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 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

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_font_size

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

Definition at line 18 of file constants.hpp.

5.2.1.3 frames_per_second

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

Definition at line 8 of file constants.hpp.

5.2.1.4 max_val

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

Definition at line 16 of file constants.hpp.

5.2.1.5 min_val

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

Definition at line 15 of file constants.hpp.

5.2.1.6 scene_height

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

Definition at line 7 of file constants.hpp.

5.2.1.7 scene_width

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

Definition at line 6 of file constants.hpp.

5.2.1.8 sidebar_width

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

Definition at line 10 of file constants.hpp.

5.2.1.9 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 StackScene

Typedefs

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

Enumerations

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

5.6.1 Typedef Documentation

5.6.1.1 CircularLinkedListScene

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

Definition at line 108 of file base_linked_list_scene.hpp.

5.6.1.2 DoublyLinkedListScene

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

Definition at line 106 of file base_linked_list_scene.hpp.

5.6.1.3 LinkedListScene

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

Definition at line 105 of file base_linked_list_scene.hpp.

5.6.2 Enumeration Type Documentation

5.6.2.1 SceneId

```
enum scene::SceneId
```

Enumerator

Menu	
Array	
DynamicArray	
LinkedList	
DoublyLinkedList	
CircularLinkedList	
Stack	
Queue	

Definition at line 16 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)
- template<typename T >
  T get_random (T low, T high)

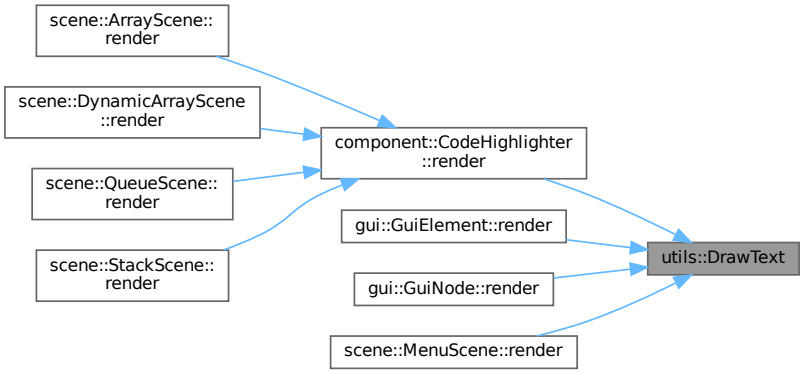
5.8.1 Function Documentation

5.8.1.1 DrawText()

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

Definition at line 10 of file utils.cpp.

Here is the caller graph for this function:



5.8.1.2 get_random()

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

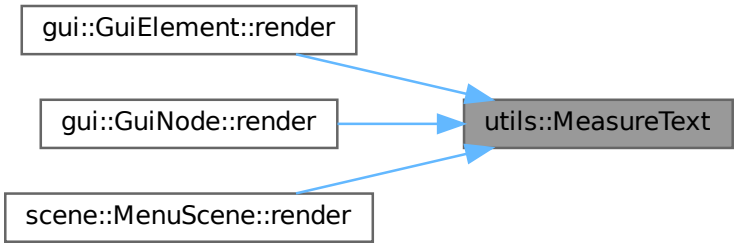
Definition at line 19 of file utils.hpp.

5.8.1.3 MeasureText()

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

Definition at line 19 of file utils.cpp.

Here is the caller graph for this function:

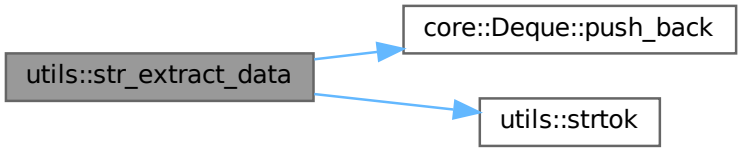


5.8.1.4 str_extract_data()

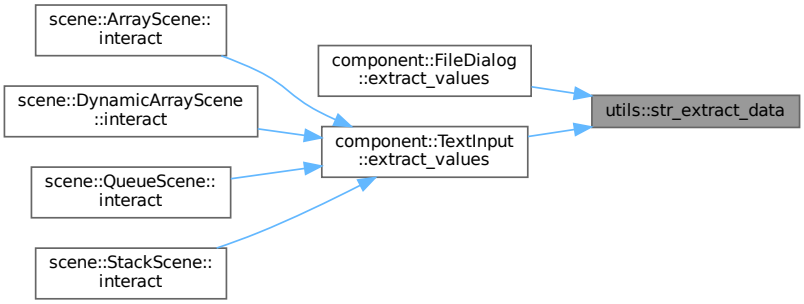
```
core::Deque< int > utils::str_extract_data (
            char str[constants::text_buffer_size] )
```

Definition at line 26 of file utils.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

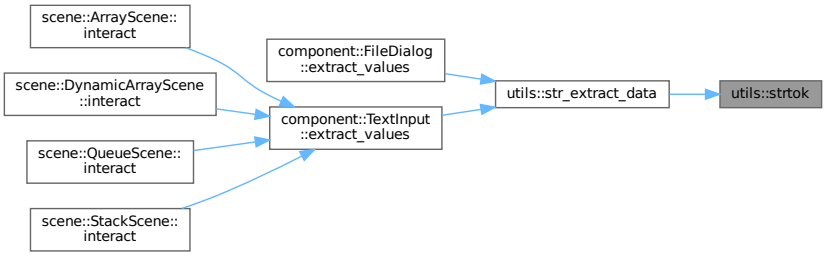


5.8.1.5 strtok()

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

Definition at line 69 of file utils.cpp.

Here is the caller graph for this function:

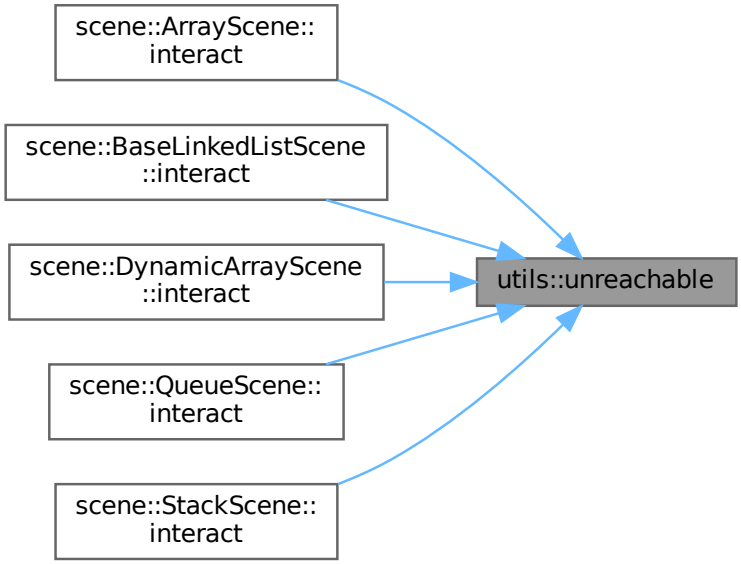


5.8.1.6 unreachable()

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

Definition at line 61 of file utils.cpp.

Here is the caller graph for this function:



5.8.1.7 val_in_range()

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

Definition at line 57 of file utils.cpp.

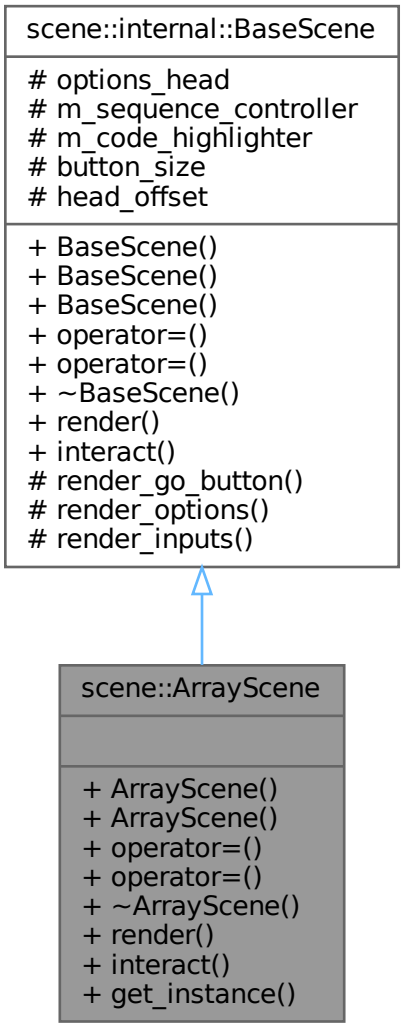
Chapter 6

Class Documentation

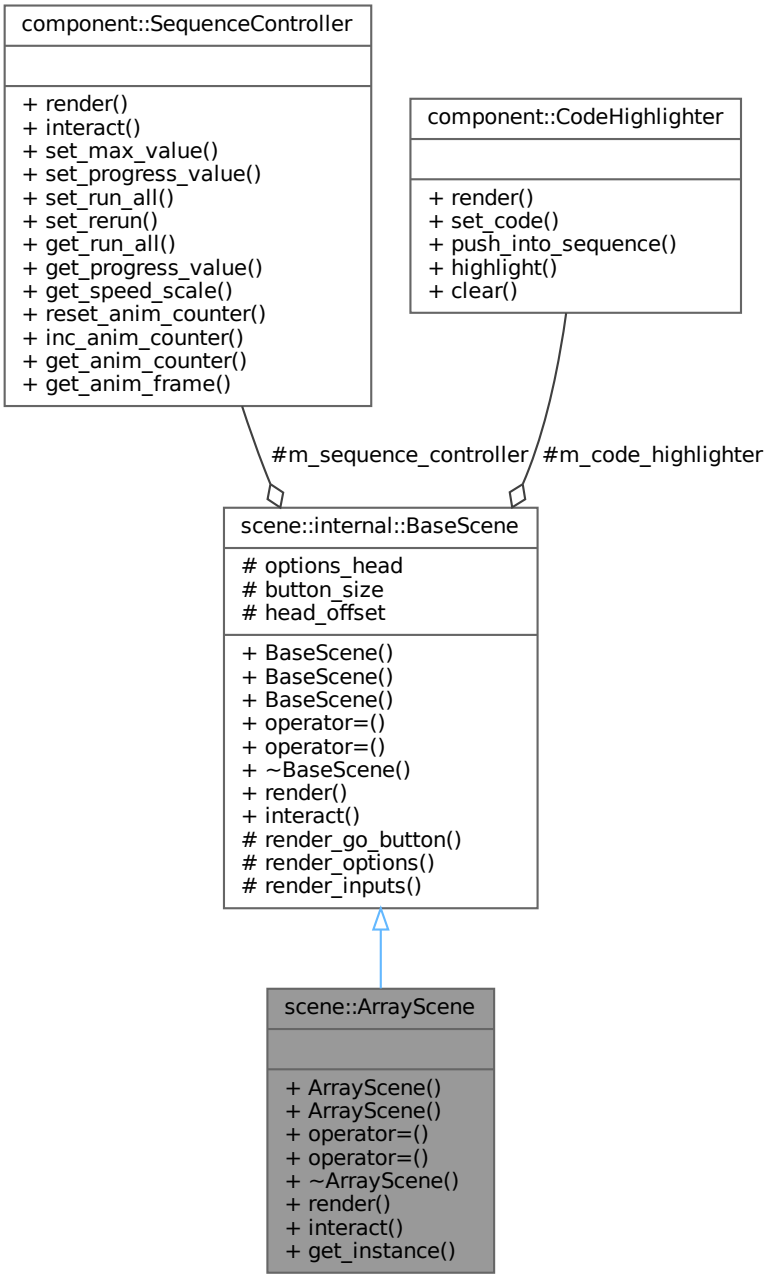
6.1 scene::ArrayScene Class Reference

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

Inheritance diagram for scene::ArrayScene:



Collaboration diagram for scene::ArrayScene:



Public Member Functions

- ArrayScene (const ArrayScene &)=delete
- ArrayScene (ArrayScene &&)=delete
- ArrayScene & operator= (const ArrayScene &)=delete
- ArrayScene & operator= (ArrayScene &&)=delete
- ~ArrayScene () override=default
- 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 ()

Static Public Member Functions

- static ArrayScene & get_instance ()

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::SequenceController m_sequence_controller
- component::CodeHighlighter m_code_highlighter

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

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

6.1.1 Detailed Description

Definition at line 18 of file array_scene.hpp.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 ArrayScene() [1/2]

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

6.1.2.2 ArrayScene() [2/2]

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

6.1.2.3 ~ArrayScene()

```
scene::ArrayScene::~ArrayScene ( )  [override], [default]
```

6.1.3 Member Function Documentation

6.1.3.1 get_instance()

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

Definition at line 17 of file array_scene.cpp.

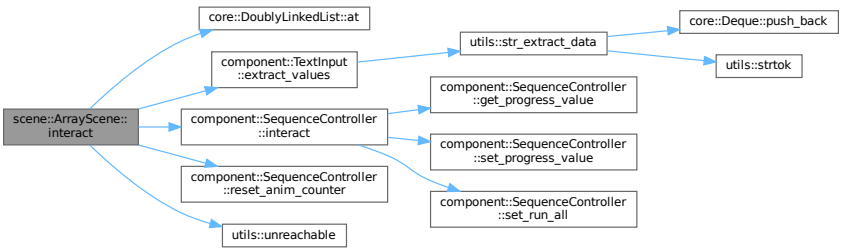
6.1.3.2 interact()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 77 of file array_scene.cpp.

Here is the call graph for this function:



6.1.3.3 operator=() [1/2]

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

6.1.3.4 operator=() [2/2]

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

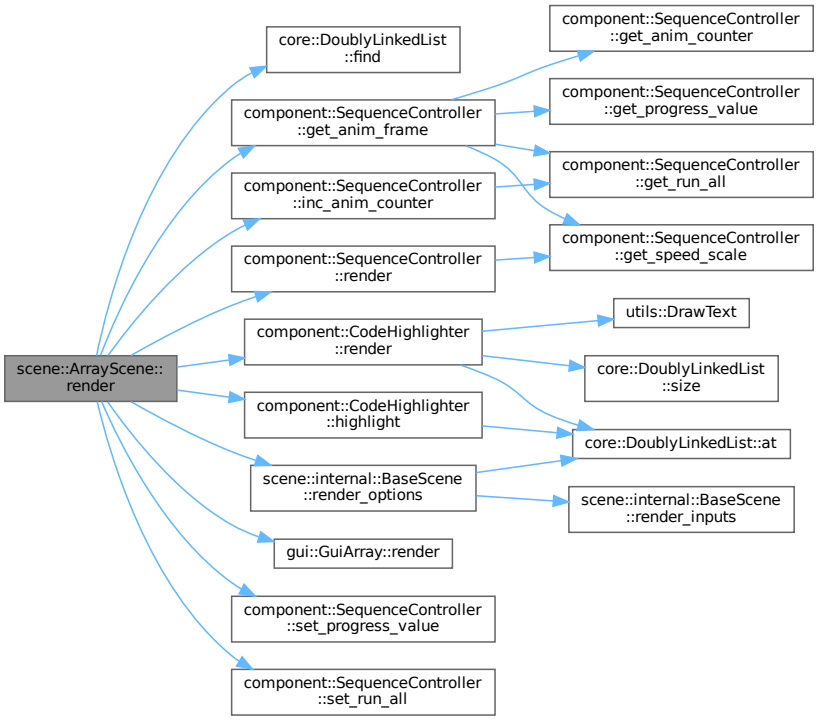
6.1.3.5 render()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 57 of file array_scene.cpp.

Here is the call graph for this function:



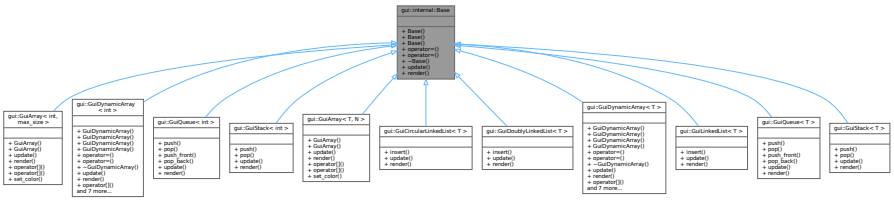
The documentation for this class was generated from the following files:

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

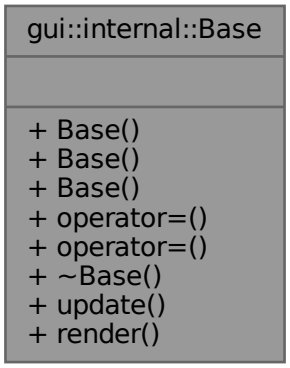
6.2 gui::internal::Base Class Reference

`#include <base_gui.hpp>`

Inheritance diagram for gui::internal::Base:



Collaboration diagram for gui::internal::Base:



Public Member Functions

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

6.2.1 Detailed Description

Definition at line 8 of file base_gui.hpp.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Base() [1/3]

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

6.2.2.2 Base() [2/3]

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

6.2.2.3 Base() [3/3]

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

6.2.2.4 ~Base()

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

6.2.3 Member Function Documentation

6.2.3.1 operator=() [1/2]

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

6.2.3.2 operator=() [2/2]

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

6.2.3.3 render()

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

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

6.2.3.4 update()

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

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

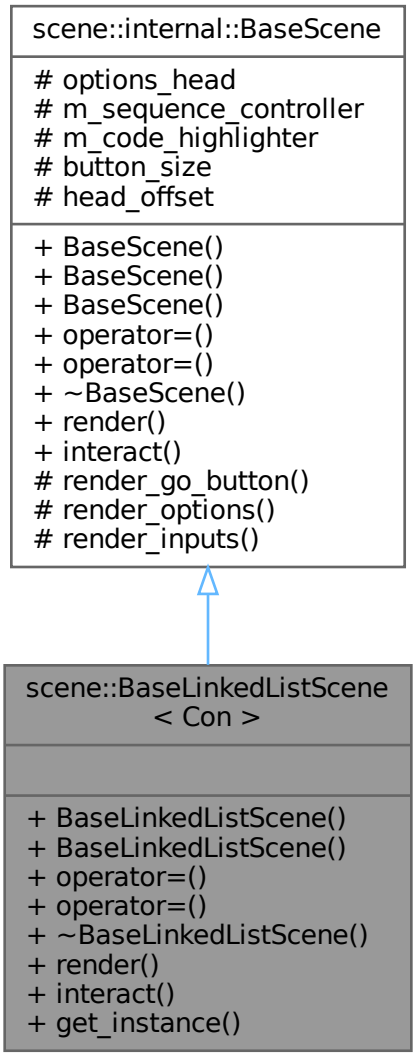
The documentation for this class was generated from the following file:

- src/gui/base_gui.hpp

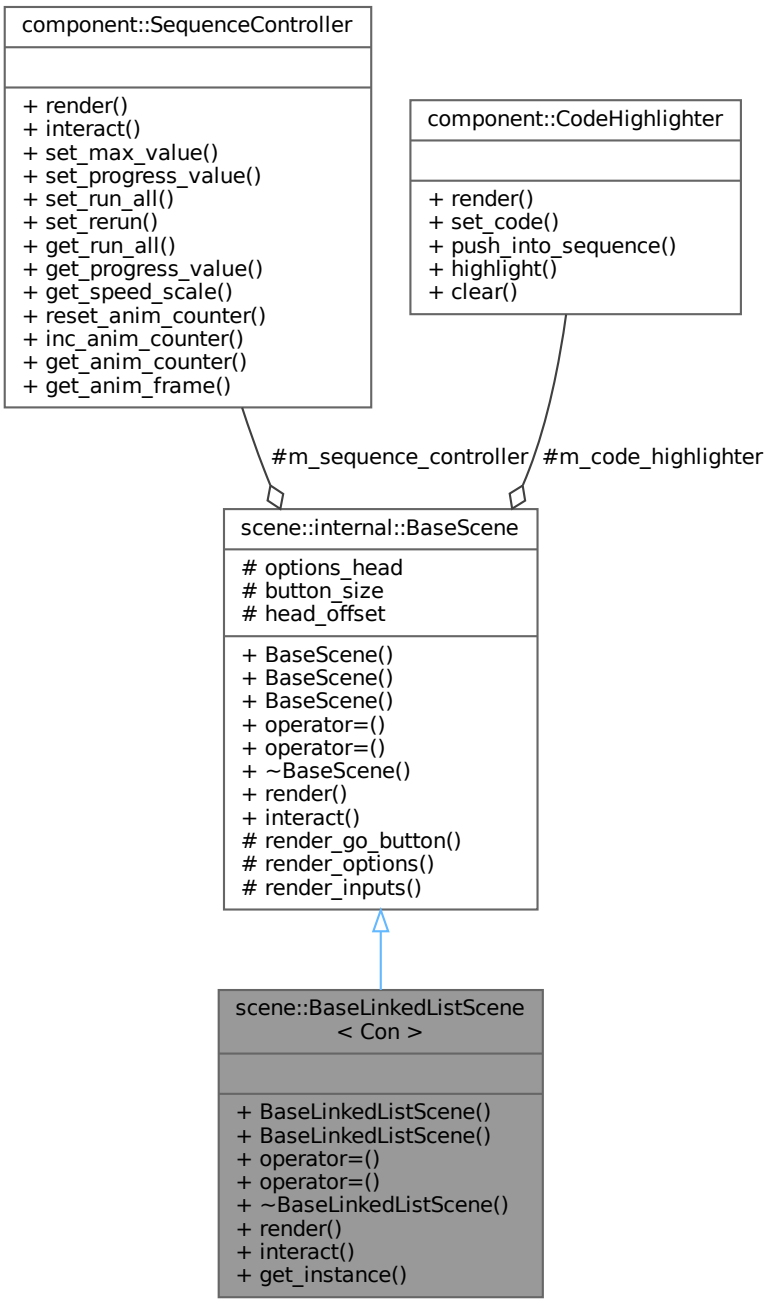
6.3 scene::BaseLinkedListScene< Con > Class Template Reference

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

Inheritance diagram for scene::BaseLinkedListScene< Con >:



Collaboration diagram for scene::BaseLinkedListScene< Con >:



Public Member Functions

- BaseLinkedListScene (const BaseLinkedListScene &)=delete
- BaseLinkedListScene (BaseLinkedListScene &&)=delete
- BaseLinkedListScene & operator= (const BaseLinkedListScene &)=delete
- BaseLinkedListScene & operator= (BaseLinkedListScene &&)=delete
- ~BaseLinkedListScene () override=default

- 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 ()

Static Public Member Functions

- static BaseLinkedListScene & get_instance ()

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::SequenceController m_sequence_controller
- component::CodeHighlighter m_code_highlighter

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

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

6.3.1 Detailed Description

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

Definition at line 17 of file base_linked_list_scene.hpp.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 BaseLinkedListScene() [1/2]

```
template<typename Con >
scene::BaseLinkedListScene< Con >::BaseLinkedListScene (
            const BaseLinkedListScene< Con > &  )  [delete]
```

6.3.2.2 BaseLinkedListScene() [2/2]

```
template<typename Con >
scene::BaseLinkedListScene< Con >::BaseLinkedListScene (
            BaseLinkedListScene< Con > &&  )  [delete]
```

6.3.2.3 ~BaseLinkedListScene()

```
template<typename Con >
scene::BaseLinkedListScene< Con >::~BaseLinkedListScene ( )  [override], [default]
```

6.3.3 Member Function Documentation

6.3.3.1 get_instance()

```
template<typename Con >
BaseLinkedListScene< Con > & scene::BaseLinkedListScene< Con >::get_instance  [static]
```

Definition at line 112 of file base_linked_list_scene.hpp.

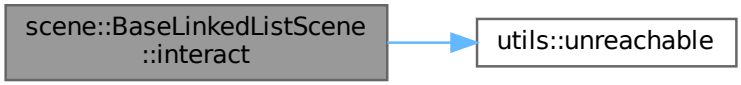
6.3.3.2 interact()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 184 of file base_linked_list_scene.hpp.

Here is the call graph for this function:



6.3.3.3 operator=() [1/2]

```
template<typename Con >
BaseLinkedListScene & scene::BaseLinkedListScene< Con >::operator= (
            BaseLinkedListScene< Con > && ) [delete]
```

6.3.3.4 operator=() [2/2]

```
template<typename Con >
BaseLinkedListScene & scene::BaseLinkedListScene< Con >::operator= (
            const BaseLinkedListScene< Con > & ) [delete]
```

6.3.3.5 render()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 163 of file base_linked_list_scene.hpp.

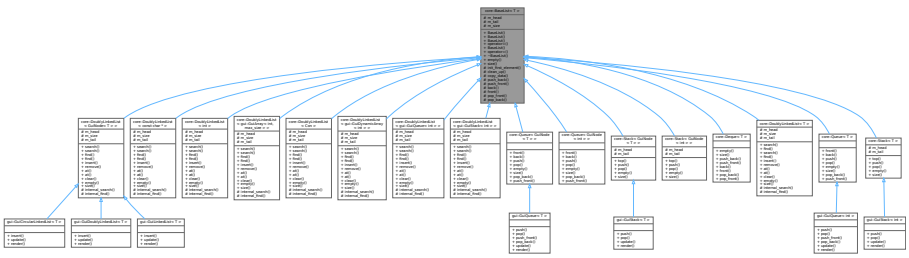
The documentation for this class was generated from the following file:

- src/scene/base_linked_list_scene.hpp

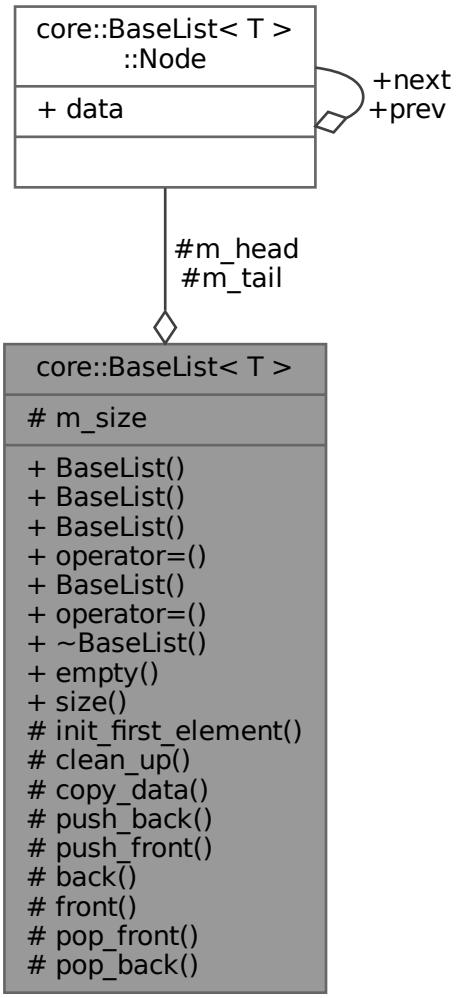
6.4 core::BaseList< T > Class Template Reference

```
#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.

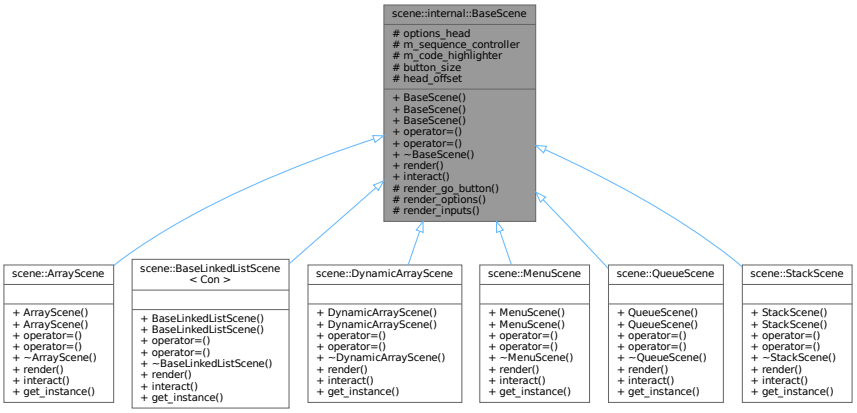
The documentation for this class was generated from the following file:

- src/core/base_list.hpp

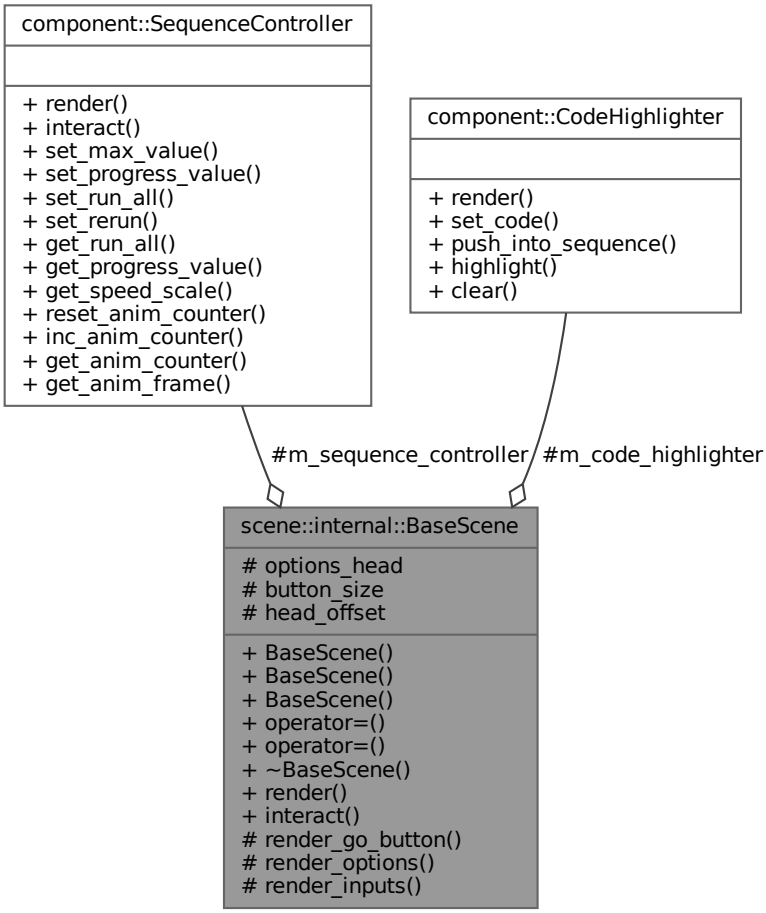
6.5 scene::internal::BaseScene Class Reference

```
#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::SequenceController m_sequence_controller
- component::CodeHighlighter m_code_highlighter

Static Protected Attributes

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

6.5.1 Detailed Description

Definition at line 11 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

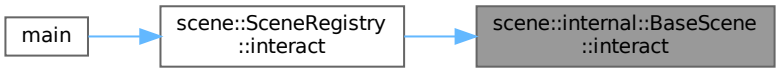
6.5.3.1 interact()

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

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

Definition at line 34 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]
```

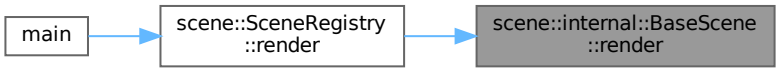
6.5.3.4 render()

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

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

Definition at line 33 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]
```

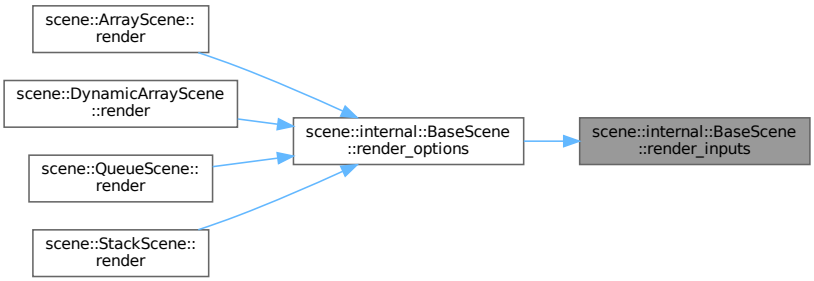
Definition at line 10 of file base_scene.cpp.

6.5.3.6 render_inputs()

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

Definition at line 19 of file base_scene.hpp.

Here is the caller graph for this function:

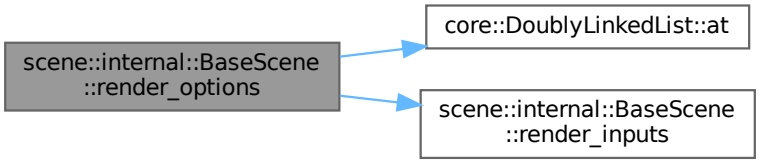


6.5.3.7 render_options()

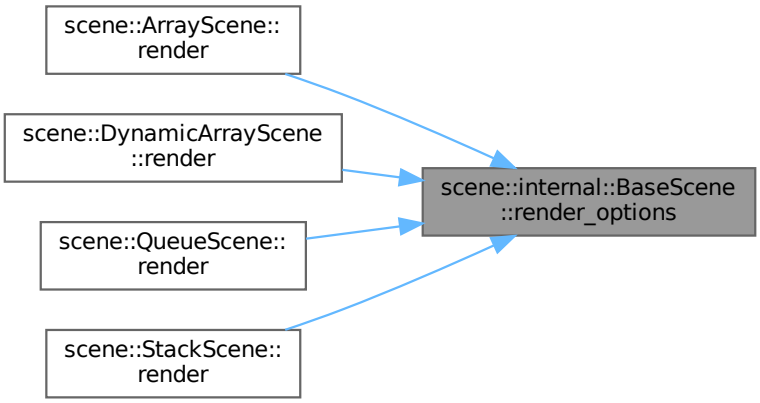
```
void scene::internal::BaseScene::render_options (
            SceneOptions & scene_config )  [protected], [virtual]
```

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 13 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 14 of file base_scene.hpp.

6.5.4.3 m_code_highlighter

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

Definition at line 22 of file base_scene.hpp.

6.5.4.4 m_sequence_controller

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

Definition at line 21 of file base_scene.hpp.

6.5.4.5 options_head

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

Definition at line 15 of file base_scene.hpp.

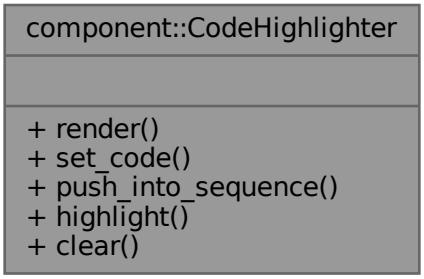
The documentation for this class was generated from the following files:

- 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.

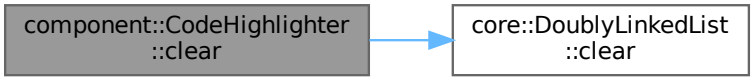
6.6.2 Member Function Documentation

6.6.2.1 clear()

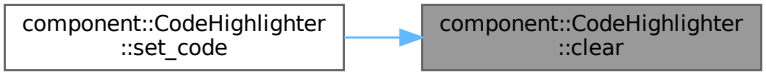
```
void component::CodeHighlighter::clear ( )
```

Definition at line 32 of file code_highlighter.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

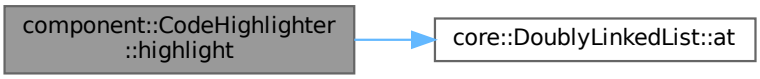


6.6.2.2 highlight()

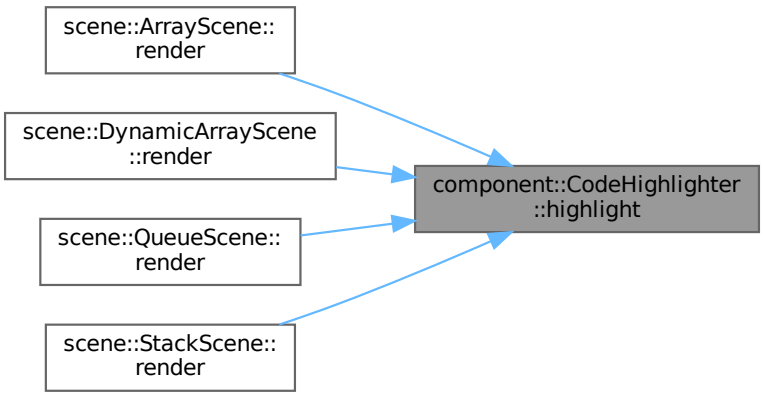
```
void component::CodeHighlighter::highlight (
            int frame_idx )
```

Definition at line 28 of file code_highlighter.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

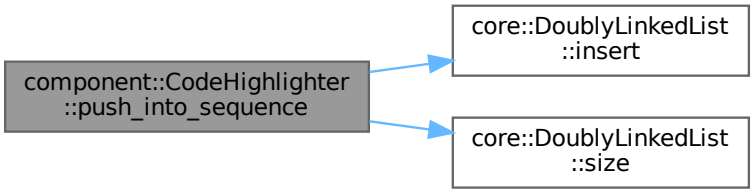


6.6.2.3 push_into_sequence()

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

Definition at line 24 of file code_highlighter.cpp.

Here is the call graph for this function:

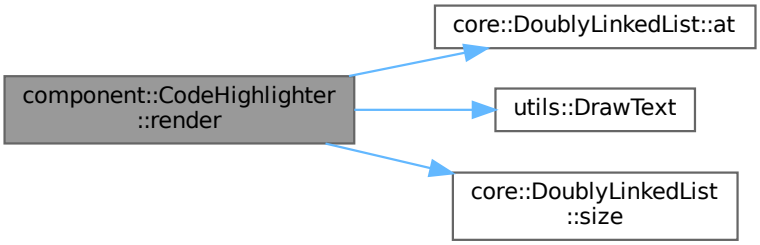


6.6.2.4 render()

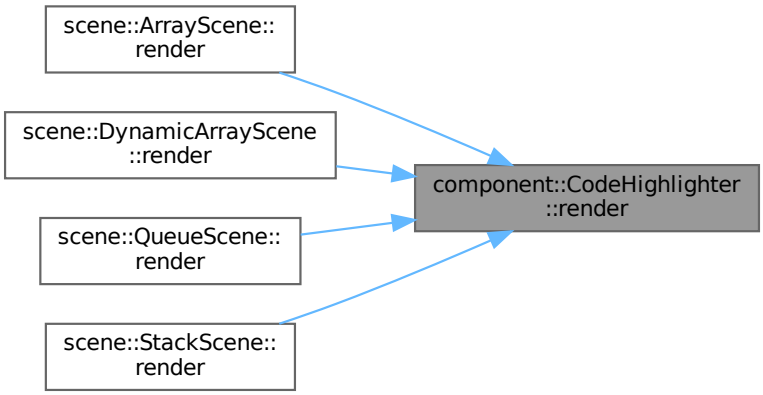
```
void component::CodeHighlighter::render ( )
```

Definition at line 8 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 19 of file code_highlighter.cpp.

Here is the call graph for this function:

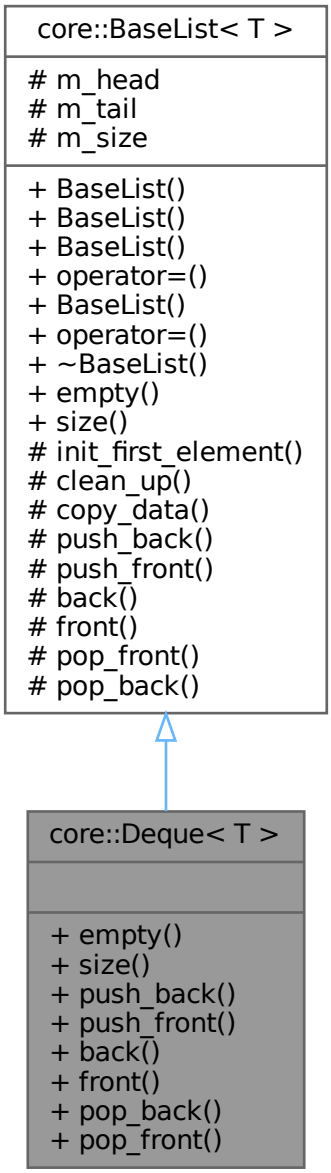
The documentation for this class was generated from the following files:

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

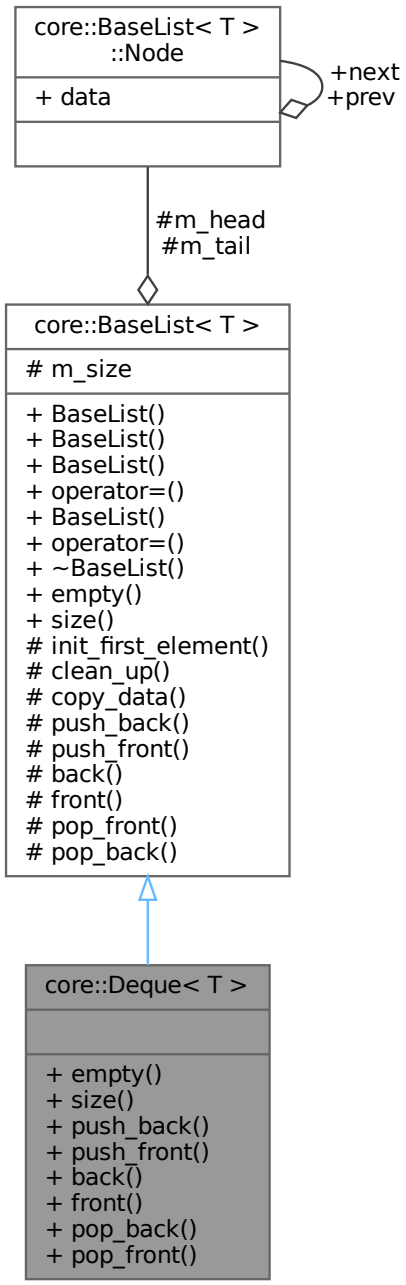
6.7 core::Deque< T > Class Template Reference

`#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.

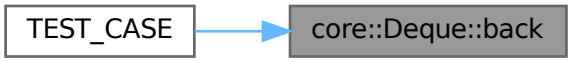
6.7.2 Member Function Documentation

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:

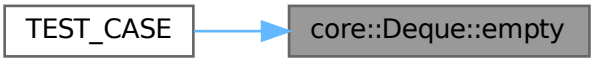


6.7.2.2 empty()

```
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:

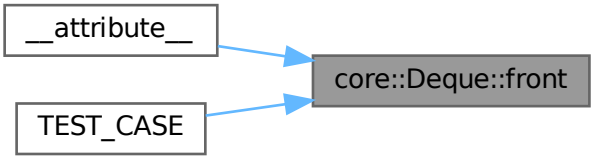


6.7.2.3 front()

```
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.

