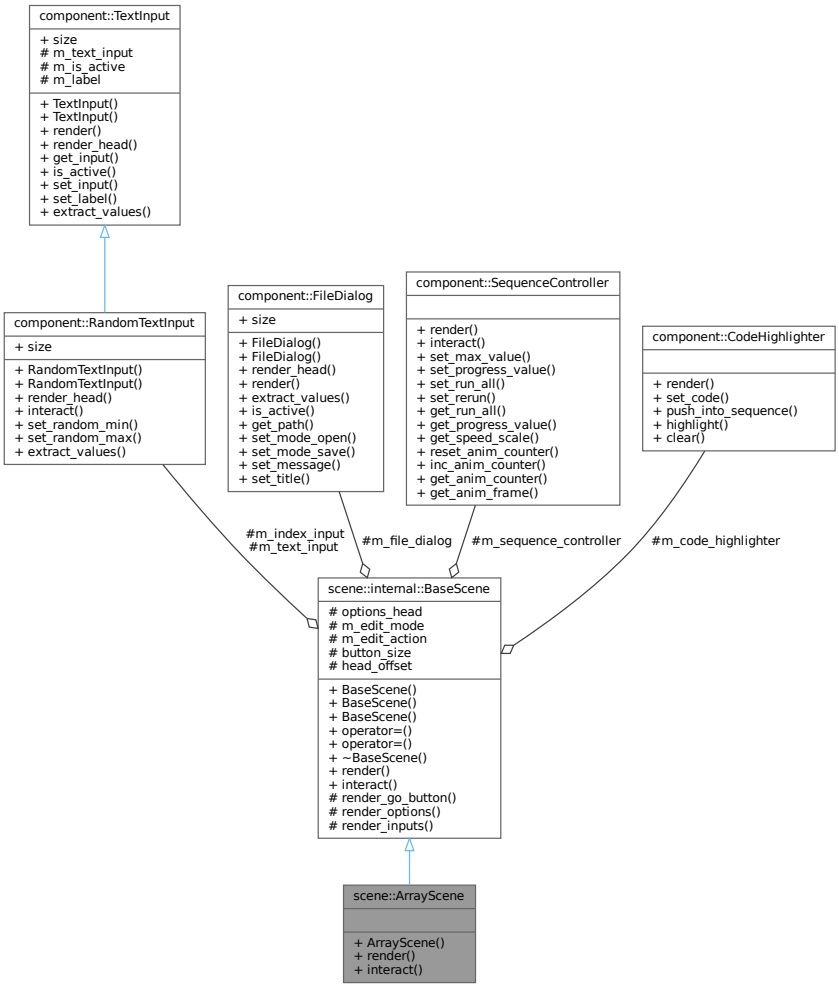
6.1 scene::ArrayScene Class Reference

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

Inheritance diagram for scene::ArrayScene:



Collaboration diagram for scene::ArrayScene:



Public Member Functions

- ArrayScene ()
- 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::RandomTextInput m_text_input {"value"}
- component::RandomTextInput 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 17 of file array_scene.hpp.

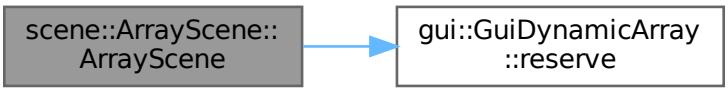
6.1.2 Constructor & Destructor Documentation

6.1.2.1 ArrayScene()

```
scene::ArrayScene::ArrayScene ( )
```

Definition at line 17 of file array_scene.cpp.

Here is the call graph for this function:



6.1.3 Member Function Documentation

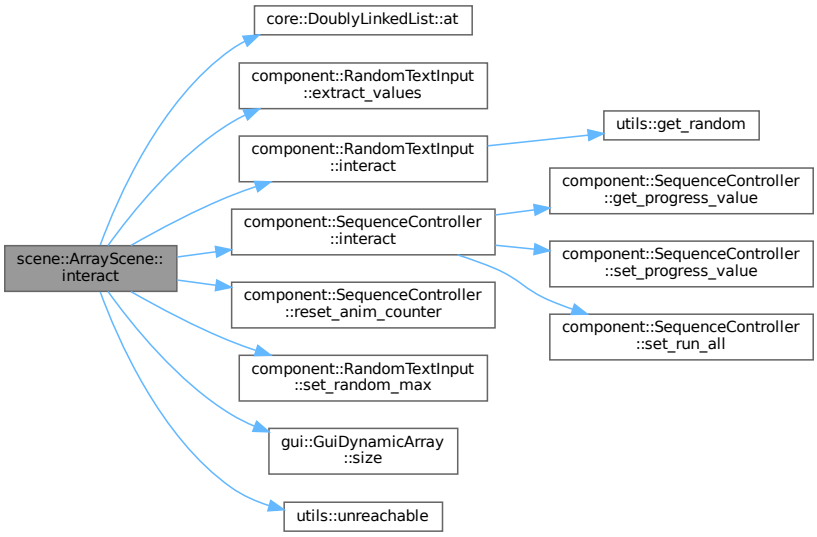
6.1.3.1 interact()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 89 of file array_scene.cpp.

Here is the call graph for this function:



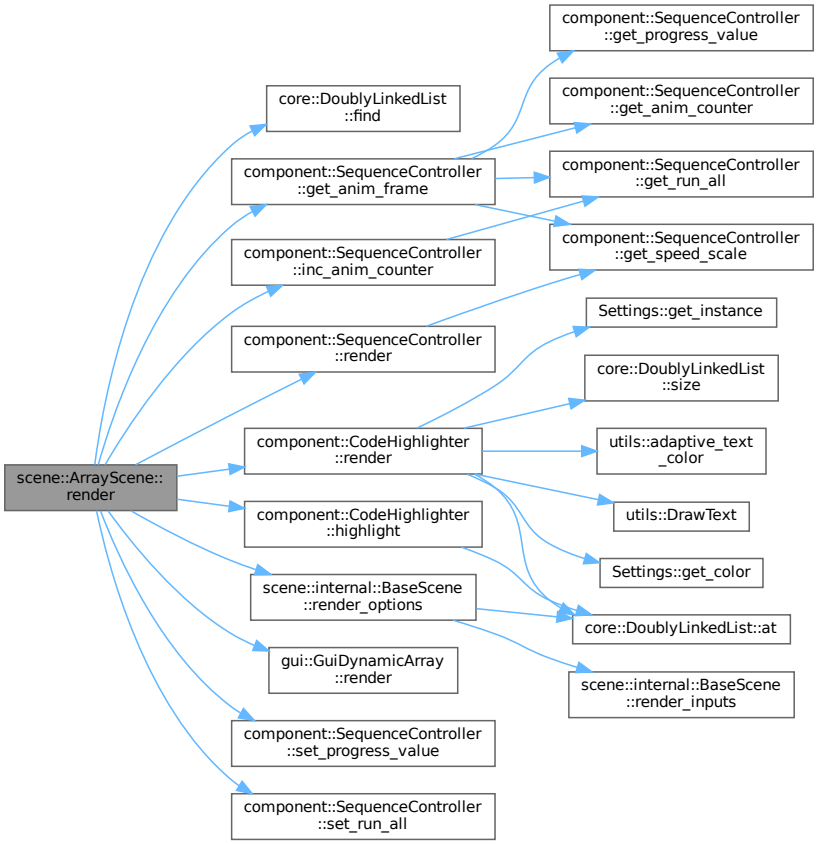
6.1.3.2 render()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 69 of file array_scene.cpp.

Here is the call graph for this function:



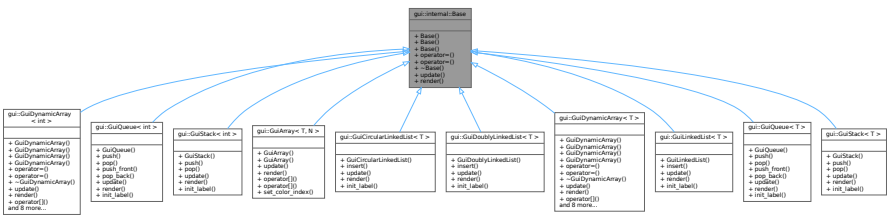
The documentation for this class was generated from the following files:

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

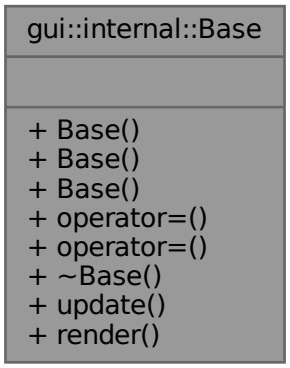
6.2 gui::internal::Base Class Reference

`#include <base_gui.hpp>`

Inheritance diagram for gui::internal::Base:



Collaboration diagram for gui::internal::Base:



Public Member Functions

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

6.2.1 Detailed Description

Definition at line 8 of file base_gui.hpp.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Base() [1/3]

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

6.2.2.2 Base() [2/3]

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

6.2.2.3 Base() [3/3]

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

6.2.2.4 ~Base()

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

6.2.3 Member Function Documentation

6.2.3.1 operator=() [1/2]

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

6.2.3.2 operator=() [2/2]

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

6.2.3.3 render()

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

Implemented in gui::GuiArray< T, N >, gui::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::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 >.

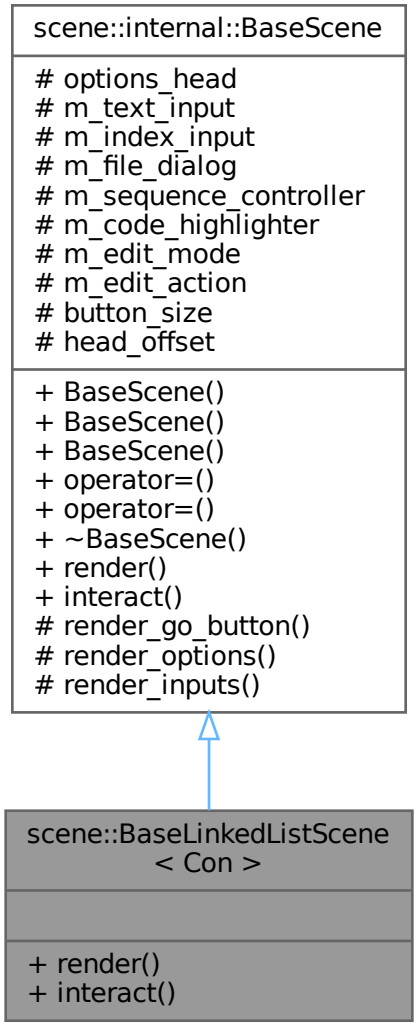
The documentation for this class was generated from the following file:

- src/gui/base_gui.hpp

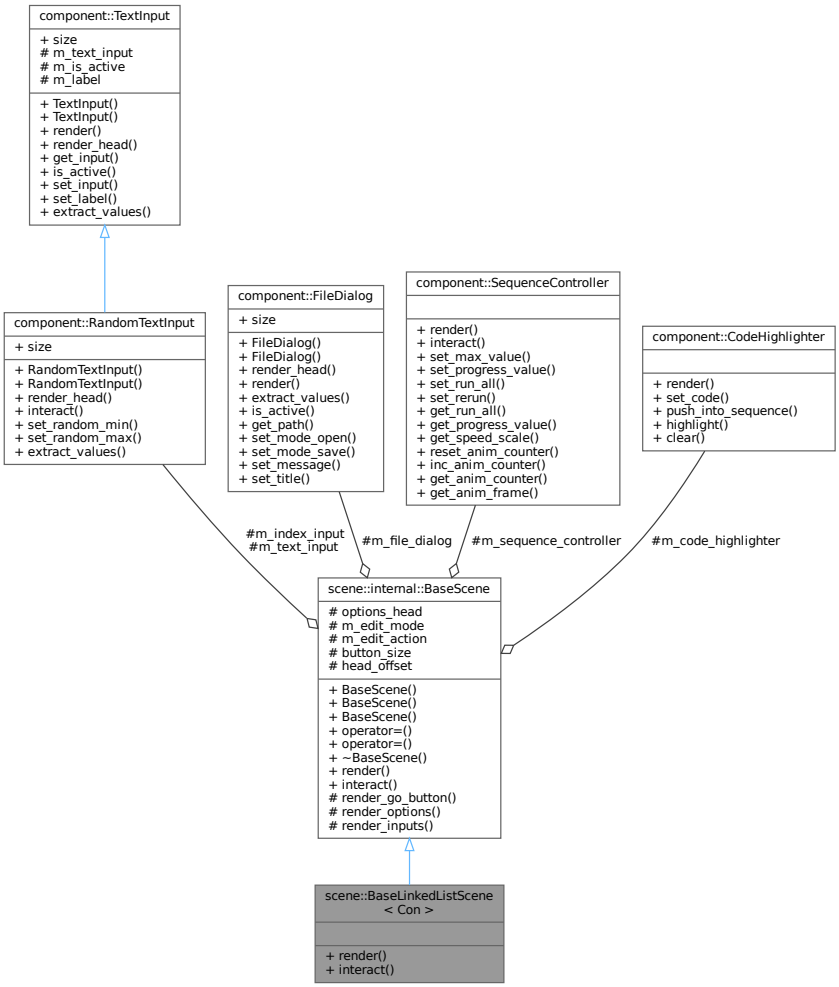
6.3 scene::BaseLinkedListScene< Con > Class Template Reference

```
#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::RandomTextInput m_text_input {"value"}
- component::RandomTextInput 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 16 of file base_linked_list_scene.hpp.

6.3.2 Member Function Documentation

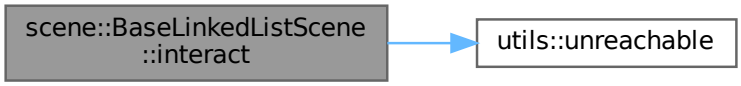
6.3.2.1 interact()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 169 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 148 of file base_linked_list_scene.hpp.

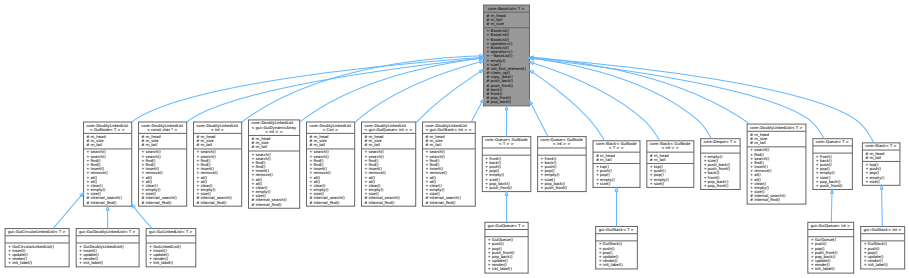
The documentation for this class was generated from the following file:

- src/scene/base_linked_list_scene.hpp

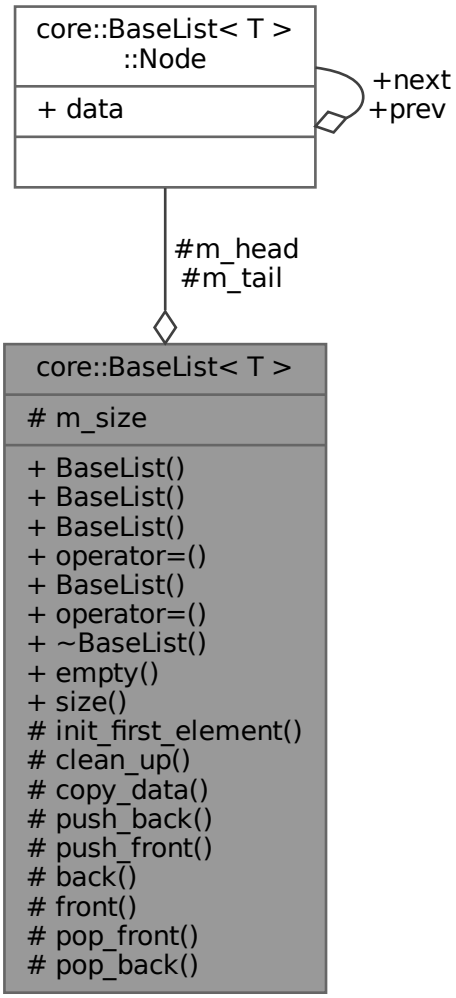
6.4 core::BaseList< T > Class Template Reference

```
#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.

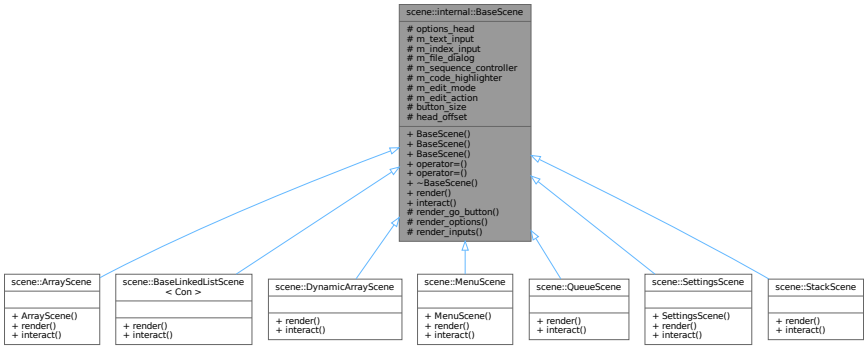
The documentation for this class was generated from the following file:

- src/core/base_list.hpp

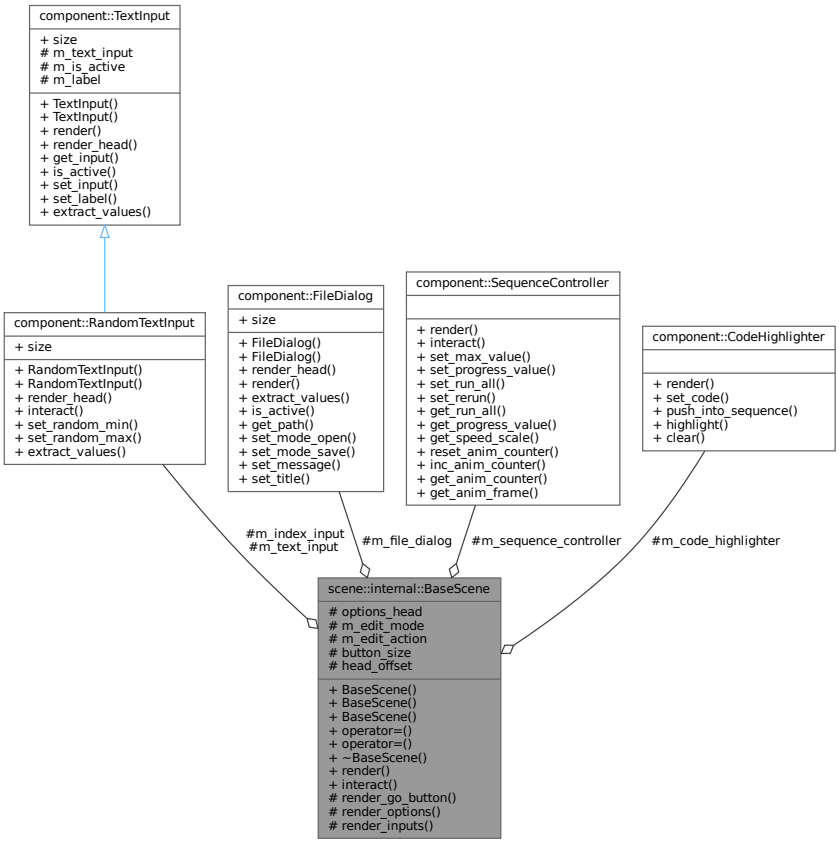
6.5 scene::internal::BaseScene Class Reference

```
#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::RandomTextInput m_text_input {"value"}
- component::RandomTextInput m_index_input {"index"}
- component::FileDialog m_file_dialog
- component::SequenceController m_sequence_controller
- component::CodeHighlighter m_code_highlighter
- bool m_edit_mode {}
- bool m_edit_action {}

Static Protected Attributes

- static constexpr Vector2 button_size {200, 50}
- static constexpr int head_offset = 20

6.5.1 Detailed Description

Definition at line 13 of file base_scene.hpp.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 BaseScene() [1/3]

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

6.5.2.2 BaseScene() [2/3]

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

6.5.2.3 BaseScene() [3/3]

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

6.5.2.4 ~BaseScene()

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

6.5.3 Member Function Documentation

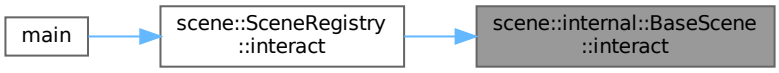
6.5.3.1 interact()

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

Reimplemented in scene::ArrayScene, scene::BaseLinkedListScene< Con >, scene::DynamicArrayScene, scene::MenuScene, scene::QueueScene, scene::SettingsScene, and scene::StackScene.

Definition at line 42 of file base_scene.hpp.

Here is the caller graph for this function:



6.5.3.2 operator=() [1/2]

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

6.5.3.3 operator=() [2/2]

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

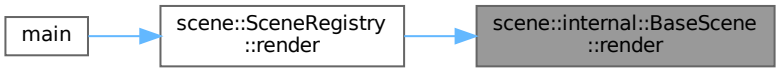
6.5.3.4 render()

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

Reimplemented in scene::ArrayScene, scene::BaseLinkedListScene< Con >, scene::DynamicArrayScene, scene::MenuScene, scene::QueueScene, scene::SettingsScene, and scene::StackScene.

Definition at line 41 of file base_scene.hpp.

Here is the caller graph for this function:



6.5.3.5 render_go_button()

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

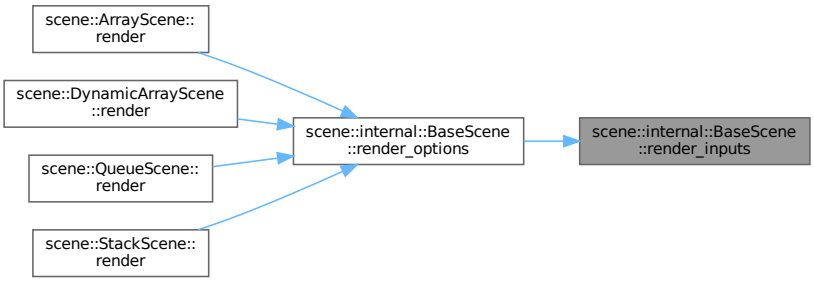
Definition at line 10 of file base_scene.cpp.

6.5.3.6 render_inputs()

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

Definition at line 21 of file base_scene.hpp.

Here is the caller graph for this function:

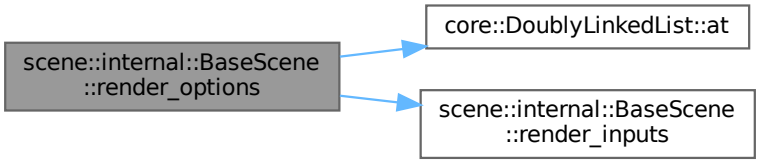


6.5.3.7 render_options()

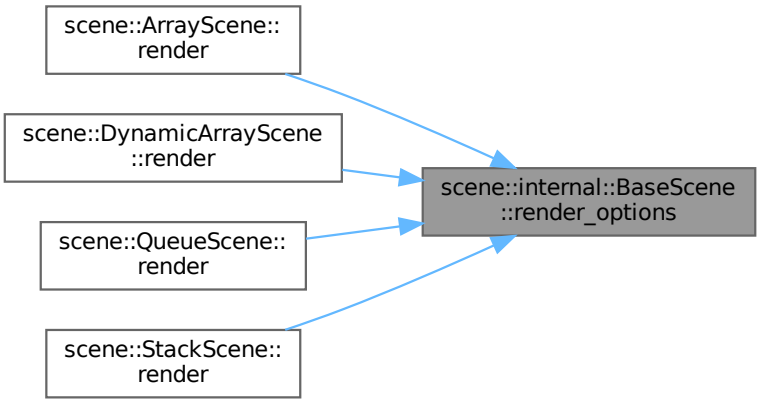
```
void scene::internal::BaseScene::render_options (
            SceneOptions & scene_config )  [protected], [virtual]
```

Definition at line 16 of file base_scene.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.4 Member Data Documentation

6.5.4.1 button_size

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

Definition at line 15 of file base_scene.hpp.

6.5.4.2 head_offset

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

Definition at line 16 of file base_scene.hpp.

6.5.4.3 m_code_highlighter

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

Definition at line 27 of file base_scene.hpp.

6.5.4.4 m_edit_action

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

Definition at line 30 of file base_scene.hpp.

6.5.4.5 m_edit_mode

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

Definition at line 29 of file base_scene.hpp.

6.5.4.6 m_file_dialog

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

Definition at line 25 of file base_scene.hpp.

6.5.4.7 m_index_input

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

Definition at line 23 of file base_scene.hpp.

6.5.4.10 options_head

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

Definition at line 17 of file base_scene.hpp.

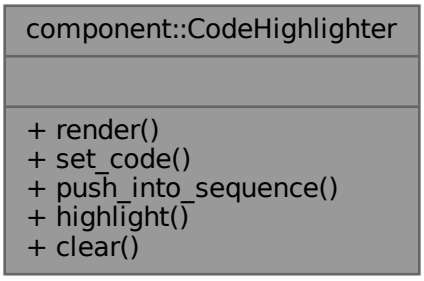
The documentation for this class was generated from the following files:

- src/scene/base_scene.hpp
- src/scene/base_scene.cpp

6.6 component::CodeHighlighter Class Reference

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

Collaboration diagram for component::CodeHighlighter:



Public Member Functions

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

6.6.1 Detailed Description

Definition at line 10 of file code_highlighter.hpp.

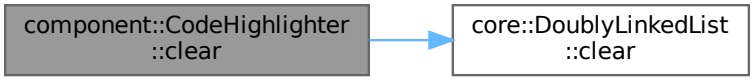
6.6.2 Member Function Documentation

6.6.2.1 clear()

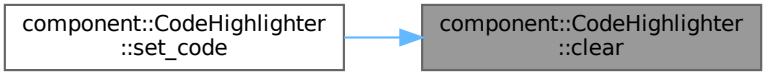
```
void component::CodeHighlighter::clear ( )
```

Definition at line 38 of file code_highlighter.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

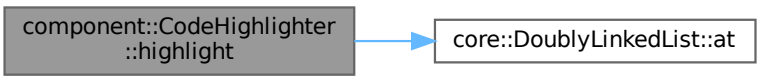


6.6.2.2 highlight()

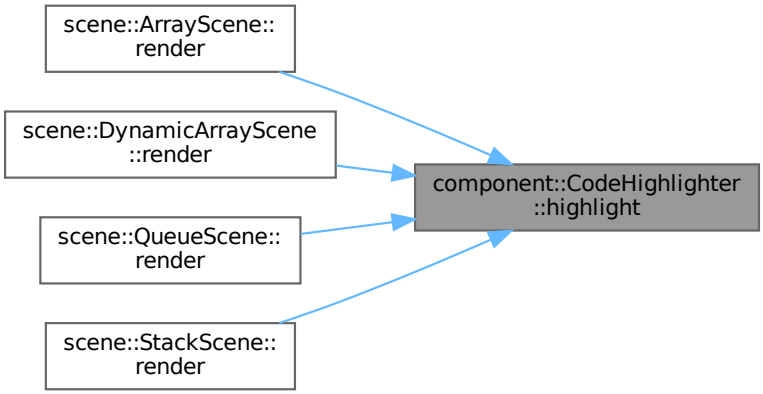
```
void component::CodeHighlighter::highlight (
            int frame_idx )
```

Definition at line 34 of file code_highlighter.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

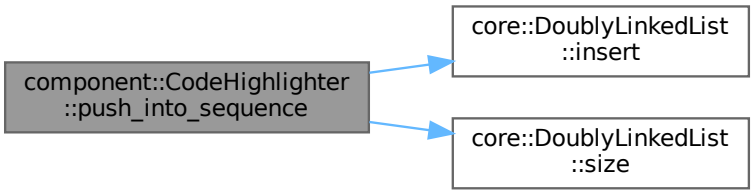


6.6.2.3 push_into_sequence()

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

Definition at line 30 of file code_highlighter.cpp.

Here is the call graph for this function:

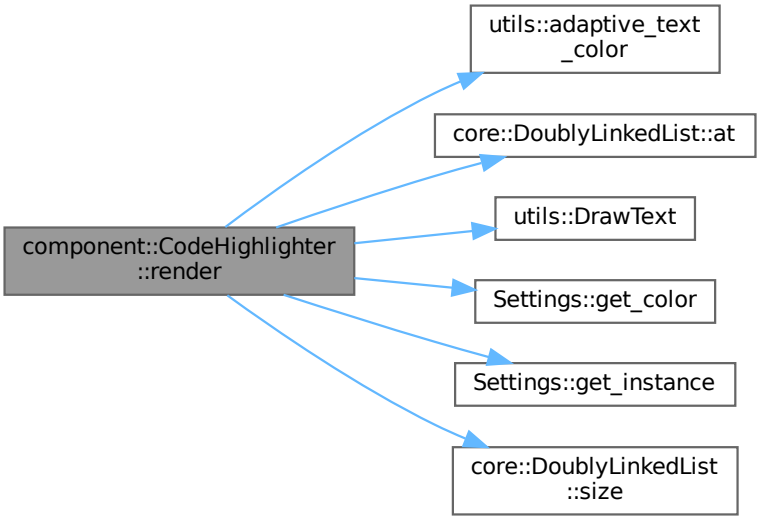


6.6.2.4 render()

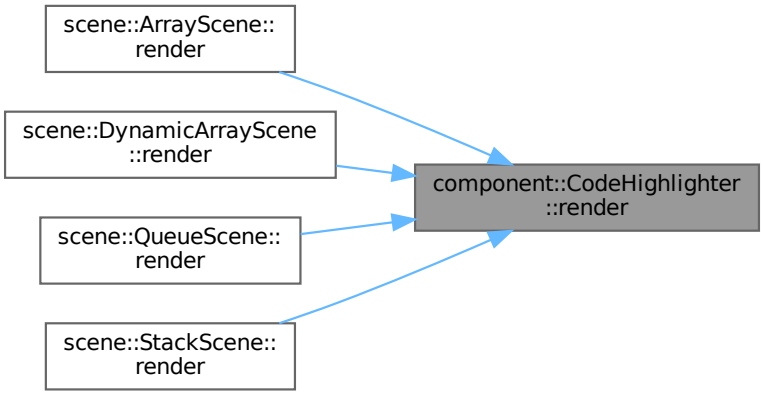
```
void component::CodeHighlighter::render ( )
```

Definition at line 9 of file code_highlighter.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.2.5 set_code()

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

Definition at line 25 of file code_highlighter.cpp.

Here is the call graph for this function:

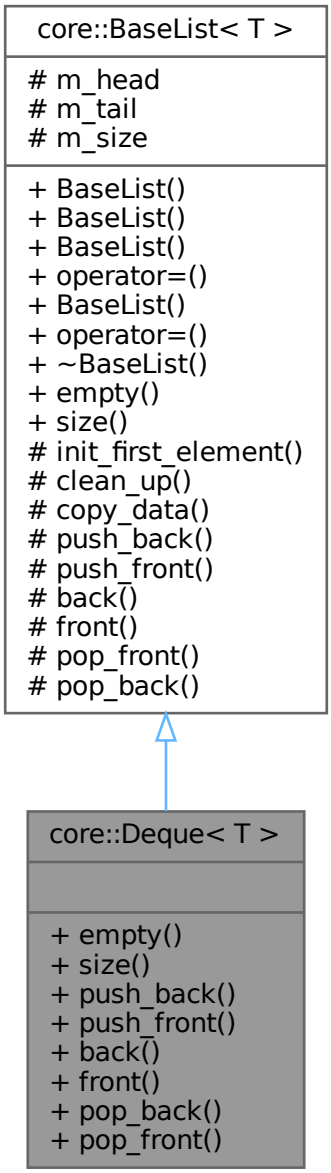
The documentation for this class was generated from the following files:

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

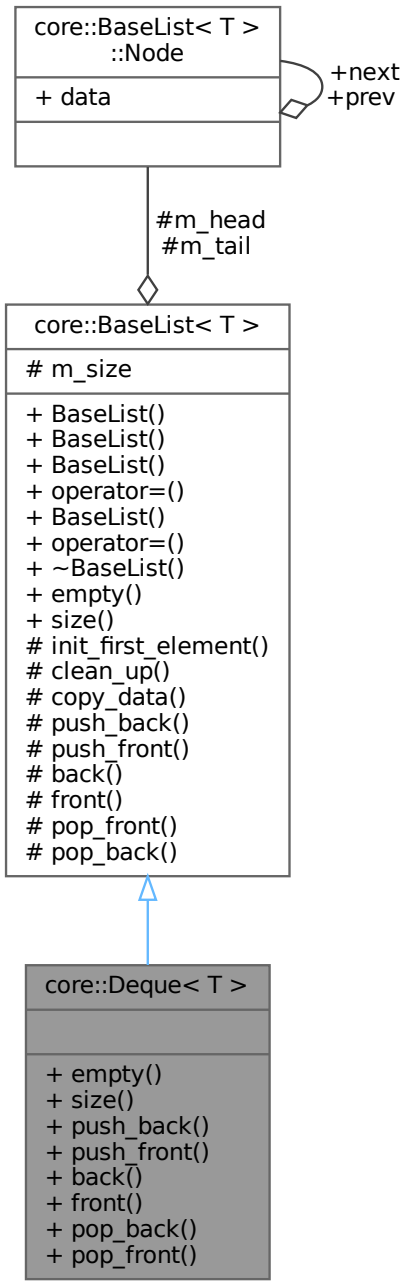
6.7 core::Deque< T > Class Template Reference

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

Inheritance diagram for core::Deque< T >:



Collaboration diagram for core::Deque< T >:



Public Member Functions

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

- T & front () const
- void pop_back ()
- void pop_front ()

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

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

Additional Inherited Members

Protected Types inherited from core::BaseList< T >

- using Node_ptr = Node *

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

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

Protected Attributes inherited from core::BaseList< T >

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

6.7.1 Detailed Description

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

Definition at line 9 of file deque.hpp.

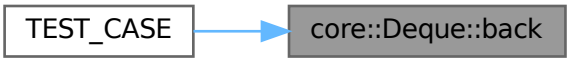
6.7.2 Member Function Documentation

6.7.2.1 back()

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

Definition at line 33 of file base_list.hpp.

Here is the caller graph for this function:

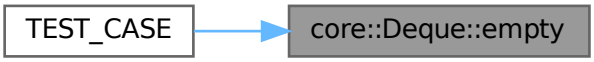


6.7.2.2 empty()

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

Definition at line 48 of file base_list.hpp.

Here is the caller graph for this function:

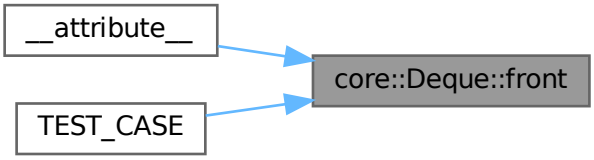


6.7.2.3 front()

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

Definition at line 34 of file base_list.hpp.

Here is the caller graph for this function:



6.7.2.4 pop_back()

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

Definition at line 37 of file base_list.hpp.

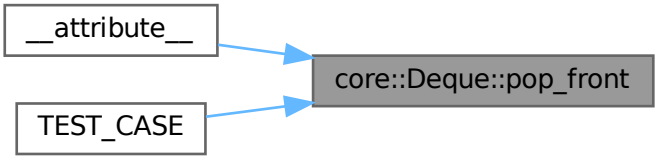
Here is the caller graph for this function:

6.7.2.5 pop_front()

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

Definition at line 36 of file base_list.hpp.

Here is the caller graph for this function:

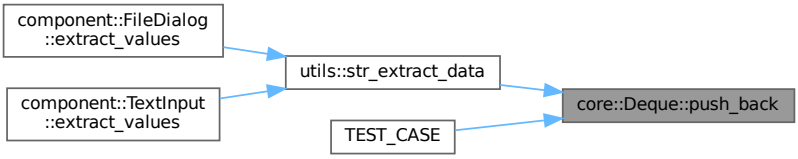


6.7.2.6 push_back()

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

Definition at line 30 of file base_list.hpp.

Here is the caller graph for this function:



6.7.2.7 push_front()

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

Definition at line 31 of file base_list.hpp.

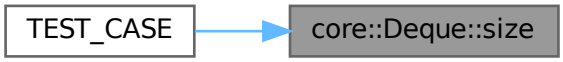
Here is the caller graph for this function:

6.7.2.8 size()

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

Definition at line 49 of file base_list.hpp.

Here is the caller graph for this function:



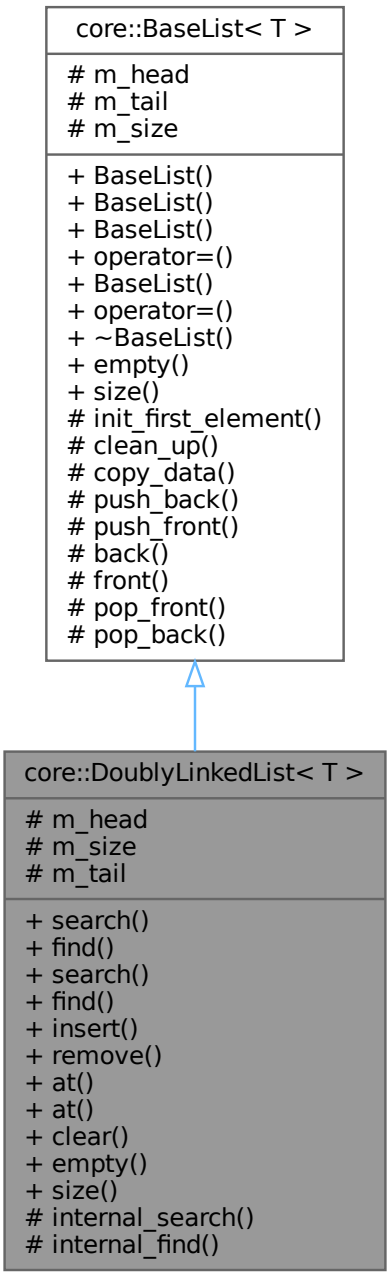
The documentation for this class was generated from the following file:

- src/core/deque.hpp

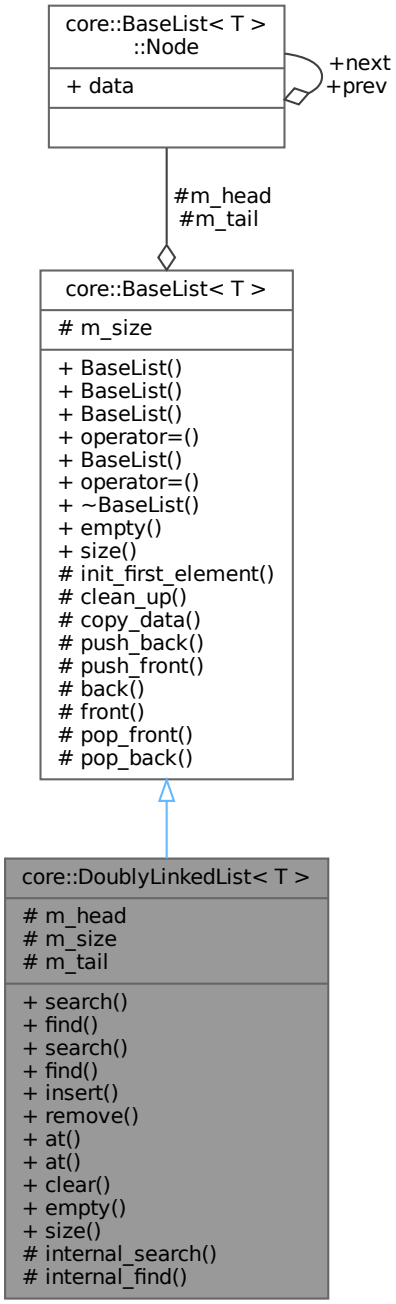
6.8 core::DoublyLinkedList< T > Class Template Reference

`#include <doubly_linked_list.hpp>`

Inheritance diagram for core::DoublyLinkedList< T >:



Collaboration diagram for core::DoublyLinkedList< T >:



Public Member Functions

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

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

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

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

Protected Types

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

Protected Types inherited from core::BaseList< T >

- using Node_ptr = Node *

Protected Member Functions

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

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

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

Protected Attributes

- Node_ptr m_head
- std::size_t m_size
- Node_ptr m_tail

Protected Attributes inherited from core::BaseList< T >

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

6.8.1 Detailed Description

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

Definition at line 11 of file doubly_linked_list.hpp.

6.8.2 Member Typedef Documentation

6.8.2.1 Base

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

Definition at line 13 of file doubly_linked_list.hpp.

6.8.2.2 cNode_ptr

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

Definition at line 16 of file doubly_linked_list.hpp.

6.8.2.3 Node

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

Definition at line 14 of file doubly_linked_list.hpp.

6.8.2.4 Node_ptr

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

Definition at line 15 of file doubly_linked_list.hpp.

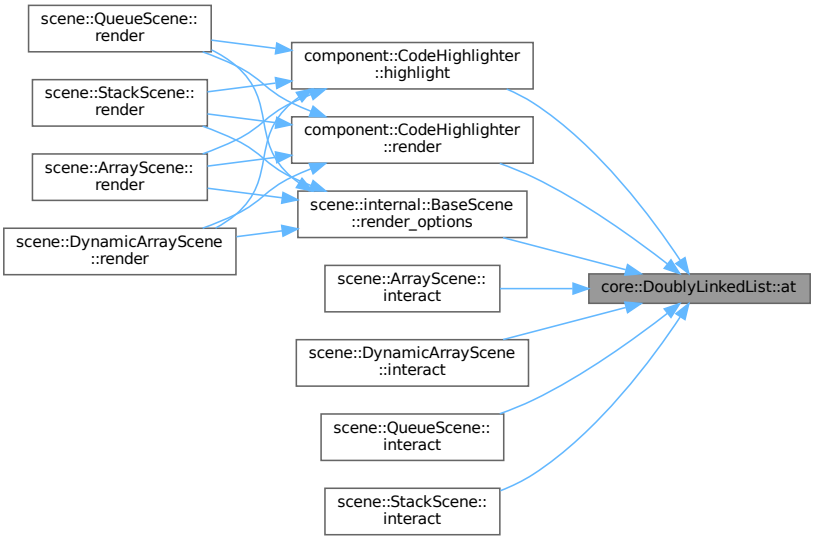
6.8.3 Member Function Documentation

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:



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:

6.8.3.4 empty()

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

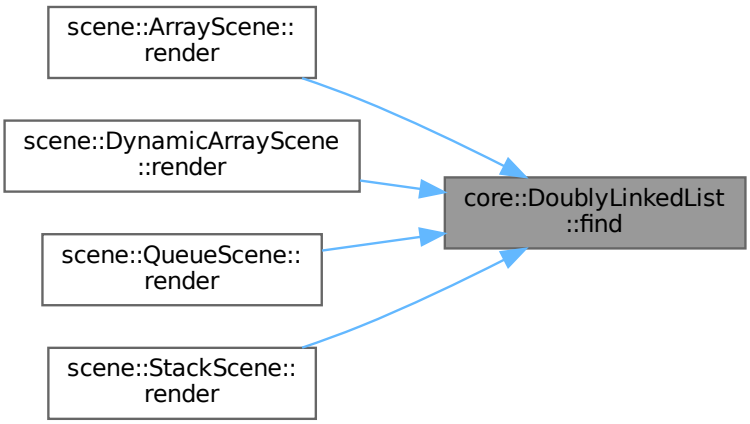
Definition at line 48 of file base_list.hpp.

6.8.3.5 find() [1/2]

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

Definition at line 83 of file doubly_linked_list.hpp.

Here is the caller graph for this function:



6.8.3.6 find() [2/2]

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

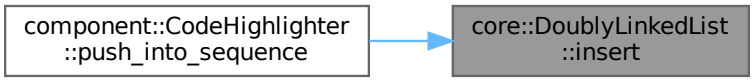
Definition at line 95 of file doubly_linked_list.hpp.

6.8.3.7 insert()

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

Definition at line 101 of file doubly_linked_list.hpp.

Here is the caller graph for this function:



6.8.3.8 internal_find()

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

Definition at line 63 of file doubly_linked_list.hpp.

6.8.3.9 internal_search()

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

Definition at line 47 of file doubly_linked_list.hpp.

6.8.3.10 remove()

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

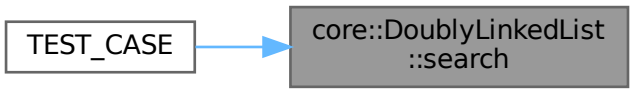
Definition at line 124 of file doubly_linked_list.hpp.

6.8.3.11 search() [1/2]

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

Definition at line 77 of file doubly_linked_list.hpp.

Here is the caller graph for this function:



6.8.3.12 search() [2/2]

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

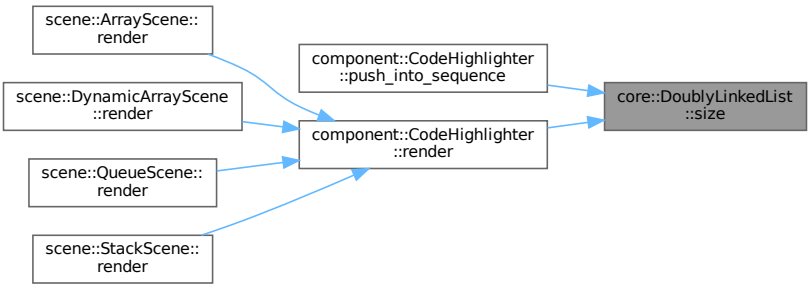
Definition at line 89 of file doubly_linked_list.hpp.

6.8.3.13 size()

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

Definition at line 49 of file base_list.hpp.

Here is the caller graph for this function:



6.8.4 Member Data Documentation

6.8.4.1 m_head

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

Definition at line 22 of file base_list.hpp.

6.8.4.2 m_size

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

Definition at line 24 of file base_list.hpp.

6.8.4.3 m_tail

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

Definition at line 23 of file base_list.hpp.

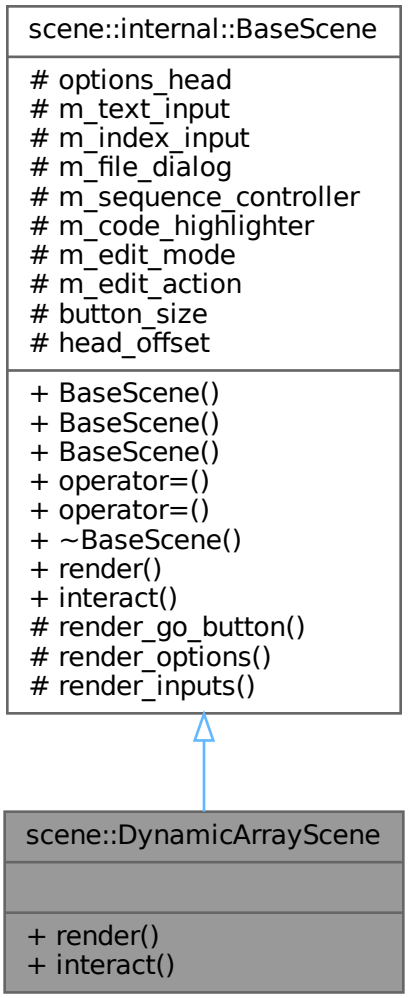
The documentation for this class was generated from the following file:

- src/core/doubly_linked_list.hpp

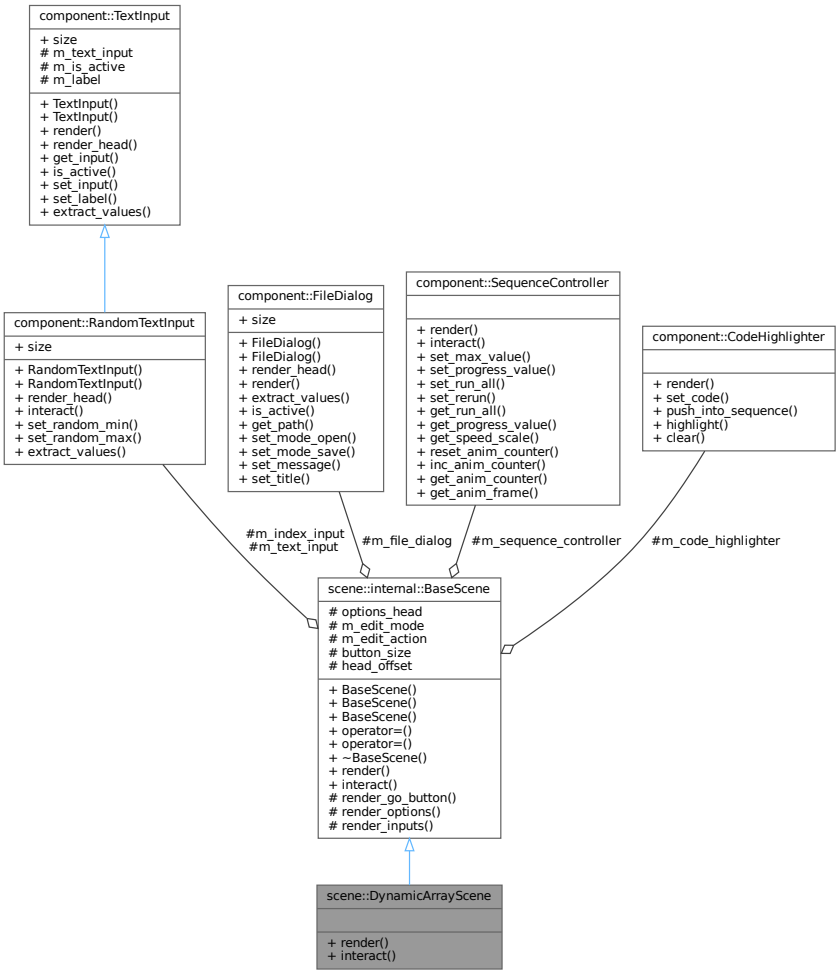
6.9 scene::DynamicArrayScene Class Reference

`#include <dynamic_array_scene.hpp>`

Inheritance diagram for scene::DynamicArrayScene:



Collaboration diagram for scene::DynamicArrayScene:



Public Member Functions

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

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

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

Additional Inherited Members

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

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

Protected Attributes inherited from scene::internal::BaseScene

- float options_head {}
- component::RandomTextInput m_text_input {"value"}
- component::RandomTextInput 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 17 of file dynamic_array_scene.hpp.

6.9.2 Member Function Documentation

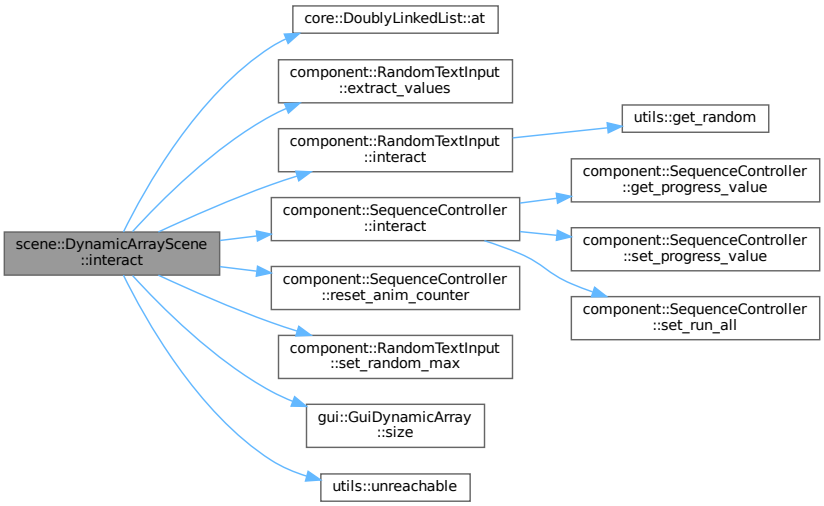
6.9.2.1 interact()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 99 of file dynamic_array_scene.cpp.

Here is the call graph for this function:



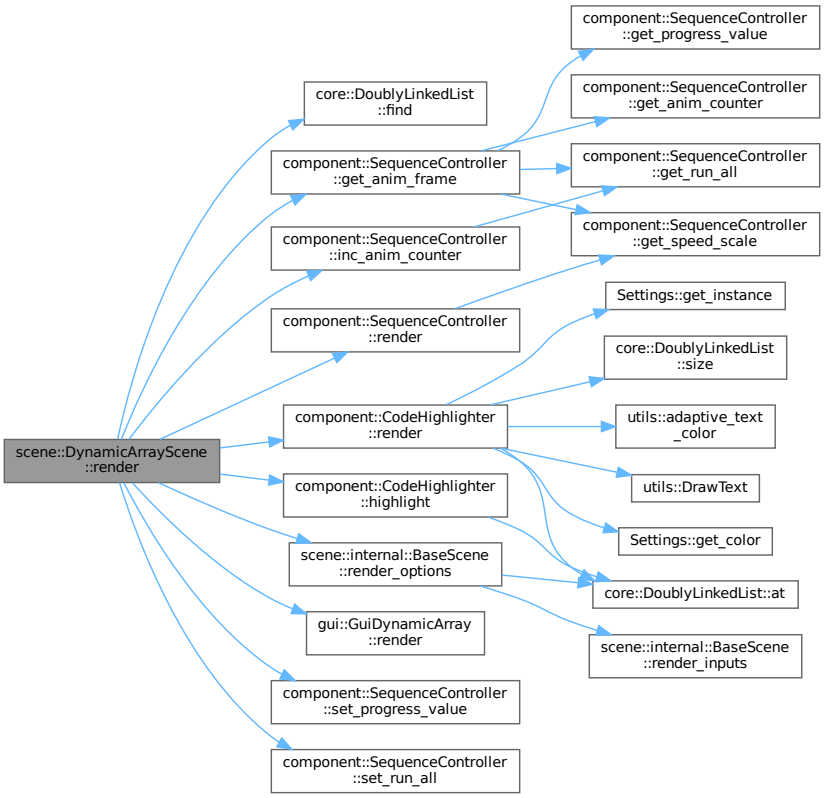
6.9.2.2 render()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 79 of file dynamic_array_scene.cpp.

Here is the call graph for this function:



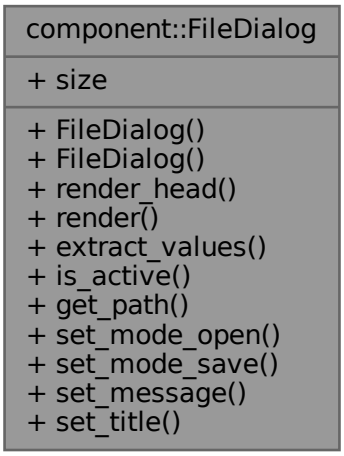
The documentation for this class was generated from the following files:

- src/scene/dynamic_array_scene.hpp
- src/scene/dynamic_array_scene.cpp

6.10 component::FileDialog Class Reference

`#include <file_dialog.hpp>`

Collaboration diagram for component::FileDialog:



Public Member Functions

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

Static Public Attributes

- static constexpr Vector2 size {200, 50}

6.10.1 Detailed Description

Definition at line 13 of file file_dialog.hpp.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 FileDialog() [1/2]

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

Definition at line 16 of file file_dialog.cpp.

6.10.2.2 FileDialog() [2/2]

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

Definition at line 13 of file file_dialog.cpp.

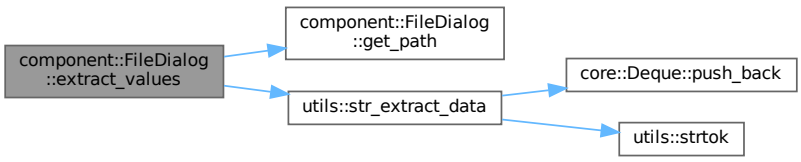
6.10.3 Member Function Documentation

6.10.3.1 extract_values()

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

Definition at line 49 of file file_dialog.cpp.

Here is the call graph for this function:

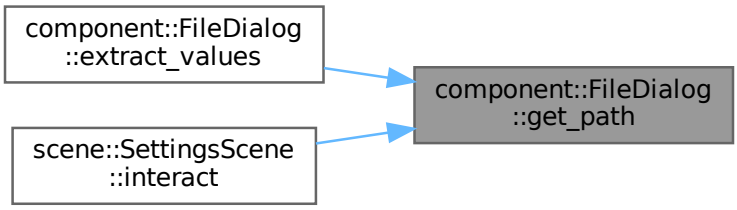


6.10.3.2 get_path()

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

Definition at line 66 of file file_dialog.cpp.

Here is the caller graph for this function:



6.10.3.3 is_active()

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

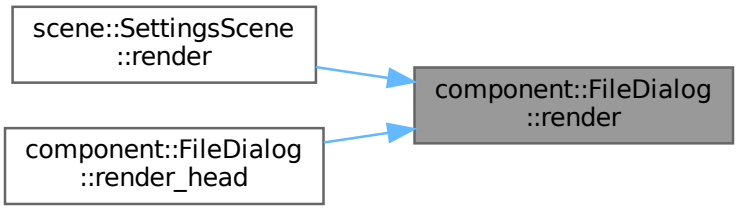
Definition at line 57 of file file_dialog.cpp.

6.10.3.4 render()

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

Definition at line 18 of file file_dialog.cpp.

Here is the caller graph for this function:

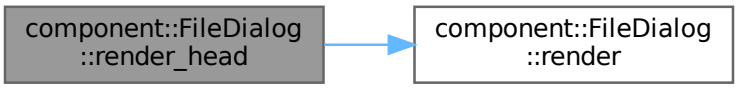


6.10.3.5 render_head()

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

Definition at line 43 of file file_dialog.cpp.

Here is the call graph for this function:



6.10.3.6 set_message()

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

Definition at line 63 of file file_dialog.cpp.

6.10.3.7 set_mode_open()

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

Definition at line 59 of file file_dialog.cpp.

6.10.3.8 set_mode_save()

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

Definition at line 61 of file file_dialog.cpp.

6.10.3.9 set_title()

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

Definition at line 65 of file file_dialog.cpp.

6.10.4 Member Data Documentation

6.10.4.1 size

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

Definition at line 25 of file file_dialog.hpp.

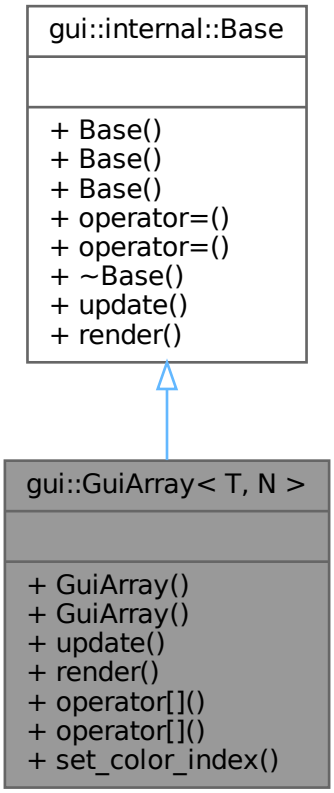
The documentation for this class was generated from the following files:

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

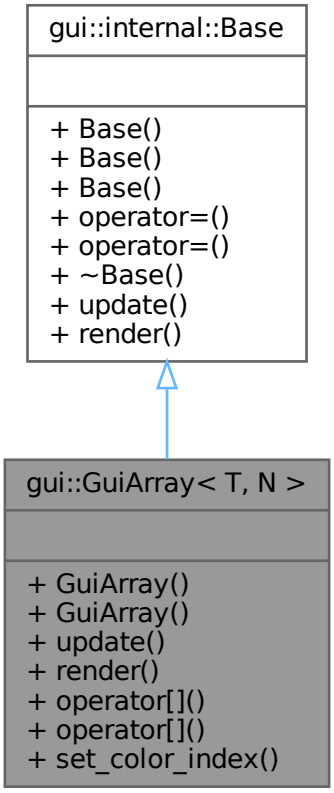
6.11 gui::GuiArray< T, N > Class Template Reference

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

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



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



Public Member Functions

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

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

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

6.11.1 Detailed Description

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

Definition at line 16 of file array_gui.hpp.

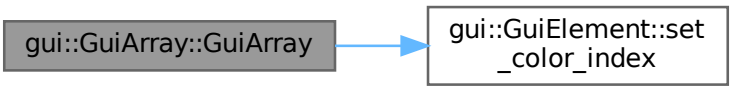
6.11.2 Constructor & Destructor Documentation

6.11.2.1 GuiArray() [1/2]

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

Definition at line 39 of file array_gui.hpp.

Here is the call graph for this function:



6.11.2.2 GuiArray() [2/2]

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

Definition at line 47 of file array_gui.hpp.

6.11.3 Member Function Documentation

6.11.3.1 operator[]() [1/2]

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

Definition at line 73 of file array_gui.hpp.

6.11.3.2 operator[]() [2/2]

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

Definition at line 78 of file array_gui.hpp.

6.11.3.3 render()

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

Implements gui::internal::Base.

Definition at line 54 of file array_gui.hpp.

6.11.3.4 set_color_index()

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

Definition at line 83 of file array_gui.hpp.

6.11.3.5 update()

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

Implements gui::internal::Base.

Definition at line 63 of file array_gui.hpp.

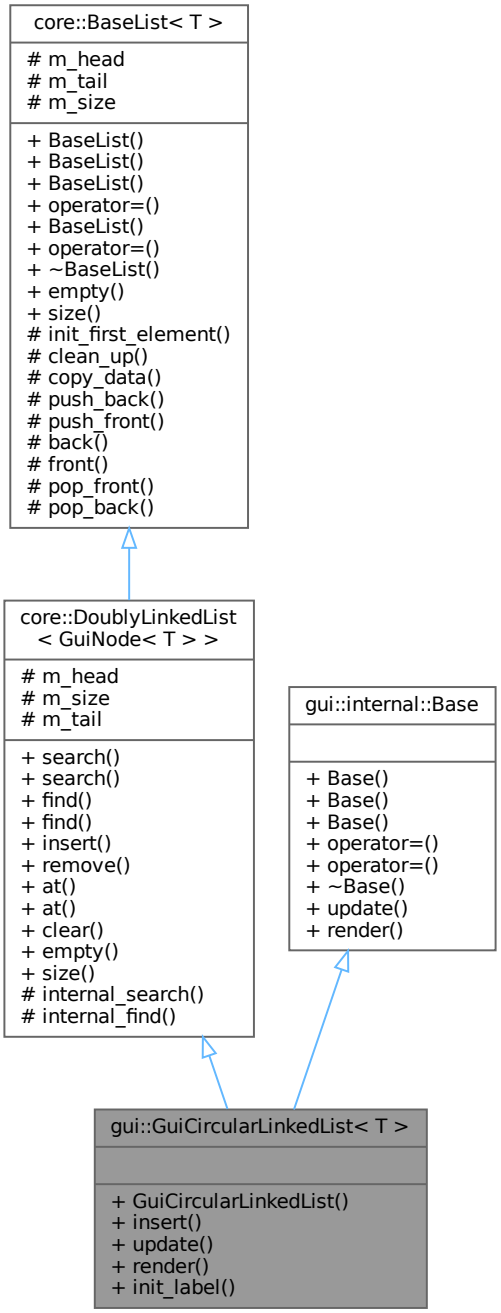
The documentation for this class was generated from the following file:

- src/gui/array_gui.hpp

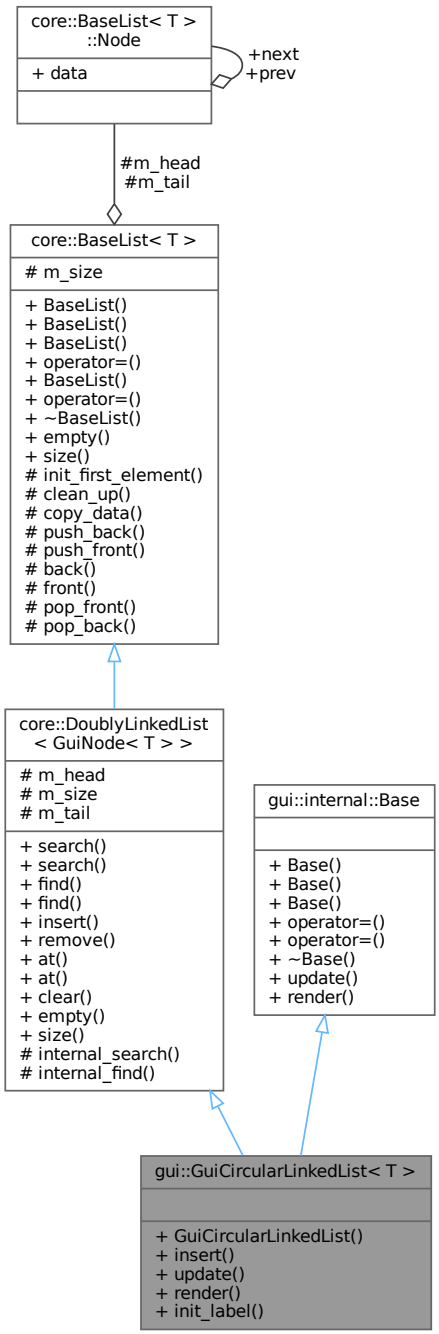
6.12 gui::GuiCircularLinkedList< T > Class Template Reference

`#include <circular_linked_list_gui.hpp>`

Inheritance diagram for gui::GuiCircularLinkedList< T >:



Collaboration diagram for gui::GuiCircularLinkedList< T >:



Public Member Functions

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

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

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

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

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

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

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

Additional Inherited Members

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

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

Protected Types inherited from core::BaseList< T >

- using Node_ptr = Node *

Protected Member Functions inherited from core::DoublyLinkedList< GuiNode< T > >

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

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

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

Protected Attributes inherited from core::DoublyLinkedList< GuiNode< T > >

- Node_ptr m_head
- std::size_t m_size
- Node_ptr m_tail

Protected Attributes inherited from core::BaseList< T >

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

6.12.1 Detailed Description

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

Definition at line 19 of file circular_linked_list_gui.hpp.

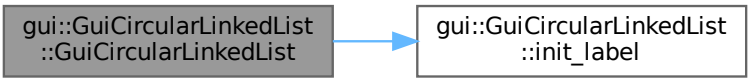
6.12.2 Constructor & Destructor Documentation

6.12.2.1 GuiCircularLinkedList()

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

Definition at line 65 of file circular_linked_list_gui.hpp.

Here is the call graph for this function:



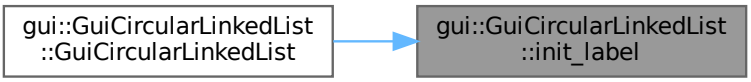
6.12.3 Member Function Documentation

6.12.3.1 init_label()

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

Definition at line 50 of file circular_linked_list_gui.hpp.

Here is the caller graph for this function:



6.12.3.2 insert()

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

Definition at line 72 of file circular_linked_list_gui.hpp.

6.12.3.3 render()

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

Implements gui::internal::Base.

Definition at line 129 of file circular_linked_list_gui.hpp.

6.12.3.4 update()

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

Implements gui::internal::Base.

Definition at line 143 of file circular_linked_list_gui.hpp.

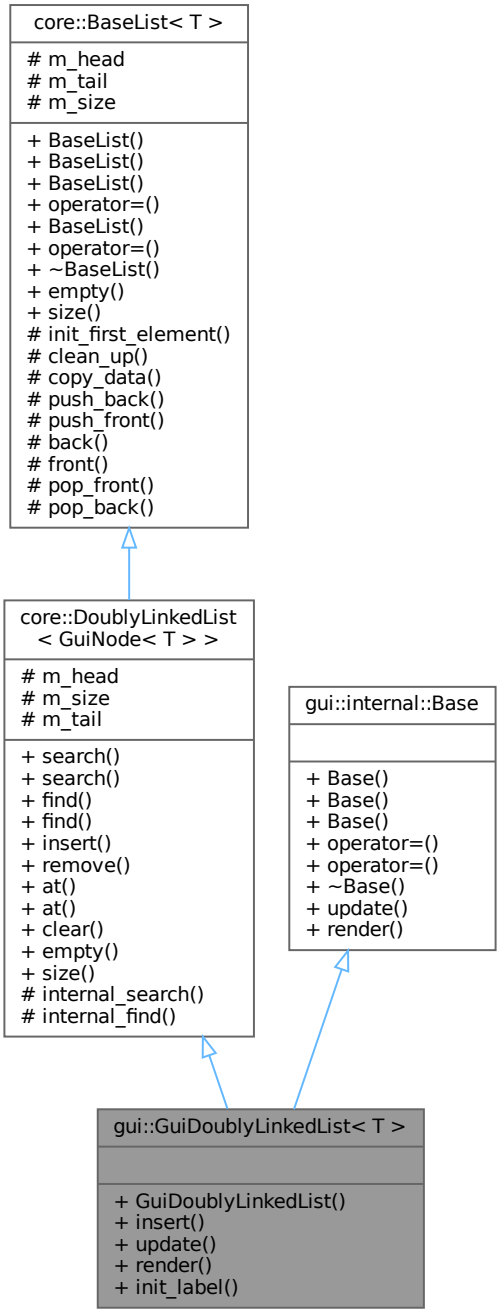
The documentation for this class was generated from the following file:

- src/gui/circular_linked_list_gui.hpp

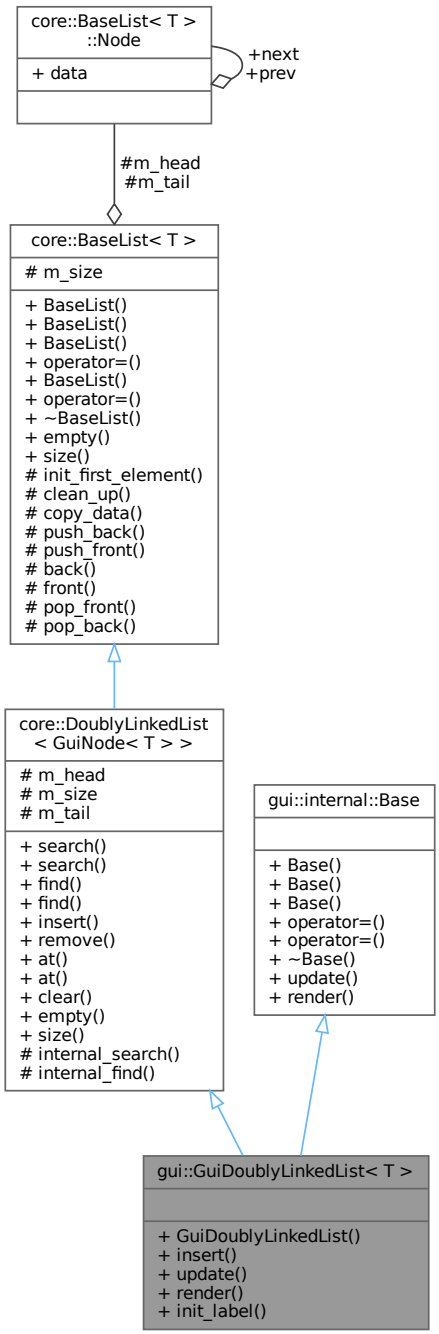
6.13 gui::GuiDoublyLinkedList< T > Class Template Reference

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

Inheritance diagram for gui::GuiDoublyLinkedList< T >:



Collaboration diagram for gui::GuiDoublyLinkedList< T >:



Public Member Functions

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

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

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

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

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

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

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

Additional Inherited Members

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

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

Protected Types inherited from core::BaseList< T >

- using Node_ptr = Node *

Protected Member Functions inherited from core::DoublyLinkedList< GuiNode< T > >

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

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

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

Protected Attributes inherited from core::DoublyLinkedList< GuiNode< T > >

- Node_ptr m_head
- std::size_t m_size
- Node_ptr m_tail

Protected Attributes inherited from core::BaseList< T >

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

6.13.1 Detailed Description

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

Definition at line 17 of file doubly_linked_list_gui.hpp.

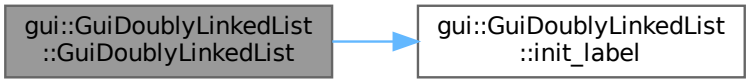
6.13.2 Constructor & Destructor Documentation

6.13.2.1 GuiDoublyLinkedList()

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

Definition at line 62 of file doubly_linked_list_gui.hpp.

Here is the call graph for this function:



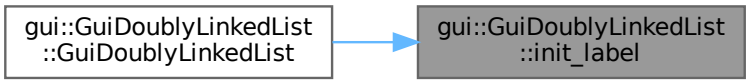
6.13.3 Member Function Documentation

6.13.3.1 init_label()

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

Definition at line 47 of file doubly_linked_list_gui.hpp.

Here is the caller graph for this function:



6.13.3.2 insert()

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

Definition at line 69 of file doubly_linked_list_gui.hpp.

6.13.3.3 render()

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

Implements gui::internal::Base.

Definition at line 105 of file doubly_linked_list_gui.hpp.

6.13.3.4 update()

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

Implements gui::internal::Base.

Definition at line 118 of file doubly_linked_list_gui.hpp.

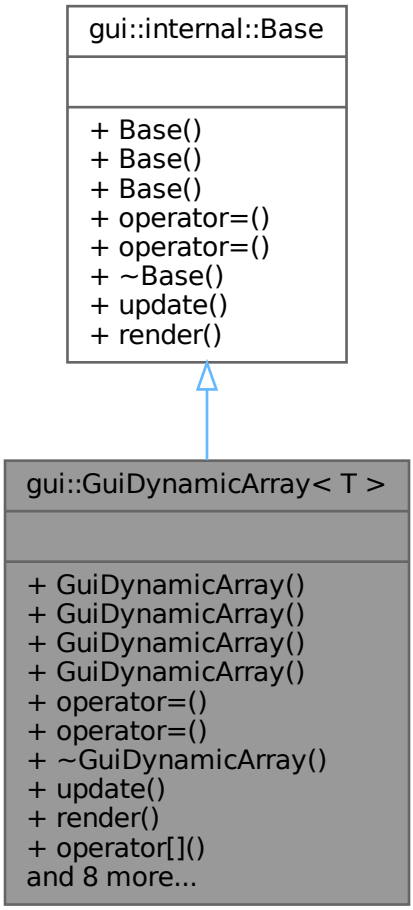
The documentation for this class was generated from the following file:

- src/gui/doubly_linked_list_gui.hpp

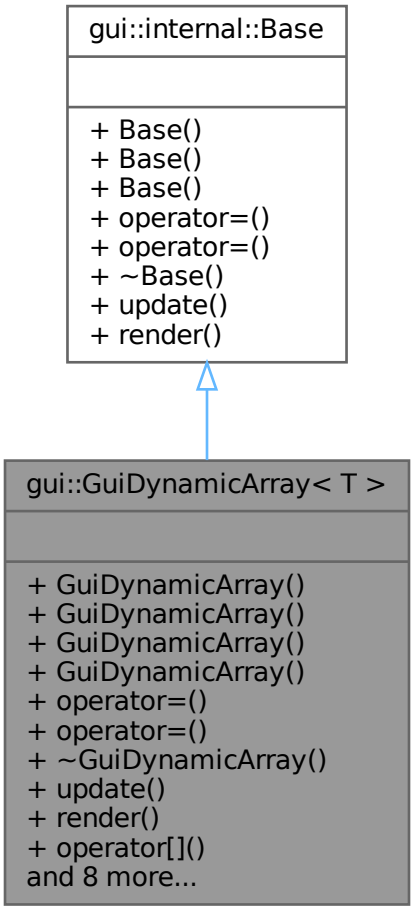
6.14 gui::GuiDynamicArray< T > Class Template Reference

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

Inheritance diagram for gui::GuiDynamicArray< T >:



Collaboration diagram for gui::GuiDynamicArray< T >:



Public Member Functions

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

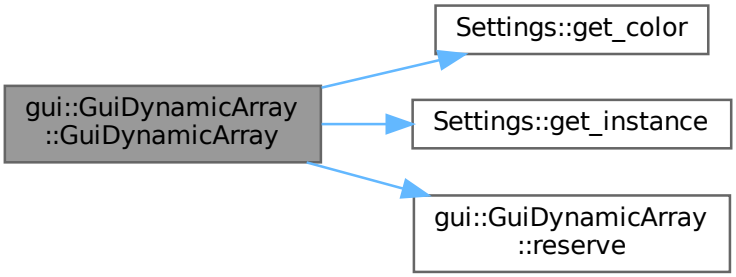
Definition at line 99 of file dynamic_array_gui.hpp.

6.14.2.2 GuiDynamicArray() [2/4]

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

Definition at line 106 of file dynamic_array_gui.hpp.

Here is the call graph for this function:



6.14.2.3 GuiDynamicArray() [3/4]

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

Definition at line 117 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 127 of file dynamic_array_gui.hpp.

6.14.2.5 ~GuiDynamicArray()

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

Definition at line 165 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 209 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 135 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 151 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 194 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 199 of file dynamic_array_gui.hpp.

6.14.3.6 pop()

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

Definition at line 230 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 219 of file dynamic_array_gui.hpp.

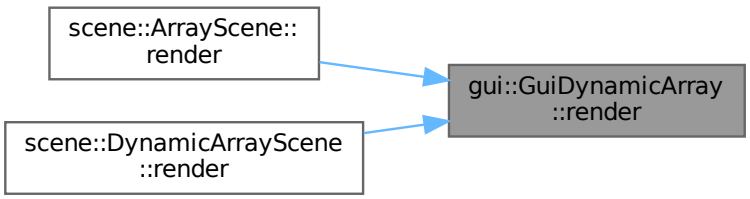
6.14.3.8 render()

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

Implements gui::internal::Base.

Definition at line 173 of file dynamic_array_gui.hpp.

Here is the caller graph for this function:

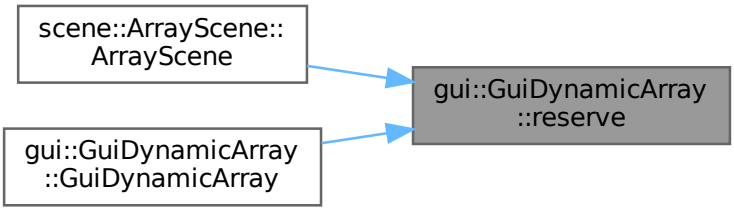


6.14.3.9 reserve()

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

Definition at line 56 of file dynamic_array_gui.hpp.

Here is the caller graph for this function:



6.14.3.10 set_color_index()

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

Definition at line 204 of file dynamic_array_gui.hpp.

6.14.3.11 shrink_to_fit()

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

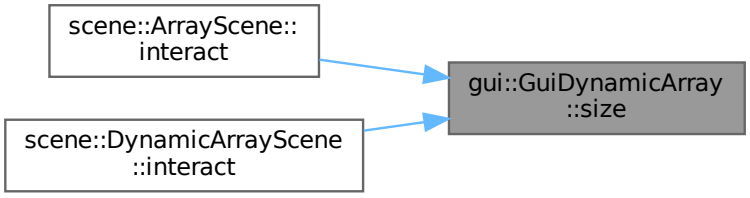
Definition at line 79 of file dynamic_array_gui.hpp.

6.14.3.12 size()

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

Definition at line 214 of file dynamic_array_gui.hpp.

Here is the caller graph for this function:



6.14.3.13 update()

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

Implements gui::internal::Base.

Definition at line 184 of file dynamic_array_gui.hpp.

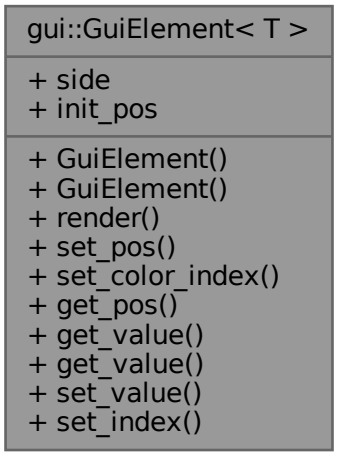
The documentation for this class was generated from the following file:

- 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
```

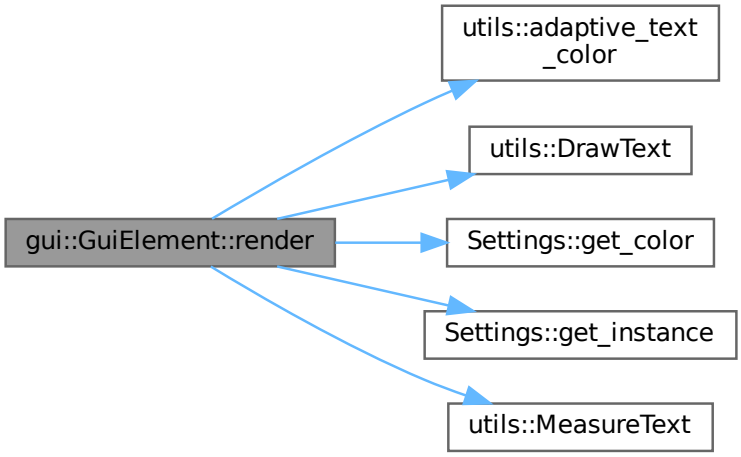
Definition at line 105 of file element_gui.hpp.

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:

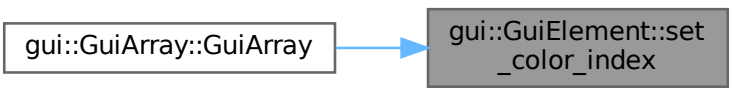


6.15.3.5 set_color_index()

```
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.

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.

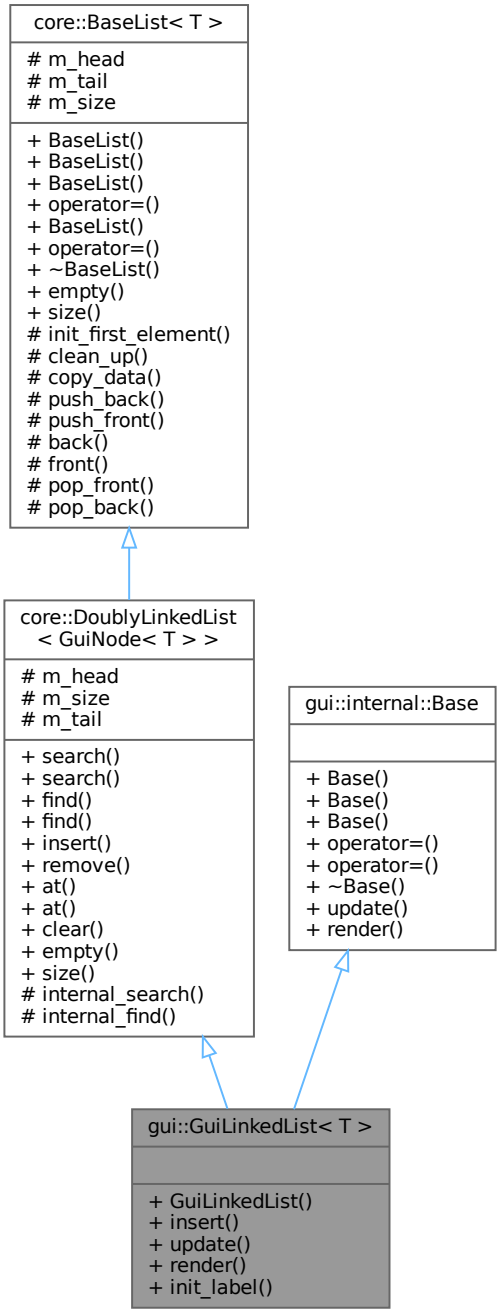
The documentation for this class was generated from the following file:

- src/gui/element_gui.hpp

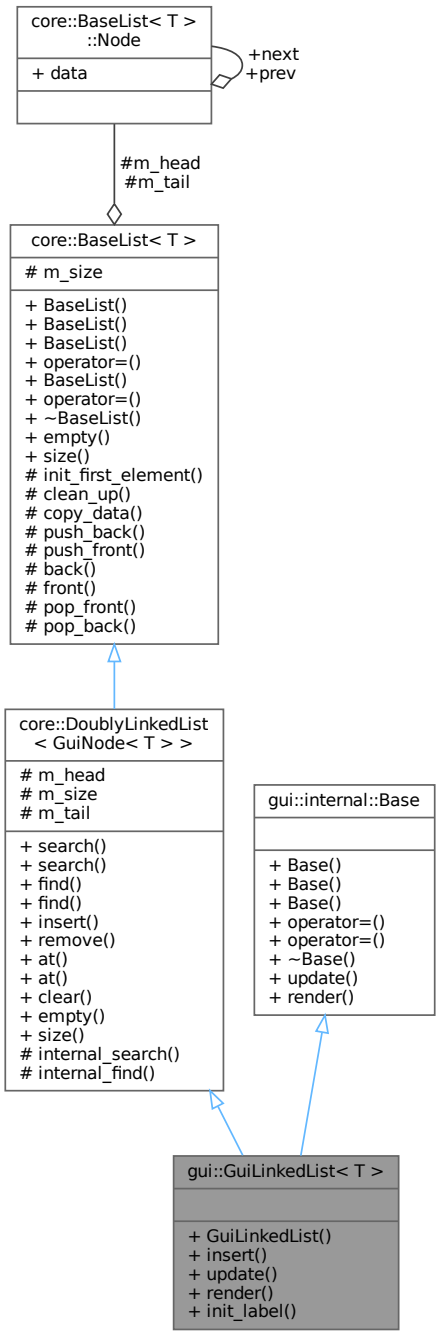
6.16 gui::GuiLinkedList< T > Class Template Reference

```
#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.

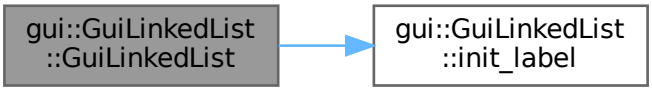
6.16.2 Constructor & Destructor Documentation

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:



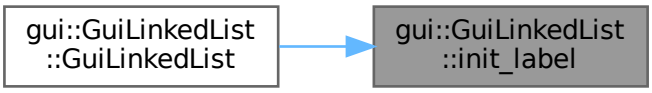
6.16.3 Member Function Documentation

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.

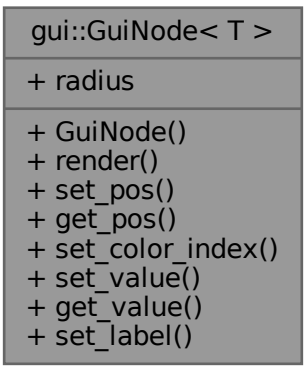
The documentation for this class was generated from the following file:

- 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
```

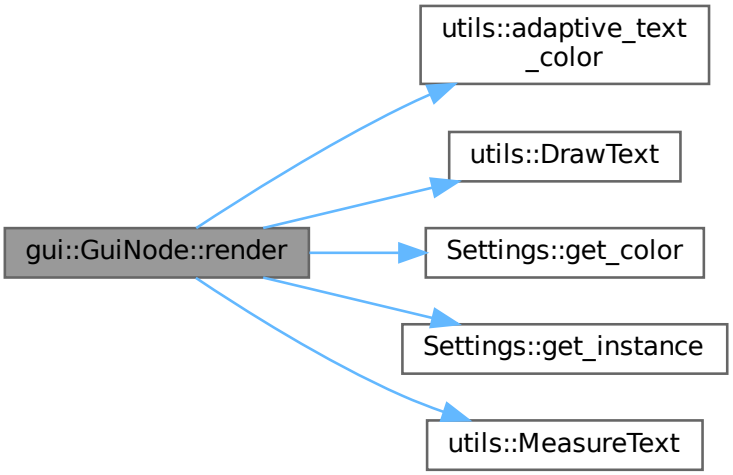
Definition at line 87 of file node_gui.hpp.

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.

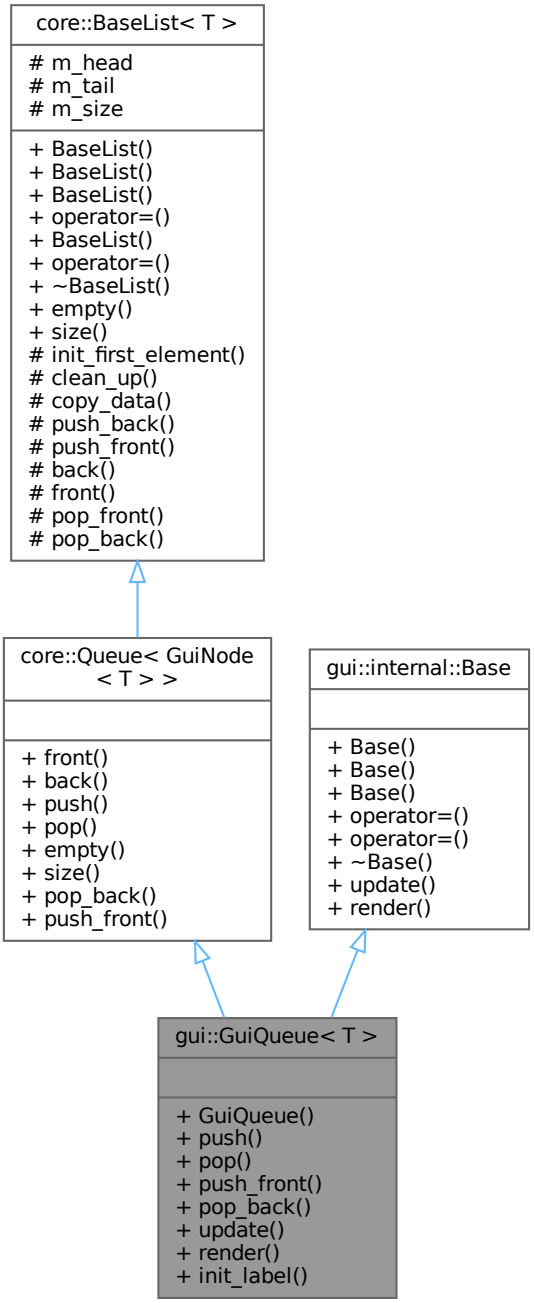
The documentation for this class was generated from the following file:

- src/gui/node_gui.hpp

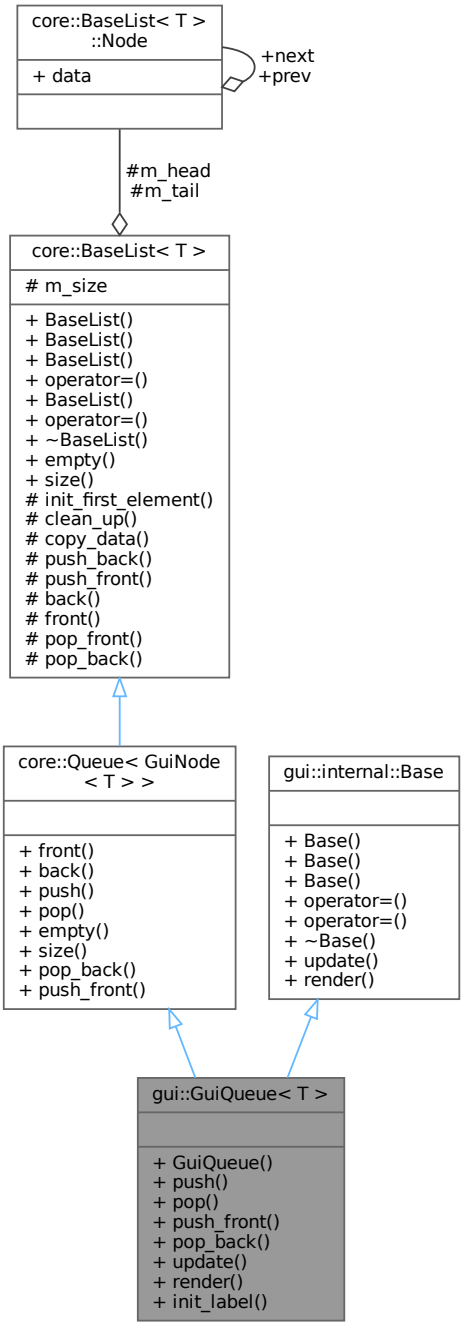
6.18 gui::GuiQueue< T > Class Template Reference

`#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.

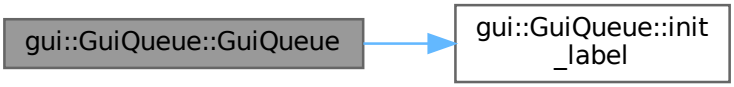
6.18.2 Constructor & Destructor Documentation

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:



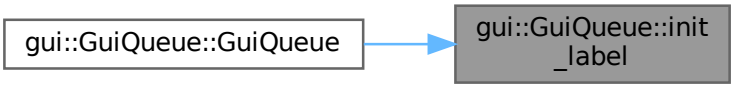
6.18.3 Member Function Documentation

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.

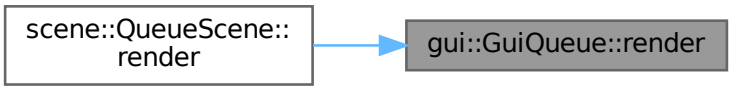
6.18.3.6 render()

```
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.

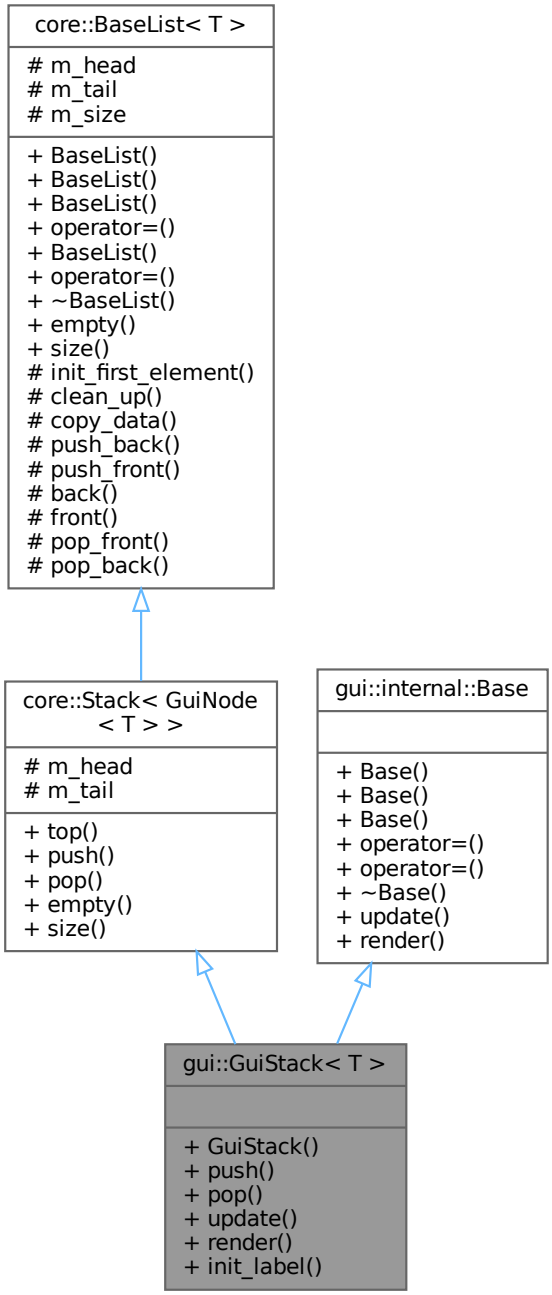
The documentation for this class was generated from the following file:

- src/gui/queue_gui.hpp

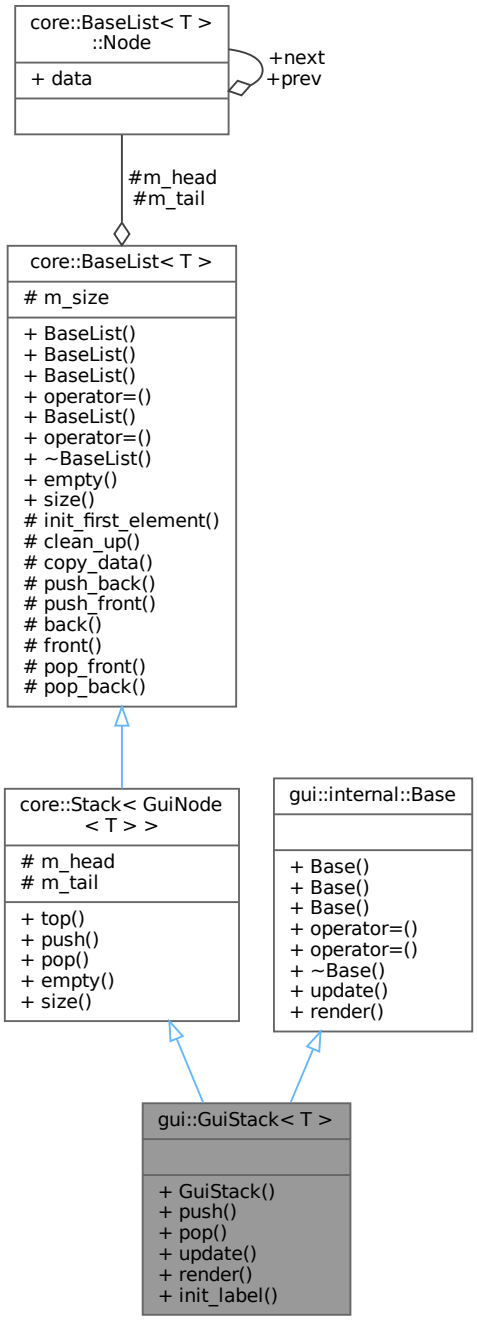
6.19 gui::GuiStack< T > Class Template Reference

`#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.

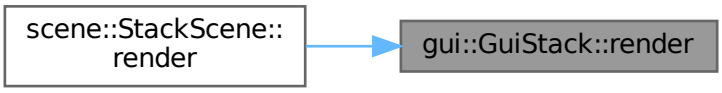
6.19.3.4 render()

```
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.

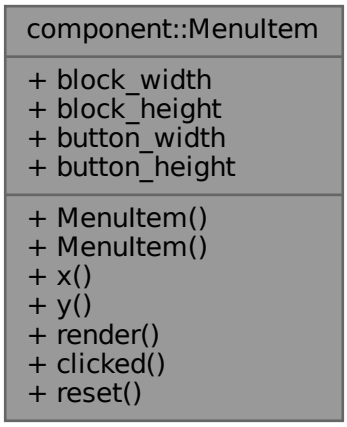
The documentation for this class was generated from the following file:

- 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.

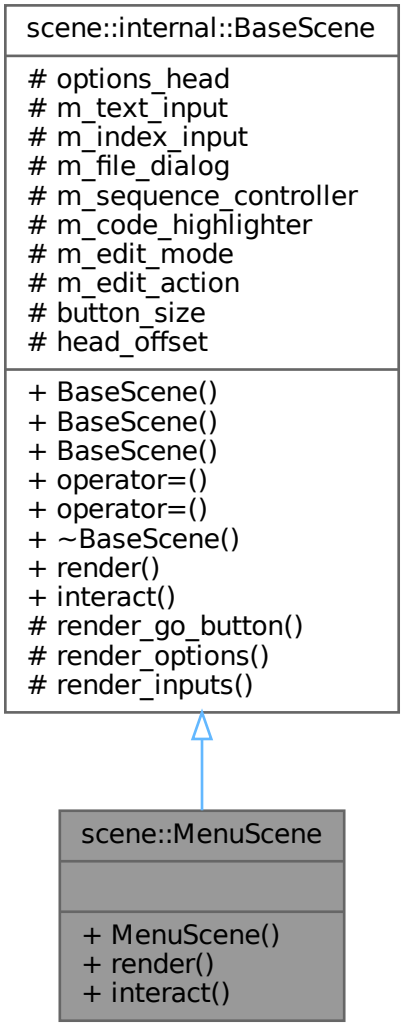
The documentation for this class was generated from the following files:

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

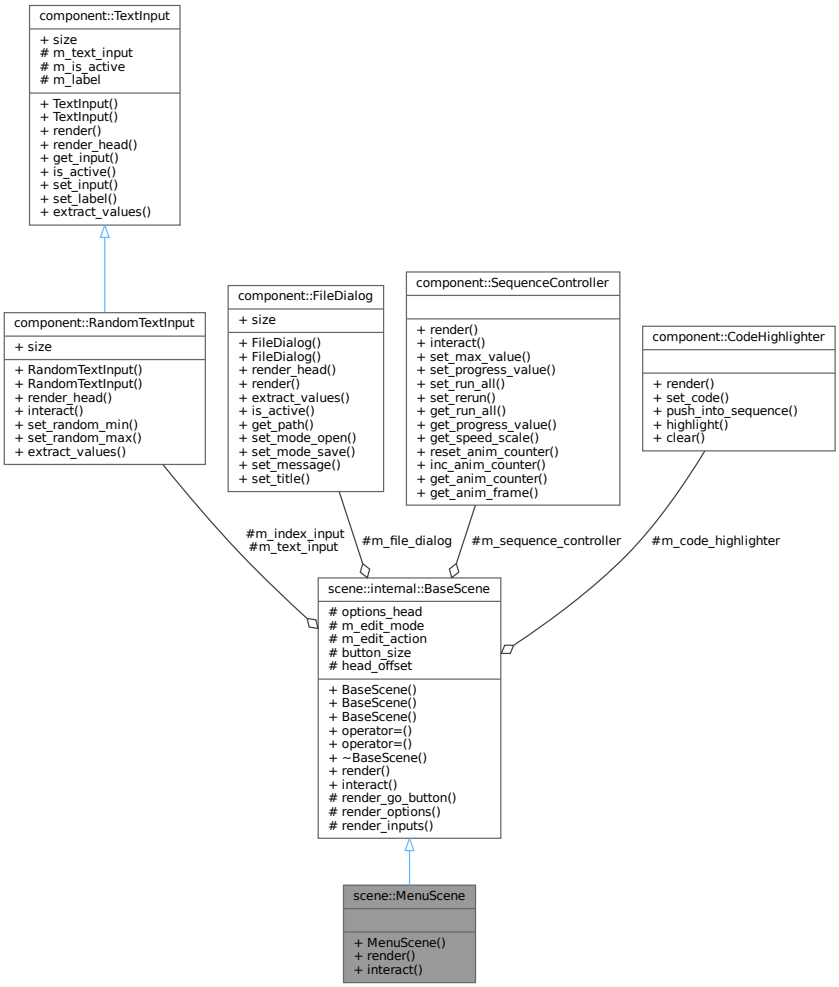
6.21 scene::MenuScene Class Reference

```
#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::RandomTextInput m_text_input {"value"}
- component::RandomTextInput m_index_input {"index"}
- component::FileDialog m_file_dialog
- component::SequenceController m_sequence_controller
- component::CodeHighlighter m_code_highlighter
- bool m_edit_mode {}
- bool m_edit_action {}

Static Protected Attributes inherited from scene::internal::BaseScene

- static constexpr Vector2 button_size {200, 50}
- static constexpr int head_offset = 20

6.21.1 Detailed Description

Definition at line 11 of file menu_scene.hpp.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 MenuScene()

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

Definition at line 14 of file menu_scene.cpp.

6.21.3 Member Function Documentation

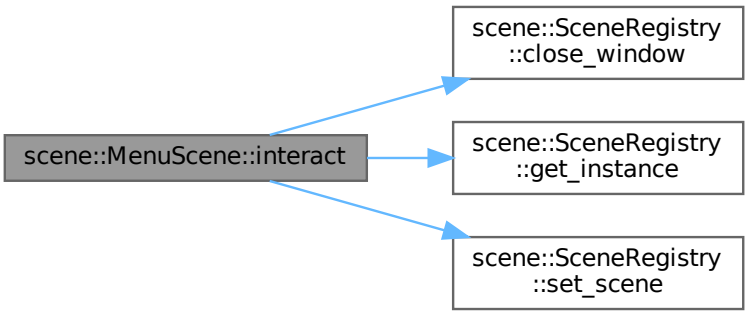
6.21.3.1 interact()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 125 of file menu_scene.cpp.

Here is the call graph for this function:



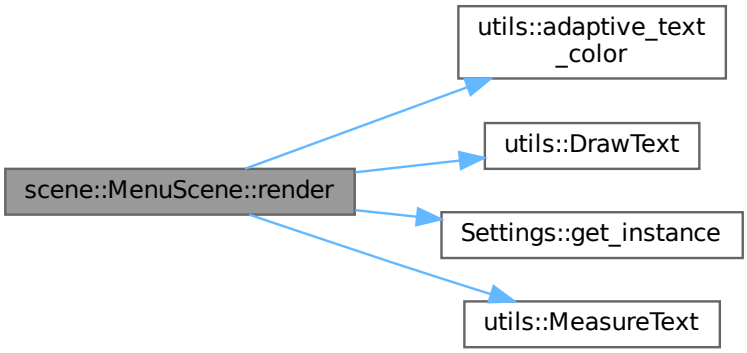
6.21.3.2 render()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 52 of file menu_scene.cpp.

Here is the call graph for this function:



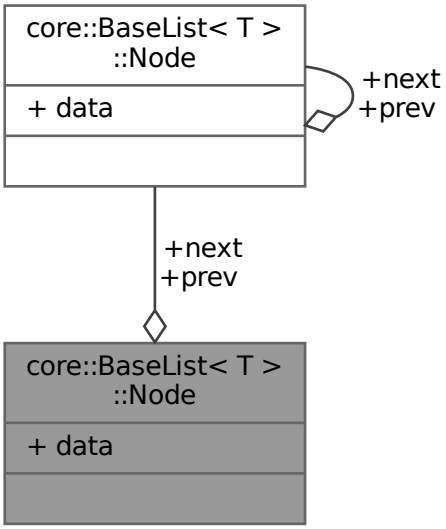
The documentation for this class was generated from the following files:

- src/scene/menu_scene.hpp
- src/scene/menu_scene.cpp

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

`#include <base_list.hpp>`

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



Public Attributes

- T data {}
- Node_ptr prev {}
- Node_ptr next {}

6.22.1 Detailed Description

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

Definition at line 16 of file base_list.hpp.

6.22.2 Member Data Documentation

6.22.2.1 data

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

Definition at line 17 of file base_list.hpp.

6.22.2.2 next

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

Definition at line 19 of file base_list.hpp.

6.22.2.3 prev

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

Definition at line 18 of file base_list.hpp.

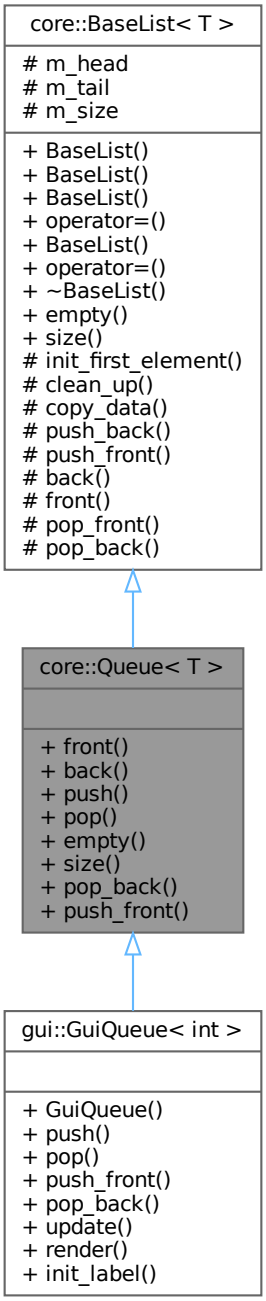
The documentation for this struct was generated from the following file:

- src/core/base_list.hpp

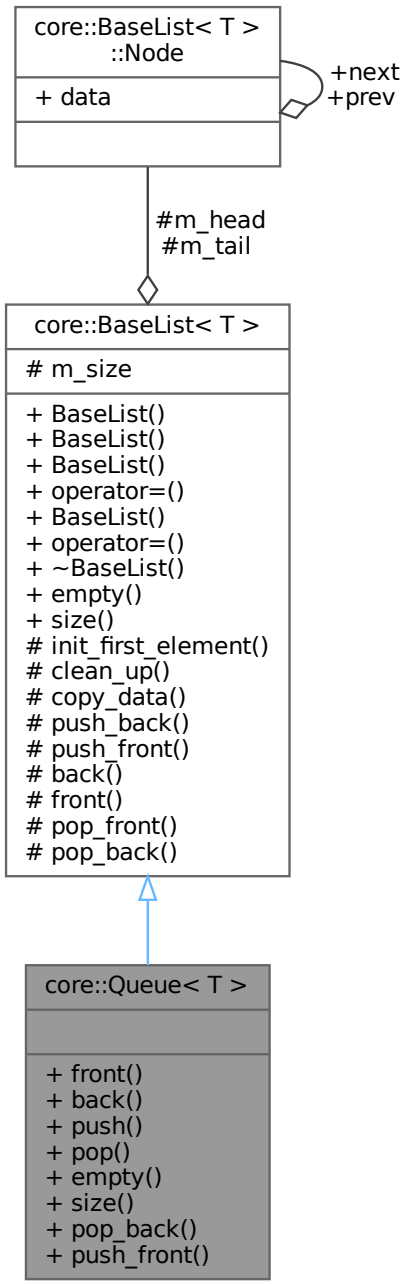
6.23 core::Queue< T > Class Template Reference

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

Inheritance diagram for core::Queue< T >:



Collaboration diagram for core::Queue< T >:



Public Member Functions

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

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

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

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

Additional Inherited Members

Protected Types inherited from core::BaseList< T >

- using Node_ptr = Node *

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

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

Protected Attributes inherited from core::BaseList< T >

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

6.23.1 Detailed Description

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

Definition at line 9 of file queue.hpp.

6.23.2 Member Function Documentation

6.23.2.1 back()

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

Definition at line 36 of file queue.hpp.

6.23.2.2 empty()

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

Definition at line 48 of file base_list.hpp.

6.23.2.3 front()

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

Definition at line 31 of file queue.hpp.

6.23.2.4 pop()

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

Definition at line 46 of file queue.hpp.

6.23.2.5 pop_back()

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

Definition at line 37 of file base_list.hpp.

6.23.2.6 push()

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

Definition at line 41 of file queue.hpp.

6.23.2.7 push_front()

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

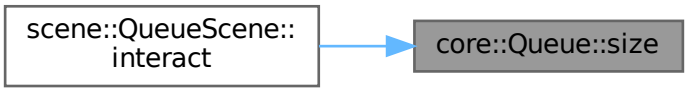
Definition at line 31 of file base_list.hpp.

6.23.2.8 size()

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

Definition at line 49 of file base_list.hpp.

Here is the caller graph for this function:



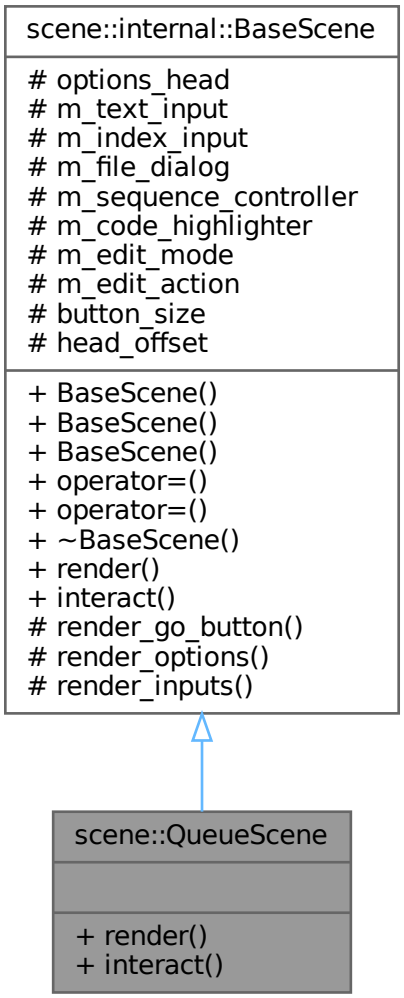
The documentation for this class was generated from the following file:

- src/core/queue.hpp

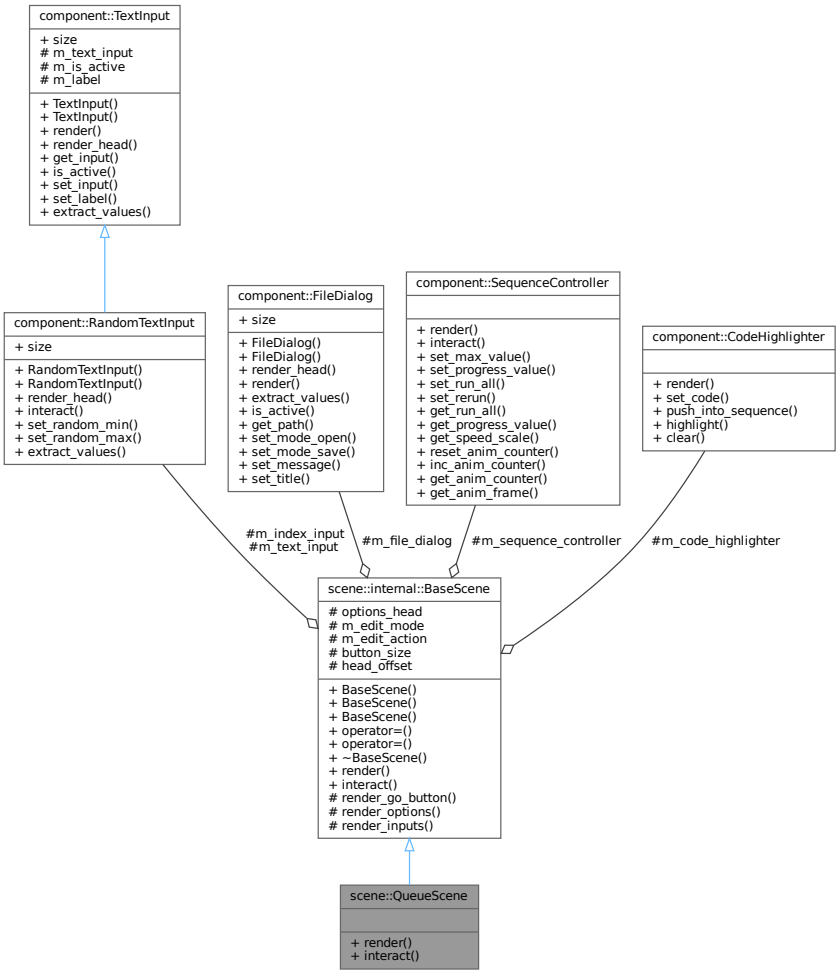
6.24 scene::QueueScene Class Reference

`#include <queue_scene.hpp>`

Inheritance diagram for scene::QueueScene:



Collaboration diagram for scene::QueueScene:



Public Member Functions

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

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

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

Additional Inherited Members

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

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

Protected Attributes inherited from scene::internal::BaseScene

- float options_head {}
- component::RandomTextInput m_text_input {"value"}
- component::RandomTextInput 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 15 of file queue_scene.hpp.

6.24.2 Member Function Documentation

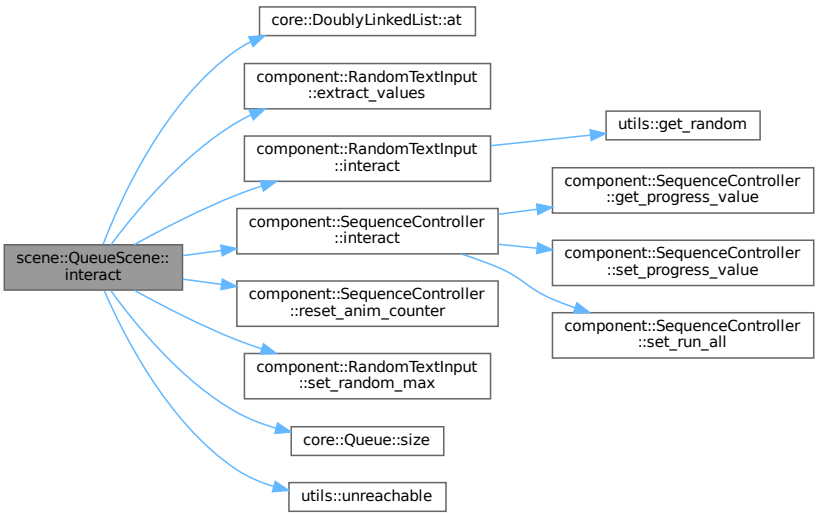
6.24.2.1 interact()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 71 of file queue_scene.cpp.