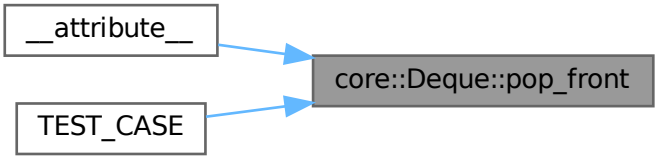
Here is the caller graph for this function:

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:

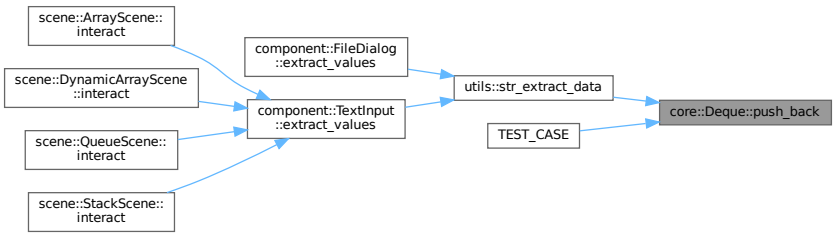


6.7.2.6 push_back()

```
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.

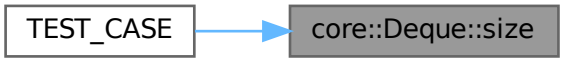
Here is the caller graph for this function:

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:



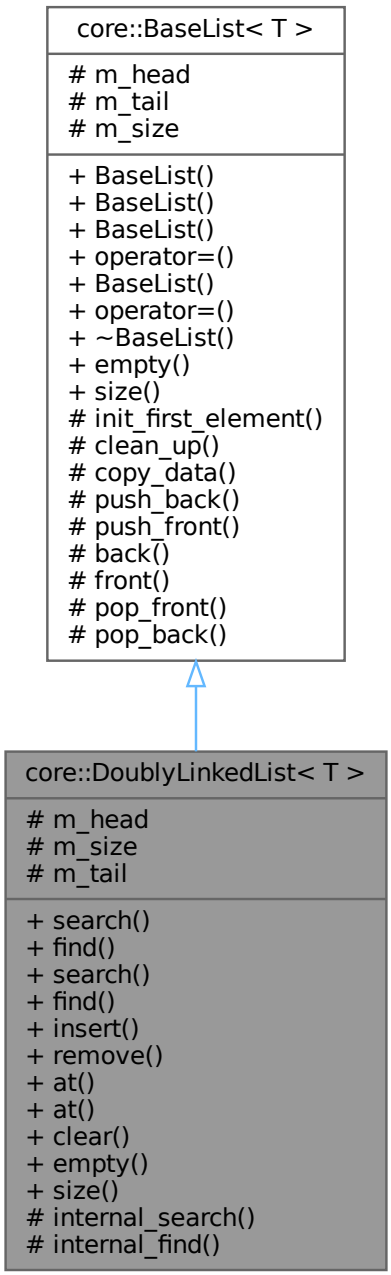
The documentation for this class was generated from the following file:

- src/core/deque.hpp

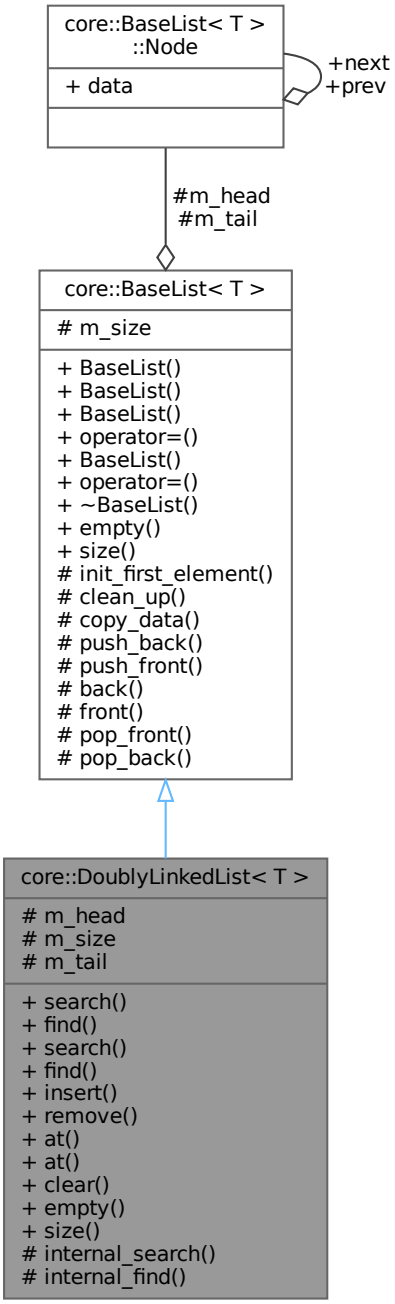
6.8 core::DoublyLinkedList< T > Class Template Reference

`#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.

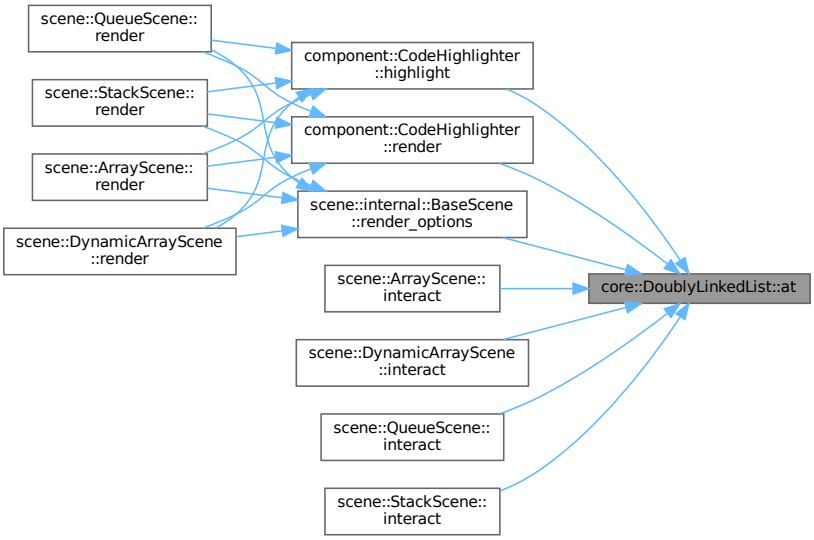
6.8.3 Member Function Documentation

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
```

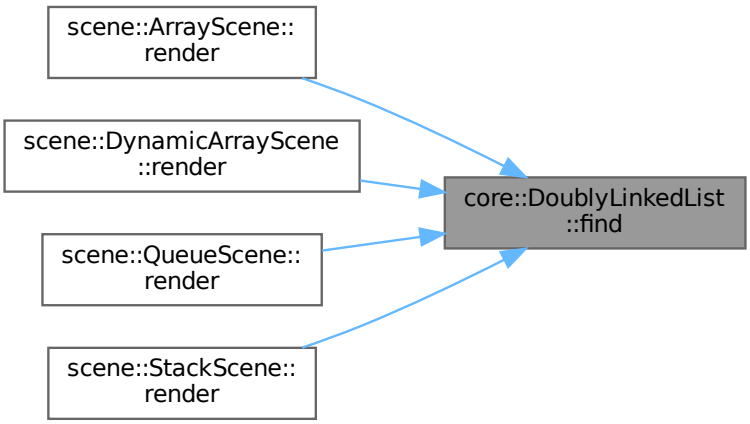
Definition at line 48 of file base_list.hpp.

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
```

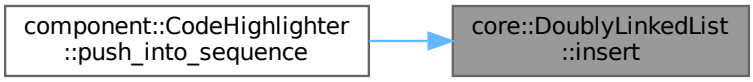
Definition at line 95 of file doubly_linked_list.hpp.

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 )
```

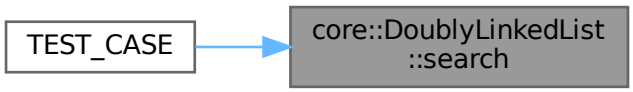
Definition at line 124 of file doubly_linked_list.hpp.

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
```

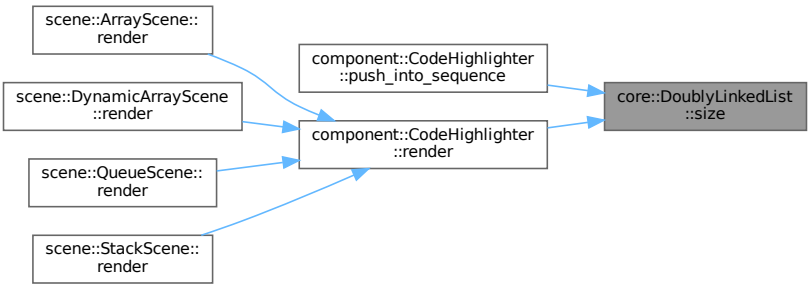
Definition at line 89 of file doubly_linked_list.hpp.

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.

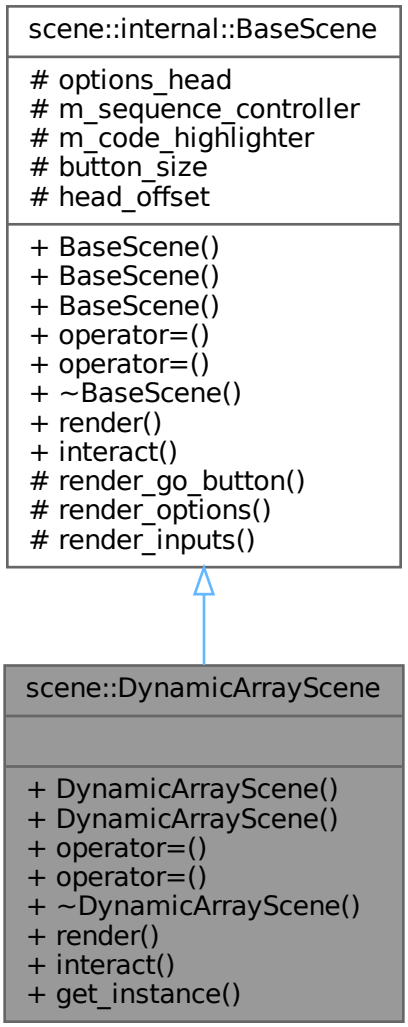
The documentation for this class was generated from the following file:

- src/core/doubly_linked_list.hpp

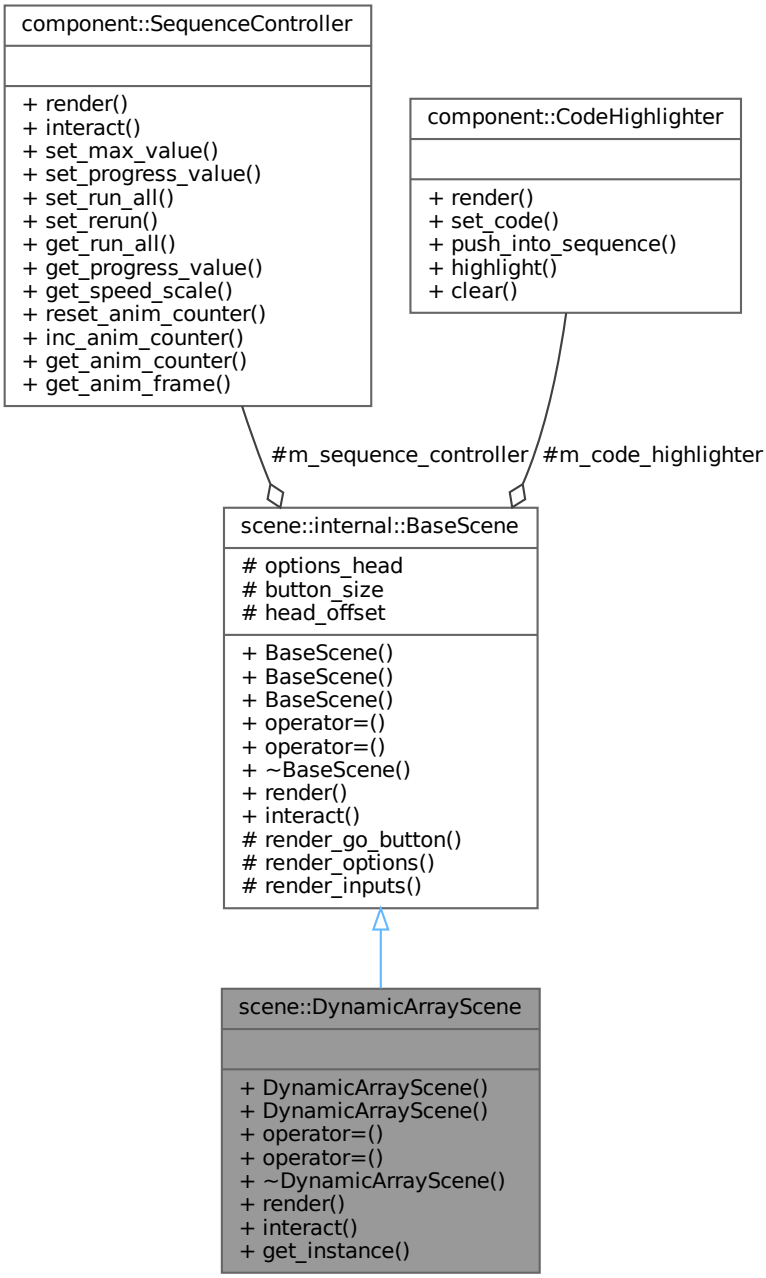
6.9 scene::DynamicArrayScene Class Reference

`#include <dynamic_array_scene.hpp>`

Inheritance diagram for scene::DynamicArrayScene:



Collaboration diagram for scene::DynamicArrayScene:



Public Member Functions

- DynamicArrayScene (const DynamicArrayScene &)=delete
- DynamicArrayScene (DynamicArrayScene &&)=delete
- DynamicArrayScene & operator= (const DynamicArrayScene &)=delete
- DynamicArrayScene & operator= (DynamicArrayScene &&)=delete
- ~DynamicArrayScene () override=default
- 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 ()

Static Public Member Functions

- static DynamicArrayScene & get_instance ()

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::SequenceController m_sequence_controller
- component::CodeHighlighter m_code_highlighter

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

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

6.9.1 Detailed Description

Definition at line 18 of file dynamic_array_scene.hpp.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 DynamicArrayScene() [1/2]

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

6.9.2.2 DynamicArrayScene() [2/2]

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

6.9.2.3 ~DynamicArrayScene()

```
scene::DynamicArrayScene::~DynamicArrayScene ( )  [override], [default]
```

6.9.3 Member Function Documentation

6.9.3.1 get_instance()

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

Definition at line 17 of file dynamic_array_scene.cpp.

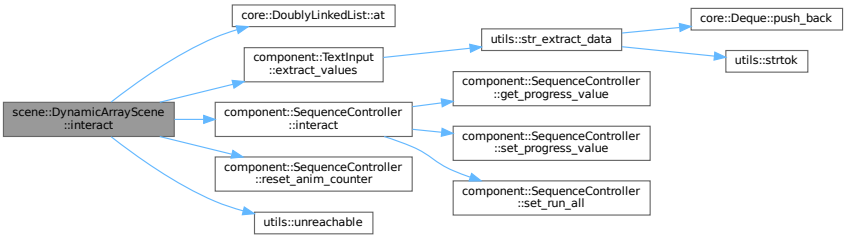
6.9.3.2 interact()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 81 of file dynamic_array_scene.cpp.

Here is the call graph for this function:



6.9.3.3 operator=() [1/2]

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

6.9.3.4 operator=() [2/2]

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

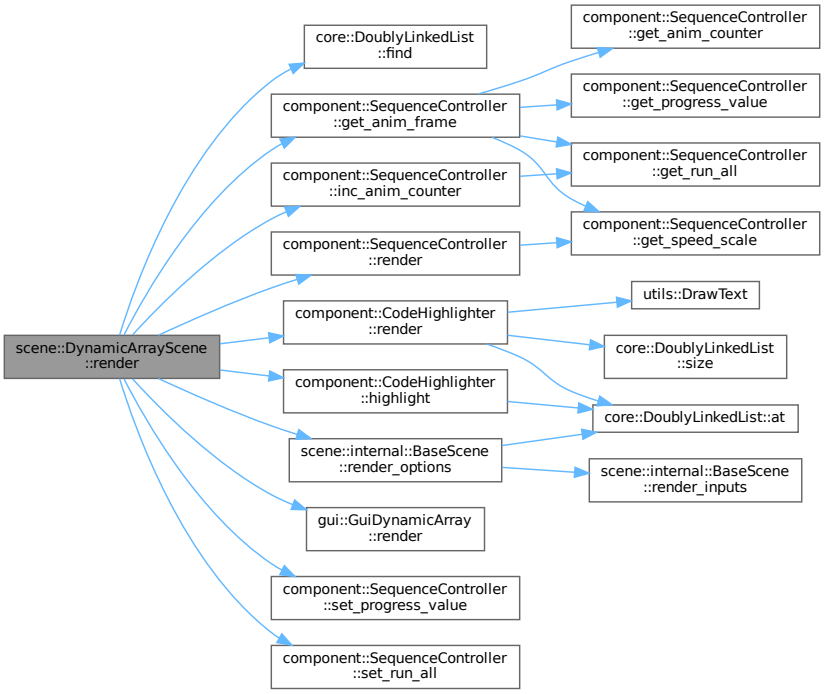
6.9.3.5 render()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 61 of file dynamic_array_scene.cpp.

Here is the call graph for this function:



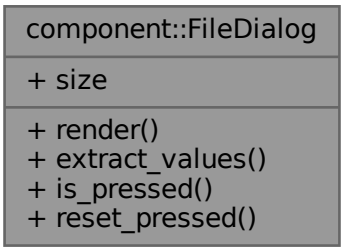
The documentation for this class was generated from the following files:

- 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

- void render (float &options_head, float head_offset)
- core::Deque< int > extract_values ()
- bool is_pressed () const
- void reset_pressed ()

Static Public Attributes

- static constexpr Vector2 size {200, 50}

6.10.1 Detailed Description

Definition at line 11 of file file_dialog.hpp.

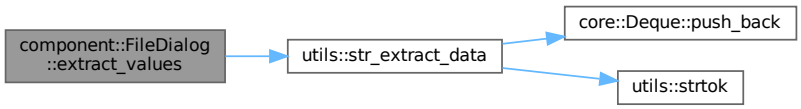
6.10.2 Member Function Documentation

6.10.2.1 extract_values()

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

Definition at line 36 of file file_dialog.cpp.

Here is the call graph for this function:



6.10.2.2 is_pressed()

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

Definition at line 49 of file file_dialog.cpp.

6.10.2.3 render()

```
void component::FileDialog::render (
            float & options_head,
            float head_offset )
```

Definition at line 12 of file file_dialog.cpp.

6.10.2.4 reset_pressed()

```
void component::FileDialog::reset_pressed ( )
```

Definition at line 53 of file file_dialog.cpp.

6.10.3 Member Data Documentation

6.10.3.1 size

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

Definition at line 19 of file file_dialog.hpp.

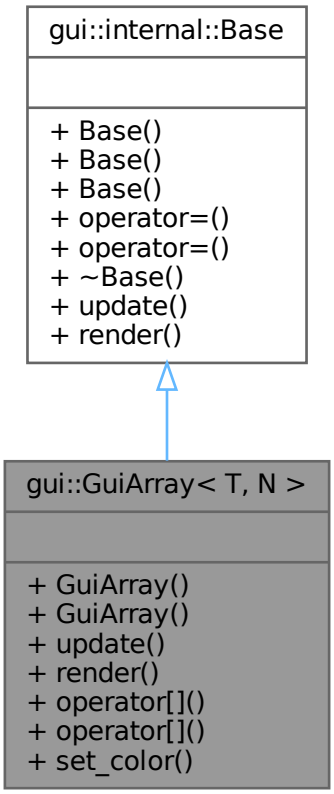
The documentation for this class was generated from the following files:

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

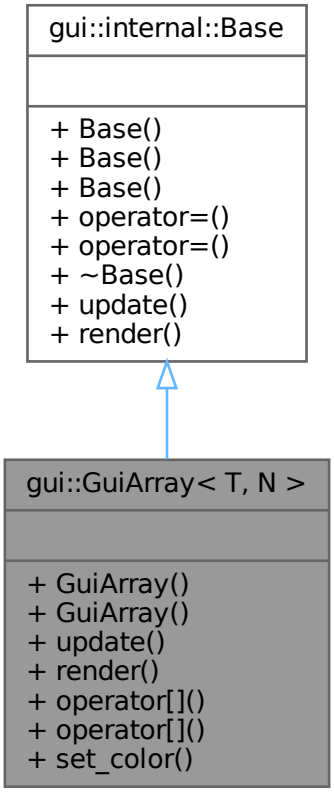
6.11 gui::GuiArray< T, N > Class Template Reference

```
#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 (std::size_t idx, Color color)

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 15 of file array_gui.hpp.

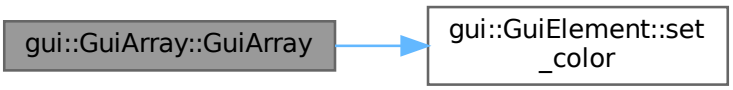
6.11.2 Constructor & Destructor Documentation

6.11.2.1 GuiArray() [1/2]

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

Definition at line 40 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 48 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 74 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 79 of file array_gui.hpp.

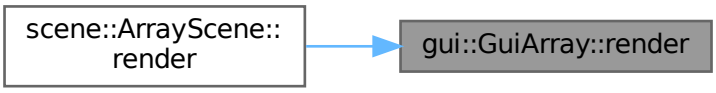
6.11.3.3 render()

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

Implements gui::internal::Base.

Definition at line 55 of file array_gui.hpp.

Here is the caller graph for this function:



6.11.3.4 set_color()

```
template<typename T , std::size_t N>
void gui::GuiArray< T, N >::set_color (
            std::size_t idx,
            Color color )
```

Definition at line 84 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 64 of file array_gui.hpp.

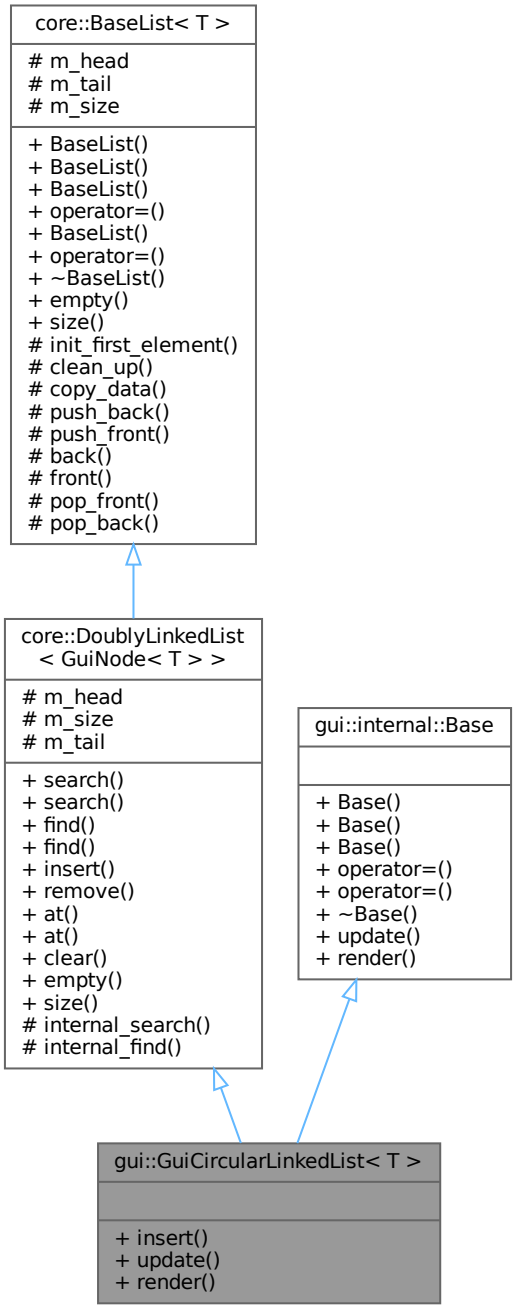
The documentation for this class was generated from the following file:

- src/gui/array_gui.hpp

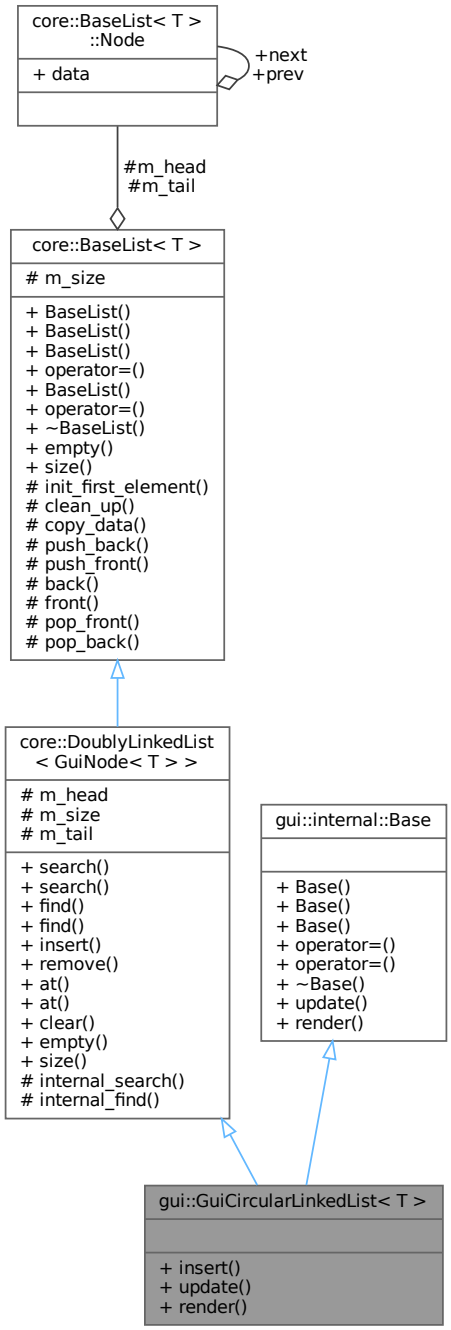
6.12 gui::GuiCircularLinkedList< T > Class Template Reference

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

Inheritance diagram for gui::GuiCircularLinkedList< T >:



Collaboration diagram for gui::GuiCircularLinkedList< T >:



Public Member Functions

- void insert (std::size_t index, const T &elem)
- void update () override
- void render () override

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 18 of file circular_linked_list_gui.hpp.

6.12.2 Member Function Documentation

6.12.2.1 insert()

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

Definition at line 48 of file circular_linked_list_gui.hpp.

6.12.2.2 render()

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

Implements gui::internal::Base.

Definition at line 99 of file circular_linked_list_gui.hpp.

6.12.2.3 update()

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

Implements gui::internal::Base.

Definition at line 114 of file circular_linked_list_gui.hpp.

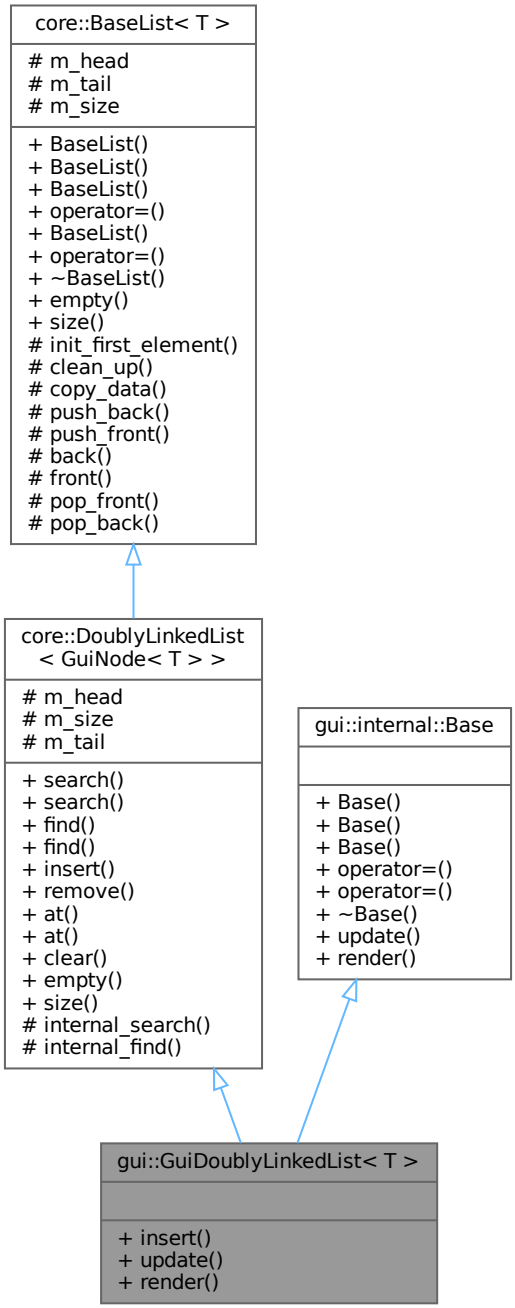
The documentation for this class was generated from the following file:

- src/gui/circular_linked_list_gui.hpp

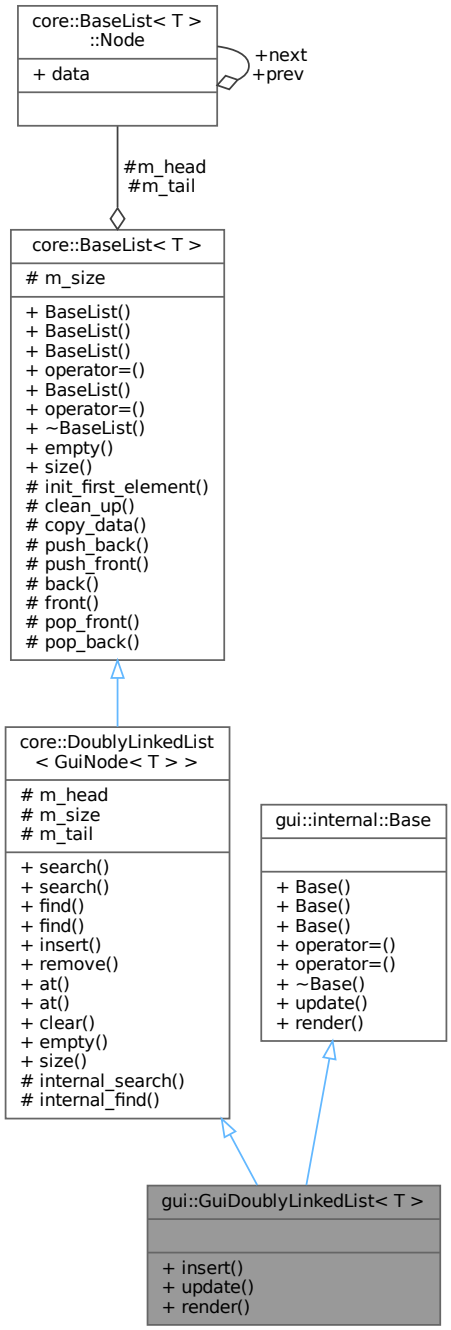
6.13 gui::GuiDoublyLinkedList< T > Class Template Reference

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

Inheritance diagram for gui::GuiDoublyLinkedList< T >:



Collaboration diagram for gui::GuiDoublyLinkedList< T >:



Public Member Functions

- void insert (std::size_t index, const T &elem)
- void update () override
- void render () override

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 16 of file doubly_linked_list_gui.hpp.

6.13.2 Member Function Documentation

6.13.2.1 insert()

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

Definition at line 45 of file doubly_linked_list_gui.hpp.

6.13.2.2 render()

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

Implements gui::internal::Base.

Definition at line 78 of file doubly_linked_list_gui.hpp.

6.13.2.3 update()

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

Implements gui::internal::Base.

Definition at line 92 of file doubly_linked_list_gui.hpp.

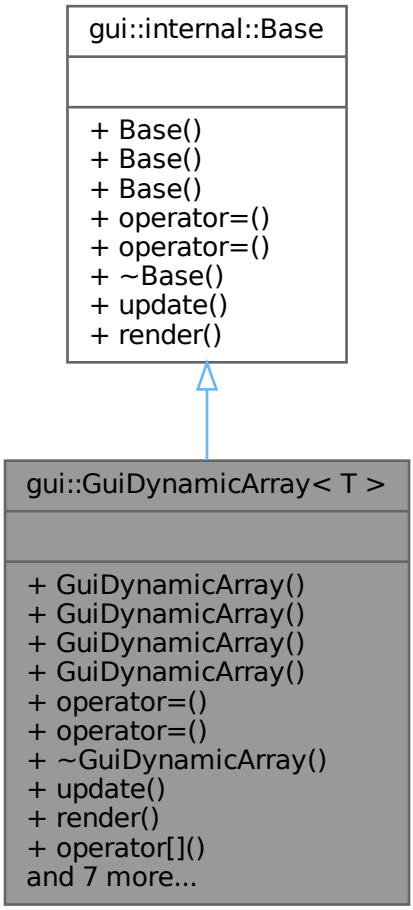
The documentation for this class was generated from the following file:

- src/gui/doubly_linked_list_gui.hpp

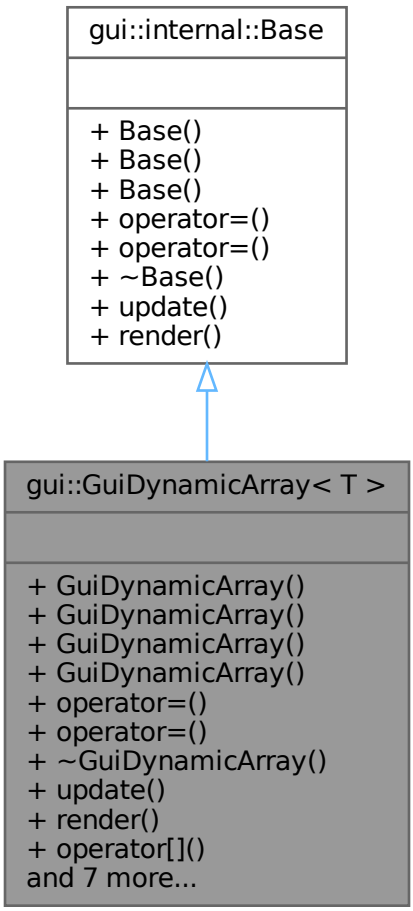
6.14 gui::GuiDynamicArray< T > Class Template Reference

```
#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 (std::size_t idx, Color color)
- void realloc (std::size_t capacity)
- std::size_t capacity () const
- std::size_t size () const
- void push (const T &value)
- void pop ()

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

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

6.14.1 Detailed Description

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

Definition at line 16 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
```

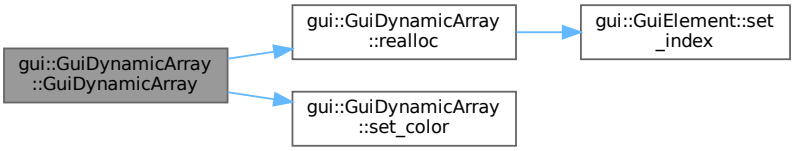
Definition at line 78 of file dynamic_array_gui.hpp.

6.14.2.2 GuiDynamicArray() [2/4]

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

Definition at line 85 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 96 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 106 of file dynamic_array_gui.hpp.

6.14.2.5 ~GuiDynamicArray()

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

Definition at line 144 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 188 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 114 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 130 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 173 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 178 of file dynamic_array_gui.hpp.

6.14.3.6 pop()

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

Definition at line 209 of file dynamic_array_gui.hpp.

6.14.3.7 push()

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

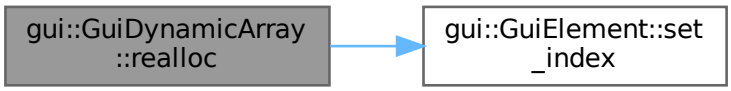
Definition at line 198 of file dynamic_array_gui.hpp.

6.14.3.8 realloc()

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

Definition at line 56 of file dynamic_array_gui.hpp.

Here is the call graph for this function:



Here is the caller graph for this function:

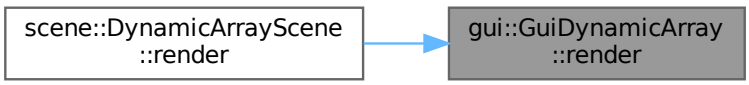
6.14.3.9 render()

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

Implements gui::internal::Base.

Definition at line 152 of file dynamic_array_gui.hpp.

Here is the caller graph for this function:

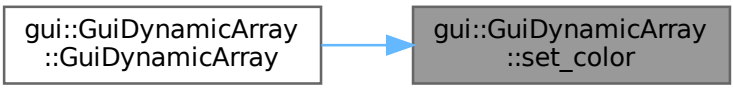


6.14.3.10 set_color()

```
template<typename T >
void gui::GuiDynamicArray< T >::set_color (
            std::size_t idx,
            Color color )
```

Definition at line 183 of file dynamic_array_gui.hpp.

Here is the caller graph for this function:



6.14.3.11 size()

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

Definition at line 193 of file dynamic_array_gui.hpp.

6.14.3.12 update()

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

Implements gui::internal::Base.

Definition at line 163 of file dynamic_array_gui.hpp.

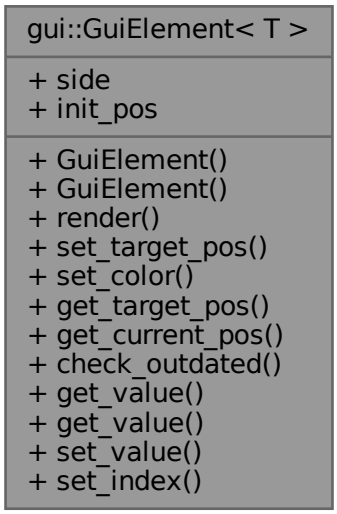
The documentation for this class was generated from the following file:

- 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_target_pos (Vector2 pos)
- void set_color (Color color)
- Vector2 get_target_pos () const
- Vector2 get_current_pos () const
- bool check_outdated () 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 16 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 53 of file element_gui.hpp.

6.15.3 Member Function Documentation

6.15.3.1 check_outdated()

```
template<typename T >
bool gui::GuiElement< T >::check_outdated
```

Definition at line 131 of file element_gui.hpp.

6.15.3.2 get_current_pos()

```
template<typename T >
Vector2 gui::GuiElement< T >::get_current_pos
```

Definition at line 126 of file element_gui.hpp.

6.15.3.3 get_target_pos()

```
template<typename T >
Vector2 gui::GuiElement< T >::get_target_pos
```

Definition at line 121 of file element_gui.hpp.

6.15.3.4 get_value() [1/2]

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

Definition at line 136 of file element_gui.hpp.

6.15.3.5 get_value() [2/2]

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

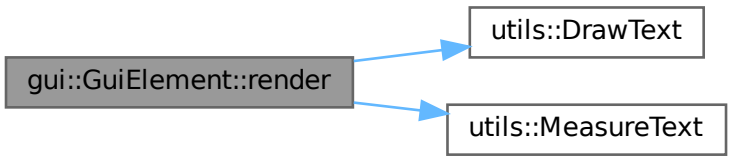
Definition at line 141 of file element_gui.hpp.

6.15.3.6 render()

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

Definition at line 57 of file element_gui.hpp.

Here is the call graph for this function:

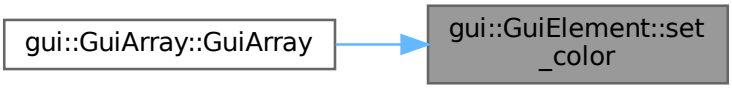


6.15.3.7 set_color()

```
template<typename T >
void gui::GuiElement< T >::set_color (
            Color color )
```

Definition at line 116 of file element_gui.hpp.

Here is the caller graph for this function:

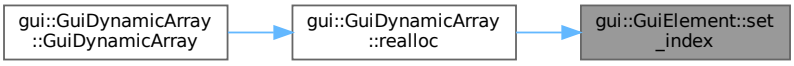


6.15.3.8 set_index()

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

Definition at line 151 of file element_gui.hpp.

Here is the caller graph for this function:



6.15.3.9 set_target_pos()

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

Definition at line 109 of file element_gui.hpp.

6.15.3.10 set_value()

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

Definition at line 146 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 29 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 28 of file element_gui.hpp.

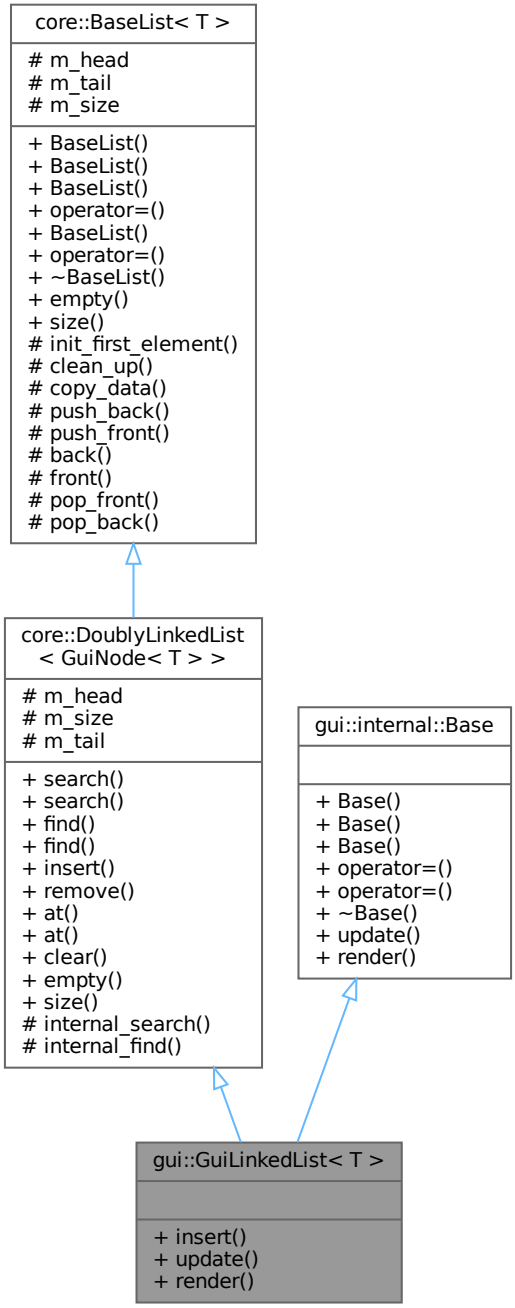
The documentation for this class was generated from the following file:

- src/gui/element_gui.hpp

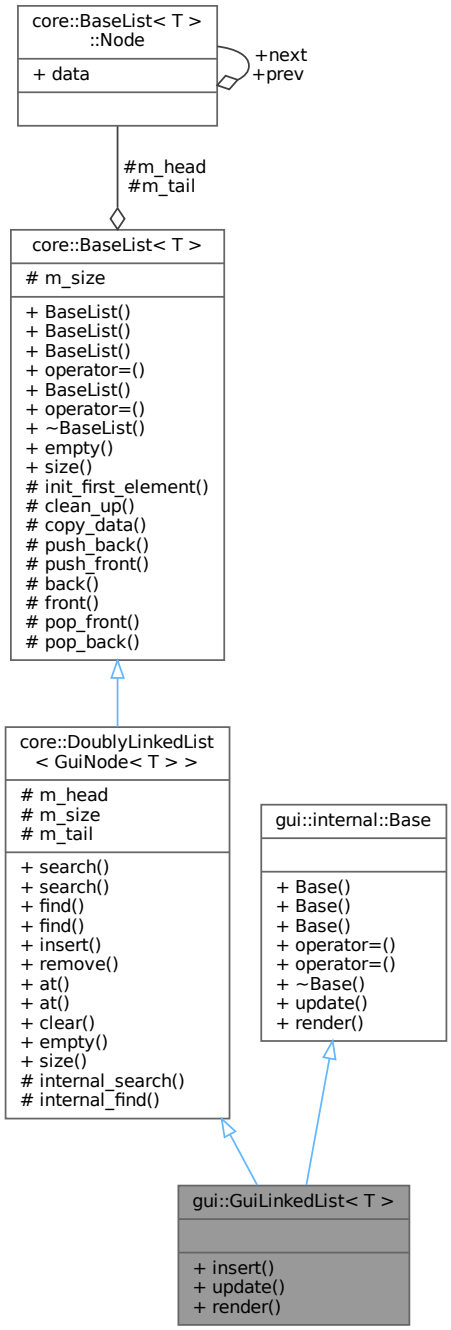
6.16 gui::GuiLinkedList< T > Class Template Reference

`#include <linked_list_gui.hpp>`

Inheritance diagram for gui::GuiLinkedList< T >:



Collaboration diagram for gui::GuiLinkedList< T >:



Public Member Functions

- void insert (std::size_t index, const T &elem)
- void update () override
- void render () override

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 16 of file linked_list_gui.hpp.

6.16.2 Member Function Documentation

6.16.2.1 insert()

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

Definition at line 45 of file linked_list_gui.hpp.

6.16.2.2 render()

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

Implements gui::internal::Base.

Definition at line 70 of file linked_list_gui.hpp.

6.16.2.3 update()

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

Implements gui::internal::Base.

Definition at line 84 of file linked_list_gui.hpp.