Here is the call graph for this function:



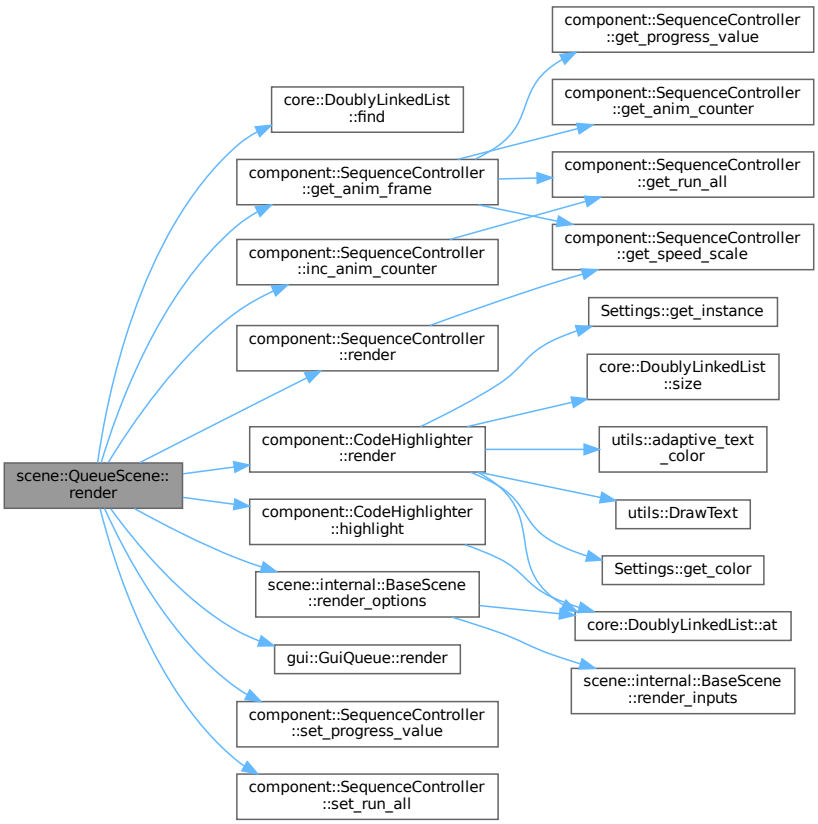
6.24.2.2 render()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 51 of file queue_scene.cpp.

Here is the call graph for this function:



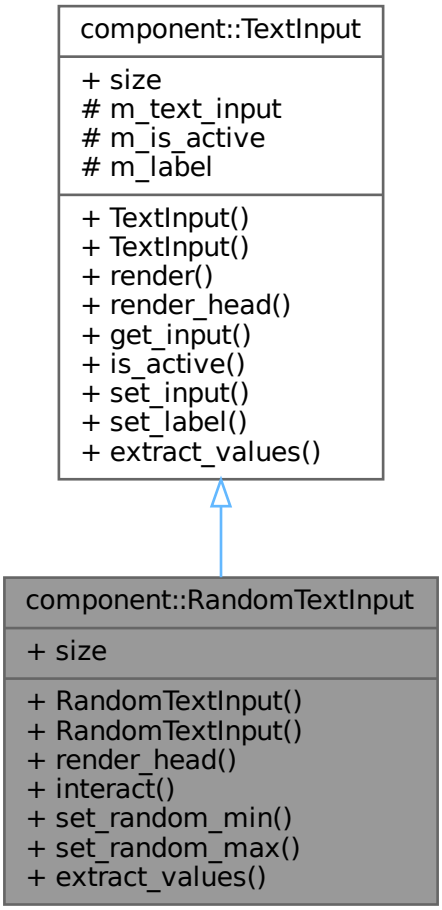
The documentation for this class was generated from the following files:

- src/scene/queue_scene.hpp
- src/scene/queue_scene.cpp

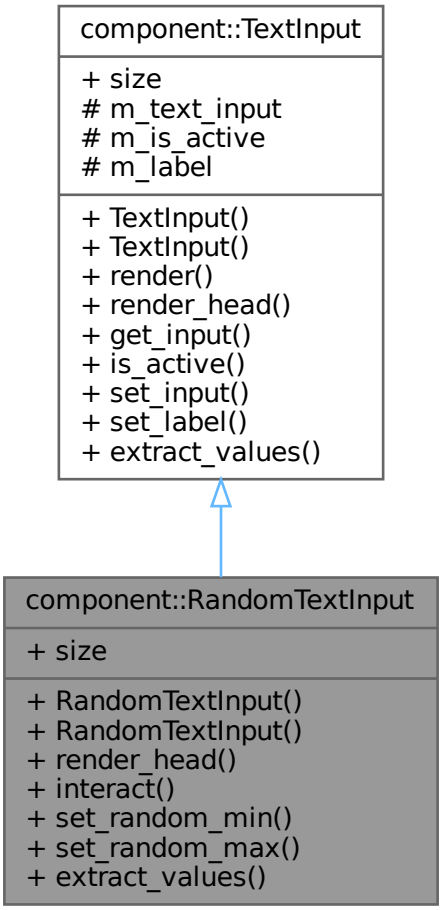
6.25 component::RandomTextInput Class Reference

`#include <random_text_input.hpp>`

Inheritance diagram for component::RandomTextInput:



Collaboration diagram for component::RandomTextInput:



Public Member Functions

- RandomTextInput ()=default
- RandomTextInput (const char *label)
- void render_head (float &options_head, float head_offset)
- bool interact ()
- void set_random_min (int value)
- void set_random_max (int value)
- core::Deque< int > extract_values ()

Public Member Functions inherited from component::TextInput

- TextInput ()=default
- TextInput (const char *label)
- void render (float x, float y)
- void render_head (float &options_head, float head_offset)

- std::string get_input () const
- bool is_active () const
- void set_input (const char *input, int len)
- void set_label (const char *const label)
- core::Deque< int > extract_values ()

Static Public Attributes

- static constexpr Vector2 size

Static Public Attributes inherited from component::TextInput

- static constexpr Vector2 size {200, 50}

Additional Inherited Members

Protected Attributes inherited from component::TextInput

- char m_text_input [constants::text_buffer_size] = ""
- bool m_is_active {}
- const char * m_label {}

6.25.1 Detailed Description

Definition at line 13 of file random_text_input.hpp.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 RandomTextInput() [1/2]

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

6.25.2.2 RandomTextInput() [2/2]

```
component::RandomTextInput::RandomTextInput (
            const char * label )
```

Definition at line 14 of file random_text_input.cpp.

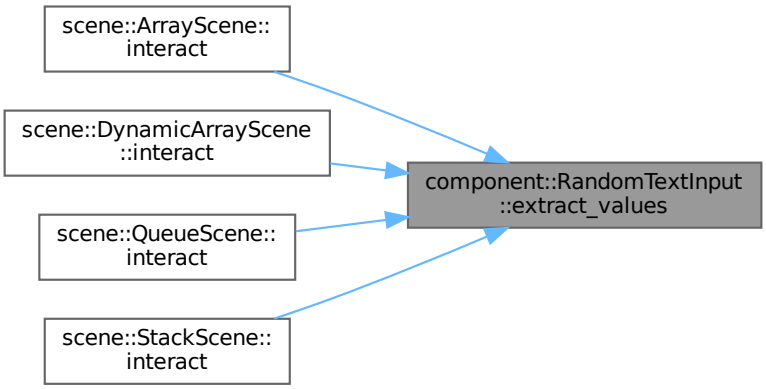
6.25.3 Member Function Documentation

6.25.3.1 extract_values()

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

Definition at line 30 of file text_input.cpp.

Here is the caller graph for this function:

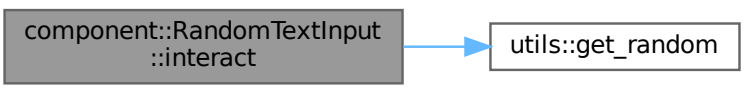


6.25.3.2 interact()

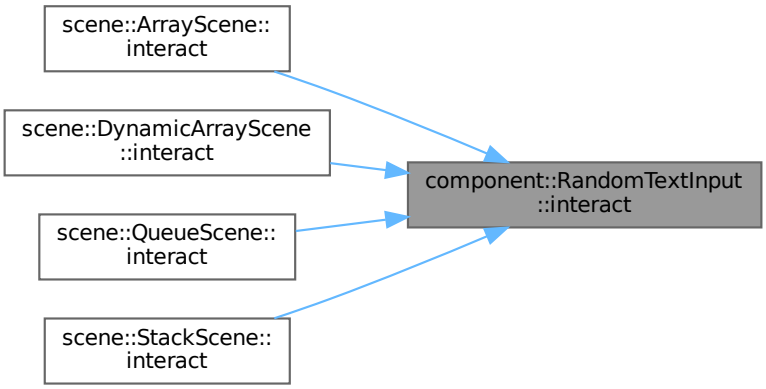
```
bool component::RandomTextInput::interact ( )
```

Definition at line 30 of file random_text_input.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

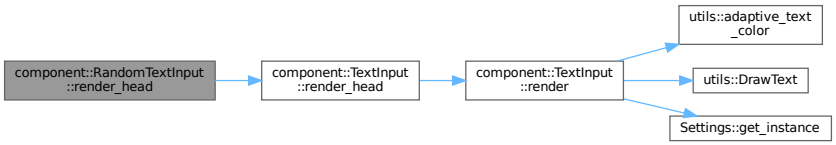


6.25.3.3 render_head()

```
void component::RandomTextInput::render_head (
            float & options_head,
            float head_offset )
```

Definition at line 20 of file random_text_input.cpp.

Here is the call graph for this function:

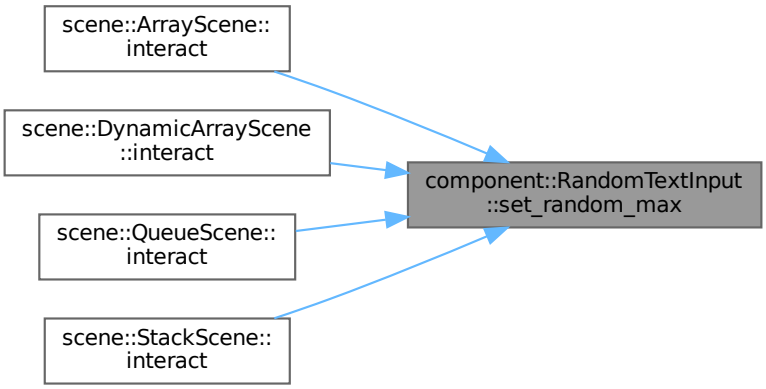


6.25.3.4 set_random_max()

```
void component::RandomTextInput::set_random_max (
            int value )
```

Definition at line 18 of file random_text_input.cpp.

Here is the caller graph for this function:



6.25.3.5 set_random_min()

```
void component::RandomTextInput::set_random_min (
            int value )
```

Definition at line 16 of file random_text_input.cpp.

6.25.4 Member Data Documentation

6.25.4.1 size

```
constexpr Vector2 component::TextInput::size  [static], [constexpr]
```

Definition at line 19 of file text_input.hpp.

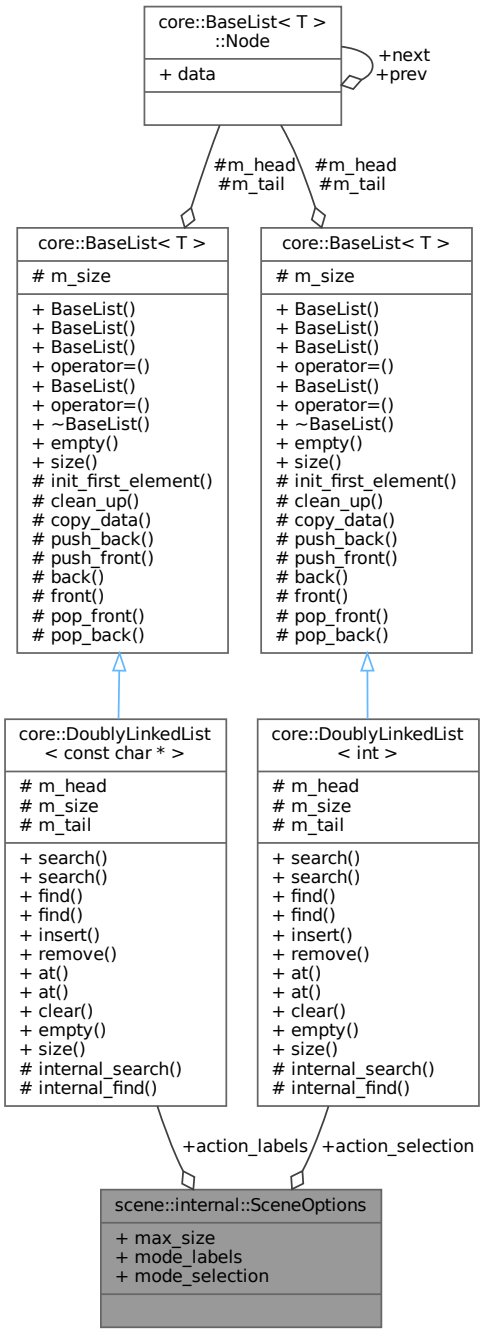
The documentation for this class was generated from the following files:

- src/component/random_text_input.hpp
- src/component/random_text_input.cpp

6.26 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

6.26.1 Detailed Description

Definition at line 10 of file scene_options.hpp.

6.26.2 Member Data Documentation

6.26.2.1 action_labels

```
core::DoublyLinkedList<const char*> scene::internal::SceneOptions::action_labels
```

Definition at line 14 of file scene_options.hpp.

6.26.2.2 action_selection

```
core::DoublyLinkedList<int> scene::internal::SceneOptions::action_selection
```

Definition at line 15 of file scene_options.hpp.

6.26.2.3 max_size

```
const std::size_t scene::internal::SceneOptions::max_size {}
```

Definition at line 11 of file scene_options.hpp.

6.26.2.4 mode_labels

```
const char* scene::internal::SceneOptions::mode_labels {}
```

Definition at line 12 of file scene_options.hpp.

6.26.2.5 mode_selection

```
int scene::internal::SceneOptions::mode_selection {}
```

Definition at line 13 of file scene_options.hpp.

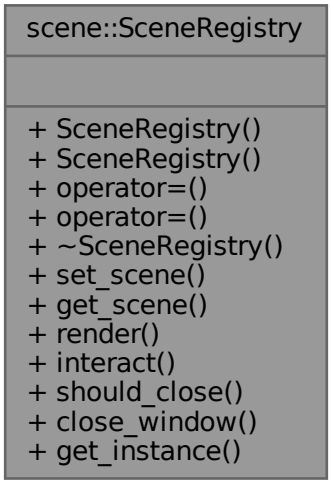
The documentation for this struct was generated from the following file:

- src/scene/scene_options.hpp

6.27 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 ()

6.27.1 Detailed Description

Definition at line 30 of file scene_registry.hpp.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 SceneRegistry() [1/2]

```
scene::SceneRegistry::SceneRegistry (
            const SceneRegistry &  )  [delete]
```

6.27.2.2 SceneRegistry() [2/2]

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

6.27.2.3 ~SceneRegistry()

```
scene::SceneRegistry::~SceneRegistry ( )  [default]
```

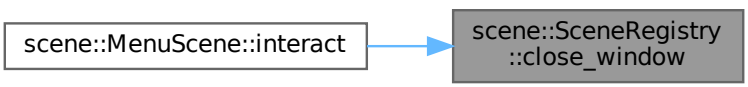
6.27.3 Member Function Documentation

6.27.3.1 close_window()

```
void scene::SceneRegistry::close_window ( )
```

Definition at line 25 of file scene_registry.cpp.

Here is the caller graph for this function:

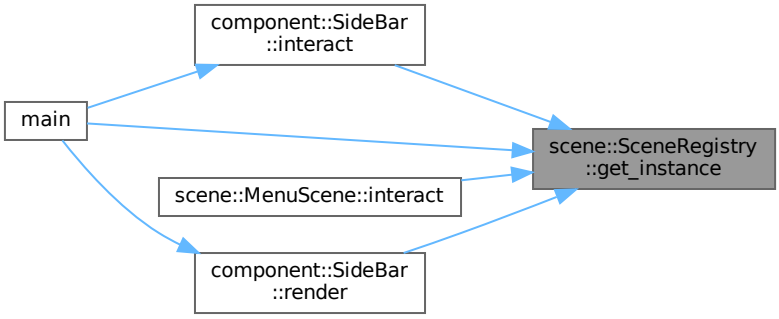


6.27.3.2 get_instance()

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

Definition at line 7 of file scene_registry.cpp.

Here is the caller graph for this function:

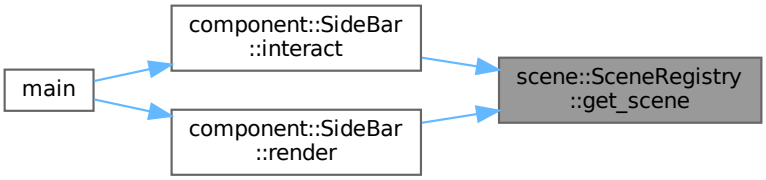


6.27.3.3 get_scene()

```
int scene::SceneRegistry::get_scene ( ) const
```

Definition at line 17 of file scene_registry.cpp.

Here is the caller graph for this function:

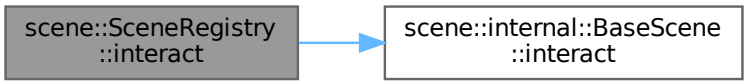


6.27.3.4 interact()

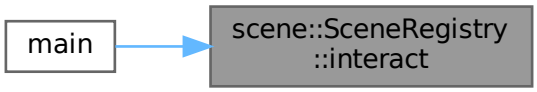
```
void scene::SceneRegistry::interact ( )
```

Definition at line 21 of file scene_registry.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.27.3.5 operator=() [1/2]

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

6.27.3.6 operator=() [2/2]

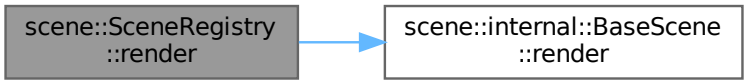
```
SceneRegistry & scene::SceneRegistry::operator= (
            SceneRegistry &&  )  [delete]
```

6.27.3.7 render()

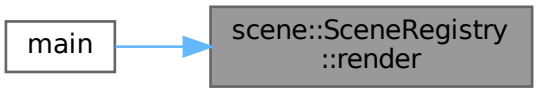
```
void scene::SceneRegistry::render ( )
```

Definition at line 19 of file scene_registry.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

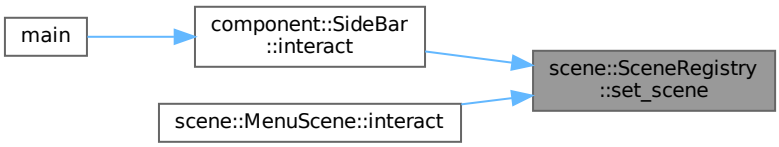


6.27.3.8 set_scene()

```
void scene::SceneRegistry::set_scene (
            int scene_type )
```

Definition at line 12 of file scene_registry.cpp.

Here is the caller graph for this function:

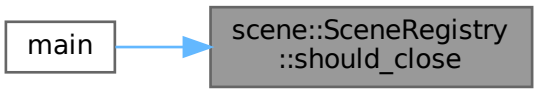


6.27.3.9 should_close()

```
bool scene::SceneRegistry::should_close ( ) const
```

Definition at line 23 of file scene_registry.cpp.

Here is the caller graph for this function:



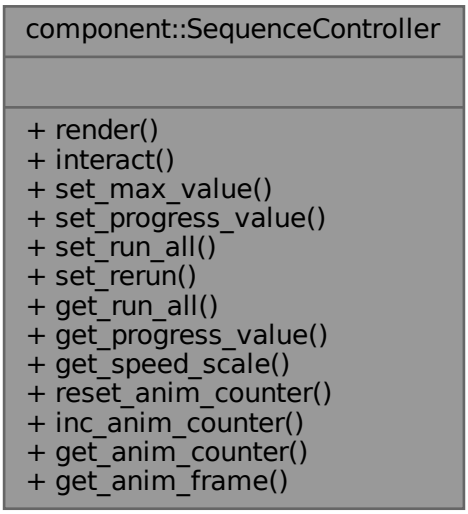
The documentation for this class was generated from the following files:

- src/scene/scene_registry.hpp
- src/scene/scene_registry.cpp

6.28 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

6.28.1 Detailed Description

Definition at line 8 of file sequence_controller.hpp.

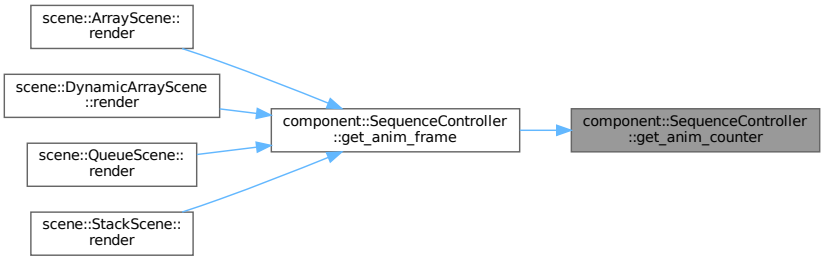
6.28.2 Member Function Documentation

6.28.2.1 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:

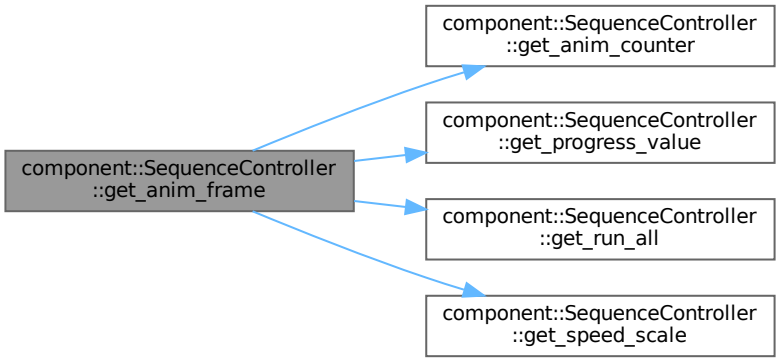


6.28.2.2 get_anim_frame()

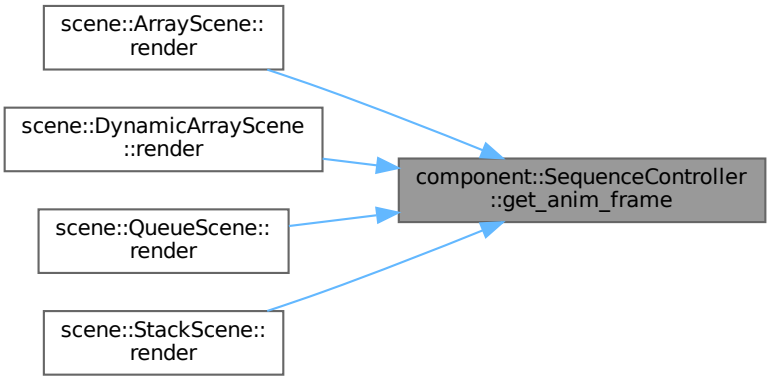
```
int component::SequenceController::get_anim_frame ( ) const
```

Definition at line 42 of file sequence_controller.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

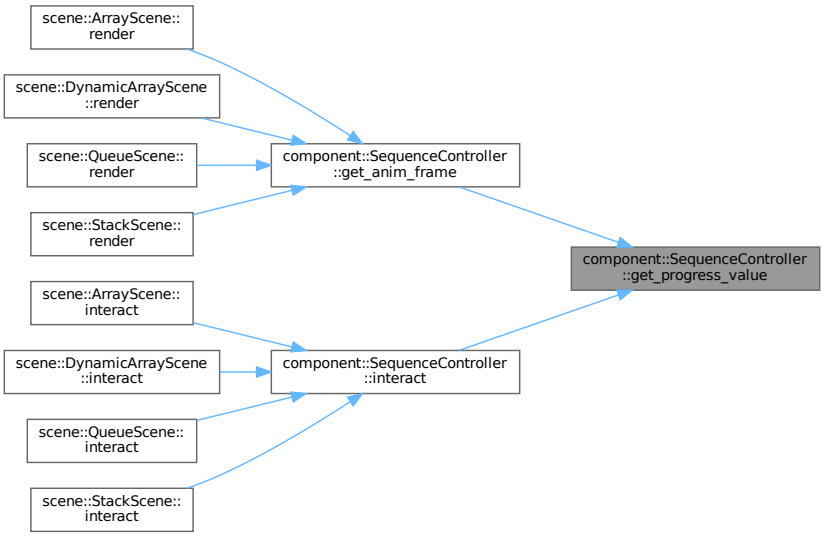


6.28.2.3 get_progress_value()

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

Definition at line 21 of file sequence_controller.cpp.

Here is the caller graph for this function:

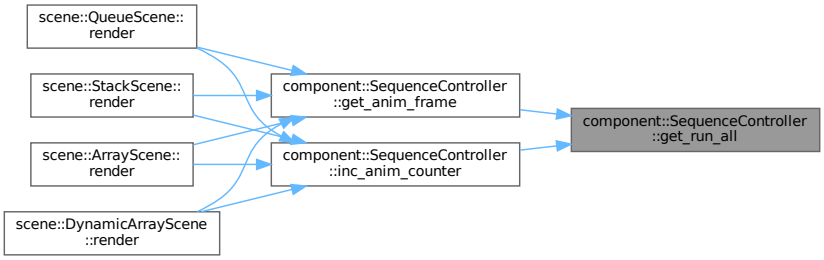


6.28.2.4 get_run_all()

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

Definition at line 19 of file sequence_controller.cpp.

Here is the caller graph for this function:

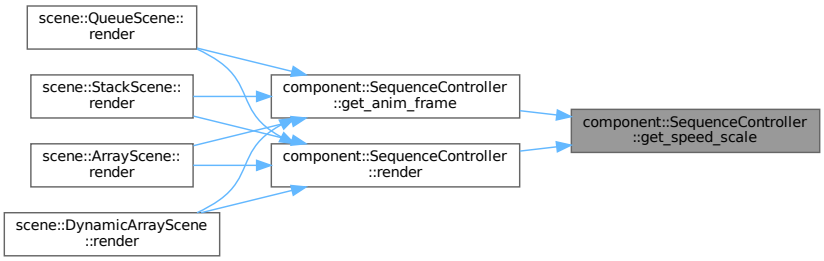


6.28.2.5 get_speed_scale()

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

Definition at line 23 of file sequence_controller.cpp.

Here is the caller graph for this function:

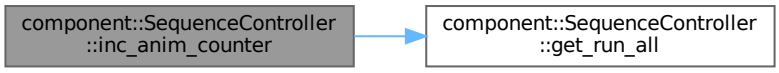


6.28.2.6 inc_anim_counter()

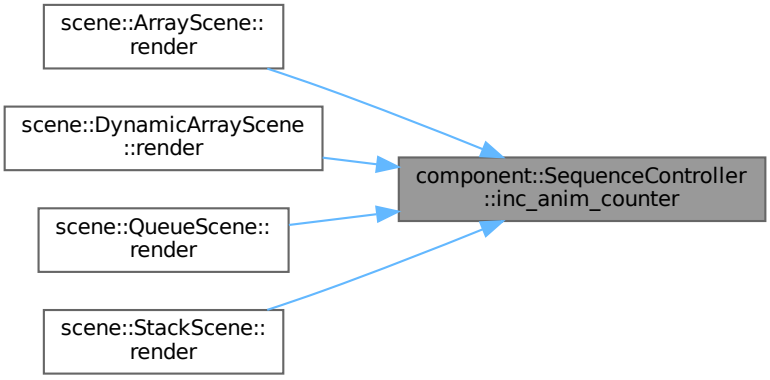
```
void component::SequenceController::inc_anim_counter ( )
```

Definition at line 29 of file sequence_controller.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

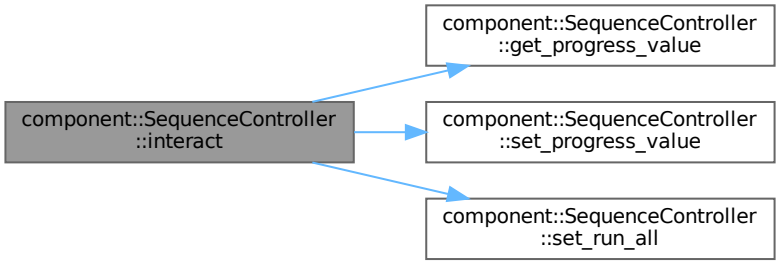


6.28.2.7 interact()

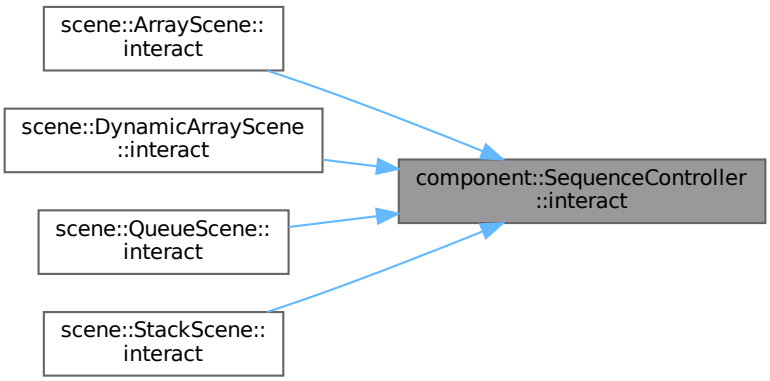
```
bool component::SequenceController::interact ( )
```

Definition at line 90 of file sequence_controller.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

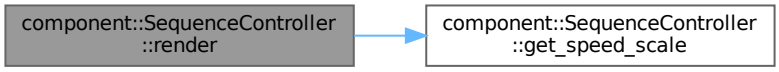


6.28.2.8 render()

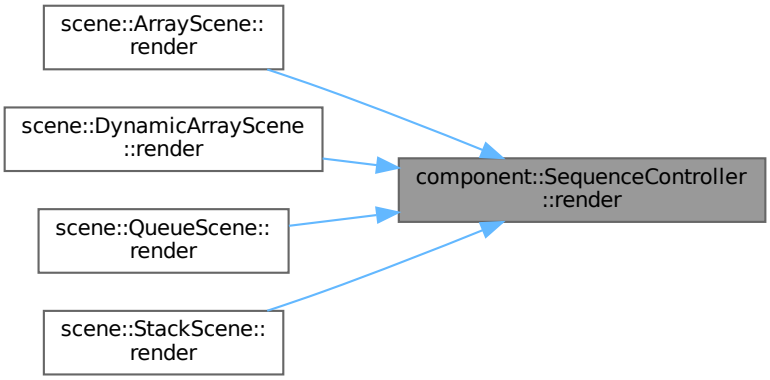
```
void component::SequenceController::render ( )
```

Definition at line 51 of file sequence_controller.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

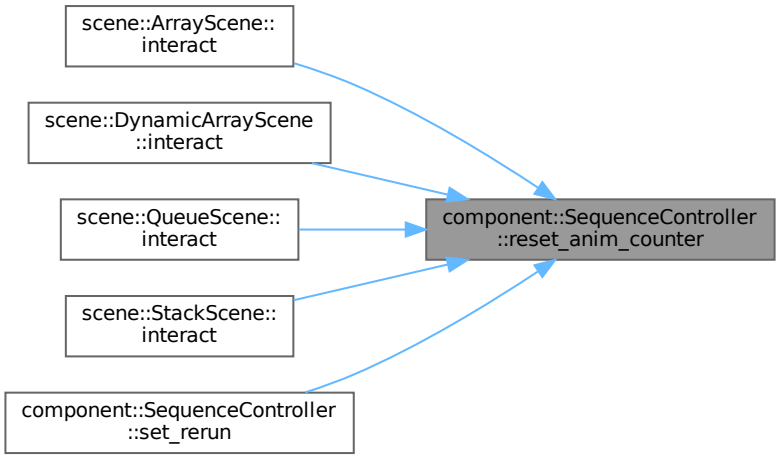


6.28.2.9 reset_anim_counter()

```
void component::SequenceController::reset_anim_counter ( )
```

Definition at line 27 of file sequence_controller.cpp.

Here is the caller graph for this function:



6.28.2.10 set_max_value()

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

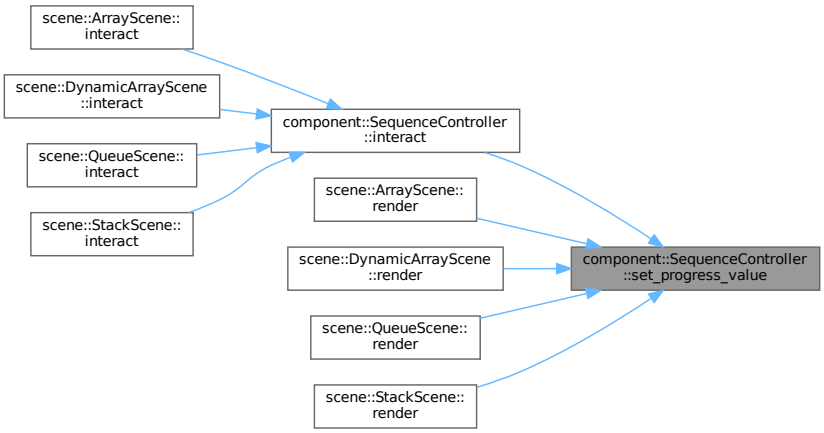
Definition at line 11 of file sequence_controller.cpp.

6.28.2.11 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:

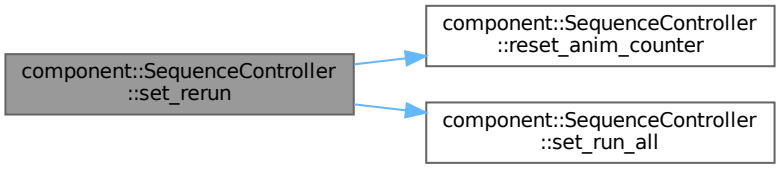


6.28.2.12 set_rerun()

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

Definition at line 37 of file sequence_controller.cpp.

Here is the call graph for this function:

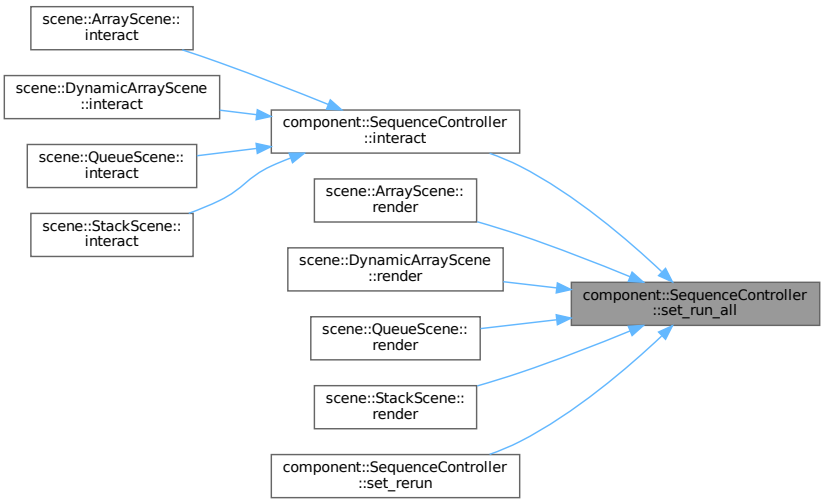


6.28.2.13 set_run_all()

```
void component::SequenceController::set_run_all (
            bool run_all )
```

Definition at line 17 of file sequence_controller.cpp.

Here is the caller graph for this function:



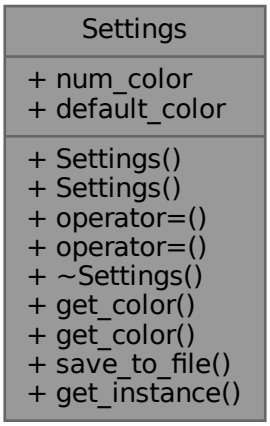
The documentation for this class was generated from the following files:

- src/component/sequence_controller.hpp
- src/component/sequence_controller.cpp

6.29 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

6.29.1 Detailed Description

Definition at line 10 of file settings.hpp.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 Settings() [1/2]

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

6.29.2.2 Settings() [2/2]

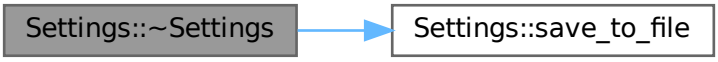
```
Settings::Settings (
            Settings &&  )  [delete]
```

6.29.2.3 ~Settings()

```
Settings::~Settings ( )
```

Definition at line 24 of file settings.cpp.

Here is the call graph for this function:



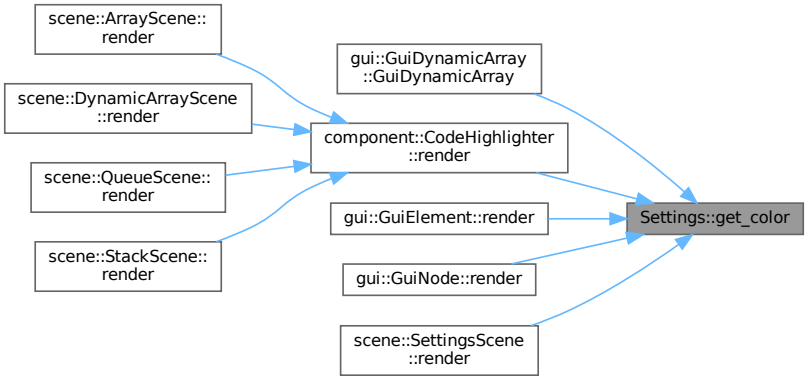
6.29.3 Member Function Documentation

6.29.3.1 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:



6.29.3.2 get_color() [2/2]

```
Color Settings::get_color (
            std::size_t index ) const
```

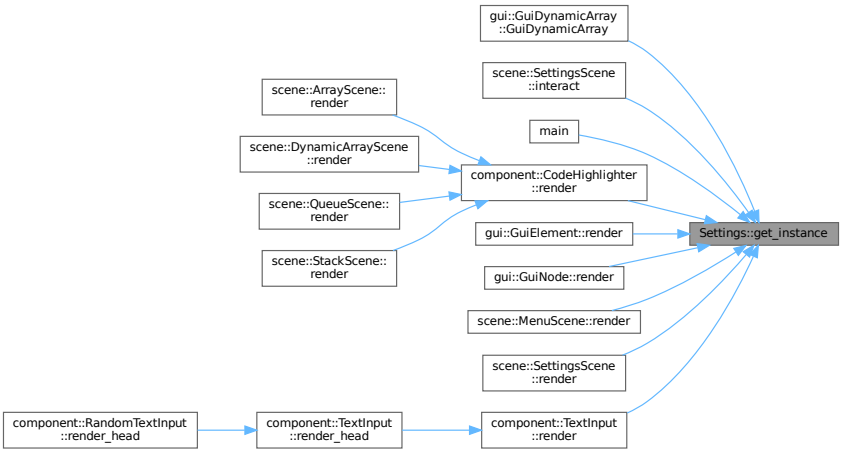
Definition at line 28 of file settings.cpp.

6.29.3.3 get_instance()

```
Settings & Settings::get_instance ( )  [static]
```

Definition at line 10 of file settings.cpp.

Here is the caller graph for this function:



6.29.3.4 operator=() [1/2]

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

6.29.3.5 operator=() [2/2]

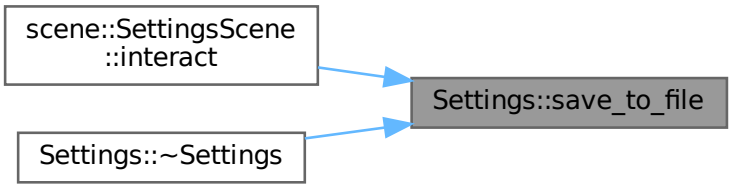
```
Settings & Settings::operator= (
            Settings &&  )  [delete]
```

6.29.3.6 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:



6.29.4 Member Data Documentation

6.29.4.1 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.

6.29.4.2 num_color

```
constexpr int Settings::num_color = 9  [static], [constexpr]
```

Definition at line 12 of file settings.hpp.

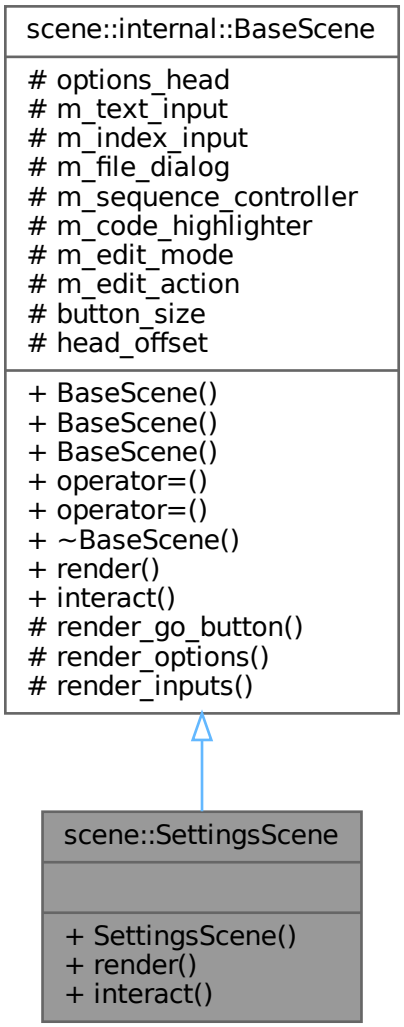
The documentation for this class was generated from the following files:

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

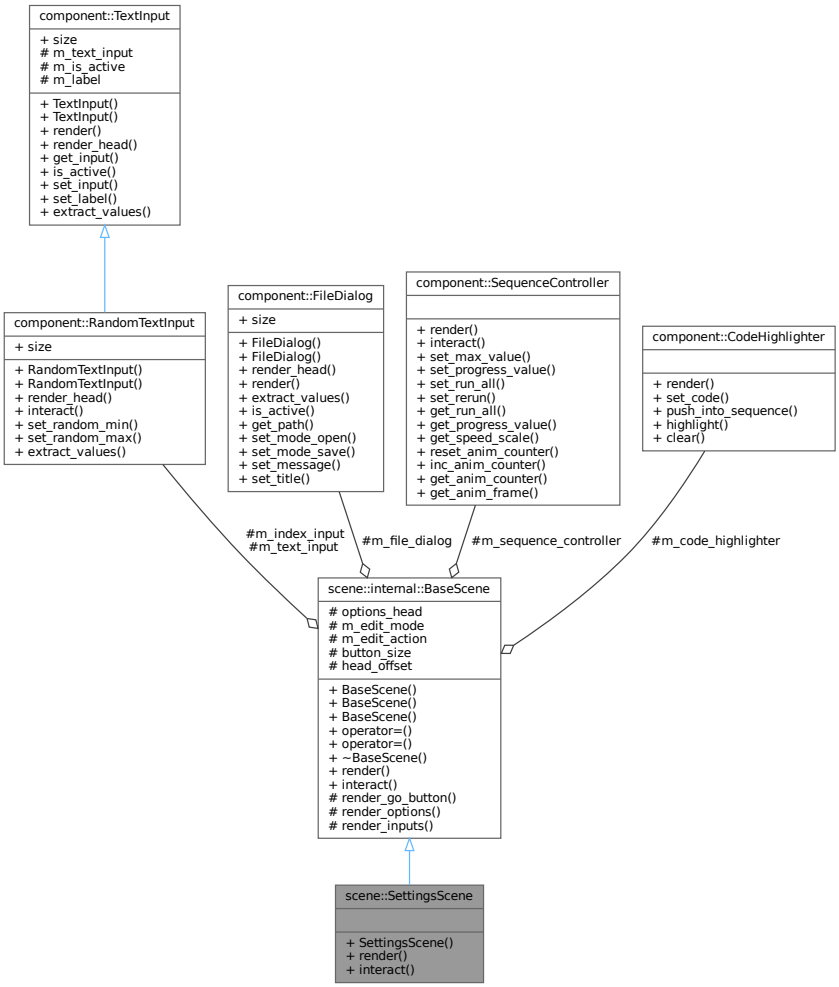
6.30 scene::SettingsScene Class Reference

`#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::RandomTextInput m_text_input {"value"}
- component::RandomTextInput 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

6.30.1 Detailed Description

Definition at line 16 of file settings_scene.hpp.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 SettingsScene()

```
scene::SettingsScene::SettingsScene ( )
```

Definition at line 47 of file settings_scene.cpp.

6.30.3 Member Function Documentation

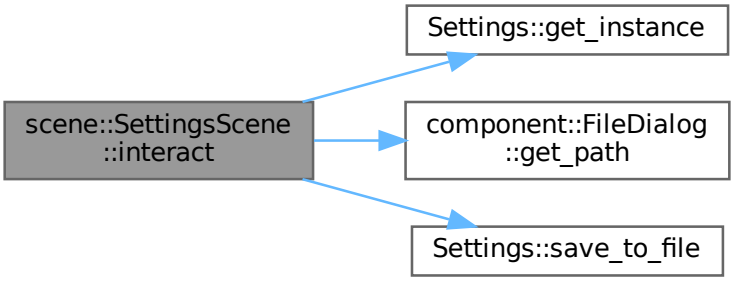
6.30.3.1 interact()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 144 of file settings_scene.cpp.

Here is the call graph for this function:



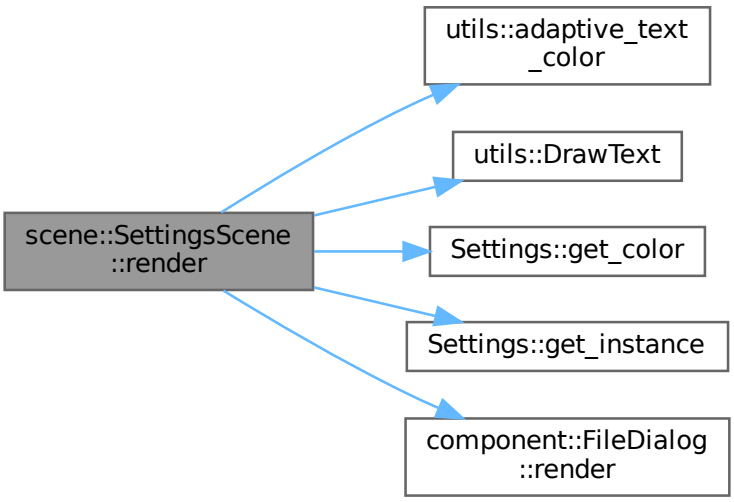
6.30.3.2 render()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 70 of file settings_scene.cpp.

Here is the call graph for this function:



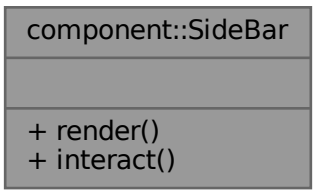
The documentation for this class was generated from the following files:

- src/scene/settings_scene.hpp
- src/scene/settings_scene.cpp

6.31 component::SideBar Class Reference

`#include <sidebar.hpp>`

Collaboration diagram for component::SideBar:



Public Member Functions

- void render ()
- void interact ()

6.31.1 Detailed Description

Definition at line 11 of file sidebar.hpp.

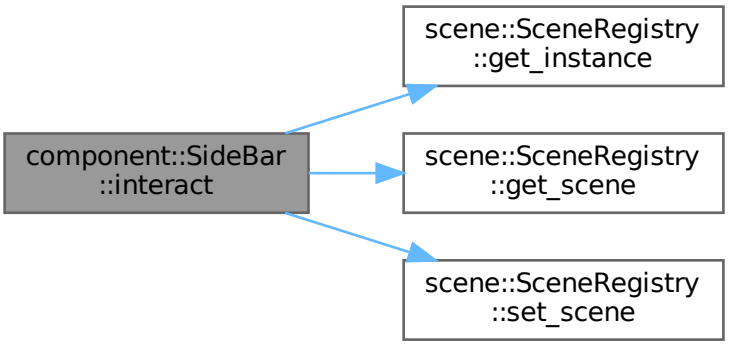
6.31.2 Member Function Documentation

6.31.2.1 interact()

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

Definition at line 48 of file sidebar.cpp.

Here is the call graph for this function:



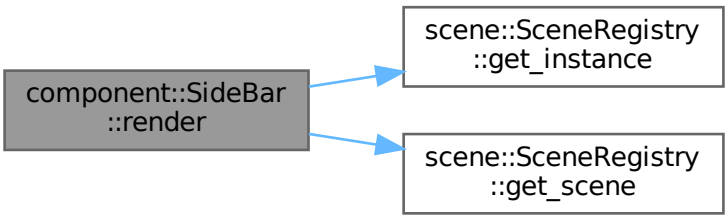
Here is the caller graph for this function:

6.31.2.2 render()

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

Definition at line 11 of file sidebar.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

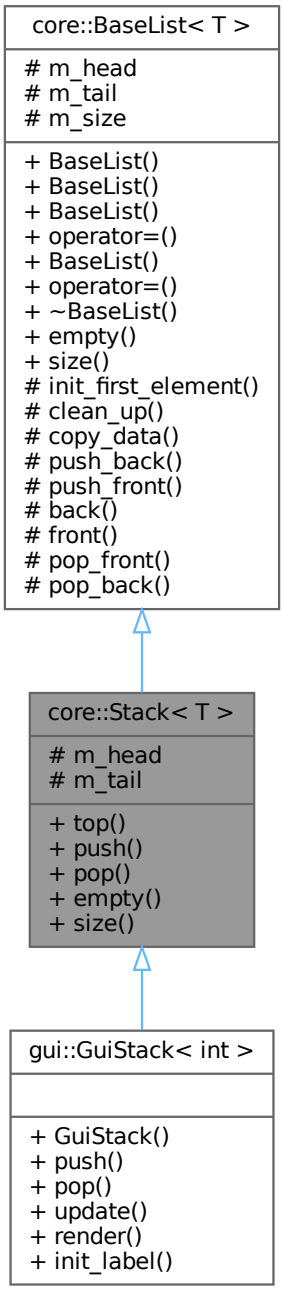
The documentation for this class was generated from the following files:

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

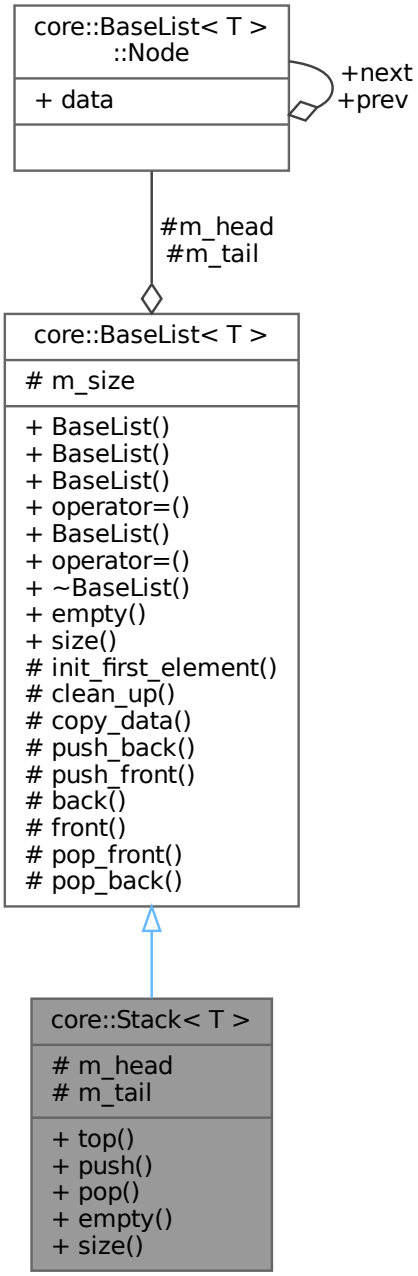
6.32 core::Stack< T > Class Template Reference

```
#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 ()

6.32.1 Detailed Description

template<typename T>
class core::Stack< T >

Definition at line 9 of file stack.hpp.

6.32.2 Member Typedef Documentation

6.32.2.1 Base

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

Definition at line 11 of file stack.hpp.

6.32.3 Member Function Documentation

6.32.3.1 empty()

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

Definition at line 48 of file base_list.hpp.

6.32.3.2 pop()

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

Definition at line 38 of file stack.hpp.

6.32.3.3 push()

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

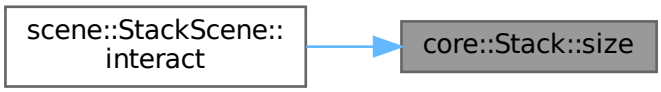
Definition at line 33 of file stack.hpp.

6.32.3.4 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:



6.32.3.5 top()

```
template<typename T >
T & core::Stack< T >::top
```

Definition at line 28 of file stack.hpp.

6.32.4 Member Data Documentation

6.32.4.1 m_head

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

Definition at line 22 of file base_list.hpp.

6.32.4.2 m_tail

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

Definition at line 23 of file base_list.hpp.

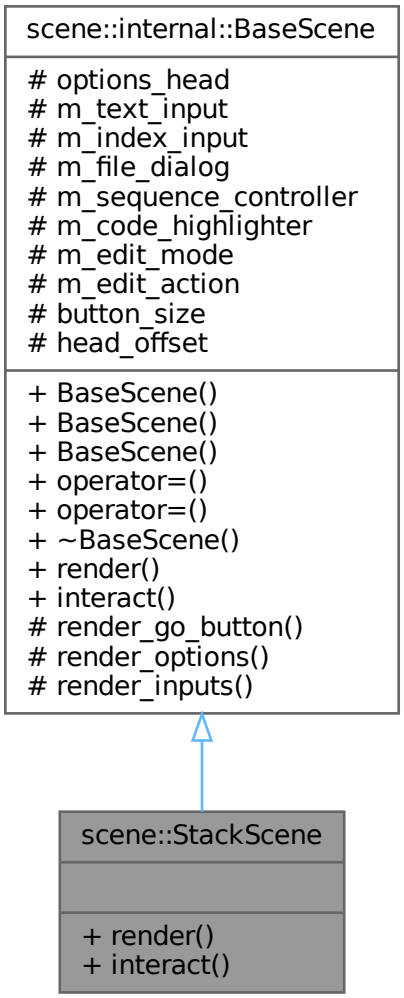
The documentation for this class was generated from the following file:

- src/core/stack.hpp

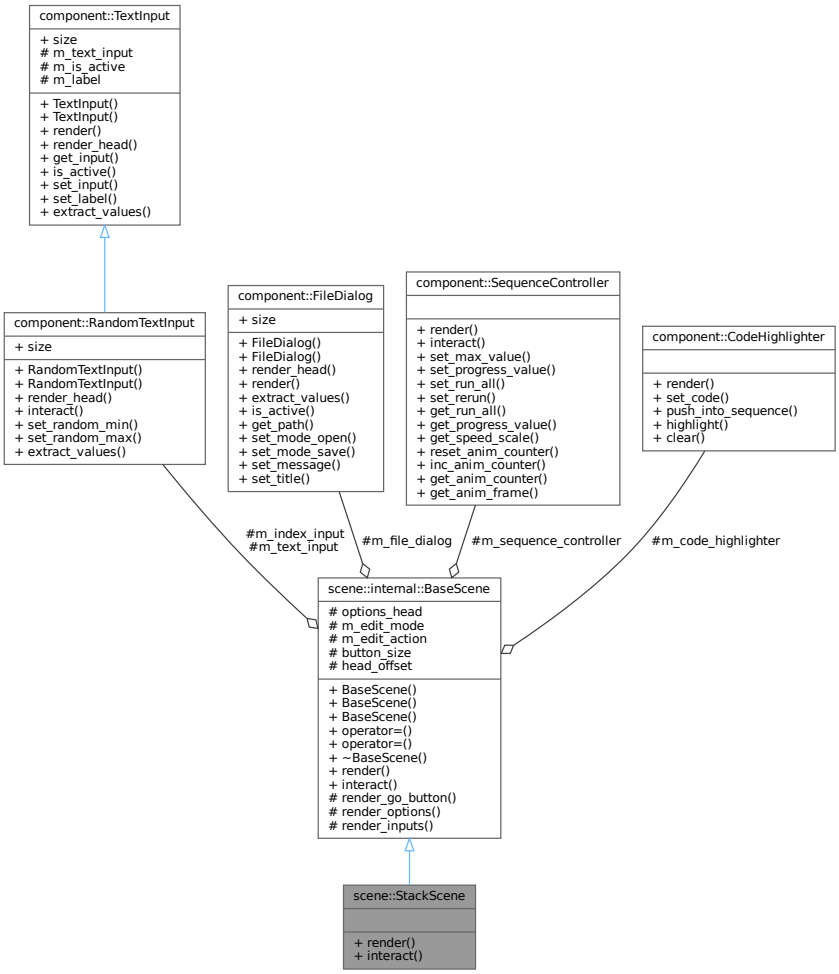
6.33 scene::StackScene Class Reference

`#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::RandomTextInput m_text_input {"value"}
- component::RandomTextInput 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

6.33.1 Detailed Description

Definition at line 13 of file stack_scene.hpp.

6.33.2 Member Function Documentation

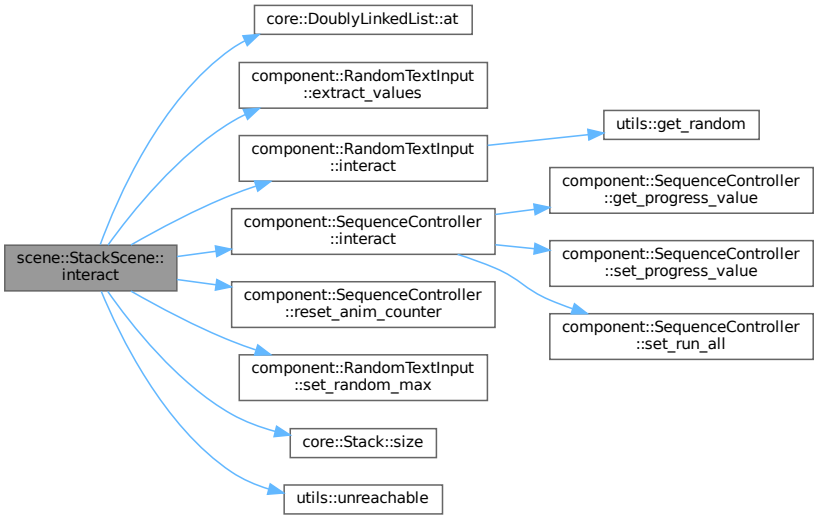
6.33.2.1 interact()

```
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:



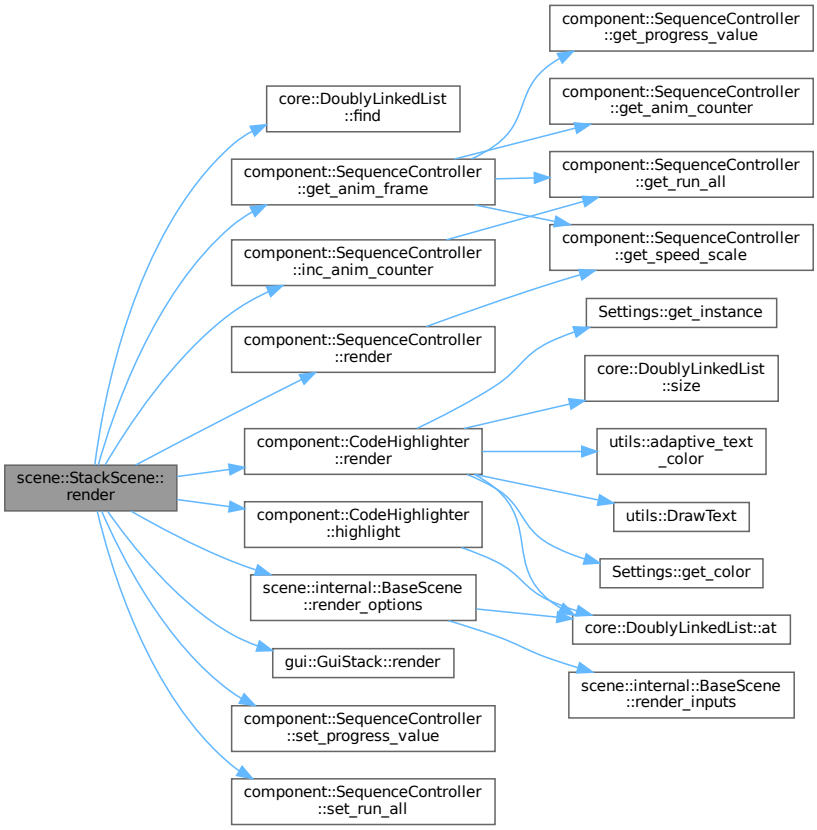
6.33.2.2 render()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 17 of file stack_scene.cpp.