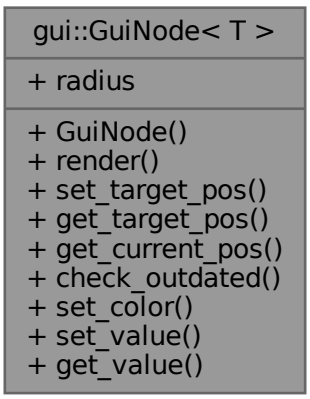
The documentation for this class was generated from the following file:

- 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_target_pos (Vector2 pos)
- Vector2 get_target_pos () const
- Vector2 get_current_pos () const
- bool check_outdated () const
- void set_color (Color color)
- void set_value (const T &value)
- T & get_value ()

Static Public Attributes

- static constexpr int radius = 20

6.17.1 Detailed Description

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

Definition at line 15 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 45 of file node_gui.hpp.

6.17.3 Member Function Documentation

6.17.3.1 check_outdated()

```
template<typename T >
bool gui::GuiNode< T >::check_outdated
```

Definition at line 119 of file node_gui.hpp.

6.17.3.2 get_current_pos()

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

Definition at line 114 of file node_gui.hpp.

6.17.3.3 get_target_pos()

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

Definition at line 109 of file node_gui.hpp.

6.17.3.4 get_value()

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

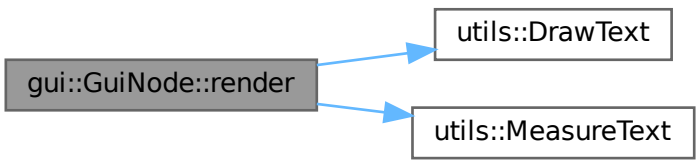
Definition at line 97 of file node_gui.hpp.

6.17.3.5 render()

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

Definition at line 48 of file node_gui.hpp.

Here is the call graph for this function:



6.17.3.6 set_color()

```
template<typename T >
void gui::GuiNode< T >::set_color (
            Color color )
```

Definition at line 87 of file node_gui.hpp.

6.17.3.7 set_target_pos()

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

Definition at line 102 of file node_gui.hpp.

6.17.3.8 set_value()

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

Definition at line 92 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.

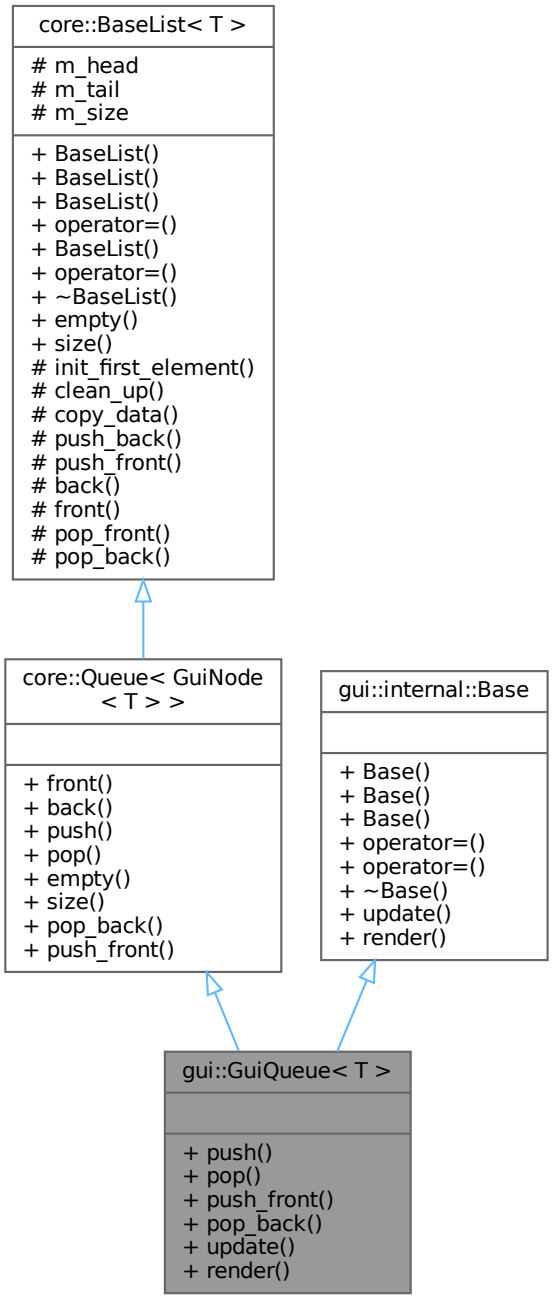
The documentation for this class was generated from the following file:

- src/gui/node_gui.hpp

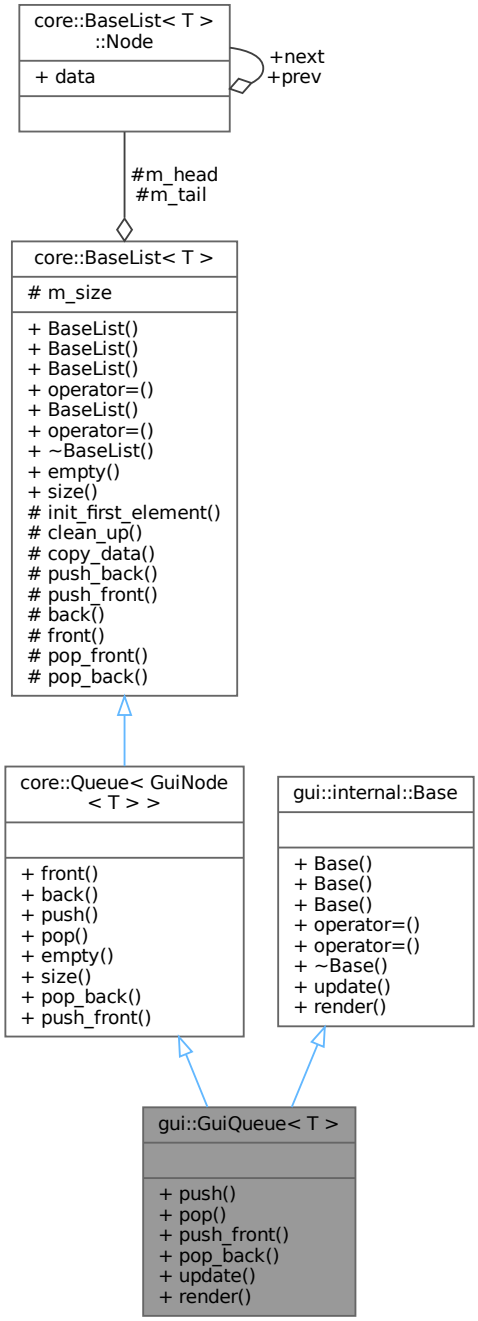
6.18 gui::GuiQueue< T > Class Template Reference

`#include <queue_gui.hpp>`

Inheritance diagram for gui::GuiQueue< T >:



Collaboration diagram for gui::GuiQueue< T >:



Public Member Functions

- void push (const T &elem)
- void pop ()
- void push_front (const T &elem)
- void pop_back ()
- void update () override
- void render () override

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 16 of file queue_gui.hpp.

6.18.2 Member Function Documentation

6.18.2.1 pop()

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

Definition at line 54 of file queue_gui.hpp.

6.18.2.2 pop_back()

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

Definition at line 64 of file queue_gui.hpp.

6.18.2.3 push()

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

Definition at line 49 of file queue_gui.hpp.

6.18.2.4 push_front()

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

Definition at line 59 of file queue_gui.hpp.

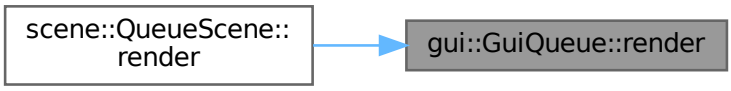
6.18.2.5 render()

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

Implements gui::internal::Base.

Definition at line 89 of file queue_gui.hpp.

Here is the caller graph for this function:



6.18.2.6 update()

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

Implements gui::internal::Base.

Definition at line 103 of file queue_gui.hpp.

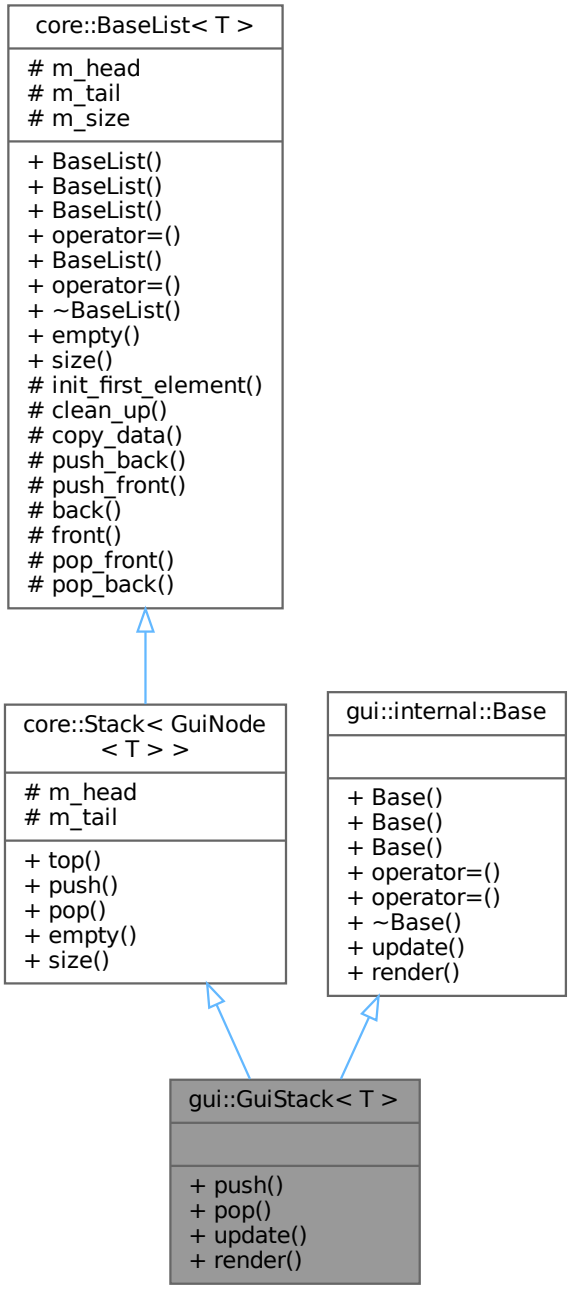
The documentation for this class was generated from the following file:

- src/gui/queue_gui.hpp

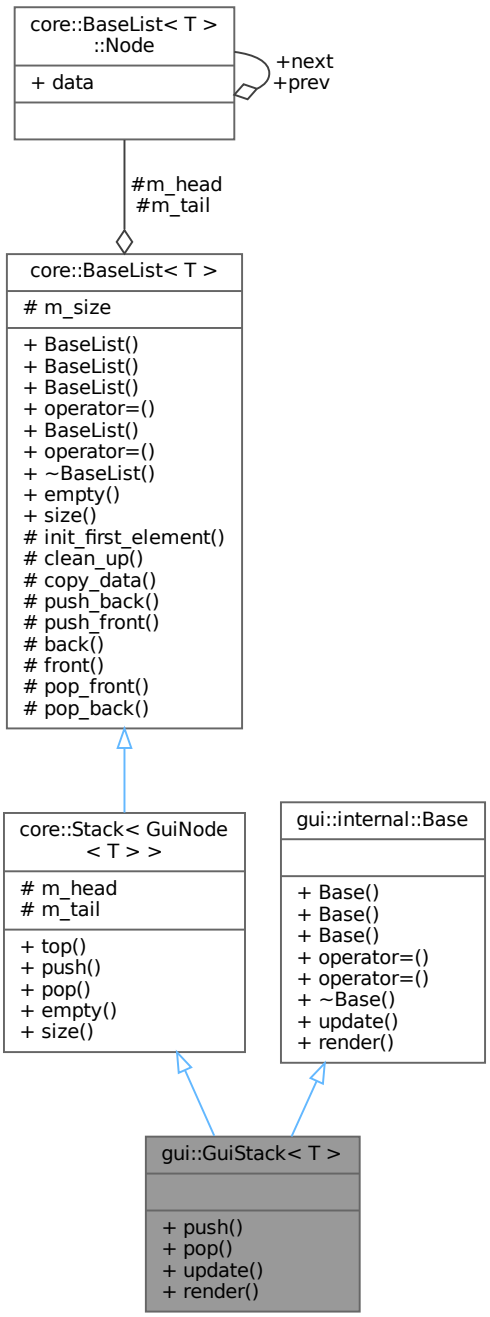
6.19 gui::GuiStack< T > Class Template Reference

`#include <stack_gui.hpp>`

Inheritance diagram for gui::GuiStack< T >:



Collaboration diagram for gui::GuiStack< T >:



Public Member Functions

- void push (const T &elem)
- void pop ()
- void update () override
- void render () override

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 16 of file stack_gui.hpp.

6.19.2 Member Function Documentation

6.19.2.1 pop()

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

Definition at line 50 of file stack_gui.hpp.

6.19.2.2 push()

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

Definition at line 45 of file stack_gui.hpp.

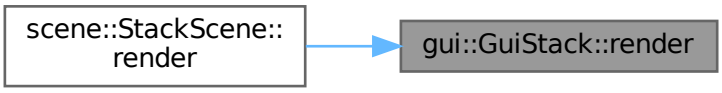
6.19.2.3 render()

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

Implements gui::internal::Base.

Definition at line 75 of file stack_gui.hpp.

Here is the caller graph for this function:



6.19.2.4 update()

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

Implements gui::internal::Base.

Definition at line 89 of file stack_gui.hpp.

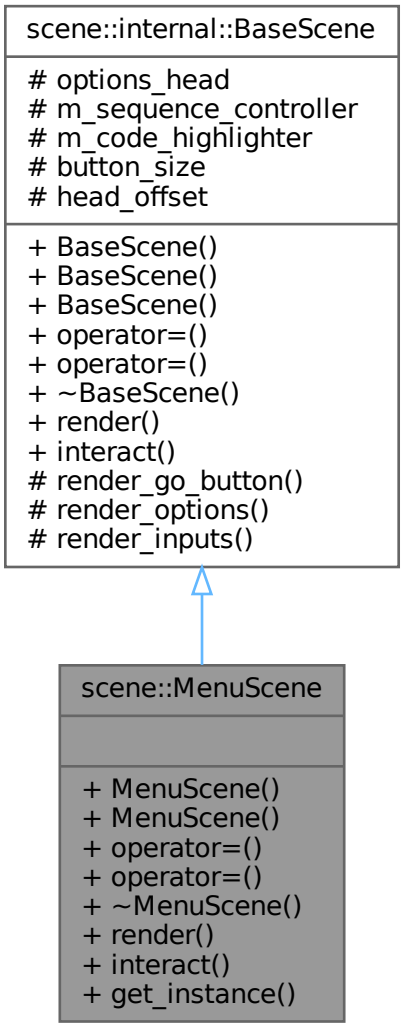
The documentation for this class was generated from the following file:

- src/gui/stack_gui.hpp

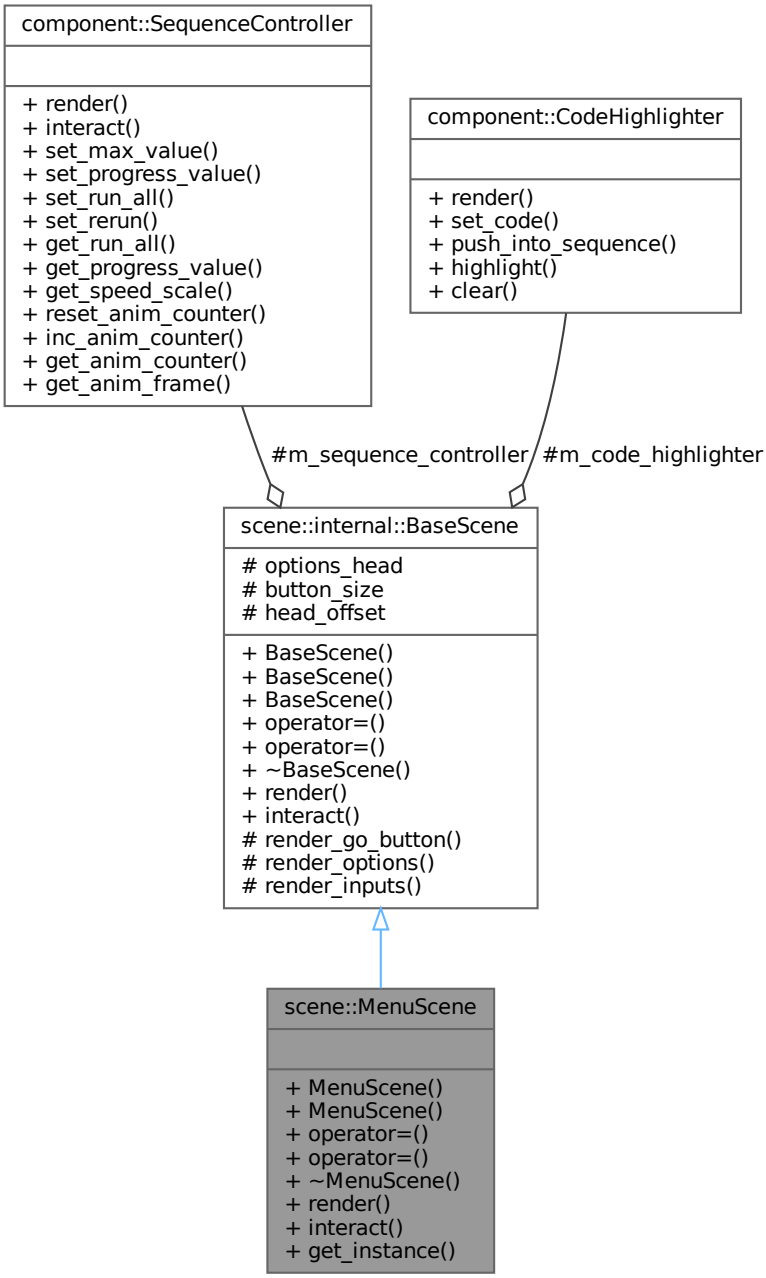
6.20 scene::MenuScene Class Reference

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

Inheritance diagram for scene::MenuScene:



Collaboration diagram for scene::MenuScene:



Public Member Functions

- MenuScene (const MenuScene &)=delete
- MenuScene (MenuScene &&)=delete
- MenuScene & operator= (const MenuScene &)=delete
- MenuScene & operator= (MenuScene &&)=delete
- ~MenuScene () override=default
- 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 ()

Static Public Member Functions

- static MenuScene & get_instance ()

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::SequenceController m_sequence_controller
- component::CodeHighlighter m_code_highlighter

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

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

6.20.1 Detailed Description

Definition at line 8 of file menu_scene.hpp.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 MenuScene() [1/2]

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

6.20.2.2 MenuScene() [2/2]

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

6.20.2.3 ~MenuScene()

```
scene::MenuScene::~MenuScene ( )  [override], [default]
```

6.20.3 Member Function Documentation

6.20.3.1 get_instance()

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

Definition at line 12 of file menu_scene.cpp.

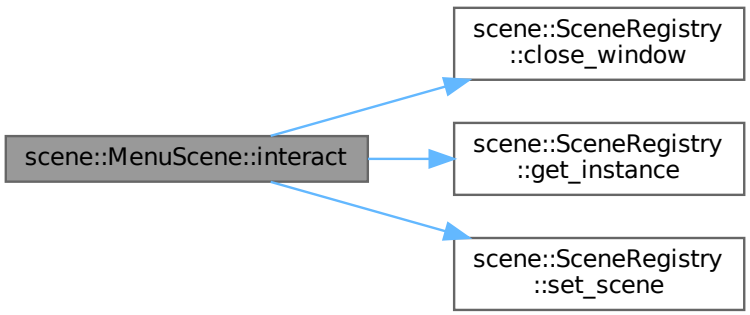
6.20.3.2 interact()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 86 of file menu_scene.cpp.

Here is the call graph for this function:



6.20.3.3 operator=() [1/2]

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

6.20.3.4 operator=() [2/2]

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

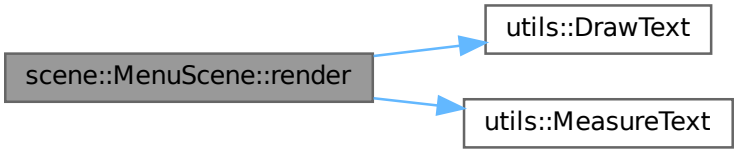
6.20.3.5 render()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 17 of file menu_scene.cpp.

Here is the call graph for this function:



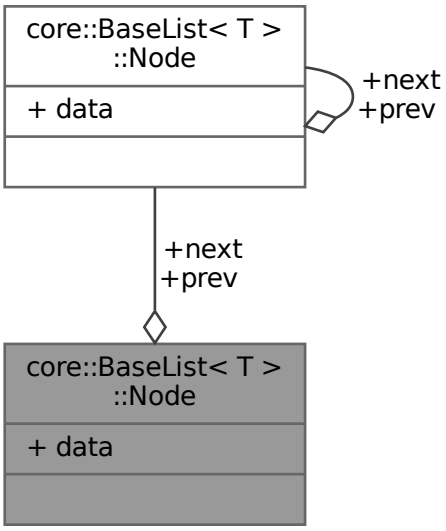
The documentation for this class was generated from the following files:

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

6.21 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 {}

6.21.1 Detailed Description

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

Definition at line 16 of file base_list.hpp.

6.21.2 Member Data Documentation

6.21.2.1 data

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

Definition at line 17 of file base_list.hpp.

6.21.2.2 next

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

Definition at line 19 of file base_list.hpp.

6.21.2.3 prev

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

Definition at line 18 of file base_list.hpp.

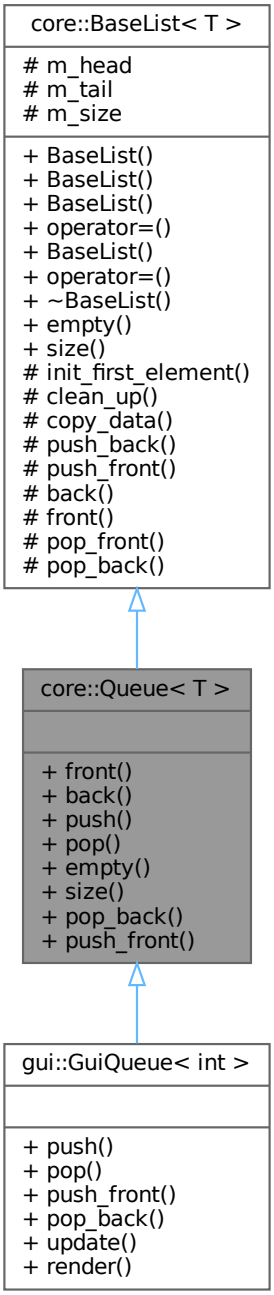
The documentation for this struct was generated from the following file:

- src/core/base_list.hpp

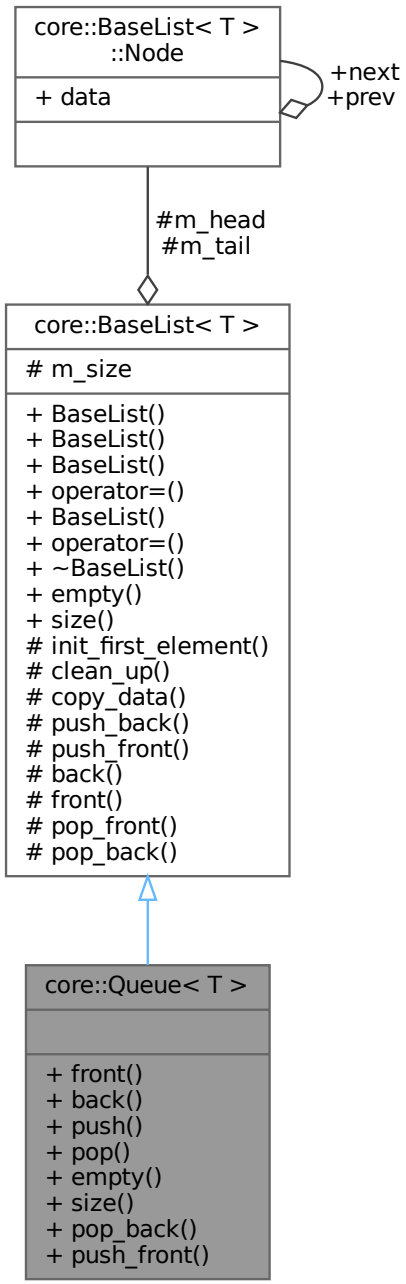
6.22 core::Queue< T > Class Template Reference

```
#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 {}

6.22.1 Detailed Description

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

Definition at line 9 of file queue.hpp.

6.22.2 Member Function Documentation

6.22.2.1 back()

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

Definition at line 36 of file queue.hpp.

6.22.2.2 empty()

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

Definition at line 48 of file base_list.hpp.

6.22.2.3 front()

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

Definition at line 31 of file queue.hpp.

6.22.2.4 pop()

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

Definition at line 46 of file queue.hpp.

6.22.2.5 pop_back()

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

Definition at line 37 of file base_list.hpp.

6.22.2.6 push()

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

Definition at line 41 of file queue.hpp.

6.22.2.7 push_front()

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

Definition at line 31 of file base_list.hpp.

6.22.2.8 size()

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

Definition at line 49 of file base_list.hpp.

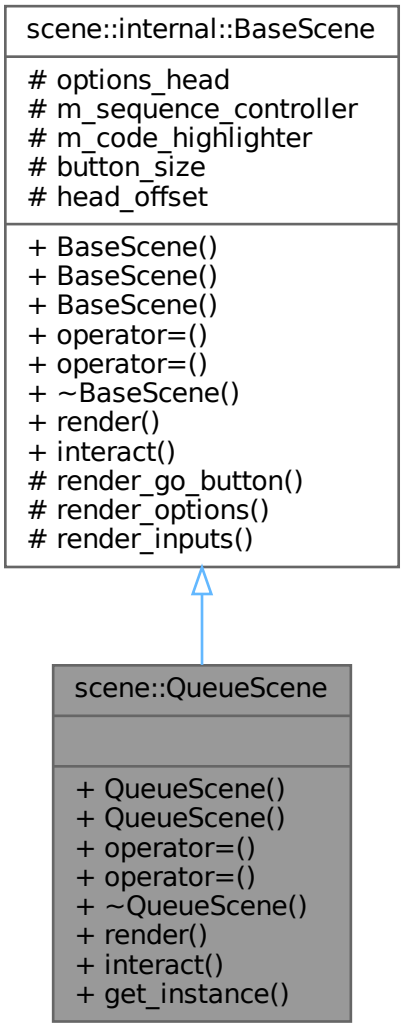
The documentation for this class was generated from the following file:

- src/core/queue.hpp

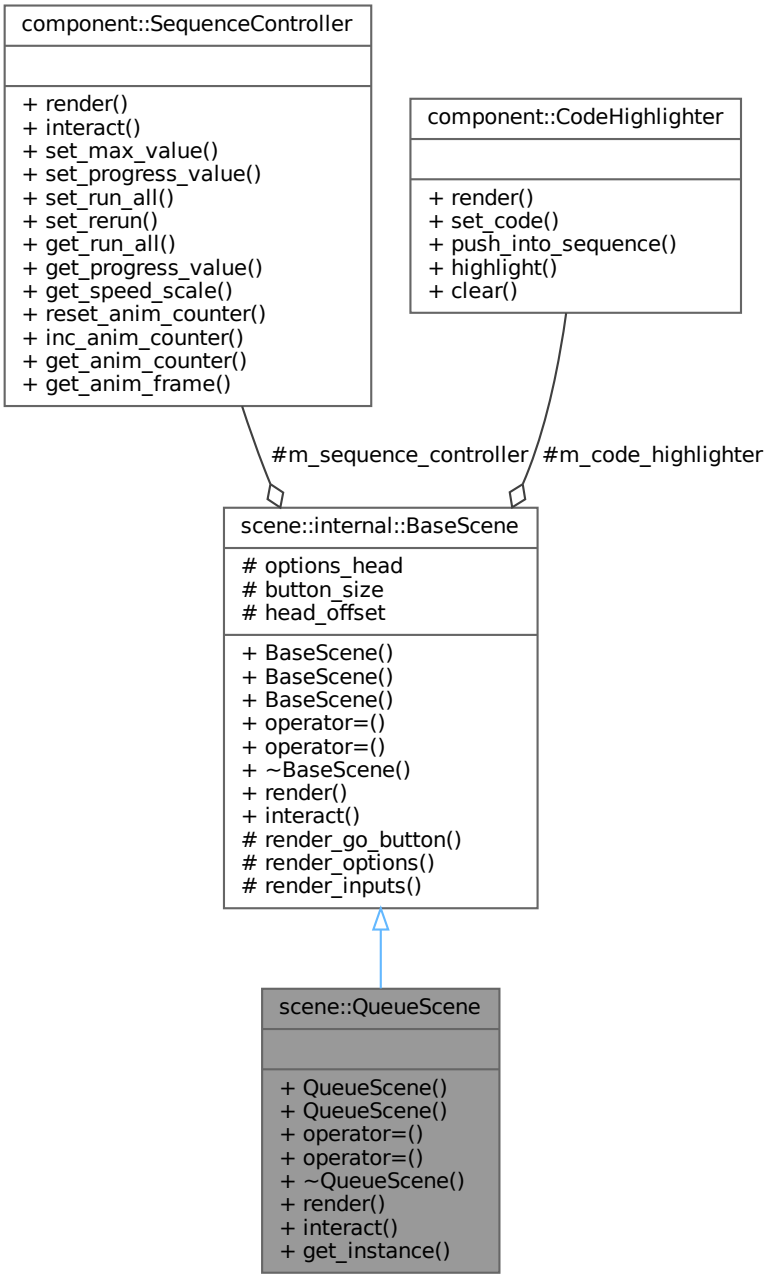
6.23 scene::QueueScene Class Reference

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

Inheritance diagram for scene::QueueScene:



Collaboration diagram for scene::QueueScene:



Public Member Functions

- QueueScene (const QueueScene &)=delete
- QueueScene (QueueScene &&)=delete
- QueueScene & operator= (const QueueScene &)=delete
- QueueScene & operator= (QueueScene &&)=delete
- ~QueueScene () override=default
- 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 ()

Static Public Member Functions

- static QueueScene & get_instance ()

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::SequenceController m_sequence_controller
- component::CodeHighlighter m_code_highlighter

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

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

6.23.1 Detailed Description

Definition at line 16 of file queue_scene.hpp.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 QueueScene() [1/2]

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

6.23.2.2 QueueScene() [2/2]

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

6.23.2.3 ~QueueScene()

```
scene::QueueScene::~QueueScene ( )  [override], [default]
```

6.23.3 Member Function Documentation

6.23.3.1 get_instance()

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

Definition at line 17 of file queue_scene.cpp.

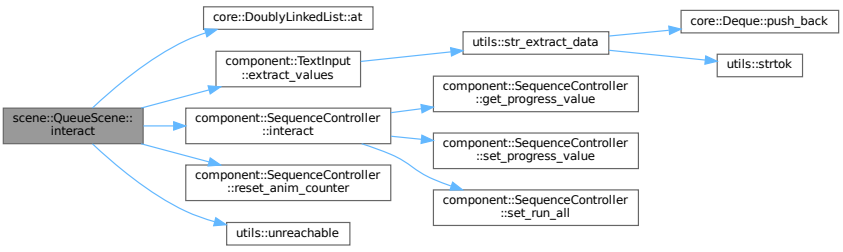
6.23.3.2 interact()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 74 of file queue_scene.cpp.

Here is the call graph for this function:



6.23.3.3 operator=() [1/2]

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

6.23.3.4 operator=() [2/2]

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

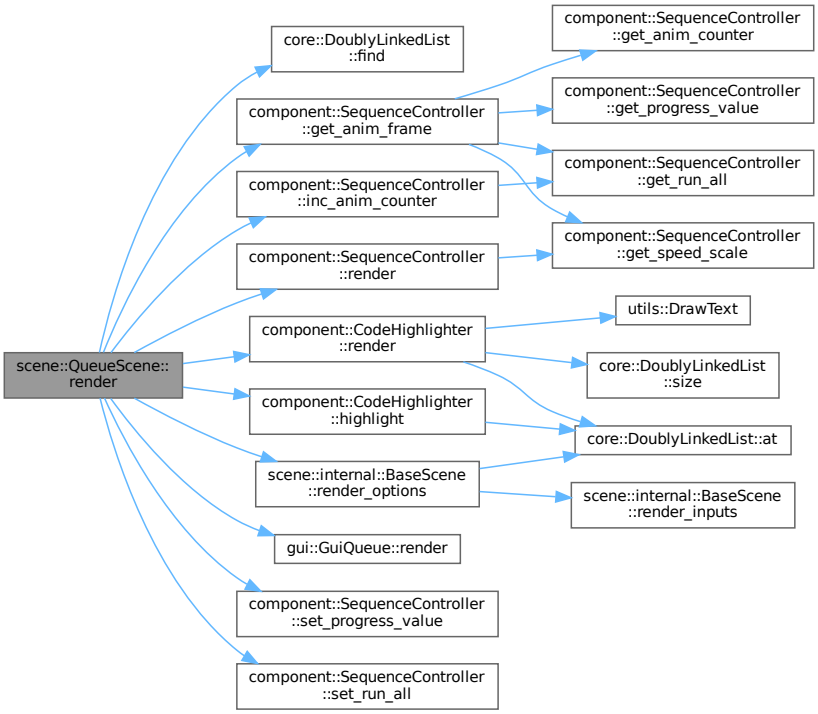
6.23.3.5 render()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 54 of file queue_scene.cpp.

Here is the call graph for this function:



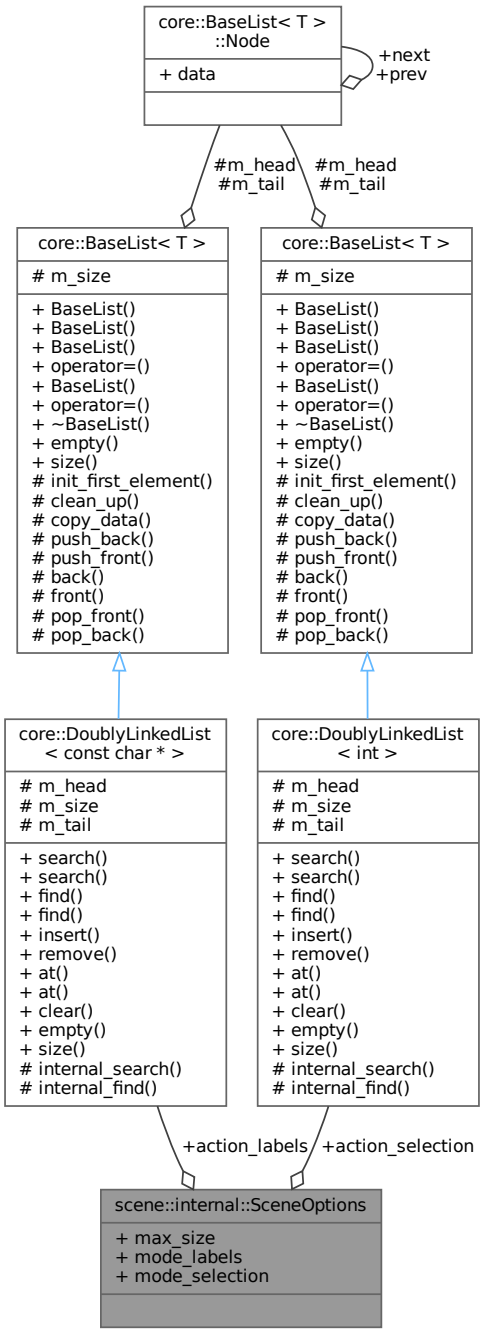
The documentation for this class was generated from the following files:

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

6.24 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

6.24.1 Detailed Description

Definition at line 10 of file scene_options.hpp.

6.24.2 Member Data Documentation

6.24.2.1 action_labels

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

Definition at line 14 of file scene_options.hpp.

6.24.2.2 action_selection

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

Definition at line 15 of file scene_options.hpp.

6.24.2.3 max_size

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

Definition at line 11 of file scene_options.hpp.

6.24.2.4 mode_labels

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

Definition at line 12 of file scene_options.hpp.

6.24.2.5 mode_selection

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

Definition at line 13 of file scene_options.hpp.

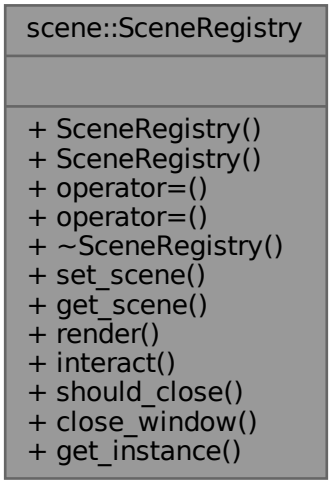
The documentation for this struct was generated from the following file:

- src/scene/scene_options.hpp

6.25 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 ()

6.25.1 Detailed Description

Definition at line 27 of file scene_registry.hpp.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 SceneRegistry() [1/2]

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

6.25.2.2 SceneRegistry() [2/2]

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

6.25.2.3 ~SceneRegistry()

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

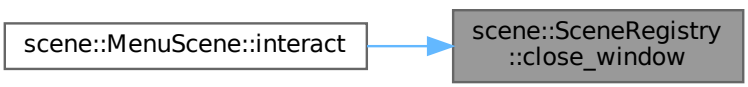
6.25.3 Member Function Documentation

6.25.3.1 close_window()

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

Definition at line 25 of file scene_registry.cpp.

Here is the caller graph for this function:

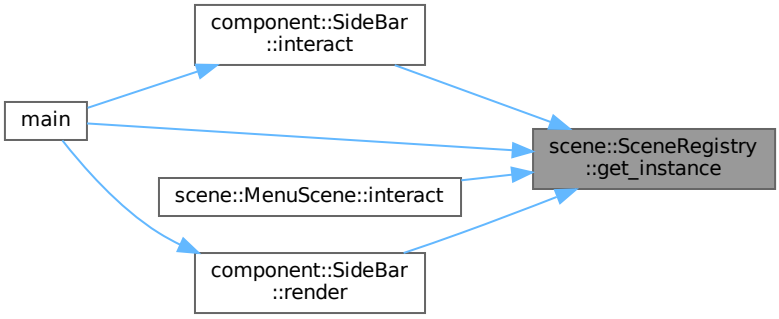


6.25.3.2 get_instance()

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

Definition at line 7 of file scene_registry.cpp.

Here is the caller graph for this function:

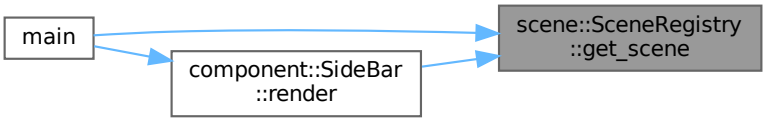


6.25.3.3 get_scene()

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

Definition at line 17 of file scene_registry.cpp.

Here is the caller graph for this function:

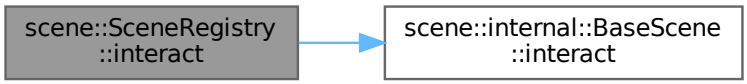


6.25.3.4 interact()

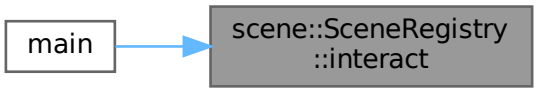
```
void scene::SceneRegistry::interact ( )
```

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:



6.25.3.5 operator=() [1/2]

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

6.25.3.6 operator=() [2/2]

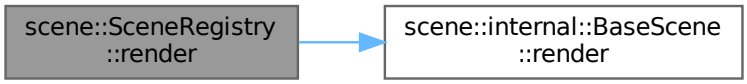
```
SceneRegistry & scene::SceneRegistry::operator= (
            SceneRegistry &&  )  [delete]
```

6.25.3.7 render()

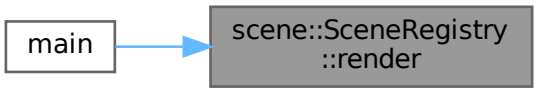
```
void scene::SceneRegistry::render ( )
```

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:

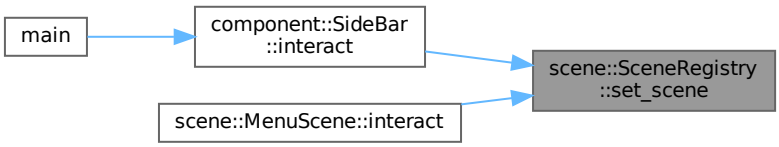


6.25.3.8 set_scene()

```
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:

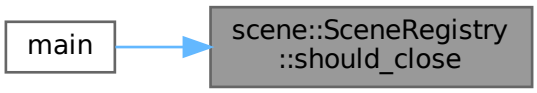


6.25.3.9 should_close()

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

Definition at line 23 of file scene_registry.cpp.

Here is the caller graph for this function:



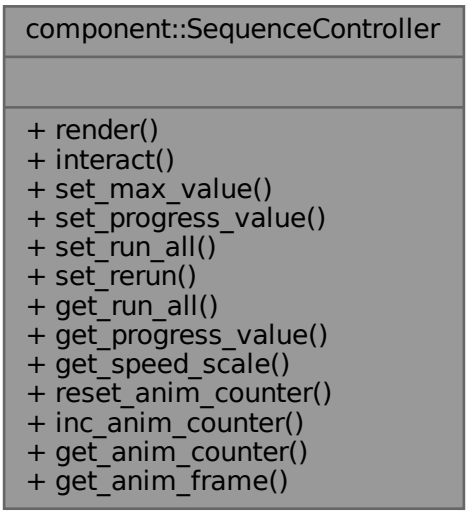
The documentation for this class was generated from the following files:

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

6.26 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

6.26.1 Detailed Description

Definition at line 8 of file sequence_controller.hpp.