Here is the call graph for this function:



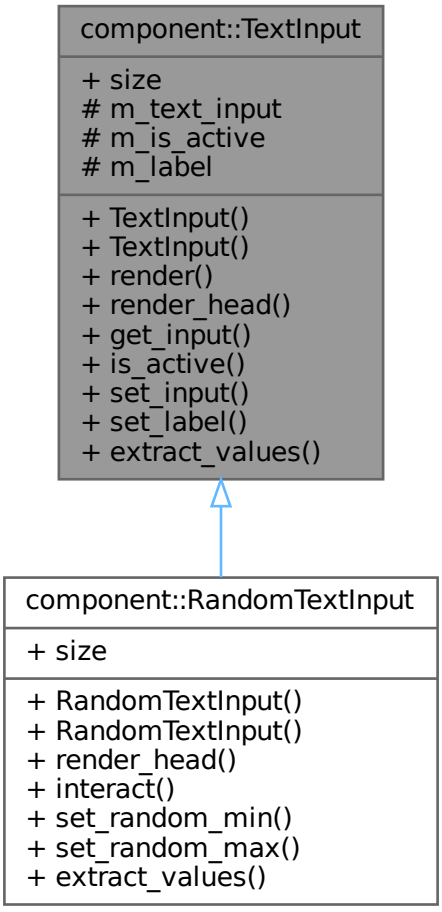
The documentation for this class was generated from the following files:

- src/scene/stack_scene.hpp
- src/scene/stack_scene.cpp

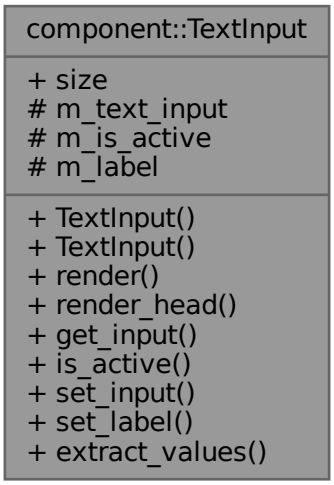
6.34 component::TextInput Class Reference

`#include <text_input.hpp>`

Inheritance diagram for component::TextInput:



Collaboration diagram for component::TextInput:



Public Member Functions

- TextInput ()=default
- TextInput (const char ∗label)
- void render (float x, float y)
- void render_head (float &options_head, float head_offset)
- std::string get_input () const
- bool is_active () const
- void set_input (const char ∗input, int len)
- void set_label (const char ∗const label)
- core::Deque< int > extract_values ()

Static Public Attributes

- static constexpr Vector2 size {200, 50}

Protected Attributes

- char m_text_input [constants::text_buffer_size] = ""
- bool m_is_active {}
- const char ∗ m_label {}

6.34.1 Detailed Description

Definition at line 12 of file text_input.hpp.

6.34.2 Constructor & Destructor Documentation

6.34.2.1 TextInput() [1/2]

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

6.34.2.2 TextInput() [2/2]

```
component::TextInput::TextInput (
            const char * label )
```

Definition at line 14 of file text_input.cpp.

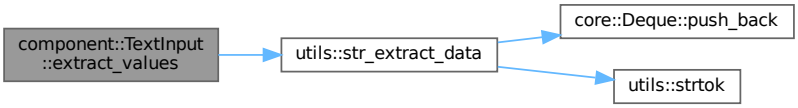
6.34.3 Member Function Documentation

6.34.3.1 extract_values()

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

Definition at line 48 of file text_input.cpp.

Here is the call graph for this function:



6.34.3.2 get_input()

```
std::string component::TextInput::get_input ( ) const
```

Definition at line 38 of file text_input.cpp.

6.34.3.3 is_active()

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

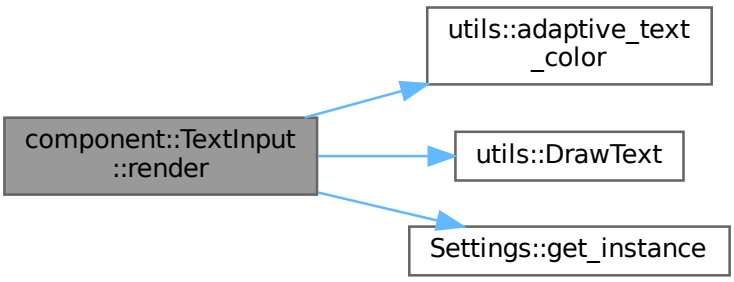
Definition at line 40 of file text_input.cpp.

6.34.3.4 render()

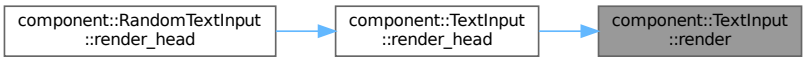
```
void component::TextInput::render (
            float x,
            float y )
```

Definition at line 16 of file text_input.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

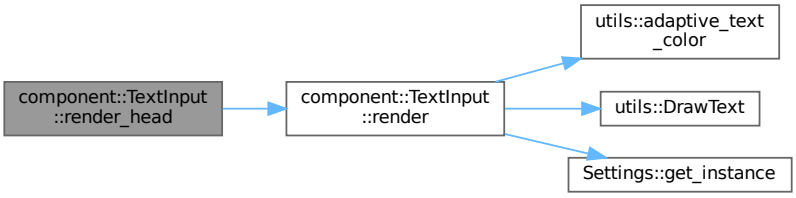


6.34.3.5 render_head()

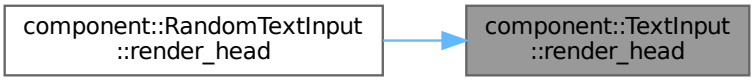
```
void component::TextInput::render_head (
            float & options_head,
            float head_offset )
```

Definition at line 33 of file text_input.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.34.3.6 set_input()

```
void component::TextInput::set_input (
            const char * input,
            int len )
```

Definition at line 44 of file text_input.cpp.

6.34.3.7 set_label()

```
void component::TextInput::set_label (
            const char *const label )
```

Definition at line 42 of file text_input.cpp.

6.34.4 Member Data Documentation

6.34.4.1 m_is_active

```
bool component::TextInput::m_is_active {}  [protected]
```

Definition at line 15 of file text_input.hpp.

6.34.4.2 m_label

```
const char* component::TextInput::m_label {}  [protected]
```

Definition at line 16 of file text_input.hpp.

6.34.4.3 m_text_input

```
char component::TextInput::m_text_input[constants::text_buffer_size] = ""  [protected]
```

Definition at line 14 of file text_input.hpp.

6.34.4.4 size

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

Definition at line 19 of file text_input.hpp.

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

- src/component/text_input.hpp
- src/component/text_input.cpp

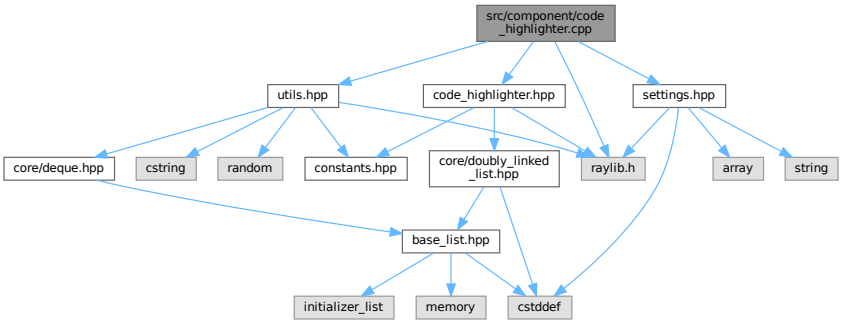
Chapter 7

File Documentation

7.1 src/component/code_highlighter.cpp File Reference

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

Include dependency graph for code_highlighter.cpp:



Namespaces

- namespace component

7.2 code_highlighter.cpp

Go to the documentation of this file.

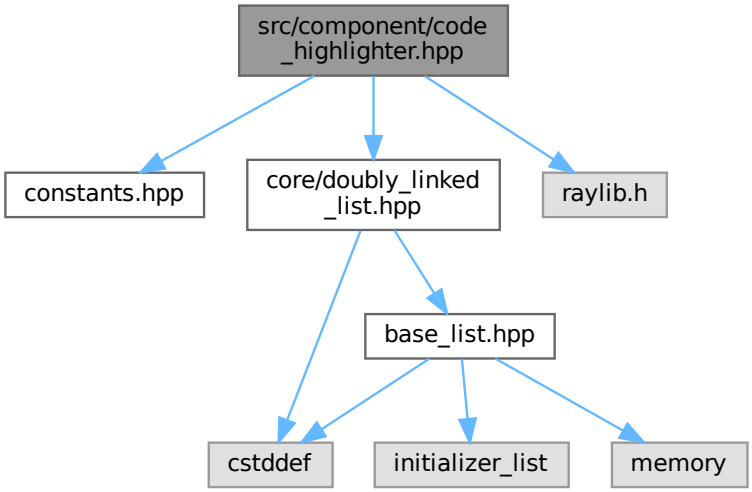
```
00001 #include "code_highlighter.hpp"
00002
00003 #include "raylib.h"
00004 #include "settings.hpp"
00005 #include "utils.hpp"
00006
00007 namespace component {
00008
```

```
00009 void CodeHighlighter::render() {
00010     for (int i = 0; i < m_src_code.size(); ++i) {
00011         const Settings& settings = Settings::get_instance();
00012
00013         int color_index = (i == m_highlighted_line) ? 4 : 0;
00014         Color bg_color = settings.get_color(color_index);
00015         Color text_color = utils::adaptive_text_color(bg_color);
00016
00017         Rectangle shape{head_pos.x, head_pos.y + i * height, width, height};
00018         Vector2 text_head = {head_pos.x + 10, head_pos.y + i * height + 5};
00019
00020         DrawRectangleRec(shape, bg_color);
00021         utils::DrawText(m_src_code.at(i), text_head, text_color, 20, 2);
00022     }
00023 }
00024
00025 void CodeHighlighter::set_code(core::DoublyLinkedList<const char*>&& src_code) {
00026     clear();
00027     m_src_code = src_code;
00028 }
00029
00030 void CodeHighlighter::push_into_sequence(int line_number) {
00031     m_sequence.insert(m_sequence.size(), line_number);
00032 }
00033
00034 void CodeHighlighter::highlight(int frame_idx) {
00035     m_highlighted_line = m_sequence.at(frame_idx);
00036 }
00037
00038 void CodeHighlighter::clear() {
00039     m_src_code.clear();
00040     m_sequence.clear();
00041 }
00042
00043 }  // namespace component
```

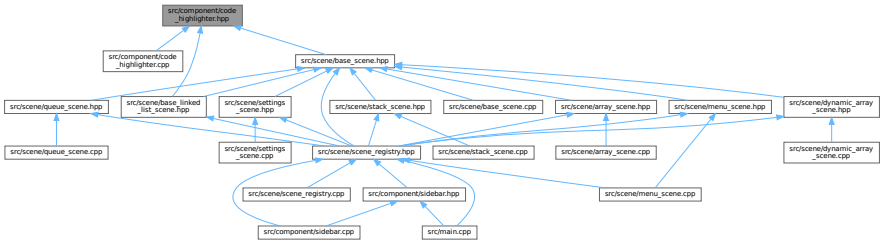
7.3 src/component/code_highlighter.hpp File Reference

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

Include dependency graph for code_highlighter.hpp:



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



Classes

- class component::CodeHighlighter

Namespaces

- namespace component

7.4 code_highlighter.hpp

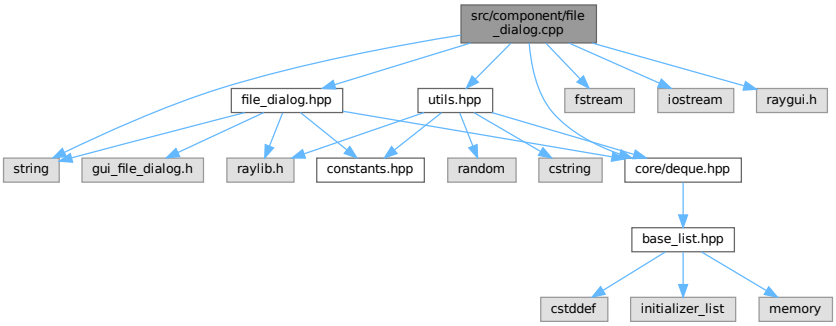
Go to the documentation of this file.

```
00001 #ifndef COMPONENT_CODE_HIGHLIGHTER_HPP_
00002 #define COMPONENT_CODE_HIGHLIGHTER_HPP_
00003
00004 #include "constants.hpp"
00005 #include "core/doubly_linked_list.hpp"
00006 #include "raylib.h"
00007
00008 namespace component {
00009
00010 class CodeHighlighter {
00011 private:
00012     static constexpr int width = 400;
00013     static constexpr int height = 30;
00014     static constexpr Vector2 head_pos{constants::scene_width - width,
00015                                       2.5F * height};
00016
00017     core::DoublyLinkedList<const char*> m_src_code;
00018     core::DoublyLinkedList<int> m_sequence;
00019     int m_highlighted_line{-1};
00020
00021 public:
00022     void render();
00023     void set_code(core::DoublyLinkedList<const char*>&& src_code);
00024     void push_into_sequence(int line_number);
00025     void highlight(int frame_idx);
00026     void clear();
00027 };
00028
00029 }  // namespace component
00030
00031 #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.

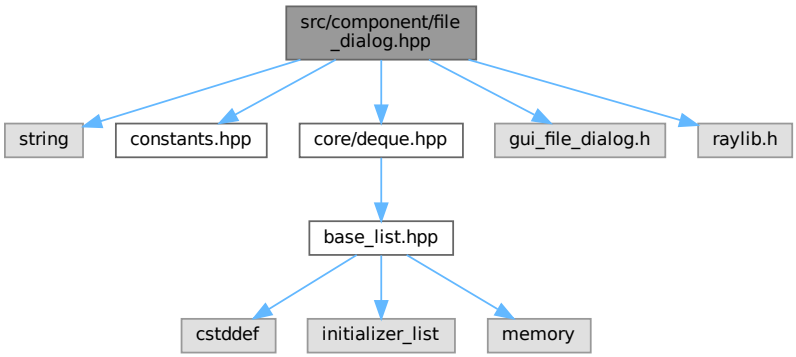
```
00001 #include "file_dialog.hpp"
00002
00003 #include <fstream>
00004 #include <iostream>
00005 #include <string>
00006
00007 #include "core/deque.hpp"
00008 #include "raygui.h"
00009 #include "utils.hpp"
00010
00011 namespace component {
00012
00013 FileDialog::FileDialog(int mode, const char* title, const char* message)
00014     : m_mode{mode}, m_title{title}, m_message{message} {}
00015
00016 FileDialog::FileDialog() : FileDialog(0, "Open file...", "Open file") {}
00017
00018 int FileDialog::render(float x, float y) {
00019     m_file_dialog_state.title = m_title;
00020     m_file_dialog_state.fileName = m_file_input;
00021     m_file_dialog_state.message = m_message;
00022     m_file_dialog_state.dialogType = m_mode;
00023
00024     int result = -1;
00025     if (m_file_dialog_state.windowActive) {
00026         GuiLock();
00027         result = GuiFileDialog(&m_file_dialog_state);
00028         if (result >= 0) {
00029             m_file_dialog_state.windowActive = false;
00030         }
00031     }
```

```
00032
00033     const Rectangle shape{x, y, size.x, size.y};
00034
00035     if (GuiButton(shape, GuiIconText(ICON_FILE_OPEN, "Select file"))) {
00036         m_file_dialog_state.windowActive = true;
00037     }
00038
00039     GuiUnlock();
00040     return result;
00041 }
00042
00043 int FileDialog::render_head(float& options_head, float head_offset) {
00044     int ret = render(options_head, constants::scene_height - size.y);
00045     options_head += (size.x + head_offset);
00046     return ret;
00047 }
00048
00049 core::Deque<int> FileDialog::extract_values() {
00050     std::ifstream ifs(get_path());
00051     char buffer[constants::text_buffer_size]{};  // NOLINT
00052     ifs » buffer;
00053
00054     return utils::str_extract_data(buffer);  // NOLINT
00055 }
00056
00057 bool FileDialog::is_active() const { return m_file_dialog_state.windowActive; }
00058
00059 void FileDialog::set_mode_open() { m_mode = DIALOG_OPEN_FILE; }
00060
00061 void FileDialog::set_mode_save() { m_mode = DIALOG_SAVE_FILE; }
00062
00063 void FileDialog::set_message(const char* message) { m_message = message; }
00064
00065 void FileDialog::set_title(const char* title) { m_title = title; }
00066 std::string FileDialog::get_path() { return m_file_input; }
00067
00068 }  // namespace component
```

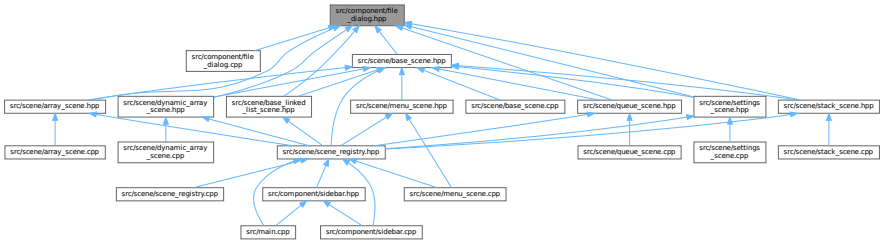
7.7 src/component/file_dialog.hpp File Reference

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

Include dependency graph for file_dialog.hpp:



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



Classes

- class component::FileDialog

Namespaces

- namespace component

7.8 file_dialog.hpp

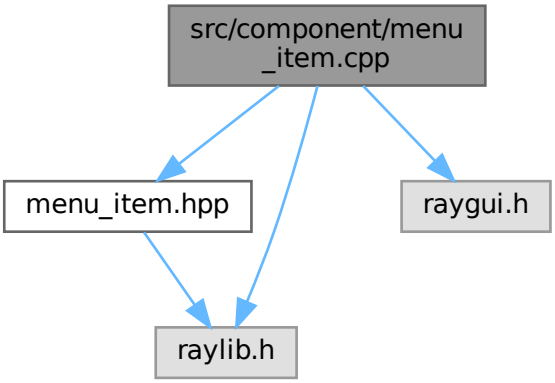
Go to the documentation of this file.

```
00001 #ifndef COMPONENT_FILE_DIALOG_HPP_
00002 #define COMPONENT_FILE_DIALOG_HPP_
00003
00004 #include <string>
00005
00006 #include "constants.hpp"
00007 #include "core/deque.hpp"
00008 #include "gui_file_dialog.h"
00009 #include "raylib.h"
00010
00011 namespace component {
00012
00013 class FileDialog {
00014 private:
00015     GuiFileDialogState m_file_dialog_state{
00016         InitGuiFileDialog(GetWorkingDirectory())};
00017
00018     char m_file_input[constants::text_buffer_size] = "";  // NOLINT
00019
00020     int m_mode{};
00021     const char* m_message;
00022     const char* m_title;
00023
00024 public:
00025     static constexpr Vector2 size{200, 50};
00026
00027     FileDialog();
00028     FileDialog(int mode, const char* title, const char* message);
00029
00030     int render_head(float& options_head, float head_offset);
00031     int render(float x, float y);
00032     core::Deque<int> extract_values();
00033     bool is_active() const;
00034     std::string get_path();
00035     void set_mode_open();
00036     void set_mode_save();
00037     void set_message(const char* message);
00038     void set_title(const char* title);
00039 };
00040
00041 }  // namespace component
00042
00043 #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

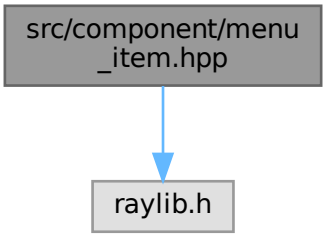
Go to the documentation of this file.

```
00001 #include "menu_item.hpp"
00002
00003 #include "raygui.h"
00004 #include "raylib.h"
00005
00006 namespace component {
00007
00008 MenuItem::MenuItem(int scene, const char* text, int x, int y,
00009                    const char* img_path)
00010     : m_scene{scene},
00011       m_text{text},
00012       m_x{x},
00013       m_y{y},
00014       m_texture{LoadTextureFromImage(LoadImage(img_path))} {}
00015
00016 int MenuItem::x() const { return m_x; }
00017 int MenuItem::y() const { return m_y; }
00018
00019 void MenuItem::render() {
00020     auto mouse = GetMousePosition();
00021     const Rectangle bound{(float)m_x, (float)m_y, block_width, block_height};
00022     const Rectangle text_bound{(float)m_x + 20,
00023                                (float)m_y + block_height - button_height,
00024                                button_width - 20, button_height};
00025
00026     DrawRectangleRec(bound, RAYWHITE);
00027     DrawTexture(m_texture, m_x, m_y, WHITE);
00028     GuiLabelButton(text_bound, m_text);
00029     DrawRectangleLinesEx(bound, 2, BLACK);
```

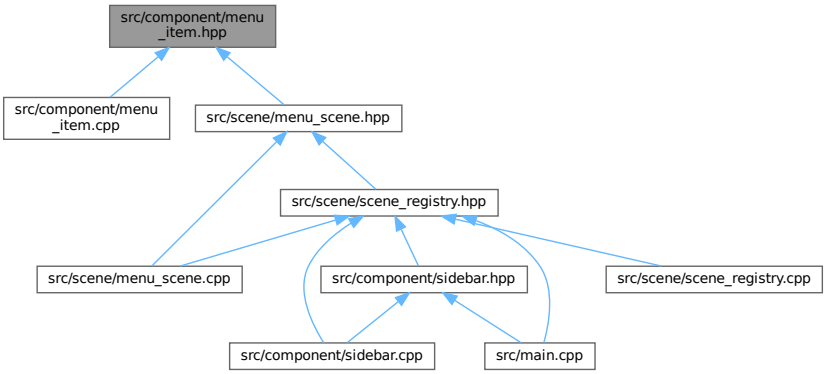
```
00030
00031     if (CheckCollisionPointRec(mouse, bound)) {
00032         DrawRectangle(m_x, m_y, block_width, block_height,
00033                       ColorAlpha(BLUE, 0.3));
00034         m_clicked = IsMouseButtonPressed(MOUSE_LEFT_BUTTON);
00035     }
00036 }
00037
00038 bool MenuItem::clicked() const { return m_clicked; }
00039
00040 void MenuItem::reset() { m_clicked = false; }
00041
00042 }  // namespace component
```

7.11 src/component/menu_item.hpp File Reference

`#include "raylib.h"`
Include dependency graph for menu_item.hpp:



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



Classes

- class component::MenuItem

Namespaces

- namespace component

7.12 menu_item.hpp

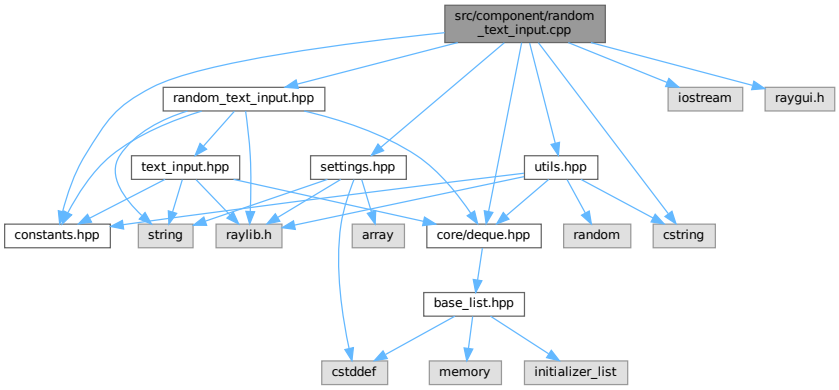
Go to the documentation of this file.

```
00001 #ifndef COMPONENT_MENU_ITEM_HPP_
00002 #define COMPONENT_MENU_ITEM_HPP_
00003
00004 #include "raylib.h"
00005
00006 namespace component {
00007
00008 class MenuItem {
00009 private:
00010     int m_scene{};
00011     int m_x{};
00012     int m_y{};
00013     Texture2D m_texture{};
00014     const char* m_text{};
00015
00016     bool m_clicked{};
00017
00018 public:
00019     static constexpr int block_width = 300;
00020     static constexpr int block_height = 200;
00021     static constexpr int button_width = block_width;
00022     static constexpr int button_height = 50;
00023
00024     MenuItem() = default;
00025     MenuItem(int scene, const char* text, int x, int y, const char* img_path);
00026
00027     int x() const;
00028     int y() const;
00029
00030     void render();
00031     bool clicked() const;
00032     void reset();
00033 };
00034
00035 }  // namespace component
00036
00037 #endif  // COMPONENT_MENU_ITEM_HPP_
```

7.13 src/component/random_text_input.cpp File Reference

```
#include "random_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 random_text_input.cpp:



Namespaces

- namespace component

7.14 random_text_input.cpp

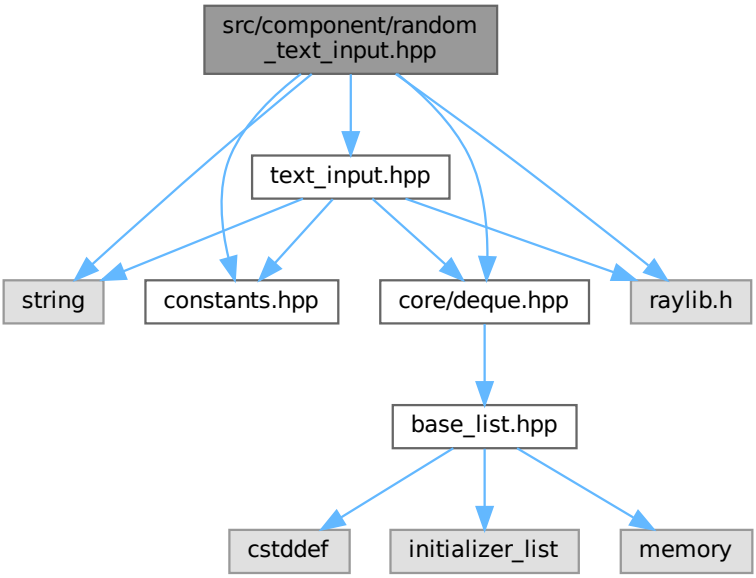
Go to the documentation of this file.

```
00001 #include "random_text_input.hpp"
00002
00003 #include <cstring>
00004 #include <iostream>
00005
00006 #include "constants.hpp"
00007 #include "core/deque.hpp"
00008 #include "raygui.h"
00009 #include "settings.hpp"
00010 #include "utils.hpp"
00011
00012 namespace component {
00013
00014 RandomTextInput::RandomTextInput(const char* label) : TextInput{label} {}
00015
00016 void RandomTextInput::set_random_min(int value) { m_random_min = value; }
00017
00018 void RandomTextInput::set_random_max(int value) { m_random_max = value; }
00019
00020 void RandomTextInput::render_head(float& options_head, float head_offset) {
00021     TextInput::render_head(options_head, 0);
00022
00023     Rectangle shape = {options_head, constants::scene_height - size.y, size.y,
00024                        size.y};
00025     m_set_random = GuiButton(shape, "#78#");
00026
00027     options_head += (shape.width + head_offset);
00028 }
00029
00030 bool RandomTextInput::interact() {
00031     if (m_set_random) {
00032         auto value = utils::get_random(m_random_min, m_random_max);
00033         m_set_random = false;
00034         std::strncpy(m_text_input, std::to_string(value).c_str(),
00035                      constants::text_buffer_size);
00036         return true;
00037     }
00038
00039     return false;
00040 }
00041
00042 }  // namespace component
```

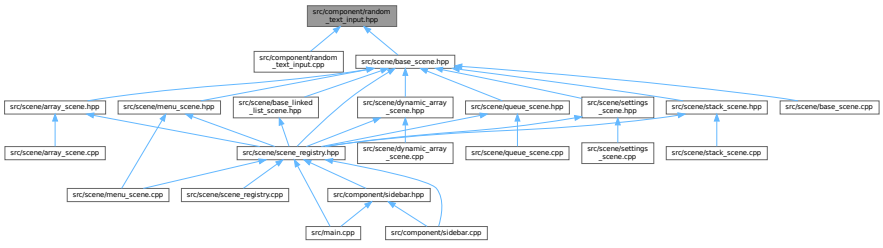
7.15 src/component/random_text_input.hpp File Reference

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

Include dependency graph for random_text_input.hpp:



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



Classes

- class component::RandomTextInput

Namespaces

- namespace component

7.16 random_text_input.hpp

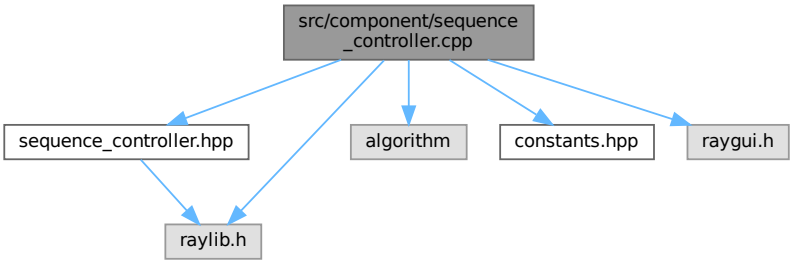
Go to the documentation of this file.

```
00001 #ifndef COMPONENT_RANDOM_TEXT_INPUT_HPP_
00002 #define COMPONENT_RANDOM_TEXT_INPUT_HPP_
00003
00004 #include <string>
00005
00006 #include "constants.hpp"
00007 #include "core/deque.hpp"
00008 #include "raylib.h"
00009 #include "text_input.hpp"
00010
00011 namespace component {
00012
00013 class RandomTextInput : public TextInput {
00014 private:
00015     int m_random_min{constants::min_val};
00016     int m_random_max{constants::max_val};
00017     bool m_set_random{};
00018
00019 public:
00020     using TextInput::size;
00021
00022     RandomTextInput() = default;
00023     RandomTextInput(const char* label);
00024
00025     using TextInput::extract_values;
00026
00027     void render_head(float& options_head, float head_offset);
00028     bool interact();
00029     void set_random_min(int value);
00030     void set_random_max(int value);
00031 };
00032
00033 }  // namespace component
00034
00035 #endif  // COMPONENT_RANDOM_TEXT_INPUT_HPP_
```

7.17 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

7.18 sequence_controller.cpp

Go to the documentation of this file.

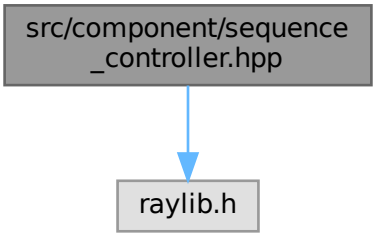
```
00001 #include "sequence_controller.hpp"
00002
00003 #include <algorithm>
00004
00005 #include "constants.hpp"
00006 #include "raygui.h"
00007 #include "raylib.h"
00008
00009 namespace component {
00010
00011 void SequenceController::set_max_value(int num) { m_num_steps = num; }
00012
00013 void SequenceController::set_progress_value(int value) {
00014     m_progress_value = value;
00015 }
00016
00017 void SequenceController::set_run_all(bool run_all) { m_run_all = run_all; }
00018
00019 bool SequenceController::get_run_all() const { return m_run_all; }
00020
00021 int SequenceController::get_progress_value() const { return m_progress_value; }
00022
00023 float SequenceController::get_speed_scale() const {
00024     return (float)m_speed / speed_scale;
00025 }
00026
00027 void SequenceController::reset_anim_counter() { m_anim_counter = 0; }
00028
00029 void SequenceController::inc_anim_counter() {
00030     if (get_run_all()) {
00031         ++m_anim_counter;
00032     }
00033 }
00034
00035 int SequenceController::get_anim_counter() const { return m_anim_counter; }
00036
00037 void SequenceController::set_rerun() {
00038     reset_anim_counter();
00039     set_run_all(true);
00040 }
00041
00042 int SequenceController::get_anim_frame() const {
00043     if (get_run_all()) {
00044         return 2.0F * get_anim_counter() * get_speed_scale() /
00045                constants::frames_per_second;
00046     } else {
00047         return get_progress_value();
00048     }
00049 }
00050
00051 void SequenceController::render() {
00052     Rectangle replay_shape{button_size.x * 0.5F,
00053                            constants::scene_height - 1.5F * button_size.x,
00054                            button_size.x, button_size.y};
00055
00056     Rectangle prev_frame_shape{
00057         replay_shape.x + replay_shape.width + button_size.x * 0.5F,
00058         replay_shape.y, button_size.x, button_size.y};
00059
00060     Rectangle progress_shape{prev_frame_shape.x + button_size.x * 1.5F,
00061                              replay_shape.y, 360, button_size.y};
00062
00063     Rectangle next_frame_shape{
00064         progress_shape.x + progress_shape.width + button_size.x * 0.5F,
00065         replay_shape.y, button_size.x, button_size.y};
00066
00067     Rectangle prev_speed_shape{prev_frame_shape.x + 240,
00068                                prev_frame_shape.y - 1.5F * button_size.y,
00069                                button_size.x, button_size.y};
00070
00071     Rectangle next_speed_shape{next_frame_shape.x,
00072                                next_frame_shape.y - 1.5F * button_size.y,
00073                                button_size.x, button_size.y};
00074
00075     Rectangle speed_shape{prev_speed_shape.x + 1.5F * button_size.x,
00076                           prev_speed_shape.y, 120, button_size.y};
00077
00078     m_prev_speed = GuiButton(prev_speed_shape, "#114#");
00079     m_next_speed = GuiButton(next_speed_shape, "#115#");
00080     GuiStatusBar(speed_shape, TextFormat("Speed: %.2fx", get_speed_scale()));
00081
00082     m_replay = GuiButton(replay_shape, "#75#");
```

```
00083     m_prev_frame = GuiButton(prev_frame_shape, "#72#");
00084     m_progress_value =
00085         (int)GuiProgressBar(progress_shape, nullptr, nullptr,
00086                             (float)m_progress_value, 0, (float)m_num_steps);
00087     m_next_frame = GuiButton(next_frame_shape, "#73#");
00088 }
00089
00090 bool SequenceController::interact() {
00091     if (m_replay) {
00092         set_progress_value(0);
00093         set_run_all(true);
00094         return true;
00095     }
00096
00097     if (m_prev_frame) {
00098         set_progress_value(std::max(get_progress_value() - 1, 0));
00099         return true;
00100     }
00101
00102     if (m_next_frame) {
00103         set_progress_value(std::min(get_progress_value() + 1, m_num_steps));
00104         return true;
00105     }
00106
00107     if (m_prev_speed) {
00108         m_speed = std::max(m_speed - 1, 2);
00109         return true;
00110     }
00111
00112     if (m_next_speed) {
00113         m_speed = std::min(m_speed + 1, 6);
00114         return true;
00115     }
00116
00117     return false;
00118 }
00119
00120 }  // namespace component
```

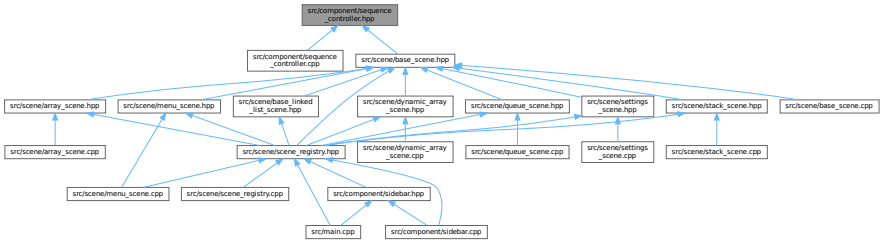
7.19 src/component/sequence_controller.hpp File Reference

`#include "raylib.h"`

Include dependency graph for sequence_controller.hpp:



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



Classes

- class component::SequenceController

Namespaces

- namespace component

7.20 sequence_controller.hpp

Go to the documentation of this file.

```
00001 #ifndef COMPONENT_SEQUENCE_CONTROLLER_HPP_
00002 #define COMPONENT_SEQUENCE_CONTROLLER_HPP_
00003
00004 #include "raylib.h"
00005
00006 namespace component {
00007
00008 class SequenceController {
00009 private:
00010     static constexpr Vector2 button_size{25, 25};
00011     static constexpr int speed_scale = 4;
00012
00013     bool m_replay{};
00014     bool m_prev_frame{};
00015     bool m_next_frame{};
00016     int m_progress_value{};
00017     int m_num_steps{};
00018     bool m_run_all{};
00019     int m_anim_counter{};
00020
00021     bool m_prev_speed{};
00022     bool m_next_speed{};
00023     int m_speed{speed_scale};
00024
00025 public:
00026     void render();
00027     bool interact();
00028
00029     void set_max_value(int num);
00030     void set_progress_value(int value);
00031     void set_run_all(bool run_all);
00032     void set_rerun();
00033
00034     bool get_run_all() const;
00035     int get_progress_value() const;
00036     float get_speed_scale() const;
00037
00038     void reset_anim_counter();
00039     void inc_anim_counter();
00040     int get_anim_counter() const;
00041     int get_anim_frame() const;
00042 };
00043
00044 }  // namespace component
00045
00046 #endif  // COMPONENT_SEQUENCE_CONTROLLER_HPP_
```

7.21 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

7.22 sidebar.cpp

Go to the documentation of this file.

```
00001 #include "sidebar.hpp"
00002
00003 #include "constants.hpp"
00004 #include "raygui.h"
00005 #include "raylib.h"
00006 #include "scene/scene_registry.hpp"
00007 #include "utils.hpp"
00008
00009 namespace component {
00010
00011 void SideBar::render() {
00012     (m_edit_mode) ? GuiLock() : GuiUnlock();
00013
00014     scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();
00015     int options_head = 2 * constants::sidebar_width;
00016
00017     constexpr float scale = 0.2;
00018
00019     constexpr Rectangle menu_button_shape{20, 20, button_height * 2,
00020                                           button_height};
00021     constexpr Rectangle selection_shape{
00022         menu_button_shape.x + menu_button_shape.width + 10, menu_button_shape.y,
00023         button_width, button_height};
00024     constexpr Rectangle settings_button_shape{
00025         constants::scene_width - button_height - 20, 20, button_height,
00026         button_height};
00027
00028     m_next_scene = registry.get_scene();
00029
00030     bool menu_is_next = m_next_scene == scene::Menu;
00031     bool settings_is_next = m_next_scene == scene::Settings;
00032
00033     if (!menu_is_next) {
00034         m_return_menu = GuiButton(menu_button_shape, "#118#Menu");
00035     }
00036
00037     if (!menu_is_next && !settings_is_next) {
00038         if (GuiDropdownBox(selection_shape, sidebar_labels, &m_next_scene,
00039                            m_edit_mode)) {
00040             m_pressed = true;
00041             m_edit_mode ^= 1;
00042         }
00043     }
00044
00045     m_return_settings = GuiButton(settings_button_shape, "#142#");
```

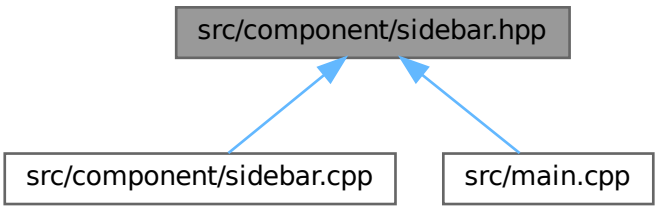
```
00046 }
00047
00048 void SideBar::interact() {
00049     scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();
00050     bool menu_is_current = registry.get_scene() == scene::Menu;
00051     bool settings_is_current = registry.get_scene() == scene::Settings;
00052
00053     if (!menu_is_current) {
00054         if (m_return_menu) {
00055             registry.set_scene(scene::Menu);
00056             m_return_menu = false;
00057             return;
00058         }
00059     }
00060
00061     if (!menu_is_current && !settings_is_current) {
00062         if (m_pressed) {
00063             registry.set_scene(m_next_scene);
00064             m_pressed = false;
00065             return;
00066         }
00067     }
00068
00069     if (m_return_settings) {
00070         if (settings_is_current) {
00071             registry.set_scene(m_scene_before_settings);
00072         } else {
00073             m_scene_before_settings = registry.get_scene();
00074             registry.set_scene(scene::Settings);
00075         }
00076         m_return_settings = false;
00077         return;
00078     }
00079 }
00080
00081 }  // namespace component
```

7.23 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

7.24 sidebar.hpp

Go to the documentation of this file.

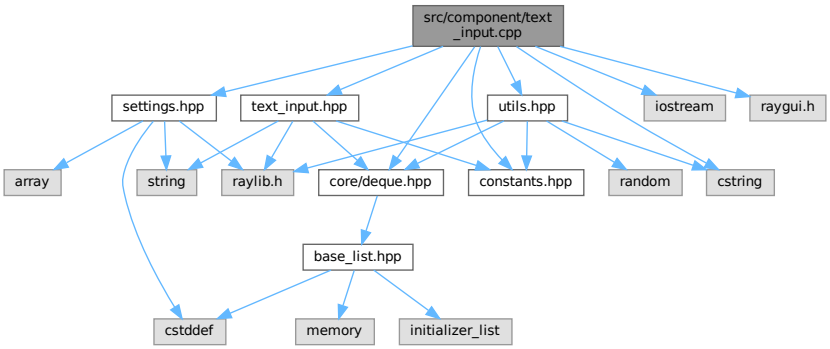
```
00001 #ifndef COMPONENT_SIDEBAR_HPP_
00002 #define COMPONENT_SIDEBAR_HPP_
00003
00004 #include <array>
00005
00006 #include "constants.hpp"
00007 #include "scene/scene_registry.hpp"
00008
00009 namespace component {
00010
00011 class SideBar {
00012 private:
00013     static constexpr int num_scenes = 8;
00014
00015     static constexpr int button_width = constants::sidebar_width;
00016     static constexpr int button_height = 50;
00017
00018     static constexpr const char* sidebar_labels =
00019         "Array;"
00020         "Dynamic Array;"
00021         "Linked List;"
00022         "Doubly Linked List;"
00023         "Circular Linked List;"
00024         "Stack;"
00025         "Queue";
00026
00027     int m_next_scene{};
00028     bool m_edit_mode{};
00029     bool m_return_menu{};
00030     bool m_return_settings{};
00031     int m_scene_before_settings{};
00032     bool m_pressed{};
00033
00034 public:
00035     void render();
00036     void interact();
00037 };
00038
00039 }  // namespace component
00040
00041 #endif  // COMPONENT_SIDEBAR_HPP_
```

7.25 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

7.26 text_input.cpp

Go to the documentation of this file.

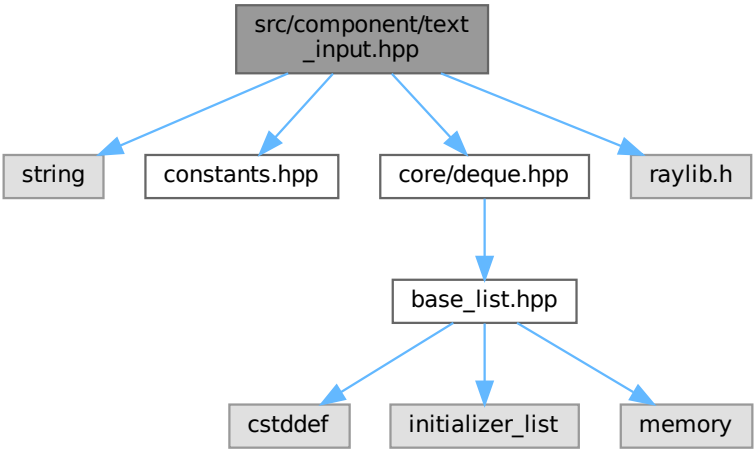
```
00001 #include "text_input.hpp"
00002
00003 #include <cstring>
00004 #include <iostream>
00005
00006 #include "constants.hpp"
00007 #include "core/deque.hpp"
00008 #include "raygui.h"
00009 #include "settings.hpp"
00010 #include "utils.hpp"
00011
00012 namespace component {
00013
00014 TextInput::TextInput(const char* label) : m_label{label} {}
00015
00016 void TextInput::render(float x, float y) {
00017     Rectangle shape{x, y, size.x, size.y};
00018
00019     utils::DrawText(
00020         m_label, {x, y - 25},
00021         utils::adaptive_text_color(
00022             Settings::get_instance().get_color(Settings::num_color - 1)),
00023         20, 2);
00024
00025     DrawRectangleRec(shape, RAYWHITE);
00026
00027     if (GuiTextBox(shape, static_cast<char*>(m_text_input),
00028                    constants::text_buffer_size, m_is_active)) {
00029         m_is_active ^= 1;
00030     }
00031 }
00032
00033 void TextInput::render_head(float& options_head, float head_offset) {
00034     render(options_head, constants::scene_height - size.y);
00035     options_head += (size.x + head_offset);
00036 }
00037
00038 std::string TextInput::get_input() const { return {m_text_input}; }
00039
00040 bool TextInput::is_active() const { return m_is_active; }
00041
00042 void TextInput::set_label(const char* const label) { m_label = label; }
```

```
00043
00044 void TextInput::set_input(const char* input, int len) {
00045     std::strncpy(static_cast<char*>(m_text_input), input, len);
00046 }
00047
00048 core::Deque<int> TextInput::extract_values() {
00049     core::Deque<int> nums = utils::str_extract_data(m_text_input);  // NOLINT
00050     return nums;
00051 }
00052
00053 }  // namespace component
```

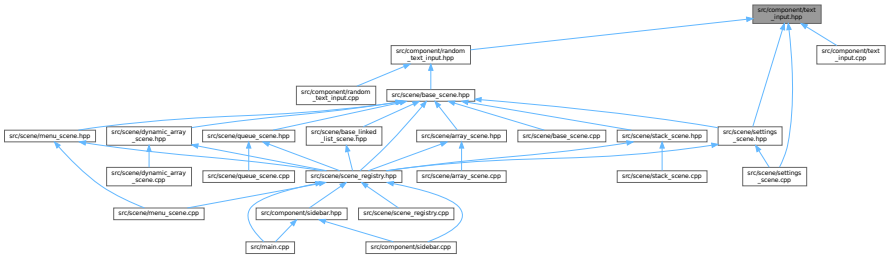
7.27 src/component/text_input.hpp File Reference

```
#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

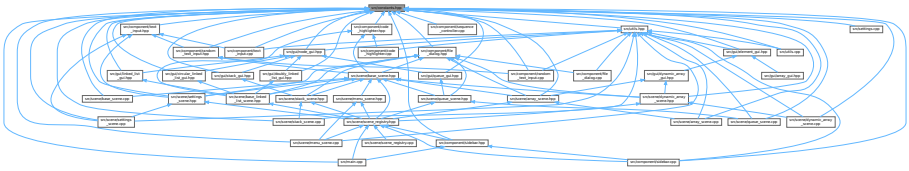
7.28 text_input.hpp

Go to the documentation of this file.

```
00001 #ifndef COMPONENT_TEXT_INPUT_HPP_
00002 #define COMPONENT_TEXT_INPUT_HPP_
00003
00004 #include <string>
00005
00006 #include "constants.hpp"
00007 #include "core/deque.hpp"
00008 #include "raylib.h"
00009
00010 namespace component {
00011
00012 class TextInput {
00013 protected:
00014     char m_text_input[constants::text_buffer_size] = "";  // NOLINT
00015     bool m_is_active{};
00016     const char* m_label{};
00017
00018 public:
00019     static constexpr Vector2 size{200, 50};
00020
00021     TextInput() = default;
00022     TextInput(const char* label);
00023
00024     void render(float x, float y);
00025     void render_head(float& options_head, float head_offset);
00026     std::string get_input() const;
00027     bool is_active() const;
00028     void set_input(const char* input, int len);
00029     void set_label(const char* const label);
00030     core::Deque<int> extract_values();
00031 };
00032
00033 }  // namespace component
00034
00035 #endif  // COMPONENT_TEXT_INPUT_HPP_
```

7.29 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"

7.30 constants.hpp

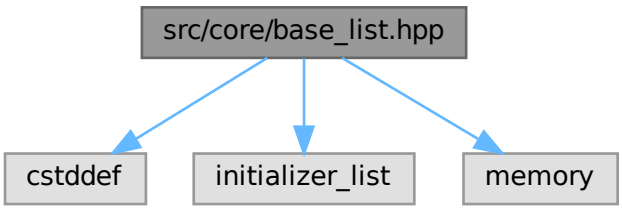
Go to the documentation of this file.

```
00001 #ifndef CONSTANTS_HPP_
00002 #define CONSTANTS_HPP_
00003
00004 namespace constants {
00005
00006 constexpr int scene_width = 1366;
00007 constexpr int scene_height = 768;
00008 constexpr int frames_per_second = 30;
00009
00010 constexpr int sidebar_width = 256;
00011 constexpr int ani_speed = 8;
00012
00013 constexpr int text_buffer_size = 512;
00014
00015 constexpr int min_val = 0;
00016 constexpr int max_val = 999;
00017
00018 constexpr int default_font_size = 60;
00019
00020 constexpr const char* default_color_path = "data/color.bin";
00021
00022 }  // namespace constants
00023
00024 #endif  // CONSTANTS_HPP_
```

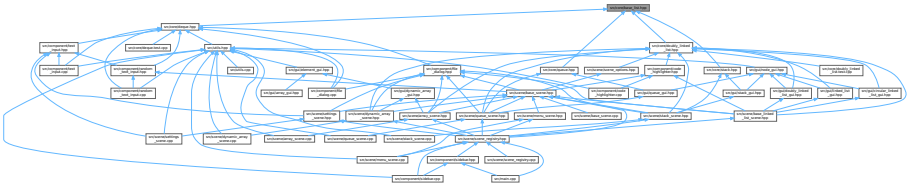
7.31 src/core/base_list.hpp File Reference

```
#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

7.32 base_list.hpp

Go to the documentation of this file.

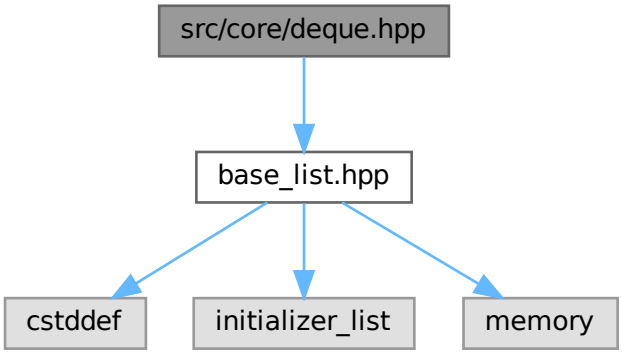
```
00001 #ifndef CORE_BASE_LIST_HPP_
00002 #define CORE_BASE_LIST_HPP_
00003
00004 #include <cstddef>
00005 #include <initializer_list>
00006 #include <memory>
00007
00008 namespace core {
00009
00010 template<typename T>
00011 class BaseList {
00012 protected:
00013     struct Node;
00014     using Node_ptr = Node*;
00015
00016     struct Node {
00017         T data{};
00018         Node_ptr prev{};
00019         Node_ptr next{};
00020     };
00021
00022     Node_ptr m_head{nullptr};
00023     Node_ptr m_tail{nullptr};
00024     std::size_t m_size{};
00025
00026     void init_first_element(const T& elem);
00027     void clean_up();
00028     void copy_data(const BaseList& rhs);
00029
00030     void push_back(const T& elem);
00031     void push_front(const T& elem);
00032
00033     T& back() const;
00034     T& front() const;
00035
00036     void pop_front();
00037     void pop_back();
00038
00039 public:
00040     BaseList() = default;
00041     BaseList(std::initializer_list<T> init_list);
00042     BaseList(const BaseList& rhs);
00043     BaseList& operator=(const BaseList& rhs);
00044     BaseList(BaseList&& rhs) noexcept;
00045     BaseList& operator=(BaseList&& rhs) noexcept;
00046     ~BaseList();
```

```
00047
00048     [[nodiscard]] bool empty() const;
00049     [[nodiscard]] std::size_t size() const;
00050 };
00051
00052 template<typename T>
00053 BaseList<T>::BaseList(const BaseList& rhs) {
00054     copy_data(rhs);
00055 }
00056
00057 template<typename T>
00058 BaseList<T>::BaseList(std::initializer_list<T> init_list) {
00059     for (const auto& elem : init_list) {
00060         push_back(elem);
00061     }
00062 }
00063
00064 template<typename T>
00065 BaseList<T>& BaseList<T>::operator=(const BaseList& rhs) {
00066     if (this != &rhs) {
00067         copy_data(rhs);
00068     }
00069
00070     return *this;
00071 }
00072
00073 template<typename T>
00074 BaseList<T>::BaseList(BaseList&& rhs) noexcept
00075     : m_head{rhs.m_head}, m_tail{rhs.m_tail}, m_size{rhs.m_size} {
00076     rhs.m_head = nullptr;
00077     rhs.m_tail = nullptr;
00078     rhs.m_size = 0;
00079 }
00080
00081 template<typename T>
00082 BaseList<T>& BaseList<T>::operator=(BaseList&& rhs) noexcept {
00083     if (this != &rhs) {
00084         clean_up();
00085
00086         m_head = rhs.m_head;
00087         m_tail = rhs.m_tail;
00088         m_size = rhs.m_size;
00089
00090         rhs.m_head = nullptr;
00091         rhs.m_tail = nullptr;
00092         rhs.m_size = 0;
00093     }
00094
00095     return *this;
00096 }
00097
00098 template<typename T>
00099 BaseList<T>::~BaseList() {
00100     clean_up();
00101 }
00102
00103 template<typename T>
00104 bool BaseList<T>::empty() const {
00105     return m_size == 0;
00106 }
00107
00108 template<typename T>
00109 std::size_t BaseList<T>::size() const {
00110     return m_size;
00111 }
00112
00113 template<typename T>
00114 void BaseList<T>::init_first_element(const T& elem) {
00115     m_head = new Node{elem, nullptr, nullptr};
00116     m_tail = m_head;
00117     m_size = 1;
00118 }
00119
00120 template<typename T>
00121 void BaseList<T>::clean_up() {
00122     Node_ptr ptr{nullptr};
00123
00124     while (m_head != nullptr) {
00125         ptr = m_head->next;
00126         delete m_head;
00127         m_head = ptr;
00128     }
00129
00130     m_tail = m_head;
00131     m_size = 0;
00132 }
00133
```

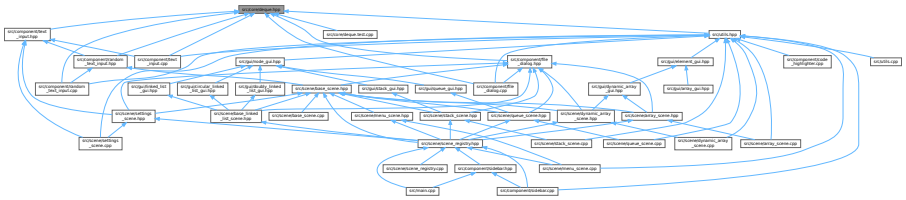
```
00134 template<typename T>
00135 void BaseList<T>::copy_data(const BaseList& rhs) {
00136     for (Node_ptr ptr = rhs.m_head; ptr != nullptr; ptr = ptr->next) {
00137         push_back(ptr->data);
00138     }
00139 }
00140
00141 template<typename T>
00142 void BaseList<T>::push_back(const T& elem) {
00143     if (empty()) {
00144         init_first_element(elem);
00145         return;
00146     }
00147
00148     m_tail->next = new Node{elem, m_tail, nullptr};
00149     m_tail = m_tail->next;
00150     ++m_size;
00151 }
00152
00153 template<typename T>
00154 void BaseList<T>::push_front(const T& elem) {
00155     if (empty()) {
00156         init_first_element(elem);
00157         return;
00158     }
00159
00160     m_head->prev = new Node{elem, nullptr, m_head};
00161     m_head = m_head->prev;
00162     ++m_size;
00163 }
00164
00165 template<typename T>
00166 T& BaseList<T>::back() const {
00167     return m_tail->data;
00168 }
00169
00170 template<typename T>
00171 T& BaseList<T>::front() const {
00172     return m_head->data;
00173 }
00174
00175 template<typename T>
00176 void BaseList<T>::pop_back() {
00177     if (size() <= 1) {
00178         clean_up();
00179         return;
00180     }
00181
00182     m_tail = m_tail->prev;
00183     delete m_tail->next;
00184     m_tail->next = nullptr;
00185     --m_size;
00186 }
00187
00188 template<typename T>
00189 void BaseList<T>::pop_front() {
00190     if (size() <= 1) {
00191         clean_up();
00192         return;
00193     }
00194
00195     m_head = m_head->next;
00196     delete m_head->prev;
00197     m_head->prev = nullptr;
00198     --m_size;
00199 }
00200
00201 }  // namespace core
00202
00203 #endif  // CORE_BASE_LIST_HPP_
```

7.33 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

7.34 deque.hpp

Go to the documentation of this file.

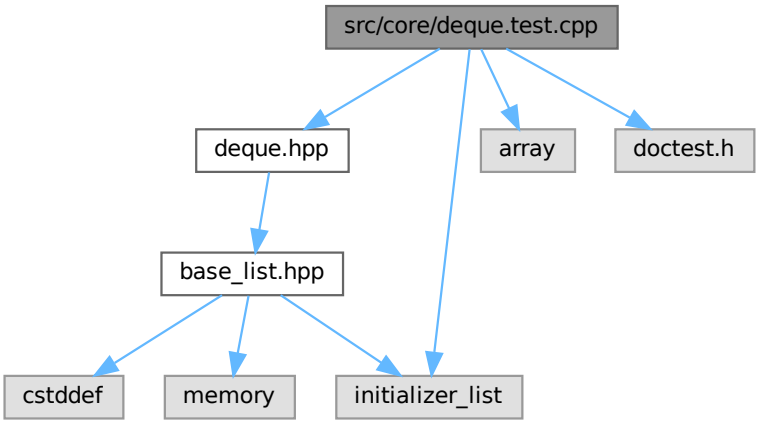
```
00001 #ifndef CORE_DEQUE_HPP_
00002 #define CORE_DEQUE_HPP_
00003
00004 #include "base_list.hpp"
00005
00006 namespace core {
00007
```

```
00008 template<typename T>
00009 class Deque : public BaseList<T> {
00010 private:
00011     using Base = BaseList<T>;
00012
00013 public:
00014     using Base::Base;
00015
00016     using Base::empty;
00017     using Base::size;
00018
00019     using Base::push_back;
00020     using Base::push_front;
00021
00022     using Base::back;
00023     using Base::front;
00024
00025     using Base::pop_back;
00026     using Base::pop_front;
00027 };
00028
00029 }  // namespace core
00030
00031 #endif  // CORE_DEQUE_HPP_
```

7.35 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}

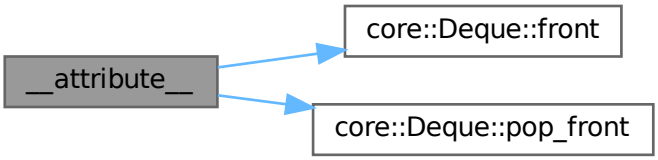
7.35.1 Function Documentation

7.35.1.1 __attribute__()

```
__attribute__ (
            (always_inline)  )  [inline]
```

Definition at line 38 of file deque.test.cpp.

Here is the call graph for this function:

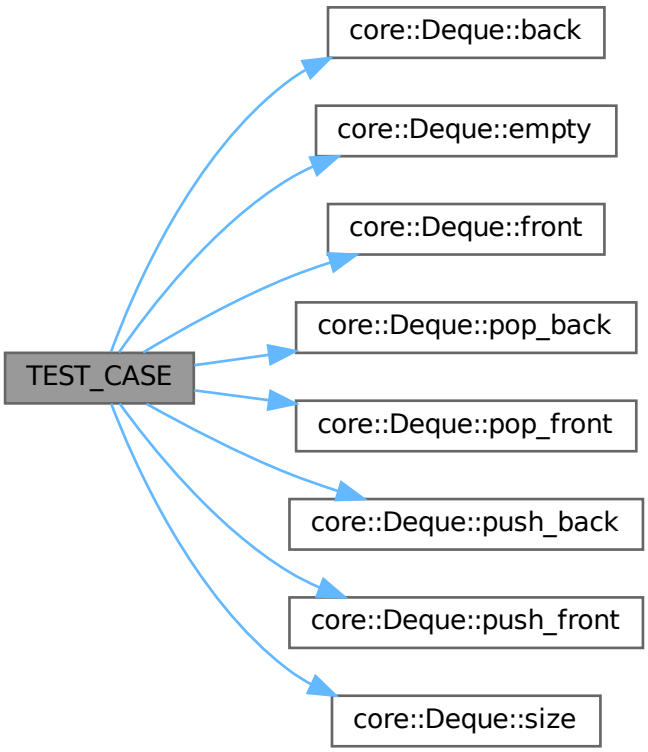


7.35.1.2 TEST_CASE() [1/2]

```
TEST_CASE (
            "core::Deque functionality"  )
```

Definition at line 8 of file deque.test.cpp.

Here is the call graph for this function:



7.35.1.3 TEST_CASE() [2/2]

```
TEST_CASE (
            "core::Deque special member functions"  )
```

Definition at line 45 of file deque.test.cpp.

7.35.2 Variable Documentation

7.35.2.1 list

```
constexpr std::array<int, 3> list {1, 2, 3}  [constexpr]
```

Definition at line 36 of file deque.test.cpp.

7.36 deque.test.cpp

Go to the documentation of this file.

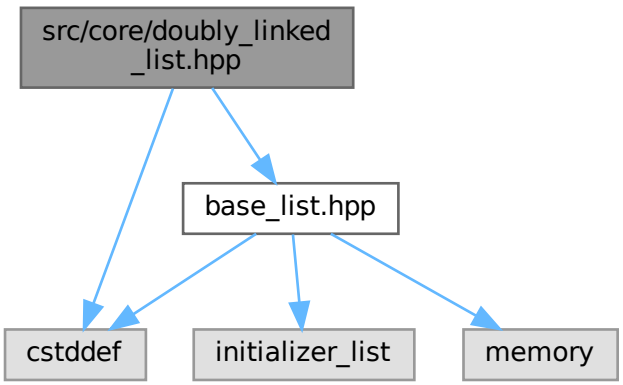
```
00001 #include "deque.hpp"
00002
00003 #include <array>
00004 #include <initializer_list>
00005
00006 #include "doctest.h"
00007
00008 TEST_CASE("core::Deque functionality") {
00009     core::Deque<int> deque;
00010     CHECK(deque.empty());
00011
00012     deque.push_back(2);
00013     deque.push_back(3);
00014     deque.push_front(1);
00015
00016     CHECK(deque.front() == 1);
00017     CHECK(deque.back() == 3);
00018     CHECK(deque.size() == 3);
00019
00020     deque.pop_back();
00021     CHECK(deque.back() == 2);
00022     CHECK(deque.size() == 2);
00023
00024     deque.pop_front();
00025     CHECK(deque.front() == 2);
00026     CHECK(deque.size() == 1);
00027
00028     deque.front() += 3;
00029     CHECK(deque.front() == 5);
00030
00031     deque.push_back(0);
00032     deque.back() -= 2;
00033     CHECK(deque.back() == -2);
00034 }
00035
00036 constexpr std::array<int, 3> list{1, 2, 3};
00037
00038 inline __attribute__((always_inline)) void check_match(core::Deque<int> deque) {
00039     for (int elem : list) {
00040         CHECK(deque.front() == elem);
00041         deque.pop_front();
00042     }
00043 }
00044
00045 TEST_CASE("core::Deque special member functions") {
00046     std::initializer_list<int> init_list{1, 2, 3};
00047
00048     SUBCASE("core::Deque(std::initializer_list<T>)") {
00049         core::Deque<int> deque{init_list};
00050         check_match(deque);
00051     }
00052
00053     SUBCASE("core::Deque(const core::Deque&)") {
00054         core::Deque<int> deque1{init_list};
00055         core::Deque<int> deque2{deque1};  // NOLINT
00056
00057         check_match(deque2);
00058         check_match(deque1);
00059     }
00060
00061     SUBCASE("core::Deque& operator=(const core::Deque&) (single)") {
00062         core::Deque<int> deque1{init_list};
00063         core::Deque<int> deque2 = deque1;  // NOLINT
00064
00065         check_match(deque2);
00066         check_match(deque1);
00067     }
00068
00069     SUBCASE("core::Deque& operator=(const core::Deque&) (multiple)") {
00070         core::Deque<int> deque1{init_list};
00071         core::Deque<int> deque2;
00072         core::Deque<int> deque3;
00073         deque3 = deque2 = deque1;
00074
00075         check_match(deque3);
00076         check_match(deque2);
00077         check_match(deque1);
00078     }
00079
00080     SUBCASE("core::Deque(core::Deque&& rhs)") {
00081         {
00082             core::Deque<int> deque1{core::Deque<int>{init_list}};
```

```
00083             check_match(deque1);
00084         }
00085         {
00086             core::Deque<int> deque1{init_list};
00087             core::Deque<int> deque2{std::move(deque1)};
00088             check_match(deque2);
00089             CHECK(deque1.empty());  // NOLINT
00090         }
00091     }
00092
00093     SUBCASE("core::Deque& operator=(core::Deque&& rhs)") {
00094         {
00095             core::Deque<int> deque1{1, 2, 3};
00096             core::Deque<int> deque2 = std::move(deque1);
00097
00098             check_match(deque2);
00099             CHECK(deque1.empty());  // NOLINT
00100         }
00101         {
00102             core::Deque<int> deque{init_list};
00103             deque = std::move(deque);
00104             check_match(deque);  // NOLINT
00105         }
00106     }
00107 }
```

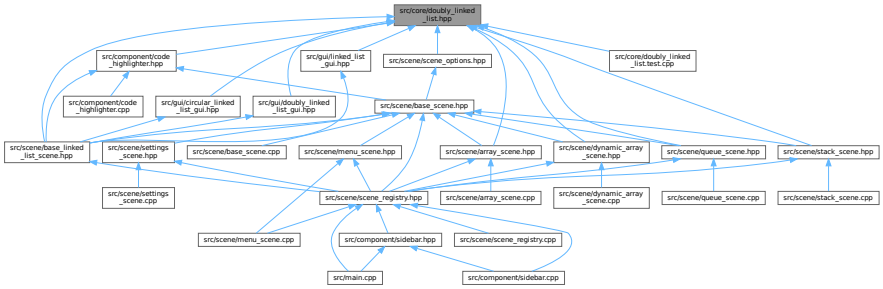
7.37 src/core/doubly_linked_list.hpp File Reference

```
#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:



Classes

- class core::DoublyLinkedList< T >

Namespaces

- namespace core

7.38 doubly_linked_list.hpp

Go to the documentation of this file.

```
00001 #ifndef CORE_DOUBLY_LINKED_LIST_HPP_
00002 #define CORE_DOUBLY_LINKED_LIST_HPP_
00003
00004 #include <cstddef>
00005
00006 #include "base_list.hpp"
00007
00008 namespace core {
00009
00010 template<typename T>
00011 class DoublyLinkedList : public BaseList<T> {
00012 protected:
00013     using Base = BaseList<T>;
00014     using Node = typename Base::Node;
00015     using Node_ptr = Node*;
00016     using cNode_ptr = const Node*;
00017
00018     using Base::m_head;
00019     using Base::m_size;
00020     using Base::m_tail;
00021
00022     Node_ptr internal_search(const T& elem);
00023     Node_ptr internal_find(std::size_t index);
00024
00025 public:
00026     using Base::Base;
00027
00028     using Base::empty;
00029     using Base::size;
00030
00031     Node_ptr search(const T& elem);
00032     Node_ptr find(std::size_t index);
00033
00034     cNode_ptr search(const T& elem) const;
00035     cNode_ptr find(std::size_t index) const;
00036
00037     Node_ptr insert(std::size_t index, const T& elem);
00038     Node_ptr remove(std::size_t index);
00039
00040     T& at(std::size_t index);
00041     T at(std::size_t index) const;
```

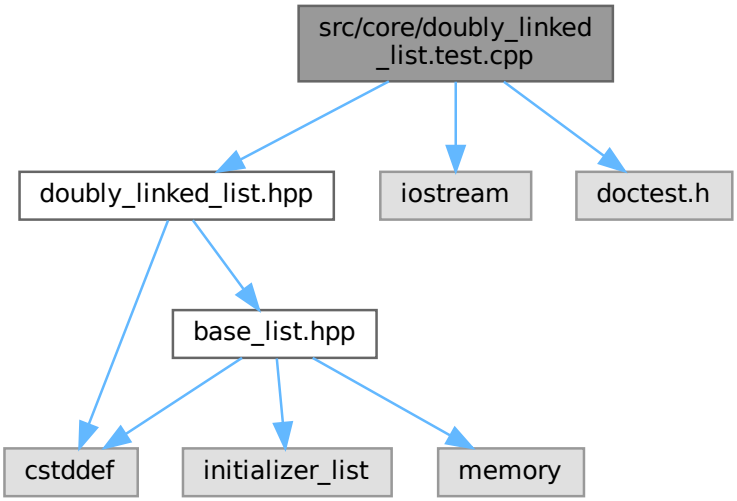
```
00042
00043     void clear();
00044 };
00045
00046 template<typename T>
00047 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::internal_search(
00048     const T& elem) {
00049     Node_ptr ptr{m_head};
00050
00051     while (ptr != nullptr) {
00052         if (ptr->data == elem) {
00053             break;
00054         }
00055
00056         ptr = ptr->next;
00057     }
00058
00059     return ptr;
00060 }
00061
00062 template<typename T>
00063 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::internal_find(
00064     std::size_t index) {
00065     Node_ptr ptr{m_head};
00066     std::size_t pos = 0;
00067
00068     while (ptr != nullptr && pos < index) {
00069         ptr = ptr->next;
00070         ++pos;
00071     }
00072
00073     return ptr;
00074 }
00075
00076 template<typename T>
00077 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::search(
00078     const T& elem) {
00079     return internal_search(elem);
00080 }
00081
00082 template<typename T>
00083 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::find(
00084     std::size_t index) {
00085     return internal_find(index);
00086 }
00087
00088 template<typename T>
00089 typename DoublyLinkedList<T>::cNode_ptr DoublyLinkedList<T>::search(
00090     const T& elem) const {
00091     return internal_search(elem);
00092 }
00093
00094 template<typename T>
00095 typename DoublyLinkedList<T>::cNode_ptr DoublyLinkedList<T>::find(
00096     std::size_t index) const {
00097     return internal_find(index);
00098 }
00099
00100 template<typename T>
00101 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::insert(
00102     std::size_t index, const T& elem) {
00103     if (index == 0) {
00104         Base::push_front(elem);
00105         return m_head;
00106     }
00107
00108     if (index >= m_size) {
00109         Base::push_back(elem);
00110         return m_tail;
00111     }
00112
00113     Node_ptr ptr = find(index);
00114     auto new_node = new Node{elem, ptr->prev, ptr};
00115
00116     ptr->prev->next = new_node;
00117     ptr->prev = new_node;
00118     ++m_size;
00119
00120     return new_node;
00121 }
00122
00123 template<typename T>
00124 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::remove(
00125     std::size_t index) {
00126     if (index >= m_size) {
00127         return nullptr;
00128     }
```

```
00129
00130     if (index == 0) {
00131         Base::pop_front();
00132         return m_head;
00133     }
00134
00135     if (index + 1 == m_size) {
00136         Base::pop_back();
00137         return nullptr;
00138     }
00139
00140     Node_ptr ptr = find(index);
00141     Node_ptr ret = ptr->next;
00142
00143     ptr->next->prev = ptr->prev;
00144     ptr->prev->next = ptr->next;
00145
00146     delete ptr;
00147     --m_size;
00148
00149     return ret;
00150 }
00151
00152 template<typename T>
00153 T& DoublyLinkedList<T>::at(std::size_t index) {
00154     return find(index)->data;
00155 }
00156
00157 template<typename T>
00158 T DoublyLinkedList<T>::at(std::size_t index) const {
00159     return find(index)->data;
00160 }
00161
00162 template<typename T>
00163 void DoublyLinkedList<T>::clear() {
00164     while (!empty()) {
00165         Base::pop_front();
00166     }
00167 }
00168
00169 }  // namespace core
00170
00171 #endif  // CORE_DOUBLY_LINKED_LIST_HPP_
```

7.39 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")

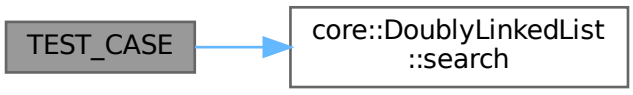
7.39.1 Function Documentation

7.39.1.1 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:



7.40 doubly_linked_list.test.cpp

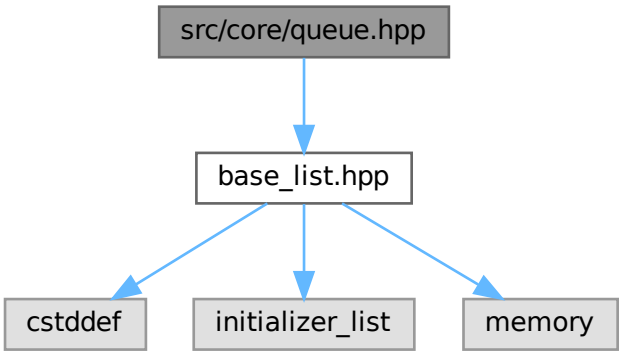
Go to the documentation of this file.

```
00001 #include "doubly_linked_list.hpp"
00002
00003 #include <iostream>
00004
00005 #include "doctest.h"
00006
00007 TEST_CASE("core::DoublyLinkedList functionality") {
00008     // List: {1, 2, 3}
00009     SUBCASE("Node_ptr search(const T& elem)") {
00010         core::DoublyLinkedList<int> dll{1, 2, 3};
00011         CHECK(dll.search(4) == nullptr);
00012         CHECK(dll.search(3)->data == 3);
00013     }
00014
00015     // List: {1, 2, 3}
00016     SUBCASE("Node_ptr find(std::size_t index)") {
00017         core::DoublyLinkedList<int> dll{1, 2, 3};
00018         CHECK(dll.find(8) == nullptr);
00019
00020         auto* ptr1 = dll.search(3);
00021         auto* ptr2 = dll.find(1);
00022
00023         CHECK(ptr1->data == 3);
00024         CHECK(ptr2->data == 2);
00025
00026         CHECK(ptr1->prev == ptr2);
00027         CHECK(ptr2->next == ptr1);
00028     }
00029
00030     SUBCASE("Node_ptr insert(std::size_t index, const T& elem)") {
00031         core::DoublyLinkedList<int> dll{1, 2, 3};
00032         auto* ptr0 = dll.search(1);
00033
00034         // List: {-1, 1, 2, 3}
00035         auto* ptr = dll.insert(0, -1);
00036
00037         CHECK(dll.size() == 4);
00038         CHECK(ptr->next == ptr0);
00039
00040         auto* ptrN = dll.search(3);
00041         // List: {-1, 1, 2, 3, 4}
00042         ptr = dll.insert(4, 4);
00043
00044         CHECK(dll.size() == 5);
00045         CHECK(ptr->prev == ptrN);
00046
00047         // List: {-1, 1, 20, 2, 3, 4}
00048         ptr = dll.insert(2, 20);  // NOLINT
00049         CHECK(ptr->prev == dll.find(1));
00050         CHECK(ptr->next == dll.find(3));
00051         CHECK(dll.size() == 6);
00052
00053         // List: {-1, 1, 20, 2, 3, 4, 69}
00054         dll.insert(69, 69);  // NOLINT
00055         CHECK(dll.search(69) == dll.find(6));
00056         CHECK(dll.size() == 7);
00057     }
00058
00059     // List: {-1, 1, 20, 2, 3, 4, 69}
00060     SUBCASE("Node_ptr remove(std::size_t index)") {
00061         core::DoublyLinkedList<int> dll{-1, 1, 20, 2, 3, 4, 69};  // NOLINT
00062
00063         CHECK(dll.remove(1000) == nullptr);
00064         CHECK(dll.size() == 7);
00065
00066         // List: {-1, 1, 20, 2, 3, 4}
00067         CHECK(dll.remove(6) == nullptr);
00068         CHECK(dll.size() == 6);
00069
00070         // List: {1, 20, 2, 3, 4}
00071         auto* ptr = dll.remove(0);
00072         CHECK(dll.size() == 5);
00073         CHECK(ptr->data == 1);
00074
00075         // List: {1, 2, 3, 4}
00076         ptr = dll.remove(1);
00077         CHECK(dll.size() == 4);
00078         CHECK(ptr->data == 2);
00079     }
00080 }
```

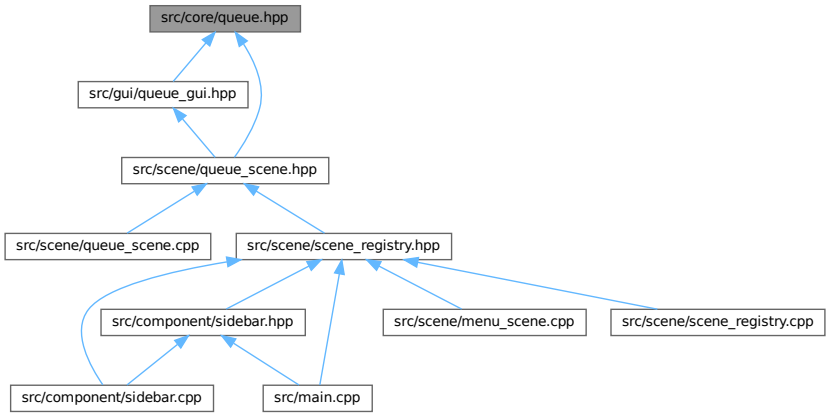
7.41 src/core/queue.hpp File Reference

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

Include dependency graph for queue.hpp:



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



Classes

- class core::Queue< T >

Namespaces

- namespace core

7.42 queue.hpp

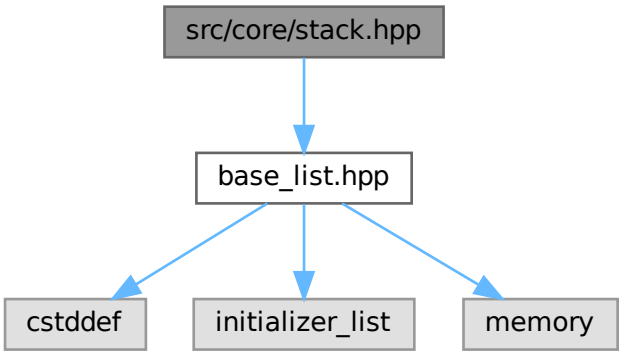
Go to the documentation of this file.

```
00001 #ifndef CORE_QUEUE_HPP_
00002 #define CORE_QUEUE_HPP_
00003
00004 #include "base_list.hpp"
00005
00006 namespace core {
00007
00008 template<typename T>
00009 class Queue : public BaseList<T> {
00010 private:
00011     using Base = BaseList<T>;
00012
00013 public:
00014     using Base::Base;
00015
00016     using Base::empty;
00017     using Base::size;
00018
00019     // for animation purpose only, not for real use
00020     using Base::pop_back;
00021     using Base::push_front;
00022
00023     T& front() const;
00024     T& back() const;
00025
00026     void push(const T& elem);
00027     void pop();
00028 };
00029
00030 template<typename T>
00031 T& Queue<T>::front() const {
00032     return Base::front();
00033 }
00034
00035 template<typename T>
00036 T& Queue<T>::back() const {
00037     return Base::back();
00038 }
00039
00040 template<typename T>
00041 void Queue<T>::push(const T& elem) {
00042     Base::push_back(elem);
00043 }
00044
00045 template<typename T>
00046 void Queue<T>::pop() {
00047     Base::pop_front();
00048 }
00049
00050 }  // namespace core
00051
00052 #endif  // CORE_QUEUE_HPP_
```

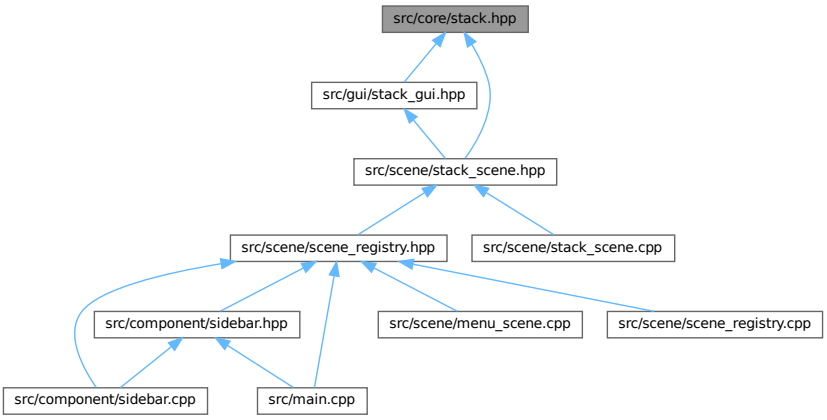
7.43 src/core/stack.hpp File Reference

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

Include dependency graph for stack.hpp:



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



Classes

- class core::Stack< T >

Namespaces

- namespace core

7.44 stack.hpp

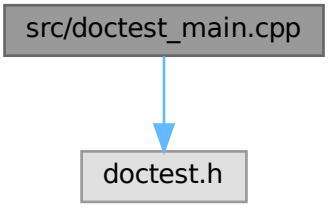
Go to the documentation of this file.

```
00001 #ifndef CORE_STACK_HPP_
00002 #define CORE_STACK_HPP_
00003
00004 #include "base_list.hpp"
00005
00006 namespace core {
00007
00008 template<typename T>
00009 class Stack : public BaseList<T> {
00010 protected:
00011     using Base = BaseList<T>;
00012     using Base::m_head;
00013     using Base::m_tail;
00014
00015 public:
00016     using Base::Base;
00017
00018     using Base::empty;
00019     using Base::size;
00020
00021     T& top() const;
00022
00023     void push(const T& elem);
00024     void pop();
00025 };
00026
00027 template<typename T>
00028 T& Stack<T>::top() const {
00029     return Base::front();
00030 }
00031
00032 template<typename T>
00033 void Stack<T>::push(const T& elem) {
00034     Base::push_front(elem);
00035 }
00036
00037 template<typename T>
00038 void Stack<T>::pop() {
00039     Base::pop_front();
00040 }
00041
00042 }  // namespace core
00043
00044 #endif  // CORE_STACK_HPP_
```

7.45 src/doctest_main.cpp File Reference

`#include "doctest.h"`

Include dependency graph for doctest_main.cpp:



Macros

- #define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN

7.45.1 Macro Definition Documentation

7.45.1.1 DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN

```
#define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN
```

Definition at line 1 of file doctest_main.cpp.

7.46 doctest_main.cpp

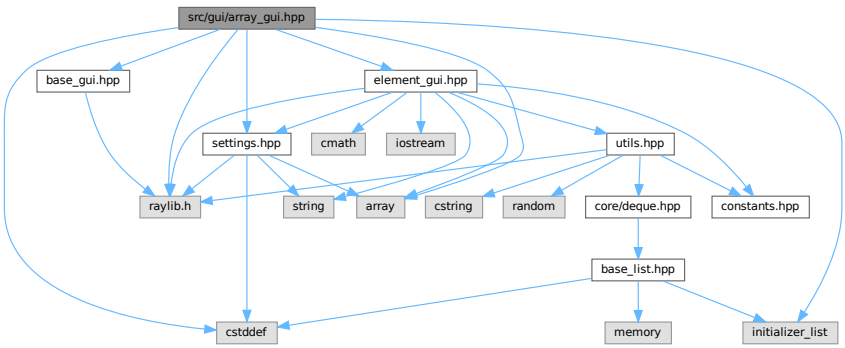
Go to the documentation of this file.

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

7.47 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:



Classes

- class gui::GuiArray< T, N >

Namespaces

- namespace gui

7.48 array_gui.hpp

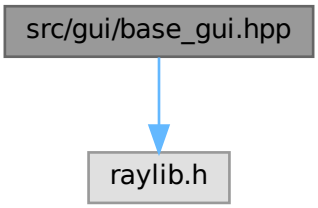
Go to the documentation of this file.

```
00001 #ifndef GUI_ARRAY_GUI_HPP_
00002 #define GUI_ARRAY_GUI_HPP_
00003
00004 #include <array>
00005 #include <cstddef>
00006 #include <initializer_list>
00007
00008 #include "base_gui.hpp"
00009 #include "element_gui.hpp"
00010 #include "raylib.h"
00011 #include "settings.hpp"
00012
00013 namespace gui {
00014
00015 template<typename T, std::size_t N>
00016 class GuiArray : public internal::Base {
00017 private:
00018     static constexpr Vector2 head_pos{
00019         constants::scene_width / 2.0F - 15 * GuiElement<T>::side,
00020         constants::scene_height / 2.0F};
00021
00022     std::array<GuiElement<T>, N> m_array{};
00023
00024     void render_link(Vector2 src, Vector2 dest) override;
00025
00026 public:
00027     GuiArray();
00028     GuiArray(std::array<GuiElement<T>, N>&& init_list);
00029     void update() override;
00030     void render() override;
00031
00032     T& operator[](std::size_t idx);
00033     T operator[](std::size_t idx) const;
00034
00035     void set_color_index(std::size_t idx, int color_index);
00036 };
00037
00038 template<typename T, std::size_t N>
00039 GuiArray<T, N>::GuiArray() {
00040     for (std::size_t i = 0; i < N; ++i) {
00041         m_array[i] = GuiElement<T>{0, i};
00042         m_array[i].set_color_index(0);
00043     }
00044 }
00045
00046 template<typename T, std::size_t N>
00047 GuiArray<T, N>::GuiArray(std::array<GuiElement<T>, N>&& init_list)
00048     : m_array{init_list} {}
00049
00050 template<typename T, std::size_t N>
00051 void GuiArray<T, N>::render_link(Vector2 src, Vector2 dest) {}
00052
00053 template<typename T, std::size_t N>
00054 void GuiArray<T, N>::render() {
00055     update();
00056
00057     for (std::size_t i = 0; i < N; ++i) {
00058         m_array[i].render();
00059     }
00060 }
00061
00062 template<typename T, std::size_t N>
00063 void GuiArray<T, N>::update() {
00064     // TODO: if not outdated then return
00065
00066     for (std::size_t i = 0; i < N; ++i) {
00067         m_array[i].set_pos(
00068             {head_pos.x + 4 * GuiElement<T>::side * i, head_pos.y});
00069     }
00070 }
00071
00072 template<typename T, std::size_t N>
00073 T& GuiArray<T, N>::operator[](std::size_t idx) {
00074     return m_array[idx].get_value();
00075 }
00076
00077 template<typename T, std::size_t N>
00078 T GuiArray<T, N>::operator[](std::size_t idx) const {
00079     return m_array[idx].get_value();
00080 }
00081
00082 template<typename T, std::size_t N>
```

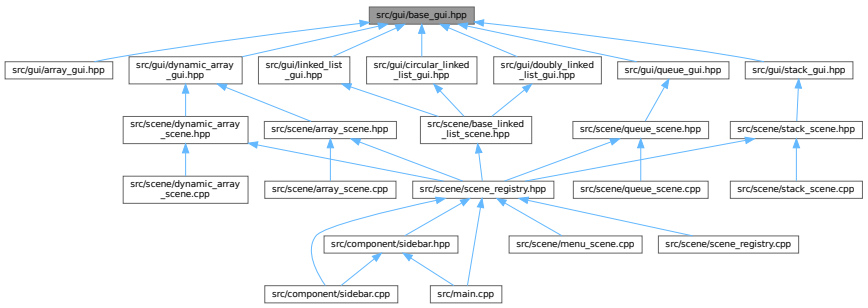
```
00083 void GuiArray<T, N>::set_color_index(std::size_t idx, int color_index) {
00084     m_array[idx].set_color_index(color_index);
00085 }
00086
00087 }  // namespace gui
00088
00089 #endif  // GUI_ARRAY_GUI_HPP_
```

7.49 src/gui/base_gui.hpp File Reference

`#include "raylib.h"`
Include dependency graph for base_gui.hpp:



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



Classes

- class gui::internal::Base

Namespaces

- namespace gui
- namespace gui::internal

7.50 base_gui.hpp

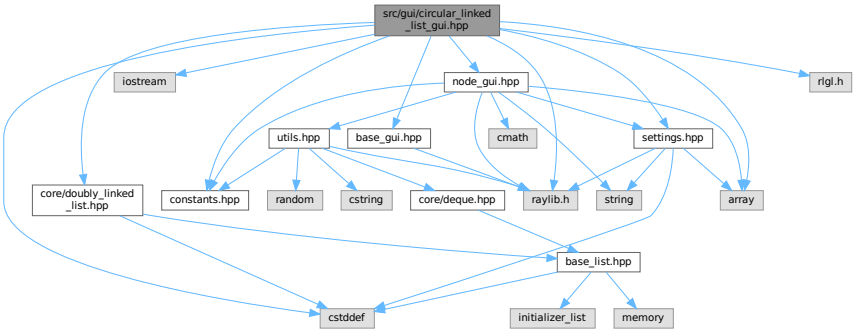
Go to the documentation of this file.

```
00001 #ifndef GUI_BASE_GUI_HPP_
00002 #define GUI_BASE_GUI_HPP_
00003
00004 #include "raylib.h"
00005
00006 namespace gui::internal {
00007
00008 class Base {
00009     virtual void render_link(Vector2 src, Vector2 dest) = 0;
00010
00011 public:
00012     Base() = default;
00013     Base(const Base&) = default;
00014     Base(Base&&) = default;
00015     Base& operator=(const Base&) = default;
00016     Base& operator=(Base&&) = default;
00017
00018     virtual ~Base() = default;
00019
00020     virtual void update() = 0;
00021     virtual void render() = 0;
00022 };
00023
00024 }  // namespace gui::internal
00025
00026 #endif  // GUI_BASE_GUI_HPP_
```

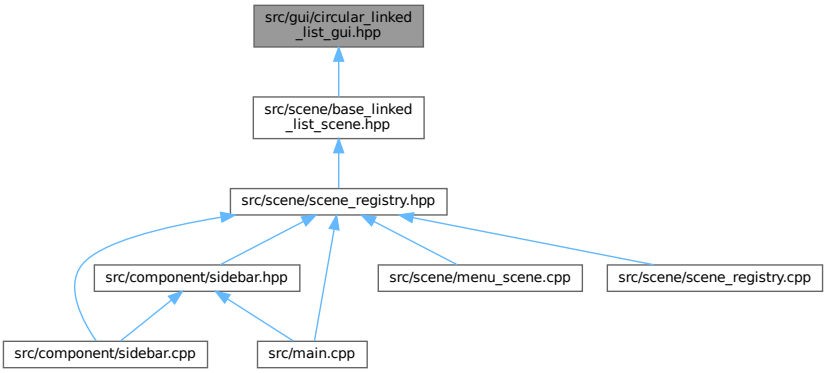
7.51 src/gui/circular_linked_list_gui.hpp File Reference

```
#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:



Classes

- class gui::GuiCircularLinkedList< T >

Namespaces

- namespace gui

7.52 circular_linked_list_gui.hpp

Go to the documentation of this file.

```
00001 #ifndef GUI_CIRCULAR_LINKED_LIST_GUI_HPP_
00002 #define GUI_CIRCULAR_LINKED_LIST_GUI_HPP_
00003
00004 #include <array>
00005 #include <cstddef>
00006 #include <iostream>
00007
00008 #include "base_gui.hpp"
00009 #include "constants.hpp"
00010 #include "core/doubly_linked_list.hpp"
00011 #include "node_gui.hpp"
00012 #include "raylib.h"
00013 #include "rlgl.h"
00014 #include "settings.hpp"
00015
00016 namespace gui {
00017
00018 template<typename T>
00019 class GuiCircularLinkedList : public core::DoublyLinkedList<GuiNode<T»,
00020                               public internal::Base {
00021 private:
00022     using Base = core::DoublyLinkedList<GuiNode<T>>;
00023
00024     static constexpr Vector2 head_pos{
00025         constants::scene_width / 2.0F - 15 * GuiNode<T>::radius,
00026         constants::scene_height / 2.0F};
00027
00028     using Base::m_head;
00029     using Base::m_tail;
00030
00031     void render_link(Vector2 src, Vector2 dest) override;
00032     void render_back_link();
00033
00034 public:
00035     using Base::Base;
```

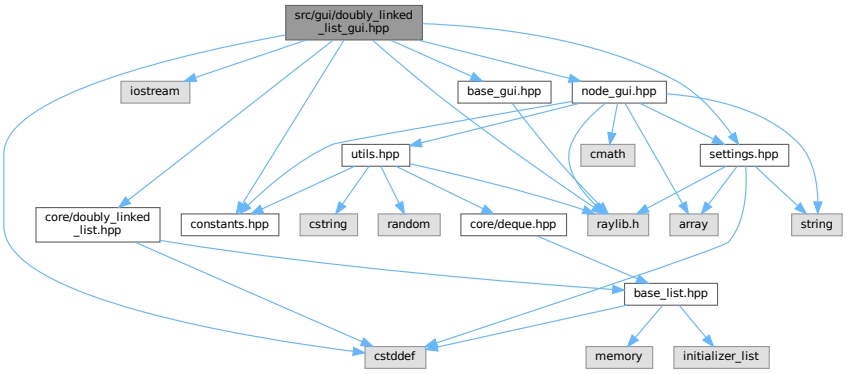
```
00036
00037     using Base::empty;
00038     using Base::size;
00039
00040     GuiCircularLinkedList(std::initializer_list<GuiNode<T>> init_list);
00041
00042     void insert(std::size_t index, const T& elem);
00043
00044     void update() override;
00045     void render() override;
00046     void init_label();
00047 };
00048
00049 template<typename T>
00050 void GuiCircularLinkedList<T>::init_label() {
00051     if (m_head != nullptr) {
00052         m_head->data.set_label("head");
00053     }
00054
00055     if (m_tail != nullptr) {
00056         if (m_head == m_tail) {
00057             m_tail->data.set_label("head/tail");
00058         } else {
00059             m_tail->data.set_label("tail");
00060         }
00061     }
00062 }
00063
00064 template<typename T>
00065 GuiCircularLinkedList<T>::GuiCircularLinkedList(
00066     std::initializer_list<GuiNode<T>> init_list)
00067     : core::DoublyLinkedList<GuiNode<T»(init_list) {
00068     init_label();
00069 }
00070
00071 template<typename T>
00072 void GuiCircularLinkedList<T>::insert(std::size_t index, const T& elem) {
00073     Base::insert(index, GuiNode{elem});
00074 }
00075
00076 template<typename T>
00077 void GuiCircularLinkedList<T>::render_link(Vector2 src, Vector2 dest) {
00078     constexpr int radius = GuiNode<T>::radius;
00079     constexpr float scaled_len = radius / 8.0F;
00080
00081     // straight line
00082     Vector2 link_pos{src.x + radius, src.y - scaled_len};
00083     Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};
00084
00085     // arrow
00086     constexpr int arrow_size = scaled_len * 5;
00087     Vector2 head{dest.x - radius + scaled_len / 2, src.y};
00088     Vector2 side_top{head.x - arrow_size, head.y - arrow_size};
00089     Vector2 side_bot{head.x - arrow_size, head.y + arrow_size};
00090
00091     // draw both
00092     const Settings& settings = Settings::get_instance();
00093     DrawRectangleV(link_pos, link_size, settings.get_color(1));
00094     DrawTriangle(head, side_top, side_bot, settings.get_color(1));
00095 }
00096
00097 template<typename T>
00098 void GuiCircularLinkedList<T>::render_back_link() {
00099     if (m_head == nullptr && m_tail == nullptr) {
00100         return;
00101     }
00102
00103     constexpr int num_points = 5;
00104     const Vector2 head_pos = m_head->data.get_pos();
00105     const Vector2 tail_pos = m_tail->data.get_pos();
00106     constexpr int radius = GuiNode<T>::radius;
00107     constexpr float scaled_len = radius / 8.0F;
00108
00109     std::array<Vector2, num_points> points{{
00110         tail_pos,
00111         {tail_pos.x + 2 * radius, tail_pos.y},
00112         {tail_pos.x + 2 * radius, tail_pos.y + 3 * radius},
00113         {head_pos.x, tail_pos.y + 3 * radius},
00114         head_pos,
00115     }};
00116
00117     constexpr int arrow_size = scaled_len * 5;
00118     Vector2 head{head_pos.x, head_pos.y + radius - scaled_len / 2};
00119     Vector2 side_left{head.x - arrow_size, head.y + arrow_size};
00120     Vector2 side_right{head.x + arrow_size, head.y + arrow_size};
00121
00122     const Settings& settings = Settings::get_instance();
```

```
00123     rlSetLineWidth(2 * scaled_len);
00124     DrawLineStrip(points.data(), num_points, settings.get_color(1));
00125     DrawTriangle(head, side_left, side_right, settings.get_color(1));
00126 }
00127
00128 template<typename T>
00129 void GuiCircularLinkedList<T>::render() {
00130     update();
00131
00132     render_back_link();
00133     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00134         if (ptr->next != nullptr) {
00135             render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
00136         }
00137
00138         ptr->data.render();
00139     }
00140 }
00141
00142 template<typename T>
00143 void GuiCircularLinkedList<T>::update() {
00144     // TODO: if not outdated then return
00145
00146     std::size_t pos = 0;
00147
00148     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00149         ptr->data.set_pos(
00150             {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
00151         ++pos;
00152     }
00153 }
00154
00155 }  // namespace gui
00156
00157 #endif  // GUI_CIRCULAR_LINKED_LIST_GUI_HPP_
```

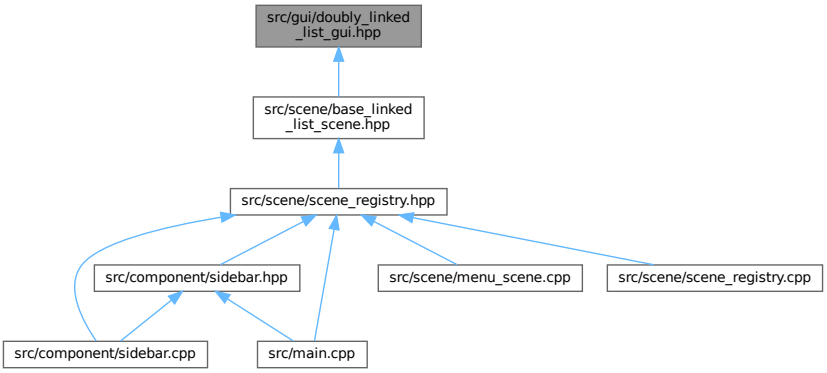
7.53 src/gui/doubly_linked_list_gui.hpp File Reference

```
#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:



Classes

- class gui::GuiDoublyLinkedList< T >

Namespaces

- namespace gui

7.54 doubly_linked_list_gui.hpp

Go to the documentation of this file.

```
00001 #ifndef GUI_DOUBLY_LINKED_LIST_GUI_HPP_
00002 #define GUI_DOUBLY_LINKED_LIST_GUI_HPP_
00003
00004 #include <cstddef>
00005 #include <iostream>
00006
00007 #include "base_gui.hpp"
00008 #include "constants.hpp"
00009 #include "core/doubly_linked_list.hpp"
00010 #include "node_gui.hpp"
00011 #include "raylib.h"
00012 #include "settings.hpp"
00013
00014 namespace gui {
00015
00016 template<typename T>
00017 class GuiDoublyLinkedList : public core::DoublyLinkedList<GuiNode<T>,
00018                             public internal::Base {
00019 private:
00020     using Base = core::DoublyLinkedList<GuiNode<T>>;
00021
00022     static constexpr Vector2 head_pos{
00023         constants::scene_width / 2.0F - 15 * GuiNode<T>::radius,
00024         constants::scene_height / 2.0F};
00025
00026     using Base::m_head;
00027     using Base::m_tail;
00028
00029     void render_link(Vector2 src, Vector2 dest) override;
00030
00031 public:
00032     using Base::Base;
00033
00034     using Base::empty;
00035     using Base::size;
```

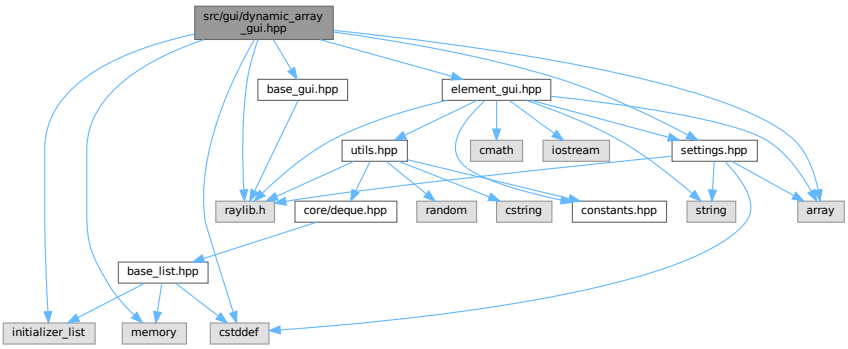
```
00036
00037     GuiDoublyLinkedList(std::initializer_list<GuiNode<T>> init_list);
00038
00039     void insert(std::size_t index, const T& elem);
00040
00041     void update() override;
00042     void render() override;
00043     void init_label();
00044 };
00045
00046 template<typename T>
00047 void GuiDoublyLinkedList<T>::init_label() {
00048     if (m_head != nullptr) {
00049         m_head->data.set_label("head");
00050     }
00051
00052     if (m_tail != nullptr) {
00053         if (m_head == m_tail) {
00054             m_tail->data.set_label("head/tail");
00055         } else {
00056             m_tail->data.set_label("tail");
00057         }
00058     }
00059 }
00060
00061 template<typename T>
00062 GuiDoublyLinkedList<T>::GuiDoublyLinkedList(
00063     std::initializer_list<GuiNode<T>> init_list)
00064     : core::DoublyLinkedList<GuiNode<T»(init_list) {
00065     init_label();
00066 }
00067
00068 template<typename T>
00069 void GuiDoublyLinkedList<T>::insert(std::size_t index, const T& elem) {
00070     Base::insert(index, GuiNode{elem});
00071 }
00072
00073 template<typename T>
00074 void GuiDoublyLinkedList<T>::render_link(Vector2 src, Vector2 dest) {
00075     constexpr int radius = GuiNode<T>::radius;
00076     constexpr float scaled_len = radius / 8.0F;
00077
00078     // straight line
00079     Vector2 link_pos{src.x + radius, src.y - scaled_len};
00080     Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};
00081
00082     // right arrow
00083     constexpr int arrow_size = scaled_len * 5;
00084     Vector2 right_head{dest.x - radius + scaled_len / 2, src.y};
00085     Vector2 right_side_top{right_head.x - arrow_size,
00086                            right_head.y - arrow_size};
00087     Vector2 right_side_bot{right_head.x - arrow_size,
00088                            right_head.y + arrow_size};
00089
00090     // left arrow
00091     Vector2 left_head{src.x + radius - scaled_len / 2, src.y};
00092     Vector2 left_side_top{left_head.x + arrow_size, left_head.y - arrow_size};
00093     Vector2 left_side_bot{left_head.x + arrow_size, left_head.y + arrow_size};
00094
00095     // draw all
00096     const Settings& settings = Settings::get_instance();
00097     DrawRectangleV(link_pos, link_size, settings.get_color(1));
00098     DrawTriangle(right_head, right_side_top, right_side_bot,
00099                  settings.get_color(1));
00100     DrawTriangle(left_head, left_side_bot, left_side_top,
00101                  settings.get_color(1));
00102 }
00103
00104 template<typename T>
00105 void GuiDoublyLinkedList<T>::render() {
00106     update();
00107
00108     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00109         if (ptr->next != nullptr) {
00110             render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
00111         }
00112
00113         ptr->data.render();
00114     }
00115 }
00116
00117 template<typename T>
00118 void GuiDoublyLinkedList<T>::update() {
00119     // TODO: if not outdated then return
00120
00121     std::size_t pos = 0;
00122
```

```
00123     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00124         ptr->data.set_pos(
00125             {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
00126         ++pos;
00127     }
00128 }
00129
00130 }  // namespace gui
00131
00132 #endif  // GUI_DOUBLY_LINKED_LIST_GUI_HPP_
```

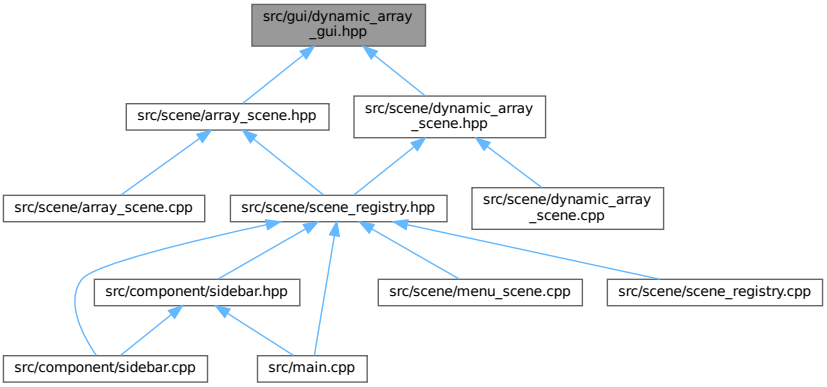
7.55 src/gui/dynamic_array_gui.hpp File Reference

```
#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:



Classes

- class gui::GuiDynamicArray< T >

Namespaces

- namespace gui

7.56 dynamic_array_gui.hpp

Go to the documentation of this file.