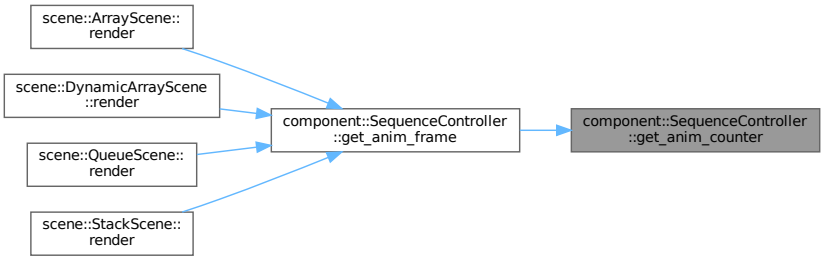
6.26.2 Member Function Documentation

6.26.2.1 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:

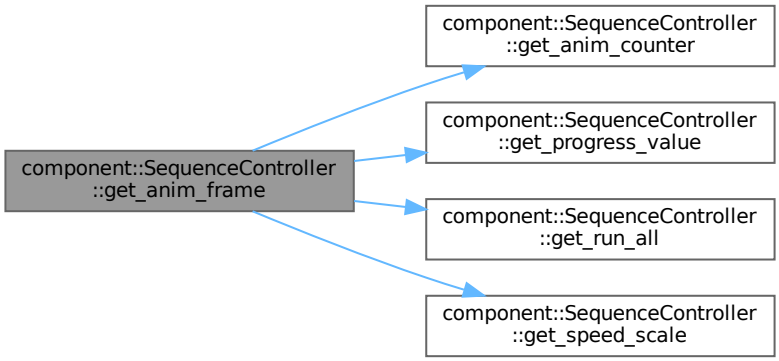


6.26.2.2 get_anim_frame()

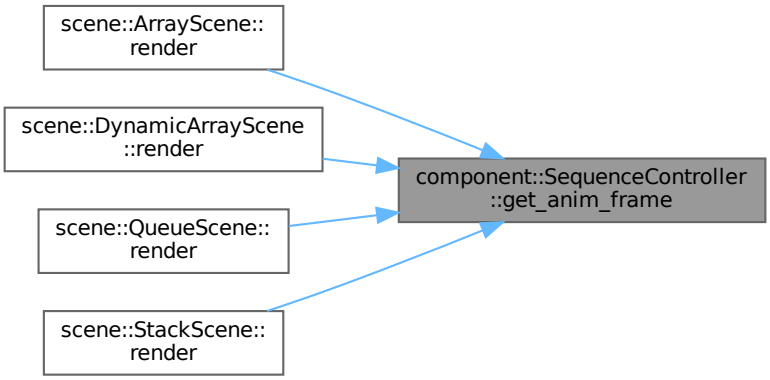
```
int component::SequenceController::get_anim_frame ( ) const
```

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:

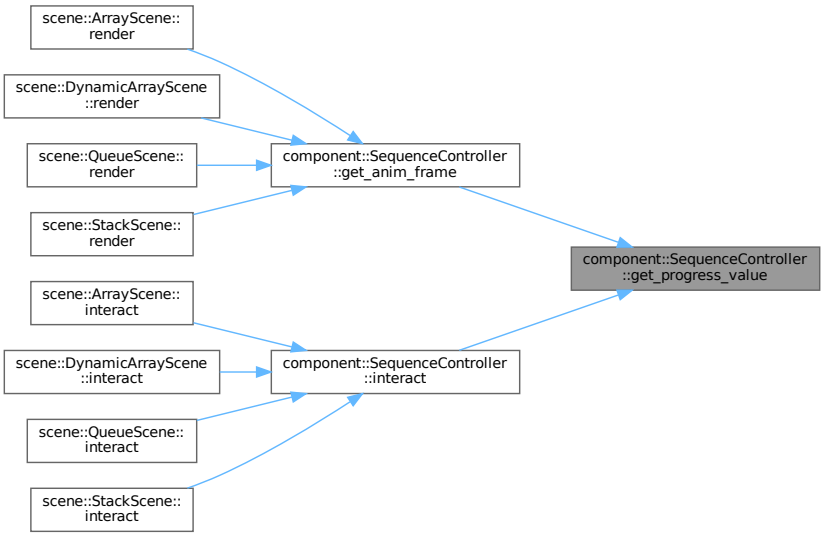


6.26.2.3 get_progress_value()

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

Definition at line 21 of file sequence_controller.cpp.

Here is the caller graph for this function:

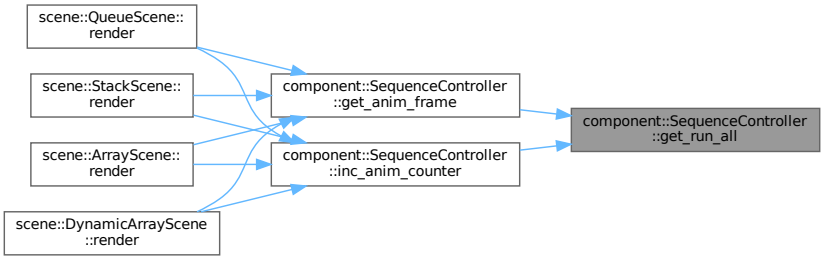


6.26.2.4 get_run_all()

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

Definition at line 19 of file sequence_controller.cpp.

Here is the caller graph for this function:

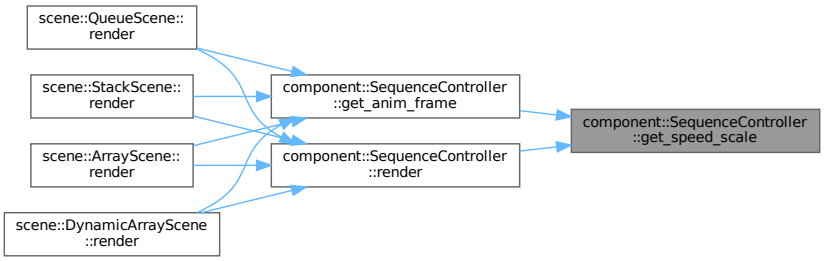


6.26.2.5 get_speed_scale()

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

Definition at line 23 of file sequence_controller.cpp.

Here is the caller graph for this function:

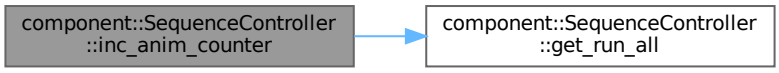


6.26.2.6 inc_anim_counter()

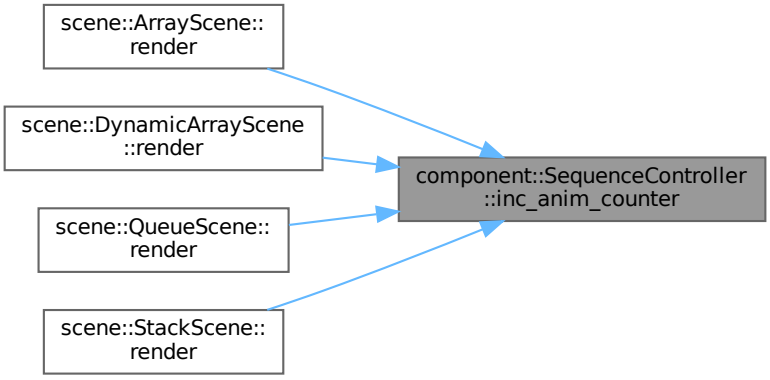
```
void component::SequenceController::inc_anim_counter ( )
```

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:

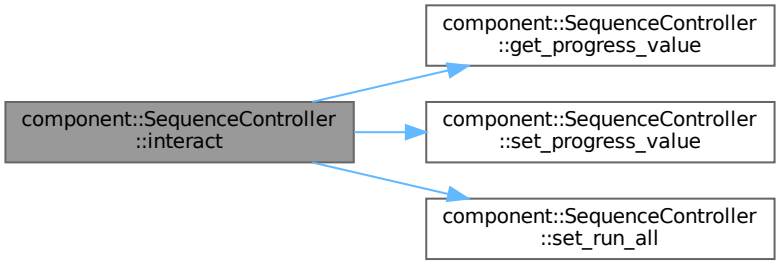


6.26.2.7 interact()

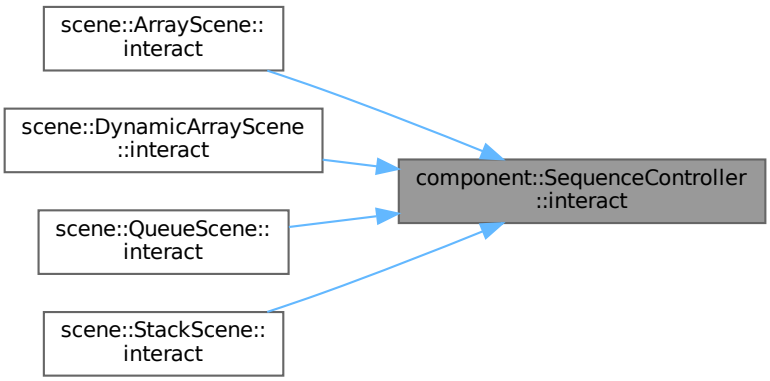
```
bool component::SequenceController::interact ( )
```

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:

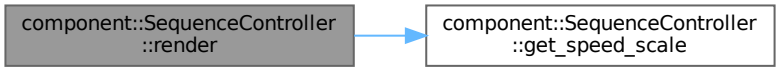


6.26.2.8 render()

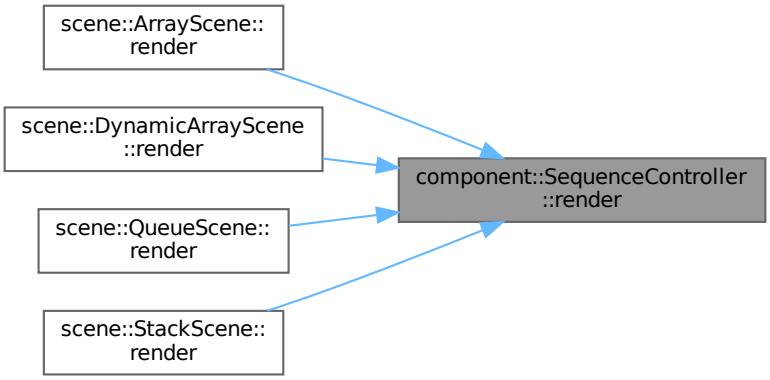
```
void component::SequenceController::render ( )
```

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:

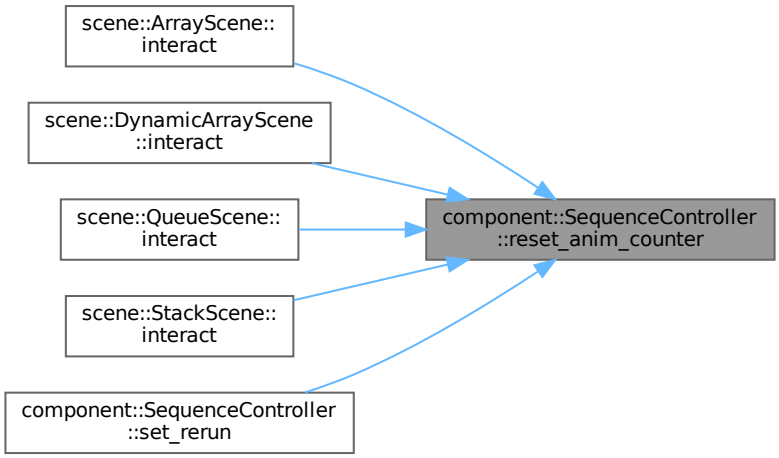


6.26.2.9 reset_anim_counter()

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

Definition at line 27 of file sequence_controller.cpp.

Here is the caller graph for this function:



6.26.2.10 set_max_value()

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

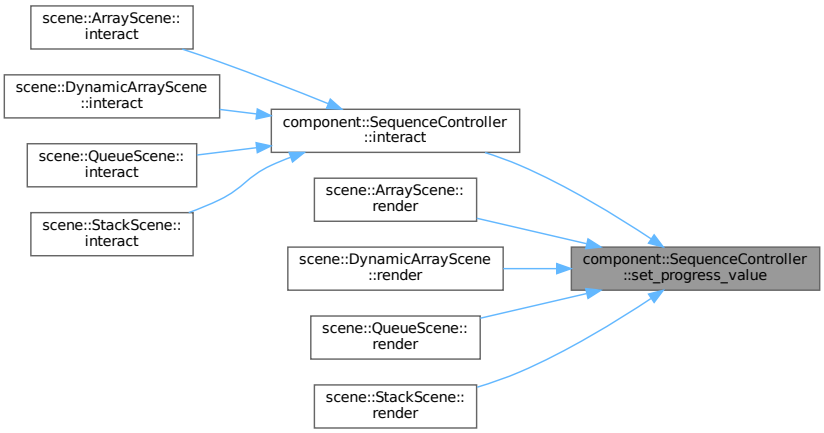
Definition at line 11 of file sequence_controller.cpp.

6.26.2.11 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:

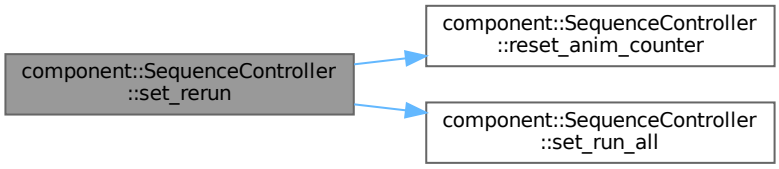


6.26.2.12 set_rerun()

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

Definition at line 37 of file sequence_controller.cpp.

Here is the call graph for this function:

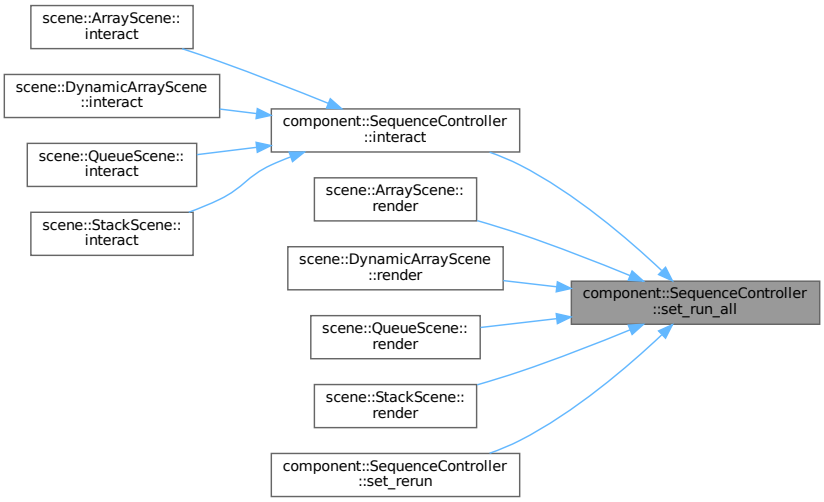


6.26.2.13 set_run_all()

```
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:



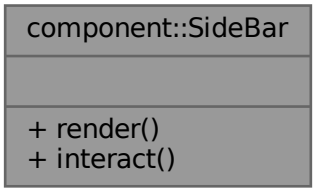
The documentation for this class was generated from the following files:

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

6.27 component::SideBar Class Reference

`#include <sidebar.hpp>`

Collaboration diagram for component::SideBar:



Public Member Functions

- void render ()
- void interact () const

6.27.1 Detailed Description

Definition at line 10 of file sidebar.hpp.

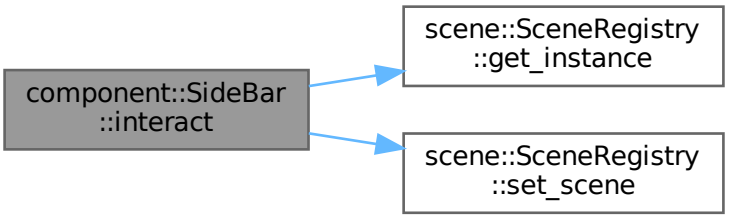
6.27.2 Member Function Documentation

6.27.2.1 interact()

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

Definition at line 22 of file sidebar.cpp.

Here is the call graph for this function:



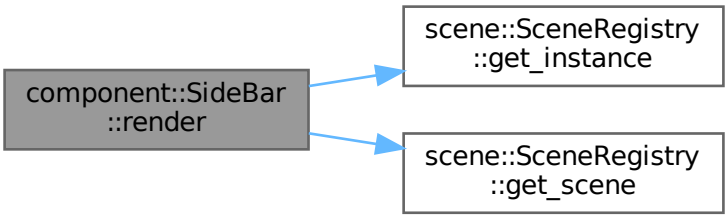
Here is the caller graph for this function:

6.27.2.2 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:

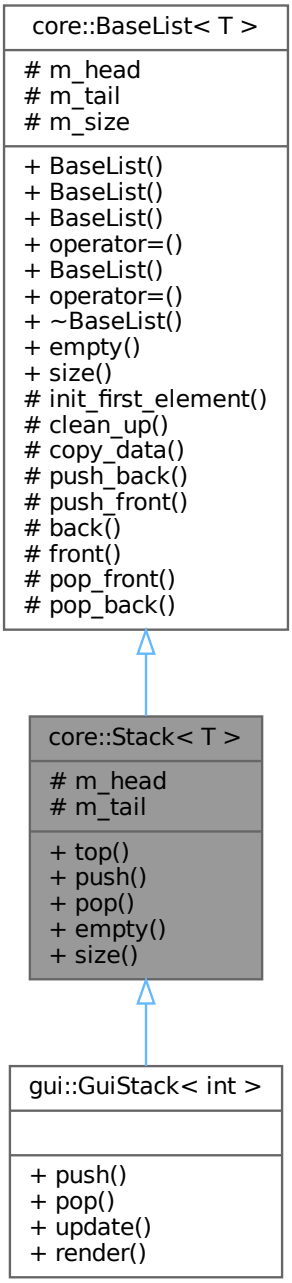
The documentation for this class was generated from the following files:

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

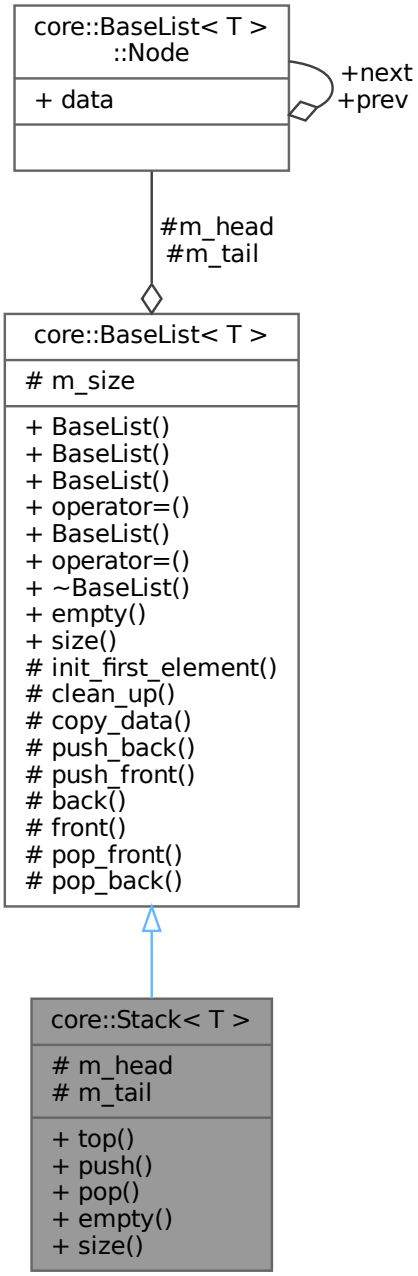
6.28 core::Stack< T > Class Template Reference

```
#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 ()

6.28.1 Detailed Description

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

Definition at line 9 of file stack.hpp.

6.28.2 Member Typedef Documentation

6.28.2.1 Base

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

Definition at line 11 of file stack.hpp.

6.28.3 Member Function Documentation

6.28.3.1 empty()

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

Definition at line 48 of file base_list.hpp.

6.28.3.2 pop()

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

Definition at line 38 of file stack.hpp.

6.28.3.3 push()

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

Definition at line 33 of file stack.hpp.

6.28.3.4 size()

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

Definition at line 49 of file base_list.hpp.

6.28.3.5 top()

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

Definition at line 28 of file stack.hpp.

6.28.4 Member Data Documentation

6.28.4.1 m_head

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

Definition at line 22 of file base_list.hpp.

6.28.4.2 m_tail

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

Definition at line 23 of file base_list.hpp.

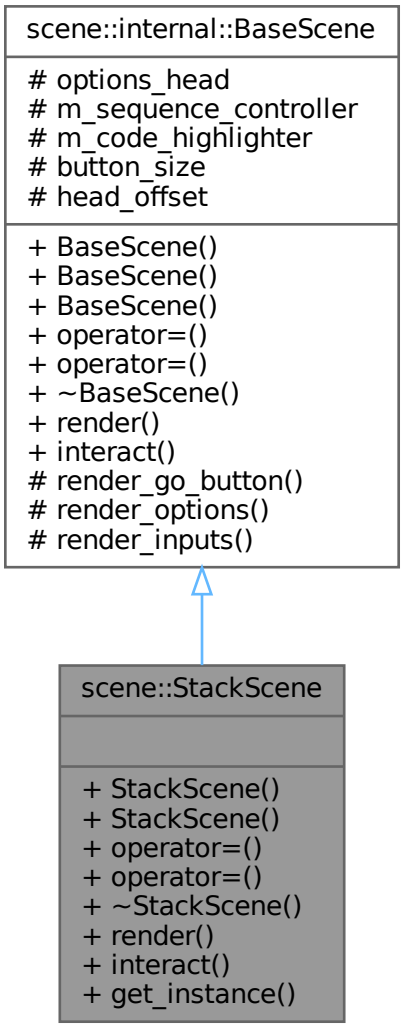
The documentation for this class was generated from the following file:

- src/core/stack.hpp

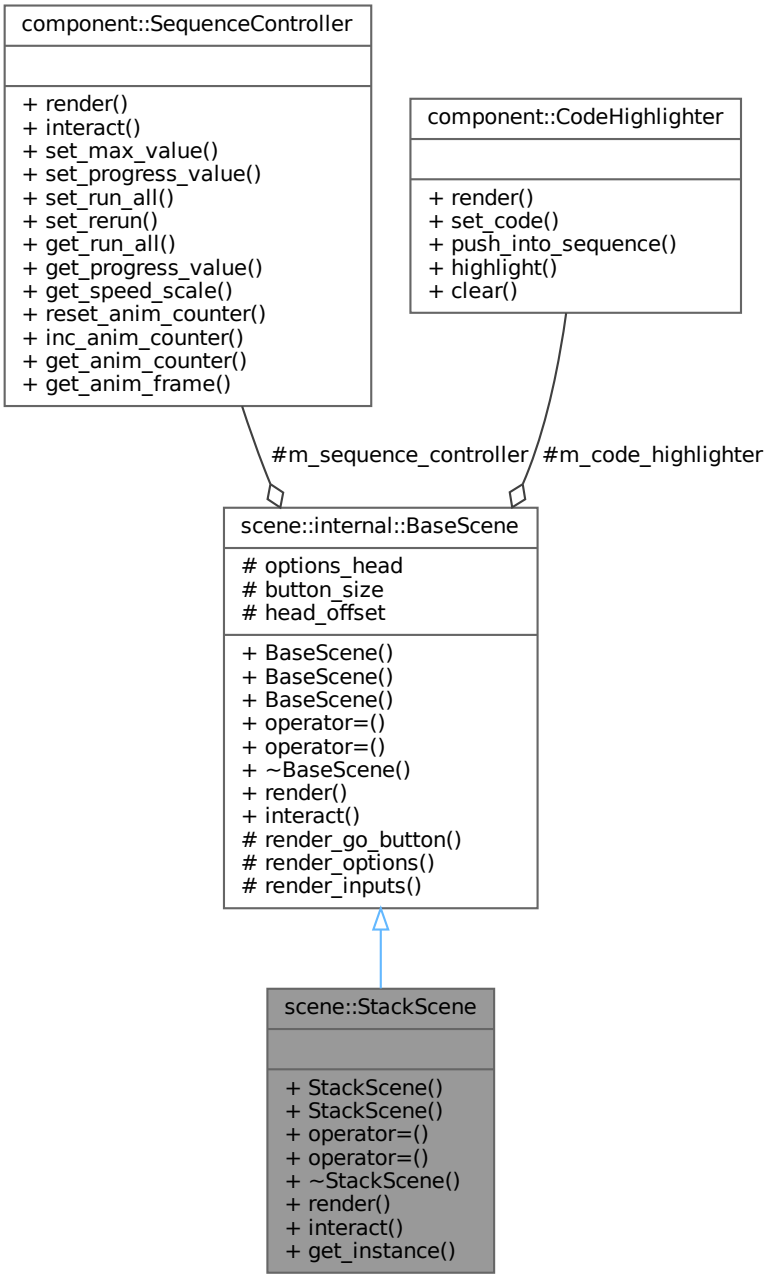
6.29 scene::StackScene Class Reference

`#include <stack_scene.hpp>`

Inheritance diagram for scene::StackScene:



Collaboration diagram for scene::StackScene:



Public Member Functions

- StackScene (const StackScene &)=delete
- StackScene (StackScene &&)=delete
- StackScene & operator= (const StackScene &)=delete
- StackScene & operator= (StackScene &&)=delete
- ~StackScene () override=default
- 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 ()

Static Public Member Functions

- static StackScene & get_instance ()

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::SequenceController m_sequence_controller
- component::CodeHighlighter m_code_highlighter

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

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

6.29.1 Detailed Description

Definition at line 14 of file stack_scene.hpp.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 StackScene() [1/2]

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

6.29.2.2 StackScene() [2/2]

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

6.29.2.3 ~StackScene()

```
scene::StackScene::~StackScene ( )  [override], [default]
```

6.29.3 Member Function Documentation

6.29.3.1 get_instance()

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

Definition at line 17 of file stack_scene.cpp.

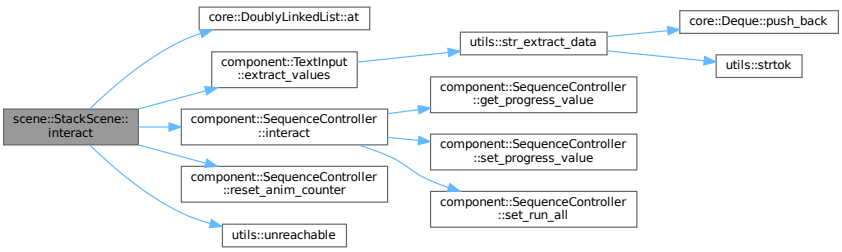
6.29.3.2 interact()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 74 of file stack_scene.cpp.

Here is the call graph for this function:



6.29.3.3 operator=() [1/2]

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

6.29.3.4 operator=() [2/2]

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

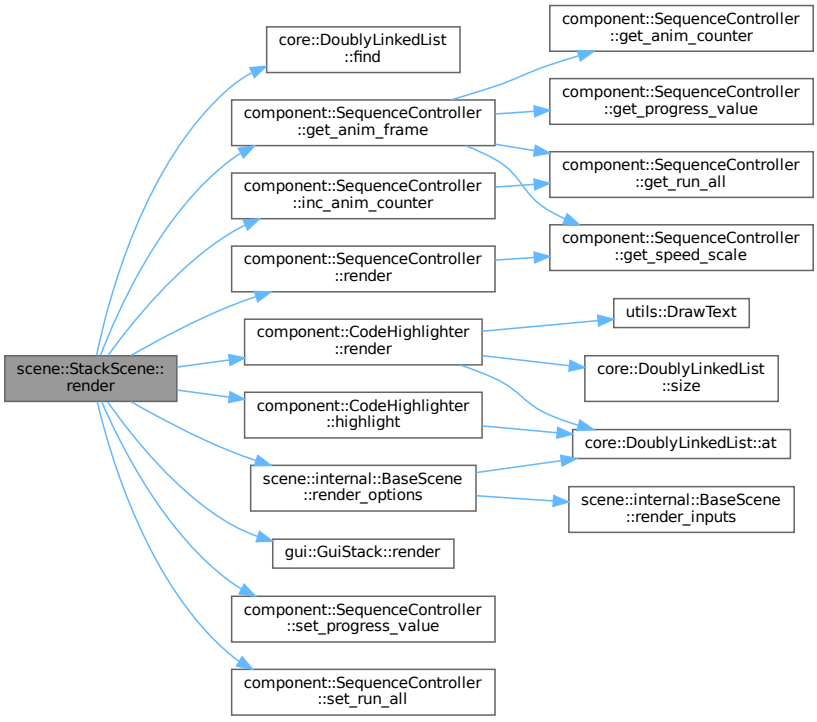
6.29.3.5 render()

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

Reimplemented from scene::internal::BaseScene.

Definition at line 22 of file stack_scene.cpp.

Here is the call graph for this function:



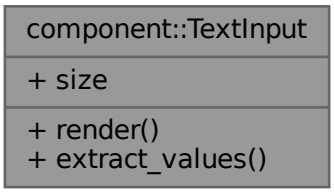
The documentation for this class was generated from the following files:

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

6.30 component::TextInput Class Reference

`#include <text_input.hpp>`

Collaboration diagram for component::TextInput:



Public Member Functions

- void render (float &options_head, float head_offset)
- core::Deque< int > extract_values ()

Static Public Attributes

- static constexpr Vector2 size {200, 50}

6.30.1 Detailed Description

Definition at line 12 of file text_input.hpp.

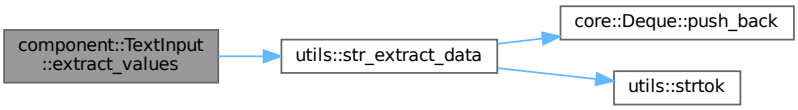
6.30.2 Member Function Documentation

6.30.2.1 extract_values()

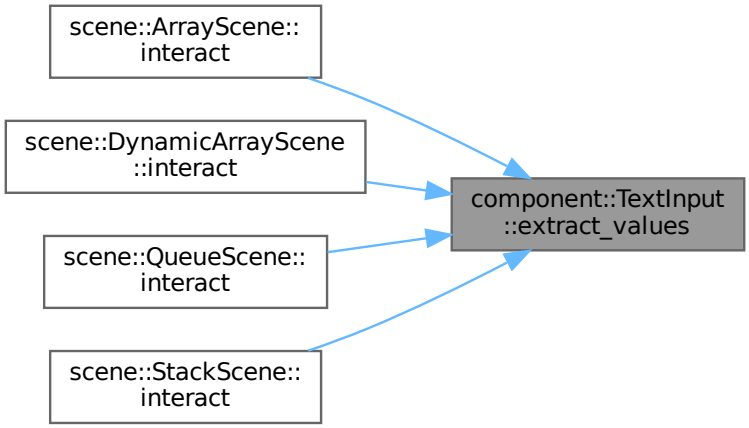
```
core::Deque< int > component::TextInput::extract_values ( )
```

Definition at line 21 of file text_input.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.30.2.2 render()

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

Definition at line 9 of file text_input.cpp.

6.30.3 Member Data Documentation

6.30.3.1 size

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

Definition at line 18 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

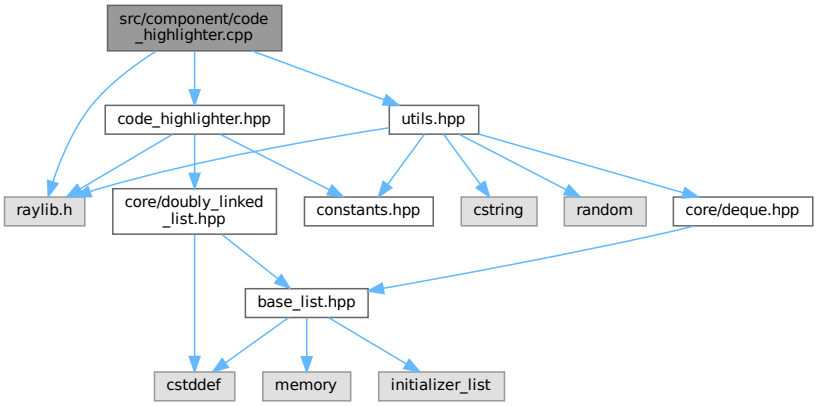
Chapter 7

File Documentation

7.1 src/component/code_highlighter.cpp File Reference

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

Include dependency graph for code_highlighter.cpp:



Namespaces

- namespace component

7.2 code_highlighter.cpp

Go to the documentation of this file.

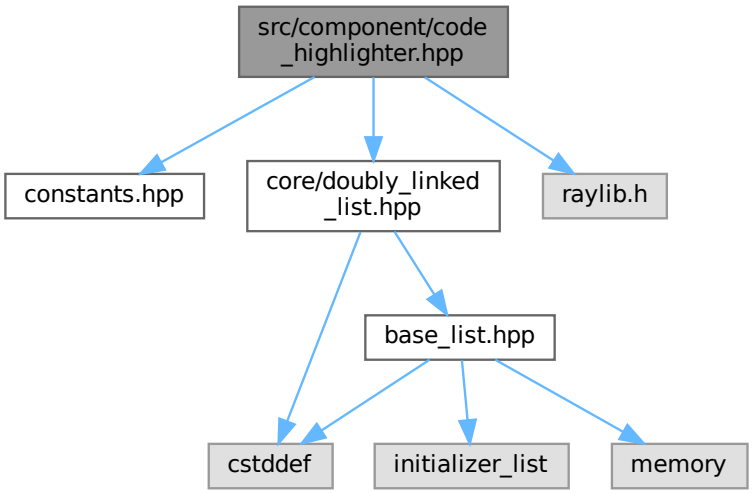
```
00001 #include "code_highlighter.hpp"
00002
00003 #include "raylib.h"
00004 #include "utils.hpp"
00005
```

```
00006 namespace component {
00007
00008 void CodeHighlighter::render() {
00009     for (int i = 0; i < m_src_code.size(); ++i) {
00010         Color bg_color = (i == m_highlighted_line) ? VIOLET : BLACK;
00011         Rectangle shape{head_pos.x, head_pos.y + i * height, width, height};
00012         Vector2 text_head = {head_pos.x + 10, head_pos.y + i * height + 5};
00013
00014         DrawRectangleRec(shape, bg_color);
00015         utils::DrawText(m_src_code.at(i), text_head, WHITE, 20, 2);
00016     }
00017 }
00018
00019 void CodeHighlighter::set_code(core::DoublyLinkedList<const char*>&& src_code) {
00020     clear();
00021     m_src_code = src_code;
00022 }
00023
00024 void CodeHighlighter::push_into_sequence(int line_number) {
00025     m_sequence.insert(m_sequence.size(), line_number);
00026 }
00027
00028 void CodeHighlighter::highlight(int frame_idx) {
00029     m_highlighted_line = m_sequence.at(frame_idx);
00030 }
00031
00032 void CodeHighlighter::clear() {
00033     m_src_code.clear();
00034     m_sequence.clear();
00035 }
00036
00037 }  // namespace component
```

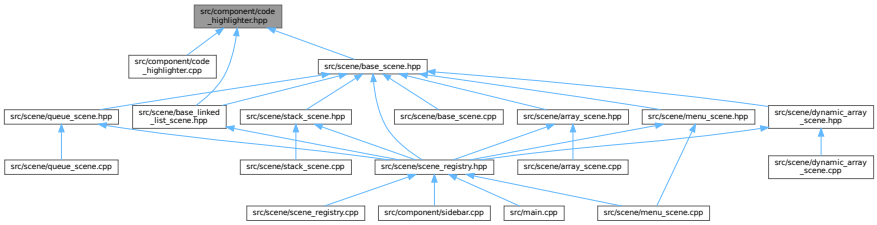
7.3 src/component/code_highlighter.hpp File Reference

```
#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.

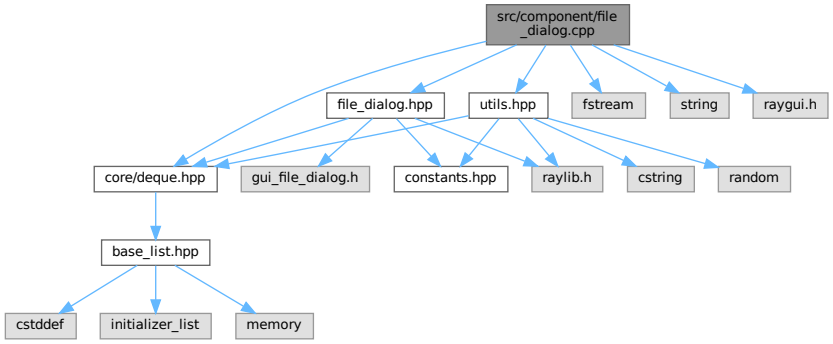
```
00001 #ifndef COMPONENT_CODE_HIGHLIGHTER_HPP_
00002 #define COMPONENT_CODE_HIGHLIGHTER_HPP_
00003
00004 #include "constants.hpp"
00005 #include "core/doubly_linked_list.hpp"
00006 #include "raylib.h"
00007
00008 namespace component {
00009
00010 class CodeHighlighter {
00011 private:
00012     static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

7.5 src/component/file_dialog.cpp File Reference

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

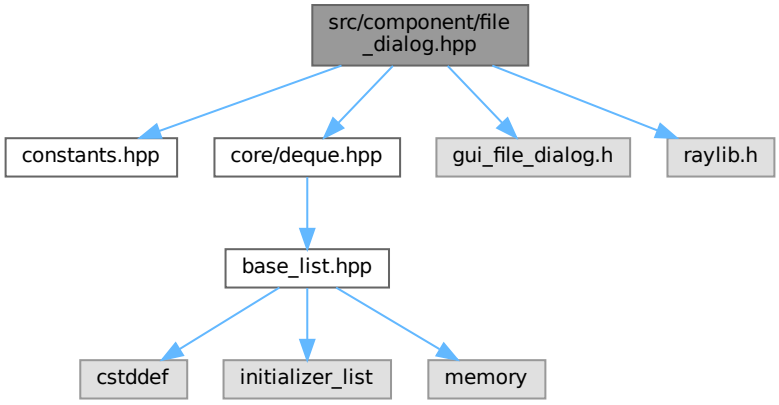
```
00001 #include "file_dialog.hpp"
00002
00003 #include <fstream>
00004 #include <string>
00005
00006 #include "core/deque.hpp"
00007 #include "raygui.h"
00008 #include "utils.hpp"
00009
00010 namespace component {
00011
00012 void FileDialog::render(float& options_head, float head_offset) {
00013     if (m_file_dialog_state.windowActive) {
00014         GuiLock();
00015     }
00016
00017     const char* const file_name =
00018         static_cast<char*>(m_file_dialog_state.fileNameText);
00019
00020     const char* const text =
00021         (m_file_dialog_state.SelectFilePressed) ? file_name : "Select file";
00022
00023     Rectangle shape{options_head, constants::scene_height - size.y, size.x,
00024                     size.y};
00025
00026     if (GuiButton(shape, GuiIconText(ICON_FILE_OPEN, text))) {
00027         m_file_dialog_state.windowActive = true;
00028     }
00029
00030     options_head += (size.x + head_offset);
00031
00032     GuiUnlock();
00033     GuiFileDialog(&m_file_dialog_state);
00034 }
00035
00036 core::Deque<int> FileDialog::extract_values() {
00037     std::string file_name;
00038     file_name += static_cast<char*>(m_file_dialog_state.dirPathText);
00039     file_name += '/';
00040     file_name += static_cast<char*>(m_file_dialog_state.fileNameText);
00041
```

```
00042     std::ifstream ifs(file_name);
00043     char buffer[constants::text_buffer_size]{};  // NOLINT
00044     ifs » buffer;
00045
00046     return utils::str_extract_data(buffer);  // NOLINT
00047 }
00048
00049 bool FileDialog::is_pressed() const {
00050     return m_file_dialog_state.SelectFilePressed;
00051 }
00052
00053 void FileDialog::reset_pressed() {
00054     m_file_dialog_state.SelectFilePressed = false;
00055 }
00056
00057 }  // namespace component
```

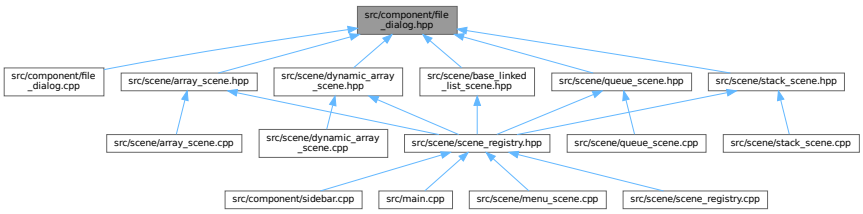
7.7 src/component/file_dialog.hpp File Reference

```
#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

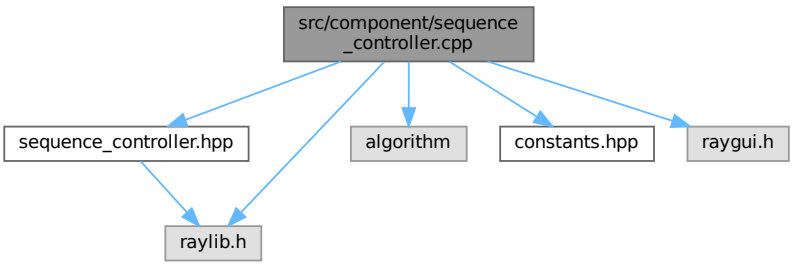
Go to the documentation of this file.

```
00001 #ifndef COMPONENT_FILE_DIALOG_HPP_
00002 #define COMPONENT_FILE_DIALOG_HPP_
00003
00004 #include "constants.hpp"
00005 #include "core/deque.hpp"
00006 #include "gui_file_dialog.h"
00007 #include "raylib.h"
00008
00009 namespace component {
00010
00011 class FileDialog {
00012 private:
00013     GuiFileDialogState m_file_dialog_state{
00014         InitGuiFileDialog(GetWorkingDirectory())};
00015
00016     char m_file_input[constants::text_buffer_size] = "";  // NOLINT
00017
00018 public:
00019     static constexpr Vector2 size{200, 50};
00020
00021     void render(float& options_head, float head_offset);
00022     core::Deque<int> extract_values();
00023     bool is_pressed() const;
00024     void reset_pressed();
00025 };
00026
00027 }  // namespace component
00028
00029 #endif  // COMPONENT_FILE_DIALOG_HPP_
```

7.9 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

7.10 sequence_controller.cpp

Go to the documentation of this file.

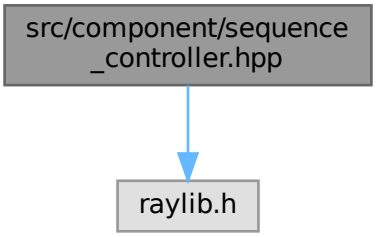
```
00001 #include "sequence_controller.hpp"
00002
00003 #include <algorithm>
00004
00005 #include "constants.hpp"
00006 #include "raygui.h"
00007 #include "raylib.h"
00008
00009 namespace component {
00010
00011 void SequenceController::set_max_value(int num) { m_num_steps = num; }
00012
00013 void 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, "<");
00079     m_next_speed = GuiButton(next_speed_shape, ">");
00080     GuiStatusBar(speed_shape, TextFormat("Speed: %.2fx", get_speed_scale()));
00081
00082     m_replay = GuiButton(replay_shape, "R");
```

```
00083     m_prev_frame = GuiButton(prev_frame_shape, "<");
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, ">");
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
```

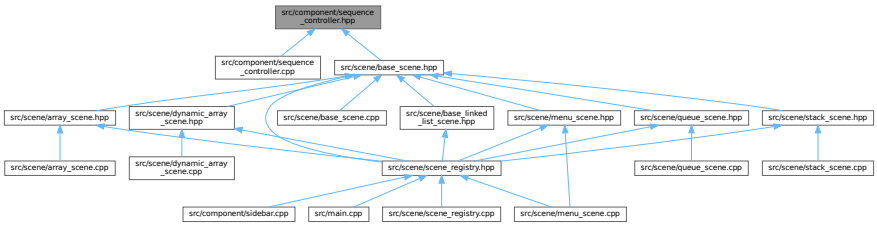
7.11 src/component/sequence_controller.hpp File Reference

`#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

7.12 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_
```

7.13 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

7.14 sidebar.cpp

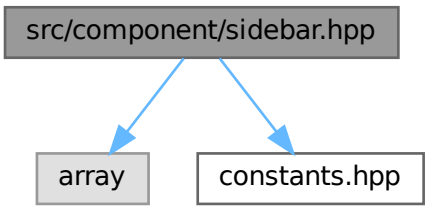
Go to the documentation of this file.

```
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     scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();
00013     int options_head = 2 * constants::sidebar_width;
00014
00015     constexpr float scale = 1.75;
00016
00017     m_next_scene = GuiToggleGroup(
00018         Rectangle{0, sidebar_width / scale, sidebar_width, button_height},
00019         sidebar_labels, registry.get_scene());
00020 }
00021
00022 void SideBar::interact() const {
00023     scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();
00024     registry.set_scene(m_next_scene);
00025 }
00026
00027 }  // namespace component
```

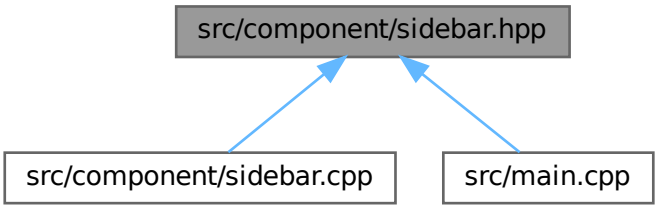
7.15 src/component/sidebar.hpp File Reference

```
#include <array>
#include "constants.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

7.16 sidebar.hpp

Go to the documentation of this file.

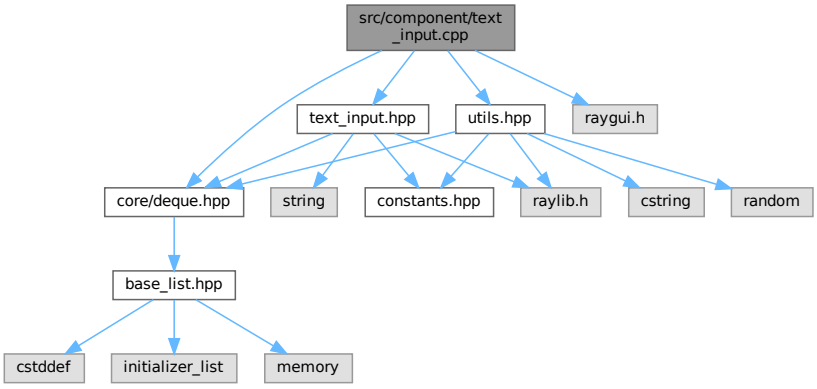
```
00001 #ifndef COMPONENT_SIDEBAR_HPP_
00002 #define COMPONENT_SIDEBAR_HPP_
00003
00004 #include <array>
00005
00006 #include "constants.hpp"
00007
00008 namespace component {
00009
00010 class SideBar {
00011 private:
```

```
00012     static constexpr int num_scenes = 8;
00013
00014     static constexpr int sidebar_width = constants::sidebar_width;
00015     static constexpr int button_height = 50;
00016
00017     static constexpr const char* sidebar_labels =
00018         "Back to Menu\n"
00019         "Array\n"
00020         "Dynamic Array\n"
00021         "Linked List\n"
00022         "Doubly Linked List\n"
00023         "Circular Linked List\n"
00024         "Stack\n"
00025         "Queue";
00026
00027     int m_next_scene{};
00028
00029 public:
00030     void render();
00031     void interact() const;
00032 };
00033
00034 }  // namespace component
00035
00036 #endif  // COMPONENT_SIDEBAR_HPP_
```

7.17 src/component/text_input.cpp File Reference

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

Include dependency graph for text_input.cpp:



Namespaces

- namespace component

7.18 text_input.cpp

Go to the documentation of this file.

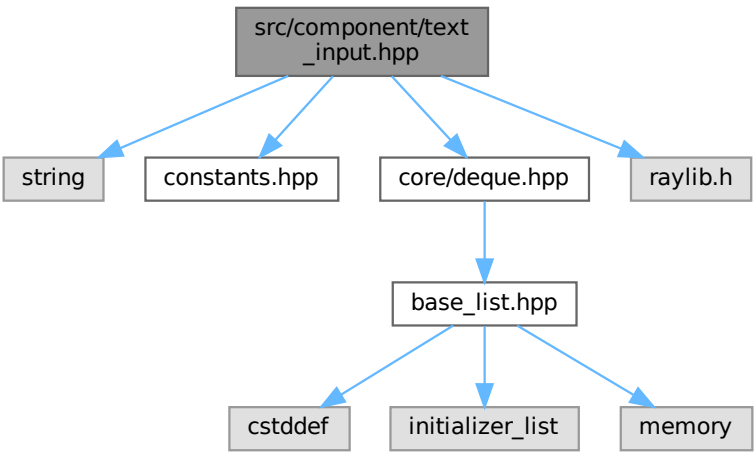
```
00001 #include "text_input.hpp"
00002
```

```
00003 #include "core/deque.hpp"
00004 #include "raygui.h"
00005 #include "utils.hpp"
00006
00007 namespace component {
00008
00009 void TextInput::render(float& options_head, float head_offset) {
00010     Rectangle shape{options_head, constants::scene_height - size.y, size.x,
00011                     size.y};
00012
00013     if (GuiTextBox(shape, static_cast<char*>(m_text_input), size.y,
00014                    m_is_active)) {
00015         m_is_active ^= 1;
00016     }
00017
00018     options_head += (size.x + head_offset);
00019 }
00020
00021 core::Deque<int> TextInput::extract_values() {
00022     core::Deque<int> nums = utils::str_extract_data(m_text_input);  // NOLINT
00023     return nums;
00024 }
00025
00026 }  // namespace component
```

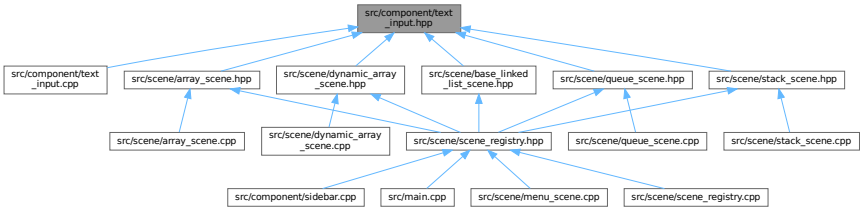
7.19 src/component/text_input.hpp File Reference

```
#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

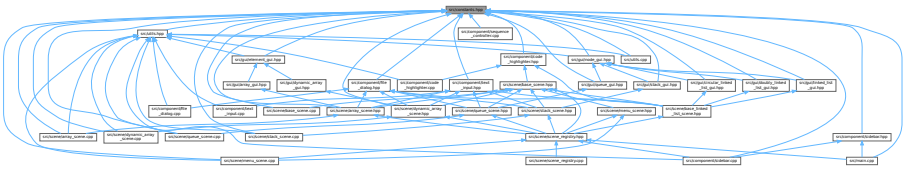
7.20 text_input.hpp

Go to the documentation of this file.

```
00001 #ifndef COMPONENT_TEXT_INPUT_HPP_
00002 #define COMPONENT_TEXT_INPUT_HPP_
00003
00004 #include <string>
00005
00006 #include "constants.hpp"
00007 #include "core/deque.hpp"
00008 #include "raylib.h"
00009
00010 namespace component {
00011
00012 class TextInput {
00013 private:
00014     char m_text_input[constants::text_buffer_size] = "";  // NOLINT
00015     bool m_is_active{};
00016
00017 public:
00018     static constexpr Vector2 size{200, 50};
00019
00020     void render(float& options_head, float head_offset);
00021     core::Deque<int> extract_values();
00022 };
00023
00024 }  // namespace component
00025
00026 #endif  // COMPONENT_TEXT_INPUT_HPP_
```

7.21 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

7.22 constants.hpp

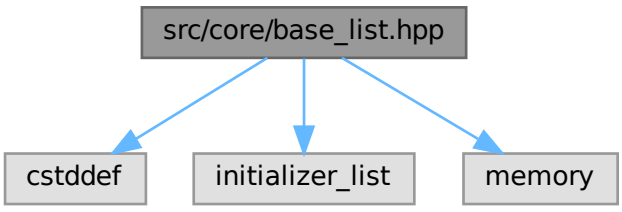
Go to the documentation of this file.

```
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 }  // namespace constants
00021
00022 #endif  // CONSTANTS_HPP_
```

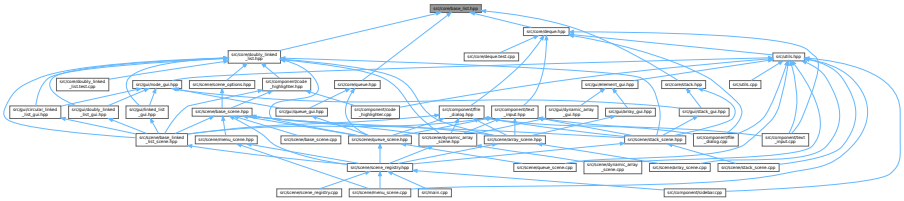
7.23 src/core/base_list.hpp File Reference

```
#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

7.24 base_list.hpp

Go to the documentation of this file.

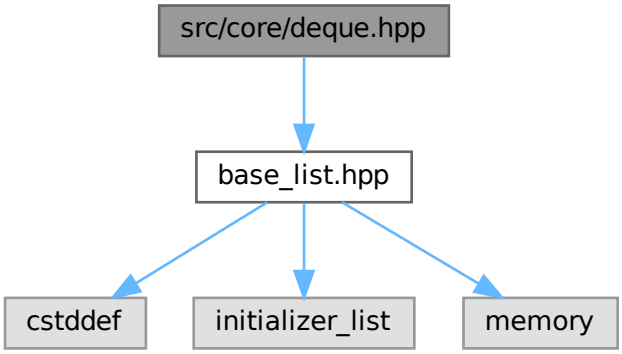
```
00001 #ifndef CORE_BASE_LIST_HPP_
00002 #define CORE_BASE_LIST_HPP_
00003
00004 #include <cstddef>
00005 #include <initializer_list>
00006 #include <memory>
00007
00008 namespace core {
00009
00010 template<typename T>
00011 class BaseList {
00012 protected:
00013     struct Node;
00014     using 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 }
```

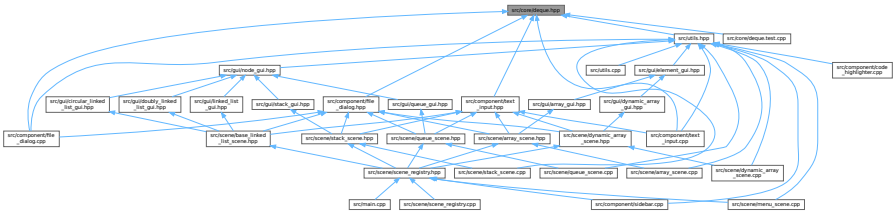
```
00133
00134 template<typename T>
00135 void BaseList<T>::copy_data(const BaseList& rhs) {
00136     for (Node_ptr ptr = rhs.m_head; ptr != nullptr; ptr = ptr->next) {
00137         push_back(ptr->data);
00138     }
00139 }
00140
00141 template<typename T>
00142 void BaseList<T>::push_back(const T& 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_
```

7.25 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

7.26 deque.hpp

Go to the documentation of this file.

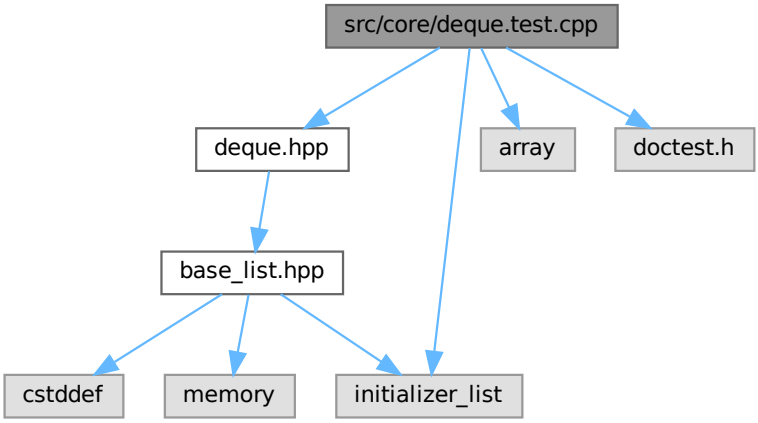
```
00001 #ifndef CORE_DEQUE_HPP_
00002 #define CORE_DEQUE_HPP_
00003
00004 #include "base_list.hpp"
00005
00006 namespace core {
```

```
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_
```

7.27 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}

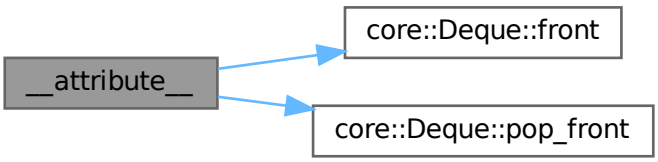
7.27.1 Function Documentation

7.27.1.1 __attribute__()

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

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

Here is the call graph for this function:

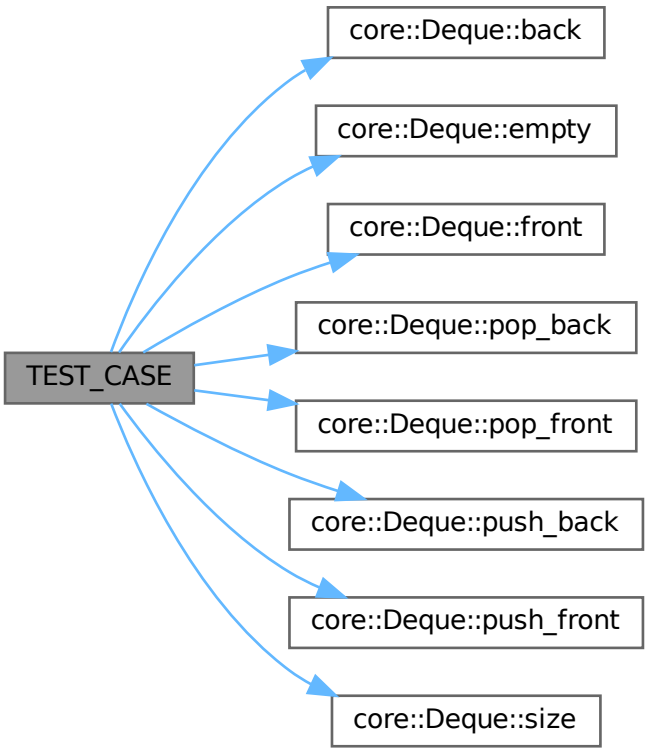


7.27.1.2 TEST_CASE() [1/2]

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

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

Here is the call graph for this function:



7.27.1.3 TEST_CASE() [2/2]

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

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

7.27.2 Variable Documentation

7.27.2.1 list

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

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

7.28 deque.test.cpp

Go to the documentation of this file.

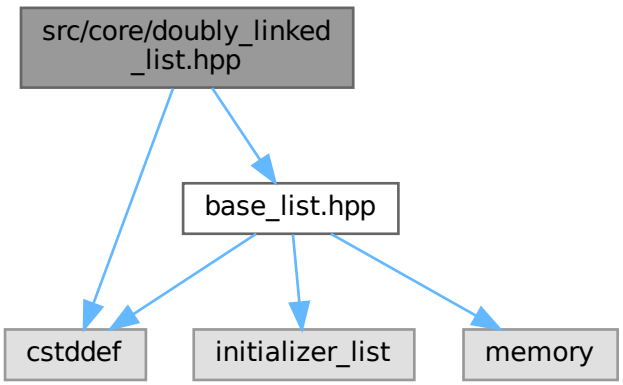
```
00001 #include "deque.hpp"
00002
00003 #include <array>
00004 #include <initializer_list>
00005
00006 #include "doctest.h"
00007
00008 TEST_CASE("core::Deque functionality") {
00009     core::Deque<int> deque;
00010     CHECK(deque.empty());
00011
00012     deque.push_back(2);
00013     deque.push_back(3);
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 }
```

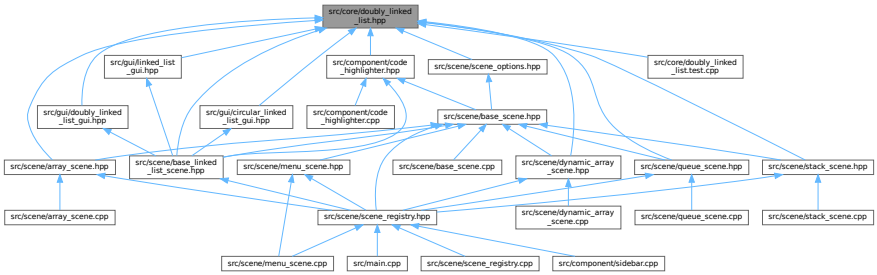
7.29 src/core/doubly_linked_list.hpp File Reference

```
#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

7.30 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
```

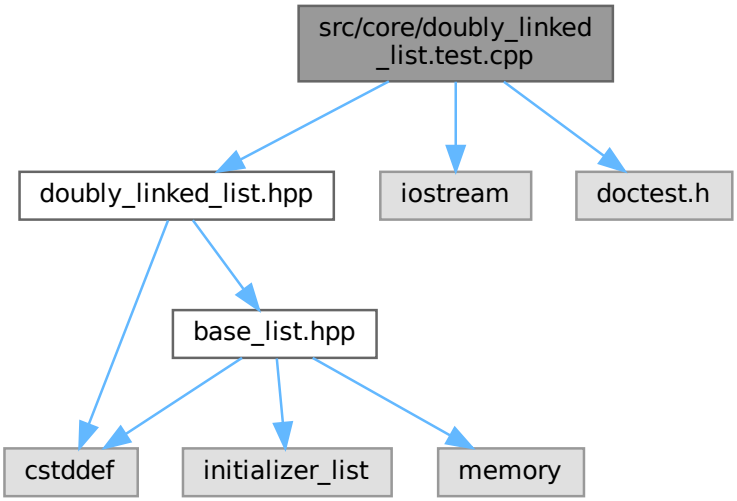
```
00043     void clear();
00044 };
00045
00046 template<typename T>
00047 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::internal_search(
00048     const T& elem) {
00049     Node_ptr ptr{m_head};
00050
00051     while (ptr != nullptr) {
00052         if (ptr->data == elem) {
00053             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_
```

7.31 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")

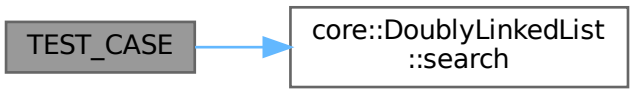
7.31.1 Function Documentation

7.31.1.1 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:



7.32 doubly_linked_list.test.cpp

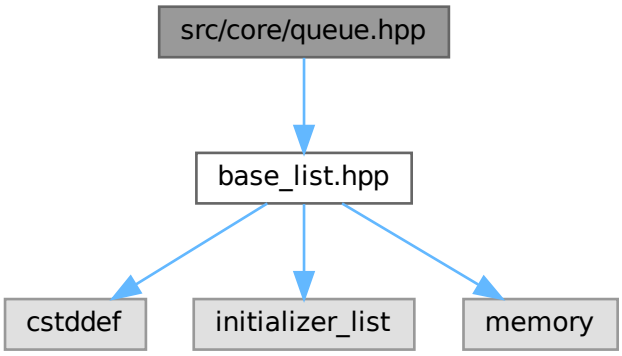
Go to the documentation of this file.

```
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 }
```

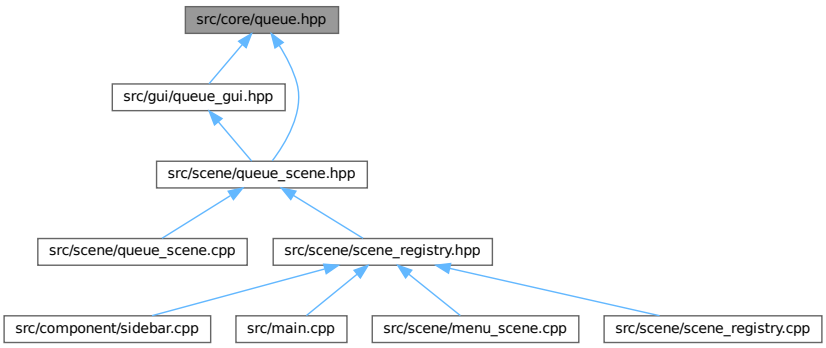
7.33 src/core/queue.hpp File Reference

```
#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

7.34 queue.hpp

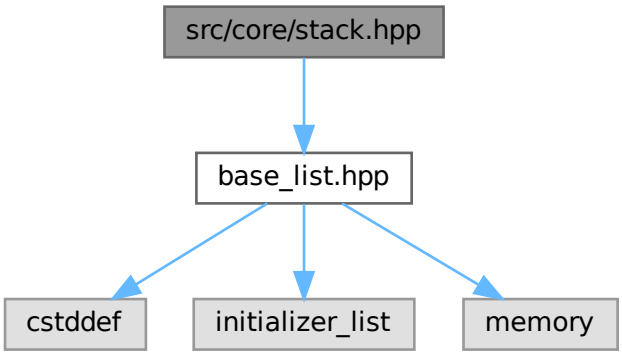
Go to the documentation of this file.

```
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_
```

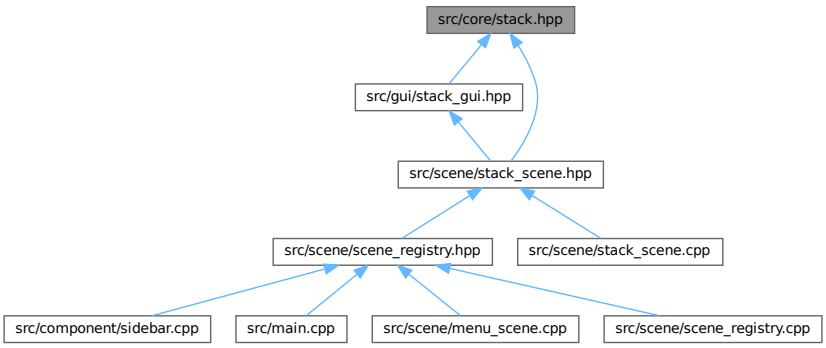
7.35 src/core/stack.hpp File Reference

```
#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

7.36 stack.hpp

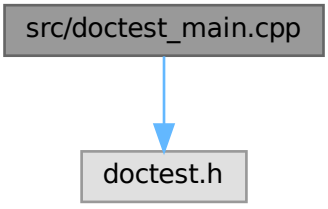
Go to the documentation of this file.

```
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_
```

7.37 src/doctest_main.cpp File Reference

`#include "doctest.h"`

Include dependency graph for doctest_main.cpp:



Macros

- #define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN

7.37.1 Macro Definition Documentation

7.37.1.1 DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN

```
#define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN
```

Definition at line 1 of file doctest_main.cpp.

7.38 doctest_main.cpp

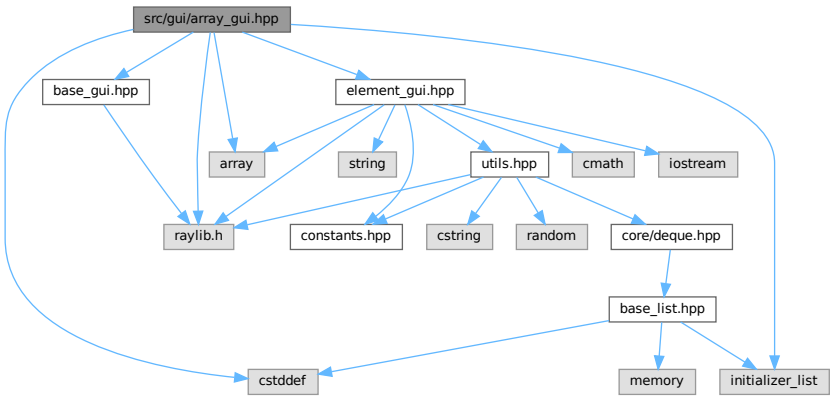
Go to the documentation of this file.

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

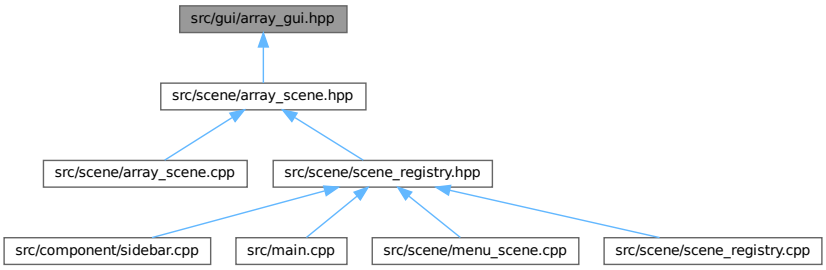
7.39 src/gui/array_gui.hpp File Reference

```
#include <array>
#include <cstddef>
#include <initializer_list>
#include "base_gui.hpp"
#include "element_gui.hpp"
#include "raylib.h"
```

Include dependency graph for array_gui.hpp:



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



Classes

- class gui::GuiArray< T, N >

Namespaces

- namespace gui

7.40 array_gui.hpp

Go to the documentation of this file.

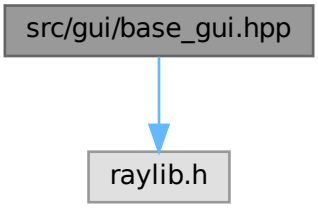
```
00001 #ifndef GUI_ARRAY_GUI_HPP_
00002 #define GUI_ARRAY_GUI_HPP_
00003
00004 #include <array>
00005 #include <cstddef>
00006 #include <initializer_list>
00007
00008 #include "base_gui.hpp"
00009 #include "element_gui.hpp"
00010 #include "raylib.h"
00011
00012 namespace gui {
00013
00014 template<typename T, std::size_t N>
00015 class GuiArray : public internal::Base {
00016 private:
00017     static constexpr Vector2 head_pos{
00018         constants::sidebar_width +
00019             (constants::scene_width - constants::sidebar_width) / 2.0F -
00020             15 * GuiElement<T>::side,
00021         constants::scene_height / 2.0F};
00022
00023     std::array<GuiElement<T>, N> m_array{};
00024
00025     void render_link(Vector2 src, Vector2 dest) override;
00026
00027 public:
00028     GuiArray();
00029     GuiArray(std::array<GuiElement<T>, N>&& init_list);
00030     void update() override;
00031     void render() override;
00032
00033     T& operator[](std::size_t idx);
00034     T operator[](std::size_t idx) const;
00035
00036     void set_color(std::size_t idx, Color color);
00037 };
00038
00039 template<typename T, std::size_t N>
00040 GuiArray<T, N>::GuiArray() {
```

```
00041     for (std::size_t i = 0; i < N; ++i) {
00042         m_array[i] = GuiElement<T>{0, i};
00043         m_array[i].set_color(BLACK);
00044     }
00045 }
00046
00047 template<typename T, std::size_t N>
00048 GuiArray<T, N>::GuiArray(std::array<GuiElement<T>, N>&& init_list)
00049     : m_array{init_list} {}
00050
00051 template<typename T, std::size_t N>
00052 void GuiArray<T, N>::render_link(Vector2 src, Vector2 dest) {}
00053
00054 template<typename T, std::size_t N>
00055 void GuiArray<T, N>::render() {
00056     update();
00057
00058     for (std::size_t i = 0; i < N; ++i) {
00059         m_array[i].render();
00060     }
00061 }
00062
00063 template<typename T, std::size_t N>
00064 void GuiArray<T, N>::update() {
00065     // TODO: if not outdated then return
00066
00067     for (std::size_t i = 0; i < N; ++i) {
00068         m_array[i].set_target_pos(
00069             {head_pos.x + 4 * GuiElement<T>::side * i, head_pos.y});
00070     }
00071 }
00072
00073 template<typename T, std::size_t N>
00074 T& GuiArray<T, N>::operator[](std::size_t idx) {
00075     return m_array[idx].get_value();
00076 }
00077
00078 template<typename T, std::size_t N>
00079 T GuiArray<T, N>::operator[](std::size_t idx) const {
00080     return m_array[idx].get_value();
00081 }
00082
00083 template<typename T, std::size_t N>
00084 void GuiArray<T, N>::set_color(std::size_t idx, Color color) {
00085     m_array[idx].set_color(color);
00086 }
00087
00088 }  // namespace gui
00089
00090 #endif  // GUI_ARRAY_GUI_HPP_
```

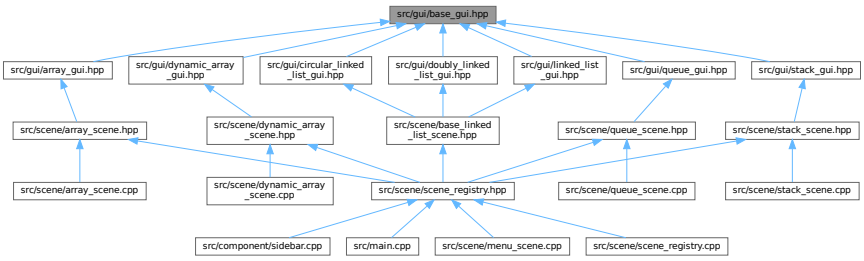
7.41 src/gui/base_gui.hpp File Reference

`#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

7.42 base_gui.hpp

Go to the documentation of this file.

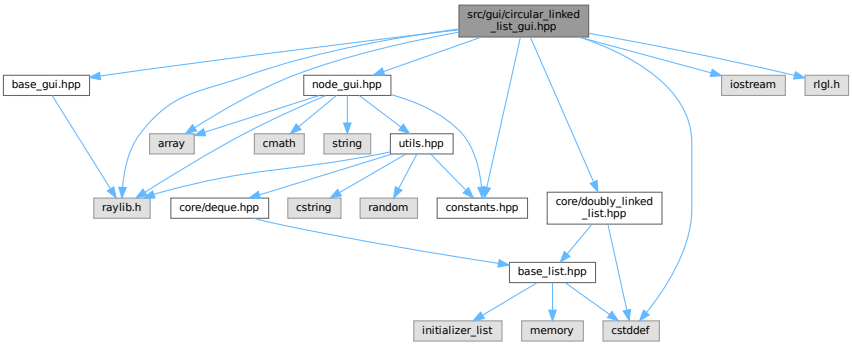
```
00001 #ifndef GUI_BASE_GUI_HPP_
00002 #define GUI_BASE_GUI_HPP_
00003
00004 #include "raylib.h"
00005
00006 namespace gui::internal {
00007
00008 class Base {
00009     virtual void render_link(Vector2 src, Vector2 dest) = 0;
00010
00011 public:
00012     Base() = default;
00013     Base(const Base&) = 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_
```

7.43 src/gui/circular_linked_list_gui.hpp File Reference

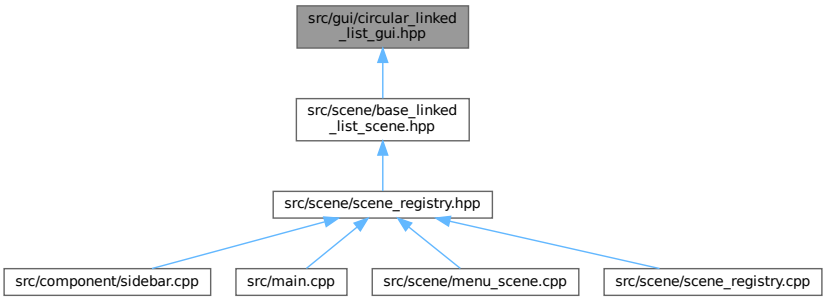
```
#include <array>
#include <cstddef>
#include <iostream>
#include "base_gui.hpp"
```

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

Include 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

7.44 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
```

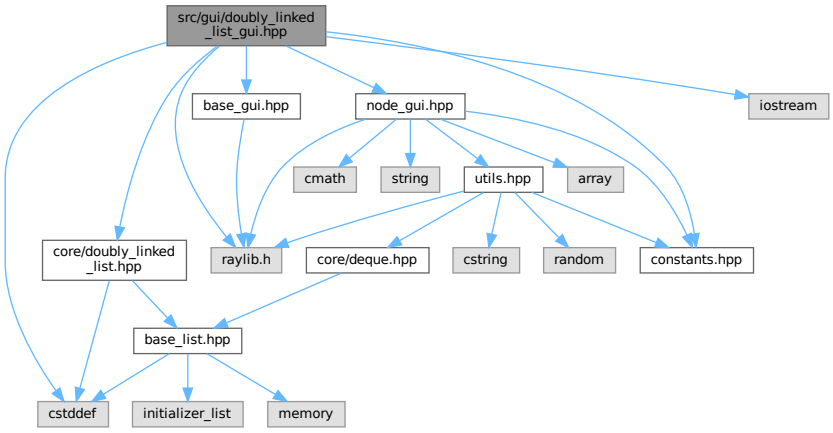
```
00004 #include <array>
00005 #include <cstddef>
00006 #include <iostream>
00007
00008 #include "base_gui.hpp"
00009 #include "constants.hpp"
00010 #include "core/doubly_linked_list.hpp"
00011 #include "node_gui.hpp"
00012 #include "raylib.h"
00013 #include "rlgl.h"
00014
00015 namespace gui {
00016
00017 template<typename T>
00018 class GuiCircularLinkedList : 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::sidebar_width +
00025             (constants::scene_width - constants::sidebar_width) / 2.0F -
00026             15 * GuiNode<T>::radius,
00027         constants::scene_height / 2.0F};
00028
00029     using Base::m_head;
00030     using Base::m_tail;
00031
00032     void render_link(Vector2 src, Vector2 dest) override;
00033     void render_back_link();
00034
00035 public:
00036     using Base::Base;
00037
00038     using Base::empty;
00039     using Base::size;
00040
00041     void insert(std::size_t index, const T& elem);
00042
00043     void update() override;
00044     void render() override;
00045 };
00046
00047 template<typename T>
00048 void GuiCircularLinkedList<T>::insert(std::size_t index, const T& elem) {
00049     Base::insert(index, GuiNode{elem});
00050 }
00051
00052 template<typename T>
00053 void GuiCircularLinkedList<T>::render_link(Vector2 src, Vector2 dest) {
00054     constexpr int radius = GuiNode<T>::radius;
00055     constexpr float scaled_len = radius / 8.0F;
00056
00057     // straight line
00058     Vector2 link_pos{src.x + radius, src.y - scaled_len};
00059     Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};
00060
00061     // arrow
00062     constexpr int arrow_size = scaled_len * 5;
00063     Vector2 head{dest.x - radius + scaled_len / 2, src.y};
00064     Vector2 side_top{head.x - arrow_size, head.y - arrow_size};
00065     Vector2 side_bot{head.x - arrow_size, head.y + arrow_size};
00066
00067     // draw both
00068     DrawRectangleV(link_pos, link_size, GRAY);
00069     DrawTriangle(head, side_top, side_bot, GRAY);
00070 }
00071
00072 template<typename T>
00073 void GuiCircularLinkedList<T>::render_back_link() {
00074     constexpr int num_points = 5;
00075     const Vector2 head_pos = m_head->data.get_current_pos();
00076     const Vector2 tail_pos = m_tail->data.get_current_pos();
00077     constexpr int radius = GuiNode<T>::radius;
00078     constexpr float scaled_len = radius / 8.0F;
00079
00080     std::array<Vector2, num_points> points{{
00081         tail_pos,
00082         {tail_pos.x + 2 * radius, tail_pos.y},
00083         {tail_pos.x + 2 * radius, tail_pos.y + 3 * radius},
00084         {head_pos.x, tail_pos.y + 3 * radius},
00085         head_pos,
00086     }};
00087
00088     constexpr int arrow_size = scaled_len * 5;
00089     Vector2 head{head_pos.x, head_pos.y + radius - scaled_len / 2};
00090     Vector2 side_left{head.x - arrow_size, head.y + arrow_size};
```

```
00091     Vector2 side_right{head.x + arrow_size, head.y + arrow_size};
00092
00093     rlSetLineWidth(2 * scaled_len);
00094     DrawLineStrip(points.data(), num_points, GRAY);
00095     DrawTriangle(head, side_left, side_right, GRAY);
00096 }
00097
00098 template<typename T>
00099 void GuiCircularLinkedList<T>::render() {
00100     update();
00101
00102     render_back_link();
00103     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00104         if (ptr->next != nullptr) {
00105             render_link(ptr->data.get_current_pos(),
00106                         ptr->next->data.get_current_pos());
00107         }
00108
00109         ptr->data.render();
00110     }
00111 }
00112
00113 template<typename T>
00114 void GuiCircularLinkedList<T>::update() {
00115     // TODO: if not outdated then return
00116
00117     std::size_t pos = 0;
00118
00119     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00120         ptr->data.set_target_pos(
00121             {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
00122         ++pos;
00123     }
00124 }
00125
00126 }  // namespace gui
00127
00128 #endif  // GUI_CIRCULAR_LINKED_LIST_GUI_HPP_
```

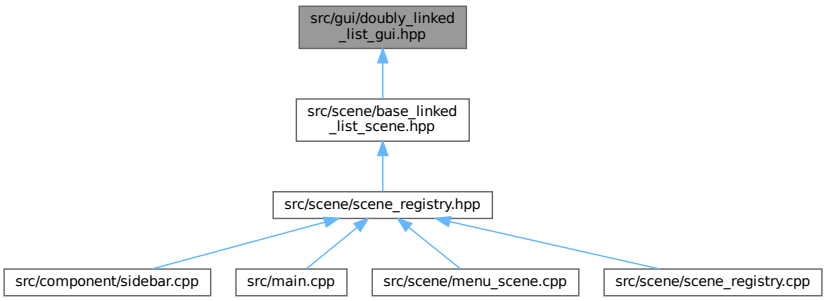
7.45 src/gui/doubly_linked_list_gui.hpp File Reference

```
#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 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

7.46 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
00013 namespace gui {
00014
00015 template<typename T>
00016 class GuiDoublyLinkedList : public core::DoublyLinkedList<GuiNode<T»,
00017                             public internal::Base {
00018 private:
00019     using Base = core::DoublyLinkedList<GuiNode<T>>;
00020
00021     static constexpr Vector2 head_pos{
00022         constants::sidebar_width +
00023             (constants::scene_width - constants::sidebar_width) / 2.0F -
00024             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     void insert(std::size_t index, const T& elem);
```

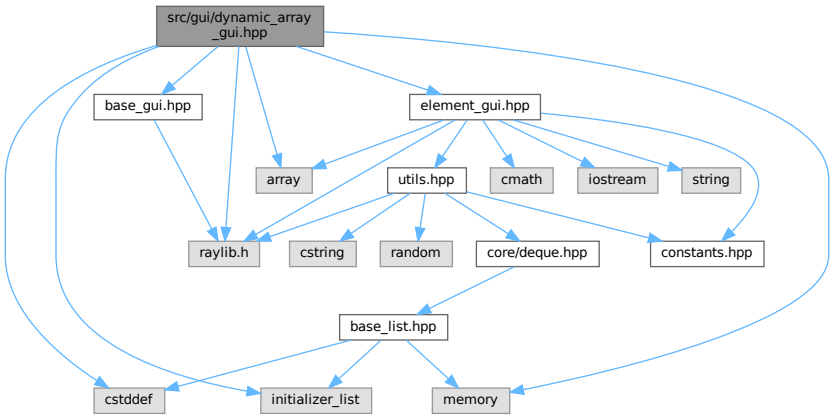
```
00039
00040     void update() override;
00041     void render() override;
00042 };
00043
00044 template<typename T>
00045 void GuiDoublyLinkedList<T>::insert(std::size_t index, const T& elem) {
00046     Base::insert(index, GuiNode{elem});
00047 }
00048
00049 template<typename T>
00050 void GuiDoublyLinkedList<T>::render_link(Vector2 src, Vector2 dest) {
00051     constexpr int radius = GuiNode<T>::radius;
00052     constexpr float scaled_len = radius / 8.0F;
00053
00054     // straight line
00055     Vector2 link_pos{src.x + radius, src.y - scaled_len};
00056     Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};
00057
00058     // right arrow
00059     constexpr int arrow_size = scaled_len * 5;
00060     Vector2 right_head{dest.x - radius + scaled_len / 2, src.y};
00061     Vector2 right_side_top{right_head.x - arrow_size,
00062                            right_head.y - arrow_size};
00063     Vector2 right_side_bot{right_head.x - arrow_size,
00064                            right_head.y + arrow_size};
00065
00066     // left arrow
00067     Vector2 left_head{src.x + radius - scaled_len / 2, src.y};
00068     Vector2 left_side_top{left_head.x + arrow_size, left_head.y - arrow_size};
00069     Vector2 left_side_bot{left_head.x + arrow_size, left_head.y + arrow_size};
00070
00071     // draw all
00072     DrawRectangleV(link_pos, link_size, GRAY);
00073     DrawTriangle(right_head, right_side_top, right_side_bot, GRAY);
00074     DrawTriangle(left_head, left_side_bot, left_side_top, GRAY);
00075 }
00076
00077 template<typename T>
00078 void GuiDoublyLinkedList<T>::render() {
00079     update();
00080
00081     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00082         if (ptr->next != nullptr) {
00083             render_link(ptr->data.get_current_pos(),
00084                         ptr->next->data.get_current_pos());
00085         }
00086
00087         ptr->data.render();
00088     }
00089 }
00090
00091 template<typename T>
00092 void GuiDoublyLinkedList<T>::update() {
00093     // TODO: if not outdated then return
00094
00095     std::size_t pos = 0;
00096
00097     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00098         ptr->data.set_target_pos(
00099             {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
00100         ++pos;
00101     }
00102 }
00103
00104 }  // namespace gui
00105
00106 #endif  // GUI_DOUBLY_LINKED_LIST_GUI_HPP_
```

7.47 src/gui/dynamic_array_gui.hpp File Reference

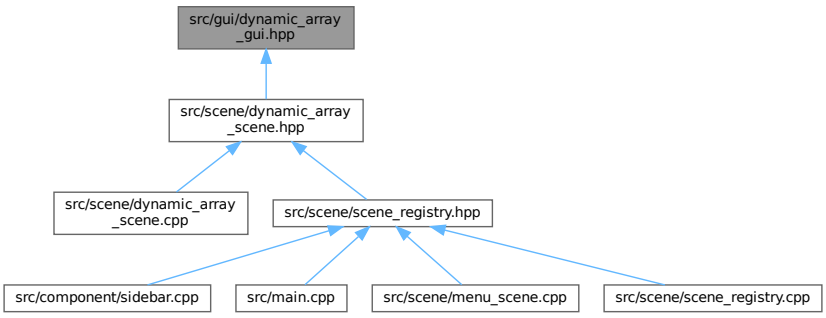
```
#include <array>
#include <cstddef>
#include <initializer_list>
#include <memory>
#include "base_gui.hpp"
#include "element_gui.hpp"
```

```
#include "raylib.h"
```