```
00001 #ifndef GUI_DYNAMIC_ARRAY_GUI_HPP_
00002 #define GUI_DYNAMIC_ARRAY_GUI_HPP_
00003
00004 #include <array>
00005 #include <cstddef>
00006 #include <initializer_list>
00007 #include <memory>
00008
00009 #include "base_gui.hpp"
00010 #include "element_gui.hpp"
00011 #include "raylib.h"
00012 #include "settings.hpp"
00013
00014 namespace gui {
00015
00016 template<typename T>
00017 class GuiDynamicArray : public internal::Base {
00018 private:
00019     static constexpr Vector2 head_pos{
00020         constants::scene_width / 2.0F - 15 * GuiElement<T>::side,
00021         constants::scene_height / 2.0F};
00022
00023     std::size_t m_capacity{2};
00024     std::size_t m_size{};
00025     GuiElement<T>* m_ptr{nullptr};
00026
00027     void render_link(Vector2 src, Vector2 dest) override;
00028
00029 public:
00030     GuiDynamicArray();
00031     GuiDynamicArray(std::initializer_list<T> init_list);
00032     GuiDynamicArray(const GuiDynamicArray& other);
00033     GuiDynamicArray(GuiDynamicArray&& other) noexcept;
00034     GuiDynamicArray& operator=(const GuiDynamicArray& other);
00035     GuiDynamicArray& operator=(GuiDynamicArray&& other) noexcept;
00036     ~GuiDynamicArray() override;
00037
00038     void update() override;
00039     void render() override;
00040
00041     T& operator[](std::size_t idx);
00042     T operator[](std::size_t idx) const;
00043
00044     void set_color_index(std::size_t idx, int color_index);
00045     void reserve(std::size_t capacity);
00046     void shrink_to_fit();
00047
00048     std::size_t capacity() const;
00049     std::size_t size() const;
00050
00051     void push(const T& value);
00052     void pop();
00053 };
00054
00055 template<typename T>
00056 void GuiDynamicArray<T>::reserve(std::size_t capacity) {
00057     capacity = std::min(capacity, static_cast<std::size_t>(8));
00058     if (m_capacity > capacity) {
00059         return;
00060     }
00061
00062     auto* new_ptr = static_cast<GuiElement<T>*>(
00063         ::operator new[](capacity * sizeof(GuiElement<T>)));
00064     for (auto i = 0; i < m_size; ++i) {
00065         ::new (&new_ptr[i]) GuiElement<T>(std::move(m_ptr[i]));
```

```
00066         m_ptr[i].~GuiElement<T>();
00067     }
00068     for (auto i = m_size; i < capacity; ++i) {
00069         ::new (&new_ptr[i]) GuiElement<T>();
00070         new_ptr[i].set_index(i);
00071     }
00072
00073     ::operator delete[](m_ptr);
00074     m_ptr = new_ptr;
00075     m_capacity = capacity;
00076 }
00077
00078 template<typename T>
00079 void GuiDynamicArray<T>::shrink_to_fit() {
00080     if (m_capacity == m_size) {
00081         return;
00082     }
00083
00084     auto* new_ptr = static_cast<GuiElement<T>*>(
00085         ::operator new[](m_size * sizeof(GuiElement<T>)));
00086     for (auto i = 0; i < m_size; ++i) {
00087         ::new (&new_ptr[i]) GuiElement<T>(std::move(m_ptr[i]));
00088     }
00089     for (auto i = 0; i < m_capacity; ++i) {
00090         m_ptr[i].~GuiElement<T>();
00091     }
00092
00093     ::operator delete[](m_ptr);
00094     m_ptr = new_ptr;
00095     m_capacity = m_size;
00096 }
00097
00098 template<typename T>
00099 GuiDynamicArray<T>::GuiDynamicArray() : m_ptr{new GuiElement<T>[m_capacity]} {
00100     for (auto i = 0; i < m_capacity; ++i) {
00101         m_ptr[i].set_index(i);
00102     }
00103 }
00104
00105 template<typename T>
00106 GuiDynamicArray<T>::GuiDynamicArray(std::initializer_list<T> init_list)
00107     : m_size{init_list.size()}, m_ptr{new GuiElement<T>[m_capacity]} {
00108     reserve(m_size);
00109
00110     for (std::size_t idx = 0; auto elem : init_list) {
00111         *(m_ptr + idx).set_value(elem);
00112         *(m_ptr + idx).set_color(Settings::get_instance().get_color(0));
00113     }
00114 }
00115
00116 template<typename T>
00117 GuiDynamicArray<T>::GuiDynamicArray(const GuiDynamicArray<T>& other)
00118     : m_capacity{other.m_capacity},
00119       m_size{other.m_size},
00120       m_ptr{new GuiElement<T>[m_capacity]} {
00121     for (auto i = 0; i < m_capacity; ++i) {
00122         m_ptr[i] = other.m_ptr[i];
00123     }
00124 }
00125
00126 template<typename T>
00127 GuiDynamicArray<T>::GuiDynamicArray(GuiDynamicArray<T>&& other) noexcept
00128     : m_capacity{other.m_capacity}, m_size{other.m_size}, m_ptr{other.m_ptr} {
00129     other.m_capacity = 0;
00130     other.m_size = 0;
00131     other.m_ptr = nullptr;
00132 }
00133
00134 template<typename T>
00135 GuiDynamicArray<T>& GuiDynamicArray<T>::operator=(
00136     const GuiDynamicArray<T>& other) {
00137     if (&other != this) {
00138         m_capacity = other.m_capacity;
00139         m_size = other.m_size;
00140
00141         m_ptr = new GuiDynamicArray<T>[m_capacity];
00142         for (auto i = 0; i < m_capacity; ++i) {
00143             m_ptr[i] = other.m_ptr[i];
00144         }
00145     }
00146
00147     return *this;
00148 }
00149
00150 template<typename T>
00151 GuiDynamicArray<T>& GuiDynamicArray<T>::operator=(
00152     GuiDynamicArray&& other) noexcept {
```

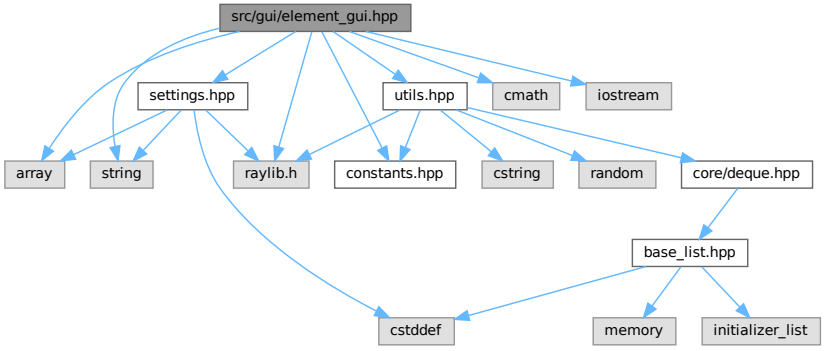
```
00153     m_capacity = other.m_capacity;
00154     m_size = other.m_size;
00155     m_ptr = other.m_ptr;
00156 
00157     other.m_capacity = 0;
00158     other.m_size = 0;
00159     other.m_ptr = nullptr;
00160 
00161     return *this;
00162 }
00163 
00164 template<typename T>
00165 GuiDynamicArray<T>::~GuiDynamicArray() {
00166     delete[] m_ptr;
00167 }
00168 
00169 template<typename T>
00170 void GuiDynamicArray<T>::render_link(Vector2 src, Vector2 dest) {}
00171 
00172 template<typename T>
00173 void GuiDynamicArray<T>::render() {
00174     update();
00175 
00176     std::size_t idx = 0;
00177 
00178     for (std::size_t i = 0; i < m_capacity; ++i) {
00179         m_ptr[i].render();
00180     }
00181 }
00182 
00183 template<typename T>
00184 void GuiDynamicArray<T>::update() {
00185     // TODO: if not outdated then return
00186 
00187     for (std::size_t i = 0; i < m_capacity; ++i) {
00188         m_ptr[i].set_pos(
00189             {head_pos.x + 4 * GuiElement<T>::side * i, head_pos.y});
00190     }
00191 }
00192 
00193 template<typename T>
00194 T& GuiDynamicArray<T>::operator[](std::size_t idx) {
00195     return m_ptr[idx].get_value();
00196 }
00197 
00198 template<typename T>
00199 T GuiDynamicArray<T>::operator[](std::size_t idx) const {
00200     return m_ptr[idx].get_value();
00201 }
00202 
00203 template<typename T>
00204 void GuiDynamicArray<T>::set_color_index(std::size_t idx, int color_index) {
00205     m_ptr[idx].set_color_index(color_index);
00206 }
00207 
00208 template<typename T>
00209 std::size_t GuiDynamicArray<T>::capacity() const {
00210     return m_capacity;
00211 }
00212 
00213 template<typename T>
00214 std::size_t GuiDynamicArray<T>::size() const {
00215     return m_size;
00216 }
00217 
00218 template<typename T>
00219 void GuiDynamicArray<T>::push(const T& value) {
00220     if (m_size == m_capacity) {
00221         reserve(std::max(m_capacity * 2, static_cast<std::size_t>(1)));
00222     }
00223 
00224     m_ptr[m_size].set_color_index(0);
00225     m_ptr[m_size].set_value(value);
00226     ++m_size;
00227 }
00228 
00229 template<typename T>
00230 void GuiDynamicArray<T>::pop() {
00231     if (m_size >= 1) {
00232         m_ptr[m_size - 1].set_color_index(1);
00233         m_ptr[m_size - 1].set_value(0);
00234         --m_size;
00235     }
00236 }
00237 
00238 }  // namespace gui
00239 
```

```
00240 #endif  // GUI_DYNAMIC_ARRAY_GUI_HPP_
```

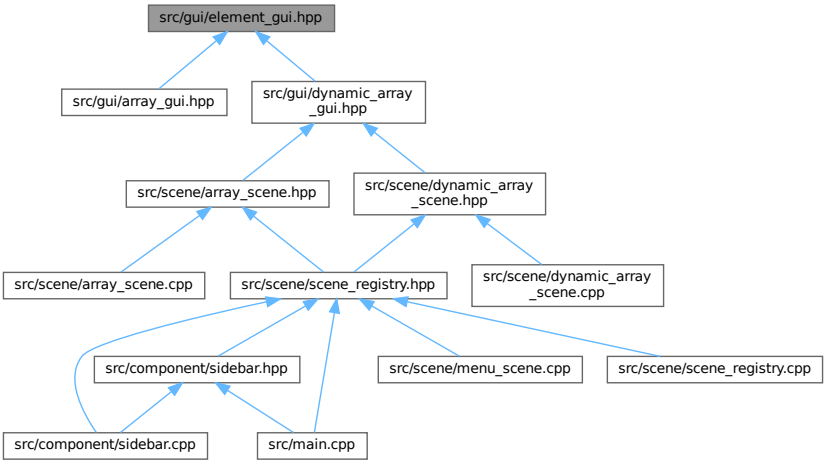
7.57 src/gui/element_gui.hpp File Reference

```
#include <array>
#include <cmath>
#include <iostream>
#include <string>
#include "constants.hpp"
#include "raylib.h"
#include "settings.hpp"
#include "utils.hpp"
```

Include dependency graph for element_gui.hpp:



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



Classes

- class gui::GuiElement< T >

Namespaces

- namespace gui

7.58 element_gui.hpp

Go to the documentation of this file.

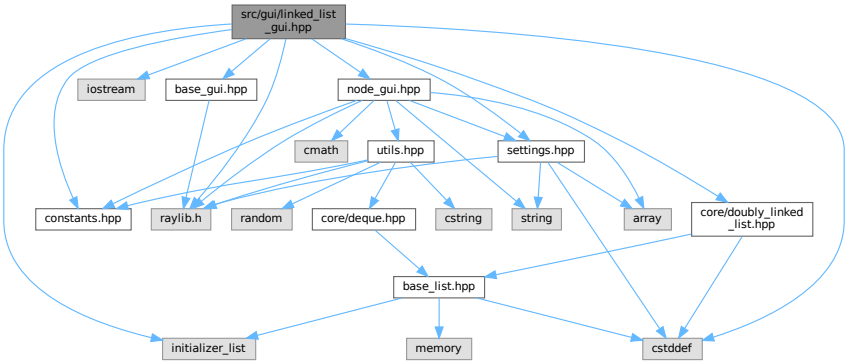
```
00001 #ifndef GUI_ELEMENT_GUI_HPP_
00002 #define GUI_ELEMENT_GUI_HPP_
00003
00004 #include <array>
00005 #include <cmath>
00006 #include <iostream>
00007 #include <string>
00008
00009 #include "constants.hpp"
00010 #include "raylib.h"
00011 #include "settings.hpp"
00012 #include "utils.hpp"
00013
00014 namespace gui {
00015
00016 template<typename T>
00017 class GuiElement {
00018 private:
00019     T m_value{};
00020     std::size_t m_index{};
00021
00022     Vector2 m_pos{init_pos};
00023     static constexpr float eps = 1e-3;
00024     int m_color_index{1};
00025
00026 public:
00027     static constexpr int side = 20;
00028     static constexpr Vector2 init_pos{
00029         constants::sidebar_width +
00030             static_cast<float>(constants::scene_width -
00031                                constants::sidebar_width) /
00032                 2,
00033         0};
00034
00035     GuiElement() = default;
00036     GuiElement(const T& value, std::size_t index);
00037
00038     void render();
00039     void set_pos(Vector2 pos);
00040     void set_color_index(int color_index);
00041     [[nodiscard]] Vector2 get_pos() const;
00042
00043     T& get_value();
00044     T get_value() const;
00045     void set_value(const T& value);
00046     void set_index(std::size_t index);
00047 };
00048
00049 template<typename T>
00050 GuiElement<T>::GuiElement(const T& value, std::size_t index)
00051     : m_value{value}, m_index{index} {}
00052
00053 template<typename T>
00054 void GuiElement<T>::render() {
00055     constexpr int label_font_size = 25;
00056     constexpr int label_font_spacing = 2;
00057     const std::string label = std::to_string(m_value);
00058     const std::string index = std::to_string(m_index);
00059
00060     const Vector2 label_size =
00061         utils::MeasureText(label.c_str(), label_font_size, label_font_spacing);
00062
00063     const Vector2 label_pos{m_pos.x - label_size.x / 2,
00064                             m_pos.y - label_size.y / 2};
00065
00066     const Vector2 index_size =
00067         utils::MeasureText(index.c_str(), label_font_size, label_font_spacing);
00068
00069     const Vector2 index_pos{m_pos.x - index_size.x / 2,
00070                             m_pos.y - 2 * side - index_size.y / 2};
00071
00072     const Color value_color =
00073         utils::adaptive_text_color(Settings::get_instance().get_color(0));
```

```
00074     const Color index_color = utils::adaptive_text_color(
00075         Settings::get_instance().get_color(Settings::num_color - 1));
00076
00077     DrawRectangle(m_pos.x - side,  // NOLINT
00078                   m_pos.y - side,  // NOLINT
00079                   2 * side, 2 * side,
00080                   Settings::get_instance().get_color(m_color_index));
00081
00082     utils::DrawText(label.c_str(), label_pos, value_color, label_font_size,
00083                     label_font_spacing);
00084
00085     utils::DrawText(index.c_str(), index_pos, index_color, label_font_size,
00086                     label_font_spacing);
00087 }
00088
00089 template<typename T>
00090 void GuiElement<T>::set_pos(Vector2 pos) {
00091     m_pos = pos;
00092 }
00093
00094 template<typename T>
00095 void GuiElement<T>::set_color_index(int color_index) {
00096     m_color_index = color_index;
00097 }
00098
00099 template<typename T>
00100 T& GuiElement<T>::get_value() {
00101     return m_value;
00102 }
00103
00104 template<typename T>
00105 T GuiElement<T>::get_value() const {
00106     return m_value;
00107 }
00108
00109 template<typename T>
00110 void GuiElement<T>::set_value(const T& value) {
00111     m_value = value;
00112 }
00113
00114 template<typename T>
00115 void GuiElement<T>::set_index(std::size_t index) {
00116     m_index = index;
00117 }
00118
00119 }  // namespace gui
00120
00121 #endif  // GUI_ELEMENT_GUI_HPP_
```

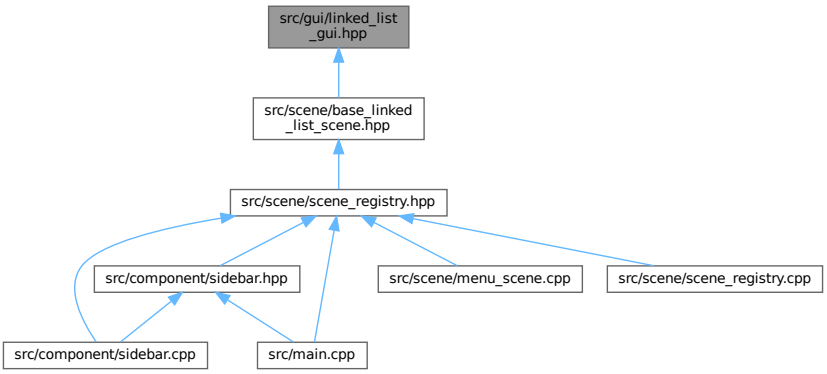
7.59 src/gui/linked_list_gui.hpp File Reference

```
#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:



Classes

- class gui::GuiLinkedList< T >

Namespaces

- namespace gui

7.60 linked_list_gui.hpp

Go to the documentation of this file.

```
00001 #ifndef GUI_LINKED_LIST_GUI_HPP_
00002 #define GUI_LINKED_LIST_GUI_HPP_
00003
00004 #include <cstddef>
00005 #include <initializer_list>
00006 #include <iostream>
```

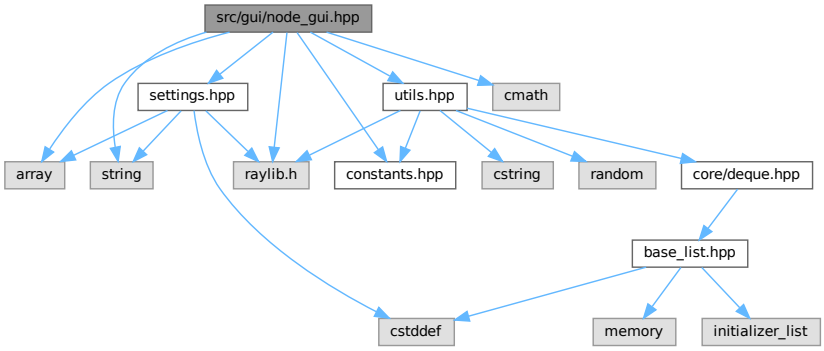
```
00007
00008 #include "base_gui.hpp"
00009 #include "constants.hpp"
00010 #include "core/doubly_linked_list.hpp"
00011 #include "node_gui.hpp"
00012 #include "raylib.h"
00013 #include "settings.hpp"
00014
00015 namespace gui {
00016
00017 template<typename T>
00018 class GuiLinkedList : public core::DoublyLinkedList<GuiNode<T>,
00019                       public internal::Base {
00020 private:
00021     using Base = core::DoublyLinkedList<GuiNode<T>>;
00022
00023     static constexpr Vector2 head_pos{
00024         constants::scene_width / 2.0F - 15 * GuiNode<T>::radius,
00025         constants::scene_height / 2.0F};
00026
00027     using Base::m_head;
00028     using Base::m_tail;
00029
00030     void render_link(Vector2 src, Vector2 dest) override;
00031
00032 public:
00033     using Base::Base;
00034
00035     using Base::empty;
00036     using Base::size;
00037
00038     GuiLinkedList(std::initializer_list<GuiNode<T>> init_list);
00039
00040     void insert(std::size_t index, const T& elem);
00041
00042     void update() override;
00043     void render() override;
00044     void init_label();
00045 };
00046
00047 template<typename T>
00048 void GuiLinkedList<T>::init_label() {
00049     if (m_head != nullptr) {
00050         m_head->data.set_label("head");
00051     }
00052
00053     if (m_tail != nullptr) {
00054         if (m_head == m_tail) {
00055             m_tail->data.set_label("head/tail");
00056         } else {
00057             m_tail->data.set_label("tail");
00058         }
00059     }
00060 }
00061
00062 template<typename T>
00063 GuiLinkedList<T>::GuiLinkedList(std::initializer_list<GuiNode<T>> init_list)
00064     : core::DoublyLinkedList<GuiNode<T»(init_list) {
00065     init_label();
00066 }
00067
00068 template<typename T>
00069 void GuiLinkedList<T>::insert(std::size_t index, const T& elem) {
00070     Base::insert(index, GuiNode{elem});
00071 }
00072
00073 template<typename T>
00074 void GuiLinkedList<T>::render_link(Vector2 src, Vector2 dest) {
00075     constexpr int radius = GuiNode<T>::radius;
00076     constexpr float scaled_len = radius / 8.0F;
00077
00078     // straight line
00079     Vector2 link_pos{src.x + radius, src.y - scaled_len};
00080     Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};
00081
00082     // arrow
00083     constexpr int arrow_size = scaled_len * 5;
00084     Vector2 head{dest.x - radius + scaled_len / 2, src.y};
00085     Vector2 side_top{head.x - arrow_size, head.y - arrow_size};
00086     Vector2 side_bot{head.x - arrow_size, head.y + arrow_size};
00087
00088     // draw both
00089     DrawRectangleV(link_pos, link_size, Settings::get_instance().get_color(1));
00090     DrawTriangle(head, side_top, side_bot,
00091                  Settings::get_instance().get_color(1));
00092 }
00093
```

```
00094 template<typename T>
00095 void GuiLinkedList<T>::render() {
00096     update();
00097
00098     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00099         if (ptr->next != nullptr) {
00100             render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
00101         }
00102
00103         ptr->data.render();
00104     }
00105 }
00106
00107 template<typename T>
00108 void GuiLinkedList<T>::update() {
00109     // TODO: if not outdated then return
00110
00111     std::size_t pos = 0;
00112
00113     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00114         ptr->data.set_pos(
00115             {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
00116         ++pos;
00117     }
00118 }
00119
00120 }  // namespace gui
00121
00122 #endif  // GUI_LINKED_LIST_GUI_HPP_
```

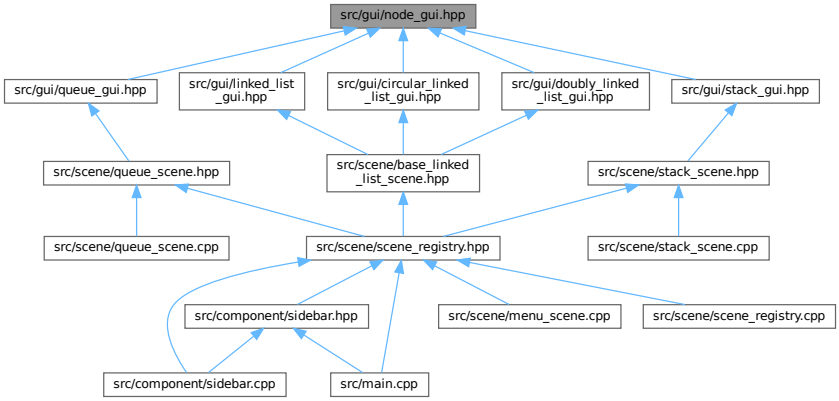
7.61 src/gui/node_gui.hpp File Reference

```
#include <array>
#include <cmath>
#include <string>
#include "constants.hpp"
#include "raylib.h"
#include "settings.hpp"
#include "utils.hpp"
```

Include dependency graph for node_gui.hpp:



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



Classes

- class gui::GuiNode< T >

Namespaces

- namespace gui

7.62 node_gui.hpp

Go to the documentation of this file.

```
00001 #ifndef GUI_NODE_GUI_HPP_
00002 #define GUI_NODE_GUI_HPP_
00003
00004 #include <array>
00005 #include <cmath>
00006 #include <string>
00007
00008 #include "constants.hpp"
00009 #include "raylib.h"
00010 #include "settings.hpp"
00011 #include "utils.hpp"
00012
00013 namespace gui {
00014
00015 template<typename T>
00016 class GuiNode {
00017 private:
00018     T m_value{};
00019     int m_color_index{0};
00020
00021     Vector2 m_pos{constants::sidebar_width +
00022                   static_cast<float>(constants::scene_width -
00023                                      constants::sidebar_width) /
00024                       2,
00025                   0};
00026     static constexpr float eps = 1e-3;
00027     const char* m_label{};
00028
00029 public:
00030     static constexpr int radius = 20;
00031
00032     explicit GuiNode(const T& value);
00033
00034     void render();
```

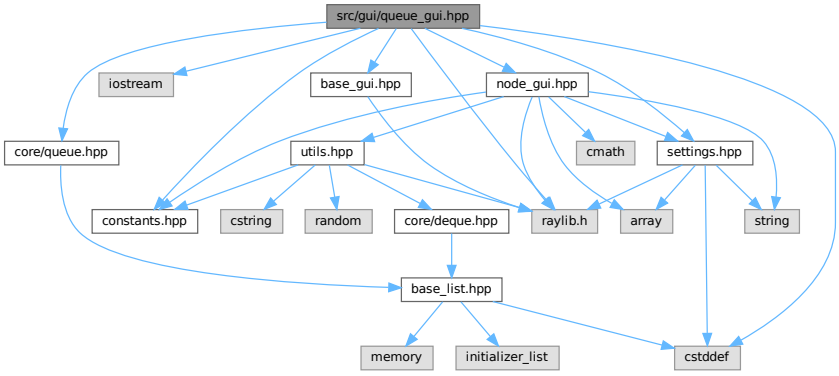
```
00035     void set_pos(Vector2 pos);
00036     [[nodiscard]] Vector2 get_pos() const;
00037     void set_color_index(int color_index);
00038     void set_value(const T& value);
00039     T& get_value();
00040     void set_label(const char* label);
00041 };
00042
00043 template<typename T>
00044 GuiNode<T>::GuiNode(const T& value) : m_value{value} {}
00045
00046 template<typename T>
00047 void GuiNode<T>::render() {
00048     constexpr int label_font_size = 25;
00049     constexpr int label_font_spacing = 2;
00050     const std::string value = std::to_string(m_value);
00051     const Settings& settings = Settings::get_instance();
00052
00053     const Vector2 value_size =
00054         utils::MeasureText(value.c_str(), label_font_size, label_font_spacing);
00055
00056     const Vector2 value_pos{m_pos.x - value_size.x / 2,
00057                             m_pos.y - value_size.y / 2};
00058
00059     const Vector2 label_size =
00060         utils::MeasureText(m_label, label_font_size, label_font_spacing);
00061
00062     const Vector2 label_pos{m_pos.x - label_size.x / 2,
00063                             m_pos.y - 2 * label_size.y};
00064
00065     const Color value_color =
00066         utils::adaptive_text_color(Settings::get_instance().get_color(0));
00067
00068     DrawCircleV(m_pos, radius, settings.get_color(m_color_index));
00069     utils::DrawText(value.c_str(), value_pos, value_color, label_font_size,
00070                     label_font_spacing);
00071
00072     utils::DrawText(m_label, label_pos, settings.get_color(5), label_font_size,
00073                     label_font_spacing);
00074 }
00075
00076 template<typename T>
00077 void GuiNode<T>::set_color_index(int color_index) {
00078     m_color_index = color_index;
00079 }
00080
00081 template<typename T>
00082 void GuiNode<T>::set_value(const T& value) {
00083     m_value = value;
00084 }
00085
00086 template<typename T>
00087 T& GuiNode<T>::get_value() {
00088     return m_value;
00089 }
00090
00091 template<typename T>
00092 void GuiNode<T>::set_pos(Vector2 pos) {
00093     m_pos = pos;
00094 }
00095
00096 template<typename T>
00097 Vector2 GuiNode<T>::get_pos() const {
00098     return m_pos;
00099 }
00100
00101 template<typename T>
00102 void GuiNode<T>::set_label(const char* label) {
00103     m_label = label;
00104 }
00105
00106 }  // namespace gui
00107
00108 #endif  // GUI_NODE_GUI_HPP_
```

7.63 src/gui/queue_gui.hpp File Reference

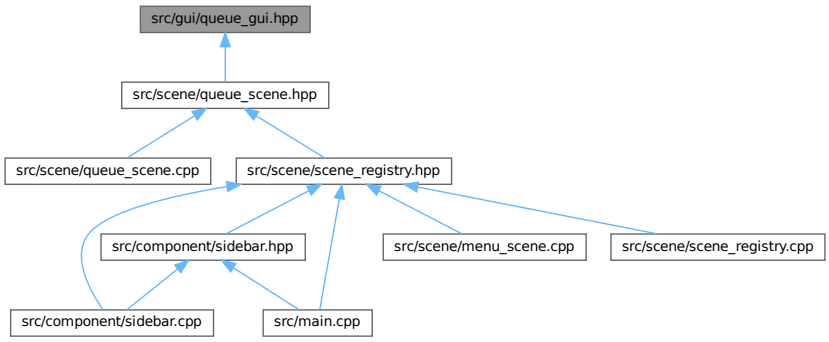
```
#include <cstddef>
#include <iostream>
#include "base_gui.hpp"
```

```
#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:



Classes

- class gui::GuiQueue< T >

Namespaces

- namespace gui

7.64 queue_gui.hpp

Go to the documentation of this file.

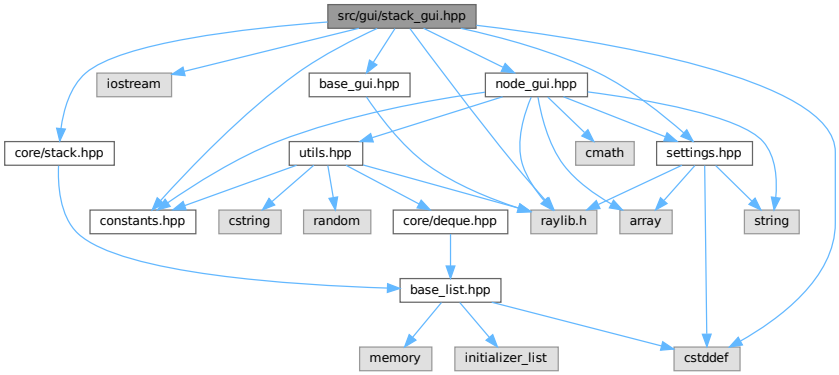
```
00001 #ifndef GUI_QUEUE_GUI_HPP_
00002 #define GUI_QUEUE_GUI_HPP_
00003
00004 #include <cstddef>
00005 #include <iostream>
00006
00007 #include "base_gui.hpp"
00008 #include "constants.hpp"
00009 #include "core/queue.hpp"
00010 #include "node_gui.hpp"
00011 #include "raylib.h"
00012 #include "settings.hpp"
00013
00014 namespace gui {
00015
00016 template<typename T>
00017 class GuiQueue : public core::Queue<GuiNode<T>, public internal::Base {
00018 private:
00019     using Base = core::Queue<GuiNode<T>>;
00020
00021     static constexpr Vector2 head_pos{
00022         constants::scene_width / 2.0F - 15 * GuiNode<T>::radius,
00023         constants::scene_height / 2.0F};
00024
00025     using Base::m_head;
00026     using Base::m_tail;
00027
00028     void render_link(Vector2 src, Vector2 dest) override;
00029
00030 public:
00031     using Base::Base;
00032
00033     using Base::empty;
00034     using Base::size;
00035
00036     GuiQueue(std::initializer_list<GuiNode<T>> init_list);
00037
00038     void push(const T& elem);
00039     void pop();
00040
00041     // for animation purpose only, not for real use
00042     void push_front(const T& elem);
00043     void pop_back();
00044
00045     void update() override;
00046     void render() override;
00047     void init_label();
00048 };
00049
00050 template<typename T>
00051 void GuiQueue<T>::init_label() {
00052     if (m_head != nullptr) {
00053         m_head->data.set_label("head");
00054     }
00055
00056     if (m_tail != nullptr) {
00057         if (m_head == m_tail) {
00058             m_tail->data.set_label("head/tail");
00059         } else {
00060             m_tail->data.set_label("tail");
00061         }
00062     }
00063 }
00064
00065 template<typename T>
00066 GuiQueue<T>::GuiQueue(std::initializer_list<GuiNode<T>> init_list)
00067     : core::Queue<GuiNode<T>(init_list) {
00068     init_label();
00069 }
00070
00071 template<typename T>
00072 void GuiQueue<T>::push(const T& elem) {
00073     Base::push(GuiNode<T>{elem});
00074 }
00075
00076 template<typename T>
00077 void GuiQueue<T>::pop() {
00078     Base::pop();
00079 }
00080
00081 template<typename T>
00082 void GuiQueue<T>::push_front(const T& elem) {
```

```
00083     Base::push_front(GuiNode<T>{elem});
00084 }
00085
00086 template<typename T>
00087 void GuiQueue<T>::pop_back() {
00088     Base::pop_back();
00089 }
00090
00091 template<typename T>
00092 void GuiQueue<T>::render_link(Vector2 src, Vector2 dest) {
00093     constexpr int radius = GuiNode<T>::radius;
00094     constexpr float scaled_len = radius / 8.0F;
00095
00096     // straight line
00097     Vector2 link_pos{src.x + radius, src.y - scaled_len};
00098     Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};
00099
00100     // arrow
00101     constexpr int arrow_size = scaled_len * 5;
00102     Vector2 head{dest.x - radius + scaled_len / 2, src.y};
00103     Vector2 side_top{head.x - arrow_size, head.y - arrow_size};
00104     Vector2 side_bot{head.x - arrow_size, head.y + arrow_size};
00105
00106     // draw both
00107     DrawRectangleV(link_pos, link_size, Settings::get_instance().get_color(1));
00108     DrawTriangle(head, side_top, side_bot,
00109                  Settings::get_instance().get_color(1));
00110 }
00111
00112 template<typename T>
00113 void GuiQueue<T>::render() {
00114     update();
00115
00116     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00117         if (ptr->next != nullptr) {
00118             render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
00119         }
00120
00121         ptr->data.render();
00122     }
00123 }
00124
00125 template<typename T>
00126 void GuiQueue<T>::update() {
00127     // TODO: if not outdated then return
00128
00129     std::size_t pos = 0;
00130
00131     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00132         ptr->data.set_pos(
00133             {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
00134         ++pos;
00135     }
00136 }
00137
00138 }  // namespace gui
00139
00140 #endif  // GUI_QUEUE_GUI_HPP_
```

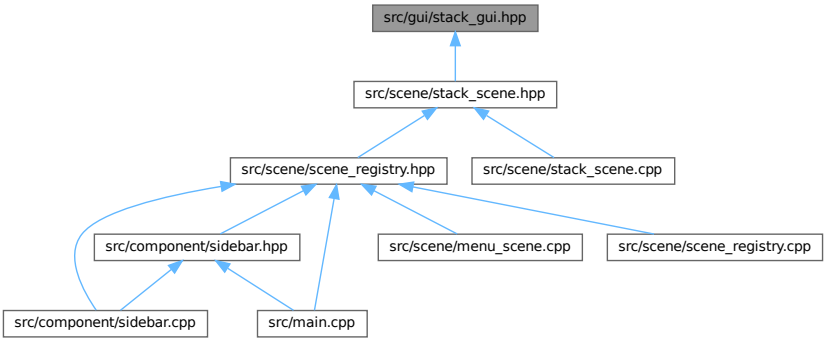
7.65 src/gui/stack_gui.hpp File Reference

```
#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:



Classes

- class gui::GuiStack< T >

Namespaces

- namespace gui

7.66 stack_gui.hpp

Go to the documentation of this file.

```
00001 #ifndef GUI_STACK_GUI_HPP_
00002 #define GUI_STACK_GUI_HPP_
00003
00004 #include <cstddef>
00005 #include <iostream>
00006
00007 #include "base_gui.hpp"
```

```
00008 #include "constants.hpp"
00009 #include "core/stack.hpp"
00010 #include "node_gui.hpp"
00011 #include "raylib.h"
00012 #include "settings.hpp"
00013
00014 namespace gui {
00015
00016 template<typename T>
00017 class GuiStack : public core::Stack<GuiNode<T>, public internal::Base {
00018 private:
00019     using Base = core::Stack<GuiNode<T>>;
00020
00021     static constexpr Vector2 head_pos{
00022         constants::scene_width / 2.0F - GuiNode<T>::radius / 2.0F,
00023         GuiNode<T>::radius * 4.0F};
00024
00025     using Base::m_head;
00026     using Base::m_tail;
00027
00028     void render_link(Vector2 src, Vector2 dest) override;
00029
00030 public:
00031     using Base::Base;
00032
00033     using Base::empty;
00034     using Base::size;
00035
00036     GuiStack(std::initializer_list<GuiNode<T>> init_list);
00037
00038     void push(const T& elem);
00039     void pop();
00040
00041     void update() override;
00042     void render() override;
00043     void init_label();
00044 };
00045
00046 template<typename T>
00047 void GuiStack<T>::init_label() {
00048     if (m_head != nullptr) {
00049         m_head->data.set_label("head");
00050     }
00051 }
00052
00053 template<typename T>
00054 GuiStack<T>::GuiStack(std::initializer_list<GuiNode<T>> init_list)
00055     : core::Stack<GuiNode<T>(init_list) {
00056     init_label();
00057 }
00058
00059 template<typename T>
00060 void GuiStack<T>::push(const T& elem) {
00061     Base::push(GuiNode<T>{elem});
00062 }
00063
00064 template<typename T>
00065 void GuiStack<T>::pop() {
00066     Base::pop();
00067 }
00068
00069 template<typename T>
00070 void GuiStack<T>::render_link(Vector2 src, Vector2 dest) {
00071     constexpr int radius = GuiNode<T>::radius;
00072     constexpr float scaled_len = radius / 8.0F;
00073
00074     // straight line
00075     Vector2 link_pos{src.x - scaled_len, src.y + radius};
00076     Vector2 link_size{2 * scaled_len, dest.y - src.y - 2 * radius};
00077
00078     // arrow
00079     constexpr int arrow_size = scaled_len * 5;
00080     Vector2 head{src.x, dest.y - radius + scaled_len / 2};
00081     Vector2 side_left{head.x - arrow_size, head.y - arrow_size};
00082     Vector2 side_right{head.x + arrow_size, head.y - arrow_size};
00083
00084     // draw both
00085     DrawRectangleV(link_pos, link_size, Settings::get_instance().get_color(1));
00086     DrawTriangle(head, side_right, side_left,
00087                  Settings::get_instance().get_color(1));
00088 }
00089
00090 template<typename T>
00091 void GuiStack<T>::render() {
00092     update();
00093
00094     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
```

```
00095         if (ptr->next != nullptr) {
00096             render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
00097         }
00098
00099         ptr->data.render();
00100     }
00101 }
00102
00103 template<typename T>
00104 void GuiStack<T>::update() {
00105     // TODO: if not outdated then return
00106
00107     std::size_t pos = 0;
00108
00109     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00110         ptr->data.set_pos(
00111             {head_pos.x, head_pos.y + 4 * GuiNode<T>::radius * pos});
00112         ++pos;
00113     }
00114 }
00115
00116 }  // namespace gui
00117
00118 #endif  // GUI_STACK_GUI_HPP_
```

7.67 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 ()

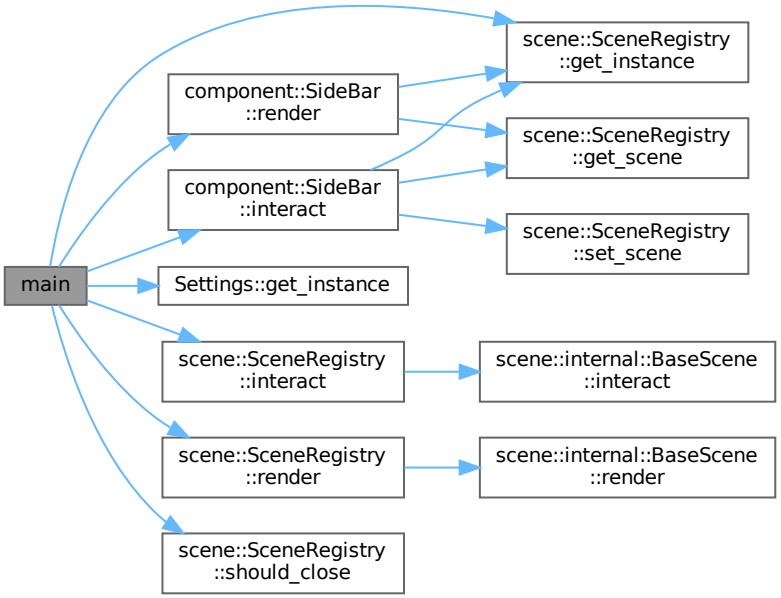
7.67.1 Function Documentation

7.67.1.1 main()

```
int main ( )
```

Definition at line 9 of file main.cpp.

Here is the call graph for this function:



7.68 main.cpp

Go to the documentation of this file.

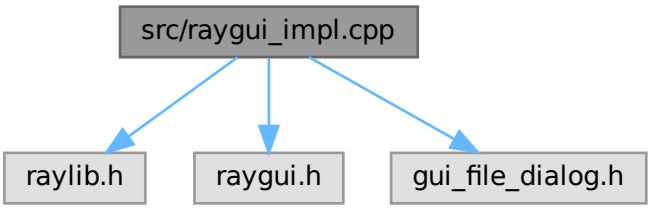
```
00001 #include <iostream>
00002
00003 #include "component/sidebar.hpp"
00004 #include "constants.hpp"
00005 #include "raygui.h"
00006 #include "scene/scene_registry.hpp"
00007 #include "settings.hpp"
00008
00009 int main() {
00010     InitWindow(constants::scene_width, constants::scene_height,
00011                "VisuAlgo.net clone in C++ by @jalsol");
00012     SetTargetFPS(constants::frames_per_second);
00013
00014     GuiLoadStyle("data/bluish_open_sans.rgs");
00015
00016     scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();
00017     component::SideBar sidebar;
00018
00019     bool should_close = false;
00020
00021     do {
00022         // NOTE: The order is important
00023         sidebar.interact();
00024         registry.interact();
00025
00026         BeginDrawing();
00027         {
00028             ClearBackground(
00029                 Settings::get_instance().get_color(Settings::num_color - 1));
00030
00031             // NOTE: The order is important
00032             registry.render();
00033             sidebar.render();
00034         }
00035         EndDrawing();
00036
00037         should_close = registry.should_close() || WindowShouldClose();
00038     } while (!should_close);
```

```
00039
00040     CloseWindow();
00041
00042     return 0;
00043 }
```

7.69 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

7.69.1 Macro Definition Documentation

7.69.1.1 GUI_FILE_DIALOG_IMPLEMENTATION

```
#define GUI_FILE_DIALOG_IMPLEMENTATION
```

Definition at line 6 of file raygui_impl.cpp.

7.69.1.2 RAYGUI_IMPLEMENTATION

```
#define RAYGUI_IMPLEMENTATION
```

Definition at line 2 of file raygui_impl.cpp.

7.70 raygui_impl.cpp

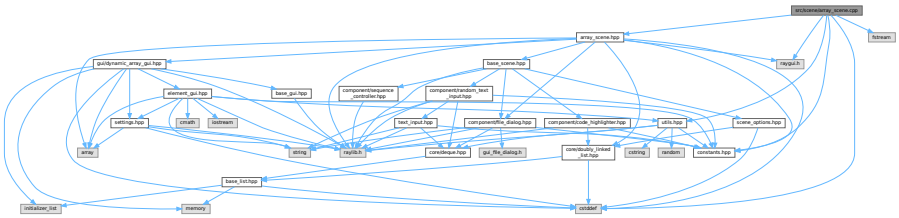
Go to the documentation of this file.

```
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"
```

7.71 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

7.72 array_scene.cpp

Go to the documentation of this file.

```
00001 #include "array_scene.hpp"
00002
00003 #include <cstddef>
00004 // #include <cstdlib>
00005 // #include <cstring>
00006 #include <fstream>
00007 // #include <iostream>
00008 // #include <limits>
00009 // #include <string>
00010
00011 #include "constants.hpp"
00012 #include "raygui.h"
00013 #include "utils.hpp"
00014
00015 namespace scene {
00016
00017 ArrayScene::ArrayScene() { m_array.reserve(scene_options.max_size); }
00018
00019 void ArrayScene::render_inputs() {
00020     int& mode = scene_options.mode_selection;
00021
00022     switch (mode) {
00023         case 0: {
```

```
00024             switch (scene_options.action_selection.at(mode)) {
00025                 case 0:
00026                     break;
00027                 case 1: {
00028                     m_text_input.render_head(options_head, head_offset);
00029                 } break;
00030                 case 2: {
00031                     m_go = (m_file_dialog.render_head(options_head,
00032                                                       head_offset) > 0);
00033                     return;
00034                 } break;
00035                 default:
00036                     utils::unreachable();
00037             }
00038         } break;
00039 
00040         case 1: {
00041             m_index_input.render_head(options_head, head_offset);
00042         } break;
00043 
00044         case 2: {
00045             m_index_input.render_head(options_head, head_offset);
00046             m_text_input.render_head(options_head, head_offset);
00047         } break;
00048 
00049         case 3: {
00050             m_text_input.render_head(options_head, head_offset);
00051         } break;
00052 
00053         case 4: {
00054             m_index_input.render_head(options_head, head_offset);
00055             m_text_input.render_head(options_head, head_offset);
00056         } break;
00057 
00058         case 5: {
00059             m_index_input.render_head(options_head, head_offset);
00060         } break;
00061 
00062         default:
00063             utils::unreachable();
00064     }
00065 
00066     m_go |= render_go_button();
00067 }
00068 
00069 void ArrayScene::render() {
00070     m_sequence_controller.inc_anim_counter();
00071 
00072     int frame_idx = m_sequence_controller.get_anim_frame();
00073     auto* const frame_ptr = m_sequence.find(frame_idx);
00074     m_sequence_controller.set_progress_value(frame_idx);
00075 
00076     if (frame_ptr != nullptr) {
00077         frame_ptr->data.render();
00078         m_code_highlighter.highlight(frame_idx);
00079     } else {  // end of sequence
00080         m_array.render();
00081         m_sequence_controller.set_run_all(false);
00082     }
00083 
00084     m_code_highlighter.render();
00085     m_sequence_controller.render();
00086     render_options(scene_options);
00087 }
00088 
00089 void ArrayScene::interact() {
00090     if (m_sequence_controller.interact()) {
00091         m_sequence_controller.reset_anim_counter();
00092         return;
00093     }
00094 
00095     m_index_input.set_random_max((int)m_array.size() - 1);
00096 
00097     if (m_text_input.interact() || m_index_input.interact()) {
00098         return;
00099     }
00100 
00101     if (!m_go) {
00102         return;
00103     }
00104 
00105     int& mode = scene_options.mode_selection;
00106 
00107     switch (mode) {
00108         case 0: {
00109             switch (scene_options.action_selection.at(mode)) {
00110                 case 0: {
```

```
00111                     interact_random();
00112                 } break;
00113
00114                 case 1: {
00115                     interact_import(m_text_input.extract_values());
00116                 } break;
00117
00118                 case 2: {
00119                     interact_file_import();
00120                 } break;
00121
00122                 default:
00123                     utils::unreachable();
00124             }
00125         } break;
00126
00127         case 1: {
00128             interact_access();
00129         } break;
00130
00131         case 2: {
00132             interact_update();
00133         } break;
00134
00135         case 3: {
00136             interact_search();
00137         } break;
00138
00139         case 4: {
00140             interact_insert();
00141         } break;
00142
00143         case 5: {
00144             interact_delete();
00145         } break;
00146
00147         default:
00148             utils::unreachable();
00149     }
00150
00151     m_go = false;
00152 }
00153
00154 void ArrayScene::interact_access() {
00155     auto index_container = m_index_input.extract_values();
00156     if (index_container.empty()) {
00157         return;
00158     }
00159
00160     std::size_t index = index_container.front();
00161     if (index >= m_array.size()) {
00162         return;
00163     }
00164
00165     m_code_highlighter.set_code({"return m_array[index];"});
00166
00167     m_sequence.clear();
00168
00169     m_array.set_color_index(index, 3);
00170     m_sequence.insert(m_sequence.size(), m_array);
00171     m_code_highlighter.push_into_sequence(0);
00172
00173     m_array.set_color_index(index, 0);
00174
00175     m_sequence_controller.set_max_value((int)m_sequence.size());
00176     m_sequence_controller.set_rerun();
00177 }
00178
00179 void ArrayScene::interact_random() {
00180     std::size_t size =
00181         utils::get_random(std::size_t{1}, scene_options.max_size);
00182     m_array = {};
00183
00184     for (std::size_t i = 0; i < size; ++i) {
00185         m_array.push(utils::get_random(constants::min_val, constants::max_val));
00186     }
00187
00188     m_array.reserve(max_size);
00189 }
00190
00191 void ArrayScene::interact_import(core::Deque<int> nums) {
00192     m_array = {};
00193     std::size_t i;  // NOLINT
00194
00195     for (i = 0; i < max_size && !nums.empty(); ++i) {
00196         m_array.push(nums.front());
00197         nums.pop_front();
```

```
00198     }
00199
00200     m_array.reserve(max_size);
00201 }
00202
00203 void ArrayScene::interact_update() {
00204     auto index_container = m_index_input.extract_values();
00205     if (index_container.empty()) {
00206         return;
00207     }
00208
00209     auto value_container = m_text_input.extract_values();
00210     if (value_container.empty()) {
00211         return;
00212     }
00213
00214     int index = index_container.front();
00215     int value = value_container.front();
00216
00217     if (!(0 <= index && index < m_array.size()) ||
00218         !utils::val_in_range(value)) {
00219         return;
00220     }
00221
00222     m_code_highlighter.set_code({
00223         "array[index] = value;",
00224     });
00225
00226     m_sequence.clear();
00227
00228     // initial state (before update)
00229     m_sequence.insert(m_sequence.size(), m_array);
00230     m_code_highlighter.push_into_sequence(-1);
00231
00232     // highlight
00233     m_array.set_color_index(index, 2);
00234     m_sequence.insert(m_sequence.size(), m_array);
00235     m_code_highlighter.push_into_sequence(0);
00236
00237     // update
00238     m_array[index] = value;
00239     m_array.set_color_index(index, 3);
00240     m_sequence.insert(m_sequence.size(), m_array);
00241     m_code_highlighter.push_into_sequence(0);
00242
00243     // undo highlight
00244     m_array.set_color_index(index, 0);
00245
00246     m_sequence_controller.set_max_value((int)m_sequence.size());
00247     m_sequence_controller.set_rerun();
00248 }
00249
00250 void ArrayScene::interact_file_import() {
00251     interact_import(m_file_dialog.extract_values());
00252 }
00253
00254 void ArrayScene::interact_search() {
00255     auto value_container = m_text_input.extract_values();
00256     if (value_container.empty()) {
00257         return;
00258     }
00259
00260     int value = value_container.front();
00261     if (!utils::val_in_range(value)) {
00262         return;
00263     }
00264
00265     m_code_highlighter.set_code({
00266         "for (i = 0; i < size; i++)",
00267         "    if (array[i] == value)",
00268         "        return i;",
00269         "return not_found",
00270     });
00271
00272     m_sequence.clear();
00273     m_sequence.insert(m_sequence.size(), m_array);
00274     m_code_highlighter.push_into_sequence(0);
00275
00276     bool found = false;
00277
00278     for (std::size_t i = 0; i < m_array.size(); ++i) {
00279         m_array.set_color_index(i, 3);
00280         m_sequence.insert(m_sequence.size(), m_array);
00281         m_code_highlighter.push_into_sequence(1);
00282
00283         if (m_array[i] == value) {
00284             found = true;
```

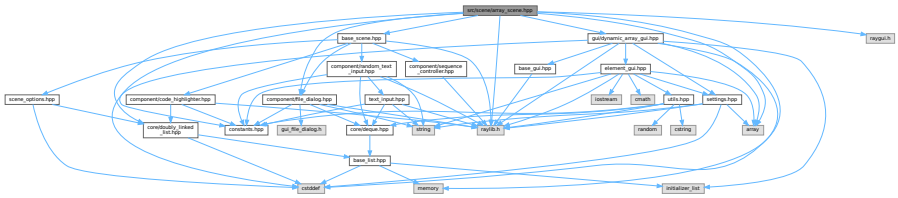
```
00285             m_array.set_color_index(i, 4);
00286             m_sequence.insert(m_sequence.size(), m_array);
00287             m_code_highlighter.push_into_sequence(2);
00288             m_array.set_color_index(i, 0);
00289             break;
00290         }
00291
00292         m_array.set_color_index(i, 0);
00293         m_sequence.insert(m_sequence.size(), m_array);
00294         m_code_highlighter.push_into_sequence(0);
00295     }
00296
00297     if (!found) {
00298         m_sequence.insert(m_sequence.size(), m_array);
00299         m_code_highlighter.push_into_sequence(3);
00300     }
00301
00302     m_sequence_controller.set_max_value((int)m_sequence.size());
00303     m_sequence_controller.set_rerun();
00304 }
00305
00306 void ArrayScene::interact_insert() {
00307     auto index_container = m_index_input.extract_values();
00308     if (index_container.empty()) {
00309         return;
00310     }
00311
00312     auto value_container = m_text_input.extract_values();
00313     if (value_container.empty()) {
00314         return;
00315     }
00316
00317     int index = index_container.front();
00318     int value = value_container.front();
00319
00320     if (m_array.size() >= max_size) {
00321         return;
00322     }
00323
00324     if (!(0 <= index && index <= m_array.size()) ||
00325         !utils::val_in_range(value)) {
00326         return;
00327     }
00328
00329     m_code_highlighter.set_code({
00330         "size++;",
00331         "for (i = size - 1; i > index; i--)",
00332         "    array[i] = array[i - 1];",
00333         "array[index] = value;",
00334     });
00335
00336     m_sequence.clear();
00337     m_sequence.insert(m_sequence.size(), m_array);
00338     m_code_highlighter.push_into_sequence(-1);
00339
00340     m_array.push(0);
00341     m_sequence.insert(m_sequence.size(), m_array);
00342     m_code_highlighter.push_into_sequence(0);
00343
00344     for (std::size_t i = m_array.size() - 1; i > index; --i) {
00345         m_array.set_color_index(i - 1, 2);
00346         m_sequence.insert(m_sequence.size(), m_array);
00347         m_code_highlighter.push_into_sequence(1);
00348
00349         m_array[i] = m_array[i - 1];
00350         m_array.set_color_index(i, 3);
00351         m_sequence.insert(m_sequence.size(), m_array);
00352         m_code_highlighter.push_into_sequence(2);
00353
00354         m_array.set_color_index(i - 1, 0);
00355         m_array.set_color_index(i, 0);
00356         m_sequence.insert(m_sequence.size(), m_array);
00357         m_code_highlighter.push_into_sequence(1);
00358     }
00359
00360     m_array.set_color_index(index, 2);
00361     m_sequence.insert(m_sequence.size(), m_array);
00362     m_code_highlighter.push_into_sequence(3);
00363
00364     m_array[index] = value;
00365     m_array.set_color_index(index, 3);
00366     m_sequence.insert(m_sequence.size(), m_array);
00367     m_code_highlighter.push_into_sequence(3);
00368
00369     m_array.set_color_index(index, 0);
00370     m_sequence.insert(m_sequence.size(), m_array);
00371     m_code_highlighter.push_into_sequence(-1);
```

```
00372
00373     m_sequence_controller.set_max_value((int)m_sequence.size());
00374     m_sequence_controller.set_rerun();
00375 }
00376
00377 void ArrayScene::interact_delete() {
00378     auto index_container = m_index_input.extract_values();
00379     if (index_container.empty()) {
00380         return;
00381     }
00382
00383     int index = index_container.front();
00384
00385     if (m_array.size() == 0) {
00386         return;
00387     }
00388
00389     if (!(0 <= index && index < m_array.size())) {
00390         return;
00391     }
00392
00393     m_code_highlighter.set_code({
00394         "for (i = index; i < size - 1; i++)",
00395         "    array[i] = array[i + 1];",
00396         "size--;",
00397     });
00398
00399     m_sequence.clear();
00400     m_sequence.insert(m_sequence.size(), m_array);
00401     m_code_highlighter.push_into_sequence(-1);
00402
00403     m_sequence.insert(m_sequence.size(), m_array);
00404     m_code_highlighter.push_into_sequence(0);
00405
00406     for (std::size_t i = index; i < m_array.size() - 1; ++i) {
00407         m_array.set_color_index(i, 3);
00408         m_sequence.insert(m_sequence.size(), m_array);
00409         m_code_highlighter.push_into_sequence(1);
00410
00411         m_array[i] = m_array[i + 1];
00412         m_array.set_color_index(i + 1, 2);
00413         m_sequence.insert(m_sequence.size(), m_array);
00414         m_code_highlighter.push_into_sequence(1);
00415
00416         m_array.set_color_index(i, 0);
00417         m_array.set_color_index(i + 1, 0);
00418         m_sequence.insert(m_sequence.size(), m_array);
00419         m_code_highlighter.push_into_sequence(0);
00420     }
00421
00422     m_array.set_color_index(m_array.size() - 1, 2);
00423     m_sequence.insert(m_sequence.size(), m_array);
00424     m_code_highlighter.push_into_sequence(2);
00425
00426     m_array.pop();
00427     m_sequence.insert(m_sequence.size(), m_array);
00428     m_code_highlighter.push_into_sequence(-1);
00429
00430     m_sequence_controller.set_max_value((int)m_sequence.size());
00431     m_sequence_controller.set_rerun();
00432 }
00433
00434 }  // namespace scene
```

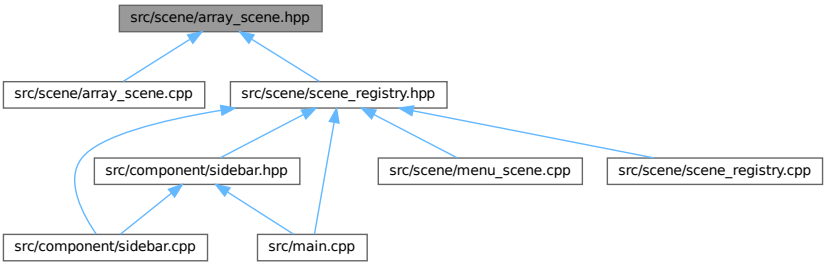
7.73 src/scene/array_scene.hpp File Reference

```
#include <array>
#include <cstddef>
#include "base_scene.hpp"
#include "component/file_dialog.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 array_scene.hpp:



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



Classes

- class scene::ArrayScene

Namespaces

- namespace scene

7.74 array_scene.hpp

Go to the documentation of this file.