Include 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

7.48 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
00013 namespace gui {
00014
00015 template<typename T>
00016 class GuiDynamicArray : public internal::Base {
00017 private:
00018     static constexpr Vector2 head_pos{
00019         constants::sidebar_width +
00020             (constants::scene_width - constants::sidebar_width) / 2.0F -
00021             15 * GuiElement<T>::side,
00022         constants::scene_height / 2.0F};
00023
00024     std::size_t m_capacity{2};
00025     std::size_t m_size{};
00026     GuiElement<T>* m_ptr{nullptr};
00027
00028     void render_link(Vector2 src, Vector2 dest) override;
00029
00030 public:
00031     GuiDynamicArray();
00032     GuiDynamicArray(std::initializer_list<T> init_list);
00033     GuiDynamicArray(const GuiDynamicArray& other);
00034     GuiDynamicArray(GuiDynamicArray&& other) noexcept;
00035     GuiDynamicArray& operator=(const GuiDynamicArray& other);
00036     GuiDynamicArray& operator=(GuiDynamicArray&& other) noexcept;
00037     ~GuiDynamicArray() override;
00038
00039     void update() override;
00040     void render() override;
00041
00042     T& operator[](std::size_t idx);
00043     T operator[](std::size_t idx) const;
00044
00045     void set_color(std::size_t idx, Color color);
00046     void realloc(std::size_t capacity);
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>::realloc(std::size_t capacity) {
00057     if (m_capacity > capacity) {
00058         return;
00059     }
00060
00061     while (m_capacity < capacity) {
00062         m_capacity *= 2;
00063     }
00064
00065     auto* new_ptr = new GuiElement<T>[m_capacity];
00066     for (auto i = 0; i < m_size; ++i) {
00067         new_ptr[i] = m_ptr[i];
00068     }
00069     for (auto i = m_size; i < m_capacity; ++i) {
00070         new_ptr[i].set_index(i);
00071     }
00072
00073     delete[] m_ptr;
00074     m_ptr = new_ptr;
00075 }
00076
00077 template<typename T>
00078 GuiDynamicArray<T>::GuiDynamicArray() : m_ptr{new GuiElement<T>[m_capacity]} {
00079     for (auto i = 0; i < m_capacity; ++i) {
00080         m_ptr[i].set_index(i);
00081     }
00082 }
00083
00084 template<typename T>
00085 GuiDynamicArray<T>::GuiDynamicArray(std::initializer_list<T> init_list)
00086     : m_size{init_list.size()}, m_ptr{new GuiElement<T>[m_capacity]} {
00087     realloc(m_size);
00088
00089     for (std::size_t idx = 0; auto elem : init_list) {
00090         *(m_ptr + idx).set_value(elem);
00091         *(m_ptr + idx).set_color(BLACK);
```

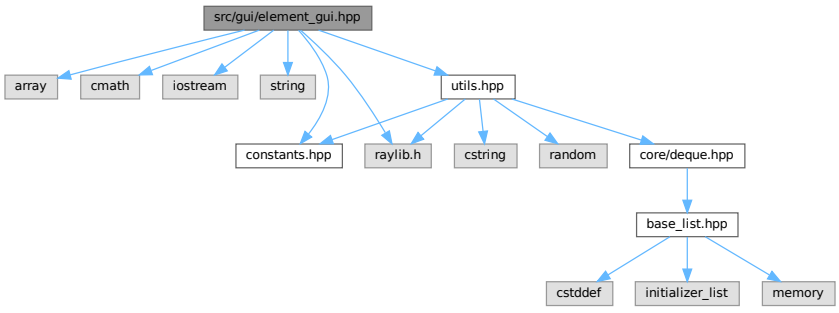
```
00092     }
00093 }
00094
00095 template<typename T>
00096 GuiDynamicArray<T>::GuiDynamicArray(const GuiDynamicArray<T>& other)
00097     : m_capacity{other.m_capacity},
00098       m_size{other.m_size},
00099       m_ptr{new GuiElement<T>[m_capacity]} {
00100     for (auto i = 0; i < m_capacity; ++i) {
00101         m_ptr[i] = other.m_ptr[i];
00102     }
00103 }
00104
00105 template<typename T>
00106 GuiDynamicArray<T>::GuiDynamicArray(GuiDynamicArray<T>&& other) noexcept
00107     : m_capacity{other.m_capacity}, m_size{other.m_size}, m_ptr{other.m_ptr} {
00108     other.m_capacity = 0;
00109     other.m_size = 0;
00110     other.m_ptr = nullptr;
00111 }
00112
00113 template<typename T>
00114 GuiDynamicArray<T>& GuiDynamicArray<T>::operator=(
00115     const GuiDynamicArray<T>& other) {
00116     if (&other != this) {
00117         m_capacity = other.m_capacity;
00118         m_size = other.m_size;
00119
00120         m_ptr = new GuiDynamicArray<T>[m_capacity];
00121         for (auto i = 0; i < m_capacity; ++i) {
00122             m_ptr[i] = other.m_ptr[i];
00123         }
00124     }
00125
00126     return *this;
00127 }
00128
00129 template<typename T>
00130 GuiDynamicArray<T>& GuiDynamicArray<T>::operator=(
00131     GuiDynamicArray&& other) noexcept {
00132     m_capacity = other.m_capacity;
00133     m_size = other.m_size;
00134     m_ptr = other.m_ptr;
00135
00136     other.m_capacity = 0;
00137     other.m_size = 0;
00138     other.m_ptr = nullptr;
00139
00140     return *this;
00141 }
00142
00143 template<typename T>
00144 GuiDynamicArray<T>::~GuiDynamicArray() {
00145     delete[] m_ptr;
00146 }
00147
00148 template<typename T>
00149 void GuiDynamicArray<T>::render_link(Vector2 src, Vector2 dest) {}
00150
00151 template<typename T>
00152 void GuiDynamicArray<T>::render() {
00153     update();
00154
00155     std::size_t idx = 0;
00156
00157     for (std::size_t i = 0; i < m_capacity; ++i) {
00158         m_ptr[i].render();
00159     }
00160 }
00161
00162 template<typename T>
00163 void GuiDynamicArray<T>::update() {
00164     // TODO: if not outdated then return
00165
00166     for (std::size_t i = 0; i < m_capacity; ++i) {
00167         m_ptr[i].set_target_pos(
00168             {head_pos.x + 4 * GuiElement<T>::side * i, head_pos.y});
00169     }
00170 }
00171
00172 template<typename T>
00173 T& GuiDynamicArray<T>::operator[](std::size_t idx) {
00174     return m_ptr[idx].get_value();
00175 }
00176
00177 template<typename T>
00178 T GuiDynamicArray<T>::operator[](std::size_t idx) const {
```

```
00179     return m_ptr[idx].get_value();
00180 }
00181
00182 template<typename T>
00183 void GuiDynamicArray<T>::set_color(std::size_t idx, Color color) {
00184     m_ptr[idx].set_color(color);
00185 }
00186
00187 template<typename T>
00188 std::size_t GuiDynamicArray<T>::capacity() const {
00189     return m_capacity;
00190 }
00191
00192 template<typename T>
00193 std::size_t GuiDynamicArray<T>::size() const {
00194     return m_size;
00195 }
00196
00197 template<typename T>
00198 void GuiDynamicArray<T>::push(const T& value) {
00199     if (m_size == m_capacity) {
00200         realloc(m_size + 1);
00201     }
00202
00203     m_ptr[m_size].set_color(BLACK);
00204     m_ptr[m_size].set_value(value);
00205     ++m_size;
00206 }
00207
00208 template<typename T>
00209 void GuiDynamicArray<T>::pop() {
00210     if (m_size >= 1) {
00211         m_ptr[m_size - 1].set_color(GRAY);
00212         m_ptr[m_size - 1].set_value(0);
00213         --m_size;
00214     }
00215 }
00216
00217 }  // namespace gui
00218
00219 #endif  // GUI_DYNAMIC_ARRAY_GUI_HPP_
```

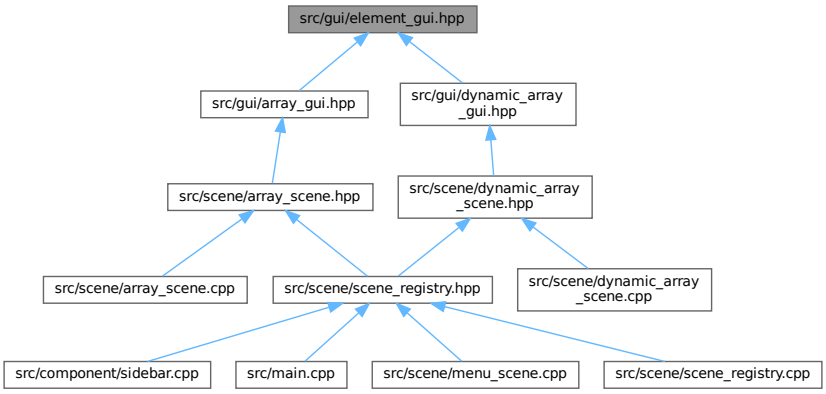
7.49 src/gui/element_gui.hpp File Reference

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

7.50 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 "utils.hpp"
00012
00013 namespace gui {
00014
00015 template<typename T>
00016 class GuiElement {
00017 private:
00018     T m_value{};
00019     std::size_t m_index{};
00020
00021     Vector2 m_current_pos{init_pos};
00022     Vector2 m_target_pos{};
00023     bool m_is_outdated{false};
00024     static constexpr float eps = 1e-3;
00025     Color m_color{GRAY};
00026
00027 public:
00028     static constexpr int side = 20;
00029     static constexpr Vector2 init_pos{
00030         constants::sidebar_width +
00031             static_cast<float>(constants::scene_width -
00032                                constants::sidebar_width) /
00033                 2,
00034         0};
```

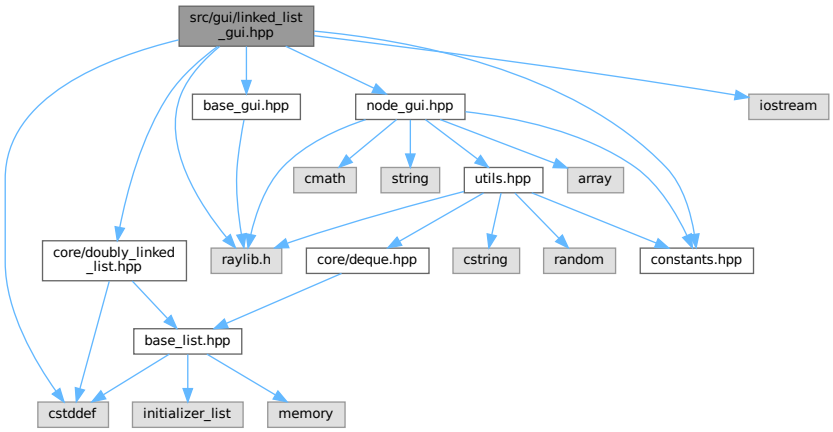
```
00035
00036     GuiElement() = default;
00037     GuiElement(const T& value, std::size_t index);
00038
00039     void render();
00040     void set_target_pos(Vector2 pos);
00041     void set_color(Color color);
00042     [[nodiscard]] Vector2 get_target_pos() const;
00043     [[nodiscard]] Vector2 get_current_pos() const;
00044     [[nodiscard]] bool check_outdated() const;
00045
00046     T& get_value();
00047     T get_value() const;
00048     void set_value(const T& value);
00049     void set_index(std::size_t index);
00050 };
00051
00052 template<typename T>
00053 GuiElement<T>::GuiElement(const T& value, std::size_t index)
00054     : m_value{value}, m_index{index} {}
00055
00056 template<typename T>
00057 void GuiElement<T>::render() {
00058     // if (m_is_outdated) {
00059     //     float diff_x = m_target_pos.x - m_current_pos.x;
00060     //     float diff_y = m_target_pos.y - m_current_pos.y;
00061
00062     //     if (std::fabs(diff_x) < eps) {
00063     //         diff_x = 0;
00064     //     }
00065
00066     //     if (std::fabs(diff_y) < eps) {
00067     //         diff_y = 0;
00068     //     }
00069
00070     //     if (diff_x == 0 && diff_y == 0) {
00071     //         m_is_outdated = false;
00072     //     } else {
00073     //         m_current_pos.x +=
00074     //             diff_x / constants::frames_per_second * constants::ani_speed;
00075     //         m_current_pos.y +=
00076     //             diff_y / constants::frames_per_second * constants::ani_speed;
00077     //     }
00078     // }
00079
00080     constexpr int label_font_size = 25;
00081     constexpr int label_font_spacing = 2;
00082     const std::string label = std::to_string(m_value);
00083     const std::string index = std::to_string(m_index);
00084
00085     const Vector2 label_size =
00086         utils::MeasureText(label.c_str(), label_font_size, label_font_spacing);
00087
00088     const Vector2 label_pos{m_current_pos.x - label_size.x / 2,
00089                             m_current_pos.y - label_size.y / 2};
00090
00091     const Vector2 index_size =
00092         utils::MeasureText(index.c_str(), label_font_size, label_font_spacing);
00093
00094     const Vector2 index_pos{m_current_pos.x - index_size.x / 2,
00095                             m_current_pos.y - 2 * side - index_size.y / 2};
00096
00097     DrawRectangle(m_current_pos.x - side,  // NOLINT
00098                   m_current_pos.y - side,  // NOLINT
00099                   2 * side, 2 * side, m_color);
00100
00101     utils::DrawText(label.c_str(), label_pos, WHITE, label_font_size,
00102                     label_font_spacing);
00103
00104     utils::DrawText(index.c_str(), index_pos, BLACK, label_font_size,
00105                     label_font_spacing);
00106 }
00107
00108 template<typename T>
00109 void GuiElement<T>::set_target_pos(Vector2 pos) {
00110     // m_target_pos = pos;
00111     // m_is_outdated = true;
00112     m_current_pos = pos;
00113 }
00114
00115 template<typename T>
00116 void GuiElement<T>::set_color(Color color) {
00117     m_color = color;
00118 }
00119
00120 template<typename T>
00121 Vector2 GuiElement<T>::get_target_pos() const {
```

```
00122     return m_target_pos;
00123 }
00124
00125 template<typename T>
00126 Vector2 GuiElement<T>::get_current_pos() const {
00127     return m_current_pos;
00128 }
00129
00130 template<typename T>
00131 bool GuiElement<T>::check_outdated() const {
00132     return m_is_outdated;
00133 }
00134
00135 template<typename T>
00136 T& GuiElement<T>::get_value() {
00137     return m_value;
00138 }
00139
00140 template<typename T>
00141 T GuiElement<T>::get_value() const {
00142     return m_value;
00143 }
00144
00145 template<typename T>
00146 void GuiElement<T>::set_value(const T& value) {
00147     m_value = value;
00148 }
00149
00150 template<typename T>
00151 void GuiElement<T>::set_index(std::size_t index) {
00152     m_index = index;
00153 }
00154
00155 }  // namespace gui
00156
00157 #endif  // GUI_ELEMENT_GUI_HPP_
```

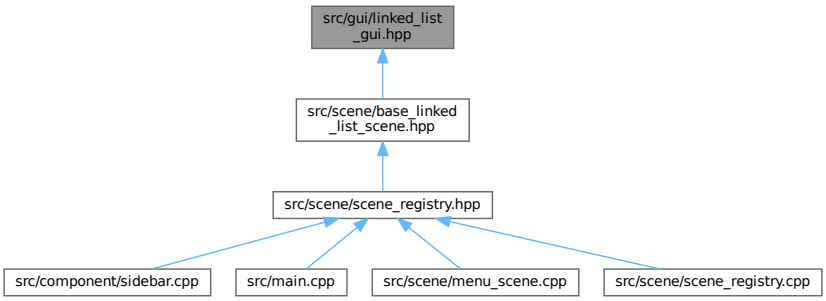
7.51 src/gui/linked_list_gui.hpp File Reference

```
#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 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

7.52 linked_list_gui.hpp

Go to the documentation of this file.

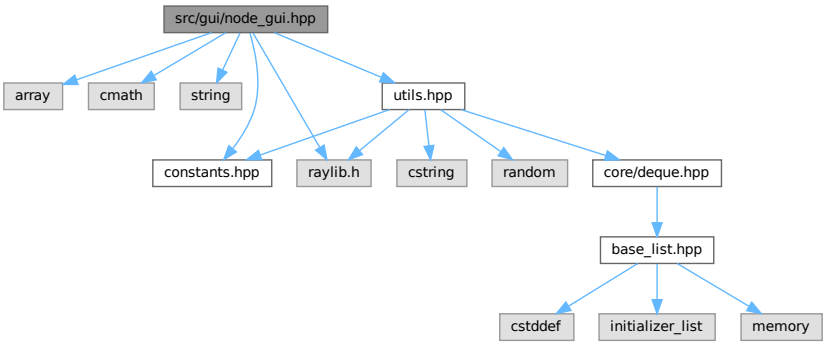
```
00001 #ifndef GUI_LINKED_LIST_GUI_HPP_
00002 #define GUI_LINKED_LIST_GUI_HPP_
00003
00004 #include <cstddef>
00005 #include <iostream>
00006
00007 #include "base_gui.hpp"
00008 #include "constants.hpp"
00009 #include "core/doubly_linked_list.hpp"
00010 #include "node_gui.hpp"
00011 #include "raylib.h"
00012
00013 namespace gui {
00014
00015 template<typename T>
00016 class GuiLinkedList : public core::DoublyLinkedList<GuiNode<T»,
00017                       public internal::Base {
00018 private:
00019     using Base = core::DoublyLinkedList<GuiNode<T>>;
00020
00021     static constexpr Vector2 head_pos{
00022         constants::sidebar_width +
00023             (constants::scene_width - constants::sidebar_width) / 2.0F -
00024             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     void insert(std::size_t index, const T& elem);
```

```
00039
00040     void update() override;
00041     void render() override;
00042 };
00043
00044 template<typename T>
00045 void GuiLinkedList<T>::insert(std::size_t index, const T& elem) {
00046     Base::insert(index, GuiNode{elem});
00047 }
00048
00049 template<typename T>
00050 void GuiLinkedList<T>::render_link(Vector2 src, Vector2 dest) {
00051     constexpr int radius = GuiNode<T>::radius;
00052     constexpr float scaled_len = radius / 8.0F;
00053
00054     // straight line
00055     Vector2 link_pos{src.x + radius, src.y - scaled_len};
00056     Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};
00057
00058     // arrow
00059     constexpr int arrow_size = scaled_len * 5;
00060     Vector2 head{dest.x - radius + scaled_len / 2, src.y};
00061     Vector2 side_top{head.x - arrow_size, head.y - arrow_size};
00062     Vector2 side_bot{head.x - arrow_size, head.y + arrow_size};
00063
00064     // draw both
00065     DrawRectangleV(link_pos, link_size, GRAY);
00066     DrawTriangle(head, side_top, side_bot, GRAY);
00067 }
00068
00069 template<typename T>
00070 void GuiLinkedList<T>::render() {
00071     update();
00072
00073     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00074         if (ptr->next != nullptr) {
00075             render_link(ptr->data.get_current_pos(),
00076                         ptr->next->data.get_current_pos());
00077         }
00078
00079         ptr->data.render();
00080     }
00081 }
00082
00083 template<typename T>
00084 void GuiLinkedList<T>::update() {
00085     // TODO: if not outdated then return
00086
00087     std::size_t pos = 0;
00088
00089     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00090         ptr->data.set_target_pos(
00091             {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
00092         ++pos;
00093     }
00094 }
00095
00096 }  // namespace gui
00097
00098 #endif  // GUI_LINKED_LIST_GUI_HPP_
```

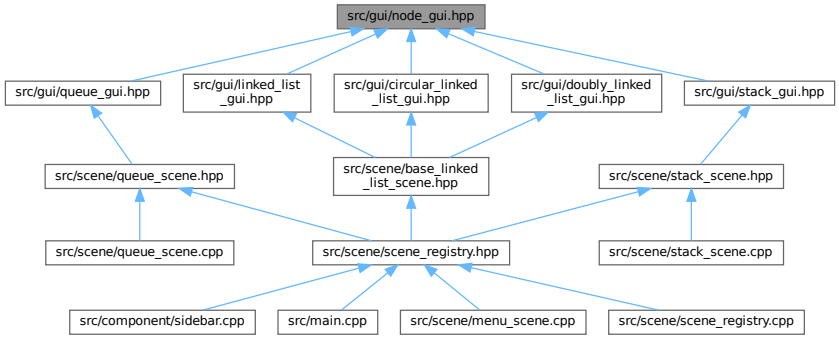
7.53 src/gui/node_gui.hpp File Reference

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

7.54 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"
```

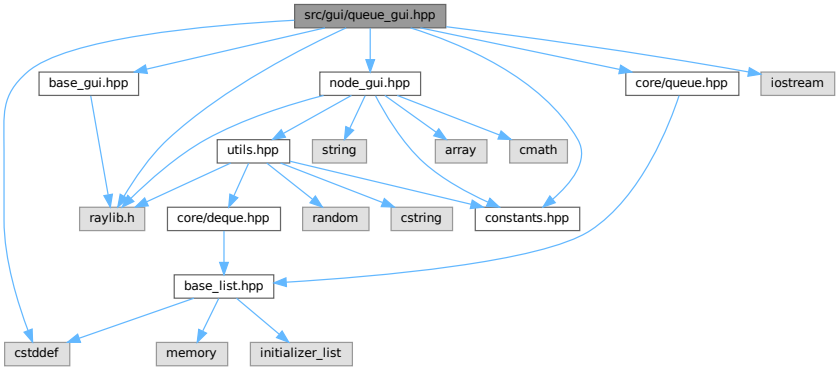
```
00009 #include "raylib.h"
00010 #include "utils.hpp"
00011
00012 namespace gui {
00013
00014 template<typename T>
00015 class GuiNode {
00016 private:
00017     T m_value{};
00018     Color m_color{BLACK};
00019
00020     Vector2 m_current_pos{constants::sidebar_width +
00021                           static_cast<float>(constants::scene_width -
00022                                              constants::sidebar_width) /
00023                               2,
00024                           0};
00025     Vector2 m_target_pos{};
00026     bool m_is_outdated{false};
00027     static constexpr float eps = 1e-3;
00028
00029 public:
00030     static constexpr int radius = 20;
00031
00032     explicit GuiNode(const T& value);
00033
00034     void render();
00035     void set_target_pos(Vector2 pos);
00036     [[nodiscard]] Vector2 get_target_pos() const;
00037     [[nodiscard]] Vector2 get_current_pos() const;
00038     [[nodiscard]] bool check_outdated() const;
00039     void set_color(Color color);
00040     void set_value(const T& value);
00041     T& get_value();
00042 };
00043
00044 template<typename T>
00045 GuiNode<T>::GuiNode(const T& value) : m_value{value} {}
00046
00047 template<typename T>
00048 void GuiNode<T>::render() {
00049     // if (m_is_outdated) {
00050     //     float diff_x = m_target_pos.x - m_current_pos.x;
00051     //     float diff_y = m_target_pos.y - m_current_pos.y;
00052
00053     //     if (std::fabs(diff_x) < eps) {
00054     //         diff_x = 0;
00055     //     }
00056
00057     //     if (std::fabs(diff_y) < eps) {
00058     //         diff_y = 0;
00059     //     }
00060
00061     //     if (diff_x == 0 && diff_y == 0) {
00062     //         m_is_outdated = false;
00063     //     } else {
00064     //         m_current_pos.x +=
00065     //             diff_x / constants::frames_per_second * constants::ani_speed;
00066     //         m_current_pos.y +=
00067     //             diff_y / constants::frames_per_second * constants::ani_speed;
00068     //     }
00069     // }
00070
00071     constexpr int label_font_size = 25;
00072     constexpr int label_font_spacing = 2;
00073     const std::string label = std::to_string(m_value);
00074
00075     const Vector2 label_size =
00076         utils::MeasureText(label.c_str(), label_font_size, label_font_spacing);
00077
00078     const Vector2 label_pos{m_current_pos.x - label_size.x / 2,
00079                             m_current_pos.y - label_size.y / 2};
00080
00081     DrawCircleV(m_current_pos, radius, m_color);
00082     utils::DrawText(label.c_str(), label_pos, WHITE, label_font_size,
00083                     label_font_spacing);
00084 }
00085
00086 template<typename T>
00087 void GuiNode<T>::set_color(Color color) {
00088     m_color = color;
00089 }
00090
00091 template<typename T>
00092 void GuiNode<T>::set_value(const T& value) {
00093     m_value = value;
00094 }
00095
```

```
00096 template<typename T>
00097 T& GuiNode<T>::get_value() {
00098     return m_value;
00099 }
00100
00101 template<typename T>
00102 void GuiNode<T>::set_target_pos(Vector2 pos) {
00103     // m_target_pos = pos;
00104     // m_is_outdated = true;
00105     m_current_pos = pos;
00106 }
00107
00108 template<typename T>
00109 Vector2 GuiNode<T>::get_target_pos() const {
00110     return m_target_pos;
00111 }
00112
00113 template<typename T>
00114 Vector2 GuiNode<T>::get_current_pos() const {
00115     return m_current_pos;
00116 }
00117
00118 template<typename T>
00119 bool GuiNode<T>::check_outdated() const {
00120     return m_is_outdated;
00121 }
00122
00123 }  // namespace gui
00124
00125 #endif  // GUI_NODE_GUI_HPP_
```

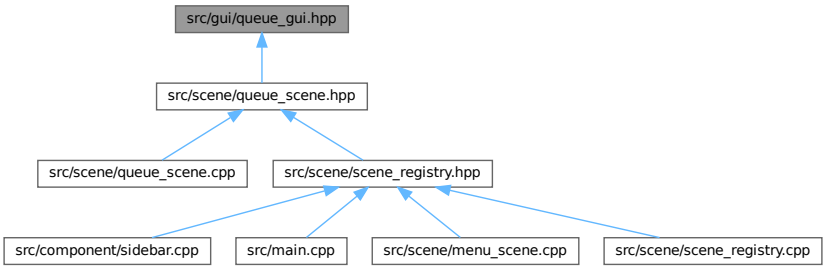
7.55 src/gui/queue_gui.hpp File Reference

```
#include <cstddef>
#include <iostream>
#include "base_gui.hpp"
#include "constants.hpp"
#include "core/queue.hpp"
#include "node_gui.hpp"
#include "raylib.h"
```

Include 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

7.56 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
00013 namespace gui {
00014
00015 template<typename T>
00016 class GuiQueue : public core::Queue<GuiNode<T>, public internal::Base {
00017 private:
00018     using Base = core::Queue<GuiNode<T>>;
00019
00020     static constexpr Vector2 head_pos{
00021         constants::sidebar_width +
00022             (constants::scene_width - constants::sidebar_width) / 2.0F -
00023             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     void push(const T& elem);
00038     void pop();
00039
00040     // for animation purpose only, not for real use
```

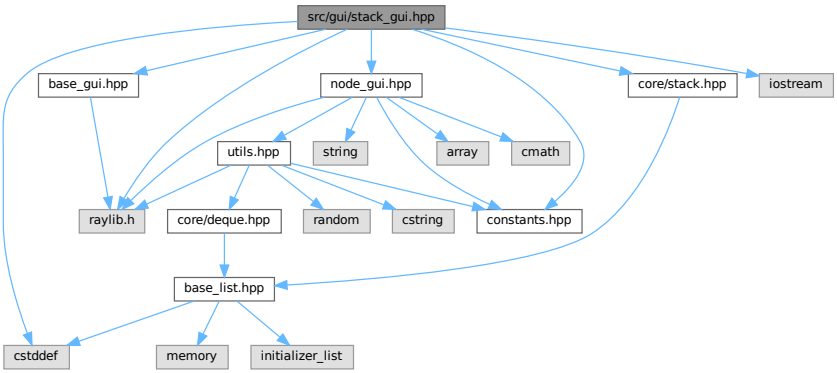
```
00041     void push_front(const T& elem);
00042     void pop_back();
00043
00044     void update() override;
00045     void render() override;
00046 };
00047
00048 template<typename T>
00049 void GuiQueue<T>::push(const T& elem) {
00050     Base::push(GuiNode<T>{elem});
00051 }
00052
00053 template<typename T>
00054 void GuiQueue<T>::pop() {
00055     Base::pop();
00056 }
00057
00058 template<typename T>
00059 void GuiQueue<T>::push_front(const T& elem) {
00060     Base::push_front(GuiNode<T>{elem});
00061 }
00062
00063 template<typename T>
00064 void GuiQueue<T>::pop_back() {
00065     Base::pop_back();
00066 }
00067
00068 template<typename T>
00069 void GuiQueue<T>::render_link(Vector2 src, Vector2 dest) {
00070     constexpr int radius = GuiNode<T>::radius;
00071     constexpr float scaled_len = radius / 8.0F;
00072
00073     // straight line
00074     Vector2 link_pos{src.x + radius, src.y - scaled_len};
00075     Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};
00076
00077     // arrow
00078     constexpr int arrow_size = scaled_len * 5;
00079     Vector2 head{dest.x - radius + scaled_len / 2, src.y};
00080     Vector2 side_top{head.x - arrow_size, head.y - arrow_size};
00081     Vector2 side_bot{head.x - arrow_size, head.y + arrow_size};
00082
00083     // draw both
00084     DrawRectangleV(link_pos, link_size, GRAY);
00085     DrawTriangle(head, side_top, side_bot, GRAY);
00086 }
00087
00088 template<typename T>
00089 void GuiQueue<T>::render() {
00090     update();
00091
00092     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00093         if (ptr->next != nullptr) {
00094             render_link(ptr->data.get_current_pos(),
00095                         ptr->next->data.get_current_pos());
00096         }
00097
00098         ptr->data.render();
00099     }
00100 }
00101
00102 template<typename T>
00103 void GuiQueue<T>::update() {
00104     // TODO: if not outdated then return
00105
00106     std::size_t pos = 0;
00107
00108     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00109         ptr->data.set_target_pos(
00110             {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
00111         ++pos;
00112     }
00113 }
00114
00115 }  // namespace gui
00116
00117 #endif  // GUI_QUEUE_GUI_HPP_
```

7.57 src/gui/stack_gui.hpp File Reference

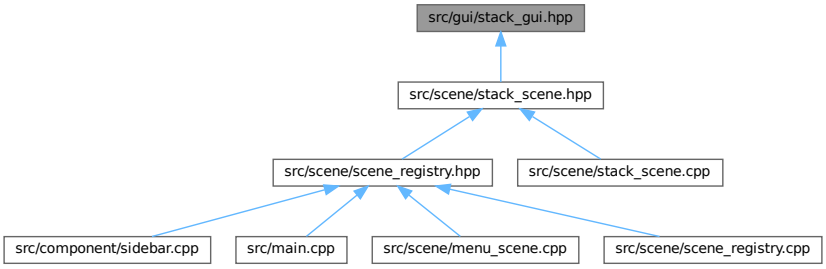
```
#include <cstddef>
#include <iostream>
```

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

Include 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

7.58 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
00013 namespace gui {
00014
00015 template<typename T>
00016 class GuiStack : public core::Stack<GuiNode<T», public internal::Base {
00017 private:
00018     using Base = core::Stack<GuiNode<T>>;
00019
00020     static constexpr Vector2 head_pos{
00021         constants::sidebar_width +
00022             (constants::scene_width - constants::sidebar_width) / 2.0F -
00023             GuiNode<T>::radius / 2.0F,
00024         GuiNode<T>::radius * 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     void push(const T& elem);
00038     void pop();
00039
00040     void update() override;
00041     void render() override;
00042 };
00043
00044 template<typename T>
00045 void GuiStack<T>::push(const T& elem) {
00046     Base::push(GuiNode<T>{elem});
00047 }
00048
00049 template<typename T>
00050 void GuiStack<T>::pop() {
00051     Base::pop();
00052 }
00053
00054 template<typename T>
00055 void GuiStack<T>::render_link(Vector2 src, Vector2 dest) {
00056     constexpr int radius = GuiNode<T>::radius;
00057     constexpr float scaled_len = radius / 8.0F;
00058
00059     // straight line
00060     Vector2 link_pos{src.x - scaled_len, src.y + radius};
00061     Vector2 link_size{2 * scaled_len, dest.y - src.y - 2 * radius};
00062
00063     // arrow
00064     constexpr int arrow_size = scaled_len * 5;
00065     Vector2 head{src.x, dest.y - radius + scaled_len / 2};
00066     Vector2 side_left{head.x - arrow_size, head.y - arrow_size};
00067     Vector2 side_right{head.x + arrow_size, head.y - arrow_size};
00068
00069     // draw both
00070     DrawRectangleV(link_pos, link_size, GRAY);
00071     DrawTriangle(head, side_right, side_left, GRAY);
00072 }
00073
00074 template<typename T>
00075 void GuiStack<T>::render() {
00076     update();
00077
00078     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00079         if (ptr->next != nullptr) {
00080             render_link(ptr->data.get_current_pos(),
00081                         ptr->next->data.get_current_pos());
00082         }
00083
00084         ptr->data.render();
00085     }
00086 }
00087
00088 template<typename T>
00089 void GuiStack<T>::update() {
00090     // TODO: if not outdated then return
```

```
00091
00092     std::size_t pos = 0;
00093
00094     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00095         ptr->data.set_target_pos(
00096             {head_pos.x, head_pos.y + 4 * GuiNode<T>::radius * pos});
00097         ++pos;
00098     }
00099 }
00100
00101 }  // namespace gui
00102
00103 #endif  // GUI_STACK_GUI_HPP_
```

7.59 src/main.cpp File Reference

```
#include <iostream>
#include "component/sidebar.hpp"
#include "constants.hpp"
#include "raygui.h"
#include "scene/scene_registry.hpp"
```

Include dependency graph for main.cpp:

Functions

- int main ()

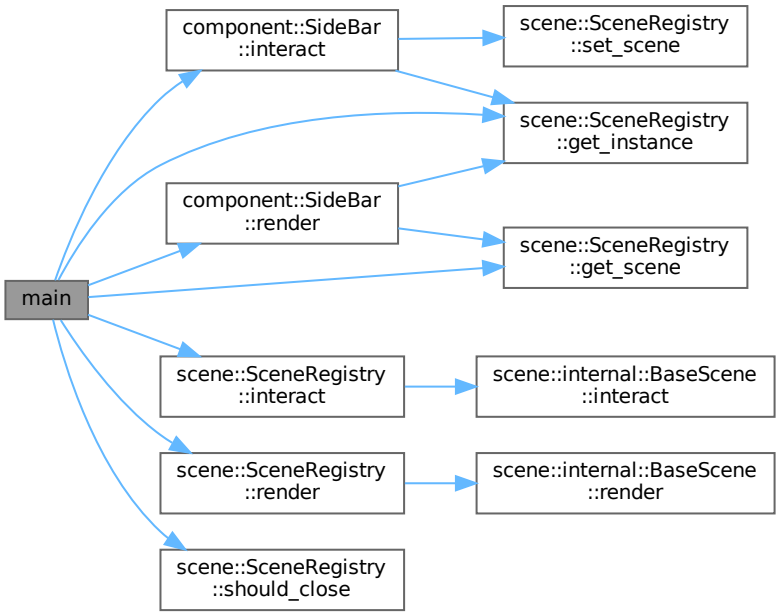
7.59.1 Function Documentation

7.59.1.1 main()

```
int main ( )
```

Definition at line 8 of file main.cpp.

Here is the call graph for this function:



7.60 main.cpp

Go to the documentation of this file.

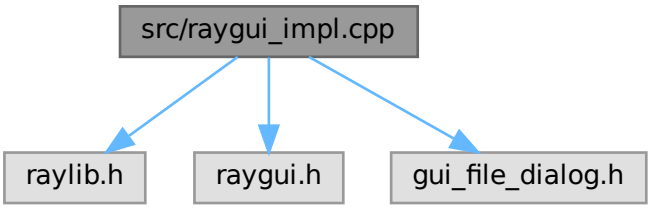
```
00001 #include <iostream>
00002
00003 #include "component/sidebar.hpp"
00004 #include "constants.hpp"
00005 #include "raygui.h"
00006 #include "scene/scene_registry.hpp"
00007
00008 int main() {
00009     InitWindow(constants::scene_width, constants::scene_height,
00010                "VisuAlgo.net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

```
00038     } while (!should_close);
00039
00040     CloseWindow();
00041
00042     return 0;
00043 }
```

7.61 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

7.61.1 Macro Definition Documentation

7.61.1.1 GUI_FILE_DIALOG_IMPLEMENTATION

```
#define GUI_FILE_DIALOG_IMPLEMENTATION
```

Definition at line 6 of file raygui_impl.cpp.

7.61.1.2 RAYGUI_IMPLEMENTATION

```
#define RAYGUI_IMPLEMENTATION
```

Definition at line 2 of file raygui_impl.cpp.

7.62 raygui_impl.cpp

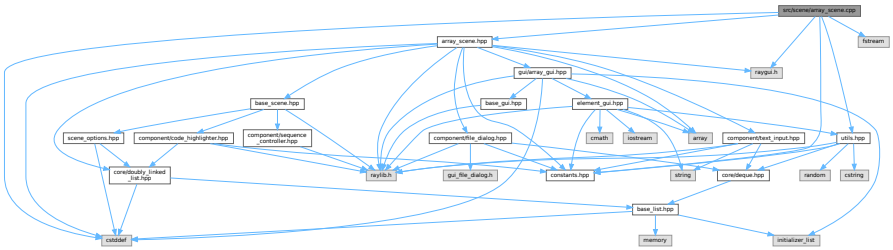
Go to the documentation of this file.

```
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"
```

7.63 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

7.64 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::get_instance() {
00018     static ArrayScene scene;
00019     return scene;
00020 }
00021
```

```
00022 void ArrayScene::render_inputs() {
00023     int& mode = scene_options.mode_selection;
00024 
00025     switch (mode) {
00026         case 0: {
00027             switch (scene_options.action_selection.at(mode)) {
00028                 case 0:
00029                     break;
00030                 case 1: {
00031                     m_text_input.render(options_head, head_offset);
00032                 } break;
00033                 case 2: {
00034                     m_file_dialog.render(options_head, head_offset);
00035                 } break;
00036                 default:
00037                     utils::unreachable();
00038             }
00039         } break;
00040 
00041         case 1: {
00042             m_index_input.render(options_head, head_offset);
00043             m_text_input.render(options_head, head_offset);
00044         } break;
00045 
00046         case 2: {
00047             m_text_input.render(options_head, head_offset);
00048         } break;
00049 
00050         default:
00051             utils::unreachable();
00052     }
00053 
00054     m_go |= render_go_button();
00055 }
00056 
00057 void ArrayScene::render() {
00058     m_sequence_controller.inc_anim_counter();
00059 
00060     int frame_idx = m_sequence_controller.get_anim_frame();
00061     auto* const frame_ptr = m_sequence.find(frame_idx);
00062     m_sequence_controller.set_progress_value(frame_idx);
00063 
00064     if (frame_ptr != nullptr) {
00065         frame_ptr->data.render();
00066         m_code_highlighter.highlight(frame_idx);
00067     } else {  // end of sequence
00068         m_array.render();
00069         m_sequence_controller.set_run_all(false);
00070     }
00071 
00072     m_code_highlighter.render();
00073     m_sequence_controller.render();
00074     render_options(scene_options);
00075 }
00076 
00077 void ArrayScene::interact() {
00078     if (m_sequence_controller.interact()) {
00079         m_sequence_controller.reset_anim_counter();
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 
```

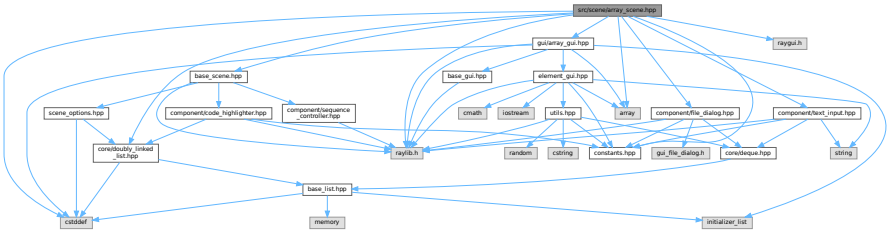
```
00109         case 1: {
00110             interact_update();
00111         } break;
00112
00113         case 2: {
00114             interact_search();
00115         } break;
00116
00117         default:
00118             utils::unreachable();
00119     }
00120
00121     m_go = false;
00122 }
00123
00124 void ArrayScene::interact_random() {
00125     m_array = {};
00126
00127     for (std::size_t i = 0; i < max_size; ++i) {
00128         m_array[i] = utils::get_random(constants::min_val, constants::max_val);
00129     }
00130 }
00131
00132 void ArrayScene::interact_import(core::Deque<int> nums) {
00133     m_array = {};
00134     std::size_t i;  // NOLINT
00135
00136     for (i = 0; i < max_size && !nums.empty(); ++i) {
00137         m_array[i] = nums.front();
00138         nums.pop_front();
00139     }
00140
00141     for (; i < max_size; ++i) {
00142         m_array[i] = 0;
00143     }
00144 }
00145
00146 void ArrayScene::interact_update() {
00147     int index = m_index_input.extract_values().front();
00148     int value = m_text_input.extract_values().front();
00149
00150     if (!(0 <= index && index < max_size) || !utils::val_in_range(value)) {
00151         return;
00152     }
00153
00154     m_code_highlighter.set_code({
00155         "a[i] = val;",
00156     });
00157
00158     m_sequence.clear();
00159
00160     // initial state (before update)
00161     m_sequence.insert(m_sequence.size(), m_array);
00162     m_code_highlighter.push_into_sequence(-1);
00163
00164     // highlight
00165     m_array.set_color(index, ORANGE);
00166     m_sequence.insert(m_sequence.size(), m_array);
00167     m_code_highlighter.push_into_sequence(0);
00168
00169     // update
00170     m_array[index] = value;
00171     m_array.set_color(index, GREEN);
00172     m_sequence.insert(m_sequence.size(), m_array);
00173     m_code_highlighter.push_into_sequence(0);
00174
00175     // undo highlight
00176     m_array.set_color(index, BLACK);
00177
00178     m_sequence_controller.set_max_value((int)m_sequence.size());
00179     m_sequence_controller.set_rerun();
00180 }
00181
00182 void ArrayScene::interact_file_import() {
00183     if (!m_file_dialog.is_pressed()) {
00184         return;
00185     }
00186
00187     interact_import(m_file_dialog.extract_values());
00188
00189     m_file_dialog.reset_pressed();
00190 }
00191
00192 void ArrayScene::interact_search() {
00193     int value = m_text_input.extract_values().front();
00194     if (!utils::val_in_range(value)) {
00195         return;
```

```
00196     }
00197
00198     m_code_highlighter.set_code({
00199         "for (i = 0; i < size; i++)",
00200         "    if (a[i] == val)",
00201         "        return i;",
00202         "return not_found",
00203     });
00204
00205     m_sequence.clear();
00206     m_sequence.insert(m_sequence.size(), m_array);
00207     m_code_highlighter.push_into_sequence(0);
00208
00209     bool found = false;
00210
00211     for (std::size_t i = 0; i < max_size; ++i) {
00212         m_array.set_color(i, ORANGE);
00213         m_sequence.insert(m_sequence.size(), m_array);
00214         m_code_highlighter.push_into_sequence(1);
00215
00216         if (m_array[i] == value) {
00217             found = true;
00218             m_array.set_color(i, GREEN);
00219             m_sequence.insert(m_sequence.size(), m_array);
00220             m_code_highlighter.push_into_sequence(2);
00221             m_array.set_color(i, BLACK);
00222             break;
00223         }
00224
00225         m_array.set_color(i, BLACK);
00226         m_sequence.insert(m_sequence.size(), m_array);
00227         m_code_highlighter.push_into_sequence(0);
00228     }
00229
00230     if (!found) {
00231         m_sequence.insert(m_sequence.size(), m_array);
00232         m_code_highlighter.push_into_sequence(3);
00233     }
00234
00235     m_sequence_controller.set_max_value((int)m_sequence.size());
00236     m_sequence_controller.set_rerun();
00237 }
00238
00239 }  // namespace scene
```

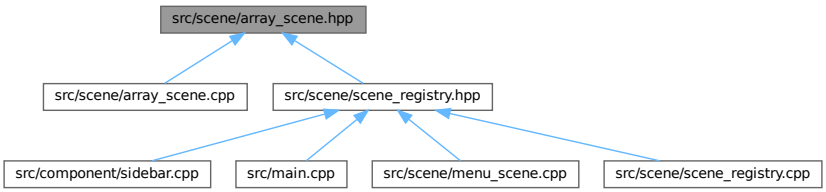
7.65 src/scene/array_scene.hpp File Reference

```
#include <array>
#include <cstddef>
#include "base_scene.hpp"
#include "component/file_dialog.hpp"
#include "component/text_input.hpp"
#include "constants.hpp"
#include "core/doubly_linked_list.hpp"
#include "gui/array_gui.hpp"
#include "raygui.h"
#include "raylib.h"
```

Include dependency graph for array_scene.hpp:



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



Classes

- class scene::ArrayScene

Namespaces

- namespace scene

7.66 array_scene.hpp

Go to the documentation of this file.