```
00001 #ifndef SCENE_ARRAY_SCENE_HPP_
00002 #define SCENE_ARRAY_SCENE_HPP_
00003
00004 #include <array>
00005 #include <cstddef>
00006
00007 #include "base_scene.hpp"
00008 #include "component/file_dialog.hpp"
00009 #include "constants.hpp"
00010 #include "core/doubly_linked_list.hpp"
00011 #include "gui/dynamic_array_gui.hpp"
00012 #include "raygui.h"
00013 #include "raylib.h"
00014
00015 namespace scene {
00016
00017 class ArrayScene : public internal::BaseScene {
00018 private:
00019     static constexpr std::size_t max_size = 8;
00020
00021     internal::SceneOptions scene_options{
00022         // max_size
```

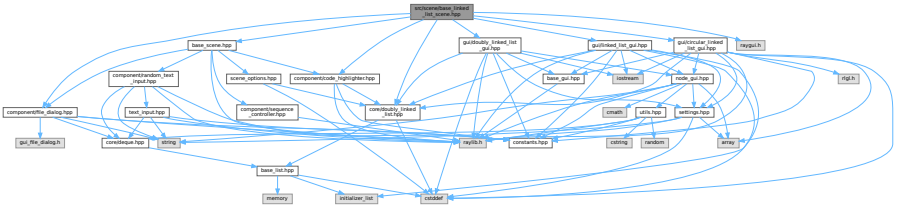
```
00023         max_size,
00024
00025         // mode_labels
00026         "Mode: Create;"
00027         "Mode: Access;"
00028         "Mode: Update;"
00029         "Mode: Search;"
00030         "Mode: Insert;"
00031         "Mode: Delete",
00032
00033         // mode_selection
00034         0,
00035
00036         // action_labels
00037         {
00038             // Mode: Create
00039             "Action: Random;Action: Input;Action: File",
00040
00041             // Mode: Access
00042             "",
00043
00044             // Mode: Update
00045             "",
00046
00047             // Mode: Search
00048             "",
00049
00050             // Mode: Insert
00051             "",
00052
00053             // Mode: Delete
00054             "",
00055         },
00056
00057         // action_selection
00058         core::DoublyLinkedList<int>{0, 0, 0, 0, 0},
00059     };
00060
00061     using internal::BaseScene::button_size;
00062     using internal::BaseScene::head_offset;
00063     using internal::BaseScene::options_head;
00064
00065     gui::GuiDynamicArray<int> m_array{};
00066     core::DoublyLinkedList<gui::GuiDynamicArray<int>> m_sequence;
00067
00068     bool m_go{};
00069     using internal::BaseScene::m_file_dialog;
00070     using internal::BaseScene::m_index_input;
00071     using internal::BaseScene::m_sequence_controller;
00072     using internal::BaseScene::m_text_input;
00073
00074     using internal::BaseScene::render_go_button;
00075     using internal::BaseScene::render_options;
00076     void render_inputs() override;
00077
00078     void interact_random();
00079     void interact_import(core::Deque<int> nums);
00080     void interact_file_import();
00081     void interact_update();
00082     void interact_search();
00083     void interact_insert();
00084     void interact_delete();
00085     void interact_access();
00086
00087 public:
00088     ArrayScene();
00089     void render() override;
00090     void interact() override;
00091 };
00092
00093 }  // namespace scene
00094
00095 #endif  // SCENE_ARRAY_SCENE_HPP_
```

7.75 src/scene/base_linked_list_scene.hpp File Reference

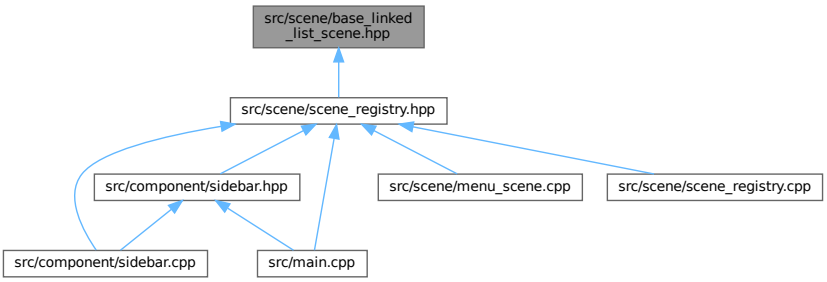
```
#include "base_scene.hpp"
#include "component/code_highlighter.hpp"
#include "component/file_dialog.hpp"
#include "core/doubly_linked_list.hpp"
```

```
#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:



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 > >

7.76 base_linked_list_scene.hpp

Go to the documentation of this file.

```
00001 #ifndef SCENE_BASE_LINKED_LIST_SCENE_HPP_
00002 #define SCENE_BASE_LINKED_LIST_SCENE_HPP_
00003
00004 #include "base_scene.hpp"
00005 #include "component/code_highlighter.hpp"
00006 #include "component/file_dialog.hpp"
00007 #include "core/doubly_linked_list.hpp"
00008 #include "gui/circular_linked_list_gui.hpp"
00009 #include "gui/doubly_linked_list_gui.hpp"
00010 #include "gui/linked_list_gui.hpp"
00011 #include "raygui.h"
00012
00013 namespace scene {
00014
00015 template<typename Con>
00016 class BaseLinkedListScene : public internal::BaseScene {
00017 private:
00018     internal::SceneOptions scene_options{
00019         // max_size
00020         8,  // NOLINT
00021
00022         // mode_labels
00023         "Mode: Create;"
00024         "Mode: Add;"
00025         "Mode: Delete;"
00026         "Mode: Update;"
00027         "Mode: Search",
00028
00029         // mode_selection
00030         0,
00031
00032         // action_labels
00033         {
00034             // Mode: Create
00035             "Action: Random;Action: Input;Action: File",
00036             // Mode: Add
00037             "",
00038             // Mode: Delete
00039             "",
00040             // Mode: Update
00041             "",
00042             // Mode: Search
00043             "",
00044         },
00045
00046         // action_selection
00047         core::DoublyLinkedList<int>{0, 0, 0, 0, 0},
00048     };
00049
00050     using internal::BaseScene::button_size;
00051     using internal::BaseScene::head_offset;
00052     using internal::BaseScene::options_head;
00053
00054     Con m_list{
00055         gui::GuiNode<int>{1},
00056         gui::GuiNode<int>{2},
00057         gui::GuiNode<int>{3},
00058     };
00059     core::DoublyLinkedList<Con> m_sequence;
00060
00061     bool m_go{};
00062     using internal::BaseScene::m_code_highlighter;
00063     using internal::BaseScene::m_file_dialog;
00064     using internal::BaseScene::m_index_input;
00065     using internal::BaseScene::m_sequence_controller;
00066     using internal::BaseScene::m_text_input;
00067
00068     using internal::BaseScene::render_go_button;
00069     using internal::BaseScene::render_options;
00070     void render_inputs() override;
00071
00072     void interact_random();
00073     void interact_import(core::Deque<int> nums);
00074     void interact_file_import();
00075
00076     void interact_add();
00077     void interact_add_head(int value);
00078     void interact_add_tail(int value);
00079     void interact_add_middle(int index, int value);
00080
00081     void interact_delete();
00082     void interact_delete_head();
```

```
00083     void interact_delete_tail();
00084     void interact_delete_middle(int index);
00085
00086     void interact_update();
00087     void interact_search();
00088
00089 public:
00090     void render() override;
00091     void interact() override;
00092 };
00093
00094 using LinkedListScene = BaseLinkedListScene<gui::GuiLinkedList<int>>;
00095 using DoublyLinkedListScene =
00096     BaseLinkedListScene<gui::GuiDoublyLinkedList<int>>;
00097 using CircularLinkedListScene =
00098     BaseLinkedListScene<gui::GuiCircularLinkedList<int>>;
00099
00100 template<typename Con>
00101 void BaseLinkedListScene<Con>::render_inputs() {
00102     int& mode = scene_options.mode_selection;
00103
00104     switch (mode) {
00105         case 0: {
00106             switch (scene_options.action_selection.at(mode)) {
00107                 case 0:
00108                     break;
00109                 case 1: {
00110                     m_text_input.render_head(options_head, head_offset);
00111                 } break;
00112                 case 2: {
00113                     m_go = (m_file_dialog.render_head(options_head,
00114                                                       head_offset) > 0);
00115                     return;
00116                 } break;
00117                 default:
00118                     utils::unreachable();
00119             }
00120         } break;
00121
00122         case 1: {
00123             m_index_input.render_head(options_head, head_offset);
00124             m_text_input.render_head(options_head, head_offset);
00125         } break;
00126
00127         case 2: {
00128             m_index_input.render_head(options_head, head_offset);
00129         } break;
00130
00131         case 3: {
00132             m_index_input.render_head(options_head, head_offset);
00133             m_text_input.render_head(options_head, head_offset);
00134         } break;
00135
00136         case 4: {
00137             m_text_input.render_head(options_head, head_offset);
00138         } break;
00139
00140         default:
00141             utils::unreachable();
00142     }
00143
00144     m_go |= render_go_button();
00145 }
00146
00147 template<typename Con>
00148 void BaseLinkedListScene<Con>::render() {
00149     m_sequence_controller.inc_anim_counter();
00150
00151     int frame_idx = m_sequence_controller.get_anim_frame();
00152     auto* const frame_ptr = m_sequence.find(frame_idx);
00153     m_sequence_controller.set_progress_value(frame_idx);
00154
00155     if (frame_ptr != nullptr) {
00156         frame_ptr->data.render();
00157         m_code_highlighter.highlight(frame_idx);
00158     } else {  // end of sequence
00159         m_list.render();
00160         m_sequence_controller.set_run_all(false);
00161     }
00162
00163     m_code_highlighter.render();
00164     m_sequence_controller.render();
00165     render_options(scene_options);
00166 }
00167
00168 template<typename Con>
00169 void BaseLinkedListScene<Con>::interact() {
```

```
00170     if (m_sequence_controller.interact()) {
00171         m_sequence_controller.reset_anim_counter();
00172         return;
00173     }
00174
00175     m_index_input.set_random_max((int)m_list.size() - 1);
00176
00177     if (m_text_input.interact() || m_index_input.interact()) {
00178         return;
00179     }
00180
00181     if (!m_go) {
00182         return;
00183     }
00184
00185     int& mode = scene_options.mode_selection;
00186
00187     switch (mode) {
00188         case 0: {
00189             switch (scene_options.action_selection.at(mode)) {
00190                 case 0: {
00191                     interact_random();
00192                 } break;
00193
00194                 case 1: {
00195                     interact_import(m_text_input.extract_values());
00196                 } break;
00197
00198                 case 2: {
00199                     interact_file_import();
00200                 } break;
00201
00202                 default:
00203                     utils::unreachable();
00204             }
00205         } break;
00206
00207         case 1: {
00208             m_index_input.set_random_max((int)m_list.size());
00209             interact_add();
00210         } break;
00211
00212         case 2: {
00213             interact_delete();
00214         } break;
00215
00216         case 3: {
00217             interact_update();
00218         } break;
00219
00220         case 4: {
00221             interact_search();
00222         } break;
00223
00224         default:
00225             utils::unreachable();
00226     }
00227
00228     m_go = false;
00229 }
00230
00231 template<typename Con>
00232 void BaseLinkedListScene<Con>::interact_random() {
00233     std::size_t size =
00234         utils::get_random(std::size_t{1}, scene_options.max_size);
00235     m_list = Con();
00236
00237     for (auto i = 0; i < size; ++i) {
00238         m_list.insert(
00239             i, utils::get_random(constants::min_val, constants::max_val));
00240     }
00241     m_list.init_label();
00242 }
00243
00244 template<typename Con>
00245 void BaseLinkedListScene<Con>::interact_import(core::Deque<int> nums) {
00246     m_sequence.clear();
00247     m_list = Con();
00248
00249     while (!nums.empty()) {
00250         if (utils::val_in_range(nums.front())) {
00251             m_list.insert(m_list.size(), nums.front());
00252         }
00253         nums.pop_front();
00254     }
00255     m_list.init_label();
00256 }
```

```
00257
00258 template<typename Con>
00259 void BaseLinkedListScene<Con>::interact_file_import() {
00260     interact_import(m_file_dialog.extract_values());
00261 }
00262
00263 template<typename Con>
00264 void BaseLinkedListScene<Con>::interact_add() {
00265     auto index_container = m_index_input.extract_values();
00266     if (index_container.empty()) {
00267         return;
00268     }
00269
00270     auto value_container = m_text_input.extract_values();
00271     if (value_container.empty()) {
00272         return;
00273     }
00274
00275     int index = index_container.front();
00276     int value = value_container.front();
00277
00278     if (!(0 <= index && index <= m_list.size())) {
00279         return;
00280     }
00281
00282     if (!utils::val_in_range(value)) {
00283         return;
00284     }
00285
00286     m_sequence.clear();
00287     m_sequence.insert(m_sequence.size(), m_list);
00288
00289     if (index == 0) {
00290         interact_add_head(value);
00291     } else if (index == m_list.size()) {
00292         interact_add_tail(value);
00293     } else {
00294         interact_add_middle(index, value);
00295     }
00296
00297     m_sequence_controller.set_max_value((int)m_sequence.size());
00298     m_sequence_controller.set_rerun();
00299 }
00300
00301 template<typename Con>
00302 void BaseLinkedListScene<Con>::interact_add_head(int value) {
00303     m_code_highlighter.set_code({
00304         "Node* node = new Node(value);",
00305         "node->next = head;",
00306         "head = next;",
00307     });
00308     m_code_highlighter.push_into_sequence(-1);
00309
00310     m_list.insert(0, value);
00311
00312     m_list.at(0).set_color_index(6);
00313     m_list.at(0).set_label("node");
00314     m_sequence.insert(m_sequence.size(), m_list);
00315     m_code_highlighter.push_into_sequence(0);
00316
00317     if (m_list.size() > 1) {
00318         m_list.at(1).set_color_index(4);
00319     }
00320
00321     m_sequence.insert(m_sequence.size(), m_list);
00322     m_code_highlighter.push_into_sequence(1);
00323
00324     if (m_list.size() > 1) {
00325         m_list.at(1).set_color_index(0);
00326         m_list.at(1).set_label("");
00327     }
00328
00329     m_list.at(0).set_color_index(4);
00330     m_list.at(0).set_label("head");
00331     m_sequence.insert(m_sequence.size(), m_list);
00332     m_code_highlighter.push_into_sequence(2);
00333
00334     m_list.at(0).set_color_index(0);
00335 }
00336
00337 template<typename Con>
00338 void BaseLinkedListScene<Con>::interact_add_tail(int value) {
00339     m_code_highlighter.set_code({
00340         "Node* node = new Node(value);",
00341         "tail->next = node;",
00342         "tail = tail->next;",
00343     });
```

```
00344     m_code_highlighter.push_into_sequence(-1);
00345
00346     std::size_t size = m_list.size();
00347
00348     m_list.insert(size, value);
00349     m_list.at(size).set_color_index(6);
00350     m_sequence.insert(m_sequence.size(), m_list);
00351     m_code_highlighter.push_into_sequence(0);
00352
00353     m_list.at(size - 1).set_color_index(4);
00354     m_sequence.insert(m_sequence.size(), m_list);
00355     m_code_highlighter.push_into_sequence(1);
00356
00357     m_list.at(size - 1).set_color_index(0);
00358     m_list.at(size - 1).set_label("");
00359     m_list.at(size).set_color_index(4);
00360     m_list.at(size).set_label("tail");
00361     m_sequence.insert(m_sequence.size(), m_list);
00362     m_code_highlighter.push_into_sequence(2);
00363
00364     m_list.at(size).set_color_index(0);
00365 }
00366
00367 template<typename Con>
00368 void BaseLinkedListScene<Con>::interact_add_middle(int index, int value) {
00369     m_code_highlighter.set_code({
00370         "Node* pre = head;",
00371         "for (i = 0; i < index - 1; ++i)",
00372         "    pre = pre->next;",
00373         "",
00374         "Node* nxt = pre->next;",
00375         "Node* node = new Node(value);",
00376         "node->next = nxt;",
00377         "pre->next = node;",
00378     });
00379     m_code_highlighter.push_into_sequence(-1);
00380
00381     m_list.at(0).set_color_index(4);
00382     m_list.at(0).set_label("head/pre");
00383     m_sequence.insert(m_sequence.size(), m_list);
00384     m_code_highlighter.push_into_sequence(0);
00385
00386     // search until index - 1
00387     for (int i = 0; i < index - 1; ++i) {
00388         m_list.at(i).set_color_index(2);
00389         m_sequence.insert(m_sequence.size(), m_list);
00390         m_code_highlighter.push_into_sequence(1);
00391
00392         m_list.at(i).set_color_index(0);
00393         m_list.at(i).set_label(i == 0 ? "head" : "");
00394         m_list.at(i + 1).set_color_index(2);
00395         m_list.at(i + 1).set_label("pre");
00396         m_sequence.insert(m_sequence.size(), m_list);
00397         m_code_highlighter.push_into_sequence(2);
00398     }
00399
00400     m_sequence.insert(m_sequence.size(), m_list);
00401     m_code_highlighter.push_into_sequence(1);
00402
00403     // reaching index - 1
00404     // cur
00405     m_list.at(index - 1).set_color_index(2);
00406     m_sequence.insert(m_sequence.size(), m_list);
00407     m_code_highlighter.push_into_sequence(3);
00408
00409     // cur->next
00410     m_list.at(index).set_color_index(7);
00411     m_list.at(index).set_label(index + 1 == m_list.size() ? "tail/nxt" : "nxt");
00412     m_sequence.insert(m_sequence.size(), m_list);
00413     m_code_highlighter.push_into_sequence(4);
00414
00415     // insert between cur and cur->next
00416     m_list.insert(index, value);
00417     m_list.at(index).set_color_index(6);
00418     m_list.at(index).set_label("node");
00419     m_sequence.insert(m_sequence.size(), m_list);
00420     m_code_highlighter.push_into_sequence(5);
00421
00422     m_list.at(index - 1).set_color_index(2);
00423     m_list.at(index + 1).set_color_index(0);
00424     m_sequence.insert(m_sequence.size(), m_list);
00425     m_code_highlighter.push_into_sequence(6);
00426
00427     m_list.at(index - 1).set_color_index(0);
00428     m_list.at(index + 1).set_color_index(7);
00429     m_list.init_label();
00430     m_sequence.insert(m_sequence.size(), m_list);
```

```
00431     m_code_highlighter.push_into_sequence(7);
00432
00433     // done
00434     m_list.at(index - 1).set_color_index(0);
00435     m_list.at(index - 1).set_label("");
00436     m_list.at(index).set_color_index(0);
00437     m_list.at(index).set_label("");
00438     m_list.at(index + 1).set_color_index(0);
00439     m_list.at(index + 1).set_label("");
00440     m_list.init_label();
00441 }
00442
00443 template<typename Con>
00444 void BaseLinkedListScene<Con>::interact_delete() {
00445     if (m_list.empty()) {
00446         return;
00447     }
00448
00449     auto index_container = m_index_input.extract_values();
00450     if (index_container.empty()) {
00451         return;
00452     }
00453
00454     int index = index_container.front();
00455
00456     if (!(0 <= index && index < m_list.size())) {
00457         return;
00458     }
00459
00460     m_sequence.clear();
00461     m_sequence.insert(m_sequence.size(), m_list);
00462
00463     if (index == 0) {
00464         interact_delete_head();
00465     } else if (index + 1 == m_list.size()) {
00466         interact_delete_tail();
00467     } else {
00468         interact_delete_middle(index);
00469     }
00470
00471     m_sequence_controller.set_max_value((int)m_sequence.size());
00472     m_sequence_controller.set_rerun();
00473 }
00474
00475 template<typename Con>
00476 void BaseLinkedListScene<Con>::interact_delete_head() {
00477     m_code_highlighter.set_code({
00478         "Node* temp = head;",
00479         "head = head->next;",
00480         "delete temp;",
00481     });
00482     m_code_highlighter.push_into_sequence(-1);
00483
00484     m_list.at(0).set_color_index(4);
00485     m_sequence.insert(m_sequence.size(), m_list);
00486     m_code_highlighter.push_into_sequence(0);
00487
00488     m_list.at(0).set_color_index(5);
00489     m_list.at(0).set_label("");
00490     if (m_list.size() > 1) {
00491         m_list.at(1).set_color_index(4);
00492         m_list.at(1).set_label("head");
00493     }
00494     m_sequence.insert(m_sequence.size(), m_list);
00495     m_code_highlighter.push_into_sequence(1);
00496
00497     m_list.remove(0);
00498     m_sequence.insert(m_sequence.size(), m_list);
00499     m_code_highlighter.push_into_sequence(2);
00500
00501     if (m_list.size() > 0) {
00502         m_list.at(0).set_color_index(0);
00503     }
00504 }
00505
00506 template<typename Con>
00507 void BaseLinkedListScene<Con>::interact_delete_tail() {
00508     m_code_highlighter.set_code({
00509         "Node* pre = head;",
00510         "Node* nxt = pre->next;",
00511         "while (nxt->next != nullptr)",
00512         "    pre = pre->next, nxt = nxt->next;",
00513         "",
00514         "delete nxt;",
00515         "tail = pre;",
00516     });
00517     m_code_highlighter.push_into_sequence(-1);
```

```
00518
00519     m_list.at(0).set_color_index(2);
00520     m_list.at(0).set_label("head/pre");
00521     m_sequence.insert(m_sequence.size(), m_list);
00522     m_code_highlighter.push_into_sequence(0);
00523
00524     m_list.at(1).set_color_index(3);
00525     if (m_list.size() == 2) {
00526         m_list.at(1).set_label("tail/nxt");
00527     } else {
00528         m_list.at(1).set_label("nxt");
00529     }
00530     m_sequence.insert(m_sequence.size(), m_list);
00531     m_code_highlighter.push_into_sequence(1);
00532
00533     int idx = 0;
00534     for (; idx + 2 < m_list.size(); ++idx) {
00535         m_sequence.insert(m_sequence.size(), m_list);
00536         m_code_highlighter.push_into_sequence(2);
00537
00538         m_list.at(idx).set_color_index(0);
00539         if (idx == 0) {
00540             m_list.at(idx).set_label("head");
00541         } else {
00542             m_list.at(idx).set_label("");
00543         }
00544
00545         m_list.at(idx + 1).set_color_index(2);
00546         m_list.at(idx + 1).set_label("pre");
00547         m_list.at(idx + 2).set_color_index(3);
00548         if (idx + 3 == m_list.size()) {
00549             m_list.at(idx + 2).set_label("tail/nxt");
00550         } else {
00551             m_list.at(idx + 2).set_label("nxt");
00552         }
00553
00554         m_sequence.insert(m_sequence.size(), m_list);
00555         m_code_highlighter.push_into_sequence(3);
00556     }
00557
00558     m_sequence.insert(m_sequence.size(), m_list);
00559     m_code_highlighter.push_into_sequence(2);
00560
00561     m_list.at(idx).set_color_index(2);
00562     m_list.at(idx).set_label("pre");
00563     m_list.at(idx + 1).set_color_index(3);
00564     m_list.at(idx + 1).set_label("tail/nxt");
00565     m_sequence.insert(m_sequence.size(), m_list);
00566     m_code_highlighter.push_into_sequence(4);
00567
00568     m_list.remove(idx + 1);
00569     m_list.at(idx).set_label("tail/pre");
00570     m_sequence.insert(m_sequence.size(), m_list);
00571     m_code_highlighter.push_into_sequence(5);
00572
00573     m_list.at(idx).set_color_index(4);
00574     m_list.init_label();
00575     m_sequence.insert(m_sequence.size(), m_list);
00576     m_code_highlighter.push_into_sequence(6);
00577
00578     m_list.at(idx).set_color_index(0);
00579 }
00580
00581 template<typename Con>
00582 void BaseLinkedListScene<Con>::interact_delete_middle(int index) {
00583     m_code_highlighter.set_code({
00584         "Node* pre = head;",
00585         "for (i = 0; i < index - 1; i++)",
00586         "    pre = pre->next;",
00587         "",
00588         "Node* node = pre->next;",
00589         "Node* nxt = node->next;",
00590         "delete node;",
00591         "pre->next = nxt;",
00592     });
00593     m_code_highlighter.push_into_sequence(-1);
00594
00595     m_list.at(0).set_color_index(4);
00596     m_list.at(0).set_label("head/pre");
00597     m_sequence.insert(m_sequence.size(), m_list);
00598     m_code_highlighter.push_into_sequence(0);
00599
00600     int idx = 0;
00601     for (; idx + 1 < index; ++idx) {
00602         m_list.at(idx).set_color_index(2);
00603         m_sequence.insert(m_sequence.size(), m_list);
00604         m_code_highlighter.push_into_sequence(1);
```

```
00605
00606         m_list.at(idx).set_color_index(0);
00607         m_list.at(idx).set_label("");
00608         m_list.at(idx + 1).set_color_index(2);
00609         m_list.init_label();
00610         m_list.at(idx + 1).set_label("pre");
00611         m_sequence.insert(m_sequence.size(), m_list);
00612         m_code_highlighter.push_into_sequence(2);
00613     }
00614
00615     m_list.at(idx).set_color_index(2);
00616     m_list.at(idx).set_label("pre");
00617     m_sequence.insert(m_sequence.size(), m_list);
00618     m_code_highlighter.push_into_sequence(3);
00619
00620     m_list.at(idx + 1).set_color_index(5);
00621     m_list.at(idx + 1).set_label("node");
00622     m_sequence.insert(m_sequence.size(), m_list);
00623     m_code_highlighter.push_into_sequence(4);
00624
00625     m_list.at(idx + 2).set_color_index(3);
00626     if (idx + 3 == m_list.size()) {
00627         m_list.at(idx + 2).set_label("tail/nxt");
00628     } else {
00629         m_list.at(idx + 2).set_label("nxt");
00630     }
00631     m_sequence.insert(m_sequence.size(), m_list);
00632     m_code_highlighter.push_into_sequence(5);
00633
00634     m_list.remove(idx + 1);
00635     m_sequence.insert(m_sequence.size(), m_list);
00636     m_code_highlighter.push_into_sequence(6);
00637
00638     m_list.at(idx + 1).set_color_index(7);
00639     m_sequence.insert(m_sequence.size(), m_list);
00640     m_code_highlighter.push_into_sequence(7);
00641
00642     m_list.at(idx).set_color_index(0);
00643     m_list.at(idx).set_label("");
00644     m_list.at(idx + 1).set_color_index(0);
00645     m_list.at(idx + 1).set_label("");
00646 }
00647
00648 template<typename Con>
00649 void BaseLinkedListScene<Con>::interact_update() {
00650     auto index_container = m_index_input.extract_values();
00651     if (index_container.empty()) {
00652         return;
00653     }
00654
00655     auto value_container = m_text_input.extract_values();
00656     if (value_container.empty()) {
00657         return;
00658     }
00659
00660     int index = index_container.front();
00661     int value = value_container.front();
00662
00663     if (!(0 <= index && index < m_list.size())) {
00664         return;
00665     }
00666
00667     m_code_highlighter.set_code({
00668         "Node* node = head;",
00669         "for (i = 0; i < index; i++)",
00670         "    node = node->next;",
00671         "",
00672         "node->value = value;",
00673     });
00674
00675     m_sequence.clear();
00676     m_sequence.insert(m_sequence.size(), m_list);
00677     m_code_highlighter.push_into_sequence(-1);
00678
00679     m_list.at(0).set_color_index(4);
00680     m_list.at(0).set_label("head/node");
00681     m_sequence.insert(m_sequence.size(), m_list);
00682     m_code_highlighter.push_into_sequence(0);
00683
00684     for (int i = 0; i < index; ++i) {
00685         m_list.at(i).set_color_index(2);
00686         m_sequence.insert(m_sequence.size(), m_list);
00687         m_code_highlighter.push_into_sequence(1);
00688
00689         m_list.at(i).set_color_index(0);
00690         m_list.at(i).set_label(i == 0 ? "head" : "");
00691         m_list.at(i + 1).set_color_index(2);
```

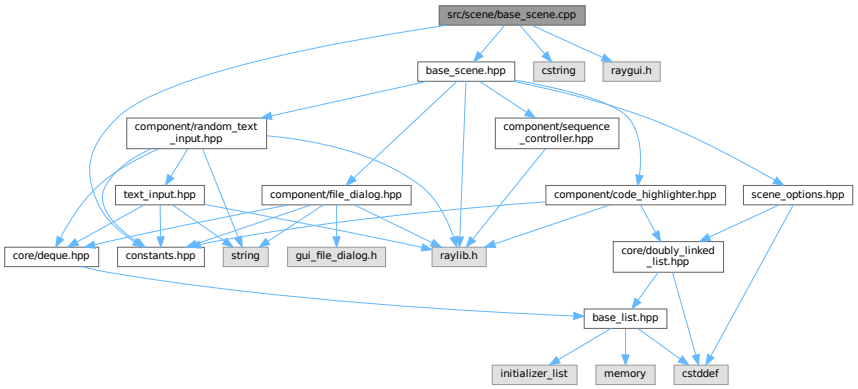
```
00692         m_list.at(i + 1).set_label(i + 2 == m_list.size() ? "tail/node"
00693                                                           : "node");
00694         m_sequence.insert(m_sequence.size(), m_list);
00695         m_code_highlighter.push_into_sequence(2);
00696     }
00697
00698     m_sequence.insert(m_sequence.size(), m_list);
00699     m_code_highlighter.push_into_sequence(1);
00700     m_sequence.insert(m_sequence.size(), m_list);
00701     m_code_highlighter.push_into_sequence(3);
00702
00703     m_list.at(index).set_color_index(3);
00704     m_list.at(index).set_value(value);
00705     m_sequence.insert(m_sequence.size(), m_list);
00706     m_code_highlighter.push_into_sequence(4);
00707
00708     m_list.at(index).set_color_index(0);
00709     m_list.at(index).set_label("");
00710     m_list.init_label();
00711
00712     m_sequence_controller.set_max_value((int)m_sequence.size());
00713     m_sequence_controller.set_rerun();
00714 }
00715
00716 template<typename Con>
00717 void BaseLinkedListScene<Con>::interact_search() {
00718     auto value_container = m_text_input.extract_values();
00719     if (value_container.empty()) {
00720         return;
00721     }
00722
00723     int value = value_container.front();
00724     if (!utils::val_in_range(value)) {
00725         return;
00726     }
00727
00728     m_code_highlighter.set_code({
00729         "Node* node = head;",
00730         "while (node != nullptr) {",
00731         "    if (node->value == value)",
00732         "        return node;",
00733         "    node = node->next;",
00734         "}",
00735         "return not_found",
00736     });
00737
00738     m_sequence.clear();
00739     m_sequence.insert(m_sequence.size(), m_list);
00740     m_code_highlighter.push_into_sequence(-1);
00741
00742     m_list.at(0).set_color_index(4);
00743     m_list.at(0).set_label("head/node");
00744     m_sequence.insert(m_sequence.size(), m_list);
00745     m_code_highlighter.push_into_sequence(0);
00746
00747     std::size_t idx = 0;
00748
00749     while (idx < m_list.size()) {
00750         m_list.at(idx).set_color_index(2);
00751         m_sequence.insert(m_sequence.size(), m_list);
00752         m_code_highlighter.push_into_sequence(1);
00753
00754         m_sequence.insert(m_sequence.size(), m_list);
00755         m_code_highlighter.push_into_sequence(2);
00756         if (m_list.at(idx).get_value() == value) {
00757             m_list.at(idx).set_color_index(3);
00758             m_sequence.insert(m_sequence.size(), m_list);
00759             m_code_highlighter.push_into_sequence(3);
00760             m_list.at(idx).set_color_index(0);
00761             m_list.at(idx).set_label(idx + 1 == m_list.size() ? "tail" : "");
00762             break;
00763         }
00764
00765         m_list.at(idx).set_color_index(0);
00766         m_list.at(idx).set_label("");
00767         m_list.init_label();
00768         ++idx;
00769         if (idx < m_list.size()) {
00770             m_list.at(idx).set_color_index(2);
00771             m_list.at(idx).set_label(idx + 1 == m_list.size() ? "tail/node"
00772                                                               : "node");
00773         }
00774         m_sequence.insert(m_sequence.size(), m_list);
00775         m_code_highlighter.push_into_sequence(4);
00776     }
00777
00778     if (idx >= m_list.size()) {
```

```
00779         m_sequence.insert(m_sequence.size(), m_list);
00780         m_code_highlighter.push_into_sequence(1);
00781
00782         m_sequence.insert(m_sequence.size(), m_list);
00783         m_code_highlighter.push_into_sequence(5);
00784
00785         m_sequence.insert(m_sequence.size(), m_list);
00786         m_code_highlighter.push_into_sequence(6);
00787     }
00788
00789     m_sequence_controller.set_max_value((int)m_sequence.size());
00790     m_sequence_controller.set_rerun();
00791 }
00792
00793 }  // namespace scene
00794
00795 #endif  // SCENE_BASE_LINKED_LIST_SCENE_HPP_
```

7.77 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

7.78 base_scene.cpp

Go to the documentation of this file.

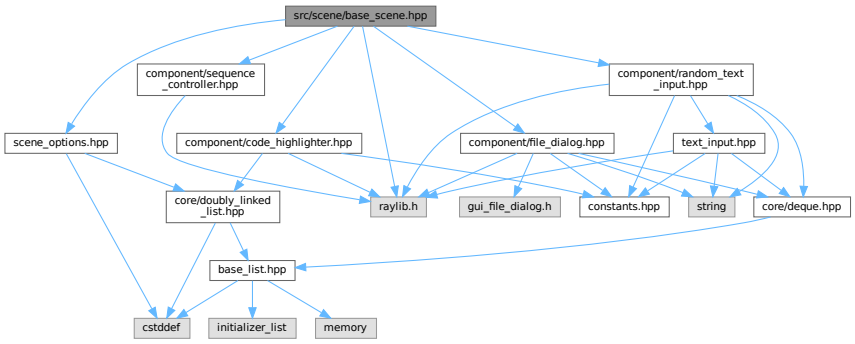
```
00001 #include "base_scene.hpp"
00002
00003 #include <cstring>
00004
00005 #include "constants.hpp"
00006 #include "raygui.h"
00007
00008 namespace scene::internal {
00009
00010 bool BaseScene::render_go_button() const {
```

```
00011     Rectangle shape{options_head, constants::scene_height - button_size.y,
00012                     button_size.y, button_size.y};
00013     return GuiButton(shape, "Go");
00014 }
00015
00016 void BaseScene::render_options(SceneOptions& scene_config) {
00017     (m_edit_mode || m_edit_action) ? GuiLock() : GuiUnlock();
00018
00019     options_head = 2 * constants::sidebar_width;
00020
00021     Rectangle mode_button_shape{options_head,
00022                                 constants::scene_height - button_size.y,
00023                                 button_size.x, button_size.y};
00024
00025     options_head += (button_size.x + head_offset);
00026
00027     int& mode = scene_config.mode_selection;
00028
00029     if (GuiDropupBox(mode_button_shape, scene_config.mode_labels, &mode,
00030                      m_edit_mode)) {
00031         m_edit_mode ^= 1;
00032     }
00033
00034     if (std::strlen(scene_config.action_labels.at(mode)) != 0) {
00035         Rectangle action_button_shape{options_head,
00036                                       constants::scene_height - button_size.y,
00037                                       button_size.x, button_size.y};
00038
00039         options_head += (button_size.x + head_offset);
00040
00041         int& action_selection = scene_config.action_selection.at(mode);
00042
00043         if (GuiDropupBox(action_button_shape,
00044                          scene_config.action_labels.at(mode), &action_selection,
00045                          m_edit_action)) {
00046             m_edit_action ^= 1;
00047         }
00048
00049         // scene_config.action_selection.at(mode) = GuiComboBox(
00050         //     action_button_shape, scene_config.action_labels.at(mode),
00051         //     scene_config.action_selection.at(mode));
00052     }
00053
00054     render_inputs();
00055 }
00056
00057 }  // namespace scene::internal
```

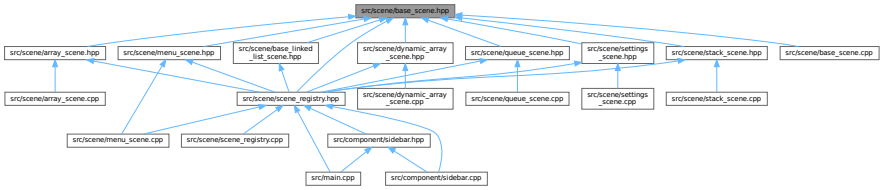
7.79 src/scene/base_scene.hpp File Reference

```
#include "component/code_highlighter.hpp"
#include "component/file_dialog.hpp"
#include "component/random_text_input.hpp"
#include "component/sequence_controller.hpp"
#include "raylib.h"
#include "scene_options.hpp"
```

Include dependency graph for base_scene.hpp:



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



Classes

- class scene::internal::BaseScene

Namespaces

- namespace scene
- namespace scene::internal

7.80 base_scene.hpp

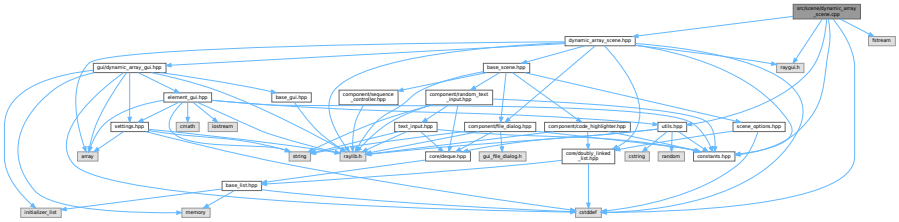
Go to the documentation of this file.

```
00001 #ifndef SCENE_BASE_SCENE_HPP_
00002 #define SCENE_BASE_SCENE_HPP_
00003
00004 #include "component/code_highlighter.hpp"
00005 #include "component/file_dialog.hpp"
00006 #include "component/random_text_input.hpp"
00007 #include "component/sequence_controller.hpp"
00008 #include "raylib.h"
00009 #include "scene_options.hpp"
00010
00011 namespace scene::internal {
00012
00013 class BaseScene {
00014 protected:
00015     static constexpr Vector2 button_size{200, 50};
00016     static constexpr int head_offset = 20;
00017     float options_head{};
00018
00019     virtual bool render_go_button() const;
00020     virtual void render_options(SceneOptions& scene_config);
00021     virtual void render_inputs() {}
00022
00023     component::RandomTextInput m_text_input{"value"};
00024     component::RandomTextInput m_index_input{"index"};
00025     component::FileDialog m_file_dialog;
00026     component::SequenceController m_sequence_controller;
00027     component::CodeHighlighter m_code_highlighter;
00028
00029     bool m_edit_mode{};
00030     bool m_edit_action{};
00031
00032 public:
00033     BaseScene() = default;
00034     BaseScene(const BaseScene&) = delete;
00035     BaseScene(BaseScene&&) = delete;
00036     BaseScene& operator=(const BaseScene&) = delete;
00037     BaseScene& operator=(BaseScene&&) = delete;
00038
00039     virtual ~BaseScene() = default;
00040
00041     virtual void render() {}
00042     virtual void interact() {}
00043 };
00044
00045 }  // namespace scene::internal
00046
00047 #endif  // SCENE_BASE_SCENE_HPP_
```

7.81 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

7.82 dynamic_array_scene.cpp

Go to the documentation of this file.

```
00001 #include "dynamic_array_scene.hpp"
00002
00003 #include <cstddef>
00004 // #include <cstdlib>
00005 // #include <cstring>
00006 #include <fstream>
00007 // #include <iostream>
00008 // #include <limits>
00009 // #include <string>
00010
00011 #include "constants.hpp"
00012 #include "raygui.h"
00013 #include "utils.hpp"
00014
00015 namespace scene {
00016
00017 void DynamicArrayScene::render_inputs() {
00018     int& mode = scene_options.mode_selection;
00019
00020     switch (mode) {
00021         case 0: {
00022             switch (scene_options.action_selection.at(mode)) {
00023                 case 0:
00024                     break;
00025                 case 1: {
00026                     m_text_input.render_head(options_head, head_offset);
00027                 } break;
00028                 case 2: {
00029                     m_go = (m_file_dialog.render_head(options_head,
00030                                                       head_offset) > 0);
00031                     return;
00032                 } break;
00033                 default:
00034                     utils::unreachable();
00035             }
00036         } break;
00037
00038         case 1: {
00039             m_index_input.render_head(options_head, head_offset);
00040         } break;
```

```
00041
00042         case 2: {
00043             switch (scene_options.action_selection.at(mode)) {
00044                 case 0: {
00045                     m_text_input.render_head(options_head, head_offset);
00046                 } break;
00047                 case 1:
00048                     break;
00049                 default:
00050                     utils::unreachable();
00051             }
00052         } break;
00053
00054         case 3: {
00055             m_index_input.render_head(options_head, head_offset);
00056             m_text_input.render_head(options_head, head_offset);
00057         } break;
00058
00059         case 4: {
00060             m_text_input.render_head(options_head, head_offset);
00061         } break;
00062
00063         case 5: {
00064             m_index_input.render_head(options_head, head_offset);
00065             m_text_input.render_head(options_head, head_offset);
00066         } break;
00067
00068         case 6: {
00069             m_index_input.render_head(options_head, head_offset);
00070         } break;
00071
00072         default:
00073             utils::unreachable();
00074     }
00075
00076     m_go |= render_go_button();
00077 }
00078
00079 void DynamicArrayScene::render() {
00080     m_sequence_controller.inc_anim_counter();
00081
00082     int frame_idx = m_sequence_controller.get_anim_frame();
00083     auto* const frame_ptr = m_sequence.find(frame_idx);
00084     m_sequence_controller.set_progress_value(frame_idx);
00085
00086     if (frame_ptr != nullptr) {
00087         frame_ptr->data.render();
00088         m_code_highlighter.highlight(frame_idx);
00089     } else {  // end of sequence
00090         m_array.render();
00091         m_sequence_controller.set_run_all(false);
00092     }
00093
00094     m_code_highlighter.render();
00095     m_sequence_controller.render();
00096     render_options(scene_options);
00097 }
00098
00099 void DynamicArrayScene::interact() {
00100     if (m_sequence_controller.interact()) {
00101         m_sequence_controller.reset_anim_counter();
00102         return;
00103     }
00104
00105     m_index_input.set_random_max((int)m_array.size() - 1);
00106
00107     if (m_text_input.interact() || m_index_input.interact()) {
00108         return;
00109     }
00110
00111     if (!m_go) {
00112         return;
00113     }
00114
00115     int& mode = scene_options.mode_selection;
00116
00117     switch (mode) {
00118         case 0: {
00119             switch (scene_options.action_selection.at(mode)) {
00120                 case 0: {
00121                     interact_random();
00122                 } break;
00123
00124                 case 1: {
00125                     interact_import(m_text_input.extract_values());
00126                 } break;
00127
```

```
00128                 case 2: {
00129                     interact_file_import();
00130                 } break;
00131 
00132                 default:
00133                     utils::unreachable();
00134             }
00135         } break;
00136 
00137         case 1: {
00138             interact_access();
00139         } break;
00140 
00141         case 2: {
00142             switch (scene_options.action_selection.at(mode)) {
00143                 case 0: {
00144                     interact_reserve();
00145                 } break;
00146 
00147                 case 1: {
00148                     interact_shrink();
00149                 } break;
00150 
00151                 default:
00152                     utils::unreachable();
00153             }
00154         } break;
00155 
00156         case 3: {
00157             interact_update();
00158         } break;
00159 
00160         case 4: {
00161             interact_search();
00162         } break;
00163 
00164         case 5: {
00165             interact_insert();
00166         } break;
00167 
00168         case 6: {
00169             interact_delete();
00170         } break;
00171 
00172         default:
00173             utils::unreachable();
00174     }
00175 
00176     m_go = false;
00177 }
00178 
00179 void DynamicArrayScene::interact_access() {
00180     auto index_container = m_index_input.extract_values();
00181     if (index_container.empty()) {
00182         return;
00183     }
00184 
00185     std::size_t index = index_container.front();
00186     if (index >= m_array.size()) {
00187         return;
00188     }
00189 
00190     m_code_highlighter.set_code({"return m_array[index];"});
00191 
00192     m_sequence.clear();
00193 
00194     m_array.set_color_index(index, 3);
00195     m_sequence.insert(m_sequence.size(), m_array);
00196     m_code_highlighter.push_into_sequence(0);
00197 
00198     m_array.set_color_index(index, 0);
00199 
00200     m_sequence_controller.set_max_value((int)m_sequence.size());
00201     m_sequence_controller.set_rerun();
00202 }
00203 
00204 void DynamicArrayScene::interact_reserve() {
00205     auto value_container = m_text_input.extract_values();
00206     if (value_container.empty()) {
00207         return;
00208     }
00209 
00210     std::size_t size = value_container.front();
00211     m_array.reserve(size);
00212 }
00213 
00214 void DynamicArrayScene::interact_shrink() { m_array.shrink_to_fit(); }
```

```
00215
00216 void DynamicArrayScene::interact_random() {
00217     std::size_t size =
00218         utils::get_random(std::size_t{1}, scene_options.max_size);
00219     m_array = {};
00220
00221     for (std::size_t i = 0; i < size; ++i) {
00222         m_array.push(utils::get_random(constants::min_val, constants::max_val));
00223     }
00224
00225     m_array.shrink_to_fit();
00226 }
00227
00228 void DynamicArrayScene::interact_import(core::Deque<int> nums) {
00229     m_array = {};
00230     std::size_t i;  // NOLINT
00231
00232     for (i = 0; i < max_size && !nums.empty(); ++i) {
00233         m_array.push(nums.front());
00234         nums.pop_front();
00235     }
00236
00237     m_array.shrink_to_fit();
00238 }
00239
00240 void DynamicArrayScene::interact_update() {
00241     auto index_container = m_index_input.extract_values();
00242     if (index_container.empty()) {
00243         return;
00244     }
00245
00246     auto value_container = m_text_input.extract_values();
00247     if (value_container.empty()) {
00248         return;
00249     }
00250
00251     int index = index_container.front();
00252     int value = value_container.front();
00253
00254     if (!(0 <= index && index < m_array.size()) ||
00255         !utils::val_in_range(value)) {
00256         return;
00257     }
00258
00259     m_code_highlighter.set_code({
00260         "array[index] = value;",
00261     });
00262
00263     m_sequence.clear();
00264
00265     // initial state (before update)
00266     m_sequence.insert(m_sequence.size(), m_array);
00267     m_code_highlighter.push_into_sequence(-1);
00268
00269     // highlight
00270     m_array.set_color_index(index, 2);
00271     m_sequence.insert(m_sequence.size(), m_array);
00272     m_code_highlighter.push_into_sequence(0);
00273
00274     // update
00275     m_array[index] = value;
00276     m_array.set_color_index(index, 3);
00277     m_sequence.insert(m_sequence.size(), m_array);
00278     m_code_highlighter.push_into_sequence(0);
00279
00280     // undo highlight
00281     m_array.set_color_index(index, 0);
00282
00283     m_sequence_controller.set_max_value((int)m_sequence.size());
00284     m_sequence_controller.set_rerun();
00285 }
00286
00287 void DynamicArrayScene::interact_file_import() {
00288     interact_import(m_file_dialog.extract_values());
00289 }
00290
00291 void DynamicArrayScene::interact_search() {
00292     auto value_container = m_text_input.extract_values();
00293     if (value_container.empty()) {
00294         return;
00295     }
00296
00297     int value = value_container.front();
00298     if (!utils::val_in_range(value)) {
00299         return;
00300     }
00301
```

```
00302     m_code_highlighter.set_code({
00303         "for (i = 0; i < size; i++)",
00304         "    if (array[i] == value)",
00305         "        return i;",
00306         "return not_found",
00307     });
00308
00309     m_sequence.clear();
00310     m_sequence.insert(m_sequence.size(), m_array);
00311     m_code_highlighter.push_into_sequence(0);
00312
00313     bool found = false;
00314
00315     for (std::size_t i = 0; i < m_array.size(); ++i) {
00316         m_array.set_color_index(i, 3);
00317         m_sequence.insert(m_sequence.size(), m_array);
00318         m_code_highlighter.push_into_sequence(1);
00319
00320         if (m_array[i] == value) {
00321             found = true;
00322             m_array.set_color_index(i, 4);
00323             m_sequence.insert(m_sequence.size(), m_array);
00324             m_code_highlighter.push_into_sequence(2);
00325             m_array.set_color_index(i, 0);
00326             break;
00327         }
00328
00329         m_array.set_color_index(i, 0);
00330         m_sequence.insert(m_sequence.size(), m_array);
00331         m_code_highlighter.push_into_sequence(0);
00332     }
00333
00334     if (!found) {
00335         m_sequence.insert(m_sequence.size(), m_array);
00336         m_code_highlighter.push_into_sequence(3);
00337     }
00338
00339     m_sequence_controller.set_max_value((int)m_sequence.size());
00340     m_sequence_controller.set_rerun();
00341 }
00342
00343 void DynamicArrayScene::interact_insert() {
00344     auto index_container = m_index_input.extract_values();
00345     if (index_container.empty()) {
00346         return;
00347     }
00348
00349     auto value_container = m_text_input.extract_values();
00350     if (value_container.empty()) {
00351         return;
00352     }
00353
00354     int index = index_container.front();
00355     int value = value_container.front();
00356
00357     if (m_array.size() >= max_size) {
00358         return;
00359     }
00360
00361     if (!(0 <= index && index <= m_array.size()) ||
00362         !utils::val_in_range(value)) {
00363         return;
00364     }
00365
00366     m_code_highlighter.set_code({
00367         "if (size == capacity)",
00368         "    capacity = max(capacity * 2, 1);",
00369         "size++;",
00370         "for (i = size - 1; i > index; i--)",
00371         "    array[i] = array[i - 1];",
00372         "array[index] = value;",
00373     });
00374
00375     m_sequence.clear();
00376     m_sequence.insert(m_sequence.size(), m_array);
00377     m_code_highlighter.push_into_sequence(-1);
00378
00379     m_sequence.insert(m_sequence.size(), m_array);
00380     m_code_highlighter.push_into_sequence(0);
00381
00382     if (m_array.size() == m_array.capacity()) {
00383         m_array.reserve(m_array.size() * 2);
00384         m_sequence.insert(m_sequence.size(), m_array);
00385         m_code_highlighter.push_into_sequence(1);
00386     }
00387
00388     m_array.push(0);
```

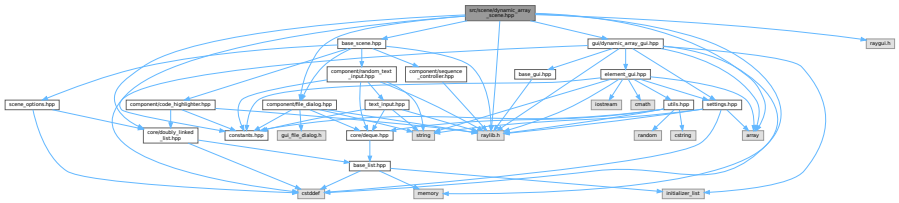
```
00389     m_sequence.insert(m_sequence.size(), m_array);
00390     m_code_highlighter.push_into_sequence(2);
00391
00392     for (std::size_t i = m_array.size() - 1; i > index; --i) {
00393         m_array.set_color_index(i - 1, 2);
00394         m_sequence.insert(m_sequence.size(), m_array);
00395         m_code_highlighter.push_into_sequence(3);
00396
00397         m_array[i] = m_array[i - 1];
00398         m_array.set_color_index(i, 3);
00399         m_sequence.insert(m_sequence.size(), m_array);
00400         m_code_highlighter.push_into_sequence(4);
00401
00402         m_array.set_color_index(i - 1, 0);
00403         m_array.set_color_index(i, 0);
00404         m_sequence.insert(m_sequence.size(), m_array);
00405         m_code_highlighter.push_into_sequence(3);
00406     }
00407
00408     m_array.set_color_index(index, 2);
00409     m_sequence.insert(m_sequence.size(), m_array);
00410     m_code_highlighter.push_into_sequence(5);
00411
00412     m_array[index] = value;
00413     m_array.set_color_index(index, 3);
00414     m_sequence.insert(m_sequence.size(), m_array);
00415     m_code_highlighter.push_into_sequence(5);
00416
00417     m_array.set_color_index(index, 0);
00418     m_sequence.insert(m_sequence.size(), m_array);
00419     m_code_highlighter.push_into_sequence(-1);
00420
00421     m_sequence_controller.set_max_value((int)m_sequence.size());
00422     m_sequence_controller.set_rerun();
00423 }
00424
00425 void DynamicArrayScene::interact_delete() {
00426     auto index_container = m_index_input.extract_values();
00427     if (index_container.empty()) {
00428         return;
00429     }
00430
00431     int index = index_container.front();
00432
00433     if (m_array.size() == 0) {
00434         return;
00435     }
00436
00437     if (!(0 <= index && index < m_array.size())) {
00438         return;
00439     }
00440
00441     m_code_highlighter.set_code({
00442         "for (i = index; i < size - 1; i++)",
00443         "    array[i] = array[i + 1];",
00444         "size--;",
00445     });
00446
00447     m_sequence.clear();
00448     m_sequence.insert(m_sequence.size(), m_array);
00449     m_code_highlighter.push_into_sequence(-1);
00450
00451     m_sequence.insert(m_sequence.size(), m_array);
00452     m_code_highlighter.push_into_sequence(0);
00453
00454     for (std::size_t i = index; i < m_array.size() - 1; ++i) {
00455         m_array.set_color_index(i, 3);
00456         m_sequence.insert(m_sequence.size(), m_array);
00457         m_code_highlighter.push_into_sequence(1);
00458
00459         m_array[i] = m_array[i + 1];
00460         m_array.set_color_index(i + 1, 2);
00461         m_sequence.insert(m_sequence.size(), m_array);
00462         m_code_highlighter.push_into_sequence(1);
00463
00464         m_array.set_color_index(i, 0);
00465         m_array.set_color_index(i + 1, 0);
00466         m_sequence.insert(m_sequence.size(), m_array);
00467         m_code_highlighter.push_into_sequence(0);
00468     }
00469
00470     m_array.set_color_index(m_array.size() - 1, 2);
00471     m_sequence.insert(m_sequence.size(), m_array);
00472     m_code_highlighter.push_into_sequence(2);
00473
00474     m_array.pop();
00475     m_sequence.insert(m_sequence.size(), m_array);
```

```
00476     m_code_highlighter.push_into_sequence(-1);
00477
00478     m_sequence_controller.set_max_value((int)m_sequence.size());
00479     m_sequence_controller.set_rerun();
00480 }
00481
00482 }  // namespace scene
```

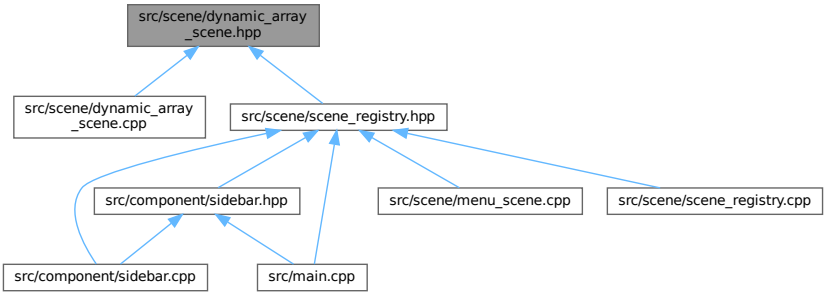
7.83 src/scene/dynamic_array_scene.hpp File Reference

```
#include <array>
#include <cstddef>
#include "base_scene.hpp"
#include "component/file_dialog.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:



Classes

- class scene::DynamicArrayScene

Namespaces

- namespace scene

7.84 dynamic_array_scene.hpp

Go to the documentation of this file.

```
00001 #ifndef SCENE_DYNAMIC_ARRAY_SCENE_HPP_
00002 #define SCENE_DYNAMIC_ARRAY_SCENE_HPP_
00003
00004 #include <array>
00005 #include <cstddef>
00006
00007 #include "base_scene.hpp"
00008 #include "component/file_dialog.hpp"
00009 #include "constants.hpp"
00010 #include "core/doubly_linked_list.hpp"
00011 #include "gui/dynamic_array_gui.hpp"
00012 #include "raygui.h"
00013 #include "raylib.h"
00014
00015 namespace scene {
00016
00017 class DynamicArrayScene : public internal::BaseScene {
00018 private:
00019     static constexpr std::size_t max_size = 8;
00020
00021     internal::SceneOptions scene_options{
00022         // max_size
00023         max_size,
00024
00025         // mode_labels
00026         "Mode: Create;"
00027         "Mode: Access;"
00028         "Mode: Allocate;"
00029         "Mode: Update;"
00030         "Mode: Search;"
00031         "Mode: Insert;"
00032         "Mode: Delete",
00033
00034         // mode_selection
00035         0,
00036
00037         // action_labels
00038         {
00039             // Mode: Create
00040             "Action: Random;Action: Input;Action: File",
00041
00042             // Mode: Access
00043             "",
00044
00045             // Mode: Allocate
00046             "Action: Reserve;Action: Shrink",
00047
00048             // Mode: Update
00049             "",
00050
00051             // Mode: Search
00052             "",
00053
00054             // Mode: Insert
00055             "",
00056
00057             // Mode: Delete
00058             "",
00059         },
00060
00061         // action_selection
00062         core::DoublyLinkedList<int>{0, 0, 0, 0, 0, 0},
00063     };
00064
00065     using internal::BaseScene::button_size;
00066     using internal::BaseScene::head_offset;
00067     using internal::BaseScene::options_head;
00068
00069     gui::GuiDynamicArray<int> m_array{};
00070     core::DoublyLinkedList<gui::GuiDynamicArray<int>> m_sequence;
00071
00072     bool m_go{};
00073     using internal::BaseScene::m_file_dialog;
00074     using internal::BaseScene::m_index_input;
00075     using internal::BaseScene::m_sequence_controller;
00076     using internal::BaseScene::m_text_input;
00077
00078     using internal::BaseScene::render_go_button;
00079     using internal::BaseScene::render_options;
00080     void render_inputs() override;
00081
00082     void interact_random();
```

```
00083     void interact_import(core::Deque<int> nums);
00084     void interact_file_import();
00085     void interact_update();
00086     void interact_search();
00087     void interact_insert();
00088     void interact_delete();
00089     void interact_reserve();
00090     void interact_shrink();
00091     void interact_access();
00092
00093 public:
00094     void render() override;
00095     void interact() override;
00096 };
00097
00098 }  // namespace scene
00099
00100 #endif  // SCENE_DYNAMIC_ARRAY_SCENE_HPP_
```

7.85 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

7.86 menu_scene.cpp

Go to the documentation of this file.

```
00001 #include "menu_scene.hpp"
00002
00003 #include <iostream>
00004
00005 #include "constants.hpp"
00006 #include "raygui.h"
00007 #include "raylib.h"
00008 #include "scene_registry.hpp"
00009 #include "settings.hpp"
00010 #include "utils.hpp"
00011
00012 namespace scene {
00013
00014 MenuScene::MenuScene() {
00015     constexpr int block_width = component::MenuItem::block_width;
00016     constexpr int block_height = component::MenuItem::block_height;
00017     constexpr int button_width = component::MenuItem::button_width;
00018     constexpr int button_height = component::MenuItem::button_height;
00019     constexpr int gap = 20;
00020
```

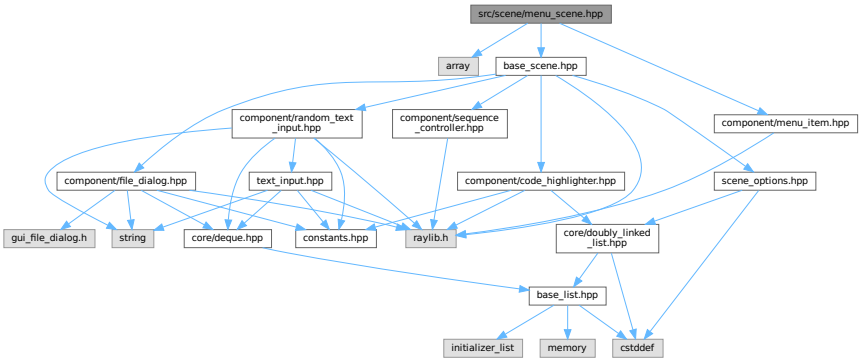
```
00021     constexpr int first_row_y =
00022         constants::scene_height / 16.0F * 5 - block_height / 2.0F;
00023
00024     // first row
00025     {
00026         constexpr int row_width =
00027             3 * component::MenuItem::block_width + 2 * gap;
00028         constexpr int row_x = constants::scene_width / 2.0F - row_width / 2.0F;
00029         constexpr int row_y = first_row_y;
00030
00031         for (auto i = 0; i < 3; ++i) {
00032             m_menu_items[i] = component::MenuItem(
00033                 i, labels[i], row_x + i * (block_width + gap), row_y,
00034                 img_paths[i]);
00035         }
00036     }
00037
00038     // second row
00039     {
00040         constexpr int row_width = 4 * block_width + 3 * gap;
00041         constexpr int row_x = constants::scene_width / 2.0F - row_width / 2.0F;
00042         constexpr int row_y = first_row_y + block_height + gap;
00043
00044         for (auto i = 3; i < 7; ++i) {
00045             m_menu_items[i] = component::MenuItem(
00046                 i, labels[i], row_x + (i - 3) * (block_width + gap), row_y,
00047                 img_paths[i]);
00048         }
00049     }
00050 }
00051
00052 void MenuScene::render() {
00053     const Color text_color = utils::adaptive_text_color(
00054         Settings::get_instance().get_color(Settings::num_color - 1));
00055
00056     // Menu text
00057     constexpr int menu_font_size = 60;
00058     constexpr int menu_font_spacing = 5;
00059
00060     constexpr const char* menu_text = "CS162 - VisuAlgo.net clone in C++";
00061
00062     const Vector2 menu_text_size =
00063         utils::MeasureText(menu_text, menu_font_size, menu_font_spacing);
00064
00065     const Vector2 menu_text_pos{
00066         constants::scene_width / 2.0F - menu_text_size.x / 2,
00067         constants::scene_height / 16.0F - menu_text_size.y / 2};
00068
00069     utils::DrawText(menu_text, menu_text_pos, text_color, menu_font_size,
00070                     menu_font_spacing);
00071
00072     // Sub text
00073     constexpr int sub_font_size = 30;
00074     constexpr int sub_font_spacing = 2;
00075
00076     constexpr const char* sub_text = "By Quang-Truong Nguyen (@jalsol)";
00077
00078     const Vector2 sub_text_size =
00079         utils::MeasureText(sub_text, sub_font_size, sub_font_spacing);
00080
00081     const Vector2 sub_text_pos{
00082         constants::scene_width / 2.0F - sub_text_size.x / 2,
00083         menu_text_pos.y + menu_text_size.y / 2 + sub_text_size.y};
00084
00085     utils::DrawText(sub_text, sub_text_pos, text_color, sub_font_size,
00086                     sub_font_spacing);
00087
00088     // Button
00089     constexpr int block_width = 300;
00090     constexpr int block_height = 200;
00091     constexpr int button_width = block_width;
00092     constexpr int button_height = 50;
00093     constexpr int gap = 20;
00094     constexpr int first_row_y =
00095         constants::scene_height / 16.0F * 5 - block_height / 2.0F;
00096
00097     for (auto i = 0; i < 7; ++i) {
00098         m_menu_items[i].render();
00099     }
00100
00101     const Rectangle quit_button_shape{
00102         constants::scene_width / 2.0F - 128,
00103         constants::scene_height / 16.0F * 15 - block_height / 2.0F, 256, 64};
00104
00105     m_quit = GuiButton(quit_button_shape, "Quit");
00106
00107     // Bottom text
```

```
00108     constexpr int bot_font_size = 20;
00109     constexpr int bot_font_spacing = 2;
00110
00111     constexpr const char* bot_text =
00112         "(pls read the src code, i tried so hard for this)";
00113
00114     const Vector2 bot_text_size =
00115         utils::MeasureText(bot_text, bot_font_size, bot_font_spacing);
00116
00117     const Vector2 bot_text_pos{
00118         constants::scene_width / 2.0F - bot_text_size.x / 2,
00119         constants::scene_height - 1.5F * bot_text_size.y};
00120
00121     utils::DrawText(bot_text, bot_text_pos, text_color, bot_font_size,
00122                     bot_font_spacing);
00123 }
00124
00125 void MenuScene::interact() {
00126     scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();
00127
00128     if (m_quit) {
00129         registry.close_window();
00130         return;
00131     }
00132
00133     for (auto i = 0; i < 7; ++i) {
00134         if (m_menu_items[i].clicked()) {
00135             m_next_scene = i;
00136             m_start = true;
00137         }
00138     }
00139
00140     for (auto i = 0; i < 7; ++i) {
00141         m_menu_items[i].reset();
00142     }
00143
00144     if (m_start) {
00145         registry.set_scene(m_next_scene);
00146         m_start = false;
00147     }
00148 }
00149
00150 }  // namespace scene
```

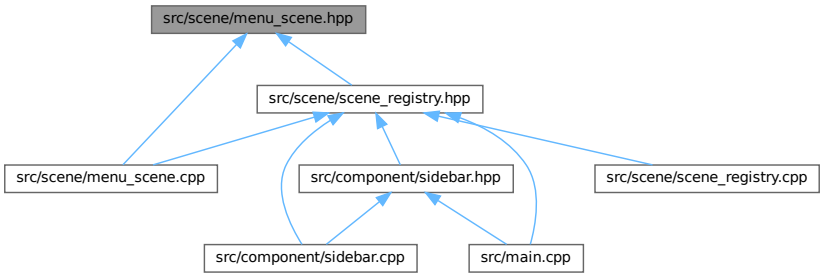
7.87 src/scene/menu_scene.hpp File Reference

```
#include <array>
#include "base_scene.hpp"
#include "component/menu_item.hpp"
```

Include dependency graph for menu_scene.hpp:



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



Classes

- class scene::MenuScene

Namespaces

- namespace scene

7.88 menu_scene.hpp

Go to the documentation of this file.

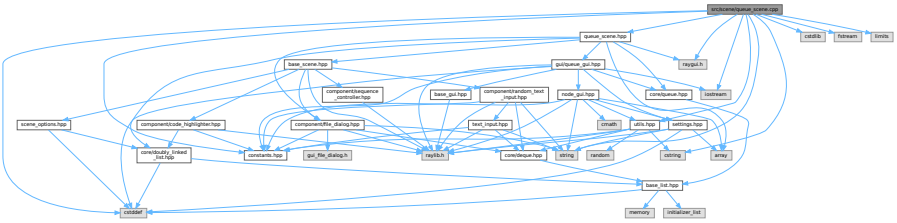
```
00001 #ifndef SCENE_MENU_SCENE_HPP_
00002 #define SCENE_MENU_SCENE_HPP_
00003
00004 #include <array>
00005
00006 #include "base_scene.hpp"
00007 #include "component/menu_item.hpp"
00008
00009 namespace scene {
00010
00011 class MenuScene : public internal::BaseScene {
00012 private:
00013     bool m_start{};
00014     bool m_quit{};
00015     int m_next_scene{};
00016
00017     static constexpr std::array<const char*, 7> labels = {{
00018         "Array",
00019         "Dynamic Array",
00020         "Linked List",
00021         "Doubly Linked List",
00022         "Circular Linked List",
00023         "Stack",
00024         "Queue",
00025     }};
00026
00027     static constexpr std::array<const char*, 7> img_paths = {{
00028         "data/preview/array.png",
00029         "data/preview/dynamic_array.png",
00030         "data/preview/linked_list.png",
00031         "data/preview/doubly_linked_list.png",
00032         "data/preview/circular_linked_list.png",
00033         "data/preview/stack.png",
00034         "data/preview/queue.png",
00035     }};
00036
00037     std::array<component::MenuItem, 7> m_menu_items{};
00038
00039 public:
```

```
00040     MenuScene();
00041     void render() override;
00042     void interact() override;
00043 };
00044
00045 }  // namespace scene
00046
00047 #endif  // SCENE_MENU_SCENE_HPP_
```

7.89 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

7.90 queue_scene.cpp

Go to the documentation of this file.

```
00001 #include "queue_scene.hpp"
00002
00003 #include <cstddef>
00004 #include <cstdlib>
00005 #include <cstring>
00006 #include <fstream>
00007 #include <iostream>
00008 #include <limits>
00009 #include <string>
00010
00011 #include "constants.hpp"
00012 #include "raygui.h"
00013 #include "utils.hpp"
00014
00015 namespace scene {
00016
00017 void QueueScene::render_inputs() {
00018     int& mode = scene_options.mode_selection;
00019
00020     switch (mode) {
```

```
00021         case 0: {
00022             switch (scene_options.action_selection.at(mode)) {
00023                 case 0:
00024                     break;
00025                 case 1: {
00026                     m_text_input.render_head(options_head, head_offset);
00027                 } break;
00028                 case 2: {
00029                     m_go = (m_file_dialog.render_head(options_head,
00030                                                       head_offset) > 0);
00031                     return;
00032                 } break;
00033                 default:
00034                     utils::unreachable();
00035             }
00036         } break;
00037 
00038         case 1: {
00039             m_text_input.render_head(options_head, head_offset);
00040         } break;
00041 
00042         case 2:
00043             break;
00044         default:
00045             utils::unreachable();
00046     }
00047 
00048     m_go |= render_go_button();
00049 }
00050 
00051 void QueueScene::render() {
00052     m_sequence_controller.inc_anim_counter();
00053 
00054     int frame_idx = m_sequence_controller.get_anim_frame();
00055     auto* const frame_ptr = m_sequence.find(frame_idx);
00056     m_sequence_controller.set_progress_value(frame_idx);
00057 
00058     if (frame_ptr != nullptr) {
00059         frame_ptr->data.render();
00060         m_code_highlighter.highlight(frame_idx);
00061     } else {  // end of sequence
00062         m_queue.render();
00063         m_sequence_controller.set_run_all(false);
00064     }
00065 
00066     m_code_highlighter.render();
00067     m_sequence_controller.render();
00068     render_options(scene_options);
00069 }
00070 
00071 void QueueScene::interact() {
00072     if (m_sequence_controller.interact()) {
00073         m_sequence_controller.reset_anim_counter();
00074         return;
00075     }
00076 
00077     m_index_input.set_random_max((int)m_queue.size() - 1);
00078 
00079     if (m_text_input.interact() || m_index_input.interact()) {
00080         return;
00081     }
00082 
00083     if (!m_go) {
00084         return;
00085     }
00086 
00087     int& mode = scene_options.mode_selection;
00088 
00089     switch (mode) {
00090         case 0: {
00091             switch (scene_options.action_selection.at(mode)) {
00092                 case 0: {
00093                     interact_random();
00094                 } break;
00095 
00096                 case 1: {
00097                     interact_import(m_text_input.extract_values());
00098                 } break;
00099 
00100                 case 2: {
00101                     interact_file_import();
00102                 } break;
00103 
00104                 default:
00105                     utils::unreachable();
00106             }
00107         } break;
```

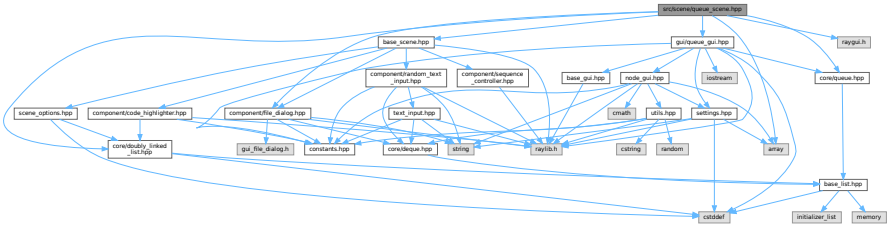
```
00108
00109         case 1: {
00110             interact_push();
00111         } break;
00112
00113         case 2: {
00114             interact_pop();
00115         } break;
00116
00117         default:
00118             utils::unreachable();
00119     }
00120
00121     m_go = false;
00122 }
00123
00124 void QueueScene::interact_random() {
00125     std::size_t size =
00126         utils::get_random(std::size_t{1}, scene_options.max_size);
00127     m_queue = gui::GuiQueue<int>();
00128
00129     for (auto i = 0; i < size; ++i) {
00130         m_queue.push(utils::get_random(constants::min_val, constants::max_val));
00131     }
00132     m_queue.init_label();
00133 }
00134
00135 void QueueScene::interact_import(core::Deque<int> nums) {
00136     m_sequence.clear();
00137     m_queue = gui::GuiQueue<int>();
00138
00139     while (!nums.empty()) {
00140         if (utils::val_in_range(nums.front())) {
00141             m_queue.push(nums.front());
00142         }
00143         nums.pop_front();
00144     }
00145     m_queue.init_label();
00146 }
00147
00148 void QueueScene::interact_file_import() {
00149     interact_import(m_file_dialog.extract_values());
00150 }
00151
00152 void QueueScene::interact_push() {
00153     auto value_container = m_text_input.extract_values();
00154     if (value_container.empty()) {
00155         return;
00156     }
00157
00158     int value = value_container.front();
00159
00160     if (m_queue.size() >= scene_options.max_size) {
00161         return;
00162     }
00163
00164     m_code_highlighter.set_code({
00165         "Node* node = new Node(value);",
00166         "tail->next = node;",
00167         "tail = tail->next;",
00168     });
00169
00170     m_sequence.clear();
00171     m_sequence.insert(m_sequence.size(), m_queue);
00172     m_code_highlighter.push_into_sequence(-1);
00173
00174     m_queue.push(value);
00175     m_queue.back().set_color_index(6);
00176     m_sequence.insert(m_sequence.size(), m_queue);
00177     m_code_highlighter.push_into_sequence(0);
00178
00179     m_queue.pop_back();
00180     if (!m_queue.empty()) {
00181         m_queue.back().set_color_index(4);
00182     }
00183     m_queue.push(value);
00184     m_queue.back().set_color_index(6);
00185     m_sequence.insert(m_sequence.size(), m_queue);
00186     m_code_highlighter.push_into_sequence(1);
00187
00188     m_queue.pop_back();
00189     if (!m_queue.empty()) {
00190         m_queue.back().set_color_index(0);
00191         m_queue.back().set_label("");
00192     }
00193     m_queue.push(value);
00194     m_queue.back().set_color_index(3);
```

```
00195     m_queue.init_label();
00196     m_sequence.insert(m_sequence.size(), m_queue);
00197     m_code_highlighter.push_into_sequence(2);
00198 
00199     m_queue.back().set_color_index(0);
00200 
00201     m_sequence_controller.set_max_value((int)m_sequence.size());
00202     m_sequence_controller.set_rerun();
00203 }
00204 
00205 void QueueScene::interact_pop() {
00206     if (m_queue.empty()) {
00207         return;
00208     }
00209 
00210     m_code_highlighter.set_code({
00211         "Node* temp = head;",
00212         "head = head->next;",
00213         "delete temp;",
00214     });
00215 
00216     m_sequence.clear();
00217     m_sequence.insert(m_sequence.size(), m_queue);
00218     m_code_highlighter.push_into_sequence(-1);
00219 
00220     m_queue.front().set_color_index(5);
00221     m_sequence.insert(m_sequence.size(), m_queue);
00222     m_code_highlighter.push_into_sequence(0);
00223 
00224     auto old_front = m_queue.front();
00225     m_queue.pop();
00226 
00227     if (!m_queue.empty()) {
00228         m_queue.front().set_color_index(3);
00229         if (m_queue.size() == 1) {
00230             m_queue.front().set_label("head/tail");
00231         } else {
00232             m_queue.front().set_label("head");
00233         }
00234     }
00235 
00236     m_queue.push_front(old_front.get_value());
00237     m_queue.front().set_color_index(5);
00238     m_sequence.insert(m_sequence.size(), m_queue);
00239     m_code_highlighter.push_into_sequence(1);
00240 
00241     m_queue.pop();
00242     m_queue.init_label();
00243     m_sequence.insert(m_sequence.size(), m_queue);
00244     m_code_highlighter.push_into_sequence(2);
00245 
00246     if (!m_queue.empty()) {
00247         m_queue.front().set_color_index(0);
00248     }
00249 
00250     m_sequence_controller.set_max_value((int)m_sequence.size());
00251     m_sequence_controller.set_rerun();
00252 }
00253 
00254 }  // namespace scene
```

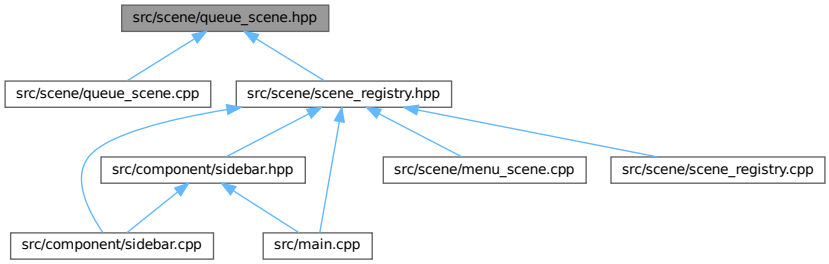
7.91 src/scene/queue_scene.hpp File Reference

```
#include <array>
#include "base_scene.hpp"
#include "component/file_dialog.hpp"
#include "core/doubly_linked_list.hpp"
#include "core/queue.hpp"
#include "gui/queue_gui.hpp"
#include "raygui.h"
```

Include dependency graph for queue_scene.hpp:



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



Classes

- class scene::QueueScene

Namespaces

- namespace scene

7.92 queue_scene.hpp

Go to the documentation of this file.

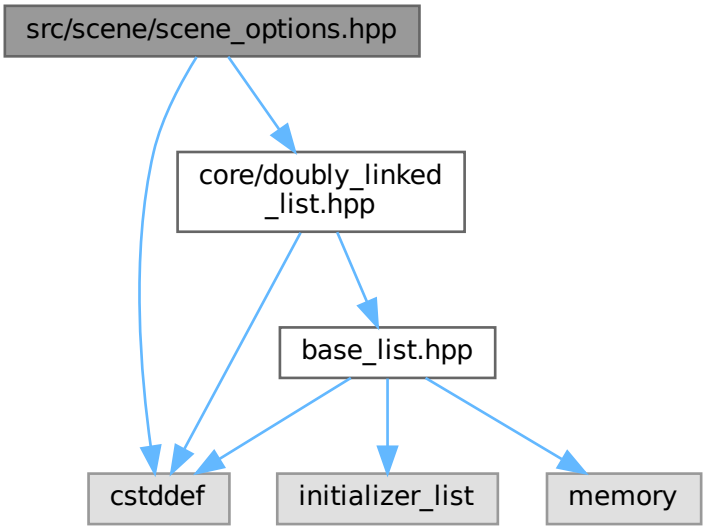
```
00001 #ifndef SCENE_QUEUE_SCENE_HPP_
00002 #define SCENE_QUEUE_SCENE_HPP_
00003
00004 #include <array>
00005
00006 #include "base_scene.hpp"
00007 #include "component/file_dialog.hpp"
00008 #include "core/doubly_linked_list.hpp"
00009 #include "core/queue.hpp"
00010 #include "gui/queue_gui.hpp"
00011 #include "raygui.h"
00012
00013 namespace scene {
00014
00015 class QueueScene : public internal::BaseScene {
00016 private:
00017     internal::SceneOptions scene_options{
00018         // max_size
00019         8,  // NOLINT
00020
```

```
00021         // mode_labels
00022         "Mode: Create;"
00023         "Mode: Push;"
00024         "Mode: Pop",
00025
00026         // mode_selection
00027         0,
00028
00029         // action_labels
00030         {
00031             // Mode: Create
00032             "Action: Random;"
00033             "Action: Input;"
00034             "Action: File",
00035
00036             // Mode: Push
00037             "",
00038
00039             // Mode: Pop
00040             "",
00041         },
00042
00043         // action_selection
00044         core::DoublyLinkedList<int>{0, 0, 0},
00045     };
00046
00047     using internal::BaseScene::button_size;
00048     using internal::BaseScene::head_offset;
00049     using internal::BaseScene::options_head;
00050
00051     gui::GuiQueue<int> m_queue{
00052         gui::GuiNode<int>{1},
00053         gui::GuiNode<int>{2},
00054         gui::GuiNode<int>{3},
00055     };
00056     core::DoublyLinkedList<gui::GuiQueue<int>> m_sequence;
00057
00058     bool m_go{};
00059     using internal::BaseScene::m_code_highlighter;
00060     using internal::BaseScene::m_file_dialog;
00061     using internal::BaseScene::m_sequence_controller;
00062     using internal::BaseScene::m_text_input;
00063
00064     using internal::BaseScene::render_go_button;
00065     using internal::BaseScene::render_options;
00066     void render_inputs() override;
00067
00068     void interact_random();
00069     void interact_import(core::Deque<int> nums);
00070     void interact_file_import();
00071     void interact_push();
00072     void interact_pop();
00073
00074 public:
00075     void render() override;
00076     void interact() override;
00077 };
00078
00079 }  // namespace scene
00080
00081 #endif  // SCENE_QUEUE_SCENE_HPP_
```

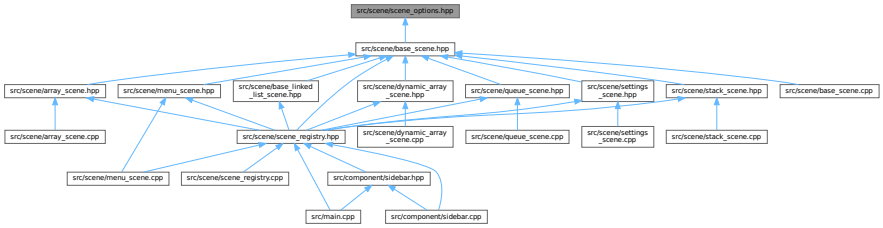
7.93 src/scene/scene_options.hpp File Reference

```
#include <cstddef>
#include "core/doubly_linked_list.hpp"
```

Include dependency graph for scene_options.hpp:



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



Classes

- struct scene::internal::SceneOptions

Namespaces

- namespace scene
- namespace scene::internal

7.94 scene_options.hpp

Go to the documentation of this file.

```
00001 #ifndef SCENE_SCENE_OPTIONS_HPP_
00002 #define SCENE_SCENE_OPTIONS_HPP_
00003
00004 #include <cstddef>
00005
00006 #include "core/doubly_linked_list.hpp"
00007
00008 namespace scene::internal {
00009
00010 struct SceneOptions {
00011     const std::size_t max_size{};
00012     const char* mode_labels{};
00013     int mode_selection{};
00014     core::DoublyLinkedList<const char*> action_labels;
00015     core::DoublyLinkedList<int> action_selection;
00016 };
00017
00018 }  // namespace scene::internal
00019
00020 #endif  // SCENE_SCENE_OPTIONS_HPP_
```

7.95 src/scene/scene_registry.cpp File Reference

`#include "scene_registry.hpp"`
Include dependency graph for scene_registry.cpp:

Namespaces

- namespace scene

7.96 scene_registry.cpp

Go to the documentation of this file.

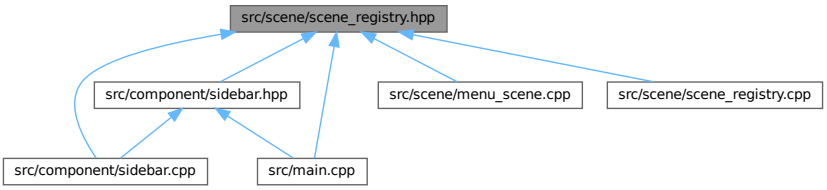
```
00001 #include "scene_registry.hpp"
00002
00003 namespace scene {
00004
00005 SceneRegistry::SceneRegistry() { set_scene(Menu); }
00006
00007 SceneRegistry& SceneRegistry::get_instance() {
00008     static SceneRegistry registry;
00009     return registry;
00010 }
00011
00012 void SceneRegistry::set_scene(int scene_type) {
00013     m_current_scene = scene_type;
00014     scene_ptr = m_registry.at(scene_type).get();
00015 }
00016
00017 int SceneRegistry::get_scene() const { return m_current_scene; }
00018
00019 void SceneRegistry::render() { scene_ptr->render(); }
00020
00021 void SceneRegistry::interact() { scene_ptr->interact(); }
00022
00023 bool SceneRegistry::should_close() const { return m_should_close; }
00024
00025 void SceneRegistry::close_window() { m_should_close = true; }
00026
00027 }  // namespace scene
```

7.97 src/scene/scene_registry.hpp File Reference

```
#include <array>
#include <memory>
#include "array_scene.hpp"
#include "base_linked_list_scene.hpp"
#include "base_scene.hpp"
#include "dynamic_array_scene.hpp"
#include "menu_scene.hpp"
#include "queue_scene.hpp"
#include "settings_scene.hpp"
#include "stack_scene.hpp"
```

Include dependency graph for scene_registry.hpp:

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



Classes

- class scene::SceneRegistry

Namespaces

- namespace scene

Enumerations

- enum scene::SceneId {
  scene::Array , scene::DynamicArray , scene::LinkedList , scene::DoublyLinkedList ,
  scene::CircularLinkedList , scene::Stack , scene::Queue , scene::Menu ,
  scene::Settings }

7.98 scene_registry.hpp

Go to the documentation of this file.

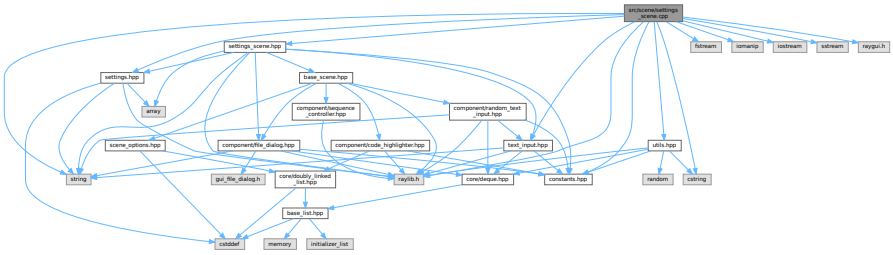
```
00001 #ifndef SCENE_SCENE_REGISTRY_HPP_
00002 #define SCENE_SCENE_REGISTRY_HPP_
00003
00004 #include <array>
00005 #include <memory>
00006
00007 #include "array_scene.hpp"
00008 #include "base_linked_list_scene.hpp"
00009 #include "base_scene.hpp"
00010 #include "dynamic_array_scene.hpp"
00011 #include "menu_scene.hpp"
00012 #include "queue_scene.hpp"
00013 #include "settings_scene.hpp"
00014 #include "stack_scene.hpp"
00015
00016 namespace scene {
00017
00018 enum SceneId {
00019     Array,
00020     DynamicArray,
00021     LinkedList,
00022     DoublyLinkedList,
00023     CircularLinkedList,
00024     Stack,
00025     Queue,
00026     Menu,
00027     Settings,
00028 };
00029
00030 class SceneRegistry {
00031 private:
00032     internal::BaseScene* scene_ptr{};
00033     SceneRegistry();
00034
00035     bool m_should_close{};
00036     int m_current_scene{};
00037
00038     const std::array<const std::unique_ptr<internal::BaseScene>, 9> m_registry{{
00039         std::make_unique<ArrayScene>(),
00040         std::make_unique<DynamicArrayScene>(),
00041         std::make_unique<LinkedListScene>(),
00042         std::make_unique<DoublyLinkedListScene>(),
00043         std::make_unique<CircularLinkedListScene>(),
00044         std::make_unique<StackScene>(),
00045         std::make_unique<QueueScene>(),
00046         std::make_unique<MenuScene>(),
00047         std::make_unique<SettingsScene>(),
00048     }};
00049
00050 public:
00051     SceneRegistry(const SceneRegistry&) = delete;
00052     SceneRegistry(SceneRegistry&&) = delete;
00053     SceneRegistry& operator=(const SceneRegistry&) = delete;
00054     SceneRegistry& operator=(SceneRegistry&&) = delete;
00055     ~SceneRegistry() = default;
00056
00057     static SceneRegistry& get_instance();
00058
00059     void set_scene(int scene_type);
00060     int get_scene() const;
00061     void render();
00062     void interact();
00063     bool should_close() const;
00064     void close_window();
00065 };
00066
00067 }  // namespace scene
00068
00069 #endif  // SCENE_SCENE_REGISTRY_HPP_
```

7.99 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

7.100 settings_scene.cpp

Go to the documentation of this file.

```
00001 #include "settings_scene.hpp"
00002
00003 #include <cstring>
00004 #include <fstream>
00005 #include <iomanip>
00006 #include <iostream>
00007 #include <sstream>
00008 #include <string>
00009
00010 #include "component/text_input.hpp"
00011 #include "constants.hpp"
00012 #include "raygui.h"
00013 #include "raylib.h"
00014 #include "settings.hpp"
00015 #include "utils.hpp"
00016
00017 namespace scene {
00018
00019 void SettingsScene::open_from_file(const std::string& path) {
00020     Settings& settings = Settings::get_instance();
00021     std::ifstream file_in(path, std::ios::binary);
00022
00023     if (!file_in.is_open()) {
00024         std::ofstream file_out(path, std::ios::binary);
00025
00026         for (auto i = 0; i < Settings::num_color; ++i) {
00027             unsigned value = Settings::default_color.at(i);
00028             file_out.write(reinterpret_cast<const char*>(&value),
00029                            sizeof(value));
00030         }
00031
00032         file_out.close();
00033
00034         file_in.close();
00035         file_in.open(path, std::ios::binary);
00036     }
```

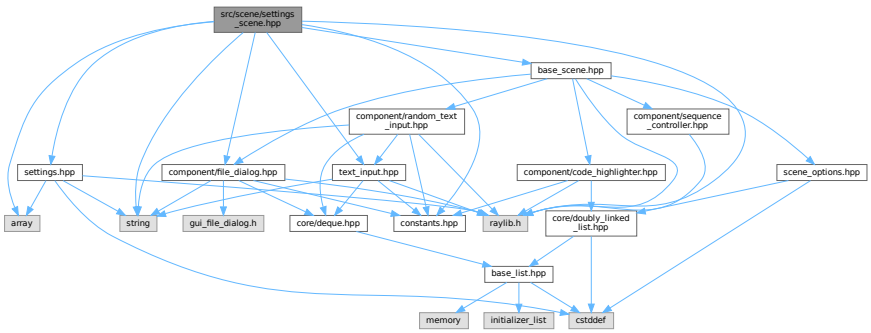
```
00037
00038     unsigned hex_value;
00039     for (auto i = 0; i < Settings::num_color; ++i) {
00040         file_in.read(reinterpret_cast<char*>(&hex_value), sizeof(hex_value));
00041         settings.get_color(i) = GetColor(hex_value);
00042     }
00043
00044     set_buffer();
00045 }
00046
00047 SettingsScene::SettingsScene() {
00048     open_from_file(constants::default_color_path);
00049 }
00050
00051 void SettingsScene::set_buffer() {
00052     std::stringstream sstr;
00053
00054     for (auto i = 0; i < Settings::num_color; ++i) {
00055         sstr « std::setfill('0') « std::setw(6) « std::hex
00056              « ((unsigned)ColorToInt(Settings::get_instance().get_color(i)) »
00057                  8);
00058         m_inputs.at(i).set_input(sstr.str().c_str(), 7);
00059         sstr.str(std::string());
00060     }
00061 }
00062
00063 void SettingsScene::set_color() {
00064     for (auto i = 0; i < Settings::num_color; ++i) {
00065         Settings::get_instance().get_color(i) =
00066             utils::color_from_hex(m_inputs.at(i).get_input());
00067     }
00068 }
00069
00070 void SettingsScene::render() {
00071     Settings& settings = Settings::get_instance();
00072     constexpr int second_col_x = constants::scene_width / 2 + head_pos.y;
00073     int second_col_y = 100;
00074     constexpr int vertical_gap = 30;
00075     const Color text_color =
00076         utils::adaptive_text_color(settings.get_color(Settings::num_color - 1));
00077
00078     auto [head_x, head_y] = head_pos;
00079     const auto input_size = component::TextInput::size;
00080
00081     for (auto i = 0; i < m_inputs.size(); ++i) {
00082         Vector2 input_head;
00083
00084         if (i + 1 != m_inputs.size()) {
00085             input_head = {(float)head_x, (float)head_y};
00086         } else {
00087             input_head = {(float)second_col_x, (float)second_col_y + 400};
00088         }
00089
00090         // to be honest, I don't exactly know how TextFormat works
00091         // there are some bizarre behaviors which make me call set_label
00092         // every frame
00093         if (i + 1 != m_inputs.size()) {
00094             m_inputs.at(i).set_label(TextFormat("Color %d", i + 1));
00095         } else {
00096             m_inputs.at(i).set_label("Background color");
00097         }
00098
00099         m_inputs.at(i).render(input_head.x, input_head.y);
00100
00101         const Rectangle preview_shape{input_head.x + input_size.x + 10,
00102                                       input_head.y, input_size.y, input_size.y};
00103
00104         DrawRectangleRec(preview_shape, settings.get_color(i));
00105
00106         if (m_selected == i) {
00107             DrawRectangleLinesEx(preview_shape, 3, settings.get_color(5));
00108         } else {
00109             DrawRectangleLinesEx(preview_shape, 2, text_color);
00110         }
00111
00112         head_y += input_size.y + vertical_gap;
00113     }
00114
00115     {
00116         Color& color = settings.get_color(m_selected);
00117         auto new_color = GuiColorPicker({second_col_x, (float)second_col_y,
00118                                          4 * input_size.y, 4 * input_size.y},
00119                                         nullptr, color);
00120
00121         if (ColorToInt(color) != ColorToInt(new_color)) {
00122             color = new_color;
00123             set_buffer();
```

```
00124         }
00125     }
00126
00127     {
00128         second_col_y += 4 * input_size.y;
00129         utils::DrawText("Import config",
00130                         {second_col_x + 10, (float)second_col_y}, text_color,
00131                         20, 2);
00132         m_open = m_open_file.render(second_col_x, (float)second_col_y + 25);
00133     }
00134
00135     {
00136         second_col_y += component::FileDialog::size.y + vertical_gap;
00137         utils::DrawText("Export config",
00138                         {second_col_x + 10, (float)second_col_y}, text_color,
00139                         20, 2);
00140         m_save = m_save_file.render(second_col_x, (float)second_col_y + 25);
00141     }
00142 }
00143
00144 void SettingsScene::interact() {
00145     if (m_open > 0) {
00146         open_from_file(m_open_file.get_path());
00147         return;
00148     }
00149
00150     if (m_save > 0) {
00151         Settings::get_instance().save_to_file(m_save_file.get_path());
00152         return;
00153     }
00154
00155     const Vector2 mouse = GetMousePosition();
00156     const bool left_clicked = IsMouseButtonPressed(MOUSE_LEFT_BUTTON);
00157     auto [head_x, head_y] = head_pos;
00158
00159     for (auto i = 0; i < m_inputs.size(); ++i) {
00160         if (m_inputs.at(i).is_active()) {
00161             m_selected = i;
00162         }
00163     }
00164
00165     set_color();
00166 }
00167
00168 }  // namespace scene
```

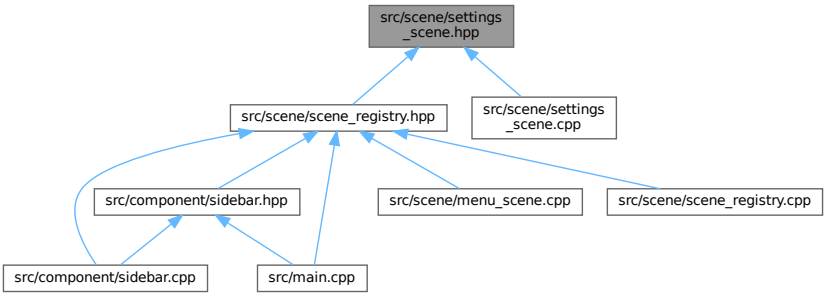
7.101 src/scene/settings_scene.hpp File Reference

```
#include <array>
#include <constants.hpp>
#include <string>
#include "base_scene.hpp"
#include "component/file_dialog.hpp"
#include "component/text_input.hpp"
#include "raylib.h"
#include "settings.hpp"
```

Include dependency graph for settings_scene.hpp:



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



Classes

- class scene::SettingsScene

Namespaces

- namespace scene

7.102 settings_scene.hpp

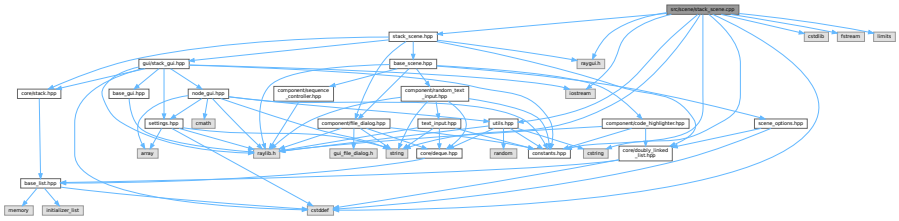
Go to the documentation of this file.

```
00001 #ifndef SCENE_SETTINGS_SCENE_HPP_
00002 #define SCENE_SETTINGS_SCENE_HPP_
00003
00004 #include <array>
00005 #include <constants.hpp>
00006 #include <string>
00007
00008 #include "base_scene.hpp"
00009 #include "component/file_dialog.hpp"
00010 #include "component/text_input.hpp"
00011 #include "raylib.h"
00012 #include "settings.hpp"
00013
00014 namespace scene {
00015
00016 class SettingsScene : public internal::BaseScene {
00017 private:
00018     static constexpr Vector2 head_pos{400, 70};
00019     std::array<component::TextInput, Settings::num_color> m_inputs{};
00020
00021     int m_selected{};
00022
00023     component::FileDialog m_open_file;
00024     component::FileDialog m_save_file{3, "Save file...", "Save file"};
00025     int m_open{};
00026     int m_save{};
00027
00028     void set_buffer();
00029     void set_color();
00030     void open_from_file(const std::string& path);
00031
00032 public:
00033     SettingsScene();
00034
00035     void render() override;
00036     void interact() override;
00037 };
00038
00039 }  // namespace scene
00040
00041 #endif  // SCENE_SETTINGS_SCENE_HPP_
```

7.103 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

7.104 stack_scene.cpp

Go to the documentation of this file.

```
00001 #include "stack_scene.hpp"
00002
00003 #include <cstddef>
00004 #include <cstdlib>
00005 #include <cstring>
00006 #include <fstream>
00007 #include <iostream>
00008 #include <limits>
00009 #include <string>
00010
00011 #include "constants.hpp"
00012 #include "raygui.h"
00013 #include "utils.hpp"
00014
00015 namespace scene {
00016
00017 void StackScene::render() {
00018     m_sequence_controller.inc_anim_counter();
00019
00020     int frame_idx = m_sequence_controller.get_anim_frame();
00021     auto* const frame_ptr = m_sequence.find(frame_idx);
00022     m_sequence_controller.set_progress_value(frame_idx);
00023
00024     if (frame_ptr != nullptr) {
00025         frame_ptr->data.render();
00026         m_code_highlighter.highlight(frame_idx);
00027     } else {  // end of sequence
00028         m_stack.render();
00029         m_sequence_controller.set_run_all(false);
00030     }
00031
00032     m_code_highlighter.render();
```

```
00033     m_sequence_controller.render();
00034     render_options(scene_options);
00035 }
00036
00037 void StackScene::render_inputs() {
00038     int& mode = scene_options.mode_selection;
00039
00040     switch (mode) {
00041         case 0: {
00042             switch (scene_options.action_selection.at(mode)) {
00043                 case 0:
00044                     break;
00045                 case 1: {
00046                     m_text_input.render_head(options_head, head_offset);
00047                 } break;
00048                 case 2: {
00049                     m_go = (m_file_dialog.render_head(options_head,
00050                                                       head_offset) > 0);
00051                     return;
00052                 } break;
00053                 default:
00054                     utils::unreachable();
00055             }
00056         } break;
00057
00058         case 1: {
00059             m_text_input.render_head(options_head, head_offset);
00060         } break;
00061
00062         case 2:
00063             break;
00064         default:
00065             utils::unreachable();
00066     }
00067
00068     m_go |= render_go_button();
00069 }
00070
00071 void StackScene::interact() {
00072     if (m_sequence_controller.interact()) {
00073         m_sequence_controller.reset_anim_counter();
00074         return;
00075     }
00076
00077     m_index_input.set_random_max((int)m_stack.size() - 1);
00078     if (m_text_input.interact() || m_index_input.interact()) {
00079         return;
00080     }
00081
00082     if (!m_go) {
00083         return;
00084     }
00085
00086     int& mode = scene_options.mode_selection;
00087
00088     switch (mode) {
00089         case 0: {
00090             switch (scene_options.action_selection.at(mode)) {
00091                 case 0: {
00092                     interact_random();
00093                 } break;
00094
00095                 case 1: {
00096                     interact_import(m_text_input.extract_values());
00097                 } break;
00098
00099                 case 2: {
00100                     interact_file_import();
00101                 } break;
00102
00103                 default:
00104                     utils::unreachable();
00105             }
00106         } break;
00107
00108         case 1: {
00109             interact_push();
00110         } break;
00111
00112         case 2: {
00113             interact_pop();
00114         } break;
00115
00116         default:
00117             utils::unreachable();
00118     }
00119
```

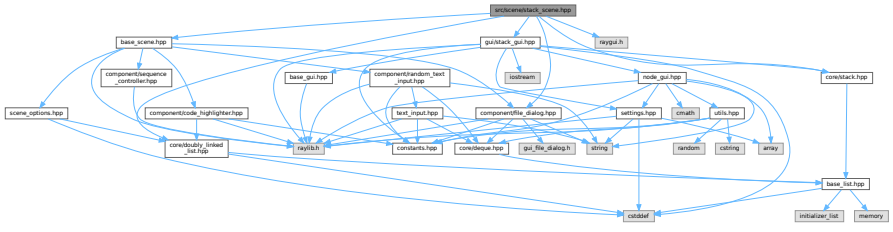
```
00120     m_go = false;
00121 }
00122
00123 void StackScene::interact_random() {
00124     std::size_t size =
00125         utils::get_random(std::size_t{1}, scene_options.max_size);
00126     m_stack = gui::GuiStack<int>();
00127
00128     for (auto i = 0; i < size; ++i) {
00129         m_stack.push(utils::get_random(constants::min_val, constants::max_val));
00130     }
00131     m_stack.init_label();
00132 }
00133
00134 void StackScene::interact_import(core::Deque<int> nums) {
00135     m_sequence.clear();
00136     m_stack = gui::GuiStack<int>();
00137
00138     while (!nums.empty()) {
00139         if (utils::val_in_range(nums.back())) {
00140             m_stack.push(nums.back());
00141         }
00142         nums.pop_back();
00143     }
00144     m_stack.init_label();
00145 }
00146
00147 void StackScene::interact_push() {
00148     auto value_container = m_text_input.extract_values();
00149     if (value_container.empty()) {
00150         return;
00151     }
00152
00153     int value = value_container.front();
00154
00155     if (m_stack.size() >= scene_options.max_size) {
00156         return;
00157     }
00158
00159     m_code_highlighter.set_code({
00160         "Node* node = new Node(value);",
00161         "node->next = head;",
00162         "head = node;",
00163     });
00164
00165     m_sequence.clear();
00166     m_sequence.insert(m_sequence.size(), m_stack);
00167     m_code_highlighter.push_into_sequence(-1);
00168
00169     m_stack.push(value);
00170     m_stack.top().set_color_index(6);
00171     m_sequence.insert(m_sequence.size(), m_stack);
00172     m_code_highlighter.push_into_sequence(0);
00173
00174     m_stack.pop();
00175     if (!m_stack.empty()) {
00176         m_stack.top().set_color_index(4);
00177     }
00178     m_stack.push(value);
00179     m_stack.top().set_color_index(6);
00180     m_sequence.insert(m_sequence.size(), m_stack);
00181     m_code_highlighter.push_into_sequence(1);
00182
00183     m_stack.pop();
00184     if (!m_stack.empty()) {
00185         m_stack.top().set_color_index(0);
00186         m_stack.top().set_label("");
00187     }
00188     m_stack.push(value);
00189     m_stack.top().set_color_index(3);
00190     m_stack.init_label();
00191     m_sequence.insert(m_sequence.size(), m_stack);
00192     m_code_highlighter.push_into_sequence(2);
00193
00194     m_stack.top().set_color_index(0);
00195
00196     m_sequence_controller.set_max_value((int)m_sequence.size());
00197     m_sequence_controller.set_rerun();
00198 }
00199
00200 void StackScene::interact_pop() {
00201     if (m_stack.empty()) {
00202         return;
00203     }
00204
00205     m_code_highlighter.set_code({
00206         "Node* temp = head;",
```

```
00207         "head = head->next;",
00208         "delete temp;",
00209     });
00210
00211     m_sequence.clear();
00212     m_sequence.insert(m_sequence.size(), m_stack);
00213     m_code_highlighter.push_into_sequence(-1);
00214
00215     m_stack.top().set_color_index(5);
00216     m_sequence.insert(m_sequence.size(), m_stack);
00217     m_code_highlighter.push_into_sequence(0);
00218
00219     auto old_top = m_stack.top();
00220     m_stack.pop();
00221
00222     if (!m_stack.empty()) {
00223         m_stack.top().set_color_index(3);
00224         m_stack.top().set_label("head");
00225     }
00226
00227     m_stack.push(old_top.get_value());
00228     m_stack.top().set_color_index(5);
00229     m_sequence.insert(m_sequence.size(), m_stack);
00230     m_code_highlighter.push_into_sequence(1);
00231
00232     m_stack.pop();
00233     m_sequence.insert(m_sequence.size(), m_stack);
00234     m_code_highlighter.push_into_sequence(2);
00235
00236     if (!m_stack.empty()) {
00237         m_stack.top().set_color_index(0);
00238     }
00239
00240     m_sequence_controller.set_max_value((int)m_sequence.size());
00241     m_sequence_controller.set_rerun();
00242 }
00243
00244 void StackScene::interact_file_import() {
00245     interact_import(m_file_dialog.extract_values());
00246 }
00247
00248 }  // namespace scene
```

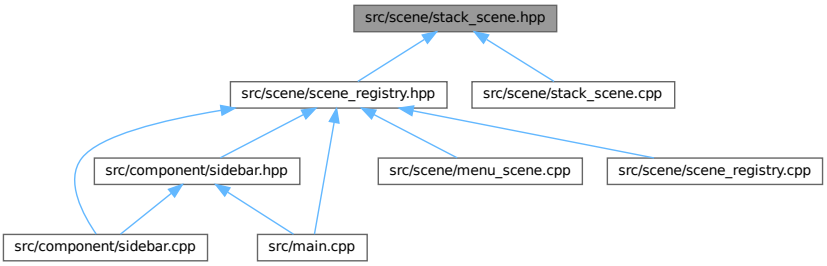
7.105 src/scene/stack_scene.hpp File Reference

```
#include "base_scene.hpp"
#include "component/file_dialog.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:



Classes

- class scene::StackScene

Namespaces

- namespace scene

7.106 stack_scene.hpp

Go to the documentation of this file.

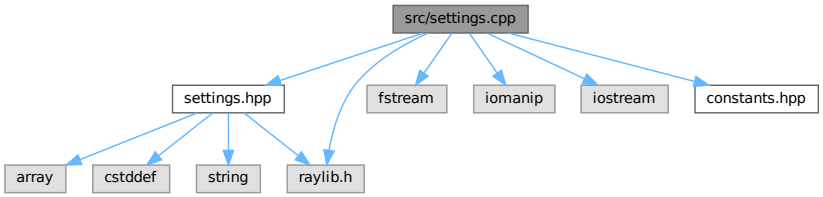
```
00001 #ifndef SCENE_STACK_SCENE_HPP_
00002 #define SCENE_STACK_SCENE_HPP_
00003
00004 #include "base_scene.hpp"
00005 #include "component/file_dialog.hpp"
00006 #include "core/doubly_linked_list.hpp"
00007 #include "core/stack.hpp"
00008 #include "gui/stack_gui.hpp"
00009 #include "raygui.h"
00010
00011 namespace scene {
00012
00013 class StackScene : public internal::BaseScene {
00014 private:
00015     internal::SceneOptions scene_options{
00016         // max_size
00017         8,  // NOLINT
00018
00019         // mode_labels
00020         "Mode: Create;"
00021         "Mode: Push;"
00022         "Mode: Pop",
00023
00024         // mode_selection
00025         0,
00026
00027         // action_labels
00028         {
00029             // Mode: Create
00030             "Action: Random;"
00031             "Action: Input;"
00032             "Action: File",
00033
00034             // Mode: Push
00035             "",
00036
00037             // Mode: Pop
00038             "",
00039         },
00040
```

```
00041         // action_selection
00042         core::DoublyLinkedList<int>{0, 0, 0},
00043     };
00044
00045     using internal::BaseScene::button_size;
00046     using internal::BaseScene::head_offset;
00047     using internal::BaseScene::options_head;
00048
00049     gui::GuiStack<int> m_stack{
00050         gui::GuiNode<int>{1},
00051         gui::GuiNode<int>{2},
00052         gui::GuiNode<int>{3},
00053     };
00054     core::DoublyLinkedList<gui::GuiStack<int>> m_sequence;
00055
00056     bool m_go{};
00057     using internal::BaseScene::m_code_highlighter;
00058     using internal::BaseScene::m_file_dialog;
00059     using internal::BaseScene::m_sequence_controller;
00060     using internal::BaseScene::m_text_input;
00061
00062     using internal::BaseScene::render_go_button;
00063     using internal::BaseScene::render_options;
00064     void render_inputs() override;
00065
00066     void interact_random();
00067     void interact_import(core::Deque<int> nums);
00068     void interact_push();
00069     void interact_pop();
00070     void interact_file_import();
00071
00072 public:
00073     void render() override;
00074     void interact() override;
00075 };
00076
00077 }  // namespace scene
00078
00079 #endif  // SCENE_STACK_SCENE_HPP_
```

7.107 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:



7.108 settings.cpp

Go to the documentation of this file.

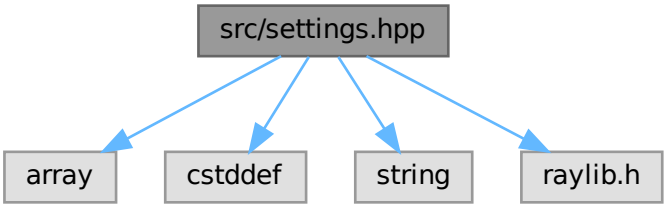
```
00001 #include "settings.hpp"
00002
```

```
00003 #include <fstream>
00004 #include <iomanip>
00005 #include <iostream>
00006
00007 #include "constants.hpp"
00008 #include "raylib.h"
00009
00010 Settings& Settings::get_instance() {
00011     static Settings settings;
00012     return settings;
00013 }
00014
00015 void Settings::save_to_file(const std::string& path) {
00016     std::ofstream file_out(path, std::ios::binary);
00017
00018     for (auto i = 0; i < num_color; ++i) {
00019         unsigned value = ColorToInt(m_colors.at(i));
00020         file_out.write(reinterpret_cast<const char*>(&value), sizeof(value));
00021     }
00022 }
00023
00024 Settings::~Settings() { save_to_file(constants::default_color_path); }
00025
00026 Color& Settings::get_color(std::size_t index) { return m_colors.at(index); }
00027
00028 Color Settings::get_color(std::size_t index) const {
00029     return m_colors.at(index);
00030 }
```

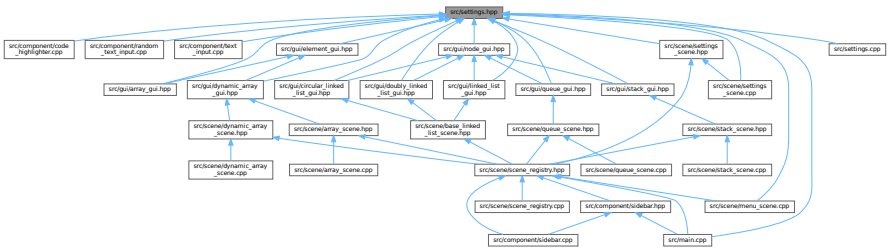
7.109 src/settings.hpp File Reference

```
#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:



Classes

- class Settings

7.110 settings.hpp

Go to the documentation of this file.

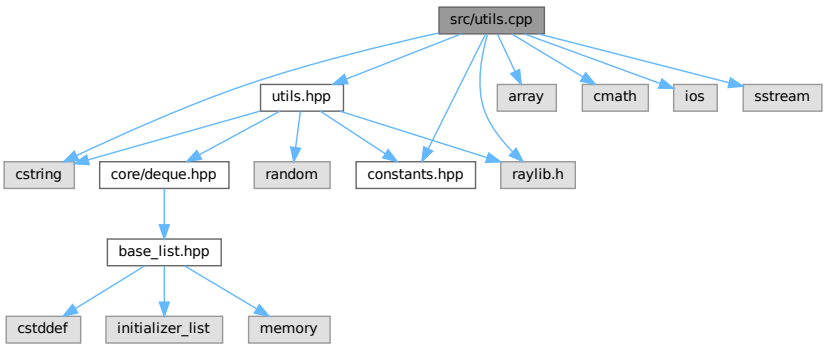
```
00001 #ifndef SETTINGS_HPP_
00002 #define SETTINGS_HPP_
00003
00004 #include <array>
00005 #include <cstddef>
00006 #include <string>
00007
00008 #include "raylib.h"
00009
00010 class Settings {
00011 public:
00012     static constexpr int num_color = 9;
00013     static constexpr std::array<unsigned, num_color> default_color{{
00014         0x00000000,
00015         0x82828200,
00016         0xffa10000,
00017         0x00e43000,
00018         0x873cbe00,
00019         0xe6293700,
00020         0x0079f100,
00021         0xff6dc200,
00022         0xf5f5f500,
00023     }};
00024
00025 private:
00026     Settings() = default;
00027     std::array<Color, num_color> m_colors{};
00028
00029 public:
00030     Settings(const Settings&) = delete;
00031     Settings(Settings&&) = delete;
00032     Settings& operator=(const Settings&) = delete;
00033     Settings& operator=(Settings&&) = delete;
00034     ~Settings();
00035
00036     static Settings& get_instance();
00037
00038     Color& get_color(std::size_t index);
00039     Color get_color(std::size_t index) const;
00040
00041     void save_to_file(const std::string& path);
00042 };
00043
00044 #endif  // SETTINGS_HPP_
```

7.111 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)

7.112 utils.cpp

Go to the documentation of this file.

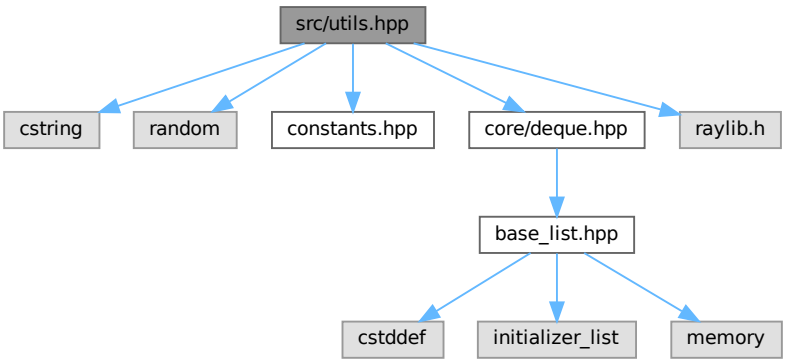
```
00001 #include "utils.hpp"
00002
00003 #include <array>
00004 #include <cmath>
00005 #include <cstring>
00006 #include <ios>
00007 #include <sstream>
00008
00009 #include "constants.hpp"
00010 #include "raylib.h"
00011
00012 namespace utils {
00013
00014 void DrawText(const char* text, Vector2 pos, Color color, float font_size,
00015               float spacing) {
00016     static Font font = LoadFontEx("data/open_sans.ttf",
00017                                   constants::default_font_size, nullptr, 0);
00018
00019     Vector2 pos_vec{static_cast<float>(pos.x), static_cast<float>(pos.y)};
00020     DrawTextEx(font, text, pos_vec, font_size, spacing, color);
00021 }
00022
00023 Vector2 MeasureText(const char* text, float font_size, float spacing) {
```

```
00024     static Font font = LoadFontEx("data/open_sans.ttf",
00025                                   constants::default_font_size, nullptr, 0);
00026
00027     return MeasureTextEx(font, text, font_size, spacing);
00028 }
00029
00030 core::Deque<int> str_extract_data(
00031     char str[constants::text_buffer_size]) {  // NOLINT
00032     char str_copy[constants::text_buffer_size];
00033     strncpy(str_copy, str, constants::text_buffer_size);
00034
00035     char* save_ptr = nullptr;
00036     char* token = utils::strtok(str_copy, ",", &save_ptr);
00037
00038     if (token == nullptr) {
00039         return {};
00040     }
00041
00042     core::Deque<int> ret;
00043
00044     constexpr int base = 10;
00045     int num = static_cast<int>(std::strtol(token, nullptr, base));
00046     ret.push_back(num);
00047
00048     while (true) {
00049         token = utils::strtok(nullptr, ",", &save_ptr);
00050         if (token == nullptr) {
00051             break;
00052         }
00053
00054         num = static_cast<int>(std::strtol(token, nullptr, base));
00055         ret.push_back(num);
00056     }
00057
00058     return ret;
00059 }
00060
00061 bool val_in_range(int num) {
00062     return constants::min_val <= num && num <= constants::max_val;
00063 }
00064
00065 void unreachable() {
00066 #if defined(_MSC_VER)
00067     __assume(0);
00068 #else
00069     __builtin_unreachable();
00070 #endif
00071 }
00072
00073 char* strtok(char* str, const char* delim, char** save_ptr) {
00074     return
00075 #if defined(_MSC_VER)
00076         strtok_s(str, delim, save_ptr);
00077 #else
00078         strtok_r(str, delim, save_ptr);
00079 #endif
00080 }
00081
00082 Color color_from_hex(const std::string& hex) {
00083     std::stringstream stream(hex + "ff");
00084     unsigned int value;
00085     stream >> std::hex >> value;
00086     return GetColor(value);
00087 }
00088
00089 // https://stackoverflow.com/a/3943023
00090 Color adaptive_text_color(Color bg_color) {
00091     constexpr std::array<float, 3> threshold{{0.2126, 0.7152, 0.0722}};
00092     const std::array<int, 3> colors = {{bg_color.r, bg_color.g, bg_color.b}};
00093     float sum = 0;
00094
00095     for (auto i = 0; i < 3; ++i) {
00096         float value = (float)colors.at(i) / 255.0F;
00097         if (value <= 0.04045) {
00098             value /= 12.92;
00099         } else {
00100             value = std::pow(((value + 0.055) / 1.055), 2.4);
00101         }
00102
00103         sum += value;
00104     }
00105
00106     return (sum > 0.179) ? BLACK : WHITE;
00107 }
00108
00109 }  // namespace utils
```

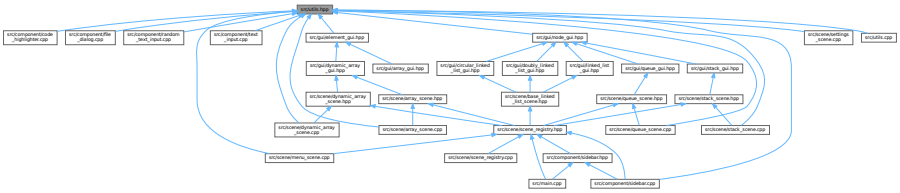
7.113 src/utils.hpp File Reference

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

Include dependency graph for utils.hpp:



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



Namespaces

- namespace utils

Functions

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

7.114 utils.hpp

Go to the documentation of this file.

```
00001 #ifndef UTILS_HPP_
00002 #define UTILS_HPP_
00003
00004 #include <cstring>
00005 #include <random>
00006
00007 #include "constants.hpp"
00008 #include "core/deque.hpp"
00009 #include "raylib.h"
00010
00011 namespace utils {
00012
00013 void DrawText(const char* text, Vector2 pos, Color color, float font_size,
00014               float spacing);
00015
00016 Vector2 MeasureText(const char* text, float font_size, float spacing);
00017
00018 template<typename T>
00019 T get_random(T low, T high) {
00020     if (low > high) {
00021         return low;
00022     }
00023
00024     static std::random_device ran_dev;
00025     static std::mt19937 prng(ran_dev());
00026     std::uniform_int_distribution<T> dist{low, high};
00027     return dist(prng);
00028 }
00029
00030 core::Deque<int> str_extract_data(
00031     char str[constants::text_buffer_size]);  // NOLINT
00032
00033 bool val_in_range(int num);
00034
00035 void unreachable();
00036
00037 char* strtok(char* str, const char* delim, char** save_ptr);
00038
00039 Color color_from_hex(const std::string& hex);
00040
00041 Color adaptive_text_color(Color bg_color);
00042
00043 }  // namespace utils
00044
00045 #endif  // UTILS_HPP_
```

Index