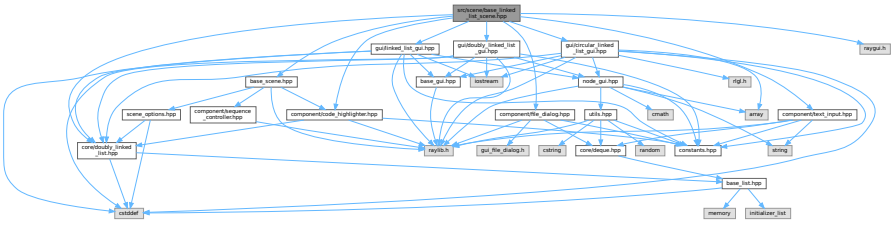
```
00001 #ifndef SCENE_ARRAY_SCENE_HPP_
00002 #define SCENE_ARRAY_SCENE_HPP_
00003
00004 #include <array>
00005 #include <cstddef>
00006
00007 #include "base_scene.hpp"
00008 #include "component/file_dialog.hpp"
00009 #include "component/text_input.hpp"
00010 #include "constants.hpp"
00011 #include "core/doubly_linked_list.hpp"
00012 #include "gui/array_gui.hpp"
00013 #include "raygui.h"
00014 #include "raylib.h"
00015
00016 namespace scene {
00017
00018 class ArrayScene : public internal::BaseScene {
00019 private:
00020     static constexpr std::size_t max_size = 8;
00021
00022     internal::SceneOptions scene_options{
00023         // max_size
00024         max_size,
00025
00026         // mode_labels
00027         "Mode: Create;"
00028         "Mode: Update;"
00029         "Mode: Search",
00030
00031         // mode_selection
00032         0,
00033
00034         // action_labels
00035         {
00036             // Mode: Create
00037             "Action: Random;"
00038             "Action: Input;"
00039             "Action: File",
00040
00041             // Mode: Update
00042             "",
00043
```

```
00044             // Mode: Search
00045             "",
00046         },
00047
00048         // action_selection
00049         core::DoublyLinkedList<int>{0, 0, 0},
00050     };
00051
00052     using internal::BaseScene::button_size;
00053     using internal::BaseScene::head_offset;
00054     using internal::BaseScene::options_head;
00055
00056     gui::GuiArray<int, max_size> m_array{};
00057     core::DoublyLinkedList<gui::GuiArray<int, max_size>> m_sequence;
00058
00059     bool m_go{};
00060     component::TextInput m_text_input;
00061     component::TextInput m_index_input;
00062     component::FileDialog m_file_dialog;
00063     using internal::BaseScene::m_code_highlighter;
00064     using internal::BaseScene::m_sequence_controller;
00065
00066     ArrayScene() = default;
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     void interact_update();
00076     void interact_search();
00077
00078 public:
00079     ArrayScene(const ArrayScene&) = delete;
00080     ArrayScene(ArrayScene&&) = delete;
00081     ArrayScene& operator=(const ArrayScene&) = delete;
00082     ArrayScene& operator=(ArrayScene&&) = delete;
00083     ~ArrayScene() override = default;
00084
00085     static ArrayScene& get_instance();
00086
00087     void render() override;
00088     void interact() override;
00089 };
00090
00091 }  // namespace scene
00092
00093 #endif  // SCENE_ARRAY_SCENE_HPP_
```

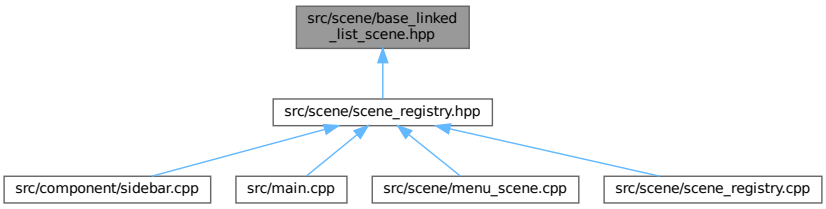
7.67 src/scene/base_linked_list_scene.hpp File Reference

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

Include dependency graph for base_linked_list_scene.hpp:



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



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 > >

7.68 base_linked_list_scene.hpp

Go to the documentation of this file.

```
00001 #ifndef SCENE_BASE_LINKED_LIST_SCENE_HPP_
00002 #define SCENE_BASE_LINKED_LIST_SCENE_HPP_
00003
00004 #include "base_scene.hpp"
00005 #include "component/code_highlighter.hpp"
00006 #include "component/file_dialog.hpp"
00007 #include "component/text_input.hpp"
00008 #include "core/doubly_linked_list.hpp"
00009 #include "gui/circular_linked_list_gui.hpp"
00010 #include "gui/doubly_linked_list_gui.hpp"
00011 #include "gui/linked_list_gui.hpp"
00012 #include "raygui.h"
00013
00014 namespace scene {
00015
00016 template<typename Con>
00017 class BaseLinkedListScene : public internal::BaseScene {
00018 private:
00019     internal::SceneOptions scene_options{
00020         // max_size
00021         8,  // NOLINT
00022
00023         // mode_labels
00024         "Mode: Create;"
00025         "Mode: Add;"
00026         "Mode: Delete;"
00027         "Mode: Update;"
00028         "Mode: Search",
00029
00030         // mode_selection
00031         0,
```

```
00032
00033         // action_labels
00034         {
00035             // Mode: Create
00036             "Action: Random;Action: Input;Action: File",
00037             // Mode: Add
00038             "",
00039             // Mode: Delete
00040             "",
00041             // Mode: Update
00042             "",
00043             // Mode: Search
00044             "",
00045         },
00046
00047         // action_selection
00048         core::DoublyLinkedList<int>{0, 0, 0, 0, 0},
00049     };
00050
00051     using internal::BaseScene::button_size;
00052     using internal::BaseScene::head_offset;
00053     using internal::BaseScene::options_head;
00054
00055     Con m_list{
00056         gui::GuiNode<int>{1},
00057         gui::GuiNode<int>{2},
00058         gui::GuiNode<int>{3},
00059     };
00060     core::DoublyLinkedList<Con> m_sequence;
00061
00062     bool m_go{};
00063     component::TextInput m_text_input;
00064     component::TextInput m_index_input;
00065     component::FileDialog m_file_dialog;
00066     using internal::BaseScene::m_code_highlighter;
00067     using internal::BaseScene::m_sequence_controller;
00068
00069     BaseLinkedListScene() = default;
00070
00071     using internal::BaseScene::render_go_button;
00072     using internal::BaseScene::render_options;
00073     void render_inputs() override;
00074
00075     void interact_random();
00076     void interact_import(core::Deque<int> nums);
00077     void interact_file_import();
00078
00079     void interact_add();
00080     void interact_add_head(int value);
00081     void interact_add_tail(int value);
00082     void interact_add_middle(int index, int value);
00083
00084     void interact_delete();
00085     void interact_delete_head();
00086     void interact_delete_tail();
00087     void interact_delete_middle(int index);
00088
00089     void interact_update();
00090     void interact_search();
00091
00092 public:
00093     BaseLinkedListScene(const BaseLinkedListScene&) = delete;
00094     BaseLinkedListScene(BaseLinkedListScene&&) = delete;
00095     BaseLinkedListScene& operator=(const BaseLinkedListScene&) = delete;
00096     BaseLinkedListScene& operator=(BaseLinkedListScene&&) = delete;
00097     ~BaseLinkedListScene() override = default;
00098
00099     static BaseLinkedListScene& get_instance();
00100
00101     void render() override;
00102     void interact() override;
00103 };
00104
00105 using LinkedListScene = BaseLinkedListScene<gui::GuiLinkedList<int>>;
00106 using DoublyLinkedListScene =
00107     BaseLinkedListScene<gui::GuiDoublyLinkedList<int>>;
00108 using CircularLinkedListScene =
00109     BaseLinkedListScene<gui::GuiCircularLinkedList<int>>;
00110
00111 template<typename Con>
00112 BaseLinkedListScene<Con>& BaseLinkedListScene<Con>::get_instance() {
00113     static BaseLinkedListScene scene;
00114     return scene;
00115 }
00116
00117 template<typename Con>
00118 void BaseLinkedListScene<Con>::render_inputs() {
```

```
00119     int& mode = scene_options.mode_selection;
00120
00121     switch (mode) {
00122         case 0: {
00123             switch (scene_options.action_selection.at(mode)) {
00124                 case 0:
00125                     break;
00126                 case 1: {
00127                     m_text_input.render(options_head, head_offset);
00128                 } break;
00129                 case 2: {
00130                     m_file_dialog.render(options_head, head_offset);
00131                 } break;
00132                 default:
00133                     utils::unreachable();
00134             }
00135         } break;
00136
00137         case 1: {
00138             m_index_input.render(options_head, head_offset);
00139             m_text_input.render(options_head, head_offset);
00140         } break;
00141
00142         case 2: {
00143             m_index_input.render(options_head, head_offset);
00144         } break;
00145
00146         case 3: {
00147             m_index_input.render(options_head, head_offset);
00148             m_text_input.render(options_head, head_offset);
00149         } break;
00150
00151         case 4: {
00152             m_text_input.render(options_head, head_offset);
00153         } break;
00154
00155         default:
00156             utils::unreachable();
00157     }
00158
00159     m_go |= render_go_button();
00160 }
00161
00162 template<typename Con>
00163 void BaseLinkedListScene<Con>::render() {
00164     m_sequence_controller.inc_anim_counter();
00165
00166     int frame_idx = m_sequence_controller.get_anim_frame();
00167     auto* const frame_ptr = m_sequence.find(frame_idx);
00168     m_sequence_controller.set_progress_value(frame_idx);
00169
00170     if (frame_ptr != nullptr) {
00171         frame_ptr->data.render();
00172         m_code_highlighter.highlight(frame_idx);
00173     } else {  // end of sequence
00174         m_list.render();
00175         m_sequence_controller.set_run_all(false);
00176     }
00177
00178     m_code_highlighter.render();
00179     m_sequence_controller.render();
00180     render_options(scene_options);
00181 }
00182
00183 template<typename Con>
00184 void BaseLinkedListScene<Con>::interact() {
00185     if (m_sequence_controller.interact()) {
00186         m_sequence_controller.reset_anim_counter();
00187         return;
00188     }
00189
00190     if (!m_go) {
00191         return;
00192     }
00193
00194     int& mode = scene_options.mode_selection;
00195
00196     switch (mode) {
00197         case 0: {
00198             switch (scene_options.action_selection.at(mode)) {
00199                 case 0: {
00200                     interact_random();
00201                 } break;
00202
00203                 case 1: {
00204                     interact_import(m_text_input.extract_values());
00205                 } break;
```

```
00206
00207                 case 2: {
00208                     interact_file_import();
00209                 } break;
00210
00211                 default:
00212                     utils::unreachable();
00213             }
00214         } break;
00215
00216         case 1: {
00217             interact_add();
00218         } break;
00219
00220         case 2: {
00221             interact_delete();
00222         } break;
00223
00224         case 3: {
00225             interact_update();
00226         } break;
00227
00228         case 4: {
00229             interact_search();
00230         } break;
00231
00232         default:
00233             utils::unreachable();
00234     }
00235
00236     m_go = false;
00237 }
00238
00239 template<typename Con>
00240 void BaseLinkedListScene<Con>::interact_random() {
00241     std::size_t size =
00242         utils::get_random(std::size_t{1}, scene_options.max_size);
00243     m_list = Con();
00244
00245     for (auto i = 0; i < size; ++i) {
00246         m_list.insert(
00247             i, utils::get_random(constants::min_val, constants::max_val));
00248     }
00249 }
00250
00251 template<typename Con>
00252 void BaseLinkedListScene<Con>::interact_import(core::Deque<int> nums) {
00253     m_sequence.clear();
00254     m_list = Con();
00255
00256     while (!nums.empty()) {
00257         if (utils::val_in_range(nums.front())) {
00258             m_list.insert(m_list.size(), nums.front());
00259         }
00260         nums.pop_front();
00261     }
00262 }
00263
00264 template<typename Con>
00265 void BaseLinkedListScene<Con>::interact_file_import() {
00266     if (!m_file_dialog.is_pressed()) {
00267         return;
00268     }
00269
00270     interact_import(m_file_dialog.extract_values());
00271
00272     m_file_dialog.reset_pressed();
00273 }
00274
00275 template<typename Con>
00276 void BaseLinkedListScene<Con>::interact_add() {
00277     int index = m_index_input.extract_values().front();
00278     int value = m_text_input.extract_values().front();
00279
00280     if (!(0 <= index && index <= m_list.size())) {
00281         return;
00282     }
00283
00284     if (!utils::val_in_range(value)) {
00285         return;
00286     }
00287
00288     m_sequence.clear();
00289     m_sequence.insert(m_sequence.size(), m_list);
00290
00291     if (index == 0) {
00292         interact_add_head(value);
```

```
00293     } else if (index == m_list.size()) {
00294         interact_add_tail(value);
00295     } else {
00296         interact_add_middle(index, value);
00297     }
00298
00299     m_sequence_controller.set_max_value((int)m_sequence.size());
00300     m_sequence_controller.set_rerun();
00301 }
00302
00303 template<typename Con>
00304 void BaseLinkedListScene<Con>::interact_add_head(int value) {
00305     m_code_highlighter.set_code({
00306         "Node* node = new Node(value);",
00307         "node->next = head;",
00308         "head = next;",
00309     });
00310     m_code_highlighter.push_into_sequence(-1);
00311
00312     m_list.insert(0, value);
00313
00314     m_list.at(0).set_color(BLUE);
00315     m_sequence.insert(m_sequence.size(), m_list);
00316     m_code_highlighter.push_into_sequence(0);
00317
00318     if (m_list.size() > 1) {
00319         m_list.at(1).set_color(VIOLET);
00320     }
00321
00322     m_sequence.insert(m_sequence.size(), m_list);
00323     m_code_highlighter.push_into_sequence(1);
00324
00325     if (m_list.size() > 1) {
00326         m_list.at(1).set_color(BLACK);
00327     }
00328
00329     m_list.at(0).set_color(VIOLET);
00330     m_sequence.insert(m_sequence.size(), m_list);
00331     m_code_highlighter.push_into_sequence(2);
00332
00333     m_list.at(0).set_color(BLACK);
00334 }
00335
00336 template<typename Con>
00337 void BaseLinkedListScene<Con>::interact_add_tail(int value) {
00338     m_code_highlighter.set_code({
00339         "Node* node = new Node(value);",
00340         "tail->next = node;",
00341         "tail = tail->next;",
00342     });
00343     m_code_highlighter.push_into_sequence(-1);
00344
00345     std::size_t size = m_list.size();
00346
00347     m_list.insert(size, value);
00348     m_list.at(size).set_color(BLUE);
00349     m_sequence.insert(m_sequence.size(), m_list);
00350     m_code_highlighter.push_into_sequence(0);
00351
00352     m_list.at(size - 1).set_color(VIOLET);
00353     m_sequence.insert(m_sequence.size(), m_list);
00354     m_code_highlighter.push_into_sequence(1);
00355
00356     m_list.at(size - 1).set_color(BLACK);
00357     m_list.at(size).set_color(VIOLET);
00358     m_sequence.insert(m_sequence.size(), m_list);
00359     m_code_highlighter.push_into_sequence(2);
00360
00361     m_list.at(size).set_color(BLACK);
00362 }
00363
00364 template<typename Con>
00365 void BaseLinkedListScene<Con>::interact_add_middle(int index, int value) {
00366     m_code_highlighter.set_code({
00367         "Node* pre = head;",
00368         "for (i = 0; i < index - 1; ++i)",
00369         "    pre = pre->next;",
00370         "",
00371         "Node* nxt = pre->next;",
00372         "Node* node = new Node(value);",
00373         "node->next = nxt;",
00374         "pre->next = node;",
00375     });
00376     m_code_highlighter.push_into_sequence(-1);
00377
00378     m_list.at(0).set_color(VIOLET);
00379     m_sequence.insert(m_sequence.size(), m_list);
```

```
00380     m_code_highlighter.push_into_sequence(0);
00381
00382     // search until index - 1
00383     for (int i = 0; i < index - 1; ++i) {
00384         m_list.at(i).set_color(ORANGE);
00385         m_sequence.insert(m_sequence.size(), m_list);
00386         m_code_highlighter.push_into_sequence(1);
00387
00388         m_list.at(i).set_color(BLACK);
00389         m_list.at(i + 1).set_color(ORANGE);
00390         m_sequence.insert(m_sequence.size(), m_list);
00391         m_code_highlighter.push_into_sequence(2);
00392     }
00393
00394     m_sequence.insert(m_sequence.size(), m_list);
00395     m_code_highlighter.push_into_sequence(1);
00396
00397     // reaching index - 1
00398     // cur
00399     m_list.at(index - 1).set_color(ORANGE);
00400     m_sequence.insert(m_sequence.size(), m_list);
00401     m_code_highlighter.push_into_sequence(3);
00402
00403     // cur->next
00404     m_list.at(index).set_color(PINK);
00405     m_sequence.insert(m_sequence.size(), m_list);
00406     m_code_highlighter.push_into_sequence(4);
00407
00408     // insert between cur and cur->next
00409     m_list.insert(index, value);
00410     m_list.at(index).set_color(BLUE);
00411     m_sequence.insert(m_sequence.size(), m_list);
00412     m_code_highlighter.push_into_sequence(5);
00413
00414     m_list.at(index - 1).set_color(ORANGE);
00415     m_list.at(index + 1).set_color(BLACK);
00416     m_sequence.insert(m_sequence.size(), m_list);
00417     m_code_highlighter.push_into_sequence(6);
00418
00419     m_list.at(index - 1).set_color(BLACK);
00420     m_list.at(index + 1).set_color(PINK);
00421     m_sequence.insert(m_sequence.size(), m_list);
00422     m_code_highlighter.push_into_sequence(7);
00423
00424     // done
00425     m_list.at(index - 1).set_color(BLACK);
00426     m_list.at(index).set_color(BLACK);
00427     m_list.at(index + 1).set_color(BLACK);
00428 }
00429
00430 template<typename Con>
00431 void BaseLinkedListScene<Con>::interact_delete() {
00432     if (m_list.empty()) {
00433         return;
00434     }
00435
00436     int index = m_index_input.extract_values().front();
00437
00438     if (!(0 <= index && index < m_list.size())) {
00439         return;
00440     }
00441
00442     m_sequence.clear();
00443     m_sequence.insert(m_sequence.size(), m_list);
00444
00445     if (index == 0) {
00446         interact_delete_head();
00447     } else if (index + 1 == m_list.size()) {
00448         interact_delete_tail();
00449     } else {
00450         interact_delete_middle(index);
00451     }
00452
00453     m_sequence_controller.set_max_value((int)m_sequence.size());
00454     m_sequence_controller.set_rerun();
00455 }
00456
00457 template<typename Con>
00458 void BaseLinkedListScene<Con>::interact_delete_head() {
00459     m_code_highlighter.set_code({
00460         "Node* temp = head;",
00461         "head = head->next;",
00462         "delete temp;",
00463     });
00464     m_code_highlighter.push_into_sequence(-1);
00465
00466     m_list.at(0).set_color(VIOLET);
```

```
00467     m_sequence.insert(m_sequence.size(), m_list);
00468     m_code_highlighter.push_into_sequence(0);
00469
00470     m_list.at(0).set_color(RED);
00471     if (m_list.size() > 1) {
00472         m_list.at(1).set_color(VIOLET);
00473     }
00474     m_sequence.insert(m_sequence.size(), m_list);
00475     m_code_highlighter.push_into_sequence(1);
00476
00477     m_list.remove(0);
00478     m_sequence.insert(m_sequence.size(), m_list);
00479     m_code_highlighter.push_into_sequence(2);
00480
00481     if (m_list.size() > 0) {
00482         m_list.at(0).set_color(BLACK);
00483     }
00484 }
00485
00486 template<typename Con>
00487 void BaseLinkedListScene<Con>::interact_delete_tail() {
00488     m_code_highlighter.set_code({
00489         "Node* pre = head;",
00490         "Node* nxt = pre->next;",
00491         "while (nxt->next != nullptr)",
00492         "    pre = pre->next, nxt = nxt->next;",
00493         "",
00494         "delete nxt;",
00495         "tail = pre;",
00496     });
00497     m_code_highlighter.push_into_sequence(-1);
00498
00499     m_list.at(0).set_color(ORANGE);
00500     m_sequence.insert(m_sequence.size(), m_list);
00501     m_code_highlighter.push_into_sequence(0);
00502
00503     m_list.at(1).set_color(GREEN);
00504     m_sequence.insert(m_sequence.size(), m_list);
00505     m_code_highlighter.push_into_sequence(1);
00506
00507     int idx = 0;
00508     for (; idx + 2 < m_list.size(); ++idx) {
00509         m_sequence.insert(m_sequence.size(), m_list);
00510         m_code_highlighter.push_into_sequence(2);
00511
00512         m_list.at(idx).set_color(BLACK);
00513         m_list.at(idx + 1).set_color(ORANGE);
00514         m_list.at(idx + 2).set_color(GREEN);
00515         m_sequence.insert(m_sequence.size(), m_list);
00516         m_code_highlighter.push_into_sequence(3);
00517     }
00518
00519     m_sequence.insert(m_sequence.size(), m_list);
00520     m_code_highlighter.push_into_sequence(2);
00521
00522     m_list.at(idx).set_color(ORANGE);
00523     m_list.at(idx + 1).set_color(GREEN);
00524     m_sequence.insert(m_sequence.size(), m_list);
00525     m_code_highlighter.push_into_sequence(4);
00526
00527     m_list.remove(idx + 1);
00528     m_sequence.insert(m_sequence.size(), m_list);
00529     m_code_highlighter.push_into_sequence(5);
00530
00531     m_list.at(idx).set_color(VIOLET);
00532     m_sequence.insert(m_sequence.size(), m_list);
00533     m_code_highlighter.push_into_sequence(6);
00534
00535     m_list.at(idx).set_color(BLACK);
00536 }
00537
00538 template<typename Con>
00539 void BaseLinkedListScene<Con>::interact_delete_middle(int index) {
00540     m_code_highlighter.set_code({
00541         "Node* pre = head;",
00542         "for (i = 0; i < index - 1; i++)",
00543         "    pre = pre->next;",
00544         "",
00545         "Node* node = pre->next;",
00546         "Node* nxt = node->next;",
00547         "delete node;",
00548         "pre->next = nxt;",
00549     });
00550     m_code_highlighter.push_into_sequence(-1);
00551
00552     m_list.at(0).set_color(VIOLET);
00553     m_sequence.insert(m_sequence.size(), m_list);
```

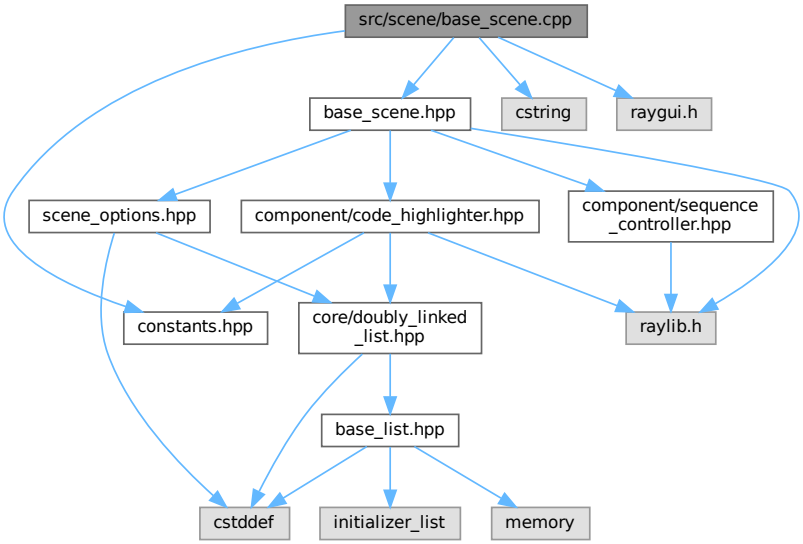
```
00554     m_code_highlighter.push_into_sequence(0);
00555
00556     int idx = 0;
00557     for (; idx + 1 < index; ++idx) {
00558         m_list.at(idx).set_color(ORANGE);
00559         m_sequence.insert(m_sequence.size(), m_list);
00560         m_code_highlighter.push_into_sequence(1);
00561
00562         m_list.at(idx).set_color(BLACK);
00563         m_list.at(idx + 1).set_color(ORANGE);
00564         m_sequence.insert(m_sequence.size(), m_list);
00565         m_code_highlighter.push_into_sequence(2);
00566     }
00567
00568     m_list.at(idx).set_color(ORANGE);
00569     m_sequence.insert(m_sequence.size(), m_list);
00570     m_code_highlighter.push_into_sequence(3);
00571
00572     m_list.at(idx + 1).set_color(RED);
00573     m_sequence.insert(m_sequence.size(), m_list);
00574     m_code_highlighter.push_into_sequence(4);
00575
00576     m_list.at(idx + 2).set_color(GREEN);
00577     m_sequence.insert(m_sequence.size(), m_list);
00578     m_code_highlighter.push_into_sequence(5);
00579
00580     m_list.remove(idx + 1);
00581     m_sequence.insert(m_sequence.size(), m_list);
00582     m_code_highlighter.push_into_sequence(6);
00583
00584     m_list.at(idx + 1).set_color(PINK);
00585     m_sequence.insert(m_sequence.size(), m_list);
00586     m_code_highlighter.push_into_sequence(7);
00587
00588     m_list.at(idx).set_color(BLACK);
00589     m_list.at(idx + 1).set_color(BLACK);
00590 }
00591
00592 template<typename Con>
00593 void BaseLinkedListScene<Con>::interact_update() {
00594     int index = m_index_input.extract_values().front();
00595     int value = m_text_input.extract_values().front();
00596
00597     if (!(0 <= index && index < m_list.size())) {
00598         return;
00599     }
00600
00601     m_code_highlighter.set_code({
00602         "Node* node = head;",
00603         "for (i = 0; i < index; i++)",
00604         "    node = node->next;",
00605         "",
00606         "node->value = value;",
00607     });
00608
00609     m_sequence.clear();
00610     m_sequence.insert(m_sequence.size(), m_list);
00611     m_code_highlighter.push_into_sequence(-1);
00612
00613     m_list.at(0).set_color(VIOLET);
00614     m_sequence.insert(m_sequence.size(), m_list);
00615     m_code_highlighter.push_into_sequence(0);
00616
00617     for (int i = 0; i < index; ++i) {
00618         m_list.at(i).set_color(ORANGE);
00619         m_sequence.insert(m_sequence.size(), m_list);
00620         m_code_highlighter.push_into_sequence(1);
00621
00622         m_list.at(i).set_color(BLACK);
00623         m_list.at(i + 1).set_color(ORANGE);
00624         m_sequence.insert(m_sequence.size(), m_list);
00625         m_code_highlighter.push_into_sequence(2);
00626     }
00627
00628     m_sequence.insert(m_sequence.size(), m_list);
00629     m_code_highlighter.push_into_sequence(1);
00630     m_sequence.insert(m_sequence.size(), m_list);
00631     m_code_highlighter.push_into_sequence(3);
00632
00633     m_list.at(index).set_color(GREEN);
00634     m_list.at(index).set_value(value);
00635     m_sequence.insert(m_sequence.size(), m_list);
00636     m_code_highlighter.push_into_sequence(4);
00637
00638     m_list.at(index).set_color(BLACK);
00639
00640     m_sequence_controller.set_max_value((int)m_sequence.size());
```

```
00641     m_sequence_controller.set_rerun();
00642 }
00643
00644 template<typename Con>
00645 void BaseLinkedListScene<Con>::interact_search() {
00646     int value = m_text_input.extract_values().front();
00647     if (!utils::val_in_range(value)) {
00648         return;
00649     }
00650
00651     m_code_highlighter.set_code({
00652         "Node* node = head;",
00653         "while (node != nullptr) {",
00654         "    if (node->value == value)",
00655         "        return node;",
00656         "    node = node->next;",
00657         "}",
00658         "return not_found",
00659     });
00660
00661     m_sequence.clear();
00662     m_sequence.insert(m_sequence.size(), m_list);
00663     m_code_highlighter.push_into_sequence(-1);
00664
00665     m_list.at(0).set_color(VIOLET);
00666     m_sequence.insert(m_sequence.size(), m_list);
00667     m_code_highlighter.push_into_sequence(0);
00668
00669     std::size_t idx = 0;
00670
00671     while (idx < m_list.size()) {
00672         m_list.at(idx).set_color(ORANGE);
00673         m_sequence.insert(m_sequence.size(), m_list);
00674         m_code_highlighter.push_into_sequence(1);
00675
00676         m_sequence.insert(m_sequence.size(), m_list);
00677         m_code_highlighter.push_into_sequence(2);
00678         if (m_list.at(idx).get_value() == value) {
00679             m_list.at(idx).set_color(GREEN);
00680             m_sequence.insert(m_sequence.size(), m_list);
00681             m_code_highlighter.push_into_sequence(3);
00682             m_list.at(idx).set_color(BLACK);
00683             break;
00684         }
00685
00686         m_list.at(idx).set_color(BLACK);
00687         ++idx;
00688         if (idx < m_list.size()) {
00689             m_list.at(idx).set_color(ORANGE);
00690         }
00691         m_sequence.insert(m_sequence.size(), m_list);
00692         m_code_highlighter.push_into_sequence(4);
00693     }
00694
00695     if (idx >= m_list.size()) {
00696         m_sequence.insert(m_sequence.size(), m_list);
00697         m_code_highlighter.push_into_sequence(1);
00698
00699         m_sequence.insert(m_sequence.size(), m_list);
00700         m_code_highlighter.push_into_sequence(5);
00701
00702         m_sequence.insert(m_sequence.size(), m_list);
00703         m_code_highlighter.push_into_sequence(6);
00704     }
00705
00706     m_sequence_controller.set_max_value((int)m_sequence.size());
00707     m_sequence_controller.set_rerun();
00708 }
00709
00710 }  // namespace scene
00711
00712 #endif  // SCENE_BASE_LINKED_LIST_SCENE_HPP_
```

7.69 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

7.70 base_scene.cpp

Go to the documentation of this file.

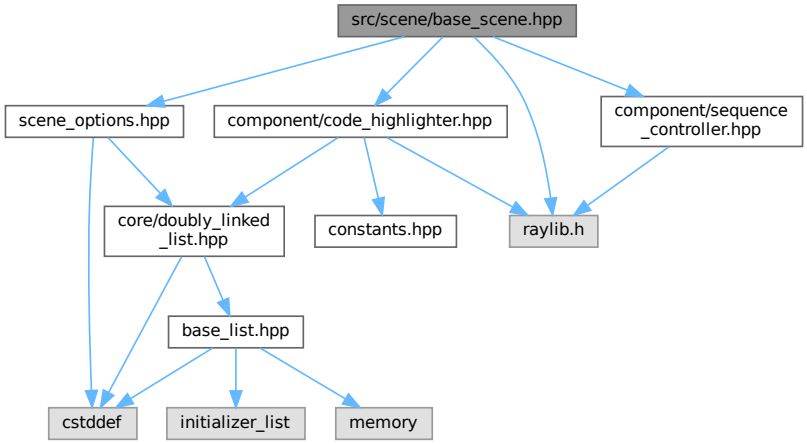
```
00001 #include "base_scene.hpp"
00002
00003 #include <cstring>
00004
00005 #include "constants.hpp"
00006 #include "raygui.h"
00007
00008 namespace scene::internal {
00009
00010 bool BaseScene::render_go_button() const {
00011     Rectangle shape{options_head, constants::scene_height - button_size.y,
00012                     button_size.y, button_size.y};
00013     return GuiButton(shape, "Go");
00014 }
00015
00016 void BaseScene::render_options(SceneOptions& scene_config) {
00017     options_head = 2 * constants::sidebar_width;
00018
00019     Rectangle mode_button_shape{options_head,
00020                                 constants::scene_height - button_size.y,
00021                                 button_size.x, button_size.y};
00022
00023     options_head += (button_size.x + head_offset);
00024
00025     int& mode = scene_config.mode_selection;
00026
00027     mode = GuiComboBox(mode_button_shape, scene_config.mode_labels, mode);
00028
00029     if (std::strlen(scene_config.action_labels.at(mode)) != 0) {
00030         Rectangle action_button_shape{options_head,
```

```
00031                                   constants::scene_height - button_size.y,
00032                                   button_size.x, button_size.y};
00033
00034         options_head += (button_size.x + head_offset);
00035
00036         scene_config.action_selection.at(mode) = GuiComboBox(
00037             action_button_shape, scene_config.action_labels.at(mode),
00038             scene_config.action_selection.at(mode));
00039     }
00040
00041     render_inputs();
00042 }
00043
00044 }  // namespace scene::internal
```

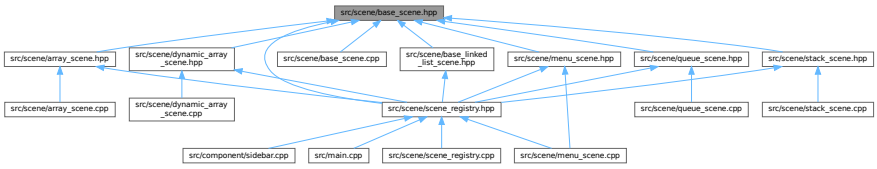
7.71 src/scene/base_scene.hpp File Reference

```
#include "component/code_highlighter.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

7.72 base_scene.hpp

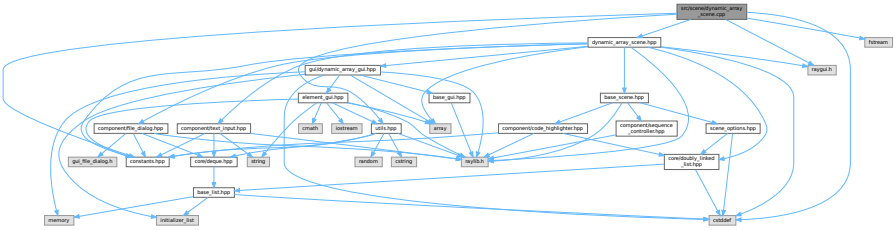
Go to the documentation of this file.

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

7.73 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

7.74 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 DynamicArrayScene& DynamicArrayScene::get_instance() {
00018     static DynamicArrayScene scene;
00019     return scene;
00020 }
00021
00022 void DynamicArrayScene::render_inputs() {
00023     int& mode = scene_options.mode_selection;
00024
00025     switch (mode) {
00026         case 0: {
00027             switch (scene_options.action_selection.at(mode)) {
00028                 case 0:
00029                     break;
00030                 case 1: {
00031                     m_text_input.render(options_head, head_offset);
00032                 } break;
00033                 case 2: {
00034                     m_file_dialog.render(options_head, head_offset);
00035                 } break;
00036                 default:
00037                     utils::unreachable();
00038             }
00039         } break;
00040
00041         case 1: {
00042             m_index_input.render(options_head, head_offset);
00043             m_text_input.render(options_head, head_offset);
00044         } break;
00045
00046         case 2:
00047         case 3: {
00048             m_text_input.render(options_head, head_offset);
00049         } break;
00050
00051         case 4:
00052             break;
00053
00054         default:
00055             utils::unreachable();
00056     }
00057
00058     m_go |= render_go_button();
00059 }
00060
00061 void DynamicArrayScene::render() {
00062     m_sequence_controller.inc_anim_counter();
00063
00064     int frame_idx = m_sequence_controller.get_anim_frame();
00065     auto* const frame_ptr = m_sequence.find(frame_idx);
00066     m_sequence_controller.set_progress_value(frame_idx);
00067
00068     if (frame_ptr != nullptr) {
00069         frame_ptr->data.render();
00070         m_code_highlighter.highlight(frame_idx);
00071     } else {  // end of sequence
00072         m_array.render();
00073         m_sequence_controller.set_run_all(false);
```

```
00074     }
00075
00076     m_code_highlighter.render();
00077     m_sequence_controller.render();
00078     render_options(scene_options);
00079 }
00080
00081 void DynamicArrayScene::interact() {
00082     if (m_sequence_controller.interact()) {
00083         m_sequence_controller.reset_anim_counter();
00084         return;
00085     }
00086
00087     if (!m_go) {
00088         return;
00089     }
00090
00091     int& mode = scene_options.mode_selection;
00092
00093     switch (mode) {
00094         case 0: {
00095             switch (scene_options.action_selection.at(mode)) {
00096                 case 0: {
00097                     interact_random();
00098                 } break;
00099
00100                case 1: {
00101                    interact_import(m_text_input.extract_values());
00102                } break;
00103
00104                case 2: {
00105                    interact_file_import();
00106                } break;
00107
00108                default:
00109                    utils::unreachable();
00110            }
00111        } break;
00112
00113        case 1: {
00114            interact_update();
00115        } break;
00116
00117        case 2: {
00118            interact_search();
00119        } break;
00120
00121        case 3: {
00122            interact_push();
00123        } break;
00124
00125        case 4: {
00126            interact_pop();
00127        } break;
00128
00129        default:
00130            utils::unreachable();
00131    }
00132
00133    m_go = false;
00134 }
00135
00136 void DynamicArrayScene::interact_random() {
00137     std::size_t size =
00138         utils::get_random(std::size_t{1}, scene_options.max_size);
00139     m_array = {};
00140
00141     for (std::size_t i = 0; i < size; ++i) {
00142         m_array.push(utils::get_random(constants::min_val, constants::max_val));
00143     }
00144 }
00145
00146 void DynamicArrayScene::interact_import(core::Deque<int> nums) {
00147     m_array = {};
00148     std::size_t i;  // NOLINT
00149
00150     for (i = 0; i < max_size && !nums.empty(); ++i) {
00151         m_array.push(nums.front());
00152         nums.pop_front();
00153     }
00154 }
00155
00156 void DynamicArrayScene::interact_update() {
00157     int index = m_index_input.extract_values().front();
00158     int value = m_text_input.extract_values().front();
00159
00160     if (!(0 <= index && index < m_array.size()) ||
```

```
00161         !utils::val_in_range(value)) {
00162         return;
00163     }
00164
00165     m_code_highlighter.set_code({
00166         "a[i] = val;",
00167     });
00168
00169     m_sequence.clear();
00170
00171     // initial state (before update)
00172     m_sequence.insert(m_sequence.size(), m_array);
00173     m_code_highlighter.push_into_sequence(-1);
00174
00175     // highlight
00176     m_array.set_color(index, ORANGE);
00177     m_sequence.insert(m_sequence.size(), m_array);
00178     m_code_highlighter.push_into_sequence(0);
00179
00180     // update
00181     m_array[index] = value;
00182     m_array.set_color(index, GREEN);
00183     m_sequence.insert(m_sequence.size(), m_array);
00184     m_code_highlighter.push_into_sequence(0);
00185
00186     // undo highlight
00187     m_array.set_color(index, BLACK);
00188
00189     m_sequence_controller.set_max_value((int)m_sequence.size());
00190     m_sequence_controller.set_rerun();
00191 }
00192
00193 void DynamicArrayScene::interact_file_import() {
00194     if (!m_file_dialog.is_pressed()) {
00195         return;
00196     }
00197
00198     interact_import(m_file_dialog.extract_values());
00199
00200     m_file_dialog.reset_pressed();
00201 }
00202
00203 void DynamicArrayScene::interact_search() {
00204     int value = m_text_input.extract_values().front();
00205     if (!utils::val_in_range(value)) {
00206         return;
00207     }
00208
00209     m_code_highlighter.set_code({
00210         "for (i = 0; i < size; i++)",
00211         "    if (a[i] == val)",
00212         "        return i;",
00213         "return not_found",
00214     });
00215
00216     m_sequence.clear();
00217     m_sequence.insert(m_sequence.size(), m_array);
00218     m_code_highlighter.push_into_sequence(0);
00219
00220     bool found = false;
00221
00222     for (std::size_t i = 0; i < m_array.size(); ++i) {
00223         m_array.set_color(i, ORANGE);
00224         m_sequence.insert(m_sequence.size(), m_array);
00225         m_code_highlighter.push_into_sequence(1);
00226
00227         if (m_array[i] == value) {
00228             found = true;
00229             m_array.set_color(i, GREEN);
00230             m_sequence.insert(m_sequence.size(), m_array);
00231             m_code_highlighter.push_into_sequence(2);
00232             m_array.set_color(i, BLACK);
00233             break;
00234         }
00235
00236         m_array.set_color(i, BLACK);
00237         m_sequence.insert(m_sequence.size(), m_array);
00238         m_code_highlighter.push_into_sequence(0);
00239     }
00240
00241     if (!found) {
00242         m_sequence.insert(m_sequence.size(), m_array);
00243         m_code_highlighter.push_into_sequence(3);
00244     }
00245
00246     m_sequence_controller.set_max_value((int)m_sequence.size());
00247     m_sequence_controller.set_rerun();
```

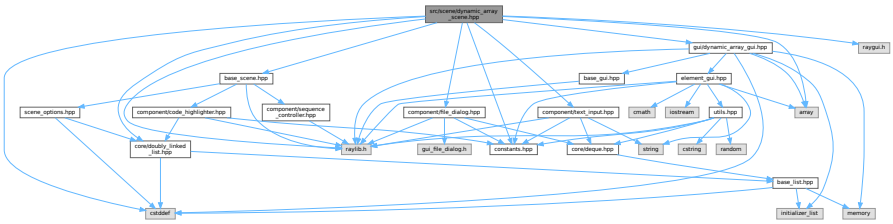
```
00248 }
00249
00250 void DynamicArrayScene::interact_push() {
00251     int value = m_text_input.extract_values().front();
00252
00253     if (m_array.size() >= max_size) {
00254         return;
00255     }
00256
00257     m_code_highlighter.set_code({
00258         "if (size == capacity)",
00259         "    capacity *= 2;",
00260         "array[size] = value;",
00261         "size++;",
00262     });
00263
00264     m_sequence.clear();
00265     m_sequence.insert(m_sequence.size(), m_array);
00266     m_code_highlighter.push_into_sequence(-1);
00267
00268     m_sequence.insert(m_sequence.size(), m_array);
00269     m_code_highlighter.push_into_sequence(0);
00270
00271     if (m_array.size() == m_array.capacity()) {
00272         m_array.realloc(m_array.size() + 1);
00273         m_sequence.insert(m_sequence.size(), m_array);
00274         m_code_highlighter.push_into_sequence(1);
00275     }
00276
00277     m_array.push(value);
00278     m_array.set_color(m_array.size() - 1, GREEN);
00279     m_sequence.insert(m_sequence.size(), m_array);
00280     m_code_highlighter.push_into_sequence(2);
00281
00282     m_array.set_color(m_array.size() - 1, BLACK);
00283     m_sequence.insert(m_sequence.size(), m_array);
00284     m_code_highlighter.push_into_sequence(3);
00285
00286     m_sequence_controller.set_max_value((int)m_sequence.size());
00287     m_sequence_controller.set_rerun();
00288 }
00289
00290 void DynamicArrayScene::interact_pop() {
00291     if (m_array.size() == 0) {
00292         return;
00293     }
00294
00295     m_code_highlighter.set_code({
00296         "array[size - 1] = 0;",
00297         "size--;",
00298     });
00299
00300     m_sequence.clear();
00301     m_sequence.insert(m_sequence.size(), m_array);
00302     m_code_highlighter.push_into_sequence(-1);
00303
00304     m_array.set_color(m_array.size() - 1, ORANGE);
00305     m_sequence.insert(m_sequence.size(), m_array);
00306     m_code_highlighter.push_into_sequence(0);
00307
00308     m_array.pop();
00309     m_sequence.insert(m_sequence.size(), m_array);
00310     m_code_highlighter.push_into_sequence(1);
00311
00312     m_sequence_controller.set_max_value((int)m_sequence.size());
00313     m_sequence_controller.set_rerun();
00314 }
00315
00316 }  // namespace scene
```

7.75 src/scene/dynamic_array_scene.hpp File Reference

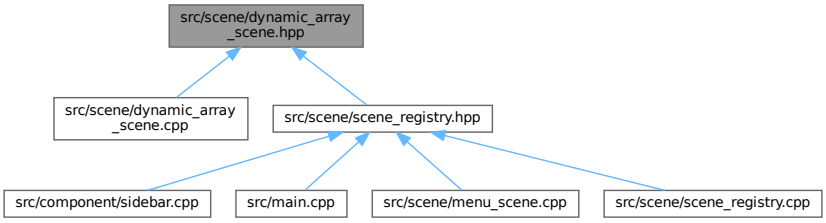
```
#include <array>
#include <cstddef>
#include "base_scene.hpp"
#include "component/file_dialog.hpp"
#include "component/text_input.hpp"
#include "constants.hpp"
```

```
#include "core/doubly_linked_list.hpp"
#include "gui/dynamic_array_gui.hpp"
#include "raygui.h"
#include "raylib.h"
```

Include dependency graph for dynamic_array_scene.hpp:



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



Classes

- class scene::DynamicArrayScene

Namespaces

- namespace scene

7.76 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 "component/text_input.hpp"
00010 #include "constants.hpp"
00011 #include "core/doubly_linked_list.hpp"
00012 #include "gui/dynamic_array_gui.hpp"
00013 #include "raygui.h"
00014 #include "raylib.h"
00015
00016 namespace scene {
```

```
00017
00018 class DynamicArrayScene : public internal::BaseScene {
00019 private:
00020     static constexpr std::size_t max_size = 8;
00021
00022     internal::SceneOptions scene_options{
00023         // max_size
00024         max_size,
00025
00026         // mode_labels
00027         "Mode: Create;"
00028         "Mode: Update;"
00029         "Mode: Search;"
00030         "Mode: Push;"
00031         "Mode: Pop",
00032
00033         // mode_selection
00034         0,
00035
00036         // action_labels
00037         {
00038             // Mode: Create
00039             "Action: Random;Action: Input;Action: File",
00040
00041             // Mode: Update
00042             "",
00043
00044             // Mode: Search
00045             "",
00046
00047             // Mode: Push
00048             "",
00049
00050             // Mode: Pop
00051             "",
00052         },
00053
00054         // action_selection
00055         core::DoublyLinkedList<int>{0, 0, 0, 0, 0},
00056     };
00057
00058     using internal::BaseScene::button_size;
00059     using internal::BaseScene::head_offset;
00060     using internal::BaseScene::options_head;
00061
00062     gui::GuiDynamicArray<int> m_array{};
00063     core::DoublyLinkedList<gui::GuiDynamicArray<int>> m_sequence;
00064
00065     bool m_go{};
00066     component::TextInput m_text_input;
00067     component::TextInput m_index_input;
00068     component::FileDialog m_file_dialog;
00069     using internal::BaseScene::m_sequence_controller;
00070
00071     DynamicArrayScene() = default;
00072
00073     using internal::BaseScene::render_go_button;
00074     using internal::BaseScene::render_options;
00075     void render_inputs() override;
00076
00077     void interact_random();
00078     void interact_import(core::Deque<int> nums);
00079     void interact_file_import();
00080     void interact_update();
00081     void interact_search();
00082     void interact_push();
00083     void interact_pop();
00084
00085 public:
00086     DynamicArrayScene(const DynamicArrayScene&) = delete;
00087     DynamicArrayScene(DynamicArrayScene&&) = delete;
00088     DynamicArrayScene& operator=(const DynamicArrayScene&) = delete;
00089     DynamicArrayScene& operator=(DynamicArrayScene&&) = delete;
00090     ~DynamicArrayScene() override = default;
00091
00092     static DynamicArrayScene& get_instance();
00093
00094     void render() override;
00095     void interact() override;
00096 };
00097
00098 }  // namespace scene
00099
00100 #endif  // SCENE_DYNAMIC_ARRAY_SCENE_HPP_
```

7.77 src/scene/menu_scene.cpp File Reference

```
#include "menu_scene.hpp"
#include <iostream>
#include "constants.hpp"
#include "raygui.h"
#include "scene_registry.hpp"
#include "utils.hpp"
```

Include dependency graph for menu_scene.cpp:

Namespaces

- namespace scene

7.78 menu_scene.cpp

Go to the documentation of this file.

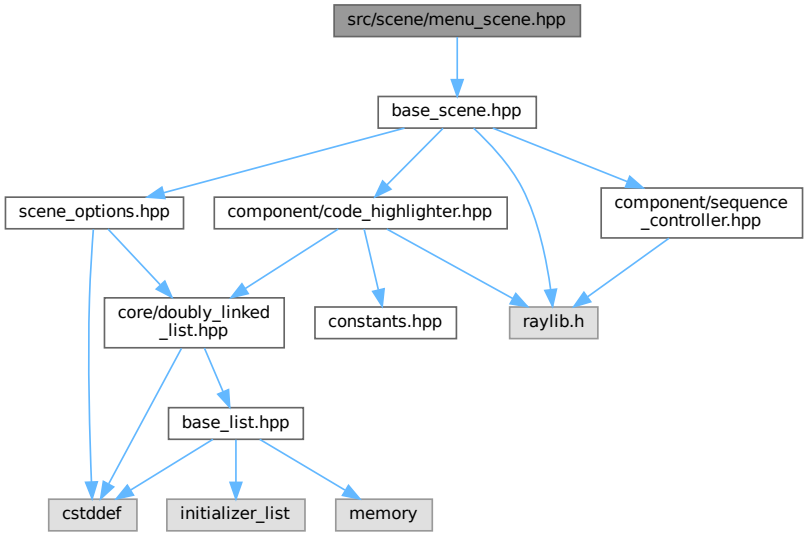
```
00001 #include "menu_scene.hpp"
00002
00003 #include <iostream>
00004
00005 #include "constants.hpp"
00006 #include "raygui.h"
00007 #include "scene_registry.hpp"
00008 #include "utils.hpp"
00009
00010 namespace scene {
00011
00012 MenuScene& MenuScene::get_instance() {
00013     static MenuScene scene;
00014     return scene;
00015 }
00016
00017 void MenuScene::render() {
00018     // Menu text
00019     constexpr int menu_font_size = 60;
00020     constexpr int menu_font_spacing = 5;
00021
00022     constexpr const char* menu_text = "CS162 - VisuAlgo.net clone in C++";
00023
00024     const Vector2 menu_text_size =
00025         utils::MeasureText(menu_text, menu_font_size, menu_font_spacing);
00026
00027     const Vector2 menu_text_pos{
00028         constants::scene_width / 2.0F - menu_text_size.x / 2,
00029         constants::scene_height / 3.0F - menu_text_size.y / 2};
00030
00031     utils::DrawText(menu_text, menu_text_pos, BLACK, menu_font_size,
00032                     menu_font_spacing);
00033
00034     // Sub text
00035     constexpr int sub_font_size = 30;
00036     constexpr int sub_font_spacing = 2;
00037
00038     constexpr const char* sub_text = "By Quang-Truong Nguyen (@jalsol)";
00039
00040     const Vector2 sub_text_size =
00041         utils::MeasureText(sub_text, sub_font_size, sub_font_spacing);
00042
00043     const Vector2 sub_text_pos{
00044         constants::scene_width / 2.0F - sub_text_size.x / 2,
```

```
00045         menu_text_pos.y + menu_text_size.y / 2 + sub_text_size.y};
00046
00047     utils::DrawText(sub_text, sub_text_pos, BLACK, sub_font_size,
00048                     sub_font_spacing);
00049
00050     // Button
00051     constexpr int button_width = 256;
00052     constexpr int button_height = 64;
00053
00054     const Rectangle start_button_shape{
00055         constants::scene_width / 2.0F - button_width / 2.0F,
00056         constants::scene_height / 16.0F * 9 - button_height / 2.0F,
00057         button_width, button_height};
00058
00059     m_start = GuiButton(start_button_shape, "Start");
00060
00061     const Rectangle quit_button_shape{
00062         start_button_shape.x,
00063         constants::scene_height / 16.0F * 11 - button_height / 2.0F,
00064         button_width, button_height};
00065
00066     m_quit = GuiButton(quit_button_shape, "Quit");
00067
00068     // Bottom text
00069     constexpr int bot_font_size = 20;
00070     constexpr int bot_font_spacing = 2;
00071
00072     constexpr const char* bot_text =
00073         "(pls read the src code, i tried so hard for this)";
00074
00075     const Vector2 bot_text_size =
00076         utils::MeasureText(bot_text, bot_font_size, bot_font_spacing);
00077
00078     const Vector2 bot_text_pos{
00079         constants::scene_width / 2.0F - bot_text_size.x / 2,
00080         constants::scene_height - 1.5F * bot_text_size.y};
00081
00082     utils::DrawText(bot_text, bot_text_pos, BLACK, bot_font_size,
00083                     bot_font_spacing);
00084 }
00085
00086 void MenuScene::interact() {
00087     scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();
00088
00089     if (m_start) {
00090         registry.set_scene(Array);
00091         m_start = false;
00092     } else if (m_quit) {
00093         registry.close_window();
00094         m_quit = false;
00095     }
00096 }
00097
00098 }  // namespace scene
```

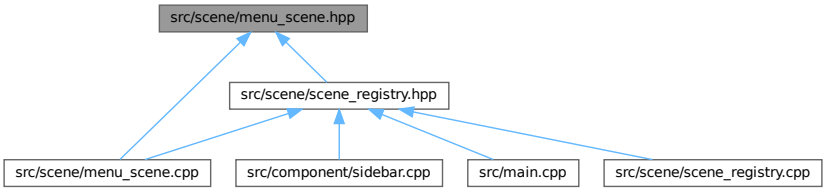
7.79 src/scene/menu_scene.hpp File Reference

```
#include "base_scene.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

7.80 menu_scene.hpp

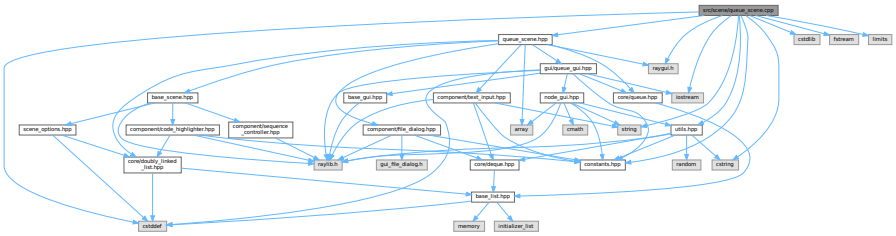
Go to the documentation of this file.

```
00001 #ifndef SCENE_MENU_SCENE_HPP_
00002 #define SCENE_MENU_SCENE_HPP_
00003
00004 #include "base_scene.hpp"
00005
00006 namespace scene {
00007
00008 class MenuScene : public internal::BaseScene {
00009 private:
00010     bool m_start{};
00011     bool m_quit{};
00012
00013     MenuScene() = default;
00014
00015 public:
00016     MenuScene(const MenuScene&) = delete;
00017     MenuScene(MenuScene&&) = delete;
00018     MenuScene& operator=(const MenuScene&) = delete;
00019     MenuScene& operator=(MenuScene&&) = delete;
00020     ~MenuScene() override = default;
00021
00022     static MenuScene& get_instance();
00023
00024     void render() override;
00025     void interact() override;
00026 };
00027
00028 }  // namespace scene
00029
00030 #endif  // SCENE_MENU_SCENE_HPP_
```

7.81 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

7.82 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 QueueScene& QueueScene::get_instance() {
00018     static QueueScene scene;
00019     return scene;
00020 }
00021
00022 void QueueScene::render_inputs() {
00023     int& mode = scene_options.mode_selection;
00024
00025     switch (mode) {
00026         case 0: {
00027             switch (scene_options.action_selection.at(mode)) {
00028                 case 0:
00029                     break;
00030                 case 1: {
00031                     m_text_input.render(options_head, head_offset);
00032                 } break;
00033                 case 2: {
00034                     m_file_dialog.render(options_head, head_offset);
00035                 } break;
00036                 default:
00037                     utils::unreachable();
00038             }
00039         } break;
00040
00041         case 1: {
00042             m_text_input.render(options_head, head_offset);
00043         } break;
00044
00045         case 2:
00046             break;
00047         default:
00048             utils::unreachable();
00049     }
00050
00051     m_go |= render_go_button();
00052 }
00053
00054 void QueueScene::render() {
00055     m_sequence_controller.inc_anim_counter();
00056
00057     int frame_idx = m_sequence_controller.get_anim_frame();
00058     auto* const frame_ptr = m_sequence.find(frame_idx);
00059     m_sequence_controller.set_progress_value(frame_idx);
00060
00061     if (frame_ptr != nullptr) {
00062         frame_ptr->data.render();
00063         m_code_highlighter.highlight(frame_idx);
00064     } else {  // end of sequence
00065         m_queue.render();
00066         m_sequence_controller.set_run_all(false);
00067     }
00068
00069     m_code_highlighter.render();
00070     m_sequence_controller.render();
00071     render_options(scene_options);
00072 }
00073
00074 void QueueScene::interact() {
00075     if (m_sequence_controller.interact()) {
00076         m_sequence_controller.reset_anim_counter();
00077         return;
00078     }
00079
00080     if (!m_go) {
00081         return;
00082     }
```

```
00083
00084     int& mode = scene_options.mode_selection;
00085
00086     switch (mode) {
00087         case 0: {
00088             switch (scene_options.action_selection.at(mode)) {
00089                 case 0: {
00090                     interact_random();
00091                 } break;
00092
00093                 case 1: {
00094                     interact_import(m_text_input.extract_values());
00095                 } break;
00096
00097                 case 2: {
00098                     interact_file_import();
00099                 } break;
00100
00101                 default:
00102                     utils::unreachable();
00103             }
00104         } break;
00105
00106         case 1: {
00107             interact_push();
00108         } break;
00109
00110         case 2: {
00111             interact_pop();
00112         } break;
00113
00114         default:
00115             utils::unreachable();
00116     }
00117
00118     m_go = false;
00119 }
00120
00121 void QueueScene::interact_random() {
00122     std::size_t size =
00123         utils::get_random(std::size_t{1}, scene_options.max_size);
00124     m_queue = gui::GuiQueue<int>();
00125
00126     for (auto i = 0; i < size; ++i) {
00127         m_queue.push(utils::get_random(constants::min_val, constants::max_val));
00128     }
00129 }
00130
00131 void QueueScene::interact_import(core::Deque<int> nums) {
00132     m_sequence.clear();
00133     m_queue = gui::GuiQueue<int>();
00134
00135     while (!nums.empty()) {
00136         if (utils::val_in_range(nums.front())) {
00137             m_queue.push(nums.front());
00138         }
00139         nums.pop_front();
00140     }
00141 }
00142
00143 void QueueScene::interact_file_import() {
00144     if (!m_file_dialog.is_pressed()) {
00145         return;
00146     }
00147
00148     interact_import(m_file_dialog.extract_values());
00149
00150     m_file_dialog.reset_pressed();
00151 }
00152
00153 void QueueScene::interact_push() {
00154     int value = m_text_input.extract_values().front();
00155
00156     if (m_queue.size() >= scene_options.max_size) {
00157         return;
00158     }
00159
00160     m_code_highlighter.set_code({
00161         "Node* node = new Node(value);",
00162         "tail->next = node;",
00163         "tail = tail->next;",
00164     });
00165
00166     m_sequence.clear();
00167     m_sequence.insert(m_sequence.size(), m_queue);
00168     m_code_highlighter.push_into_sequence(-1);
00169
```

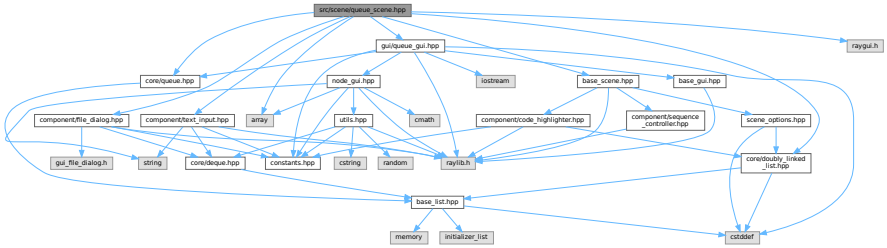
```
00170     m_queue.push(value);
00171     m_queue.back().set_color(BLUE);
00172     m_sequence.insert(m_sequence.size(), m_queue);
00173     m_code_highlighter.push_into_sequence(0);
00174 
00175     m_queue.pop_back();
00176     if (!m_queue.empty()) {
00177         m_queue.back().set_color(VIOLET);
00178     }
00179     m_queue.push(value);
00180     m_queue.back().set_color(BLUE);
00181     m_sequence.insert(m_sequence.size(), m_queue);
00182     m_code_highlighter.push_into_sequence(1);
00183 
00184     m_queue.pop_back();
00185     if (!m_queue.empty()) {
00186         m_queue.back().set_color(BLACK);
00187     }
00188     m_queue.push(value);
00189     m_queue.back().set_color(GREEN);
00190     m_sequence.insert(m_sequence.size(), m_queue);
00191     m_code_highlighter.push_into_sequence(2);
00192 
00193     m_queue.back().set_color(BLACK);
00194 
00195     m_sequence_controller.set_max_value((int)m_sequence.size());
00196     m_sequence_controller.set_rerun();
00197 }
00198 
00199 void QueueScene::interact_pop() {
00200     if (m_queue.empty()) {
00201         return;
00202     }
00203 
00204     m_code_highlighter.set_code({
00205         "Node* temp = head;",
00206         "head = head->next;",
00207         "delete temp;",
00208     });
00209 
00210     m_sequence.clear();
00211     m_sequence.insert(m_sequence.size(), m_queue);
00212     m_code_highlighter.push_into_sequence(-1);
00213 
00214     m_queue.front().set_color(RED);
00215     m_sequence.insert(m_sequence.size(), m_queue);
00216     m_code_highlighter.push_into_sequence(0);
00217 
00218     auto old_front = m_queue.front();
00219     m_queue.pop();
00220 
00221     if (!m_queue.empty()) {
00222         m_queue.front().set_color(GREEN);
00223     }
00224 
00225     m_queue.push_front(old_front.get_value());
00226     m_queue.front().set_color(RED);
00227     m_sequence.insert(m_sequence.size(), m_queue);
00228     m_code_highlighter.push_into_sequence(1);
00229 
00230     m_queue.pop();
00231     m_sequence.insert(m_sequence.size(), m_queue);
00232     m_code_highlighter.push_into_sequence(2);
00233 
00234     if (!m_queue.empty()) {
00235         m_queue.front().set_color(BLACK);
00236     }
00237 
00238     m_sequence_controller.set_max_value((int)m_sequence.size());
00239     m_sequence_controller.set_rerun();
00240 }
00241 
00242 }  // namespace scene
```

7.83 src/scene/queue_scene.hpp File Reference

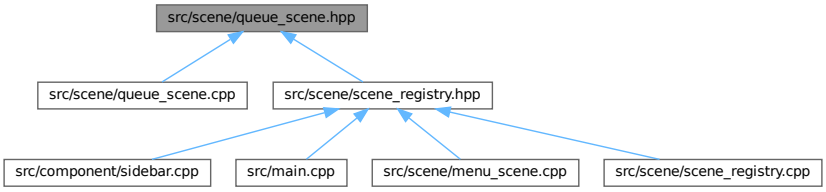
```
#include <array>
#include "base_scene.hpp"
#include "component/file_dialog.hpp"
#include "component/text_input.hpp"
```

```
#include "core/doubly_linked_list.hpp"
#include "core/queue.hpp"
#include "gui/queue_gui.hpp"
#include "raygui.h"
```

Include dependency graph for queue_scene.hpp:



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



Classes

- class scene::QueueScene

Namespaces

- namespace scene

7.84 queue_scene.hpp

Go to the documentation of this file.

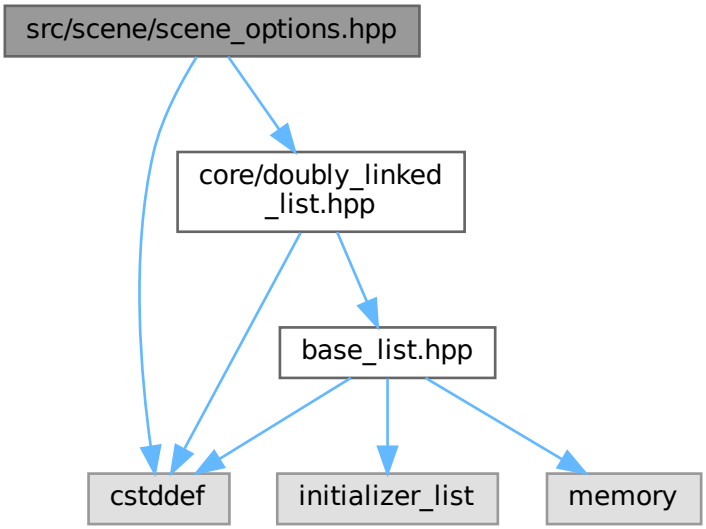
```
00001 #ifndef SCENE_QUEUE_SCENE_HPP_
00002 #define SCENE_QUEUE_SCENE_HPP_
00003
00004 #include <array>
00005
00006 #include "base_scene.hpp"
00007 #include "component/file_dialog.hpp"
00008 #include "component/text_input.hpp"
00009 #include "core/doubly_linked_list.hpp"
00010 #include "core/queue.hpp"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

```
00017 private:
00018     internal::SceneOptions scene_options{
00019         // max_size
00020         8,  // NOLINT
00021
00022         // mode_labels
00023         "Mode: Create;"
00024         "Mode: Push;"
00025         "Mode: Pop",
00026
00027         // mode_selection
00028         0,
00029
00030         // action_labels
00031         {
00032             // Mode: Create
00033             "Action: Random;"
00034             "Action: Input;"
00035             "Action: File",
00036
00037             // Mode: Push
00038             "",
00039
00040             // Mode: Pop
00041             "",
00042         },
00043
00044         // action_selection
00045         core::DoublyLinkedList<int>{0, 0, 0},
00046     };
00047
00048     using internal::BaseScene::button_size;
00049     using internal::BaseScene::head_offset;
00050     using internal::BaseScene::options_head;
00051
00052     gui::GuiQueue<int> m_queue{
00053         gui::GuiNode<int>{1},
00054         gui::GuiNode<int>{2},
00055         gui::GuiNode<int>{3},
00056     };
00057     core::DoublyLinkedList<gui::GuiQueue<int>> m_sequence;
00058
00059     bool m_go{};
00060     component::TextInput m_text_input;
00061     component::FileDialog m_file_dialog;
00062     using internal::BaseScene::m_code_highlighter;
00063     using internal::BaseScene::m_sequence_controller;
00064
00065     QueueScene() = default;
00066
00067     using internal::BaseScene::render_go_button;
00068     using internal::BaseScene::render_options;
00069     void render_inputs() override;
00070
00071     void interact_random();
00072     void interact_import(core::Deque<int> nums);
00073     void interact_file_import();
00074     void interact_push();
00075     void interact_pop();
00076
00077 public:
00078     QueueScene(const QueueScene&) = delete;
00079     QueueScene(QueueScene&&) = delete;
00080     QueueScene& operator=(const QueueScene&) = delete;
00081     QueueScene& operator=(QueueScene&&) = delete;
00082     ~QueueScene() override = default;
00083
00084     static QueueScene& get_instance();
00085
00086     void render() override;
00087     void interact() override;
00088 };
00089
00090 }  // namespace scene
00091
00092 #endif  // SCENE_QUEUE_SCENE_HPP_
```

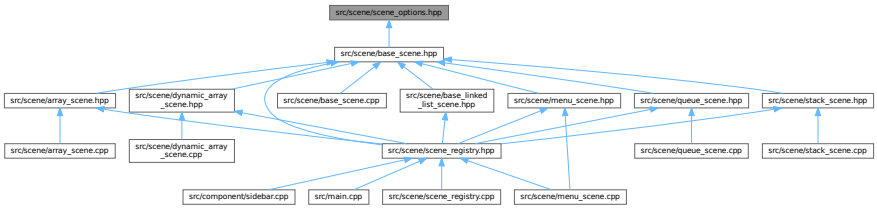
7.85 src/scene/scene_options.hpp File Reference

```
#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

7.86 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_
```

7.87 src/scene/scene_registry.cpp File Reference

`#include "scene_registry.hpp"`

Include dependency graph for scene_registry.cpp:

Namespaces

- namespace scene

7.88 scene_registry.cpp

Go to the documentation of this file.

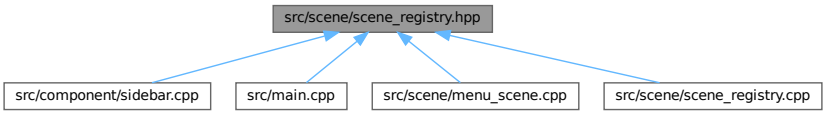
```
00001 #include "scene_registry.hpp"
00002
00003 namespace scene {
00004
00005 SceneRegistry::SceneRegistry() { set_scene(Menu); }
00006
00007 SceneRegistry& SceneRegistry::get_instance() {
00008     static SceneRegistry registry;
00009     return registry;
00010 }
00011
00012 void SceneRegistry::set_scene(int scene_type) {
00013     m_current_scene = scene_type;
00014     scene_ptr = m_registry.at(scene_type);
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
```

7.89 src/scene/scene_registry.hpp File Reference

```
#include <array>
#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 "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::Menu , scene::Array , scene::DynamicArray , scene::LinkedList ,
  scene::DoublyLinkedList , scene::CircularLinkedList , scene::Stack , scene::Queue }

7.90 scene_registry.hpp

Go to the documentation of this file.

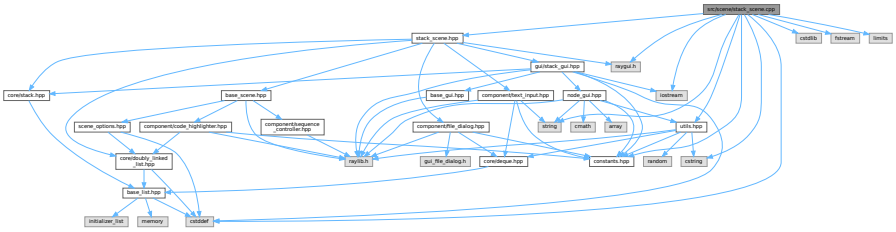
```
00001 #ifndef SCENE_SCENE_REGISTRY_HPP_
00002 #define SCENE_SCENE_REGISTRY_HPP_
00003
00004 #include <array>
00005
00006 #include "array_scene.hpp"
00007 #include "base_linked_list_scene.hpp"
00008 #include "base_scene.hpp"
00009 #include "dynamic_array_scene.hpp"
00010 #include "menu_scene.hpp"
00011 #include "queue_scene.hpp"
00012 #include "stack_scene.hpp"
00013
00014 namespace scene {
00015
00016 enum SceneId {
00017     Menu,
00018     Array,
00019     DynamicArray,
00020     LinkedList,
00021     DoublyLinkedList,
00022     CircularLinkedList,
00023     Stack,
00024     Queue,
00025 };
00026
00027 class SceneRegistry {
00028 private:
00029     internal::BaseScene* scene_ptr{};
00030     SceneRegistry();
00031
00032     bool m_should_close{};
00033     int m_current_scene{};
00034
00035     const std::array<internal::BaseScene* const, 8> m_registry{{
00036         &MenuScene::get_instance(),
00037         &ArrayScene::get_instance(),
00038         &DynamicArrayScene::get_instance(),
00039         &LinkedListScene::get_instance(),
00040         &DoublyLinkedListScene::get_instance(),
00041         &CircularLinkedListScene::get_instance(),
00042         &StackScene::get_instance(),
00043         &QueueScene::get_instance(),
00044     }};
00045
00046 public:
00047     SceneRegistry(const SceneRegistry&) = delete;
00048     SceneRegistry(SceneRegistry&&) = delete;
00049     SceneRegistry& operator=(const SceneRegistry&) = delete;
00050     SceneRegistry& operator=(SceneRegistry&&) = delete;
00051     ~SceneRegistry() = default;
00052
00053     static SceneRegistry& get_instance();
00054
00055     void set_scene(int scene_type);
00056     int get_scene() const;
00057     void render();
00058     void interact();
00059     bool should_close() const;
00060     void close_window();
00061 };
00062
00063 }  // namespace scene
00064
00065 #endif  // SCENE_SCENE_REGISTRY_HPP_
```

7.91 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

7.92 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 StackScene& StackScene::get_instance() {
00018     static StackScene scene;
00019     return scene;
00020 }
00021
00022 void StackScene::render() {
00023     m_sequence_controller.inc_anim_counter();
00024
00025     int frame_idx = m_sequence_controller.get_anim_frame();
00026     auto* const frame_ptr = m_sequence.find(frame_idx);
00027     m_sequence_controller.set_progress_value(frame_idx);
00028
00029     if (frame_ptr != nullptr) {
00030         frame_ptr->data.render();
00031         m_code_highlighter.highlight(frame_idx);
00032     } else {  // end of sequence
00033         m_stack.render();
00034         m_sequence_controller.set_run_all(false);
00035     }
00036
00037     m_code_highlighter.render();
00038     m_sequence_controller.render();
00039     render_options(scene_options);
00040 }
00041
00042 void StackScene::render_inputs() {
00043     int& mode = scene_options.mode_selection;
00044
00045     switch (mode) {
```

```
00046         case 0: {
00047             switch (scene_options.action_selection.at(mode)) {
00048                 case 0:
00049                     break;
00050                 case 1: {
00051                     m_text_input.render(options_head, head_offset);
00052                 } break;
00053                 case 2: {
00054                     m_file_dialog.render(options_head, head_offset);
00055                 } break;
00056                 default:
00057                     utils::unreachable();
00058             }
00059         } break;
00060
00061         case 1: {
00062             m_text_input.render(options_head, head_offset);
00063         } break;
00064
00065         case 2:
00066             break;
00067         default:
00068             utils::unreachable();
00069     }
00070
00071     m_go |= render_go_button();
00072 }
00073
00074 void StackScene::interact() {
00075     if (m_sequence_controller.interact()) {
00076         m_sequence_controller.reset_anim_counter();
00077         return;
00078     }
00079
00080     if (!m_go) {
00081         return;
00082     }
00083
00084     int& mode = scene_options.mode_selection;
00085
00086     switch (mode) {
00087         case 0: {
00088             switch (scene_options.action_selection.at(mode)) {
00089                 case 0: {
00090                     interact_random();
00091                 } break;
00092
00093                 case 1: {
00094                     interact_import(m_text_input.extract_values());
00095                 } break;
00096
00097                 case 2: {
00098                     interact_file_import();
00099                 } break;
00100
00101                 default:
00102                     utils::unreachable();
00103             }
00104         } break;
00105
00106         case 1: {
00107             interact_push();
00108         } break;
00109
00110         case 2: {
00111             interact_pop();
00112         } break;
00113
00114         default:
00115             utils::unreachable();
00116     }
00117
00118     m_go = false;
00119 }
00120
00121 void StackScene::interact_random() {
00122     std::size_t size =
00123         utils::get_random(std::size_t{1}, scene_options.max_size);
00124     m_stack = gui::GuiStack<int>();
00125
00126     for (auto i = 0; i < size; ++i) {
00127         m_stack.push(utils::get_random(constants::min_val, constants::max_val));
00128     }
00129 }
00130
00131 void StackScene::interact_import(core::Deque<int> nums) {
00132     m_sequence.clear();
```

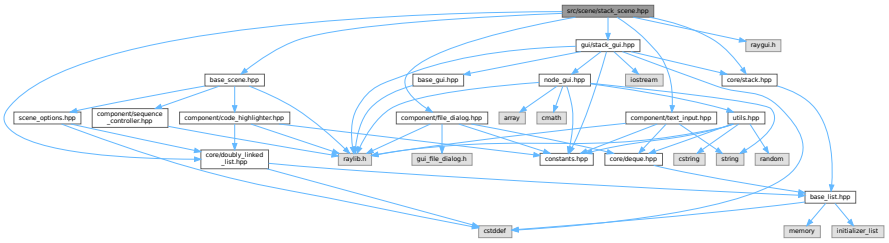
```
00133     m_stack = gui::GuiStack<int>();
00134
00135     while (!nums.empty()) {
00136         if (utils::val_in_range(nums.back())) {
00137             m_stack.push(nums.back());
00138         }
00139         nums.pop_back();
00140     }
00141 }
00142
00143 void StackScene::interact_push() {
00144     int value = m_text_input.extract_values().front();
00145
00146     if (m_stack.size() >= scene_options.max_size) {
00147         return;
00148     }
00149
00150     m_code_highlighter.set_code({
00151         "Node* node = new Node(value);",
00152         "node->next = head;",
00153         "head = node;",
00154     });
00155
00156     m_sequence.clear();
00157     m_sequence.insert(m_sequence.size(), m_stack);
00158     m_code_highlighter.push_into_sequence(-1);
00159
00160     m_stack.push(value);
00161     m_stack.top().set_color(BLUE);
00162     m_sequence.insert(m_sequence.size(), m_stack);
00163     m_code_highlighter.push_into_sequence(0);
00164
00165     m_stack.pop();
00166     if (!m_stack.empty()) {
00167         m_stack.top().set_color(VIOLET);
00168     }
00169     m_stack.push(value);
00170     m_stack.top().set_color(BLUE);
00171     m_sequence.insert(m_sequence.size(), m_stack);
00172     m_code_highlighter.push_into_sequence(1);
00173
00174     m_stack.pop();
00175     if (!m_stack.empty()) {
00176         m_stack.top().set_color(BLACK);
00177     }
00178     m_stack.push(value);
00179     m_stack.top().set_color(GREEN);
00180     m_sequence.insert(m_sequence.size(), m_stack);
00181     m_code_highlighter.push_into_sequence(2);
00182
00183     m_stack.top().set_color(BLACK);
00184
00185     m_sequence_controller.set_max_value((int)m_sequence.size());
00186     m_sequence_controller.set_rerun();
00187 }
00188
00189 void StackScene::interact_pop() {
00190     if (m_stack.empty()) {
00191         return;
00192     }
00193
00194     m_code_highlighter.set_code({
00195         "Node* temp = head;",
00196         "head = head->next;",
00197         "delete temp;",
00198     });
00199
00200     m_sequence.clear();
00201     m_sequence.insert(m_sequence.size(), m_stack);
00202     m_code_highlighter.push_into_sequence(-1);
00203
00204     m_stack.top().set_color(RED);
00205     m_sequence.insert(m_sequence.size(), m_stack);
00206     m_code_highlighter.push_into_sequence(0);
00207
00208     auto old_top = m_stack.top();
00209     m_stack.pop();
00210
00211     if (!m_stack.empty()) {
00212         m_stack.top().set_color(GREEN);
00213     }
00214
00215     m_stack.push(old_top.get_value());
00216     m_stack.top().set_color(RED);
00217     m_sequence.insert(m_sequence.size(), m_stack);
00218     m_code_highlighter.push_into_sequence(1);
00219
```

```
00220     m_stack.pop();
00221     m_sequence.insert(m_sequence.size(), m_stack);
00222     m_code_highlighter.push_into_sequence(2);
00223
00224     if (!m_stack.empty()) {
00225         m_stack.top().set_color(BLACK);
00226     }
00227
00228     m_sequence_controller.set_max_value((int)m_sequence.size());
00229     m_sequence_controller.set_rerun();
00230 }
00231
00232 void StackScene::interact_file_import() {
00233     if (!m_file_dialog.is_pressed()) {
00234         return;
00235     }
00236
00237     interact_import(m_file_dialog.extract_values());
00238
00239     m_file_dialog.reset_pressed();
00240 }
00241
00242 }  // namespace scene
```

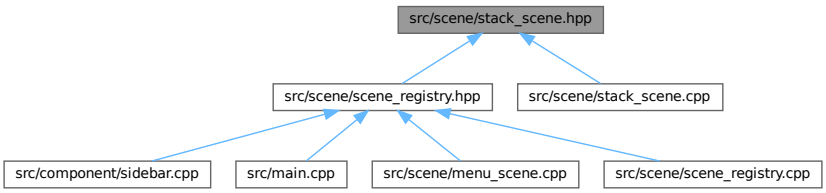
7.93 src/scene/stack_scene.hpp File Reference

```
#include "base_scene.hpp"
#include "component/file_dialog.hpp"
#include "component/text_input.hpp"
#include "core/doubly_linked_list.hpp"
#include "core/stack.hpp"
#include "gui/stack_gui.hpp"
#include "raygui.h"
```

Include dependency graph for stack_scene.hpp:



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



Classes

- class scene::StackScene

Namespaces

- namespace scene

7.94 stack_scene.hpp

Go to the documentation of this file.

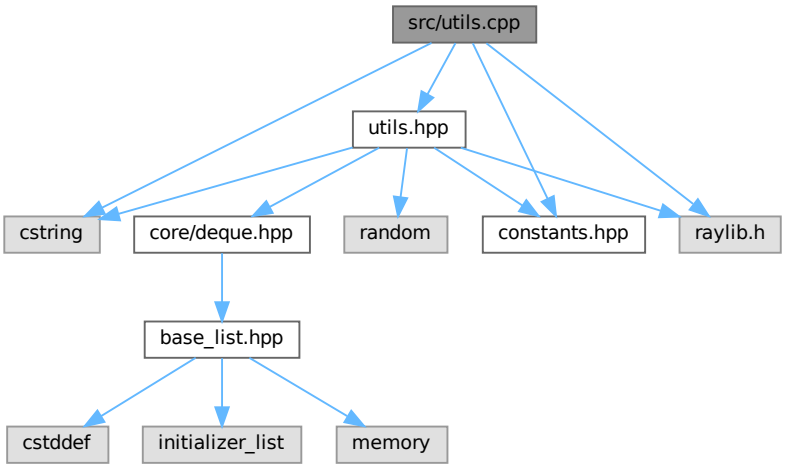
```
00001 #ifndef SCENE_STACK_SCENE_HPP_
00002 #define SCENE_STACK_SCENE_HPP_
00003
00004 #include "base_scene.hpp"
00005 #include "component/file_dialog.hpp"
00006 #include "component/text_input.hpp"
00007 #include "core/doubly_linked_list.hpp"
00008 #include "core/stack.hpp"
00009 #include "gui/stack_gui.hpp"
00010 #include "raygui.h"
00011
00012 namespace scene {
00013
00014 class StackScene : public internal::BaseScene {
00015 private:
00016     internal::SceneOptions scene_options{
00017         // max_size
00018         8,  // NOLINT
00019
00020         // mode_labels
00021         "Mode: Create;"
00022         "Mode: Push;"
00023         "Mode: Pop",
00024
00025         // mode_selection
00026         0,
00027
00028         // action_labels
00029         {
00030             // Mode: Create
00031             "Action: Random;"
00032             "Action: Input;"
00033             "Action: File",
00034
00035             // Mode: Push
00036             "",
00037
00038             // Mode: Pop
00039             "",
00040         },
00041
00042         // action_selection
00043         core::DoublyLinkedList<int>{0, 0, 0},
00044     };
00045
00046     using internal::BaseScene::button_size;
00047     using internal::BaseScene::head_offset;
00048     using internal::BaseScene::options_head;
00049
00050     gui::GuiStack<int> m_stack{
00051         gui::GuiNode<int>{1},
00052         gui::GuiNode<int>{2},
00053         gui::GuiNode<int>{3},
00054     };
00055     core::DoublyLinkedList<gui::GuiStack<int>> m_sequence;
00056
00057     bool m_go{};
00058     component::TextInput m_text_input;
00059     component::FileDialog m_file_dialog;
00060     using internal::BaseScene::m_code_highlighter;
00061     using internal::BaseScene::m_sequence_controller;
00062
00063     StackScene() = default;
00064
00065     using internal::BaseScene::render_go_button;
00066     using internal::BaseScene::render_options;
00067     void render_inputs() override;
00068
00069     void interact_random();
00070     void interact_import(core::Deque<int> nums);
00071     void interact_push();
00072     void interact_pop();
00073     void interact_file_import();
```

```
00074
00075 public:
00076     StackScene(const StackScene&) = delete;
00077     StackScene(StackScene&&) = delete;
00078     StackScene& operator=(const StackScene&) = delete;
00079     StackScene& operator=(StackScene&&) = delete;
00080     ~StackScene() override = default;
00081
00082     static StackScene& get_instance();
00083
00084     void render() override;
00085     void interact() override;
00086 };
00087
00088 }  // namespace scene
00089
00090 #endif  // SCENE_STACK_SCENE_HPP_
```

7.95 src/utils.cpp File Reference

```
#include "utils.hpp"
#include <cstring>
#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)

7.96 utils.cpp

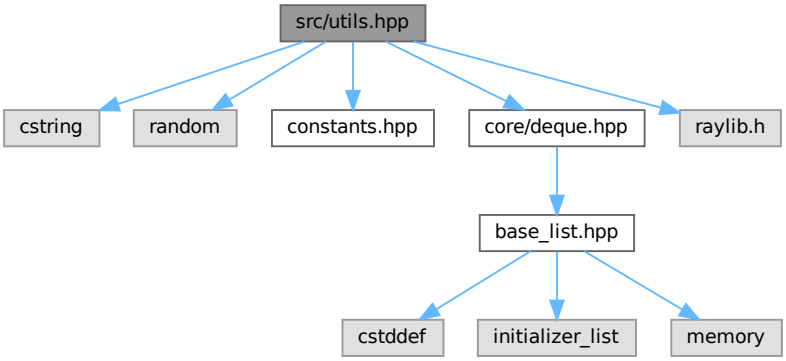
Go to the documentation of this file.

```
00001 #include "utils.hpp"
00002
00003 #include <cstring>
00004
00005 #include "constants.hpp"
00006 #include "raylib.h"
00007
00008 namespace utils {
00009
00010 void DrawText(const char* text, Vector2 pos, Color color, float font_size,
00011               float spacing) {
00012     static Font font = LoadFontEx("data/open_sans.ttf",
00013                                   constants::default_font_size, nullptr, 0);
00014
00015     Vector2 pos_vec{static_cast<float>(pos.x), static_cast<float>(pos.y)};
00016     DrawTextEx(font, text, pos_vec, font_size, spacing, color);
00017 }
00018
00019 Vector2 MeasureText(const char* text, float font_size, float spacing) {
00020     static Font font = LoadFontEx("data/open_sans.ttf",
00021                                   constants::default_font_size, nullptr, 0);
00022
00023     return MeasureTextEx(font, text, font_size, spacing);
00024 }
00025
00026 core::Deque<int> str_extract_data(
00027     char str[constants::text_buffer_size]) {  // NOLINT
00028     char str_copy[constants::text_buffer_size];
00029     strncpy(str_copy, str, constants::text_buffer_size);
00030
00031     char* save_ptr = nullptr;
00032     char* token = utils::strtok(str_copy, ",", &save_ptr);
00033
00034     if (token == nullptr) {
00035         return {0};
00036     }
00037
00038     core::Deque<int> ret;
00039
00040     constexpr int base = 10;
00041     int num = static_cast<int>(std::strtol(token, nullptr, base));
00042     ret.push_back(num);
00043
00044     while (true) {
00045         token = utils::strtok(nullptr, ",", &save_ptr);
00046         if (token == nullptr) {
00047             break;
00048         }
00049
00050         num = static_cast<int>(std::strtol(token, nullptr, base));
00051         ret.push_back(num);
00052     }
00053
00054     return ret;
00055 }
00056
00057 bool val_in_range(int num) {
00058     return constants::min_val <= num && num <= constants::max_val;
00059 }
00060
00061 void unreachable() {
00062 #if defined(_MSC_VER)
00063     __assume(0);
00064 #else
00065     __builtin_unreachable();
00066 #endif
00067 }
00068
00069 char* strtok(char* str, const char* delim, char** save_ptr) {
00070     return
00071 #if defined(_MSC_VER)
00072         strtok_s(str, delim, save_ptr);
00073 #else
00074         strtok_r(str, delim, save_ptr);
00075 #endif
00076 }
00077
00078 }  // namespace utils
```

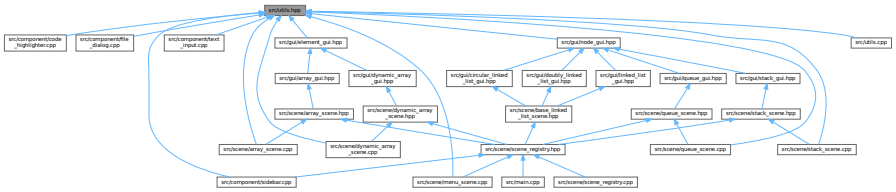
7.97 src/utils.hpp File Reference

```
#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)

7.98 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     static std::random_device ran_dev;
00021     static std::mt19937 prng(ran_dev());
00022     static std::uniform_int_distribution<T> dist{low, high};
00023     return dist(prng);
00024 }
00025
00026 core::Deque<int> str_extract_data(
00027     char str[constants::text_buffer_size]);  // NOLINT
00028
00029 bool val_in_range(int num);
00030
00031 void unreachable();
00032
00033 char* strtok(char* str, const char* delim, char** save_ptr);
00034
00035 }  // namespace utils
00036
00037 #endif  // UTILS_HPP_
```

Index